PONTIFÍCIA UNIVERSIDADE CATÓLICA DE SÃO PAULO

PUC/SP

Custódio Thomaz Kerry Martins

Uma Engenharia Didática para explorar o aspecto de processo dinâmico presente nos algoritmos

DOUTORADO EM EDUCAÇÃO MATEMÁTICA

São Paulo
2010

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE SÃO PAULO

PUC/SP

Custódio Thomaz Kerry Martins

Uma Engenharia Didática para explorar o aspecto de processo dinâmico presente nos algoritmos

Tese apresentada à Banca Examinadora da Pontifícia Universidade Católica de São Paulo, como exigência parcial para obtenção do título de DOUTOR EM EDUCAÇÃO MATEMÀTICA, sob orientação da Prof(a). Dr(a). Barbara Lutaif Bianchini.

São Paulo
2010

Banca Examinadora

_____

_____

_____

_____

_____

Autorizo, exclusivamente para fins acadêmicos e científicos, a reprodução total ou parcial desta Tese por processos de fotocopiadoras ou eletrônicos.

Assinatura: _____ Local e Data: _____

Dedico este trabalho aos meus queridos: Plínio, Fausto e Heloisa.

AGRADECIMENTOS

Aos professores doutores deste programa, que acolheram minha chegada: Professora Sonia Barbosa Camargo Igliori, Professora Janete Bolite Frant, Professora Tânia Maria Campos, Professor Ubiratan D'Ambrosio, Professora Maria Cristina Maranhão, Professor Benedito Antonio da Silva. Ao secretário Francisco Olimpio da Silva, a quem recorri por diversas vezes.

Aos professores e colegas do grupo TecMEM com quem trabalhei a fase inicial do meu projeto, aos professores e colegas do grupo GPEA que me envolveram em várias experiências e discussões, e me animaram na segunda fase deste trabalho.

Agradeço especialmente à minha orientadora Professora Doutora Barbara Lutaif Bianchini, sempre disposta, acessível e empenhada em me atender.

Agradeço ao Centro Universitário da FEI, que me ofereceu preciosos recursos e tempo para o desenvolvimento deste trabalho.

Aos professores doutores que constituíram a Banca de Qualificação, pelas críticas e sugestões valiosas.

Aos professores da FEI e da PUC-SP, com quem trabalho, ou trabalhei, e pude sempre aprender ao longo de minha jornada de professor, especialmente aos professores Dirceu Douglas Salvetti, Milton Rodrigues e Daniel Couto Gatti.

Aos meus alunos, particularmente àqueles que se envolveram na participação deste trabalho.

Aos meus familiares, pelo apoio incondicional e sincero.

RESUMO

A pesquisa foi planejada com o projeto e a realização de uma seqüência de quatro atividades por pequenos grupos de alunos iniciantes em um curso de bacharelado de Ciência da Computação. Essa seqüência de atividades passou por duas experimentações, a primeira no segundo semestre de 2008 e a segunda no segundo semestre de 2009. O foco de interesse é o processo de aprendizagem em disciplinas de Introdução aos Algoritmos e Programação que figuram na grade curricular de cursos superiores como Ciência da Computação e Engenharia. O alvo estabelecido foi investigar como o estudante revela, trata e domina a noção de processo dinâmico inerente a um algoritmo ou a um programa, e o reflexo desse domínio nas atividades de entendimento e elaboração de algoritmos. A teoria dos Registros de Representação Semiótica (Duval, 2008), as idéias da Dialética Ferramenta-Objeto (Maranhão, 2008) e a noção de Engenharia Didática (Artigue, 2009) compõem os aportes teórico-metodológicos principais da pesquisa. Trata-se de uma pesquisa qualitativa, com eixo em uma Engenharia Didática composta por uma seqüência de quatro atividades. O domínio da noção de processo dinâmico, e o emprego dessa noção, foram revelados nas falas, nos gestos e nos registros escritos dos estudantes. As observações e análises permitiram concluir que o trabalho com as atividades favoreceram o entendimento e o domínio da noção de processo dinâmico dos algoritmos, e isso contribuiu para o aprendizado de elaboração e entendimento dos algoritmos e programas.

Palavras-chave: aprendizagem de algoritmos, processo dinâmico, engenharia didática.

ABSTRACT

The research was planned through the design and implementation of a four activities sequence, for small student groups, beginning a Bachelor of Computer Science course. This sequence of activities has been through two trials, the first in the second half of 2008 and second in the second half of 2009. The interest focus is the disciplines learning process in introduction to algorithms and programming, contained in the higher education courses curriculum as Computer Science and Engineering. This research aims to investigate how students produce the perception and the dynamic aspects knowledge, belonging to algorithms, and the reflection of this domain in the algorithms understanding activities and development. The Semiotic Representations Records theory (Duval, 2008), the Tool-Object Dialectic ideas (Maranhão, 2008) and the Didactic Engineering notion (Artigue, 2009), compose the main theoretical-methodological contributions to research. This is a qualitative research, with axis in a Didactic Engineering, composed of a four activities sequence. The domain of the concept of dynamic process, and the use of this notion, were revealed in the speech, gestures and written records of students. The observations and analysis showed that the work with the activities, promoted the understanding and mastery of the algorithms dynamic process notions, which contributed to the development learning of algorithms and programs.

Keywords: algorithms learning, dinamic process, didactic engineering.

LISTA DE QUADROS

LISTA DE FIGURAS

SUMÁRIO

SUMÁRIO

SUMÁRIO

APRESENTAÇÃO

A linha central deste trabalho se concentra nas preocupações com as atividades de aprendizagem que ocorrem em disciplinas de introdução aos algoritmos e lógica de programação, em cursos superiores de Ciências Exatas, especialmente: Ciência da Computação, Engenharia e Sistemas de Informação.

Este trabalho de pesquisa é parte do projeto "*A aprendizagem de álgebra com a utilização de ferramentas tecnológicas*", do Grupo de Pesquisa em Educação Algébrica (GPEA) deste programa de pós-graduação.

Minha vivência como professor e como coordenador de disciplinas de introdução aos algoritmos e linguagens de programação, desde 1980 nos cursos de Engenharia da FEI (antiga Faculdade de Engenharia Industrial, atual Fundação Educacional Inaciana), e depois nos cursos de Ciência da Computação da PUC-SP (1987) e da FEI (1998), sempre foi repleta de preocupações e ações com o sentido de favorecer o processo de aprendizagem dos alunos, diante das dificuldades que tais alunos revelam frequentemente.

Essas preocupações são justificadas em função da experiência como professor, e também de indicações encontradas em alguns trabalhos acadêmicos (Barbosa (2001), Sajaniemi (2005), Bercht, Ferreira e Silveira (2005)), que apontam a necessidade de um cuidado especial na orientação dos processos de construção de conhecimentos, pelos estudantes, para a elaboração de algoritmos e programas de computador.

Por outro lado, algumas diretrizes devem ser consideradas na definição dos planos das disciplinas. A Sociedade Brasileira de Computação – SBC, em

consonância com as Diretrizes Curriculares do Ministério de Educação e Cultura, recomenda como um dos componentes (aspectos técnicos) do perfil dos formandos nos cursos da área de Computação:

> Os egressos de cursos de computação devem ser profissionais com os seguintes conhecimentos técnicos, que podem variar de acordo com as especificidades de cada curso:
> - Processo de projeto para construção de soluções de problemas com base científica;
> - Modelagem e especificação de soluções computacionais para diversos tipos de problemas;
> - Validação da solução de um problema de forma efetiva;
> - Projeto e implementação de sistemas de computação; e
> - Critérios para seleção de *software* e *hardware* adequados às necessidades empresariais, industriais, administrativas, de ensino e de pesquisa. (SOCIEDADE BRASILEIRA DE COMPUTAÇÃO, 2003, p.2)

Mesmo desenvolvidas nas séries iniciais dos cursos de computação, as disciplinas de introdução a algoritmos e programação agregam aos seus objetivos as três primeiras características relacionadas na recomendação.

O trabalho se organiza com o projeto de uma Engenharia Didática, cuja proposta é tratar os aspectos de processo dinâmico presentes nos algoritmos e programas. A exploração dos conceitos envolvidos em uma Engenharia Didática será desenvolvida no capítulo 3. Nas atividades, constituintes da Engenharia Didática, são inseridos aplicativos que ilustram o mecanismo de execução de alguns algoritmos e que são experimentados pelos estudantes no decorrer do desenvolvimento dos trabalhos. Esses aplicativos estão gravados no disco (CD) que acompanha este trabalho.

A primeira meta é verificar se essas atividades favorecem a percepção e o entendimento dos aspectos dinâmicos, representados inicialmente de forma estática, nos textos dos algoritmos; e em conseqüência, favorecem a construção de conhecimentos necessários para as tarefas de elaboração de algoritmos e programas, que são propostas na fase inicial (primeiras seis ou sete semanas) dos cursos dessas disciplinas. Por outro lado, com a observação da atuação dos alunos no desenrolar das atividades, o objetivo é buscar entender como os estudantes revelam e empregam a noção de processo dinâmico, que deve ser vinculada à representação estática dos algoritmos e programas.

O alvo estabelecido é investigar como o estudante revela, trata e domina a noção de processo dinâmico inerente a um algoritmo ou a um programa.

Ao compor a observação de produções dos alunos (gestos, expressões, rascunhos de esboços, registros escritos) com aspectos de fundamentos teóricos, é possível buscar a caracterização de processos de elaboração de conhecimentos pela coordenação entre formas de registros de representação utilizados: as falas dos estudantes, seus registros escritos e a formalização do algoritmo ou do programa. A teoria dos registros de representação semiótica (Duval, 2008) oferece as bases para esse estudo, seus fundamentos serão apresentados no capítulo 3.

A intenção é observar e analisar situações em que os alunos realizem atividades de tratamento de problemas computacionais, e desenvolvimento de algoritmos e programas, e tomar os resultados dessas observações e análises como base para a elaboração de sugestões didáticas que venham a compor meios facilitadores, estratégias e ferramentas, no processo de aprendizagem de desenvolvimento de algoritmos.

Optou-se por conceber uma investigação qualitativa, tendo em vista as finalidades especificadas: buscar entender como os alunos dominam e revelam aquela noção, e verificar se as atividades favorecem a constituição da mesma noção.

Como descreve Machado (2008), uma Engenharia Didática envolve quatro fases principais: análises preliminares, análises *a priori*, experimentação e análises *a posteriori*, e conclusões. A fase análises preliminares deve se constituir por um estudo descritivo e analítico dos fatores que envolvem o objeto que o trabalho de pesquisa busca clarificar. Dessa forma, a fase análises preliminares deve incluir: estudo epistemológico dos objetos envolvidos na aprendizagem, levantamento das práticas didáticas correntes, análise das concepções e dificuldades reveladas pelos estudantes. Como já mencionado, no capítulo 3 é desenvolvido um aprofundamento sobre a metodologia Engenharia Didática.

No capítulo 1, é elaborada uma descrição e análise das atividades presentes em uma disciplina de introdução aos algoritmos e programação. Com

esse capítulo, o objetivo é desenvolver uma parte da fase análises preliminares da Engenharia Didática.

O segundo capítulo é voltado para a visão de algumas obras de investigação recentes, relacionadas a esta pesquisa, e completa a fase análises preliminares da Engenharia Didática.

O capítulo 3 organiza os fundamentos teóricos e fundamentos metodológicos empregados nesta investigação.

O capítulo 4 apresenta o desenvolvimento da Engenharia Didática, ou seja: a descrição, e as observações e análises, das atividades que constituem a Engenharia planejada.

Nos apêndices A e B estão registradas as respostas escritas dos grupos que realizaram as atividades. O apêndice C apresenta a transcrição de entrevistas com alguns participantes das atividades.

1 UMA DISCIPLINA DE INTRODUÇÃO AOS ALGORITMOS

1.1 Caracterização geral das atividades na disciplina

As disciplinas de introdução aos algoritmos e programação, no âmbito geral, apresentam como objetivo o aprendizado para o tratamento de problemas computacionais.

Tal tratamento envolve várias atividades: entendimento da situação proposta como problema, articulação de conhecimentos já disponíveis e busca de novos conhecimentos para a elaboração de alguma estratégia de resolução, organização e detalhamento das ações que devem representar a estratégia de resolução constituída e a elaboração e a descrição do algoritmo ou do programa correspondente, realizada com a utilização de alguma linguagem algorítmica ou de alguma linguagem de programação.

Essa seqüência tem como ponto de partida a proposta de um problema computacional, e como ponto de chegada o texto de um algoritmo ou de um programa, que deve representar um método de resolução do problema proposto.

O estudante vivencia várias atividades: leitura e entendimento da proposta do problema, articulação e busca de conhecimentos, abstração, criatividade, organização lógica e descrição. A vivência em sala de aula permite afirmar que a conjugação dessas várias atividades é um dos fatores que contribuem para as dificuldades experimentadas pelos alunos.

Um dos componentes que se pode destacar nesse quadro é a atividade de abstração: as relações e os elementos reais ou concretos, que constituem o

cenário do problema proposto, deverão sofrer um processo de abstração e modelagem para que possam ser representados a partir dos recursos disponíveis na linguagem utilizada. Outro destaque pode ser dado ao aspecto dinâmico presente nos algoritmos e nos programas: o texto de um algoritmo ou de um programa representa um encadeamento de ações de um sistema computacional que são realizadas, uma a uma, ao longo do tempo. A dinâmica é definida pela organização do texto do algoritmo ou do programa, com suas variáveis e instruções, particularmente as estruturas de controle de fluxo de processamento.

1.1.1 Exemplo de atividade de abstração

Mesmo em casos muito simples, a elaboração do algoritmo pode demandar alguma atividade de abstração; pode-se colocar como exemplo o seguinte experimento: *exibir em ordem crescente os valores de três fichas numeradas extraídas aleatoriamente e sem reposição de uma urna que contêm inicialmente 100 fichas numeradas de 1 a 100*.

Na situação concreta desse experimento, a estratégia para realizar o sorteio e obter a classificação em ordem crescente dos três valores é imediata, e pode ser descrita assim: *depois de realizar as extrações das fichas, basta olhar os três valores obtidos no sorteio e anunciá-los em ordem crescente*; no cenário concreto, para o estudante, “sortear e dispor três valores em ordem crescente” é uma ação elementar.

No âmbito do tratamento computacional do problema, o estudante deverá: definir uma pequena estrutura de dados para o armazenamento dos valores, constituir um mecanismo para representar os sorteios aleatórios e construir um mecanismo de seleção para exibir os valores em ordem crescente.

Aquilo que, no plano concreto, era uma seqüência de operações ou ações imediatas e elementares, no âmbito do tratamento computacional, exige do estudante a constituição de uma estrutura de dados que, de forma abstrata, deverá representar as informações (valores sorteados) relativas à situação do problema, e o desenvolvimento de uma rotina composta por várias pequenas ações intermediárias, com a finalidade de obter o resultado desejado, e que

devem ser descritas em função dos recursos oferecidos pelo sistema de linguagem empregado.

Uma possibilidade para o programa, em linguagem C/C++, pode ser colocada assim:

```
int main( ){
    int num1, num2, num3, temp;
    srand(time(NULL));
    num1=1+rand()%100;
    do{
       num2=1+rand()%100;
    }while(num2==num1);
    do{
       num3=1+rand()%100;
    }while(num3==num1 || num3==num2);
    if(num1>num2){
       temp=num1; num1=num2; num2=temp;
    }
    if(num2>num3){
       temp=num2; num2=num3; num3=temp;
    }
    if(num1>num2){
       temp=num1; num1=num2; num2=temp;
    }
    cout<<"resultados: "<<num1<<"   "<<num2<<"   "<<num3<<endl;
    system("pause");
    return(0);
}
```

Torna-se evidente a distância entre o que é a execução do experimento concreto e a constituição do processo computacional correspondente. A execução do experimento concreto pode ser descrita por uma seqüência de duas ou três ações principais, já o processo computacional exige, nesse exemplo, uma série com vinte instruções.

1.1.2 Atividades de criação e descrição

Outro aspecto de destaque, entre as dificuldades vivenciadas pelos estudantes, é o conjunto das características da linguagem empregada, seja ela uma linguagem algorítmica ou uma linguagem de programação. Tais linguagens apresentam fatores opostos àqueles presentes na língua natural: numa linguagem de programação não se trabalha com redundâncias, nem com ambigüidades, nem com implícitos, e nessa linguagem as construções se desenvolvem com base nas expressões das estruturas lógicas formais e nas expressões algébricas.

O desenvolvimento de um algoritmo exige o domínio de conhecimentos relativos ao problema colocado – em situações de sala de aula os problemas propostos envolvem conhecimentos escolares de matemática, física, ciências em geral e vivências do cotidiano – e demandam algumas atividades de abstração, tanto para a constituição da estrutura de dados adequada como para a elaboração do algoritmo: é necessário desdobrar o plano de solução concebido diante das restrições ou limitações que são próprias da linguagem utilizada.

Para a elaboração de um algoritmo, inicialmente o estudante deve compreender claramente a proposta do problema, esse pode ser o primeiro entrave a ser ultrapassado. Depois disso, a tarefa exige a articulação de conhecimentos já disponíveis e a busca de novos conhecimentos, frente à proposta do problema, com o objetivo de compor um plano de ações que estabeleça um método de resolução.

Nessa fase, a tarefa demanda capacidade de criação e de abstração: em função das características da situação do problema e do esboço do método de resolução, cria-se um modelo que pode conter relações algébricas, aritméticas e lógicas. A fase seguinte é a elaboração da descrição do algoritmo; a execução dessa etapa depende do domínio dos elementos da linguagem algorítmica utilizada (sintaxe e semântica) e da habilidade em estabelecer os vínculos entre o plano de ações já delineado e os recursos que são próprios da linguagem algorítmica.

Ao final, o que se tem é uma representação estática (uma seqüência de linhas de texto) de um processo dinâmico; a representação estática indica uma lista de ações que constitui um método de resolução do problema proposto.

As fases descritas geralmente apresentam sobreposições, ou seja: durante a atividade em uma das fases do processo de desenvolvimento de um algoritmo, pode ocorrer a necessidade de se retomar alguma fase anterior diante da percepção de alguma característica ou fator que não havia sido notado.

A construção de programas procedimentais para computador se faz a partir de uma combinação adequada entre um algoritmo e uma estrutura de dados. Tanto o algoritmo como a estrutura de dados são definidos em função da natureza e das especificidades do problema computacional que se busca resolver e em função dos recursos da linguagem de programação a ser utilizada. A estrutura de dados corresponde ao conjunto de informações envolvidas na situação do problema, e o algoritmo representa uma seqüência de ações que ao ser realizada, por manipulação e transformações dos dados, conduz à resolução de uma instância do problema.

Assim, a prática de elaboração de algoritmos, que é fundamental na formação dos estudantes de Ciência da Computação e Engenharia, exige o desenvolvimento de capacidades variadas que, para muitos alunos, pode ser traduzida por dificuldades e entraves.

1.2 Caracterização da disciplina

1.2.1 Contexto e objetivos da disciplina

Um dos eixos principais da grade curricular de um curso de Ciência da Computação ou Sistemas de Informação ou Engenharia de Computação é composto por uma rede de disciplinas que devem tratar do desenvolvimento de capacidades e habilidades para o trabalho de elaboração e análise de algoritmos e implementação de programas. Na série inicial, normalmente encontram-se uma ou duas disciplinas introdutórias a esses conteúdos: introdução ao desenvolvimento de algoritmos e introdução a uma linguagem de programação de propósito geral (Pascal, Java, C/C++); em alguns cursos, no primeiro semestre, pode haver uma disciplina com uma carga semanal de quatro ou seis aulas; em outros o trabalho equivalente pode ser realizado em duas disciplinas: Laboratório

de Programação e Desenvolvimento de Algoritmos, com carga horária total equivalente.

Na seqüência dos cursos citados acima, essas disciplinas de introdução são complementadas ou aprofundadas em várias direções: conceitos de linguagens de programação, projetos de estruturas de dados, análise de complexidade de algoritmos, metodologias para modelagem e programação.

Nas disciplinas iniciais o objetivo principal é o desenvolvimento de capacidades e habilidades para o tratamento de problemas computacionais e a vivência em ambientes de programação. Os problemas computacionais, propostos nesses cursos introdutórios, envolvem conceitos trabalhados no ensino fundamental ou no ensino médio (matemática e ciências), conceitos do cotidiano (contagens simples, calendário, jogos, etc.) e conceitos específicos da Ciência da Computação.

A expressão *tratamento de problemas computacionais*, mencionada acima, engloba aquela série de atividades já descrita:

- leitura, interpretação e compreensão do texto com a proposta do problema;
- busca e articulação dos conhecimentos exigidos para a constituição do processo de resolução;
- organização lógica do processo de resolução;
- descrição do método de resolução delineado, utilizando-se uma linguagem algorítmica;
- elaboração, implementação e depuração do programa correspondente.

As atividades da disciplina são conduzidas a partir de dois pólos principais: problemas computacionais e recursos de uma linguagem algorítmica e de uma linguagem de programação.

Uma linguagem algorítmica é uma linguagem formal, mas que não é implementada em um sistema computacional, sendo utilizada para expressar os algoritmos computacionais em fase anterior à implementação do programa.

Um sistema de linguagem de programação é definido pelos próprios elementos da linguagem de programação (sintaxe, semântica, vocabulário) e por

um programa especial (compilador ou interpretador) responsável por converter o programa fonte, escrito na linguagem de programação, em um programa executável pelo sistema computacional.

1.2.2 Primeiros conceitos abordados na disciplina

As primeiras atividades da disciplina introduzem os conceitos de problema computacional, algoritmo e programa.

A idéia de problema computacional é introduzida a partir de algumas características: um problema computacional é um problema cuja resolução envolva o trabalho de transformação ou manipulação de informações; um problema computacional indaga sobre uma situação genérica, de tal forma que a resposta ao problema é a descrição de um processo que represente um método de resolução.

Um problema proposto colocado numa situação de aula de matemática, por exemplo:

> Um lote de 3872 parafusos deve ser embalado em caixas com 40 unidades e caixas de 10 unidades, utilizando-se preferencialmente as caixas grandes. Quantas caixas grandes e quantas caixas pequenas são necessárias para embalar os parafusos? Quantos parafusos não serão embalados por não completarem uma caixa pequena?

pode ser modificado para ser colocado numa situação de aprendizagem de algoritmos e programação assim:

> Um lote de vários parafusos deve ser embalado em caixas com 40 unidades e caixas de 10 unidades, utilizando-se preferencialmente as caixas grandes. Conhecendo-se a quantidade de parafusos do lote, como determinar quantas caixas grandes e quantas caixas pequenas são necessárias para embalar os parafusos e quantos parafusos não serão embalados por não completarem uma caixa pequena? (Martins e Rodrigues, 2008, p.64)

Na situação de uma aula de matemática, a resolução do problema é constituída por duas operações de divisão:

```
3872 | 40          32 | 10
 272   96           2    3
  32
```

e a resposta pode ser expressa assim: *para embalar os 3872 parafusos são necessárias 96 caixas grandes e 3 caixas pequenas, 2 parafusos não serão embalados.*

Na versão modificada para uma situação de aprendizagem de algoritmos e programação, a resposta deve ser a elaboração do processo:

- realizar a entrada da quantidade de parafusos;
- realizar o cálculo do quociente da divisão da quantidade de parafusos por 40, este valor é a quantidade de caixas grandes;
- realizar o cálculo do resto da mesma divisão: o valor obtido é a quantidade de parafusos que não chega a completar uma caixa grande e serão dispostos em caixas pequenas;
- realizar o cálculo do quociente da divisão do resto, obtido na operação anterior, por 10; este valor é a quantidade de caixas pequenas;
- realizar o cálculo do resto da divisão anterior: o valor é a quantidade de parafusos que não serão embalados;
- exibir os resultados: quantidade de caixas grandes, quantidade de caixas pequenas e quantidade de parafusos não embalados.

O processo descrito é um algoritmo: uma seqüência finita de ações elementares, de execução finita, que a cada execução produz a resolução de um caso do problema. A produção de um algoritmo é sempre vinculada à existência de um problema computacional que deve ser resolvido. A construção de um algoritmo só faz sentido se ele for necessário.

1.2.3 Outros conceitos iniciais tratados na disciplina

Paralelamente ao conceito de problema computacional, são apresentados e discutidos os primeiros conceitos relativos à linguagem algorítmica e ao sistema de linguagem de programação.

O computador é descrito como um conjunto de dispositivos que executa, desde que seja preparado (programado) para isso, a manipulação de dados que representam as informações da situação concreta.

Uma linguagem de programação possui um conjunto de tipos de dados primitivos que suportam a representação das informações. O armazenamento dos dados na memória principal do computador ocorre por meio das variáveis, cada variável é um conjunto de células de memória (*bytes*) que pode armazenar um dado em cada instante. O sistema computacional, ao executar um programa, realiza a alocação das variáveis necessárias ao processo e a partir daí os seus conteúdos podem ser definidos ou redefinidos por instruções ou comandos do programa. Essa é a essência de um programa sob o paradigma de programação imperativa: as variáveis armazenam dados e esses dados podem ser definidos ou redefinidos como efeito da execução de alguma instrução do programa.

Na fase inicial do curso da disciplina, os processos construídos envolvem as operações de “entrada” e armazenamento de dados, as operações de “saída” e as operações de atribuição. A entrada de um dado consiste na ação do usuário fornecer (digitar) o dado e na ação do sistema receber e armazenar o dado em uma variável. A saída de um dado é a ação do sistema produzir, no dispositivo de saída (monitor de vídeo), a exibição do dado. As instruções de atribuição contêm uma variável que é o alvo da atribuição (é a variável cujo conteúdo será redefinido) e uma expressão que deve ser avaliada pelo sistema e cujo resultado será disposto como novo conteúdo da variável alvo.

No exemplo do problema dos parafusos, a descrição do algoritmo pode ser vista no Quadro 1.

Quadro 1: Algoritmo – problema dos parafusos

```
exemplo( )
leia(quantparafusos);
quantgrandes← quantparafusos div 40;
resto← quantparafusos mod 40;
quantpequenas← resto div 10;
sobra← resto mod 10;
imprima(quantgrandes);
imprima(quantpequenas);
imprima(sobra);
```

Nesse exemplo, são empregadas as variáveis `quantparafusos`, `quantgrandes`, `resto`, `quantpequenas` e `sobra`, todas essas variáveis devem ser de tipo inteiro, ou seja, cada uma dessas variáveis possui capacidade para armazenar um valor inteiro a cada momento.

As linguagens de programação possuem alguns tipos primitivos de dados, esses tipos primitivos podem ser organizados em três grandes categorias:

- tipos numéricos que podem representar quantidades, medidas, valores monetários, etc.;
- tipo lógico que representa os valores lógicos (verdadeiro ou falso);
- tipos alfanuméricos que podem representar dados não numéricos: nome de um produto, nome de uma cidade, código não numérico de uma peça, etc..

Além da natureza dos dados, a concepção de um tipo primitivo compreende a constituição de operadores para os dados daquele tipo (operadores aritméticos e de relação de ordem no caso de tipos numéricos), a especificação de conjuntos ou faixas de valores admitidos, e também a forma de implementação física no sistema.

O processo é composto por uma instrução de entrada (`leia(quantparafusos)`), quatro instruções de atribuição (`quantgrandes←quantparafusos div 40;` ...) e três instruções de saída (`imprima(quantgrandes);` ...); é essencial a ordem de disposição e, como conseqüência, a ordem de execução dessas instruções.

Nas instruções de atribuição, as expressões envolvem operadores da aritmética de valores inteiros (cálculo do quociente `div` e cálculo do resto `mod` de divisões entre valores inteiros).

A partir do texto do algoritmo, a próxima etapa do trabalho é a elaboração do texto do programa que deverá representar o processo já descrito em linguagem algorítmica. Essa elaboração consiste em converter a descrição do processo, obtida em linguagem algorítmica, para uma linguagem de programação, com o acréscimo de detalhes relativos à constituição da interface entre o usuário

do programa e o sistema computacional, e de outros elementos que são específicos da linguagem ou do ambiente de programação utilizado.

1.2.4 Elementos do texto de um programa

Para o exemplo apresentado, com a utilização da linguagem C/C++, o texto do programa é o seguinte:

```
#include <iostream>
using namespace std;

int main( ){
  int quantparafusos, quantgrandes, quantpequenas,
      sobra, resto;
  cout<<"digite a quantidade de parafusos a embalar: ";
  cin>>quantparafusos;
  quantgrandes= quantparafusos / 40;
  resto= quantparafusos % 40;
  quantpequenas= resto / 10;
  sobra= resto % 10;
  cout<<"resultados: "<<endl;
  cout<<"    quantidade de caixas  grandes: "
      <<quantgrandes<<endl;
  cout<<"    quantidade de caixas pequenas: "
      <<quantpequenas<<endl;
  cout<<"    parafusos nao embalados: "
      <<sobra<<endl;
  system("pause");
  return(0);
}
```

A Figura 1 é um exemplo da janela de execução do programa.

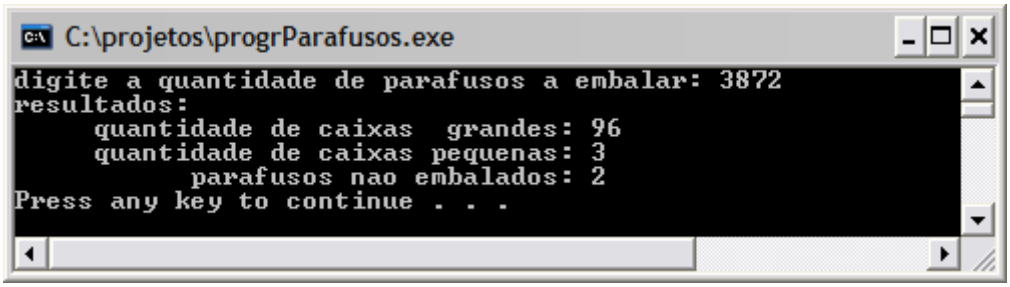


Figura 1 – Modelo de janela de execução

A conversão das instruções de atribuição e das expressões aritméticas é feita com o ajuste de alguns dos símbolos utilizados, conforme o Quadro 2.

Quadro 2: Símbolos de operadores aritméticos e de atribuição

operação	representação no algoritmo	representação no programa
operador atribuição	←	=
operador quociente	**div**	/
operador resto	**mod**	%
operador multiplicação	*	*
operador adição	+	+
operador subtração	-	-

Observação: entre os operadores aritméticos, a ordem de prioridade é a habitual da matemática, a associatividade é a usual (da esquerda para a direita) e, para especificar agrupamentos, utiliza-se pares de parênteses em vários níveis (não são utilizados colchetes ou chaves).

A instrução de entrada (`leia(quantparafusos)`) é descrita no programa por `cin>>quantparafusos` e precedida por uma instrução de saída: `cout<<"digite a quantidade de parafusos a embalar: "` cuja finalidade é orientar sobre a digitação de dados que o usuário deve realizar. Nas instruções de saída (`imprima(quantgrandes); ...`) cada resultado é acompanhado de uma mensagem que aponta o significado correspondente: `cout<<" quantidade de caixas grandes: "<<quantgrandes<<endl`.

Na primeira linha do texto do programa é disposta a diretiva de inclusão de uma biblioteca (`#include <iostream>`), esse é um componente do sistema de linguagem em que estão definidos diversos recursos necessários para as operações de entrada e saída de dados. O sistema de linguagem C++ possui várias outras bibliotecas que agrupam recursos organizados por finalidade, por exemplo: na biblioteca `cmath` estão definidas várias funções matemáticas elementares (trigonométricas, exponenciais, logarítmicas, etc.), na biblioteca `ctime` estão definidos recursos para manipulação do relógio do sistema.

Na segunda linha está uma instrução (`using namespace std`) cujo efeito, no caso desse exemplo, é permitir a omissão do identificador `std` à esquerda de

cada instrução `cin` e `cout`; sem essa instrução seria necessário escrever `std::cin` e `std::cout`, para especificar que as ações correspondentes (entradas e saídas) devem ser executadas no respectivo dispositivo padrão (`std` – standard), mais especificamente: teclado e tela do monitor.

A linha `int main( ) {` contêm a declaração da função principal do programa. Na fase inicial do curso da disciplina, todos os programas são compostos apenas pela função principal, mas um programa pode conter além dessa função principal (`main( )`) várias outras funções auxiliares. Sempre que um programa C/C++ é executado, o início e o final de tal execução ocorre na função principal e durante a execução dessa função pode ocorrer o acionamento de alguma função auxiliar; quando isso acontece a execução da função principal é suspensa e a função auxiliar passa a ser executada: ao final da execução da função auxiliar é retomada a execução da função principal no ponto em que ocorreu a interrupção. Uma função auxiliar também pode acionar outra função auxiliar e o mecanismo de interrupção e retomada de execução é o mesmo.

O conceito e os recursos de modularização de um programa, que correspondem à possibilidade de construção e utilização de funções auxiliares em C ou C++, são tratados após algumas semanas do início do curso da disciplina.

A especificação de tipo `int` colocada na declaração indica que ao final de sua execução, a função `main( )` irá devolver ao sistema operacional um valor inteiro, esse valor a ser retornado é especificado na instrução `return(0)`. A chave `{` marca o início do bloco de instruções que define a função principal, na última linha `}` marca o fim desse bloco.

A instrução `system("pause")` produz uma interrupção na execução do programa antes de seu encerramento; essa interrupção é necessária para que o usuário possa visualizar o conteúdo da janela produzida pela execução do programa. A necessidade dessa instrução pode não ocorrer, depende do ambiente de programação utilizado.

A instrução `int quantparafusos, quantgrandes, quantpequenas, sobra, resto` é denominada declaração de variáveis; o primeiro elemento nessa declaração é a especificação do tipo (`int`) desejado para as variáveis e a

seguir há uma lista de identificadores das variáveis que se pretende utilizar no processo. Nesse exemplo, cinco variáveis de tipo valor inteiro são utilizadas; essas variáveis constituem a estrutura de dados desse programa.

A ação do sistema computacional relativa à declaração de variáveis é a alocação de um conjunto de células de memória para cada variável e a vinculação de cada um deles com o identificador da variável especificado na declaração. O sistema trabalha internamente com um sistema de endereçamento desses conjuntos de células, esse sistema de endereços possibilita o acesso aos dados correspondentes; no texto do programa a referência às variáveis é estabelecida por seus identificadores.

Essa declaração produz, na memória principal do sistema, a estrutura de dados necessária para o tratamento do problema computacional cujo método de resolução é representado pelo programa. Nas disciplinas de introdução aos algoritmos e programação essas estruturas de dados são bastante simples; estruturas mais complexas (filas, pilhas, árvores, ...) são abordadas em disciplinas avançadas que complementam a inicial.

A resolução do problema utilizado como exemplo exigiu o uso apenas de variáveis de tipo valor numérico inteiro, mas em outras situações pode haver a necessidade do emprego de outros tipos como numérico real, caractere e lógico.

1.2.5 Ambiente de programação

Um ambiente de programação é composto basicamente por uma ferramenta que permita a produção do texto do programa (editor de textos), outras ferramentas que facilitem a manipulação dos arquivos correspondentes a esses textos, uma ferramenta que faça a compilação do programa e outra que acione a execução do programa compilado.

A compilação do programa é a conversão do programa descrito na linguagem de alto nível (C /C++, por exemplo) para a linguagem de máquina do computador, pois o sistema computacional não executa diretamente o programa

descrito em uma linguagem de alto nível. Nesse processo de compilação, os ambientes de programação atuais têm capacidade para indicar diversos tipos de erro de sintaxe cometidos pelo programador; nessas situações o ambiente emite mensagens e apontamentos que auxiliam na localização ou correção dos mesmos. Esses ambientes oferecem também outras ferramentas que podem facilitar as tarefas de depuração do programa.

1.3 Etapas no tratamento de problemas computacionais

1.3.1 Entendimento da proposta do problema

Na situação em que esse trabalho introdutório é feito em uma única disciplina, uma abordagem que pode ser empregada é, desde as primeiras aulas, propor atividades que possam promover os primeiros contatos do estudante com o cenário mais completo possível de tratamento de problemas computacionais com um conjunto mínimo de recursos da linguagem algorítmica, da linguagem de programação e do ambiente de programação; esses conjuntos de recursos são ampliados gradativamente ao longo do curso.

Essa forma de abordagem leva o estudante a vivenciar, já na fase inicial, o cenário completo da tarefa básica que é a elaboração de programas para a resolução de problemas computacionais. Esse cenário envolve desde a leitura e o entendimento da proposta de um problema computacional até a implementação e a depuração do programa que traduz seu método de resolução.

O primeiro aspecto é o aprendizado para a leitura e o entendimento da proposta do problema; nesse sentido pode ser interessante a tentativa de padronizar minimamente a estrutura dos textos com as propostas dos problemas. No problema da embalagem dos parafusos, já descrito, pode-se observar a seguinte estrutura:

- descrição breve e completa do cenário do problema:

  Um lote de vários parafusos deve ser embalado em caixas com 40 unidades e caixas de 10 unidades utilizando-se preferencialmente as caixas grandes.

- indicação dos elementos que deverão "alimentar" o processo, ou seja: os dados que deverão ser introduzidos pelo usuário:

  Conhecendo-se a quantidade de parafusos do lote ...

- indicação dos elementos que deverão constituir o resultado do processo:

  [...] como determinar quantas caixas grandes e quantas caixas pequenas são necessárias para embalar os parafusos e quantos parafusos não serão embalados por não completarem uma caixa pequena?

Outro exemplo em que o mesmo padrão pode ser observado:

Uma pequena cooperativa agro-industrial deve produzir manteiga comum e manteiga especial na proporção 4:1, ou seja: para cada 4 kg de manteiga comum deve produzir 1 kg de manteiga especial. Sabe-se que a produção de 1 kg de manteiga comum consome 1,6 kg de creme de leite e que a produção de 1 kg de manteiga especial consome 2,2 kg de creme. Conhecendo-se a quantidade (kg) de creme de leite disponível, como determinar as quantidades de manteiga comum e especial que a cooperativa deve produzir? (Adaptado de notas de aulas)

- descrição breve e completa do cenário do problema:

  Uma pequena cooperativa agro-industrial deve produzir manteiga comum e manteiga especial na proporção 4:1, ou seja: para cada 4 kg de manteiga comum deve produzir 1 kg de manteiga especial. Sabe-se que a produção de 1 kg de manteiga comum consome 1,6 kg de creme de leite e que a produção de 1 kg de manteiga especial consome 2,2 kg de creme.

- indicação dos elementos que deverão "alimentar" o processo, ou seja: os dados que deverão ser introduzidos pelo usuário:

  Conhecendo-se a quantidade (kg) de creme de leite disponível [...]

- indicação dos elementos que deverão constituir o resultado do processo:

> *[...] como determinar as quantidades de manteiga comum e especial que a cooperativa deve produzir?*

Agora um exemplo em que não há essa estrutura:

> Elaborar um programa para calcular a adição, a subtração, a multiplicação e a divisão de dois números complexos. Utilize a biblioteca de sua escola ou a Internet para pesquisar a fórmula adequada. (ZAMBONI; PAMBOUKIAN; BARROS, 2006, p. 31)

A padronização mínima na fase inicial do curso da disciplina é interessante no sentido de facilitar ao estudante o exercício de interpretação e compreensão das propostas dos problemas. Nas fases mais adiantadas, quando o estudante já tiver atingido alguma autonomia, essa padronização deve ser abandonada, pois é essencial que o aluno desenvolva sua capacidade, como futuro profissional da área, para lidar com as propostas dos problemas colocados: identificar os objetivos, as informações disponíveis e as informações que se pretende obter.

1.3.2 Criação do método de resolução

Essa etapa da tarefa talvez seja a de maior grau de dificuldade, pois exige a combinação de várias capacidades do estudante. Esboçar um método de resolução é estabelecer relações adequadas entre as informações que se pretende obter com aquelas que são disponíveis, não basta que tais relações sejam relações verdadeiras: o que deve ser produzido é um encadeamento de relações intermediárias válidas em que, a partir das informações disponíveis, possam ser obtidas as informações desejadas.

A produção desse encadeamento demanda: conhecimentos já construídos pelo aluno, percepção de elementos da própria proposta do problema, transformação desses conhecimentos e elementos em ações ou operações que conduzam ao método de resolução do problema.

O aluno deve desenvolver a capacidade de criação e de organização de ações que traduzam um método de resolução do problema. Vale lembrar que, nessa construção, o estudante trabalha com abstrações a partir daquilo que é o

cenário concreto do problema para a elaboração de uma representação de seu método de resolução.

Desde que haja tempo disponível, é interessante propor seqüências de problemas que sejam de naturezas diferentes, mas cujos métodos de resolução sejam semelhantes. Por exemplo:

> Um reservatório de água, inicialmente vazio, deve ser abastecido por uma bomba de recalque. Conhecendo-se a vazão da bomba, em litros por segundo, e a capacidade do reservatório, em litros, como calcular o tempo necessário para abastecer o reservatório? O tempo necessário deve ser expresso na forma: *h* horas, *m* minutos e *s* segundos (considere todas as quantidades inteiras). (MARTINS; RODRIGUES, 2008, p.56)

> Uma impressora, que tem a capacidade de imprimir 5 páginas de texto por minuto, deverá produzir a impressão de todas as páginas de uma lista telefônica. Conhecendo-se a quantidade de páginas da lista a ser impressa, como determinar o tempo necessário para a impressão? Esse tempo deverá ser expresso na forma: *h* horas, *m* minutos e *s* segundos. (MARTINS; RODRIGUES, 2008, p.65)

É importante também valorizar a participação dos alunos com a apresentação e a discussão de pelo menos dois métodos de resolução diferentes produzidos por eles; essa prática, além de incentivar a postura participativa do estudante, costuma levar a outros resultados interessantes: para o aluno é muito mais natural analisar, criticar ou questionar aquilo que seu colega produziu do que a solução apresentada pelo professor.

Ao mesmo tempo em que o senso crítico é trabalhado, os alunos percebem concretamente que o processo de resolução, em geral, não é único, e que pode haver comparações sobre a clareza ou sobre a eficiência de um método em relação a outro. Por exemplo: para o problema dos parafusos, a maioria das soluções produzidas é aquela já apresentada (inicialmente estabelecer agrupamentos de 40 e depois de 10 parafusos), mas é considerável a quantidade de alunos que pensam em agrupar primeiro de 10 em 10 peças e depois organizar de 4 em 4 os agrupamentos de 10. Para o problema da impressora também é comum a produção de pelo menos duas estratégias de resolução, exibidas no Quadro 3.

Quadro 3: Algoritmos – problema da impressora

tempoimpressão( )

```
leia(qpag);
totmin ← qpag div 5;
resto ← qpag mod 5;
qh ← totmin div 60;
qm ← totmin mod 60;
qs ← resto * 12;
imprima(qh);
imprima(qm);
imprima(qs);
```

tempoimpressão( )

```
leia(qpag);
totseg ← qpag * 12;
qh ← totseg div 3600;
resto ← totseg mod 3600;
qm ← resto div 60;
qs ← resto mod 60;
imprima(qh);
imprima(qm);
imprima(qs);
```

Na fase de elaboração do método de resolução, os alunos são encorajados a utilizar diversas formas de registro: produção de um texto em língua natural, elaboração de figuras, combinação de textos e elementos gráficos.

Nessa fase é bastante comum a tentativa de obter o processo geral de solução a partir de um ou de alguns casos particulares do problema e, pelo menos para os problemas propostos nas primeiras semanas, essas tentativas produzem o efeito desejado (obter o método de resolução do problema).

Pode-se observar que quanto maior for o grau de organização e detalhes do esboço do método de resolução produzido, mais facilidade o aluno terá ao produzir o registro formal em linguagem algorítmica e depois em linguagem de programação.

Uma estratégia de aula que pode ser empregada é, depois de identificar os elementos centrais do processo de resolução, ainda com uma organização pouco clara e detalhes sem aprofundamento, introduzir uma descrição intermediária do processo, já muito próxima do texto que será produzido em linguagem algorítmica. Um exemplo:

- realizar a entrada da quantidade de parafusos;
- realizar o cálculo do quociente da divisão da quantidade de parafusos por 40, este valor é a quantidade de caixas grandes;
- realizar o cálculo do resto da mesma divisão, o valor obtido é a quantidade de parafusos que não chega a completar uma caixa grande, estes parafusos serão dispostos em caixas pequenas;
- realizar o cálculo do quociente da divisão do resto, obtido na operação anterior, por 10, este valor é a quantidade de caixas pequenas;

- realizar o cálculo do resto da divisão anterior, o valor é a quantidade de parafusos que não serão embalados;
- exibir os resultados: quantidade de caixas grandes, quantidade de caixas pequenas e quantidade de parafusos não embalados.

Outro exemplo (problema da impressão da lista telefônica):

> Inicialmente deve ser feita a entrada da quantidade de páginas (informação inicial) a imprimir. Depois disso, calcula-se o quociente inteiro e o resto da divisão da quantidade de páginas por 5. O quociente corresponde ao total de minutos (valor inteiro) necessários para a impressão. Essa quantidade de minutos deve ser dividida por 60 para determinar-se a quantidade de horas (quociente) e a parcela de minutos (resto) do tempo de impressão. A quantidade de segundos será o produto do resto daquela primeira divisão por 12. O resto daquela primeira divisão é a quantidade de páginas cuja impressão não chega a completar um minuto e se a velocidade é de 5 páginas por minuto, isso significa que a impressão de cada página leva 12 segundos.

Nessas descrições, além de destacar as ações, é interessante indicar os significados dos dados que são transformados e obtidos. Essa prática deve favorecer o entendimento da vinculação entre as informações do cenário real do problema e o modelo que se elabora para representá-lo.

1.3.3 Descrição do algoritmo e implementação do programa

Se o estudante alcançar um bom grau de organização e detalhes na elaboração do esboço do processo de resolução, a descrição formal do algoritmo, em linguagem algorítmica, é produzida mais facilmente: na fase inicial da disciplina, o aluno deve ter o domínio sobre uma pequena quantidade de instruções da linguagem algorítmica (entrada, saída e atribuição) e sobre poucos elementos sintáticos.

Um tipo de exercício que pode ser utilizado na busca da capacidade para a produção do texto do algoritmo é fornecer ao aluno, junto com a proposta do problema, uma descrição do método de resolução, por exemplo:

Considere o seguinte problema, adaptado de Martins e Rodrigues (2008, p.66):

"Os recursos financeiros de uma prefeitura são provenientes de 3 fontes:

- *impostos municipais – recolhidos pela própria prefeitura,*
- *impostos estaduais – repassados à prefeitura pelo governo do estado, e*
- *impostos federais – repassados à prefeitura pelo governo federal.*

Conhecendo-se os valores relativos a essas três fontes, como obter as porcentagens correspondentes a esses volumes de recursos?"

e o seguinte roteiro de método de resolução:

- realizar a entrada dos volumes (três fontes): Vmun, Vest, Vfed
- obter a soma dos três volumes: tot = Vmun + Vest + Vfed
- obter as porcentagens correspondentes:
  Pmun = Vmun/tot x 100
  Pest = Vest/tot x 100
  Pfed = Vfed/tot x 100
- exibir as porcentagens calculadas: Pmun, Pest, Pfed.

Elabore a descrição do algoritmo correspondente.

A resolução desse tipo de exercício, pelo aluno, favorece a identificação do valor que a organização e os detalhes do esboço do método de resolução devem merecer e, por outro lado, leva a uma prática de conversão entre as duas formas de expressar o método de resolução: a expressão com os elementos da língua natural e da linguagem algébrica, e a expressão formalizada com a linguagem algorítmica. A conversão é uma operação que envolve dois registros de representação do mesmo objeto, no caso o método de resolução do problema; esses conceitos serão apresentados no capítulo 3.

Na fase de implementação e testes do programa, o estudante deve operar um ambiente de programação para:

- realizar a elaboração do texto do programa (com o emprego do editor de texto do ambiente), que se constitui em outra tarefa de conversão entre registros de representação: a linguagem algorítmica e a linguagem de programação;
- acionar a compilação do programa e, a partir da observação das mensagens e apontamentos colocados pelo sistema, buscar corrigir as falhas indicadas e então acionar novamente a compilação; em muitos casos esse ciclo (acionar o

compilador – corrigir – acionar o compilador ...) é repetido algumas vezes até que se obtenha a compilação do programa;

- acionar a execução do programa compilado e analisar os resultados apresentados por sua execução; nessa fase o estudante deve exercitar sua visão crítica sobre os resultados produzidos, para reconhecer e garantir a validade do programa construído. Nesse momento é imprescindível a noção de processo dinâmico inerente ao programa; as respostas geradas pela execução do programa devem ser entendidas como o resultado do mecanismo dinâmico representado pelo programa.

A fase de implementação do programa consiste em produzir mais uma representação do método de resolução do problema computacional, com o emprego de uma linguagem de programação que é rígida em seus elementos sintáticos e semânticos; essa produção requer um aprofundamento no grau de detalhamento relativo à constituição da interface entre o sistema computacional e o usuário do programa, e outros elementos que são específicos da linguagem de programação.

Para essa fase dos trabalhos, as aulas são conduzidas em laboratórios equipados com computadores preparados para a operação do ambiente de programação.

Durante as primeiras semanas do curso as dificuldades dos estudantes em termos de operação do ambiente são freqüentes, e assim as atividades de observação e análise de resultados apresentados são colocadas, pelos próprios alunos, num plano secundário. Eles tomam como meta principal realizar a compilação e a execução do programa; depois de algum tempo de vivência com o ambiente, ao superarem as dificuldades em operar as ferramentas, torna-se necessária a intervenção do professor com a indicação da importância daquelas atividades de análise crítica sobre os resultados produzidos com a execução dos programas.

As observações no dia a dia de sala de aula permitem afirmar que, com o transcorrer dos trabalhos, uma parcela significativa dos estudantes atinge um bom grau de domínio sobre as ferramentas do ambiente, e também sobre os elementos iniciais (sintaxe e semântica) da linguagem de programação. Esses

alunos demonstram um melhor aproveitamento e desempenho ao passarem para as próximas etapas da disciplina.

1.4 Os próximos recursos: estruturas de controle

Na etapa inicial são tratados vários conceitos fundamentais (algoritmo, programa, sistema de linguagem de alto nível, ambiente de programação, variável, instruções de entrada e saída, primeiros tipos primitivos, instruções de atribuição, operadores aritméticos) que permitem o tratamento de problemas computacionais cujos métodos de resolução exijam apenas a constituição de fluxos de processamento seqüenciais, ou seja, os métodos de resolução são expressos por uma seqüência ordenada de instruções que devem ser executadas uma a uma, exatamente uma única vez, desde a primeira até a última. No próximo bloco da disciplina são introduzidas as estruturas de controle de seleção e de repetição, que permitem a constituição de fluxos de processamento alternativos e repetitivos.

Com os controles de seleção, torna-se possível condicionar a execução de um determinado bloco de instruções ao valor de uma expressão lógica ou numérica, por exemplo:

```
se a>b então t←a;  a←b;  b←t;
```

nessa construção, a seqüência das três atribuições dispostas no interior da moldura retangular só será executada se o valor armazenado como conteúdo da variável **a** for maior do que o valor definido como conteúdo da variável **b**; caso isso não ocorra (caso **a** não seja maior do que **b**), o bloco com as três atribuições não será executado e o fluxo de processamento seguirá para a próxima instrução, aquela que estiver abaixo da instrução `se ... então ...`.

Normalmente trabalha-se com duas ou três formas de estruturas de controle de seleção: controle de seleção de um ramo, controle de seleção de dois ramos e controle de seleção de vários ramos. Os dois primeiros tipos estabelecem o mecanismo de seleção a partir da avaliação de uma expressão lógica e o terceiro tipo realiza o processo de seleção com base em uma

expressão numérica de tipo inteiro. As formas gerais dessas estruturas de controle estão descritas no Quadro 4.

Quadro 4: Estruturas de controle de seleção

linguagem algorítmica	linguagem C/C++
estrutura de controle de seleção de um ramo	
se *expressão* **então** *bloco;* *próxima instrução;*	**if(***expressão***){** *bloco;* **}** *próxima instrução;*
estrutura de controle de seleção de dois ramos	
se *expressão* **então** *blocoA;* **senão** *blocoB;* *próxima instrução;*	**if(***expressão***){** *blocoA;* **}** **else{** *blocoB;* **}** *próxima instrução;*
estrutura de controle de seleção de vários ramos	
caso *expressão* **seja** *canal1: bloco1;* *canal2: bloco2;* ... *canalN: blocoN;* **senão** *blocoA;* *próxima instrução;*	**switch(***expressão***){** **case** *canal1: bloco1;* **break;** **case** *canal2: bloco2;* **break;** ... **case** *canalN: blocoN;* **break;** **default:** *blocoA;* **}** *próxima instrução;*

Nas duas primeiras formas (seleção de um ramo e seleção de dois ramos) a *expressão*, que se configura como o elemento chave do controle, é uma expressão lógica (booleana) que pode ser obtida pela utilização de operadores relacionais (operadores de relação de ordem) ou operadores lógicos (conjunção, disjunção, negação); por exemplo:

```
se  ladoA<ladoB  ou  ladoA<ladoC  então ...

if( ladoA<ladoB || ladoA<ladoC ) ...
```

Na terceira forma (seleção de vários ramos) a *expressão* de controle é uma expressão de tipo inteiro ou de tipo ordinal e cada *canal* é uma constante com o mesmo tipo da expressão. O sistema avalia a *expressão* e aciona o canal cujo valor coincida com o valor da *expressão*, ou o canal alternativo (*senão*) se não houver a coincidência, e assim executa o *bloco* de instruções correspondente.

No Quadro 5 está descrito um exemplo de algoritmo com a utilização de estruturas de controle de seleção.

Quadro 5: Algoritmo – exemplo de controles de seleção

```
diasnomes( )
leia(mes); leia(ano);
caso mes seja
     2 :  bst← ano mod 4=0  e  ano mod 100≠0  ou  ano mod 400=0;
          se bst então  quantdias← 29;
                 senão  quantdias← 28;
     4 :
     6 :
     9 :
    11 :  quantdias← 30;
   senão  quantdias← 31;
imprima(quantdias);
```

As estruturas de controle de repetição permitem a elaboração de fluxos de processamento repetitivos, assim: um bloco de instruções descrito uma única vez no algoritmo ou no programa poderá ter sua execução repetida por várias vezes. Um exemplo clássico de aplicação dessa categoria de recursos é o processo de cálculo do máximo divisor comum entre dois inteiros maiores do que zero: a partir da entrada de dois valores calcula-se o resto da divisão de um pelo outro e enquanto o resto obtido for diferente de zero, toma-se o próximo dividendo como o divisor anterior; o resto obtido como próximo divisor e calcula-se o novo resto; quando for obtido o resto igual a zero encerra-se o processo e o máximo divisor comum será o valor do último divisor. A descrição do algoritmo encontra-se no Quadro 6.

Quadro 6: Algoritmo – cálculo do máximo divisor comum

```
maxdiv( )
leia(a); leia(b);
resto← a mod b;
enquanto resto>0 faça
    a← b;
    b← resto;
    resto← a mod b;
imprima(b);
```

Nessa construção, as três atribuições dispostas no bloco sob a ação do controle `enquanto ... faça ...` poderão ser executadas por vezes repetidas; o elemento chave do controle nesse caso é uma expressão lógica (`resto>0`), da mesma forma já descrita para as duas primeiras estruturas de controle de seleção.

A dinâmica correspondente envolve a avaliação da expressão de controle (verdadeira ou falsa) e a execução do bloco disposto logo abaixo da instrução se a expressão de controle for verdadeira; nesse caso, ao final de cada execução do bloco, o sistema volta a avaliar automaticamente a expressão de controle e repete o processo; caso o valor lógico da expressão de controle seja falso, o bloco com as três atribuições não será executado e o fluxo de processamento, com a finalização do anel de repetição, seguirá para a próxima instrução registrada abaixo do bloco.

A instrução `enquanto ... faça ...` é denominada instrução de controle de repetição por teste lógico *a priori* (a primeira avaliação da expressão de controle ocorre antes da primeira execução do bloco), e a instrução `faça ... enquanto ...`, que realiza um mecanismo semelhante, é denominada instrução de controle de repetição por teste lógico *a posteriori* (a primeira avaliação da expressão de controle ocorre após a primeira execução do bloco).

Além dessas duas formas de controle, as linguagens de programação podem oferecer mais um tipo de instrução de repetição, que é denominado controle por variável contadora ou por contagem. Esse terceiro tipo de controle pode apresentar características particulares em cada linguagem de programação.

Esse controle corresponde essencialmente a especificar-se uma faixa de variação para a variável contadora (desde um valor inicial até um valor final com um determinado passo de variação) e ter o processo de repetição vinculado àquela variação, assim: para cada valor da variável contadora, desde o valor inicial até o valor final, executa-se uma vez o bloco de instruções.

Um exemplo de aplicação dessa terceira forma é obter a soma dos divisores próprios de um valor inteiro maior do que um, a descrição do algoritmo encontra-se no Quadro 7.

Quadro 7: Algoritmo – soma dos divisores próprios

```
somadivisores( )
leia(valor);
soma← 0;
para d de 1 até (valor div 2) passo 1 faça
   resto← valor mod d;
   se resto=0 então  soma← soma+d;
imprima(soma);
```

Observação: do ponto de vista lógico, bastariam a primeira forma de controle de seleção (`se ... então ...`) e a primeira forma de controle de repetição (`enquanto ... faça ...`), as outras formas sempre podem ser reduzidas a estas duas. O motivo que justifica a existência dessas várias formas é a busca da naturalidade nas formas de expressão das linguagens de programação.

As formas gerais das estruturas de repetição estão descritas no Quadro 8.

Nas duas primeiras formas (repetição por teste lógico) a *expressão*, que define o elemento chave do controle, é uma expressão lógica (booleana) que pode ser obtida pela utilização de operadores relacionais ou operadores lógicos.

Na terceira forma (repetição por contagem) os parâmetros *inic*, *fim* e *p* devem ser de tipo compatível com o tipo da variável de controle *vc*.

Quadro 8: Estruturas de controle de repetição

linguagem algorítmica	linguagem C/C++
estrutura de controle de repetição com teste lógico *a priori*	
enquanto *expressão* **faça** *bloco;* *próxima instrução;*	**while(***expressão***){** *bloco;* **}** *próxima instrução;*
estrutura de controle de repetição com teste lógico *a posteriori*	
faça *bloco;* **enquanto** *expressão* **;** *próxima instrução;*	**do{** *bloco;* **}while(***expressão***);** *próxima instrução;*
estrutura de controle de repetição por variável contadora	
para ***vc*** **de** *inic* **até** *fim* **passo** *p* **faça** *bloco;* *próxima instrução;*	**for(*vc*=***inic***; vc<=***fim***; vc=vc+***p***){** *bloco;* **}** *próxima instrução;*

A instrução `for ...`, da linguagem C/C++, é mais flexível do que aquilo que foi descrito: seu primeiro argumento, que na descrição foi colocado como `vc=inic`, pode conter outras instruções; esse primeiro argumento será executado uma única vez antes da primeira execução do *bloco*; seu segundo argumento (na descrição: `vc<=fim`) é uma expressão lógica que será avaliada antes de cada execução do *bloco*, seu valor lógico (verdadeiro ou falso) é que determina a execução do *bloco* (verdadeiro) ou a finalização do processo de repetição (falso); o terceiro argumento, que assim como o primeiro pode conter várias instruções, será executado após cada execução do bloco que constitui o anel de repetição.

O tratamento de um problema computacional, cujo método de resolução envolva o emprego das estruturas de controle de seleção, exige do estudante a combinação de modelos lógicos mais complexos: ao buscar a construção do processo de resolução do problema, o aluno deve perceber a necessidade de estabelecer diferentes seqüências de ações que ficarão subordinadas à

ocorrência de alguns elementos próprios do cenário do problema e que deverão ser traduzidos pelos respectivos controles; a construção exige, em um primeiro momento, o reconhecimento de elementos que serão responsáveis por acionar as respectivas seqüências de ações, e depois a organização e a vinculação dessas seqüências aos elementos que deverão constituir os controles de seleção.

Na resolução do problema:

> O proprietário de um conjunto de salas comerciais fixou como vencimento de seus aluguéis o dia 15 de cada mês. Se o pagamento for efetuado antes do dia 9 é concedido um desconto de 8% mas se o pagamento ocorrer após o vencimento (dia 15) é cobrada multa de 20% e juros (simples) de 0,86% a cada dia de atraso. Conhecendo-se o valor original do aluguel e o dia (dentro do mês de vencimento) em que é feito o pagamento, como calcular o total a ser pago? (Adaptado de notas de aulas)

o aluno deve perceber o dia do pagamento como elemento que deverá compor os controles de seleção e estabelecer a vinculação entre as três alternativas – pagamento antes dia 9 **ou** pagamento entre o dia 9 e o dia 15 **ou** pagamento depois do dia 15 – e as ações correspondentes: realizar o desconto ou manter o valor original ou aplicar multa e juros. Já para a resolução do problema:

> A montagem de uma determinada placa de circuito eletrônico exige a utilização de três tipos de componentes: dois componentes do tipo A, três componentes do tipo B e sete componentes do tipo C. Conhecendo-se as quantidades disponíveis de cada tipo de componente, como determinar a quantidade máxima de placas que podem ser montadas? (MARTINS; RODRIGUES, 2008, p.75)

os elementos que devem definir o processo de seleção não aparecem prontos na proposta; tais elementos deverão ser produzidos por operações aritméticas (cálculo de quocientes) e depois utilizados para compor as estruturas de controle de seleção; nessa situação, o estudante deverá pensar nas quantidades de agrupamentos de cada tipo de componente e na seleção da menor dessas quantidades. Nesse caso, uma estratégia que pode auxiliar na construção do método geral de resolução é propor que o aluno resolva um caso particular do problema, antes de buscar a solução geral, por exemplo:

> A montagem de uma determinada placa de circuito eletrônico exige a utilização de três tipos de componentes: dois componentes do tipo A, três componentes do tipo B e sete componentes do tipo C. Se as quantidades disponíveis forem: 87 componentes de tipo A, 148 de tipo B e 315 componentes de tipo C, qual a quantidade máxima de placas que podem ser montadas?

O trabalho com um caso particular do problema pode sugerir ou favorecer a constituição do método de resolução geral.

A criação e organização de um processo que envolva o emprego das estruturas de controle de repetição também exigem do estudante a combinação de modelos lógicos complexos: ao buscar a construção do processo de resolução de um problema, o aluno tem que notar a necessidade de organizar uma determinada seqüência de ações que ficará sujeita a várias repetições e, ao mesmo tempo, configurar o controle desse processo. Por exemplo: ao tratar a resolução do problema, adaptado de Salvetti e Barbosa (1998, p.24):

> Uma criação de coelhos é iniciada com um casal de recém-nascidos. Sabendo-se que um casal torna-se adulto depois de dois meses do seu nascimento e que, a partir de então (inclusive), gera um novo casal a cada mês, e supondo-se que não ocorra qualquer morte, quantos meses são necessários para que a criação supere uma determinada quantidade **q** de casais?

é necessário que o aluno estabeleça as relações aritméticas que representem a "evolução", mês a mês, da criação, organize tais relações como uma seqüência que terá sua execução repetida por várias vezes, e constitua o elemento que deverá realizar o controle do processo de repetição.

Uma estratégia que pode favorecer a construção do método de resolução desse problema é propor a construção de um quadro que represente a evolução da criação de coelhos até o mês 7, conforme a Figura 2.

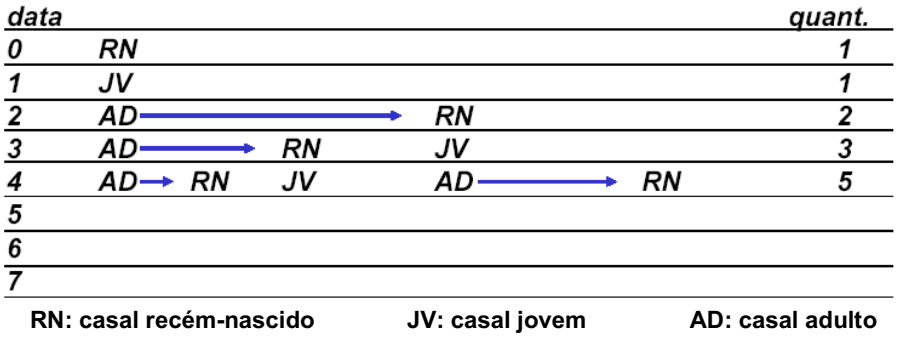


Figura 2 – Representação da evolução

Depois disso, propor que uma tabela com as quantidades de casais de coelhos seja completada, até o mês 10, conforme indicado na Figura 3.

data	RN	JV	AD	Total
0	1	0	0	1
1				
2				
3				
4				
...				

Figura 3 – Quantidades de casais

E ainda, explicitar e justificar as relações que foram empregadas para completar a tabela. Por exemplo:

- a quantidade de casais jovens num determinado mês é igual à quantidade de casais recém-nascidos do mês anterior;
- a quantidade de casais recém-nascidos num determinado mês é igual à quantidade de casais adultos do mesmo mês;
- a quantidade de casais adultos num determinado mês é igual à soma das quantidades de casais jovens e casais adultos do mês anterior.

A partir dessas tarefas iniciais, o estudante deve perceber que o mecanismo utilizado para completar o segundo quadro poderá ser convertido numa seqüência de instruções que definem a composição do bloco de ações cuja execução deverá ser repetida várias vezes. O controle desse processo de repetição será constituído com base na questão colocada na proposta do problema: *quantos meses são necessários para que a criação supere uma determinada quantidade* ***q*** *de casais?*

De forma geral, a elaboração de algoritmos ou programas com as estruturas de controle de seleção e de repetição depende, depois da compreensão da proposta do problema, da habilidade em combinar vários fatores: o estudante deve perceber a necessidade do uso das estruturas de controle, deve escolher ou construir os elementos que serão empregados como chaves nos controles, deve organizar os blocos de ações que ficarão sujeitos a esses controles e associar adequadamente controles e seqüências de instruções. Normalmente observa-se depois de concluído o esboço do método de resolução, já durante a elaboração do algoritmo ou programa, um processo de ajustes nessa construção, com alguns refinamentos ou mesmo modificações sobre o projeto esboçado, até que se obtenha o método pretendido.

1.5 Recursos da fase final da disciplina

Depois dos trabalhos com as formas básicas de fluxos de processamento (fluxos seqüenciais, fluxos alternativos e fluxos repetitivos) e com os tipos de dados primitivos, conforme a carga horária da disciplina e a grade curricular do curso, normalmente são introduzidos o conceito de modularização, que é essencial no paradigma de programação estruturada, e o conceito de tipos de dados agregados, que representa a possibilidade de tratamento de estruturas de dados mais complexas.

Para o conceito de modularização, a abordagem pode ser iniciada com a identificação de partes menores e mais específicas do problema a ser resolvido; depois dessa identificação inicial cada parte menor (“sub-problema”) é tratada e o resultado desse tratamento será um módulo auxiliar do algoritmo ou do programa: cada módulo auxiliar será responsável por resolver sua tarefa específica. Ao final será construído o módulo principal que deverá representar o método de resolução do problema proposto, a partir da articulação dos módulos auxiliares.

A introdução dos tipos de dados agregados permite a constituição de agrupamentos de dados homogêneos com organização seqüencial (listas

lineares) ou agrupamentos heterogêneos com organização em campos (registros); basicamente esses recursos trazem a possibilidade de organização e tratamento de quantidades maiores de dados.

Esses conceitos, modularização e tipos agregados, são novos degraus no processo de construção dos conhecimentos dos alunos. A experiência em sala de aula indica que, de forma geral, aqueles estudantes que alcançaram um bom domínio sobre os conceitos anteriores, conseguem transpor mais facilmente esses novos degraus; ao contrário, os estudantes que não tenham conseguido um domínio adequado sobre os recursos iniciais, raramente apresentam alguma evolução diante desses novos conceitos. Dessa forma, é importante buscar estratégias e recursos que possam favorecer o aprendizado durante as fases iniciais da disciplina, para depois trazer o foco para esses últimos estágios.

2 REVISÃO BIBLIOGRÁFICA

Alguns trabalhos acadêmicos indicam situações de dificuldade dos alunos, em disciplinas de introdução aos algoritmos e programação, e sugerem o emprego de algumas abordagens, estratégias ou ferramentas alternativas.

Em alguns desses trabalhos e também nas especificações de objetivos das disciplinas e nas descrições dos perfis profissionais, que são registradas em textos institucionais, como por exemplo, no documento Sociedade Brasileira de Computação (2003), surge de forma recorrente a noção de competência. A primeira seção deste capítulo é destinada a apresentar brevemente essa noção, e as outras são resultados de estudos e leituras com a finalidade conhecer análises e resultados desenvolvidos em trabalhos acadêmicos recentes. Este capítulo complementa a fase análises preliminares da Engenharia Didática.

2.1 A noção de competência

Perrenoud (1999, p.33) descreve a noção de competência como “uma capacidade de agir eficazmente em um determinado tipo de situação, apoiada em conhecimentos, mas sem limitar-se a eles”, e acrescenta que o tratamento de uma situação, ou a manifestação de uma competência, pode exigir a articulação e sinergia de vários recursos cognitivos complementares, entre os quais estão os conhecimentos.

O mesmo trabalho destaca a dificuldade de se encontrar uma definição clara e geral de competência, e admite que a palavra comporta significados variados. Dessa forma, produzir uma definição geral e abrangente é uma tarefa complexa e delicada, mas, por outro lado, é uma noção importante quando se trata sobre ensino e aprendizagem. Mesmo na ausência de uma definição que seja universal, o autor considera que a noção de competência deve ser construída.

Perrenoud (1999) desenvolve a noção de competência a partir de outros dois conceitos: hábito e esquema. O hábito é caracterizado como uma operação conhecida e dominada pelo sujeito que pode repeti-la, pronta e identicamente, sempre que necessária. O conceito de esquema é explicado da seguinte maneira:

> Em sua concepção piagetiana, o esquema, como estrutura invariante de uma operação ou de uma ação, não condena a uma repetição idêntica. Ao contrário, permite, por meio de acomodações menores, enfrentar uma variedade de situações de estrutura igual. (PERRENOUD, 1999, p.23).

Ainda segundo o autor, os esquemas são construídos pela prática, mas podem se apoiar em saberes ou teorias, e conservam-se no estado prático, mesmo que o sujeito não tenha uma consciência precisa de sua existência, de seu funcionamento ou de sua criação.

A noção de competência é expressa assim:

> [...] ela (uma competência) orquestra um conjunto de esquemas. Um esquema é uma totalidade construída, que sustenta uma ação ou operação única, enquanto uma competência com uma certa complexidade envolve diversos esquemas de percepção, pensamento, avaliação e ação, que suportam inferências, antecipações, transposições analógicas, generalizações, apreciação de probabilidades, estabelecimento de um diagnóstico a partir de um conjunto de índices, busca de informações pertinentes, formação de uma decisão, etc. (PERRENOUD, 1999, p.24)

As afirmações evidenciam a intenção do autor em destacar os conhecimentos como componentes de uma competência e, ao mesmo tempo, indicar que competência não se limita a eles.

Com essa visão, ao considerar-se a competência para a elaboração de algoritmos, pode-se apontar como um exemplo de esquema o par sintaxe-semântica das instruções da linguagem algorítmica empregada: o estudante deve dominar tal esquema (combinação adequada entre sintaxe e semântica), mas para a criação de um algoritmo outros elementos são requeridos.

Em outro trabalho do mesmo autor, o conceito de competência é colocado como

> [...] a aptidão para enfrentar uma família de situações análogas, mobilizando de uma forma correta, rápida, pertinente e criativa, múltiplos recursos cognitivos: saberes, capacidades, micro-competências, informações, valores, atitudes, esquemas de percepção, de avaliação e de raciocínio. (PERRENOUD, 2002, p.19)

Com esse entendimento, aflora a possibilidade de considerar-se competência como composição de micro-competências; essa visão mostra-se interessante quando se trata de criar atividades de ensino e aprendizagem, pois desdobrar uma competência em micro-competências pode facilitar a organização das atividades, a análise do desenvolvimento das mesmas e a avaliação dos resultados alcançados.

A composição da competência para a produção de algoritmos pode ser explicada a partir de outras competências menos complexas, ou seja: a partir de micro-competências, tais como: elaborar e vincular modelos abstratos a situações concretas, representar formalmente (linguagem algorítmica) o desenvolvimento de solução de um modelo, interpretar e validar a solução obtida, entre outras.

Perrenoud (1999) reforça a importância de diferenciar esquema e competência. Um esquema sustenta uma ação ou operação única que pode apresentar variação, mas que mantêm uma estrutura invariante. Uma competência engloba um repertório organizado de ações e operações possíveis de serem acionadas diante de situações análogas, mas variáveis, inclusive com aspectos inéditos, que se articulam em novas direções e além de pequenas acomodações ou variações. Uma competência se manifesta com o envolvimento de diversos componentes: percepção, pensamento, ação e avaliação, que sustentam inferências, antecipações, transposições analógicas, deduções, decisões.

Machado (2002) aborda a idéia de competência a partir de três características principais: a pessoalidade, o âmbito e a mobilização.

O autor explica a pessoalidade em função da existência do consenso tácito sobre a semântica da palavra *competência*, que admite atribuir competência a uma pessoa, mas não a um objeto ou artefato. Justifica tal fato com o argumento de que o conhecimento é sempre pessoal, ao considerar que cada pessoa, na

sua individualidade, constrói o seu significado para um conhecimento. Novamente fica evidente a forte relação entre competência e conhecimento.

O âmbito de uma competência é descrito pela referência ao contexto em que a competência se manifesta, o autor afirma:

> [...] as competências representam potenciais desenvolvidos sempre em contextos de relações disciplinares significativas, prefigurando ações a serem realizadas em determinado âmbito de atuação. (MACHADO, 2002, p.142)

Em outra parte do trabalho, o autor destaca a possibilidade de se traduzir uma competência como um feixe de habilidades, e caracteriza habilidade como uma forma de realização da competência em contextos mais específicos:

> [...] é como se as habilidades fossem micro-competências, ou como se as competências fossem macro-habilidades. (Machado, 2002, p.145)

Como exemplo é válido citar, em relação à competência para a criação de algoritmos, a habilidade (micro-competência) para relacionar um modelo abstrato a alguma representação formal de sua resolução.

O terceiro aspecto indicado por Machado (2002), a mobilização, é explicado pela associação entre competência e a mobilização de saberes: uma competência não é o acúmulo de conhecimentos, é a possibilidade de realizar uma ação, é a possibilidade de recorrer ao conhecimento disponível para realizar algo que se projeta. O autor afirma:

> [...] nenhum conhecimento deveria justificar-se como um fim em si mesmo: as pessoas é que contam, com seus anseios, com a diversidade de seus projetos. Assim como um dado nunca se transforma em informação se não houver uma pessoa que se interesse por ele, que o interprete e que lhe atribua um significado, todo o conhecimento do mundo “não vale um tostão furado” se não estiver a serviço da inteligência, ou seja, dos projetos das pessoas. (MACHADO, 2002, p.146)

A prática didática resultante das aplicações desses conceitos é denominada abordagem por competências (Perrenoud, 2002).

Uma abordagem por competências envolve: designar e escolher os alvos da aprendizagem vinculados às competências que se pretenda que o estudante desenvolva; planejar as atividades de ensino de tal forma que o eixo principal das mesmas seja definido pelas competências almejadas; planejar as atividades de avaliação de maneira que aquelas competências sejam os pólos principais. Tanto

o professor como o estudante devem ter clareza sobre aquilo que se coloca como metas a serem atingidas e também sobre os recursos, estratégias e práticas que irão compor os cenários de ensino e aprendizagem.

Os conhecimentos, objetos da aprendizagem, devem sofrer um processo de enraizamento, de tal forma que possam ser colocados em funcionamento de forma rápida e adequada, em situações diferentes daquelas em se processou a respectiva aprendizagem.

Perrenoud (1999) ressalta que a abordagem por competências exige tanto a preparação e a postura especial do professor em suas várias práticas, como atenção dirigida ao risco de não se estreitar a finalidade do ensino apenas como utilitarista.

2.2 Pesquisas recentes

A busca por trabalhos acadêmicos relacionados ao objeto desta pesquisa envolveu algumas fontes principais: os registros de eventos patrocinados pela Sociedade Brasileira de Computação, bancos de artigos científicos de algumas instituições de ensino e pesquisa, livros e artigos disponíveis na biblioteca eletrônica da *Association for Computing Machinery – ACM*. A quantidade de trabalhos encontrados, relacionados ao tema desta pesquisa, não é grande, e, ao menos nos eventos sob patrocínio da Sociedade Brasileira de Computação, há algum crescimento dessa quantidade.

Entre os artigos encontrados foram selecionados aqueles que mais se aproximavam do foco de interesse deste trabalho de pesquisa.

2.2.1 Uma proposta de abordagem por competências

O artigo de Delgado et al. (2005) identifica e analisa várias competências a serem desenvolvidas pelo estudante durante o curso de uma disciplina de introdução aos algoritmos e programação.

Na sua primeira parte, o trabalho trata da importância do tema, dos problemas de aprendizagem e suas conseqüências, da preocupação da Sociedade Brasileira de Computação (SBC) com a formação dos profissionais da Computação, e indica o movimento de pesquisadores na busca de análises e sugestões didáticas. Na seqüência, apresenta uma classificação dos diferentes tipos de competências e uma breve descrição da metodologia experimentada; são relacionadas, ainda, as competências envolvidas, as tarefas que podem ser propostas para atender ao desenvolvimento das competências identificadas, e então são apresentadas as conclusões.

O artigo, inicialmente, faz referência aos desafios na área de educação para a informática diante do processo acelerado e variado de evolução de novas tecnologias fundadas nos avanços da Ciência da Computação. Como reflexo desse cenário, o meio acadêmico ocupa-se com pesquisas sobre a formação superior dos profissionais para a atuação em computação: perfil profissional, perfil comportamental, definição de currículos e ementas, aplicação de metodologias e práticas didáticas.

O ensino de Algoritmos e Programação tem foco especial uma vez que essa matéria é componente essencial da maioria dos perfis relacionados à área de computação e, de acordo com os trabalhos de vários autores, é percebida pelos estudantes como obstáculo significativo e fonte de medos e frustrações que acabam por levar a reprovações, apatia, desgaste de auto-estima, desistência e abandono.

Delgado et al. (2005) apontam a necessidade de se trabalhar, no ambiente escolar, com abordagens mais adequadas para a aprendizagem desses assuntos. No Brasil, os eventos WEI–Workshop de Educação para a Informática e SBIE–Simpósio Brasileiro de Informática na Educação, patrocinados pela SBC, contaram com alguns trabalhos em que as propostas indicavam o emprego de ferramentas computacionais, a criação e aplicação de novas metodologias, ou ainda a integração de ferramentas e metodologias.

No trabalho de Delgado et al. (2004) há uma proposta de abordagem que foi experimentada em duas situações distintas: uma disciplina introdutória de Algoritmos e Programação (120 horas), em um curso de Ciência da Computação, e uma Oficina de Programação (30 horas), com uma turma de estudantes do

ensino médio. As experiências indicaram a confirmação de duas premissas contidas na abordagem proposta: a importância e a necessidade de práticas onde o papel do discente seja o de construtor de seu próprio conhecimento, e a necessidade de buscar-se a articulação entre as práticas de trabalho e a discussão dos saberes formais envolvidos como suporte a tais práticas.

Os experimentos apontaram também outros fatores: a aprendizagem baseada em conteúdos não pode suprir todas as necessidades que são próprias da matéria Algoritmos e Programação e, em relação à dimensão cognitiva, a teoria dos registros de representações semióticas (Duval, 2008) sobre o domínio de múltiplas representações e das conversões entre elas, desenvolvida no âmbito da Educação Matemática, pode ser vinculada à aprendizagem de algoritmos e programação.

O artigo de Delgado et al. (2005) é fruto da análise e avaliação desses experimentos. A reflexão sobre seus resultados levaram os autores à conclusão de que o processo de aprendizagem deve ser pautado pelo trabalho com problemas e projetos, com propostas de tarefas que desafiem os estudantes a mobilizarem seus conhecimentos e a complementá-los. Nesse cenário, a ação do professor deve ser conceber, refinar e regular situações de aprendizagem.

Segundo Delgado et al. (2004), a competência mais geral, estabelecida *a priori*, que é a leitura, a compreensão e a construção de algoritmos e de programas, pode ser desdobrada em sub-competências mais específicas e mais simples de serem enunciadas e compreendidas, o conjunto dessas sub-competências deve indicar a estruturação da disciplina, as práticas didáticas e os critérios de avaliação. Com essa forma de organizar a disciplina e as suas atividades, o processo de aprendizagem ganha flexibilidade, e o aluno passa a ter maior clareza sobre as metas especificadas e pode situar-se mais adequadamente no processo.

A terceira seção do trabalho de Delgado et al. (2005) traz uma descrição sucinta da abordagem proposta em Delgado et al. (2004) que organiza-se em três fases: resolução de problemas, com a valorização da autonomia cognitiva;
experiência da formalização, com a valorização da concisão e da precisão da linguagem empregada; e construção de algoritmos.

A primeira fase não trata de algoritmos ou programação, é trabalhada a resolução de problemas que envolvem essencialmente raciocínio lógico e aritmético. A finalidade é desenvolver a autonomia na busca de soluções próprias para os problemas propostos. Os elementos principais são a percepção e a interpretação do problema, a prática de analisar desafios variados e a reflexão para validar suas próprias soluções.

A segunda fase busca a aproximação com a formalização: inicialmente trabalha-se com as tentativas de formalização, no âmbito da língua natural, até que o grupo considere satisfatório o grau de formalização atingido, e depois trabalha-se a capacidade individual de julgamento sobre a adequação da formalização obtida em sua própria construção. Ao formalizar, o estudante deve focalizar o processo de resolução e não mais a solução em si.

A transição entre a segunda e a terceira fase é gradativa, a complexidade dos problemas é gradualmente incrementada e as soluções passam a exigir elaborações mais refinadas, nessa situação é exigido o ato de explicitar procedimentos. Nesse crescente são apresentados os conceitos de algoritmo e programa.

Como ganhos ou vantagens iniciais, o trabalho indica a consciência da necessidade de envolvimento com as atividades propostas, a postura mais interessada e ativa, a descoberta de técnicas simples como divisão em casos, minimização e refinamentos.

O trabalho de Delgado et al. (2005) identifica e desdobra as competências em sub-competências (micro-competências), um resumo da relação elaborada no trabalho pode ser descrito da seguinte maneira:

Sub-competências da fase “resolução de problemas”:

- compreender as relações entre diversas estruturas abstratas e suas representações;
- analisar e modelar situações;
- interpretar e avaliar a solução do modelo no cenário original.

Sub-competências da fase “formalização”:

- compreender o valor descritivo das várias representações;
- descrever processos na forma oral e escrita;
- identificar a existência de diversos níveis de formalização;

- mapear as relações entre diversas estruturas abstratas e suas representações de forma primária e intuitiva.

Sub-competências da fase “construção de algoritmos”:

- compreender procedimentos expressos em modelos computacionais;
- dominar o funcionamento das estruturas simbólicas formais para a construção de procedimentos computacionais;
- formular o conceito de programa (situado no universo da prática existente);
- avaliar o grau de similaridade entre suas formulações e as novas experiências.

Com a abordagem por competências, Delgado et al. (2005) afirmam ter obtido os seguintes benefícios:

- As competências podem ser detalhadas em sub-competências, e estas organizadas das mais elementares às mais complexas. Essa organização facilita a postura do aluno para superar os desafios e alcançar a competência mais elaborada.
- A simplificação da competência a ser atingida facilita, para o aluno, a identificação, a avaliação e a valorização de suas possibilidades e deficiências; ele reconhece que é necessário o seu comprometimento com o aprendizado de coisas simples, uma de cada vez, para alcançar a competência almejada.
- A autonomia é alcançada com mais rapidez a partir da segurança percebida pelo aluno ao acompanhar seus progressos no mecanismo de construção do conhecimento.
- Aparentemente, ocorre melhoria na auto-estima, pois o aluno passa a contar com parâmetros mais claros para situar-se em termos do que se espera dele e daquilo que ele já consegue realizar.
- Melhora no relacionamento pessoal, uma vez que as situações de negociação e de cooperação são colocadas e valorizadas.

2.2.2 Trabalhos com a introdução da idéia de *papéis de variáveis*

Os artigos Sajaniemi e Kuittinem (2003) e Sajaniemi (2005) analisam o emprego de um sistema de animação de programas que incorpora a noção de *papel de variável* em disciplinas de introdução à programação. Os autores justificam o foco de interesse em função das dificuldades que freqüentemente os estudantes apresentam nessas matérias.

Os trabalhos destacam a adequação das ferramentas de visualização e animação, associadas aos mecanismos de funcionamento de cada programa e não aos elementos gerais da linguagem de programação. Os autores exemplificam tal adequação com uma situação semelhante ao que se tem no seguinte programa:

```
program dobros;
var quantidade, valor, dobro: integer;
begin
   repeat
      write('digite a quantidade: '); readln(quantidade);
   until quantidade>0;
   while quantidade>0 do
   begin
      write('digite o valor: '); readln(valor);
      dobro=2*valor;
      write('o dobro de ', valor, ' vale ', dobro);
      quantidade=quantidade-1;
   end;
end.
```

Nesse exemplo, o confronto **`quantidade>0`** aparece duas vezes, e do ponto de vista dos elementos da linguagem de programação são idênticos – a expressão lógica **`quantidade>0`** resulta ou verdadeiro ou falso, conforme o conteúdo de **`quantidade`** seja maior do zero ou não.

No âmbito do programa, as duas expressões carregam significados diferentes: a primeira controla (aceita ou rejeita) a entrada de um valor pelo usuário, e a segunda estabelece o controle de uma estrutura de repetição por um processo de contagem. Na parte inicial do programa não é possível prever como ocorre a evolução dos conteúdos da variável **`quantidade`**, essa evolução ocorre

conforme sejam as entradas realizadas pelo usuário até que um valor maior do que zero seja digitado. Na segunda parte, a evolução dos conteúdos ocorre de forma previsível: uma sucessão de valores inteiros consecutivos em ordem decrescente até atingir o valor zero.

Os artigos de Sajaniemi (2005) e Sajaniemi e Kuittinem (2003) definem *papel de variável* como o conjunto de características dinâmicas da variável, correspondente à evolução de seus sucessivos valores, e às suas relações com outras variáveis ou eventos externos.

Sajaniemi (2005) desenvolve uma classificação que organiza dez categorias gerais de papéis de variáveis, e que são suficientes para a classificação da quase totalidade das variáveis empregadas em programas tratados por disciplinas introdutórias de programação. Para cada uma das dez categorias, estabelece uma forma visual de representação que é empregada no sistema de animação de programas.

A primeira categoria "*fixed value*" acomoda as variáveis cujos valores não são modificados após terem sido definidos durante uma sessão de execução do programa. A forma visual associada a essa categoria, utilizada no ambiente de animação de programas, é uma pedra pesada lapidada com o valor do conteúdo da variável representada; a idéia é indicar na figura a dificuldade para se ter a modificação do conteúdo da variável.

Na segunda categoria "*stepper*" estão as variáveis cujos conteúdos componham uma sucessão previsível de valores, tal sucessão pode depender de valores de outras variáveis, por exemplo: de variáveis que limitem um processo de contagem. A representação visual para essa categoria é uma seqüência de pegadas (pés humanos), com destaque para o estado da variável a cada momento e a indicação de alguns valores anteriores e também dos próximos. A intenção é caracterizar o caminhar por um trajeto estabelecido.

Uma variável da terceira categoria, denominada "*follower*", será utilizada quando houver a necessidade de se reter o conteúdo de outra variável que sofrerá, em seguida, alguma alteração. Assim, "*follower*" é uma variável que assume uma sucessão de valores especificados pelos conteúdos de outra variável que será atualizada imediatamente depois de ter redefinido a variável

“*follower*”. A representação visual dessa categoria é feita com a figura da cabeça de um cão de caça que aponta seu alvo. A idéia é representar a variável “*follower*” em perseguição à outra variável.

A quarta categoria “*most-recent holder*” corresponde à necessidade de se armazenar o valor mais recente ao se percorrer uma sucessão de vários valores ou ao se realizar uma sucessão de várias entradas. Sua representação visual é elaborada por um par de quadros retangulares, um com o valor atualizado e o outro com o valor anterior da variável. O maior destaque para o quadro com o valor atualizado é dado em função das cores de fundo.

A quinta categoria “*most-wanted holder*” corresponde à necessidade de se armazenar o “melhor valor” ao se percorrer uma sucessão de vários valores ou ao se realizar uma sucessão de várias entradas; o critério para definir o que é o “melhor valor” varia conforme o cenário da situação tratada: em um cenário o “melhor valor” pode ser o maior entre todos os valores de uma sucessão, em outro pode ser o nome do vendedor com menor tempo de serviço na equipe de vendas de uma empresa. A representação é feita com duas flores de cores diferentes: uma de cor viva para o valor atual (o melhor até aquele momento) e outra de cor cinza para o valor anterior (o melhor até o momento anterior).

Uma variável da sexta categoria, denominada “*gatherer*”, será utilizada para acumular algum efeito que se observa ao percorrer uma sucessão de vários valores; cada novo conteúdo de uma variável “*gatherer*” é definido por uma combinação de seu valor anterior com algum novo dado. Por exemplo: obter o produto de vários fatores que serão digitados pelo usuário. A representação de uma variável “*gatherer*” é feita com a figura de uma caixa com tampa, com as indicações do valor mais recente e do valor anterior. A idéia é constituir ou reconstituir o conteúdo da caixa a cada combinação do conteúdo anterior com algum novo dado.

A categoria “*transformation*” representa variáveis cujos conteúdos são obtidos sempre da mesma maneira a partir de valores de alguma outra variável. Por exemplo: uma medida em polegadas obtida a partir da respectiva medida em metros. A representação visual indica a vinculação entre as duas variáveis por uma seta.

A oitava categoria é denominada "*one-way flag*", e indica a utilização de uma variável cujo conteúdo é um entre dois valores possíveis, e que terá seu valor redefinido apenas uma vez. A representação é feita por uma lâmpada acesa que mostra em seu corpo o valor inicial e que se quebra no momento em que aquele valor inicial é modificado.

A nona categoria, "*temporary*", indica o emprego de alguma variável que deverá armazenar um dado por um período de tempo muito pequeno, tal dado pode ser o conteúdo de outra variável ou uma entrada. O exemplo típico é o processo de permuta de conteúdos de duas variáveis. A representação é o facho de uma lanterna que permanece acesa apenas enquanto a variável cumpre seu papel de armazenamento temporário.

A última categoria "*organizer*" corresponde à utilização de uma lista de vários elementos (vetor unidimensional) cujos valores inicialmente definidos podem ter suas disposições modificadas, ou seja: os valores iniciais não são alterados, mas a disposição dos mesmos na lista pode ser modificada.

O trabalho de Sajaniemi (2005) indica que a classificação proposta pode ser estendida para variáveis de tipos estruturados, para parâmetros de funções ou métodos auxiliares e para variáveis de tipo ponteiro (o tipo ponteiro é um tipo particular que representa a possibilidade de tratar os endereços de variáveis); o artigo apresenta também uma discussão sobre a possibilidade de ocorrer mudança do papel de uma variável durante a execução do programa.

As características observadas para a elaboração das categorias de papéis de variáveis são vinculadas à funcionalidade das mesmas no âmbito do programa ou do algoritmo em que são empregadas, e não aos elementos gerais da linguagem utilizada para a construção dos algoritmos e dos programas. Além disso, o ato de classificar uma variável pode ser influenciado pelos conhecimentos ou pelas vivências de quem faz a classificação. Os autores descrevem, como exemplo, a classificação de uma variável que represente os elementos da seqüência de Fibonacci, assim: para uma pessoa que conheça a seqüência, parece mais natural classificar a variável como "*stepper*", enquanto que para quem não tenha tal conhecimento a classificação seria de uma variável "*gatherer*".

O artigo de Sajaniemi e Kuittinem (2003) relata uma experiência de introdução da noção de *papel de variável* com três grupos de estudantes em uma disciplina de introdução à programação. Os grupos foram denominados por *grupo tradicional*, *grupo dos papéis* e *grupo da animação*.

A noção de *papel de variável* foi introduzida nos trabalhos com o segundo e o terceiro grupos, e não nos trabalhos com o primeiro grupo. O sistema de animação foi empregado nos trabalhos do terceiro grupo, enquanto os dois primeiros utilizaram como recurso para visualização, apenas as ferramentas do ambiente de programação (Turbo Pascal).

A experiência foi conduzida em cinco semanas com quatro horas de aulas e duas horas de exercícios a cada semana. Nas últimas quatro sessões de exercícios, os grupos trabalharam com a implementação de quatro programas, um programa em cada sessão de exercício.

Sajaniemi e Kuittinem (2003) relatam, como resultado, que a introdução da noção de *papel de variável* favorece a habilidade para a leitura e para a elaboração de programas, além disso, indicam que o emprego do sistema de animação que contempla a representação dos papéis de variáveis, modifica a forma com que os alunos descrevem programas: ocorre o destaque para as relações entre os dados e a estrutura do programa, o que não acontece quando a animação é trabalhada sem a inclusão daquela noção. Indicam também o maior envolvimento dos estudantes do grupo da animação. Os alunos consideraram simples operar o sistema de animação, enquanto os alunos dos outros grupos tiveram dificuldades para operar as ferramentas de depuração do ambiente de programação.

O trabalho de Sajaniemi (2005) apresenta orientações e sugestões sobre as formas de introduzir os papéis de variáveis em uma disciplina inicial de programação, apresenta também algumas reflexões sobre como a noção de papeis é percebida por programadores profissionais e por professores que trabalham em disciplinas de introdução à programação.

No trabalho de Laakso et al. (2008) é apresentado um ambiente para a prática de depuração de programas, que tem como um dos apoios o conceito de *papel de variável*, em uma disciplina inicial de introdução à programação.

Laakso et al. (2008) relacionam algumas abordagens que podem ser consideradas para analisar como um estudante entende um programa.

A primeira, denominada *abordagem funcional*, é construída a partir da hipótese de que entender um programa significa elaborar, ativar e instanciar blocos de conhecimento que correspondem aos conceitos gerais de programação. Assim, o entendimento de um programa envolve: ativar conhecimentos já elaborados, extrair referências do texto do programa e inferir informações a partir dos conhecimentos ativados.

A segunda, *abordagem estrutural*, explica o entendimento de um programa em função da percepção de como se organiza a estrutura do mesmo, com suas instruções de controle e suas funcionalidades. O aspecto dominante é a estrutura do programa.

A terceira abordagem, denominada *modelos mentais*, destaca dois campos de entendimento: o *modelo do programa*, que se relaciona com a estrutura do programa, e o *modelo da situação* que é relacionado ao domínio do problema. Nessa abordagem, entender o programa é articular o modelo do programa e o modelo da situação.

A quarta abordagem coloca em foco o processo de seleção de informações ao se fazer a leitura de um programa; essa leitura não é seqüencial, são realizados avanços e retrocessos em função dos aspectos que se busca entender.

Os autores indicam que a abordagem *modelos mentais*, para o caso de depuração de um programa, é a mais adequada, pois devem ser favorecidas as ligações entre o modelo do programa e o modelo da situação. Um dos meios para favorecer tais ligações é incorporar o conceito de papéis de variáveis e explorar as características que definem cada papel para aprofundar o entendimento do programa e facilitar sua depuração.

O artigo de Laakso et al. (2008) descreve uma ferramenta visual para suporte a atividades de depuração de programas, e propõe sua utilização em matérias de introdução à programação com o seguinte objetivo: favorecer o entendimento dos mecanismos presentes na execução de um programa e assim apoiar o aprendizado de programação.

2.2.3 Outros estudos recentes

Outros trabalhos registram relatos de preocupações com a aprendizagem de algoritmos e propostas de adaptações de abordagens, com a finalidade de favorecer o processo de aprendizagem. Algumas declarações que destacam a situação de dificuldade:

> Durante anos, os especialistas têm experimentado várias abordagens no ensino dessa disciplina, considerada como de difícil apreensão pelos alunos, uma vez que não é suficiente apresentar um algoritmo para que o aluno seja capaz de criar outro, em situações parecidas. (CAROLI, A.J. na apresentação da obra de BARBOSA, 2001, p.9)

> Participando do processo ensino-aprendizagem de desenvolvimento de algoritmos por dez anos, pudemos constatar a enorme dificuldade experimentada pelos estudantes. Essas dificuldades não são observadas somente por nós, como revela o texto de Manber – 1989, em que o autor afirma que os estudantes têm dificuldades em obter soluções e em compreender as soluções apresentadas. (BARBOSA, 2001, p.13)

> A criação de ambientes que apoiem esse aprendizado é de grande interesse, já que o processo de construção do conhecimento necessário à produção de algoritmos para a programação constitui uma árdua tarefa ao aprendiz. (BERCHT; FERREIRA; SILVEIRA, 2005, p.2)

Barbosa (2001) tomou como foco principal de seu trabalho a forte vinculação entre as atividades de aprendizagem e os sistemas de representação semiótica. Considerou como referencial de seu trabalho a teoria de Registros de representação semiótica (MACHADO, 2008a) que, em uma de suas bases fundamentais sobre o funcionamento cognitivo do pensamento humano, indica não haver apreensão conceitual sem a coordenação de vários sistemas de representação semiótica.

Nas situações planejadas para suas observações, Barbosa (2001) estabeleceu como objetivo a análise de representações produzidas pelos alunos com utilização de registros em língua natural, em comparação com as respectivas representações quando utilizada uma linguagem algorítmica (pseudocódigo); em seu trabalho, a autora, constituiu uma seqüência de atividades (Engenharia Didática) cuja experimentação foi a fonte de seus dados. Os resultados de suas observações revelaram não haver, em geral, uma congruência entre as duas formas de representação: o registro em língua natural e o registro em pseudocódigo.

Barbosa (2001) declara que atos cognitivos, como apreensão conceitual de um objeto, discriminação de uma diferença, compreensão de uma inferência, não ocorrem sem a produção de pelo menos dois registros de representação semiótica para o objeto em estudo.

O trabalho de Pereira Júnior e Rapkiewicz (2005) indica a importância de se articular estratégias e ferramentas de apoio ao aprendizado de algoritmos:

> Destaca-se em vários trabalhos e relatos de professores e alunos que o processo de ensino e aprendizagem de algoritmos e programação apresenta um conjunto de dificuldades que têm origem em diversas vertentes e razões. Na análise da literatura constata-se a existência de três vertentes: ferramentas, estratégias e a união de ambas. O que se averigua é a obtenção de melhores resultados com aplicação da última vertente, a união de ferramentas e estratégias, sob a forma de redução do índice de evasão e repetência. A partir desse resultado foi proposta a arquitetura de um ambiente de aprendizagem que se fundamenta e guia uma estratégia, o que representa seu diferencial em relação a muitas outras ferramentas, uma vez que em grande parte visam somente à execução de programas. (PEREIRA JÚNIOR; RAPKIEWICZ, 2005, p.6)

O artigo de Garcia, Rezende e Calheiros (1996) propõe a utilização, em situações didáticas, de um ambiente para suporte ao entendimento e à implementação de estruturas de dados e algoritmos com o emprego de animações gráficas. Esse ambiente permite não apenas a participação passiva do estudante, com a gravação de eventos anteriores à participação, mas também, em certo grau, a participação ativa e construtiva a partir de recursos mínimos e padronizados para a criação de animações.

A utilização desse ambiente levou a um melhor desempenho dos alunos segundo o relato dos autores. Assim, a presença das animações gráficas das estruturas de dados no ambiente é apontada como um dos fatores que contribuíram para a melhoria de aproveitamento dos alunos. No trabalho encontra-se a afirmação:

> Do mesmo modo que a experimentação durante a operação das animações enriquece o aprendizado mais do que a mera observação passiva delas, é de se esperar que, com um sistema onde a própria implementação das animações gráficas é facilitada a ponto de poder ser realizada pelo estudante, a absorção do funcionamento dos algoritmos seja ainda mais intensa. (GARCIA; REZENDE; CALHEIROS, 1996, p.1)

Pimentel e Omar (2008) vinculam especificamente a noção dinâmica, presente em um programa ou em um algoritmo, às dificuldades dos estudantes; entre as fontes dessas dificuldades relacionam:

> [...] o elevado nível de abstração envolvido; as metodologias tradicionais de ensino, que privilegiam a aprendizagem de conceitos dinâmicos utilizando principalmente abordagens e materiais de natureza estática; [...] (PIMENTEL; OMAR, 2008, p.3)

As conclusões e os resultados dos trabalhos relacionados reforçam os motivos e as justificativas para a elaboração desta pesquisa, e indicam algumas possibilidades de referenciais teóricos, como a noção de competência (Perrenoud, 2002) e a teoria dos Registros de representação semiótica (Duval, 2008). Os trabalhos apontam também a relevância da noção de processo dinâmico presente em algoritmos e programas, nos planejamentos de atividades de ensino.

As obras de Barbosa (2001) e Delgado et al. (2005) apontam o potencial da teoria dos Registros de representação semiótica para a constituição das bases teóricas desta investigação.

O suporte das noções da abordagem por competências, empregado nos trabalhos de Delgado et al. (2005), contribui nesta pesquisa com o sentido de oferecer uma visão mais abrangente sobre o processo de aprendizagem e seus propósitos.

A idéia sobre os papéis de variáveis, contida nos artigos de Sajaniemi e Kuittinem (2003), Sajaniemi (2005) e Laakso et al. (2008), confirmam a importância que se deve atribuir aos aspectos dinâmicos presentes nos algoritmos e nos programas. Toda a construção da classificação dos papéis de variáveis envolve, em seus critérios e também na sua exploração, os aspectos de mecanismo dinâmico relacionado ao comportamento das variáveis.

A adequação de se empregar ferramentas e estratégias para apoiar as atividades de aprendizagem de elaboração de algoritmos é indicada nos artigos de Bercht, Ferreira e Silveira (2005) e Pereira Júnior e Rapkiewicz (2005).

Os artigos Pimentel e Omar (2008) e Garcia, Rezende e Calheiros (1996), apontam diretamente a relevância dos aspectos dinâmicos diante das representações estáticas de processos representados nos algoritmos e nos programas.

3 FUNDAMENTOS TEÓRICOS E METODOLÓGICOS

3.1 Registros de representação semiótica

A atividade de tratamento de problemas computacionais é o foco central de interesse em disciplinas de Desenvolvimento de Algoritmos e Programação. Tal atividade configura-se como uma composição de algumas passagens por registros de representação de um processo, que traduz o método de resolução do problema.

A composição envolve formas intermediárias variadas, e o produto final da atividade é a representação de um método de resolução do problema computacional, descrito em uma linguagem algorítmica ou em uma linguagem de programação.

A criação e a elaboração do método de resolução envolvem abstrações, de maneira similar ao que ocorre com os objetos matemáticos, e assim exigem que o estudante, ou o professor, se utilize de alguma forma de registro de representação semiótica para que se possa acessá-lo, seja para o seu entendimento, para sua discussão ou para sua construção.

Duval (2008) descreve as noções que constituem a teoria dos registros de representação semiótica, concebida para buscar o entendimento do processo de aprendizagem de objetos matemáticos, mas que pode ser empregada em outros cenários, como no aprendizado para construção de algoritmos.

O desenvolvimento de algoritmos e programa ocorre com o emprego de duas formas de registros de representação específicas, que são os registros em

linguagem algorítmica e em linguagem de programação. Por outro lado, o desenvolvimento envolve também o emprego de registros em língua natural e registros algébricos, principalmente nas fases iniciais das construções, além dos registros informais (esboços e figuras) revelados principalmente nos atos de concepção dos processos.

O estudante, para revelar que sabe como resolver o problema, deve elaborar a construção de um método com o emprego de um registro de representação. Quando o estudante declara “esse é o processo de resolução”, ou seja, quando o estudante já criou para si o método de resolução, certamente o que ele exibe é um registro de representação de tal processo, para isso o estudante pode lançar mão dos recursos da língua natural, de esboços com a combinação de recursos gráficos e da escrita, de registros em linguagem algorítmica ou em linguagem de programação.

Durante a fase de descoberta e primeiro esboço do método de resolução, o estudante articula as ferramentas de que dispõe: conhecimentos anteriores relativos à especificidade do problema em tratamento, conceitos matemáticos, organização e argumentação lógica. Nessa fase, normalmente a representação é elaborada com a combinação de recursos verbais, recursos da escrita, rascunhos ou figuras, e pode envolver expressões algébricas, desenhos, esquemas, expressões lógicas. A garantia de que o esboço representa um processo correto ou adequado é obtida pelo encadeamento de justificativas que se assemelha ao que é feito na elaboração de uma demonstração de alguma proposição matemática.

Nessa fase inicial, que envolve também o entendimento da proposta do problema, o estudante conta com liberdade na escolha do registro de representação; normalmente o escolhido é aquele com o qual ele tem mais facilidade ao trabalhar ou aquele que acomode bem os aspectos oriundos do próprio problema ou do encaminhamento da construção. Qualquer que seja a forma inicial escolhida, o estudante já deve ter em mente as fases seguintes, quando deverá produzir a representação do processo com o emprego de uma linguagem algorítmica ou uma linguagem de programação.

Por exemplo: ao abordar o problema dos parafusos, o estudante poderá imaginar um cenário concreto, com vários parafusos em um recipiente de um lado

e, de outro, várias caixas grandes e várias caixas pequenas, com as capacidades especificadas na proposta (as grandes comportam 40 parafusos cada uma e as pequenas 10 parafusos), e então pensar em agrupar os parafusos em lotes de 40 e dispor sucessivamente esses lotes nas caixas grandes, até que a quantidade remanescente de parafusos seja insuficiente para completar um lote de 40. A partir daí os agrupamentos devem ser em lotes de 10 parafusos que serão dispostos em caixas pequenas, até que a sobra seja menor do que 10 (insuficiente para completar uma caixa pequena). Depois disso as caixas utilizadas e a sobra de parafusos seriam contabilizadas para obter-se a resposta solicitada.

Outra possibilidade é o estudante imaginar um caso específico do problema e se perguntar, por exemplo: “se forem 264 parafusos a embalar, quais serão as respostas?”. Nessa situação o aluno poderia pensar em realizar subtrações sucessivas, primeiro com o preenchimento das caixas grandes e depois das caixas pequenas, assim:

264-40-40-40-40-40-40=24 → 6 caixas grandes, sobram 24

24-10-10=4 → 2 caixas pequenas, sobram 4 parafusos.

Essa segunda possibilidade poderia servir também para o aluno certificar-se do entendimento da proposta do problema, ou ainda para complementar a primeira forma de esboço.

O próximo passo seria ajustar o mecanismo já delineado aos recursos que a linguagem algorítmica ou linguagem de programação oferecem. Assim, planejar duas estruturas de controle de repetição com o seguinte esboço: “enquanto a quantidade de parafusos for suficiente para completar uma caixa grande subtrair sucessivamente 40 dessa quantidade, e depois, enquanto a quantidade for suficiente para completar uma caixa pequena, subtrair 10 dessa quantidade; além disso, contabilizar cada caixa grande e cada caixa pequena que for utilizada, ou seja, contabilizar a cada subtração, uma caixa grande ou uma caixa pequena.”

Outra possibilidade é planejar operações de divisão no lugar das subtrações sucessivas, assim: “dividir a quantidade de parafusos por 40, o quociente será a quantidade de caixas grandes e o resto dessa divisão é a parte do lote inicial de parafusos que deverão ser dispostos em caixas pequenas;

depois, dividir o resto da divisão anterior por 10, o quociente será a quantidade de caixas pequenas e o resto será a sobra de parafusos (quantidade insuficiente para completar uma caixa pequena).”

Depois dessa etapa intermediária de planejamento, o trabalho concentra-se na elaboração da descrição do algoritmo ou do programa; o projeto delineado até então deve ser refinado e descrito a partir dos recursos da linguagem algorítmica ou da linguagem de programação. Os algoritmos correspondentes estão descritos nos Quadros 9 e 10.

Quadro 9: Algoritmo – problema dos parafusos com subtrações sucessivas

```
embalarparafusos( )
leia(quantparaf);
qcgrandes←0; qcpequenas←0;
enquanto quantparaf≥40 faça
   quantparaf←quantparaf-40;
   qcgrandes←qcgrandes+1;
enquanto quantparaf≥10 faça
   quantparaf←quantparaf-10;
   qcpequenas←qcpequenas+1;
imprima(qcgrandes);
imprima(qcpequenas);
imprima(quantparaf);
```

Quadro 10: Algoritmo – problema dos parafusos com divisões

```
embalarparafusos( )
leia(quantparaf);
qcgrandes←quantparaf div 40;
sobra←quantparaf mod 40;
qcpequenas←sobra div 10;
sobra←sobra mod 10;
imprima(qcgrandes);
imprima(qcpequenas);
imprima(quantparaf);
```

Essa breve exposição evidencia que as abstrações são necessárias e que as construções envolvidas demandam o trânsito entre alguns de registro de representação, assim é natural considerar-se a teoria dos registros de representação semiótica como um suporte deste trabalho.

Por outro lado, o artigo de Colombo, Flores e Moretti (2008), que investigou o emprego dessa teoria como base teórica no desenvolvimento de dissertações e teses em programas de pós-graduação em Educação Matemática, no período de

1990 a 2005, indica o crescimento da utilização, a abrangência que a teoria oferece e destaca a relevância e a importância de seu emprego nas explorações da Educação Matemática.

Um primeiro pressuposto da teoria é relativo à forma de perceber o estudante como sujeito que atua no processo de seu aprendizado, assim: o estudante constitui seus conhecimentos a partir de interações em um ambiente complexo que envolve o professor, as informações, os objetos da aprendizagem, o grupo de alunos, a linguagem.

Outro pressuposto é a abordagem cognitiva que se justifica pela exigência de se buscar uma formação inicial em matemática mais apurada, com o sentido de atender às necessidades impostas pelas características dos meios informáticos e tecnológicos que se colocam no cotidiano. A abordagem cognitiva se justifica também como um meio a contribuir para o aprimoramento das ferramentas gerais de raciocínio, análise e visualização que o estudante deve desenvolver desde o início de sua formação.

A proposta de Duval (2008), para a teoria dos registros de representação semiótica, define a particularidade do desenvolvimento cognitivo requerido no campo da matemática, a partir de características que lhes são próprias: a importância fundamental das representações semióticas como meios de acesso aos conceitos abstratos, e a exigência do emprego de uma extensa variedade dessas representações: língua natural, figuras geométricas, gráficos, sistemas de numeração, expressões algébricas.

Algumas dessas características se repetem nas situações de tratamento de problemas computacionais, pois é exigida a realização de abstrações – as informações que compõem o cenário do problema devem ser modeladas e representadas por variáveis do algoritmo ou do programa; e a manipulação dos dados, representações das informações reais, deve ser referenciada por operações internas ao sistema computacional cuja organização representa um método de resolução do problema.

Duval (2008), ao descrever os aspectos dos registros de representação semiótica, aponta duas categorias principais de formas de trabalho com os registros: tratamento e conversão. As atividades de tratamento são internas ao

registro de representação, ou seja, tais operações ocorrem num mesmo sistema de representação, sem a exigência de se utilizar outros registros; as atividades de conversão são caracterizadas justamente pela transformação de uma forma de registro em outra, empregando-se, portanto, mais de um sistema de representação.

Os trabalhos de desenvolvimento de um algoritmo envolvem tanto tratamentos como conversões de registros; os tratamentos ocorrem, por exemplo, nos momentos em que se procede a alguma pequena modificação no texto de um algoritmo, com o objetivo de corrigi-lo ou de ajustar sua finalidade; as conversões podem ser apontadas nos momentos em que o planejamento inicial já foi delineado (língua natural, figuras, esboços) e então se faz a construção do texto do algoritmo (linguagem algorítmica) ou do texto do programa (linguagem de programação).

Algumas vezes as conversões ocorrem já nos primeiros contatos com o problema, a partir da leitura de seu enunciado; esses trabalhos ocorrem muitas vezes em ciclos e com sobreposições, pois no momento em que se descreve o texto do algoritmo pode ser necessária a retomada e reformulação do plano inicial em função de se perceber alguma característica que até então não havia sido notada ou tratada adequadamente. É freqüente também observar a tentativa do estudante em iniciar a construção do texto do algoritmo logo em seguida ao primeiro contato com a proposta do problema, e nesses casos é mais evidente a necessidade de idas e vindas do trabalho de construção do texto do algoritmo para a atividade de planejamento e vice-versa.

Mesmo as tarefas que são aparentemente de tratamento, algumas vezes, podem exigir também alguma atividade de conversão. A modificação de uma instrução no algoritmo pode prescindir de uma argumentação que não será expressa propriamente no texto do algoritmo, mas será articulada de alguma forma, seja com a finalidade de comunicação, seja com o objetivo de uma construção mental.

Pode-se imaginar, por exemplo, que o aluno, ao construir o texto do algoritmo para o problema dos parafusos, tenha se enganado ao descrever a expressão de controle na instrução de repetição, em vez de `enquanto`

`quantparaf≥40 faça` tenha escrito `enquanto quantparaf>40 faça`; para perceber a falha e corrigir a expressão, o estudante deverá relacionar o confronto `quantparaf>40` aos aspectos concretos que a expressão representa, e estabelecer uma fala ou um raciocínio semelhante ao seguinte: “se houver ainda exatamente 40 parafusos, mais uma caixa grande poderá ser completada, assim, se a quantidade de parafusos for igual a 40, o mecanismo das atribuições (`quantparaf←quantparaf-40` e `qcgrandes←qcgrandes+1`) ainda deve ser repetido mais uma vez.”

Como exemplo de uma atividade de tratamento estrito é válido citar a seguinte situação: o aluno pode se decidir por trocar a atribuição `qcgrandes←qcgrandes+1` pela aplicação do operador de incremento unitário `qcgrandes++`, que tem efeito idêntico ao da atribuição; para essa modificação não é relevante o sentido ou o objetivo da instrução, não é essencial que se pense sobre o significado correspondente.

As conversões são aquelas atividades que exigem a manipulação simultânea ou conjugada de pelo menos duas formas de registros de representação do objeto.

O processo da resolução de um problema computacional passa por pelo menos três atividades de conversão: a primeira ocorre a partir da leitura do enunciado do problema e o início do esboço do método de resolução, a segunda é a produção do texto do algoritmo a partir do plano de solução delineado, e a terceira é a produção do texto com o código do programa a partir do algoritmo. Vale observar que os registros produzidos na segunda e na terceira conversões são objetos estáticos (textos) que representam um processo dinâmico de manipulação e transformação de dados.

A terceira conversão envolve o trabalho com dois sistemas de representação que são próximos: a linguagem algorítmica e a linguagem de programação. A concepção da linguagem algorítmica, apesar de independente da linguagem de programação, considera as características de operação do sistema computacional, e esse sentido leva a linguagem algorítmica a oferecer um conjunto de recursos que é possível de se traduzir em termos dos elementos da linguagem de programação.

A elaboração do texto de um programa, construído a partir do texto de um algoritmo, corresponde ao trabalho de uma tradução de cada instrução do algoritmo para uma instrução do programa, com acréscimos que representam os detalhes de interface do sistema computacional com o ambiente, cujos pormenores não são tratados nos algoritmos, e mais alguns elementos que são próprios da linguagem de programação ou do ambiente de programação utilizado.

Nessa atividade de conversão (algoritmo→programa) os estudantes não enfrentam grandes dificuldades, e depois de algum tempo de vivência, a prática é realizada quase que mecanicamente. Essa característica da conversão indica o fenômeno de congruência entre os dois sistemas de representação.

A segunda conversão se dá entre registros não congruentes; nessa conversão, o registro inicial é produzido em função dos recursos disponíveis e escolhidos pelo aluno, geralmente com uma combinação entre língua natural, elementos algébricos, argumentações e operações lógicas e elementos visuais. Não se coloca qualquer imposição ou restrição sobre a forma dessa produção, as escolhas são feitas pelo aluno ou pelo grupo de alunos que trabalha na busca do método de resolução do problema, em função tanto das percepções sobre características do problema, quanto do domínio que o estudante tenha sobre o conjunto de recursos possíveis e adequados para estabelecer a abordagem do problema; um dos aspectos dessa elaboração é a noção de dinâmica de fluxo de execução e dinâmica de tratamento de dados.

Duval (2008) descreve a não congruência entre registros de representação pela complexidade que a conversão pode apresentar, e pela inexistência de um conjunto de regras que possam ser aplicadas para obter-se a conversão; assim, um elemento presente no registro inicial pode requerer, para sua conversão ao outro registro, uma série de ações não imediatas e não mecanizáveis; ou ainda: um elemento de um dos registros pode até ocultar-se na outra representação. O trabalho destaca também a importância de se trabalhar os dois sentidos da conversão, pois as operações correspondentes, em geral, não são equivalentes, podem exigir análises e mecanismos muito diferentes, mas favorecem o aprofundamento do domínio que o aluno estabelece sobre o objeto representado.

Uma situação de não congruência pode ser apontada no exemplo do problema dos parafusos: a estratégia de resolução com as subtrações sucessivas

pode sugerir, em sua elaboração inicial, dois processos intermediários independentes: “enquanto a quantidade de parafusos for suficiente para completar uma caixa grande, subtrair sucessivamente 40 dessa quantidade, e depois, quando a quantidade de parafusos remanescentes for menor do que 40 e enquanto for suficiente para completar uma caixa pequena, subtrair 10 dessa quantidade, e depois contabilizar cada caixa grande e cada caixa pequena que tenha sido preenchida”, essa elaboração não indica a necessidade de se vincular as subtrações sucessivas aos processos de contagem das caixas.

É natural imaginar duas etapas independentes: uma que corresponde a agrupar e dispor os parafusos nas caixas (subtrações), e outra que é realizar a contagem das caixas. Na descrição do processo de resolução em linguagem algorítmica, ou em linguagem de programação, esses mecanismos intermediários (subtrações e contagens) devem ser conjugados: a cada subtração, que representa a separação de um grupo de quarenta ou de dez parafusos, deve ocorrer a atualização do respectivo processo de contagem de caixas.

Essa combinação entre subtrações e contagens indica o aspecto dinâmico próprio de um algoritmo, que pode ser explicitado na produção do registro em língua natural, mas que fica encoberto quando se faz a representação correspondente em linguagem algorítmica ou em linguagem de programação.

O texto de um algoritmo é um elemento essencialmente estático que representa sempre um processo dinâmico a ser realizado pelo sistema computacional. As ações representadas nas instruções de um algoritmo ocorrem ao longo de um período de tempo e a cronologia dessas ações fica definida pela organização seqüencial do texto do algoritmo, por suas estruturas de controle (seleção e repetição) e por suas instruções que provocam a execução de módulos auxiliares.

O trabalho de Duval (2008) destaca a importância das atividades de conversão nos processos de constituição de conhecimentos, e sugere que, nos planejamentos de atividades, o professor deve considerar e valorizar esse aspecto. O professor deve propor atividades que conduzam os estudantes a conversões e não apenas a tratamento de registros. É com a prática das atividades de conversão que o estudante consolida seus conhecimentos sobre os objetos representados, e ganha autonomia nas suas manipulações. Duval (2008)

indica também que, nas atividades de exploração e pesquisa do processo de aprendizagem, a presença das tarefas de conversão de registros constitui-se em importante elemento de referência para os estudos e análises.

3.2 Dialética ferramenta-objeto

A atividade de constituir o processo de resolução de um problema computacional alinha-se com as diretrizes propostas pela teoria da dialética ferramenta-objeto, estabelecida por Régine Douady e analisada no trabalho de Maranhão (2008). Tal atividade pode ser interpretada como a descoberta e domínio de um novo objeto até então desconhecido; é possível que não se tenha a abrangência de um objeto matemático (conceito, propriedade ou proposição) como indicado na teoria, mas o processo construtivo é muito semelhante. Em resumo: diante da proposta do problema, o estudante deve recorrer a conhecimentos anteriores que são necessários para a construção, reorganizar tais elementos, criar e acrescentar novas relações, e com eles produzir o método de resolução e garantir a sua validade.

A construção ocorre em ciclos de desenvolvimento, a maioria das vezes com sobreposições, e envolve: *retomar* conhecimentos já disponíveis, *descobrir* e *estabelecer* relações entre esses conhecimentos e a situação do problema proposto que possam conduzir ao método de resolução procurado, *organizar* adequadamente essas relações e *descrever* o processo de resolução.

Nesse processo, é possível identificar as características delineadas na teoria da dialética ferramenta-objeto. A primeira delas pode ser apontada na seleção ou escolha de cada problema a ser proposto; a colocação do problema deve ser definida de tal forma que o trabalho do estudante não seja uma atividade apenas de reproduzir mecanismos já tratados, a resolução do problema deve exigir alguma atividade de criação. Por exemplo, seria pouco produtivo o professor propor e discutir a resolução do problema dos parafusos e em seguida recolocar a proposta com a modificação somente das capacidades dos dois tipos de caixas; em vez disso a proposta poderia ser modificada assim:

> Uma determinada quantidade de parafusos deve ser embalada em caixas de 40 unidades e caixas de 10 unidades, de tal forma que a quantidade de caixas grandes seja a mais próxima possível da quantidade de caixas pequenas – observe os exemplos descritos a seguir. Conhecendo-se a quantidade de parafusos disponíveis, como obter a quantidade de caixas grandes, a quantidade de caixas pequenas e também a quantidade de parafusos que não serão embalados por não completarem uma caixa pequena?
>
> Veja os exemplos:
>
> - para embalar 607 parafusos deve-se empregar: 12 caixas grandes e 12 caixas pequenas, havendo, assim, uma sobra de 7 parafusos.
> - para embalar 613 parafusos deve-se empregar: 12 caixas grandes e 13 caixas pequenas, havendo, assim, uma sobra de 3 parafusos.
> - para embalar 628 parafusos deve-se empregar: 12 caixas grandes e 14 caixas pequenas, havendo, assim, uma sobra de 8 parafusos.
> - para embalar 631 parafusos deve-se empregar: 13 caixas grandes e 11 caixas pequenas, havendo, assim, uma sobra de um parafuso.
> - para embalar 645 parafusos deve-se empregar: 13 caixas grandes e 12 caixas pequenas, havendo, assim, uma sobra de 5 parafusos. (Martins e Rodrigues, 2008, p.81)

Diante dessa nova proposta, para a construção do método de resolução, o estudante aproveitaria uma pequena parte do tratamento anterior (o agrupamento inicial seria feito em lotes de 50 parafusos), e deveria criar uma nova construção correspondente à análise e definição posterior ao cálculo ou à contagem da quantidade de agrupamentos com 50 parafusos, para o ajuste das quantidades de caixas grandes e pequenas, tendo em vista cumprir o requisito da maior proximidade de tais quantidades.

Outro aspecto que pode ser apontado é a sugestão ou orientação para o trabalho com a interação entre domínios (domínio numérico e domínio algébrico) que se constitui como elemento essencial na teoria; no próprio enunciado do problema, com os exemplos dados, se coloca a indicação de que o estudante poderá considerar quantias particulares para encaminhar a análise dos casos possíveis de ajustes das quantidades de caixas. Assim, o estudante realiza operações aritméticas específicas para apoio a essas análises, estende suas conclusões para quantidades variáveis e estabelece as correspondentes relações

lógicas e algébricas. Essa forma de conduta corresponde a realizar uma interação entre o domínio numérico, quando o aluno trata os casos particulares, e o domínio algébrico, quando o aluno organiza suas conclusões para o caso geral, com o emprego de relações lógicas e operações algébricas e não mais numéricas.

Em um momento posterior, o estudante pode retomar os casos particulares com a finalidade de verificar parcialmente a validade de suas construções lógicas e algébricas.

Em situações mais específicas podem ocorrer interações com o envolvimento de outros domínios, mas de forma geral recorre-se ao domínio numérico para as verificações parciais de validade e ajustes de mecanismos; isso é esperado, pois em um curso de introdução aos algoritmos coloca-se a ênfase em processos numéricos.

Em outras situações o domínio geométrico ganha importância, por exemplo, diante da proposta:

> Deseja-se recortar vários quadrados a partir de uma placa de papelão retangular, sempre com a maior medida de lado possível. Conhecendo-se as medidas dos lados da placa de papelão, como determinar a seqüência de medidas dos lados dos quadrados que se pode recortar? Considere que as medidas dos lados da placa são valores inteiros em centímetros. (Adaptado de notas de aulas)

é pouco provável que o aluno não recorra aos aspectos geométricos para iniciar a construção do método de resolução, mas também nesta situação é esperado que, em algum momento, o estudante acione o domínio numérico para avaliar a validade de suas construções. Nesta situação, provavelmente, haverá o trânsito entre o domínio concreto (imaginar a seqüência de recortes da placa), o domínio algorítmico (expressões lógicas e algébricas), o domínio geométrico e o domínio numérico.

Maranhão (2008) indica a prática de interações entre domínios como um caminho para que o aprendiz possa consolidar os avanços de seus conhecimentos com autonomia; as interações entre domínios têm a finalidade de produzir suportes de validação e evolução para novos conhecimentos; não são simples deslocamentos de um conceito de um domínio para outro. As interações envolvem a elaboração de complementos e progressão de conhecimentos já desenvolvidos em um dos domínios, que naquele estágio e no âmbito do domínio

tratado parece esgotado, mas com a conjugação das interações se podem agregar novos contornos e então dar continuidade à evolução.

O processo da dialética ferramenta-objeto oferece elementos para as explicações sobre as atividades de criação dos métodos de resolução. As fases constituintes desse processo acomodam o andamento da atividade dos alunos na resolução de um problema computacional.

Na fase inicial (chamada *antigo*) o estudante busca no seu repertório de conhecimentos os primeiros aspectos que deverão conduzir o trabalho preliminar de criação, esses saberes já disponíveis funcionam como *ferramentas*.

Na fase seguinte, denominada *pesquisas*, o estudante, ao perceber a dificuldade em completar seu trabalho de criação, deve mobilizar *conhecimentos implícitos*, que são conhecimentos novos, possíveis de serem reconhecidos pelo professor, mas ainda não explicados completamente pelo aluno.

A terceira fase (*explicitação*) envolve o diálogo entre alunos e entre professor e alunos com a finalidade de se obter a descrição dos elementos que emergiram com o desenvolvimento do trabalho até este ponto: dificuldades, entraves, ações e resultados.

A quarta fase (*novo implícito*) é constituída pela necessidade de serem especificados e validados os novos conhecimentos construídos.

A quinta fase corresponde à *institucionalização* dos novos conhecimentos, ou seja, deve ocorrer a difusão, entre os componentes do grupo, dos conhecimentos constituídos; as duas últimas fases envolvem o trabalho de familiarização com os elementos novos e a atividade de *novo problema* que deve iniciar um novo ciclo com a reutilização dos novos conhecimentos.

Cada etapa do curso de uma disciplina de introdução aos algoritmos pode ser organizada em fases que correspondem ao processo da dialética ferramenta-objeto, assim: em uma fase preliminar um novo conjunto de recursos (elementos da linguagem algorítmica e da linguagem de programação) é apresentado ao aluno; com isso o estudante faz o primeiro contato com tal conjunto de recursos e este começa a torna-se disponível para emprego na resolução de problemas computacionais. A apresentação se faz por leituras indicadas, exemplos de

aplicação ou exposições do professor; a elaboração do conhecimento do aluno se inicia.

A evolução dos conhecimentos deve ocorrer com a prática do tratamento de problemas computacionais. A cada novo problema, as informações sobre os recursos da linguagem algorítmica ou da linguagem de programação e os conhecimentos relativos à situação do problema são retomados. A partir desses elementos (*antigo*) se inicia o processo de criação do método de resolução que avança por sucessivos ajustes e adequações entre a estratégia de resolução que o aluno desenvolve e os recursos disponíveis da linguagem algorítmica (*pesquisas*); nesse estágio caracteriza-se o *novo implícito*. Essa é a fase em que as dificuldades ocorrem com maior freqüência: não basta que o processo de resolução seja apenas organizado em linhas gerais, o detalhamento da criação é essencial, pois é desse detalhamento que depende a produção do texto do algoritmo ou do programa. Algumas vezes, o tratamento do problema pode exigir um desenvolvimento algébrico complexo para o estudante e, no texto do algoritmo, aquele desenvolvimento pode tornar-se transparente.

Por exemplo, o processo de resolução do problema:

> Uma pequena cooperativa agro-industrial deve produzir manteiga comum e manteiga especial na proporção 4:1, ou seja: para cada 4kg de manteiga comum deve produzir 1 kg de manteiga especial. Sabe-se que a produção de 1 kg de manteiga comum consome 1,6 kg de creme de leite e que a produção de 1 kg de manteiga especial consome 2,2 kg de creme. Conhecendo-se a quantidade (kg) de creme de leite disponível como determinar as quantidades de manteiga comum e especial que a cooperativa deve produzir? (Adaptado de notas de aulas)

demanda a constituição de um modelo de sistema de equações lineares para traduzir a proporcionalidade entre as quantidades dos tipos de manteiga e a restrição que é definida pela quantidade de creme de leite disponível, e depois as transformações desse modelo para a determinação das soluções.

Tal desenvolvimento algébrico não é levado ao texto do algoritmo, apenas as expressões que definem as soluções irão compor o algoritmo. Em outras situações pode ocorrer o contrário disto: um detalhe, não previsto no projeto da estratégia de resolução, deve ser criado e explicitado no algoritmo. Por exemplo, ao delinear a estratégia de resolução do seguinte problema:

> O gerente de uma biblioteca virtual anotou, dia a dia, durante o mês de abril a quantidade diária de visitantes. Conhecendo-se as quantidades anotadas em ordem cronológica, como obter a quantidade total de visitantes durante o mês e também a quantidade de dias necessários, desde o início do mês, para que a quantidade acumulada superasse 3000 visitas? (Adaptado de notas de aulas)

o estudante pode não perceber que deverá introduzir, na construção do algoritmo, um artifício vinculado ao controle para verificar se a quantidade acumulada superou 3000 visitas, como indicado no Quadro 11.

Quadro 11: Algoritmo – problema da biblioteca

```
biblioteca( )
total←0; dia3000←0; dia←0;
enquanto dia<30 faça
   dia←dia+1;
   leia(visitas);
   total←total+visitas;
   se total>3000 e dia3000=0 então dia3000←dia;
imprima(total);
se dia3000>0
  então imprima(dia3000);
  senão imprima("não superada a quantidade 3000");
```

No esboço do método de resolução, geralmente, não ficaria explícito o artifício de definir inicialmente `dia3000` com o valor 0 e acrescentar à expressão de controle `total>3000` o confronto `dia3000=0`.

O estágio de *explicitação* ocorre com a comunicação entre os alunos e entre o professor e o aluno: o estudante expõe sua produção, suas dificuldades ou os resultados percebidos, o professor destaca as características importantes ou mais relevantes para algum aprofundamento. A fase de validação normalmente envolve a simulação do algoritmo ou a observação da execução do programa para algumas instâncias do problema; rigorosamente, apenas essas simulações ou execuções não são suficientes para garantir a validade do algoritmo construído: tal mecanismo é valioso por permitir a percepção, pelo aluno, da necessidade de se retrabalhar as fases anteriores para eliminação de falhas ou enganos introduzidos.

A *institucionalização* se faz com a exposição e discussão sobre as produções dos alunos e pode incluir a apresentação de alguma estratégia de resolução que o professor avalie como interessante ou importante. O critério para essa avaliação depende de vários fatores, desde clareza, organização,

naturalidade, até a exploração de aspectos relativos às características dos recursos das linguagens ou dos fatores definidos pela própria situação do problema. Nesta fase o professor destaca e reforça os aspectos que o estudante deve conhecer e reter.

Outra abordagem possível é realizar confrontos e comparações entre as estratégias e algoritmos que os alunos tenham construído. Com essa forma de trabalho verifica-se um envolvimento maior dos alunos nas discussões e questionamentos; quando a estratégia de resolução e o algoritmo são aqueles que o professor traz prontos e apresenta, o grau de envolvimento nas discussões é menor. O novo explícito é o resultado desta fase da dialética.

A fase de familiarização pode ser composta pela proposta de novos problemas, discussão de outros exemplos ou ainda atividades parciais, como por exemplo, produzir o texto de um algoritmo a partir de uma estratégia de resolução do problema já delineada. O andamento de uma disciplina de introdução aos algoritmos avança com a combinação desses ciclos de trabalho, com a introdução gradativa de novos conjuntos de recursos das linguagens e com a variação do grau de complexidade dos problemas computacionais propostos.

3.3 Ideografia dinâmica e modelo mental

A obra de Lévy (1998) introduz a idéia de ideografia dinâmica, um objeto imaginário que se desenvolve em função das possibilidades resultantes da evolução dos recursos computacionais. A ideografia dinâmica é um projeto que envolve a interação homem-máquina, e tem por objetivo explorar aspectos no âmbito dos signos e da cognição, da linguagem e do pensamento.

O autor afirma que recursos de modelagem e simulação visual por computador, já empregados em vários campos, podem ser descritos como possuidores de numerosos elementos que são organizados para o seu projeto da ideografia dinâmica, por exemplo: sistemas para estudos de biologia molecular, sistemas para construções geométricas, sistemas para simulação de processos de engenharia. Esses sistemas são referidos como ideografias dinâmicas especializadas.

O artigo de Nicolau (2009) confirma essa idéia de ideografias dinâmicas especializadas, ao caracterizar a emergência e estruturação de uma *linguagem funcional*, gerada a partir do objetivo de se oferecer aplicativos avançados em áreas de utilização que devem alcançar escala mundial, com a superação de fronteiras geográficas ou culturais. O autor relaciona *softwares* de editoração, jogos, sistemas operacionais, ferramentas de desenho, que buscam cumprir os requisitos de usabilidade no âmbito universal das culturas.

Essa *linguagem funcional* se organiza em função do incremento de recursos que ocorre com a evolução das tecnologias da comunicação e da informática. O processo de consolidação passa pelo aprimoramento de símbolos que se definem por múltiplas faces: suas formas visuais, seus sons, seus aspectos verbais. Essas características se ajustam às capacidades cognitivas de elaborações a partir de signos que se dirigem à percepção dos sentidos, com independência em relação aos aspectos próprios de cada cultura (língua, comportamentos, práticas cotidianas, tradições). O autor afirma:

> É nesse contexto que se situa a ideografia global, surgida primeiramente devido ao interesse mercadológico de compor programas de interface gráfica para comercialização no mundo inteiro. Essa é a lógica da globalização já identificada na padronização que os produtos vêm adotando, como parte das estratégias de venda a um número cada vez maior de consumidores. Nas mãos de usuários de diferentes nacionalidades e culturas essa ideografia global poderá se tornar uma *linguagem funcional* de grande utilidade para criação e uso de programas e aplicativos livres. (NICOLAU, 2009, p.13)

Lévy (1998) descreve a ideografia dinâmica como linguagem e como tecnologia intelectual. Enquanto linguagem, a ideografia dinâmica deve agregar, de um lado, as características das imagens visuais estáticas, inclusive a liberdade na criação de signos e prolongamento da capacidade de imaginação e, de outro, os elementos presentes nos sistemas computacionais, retratados em seus mecanismos de interface visual, com suportes para modelos espaços-temporais fundados em movimentos, campos de relações e ícones.

O autor explica que a escrita tradicional, desde suas origens, se desenvolve sobre um meio estático e com forma que permite sua linearização, e que a partir do início do século XX, surgem linguagens menos lineares ou estáticas (fotografia e cinema), mas que não possibilitam as interações.

Com o desenvolvimento das tecnologias, as linguagens ganham mais alguns suportes importantes – a possibilidade de "navegar" por textos e hipertextos enriquecidos por imagens e sons, mas as interações possíveis são aquelas que o leitor realiza com o sentido de buscar as direções do foco de seus interesses.

A ideografia dinâmica, projetada por Lévy (1998), deve ultrapassar essas características, é concebida como uma nova forma de linguagem com suporte informático, capaz de oferecer símbolos dinâmicos dotados de memória e de potencial para reagir autonomamente. Nessa escrita os caracteres não carregam os significados apenas por suas formas, mas também por seus movimentos e metamorfoses. Os potenciais para memorização e reação dependem diretamente das crescentes capacidades dos sistemas computacionais e de suas articulações em redes.

A obra de Lévy (1998) se organiza em quatro partes principais. Na introdução o autor oferece uma visão geral de seu projeto. Na segunda parte analisa extensamente a ideografia dinâmica como linguagem, e coloca as discussões a partir de confrontos de sua proposta com a escrita tradicional, com a língua, com as linguagens de programação e com o cinema.

Na terceira parte da obra há o aprofundamento da proposição da ideografia dinâmica como tecnologia intelectual.

> A ideografia dinâmica não se concebe como pura e simples projeção do imaginário de seus exploradores nas telas, mas muito mais como tecnologia intelectual de auxílio à imaginação. Por um lado, a ideografia dinâmica traduzirá, semiotizará e reificará os quase-objetos indeterminados da imaginação; por outro, fabricará signos destinados a ser introjetados e retomados pela atividade imaginante de sujeitos e de coletivos. (LÉVY, 1998, p.100)

Enquanto tecnologia intelectual, a ideografia dinâmica pressupõe o papel fundamental da imaginação nas funções cognitivas, e assim, a proposição indica a construção de um instrumento que se ofereça para prolongar, sustentar e ampliar a atividade espontânea de construção e simulação de modelos mentais que são realizados nas ações de pensamento e comunicação.

O projeto de Lévy (1998) é desenvolvido com várias referências à noção de *modelo mental* e com o pressuposto da imaginação como elemento central da inteligência. O autor esclarece que considerar a imaginação como atividade de

produção e simulação de modelos mentais tem o propósito de simplificar suas análises e elaborações, mas afirma que há outros elementos que completam o papel da imaginação e que não são levados em conta no curso de suas discussões.

Para introduzir a noção de modelo mental, Lévy (1998) indica que cada indivíduo cria para si representações internas de áreas de conhecimento e domínio de ações que percebe em sua vivência. O sujeito recorre a essas representações internas para atos como lembrar, raciocinar, planejar, definir decisões.

As representações internas ou representações mentais são tratadas, no âmbito de uma das correntes da psicologia cognitiva, em três categorias: representações proposicionais, modelos mentais e imagens.

Moreira (1996) sintetiza assim essa classificação: representações proposicionais são constituídas por encadeamentos de símbolos que se assemelham às construções da língua natural; modelos mentais são construídos como análogos de estruturas presentes no mundo exterior, que podem ser estruturas concretas ou abstratas; e imagens são resultados de alguma forma particular de considerar-se um modelo mental, definida a partir de algum ponto de vista específico.

Nessas considerações, o autor elabora a metáfora da atividade mental em correspondência à atividade de um sistema computacional:

> [...] os modelos mentais e as imagens são representações de alto nível, essenciais para o entendimento da cognição humana. Ainda que em seu nível básico o cérebro humano possa computar as imagens e os modelos em algum código proposicional (o "mentalês"), o uso destas representações liberta a cognição humana da obrigação de operar proposicionalmente em "código de máquina". Estas representações de alto nível podem ser comparadas às linguagens de programação dos computadores. Em última análise, o computador trabalha em um código binário, mas o programador não: ele usa linguagens de alto nível que lhe permitem pensar sobre o que o computador tem que fazer usando o código binário. (MOREIRA, 1996, p.195)

Lévy (1998) considera que os indivíduos raciocinam com modelos mentais, articulando-os conforme sejam exigidos. Um modelo mental possui como característica principal o fato de se estruturar de tal modo que seja um análogo do real e, ao mesmo tempo, sirva adequadamente no sentido de permitir operações necessárias para o tratamento das situações em que se configura.

Um modelo mental possui em seu núcleo a essência daquilo que é o seu correspondente real, e, como complementos, os operadores que devem cumprir a necessidade que se coloca com a situação que deve ser trabalhada na mente. Assim, por um lado, um modelo mental se estabelece como representante análogo de algum objeto real, e, por outro lado, como recurso adequado, capaz de operar satisfatoriamente diante de uma necessidade.

Moreira (1996) registra que um mesmo objeto pode ser representado por diferentes modelos mentais, cujas configurações dependem do tipo de emprego que se pretende para tais modelos, e também do grau de conhecimento e de interesse do sujeito sobre aquele objeto. Por exemplo: o motorista de um caminhão e o gerente de logística de uma transportadora rodoviária produzem, certamente, modelos mentais para o objeto caminhão com núcleos semelhantes, mas com funções e operadores (os complementos) diferentes.

Lévy (1998) encontra apoio nesses conceitos para eleger como meta da ideografia dinâmica a produção de elementos externos, correspondentes às representações internas, com a vantagem de superar limites biológicos como memória, atenção ou concentração, e assim oferecer maior liberdade e alcance nas elaborações mais complexas.

A Engenharia Didática, que integra este trabalho, tem a intenção de oferecer objetos com a finalidade de facilitar a constituição de modelos mentais para algoritmos pelo estudante, especificamente busca favorecer a percepção do processo dinâmico que um algoritmo indica e que a representação estática textual apenas anuncia. Tais objetos não possuem o conjunto de características preconizadas no projeto da ideografia dinâmica (a capacidade de reagir e memorizar), mas levam a pensar no projeto de sistemas que poderiam ser sementes nessa direção.

3.4 Metodologia e procedimentos metodológicos

3.4.1 Engenharia Didática

O termo Engenharia Didática tem sua origem na metáfora da atuação do professor-pesquisador vista como o trabalho de um engenheiro, que ao conceber seu produto (um sistema, um mecanismo, um dispositivo) tem que levar em conta toda a malha de fatores que define os contornos da situação que deverá acolher o resultado de seu projeto.

O engenheiro tem suas bases de construção no conhecimento científico, mas não se limita aos objetos já clarificados pela ciência. Não basta pensar apenas nos objetivos de seu produto, há de se considerar a complexidade de elementos que podem ter influência sobre a realização ou funcionalidade do seu projeto.

A idéia da Engenharia Didática, conforme Artigue (2009), surge na Didática da Matemática, vertente de educadores franceses, no início da década de 1980, e se articula inicialmente com a Teoria das Situações Didáticas, produção teórica daquela escola. A Engenharia Didática se estabelece tanto como metodologia de investigação, quanto como prática de desenvolvimento didático com base em investigações. Os anseios colocados nessa época podem ser resumidos em duas questões:

Como levar em conta a complexidade dos cenários de uma sala de aula no desenvolvimento de uma atividade de investigação, que tem seu centro principalmente disposto em experimentos e questionários?

Como pensar sobre as relações entre investigação e ação nos sistemas educacionais?

Artigue (2009) declara que, em resposta a essas questões, emergiram duas idéias: “phénoméno-technique” para a metodologia ou prática de investigação e “engenharia didática” para ações sobre as situações de ensino com base em investigação. Em pouco tempo o primeiro termo foi abandonado, e a expressão Engenharia Didática passou a abrigar também o primeiro sentido.

O trabalho de Artigue (2009) destaca algumas características da Teoria das Situações Didáticas, que afetam fortemente a noção de Engenharia Didática.

Em primeiro plano, o papel central que desempenha a noção de situação e a forma pela qual a aprendizagem é vista: um processo de adaptação que depende das características das situações em que o mesmo se desenrola. Nesse panorama, o alvo da teoria é o entendimento dessa dependência e o desenvolvimento de conceitos, ferramentas e abordagens metodológicas que favoreçam o processo de adaptação, ou seja, o aprendizado.

Um segundo foco apontado é a atenção dirigida à epistemologia do conhecimento alvo do aprendizado. A teoria busca vincular um pequeno conjunto de situações fundamentais a cada objeto do conhecimento a ser tratado. Tal conjunto de situações deve atender adequadamente a essência da epistemologia daquele bloco de saberes.

Outro fator é a importância atribuída às características que definem a forma de interação do estudante com o cenário em que a situação é colocada; a aprendizagem deve evoluir pela vivência da própria situação e não pela condução do professor.

Ainda conforme explica Artigue (2009), a Teoria das Situações Didáticas valoriza a distinção entre as três funcionalidades do conhecimento matemático: a ação, a expressão ou a comunicação e a demonstração. Além disso, coloca como funções essenciais do professor a concepção e a organização da situação e a administração dos processos de discussão e institucionalização dos conhecimentos.

Essas características e fatores, relacionados à Teoria das Situações Didáticas, definem as faces e a forma de organização de uma Engenharia Didática. A concepção e realização de uma Engenharia Didática ocorrem em algumas fases principais: análises preliminares, análises *a priori*, experimentação e análises *a posteriori*, e validação. Apesar de transparecer a idéia seqüencial, o processo pode envolver retrocessos e avanços em consonância com as observações e os trabalhos de análises.

Uma particularidade da Engenharia Didática, como metodologia de investigação, é o sistema de validação interna. Essa característica exige atenção

com a sintonia entre os vários componentes e fatores que definem a Engenharia: os suportes teóricos, as análises preliminares, os confrontos e implicações entre análises *a priori* e *a posteriori*, o pressuposto papel do estudante como responsável pela constituição de seu aprendizado, a atuação do professor como mediador na realização das atividades planejadas.

Na fase "análises preliminares", uma Engenharia Didática busca entender os aspectos do objeto de conhecimento em foco a partir de três fontes: a epistemologia do saber, o estado das práticas e realizações didáticas e os traços cognitivos do sujeito do aprendizado. Em função dessas análises preliminares, se configura a Engenharia Didática.

A fase "análise *a priori*" envolve uma parte descritiva acompanhada de outra parte preditiva. Nessa fase são explicitadas as escolhas mais abrangentes relativas à organização geral da Engenharia e também as definições mais localizadas, relativas à proposição de cada atividade; o sentido é permitir avaliar as produções dos estudantes nas situações didáticas planejadas, diante do que se coloca como expectativa.

O processo de validação interna exige que as expectativas descritas sejam específicas, de tal modo que o aprendizado almejado possa ser apontado ao observar-se a realização dos estudantes.

O trabalho de Carneiro (2005) assinala:

> [...] as hipóteses não podem ser muito amplas, a ponto de por em jogo processos de aprendizagem, a longo prazo. Ao expressá-las, é preciso ter consciência de que vamos voltar a elas, durante a experimentação, checando-as, inquirindo-as. Será que o plano funciona? Será que nossas hipóteses são válidas? (CARNEIRO, 2005, p.103)

As fases experimentação e análise *a posteriori* e validação se iniciam ao colocar-se em prática o plano delineado na fase análise *a priori*, ou seja: ao produzir-se o desenrolar da atividade experimental.

No artigo de Almouloud e Coutinho (2008) encontra-se a indicação de que, durante a experimentação, a partir das análises locais subseqüentes à realização de cada sessão de atividades, podem surgir indicativos da necessidade de ajustes no plano inicial, e a conseqüente revisão das análises *a priori* ou das análises preliminares. Os autores afirmam:

> A fase da experimentação é clássica: é o momento de se colocar em funcionamento todo o dispositivo construído, corrigindo-o se necessário, quando as análises locais do desenvolvimento experimental identificam essa necessidade, o que implica em um retorno à análise *a priori*, em um processo de complementação. (ALMOULOUD e COUTINHO, 2008, p.68)

O propósito mais amplo das análises *a posteriori* é construir conclusões em função das associações adequadas e pertinentes entre os objetivos delineados *a priori* e as observações e análises realizadas sobre o desenvolvimento experimental, tendo em vista também a avaliação sobre a reprodutibilidade e a regularidade dos eventos observados.

O trabalho de Artigue (2009) assinala que, hoje em dia, a Engenharia Didática, apesar de conviver com outras metodologias mais recentes, ainda é uma ferramenta largamente utilizada, e que segue em processo de evolução, inclusive com trânsito cada vez maior por pesquisas sobre Educação de outras disciplinas, e também por outras culturas da Educação Matemática. Tal evolução envolve uma visão mais complexa sobre o papel do professor e admite maior flexibilidade nas relações de responsabilidades em sala de aula entre professor e estudante.

3.4.2 Procedimentos metodológicos

O estudante em atividades de construção de algoritmos e programas é a fonte para a construção dos dados neste trabalho. O alvo é investigar como o estudante revela, trata e domina a noção de processo dinâmico inerente a um algoritmo ou programa, para isso a pesquisa se estabelece a partir do trabalho de observação de alunos em atividades de tratamento de problemas computacionais.

Certamente as respostas não seriam obtidas se fosse colocada aos alunos a questão direta "*como você percebe, considera e se utiliza da noção de processo dinâmico inerente aos algoritmos, ao tratar um problema computacional?*". Por outro lado, tal noção não se materializa explicitamente nos registros dos textos nem de algoritmos nem de programas, dessa forma é possível prever a

necessidade de se observar as falas, os gestos, os questionamentos, os esboços e rascunhos escritos, as argumentações dos estudantes durante o processo de desenvolvimento de um algoritmo ou programa, além dos registros de textos dos algoritmos e programas produzidos.

É possível considerar uma analogia parcial entre o texto de um algoritmo e o mapa de um guia rodoviário: a figura do mapa (um elemento estático) representa percursos possíveis entre algumas localidades, o texto de um algoritmo (também um elemento estático) representa as possibilidades de seqüências de ações de um sistema computacional. Ao planejar um deslocamento, a pessoa, de posse do mapa, pode imaginar a representação de sua posição a cada momento do movimento e seu deslocamento no próximo momento; ao planejar a construção do algoritmo, a cada momento, pode-se imaginar a execução de uma instrução e a próxima ação no momento seguinte.

A representação figural do percurso (mapa) ou o registro do texto do algoritmo, isoladamente, não contêm elementos que revelem explicitamente as atividades de planejamento de suas criações nem a noção de seqüenciamento ao longo do tempo. O acesso à noção do dinâmico só é possível com o acompanhamento e observações dos trabalhos de desenvolvimento, é certo que interessa também o produto final que se estabelece no texto do algoritmo ou do programa.

As características relacionadas acima apontam a adequação de uma abordagem qualitativa para este trabalho de investigação; não interessa especialmente a classificação ‘certa’ ou ‘errada’ da produção do aluno, mas sim a forma e os contornos de como se desenrola essa produção e seus caminhos. A essência do trabalho é composta a partir da observação e interpretação cuidadosa dos estudantes em tarefas de desenvolvimento de algoritmos e programas.

No prefácio da obra *Pesquisa Qualitativa em Educação Matemática*, D’Ambrosio (2004) aponta um dos méritos da pesquisa qualitativa com essas palavras:

> No meu entender, é o caminho para escapar da mesmice. Lida e dá atenção às pessoas e às suas idéias, procura fazer sentido de discursos e narrativas que estariam silenciosas. E a análise dos resultados permitirá propor os próximos passos. (D’AMBROSIO, 2004, p.21)

Com o objetivo de garantir a credibilidade da pesquisa, Borba e Araújo (2004) recomendam a articulação de alguns elementos que devem organizar essas atividades de observação e interpretação; neste trabalho foram colocados em prática três elementos principais.

O primeiro desses elementos é o registro de anotações, durante a realização das observações (*diário de campo*), de aspectos percebidos como pontos de interesse especiais, necessidades de intervenções ou ocorrências não previstas, e fatores que poderiam contribuir para ajustes das propostas das próximas atividades. Outro componente é a gravação (som e imagem) das atuações dos grupos em atividade: esses registros podem oferecer os detalhes dos comportamentos (falas, gestos, expressões, argumentações) dos alunos durante as atividades. O terceiro é a coleção de respostas escritas produzidas pelos grupos: nestas respostas estão, além dos textos de algoritmos e programas construídos, alguns aspectos intermediários trabalhados durante os processos de construções e também aspectos relativos ao entendimento das propostas dos problemas e ao domínio de recursos próprios da linguagem algorítmica e da linguagem de programação.

Nesta pesquisa, foi planejada e realizada uma engenharia didática que envolve a proposta de uma seqüência de quatro atividades de estudos, discussões e resoluções de problemas computacionais, realizadas por grupos de três estudantes iniciantes (primeiro semestre) de um curso de Ciência da Computação, matriculados em uma disciplina de introdução ao desenvolvimento de algoritmos e programação. No próximo capítulo, com a apresentação das atividades que constituem a Engenharia Didática, as quantidades de grupos participantes serão especificadas.

Na primeira experimentação (segundo semestre de 2008), as atividades foram realizadas por dez alunos (nem todos participaram das quatro atividades), esses alunos se dispuseram a participar dos trabalhos em horários além do tempo normal de aulas, nos laboratórios de informática da instituição que oferece o curso. Na segunda experimentação (segundo semestre de 2009), todos os alunos (cerca de 30 estudantes) da mesma disciplina foram convidados a participar dos trabalhos que ocorreram durante os horários regulares de aulas. A segunda

experimentação ocorreu com o objetivo de confirmar resultados obtidos na primeira experimentação, e verificar a reprodutibilidade da Engenharia.

A etapa inicial de cada atividade é a leitura de um texto breve com a apresentação de alguns conceitos (linguagem algorítmica e linguagem de programação), ao final da leitura são propostas algumas questões que devem ser discutidas pelos alunos; nessa etapa o objetivo é colocar o estudante diante dos conceitos relacionados nas próximas etapas da atividade, para a retomada e aprofundamento de tais conceitos e recursos.

Na etapa seguinte, os estudantes devem trabalhar com a experimentação e a operação de um aplicativo que representa um algoritmo e ilustra seu mecanismo de execução: para isso é descrita a proposta de um problema computacional cujo método de resolução é representado no aplicativo que exibe a execução do algoritmo. O objetivo nessa etapa é tornar evidente o processo dinâmico vinculado à sua representação estática (o texto do algoritmo); o esperado é que esse procedimento favoreça o domínio pelo estudante da noção da dinâmica própria de um algoritmo. A Figura 4 é a tela obtida pela execução do aplicativo, elaborado pelo pesquisador, que foi utilizado na quarta atividade.

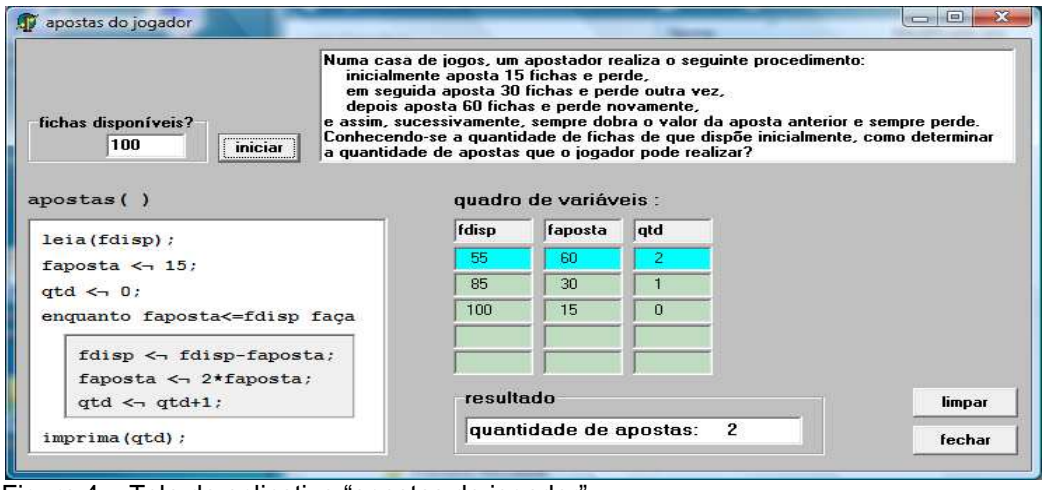


Figura 4 – Tela do aplicativo “apostas do jogador”

Na terceira etapa é colocada a proposta de um problema computacional: os estudantes devem elaborar o processo de resolução e produzir a representação de tal processo na forma de um algoritmo e construir o programa correspondente.

Os métodos de resolução exigem a construção de algoritmos com estruturas de fluxo seqüencial nas duas primeiras atividades e fluxos alternativos e repetitivos nas outras duas atividades. Nessa terceira etapa o esperado é que o grupo de alunos discuta a respeito da construção do método de resolução, em todas as suas fases: entendimento da proposta do problema, levantamento das características e necessidades, concepção e organização do método, representação do algoritmo e implementação do programa.

Para essa terceira etapa, além da proposta do problema computacional, são colocadas algumas questões cujas respostas devem delinear a construção do método de resolução. O grupo de alunos têm acesso a um pequeno aplicativo, elaborado pelo pesquisador, que corresponde à implementação de um método de resolução do problema, cuja experimentação deve auxiliar tanto no entendimento da proposta do problema quanto na verificação da validade do algoritmo e programa construídos, com o confronto de seus resultados com os resultados obtidos pela execução do programa produzido pelo grupo.

A Figura 5 é uma janela produzida pela execução do aplicativo que foi introduzido para a terceira parte da quarta atividade da engenharia.

Figura 5 – Tela do aplicativo “propagação do vírus”

As sessões de trabalho das atividades, durante a primeira experimentação, foram acompanhadas e gravadas em áudio e vídeo, as gravações constituem um dos suportes para análises posteriores às sessões. Na segunda experimentação

não foi possível esse procedimento (gravação), pois havia na sala dez ou onze grupos trabalhando simultaneamente e, nessa situação, os detalhes dos diálogos e as imagens dos alunos em atividade ficariam perdidos na filmagem.

Cada atividade foi planejada para ter a duração aproximada de noventa minutos e essas sessões ocorreram em quatro semanas consecutivas, com uma atividade por semana, a partir da terceira semana do início do curso da disciplina. Durante a primeira experimentação, após a realização da primeira atividade, os planos iniciais das outras atividades foram revistos com a finalidade de reduzir o tempo necessário para a realização das tarefas propostas, pois para a realização da primeira atividade o tempo necessário foi muito maior do que o previsto.

4 ANÁLISES DAS ATIVIDADES E RESULTADOS

A seqüência de atividades foi planejada para ser realizada por grupos de estudantes do primeiro semestre de um curso noturno de Ciência da Computação, a partir da terceira semana do curso de uma disciplina de introdução ao desenvolvimento de algoritmos.

Durante as três primeiras semanas de aulas, anteriores à realização da primeira atividade, são apresentados conceitos e recursos iniciais relacionados à linguagem algorítmica e à linguagem de programação. Tais conceitos são retomados na fase inicial da primeira e da segunda atividade, em que é proposta a leitura de um texto que resume aqueles conceitos: problema computacional, algoritmo, programa, sistema de linguagem, primeiros tipos primitivos, variável e primeiras instruções da linguagem algorítmica e da linguagem de programação.

O trabalho inicial (duas primeiras semanas), nas aulas de laboratório, visa a apresentação do ambiente de programação e da forma geral de constituição de um programa. A partir da terceira semana, nas aulas de laboratório, é iniciado o trabalho de prática de implementação de programas. Nas aulas de teoria, durante as três primeiras semanas, os alunos realizam os primeiros trabalhos de construção de algoritmos em que são aplicados aqueles recursos iniciais.

Nas próximas aulas (quarta, quinta e sexta semanas) são apresentadas as estruturas de controle de fluxo de processamento: controles de seleção e controles de repetição. A apresentação das estruturas de controle de seleção antecede a realização da terceira atividade, e as estruturas de controle de repetição são apresentadas antes da realização da quarta atividades. A terceira

atividade retoma, também com a proposta de uma leitura, as estruturas de controle de seleção. A quarta atividade, da mesma forma, envolve o trabalho com as estruturas de controle de repetição.

Para a primeira experimentação (segundo semestre de 2008), como a maioria dos alunos já tem algum compromisso profissional durante o dia, o recrutamento dos estudantes para as atividades foi feito considerando-se apenas as possibilidades de horários disponíveis para os encontros. Os estudantes se dispuseram a participar, a partir de um convite feito pelo pesquisador, ou seja, a participação foi voluntária. Essas atividades não substituíram as aulas normais programadas para o curso, e foram realizadas aos sábados, ocupando uma parte do período da manhã.

Para a segunda experimentação (segundo semestre de 2009), as atividades foram programadas para ocorrerem durante uma parte do tempo das aulas normais, a partir da terceira semana do curso da disciplina. Todos os alunos da turma foram convidados, e a maioria participou das quatro atividades previstas, também aqui a participação não foi obrigatória.

A escolha pela proposta de trabalhos em pequenos grupos (dois ou três componentes) foi definida em função das finalidades da investigação: um dos aspectos a ser observado é o emprego, pelos alunos, dos registros em língua natural durante todo o processo de tratamento dos problemas computacionais, tanto para comunicar suas idéias, como para argumentar e justificar suas construções. Nos trabalhos em grupos, pela necessidade de se partilhar as idéias, os ajustes e os elementos da elaboração dos algoritmos, torna-se essencial o emprego dos registros de fala, além dos registros escritos.

4.1 Análises – primeira atividade

4.1.1 Apresentação da introdução da atividade

A primeira atividade, na primeira experimentação, iniciou-se com uma introdução, em que foi proposta a leitura de um texto breve com informações gerais e especificação dos objetivos da disciplina. Em seqüência foi proposta uma

pequena série de questões para confronto entre os conteúdos, os objetivos da disciplina e as expectativas do estudante.

O texto para essa fase introdutória foi o seguinte:

A disciplina *Computação e Desenvolvimento de Algoritmos 1* – informações gerais e objetivos

Esta disciplina, desenvolvida durante o primeiro semestre do curso de Ciência da Computação, é introdutória em um dos eixos (Programação e Algoritmos) que compõem a sua organização curricular; dessa forma, outras disciplinas deverão complementar e aprofundar essa introdução; na grade de disciplinas do curso de Ciência de Computação da FEI, completam esse eixo as seguintes disciplinas: Computação e Desenvolvimento de Algoritmos 2, Linguagens e Técnicas de Programação 1, Linguagens e Técnicas de Programação 2, Estruturas de Dados e Análise e Complexidade de Algoritmos. O curso está organizado para 12 ou 13 semanas de aulas efetivas (já excluídas as semanas com atividades de avaliação ou feriados) com 4 aulas de teoria e 2 aulas de laboratório por semana. Nas aulas de teoria o foco principal é a resolução de problemas computacionais e os respectivos desenvolvimentos de algoritmos e nas aulas de laboratório o centro é a implementação e depuração de programas.

Os objetivos gerais colocados para essa disciplina:

Ao longo do curso o aluno deverá desenvolver capacidades e habilidades para:

- compreender, discutir e analisar situações correspondentes a problemas computacionais variados que envolvam conceitos já trabalhados no Ensino Fundamental ou no Ensino Médio, conceitos relativos ao cotidiano e problemas computacionais específicos da área de Informática;
- construir os métodos de resolução desses problemas e descrever esses métodos utilizando uma linguagem algorítmica;
- implementar programas de computador, correspondentes às soluções desses problemas, utilizando um ambiente de programação.

Na primeira fase do curso (6 ou 7 semanas de aulas), em função desses objetivos, serão apresentados alguns conceitos básicos iniciais e um conjunto de recursos essenciais de uma linguagem algorítmica e de uma linguagem de programação (C/C++). Entre os conceitos básicos iniciais estão: problema computacional, algoritmo, programa, ambiente de programação, variável e tipos primitivos de dados.

As atividades são principalmente as propostas de problemas computacionais, assim, os recursos apresentados relativos à linguagem algorítmica e à linguagem de programação, são aplicados para a constituição dos métodos de resolução dos

problemas propostos e as construções dos correspondentes algoritmos e programas. Nessa primeira fase são introduzidos os tipos numéricos primitivos que permitem o tratamento de dados representantes de informações de natureza numérica: quantidades, medidas, valores monetários, taxas, etc., o tipo lógico que possibilita o tratamento de expressões com os operadores de relação de ordem e operadores lógicos, e as estruturas de controle de fluxo de processamento (estruturas de controle de seleção e estruturas de controle de repetição). Na segunda fase é apresentado o tipo cadeia de caracteres para o tratamento de informações como nomes de pessoas, nomes de produtos ou objetos, códigos não numéricos ou textos em geral, e as noções e recursos para a organização modular de algoritmos e programas que consiste em decompor os processos em módulos menores e mais simples e a correspondente articulação desses módulos para a elaboração do processo necessário para a situação do problema computacional que se busca resolver.

Questões iniciais:

- Antes de ter conhecimento sobre as orientações dessa disciplina, quais assuntos você esperava que fossem tratados?
- Há algum outro assunto que você considera que seria importante ser tratado nessa disciplina?
- Tendo em vista os objetivos especificados, você julga necessário o acréscimo de outras capacidades ou habilidades entre aquelas já colocadas como focos dos objetivos?

4.1.2 Análise *a priori* – introdução da atividade

Com essa tarefa preliminar, um dos objetivos é favorecer a aproximação entre os estudantes de cada grupo. As questões colocadas devem permitir a troca de idéias sobre o desenvolvimento da disciplina, e provocar um diálogo inicial em que a despreocupação com o fato de produzir respostas corretas deve facilitar a discussão sobre as opiniões expostas, e com isso acelerar a aproximação dos componentes.

Os diálogos e discussões (registros orais), durante os trabalhos com as atividades, são aspectos importantes das observações que se pretende fazer, dessa forma é interessante favorecer a aproximação entre os componentes com a colocação dessa introdução à primeira atividade.

Outro objetivo é exercitar o procedimento, pelos estudantes, de descrever (registro escrito) as conclusões do grupo. É também um objetivo colher elementos, a partir das expectativas manifestadas pelos alunos, que possam auxiliar nos ajustes das próximas atividades deste trabalho e também da própria proposta da disciplina. Esses três objetivos não foram comunicados aos alunos.

Para a segunda experimentação, essa leitura introdutória e as correspondentes questões não foram propostas. O motivo principal dessa modificação foi a restrição do tempo disponível para a realização das atividades, estritamente limitado em 100 minutos. Durante a primeira experimentação o tempo ultrapassou esse limite. Além disso, na segunda experimentação, quase todos os alunos da turma participaram e, dessa forma, tiveram maior liberdade para definirem os grupos de trabalho, segundo suas preferências de amizade e afinidade. Com tal liberdade, o objetivo de favorecer a aproximação entre os componentes perdeu importância.

4.1.3 Observação e análise *a posteriori* – introdução da atividade

A primeira atividade, na primeira experimentação, foi realizada por sete estudantes organizados em três grupos, um com três componentes e outros dois com dois componentes. Na segunda experimentação, foram dez grupos com três alunos e dois grupos com dois alunos.

Cada grupo recebeu um caderno com o material impresso das propostas das tarefas e um disco (CD) com dois aplicativos. O primeiro aplicativo para a ilustração de um processo de resolução do problema das lâmpadas (segunda parte desta primeira atividade) e segundo que produz resultados para o problema da farinha (terceira parte desta primeira atividade), mas sem a representação do algoritmo correspondente.

Cada grupo recebeu ainda canetas, folhas de papel e uma calculadora simples para apoio na realização das tarefas. Na segunda experimentação as calculadoras não foram distribuídas, mas os alunos foram orientados a utilizar a ferramenta calculadora do ambiente operacional.

A sala utilizada possui microcomputadores configurados de tal forma a tornar possíveis as operações exigidas: ambiente operacional para execução dos aplicativos fornecidos e ambiente para a edição e execução dos programas construídos pelos grupos. As atividades foram planejadas com a utilização de apenas um equipamento para cada grupo, pois o objetivo era que os trabalhos fossem conduzidos pelo grupo e não de forma individual.

Antes do início das atividades, o professor-observador explicou a finalidade das atividades como material de suporte a um trabalho de pesquisa sobre o processo de aprendizagem para o tratamento de problemas computacionais. Apenas o professor-observador realizou as tarefas de condução e observação das atividades.

Houve falha na tentativa de filmar os trabalhos dos grupos: o registro de áudio foi perdido para dois dos três grupos que realizaram a primeira atividade.

A finalidade principal da tarefa introdutória era acelerar o entrosamento entre os componentes de cada grupo. Os dois alunos do terceiro grupo realizaram a leitura proposta praticamente sem nenhuma discussão, e não registraram suas respostas às três perguntas feitas – um dos componentes desse grupo anotou nas folhas de rascunho brevemente para a primeira questão "*programação orientada a objeto, linguagens C/C++, JAVA*" e para as outras duas: "*não*" e "*não*". Esse terceiro grupo foi o único que completou as tarefas propostas e durante a última meia hora do encontro, apenas um dos alunos permaneceu para finalizar os registros escritos das respostas, o outro aluno informou que tinha compromisso e não poderia continuar por mais tempo, mas nesse momento as tarefas já haviam sido concluídas e faltava apenas a transcrição das respostas para o caderno de atividades.

Os outros dois grupos, pouco tempo depois de iniciada a leitura, começaram as discussões e trocas de idéias. O segundo grupo parece não ter entendido o sentido das questões propostas ao final da leitura introdutória, as respostas registradas foram: "*tínhamos consciência que o estudo proposto seria com o intuito de avaliação do aluno, porém não conhecíamos o foco da pesquisa*" – parece que houve confusão entre aquilo que foi proposto como tarefa de introdução e a exposição oral inicial do professor-observador, que pretendia delinear os objetivos da realização das atividades como parte do trabalho de

pesquisa. O termo "*avaliação do aluno*" parece ter sido utilizado com o sentido de analisar os comportamentos e posturas durante o desenvolvimento das tarefas.

O primeiro grupo respondeu à primeira questão assim: "*esperávamos que as aulas tratassem somente de lógica de programação e desenvolvimento de algoritmo, mas percebemos que as aulas também tratam de raciocínio lógico como resolução de problemas.*" – parece que os alunos perceberam a vinculação essencial entre elaboração de algoritmo e problema computacional. Entre as respostas às outras duas perguntas percebeu-se a intenção de sugerir a introdução de situações profissionais concretas no repertório de problemas tratados: "*poderíamos desenvolver projetos oferecendo soluções para empresa*", essa contribuição deverá ser considerada nos planejamentos da disciplina dos próximos semestres – mesmo com a introdução apenas dos conceitos e recursos iniciais de linguagem deve ser possível maior aproximação com casos profissionais concretos, e talvez isso seja um fator a favorecer a motivação do estudante.

A partir da observação dos trabalhos nos grupos, verificou-se que o objetivo de favorecer a aproximação entre os componentes foi atendido para o primeiro e segundo grupos, e apenas parcialmente cumprido para o terceiro grupo. No terceiro grupo, foram poucas as discussões e diálogos, tanto nessa fase introdutória como nas outras fases da primeira atividade.

4.1.4 Apresentação da primeira parte da atividade

Após a introdução foi colocado outro texto para leitura com a apresentação resumida dos seguintes conceitos: *problema computacional*, *algoritmo e programa*, *sistema de linguagem e ambiente de programação*, *tipos de dados e variáveis* e *primeiras instruções: entrada, saída e atribuição*.

O texto proposto para essa leitura é o seguinte:

Problema computacional

> De forma direta pode-se destacar duas características principais que definem o conceito de problema computacional: o seu método de resolução deve envolver a manipulação ou transformação de informações e a sua proposta deve envolver uma situação geral de tal forma que a resposta é um processo de

resolução que deve atender adequadamente o alcance colocado na proposta do problema.

Algoritmo e programa

O método de resolução de um problema computacional pode ser descrito, em princípio, empregando-se qualquer forma de registro (oral, escrito, esquemas com recursos de desenho, etc.), mas, tendo em vista que a intenção é utilizar o computador para executar o processo de resolução, deve-se organizar o método de maneira a tornar possível a construção do programa correspondente. Assim, no âmbito do sistema computacional, o método de resolução será descrito por um programa, ou seja, uma seqüência de instruções, descritas em uma linguagem de programação, que ao ser executada conduz à resolução de uma instância do problema. Com a finalidade de possibilitar ou facilitar a implementação de um programa, é introduzida uma construção intermediária, utilizando-se uma linguagem algorítmica, para a descrição do método de resolução, assim, o algoritmo é também a descrição organizada do método de resolução de um problema; no algoritmo não estão presentes alguns detalhes que devem ser especificados no programa, essencialmente, esses detalhes são relativos às interfaces que o sistema computacional deve oferecer ao usuário. A concepção de uma linguagem algorítmica leva em conta o sistema de linguagem de programação que será empregado, pois é interessante que as traduções de algoritmos em programas sejam diretas a menos dos detalhes das interfaces e de alguns outros aspectos próprios do sistema de linguagem de programação, assim, a passagem de um algoritmo para um programa deve corresponder a uma tradução direta com o acréscimo de alguns detalhes ausentes no algoritmo.

Sistema de linguagem e ambiente de programação

Um sistema de linguagem de programação de alto nível é constituído pelas regras de sintaxe e semântica da linguagem (vocabulário, elementos de organização como separadores e delimitadores, etc.) e por um programa especial – compilador ou interpretador – que faz a conversão do programa fonte – escrito na linguagem de programação – para o programa executável – com instruções em linguagem de máquina – que é efetivamente o programa a ser executado pelo sistema computacional; a ação do compilador é interna ao sistema computacional.

Um ambiente de programação é um conjunto articulado de recursos que existe para facilitar a implementação de um programa, basicamente: construção do texto do programa fonte, manutenção dos arquivos relacionados, acionamento do compilador, execução e depuração do programa.

Tipos de dados e variáveis

Uma linguagem de programação de alto nível deve oferecer alguns tipos de dados (tipos primitivos de dados) a partir dos quais o programador pode representar as informações envolvidas no problema computacional; normalmente esses tipos de dados estão agrupados em três categorias: tipo numérico, tipo lógico e tipo alfanumérico. Os tipos primitivos de dados, além da natureza, definem também os operadores disponíveis para aqueles dados. Por exemplo: um dos tipos de dados numéricos disponíveis em C++ é o tipo *int* cujos valores são números inteiros e com os quais se pode operar adição, subtração, multiplicação, quociente inteiro e resto de divisão.

Variáveis são abstrações de conjuntos de células de memória do computador com a capacidade de realizar o armazenamento de um dado a cada momento da execução do programa. Em um sistema computacional, a essência das computações é justamente definir ou redefinir conteúdos de variáveis a partir da execução de instruções de um programa. Os principais atributos de uma variável são: nome (identificador que o programador escolhe para fazer qualquer referência à variável), endereço (identificador que o sistema vincula ao nome da variável e que utiliza internamente para as referências à variável), conteúdo (valor do dado armazenado no conjunto de células de memória) e tipo (tipo dos dados que a variável pode armazenar).

Primeiras instruções: entrada, saída e atribuição

As primeiras instruções (entrada, saída e atribuição) permitem a construção de algoritmos ou programas com fluxos de processamento seqüenciais, nesse grupo as instruções que compõem o processo são executadas exatamente uma vez cada uma desde a primeira até a última instrução; o fluxo de processamento tem como partida a instrução inicial e segue, executando uma a uma das instruções, até a instrução final.

A instrução de entrada, no algoritmo, tem a seguinte forma geral: **leia(<variável>)** e a ação correspondente é a interação entre o usuário que fornece o dado e o sistema que recebe e armazena o dado fornecido como conteúdo da **<variável>** alvo da leitura. A instrução de saída, com a seguinte forma geral: **imprima(<expressão>)**, indica a ação do sistema em produzir como saída o valor da **<expressão>** disposta como parâmetro da instrução, ou seja: o sistema avalia a **<expressão>** e o resultado obtido é levado ao dispositivo de saída.

A instrução de atribuição (**<variável>** ← **<expressão>**) indica a ação interna do sistema correspondente ao cálculo do valor da **<expressão>** e armazenamento de tal valor como novo conteúdo da **<variável>** alvo da atribuição, assim: o sistema busca, na memória principal, os conteúdos das variáveis que figuram na **<expressão>**, realiza as operações, observando as regras de precedência e associatividade para os operadores, e leva o valor obtido ao final da avaliação da **<expressão>** para definir ou redefinir o conteúdo da **<variável>** alvo da atribuição.

Veja no exemplo abaixo a transcrição do algoritmo para o programa em C++ dessas primeiras instruções.

```
principal( )
leia(qf);
g2←qf*(qf-1)/2;
imprima(g2);
```

```
#include <iostream>
using namespace std;

int main( ){
  int qf, g2;
  cout<<"quantidade de fichas? ";
  cin>>qf;
  g2=qf*(qf-1)/2;
  cout<<"grupos com 2 fichas: "<<g2<<endl;
  system("pause");
  return(0);
}
```

4.1.5 Análise *a priori* – primeira parte da atividade

A expectativa relacionada a essa leitura é que os alunos possam melhor situar-se sobre os conceitos que serão manipulados ao longo das etapas seguintes desta atividade e também das próximas. O domínio desses conceitos iniciais deve facilitar a progressão do aprendizado nas próximas fases, uma vez que estarão presentes em todas elas.

Esses conceitos já haviam sido apresentados e tratados durante as aulas normais nas primeiras duas ou três semanas do curso. O texto para a leitura é relativamente pequeno e não explora o aprofundamento e detalhes dos conceitos colocados, ao contrário: a intenção é produzir uma visão do conjunto de conceitos e recursos que serão envolvidos nas tarefas da atividade. A leitura deve ocupar pouco tempo e espera-se que a realização das tarefas propostas na seqüência possa produzir um melhor domínio dos conceitos iniciais.

Sob a ótica da teoria da dialética ferramenta-objeto (Maranhão, 2008), essa primeira parte da atividade pode ser considerada como um dos elementos que irão compor o *antigo*. Esses são os elementos que, no âmbito dos recursos da linguagem algorítmica ou da linguagem de programação, serão necessários para apoiar as próximas construções.

4.1.6 Observações e análise *a posteriori* – primeira parte da atividade

Foi possível perceber que, durante essa leitura, alguns grupos, tanto na primeira como na segunda experimentação, buscaram o material de apoio às aulas normais para complementar a leitura. Esse comportamento indica que parte dos alunos se interessou por buscar algum aprofundamento ou algum detalhamento dos conceitos abordados.

Na primeira experimentação, verificou-se que os componentes do primeiro grupo produziram uma discussão mais longa sobre os conceitos de algoritmo e programa. Em uma parte dos diálogos, um dos componentes desse grupo afirmou que "*o algoritmo é uma receita que dá o raciocínio lógico para resolver [...]*", fica

evidente que o grupo percebeu a noção de algoritmo e seu sentido relacionado ao problema computacional.

Na segunda experimentação foram percebidas e anotadas duas declarações semelhantes: "*essa* (parte) *do algoritmo é que faz a solução, depois* (construção do programa) *é só completar*" e "*olhando o algoritmo você sabe a resolução*".

Na segunda experimentação, pelo menos três grupos encaminharam algumas discussões sobre os conceitos *variável* e *atribuição*. Em um dos diálogos houve a seguinte declaração: "*no comando de atribuição, ela* (a variável) *pega e leva para ela o resultado da conta* (expressão)". Em outro grupo, foi possível perceber a frase: "*quando ele* (o sistema) *chega nesse tipo de comando* (uma atribuição)*, faz o cálculo e guarda o valor naquele lugar* (a variável)". Em outro diálogo, em um terceiro grupo, foi dito: "*ele* (o comando de atribuição) *serve para limpar ela* (a variável) *e depois joga o novo dado*".

É interessante destacar que nessas três declarações foram empregadas expressões que revelam a constituição da noção de processo dinâmico, tanto a dinâmica do fluxo de processamento seqüencial ("*quando ele chega*"), quanto a dinâmica do fluxo de dados (*"depois joga o novo dado*" e "*guarda o valor naquele lugar*".

Na primeira experimentação, o segundo grupo conversou pouco durante a leitura e com o avanço dos trabalhos aumentou, aos poucos, a freqüência dos diálogos. O terceiro grupo praticamente nada discutiu em relação à leitura proposta. O tempo para a realização dessas fases iniciais da primeira atividade foi subestimado, o segundo grupo ocupou pelo menos 50 minutos até o início da segunda parte dessa atividade, e o previsto era um máximo de 30 minutos.

Na segunda experimentação, observou-se maior descontração e velocidade nos diálogos e discussões. Isso pode ser atribuído a dois fatores: a ausência de filmadoras e microfones, e a sala repleta de alunos, com mais ruídos, sons e movimentos. É possível afirmar que a presença dos equipamentos para filmagem e a pequena quantidade de alunos inibiram significativamente os participantes na primeira experimentação.

4.1.7 Apresentação da segunda parte da atividade

Na segunda parte dessa primeira atividade foi colocada a seguinte proposta de problema e algumas questões. Os alunos receberam também um aplicativo com a ilustração do funcionamento de um algoritmo que representa o método de resolução do problema. O objetivo principal da presença desse aplicativo é favorecer o entendimento do aspecto dinâmico presente no algoritmo.

> Um fabricante de lâmpadas utiliza um dispositivo mecânico que prepara e embala cada unidade do produto a uma velocidade constante de 15 lâmpadas por minuto. Conhecendo-se a quantidade de lâmpadas de um lote que deve ser embalado, como determinar o tempo necessário para completar a operação? Esse tempo deverá ser expresso em horas, minutos e segundos.

As questões propostas foram as seguintes:

- Antes de colocar em execução o aplicativo, procure responder: se o lote possuir 5438 lâmpadas, qual será o tempo necessário para completar-se a operação? O seu resultado coincide com a resposta emitida pela execução do algoritmo?

- Como pode ser descrita a finalidade da instrução `qhoras←totmin div 60` disposta no algoritmo?

- Se as instruções
`totmin←qlamp div 15` e `resto←qlamp mod 15`
tiverem suas disposições invertidas, o algoritmo permanecerá correto? Justifique.

- E se forem invertidas as instruções `resto←qlamp mod 15` e `qseg←resto*4` ? Justifique.

4.1.8 Análise *a priori* – segunda parte da atividade

Nessa segunda parte da primeira atividade, o grupo deverá observar o aplicativo com a ilustração da execução passo a passo de um algoritmo.

A primeira questão proposta deve levar o grupo a resolver uma instância específica do problema (lote com 5438 lâmpadas), ou seja, o grupo deverá organizar e realizar uma pequena seqüência de operações aritméticas (cálculo de quociente, resto e produto) e, depois, colocar em execução o aplicativo de ilustração do algoritmo fornecido.

Espera-se que os alunos relacionem as suas ações – as operações que tenham realizado para obter o resultado da instância do problema – com aquelas descritas no texto do algoritmo e executadas com o funcionamento do aplicativo que representa o algoritmo. O trabalho com esse tipo de relação corresponde a uma prática de interação entre domínios, especificada na teoria da dialética ferramenta-objeto (Maranhão, 2008).

A segunda questão deve fazer convergir para a especificação dessa relação. Pode-se prever nessas tarefas o trânsito entre o domínio numérico (as operações aritméticas que o grupo deverá realizar) e o domínio das expressões algébricas que compõem as instruções do algoritmo, representado no aplicativo. Sob a visão da teoria dos registros de representação semiótica (Duval, 2008), o trabalho com essa segunda questão corresponde a uma conversão de registros: o registro em linguagem algorítmica e o registro em língua natural.

As outras duas questões devem levar a uma análise sobre a importância da ordem na seqüência de execução das instruções: o aluno deverá concluir que o fluxo de processamento fica determinado em função da disposição das instruções do algoritmo. O mecanismo dinâmico de execução do algoritmo fica definido a partir de uma representação estática que é o texto do algoritmo. A idéia dos aspectos dinâmicos deverá ser reforçada com a observação do aplicativo em execução, essa idéia pode não ser evidente no registro estático do algoritmo.

A contribuição pretendida, com a experimentação do aplicativo, tem o sentido de favorecer a constituição de um modelo mental (Moreira, 1996) adequado para os algoritmos, com o acréscimo de mais um aspecto ao registro do algoritmo; tal aspecto é a explicitação da noção de processo dinâmico que o registro do algoritmo deve representar.

4.1.9 Observações e análise *a posteriori* – segunda parte da atividade

Na segunda parte da atividade não foi solicitada a construção de algoritmo ou programa. Cada grupo deveria entender a proposta do problema e observar o mecanismo de funcionamento do algoritmo com a execução do aplicativo. O aplicativo ressalta a dinâmica do fluxo de processamento e do fluxo de dados que

deve ser relacionado ao registro do método de resolução que é descrito em linguagem algorítmica.

Na primeira experimentação, o segundo grupo registrou como resposta à primeira pergunta apenas: "*não, os segundos não coincidiram*" e a partir das anotações complementares (rascunhos), foi possível perceber que a origem do erro foi a falta de uma etapa das operações: a multiplicação do resto da primeira divisão (5438 por 15) por 4. Aparentemente, a pressa em seguir para as próximas questões não permitiu que a divergência de resultados fosse discutida e analisada, mas os estudantes perceberam a falha (com a execução do aplicativo que representava o algoritmo), no entanto não corrigiram o registro da respectiva resposta. Os outros dois grupos operaram corretamente e encontraram resultados que coincidiam com a resposta emitida pela execução do aplicativo.

Na segunda experimentação, nove dos dez grupos responderam que houve a coincidência entre seus resultados e a resposta do aplicativo. Em um único grupo, que utilizou uma calculadora, houve divergência, pois os cálculos foram encaminhados com aproximação do valor de um dos quocientes envolvidos. Apesar de não constar no relatório de respostas, como o grupo percebeu a divergência, solicitou orientação do professor e conseguiu perceber e entender a origem da divergência.

Nos trabalhos de outros dois grupos foi possível notar que houve uma revisão da seqüência de operações indicada inicialmente, em função de terem encontrado resultados diferentes daqueles que o aplicativo apresentava.

Tanto na primeira como na segunda experimentação, a possibilidade de recorrer ao aplicativo foi um elemento auxiliar para o entendimento da proposta do problema e para a verificação de resultados. Foi possível notar que os grupos acionaram a execução do aplicativo pelo menos duas ou três vezes.

As outras questões dessa segunda parte foram respondidas corretamente, assim é possível afirmar que os alunos perceberam corretamente a importância da ordem de disposição das instruções no corpo do algoritmo, ou seja, a relação entre a organização seqüencial do registro do algoritmo e sua execução ao longo do tempo. O registro do processo de resolução do problema em linguagem

algorítmica, acrescido dos aspectos ressaltados pela animação, tornou mais evidente a importância da ordem de disposição das instruções.

Na primeira experimentação, o primeiro grupo respondeu corretamente às duas primeiras questões. Atribuiu à instrução `qhoras←totmin div 60` o seu sentido correto, com a conversão adequada do registro em linguagem algorítmica para o registro em língua natural do significado envolvido.

O grupo respondeu à terceira e à quarta questões assim: "*não pois são instruções para diferentes cálculos, um não depende do outro*" e "*sim pois utilizamos o valor obtido na variável resto para executar a próxima instrução*", o que leva a concluir que o grupo não fez a leitura correta das perguntas colocadas, mas percebeu a possível inversão de ordem entre as duas instruções colocadas na terceira questão e indicou a falha que seria introduzida com a inversão descrita na quarta questão.

Os outros dois grupos também perceberam corretamente a importância da ordem de disposição das instruções no algoritmo. As respostas do terceiro grupo foram: "*sim, pois elas são independentes, o conjunto de uma não afeta a outra*" e "*não, pois elas são dependentes, o conjunto de resto←qlamp mod 15 será utilizado em qseg← resto*4; a inversão altera o algoritmo*". O segundo grupo anotou as seguintes respostas: "*sim, pois o cálculo do resto é indiferente podendo inverter sua posição e resultado continuará o mesmo*" e "*não, pois não existirá valor alocado na variável resto*".

É interessante observar que na elaboração dos registros escritos dessas respostas os grupos não fizeram referência ao significado das variáveis ou expressões que figuram no registro do algoritmo, mas nos registros de fala foi recorrente a associação entre os elementos do algoritmo e seus significados no âmbito do problema enunciado e seu processo de resolução.

Foi possível observar que os três grupos realizaram várias execuções do aplicativo que ilustra o mecanismo do processo definido pelo algoritmo, inclusive com a tentativa de testes em que a quantidade de lâmpadas fornecida era um valor incompatível com a situação do problema (valor não inteiro ou valor inteiro negativo).

Nesses casos, a execução do aplicativo ou foi interrompida ou foi apresentado um conjunto de resultados incoerentes. Nenhum dos grupos chegou a questionar esse comportamento do aplicativo, mas isso leva a pensar sobre o interesse em acrescentar alguma forma de verificação e validação do valor (quantidade de lâmpadas) informado pelo usuário.

O segundo grupo discutiu sobre a velocidade da execução do aplicativo: um dos alunos declarou que tal execução era um pouco lenta, o outro aluno respondeu que aquela velocidade estava adequada, pois se a execução fosse mais rápida talvez não fossem visíveis os efeitos das instruções sobre os conteúdos das variáveis. Essas discussões reforçam a idéia de que a ilustração do mecanismo do algoritmo é um meio interessante para favorecer a constituição da noção de processo dinâmico presente no algoritmo. A animação agrega ao registro em linguagem algorítmica, que é um registro estático, a noção da dinâmica do processo de resolução.

Ao final do encontro, quando questionados sobre a ilustração dada pelo aplicativo, os estudantes declararam ter sido interessante essa possibilidade, pois dessa forma fica mais clara a ação correspondente a cada instrução do algoritmo, nesse questionamento nenhum dos alunos fez menção específica à ligação entre a ordem de disposição das instruções no algoritmo e a seqüência de execução das mesmas ao longo do tempo, parece que essa ligação é vista com naturalidade, e ao menos nesse caso, em que o algoritmo é uma seqüência simples de instruções (sem estruturas lógicas de seleção ou repetição), a relação entre a ordem de disposição no texto e a ordem de execução das instruções não acarreta dificuldade.

Na segunda experimentação, as respostas escritas foram semelhantes às que foram observadas na primeira experimentação. Em um dos grupos houve a seguinte declaração: "*assim* (com a execução do aplicativo) *dá para ver melhor como* (o algoritmo) *funciona*".

Em outro grupo, um dos alunos, apontando com o lápis uma instrução do algoritmo em execução, declarou: "*olha agora, é esse passo* (essa expressão na instrução de atribuição) *que leva o valor da* (variável) *qseg*". Essas declarações confirmam a idéia de que a animação do algoritmo, exibida pelo aplicativo,

favorece a percepção da noção de dinamismo, relativa tanto ao mecanismo do fluxo de processamento como ao mecanismo do fluxo de dados.

Mesmo para a situação de fluxo de processamento exclusivamente seqüencial, é possível afirmar que a animação produzida pelo aplicativo acarreta os efeitos esperados: melhora o entendimento da noção de processo dinâmico, e aprofunda os significados que o estudante deve associar ao registro de representação do processo, construído em linguagem algorítmica.

4.1.10 Apresentação da terceira parte da atividade

Na terceira parte dessa atividade foi proposto o problema a seguir e foram colocadas algumas questões.

> Com uma balança de dois pratos (balança do tipo gangorra) deseja-se confirmar a massa de uma porção com 387g de farinha. Para essa confirmação estão disponíveis, além da balança, alguns elementos de referência: vários elementos com massa 100g, outros com massa 30g, outros com 5g e outros com 1g.

As questões propostas foram as seguintes:

- Descreva a escolha de elementos de referência que você utilizaria para confirmar a pesagem com a colocação dos elementos de referência apenas sobre o prato vazio da balança. Há outras possibilidades de escolha? Há alguma delas que possa ser classificada como a mais adequada?
- Se em vez de 387g fossem 448g de farinha, como deveria ser a escolha dos elementos de referência?
- Agora procure descrever as relações aritméticas entre a quantidade de farinha e as quantidades de cada tipo de elemento de referência.
- Considerando-se que você tem em mãos uma calculadora simples (quatro operações) e uma caneta e uma folha de papel para anotações, procure descrever detalhadamente a seqüência de operações que você deve realizar, com esses recursos (calculadora, caneta e papel), para determinar as quantidades de cada tipo de elemento de referência para uma quantidade genérica **qf** de farinha.
- Com a mesma finalidade (determinar as quantidades de cada tipo de elemento de referência para uma quantidade genérica **qf** de farinha), procure descrever uma seqüência de operações considerando-se que a calculadora opere apenas com valores inteiros (adição, subtração, multiplicação e divisão com cálculo de quociente e cálculo de resto).
- Faça a descrição do algoritmo (utilize as formas de instruções já apresentadas) correspondente à seqüência de operações obtida na questão anterior.

- Indique o significado, diante da proposta do problema, de cada variável que você empregou no algoritmo.
- Faça a implementação e testes do programa correspondente ao algoritmo que você construiu.
- Se a porção de farinha possuir apenas 58g, o resultado obtido a partir da execução do seu programa é válida?
- Observe a seqüência de linhas que constituem o seu programa e procure responder: a única ordem de disposição de linhas correta é essa que figura no seu programa ou é possível alguma inversão? Se for possível alguma inversão, indique algumas possibilidades.

Na segunda experimentação, foram feitas algumas modificações nessa seqüência de questões. As modificações foram resultado das observações realizadas com a primeira experimentação, e tiveram como objetivo reduzir o tempo necessário para a sua realização e a simplificação de alguns aspectos que, com a primeira experimentação, mostraram-se desnecessários ou pouco importantes.

Na primeira experimentação, foram colocadas as três questões:

- Agora procure descrever as relações aritméticas entre a quantidade de farinha e as quantidades de cada tipo de elemento de referência.
- Considerando-se que você tem em mãos uma calculadora simples (quatro operações) e uma caneta e uma folha de papel para anotações, procure descrever detalhadamente a seqüência de operações que você deve realizar, com esses recursos (calculadora, caneta e papel), para determinar as quantidades de cada tipo de elemento de referência para uma quantidade genérica **qf** de farinha.
- Com a mesma finalidade (determinar as quantidades de cada tipo de elemento de referência para uma quantidade genérica **qf** de farinha), procure descrever uma seqüência de operações considerando-se que a calculadora opere apenas com valores inteiros (adição, subtração, multiplicação e divisão com cálculo de quociente e cálculo de resto).

e, na segunda experimentação, essas foram reduzidas a uma única:

> Agora procure descrever a seqüência de relações aritméticas (cálculos aritméticos) empregadas para obter as quantidades de cada tipo de elemento de referência a partir da quantidade de farinha.

pois verificou-se que as respostas produzidas eram praticamente equivalentes e exigiam as mesmas discussões e construções.

A questão

- *Indique o significado, diante da proposta do problema, de cada variável que você empregou no algoritmo.*

foi agregada à proposta de construção do algoritmo, com a seguinte redação:

Faça a descrição de um algoritmo (utilize as formas de instruções já apresentadas) que represente um método de resolução do problema proposto. Escolha identificadores de variáveis que indiquem os significados das informações correspondentes.

4.1.11 Análise *a priori* – terceira parte da atividade

A proposta do problema inicial, uma instância do caso geral proposto mais adiante, deve levar a uma seqüência de operações que depois deverá ser generalizada. Mais uma vez, a ordem cronológica das operações deve ser percebida como essencial para a resolução correta do problema. Espera-se que o grupo entenda e declare como melhor escolha aquela que corresponde ao emprego da menor quantidade de elementos de referência. As primeiras questões exigem o trabalho do grupo com operações aritméticas (casos particulares do problema computacional que será proposto na seqüência), a organização dessas operações aritméticas poderá facilitar a organização da série de instruções que irão compor o algoritmo e o programa que deverão ser construídos. Essas questões envolvem a passagem por domínios distintos: o domínio numérico-aritmético e o domínio algébrico-algorítmico.

Com a terceira questão colocada espera-se favorecer a generalização que será proposta na seqüência desta parte da atividade. Com as três próximas questões, pretende-se que os estudantes consigam estabelecer a generalização do processo, e também, ao menos num grau inicial, consigam obter uma descrição organizada do método de resolução.

Outra intenção é levar o grupo a perceber as diferenças entre operar com uma calculadora comum (operações aritméticas com valores racionais) e operar com a aritmética para valores inteiros. Os grupos terão à disposição uma calculadora simples, além de caneta e papel para anotações. Por outro lado, os alunos devem perceber que o trabalho com casos particulares do problema pode ser considerado como apoio no desenvolvimento do método de resolução geral.

Essa primeira série de questões deve organizar as fases iniciais descritas no processo da dialética ferramenta-objeto: retomada dos conceitos iniciais (o *antigo*: variável, instrução de atribuição, operadores aritméticos, instrução de

entrada e instrução de saída, além dos elementos aritméticos e algébricos) e primeiros encaminhamentos de construção do método de resolução (*pesquisas*).

No trabalho com as três últimas questões, espera-se que os estudantes discutam a organização das ações já encaminhadas na fase anterior e percebam a importância de organizar a ordem adequada de ações que serão representadas no algoritmo e depois no programa. Isso corresponde a considerar-se a vinculação entre a organização seqüencial do texto do algoritmo e a noção de fluxo de processamento e fluxo de dados.

Nessa fase os alunos devem trabalhar diretamente com pelo menos três formas de registros de representação do método de resolução: o primeiro registro com o emprego dos elementos que cada estudante selecionou (língua natural, expressões algébricas, esboços com elementos gráficos), e os outros dois com o emprego da linguagem algorítmica e da linguagem de programação. As conversões entre as formas de registros, além daquelas que se pode indicar inicialmente como construções principais (esboços→algoritmo e algoritmo→programa), devem ocorrer também com o envolvimento dos outros pares e sentidos (esboços←→algoritmo, algoritmo←→programa e esboços←→programa), nos momentos em que os alunos estiverem tratando as explicações, justificativas e argumentações que embasaram as construções.

Com essa outra série de questões deve ocorrer o desdobramento das próximas fases do processo da dialética ferramenta-objeto: *explicitação* marcada principalmente pelas discussões e trocas de idéias entre os membros do grupo na busca do esboço do processo de solução; o *novo implícito* representado pelas primeiras tentativas de descrições organizadas, e validação do processo de resolução (registros em língua natural e em linguagem algorítmica) e o início de sua *institucionalização* no interior do grupo, com o conjunto de justificativas e explicações, no sentido de se partilhar a produção obtida.

Segundo Maranhão (2008), a fase *institucionalização* deve corresponder à intervenção do professor, diante do conjunto de todos os alunos da classe, com o objetivo de destacar os aspectos essenciais dos objetos constituídos. A forma que foi planejada para as atividades não permitiu que a *institucionalização* fosse completada durante a realização de cada uma delas.

A questão proposta após a implementação do programa, deve conduzir a uma discussão sobre a resposta produzida para um caso em que a massa da farinha seja menor do que 100 g. O grupo deve perceber que, diferentemente do que seria a respectiva ação real de determinar as quantidades de elementos de referência, o mecanismo representado no algoritmo irá efetivamente calcular quantos elementos de 100 g devem ser utilizados, inclusive na situação em que essa quantidade é nula. Esse fato deve reforçar a idéia de fluxo de processamento seqüencial e invariante que o algoritmo representa.

Com a última questão, o grupo deverá refletir sobre a vinculação entre a disposição das instruções do programa e a ordem de execução. Espera-se que os alunos percebam que apenas algumas inversões são possíveis, mesmo que sejam pouco naturais, e que as instruções que correspondem propriamente ao mecanismo de cálculo talvez não permitam inversões. O algoritmo deverá envolver um encadeamento de operações de divisão, com cálculo do quociente inteiro e cálculo do resto, cujo valor deverá ser o dividendo na operação seguinte.

Uma alternativa pouco natural será repetir operações já realizadas, sempre a partir da quantidade de farinha cuja massa deseja-se confirmar, nesse caso as operações não seriam encadeadas e assim seriam possíveis várias inversões de ordem. Outra possibilidade, pouco natural mas correta, seria o cálculo do resto de uma divisão antes do cálculo do quociente da mesma.

A intenção é ressaltar a relação entre a organização do texto do algoritmo e sua execução ao longo do tempo, ou seja: valorizar o aspecto dinâmico que deve ser associado ao registro estático do método de resolução, descrito em linguagem algorítmica ou em linguagem de programação. A associação texto estático-processo dinâmico é um componente importante na constituição nos modelos mentais que os estudantes desenvolvem.

4.1.12 Observações e análise *a posteriori* – terceira parte da atividade

Na terceira parte da atividade foi proposto mais um problema computacional (o problema da farinha) e fornecido um aplicativo que produz, a cada execução, a resposta a um caso do problema. Nesse aplicativo não foi

colocada a representação do algoritmo, nem a representação das variáveis utilizadas. O aplicativo, inicialmente, apoiou o entendimento da proposta do problema, e depois forneceu referência para as tarefas de testes e validação do algoritmo e do programa produzidos pelo grupo.

O conjunto de questões colocadas inicialmente tinha o propósito de conduzir os trabalhos do grupo para encaminhamento da construção do método de resolução do problema, assim: mobilizar conceitos já sob domínio dos alunos, e iniciar as tentativas de construção do método de resolução (mobilizar o *antigo* e promover as *pesquisas,* no processo da dialética ferramenta-objeto).

Em relação à primeira questão proposta, tanto na primeira como na segunda experimentação, em todos os grupos houve consenso sobre a possibilidade de se classificar uma das escolhas como a mais adequada.

Um dos alunos, do terceiro grupo da primeira experimentação, chegou a cogitar a alternativa de se colocar elementos de referência também no prato da balança em que foi disposta a farinha, para a otimização da quantidade de massas de referência utilizadas na confirmação da pesagem.

Esse terceiro grupo declarou que seria desnecessário responder às duas últimas perguntas da primeira seqüência, com a argumentação de que na terceira pergunta a resposta produzida já contemplava as respostas das outras duas questões, o registro elaborado pelo grupo já constituía um esboço do método de resolução.

Nessa fase das tarefas, durante a segunda experimentação, um dos alunos declarou "*acho que descobri como resolve ... é parecido com o outro*", em seguida passou a descrever as operações de cálculo necessárias, de forma organizada mas ainda incompleta. Isso caracteriza as fases *antigo* e *pesquisas* da dialética ferramenta-objeto. Foi possível perceber que pouco tempo depois disso, os outros componentes do grupo já haviam entendido as linhas gerais de construção propostas pelo colega, e já iniciavam o registro da resposta escrita.

Para a maioria dos outros grupos, essa parte das tarefas exigiu mais tempo de discussões, e, diante do que foi observado, a conclusão é que os estudantes conseguiram estabelecer as associações entre a solução do caso particular do problema e a organização da série de operações que configura o processo de

solução geral, e depois conseguiram produzir as conversões de registros para completar a construção do algoritmo e do programa.

Foi possível observar que um dos grupos, na segunda experimentação, depois de ter construído o programa, percebeu que havia divergência entre a resposta apresentada pelo programa e aquela produzida pelo aplicativo, e então retomou a descrição do algoritmo para a respectiva correção. O aplicativo a finalidade de fornecer referência no momento de verificação do funcionamento adequado do programa constuido.

Na primeira experimentação, com a seqüência das três questões:

- *Agora procure descrever as relações aritméticas ...*
- *Considerando-se que você tem em mãos ...*
- *Com a mesma finalidade ... ,*

a intenção era delimitar e dirigir os trabalhos e discussões no sentido de se obter o processo de resolução. Para o terceiro grupo, a resposta correspondente ao solicitado na primeira dessas questões já indicava o processo de resolução em um grau de organização e detalhe suficiente para a descrição formal do algoritmo. Esse grupo concluiu corretamente as tarefas de elaboração do algoritmo e implementação do programa, produziu o registro em linguagem algorítmica (não registrou o texto do programa produzido no caderno da atividade, mas exibiu a construção e a execução do mesmo durante o encontro).

Antes de iniciar o complemento da primeira atividade, um dos componentes desse terceiro grupo teve que se retirar, dessa forma o outro aluno trabalhou individualmente nessa última tarefa, e produziu corretamente o que foi solicitado. Os outros dois grupos, da primeira experimentação, não tiveram tempo para realizar a tarefa colocada como complemento da primeira atividade.

Na primeira experimentação, o primeiro e o segundo grupo responderam adequadamente às duas primeiras questões e explicitaram as escolhas consideradas mais adequadas, o segundo grupo registrou ainda outras alternativas de escolha, e apontou a que foi considerada como mais adequada.

Apesar de algumas falhas nos registros das respostas às próximas duas questões, o segundo grupo produziu os sentidos corretos das mesmas. As últimas questões, em que eram solicitadas a construção do algoritmo e do programa,

foram respondidas de maneira correta, pelo segundo grupo, isso evidencia que apesar das dificuldades nos registros das respostas às questões anteriores, o grupo conseguiu produzir e representar corretamente o processo de resolução do problema. O primeiro grupo também encontrou dificuldade ao registrar as respostas às terceira e quarta questões, não chegou a responder a quinta questão; apesar disso o grupo apresentou corretamente o texto do algoritmo e isso reforça a observação de que apesar dessas dificuldades, ao registrar as respostas anteriores, essas etapas são importantes para alcançar a produção do processo de resolução.

Além de retomarem o *antigo*, os grupos passaram pelas fases de *pesquisas*, *explicitação e novo implícito* do processo da dialética ferramenta-objeto.

O primeiro grupo completou a construção do programa depois de encerrado o encontro, dessa forma o texto do programa não foi registrado no caderno da atividade, mas foi enviado posteriormente por correio eletrônico.

Ao final do encontro da primeira experimentação, dois alunos que ainda permaneciam na saída da sala, descreveram como proveitosa a oportunidade de realizarem o trabalho dessa atividade, e sugeriram que também nas aulas normais o professor ofereça esses tipos de aplicativos de apoio didático.

4.1.13 Apresentação do complemento da atividade

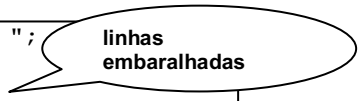
Como complemento a essa primeira atividade foi colocado o problema:

> A contagem de pontos em um torneio de futebol foi definida da seguinte maneira:
>
> vitória por goleada – 6 pontos
> vitória simples – 3 pontos
> empate – 1 ponto
> derrota – nenhum ponto
>
> Como determinar a quantidade mínima de jogos para que um time complete exatamente uma dada pontuação?
> Por exemplo: qual a quantidade mínima de jogos para que um time complete exatamente 28 pontos?
> Resposta: o time deve realizar no mínimo 6 jogos e obter: 4 vitórias por goleada, 1 vitória simples e 1 empate.

Tarefa: as instruções descritas a seguir devem compor um programa que represente um método de resolução do problema, porém estão dispostas fora de ordem, faça a organização dessas linhas de maneira a obter um programa adequado.

```
#include <iostream>
using namespace std;

int main( ){
  int totpontos, goleadas, vitsimples, empates, quantjogos;
  cout<<"digite a quantidade desejada de pontos: ";
  cin>>totpontos;
    cout<<vitsimples<<" vitoria(s) simples, ";
    quantjogos=goleadas+vitsimples+empates;
    totpontos=totpontos%6;
    vitsimples=totpontos/3;
    cout<<goleadas<<" goleada(s), ";
    goleadas=totpontos/6;
    cout<<"quantidade minima de jogos: "<<quantjogos<<endl;
    cout<<empates<<" empate(s)."<<endl;
    empates=totpontos%3;
  system("PAUSE");
  return(0);
}
```



Observação: esse critério de pontuação não é real, mas há alguns anos foi aplicado (campeonato paulista no início da década de 1990) um critério semelhante ao descrito no problema, assim: três pontos para vitória com pelo menos dois gols de diferença, dois pontos para vitória por um gol, um ponto para empate e nenhum ponto para derrota.

4.1.14 Análise *a priori* – complemento da atividade

Com o trabalho desta tarefa pretende-se que o grupo consiga imaginar e relacionar o método de resolução do problema ao conjunto de instruções fornecido fora de ordem.

O esperado é que o grupo consiga perceber, na lista de instruções fornecida, a presença dos significados que são correspondentes às operações necessárias para a resolução, e que a partir daí possa reordenar a lista de maneira a descrever adequadamente o método de solução. Uma parte das instruções do texto do programa não se refere ao processo de resolução e sim à forma geral de composição de um programa e à operação de entrada de dado (pontuação que o time deve completar), essa coleção de instruções está disposta

corretamente, assim os alunos deverão se concentrar no trabalho com a série de instruções que fazem a composição do processo de resolução do problema. Esse trabalho envolve a manipulação do registro em linguagem de programação e exige o trabalho de conversão quando o grupo buscar os significados que devem ser associados a cada instrução do programa.

O desenvolvimento dessa parte da atividade exige a noção de processo dinâmico do fluxo de processamento, vinculado à disposição das instruções componentes do registro em linguagem de programação, e também a dinâmica dos dados que representam as informações (pontuação, quantidade de vitórias por goleada, quantidade de vitórias simples, quantidade de empates e quantidade de partidas). Nessa parte da atividade espera-se que o grupo trabalhe com a conversão de registros no sentido inverso, ou seja: a partir do registro em linguagem de programação, os alunos deverão produzir os registros em língua natural e registros algébricos, ao menos para que possa haver a comunicação sobre o sentido ou significado das instruções que devem ser reorganizadas, e então retornar ao registro em linguagem de programação para elaborar a reordenação das linhas do programa.

Essa última parte da primeira atividade não foi colocada na segunda experimentação. Durante a primeira experimentação, apenas um aluno de um dos grupos pode completar essa tarefa, para os outros dois grupos as tarefas anteriores ocuparam um tempo maior do que havia sido estimado inicialmente e dessa forma não puderam trabalhar nessa última parte. Na segunda experimentação, a restrição do tempo para cada atividade foi mais rígida pois os trabalhos foram desenvolvidos durante o tempo de aulas normais, com a limitação tanto dos horários de funcionamento dos laboratórios, como da disponibilidade dos mesmos.

O único aluno que trabalhou nessa última parte da primeira atividade conseguiu chegar à solução. Pelo que foi possível observar, o programa correspondente foi implementado e depois de três ou quatro tentativas o aluno já havia encontrado a ordem correta de disposição das instruções.

4.2 Análises – segunda atividade

4.2.1 Apresentação da primeira parte da atividade

A primeira parte dessa atividade é a proposta de um texto para leitura com a apresentação do tipo de dados real (*float* e *double*) e a retomada do conceito de variável. O texto proposto para leitura é o seguinte:

> Variáveis e tipos primitivos numéricos
>
> Os sistemas de linguagem de alto nível, entre eles C ou C++, possuem alguns tipos primitivos de dados, tais tipos especificam a natureza das informações (valor numérico, valor lógico, cadeia de caracteres) que podem ser representadas com sua utilização e definem também os operadores (operadores aritméticos, operadores de relação de ordem, operadores lógicos, operadores para tratamento de cadeias de caracteres) que podem ser empregados na manipulação dos dados correspondentes.
>
> Na primeira e também nesta segunda atividade as tarefas propostas envolvem a utilização dos tipos numéricos e respectivos operadores aritméticos. As tarefas da primeira atividade exigem a utilização de variáveis e constantes de tipo numérico inteiro enquanto nessa segunda atividade a maioria dos valores envolvidos deve ser representada por variáveis e constantes de tipo numérico real.
>
> O sistema computacional armazena os valores de variáveis em conjuntos finitos de células de memória - 4 *bytes* para o tipo *float* ou 8 *bytes* para o tipo *double* – dessa forma, apesar de serem denominados tipos reais, os valores efetivamente armazenados e tratados são valores racionais com representação finita, assim os valores reais ou valores racionais com representação não finita são registrados por aproximações de valores racionais com representações finitas. O tipo *float* admite valores com até 8 dígitos significativos e ordem de grandeza de 10^{38}, o tipo *double* trabalha com até 15 dígitos significativos e ordem de grandeza 10^{308}. As formas de representação interna, tanto para o tipo inteiro como para o tipo real, são orientadas por padrões definidos internacionalmente para que se alcance algum grau de uniformidade entre os vários sistemas e com isso tornar possível o trânsito de dados entre sistemas distintos.
>
> As variáveis de tipos numéricos são empregadas para representar informações como medidas, taxas, quantidades, valores monetários, etc.. Em algumas situações o tipo inteiro (*int*) é mais adequado, em outras o tipo real, é a natureza da informação a ser tratada que indica o tipo adequado, por exemplo: no problema do torneio de futebol é mais adequado representar a quantidade de pontos ou a quantidade de partidas por variáveis de tipo numérico inteiro, naquela situação não cabe imaginar frações de pontos ou frações de partidas; além da própria natureza das informações, os mecanismos de cálculo foram constituídos a partir de operações de divisões na aritmética inteira (quociente inteiro e resto de divisão). Na aritmética de valores reais (*float* ou *double*) os operadores (adição, subtração, multiplicação e divisão) são os usuais, com aquele mecanismo que se encontra nas calculadoras convencionais. Numa situação em que as informações tratadas forem, por exemplo, preços de produtos, o mais adequado é representá-las

por variáveis de tipo numérico real, pois o esperado é poder representar parcelas não inteiras (centavos) entre tais valores.

O sistema de linguagem C ou C++ possui, além dos operadores aritméticos, outras ferramentas para o tratamento de valores numéricos que essencialmente são as funções matemáticas (trigonométricas, exponencial, logaritmo, potenciação, etc.), essas funções são organizadas em um elemento complementar do sistema de linguagem, a biblioteca de funções matemáticas cmath. Sempre que houver a necessidade de utilizar-se alguma dessas funções deve ser disposta a diretiva de inclusão (**`#include <cmath>`**) no topo do texto do programa.

Nas operações aritméticas que envolverem mistura de tipos numéricos, um operando de tipo inteiro e outro de tipo real, o sistema de linguagem realiza automaticamente a conversão do valor de tipo inteiro para o correspondente valor real e depois opera os valores com a aritmética real, esse mecanismo de conversão automático ocorre sempre no sentido do tipo mais restrito para o tipo mais amplo. Em uma instrução de atribuição em que a variável alvo é de tipo inteiro e o resultado da respectiva expressão é de tipo real, o sistema faz o truncamento do valor real obtido como resultado da expressão (a parte não inteira do valor é perdida) antes de completar a atribuição; em outros sistemas de linguagem (Pascal e Java, por exemplo) esse comportamento pode ser diferente.

As tarefas da segunda atividade envolvem a utilização dos mesmos recursos (linguagem algorítmica e linguagem de programação) já empregados na primeira atividade: instruções de entrada e de saída, e instruções de atribuição com o emprego de variáveis de tipos numéricos e seus operadores. Para a declaração de variáveis numéricas de tipo real utiliza-se no programa, por exemplo: **`float area;`** onde **`float`** indica o tipo real e **`area`** é o identificador da variável declarada; como efeito dessa instrução o sistema realiza a alocação de um conjunto de células de memória (4 bytes) e faz a vinculação do endereço desse conjunto de células ao identificador da variável **`area`**, aquele conjunto de células terá capacidade para armazenar um valor real (float) a cada momento durante a execução do programa.

Além da leitura foram propostas as seguinte questões:

- No âmbito de um algoritmo ou programa, como pode ser descrito o papel de uma variável?

- Que ações do sistema computacional podem ser relacionadas à execução do seguinte comando de atribuição: **`valor ← valor+1`** ?

4.2.2 Análise *a priori* – primeira parte da atividade

O objetivo com essa primeira parte da segunda atividade é levar o grupo a entender que além do tipo inteiro (*int*), o sistema de linguagem oferece outra categoria de tipo numérico que é denominada tipo numérico real (*float* ou *double*); é importante que o estudante perceba que apesar de se ter tipos numéricos nas

duas categorias, suas naturezas são distintas, bem como os operadores aritméticos de divisão apresentam comportamentos específicos; como conseqüência, a escolha por um ou por outro tipo numérico para alguma variável deve exigir uma pequena análise sobre a natureza da informação que se pretende representar pela variável.

Outro objetivo, nessa primeira parte, é ampliar o entendimento do conceito de variável e da instrução de atribuição. Foram colocadas duas questões em seqüência à leitura proposta inicialmente: uma para provocar a reflexão do grupo acerca do papel de uma variável, em um algoritmo ou em um programa, e outra relativa ao mecanismo representado por uma instrução de atribuição.

4.2.3 Observações e análise *a posteriori* – primeira parte da atividade

Na primeira experimentação, a segunda atividade foi realizada por dois grupos de três alunos, e na segunda experimentação por onze grupos, dois deles com dois alunos e nove com três alunos.

Na primeira experimentação, o segundo grupo teve como componentes três estudantes que participaram da primeira atividade, e o primeiro grupo foi composto por dois alunos que não participaram da primeira atividade e um aluno que realizou aquela primeira atividade. Esses dois alunos que não haviam participado do primeiro encontro receberam uma breve orientação sobre os conceitos que foram abordados na atividade anterior e também sobre a composição da série de tarefas e a forma de atuação esperada durante a realização. Na segunda experimentação, dois alunos que não participaram da primeira atividade receberam orientações semelhantes.

Na primeira experimentação, logo após os grupos terem iniciado a primeira tarefa proposta, leitura do texto introdutório à segunda atividade, observou-se no primeiro grupo um breve diálogo com o sentido de verificar a compreensão correta dos primeiros conceitos apresentados: um dos alunos afirmou "*ele fala aqui dos números diferentes que podem aparecer ...*", outro componente concorda e declara "*no outro dia era só inteiro ...*". Depois dessas conversas, a leitura prosseguiu praticamente sem qualquer outro diálogo, apenas algo como

"*podemos seguir?*". O segundo grupo logo solicitou a presença do professor, pois estavam confusos em relação às ações do sistema computacional de alocar e armazenar.

Esse segundo grupo também completou a leitura inicial quase sem diálogos, mas depois, ao discutirem as duas questões propostas, retornaram à leitura do texto e nesse momento ocorreram vários diálogos que envolviam tanto a construção das respostas às questões como alguns aspectos dos conceitos apresentados no texto.

Entre os diálogos desse segundo grupo, é interessante destacar uma parte, relativa à atribuição `valor←valor+1`, em que um dos alunos declara: "*ele vai usar a variável valor como que de uma variável auxiliar, ela tem um valor antes da execução e muda depois*" e o outro aluno completa: "*o valor muda, mas vai ser no mesmo espaço*". As respostas registradas no caderno, pelo segundo grupo foram: "*O papel da variável é o armazenamento de uma informação ou valor*" e "*o sistema atribui à variável valor, a quantidade armazenada anteriormente nessa mesma variável mais 1.*".

Dessa forma, é interessante observar que a noção processo dinâmico vinculada à atribuição `valor←valor+1` foi expressa intensamente nos registros de fala e de forma menos evidente nos registros escritos.

Na segunda experimentação foi observado fato semelhante. Em um dos grupos, ao conversarem sobre a atribuição `valor←valor+1`, foi notado o gesto de um dos alunos, que indicava o transporte, com uma das mãos, de algo (o resultado da expressão `valor+1`) para o destino que era indicado pela outra mão (a variável `valor`).

O segundo grupo da primeira experimentação, antes do consenso sobre a resposta (registro escrito) que seria dada à segunda questão, ainda discutiu bastante em busca da clareza da resposta. O primeiro grupo respondeu às duas questões assim: "*O papel de uma variável em um programa ou algoritmo é armazenar dados a serem manipulados.*" e "*pega o valor original da variável, soma 1, depois atribui de volta para a variável o novo valor.*", nessa segunda resposta é evidente a idéia de dinamismo para descrever a ação do sistema, mesmo no registro escrito.

Entre as respostas (registros escritos) elaboradas na segunda experimentação, foi possível observar forma de expressão semelhante ("*o sistema pega o valor...*") em um dos grupos. Em um dos rascunhos foi observada uma figura com um retângulo, um quadrado e uma seta em curva com o sentido do retângulo para o quadrado, tal figura representa a atribuição (abaixo do quadrado está escrito `valor`).

Essas observações permitem afirmar que, mesmo no registro de uma instrução isolada, o aluno pode buscar a interpretação do aspecto dinâmico correspondente.

4.2.4 Apresentação da segunda parte da atividade

Na segunda parte dessa segunda atividade, foi colocada a proposta do problema do transporte de carvão, e fornecido um aplicativo com a animação do algoritmo que representa um método de resolução do mesmo.

> Proposta do problema:
> O transporte de carvão, desde a mina em que é feita a extração, até a siderúrgica em que o carvão é consumido, é realizado em duas etapas: por ferrovia da mina até um porto marítimo, e depois por um navio até a siderúrgica; nessas operações de transporte ocorrem perdas: 1,7% do carvão extraído da mina é perdido na primeira etapa e 0,8% do carvão que chega ao porto é perdido na segunda etapa do transporte. Conhecendo-se a quantidade (toneladas) de carvão encomendada pela siderúrgica como determinar a quantidade necessária de carvão que deve ser extraída da mina para atender tal encomenda?
>
> Questões:
>
> - Execute o aplicativo para responder: se a siderúrgica encomendar 12,5t de carvão, qual deverá ser a quantidade extraída da mina?
> - Determine a quantidade de carvão que será entregue à siderúrgica se o operador da mina extrair e carregar o trem com 1,5t de carvão. Para responder a esta questão faça algumas execuções do aplicativo para buscar uma aproximação da quantidade que será entregue, por exemplo: execute o aplicativo colocando 1,45t como quantidade a ser entregue na siderúrgica e depois procure melhorar a aproximação com quantidades a entregar mais adequadas; anote suas tentativas.
> - As linhas 2 e 3 do algoritmo podem ter suas disposições invertidas? O algoritmo permanecerá correto? Justifique.

4.2.5 Análise *a priori* – segunda parte da atividade

Essa segunda parte da atividade é colocada com a finalidade de levar o aluno a observar o mecanismo do algoritmo – fluxo seqüencial – com uma instrução de entrada, duas instruções de atribuição e uma instrução de saída. No âmbito do processo de resolução, estabelecido pelas expressões algébricas, o estudante deve perceber que as taxas de perdas incidem sobre bases (volumes de carvão) diferentes, e tanto essas taxas como também as quantidades são representadas mais naturalmente por valores não inteiros.

O algoritmo de cálculo corresponde a duas atribuições: a primeira determina a quantidade de carvão que deve ser embarcada no navio, a partir do volume da encomenda realizada, e a segunda atribuição define a quantidade a ser extraída da mina em função da quantidade a ser embarcada no navio. Esse mecanismo deve ser percebido com a execução do algoritmo, em resposta às duas primeiras questões. É essencial a ordem de execução dessas atribuições.

A segunda questão propõe a utilização do aplicativo para a resposta a uma pergunta que inverte a situação que o algoritmo trata diretamente. Nessa questão a orientação é que o grupo procure determinar o valor da resposta por aproximação, com a utilização do aplicativo. Essa questão deve ressaltar a funcionalidade do algoritmo, assim: o dado que o processo recebe inicialmente é o volume encomendado, e o dado que o processo produz e exibe ao final de sua execução é o volume que deve ser extraído da mina.

A terceira questão proposta deve levar a uma reflexão sobre a relevância da ordem de disposição e a conseqüente ordem de execução das operações. Assim, o que se pretende é que o grupo perceba tanto a dinâmica de fluxo de processamento, como o mecanismo de fluxo de dados, relacionados à organização do texto do algoritmo.

4.2.6 Observações e análise *a posteriori* – segunda parte da atividade

A primeira questão proposta exigia a execução do aplicativo e a observação dessa execução. Os elementos da interface do aplicativo são suficientes para essa parte da tarefa.

Na primeira experimentação, os dois grupos responderam corretamente à questão. Um dos componentes do segundo grupo percebeu rapidamente o mecanismo descrito na proposta do problema, quando outro aluno desse grupo falou "*a primeira aqui eu já fiz (a conta), é 2,5% que é a soma de 1,7% de perda...*" o primeiro logo retrucou: "*não, não, quando ele perde 1,7 é do total que ele tinha antes e 0,8 do que sobrou depois, você não pode somar os dois.*", essa observação foi aceita de imediato, mas na seqüência o grupo ainda discutiu bastante sobre quais operações deveriam ser realizadas para compor o método de resolução.

Antes de observar o aplicativo que continha o algoritmo e a animação de sua execução, o grupo discutiu e procurou elaborar a estratégia de resolução, e chegou a perceber a possibilidade de associar às taxas os respectivos fatores, mas teve dificuldades ao planejar as operações de cálculo: deveriam multiplicar ou dividir? Fazer incidir o fator sobre uma quantidade ainda não conhecida? Foi possível observar que dois dos alunos do grupo entenderam a construção utilizada no aplicativo e explicaram o processo ao terceiro estudante.

Ainda na primeira experimentação, o trabalho do primeiro grupo ocorreu de modo diferente, a partir da observação do aplicativo o grupo buscou justificar as expressões de cálculo apresentadas, esse trabalho foi mais rápido do que aquele realizado pelo segundo grupo.

Nos dois casos, as discussões indicaram o trabalho de conversões entre registros: de um lado, a descrição do processo com elementos da língua natural e esboços com elementos de expressões algébricas, e de outro o processo descrito em linguagem algorítmica. O sentido principal da conversão foi do registro em linguagem algorítmica para o registro em língua natural, mas o outro sentido também foi mobilizado: percebeu-se que em alguns momentos houve a construção (ou reconstrução) de alguma instrução de atribuição, que acompanhava a discussão.

Na segunda experimentação, para três grupos, a fase de discussões sobre o problema e o algoritmo apresentado foi mais demorada. Essa fase exigiu esforço daqueles que haviam entendido o processo de resolução e tentavam explicar aos outros colegas. Nessa situação, várias vezes o aplicativo foi acionado como meio auxiliar nas explicações. Aparentemente, nos três grupos, o trabalho

só avançou depois de todos terem entendido o processo representado no algoritmo. A possibilidade de emprego do aplicativo favoreceu o trabalho de explicações e justificativas dos alunos sobre a proposta do problema e também sobre o método de resolução do problema.

A terceira questão, na primeira experimentação, foi respondida pelos dois grupos assim: "*não permanecerá correto pois uma linha depende da outra*" e "*não! pois a variável qporto armazena um valor já calculado, que se torna necessário para o cálculo da variável qextraida.*" Houve pouco diálogo na busca dessas respostas, mesmo para o segundo grupo que discutiu com mais detalhes a estratégia de resolução, nesse momento não havia dúvidas em relação ao encadeamento das operações. A discussão só ocorreu com o propósito de definir a forma da frase que seria colocada como resposta: já havia consenso sobre o sentido da resposta que seria registrada.

O pouco tempo necessário para a produção dessas respostas deixa claro que os participantes perceberam a importância da ordem de disposição das instruções no algoritmo e, em correspondência, a ordem de sua execução.

Na segunda experimentação, observou-se que, em todos os grupos, o aplicativo foi acionado várias vezes como suporte à busca da segunda resposta, como era esperado. Também nessa segunda experimentação foram observados vários diálogos, na maioria dos grupos, com a finalidade de esclarecer a construção do algoritmo apresentado. Alguns exemplos de expressões: "*precisa guardar o valor ...*", "*este valor vai ...*", "*primeiro faz entrar, depois as contas ...*"

Um dos grupos colocou como resposta várias associações entre quantidades de carvão, no sentido quantidade extraída → quantidade entregue mas com os valores invertidos, ou seja: com os pares de valores obtidos pela execução do aplicativo. Outro grupo elaborou como resposta uma aproximação pouco refinada, mas com o sentido correto.

Na segunda experimentação, o questionamento sobre a possibilidade de inverter-se a disposição das duas atribuições foi respondido corretamente por todos os grupos. A idéia de execução das ações ao longo do tempo, relacionada à ordem de disposição das instruções, foi entendida em todos os grupos.

A segunda parte dessa segunda atividade alcançou seus objetivos: o aplicativo foi empregado como suporte às discussões e argumentações, provocou a atividade de conversão de registros de representação e levou a uma associação adequada entre a ordem de disposição das instruções e o mecanismo de execução ao longo do tempo.

4.2.7 Apresentação da terceira parte da atividade

Na terceira parte da segunda atividade encontra-se a proposta do problema de cálculo do IPVA, o encaminhamento de solução é sugerido a partir de algumas questões. A proposta é acompanhada por um aplicativo que realiza o cálculo e exibe o valor do IPVA. O texto da proposta é o seguinte:

> Proposta do problema:
>
> O IPVA (Imposto sobre Propriedade de Veículos Automotores) é um imposto anual, dessa forma, para veículos novos, o valor cobrado é proporcional à quantidade de meses transcorridos do mês da compra (licenciamento do veículo), inclusive, até o final do ano. O valor integral (anual) do imposto é calculado como 4% do valor do veículo registrado na nota de compra/venda. Conhecendo-se o valor registrado na nota e o número correspondente ao mês de licenciamento de um veículo novo, como obter o valor do respectivo IPVA?
>
> Experimente a execução do aplicativo **ipva_veicnovo** que se encontra no disco (CD) fornecido.
>
> - Qual seria o valor integral do IPVA para um veículo cujo valor da nota de compra/venda seja de R$30000,00?
> - Descreva uma expressão matemática que relacione o valor da nota de compra/venda com o valor integral do IPVA.
> - Que fração do valor integral do IPVA deve corresponder ao valor a ser recolhido se o licenciamento do veículo novo for efetivado no mês de julho (mês 7)? E se o licenciamento ocorrer em dezembro?
> - Descreva uma expressão matemática que relacione o valor integral do IPVA e o número do mês do licenciamento com o valor do IPVA a ser recolhido. **IPVA recolhido= ???**
> - Faça a descrição de um algoritmo que represente um método de resolução do problema proposto (articule de forma adequada as relações matemáticas obtidas nas questões anteriores).Escolha identificadores de variáveis que indiquem os respectivos significados.
> - Faça a implementação e testes do programa correspondente ao algoritmo que você construiu.
> - Observe a seqüência de linhas que constituem o seu programa e procure responder: a única ordem de disposição de linhas correta é essa que figura

no seu programa ou é possível alguma inversão? Se for possível alguma inversão, indique algumas possibilidades.

4.2.8 Análise *a priori* – terceira parte da atividade

Nessa terceira parte da segunda atividade, o grupo deverá construir o algoritmo e o respectivo programa, que representem um método de resolução do problema proposto. O aplicativo fornecido deve se constituir em apoio para o entendimento da proposta do problema e deve produzir as referências para o aluno verificar a validade e adequação de seu algoritmo e de seu programa.

Na perspectiva da dialética ferramenta-objeto (Maranhão, 2008), a elaboração nessa terceira parte da atividade, envolve: acessar o *antigo*, que pode ser apontado como a rede constituída de elementos da linguagem algorítmica, elementos da linguagem de programação e elementos da matemática (contagem por diferença, relação de proporcionalidade e aplicação de cálculo de porcentagem); as fases *pesquisa* e *explicitação* são dirigidas e se desenvolvem pelo trabalho com a série de questões propostas (o processo de resolução começa a tomar forma e organização); o *novo implícito* é o par algoritmo-programa que deve ser construído; a *institucionalização* apenas se inicia no interior de cada grupo, com a validação da construção obtida, em função de testes do programa implementado e interpretação do processo e dos resultados obtidos.

Com a visão da teoria dos registros de representação semiótica (Duval, 2008), a tarefa se desenrola com atividades de conversões e tratamentos: o registro da proposta do problema (língua natural), os registros numéricos e algébricos, o registro do algoritmo e o registro do programa.

As primeiras perguntas devem favorecer o planejamento das instruções de atribuição necessárias para o cálculo do valor do IPVA. O grupo deverá discutir e, ao final, explicitar as relações funcionais entre as informações envolvidas: o valor integral do IPVA com o valor do veículo e depois a proporcionalidade entre a quantidade de meses, calculada em função do número que representa o mês de licenciamento, e o valor do IPVA a ser pago.

O método de resolução poderá ser elaborado com a base de uma única instrução de atribuição, mas espera-se que a construção seja descrita em duas ou três etapas intermediárias e mais simples.

A questão colocada ao final deve levar a uma revisão e interpretação do processo de resolução construído. O esperado é que, com essa revisão, o grupo possa se aprofundar na compreensão e justificativas do processo elaborado, além disso, outra finalidade é levar o grupo a discutir a respeito da importância da ordem de disposição e, como efeito, a ordem de execução das instruções que constituem o processo representado pelo algoritmo ou programa. Nesse algoritmo devem ser empregadas variáveis inteiras (mês ou quantidade de meses) e variáveis não inteiras (valor do veículo e valor do IPVA).

4.2.9 Observações e análise *a posteriori* – terceira parte da atividade

Na terceira parte da atividade foi proposto um problema computacional (o problema do cálculo do IPVA) e fornecido um pequeno aplicativo que opera o cálculo solicitado no problema, mas não contêm a representação do algoritmo e nem a representação do mecanismo de operação, o aplicativo poderia ser empregado para permitir o confronto de resultados tanto na fase de elaboração do algoritmo como na fase de testes do programa produzido ao final.

Nas duas experimentações, a primeira questão foi respondida pelos grupos a partir da execução do aplicativo, mas houve algum processo de reflexão e discussão, com a finalidade de entender ou validar aquele resultado obtido pelo aplicativo, um dos componentes do segundo grupo, da primeira experimentação, declarou: "*se o carro é de janeiro, paga o valor todo do imposto*"; na segunda experimentação foi percebida outra declaração: "... no fim do ano vai pagar bem menos...".

Na primeira experimentação, o segundo grupo respondeu à segunda questão já com a indicação do processo de cálculo que a proposta do problema solicita, e não apenas à pergunta dessa segunda questão, a expressão de cálculo registrada como resposta ficou assim: *IPVA recolhido=valor*0,04*12/12*. O primeiro grupo respondeu assim: *IPVA integral=valornota*0,04*.

Na segunda experimentação, todos os grupos produziram respostas adequadas a essas primeiras questões. Um dos grupos registrou como resposta à primeira questão: "*A cada mês há um desconto de R$100,00, exceto se o mês for janeiro, no qual o valor pago é integral.*", e não explicitou o valor R$1200,00.

Para a produção dessas respostas foi possível perceber que os grupos, nas duas experimentações, retornaram algumas vezes ao enunciado do problema e também à execução do aplicativo.

A terceira questão, que solicitava a fração correspondente aos meses de julho e dezembro, foi respondida corretamente por todos os grupos. Foi possível perceber que alguns grupos encaminharam algumas discussões no momento de definir essa resposta. Uma afirmação, relacionada à proporcionalidade, foi: "*vai caindo, de um em um, do começo até o fim do ano*".

Na primeira experimentação, a quarta questão dessa seqüência também foi respondida de forma satisfatória pelos dois grupos; a resposta do primeiro grupo indica a necessidade de cálculo prévio do IPVA integral:

IPVA recolhido=(12-mês+1)/12 . IPVA integral.

A resposta do segundo grupo inclui na própria expressão o cálculo do IPVA integral, assim: *IPVA recolhido= valor*0.04*(13-mês)/12*. Na elaboração dessas respostas os grupos utilizaram um tempo maior, mas aparentemente a idéia de proporcionalidade dos valores já estava estabelecida, a dificuldade maior foi com a elaboração da expressão algébrica para representar essas operações de proporcionalidade.

Na segunda experimentação, um dos grupos colocou como resposta a expressão correspondente ao caso particular tratado na questão inicial (valor do veículo igual a R$30000,00), essa falha permaneceu nas próximas construções. Outro grupo confundiu o valor do IPVA com a fração correspondente e respondeu: IPVArecolhido=(13-mês)/12, mas na construção do algoritmo a expressão correspondente foi registrada corretamente. Outro grupo ainda, respondeu: IPVA recolhido = IPVA integral/12, tal grupo introduziu a idéia de valor mensal do IPVA, e empregou corretamente essa idéia na construção do algoritmo. A Figura 6 é a imagem do registro da resposta do grupo.

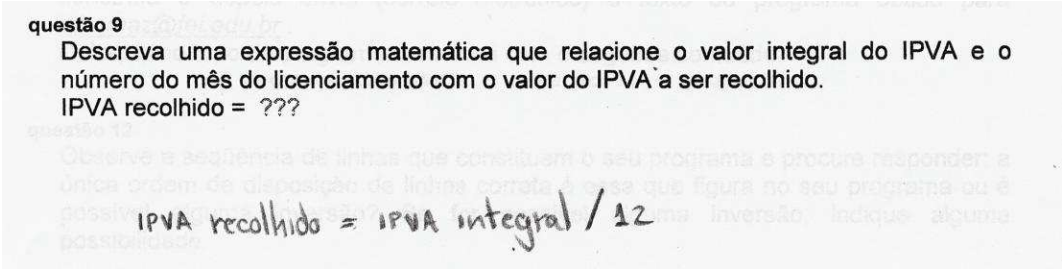


Figura 6 – Protocolo – questão 9 – segunda experimentação

Essas indicações são correspondentes a alguns desvios que foram introduzidos na tarefa de leitura das propostas ou na produção das expressões algébricas, assim: os desvios são resultantes das atividades de conversão ou tratamento dos registros.

Pode-se afirmar que os grupos passaram pela mobilização do *antigo*, com a recuperação de alguns elementos da matemática (proporcionalidade, porcentagem, contagem) e pelas fases *pesquisa* e *novo implícito*, com a elaboração das expressões de cálculo, indicativas da organização do método de resolução.

Nas últimas questões foram solicitadas as construções do algoritmo e do programa, e como era esperado, a partir das tarefas anteriores, os novos diálogos e discussões foram relativos à forma de organização do processo já delineado nas questões anteriores.

Nas duas experimentações, quase todos os grupos construíram corretamente o algoritmo e também o programa. No momento de realização de testes do programa, os grupos utilizaram o aplicativo como referência para o confronto de resultados obtidos. Apenas um dos grupos, na segunda experimentação, levou a falha já apontada (tratou o caso particular) para o algoritmo e depois para o programa.

Na primeira experimentação, a última questão foi respondida de formas diferentes pelos dois grupos: o segundo grupo considerou que não havia qualquer possibilidade de inversão na disposição das instruções do programa e o primeiro grupo percebeu que uma possibilidade de inversão era a mudança de ordem nas operações de entrada, e também entre as operações de saída, visto que o algoritmo e o programa desse primeiro grupo produziam duas saídas: o valor integral do IPVA e o valor do IPVA a ser recolhido.

Na segunda experimentação, alguns grupos responderam que não era possível qualquer inversão, e outros responderam que era possível inverter-se a ordem das entradas (valor do veículo e mês da compra) e a ordem das saídas (valor integral e valor a ser recolhido). Nessa questão, um dos grupos apontou a possibilidade de eliminar-se a atribuição correspondente ao cálculo do valor integral do IPVA, com sua incorporação ao cálculo do IPVA a ser recolhido. A conclusão é que os grupos realizaram alguma atividade de análise sobre o processo estabelecido com a construção do programa.

4.3 Análises – terceira atividade

4.3.1 Apresentação da introdução da atividade

A terceira atividade iniciou-se com uma introdução, em que foi proposta a leitura de um texto com a apresentação dos recursos relativos às estruturas de controle de seleção de um ramo e de dois ramos.

O texto é o seguinte:

> **Instruções de controle de seleção**
>
> Nas duas primeiras atividades todos os algoritmos apresentam fluxos de processamento seqüenciais simples, ou seja: cada instrução é executada exatamente uma vez, desde a primeira até a última, na ordem em que aparecem no texto do algoritmo. Nesta terceira atividade são introduzidas as instruções de controle de seleção de um ramo e de dois ramos; com o emprego dessas instruções torna-se possível a construção de algoritmos com fluxos de processamento alternativos, assim: uma instrução de controle de seleção corresponde a uma "escolha" pelo sistema entre duas ações possíveis, com isso uma parte das instruções que figuram no algoritmo ou programa poderá não ser executada.
>
> A instrução de controle de seleção de um ramo possui a seguinte sintaxe geral:

se ***<expressão de controle>*** **então** ***<bloco de instruções>*** ;

<próxima instrução>;

A construção correspondente em C/C++ é:

```
if( <expressão de controle> ){
      bloco de instruções ;
}
<próxima instrução>;
```

A ***<expressão de controle>*** é uma expressão lógica (booleana) que é avaliada pelo sistema no momento em que o fluxo de processamento atinge a instrução ***se ... então ...*** , o valor da expressão é um valor lógico - verdadeiro ou falso - e tal valor é o elemento que define o caminho do fluxo de processamento, assim: se a ***<expressão de controle>*** for avaliada como verdadeira o sistema realiza a execução do ***<bloco de instruções>*** e depois segue para a ***<próxima instrução>***, por outro lado, se a ***<expressão de controle>*** for avaliada como falsa o sistema não executa o ***<bloco de instruções>*** e segue diretamente para a ***<próxima instrução>***; portanto, nessa construção, as alternativas são: o sistema realiza a execução do ***<bloco de instruções>*** ou não realiza essa execução, e depois segue para a ***<próxima instrução>***.

No texto do algoritmo o ***<bloco de instruções>*** é definido pela colocação de uma moldura retangular e no programa a delimitação é feita com um par de chaves { ... }. Não há qualquer restrição sobre a constituição do ***<bloco de instruções>*** , nem relativa à quantidade de instruções e nem sobre a natureza das mesmas; caso o ***<bloco de instruções>*** seja constituído por uma única instrução pode-se omitir sua delimitação, tanto no algoritmo como no programa.

A instrução de controle de seleção de dois ramos possui a seguinte sintaxe geral:

se ***<expressão de controle>*** **então** ***<bloco de instruções 1>*** ;
senão ***<bloco de instruções 2>*** ;

<próxima instrução>;

A construção correspondente em C/C++ é:

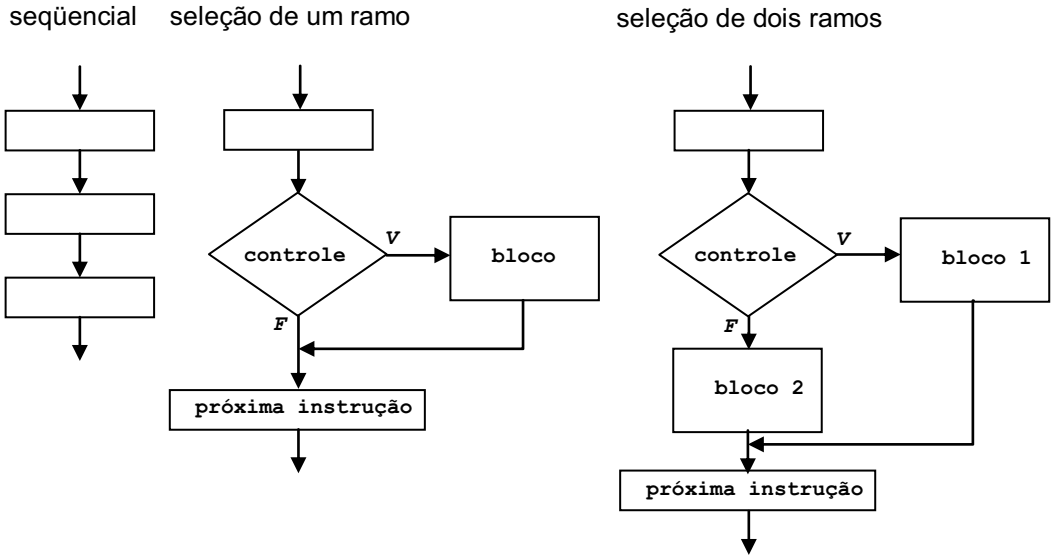
```
if( <expressão de controle> ){
      <bloco de instruções 1>;
}
else{
      <bloco de instruções 2>;
}
<próxima instrução>;
```

O mecanismo de comportamento é semelhante ao que ocorre com a instrução de controle de seleção de um ramo: o sistema avalia a ***<expressão de controle>*** e se o valor obtido for verdadeiro o fluxo segue para o ***<bloco de instruções 1>*** e depois para a ***<próxima instrução>***, caso o valor seja

falso será executado o ***<bloco de instruções 2>*** e depois o fluxo de processamento segue para a ***<próxima instrução>***.

Neste caso, as alternativas são: ou realizar a execução do ***<bloco de instruções 1>*** ou realizar a execução do ***<bloco de instruções 2>*** e depois executar a ***<próxima instrução>***.

As figuras abaixo representam as possibilidades de fluxo de processamento em três situações: fluxo seqüencial simples, fluxos alternativos com controle de seleção de um ramo e fluxos alternativos com controle de seleção de dois ramos.



A constituição da ***<expressão de controle>*** pode envolver os operadores de relação de ordem e também os operadores lógicos; observe a seguir um exemplo de construção com o emprego desses operadores.

Exemplo:

```
VerificaData( )
leia(Dia); leia(Mes); leia(Ano);
se Mes>0 e Mes<13 e Ano>1900
então DiasNoMes←31;
      se Mes=2 então Bst←Ano mod 4=0 e Ano mod 100≠0
                         ou Ano mod 400=0;
                     se Bst então DiasNoMes←29;
                            senão DiasNoMes←28;
      se Mes=4 ou Mes=6 ou Mes=9 ou Mes=11 então DiasNoMes←30;
      se Dia≥1 e Dia≤DiasNoMes então imprima("data válida");
                                senão imprima("data inválida");
senão imprima("data inválida");
```

código correspondente:

```
int main( ){
   int Dia, Mes, Ano, DiasNoMes;
   bool Bst;
   cout<<"digite a data:"<<endl;
   cout<<"dia? "; cin>>Dia;
   cout<<"mês? "; cin>>Mes;
   cout<<"ano? "; cin>>Ano;
   if(Mes>0 && Mes<13 && Ano>1900){
     DiasNoMes=31;
     if(Mes==2){
       Bst= Ano%4==0 && Ano%100!=0 || Ano%400==0;
       if(Bst) DiasNoMes=29;
       else    DiasNoMes=28;
     }
     if( Mes==4 || Mes==6 || Mes==9 || Mes==11 )
        DiasNoMes=30;
     if(Dia>=1 && Dia<=DiasNoMes)
        cout<<"data válida "<<endl;
     else
        cout<<"data inválida "<<endl;
   }
   else
     cout<<"data inválida "<<endl;
   system("PAUSE");
   return(0);
}
```

Os operadores de relação de ordem são os usuais, representados na linguagem algorítmica por: **<** , **≤**, **=**, **≥**, **>**, **≠** e na linguagem C/C++ por **<**, **<=**, **==**, **>=**, **>**, **!=** . Os operadores lógicos **e** , **ou** e a negação **não** são indicados na linguagem C/C++ por **&&** , **||** e **!** . Em C/C++ os operadores de relação de ordem têm prioridade sobre os operadores lógicos e entre os operadores lógicos a conjunção && têm prioridade sobre a disjunção **||** .

Questões:

- Qual a finalidade das estruturas de controle de seleção em um algoritmo ou programa?

- Observe a seqüência de instruções abaixo:

```
...
1. float x;
2. cout<<"valor de x? ";
3. cin>>x;
4. x=3*x;
5. if(x<7.2){
6.    x=x+10;
7.    x=1.5*x;
8. }
9. else{
10.         x=2.5*x;
11.         x=x+10;
12.      }
13.      cout<<"novo x: "<<x<<endl;
...
```

- Quais linhas de instruções serão executadas se o valor fornecido como conteúdo inicial da variável **x** for 4.4? Nesse caso, qual será o valor

armazenado como conteúdo dessa variável ao final da execução da seqüência de instruções?

- E se o conteúdo inicial de **x** for 1.8?
- Quais das seqüências abaixo são equivalentes àquela descrita acima?

seqüência 1

```
...
  float x;
  cout<<"valor de x? ";
  cin>>x;
  x=3*x;
  if(x>=7.2){
     x=2.5*x;
     x=x+10;
  }
  else{
     x=x+10;
     x=1.5*x;
  }
  cout<<"novo x: "<<x<<endl;
...
```

seqüência 2

```
...
  float x;
  cout<<"valor de x? ";
  cin>>x;
  x=3*x;
  if(x>=7.2){
     x=2.5*x;
     x=x+10;
  }
  if(x<7.2){
     x=x+10;
     x=1.5*x;
  }
  cout<<"novo x: "<<x<<endl;
...
```

seqüência 3

```
...
  float x;
  cout<<"valor de x? ";
  cin>>x;
  x=3*x;
  if(x<7.2){
     x=x+10;
     x=1.5*x;
  }
  x=2.5*x;
  x=x+10;
  cout<<"novo x: "<<x<<endl;
...
```

4.3.2 Análise *a priori* – primeira parte da atividade

A introdução à terceira atividade é a proposta de um texto para leitura com a apresentação das estruturas de controle de seleção (controles de seleção de um ramo e de seleção de dois ramos), o tipo booleano, os operadores de relação de ordem e os operadores lógicos, e algumas questões para reflexão.

O texto apresenta uma descrição da sintaxe e do mecanismo de funcionamento das estruturas de controle de seleção, e um exemplo de algoritmo com o emprego de algumas dessas estruturas. O objetivo é que o aluno retome a finalidade e o entendimento da formalização e da semântica dessas instruções de controle.

O texto introduz três registros de representação para cada uma das duas estruturas de controle: linguagem algorítmica, linguagem de programação e fluxograma.

No sentido da dialética ferramenta-objeto, os elementos apresentados se configuram com parte do *antigo* nos processos de construção que os grupos devem realizar.

A primeira questão colocada deve levar o grupo a uma breve discussão e análise das informações contidas no texto, e a uma síntese sobre o conjunto desses recursos (controles de seleção).

Com as outras duas questões propostas o esperado é que o grupo consiga especificar, nesses casos particulares, como é o funcionamento do controle de seleção e declare que apenas um dos blocos de instruções será executado, ou o do ramo **if** ou o do **else**. A constituição simples dessas linhas de programa tem o propósito de concentrar o foco da atenção no mecanismo do controle de seleção. Essa questão coloca em foco o aspecto dinâmico que o controle **if ... else ...** representa.

A última questão proposta nessa primeira parte deve levar a uma análise das constituições das instruções de controle e correspondentes comportamentos: o grupo deve perceber que um mesmo efeito pode ser obtido a partir de diferentes construções, e que, a instrução **if ... else ...** pode ser reproduzida por duas instruções **if ...** em seqüência.

Para a segunda experimentação, a proposta da terceira questão foi reformulada, pois se observou, com a primeira experimentação, que o texto inicial acarretou dificuldades na interpretação. A intenção foi manter o mesmo propósito, mas com uma forma mais direta da pergunta. Com a reformulação a proposta é a seguinte:

Observe a seqüência de instruções abaixo e considere a afirmação:

para mesmos valores iniciais de x ela produzirá as mesmas "saídas" que foram resultados da execução da seqüência acima.

A afirmação é verdadeira ou falsa? Justifique.

```
...
 float x;
 cout<<"valor de x? ";
 cin>>x;
 x=3*x;
 if(x>=7.2){
    x=2.5*x;
    x=x+10;
 }
 if(x<7.2){
    x=x+10;
    x=1.5*x;
 }
 cout<<"novo x: "<<x<<endl;
...
```

4.3.3 Observações e análise *a posteriori* – primeira parte da atividade

Na primeira experimentação, a terceira atividade foi realizada por um único grupo de três alunos. Durante a semana que antecedeu esse encontro, as aulas normais foram suspensas para dar lugar a atividades diferenciadas como palestras, mesas de discussão e apresentações de trabalhos (Semana Acadêmica), em função disso, apenas três alunos compareceram para a realização desta atividade, esses alunos já haviam participado das atividades anteriores.

Na segunda experimentação houve a participação de nove grupos, um deles com quatro alunos e os outros com três alunos cada um.

Na primeira experimentação, a leitura inicial foi concluída sem muitos diálogos ou discussões, mas para a elaboração das respostas iniciais, o grupo retomou o texto algumas vezes. No algoritmo colocado como exemplo, os alunos reconheceram o controle de seleção correspondente à caracterização de ano bissexto, em uma das aulas eles já haviam trabalhado com esse critério de verificação para a resolução de um problema de contagem de dias.

A produção da resposta à primeira questão exigiu algumas discussões, e apesar da redação da resposta ter sido registrada de forma incompleta –*Análise*

das informações, para a validação de uma condição. – foi possível perceber, entre os diálogos, uma afirmação que revela o entendimento mais completo sobre a finalidade das estruturas de controle de seleção. Pouco antes de chegarem a um consenso sobre a resposta que seria escrita, um dos alunos, com a concordância dos outros dois, afirmou: "*ele cata os dados, analisa a condição e fala pra onde você vai ...* ". Ao mesmo tempo em que fazia a afirmação, o aluno gesticulava com movimentos de sua mão direita para reforçar a idéia de caminhos alternativos à esquerda ou à direita.

As respostas elaboradas para as próximas duas perguntas, que exigiam a simulação de uma seqüência de instruções, foram formuladas corretamente. Para a elaboração dessas respostas, apesar dos cálculos serem bastante simples, o grupo utilizou a calculadora. A respeito da instrução de controle *if* , disposta na linha 5 da seqüência, é interessante observar que não foi citada como uma das linhas executadas na segunda questão e que foi citada na terceira questão, mas como parte integrante do bloco de instruções vinculado ao ramo do *if* ; o esperado era que as faixas de linhas executadas fossem expressas assim:

segunda questão: 1 a 5 e 9 a 13

terceira questão: 1 a 5, 6 a 7 e 13 ou 1 a 7 e 13

e no entanto as respostas foram:

segunda questão: 1 a 4 e 9 a 13

terceira questão: 1 a 4, 5 a 7 e 13;

apesar do mecanismo de controle ter sido simulado corretamente, a execução da instrução *if* correspondente não foi indicada como o esperado, talvez isso se explique pelo fato de que a execução da instrução *if* não produza modificações nos conteúdos de variáveis, a ação correspondente é definir o caminho do fluxo de processamento.

A última questão dessa primeira série não foi interpretada nem respondida como era o esperado, o grupo entendeu a equivalência entre as seqüências de instruções como ações idênticas a cada passo do processo e não como ações diferentes, mas que produzem os mesmos efeitos; a forma como foi proposta a questão "quais das seqüências abaixo são equivalentes àquela descrita acima?" deve ser revista e modificada, talvez uma redação mais adequada seja: "quais das seqüências abaixo produzem resultados iguais aos produzidos pela execução

da seqüência descrita acima?". Durante os diálogos para a discussão dessa questão foi possível notar que os alunos perceberam corretamente a não equivalência da seqüência 3, mas o argumento para a não equivalência das duas primeiras seqüências foi constituído apenas a partir da constatação da inversão do confronto disposto como expressão de controle na instrução *if*. Na seqüência original o confronto é *x<7.2* e nas seqüências 1 e 2 é *x>=7.2*, essa foi essencialmente a justificativa para as não equivalências.

Na segunda experimentação, um dos grupos não registrou a primeira resposta, mas foi possível localizar nos rascunhos a indicação de que a questão foi tratada: "ela *serve para dar caminhos alternativos*". Dois grupos registraram valores incorretos da variável **x**, na segunda ou na terceira questões, apesar de indicarem corretamente as linhas que seriam executadas.

As faixas de linha que seriam executadas foram indicadas de várias maneiras diferentes, mas com o mesmo sentido: alguns grupos apontaram apenas as duas linhas do ramo *if* ou do ramo *else*. Outros apontaram as faixas completas ou quase completas.

A análise sobre a equivalência de construções, solicitada na quarta questão, foi elaborada de maneira adequada pela maioria dos grupos. Apenas um grupo respondeu que as seqüências não eram equivalentes, com uma justificativa que não foi permitiu interpretação. Algumas das justificativas registradas são incompletas, mas permitem concluir que a análise foi trabalhada e que o grupo percebeu a equivalência.

Em função das respostas registradas, pode-se avaliar que a maioria dos grupos conseguiu o objetivo central da primeira parte da atividade que era retomar os conceitos envolvidos nas estruturas de controle de seleção.

4.3.4 Apresentação da segunda parte da atividade

Na segunda parte da terceira atividade é colocado o problema das medidas dos ângulos internos de um triângulo. A proposta do problema é acompanhada de

um aplicativo, representante de um algoritmo, que é um método de resolução, e que ilustra o seu mecanismo de execução. Assim:

> Considere a proposta do problema:
>
> Conhecendo-se as medidas de dois dos três ângulos internos de um triângulo (medidas dadas por valores inteiros em graus), como verificar se tais medidas são compatíveis (se podem ser, de fato, medidas de dois ângulos internos de um triângulo) e, em caso afirmativo, como determinar a medida do terceiro ângulo interno do triângulo?
>
> No disco (CD) há um pequeno aplicativo (**verifica_triangulo**) que ilustra o método de resolução desse problema, com descrição em linguagem algorítmica, o aplicativo ilustra também a execução do algoritmo.
> .
>
> Questões:
>
> - Se as duas medidas conhecidas forem 30^o e 50^o, qual será a medida do terceiro ângulo?
> - Descreva quatro casos (quatro duplas de valores inteiros) para os quais a resposta produzida pelo algoritmo seja 70^o.
> - Descreva um caso em que as medidas sejam incompatíveis com a existência do triângulo.
> - Como você descreve a finalidade da instrução colocada na linha 4 do algoritmo?

4.3.5 Análise *a priori* - segunda parte da atividade

O trabalho com a segunda parte dessa terceira atividade deve contribuir para o entendimento das estruturas de controle de seleção em seus vários aspectos: finalidades, sintaxe, semântica e dinâmica do fluxo de processamento correspondente. Os alunos devem produzir, de forma consistente, bases de informações e conhecimentos, para que depois possam articular tais elementos diante de propostas de outros problemas, com o propósito de produzirem os métodos de resolução adequados a essas outras situações.

Essa fase inicial da terceira atividade retoma noções já apresentadas nas aulas normais do curso e pretende que o aluno possa, em função das discussões e análises em torno das leituras e tarefas propostas, ampliar sua autonomia diante de tais conceitos e recursos para o enfrentamento de novas situações ou problemas propostos.

Para produzir as respostas das três primeiras questões colocadas espera-se que o grupo manipule a execução do aplicativo com a animação do algoritmo. A partir da observação dessas execuções, os alunos devem perceber a atuação dos controles de seleção para a definição dos caminhos de fluxo de processamento.

O alvo principal é exatamente esse entendimento: a constituição e representação do registro, em linguagem algorítmica, de uma estrutura de controle de seleção, e a associação, a esse registro, do mecanismo de fluxo de processamento alternativo representado.

A instrução da linha 4 deve ser percebida como a base para o processo de seleção, e entendida como aquela instrução que define as alternativas de fluxos de processamento, correspondentes à existência ou não existência do triângulo. Nesse cenário, o estudante deve perceber e entender a estrutura de controle na sua constituição completa: a expressão lógica que é a chave do controle, a instrução que define os fluxos alternativos e as coleções de instruções que configuram os ramos dos caminhos alternativos. É essa noção completa, representada pelo registro da estrutura, que o aluno deve dominar.

4.3.6 Observações e análise *a posteriori* - segunda parte da atividade

Na segunda parte dessa atividade, os alunos encontraram a proposta de um problema computacional cuja resolução envolve a relação entre as medidas dos três ângulos internos de um triângulo.

Na primeira experimentação, um dos alunos logo lembrou a relação que deveria ser o elemento central do método de resolução: "*a soma de todos os ângulos deve ser 180 ...*" (omitiu a unidade de medidas), mas o entendimento efetivo da proposta do problema não foi imediato e a observação do aplicativo com o método de resolução animado contribuiu para tal entendimento. Quando os alunos olharam o algoritmo fornecido, perceberam que a possibilidade de existência do triângulo seria verificada a partir da validade da expressão *alfa>0 e beta>0 e soma<180*, que foi disposta como expressão de controle da estrutura de

seleção `se ... então ...`. Uma das declarações dos alunos foi "*não pode passar de 180 senão não dá ...*" (omitiu a unidade de medidas).

As três primeiras questões propostas na seqüência foram respondidas corretamente e o grupo colocou em execução o aplicativo para confirmar as respostas. Em um dos diálogos, depois de terem chegado a consenso sobre uma parte das respostas, um dos alunos falou "*testa aí no programa pra gente ver*.".

A segunda questão dessa série não foi compreendida completamente, já que pedia quatro duplas de valores e o grupo registrou como resposta apenas duas duplas. A terceira questão, que solicitava um caso em que as medidas fossem incompatíveis, foi respondida com dois pares de valores e a intenção do grupo parece ter sido evidenciar duas classes de casos de incompatibilidade, uma com um dos valores igual a zero, e outra com a soma dos valores maior do que 180^0.

A resposta à quarta questão foi registrada assim: "*tem como finalidade comparar os valores se são compatíveis para formar um triângulo*", apesar de não ter sido registrada explicitamente a idéia de constituição de fluxos alternativos, foi possível perceber que os alunos entenderam essa noção: um dos ramos indica a existência do triângulo e o outro a incompatibilidade dos valores. O conjunto de respostas e os diálogos observados permitem concluir que o registro, em linguagem algorítmica, foi entendido na sua totalidade.

Durante a elaboração do registro escrito das respostas, o grupo ainda executou o aplicativo com a representação do algoritmo pelo menos outras duas vezes, para com isso confirmar o entendimento da situação do problema e do mecanismo dinâmico representado pelo programa.

Na segunda experimentação, foram observadas e anotadas algumas declarações, que ocorreram nessa fase de trabalho da terceira atividade, e que remetem à noção de processo dinâmico relacionado ao registro, em linguagem algorítmica, do controle de seleção: "*precisa calcular e olhar primeiro, para depois escolher por onde vai*", "*aqui ele vê qual o caminho que segue, vai por este* (aponta uma região) *ou por este* (aponta outra região na tela), *só um dos dois*", "*o caminho* (selecionado) *depende dos cálculos antes*", "*ele vai por cima ou vai por*

baixo", "*por aqui* (aponta uma região) *sempre passa*" e "*segue por este atalho ou pelo outro*".

Além das declarações, foi possível perceber alguns gestos típicos da indicação de alternativas: movimento com uma das mãos e braço oscilando entre dois pontos, movimento circular (meia volta) de uma das mãos com os dedos indicador e médio em "V", oscilação da cabeça e dos olhos para cima e para baixo, e movimento da caneta com a indicação de dois percursos.

As declarações e os gestos indicam a vinculação da dinâmica do fluxo de processamento ao registro do algoritmo. Um dos componentes que contribuiu para estabelecer essa vinculação foi o aplicativo com a ilustração do algoritmo.

Na segunda experimentação, as respostas escritas dessa segunda parte da atividade foram todas adequadas. Como resposta à questão "*Como você descreve a finalidade da instrução ...*", oito dos nove grupos registraram a finalidade específica da instrução, com frases semelhantes à seguinte: *a finalidade é verificar se os valores podem ser medidas de dois ângulos internos de um triângulo*, e um dos grupos demonstrou o sentido mais completo, ao considerar toda a estrutura de seleção e não apenas a instrução de controle disposta na linha 4 do algoritmo, a resposta registrada foi: "*se o ângulo alfa for maior que zero e o ângulo beta for maior que zero e a soma de alfa mais beta for menor que 180º, ele atribuirá o valor de gama e imprimirá 'compatível'. Senão imprimirá 'incompatível'*".

Foi possível notar que o aplicativo, com a animação do algoritmo, foi acionado por várias vezes durante as discussões e elaboração das respostas, em todos os grupos. Em relação a esse aspecto, a conclusão é que o aplicativo cumpriu sua finalidade de contribuir como componente dos entendimentos, discussões e elaboração das respostas.

4.3.7 Apresentação da terceira parte da atividade

A terceira parte dessa atividade propõe o trabalho com o problema do prêmio do motorista. A redação da proposta está transcrita a seguir.

Proposta do problema:

Uma empresa de transportes rodoviários registra, para cada um de seus motoristas, o total mensal de quilômetros percorridos e realiza pagamentos de prêmios da seguinte maneira: para quilometragens percorridas de até 12000 km o prêmio é de 5,5% do salário bruto do motorista e para quilometragens superiores a 12000 km, além dos 5,5%, o motorista recebe mais 2,8% do salário bruto a cada 600km que exceder os 12000 km; nessa última situação uma eventual parcela menor do que 600 km também recebe o adicional de prêmio.

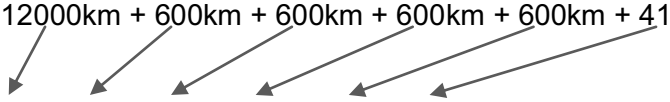
Conhecendo-se o valor do salário bruto de um motorista e o seu total mensal de quilômetros percorridos, como calcular o valor do prêmio correspondente? Observação: a quilometragem percorrida deverá ser representada por uma variável de tipo inteiro.

Veja os exemplos de cálculo:
exemplo 1: salário bruto R$1270,00 e quilometragem percorrida de 9863km

quilometragem percorrida (9863km) menor do que 12000km
→ prêmio: 5,5% de R$1270,00 → prêmio=R$69,85.

exemplo 2: salário bruto R$1270,00 e quilometragem percorrida de 14812km

14812km = 12000km + 600km + 600km + 600km + 600km + 412km



→ prêmio: 5,5%+2,8%+2,8%+2,8%+2,8%+2,8% de R$1270,00
→ prêmio: 19,5% de R$1270 → prêmio=R$247,65.

Experimente a execução do aplicativo **premio_motorista** que se encontra no disco (CD) fornecido.

Questões

- Para um percurso total de 13050 km a porcentagem que definirá o prêmio será 11,1%. Explique o processo aplicado para o cálculo de tal valor.
- Descreva uma expressão matemática que indique a forma de cálculo do valor do prêmio a partir da taxa (%) e do valor do salário bruto. Suponha que a taxa já tenha sido calculada.
- Organize um conjunto de expressões matemáticas para traduzir a relação entre a quilometragem total e a taxa de porcentagem que deve ser aplicada para o cálculo do valor do prêmio.
- Faça a descrição de um algoritmo que represente um método de resolução do problema proposto (articule de forma adequada as relações matemáticas e lógicas obtidas nas questões anteriores). Utilize identificadores de variáveis que indiquem o significado, diante da proposta do problema, de cada informação representada.
- Faça a implementação e testes do programa correspondente ao algoritmo que você construiu. Transcreva na área abaixo o texto de seu programa.

4.3.8 Análise *a priori* – terceira parte da atividade

A leitura da proposta do problema em conjunto com os exemplos descritos e também a possibilidade de experimentar o aplicativo, que realiza o o processo de cálculo, devem levar ao entendimento da situação do problema.

A resposta à primeira questão deve levar a uma nova leitura da proposta do problema e dos exemplos de cálculos e, ao final, poderá servir como mais um caso de teste do programa construído.

Para obter o valor da taxa do prêmio (primeira questão), espera-se que os alunos façam uma adaptação sobre os exemplos fornecidos, e reproduzam uma partição, agora aplicada ao valor 13050 km, semelhante ao que foi colocado no segundo exemplo.

Espera-se também que o grupo perceba, para responder à segunda questão, que esse processo de partição será traduzido por operações aritméticas: subtração, cálculo de quociente e resto de uma divisão (cálculos aritméticos com valores inteiros).

Nessa fase da atividade, o grupo deve, na visão da dialética ferramenta-objeto, organizar o *antigo* e as *pesquisas*, com isso, o processo de resolução começa a ser delineado e se dá o início da constituição do *novo implícito*.

Para obter a resposta à segunda questão, o grupo deverá traduzir o valor da taxa (porcentagem) em fator multiplicativo correspondente, a ser aplicado ao salário bruto. O trabalho com a conversão de registros (texto da proposta do problema e registro algébrico) ocorre.

Para produzir a resposta à terceira questão, o grupo deve perceber que o processo deve ser desdobrado em duas formas: uma para quilometragens até 12000 e outra para valores superiores. A organização do processo deve envolver, além das expressões aritméticas, um mecanismo de seleção entre as duas alternativas. O *novo implícito* deve ser delineado e o trabalho com o registro do algoritmo (linguagem algorítmica) deve ser iniciado. Além dos elementos lógicos e algébricos, a concepção do processo de resolução já deve antecipar a visão das estruturas de controle que irão compor o registro do algoritmo.

Para a elaboração e descrição do algoritmo, a expectativa é que o grupo organize de maneira adequada o esboço do processo já delineado pelas respostas anteriores.

Uma vez que o grupo tenha obtido as expressões de cálculo solicitadas nas questões anteriores, a constituição do algoritmo, e depois a implementação do programa, exigem a construção de estruturas de controle de seleção que devem definir os caminhos de fluxo de execução dos processos de cálculo em correspondência aos critérios descritos na proposta do problema. Nessa fase completam-se as atividades de conversões e tratamento de registros de representação.

Espera-se que os alunos utilizem a execução do aplicativo como referência na realização dos testes do programa produzido, e discutam as respostas apresentadas pela execução do programa. Essa fase envolve o início da *institucionalização*, que não se completa durante a realização da atividade, e *validação* do *novo*, que se estabelece na construção do programa.

4.3.9 Observações e análise *a posteriori* – terceira parte da atividade

Na terceira parte dessa atividade foi proposto o problema de cálculo do prêmio do motorista. O próprio enunciado do problema já indicava a essência do mecanismo de cálculo que deveria ser construído, foram incluídos ainda dois exemplos de casos particulares com o intuito de facilitar o completo entendimento da situação problema.

Foi possível perceber, nas duas experimentações que os dois casos de exemplos foram ainda tomados como referências para os testes ao final da implementação do programa.

Nas duas experimentações, os alunos entenderam sem muita dificuldade o cenário do problema, e mesmo durante a leitura inicial já iniciaram as discussões com vistas ao processo de resolução que deveriam construir.

Na primeira experimentação um dos alunos declarou: “*acho que vai ter que fazer uma regra de três ...*”, mais tarde, quando discutiam sobre a elaboração da

resposta à primeira questão, esse encaminhamento foi rejeitado por um dos alunos com o argumento de que a variação da taxa do prêmio não ocorria exatamente como a variação da quilometragem, ou seja, tal aluno percebeu que a relação não era exatamente linear ou proporcional.

Ao responderem à primeira questão colocada, apesar da intenção não ter sido esta, o grupo produziu um resumo estruturado de todo o método de resolução, inclusive com as primeiras indicações das expressões algébricas que seriam necessárias.

Na segunda experimentação, um dos grupos também elaborou parcialmente um método de resolução, mas ainda incompleto. A maioria dos outros grupos indicou a mesma forma de partição da quilometragem que foi colocada no exemplo dado.

Na primeira experimentação, um dos alunos levantou uma dúvida em relação ao processo de cálculo, com a indagação "*da diferença ele vai sempre subtrair 600, ou dividir?*", não há evidência concreta, mas é quase certo que tal aluno percebeu que o processo poderia ser constituído a partir de subtrações sucessivas, mas preferiu a alternativa de trabalhar com a divisão, talvez para evitar a necessidade de constituir uma estrutura de controle de repetição e um processo de contagem já que tal recurso não estava mencionado nos textos dessa terceira atividade.

A resposta à primeira questão produzida pelo grupo (primeira experimentação) foi escrita assim: "*Se o valor percorrido for menor 12000 o motorista vai receber apenas 5,5%. Subtrai os 12000 dos 13050, essa subtração gerou um resto. Esse resto é dividido por 600 (DIV inteiro) e o resto da divisão soma-se 1 ao quociente * 2,8 + 5,5 .*"; esse registro não exprime exatamente o processo geral de solução, mas indica de forma compacta, e razoavelmente estruturada, o seu esboço e seu desdobramento em ações realizadas ao longo do tempo: o controle de seleção – "*se o valor percorrido for menor 12000 ...*" e as operações necessárias – "*subtrai os 12000 dos 13050 ... esse resto é dividido por 600 ... soma-se 1 ao quociente * 2,8+5,5*".

A descrição oral correspondente, produzida por um dos alunos do grupo, estava mais bem organizada do que o registro escrito, a seguir está colocada a

transcrição de algumas partes dessa descrição oral: "*para fazer esse cálculo, primeiro precisa saber se é maior que 12000, ... se você pega esse valor, subtrai por 12000 ... aqui ele já tem os 5% ... aqui a gente pode dividir por 600, vai dar um valor x ... depois a gente vai pegar o módulo ... o resto por 600 ... e tem que ver se o módulo é maior que zero, se for maior soma 1 ao x, senão vai ser só o valor do x, daí a gente pega esse valor do x e multiplica por 2,8, soma com 5,5...*"; isso evidencia a maior facilidade no emprego do registro de fala do que no registro escrito. O esperado com essa questão era que o grupo elaborasse a decomposição do valor 13050, assim: 13050=12000+600+450, em correspondência fizesse a vinculação das taxas a cada uma das três parcelas, e calculasse a soma: 5,5+2,8+2,8=11,1 para assim obter a taxa resultante.

Foi possível observar, nas duas experimentações, que os exemplos fornecidos junto ao enunciado do problema e a execução do aplicativo de cálculo foram retomados várias vezes em todas as fases dessa terceira parte da atividade. O aplicativo foi utilizado para dar suporte tanto no início, para a produção das primeiras respostas, quanto depois, quando já delineava o método de resolução do problema. Algumas vezes a execução do aplicativo tinha a finalidade de clarificar a situação do problema, e outras vezes para confirmar, pelo menos em parte, algumas conclusões estabelecidas pelo grupo.

Na primeira experimentação, a segunda questão foi respondida de forma incorreta: "*Premio=sb*(1-taxa)*", mas tal engano foi depois eliminado, quando da construção do algoritmo e do programa.

Na segunda experimentação, oito dos nove grupos responderam com expressões semelhantes a *premio=salario x taxa / 100* ou
premio = salario x taxa. Alguns grupos entenderam a taxa como valor de porcentagem e para outros a taxa já se constituía como fator. Um dos grupos registrou uma resposta incorreta, e outro grupo tentou descrever, na expressão solicitada, o cálculo do valor da taxa, e não apenas sua aplicação para o cálculo do valor do prêmio.

A resposta à terceira questão dessa série não foi registrada no caderno da atividade e ao final dos trabalhos, quando o grupo foi questionado sobre a falta dessa resposta, a justificativa colocada pelos alunos foi no sentido de que a resposta elaborada para a questão inicial já respondia também a terceira questão,

de fato foi possível perceber naquela resposta um esboço da relação entre a quilometragem total e a taxa relativa ao prêmio, principalmente se forem consideradas as falas dos alunos que antecederam a produção do texto colocado como resposta.

Na segunda experimentação, as respostas à próxima questão foram adequadas, nem todas estavam corretas ou completas. Na maioria delas os alunos recorreram a alguma combinação entre registros algébricos e registros da língua natural para organizarem as relações que deviam ser elaboradas.

As próximas questões solicitavam as construções do algoritmo e do programa representantes do método de resolução do problema.

Na primeira experimentação, a descrição do algoritmo apresentou apenas pequenas falhas de notação e o programa foi produzido corretamente. Foi possível perceber que em vários momentos o grupo recorreu às respostas anteriores e também à execução do aplicativo, ao construírem o algoritmo e depois o programa. A Figura 7 exibe o protocolo do grupo.

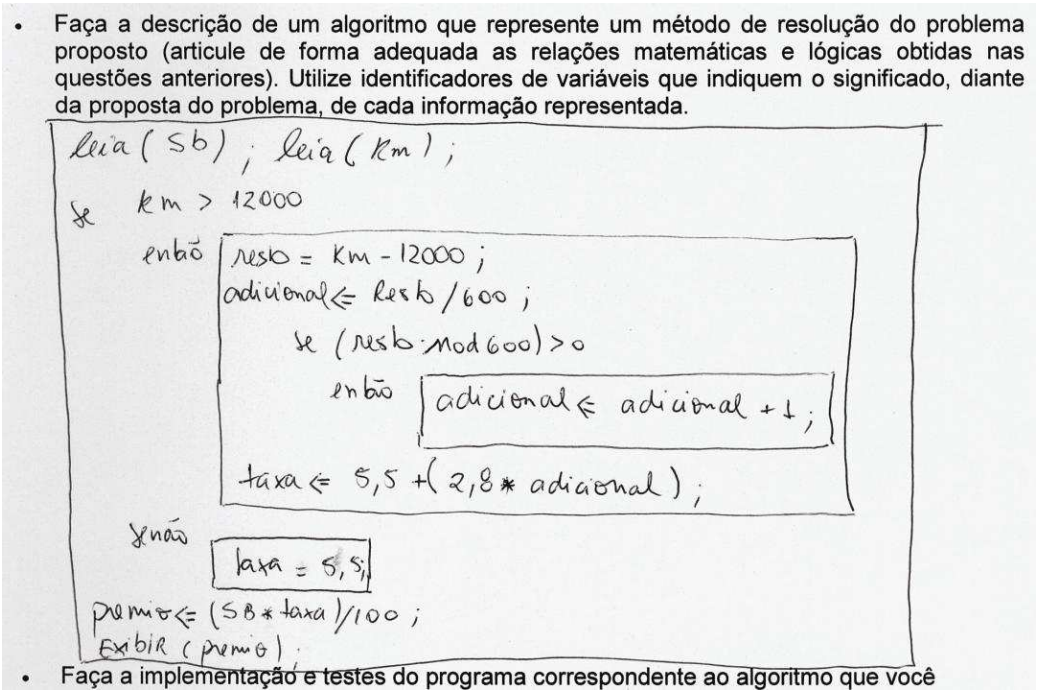


Figura 7 – Protocolo – algoritmo prêmio do motorista – primeira experimentação

Antes de iniciarem a elaboração da descrição do algoritmo, os alunos ainda discutiram sobre a necessidade e a forma da estrutura de controle de seleção para definir as duas alternativas de cálculo: percurso de até 12000km e percurso

acima dessa quantidade, mesmo já com as discussões iniciais que praticamente definiram todos os contornos da estratégia de resolução.

Na primeira experimentação, o desenvolvimento dos trabalhos ocorreu em três ciclos de discussões ou análises: no primeiro ciclo, rapidamente os alunos realizaram a leitura da proposta do problema, observaram a execução do aplicativo que calcula e exibe o valor do prêmio e a taxa correspondente; nesse primeiro ciclo o grupo já enunciava uma tentativa de esboço de método de resolução. No segundo ciclo, quando o grupo procurava responder às questões iniciais, a leitura do problema foi retomada e também a observação do aplicativo, nesse momento o esboço do método de resolução foi mais bem organizado e detalhado. No terceiro ciclo, quando os alunos deveriam produzir o texto do algoritmo e do programa, novamente o problema e o aplicativo foram revisitados, com o objetivo de completar a estratégia de resolução e verificar sua validade; nesse terceiro ciclo, em alguns momentos, transpareceu no comportamento dos alunos a necessidade de se utilizar a linguagem algorítmica formalizada para que o processo já planejado ficasse exatamente definido, ou seja, os estudantes perceberam dificuldades com o uso dos registros em língua natural no trabalho de especificar exatamente a estratégia de solução.

Na segunda experimentação, foi possível observar procedimento semelhante, relativo à dificuldade de expressão do método de resolução em língua natural e a necessidade de especificar o registro em linguagem algorítmica. Alguns dos grupos que haviam respondido de maneira incompleta à questão anterior, conseguiram uma descrição de algoritmo correta ou com pequenas falhas. Alguns grupos incluíram a saída do salário total ou a saída da taxa (porcentagem) relativa ao prêmio, além do próprio valor do prêmio. A Figura 8 exibe o protocolo de um desses grupos.

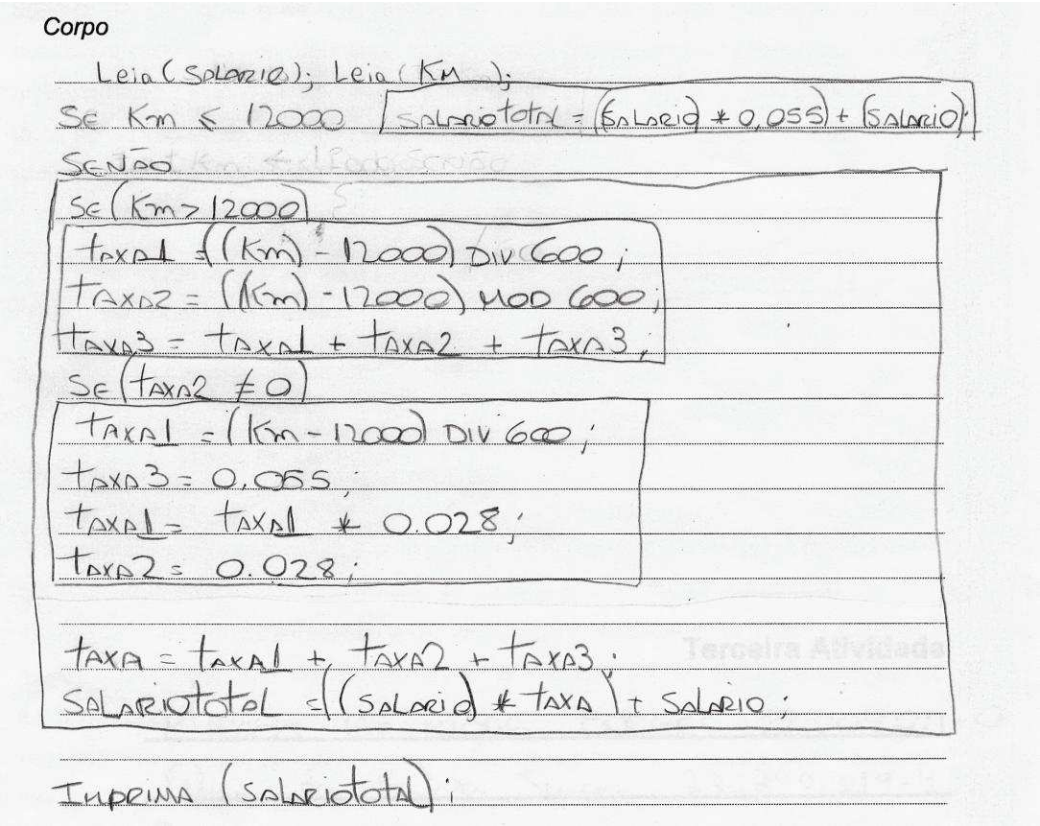


Figura 8 – Protocolo – algoritmo prêmio do motorista – segunda experimentação

Foi possível notar o trabalho com as conversões de registros de representação, que foi mais intenso quando os grupos iniciavam a elaboração do algoritmo com base na resposta à questão anterior.

É possível afirmar que o trabalho dos alunos nessa terceira parte da atividade, pode ser explicado segundo as fases da dialética ferramenta-objeto, de maneira similar ao que já havia ocorrido na primeira e na segunda atividade.

4.4 Análises – quarta atividade

4.4.1 Apresentação da introdução da atividade

A quarta atividade iniciou-se com a proposta de leitura de um texto com a apresentação dos recursos relativos às estruturas de controle de repetição. O texto é descrito a seguir.

Instruções de controle de repetição

As instruções de controle de repetição permitem a definição de fluxos de processamento repetitivos, ou seja: possibilitam a construção de algoritmos ou programas em que uma parte das instruções, apesar de figurar no texto do algoritmo ou do programa uma única vez, poderá ser executada por repetidas vezes.

Esses mecanismos de funcionamento correspondem a uma das características essenciais de um sistema computacional: a capacidade de repetir, com rapidez e sempre da mesma maneira, uma seqüência de ações por numerosas vezes.

As estruturas de repetição são compostas pela instrução de controle propriamente dita e um bloco de instruções que fica sujeito ao controle, ou seja, que pode ter a execução repetida por várias vezes. As estruturas de repetição com controle por expressões lógicas podem ser dispostas em duas categorias: controle com pré-teste e controle com pós-teste lógico.

A forma geral de uma estrutura de repetição por controle lógico com pré-teste é a seguinte:

```
enquanto <expressão de controle> faça
    <bloco de instruções> ;
<próxima instrução>;
```

A ***<expressão de controle>***, da mesma forma que ocorre nas instruções de controle de seleção, é uma expressão lógica que pode ser constituída a partir dos operadores de relação de ordem e dos operadores lógicos. Durante a execução, quando o fluxo de processamento chega a essa instrução, a ***<expressão de controle>*** é avaliada, se o valor for verdadeiro o ***<bloco de instruções>*** é executado, após sua execução a ***<expressão de controle>*** é avaliada novamente e o mecanismo se repete; se o valor da ***<expressão de controle>*** for falso o bloco não é executado e o fluxo segue para a ***<próxima instrução>***, com o encerramento da execução do anel de repetição. Com essa primeira forma de controle repetitivo se na primeira vez em que a ***<expressão de controle>*** é avaliada seu resultado for falso, o ***<bloco de instruções>*** não será executado nenhuma vez.

A sintaxe correspondente na linguagem C/C++ é:

```
while(<expressão de controle> ){
    <bloco de instruções> ;
}
<próxima instrução>;
```

A forma geral de uma estrutura de repetição por controle lógico com pós-teste é a seguinte:

```
faça
  <bloco de instruções> ;
enquanto <expressão de controle> ;
<próxima instrução>;
```

Seu mecanismo de funcionamento é semelhante ao anterior, mas com essa forma de estrutura de controle o ***<bloco de instruções>*** é executado pela primeira vez antes da avaliação da ***<expressão de controle>***, ou seja, nessa estrutura o ***<bloco de instruções>*** é executado pelo menos uma vez.

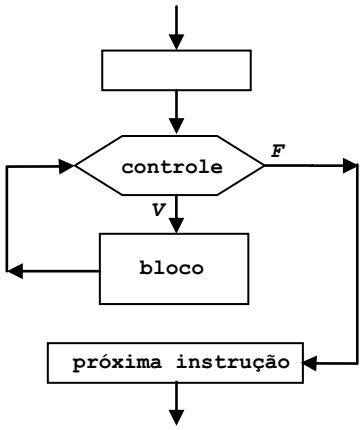
A sintaxe correspondente na linguagem C/C++ é:

```
do{
    <bloco de instruções> ;
}while(<expressão de controle> );
<próxima instrução>;
```

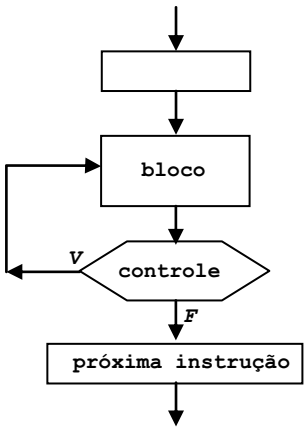
Na construção dessas formas de estrutura de controle pode-se prever que, em algum momento, alguma das ações contidas no ***<bloco de instruções>*** deve promover a alteração do valor lógico da ***<expressão de controle>*** , pois se isso não ocorrer pode ficar configurado um anel de repetição infinita.

As figuras abaixo representam aquelas duas formas de estruturas de controle de fluxo de processamento repetitivo.

controle lógico por pré-teste



controle lógico por pós-teste



Exemplo:

```
recortes( )
leia(a); leia(b);
recorte←0;
enquanto a>0 faça
   recorte←recorte+1;
   se b>a então  x←a; a←b; b←x;
   a←a-b;
   imprima(recorte,b);
imprima("fim");
```

Código em C/C++ :

```
#include <iostream>
using namespace std;
int main( ){
    int a, b, x, recorte;
    recorte=0;
    cout<<"digite a medida A: "; cin>>a;
    cout<<"digite a medida B: "; cin>>b;
    while(a>0){
        recorte=recorte+1;
        if(b>a){
            x=a; a=b; b=x;
        }
        a=a-b;
        cout<<"  "<<recorte<<"o. recorte: "<<b<<endl;
    }
    cout<<"fim"<<endl;
    system("PAUSE");
    return(0);
}
```

Questões

- Qual a finalidade das estruturas de controle de repetição em um algoritmo ou programa?

Observe a seqüência de instruções abaixo:

```
...
1.    int soma, parcela;
2.    soma=0; parcela=5;
3.    while(parcela<200){
4.       soma=soma+parcela;
5.       parcela=parcela*3;
6.    }
7.    cout<<"valor da parcela: "<<parcela<<endl;
8.    cout<<"valor da soma: "<<soma<<endl;
...
```

- Quantas vezes será executada a linha 2 dessa seqüência de instruções?
- Quantas vezes será executada a linha 4 dessa seqüência de instruções?
- Quantas vezes será avaliada a expressão lógica de controle **`parcela<200`**?
- Que valores serão exibidos como resultados da execução?

4.4.2 Análise *a priori* – primeira parte da atividade

A leitura inicial e a primeira questão proposta devem conduzir o grupo a uma reflexão sobre as possibilidades de mecanismos de controle para a constituição de fluxos de processamento repetitivos.

Esses elementos deverão compor parte do antigo a ser acessado, sob o olhar da dialética ferramenta-objeto, para o trabalho na terceira parte da atividade.

Os mecanismos representados pelos registros dos controles de repetição remetem diretamente à noção de processo dinâmico (movimento "circular" do fluxo de processamento) controlado pelas instruções do algoritmo, mais especificamente: a partir das expressões de controle utilizadas.

Uma das expectativas é que os grupos declarem explicitamente a noção de processo dinâmico com o emprego de expressões como: *o fluxo volta ...* ou *o fluxo segue ...* ou *o fluxo pula para* ..., em algumas fases dessa atividade.

As próximas questões exigem do estudante a simulação do processo, e ao simular a execução, a idéia de fluxo de processamento repetitivo (anel de repetição) deve ser associada à constituição formal do registro da estrutura de controle de repetição. A intenção é que o aluno perceba e domine a idéia de que o registro de representação da estrutura de controle deve ser relacionado à noção de movimento circular do fluxo de processamento, e que cada instrução do bloco vinculado ao controle de repetição, ao ter sua execução repetida por várias vezes, produzirá efeito distinto a cada vez.

4.4.3 Observações e análise *a posteriori* – primeira parte da atividade

Na primeira experimentação, a quarta atividade foi realizada por três grupos, um com três alunos e outros dois com dois alunos cada. Na aula normal que antecedeu a essa data, a turma trabalhou com a apresentação e as primeiras aplicações das estruturas de controle de repetição em linguagem algorítmica.

Na segunda experimentação houve a participação de dez grupos, cinco com dois alunos e cinco com três alunos.

Nas duas experimentações, a leitura e as discussões relativas às primeiras questões foram realizadas pelos grupos em cerca de vinte minutos. Durante as conversas iniciais foi possível perceber algumas expressões indicativas da idéia de movimento: alguns alunos gesticularam, para indicar o fluxo de processamento, com movimentos circulares de mão e braço e, além disso, um

deles com a caracterização da “descida” mais lenta do que a “subida” ao apontar que durante a “descida” ocorre a execução do bloco de instruções e que na “subida” ocorre apenas a volta ao topo da estrutura; os diálogos incluíram termos como: *“... aí ele volta no início ...*”, “*... ele para quando o valor ultrapassa ...*”, “*... antes de continuar, o algoritmo vê a condição ...*”, “... ele vai voltar e repetir...”, “...vai para cima e faz outra vez...”

O primeiro grupo da primeira experimentação não registrou no caderno da atividade a resposta à primeira questão, os outros dois grupos responderam assim: “*Executa repetição de um código.*” e “*Repetir a linha de comando até que o resultado especificado na condição seja obtido ou se torne falso.*”, a primeira resposta não faz qualquer referência ao controle do processo.

A interferência do professor poderia ter ocorrido com o sentido de melhorar as respostas registradas. As próximas três questões (*Quantas vezes será executada a linha 2 ...*, *Quantas vezes será executada a linha 4 ...* e *Quantas vezes será avaliada ...*) foram respondidas de maneira correta pelo segundo e pelo terceiro grupo, e de forma incorreta pelo primeiro grupo; não foi possível perceber a origem das falhas desse primeiro grupo, as anotações registradas nos rascunhos indicam que o trabalho de simular a execução foi realizado e que estava pelo menos parcialmente correto, os diálogos também indicavam que o trabalho era conduzido da forma esperada.

Na segunda experimentação a resposta à primeira questão de um dos grupos foi “*repetir determinado bloco de instrução para determinada condição*”, mais um dos grupos registrou resposta semelhante, e os outros não especificaram a associação entre o processo de repetição e o controle respectivo, estabelecido por uma expressão lógica. Parece que o processo de repetição fica em primeiro plano, e o controle em plano inferior.

A questão “*Que valores serão exibidos ...*” foi respondida satisfatoriamente pelos três grupos: o primeiro grupo, que registrou respostas incorretas às questões anteriores, mostrou, nesta resposta, que realizou a simulação de forma satisfatória e apenas falhou ao apontar o valor da `parcela` obtido na penúltima execução do bloco e não aquele obtido na última; o segundo grupo falhou ao apontar todos os valores intermediários, ou seja: o grupo apontou a evolução dos

valores das duas variáveis `parcela` e `soma` durante toda a execução do anel de repetição, mas como as instruções de saída foram dispostas adiante da estrutura de controle de repetição, apenas os últimos valores seriam exibidos, a Figura 9 exibe o protocolo correspondente; o terceiro grupo falhou ao calcular, na última operação do processo, o produto 135x3 como 505, vale acrescentar que apenas o segundo grupo lançou mão do uso da calculadora que foi oferecida.

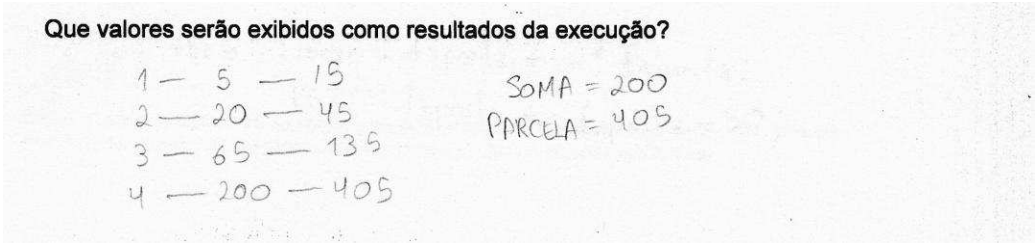


Figura 9 – Protocolo – simulação de algoritmo – primeira experimentação

A última questão dessa série, na primeira experimentação, foi respondida assim: "*Não vai ser executado nenhuma vez se parcela<10.*", "*Ela terá a sua estrutura de repetição executada menos vezes.*" e "*Só será executado o bloco do while uma vez.*", o terceiro grupo respondeu de forma correta, o primeiro de forma incorreta e o segundo de forma incompleta.

A resposta do primeiro grupo deixa evidente que esses alunos não conseguiram trabalhar corretamente a simulação do processo representado; o segundo grupo respondeu a essa questão sem realizar a simulação do processo correspondente, mas percebeu que a modificação na expressão de controle acarretaria uma quantidade menor de repetições do bloco correspondente; a resposta desse segundo grupo indica uma necessidade de se reformular a proposta da questão com a inserção de uma indicação para que a simulação seja realizada.

Na segunda experimentação, foi possível perceber que todos os grupos realizaram a simulação do processo e responderam corretamente às questões que indagavam sobre as quantidades de execuções. Dois dos grupos, colocaram como resposta à questão que solicitava os resultados exibidos, valores intermediários no processo. Esses grupos não vincularam corretamente o bloco de instruções sujeito à repetição ao respectivo controle. A Figura 10 exibe um desses protocolos.

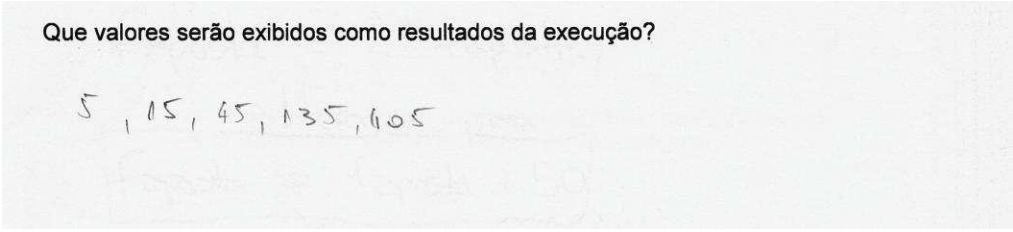


Figura 10 – Protocolo – simulação do algoritmo – segunda experimentação

4.4.4 Apresentação da segunda parte da atividade

Na segunda parte da atividade é proposto o trabalho com o problema das apostas do jogador. O texto da proposta está transcrito a seguir.

> Proposta do problema:
>
> "Numa casa de jogos, um apostador realiza o seguinte procedimento:
> inicialmente aposta 15 fichas e perde,
> em seguida aposta 30 fichas e perde outra vez,
> depois aposta 60 fichas e perde novamente,
> e assim, sucessivamente, sempre dobra a quantidade de fichas da aposta anterior e sempre perde.
>
> Conhecendo-se a quantidade de fichas de que dispõe inicialmente, como determinar a quantidade de apostas que o jogador pode realizar?"
>
> No disco (CD) há um pequeno aplicativo (**aposta_jogador**) que ilustra o método de resolução desse problema com descrição em linguagem algorítmica, o aplicativo ilustra também a execução do algoritmo.
>
> Questões:
>
> - Se a quantidade de fichas disponíveis inicialmente é 100, quantas apostas o jogador pode fazer? Calcule sem utilizar o aplicativo e depois confronte sua resposta com o resultado fornecido pela execução do programa.
> - Qual seria a quantia mínima necessária se o desejo do apostador fosse realizar exatamente 6 apostas. Confronte seu resultado com a resposta gerada pelo aplicativo.
> - Que modificações seriam necessárias no algoritmo se em vez de dobrar o valor da aposta a cada vez, o jogador aumentasse em 50 fichas a quantidade da aposta anterior?

4.4.5 Análise *a priori* - segunda parte da atividade

As questões propostas devem levar o grupo a operar e observar a animação do algoritmo, a intenção é evidenciar a circularidade do fluxo de processamento, especialmente a avaliação da expressão lógica do controle e o desdobramento dessa ação com a execução repetida do bloco de instruções. O registro de representação do controle de repetição deve ser percebido em seu aspecto dinâmico, responsável por estabelecer o fluxo repetitivo de execução.

A segunda questão deve levar ao processo de cálculo manual, e depois à verificação da resposta com a execução do aplicativo e sua observação; novamente o grupo deverá observar a execução do aplicativo e a evolução dos conteúdos das variáveis. O aluno deve perceber que a cada execução, apesar de serem as mesmas instruções no bloco sob o controle de repetição, os efeitos não serão os mesmos, ou seja: as instruções, a cada execução, trabalham com conteúdos de variáveis diferentes.

O registro de representação (linguagem algorítmica) deve ser entendido com seu aspecto de estrutura, que comporta a forma de controle, a expressão de controle e o bloco de instruções sujeito ao processo de repetição.

Com a terceira questão o que se pretende é que o aluno observe especialmente as operações de atribuição de valor dispostas no bloco sob a ação da estrutura de controle de repetição, e sua modificação com a alteração indicada.

As observações da animação devem levar o estudante a perceber como se dá o mecanismo de funcionamento definido pelo controle de repetição e a evolução dos valores armazenados nas variáveis `fdisp`, `faposta` e `qtd`.

4.4.6 Observações e análise *posteriori* - segunda parte da atividade

A tarefa na segunda parte desta atividade envolvia o entendimento da proposta do problema das apostas do jogador, e a operação e a observação do aplicativo com a animação do algoritmo que foi oferecido.

Na primeira experimentação, as respostas produzidas para a primeira questão (*Se a quantidade de fichas disponíveis ...*) foram: "*2*", "*jogada 3 não poderá ser feita pois o jogador não possui fichas disponíveis*" e "*serão 2 apostas*"; o segundo grupo anotou um pequeno esquema com a indicação dos cálculos realizados que levaram àquela resposta.

Nas duas experimentações, foi possível notar que o tratamento foi conduzido de acordo com o que foi proposto, ou seja: primeiro os alunos reproduziram o mecanismo das apostas com conversas e algumas anotações em rascunho, e depois colocaram em execução o aplicativo para a confirmação de suas conclusões.

Nessa fase do trabalho, um dos alunos, antes de colocar o aplicativo em operação, declarou sua previsão de que seria necessário construir um controle de repetição e nesse controle a chave seria a quantidade de fichas disponíveis em confronto com o volume da aposta: "*... antes de fazer o jogo o sistema vai ter ... pegar a aposta e o dinheiro que sobrou da outra vez e olhar se dá ...*", apesar de não ter sido solicitado, o aluno projetava o esboço do método de resolução com o controle de repetição e a noção de movimento explícita.

Na segunda experimentação, em um dos grupos foi registrado o diálogo: "*para poder apostar tem que ver se tem suficiente* (quantidade de fichas)" e outro aluno respondeu: "*quem vai fazer isso é o enquanto* (a instrução de controle)*, é aí que coloca a condição para continuar apostando*". Outra declaração registrada: "*nessa área (bloco na estrutura de repetição) fica o cálculo da aposta e da sobra*". Essas afirmações levam a concluir que esses alunos conseguiram interpretar adequadamente a vinculação entre a representação estática (texto do algoritmo) e a realização do processo dinâmico correspondente.

As outras duas questões propostas também foram respondidas corretamente pelos grupos. A segunda questão foi efetivamente trabalhada a partir da experimentação do aplicativo e o terceiro grupo, além da operação do aplicativo, indicou nos rascunhos um esboço do processo de cálculo correspondente à situação proposta. O segundo grupo registrou a seguinte resposta à terceira questão: "*A linha de comando que diz que a aposta é dobrada, aposta=aposta*2 seria assim: aposta=aposta+50*", é interessante notar nessa resposta que, além de atribuir um aspecto de vida própria ao algoritmo (*a linha de*

comando que diz ...) o grupo explicitou diretamente a conversão realizada entre os registros de representação em língua natural (*a aposta é dobrada*) e em linguagem algorítmica (`aposta←aposta*2`).

Na segunda experimentação, as respostas registradas por todos os grupos foram corretas. Dois dos grupos, registraram nos rascunhos anotações semelhantes ao seguinte: muda dobrar (2xaposta) para aumenta 50 (aposta+50), isso evidencia o mesmo trabalho de conversão de registros, já indicado no parágrafo anterior. A Figura 11 exibe um desses rascunhos.

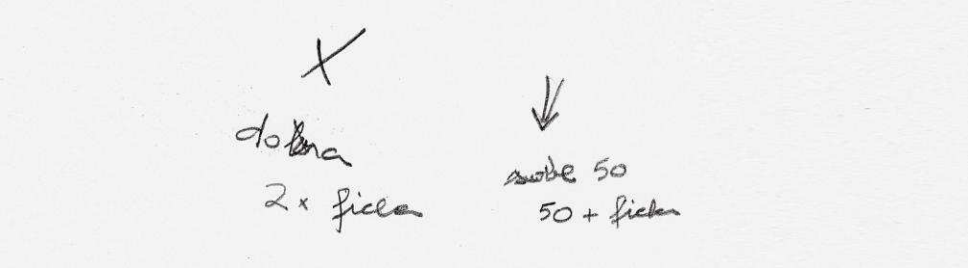


Figura 11 – Protocolo – fragmento de rascunho de um grupo

Tendo em vista as respostas registradas e as observações realizadas, a conclusão é que essa fase da atividade alcançou o seu propósito: favorecer o entendimento da constituição e da dinâmica correspondente ao registro (linguagem algorítmica) de uma estrutura de controle de repetição.

4.4.7 Apresentação da terceira parte da atividade

A terceira parte dessa atividade é a proposta de trabalho com o problema da propagação do vírus. A redação da proposta é a seguinte:

> Considere a seguinte proposta de problema:
>
> "Um vírus de computador propaga-se contaminando arquivos de tipo texto (txt) e arquivos de tipo executável (exe) do disco rígido de um microcomputador, assim:
>
> - a cada dia, um arquivo *txt* já afetado pelo vírus contamina outro arquivo de texto (*txt*) e um arquivo executável (*exe*) ainda não afetados;
> - a cada dia, um arquivo *exe* já afetado pelo vírus contamina outros dois arquivos executáveis (*exe*) e um arquivo de texto (*txt*) ainda não afetados.

Se em uma determinada data um computador for contaminado por esse vírus, em um único arquivo *txt*, depois de quantos dias a quantidade total de arquivos contaminados terá superado uma dada quantidade?"

Supor que nenhum arquivo contaminado seja removido e que as quantidades de arquivos *txt* e *exe* gravados no disco sejam suficientes para que a propagação ocorra livremente.

- Complete o quadro, até a data 6, com as informações correspondentes:

data	arquivos contaminados		
	txt	exe	total
0 (inicial)	1	0	1
1	2	1	3
2	5	5	10
3	15	20	35
4			
5			
6			

Experimente a execução do aplicativo **propaga_virus** que se encontra no disco (CD) fornecido.

Observação: apenas as variáveis principais são exibidas no quadro de variáveis.

Questões

- Complete de forma adequada as lacunas dispostas nas afirmações abaixo com o emprego de algumas das expressões:

  a quantidade

  o dobro da quantidade

  o triplo da quantidade

  o quádruplo da quantidade

  A quantidade de arquivos de texto contaminados numa determinada data pode ser obtida somando-se ***LACUNA 1*** *de arquivos de texto contaminados na data anterior com* ***LACUNA 2*** *de arquivos executáveis contaminados na data anterior.*

  LACUNA 1 :

  LACUNA 2 :

  A quantidade de arquivos executáveis contaminados numa determinada data pode ser obtida somando-se ***LACUNA 3*** *de arquivos executáveis contaminados na data anterior com* ***LACUNA 4*** *de arquivos de texto contaminados na data anterior.*

LACUNA 3 :

LACUNA 4 :

- Se **txt** e **exe** representam as quantidades atuais de arquivos de texto e de arquivos executáveis contaminados e **txtant** e **exeant** as correspondentes quantidades na data anterior, complete as instruções de atribuição que estabelecem as relações entre essas quantidades:

txt←

exe←

- Faça a descrição de um algoritmo que represente um método de resolução do problema proposto. Utilize identificadores de variáveis que indiquem o significado, diante da proposta do problema, de cada informação representada.
- Faça a implementação e testes do programa correspondente ao algoritmo que você construiu. Transcreva na área abaixo o texto de seu programa.

4.4.8 Análise *a priori* - terceira parte da atividade

A primeira questão proposta (*Complete o quadro ...*) deve acarretar a releitura da proposta do problema, para completar o entendimento da situação, e, a partir da execução do aplicativo, deve produzir a compreensão da evolução dos conteúdos das variáveis, representadas no topo do quadro fornecido e na janela do aplicativo em execução.

O aplicativo deve ser um suporte importante no entendimento da proposta do problema e do mecanismo dinâmico de cálculo necessário para a resolução.

No quadro a ser preenchido, a evolução de datas fica definida nas várias linhas com início na primeira (data inicial mais ao alto) e crescimento de cima para baixo, como é usualmente feito. No quadro de variáveis do aplicativo essa representação é diferente, a intenção é ressaltar que a cada nova atribuição de valor à variável seu conteúdo anterior é perdido, o fundo de cada célula do quadro com o conteúdo atual da variável é branco (logo abaixo do identificador da variável), e as outras células têm fundo cinza e seus conteúdos sofrem um deslocamento para baixo a cada nova atribuição. A Figura 5 é um exemplo de janela obtida com a execução do aplicativo “propagação do vírus”.

A intenção é que o aluno perceba o conteúdo de cada variável sofrendo modificações a cada execução do bloco de instruções na estrutura de repetição.

O registro de representação da estrutura de controle deve ser percebida também em seus aspectos de mecanismo dinâmico.

As próximas duas questões propostas (*Complete de forma adequada as lacunas ...* e *... complete as instruções de atribuição ...*) têm o objetivo de levar os alunos a explicitarem as relações entre as quantidades de arquivos afetados pelo vírus em uma data, e as quantidades da data anterior.

Na verdade, essas relações já estão presentes na proposta do problema e terão sido trabalhadas também durante a tarefa de completar o quadro; o propósito é que os alunos possam colocar em foco especificamente tais relações e produzam as correspondentes expressões de explicitação, com o emprego da língua natural na segunda, e da linguagem algorítmica na terceira questão. Esse exercício é uma prática de conversões de registros: língua natural, elementos algébricos e linguagem algorítmica.

Na terceira questão foi encaminhada a idéia de se trabalhar com dois pares de variáveis para a representação das quantidades de arquivos afetados, assim: `txt` e `txtant` para os arquivos de tipo txt e `exe` e `exeant` para os arquivos executáveis, mas não foi dada ênfase a essa construção. Espera-se que os alunos percebam a necessidade disso ao elaborarem a construção do algoritmo e do programa.

Ao final dessa fase, antes da elaboração do algoritmo, os grupos devem ter cumprido as primeiras etapas da dialética ferramenta-objeto. O *antigo*, nesse caso, constituído pelos elementos das linguagens, e dos elementos algébricos que definem as relações entre as quantidades envolvidas; a *pesquisa*, a *explicitação* e o *novo implícito*, com os trabalhos de preencher o quadro e a elaborar as primeiras respostas. A quinta fase (*institucionalização*) se desenrola nas fases finais: construções e testes do algoritmo e do programa.

Nas duas últimas questões são solicitadas as construções do algoritmo e do programa; o esperado é que o grupo retome os desenvolvimentos anteriores, especialmente as respostas das duas questões anteriores, e organize tais trabalhos com vistas à produção do algoritmo e do programa.

Além das relações entre as quantidades de arquivos afetados pelo vírus, o algoritmo deve conter o processo correspondente à evolução das datas e também

o controle de repetição com base na quantidade total de arquivos afetados, e a informação inicial que é a quantidade que deve ser superada.

Vale ressaltar que, no encaminhamento das duas questões anteriores, esses aspectos não foram destacados, mas tanto no quadro da primeira questão como na janela do aplicativo, o avanço das datas é explicitado, e a condição que deverá definir o controle de repetição aparece em destaque (fundo vermelho) em uma das células do aplicativo.

Assim, os estudantes deverão conjugar os elementos já produzidos por eles mesmos, nas três primeiras questões, e os elementos disponíveis com a visualização do aplicativo em execução. A última questão além de solicitar a implementação do programa, deve levar o grupo a realizar o confronto entre os resultados produzidos pelo programa construído e aqueles fornecidos a partir da execução do aplicativo, mas essa indicação do confronto não foi colocada explicitamente.

4.4.9 Observações e análise *a posteriori* - terceira parte da atividade

A intenção nessa terceira parte da atividade era que os alunos produzissem a descrição do algoritmo para a resolução do problema da propagação do vírus e a implementação do programa correspondente.

Na fase inicial dos trabalhos, os grupos realizaram a leitura da proposta do problema e experimentaram a execução do aplicativo que foi oferecido. Nas duas experimentações foi possível verificar que o aplicativo foi acionado várias vezes, em diferentes momentos dos trabalhos.

Na primeira experimentação, as respostas registradas pelos três grupos à primeira questão (“*complete o quadro...*”), todas corretas, indicam que os estudantes entenderam o mecanismo de propagação descrito na proposta do problema e representado pela execução do aplicativo que complementava a proposta.

Foi possível perceber que o aplicativo foi executado pelos estudantes ao menos uma vez enquanto o grupo buscava completar o cálculo, para completar o

entendimento do mecanismo descrito ou para confrontar os valores que seriam registrados nas células do quadro. Entre as finalidades do acesso ao aplicativo estavam exatamente facilitar o entendimento da proposta e permitir o confronto de valores produzidos pelo trabalho do grupo e aqueles produzidos pela execução do aplicativo.

A questão seguinte também foi respondida corretamente pelos três grupos, a menos da segunda parte registrada pelo terceiro grupo em que houve um engano, que deve ser atribuído a uma falha na leitura da proposta da questão ou no momento de escrever a resposta, uma vez que a terceira questão, que exigia a conversão das relações solicitadas na questão anterior para a linguagem algorítmica, foi respondida corretamente pelos três grupos.

O comportamento dos alunos, durante a elaboração das respostas a essas duas questões, mostra que todos os grupos retomaram a execução do aplicativo disponível para certificarem-se das relações percebidas e declaradas.

Na segunda experimentação, também quase todas as respostas registradas estavam corretas, apenas o registro de dois valores no quadro de quantidades estavam incorretos. Tal falha pode ser interpretada como decorrente de falta de atenção ou cuidado, pois as relações necessárias para seus cálculos foram descritas corretamente.

Na primeira experimentação, na última parte dessa atividade que solicitava a construção do algoritmo e a implementação do programa correspondente, apesar das relações entre as quantidades de arquivos afetados terem sido declaradas satisfatoriamente, apenas o terceiro grupo conseguiu realizar as construções completamente corretas.

O primeiro grupo conseguiu perceber um erro (relativo às vinculações entre as quantidades de arquivos afetados) durante a fase de testes do programa produzido e assim corrigir o programa, mas não tiveram o cuidado de corrigir a descrição do algoritmo, vale ressaltar novamente a importância da disponibilidade do aplicativo para que o grupo pudesse realizar o confronto entre os resultados e a partir disso rediscutir as construções elaboradas.

O segundo grupo, apesar de ter respondido corretamente às questões anteriores com as relações entre as quantidades de arquivos afetados, não

registrou corretamente no caderno da atividade nem o algoritmo e nem o programa, mas construiu corretamente a expressão na instrução de controle e também o mecanismo de evolução das datas, o grupo percebeu que a construção desenvolvida não estava correta e preferiu encerrar os trabalhos com a entrega do caderno mesmo com o erro percebido, o grupo não pode permanecer trabalhando em virtude de outro compromisso de um dos alunos, naquele momento já estavam em atividade há um pouco mais de uma hora e meia.

O terceiro grupo completou corretamente a implementação do programa e também fez alguns testes confrontando os valores gerados pelo programa com aqueles gerados pelo aplicativo, também para esse grupo o acesso ao aplicativo se mostrou importante. Os três grupos construíram corretamente a instrução de controle de repetição e o mecanismo de avanço das datas, esses eram os elementos ainda não trabalhados nas questões anteriores e articularam de forma adequada essas novas construções com aquelas já esboçadas nas fases anteriores dessa última parte da quarta atividade.

Na segunda experimentação, os registros dos algoritmos e programas também foram satisfatórios, a maioria dos grupos descreveu corretamente os dois registros.

Um dos grupos não completou a construção do programa, e nesse grupo, a construção do algoritmo não estava correta. Um dos alunos do grupo, alguns dias depois da realização da atividade, procurou o professor-pesquisador para apresentar a resolução correta dessa última fase da atividade, e nessa ocasião afirmou que os outros dois membros do grupo não puderam concluir as tarefas, mas que ele percebeu a falha no algoritmo construído e afirmou que se sentia desconfortável com o fato. O algoritmo e o programa apresentado pelo aluno nesse momento estavam corretos.

Outros dois outros grupos apresentaram os algoritmos e programas com falhas: o primeiro não reproduziu as relações corretamente construídas nas questões anteriores, e o outro não percebeu corretamente o controle da repetição que deveria ser definido. Nesses dois grupos, a fase final em que deveriam analisar e discutir os resultados apresentados pelo programa, parece não ter sido realizada.

Apesar dessas ocorrências de erros, pode-se afirmar que a atividade cumpriu bem seus propósitos: os alunos exercitaram os trabalhos de conversão de registros, a maioria dos grupos descobriu e construiu de forma satisfatória o processo de resolução e, em termos gerais, os alunos demonstraram empenho e vontade na elaboração dos trabalhos.

4.5 Apresentação e análises das entrevistas

Nas duas semanas seguintes à realização da quarta atividade da Engenharia Didática, foram realizadas duas pequenas sessões de entrevistas, com cinco dos alunos que participaram da segunda experimentação. A transcrição dos diálogos dessas entrevistas está no Apêndice C.

Os participantes das entrevistas foram selecionados em função das suas disponibilidades de horário. Foram organizados dois grupos de alunos: o primeiro com três alunos e o outro com dois alunos. Cada sessão de entrevista teve duração aproximada de vinte minutos.

O objetivo com tais entrevistas foi obter as impressões gerais e opiniões dos participantes sobre as atividades.

As entrevistas tiveram como eixo principal três perguntas que foram dirigidas aos grupos:

Vocês podem falar um pouco sobre a participação nas atividades?

Há algum aspecto nas atividades que mereça destaque?

Vocês gostariam sugerir melhorias na constituição das atividades?

Uma das impressões declaradas pelos alunos, na primeira entrevista, envolve a forma de realização dos trabalhos: um dos entrevistados declarou, com a concordância dos outros, que o seu grupo funcionou efetivamente como equipe de trabalho, todos os componentes estavam empenhados e tinham a preocupação de partilhar as dúvidas, discussões e descobertas. Na outra entrevista observaram-se outras afirmações com sentido semelhante: mesmo os

alunos com mais dificuldades acompanharam o trabalho e puderam contribuir para a elaboração das tarefas.

Nas duas entrevistas houve destaque para o aspecto de aproximação entre o trabalho de criação do algoritmo e o trabalho de implementação do respectivo programa. Esse destaque não era esperado, pois os alunos, além das aulas normais semanais em laboratórios de informática, em que se faz a prática de implementação de programas, contam com a possibilidade de acesso aos mesmos laboratórios em horários que lhes sejam mais convenientes; contam também com as facilidades de acesso aos recursos necessários em computadores pessoais, pois o sistema utilizado não exige qualquer característica especial em termos de equipamento, e é gratuito. Essas observações mostram a importância que os alunos atribuem ao trabalho realizado em seu ciclo completo, em uma única sessão: abordagem e tratamento do problema, criação do algoritmo, e construção e testes do programa.

A respeito do tempo programado para as atividades, as impressões declaradas indicam que foi adequado, a menos da segunda atividade, em que um dos entrevistados afirmou ter havido alguma dificuldade para concluir as tarefas no limite do tempo estabelecido. O esperado era que a maioria dos alunos apontasse algum entrave, pois o que ocorreu efetivamente, nas quatro atividades da segunda experimentação, foi o encerramento das tarefas antes do limite de tempo para pouquíssimos grupos.

Sobre o aproveitamento, um dos entrevistados declarou que foi muito bom ter participado, pois o trabalho com a atividade levou a uma "*ampliação do conhecimento*", e ajudou a "*abrir a cabeça*". Outro aluno afirmou que seu desempenho melhorou, pois agora ele conseguia começar a "*encarar*" a proposta dos problemas e conseguia entender melhor os significados de um algoritmo. Essas declarações levam a concluir que esses alunos tiraram proveito da participação, e perceberam que houve alguma evolução em termos da própria aprendizagem.

Quando questionados sobre o interesse em relação à coleção de problemas que foram trabalhados nas atividades, os alunos consideraram que era um conjunto adequado, e entre os que mais vezes foram lembrados estão o problema do prêmio do motorista e o problema da propagação do vírus. Outros

problemas também foram citados: problema da farinha, problema das apostas do jogador e problema do transporte de carvão. Aqueles que foram citados mais vezes são os propostos nas duas últimas atividades, e cujos métodos de resolução os alunos construíram. Talvez isso justifique a maior freqüência de indicações.

Sobre a possibilidade de trabalhar com os programas ilustrativos, os entrevistados confirmaram o propósito planejado: houve declarações com o sentido de que os aplicativos deixam evidente o mecanismo de funcionamento, e apontaram principalmente os mecanismos relativos aos controles de seleção e de repetição, mas também foi apontado o mecanismo de transformação de dados. Um dos alunos afirmou que "*fica melhor de ver aquilo* (o que ocorre com a execução do algoritmo)", outro declarou que "*dá para ver ... ela muda* (a definição ou redefinição do conteúdo de uma variável)".

Dois dos entrevistados citaram exatamente a finalidade dos aplicativos para favorecer o entendimento da situação do problema (os dados que devem alimentar o processo, e os dados que devem ser os resultados produzidos pelo processo). A terceira finalidade, que era oferecer referências no momento de testar e analisar as próprias construções, também foi citada pelos alunos.

4.6 Conclusões e considerações finais

Neste trabalho de pesquisa, o alvo estabelecido foi investigar como o estudante revela, trata e domina a noção de processo dinâmico inerente a um algoritmo ou a um programa. Dessa forma, o planejamento dos experimentos incluiu a observação dos alunos ao trabalharem o desenvolvimento das quatro atividades, pois apenas o acesso aos protocolos escritos não seria suficiente para a constituição das conclusões. Isoladamente, o registro escrito de um algoritmo (ou de um programa) não revela explicitamente a mobilização da noção de processo dinâmico.

As quatro atividades foram estruturadas tendo em vista os referenciais teóricos utilizados, o conjunto e a evolução dos assuntos tratados nas seis semanas iniciais do curso da disciplina, e o objetivo desta pesquisa.

As escolhas dos problemas a serem trabalhados e a organização, tanto dos problemas como dos roteiros para a constituição dos respectivos métodos de resolução, levaram em conta as fases de atividades indicadas na dialética ferramenta-objeto, e as atividades de conversão de registros apontadas na teoria dos registros de representação. Por outro lado, o objetivo de explorar os aspectos dinâmicos que os algoritmos representam, levaram à inserção, nas atividades, dos aplicativos que ilustram o mecanismo dinâmico relativo à execução dos mesmos. O propósito principal da colocação desses aplicativos foi favorecer a elaboração, pelos alunos, de modelos mentais adequados para completarem a representação estática estabelecida nos textos dos algoritmos.

O trabalho de planejamento das atividades, que seguiu as orientações da metodologia Engenharia didática, exigiu o estudo das teorias já relacionadas, levou à leitura de artigos científicos recentes que permitiram situar adequadamente esta investigação, e à reflexão sobre a forma de compor as tarefas propostas nas atividades.

As observações realizadas indicam que os estudantes demonstram a noção de processo dinâmico em suas falas, expressões e gestos. Em várias ocasiões foram percebidos gestos que podem ser considerados sinais típicos daquela noção. Alguns dos gestos já descritos ao longo das análises das atividades: movimento com uma das mãos e braço oscilando entre dois pontos; movimento circular (meia volta) de uma das mãos com os dedos indicador e médio em “V”; oscilação da cabeça e dos olhos para cima e para baixo; oscilação da cabeça de um lado para o outro; movimento de uma caneta com a indicação de dois percursos alternativos; simulação de traços em seqüência, com uma caneta, indicando a execução, uma a uma, das instruções contidas no algoritmo; movimento de uma das mãos recolhendo algo (um dado obtido no processo) e depositando-o na outra mão (essa outra mão faz a representação de uma variável que recebe um novo conteúdo).

Além dos gestos, várias expressões, também já mencionadas, são indicativos da noção de processo dinâmico: “*ele joga o dado* (um valor)”, “*vai*

seguir para baixo", "*volta e faz outra vez*", "*ele pula esta parte e vai ...* ", "*o programa escolhe se vai por este caminho ou não*".

O conjunto de observações e análises permite afirmar que as atividades mostraram-se adequadas aos objetivos deste trabalho de investigação. As atividades, elaboradas segundo a metodologia da Engenharia Didática, promoveram vários exercícios de conversão e de tratamento de registros de representação, tais registros representavam processos de resolução de problemas computacionais. As conversões e os tratamentos envolveram registros em língua natural, registros algébricos, registros em linguagem algorítmica, registros em linguagem de programação, além de esboços (registros não formais) em que eram empregados fragmentos de registros algébricos, expressões em língua natural e elementos gráficos.

A compatibilidade verificada entre os elementos que configuraram as análises *a priori* e a constituição das análises a *posteriori* , permite concluir que os estudantes perceberam a noção de processo dinâmico dos algoritmos, revelaram essa noção em suas falas, gestos e registros escritos, e isso demonstra que os alunos incorporaram a noção, ou seja: agregaram esse aspecto (noção de processo dinâmico) aos registros dos algoritmos e dos programas. Com isso, os estudantes puderam mobilizar a noção de processo dinâmico nas suas atividades de concepção de algoritmos para a resolução de problemas computacionais.

Segundo a teoria dos registros de representação (Duval, 2008), são as atividades de conversão que produzem a construção dos conhecimentos relativos aos objetos trabalhados, neste caso, os algoritmos e os programas.

O aspecto em destaque neste trabalho, a noção de processo dinâmico inerente aos algoritmos, foi colocado em evidência a partir, principalmente, do suporte constituído pelos aplicativos de animação dos algoritmos. Esses aplicativos agregaram explicitamente, aos registros estáticos dos algoritmos, a noção de dinamismo.

Os alunos revelaram, em várias ocasiões, a presença dessa noção em suas construções, algumas vezes com gestos, outras vezes com expressões em língua natural, e também com a produção de esboços informais para os processos. Os aspectos mais freqüentemente observados eram relativos ao

mecanismo de fluxo de processamento, mas também a dinâmica de transformação de dados foi notada.

Os aspectos dinâmicos revelados nas falas e nos gestos dos estudantes, e algumas vezes nos registros escritos, indicam o domínio da noção de processo dinâmico. Pode-se afirmar que esse domínio foi importante tanto para as atividades de comunicação (a explicação e o entendimento de um algoritmo), quanto para as atividades de concepção. A noção de processo dinâmico foi incorporada aos registros de representação e aos modelos mentais que foram elaborados pelos estudantes.

Outro fator que apóia as conclusões deste trabalho é a organização das tarefas propostas em cada atividade. Essa organização tem por base as fases de desenvolvimento da dialética ferramenta-objeto (Maranhão, 2008), que orientam e explicam a construção de conhecimentos novos. No âmbito dos objetos da matemática, os conhecimentos novos se estabelecem como propriedades, proposições ou relações sobre aqueles objetos. No âmbito deste trabalho, os conhecimentos novos são os processos que descrevem os métodos de resolução de problemas, ou seja: o novo conhecimento é a descoberta do método de resolução de um problema computacional e a construção, em linguagem algorítmica ou em linguagem de programação, dos registros que representam o método descoberto. Em todas as atividades da Engenharia Didática, a maioria dos grupos conseguiu construir adequadamente os algoritmos e os programas solicitados.

A análise das entrevistas, realizadas ao final da segunda experimentação, também trouxe elementos que sustentam os resultados desta investigação. As impressões colhidas demonstram que os alunos se motivaram com as atividades propostas e avaliaram a participação como muito proveitosa. Afirmaram, também, que é mais interessante e mais adequada a prática de implementação logo após a elaboração do algoritmo. Nas aulas normais, em geral, a implementação dos programas é realizada cinco dias depois de terem trabalhado a construção dos algoritmos: as aulas de teoria (elaboração de algoritmos) ocorrem nas noites de sexta-feira, e as aulas de laboratório (implementação de programas) nas noites de quarta-feira.

A avaliação, pelos entrevistados, da coleção de problemas que constituíram as atividades, foi positiva: os alunos lembraram-se de todos os problemas e indicaram algum destaque para o problema do prêmio do motorista e para o problema da propagação do vírus. Algumas respostas tinham o sentido de sugerir que a forma de trabalho nas atividades fosse adaptada e levada para as aulas tradicionais.

Na primeira experimentação, foi colhida uma contribuição, de um dos grupos, que deverá ser considerada nos próximos planejamentos da disciplina. O grupo apontou o interesse em se aproximar de situações profissionais, já nas atividades dessa disciplina introdutória: "*poderíamos desenvolver projetos oferecendo soluções para empresa*". A inserção de atividades com essa característica deve favorecer a motivação do estudante.

A limitação principal relativa à realização dos experimentos, desenvolvidos nesta investigação, localiza-se na quantidade de alunos matriculados na disciplina. A turma que realizou a segunda experimentação, no segundo semestre de 2009, foi uma turma relativamente pequena (trinta e quatro matriculados), e para quantidades maiores o trabalho com a turma completa não teria sido possível. A alternativa seria estudar uma forma para levar essas atividades para as aulas de laboratório da mesma disciplina, mas isso esbarraria no propósito de promover a prática de implementação de programas, fundamental no plano das aulas de laboratório.

A realização desta investigação trouxe novas perspectivas para a atuação do professor-pesquisador, revelou novas alternativas que podem ser úteis tanto na concepção do plano da disciplina, como nos planos das atividades locais das aulas de teoria e das aulas de laboratório.

Um próximo eixo que pode ser vislumbrado a partir deste trabalho é a sua ampliação, com o propósito de atender as próximas etapas da disciplina, por exemplo: o trabalho com conceitos e recursos para a modularização de algoritmos e programas, neste novo bloco de conteúdos é muito importante a noção dinâmica de fluxo de dados que se estabelece pelos parâmetros definidos nos módulos auxiliares e pelos argumentos que figuram nas instruções que acionam tais módulos auxiliares.

Uma perspectiva que também pode ser considerada é a criação de um sistema para a construção e a execução animada de algoritmos, nos moldes da ideografia dinâmica de Lévy (1998). O sistema deveria oferecer os elementos e mecanismos básicos da linguagem algorítmica para as construções, e, de forma automática, permitir a execução animada do algoritmo construído. Nesse sentido, o sistema faria a função de estender as capacidades biológicas de raciocínio e memória: o estudante poderia experimentar diretamente suas construções e, assim, avaliar a adequação do processo logo em seguida à criação do algoritmo.

REFERÊNCIAS

ALMOULOUD, S. A. E COUTINHO, C. Q. S. **Engenharia Didática: características e seus usos em trabalhos apresentados no GT-19/ANPEd** REVEMAT – V3.6, p.62-77 - UFSC, 2008
Disponível em <http://www.redemat.mtm.ufsc.br/revemat/2008_pdf/ revista_2008_06_completo.pdf. Acesso em out. 2009.

ARAÚJO, J. L. e BORBA, M. C. **Construindo pesquisas coletivamente em Educação Matemática.** In: BORBA, M. C. e ARAÚJO, J. L. (orgs.). **Pesquisa Qualitativa em Educação Matemática** – Coleção Tendências em Educação Matemática. Belo Horizonte, MG. Autêntica,2004.

ARTIGUE, M. **Didactical Design in Mathematics Education** Proceedings of NORMA08 – Nordic Research in Mathematics Education, 2009

BARBOSA, L. M. **Ensino de algoritmos em cursos de computação**. São Paulo: Educ, 2001.

BERCHT, M.; FERREIRA, L. F.; SILVEIRA, S. R. Aprendendo a construir algoritmos através da mediação digital. **Revista Novas Tecnologias na Educação**, Cinted-UFRGS v.3 n.1, mai.2005. Disponível em <www.cinted.ufrgs.br/renote/maio2005/>. Acesso em abr. 2007.

CARNEIRO, V. C. G. **Engenharia Didática: um referencial para ação investigativa e para formação de professores de matemática** Zetetiké – Cempem – FE – Unicamp – v.13 n.23, 2005

COLOMBO, J. A. A.; FLORES, C. R.; MORETTI, M. T. **Registros de representação semiótica nas pesquisas brasileiras em Educação Matemática: pontuando tendências.** Zetetiké – Cempem – Unicamp – v.16 – n. 29. 2008.

D'AMBROSIO, U. **Prefácio da obra**. In: BORBA, M. C. e ARAÚJO, J. L. (orgs.). **Pesquisa Qualitativa em Educação Matemática** – Coleção Tendências em Educação Matemática. Belo Horizonte, MG. Autêntica,2004.

DELGADO, C.; XEXEO, J. A. M.; SOUZA, I. F.; RAPKIEWICZ, C. E.; PEREIRA JÚNIOR, J. C. **Identificando competências associadas ao aprendizado de leitura e construção de algoritmos**. XXV Congresso da Sociedade Brasileira de Computação. jul.2005.
Disponível em <http://www.unisinos.br/congresso/sbc2005/?sessao=wei>. Acesso em ago. 2007.

DELGADO, C.; XEXEO, J. A. M.; SOUZA, I. F.; CAMPOS, M., RAPKIEWICZ, C. E. **Uma abordagem pedagógica para a iniciação ao estudo de algoritmos**. XXIV Congresso da Sociedade Brasileira de Computação, 2004.
Disponível em <http://www.sbc2004.ufba.br>. Acesso em ago. 2007.

DUVAL, R. **Registros de representações semióticas e funcionamento cognitivo da compreensão em matemática.** In: MACHADO, S. D. A. (org.). **Aprendizagem em matemática: registros de representação semiótica.** Campinas, SP: Papirus,2008.

GARCIA, I. C.; REZENDE P. J.; CALHEIROS, F. C. **Astral: um ambiente para ensino de estruturas de dados através de animações de algoritmos.** In: **Congresso Iberoamericano de Educação Superior em Computação**, 5., 1996, México. Disponível em <www.ic.unicamp.br/~rezende/garcia.htm>. Acesso em jan. 2007.

LAAKSO, M.; MALMI, L.; KORHONEN, A.; RAJALA, T.; KAILA, E.; SALAKOSKI, T. **Using Roles of Variables to Enhance Novice's Debugging Work.** Issues in Informing Science and Information Technology, 2008

LAVILLE, C.; DIONNE, J. **A construção do saber** . Porto Alegre: Artmed, 1999.

MACHADO, N. J. **Sobre a idéia de competência.** In: Perrenoud, P. et al. **As competências para ensinar no século XXI: a formação dos professores e o desafio da avaliação.** Porto Alegre: Artmed Editora, 2002

MACHADO, S. D. A. **Engenharia Didática.** In: MACHADO, S. D. A. (org.). **Educação Matemática: uma (nova) introdução.** São Paulo: EDUC, 2008.

MACHADO, S. D. A. (org). **Aprendizagem em Matemática: Registros de representação semiótica**. Campinas, SP: Papirus, 2008a.

MACHADO, S. D. A. (org) . **Educação matemática: Uma (nova) introdução**. São Paulo: Educ, 2008b.

MANBER, U. **Introduction to Algorithms: a creative approach**. Addison-Wesley, 1989.

MARANHÃO, M. C. S. A. **Dialética ferramenta-objeto.** In: MACHADO, S. D. A. (org.). **Educação matemática: uma (nova) introdução.** São Paulo: EDUC, 2008.

MARTINS, C. T. K.; RODRIGUES, M. **Estudo de Algoritmos – Soluções em C++.** São Paulo: Edição do Autor, 2008

NICOLAU, M. **A ideografia global dos aplicativos de computador: uma *linguagem funcional* que transcende culturas no ciberespaço**.
Revista do Programa de Pós-Graduação em Comunicação da Universidade Federal da Paraíba, v. 2 n.1 – jun.2009. Disponível em <www.cchla.ufpb.br/culturasmidiaticas/pdf/02/artigo_ideografia_nicolau.pdf.
Acesso em nov. 2009.

PEREIRA JÚNIOR, J. C. R.; RAPKIEWICZ, C. E. **Um Ambiente Virtual para apoio a uma Metodologia para Ensino de Algoritmos e Programação**. Revista Novas Tecnologias na Educação, Cinted-UFRGS v.3 n.2, nov.2005. Disponível em <www.cinted.ufrgs.br/renote/nov2005/ >. Acesso em abr. 2007.

PERRENOUD, P. **Construir as competências desde a escola.** Porto Alegre: Artmed Editora, 1999

PERRENOUD, P. **A formação dos professores no século XXI.** In: Perrenoud, P. et al. **As competências para ensinar no século XXI: a formação dos professores e o desafio da avaliação.** Porto Alegre: Artmed Editora, 2002

PIMENTEL, E. P.; OMAR, O. **Ensino de Algoritmos baseado na Aprendizagem Significativa utilizando o Ambiente de Avaliação NetEdu**. Anais do XXVIII Congresso da Sociedade Brasileira de Computação (Workshop sobre Educação em Computação), 2008. Disponível em <www.prodepa.gov.br/sbc2008/anais/pdf/arq0120.pdf>. Acesso em abr. 2009.

SAJANIEMI, J.; KUITTINEN, M. **Program Animation Based on the Roles of Variables.** Symposium on Software Visualization – Association for Computing Machinery 2003

SAJANIEMI, J. **Roles of Variables and Learning to Program** Proceedings of the 3rd Panhellenic Conference “Didactic of Informatics” - 2005
Disponível em <http://www.joensuu.fi/~saja/var_roles/abstracts/didinf05.html> Acesso em set. 2009.

SALVETTI, D. D.; BARBOSA, L. M. **Algoritmos** São Paulo: Makron, 1998

SOCIEDADE BRASILEIRA DE COMPUTAÇÃO. **Currículo de referência da SBC para cursos de graduação em Computação e Informática. – versão 2003.** Porto Alegre: SBC, 2003.
Disponível em <www.sbc.org.br/index.php?language=1&subject=28>. Acesso em abr. 2009.

ZAMBONI, L. C.; PAMBOUKIAN, S. V. D.; BARROS, E. A. R. **C++ para universitários**. São Paulo: Páginas e Letras, 2006.

APÊNDICE A -

RESPOSTAS DOS ALUNOS – PRIMEIRA EXPERIMENTAÇÃO

PRIMEIRA ATIVIDADE

Antes de ter conhecimento sobre as orientações dessa disciplina, quais assuntos você esperava que fossem tratados?	
Grupo 1	Esperávamos que as aulas tratassem somente de lógica de programação e desenvolvimento de algoritmo, mas percebemos que a aula também trata de raciocínio lógico como resolução de problemas.
Grupo 2	Tínhamos consciência que o estudo proposto seria com o intuito de avaliação do aluno, porém não conhecíamos o foco da pesquisa.
Grupo 3	Sem resposta.

Há algum outro assunto que você considera que seria importante ser tratado nessa disciplina?	
Grupo 1	Não.
Grupo 2	A base de disciplinas voltadas à área de exatas, pois nem todos tiveram oportunidade, nem todos aproveitaram as oportunidades que tiveram e nem todos têm consciência da necessidade.
Grupo 3	Sem resposta.

Tendo em vista os objetivos especificados, você julga necessário o acréscimo de outras capacidades ou habilidades entre aquelas já colocadas como focos dos objetivos?	
Grupo 1	A gente tinha que ter assuntos focados em problemas do cotidiano. Poderíamos desenvolver projetos oferecendo soluções para empresas.
Grupo 2	Sem acréscimo de outras capacidades.
Grupo 3	Sem resposta.

Antes de colocar em execução o aplicativo, procure responder: se o lote possuir 5438 lâmpadas, qual será o tempo necessário para completar-se a operação? O seu resultado coincide com a resposta emitida pela execução do algoritmo?

Grupo 1	5438 | 15 — 8 362 → 8 x 4 = 32 segundos 362 | 60 — 2 6 (minutos, horas) → O resultado coincide
Grupo 2	Não os segundos não coincidiram
Grupo 3	1 l 5438 l 4 s x H min seg X = 6 : 02 : 32 5438 x 4 = 21752 21752 | 60 — 32 362 | 60 — 2 6

Como pode ser descrita a finalidade da instrução `qhoras←totmin div 60` disposta no algoritmo?

Grupo 1	Cada hora tem 60 minutos, logo a instrução está calculando a quantidade de horas
Grupo 2	É a conversão de minutos para hora.
Grupo 3	Extrai a quantidade inteira de horas de uma quantidade arbitrária de minutos A variável recebe o valor correspondente a divisão inteira da expressão

Se as instruções `totmin←qlamp div 15` e `resto←qlamp mod 15` tiverem suas disposições invertidas, o algoritmo permanecerá correto? Justifique.

Grupo 1	Não pois são instruções para diferentes cálculos, um não depende do outro.
Grupo 2	Sim, pois o cálculo do resto é indeferente podendo inverter sua posição e resultado continuará o mesmo.
Grupo 3	Sim, pois elas são independentes. O conjunto de uma não afeta a outra.

E se forem invertidas as instruções `resto←qlamp mod 15` e `qseg←resto*4` ? Justifique.	
Grupo 1	Sim, pois utilizamos o valor obtido na variável resto para executar a próxima instrução.
Grupo 2	Não, pois não existirá valor alocado na variável resto.
Grupo 3	Não, pois elas são dependentes. O conjunto de resto←qlamp mod 15 será utilizado em qseg←resto*4. A inversão altera o algoritmo.

Descreva a escolha de elementos de referência que você utilizaria para confirmar a pesagem com a colocação dos elementos de referência apenas sobre o prato vazio da balança. Há outras possibilidades de escolha? Há alguma delas que possa ser classificada como a mais adequada?	
Grupo 1	A mais adequada é a que usa a menor quantidade possível de pesos. 3 100g 2 30g 5 5g 2 1g
Grupo 2	Forma adequada 3 100g 2 30g 5 5g 2 1g ou 3 100g 17 5g 2 1g
Grupo 3	Qualquer combinação, desde que não tenha limite de elementos de referência. Na opinião do grupo a mais adequada é a que utiliza a menor quantidade de elementos. 3 100g 2 30g 5 5g 2 1g

Se em vez de 387g fossem 448g de farinha, como deveria ser a escolha dos elementos de referência?	
Grupo 1	4 100g 1 30g 3 5g 3 1g
Grupo 2	Forma adequada 4 100g 1 30g 3 5g 3 1g ou 4 100g 9 5g 3 1g
Grupo 3	4 100g 1 30g 3 5g 3 1g

Agora procure descrever as relações aritméticas entre a quantidade de farinha e as quantidades de cada tipo de elemento de referência.	
Grupo 1	qtd : 100 = resto : 30 = resto : 15 = resto : 1=
Grupo 2	3x1+3x5+1x30+4x100=448g Faço o processo inverso e multiplico ao invés de dividir.
Grupo 3	q100←(peso div 100) q30←(peso mod 100) div 30 q5←((peso mod 100) mod 30) div 5 q1←(((peso mod 100) mod 30) mod 5)

Considerando-se que você tem em mãos uma calculadora simples (quatro operações) e uma caneta e uma folha de papel para anotações; procure descrever detalhadamente a seqüência de operações que você deve realizar, com esses recursos (calculadora, caneta e papel), para determinar as quantidades de cada tipo de elemento de referência para uma quantidade genérica **qf** de farinha.	
Grupo 1	Calculadora: dividiria a qtd de farinha por 100 e acharia o resto multiplicando o valor que apareceria depois da vírgula por 100. Faria isso novamente por 30, depois por 5 e por último por 1. Papel e caneta: faria a divisão da quantidade de farinha por 100, usaria o resto para fazer a próxima divisão por 30, usaria o resto para dividir por 5, usaria o resto para dividir por 1.
Grupo 2	Dividimos a quantidade de farinha pelo maior peso pré determinado achando assim seu múltiplo comum, e esta seqüência continua com o resto até o último algarismo correspondente ser calculado com os pesos disponíveis.
Grupo 3	Obs: considerando que na questão 3 chegamos a uma solução genérica, basta aplicá-la para resolver estas questões.

Com a mesma finalidade (determinar as quantidades de cada tipo de elemento de referência para uma quantidade genérica **qf** de farinha), agora procure descrever uma seqüência de operações imaginando que a calculadora opere apenas com valores inteiros (adição, subtração, multiplicação e divisão com cálculo de quociente e cálculo de resto).	
Grupo 1	Sem resposta
Grupo 2	Como foi usado todos os elementos disponíveis a detalhação fica da seguinte maneira: Divide 387 por 100 O resto da 1 divisão por 30 O resto da 2 divisão por 5 O resto da 3 divisão por 1
Grupo 3	Obs: considerando que na questão 3 chegamos a uma solução genérica, basta aplicá-la para resolver estas questões.

Faça a descrição do algoritmo (utilize as formas de instruções já apresentadas) correspondente à seqüência de operações obtida na questão anterior.

Grupo 1

```
farinha()
leia(qtd);
s100←qtd div 100;
resto←qtd mod 100;
s30←resto div 30;
resto←resto mod 30;
s5←resto div 5;
s1←resto mod 5;
imprima(s100);
imprima(s100);
imprima(s100);
imprima(s100);
```

Grupo 2

Objetivo: determinar a quantidade dos pessos utilizados para provar a quantidade de farinha.

Entrada: qf (inteiro) Saída: qtp1, qtp2, qtp3, qtp4 (inteiro);

```
pesos( )
leia(qf);
qtp1←qf div 100;
resto←qf mod 100;
qtp2←resto div 30;
resto←resto mod 30;
qtp3←resto div 5;
resto←resto mod 5;
qtp4←resto/1;
imprima(qtp1);
imprima(qtp2);
imprima(qtp3);
imprima(qtp4);
```

Grupo 3

```
peso( )
leia(peso_total);
q100←peso_total div 100;
resto←peso_total mod 100;
q30←resto div 30;
resto←resto mod 30;
q5←resto div 5;
q1←resto mod 5;
imprima(q100, q30, q5, q1);
```

Indique o significado, diante da proposta do problema, de cada variável que você empregou no algoritmo.	
Grupo 1	qtd = quantidade de farinha (inteiro – int) s100 = massa de referência de 100g (inteiro – int) s30 = massa de referência de 30g (inteiro – int) s5 = massa de referência de 5g (inteiro – int) s1 = massa de referência de 1g (inteiro – int) resto = variável temporária (inteiro – int)
Grupo 2	qf → quantidade de farinha. qtp1 ao qtp4 → armazena a quantidade de elementos. resto → responsável pela divisão dos próximos pesos.
Grupo 3	peso_total → variável de entrada. q100 → variável de atribuição (armazena o valor correspondente a divisibilidade inteira por 100) q30 → variável de atribuição (armazena o valor correspondente a divisão inteira por 30) q5 → variável de atribuição (armazena o valor correspondente a divisão inteira por 5) q1 → variável de atribuição (armazena o valor correspondente ao resto da divisão inteira por 5) resto → variável de atribuição (armazena o valor correspondente ao resto da divisão das expressões)

Faça a implementação e testes do programa correspondente ao algoritmo que você construiu e depois transcreva, abaixo, o texto do programa obtido.	
Grupo 1	O grupo construiu o programa mas não transcreveu

Grupo 2	int main( )
	{
	int qf, qtp1, qtp2, qtp3, qtp4, resto;
	count<<"digite a quantidade “;
	cin>>qf;
	qtp1=qf/100;
	resto=qf%100;
	qtp2=resto/30;
	resto=resto%30;
	qtp3=resto/5;
	resto=resto%5;
	qtp4=resto/1;
	cout<<"elemento 100g"<<qtp1<<endl;
	cout<<"elemento 30g"<<qtp2<<endl;
	cout<<"elemento 5g"<<qtp3<<endl;
	cout<<"elemento 1g"<<qtp4<<endl;
	system(“PAUSE”);
	return(0);
	}

Grupo 3	O grupo construiu o programa mas não transcreveu

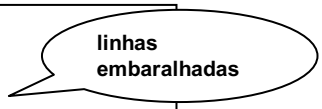
Se a porção de farinha possuir apenas 58g, o resultado obtido a partir da execução do seu programa é válida?	
Grupo 1	Sem resposta
Grupo 2	Sim.
Grupo 3	Sim.

Observe a seqüência de linhas que constituem o seu programa e procure responder: a única ordem de disposição de linhas correta é essa que figura no seu programa ou é possível alguma inversão? Se for possível alguma inversão, indique algumas possibilidades.	
Grupo 1	Sem resposta
Grupo 2	Não é possível inversão das linhas a não ser a disposição dos resultados.
Grupo 3	Inversões: resto←resto mod 30; q30←resto div 30; e q1←resto mod 5; q5←resto div 5;

Tarefa: as instruções descritas a seguir devem compor um programa que represente um método de resolução do problema mas algumas estão dispostas fora de ordem, faça a organização dessas linhas de maneira a obter um programa adequado.

```
#include <iostream>
using namespace std;

int main( ){
  int totpontos, goleadas, vitsimples, empates, quantjogos;
  cout<<"digite a quantidade desejada de pontos: ";
  cin>>totpontos;
    cout<<vitsimples<<" vitoria(s) simples, ";
    quantjogos=goleadas+vitsimples+empates;
    totpontos=totpontos%6;
    vitsimples=totpontos/3;
    cout<<goleadas<<" goleada(s), ";
    goleadas=totpontos/6;
    cout<<"quantidade minima de jogos: "<<quantjogos<<endl;
    cout<<empates<<" empate(s)."<<endl;
    empates=totpontos%3;
  system("PAUSE");
  return(0);
}
```



Transcreva, abaixo, a seqüência reorganizada das linhas.

Grupo 1	Sem resposta
Grupo 2	Sem resposta

Grupo 3	goleadas=totpontos/6;
	totpontos=totpontos%6;
	vitsimples=totpontos/3;
	empates=totpontos%3;
	quantjogos=goleadas+vitsimples+empates;
	cout<<goleadas<<” goleada(s), “;
	cout<<vitsimples<<” vitoria(s) simples, “;
	cout<<empates<<” empate(s).”<<endl;
	cout<<”quantidade mínima de jogos: “<<quantjogos<<endl;

SEGUNDA ATIVIDADE

No âmbito de um algoritmo ou programa, como pode ser descrito o papel de uma variável?	
Grupo 1	O papel de uma variável em um programa ou algoritmo é armazenar dados a serem manipulados.
Grupo 2	O papel da variável é o armazenamento de uma informação ou valor.

Que ações do sistema computacional podem ser relacionadas à execução do seguinte comando de atribuição: **`valor ← valor+1`** ?	
Grupo 1	Pega o valor original da variável, soma 1, depois atribui de volta para a variável o novo valor.
Grupo 2	O sistema atribui à variável valor, a quantidade armazenada anteriormente nessa mesma variável mais 1.

Execute o aplicativo para responder: se a siderúrgica encomendar 12,5t de carvão, qual deverá ser a quantidade extraída da mina?	
Grupo 1	Quantidade extraída: 12,8187 toneladas.
Grupo 2	Kkg 12,8187.

Determine a quantidade de carvão que será entregue à siderúrgica se o operador da mina extrair e carregar o trem com 1,5t de carvão. Para responder a esta questão faça algumas execuções do aplicativo para buscar uma aproximação da quantidade que será entregue, por exemplo: execute o aplicativo colocando 1,45t como quantidade a ser entregue na siderúrgica e depois procure melhorar a aproximação com quantidades a entregar mais adequadas; anote suas tentativas.	
Grupo 1	1º. 1.4 2º. 1.47 3º. 1.46 4º. 1.463 5º. 1.462 6º. 1.4625 7º. 1.4626 8º. 1.46265 9º. 1.46266 Quantidade entregue 1.46266
Grupo 2	Para obter o vagão do trem com 1,5 t é necessário extrair 1,4627 t da mina. Quantidade de tentativas: 1,4 1,5 1,45 1,463 1,46231 1,4629 1,4628 1,4627

As linhas 2 e 3 do algoritmo podem ter suas disposições invertidas? O algoritmo permanecerá correto? Justifique.	
Grupo 1	Não, permanecerá correto pois uma linha depende da outra.
Grupo 2	Não! Pois a variável “qporto” armazena um valor já calcula, que se torna necessário para o cálculo da variável “qextraida”.

Qual seria o valor integral do IPVA para um veículo cujo valor da nota de compra/venda seja de R$30000,00? Observe que se o mês de licenciamento do veículo for janeiro (mês 1), o valor a ser recolhido será igual ao valor integral.	
Grupo 1	Valor de R$ 1.200,00.
Grupo 2	R$ 1.200,00

Descreva uma expressão matemática que relacione o valor da nota de compra/venda com o valor integral do IPVA.
IPVA integral = ???

Grupo 1	IPVA integral = valornota*0,04.
Grupo 2	IPVA integral = valor * 0,04 * 12/12

Que fração do valor integral do IPVA deve corresponder ao valor a ser recolhido se o licenciamento do veículo novo for efetivado no mês de julho (mês 7) ? E se o licenciamento ocorrer em dezembro?

Grupo 1	Mês 7 = 6/12 Mês 12 = 1/12.
Grupo 2	7 → 6/12 → 600,00 12 → 1/12 → 100,00.

Descreva uma expressão matemática que relacione o valor integral do IPVA e o número do mês do licenciamento com o valor do IPVA a ser recolhido.
IPVA recolhido = ???

Grupo 1	IPVA recolhido = (12 – mês + 1)/12 . IPVA Integral.
Grupo 2	IPVA recolhido = valor * 0,04 * (13 – mês)/12

Faça a descrição de um algoritmo que represente um método de resolução do problema proposto (articule de forma adequada as relações matemáticas obtidas nas questões anteriores). Escolha identificadores de variáveis que indiquem os respectivos significados.

Grupo 1

```
leia(vnota); leia(mescomp);
ipvaI← vnota*0.04;
ipvaR← ((13-mescomp)/12)*ipvaI;
imprima(ipvaI);
imprima(ipvaR);
```

Grupo 2

```
IPVA( )
leia(mes); leia(vcarro);
ipva← vcarro*0,04*(13-mes)/12;
imprima(ipva);
```

Faça a implementação e testes do programa correspondente ao algoritmo que você construiu. Transcreva na área abaixo o texto de seu programa.

Grupo 1

```
int main(int argc, char *argv[])
{
  float vnota, mescomp, ipvaI, ipvaR;
  cout<<"digite valor da nota:";
  cin>>vnota;
  cout<<"digite o mês da compra:;
  cin>>mescomp;
  ipvaI=vnota*0.04;
  ipvaR=((13-mescomp)/12)*ipvaI;
  cout<<"valor ipva integral "<<ipvaI<<endl;
  cout<<"valor ipva recolhido "<<ipvaR<<endl;
  system("PAUSE");
  return(0);
}
```

Grupo 2

```
int main( )
{
  float vcarro, ipva;
  int mes;
  cout<<" valor do carro"<<endl;
  cin>>vcarro;
  cout<<"mês"<<endl;
  cin>>mes;
  ipva=vcarro*0.04*((13-mes)/12);
  cout<<"valor a ser pago é"<<ipva<<endl;
  system("PAUSE");
  return(0);
}
```

Observe a seqüência de linhas que constituem o seu programa e procure responder: a única ordem de disposição de linhas correta é essa que figura no seu programa ou é possível alguma inversão? Se for possível alguma inversão, indique alguma possibilidade.

Grupo 1	É possível a inversão nas linhas 4 e 5 por 6 e 7 e nas 10 por 11.
Grupo 2	Não.

TERCEIRA ATIVIDADE

Qual a finalidade das estruturas de controle de seleção em um algoritmo ou programa?

Grupo 1	Análise das informações, para a validação de uma condição.

Observe a seqüência de instruções abaixo:

```
...
1.  float x;
2.  cout<<"valor de x? ";
3.  cin>>x;
4.  x=3*x;
5.  if(x<7.2){
6.     x=x+10;
7.     x=1.5*x;
8.  }
9.  else{
10.    x=2.5*x;
11.    x=x+10;
12. }
13. cout<<"novo x: "<<x<<endl;
...
```

Quais linhas de instruções serão executadas se o valor fornecido como conteúdo inicial da variável **x** for 4.4? Nesse caso, qual será o valor armazenado como conteúdo dessa variável ao final da execução da seqüência de instruções?

Grupo 1	1 ao 4 e 9 até 13	43

E se o conteúdo inicial de **x** for 1.8?

Grupo 1	1 ao 4 e 5 ao 7 e 13	23.1

Quais das seqüências abaixo são equivalentes àquela descrita acima?

seqüência 1

```
...
 float x;
 cout<<"valor de x? ";
 cin>>x;
 x=3*x;
 if(x>=7.2){
    x=2.5*x;
    x=x+10;
 }
 else{
    x=x+10;
    x=1.5*x;
 }
 cout<<"novo x: "<<x<<endl;
...
```

seqüência 2

```
...
 float x;
 cout<<"valor de x? ";
 cin>>x;
 x=3*x;
 if(x>=7.2){
    x=2.5*x;
    x=x+10;
 }
 if(x<7.2){
    x=x+10;
    x=1.5*x;
 }
 cout<<"novo x: "<<x<<endl;
...
```

seqüência 3

```
...
 float x;
 cout<<"valor de x? ";
 cin>>x;
 x=3*x;
 if(x<7.2){
    x=x+10;
    x=1.5*x;
 }
 x=2.5*x;
 x=x+10;
 cout<<"novo x: "<<x<<endl;
...
```

Grupo 1	Nenhuma das 3.

Se as duas medidas conhecidas forem 30^o e 50^o, qual será a medida do terceiro ângulo?

Grupo 1	100

Descreva quatro casos (quatro duplas de valores inteiros) para os quais a resposta produzida pelo algoritmo seja 70º. Experimente os pares de valores escolhidos com a execução do aplicativo.	
Grupo 1	40 e 70 50 e 60

Descreva um caso em que as medidas sejam incompatíveis com a existência do triângulo. Experimente.	
Grupo 1	0 e 30 100 e 90

Como você descreve a finalidade da instrução colocada na linha 4 do algoritmo?	
Grupo 1	Tem como finalidade comparar os valores se são compatíveis para formar um triângulo.

Para um percurso total de 13050km a porcentagem que definirá o prêmio será 11,1%. Explique o processo aplicado para o cálculo dessa taxa de porcentagem.	
Grupo 1	Se o valor percorrido for menor 12000 o motorista vai receber apenas 5,5%. Subtrai os 12000 dos 13050, essa subtração gerou um resto. Esse resto é dividido por 600 (DIV inteiro) e o resto da divisão soma-se 1 ao quociente * 2,8 +5,5.

Descreva uma expressão matemática que indique a forma de cálculo do valor do prêmio a partir da taxa (%) e do valor do salário bruto. Suponha que a taxa já tenha sido calculada.	
Grupo 1	Premio = SB * (1 – taxa)

Organize um conjunto de expressões matemáticas para traduzir a relação entre a quilometragem total e a taxa de porcentagem que deve ser aplicada para o cálculo do valor do prêmio.	
Grupo 1	Sem registro de resposta.

Faça a descrição de um algoritmo que represente um método de resolução do problema proposto (articule de forma adequada as relações matemáticas e lógicas obtidas nas questões anteriores). Utilize identificadores de variáveis que indiquem o significado, diante da proposta do problema, de cada informação representada.

Grupo 1

```
leia(sb); leia(km);
se km>12000
   então   resto=km-12000;
           adicional←resto/600;
                se (resto mod 600)>0
                  então adicional←adicional+1;
           taxa←5,5+2,8*adicional;
   senão   taxa←5,5;
premio←(sb*taxa)/100;
exibir(premio);
```

Faça a implementação e testes do programa correspondente ao algoritmo que você construiu. Transcreva na área abaixo o texto de seu programa.

Grupo 1

(o texto do programa não foi registrado no caderno da atividade, o arquivo correspondente foi remetido por correio eletrônico por um dos componentes do grupo.)

```
int main( )
{  float sb, premio, taxa;
   int km, resto, adicional;
   cout<<"salário bruto?"; cin>>sb;
   cout<<"quilometragem?"; cin>>km;
   if(km>12000)
   {  resto=km-12000;
      adicional=resto/600;
      if(resto%600>0) adicional=adicional+1;
      taxa=5.5+(2.8*adicional);
   }
   else
   {  taxa=5.5;
   }
   premio=(sb*taxa)/100;
   cout<<"valor do premio: "<<premio<<endl;
   system("pause"); return(0);
}
```

QUARTA ATIVIDADE

Qual a finalidade das estruturas de controle de repetição em um algoritmo ou programa?	
Grupo 1	Permitem a definição de fluxos de processamento repetitivos (a resposta foi registrada nos rascunhos, mas não foi transcrita para o caderno).
Grupo 2	Executa repetição de um código.
Grupo 3	Repetir a linha de comando até que o resultado especificado na condição seja obtido ou se torne falso.

Observe a seqüência de instruções abaixo:

```
...
   1. int soma, parcela;
   2. soma=0; parcela=5;
   3. while(parcela<200){
   4.   soma=soma+parcela;
   5.   parcela=parcela*3;
   6. }
   7. cout<<"valor da parcela: "<<parcela<<endl;
   8. cout<<"valor da soma: "<<soma<<endl;
...
```

Quantas vezes será executada a <u>linha 2</u> dessa seqüência de instruções?

Grupo 1	3
Grupo 2	Executável uma vez.
Grupo 3	Apenas uma vez pois ela está fora da estrutura de repetição.

Quantas vezes será executada a <u>linha 4</u> dessa seqüência de instruções?

Grupo 1	3
Grupo 2	Executável 4 vezes.
Grupo 3	4 vezes pois na quinta vez a condição se tornará falsa.

Quantas vezes será avaliada a expressão lógica de controle **`parcela<200`**?	
Grupo 1	4
Grupo 2	5 vezes
Grupo 3	Quantas vezes forem necessárias até a condição se tornar falsa.

Que valores serão exibidos como resultados da execução?	
Grupo 1	Parcela = 135 Soma = 200
Grupo 2	Parcela valor 505. Soma valor 200.
Grupo 3	1º. P=15, S=5 2º. P=45, S=20 3º. P=135, S=65 4º. P=405, S=200

Descreva como será a modificação do comportamento do programa se a condição **`parcela<200`** for alterada para **`parcela<10`** .	
Grupo 1	Não vai ser executado nenhuma vez se parcela<10.
Grupo 2	Só será executado o bloco do while uma vez.
Grupo 3	Ela terá a sua estrutura de repetição executada menos vezes.

Se a quantidade de fichas disponíveis inicialmente é 100, quantas apostas o jogador pode fazer? Calcule sem utilizar o aplicativo e depois confronte sua resposta com o resultado fornecido pela execução do programa.			
Grupo 1	2		
Grupo 2	Serão 2 apostas.		
Grupo 3	Jogada 1 100: 100-15 aposta: 15*2=30	Jogada 2 85: 85-30 aposta: 30*2=60	Jogada 3 não poderá ser feita pois o jogador não possível fichas disponíveis.

Qual seria a quantia mínima necessária se o desejo do apostador fosse realizar exatamente 6 apostas. Confronte seu resultado com a resposta gerada pelo aplicativo.

Grupo 1	945
Grupo 2	Quantidade mínima de R$945,00
Grupo 3	945

Que modificações seriam necessárias no algoritmo se em vez de dobrar a quantidade da aposta a cada vez, o jogador aumentasse em 50 fichas a quantidade da aposta anterior?

Grupo 1	faposta = faposta + 50
Grupo 2	Trocar: faposta←faposta+50;
Grupo 3	A linha de comando que diz que a aposta é dobrada, aposta=aposta*2, seria assim: aposta=aposta+50.

Complete o quadro, até a data 6, com as informações correspondentes:

Grupo 1

4	50	75	125
5	175	275	450
6	625	1000	1625

Grupo 2

4	50	75	125
5	175	275	450
6	625	1000	1625

Grupo 3

4	15+20+15 50	15+40+20 75	125
5	50+75+50 175	50+150+75 275	450
6	175+275+175 625	175+ 825	1450

Complete de forma adequada as lacunas dispostas nas afirmações abaixo com o emprego de algumas das expressões:

a quantidade
o dobro da quantidade
o triplo da quantidade
o quádruplo da quantidade

Grupo 1	**LACUNA 1 :** o dobro de txt **LACUNA 2 :** a quantidade de exe **LACUNA 3 :** triplo de exe **LACUNA 4 :** quantidade de txt
Grupo 2	**LACUNA 1 :** o dobro da quantidade **LACUNA 2 :** a quantidade **LACUNA 3 :** a quantidade **LACUNA 4 :** o triplo da quantidade
Grupo 3	**LACUNA 1 :** o dobro **LACUNA 2 :** a quantidade **LACUNA 3 :** triplo da quantidade **LACUNA 4 :** a quantidade

Se **txt** e **exe** representam as quantidades atuais de arquivos de texto e de arquivos executáveis contaminados e **txtant** e **exeant** as correspondentes quantidades na data anterior, complete as instruções de atribuição que estabelecem as relações entre essas quantidades:

Grupo 1	**txt**← 2 x txtant + exeant **exe**← 3 x exeant + txtant
Grupo 2	**txt**← 2 * txt_ant + exe_ant **exe**← txt_ant + 3 * exe_ant
Grupo 3	**txt**← txtant * 2 + exeant **exe**← exeant * 3 + txtant

Faça a descrição de um algoritmo que represente um método de resolução do problema proposto. Utilize identificadores de variáveis que indiquem o significado, diante da proposta do problema, de cada informação representada.

Grupo 1

```
Entrada: QTD inteiro;
Saída: data inteiro;
     Leia(qtd);
     Txt=1;
     Exe=0;
     Data=1;
     Soma=1;
     Enquanto Soma<=qtd) faça
       Data←Data+1;
       Txt←(Txt*2)+Exe;
       Exe←(Exe*3)+Txt;
       Txtabt←Txt;
       Soma←Exe+Txt;
       Imprima(Data);
```

Grupo 2

```
Dias superados( )
Leia(qsurp);
Txtant←1; exeant←0; dias←0;
Total←0;
Enquanto (total<=supr) faça
   txt←(2*txtant)+exeant;
   exe←txtant+(3*exeant);
   total←txt+exe;
   txtant←txt;
   exeant←exe;
   dias←dias+1;
imprima(dias);
```

Grupo 3

```
   Virus
leia(qtdsup);
data←0;
Txt←1;
exe←0;
qtdtotal←0;
  enquanto (qtdtotal<=qtdsup)
     txt←txt*2+exe;
     exe←exe*3+txt;
     qtdtotal←exe+txt;
     data←data+1;
  imprima(data);
```

Faça a implementação e testes do programa correspondente ao algoritmo que você construiu. Transcreva na área abaixo o texto de seu programa.

Grupo 1

```
   int qtd, txt, exe, soma, data, txtant, exeant;
   soma=1;
   txt=1;
   exe=0;
   data=0;
   cout<<"entre com um valor de quantidade:";
   cin>>qtd;
   while(soma<=qtd){
     data=data+1;
     txtant=txt;
     exeant=exe;
     txt=(txt*2)+exe;
     exe=(exe*3)+txtant;
     soma=exe+txt }
   cout<<"o valor da data é : "<<data<<endl;
   system("PAUSE");
   return(0);
}
```

Grupo 2

```
int qsupr, txt, txt_ant, exe, exe_ant, total, dias;
cout<<"Digite a quantidade superada:";
cin>>qsupr;
txt_ant=1; exe_ant=0; dias=0; total=0;
while(total<=qsupr){
  txt=(2*txt_ant)+exe_ant;
  exe=txt_ant+(3*exe_ant);
  total=txt+exe;
  txt_ant=txt;
  exe_ant=exe;
  dias=dias+1;
  data=data+1;
}
cout<<"Dias:"<<dias<<endl;
```

Grupo 3

```
int qtdtotal, qtdsup, txt, exe, data;
cout<<"Digite a quantidade a superar: ";
cin>>qtdsup;
data=0;
txt=1;
exe=0;
qtdtotal=0;
while(qtdtotal<=qdtsup){
  txt=txt*2+exe;
  exe=exe*3+txt;
  qtdtotal=exe+txt;
  data=data+1;
}
cout<<"Quantidade de dias:"<<data<<endl;
```

APÊNDICE B -

RESPOSTAS DOS ALUNOS – SEGUNDA EXPERIMENTAÇÃO

PRIMEIRA ATIVIDADE

questão 1 Antes de colocar em execução o aplicativo, procure responder: se o lote possuir 5438 lâmpadas, qual será o tempo necessário para completar-se a operação? O seu resultado coincide com a resposta emitida pela execução do algoritmo?	
Grupo 1	Dados: 15 lâmpadas por minuto 5438 → 362 minutos ou 6hs, 2 min, 32 segundos Sim, coincide com o resultado algoritmo.
Grupo 2	Sim, o nosso resultado foi: → 6: 02: 32
Grupo 3	06: 02: 32 Resposta idêntica ao softwere
Grupo 4	6h 2 min 32 seg Sim.
Grupo 5	5438/15 = 362,53 resto 0 demorará 6 horas 3 minutos e 32 segundos 06: 02: 32 O resultado coincide com o algoritmo.
Grupo 6	15 lâmpadas/minuto 5438/15 = 362 min resto 8 lâmpadas 362/60 = 6horas resto 2 min 15l --- 60s 8l --- x x = (60x8)/15 = 32 segundos 6 horas, 2 minutos e 32 segundos O resultado coincide com a resposta do algoritmo.
Grupo 7	5438 (lote de lâmpada)/15 (quantidade de minutos) = 362 (total de minutos) resto 8 (sobra de lâmpadas que leva menos de 1 minuto p/ embalar) 60s → 15 lamp x → 8 lamp x + tot seg = (60x8)/15 = 32 seg total horas = 362/60 = 6 horas total min = 362 mod 60 = 2m portanto o resultado é 6 horas, 2 minutos e 32 segundos coincidindo com o resultado do programa executado.
Grupo 8	5438/15 = 362 resto 8 362/60 = 6 resto 2 8x4 = 32 R: 6 horas, 2 minutos e 32 segundos
Grupo 9	As horas e os minutos coincidem, porém, os segundos exatos não. 5438/15 = 362,5333... 362,53/60 = 6h resto 2,53 2 min 0,53.60 = 31,8s Resultado: 6h 2min 31,8s Resultado obtido pelo sistema: 6h 2min 32s

Grupo 10	15 lâmpadas = 1minuto = 60 segundos então 1 lâmpada = 4 segundos 5438 lâmpadas x 4 segundos = 21752 segundos horas → 21752/3600 = 6 horas resto 152 minutos →152/60 = 2 minutos resto 32 segundos O resultado coincide.

questão 2 Como pode ser descrita a finalidade da instrução `qhoras←totmin div 60` disposta no algoritmo?	
Grupo 1	totalminutos é dividido por 60, para transformar em horas. Neste caso, qhoras está recebendo o valor da divisão totmin/60.
Grupo 2	A variável “qhoras” recebe o resultado da divisão da variável “totmin” por 60.
Grupo 3	qhoras ← totalmin div 60 qhoras recebe totalmin divido 60, para apresentar o valor em horas.
Grupo 4	Descobrir a quantidade de horas.
Grupo 5	A quantidade de horas é igual ao total de minutos dividido por 60.
Grupo 6	A instrução calcula o total de horas exatas (valor inteiro) baseado no total do tempo em minutos.
Grupo 7	Calcular a quantidade de horas do processo.
Grupo 8	A finalidade dessa instrução é atribuir a quantidade de horas a variável qhoras.
Grupo 9	É a instrução para encontrar a quantidade de horas totais.
Grupo 10	Ela divide o total encontrado de minutos por 60 e armazena na variável “qhoras”, para encontrar o valor em horas.

questão 3 Se as instruções `totmin←qlamp div 15` e `resto←qlamp mod 15` tiverem suas disposições invertidas, o algoritmo permanecerá correto? Justifique.	
Grupo 1	Sim, o algoritmo permanecerá o mesmo, porque o resultado da divisão qlamp/15 será atribuído a "totmin" e na próxima instrução o resto da divisão qlamp/15 será atribuído a variável "resto". Concluindo: estas duas instruções não irá alterar o resultado do algoritmo se as disposições forem invertidas.
Grupo 2	Sim, pois são independentes, isto é, utiliza-se uma mesma variável para realizar as operações de divisão inteira sem que essa variável seja modificada.
Grupo 3	Sim, pois um não depende do outro.
Grupo 4	Sim, pois uma instrução é independente da outra.
Grupo 5	Sim, pois uma é independente da outra.
Grupo 6	Não, porque uma não depende da outra. Ambas dependem, apenas, do número de lâmpadas.
Grupo 7	Sim, pois a 1ª instrução recolhe o quociente inteiro da divisão, enquanto a segunda instrução recolhe o resto da mesma divisão, ou seja, a ordem do recebimento não altera o processo do algoritmo.
Grupo 8	Sim, permanecerá correto, pois, são variáveis independentes (uma não depende da outra)
Grupo 9	Permanecerá correto pois as variáveis resultantes destas instruções são independentes.
Grupo 10	Sim, pois as variáveis que recebem as informações não dependem umas das outras para resolver as expressões e armazená-las.

questão 4 E se forem invertidas as instruções **`resto←qlamp mod 15`** e **`qseg←resto*4`** ? Justifique.	
Grupo 1	Estas instruções não poderão ser invertidas, pois para calcular “qseg” dependemos do valor de “resto”.
Grupo 2	Não, porque se qseg←resto*4 for colocado antes da outra instrução o resto não terá valor atribuído.
Grupo 3	Não, pois qseg depende do resultado de resto.
Grupo 4	Ficará incorreto, pois é necessário primeiro calcular o resto.
Grupo 5	Não ficará correto, pois o resto deve ser calculado antes que a quantidade de segundos.
Grupo 6	Sim, porque para o cálculo da quantidade de segundos é necessário o valor do resto.
Grupo 7	Neste caso não pois a atribuição quantidade de segundos depende do resultado obtido na primeira instrução.
Grupo 8	Não, pois, a variável resto será atribuída a outra variável.
Grupo 9	Ficaria incorreto, pois a variável qseg depende do valor atribuído a variável resto. Portanto, deve-se atribuir um valor a variável antes que ela possa ser utilizada.
Grupo 10	Não, pois a variável “qseg” está recebendo o valor de “resto*4”, e “resto” foi “descoberto” a partir da expressão “qlamp mod 15”, então se resto não houvesse sido descoberto a primeira expressão (resto*4) não faria sentido e o programa não iria funcionar corretamente.

questão 5 Descreva a escolha de elementos de referência que você utilizaria para confirmar a pesagem com a colocação dos elementos de referência apenas sobre o prato vazio da balança. Há outras possibilidades de escolha? Há alguma delas que possa ser classificada como a mais adequada? Observação: no disco (CD) há um aplicativo (**massa_farinha**) que pode ser utilizado para confirmar ou rejeitar suas respostas.	
Grupo 1	387/100 = 3 resto 87 ; 87/30 = 2 resto 27 ; 27/5 = 5 resto 2 ; 2/1 = 2 resto 0 Pesos a serem utilizados: 3x100g + 2x30g + 5x5g + 2x1g Existem outras possibilidades, porém como trabalhamos com quociente e resto, em algoritmo é mais fácil começar montar as combinações maiores indo para as menores.
Grupo 2	3 → 100g → Sim, diversão 2 → 30g 5 → 5g → Sim, esse ao lado sim utiliza menos elementos de referência 2 → 1g
Grupo 3	100x3 ; 30x2 ; 5x5 ; 1x2 A melhor forma que o grupo encontrou foi essa, mas há outras inúmeras formas. A mais adequada foi a que o grupo encontrou.
Grupo 4	3 elementos de 100g 2 elementos de 30g 5 elementos de 5g 2 elementos de 1g Sim A que está descrita é a mais adequada, pois usa menos elemntos.
Grupo 5	3x100g 2x30g 3x5g 2x1g Existem outras possibilidades, porém essa escolha é a que mais otimizaria a pesagem.
Grupo 6	3 elementos de massa 100g 2 elementos de massa 30g 5 elementos de massa 5 g 2 elementos de massa 1g Sim, há outras possibilidades como: 77 elementos de 5g + 2 de 1g ou 387 elementos de 1g ... etc... Sim, o que exige menos elementos, por ser mais prático.
Grupo 7	3x100g = 300g 2x30g = 60g 5x5g = 25g 2x1g = 2g Sim, há outras possibilidades porém essa configuração é a que usa menos elementos, portanto é a mais adequada.
Grupo 8	Sim, há outras possibilidades. A forma mais adequada seria 3 elementos com 100g, 2 elementos de 30g, 5 elementos de 5g, 2 elementos de 1g.

Grupo 9	A escolha mais adequada é a que utiliza a menor quantidade de elementos de referência: 3 de 100g, 2 de 30g, 5 de 5g e 2 de 1g. Porém, há outras possibilidades.
Grupo 10	Passo I 3 elementos de 100g II 2 elementos de 30g III 5 elementos de 5g IV 2 elementos de 1g Sim, existem outras possibilidades como colocar elementos de 30g, 5g e 1g, porém a mais prática e que utiliza menos elementos é este modo descrito.

questão 6 Se em vez de 387g fossem 448g de farinha, como deveria ser a escolha dos elementos de referência?	
Grupo 1	448/100 = 4 resto 48 ; 48/30 = 1 resto 18 ; 18/5 = 3 resto 3 ; 3/1 = 3 resto 0 Pesos a serem utilizados: 4x100g + 1x30g + 3x5g + 3x1g
Grupo 2	4 → 100g 1 → 30g 3 → 5g 3 → 1g
Grupo 3	100x4 ; 1x30 ; 3x5 ; 3x1
Grupo 4	4 elementos de 100g 1 elemento de 30g 3 elementos de 5g 3 elementos de 1g
Grupo 5	4x100g 1x30g 3x5g 3x1g
Grupo 6	4 elementos de 100g 1 elemento de 30g 3 elementos de 5g 3 elementos de 1g
Grupo 7	4x100g = 400g 1x30g = 30g 3x5g = 15g 3x1g = 3g
Grupo 8	4 elementos de 100g, 1 elemento de 30g, 3 elementos de 5g, 3 elementos de 1g.
Grupo 9	4 de 100g, 1 de 30g, 3 de 5g e 3 de 1g.
Grupo 10	Existem várias maneiras, mas a que escolhemos foi: 4 elementos de 100g 1 elemento de 30g 3 elementos de 5g 3 elementos de 1g

questão 7 Agora procure descrever a seqüência de relações aritméticas (cálculos aritméticos) empregadas para obter as quantidades de cada tipo de elemento de referência a partir da quantidade de farinha.	
Grupo 1	Dividir a quant farin por 100 → quociente e resto quociente → pesos 100 resto dividir por 30 o quociente é pesos de 30 o resto dividir por 5 o quociente é pesos de 5 o resto dividir por 1 o quociente é pesos de 1
Grupo 2	peso100g ← farinha div 100 resto ← farinha mod 100 peso30g ← resto div 30 resto ← resto mod 30 peso5g ← resto div 5 resto ← resto mod 5 peso1g ←resto div 1
Grupo 3	Farinha/100g Resto/30 Resto/5
Grupo 4	É pego o valor total e dividido por 100, depois o resto é dividido por 30, o que sobrar é dividido por 5 e por fim o resto dessa divisão é dividido por 1.
Grupo 5	Divide-se a quantidade por 100, o resultado por 30, o resultado por 5 e o resto será de 1g.
Grupo 6	448/100 = 4 de 100g resto 48 48/30 = 1 de 30g resto 18 18/ 5 = 3 de 5 g resto 3 3 de 1g
Grupo 7	287 (qtde de farinha)/100 = 3 (qtde de elementos de 100g) resto 87 87/30 = 2 (qtde de elementos de 30g) resto 27 27/5 = 5 (qtde de elementos de 5 g) resto 2 2/1 = 2 (qtde de elementos de 1g)
Grupo 8	Elementos disponíveis: 100g 30g 5g 1g Dividir a quantidade desejada de farinha por 100, resto dessa divisão será dividido por 30, o resto será dividido por 5 e o resto dividido por 1.

Grupo 9	Divide-se a massa da farinha péla massa do maior elemento de referência e o resto da divisão pela massa do próximo maior elemento e assim sucessivamente até que não sobre resto. Para que o resultado seja exato todas as variáveis devem ser do tipo inteiro. Ex: massa farinha/100 resto/30 resto/5 resto/1
Grupo 10	Dividimos o total de massa pelos elementos informados. Passo-a-passo x = quantidade de farinha 100g = elemento de 100g 30g = elemento de 30g 5g = elemento de 5g 1g = elemento de 1g resto = sobra das divisões x/100 = 100g resto/30 = 30g resto/5 = 5g resto = 1 Exemplo: 387/100 = 3 elementos de 100g resto 87 87/30 = 2 elementos de 30g resto 27 27/5 = 5 elementos de 5g resto 2 2 elementos de 1g

questão 8

Faça a descrição de um algoritmo (utilize as formas de instruções já apresentadas) que represente um método de resolução do problema proposto. Escolha identificadores de variáveis que indiquem os significados das informações correspondentes.

Grupo 1

Assinatura
Objetivo: Determinar Quantidade de elementos
Entrada: QUANTF - INTEIRO
Saída: ELEME100, ELEME30, ELEME5, ELEME1 - INTEIRO
Corpo

```
Balança( )
      Leia(QUANTF);
      Ele100 ← QuantF div 30;
      Ele30 ← (QuantF mod 100) div 30;
      Ele5 ← (Ele30 mod 30) div 5;
      Ele1 ← (Ele5 mod 5) div 1;

      Imprime (Ele100);
      Imprime (Ele30);
      Imprime (Ele5);
      Imprime (Ele1);
```

Grupo 2

Assinatura
Objetivo: Determinar as quantidades dos elementos de referência para pesagem da farinha
Entrada: farinha - inteiro
Saída: peso100, peso30, peso5, peso1 - inteiro
Corpo

```
Farinha( )
      leia(farinha);
      peso100 ← farinha div 100;
      resto ← farinha mod 100;
      peso30 ← resto div 30;
      resto ← resto mod 30;
      peso5 ← resto div 5;
      resto ← resto mod 5;
      peso1 ← resto;
      imprima(peso100g; imprima(peso30g);
      imprima(peso5g); imprima(peso1g);
```

Grupo 3

Assinatura Massa de farinha
Objetivo: Confirmar a massa de uma porção de farinha
Entrada: massa – tipo inteiro
Saída: qtd100, qtd30, qtd5, qtd1 – tipo inteiro
Corpo

```
Farinha
      leia(massa)
      qtd100 ← massa div 100;
      resto ← massa mod 100;
      qtd30 ← resto div 30;
      resto1 ← resto mod 30;
      qtd5 ← resto1 div 5;
      qtd1 ← resto1 mod5;
      imprima (qtd100);
      imprima (qtd30);
      imprima (qtd5);
      imprima (qtd1);
```

Grupo 4

Assinatura
Objetivo: Determinar as quantidades necessárias de cada tipo de elementos de referência para realizar a confirmação
Entrada: Qndefar (int')
Saída: ele100, ele30, ele5, ele1
Corpo

```
Farinha ( )
      Leia (Qndfar);
      ele100 ← Qndfar div 100;
      ele30 ← (Qndfar mod 100) div 30;
      ele5 ← ((Qndfar mod 100) mod 30) div 5;
      ele1 ← ((Qndfar mod 100) mod 30) mod 5;
      imprima (ele100);
      imprima (ele30);
      imprima (ele5);
      imprima (ele1);
```

Grupo 5

Assinatura
Objetivo: Otimizar as massas para pesagem
Entrada: peso → tipo inteiro
Saída: qt100, qt30, qt5, qt1 → tipo inteiro
Corpo

```
Pesagem ( )
      leia (peso);
      qt100 ← peso div 100;
      calc1 ← peso mod 100;
      qt30 ← calc1 div 30;
      calc2 ← calc1 mod 30;
      qt5 ← calc2 div 5;
      qt1 ← calc2 mod 5;
      imprime (qt100);
      imprime (qt30);
      imprime (qt5);
      imprime (qt1);
```

<table>
<tr><td>Grupo 6</td><td>Assinatura
Objetivo: Obter a quantidade de elementos correspondente a massa da farinha.
Entrada: Massa da farinha - real
Saída: Elementos de 100g, de 30g, de 5g e de 1g - inteiro
Corpo
FARINHA( )
<pre>leia (farinha);
qtd100 = farinha DIV 100;
qtd30 = (farinha MOD 100) DIV 30;
qtd5 = ((farinha MOD 100) MOD 30) DIV 5;
qtd1 = ((farinha MOD 100) MOD 30) MOD 5;
imprima(qtd100);
imprima(qtd30);
imprima(qtd5);
imprima(qtd1);</pre></td></tr>
<tr><td>Grupo 7</td><td>Assinatura
Objetivo: Determinar a quantidade necessária de cada elemento de referência.
Entrada: pfarinha – tipo inteiro
Saída: g100; g30; g5; g1;
Corpo
Farinha ( )
<pre>Leia(pfarinha);
g100 ← pfarinha DIV 100;
gparcial ← pfarinha MOD 100;
g30 ← gparcial DIV 30;
gparcial ← gparcial MOD 30;
g5 ← gparcial DIV 5;
gparcial ← gparcial MOD 5;
g1 ← gparcial DIV 1;
imprima (g100);
imprima (g30);
imprima (g5);
imprima (g1);</pre></td></tr>
<tr><td>Grupo 8</td><td>Assinatura
Objetivo: Especificar a quantidade de cada elemento
Entrada: Quantidade total de farinha
Saída: Quantidade de cada elemento
Corpo
Farinha ( )
<pre>Leia (totfarinha);
Elemento1 ← totfarinha DIV 100;
Elemento2 ← (totfarinha MOD 100) DIV 30;
Elemento3 ← ((totfarinha MOD 100) MOd 30) DIV 5;
Elemento4 ← (((totfarinha MOD 100)MOD 30)MOD 5)DIV 1;
Imprima (Elemento1);
Imprima (Elemento2);
Imprima (Elemento3);
Imprima (Elemento4);</pre></td></tr>
<tr><td></td><td></td></tr>
</table>

Grupo 9	*Assinatura*
	Objetivo: Determinar a quantidade de cada tipo de elemento
	Entrada: quantimassa – tipo inteiro
	Saída: e100, e30, e5, e1 – tipo inteiro
	Corpo
	`Quantidade( )`
	`Leria (quantmassa);`
	`e100 ← quantimassa/100;`
	`resto ← quantmassa mod 100;`
	`e30 ← resto/30;`
	`resto ← resto mod 30;`
	`e5 ← resto/5;`
	`resto ← resto m od 5;`
	`e1 ← resto/1;`
	`imprima(e100);`
	`imprima(e30);`
	`imprima(e5);`
	`imprima(e1);`

Grupo 10	*Assinatura*
	Objetivo: Determinar as quantidades necessárias de cada tipo dos elementos
	Entrada: Massa de farinha (farinha)
	Saída: Elementos: (cemg), (trintag), (cincog) e (umg).
	Corpo
	`Balanca( )`
	`leia (farinha);`
	`cemg ← farinha DIV 100;`
	`resto ← farinha MOD 100;`
	`trintag ← resto DIV 30;`
	`cincog ← (resto MOD 30) DIV 5;`
	`umg ← resto MOD 5;`
	`imprima (cemg);`
	`imprima (trintag);`
	`imprima (cincog);`
	`imprima (umg);`

questão 9

Faça a implementação e testes do programa correspondente ao algoritmo que você construiu e depois envie (correio eletrônico) o texto do programa obtido para *cthomaz@fei.edu.br* .

Coloque no topo do programa uma linha com o seguinte conteúdo:

`// Primeira atividade – questão 9 – grupo ??`

Grupo 1

```
int main(int argc, char *argv[])
{   int quantF, elemento100, elemento30, elemento5, elemento1;
    cout << "Digite a quantidade de farinha: ";
    cin >> quantF;
    elemento100 = quantF / 100;
    elemento30 = (quantF % 100) / 30;
    elemento5 = (quantF % 30) / 5;
    elemento1 = (quantF % 5) / 1;

    cout << "Quantidade Elementos de 100g: " << elemento100 <<
endl;
    cout << "Quantidade Elementos de 30g: " << elemento30 << endl;
    cout << "Quantidade Elementos de 5g: " << elemento5 << endl;
    cout << "Quantidade Elementos de 1g: " << elemento1 << endl;

    system("PAUSE");
    return EXIT_SUCCESS;
}
```

Grupo 2

```
int main(int argc, char *argv[])
{
    int farinha,peso100, peso30, peso1, peso5, resto;

    cout<< "entre com o peso da farinha:";
    cin>>farinha;

    peso100= farinha / 100;
    resto= farinha %100;
    peso30=resto /30;
    resto= resto %30;
    peso5=resto /5;
    resto = resto %5;
    peso1= resto;

    cout <<"elementos de referencia:\n" <<peso100<<" de
100g;\n"<<peso30<<" de 30g;\n"<<peso5<<" de 5g;\n"<<peso1<<" de
1g;"<<endl;
    system("PAUSE");
    return EXIT_SUCCESS;
}
```

Grupo 3	`int main(int argc, char *argv[])`
	`{ int massa, resto, qtd100, qtd30, resto1, qtd5, qtd1;`
	`cout << "massa de farinha? ";`
	`cin >> massa;`
	`qtd100 = massa / 100;`
	`resto=massa % 100;`
	`qtd30 = resto / 30;`
	`resto1 = resto % 30;`
	`qtd5 = resto1 / 5;`
	`qtd1 = resto1 % 5;`
	`cout << qtd100 << " referencias de 100g" << endl;`
	`cout << qtd30 << " referencias de 30g" << endl;`
	`cout << qtd5 << " referencias de 5g" << endl;`
	`cout << qtd1 << " referencias de 1g" << endl;`
	`system("PAUSE");`
	`return EXIT_SUCCESS;`
	`}`

Grupo 4	`int main(int argc, char *argv[])`
	`{ int qtdeFarinha, ele100, ele30, ele5, ele1;`
	`cout << "Informe a quantidade de farinha: ";`
	`cin >> qtdeFarinha;`
	
	`ele100 = qtdeFarinha / 100;`
	`ele30 = (qtdeFarinha % 100) / 30;`
	`ele5 = ((qtdeFarinha % 100) % 30) / 5;`
	`ele1 = ((qtdeFarinha % 100) % 30) % 5;`
	
	`cout << "Serão utilizados " << ele100 << " elementos de 100g" << endl;`
	`cout << "Serão utilizados " << ele30 << " elementos de 30g" << endl;`
	`cout << "Serão utilizados " << ele5 << " elementos de 5g" << endl;`
	`cout << "Serão utilizados " << ele1 << " elementos de 1g" << endl;`
	`system("PAUSE");`
	`return EXIT_SUCCESS;`
	`}`

Grupo 5	`int main(int argc, char *argv[])`
	`{`
	`// Primeira atividade - questão 9 - grupo 5`
	`int qt100, qt30, qt5, qt1, peso, calc1, calc2;`
	`cout << "Coloque o peso: ";`
	`cin >> peso;`
	`qt100 = peso / 100;`
	`calc1 = peso % 100;`
	`qt30 = calc1 / 30;`
	`calc2 = calc1 % 30;`
	`qt5 = calc2 / 5;`
	`qt1 = calc2 % 5;`
	`cout << "100g: " << qt100 << endl;`
	`cout << "30g: " << qt30 << endl;`
	`cout << "5g: " << qt5 << endl;`
	`cout << "1g: " << qt1 << endl;`
	`system("PAUSE");`
	`return EXIT_SUCCESS;`
	`}`

Grupo 6	`int main( ){`
	`  int farinha, qtd100, qtd30, qtd5, qtd1;`
	`  cout<<”massa a confirmer: “; cin>>farinha;`
	`  qtd100 = farinha / 100;`
	`  qtd30 = (farinha % 100) / 30;`
	`  qtd5 = ((farinha % 100) % 30) / 5;`
	`  qtd1 = ((farinha % 100) % 30) % 5;`
	`  cout<<”elementos de 100g “<<qtd100<<endl;`
	`  cout<<”elementos de 30g  “<<qtd30<<endl;`
	`  cout<<”elementos de 5g   “<<qtd5<<endl;`
	`  cout<<”elementos de 1g   “<<qtd1<<endl;`
	`  system(“pause”); return(0);`
	
	`}`

Grupo 7	`int main(int argc, char *argv[])`
	`{`
	`    int pfarinha, g100, g30, g5, g1, gparcial;`
	
	`    cout << "Digite o peso da porção de farinha (g): ";`
	`    cin >> pfarinha;`
	
	`    g100 = pfarinha / 100;`
	`    gparcial = pfarinha % 100;`
	`    g30 = gparcial / 30;`
	`    gparcial = gparcial % 30;`
	`    g5 = gparcial / 5;`
	`    gparcial = gparcial % 5;`
	`    g1 = gparcial / 1;`
	
	`    cout << "A quantidade de elementos: " << g100 <<" de 100g, " << g30 <<" de 30g, " << g5 <<" de 5g e " << g1 <<" de 1g. " << endl;`
	
	`    system("PAUSE");`
	`    return EXIT_SUCCESS;`
	`}`

Grupo 8	`int main(int argc, char *argv[])`
	`{   int totfarinha, elemento1, elemento2, elemento3, elemento4;`
	
	`    cout << " Digite a quantidade de farinha: ";`
	`    cin >> totfarinha;`
	
	`    elemento1 = totfarinha / 100;`
	`    elemento2 = (totfarinha % 100 ) / 30;`
	`    elemento3 = ( (totfarinha % 100) % 30) / 5;`
	`    elemento4 = ( ( ( totfarinha % 100) % 30 )% 5) /1;`
	
	`    cout  <<"Elementos: " << elemento1 <<"Elemento<s>100g" << elemento2 <<"Elemento<s>30g" << elemento3<< "Elemento<s>5g" <<elemento4<< "Elemento<s>1g" <<endl;`
	
	`    system("PAUSE");`
	`    return EXIT_SUCCESS;`
	`}`

Grupo 9

```
int main(int argc, char *argv[])
{
    int e100,e30,e5,e1,resto,quantMassa;

    cout << "Digite a quantidade de massa: ";
    cin >> quantMassa;

    e100 = quantMassa / 100;
    resto = quantMassa % 100;
    e30 = resto / 30;
    resto = resto % 30;
    e5 = resto / 5;
    resto = resto % 5;
    e1 = resto;
    cout << "Quantidade de elemento(s):" << endl;
    cout << "100g: " << e100 << endl;
    cout << "30g: " << e30 << endl;
    cout << "5g: " << e5 << endl;
    cout << "1g: " << e1 << endl;
    system("PAUSE");
    return EXIT_SUCCESS;
}
```

Grupo 10

```
int main(int argc, char *argv[])
{
    // Primeira atividade - questao 9 - grupo 10
    int farinha, cemg, trintag, cincog, umg , resto;

    cout << "Digite a quantidade de farinha: "; cin >> farinha;
    cemg = farinha / 100;
    resto = farinha % 100;
    trintag = resto / 30;
    resto = trintag % 30;
    cincog = resto / 5;
    umg = cincog % 5;

    cout << "Quantidade de elementos de 100g: " << cemg << endl;
    cout << "Quantidade de elementos de 30g: " << trintag << endl;
    cout << "Quantidade de elementos de 5g: " << cincog << endl;
    cout << "Quantidade de elementos de 1g: " << umg << endl;

    system("PAUSE");
    return EXIT_SUCCESS;
}
```

questão 10 Se a porção de farinha possuir apenas 58g, quais os resultados obtidos a partir da execução do seu programa?	
Grupo 1	0X100g, 1X30g, 5x5g, 3x1g
Grupo 2	1 – 30g 5 – 5g 3 – 1g
Grupo 3	0 elementos de 100 1 elemento de 30 5 elementos de 5 3 elementos de 1
Grupo 4	0 elementos de 100g 1 elemento de 30g 5 elementos de 5g 3 elementos de 1g
Grupo 5	1x30g 5x5g 3x1g
Grupo 6	1 elemento de 30g 5 elementos de 5g 3 elementos de 1g
Grupo 7	O programa mostra 1x30g = 30g 5x5g = 25g 3x1g = 3g
Grupo 8	0 → elementos 100g. 1 → elementos 50g. 5 → elementos 5g. 3 → elementos 1g.
Grupo 9	1 de 30g, 5 de 5g e 3 de 1g.
Grupo 10	0 de 100g, 1 de 30g, 5 de 5g e 3 de 1g

questão 11

Observe a seqüência de linhas que constituem o seu programa e procure responder: a única ordem de disposição de linhas correta é essa que figura no seu programa ou é possível alguma inversão? Se for possível alguma inversão, indique alguma possibilidade.

Grupo 1	Neste algoritmo é possível realizar a inversão apenas entre as saídas.

Grupo 2	no programa
	peso100 = farinha/100;
	resto = farinha % 100;
	
	inversão
	resto = farinha %100;
	peso100 = farinha/100;

Grupo 3	Não é possível, pois a conta seguinte depende da conta anterior.
Grupo 4	É possível alguma inversão. Ex: Entre a segunda e quinta linha pode ser feita qualquer inversão.
Grupo 5	É possível inverter as linhas de comando qt5 e qt1 somente.

Grupo 6	Sim, pois as instruções de cálculo de valor não dependem dos resultados dos cálculos anteriores. Apenas dependem do valor de massa da farinha.
	Ex: m1 = ((farinha %100) % 30) % 5;
	m5 = ((farinha %100) % 30) / 5;
	m30 = (farinha %100) / 30;
	m100 = farinha/100;

Grupo 7	Não, pois no nosso programa a sequência posterior depende sempre do resultado da anterior.
Grupo 8	Sim é possível inversões, elemento1, elemento2, elemento3, elemento4, são independentes é possível trocar as variáveis de posição.
Grupo 9	Não é possível nenhuma inversão, pois todos os cálculos dependem do resto do cálculo anterior.
Grupo 10	Não é possível, pois todas as linhas dependem do cálculo que foi feito a partir da expressão anterior, então o programa iria rodar linha por linha, e caso, por exemplo, a expressão que armazena soluções à variável “cincog” efetuar a “conta” antes das outras expressões serem executadas, a variável armazenará mais elementos, pois não foram feitas divisões anteriores à ela. E ainda que as expressões são baseadas nos restos de outras divisões das primeiras expressões.

SEGUNDA ATIVIDADE

questão 1 No âmbito de um algoritmo ou programa, como pode ser descrito o papel de uma variável?	
Grupo 1	São empregadas para representar informações, como, por exemplo: quantidades, taxas e valores.
Grupo 2	A variável serve para armazenar informações e/ou dados.
Grupo 3	A função de uma variável é armazenar valores dos tipos: numérico, lógico e cadeia de caracteres.
Grupo 4	São empregadas para representar informações como medidas, taxas, quantidades, valores monetários, etc...
Grupo 5	É uma célula onde podemos armazenar valores.
Grupo 6	Receber valores inseridos pelo usuário ou modificados pelo sistema. Esses valores podem ser lógicos, numéricos ou alfanuméricos.
Grupo 7	Variável é um espaço lógico, no qual se armazena informações (dados) que serão utilizado no processo.
Grupo 8	A variável é responsável por armazenar valores durante a execução do programa.
Grupo 9	Tem o papel de armazenar valores, podendo ser numérico ou alfanumérico. Assim podem ser efetuados cálculos, armazenar conteúdos, etc.
Grupo 10	Variáveis podem receber números, letras, funções, qualquer informação.
Grupo 11	Armazenar dados temporareamente

questão 2 Que ações do sistema computacional podem ser relacionadas à execução do seguinte comando de atribuição: **`valor ← valor+1`** ?	
Grupo 1	Atribuirá, à variável “valor” a soma de 1.
Grupo 2	O programa irá adicionar o valor +1 à variável “valor”.
Grupo 3	A variável valor recebe ela mesma somada com 1. Sintaxe: int valor; valor = valor + 1;
Grupo 4	Valor recebe a soma de valor + 1
Grupo 5	O “valor” da expressão é somado a mais um, atribuindo então a essa variável um novo valor.
Grupo 6	Somar “1” ao valor da variável.
Grupo 7	Nesta ação o sistema pega o dado (valor) armazenado na variável valor e som a 1 ao mesmo, o resultado obtido e relocado a variável valor, substituindo sua informação inicial.
Grupo 8	soma e atribuição
Grupo 9	Atribuir à variável “valor” o conteúdo de “valor + 1.”
Grupo 10	Recebe determinado valor ou função, para a atividade seguinte, seja um cálculo, expressão ou uma ação.
Grupo 11	Adição e atribuição

questão 3 Execute o aplicativo para responder: se a siderúrgica encomendar 12,5t de carvão, qual deverá ser a quantidade extraída da mina?	
Grupo 1	Deve-se extrair 12.8187 toneladas.
Grupo 2	12,8187 toneladas.
Grupo 3	Com base na execução do programa deverá ser extraída da mina de carvão 12,8187t.
Grupo 4	12,8187 toneladas
Grupo 5	12,8187 toneladas
Grupo 6	12.8187 toneladas
Grupo 7	deverá ser extraído 12,8187 toneladas de carvão.
Grupo 8	Será 12.8187 T
Grupo 9	12,8187 toneladas
Grupo 10	12.8187 toneladas
Grupo 11	12.8187 toneladas

questão 4

Determine a quantidade de carvão que será entregue à siderúrgica se o operador da mina extrair e carregar o trem com 1,5t de carvão. Para responder a esta questão faça algumas execuções do aplicativo para buscar uma aproximação da quantidade que será entregue, por exemplo: execute o aplicativo colocando 1,45t como quantidade a ser entregue na siderúrgica e depois procure melhorar a aproximação com quantidades a entregar mais adequadas; anote suas tentativas.

Grupo 1	Tentativas: 1,463t 1.462t 1,4625t 1,4626t Quantidade encomendada será 1,4627t.
Grupo 2	1,45t = 1,4870t 1,46t = 1,4972t 1,47t = 1,5075t 1,465t = 1,5024t 1,462t = 1,4993t 1,463t = 1,5003t 1,4628 = 1,5001 1,4627 = 1,5t
Grupo 3	Q. Encomendada Q. Extraída 1,47t 1,5075t
Grupo 4	1,45 = 1,4870t 1,46 = 1,4972t 1,47 = 1,5075t 1,464 = 1,5013t 1,463 = 1,5003t 1,4625 = 1,4998t 1,4626 = 1,4999t 1,4627 = 1,5 toneladas
Grupo 5	mina siderúrgica 1,45t = 1,4870t 1,46t = 1,4972t 1,47t = 1,5075t 1,48t = 1,5177t 1,49t = 1,5280t 1,50t = 1,5382t
Grupo 6	1,45 fica : 1,4870 toneladas 1,40 fica: 1,5177 toneladas 1,47 fica: 1,5075 toneladas 1,468 fica: 1,5054 toneladas 1,4627 fica: 1,5000 toneladas
Grupo 7	quantidade encomendada 1,45 1,47 1,465 1,46267 qporto 1,46 1,4819 1,4768 1,4745 qextraida 1,4870 1,5075 1,5024 1,50

Grupo 8	1.2 → 1,2306 1,46 → 1,4972 1, 4659 → 1,5033 1,4627 → 1,5000
Grupo 9	Extraída → Entregue 1,4877 → 1,45t 1,4972 → 1,46t 1,4993 → 1,462t 1,5001t → 1,4628t 1,5 t → 1,4627t
Grupo 10	1.45t → 1.48t 1.47t → 1.5075t 1.46275 → o resultado será 1.5000 toneladas
Grupo 11	1.45 → 1.4870 1.47 → 1.5075 1.46 → 1.4972 1.465 → 1.5024 1.463 → 1.5003 1.4629 → 1.5002 1.4628 → 1.5001 1.4627 → 1.5000

questão 5 As linhas 2 e 3 do algoritmo podem ter suas disposições invertidas? O algoritmo permanecerá correto? Justifique.	
Grupo 1	Não, pois, a Linha 3 é dependente do resultado da linha 2
Grupo 2	Não porquê na linha e é necessário a variável “qporto” conter um valor, e a variável recebe seu valor na linha 2. E mesmo invertendo as variáveis, o resultado será um número diferente.
Grupo 3	Não, pois o sistema irá tentar utilizar a variável qporto antes de ser atribuído um valor a ela. Isso resultará em um erro.
Grupo 4	Não pode, pois qextraida depende de qporto.
Grupo 5	Não, pois na 2ª linha ele determina o valor da variável “qporto” que é utilizado n a 3ª linha para determinar o valor da variável “qextraida”. Ou seja, “qextraida” depende de “qporto”
Grupo 6	Não, porque a variável utilizada na operação , da terceira linha tem o valor atribuído na segunda linha. Se tivesse invertidas o resultado seria “0”;
Grupo 7	não, pois a segunda execução depende da primeira.
Grupo 8	Não, pois a linha 3 depende de um valor obtido na linha 2
Grupo 9	Não, o algoritmo não permanecerá correto, pois a linha 3 depende da linha 2, na qual é atribuída a variável “qporto” um valor que será usada na linha 3.
Grupo 10	Não. A linha 3 depende da linha 2.
Grupo 11	Não, pois a linha 3 depende da linha 2.

questão 6 Qual seria o valor integral do IPVA para um veículo cujo valor da nota de compra/venda seja de R$30000,00? Observe que se o mês de licenciamento do veículo for janeiro (mês 1), o valor a ser recolhido será igual ao valor integral.	
Grupo 1	O valor integral é R$ 1200,00.
Grupo 2	A cada mês há um desconto de R$100,00, exceto se o mês for janeiro, no qual o valor pago é integral.
Grupo 3	Com base no valor de R$30 000,00 o valor integral será de R$1 2000 00.
Grupo 4	Valor integral será R$ 1 200, 00.
Grupo 5	1200,00
Grupo 6	O valor integral é R$ 1.200,00 referente a 4% do valor do veículo.
Grupo 7	1.200,00
Grupo 8	IPVA = R$1200,00
Grupo 9	Seria R$ 1 200, 00
Grupo 10	Sim, os respectivos valores são iguais a R$ 1200,00
Grupo 11	R$ 1200,00

questão 7 Descreva uma expressão matemática que relacione o valor da nota de compra/venda com o valor integral do IPVA. IPVA integral = ???	
Grupo 1	IPVA integral = Valor da nota compra e venda * 0,04
Grupo 2	ipva integral = valo Nota * 0.04;
Grupo 3	valorVeiculo * 0,04 = IPVA Integral.
Grupo 4	IPVA integral = 30 000 * 0,04
Grupo 5	IPVA integral - notacompra * (4/100).
Grupo 6	IPVA integral = nota compra/venda *0,04
Grupo 7	valor integral = valor do veiculo x (4/100)
Grupo 8	valornota . 0.04 = IPVA integral
Grupo 9	IPVA integral = valor Nota * 0,04
Grupo 10	IPVA integral = valor nota * 0.04;
Grupo 11	IPVA integral = valor da nota * 0,04

questão 8

Que fração do valor integral do IPVA deve corresponder ao valor a ser recolhido se o licenciamento do veículo novo for efetivado no mês de julho (mês 7)? E se o licenciamento ocorrer em dezembro?

Grupo 1

mês	1	2	3	4	5	6	7	8	9	10	11	12
fração	12/12	11/12	10/12	9/12	8/12	7/12	6/12	5/12	4/12	3/12	2/12	1/12

Grupo 2

mês	1	2	3	4	5	6	7	8	9	10	11	12
fração	12/12	11/12	10/12	9/12	8/12	7/12	6/12	5/12	4/12	3/12	2/12	1/12

Grupo 3

mês	1	2	3	4	5	6	7	8	9	10	11	12
fração	12/12	11/12	10/12				6/12					1/12

Grupo 4

mês	1	2	3	4	5	6	7	8	9	10	11	12
fração	12/12	11/12	10/12	9/12	8/12	7/12	6/12	5/12	4/12	3/12	2/12	1/12

Grupo 5

mês	1	2	3	4	5	6	7	8	9	10	11	12
fração	12/12	11/12	10/12	9/12	8/12	7/12	6/12	5/12	4/12	3/12	2/12	1/12

Grupo 6

mês	1	2	3	4	5	6	7	8	9	10	11	12
fração	12/12	11/12	10/12	9/12	8/12	7/12	6/12	5/12	4/12	3/12	2/12	1/12

Grupo 7

mês	1	2	3	4	5	6	7	8	9	10	11	12
fração	12/12	11/12	10/12				6/12					1/12

Grupo 8

mês	1	2	3	4	5	6	7	8	9	10	11	12
fração	12/12	11/12	10/12	9/12	8/12	7/12	6/12	5/12	4/12	3/12	2/12	1/12

Grupo 9

mês	1	2	3	4	5	6	7	8	9	10	11	12
fração	12/12	11/12	10/12	9/12	8/12	7/12	6/12	5/12	4/12	3/12	2/12	1/12

Grupo 10

mês	1	2	3	4	5	6	7	8	9	10	11	12
fração	12/12	11/12	10/12	9/12	8/12	7/12	6/12	5/12	4/12	3/12	2/12	1/12

Julho → 6/12 --- Dezembro → 1/12

Grupo 11

mês	1	2	3	4	5	6	7	8	9	10	11	12
fração	12/12	11/12	10/12				6/12					1/12

questão 9 Descreva uma expressão matemática que relacione o valor integral do IPVA e o número do mês do licenciamento com o valor do IPVA a ser recolhido. IPVA recolhido = ???	
Grupo 1	IPVA R = [{Vnota * 0,04)/12] * [12 – (mes atual -1)]
Grupo 2	Ipva recolhido =e ipva integral – (100 * mês pago);
Grupo 3	IPVA R = ((valorVeiculo * 0,04) /12) * (12 – (mesAtual – 1))
Grupo 4	IPVA Recolido = [(valor Nota * 0,04) /12] * [12 – (mês atual -1)]
Grupo 5	IPVArecolhido = (IPVAtotal/12) *(13 – mes);
Grupo 6	IPVA recolhido = IPVA integral * (13 – mês)/12
Grupo 7	IPVA recolhido = IPVA integral/12
Grupo 8	Valor Nota = [(13 – mes)/ 12] . 0,04 = IPVA recolhido
Grupo 9	IPVA recolhido = (IPVA integral/12) . (13 – numero do mes)
Grupo 10	IPVA recolhido = (13 – mes)/12;
Grupo 11	IPVA recolhido = (IPVA integral * (13 – mes))/12

questão 10

Faça a descrição de um algoritmo que represente um método de resolução do problema proposto (articule de forma adequada as relações matemáticas obtidas nas questões anteriores). Escolha identificadores de variáveis que indiquem os respectivos significados.

Grupo 1

Assinatura
Objetivo: Calcular o valor do IPVA
Entrada: Valor da nota compra/venda, mes da compra
Saída: Valor a ser recolhido
Corpo

```
Leia (nota); Leia (mes);
IpvaIntegral → nota *0,04;
IpvaRecolhido → ((nota*0,04)/12) * (12 – (mes-1));
Imprima (IpvaIntegral); Imprima (IpvaRecolhido);
```

Grupo 2

Assinatura
Objetivo:
Entrada:
Saída:
Corpo

```
Calcula IPVA ( )
```

```
leia(ValorNota); leia(mesPago);

valorIntegral = valorNota * 0.04;
valorRecolhido = valorIntegral – (100*(mesPago – 1));

imprima (valorIntegral);
imprima (valorRecolhido);
```

Grupo 3

Assinatura
Objetivo: OBTER O VALOR INTEGRAL E DE RECOLHIMENTO DO IPVA
Entrada: VALORVEICULO(FLOAT); VALORMES(INT);
Saída: IPVAINTEGRAL; IPVARECOLHIDO;
Corpo

```
IPVA( )
      LEIA(VELORVEICULO); LEIA(VALORMES);
      IPVAINTEGRAL ← VALORVEICULO * 0.04;
      IPVARECOLHIDO ← (IPVAINTEGRAL/12)*(12– (VALORMES-1));
      IMPRIMA(IPVAINTEGRAL);
      IMPRIMA(IPVARECOLHIDO);
```

Grupo 4

Assinatura
Objetivo: Calcular o valor do IPVA recolhido
Entrada: Valor Nota, Valor atual → reaia
Saída: IPVAR → reais
Corpo

```
IPVA Recolhido ( )
      leia (valor Nota), leia (mes Atual);
      IPVARec ← ((valorNota*0,04/12) * (12 – (mesAtual– 1);
      imprima (IPVARec);
```

Grupo 5	*Assinatura*
	Objetivo: Calcular o valor recolhido IPVA
	Entrada: valorcompra, mes – tipo real
	Saída: IPVA Recolhido – tipo real
	Corpo
	`IPVA ( )`
	`leia(valorDaCompra), leia(mes);`
	`IPVAtotal = valorDaCompra * (4/100);`
	`IPVArecolhido = (IPVAtotal/12) * (13 – mes);`
	`imprima (IPVArecolhido);`

Grupo 6	*Assinatura*
	Objetivo:Calcular o valor integral e o valor a ser pago
	Entrada: IPVA; valor da nota, mês - Real
	Saída: IPVA pago, integral: Real
	Corpo
	`IPVA ( );`
	`leia(valordanota; leia(mes);`
	`integral ← valordanota * 0.04;`
	`IPVApago ← integral * (13 – mes)/12;`
	`imprima(integral); imprima(IPVApago);`

Grupo 7	*Assinatura*
	Objetivo: Obter o valor a ser recolhido como pagamento do IPVA
	Entrada: valorregnota – tipo real / mes – tipo inteiro
	Saída: valorrecolhido – tipo real
	Corpo
	// segunda atividade – questão 11 – grupo 07
	`IPVA( )`
	`leia(valorregnota); leia(mes);`
	`valorint ← valorregnota * (4/100);`
	`valormes ← valorint/12;`
	`valorrecolhido ← valormes * (13 – mes);`
	`imprima (valorrecolhido);`

Grupo 8	*Assinatura*
	Objetivo: Obter valor do IPVA
	Entrada: valor da nota
	Saída: valor do IPVA
	Corpo
	`float ipva, valorNota, mes;`
	`leia(valorNota);`
	`leia(mes);`
	`ipva = valorNota * ((13 – mes) /12) * 0.04;`
	`imprima (ipva);`

Grupo 9	*Assinatura*
	Objetivo: Calcular o imposto integral e o imposto a ser recolhido.
	Entrada: Valor da nota de compra/venda; número do mês do licenciamento
	Saída: Imposto integral; imposto a ser recolhido
	Corpo
	`IPVA ( )`
	`leia(valorNota); leia(mesLicenciamento);`
	
	`ipvaIntegral ← valorNota * 0.04;`
	`ipvarecolhido ← (ipvaIntegral/12)*(13-mesLicenciamento);`
	
	`imprima(ipaIntegral); imprima(ipvaRecolhido);`

Grupo 10	*Assinatura*
	Objetivo: Cálculo IPVA
	Entrada: valor, mês
	Saída: integral, recolhido → tipo real
	Corpo
	`leia(valor), leia(mes);`
	`ipvaint ← valor *0,04;`
	`ipvarec ← ((13-mes)/ 12) * ipvaint;`
	`imprima(ipvaint);`
	`imprima(ipvarec);`

Grupo 11	*Assinatura*
	Objetivo: obter o valor a ser recolhido do IPVA
	Entrada: valorNota (float), mes (int)
	Saída: ipvarecolhido (float)
	Corpo
	`IPVA ( )`
	`leia(valorNota), leia(mes);`
	`ipvaIntegral ← valorNota *  0,04;`
	`ipvarecolhido ← (ipvaintegral * (13-mes)) DIV 12;`
	`imprima (ipvarecolhido);`

questão 11

Faça a implementação e testes do programa correspondente ao algoritmo que você construiu e depois envie (correio eletrônico) o texto do programa obtido para *cthomaz@fei.edu.br* .

Coloque no topo do programa uma linha com o seguinte conteúdo:

`// Segunda atividade – questão 11 – grupo ??`

Grupo 1

```
int main(int argc, char *argv[])
{
    float nota, mes, ipvaIntegral, ipvaRecolhido;
    cout << "Digite o valor da nota: " ;
    cin >> nota;
    cout << "Digite o mes de compra: " ;
    cin >> mes;
    ipvaRecolhido = ((nota * 0.04) / 12 ) * ( 12 - (mes - 1));
    ipvaIntegral = nota * 0.04;
    cout << "Valor do  IPVA integral: " << ipvaIntegral <<endl;
    cout << "Valor do IPVA recolhido: " << ipvaRecolhido << endl;
    system("PAUSE");
    return EXIT_SUCCESS;
}
```

Grupo 2

```
int main(int argc, char *argv[])
{
    double valorNota = 0,
           valorIntegral = 0,
           valorRecolhido = 0;

    int mesPago = 0;
    cout << "\n   Valor da Nota: "; cin >> valorNota;
    cout << "\n   Mes da Compra: "; cin >> mesPago;
    valorIntegral = valorNota * 0.04;
    valorRecolhido = valorIntegral - (100 * (mesPago - 1));
    cout << "\n\n      Valor Integral:  RS " << valorIntegral <<
"\n      Valor Recolhido:  RS " << valorRecolhido << "\n\n";

    system("PAUSE");
    return EXIT_SUCCESS;
}
```

Grupo 3

```
int main(int argc, char *argv[])
{
    float valorVeiculo,IPVAIntegral,IPVARecolhido;
    int valorMes;
    cout << "Digite o valor do veiculo: ";
    cin >> valorVeiculo;

    cout << "Digite o mes: ";
    cin >> valorMes;
    IPVAIntegral = valorVeiculo * 0.04;
    IPVARecolhido = (IPVAIntegral / 12) * ( 12 - (valorMes - 1));

    cout << "O valor do IPVA Integral e de: " << IPVAIntegral <<
endl;
    cout << "O valor do IPVA Recolhido nesse mes e de: " <<
IPVARecolhido << endl;
    system("PAUSE");
    return EXIT_SUCCESS;
}
```

Grupo 4

```
int main(int argc, char *argv[])
{
        float valorNota, mesAtual, ipvaRec;
        cout << "Entre com o valor da nota: ";
        cin >> valorNota;
        cout << "Entre com o mes atual: ";
        cin >> mesAtual;

        ipvaRec = ((valorNota * 0.04) / 12) * (12 - (mesAtual -
1));

        cout << "Valor IPVA recolhido: " << ipvaRec << endl;


    system("PAUSE");
    return EXIT_SUCCESS;
}
```

Grupo 5

```
int main(int argc, char *argv[])
{   //Segunda Atividade - Questao 11 - Grupo 05
    float IPVATotal, ValorDaCompra, Mes, IPVARecolhido;

    cout << "Digite O Valor Da Compra: ";
    cin >> ValorDaCompra;

    cout << "Digite O Mes Da Compra: ";
    cin >> Mes;

    IPVATotal = ValorDaCompra * 0.04;
    IPVARecolhido = (IPVATotal / 12) * (13 - Mes);

    cout << "O Valor Do IPVA A Ser Recolhido Sera: " <<
IPVARecolhido << endl;

    system("PAUSE");
    return EXIT_SUCCESS;
}
```

Grupo 6

```
int main(int argc, char *argv[])
{
    float valorDaNota, mes, IPVApago, integral;

    cout << "Digite o valor do veiculo descrito na nota de
compra/venda: ";
    cin >> valorDaNota;
    cout << "Digite o mes de compra d veiculo: ";
    cin >> mes;

    integral = valorDaNota * 0.04;
    IPVApago = integral * (13-mes)/12;

    cout << "\nValor integral do IPVA: " << integral << endl;
    cout << "Valor a ser pago: " << IPVApago << endl;
    system("PAUSE");
    return EXIT_SUCCESS;
}
```

Grupo 7

```
int main(int argc, char *argv[])
{   float valorregnota, valorint, valormes, valorrecolhido;
    int mes;

    cout << " Entre com o valor de compra e venda do veiculo: ";
    cin >> valorregnota;
    cout << " Entre com o mes de compra do veiculo: ";
    cin >> mes;

    valorint = valorregnota * 4/100;
    valormes = valorint / 12;
    valorrecolhido = valormes * (13-mes);

    cout << " Valor a ser recolhido como pagamento do IPVA: " <<
valorrecolhido << endl;

    system("PAUSE");
    return EXIT_SUCCESS;
}
```

Grupo 8

```
int main(int argc, char *argv[])
{
    float ipva, valorNota, mes;

    cout << " Entre com o valor do veiculo : ";
    cin >> valorNota;

    cout << " Entre com o mes da compra : ";
    cin >> mes;

    ipva = valorNota*((13-mes)/12)*0.04;

    cout << " Valor do IPVA : " << ipva << endl;
    system("PAUSE");
    return EXIT_SUCCESS;
}
```

Grupo 9

```
int main(int argc, char *argv[])
{
/*
    Segunda atividade - Questão 11 - Grupo 09
*/
    float ipvaIntegral, ipvaRecolhido, mesLicenciamento,
valorNota;

    cout << "Entre com o valor da Nota de compra/venda: R$ "; cin
>> valorNota;
    cout << "Entre com o mes do licenciamento: "; cin >>
mesLicenciamento;

    ipvaIntegral = valorNota * 0.04;
    ipvaRecolhido = (ipvaIntegral / 12) * (13 - mesLicenciamento);

    cout << "\nValor do IPVA: ";
    cout << "\n  Imposto Integral: R$ " << ipvaIntegral;
    cout << "\n  Imposto a ser Recolhido: R$ " << ipvaRecolhido <<
endl << endl;
    system("PAUSE");
    return EXIT_SUCCESS;
}
```

Grupo 10	`int main(int argc, char *argv[])`
	`{ float valor, mes, ipvaint, ipvarec;`
	`//Entrada`
	`cout<<"Digite o valor da NF compra/venda: ";`
	`cin>>valor;`
	`cout<<"Digite o mes de compra: ";`
	`cin>>mes;`
	
	`//Processamento`
	`ipvaint = valor*0.04;`
	`ipvarec = ((13-mes)/12)*ipvaint;`
	
	`//Saida`
	`cout<<"Valor integral do IPVA: " <<ipvaint<<endl;`
	`cout<<"Valor recolhido do IPVA: " <<ipvarec<<endl;`
	`system("PAUSE");`
	`return EXIT_SUCCESS;`
	`}`

Grupo 11	`int main(int argc, char *argv[])`
	`{ float valorNota, ipvaIntegral, ipvaRecolhido;`
	`int mesLicenciamento;`
	`cout << "Entre com o valor da nota: ";`
	`cin >> valorNota;`
	`cout << "Entre com o mes de licenciamento: ";`
	`cin >> mesLicenciamento;`
	`ipvaIntegral = valorNota * 0.04;`
	`ipvaRecolhido = (ipvaIntegral * (13 - mesLicenciamento)) / 12.0;`
	`cout << "O valor do IPVA integral eh: " << ipvaIntegral << endl;`
	`cout << "O valor do IPVA recolhido eh: " << ipvaRecolhido << endl;`
	`system("PAUSE");`
	`return EXIT_SUCCESS;`
	`}`

questão 12 Observe a seqüência de linhas que constituem o seu programa e procure responder: a única ordem de disposição de linhas correta é essa que figura no seu programa ou é possível alguma inversão? Se for possível alguma inversão, indique alguma possibilidade.	
Grupo 1	Dá pra trocar entradas e saídas,
Grupo 2	É possível a inversão pois na linha onde se calcula o valor da variável “valor Recolhido” ao inves de pegar o valor de “valorIntegral” basta colocar a fórmula que calcula o valor desta variável e somente depois se ter o valor desta variável “valorNota”.
Grupo 3	Sim, é a única ordem de disposição, pois se alterar ela, a variável IPVAIntegral não irá valer.
Grupo 4	A única ordem de disposição de linhas que pode ser invertida são as linhas de “entrada”
Grupo 5	Não é possível pois as variáveis dependem uma das outras.
Grupo 6	Não é possível nenhuma inversão.
Grupo 7	Não, porque cada linha de execução depende sempre da anterior.
Grupo 8	Sim, somente a leitura dos valores pode ser invertida.
Grupo 9	Não impossível inversão, pois a linha que calcula o IPVA Recolhido, depende do cálculo da linha anterior.
Grupo 10	Não, não há possibilidade de inversão.
Grupo 11	Não é possível nenhuma inversão.

TERCEIRA ATIVIDADE

questão 1 Qual a finalidade das estruturas de controle de seleção em um algoritmo ou programa?	
Grupo 1	A finalidade é a contrução de algoritmos com fluxos de processamento alternativo
Grupo 2	Verificar se há condições de ser executada um certo bloco de ações.
Grupo 3	Definir condições a serem executadas dependendo dos valores de entrada.
Grupo 4	Esta estrutura é possível a construção de algoritmos com fluxos de processamento alternativos.
Grupo 5	Construir dentro dos algoritmos caminhos alternativos
Grupo 6	Construir algoritmo com ramificações, por exemplo, temos uma condição e através da mesma o processo pode escolhas diferenças.
Grupo 7	Sem resposta.
Grupo 8	É decidir entre a execução de partes diferentes do algoritmo dependendo de uma condição
Grupo 9	Criar processos alternativos no algoritimo tendo mais de uma opção.

questão 2

Observe a seqüência de instruções abaixo:

```
1)      float x;
2)      cout<<"valor de x? ";
3)      cin>>x;
4)      x=3*x;
5)      if(x<7.2){
6)         x=x+10;
7)         x=1.5*x;
8)      }
9)      else{
10)        x=2.5*x;
11)        x=x+10;
12)     }
13)     cout<<"novo x: "<<x<<endl;
```

Quais linhas de instruções serão executadas se o valor fornecido como conteúdo inicial da variável **x** for **4.4**? Nesse caso, qual será o valor armazenado como conteúdo dessa variável ao final da execução da seqüência de instruções?

E se o conteúdo inicial de **x** for **1.8**?

Grupo 1	Serão executadas as linhas, 9, 10, 11 12, 13. X será 43. Serão executadas as linhas 5, 6, 7, 8 e 13. E o resultado de x será 23,1.
Grupo 2	9.,10.,11.,12.,13. X=13,2 5.,6.,7.,8.,13. x=5,4
Grupo 3	Linhas: 1,2,3,4,9,10,11,12,13... Resultado=43 Linhas: 1,2,3,4,5,6,7,8,13... Resultado=23.1
Grupo 4	Linhas 10, 11 variável final será 43 Linhas 6, 7 variável final será 23.1
Grupo 5	Serão executadas as linhas 10 e 11. A variável passará à valor 43 Serão executadas as linhas 6 e 7. A variável passará à armazenar 22,65
Grupo 6	linhas executadas 2, 3, 4, 5, 9, 10, 11 e 13 e valor armazenado é 43. linhas executadas 2, 3, 4, 5, 6, 7 e 13. e valor armazenado é 23,1.
Grupo 7	4, 9 a 13 O valor e 43 4 a 8 e 13 O valor e 23,1
Grupo 8	As linhas 10, 11 e 13. E o resultado armazenado será 43. As linhas 6, 7 e 13. E o resultado será 23,1.
Grupo 9	Linhas 1, 2, 3, 4, 9, 10, 11, 12, 13. X = 42 Linhas 1, 2, 3, 4, 5, 6, 7, 8, 13 X = 23,1

questão 3

Observe a seqüência de instruções abaixo e considere a afirmação:

para mesmos valores iniciais de x ela produzirá as mesmas "saídas" que foram resultados da execução da seqüência acima.

A afirmação é verdadeira ou falsa? Justifique.

```
...
 float x;
 cout<<"valor de x? ";
 cin>>x;
 x=3*x;
 if(x>=7.2){
    x=2.5*x;
    x=x+10;
 }
 if(x<7.2){
    x=x+10;
    x=1.5*x;
 }
 cout<<"novo x: "<<x<<endl;
...
```

Grupo 1	É verdadeira, pois
Grupo 2	Sim, porque as condições e os blocos de comandos foram trocados.
Grupo 3	Verdadeira, pois, mesmo com a diferença apresentada na sintaxe, as condições são as mesmas.
Grupo 4	Verdadeira, se o valor de x na linha 4 for diferente de 7,2 e falso se o valor de x na linha 4 for igual a 7,2
Grupo 5	Falsa, pois para um valor de X ≥ 7,2 ele irá executar uma função, para X < 7,2 fará outra função.
Grupo 6	É verdadeira, por que as condições são iguais apenas foram alteradas a sequência de execução.
Grupo 7	Sim, pois os comandos alterados não alteram o valor do resultado.
Grupo 8	Verdadeira, pois a sequência dos blocos foi apenas invertida, e junto a condição.
Grupo 9	Verdadeira, pois o programa é o mesmo, mas em outra ordem e escrito de outra forma.

questão 4 Se as duas medidas conhecidas forem 30º e 50º, qual será a medida do terceiro ângulo?	
Grupo 1	180º-80º = 100º medida do terceiro ângulo = 100º
Grupo 2	100º
Grupo 3	100º
Grupo 4	o terceiro angulô terá 100º
Grupo 5	30º+50º-180º=100º
Grupo 6	100º
Grupo 7	A medida será 100º
Grupo 8	100º
Grupo 9	100º

questão 5 Descreva quatro casos (quatro duplas de valores inteiros) para os quais a resposta produzida pelo algoritmo seja 70º. Experimente os pares de valores escolhidos com a execução do aplicativo.	
Grupo 1	55º,55º 60º,50º 65º,45º 70º,40º
Grupo 2	100º e 10º 90º e 20º 80º e 30º 70º e 40º
Grupo 3	55º,55º 70º,40º 50º,60º 90º,20º
Grupo 4	1º - 55º e 55º 2º - 50 e 60º 3º - 10 e 100º 4º - 20 e 90º
Grupo 5	55º+55º-180º=70º 45+60-180º=70º 20º+90º-180º=70º ou seja 30º+80º-180º=70º alfa+beta-180º=70º
Grupo 6	50º e 60º 100º e 10º 80º e 30º 70º e 40º
Grupo 7	medidas 10º e 100º medidas 20º 90º medidas 30º 80º medidas 40º 70º
Grupo 8	55º + 55º 45º + 65º 50º + 60º 40º + 70º
Grupo 9	alfa: 100º beta: 10º alfa: 90º beta: 20º alfa: 55º beta: 55º alfa: 70º beta: 40º

questão 6 Descreva um caso em que as medidas sejam incompatíveis com a existência do triângulo. Experimente.	
Grupo 1	100º,80º é imconpativel com a existência de um triângulo
Grupo 2	Um ângulo 30º e o outro 160º ângulo=180º-(30º+160º) ângulo=180º-190º ângulo=10º
Grupo 3	Caso o resultado da soma alfa+beta seja maior que 180º ou algum dos valores de entrada for igual a zero, os valores serão incompatíveis.
Grupo 4	100º e 200º
Grupo 5	se alfa + beta + gama ≠ 180º 50 + 50 + 50 = 150º ≠ 180º 120 + 70 + 100 = 290º ≠ 180º
Grupo 6	quando um dos ângulos de entrada for igual a 0.
Grupo 7	medidas 10º e 90º
Grupo 8	120º e 120º
Grupo 9	alfa: 80 beta: 110

questão 7 Como você descreve a finalidade da instrução colocada na linha 4 do algoritmo?	
Grupo 1	se o ângulo alfa for maior que zero e o ângulo beta for maior que zero e a soma de alfa mais beta for menor que 180º. Ele atribuirá o valor de gama e imprimirá “compatível”. Senão imprimirá “incompativel”
Grupo 2	Verifica se as medidas dos ângulos correspondem a um triângulo.
Grupo 3	Condição que define se os valores de entrada são ou não são compatíveis.
Grupo 4	Alfa e Beta precisa ser maior que Ø (zero) e a soma de ALFA e Beta precisa ser menor que 180,
Grupo 5	Ela verificará se os valores correspondem a um triângulo, ou seja, se os valores dos ângulos são maiores que 0º e a soma deles não for menor que 180º, pois 180º é a medida dos 3 ângulos do triângulo.
Grupo 6	determinar se as medidas são compatíveis com os ângulos interno do triângulo.
Grupo 7	É definido como os ângulos não podem ser menores que zero. E a soma deve ser menor que 180º.
Grupo 8	É a instrução para verificar a existência de uma condição, verdadeira ou falsa. Nesse caso verifica se os ângulos inseridos são maiores que 0º, e se a soma entre eles é menor que 180º, pois só assim pode haver um triângulo.
Grupo 9	A linha apresenta a condição para que os ângulos possam estar no triângulo.

questão 7 Para um percurso total de 13050km a porcentagem que definirá o prêmio será 11,1%. Explique o processo aplicado para o cálculo dessa taxa de porcentagem.	
Grupo 1	Kilometragem até 12000 5,5% do salario acima de 12000 mais 2,8% a cada 600km e eventuais parcela menores
Grupo 2	12000km - 5,5% + 1050/600=1,3 600km - 2,8% + 450km - 2,8% + 13050km - 11.1%
Grupo 3	5,5% para até 12000km somados a 5,6% pelos 1050km excedentes.
Grupo 4	12.000 → 5.5% + 600 → 2,8% + 450 → 2,8% 13050 11,1
Grupo 5	13050km = 12000km + 600km + 450km ↓ ↓ ↓ porcentagem = 5,5% + 2,8% + 2,8% = 11,1
Grupo 6	13050km = 12000 + 600 + 450 ↓ ↓ ↓ 5,5% + 2,8% + 2,8% = 11,1% Acima de 12000km ele soma 5,5%, o rescante de 12000 ele divide por 600 e a cada 600 ele soma 2,8% e o que sobrou de divisão de 600 soma 2,8%.
Grupo 7	Percorreu 12000km e recebeu 5,5%. Percorreu mais 600km e recebeu 2,8. Percorreu mais 450 e recebeu 2,8.
Grupo 8	13050 – 12000 = 1050km = 600 + 450 prêmio: 5,5% + 2,8% + 2,8% do salário bruto ↓ 11,1%
Grupo 9	Foram 5,5% devido a quilometragem ter passado dos 12000km, mais 2,8% a cada 600km passados de 12000km. E nesse caso passou mais duas vezes, chegando assim aos 11,1%

questão 8 Descreva uma expressão matemática que indique a forma de cálculo do valor do prêmio a partir da taxa (%) e do valor do salário bruto. Suponha que a taxa já tenha sido calculada.	
Grupo 1	OBS= taxa será escrita em forma decimal, exempo = 1,05 premio = (salário bruto x taxa) – salario bruto
Grupo 2	valorpremio=salario+(salário*0,055)+((quilometragem-12000)/600)*0,028)
Grupo 3	valorprêmio = salariobruto * (taxa/100)
Grupo 4	Premio = salário*(taxa/100)
Grupo 5	premio = salario * taxa
Grupo 6	premio = salário * taxa/100
Grupo 7	prêmio = saláriobruto * taxa
Grupo 8	Taxa/100*salárioBruto = Prêmio
Grupo 9	Premio = (salariobruto x taxa)/100

questão 9 Organize um conjunto de expressões matemáticas para traduzir a relação entre a quilometragem total e a taxa de porcentagem que deve ser aplicada para o cálculo do valor do prêmio.	
Grupo 1	Se Km > 12000 = (kilometragem - 12000 / 600) x 0,028 → taxa Se Km ≥ 12000 = 0,028 → taxa Km 12000 = 0,055 → taxa taxa = acima 12000 + Km12000 + Calculo de Parcela
Grupo 2	prêmiofixo=quilometragem-12000 premiovariavel=(quilometragem-12000)/600 somapremiovariavel=se(premiovariavel mod10>0; premiovariavel+1;premiovariavel) somapremiofixo=se(premiofixo≥0;1;0) porcentagem=somapremiofixo*0,055+somapremiofixo(premiovariavel*0,028)
Grupo 3	Bônus = 5,5% ↓ se (km > 12000) {exc1=((km-12000)/600)*2.8} se ((km-12000)mod600) ≠0{exc2=2.8} bonus = bonus+exc1+exc2
Grupo 4	SALDOkm=(kmtotal-12000) PremioParcial=((saldokm/600)*2,8)+5,5 Premiototal=((saldokm mod 600)*2,8) + PremioParcial)
Grupo 5	taxa = 5,5%((quilômetros-12000)/600).2,8%)+ {caso haja resto da divisão anterior colocar = 1.2,8% caso não haja resto na divisão anterior colocar 0.2,8% ou seja taxa: 0,055+(((Quilômetros-12000)/600).0,028 + {1.0,028 0.0,028)
Grupo 6	km <=12000 = salário * 5,5/100 km > 12000 = 5.5/100 + ((km-12000)/600))*2.8/100 +2.8/100, se (km-12000)mod600<>0
Grupo 7	((km-12000)/600*2,8)+5,5
Grupo 8	salárioBruto*(5,5/100) = Prêmio, se quilometragem ≤ 12.000km salário*(5,5/100+((km-12000)div600+{se(km-12000)mod600>0, soma1}+2,8/100)=Prêmio {se(km-12000)mod600>0, soma0} se quilometragem > 12.000km
Grupo 9	Se km ≤ 12000 então taxa = 5,5% Se km > 12000 então taxa = 5,5% + 2,8% a cada 600km

questão 10

Faça a descrição de um algoritmo que represente um método de resolução do problema proposto (articule de forma adequada as relações matemáticas e lógicas obtidas nas questões anteriores). Utilize identificadores de variáveis que indiquem o significado, diante da proposta do problema, de cada informação representada.

Grupo 1

Assinatura
Objetivo: Identificar o valor do prêmio
Entrada: Salário bruto, total mensal de quilômetros – tipo float
Saída: Prêmio
Corpo

```
 Leia (salario); Leia (Km);
Se km ≤ 12000 salariototal=((salario)*0,055)+(salario)
Senão
  Se (Km > 12000)
    taxa1 = ((Km) - 12000) div 600;
    taxa2 = ((Km) - 12000) mod 600;
    taxa2 = taxa1 + taxa2 + taxa3;

  Se (taxa 2 ≠ 0)
    taxa1 = (km - 12000) div 600;
    taxa3 = 0,055;
    taxa1 = taxa1 * 0,028;
    taxa2 = 0,028;

  taxa = taxa1 + taxa2 + taxa3;
  salariototal = ((salario) * taxa) + salario;
Imprima (salariototal);
```

Grupo 2

Assinatura
Objetivo: calcular o salário final
Entrada: salário-real, quilometragem - inteiro
Saída: salário prêmio - real
Corpo

```
 Salário()
  Leia(salario); leia(quilometragem);
  salarioPremio=salario;
  se salario≥12000 então
  premioFixo =salario+(salario*0,055);
  premiovariavel=((quilometragem-12000)/600)*0,08
  intPremioVariavel=(int)premiovariavel),
  if(intPremioVariavel!=OK
  premioVariavel=premioVariavel+1;
  salarioPremio=premiofixo+premioVariavel;

  imprima(salarioPremio)
```

Grupo 3

Assinatura
Objetivo: OBTER O VALOR DA PREMIAÇÃO
Entrada: SALARIO (REAL), KM(INTEIRO)
Saída: PREMIO(REAL)
Corpo

```
PREMICAO()
LEIA(KM), LEIA(SALARIO);
SE KM ≤ 12000;
      ENTÃO    PREMIO←SALARIO*0,055;
      SENÃO    KMEXTRA←(KM-12000)DIV 600;
               KMEXTRA←KMEXTRA*2.8+5.5;
               RESTO←(KM-12000)MOD600;
             SE RESTO ≠ 0;
                ENTÃO  KMEXTRA←KMEXTRA+2.8;
                       PREMIO←SALARIO*(KMEXTRA*0.01);
                SENÃO  PREMIO←SALARIO*(KMEXTRA*0.01);
IMPRIMA(PREMIO);
```

Grupo 4

Assinatura
Objetivo: Calcular valor do Premio
Entrada: Salario Bruto, Quilometragem
Saída: Premio
Corpo

```
Premio()
Leia (Salario) Leia (Km)
se Km > 12000
  então
        SaldoKm←(Km-12000)
        PremioParcial←((SaldoKm/600)*2,8)+5,5
      Premiototal←((Saldokm mod600)*2,8)+Premio Parcial
      Premio←(Salario*(Premiototal)
      Salariototal←(Salario+Premio)
             Imprima(Salariototal(
  Senão Imprima(Salario)
```

Grupo 5

Assinatura
Objetivo: Calcular o valor de prêmio que o motorista receberá;
Entrada: Salário Bruto (SalarioBruto) e Quilometragem (QuilometrosRodados);
Saída: Valor do Prêmio (Premio).
Corpo

```
motorista()
leia(SalarioBruto); leia (QuilometrosRodados);
se QuilometrosRodados ≤ 12000
então
    Premio ← SalarioBruto*0,055;
senão
    DiferencaDeQuilometragem ←QuilometrosRodados – 12000;
    DivisaoDeQuilometragem ←DiferencadeQuilometragem/600;
    RestoDaDivisao←DiferencaDeQuilometragem MOD 600;
    se RestoDaDivisao > 0 então RestoDaDivisao ← 1;
                          senão RestoDaDivisao ← 0;
    Premio
    ←(0,055+((0,028*DivisaoDeQuilometragem)+(0,028*RestoDa
    Divisao)))*SalarioBruto;
Imprima (Premio);
```

Grupo 6

Assinatura
Objetivo: Determinar o valor do premio em relação com a quilometragem rodada
Entrada: km – tipo:inteiro , salBruto – tipo:real
Saída: premio – tipo:real
Corpo

```
Premio()
Leia (km) leia (SalBruto)

Se km ≤ 12.000 então prêmio ← salBruto*(5.5/100);

senão
    Parcextra ← (km – 12000) div 600;
    resto ← (km – 12000) mod 600;
    se resto <> 0 parcextra ← parcextra + 1;
    premio ← salBruto*(5.5/100)+parcextra*(2.8/100);

imprima (premio);

//Terceira Atividade – questão 11 – grupo 06
```

Grupo 7

Assinatura
Objetivo: Calcular o valor do prêmio
Entrada: salario bruto, km percorridos,
Saída: valorPremio
Corpo

```
Premio

   leia (salarioBruto)
   leia (km Percorrido)

   se kmPercorrido ≥ 12000
      se ((km-12000) % 600 > 0)
   então premio ← premio + ((km-12000)/600 + 1)*0,028
                  *salarioBruto;

    senão   premio = premio + ((km-12000)/600) * 0.028 *
            salarioBruto;

Imprima(premio);
```

Grupo 8

Assinatura
Objetivo: Obter o prêmio
Entrada: quilometragem (inteiro); salário (real);
Saída: prêmio (real)
Corpo

```
leia(km); leia(salário);
prêmio = salário * 0,055;
    se km >= 12000
    então

 se (km-12000)mod600>0
  então prêmio=prêmio+((km=12000)div600+1)*0,028*salário;

  senão prêmio=prêmio+((km-12000)div600)*0,028*salário;

imprima(prêmio);
```

Grupo 9

Assinatura
Objetivo: Calcular o valor do prêmio
Entrada: km(int), salarioBruto(real)
Saída: premio (real)
Corpo

```
Premio()
leia(km), leia (salarioBruto);
  se km ≤ 12000
     então  premio = salarioBruto * 0,055
  senão
     resto ←km-1200;
     resto ←restoDIV600;
     sobra ← restoMOD600;
   se sobra>0
     então premio←salarioBruto*(0,055+(resto+1)*0,028)
     senão premio←salarioBruto*(0,055+resto*0,028)
imprima(premio)
```

questão 11

Faça a implementação e testes do programa correspondente ao algoritmo que você construiu e depois envie (correio eletrônico) o texto do programa obtido para *cthomaz@fei.edu.br* .

Coloque no topo do programa uma linha com o seguinte conteúdo:

`// Terceira atividade – questão 11 – grupo ??`

Grupo 1

```
int main(int argc, char *argv[])
{   int km ;
    float salario , taxa1 , taxa2 , taxa3 , taxa ,salariototal;
    cout <<" Entre com a kilometragem: ";
    cin >> km;
    cout <<" Entre com o salario: ";  cin >> salario;
    if ( km <= 12000) {
         salariototal = (salario * 0.055) + salario;

         }else{
    if ( km > 12000) {
       taxa1 = ( km - 12000) / 600 ;
       taxa2 = ( km -12000) % 600;
       taxa3 = 0.055;
       taxa1 = taxa1 * 0.028;
       taxa = taxa1 +taxa2 + taxa3 ;
         }
    if ( taxa2 != 0) {
       taxa1 = ( km - 12000) / 600 ;
       taxa3 = 0.055;
       taxa1 = taxa1 * 0.028;
       taxa2 = 0.028;
         }
         taxa = taxa1 +taxa2 + taxa3;
         salariototal = (salario * taxa) + salario;
         }
         cout << "Salario total : " << salariototal << endl;
    system("PAUSE");
    return EXIT_SUCCESS;
}
```

Grupo 2

```
int main(int argc, char *argv[])
{
    float salario, salarioPremio, premioFixo, premioVariavel;
    int quilometragem;
    cout<<"Digite o salario: ";
    cin>>salario;
    cout<<"Digite a quilometragem: ";
    cin>>quilometragem;
    salarioPremio=salario;
    if (quilometragem>=12000){
        premioFixo=salario+(salario*0.055);
        premioVariavel=((quilometragem-12000)/600);
        if ((quilometragem-12000) % 600 != 0){
            premioVariavel=premioVariavel+1;
        }
        salarioPremio = premioFixo +
(salario*premioVariavel*0.028);
    }
    cout<<"O salario final sera: R$"<<salarioPremio<<endl;
    system("PAUSE");
    return EXIT_SUCCESS;
}
```

Grupo 3

```
int main(int argc, char *argv[])
{
    int KM;
    float salarioBruto, premio, resto, KMextra;

    cout << "Digite o valor do Salario Bruto: ";
    cin >> salarioBruto;
    cout << "Digite o KM percorrido: ";
    cin >> KM;

    if (KM <= 12000) {
           premio = salarioBruto * 0.055;
    }
    else {
           KMextra = (KM - 12000) / 600;
           KMextra = KMextra * 2.8 + 5.5;
           resto = (KM - 12000) % 600;
                  if (resto != 0) {
                            KMextra = KMextra + 2.8;
                            premio = salarioBruto * (KMextra *
0.01);
                  }
                  else {
                            premio = salarioBruto * (KMextra *
0.01);
                  }
    }

    cout << "O valor da Premiacao: " << premio << endl;

    system("PAUSE");
    return EXIT_SUCCESS;
}
```

Grupo 4

```
int main(){

   float salario, premio, salariototal, premioparcial;
   int km, saldokm;
   cout<<"salario?"; cin>>salario;
   cout<<"quilometros?; cin>>km;
   if(km>12000){
      saldokm=km-12000;
      premioparcial=((saldokm/600)*2.8)+5.5;
      premiototal=(saldokm%600)*2.8)+premioparcial;
      premio=salario*premiototal;
      salariototal=salario+premio;
      cout<<"resultado: "<<salariototal;
   }
   else {
     cout<<"resultado: "<<salario;
   }

   system("pause");
   return(0);


}
```

Grupo 5

```
int main(int argc, char *argv[])
{   int QuilometrosRodados, DiferencaDeQuilometragem,
RestoDaDivisao, DivisaoDeQuilometragem;
    float Premio, SalarioBruto, SalarioLiquido;
    cout << "Digite o Salario do Motorista: \n";
    cin >> SalarioBruto;
    cout << "Digite a quantidade de quilomentros rodados: \n";
    cin >> QuilometrosRodados;
    if (QuilometrosRodados <= 1200) {
        Premio = SalarioBruto * 0.055;
    }
    else {
          DiferencaDeQuilometragem  = QuilometrosRodados - 12000;
          DivisaoDeQuilometragem  = DiferencaDeQuilometragem /
600;
          RestoDaDivisao = DiferencaDeQuilometragem % 600;
        if (RestoDaDivisao > 0) {
                RestoDaDivisao = 1;
        } else {
                RestoDaDivisao = 0;
        }
      Premio = (0.055 + ((0.028 * DivisaoDeQuilometragem) + (0.028
* RestoDaDivisao))) * SalarioBruto;
    }

    cout << "O Premio que o Motorista recebera sera de: R$ " <<
Premio << endl;


    system("PAUSE");
    return EXIT_SUCCESS;
}
```

Grupo 6

```
int main(int argc, char *argv[])
{   int km, parcExtra, resto;
    float salarioBruto, premio;

    cout << "Entre com a quilometragem percorrida pelo
funcionario: ";
    cin >> km;
    cout << "Entre com o salario bruto do funcionario
: ";
    cin >> salarioBruto;

    if (km <= 12000) {
       premio = salarioBruto * (5.5 / 100);
    } else {
      parcExtra = (km - 12000) / 600;
      resto = (km - 12000) % 600;
      if (resto != 0) {
         parcExtra = parcExtra + 1;
      }
      premio = salarioBruto * ((5.5 / 100) + parcExtra * (2.8 /
100));
    }

    cout << "Valor do premio correspondente: " << premio << endl;

    system("PAUSE");
    return EXIT_SUCCESS;
}
```

Grupo 7

```
int main(int argc, char *argv[])
{
    int km;
    float salarioBruto, premio;
    cout << "Entre com KM: "; cin >> km;
    cout << "Entre com o salario bruto: "; cin >> salarioBruto;
    premio = salarioBruto * 0.055;
    if ( km >= 12000 ) {
        if (( km - 12000 ) % 600 > 0 ) {
            premio=premio+((km-12000)/600 + 1)*0.028*salarioBruto;
        }
        else {
            premio = premio+((km-12000)/600)*0.028*salarioBruto;
        }
    }
    cout << "\n\nPremio: " << premio << endl << endl;
    system("PAUSE");
    return EXIT_SUCCESS;
}
```

Grupo 8

```
int main(int argc, char *argv[])
{
    int km;
    float salario, premio;
    cout << "Entre com a quilometragem: "; cin >> km;
    cout << "Entre com o salario: "; cin >> salario;
    premio = salario * 0.055;
    if ( km >= 12000 ) {
        if (( km - 12000 ) % 600 > 0 ) {
            premio += ((km-12000)/600 + 1)*0.028*salario;
        }
        else {
            premio += ((km-12000)/600)*0.028*salario;
        }
        }
    cout << "\n\nPremio: " << premio << endl << endl;
    system("PAUSE"); return EXIT_SUCCESS;
}
```

Grupo 9

```
int main(int argc, char *argv[])
{   float salarioBruto, premio;
    int km, resto, mod;
    cout << "Entre com o salario bruto: ";
    cin >> salarioBruto;
    cout << "Entre com a quilometragem rodada: ";
    cin >> km;
    if (km <= 12000) {
        premio = salarioBruto * 0.055;
    } else {
        resto = km - 12000; resto = resto / 600;
        mod = resto % 600;
        if (mod > 0) {
            premio = salarioBruto * (0.055 + (resto + 1) * 0.028);
        } else {
            premio = salarioBruto * (0.055 + resto * 0.028);
          }
      }
    cout << "O valor do premio eh: R$" << premio << endl;
    system("PAUSE"); return EXIT_SUCCESS;
}
```

QUARTA ATIVIDADE

questão 1 Qual a finalidade das estruturas de controle de repetição em um algoritmo ou programa?	
Grupo 1	Executar os blocos de estruções repetidamente.
Grupo 2	Permitem a definição de fluxos de processamento repetitivos
Grupo 3	Repetir um bloco de instruções de acordo com o parâmetro definido. Exemplo: repita tal instrução até o x alcançar 0.
Grupo 4	Possibilitam a construção de algoritmos ou programas em que uma parte das instruções poderá ser executada por repetidas vezes
Grupo 5	Permitir a definição de fluxos de processamento repetitivos.
Grupo 6	Com a finalidade de possibilitar estruturas, em que os processos ou parte deles possam ser repetidos mais de uma vez.
Grupo 7	A capacidade de repetir, com rapidez e sempre da mesma maneira, uma sequência de ações por numerosas vezes.
Grupo 8	Repetir por várias vezes a mesma instrução da mesma forma e com rapidez
Grupo 9	Repetir determinado bloco de instrução para determinada condição.
Grupo 10	Executar diversas vezes um algoritmo enquanto uma condição é verdadeira.

questão 2

Observe a seqüência de instruções abaixo:

```
  ...
9.   int soma, parcela;
10.  soma=0; parcela=5;
11.  while(parcela<200){
12.     soma=soma+parcela;
13.     parcela=parcela*3;
14.  }
15.  cout<<"valor da parcela: "<<parcela<<endl;
16.  cout<<"valor da soma: "<<soma<<endl;
  ...
```

Quantas vezes será executada a linha 2 dessa seqüência de instruções?
Quantas vezes será executada a linha 4 dessa seqüência de instruções?
Quantas vezes será avaliada a expressão lógica de controle **`parcela<200`** ?
Que valores serão exibidos como resultados da execução?

Grupo 1	linha 2 linha 4 parcela<200 valores exibidos	1 vez só 4 vezes 5 vezes 5 20 65 200
Grupo 2	linha 2 linha 4 parcela<200 valores exibidos	1 unica vez 4 vezes 5 vezes soma=200 parcela=405
Grupo 3	linha 2 linha 4 parcela<200 valores exibidos	1 vez 4 vezes 5 vezes Parcela=405 Soma=200
Grupo 4	linha 2 linha 4 parcela<200 valores exibidos	1 vez 4 vezes 5 vezes valor da parcela = 405 valor da soma = 180
Grupo 5	linha 2 linha 4 parcela<200 valores exibidos	uma vez 4 vezes 5 vezes valor da parcela=405 valor da soma=200
Grupo 6	linha 2 linha 4 parcela<200 valores exibidos	1 (uma) vez enquanto o número de parcelas for <200 (menor que 200) ou seja 4 vezes 5 (cinco) vezes soma=200 parcela=405
Grupo 7	linha 2 linha 4 parcela<200 valores exibidos	1 vez 4 vezes 5 vezes 5, 15, 45, 135, 405
Grupo 8	linha 2 linha 4 parcela<200 valores exibidos	1 vez 4 vezes 5 vezes Parcela → 135 Soma → 200

Grupo 9	linha 2 linha 4 parcela<200 valores exibidos	1 vez → 1-15 → 2-45 → 3-135 →4-405 → sai do loop 4 vezes 5 vezes 1-5-15 2-20-45 3-65-135 4-200-405 SOMA = 200 PARCELA = 405
Grupo 10	linha 2 linha 4 parcela<200 valores exibidos	1 vez 3 vezes 4 vezes soma → 200 parcela → 405

questão 3 Se a quantidade de fichas disponíveis inicialmente é 100, quantas apostas o jogador pode fazer? Calcule sem utilizar o aplicativo e depois confronte sua resposta com o resultado fornecido pela execução do programa.	
Grupo 1	1ª 100-15=85 30x2=60 15x2=30 60-55=15 2ª 85-30=55
Grupo 2	Nossa resposta: 2 vezes Aplicativo: 2 vezes
Grupo 3	2 apostas e sobram 55 fichas
Grupo 4	Quantidade de apostas: 2 Quantidade Disponível – aposta 100-15=85 15x2=30 85-30=55 55-60 = -5 não é possível apostar mais. 30x2=60
Grupo 5	Aposta 15 30 60 Jogo 0 1 2 3 2 jogos Fichas 100 85 55
Grupo 6	100 fichas disponiveis – aposta inicial = 15 fichas 100-15 = 85 fichas disponiveis – 1ª rodada = 15x2 = 30 fichas 85-30 = 55 fichas disponiveis – 2ª rodada = 30x2 = 60 fichas 2 rodadas
Grupo 7	2 vezes
Grupo 8	2 vezes
Grupo 9	1 – 15 – 85 2 – 30 – 55 2 apostas X 3 – 60 - X
Grupo 10	100-15=85 85-30-55 2 apostas

questão 4 Qual seria a quantia mínima necessária se o desejo do apostador fosse realizar exatamente 6 apostas. Confronte seu resultado com a resposta gerada pelo aplicativo.	
Grupo 1	15x1=15 15+30+60+120+240+480=945 15x2=30 30x3=60 60x4=120 120x5=240 240x6=480
Grupo 2	945
Grupo 3	945 fichas
Grupo 4	Quantidade mínima de fichas: 945 15x2=30x2=60x2=120x2=240x2=480 1 2 3 4 5 6 15+30+60+120+240+480=945
Grupo 5	Aposta 15 30 60 120 240 480 Jogo 0 1 2 3 4 5 6 945 fichas Fichas X
Grupo 6	945 fichas
Grupo 7	15-30-60-120-240-480 945
Grupo 8	945 fichas
Grupo 9	1 – 15 2 – 30 3 – 60 4 – 120 5 – 240 6 – 480 945
Grupo 10	15+30+60+120+240+480 - 945

questão 5 Que modificações seriam necessárias no algoritmo se em vez de dobrar a quantidade da aposta a cada vez, o jogador aumentasse em 50 fichas a quantidade da aposta anterior?	
Grupo 1	(...) enquanto faposta <= fdisp faça fdisp ← fdips - faposta faposta ← faposta +50; qdt ← qdt +1; (...)
Grupo 2	faposta ← faposta + 50
Grupo 3	Substituir ficha = ficha * 2 por Ficha = ficha + 50
Grupo 4	Onde está a expressão: faposta ← 2*faposta , ficaria: faposta ← 50+faposta;
Grupo 5	No laço, alterar a linha de comando Faposta←2+faposta; para: faposta←50+faposta;
Grupo 6	Faposta ← faposta+50; ou seja, ao invés de multiplicar por dois o valor da aposta anterior, basta somar 50 a este valor.
Grupo 7	faposta = 2*faposta alterasse pra faposta = faposta + 50
Grupo 8	Alterar a linha 6 para: faposta ← 50 + faposta
Grupo 9	Linha 6: Trocar: faposta ← 2*faposta; Por : faposta ← faposta + 50;
Grupo 10	substituir faposta = 2*faposta por: faposta = faposta + 50;

questão 6

Complete o quadro, até a data 6, com as informações correspondentes:

Grupo 1	4	50	75	125
	5	175	275	450
	6	625	1000	1625
Grupo 2	4	50	75	125
	5	175	275	450
	6	625	1000	1625
Grupo 3	4	50	75	125
	5	175	275	450
	6	625	1000	1625
Grupo 4	4	50	75	125
	5	175	275	450
	6	625	1000	1625
Grupo 5	4	50	75	125
	5	175	275	450
	6	625	1000	1625
Grupo 6	4	50	75	125
	5	175	275	450
	6	625	1000	1.625
Grupo 7	4	50	75	125
	5	175	275	450
	6	625	1000	1625
Grupo 8	4	50	75	125
	5	175	275	450
	6	625	1000	1625
Grupo 9	4	50	75	125
	5	175	275	450
	6	625	1000	1625
Grupo 10	4	50	75	125
	5	175	275	450
	6	625	1000	1625

questão 7 Complete de forma adequada as lacunas dispostas nas afirmações abaixo com o emprego de algumas das expressões: *a quantidade* *o dobro da quantidade* *o triplo da quantidade* *o quádruplo da quantidade*	
Grupo 1	LACUNA 1 o dobro da quantidade LACUNA 2 a quantidade LACUNA 3 o triplo da quantidade LACUNA 4 a quantidade
Grupo 2	LACUNA 1 o dobro da quantidade LACUNA 2 a quantidade LACUNA 3 o triplo da quantidade exe. LACUNA 4 a quantidade txt.
Grupo 3	LACUNA 1 o dobro da quantidade LACUNA 2 a quantidade LACUNA 3 o triplo da quantidade LACUNA 4 a quantidade
Grupo 4	LACUNA 1 o dobro da quantidade LACUNA 2 a quantidade LACUNA 3 o triplo da quantidade LACUNA 4 a quantida
Grupo 5	LACUNA 1 o dobro da q LACUNA 2 a quantidade LACUNA 3 o triplo da quantidade LACUNA 4 a quantidade
Grupo 6	LACUNA 1 o dobro da quantidade LACUNA 2 a quantidade LACUNA 3 o triplo da quantidade LACUNA 4 a quantidade
Grupo 7	LACUNA 1 o dobro da quantidade LACUNA 2 a quantidade LACUNA 3 o triplo da quantidade LACUNA 4 a quantidade
Grupo 8	LACUNA 1 o dobro da quantidade LACUNA 2 a quantidade LACUNA 3 o triplo da quantidade LACUNA 4 a quantidade
Grupo 9	LACUNA 1 o dobro da quantidade LACUNA 2 a quantidade LACUNA 3 o triplo da quantidade LACUNA 4 a quantidade
Grupo 10	LACUNA 1 o dobro LACUNA 2 a quantidade LACUNA 3 o triplo LACUNA 4 a quantidade

questão 8

Se **txt** e **exe** representam as quantidades atuais de arquivos de texto e de arquivos executáveis contaminados e **txtant** e **exeant** as correspondentes quantidades na data anterior, complete as instruções de atribuição que estabelecem as relações entre essas quantidades:

Grupo 1	txt← 2*txtant+exeant; exe← 3*exeant+txtant;
Grupo 2	txt← txtant+txtant+exant exe← exant*3+txtant
Grupo 3	txt← txtant*2 + exeant*3; exe← exeant*3 + txtant;
Grupo 4	txt← txtant*2 + exeant; exe← exeant*3 + txtant;
Grupo 5	txt← (txtabt*2)+exeant exe← txtant+(exeant*3)
Grupo 6	txt← 2*txtant+exeant; exe← 3*exeant+txtant;
Grupo 7	txt← (2*txtant)+(exeant) exe← (3*exeant)+(txtant)
Grupo 8	txt← 2*txtant + exeant exe← 3*exeant + txtant
Grupo 9	txt← (txtant*2) + exeant; exe← (exeant*3) + txtant;
Grupo 10	txt← 2*txtant + exeant exe← 3*exeant + txtant

questão 9

Faça a descrição de um algoritmo que represente um método de resolução do problema proposto. Utilize identificadores de variáveis que indiquem o significado, diante da proposta do problema, de cada informação representada.

Grupo 1

Objetivo: Obter o número de dias necessários para que o número de arquivos infectados seja igual ao que o usuário digita.
Entrada: total de arquivos infectados – inteiro.
Saída: número de dias – inteiro.

Corpo

```
virus()

leia(arquivosInfectados); txt ← 1;
enquanto(total<arquivosInfectados)faça

        txt←2*txt+exe;
        exe←3*exe+txt;
        total←txt+exe;
        dias←dias+1;

Imprima(dias);
```

Grupo 2

Objetivo: Calculo quantade de dias
Entrada: q.arquivos, q.virtus txt, q.virus exe
Saída: quantidade de dias

Corpo

```
Leia(qArquivos); Leia(vírus.txt); Leia(vírusexe);

Total vírus ← vírustxtant+vírusexeant

Enquanto (totalvírus < arquivos)
    vírustxt ← vírustxtant*2+vírusexeant;
    vírusexe ← vírusexeant*3+vírustxtant;
    vírusexeant ← vírusexe;
    vírustxtant ← vírustxt;
    dias=dias+1
    totalvírus = virustxtant + vírusexeant;
imprima(totalvírus);
imprima(dias);
```

Grupo 3	Objetivo: obter a quantidade total de arquivos infectados
	Entrada: txt, exe, dataFinal – tipo inteiro
	Saída:text, exe, qtotal – tipo inteiro
	
	Corpo
	`virus()`
	`Leia(txt); Leia(exe); Leia(dataFinal);`
	`Enquanto data < dataFinal faça`
	`data = data+1; txtant=txt; exeant=exe;`
	`txt = txt*2+exeant;`
	`exe = txtant+3*exe;`
	`qtotal = txt+exe;`
	`imprima(txt);`
	`imprima(exe);`
	`imprima(qtotal);`

Grupo 4	Objetivo: Determinar a quantidade de dias para superar certa quantida de arquivos
	Entrada: Quantidade de arquivo (Arquivos) – tipo inteiro;
	Saída: Quantidade de dias para que os arquivos infectados superem o número total de arquivos (data) – tipo inteiro.
	
	Corpo
	`PropagaVirus()`
	
	`leia(Arquivos);`
	`DATA ← 0;`
	`TEXTO ← 1;`
	`EXECUTAVEL ← 0;`
	`FAÇA`
	`textoanterior←texto;`
	`executavelanterior←executavel;`
	`executável←(executavelanterior*3)+textoanterior;`
	`texto←(textoanterior*2)+executavelanterior;`
	`total←texto+executável;`
	`data←1+data;`
	`Enquanto (total ≤ arquivos);`
	`Imprima(data);`

Grupo 5

Objetivo: Calcular quantidade de dias total para contaminar todos os arquivos
Entrada:quantidade de arquivos
Saída: dias

Corpo

```
leia totalArquivos;
  arqioivosContaminados ← 1;
  dias ← 0;
  txt ← 1;
  textAntigo ← 1;
  exeAntigo ← 1;
   faça
    txt ← (txtAntigo*2)+exeAntigo;
    exe ← txtAntigo+(exeAntigo*3);
    dias ← dias+1;
    txtAngito ← text;
    exeAntigo ← exe;
   enquanto
    arquivosContaminados ≤ totalArquivos;

imprima dias;
```

Grupo 6

Objetivo: Obter quantidade de dias até a meta estipulada de arquivos infectados seja ultrapassada
Entrada: qtdearq (inteiro)
Saída: totaldias(inteiro)

Corpo

```
propaga_virus()
 leia(qtdearq);
 arqinfect ← 0; qtdedias ← 0;
 txt ← 1;
 exe ← 0;
 enquanto(arqinfect <= qtdearq)faça

   txtant ← txt;
   exeant ← exe;
   txt ← 2*txtant + exeant;
   exe ← 3*exeant + txtant;
   arqinfect ← txt + exe;
   qtdedias ← qtdedias + 1;

   Qtdedias ← qtdedias – 1;
 Imprima (qtdedias);
```

Grupo 7

Objetivo:
  dias para superar a quantidade total
Entrada:
  Quantidade de txt, exe;
Saída:
  dias

Corpo

```
leia(txt), leia(exe);
  txtant ← txt;
  exeant ← exe;

  txt ← txtant + exeant;
  exe ← 3*exeant + txtant;
  total ← txt + exe;
  dias ← dias + 1;
imprima – dias para superar quantidade total
```

Grupo 8

Objetivo: Determinar quantos dias a quantidade total de arquivos contaminados superará uma quantidade dada.
Entrada: QndeTotalArquivos (int)
Saída: QndeDias

Corpo

```
Vírus()

Leia (QndeTotalArquivos);
  txt ← 1; total ← txt
enquanto (total ≤ QndeTotalArquivos) faça
  txtant ← txt;
  exeAnt ← exe;
  txt ← 2*txtAnt + exeAnt;
  exe ← 3*exeAnt + txtAnt;
  total ← exe + txt;
  dias ← dias + 1;
imprima (dias)
```

Grupo 9

Objetivo: Descobrir o número de dias que o total de arquivos contaminados supera uma determinada quantidade
Entrada: Quantidade a ser superada
Saída: Número de dias para superar determinada quantidade

Corpo

```
Virus()

leia(qtd);
enquanto (total <= qtd) faça
    txt = (2*txtant) + exeant;
    exe = txtant + (3*exeant);
    total = exe + txt;
    txtant = txt;
    exeant = exe;
    dia = dia + 1;

Imprima(dia)
```

Grupo 10

Objetivo:
Entrada:
Saída:

Corpo

```
leia(totalDeterminado);
txtant ← 1; exeant ← 0; data ← 0; total ← 0;
enquanto (totalDeterminado > 0) então
    txt ← 2*txtant + exeant;
    exe ← 3*exeant + txtant;
    txtant ← txt; exeant ← exe;
    totalDeterminado = totalDeterminado-total;
    data = data + 1;

imprima (data);
```

questão 10

Faça a implementação e testes do programa correspondente ao algoritmo que você construiu e depois envie (correio eletrônico) o texto do programa obtido para *cthomaz@fei.edu.br* .

Coloque no topo do programa uma linha com o seguinte conteúdo:

// Quarta atividade – questão 10 – grupo ??

Grupo 1

```
int main(int argc, char *argv[])
{
    int arquivosInfectados, exe, txt, dias, total;
    dias=0; exe=0; total=0;
    cout << "Digite o numero de arquivos infectados: ";
    cin >> arquivosInfectados;
    txt = 1;

    while( total < arquivosInfectados){
         txt = 2 * txt + exe;
         exe = 2 * exe + txt;
         total = txt + exe;
         dias = dias + 1;
    }
    cout << "\nSerao necessarios " << dias << " para que a
quantidade de virus alcance a quantidade digitada." << endl;

    system("PAUSE");
    return EXIT_SUCCESS;
}
```

Grupo 2

```
int main(int argc, char *argv[])
{   int qArquivos, virusTxt, virusExe, totalVirus, dias,
virusTxtAnt, virusExeAnt;

    cout<<"Digite a quantidade de arquivos a ser superada: ";
    cin>> qArquivos;
    cout<<"Digite o valor de arquivos txt infectados: ";
    cin>> virusTxtAnt;
    cout<<"Digite a quantidade de arquivos exe infectados: ";
    cin>>virusExeAnt;

    totalVirus = virusTxtAnt + virusExeAnt;
    dias = 0;
    while (totalVirus < qArquivos){
          virusTxt =(virusTxtAnt*2 ) + virusExeAnt;
          virusExe = virusExeAnt * 3 + virusTxtAnt;
          virusTxtAnt = virusTxt;
          virusExeAnt = virusExe;
          totalVirus = virusTxtAnt + virusExeAnt;
          dias = dias + 1;
        }
        cout<<"Total de arquivos contaminados: " <<
totalVirus<<endl;
        cout<<"Total de dias: " <<dias<<endl;

    system("PAUSE");
    return EXIT_SUCCESS;
}
```

Grupo 3

```
int main(int argc, char *argv[])
{
    int txt,txtant,exe,exeant,datafinal,datainicial,total;

    cout << "Digite a quantidade de arquivos txt infectados:";
    cin >> txt;
    cout << "Digite a quantidade de arquivos exe infectados:";
    cin >> exe;
    cout << "Digite a data final de infeccao:";
    cin >> datafinal;
    datainicial = 0;

    while(datainicial < datafinal){
        datainicial = datainicial + 1;
        txtant = txt;
        exeant = exe;
        txt = txt * 2 + exeant;
        exe = txtant + 3 * exe;
    }

    total = txt + exe;

    cout << "O valor total de arquivos txt infectados em " <<
datainicial << " dias e de " << txt << endl;
    cout << "O valor total de arquivos exe infectados em " <<
datainicial << " dias e de " << exe << endl;
    cout << "O valor total de arquivos infectados em " <<
datainicial << " dias e de " << total << endl;
    system("PAUSE");
    return EXIT_SUCCESS;
}
```

Grupo 4

```
int main(int argc, char *argv[])
{
    int TextoAnterior, Texto, ExecutavelAnterior, Executavel,
Arquivos, Data, Total;

    cout << "Quantidade a ser superada: ";
    cin >> Arquivos;

    Data = 0;   Texto = 1;  Executavel = 0;

    do {
      TextoAnterior = Texto;
    ExecutavelAnterior = Executavel;
    Executavel = (ExecutavelAnterior * 3) + TextoAnterior;
    Texto = (TextoAnterior * 2) + ExecutavelAnterior;
    Total = Texto + Executavel;
    Data = 1 + Data;
    }
    while ( Total <= Arquivos);

    cout << "Quantidade de dias para superar os arquivos: " <<
Data << endl;
    system("PAUSE");
    return EXIT_SUCCESS;
}
```

Grupo 5

```
int main(int argc, char *argv[])
{int arquivosContaminados, dias, totalArquivos, txtAntigo,
exeAntigo, txt, exe;

    cout << "Entre com a quantidade de arquivos: ";
    cin >> totalArquivos;

    arquivosContaminados = 1;
    dias = 0;
    txt = 1;
    exe = 0;
    txtAntigo = 1;
    exeAntigo=1;

            do {
                txt = (txtAntigo * 2) + exeAntigo;
                exe = txtAntigo + (exeAntigo * 3);
                dias = dias + 1;
                arquivosContaminados = txt + exe;
                txtAntigo = txt;
                exeAntigo = exe;
                }
                while (arquivosContaminados <= totalArquivos);

    cout << "Dias para infectar todos os arquivos da estacao: " <<
dias << " dias"<< endl;


    system("PAUSE");
    return EXIT_SUCCESS;
}
```

Grupo 6

```
int main(int argc, char *argv[])
{   int qtdedias, arqtxt, arqexe, qtdearq, arqinfect, txtant,
exeant;

    cout << "Digite a quantidade total de arquivos: ";
    cin >> qtdearq;
    arqtxt = 1;
    arqexe = 0;
    qtdedias = 0;
    arqinfect = 0;

    while ( arqinfect <= qtdearq) {
          txtant = arqtxt;
          exeant = arqexe;
          arqtxt = 2*txtant + exeant;
          arqexe = 3*exeant + txtant;
          arqinfect = arqtxt + arqexe;
          qtdedias = qtdedias + 1;
    }
    qtdedias = qtdedias - 1;
    cout << "Quantidade de dias ate a meta de arquivos infectados
ser superada: " << qtdedias << endl;

    system("PAUSE");
    return EXIT_SUCCESS;
}
```

Grupo 7

Sem resposta

Grupo 8

```
int main(int argc, char *argv[])
{   int exe, exeAnt, txt, txtAnt, dias, quantTotal, total;
    cout << "Entre com a quantidade total a superar: ";
    cin >> quantTotal;
    txt = 1; exe = 0; total = txt; dias = 0;
    while (total <= quantTotal) {
        txtAnt = txt;
        exeAnt = exe;
        txt = 2 * txtAnt + exeAnt;
        exe = 3 * exeAnt + txtAnt;
        total = txt + exe;
        dias = dias + 1;
    }
    cout << "para superar quantidade total: " << dias << endl;
    system("PAUSE");
    return EXIT_SUCCESS;
}
```

Grupo 9

```
int main(int argc, char *argv[])
{   int dia, exe, exeant, txt, txtant, total, qtd;
        cout << "Quantidade a ser superada:";
        cin >> qtd;
        txtant = 1;
        total = 0;
        exeant = 0;
        dia = 0;
        while ( total <= qtd ){
              txt = (2 * txtant) + exeant;
              exe = txtant + (3 * exeant);
              total = exe+txt;
              txtant = txt;
              exeant = exe;
              dia = dia + 1;
        }
        cout << "Dias para susperar: " << dia << endl;
    system("PAUSE");
    return EXIT_SUCCESS;
}
```

Grupo 10

```
int main(int argc, char *argv[])
{
    int data, txt, exe, txtant, exeant, total, totalDeterminado ;
    cout << "entre de arquivos: ";
    cin >> totalDeterminado;
    txtant = 1; exeant = 0; data = 0; total = 0;
    while ( totalDeterminado > 0 ){
        txt = 2 * txtant + exeant;
        exe = 3 * exeant + txtant;
        txtant = txt;
        exeant = exe;
        total = txt + exe;
        totalDeterminado = totalDeterminado - total;
        data = data + 1;
    }
    cout << "Quantidade de dias: " << data << "\n\n";
    system("PAUSE");
    return EXIT_SUCCESS;
}
```

APÊNDICE C -

TRANSCRIÇÃO DAS ENTREVISTAS

PRIMEIRA ENTREVISTA

Professor: Então, eu queria que vocês falassem um pouco sobre a participação nas atividades. O que vocês acharam de participar das atividades?

Aluno 1: Das atividades de laboratório?

Professor: Isso, das atividades de laboratório.

Aluno 2: Ah, eu achei bom...

Aluno 1: Bom, tão bom que você pediu na última aula...

Aluno 2: Na verdade pra mim, eu consegui desenvolver mais na aula prática.

Aluno 1: É, porque você desenvolve a prática.

Aluno 2: É, na teoria é uma coisa, mas colocar o programa pra rodar é outra coisa, né?! Qualquer vírgula, qualquer ponto e o programa não funciona.

Aluno 1: Até porque lá na máquina a gente tinha uma interface lá, bonita. Quando você clicava lá, se interessa até mais, quando clicava em iniciar, ele fazia toda a continha e chegava no resultado. Certinho...

Aluno 2: Achei bom, quando a gente desenvolvia a teoria, ia lá e desenvolvia a prática.

Professor: Vocês têm alguma crítica ou sugestão sobre a forma como a gente propôs aquelas atividades? Vocês sentiram falta ou excesso de alguma coisa?

Aluno 2: Não, não senti falta ou excesso de alguma coisa, não. Mesmo porque eu não tinha nenhum trabalho na área, nenhuma noção de programação.

Professor: Foi coisa completamente nova pra vocês?

Aluno 1: Sim, completamente nova.

Aluno 1: Então se a gente achar que está faltando alguma coisa, a gente não sabe se está faltando alguma coisa, porque a gente é cru, é novidade.

Aluno 2: Mas deu para entender bem as atividades, o que ela realmente queria.

Professor: Bom, em todas elas a gente pediu uma leitura de texto, depois aquele trabalho com programa ilustrativo, o algoritmo pronto e depois vocês desenvolviam. A gente colocou como suporte mais um programa ilustrativo mas esse sem o algoritmo, vocês viram só o que seria a interface, com entradas e saídas. Que parte vocês acham que ajudou mais a aprendizagem de vocês?

Aluno 2: Das três atividades?

Professor: Isso, a leitura, o trabalho com aquele pronto ou a elaboração do algoritmo mesmo, que era sempre a última parte de cada atividade.

Aluno 1: Eu acho que elaborar o algoritmo é onde você aprende.

Aluno 2: Eu acho que o que ajudou mais foi elaborar, como a gente já teve uma base teórica nas aulas, o que mais ajudou foi elaborar, montar tudo...

Aluno 1: É, aí você vai ver o negócio funcionando e se entusiasma com ele funcionando. O que você fez tudo “ok”, ali.

Professor: Vocês se lembram de usar os aplicativos que ilustravam, aquele primeiro que tinha o algoritmo pronto, e mesmo o segundo que tinha só a interface. Vocês usaram esses recursos?

Aluno 2: Primeiro a gente montava o algoritmo depois a gente ia conferir. A gente montava, fazia e depois ia conferir nele.

Professor: Então nessa parte de confronto ele serviu bem.

Aluno 2: Acho que só no último, que a gente ia olhar nele primeiro.

Aluno 1: É, a gente quebrou a cabeça um pouquinho.

Aluno 2: Mas os outros três a gente montou...

Professor: E aquele que estava com o algoritmo pronto e a representação do mecanismo de funcionamento? Vocês pararam para olhar aquilo?

Aluno 2: Ah, sim, o pronto já, né?

Professor: Isso, tinha uma janela com o enunciado e do lado o algoritmo. Foi bom aquilo para vocês?

Aluno 2: Foi, naquele a gente colocava pra calcular, várias coisas. A gente gostou também, ficou observando. A gente olhou ele em todos, ficou analisando como montava.

Aluno 1: É a interface, só clicar em iniciar e via ele fazer tudo. Um negócio interessante.

Professor: A idéia desse aplicativo que a gente mostrava o algoritmo em funcionamento era produzir uma representação melhor e facilitar a compreensão de vocês sobre o mecanismo de funcionamento. Então no seqüencial aparecia lá, a primeira instrução e execução, depois a segunda...

Aluno 1: Exatamente, ia destacando as instruções.

Professor: Isso vocês acham que valeu?

Aluno 1: Valeu, ele vai destacando as etapas, até achar a resposta. Aquilo achei interessante, aquilo valeu. Até falei 'que legal isso aqui você clica num botão e ele vai passo-a-passo'.

Professor: E vocês acham interessante que a gente tivesse tido mais aplicativos desses nas aulas de teoria, fora do laboratório?

Aluno 1: Acho que sim...

Aluno 2: Eu também. Não sei assim, alguns alunos já conhecem programação, então pra eles tanto teoria quanto prática...

Aluno 1: Mas acho que até pra eles a prática é bom. Programador quando programa eles gostam de programar.

Aluno 2: É uma linguagem para ver as outras linguagens, pra gente assim que não conhece programação é interessante essa atividade.

Aluno 1: Interessante. Porque você sai das duas primeiras aulas e já vai pro laboratório com a cuca fresquinha, com a programação na cabeça. Já vai montar, última aula, e pronto. Você consegue desenvolver melhor.

Aluno 2: É bem diferente quando a gente tem só teoria, como na quarta-feira. Como agora, a gente viu os divisores, que o professor Cláudio deu. Aí ficou confuso assim. Na sexta-feira, como é mesmo que fazia? A gente montou tudo na sexta! Seria interessante ter mais aulas no laboratório com essas atividades.

Professor: Vou ver se consigo programar para a última semana, porque agora na verdade, a gente só tem mais duas sextas-feiras, essa semana, depois tem feriado, mais uma e depois começam as provas. Então vou ver se programo mais alguma para gente fazer em laboratório depois do feriado, vai ser dia 27. Ainda vou avisar a turma.

Aluno 1: É produtivo, sinceramente é muito produtivo.

Aluno 2: Eu gosto de fazer essas atividades. Nossa maior dificuldade não era montar o algoritmo, a gente montava mas depois pra conseguir fazer.

Aluno 1: Mas depois do algoritmo pronto a gente erra pouca coisa.

Aluno 2: Mas ainda erra, né.

Professor: Nessa época das atividades vocês sentiam ainda dificuldade em montar o algoritmo...

Aluno 2: Montar o programa, não o algoritmo.

Professor: Mais no programa do que no algoritmo?

Aluno 2: É, mais no programa.

Aluno 1: Eu acho ao contrário, mais difícil montar o algoritmo. O programa se ele dá um erro você quebra a cabeça e acha, mas o algoritmo se você errou ali você vai errar tudo.

Aluno 2: Eu montei alguns algoritmos lá e não consegui colocar pra funcionar. Estava faltando alguma coisa.

Professor: É, a tarefa mais complexa aí é montar o algoritmo. A expectativa é que com o tempo, a prática, passe a ser quase automática, a transcrição do algoritmo para o programa. Temos que manter o foco principal na elaboração do algoritmo. Nessas últimas aulas são um pouco mais delicadas essas implementações, já envolvem articulações um pouco diferentes, envolve aquela questão de passar parâmetros. Então aí o código do programa passa a ser um pouco mais difícil de definir, mas a complexidade maior continua sendo o algoritmo. Em relação aos problemas que a gente colocou naquelas atividades, tinha lá do motorista, embalagem das lâmpadas, do vírus...

Aluno 2: Da embalagem das lâmpadas não foi difícil. O mais difícil foi o da ferrovia, do carvão.

Professor: E vocês gostaram dessa coleção de problemas que a gente levou?

Aluno 1: Legal, porque envolveu até um cotidiano...

Aluno 2: Todos aqueles problemas existem aos milhares.

Aluno 1: Você pode usar em qualquer coisa aí fora.

Aluno 2: É, aquele do motorista, não tem só aquilo, nas empresas por aí tem hora extra, desconto, várias outras coisas. É mais complicado ainda.

Professor: Mas aqueles problemas foram significativos para vocês?

Aluno 2: Foram...

Aluno 1: Foram, foram. Você já imagina como vai utilizar isso, aquilo é um exemplo vamos dizer assim.

Aluno 2: Não é igual aos problemas da escola, né...

Aluno 1: Envolve tudo.

Professor: O que vocês enfrentarem maior dificuldade então foi no do transporte de carvão...

Aluno 1: É, do carvão.

Professor: No problema do vírus, a última atividade, vocês trabalharam legal?

Aluno 1: Do vírus foi mais ou menos aquele que a gente viu em sala de aula, dos coelhos.

Aluno 2: Isso, aí a gente usou os coelhos como base.

Aluno 1: Exatamente, a gente usou como base.

Aluno 2: Não foi tão difícil assim, a gente achou a diferença de um pro outro.

Professor: Por semelhança vocês conseguiram....

Aluno 2: A gente acabou achando uma coisa em comum entre eles.

Aluno 1: Então esse foi fácil, agora o mais difícil foi do carvão.

Aluno 2: É, a gente teve que olhar lá o programa...

Aluno 1: Uma espiadinha.

Aluno 2: Para montar o algoritmo a gente olhou lá pra ver como funcionava.

Professor: No problema do carvão vocês usaram mais intensamente aquele aplicativo que dava entrada e saída.

Aluno 2: Aham...

Professor: Bom, o que mais vocês querem falar? A gente tem ainda cinco minutinhos ...

Aluno 1: Acho que no momento assim não tem nada pra acrescentar, né...

Professor: Como foi o desempenho de vocês na prova?

Aluno 2: Eu achei que ia ser bem mal assim... Mas não foi não.

Professor: Quer dizer, você foi melhor do que...

Aluno 2: Eu esperava.

Aluno 1: Eu também.

Aluno 2: Não fui tão bem, mas fui melhor que eu esperava.

Aluno 1: Comigo aconteceu a mesma coisa.

Professor: Vocês viram a prova corrigida depois, não?

Aluno 1: A vista? Sim.

Professor: Vocês sentiram coerência entre o que fizemos nas aulas, nas atividades e aquilo que foi proposto na prova?

Aluno 2: Sim, inclusive pela última questão da prova, algoritmo.

Aluno 1: É, eu entendi o algoritmo, mas depois errei e...

Aluno 2: Tinha bastante coisa assim que a gente tinha visto na aula.

Aluno 1: Não os problemas, né. Mas a base...

Aluno 2: A base.

Professor: Voltando para as atividades. Aquela parte inicial das atividades que envolvia leitura, vocês lembram de ter feito a leitura?

Aluno 2: Lembro, lembro. Acho que a gente só não leu a que falava sobre variáveis, essa a gente acabou não lendo. Mas as outras a gente leu.

Professor: Para você valeu essa leitura?

Aluno 2: Sim.

Aluno 1: Mesmo porque as perguntas estavam baseadas na leitura.

Aluno 2: Tinha que ler para entender as questões.

Professor: Mais alguma coisa a falar?

Aluno 1: No momento não. Você tem mais alguma pergunta?

Aluno 2: Mais atividades assim...

Professor: Eu vou ver se consigo programar alguma coisa pra sexta-feira dia 27. Amanhã a gente faz mais algum exercício com subprogramas, mas pra amanhã não vou conseguir montar nenhuma atividade. Vocês viram que eu coloquei no *Moodle* o código do programa? Uma parte dos alunos não conseguiu anotar então coloquei no *Moodle*.

Aluno 2: Eu não consegui anotar. Depois eu vou ver.

Professor: Eu agradeço mais uma vez a boa vontade de vocês em participar desse trabalho que pra mim é muito importante, e eu sei que o tempo de vocês é precioso.

Aluno 1: Imagina, professor. Se precisar.

Professor: Bem, obrigado e até amanhã.

SEGUNDA ENTREVISTA

Professor: Bom, eu quero que vocês falem um pouco sobre a participação de vocês nas atividades. O que vocês acharam das atividades, se foi bom, se foi ruim, se foi produtivo.

ALUNO 1: Eu acho que foi bem produtivo. A parte que a gente conversou que é mais complicada é essa parte de descrever assim os parâmetros de contas, né. A parte de fórmulas, onde a gente mais apanhava para achar a forma ideal para a pessoa conseguir entender. A gente até sabia como era, mas pra colocar ali a gente sofria um pouco.

Professor: E aqueles aplicativos ajudaram a ... ?

ALUNO 1: Ajudaram bastante.

Professor: Como é que vocês usaram aqueles aplicativos?

ALUNO 1: Na verdade a gente via o algoritmo funcionando e desenvolvia a partir dali.

Professor: Como é que vocês foram na prova?

ALUNO 1: Fomos bem.

Professor: Que nota vocês tiraram?

ALUNO 1: Nove.

ALUNO 2: Nove e meio.

ALUNO 3: Nove.

Professor: Vocês tinham alguma experiência anterior em programação?

ALUNO 1: Não.

ALUNO 2: Não.

ALUNO 3: Não.

Professor: Aqueles aplicativos que ilustravam o funcionamento do algoritmo. Vocês tiveram alguma vantagem com aquilo? Vocês já tinham a idéia de como era o funcionamento, alguma visualização ... ?

ALUNO 2: Eu como leigo, é interessante porque ele não dava o valor direto, ia mostrando passo a passo e você conseguia enxergar como estava fazendo aquilo em cada parte. Conseguia visualizar ele fazendo a primeira conta...

ALUNO 3: Eu achei interessante.

Professor: E isso ajuda a entender melhor aquilo que está descrito no algoritmo?

ALUNO 1: Sim, sim.

Professor: Sobre os problemas que a gente levou naquelas atividades e também os problemas que a gente apresenta na sala de aula e nos laboratórios. O que vocês têm a dizer?

ALUNO 1: Interessantes, soluções simples implementadas pelos algoritmos. Bem legal.

Professor: Vocês acharam bem colocados ou tem problemas que acharam chatos ou algo assim? Que problema chamou atenção, que vocês se lembram?

ALUNO 2: O que chamou mais a atenção foi aquele que tinha que elaborar a equação. Chamava mais a atenção, era mais interessante de fazer.

Professor: O que você tinha que construir?

ALUNO 2: Isso.

Professor: Algum deles vem na cabeça de vocês assim, além desse?

ALUNO 2: O caso do vírus, o caso do motorista.

ALUNO 3: O que eu lembro assim era o de trigonometria, de calcular triângulos. É uma coisa que eu aprendi de uma maneira diferente de usar. Tem também o IPVA, o prêmio do caminhão.

Professor: Quer dizer, pra você foi bom então reencontrar o...

ALUNO 3: Foi, foi, até pra relembrar. No desenho técnico, o autocad, a gente não faz mais conta, mas é legal pensar no cara que fez o autocad, quanta conta ele teve que fazer para programar aquilo.

Professor: E vocês, se lembram de algum problema que...

ALUNO 1: Eu lembro daquele do caminhão, da quilometragem.

Professor: Aquele do motorista?

ALUNO 1: Isso.

Professor: E você?

ALUNO 2: Aquele da data, era legal.

Professor: O de transformar a representação da data ...

Professor: Vocês têm idéia da utilidade daqueles aplicativos que a gente levou na atividade, tanto aqueles que ilustravam o funcionamento do algoritmo, quanto os que mostravam a interface de entrada e saída? Vocês usaram pouco, muito?

ALUNO 3: Achei aquilo muito bom, usamos bastante. Ajuda a abrir a cabeça, pra ver e entender.

ALUNO 1: A gente comparava depois com o que tinha feito ..., se estava certo.

Professor: O uso principal, então, foi para fazer confrontos.

ALUNO 1: Isso.

Professor: Entre o que estava pronto e o que vocês tinham feito. E esse trabalho vocês acham que é bom de fazer, é bom ter aquela experiência.

ALUNO 2: Sim, por aí fica bom de conferir com aquele já pronto. Comparar o algoritmo com o já pronto.

ALUNO 3: É bem interessante, ver aquilo primeiro, perceber, ... para depois pensar na solução.

Professor: Tem justamente esses dois aspectos, facilitar o entendimento da proposta do problema, e esse segundo que é essa parte de confrontar e verificar se está construindo corretamente. Vocês têm alguma sugestão ou crítica a fazer, agora especificamente sobre as atividades? Algo que deveria ser acrescentado, reduzido ou retirado?

SILÊNCIO

Professor: O tempo disponível foi razoável? Vocês não tiveram que correr demais?

ALUNO 2: Não.

ALUNO 3: Teve um lá que agente correu mais, sofreu um pouco, no dia do IPVA.

Professor: Mas de maneira geral vocês não tiveram que acelerar demais?

ALUNO 2: Não.

Professor: E dos outros colegas da turma, o que vocês perceberam? Teve gente que não gostou ... ?

ALUNO 3: Em todas, o meu grupo foi bom. Todo mundo fez, falou, ... eu entendi e os outros também.

ALUNO 2: O meu (grupo) também, ninguém encostou. O grupo conversou, corrigiu quando estava errado. É bom fazer o programa, ver se funciona, a gente queria chegar logo lá.

Professor: Nas aulas de teoria a gente acaba implementando algumas poucas vezes...

ALUNO 1: A gente fazia e já sabia se é certo ou errado, olhando o outro pronto.

Professor: O que vocês mais querem falar. Sobre o curso de maneira mais geral. A gente praticamente já encerrou, temos só mais três últimas aulas. O que vocês acharam?

ALUNO 2: Para mim foi bom, muita coisa nova, é tudo novo, aprender os comandos e tal. Para mim é bem interessante.

ALUNO 3: Para mim também, é importante, precisa saber essas coisas de programação.

Professor: E nessa segunda parte do curso vocês continuam conseguindo acompanhar bem, trabalhar nos exercícios, estão conseguindo fazer?

ALUNO 1: Sim.

Professor: Alguma crítica que vocês queiram fazer ao curso ou ao material de uso?

Alunos: Não.

ALUNO 3: Acho que só essa parte de não ter mais interação com a máquina, de assistir muito o professor fazer. Organizar as aulas como as atividades pra gente poder ver aquilo que está implementando no mesmo dia. Para mim funciona bem. A teoria e já ir implementando.

Professor: Aquela parte inicial das atividades, foram duas ou três páginas de leitura. Vocês fizeram essas leituras?

ALUNO 2: A gente lia, mas não falava muito. Depois tinha que voltar de vez em quando, olhar outra vez.

Professor: Bom, eu quero outra vez agradecer a boa vontade de vocês por terem participado das atividades e terem vindo falar hoje. É importante pro meu trabalho da pós-graduação e tem esse outro aspecto que eu acho também importante, na organização da disciplina, trazer elementos novos. Só tenho a agradecer pela boa vontade e disposição de vocês.

ALUNO 1: Aproveitando a oportunidade professor. Qual é o objetivo das atividades?

Professor: O foco é o processo de aprendizagem, quero entender melhor como acontece a construção do conhecimento para a criação dos algoritmos. Um dos aspectos que eu presto mais atenção é a noção do mecanismo dinâmico que temos por trás do algoritmo, algo como você ter um papel com dez linhas no algoritmo, aquilo é uma coisa estática está escrito no papel, mas por trás daquilo você tem que ver e pensar um mecanismo dinâmico. Então é esse tipo de foco que eu tenho no trabalho, como acontece o progresso no aprendizado, nesse sentido. Como o aluno consegue construir o algoritmo. Em resumo é isso, não basta olhar o algoritmo como estático, a representação é estática, mas você tem que ter o dinamismo escondido naquele texto. Isso é o foco. Por isso trouxemos para as atividades esses aplicativos que representam os algoritmos, uma parte a gente tem no próprio ambiente de programação, mas não é tão evidente. Alguns aspectos a gente tem no sistema de programação, alguns ambientes tem ferramentas potentes, mas quando pensamos em algoritmos, não existe, vai fazendo em lápis e papel, é a situação mais natural. Mas você tem que ter na cabeça aquilo que está escrevendo no papel, pelo menos na sua cabeça tem que estar os mecanismos dinâmicos do algoritmo. Esse aspecto considero que é importante, no meu trabalho eu quero conseguir entender como isso acontece, o discurso do professor na sala de aula não é 100% entendido, não é todo mundo que forma a imagem do mecanismo, a partir do que expõe o professor em sala e nem daquilo que é lido. Em resumo é isso, buscar outros recursos para favorecer esse aspecto.

ALUNO 1: É legal, foi assim mesmo.

Professor: Já estamos todos atrasados, agradeço de novo pela disposição de vocês.