Introdução

Roland Teodorowitsch

Fundamentos de Programação - Escola Politécnica - PUCRS

16 de maio de 2023

“Computers are good at following instructions, but not at reading your mind.”
(Donald Knuth)

Sumário

Sumário

Programas de Computador

Programas de computador

- **computadores**
  - são máquinas usadas para automatizar a execução de tarefas
  - computadores podem executar instruções básicas de forma bastante simples
  - são máquinas de flexíveis e de propósito geral que podem ser usadas para diferentes atividades
  - são máquinas que armazenam dados (números, palavras, imagens), interagem com dispositivos (monitor, sistema de som, impressora) e executam **programas**
- **programa** (de computador)
  - permite que um computador desempenhe funções diferentes
  - é uma sequência de instruções/passos que diz ao computador como realizar determinada tarefa

Programas de computador

- o computador e os dispositivos periféricos são coletivamente chamados de **hardware**
- o programas que o computador executa são chamados de **software**
- **programação** corresponde à ação de projetar e implementar programas de computador
- aplicações complexas são formadas por um grande número de instruções simples engenhosamente combinadas - e para implementar tais aplicações um programador precisa trilhar um caminho que inicia com o entendimento de uma série de construções mais simples (o que é o objetivo desta disciplina)

A anatomia de um computador

A anatomia de um computador

- conhecer os blocos básicos que compõe um computador pode ajudar no processo de programação
- o **processador** ou **UCP** (Unidade Central de Processamento ou *CPU*) é o coração/cérebro do computador
  - é formado por um número bastante grande de elementos estruturais chamados **transistores**
  - localiza e executa as instruções dos programas
  - é capaz de realizar operações aritméticas, buscar dados na memória externa e salvar dados processados em unidades de armazenamento

A anatomia de um computador

- há 2 tipos de armazenamento
  - primário: circuitos de memória que mantêm a informação enquanto estão energizados (ou seja, são voláteis)
  - secundário: dispositivos como discos rígidos, que são mais lentos, porém são mais baratos, de maior capacidade e mantêm a informação de forma não volátil

A anatomia de um computador

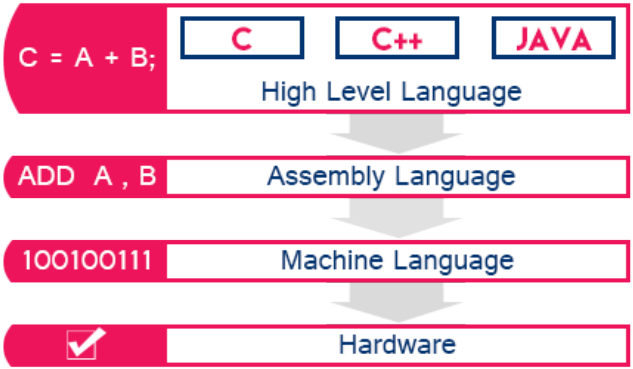
- o computador armazena tanto dados quanto programas
- eles são armazenados em dispositivos de armazenamento secundário e carregados para a memória quando forem executados
- um programa altera os dados na memória e salva os dados modificados no armazenamento secundário
- para interagir com usuários humanos, um computador necessita de **periféricos** ou **dispositivos**
- o computador envia dados para dispositivos de saída (*output*): monitor, auto-falantes, impressoras
- e obtém informação de dispositivos de entrada (*input*): teclado, mouse
- computadores podem ser conectados entre si, formando uma **rede de computadores** que permite a troca de dados entre computadores

A linguagem de programação Java

A linguagem de programação Java

- em vez de escrever um programa instrução por instrução, programadores usam **linguagens de alto nível**
- em uma linguagem de alto nível é possível usar ações que são traduzidas por um **compilador** para as instruções que o computador entende
- há também linguagens interpretadas, onde um **interpretador** lê as ações e as executa sem convertê-las para instruções do processador
- muitas linguagens têm sido desenvolvidas para as mais diferentes finalidades

Linguagens de alto nível, de montagem, de máquina



https://www.cs.mtsu.edu/ xyang/images/computer-languages.png

A linguagem de programação Java

- em 1991, um grupo liderado por James Gosling e Patrick Naughton na Sun Microsystems projetou uma linguagem (codinome *"Green"*) para ser usada em dispositivos de consumo, mas acabou sendo usada a partir de 1994 para escrever **applets** (código Java que pode ser obtido da Internet e executado localmente por um navegador)
- a linguagem passou a ser chamada de Java e cresceu muito desde então

A linguagem de programação Java

- Java possui uma **biblioteca** rica que permite escrever programas portáveis
- edições variando de *"micro"* a *"enterprise"* permitem escrever programas em Java para todas as necessidades (*smart cards*, telefones celulares, servidores)
- como foi projetada para a Internet, possui um foco forte na segurança e na portabilidade
- Java consegue ser portável pois não converte o código-fonte diretamente para instruções do processador, mas para **bytecodes** que são interpretados por uma **máquina virtual** Java

Familiarizando-se com o ambiente de programação

Familiarizando-se com o ambiente de programação

- há muitas maneiras de se programar em Java:
  - usar um **editor** simples (`notepad`, `vi`, `gedit`), chamando o compilador (`javac`) e a seguir executando o programa compilado (`java`)
  - usar um **IDE** (*Integrated Development Environment*), tal como BlueJ, Eclipse, NetBeans, etc.
- vamos começar com o modo mais “rústico”...

Usando a linha de comando

- use um editor qualquer para digitar o seguinte código:

```
public class MeuPrimeiroPrograma {
  public static void main (String[] args) {
    System.out.println("Este e' o meu primeiro programa...");
  }
}
```

- ATENÇÃO: mantenha as letras maiúsculas e minúsculas literalmente conforme o exemplo!

Usando a linha de comando

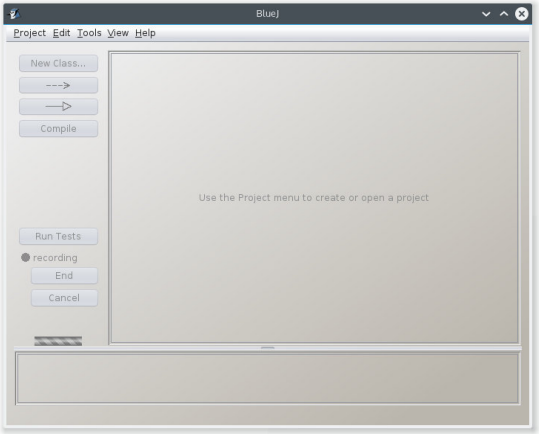
- abra um interpretador de comandos e vá para o diretório onde você salvou o arquivo
- salve-o com o nome `MeuPrimeiroPrograma.java`
- compile-o com `javac MeuPrimeiroPrograma.java`
- rode-o com `java MeuPrimeiroPrograma`
- certifique-se que o ambiente java esteja no PATH (caso não esteja, use o caminho completo até os arquivos `javac` e `java`)
- NÃO funcionou? NÃO se desespere o professor vai mostrar como fazer.

Usando o BlueJ

- agora com o...

Usando o BlueJ

- abra o BlueJ:



Usando o BlueJ

- selecione `Project` e `New Project`
- crie um diretório novo para o seu projeto (por exemplo: `MeuSegundoPrograma`) e selecione o botão `Create`
- use o botão `New Class` para criar uma nova classe (por exemplo: `MeuSegundoPrograma`)
- com o botão direito do mouse, clique duas vezes a caixa criada

Usando o BlueJ

- substitua o código por:

```
/*
 * Este e' um comentario de varias linhas
 * (podem ocorrer problemas com acentos...)
 */
public class MeuSegundoPrograma {
  public static void main (String[] args) {
    // Este e' um comentario de uma linha
    System.out.println("A cada dia aprendo coisas novas...");
  }
}
```

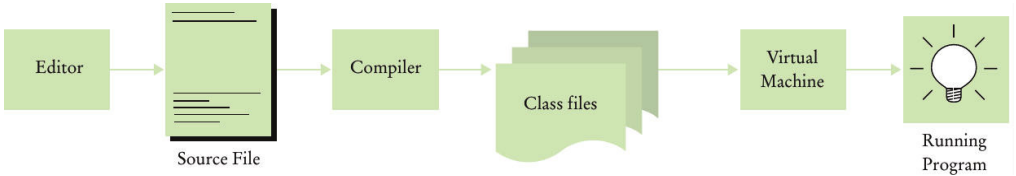
- pressione o botão `Compile` (preste atenção para verificar se não houve erros de digitação)

Usando o BlueJ

- pressione o botão `Close` da janela de edição da “classe” e volte para a janela do projeto
- abra o menu de opções clicando com o botão direito do *mouse* sobre a caixa da classe que foi criada, selecione a segunda opção: “`void main(String[] args)`” e, a seguir, pressione o botão `Ok`

Considerações

- `javac` converte o código-fonte em linguagem de alto nível em um arquivo `.class` (que contém instruções para a máquina virtual Java)
- `java` executa o arquivo `.class` na máquina virtual Java
- em uma IDE, tudo isto é feito de forma transparente



(HORSTMANN, 2013, p. 9)

Organize-se

- você pode aproveitar o *template* que o BlueJ gera ao se criar uma classe e usá-lo como padrão para seus futuros programas
- programas (em código-fonte ou já convertidos em instruções) são armazenados em arquivos e para manter estes arquivos bem organizados é importante criar uma hierarquia de diretórios (ou pastas)
- por exemplo, dentro do seu diretório de documentos crie um subdiretório para a universidade
- dentro do subdiretório da universidade, crie um subdiretório para cada disciplina
- dentro do subdiretório da disciplina, crie subdiretórios para apostilas, exercícios, trabalhos, etc.
- dentro do subdiretório de exercícios, crie um subdiretório para cada exercício

Faça cópias de segurança (*backups*)

- você vai perder muito tempo escrevendo programas, então é bom certificar-se de nada de errado vai acontecer com eles
- faça cópias de segurança (em *pen drives*, no *smartphone*, na Internet ou onde achar melhor)
- se for um projeto maior, mantenha um histórico de *b*ackups (por exemplo, com *backups* diários não sobrepostos) - pode-se também criar um mecanismo de rotação
- faça *backups* com frequência
- certifique-se de que o *backups* funcionou
- tome cuidado para que ninguém roube os dados do seu *backup*
- se ocorrer algum problema com seus arquivos, relaxe e recupere-o do *backup*

Não se assuste...

Java é uma linguagem poderosa e que, em um primeiro momento, pode parecer complicada. Por isso os conceitos, comandos e recursos da linguagem serão apresentados, explicados e utilizados de forma mais completa gradualmente.

Analisando seu primeiro programa

Meu Primeiro Programa

- este foi o primeiro programa que testamos

```
public class MeuPrimeiroPrograma {
  public static void main (String[] args) {
    System.out.println("Este e' o meu primeiro programa...");
  }
}
```

- na linha `public class MeuPrimeiroPrograma`:
  - `class` indica que estamos criando uma **classe** chamada `MeuPrimeiroPrograma`
  - um programa Java consiste de uma ou mais classes, que são os blocos de construção fundamentais dos programas Java
  - **`public`** indica que a classe poderá ser acessada por todos
  - o nome da classe deve corresponder ao nome do arquivo de código-fonte (então este arquivo deve ser chamado de `MeuPrimeiroPrograma.java`)

Meu Primeiro Programa

- os conteúdos de uma classe (ou de um método ou de um comando) são chamados de blocos e são delimitados por **{** e **}**
- dentro da classe, `public static void main(String[] args)`:
  - declara um **método** chamado `main`
  - um método contém um conjunto de instruções de programação que descrevem como executar determinada tarefa
  - toda aplicação Java deve conter um método `main`, sendo que a execução da aplicação começará exatamente por este método
  - como veremos mais adiante, uma aplicação poderá conter outros métodos além de `main`
  - o termo `static` será explicado na Unidade 8
  - da mesma forma, `String[] args` será explicado na Unidade 7

Meu Primeiro Programa

- o bloco de um método é composto fundamentalmente por um ou mais **comandos** terminados por `;`
- nunca esqueça de colocar um `;` depois de cada declaração, comando ou chamada
- os espaços, tabulações ou linhas em branco em um programa em Java são desconsiderados na compilação, então não se esqueça sempre o `;` corretamente
- os comandos de blocos em Java são executados um por um sequencialmente

Meu Primeiro Programa

- o único comando do método `main` do primeiro exemplo é `System.out.println("Este e' o meu primeiro programa...");`
  - este comando é uma chamada de um método da biblioteca Java que não precisa ser implementado
  - o nome do método é `System.out.println`
  - depois de uma chamada de método aparece sempre `(` e `)`
  - entre os parâmetros aparece, conforme o caso, uma lista de **parâmetros** ou **argumentos**, separados por vírgula, a serem enviados para o método
  - nesta chamada de método o parâmetro é `"Este e' o meu primeiro programa..."`, que é uma **cadeia de caracteres** (*string*, ou seja, um texto formado por caracteres)

Meu Primeiro Programa

- `println()` imprime e vai para o início da próxima linha, enquanto `print()` permanece na mesma linha
- primeiro experimente criar um método `main` com:

```
System.out.println("Uma linha...");
System.out.println("Depois outra...");
```

- depois experimente usar:

```
System.out.print("Uma so' linha ");
System.out.println("em duas partes...");
```

Estrutura básica para programas simples

- para nossos primeiros passos em Java usaremos a estrutura a seguir, colocando todas as declarações, comandos e chamadas dentro do método `main`

```
public class NomeDaClasse {
  public static void main(String[] args) {
    // ...
  }
}
```

Comentários

- comentários são trechos de texto que aparecem no meio do código-fonte Java e que não precisam seguir a sintaxe da linguagem
- são desconsiderados no processo de compilação
- mas facilitam o entendimento do código-fonte (pelo menos deveriam...)
- há dois tipos de comentários
  - os que iniciam com `//` vão apenas até o final da mesma linha e
  - os que podem iniciar numa linha com `/*` e terminar em outra linha com `*/`

Exercícios

Exercício 1

Crie um programa em Java que escreva o seu nome em letras maiores formadas pelo caracter $\star$.
Como, por exemplo:

```
****      ***    *         ***    *    *   ****
*    *   *    *  *        *    *  **   *  *    *
****    *    *  *        *****  * * *  *  *    *
*    *  *    *  *        *    *  *   ** *  *    *
*    *   ***   *****    *    *  *    *   ****
```

Exercício 2

Crie um programa em Java que imprima um rosto parecido (não precisa ser exatamente igual) ao rosto mostrado abaixo ou outra ilustração formada por caracteres de texto (você pode aproveitar alguma ilustração pronta obtida da Internet).
Sugestão:

```
  /////
 +"""""+
(| o o |)
 |  ^  |
 | `-' |
 +-----+
```

Erros

Erros

- em qualquer linguagem de programação há uma série de regras que devem ser seguidas para que um programa possa ser corretamente compilado ou interpretado
- caso você digite alguma chamada ou comando errado, por exemplo, o compilador não vai conseguir determinar o que você realmente queria que o seu programa fizesse e vai gerar um **erro**
- pode-se considerar 2 categorias de erros:
  - erros de compilação (*compile-time errors*)
    - erros de sintaxe (digitação, maiúsculas/minúsculas, pontuação; posição de comandos, correspondência de parenteses/colchetes; etc.)
    - não é gerado nenhum arquivo `.class`
    - corrige-se o primeiro erro listado e recompila-se o programa
  - erros de execução (*run-time errors*)
    - erros de lógica
    - o programa executa, mas produz um resultado indesejado
    - o programa pode ter a sua execução abortada

Erros de Compilação

- uma chamada a um método desconhecido, um comando escrito de forma errada ou, por exemplo, não usar `;` depois de um comando vão gerar **erros em tempo de compilação**, que são frequentemente chamados de `erros de sintaxe`
- nestes casos **NÃO** será gerado um arquivo `.class` com instruções para a máquina virtual Java
- o compilador tentará explicar para você o que está errado através de uma mensagem que indica o número da linha com erro (tente compreender o que está nesta mensagem!)
- é comum que depois de digitar um programa, principalmente quando se está iniciando os primeiros passos na programação em Java, que se tenha que passar por várias rodadas de correção de erros de sintaxe antes de conseguir compilar um programa com sucesso

Erros de Execução

- quando o programa compila, mas a execução não funciona como deveria, isto é chamado de **erro de execução** (algumas vezes chamados de **erros de lógica**)
- também é comum conseguir compilar um programa e ter que fazer ajustes até que ele funcione como deveria...
- alguns erros mais graves que ocorrem durante a execução podem gerar **exceções**: uma mensagem de erro da máquina virtual Java

Experimentando erros

Com o seu primeiro programa, faça as seguintes alterações, compilando e, se possível, tentando executar o programa gerado:

- troque `System.out.println` por `System.ou.println`
- use `system.out.println` (iniciando com minúscula)
- elimine a palavra `void` da declaração do método
- remova o caracter `;`
- remova um dos caracteres `}`
- substitua `System.out.println("Este e' o meu primeiro programa...");` por `System.out.println(1/0);`
- substitua `main` por `nain`

Solução de problemas: projeto de algoritmos

Solução de problemas: projeto de algoritmos

- **algoritmos** são planos detalhados que descrevem os passos para resolver um problema específico
- você já conhece alguns algoritmos
  - calcular a área de uma figura geométrica (círculo, quadrado, retângulo, etc.)
  - encontrar o comprimento da hipotenusa de um triângulo retâgulo
  - etc.
- alguns problemas são mais complexos e requerem mais passos
  - calcular o valor de Pi com determinado número de casas decimais
  - calcular a trajetória de um míssil
  - etc.

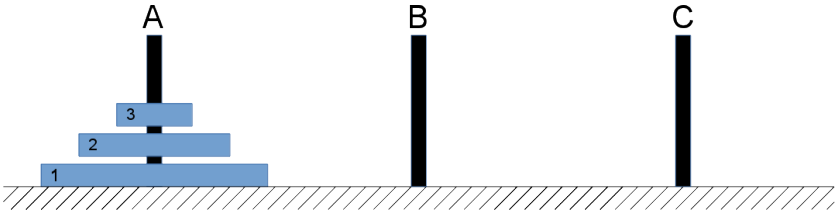
Algoritmos

- “Algoritmo é um conjunto finito de regras, bem definidas, para a solução de um problema em um tempo finito.” (ORTH, 2001)
- “Algoritmo é um texto (do tipo receita de bolo) onde cada linha contém uma ação primitiva (ação elementar passível de execução por um humano ou uma máquina). A função do algoritmo, quando executado, é a de agir (operar) sobre os dados, transformando-os em informações.” (PINTO, 1990)
- “An algorithm for solving a problem is a sequence of steps that is unambiguous, executable, and terminating.” (HORSTMANN, 2013, p. 19)
- A existência de um algoritmo é um pré-requisito essencial para programar uma tarefa.

“An algorithm must be seen to be believed.”
(Donald Knuth)

Desafio 1: Torre de Hanói

- No problema da Torre de Hanói há sempre três pinos (A, B, C)
- No pino mais à esquerda, há um determinado número de discos (1, 2, 3, ...) que devem ser transpostos para o pino mais à direita, seguindo-se as seguintes regras:
  - Só é possível mover um disco de cada vez
  - Um disco somente pode ser colocado sobre outro disco maior do que ele
- Escreva um algoritmo para resolver o problema da Torre de Hanói com o menor número possível de movimentos para 3 discos.



Torre de Hanói: Solução 1

- Mova o disco menor para o pino mais à direita
- Mova o disco médio para o pino central
- Mova o disco menor para o pino central (sobre o disco médio)
- Mova o disco maior para o pino mais à direita
- Mova o disco menor para o pino mais à esquerda
- Mova o disco médio para o pino mais à direita
- Mova o disco menor para o pino mais à direita

Torre de Hanói: Solução 2

- Considere:
  - os pinos A, B e C
  - os discos 1, 2 e 3
  - a operação mover(disco, pino)
- A solução consiste nos seguintes passos:
  - mover(3, C)
  - mover(2, B)
  - mover(3, B)
  - mover(1, C)
  - mover(3, A)
  - mover(2, C)
  - mover(3, C)

Desafio 2: Problema do Lobo, da Cabra e do Repolho

- Um homem que mora em um lado do rio deseja levar um lobo, uma cabra e um repolho para o outro lado do rio para vendê-los no vilarejo.
- Mas a sua canoa não é muito grande e ele terá que atravessar o rio levando apenas um item de cada vez.
- E ele não pode deixar a cabra sozinha com o repolho, pois a cabra comeria o repolho.
- Da mesma forma ele não pode deixar o lobo sozinho com a cabra, pois o lobo mataria a cabra.
- Ajude este homem a resolver este problema: escreva um algoritmo para resolvê-lo!

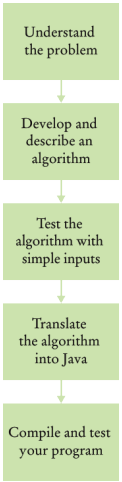
https://scientificgems.files.wordpress.com/2014/01/wgc.png

Para ajudar na solução de alguns problemas...

Sherlock Holmes — personagem de Arthur Conan Doyle

“Uma vez eliminado o impossível, o que restar, não importa o quão improvável, deve ser a verdade.”

Processo de Desenvolvimento de *Software* (HORSTMANN, 2013, p. 19)



Solução de problemas: projeto de algoritmos

Problema: Você colocou $10.000 numa conta bancária que recebe 5% de juros por ano. Quantos anos levará para que você tenha o dobro do valor original?

Como você pode solucionar isto?

- método manual: faça uma tabela como a tabela a seguir e acrescente linhas até alcançar o dobro do valor inicial

year	balance
0	10000
1	10000.00 x 1.05 = 10500.00
2	10500.00 x 1.05 = 11025.00
3	11025.00 x 1.05 = 11576.25
4	11576.25 x 1.05 = 12155.06

- use uma planilha: crie uma fórmula para cada linha e repita a fórmula para as demais linhas

Solução de problemas: projeto de algoritmos

- um programa de computador pode automatizar esta tarefa
- mas antes precisamos de uma descrição clara e inequívoca, que não necessite de nenhuma adivinhação para ser implementada
- o primeiro passo neste sentido é descrever a solução usando **pseudocódigo**

```
Inicie com um valor de ano igual a zero e um total de $10.000
Repita o seguinte enquanto o total seja menor do que $20.000
     Adicione 1 ao valor do ano
     Multiplique o total por 1,05 (crescimento de 5%)
Informe o último valor atribuído ao ano como resposta
```

Pseudocódigo

- consiste numa descrição informal de uma sequência de passos para resolver um problema
- trata-se de uma descrição que não pode ser entendida diretamente pelo computador, mas que pode ser facilmente convertida para alguma linguagem de programação
- é metade do caminho entre a linguagem natural e uma linguagem de programação
- deve descrever uma sequência de passos que é inequívoca (não ambígua), executável e finita

Pseudocódigo

Alguns exemplos de construções que aparecem em pseudocódigo são:

- obtenção de dados
  ```
  Leia as notas do aluno
  ```
- definições, atribuições e operações
  ```
  custo total = preço de compra + custo operacional
  Multiplique o subtotal por 1,05
  Remova o primeiro e o último caracter da palavra
  ```
- decisões e repetições
  ```
  Se custo total 1 < custo total 2
  Enquanto o total for menor do que $10.000
  Para cada imagem da sequência
  ```
- apresentação de resultados
  ```
  Escolha o carro 1
  Informe o último valor atribuído ao ano como resposta
  ```

Pseudocódigo

- usa-se indentação (tabulações ou deslocamentos horizontais para a direita) para indicar que comandos ou expressões devem ser selecionados ou repetidos

```
Para cada carro
    custo de operação = 10 x custo anual de combustível
    custo total = preço de compra + custo operacional
```

- aqui a indentação indica que ambas as expressões devem ser executadas para cada carro

Um problema

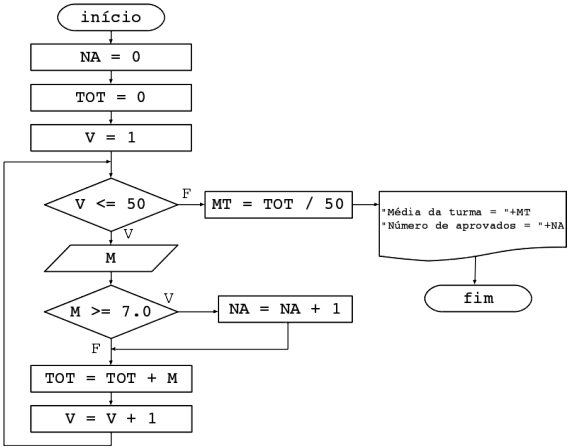
Ler a média de 50 alunos, calculando a média da turma e contando o número de alunos aprovados (com média maior ou igual a 7,0).

Solução 1: Algoritmo em Português Estruturado

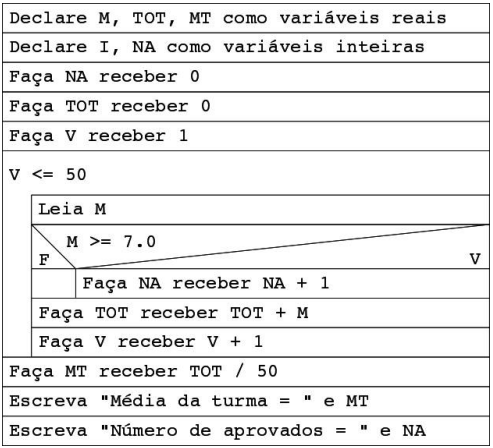
```
algoritmo media_de_50_alunos_com_aprovados;
início
    real:    M,     // média de um aluno
             TOT,   // acumulador
             MT;    // média da turma
    inteiro: V,     // variável de controle
             NA;    // número de aprovados

    NA = 0;
    TOT = 0;
    V = 1;
    enquanto (V <= 50) faça
    início
        leia(M);
        se (M >= 7.0) então NA = NA + 1
        TOT = TOT + M;
        V = V + 1;
    fim;
    MT = TOT / 50;
    escreva ("Média da turma = ",MT);
    escreva ("Número de aprovados = ",NA);
fim.
```

Solução 2: Fluxograma



Solução 3: Diagrama de Chapin



Exercícios

1. Mostre como seria a “implementação de” uma solução para resolução de equações do segundo grau, usando a fórmula de Bhaskara, no formato:
   - Fluxograma
   - Diagrama de Chapin
   - Algoritmo em Português Estruturado
2. Escreva um algoritmo para calcular e mostrar a soma dos dez primeiros números inteiros positivos: $1 + 2 + 3 + \ldots + 10$.
3. Escreva um algoritmo para calcular e mostrar o produto dos dez primeiros números inteiros positivos: $1 \times 2 \times 3 \times \ldots \times 10$.

Resposta do Exercício 1

```
algoritmo Bhaskara;
início
    real:    a,b,c,delta,x1,x2;

    leia(a,b,c);
    se (a==0) entao escreva ("Não é uma equacao do segundo grau!");
              senão início
                       delta = b*b - 4*a*c;
                       se (delta<0) então escreva ("Não tem raízes reais!");
                                    senão início
                                             se (delta==0) então início
                                                              x1 = -b/(2*a);
                                                              escreva (x1);
                                                           fim;
                                                  senão início
                                                              delta = sqrt(delta); // sqrt() = raiz quadrada
                                                              x1 = (-b+delta)/(2*a);
                                                              x2 = (-b-delta)/(2*a);
                                                              escreva (x1,x2);
                                                           fim;
                                             fim;
                  fim;
fim.
```

Resposta do Exercício 2

```
algoritmo Somatorio_dos_10_primeiros_inteiros_positivos;
início
    inteiro:    i,soma;

    soma = 0;
    i = 1;
    enquanto (i<=10) faça
    início
       soma = soma + i;
       i = i + 1;
    fim
    escreva (soma);
fim.
```

Resposta do Exercício 3

```
algoritmo Produtorio_dos_10_primeiros_inteiros_positivos;
início
    inteiro:    i,prod;

    prod = 1;
    i = 1;
    enquanto (i<=10) faça
    início
       prod = prod * i;
       i = i + 1;
    fim
    escreva (prod);
fim.
```

Referências

Referências

HORSTMANN, C. **Java for Everyone – Late Objects**. 2. ed. Hoboken: Wiley, 2013. xxxiv, 589 p.

ORTH, Afonso Inácio. **Algoritmos e Programação com Resumo das Linguagens PASCAL e C**. Porto Alegre: AIO, 2001. 176 p.

PINTO, Wilson Silva. **Introdução ao Desenvolvimento de Algoritmos e Estruturas de Dados**. São Paulo: Editora Érica. 1990. 204 p.