Algoritmos Computacionais

Aula 2 – Lógica de Programação

Prof. MSc. Odair Jacinto da Silva
odair.silva@metrocamp.edu.br

Grupo ibmec
METROCAMP
IBTA CAMPINAS

Lógica de Programação

- Uso correto dos processos de **raciocínio e simbolizações formais** de programação de computadores.
- Objetivo: resolver com qualidade os problemas que se deseja programar.
- O raciocínio lógico pode ser representado em **qualquer** linguagem de programação.

Algoritmos

- **Definição:** “um algoritmo é um procedimento consistindo de um **conjunto finito de regras não ambíguas que especificam uma seqüência finita de operações** necessárias à solução de um problema ou de uma classe de problemas.”

- Outra definição: seqüência de passos a serem seguidos para obter um resultado desejado.

Ambiguidade

- **Ambiguidade** é a **qualidade ou estado do que é ambíguo**, ou seja, **aquilo que pode ter mais de um sentido ou significado**. É aquilo que apresenta **indecisão, hesitação, imprecisão, incerteza, indeterminação.**
- A função da ambiguidade é sugerir significados diversos para uma mesma mensagem.
- Embora funcione como recurso estilístico, a ambiguidade também pode ser um vício de linguagem, que decorre da má colocação da palavra na frase. Nesse caso compromete o significado da frase.

Ambiguidade

- *Maria comeu um doce e sua irmã também*. (Maria comeu um doce, e sua irmã também).
- *Mataram o porco do meu tio.* (Mataram o porco que era do meu tio).
- *O guarda deteve o suspeito em sua casa*. (Na casa de quem: do guarda ou do suspeito?).

Exemplos

- **Exemplo:**
  - uma receita de bolo
  - programar a gravação de um filme com data e hora marcada.

Requisitos para a Execução Correta de um Algoritmo

- A descrição das ações de um algoritmo deve ser clara e precisa.
- As ações devem ser apresentadas na seqüência correta.
- A partir de um estado inicial, após um período de tempo finito, produzem um estado final previsível e bem definido.

Conclusão

- Um algoritmo deve fixar um padrão de comportamento a ser seguido com o objetivo de alcançar a solução de um problema, garantindo que sempre que executado, sob as mesmas condições, produza o mesmo resultado.

Importância

- Abstração dos detalhes computacionais (que podem ser acrescentados posteriormente);

- Construída uma solução algorítmica para um problema, **esta pode ser traduzida para qualquer** linguagem de programação.

Exemplos de Algoritmos Simples do Dia-a-Dia

Algoritmo para fritar um ovo

1. pegar frigideira, ovo, óleo e sal
2. colocar óleo na frigideira
3. acender o fogo
4. colocar a frigideira no fogo
5. esperar o óleo esquentar
6. colocar o ovo
7. retirar quando pronto

Estrutura Seqüencial

Exemplos de Algoritmos Simples do Dia-a-Dia

Algoritmo para trocar uma lâmpada

1. acionar o interruptor
2. **se** (a lâmpada não acender)

   pegar uma escada

   posicionar a escada embaixo da lâmpada

   buscar uma lâmpada nova

   subir na escada

   retirar a lâmpada queimada

   colocar a lâmpada nova

Estrutura Condicional Simples

Exemplos de Algoritmos Simples do Dia-a-Dia

Algoritmo para fazer uma prova

1. ler a prova
2. pegar a caneta
3. **enquanto** ((houver questão em branco) e (tempo não terminou)) faça
   - **se** (souber a questão)
     - resolvê-la
   - **senão**
     - pular para outra
4. entregar a prova

Estrutura de Repetição

Estrutura Condicional Composta

Importante!

- Não existe uma única solução correta.
- O bom senso e a prática de lógica de programação é que indicarão qual a solução mais adequada, que com menos esforço e maior objetividade produz o resultado desejado.

Representação de Algoritmos

Representação de Algoritmos

- Existem várias formas para representar o raciocínio lógico ou uma lógica de programação.
  - **Pseudocódigo;**
  - **Fluxograma;**
  - **Linguagem de Programação (por exemplo, Linguagem C).**

Representação de Algoritmos - Exemplos

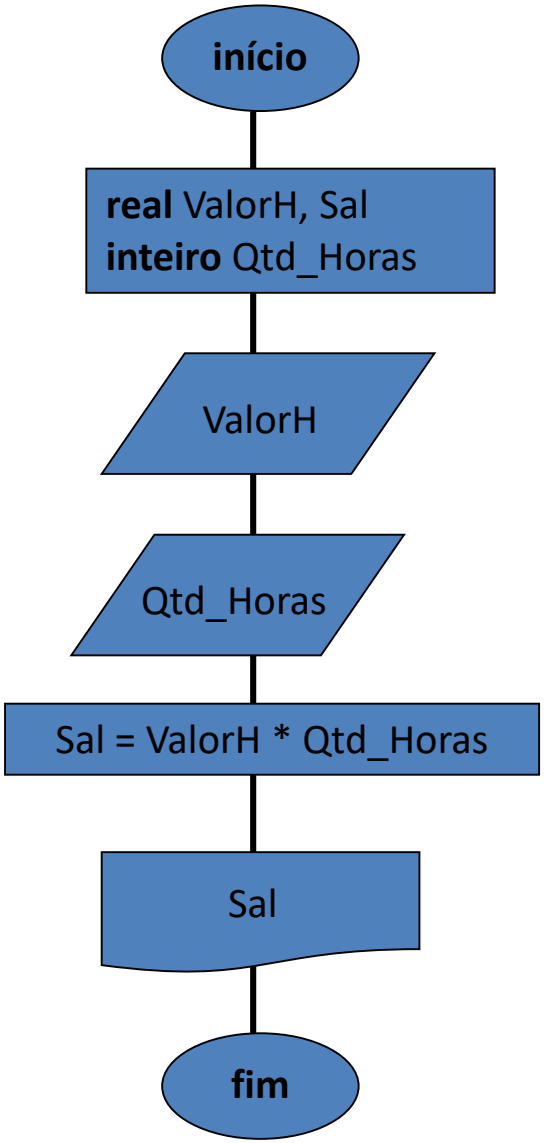
Desenvolver um algoritmo para calcular o salário bruto de um funcionário horista, sabendo a quantidade de horas (*Qtd_Horas*) trabalhadas e o valor da sua hora trabalho (*ValorH*).

Representação de Algoritmos - Exemplos

Pseudocódigo

```
Algoritmo Calculo_Salario
início
   real ValorH, Sal
   inteiro Qtd_Horas
      ler ValorH
      ler Qtd_Horas
      Sal = ValorH * Qtd_Horas
      mostrar Sal
fim
```

Fluxograma



Representação de Algoritmos - Exemplos

Linguagem C

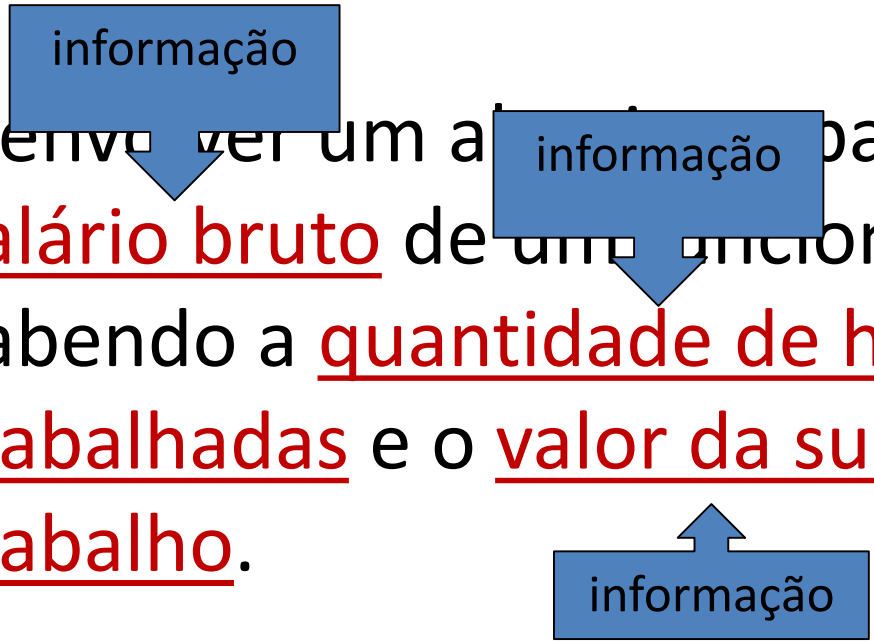
```
#include <stdio.h>
#include <stdlib.h>
int main() {
   float valorHora=0, salario=0;
   int qtdHoras=0;
   printf("Valor hora (R$): ");
   scanf("%f",&valorHora);
   printf("Quantidade de Horas Trabalhadas: ");
   scanf("%d",&qtdHoras);
   salario=valorHora*qtdHoras;
   printf("Salario Total R$%f",salario);
   system("pause");
   return 0;
}
```

Representação de Algoritmos

- Fazer uso do **Pseudocódigo** ou do **Fluxograma** ajuda bastante a organizar o raciocínio lógico para solucionar problemas computacionais.
- É preciso, portanto, seguir alguns **Itens Fundamentais** para **padronizar** um algoritmo (ou um programa), para que possamos treinar a solução de problemas computacionais e também desenvolvê-lo em qualquer linguagem de programação.

Itens Fundamentais para a Representação de Programas

Itens Fundamentais



Desenvolver um algoritmo para calcular o salário bruto de um funcionário horista, sabendo a quantidade de horas trabalhadas e o valor da sua hora trabalho.

Pergunta: como estas informações são armazenadas para serem manipuladas pelos programas?

Variáveis

- Uma variável é a representação simbólica dos dados envolvidos na solução de problemas computacionais.
- Cada variável corresponde a uma posição de memória do computador, cujo conteúdo pode variar ao longo do tempo de execução do programa.
- Embora uma variável possa assumir diferentes valores, ela só pode armazenar um valor a cada instante.
- Toda variável é reconhecida por um **identificador** e pelo seu **tipo**.

Variáveis - Identificador

- Um identificador é formado por um ou mais caracteres tal que:
  - o primeiro caracter deve ser, obrigatoriamente, uma letra e o demais podem ser letras ou dígitos;
  - não é permitido o uso de símbolos especiais na formação dos identificadores.

Variáveis - Identificador

Exemplos:

– Identificadores permitidos:

A Nota Nota_Trabalho1

– Identificadores não permitidos:

E(13) X-1 A:B 5B

– É recomendável que os nomes das variáveis sejam os mais significativos possíveis, isto é, que reflitam, da melhor maneira, a natureza dos valores que nelas estão sendo armazenados. Isto ajuda muito no entendimento do algoritmo.

Variáveis - Tipos

- Os dados (ou informações) manipulados em um programa computacional podem ser, inicialmente, de três **tipos**:

	Na Linguagem C
inteiro	**int**
real	**float**
caracter	**char**

Variáveis - Tipos

• **inteiro:** toda e qualquer informação numérica que pertença ao conjunto dos números inteiros relativos (negativa, nula ou positiva).

Exemplos:

3 0 -12 200

Variáveis - Tipos

• **real:** toda e qualquer informação numérica que pertença ao conjunto dos números reais (negativa, nula ou positiva).

Exemplos:

30.5 0.0 -1.25 1.80

Variáveis - Tipos

- **caracter:** toda e qualquer informação composta por um caracter alfanumérico:

  numéricos (0 .. 9), alfabéticos (A ... Z, a ... z) e especiais (por exemplo, #, ?, !, @, etc.)

  Exemplos:

  ‘A’ ‘1’ ‘b’ ‘@’

Observação: Um valor do tipo caracter é sempre representado entre aspas simples.

Declaração de Variáveis

- Cada variável só pode armazenar valores de um mesmo tipo.
- No momento em que se declara uma variável, é feita a associação do identificador com a respectiva posição de memória.
- Qualquer referência que se faça ao seu identificador implica na referência ao conteúdo do local da memória representada pelo mesmo.

Declaração de Variáveis

• Para indicar o tipo de uma ou mais variáveis é usada a **declaração de variáveis:**

Nome-do-tipo lista-de-variáveis;

Exemplos na Linguagem C:

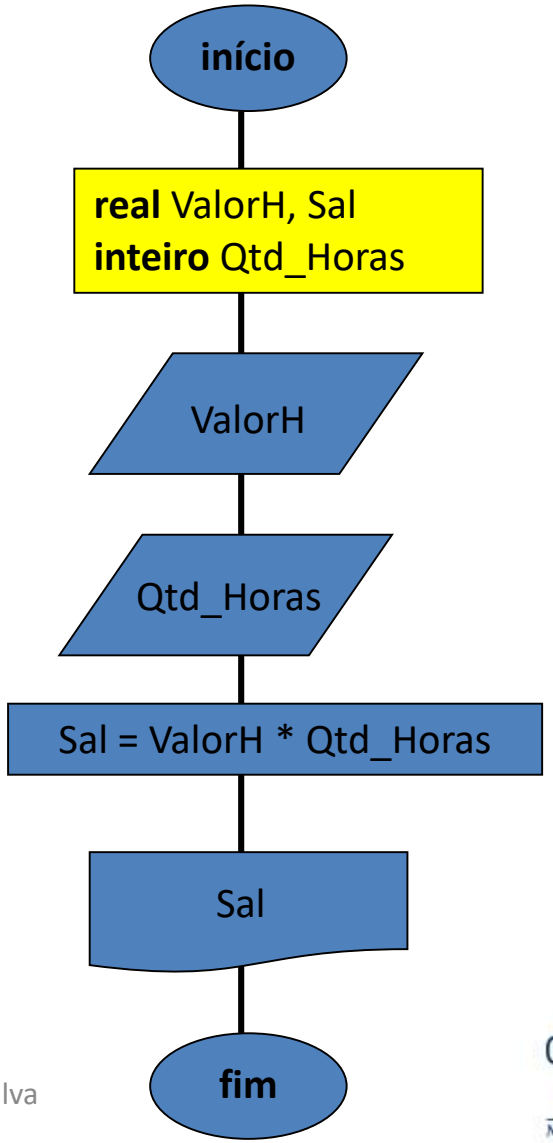
```
int   idade, Qtd_Horas;
float Salario, ValorH;
char  letra;
```

Declaração de Variáveis

Pseudocódigo

```
Algoritmo Calculo_Salario
início
   real ValorH, Sal
   inteiro Qtd_Horas
      ler ValorH
      ler Qtd_Horas
      Sal = ValorH * Qtd_Horas
      mostrar Sal
fim
```

Fluxograma



Representação de Algoritmos

Linguagem C

```
#include <stdio.h>
#include <stdlib.h>
int main() {
   //Declaração de variáveis
   float valorHora=0, salario=0;
   int qtdHoras=0;
   //Entrada de Dados
   printf("Valor hora (R$): ");
   scanf("%f",&valorHora);
   printf("Quantidade de Horas Trabalhadas: ");
   scanf("%d",&qtdHoras);
   //Processamento
   salario=valorHora*qtdHoras;
   //Saída - Apresentação dos resultados
   printf("Salario Total R$ %f",salario);
   system("pause");
   return 0;
}
```

Comandos de Entrada e Saída

- As unidades de entrada e saída são dispositivos que possibilitam a comunicação entre o usuário e o computador.

Comando de Entrada na Linguagem C

Comando de Entrada

• Sua finalidade é atribuir o dado (a informação) a ser fornecido pelo usuário à variável identificada.

scanf(<formato>,&<variável);

Exemplo:

```
scanf("%f",&valorHora);
scanf("%d",&qtdHoras);
```

Processamento - Comando de Atribuição

- Define-se **comando** como sendo a descrição de uma ação a ser executada em um dado momento.

- O **comando de atribuição** permite o armazenamento de um valor em uma certa variável (ou seja, em uma certa área de memória).

- O tipo deste valor tem de ser compatível com o tipo da variável na qual está sendo armazenado.

Processamento - Comando de Atribuição

Exemplo:

```
salario = valorHora*qtdHoras;
```

Processamento

Processamento - Comando de Atribuição

- **Forma geral:** identificador = expressão

onde:

identificador	é o nome da variável à qual está sendo atribuído um valor;
=	é o símbolo de atribuição;
expressão	pode ser um valor qualquer ou uma expressão aritmética.

Processamento - Comando de Atribuição

- **Exemplos na Linguagem C:**

```
int   A, B, C;
float  x,y;
A = 10;
B = 5;
C = A * B;
x = 5.5;
y = 5.5 / 2;
```

Comandos Saída na Linguagem C

- Para que o programa possa mostrar os dados que calculou, como resposta ao problema que resolveu, usa-se um comando de saída.
- Tem por finalidade exibir o conteúdo da variável identificada.

printf(<formato>,<variáveis>);

- Exemplo:

printf("%d", idade);

printf("Sua idade é %d", idade);

printf("Preço R$ %6.2f", preco);

Exercícios de Fixação

1. Fazer um fluxograma e seu programa que leia a base e a altura de um triângulo e imprima a área desse triângulo. Este cálculo pode ser feito com a seguinte fórmula: area = (base * altura)/2.
2. Fazer um algoritmo e seu programa que leia o saldo de uma aplicação e escreva o novo saldo, considerando um reajuste de 2%.
3. A conversão de graus Celsius para Farenheit pode ser obtida por: TF = 1.8 * TC + 32. Fazer um programa que leia um valor qualquer em graus Celsius e calcule e escreva o valor convertido em graus Farenheit. Apresente as mensagens adequadas de entrada e saída.
4. Faça um algoritmo que receba três notas de avaliações de um aluno e, em seguida, calcule e mostre a média aritmética destas notas.