Linguagem de Programação e Algoritmos

Prof. Me. Diogo Tavares da Silva

Aula 2 – Conceitos Fundamentais

1. Conceitos fundamentais de programação

Embora possam existir diversas linguagens de programação e paradigmas, alguns conceitos do projeto de algoritmos são fundamentais à qualquer linguagem utilizada. Dentre tais conceitos, podemos destacar:

- Tipos de dados
- Variáveis
- Atribuição
- Operações lógico-aritméticas
- Entrada e saída de dados

Analisaremos estes conceitos na aula de hoje...

1.1 O conceito de tipos de dados

- Mundo real --> Informações sobre fenômenos
    - "*Maria tem 6 anos de experiência*"
    - "*A área coberta é de 10,5 m^2*"
    - "*O nome da faculdade é "Universidade Paulista - UNIP""*
    - "O interruptor está acionado"
- Computacionalmente --> Dados observáveis que podem ser de determinados tipos.
    - 6 - INTEIRO
    - 10.5 - REAL ("Ponto flutuante")
    - "Universidade Paulista - UNIP" - LITERAL
    - Verdadeiro - LÓGICO

Neste contexto, podemos definir **Tipos de Dados** como quais são as principais formas de se definir, representar computacionalmente e se tratar os dados que devem ser computador.

Por que entender qual o tipo de dado estou manipulando é importante?

R: Por que eles definem quais são os tipos de dados suportados, como serão arranjados computacionalmente e qual é o conjunto de operações aplicáveis a esses dados.

1.1.1 Classificação dos tipos de dados

Basicamente podemos classificar os tipos de dados em "**tipos primitivos**" e "**Tipos não primitivos**".

- Tipos primitivos: Implementados nativamente pela linguagem
- Tipos não-primitivos: Estruturas mais complexas que são implementadas ou importadas por bibliotecas.

Podemos classificar ainda os tipos como "**Tipos simples**" e "**Tipos compostos**":

- Tipos simples: estruturados para armazenas apenas uma unidade de informação
- Tipos compostos: Estruturados para armazenar coleções de dados.

No presente momento, vamos analisar os tipos primitivos simples...

1.1.2 Tipos primitivos simples

Os tipos primitivos simples são aqueles oferecidos nativamente pela linguagem e em que estrutura armazena uma única unidade de informação.

Podem ser classificados de acordo com a natureza dos dados que armazenam:

- **Inteiro:** Armazena valores numéricos inteiros.
  - Tamanho: geralmente o mesmo da arquitetura (32 ou 64 bits).
  - Operações suportadas: Atribuição, Lógico-Aritméticas
- **Real:** Armazena valores numéricos decimais (“com vírgula”)
  - Tamanho: geralmente dois tamanhos:
    - ***float***: Metade dos bits representa o número (MANTISSA) e a outra metade o expoente de conversão em base 10.
      - Ex: 2.678 = 2378 * 10^-3
        - EXPOENTE: -3 MANTISSA: 2378
      - Por que? Números são representados em base binária. Como representar a vírgula?
    - ***double:*** Usa o tamanho de um inteiro para cada parte do número (dupla precisão)
  - Operações suportadas: Atribuição, Lógico-Aritméticas
- **Literal:** Armazena valores literais (caracteres) ou cadeias de caracteres (“*strings*”)
  - Tamanho: Depende da codificação de caracteres utilizada.
    - Caso ASCII: 8 bits por caractere
  - Operações suportadas: Atribuição, Lógicas, comparacionais, literais (“concatenação”).
- **Lógico:** Armazena um valor lógico booleano (VERDADEIRO) ou (FALSO)
  - Tamanho: 1 bit
  - Operações: lógicas

1.2 O conceito de Variável

“Uma variável é uma referência (rótulo) para um local da memória em que armazenamos um valor ou conjunto de valores que serão manipulados ao longo do programa”

Usada para armazenar valores que que serão utilizados ao longo do programa para dos devidos processamentos. Seu nome se dá ao fato que seu valor varia conforme é processado pelo algoritmo.

1.2.1 Declaração de variáveis

Em linguagem que possuem **tipagem estática** (o tipo da variável permanece o mesmo ao longo do programa) devemos declarar o tipo da variável.

Isso permitirá ao compilador reservar o espaço correto na memória para acomodar a variável e criar o contexto para saber quais operações a variável suporta.

Sintaxe da declaração em PORTUGOL:

```
DECLARE idade : INTEIRO
DECLARE velocidade : REAL
DECLARE nome : LITERAL
DECLARE ligado : LÓGICO
```

1.2.2 Dando nomes para variáveis

Deve-se utilizar nomes que se relacionam à função das variáveis no programa.

Padrões de projetos de nomes:

- snake_case
- camelCase
- PascalCase
- kebab-case

1.2.3 Constantes

Constantes, assim como variáveis, são rótulos para endereços de memória em que armazenamos dados. A diferença está no fato de que uma constante não

tem seu valor alterado ao longo do programa. Usamos basicamente para definir nome a valores

Sintaxe em PORTUGOL:

```
DEFINA PI 3.141516
```

1.3 A operação de Atribuição

Como dito, variáveis irão armazenar valores necessários para realizar os processamentos que transformarão entradas em saídas

Como o valor da variável irá variar ao longo do programa, precisamos de uma instrução que atualize seu valor. Esta instrução é chamada de atribuição

Sintaxe em PORTUGOL: `DECLARE A, B, SOMA : INTEIRO`

```
A <- 10
```

```
B <- 20
```

```
SOMA <- A+B
```

1.4 As operações lógico-aritméticas

1.4.1Aritméticas

Nome	Símbolo	O que faz
soma	+	
subtração	-	
multiplicação	*	
Divisão	/	
Resto de divisão	mod	Retorna o resto da divisão inteira

No operador **resto de divisão** colocamos o número dividido na frente e o divisor em seguida:

20/3 = 6 (divisão inteira)

20mod3 = 2

1.4.2 Comparacionais

Nome	Símbolo	O que faz
Maior que	>	
Menor que	<	
Maior ou igual	>=	
Menor ou igual	<=	
Igual a	==	
Diferente de	!= ou <>	

1.4.3 Lógicos

Nome	Símbolo	O que faz
NÃO	NÂO	A | NÃO A 0 | 1 1 | 0
OU	OU	A | B | A E B 0 | 0 | 0 0 | 1 | 1 1 | 0 | 1 1 | 1 | 1
NÂO	NÃO	A | B | A E B 0 | 0 | 0 0 | 1 | 0 1 | 0 | 0 1 | 1 | 1

1.5 Entrada e Saída de dados

O computador precisa interagir com o mundo externo para que possamos obter os dados que serão processados e devolver os resultados.

Para isso é necessário se utilizar de instruções de entrada e saída.

1.5.1 Entrada de dados

Utilizamos o comando de entrada para receber dados, geralmente pelo teclado

Sintaxe em PORTUGOL:

```
DECLARE A,B : INTEIRO
LEIA (A)
LEIA (B)
SOMA <– A+B
```

1.5.2 Saída de dados

Utilizamos o comando de saída para exibir dados, geralmente pelo monitor em formato literal.

Sintaxe em PORTUGOL:

```
DECLARE A,B : INTEIRO
LEIA (A)
LEIA (B)
SOMA <– A+B
ESCREVA(“O valor da soma é ”,SOMA)
```

1.6 Escrevendo nosso primeiro algoritmo

Uma empresa que constrói reservatórios industriais está tendo problemas para estimar a o custo e a quantidade de tinta necessária para pintar seus **reservatórios cilíndricos**, por isso, contratou um programador (no caso, você!) para escrever um algoritmo que gere esta estimativa.

- O programa deve receber as dimensões do reservatório, quantidade de litros por lata de tinta, rentabilidade da tinta (m^2 x litro) e o custo por lata.
- O programa deve calcular e retornar quantas latas são necessárias para pintar o reservatório e o custo total da pintura.
- Para fins de garantia das estimativas, deve-se sempre adicionar uma lata de tinta adicional para garantir a pintura completa e possíveis perdas

Análise –> Projeto -> Implementação