

Linguagem de Programação

Módulo 5



Universidade Federal do Rio Grande do Norte – UFRN

Escola Agrícola de Jundiaí – EAJ

Profa. Alessandra Mendes Pacheco

Algoritmo – Escopo

- Contexto **delimitante** aos quais valores e expressões estão associados.
- O escopo é utilizado para definir o **grau de ocultação da informação**, isto é, a visibilidade e acessibilidade às variáveis e aos dados em diferentes partes do algoritmo.
- A função *main* do C++, por exemplo, tem o seu escopo definido pelas chaves **{ }**.

```
1 /* Primeiro algoritmo do curso de Linguagem de Programação
2  Linguagem de programação C++
3  Data: 10/03/23
4  Profa Alessandra
5 */
6
7 #include<iostream>
8 using namespace std;
9
10
11 int main(){
12     cout << "Oi mundo!\n";
13     system("PAUSE");
14 }
15
16
17
```

//inicio do algoritmo

// escreva("oi mundo!")

// pausa na execução

Escopo da função *main()*

O escopo diz respeito à visibilidade!

Comando de Saída – Interação

- O comando ou operação de saída permite que o programa **forneça dados ao usuário** através de mensagens na tela.
- Também é utilizado para **requerer** que alguma ação específica seja executada exibindo uma solicitação na tela.
 - Os caracteres especiais “\n” e “\t” não são impressos. Eles adicionam à mensagem uma linha em branco ou um espaço de tabulação, respectivamente.

- Em pseudocódigo: `escreva ("Oi mundo!");`
- Em C++: `cout << "Oi Mundo!";`

```
1 #include<iostream>
2 using namespace std;
3
4
5 int main(){
6
7     cout << "Oi mundo!\n";
8     system("PAUSE");
9
10 }
11
```

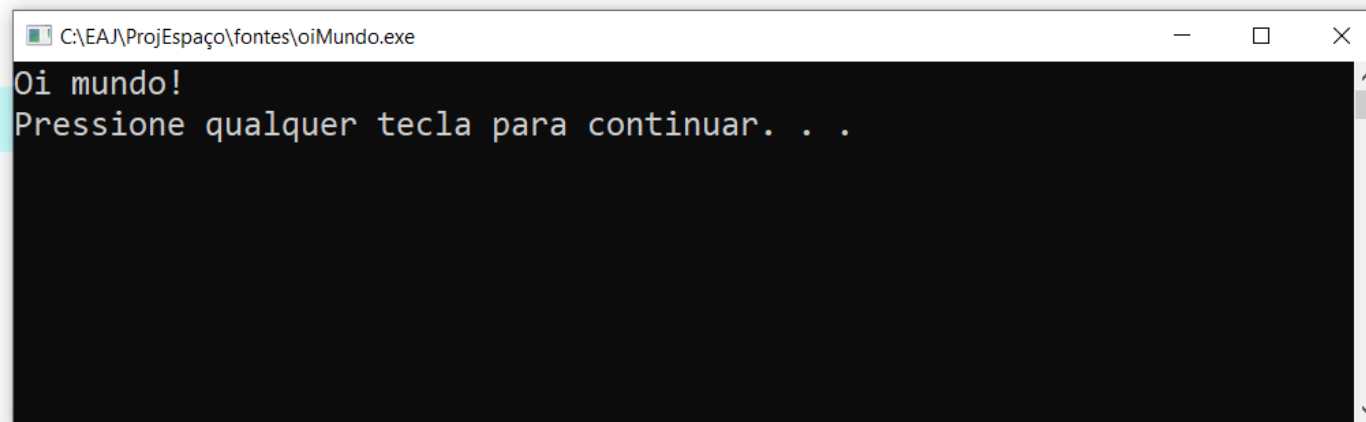
Comando de saída

Comando de Saída – Interação

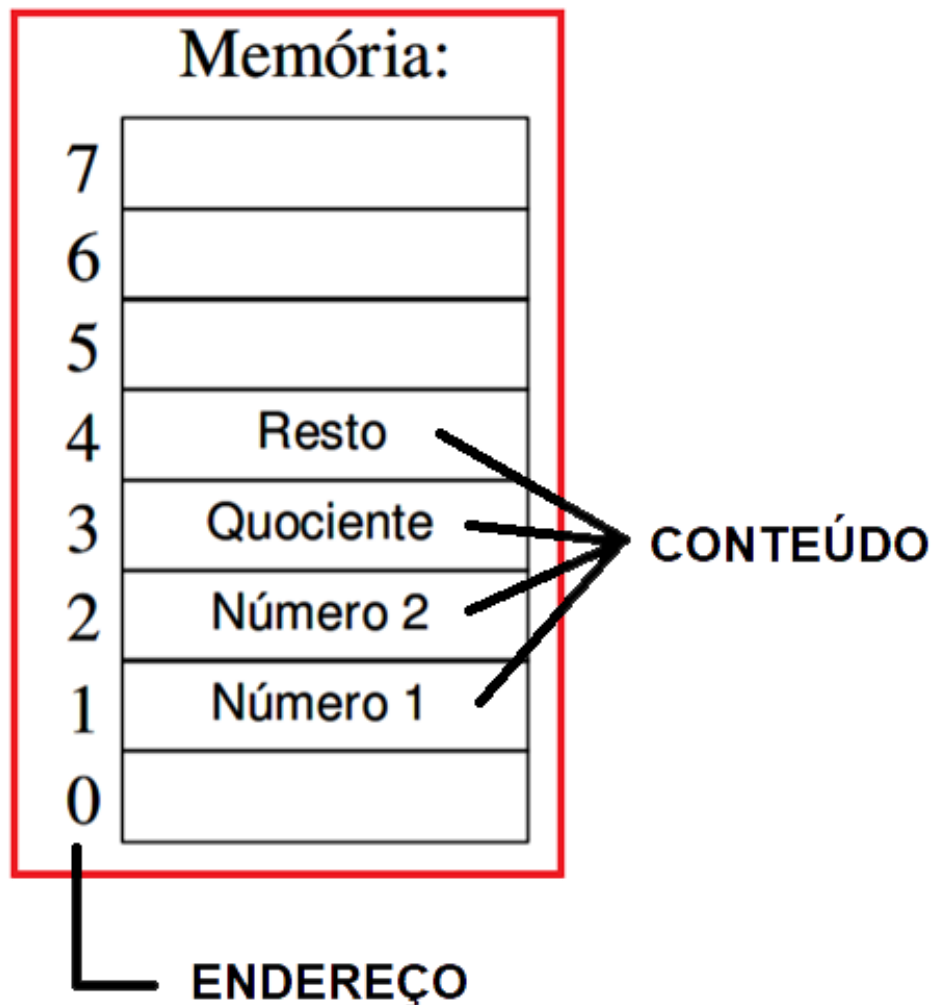
- Exemplo de algoritmo:

arquivo fonte: oiMundo.cpp

```
1 #include<iostream>
2 using namespace std;
3
4
5 int main(){
6
7     cout << "Oi mundo!\n";
8     system("PAUSE");
9
10 }
11
12
```



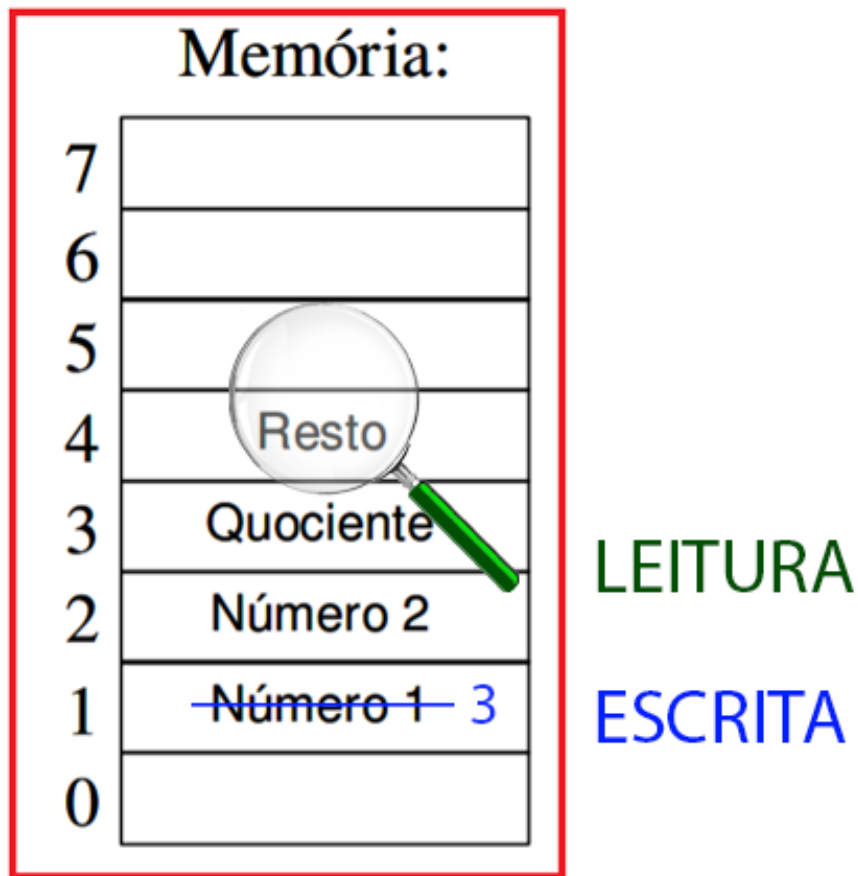
A Memória do Computador



- A memória de um computador é o local em que são **guardadas** as informações necessárias para execução de programas.
- A memória pode ser entendida como uma **sequência de células** nas quais se podem armazenar dados (**conteúdo**).
- Como as células estão ordenadas, atribui-se a cada uma delas um número (**endereço**) correspondente a sua posição na sequência.

Assim, **cada célula de memória é caracterizada pelo seu endereço e seu conteúdo.**

A Memória do Computador



- Esta memória é **volátil** (uma vez desligado o computador, as informações são perdidas).
- Os algoritmos utilizam esta memória para **salvar informações durante a sua execução**, ou seja, ela é fonte de armazenamento de dados.
- O computador **manipula as informações** contidas em sua memória através da leitura e da escrita.
 - A **leitura** localiza a célula correspondente ao endereço desejado e consulta o valor armazenado (o valor não é alterado).
 - A **escrita** localiza a célula correspondente ao endereço desejado e substitui o seu conteúdo pelo novo valor (o conteúdo anterior é perdido de forma irreversível).

Variáveis



~ MEMÓRIA Conjunto de locais para armazenar os dados.
Os dados podem ser armazenados na memória por meio das variáveis.

~ VARIÁVEL Local onde um dado específico é guardado.
Através do nome da variável, o seu dado pode ser acessado (leitura) e modificado (escrita) pelo algoritmo.

~ DADO Valor de um tipo específico que é armazenado em uma variável.

Variáveis

- São abstrações das células de memória.
- Uma variável possui **NOME**, **TIPO** e **CONTEÚDO**.
 - *As regras para nomes de variáveis mudam de uma linguagem para outra. Normalmente não podem começar por números, não podem ter letras que não pertençam ao alfabeto inglês, não podem ter espaços, não podem ser palavras reservadas da linguagem de programação e não podem possuir caracteres especiais (exceto “_”).*
- Variáveis devem ser **declaradas antes de serem utilizadas** e ao declarar uma variável, o computador reserva um espaço na memória para ela.
- Cada tipo de variável ocupa um tamanho diferente na memória.
- Exemplo de declaração de variável: `Tipo nome_da_variável;`
- Exemplo de declaração de variável em C++: `int idade;`

Variáveis

- O nome da variável é **único** em todo o algoritmo.
 - A variável “NOME” é diferente da variável “nome”.
- O **conteúdo** da variável deve ser **do mesmo tipo** usado na **declaração** da variável.
- O **uso** de uma variável (leitura) em uma expressão representa o seu conteúdo naquele momento.
- Na atribuição, o conteúdo da variável é **substituído** por outro (escrita).

Uma constante é uma “variável especial” pois também reservará um espaço de memória para o seu dado. Porém, uma constante armazenará um valor ÚNICO, que NÃO mudará durante o algoritmo.

Tipos Primitivos

- Definem o tipo de dado que poderá ser armazenado naquela variável.

Uma vez declarado o tipo, este não poderá ser modificado.

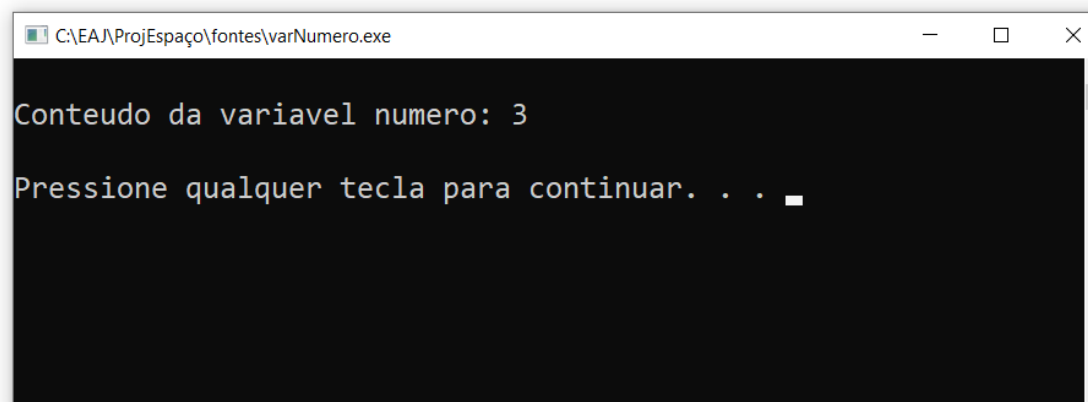
- Os tipos primitivos são tipos básicos implementados por todas as linguagens de programação.
- São tipos primitivos:
 - **Inteiro** (`int`): números positivos e negativos sem casas decimais;
 - **Real** (`float`): números positivos e negativos que possuem casas decimais;
 - *Obs: o tipo `double` é um `float` com maior precisão nas casas decimais*
 - **Caracter** (`char`): um único caracteres simples;
 - As sequencias de caracteres (textos) são *strings*.
 - **Lógico** (`bool`): verdadeiros (*true, 1*) ou falsos (*false, 0*).

Tipos Primitivos

- Exemplo de algoritmo:

arquivo fonte: varNumero.cpp

```
1 #include<iostream>
2 using namespace std;
3
4
5 int main(){
6     int numero;
7     numero = 3;
8
9     cout << "\nConteudo da variavel numero: " << numero << "\n\n";
10    system("PAUSE");
11
12 }
13
14
```



```
C:\EJA\ProjEspaço\fontes\varNumero.exe
Conteudo da variavel numero: 3
Pressione qualquer tecla para continuar. . . .
```

Comando de Entrada – Interação

- O comando ou operação de entrada permite que o usuário **forneça dados ao programa** através da digitação.
- O computador **armazenará** o dado lido na memória utilizando uma **variável**.
- O dado digitado deverá ser obrigatoriamente **do mesmo tipo** declarado para a variável que o armazenará.

```
1 #include<iostream>
2 using namespace std;
3
4
5 int main(){
6     int idade;
7
8     cout << "\nQual a sua idade? ";
9     cin >> idade;
10    cout << "\nA idade digitada foi " << idade << "\n\n";
11    system("PAUSE");
12
13 }
14
```

Comando de entrada

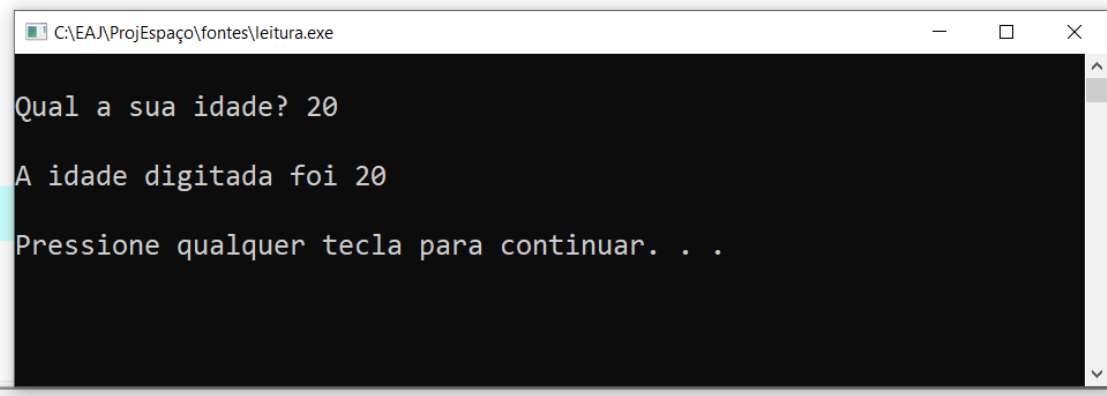
- Em pseudocódigo: `leia(idade);`
- Em C++: `cin >> idade;`

Comando de Entrada – Interação

- Exemplo de algoritmo:

arquivo fonte: leitura.cpp

```
1#include<iostream>
2using namespace std;
3
4
5int main(){
6    int idade;
7
8    cout << "\nQual a sua idade? ";
9    cin >> idade;
10   cout << "\nA idade digitada foi " << idade << "\n\n";
11   system("PAUSE");
12
13}
14
15
```



```
C:\EA\ProjEspaço\fontes\leitura.exe
Qual a sua idade? 20
A idade digitada foi 20
Pressione qualquer tecla para continuar. . .
```

Operador de Atribuição

- O operador de atribuição permite **inserir** ou **modificar** o valor (dado) de uma variável.
- Representado na maioria das linguagens de programação pelo símbolo de igualdade **=**.
- O valor absoluto ao lado direito da expressão, ou o resultado dela, é **atribuído** à variável localizada do lado esquerdo.

```
1 #include<iostream>
2 using namespace std;
3
4
5 int main(){
6     int numero;
7     numero = 3; Operador de atribuição
8
9     cout << "\nConteúdo da variavel numero: " << numero << "\n\n";
10    system("PAUSE");
11
12 }
13
```

- Em pseudocódigo: `numero ← 3;`
- Em C++: `numero = 3;` (leia-se número recebe 3)

Operadores e Expressões Aritméticas

- Os operadores aritméticos executam **operações matemáticas**, como adição e subtração, a partir dos seus operandos.
- Existem dois tipos de operadores aritméticos: **unários** e **binários**.
- Restrições:**
 - Os operadores +, -, *, / podem operar sobre números inteiros ou reais;
 - O operador % aceita apenas operandos inteiros;
 - O denominador em uma operação de divisão deve ser diferente de 0.

OPERADOR	OPERAÇÃO
+	ADIÇÃO
-	SUBTRAÇÃO
*	MULTIPLICAÇÃO
/	DIVISÃO
%	RESTO DA DIVISÃO INTEIRA
++	INCREMENTO
--	DECREMENTO

- Em pseudocódigo: `numero ← 3+4;`
- Em C++: `numero = 3+4;` (leia-se número recebe 3+4)

Operadores e Expressões Aritméticas

- Operadores unários de **incremento** (++) e **decremento** (--):
 - ++ soma 1 ao seu operando (incremento)
 - subtrai 1 do seu operando (decremento)
- Podem ser utilizados como **prefixo** (incremento ou decremento anterior) ou **sufixo** (incremento ou decremento posterior).
 - x++ : usa o valor de x e posteriormente o incrementa
 - ++x : incrementa o valor de x antes do uso do seu valor

OPERADOR	OPERAÇÃO
+	ADIÇÃO
-	SUBTRAÇÃO
*	MULTIPLICAÇÃO
/	DIVISÃO
%	RESTO DA DIVISÃO INTEIRA
++	INCREMENTO
--	DECREMENTO

Exemplos: `x=10; y = ++x; Resultado: x = 11 e y == 11`
 `x=10; y = x++; Resultado: x = 11 e y == 10`

Atividades Práticas

■ Reconhecendo algoritmos

- Sugerimos o acesso ao link <https://compute-it.toxicode.fr/> e a execução dos passos solicitados em cada nível para melhor compreensão dos conceitos (siga no máximo de níveis que conseguir!).
- Sugerimos ainda assistir aos vídeos “Como ensinar linguagem de programação para uma criança”, disponível no link <https://www.youtube.com/watch?v=pdhqwbUWf4U>, e “Pensamento Computacional Desplugado”, disponível no link <https://www.youtube.com/watch?v=Bxg8QC93joo>.

■ Construindo programas de computador que apresentem telas iniciais no console e interajam com o usuário

- Elabore um algoritmo que solicite o nome do usuário e apresente uma tela de boas vindas individualizada.



Dúvidas?

alemendes@gmail.com