

## MAC 2166 – Introdução à Computação

POLI - PRIMEIRO SEMESTRE DE 2007

Material Didático

Prof. Ronaldo Fumio Hashimoto

<p>ALGUNS PADRÕES DE PROGRAMAÇÃO: DECLARAÇÃO DE VARIÁVEIS, EXPRESSÃO ARITMÉTICA, EXPRESSÃO RELACIONAL, LEITURA, IMPRESSÃO, ATRIBUIÇÃO</p>
---

### Declaração de Variáveis

A declaração de uma variável que guarda números inteiros em C de nome `<nome_da_variavel>` é feita da seguinte forma:

```
int <nome_da_variavel>;
```

Exemplo: declaração de uma variável inteira "soma"

```
int soma;
```

Se você quiser declarar várias variáveis, é possível fazer da seguinte forma:

```
int <nome_da_variavel_1>, <nome_da_variavel_2>, <nome_da_variavel_3>, ..., <nome_da_variavel_n>;
```

Exemplo: declaração de duas variáveis inteiras "num" e "soma".

```
int num, soma;
```

### Expressão Aritmética

Expressões aritméticas são expressões matemáticas envolvendo números inteiros, variáveis inteiras, e os operadores "+" (soma), "-" (subtração), "/" (quociente de divisão inteira), "%" (resto de uma divisão inteira) e "\*" (multiplicação).

Exemplos:

- `num1 + num2 * 3`
- `num + 3 / 2`
- `num * 3 + 2`

Operador Aritmético	Associatividade
*, /, %	da esquerda para a direita
+, -	da esquerda para a direita

Tabela 1: Precedência dos Operadores Aritméticos.

## Precedência de Operadores

Qual seria o resultado da expressão:  $2 + 3 * 4$ ? Sua resposta provavelmente seria 14, pois é o resultado de  $2 + (3 * 4)$ , mas porque não 20, resultado de  $(2 + 3) * 4$ ? A resposta está na prioridade com que as operações são realizadas, ou precedência dos operadores. A operação "\*" tem maior precedência que a operação "+", e portanto é feita primeiro.

A Tabela 1 mostra a precedência dos operadores em C. Na dúvida, ou até para deixar mais claro e fácil de entender, use parênteses. Além de números as expressões podem conter o nome de variáveis, como na soma "num1 + num2".

Um outro fator importante é o tipo dos valores utilizados pelos operadores, no caso, estamos trabalhando apenas com o tipo inteiro (int). Isso é muito importante para entender o resultado de algumas expressões. Por exemplo, usando agora o compilador, faça um programa que imprima o valor da expressão  $(3 / 4 * 100)$ . O resultado é **zero**. Por quê?

Como a precedência de / e \* são iguais, a tabela diz também que esses operadores são calculados da esquerda para a direita, ou seja, o resultado de  $3/4$  é multiplicado por 100, e o resultado final esperado seria 75. Porém, o resultado do seu programa deve ter sido **zero**. Por que isso?

Como todas as operações são inteiras, o resultado de  $3/4$  é **zero** (e não 0.75, que é um número real). Sendo assim, o resultado de  $9/2$  é 4,  $9/3$  é 3,  $9/4$  é 2, e assim por diante. A parte fracionária é simplesmente eliminada (ou truncada ao invés de ser aproximada para um valor inteiro mais próximo), ou seja, mesmo o resultado de  $99999/100000$  é **zero**.

Considere as variáveis inteiras  $x = 2$  e  $y = 3$ . Verifique o valor das seguintes expressões:

Expressão	Valor
$x / y$	0
$y / x$	1
$y / x * 10$	10
$x + y * 4$	14
$(x + y) * 4$	20

## Expressão Relacional

Várias instruções dependem do resultado de comparações (ou condições) do tipo  $num1 > num2$  ( $num1$  é maior que  $num2$ ). O resultado de uma condição é **verdadeiro** ou **falso**.

Expressões relacionais são expressões que envolvem comparações simples envolvendo operadores relacionais "<" (menor), ">" (maior), "<=" (menor ou igual), ">=" (maior ou igual), "!=" (diferente), "==" (igual).

Uma comparação simples só pode ser feita entre pares de expressões aritméticas da forma:

`<expr_aritmética_01> <oper_relacional> <expr_aritmética_02>`

onde `<expr_aritmética_01>` e `<expr_aritmética_02>` são expressões aritméticas e `<oper_relacional>` é um operador relacional.

No decorrer do curso iremos aprender como fazer comparações mais complexas utilizando operadores lógicos. Vamos deixar este tópico para ser discutido mais adiante.

## Leitura pelo Teclado

A leitura de um número inteiro pelo teclado (fornecido pelo usuário) é feita usando a função `scanf` da seguinte forma:

```
scanf ("%d", &<nome_da_variavel>);
```

Exemplo: `scanf ("%d",&idade);`

É possível também ler dois ou mais números. Por exemplo,

```
scanf ("%d %d %d", &<nome_da_variavel_01>, &<nome_da_variavel_02>, &<nome_da_variavel_03>);
```

lê três números inteiros do teclado armazenando-os na variáveis `<nome_da_variavel_01>`, `<nome_da_variavel_02>` e `<nome_da_variavel_03>`. Observe que o `scanf` tem três `%d` e tem um `&` antes de cada variável.

Se você tem dúvida de como funciona isto, faça um programa simples que leia dois inteiros via teclado (com somente um `scanf`) e imprima sua soma.

## Impressão na Tela

A impressão de uma mensagem na tela é feita usando a função `printf`. A mensagem deve ser colocada entre aspas da seguinte forma:

```
printf ("<mensagem>");
```

Basicamente, a função `printf` imprime todos os caracteres que estão entre aspas, com exceção da seqüência de caracteres `"%d"` e `"\n"`.

Considere o exemplo:

```
printf ("Os numeros lidos foram %d e %d\n", num1, num2);
```

Para cada seqüência de caracteres `"%d"`, a função `printf` imprime na tela um número inteiro que é resultado das expressões aritméticas contidas no `printf` separadas por vírgula. Assim, o primeiro `"%d"` imprime na tela o conteúdo da variável `"num1"` e segundo `"%d"` imprime na tela o resultado da expressão `"num2"` (uma vez que a expressão com a variável `"num1"` vem antes da expressão com a variável `"num2"` no `printf` do exemplo acima).

Se você tem dúvidas, compile e execute o programa abaixo (para o Dev-C++):

```
1 # include <stdio.h>
2 # include <stdlib.h>
3
4 int main () {
5
6     /* declaracoes */
7
8     int num1, num2;
```

```

9
10 /* programa */
11
12 printf("Entre com dois numeros inteiros: ");
13 scanf("%d %d", &num1, &num2);
14
15 printf ("Os numeros lidos foram %d e %d\n", num1, num2);
16
17 /* fim do programa */
18
19 system ("pause");
20 return 0;
21 }

```

## Atribuição

Suponha que você queira guardar a soma dos dois números lidos do programa anterior em uma outra variável de nome *soma*. Para isso, devemos usar uma atribuição de variável. A atribuição de uma variável é uma operação que armazena o resultado de uma expressão aritmética (*expr\_arimética*) em uma variável (*nome\_da\_variável*) da seguinte forma:

$$\text{nome\_da\_variável} = \text{expr\_arimética};$$

Em uma atribuição, a variável (**SEMPRE UMA E UMA ÚNICA VARIÁVEL**) do lado **esquerdo** do símbolo = recebe o valor da expressão aritmética do lado **direito**.

Exemplos:

- $\text{soma} = \text{num1} + \text{num2};$
- $\text{z} = \text{x} / \text{y};$
- $\text{z} = \text{y} / \text{x};$
- $\text{z} = \text{y} / \text{x} * 10;$
- $\text{x} = \text{x} + \text{y} * 4;$
- $\text{y} = (\text{x} + \text{y}) * 4;$

## Atribuição e Comparação

Note a diferença entre o operador de atribuição = e o operador relacional ==.

1.  $\text{a}=\text{b}$
2.  $\text{a}==\text{b}$

O primeiro armazena o conteúdo da variável *b* na variável *a*. O segundo, com significado bem diferente, compara se o conteúdo da variável *a* é igual ao conteúdo da variável *b*.

## Um Programa para Testar

Se você tem dúvidas, compile e execute o programa abaixo (para o Dev-C++):

```
1 # include <stdio.h>
2 # include <stdlib.h>
3
4 int main () {
5
6     /* declaracoes */
7
8     int num1, num2, soma;
9
10    /* programa */
11
12    printf("Entre com dois numeros inteiros: ");
13    scanf("%d %d", &num1, &num2);
14
15    printf ("Os numeros lidos foram %d e %d\n", num1, num2);
16
17    soma = num1 + num2;
18
19    printf("O resultado da soma de %d com %d eh igual a %d\n", num1, num2, soma);
20
21    /* fim do programa */
22
23    system ("pause");
24    return 0;
25 }
```

## Dúvidas

Dúvidas deste material pode ser enviadas para o “Fórum para assuntos específicos da turma WEB”.