Linguagem de Programação I

- Operadores -

Prof. Ulysses Santos Sousa

ulyssessousa@ifma.edu.br

Aula 02

Roteiro

- **Operadores**
  - Aritméticos
  - Endereço
  - Relacionais
- **Função *scanf()***
- **Quantificador *const***
- **Conversões de tipo**
- **Funções *getche()*, *getch()*, *getchar()* e *putchar()***
- **Operadores de Incremento e Decremento**
- **Operadores Aritméticos de Atribuição**
- **Operadores Lógicos**
- **Operador Condicional Ternário**

Operadores

- **A Linguagem C possui vários operadores sendo divido em quatro classes:**
  - Aritméticos;
  - Relacionais;
  - Lógicos;
  - Bit a bit.
- **Além desses, C tem alguns operadores especiais para tarefas particulares.**

Operadores

- **Operador de atribuição**
  - É representado pelo sinal **= *(igual)*;**
  - Forma geral:
    - **nome_da_variavel = expressão;**
  - Exemplo:
    - ***x* = 3;**
  - Atribuições múltiplas:
    - São executadas da direita para a esquerda;
    - Exemplo: ***x* = *y* = 3;**

Operadores

- **Operadores Aritméticos Binários**

Operadores aritméticos	Operação
*	Multiplicação
/	Divisão
%	Módulo (resto)
+	Soma
-	Subtração

- **Operador Aritmético Unário**
  - Operador **–** (menos unário)
  - É usado para indicar a troca do sinal algébrico do valor associado.

Operadores

- **Operador de endereços (&)**
  - Opera sobre uma variável e resulta o seu endereço.
  - Um endereço é a referência utilizada para localizar variáveis.
  - Endereços de memória são impressos em hexadecimal e o formato usado é *%p*.

Operadores

- **Operador de endereços (&)**

```c
#include <stdio.h>
#include <stdlib.h>

int main()
{
    int x;
    x = 2;
    printf("Valor=%d, endereco=%p\n", x, &x);
    return 0;
}
```

Operadores

- **Operadores Relacionais**

Operadores relacionais	Ação
==	Igual
>	Maior
<	Menor
>=	Maior ou igual
<=	Menor ou igual
!=	Diferente (não igual a)

Função *scanf()*

- **Permite ler dados a partir da entrada padrão (teclado).**
- **A sintaxe é similar à de *printf()***
  - scanf("expressão de controle", lista de argumentos)
  - Os argumentos de ***scanf()*** devem ser endereços de variáveis.

Função *scanf()*

- **A expressão de controle pode conter códigos de formatação precedidos de um sinal % ou o caractere * colocado após % e antes do código de formatação.**
- **O caractere * indica um valor deve ser lido, mas não deve ser atribuído à nenhuma variável.**
- **Exemplo:**
  - scanf(“%*c”);

Função *scanf()*

Código de formatação para scanf()	Formato
%c	Caractere simples.
%d	Inteiro decimal com sinal.
%i	Inteiros decimal com sinal.
%e	Notação científica.
%g	Usa %e ou %f, o que for menor.
%o	Inteiro octal.
%f	Ponto flutuante em decimal.
%u	Inteiro decimal sem sinal.
%s	String de caracteres
%x	Inteiro hexadecimal
%ld	Inteiro decimal longo.
%lf	Ponto flutuante longo (double).
%Lf	Double longo.

Quantificador *const*

- **A palavra-chave *const* assegura que a variável associada não será alterada em todo o programa.**
- **Este quantificador é utilizado para declarar valores constantes.**
- **Obrigatoriamente, as variáveis associadas ao quantificador *const* devem ser inicializadas.**
- **Exemplo:**
  - const int x = 10;

Conversões de tipo

- **Conversão automática**
  - Quando dois ou mais operandos de tipos diferentes estão na mesma expressão, o conteúdo da variável de menor tamanho é convertido ao tipo da variável de maior tamanho.
  - O resultado a expressão é convertido para o tipo da variável à esquerda do operador de atribuição.
  - Problema:
    - Nem sempre a conversão automática produz resultados corretos.

Conversões de tipo

- **Conversão automática**

```c
#include <stdio.h>
#include <stdlib.h>

int main()
{
    int varInt = 200000000;
    int onze = 11;
    varInt = (varInt * onze) / onze; /*Valor muito grande*/
    printf("varInt = %d\n", varInt); /*Resposta errada*/
    return 0;
}
```

Conversões de tipo

- **Conversão explícita**
  - Operador de molde (ou ***cast***)
    - Consiste em escrever o nome do tipo desejado entre parênteses antes do valor ou expressão a ser avaliada.
    - O resultado é a expressão convertida para o tipo especificado.
  - Sintaxe do operador de molde
    - (tipo desejado) variável; ou
    - (tipo desejado) (expressão)

Conversões de tipo

- **Conversão explícita**

```c
#include <stdio.h>
#include <stdlib.h>

int main()
{
    int varInt = 200000000;
    int onze = 11;
    varInt = ((double)varInt * onze) / onze; /*Converte para double*/
    printf("varInt = %d\n", varInt); /*Resposta correta*/
    return 0;
}
```

Funções *getche()* e *getch()*

- **Leem um caractere no instante em que ele é digitado sem a necessidade de pressionar a tecla [ENTER].**
- **Pertencem à biblioteca *conio.h*.**
- ***getche()***
  - Retorna o caractere lido do teclado
  - Exemplo: x = getche()
- ***getch()***
  - Retorna o caractere lido do teclado, mas o caractere não é impresso no vídeo.

Funções *getche()* e *getch()*

```c
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
int main()
{
    char c;
    c = getch();
    printf("\nc = %c\n", c);
    return 0;
}
```

Funções *getchar()* e *putchar()*

- **Estão definidas no arquivo *stdio.h*;**
- ***getchar()***
  - Aguarda o próximo caractere da entrada padrão e retorna o caractere lido.
  - Só termina a leitura quando a tecla [ENTER] é pressionada.
- ***putchar()***
  - Imprime um caractere na saída padrão.

Operadores de Incremento e Decremento

- **++**
  - Opera sobre o nome de uma variável e adiciona 1 ao valor da variável operando.
  - Exemplo:
    - x++
    - ++x

    } Equivalem a **x = x + 1**
- **--**
  - Opera sobre o nome de uma variável e decrementa 1 ao valor da variável operando.
  - Exemplo:
    - x--
    - --x

    } Equivalem a **x = x - 1**

Operadores de Incremento e Decremento

- **Operador prefixado**
  - Incrementa (ou decrementa) a variável operando antes da execução da instrução em que ela aparece.
  - Exemplo:

```
x = 10;
n = ++x;
printf("n = %d x = %d\n", n, x);
```

    - A saída será:
      - n = 11 x = 11

Operadores de Incremento e Decremento

- **Operador pós-fixado**
  - Incrementa (ou decrementa) a variável operando logo após a execução da instrução em que ela aparece.
  - Exemplo:

```
x = 10;
n = x++;
printf("n = %d x = %d\n", n, x);
```

    - A saída será:
      - n = 10 x = 11

Operadores Aritméticos de Atribuição

- **Combinam operações aritméticas com a operação de atribuição.**
- **Objetivo:**
  - Oferecer uma maneira mais curta e clara de escrita de certas expressões de atribuição.
- **Operadores:**
  - +=, -=, *=, /= e %=
- **Regra geral:**
  - **x op = exp** equivale a **x = x op (exp)**

Operadores Aritméticos de Atribuição

- **Exemplos:**

```
i  += 2;        equivale a      i = i + 2;
x  *= y+1;      equivale a      x = x * (y+1);
t  /= 2.5;      equivale a      t = t / 2.5;
p  %= 5;        equivale a      p = p % 5;
d  -= 3;        equivale a      d = d - 3;
```

Operadores Lógicos

- **Operadores Lógicos**

Operadores lógicos	Ação
&&	AND
||	OR
!	NOT

Operadores Lógicos

- **Tabela verdade do operador lógico OR (| |):**

P	Q	P || Q
V	V	V
V	F	V
F	V	V
F	F	F

Operadores Lógicos

- **Tabela verdade do operador lógico AND (&&):**

P	Q	P && Q
V	V	V
V	F	F
F	V	F
F	F	F

Operadores Lógicos

- **Tabela verdade do operador lógico NOT (!):**

P	!P
V	F
F	V

Precedência dos Operadores

- **Ordem de precedência dos operadores**

Prioridade	Operador
1ª	Aritmético
2ª	Relacional
3ª	Lógico – not
4ª	Lógico – and
5ª	Lógico - or

Operador Condicional Ternário ?:

- **Possui uma construção bem diferente dos demais operadores;**
- **É o único operador de C que opera sobre três expressões.**
- **Sintaxe geral:**
  - exp1 ? exp2 : exp3
- **exp1 é avaliada primeiro**
  - se for verdadeira, **exp2** é avaliada e será o resultado da expressão condicional.
  - se for falsa, **exp3** é avaliada e será o resultado da expressão condicional.

Operador Condicional Ternário ?:

- **Exemplos:**

```
max = (a > b) ? a: b;
```

```
abs = (x > 0 ) ? x : -x; /* Abs é o valor absoluto de x */
```

```
printf("%s", (x%2)? "Impar" : "Par");
```

Referências

- MIZRAHI, V. V. Treinamento em Linguagem C. 2ª Edição. São Paulo: Person Prentice Hall, 2008.

- SCHILDT, H. C, Completo e Total. 3ª Edição revista e atualizada; Tradução e revisão técnica: Roberto Carlos Mayer. São Paulo: Makron Books, 1996.