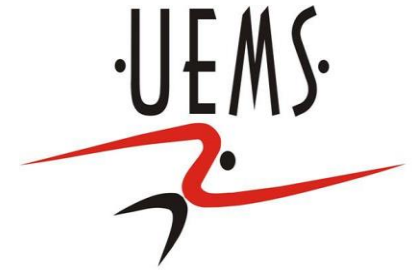


PCI- OPERADORES, EXPRESSÕES E FUNÇÕES.



Profa. Mercedes Gonzales
Márquez

Operações aritméticas e expressões

- Expressões aritméticas são formadas por valores e operadores aritméticos. Os operadores aritméticos são:

Símbolo	Função
+	Adição
-	subtração
*	Multiplicação
/	Divisão real
%	Resto da divisão inteira
/	Quociente da divisão inteira

Operações aritméticas e expressões

$$\begin{array}{r} \diamond 13 \mid _5_ \\ 3 \quad 2 \end{array}$$

- Quociente da divisão Inteira : $13/5=2$
- Resto da divisão inteira : $13\%5=3$
- Divisão real : $13/5=2.6$

$$\begin{array}{r} \diamond 29 \mid _4_ \\ 1 \quad 7 \end{array}$$

- Quociente da divisão Inteira : $29/4=7$
- Resto da divisão inteira : $29\%4=1$
- Divisão real : $29/4=7.25$

Operações (Precedência)

- Precedência é a ordem na qual as operações serão calculadas quando o programa for executado. Em C, os operadores são aplicados na seguinte ordem:
 1. * e /, na ordem em que aparecerem na expressão.
 2. %,
 3. + e -, na ordem em que aparecerem na expressão.

Exemplos:

1. $7 + 11 \% 3$ é igual a 9 $\rightarrow 7 + 2=9$
2. $7 * 11 \% 3$ é igual a 2 $\rightarrow 77\%3=2$

Operações (Precedência)

Use parênteses para indicar que o resultado da expressão interna deve ser calculado antes de se permitir que outras expressões executem sobre ela.

Exemplos:

1. $7 + 11 \% 3$ é igual a 9 $\rightarrow 7 + 2=9$

2. $(7 + 11) \% 3$ é igual a 0 $\rightarrow 18\%3=0$

- Em expressões mais complexas, sempre use parênteses para deixar claro em qual ordem as expressões devem ser avaliadas.

Operações aritméticas e expressões

Exemplos:

```
#include<stdio.h>
int main(){
    int a=20, b=10;
    float c=1.5,d;
    d=c*b/a; /*atribui 0.75 para d*/
    printf ("%f\n",d);
    d=c*(b/a); /*atribui 0.0 para d pois a divisão entre inteiros
        resulta em um inteiro*/
    printf ("%f",d);
}
```

Operações aritméticas e expressões

```
#include<stdio.h>
int main(){
    int a=20, b=10;
    float c=1.5, d;
    d= (((a+5)*10)/2)+b; /*atribui 135 para d*/
}
```

Operadores de incremento e decremento

- `x++` ou `++x` incrementam o valor da variável `x` em uma unidade.
- `x--` ou `--x` decrementam o valor da variável `x` em uma unidade.
- Eles podem ser incremento ou decremento prefixo e incremento ou decremento sufixo, dependendo da posição do operador de incremento ou de decremento em relação à variável.

Operadores de incremento e decremento

- Operador prefixo: primeiro, a variável é alterada, depois o comando escreve o valor da variável.
- Exemplo:

```
#include <stdio.h>
int main() {
    int x = 10;
    printf("%d\n", ++x);
    printf("%d\n", x);
}
```

Saída: 11
11

Operadores de incremento e decremento

- Operador sufixo: primeiro, o comando escreve o valor da variável e depois a variável é alterada.

Exemplo:

```
#include <stdio.h>
int main() {
    int x = 10;
    printf("%d\n", x++);
    printf("%d\n", x);
}
```

Saída: 10
11

Operadores de incremento e decremento

- Exemplo de uma expressão que inclui operador de incremento.

```
#include <stdio.h>
int main() {
    int x = 10;
    printf("%d\n", 3 * x++);
    printf("%d\n", x);
}
```

Saída: 30

11

Atribuições simplificadas

- Se x for o identificador da variável e $\$$ for um operador aritmético, a atribuição $x = x \$$ (expressão) pode ser indicada, simplesmente, por $x \$=$ expressão;

Operador	Exemplo	Correspondente
<code>+=</code>	<code>a += b;</code>	<code>a = a + b;</code>
<code>-=</code>	<code>a -= b;</code>	<code>a = a - b;</code>
<code>*=</code>	<code>a *= b;</code>	<code>a = a * b;</code>
<code>/=</code>	<code>a /= b;</code>	<code>a = a / b;</code>
<code>%=</code>	<code>a %= b;</code>	<code>a = a % b;</code>

Atribuições simplificadas

Exemplos:

- As seguintes atribuições são equivalentes:
 $a = a + 1$; $a += 1$; $a++$; $++a$;
- A expressão $j^*=x+2$ é equivalente a $j=j^*(x+2)$
- $x *= 4$; equivale a $x = x*4$;
- $x += 5$; equivale a $x = x + 5$;
- $x %= y + 1$; equivale a $x = x \% (y + 1)$;
- $x -= 5$; equivale a $x = x - 5$;
- $x /= 2$; equivale a $x = x/2$;

Atribuições simplificadas

```
#include <stdio.h>
int main() {
    int x,a,b;
    x=2;
    a=10;
    b=3;
    printf ("%d\n",x);
    x++;
    printf ("%d\n",x);
    x+=a;
    printf ("%d\n",x);
    x*=b;
    printf ("%d\n",x);
}
```

Conversão de valores de tipos diferentes

- É possível converter valores de alguns tipos em outros tipos de duas formas: implícita e explícita.
- Conversão implícita: Atribui-se variáveis de um tipo em variáveis de outro tipo, cuidando que a capacidade (tamanho) do destino seja maior do que a da origem para que não haja perda de informação.

– Exemplo sem perda de informação:

```
double a;  
float b = 2.2;  
a = b;
```

– Exemplo com perda de informação:

```
int a, b;  
double x = 2.2, y = -1.65;  
a = x;          /* a = 2 */
```

Conversão de valores de tipos diferentes

- Conversão explícita:
 - Informa-se explicitamente o tipo para o qual o valor deve ser convertido, usando a seguinte notação:
(tipo) valor
- Não é possível modificar o tipo de uma variável, apenas converter o tipo de expressão.

Conversão de valores de tipos diferentes

Exemplo:

```
float a;
```

```
int b = 25, c = 3;
```

```
a = b /c;
```

```
printf ("%f\n",a);
```

```
a = (float)(b/c);
```

```
printf ("%f\n",a);
```

```
a = (float)b / (float)c;
```

```
printf ("%f\n",a);
```

```
a = b / (float)c;
```

```
printf ("%f\n",a);
```

```
a = (float)b /c;
```

```
printf ("%f",a);
```

```
8.000000
8.000000
8.333333
8.333333
8.333333
Process returned 8 (0x8)   execution time : 0.108 s
Press any key to continue.
```

Funções matemáticas

Várias funções matemáticas são disponibilizadas pela linguagem C.

Função	Resultado
$\text{Sin}(x)$	Seno de um ângulo
$\text{cos}(x)$	Coseno de um ângulo
$\text{tan}(x)$	Tangente do ângulo
$\text{exp}(x)$	O número e elevado a x.
$\text{ln}(x)$	Logaritmo neperiano de x
$\text{sqrt}(x)$	Raiz quadrada de x
$\text{log10}(x)$	Logaritmo base 10 de x
$\text{round}(x)$	Arredondamento do ponto flutuante x.
$\text{pow}(x,n)$	Potencia (x é a base e n é o expoente)

Funções matemáticas

Exemplo 1.

```
#include <stdio.h>
#include <math.h> /* biblioteca padrão para funções matemáticas básicas*/
# define PI 3.1415926536 /* atribui o valor para PI*/
int main() {
    float a=1.0,b;
    printf("O valor 4.3 arredondado usando round() e %f\n", round(4.3));
    b=exp(a);
    printf("a=%f, b=%f\n",a,b);
    a=2.0;
    a=pow(a,3.0);
    printf("a=%f\n",a);
    a=cos(PI/2);
    printf("a=%f\n",a);
}
```

Funções matemáticas

Exemplo:

Faça um programa que calcule a área de um triângulo em função de seus lados a , b e c .

$$AREA = \sqrt{\rho x(\rho - a)x(\rho - b)x(\rho - c)}$$

Onde $\rho = (a + b + c) / 2$ é o semi-perímetro do triângulo.

```
#include <stdio.h>
#include <math.h>
int main(){
    float a, b, c, Area, SemiPer;
    printf("Digite os comprimentos dos lados do triangulo");
    scanf("%f %f %f", &a, &b, &c);
    SemiPer = (a + b + c)/2;
    Area = sqrt(SemiPer * (SemiPer - a) * (SemiPer - b) * (SemiPer - c));
    printf("A area do triangulo de lados %f , %f e %f e' igual a %f \n", a, b,
c, Area);
}
```

Expressões relacionais

- Expressões relacionais são aquelas que realizam uma comparação entre duas expressões e retornam zero (0), se o resultado for falso e um (1), ou qualquer outro número diferente de zero, se o resultado for verdadeiro.
- Os operadores relacionais são:

Operador relacional	Resultado
$x = y$	verdade se x for igual a y
$x \neq y$	verdade se x for diferente de y
$x < y$	verdade se x for menor que y
$x > y$	verdade se x for maior que y
$x \leq y$	verdade se x for menor ou igual a y
$x \geq y$	verdade se x for maior ou igual a y

Exemplo: A saída produzida por

```
printf("%d %d", 5<6, 6>5);
```

será 1 1.

Expressões lógicas

- Expressões lógicas retornam verdadeiro ou falso e são formadas por valores lógicos (como os que resultam das expressões relacionais) e os operadores lógicos tais como e (and), ou (or) e não (not).
- Os operadores lógicos são:

Operador lógico	Resultado
$! x$	<i>verdade se e só se x for falso</i>
$x \&\& y$	<i>verdade se e só se x e y forem verdade</i>
$x y$	<i>verdade se e só se x ou y for verdade</i>

Expressões lógicas (precedência)

- Numa expressão contendo operadores aritméticos, relacionais e lógicos, a avaliação é efetuada na seguinte ordem:
 1. primeiro avaliam-se todos os operadores aritméticos;
 2. em seguida, avaliam-se os operadores relacionais;
 3. só então, avaliam-se os operadores lógicos

Operador Conjunção (&&)

- Retorna verdadeiro quando ambas as expressões são verdadeiras. Sua tabela verdade é:

<expressão1>	<expressão2>	resultado
V	V	V
V	F	F
F	V	F
F	F	F

Exemplos para a=0, b=0, x=6,y=10,z=30.

(a == 0) && (b == 0) → V && V = V

(x >= y) && (y >= z) && (x != z) → F && F & V = F

Operador Disjunção (||)

- Retorna verdadeiro quando pelo menos uma das expressões é verdadeira.. Sua tabela verdade é:

<expressão1>	<expressão2>	resultado
V	V	V
V	F	V
F	V	V
F	F	F

Exemplos para a=0, b=0, x=6,y=10,z=30.

$(a == 0) \parallel (b == 0) \rightarrow V \parallel V = V$

$(x >= y) \parallel (y >= z) \parallel (x != z) \rightarrow F \parallel F \parallel V = V$

Operador Negação (!)

- Retorna verdadeiro quando a expressão é falsa e vice-versa. Sua tabela verdade é:

<code><expressão></code>	resultado
V	F
F	V

Exemplos para a=0, b=0.

`!(a == 0) → !V = F`

`!(a >= b) → !V = F`

Tipo Bool

- Usando a biblioteca `stdbool.h` podemos usar um tipo de dados chamado `bool` (booleano ou lógico) que considera os valores `true` (=verdadeiro) e `false` (=falso) no programa.

```
bool b1, b2;
```

```
b1 = true; b2 = false;
```

```
if (b1) printf("%d\n", b2); // Imprime 0
```

Operações lógicas e expressões

```
#include <stdio.h> /*Biblioteca para incluir comandos de entrada e  
saída como printf*/
```

```
#include <stdbool.h>
```

```
int main(){
```

```
    bool v1=true,v2=false,v3;
```

```
    v3=v1&&v2;
```

```
    printf("%d,%d,%d\n",v1,v2,v3);
```

```
    v3=v1||v2;
```

```
    printf("%d,%d,%d\n",v1,v2,v3);
```

```
    v3=!v1;
```

```
    printf("%d,%d,%d\n",v1,v2,v3);
```

```
    v3=(v1&&v2)||!v2;
```

```
    printf("%d,%d,%d\n",v1,v2,v3);
```

```
}
```

Introdução a Estruturas de Controle de Fluxo

Já vimos Estrutura Sequencial.
Veremos:

- Estrutura Condicional
- Estrutura de Repetição

Estrutura Condicional

Em muitas tarefas de programação desejamos que o programa execute instruções diferentes dependendo de alguma condição lógica.

Estrutura if..

Executa código somente se uma condição for verdadeira (resultado da expressão diferente de zero)

Exemplo:

Estrutura Condicional

```
int main( ){  
    int idade;  
    printf("Digite sua idade: ");  
    scanf("%d", &idade);  
    if (idade >= 18) {  
        printf("Voce e maior de idade,,");  
    }  
}
```

Estrutura Condicional

Estrutura if.. else ...

Condição verdadeira: executa o primeiro bloco

Caso contrário: executa o segundo bloco

```
if (expressão) {  
    sentença;  
    ...  
} else{  
    sentença;  
    ...  
}
```


Estrutura Condicional

```
int main() {
    int idade, diferenca_tempo;
    printf("Digite sua idade: ");
    scanf("%d", &idade);
    if (idade >= 18) {
        diferenca_tempo = idade - 18;
        printf("Voce eh maior de idade ha %d ano(s)",
diferenca_tempo);
    } else {
        diferenca_tempo = 18 - idade;
        printf("Espere mais %d ano(s)!\n", diferenca_tempo);
    }
}
```

Estrutura de Repetição

Estrutura que permite repetir a execução de um bloco sob o controle de uma condição ou um número pré-determinado de vezes.

- Exemplos:
 - Preencher uma tabela
 - Aplicar operação a todos elementos da lista
 - Testar vários números
 - Percorrer matrizes, vetores, listas

Estrutura de Repetição

Repetição controlada por condição

Comando while

Estrutura:

```
while ( condicao ){  
comandos;  
}
```

Enquanto a condição for verdadeira ($\neq 0$), ele executa o(s) comando(s);

Estrutura de Repetição

Repetição controlada por condição (duas questões)

1. O que acontece se a condição for falsa na primeira vez?

while (a!=a)

a=a+1;

R: Ele nunca entra na repetição (loop).

2. O que acontece se a condição for sempre verdadeira?

while (a == a)

a=a+1;

R: Ele entra na repetição e nunca sai (loop infinito)

Estrutura de Repetição

Exemplo : Imprimir os 10 primeiros números inteiros

```
int i=1;  
while (i<=10){  
    printf(“%d”,i);  
    i++;  
}
```

Estrutura de Repetição

Exemplo 1: Imprimir os n primeiros números inteiros

```
int i=1,n;  
scanf("%d",&n);  
while (i<=n){  
    printf("%d ",i);  
    i++;  
}
```

Estrutura de Repetição

Exemplo 2: Faça um algoritmo que determine os quadrados de um conjunto de números inteiros positivos.

```
int num;  
scanf ("%d",&num);  
while (num>0){  
    printf("%d",num*num);  
    scanf ("%d",&num);  
}
```