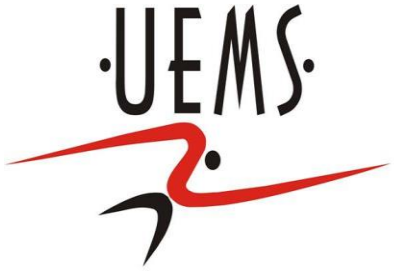

PCI- Operadores, Expressões e Funções.

Profa. Mercedes Gonzales Márquez

Constantes

- São valores previamente determinados e que não se alteram ao longo do programa.
- Assim como as variáveis, as constantes também possuem um tipo. Os tipos permitidos são exatamente os mesmos das variáveis, mais o tipo string, que corresponde a uma sequência de caracteres. I Exemplos: 90, 0.23, ’c’, “Ana Ortiz".

- Constantes devem ser definidas logo no começo do programa, após as declarações das bibliotecas, de forma que elas possam ser usadas em qualquer ponto do programa.

Constantes

- Podemos definir uma constante usando o seguinte comando:

  #define CONSTANTE valor

Exemplos:

#define PI 3.14159

#define DIAS_FEVEREIRO 28

#define DIAS_DA_SEMANA 7

#define TAMANHO 100

Constantes

- Podemos definir uma constante usando o seguinte comando:

  #define CONSTANTE valor

Exemplos:

#define PI 3.14159

#define DIAS_FEVEREIRO 28

#define DIAS_DA_SEMANA 7

- Ao se definir uma constante (usando #define) não é alocado um espaço de memória para ela. O compilador apenas substitui o valor da constante em todo lugar que ela é usada no programa

Constantes e Variáveis

- Uma constante é uma expressão e, como tal, pode ser atribuída a uma variável ou ser usada em qualquer outro lugar onde uma expressão seja permitida.

Exemplo:

```
double raio, comp_circ;

r = 1.0;
comp_circ= 2 * PI * r;
```

Tipo Bool

- Definindo um novo tipo

Podemos usar o comando typedef para definir novos tipos de variáveis ou abreviar tipos existentes.

typedef enum {false,true} bool; /* o tipo bool só armazena 0/false e 1/true*/

- Usando a biblioteca stdbool.h;

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

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 (Precedência)

- Precedência é a ordem na qual os operadores serão calculados quando o programa for executado. Em C, os operadores são calculados 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 → 7 + 2=9
2. 7 * 11 % 3 é igual a 2 → 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 → 7 + 2=9
2. (7 + 11) % 3 é igual a 0 → 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:

```
int main(){
    int a=20, b=10;
    float c=1.5,d;
    d=c*b/a; /*atribui 0.75 para d*/
    printf (“%f”,d);
    d=c*(b/a); /*atribui 0.0 para d pois a divisão entre inteiros
     resulta em um inteiro*/
}
```

Operações aritméticas e expressões

```
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 a expressão retorna 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, a expressão retorna o valor da variável, 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

- Em uma expressão, os operadores de incremento e decremento são sempre calculados antes dos demais (maior precedência).

Exemplo:

```
#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
+=	`a += b;`	`a = a + b;`
-=	`a -= b;`	`a = a - b;`
*=	`a *= b;`	`a = a * b;`
/=	`a /= b;`	`a = a / b;`
%=	`a %= b;`	`a = a % b;`

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;.

Conversão de valores de tipos diferentes

- É possível converter valores de alguns tipos em outros tipos de duas formas: implícita e explícita.
- Conversão implícita: A capacidade (tamanho) do destino deve ser 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 */
b = y;                    /* b = -1 */
```

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.

Exemplo:

```
float a;
int b = 25, c = 3;
a = (float) b / (float) c;
```

Funções matemáticas

Várias funções matemáticas são disponibilizadas pela linguagem C.

Função	Resultado
Sin(x)	Seno de um ângulo
cos(x)	Coseno de um ângulo
tan(x)	Tangente do ângulo
exp(x)	O número e elevado a x.
ln(x)	Logaritmo neperiano de x
sqrt(x)	Raiz quadrada de x
log10(x)	Logaritmo base 10 de x
Round(x)	Arredondamento do ponto flutuante x.

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 x, y, z, Area, SemiPer;
        printf("Digite os comprimentos dos lados do triangulo");
        scanf("%f %f %f", &x, &y, &z);
        SemiPer = (x + y + z)/2;
        Area = sqrt(SemiPer * (SemiPer - x) * (SemiPer - y) * (SemiPer - z));
        printf("A area do triangulo de lados %f , %f e %f e' igual a %f \n", x, y,
z, 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
$x != y$	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
$x <= y$	verdade se x for menor ou igual a y
$x >= y$	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*	*verdade* se e só se *x* for *falso*
x && *y*	*verdade* se e só se *x* e *y* forem *verdade*
x || *y*	*verdade* se e só se *x* ou *y* for *verdade*

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) || (b == 0) → V || V = V

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

Operador Negação (!)

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

<expressão>	resultado
V	F
F	V

Exemplos para a=0, b=0.

!(a == 0) → !V= F

!(a >= b) → !V=F

Operações lógicas e expressões

```
#include <stdio.h> /*Biblioteca para incluir comandos de entrada e
                     saída como printf*/
typedef enum {false,true} bool;
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);
}
```

Exercícios de comandos de entrada e saída

1. Considerando que uma pessoa dorme 8 horas, faça um algoritmo que leia a idade da pessoa e imprima quantos anos ela tem e quantos anos da sua vida, ela passou dormindo.
2. Escreva um algoritmo para ler o nome e a idade de uma pessoa, e exibir quantos dias de vida ela possui. Considere sempre anos completos, e que um ano possui 365 dias. Ex: uma pessoa com 19 anos possui 6935 dias de vida; veja um exemplo de saída: MARIA, VOCÊ JÁ VIVEU 6935 DIAS.
3. Faça um algoritmo que leia o valor do salário de um funcionário, calcule e mostre seu novo salário, sabendo que o mesmo recebeu um aumento de 21,3%.
4. Leia os valores de dois catetos de um triângulo e calcule e mostre o valor da hipotenusa

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 condicão for verdadeira (!=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);
}
```