

Tipos de Dados – Constantes e Variáveis

Constantes

- Caracteres – 'a', 'A', '\0'
- Cadeia de caracteres – "UERJ", "IME"
- Constantes inteiras – 2018, -33
- Constantes inteiras longas – 1234L
- Constantes octais – 077, 010
- Constantes hexadecimais – 0xFFFF, 0x10
- Constantes em ponto flutuante – 3.14159, 3.14e0

Tipos de Dados – Constantes e Variáveis

Variáveis

<tipo> <lista de variáveis>

- Ponto flutuante

```
int main ()  
{  
    int a, b;  
    float razao;  
  
    a = 10;  
    b = 3;  
    razao = a/b;  
    printf ("o resultado da divisao e': %f", razao);  
    return 0;  
}
```

```
o resultado da divisao e': 3.000000  
-----
```

Tipos de Dados – Constantes e Variáveis

Variáveis

- Inteiras
- Caracter
- Sem sinal
- Cadeia de caracteres
- Ponteiro

int <nome da variável>

char <nome da variável>

unsigned tipo da variável

char cadeia [80]

char *ptr

Operadores e Expressões

As expressões combinam operandos e operadores para um único resultado.

Operandos – constantes, variáveis ou valores fornecidos por funções.

O resultado de uma expressão também constitui um tipo que, em geral, é o mesmo dos operandos envolvidos.

() [] ->	→
! ++ -- * & (tipo) sizeof ()	←
*/ % \	→
+ -	→
>> <<	→
< <= >= >	→
== !=	→
&	→
^	→
	→
&&	→
	→
? (): ()	→
= += -=	←

Operadores e Expressões

Operador de atribuição soma = a + b;
 a = b = c = 3.0;

Operadores aritméticos *, /, %, +, -

Operadores de incremento/decremento ++ --

 m = 3.0 * n++;

 m = 3.0 * ++n;

 i++; <= => i = i + 1;

Operadores em bits &, |, ^, <<, >>

Operadores de atribuição composta

 <variável> = <variável> <operando> <expressão>

 a = a + b; -----> a += b;

Operadores de endereço

 & - devolve o endereço

 * - devolve o conteúdo da posição de memória apontada

Conversão de Tipos

Ocorre quando há operadores de tipos distintos em uma expressão.

- char e short são sempre convertidos para int
- float é sempre convertido para double

Exercício

O que será impresso no programa?

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

```
int main ()
```

```
{
```

```
    int a, b;
```

```
    a = b = 5;
```

```
    printf ("%d\n", a + b);
```

```
    printf ("%d\n", a++ + b);
```

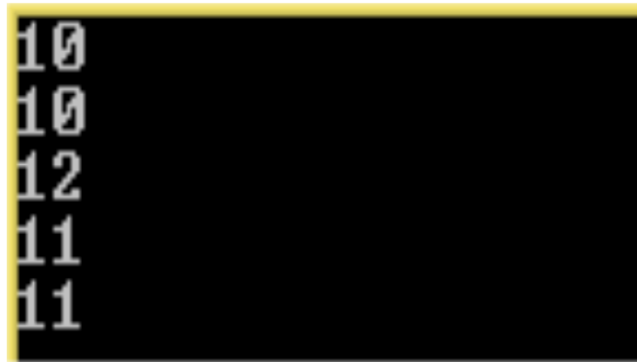
```
    printf ("%d\n", ++a + b);
```

```
    printf ("%d\n", --a + b);
```

```
    printf ("%d\n", a++ + b);
```

```
    return 0;
```

```
}
```



```
10
10
12
11
11
```

Entrada de Dados

Função scanf ()

scanf ("expressão de controle", endereço1, endereço2, ...);

#include <stdio.h>

int main ()

{

int a, b, soma;

scanf ("%d %d", &a, &b);

soma = a + b;

printf ("Soma = %d\n", soma);

return 0;

}

Entrada de Dados

Strings

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

```
int main ()
```

```
{
```

```
    char nome[30];
```

```
    printf ("Digite o seu nome: ");
```

```
    scanf ("%s", nome);          scanf ("%s", &nome[0]);
```

```
    printf ("Como vai %s?\n", nome);
```

```
    return 0;
```

```
}
```

Entrada de Dados

Função gets ()

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

```
int main ()
```

```
{
```

```
    char nome[30];
```

```
    printf ("Digite o seu nome: ");
```

```
    gets (nome);
```

```
    printf ("Como vai %s?\n", nome);
```

```
    return 0;
```

```
}
```

Entrada de Dados

Função getch() e getche()

getch() – não imprime o caracter lido na tela

getche() – imprime o caracter lido na tela

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

```
#include <conio.h>
```

```
int main ()
```

```
{
```

```
    char ch;
```

```
    printf ("Digite um caracter: ");
```

```
    ch = getche ();
```

```
    printf ("\n O caracter digitado foi %c?\n", ch);
```

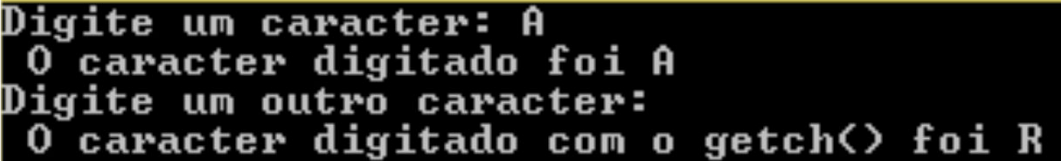
```
    printf ("Digite um outro caracter: ");
```

```
    ch = getch ();
```

```
    printf ("\n O caracter digitado com o getch() foi %c\n", ch);
```

```
    return 0;
```

```
}
```



```
Digite um caracter: A
O caracter digitado foi A
Digite um outro caracter:
O caracter digitado com o getch() foi R
```

Exercício

Escreva um programa que leia do teclado uma temperatura em graus Celsius, faça a conversão para Fahrenheit e imprima o resultado.

```
#include <stdio.h>

int main ()
{
    float ctemp, ftemp;

    printf ("Digite a temperatura em graus Celsius: ");
    scanf ("%f", &ctemp);
    ftemp = 1.8 * ctemp + 32;
    printf ("\nA temperatura convertida em Fahrenheit e': %4.2f", ftemp);

    return 0;
}
```

```
Digite a temperatura em graus Celsius: 40
A temperatura convertida em Fahrenheit e': 104.00
```

Desvio Condicional

Operadores relacionais

`>, >=, <, <=, ==, !=`

Operadores lógicos

`&&, ||, !`

Ex.: `(a > 0) && (a <= 4)`

`(a > 0) || (a == 3)`

Desvio Condicional

Comando IF

```
if (<condição>)  
    <comando1>;
```

```
if (salario > 10000)  
    {  
        inps = salario * 0.11;  
        irpf = salario * 0.27;  
    }
```

```
if (<condição>)  
    <comando1>  
else  
    <comando2>;
```

Desvio Condicional

`if (a != 0)` equivale a `if (a)`

`if (a == 8)` é diferente de `if (a = 8)`

```
if ((ch = getch()) >= 'a')
    if (ch <= 'z')
        puts ("Voce digitou uma letra!");
    else
        puts ("Voce nao digitou uma letra!");
```

Comando switch

```
int main()
{
    char opcao;
    if (opcao == 'A')
        { <comandos>}
    else if (opcao == 'E')
        { <comandos>}
    else if (opcao == 'I')
        { <comandos>}
    else puts ("opcao invalida");
}
```

```
switch (opcao)
{
    case 'A': <comandos>;
    case 'E': <comandos>;
    case 'I': <comandos>;
    default: puts ("opcao invalida");
}
```


Estruturas de Repetição

while (expressão)

<comandos>

- Caso o while esteja vazio (sem expressão) é considerado sempre verdadeiro

while ()

<comandos>

- Estrutura while sem comando

while (x [i++] != num)

; /* comando nulo */

Neste caso o vetor *x* é percorrido enquanto o conteúdo de uma posição for diferente do conteúdo da variável *num*.

Estruturas de Repetição

```
for (<expressão1>; <expressão2>; <expressão3>)  
    <comando>
```

expressão1 – inicialização da variável de controle

expressão2 – teste lógico de parada

expressão3 – incremento da estrutura

```
int main ()  
{  
    int i = 1;  
    for (; i<=10; i++)  
        { printf ("%d\n", i); }  
    return 0;  
}
```

```
int main ()  
{  
    int i;  
    for (i = 1;; i++)  
    {  
        if (i<=10)  
            printf ("%d\n", i);  
        else break;  
    }  
    return 0;  
}
```

Estruturas de Repetição

do – while

do

<comando>

while (<expressão>);

```
int main ()
```

```
{
```

```
    int i = 1;
```

```
    do
```

```
        printf ("%d\n", i++);
```

```
    while (i <=10);
```

```
    return 0;
```

```
}
```

Comandos break e continue

break

Comando utilizado dentro de uma estrutura de repetição ou o switch e que faz com que o loop seja imediatamente interrompido.

```
while (<expressão>)
```

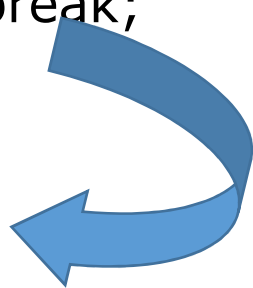
```
{
```

```
    if (<expressão2>)
```

```
        break;
```

```
}
```

```
<comando>
```



Comandos break e continue

continue

O comando desvia o fluxo de execução para a próxima iteração dentro de uma estrutura de repetição.

```
while (<expressão>)  
{
```

```
    if (<expressão2>)  
        continue;
```

```
}  
<comando>
```

