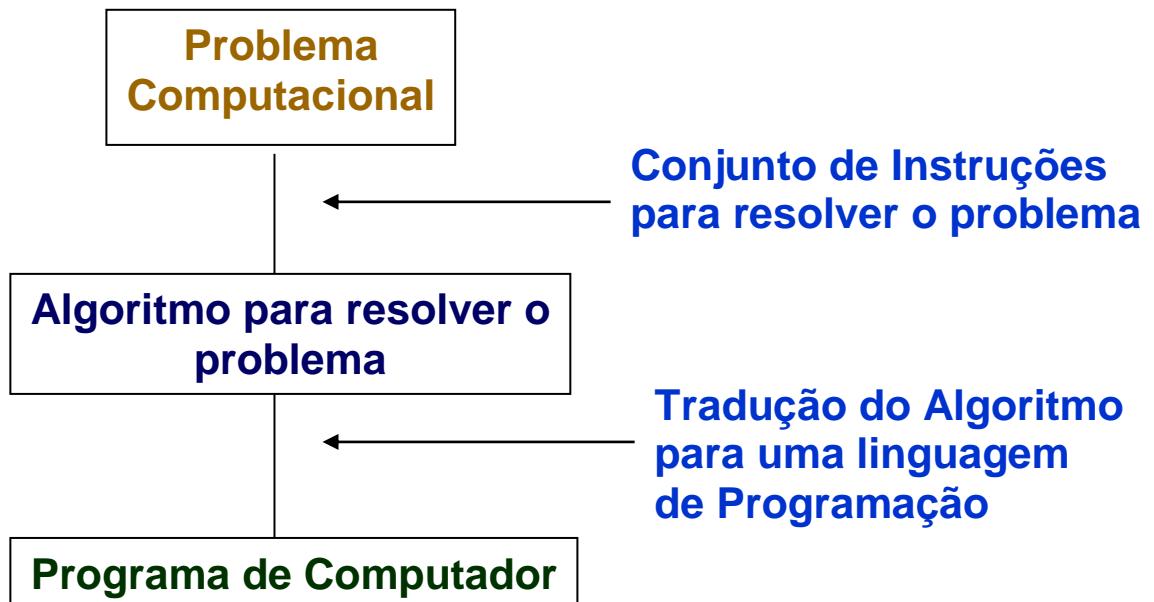


# TURBO C

## Definições Simples de Alguns Termos

- **Algoritmo** - Conjunto de Instruções para resolver um problema.
- **Programa de Computador** - Conjunto de Instruções escritas em uma linguagem de programação para resolver um problema, também pode ser entendido como um software.



- **Linguagem de Programação** – Linguagem entendida pelo homem usada para escrever os programas de computador. *Ex.: Linguagem C, C++*
- **Código Fonte** – Arquivo texto com as instruções de um programa de computador. *Ex.: \*.C, \*.CPP*
- **Linguagem de Máquina** – Linguagem entendida pelo computador.
- **Compilar um Programa** – Transformar o programa da Linguagem de Programação para a Linguagem de Máquina.

- **Compilador** – Software que compila os programas. *Ex. Turbo C.*
- **Biblioteca de Funções** – Arquivo que contém os comandos (instruções) usados pelos programas. *Ex.: stdio.h, conio.h e math.h*
- **Portugol** – Pseudolínguagem, parecida com o português, usada para escrever os algoritmos
- **Identação** – Organização e estruturação dos comandos (instruções).

## Turbo C

- É um ambiente para escrevermos nossos programas em linguagem C. Será usado para transformarmos nossos algoritmos em programas de computador
- Interface de caractere, ou seja, não é gráfica. (*Parecida com o Antigo DOS*)
- Procurar trabalhar mais com o teclado e menos com o mouse
- Não precisa ser instalado, pode ser apenas copiado.
- Copiamos o diretório (pasta) do Turbo C++
- Iniciamos o Turbo C++ executando o arquivo TC.EXE
- Fica no subdiretório ..\BIN

## Usando o Turbo C++

- **Abrir o Turbo C++**

C:\TC\BIN\TC.EXE

- **Navegar Pelos Menus**

Mouse ou <F10> ou <Alt> + <Letra>

- **Digitar o Programa de Exemplo**

Menu: File / New

Começar a Digitar

- **Compilar o Programa de Exemplo**

Menu: Compile / Compile ou <Alt> + <F9>

- **Corrigir os Erros**

Ignorar o Warning “Function should return a value”

- **Executar o Programa de Exemplo**  
Menu: Run / Run ou <Ctrl> + <F9>
- **Visualizar o que Tela de Execução do Programa ( User Screen )**  
Menu: Window / user screen ou <Alt> + <F5>
- **Salvar o Programa de Exemplo**  
Menu: File / Save ou <F2>
- **Fechar o Programa de Exemplo**  
Menu: Window / Close ou <Alt> + <F3>
- **Sair do Turbo C++**  
Menu: File / Quit ou <Alt> + <X>

## Meu Primeiro Programa (Imprimir “Hello World” na Tela)

**Algoritmo para imprimir na tela a frase: “Hello, world”**

1 [Início]  
    escreva (“Hello, World”)  
2 [Fim]

**Fases da tradução do algoritmo para a linguagem de programação C, transformando-o em um programa:**

```
{  
printf("Hello World");  
}
```

Início - {  
fim - }  
imprima - printf  
Todo comando termina com um “;”

```
#include <stdio.h>  
main()  
{  
printf("Hello World ");  
}
```

Um programa é composto de uma ou mais funções  
A principal função é a **main ()**  
Precisamos definir as bibliotecas de funções que vamos usar:  
Sintaxe.: #include <nome-da-biblioteca.h>

# Como é um Programa em Linguagem C

- Precisamos definir as bibliotecas de funções que vamos usar:

Ex.: #include <nome-da-biblioteca.h>

É possível incluir várias instruções para o compilador C no código-fonte de um programa em C. Essas instruções são chamadas de *diretivas do pré-processador* e, ainda que atualmente não façam parte da linguagem C, expandem o escopo do ambiente de programação em C. Todas as diretivas do pré-processador começam com o sinal gráfico #.

A diretiva do pré-processador **#include** instrui o compilador a incluir outro arquivo-fonte dentro daquele que possui a diretiva **#include** nele.

O arquivo-fonte/biblioteca, para ser lido, deve estar entre aspas ou entre os sinais de maior e menor. Por exemplo,

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

instruem o compilador a ler e compilar o arquivo-cabeçalho para as rotinas de biblioteca de arquivos em disco.

- Um programa é composto de uma ou mais funções

A principal função é a **main ()**

- Cada função começa com um “{“ e termina com um “}”
- Programar em **letras minúsculas**, pois a **linguagem C faz diferenciação entre letras minúsculas e letras maiúsculas**. É importante saber que as letras maiúsculas e minúsculas são tratadas como caracteres distintos. Por exemplo, em algumas linguagens, os nomes de variáveis **count**, **Count** e **COUNT** são três maneiras de se especificar a mesma variável. Entretanto, na linguagem C, serão três variáveis diferentes.

- Todo comando termina com um “;”
- Comentários em C podem ser colocados em qualquer lugar em um programa e são cercados por duas marcas. A marca de começo é **/\*** e a marca de fim de comentário, **\*/**. Por exemplo:

```
/* Processamento */
```

- Os arquivos terão a extensão **\*.C** ou **\*.CPP** (para C++)
- A identação não é obrigatória em C.

```
#include <stdio.h>  
main(){ printf("Hello World");}
```

## Incrementando o Programa Hello World

```
#include <stdio.h>
#include <conio.h>
main()
{
    clrscr();
    printf("Hello, \n world");
    getch();
}
```

**clrscr()** - Limpa a Tela  
**getch()** - Espera o Usuário teclar <Enter>  
**\n** - Faz o comando **printf** pular de linha  
**conio.h** - Biblioteca com as funções: **clrscr()** e **getch()**

**Obs.:** *Não precisamos nos preocupar, por enquanto, com quais bibliotecas de funções vamos usar e quais os detalhes dos comandos, pois, isto será informado durante as aulas.*

## Declaração de Variáveis e Constantes Em C

## Tipos de Dados Básicos

Tipo C	Tipo Portugol	Bytes	Escala
char	caracter	1	1 Alfanumérico
char[N]	caracter / string	N	N Alfanuméricos
int	Inteiro	2	-32768 a 32767
float	real	4	3.4E-38 a 3.4E+38
double	real	8	1.7E-308 a 1.7E+308
void		0	sem valor

### Declarando Variáveis:

```
float A, B, VALOR;
double A, B, VALOR;
char SEXO, LETRA;
Char[50] NOME, MSG;
int Y; ←
int X = 0; ←
```

### Declarando Constantes:

```
const int X1 = 10;
const float A = 9.99;
const float pi = 3.14;
```

- Declarei variável Y do tipo int sem valor inicial
- Declarei variável X do tipo int com valor inicial = 0

✓ Podemos usar LETRAS MAIÚSCULAS para os nomes das variáveis

✓ Quando o tipo char quando tem várias posições, pode ser apelidado de string

✓ Podemos declarar mais de uma variável do mesmo tipo de uma única vez, separando-as por vírgula

✓ Podemos definir um valor inicial para uma variável

✓ As constantes obrigatoriamente devem possuir um valor inicial

✓ Uma vez definido um valor inicial para uma constante, não poderemos alterar este valor durante a execução do programa

# Definição de Valores para Variáveis Em C

Podemos definir um valor para uma variável de duas formas:

1a - Podemos atribuir valores diretamente para as variáveis:

**VALOR = 10.25;**

**OPCAO = 'A'**

**X = 256;**

Em Portugol ←

Em C =

2a - Podemos ler os valores que o usuário digitar

**scanf ("%f", &VALOR);**

**scanf ("%c", &OPCAO);**

**scanf ("%i", &X);**

Lendo as variáveis:

- VALOR do tipo float
- OPCAO do tipo char
- X do tipo int

Em Portugol leia (variável)

Em C **scanf ("Formato", &variável)**

Formatos:

%d ou %i – int

%c – char

%f – float ou double

%s - string - char[N]

Os formatos seguem a tabela ao lado, de acordo com o tipo da variável

## Incrementando a Função “printf”

Já usamos a função “printf” para imprimir textos na tela:

printf (“Hello, \n World”);	→	Hello, World
printf (“SOMA”);	→	SOMA
printf (“X”);	→	X

Podemos usar a função “printf” para imprimir o valor das variáveis na tela:

printf (“%c”, SEXO);	→	M
printf (“%f”, SOMA);	→	346.29
printf (“%i”, X);	→	25

Podemos usar a função “printf” para imprimir o valor das variáveis e textos na tela :

printf (“O sexo eh %c”, SEXO);	→	O sexo eh M
printf (“Soma = %f”, SOMA);	→	Soma = 346.29
printf (“Valor de x = %i”, X);	→	Valor de x = 25

## A função “printf” usa a mesma tabela de tipos de formatos da “scanf”

```
printf ("Formato", variável);
```

```
printf ("Formato1 Formato2", variável1, variável2);
```

```
printf ("Texto bla bla bla Formato1 texto bla bla bla Formato2", variável1, variável2);
```

Formatos:

%d ou %i – int

%c – char

%f – float ou double

%s - string - char[N]

✓ Os formatos são os mesmos para as funções: printf e scanf, conforme tabela ao lado. Ex.: %f para float

✓ Colocamos os formatos onde desejamos imprimir os valores das variáveis

✓ Os formatos ficam entre aspas juntamente com o texto

✓ Separamos por vírgulas, os nomes das variáveis que desejamos imprimir

- ✓ As variáveis, ficam sem aspas e na mesma ordem de seus respectivos formatos
- ✓ Não usamos & antes dos nomes da variáveis (*scanf usa &, printf não usa &*)
- ✓ Estamos nos referindo a variáveis, porém, podemos usar scanf para constantes e para resultados de operações também. Ex. *printf (“O resultado de 5 + 12 = %i”, 5+12);*

# Operadores em C

## Atribuição

=

## Aritméticos

- subtração

- + adição

- \* multiplicação

- / divisão

- % resto da divisão de números inteiros

## Exponenciação

**pow** (Base, Expoente) Exponenciação

## Relacionais

- > maior que

- < menor que

- >= maior ou igual

- <= menor ou igual

- == igual

- != diferente

## Lógicos

- && (e)

- || (ou)

- ! (não)

- Podemos usar parênteses para mudar a ordem de precedência (*como na matemática*)
- **pow** na verdade não é um operador, mas uma função e para usá-la, precisamos incluir a biblioteca `math.h`

## Algoritmo e Programa Usando Variáveis e Operadores

---

**Ler dois números inteiros, calcular e imprimir: a soma, a diferença, a multiplicação e a divisão entre eles.**

inicio

inteiro: A, B, SOMA, DIF, MULT;  
    real: DIV;  
    leia (A);  
    leia (B);  
    SOMA  $\leftarrow$  A+B;  
    DIF  $\leftarrow$  A-B;  
    MULT  $\leftarrow$  A\*B;  
    DIV  $\leftarrow$  B/A;  
    imprima (SOMA);  
    imprima (DIF);  
    imprima (MULT);  
    imprima (DIV);  
  fim.

```
#include <stdio.h>
#include <conio.h>
main()
{
    int A, B, SOMA, DIF, MULT;
    float DIV;
    clrscr();
    printf("Informe o Valor de A: \n");
    scanf("%d", &A);
    printf("Informe o Valor de B: \n");
    scanf("%d", &B);
    SOMA = A+B;
    DIF = A-B;
    MULT = A*B;
    DIV = B/A;
    printf ("Soma de A e B %d \n", SOMA);
    printf ("Diferenca de A e B %d \n", DIF);
    printf ("Multiplicacao de A e B %d \n", MULT);
    printf ("Divisao de A e B %d ", DIV);
    getch();
}
```

exemplo06

## Comando de Seleção Simples - IF

```
if (condição)
    comando1;
```

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

```
if (condição)
{
    comando1;
    comando2;
    comando3;
}
```

```
if (condição)
{
    comando1;
    comando2;
    comando3;
}
```

```
else
{
    comando4;
    comando5;
}
```

```
if (condição) comando1;
```

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

- Precisamos colocar as expressões / condições entre parênteses ()

- Não usamos então (then) em Linguagem C

- Quando temos apenas 1 comando dentro do if ou do else, não usamos chaves {}

- Quando temos mais de 1 comando dentro do if ou do else, devemos usar chaves {}

- Podemos escrever mais de um comando na mesma linha, porém, dificulta a manutenção

### Portugol - Linguagem C

Se	if
Condição	(Condição)
então	(Não existe)
senão	else
inicio	{
fim	}
= (igual)	== (dois iguais)
≠ (diferente)	!= (não aceita <>)
e	&&
ou	
negado	!
>	>
<	<
>=	>=
<=	<=

Escrever um programa em C para ler 2 números inteiros e a operação desejada entre eles, calcular e imprimir o resultado de acordo com a opção escolhida. As opções são 1 para Adição e 2 para Subtração.

```
#include <stdio.h>
#include <conio.h>
main()
{
    int A, B, X, OP;
    clrscr();
    printf ("Informe o valor do 1o Numero:\n");
    scanf("%i", &A);
    printf ("Informe o valor do 2o Numero:\n");
    scanf("%i", &B);
    printf ("Informe a operacao desejada:\n");
    printf ("1 - Adicao, 2 - Subtracao\n");
    scanf("%i", &OP);
```

```
if (OP == 1)
{
    printf("Escolheu Adicao\n", OP);
    X = A+B;
    printf("Resultado = %i",X);
}
else if (OP == 2)
{
    printf("Escolheu Subtracao\n");
    X = A-B;
    printf("Resultado = %i",X);
}
else
    printf("ERRO: Opcão %i invalida !",OP);
getch();
```

exemplo12

Escrever um programa em C para ler 2 números reais, calcular e imprimir a divisão entre eles. Lembre-se que não podemos dividir nenhum número por zero.

```
#include <stdio.h>
#include <conio.h>
main()
{
    float A, B, X;
    clrscr();
    printf ("Informe o valor do 1o Numero:\n");
    scanf("%f", &A);
    printf ("Informe o valor do 2o Numero:\n");
    scanf("%f", &B);
    if (B == 0)
        printf("ERRO: O Denominador = Zero");
    else
    {
        X = A/B;
        printf("Resultado = %f",X);
    }
    getch();
}
```

exemplo13

```
#include <stdio.h>
#include <conio.h>
main()
{
    float A, B;
    clrscr();
    printf ("Informe o valor do 1o Numero:\n");
    scanf("%f", &A);
    printf ("Informe o valor do 2o Numero:\n");
    scanf("%f", &B);
    if (B != 0)
        printf("Resultado = %f",A/B);
    else
        printf("ERRO: O Denominador = Zero");
    getch();
}
```

exemplo14

São mostradas duas lógicas diferentes para o mesmo problema. Existem várias programas corretos para um problema

**Escrever um programa em C para cada uma das Especificações abaixo:**

**Programa1: Ler dois números inteiros e dizer: se o primeiro é o maior ou se o segundo é o maior ou se eles são iguais.**

```
#include <stdio.h>
#include <conio.h>
main()
{
    int A, B;
    clrscr();
    printf ("Informe o valor do 1o Numero:\n");
    scanf("%i", &A);
    printf ("Informe o valor do 2o Numero:\n");
    scanf("%i", &B);
    if (A > B)
        printf("O primeiro numero eh o maior");
    else if (B > A)
        printf("O segundo numero eh o maior");
    else
        printf("Os dois numeros sao iguais");
    getch();
}
```

exemplo15

## Programa 2: Ler um número inteiro e dizer se ele é par ou ímpar.

```
#include <stdio.h>
#include <conio.h>
main()
{
    int A;
    clrscr();
    printf ("Informe um Numero:\n");
    scanf("%i", &A);
    if (A%2 == 0)
        printf("Numero par!");
    else
        printf("Numero impar!");
    getch();
}
```

exemplo17

## Case

```
switch (Variável)
{
    case Valor1:
        comando1;
        comando2;
        break;
    case Valor2:
        comando1;
        break;
    case Valor3,4,5:
        comando1;
        break;
        -
    case Valor n:
        comando1;
        -
        comando n;
        break;
    default:
        comando1;
        -
        comando n;
        break;
}
```

O comando switch case equivale a vários IFs aninhados

Especificamos uma variável entre parênteses ( )

Passamos alguns valores como condições para a variável, um case para cada opção

Para cada valor (cada case) especificamos um ou mais comandos a serem executados

Podemos especificar mais de um valor em um único Case, devemos apenas separar os valores por vírgula

Caso a variável não seja igual a nenhum dos valores, serão executados os comandos dentro da cláusula default

Cada cláusula case ou default é finalizada com um comando break;

As opções do comando switch devem ficar entre chaves { }

# IFS Aninhados X Case

```
#include <stdio.h>
#include <conio.h>
main()
{
    float A, B;
    int OP;
    clrscr();
    printf("Informe o Valor de A: \n");
    scanf("%f", &A);
    printf("Informe o Valor de B: \n");
    scanf("%f", &B);
    printf("Informe a Operacao Desejada \n");
    printf("1-Adicao, 2-Subtracao:\n");
    scanf("%i", &OP);
    if (OP == 1)
        printf("O resultado eh %f", A+B);
    else if (OP == 2)
        printf("O resultado eh %f", A-B);
    else
        printf("ERRO: Opcao %i invalida !", OP);
    getch();
}
```

```
#include <stdio.h>
#include <conio.h>
main()
{
    float A, B;
    int OP;
    clrscr();
    printf("Informe o Valor de A: \n");
    scanf("%f", &A);
    printf("Informe o Valor de B: \n");
    scanf("%f", &B);
    printf("Informe a Operacao Desejada \n");
    printf("1-Adicao, 2-Subtracao:\n");
    scanf("%i", &OP);
    switch (OP)
    {
        case 1:
            printf("O resultado eh %f", A+B);
            break;
        case 2:
            printf("O resultado eh %f", A-B);
            break;
        default:
            printf("ERRO: Opcao %i invalida !", OP);
            break;
    }
    getch();
}
```

# Laços de Repetição (Loopings)

## Repetição com Teste no Início

C: while - Portugol: enquanto

**while (Condição) Exemplo:**

```
{           a = 1
    comando1;   While (a <6)
    comando2;
    ---
    comando n;   printf("%d\n", a);
    a=a+1;
}
```

## Repetição com Teste no Final

C: do - Portugol: faça

**do Exemplo:**

```
{           a = 1
    comando1;   do
    comando2;
    ---
    comando n;   printf("%d\n", a);
    a=a+1;
} While (Condição); } While (a < 6);
```

## Repetição Automática

C: for - Portugol: para

**for (inicio; fim; incremento)**

```
{
    comando1;
    comando2;
    ---
    comando n;
}
```

**Exemplo:**

```
for (a=1; a<6; a++)
{
    printf("%d\n", a);
}
```

- Nos 3 Laços de Repetição, os comandos só serão executados enquanto a condição for verdadeira

- Os 3 trechos de código de exemplo fazem a mesma coisa e produzem o mesmo resultado

- O **do** tem por ponto e vírgula no final

- No **for** os parâmetros: início, fim e incremento são separados por ponto e vírgula.

## Exemplos de Laços de Repetição (Loopings)

```
a = 1
While (a <6)
{
    printf("%d\n", a);
    a=a+1;
}
```

**Executa 5 Vezes**

```
a = 1
do
{
    printf("%d\n", a);
    a=a+1;
} While (a < 6);
```

**Executa 5 Vezes**

```
for (a=1; a<6; a++)
{
    printf("%d\n", a);
}
```

**Executa 5 Vezes**

```
a = 7
While (a <6)
{
    printf("%d\n", a);
    a=a+1;
}
```

**Executa 0 Vezes**

```
a = 7
do
{
    printf("%d\n", a);
    a=a+1;
} While (a < 6);
```

**\*\* Executa 1 Vez \*\***

```
for (a=7; a<6; a++)
{
    printf("%d\n", a);
}
```

**Executa 0 Vezes**

» **while** - teste é feito no início, se a condição for falsa os comandos não serão executados

» **do** - teste é feito no final, executa a primeira vez para depois testar, com isso, mesmo que a condição seja falsa, os comandos serão executados ao menos uma vez

» **for** - teste, inicialização e incremento são feitos no início, se a condição for falsa os comandos não serão executados

» Para cada situação temos um tipo de looping mais adequado

## **Incrementos e Abreviações**

---

No laço de repetição **for** o incremento ou decremento é automático

Nos laços de repetição **do** e **while** o incremento ou decremento é manual

### **Principais abreviações:**

**a++;** é a mesma coisa que **a=a+1;**

**a--;** é a mesma coisa que **a=a-1;**

### **Outras abreviações:**

**a+=5;** é a mesma coisa que **a=a + 5;**

**a-=5;** é a mesma coisa que **a=a - 5;**

**a\*=5;** é a mesma coisa que **a=a \* 5;**

**a/=5;** é a mesma coisa que **a=a / 5;**

**Os mais usados são a++ e a--**

**Esta é uma das vantagens da linguagem C**

## Comando break

---

```
#include <stdio.h>
main()
{
    int A;
    clrscr();
    A=1;
    while (A<6)
    {
        printf("%d\n",A);
        if (A==3) break;
        A++;
    }
    getch();
}
```

- O comando **break** força a saída de qualquer laço de repetição
- Quando o comando **break** é executado, o programa sai do looping mesmo que a condição do looping seja verdadeira
- Vai para o próximo comando depois do final do looping, no exemplo ao lado vai para o **getch()**
- No exemplo o **break** é executado se A for igual a 3
- Podemos usar o **break** para: **while**, **do** ou **for**

## Exercício Resolvido:

Escreva 3 programas em C: um usando WHILE, outro usando DO e outro usando FOR.

Os programas devem somar todos os números inteiros de 1 até 50.

```
#include <stdio.h>
#include <conio.h>
main()
{
    int I=1, SOMA=0;
    clrscr();
    while (I<=50)
    {
        SOMA=SOMA+I;
        I++;
    }
    printf("SOMA = %i", SOMA);
    getch();
}
```

**Programa 1**  
**Usando WHILE**

```
#include <stdio.h>
#include <conio.h>
main()
{
    int I=1, SOMA=0;
    clrscr();
    do
    {
        SOMA=SOMA+I;
        I++;
    } while (I<=50);
    printf("SOMA = %i", SOMA);
    getch();
}
```

**Programa 2**  
**Usando DO**

```
#include <stdio.h>
#include <conio.h>
main()
{
    Int I, SOMA=0;
    clrscr();
    for (I=1;I<=50;I++)
    {
        SOMA=SOMA+I;
    }
    printf("SOMA = %i", SOMA);
    getch();
}
```

**Programa 3**  
**Usando FOR**

## Imprimindo Colorido

- A função printf imprime sempre da mesma cor
- A função cprintf imprime colorido
- Trabalha em conjunto com os comandos

`textcolor (No. Cor)`

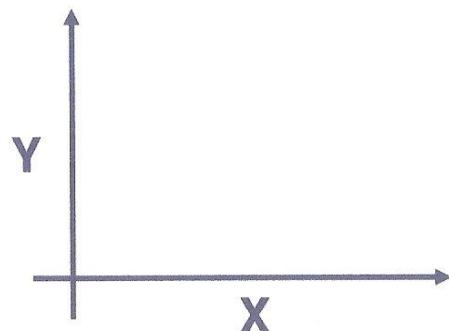
Cor do Texto

`textbackground (No. Cor)`

Cor de Fundo do Texto

`gotoxy (x,y)`

Define Coluna e Linha do Cursor



0	PRETO
1	AZUL
2	VERDE
3	CIANO
4	VERMELHA
5	MAGENTA
6	MARROM
7	CINZA CLARO
8	CINZA ESCURO
9	AZUL CLARO
10	VERDE CLARO
11	CIANO CLARO
12	ALARANJADO
13	MAGENTA CLARO
14	AMARELO
15	BRANCO

## **Imprimindo Colorido Exemplo**

---

```
#include <stdio.h>
#include <conio.h>
main()
{
    int cor;
    for (cor=0; cor <16; cor++)
    {
        gotoxy(20,cor+2);
        textcolor(cor);
        textbackground(15-cor);
        cprintf("Cor da Letra No. %i e Cor de Fundo No. %i",cor, 15-cor);
        getch();
    }
    getch();
}
```

# Trabalhando com Caracter e com String

## Caracter

char SEXO, LETRA, OP;	Só uma posição
SEXO = 'F';	Comando para ler “scanf()”
scanf("%c", &LETRA);	Comando para imprimir “printf()”
printf("A Opcao %c", "OP");	Formato “%c”

## String

char NOME[50] = "Paulo Silva", MSG[100];	Várias posições, [Tamanho]
scanf("%s", &MSG);	Comando adequado para ler “gets()”
gets(MSG);	Comandos para imprimir “puts()” ou “printf()”
printf("A Mensagem: %s", MSG);	Formato “%s”
puts(MSG);	

- O “scanf” lê strings desde que não tenham espaços em branco, o ideal é usar o “gets” para strings
- Não precisamos colocar “&” no comando “gets()”
- Não devemos combinar scanf e gets

## Programas de Exemplo usando Strings

### Programa c/ Strings correto

```
#include <stdio.h>
#include <conio.h>
main()
{
    char nome1[50], nome2[50], nome3[50];
    clrscr();
    printf("Informe um Nome Completo1: \n");
    gets(nome1);
    printf("Informe um Nome Completo2: \n");
    gets(nome2);
    printf("Informe um Nome Completo3: \n");
    gets(nome3);
    printf("O Nome1 lido eh: %s \n", nome1);
    printf("O Nome2 lido eh: %s \n", nome2);
    printf("O Nome3 lido eh: %s \n", nome3);
    getch();
}
```

Programa está correto, pois, usa a função gets(). Lê as strings corretamente

### Programa c/ Strings errado

```
#include <stdio.h>
#include <conio.h>
main()
{
    char nome1[50], nome2[50], nome3[50];
    clrscr();
    printf("Informe um Nome Completo1: \n");
    scanf("%s", &nome1);
    printf("Informe um Nome Completo2: \n");
    scanf("%s", &nome2);
    printf("Informe um Nome Completo3: \n");
    scanf("%s", &nome3);
    printf("O Nome1 lido eh: %s \n", nome1);
    printf("O Nome2 lido eh: %s \n", nome2);
    printf("O Nome3 lido eh: %s \n", nome3);
    getch();
}
```

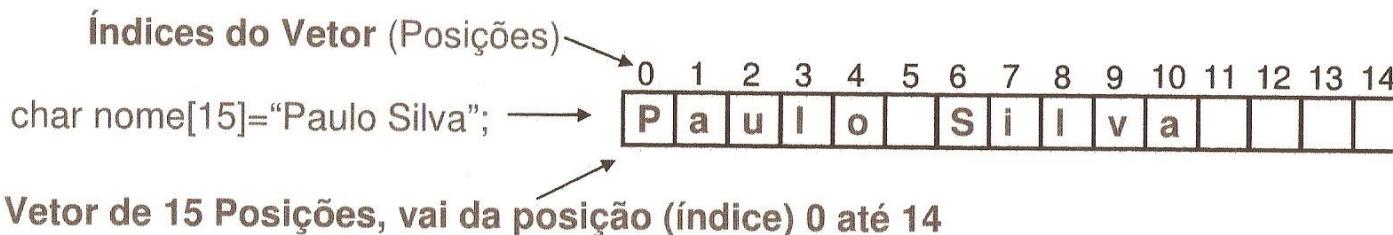
Programa está errado, pois, usa a função scanf(). Problemas para ler strings com espaço

## Entendendo Melhor uma String

Uma string em C é uma cadeia de caracteres, ou seja, um vetor de caracteres.

- Um “vetor de caracteres” é considerado uma “String” e tratamos com “%s”
- Cada “elemento” de um vetor de caracteres é “Char” e tratamos com “%c”
- Cada “índice” de um vetor é “Int” e tratamos com “%i”
- O 1º elemento de um vetor tem o índice=0(Zero)

Exemplo:



nome[1] == 'a'      Elemento com índice 1 = a  
nome[6] == 'S'      Elemento com índice 6 = S

## Programa de Exemplo usando String

O Programa abaixo cria e inicializa uma string (vetor de caracteres) e faz um looping para imprimir cada elemento do vetor e seu Índice.

- O Vetor inteiro é chamado “nome” e é do tipo String – “%s”
- Cada índice do vetor é chamado de “i” e é do tipo Inteiro – “%i”
- Cada elemento do vetor é chamado “nome[i]” e é do tipo Character – “%c”

```
#include <stdio.h>
#include <conio.h>
main()
{
    char nome[15] = "Paulo Silva";
    int i;
    printf("String: %s \n\n", nome);
    for (i=0;i<15;i++)
    {
        printf("Posicao %i Caracter %c de %s \n", i, nome[i], nome);
    }
    getch();
}
```

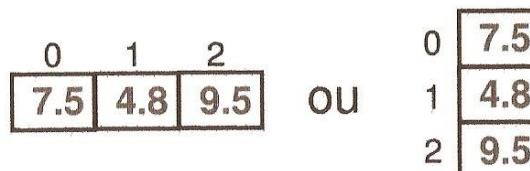
## Vetor

- Armazena um conjunto de dados do mesmo tipo
- Podemos criar vetores de vários tipos de dados: int, float, char etc...
- Um vetor do tipo “char” é considerado uma String
- Um vetor pode ser comparado a um conjunto de variáveis do mesmo tipo, porém quando temos muitos elementos é melhor usarmos um vetor ao invés de N variáveis
- O tamanho de um vetor é determinado pela quantidade de elementos em sua declaração. Tipo Nome-do-vetor[Tamanho]
- Acessamos cada elemento através de seu índice. Vetor [Índice]
- O 1º. Índice de um vetor é sempre Zero
- O vetor fica na memória do computador, podemos desenhá-lo na horizontal ou na vertical. Fica a seu gosto.

### Declaração de Vetor:

Tipo variável [tamanho];

- O trecho de código ao lado mostra um **Vetor** chamado NOTAS do tipo **float** com **3 posições**



```
main()
{
    float NOTAS[3];
    NOTAS[0]=7.5;
    NOTAS[1]=4.8;
    NOTAS[2]=9.5;
}
```

## Exercício Resolvido:

**Escreva um programa em C para ler as notas de 10 Alunos. Calcular a Média Final da turma. Imprimir as notas lidas e a média da turma. Se média for igual ou superior a 7 imprimi-la de azul, caso contrário imprimir de vermelho. Dica Você vai usar um vetor.**

```
#include <stdio.h>
#include <conio.h>
main()
{
    float NOTAS[10], SOMA=0, MEDIA;
    int I;
    clrscr();
    for (I=0; I<10; I++)
    {
        printf("Informe a nota do %i o. aluno:\n", I+1);
        scanf("%f", &NOTAS[I]);
        SOMA=SOMA+NOTAS[I];
    }
    MEDIA=SOMA/10;
    clrscr();
    for (I=0; I<10; I++)
    {
        printf("Nota aluno %i %f\n", I+1, NOTAS[I]);
    }
    if (MEDIA >=7)
        textcolor(1);
    else
        textcolor(4);
    cprintf("Media da turma = %f", MEDIA);
    getch();
}
```

O programa ao lado não trata as notas digitadas erradas. Ele deveria aceitar somente notas entre 0 e 10.

Para fazer esta validação, insira o um looping conforme trecho de código abaixo:

```
do
{
    printf("Informe a nota do %i o. aluno:\n", I+1);
    scanf("%f", &NOTAS[I]);
} while (NOTAS[I] >= 0 ^ NOTAS[I] <=10);
```

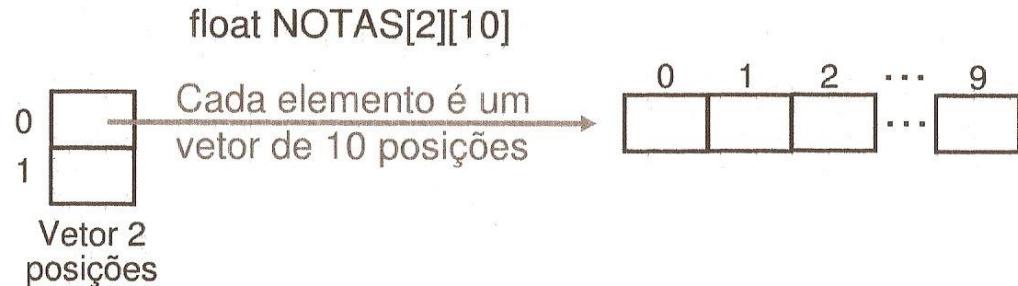
# Matriz

- Podemos criar vetores de várias dimensões
- Quando um vetor tem mais de uma dimensão é chamado de matriz
- Cada dimensão pode ter tamanhos diferentes, porém todas são do mesmo tipo
- Acessamos cada elemento da matriz passando os índices de cada dimensão
- Uma matriz armazena um conjunto de vetores
- Podemos ver as matrizes como um vetor dentro de outro

**Declaração de Matriz:**

Tipo variável [Dim1] [Dim2]... [DimN];

Exemplo:



■ O trecho de código ao lado mostra uma **Matriz** chamada NOTAS do tipo **float** com **2 Dimensões**:  
1<sup>a</sup>. Dim c/ **2 posições** e 2<sup>a</sup>. Dim c/ **3 posições**

	0	1	2
0	7.5	4.8	9.5
1	6.5	8.5	6

```
main()
{
    float NOTAS[2][3];
    NOTAS[0][0]=7.5;
    NOTAS[0][1]=4.8;
    NOTAS[0][2]=9.5;
    NOTAS[1][0]=6.5;
    NOTAS[1][1]=8.5;
    NOTAS[1][2]=6;
}
```