



**TECNOLOGIA E INOVAÇÃO
EM PROL DA INDÚSTRIA**



Técnico em Informática

Programação de aplicativos – 140h

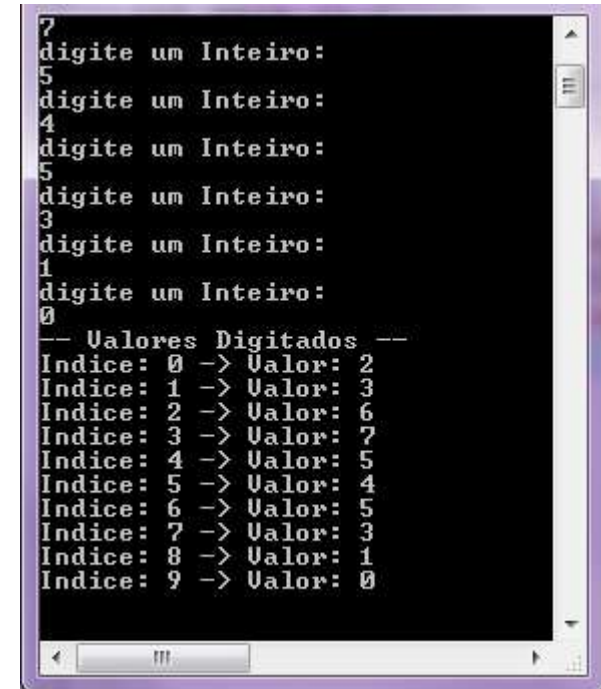
Prof^a: Francisleide Almeida

Exercício1

- Construa um programa que declare e receba um vetor de inteiros com 10 elementos com números fornecidos pelo usuário, através da entrada padrão e depois exiba os índices e seus valores armazenados.

Resposta

```
#include <stdio.h>
int main () {
int vetorInteiros[10];
for(int x=0; x<10; x++){
    printf("digite um Inteiro: \n");
    scanf("%d",&vetorInteiros[x]);
    if(x==9){
        printf("-- Valores Digitados -- \n");
        for(int y=0; y<10; y++){
            printf("Indice: %d -> Valor: %d \n", y, vetorInteiros[y]); } } }
getchar();
return(0); }
```



```
7
digite um Inteiro:
5
digite um Inteiro:
4
digite um Inteiro:
5
digite um Inteiro:
3
digite um Inteiro:
1
digite um Inteiro:
0
-- Valores Digitados --
Indice: 0 -> Valor: 2
Indice: 1 -> Valor: 3
Indice: 2 -> Valor: 6
Indice: 3 -> Valor: 7
Indice: 4 -> Valor: 5
Indice: 5 -> Valor: 4
Indice: 6 -> Valor: 5
Indice: 7 -> Valor: 3
Indice: 8 -> Valor: 1
Indice: 9 -> Valor: 0
```

Exercício2

- Construa um programa que declare e receba um vetor de inteiros com 10 elementos e que o conteúdo de cada posição do vetor será o seu índice ao quadrado. Criei um **for** para receber o conteúdo e outro para imprimir.

Resposta

```
#include <stdio.h>

int main ( ) {

    int i;

    int vetor[10]; // declara um vetor de inteiros

    // Insere o quadrado dos números de 0 a 9 em cada posição do vetor
    for ( i = 0 ; i < 10 ; i ++ ) {
        vetor[i] = i * i;
    }

    //Exibe o conteúdo do vetor
    for ( i = 0 ; i < 10 ; i ++ ) {
        printf("\n%d" , vetor[i] );
    }

    printf("\n");
}
```

Exercício3

- Faça um programa que leia um vetor tamanho 10 com dados obtidos a partir do usuário e exiba os valores recebidos de forma invertida.

Resposta

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

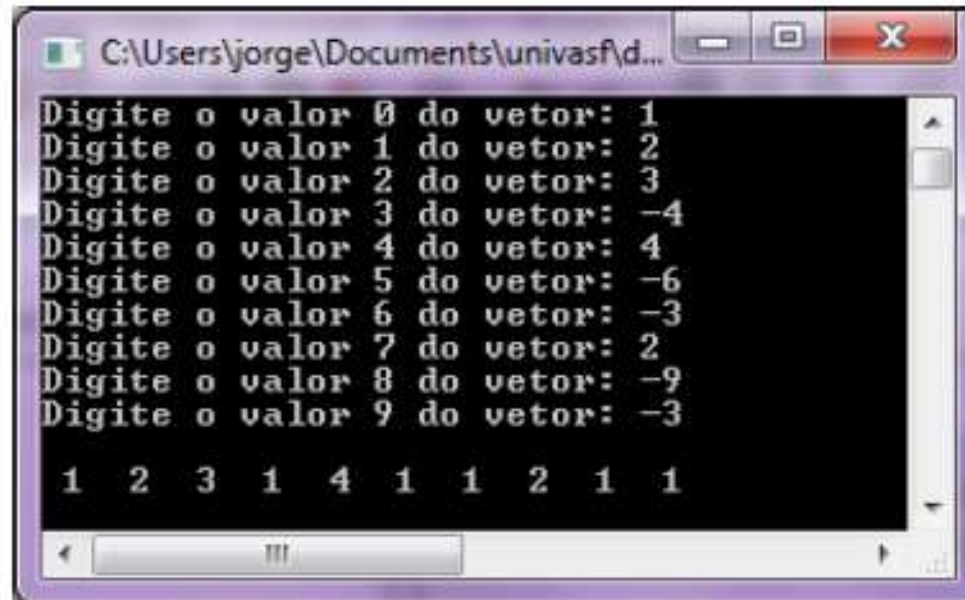
int main () {
    int vet1[10];
    for (int x=0; x<10;x++)
    {
        printf ("Indice: %d - Digite um inteiro:", x);
        scanf ("%d", &vet1[x]);
        if (x==9){
            printf("\n \n -- Valores digitados --\n \n Vetor 2(Invertido)\n \n");
            for (int y=9; y>=0; y--){
                printf ("Indice: %d - Valor %d \n", y, vet1[y]);
            }
        }
    }
    getch();
    return(0);
}
```

Exercício4

- Faça um programa que leia um vetor de 10 posições e crie um segundo vetor substituindo os valores negativos por 1. Exiba os resultados do segundo vetor.

Resposta

```
#include <stdio.h>
#include <conio.h>
int main () {
    int vet[10];
    int i=0;
    do {
        printf ("Digite o valor %d do vetor: ", i);
        scanf ("%d", &vet[i]);
        i++; }
    while (i<=9);
    printf ("\n");
    for (i=0; i<10; i++){
        if (vet[i]<0)
            vet[i]=1;
        printf(" %d ", vet[i]);
    }
    getch();
    return(0);
}
```



```
C:\Users\jorge\Documents\univas\d...
Digite o valor 0 do vetor: 1
Digite o valor 1 do vetor: 2
Digite o valor 2 do vetor: 3
Digite o valor 3 do vetor: -4
Digite o valor 4 do vetor: 4
Digite o valor 5 do vetor: -6
Digite o valor 6 do vetor: -3
Digite o valor 7 do vetor: 2
Digite o valor 8 do vetor: -9
Digite o valor 9 do vetor: -3

1 2 3 1 4 1 1 2 1 1
```

Strings

Strings

- Uma **string** é uma cadeia ou sequência de caracteres. As strings são usadas para armazenar nomes, palavras e frases.
 - Na linguagem C strings são vetores de caracteres que possuem um caractere que indica o término de seu conteúdo, o caractere nulo '\0' (contra barra zero).
 - **Declaração de strings**
 - Como a string possui o caractere nulo para delimitar o final do seu conteúdo, o tamanho da string deve ser definido com um caractere a mais do que será efetivamente necessário.
- Sintaxe:** char identificador-da-string [tamanho+1];

Strings

- Ex:

`char vetc [6];`

vetc é um vetor de caracteres (string) de tamanho 6. Pode receber uma palavra de no máximo 5 letras

Inicialização de Strings

- Uma string pode ser inicializada na sua declaração com uma sequência de caracteres entre chaves e separadas por vírgula.

```
char vetc[6]= {'T', 'e', 'x', 't', 'o', '\0'};
```

- *Lembre-se que o compilador só reconhecerá um caractere se este estiver entre aspas simples, logo usar uma atribuição do tipo {t,e,x,t,o,\0} ou {texto\0} **irá gerar um erro de compilação.***

Inicialização de Strings

- Uma string pode também ser inicializada por uma sequência de caracteres entre aspas duplas. Neste caso, não é necessário o uso de aspas simples e virgulas, o compilador C coloca automaticamente o '\0' no final.

```
char vetc[6] = "Texto";
```

- Assim como vetores e matrizes, na inicialização de uma string o seu tamanho pode ser omitido.

```
char vetc[ ] = "Texto";
```

```
/* vetor não-dimensionado, o compilador coloca  
automaticamente o '\0' no final */
```

Leitura de Strings

- *Utilizando a função **scanf()***
- A sintaxe para receber uma string por meio da *scanf()* é:
scanf(“%s”, nome_da_string);
- *OBS.: não é necessário colocar o operador &, pois o nome da string em si já é um endereço de memória.*

Leitura de Strings

- **Exemplo:** Crie um aplicativo em C que peça ao usuário seu nome, armazene em uma String, peça o sobrenome, armazene em outra string e exiba o nome do usuário de maneira formal (Sobrenome, Nome).

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    char nome[21], sobrenome[21];

    printf("Primeiro nome: ");
    scanf("%s", nome);

    printf("Ultimo sobrenome: ");
    scanf("%s", sobrenome);

    printf("Ola senhor %s, %s. Bem-vindo ao curso de linguagem C.\n", sobrenome, nome);

    system("pause");
}
```


Leitura de Strings

- O que aconteceria se fosse digitado um nome composto no 'nome' ou no 'sobrenome'?
- *A `scanf()` vai simplesmente cortar seu nome composto. Essa função pega tudo até encontrar um espaço em branco, caractere *new line* `\n`, tab ou ENTER*

Strings

- Tente compilar o código a seguir:

```
#include <stdio.h>
#include <stdlib.h>
int main ()
{
    char nome[] = "fulano";
    char sobrenome[] = "de tal";
    char nomeCompleto[] = nome + sobrenome;
    int i=0;

    while(nomeCompleto[i] != '\0'){
        printf("%c",nomeCompleto[i]);
        nomeCompleto[i++];
    }

    return(0);
    system("pause");
}
```

Strings

- Porque não compila:
 - Um erro muito comum no uso de string em C esta sendo cometido na linha 7. `char nomeCompleto[] = nome + sobrenome`. String não poder ser concatenadas utilizando o operador `+`. Existe uma diretiva em C que implementa diversas funções de manipulação de valores em string.

Strings

- ***Funções para Manipulação de Strings***
 - **Get String (gets)**

A função **gets()** lê os caracteres do dispositivo padrão de entrada (teclado) para o seu argumento – um vetor do tipo **char** – até que um caractere de nova linha ou o indicador de fim de arquivo seja encontrado. Um caractere **NULL** (`'\0'`) será adicionado ao vetor quando a leitura terminar. Sua forma geral é:

gets (nome_da_string);

Strings

- ***Funções para Manipulação de Strings***

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    char string[100];

    printf("Digite o seu nome: ");
    gets(string);
    printf ("\n\n Ola %s\n",string);

    system("pause");
}
```

OBS.: Mais uma vez,
nunca use & quando
for armazenar uma
string.

Como o primeiro argumento da função **printf()** é uma string também é válido fazer:

`printf (string);` /* isto simplesmente imprimirá a string. */

Strings

- **Get String (gets)**
- Não é uma função segura, pois o tamanho da string não é especificado. A função *scanf()* pega tudo até aparecer o primeiro espaço em branco, e pára. Já a *gets()* não, ela pega tudo até aparecer uma *new line* `\n`, inclusive nada. Ou seja, se você der um ENTER, a *gets()* vai armazenar esse enter na string.

Strings

- ***Funções para Manipulação de Strings***
 - **Get String (gets) e Scanf**

```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
    char nome[31], sobrenome[31], nascimento[11];
    int idade;

    printf("Nome: ");
    gets(nome);

    printf("Sobrenome: ");
    gets(sobrenome);

    printf("Idade: ");
    scanf("%d", &idade);

    printf("Data de nascimento: ");
    gets(nascimento);

    printf("\nNome completo: %s %s\n", nome, sobrenome);
    printf("Idade: %d\n", idade);
    printf("Data de nascimento: "); puts(nascimento);

    system("pause");
}
```

A função *scanf()* pega tudo até aparecer o primeiro espaço em branco, e pára antes dele. Já a *gets()* não, ela pega tudo até aparecer uma *new line* `\n`, inclusive nada. Ou seja, se você der um ENTER, a *gets()* **vai armazenar esse ENTER na string.**

Strings

- ***Funções para Manipulação de Strings***
 - **Get String (gets) e Scanf**

O problema é que a função *gets()* vai pegar o que está armazenado nesse buffer e vai armazenar o que estiver lá na string de data de nascimento!
E como evitar isso? É só apagar esse ENTER que está no buffer, usando o *fflush(stdin)* caso use Windows, ou *__fpurge(stdin)* caso use Linux.

```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
    char nome[31], sobrenome[31], nascimento[11];
    int idade;

    printf("Nome: ");
    gets(nome);

    printf("Sobrenome: ");
    gets(sobrenome);

    printf("Idade: ");
    scanf("%d", &idade);

    fflush(stdin);

    printf("Data de nascimento: ");
    gets(nascimento);

    printf("\nNome completo: %s %s\n", nome, sobrenome);
    printf("Idade: %d\n", idade);
    printf("Data de nascimento: "); puts(nascimento);

    system("pause");
}
```

Strings

- ***Funções para Manipulação de Strings***
 - **Get String (gets) e Scanf**

É possível alterar o funcionamento da ***scanf()***. Por exemplo, se quisermos ler strings que tenham espaço, nós temos que dizer isso dentro da função. Para dizer para a ***scanf()*** parar de pegar nossa string somente quando encontrar um caractere de NEW LINE (um enter). Usamos o operador: `[\n]` Logo, nosso código da ***scanf()*** para ler strings com espaços e armazenar na variável "str" é:

```
scanf ( "[%\n]", str);
```

Strings

- ***Funções para Manipulação de Strings***
 - **Get String (gets) e Scanf**
- Podemos ainda limitar o tamanho de nossa string, basta colocar um numero inteiro ao lado do %, representando o número de caracteres máximo, o que é uma excelente prática, pois essa função pode ocasionar problemas na memória, caso você estoure os limites da string.

Por exemplo:

```
scanf ( "%256[^\n]" str);
```


Strings

- ***Funções para Manipulação de Strings***

- **Get String (gets) e Scanf**

função ***gets()*** peca nesse quesito, de tamanho da string, pois podemos digitar mais caracteres do que a string alocou de memória, e "quebraríamos" o programa por conta de um *overflow*. Uma solução para isso é usar a função ***fgetc()***, que é mais segura. Ela recebe três dados:

1. A string que vai armazenar o que vai ser digitado (no nosso caso é a variável "str");
2. O tamanho da string e de onde vai ler (ela pode ler de um arquivo de texto, por exemplo);
3. Para ler do teclado, usamos ***stdin***.

Por exemplo:
fgetc(str, 256, stdin);

Strings

- ***Funções para Manipulação de Strings***

- **Outras funções**

- Incluir a biblioteca **string.h**
 - *Strings não podem ser comparadas com o operador de comparação padrão (==), neste caso deve-se usar função **strcmp()** ou a função **stricmp()**.*
 - **strcmp(s1,s2)** – Retorna 0 se s1 e s2 são iguais; menor que 0 se s1<s2; maior que 0 se s1>s2 (comparação alfabética).
 - **stricmp(s1,s2)** – Retorna 0 se s1 e s2 são iguais; menor que 0 se s1<s2; maior que 0 se s1>s2 (comparação alfabética). Essa função considera letras maiúsculas ou minúsculas como símbolos iguais.

Strings

- ***Funções para Manipulação de Strings***

- **Outras funções**

- Incluir a biblioteca **string.h**
 - *Strings não podem ser atribuídas com o operador de atribuição (=), para uma atribuição usa-se a função **strcpy()**.*

strcpy(s1,s2) - Copia s2 em s1.

- OBS.: A string-destino deve ser grande o suficiente para armazenar a string origem e seu caractere **NULL** de terminação que também é copiado.

Strings

- *Strings não podem ser concatenadas com o operador (+), para tal usa-se a função **strcat()**.*
 - **strcat(s1,s2)** – Concatena s2 ao final de s1.
 - **strlen(s)** – Retorna o número de caracteres em s (sem contar o caracter nulo (/0)).

Atividades

- 1 - Construa um programa que leia um nome com no máximo 15 caracteres e imprimir esse nome tantas vezes quanto fores seus caracteres.
- 2 - Construa um programa que leia um nome com no máximo 15 caracteres e quantas letras "A" (maiúsculo ou minúsculo) tem no nome.
- 3 - Construa um programa que leia uma sigla do estado de uma pessoa e imprima uma das mensagens: Carioca (caso seja informado RJ ou rj), Paulista (Caso seja informado SP ou sp), Mineiro (caso seja informado MG ou mg) ou Outros Estados (Caso não seja nenhuma das opções anteriores). Utilize a função strcmp para comparar as siglas maiúsculas e minúsculas.
- 4 - Construa um programa para Preencher um vetor com 5 números inteiros, solicitados no teclado e mostrar outro vetor com o cubo dos números do primeiro vetor.
- 5 - Construa um programa para receber um nome do teclado e imprimi-lo de trás pra frente.