

ESTRUTURA DE DADOS

Dados Estruturados

VETORES E MATRIZES

Vetores

- Vetores, matrizes bidimensionais e matrizes de qualquer dimensão são caracterizadas por terem todos os elementos pertencentes ao mesmo tipo de dado.
- Um vetor tem a seguinte forma geral:

tipo_da_variável nome_da_variável [tamanho];

Exemplo: **float** exemplo[20];

O C irá reservar $4 \times 20 = 80$ bytes. Estes bytes são reservados de maneira contígua.

Vetores

- Na linguagem C a numeração começa sempre em zero. Isto significa que, no exemplo, os dados serão indexados de 0 a 19. Para acessá-los vamos escrever:

exemplo[0]

exemplo[1]

.

.

.

exemplo[19]

```
#include <stdio.h>

int main () {
    int num[100]; /* Declara um vetor de inteiros de 100 posicoes */
    int count = 0;
    int totalnums;

    do {
        printf ("\n Entre com um numero (-999 p/ terminar): ");
        scanf ("%d", &num[count]);
        count++;
    } while (num[count-1] != -999);

    totalnums = count - 1;
    printf ("\n\n\n\t Os números que você digitou foram:\n\n");

    for (count = 0; count < totalnums; count++)
        printf (" %d", num[count]);

    return (0);
}
```

Vetores

- **Obs:** No último exemplo, se o usuário digitar mais de 100 números, o programa tentará ler normalmente, mas o programa os escreverá em uma parte não alocada de memória, pois o espaço alocado foi para somente 100 inteiros. Isto pode resultar nos mais variados erros no instante da execução do programa...

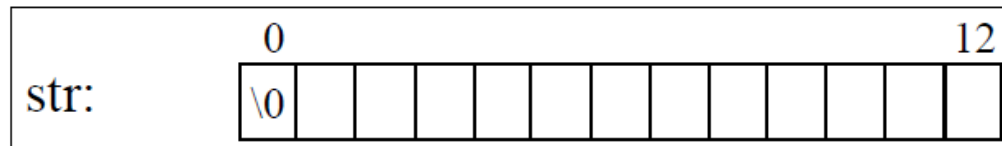
STRINGS

Strings

- Strings são vetores de **chars**. A **declaração geral** para uma string é:
char nome_da_string [tamanho];
- Devemos apenas ficar atentos para o fato de que as strings têm o seu último elemento como um '\0'.
- Devemos lembrar que o tamanho da string deve incluir o '\0' final.

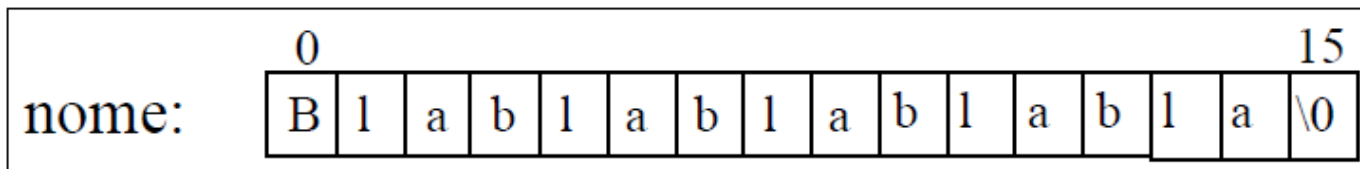
Ex: char str[13];

Define um string de nome “str” e reserva para ele um espaço de 13 (12 + ‘\0’) bytes na memória



Strings

- Ex: `char nome[16] = "Blablablabla";`
 - Define um string de nome "nome", reserva para ele um espaço de memória de 16 (15 + '\0') bytes e inicia-o com o texto indicado



- Os caracteres podem ser individualmente acessados por indexação.
Ex: `nome[0] = 'B'; nome[10] = 'l'`

Strings

- A biblioteca padrão do C possui diversas funções que manipulam strings. Estas funções são úteis pois não se pode, por exemplo, igualar duas strings:

string1 = string2; / **NÃO faça isto** */*

Fazer isto é um desastre!!!

Funções para Manipulação de Strings

- **gets**: a função gets() lê uma string do teclado. Sua forma geral é:

gets (nome_da_string);

Exemplo

```
#include <stdio.h>
void main (){
    char string[100];
    printf ("Digite o seu nome: ");
    gets (string);
    printf ("\n\n Ola %s", string);
}
```

Funções para Manipulação de Strings

- **strcpy:** a função `strcpy()` copia a string-origem para a string-destino. Sua forma geral é:

strcpy (string_destino, string_origem);

- **strlen(str):** retorna o tamanho de str
- **strcat(dest, fonte):** concatena fonte no fim de dest
- **strcmp(string1,string2):** compara duas strings. Se as duas forem idênticas, a função retorna zero. Se elas forem diferentes, a função retorna não-zero.

As funções apresentadas estão no arquivo cabeçalho *string.h*.

Funções para manipulação de strings

Exemplo 1 – strcpy, gets

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

```
void main (){
```

```
    char str1[100], str2[100], str3[100];
```

```
    printf ("Entre com uma string: ");
```

```
    gets (str1);
```

```
    strcpy (str2, str1); /* Copia str1 em str2 */
```

```
    strcpy (str3, "Voce digitou a string ");
```

```
    /* Copia "Voce digitou a string" em str3 */
```

```
    printf ("\n\n%s%s", str3, str2);
```

```
}
```

Funções para manipulação de strings

Exemplo 2 - strcat

```
#include <stdio.h>
#include <string.h>
void main (){

    char str1[100], str2[100];

    printf ("Entre com uma string: ");
    gets (str1);
    strcpy (str2, "Voce digitou a string ");
    strcat (str2, str1);
    /* str2 armazenara' Voce digitou a string + o conteudo de str1 */
    printf ("\n\n%s", str2);
}
```

Funções para manipulação de strings

Exemplo 3 - strlen

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

```
void main (){
```

```
    int size;  
    char str[100];
```

```
    printf ("Entre com uma string: ");  
    gets (str);  
    size = strlen (str);  
    printf ("\n\n A string que voce digitou tem tamanho %d", size);
```

```
}
```

Funções para manipulação de strings

Exemplo 4 - strcmp

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

```
void main (){
```

```
    char str1[100], str2[100];
```

```
    printf ("Entre com uma string: ");
```

```
    gets (str1);
```

```
    printf ("\n\n Entre com outra string: ");
```

```
    gets (str2);
```

```
    if (strcmp(str1,str2))
```

```
        printf ("\n\n As duas strings são diferentes.");
```

```
    else
```

```
        printf ("\n\n As duas strings são iguais.");
```

```
}
```

MATRIZES BIDIMENSIONAIS

Matrizes bidimensionais

- Forma geral da declaração

tipo_da_variável nome_da_variável [altura][largura];

Exemplo

```
#include <stdio.h>
void main (){
    int mtrx [20][10];
    int i, j, count;
    count = 1;
    for (i=0; i<20; i++)
        for (j=0; j<10; j++){
            mtrx[i][j] = count;
            count++;
        }
}
```

Matrizes multidimensionais

- Uma matriz *N-dimensional* funciona basicamente como outros tipos de matrizes. Basta lembrar que o índice que varia mais rapidamente é o índice mais à direita.
- A lista de valores é composta por valores (do mesmo tipo da variável) separados por vírgula. Os valores devem ser dados na ordem em que serão colocados na matriz.
- Exemplo

```
float vect [6] = { 1.3, 4.5, 2.7, 4.1, 0.0, 100.1 };  
int matrix [3][4] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12 };  
char str [10] = { 'J', 'o', 'a', 'o', '\0' };  
char str [10] = "Joao";  
char str_vect [3][10] = { "Joao", "Maria", "Jose" };
```

Matrizes multidimensionais

- Inicialização sem especificação de tamanho
`char mess [] = "Linguagem C: flexibilidade e poder.";`
`int matrix [][] = { 1,2,2,4,3,6,4,8,5,10 };`
- No primeiro exemplo, a string *mess* terá tamanho 36. No segundo exemplo o valor não especificado será 5.
- O compilador C vai, neste caso verificar o tamanho do que você declarou e considerar como sendo o tamanho da matriz.
- Isto ocorre na hora da compilação e não poderá mais ser mudado durante o programa, sendo muito útil, por exemplo, quando vamos inicializar uma string e não queremos contar quantos caracteres serão necessários.

Lista 3 - Exercícios