AED1

Vetores e
Um pouco de Strings

Profa.: Márcia Sampaio Lima

EST - UEA

Matrizes

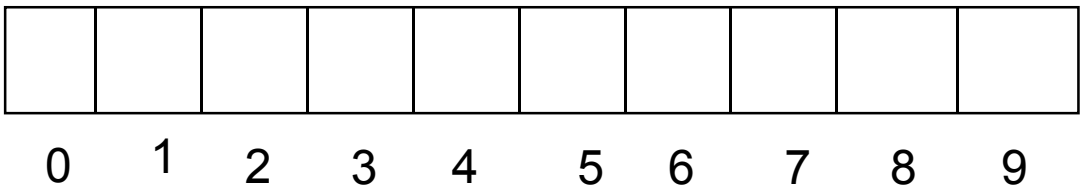
- A matriz é um tipo de dado usado para representar uma certa quantidade de variáveis que são referenciadas pelo mesmo nome.
- Consiste em locações contíguas de memória.
- O endereço mais baixo corresponde ao primeiro elemento.
- A matriz é um conjunto ordenado de dados que possuem o mesmo tipo.

Matriz unidimensional

- Sintaxe: `tipo nome[tamanho];`
- As matrizes têm 0 como índice do primeiro elemento, portanto sendo declarada uma matriz de inteiros de 10 elementos, o índice varia de 0 a 9.

Matriz unidimensional

- Matrizes unidimensionais são:
  - Variáveis, Compostas, Homogêneas, Identificadas pelo mesmo nome ....
  - Individualizados por um índice → cada elemento do vetor é identificado por um índice.



Vnotas

Vnotas [0]
Vnotas [1]
Vnotas [2]
...
Vnotas [9]

Matriz unidimensional

- Quando o compilador C encontra uma declaração de matriz ele reserva um espaço na memória do computador suficientemente grande para armazenar o número de células especificadas em tamanho.
- Por exemplo:

```
float exemplo[20];
```

Matriz unidimensional

- C irá reservar
  - **4 × 20 = 80 bytes**.
  - Estes bytes são reservados de maneira contígua.
  - Neste exemplo, os dados serão indexados de 0 a 19.
  - Para acessá-los escrevemos:

```
exemplo[0]
exemplo[1]
..
exemplo[19]
```

Matriz unidimensional

- Mas ninguém o impede de escrever:
  `exemplo[30]`
  `exemplo[103]`
- Por quê?
  - C não verifica se o índice está dentro dos limites válidos.
  - Se o programador não tiver atenção com os limites de validade pode:
    - ter dados sobrescritos ou de ver o computador travar.
    - Vários erros sérios (*bugs*) podem surgir.

Matriz unidimensional

```
#include <stdio.h>
  main( )
  {
      int x[10];
      int t;
      for(t=0;t<10;t++)
      {
            x[t] = t*2;
            printf ("%d\n", x[t]);
      }
  }
```

Matriz unidimensional

```
#include <stdio.h>
   main( )
   {
        int x[10];
        int t;
        for(t=0;t<10;t++)
        {
              x[t] = t*2;
              printf ("%d\n", x[t]);
        }
         printf ("%d\n", x[100]);
   }
```

Matrizes

- Qual o erro?
  - ```
    # include <stdio.h>
    main( )
    {
      int erro[10], i;
      for( i = 0; i<100; i++)
      {
              erro[i]=1;
              printf (" %d\n ", erro[i] );
      }
    }
    ```
    ```

Matrizes

- Qual o erro?
  - **# include** <stdio.h>

```
main( )
{
   int erro[10], i;
   for( i = 0; i<100; i++)
   {
            erro[i]=1;
            printf (" %d\n ", erro[i] );
   }
}
```

Matrizes unidimensional

- **Uso de Constantes para definir o tamanho de um vetor**

```
#define TAM_MAX 10
 main(){
 double VetReais[TAM_MAX];
 for(int i=0; i< TAM_MAX; i++)
       VetReais[i] = TAM_MAX - i;
 }
```

Matrizes unidimensional

- É possível inicializar o vetor no momento de sua declaração.
- `nome_vetor[tamanho]={lista_de_valores};`
- Todos os elementos da lista de valores devem ser separados por virgula e serem todas do mesmo tipo de dados especificado.

Matrizes unidimensionais

```
#include <stdio.h>
main( )
{
int i;
 int x[10] ={0,1,2,3,4,5,6,7,8,9};
for(i=0; i<10; i++)
  printf("%d\n", x[i] );
}
```

Matrizes unidimensionais

- Os vetores são muito usados para criar uma string de caracteres, pois em C não existe nenhum tipo de dados para definir uma string.
- A declaração de um vetor contendo uma string sempre deve ser maior que o número de caracteres, pois o compilador acrescenta automaticamente no final da string um espaço nulo que indica o seu término

Um pouco de Strings

- Uma **string** é uma matriz tipo **char** que termina com **'\0'**.
- O caractere **'\0'** tem o código numérico **00**, portanto pode-se verificar o final de uma **string** procurando o valor numérico **zero** armazenado na matriz de caracteres.
- Constantes **strings** são uma lista de caracteres que aparecem entre aspas, não sendo necessário colocar o **'\0'**, que é colocado pelo compilador.

Matrizes unidimensionais

- Declaração e inicialização de um vetor de string.

```
#include<stdio.h>
 void main()
{
   char c1 = 'a';
   char vet1[30]="Aprendendo a mexer com string\n";
   printf("A variavel c1 do tipo char : %c\n",c1);
   printf("O vetor do tipo char contem a string: %s",vet1);

} /*fim do programa*/
```

Matrizes unidimensionais

- **FUNÇÃO GETS()**

  Sintaxe: `gets(nome_matriz);`

  - É utilizada para leitura de uma **string** através do teclado, até que a tecla <ENTER> seja pressionada.
  - A função **gets()** não testa limites na matriz em que é chamada.

Exemplo 1

```
#include<stdio.h>
main(){
   char nome[10];
   printf("Informe o nome:");
   gets(nome);
   printf("%s", nome);
}
```

Exemplo 2

```
#include<stdio.h>
main(){
   char nome[10];
   printf("Informe o nome:");
   gets(nome);
   for (int i = 0; i <10; i++){
       if (nome[i] != 0)
         { printf("%c", nome[i]);
         }
         else{break;
         }
   }
}
```

Exercício

- Encontrar o maior valor dentro de um vetor
- Faça um programa em C que leia uma seqüência de caracteres de tamanho indeterminado e informe a quantidade de caracteres desta seqüência.
  - **Sabendo que o último caractere de uma string é '\0' ou 0.**

Matrizes unidensionais

- Passagem por parâmetro
  - Para passar uma matriz ou vetor como parâmetro, basta declarar o parâmetro da mesma forma que a matriz foi declarada.

Matrizes unidensionais

```
#define TAM_MAX 10
   void ImprimeVet (int Tam, int Vet[TAM_MAX])
   {    int i;
        for (i=0; i< Tam; i++)
           { printf("%d", Vet[i]);
           }
        }

        void main()
        {
          int Notas[TAM_MAX];
           ImprimeVet(TAM_MAX, Notas); // Passa o vetor
   'Notas' como parâmetro
        }
```

- A passagem dos elementos de um vetor como parâmetro é idêntica à passagem de uma variável.

Matrizes unidemsionais

- Um vetor é sempre passado por referência, logo, qualquer alteração em seus elementos altera a variável usada como parâmetro na chamada da rotina.

```
...
printf("\nSoma: %d", soma(idade));
....
int soma(int vet[5]){
   int total =0;
   for (int i = 0; i<5; i++){
        total += vet[i];
   }
   vet[3]=0;
   return total;

}
```

Exercício

- Faça um programa em C que declare dois vetores , cada um com tamanho 20. Em seguida faça uma função que receba os vetores (um por vez), calcule e retorne a média dos elementos dos vetores.

Matrizes unidimensionais

- **Inicialização sem especificação de tamanho**
  - Em alguns casos, inicializar matrizes das quais não sabemos o tamanho *a priori*.
  - O compilador C vai, neste caso verificar o tamanho declarado e considerar como sendo o tamanho da matriz.
  - Isto ocorre na hora da compilação e não poder ser mudado durante o programa.

Matrizes unidimensionais

- **Inicialização sem especificação de tamanho**
  - Útil quando vamos inicializar uma string e não queremos contar quantos caracteres serão necessários. Alguns exemplos:

```
char mess [] = "Linguagem C:
flexibilidade e poder.";
```

  - A string mess terá tamanho 36.

Um pouco de string

- Função puts()
  - Escreve o seu argumento no dispositivo padrão de saída (vídeo), coloca um ’**\n**’ no final.
  - `puts(“menssagem");`

Um pouco de string

- Função strlen()
  - Retorna o tamanho de uma string fornecida
  - Definida na biblioteca string.h
  - Exemplo:

  - `printf("\n%d",strlen(mess));`

Um pouco de String...

- **FUNÇÃO STRCPY()**
  Sintaxe: `strcpy(destino, origem);`

  - Faz parte da biblioteca **string.h**.
  - Copia o conteúdo de uma **string** para uma variável tipo **string** (um vetor de **char**).
  - ```
    # include <stdio.h>
    # include <string.h>
    main()
    {
    char str[20];
    strcpy(str,"alo");
    puts(str);
    }
    ```
    ```

Um pouco de String...

- Sintaxe: strcat(string1, string2);
  - Concatena duas strings. Não verifica tamanho.
  - ```
    #include <string.h>
    #include <stdio.h>
    main()
    {
    char um[20], dois[10];
    strcpy (um, "bom");
    strcpy (dois," dia");
    strcat (um, dois); // une a string dois à string um
    printf ("%s\n", um);
    }
    ```
    ```

Um pouco de String...

- **FUNÇÃO STRCMP()**
  Sintaxe: strcmp(s1, s2);

- Essa função compara duas strings, se forem iguais devolve 0.

```
main( )
{
     char s[80];
     printf("Digite a senha:");
     gets(s);
     if (strcmp(s,"laranja")) printf("senha inválida\n");
     else printf("senha ok!\n") ;
}
```

Matrizes multidimensional

- Sintaxe: `tipo nome[tamanho][tamanho];`

- A matriz multidimensional funciona como a matriz de uma dimensão (vetor), mas tem mais de um índice.
- As dimensões são declaradas em seqüência entre colchetes.
- A verificação de limites não é feita pela linguagem, nem mensagem de erros são enviadas, o programa tem que testar os limites das matrizes.

Matrizes multidimensional

```
# include <stdio.h>
  main ( )
  {
    int x[10][10];
    int t, p=0;
    for( t = 0; t<10; t++,p++)
       {
           x[t][p] = t*p;
           printf("%d\n", x[t][p] );
       }
  }
```