CENTRO PAULA SOUZA

Linguagem de Programação

Estrutura de um programa

- Matrizes e Strings
- Material: LP_Aula05

GOVERNO DO ESTADO SÃO PAULO

Matrizes

Matrizes são usadas para tratamento de conjuntos de dados que possuem as características idênticas.

Um conjunto recebe um nome comum e os elementos do conjunto são referenciados através de índices.

Podem ser: **unidimensionais** ou **multidimensionais**

Matrizes

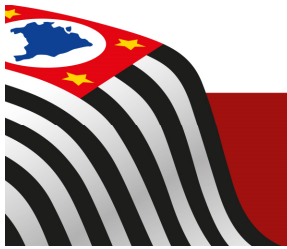
É importante notar que matrizes de qualquer dimensão são caracterizadas por terem todos os elementos pertencentes **ao mesmo tipo de dado.**

Para se declarar uma matriz/vetor podemos utilizar a seguinte forma geral:

```
tipo_da_variável nome_da_variável [tamanho];
```

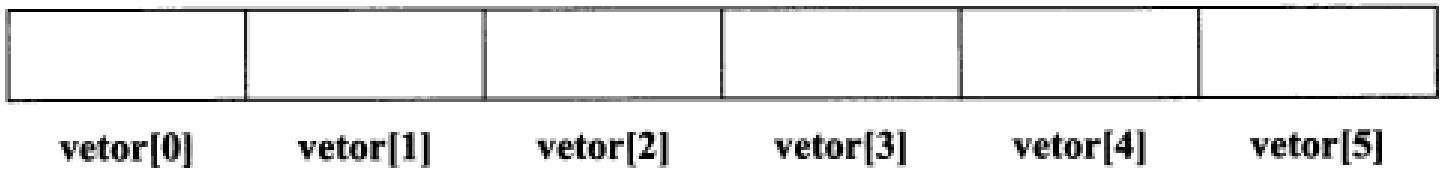
Exemplos de declarações

```
int numeros[1000]; /* conjunto de 1000 numeros inteiros */

float notas[65]; /* conjunto de 65 numeros reais */

char nome[40]; /* conjunto de 40 caracteres */
```



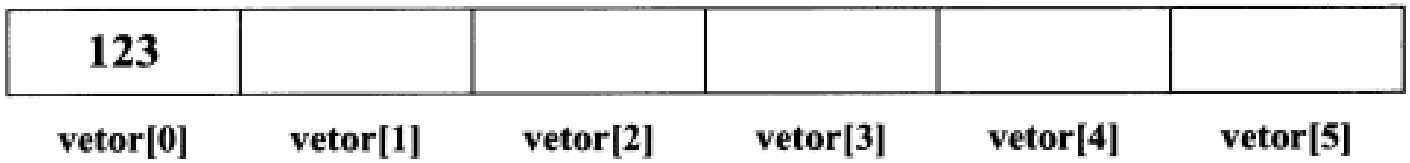
Exemplo – Vetor de inteiros de 6 posições

```
int vetor[6];
```



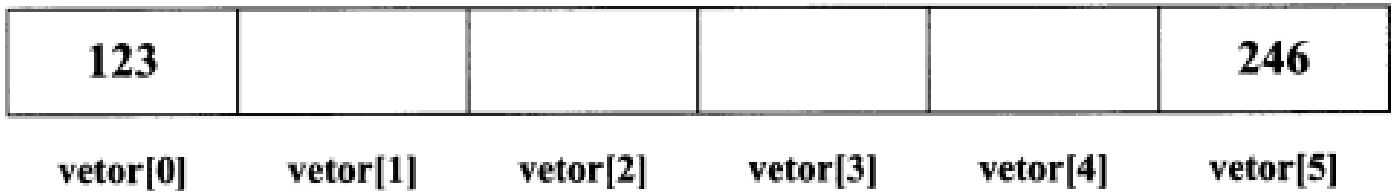
Exemplo: Coloque o valor 123 na primeira posição do vetor:

```
vetor[0] = 123;
```



Coloque na última posição do vetor o dobro do valor do primeiro elemento.

```
vetor[5] = vetor[0]*2;
```



Sobre espaço em Memória

O espaço de memória, em bytes, ocupado por um vetor é igual a:

- espaço = tamanho * (número de bytes ocupado por tipo)

Carga Inicial de Vetor

Da mesma forma que as variáveis, os vetores quando criados contêm valores aleatórios (LIXO) em cada uma das sua posições.

É possível iniciar automaticamente todos os elementos de um vetor através da sintaxe:

```
tipo var[n] = { valor1, valor2, ..., valorn };
```

Exemplo

Exemplo: Declare e inicie um vetor com todas as vogais do alfabeto.

```
char vogal[5] = {'a', 'e', 'i', 'o', 'u'};
```

Evita-se, assim, escrever o seguinte conjunto de código:

```
char vogal[5];
vogal[0] = 'a';
vogal[1] = 'e';
vogal[2] = 'i';
vogal[3] = 'o';
vogal[4] = 'u';
```

Nota

Um vetor declarado com N elementos, e se forem colocados apenas k valores (k<N) na carga inicial do vetor. Os elementos não carregados ficarão com o valor ZERO.

Suponhamos a seguinte declaração:

```
int v[10] = {10,20,30};
```

No exemplo anterior, os três primeiros elementos do vetor (índices 0, 1 e 2) ficam iniciados com os valores 10, 20 e 30, respectivamente, e todos os outros ficam iniciados com o valor 0.

Assim, as seguintes instruções são equivalentes

```
int v[10] = {10,20,30};
int v[10] = {10,20,30,0,0,0,0,0,0,0};
```

Exemplo (vetor 1)

```
#define DIM 5
#include <stdio.h>
int main()
{
  int vetor[DIM];
  int i, num;

  printf("\nEste programa gera um vetor contendo numeros inteiros.\n");
  printf("Entre com o numero inicial do conjunto. ");
  scanf("%d", &num);

  /* Geracao do conjunto */
  for (i=0 ; i<DIM; i++) vetor[i] = num++;

  /* Impressao do conjunto */
  for (i=0; i <DIM; i++)
    printf("Elemento %d = %d\n", i, vetor[i]);
  system("PAUSE");
  return 0;
}
```

Observe os comentários do professor.

Exemplo (Vetor 2)

```
#include <stdio.h>
int main ()
{
         int num[100];  /* Declara um vetor de inteiros de 100 posicoes */
         int count=0;
         int totalnums;
         do
        {
         printf ("\nEntre com um numero (-5 p/ terminar): ");
         scanf ("%d",&num[count]);
         count++;
        } while (num[count-1]!=-5);
         totalnums=count-1;
         printf ("\n\n\n\t Os numeros que voce digitou foram:\n\n");
         for (count=0;count<totalnums;count++)
                  printf (" %d",num[count]);
    printf("\n");
    system("pause");
    return(0);
}
```

Sua tarefa

Reescreva o programa anterior, fazendo com que a cada leitura verifique se a dimensão do vetor não foi ultrapassada.

Caso o usuário entre com 10 números, o programa deverá abortar o loop de leitura automaticamente.

O uso do Flag (-5) não deve ser retirado.

Exercícios Resolvido

Escreva um programa que declare um vetor com n=10 números reais e coloque na i-ésima posição o resultado de i*(n-1)

```
 1: #include <stdio.h>
 2:
 3: const int n=10;
 4:
 5: main()
 6: {
 7:      float v[n];
 8:      int i;
 9:      for (i=0;i<n;i++)
10:          v[i] = i*(n-i);
11:
12:      for (i=0;i<n;i++)
13:          printf("%f\n",v[i]);
14:
15: }
```

Exercícios

1) Desenvolva um programa que faça a leitura de 10 valores no vetor A.

- Construir um vetor B do mesmo tipo, observando a seguinte formatação:
  - Se o valor do índice for par, o valor deverá ser multiplicado por 5;
  - Se o valor do índice for ímpar, deverá ser somado com 5.
- Ao final mostrar os conteúdos dos dois vetores invertidos (listar ao contrário).

Exercícios

2) Desenvolver um programa que leia 5 elementos de um vetor A.

- No final, apresente:
  - A soma de todos os valores ímpares.
  - A soma de todos os valores pares.
  - A soma total.
  - E a porcentagem de números ímpares em relação aos pares.

Exercícios

3) Crie um programa que leia 8 valores em um vetor A.

- Construir um vetor B com a mesma dimensão. Os elementos do vetor A devem ser multiplicados por 3 e armazenado o resultado no vetor B. Ou seja,
- B[1] = A[1] * 3; por exemplo.

Exercícios

4) Crie um vetor A com 5 posições.

- O usuário deverá fornecer o valor para cada campo, ao final será mostrado como resultado este mesmo vetor A, porém em ordem crescente de valores.

Vetor Multidimensional

Matrizes com duas dimensões:

- É a mais comum.
- Estará sempre fazendo menção a linhas e colunas. Sendo representada por seu nome e tamanho.
- Desta forma seria uma matriz de duas dimensões:
  - `int matriz[10][20];`
- O exemplo define uma matriz quadrada de 10 linhas por 20 colunas.

Exemplo (Matriz – 1)

```
#define DIML 3
#define DIMC 5

#include<stdio.h>

int main()
{
     int i, j;
     int matriz[DIML][DIMC];
     printf("\nEntre com 15 valores inteiros: \n");
     for (i=0; i<DIML; i++)
         for (j=0; j<DIMC; j++)
             scanf("%d", &matriz[i][j]);

     for (i=0; i<DIML; i++)
     {
         for (j=0; j<DIMC; j++)
             printf("%4d", matriz[i][j]);
         printf("\n");
     }
     system("PAUSE");
     return(0);
}
```

Matriz de duas dimensões

A matriz é armazenada na memória linha a linha e a Tabela ao lado ilustra esta idéia com uma matriz de números inteiros de três por três.

Estamos assumindo que cada número inteiro ocupa quatro bytes, o endereço aponta um byte e a matriz está armazenada a partir do endereço 1000.

End	Elemento
1000	m[0][0]
1004	m[0][1]
1008	m[0][2]
1012	m[1][0]
1016	m[1][1]
1020	m[1][2]
1024	m[2][0]
1028	m[2][1]
1032	m[2][2]

Exercício Resolvido [Jogo da Velha]

Exercício:

Escreva um programa que coloque o tabuleiro do jogo da velha nesse estado, depois de ter sido iniciado com espaços durante a declaração do mesmo.

X		O
	X	
		O

Exercício Resolvido [Jogo da Velha]

```
 1: #include <stdio.h>
 2: #define DIM 3
 3:
 4: main()
 5: {
 6: char Velha[DIM][DIM]= {{' ',' ',' '},{' ',' ',' '},{' ',' ',' '}};
 7: int i,j;
 8:
 9: Velha [0][0] = 'X';
10: Velha [1][1] = 'X';
11: Velha [0][2] = '0';
12: Velha [2][2] = '0';
13:
14: for (i=0;i<DIM;i++)
15:     {
16:         for (j=0;j<DIM;j++)
17:             printf("%c %c", Velha [i][j],j==DIM-1?' ':'|');
18:         if (i!=DIM-1) printf("\n———\n");
19:     }
20: }
```

Exercícios

1) Desenvolver um programa que efetue a leitura de 8 RAs de alunos e também suas duas notas bimestrais.

- Ao final o programa deverá exibir o RA do aluno bem como sua média final.

Exercícios

2) Faça um programa que leia duas matrizes A e B, cada uma com uma dimensão de 4 linhas por duas colunas.

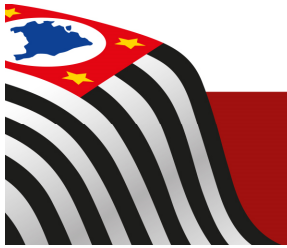
- Construa uma matriz C com a mesma dimensão que seus elementos deverão conter as somas dos valores de mesma posição na matriz A e B.
- Ex:
- MatrizC[0][1] = MatrizA[0][1] + MatrizB[0][1]

Strings

A linguagem C possui algumas limitações no que diz respeito ao tratamento de vetores e string.

Não é possível atribuir uma *string* a uma variável ou concatenar uma string a outra, pelo menos, não diretamente.

Entretanto, C possui uma biblioteca vasta de funções para manipular praticamente todas as operações necessárias sobre strings. É o que veremos neste material.



Distinção entre caracteres e strings

Uma string é um conjunto de caracteres armazenados em um vetor.

Exemplos de *Strings:*

```
"Luís"
"Zé Manuel Saraiva de Carvalho"
"Bolo de Chocolate com 1,2 kg de peso"
"A"
```

Exemplos de Caracteres:

```
'L'
'>'
'A'
```

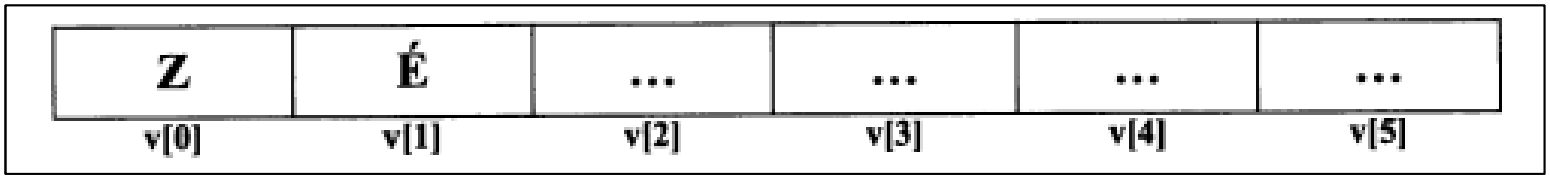
Como em C as *strings* não são um tipo básico, a única forma de representar um conjunto de caracteres é recorrendo a um vetor (de caracteres).

Sintaxe de Vetores

A declaração de strings obedece a sintaxe à sintaxe de declaração de vetores de caracteres em C.

Problema:

- Se declararmos um vetor (v) com 100 posições para o nome de uma pessoa e colocarmos lá ZÉ, como podemos saber quais e quantos dos caracteres estamos efetivamente usando?



Delimitador “\0”

Esse é o caractere universalmente usado para terminar as strings em C.

Nota: se quisermos um vetor que contenha 20 caracteres para o nome de uma pessoa, temos que reservar 1 para o delimitador.

Exemplo:

Declaração de um vetor que conterá uma *string* com 20 caracteres úteis.

```
char s[21];
```

ou, de forma mais legível,

```
char s[20+1]; /* 20 caracteres para usar + 1 para o delimitador */
```

Carga Inicial de Automática de Strings

A declaração e a carga inicial segue a sintaxe normal de vetores.

Exemplos:

```
char nome[20] = "André";
char nome[20] = {'A','n','d','r','é'};
char nome[] = "André";   /* Equivalente a char nome[5+1] = "André"; */
char *nome = "André";    /* idem */
```

Leitura e escrita de strings

Leitura: printf e puts (put string).

Escrita: scanf e gets (get string).

Função: printf

```
char NomeProprio[100]  = "Carla Marina";
char Sobrenome[50] = "Silva";

printf("Nome: %s, %s \n", Sobrenome, NomeProprio);
```

Nome: Silva, Carla Marina

Função puts (put string)

Permite unicamente a escrita de strings. Ao final da escrita sempre faz uma mudança de linha.

```
puts("Hello World")  é equivalente a   printf("Hello World\n")
```

Função scanf

Permite realizar leitura de strings usando o formato %s.

A variável que recebe a string não é precedida pelo & (diferentemente dos outros tipo).

```
 1: #include <stdio.h>
 2:
 3:
 4: main()
 5: {
 6: char Nome[50],Sobrenome[50];
 7: printf("Introduza o Nome: "); scanf("%s",Nome);
 8: printf("Introduza o sobrenome: ");scanf("%s",Sobrenome);
 9: printf("Nome Completo: %s %s\n",Nome, Sobrenome);
10: }
```

```
Introduza o Nome: Carla
Introduza o Sobrenome: Marina
Nome Completo: Carla Marina
```

Função gets (get string)

A função gets permite colocar, na variável que recebe por parâmetro, todos os caracteres introduzidos pelo usuário. Ao contrário do scanf, não está limitada à leitura de uma única palavra.

```
 1: #include <stdio.h>
 2:
 3:
 4: main()
 5: {
 6: char Nome[50];
 7: printf("Introduza o Nome Completo: ");
 8: gets(Nome);
 9: printf("Nome Completo: %s\n",Nome);
10: }
```

```
Introduza o Nome Completo: João Manuel Nunes
Nome Completo: João Manuel Nunes
```

Principais funções de manipulação de strings.

Veremos:

1. **Função strlen.**
2. **Função char *strcpy(char *dest, char *orig).**
3. **Função char *strcat(char *dest, char *orig).**
4. **Função int strcountc(char *s, char ch).**
5. **Função int strcountd(char *s).**
6. **Função int indchr(char *s, char ch).**
7. **Função int strpal(char *s).**
8. **Função char *strrev(char *s).**
9. **Função int strcmp(char *s1, char *s2).**
10. **Função char *strpad(char *s).**
11. **Função char *strdelc(char *s, char ch).**

Função strlen.

Retorna o número de caracteres existentes em uma string sem considerar o delimitador '\0'.

```
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char *argv[])
{
  char nome[50];
  system("CLS");
  printf("Digite seu nome completo: ");
  gets(nome);
  printf("Nome completo: %s\n", nome);
  int carac;
  carac = strlen(nome);
  printf("Quantidade de digitos: %i\n", carac);
  system("PAUSE");
  return 0;
}
```

```
n("")          → 0
n("strlen")    → 6
```

Função char *strcpy(char *dest, char *orig).

Copia a string orig para a string dest.

Repare que a função é do tipo char *, quer dizer que recebe como resultado algo do tipo char *.

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

int main(int argc, char *argv[])
{
  char nomeorig[50];
  system("CLS");
  printf("Digite seu nome completo: ");
  gets(nomeorig);
  printf("Nome origem: %s\n", nomeorig);
  char nomedest[50];
  strcpy(nomedest, nomeorig);
  printf("Nome copiado: %s\n", nomedest);
  system("PAUSE");
  return 0;
}
```

Função char *strcat(char *dest, char *orig)

Coloca a string orig imediatamente após a string dest. (Concatenação)

Exemplo

Programa que lê nome e sobrenomes de pessoas e os mostra na tela no formato Sobrenome, Nome.

```
#include <stdio.h>
#include <stdlib.h>
#define DIM 21
#define SEP_NOME ", "

int main(int argc, char *argv[])
{
  system("CLS");
  char nome[DIM], sobrenome[DIM], completo[DIM];
  printf("\nDigite seu nome: ");
  gets(nome);
  if (strlen(nome)==0) exit(0); /* Nome vazio - encerra programa */
  printf("\nDigite o sobrenome: ");
  gets(sobrenome);
  strcpy(completo, sobrenome); /* Copia sobrenome */
  strcat(completo, SEP_NOME);  /* Copia separador ',' após sobrenome*/
  strcat(completo, nome);      /* Copia nome após separador */
  puts(completo);
  system("PAUSE");
  return 0;
}
```

Função int strcountc(char *s, char ch). E Função int strcountd(char *s).

A primeira retorna o número de ocorrências do caracetere ch no string.

Já a segunda devolve o número de dígitos numéricos contidos na string.

```
strcountc("abacate",'a')          → 3
strcountc("abacate",'y')          → 0

strcountd("15 abacates")          → 2
strcountd("quinze abacates")      → 0
```

Função int indchr(char *s, char ch).

Devolve o índice da primeira ocorrência do caractere ch na string s. Se não existir retorna -1.

```
indchr("15 abacates",'a')        → 3
indchr("15 abacates",'#')        → -1
```

Função int strpal(char *s) [Palíndromo]

Verifica se a string s é um palíndromo, isto é, se é lida da mesma forma da esquerda para a direita e da direita para a esquerda.

```
strpal("matam")     → <TRUE>
strpal("assa")      → <TRUE>
strpal("ba")        → <FALSE>
strpal("assar")     → <FALSE>
```

Função char *strrev(char *s).

Inverte a string s e devolve-a invertida.

```
#include <stdio.h>
#include <stdlib.h>
#define DIM 21

int main(int argc, char *argv[])
{
  char palavra[DIM];
  system("CLS");
  printf("\nDigite uma palavra: ");
  gets(palavra);
  strrev(palavra);  /* Inverte a palavra */
  puts(palavra);
  system("PAUSE");
  return 0;
}
```

Função int strcmp(char *s1, char *s2).

Compara alfabeticamente a string s1 e a s2. Devolve um inteiro:

< 0	Se s1 é alfabeticamente menor que s2
0	Se s1 e s2 são iguais
> 0	Se s1 é alfabeticamente maior que s2

Invocação	Res	Observações
strcmp("abc","abxpo")	<0	Pois 'c'<'x'
strcmp("beatriz","carlos")	<0	Pois 'b'<'c'
strcmp("carlos","carla")	>0	Pois 'o'>'a'
strcmp("carlos","beatriz")	>0	Pois 'c'>'b'
strcmp("mario","maria")	>0	Pois 'o'>'a'

Função char *strpad(char *s).

Coloca um espaço em branco após cada um dos caracteres da string s.

```
char s[20] = "ola";
strpad(s);
printf(s);          →  o□l□a□
```

Função char *strdelc(char *s, char ch).

Apaga todas as ocorrências do caractere ch contidos na string s.

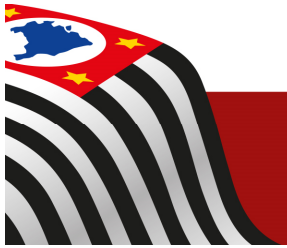
```
strdelc("apaga caracteres",'a')    → "pg crcteres"
strdelc("apaga caracteres",'e')    → "apaga caractrs"
```

CENTRO PAULA SOUZA

Fatec
Mogi Mirim
Arthur de Azevedo

Prof. Me. Marcos Roberto de Moraes, o Maromo

FIM



GOVERNO DO ESTADO
SÃO PAULO

CENTRO PAULA SOUZA

Fatec
Mogi Mirim
Arthur de Azevedo

Referências Bibliográficas

DAMAS, L. M. D. **Linguagem C**. LTC, 2007.

HERBERT, S. **C completo e total**. 3a. ed. Pearson, 1997.