VETORES, MATRIZES, STRINGS

- **Variáveis**
- Variáveis compostas homogêneas
  - Vetores
  - Matrizes
  - Strings
- Variáveis compostas heterogêneas
  - Estruturas
    - Registros
    - Arquivos
- Variáveis Compostas
  - Um conjunto de variáveis identificadas por um mesmo nome.
  - Homogêneas (vetores e matrizes)
  - Heterogêneas (estruturas)

- **Variáveis Compostas Homogêneas**
- Correspondem a posições da memória:
  - identificadas por um único nome
  - individualizadas por índices
  - cujo conteúdo é de um mesmo tipo

ARRANJOS UNIDIMENSIONAIS - VETORES

- Armazenam conjuntos de dados cujos elementos podem ser endereçados por um único índice.

Notas:

9.7	3.4	1.8	5,1	7.5	2.9	10	7,3	6,5	4,4
Posição: 0	1	2	3	4	5	6	7	8	9

- Esse é um vetor de 10 elementos, isto é, tem 10 variáveis, todas com o mesmo nome e diferentes por sua posição dentro do arranjo que é indicada por um índice.
- Quando se tem somente uma linha, podemos omiti-la e colocar somente a coluna.
  A_0= 6.1 A_1= 2.3 A_2= 9.4 A_3= 5.1 A_4= 8.9 A_5= 9.8 A_6= 10 A_7= 7.0 A_8= 6.3 A_9= 4.4

- Em algoritmos, representamos da seguinte forma:
  A[0]= 6.1 A[1]= 2.3 A[2]= 9.4 A[3]= 5.1 A[4]= 8.9 A[5]= 9.8 A[6]= 10 A[7]= 7.0 A[8]= 6.3 A[9]= 4.4

Arranjos Multidimensionais - MATRIZES

- Armazenam conjuntos de dados cujos elementos necessitam ser endereçados por mais de um índice.
- Também são conhecidos como arrays

Coluna

Linha	0	1	2	…	n-1
0	23	56	78	…	29
1	35	98	23	…	76
2	65	56	43	…	11
⋮	⋮	⋮	⋮	⋱	⋮
m-1	41	13	54	82	20

ALGORITMOS COM VETORES

- Um algoritmo com vetor implica várias seções para que possa funcionar corretamente. Essas seções são independentes.

Trecho de Dimensionamento – Declaração de Vetores

- Para dimensionar um vetor, usamos o seguinte comando na declaração de variáveis:

```
<tipo> <nome> [<tamanho>];
```

- onde tamanho, na prática, é o número de elementos:
  - [5] 5 elementos (de 0 a 4)
  - tipo: poderá ser: int, float, char, double (qualquer tipo válido em C)
  - nome: será o que dado ao vetor (qualquer nome válido em C)
  - No exemplo acima seria: float Notas[10];
  - Exemplos:
    - float VReais[100];
    - int VInteiros[5];
    - char Nome[50];
    - float forma[7];

Armazenamento de Vetores

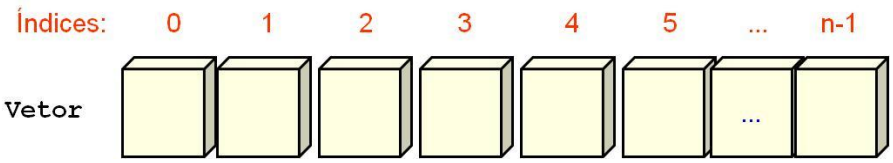
- O compilador C aloca uma porção contígua na memória do computador para armazenar os elementos das matrizes e vetores, sendo que o endereço mais baixo corresponde ao primeiro elemento e o endereço mais alto ao último elemento.
- Exemplo: float exemplo [20]; >>>o C irá reservar 4x20=80 bytes.

Elementos dos Vetores

- Elementos: conteúdo de cada uma das posições do vetor:
- Exemplos:
  - Posição 0 do vetor Notas >> elemento 9.7
  - Posição 5 do vetor Notas >> elemento 2.9

Índices de Vetores

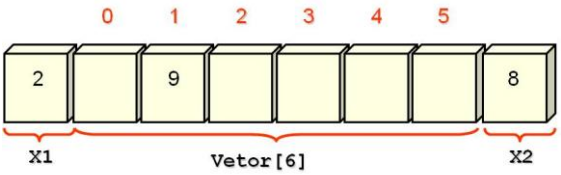
- Correspondem às posições de cada elemento. Exemplo **int Vetor[n];**



  - Índice do primeiro elemento: zero
  - Índice do último elemento: n – 1
  - Quantidade de elementos: n

Limites de Vetores

- Índices fora dos limites podem causar comportamento anormal do código.



- Verificação de Limites - deve ser feita pelo programador
- A linguagem C não verifica os limites de um vetor, ou seja, não verifica se o índice que você usou está dentro dos limites válidos. Portanto, é possível ultrapassar o fim de um vetor e escrever em outra variáveis

```
int X1;
int Vetor[6];
int X2
```

```
Vetor[1]  = 9;
Vetor[-1] = 2;
Vetor[6]  = 8;
```

Inicialização de Vetores

- Atribuir valores na declaração do vetor:
  - vetores com tamanho pré-definido

```
int vetor[5] = {1,2,3,4,5};
```

  - O tamanho de um vetor é pré-definido, após a compilação, não pode ser mudado. Mantêm o mesmo tamanho durante a execução do programa.

Estruturas de Dados Estáticas

  - vetores sem tamanho pré-definido

```
int vetor[ ] = {1,2,3,4,5};
```

(é obrigatório inicializar na declaração)

- Atribuir valores no corpo da função

```
int vetor[3];
V[0] = 1;
V[1] = 2;
v[2] = 3;
```

- **Exercício 1: Dado o vetor V[10] e sendo X=2 e Y= 4 determine:**

4	5	3	2	5	7	3	1	8	0
0	1	2	3	4	5	6	7	8	9

1. V[0] = 4
2. V[9] = 0
3. V[X] = 5
4. V[X+Y] = 3
5. V[V[7]] = 5
6. V[8-V[3]] = 3
7. V[2+3] = 7
8. V[X*Y] = 8
9. V[X] + V[Y]= 8
10. V[V[X+4]] = 4
11. V[V[V[7]] = 7
12. V[V[8]-V[3]]= 3

Trecho de Entrada de Dados

- Normalmente, uma estrutura de repetição.
- Se for a estrutura do **for**, deverá ter o valor final igual à última posição do vetor.
- Se for a estrutura do **while** ou **do ...while**, deverá ter uma variável (contador) que deverá ser inicializada fora do laço e incrementada dentro do laço. Esse contador e nunca poderá assumir um valor maior do que a última posição do vetor.
- Exemplo

```
for ( i = 0; i < tamanho; i ++)
{
        printf (" Entre com o Elemento %d = %d\n", i+1);
        scanf ("%d", &nome[i]);
}
```

- O trecho anterior poderá ser incrementado com outros comandos;

Trecho de Impressão de Dados

- Normalmente, uma estrutura de repetição.
- Se for a estrutura do **for**, deverá ter o valor final igual à última posição do vetor.
- Se for a estrutura do **while** ou **do ...while**, deverá ter uma variável (contador) que deverá ser inicializada fora do laço e incrementada dentro do laço. Esse contador e nunca poderá assumir um valor maior do que a última posição do vetor.
- Exemplo

```
for ( i = 0; i < tamanho; i ++)
{
        printf (" nome [%d] = %d\n", i, nome[i]);
}
```

Dica: Definir o tamanho de vetores com constantes flexibiliza a manutenção do código. Podemos, portanto, usar a diretiva

#define <constante> <valor>

para definir uma constante com a dimensão desejada para os vetores. Na declaração de variáveis usamos a sintaxe

<tipo> <nome> [constante];

para declarar o vetor com a dimensão pré-definida. Assim, se precisamos alterar a dimensão do vetor basta mudar o valor da constante na diretiva.

Vetores: Exemplo

- Leitura dos dados de um vetor:

```
#define DIM 10

for (i=0; i < DIM; i++)
{
    printf("Digite um número: ");
       scanf("%f", &Vet[i]);
}
```

- Colocar os números de 1 a 5 num vetor:

```
for (i=0; i<5; i++)
       Vetor[i] = i + 1;
```

- Colocar os números de 5 a 1 num vetor:

```
for (i=0; i<5; i++)
       Vetor[i] = 5 - i;
```

- Copiar dados de um vetor para outro:

```
#define DIM 10
double VReais[DIM], VCopia[DIM];
for (i=0; i< DIM; i++)
       VCopia[i] = VReais[i];
```

Exercício 2: Ler N valores inteiros (N ≤ 100) até que seja digitado o valor zero. A seguir, inverter o vetor, trocando o 1º elemento com o último, o 2º com o penúltimo, e assim sucessivamente. Ao final, imprimir o vetor invertido.

```
#include <stdio.h>
#include <stdlib.h>
#define DIM 101
int main() {
      /* Declaracao de variaveis */
      int vet[DIM];   /* Vetor a ser inserido */
      int i;          /* Contador */
      int tam;        /* Tamanho do vetor */
      int aux;        /* Variavel auxiliar */
      /* Leitura dos elementos do vetor "vet" */
      printf("Digite os elementos do vetor (ZERO para encerrar):\n");
      i = -1;          /* Inicializa contador */
      do {
            i++;       /* Incrementa contador */
            scanf("%d", &vet[i]); /* Le elemento a partir do teclado */
      } while (vet[i] != 0 && i <= tam);
      tam = i - 1;       /* Definicao do tamanho do vetor */
      /* Impressao do vetor inserido: */
      printf("\nVetor inserido:\n");
      for (i=0; i <= tam; i++)
            printf("Elemento %d : %d \n", i, vet[i]);
      /* Inversao dos elementos de "vet" */
            for (i=0; i <= tam/2; i++) {
                  aux        = vet[i];
                  vet[i]     = vet[tam-i];
                  vet[tam-i] = aux;
            }
      /* Impressao do vetor invertido: */
      printf("\nVvetor invertido:\n");
      for (i=0; i <= tam; i++)
            printf("Elemento %d : %d \n", i, vet[i]);
            return 0;
      }
```

Passagem de Vetores

- O compilador interpreta o nome de um vetor como o endereço do primeiro elemento do vetor. Portanto, os vetores são sempre passados por referência, sem usar uma notação especial.

```
<tipo> funcao(int vet[], ...)
{
   ...
}
```

- Ao passar um vetor como parâmetro, se ele for alterado dentro da função, as alterações ocorrerão no próprio vetor e não em uma cópia.
- Ao retornar um vetor como valor de retorno, não é feita uma cópia deste vetor.
- Ao passar um vetor como parâmetro, não é necessário fornecer o seu tamanho na declaração da função. Porém, é importante lembrar que o vetor tem um tamanho que deve ser considerado pela função durante a manipulação dos dados.

Exemplos

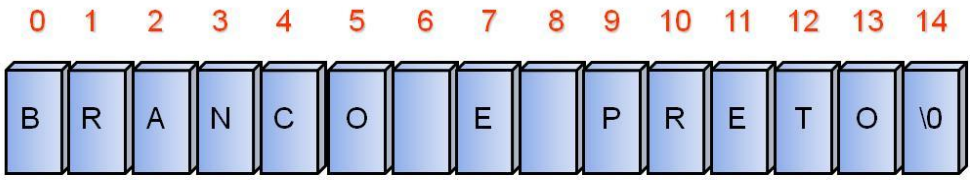
```
//E03 Este programa le dois vetores e os imprime na tela,utilizando procedimentos com
      passagem de vetores como parametros

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

#define ELEM2 4

/* Função para leitura de vetores */
void ler (int vet[], int elem) {
      int i;         /* Contador */
      for (i = 0; i < elem; i++) {
            printf("vetor[%d]: ", i);
            scanf("%d", &vetor[i]);
      }
}

/* Função para escrita de vetores na tela */
void escrever (int vetor[], int elem) {
      int i;         /* Contador */
      for (i = 0; i < elem; i++)
            printf("vetor[%d]: %d\n", i, vetor[i]);
}

/* Funcao principal */
int main()  {
      /* Declaracao de variaveis  */
    int ELEM1;
      printf("Digite o tamanho do 1o. vetor \n");
      scanf("%d", ELEM1);
      int vet1[ELEM1];
      int vet2[ELEM2];
      /* Aquisição de dados */
      printf("Digite os valores dos elementos do 1o. vetor (%d elementos)\n", ELEM1);

      ler(vet1, N_ELEM_1);
      printf("\nDigite os valores dos elementos do 2o. vetor (%d elementos)\n",
      ELEM2);

      ler(vet2, N_ELEM_2);
      printf("\n\n");
      /* Impressao de dados */
      printf("Primeiro vetor:\n");
      escrever(vet1, ELEM1);
      printf("\n");
      printf("Segundo vetor inserido:\n");
      escrever(vet2, ELEM2);
      return 0;
}
```

Strings

- Uma string é uma série de caracteres terminada com um caracter nulo, representado por '\0‘ (valor inteiro zero)
- Ao definir uma string, deve-se levar em consideração, além do número de caracteres da string, o caractere nulo que termina a string.
- É representada por um vetor de caracteres.
  - Posibilita que os caracteres que formam a string sejam acesados individualmente
  - grande flexibilidade

Armazenamento de uma String

- Como constante, a string é uma série de caracteres delimitada por aspas: “branco e preto” armazenada conforme a configuração abaixo.

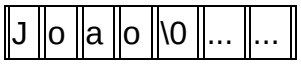



```
Vetor[0]  = 'B';        Vetor[6]  = ' ';
Vetor[7]  = 'E';        Vetor[14] = '\0';
```

Declaração de Strings

```
char  nome_da_string [tamanho];
```

- Declara um vetor de caracteres (uma string) com número de posições igual a *tamanho*.
- Temos que reservar um caractere para ser o terminador nulo,
  - comprimento da string > que a maior string que pretendemos armazenar (+1 caracter)
- Exemplo: `char string[7] = “João”;`



  - onde: as duas células não usadas têm valores indeterminados. (o C não inicializa variáveis)
  - Células que são inicializadas: 'J', 'o', 'a', 'o' e '\0'

Manipulação de Strings

- String não é um tipo básico no C => operações simples como atribuição e comparação não podem ser feitas diretamente com os operadores conhecidos (=, >, <, etc)
- 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;  /* NAO faca isto */`
- Qualquer operação com strings exige o processamento individual dos elementos do vetor, ou seja o processamento individual de cada caracter.

Leitura a partir do teclado

- Para ler uma string fornecida pelo usuário podemos usar a função gets() e a função scanf(). A diferençadestas leitura do uso destas funções é apresentado abaixo.

Função gets()

- Lê string até o primeiro enter

```
gets (nome_da_string);
```

- Exemplo: "BRANCO E PRETO"

```
char str[ 20 ];
gets (str);          // str= "BRANCO E PRETO"
```

- A chamada gets(str) lê uma string e a armazena no vetor str.
- O <enter> para finalizar a entrada, é automaticamente substituído por '\0'.

Função scanf()

- Lê string até o primeiro espaço em branco

```
scanf ("%s", nome_da_string);
```

- Exemplo: "BRANCO E PRETO"

```
char str[ 20 ];
scanf ("%d",str);          // str= "BRANCO"
```

Acesso a uma String

- Como as strings são vetores de caracteres, para se acessar um determinado caracter de uma string, basta "indexar", ou seja, usar um índice para acessar o caracter desejado dentro da string.
- Exemplo `str[1] = 'R';` acessa a segunda letra de str :

```
#include <stdio.h>
int main()
{
    char str[20] = "PRETO E BRANCO";
    printf("\n\nString: %s", str);
    printf("\nTerceira letra: %c", str[1]);
    str[2] = 'A';
    printf("\nAgora a Terceira letra é: %c", str[2]);
    printf("\n\nString resultante: %s", str);
        return(0);
}
```

- terminador nulo está na posição 15.
- posições 0 a 15,=> caracteres são válidos
- %s indica que printf() deve colocar uma string na tela

Inicialização de Strings

- Atribuir valores na declaração do vetor:
  - Sintaxe padrão: os caracteres são fornecidos entre chaves e separados por virgulas

```
char  nome [7] = {'B','R','A','N','C','O','\0'};
```

  - Sintaxe própria: Os carcteres são fornecidos entre aspas (string constante).

```
char  nome [15] = "BRANCO E PRETO";
```

    - Mais compacta
    - o caracter nulo é incluido automaticamente

Atribuição de valores

- De acordo com as formas de atribuição abaixo verifica-se que não é possível atribuir um valor a uma string usando o operador "="

```
char  curso[15] = "Sistemas";  /* Válido somente na declaracao!
```

Para atribuir valor a uma string no corpo da função utiliza-se a função strcpy()

```
char  curso[15];
curso = "Processamento";
```

NÃO

- Exemplos:

Comandos Válidos

```
char A[ ] = "um", B[3] = "um";
char C[3] = {'u','m','\0'};
char D[ ] = {'u','m','\0'};
char F[3];
gets(F);
strcpy(F,"um");
F[0] = 'u';
F[1] = 'm';
F[0] = '\0';}
```

Comandos Inválidos

- **char E[ ];**
  - **precisa da dimensão**
- **F="um";**
- **F[3] = "um";**

Função strcpy()

- A função strcpy (curso,"processamento"); copia a string-origem (processamento) para a string destino (curso).

```
strcpy(string_destino,string_origem); /* Requer biblioteca string.h
```

- Exemplo

```
#include <stdio.h>
#include <string.h>
int main () {
  char str1[100],str2[100],str3[100];
  printf ("Entre com uma string: ");
  gets (str1);
  printf(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);
  return(0);
  }
```

- Nos programas que tratam de string muitas vezes pode-se aproveitar o fato de que uma string termina com '\0' (isto é, o número inteiro 0).
- Exemplo, o programa abaixo que serve para igualar duas strings (isto é, copia os caracteres de uma string para o vetor da outra) :

```
#include <stdio.h>
int main ()
{
     int count;
```

```
        char str1[100],str2[100];
        // O programa le str1 que sera copiada para str2
        for (count=0;str1[count];count++)
                str2[count]=str1[count];
           str2[count]='\0';
}
```

- Como a string que está sendo copiada termina em '\0', quando o elemento encontrado em str1[count] é o '\0', o valor retornado para o teste condicional é falso (nulo), terminando a execução do laço e, portanto, da cópia da string

Função strcat()

- A função strcat() concatena a string-origem com a string destino.

```
strcat (string_destino, string_origem);   // requer a biblioteca string.h
```

- A string de origem permanece inalterada e será anexada ao fim da string de destino.
- Exemplo:

```
#include <stdio.h>
#include <string.h>
int main () {
{
  char str1[100],str2[100];
  printf ("Entre com uma string: ");
  gets (str1);
  strcpy (str2,"Voce digitou a string ");
  strcat (str2,str1);//str2 armazena: Voce digitou a string + o conteudo de str1
  printf ("\n\n%s",str2);
  return(0);
}
```

Função strlen()

- A função strcat()retorna o comprimento da string fornecida

```
strlen(string); /* Requer biblioteca string.h
```

- '\0' não é contado => o comprimento do vetor da string deve ser um a mais que o inteiro retornado por strlen().
- Exemplo

```
#include <stdio.h>
#include <string.h>
int main () {
  int tamanho;
  char str[100];
  printf ("Entre com uma string: ");
  gets (str);
  tamanho = strlen (str);
  printf ("\n\nA string que voce digitou tem tamanho %d",tamanho);
  return(0);
}
```

Função strcmp()

- A função strcmp ()compara a string 1 com a string 2.

```
strcmp(string1, string2); /* Requer biblioteca string.h
```

- Se as duas forem idênticas a função retorna zero. Se elas forem diferentes a função retorna não-zero.

ATENÇÃO

```
#include <stdio.h>
#include <string.h>
main(void) {
char A[] = "um";
char B[] = "um";
if( A==B) printf("%s == %s é verdadeiro", A, B);
else printf("%s == %s é falso", A, B);
}
```

Compara o endereço do vetor A com o endereço do vetor B

```
" um == um é falso "
```

```
#include <stdio.h>
#include <string.h>
main(void) {
char A[] = "um";
char B[] = "um";
if(strcmp( A,B)==0) printf("%s == %s é verdadeiro", A, B);
else printf("%s == %s é falso", A, B);
}
```

Compara o conteúdo do vetor A com o conteúdo do vetor B

```
" um == um é verdadeiro "
```

- Para verificar se uma string é igual a outra, precisa comparar os caracteres correspondentes um a um
- Exemplo: Módulo de Comparação:

```
int strcmp(char s[], char t[]) {
int i=0;
while( s[i]==t[i] && s[i]!='\0' ) i++;
return s[i]-t[i];
}
```

- O laço while só pára quando s[i] for diferente de t[i] ou se ambos forem nulos
- Exemplo

```
#include <stdio.h>
#include <string.h>
int main () {
  char str1[100],str2[100];
  printf ("Entre com uma string: ");
  gets (str1);
  printf ("\n\nEntre com outra string: ");
  gets (str2);
  if (strcmp(str1,str2))
          printf ("\n\nAs duas strings são diferentes.");
  else printf ("\n\nAs duas strings são iguais.");
  return(0);
}
```

Exercícios

- E01 Leia 5 strings e as exiba na tela:

```
#include <stdio.h>
#include <stdlib.h>
main ()
{
char strings [5][100];
int count;
for (count=0;count<5;count++)  {
  printf ("\n\nDigite uma string: ");
  gets (strings[count]);
  }
printf ("\n\n\nAs strings que voce digitou foram:\n\n");
for (count=0;count<5;count++)
  printf ("%s\n",strings[count]);
system("pause");
}
```

■ E02 Codificação da Função srtcat(), strcmp(), strlen()

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

int main() {

/* Declaracao de variaveis */
  char str1[10], str2[10];
     int i;
  printf("Copiando uma string em outra\n");
  printf("\n");

/* Recebe uma string como entrada */
  printf("Digite string1:\n");
  scanf("%s", &str1);
  printf("\n");

/* Copia a string contida em "str1" para "str2" */
  strcpy(str2,str1);

/* Impressao dos resultados */
  printf("string1 (original): %s\n", str1);
  printf("string2 (copia)   : %s\n", str2);
/* Recebe uma string como entrada */
  printf("Digite string1:\n");
  scanf("%s", &str1);
  printf("\n");

/* Verifica o tamanho da string */
    for(i=0;str1[i]!='\0';i++) {
      printf("tamanho: %d\n", i);
      printf("starlen: %d\n",strlen(str1));}

/* Compara duas strings */
  printf("Digite string1:\n");
  scanf("%s", &str1);
  printf("Digite string2:\n");
  scanf("%s", &str2);
  printf("\n");
  i=0;

  while( str1[i]==str2[i] && str1[i]!='\0' ) i++;
    printf("%d\n", str1[i]-str2[i]);
  getch();
}
```

■ E03 Codificação dos módulos da Função srtcat(),strcmp(), strlen()

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

#define DIM 10

int stcat(char *, char *);
int stcmp(char *, char *);
int stlen(char *);
int main()
{
  /* Declaracao de variaveis */
  char vet1[15]="branco e preto" , vet2[16]= "amarelo e verde";
  char vet3[15]="branco e preto" , vet4[16]= "aranco e preto";

    fflush(stdin);

    printf ("%s\t%s\n", vet1,vet2);
    stcat(vet1,vet2);
    printf ("%s\t%s\n\n\n", vet1,vet2);
    printf ("%d\n\n\n", stlen(vet1));
    printf ("%d\n\n\n", stcmp(vet3,vet4));
  system("pause");
  return 0;
}
```

```
int  stcat(char *v1, char *v2)
     {
       int i=0;
          while (*v1!='\0') v1++;
          while (*v2!='\0') {
                *v1=*v2;
                v2++;
                v1++;
          }
         *v1='\0';
      return 0;
     }

int  stcmp(char *v1, char *v2)
     {
          while (*v1==*v2 && *v2!='\0')
                {
                  v1++;
                  v2++;
                  }
      return (*v1-*v2);
     }

int  stlen(char *v1)       {
          int i=0;
          while (*v1!='\0')
               {i++;
                v1++;}
      return (i);
     }
```

- E04 - Faça um programa que leia quatro palavras pelo teclado, e armazene cada palavra em uma string. Depois, concatene todas as strings lidas numa única string. Por fim apresente esta como resultado ao final do programa.

Matrizes

- São arranjos multi-dimensionais que armazenam conjuntos de dados cujos elementos necessitam ser endereçados por mais de um índice.
- Também são conhecidas como tabelas ou arrays
- É um conjunto homogêneo, cujos elementos são distribuídos em linhas e colunas.
- Exemplo: Arranjo de **2** dimensões ***4*** x ***4***

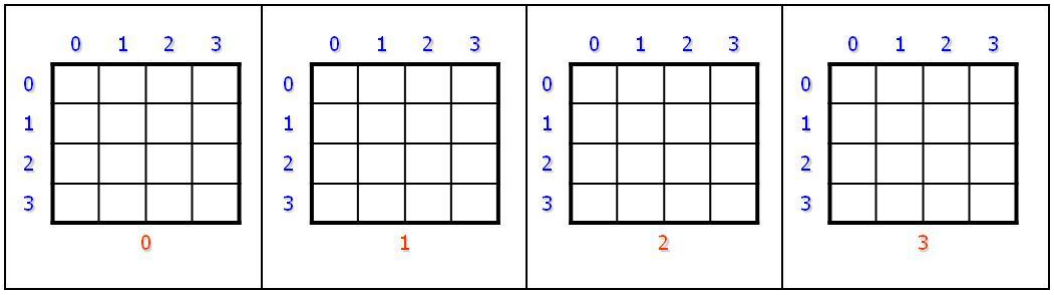
	0	1	2	3
0	17	21	34	16
1	15	56	19	19
2	99	56	34	52
3	85	15	54	20

- Representação: estrutura bidimensional com o número de linhas e colunas correspondentesàs dimensões
- Seja ***A*** uma matriz ***m*** x ***n***
  - Linhas são indexadas de 0 a m-1
  - Colunas são indexadas de 0 a n-1
  - Para acessar um elemento de ***A*** >> ***A***[***i***] [***j***]
    - i >> número da linha;
    - j >> numero da coluna.
    - Exemplo:

      A[2] [3] = 52
      A[0] [0] = 17
      A[3] [3] = 20

- Tecnicamente matriz é um vetor de vetores, ou seja, um conjunto de vetores

- Arranjo de **3** dimensões ***4*** x ***4*** x ***4***



- Repete-se a estrutura bidimensional o mesmo número de vezes que o elemento da terceira dimensão
- O tamanho de uma matriz é pré-definido => após a compilação, não pode ser mudado, mantendo o mesmo tamanho durante a execução do programa.

Estruturas de Dados Estáticas

- A quantidade de elementos de uma matriz é calculada através da multiplicação dos índices:
- Exemplos:
  - Matriz M 2 x 3 tem 6 elementos
  - Matriz A 2 x 3 x 4 tem 24 elementos

Exercício:

i \ j	0	1	2	3	4	5
0	5	4	3	2	1	0
1	3	1	2	0	4	5
2	3	4	5	0	2	1

Seja X = 1 e Y = 4

1. M[0] [0] = **5**
2. M[2] [4] = **2**
3. M[X] [Y] = **4**
4. M[2] [X+Y] = **1**
5. M[M[1] [3]] [M[X] [Y]] = **1**
6. M[6-M[0] [0]] [X+Y]= **5**

Declaração de Matrizes:

- Para dimensionar uma mnatriz, usamos o seguinte comando na declaração de variáveis:

```
<tipo> <nome> [<tamanho1>][<tamanho2>]...;
```

- Exemplo1: Declaração de uma matriz 3 x 4 de números inteiros:
  - **int A [3] [4];**
  - cria um vetor A, cujos elementos A[0], A[1], A[2] são vetores, contendo 4 elementos cada
- Exemplo2:

```
float MatReais[100] [100];
int   Matriz[5][9];
char  Nome_cliente[50][30];
float cubo[20][12][7];
```

Manipulação de Matrizes:

- Para processar uma matriz, utiliza-se laços ***for*** encaixados
  - um para variar o índice das linhas
  - outro para variar o índice das colunas
- Os índices variam de zero ao valor declarado, menos um;
- Ex1 - : Preencher a matriz mtrx sequencialmente por linhas, com os números de 1 a 200.

```
#include <stdio.h>
int 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++;
                }
return(0);
}
```

- Ex2 - Gerar coordenadas para os elementos de uma matriz

```
#include <stdio.h>
void main() {
        int i, j;
        for ( i = 0 ; i < 3 ; i++) {
                printf("\n");
                for ( j = 0 ; j < 4 ; j++)
                        printf("[%d][%d] ", i, j);
                }
        }
```

A saída produzida pelo programa é:

```
[0][0] [0][1] [0][2] [0][3]
[1][0] [1][1] [1][2] [1][3]
[2][0] [2][1] [2][2] [2][3]
```

Ou seja, as coordenadas de cada um dos elementos de uma matriz 3 x 4

- Ex3 - Ler uma matriz quadrada de ordem n e exibir apenas os elementos da diagonal principal

```
main( ) {
int i, j, n, V[100][100];
printf("Entre com a dimensao da matriz: ");
scanf("%d",&n);
printf("Entre com os elementos da matriz\n\n");
for ( i = 0 ; i < n ; i++ )              // percorre cada linha da matriz
 for ( j = 0 ; j < n ; j++ )             // percorre cada coluna da matriz
        scanf("%d", &V[i][j] );
for ( i = 0 ; i < n ; i++) {
 printf("\n");                          // muda de linha na tela
 for ( j = 0 ; j < n ; j++ ) {          // percorre cada coluna da matriz
        if( i == j ) printf("%d ",V[i][j] );
        printf("\t");}
        }
```

- Ex4 - Ler uma matriz quadrada de ordem n e exibir apenas os elementos da diagonal principal

```
#include <stdio.h>
main( ) {
int i, j, n, V[100][100];
printf("Entre com a dimensao da matriz: ");
scanf("%d",&n);
printf("Entre com os elementos da matriz\n\n");

for ( i = 0 ; i < n ; i++ )              // percorre cada linha da matriz
 for ( j = 0 ; j < n ; j++ )    // percorre cada coluna da matriz
        scanf("%d", &V[i][j] );

for ( i = 0 ; i < n ; i++) {
 printf("\n");                          // muda de linha na tela
 for ( j = 0 ; j < n ; j++ ) {  // percorre cada coluna da matriz
        if( i == j ) printf("%d ",V[i][j] );
        printf("\t");}
        }
 }
}
```

Inicialização de Matrizes:

- Atribuir valores na declaração do vetor:

```
int vetor[5] = {1,2,3,4,5};
```

- Atribuir valores na declaração da matriz:

```
float matriz[2][3] = {{1,2,3},{4,5,6}};
```

	0	1	2
0	1	2	3
1	4	5	6

- Ex5 - Ler, somar e imprimir o resultado entre duas matrizes inteiras que comportam 25 elementos cada:

```
main() {
int  MA[5][5], MB[5][5], MR[5][5];
int i, j;
for ( i = 0 ; i < 5 ; i++ )              // percorre cada linha da matriz
 for ( j = 0 ; j < 5 ; j++ )             // percorre cada coluna da matriz
        scanf("%d" %d", &MA[i][j], &MBi][j]);
 for ( i = 0 ; i < 5 ; i++ ){
        printf("\n");                    // muda de linha na tela
        for ( j = 0 ; j < 5 ; j++ ){     // percorre cada coluna da matriz
               MR[i][j] = MA[i][j] + MB[i][j];
               printf("%d\t" , MR[i][j]);
        }
 }
```

- Ex6 - Ler, multiplicar e imprimir o resultado entre duas matrizes inteiras que comportam 9 elementos:

```
main() {
int  MA[3][3], MB[3][3], MR[3][3];
int i, j, k;
for(i=0;i<3;i++)
 for(j=0;j<3; j++)
        scanf("%d", &MA[i][j]);
for(i=0;i<3;i++)
 for(j=0;j<3; j++)
        scanf("%d", &MB[i][j]);
for(i=0;i<3;i++){                        // percorre as linhas de MA e MR
 for(j=0;j<3; j++){                 // percorre as colunas de MB e MR
        MR[i][j]=0;
        for(k=0;k<3; k++)  //percorre coluna de MA e linha de MB
               MR[i][j]=MR[i][j]+MA[i][k] * MB[k][j];
        }
 }
for(i=0;i<3;i++){
 printf("\n");
 for(j=0;j<3; j++)
        printf("%d\t" , MR[i][j]);
 }
}
```

- Ex7 - Faça um programa que imprima uma matriz quadrada de dimensão N contendo:
  - o número 1 nos elementos abaixo da diagonal principal
  - o número 0 nos demais elementos
  - N deve ser menor ou igual a 20.

```
int main() {
int mat[20][20], n , i, j;
for (i=0; i<n; i++) {
 for (j=0; j<n; j++) {
        if (i > j)   /* Verifica se termo está abaixo da dp */
               mat[i][j] = 1;
        else
               mat[i][j] = 0;
        }
}
```

- Matrizes de strings

- Matrizes de strings são matrizes bidimensionais de **chars**.
- Uma string é um vetor, logo, um vetor de strings corresponde a uma lista de vetores.

```
char <nome> [<num_de _string>][<compr_das_strings>];
```

- Para acessar uma string individual é só usar apenas o primeiro índice.

```
<nome> [índice];
```

```
char strings [100][100];
for (i=0;i<100;i++) {
 printf ("\n\nDigite uma string: ");
 gets (strings[i]);
 printf ("%s\n",strings[i]);
 }
```

Inicialização de Matrizes

- Pode-se inicializar matrizes, assim como inicializar variáveis. Na declaração de uma matriz:

```
tipo nome_da_variável [tam1][tam2] ... [tamN] = {lista_de_valores};
```

- 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.
- Exemplos:

```
float vect [6] = { 1.3, 4.5, 2.7, 4.1, 0.0, 100.1 };    // inicialização de vetor
char str [6] = { 'J', 'o', 'a', 'o', '\0' };             // inicialização de vetor
char str [6] = "Joao";                                   // inicialização de vetor
```

	0	1	2	3	4	5
vtr	1.3	4.5	2.7	1.8	3.6	2.5
st1	'J'	'O'	'A'	'O'	'\0'	
st2	'J'	'O'	'A'	'O'	'\0'	

```
int mtz [3][4] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12 };
int mtz [ ][4] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12 };
```

i \ j	0	1	2	3
0	1	2	3	4
1	5	6	7	8
2	9	10	11	12

```
char mstr [3][6] = { "Joao", "Maria", "Jose" };
```

i \ j	0	1	2	3	4	5
0	'M'	'A'	'R'	'I'	'A'	'\0'
1	'J'	'O'	'A'	'O'	'\0'	
2	'J'	'O'	'S'	'E'	'\0'	

- Ex8 - Inicializando e exibindo um menu de opções

```
#include <stdio.h>
main( )   {
char menu[4][7]={"ABRIR","EDITAR","SALVAR", "SAIR"};
int i;
for(i=0; i<4; i++)
    printf(menu[i]);
}
```

i \ j	0	1	2	3	4	5	6
0	A	B	R	I	R	\0	
1	E	D	I	T	A	R	\0
2	S	A	L	V	A	R	\0
3	S	A	I	R	\0		

Note que a matriz menu é usada no programa como um vetor de strings.

- Ex9 - Leia 5 strings e as exiba na tela:

```
#include <stdio.h>
main () {
char strings [5][100];
int cont;
for (cont=0;cont<5;cont++) {
 printf ("\n\nDigite uma string: ");
 gets (strings[cont]);
 }
printf ("\n\n\nAs strings que voce digitou foram:\n\n");
for (cont=0;cont<5;cont++) {
 printf ("%s\n",strings[cont]);
}
```

- Ex10A - Construir um algoritmo que leia a tabela e informe ao usuário a distância entre duas cidades fornecidas por ele, até que ele forneça duas cidades iguais (origem e destino).

	BELÉM		MANAUS	PORTO VELHO	RIO DE JANEIRO	SALVADOR	
BELÉM		1611	5298	4397	3250	2100	2933
FORTALEZA	1611		5763	4865	2805	1389	3127
MANAUS	5298	5763		901	4374	5009	3971
PORTO VELHO	4397	4865	901		3473	4023	3070
RIO DE JANEIRO	3250	2805	4374	3473		1649	429
SALVADOR	2100	1389	5009	4023	1649		1962
SÃO PAULO	2933	3127	3971	3070	429	1962	

- Ex10B - Construir um algoritmo que permita ao usuário informar várias cidades, até inserir “xx”, e que imprima a distância total para cumprir todo o percurso especificado entre as cidades fornecidas.