Variáveis compostas.

Estruturas de dados homogêneas: vetores (Arrays)

Lista 3

Estruturas de dados homogêneas (vetores e matrizes)

- Estruturas de dados homogêneas (arrays) são estruturas de dados que consistem em itens de dados do mesmo tipo.
- Essas estruturas são entidades “estáticas”, já que permanecem do mesmo tamanho ao longo da execução do programa.

- Uma estrutura de dados homogênea é um grupo de posições de memória consecutivas, todas de mesmo nome e mesmo tipo (tamanho).
- Para fazer referência a uma posição particular ou elemento, especificamos o nome da estrutura e o número da posição daquele elemento.

Estruturas de dados homogêneas (vetores)

c [0]	5
c [1]	4
c [2]	3
c [3]	6
c [4]	-3
c [5]	7
c [6]	8
c [7]	-2
c [8]	1
c [9]	9

Estrutura (vetor) c

- Para criar um vetor de 10 elementos do tipo int precisamos reservar a memoria para 10 elementos.
- Isso pode ser feito usando a declaração:

```
int c[10];
```

Os 10 elementos do vetor c são denominados c[0], c[1], c[2], . . . , c [9].

Estruturas de dados homogêneas (vetores)

- Podemos criar vários vetores com uma única declaração.
- Por exemplo, podemos criar um vetor a de 10 elementos do tipo **int** e outro vetor b de 50 elementos do tipo **int** em uma única declaração:

```
int a[10], b[50];
```

- Podem ser declarados vetores de outros tipos:
  - float
  - double
  - char
- Por exemplo, pode ser usado um vetor do tipo **char** para armazenar um linha (string) de caracteres.

Estruturas de dados homogêneas (vetores)

```
int c[10]
```

- Atribuição de valor:

```
c[0] = 5;
c[1] = 10;
```

- Impressão:

```
printf(" %i ", c[0] );
```

Estruturas de dados homogêneas (vetores)

- Operações com elementos do vetor:

```
x = c[0] + 5;

c[1] = y – 10;
```

- Para imprimir a soma dos valores contidos nos três primeiros elementos do vetor `c` podemos escrever:

```
s = c[0] + c[1] + c[2] ;
```

- Para dividir por 2 o valor do sétimo elemento do vetor `c` e atribuir o resultado à variável x podemos escrever:

```
x =  c[6] / 2;
```

Estruturas de dados homogêneas (vetores)

- Os vetores ocupam um espaço determinado na memória.
- O programador especifica o tipo de cada elemento e o número de elementos exigidos pelo vetor, de forma que o compilador possa reservar a quantidade apropriada de memória.

Exemplo 1: Criação e Inicialização de um vetor

```c
#include <stdio.h>

int main()
{
  int c[10];
  int i;

  for ( i = 0; i < 10; i++ ) // inicializa o array
    c[i] = 5;

  printf("Elemento           Valor \n");

  for ( i = 0; i < 10; i++ ) // imprime o array
    printf(" %i              %i \n", i, c[i]);

  return 0;
}
```

Exemplo 2: Inicialização de um vetor

Os elementos de um vetor também podem ser inicializados na declaração do vetor na forma de uma lista de valores separados por vírgulas (entre chaves).

```
 3  #include <stdio.h>
 4
 5  int main()
 6  {
 7    int n[15] = {11, 22, 33, 44, 55, 66, 77};
 8    int i;
 9
10    printf("Elemento      Valor \n");
11    for ( i = 0; i < 15; i++ ) // imprime o array
12      printf(" %i              %i \n", i, n[i]);
13
14    return 0;
15  }
```

Exemplo 3: Valores digitados por usuário

- Os valores dos elementos são fornecidos pelo usuário

```
 3  #include <stdio.h>
 4
 5  int main()
 6  {
 7    int n[5];
 8    int i;
 9
10    for ( i = 0; i < 5; i++ ) // inicializa o array
11     {
12       printf("\n Digite elemento %i do vetor ", i);
13       scanf("%i", &n[i]);
14     }
15
16    printf("\n Elemento      Valor \n");
17
18    for ( i = 0; i < 5; i++ ) // imprime o array
19      printf(" %i              %i \n", i, n[i]);
20
21    return 0;
22  }
```

Estruturas de dados homogêneas (vetores)

Definição do tamanho do vetor

- Na declaração do tipo:

```
int n[10];
```

usamos um número especifico para definir o tamanho do vetor.

- As vezes é desejável definir o tamanho do vetor na forma de uma variável
- Somente variáveis do tipo “constantes” podem ser usadas para declarar o tamanho de um vetor.
- Variáveis “constantes” devem ser inicializadas com uma expressão constante e não podem ser modificadas.
- Variáveis constantes são também chamadas de constantes com nome ou variáveis somente leitura.

Ex:

```
const int arraySize = 10;
int n[arraySize];
```

Atenção: Dependendo do compilador os vetores declarados dessa forma não poderão ser inicializados no estilo:

int n[arraySize] = {0, 2, 3};

A inicialização deve ser feita manualmente.

Exemplo 4: Usando constante para definir o tamanho do vetor

Obs.: arraySize – é uma constante que determina o tamanho do vetor

```
 3  #include <stdio.h>
 4
 5  int main()
 6  {
 7    const int arraySize = 10; // tamanho do vetor
 8    int n[arraySize];
 9
10    int i;
11
12    for ( i = 0; i < arraySize; i++ ) // inicializa o array
13     {
14       printf("\n Digite elemento %i do vetor ", i);
15       scanf("%i", &n[i]);
16     }
17
18    printf("\n Elemento      Valor \n");
19
20    for ( i = 0; i < arraySize; i++ ) // imprime o array
21      printf(" %i                %i \n", i, n[i]);
22
23    return 0;
24  }
```

Exemplo 5: Soma dos elementos do vetor

```
6     const int arraySize = 10; // tamanho do vetor
7     float n[arraySize];
8     float sum = 0;
9     int i;
10  //-------------------------------------------------------
11    for ( i = 0; i < arraySize; i++ ) // inicializa o array
12     {
13       printf("\n Digite elemento %i do vetor ", i);
14       scanf("%f", &n[i]);
15     }
16  //-------------------------------------------------------
17    printf("\n Elemento      Valor \n");
18    for ( i = 0; i < arraySize; i++ ) // imprime o array
19      printf(" %i              %.2f \n", i, n[i]);
20  //-------------------------------------------------------
21    for ( i = 0; i < arraySize; i++ )
22     {
23       sum = sum + n[i];
24     }
25
26    printf("\n Soma dos elementos do vetor = %.2f \n\n", sum);
27
```

Operações com string

- Uma sequencia de caracteres (string) na realidade é um vetor que termina com o carácter especial **null** ('\0')
- Por exemplo:

```
char greeting[6] = {'H', 'e', 'l', 'l', 'o', '\0'};
```

ou

```
char greeting[] = "Hello";
```

Na realidade não tem necessidade de colocar o **null** no final da sequencia de caracteres pois o compilador já faz isso automaticamente.

Exemplo 6: Operações com string

```c
#include <stdio.h>

int main ()
{
   char greeting[6] = {'H', 'e', 'l', 'l', 'o', '\0'};
   char greeting2[] = "World";
   printf("Mensagem: %s %s \n", greeting, greeting2 );

   char str[100];

   printf( "\n Digite mensagem: ");
   scanf("%s", str);

   printf( "\n Voce digitou: %s ", str);

   return 0;
}
```

Exemplo 7: calculo de votos

- Programa recebe as escolhas dos 10 eleitores e calcula os votos para 5 candidatos.
- **Modifique programa permitindo:**
  - receber a quantidade de votos a ser analisada
  - receber os votos para 7 candidatos
  - calcular a percentagem de votos de cada candidato e votos nulos

```
 7    const int arraySize = 6; // tamanho do vetor
 8    int cand[arraySize];
 9    int i, v;
10    int n = -1;
11
12    printf("\n Apuração de votos: ");
13    for ( i = 0; i < arraySize; i++ ) // inicializa o array
14       cand[i] = 0;
15
16    for ( i = 0; i < 10; i++ )
17     {
18       printf("\n Escolha o candidato (1..5): ");
19       scanf("%i", &v);
20       if(v >= 0 && v <=5 )
21         cand[v]++;
22       else
23         cand[0]++;
24     }
25
26    printf("\n Candidato      Votos \n");
27    for ( i = 1; i < arraySize; i++ ) // imprime o array
28      printf(" %i                %i \n", i, cand[i]);
29    printf("\n Votos nulos: %i \n", cand[0]);
```

Variáveis compostas.

Estruturas de dados homogêneas (Matrizes)

Definição de Matriz

- Matriz é uma variável composta homogênea multidimensional;
- Ela é formada por uma sequência de variáveis de mesmo tipo, que possuem o mesmo identificador (mesmo nome) e são alocadas sequencialmente na memória;
- Como as variáveis têm o mesmo nome, o que as distingue são índices que referenciam sua localização dentro da estrutura;
- Uma variável do tipo matriz precisa de um índice para cada uma das suas dimensões.

Estruturas multidimensionais (matrizes)

	0	1	2
0	a[0][0]	a[0][1]	a[0][2]
1	a[1][0]	a[1][1]	a[1][2]
2	a[2][0]	a[2][1]	a[2][2]

- Para identificar um elemento específico de uma tabela, precisamos especificar dois índices:
  - o primeiro (por convenção) identifica a linha do elemento
  - o segundo (por convenção) identifica a coluna do elemento
- As estruturas que exigem dois índices são chamadas de estruturas bidimensionais ou matrizes.

Estruturas multidimensionais (matrizes)

- As estruturas multidimensionais em C/C++ podem ter vários índices
- Um uso comum é representação das tabelas de valores que consistem em informações organizadas em linhas e colunas.
- A quantidade dos índices pode chegar até 12 (dependendo da versão da linguagem).

Matrizes

- Para criar uma matriz 2 x 2, i.e. uma matriz de 4 elementos do tipo int, podemos escrever:

```
int m[2][2];
```

- Para declarar e ao mesmo tempo inicializar (i.e. atribuir os valores iniciais) podemos escrever:

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

- Os valores são agrupados por linha e colocados entre chaves
- Se não houver inicializadores suficientes para uma determinada linha, os elementos restantes daquela linha são inicializados com 0

Matrizes

- Atribuição de valor:

```
m[0][0] = 5;
m[1][0] = 10;
```

- Impressão do elemento m[0][0]:

```
printf(" %i ", m[0][0] );
```

Matrizes

- Operações com elementos de uma matriz:

```
x = m[0][0] + 5;

m[0][0] = 5 – 10;
```

- Para calcular a soma dos valores contidos na primeira **linha** da matriz m podemos escrever:

```
sum_line = m[0][0] + m[0][1];
```

- Para calcular a soma dos valores contidos na primeira **coluna** da matriz m podemos escrever:

```
sum_column = m[0][0] + m[1][0];
```

Matrizes

- Exatamente como no caso dos vetores, devemos reservar um determinado espaço na memoria para armazenar uma matriz.
- Cuidado na hora de atribuir os valores!
- A linguagem em si não avisa quando o limite de uma matriz foi excedido.

Exemplo 8: Criação e inicialização de uma matriz 2x2

```c
// Exemplo 1: matrizes
#include <stdio.h>

int main()
{
  int i, j;
  int m[2][2] = { { 1, 2}, { 3, 4} };
  //int m[2][2] = { { 1, 2} };

  printf("\n Programa imprime uma matriz 2x2 com valores pre definidos: \n\n");
  for ( i=0; i<2; i++)
   {
     for ( j=0; j<2; j++)
          printf("%i   ", m[i][j] );
     printf("\n");
   }

  return 0;
}
```

Exemplo 9: Leitura de dados para uma matriz 2 x 2

```c
#include <stdio.h>

int main()
{
  const int line = 2, column = 2;
  int i, j;
  int m[line][column];
//-------------------------------------------------------------
 printf("\n Leitura de dados para matriz 2x2: \n\n");
 for ( i=0; i < line; i++)
   {
     for ( j=0; j < column; j++)
     {
       printf("\n Digite elemento [%i][%i] da matriz: ", i, j);
       scanf("%i", &m[i][j]);
     }
   }
//-------------------------------------------------------------
  printf("\n\n\n Matriz m: \n\n");
  for ( i=0; i < line; i++)
   {
     for ( j=0; j < column; j++)
          printf("%i   ", m[i][j] );
     printf("\n");
   }

  return 0;
}
```