Centro *Universitário* de Barra Mansa
Curso: Sistemas de Informação – 4º período
Disciplina: Programação Orientada a Objetos II

Unidade I - Matriz

Prof. M.Sc. Anderson Simeão

1.1 – Introdução

- Uma matriz é um tipo de dado em C++ usado para representar uma coleção de variáveis de mesmo tipo e que compartilham um mesmo nome.
- Imagine o seguinte problema: calcular a média aritmética das notas de prova de cinco alunos. Poderia ser escrito o seguinte programa:

1.1 – Introdução

```
#include <iostream>
using namespace std;

int main()
{
  int nota1, nota2, nota3, nota4, nota5;

  cout << "Informe a nota do aluno 1: ";
  cin >> nota1;
  cout << "Informe a nota do aluno 2: ";
  cin >> nota2;
  cout << "Informe a nota do aluno 3: ";
  cin >> nota3;
  cout << "Informe a nota do aluno 4: ";
  cin >> nota4;
  cout << "Informe a nota do aluno 5: ";
  cin >> nota5;

  int media = (nota1 + nota2 + nota3 + nota4 + nota5) / 5;
  cout << "Média calculada: " << media << endl;

  system("PAUSE");
  return 0;
}
```

1.1 - Introdução

- Imagine agora se fosse necessário encontrar a média aritmética das notas de uma turma de 50 alunos ou mesmo de toda a escola com 1000 alunos. Seria uma tarefa bem mais trabalhosa!
- É evidente que necessita-se de uma maneira conveniente para realizar tal tarefa.
- Matriz é o tipo de dado oferecido pela linguagem C++ para esse propósito.

1.1 - Introdução

- Uma matriz é um conjunto de variáveis do mesmo tipo, referidas por um único nome, em que cada variável é diferenciada por meio de um número denominado "índice". Os colchetes são usados para conter o índice.
- A declaração

```
int notas[5];
```

aloca memória para armazenar cinco variáveis do tipo **int** e anuncia que notas é uma matriz de cinco elementos.

- Vamos reescrever o programa anterior utilizando uma matriz:

1.1 – Introdução

```
#include <iostream>
using namespace std;

int main()
{
  int notas[5], media = 0;

  for(int i=0; i<5; i++)
  {
    cout << “Informe a nota do aluno “ << (i + 1) << “ : “;
    cin >> notas[i];
    media = media + notas[i];
  }

  media = media / 5;
  cout << "Média calculada: " << media << endl;

  system("PAUSE");
  return 0;
}
```

1.2 – Declaração da Matriz

- Em C++, as matrizes precisam ser declaradas, como quaisquer outras variáveis, para que o compilador conheça o tipo de seus elementos e reserve espaço de memória suficiente para armazená-las.
- Os elementos da matriz são guardados em uma sequência contínua de memória, isto é, um seguido ao outro, o que não ocorre quando criamos variáveis separadas.
- O que diferencia a declaração de uma matriz da declaração de qualquer outra variável é a parte que se segue ao nome, ou seja, o par de colchetes ([e]) que envolve um número inteiro indicando o tamanho da matriz.

1.2 – Declaração da Matriz

- A instrução

```
int notas[5];
```

  informa que a matriz notas é formada por cinco elementos do tipo **int**. Por definição, uma matriz é composta por elementos de um único tipo.

- O valor que dimensiona a matriz, na sua declaração, deve ser uma constante inteira. Assim, não se pode dimensionar uma matriz por meio de uma variável.

1.3 – Referindo aos Elementos da Matriz

- Cada um dos elementos da matriz é referido individualmente por meio de um número inteiro, entre colchetes, seguindo o nome da matriz. Esse número tem um significado diferente quando se refere a um elemento da matriz e na declaração da matriz, onde indica sua dimensão.
- Quando se refere a um elemento da matriz, esse número especifica a posição do elemento na matriz. Os elementos são sempre numerados por índices iniciados por zero.

1.3 – Referindo aos Elementos da Matriz

- Por exemplo, a instrução

```
notas[2] = 90;
```

- Atribui o valor 90 ao terceiro elemento da matriz, pois a numeração começa em zero. O último elemento da matriz possui um índice de uma unidade menor que a dimensão da matriz.
- O índice utilizado para referir elementos de uma matriz pode ser o valor de uma variável inteira ou uma constante. No exemplo visto anteriormente, foi utilizada a variável `i` como índice.

1.3 – Referindo aos Elementos da Matriz

- Observe a instrução:

```
cin >> notas[i];
```

- Quando se escreve `notas[i]`, está sendo escrito o nome de uma variável do tipo int como outra qualquer. Assim, em todo lugar onde se pode utilizar o nome de uma variável `int`, pode-se usar `notas[i]`.

1.4 – Matrizes de Outros Tipos de Elementos

- O fato de uma matriz ser composta por uma série de elementos de um único tipo de dado permite a escolha de qualquer tipo de variável para a matriz. Suponhamos que se queira que as notas dos alunos fiquem no intervalo de 0 a 10 e que não seja desprezada a parte fracionária. A solução seria usar uma matriz de elementos do tipo **float**.

1.4 – Matrizes de Outros Tipos de Elementos

```
#include <iostream>
#include <iomanip>

using namespace std;

int main()
{
  float notas[5], media = 0.0;

  cout << setprecision(2);

  for(int i=0; i<5; i++)
  {
    cout << "Informe a nota do aluno " << (i + 1) << " : ";
    cin >> notas[i];
    media = media + notas[i];
  }

  media = media / 5;
  cout << "Média calculada: " << media << endl;

  system("PAUSE");
  return 0;
}
```

Exercício: Receber o preço de custo e o percentual de lucro de cinco produtos . Calcular o preço de venda de cada produto. Após o cálculo, deverá ser mostrado o preço de custo e o preço de venda de cada produto.

1.5 – Matriz Bidimensional

- Uma matriz bidimensional se caracteriza pelo fato de possuir "linhas" e "colunas".
- Para armazenar/acessar elementos de uma matriz bidimensional é necessária a utilização de dois índices, um para linha e outro para coluna.

1.5 – Matriz Bidimensional

- Exemplo: Armazenar 9 números inteiros em uma matriz. Após isso, exibir os elementos armazenados.

1.5 – Matriz Bidimensional

```
#include <iostream>
#include <stdio.h>
#include <stdlib.h>
using namespace std;

int main()
{
  int mnum[3][3], l = 0, c = 0;

  for (l=0;l<3;l++)
  {
   for (c=0;c<3;c++)
   {
     cout << "Informe um numero inteiro: ";
     cin >> mnum[l][c];
   }
  }

  for (l=0;l<3;l++)
  {
   for (c=0;c<3;c++)
   {
     cout << mnum[l][c] << " ";
   }
   cout << "\n";
  }
}
```

Exercício: Receber e armazenar em uma matriz 16 números inteiros. Após isso, receber mais um número inteiro. Esse número deverá ser procurado dentro da matriz e, caso seja encontrado, deverá ser exibida a posição em que ele se encontra dentro da mesma, ou seja, linha e coluna.