visionlab
visualization laboratory

Pontifícia Universidade Católica
do Rio de Janeiro

Introdução a Programação de Jogos

Aula 05 – Vetores e Matrizes

Prof. Augusto Baffa
< abaffa@inf.puc-rio.br>

Introdução

- Até agora nós temos usado variáveis simples para armazenar valores usados por nossos programas.

- Em várias situações, precisamos armazenar um **conjunto de valores**.

- A partir de agora vamos aprender a usar um mecanismo que nos permite armazenar um conjunto de valores na memória do computador.
  - Posteriormente, estes valores podem ser livremente processados de forma eficiente, pois já estariam na memória do computador.

Vetores

- Podemos armazenar um conjunto de valores na memória do computador através do uso de **vetores (arrays)**

- O vetor é a forma mais simples de **organizarmos dados** na memória do computador.

- Com vetores, os valores são armazenados na memória do computador **em sequência**, um após o outro, e podemos livremente acessar qualquer valor do conjunto.

Vetores

- Ao declarar um vetor (conceito análogo ao de declaração de uma variável simples) é necessário informar **dimensão do vetor**.
  - O tamanho define o **número máximo de elementos** que poderá ser armazenado no espaço de memória que é reservado para o vetor.
- Também é necessário informar o **tipo dos valores** que serão armazenados no vetor (por exemplo, int, float ou double).
  - Em um vetor, só podemos armazenar valores de um mesmo tipo.

Declaração e Inicialização

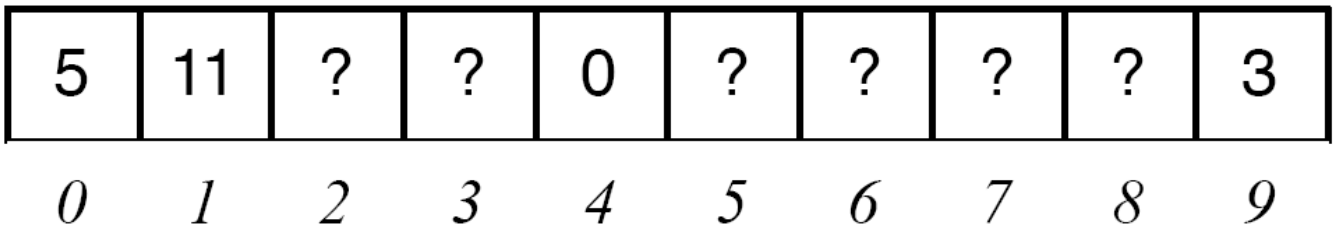
- **Declaração de um vetor:**

```
int meu_vetor[10];
```

  - Reserva um espaço de memória para armazenar 10 valores inteiros no vetor chamado meu_vetor.

- **Inicialização de algumas posições do vetor meu_vetor:**

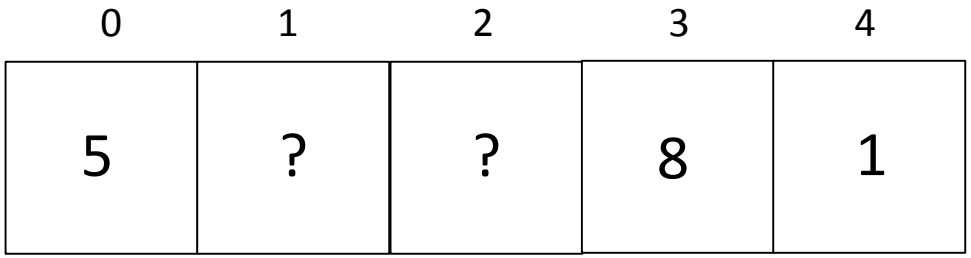
```
meu_vetor[0] = 5;
meu_vetor[1] = 11;
meu_vetor[4] = 0;
meu_vetor[9] = 3;
```



Declaração e Inicialização

- É possível **acessar os valores do vetor** através de seu **índice**.

```
int meuvetor[5];
```




```
meuvetor[0] = 5;
meuvetor[3] = 8;
meuvetor[4] = 1;
```

Declaração e Inicialização

- **Exemplos de Declaração:**

```
int a, b[20];

float c[10];

double d[30], e, f[5];
```

- **Declaração e Inicialização:**

```
int teste[5] = {12, 5, 34, 32, 9};

float vetor1[3] = {2.5, 5.8, 10.1};
```

Exemplo 1: Imprimindo os Valores Armazenados em um Vetor

```
#include <stdio.h>

int main(void)
{
   int i;
   float v[6] = {2.3, 5.4, 1.0, 7.6, 8.8, 3.9};

   for (i=0; i<6; i++)
   {
      printf("%f", v[i]);
   }

   return 0;
}
```

Exemplo 2: Somatório dos Valores Armazenados em um Vetor

```
#include <stdio.h>

int main(void)
{
   int i;
   float v[6] = {2.3, 5.4, 1.0, 7.6, 8.8, 3.9};
   float s = 0.0;

   for (i=0; i<6; i++)
   {
      s = s + v[i];
   }
   printf("%f", s);
   return 0;
}
```

Exemplo 3: Encontrar o Maior Valor

```
#include <stdio.h>

int main(void)
{
   int i;
   float v[6] = {2.3, 5.4, 1.0, 7.6, 8.8, 3.9};
   float maior_valor = v[0];
   for (i=0; i<6; i++)
   {
     if(v[i] > maior_valor)
       maior_valor = v[i];
   }
   printf("%f", maior_valor);
   return 0;
}
```

Exemplo 4: Calculo da Média

- Dada uma turma com n alunos (onde n é conhecido a priori), crie um programa para obter a notas dos alunos e calcula a média da turma.

```
7.5
8.4
9.1
4.0
5.7
4.3
```

```c
#include <stdio.h>
#define NUM_ALUNOS 6

int main (void)
{
  float notas[NUM_ALUNOS];
  float media, soma = 0.0;
  int i;
  /* leitura dos dados via teclado para o vetor */
  for(i=0; i<NUM_ALUNOS; i++)
  {
    printf("Entre com a nota do aluno %d: ", i+1);
    scanf("%f", &notas[i]);
  }
  /* soma das medias dos alunos */
  for(i=0;i<NUM_ALUNOS;i++)
    soma = soma + notas[i];
  media = soma/NUM_ALUNOS;
  printf("Media da turma = %.2f\n.", media);
  return 0;
}
```

Matrizes

- Uma **matriz** representa e armazena um conjunto bidimensional de valores na memória do computador.

- É uma **tabela de variáveis** de mesmo tipo que ocupa uma região contínua de memória.

- Exemplo de matriz de inteiros:

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

Matrizes

- Para declarar uma matriz, precisamos especificar o **tipo** das variáveis da matriz e o **tamanho das duas dimensões** da matriz.

```
tipo nome_matriz[tamanho_x][tamanho_y];
```

- **Exemplo:**

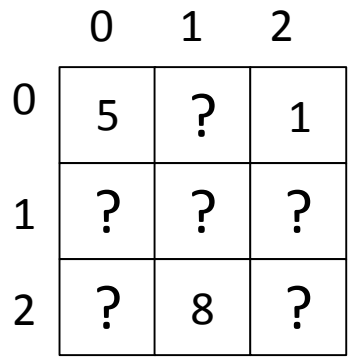
```
int minha_matriz[3][3];
```

?	?	?
?	?	?
?	?	?

Matrizes

- É possível **acessar os valores da matriz** através de seu **índice bidimensional**.

```
int minha_matriz[3][3];
```




```
minha_matriz[0][0] = 5;
minha_matriz[1][2] = 8;
minha_matriz[2][0] = 1;
```

Matrizes

- **Exemplos de Declaração:**

```
int a[10][10];
float matriz1[20][20];
int mapa[100][100];
```

- **Declaração e Inicialização:**

```
int teste[3][3] =
{{2,5,1},{3,7,2},{9,1,5}};
```

Matrizes

Exemplo 1:

"Crie um programa que represente o conteúdo da tabela de notas abaixo e escreva a média de cada uma dos alunos"

	Nota1	**Nota2**	**Nota3**
Aluno 1	7.5	8.5	7.8
Aluno 2	8.4	10.0	9.5
Aluno 3	9.2	6.8	9.1
Aluno 4	4.0	5.2	4.6
Aluno 5	5.7	3.4	4.3
Aluno 6	4.3	6.0	5.8

```
#include <stdio.h>

int main(void)
{
  float notas[3][6], media;
  int x, y;
  ...
  for(y = 0; y < 6; y++)
  {
     media = 0;
     for(x = 0; x < 3; x++)
     {
        media = media + notas[x][y];
     }
     media = media/3;
     printf("Aluno %d Media: %f", y, media);
  }
  return 0;
}
```

Exercícios

Lista de Exercícios 04 – Vetores

http://www.inf.puc-rio.br/~abaffa/iue1503/