Aula 9: Vetores

Prof. Sérgio Montazzolli Silva
smsilva@uel.br

Introdução

- Vetores são estruturas que armazenam sequências de dados do mesmo tipo.

- Por exemplo, ao invés de criarmos muitas variáveis individuais do mesmo tipo para representar uma certa sequência de valores (num1,num2,num3,...), podemos criar apenas um vetor.

- Na matemática, escrevemos um vetor da seguinte forma:

$$\vec{x} = (x_1, x_2, \ldots, x_n)$$

Introdução

- Na denotação anterior, não podemos saber o tipo dos números (reais, inteiros, complexos, etc), mas podemos saber o tamanho do vetor, que é n

- Em programação C, devido ao alto formalismo exigido, especificar o tipo e o tamanho do vetor é obrigatório!

- Nesta aula trataremos da criação de vetores estáticos

Definição

- Em C, vetores são definidos da seguinte forma:
  - **tipo** nome_do_vetor[tamanho];

- Exemplos:
  - Vetor de 10 números inteiros:
    - **int** meu_vetor_inteiro[10];
  - Vetor de 55 caracteres:
    - **char** meu_vetor_char[55];
  - Vetor de 3210 números reais:
    - **float** meu_vetor_real[3210];

Atribuição de valores

- Para atribuir um valor *v* para a i-ésima posição de um vetor **a**, escrevemos:
  - a[i] = v;
- As posições de um vetor se iniciam em 0 (zero), logo, para atribuir um valor para *primeira* posição de um vetor, escrevemos:
  - vetor[0] = valor;
- Para a segunda posição:
  - vetor[1] = valor;
- E assim por diante...

Atribuição de valores

- A atribuição pode ser feita através de variáveis, e deve respeitar o **tipo** do vetor, por exemplo:
  - Dado um vetor de números inteiros:
    - **int** X[12];
  - Uma variável inteira:
    - **int** z = 60;
  - Uma variável real:
    - **float** r = 13.2;
  - Atribuir um valor de mesmo tipo a um vetor é correto:
    - X[4] = z;
  - Porém a seguinte operação de atribuição retornará um erro ou aviso, e deve ser evitada:
    - X[4] = r;

Atribuição de valores

- Lembre-se que ao iniciar um vetor, da mesma forma que ocorre com qualquer variável, todos os dados deste vetor não foram inicializados. Logo, eles podem conter qualquer valor.
- Existe uma forma fácil de escrever todos os valores que devem ser inicializados a cada posição de um vetor, que é usar {} com números separados por vírgula. Exemplo:

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

  - Este exemplo cria uma variável do tipo vetor de inteiros, com o nome "vetor", onde na posição um (de índice 0) é atribuído o número 1, na posição dois (de índice 1), o número 2, e assim por diante até o número 5 ser atribuido.

  - Também é possível escrever da seguinte forma:
    - **int** vetor[] = {1,2,3,4,5}
    - Como na atribuição existem 5 valores separados por vírgula, é criado um vetor de tamanho 5.

Leitura através de scanf()

- Como vetores são sequências de valores, é possível também fazer a leitura de valores utilizando a função scanf()
- Basta escolher uma posição do vetor, e passa-la como argumento da função scanf(), utilizando também o símbolo &.
- Exemplo:
  - **int** vetor[5];
  - **scanf**("%i", &vetor[2]);
  - As duas linhas de comando acima criam um vetor de inteiros de tamanho 5, e armazena na terceira posição (2 quando conta-se a partir de zero) o valor entrado pelo usuário

Escrita com printf()

- Para escrevermos na tela um elemento de um vetor, basta utilizarmos os printf() de forma semelhante a variáveis comuns, adicionando a posição do vetor que queremos imprimir:

```
int vetor[5]={1,2,3,4,5};
printf("%i",vetor[2]);
```

- Podemos também imprimir todos os valores na tela através de um loop do tipo FOR:

```
for (i = 0; i < 5; i++) {
        printf("%d\n",vetor[i]);
}
```

Strings

- Strings são vetores de caracteres, onde o último caractere é "\0“ (barra zero)
- Por exemplo:
  - **char** nao_eh_string[] = {'A','u','l','a',' ','C'};
  - **char** eh_string[] = {'A','u','l','a',' ','C','\0'};

- Usando uma notação mais enxuta:
  - **char** eh_string[] = "Aula C";
    - Note que apesar de não haver o caractere '\0' ao final, o compilador o insere automaticamente ao criar o vetor
    - Isso apenas ocorre quando se cria vetores do tipo **char** utilizando “” na atribuição
    - Não funciona para outros tipos de dados, como inteiros e reais

Strings

- Uma string pode ser impressa na tela utilizando printf() através do especificador %s

- Por exemplo:

```
char string[] = "minha primeira string!";
printf("Esta eh a %s\n",string);
```

- Veremos mais sobre manipulação de strings em outro momento.

Exercícios em Sala

- Escreva um programa que crie um vetor de números inteiros de 10 posições. Atribua valores de 0 a 9 para cada posição, e imprima este vetor de trás para frente.

- Agora crie a string "*Eu estudo C*" e imprima esta string de trás para frente.

Lembre-se: um vetor ou string de tamanho N se inicia na posição 0 e termina na posição N-1

Exercícios em Sala

- Faça um programa que crie um vetor de números reais de 5 posições e permita que o usuário entre com um valor para cada posição (através da função scanf()). Ao final, imprima na tela todos os números digitados pelo usuário.