Aula 7: Vetores Numéricos

Professor(a): João Eduardo Montandon (103)
Virgínia Fernandes Mota (106)
jemaf.github.io
http://www.dcc.ufmg.br/∼virginiaferm

INTRODUÇÃO A PROGRAMAÇÃO - SETOR DE INFORMÁTICA

Estruturas de Dados

- Em diversas situações os tipos básicos de dados (inteiro, real, caracter, ...) não são suficientes para representar a informação que se deseja armazenar.
  - Exemplo, uma palavra: "AULA".
  - Valor de 50 produtos em uma loja
- Existe a possibilidade de construção de novos tipos de dados a partir da composição (ou abstração) de tipos de dados primitivos.
- Esses novos tipos tem um formato denominado ESTRUTURA DE DADOS, que define como os tipos primitivos estão organizados.

Motivação

- Problema 1: Como poderíamos fazer um algoritmo para ler 50 notas de uma turma e calcular sua média?

```c
#include <stdio.h>
int main(){
    int i;
    float media, nota, soma = 0.0;
    for (i = 0; i < 50; I++){
        printf("Digite sua nota");
        scanf("%f", &nota);
        soma = soma + nota;
    }
    media = soma / 50;
    printf("%f", media);
    return 0;
}
```

- Mas e se eu precisar saber o valor da 1a, 4a ou 40a nota?

Motivação

- Problema 2: Fazer um programa para ler 50 notas de uma turma e calcular a sua média. Imprimir as notas lidas juntamente com a média da turma como na tabela.

Nota	Média
8.0	7.75
4.6	7.75
2.3	7.75
7.8	7.75
...	...

- Como fazê-lo? No exemplo anterior, uma nota é sobreposta por outra em cada iteração do for.
- A solução é armazenar todas as 50 notas lidas... Como?

Variáveis compostas homogêneas

- Quando uma determinada estrutura de dados for **composta de variáveis com o mesmo tipo primitivo**, temos um conjunto homogêneo de dados.
- Essas variáveis são chamadas de variáveis compostas homogêneas.

Variáveis compostas homogêneas unidimensionais

- As variáveis compostas homogêneas unidimensionais são utilizadas para representar arranjos unidimensionais de elementos de um mesmo tipo, em outras palavras, são utilizadas para representar **vetores**.

Vetores: Declaração

- A sintaxe para declaração de uma variável deste tipo é a seguinte:
  tipo identificador [qtd de elementos];
- Exemplo: vetor com 5 elementos do tipo inteiro
  int dados[5]; //5 elementos quaisquer do tipo inteiro.

Vetores: Referência (Manipulação)

- Cada um dos elementos de um vetor é referenciado individualmente por meio de um número inteiro entre colchetes após o nome do vetor.
- int dados[5];

3	2	4	7	1

- Considerando o vetor dados, quais valores serão atribuídos a X, Y e Z nos exemplos abaixo???
  X = dados[0];
  Y = dados[1];
  Z = dados[4];

Vetores: Referência (Manipulação)

- Cada um dos elementos de um vetor é referenciado individualmente por meio de um número inteiro entre colchetes após o nome do vetor.
- int dados[5];

3	2	4	7	1

- Considerando o vetor dados, quais valores serão atribuídos a X, Y e Z nos exemplos abaixo???
  X = dados[0]; 3
  Y = dados[1]; 2
  Z = dados[4]; 1

Vetores: Exemplos

- O programa a seguir, usa o comando **for** para inicializar com zeros os elementos de um array (vetor) inteiro n de 10 elementos e o imprime sob a forma de uma tabela.

```
1  #include <stdio.h>
2  int main(){
3      int n[10], i;
4      for (i = 0; i < 10; i++) {
5          n[i] = 0;
6      }
7      printf("Elemento \t  Valor \n");
8      for (i = 0; i < 10; i++) {
9          printf("%d \t  %d \n",  i, n[i]);
10     }
11     return 0;
12 }
```

Vetores: Exemplos

- O programa abaixo inicializa os dez elementos de um array **s** com os valores: 2, 4, 6, ..., 20 e imprime o array em um formato de tabela.

```
1  #include <stdio.h>
2  int main(){
3      int N = 10;
4      int s[N], i;
5      for (i = 0; i < N; i++) {
6          s[i] = 2 + 2*i;
7      }
8      printf("Elemento \t  Valor \n");
9      for (i = 0; i < N; i++) {
10         printf("%d \t %d \n",  i, n[i]);
11     }
12     return 0;
13 }
```

Vetores e Subrotinas

- Vetores são passados sempre por referência.
- O tamanho do vetor pode ser omitido.
- No exemplo abaixo é apresentado um procedimento imprimeVetor que imprime um vetor de tamanho tam.

```
1  #include <stdio.h>
2  #define TAMANHO 10
3  void imprimeVetor (int vet[], int tam) {
4       int i;
5       for (i = 0; i <= tam - 1; i++) {
6            printf("%d \n", vet[i]);
7       }
8  }
9
10 int main(){
11      int s[TAMANHO], i;
12      for (i = 0; i <= TAMANHO - 1; i++) {
13           printf ("Informe o valor do vetor na posicao %d: ", i);
14           scanf ("%d", &s[i]);
15      }
16      imprimeVetor(s, TAMANHO);
17      return 0;
18 }
```

Exemplo

Exemplo 1:
Criar uma função que receba um vetor de números reais e seu tamanho e retorne o índice do maior valor contido no vetor. Se houver mais de uma ocorrência do maior valor, retornar o índice do primeiro. Faça um programa principal para testar a função.

Exemplo

```c
#include <stdio.h>
int encontraMaior (float vet[], int tam) {
    int indice, i;
    float maior = vet[0];
    indice = 0;
    for (i = 1; i < tam; i++) {
        if ( vet[i] > maior) {
            maior = vet[i];
            indice = i;
        }
    }
    return indice;
}

int main(){
    float vetor[6] = {3.0, 4.3, 5.6, 2.8, 7.9, 3.4};
    int posicao;
    posicao = encontraMaior(vetor, 6);
    printf("O maior valor se encontra na posicao %d e vale %d", posicao,
        vetor[posicao]);
    return 0;
}
```

Exemplo

Exemplo 2:
Criar um procedimento em C que receba um vetor de números reais e um valor inteiro representando o seu tamanho. Essa função deverá ordenar o vetor em ordem crescente.

Exemplo

```c
#include <stdio.h>
void ordenaVetor (float vet[], int tam) {
     int i, j;
     float aux;
     for (i = 0; i <= (tam - 2); i++){
          for (j = (tam-1); j > i; j- -){
               if (vet[j] < vet[j-1]){
                    aux = vet[j];
                    vet[j] = vet[j-1];
                    vet[j-1] = aux;
               }
          }
     }
}

int main(){
     int i;
     float vet[5]={11.0,22.0,3.0,44.0,5.0};
     ordenaVetor(vet, 5);
     for (i=0; i < 5; i++)
          printf("%.2f \n",vet[i]);
     return 0;
}
```

Alocação estática x alocação dinâmica

- Até agora vimos como alocar um espaço estático para as variáveis.
- A alocação estática é uma estratégia de alocação de memória na qual a memória que um tipo de dados pode vir a necessitar é alocada toda de uma vez.
  - int v[10]; $\rightarrow$ aloca um espaço contíguo de 10 valores inteiros
- Mas e se eu não souber o tamanho que devo alocar?

Alocação estática x alocação dinâmica

- A alocação dinâmica é uma técnica que aloca a memória sob demanda.
- Os endereços podem ser alocados, liberados e realocados para diferentes propósitos, durante a execução do programa.
- Em C usamos **malloc(n)** para alocar um bloco de memória de tamanho n bytes.
- é responsabilidade do programador de liberar a memória após seu uso.
- Veremos como utilizar a alocação dinâmica mais adiante no curso!!

Exercícios

1. Faça um algoritmo que leia, via teclado, 20 valores do tipo inteiro e determine qual o menor valor existente no vetor e imprima o valor e seu índice no vetor.
2. Desenvolva um programa que leia um vetor de números reais, um escalar e imprima o resultado da multiplicação do vetor pelo escalar.
3. Faça um programa que leia 2 vetores de tamanho 10 e calcule o produto escalar deles.
4. Faça um programa que leia um vetor de um tamanho escolhido pelo usuário e calcule a média aritmética de seus valores.

Na próxima aula...

Vetores de Caracteres