Universidade Federal do Espirito Santo
Centro Tecnologico

{introcomp}

Working 6: Vetores I

Objetivos:

- Compreender o funcionamento de vetores e dominar sua implementação;

Prazo de Envio: Sexta, 14/10, 23:55.

PET
Engenharia de Computação

1 INTRODUÇÃO

Vetores é uma estrutura de dados crucial para o domínio de programação e o desenvolvimento de tecnologias computacionais. Em diversos algoritmos otimizados os vetores são utilizados. Ordenar uma quantidade de dados, procurar um item, achar o melhor caminho são umas das várias implementações possíveis com vetores. Vamos aprender mais sobre ele e dominar seu funcionamento e dicas de implementação!

1.1 SUA IMPORTÂNCIA

Imagine que se queira fazer um programa para manusear dados de um mesmo tipo - as idades das pessoas de uma família, ou os salários dos funcionários de uma empresa, por exemplo. Uma alternativa seria a utilização de muitas variáveis ou então a leitura de valores sempre que necessários. Mas será que essas seriam as melhores soluções? E no caso de e um programa necessitarmos de manipular 'n' dados com 'n' lido pelo teclado? Como criaríamos 'n' variáveis em tempo de execução? Não é possível declarar 'n' variáveis, mas com vetores isso é possível! Vemos entender como ele é representado.

2 REPRESENTAÇÂO

Veja a seguir a maneira mais comum de se representar um vetor graficamente:

	0	1	2	3	4	5	6	7	8	9
vet	2	3	5	7	11	13	17	19	23	29

A figura acima representa um vetor de inteiros chamado vet com os nove primeiros números primos.
O terceiro elemento, de índice igual a 2, é o número 5 e o de índice 4 é o número 11. É importante lembrar que o primeiro índice de um vetor, na linguagem C, é o zero. Analisando a Figura acima fica evidente outras importantes características dessa estrutura: a sequencialização e a indexação de seus elementos e a sua dimensão (tamanho).

3 DEFINIÇÂO

Na linguagem C, o vetor e declarado da seguinte maneira:

```
1 <tipoDosDados> <nomeDoVetor>[<tamanhoDoVetor>];
```

Onde **tipoDosDados** representa o tipo dos dados dos elementos que são armazenados no vetor, **nomeDoVetor** o nome pelo qual o vetor sera referenciado e **tamanhoDoVetor** o número de elementos que podem ser armazenados no vetor. Veja um exemplo:

```
int idade[3];
char nome[10];
double velocidade[3];
```

Neste exemplo foi definido três vetores de nomes idade, nome e velocidade. Respectivamente, de tamanhos 3, 10 e 3, ou seja, armazenam 3, 10 e 3 elementos do tipo declarado.

3.1 DEFINIÇÂO EM TEMPO DE COMPILAÇÂO

A definição em tempo de compilação é caracterizada pelo número inteiro entre colchetes na declaração do vetor. Todas as declarações mostradas no exemplo acima são exemplos de vetores definidos em tempo de compilação, antes da execução do programa.

3.2 DEFINIÇÂO EM TEMPO DE EXECUÇÂO

A definição do tamanho em tempo de execução pode ser implementada substituindo o número inteiro que é colocado entre colchetes na declaração do vetor por uma variável que só ganha um valor durante a execução do programa. O exemplo abaixo mostra como isso pode ser feito na linguagem C:

```
# include <stdio .h>

int main (){
  int num ;

  printf (" Quantas notas deseja armazenar no vetor ?");
  scanf ("%d",&num);

  float notas[num];

  return 0;
}
```

Repare que declarando o vetor notas dessa forma, o desperdicio de memoria pode ser eliminado, já que o vetor é utilizado por completo. Isso pode ser feito desde que se tenha conhecimento previo do número de elementos que sao armazenados e que se possa informa-lo a aplicaçao.

4 OPERAÇÔES

Na linguagem C, não existem operações pré-definidas para a manipulação de um vetor como um todo. As operações só podem ser feitas para cada elemento do vetor, individualmente. Para acessar um elemento de um vetor, são necessários apenas seu nome e o índice que informa sua localização. Na linguagem C, a sintaxe de acesso a um elemento um vetor é dada por:

```
<nome_vetor>[<indice>]
```

no qual o índice pode ser tanto um número inteiro maior ou igual a zero quanto uma expressão inteira composta por variáveis e números.
Um elemento acessado por um índice pode ser manipulado como uma variavel qualquer e ser utilizado em diversas operações, como atribuição, operações aritméticas, etc. O exemplo a seguir mostra algumas dessas operações em Linguagem C.

```c
# include <stdio.h>

int main (){
  int i;
  int vet1[10];
  int vet2[10];
  float notas[8];
  float soma ;
  float media ;

  vet1 [0]=43;

  for ( i=0; i<5; i++) {
    vet1 [2*i+1] = 10;
  }

  for ( i=0; i<10; i++){
    vet2[i] = 0;
  }

  for ( i=0; i<8; i++) {
    scanf ("%f" ,&notas[i]);
  }

  for ( i=0; i<8; i++) {
    soma += notas[i];
  }

  media = soma/8;
  return 0;
}
```

Neste exemplo, primeiramente são declarados a variável i, do tipo int; os vetores vet1 e vet2, ambos do tipo int e de tamanho igual a 10; o vetor notas do tipo float e de tamanho igual a 8; e as variáveis soma e media, ambas do tipo float. Em seguida, o primeiro elemento do vetor vet1, ou seja, o elemento de índice 0 (zero), recebe o valor 43; todos os elementos de vet1 de índice ímpar recebem o valor 10; todos os elementos de vet2 recebem o valor 0; cada uma das 8 posições do vetor notas é preenchida por um valor lido do teclado; e, por último, é calculada a média dos valores contidos no vetor notas e armazenada na variável média .
É importante ressaltar a utilização de expressões como índice, pois ela é essencial para o acesso rápido a todos os elementos do vetor.

5 VETORES EM COMANDOS DE ENTRADA E SAIDA

Para exbir um elemento de um vetor utilizando o comando **printf**, basta acessar o elemento como no exemplo a seguir:

```c
int vet[3] = {1,2,3};
printf ("%d\n",vet[0]);
```

Vale frisar de que não é possível imprimir todo o vetor apenas colocando o nome do vetor, sendo necessário exibir elemento por elemento, o que pode ser feito por meio de um comando **for**.

Para possibilitar entrada de dados em um vetor também utiliza-se o acesso aos elementos por índice. Veja o exemplo:

```
int vet[3];
scanf("%d %d %d", &vet[0], &vet[1], &vet[2]);
```

Lembrando de adicionar o ’’ em cada elemento utilizado.

6 VETORES COMO PARÂMETROS DE UMA FUNÇÃO

Um tipo de dados delimita o conjunto de valores possíveis que uma determinada vDiferente das variáveis já aprendidas, quando um vetor é passado como parâmetro de uma função, seu valor alterado dentro da função também é alterado fora dela. Construindo uma função com um vetor como parâmetro de entrada:

```
void soma1 (int vet[], int tam){
  int i;
  for (i=0;i<tam;i++){
    vet[i]++;
  }
}
```

Observa-se que foi necessário ter como parâmetro de entrada o tamanho do vetor, pois, sem ele não saberíamos onde o vetor “para”. Observe que o parâmetro foi definido com o <tipo> e logo após o nome do vetor e colchetes vazio. Como exemplo de que o valor dos elementos do vetor são alterados fora da função, utilize esta função dentro de uma **main** chamando-a e, logo após, imprima todo o vetor. Viu como realmente é alterado os elementos do vetor na main?

7 ALGUNS CUIDADOS

É necessário algumas precauções ao se utilizar vetores em programação. A seguir algumas dicas na hora de se implementar vetores:

- É necessário atenção para não acessar indevidamente uma célula de um vetor;
- Cuidado na implementação para não extrapolar o tamanho fixo do vetor, acessando índices inexistentes (stack-based buffer overflow), ocasionando às vezes o que chamamos de “Segmentation Fault” ou “Falha de Segmentação”;
- Declaração de vetores com tamanhos muito grandes onde há um grande consumo de memória.

Praticando

Agora vamos praticar! Para todos os praticando em que é pedido que se escreva um programa, escreva o código do seu programa e nos envie o .c (código fonte) correspondente.

1. Faça um programa que receba dois vetores de inteiros, A e B, de mesmo tamanho, multiplique os elementos de mesmo índice de ambos os vetores e armazene em um terceiro vetor C. O programa deverá retornar o MAIOR elemento do vetor C.
   A entrada do programa é composta por vários conjuntos de teste. A primeira linha do programa deverá conter um inteiro N que representará o tamanho dos vetores A e B, e nas duas linhas seguintes, os valores inteiros que A e B armazenarão respectivamente. O programa deverá retornar o maior elemento do vetor C, que representa a multiplicação de A por B. O final da entrada é indicado quando N = 0.

Exemplo de Entrada	**Exemplo de Saida**
4 1 3 5 8 2 7 8 9 5 1 7 7 8 4 2 3 4 5 2 0	72 40

2. Faça um programa que receba um vetor de N inteiros e dois números i e j (i<=j), representando índices do vetor. Esta função deve retornar:

   - -2, caso os elementos entre i e j não estejam em ordem;
   - -1, caso os elementos entre i e j estejam em ordem decrescente;
   - 0, caso os elementos entre i e j sejam todos iguais;
   - 1, caso os elementos entre i e j estejam em ordem crescente;

   A entrada é composta por vários casos de teste. A primeira linha do programa deve contém um inteiro N que representa o tamanho do vetor e em seguida, os valores inteiros que o vetor armazenará. Na linha seguinte será informado o valor de i e j, respectivamente. O programa termina quando N = 0.

Exemplo de Entrada	**Exemplo de Saida**
6 1 1 2 2 3 3 0 5 6 1 1 2 2 3 -3 0 5 0	1 -2

3. O Carnaval é um feriado celebrado normalmente em fevereiro; em muitas cidades brasileiras, a principal atração são os desfiles de escolas de samba. As várias agremiações desfilam ao som de seus sambas-enredos e são julgadas pela liga das escolas de samba para determinar a campeã do Carnaval. Cada agremiação é avaliada em vários quesitos; em cada quesito, cada escola recebe N notas que variam de 5,0 a 10,0. A nota final da escola em um dado quesito é a soma das três notas centrais recebidas pela escola, excluindo a maior e a menor das cinco notas.
   Como existem muitas escolas de samba e muitos quesitos, o presidente da liga pediu que você escrevesse um programa que, dadas as notas da agremiação, calcula a sua nota final num dado quesito.

   A entrada é composta por vários conjuntos de teste. A primeira linha contém um inteiro N, que representa a quantidade de notas a serem lidas. A linha seguinte representa as N notas separados por espaço. A entrada termina quando N = 0.

   A saída do programa deverá ser a nota final que cada escola receberá. **DICA: Ordene**

Exemplo de Entrada	Exemplo de Saida
5 6.4 8.2 8.2 7.4 9.1 6 10.0 10.0 5.0 5.0 10.0 5.0 0	23.8 30.0

   o vetor.

Desafios

1. Pão a metro é um tipo de sanduíche gigante que é uma excelente opção de lanche para torneios de programação, embora a experiência já tenha mostrado que o oferecimento de sanduiches pode gerar reclamação dos competidores. Outro grande problema é que algumas pessoas são mais gulosas que outras e, dessa maneira, acabam pegando pedaços maiores que os pedaços dos outros. Para a nal da OBI, a coordenação estava pensando em providenciar pão a metro para os competidores, porém tais problemas os zeram recuar na idéia.

   Embora a idéia tenha sido momentaneamente abandonada, uma idéia simples surgiu: cortar previamente o pão em fatias de tamanho iguais e distribuí-las entre as pessoas. O único problema com tal idéia é que se o número de pessoas for muito grande, ca impraticável ter apenas um pão. Por exemplo, se quisermos que 1.000 pessoas recebam 20 centímetros de sanduíche, seria necessário um sanduíche de 20.000 centímetros, ou 200 metros!
   Alguém levantou a seguinte hipótese: se houvessem N pessoas e fossem encomendados K sanduíches de empresas diferentes, cada qual com uma determinada metragem (tamanho) Mi (1 <= i <= K), seria possível retirar desses pães N fatias de mesmo tamanho, possivelmente sobrando partes nao utilizadas. A questão seria: qual o tamanho inteiro

máximo que essas fatias poderão ter?

Por exemplo, se tivermos K = 4, com os tamanhos (em centímetros) M1 = 120, M2 = 89, M3 = 230 e M4 = 177 e N = 10, podemos retirar N fatias iguais de tamanho máximo 57, pois assim conseguimos 2 fatias no primeiro pão, 1 no segundo, 4 no terceiro e 3 no quarto, totalizando as 10 fatias necessárias. Se tentarmos cortar fatias de tamanho 58, só será possível obter 3 fatias do terceiro pão, totalizando 9 e, portanto, 57 é realmente o melhor que podemos obter. Note que não podemos usar duas ou mais fatias menores de diferentes pães para formarmos uma fatia do tamanho selecionado. (caria muito deselegante dar um lanche recortado...)

Escreva um programa que, dados os tamanhos de pão disponíveis (em centímetros) e a quantidade de pessoas a serem atendidas, retorne o tamanho inteiro máximo (em centímetros) da fatia que pode ser cortada de maneira a atender todas as pessoas.

A entrada contém vários conjuntos de testes. A primeira linha de cada entrada contém um inteiro N que indica a quantidade de pessoas (1 <= N <= 10.000). A segunda linha contém um inteiro K (1 <= K <= 10.000) que é a quantidade de sanduíches disponível. Na terceira linha há K inteiros M (1 <= M <= 10.000) separados por um espaço em branco representando o tamanho de cada pão. A entrada termina quando N = K = 0.

Seu programa deve imprimir, na saída padrão, uma única linha, contendo o tamanho inteiro máximo da fatia que pode ser cortada.

Exemplo de Entrada	Exemplo de Saida
10 4 120 89 230 177 3 2 45 85 7 7 100 98 99 505 102 97 101 0 0	57 42 101