Vetores

Discente: Valter Nascimento Felizardo Neto
Matricula : 31321ECA019
Disciplina : Programação Aplicada para EAC

Vetor em Algoritmos

- Definição de Vetor: Segundo o livro “Fundamentos da programação de computadores” de auditoria de Ana Fernanda Gomes Ascencio e Edilene Aparecida Veneruchi De Campo é definido como: “Vetor também é conhecido como variável composta homogênea unidimensional. Isto quer dizer que se trata de um conjunto de variáveis de mesmo tipo, que possuem o mesmo identificador (nome) e são alocadas sequencialmente na memória. Como as variáveis têm o mesmo nome, o que as distingue é um índice que referencia a sua localização dentro da estrutura”.

Vetor em C

- Os elementos são indexados de 0 até n-1, onde n é a quantidade de elementos do vetor. O valor de n também é chamado de dimensão ou tamanho do vetor. O vetor tem tamanho fixo durante a execução do programa, definido na declaração. Durante a execução não é possível aumentar ou diminuir o tamanho do vetor. Note que a numeração começa em zero, e não em um. Essa é uma fonte comum de erros. A Figura 1 ilustra um vetor com 10 elementos, denominados v0, v1, … v9, todos eles de tipo int.

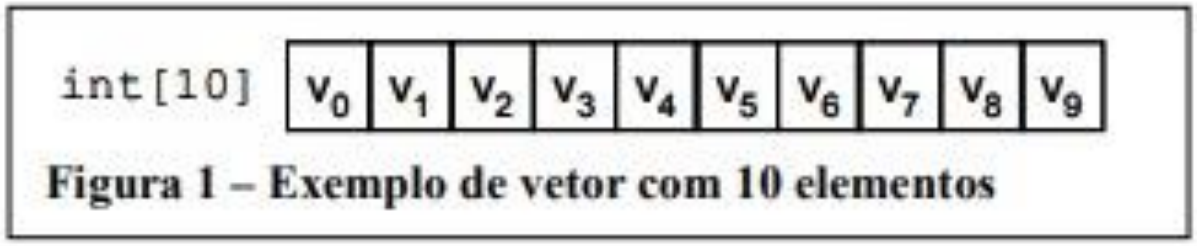

Figura 1 – Exemplo de vetor com 10 elementos

Declaração de Vetores

- A declaração de vetores obedece à mesma sintaxe da declaração de variáveis. A diferença está no valor entre colchetes, que determina quantos elementos ele armazenará, ou seja, em outras palavras, determina o seu tamanho ou dimensão.

- Por exemplo, para declarar um vetor com 10 números inteiros:

int vetor[10];

- O tamanho precisa ser necessariamente um número inteiro e constante. Ele não pode ser resultado de uma expressão:
  - *int tamanho = 10; int vetor[tamanho*2]; // ERRADO!*

Acesso ao conteúdo de Vetores

- Com os vetores, a expressão de referência de memória é o operador de índice [ ]. Ele utiliza uma referência de memória (normalmente uma variável do tipo vetor) e um número inteiro (o índice). Ele retorna uma referência para o elemento correspondente ao índice. O tipo do valor retornado é o mesmo tipo da declaração do vetor. Por exemplo, para atribuir o valor 3 na primeira posição do vetor, escrevemos:
  - *vetor[0] = 3;*
- Note que o índice zero indica a primeira posição no vetor. A expressão vetor[0] referencia a posição de memória correspondente ao elemento de índice zero no vetor. Para somar os primeiros três elementos e armazenar o valor calculado no quarto elemento, escrevemos:
  - *vetor[3] = vetor[0] + vetor[1] + vetor[2];*

Acesso ao conteúdo de Vetores

- Em expressões, uma referência indexada a um vetor pode ser usada da mesma forma e nas mesmas posições em que usaríamos variáveis convencionais de mesmo tipo. Tudo se passa como se tivéssemos várias variáveis declaradas simultaneamente, todas de mesmo tipo, e com “nomes” vetor[0], vetor[1], e assim por diante. É muito comum utilizar a estrutura de repetição for para percorrer todos os elementos de um vetor. Por exemplo, para imprimir todos os elementos de um vetor de 100 elementos:
- *int indice;*
- *int vetor[100];*
- *...*
- *for (indice = 0; indice < 100; indice++) { printf("%d", vetor[indice]); }*

Exemplo

- Um programa que lê dez números e os imprime em ordem inversa. Para isso, é necessário armazenar os 10 números para poder imprimi-los de trás para frente. Seria possível utilizar 10 variáveis distintas, mas a solução com vetor é bem mais elegante. Mais ainda, se fossem 10.000 números, e não 10, ficaria impraticável usar variáveis distintas.

```c
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[]) {

	int valores[10];
	int indice;

	printf("Escreva 10 números inteiros: ");
	for (indice = 0; indice < 10; indice++) {
		scanf("%d", &valores[indice] );
	}

	printf("Valores em ordem reversa:\n");
	for (indice = 9; indice >= 0; indice--) {
		printf("%d ", valores[indice]);
	}

	return 0;
}
```

Conteúdo inicial de vetores

- Na declaração, pode-se definir o valor inicial de cada elemento de um vetor. A sintaxe é semelhante à declaração de uma variável comum com valor inicial, mas os elementos são listados entre as chaves { e } e separados por vírgula.

- *tipo variável[n] = { elem0, elem1, elem2, elem3, ... elemn-1 };*

- Como o número de elementos do vetor pode ser inferido a partir da lista entre chaves, podemos omitir o tamanho do vetor:

- *tipo variável[] = { elem0, elem1, elem2, elem3, ... elemn-1 };*

Regras para acesso ao vetor

- O programador deve observar algumas restrições na manipulação de variáveis que representam vetores.
- **Índices inválidos**: Os elementos são numerados sempre de 0 até tamanho-1. Caso o programa tente acessar erroneamente um elemento de índice negativo ou de índice além do tamanho do vetor, as consequências poderão ser imprevisíveis. No melhor dos casos, o sistema operacional detectará essa anomalia e o programa será finalizado sinalizando um erro de execução.

- **Atribuir o valor de todos os elementos de uma só vez:**
- Exceto na declaração do vetor, não é possível atribuir valores a todos os elementos em uma só linha. Cada elemento precisa ser acessado individualmente. Tampouco é possível usar um único scanf para ler todo o conteúdo do vetor. O código abaixo está, portanto, errado:
- *int vetor[10]; //* inicializar todos os elementos com valor 0
  vetor = 0; // ERRADO!
- O correto é utilizar uma estrutura de repetição for para atribuir o valor a cada elemento.
- *int vetor[10];*
- *int indice; //* inicializar todos os elementos com o valor 0
- *for (indice = 0; indice < 10; indice++) {*
- *vetor[indice] = 0;*
- *}*

- **Copiar vetores:**
- Tampouco é possível copiar o conteúdo de um vetor para um outro, mesmo que os dois sejam de mesmo tamanho e os elementos sejam de mesmo tipo.
- *int vetorA[10], vetorB[10];*
- // copiar o conteúdo do vetor B para o vetor A
- *vetorA = vetorB;* // ERRADO!
- O correto é utilizar uma estrutura de repetição for para copiar um elemento de cada vez.
- *int vetorA[10], vetorB[10];*
- *int indice;* // copiar o conteúdo do vetor B para o vetor A for *(indice = 0; indice < 10; indice++) { vetorA[indice] = vetorB[indice]; }*

Vetor de tamanho variável

Como proceder quando o tamanho do vetor não pode ser previsto até o momento da execução?

Considere um programa que lê n valores e os armazena no vetor. O valor n é informado pelo usuário durante a execução do programa. A linguagem C não oferece recursos para se declarar um vetor cujo tamanho se ajuste automaticamente ao número n de elementos. A solução mais simples é declarar o vetor com o tamanho máximo necessário para tratar o pior caso. O número de elementos realmente utilizado ficará armazenado em uma outra variável.

Exemplo

- Programa que lê n valores, armazena-os em um vetor e em seguida os imprime em ordem inversa.

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[]) {

     int valores[100];
int numero_valores;
int indice;

printf("Quantos valores? (no máximo 100): ");
scanf("%d", &numero_valores);
if ( (numero_valores > 100) || (numero_valores < 0) ) {
     printf("Número de valores inválido\n");
     return 1;
}

printf("Escreva %d números inteiros: ", numero_valores);
for (indice = 0; indice < numero_valores; indice++) {
     scanf("%d", &valores[indice] );
}

printf("Valores em ordem reversa:\n");
for (indice = numero_valores-1; indice >= 0; indice--) {
     printf("%d ", valores[indice]);
}

return 0;
}
```

Exercício 1

Programa que lê n valores, armazena-os em um vetor e em seguida os imprime em ordem inversa.

```c
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char *argv[]) {
int valores[10];
//A variável valores é declarada como um vetor de 10 elementos de tipo inteiro.Ele armazenará os valores digitados pelo usuário.
int numero_valores;
//A variável numero_valores armazena quantos elementos do vetor valores estão realmente sendo utilizados para armazenar os valores digitados pelo usuário.
int indice;
//A declaração cria uma variável contadora para o for de leitura e o for de escrita de valores
 printf("Quantos valores? (no maximo 10): ");
 scanf("%d", &numero_valores);
//Pede ao usuário para informar de quantos elementos consiste a lista de números. Esse valor será armazenado na variável numero_valores e passará a controlar o número de repetições do for.
if ( (numero_valores > 10) || (numero_valores < 0) ) {
 printf("Número de valores inválido\n");
 return 1;
//Verifica se o número de elementos que o usuário digitou está dentro do limite suportado pelo programa. Esta verificação é essencial para garantir que nenhuma operação de indexação do vetor seja realizada com índice maior que a capacidade do vetor, ou com índice negativo, o que poderia comprometer a integridade do programa.
 }
 printf("Escreva %d numeros inteiros: ", numero_valores);
 for (indice = 0; indice < numero_valores; indice++) {
 scanf("%d", &valores[indice] );
//A estrutura de repetição for executa numero_valores vezes, variando o conteúdo de indice de 0 até numero_valores-1. A cada repetição, o comando scanf lê um número inteiro e o armazena no indice-ésimo elemento do vetor valores. Note que tal como no scanf para variáveis comuns, o nome do vetor é precedido pelo símbolo &.
 }
 printf("Valores em ordem reversa:\n");
 for (indice = numero_valores-1; indice >= 0; indice--) {
 printf("%d ", valores[indice]);
//O vetor contém numero_valores números lidos no for anterior. Agora, vamos usar novamente a estrutura de repetição for para imprimir o vetor de trás para frente. Por este motivo, fazemos o índice variar de numero_valores-1 para 0.
 }
 return 0;
}
```

Exercício 2

Faça um programa que preencha um vetor com nove números inteiros, calcule e mostre os números primos e suas respectivas posições.

```c
//Exercicio 1: Faça um programa que preencha um vetor com nove números inteiros, calcule e mostre os números primos e suas respectivas posições.
#include <stdio.h>
#include <stdlib.h>

int main()
{
    int num[10];//A variável num é declarada como um vetor de 10 elementos de tipo inteiro.Ele armazenará os valores digitados pelo usuário.
    int i,j,cont,resto;//As variaveis i,j,cont e resto são declaradas ara realizar as operação adjacentes.
    i=0;// É zerado a variavel por precausão, para evitar lixo de memoria.
    j=0;// É zerado a variavel por precausão, para evitar lixo de memoria.
    resto=0;// É zerado a variavel por precausão, para evitar lixo de memoria.
    for(i=1;i<=9;i++){
        printf("\n Preencha o vetor de posicao %i:",i);//Pede ao usuário para informar os elementos que vão completar o vetor.Esse valor será armazenado na variável num[i].
        scanf("%i",&num[i]);//recebe o valor e salva.
    }
    for(i=1;i<=9;i++){ //como no enunciado é dito que o numero de de valores inteiros seria nove, o for já esta programado para repetir 9 vezes.
        cont=0;// É zerado a variavel por precausão, para evitar lixo de memoria.
        for(j=1;j<=num[i];j++){//numero de repetição da variavel J para poder mostrar os numeros primos.
            resto=(num[i]%j);//operação que salva o resto da divisao na variavel "resto".
            if (resto==0){//Faz a função de comparar, dizer se é verdade, no caso se o resto for igual a 1 ira executar o comando abaixo.
                cont=cont+1;// se o resto for igual a 1 vai ser soma 1 (+1) na variavel cont.
            }
        }
        if (cont <= 2){//Se cont for menor ou igual a 2 o if vai executar os comandos a seguir.
            printf("\n %i e primo",num[i]);// Mostra o numero primo.
            printf("\n se encontra na posicao %i ",i);// Mostra a posição no vetor do numero primo
        }

    }
    return 0;//encerra o programa.
}
```