

Linguagem C Vetores

Prof. Edwar Saliba Júnior
Fevereiro de 2011


Vetores

- Sintaxe:

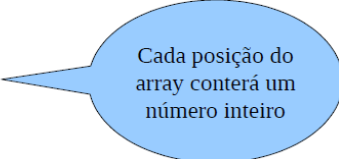
```
tipo_variável nome_vetor[tamanho];
```

- Um vetor é uma variável que possui várias ocorrências de um mesmo tipo. Cada ocorrência é acessada através de um índice;
- Os vetores também são chamados de *arrays* ou matrizes unidimensionais por possuírem somente um índice;
- Para definir um vetor em C deve-se indicar a quantidade de ocorrência que este terá, colocando na sua definição o valor entre [];
- Os índices de um vetor em C, sempre começarão em zero. Portanto, para se acessar a primeira ocorrência de um vetor deve-se indicar o índice zero.

Vetores

- Ocupam posições contíguas na memória;
- O índice varia obrigatoriamente de 0 a N-1, onde N é o tamanho do vetor;
- Os elementos podem ser de qualquer tipo, mas são sempre do mesmo tipo;
- Exemplo: `int notas[4];` 

--	--	--	--



Cada posição do array conterá um número inteiro

 - O vetor tem 4 elementos do tipo inteiro;
 - Os elementos estão nas posições 0, 1, 2 e 3;
 - Os elementos são referenciados como `notas[0]`, `notas[1]`, `notas[2]` e `notas[3]`;
- Cada posição deve ser tratada exatamente como uma variável do tipo do vetor.

Exemplos de Declarações de Vetores

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main()
5  {
6      float salario[100]; // Armazena o salário de 100 pessoas.
7      char palavra[20]; // Armazena uma palavra de 20 caracteres.
8      int primos[100]; // Armazena os 100 primeiros primos.
9      double altura[1000]; // Armazena a altura de 1000 pessoas.
10     .
11     .
12     .
13
14     return 0;
15 }
16
```

Atribuição de Valores ao Vetor

- Exemplo: `int notas[4];`
 - O vetor tem 4 elementos;
 - Os elementos estão nas posições 0, 1, 2 e 3;
 - Os elementos são referenciados como: `notas[0]`, `notas[1]`, `notas[2]` e `notas[3]`;
- Atribuição de valores aos elementos do vetor:
 - `notas[0] = 5;`
 - `notas[1] = 2;`
 - `notas[2] = 6;`
 - `notas[3] = 10;`

notas	5	2	6	10
	0	1	2	3

Inicialização de Vetores

- `int a[4]={1,2,3};`
 - Cria um vetor de quatro inteiros e inicializa `a[0]=1`, `a[1]=2`, e `a[2]= 3`; `a[3]` fica valendo zero;
- `int a[]={1,2,3};`
 - Cria um vetor de apenas três elementos, com inicialização equivalente ao anterior;
- `char a[3]={'b','k','n'};`
 - Cria um vetor de três caracteres e inicializa `a[0]='b'`, `a[1]='k'`, e `a[2]='n'`;
- `int v[10];`
 - Cria um vetor de dez posições inteiras, não inicializado.

Definição de Constantes

- Constantes são valores fixos com nomes definidos;
- Há 2 formas de especificar constantes em C:

- Diretiva de pré-processamento `#define`:

```
#define identificador valor
```

- Com o uso da palavra reservada `const`:

```
const tipo identificador = valor;
```

- Exemplos:

```
const int num = 10;
```

```
#define NUM 10
```

Diferença entre: `const` e `#define`

- Constante definida com `const`:
 - É um valor que está na memória do computador, mas não pode ser alterado;
 - O tipo da constante é definido na declaração;
 - `const`: palavra reservada da linguagem C;
- Constante definida com `#define`:
 - Não existe na memória;
 - É diretiva de pré-processamento, ou seja, o compilador substitui todas as suas utilizações no código-fonte, por seu valor;
 - São conhecidas como constantes simbólicas;
 - Identificadores de constantes simbólicas, geralmente são escritos com letras maiúsculas.



Exemplo da Utilização de Vetores

- Dado o seguinte problema:
 - Deseja-se calcular a média aritmética das notas de uma classe, em uma avaliação que foi aplicada para uma determinada disciplina. Esta classe é composta por 40 alunos. Faça um programa para facilitar este cálculo.

Resolução do Problema Anterior Sem Utilizar Vetor

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main()
5  {
6      float media, notaAluno01, notaAluno02, notaAluno03, ..., notaAluno40;
7
8      printf("\nDigite a nota do 1o. aluno: ");
9      scanf("%f",&notaAluno01);
10     printf("\nDigite a nota do 2o. aluno: ");
11     scanf("%f",&notaAluno02);
12     printf("\nDigite a nota do 3o. aluno: ");
13     scanf("%f",&notaAluno03);
14     .
15     .
16     .
17     printf("\nDigite a nota do 40o. aluno: ");
18     scanf("%f",&notaAluno40);
19
20     media = (notaAluno01 + notaAluno02 + notaAluno03 + ... + notaAluno40) / 40;
21
22     printf("A média da turma é: %f",media);
23
24     return 0;
25 }
```

- Criação de 40 variáveis, um para cada aluno. Absurdo! E se fossem 1000 alunos?

Resolvendo o Problema Anterior Utilizando Vetor

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main()
5  {
6      float media, notaAluno[40];
7      int i;
8
9      for(i = 0; i < 40; i++){
10         printf("\nDigite a nota do %io. aluno: ",i);
11         scanf("%f",&notaAluno[i]);
12     }
13
14     for(i = 0; i < 40; i++)
15         media += notaAluno[i];
16
17     media /= 40;
18
19     printf("A média da turma é: %f",media);
20
21     return 0;
22 }
23
```

- Agora sim, está muito melhor!

Resolvendo o Problema Anterior Utilizando Vetor e Constantes

```
1  #include <stdio.h>
2
3  int main()
4  {
5      const int Qtdade_Alunos = 40;
6      float media, notaAluno[Qtdade_Alunos];
7      int i;
8
9      for(i = 0; i < Qtdade_Alunos; i++){
10         printf("\nDigite a nota do %io. aluno: ",i);
11         scanf("%f",&notaAluno[i]);
12     }
13
14     for(i = 0; i < Qtdade_Alunos; i++)
15         media += notaAluno[i];
16
17     media /= Qtdade_Alunos;
18
19     printf("A média da turma é: %f",media);
20
21     return 0;
22 }
23
```

```
1  #include <stdio.h>
2  #define QTDADE_ALUNOS 40
3
4  int main()
5  {
6      float media, notaAluno[QTDADE_ALUNOS];
7      int i;
8
9      for(i = 0; i < QTDADE_ALUNOS; i++){
10         printf("\nDigite a nota do %io. aluno: ",i);
11         scanf("%f",&notaAluno[i]);
12     }
13
14     for(i = 0; i < QTDADE_ALUNOS; i++)
15         media += notaAluno[i];
16
17     media /= QTDADE_ALUNOS;
18
19     printf("A média da turma é: %f",media);
20
21     return 0;
22 }
23
```

- Agora muito melhor! Pois, se eu precisar aumentar ou diminuir a quantidade de alunos, basta que eu faça a alteração em um único lugar, ou seja, na constante.

Passando Vetor como Parâmetro

- Ao passarmos um vetor como parâmetro, na verdade está sendo passado o endereço da variável;
- Deve ser passado também, o número de elementos do vetor.

```
1  #include <stdio.h>
2  #define N 5
3
4  int maximo(int array[], int n){
5      int max=array[0];
6      int i;
7
8      for (i=1; i<n;i++)
9          if (array[i]> max)
10             max = array[i];
11
12     return max;
13 }
14
15 int main() {
16     int valores[N], i, max;
17
18     for(i = 0; i < N; ++i){
19         printf("\nDigite %do. número: ", i);
20         scanf("%d", &valores[i]);
21     }
22
23     max = maximo(valores, N);
24     printf("\nValor Maximo: %d\n", max);
25
26     return 0;
27 }
```



Exercícios

- Faça um programa que receba o nome de cinco produtos e seus respectivos preços, calcule e mostre:
 - a quantidade de produtos com preço inferior a R\$50,00;
 - o nome dos produtos com preço entre R\$50,00 e R\$100,00;
 - a média dos preços dos produtos com preço superior a R\$100,00.

Exercícios

- Faça um programa onde o usuário do *software* preencha dois vetores (X e Y) de 10 posições cada, com números inteiros. Calcule e mostre os seguintes resultados:
 - A união de X e Y (todos elementos de X e de Y sem repetições);
 - A diferença de X e Y (todos os elementos de X que não existam em Y, sem repetições);
 - A soma entre X e Y (a soma de cada elemento de X com o elemento de mesma posição em Y).

Bibliografia

- LAUREANO, Marcos. **Programação em C para ambiente Linux**. Disponível em: <<http://br-c.org/doku.php>>. Acesso em: 06 fev. 2011.
- MURTA, Cristina Duarte. *Slides da disciplina de Programação de Computadores I*. CEFET-MG, 2010.
- SENNE, Edson Luiz França. **Primeiro Curso de Programação em C**. 2. ed. Florianópolis: Visual Books, 2006.