Linguagem de Programação C

Vetores

Cristiano Lehrer

http://www.ybadoo.com.br/

Declaração de Vetores (1/5)

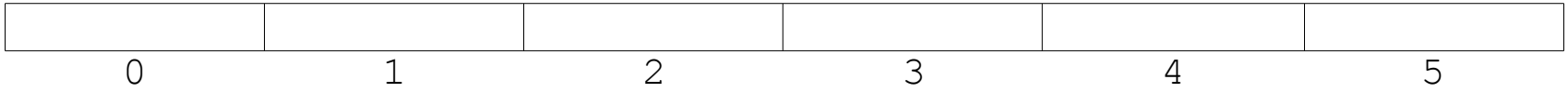
- Um vetor pode conter elementos de qualquer tipo de dados.
- No entanto, os elementos do vetor são todos do mesmo tipo, o qual é definido na declaração do mesmo:
  - `tipo nome[número de elementos]`
    - `tipo` – corresponde ao tipo de dados de cada um dos elementos do vetor.
    - `nome` – indica o nome pelo qual esse vetor vai ser conhecido.
    - `número de elementos` – valor constante que indica quantos elementos tem o vetor.

Declaração de Vetores (2/5)

- Declaração de um vetor de inteiros com 20 elementos:
  - `int g[20];`
- Declaração de um vetor de reais com 100 elementos:
  - `float renda[100];`
    - `float` – tipo de cada um dos elementos do vetor;
    - `100` – número de elementos do vetor;
    - `renda` – nome do vetor.

Declaração de Vetores (3/5)

- Em C, os índices de um vetor com **n** elementos variam sempre entre **0** e **n - 1**.
- O índice do primeiro elemento, de qualquer vetor em C, é sempre **0** (ZERO);
- Considere a declaração do seguinte vetor:
  - `int vetor[6];`



Declaração de Vetores (4/5)

- Os elementos de um vetor podem ser acessados individualmente, informando a posição desejada dentro de colchetes.
- Considere a declaração do seguinte vetor:

```
int vetor[6];

vetor[0] = 123;

vetor[5] = vetor[0] * 2;

vetor[2] = vetor[0] + vetor[5];
```

123		369			246
vetor[0]	vetor[1]	vetor[2]	vetor[3]	vetor[4]	vetor[5]

Declaração de Vetores (5/5)

- Desenvolver um programa em C que inicialize um vetor de seis elementos, colocando em cada posição do mesmo o índice dessa mesma posição.
  - A posição ocupada por um elemento de um vetor é chamada de índice desse elemento no vetor.

```
int vetor[6], i;

for(i = 0; i < 6; i++)

  vetor[i] = i;
```

0	1	2	3	4	5
vetor[0]	vetor[1]	vetor[2]	vetor[3]	vetor[4]	vetor[5]

Carga Inicial Automática de Vetores (1/2)

- Tal como as variáveis, os vetores quando são criados contêm valores aleatórios (LIXO) em cada uma das suas posições.

```
tipo var[n] = {valor1, valor2, ..., valorn};
```

- Se um vetor for declarado com **n** elementos e forem colocados **k** valores (**k < n**) na carga inicial do vetor, então os primeiros **k** elementos de vetor serão iniciados com os respectivos valores e os restantes serão iniciados com o valor ZERO.

```
int v[10] = {10, 20, 30};
```

10	20	30	0	0	0	0	0	0	0
v[0]	v[1]	v[2]	v[3]	v[4]	v[5]	v[6]	v[7]	v[8]	v[9]

Carga Inicial Automática de Vetores (2/2)

```
char vogal[5] = {'a', 'e', 'i', 'o', 'u'};
```

- A forma acima evita o código abaixo:

```
char vogal[5];
vogal[0] = 'a';
vogal[1] = 'e';
vogal[2] = 'i';
vogal[3] = 'o';
vogal[4] = 'u';
```

Observações sobre Vetores (1/3)

- Os elementos de um vetor são sempre armazenados em posições contíguas de memória.
- Os elementos de um vetor declarado sem qualquer carga inicial contêm valores aleatórios.
- O índice do primeiro elemento de um vetor é sempre `0` (ZERO).
- Os índices de um vetor com `n` elementos variam sempre entre `0` e `n - 1`.

Observações sobre Vetores (2/3)

- O valor existente em uma posição do vetor **`v`** pode ser obtido através do índice em que essa posição está armazenada **`v[índice]`**.
- O compilador não verifica se os índices utilizados em um vetor estão ou não corretos.
  - Um exemplo comum de erro de manipulação de um vetor de **`n`** elementos é a utilização do índice **`n`** (**`v[n]`**), que não pertence ao vetor e pode originar problemas graves, pois estaríamos alterando memória que não nos pertence.

Observações sobre Vetores (3/3)

- Um vetor pode ser automaticamente iniciado com o conjunto de valores existente dentro de chaves, após o sinal de atribuição (=).
  - Isso somente pode ser feito na declaração do vetor.
- Se o número de cargas iniciais for menor do que o número de elementos do vetor, os elementos em falta são iniciados com o valor ZERO.
- Pode-se declarar um vetor sem indicar qual o número de elementos que ele irá conter, desde que estes sejam colocados na sua carga inicial.
  - Nesse caso, o compilador calcula, automaticamente, o número de elementos que o vetor irá conter.
- Não se pode declarar vetores sem dimensão.
  - Se não sabemos qual a dimensão pretendida, como poderá o compilador saber qual o espaço necessário?

Passagem de Vetores para Funções (1/5)

- Suponha que tenhamos dois vetores:

```
int v[10];

int x[20];
```

- E queremos uma função para inicializar eles com zero:

```
void init1(int s[10]){
  int i;
  for(i = 0; i < 10; i++)
    s[i] = 0;
}
```

```
void init2(int s[20]){
  int i;
  for(i = 0; i < 20; i++)
    s[i] = 0;
}
```

Passagem de Vetores para Funções (2/5)

- E a função `main` ficaria da seguinte forma:

```
main(){
   int v[10];
   int x[20];

   init1(v);
   init2(x);
}
```

Passagem de Vetores para Funções (3/5)

- Em C, dentro de uma função não é possível saber com quantos elementos foi declarado um vetor que foi passado como argumento para essa função:

```
void init(int s[])
{
  int i;
  for(i = 0; i < ??; i++)
    s[i] = 0;
}
```

```
main()
{
   int v[10];

   init(v);
}
```

Passagem de Vetores para Funções (4/5)

- Para se resolver esse problema, o número de elementos do vetor deve ser passado junto para a função:

```
void init(int s[], int n)
{
  int i;
  for(i = 0; i < n; i++)
    s[i] = 0;
}
```

```
main()
{
    int v[10];
    int x[20];
    init(v, 10);
    init(x, 20);
}
```

Passagem de Vetores para Funções (5/5)

- Se for indicada qual a dimensão do vetor no parâmetro da função, o número de elementos do vetor é simplesmente ignorado pelo compilador.
- A função apenas se interessa em saber qual o tipo dos elementos do vetor.
- A dimensão a considerar é de responsabilidade exclusiva do programador.