Aulas 13: Variáveis compostas - vetores

Joice Otsuka

*Com base no livro Linguagem C: completa e descomplicada, de André Backes

Vetores

Até agora..

- Variáveis simples
- Armazenam um único valor de cada vez
- Se o valor da variável é atualizado o valor anterior é perdido

```
#include <stdio.h>

int main(){
    int x;
    x = 10;
    printf("x = %d\n",x);
    x = 5000;
    printf("x = %d\n",x);
    return 0;
}
```

Saída

```
x = 10
x = 5000
```

Problema

- Imagine o seguinte problema:
  - Leia as notas de uma turma de 100 estudantes e depois imprima as notas que são maiores do que a média da turma.
  - Poderia ser solucionado usando variáveis simples?

Problema

- Sim, mas para 100 alunos, precisaríamos de:
  - Uma variável para armazenar a nota de cada aluno
    - **100 variáveis**
  - Um comando de leitura para cada nota
    - **100 scanf()**
  - Um somatório de **100 notas**
  - Um comando de teste para cada aluno
    - **100 comandos if.**
  - Um comando de impressão na tela para cada aluno
    - **100 printf()**

Vetor

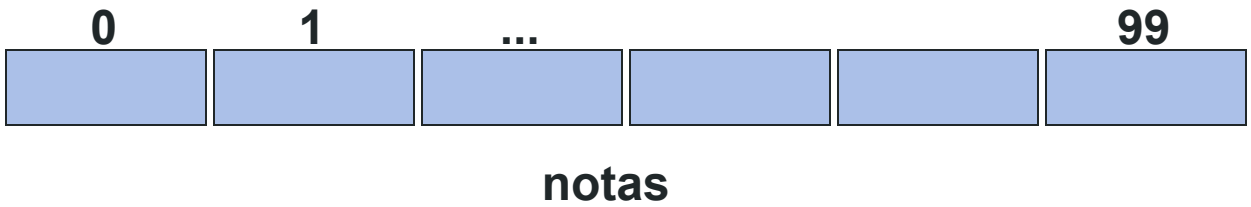
- Vetor ou *array* de uma dimensão é a forma mais familiar de dados estruturados.

- Permite a criação de uma variável composta (armazena um conjunto de valores) utilizando apenas um nome
  - homogêneas (mesmo tipo de dados)

- Sequência de elementos do mesmo tipo, onde cada elemento é identificado por um **índice**
  - **Índice sequencial de 0 ao (tamanho do vetor - 1)**
  - **Índice determina a posição de cada elemento**
  - **Acesso a cada elemento é realizado por meio do seu índice**

Vetor

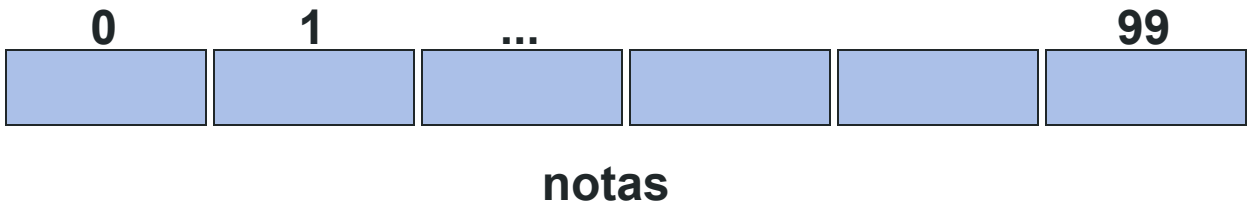
- Quando os valores têm relação entre si
  - Ex: conjunto de notas de alunos
- Podemos declarar uma única variável composta para armazenar as notas de 100 alunos
  - notas: conjunto de 100 valores acessados por meio de um índice



Declaração de vetor

- Arrays são agrupamentos de dados adjacentes na memória.
- Declaração:
  - **tipo_dado nome_vetor[N];**

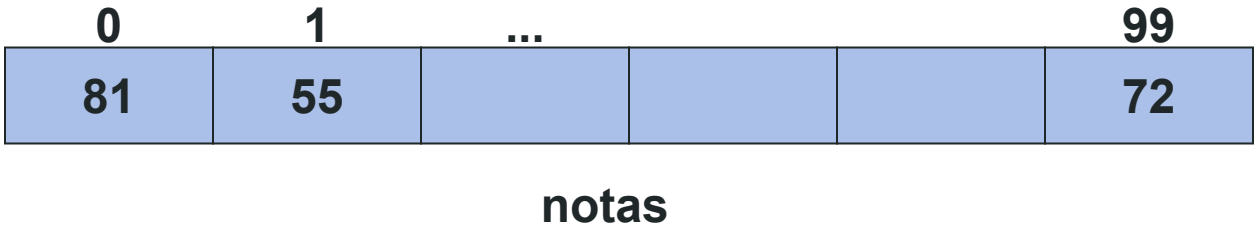
- O comando acima define um vetor denominado **nome_vetor,** capaz de armazenar **N elementos** adjacentes na memória do **tipo tipo_dado**
  - Ex: **int notas[100];**



Acesso

- Em um vetor, os elementos são acessados especificando o índice desejado entre colchetes [ ]
    - Ex: notas[0] // acessa o primeiro elemento do vetor
- A numeração do índice começa sempre do zero
- Isto significa que um vetor de 100 elementos terá índices de 0 a 99:
    - notas[0], notas[1], notas[2], ..., notas[99]

```
int notas[100];
notas[0] = 81;
notas[1] = 55;
...
notas[99] = 72;
```



Inicialização

- Um vetor pode ser inicializado na declaração por meio da atribuição de uma lista de valores separados por vírgula.
- Os valores devem ser dados na ordem em que serão colocados no vetor e devem ter o tipo declarado

```c
#include <stdio.h>

int main ()
{
  int i;
  float v[4] = { 1.5, 2.3, 3.5, 6 };
  for (i = 0; i < 4; i++)
    printf ("%.1f\n", v[i]);
  return 0;
}
```

Inicialização

- Não é necessário especificar todos os valores na inicialização, as demais posições receberão 0

```c
#include <stdio.h>

int main ()
{
  int v[5]={0};
  int i;
  for (i=0;i<5;i++){
      printf("%d\n",v[i]);
  }
```

Todas as posições são inicializadas com 0

Inicialização

```c
#include <stdio.h>

int main ()
{
  int v[]={1,2,3,4,5};
  int i;
  for (i=0;i<5;i++){
      printf("%d\n",v[i]);
  }
  return 0;
}
```

Neste caso o vetor é criado com o tamanho necessário para armazenar a lista de elementos dada na inicialização.

Vetor - Acesso

- **Atenção**
  - O que ocorre se um usuário digitar mais de 100 elementos para um array declarado com 100 elementos?
    - O programa tentará ler e armazenar todos os elementos digitados.
    - Porém, o programa os armazenará em uma parte não reservada de memória!
    - Isto pode resultar nos mais variados erros durante a execução do programa!!

O que ocorre neste caso?

```c
#include <stdio.h>

int main ()
{
  int i;
  int v1[4] = { 1, 2, 3, 4, 5 };

  printf ("valores de v1:\n");
  for (i = 0; i < 5; i++)
    printf ("%d\n", v1[i]);
  return 0;
}
```

O que ocorre neste caso?

```c
#include <stdio.h>

int main ()
{
  int i;
  int v1[4] = { 1, 2, 3, 4 };
  int v2[4] = { 5, 6, 7, 8 };
  v1[4] = 1;  // atribuição de valor a uma posição não reservada
  printf ("valores de v1:\n");
  for (i = 0; i < 4; i++)
    printf ("%d\n", v1[i]);
  printf ("valores de v2:\n");
  for (i = 0; i < 4; i++)
    printf ("%d\n", v2[i]);
  return 0;
}
```

Manipulação de vetor

- **Cada elemento do vetor** tem todas as características de uma variável e pode aparecer em expressões e atribuições (respeitando os seus tipos)

```
int v1[3] = { 1, 2, 3 };
v1[2] = v1[0] + 1;
```

Copiando um vetor

- Não é possível a atribuição de um vetor, apenas de cada um de seus elementos individualmente.

```c
#include <stdio.h>

int main(){
  int v[5] = {1,2,3,4,5};
  int v1[5];

  v1 = v; //ERRADO!

  int i;
  for(i=0; i<5; i++)
    v1[i] = v[i]; //CORRETO

  return 0;
}
```

Problema

- Voltando ao problema anterior
  - Leia as notas de uma turma de cem estudantes e depois imprima as notas que são maiores do que a média da turma.
  - Como seria uma solução usando vetor?

Vetor- Resumo

- Estrutura homogênea.
  - formado por elementos do mesmo tipo.

- Acesso por meio de índices.
  - Cada elemento do vetor tem um índice próprio, segundo sua posição no conjunto.
  - Todos os elementos da estrutura são igualmente acessíveis.
  - O tempo e o procedimento para acessar qualquer um dos elementos do array são iguais.

Exercícios

- Exercícios do AVA
- Lista 6 do Beecrowd

Referências

- BACKES, André. **Linguagem C:** completa e descomplicada. Rio de Janeiro: Elsevier, 2013. 371 p. ISBN 978-85-352-6855-3.Disponível na Biblioteca.