Vetores

emanoelim@utfpr.edu.br

Tipos de dados simples

- Os tipos de dados estudados até agora, são chamados de **tipos de dados simples**, ou primitivos:
  - char;
  - int;
  - float;
  - double;
  - ...

Tipos de dados simples

- Uma variável de um tipo de dado simples guarda um único valor;
- Isso não é muito conveniente quando precisamos guardar conjuntos de dados, por exemplo:
  - Guardar as notas de 40 alunos;
  - Guardar os valores de 100 produtos;
  - Guardar as idades de 1000 participantes de uma pesquisa.

Tipos de dados estruturados

- Estes tipos de problemas podem ser resolvidos usando **tipos de dados estruturados.**
- Permitem associar mais de um valor a uma variável.
- Dados estruturados podem ser de dois tipos:
  - Homogêneos: todos os valores são do mesmo tipo;
  - Heterogêneos: podem guardar valores de diferentes tipos;

Vetores

- Os **vetores** são tipos de dados estruturados **homogêneos**;
- Um vetor é uma variável que pode armazenar múltiplos valores;
- Se queremos guardar as notas de 40 alunos, podemos criar um vetor chamado “notas” que irá armazenar as 40 notas, em vez de criar uma variável para cada nota:

```
float notas[40];
```

quantidade de dados (uma vez que o tamanho do vetor é definido, NÃO é possível alterar seu tamanho)

tipo dos dados que serão armazenados

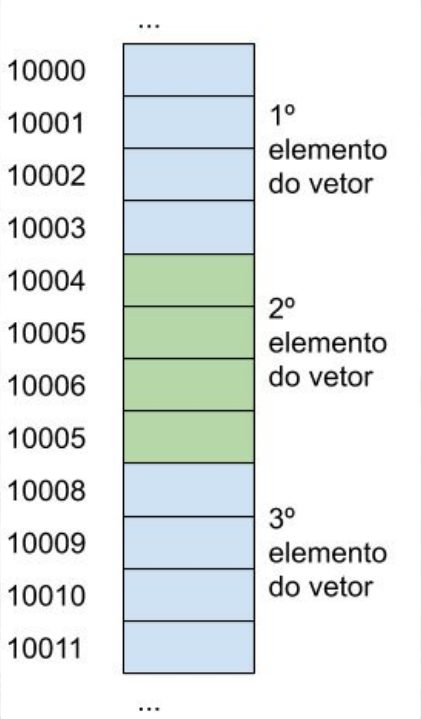
nome da variável

Vetores

- Ao declarar: `float notas[40];`
  - O compilador sabe que precisa reservar 40 blocos de memória, cada um com capacidade de armazenar um dado do tipo float: 40 x 4 = 160 endereços;
  - São reservados endereços em posições **contíguas** de memória.

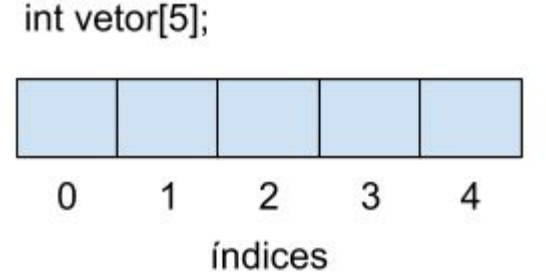
Vetores

- Vetor na memória:



Atribuindo valores

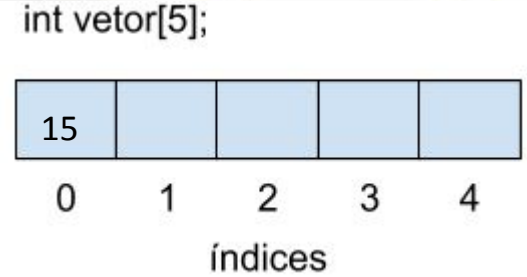
- Se um vetor pode guardar vários valores, como é feita a atribuição desses valores?
- Cada elemento do vetor é associado a um índice, que inicia em zero:



Atribuindo valores

- A partir dos índices é possível manipular os elementos do vetor:

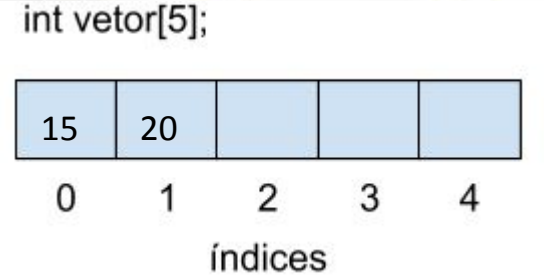
```
vetor[0] = 15;
```



Atribuindo valores

- A partir dos índices é possível manipular os elementos do vetor:

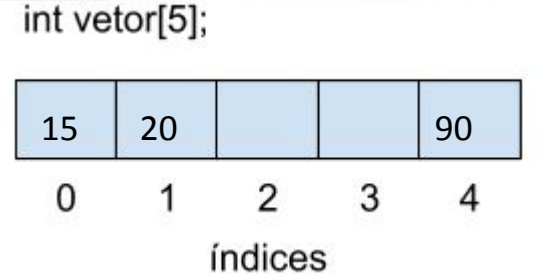
```
vetor[1] = 23;
```



Atribuindo valores

- A partir dos índices é possível manipular os elementos do vetor:

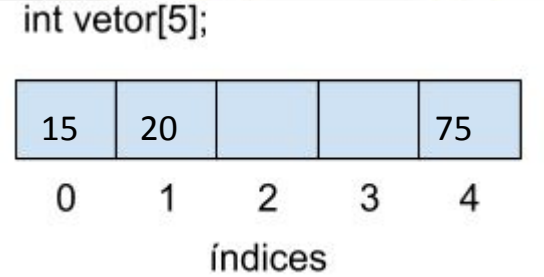
```
vetor[4] = 90;
```



Atribuindo valores

- A partir dos índices é possível manipular os elementos do vetor:

```
vetor[4] = 75;
```



Atribuindo valores

- A partir dos índices é possível manipular os elementos do vetor:

```
vetor[5] = 10;
```

```
*** stack smashing detected ***: <unknown> terminated
Aborted (core dumped)

Process returned 134 (0x86)   execution time : 0.296 s
Press ENTER to continue.
```

```
Segmentation fault (core dumped)

Process returned 139 (0x8B)   execution time : 0.227 s
Press ENTER to continue.
```

Inicializando vetores

- Um vetor pode ser inicializado no momento da declaração:

```
int vetor[5] = {1, 2, 3, 4, 5};
```

- A forma abaixo também está correta:

```
int vetor[] = {1, 2, 3, 4, 5};
```

O tamanho não foi especificado, mas como o vetor foi inicializado com 5 elementos, o compilador entende que seu tamanho é 5.

Inicializando vetores

- Cuidado:
  - O compilador não acusa erro para o seguinte código, apenas fornece um *warning*:

```
int a[2] = {1, 23, 2, 5};
printf("%d\n", a[0]);
printf("%d\n", a[1]);
printf("%d\n",
printf("%d\n", a[3]);
```

Não acontece erro, mas será reservado apenas memória para os dois primeiros itens.

```
a[2]);
```

Percorrendo vetores

- Usando laços para percorrer o vetor:

```c
#include <stdio.h>

int main(void) {
      float notas[10];
      int i;
      for(i = 0; i < 10; i++) { // i < 10, o 10 não é incluído
          printf("Digite a nota do aluno %d: ", i + 1);
          scanf("%f", &notas[i]);
      }
      printf("As notas digitadas foram:");
      for(i = 0; i < 10; i++)
          printf("\nAluno %d: %.1f", i + 1, notas[i]);
      return 0;
}
```

O número de repetições é conhecido (tamanho do vetor), portanto o for é uma estrutura ideal para percorrer um vetor.