

# Vetores

Agradecimento ao Prof. Paulo Henrique Pisani,  
pela seção do material.

# Tópicos

- Vetores

# Vetores

# Vetores

- É um conjunto de variáveis do mesmo tipo:
  - Referenciada por um mesmo identificador;
  - Cada elemento é acessado por meio de um índice.
- **Exemplo:** ler a idade de 10 pessoas e contar quantas estão acima da idade média.
  - Uma alternativa seria criar 10 variáveis;
  - Outra (bem melhor), seria criar um vetor/array de comprimento 10.

# Vetores

- Declarar vetor:

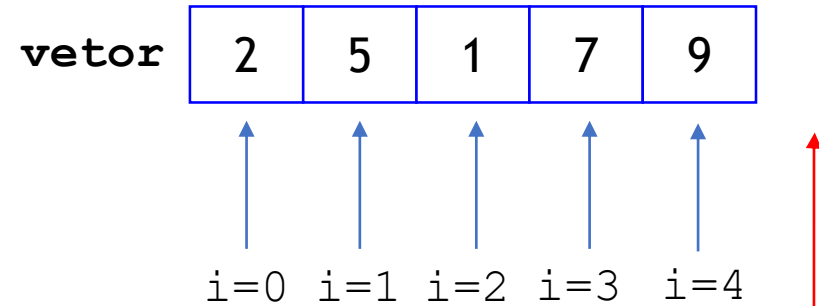
`<tipo> <nome>[<tamanho>];`

## Exemplos

```
int idades[10];  
double vetor2[5];  
int valores[3] = {10, 20, 30};
```

# Vetores

**Índices começam no 0 (zero)**



- **Acessar valores em um vetor:**

```
int vetor[5];  
vetor[4] Acessa elemento de índice 4  
vetor[3] = 7; Atribui valor no elemento de índice 3  
if (vetor[3] == 8) Acessa elemento de índice 3  
    return 0;  
vetor[5] Elemento não existe!
```

- **Ler elementos em um vetor:**

```
int idade[10];  
int i;  
for (i = 0; i < 10; i++)  
    scanf("%d", &idade[i]);
```

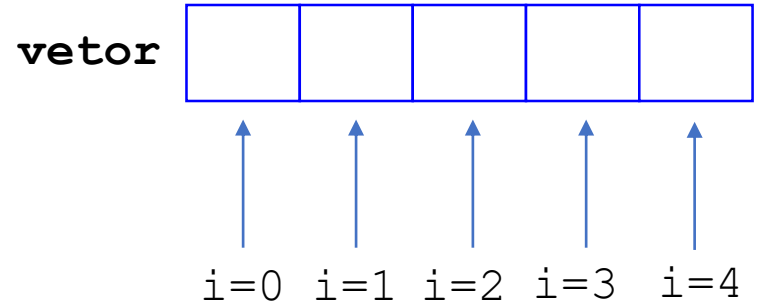
# Vetores

- Percorrer um vetor:

```
int vetor[5];  
int i;  
for (i = 0; i < 5; i++) {  
    vetor[i];  
}
```

- O que faz este código?

```
int vetor[5];  
int i;  
for (i = 0; i < 5; i++) {  
    vetor[i] = (i+1)*(i+1);  
}
```



# Exemplo

- Ler a idade de 10 pessoas e contar quantas estão acima da idade média.



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

```
int main() {
```

Declaração do vetor

```
    int idade[10];
```

```
    int i, soma=0;
```

```
    for (i = 0; i < 10; i++) {
```

```
        scanf("%d", &idade[i]);
```

Importante: lembre-se de usar o "&" no scanf!

```
        soma += idade[i];
```

```
    }
```

```
    double media = soma / 10.0;
```

Necessário para não realizar divisão inteira.

```
    int qtd_acima_media = 0;
```

```
    for (i = 0; i < 10; i++)
```

```
        if (idade[i] > media)
```

```
            qtd_acima_media++;
```

```
    printf("Ha %d pessoa(s) acima da media.", qtd_acima_media);
```

```
    return 0;
```

```
}
```

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

```
int main() {
```

```
    int idade[10];  
    int i, soma=0;  
    for (i = 0; i < 10; i++) {  
        scanf("%d", &idade[i]);  
        soma += idade[i];  
    }
```

```
    double media = soma / 10.0;
```

```
    int qtd_acima_media = 0;  
    for (i = 0; i < 10; i++)  
        if (idade[i] > media)  
            qtd_acima_media++;
```

```
    printf("Ha %d pessoa(s) acima da media.", qtd_acima_media);
```

```
    return 0;
```

```
}
```

**Atenção com os  
limites do vetor!**



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

```
int main() {
```

```
    int idade[10];  
    int i, soma=0;  
    for (i = 0; i <= 10; i++) {  
        scanf("%d", &idade[i]);  
        soma += idade[i];  
    }
```

```
    double media = soma / 10.0;
```

```
    int qtd_acima_media = 0;  
    for (i = 0; i < 20; i++)  
        if (idade[i] > media)  
            qtd_acima_media++;
```

```
    printf("Ha %d pessoa(s) acima da media.", qtd_acima_media);
```

```
    return 0;
```

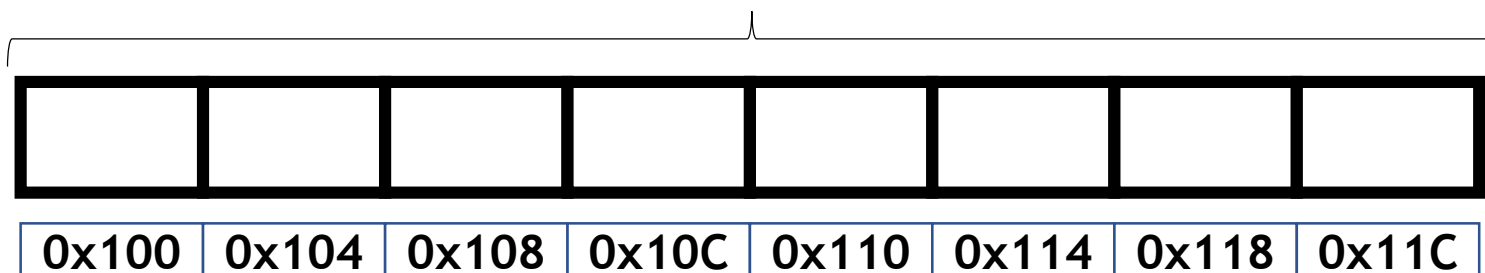
```
}
```

Comportamento  
imprevisível!!!



# Vetores são armazenados em posições consecutivas na memória!

`int vetor[8];`



Endereço de memória do primeiro elemento do vetor.

Observe que cada elemento tem 4 bytes  
(tamanho de um int = `sizeof(int)` )

Este vetor ocupa  $8 * 4 = 32$  bytes  
Ou seja, `qtd_elementos * sizeof(<tipo>)`

# Tamanhos dos tipos de dados (sizeof)

- Para saber quantos bytes um tipo de dados ocupa, usamos **sizeof** (o retorno é do tipo **long int**).

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

```
int main() {  
    int a;  
    long int b;  
    long long int c;  
    float d;  
    double e;  
    char f;
```

```
    printf("%ld %ld %ld %ld %ld %ld",  
           sizeof(a),  
           sizeof(b),  
           sizeof(c),  
           sizeof(d),  
           sizeof(e),  
           sizeof(f));
```

```
    return 0;
```

```
}
```

# Algoritmos importantes

- **Imprimir vetor:** cada elemento em uma linha, todos os elementos na mesma linha;
- **Inserção de elementos em vetor:** início, meio e final (lista estática);
- **Buscar elemento;**
- **Remoção de elementos de vetor:** início/meio e final (lista estática).

# Imprimir vetor

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

```
int main() {
```

```
    int lista[5] = {20, 30, 50, 10, 40};
```

```
    printf("\n");
```

```
    int i;
```

```
    for (i = 0; i < 5; i++)
```

```
        printf("%d ", lista[i]);
```

```
    printf("\n");
```

```
    return 0;
```

```
}
```

# Imprimir vetor

```
#include <stdio.h>

int main() {

    int lista[5] = {20, 30, 50, 10, 40};

    printf("\n");
    int i;
    for (i = 0; i < 5; i++)
        printf("%d ", lista[i]);
    printf("\n");

    return 0;
}
```

**E para imprimir na  
ordem inversa?**



# Imprimir vetor (ordem inversa)

```
#include <stdio.h>

int main() {

    int lista[5] = {20, 30, 50, 10, 40};

    printf("\n");
    int i;
    for (i = 4; i >= 0; i++)
        printf("%d ", lista[i]);
    printf("\n");

    return 0;
}
```

# Inserir elemento no **final** de um vetor

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

```
int main() {
```

```
    int lista[5];
```

```
    int i;
```

```
    for (i = 0; i < 5; i++) {
```

```
        int numero;
```

```
        scanf("%d", &numero);
```

```
        lista[i] = numero;
```

```
    }
```

```
    printf("\n");
```

```
    for (i = 0; i < 5; i++)
```


```
        printf("%d ", lista[i]);
```

```
    printf("\n");
```

```
    return 0;
```

```
}
```

Usamos um índice para  
inserir no final



# Inserir elemento no **final** de um vetor

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

```
int main() {
```

```
    int lista[5];
```

```
    int i = 0;
```

```
    while (i < 5) {
```

```
        int numero;
```

```
        scanf("%d", &numero);
```

```
        lista[i] = numero;
```

```
        i++;
```

```
    }
```

```
    printf("\n");
```

```
    for (i = 0; i < 5; i++)
```


```
        printf("%d ", lista[i]);
```

```
    printf("\n");
```

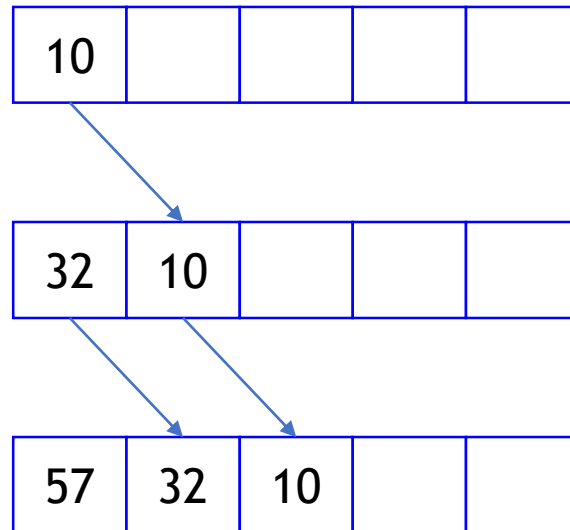
```
    return 0;
```

```
}
```

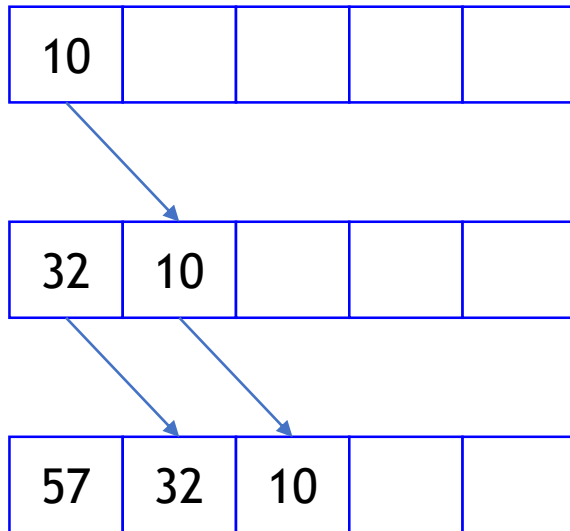
Usamos um índice para  
inserir no final



# Inserir elemento no **início** de um vetor



# Inserir elemento no **início** de um vetor



Deslocamento  
dos elementos a  
frente

```
#include <stdio.h>

int main() {
    int lista[5];
    int i;
    for (i = 0; i < 5; i++) {
        int numero;
        scanf("%d", &numero);

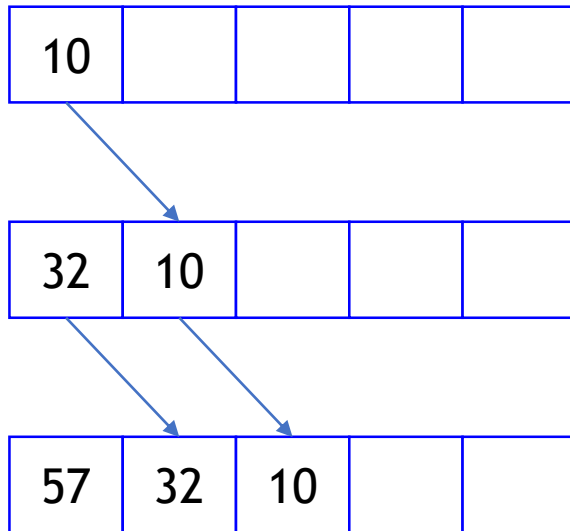
        int k;
        for (k = 5-1; k > 0; k--)
            lista[k] = lista[k-1];

        lista[0] = numero;
    }

    printf("\n");
    for (i = 0; i < 5; i++)
        printf("%d ", lista[i]);
    printf("\n");

    return 0;
}
```

# Inserir elemento no **início** de um vetor



**Podemos melhorar esse programa?**

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

```
int main() {  
    int lista[5];  
    int i;  
    for (i = 0; i < 5; i++) {  
        int numero;  
        scanf("%d", &numero);
```

Deslocamento  
dos elementos a  
frente

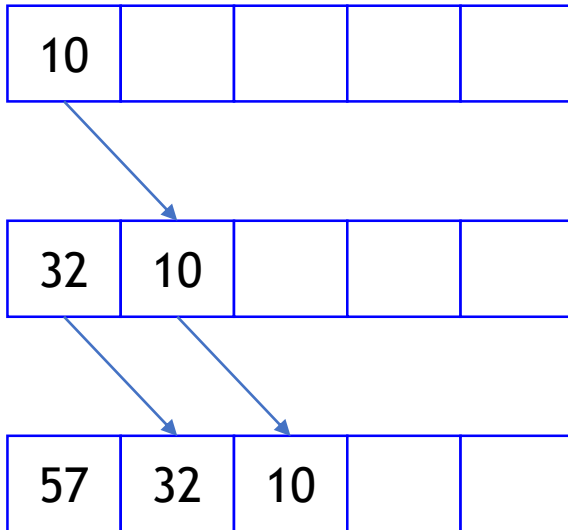
```
int k;  
for (k = 5-1; k > 0; k--)  
    lista[k] = lista[k-1];  
lista[0] = numero;
```

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

```
return 0;
```

```
}
```

# Inserir elemento no **início** de um vetor



**Podemos melhorar esse programa?**

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

```
int main() {  
    int lista[5];  
    int i;  
    for (i = 0; i < 5; i++) {  
        int numero;  
        scanf("%d", &numero);
```

Deslocamento  
dos elementos a  
frente

```
int k;  
for (k = i; k > 0; k--)  
    lista[k] = lista[k-1];  
lista[0] = numero;
```

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

```
return 0;
```

```
}
```

**E se quisermos  
trabalhar com um  
vetor de 10  
elementos?**


```
#include <stdio.h>

int main() {
    int lista[5];
    int i;
    for (i = 0; i < 5; i++) {
        int numero;
        scanf("%d", &numero);

        int k;
        for (k = i; k > 0; k--)
            lista[k] = lista[k-1];
        lista[0] = numero;
    }

    printf("\n");
    for (i = 0; i < 5; i++)
        printf("%d ", lista[i]);
    printf("\n");

    return 0;
}
```





**E se quisermos  
trabalhar com um  
vetor de 10  
elementos?**

Precisamos mudar o valor “5”  
(que está “hardcoded”) em todo  
o código! Como evitar isso?


```
#include <stdio.h>

int main() {
    int lista[5];
    int i;
    for (i = 0; i < 5; i++) {
        int numero;
        scanf("%d", &numero);

        int k;
        for (k = i; k > 0; k--)
            lista[k] = lista[k-1];
        lista[0] = numero;
    }

    printf("\n");
    for (i = 0; i < 5; i++)
        printf("%d ", lista[i]);
    printf("\n");

    return 0;
}
```



# Buscar elemento



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

```
int main() {
```

```
    int lista[5] = {20, 30, 50, 10, 40};
```

```
    int numero;
```

```
    scanf("%d", &numero);
```

```
    int i, encontrou=0;
    for (i = 0; i < 5; i++) {
        if (lista[i] == numero)
            encontrou = 1;
    }
```

```
    if (encontrou)
```

```
        printf("Encontrou!\n");
```

```
    else
```

```
        printf("NAO Encontrou!\n");
```

```
    return 0;
```

```
}
```

# Buscar elemento



```
#include <stdio.h>

int main() {

    int lista[5] = {20, 30, 50, 10, 40};

    int numero;
    scanf("%d", &numero);

    int i, encontrou=0;
    for (i = 0; i < 5; i++) {
        if (lista[i] == numero)
            encontrou = 1;
    }

    if (encontrou)
        printf("Encontrou!\n");
    else
        printf("NAO Encontrou!\n");

    return 0;
}
```

**Podemos melhorar  
esse programa?**

# Buscar elemento

```
int i, encontreu=0;
for (i = 0; i < 5; i++) {
    if (lista[i] == numero)
        encontreu = 1;
}
```

```
int i, encontreu=0;
for (i = 0; i < 5; i++) {
    if (lista[i] == numero) {
        encontreu = 1;
        break;
    }
}
```

```
int i, encontreu=0;
for (i = 0; i < 5 && !encontreu; i++) {
    if (lista[i] == numero)
        encontreu = 1;
}
```

Passando vetores  
como argumento

# Variáveis são passadas por valor

```
#include <stdio.h>

void muda_valor(int parametro) {
    parametro = 507;

    printf("%d\n", parametro);
}

int main() {

    int n = 1000;

    muda_valor(n);

    printf("%d\n", n);

    return 0;
}
```

Qual a saída  
desse programa?

# Variáveis são passadas por valor

```
#include <stdio.h>

void muda_valor(int parametro) {
    parametro = 507;

    printf("%d\n", parametro);
}

int main() {

    int n = 1000;

    muda_valor(n);

    printf("%d\n", n);

    return 0;
}
```

Qual a saída  
desse programa?

507  
1000

Ok, variáveis são  
passadas por valor!

```
#include <stdio.h>

void muda_valor(int vetor[]) {
    vetor[0] = 90;

    printf("%d\n", vetor[0]);
}

int main() {

    int v[3] = {200, 500, 300};

    muda_valor(v);

    printf("%d %d %d\n", v[0], v[1], v[2]);

    return 0;
}
```

Qual a saída  
desse programa?



# Mas vetores são passados **por referência!**

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

```
void muda_valor(int vetor[]) {  
    vetor[0] = 90;  
  
    printf("%d\n", vetor[0]);  
}
```

```
int main() {  
  
    int v[3] = {200, 500, 300};  
  
    muda_valor(v);  
  
    printf("%d %d %d\n", v[0], v[1], v[2]);  
  
    return 0;  
}
```

Qual a saída  
desse programa?

90

90 500 300

**Mas por-que é assim?**

# Variáveis

```
int matricula = 123;
```

- O identificador de uma variável é usado para acessar seu valor;

```
printf("%d\n", matricula);
```

- O endereço de memória da variável é acessado com o operador address-of &

```
printf("%p\n", &matricula);
```

# Vetores

```
int vetor[3] = {20, 500, 7};
```

- O identificador de um vetor representa o **endereço do primeiro elemento!**

```
printf("%p\n", vetor);  
printf("%p\n", &vetor[0]);
```



Retorna o mesmo valor  
nos dois casos!

# Vetores são passados **por referência!**

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

```
void muda_valor(int vetor[]) {  
    vetor[0] = 90;  
  
    printf("%d\n", vetor[0]);  
}
```

```
int main() {  
  
    int v[3] = {200, 500, 300};  
  
    muda_valor(v);  
  
    printf("%d %d %d\n", v[0], v[1], v[2]);  
  
    return 0;  
}
```

Qual a saída  
desse programa?

90

90 500 300

# Exemplo

- Criar uma função que troca todos os números primos de um vetor pelo número -1.

# Exercícios

1. Implemente os algoritmos de inserção (que vimos no início da aula) em forma de função;
2. Implemente uma função para remover o elemento de índice  $i$  de um vetor. Essa função deve deslocar (para a esquerda) os elementos que estão a frente no vetor.

# Referências

- Slides do Prof. Fabrício Olivetti:
  - <http://folivetti.github.io/courses/ProgramacaoEstruturada/>
- Slide do Prof. Monael Pinheiro Ribeiro:
  - <https://sites.google.com/site/aed2018q1/>
- CELES, W.; CERQUEIRA, R.; RANGEL, J. L. Introdução a Estruturas de Dados. Elsevier/Campus, 2004.



# Bibliografia básica

- CORMEN, T. H.; LEISERSON, C. E.; RIVEST, R. L.; STEIN, C. Algoritmos: teoria e prática. 2. ed. Rio de Janeiro, RJ: Campus, 2002.
- FORBELLONE, A. L. V.; EBERSPACHER, H. F. Lógica de programação: a construção de algoritmos e estruturas de dados. 3. ed. São Paulo, SP: Prentice Hall, 2005.
- PINHEIRO, F. A. C. Elementos de programação em C. Porto Alegre, RS: Bookman, 2012.

# Bibliografia complementar

- AGUILAR, L. J. Programação em C++: algoritmos, estruturas de dados e objetos. São Paulo, SP: McGraw-Hill, 2008.
- DROZDEK, A. Estrutura de dados e algoritmos em C++. São Paulo, SP: Cengage Learning, 2009.
- KNUTH D. E. The art of computer programming. Upper Saddle River, USA: Addison- Wesley, 2005.
- SEDGEWICK, R. Algorithms in C++: parts 1-4: fundamentals, data structures, sorting, searching. Reading, USA: Addison-Wesley, 1998.
- SZWARCFITER, J. L.; MARKENZON, L. Estruturas de dados e seus algoritmos. 3. ed. Rio de Janeiro, RJ: LTC, 1994.
- TENENBAUM, A. M.; LANGSAM, Y.; AUGENSTEIN, M. J. Estruturas de dados usando C. São Paulo, SP: Pearson Makron Books, 1995.