Programação

C++

Aula 03

Jorgiano Vidal

Agenda

- Arrays/vetores
- Acesso
- Percorrer
- Filtro
- Leitura e escrita
- Parâmetros de funções
- Referências
- Comentários

Arrays

Arrays

- Agregado homogêneo de elementos
  - Capacidade **finita** e **definida**
- Sintaxe

```
tipo ID[CAPACIDADE];
```

- Exemplos:

```
int a[10];
float notas[15];
```

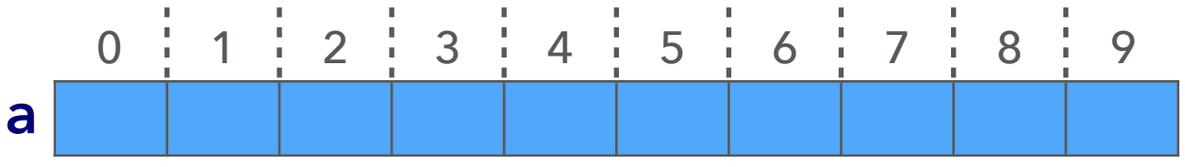
Arrays

- Agregado homogêneo de elementos
  - Capacidade **finita** e **definida**
- Sintaxe

```
tipo ID[CAPACIDADE];
```

- Exemplos:

```
int a[10];
float notas[15];
```



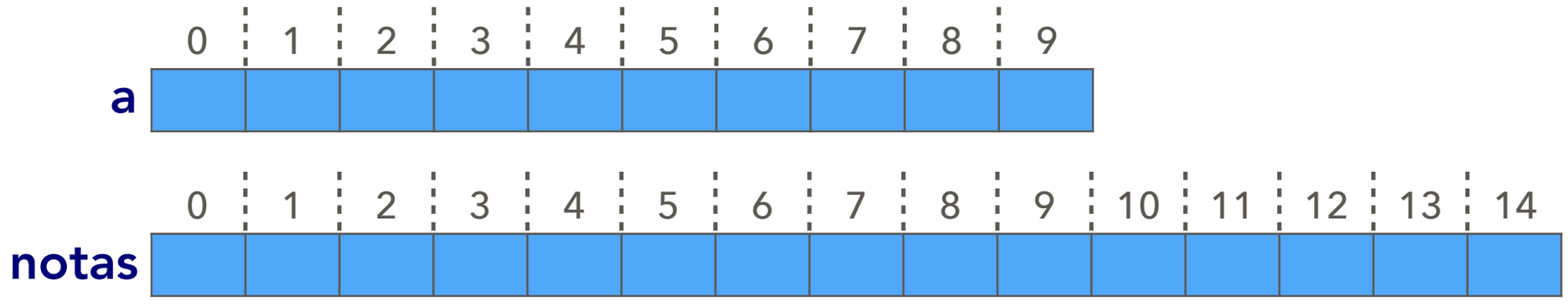
Arrays

- Agregado homogêneo de elementos
  - Capacidade **finita** e **definida**
- Sintaxe

```
tipo ID[CAPACIDADE];
```

- Exemplos:

```
int a[10];
float notas[15];
```



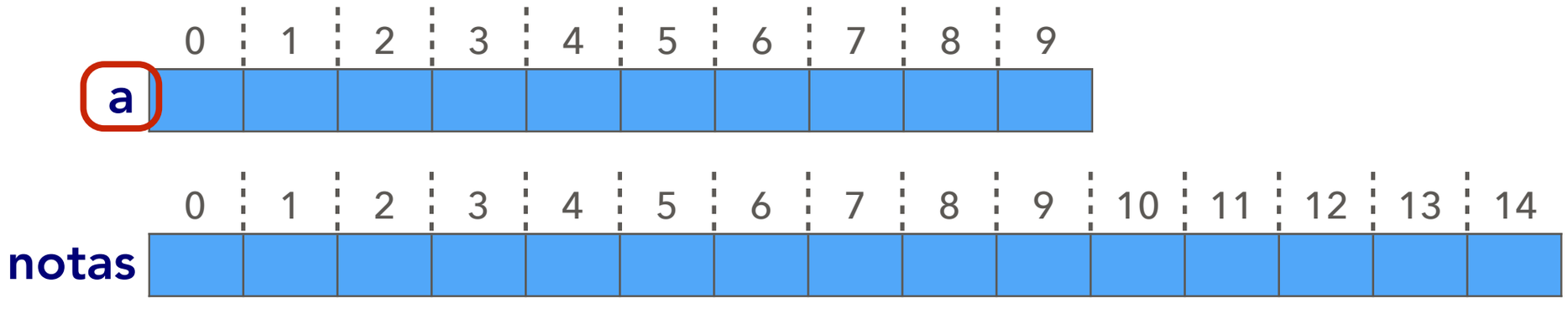
Arrays

- Agregado homogêneo de elementos
  - Capacidade **finita** e **definida**
- Sintaxe

```
tipo ID[CAPACIDADE];
```

- Exemplos:

```
int a[10];
float notas[15];
```



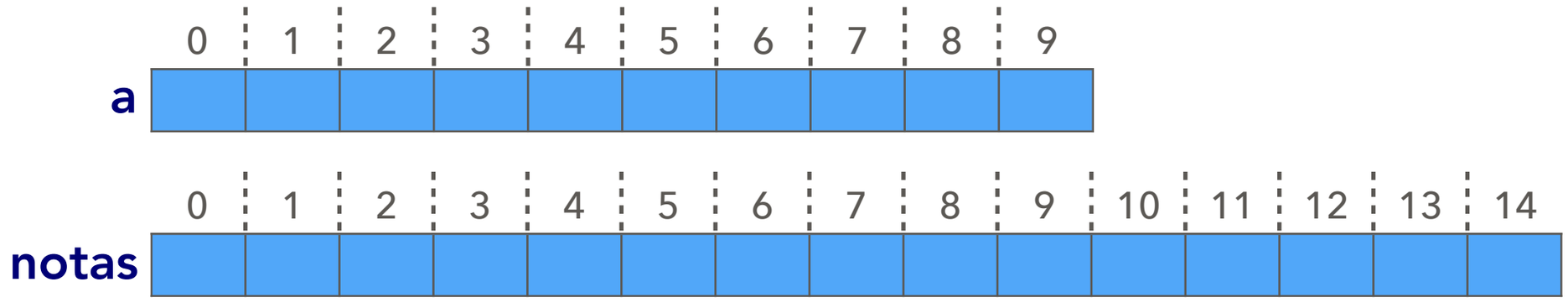
Arrays

- Agregado homogêneo de elementos
  - Capacidade **finita** e **definida**
- Sintaxe

```
tipo ID[CAPACIDADE];
```

- Exemplos:

```
int a[10];
float notas[15];
```



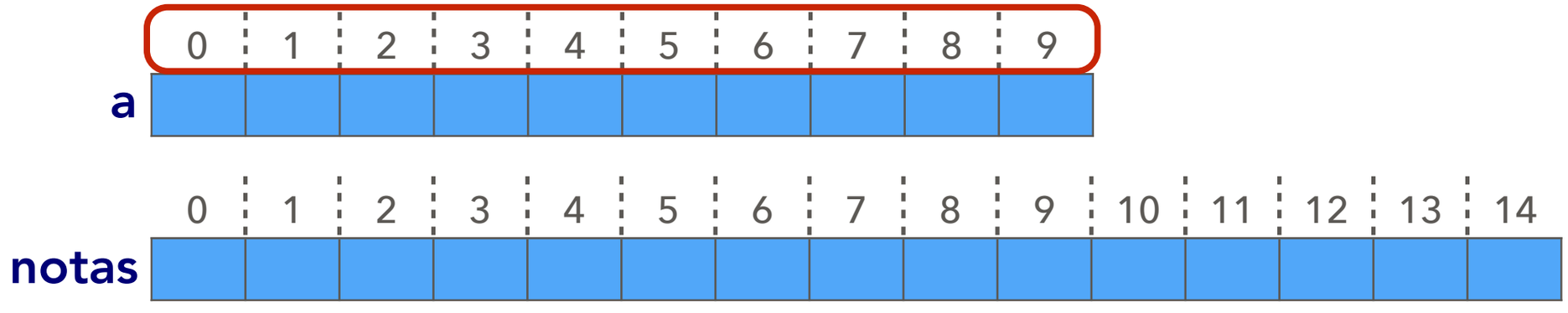
Arrays

- Agregado homogêneo de elementos
  - Capacidade **finita** e **definida**
- Sintaxe

```
tipo ID[CAPACIDADE];
```

- Exemplos:

```
int a[10];
float notas[15];
```



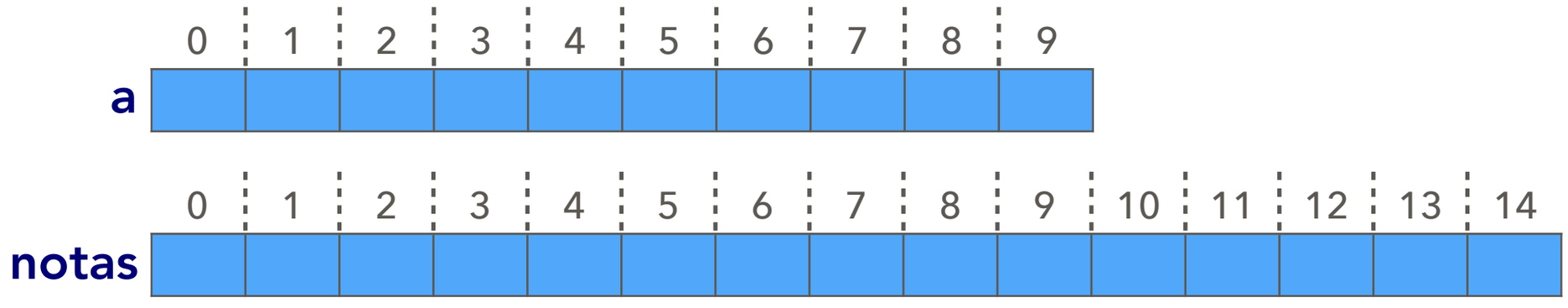
Arrays

- Agregado homogêneo de elementos
  - Capacidade **finita** e **definida**
- Sintaxe

```
tipo ID[CAPACIDADE];
```

- Exemplos:

```
int a[10];
float notas[15];
```



Acesso

- Operador de indexação

  `[]`

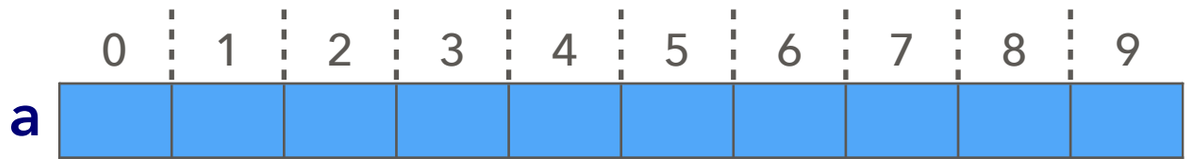
- Ao lado do identificador

```
a[1]=10;
```

- Índices válidos de `0` a `capacidade-1`
- Índice: **inteiro**

- Expressões

```
a[x+5]=c[y-3]
```



Acesso

- Operador de indexação

  `[]`

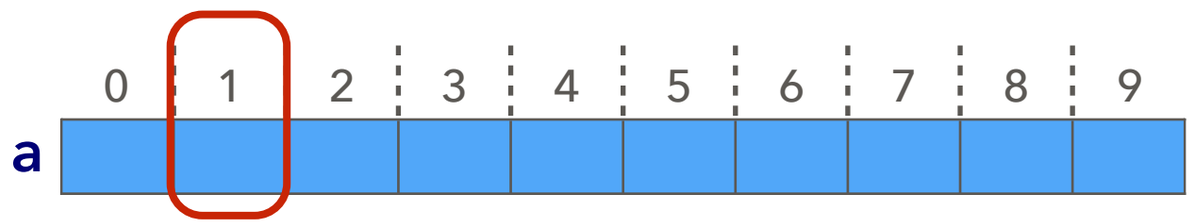
- Ao lado do identificador

```
a[1]=10;
```

- Índices válidos de `0` a `capacidade-1`
- Índice: **inteiro**

- Expressões

```
a[x+5]=c[y-3]
```



Acesso

- Operador de indexação

  `[]`

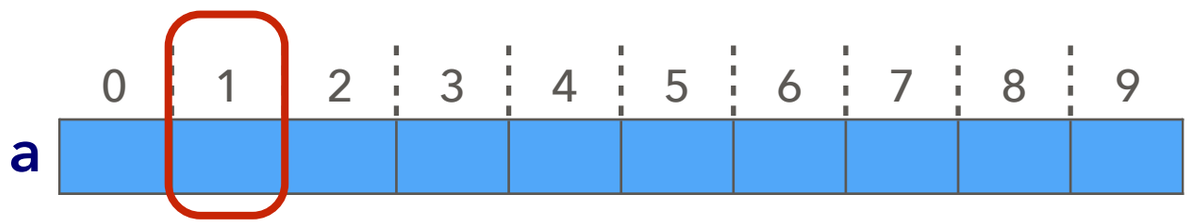
- Ao lado do identificador

```
a[1]=10;
```

- Índices válidos de `0` a `capacidade-1`
- Índice: **inteiro**

- Expressões

  a[x+5]=c[y-3]



Acesso

- Operador de indexação

  `[]`

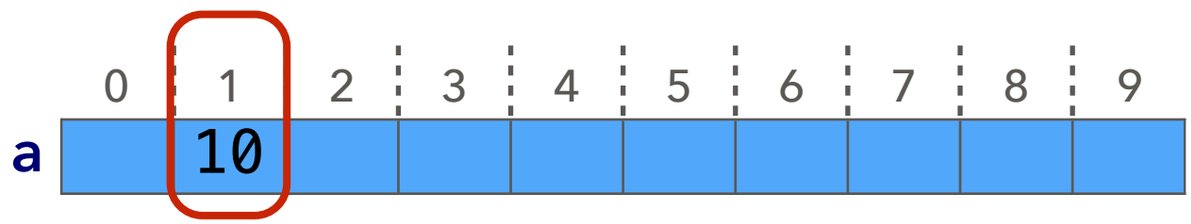
- Ao lado do identificador

```
a[1]=10;
```

- Índices válidos de `0` a `capacidade-1`
- Índice: **inteiro**

- Expressões

  `a[x+5]=c[y-3]`



Acesso

- Operador de indexação

  `[]`

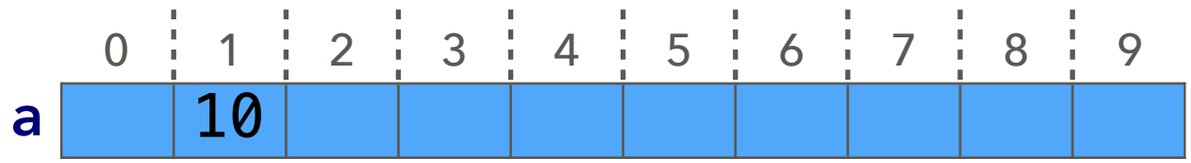
- Ao lado do identificador

```
a[1]=10;
```

- Índices válidos de `0` a `capacidade-1`
- Índice: **inteiro**

- Expressões

```
a[x+5]=c[y-3]
```



Acesso

- Operador de indexação

  `[]`

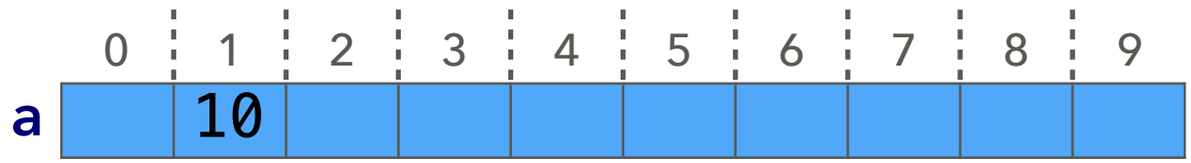
- Ao lado do identificador

```
a[1]=10;
```

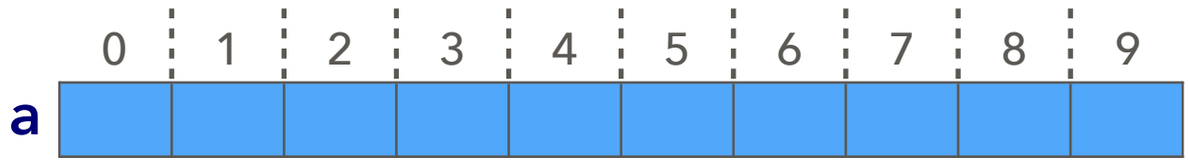
- Índices válidos de `0` a `capacidade-1`
- Índice: **inteiro**

- Expressões

```
a[x+5]=c[y-3]
```

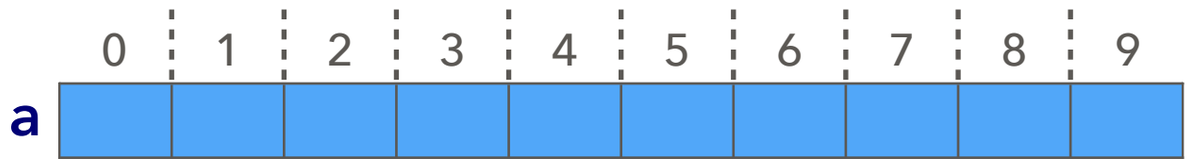


Percorrer



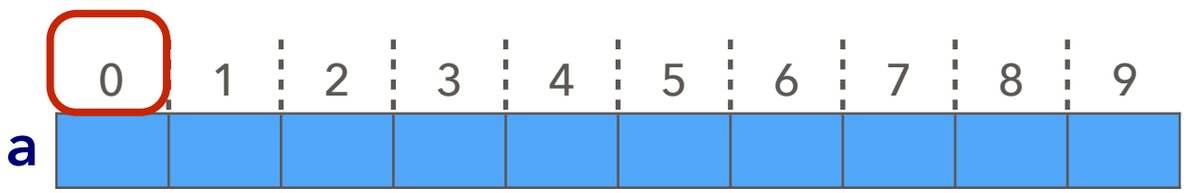
Percorrer

```
int a[10];
int i;
/* ... */
for (i=0 ; i<10 ; ++i){
   /* ... */
   /* usa elemento i */
   ... a[i]...
   /*...*/
}
```



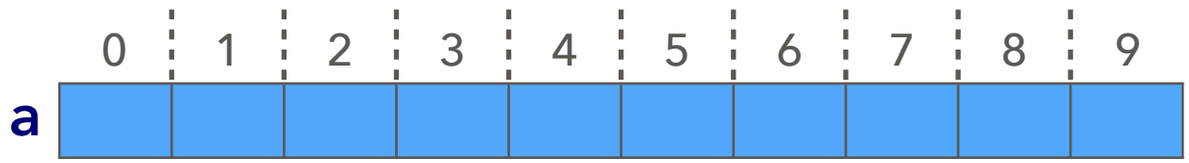
Percorrer

```
int a[10];
int i;
/* ... */
for (i=0 ; i<10 ; ++i){
   /* ... */
   /* usa elemento i */
   ... a[i]...
   /*...*/
}
```



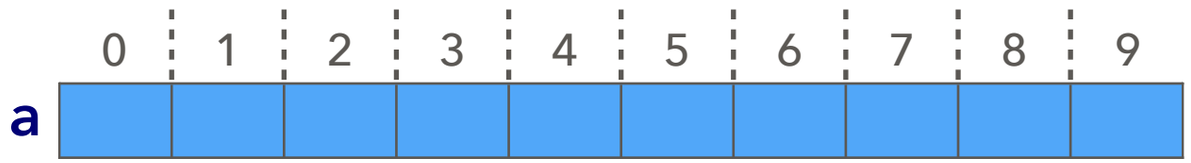
Percorrer

```
int a[10];
int i;
/* ... */
for (i=0 ; i<10 ; ++i){
   /* ... */
   /* usa elemento i */
   ... a[i]...
   /*...*/
}
```



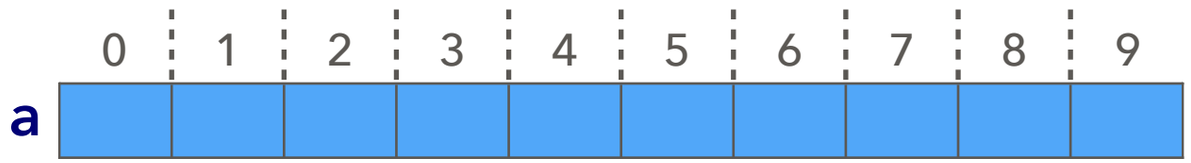
Percorrer

```
int a[10];
int i;
/* ... */
for (i=0 ; i<10 ; ++i){
   /* ... */
   /* usa elemento i */
   ... a[i]...
   /*...*/
}
```



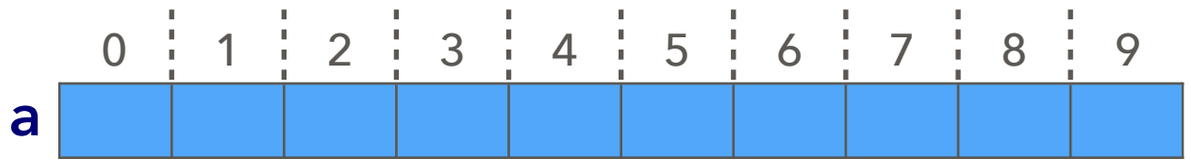
Percorrer

```
int a[10];
int i;
/* ... */
for (i=0 ; i<10 ; ++i){
   /* ... */
   /* usa elemento i */
   ... a[i]...
   /*...*/
}
```



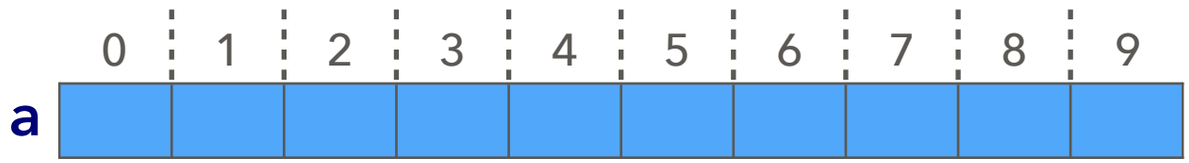
Percorrer

```
int a[10];
int i;
/* ... */
for (i=0 ; i<10 ; ++i){
   /* ... */
   /* usa elemento i */
   ... a[i]...
   /*...*/
}
```



Percorrer

```
int a[10];
int i;
/* ... */
for (i=0 ; i<10 ; ++i){
   /* ... */
   /* usa elemento i */
   ... a[i]...
   /*...*/
}
```



Percorrer

- Média

Percorrer

- Média

```
int i;
Double notas[30], soma, media;

soma = 0;
for (i = 0; i < 30; ++i)
    soma += notas[i];
media = soma / 30;
```

Percorrer

- Média

```
int i;
Double notas[30], soma, media;

soma = 0;
for (i = 0; i < 30; ++i)
    soma += notas[i];
media = soma / 30;
```

Percorrer

- Média

```
int i;
Double notas[30], soma, media;

soma = 0;
for (i = 0; i < 30; ++i)
    soma += notas[i];
media = soma / 30;
```

Percorrer

- Média

```
int i;
Double notas[30], soma, media;

soma = 0;
for (i = 0; i < 30; ++i)
    soma += notas[i];
media = soma / 30;
```

Percorrer

- Média

```
int i;
Double notas[30], soma, media;

soma = 0;
for (i = 0; i < 30; ++i)
    soma += notas[i];
media = soma / 30;
```

Percorrer

- Média

```
int i;
Double notas[30], soma, media;

soma = 0;
for (i = 0; i < 30; ++i)
    soma += notas[i];
media = soma / 30;
```

Percorrer

- Média

```
int i;
Double notas[30], soma, media;

soma = 0;
for (i = 0; i < 30; ++i)
    soma += notas[i];
media = soma / 30;
```

Percorrer

- Média

```
int i;
Double notas[30], soma, media;

soma = 0;
for (i = 0; i < 30; ++i)
    soma += notas[i];
media = soma / 30;
```

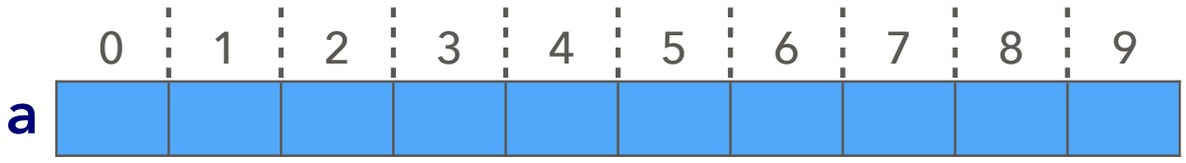
Percorrer

- Média

```
int i;
Double notas[30], soma, media;

soma = 0;
for (i = 0; i < 30; ++i)
    soma += notas[i];
media = soma / 30;
```

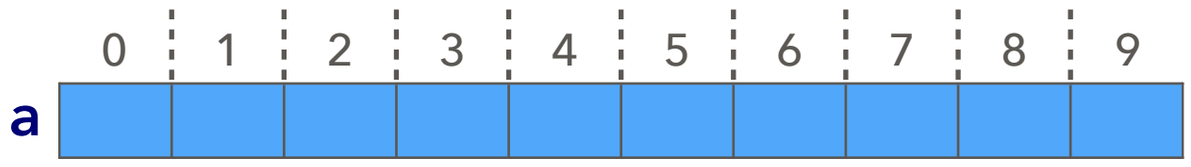
Filtro

```
int a[10];
int i;
/* ... */
for (i=0 ; i<10 ; ++i){
  /* ... */
  /* Filtra e usa elemento i */
  if (/*condicao de a[i]*/){
    ... a[i]...
  }
  /*...*/
}
```



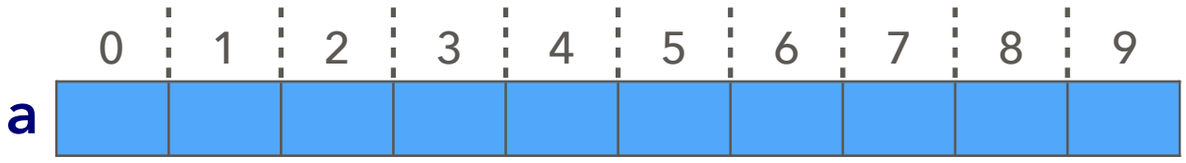
Filtro

```
int a[10];
int i;
/* ... */
for (i=0 ; i<10 ; ++i){
  /* ... */
  /* Filtra e usa elemento i */
  if (/*condicao de a[i]*/){
    ... a[i]...
  }
  /*...*/
}
```



Filtro

```
int a[10];
int i;
/* ... */
for (i=0 ; i<10 ; ++i){
  /* ... */
  /* Filtra e usa elemento i */
  if (/*condicao de a[i]*/){
    ... a[i]...
  }
  /*...*/
}
```



Filtro

- Contar pares e ímpares

- Contar pares e ímpares

```
int a[1000];
int i,pares,impares;
pares=0;
for (i=0 ; i<10000 ; ++i){
   if (a[i]%2==0)
      pares++;
}
```

- Contar pares e ímpares

```
int a[1000];
int i,pares,impares;
pares=0;
for (i=0 ; i<10000 ; ++i){
   if (a[i]%2==0)
      pares++;
}
```

- Contar pares e ímpares

```
int a[1000];
int i,pares,impares;
pares=0;
for (i=0 ; i<10000 ; ++i){
   if (a[i]%2==0)
      pares++;
}
```

- Contar pares e ímpares

```
int a[1000];
int i,pares,impares;
pares=0;
for (i=0 ; i<10000 ; ++i){
  if (a[i]%2==0)
    pares++;
}
```

- Contar pares e ímpares

```
int a[1000];
int i,pares,impares;
pares=0;
for (i=0 ; i<10000 ; ++i){
   if (a[i]%2==0)
      pares++;
}
```

- Contar pares e ímpares

```
int a[1000];
int i,pares,impares;
pares=0;
for (i=0 ; i<10000 ; ++i){
   if (a[i]%2==0)
      pares++;
}
impares=1000-pares;
```

Ler/escrever dados

```cpp
#include <iostream>

int main() {
    int a[10], i;
    for (i = 0; i < 10; ++i)
        std::cin >> a[i];

    for (i = 0; i < 10; ++i)
        std::cout << a[i];
    std::cout << std::endl;

    return 0;
}
```

Ler/escrever dados

```cpp
#include <iostream>

int main() {
    int a[10], i;
    for (i = 0; i < 10; ++i)
        std::cin >> a[i];

    for (i = 0; i < 10; ++i)
        std::cout << a[i];
    std::cout << std::endl;

    return 0;
}
```

Ler/escrever dados

```cpp
#include <iostream>

int main() {
    int a[10], i;
    for (i = 0; i < 10; ++i)
        std::cin >> a[i];

    for (i = 0; i < 10; ++i)
        std::cout << a[i];
    std::cout << std::endl;

    return 0;
}
```

Ler/escrever dados

```cpp
#include <iostream>

int main() {
    int a[10], i;
    for (i = 0; i < 10; ++i)
        std::cin >> a[i];

    for (i = 0; i < 10; ++i)
        std::cout << a[i];
    std::cout << std::endl;

    return 0;
}
```

Ler/escrever dados

```cpp
#include <iostream>

int main() {
    int a[10], i;
    for (i = 0; i < 10; ++i)
        std::cin >> a[i];

    for (i = 0; i < 10; ++i)
        std::cout << a[i];
    std::cout << std::endl;

    return 0;
}
```

Ler/escrever dados

```cpp
#include <iostream>

int main() {
    int a[10], i;
    for (i = 0; i < 10; ++i)
        std::cin >> a[i];

    for (i = 0; i < 10; ++i)
        std::cout << a[i] << " ";
    std::cout << std::endl;

    return 0;
}
```

Ler/escrever dados

```cpp
#include <iostream>

int main() {
    int a[10], i;
    for (i = 0; i < 10; ++i)
        std::cin >> a[i];

    for (i = 0; i < 10; ++i)
        std::cout << a[i] << " ";
    std::cout << std::endl;

    return 0;
}
```

Ler/escrever dados

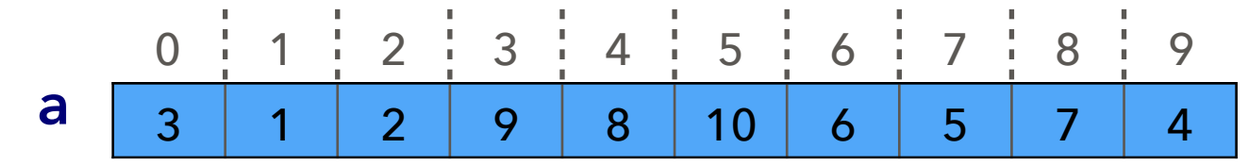
```cpp
#include <iostream>

int main() {
    int a[10], i;
    for (i = 0; i < 10; ++i)
        std::cin >> a[i];

    for (i = 0; i < 10; ++i)
        std::cout << a[i] << " ";
    std::cout << std::endl;

    return 0;
}
```

Você sabe?

- **Verificar** se um elemento x **está** em um *array*?
- **Identificar** o maior/menor elemento de um *array*?]
  - E o **índice** do maior/menor elemento?
- **Contar** os elementos positivos em um *array*?
  - E os **negativos**? **Pares**? **Ímpares**?
- **Contar** os elementos que estão acima da média dos elementos de um *array*?
- **Contar** quantos números primos em um *array* de inteiros?
- **Determinar** se existem números repetido?
  - Quantos? Quais?

Maior elemento

```cpp
#include <iostream>

int maior_array(int a[], int tamanho) {
    int maior = a[0];
    for (int i = 1; i < tamanho;++i)
        if (a[i] > maior)
            maior = a[i];
    return maior;
}

int main() {
    int n;
    std::cin >> n;
    int a[n];
    for (int i=0 ; i<n ; ++i)
        std::cin >> a[i];
    int maior = maior_array(a,n);
    std::cout << maior << std::endl;
    return 0;
}
```



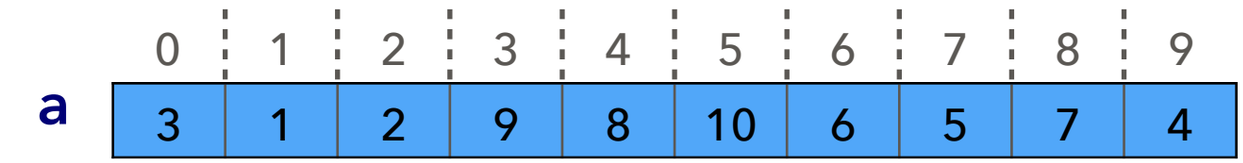
Maior elemento

```
#include <iostream>

int maior_array(int a[], int tamanho) {
    int maior = a[0];
    for (int i = 1; i < tamanho;++i)
        if (a[i] > maior)
            maior = a[i];
    return maior;
}

int main() {
    int n;
    std::cin >> n;
    int a[n];
    for (int i=0 ; i<n ; ++i)
        std::cin >> a[i];
    int maior = maior_array(a,n);
    std::cout << maior << std::endl;
    return 0;
}
```

	0	1	2	3	4	5	6	7	8	9
a	3	1	2	9	8	10	6	5	7	4

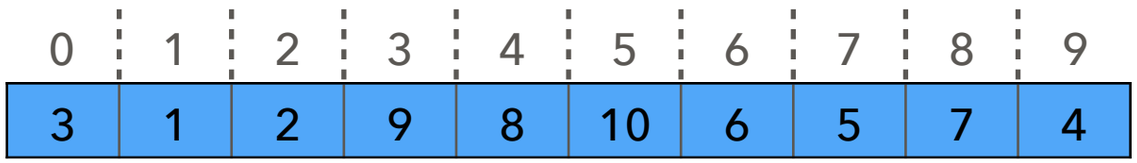
Maior elemento

```
#include <iostream>

int maior_array(int a[], int tamanho) {
    int maior = a[0];
    for (int i = 1; i < tamanho;++i)
        if (a[i] > maior)
            maior = a[i];
    return maior;
}

int main() {
    int n;
    std::cin >> n;
    int a[n];
    for (int i=0 ; i<n ; ++i)
        std::cin >> a[i];
    int maior = maior_array(a,n);
    std::cout << maior << std::endl;
    return 0;
}
```



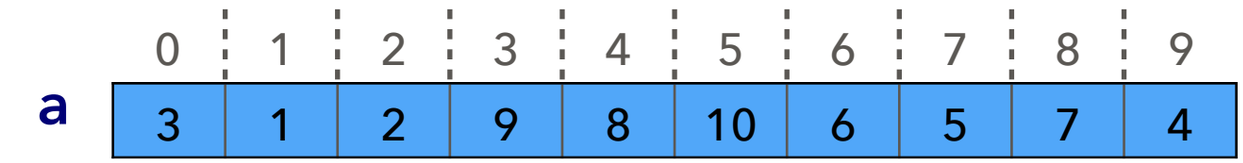
Maior elemento

```cpp
#include <iostream>

int maior_array(int a[], int tamanho) {
    int maior = a[0];
    for (int i = 1; i < tamanho;++i)
        if (a[i] > maior)
            maior = a[i];
    return maior;
}

int main() {
    int n;
    std::cin >> n;
    int a[n];
    for (int i=0 ; i<n ; ++i)
        std::cin >> a[i];
    int maior = maior_array(a,n);
    std::cout << maior << std::endl;
    return 0;
}
```



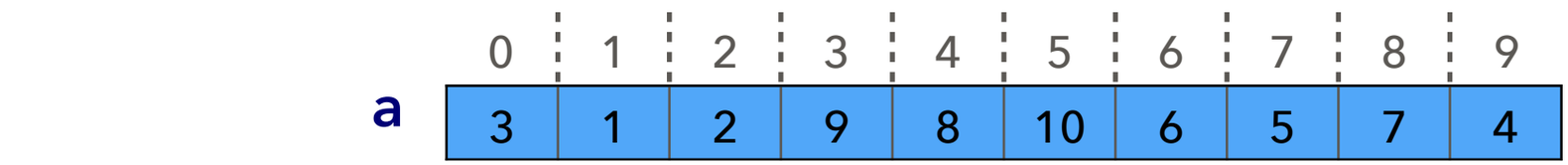
Maior elemento

```cpp
#include <iostream>

int maior_array(int a[], int tamanho) {
    int maior = a[0];
    for (int i = 1; i < tamanho;++i)
        if (a[i] > maior)
            maior = a[i];
    return maior;
}

int main() {
    int n;
    std::cin >> n;
    int a[n];
    for (int i=0 ; i<n ; ++i)
        std::cin >> a[i];
    int maior = maior_array(a,n);
    std::cout << maior << std::endl;
    return 0;
}
```



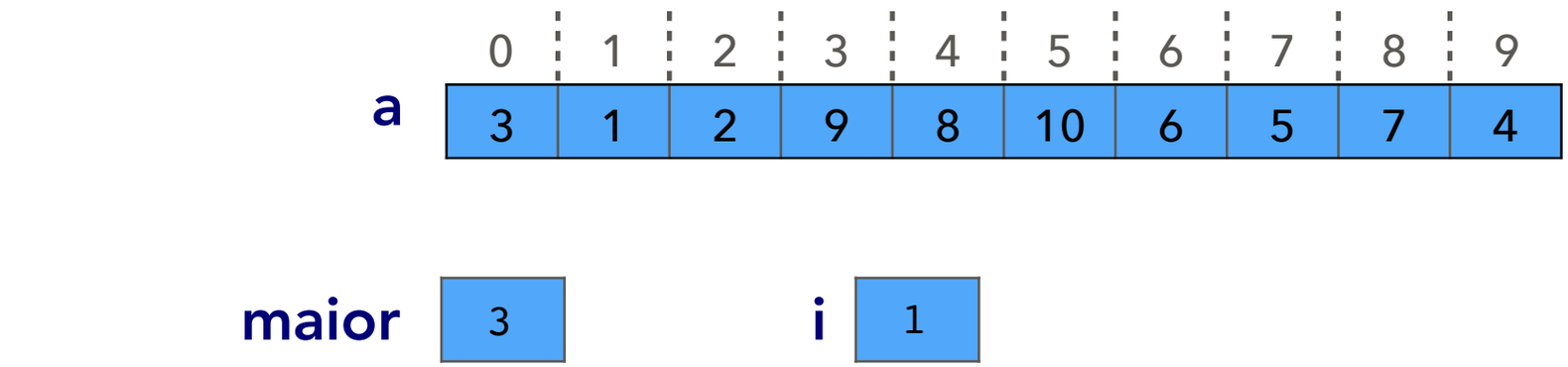
Maior elemento

```
#include <iostream>

int maior_array(int a[], int tamanho) {
    int maior = a[0];
    for (int i = 1; i < tamanho;++i)
        if (a[i] > maior)
            maior = a[i];
    return maior;
}

int main() {
    int n;
    std::cin >> n;
    int a[n];
    for (int i=0 ; i<n ; ++i)
        std::cin >> a[i];
    int maior = maior_array(a,n);
    std::cout << maior << std::endl;
    return 0;
}
```



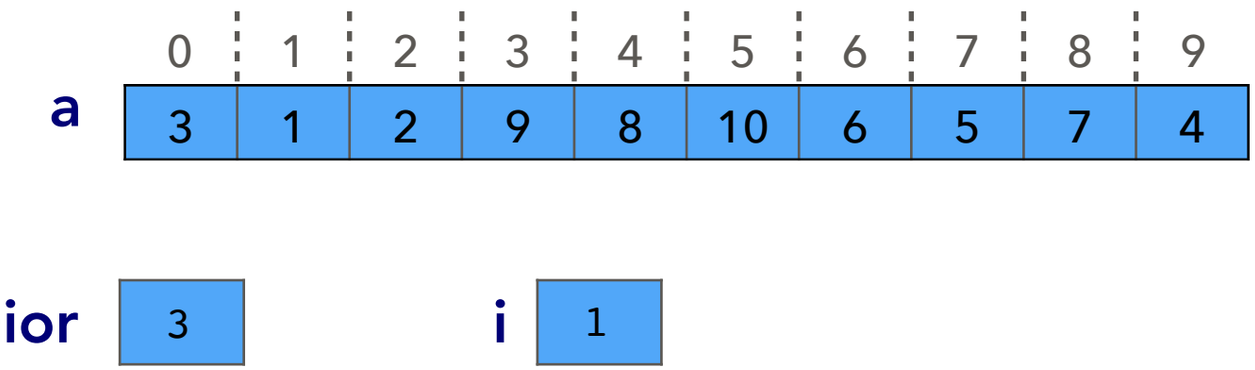
Maior elemento

```cpp
#include <iostream>

int maior_array(int a[], int tamanho) {
    int maior = a[0];
    for (int i = 1; i < tamanho;++i)
        if (a[i] > maior)
            maior = a[i];
    return maior;
}

int main() {
    int n;
    std::cin >> n;
    int a[n];
    for (int i=0 ; i<n ; ++i)
        std::cin >> a[i];
    int maior = maior_array(a,n);
    std::cout << maior << std::endl;
    return 0;
}
```



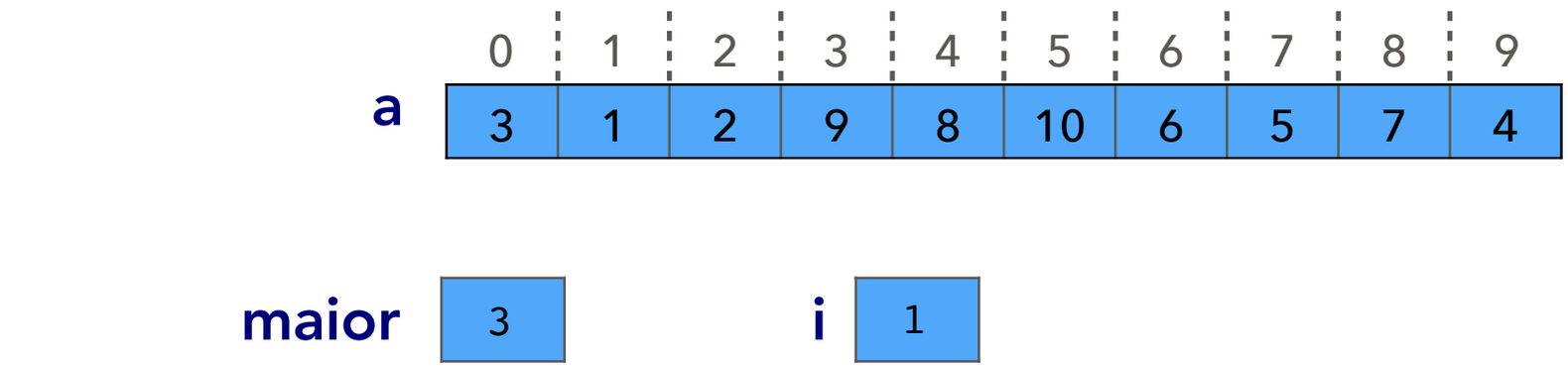
Maior elemento

```cpp
#include <iostream>

int maior_array(int a[], int tamanho) {
    int maior = a[0];
    for (int i = 1; i < tamanho;++i)
        if (a[i] > maior)
            maior = a[i];
    return maior;
}

int main() {
    int n;
    std::cin >> n;
    int a[n];
    for (int i=0 ; i<n ; ++i)
        std::cin >> a[i];
    int maior = maior_array(a,n);
    std::cout << maior << std::endl;
    return 0;
}
```



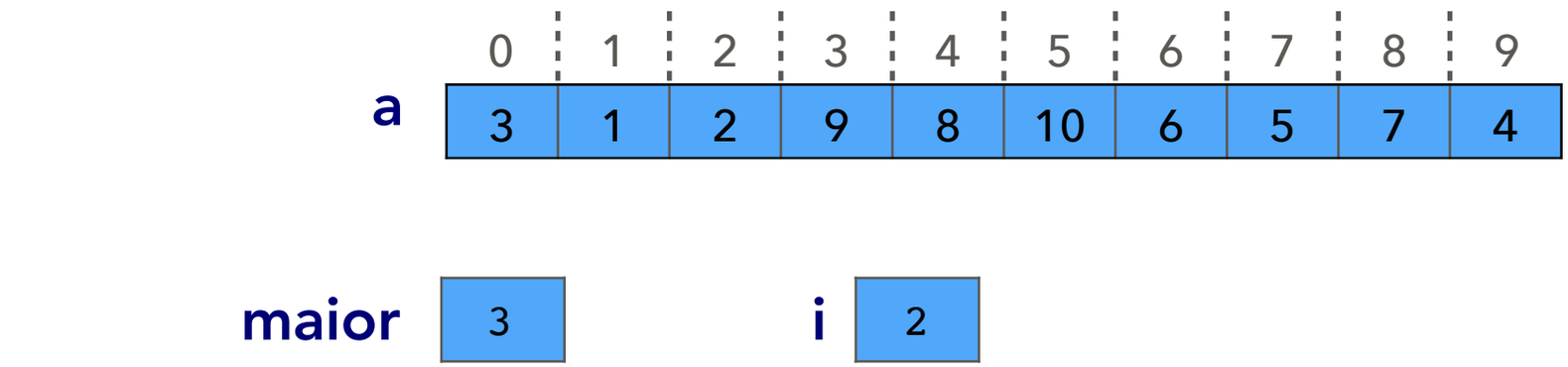
Maior elemento

```cpp
#include <iostream>

int maior_array(int a[], int tamanho) {
    int maior = a[0];
    for (int i = 1; i < tamanho;++i)
        if (a[i] > maior)
            maior = a[i];
    return maior;
}

int main() {
    int n;
    std::cin >> n;
    int a[n];
    for (int i=0 ; i<n ; ++i)
        std::cin >> a[i];
    int maior = maior_array(a,n);
    std::cout << maior << std::endl;
    return 0;
}
```



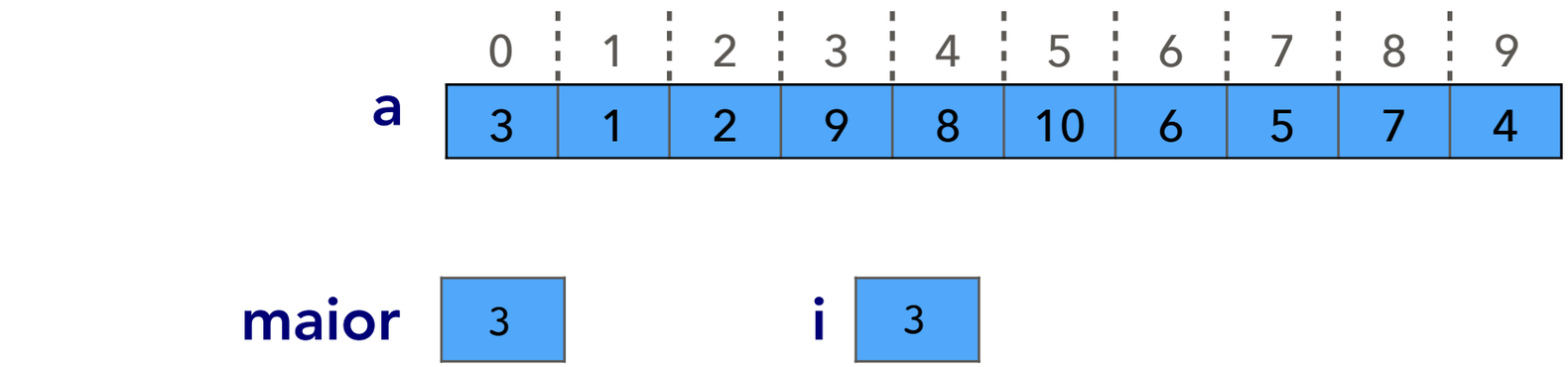
Maior elemento

```
#include <iostream>

int maior_array(int a[], int tamanho) {
    int maior = a[0];
    for (int i = 1; i < tamanho;++i)
        if (a[i] > maior)
            maior = a[i];
    return maior;
}

int main() {
    int n;
    std::cin >> n;
    int a[n];
    for (int i=0 ; i<n ; ++i)
        std::cin >> a[i];
    int maior = maior_array(a,n);
    std::cout << maior << std::endl;
    return 0;
}
```



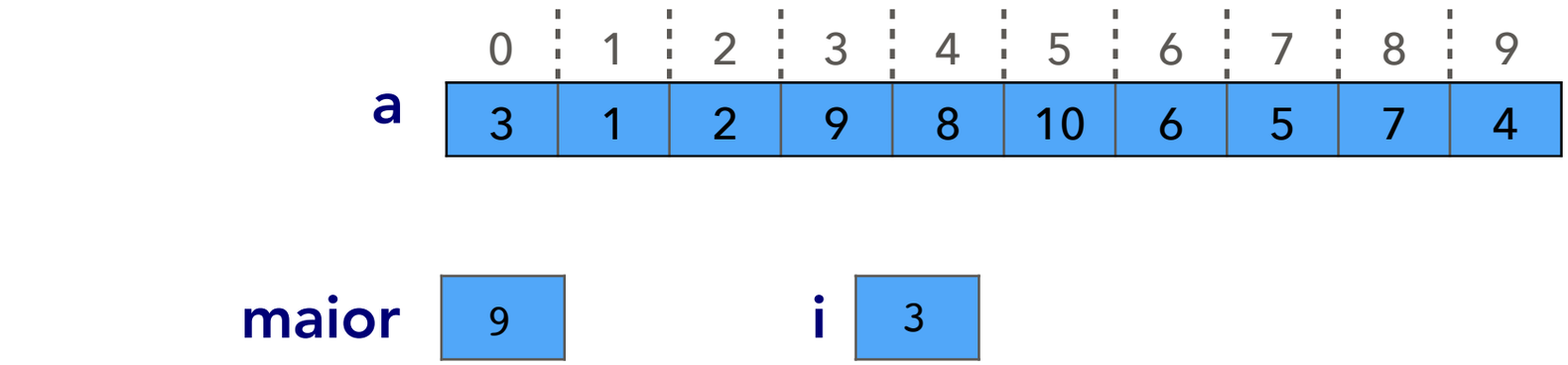
Maior elemento

```cpp
#include <iostream>

int maior_array(int a[], int tamanho) {
    int maior = a[0];
    for (int i = 1; i < tamanho;++i)
        if (a[i] > maior)
            maior = a[i];
    return maior;
}

int main() {
    int n;
    std::cin >> n;
    int a[n];
    for (int i=0 ; i<n ; ++i)
        std::cin >> a[i];
    int maior = maior_array(a,n);
    std::cout << maior << std::endl;
    return 0;
}
```



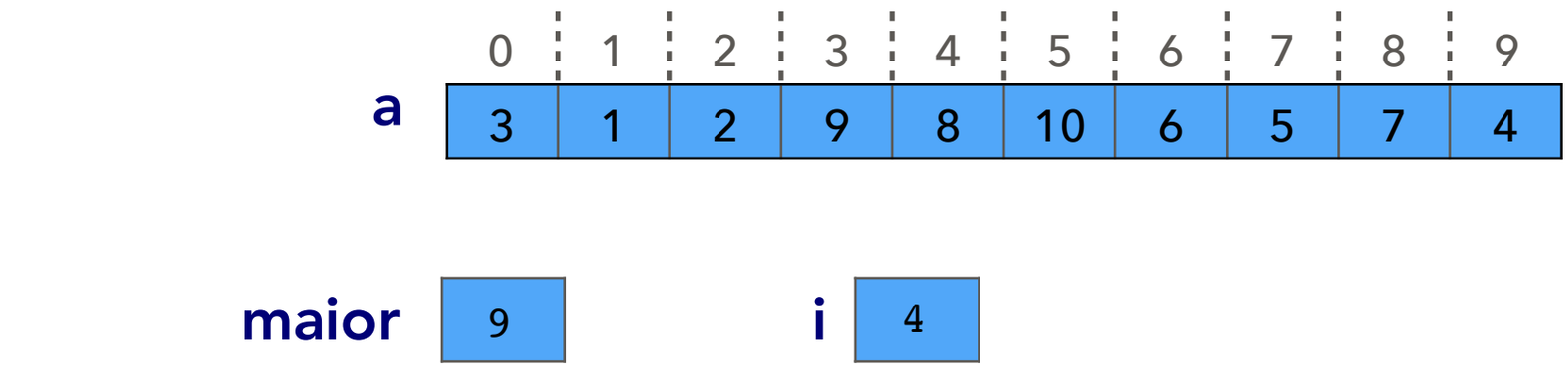
Maior elemento

```cpp
#include <iostream>

int maior_array(int a[], int tamanho) {
    int maior = a[0];
    for (int i = 1; i < tamanho;++i)
        if (a[i] > maior)
            maior = a[i];
    return maior;
}

int main() {
    int n;
    std::cin >> n;
    int a[n];
    for (int i=0 ; i<n ; ++i)
        std::cin >> a[i];
    int maior = maior_array(a,n);
    std::cout << maior << std::endl;
    return 0;
}
```



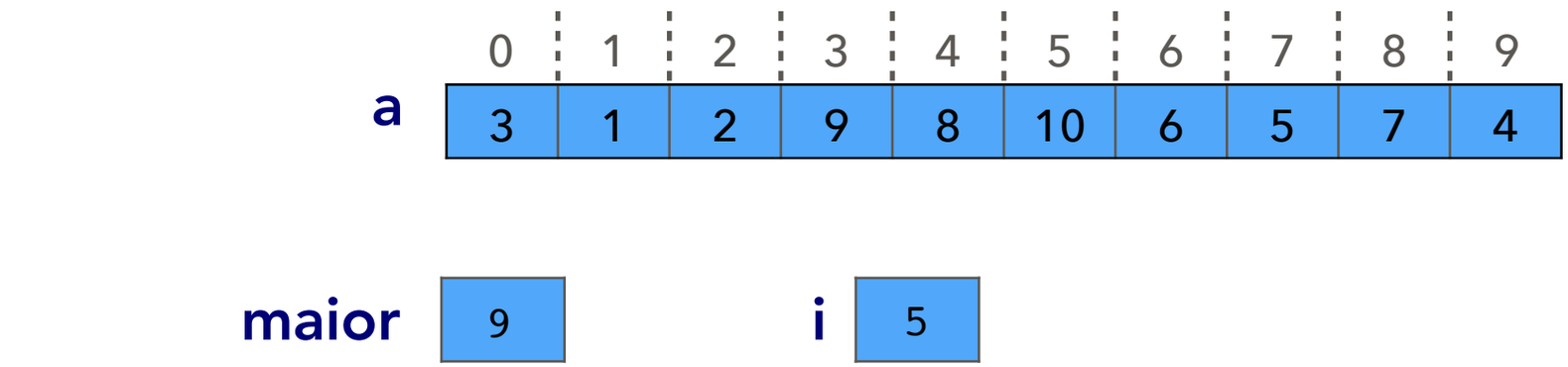
Maior elemento

```cpp
#include <iostream>

int maior_array(int a[], int tamanho) {
    int maior = a[0];
    for (int i = 1; i < tamanho;++i)
        if (a[i] > maior)
            maior = a[i];
    return maior;
}

int main() {
    int n;
    std::cin >> n;
    int a[n];
    for (int i=0 ; i<n ; ++i)
        std::cin >> a[i];
    int maior = maior_array(a,n);
    std::cout << maior << std::endl;
    return 0;
}
```



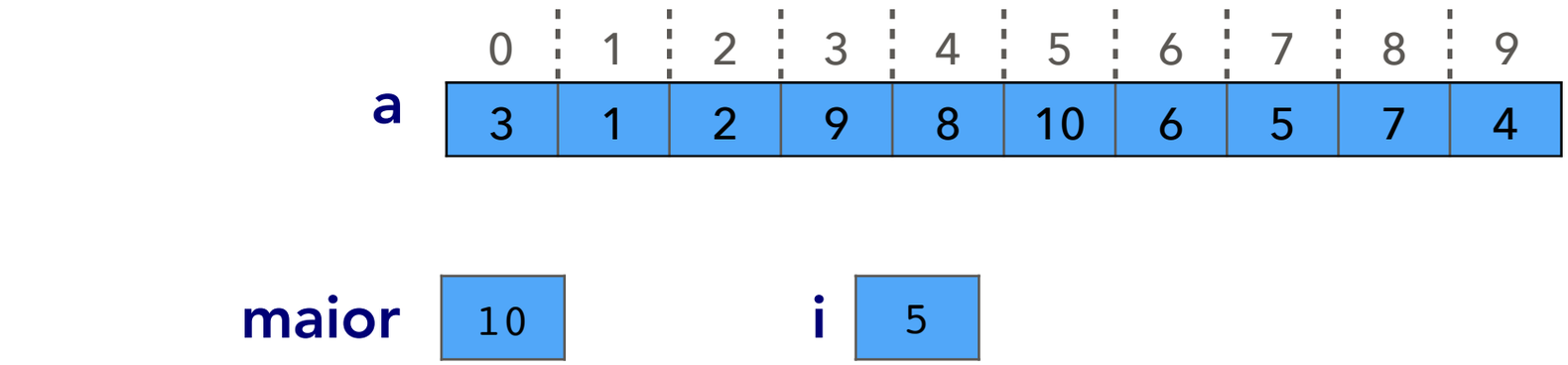
Maior elemento

```cpp
#include <iostream>

int maior_array(int a[], int tamanho) {
    int maior = a[0];
    for (int i = 1; i < tamanho;++i)
        if (a[i] > maior)
            maior = a[i];
    return maior;
}

int main() {
    int n;
    std::cin >> n;
    int a[n];
    for (int i=0 ; i<n ; ++i)
        std::cin >> a[i];
    int maior = maior_array(a,n);
    std::cout << maior << std::endl;
    return 0;
}
```



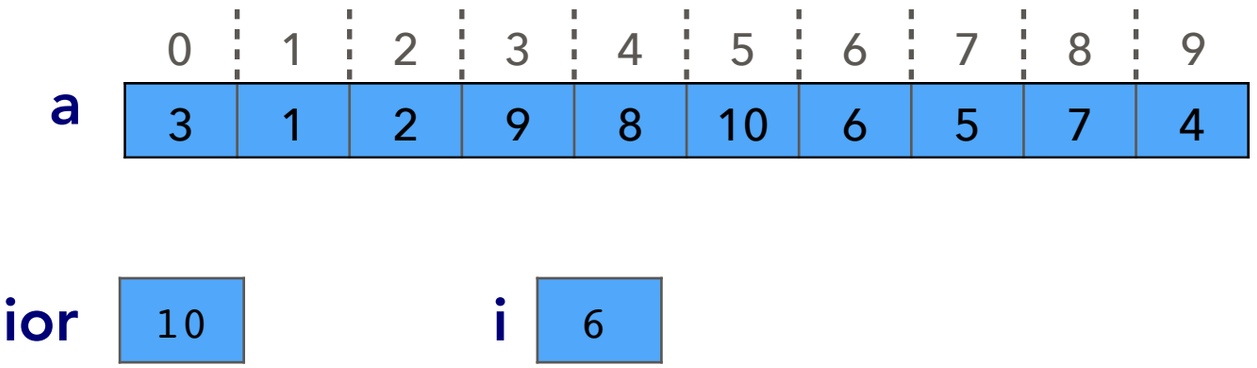
Maior elemento

```cpp
#include <iostream>

int maior_array(int a[], int tamanho) {
    int maior = a[0];
    for (int i = 1; i < tamanho;++i)
        if (a[i] > maior)
            maior = a[i];
    return maior;
}

int main() {
    int n;
    std::cin >> n;
    int a[n];
    for (int i=0 ; i<n ; ++i)
        std::cin >> a[i];
    int maior = maior_array(a,n);
    std::cout << maior << std::endl;
    return 0;
}
```



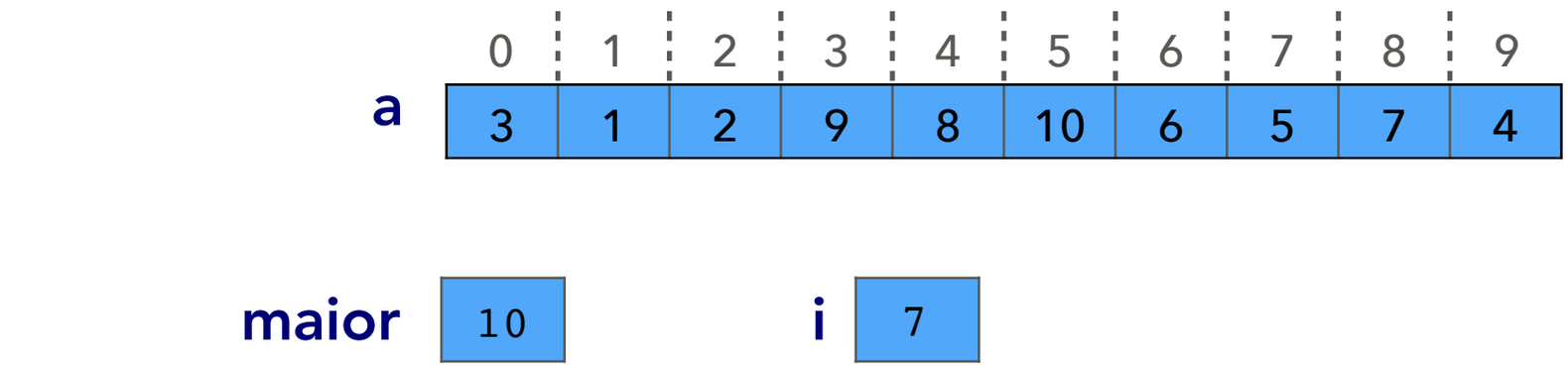
Maior elemento

```cpp
#include <iostream>

int maior_array(int a[], int tamanho) {
    int maior = a[0];
    for (int i = 1; i < tamanho;++i)
        if (a[i] > maior)
            maior = a[i];
    return maior;
}

int main() {
    int n;
    std::cin >> n;
    int a[n];
    for (int i=0 ; i<n ; ++i)
        std::cin >> a[i];
    int maior = maior_array(a,n);
    std::cout << maior << std::endl;
    return 0;
}
```



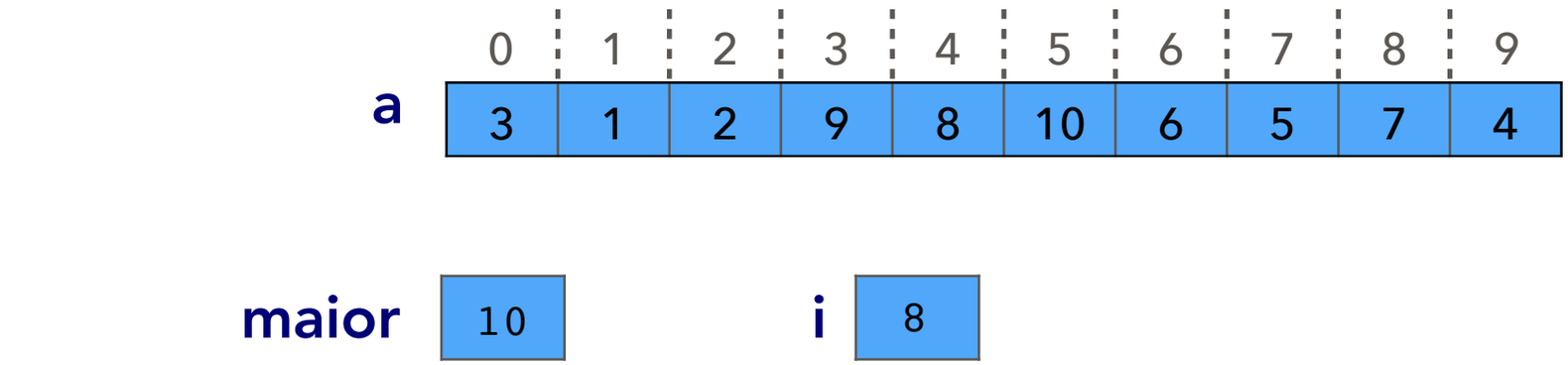
Maior elemento

```cpp
#include <iostream>

int maior_array(int a[], int tamanho) {
    int maior = a[0];
    for (int i = 1; i < tamanho;++i)
        if (a[i] > maior)
            maior = a[i];
    return maior;
}

int main() {
    int n;
    std::cin >> n;
    int a[n];
    for (int i=0 ; i<n ; ++i)
        std::cin >> a[i];
    int maior = maior_array(a,n);
    std::cout << maior << std::endl;
    return 0;
}
```



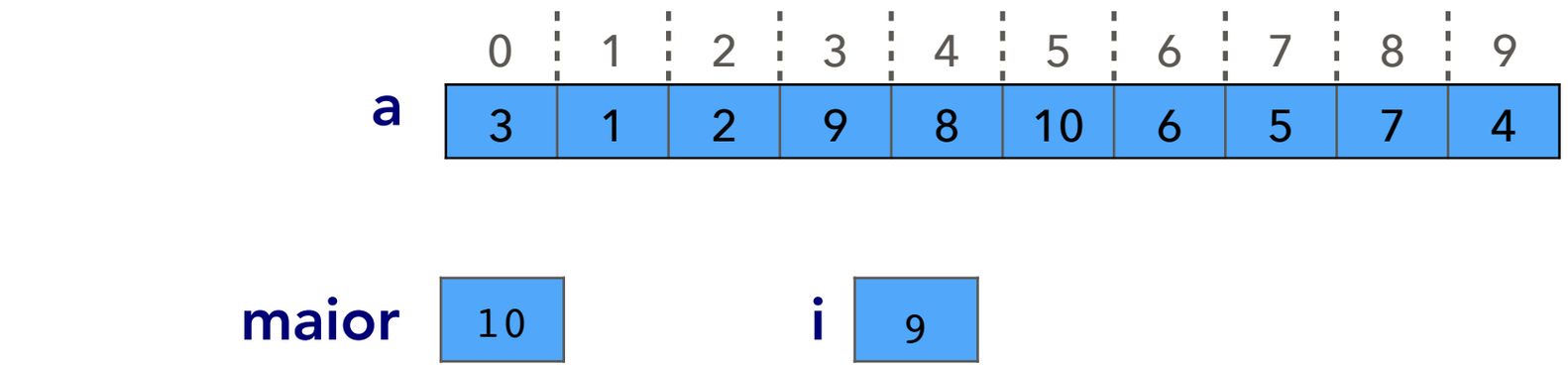
Maior elemento

```cpp
#include <iostream>

int maior_array(int a[], int tamanho) {
    int maior = a[0];
    for (int i = 1; i < tamanho;++i)
        if (a[i] > maior)
            maior = a[i];
    return maior;
}

int main() {
    int n;
    std::cin >> n;
    int a[n];
    for (int i=0 ; i<n ; ++i)
        std::cin >> a[i];
    int maior = maior_array(a,n);
    std::cout << maior << std::endl;
    return 0;
}
```



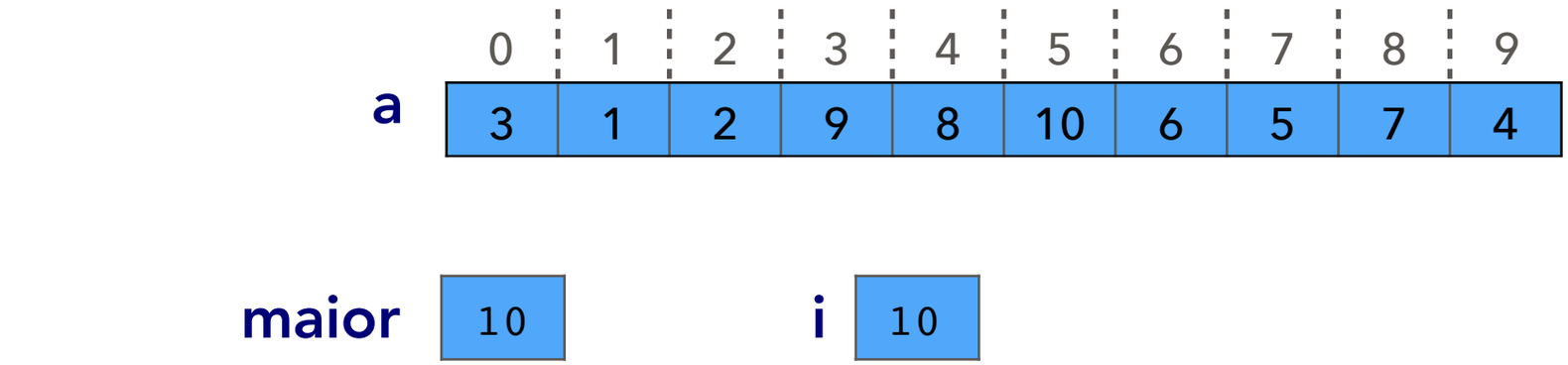
Maior elemento

```cpp
#include <iostream>

int maior_array(int a[], int tamanho) {
    int maior = a[0];
    for (int i = 1; i < tamanho;++i)
        if (a[i] > maior)
            maior = a[i];
    return maior;
}

int main() {
    int n;
    std::cin >> n;
    int a[n];
    for (int i=0 ; i<n ; ++i)
        std::cin >> a[i];
    int maior = maior_array(a,n);
    std::cout << maior << std::endl;
    return 0;
}
```



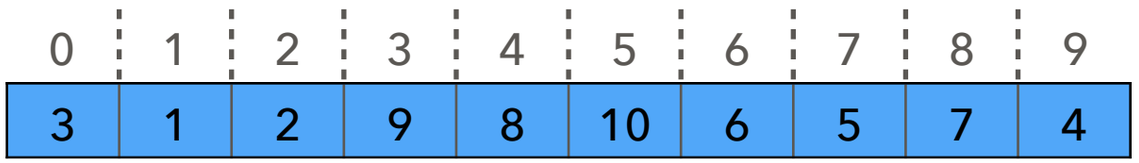
Maior elemento

```cpp
#include <iostream>

int maior_array(int a[], int tamanho) {
    int maior = a[0];
    for (int i = 1; i < tamanho;++i)
        if (a[i] > maior)
            maior = a[i];
    return maior;
}

int main() {
    int n;
    std::cin >> n;
    int a[n];
    for (int i=0 ; i<n ; ++i)
        std::cin >> a[i];
    int maior = maior_array(a,n);
    std::cout << maior << std::endl;
    return 0;
}
```



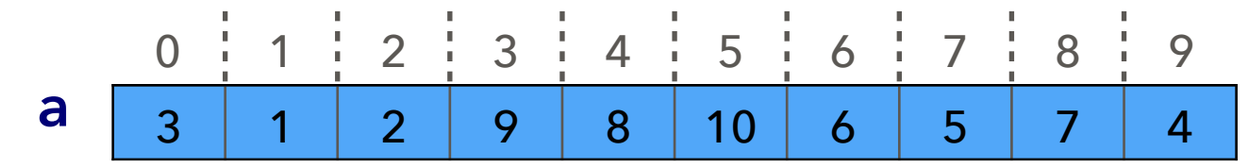
Maior elemento

```cpp
#include <iostream>

int maior_array(int a[], int tamanho) {
    int maior = a[0];
    for (int i = 1; i < tamanho;++i)
        if (a[i] > maior)
            maior = a[i];
    return maior;
}

int main() {
    int n;
    std::cin >> n;
    int a[n];
    for (int i=0 ; i<n ; ++i)
        std::cin >> a[i];
    int maior = maior_array(a,n);
    std::cout << maior << std::endl;
    return 0;
}
```



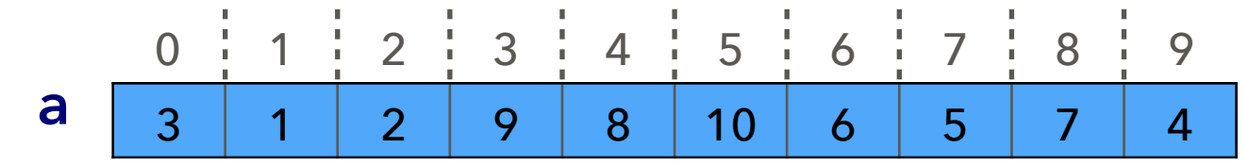
Maior elemento

```
#include <iostream>

int maior_array(int a[], int tamanho) {
    int maior = a[0];
    for (int i = 1; i < tamanho;++i)
        if (a[i] > maior)
            maior = a[i];
    return maior;
}

int main() {
    int n;
    std::cin >> n;
    int a[n];
    for (int i=0 ; i<n ; ++i)
        std::cin >> a[i];
    int maior = maior_array(a,n);
    std::cout << maior << std::endl;
    return 0;
}
```



Maior elemento

```cpp
#include <iostream>

int maior_array(int a[], int tamanho) {
    int maior = a[0];
    for (int i = 1; i < tamanho;++i)
        if (a[i] > maior)
            maior = a[i];
    return maior;
}

int main() {
    int n;
    std::cin >> n;
    int a[n];
    for (int i=0 ; i<n ; ++i)
        std::cin >> a[i];
    int maior = maior_array(a,n);
    std::cout << maior << std::endl;
    return 0;
}
```



Parâmetro de funções

- Passagem por valor
- Parâmetro = Variável local
- Escopos diferentes
- Não afetam variáveis de outras funções

Parâmetro de funções

- Passagem por valor
- Parâmetro = Variável local
- Escopos diferentes
- Não afetam variáveis de outras funções

```
int equacao(int a, int b){
  return 2*a+b;
}
```

Parâmetro de funções

- Passagem por valor
- Parâmetro = Variável local
- Escopos diferentes
- Não afetam variáveis de outras funções

```
int equacao(int a, int b){
  return 2*a+b;
}
```

```
/* ... */
a = 10;
b = 20;
x = 30,
y = 40;
c = equacao(a,b);
d = equacao(b,a);
e = equacao(x,y);
f = equacao(a+x+1,y+b+2);
/*...*/
```

Parâmetro de funções

- Passagem por valor
- Parâmetro = Variável local
- Escopos diferentes
- Não afetam variáveis de outras funções

```
int equacao(int a, int b){
  return 2*a+b;
}
```

```
/* ... */
a = 10;
b = 20;
x = 30,
y = 40;
c = equacao(a,b);
d = equacao(b,a);
e = equacao(x,y);
f = equacao(a+x+1,y+b+2);
/*...*/
```

Parâmetro de funções

- Passagem por valor
- Parâmetro = Variável local
- Escopos diferentes
- Não afetam variáveis de outras funções

```
int equacao(int a, int b){
  return 2*a+b;
}
```

a=10 b=20

```
/* ... */
a = 10;
b = 20;
x = 30,
y = 40;
c = equacao(a,b);
d = equacao(b,a);
e = equacao(x,y);
f = equacao(a+x+1,y+b+2);
/*...*/
```

Parâmetro de funções

- Passagem por valor
- Parâmetro = Variável local
- Escopos diferentes
- Não afetam variáveis de outras funções

```
int equacao(int a, int b){
  return 2*a+b;
}
```

a=10 b=20

Retorna 40

```
/* ... */
a = 10;
b = 20;
x = 30,
y = 40;
c = equacao(a,b);
d = equacao(b,a);
e = equacao(x,y);
f = equacao(a+x+1,y+b+2);
/*...*/
```

Parâmetro de funções

- Passagem por valor
- Parâmetro = Variável local
- Escopos diferentes
- Não afetam variáveis de outras funções

```
int equacao(int a, int b){
  return 2*a+b;
}
```

```
/* ... */
a = 10;
b = 20;
x = 30,
y = 40;
c = equacao(a,b);
d = equacao(b,a);
e = equacao(x,y);
f = equacao(a+x+1,y+b+2);
/*...*/
```

Parâmetro de funções

- Passagem por valor
- Parâmetro = Variável local
- Escopos diferentes
- Não afetam variáveis de outras funções

```
int equacao(int a, int b){
  return 2*a+b;
}
```

```
/* ... */
a = 10;
b = 20;
x = 30,
y = 40;
c = equacao(a,b);
d = equacao(b,a);
e = equacao(x,y);
f = equacao(a+x+1,y+b+2);
/*...*/
```

Parâmetro de funções

- Passagem por valor
- Parâmetro = Variável local
- Escopos diferentes
- Não afetam variáveis de outras funções

```
int equacao(int a, int b){
  return 2*a+b;
}
```

a=20 b=10

```
/* ... */
a = 10;
b = 20;
x = 30,
y = 40;
c = equacao(a,b);
d = equacao(b,a);
e = equacao(x,y);
f = equacao(a+x+1,y+b+2);
/*...*/
```

Parâmetro de funções

- Passagem por valor
- Parâmetro = Variável local
- Escopos diferentes
- Não afetam variáveis de outras funções

```
int equacao(int a, int b){
  return 2*a+b;
}
```

a=20 b=10

Retorna 50

```
/* ... */
a = 10;
b = 20;
x = 30,
y = 40;
c = equacao(a,b);
d = equacao(b,a);
e = equacao(x,y);
f = equacao(a+x+1,y+b+2);
/*...*/
```

Parâmetro de funções

- Passagem por valor
- Parâmetro = Variável local
- Escopos diferentes
- Não afetam variáveis de outras funções

```
int equacao(int a, int b){
  return 2*a+b;
}
```

```
/* ... */
a = 10;
b = 20;
x = 30,
y = 40;
c = equacao(a,b);
d = equacao(b,a);
e = equacao(x,y);
f = equacao(a+x+1,y+b+2);
/*...*/
```

Parâmetro de funções

- Passagem por valor
- Parâmetro = Variável local
- Escopos diferentes
- Não afetam variáveis de outras funções

```
int equacao(int a, int b){
  return 2*a+b;
}
```

```
/* ... */
a = 10;
b = 20;
x = 30,
y = 40;
c = equacao(a,b);
d = equacao(b,a);
e = equacao(x,y);
f = equacao(a+x+1,y+b+2);
/*...*/
```

Parâmetro de funções

- Passagem por valor
- Parâmetro = Variável local
- Escopos diferentes
- Não afetam variáveis de outras funções

```
int equacao(int a, int b){
  return 2*a+b;
}
```

```
/* ... */
a = 10;
b = 20;
x = 30,
y = 40;
c = equacao(a,b);
d = equacao(b,a);
e = equacao(x,y);
f = equacao(a+x+1,y+b+2);
/*...*/
```

Parâmetro de funções

- Passagem por valor
- Parâmetro = Variável local
- Escopos diferentes
- Não afetam variáveis de outras funções

```
int equacao(int a, int b){
  return 2*a+b;
}
```

```
/* ... */
a = 10;
b = 20;
x = 30,
y = 40;
c = equacao(a,b);
d = equacao(b,a);
e = equacao(x,y);
f = equacao(a+x+1,y+b+2);
/*...*/
```

Parâmetro de funções

- Passagem por valor
- Parâmetro = Variável local
- Escopos diferentes
- Não afetam variáveis de outras funções

```
int equacao(int a, int b){
  return 2*a+b;
}
```

```
/* ... */
a = 10;
b = 20;
x = 30,
y = 40;
c = equacao(a,b);
d = equacao(b,a);
e = equacao(x,y);
f = equacao(a+x+1,y+b+2);
/*...*/
```

Referência

- Nome diferente para variável existente
  - Coloque & antes do nome da variável

Referência

- Nome diferente para variável existente
  - Coloque & antes do nome da variável

```
#include <iostream>

int main(){
    int a;
    std::cin >> a; /* 20 */
    int b = a;
    int &c = a;
    /* ... */
    b = 10;
    c = 50;
    /* ... */
    return 0;
}
```

Referência

- Nome diferente para variável existente
  - Coloque & antes do nome da variável

```
#include <iostream>

int main(){
    int a;
    std::cin >> a; /* 20 */
    int b = a;
    int &c = a;
    /* ... */
    b = 10;
    c = 50;
    /* ... */
    return 0;
}
```

Referência

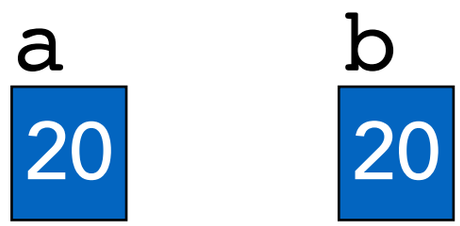
- Nome diferente para variável existente
  - Coloque & antes do nome da variável

```
#include <iostream>

int main(){
    int a;
    std::cin >> a; /* 20 */
    int b = a;
    int &c = a;
    /* ... */
    b = 10;
    c = 50;
    /* ... */
    return 0;
}
```

Referência

- Nome diferente para variável existente
  - Coloque & antes do nome da variável

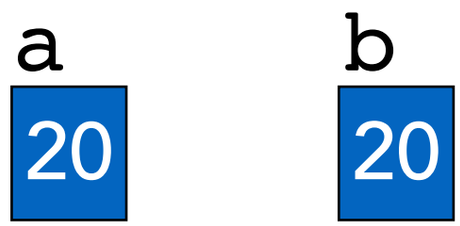
```
#include <iostream>

int main(){
    int a;
    std::cin >> a; /* 20 */
    int b = a;
    int &c = a;
    /* ... */
    b = 10;
    c = 50;
    /* ... */
    return 0;
}
```



Referência

- Nome diferente para variável existente
  - Coloque & antes do nome da variável

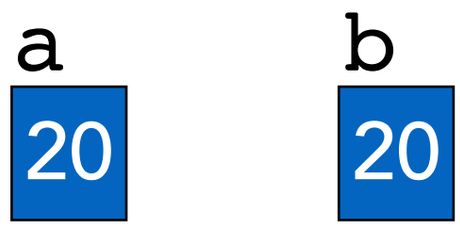
```
#include <iostream>

int main(){
    int a;
    std::cin >> a; /* 20 */
    int b = a;
    int &c = a;
    /* ... */
    b = 10;
    c = 50;
    /* ... */
    return 0;
}
```



Referência

- Nome diferente para variável existente
  - Coloque & antes do nome da variável

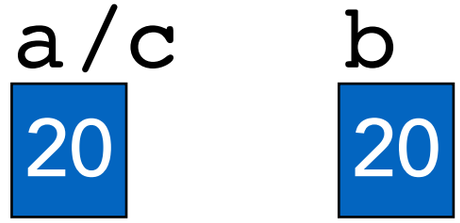
```
#include <iostream>

int main(){
    int a;
    std::cin >> a; /* 20 */
    int b = a;
    int &c = a;
    /* ... */
    b = 10;
    c = 50;
    /* ... */
    return 0;
}
```



Referência

- Nome diferente para variável existente
  - Coloque & antes do nome da variável

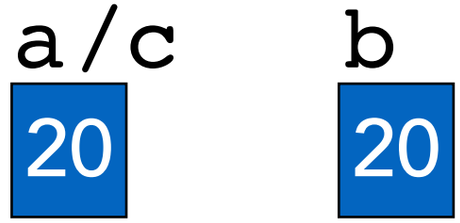
```
#include <iostream>

int main(){
    int a;
    std::cin >> a; /* 20 */
    int b = a;
    int &c = a;
    /* ... */
    b = 10;
    c = 50;
    /* ... */
    return 0;
}
```



Referência

- Nome diferente para variável existente
  - Coloque & antes do nome da variável

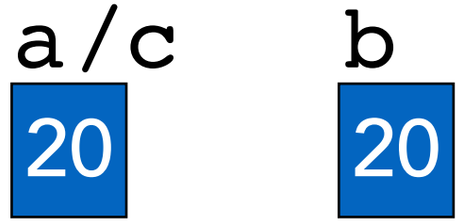
```
#include <iostream>

int main(){
    int a;
    std::cin >> a; /* 20 */
    int b = a;
    int &c = a;
    /* ... */
    b = 10;
    c = 50;
    /* ... */
    return 0;
}
```



Referência

- Nome diferente para variável existente
  - Coloque & antes do nome da variável

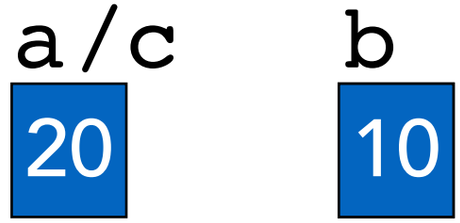
```
#include <iostream>

int main(){
    int a;
    std::cin >> a; /* 20 */
    int b = a;
    int &c = a;
    /* ... */
    b = 10;
    c = 50;
    /* ... */
    return 0;
}
```



Referência

- Nome diferente para variável existente
  - Coloque & antes do nome da variável

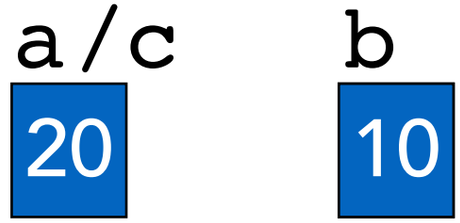
```
#include <iostream>

int main(){
    int a;
    std::cin >> a; /* 20 */
    int b = a;
    int &c = a;
    /* ... */
    b = 10;
    c = 50;
    /* ... */
    return 0;
}
```



Referência

- Nome diferente para variável existente
  - Coloque & antes do nome da variável

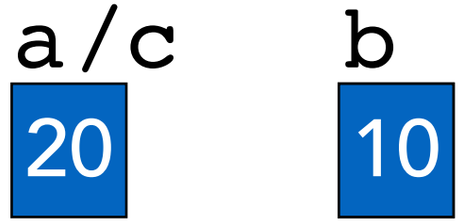
```
#include <iostream>

int main(){
    int a;
    std::cin >> a; /* 20 */
    int b = a;
    int &c = a;
    /* ... */
    b = 10;
    c = 50;
    /* ... */
    return 0;
}
```



Referência

- Nome diferente para variável existente
  - Coloque & antes do nome da variável

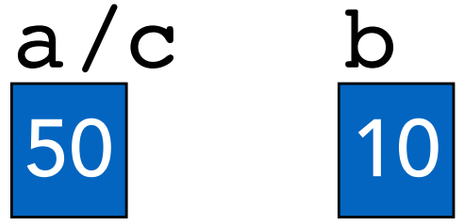
```
#include <iostream>

int main(){
    int a;
    std::cin >> a; /* 20 */
    int b = a;
    int &c = a;
    /* ... */
    b = 10;
    c = 50;
    /* ... */
    return 0;
}
```



Referência

- Nome diferente para variável existente
  - Coloque & antes do nome da variável

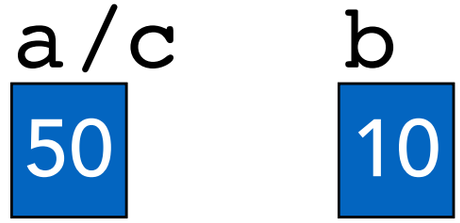
```
#include <iostream>

int main(){
    int a;
    std::cin >> a; /* 20 */
    int b = a;
    int &c = a;
    /* ... */
    b = 10;
    c = 50;
    /* ... */
    return 0;
}
```



Referência

- Nome diferente para variável existente
  - Coloque & antes do nome da variável

```
#include <iostream>

int main(){
    int a;
    std::cin >> a; /* 20 */
    int b = a;
    int &c = a;
    /* ... */
    b = 10;
    c = 50;
    /* ... */
    return 0;
}
```



Parâmetros por referência

```cpp
#include <iostream>

void incrementa(int &x){
    x = x + 1;
}

int main(){
    int n;
    std::cin >> n;
    if (n % 2 == 1)
    {
        incrementa(n);
    }
    std::cout << n << std::endl;
    return 0;
}
```

Parâmetro de funções

Parâmetro de funções

- **Cuidado** com *arrays*!
- Parâmetro referencia array único

```
void f1(int x[], int valor, int indice){
  x[indice]=valor;
}
```

```
f1(a,12,3);
```

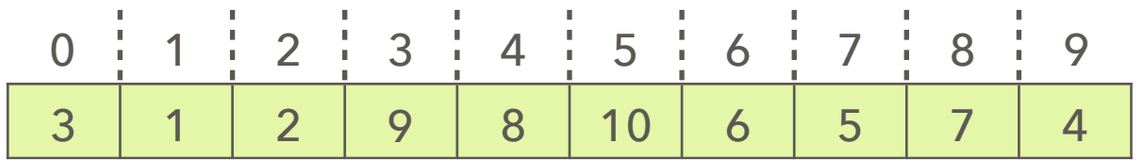
Parâmetro de funções

- **Cuidado** com *arrays*!
- Parâmetro referencia array único

```
void f1(int x[], int valor, int indice){
  x[indice]=valor;
}
```

```
f1(a,12,3);
```

a



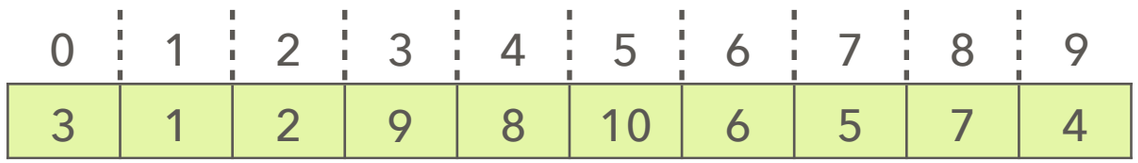
Parâmetro de funções

- **Cuidado** com *arrays*!
- Parâmetro referencia array único

```
void f1(int x[], int valor, int indice){
  x[indice]=valor;
}
```

```
f1(a,12,3);
```

a

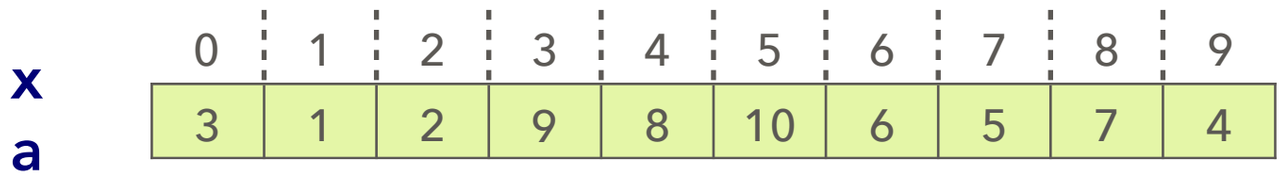


Parâmetro de funções

- **Cuidado** com *arrays*!
- Parâmetro referencia array único

```
void f1(int x[], int valor, int indice){
  x[indice]=valor;
}
```

```
f1(a,12,3);
```

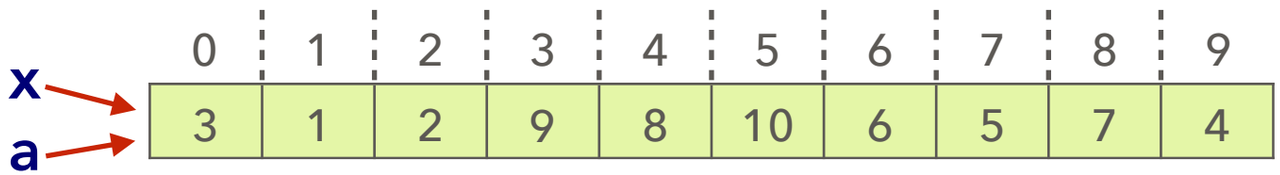


Parâmetro de funções

- **Cuidado** com *arrays*!
- Parâmetro referencia array único

```
void f1(int x[], int valor, int indice){
  x[indice]=valor;
}
```

```
f1(a,12,3);
```

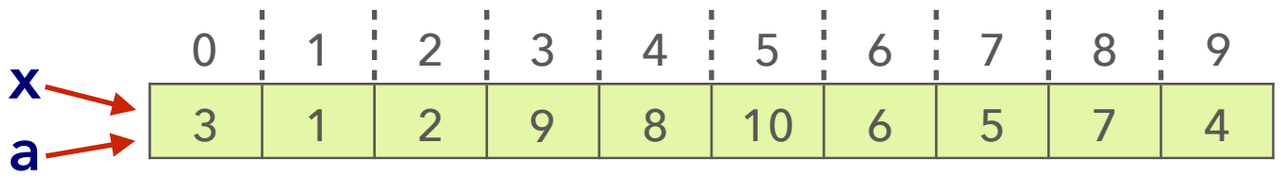


Parâmetro de funções

- **Cuidado** com *arrays*!
- Parâmetro referencia array único

```
void f1(int x[], int valor, int indice){
  x[indice]=valor;
}
```

```
f1(a,12,3);
```

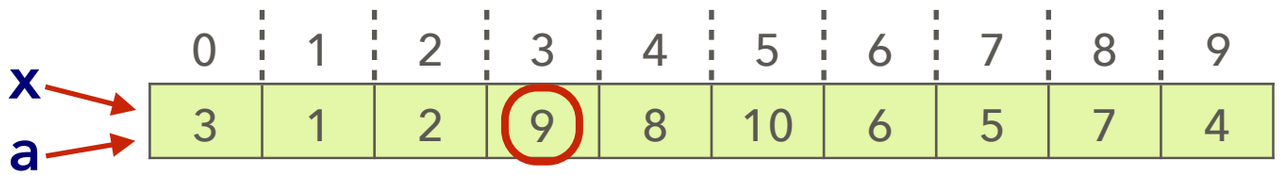


Parâmetro de funções

- **Cuidado** com *arrays*!
- Parâmetro referencia array único

```
void f1(int x[], int valor, int indice){
  x[indice]=valor;
}
```

```
f1(a,12,3);
```

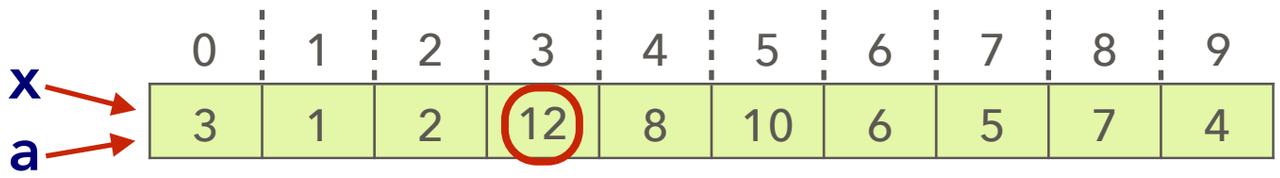


Parâmetro de funções

- **Cuidado** com *arrays*!
- Parâmetro referencia array único

```
void f1(int x[], int valor, int indice){
  x[indice]=valor;
}
```

```
f1(a,12,3);
```

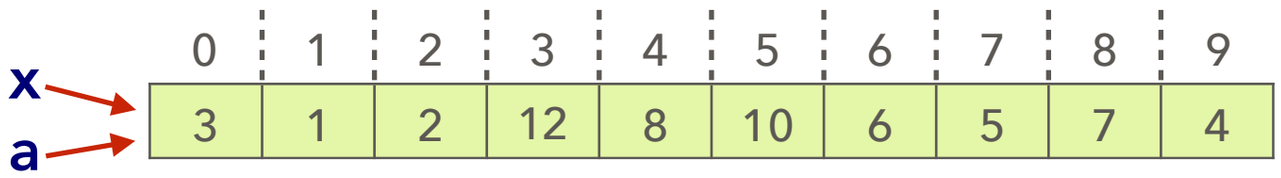


Parâmetro de funções

- **Cuidado** com *arrays*!
- Parâmetro referencia array único

```
void f1(int x[], int valor, int indice){
  x[indice]=valor;
}
```

```
f1(a,12,3);
```



Comentários

- Ajuda a entender o código
  - Documentação
- Ignorado pelo compilador
- Em C++

```
// uma linha
/* Várias linhas de
comentário. Importante
para explicar detalhes
do algoritmo.
*/
```

```cpp
#include <iostream>
#include <iomanip> // Para formatar casas decimais

/*
 Função para calcular tempo de deslocamento
 Recebe como parâmetros a velocidade, em km/h
 e a distância a ser percorrida, em km.
 Retorna o tempo de deslocamento em horas.

*/
double tempo(double velocidade, double distancia) {
  double deslocalmento = distancia/velocidade;
  return deslocalmento;
}

int main() {
  double velocidade, // em km/h
         distancia,  // em kilometros;
         tempo_deslocamento; // em horas
  std::cin >> velocidade;
  std::cin >> distancia;
  tempo_deslocamento = tempo(velocidade, distancia);
  std::cout << "Tempo de deslocamento:"
            << std::fixed << std::setprecision(2)
            <<  tempo_deslocamento << std::endl;
  return 0;
}
```

Comentários

- Ajuda a entender o código
  - Documentação
- Ignorado pelo compilador
- Em C++

```
// uma linha
/* Várias linhas de
comentário. Importante
para explicar detalhes
do algoritmo.
*/
```

```
#include <iostream>
#include <iomanip> // Para formatar casas decimais

/*
 Função para calcular tempo de deslocamento
 Recebe como parâmetros a velocidade, em km/h
 e a distância a ser percorrida, em km.
 Retorna o tempo de deslocamento em horas.

*/
double tempo(double velocidade, double distancia) {
  double deslocalmento = distancia/velocidade;
  return deslocalmento;
}

int main() {
  double velocidade, // em km/h
         distancia,  // em kilometros;
         tempo_deslocamento; // em horas
  std::cin >> velocidade;
  std::cin >> distancia;
  tempo_deslocamento = tempo(velocidade, distancia);
  std::cout << "Tempo de deslocamento:"
            << std::fixed << std::setprecision(2)
            <<  tempo_deslocamento << std::endl;
  return 0;
}
```

Comentários

- Ajuda a entender o código
  - Documentação
- Ignorado pelo compilador
- Em C++

```
// uma linha
/* Várias linhas de
comentário. Importante
para explicar detalhes
do algoritmo.
*/
```

```
#include <iostream>
#include <iomanip> // Para formatar casas decimais

/*
 Função para calcular tempo de deslocamento
 Recebe como parâmetros a velocidade, em km/h
 e a distância a ser percorrida, em km.
 Retorna o tempo de deslocamento em horas.

*/
double tempo(double velocidade, double distancia) {
  double deslocalmento = distancia/velocidade;
  return deslocalmento;
}

int main() {
  double velocidade, // em km/h
         distancia,  // em kilometros;
         tempo_deslocamento; // em horas
  std::cin >> velocidade;
  std::cin >> distancia;
  tempo_deslocamento = tempo(velocidade, distancia);
  std::cout << "Tempo de deslocamento:"
            << std::fixed << std::setprecision(2)
            <<  tempo_deslocamento << std::endl;
  return 0;
}
```

Comentários

- Ajuda a entender o código
  - Documentação
- Ignorado pelo compilador
- Em C++

```
// uma linha
/* Várias linhas de
comentário. Importante
para explicar detalhes
do algoritmo.
*/
```

```
#include <iostream>
#include <iomanip> // Para formatar casas decimais

/*
 Função para calcular tempo de deslocamento
 Recebe como parâmetros a velocidade, em km/h
 e a distância a ser percorrida, em km.
 Retorna o tempo de deslocamento em horas.

*/
double tempo(double velocidade, double distancia) {
  double deslocalmento = distancia/velocidade;
  return deslocalmento;
}

int main() {
  double velocidade, // em km/h
         distancia,  // em kilometros;
         tempo_deslocamento; // em horas
  std::cin >> velocidade;
  std::cin >> distancia;
  tempo_deslocamento = tempo(velocidade, distancia);
  std::cout << "Tempo de deslocamento:"
            << std::fixed << std::setprecision(2)
            <<  tempo_deslocamento << std::endl;
  return 0;
}
```

Prática

- Dúvidas?

Programação

C++

Aula 03

Jorgiano Vidal