# Algoritmia e Programação

## Funções.

# Conceito de função

- Uma função é um bloco de instruções que:
  - pode receber um conjunto de dados,
  - implementa uma funcionalidade específica, e
  - pode retornar um resultado.

# Vantagens das funções

- Promove a reutilização de código
    - Escreve-se uma vez, invoca-se múltiplas vezes.
- Isola pormenores da operação, promovendo a modularidade.
- Melhora a manutenção do código
    - A correcção de erros ou a actualização do código só se realiza num único ponto do programa.

# Sintaxe de uma função em C

```
tipo-de-retorno nome-da-função(declaração de parâmetros)
{
   declarações e instruções
}
```

# Sintaxe de uma função em C

- Exemplo:

```
int maximo(int a, int b)
{
   if(a > b)
      return a;
   else
      return b;
}
```

# Passagem de argumentos

- A passagem de argumentos (valores) para uma função funciona através da **cópia de valores**.
    - Quem invoca a função disponibiliza valores (**argumentos**) à função.
        - Esses valores são copiados para uma área de memória, acessível à função através dos **parâmetros** da função.
- A função opera as cópias que lhe foram disponibilizadas.
    - A função **não altera** os valores originais!

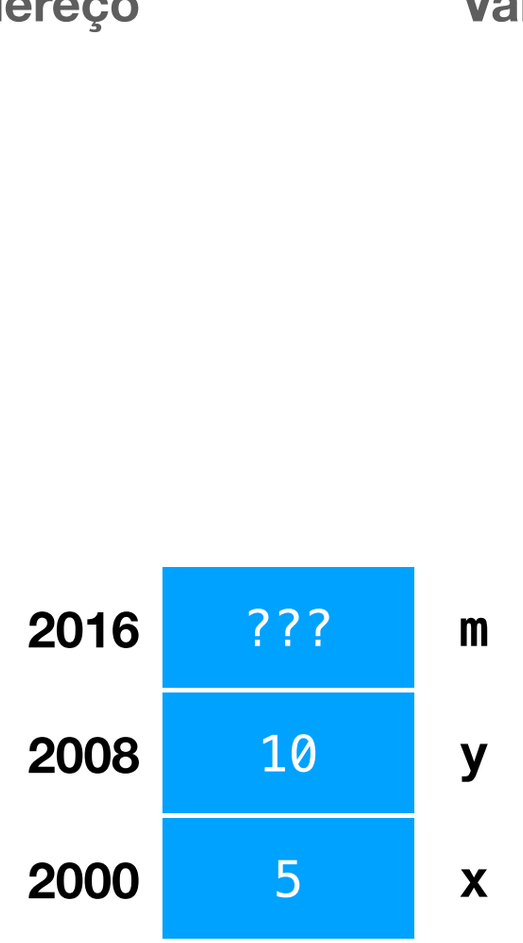
# Passagem por valor

# Passagem de argumentos por valor

```
int maximo(int a, int b)
{
   if(a > b)
      return a;
   else
      return b;
}

int main() {
   int x, y, m;
   x = 5;
   y = 10;

   m = maximo(x, y);
   ...
```

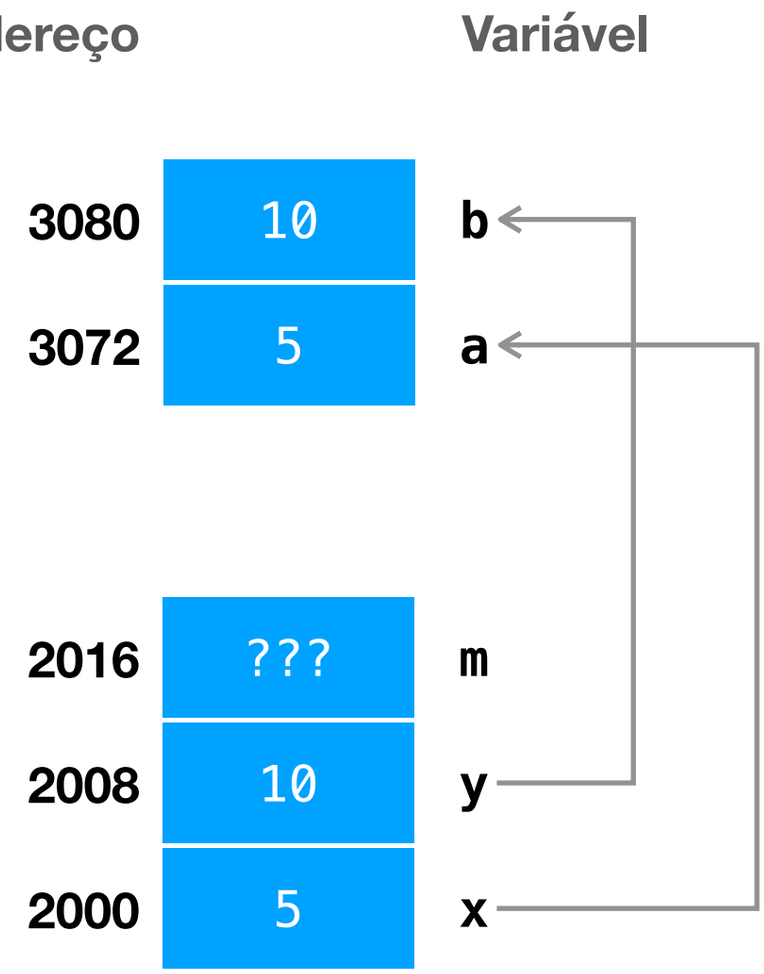
# Passagem de argumentos por valor

```
int maximo(int a, int b)
{
   if(a > b)
      return a;
   else
      return b;
}

int main() {
   int x, y, m;
   x = 5;
   y = 10;

   m = maximo(x, y);
   ...
```



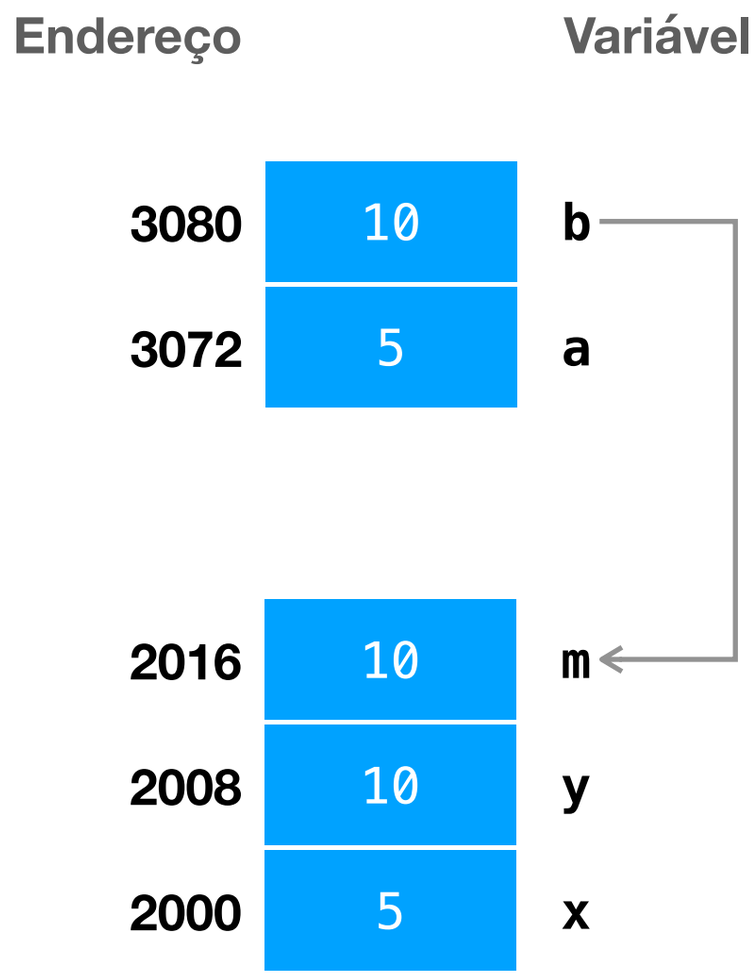
# Passagem de argumentos por valor

```
int maximo(int a, int b)
{
   if(a > b)
      return a;
   else
      return b;
}

int main() {
   int x, y, m;
   x = 5;
   y = 10;

   m = maximo(x, y);
   ...
```



# Passagem de argumentos por valor

```
int maximo(int a, int b)
{
   if(a > b)
      return a;
   else
      return b;
}

int main() {
   int x, y, m;
   x = 5;
   y = 10;

   m = maximo(x, y);
   ...
```



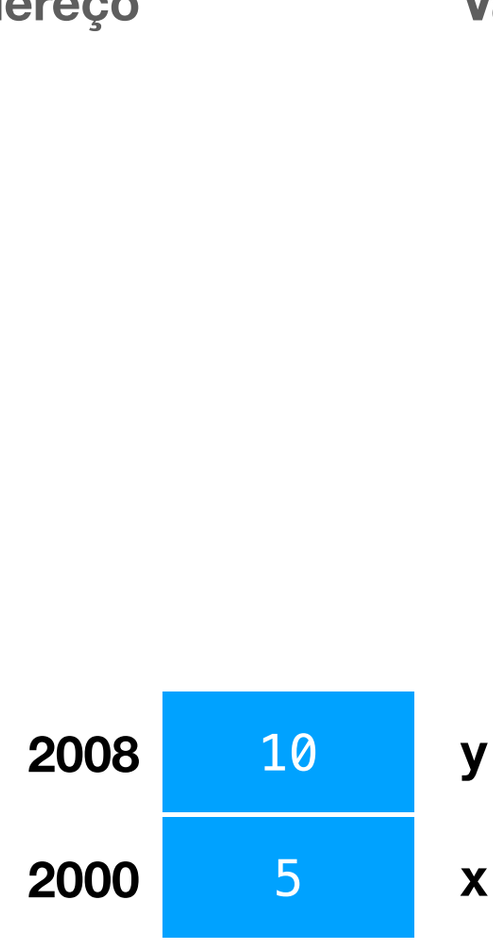
# Passagem por referência

# Passagem de argumentos por referência

```
void trocar(int *a, int *b)
{
   int aux;
   aux = *a;
   *a = *b;
   *b = aux;
}

int main() {
   int x, y;
   x = 5;
   y = 10;

   trocar(&x, &y);
   ...
```

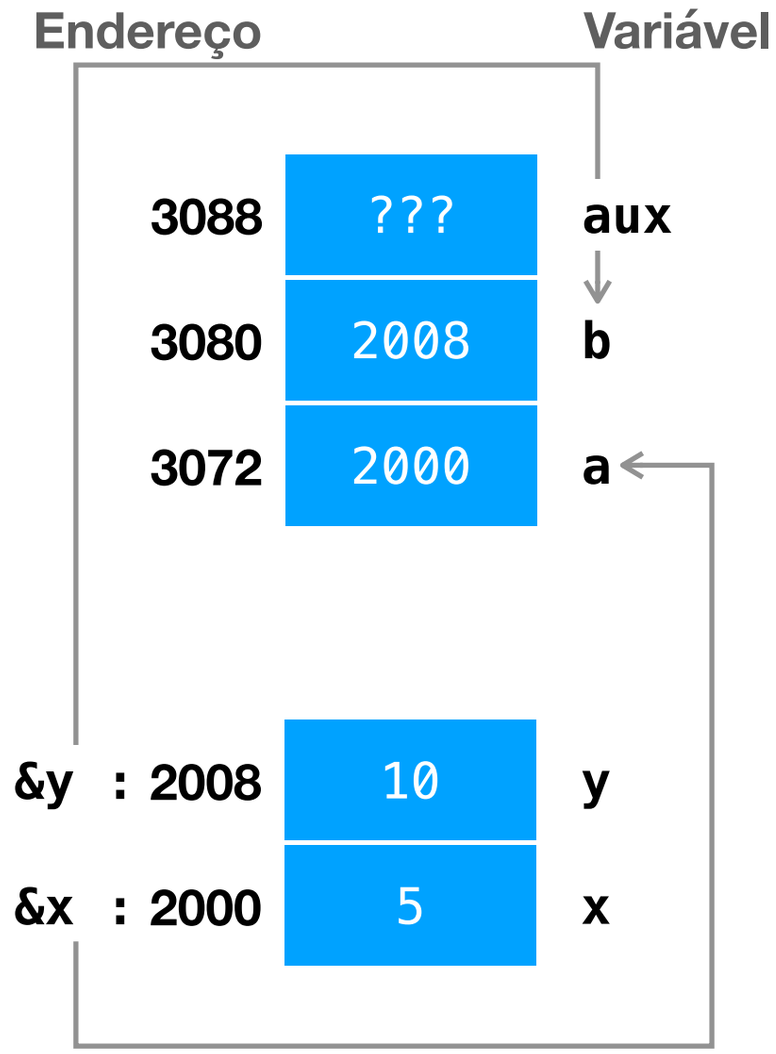
# Passagem de argumentos por referência

```
void trocar(int *a, int *b)
{
   int aux;
   aux = *a;
   *a = *b;
   *b = aux;
}

int main() {
   int x, y;
   x = 5;
   y = 10;

   trocar(&x, &y);
   ...
```



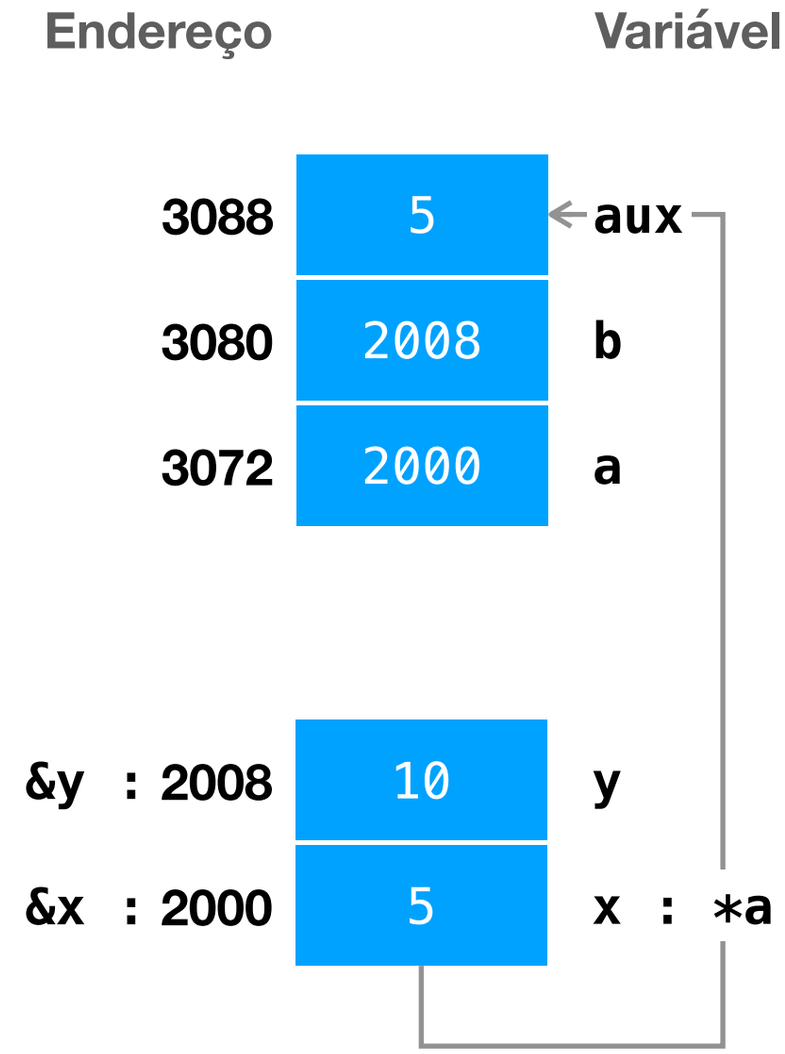
# Passagem de argumentos por referência

```
void trocar(int *a, int *b)
{
   int aux;
   aux = *a;
   *a = *b;
   *b = aux;
}

int main() {
   int x, y;
   x = 5;
   y = 10;

   trocar(&x, &y);
   ...
```



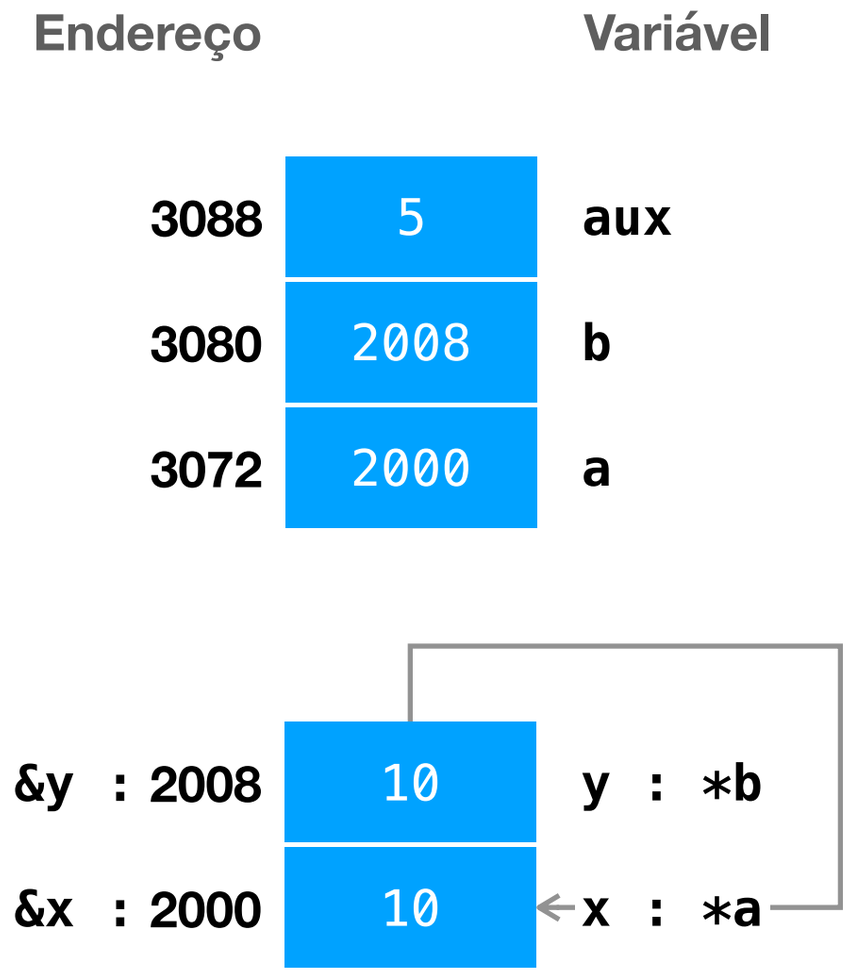
# Passagem de argumentos por referência

```
void trocar(int *a, int *b)
{
   int aux;
   aux = *a;
   *a = *b;
   *b = aux;
}

int main() {
   int x, y;
   x = 5;
   y = 10;

   trocar(&x, &y);
   ...
```



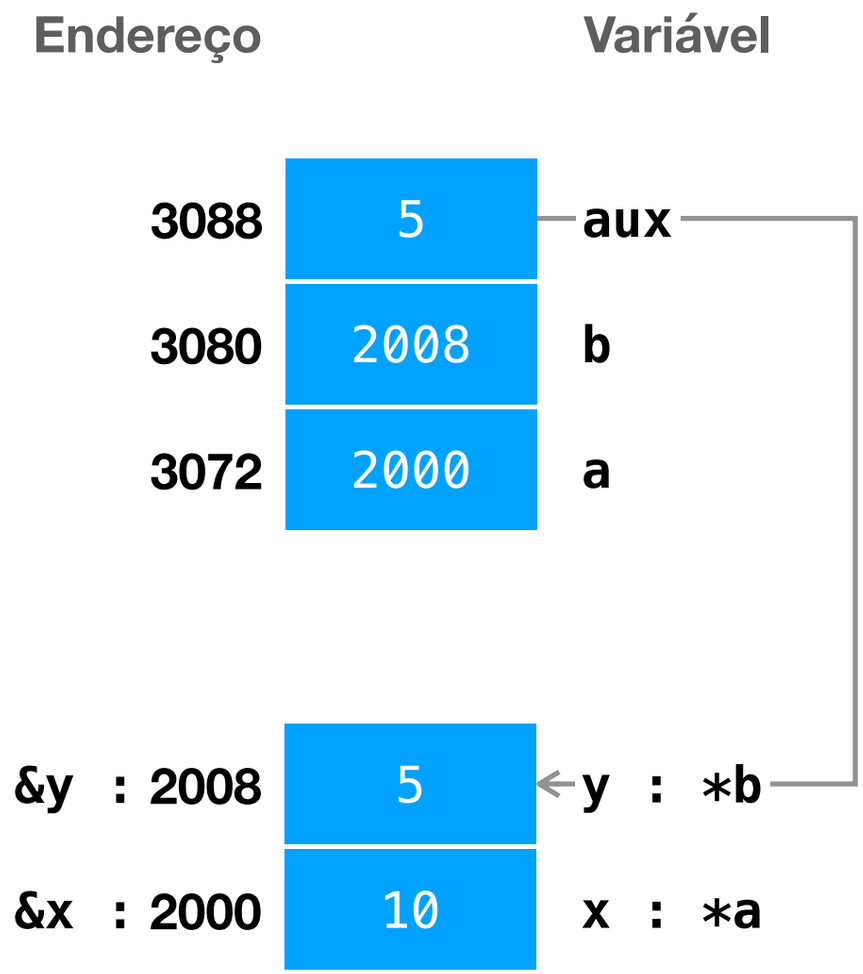
# Passagem de argumentos por referência

```
void trocar(int *a, int *b)
{
   int aux;
   aux = *a;
   *a = *b;
   *b = aux;
}

int main() {
   int x, y;
   x = 5;
   y = 10;

   trocar(&x, &y);
   ...
```



# Passagem de argumentos por referência

```
void trocar(int *a, int *b)
{
   int aux;
   aux = *a;
   *a = *b;
   *b = aux;
}

int main() {
   int x, y;
   x = 5;
   y = 10;

   trocar(&x, &y);
   ...
```



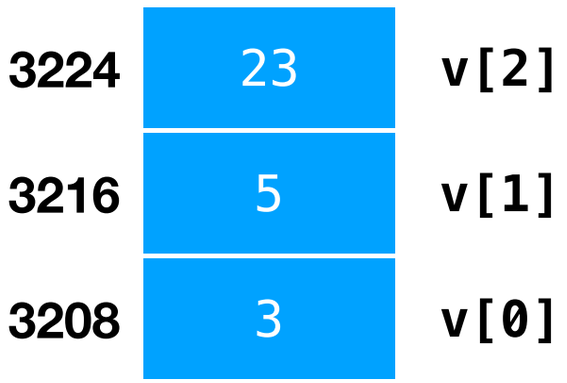
# Passagem de vectores

- O nome de um vector representa o endereço base desse vector (referência para o vector).

- A passagem de um vector para uma função é **sempre por referência**!

- Uma função pode alterar um vector que seja passado por argumento!

```
v = &v[0] = 3208
```



# Passagem de vectores

- O parâmetro que recebe o vector é assinalado com `[]`.
    - A função aceita assim um vector com dimensão indeterminada.
- Quando se passa um vector para uma função, deve ser também passada a dimensão do vector.
    - Desta forma a função pode ser desenvolvida para vectores de qualquer dimensão.

```
void ordenar(int v[], int n)
{
   // Algum código...
   if(v[i] > v[j])
   ...
}

int main() {
   int vec[20];
   // Algum código...

   ordenar(vec, 20);
   ...
```

# Exemplo: ordenação de um vector

```
void ordenar(int v[], int n)
{
   int i, j, aux;

   for(i = 0; i < n-1; i++) {
      for(j = i+1; j < n; j++) {
         if(v[i] > v[j]) {
            aux = v[i];
            v[i] = v[j];
            v[j] = aux;
         }
      }
   }
}
```