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.



**APROG - LETI**

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.



**APROG - LETI**

Sintaxe de uma função em C

*tipo-de-retorno* ***nome-da-função****(declaração de parâmetros) {*



*declarações e instruções*

*}*



**APROG - LETI**

Sintaxe de uma função em C

* Exemplo:

int **maximo**(int a, int b)



{

if(a > b)

return a;

else

return b;

}



**APROG - LETI**

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!



**APROG - LETI**

Passagem por valor



**APROG - LETI**

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);

...



**APROG - LETI**

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);

...

**Endereço** **Variável**



**2016** ??? **m**



**2008** 10 **y**



**2000** 5 **x**



**APROG - LETI**

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;

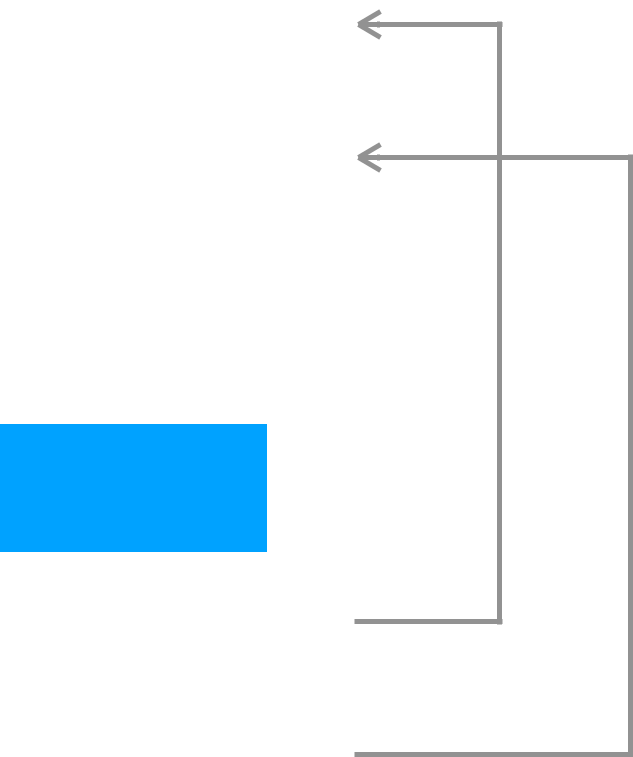
y = 10;

m = maximo(x, y);

...

**Endereço** **Variável**

|  |  |  |  |
| --- | --- | --- | --- |
| **3080** | 10 | **b** |  |
| **3072** |  | **a** |  |
|  |  |
| 5 |  |
|  |  |  |  |



**2016** ??? **m**



**2008** 10 **y**



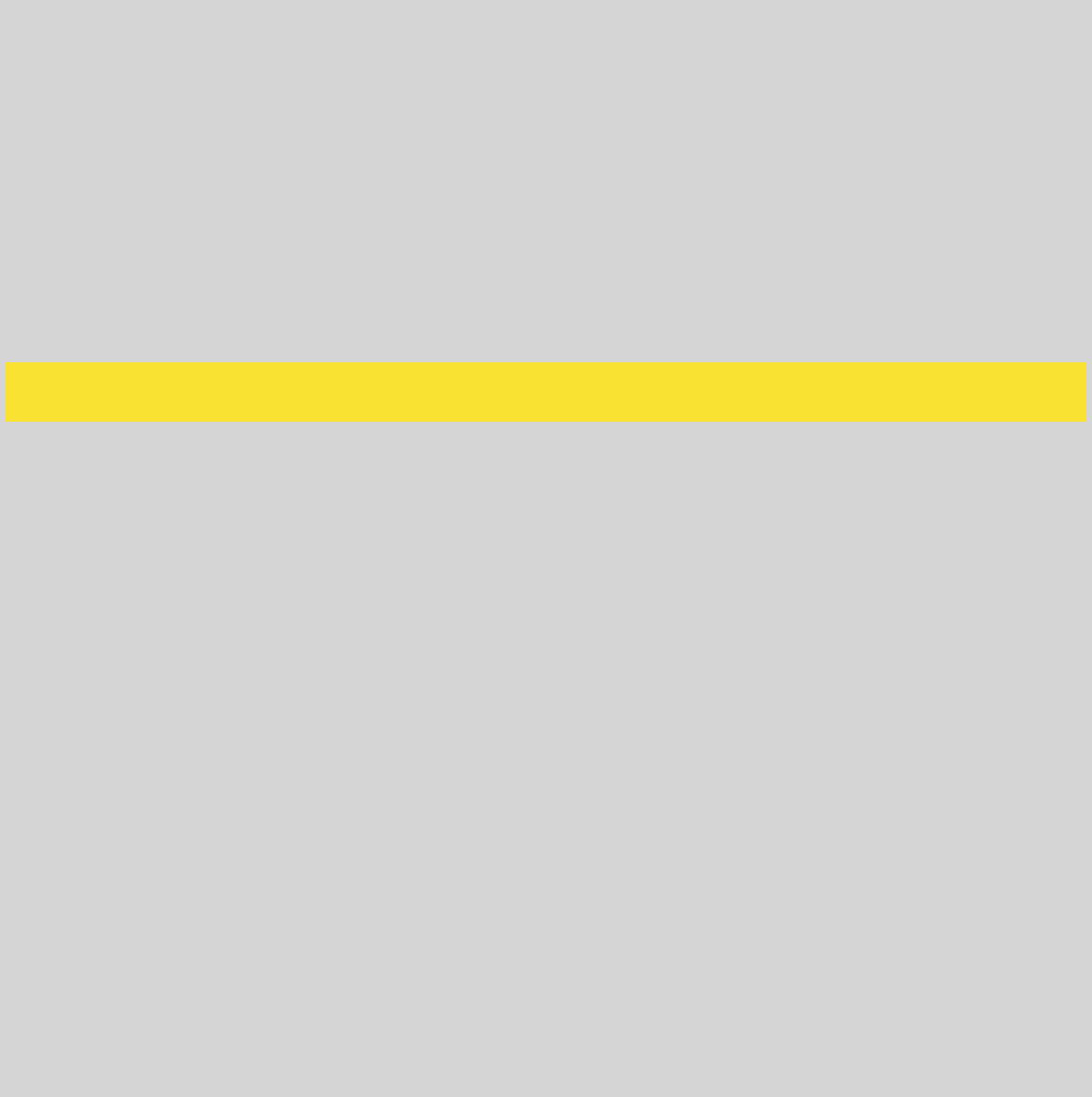
**2000** 5 **x**



**APROG - LETI**

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;

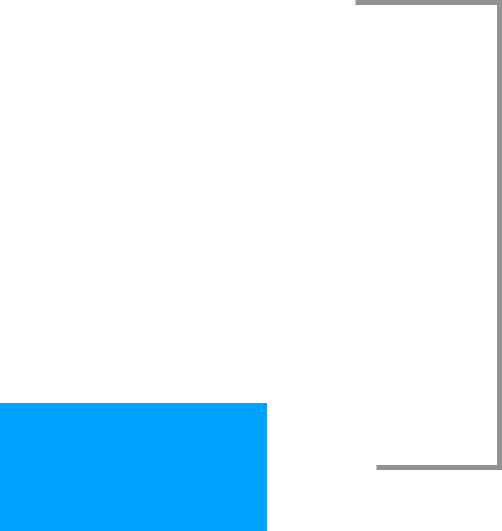
y = 10;

m = maximo(x, y);

...

**Endereço** **Variável**

|  |  |  |  |
| --- | --- | --- | --- |
| **3080** | 10 | **b** |  |
| **3072** |  | **a** |  |
|  |  |
| 5 |  |
|  |  |  |  |



**2016** 10 **m**



**2008** 10 **y**



**2000** 5 **x**



**APROG - LETI**

Passagem por referência



**APROG - LETI**

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);

...



**APROG - LETI**

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);

...

**Endereço** **Variável**



**2008** 10 **y**



**2000** 5 **x**



**APROG - LETI**

Passagem de argumentos por referência

void **trocar**(int \*a, int \*b)



{

int aux;

aux = \*a;

\*a = \*b;

\*b = aux;

}

int **main**() {

int x, y;

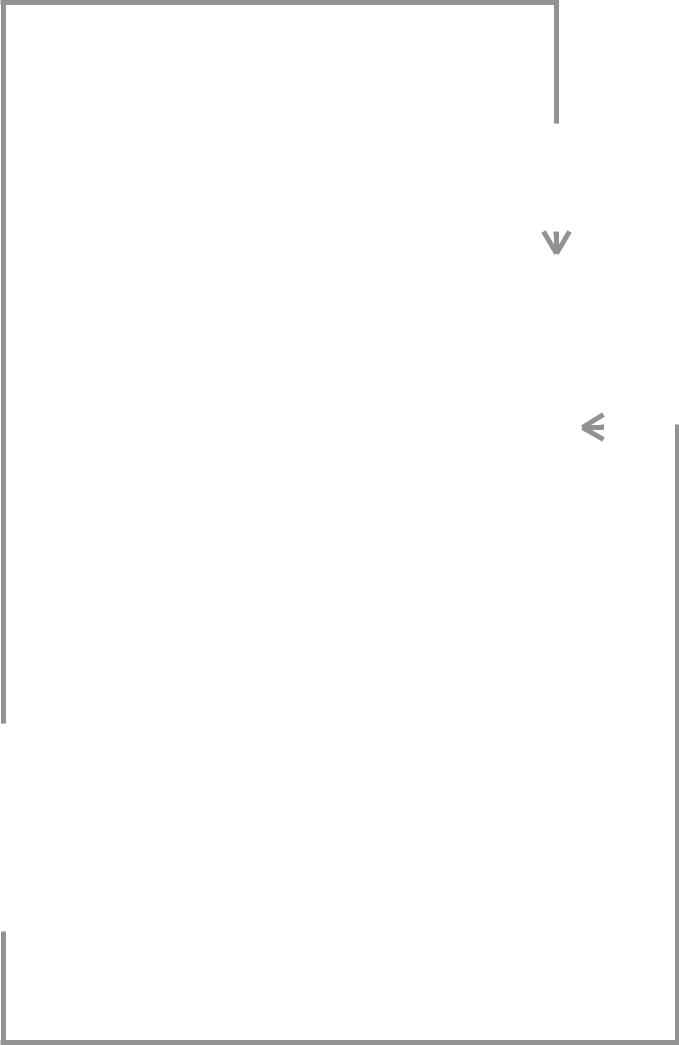
x = 5;

y = 10;

trocar(&x, &y);

...

**Endereço** **Variável**



|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **3088** |  |  |  |  |  |  |  |  |
| ??? |  | **aux** | | | |  |  |
| **3080** |  |  |  |  |  | |  |  |
|  |  |  |  |  |  |  |  |
| 2008 |  |  | | | |  |  |
|  | **b** | |  | |  |  |
| **3072** |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  | | |  |  |  |
| 2000 |  | **a** | |  |  |  |  |
|  |  |  |
|  |  |  |  |  |  |  |  |  |

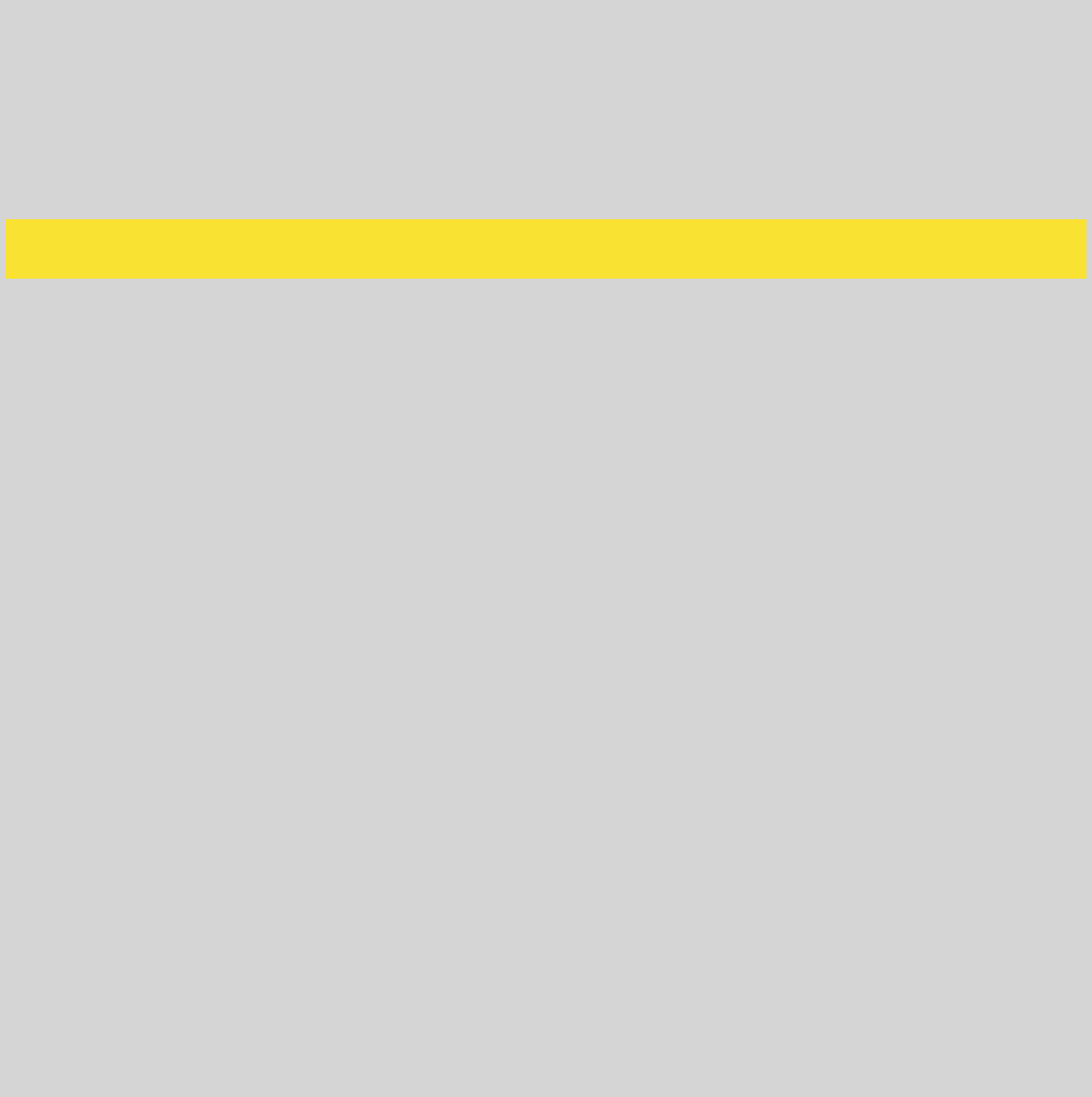
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **: 2008** |  |  |  |  |
| **&y** | 10 |  | **y** |  |
| **&x** | **: 2000** |  |  |  |  |
|  |  |  |  |
| 5 |  |  |  |
|  | **x** |  |
|  |  |  |  |  |  |



**APROG - LETI**

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);

...

**Endereço** **Variável**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **3088** |  |  |  |  |  |  |  |
| 5 |  |  |  | **aux** | |  |
|  |  |  |
| **3080** |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
| 2008 |  |  |  |  | |  |
|  |  |  | **b** |  |  |
| **3072** |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
| 2000 |  |  |  |  | |  |
|  |  |  | **a** |  |  |
|  |  |  |  |  |  |  |  |



|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **: 2008** |  |  |  |  |  |  |  |
| **&y** | 10 | |  | **y** |  | |  |
| **&x** | **: 2000** |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
| 5 | |  |  | | |  |
|  | **x : \*a** | | |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |



**APROG - LETI**

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;

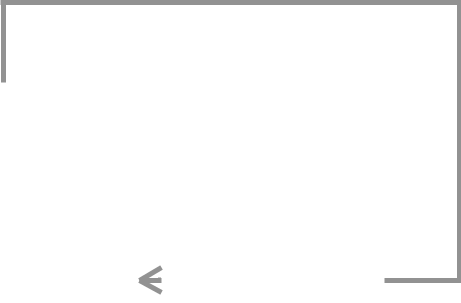
y = 10;

trocar(&x, &y);

...

**Endereço** **Variável**

|  |  |  |  |
| --- | --- | --- | --- |
| **3088** | 5 | **aux** |  |
| **3080** |  | **b** |  |
|  |  |
| 2008 |  |
| **3072** |  | **a** |  |
|  |  |
| 2000 |  |
|  |  |  |  |



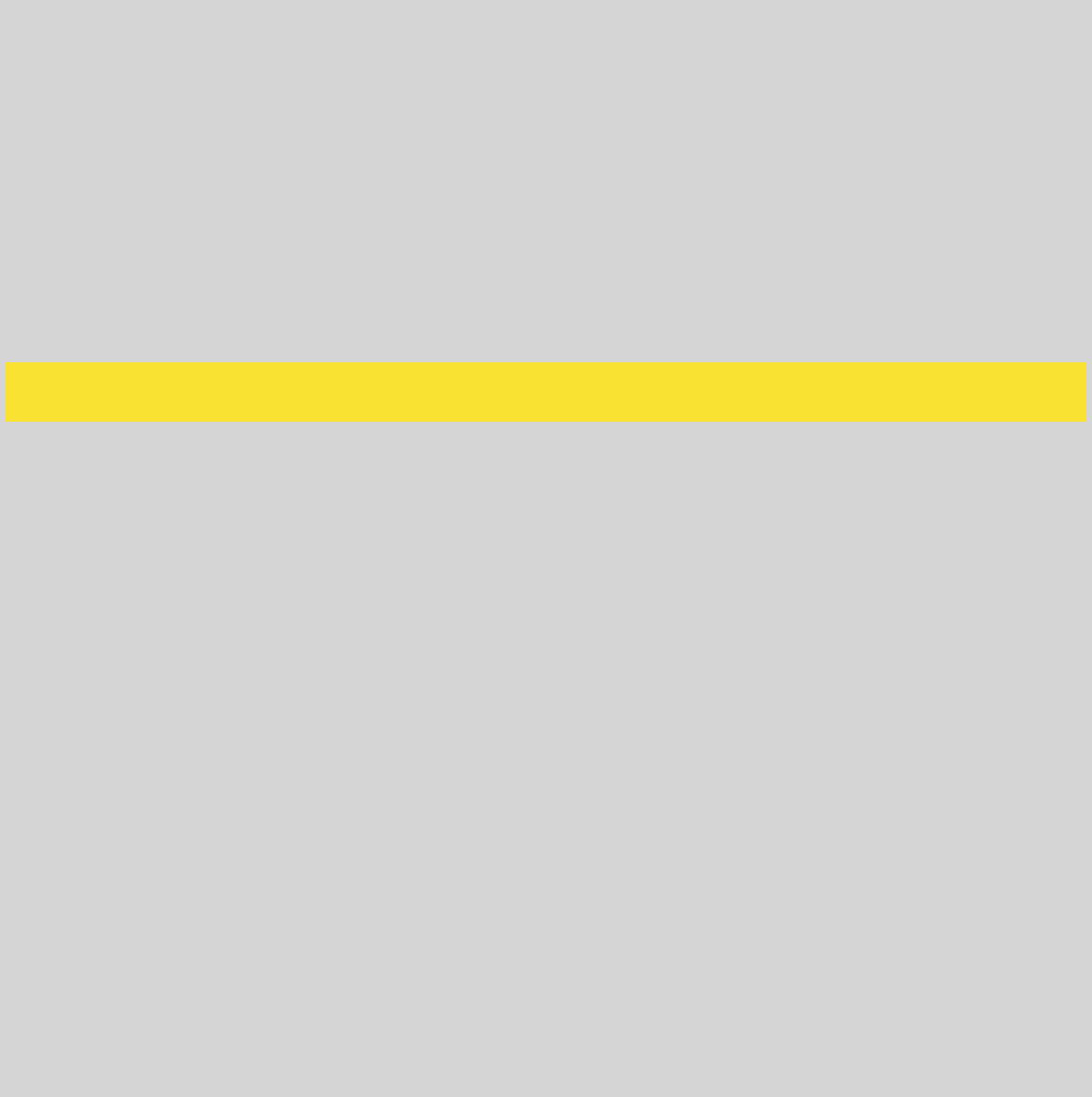
|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **: 2008** |  |  |  |  |  |  |  |
| **&y** | 10 |  |  | **y** | **:** | **\*b** |  |
|  | **: 2000** |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
| **&x** | 10 |  |  | **x** | **:** | **\*a** |  |
|  |  |
|  |  |  |  |  |  |  |  |  |



**APROG - LETI**

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);

...

**Endereço** **Variável**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **3088** |  |  |  |  |  |  |
| 5 |  |  | **aux** | |  |
|  |  |  |
| **3080** |  |  |  |  |  |  |
|  |  |  |  |  |  |
| 2008 |  |  |  | |  |
|  |  | **b** |  |  |
| **3072** |  |  |  |  |  |  |
|  |  |  |  |  |  |
| 2000 |  |  |  | |  |
|  |  | **a** |  |  |
|  |  |  |  |  |  |  |



|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **: 2008** |  |  |  |  |  |  |  |
| **&y** | 5 |  |  | **y** | **:** | **\*b** |  |
|  |  |
| **&x** | **: 2000** |  |  |  | **x** | **:** | **\*a** |  |
|  |  |  |  |
| 10 |  |  |  |
|  |  |  |  |  |  |  |  |  |



**APROG - LETI**

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!

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **3224** | 23 | **v[2]** |  |
| **v = &v[0] = 3208** | **3216** |  | **v[1]** |  |
|  |  |
| 5 |  |
|  | **3208** |  | **v[0]** |  |
|  |  |  |
|  | 3 |  |
|  |  |  |  |  |



**APROG - LETI**

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);

...



**APROG - LETI**

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;

}

}

}

}



**APROG - LETI**