

#### DCC004 - Algoritmos e Estruturas de Dados II

Armazenamento de dados em memória

Renato Martins

Email: renato.martins@dcc.ufmg.br

https://www.dcc.ufmg.br/~renato.martins/courses/DCC004

Material adaptado de PDS2 - Douglas Macharet e Flávio Figueiredo



### Um programa na memória

- Podemos representar um programa em regiões como:

  Stack
  - Pilha (Stack)
  - Heap
  - Código
  - Globais



## Compilador

Programa

## Compilador

```
$g++ hello.cpp -o hello
```

```
00000000000008a5 <main>:
                         push %rbp
 8a5:
      55
      48 89 e5
                                      %rsp,%rbp
 8a6:
                         mov
8a9: 48 8d 35 15 01 00 00 lea
                                      0x115(%rip),%rsi
                                                                # 9c5
<_ZStL19piecewise_construct+0x1>
8b0: 48 8d 3d a9 07 20 00 lea
                                      0x2007a9(%rip),%rdi
                                                                # 201060
< ZSt4cout@@GLIBCXX 3.4>
 8b7: e8 d4 fe ff ff
                         callq 790
< ZStlsISt11char traitsIcEERSt13basic ostreamIcT ES5 PKc@plt>
8bc: 48 89 c2
                                      %rax,%rdx
                         mov
8bf: 48 8b 05 0a 07 20 00 mov
                                      0x20070a(%rip),%rax
                                                                # 200fd0
<_ZSt4endlIcSt11char_traitsIcEERSt13basic_ostreamIT_T0_ES6_@GLIBCXX_3.4>
8c6: 48 89 c6
                                      %rax,%rsi
                         mov
                                      %rdx,%rdi
      48 89 d7
                         mov
                         callq 7a0 <_ZNSolsEPFRSoS_E@plt>
      e8 cf fe ff ff
                                      $0x0,%eax
      ъ8 00 00 00 00
 8d1:
                         mov
 8d6:
      5d
                         pop
                                      %rbp
 8d7: c3
                         retq
```

Programa

## Compilador

```
$g++ hello.cpp -o hello
```

```
00000000000008a5 <main>:
```

8a5: 55 push %rbp

8a6: 48 89 e5 mov %rsp,%rbp

8a9: 48 8d 35 15 01 00 00 lea 0x115(%rip),%rsi # 9c5

<\_ZStL19piecewise\_construct+0x1>

8b0: 48 8d 3d a9 07 20 00 lea 0x2007a9(%rip),%rdi # 201060

<\_ZSt4cout@@GLIBCXX\_3.4>

8b7: e8 d4 fe ff ff callq 790

<\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>

8bc: 48 89 c2 mov %rax,%rdx

8bf: 48 8b 05 0a 07 20 00 mov 0x20070a(%rip),%rax # 200fd0

<\_ZSt4endlIcSt11char\_traitsIcEERSt13basic\_ostreamIT\_T0\_ES6\_@GLIBCXX\_3.4>

8c6: 48 89 c6 mov %rax,%rsi

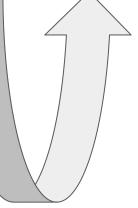
8cc: e8 cf fe ff ff callq 7a0 < ZNSolsEPFRSoS\_E@plt>

8d1: b8 00 00 00 00 mov \$0x0,%eax 8d6: 5d pop %rbp

8d7: c3 retq

DCC004 - Algoritmos e Estruturas de Dados II





- Fetch
  - Recupera instrução
  - e.g., movsd

CPU vaiexecutarmovsd

```
#include <iostream>
int main() {
double x = 71.1;
 double y = 0.01;
 std::cout << x / y;</pre>
 return 0;
                      Fetch
     CPU
                                      Memória
                    Barramento
                      Disco
```

A instrução faz store

x agora na memória

```
#include <iostream>
int main() {
double x = 71.1;
 double y = 0.01;
 std::cout << x / y;
 return 0;
                     Store x
     CPU
                                     Memória
                   Barramento
                     Disco
```

- Fetch
  - Recupera instrução
     e.g., movsd

CPU vaiexecutarmovsd

```
#include <iostream>
int main() {
 double x = 71.1;
 double y = 0.01;
 std::cout << x / y;
 return 0;
                     Fetch
     CPU
                                     Memória
                   Barramento
                     Disco
```

A instrução faz store

y agora na memória

```
#include <iostream>
int main() {
 double x = 71.1;
 double y = 0.01;
 std::cout << x / y;</pre>
 return 0;
                      Store y
     CPU
                                       Memória
                    Barramento
                      Disco
```

Load

Recuperamos
 x/y pois é
 necessário

```
#include <iostream>
int main() {
 double x = 71.1;
 double y = 0.01;
 std::cout << x / y;</pre>
 return 0;
                      Load x
     CPU
                                       Memória
                    Barramento
                      Disco
```

Load

Recuperamos
 x/y pois é
 necessário

```
#include <iostream>
int main() {
 double x = 71.1;
 double y = 0.01;
 std::cout << x / y;</pre>
 return 0;
                      Load x
     CPU
                                       Memória
                    Barramento
                      Disco
```

 Fazemos fetch de divsd (divisão)

```
#include <iostream>
int main() {
 double x = 71.1;
 double y = 0.01;
 std::cout << x / y;</pre>
 return 0;
                      Fetch /
     CPU
                                       Memória
                    Barramento
                       Disco
```

## Execução de um Programa

Ciclo fetch/execute -- Assunto de OC/SO

Executamos

```
x/y
```

Fim!

```
#include <iostream>
int main() {
  double x = 71.1;
  double y = 0.01;
  std::cout << x / y;
  return 0;
}</pre>
```

```
CPU
Execute x/y!

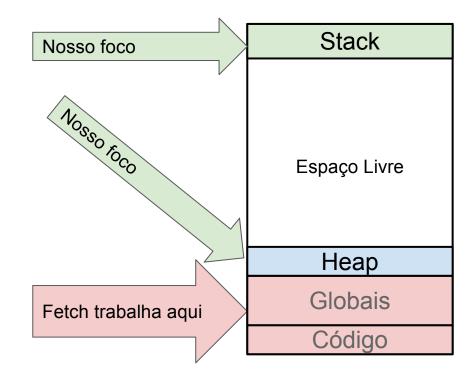
Barramento

Disco
```

## Voltando para a memória

Vamos focar no Stack/Heap

Onde moram "os dados"



 A parte de código não muda muito de uma linguagem para outra (compilador)

## Compilando

#### \$ g++ programa.cpp -o programa

```
#include <iostream>
int function(int &f) {
 f=f+3;
 return f;
}
int main() {
 int x = 7;
 int y;
 y = function(x);
 x++;
 y--;
 return 0;
```

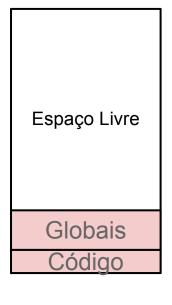
nome	ender.
	0x0048
	0x0044
	0x0040
	0x003c
	0x0038
	0x0034
	0x0030
	0x002c
	0x0028
	0x0024
	0x0020
	0x001c
	0x0018
	0x0014
	0x0010
	0x000c
	0x0008
	0x0004
	0x0000

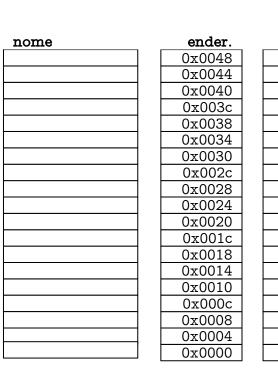
valor

## Compilando

#### \$ g++ programa.cpp -o programa

```
#include <iostream>
int function(int &f) {
 f=f+3;
 return f;
}
int main() {
 int x = 7;
 int y;
 y = function(x);
 x++;
 y--;
 return 0;
```



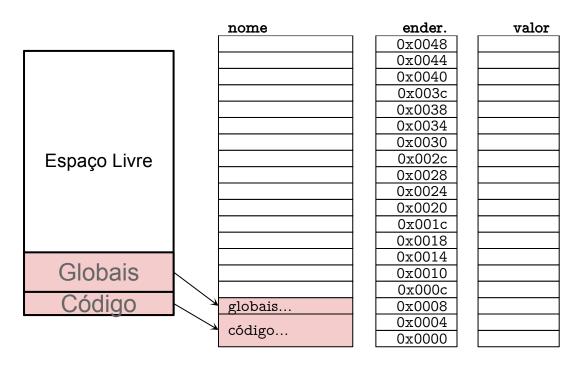


valor

#### Dando início

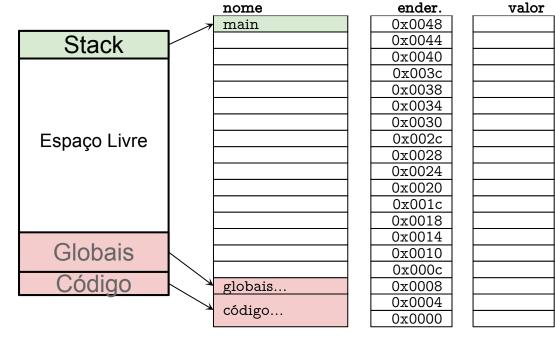
#### \$./programa

```
#include <iostream>
int function(int &f) {
 f=f+3;
 return f;
}
int main() {
 int x = 7;
 int y;
 y = function(x);
 x++;
 y--;
 return 0;
```



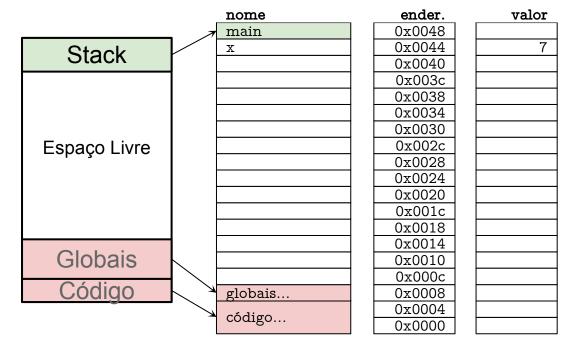
## Colocamos o main na pilha

```
#include <iostream>
int function(int &f) {
 f=f+3;
 return f;
int main() {
 int x = 7;
 int y;
 y = function(x);
 x++;
 y--;
 return 0;
```



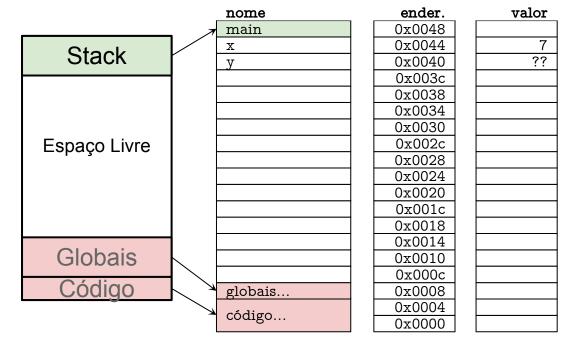
#### x = 7

```
#include <iostream>
int function(int &f) {
 f=f+3;
 return f;
}
int main() {
nt x = 7;
 int y;
 y = function(x);
 x++;
 y--;
 return 0;
```



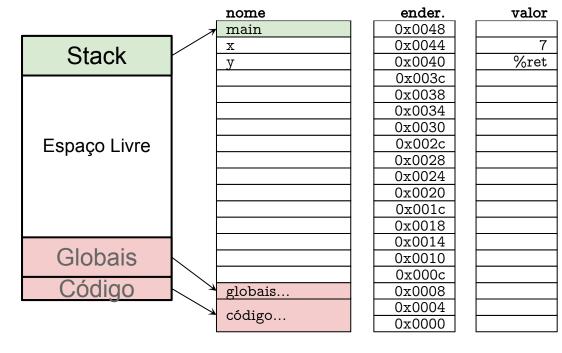
### y = lixo

```
#include <iostream>
int function(int &f) {
 f=f+3;
 return f;
}
int main() {
 int x = 7;
int y;
 y = function(x);
 x++;
 y--;
 return 0;
```



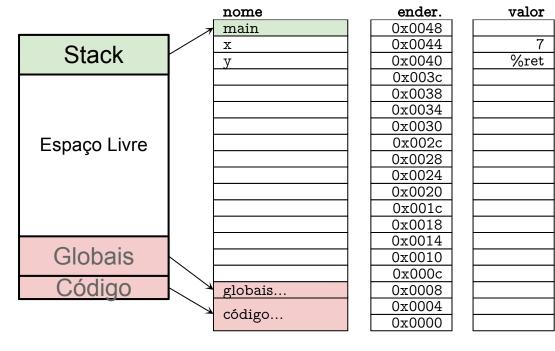
y <u>— retorno de function. Representado p</u>

```
#include <iostream>
int function(int &f) {
 f=f+3;
 return f;
}
int main() {
 int x = 7;
 int y;
 y = function(x);
 x++;
 y--;
 return 0;
```



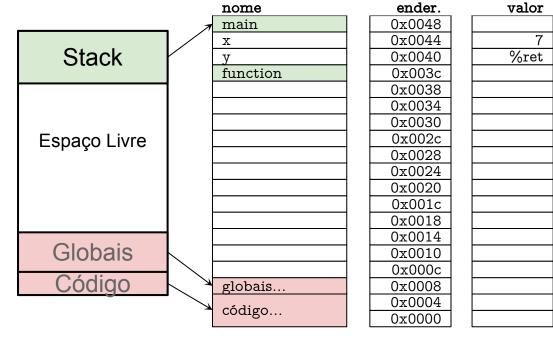
#### y = retorno de function. Representado por %ret

```
#include <iostream>
int function(int &f) {
 f=f+3;
 return f;
}
int main() {
 int x = 7;
 int y;
y = function(x);
 x++;
 y--;
 return 0;
```



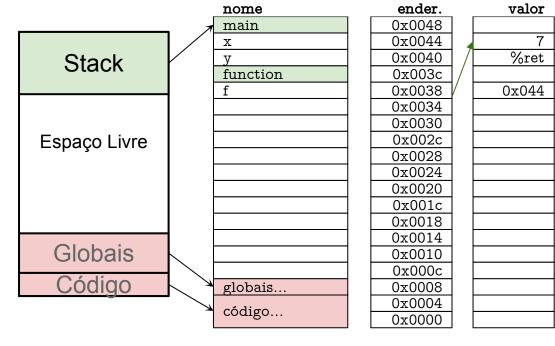
## chamamos a função function

```
#include <iostream>
int function(int &f) {
 f=f+3;
 return f;
}
int main() {
 int x = 7;
 int y;
 y = function(x);
 x++;
 y--;
 return 0;
```

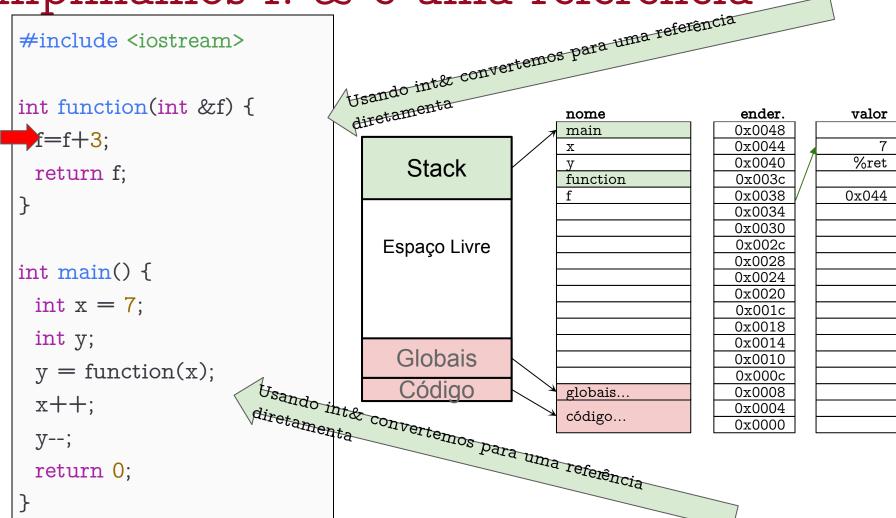


empilhamos f. & é uma referência

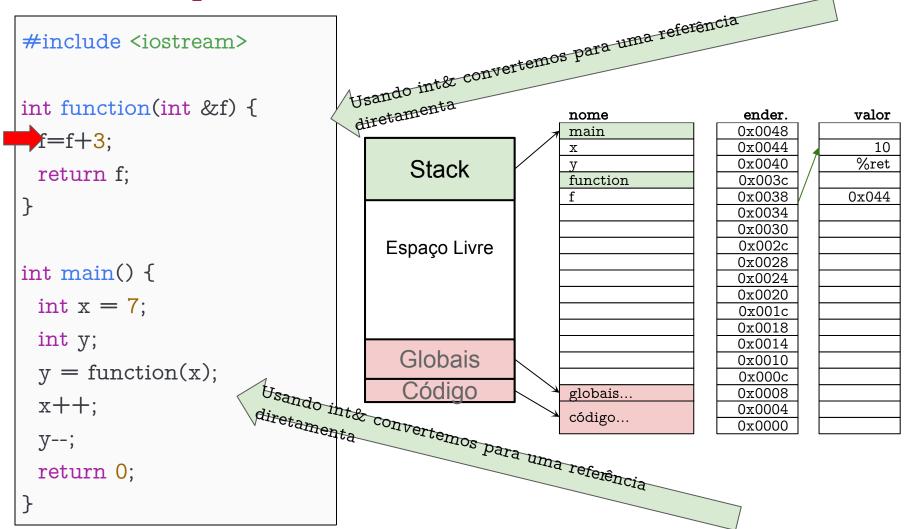
```
#include <iostream>
int function(int &f) {
 f=f+3;
 return f;
}
int main() {
 int x = 7;
 int y;
 y = function(x);
 x++;
 y--;
 return 0;
```



empilhamos f. & é uma referência

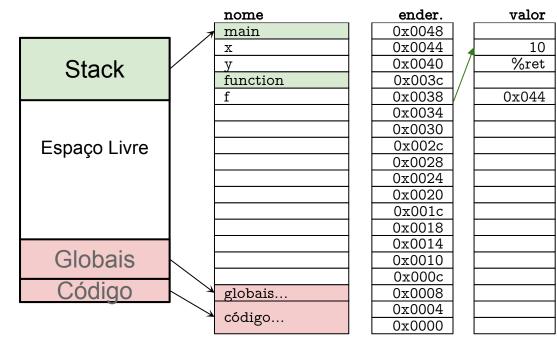


diferente do ponteiro em C, referências são imutáveis



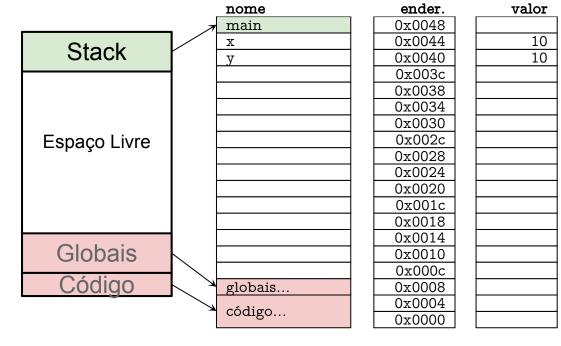
#### Uma vantagem, fim do uso excessivo de \*

```
#include <iostream>
int function(int &f) {
 f=f+3;
return f;
int main() {
 int x = 7;
 int y;
 y = function(x);
 x++;
 y--;
 return 0;
```



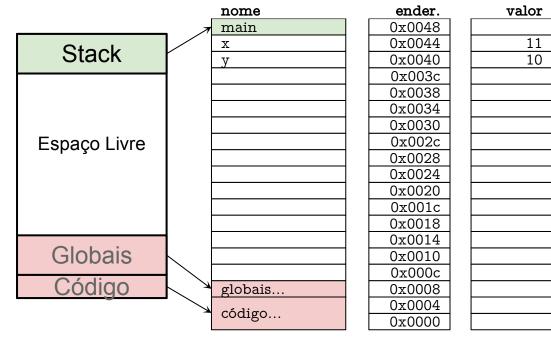
#### Voltamos para o main

```
#include <iostream>
int function(int &f) {
 f=f+3;
 return f;
}
int main() {
 int x = 7;
 int y;
y = function(x);
 x++;
 y--;
 return 0;
```



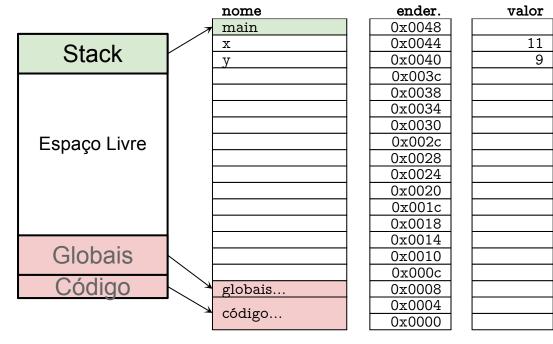
#### x e y estão em posições diferentes da memória

```
#include <iostream>
int function(int &f) {
 f=f+3;
 return f;
}
int main() {
 int x = 7;
 int y;
 y = function(x);
x++;
 y--;
 return 0;
```



#### x e y estão em posições diferentes da memória

```
#include <iostream>
int function(int &f) {
 f=f+3;
 return f;
}
int main() {
 int x = 7;
 int y;
 y = function(x);
 x++;
 return 0;
```



### Stack/Pilha

#### Controla o fluxo de execução do programa

- Comportamento similar ao TAD
   Pilha
  - Para aqueles que viram ED

 Vai guardando a posição das funções, retornos e variáveis

# Referências & e ponteiros \* Quase iguais...

- Assim como em C: & (address of)
- Porém, podemos usar como:

- Ponteiro
  - int \*valor\_ptr = &valor;

- Referência
  - int &valor\_ptr = &valor;



### Comparando

```
#include <iostream>
                                             #include <iostream>
int function(int &f) {
                                            int function(int *f) {
                                  Mesmo
                                              *f=*f+3;
 f=f+3;
                              comportamento
 return f;
                                              return *f;
int main() {
                                            int main() {
 int x = 7;
                                              int x = 7;
 int y;
                                              int y;
 y = function(x);
                                              y = function(&x);
 x++;
                                              x++;
 y--;
                                              y--;
 return 0;
                                              return 0;
```

## Referências & e ponteiros \* Quando usar cada um

- Ao alocar memória no heap, é comum utilizar \*
  - malloc em C
  - new em C++

- Referências (&) representa um ponteiro imutável útil para evitar bugs
  - Boa prática em funções



# Ainda sobre & vs \* Uma dica simples que vai lhe ajudar

- Use & sempre que possível
  - O compilador deixar passar
  - Bastante explorado em OO
  - Em funções é um bom local
- Use \* sempre que necessário
  - Vetores
  - Quando cria a variável
  - Manipulação mais baixo nível (TADs)



## Exemplos

Duas formas de passar por referência em C++

```
void function(int &i) {
  int j = 10;
  i = &j;
}
```

```
void function(int *i) {
  int j = 10;
  i = &j;
}
```

#### Duas formas de passar por referência em C++



#### Erro de compilação

```
void function(int &i) {
  int j = 10;
  i = &j;
}
```



Compila só que é uma ótima fonte de bugs!

```
void function(int *i) {
  int j = 10;
  i = &j;
}
```

## Exemplo (por valor)

```
#include <iostream>
int inc(int x) {
 return ++x;
}
int main() {
 int a = 10;
 int b = inc(a);
 std::cout << a << std::endl;</pre>
 std::cout << b << std::endl;</pre>
 std::cout << &a << std::endl;</pre>
 std::cout << &b << std::endl;</pre>
```

## Exemplo (por referência estilo C)

```
#include <iostream>
int inc(int *x) {
 *x = *x + 1:
 return *x;
int main() {
 int a = 10;
 int b = inc(\&a);
 std::cout << a << std::endl;</pre>
 std::cout << b << std::endl;
 std::cout << &a << std::endl;
 std::cout << &b << std::endl;</pre>
```

## Exemplo (por referência estilo C++)

```
#include <iostream>
int inc(int &x) {
 x++;
 return x;
int main() {
 int a = 10;
 int b = inc(a);
 std::cout << a << std::endl;</pre>
 std::cout << b << std::endl;
 std::cout << &a << std::endl;
 std::cout << &b << std::endl;</pre>
```

## Heap

### Memória alocada dinamicamente

- Uma forma simples de pensar:
  - Memória da pilha é gerida automaticamente
  - Só que temos menos controle
  - Nem sempre podemos usar a pilha
- O Heap é onde mora a memória dinâmica
  - Em C utilizamos malloc
  - Em C++ utilizamos new



## Heap

## Liberação de memória

- O programador que libera a memória
  - Em C utilizamos free
  - Em C++ utilizamos delete



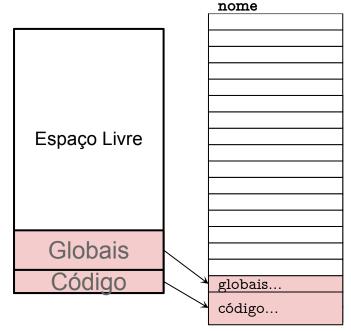
```
#include <cstdlib>
#include <iostream>
int main() {
 int *ptr_a = nullptr;
 ptr_a = new int;
 if (ptr_a == nullptr) {
   std::cout << "Memoria insuficiente!" << std::endl;</pre>
   exit(1);
 }
 std::cout << "Endereco de ptr_a: " << ptr_a << std::endl;</pre>
 *ptr_a = 90;
 std::cout << "Conteudo de ptr_a: " << *ptr_a << std::endl;</pre>
 delete ptr_a;
```

## NULL vs nullptr

- Semanticamente igual
- nullptr é mais seguro
  - Só pode ser atribuído para ponteiros

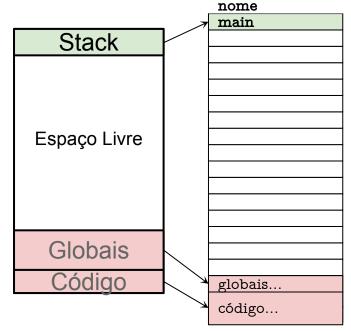
```
int i = NULL;  // OK, qual o valor de i?
int i = nullptr; // Erro de compilação
int *p = NULL;  // OK, seguro
int *p = nullptr; // OK, seguro
```

```
int main() {
  int *a = new int;
  int b = 10;
  *a = 20;
  a = &b;
  *a = 30;
}
```



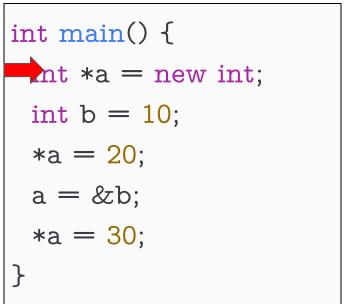
ender.	valor
0x0048	
0x0044	
0x0040	
0x003c	
0x0038	
0x0034	
0x0030	
0x002c	
0x0028	
0x0024	
0x0020	
0x001c	
0x0018	
0x0014	
0x0010	
0x000c	
8000x0	
0x0004	
0x0000	

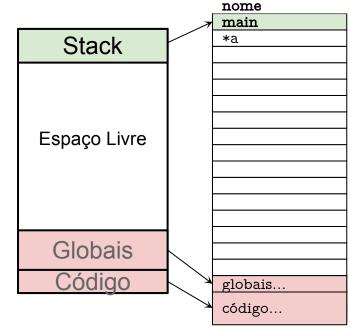
```
int main() {
  int *a = new int;
  int b = 10;
  *a = 20;
  a = &b;
  *a = 30;
}
```



ender.	valor
0x0048	
0x0044	
0x0040	
0x003c	
0x0038	
0x0034	
0x0030	
0x002c	
0x0028	
0x0024	
0x0020	
0x001c	
0x0018	
0x0014	
0x0010	
0x000c	
8000x0	
0x0004	
0x0000	

### Alocamos um inteiro no heap, a referência mora a pilha

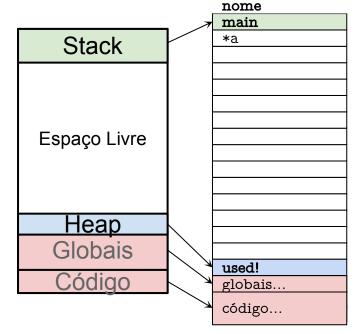




	ender.	valor
Г		Vaioi
	0x0048	
L	0x0044	
L	0x0040	
	0x003c	
	0x0038	
Γ	0x0034	
Ī	0x0030	
Γ	0x002c	
Γ	0x0028	
Ī	0x0024	
Γ	0x0020	
Γ	0x001c	
Γ	0x0018	
	0x0014	
Γ	0x0010	
Γ	0x000c	
Γ	8000x0	
Γ	0x0004	
Γ	0x0000	

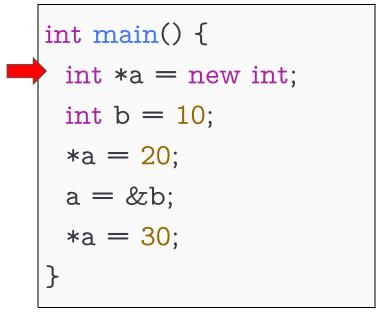
### Alocamos um inteiro no heap, a referência mora a pilha

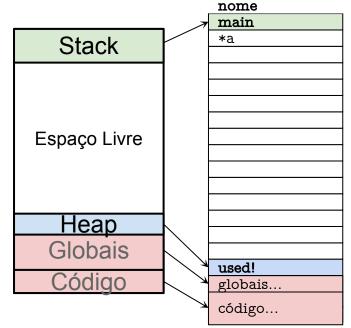
```
int main() {
    nt *a = new int;
    int b = 10;
    *a = 20;
    a = &b;
    *a = 30;
}
```



ender.	valor
0x0048	
0x0044	
0x0040	
0x003c	
0x0038	
0x0034	
0x0030	
0x002c	
0x0028	
0x0024	
0x0020	
0x001c	
0x0018	
0x0014	
0x0010	
0x000c	??
8000x0	
0x0004	
0x0000	

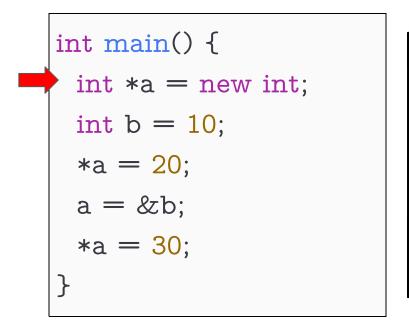
### Alocamos um inteiro no heap, a referência mora a pilha

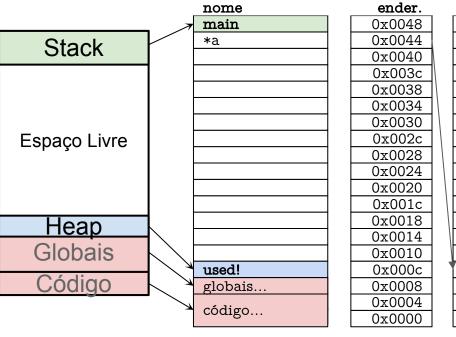




ender.		valor
0x0048		
0x0044		0x000c
0x0040		
0x003c	1	
0x0038	]\	
0x0034	] [	
0x0030	11	
0x002c	1 \	
0x0028	1 \	
0x0024	1	
0x0020	1	
0x001c	1	
0x0018	1	
0x0014		
0x0010		
0x000c	1	??
0x0008		
0x0004		
0x0000		

#### Não temos variável no heap! Acesso por \*a







valor

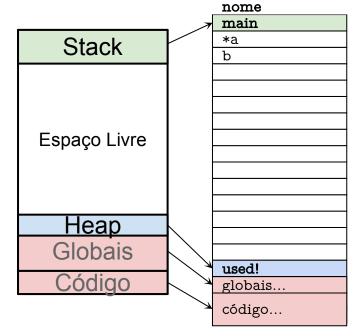
??

0x000c

# Exemplo<br/>b mora na pilha

```
int main() {
  int *a = new int;

int b = 10;
  *a = 20;
  a = &b;
  *a = 30;
}
```

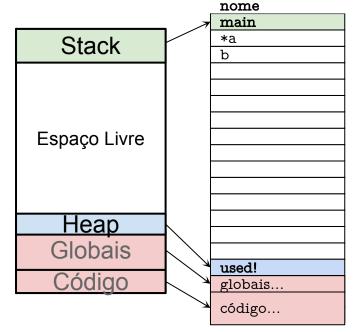


ender.		valor
0x0048		
0x0044		0x000c
0x0040	1	10
0x003c	\	
0x0038	1	
0x0034		
0x0030		
0x002c		
0x0028		
0x0024		
0x0020		
0x001c		
0x0018		
0x0014		
0x0010		
0x000c	•	??
8000x0		
0x0004		
0x0000		

## Como que o diagrama muda?

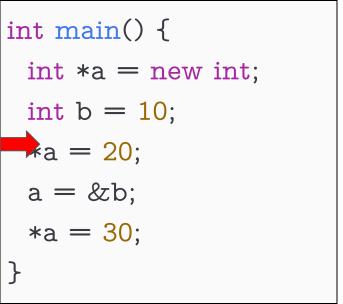
```
int main() {
  int *a = new int;
  int b = 10;

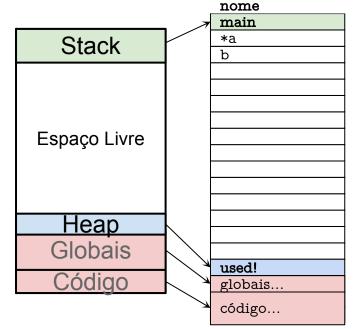
  *a = 20;
  a = &b;
  *a = 30;
}
```



ender.		valor
0x0048		
0x0044		0x000c
0x0040	1	10
0x003c	1	
0x0038	]\	
0x0034		
0x0030	11	
0x002c	1 \	
0x0028	1	
0x0024		
0x0020		
0x001c		
0x0018		
0x0014		
0x0010		
0x000c	•	??
8000x0		
0x0004		
0x0000		

### Valor do local de memória referenciado por a = 20



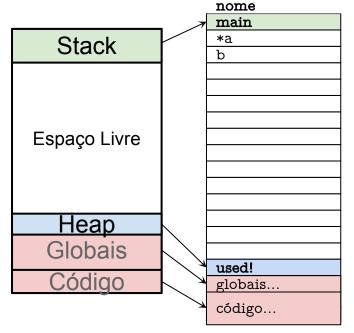


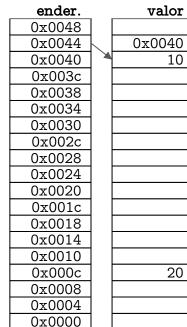
ender.		valor
0x0048		
0x0044		0x000c
0x0040		10
0x003c	\	
0x0038	1	
0x0034		
0x0030		
0x002c		
0x0028		
0x0024		
0x0020		
0x001c		
0x0018		
0x0014		
0x0010		
0x000c	1	20
8000x0		
0x0004		
0x0000		

## Como que o diagrama muda?

```
int main() {
  int *a = new int;
  int b = 10;
  *a = 20;

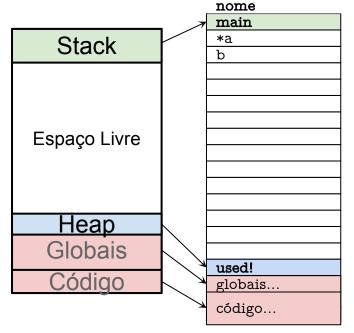
  a = &b;
  *a = 30;
}
```

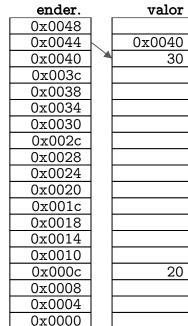




## Como que o diagrama muda?

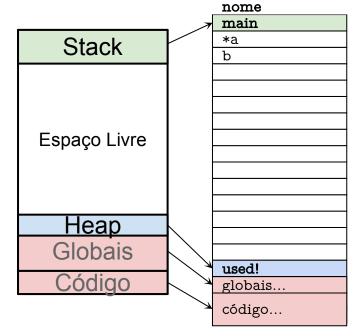
```
int main() {
  int *a = new int;
  int b = 10;
  *a = 20;
  a = &b;
  *a = 30;
}
```





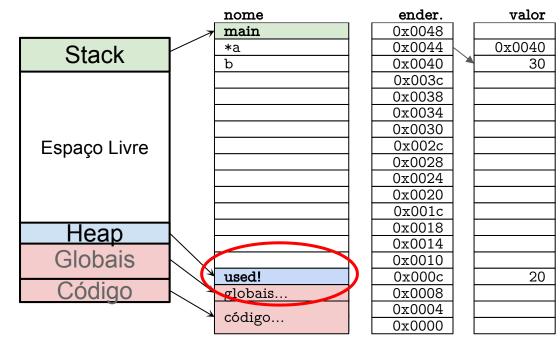
```
int main() {
  int *a = new int;
  int b = 10;
  *a = 20;
  a = &b;

*a = 30;
}
```

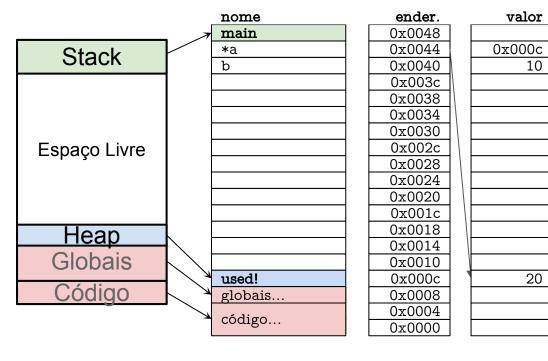


ender.		valor
0x0048		
0x0044		0x0040
0x0040	×	30
0x003c		
0x0038		
0x0034		
0x0030		
0x002c		
0x0028		
0x0024		
0x0020		
0x001c		
0x0018		
0x0014		
0x0010		
0x000c		20
8000x0		
0x0004		
0x0000		

```
int main() {
  int *a = new int;
  int b = 10;
  *a = 20;
  a = &b;
  *a = 30;
}
```

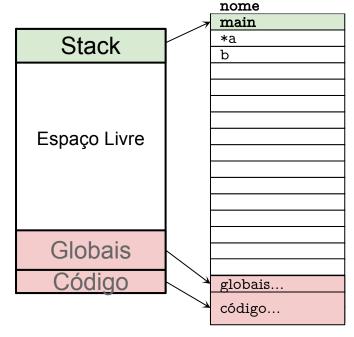


```
int main() {
  int *a = new int;
  int b = 10;
  *a = 20;
  delete a;
  a = &b;
  *a = 30;
}
```



```
int main() {
  int *a = new int;
  int b = 10;
  *a = 20;
  delete a;

a = &b;
  *a = 30;
}
```



ender.	valor
0x0048	
0x0044	??
0x0040	10
0x003c	
0x0038	
0x0034	
0x0030	
0x002c	
0x0028	
0x0024	
0x0020	
0x001c	
0x0018	
0x0014	
0x0010	
0x000c	
8000x0	
0x0004	
0x0000	

## Alocação dinâmica de vetores

- Normalmente, a alocação dinâmica é utilizada para criar vetores em tempo de execução
- Exemplo:
- int \*p = new int[10];
  - Aloca um vetor de inteiros com 10 posições. A manipulação é feita normalmente: p[i] = ...
  - O apontador p guarda o endereço (aponta) da primeira posição do vetor.

```
#include <cstdlib>
#include <iostream>
int *produto interno(int n, int *vetor a, int *vetor b) {
 int *resultado = new int[n];
 for (int i = 0; i < n; i++)
   resultado[i] = vetor a[i] * vetor b[i];
 return resultado;
}
int main() {
 int vetor_a[3] = \{1, 2, 3\};
 int vetor b[3] = \{3, 2, 1\};
 int *resultado = produto_interno(3, vetor_a, vetor_b);
 for (int i = 0; i < 3; i++)
   std::cout << resultado[i];</pre>
 delete[] resultado;
                              delete[]
```

```
int *produto(int n, int *va, int *vb) {
 int *resultado = new int[n];
 for (int i = 0; i < n; i++)
                                                                      nome
                                                                                          ender.
                                                                                                       valor
   resultado[i] = va[i] * vb[i]:
                                                                      main
                                                                                         0x0048
                                                                                         0x0044
                                                   Stack
 return resultado;
                                                                                         0x0040
                                                                                         0x003c
}
                                                                                         0x0038
                                                                                         0x0034
                                                                                         0x0030
int main() {
                                                Espaço Livre
                                                                                          0x002c
                                                                                         0x0028
\mathbf{va}[3] = \{1, 2, 3\};
                                                                                         0x0024
                                                                                         0x0020
 int vb[3] = \{3, 2, 1\};
                                                                                         0x001c
                                                                                         0x0018
 int *resultado = nullptr;
                                                                                         0x0014
                                                  Globais
 resultado = produto(3, va, vb);
                                                                                         0x0010
                                                                                         0x000c
                                                  Códiao
 for (int i = 0; i < 3; i++)
                                                                     globais...
                                                                                         8000x0
                                                                                         0x0004
                                                                      código...
   std::cout << resultado[i];</pre>
                                                                                         0x0000
 delete[] resultado;
}
```

```
int *produto(int n, int *va, int *vb) {
 int *resultado = new int[n];
 for (int i = 0; i < n; i++)
                                                                       nome
                                                                                           ender.
                                                                                                         valor
   resultado[i] = va[i] * vb[i]:
                                                                       main
                                                                                           0x0048
                                                                       va[0]
                                                                                           0x0044
                                                                                                            1
 return resultado;
                                                    Stack
                                                                       va[1]
                                                                                           0x0040
                                                                       va[2]
                                                                                           0x003c
                                                                                           0x0038
                                                                                           0x0034
                                                                                           0x0030
                                                 Espaço Livre
int main() {
                                                                                           0x002c
                                                                                           0x0028
 int va[3] = \{1, 2, 3\};
                                                                                           0x0024
                                                                                           0x0020
int vb[3] = \{3, 2, 1\};
                                                                                           0 \times 0.01 c
                                                                                           0x0018
 int *resultado = nullptr;
                                                                                           0x0014
                                                  Globais
 resultado = produto(3, va, vb);
                                                                                           0x0010
                                                                                           0x000c
                                                   Códiao
 for (int i = 0; i < 3; i++)
                                                                       globais...
                                                                                           8000x0
                                                                                           0x0004
                                                                       código...
   std::cout << resultado[i];</pre>
                                                                                           0x0000
 delete[] resultado;
}
```

```
int *produto(int n, int *va, int *vb) {
 int *resultado = new int[n];
 for (int i = 0; i < n; i++)
                                                                      nome
                                                                                           ender.
                                                                                                        valor
   resultado[i] = va[i] * vb[i]:
                                                                      main
                                                                                          0x0048
                                                                      va[0]
                                                                                          0x0044
                                                                                                           1
 return resultado;
                                                                      va[1]
                                                                                          0x0040
                                                                                                           3
                                                                      va[2]
                                                                                          0x003c
                                                   Stack
                                                                                                           3
                                                                      vb[0]
                                                                                          0x0038
                                                                      vb[1]
                                                                                          0x0034
                                                                      vb[2]
                                                                                          0x0030
int main() {
                                                 Espaço Livre
                                                                                          0x002c
                                                                                          0x0028
 int va[3] = \{1, 2, 3\};
                                                                                          0x0024
                                                                                          0x0020
 int vb[3] = \{3, 2, 1\};
                                                                                          0x001c
                                                                                          0x0018
int *resultado = nullptr;
                                                                                          0x0014
                                                  Globais
 resultado = produto(3, va, vb);
                                                                                          0x0010
                                                                                          0x000c
                                                   Códiao
 for (int i = 0; i < 3; i++)
                                                                      globais...
                                                                                          8000x0
                                                                                          0x0004
                                                                      código...
   std::cout << resultado[i];</pre>
                                                                                          0x0000
 delete[] resultado;
}
```

```
int *produto(int n, int *va, int *vb) {
 int *resultado = new int[n];
 for (int i = 0; i < n; i++)
                                                                       nome
                                                                                            ender.
                                                                                                          valor
   resultado[i] = va[i] * vb[i]:
                                                                        main
                                                                                            0 \times 0048
                                                                       va[0]
                                                                                            0x0044
                                                                                                             1
 return resultado;
                                                                       va[1]
                                                                                            0x0040
                                                                                                             3
                                                                       va[2]
                                                                                            0x003c
                                                    Stack
                                                                                                             3
                                                                       vb[0]
                                                                                            0x0038
                                                                       vb[1]
                                                                                            0x0034
                                                                       vb[2]
                                                                                            0x0030
int main() {
                                                                       *resultado
                                                                                                        nullptr
                                                                                            0x002c
                                                                                            0x0028
 int va[3] = \{1, 2, 3\};
                                                                                            0x0024
                                                                                            0x0020
 int vb[3] = \{3, 2, 1\};
                                                                                            0x001c
                                                                                            0x0018
 int *resultado = nullptr;
                                                                                            0x0014
                                                   Globais
resultado = produto(3, va, vb);
                                                                                            0x0010
                                                                                            0x000c
                                                    Códiao
 for (int i = 0; i < 3; i++)
                                                                       globais...
                                                                                            8000x0
                                                                                            0x0004
                                                                       código...
   std::cout << resultado[i];</pre>
                                                                                            0x0000
 delete[] resultado;
}
```

```
int *produto(int n, int *va, int *vb) {
int *resultado = new int[n];
 for (int i = 0; i < n; i++)
                                                                        nome
                                                                                             ender.
                                                                                                           valor
   resultado[i] = va[i] * vb[i]:
                                                                        main
                                                                                             0x0048
                                                                        va[0]
                                                                                             0x0044
                                                                                                              1
 return resultado;
                                                                                             0x0040
                                                                        va[1]
                                                                        va[2]
                                                                                             0x003c
}
                                                     Stack
                                                                        vb[0]
                                                                                             0x0038
                                                                                             0x0034
                                                                        vb[1]
                                                                        vb[2]
                                                                                             0x0030
int main() {
                                                                                                         nullptr
                                                                        *resultado
                                                                                             0x002c
                                                                        produto
                                                                                             0x0028
 int va[3] = \{1, 2, 3\};
                                                                                             0x0024
                                                                        n
                                                                                                         0x0044
                                                                                             0x0020
                                                                        *va
 int vb[3] = \{3, 2, 1\};
                                                                        *vb
                                                                                             0 \times 0.01 c
                                                                                                         0x0038
                                                                                             0x0018
 int *resultado = nullptr;
                                                                                             0x0014
                                                   Globais
 resultado = produto(3, va, vb);
                                                                                             0x0010
                                                                                             0x000c
                                                    Códiao
 for (int i = 0; i < 3; i++)
                                                                        globais...
                                                                                             8000x0
                                                                                             0x0004
                                                                        código...
   std::cout << resultado[i];</pre>
                                                                                             0x0000
 delete[] resultado;
}
```

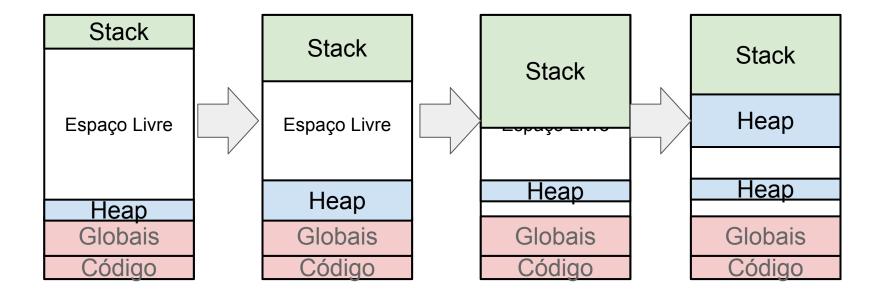
```
int *produto(int n, int *va, int *vb) {
 int *resultado = new int[n];
 for (int i = 0; i < n; i++)
                                                                       nome
                                                                                            ender.
                                                                                                          valor
   resultado[i] = va[i] * vb[i]:
                                                                        main
                                                                                            0x0048
                                                                       va[0]
                                                                                            0x0044
                                                                                                             1
 return resultado;
                                                                                            0x0040
                                                                        va[1]
                                                                       va[2]
                                                                                            0x003c
}
                                                    Stack
                                                                       vb[0]
                                                                                            0x0038
                                                                       vb[1]
                                                                                            0x0034
                                                                       vb[2]
                                                                                            0x0030
int main() {
                                                                                                        nullptr
                                                                        *resultado
                                                                                            0x002c
                                                                       produto
                                                                                            0x0028
 int va[3] = \{1, 2, 3\};
                                                                                            0x0024
                                                                       n
                                                                                                        0x0044
                                                                                            0x0020
                                                    Heap
                                                                        *va
 int vb[3] = \{3, 2, 1\};
                                                                        *vb
                                                                                            0x001c
                                                                                                        0x0038
                                                                        *resultado
                                                                                                        0x000c
                                                                                            0x0018
 int *resultado = nullptr;
                                                                       used!
                                                                                            0x0014
                                                   Globais
                                                                       used!
 resultado = produto(3, va, vb);
                                                                                            0x0010
                                                                       used!
                                                                                            0x000c
                                                    Códiao
 for (int i = 0; i < 3; i++)
                                                                       globais...
                                                                                            8000x0
                                                                                            0x0004
                                                                       código...
   std::cout << resultado[i];</pre>
                                                                                            0x0000
 delete[] resultado;
}
```

```
int *produto(int n, int *va, int *vb) {
  int *resultado = new int[n];
  for (int i = 0; i < n; i++)
                                                                        nome
                                                                                            ender.
                                                                                                          valor
   resultado[i] = va[i] * vb[i]:
                                                                        main
                                                                                            0x0048
                                                                        va[0]
                                                                                            0x0044
                                                                                                             1
return resultado;
                                                                                            0x0040
                                                                        va[1]
                                                                        va[2]
                                                                                            0x003c
                                                    Stack
                                                                        vb[0]
                                                                                            0x0038
                                                                                            0x0034
                                                                        vb[1]
                                                                       vb[2]
                                                                                            0x0030
int main() {
                                                                                                        nullptr
                                                                        *resultado
                                                                                            0x002c
                                                                       produto
                                                                                            0x0028
  int va[3] = \{1, 2, 3\};
                                                                                            0x0024
                                                                       n
                                                                                                        0x0044
                                                                                            0x0020
                                                     Heap
                                                                        *va
  int vb[3] = \{3, 2, 1\};
                                                                                                        0x0038
                                                                        *vb
                                                                                            0x001c
                                                                                                        0x000c
                                                                        *resultado
                                                                                            0x0018
  int *resultado = nullptr;
                                                                       used!
                                                                                            0x0014
                                                   Globais
                                                                       used!
  resultado = produto(3, va, vb);
                                                                                            0x0010
                                                                       used!
                                                                                            0x000c
                                                    Códiao
  for (int i = 0; i < 3; i++)
                                                                       globais...
                                                                                            8000x0
                                                                                            0x0004
                                                                        código...
   std::cout << resultado[i];</pre>
                                                                                            0x0000
  delete[] resultado;
}
```

```
int *produto(int n, int *va, int *vb) {
 int *resultado = new int[n];
 for (int i = 0; i < n; i++)
                                                                       nome
                                                                                           ender.
                                                                                                         valor
   resultado[i] = va[i] * vb[i]:
                                                                       main
                                                                                           0x0048
                                                                       va[0]
                                                                                           0x0044
                                                                                                            1
 return resultado;
                                                                       va[1]
                                                                                           0x0040
                                                    Stack
                                                                                                            3
                                                                       va[2]
                                                                                           0x003c
                                                                                                            3
                                                                       vb[0]
                                                                                           0x0038
                                                                       vb[1]
                                                                                           0x0034
                                                                       vb[2]
                                                                                           0x0030
int main() {
                                                 Espaço Livre
                                                                       *resultado
                                                                                                       0x000c
                                                                                           0x002c
                                                                                           0x0028
 int va[3] = \{1, 2, 3\};
                                                                                           0x0024
                                                                                           0x0020
                                                    Heap
 int vb[3] = \{3, 2, 1\};
                                                                                           0x001c
                                                                                           0x0018
 int *resultado = nullptr;
                                                                                           0x0014
                                                                       used!
                                                  Globais
                                                                       used!
 resultado = produto(3, va, vb);
                                                                                           0x0010
                                                                       used!
                                                                                           0x000c
                                                   Códiao
for (int i = 0; i < 3; i++)
                                                                       globais...
                                                                                           8000x0
                                                                                           0x0004
                                                                       código...
   std::cout << resultado[i];</pre>
                                                                                           0x0000
 delete[] resultado;
}
```

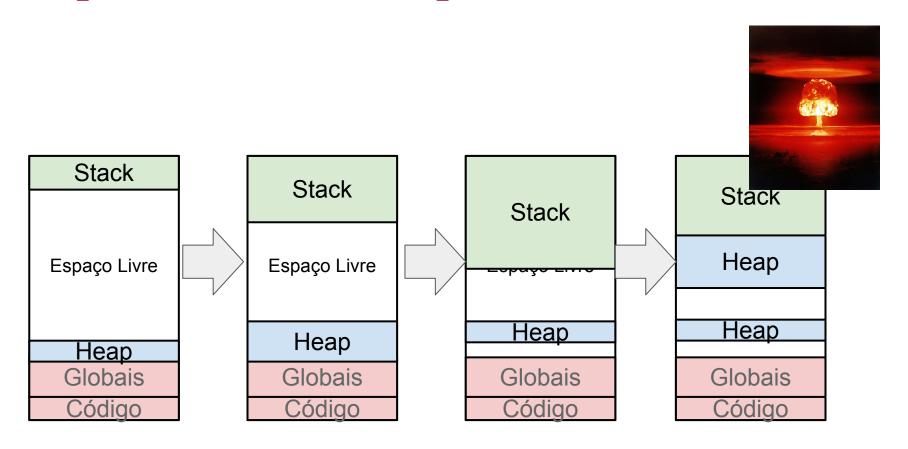
```
int *produto(int n, int *va, int *vb) {
  int *resultado = new int[n];
 for (int i = 0; i < n; i++)
                                                                        nome
                                                                                            ender.
                                                                                                          valor
   resultado[i] = va[i] * vb[i]:
                                                                        main
                                                                                            0x0048
                                                                        va[0]
                                                                                            0x0044
                                                                                                             1
 return resultado;
                                                                        va[1]
                                                                                            0x0040
                                                    Stack
                                                                        va[2]
                                                                                            0x003c
                                                                       vb[0]
                                                                                            0x0038
                                                                        vb[1]
                                                                                            0x0034
                                                                       vb[2]
                                                                                            0x0030
int main() {
                                                 Espaço Livre
                                                                        *resultado
                                                                                                        0x000c
                                                                                            0x002c
                                                                                            0x0028
 int va[3] = \{1, 2, 3\};
                                                                                            0x0024
                                                                                            0x0020
 int vb[3] = \{3, 2, 1\};
                                                                                            0 \times 0.01 c
                                                                                            0x0018
  int *resultado = nullptr;
                                                                                            0x0014
                                                   Globais
  resultado = produto(3, va, vb);
                                                                                            0x0010
                                                                                            0x000c
                                                    Códiao
  for (int i = 0; i < 3; i++)
                                                                       globais...
                                                                                            8000x0
                                                                                            0x0004
                                                                        código...
   std::cout << resultado[i];</pre>
                                                                                            0x0000
delete[] resultado;
```

## Sempre libere o heap





## Sempre libere o heap



#### Erros comuns

- Esquecer de alocar memória e tentar acessar o conteúdo da variável
- Copiar o valor do apontador ao invés do valor da variável apontada
- Esquecer de desalocar memória
  - Ela é desalocada ao fim do programa ou procedimento função onde a variável está declarada, mas pode ser um problema em loops
- Tentar acessar o conteúdo da variável depois de desalocá-la

