



PCS311

**Laboratório de Programação
Orientada a Objetos para
Engenharia Elétrica**

Aula 2: Ponteiros

Escola Politécnica da Universidade de São Paulo

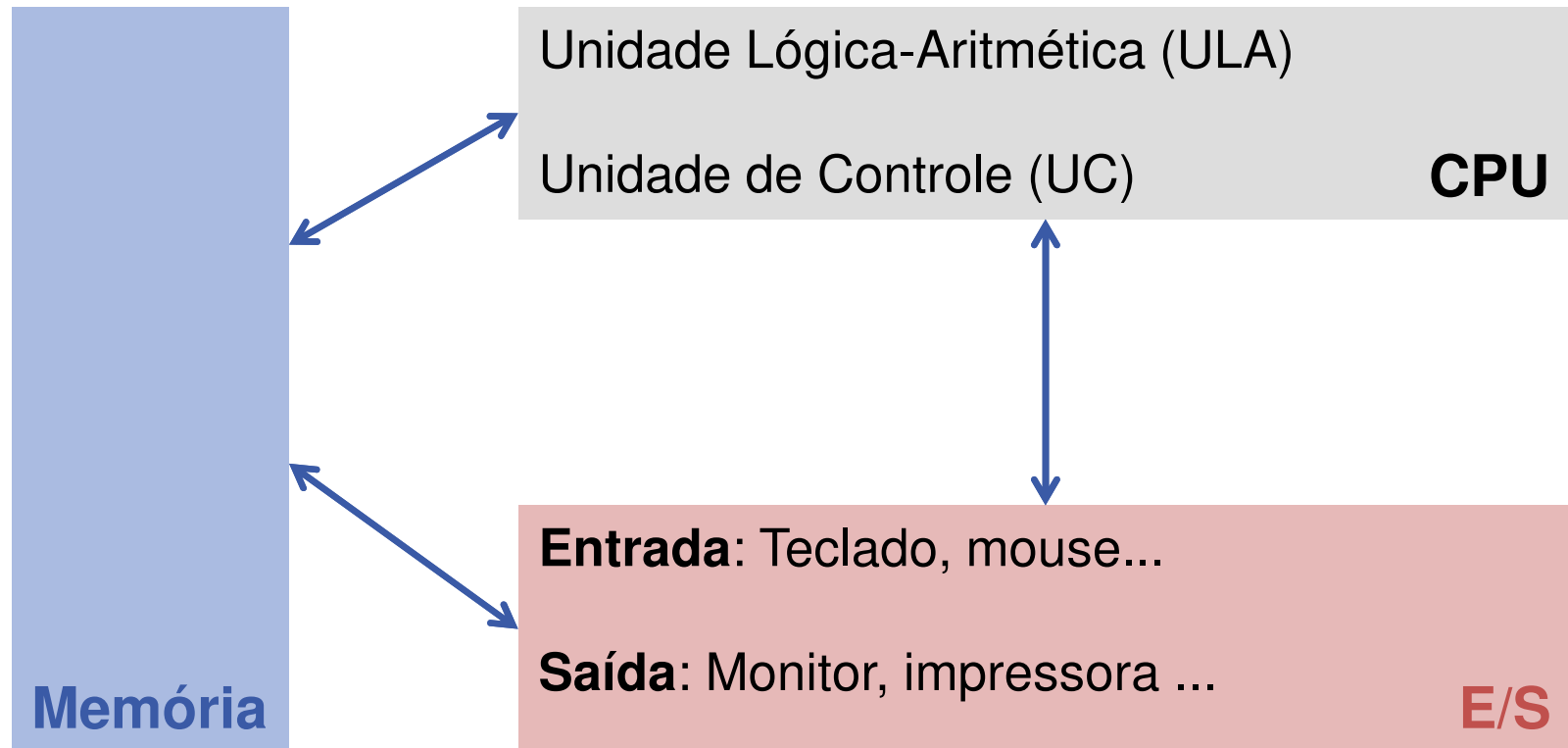
Agenda

1. Arquitetura de von Neumann
2. Ponteiros
 - Ponteiros e vetores
 - Passagem de parâmetro em C++
3. Testes e depuração
4. Qualidade de código

Arquitetura de von Neumann

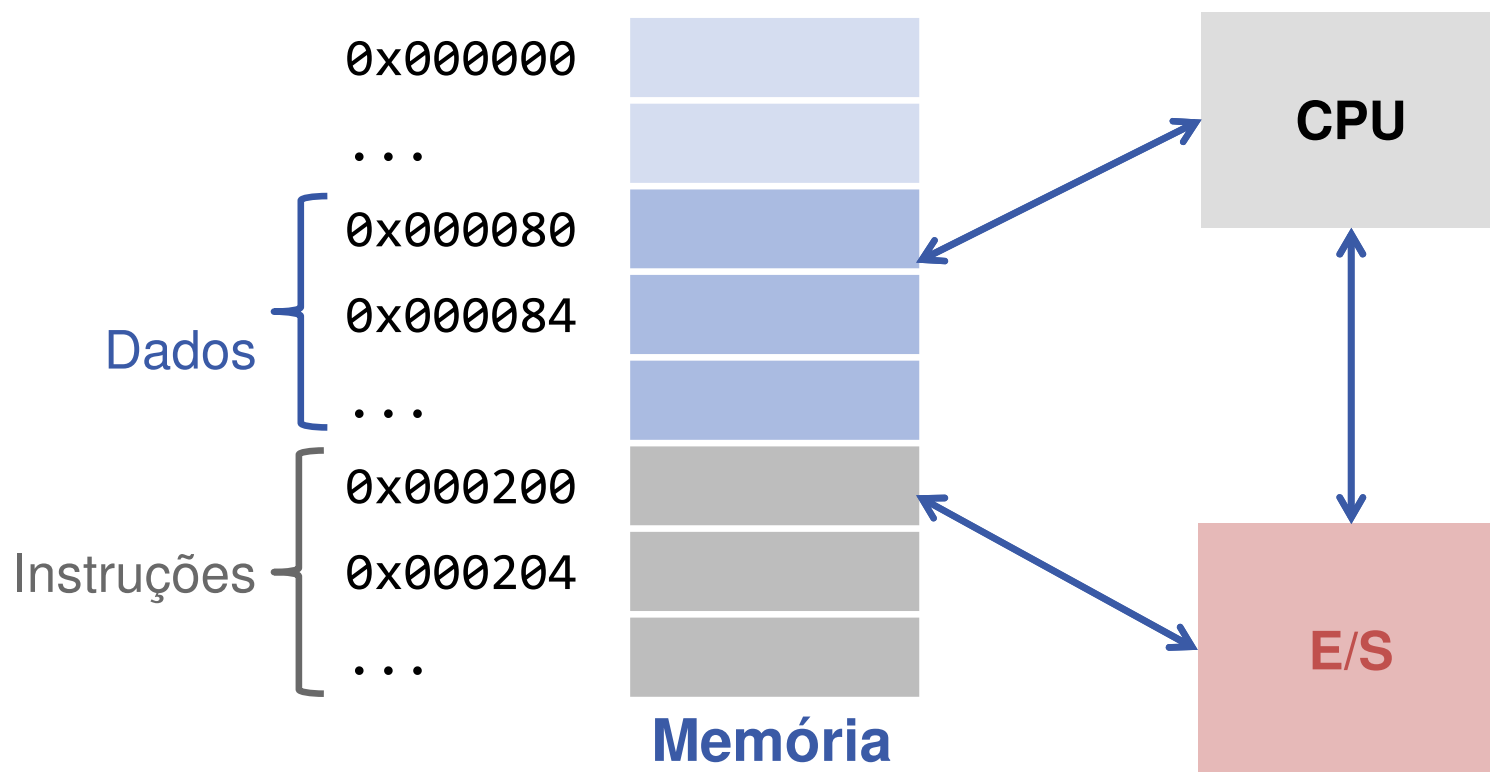
Computador

- Arquitetura de von Neumann



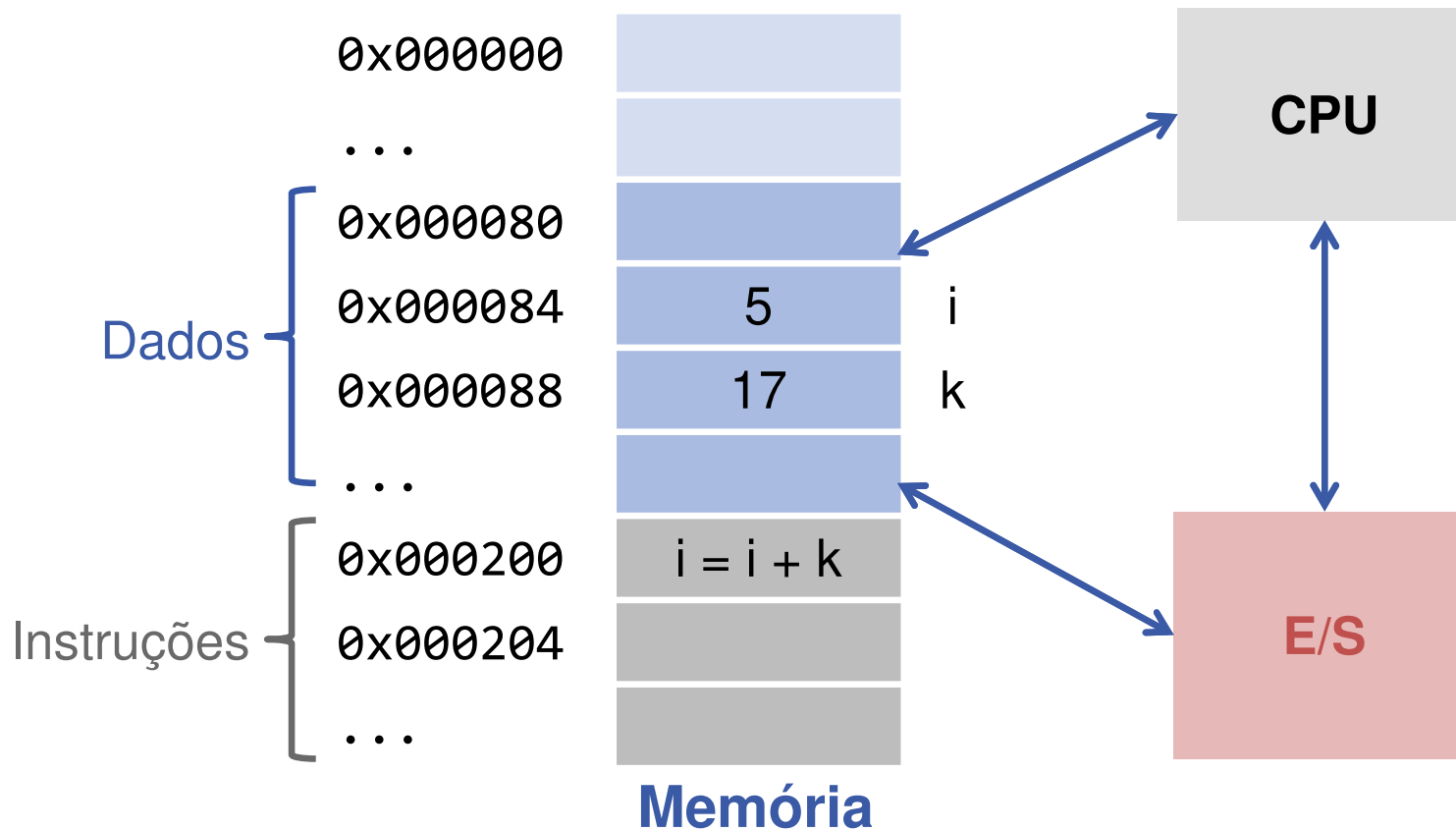
Arquitetura de von Neumann

- A memória é uma sequência de bytes
 - 1 byte = 8 bits
 - Bytes são numerados sequencialmente
 - **Dados** e instruções (programa) ficam na memória



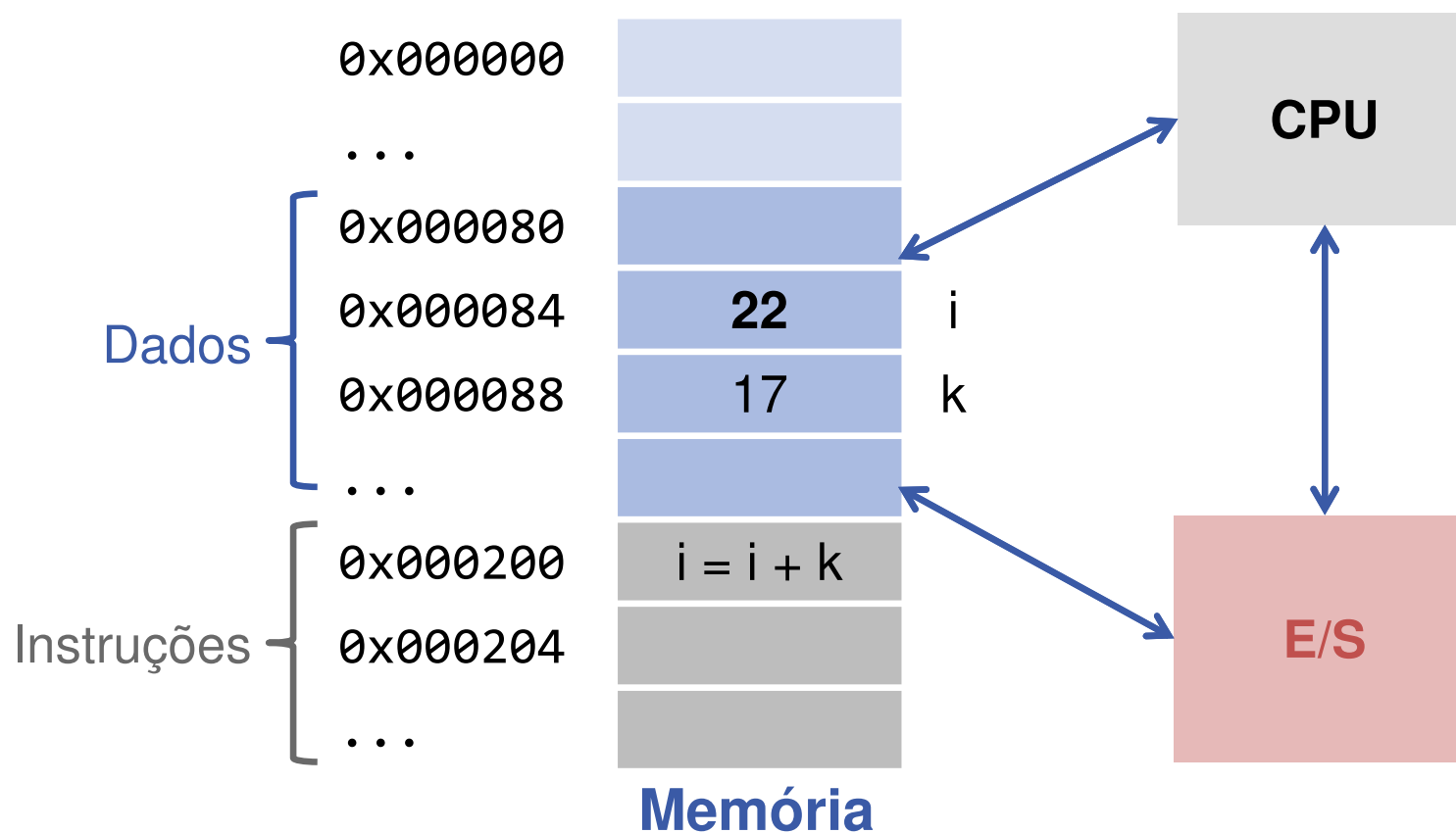
Arquitetura de von Neumann

- Exemplo: antes da execução



Arquitetura de von Neumann

- Exemplo: depois da execução

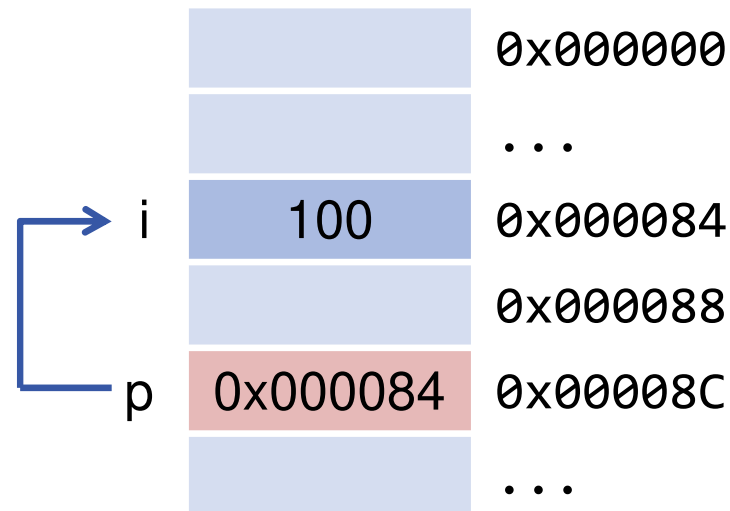


Ponteiros

Ponteiros

- Variável especial que referencia um endereço da memória
 - Também chamado de *apontador*

```
int i = 100;
```



- “p” é um ponteiro que aponta para o valor de “i”

Ponteiros

■ Declaração `<Tipo> *p;`

- O ponteiro é específico para um tipo de variável

```
12  int *p1;  
13  double *p2;
```

■ Operador &

- Obtêm o endereço de uma variável

```
6  int i = 100;  
7  double j = 5.5;  
8  
9  cout << &i << endl;  
10 cout << &j << endl;
```

p1 aponta para i

```
12  int *p1;  
13  double *p2;  
14  
15  p1 = &i;  
16  p2 = &j;  
17  
18  cout << p1 << endl;  
19  cout << p2 << endl;
```

EX01

Saída

```
0x6afef4  
0x6afee8  
0x6afef4  
0x6afee8
```

Ponteiros

- Operador * (*desreferenciação*)
 - Permite obter o valor apontado pelo ponteiro

```
6    int i = 5;
7    int *p = &i;
8
9    cout << i << endl;
10   cout << *p << endl;
```

EX02

Saída

5
5

- É possível usá-lo para alterar o valor

```
12   *p = 10;
13   cout << *p << endl;
14   cout << i << endl;
```

Saída

10
10

Usando um ponteiro

- O valor inicial de um ponteiro é indefinido

```
6    int *p1; // endereço indefinido
7
8    cout << *p1 << endl; // Ops... Problema!
```

EX03

■ NULL

- Representa que o ponteiro aponta para **nenhum** valor

```
10   int *p2; // endereço indefinido
11   p2 = NULL; // nenhum valor
12
13   if (p2 == NULL) { É possível testar
14       cout << "Null" << endl;
15   }
```

EX03

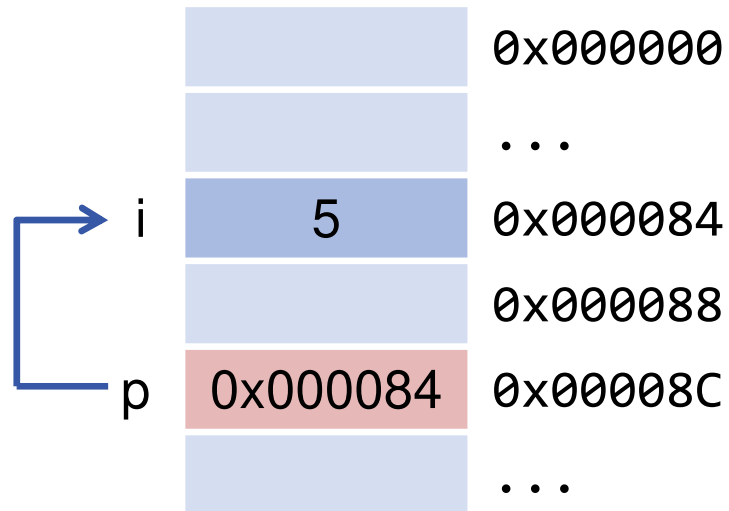
- NULL está definido em várias bibliotecas
 - Em `iostream`, por exemplo

Usando um ponteiro

- Na verdade, o ponteiro também está na memória

```
6  int i = 5;  
7  int *p = &i;
```

EX02



- Então qual é o valor de:

&i

p

*p

&p

*&i

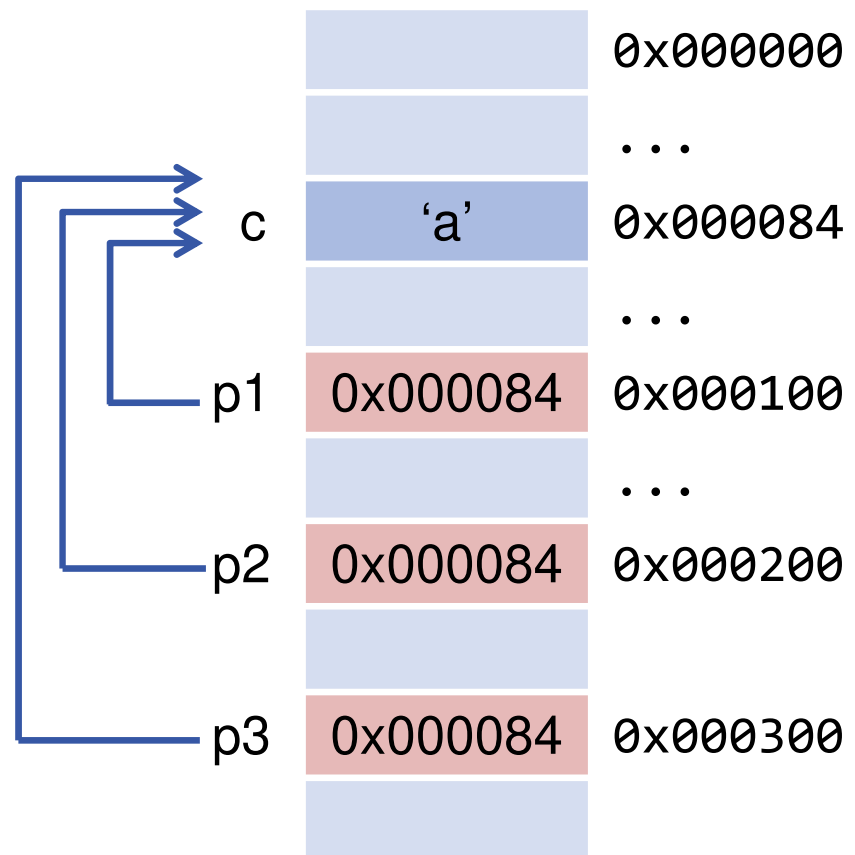
Usando um ponteiro

- Vários ponteiros podem apontar para o mesmo valor

```
6   char c = 'a'; EX04
7
8   char *p1 = &c;
9   char *p2 = &c;
10  char *p3 = p1;
11
12  cout << *p3 << endl;
13
14  *p3 = 'b';
15
16  cout << c << endl;
```

Saída

a
b



Usando um ponteiro

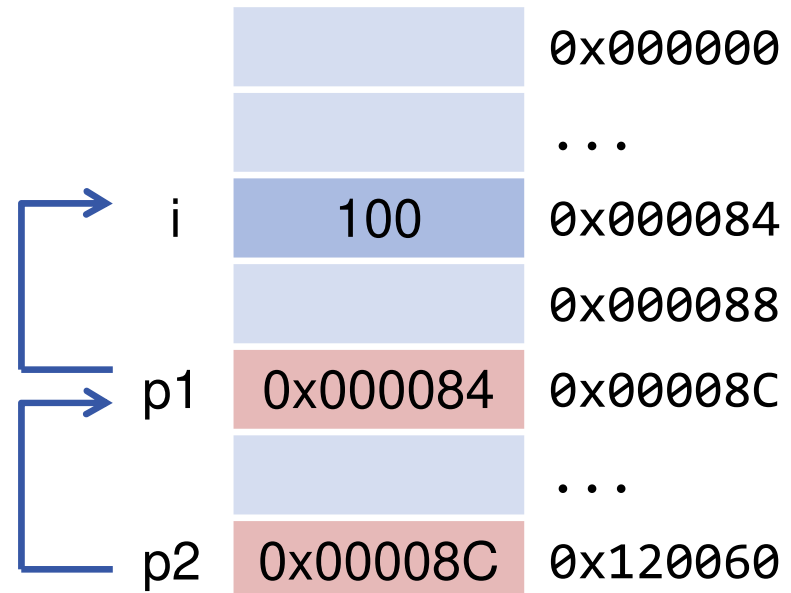
- É possível ter ponteiros de ponteiros

```
6  int i = 100;
7  int *p1 = &i;
8
9  int **p2;
10 p2 = &p1;
11
12 cout << i << endl;
13 cout << *p1 << endl;
14 cout << **p2 << endl;
```

EX05

Saída

```
100
100
100
```



p2 é um ponteiro para
ponteiro de inteiros

Ponteiros e vetores

- Um vetor *funciona* como um ponteiro
 - A variável aponta para a primeira posição do vetor

```
6    int a1[] = {1, 2, 5};
```

EX06

0x000084

a1



	0x000000
	...
1	0x000084
2	0x000088
5	0x00008C

- Um ponteiro para o vetor pode ser usado como um vetor

```
8    int *p = a1;  
9  
10   cout << p[0] << endl;  
11   cout << p[2] << endl;
```

Saída

1
5

Observação: em C++ não é possível retornar vetores. Mas é possível retornar um *ponteiro*.

Passagem de parâmetro

- Como funciona a passagem de parâmetros?
 - Qual é a saída em tela?

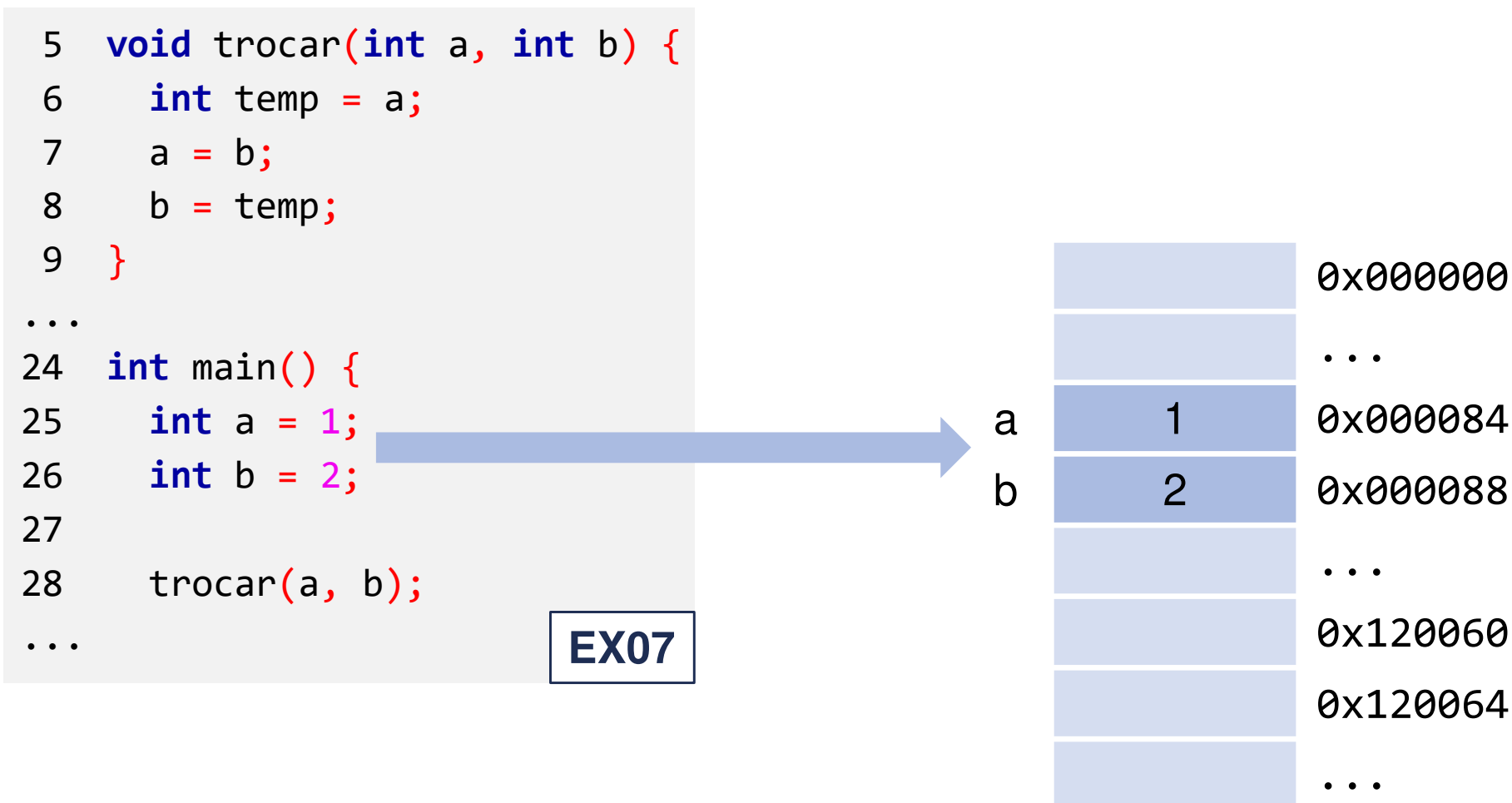
```
5 void trocar(int a, int b) {  
6     int temp = a;  
7     a = b;  
8     b = temp;  
9 }  
...  
24 int main() {  
25     int a = 1;  
26     int b = 2;  
27  
28     trocar(a, b);  
29     cout << "a: " << a << " b: " << b << endl;  
...  

```

EX07

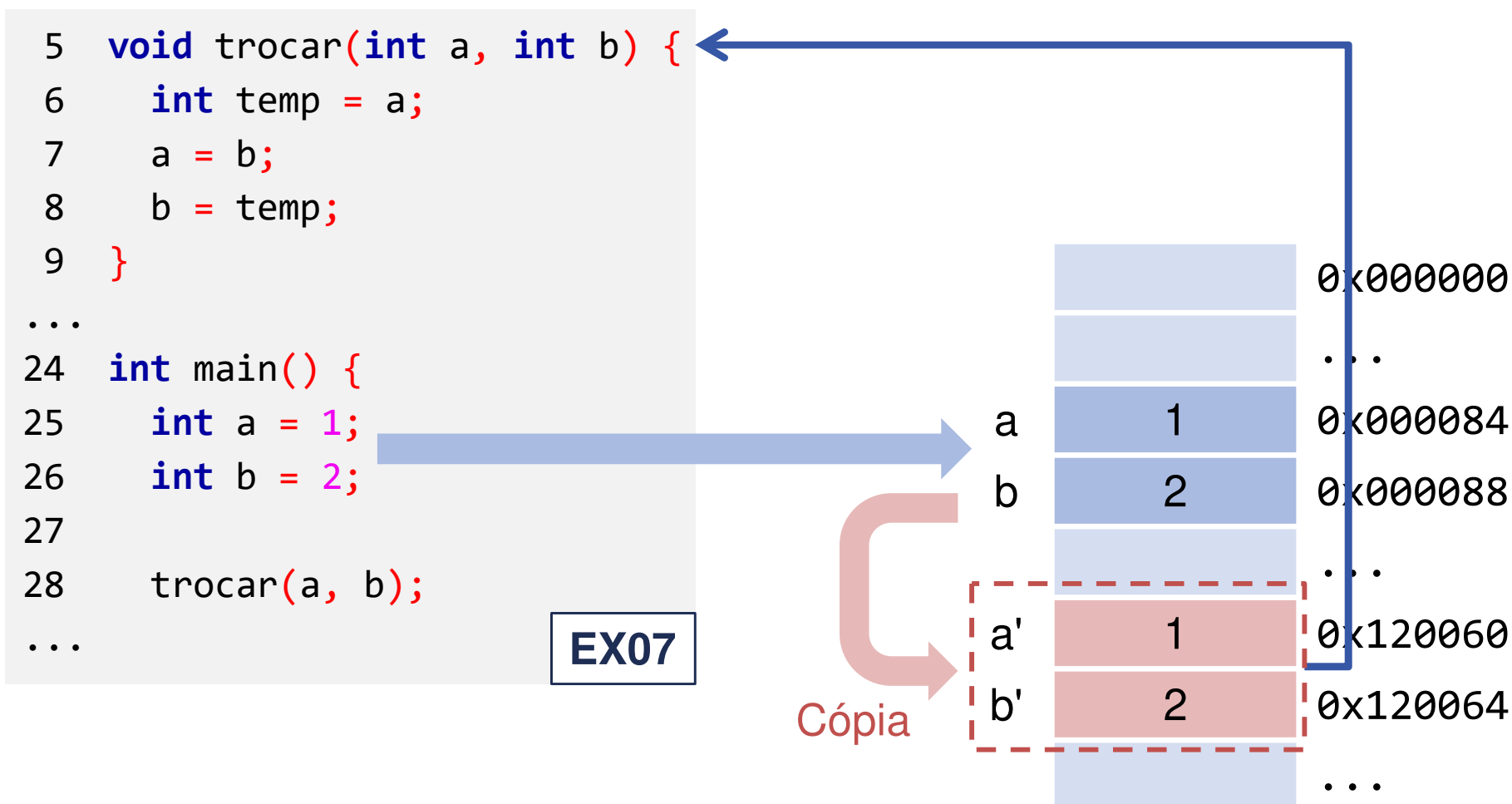
Passagem de parâmetro

- O C++ passa argumentos por **valor**
 - Ou seja, é feita uma **cópia** do valor da variável



Passagem de parâmetro

- O C++ passa argumentos por **valor**
 - Ou seja, é feita uma **cópia** do valor da variável



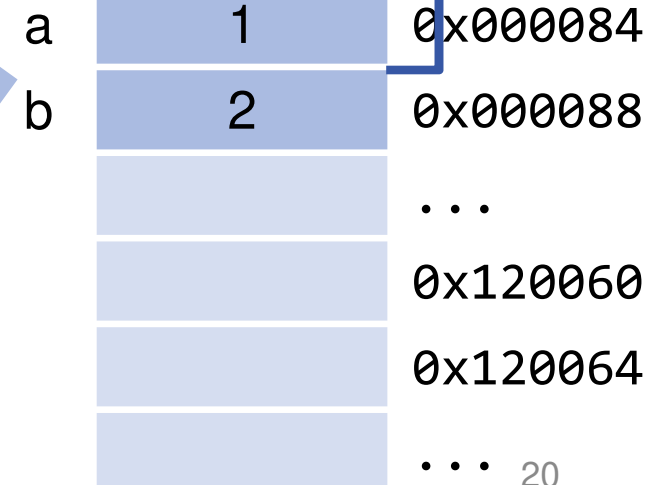
Passagem por referência

- C++ permite parâmetros por **referência**
 - Nesse caso a referência ao valor é passada como parâmetro
 - Usar o símbolo & na declaração do parâmetro

```
11 void trocar1(int& a, int& b) {  
12     int temp = a;  
13     a = b;  
14     b = temp;  
15 }
```

Por referência

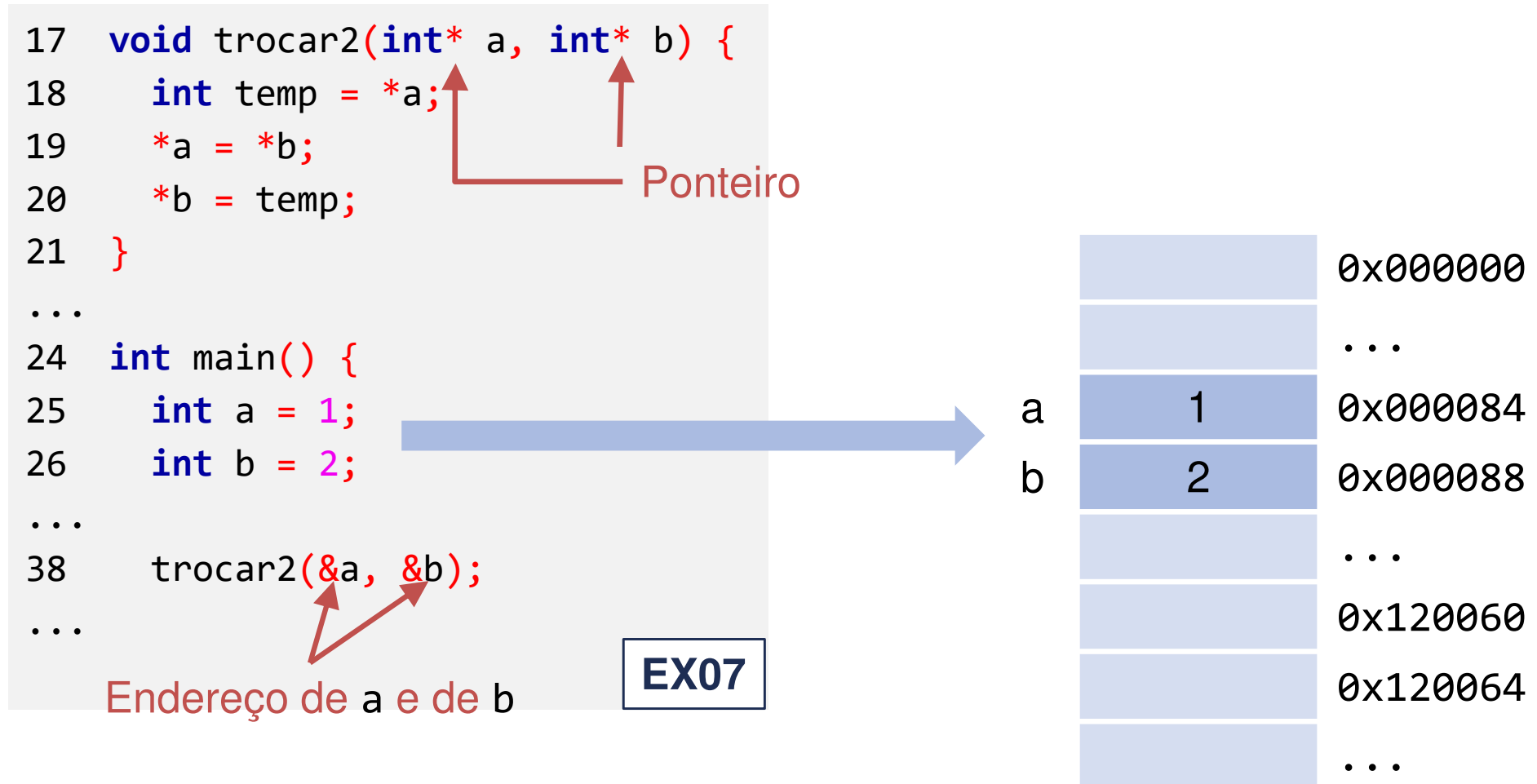
```
...  
24 int main() {  
25     int a = 1;  
26     int b = 2;  
...  
33     trocar1(a, b);  
...
```



EX07

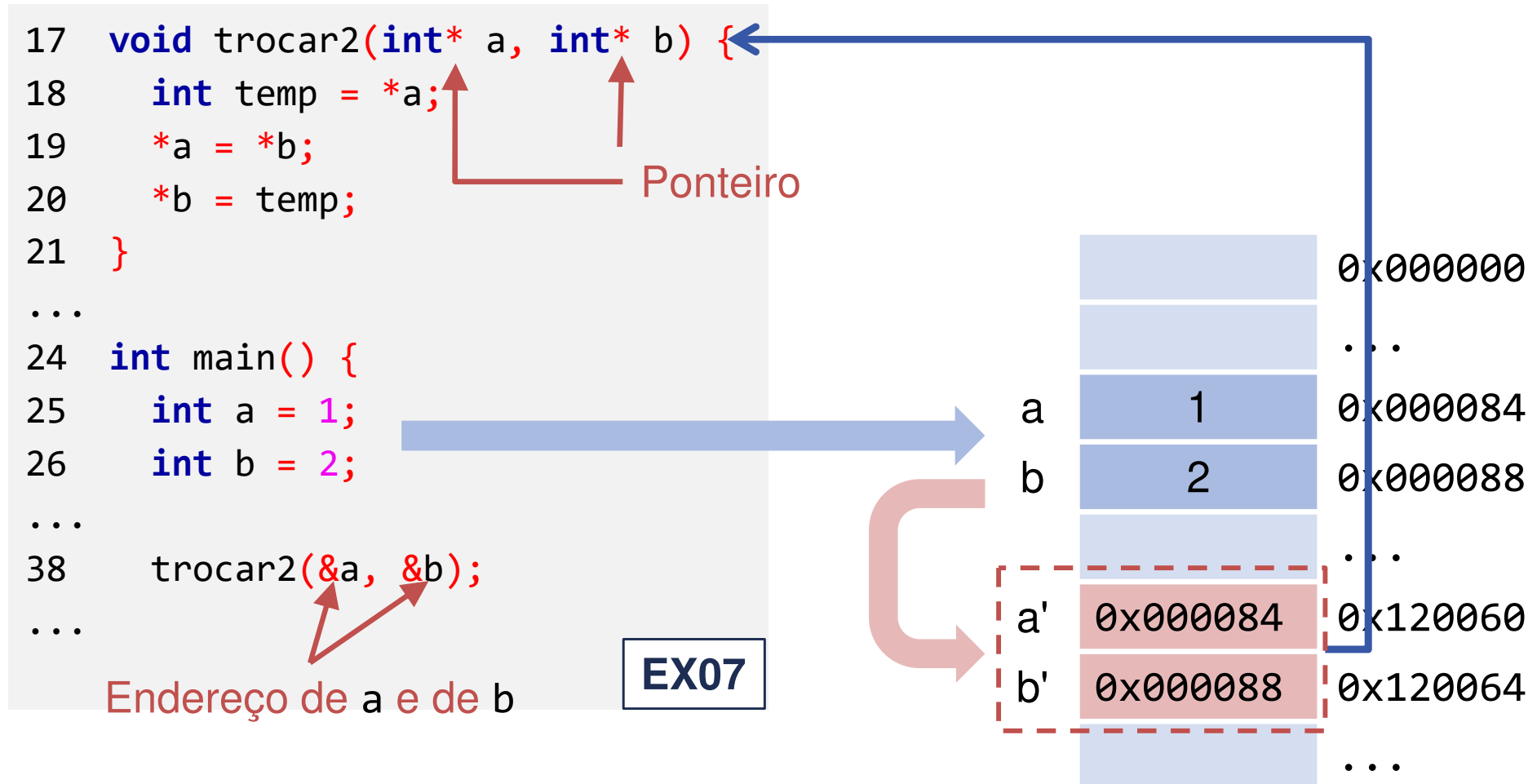
Passagem usando ponteiros

- Uma outra forma é usar ponteiros



Passagem usando ponteiros

- Uma outra forma é usar ponteiros



Bibliografia

- SAVITCH, W. **C++ Absoluto**. Pearson, 1st ed. 2003. Seção 10.1.