

Tecnologia Em Analise e Desenv. de Sistemas

Estrutura de Dados

Aula 4 – Ponteiros – Parte 1

Prof. Dr. Ricardo Luis Balieiro

Problema

```
int obtenerX();

int main() {
    int x;

    x = obtenerX();

    cout << "Valor de X: " << x << endl;

    return 0;
}

int obtenerX() {
    return 100;
}
```

Problema

- Como obter o retorno das três por meio de funções?

3 funções

```
int obterX();
int obterY();
int obterZ();

int main() {
    int x, y, z;

    x = obterX();
    y = obterY();
    z = obterZ();

    cout << "Valor de X: " << x << endl;
    cout << "Valor de Y: " << y << endl;
    cout << "Valor de Z: " << z << endl;

    return 0;
}

int obterX() {
    return 100;
}

int obterY() {
    return 200;
}

int obterZ() {
    return 300;
}
```

Problema

- Como obter o retorno das três variáveis utilizando uma única função?

```
Valor de X: 0
Valor de Y: 0
Valor de Z: 0
```

```
int obterX();
int obterY();
int obterZ();

int main() {
    int x, y, z;

    x = obterX();
    y = obterY();
    z = obterZ();

    cout << "Valor de X: " << x << endl;
    cout << "Valor de Y: " << y << endl;
    cout << "Valor de Z: " << z << endl;

    return 0;
}

int obterX() {
    return 100;
}

int obterY() {
    return 200;
}

int obterZ() {
    return 300;
}
```

Problema

```
void obterXYZ(int x, int y, int z);

int main() {
    int x, y, z;

    x = 0;
    y = 0;
    z = 0;

    obterXYZ(x, y, z);

    cout << "Valor de X: " << x << endl;
    cout << "Valor de Y: " << y << endl;
    cout << "Valor de Z: " << z << endl;

    system("pause");
    return 0;
}

void obterXYZ(int x, int y, int z) {
    x = 100;
    y = 200;
    z = 300;
}
```

Problema

```
void obterXYZ(int x, int y, int z);

int main() {
    int x, y, z;

    x = 0;
    y = 0;
    z = 0;

    obterXYZ(x, y, z);

    cout << "Valor de X: " << x << endl;
    cout << "Valor de Y: " << y << endl;
    cout << "Valor de Z: " << z << endl;

    system("pause");
    return 0;
}

void obterXYZ(int x, int y, int z) {
    x = 100;
    y = 200;
    z = 300;
}
```

Saída

Variáveis Estáticas

```
void obterXYZ(int x, int y, int z);

int main() {
    int x, y, z;

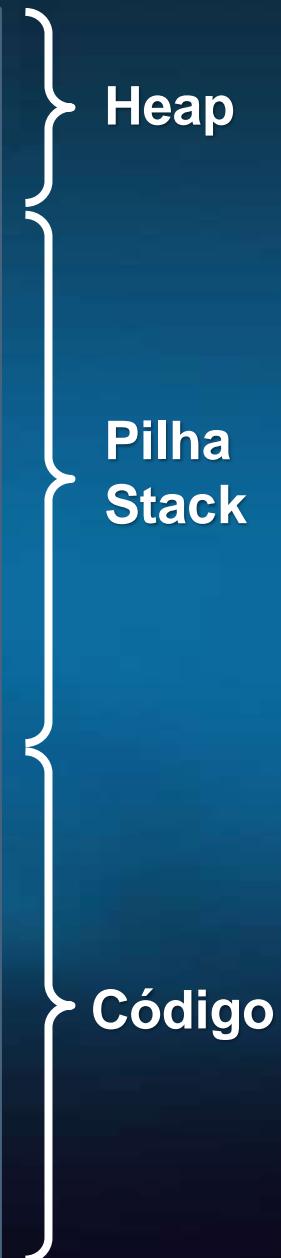
    x = 0;
    y = 0;
    z = 0;

    obterXYZ(x, y, z);

    cout << "Valor de X: " << x << endl;
    cout << "Valor de Y: " << y << endl;
    cout << "Valor de Z: " << z << endl;

    system("pause");
    return 0;
}

void obterXYZ(int x, int y, int z) {
    x = 100;
    y = 200;
    z = 300;
}
```



Problema

```
void obterXYZ(int x, int y, int z);

int main() {
    int x, y, z;

    x = 0;
    y = 0;
    z = 0;

    obterXYZ(x, y, z);

    cout << "Valor de X: " << x << endl;
    cout << "Valor de Y: " << y << endl;
    cout << "Valor de Z: " << z << endl;

    system("pause");
    return 0;
}

void obterXYZ(int x, int y, int z) {
    x = 100;
    y = 200;
    z = 300;
}
```

Saída

Variáveis Estáticas

```
void obterXYZ(int x, int y, int z);

int main() {
    int x, y, z;

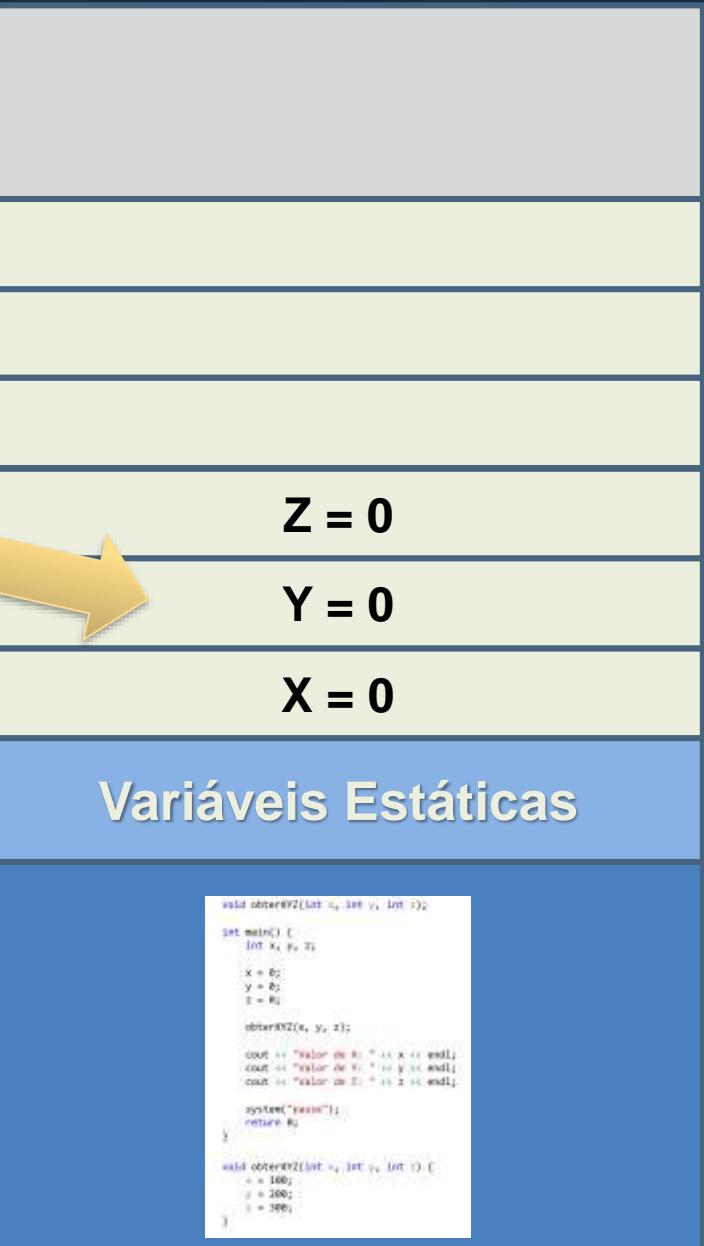
    x = 0;
    y = 0;
    z = 0;

    obterXYZ(x, y, z);

    cout << "Valor de X: " << x << endl;
    cout << "Valor de Y: " << y << endl;
    cout << "Valor de Z: " << z << endl;

    system("pause");
    return 0;
}

void obterXYZ(int x, int y, int z) {
    x = 100;
    y = 200;
    z = 300;
}
```



Heap

Pilha
Stack

Código

Problema

```
void obterXYZ(int x, int y, int z);

int main() {
    int x, y, z;

    x = 0;
    y = 0;
    z = 0;

    obterXYZ(x, y, z);

    cout << "Valor de X: " << x << endl;
    cout << "Valor de Y: " << y << endl;
    cout << "Valor de Z: " << z << endl;

    system("pause");
    return 0;
}

void obterXYZ(int x, int y, int z) {
    x = 100;
    y = 200;
    z = 300;
}
```

Saída

Z = 0
Y = 0
X = 0
Z = 0
Y = 0
X = 0

Variáveis Estáticas

```
void obterXYZ(int x, int y, int z);

int main() {
    int x, y, z;

    x = 0;
    y = 0;
    z = 0;

    obterXYZ(x, y, z);

    cout << "Valor de X: " << x << endl;
    cout << "Valor de Y: " << y << endl;
    cout << "Valor de Z: " << z << endl;

    system("pause");
    return 0;
}

void obterXYZ(int x, int y, int z) {
    x = 100;
    y = 200;
    z = 300;
}
```

Heap

Pilha
Stack

Código

Problema

```
void obterXYZ(int x, int y, int z);

int main() {
    int x, y, z;

    x = 0;
    y = 0;
    z = 0;

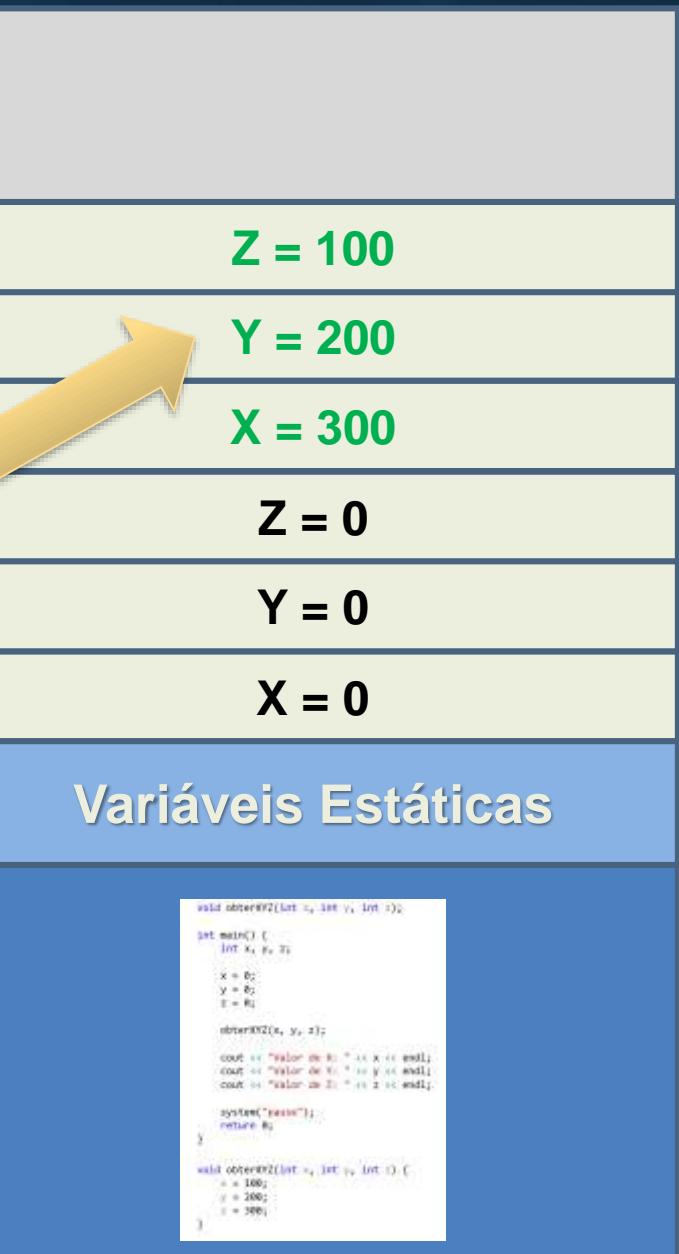
    obterXYZ(x, y, z);

    cout << "Valor de X: " << x << endl;
    cout << "Valor de Y: " << y << endl;
    cout << "Valor de Z: " << z << endl;

    system("pause");
    return 0;
}

void obterXYZ(int x, int y, int z) {
    x = 100;
    y = 200;
    z = 300;
}
```

Saída



Heap

Pilha
Stack

Código

Problema

```
void obterXYZ(int x, int y, int z);

int main() {
    int x, y, z;

    x = 0;
    y = 0;
    z = 0;

    obterXYZ(x, y, z);

    cout << "Valor de X: " << x << endl;
    cout << "Valor de Y: " << y << endl;
    cout << "Valor de Z: " << z << endl;

    system("pause");
    return 0;
}

void obterXYZ(int x, int y, int z) {
    x = 100;
    y = 200;
    z = 300;
}
```

Saída

Variáveis Estáticas

```
void obterXYZ(int x, int y, int z);

int main() {
    int x, y, z;

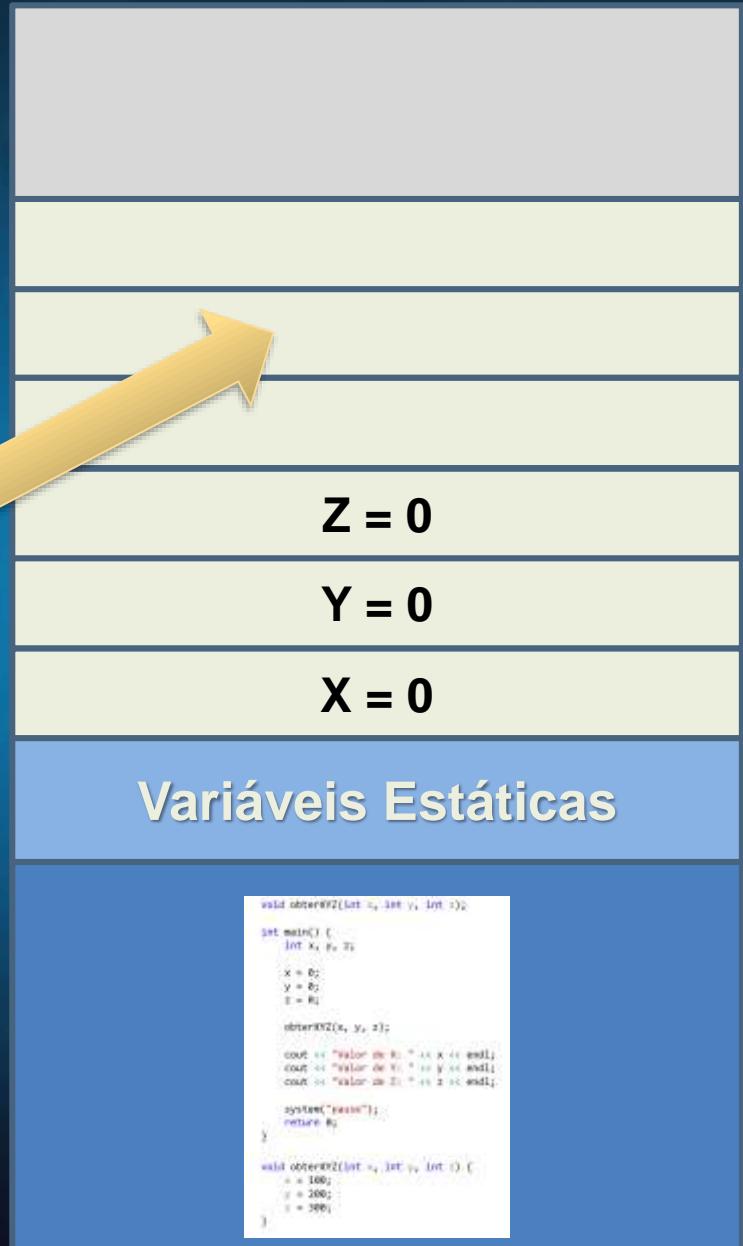
    x = 0;
    y = 0;
    z = 0;

    obterXYZ(x, y, z);

    cout << "Valor de X: " << x << endl;
    cout << "Valor de Y: " << y << endl;
    cout << "Valor de Z: " << z << endl;

    system("pause");
    return 0;
}

void obterXYZ(int x, int y, int z) {
    x = 100;
    y = 200;
    z = 300;
}
```



Heap

Pilha
Stack

Código

Problema

```
void obterXYZ(int x, int y, int z);

int main() {
    int x, y, z;

    x = 0;
    y = 0;
    z = 0;

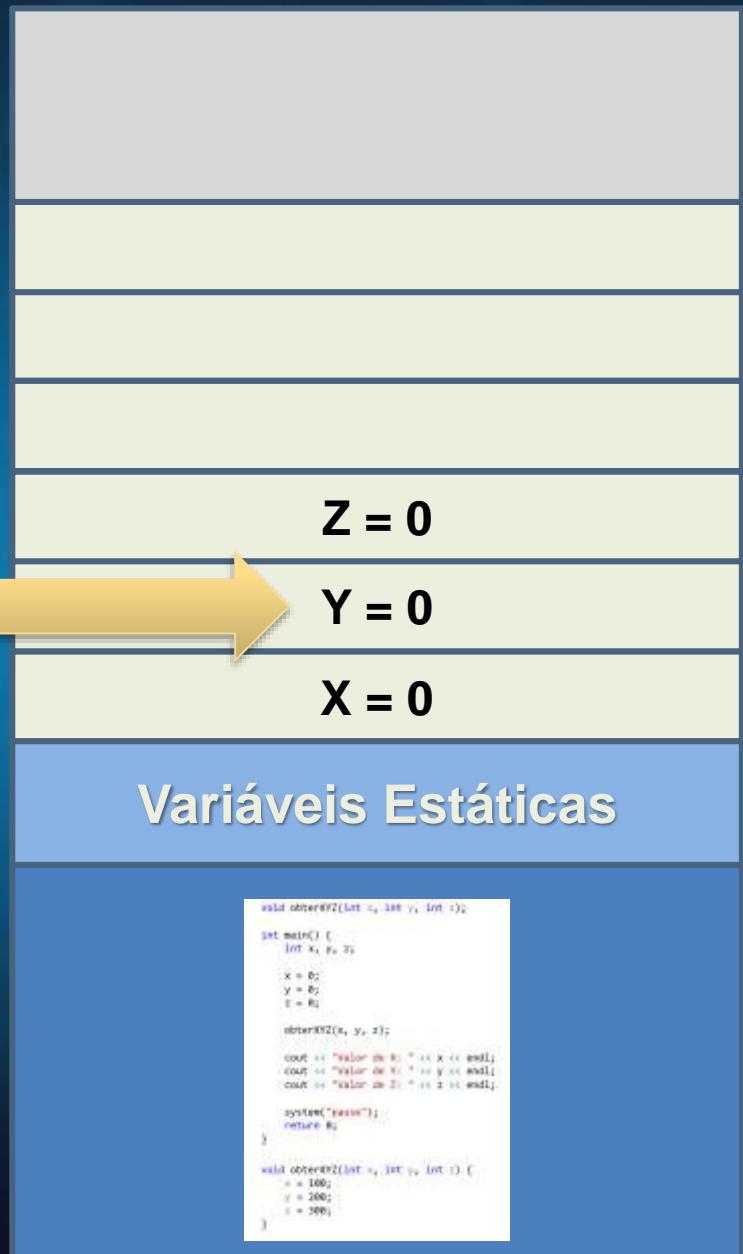
    obterXYZ(x, y, z);

    cout << "Valor de X: " << x << endl;
    cout << "Valor de Y: " << y << endl;
    cout << "Valor de Z: " << z << endl;

    system("pause");
    return 0;
}

void obterXYZ(int x, int y, int z) {
    x = 100;
    y = 200;
    z = 300;
}
```

Saída
0
0
0



Problema

```
void obterXYZ(int x, int y, int z);

int main() {
    int x, y, z;

    x = 0;
    y = 0;
    z = 0;

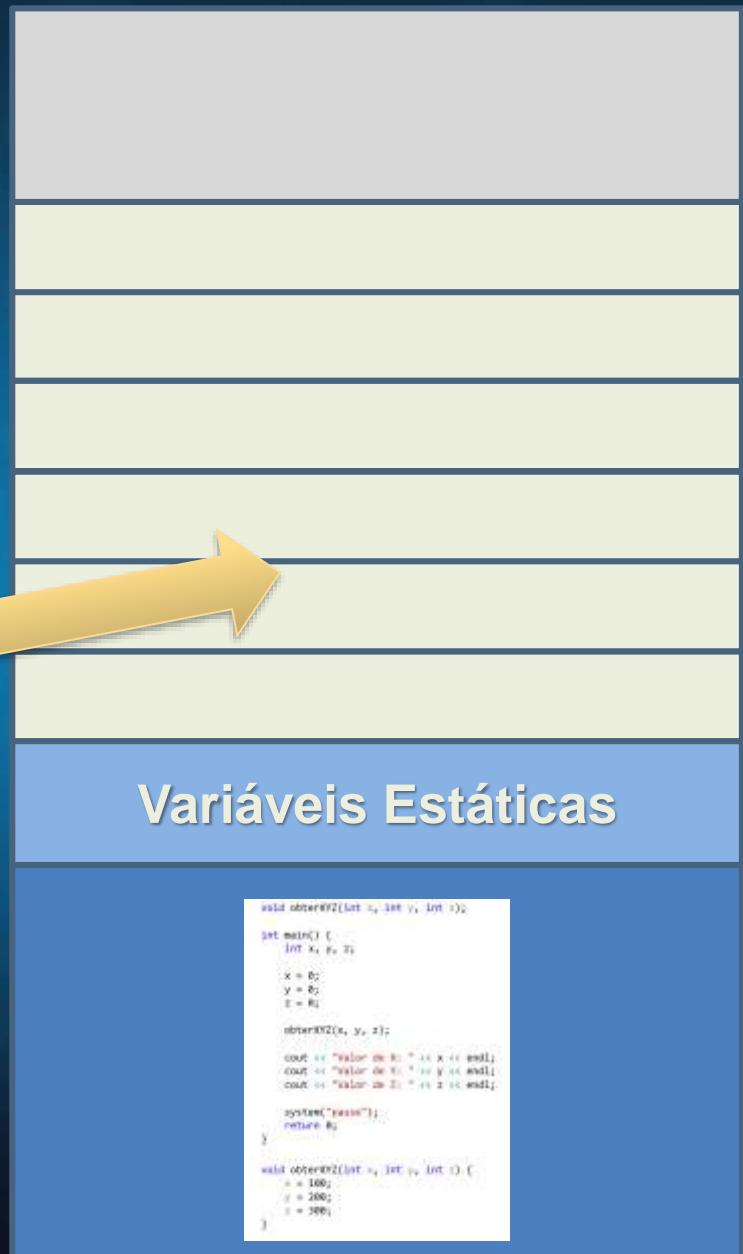
    obterXYZ(x, y, z);

    cout << "Valor de X: " << x << endl;
    cout << "Valor de Y: " << y << endl;
    cout << "Valor de Z: " << z << endl;

    system("pause");
    return 0;
}

void obterXYZ(int x, int y, int z) {
    x = 100;
    y = 200;
    z = 300;
}
```

Saída
0
0
0



Heap
Pilha Stack
Código

Zoom na memória

Ponteiros

Problema

```
void obterXYZ(int x, int y, int z);

int main() {
    int x, y, z;

    x = 0;
    y = 0;
    z = 0;

    obterXYZ(x, y, z);

    cout << "Valor de X: " << x << endl;
    cout << "Valor de Y: " << y << endl;
    cout << "Valor de Z: " << z << endl;

    system("pause");
    return 0;
}

void obterXYZ(int x, int y, int z) {
    x = 100;
    y = 200;
    z = 300;
}
```

Saída



Problema

```
void obterXYZ(int x, int y, int z);

int main() {
    int x, y, z;

    x = 0;
    y = 0;
    z = 0;

    obterXYZ(x, y, z);

    cout << "Valor de X: " << x << endl;
    cout << "Valor de Y: " << y << endl;
    cout << "Valor de Z: " << z << endl;

    system("pause");
    return 0;
}

void obterXYZ(int x, int y, int z) {
    x = 100;
    y = 200;
    z = 300;
}
```

Saída

&ndereço	Valor
0xA1	
0xA2	
0xA3	
0xA4	
0xA5	
0xA6	
0xA7	
0xA8	
0xA9	
0xAA	

Problema

```
void obterXYZ(int x, int y, int z);

int main() {
    int x, y, z;

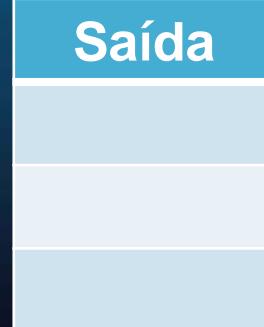
    x = 0;
    y = 0;
    z = 0;

    obterXYZ(x, y, z);

    cout << "Valor de X: " << x << endl;
    cout << "Valor de Y: " << y << endl;
    cout << "Valor de Z: " << z << endl;

    system("pause");
    return 0;
}

void obterXYZ(int x, int y, int z) {
    x = 100;
    y = 200;
    z = 300;
}
```



&ndereço	Valor
0xA1	
0xA2	
0xA3	
0xA4	
0xA5	
0xA6	
0xA7	
0xA8	
0xA9	
0xAA	

Problema

```
void obterXYZ(int x, int y, int z);

int main() {
    int x, y, z;

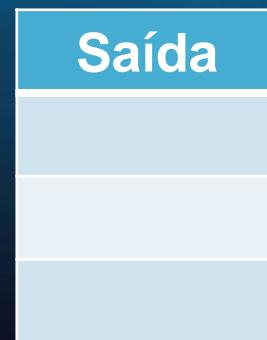
    x = 0;
    y = 0;
    z = 0;

    obterXYZ(x, y, z);

    cout << "Valor de X: " << x << endl;
    cout << "Valor de Y: " << y << endl;
    cout << "Valor de Z: " << z << endl;

    system("pause");
    return 0;
}

void obterXYZ(int x, int y, int z) {
    x = 100;
    y = 200;
    z = 300;
}
```



&ndereço	Valor
0xA1	
0xA2	
0xA3	
0xA4	
0xA5	
0xA6	
0xA7	
0xA8	
0xA9	
0xAA	

x
y
z

Problema

```
void obterXYZ(int x, int y, int z);

int main() {
    int x, y, z;

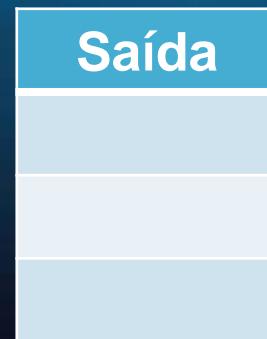
    x = 0;
    y = 0;
    z = 0;

    obterXYZ(x, y, z);

    cout << "Valor de X: " << x << endl;
    cout << "Valor de Y: " << y << endl;
    cout << "Valor de Z: " << z << endl;

    system("pause");
    return 0;
}

void obterXYZ(int x, int y, int z) {
    x = 100;
    y = 200;
    z = 300;
}
```



&ndereço	Valor
0xA1	
0xA2	
0xA3	
0xA4	
0xA5	
X	0xA6 0
y	0xA7 0
z	0xA8 0
0xA9	
0xAA	

Problema

```
void obterXYZ(int x, int y, int z);

int main() {
    int x, y, z;

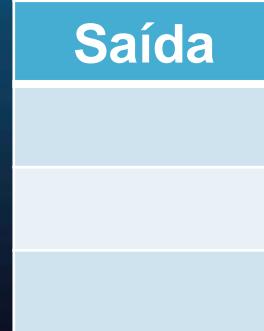
    x = 0;
    y = 0;
    z = 0;

    obterXYZ(x, y, z);

    cout << "Valor de X: " << x << endl;
    cout << "Valor de Y: " << y << endl;
    cout << "Valor de Z: " << z << endl;

    system("pause");
    return 0;
}

void obterXYZ(int x, int y, int z) {
    x = 100;
    y = 200;
    z = 300;
}
```



&ndereço	Valor
0xA1	
0xA2	0
0xA3	0
0xA4	0
0xA5	
0xA6	0
0xA7	0
0xA8	0
0xA9	
0xAA	

Problema

```
void obterXYZ(int x, int y, int z);

int main() {
    int x, y, z;

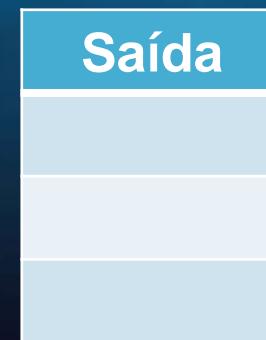
    x = 0;
    y = 0;
    z = 0;

    obterXYZ(x, y, z);

    cout << "Valor de X: " << x << endl;
    cout << "Valor de Y: " << y << endl;
    cout << "Valor de Z: " << z << endl;

    system("pause");
    return 0;
}

void obterXYZ(int x, int y, int z) {
    x = 100;
    y = 200;
    z = 300;
}
```



&ndereço	Valor
0xA1	
0xA2	100
0xA3	200
0xA4	300
0xA5	
0xA6	0
0xA7	0
0xA8	0
0xA9	
0xAA	

x
y
z



Problema

```
void obterXYZ(int x, int y, int z);

int main() {
    int x, y, z;

    x = 0;
    y = 0;
    z = 0;

    obterXYZ(x, y, z);

    cout << "Valor de X: " << x << endl;
    cout << "Valor de Y: " << y << endl;
    cout << "Valor de Z: " << z << endl;

    system("pause");
    return 0;
}

void obterXYZ(int x, int y, int z) {
    x = 100;
    y = 200;
    z = 300;
}
```



Problema

```
void obterXYZ(int x, int y, int z);
```

```
int main() {  
    int x, y, z;
```

```
    x = 0;  
    y = 0;  
    z = 0;
```

```
    obterXYZ(x, y, z);
```

```
    cout << "Valor de X: " << x << endl;  
    cout << "Valor de Y: " << y << endl;  
    cout << "Valor de Z: " << z << endl;
```

```
    system("pause");  
    return 0;
```

```
}
```

```
void obterXYZ(int x, int y, int z) {  
    x = 100;  
    y = 200;  
    z = 300;  
}
```

Saída
0
0
0

&ndereço	Valor
0xA1	
0xA2	
0xA3	
0xA4	
0xA5	
X	0xA6 0
y	0xA7 0
z	0xA8 0
	0xA9
	0xAA



Problema

```
void obterXYZ(int x, int y, int z);

int main() {
    int x, y, z;

    x = 0;
    y = 0;
    z = 0;

    obterXYZ(x, y, z);

    cout << "Valor de X: " << x << endl;
    cout << "Valor de Y: " << y << endl;
    cout << "Valor de Z: " << z << endl;

    system("pause");
    return 0;
}

void obterXYZ(int x, int y, int z) {
    x = 100;
    y = 200;
    z = 300;
}
```

Saída
0
0
0

&ndereço	Valor
0xA1	
0xA2	
0xA3	
0xA4	
0xA5	
0xA6	
0xA7	
0xA8	
0xA9	
0xAA	

Endereços de variáveis

Ponteiros

Endereços de variáveis

```
int main() {
    setlocale(LC_ALL, "Portuguese");

    int x;

    x = 100;

    cout << "Valor de X: " << x << endl;

    system("pause");
    return 0;
}
```



&ndereço	Valor
0xA1	
0xA2	100
0xA3	
0xA4	
0xA5	
0xA6	
0xA7	
0xA8	
0xA9	
0xAA	

Saída
100

Endereços de variáveis

```
int main() {  
    setlocale(LC_ALL, "Portuguese");  
  
    int x;  
  
    x = 100;  
  
    cout << "Endereço de X: " << &x << endl;  
    cout << "Valor de X: " << x << endl;  
  
    system("pause");  
    return 0;  
}
```



&ndereço	Valor
0xA1	
0xA2	100
0xA3	
0xA4	
0xA5	
0xA6	
0xA7	
0xA8	
0xA9	
0xAA	

Saída
0xA2
100

Exercício

- Crie 3 variáveis, uma do tipo int, outra do tipo float e a terceira do tipo char. Atribua valores para as mesmas e em seguida, exiba os dados e os seus endereços de memória.

Ponteiros

Ponteiros

Ponteiros

&ndereço	Valor
0xA1	
0xA2	100
0xA3	30.50
0xA4	R
0xA5	
0xA6	
0xA7	0xA2
0xA8	
0xA9	
0xAA	

int
float
char

Endereço

Ponteiros



Ponteiros são um tipo especial de variáveis que permitem armazenam endereços de memória ao invés de dados numéricos (como os tipos **int**, **float** e **double**) ou caracteres (como o tipo **char**).

Ponteiros

```
int main() {
    setlocale(LC_ALL, "Portuguese");

    int x;
    int *pX;

    x = 10;

    pX = &x;

    cout << "Endereço de &x: " << &x << endl;
    cout << "Valor de X: " << x << endl;

    cout << "Endereço de &pX: " << &pX << endl;
    cout << "Conteúdo de pX: " << pX << endl;
    cout << "Valor de *pX: " << *pX << endl;

    system("pause");
    return 0;
}
```



&ndereço	Valor
0xA1	
0xA2	
0xA3	
0xA4	
0xA5	
0xA6	
0xA7	
0xA8	
0xA9	
0xAA	

Ponteiros

```
int main() {
    setlocale(LC_ALL, "Portuguese");

    int x;
    int *pX;

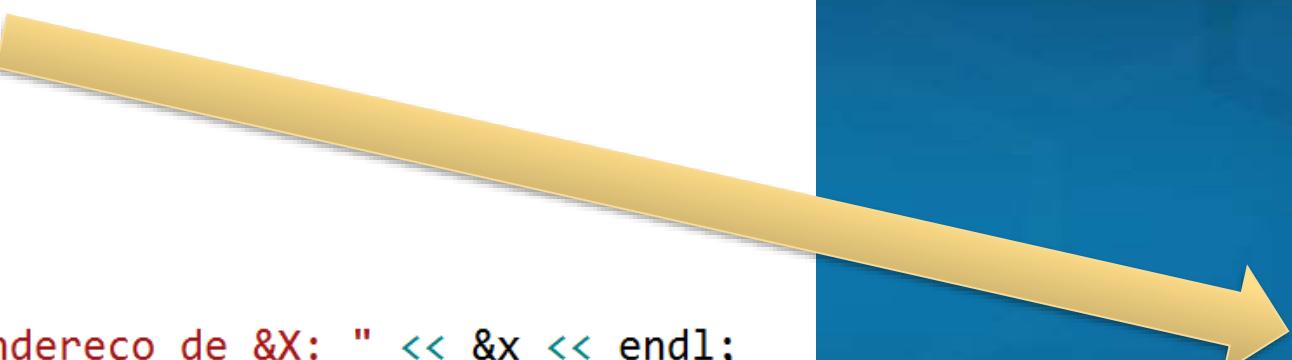
    x = 10;

    pX = &x;

    cout << "Endereço de &X: " << &x << endl;
    cout << "Valor de X: " << x << endl;

    cout << "Endereço de &pX: " << &pX << endl;
    cout << "Conteúdo de pX: " << pX << endl;
    cout << "Valor de *pX: " << *pX << endl;

    system("pause");
    return 0;
}
```



&ndereço	Valor
0xA1	
0xA2	
0xA3	
0xA4	
0xA5	
0xA6	
0xA7	
0xA8	
0xA9	
0xAA	

Ponteiros

```
int main() {
    setlocale(LC_ALL, "Portuguese");

    int x;
    int *pX;

    x = 10; x = 10;

    pX = &x;

    cout << "Endereço de &X: " << &x << endl;
    cout << "Valor de X: " << x << endl;

    cout << "Endereço de &pX: " << &pX << endl;
    cout << "Conteúdo de pX: " << pX << endl;
    cout << "Valor de *pX: " << *pX << endl;

    system("pause");
    return 0;
}
```

&Endereço	Valor
0xA1	
0xA2	10
0xA3	
0xA4	
0xA5	
0xA6	
0xA7	
0xA8	
0xA9	
0xAA	

Ponteiros

```
int main() {
    setlocale(LC_ALL, "Portuguese");

    int x;
    int *pX;

    x = 10;

    pX = &x; // A large yellow arrow points from this line to the memory diagram

    cout << "Endereço de &X: " << &x << endl;
    cout << "Valor de X: " << x << endl;

    cout << "Endereço de &pX: " << &pX << endl;
    cout << "Conteúdo de pX: " << pX << endl;
    cout << "Valor de *pX: " << *pX << endl;

    system("pause");
    return 0;
}
```

&Endereço	Valor
0xA1	
0xA2	10
0xA3	
0xA4	
0xA5	
0xA6	0xA2
0xA7	
0xA8	
0xA9	
0xAA	

Ponteiros

```
int main() {
    setlocale(LC_ALL, "Portuguese");

    int x;
    int *pX;

    x = 10;

    pX = &x;

    cout << "Endereço de &X: " << &x << endl;
    cout << "Valor de X: " << x << endl;

    cout << "Endereço de &pX: " << &pX << endl;
    cout << "Conteúdo de pX: " << pX << endl;
    cout << "Valor de *pX: " << *pX << endl;

    system("pause");
    return 0;
}
```



&ndereço	Valor
0xA1	
0xA2	10
0xA3	
0xA4	
0xA5	
pX	0xA2
0xA6	0xA2
0xA7	
0xA8	
0xA9	
0xAA	

Ponteiros

```
int main() {
    setlocale(LC_ALL, "Portuguese");

    int x;
    int *pX;

    x = 10;

    pX = &x;

    cout << "Endereço de &X: " << &x << endl;
    cout << "Valor de X: " << x << endl;

    cout << "Endereço de &pX: " << &pX << endl;
    cout << "Conteúdo de pX: " << pX << endl;
    cout << "Valor de *pX: " << *pX << endl;

    system("pause");
    return 0;
}
```

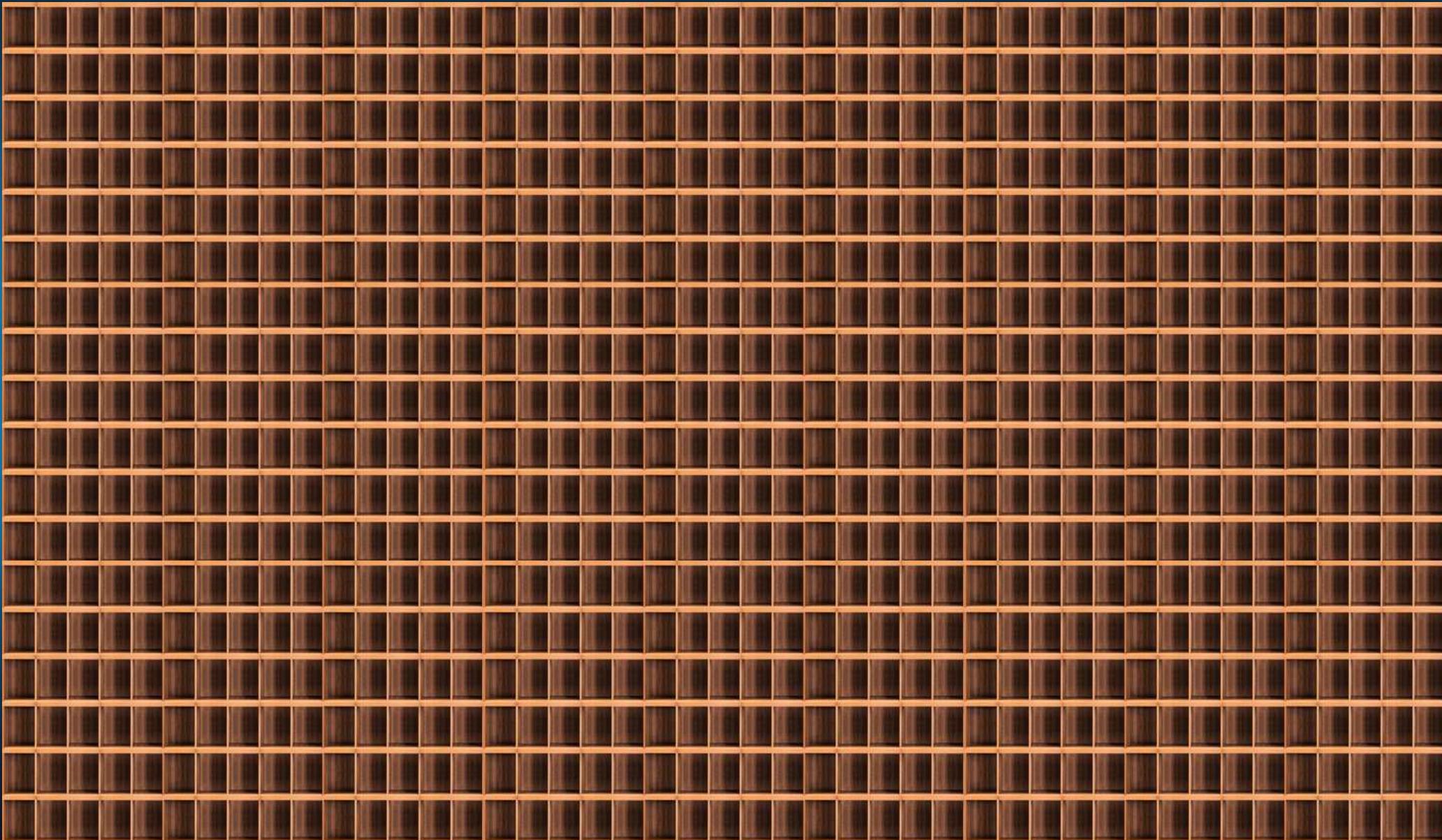
A diagram illustrating pointer behavior. On the left, a red box highlights the last three lines of the C++ code. A large yellow arrow points from this highlighted area to a memory dump table on the right. The table has two columns: 'Endereço' (Address) and 'Valor' (Value). The address column lists memory locations from 0xA1 to 0xAA. The value column shows the contents at those addresses. The entry for address 0xA2 contains the value 10, which is also highlighted in red. Address 0xA6 contains the value 0xA2, also highlighted in red, indicating that pX holds the address of x.

&ndereço	Valor
0xA1	
0xA2	10
0xA3	
0xA4	
0xA5	
0xA6	0xA2
0xA7	
0xA8	
0xA9	
0xAA	

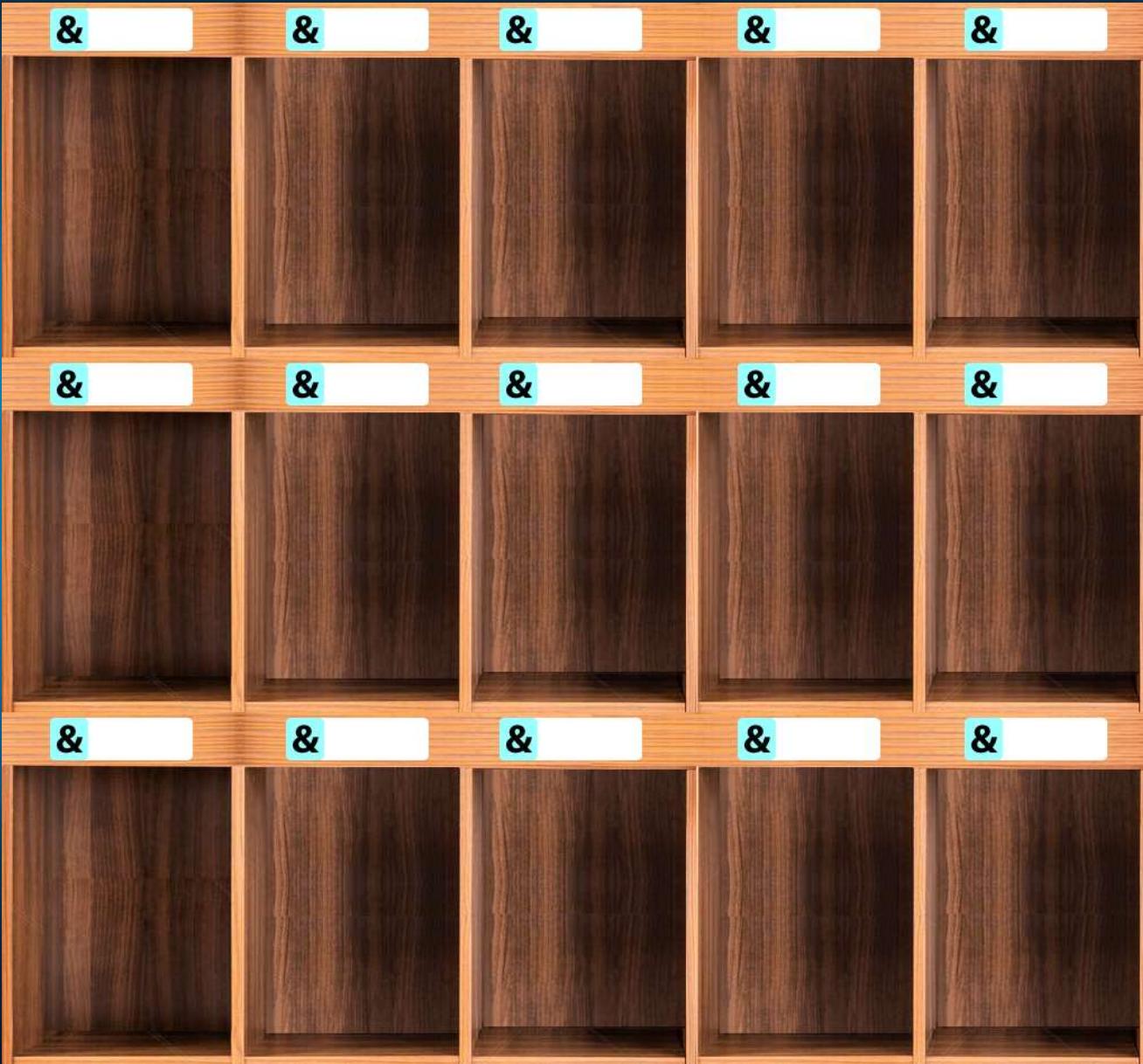
```
int main() {  
    int x, z;  
    int *pH, **pC, ***pP;  
  
    x = 8;  
    z = x;  
  
    pH = &x;  
    pC = &pH;  
    pP = &pC;  
  
    cout << "Endereço de &pH: " << &pH << endl;  
    cout << "Conteúdo de pH: " << pH << endl;  
    cout << "Valor de *pH: " << *pH << endl << endl;  
  
    cout << "Endereço de &pC: " << &pC << endl;  
    cout << "Conteúdo de pC: " << pC << endl;  
    cout << "Valor de *pC: " << *pC << endl;  
    cout << "Valor de **pC: " << **pC << endl << endl;  
  
    cout << "Endereço de &pP: " << &pP << endl;  
    cout << "Conteúdo de pP: " << pP << endl;  
    cout << "Valor de *pP: " << *pP << endl;  
    cout << "Valor de **pP: " << **pP << endl << endl;  
    cout << "Valor de ***pP: " << ***pP << endl << endl;  
}
```

O que será
exibido na tela?

Ponteiros



Ponteiros



Ponteiros

& 0xA1

& 0xA2

& 0xA3

& 0xA4

& 0xA5



Ponteiros

X

?

0xA4

& 0xA5



Ponteiros

Xavier



Ponteiros

Xavier



& 0xA1

& 0xA2

& 0xA3

& 0xA4

& 0xA5



Ponteiros

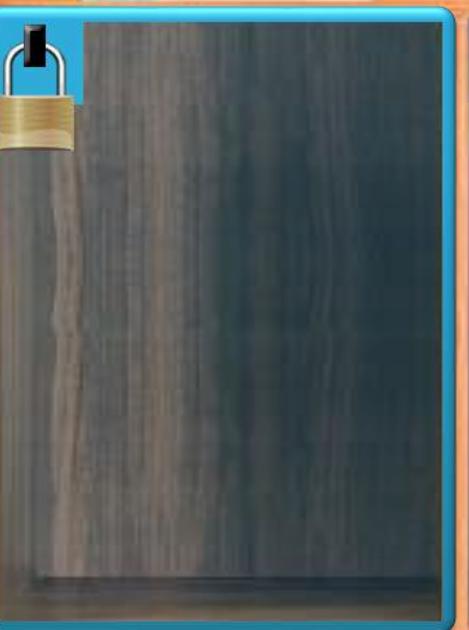
Xavier



& 0xA1



& 0xA2



& 0xA3



& 0xA4



& 0xA5



Ponteiros

Xavier



Z

?

0xA5

Ponteiros

Xavier



& 0xA1



Zorro



0xA5



Ponteiros

Xavier



Zorro



& 0xA1



& 0xA2



& 0xA3



& 0xA4



& 0xA5



Ponteiros



Ponteiros



$Z = X;$

Ponteiros

Xavier



Zorro



& 0xA1



& 0xA2



& 0xA3



& 0xA4



& 0xA5



Ponteiros

Xavier



& 0xA1

& 0xA2

& 0xA3

& 0xA4

& 0xA5



Ponteiros

Xavier



& 0xA1

& 0xA2

& 0xA3

& 0xA4

& 0xA5



Ponteiros

Xavier



& 0xA1

& 0xA2

& 0xA3

& 0xA4

& 0xA5



Ponteiros

Xavier



& 0xA1

& 0xA2

& 0xA3

& 0xA4

& 0xA5



Ponteiros

Xavier



& 0xA1

& 0xA2

& 0xA3

& 0xA4

& 0xA5



Ponteiros

Xavier



Zorro



& 0xA1



& 0xA2



& 0xA3



& 0xA4

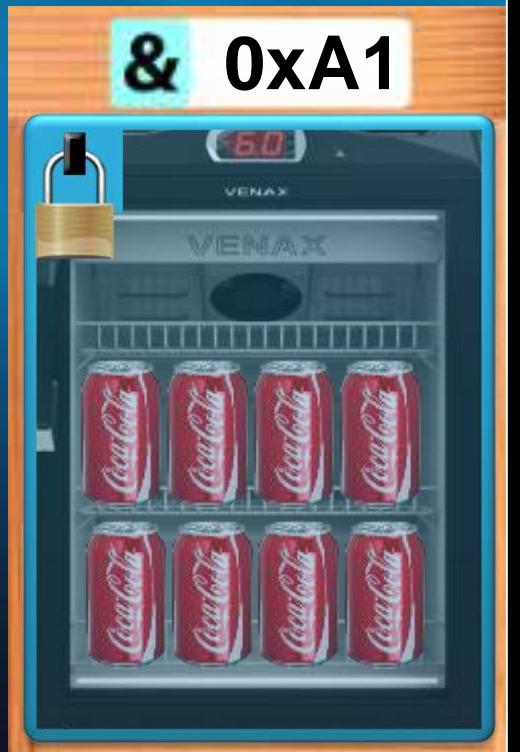


& 0xA5



Ponteiros

Xavier



PH

?

0xA5

Ponteiros

Xavier



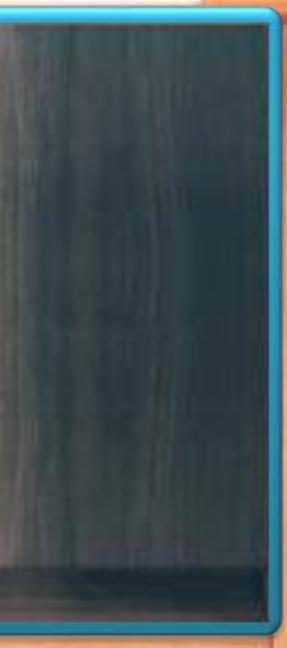
& 0xA1



Paulo Henrique



0xA5



Ponteiros

Xavier



Zorro



Paulo H enrique



& 0xA1



& 0xA2



& 0xA3



& 0xA4



& 0xA5



Ponteiros



Ponteiros



Ponteiros



PH = &X;

Ponteiros

Xavier



Zorro



Paulo H enrique



& 0xA1



& 0xA2



& 0xA3



& 0xA4



& 0xA5



Ponteiros

Xavier



Zorro



& 0xA1



& 0xA2



& 0xA3



& 0xA4



& 0xA5



Ponteiros

Xavier



Zorro



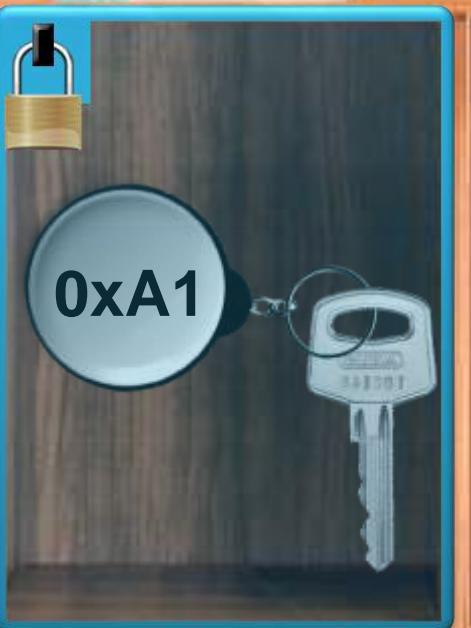
& 0xA1



& 0xA2



& 0xA3



& 0xA4



& 0xA5



Ponteiros

Xavier



Zorro



& 0xA1



& 0xA2



& 0xA3



& 0xA4



& 0xA5



Ponteiros

Xavier



Zorro



& 0xA1



& 0xA2



& 0xA3



& 0xA4



& 0xA5



Ponteiros

Xavier



Zorro



& 0xA1



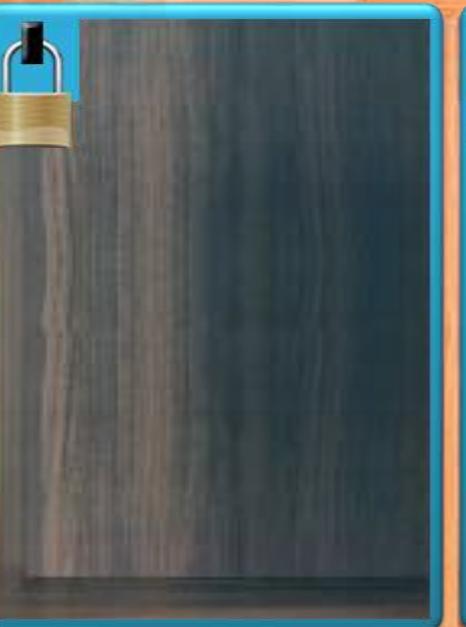
& 0xA2



& 0xA3



& 0xA4



& 0xA5



Ponteiros

Xavier



Zorro



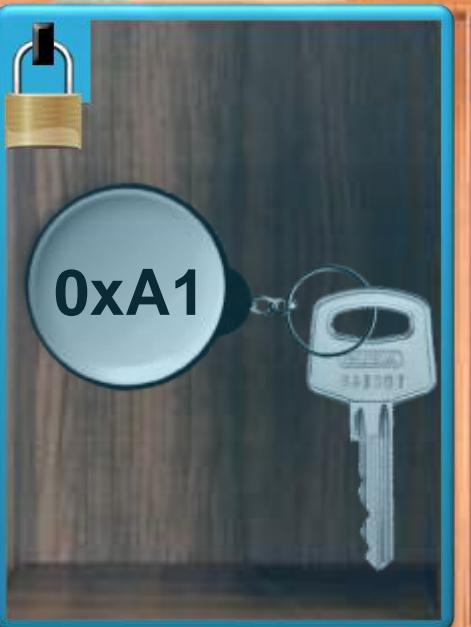
& 0xA1



& 0xA2



& 0xA3



& 0xA4



& 0xA5



Ponteiros

Xavier



Zorro



& 0xA1



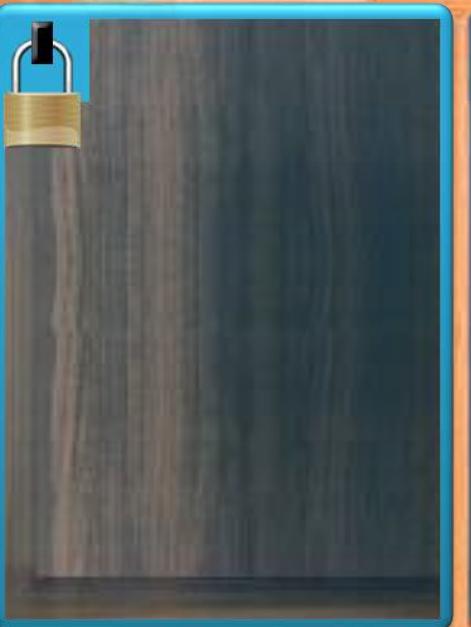
& 0xA2



& 0xA3



& 0xA4



& 0xA5



Ponteiros

Xavier



Zorro



& 0xA1



& 0xA2



& 0xA3



& 0xA4



& 0xA5



Ponteiros

Xavier



Zorro



Paulo H enrique



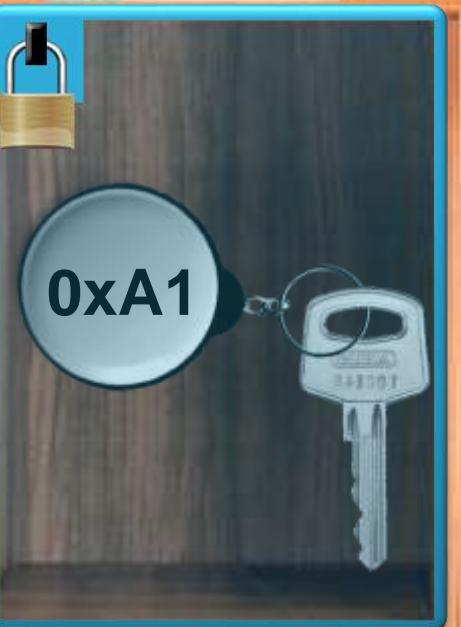
& 0xA1



& 0xA2



& 0xA3



& 0xA4



& 0xA5



Ponteiros

Xavier



Zorro



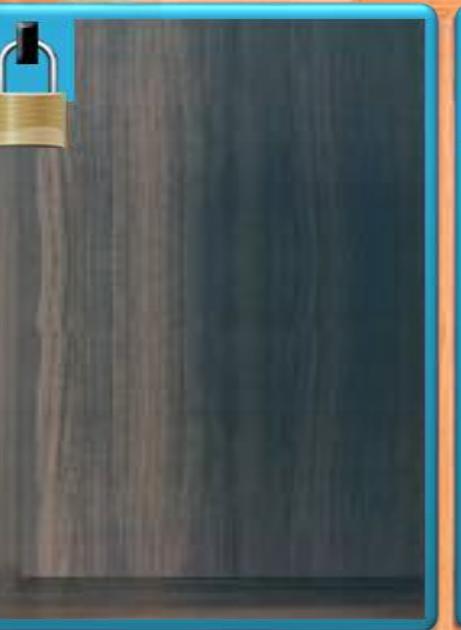
& 0xA1

& 0xA2

& 0xA3

& 0xA4

& 0xA5



Ponteiros

Xavier



Zorro



Paulo H enrique



& 0xA1



& 0xA2



& 0xA3



& 0xA4



& 0xA5



Ponteiros



Zorro



Paulo Henrique



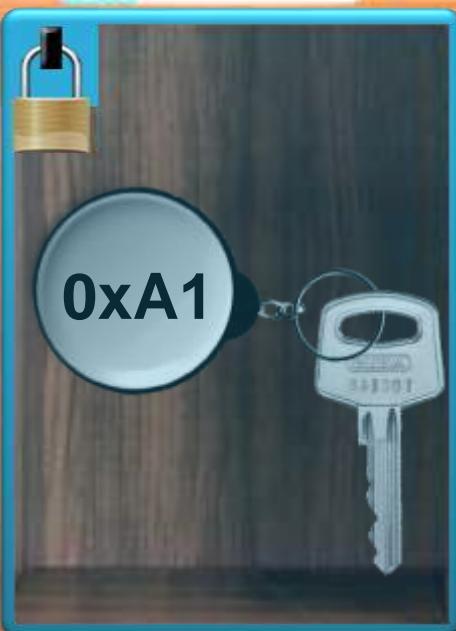
& 0xA1



& 0xA2



& 0xA3



& 0xA4



& 0xA5



Ponteiros

Xavier



Zorro



Paulo H enrique



& 0xA1



& 0xA2



& 0xA3



& 0xA4

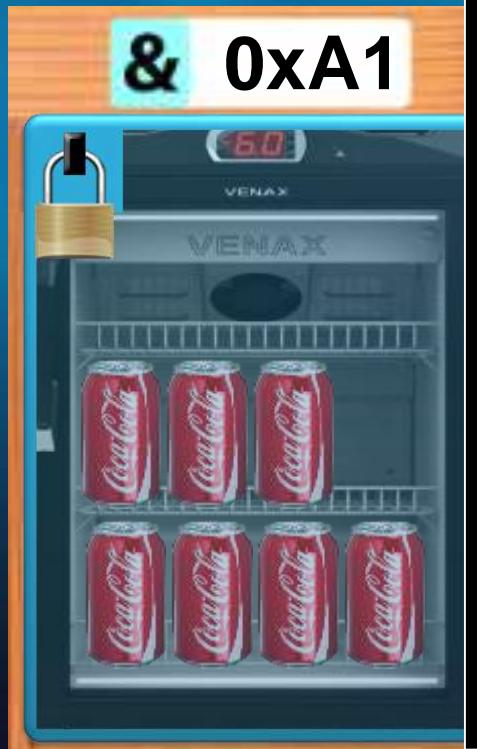


& 0xA5



Ponteiros

Xavier



PC

?

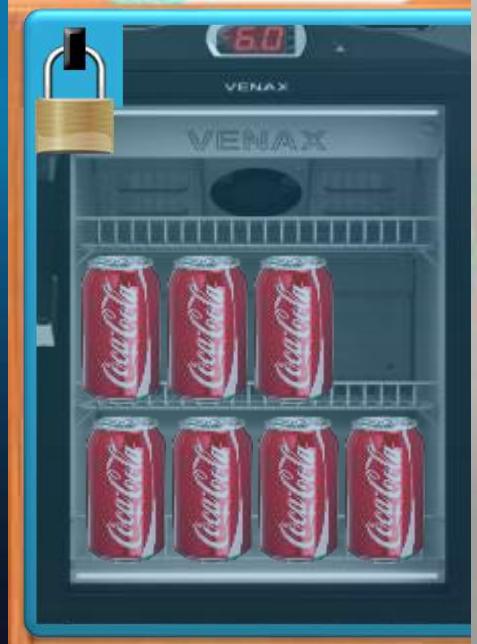


Ponteiros

Xavier



& 0xA1



Paula Cristina



& 0xA5



Ponteiros

Xavier



Zorro



Paulo H enrique



Paula C ristina



& 0xA1



& 0xA2



& 0xA3



& 0xA4



& 0xA5



Ponteiros



Ponteiros



Ponteiros



Ponteiros



PC = &PH;

Ponteiros

Xavier



Zorro



Paulo H enrique



Paula C ristina



& 0xA1



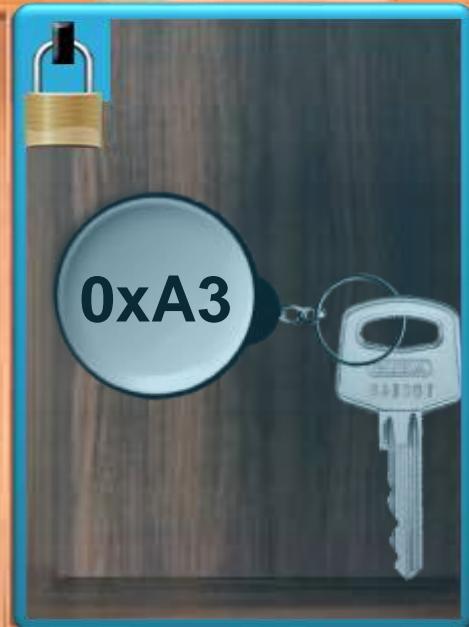
& 0xA2



& 0xA3



& 0xA4



& 0xA5



Ponteiros

Xavier



Zorro



Paulo H enrique



& 0xA1

& 0xA2

& 0xA3

& 0xA4

& 0xA5



Ponteiros

Xavier



Zorro



Paulo H enrique



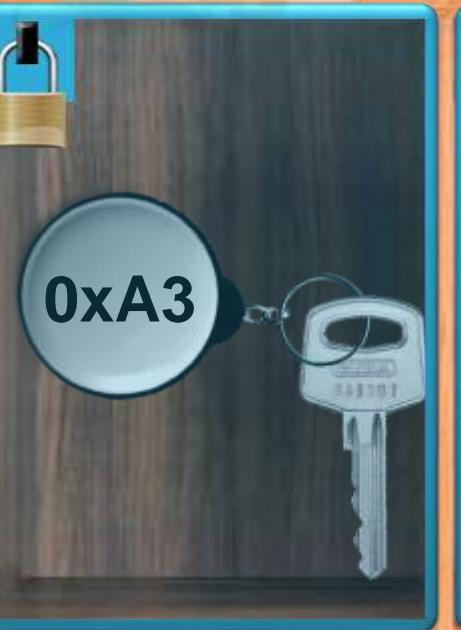
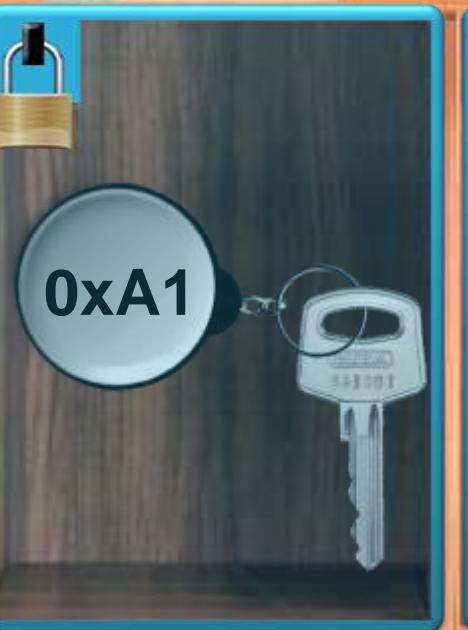
& 0xA1

& 0xA2

& 0xA3

& 0xA4

& 0xA5



Ponteiros

Xavier



Zorro



Paulo H enrique



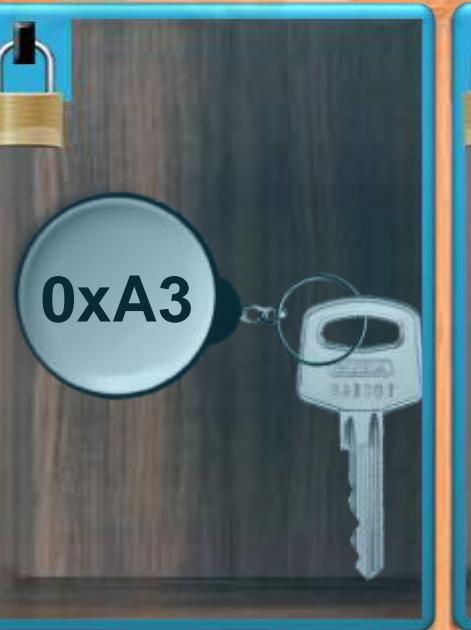
& 0xA1

& 0xA2

& 0xA3

& 0xA4

& 0xA5



Ponteiros

Xavier



Zorro



Paulo H enrique



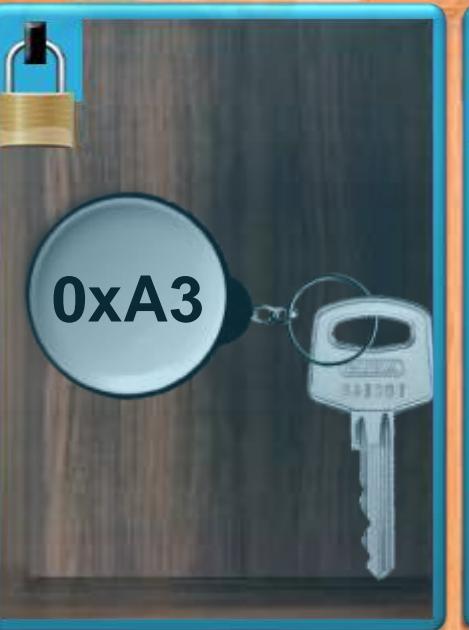
& 0xA1

& 0xA2

& 0xA3

& 0xA4

& 0xA5



Ponteiros

Xavier



Zorro



Paulo Henrique



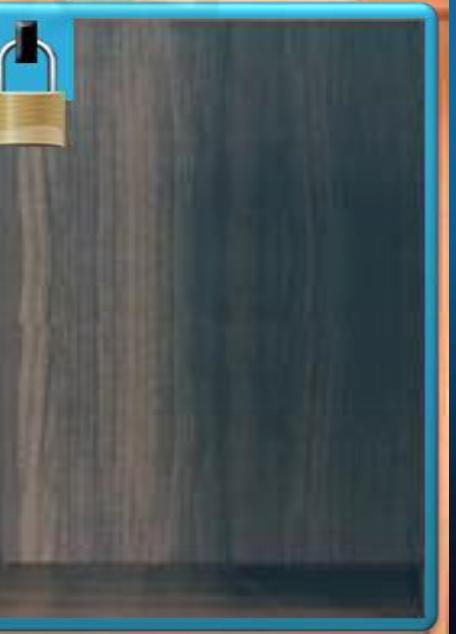
& 0xA1

& 0xA2

& 0xA3

& 0xA4

& 0xA5



Ponteiros

Xavier



Zorro



Paulo H enrique



Paula C ristina



& 0xA1



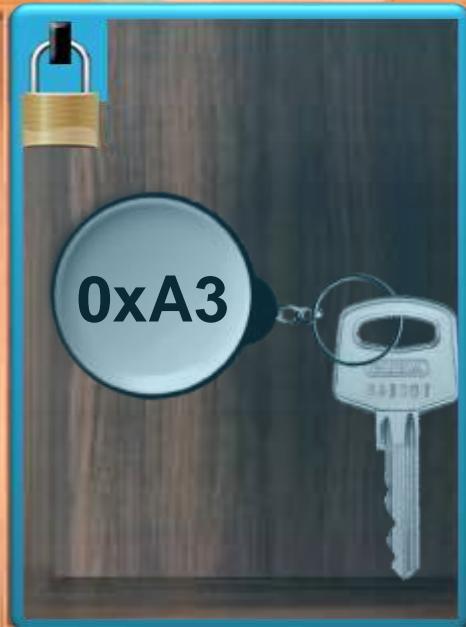
& 0xA2



& 0xA3



& 0xA4



& 0xA5



Ponteiros

Xavier



PP

?

0xA5

Ponteiros

Xavier



& 0xA1



Peter Parker



0xA5

Ponteiros

Xavier



Zorro



Paulo Henrique



Paula Cristina



Peter Parker



& 0xA1



& 0xA2



& 0xA3



& 0xA4



& 0xA5



Ponteiros



PP = &PC;

Ponteiros

Xavier



Zorro



Paulo H enrique



Paula C ristina



Peter P arker



& 0xA1



& 0xA2



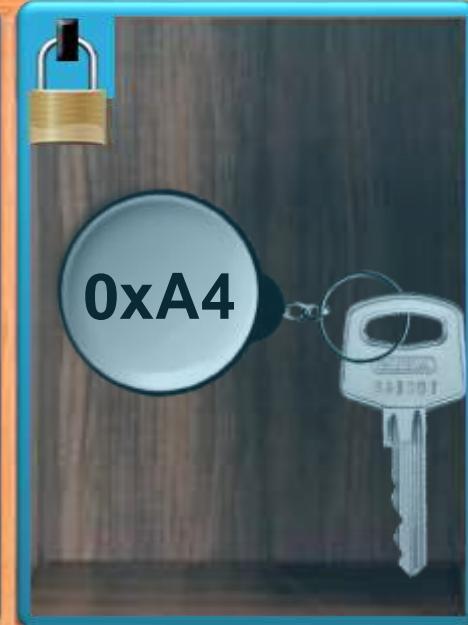
& 0xA3



& 0xA4



& 0xA5



Ponteiros

Xavier



Zorro



Paulo H enrique



Paula



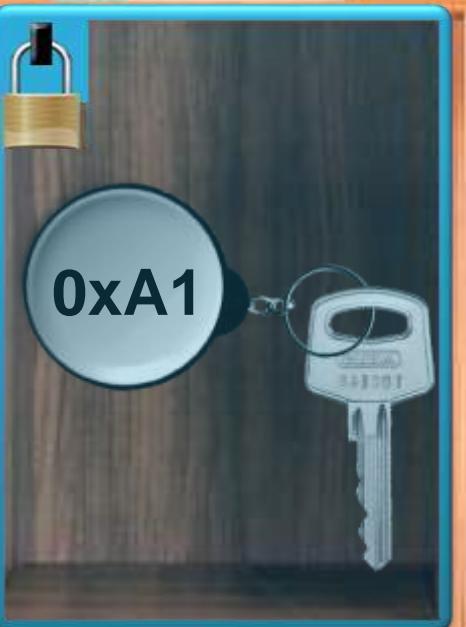
& 0xA1



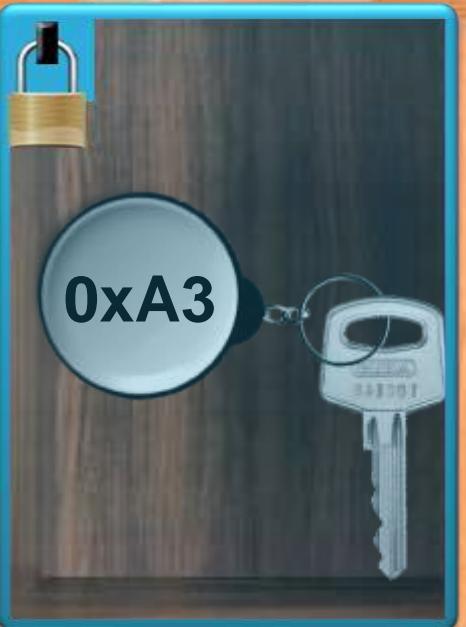
& 0xA2



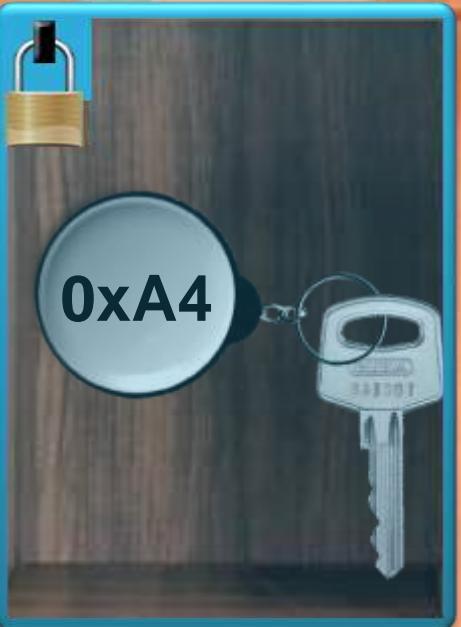
& 0xA3



& 0xA4



& 0xA5



Ponteiros

Xavier



Zorro



Paulo H enrique



Paula



& 0xA1



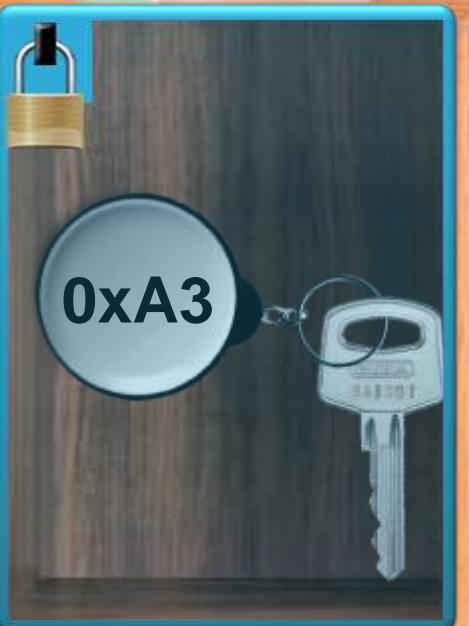
& 0xA2



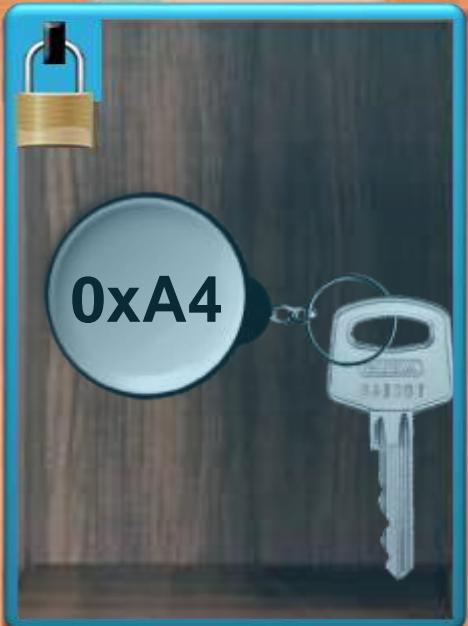
& 0xA3



& 0xA4



& 0xA5



Ponteiros

Xavier



Zorro



Paulo H enrique



Paula



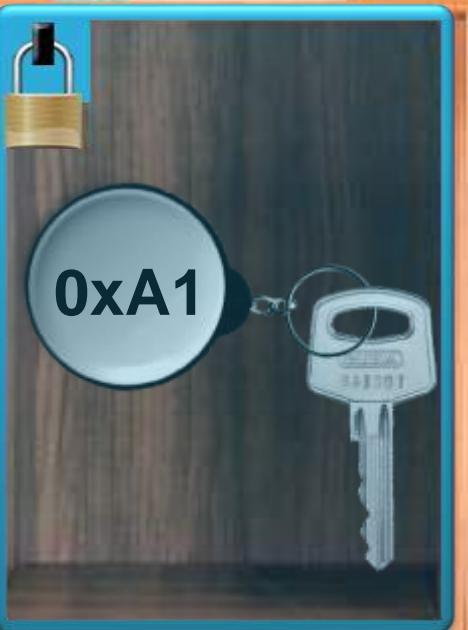
& 0xA1



& 0xA2



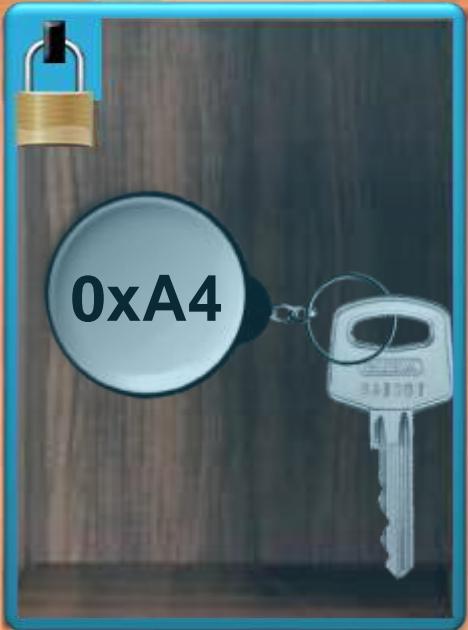
& 0xA3



& 0xA4



& 0xA5



Ponteiros

Xavier



Zorro



Paulo H enrique



Paula



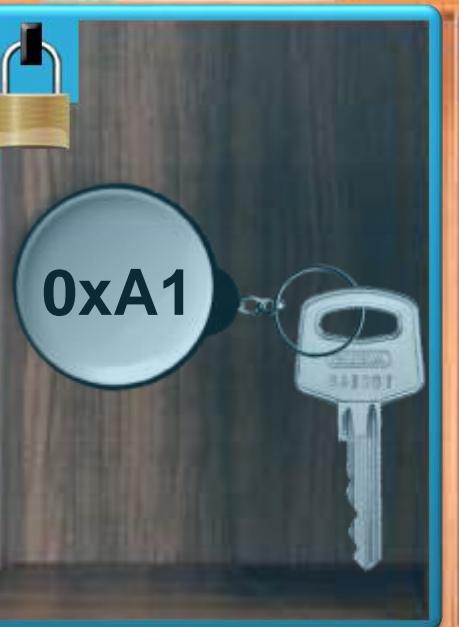
& 0xA1



& 0xA2



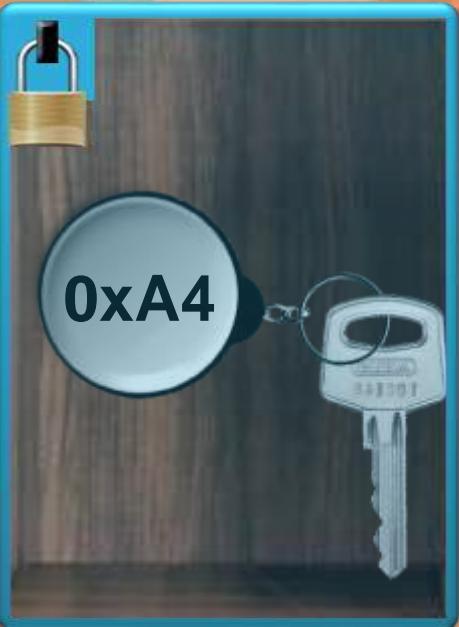
& 0xA3



& 0xA4



& 0xA5



Ponteiros

Xavier



Zorro



Paulo H enrique



Paula C ristina



Peter P arker



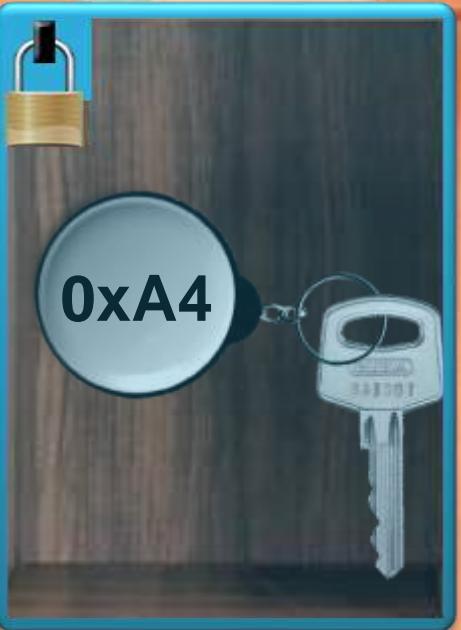
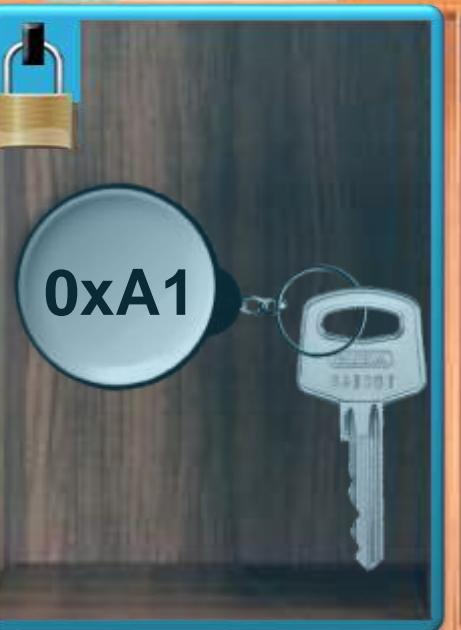
& 0xA1

& 0xA2

& 0xA3

& 0xA4

& 0xA5



Ponteiros



Passos

...

...

...

Ponteiros

Xavier



1º Passo
(OBRIGATÓRIO)
Declarar a variável

```
int x;
```

Ponteiros

Xavier



(opcional)
Atribuir valor a variável

```
int x;  
x = 8;
```

Ponteiros

Xavier



(opcional)
Exibir dados
da variável



```
int x;  
x = 8;  
cout << x;
```

Ponteiros

Xavier



(opcional)
Exibir dados
da variável

```
int x;  
x = 8;  
cout << &x;
```

MUITO
RARAMENTE

Ponteiros

Xavier



Passos

...

...

...



Ponteiros

Xavier



Paulo Henrique



& 0xA1



& 0xA2



& 0xA3



& 0xA4



& 0xA5



1º Passo

(OBRIGATÓRIO)

Declarar o ponteiro

```
int *ph;
```

Ponteiros

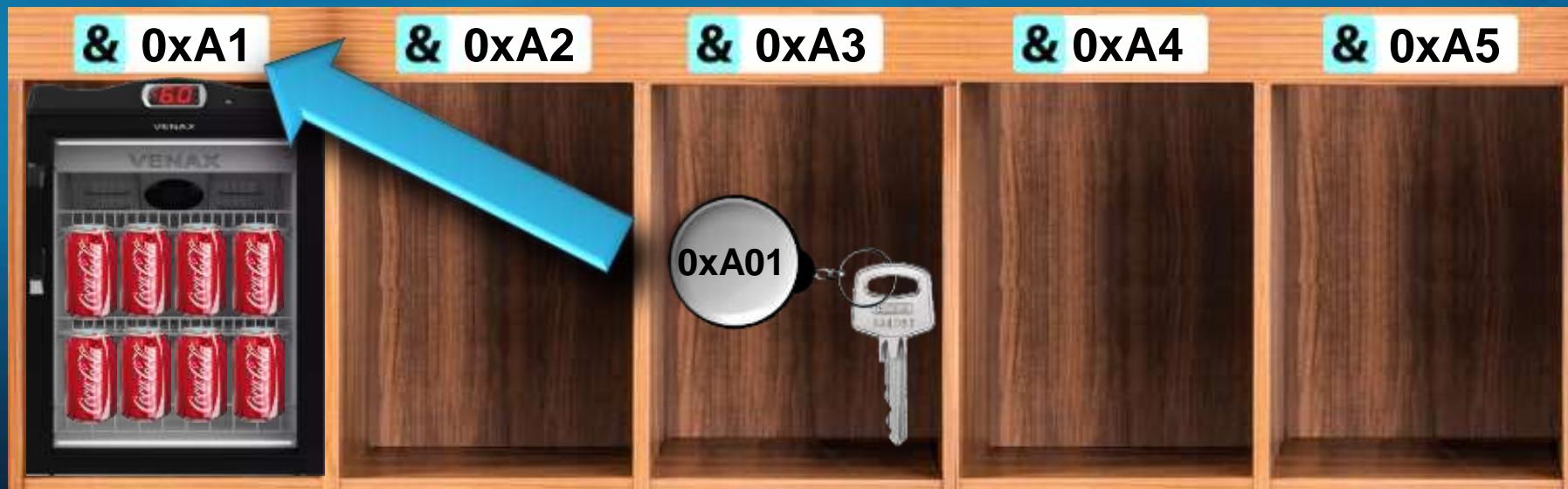
Xavier



Paulo Henrique



2º Passo
(OBRIGATÓRIO)
Apontar para
uma variável



```
int *ph;  
ph = &x;
```

Ponteiros

Xavier



Paulo Henrique



(opcional)
Exibir dados
do ponteiro



```
int *ph;  
ph = &x;  
cout << *ph;
```

Ponteiros

Xavier



Paulo Henrique



(opcional)
Atribuir algum
valor ao ponteiro



```
int *ph;  
ph = &x;  
*ph = 4;
```

Ponteiros

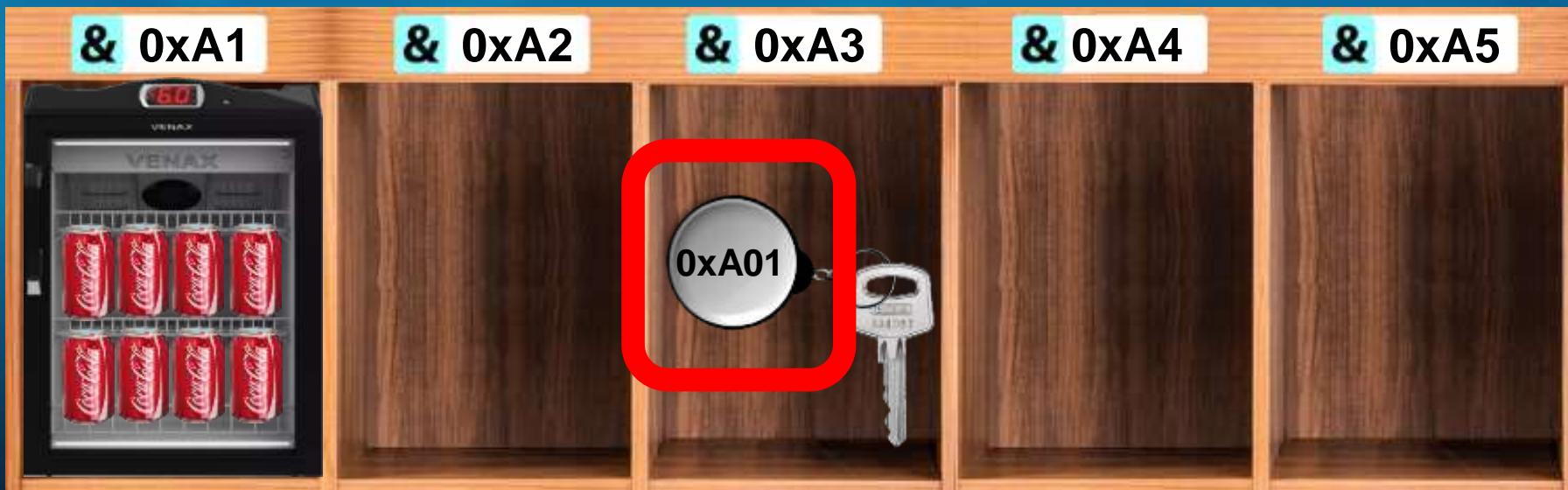
Xavier



Paulo Henrique



(opcional)
Exibir dados
do ponteiro



```
int *ph;  
ph = &x;  
cout << ph;
```

MUITO
RARAMENTE

Ponteiros

Xavier



Paulo Henrique



(opcional)
Exibir dados
do ponteiro



```
int *ph;  
ph = &x;  
cout << &ph;
```

MAIS RARAMENTE
AINDA

Ponteiros

Xavier



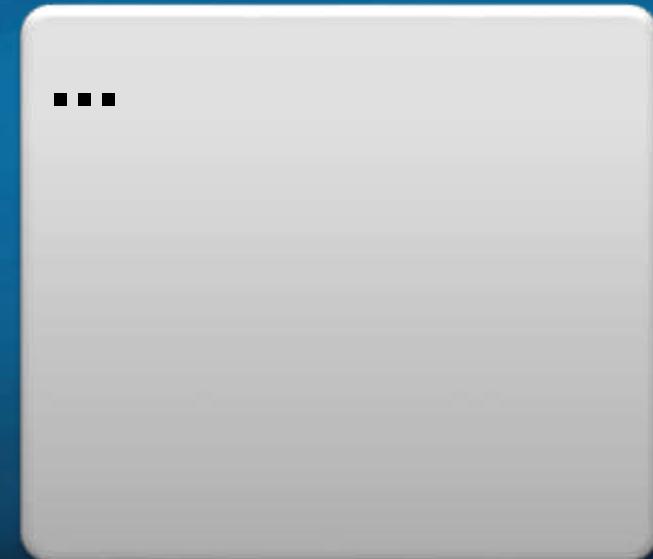
Paulo Henrique



Passos

...

...



Ponteiros

Xavier



Paulo Henrique



Paula Cristina



1º Passo

(OBRIGATÓRIO)

Declarar o ponteiro

& 0xA1



& 0xA2



& 0xA3



& 0xA4



& 0xA5



```
int **pc;
```

Ponteiros

Xavier



Paulo Henrique



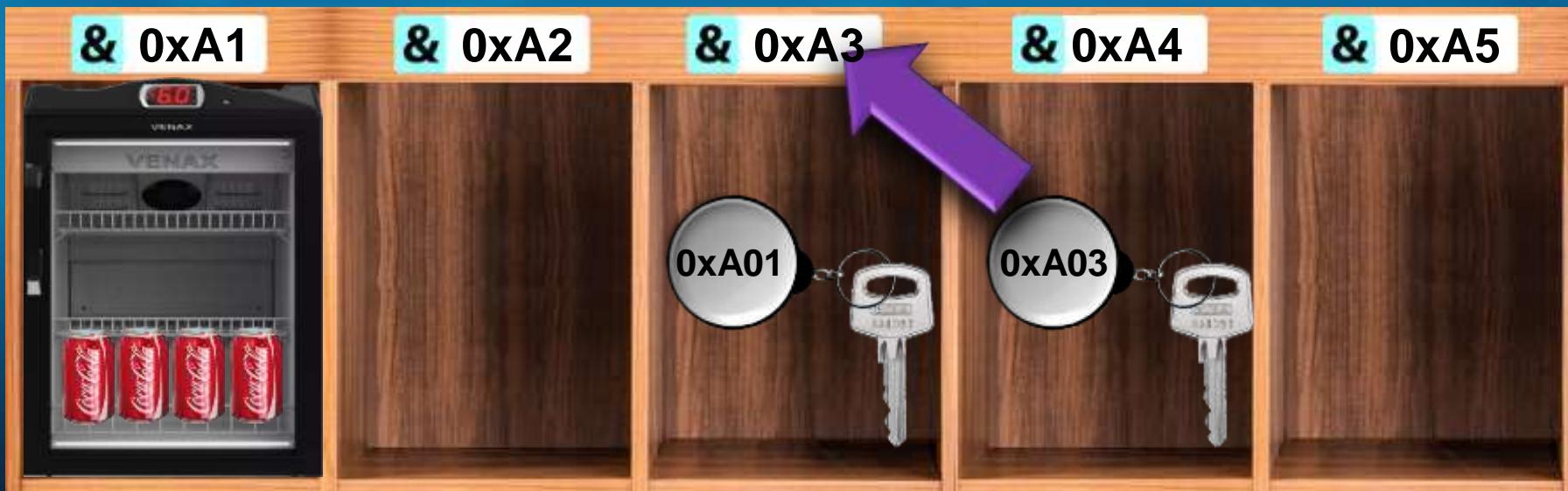
Paula Cristina



2º Passo

(OBRIGATÓRIO)

Apontar para
uma variável



```
int **pc;
```

```
pc = &ph;
```

Ponteiros

Xavier



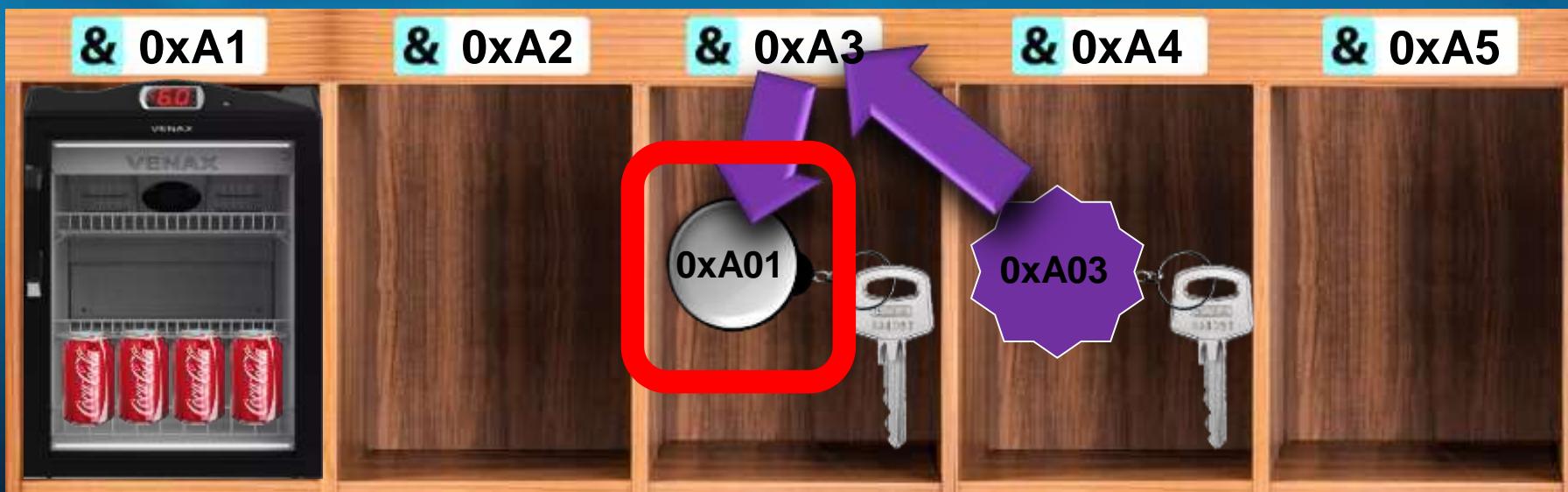
Paulo Henrique



Paula Cristina



(opcional)
Exibir dados
do ponteiro



```
int **pc;  
pc = &ph;  
cout << *pc;
```

MUITO
RARAMENTE

Ponteiros

Xavier



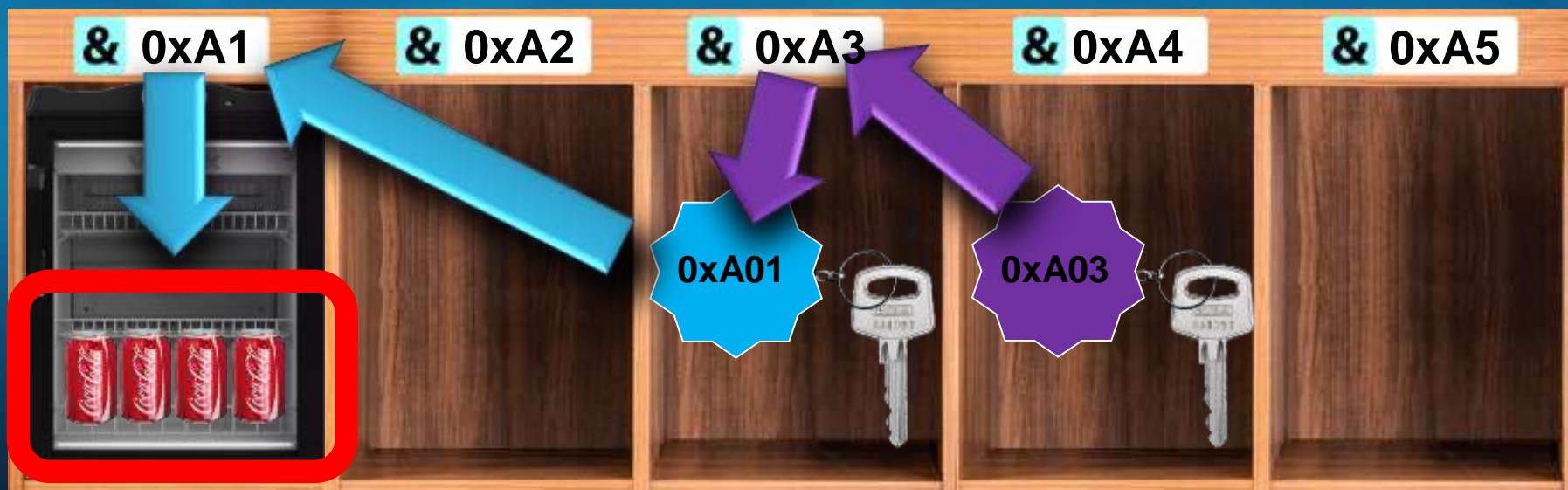
Paulo Henrique



Paula Cristina



(opcional)
Exibir dados
do ponteiro



```
int **pc;  
pc = &ph;  
cout << **pc;
```

MUITO
UTILIZADO

Ponteiros

Xavier



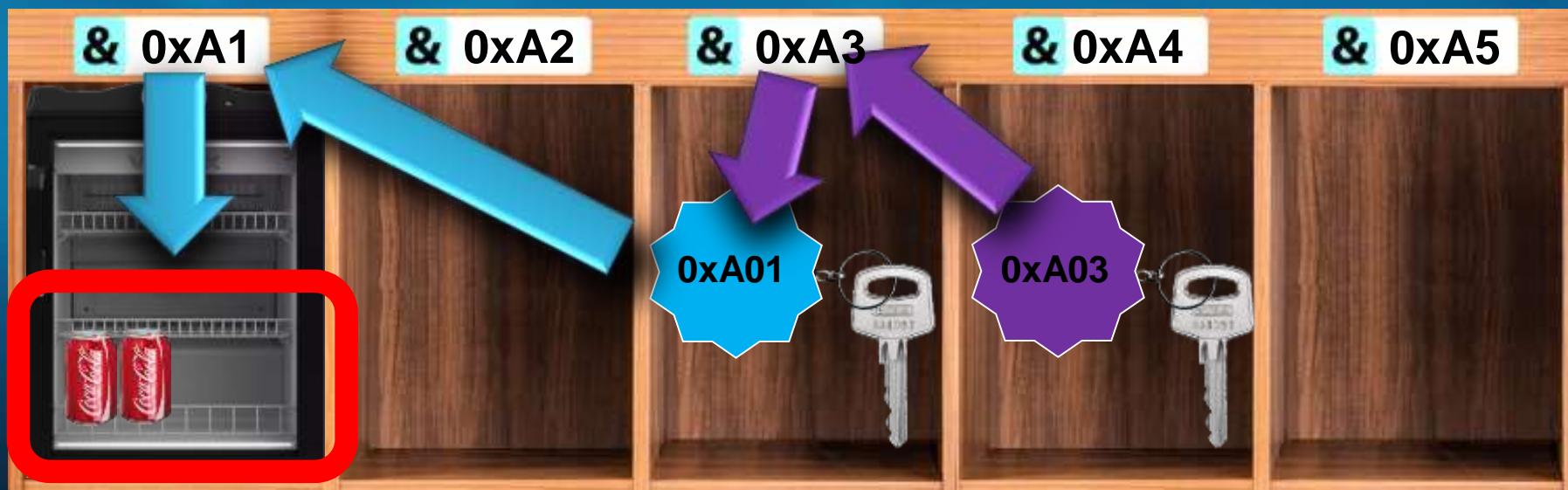
Paulo Henrique



Paula Cristina



(opcional)
Atribuir algum
valor ao ponteiro



```
int **pc;  
pc = &ph;  
**pc = 2;
```

MUITO
UTILIZADO

Ponteiros

Xavier



Paulo Henrique



Paula Cristina



(opcional)
Exibir dados
do ponteiro

& 0xA1



& 0xA2



& 0xA3



& 0xA4



& 0xA5



```
int **pc;  
pc = &ph;  
cout << pc;
```

MUITO
RARAMENTE

Ponteiros

Xavier



Paulo Henrique



Paula Cristina



(opcional)
Exibir dados
do ponteiro

& 0xA1



& 0xA2



& 0xA3



& 0xA4



& 0xA5



```
int **pc;  
pc = &ph;  
cout << &pc;
```

MAIS RARAMENTE
AINDA

Ponteiros

Xavier



Paulo Henrique



Paula Cristina



Passos

...

...

& 0xA1



& 0xA2



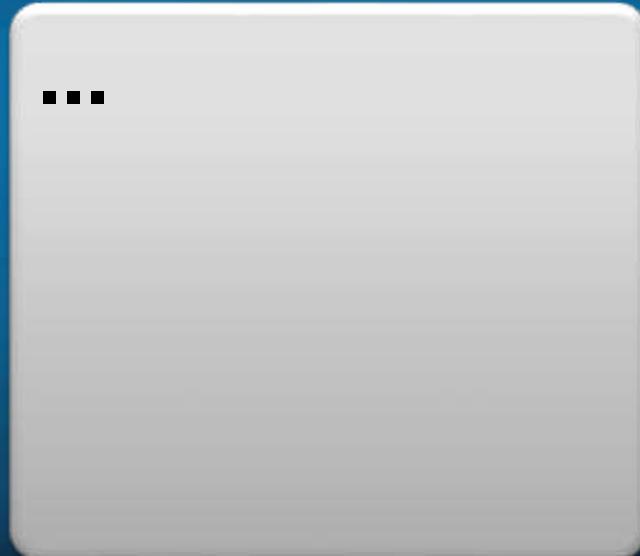
& 0xA3



& 0xA4



& 0xA5



Ponteiros

Xavier



Paulo Henrique



Paula Cristina



Peter Parker



& 0xA1



& 0xA2



& 0xA3



& 0xA4



& 0xA5



1º Passo

(OBRIGATÓRIO)
Declarar o ponteiro

```
int ***pp;
```

Ponteiros

Xavier



Paulo Henrique



Paula Cristina



Peter Parker



2º Passo
(OBRIGATÓRIO)
Apontar para
uma variável

& 0xA1



& 0xA2



& 0xA3



& 0xA4



& 0xA5



```
int ***pp;  
pp = &pc;
```

Ponteiros

Xavier



Paulo Henrique



Paula Cristina



Peter Parker



(opcional)
Exibir dados
do ponteiro

& 0xA1



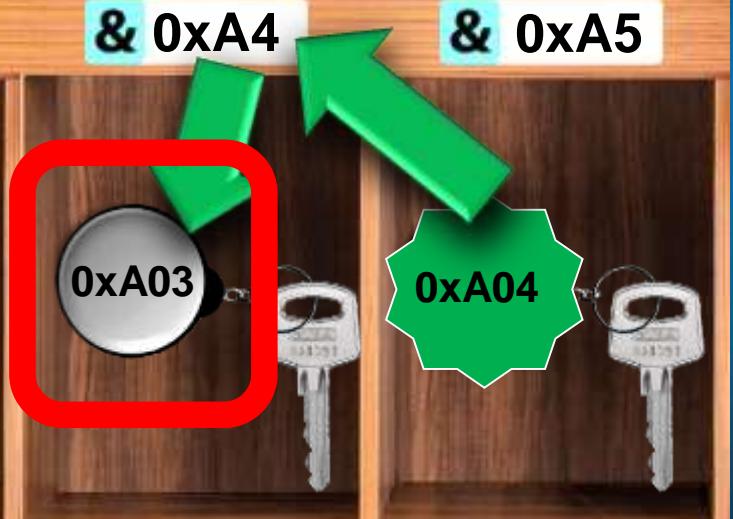
& 0xA2



& 0xA3



& 0xA4



& 0xA5

```
int ***pp;  
pp = &pc;  
cout << *pp;
```

MUITO
RARAMENTE

Ponteiros

Xavier



Paulo Henrique



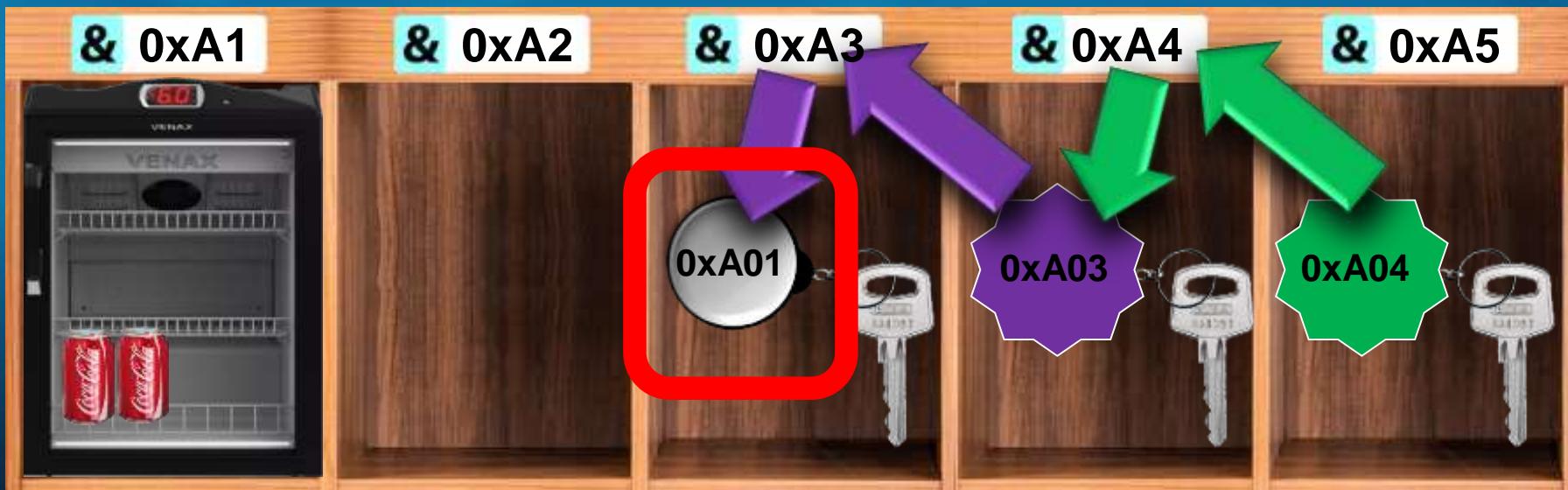
Paula Cristina



Peter Parker



(opcional)
Exibir dados
do ponteiro



```
int ***pp;  
pp = &pc;  
cout << **pp;
```

MUITO
RARAMENTE

Ponteiros

Xavier



Paulo Henrique



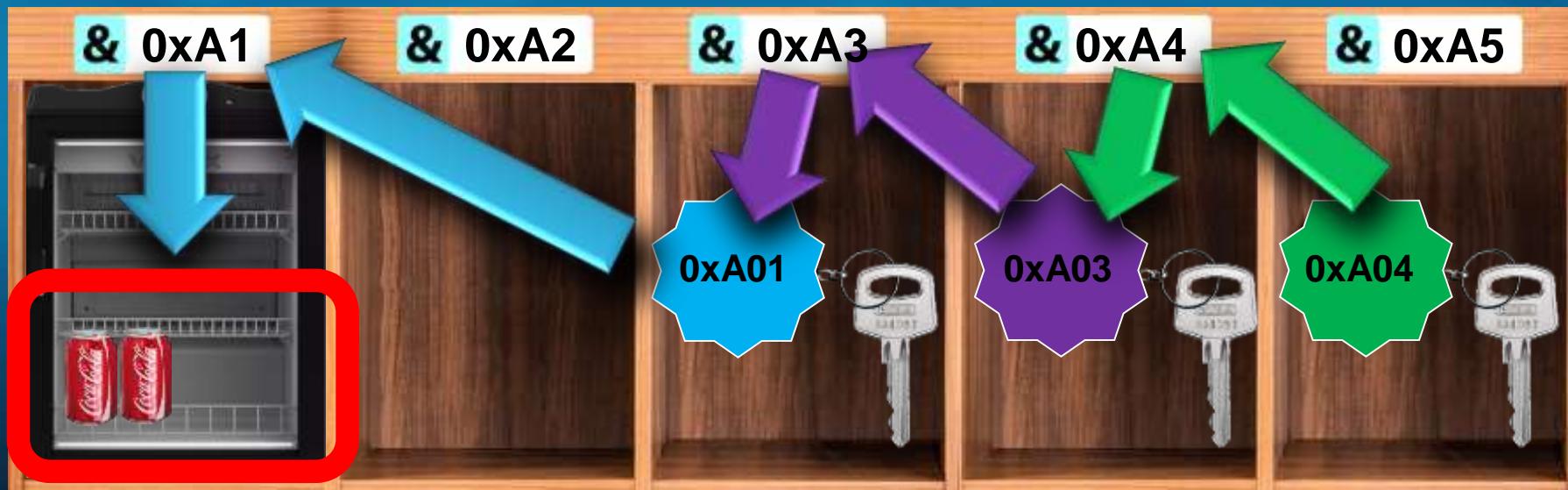
Paula Cristina



Peter Parker



(opcional)
Exibir dados
do ponteiro



```
int ***pp;  
pp = &pc;  
cout << ***pp;
```

MUITO
UTILIZADO

Ponteiros

Xavier



Paulo Henrique



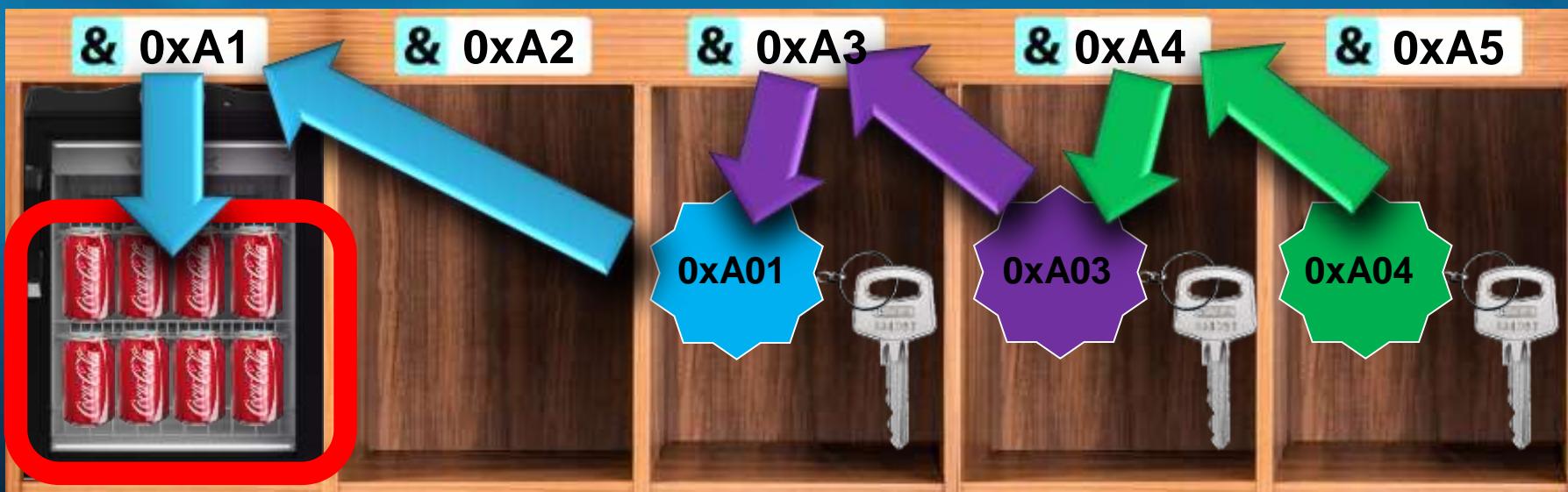
Paula Cristina



Peter Parker



(opcional)
Atribuir algum
valor ao ponteiro



```
int ***pp;  
pp = &pc;  
***pp = 8;
```

MUITO
UTILIZADO

Ponteiros

Xavier



Paulo Henrique



Paula Cristina



Peter Parker



(opcional)
Exibir dados
do ponteiro

& 0xA1



& 0xA2



& 0xA3



& 0xA4



& 0xA5



```
int ***pp;  
pp = &pc;  
cout << pp;
```

MUITO
RARAMENTE

Ponteiros

Xavier



Paulo Henrique



Paula Cristina



Peter Parker



(opcional)
Exibir dados
do ponteiro

& 0xA1



& 0xA2



& 0xA3



& 0xA4



& 0xA5



```
int ***pp;  
pp = &pc;  
cout << &pp;
```

MAIS RARAMENTE
AINDA

```
int main() {  
    int x, z;  
    int *pH, **pC, ***pP;  
  
    x = 8;  
    z = x;  
  
    pH = &x;  
    pC = &pH;  
    pP = &pC;  
  
    cout << "Endereço de &pH: " << &pH << endl;  
    cout << "Conteúdo de pH: " << pH << endl;  
    cout << "Valor de *pH: " << *pH << endl << endl;  
  
    cout << "Endereço de &pC: " << &pC << endl;  
    cout << "Conteúdo de pC: " << pC << endl;  
    cout << "Valor de *pC: " << *pC << endl;  
    cout << "Valor de **pC: " << **pC << endl << endl;  
  
    cout << "Endereço de &pP: " << &pP << endl;  
    cout << "Conteúdo de pP: " << pP << endl;  
    cout << "Valor de *pP: " << *pP << endl;  
    cout << "Valor de **pP: " << **pP << endl << endl;  
    cout << "Valor de ***pP: " << ***pP << endl << endl;  
}
```

O que será
exibido na tela?

```
int main() {
    int x, z;
    int *pH, **pC, ***pP;

    x = 8;
    z = x;

    pH = &x;
    pC = &pH;
    pP = &pC;

    cout << "Endereço de &pH: " << &pH << endl;
    cout << "Conteúdo de pH: " << pH << endl;
    cout << "Valor de *pH: " << *pH << endl << endl;

    cout << "Endereço de &pC: " << &pC << endl;
    cout << "Conteúdo de pC: " << pC << endl;
    cout << "Valor de *pC: " << *pC << endl;
    cout << "Valor de **pC: " << **pC << endl << endl;

    cout << "Endereço de &pP: " << &pP << endl;
    cout << "Conteúdo de pP: " << pP << endl;
    cout << "Valor de *pP: " << *pP << endl;
    cout << "Valor de **pP: " << **pP << endl;
    cout << "Valor de ***pP: " << ***pP << endl << endl;
}
```

&Endereço	Valor
0xA1	
0xA2	
0xA3	
X	
0xA4	
0xA5	
0xA6	
Z	
0xA7	
0xA8	
0xA9	
0xAA	

```

int main() {
    int x, z;
    int *pH, **pC, ***pP;

    x = 8;
    z = x;

    pH = &x;
    pC = &pH;
    pP = &pC;

    cout << "Endereço de &pH: " << &pH << endl;
    cout << "Conteúdo de pH: " << pH << endl;
    cout << "Valor de *pH: " << *pH << endl << endl;

    cout << "Endereço de &pC: " << &pC << endl;
    cout << "Conteúdo de pC: " << pC << endl;
    cout << "Valor de *pC: " << *pC << endl;
    cout << "Valor de **pC: " << **pC << endl << endl;

    cout << "Endereço de &pP: " << &pP << endl;
    cout << "Conteúdo de pP: " << pP << endl;
    cout << "Valor de *pP: " << *pP << endl;
    cout << "Valor de **pP: " << **pP << endl;
    cout << "Valor de ***pP: " << ***pP << endl << endl;
}

```

&ndereço	Valor
0xA1	
0xA2	
0xA3	
X	
0xA4	
0xA5	
Z	
0xA6	
0xA7	
pH	
0xA8	
pC	
0xA9	
pP	
0xAA	

```

int main() {
    int x, z;
    int *pH, **pC, ***pP;

    x = 8;
    z = x;

    pH = &x;
    pC = &pH;
    pP = &pC;

    cout << "Endereço de &pH: " << &pH << endl;
    cout << "Conteúdo de pH: " << pH << endl;
    cout << "Valor de *pH: " << *pH << endl << endl;

    cout << "Endereço de &pC: " << &pC << endl;
    cout << "Conteúdo de pC: " << pC << endl;
    cout << "Valor de *pC: " << *pC << endl;
    cout << "Valor de **pC: " << **pC << endl << endl;

    cout << "Endereço de &pP: " << &pP << endl;
    cout << "Conteúdo de pP: " << pP << endl;
    cout << "Valor de *pP: " << *pP << endl;
    cout << "Valor de **pP: " << **pP << endl;
    cout << "Valor de ***pP: " << ***pP << endl << endl;
}

```

&ndereço	Valor
0xA1	
X	0xA2 8
0xA3	
Z	0xA4
0xA5	
pH	0xA6
0xA7	
pC	0xA8
0xA9	
pP	0xAA

```

int main() {
    int x, z;
    int *pH, **pC, ***pP;

    x = 8;
    z = x;

    pH = &x;
    pC = &pH;
    pP = &pC;

    cout << "Endereço de &pH: " << &pH << endl;
    cout << "Conteúdo de pH: " << pH << endl;
    cout << "Valor de *pH: " << *pH << endl << endl;

    cout << "Endereço de &pC: " << &pC << endl;
    cout << "Conteúdo de pC: " << pC << endl;
    cout << "Valor de *pC: " << *pC << endl;
    cout << "Valor de **pC: " << **pC << endl << endl;

    cout << "Endereço de &pP: " << &pP << endl;
    cout << "Conteúdo de pP: " << pP << endl;
    cout << "Valor de *pP: " << *pP << endl;
    cout << "Valor de **pP: " << **pP << endl;
    cout << "Valor de ***pP: " << ***pP << endl << endl;
}

```

&ndereço	Valor
0xA1	
X	0xA2 8
0xA3	
Z	0xA4 8
0xA5	
pH	0xA6
0xA7	
pC	0xA8
0xA9	
pP	0xAA

```

int main() {
    int x, z;
    int *pH, **pC, ***pP;

    x = 8;
    z = x;

    pH = &x;
    pC = &pH;
    pP = &pC;

    cout << "Endereço de &pH: " << &pH << endl;
    cout << "Conteúdo de pH: " << pH << endl;
    cout << "Valor de *pH: " << *pH << endl << endl;

    cout << "Endereço de &pC: " << &pC << endl;
    cout << "Conteúdo de pC: " << pC << endl;
    cout << "Valor de *pC: " << *pC << endl;
    cout << "Valor de **pC: " << **pC << endl << endl;

    cout << "Endereço de &pP: " << &pP << endl;
    cout << "Conteúdo de pP: " << pP << endl;
    cout << "Valor de *pP: " << *pP << endl;
    cout << "Valor de **pP: " << **pP << endl;
    cout << "Valor de ***pP: " << ***pP << endl << endl;
}

```

&ndereço	Valor
0xA1	
X	0xA2
0xA3	
Z	0xA4
0xA5	
pH	0xA6
0xA7	
pC	0xA8
0xA9	
pP	0xAA

```

int main() {
    int x, z;
    int *pH, **pC, ***pP;

    x = 8;
    z = x;

    pH = &x;
    pC = &pH;
    pP = &pC;

    cout << "Endereço de &pH: " << &pH << endl;
    cout << "Conteúdo de pH: " << pH << endl;
    cout << "Valor de *pH: " << *pH << endl << endl;

    cout << "Endereço de &pC: " << &pC << endl;
    cout << "Conteúdo de pC: " << pC << endl;
    cout << "Valor de *pC: " << *pC << endl;
    cout << "Valor de **pC: " << **pC << endl << endl;

    cout << "Endereço de &pP: " << &pP << endl;
    cout << "Conteúdo de pP: " << pP << endl;
    cout << "Valor de *pP: " << *pP << endl;
    cout << "Valor de **pP: " << **pP << endl;
    cout << "Valor de ***pP: " << ***pP << endl << endl;
}

```

&ndereço	Valor
0xA1	
X	0xA2
0xA3	
Z	0xA4
0xA5	
pH	0xA6
0xA7	
pC	0xA8
0xA9	
pP	0xAA

```

int main() {
    int x, z;
    int *pH, **pC, ***pP;

    x = 8;
    z = x;

    pH = &x;
    pC = &pH;
    pP = &pC;

    cout << "Endereço de &pH: " << &pH << endl;
    cout << "Conteúdo de pH: " << pH << endl;
    cout << "Valor de *pH: " << *pH << endl << endl;

    cout << "Endereço de &pC: " << &pC << endl;
    cout << "Conteúdo de pC: " << pC << endl;
    cout << "Valor de *pC: " << *pC << endl;
    cout << "Valor de **pC: " << **pC << endl << endl;

    cout << "Endereço de &pP: " << &pP << endl;
    cout << "Conteúdo de pP: " << pP << endl;
    cout << "Valor de *pP: " << *pP << endl;
    cout << "Valor de **pP: " << **pP << endl;
    cout << "Valor de ***pP: " << ***pP << endl << endl;
}

```

&ndereço	Valor
0xA1	
X	0xA2 8
0xA3	
Z	0xA4 8
0xA5	
pH	0xA6 0xA2
0xA7	
pC	0xA8 0xA6
0xA9	
pP	0xAA 0xA8

```

int main() {
    int x, z;
    int *pH, **pC, ***pP;

    x = 8;
    z = x;

    pH = &x;
    pC = &pH;
    pP = &pC;

    cout << "Endereço de &pH: " << &pH << endl;
    cout << "Conteúdo de pH: " << pH << endl;
    cout << "Valor de *pH: " << *pH << endl << endl;

    cout << "Endereço de &pC: " << &pC << endl;
    cout << "Conteúdo de pC: " << pC << endl;
    cout << "Valor de *pC: " << *pC << endl;
    cout << "Valor de **pC: " << **pC << endl << endl;

    cout << "Endereço de &pP: " << &pP << endl;
    cout << "Conteúdo de pP: " << pP << endl;
    cout << "Valor de *pP: " << *pP << endl;
    cout << "Valor de **pP: " << **pP << endl;
    cout << "Valor de ***pP: " << ***pP << endl << endl;
}

```

&ndereço	Valor
0xA1	
X	0xA2 8 *
0xA3	
Z	0xA4 8
0xA5	
pH	0xA6 0xA2
0xA7	
pC	0xA8 0xA6
0xA9	
pP	0xAA 0xA8

```

int main() {
    int x, z;
    int *pH, **pC, ***pP;

    x = 8;
    z = x;

    pH = &x;
    pC = &pH;
    pP = &pC;

    cout << "Endereço de &pH: " << &pH << endl;
    cout << "Conteúdo de pH: " << pH << endl;
    cout << "Valor de *pH: " << *pH << endl << endl;

    cout << "Endereço de &pC: " << &pC << endl;
    cout << "Conteúdo de pC: " << pC << endl;
    cout << "Valor de *pC: " << *pC << endl;
    cout << "Valor de **pC: " << **pC << endl << endl;

    cout << "Endereço de &pP: " << &pP << endl;
    cout << "Conteúdo de pP: " << pP << endl;
    cout << "Valor de *pP: " << *pP << endl;
    cout << "Valor de **pP: " << **pP << endl;
    cout << "Valor de ***pP: " << ***pP << endl << endl;
}

```

&ndereço	Valor	
0xA1		
X	0xA2	8
0xA3		**
Z	0xA4	8
0xA5		
pH	0xA6	0xA2
0xA7		*
pC	0xA8	0xA6
0xA9		
pP	0xAA	0xA8

```

int main() {
    int x, z;
    int *pH, **pC, ***pP;

    x = 8;
    z = x;

    pH = &x;
    pC = &pH;
    pP = &pC;

    cout << "Endereço de &pH: " << &pH << endl;
    cout << "Conteúdo de pH: " << pH << endl;
    cout << "Valor de *pH: " << *pH << endl << endl;

    cout << "Endereço de &pC: " << &pC << endl;
    cout << "Conteúdo de pC: " << pC << endl;
    cout << "Valor de *pC: " << *pC << endl;
    cout << "Valor de **pC: " << **pC << endl << endl;

    cout << "Endereço de &pP: " << &pP << endl;
    cout << "Conteúdo de pP: " << pP << endl;
    cout << "Valor de *pP: " << *pP << endl;
    cout << "Valor de **pP: " << **pP << endl;
    cout << "Valor de ***pP: " << ***pP << endl << endl;
}

```

&ndereço	Valor
0xA1	
X	0xA2
0xA3	8
Z	0xA4
0xA5	8
pH	0xA6
0xA7	0xA2
pC	0xA8
0xA9	0xA6
pP	0xAA
0xAB	0xA8

Declaração Ponteiros

tipo_do_ponteiro *nome_do_ponteiro;



É o operador *asterisco* (*) que informa ao compilador que aquela variável não vai guardar um valor, mas sim um endereço de memória para aquele tipo especificado.

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 int main(){
4     //Declara um ponteiro para int
5     int *p;
6     //Declara um ponteiro para float
7     float *x;
8     //Declara um ponteiro para char
9     char *y;
10    //Declara uma variável do tipo int e um
11    //ponteiro para int
12    int soma, *p2,;
13    system(“pause”);
14    return 0;
15 }
```

Ponteiros



Apesar de usarem o mesmo símbolo, o operador ***** (**multiplicação**) não é o mesmo operador que o ***** (**referência de ponteiros**).

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 int main(){
4     int x = 3, y = 5, z;
5     z = y * x;
6     int *p;
7
8     system("pause");
9     return 0;
10 }
```

Ponteiros – Operador * e &

Ao se trabalhar com ponteiros, duas tarefas básicas serão sempre executadas:

- acessar o endereço de memória de uma variável;
- acessar o conteúdo de um endereço de memória;

Para realizar essas tarefas, iremos sempre utilizar apenas dois operadores: o operador “*” e o operador “&”.

Operador “*” versus operador “&”

“*”

Declara um ponteiro: **int *x;**

Conteúdo para onde o ponteiro aponta: **int y = *x;**

“&”

Endereço onde uma variável está guardada na memória:
&y

Exemplos – Operador * e &

```
int main() {
    setlocale(LC_ALL, "Portuguese");

    float y;
    float *pY;

    y = 500;

    pY = &y;

    *pY += 200;

    cout << "Valor de Y: " << y << endl;
    cout << "Valor de *pY: " << *pY << endl;

    system("pause");
    return 0;
}
```

Exemplos – Operador * e &

```
int main() {
    setlocale(LC_ALL, "Portuguese");

    float x, y, z, total;
    float *pX, *pY, *pZ;

    x = 5;
    y = 50;
    z = 500;

    pX = &x;
    pY = &y;
    pZ = &z;

    total = *pX + *pY + *pZ;

    cout << "Total: " << total << endl;

    system("pause");
    return 0;
}
```

&ndereço	Valor
0xA1	
0xA2	
0xA3	
0xA4	
0xA5	
0xA6	
0xA7	
0xA8	
0xA9	
0xAA	

Exemplos – Operador * e &

```
int main() {
    setlocale(LC_ALL, "Portuguese");

    float x, y, z, total;
    float *px, *py, *pz;

    x = 5;
    y = 50;
    z = 500;

    px = &x;
    py = &y;
    pz = &z;

    total = *px + *py + *pz;

    cout << "Total: " << total << endl;

    system("pause");
    return 0;
}
```

&ndereço	Valor
X	0xA1
y	0xA2
Z	0xA3
0xA4	
0xA5	
0xA6	
0xA7	
0xA8	
0xA9	
total	0xAA

Exemplos – Operador * e &

```
int main() {
    setlocale(LC_ALL, "Portuguese");

    float x, y, z, total;
    float *pX, *pY, *pZ;

    x = 5;
    y = 50;
    z = 500;

    pX = &x;
    pY = &y;
    pZ = &z;

    total = *pX + *pY + *pZ;

    cout << "Total: " << total << endl;

    system("pause");
    return 0;
}
```

&ndereço	Valor
X	0xA1
y	0xA2
Z	0xA3
	0xA4
pX	0xA5
	0xA6
pY	0xA7
	0xA8
pZ	0xA9
total	0xAA

Exemplos – Operador * e &

```
int main() {
    setlocale(LC_ALL, "Portuguese");

    float x, y, z, total;
    float *pX, *pY, *pZ;

    x = 5;
    y = 50;
    z = 500;

    pX = &x;
    pY = &y;
    pZ = &z;

    total = *pX + *pY + *pZ;

    cout << "Total: " << total << endl;

    system("pause");
    return 0;
}
```

&ndereço	Valor
X	0xA1
y	0xA2
Z	0xA3
	0xA4
pX	0xA5
	0xA6
pY	0xA7
	0xA8
pZ	0xA9
total	0xAA

Exemplos – Operador * e &

```
int main() {
    setlocale(LC_ALL, "Portuguese");

    float x, y, z, total;
    float *pX, *pY, *pZ;

    x = 5;
    y = 50;
    z = 500;

    pX = &x;
    pY = &y;
    pZ = &z;

    total = *pX + *pY + *pZ;

    cout << "Total: " << total << endl;

    system("pause");
    return 0;
}
```

&ndereço	Valor
x	0xA1
y	50
z	500
0xA4	
pX	0xA1
0xA6	
pY	0xA2
0xA8	
pZ	0xA3
total	0xAA

Exemplos – Operador * e &

```
int main() {
    setlocale(LC_ALL, "Portuguese");

    float x, y, z, total;
    float *pX, *pY, *pZ;

    x = 5;
    y = 50;
    z = 500;

    pX = &x;
    pY = &y; // Address of y
    pZ = &z;

    total = *pX + *pY + *pZ;

    cout << "Total: " << total << endl;

    system("pause");
    return 0;
}
```

&ndereço	Valor
X	0xA1
y	0xA2
Z	0xA3
	0xA4
pX	0xA5
	0xA6
pY	0xA7
	0xA8
pZ	0xA9
total	0xAA

Exemplos – Operador * e &

```
int main() {
    setlocale(LC_ALL, "Portuguese");

    float x, y, z, total;
    float *pX, *pY, *pZ;

    x = 5;
    y = 50;
    z = 500;

    pX = &x;
    pY = &y;
    pZ = &z; // Acesso ao endereço de z

    total = *pX + *pY + *pZ;

    cout << "Total: " << total << endl;

    system("pause");
    return 0;
}
```

&ndereço	Valor
X	0xA1
y	50
z	500
0xA4	
pX	0xA1
0xA6	
pY	0xA2
0xA8	
pZ	500
total	0xAA

Exemplos – Operador * e &

```
int main() {
    setlocale(LC_ALL, "Portuguese");

    float x, y, z, total;
    float *pX, *pY, *pZ;

    x = 5;
    y = 50;
    z = 500;

    pX = &x;
    pY = &y;
    pZ = &z;
    5
    total = *pX + *pY + *pZ;

    cout << "Total: " << total << endl;

    system("pause");
    return 0;
}
```

&ndereço	Valor
x	0xA1
y	50
z	500
0xA4	
pX	0xA1
0xA6	
pY	0xA2
0xA8	
pZ	0xA3
total	0xAA

Exemplos – Operador * e &

```
int main() {
    setlocale(LC_ALL, "Portuguese");

    float x, y, z, total;
    float *pX, *pY, *pZ;

    x = 5;
    y = 50;
    z = 500;

    pX = &x;
    pY = &y;
    pZ = &z;      5      50
    total = *pX + *pY + *pZ;

    cout << "Total: " << total << endl;

    system("pause");
    return 0;
}
```

&ndereço	Valor
X	0xA1
y	0xA2
Z	0xA3
	0xA4
pX	0xA5
	0xA6
pY	0xA7
	0xA8
pZ	0xA9
total	0xAA

*

Exemplos – Operador * e &

```
int main() {
    setlocale(LC_ALL, "Portuguese");

    float x, y, z, total;
    float *pX, *pY, *pZ;

    x = 5;
    y = 50;
    z = 500;

    pX = &x;
    pY = &y;
    pZ = &z;
    5   50  500
    total = *pX + *pY + *pZ;
    cout << "Total: " << total << endl;
    system("pause");
    return 0;
}
```

&ndereço	Valor
X	0xA1
y	0xA2
Z	0xA3 500
	0xA4
pX	0xA5
	0xA6
pY	0xA7
	0xA8
pZ	0xA9
total	0xAA

Exemplos – Operador * e &

```
int main() {
    setlocale(LC_ALL, "Portuguese");

    float x, y, z, total;
    float *pX, *pY, *pZ;

    x = 5;
    y = 50;
    z = 500;

    pX = &x;
    pY = &y;
    pZ = &z;
    5 50 500
    total = *pX + *pY + *pZ;

    cout << "Total: " << total << endl;

    system("pause");
    return 0;
}
```

&ndereço	Valor
X	0xA1
y	0xA2
Z	0xA3
	0xA4
pX	0xA5
	0xA6
pY	0xA7
	0xA8
pZ	0xA9
total	0xAA
	555

Exemplos – Operador * e &

```
int main() {
    setlocale(LC_ALL, "Portuguese");

    float x, y, z, total;
    float *pX, *pY, *pZ;

    x = 5;
    y = 50;
    z = 500;

    pX = &x;
    pY = &y;
    pZ = &z;

    total = *pX + *pY + *pZ;

    cout << "Total: " << total << endl;

    system("pause");
    return 0;
}
```

&ndereço	Valor
X	0xA1
y	50
Z	500
0xA4	
pX	0xA1
0xA6	
pY	0xA2
0xA8	
pZ	0xA3
total	555

Exercício

```
60 #include <iostream>
61 #include <locale.h>
62 #include <stdlib.h>
63
64 using namespace std;
65
66 int main() {
67     setlocale(LC_ALL, "Portuguese");
68
69     int i, j, *p;
70
71     i = 99;
72     p = &i;
73     j = *p + 100;
74
75     cout << j << endl;
76
77     return 0;
78 }
```

Qual a saída?

Exercício

```
82 #include <iostream>
83 #include <locale.h>
84 #include <stdlib.h>
85
86 using namespace std;
87
88 int main() {
89     setlocale(LC_ALL, "Portuguese");
90
91     int a, b, c, *p, *q;
92
93     a = 5;
94     b = 12;
95     p = &a;
96     q = &b;
97     c = *p + *q;
98
99     cout << c << endl;
100
101    return 0;
102 }
```

Qual a saída?

Ponteiros

```
int main() {
    setlocale(LC_ALL, "Portuguese");

    float x;
    float *pA, *pB;

    x = 300;

    pA = &x;
    pB = pA;

    cout << "Endereço de X: " << &x << endl;
    cout << "Conteúdo de pA: " << pA << endl;
    cout << "Conteúdo de pB: " << pB << endl;
    cout << "Valor de *pB: " << *pB << endl;

    system("pause");
    return 0;
}
```



&ndereço	Valor
0xA1	
0xA2	
0xA3	
0xA4	
0xA5	
0xA6	
0xA7	
0xA8	
0xA9	
0xAA	

Ponteiros

```
int main() {
    setlocale(LC_ALL, "Portuguese");

    float x;
    float *pA, *pB;

    x = 300;

    pA = &x;
    pB = pA;

    cout << "Endereço de X: " << &x << endl;
    cout << "Conteúdo de pA: " << pA << endl;
    cout << "Conteúdo de pB: " << pB << endl;
    cout << "Valor de *pB: " << *pB << endl;

    system("pause");
    return 0;
}
```

&ndereço	Valor
X	0xA1
0xA2	
0xA3	
0xA4	
pA	0xA5
0xA6	
pB	0xA7
0xA8	
0xA9	
0xAA	

Ponteiros

```
int main() {
    setlocale(LC_ALL, "Portuguese");

    float x;
    float *pA, *pB;

    x = 300; // Atribuição de valor para x

    pA = &x;
    pB = pA;

    cout << "Endereço de X: " << &x << endl;
    cout << "Conteúdo de pA: " << pA << endl;
    cout << "Conteúdo de pB: " << pB << endl;
    cout << "Valor de *pB: " << *pB << endl;

    system("pause");
    return 0;
}
```

&ndereço	Valor
X	0xA1 300
0xA2	
0xA3	
0xA4	
pA	0xA5
0xA6	
pB	0xA7
0xA8	
0xA9	
0xAA	

Ponteiros

```
int main() {
    setlocale(LC_ALL, "Portuguese");

    float x;
    float *pA, *pB;

    x = 300;

    pA = &x;
    pB = pA;

    cout << "Endereço de X: " << &x << endl;
    cout << "Conteúdo de pA: " << pA << endl;
    cout << "Conteúdo de pB: " << pB << endl;
    cout << "Valor de *pB: " << *pB << endl;

    system("pause");
    return 0;
}
```

&ndereço	Valor
X	0xA1
0xA2	
0xA3	
0xA4	
pA	0xA1
0xA6	
pB	0xA7
0xA8	
0xA9	
0xAA	

Ponteiros

```
int main() {
    setlocale(LC_ALL, "Portuguese");

    float x;
    float *pA, *pB;

    x = 300;

    pA = &x;
    pB = pA; // Atribuição de pB para pA

    cout << "Endereço de X: " << &x << endl;
    cout << "Conteúdo de pA: " << pA << endl;
    cout << "Conteúdo de pB: " << pB << endl;
    cout << "Valor de *pB: " << *pB << endl;

    system("pause");
    return 0;
}
```

&ndereço	Valor
X	0xA1 // Endereço de x
0xA2	
0xA3	
0xA4	
pA	0xA5 // Endereço de pA
0xA6	
pB	0xA7 // Endereço de pB
0xA8	
0xA9	
0xAA	

Ponteiros

```
int main() {
    setlocale(LC_ALL, "Portuguese");

    float x;
    float *pA, *pB;

    x = 300;

    pA = &x;
    pB = pA;

    cout << "Endereço de X: " << &x << endl;
    cout << "Conteúdo de pA: " << pA << endl;
    cout << "Conteúdo de pB: " << pB << endl;
    cout << "Valor de *pB: " << *pB << endl;

    system("pause");
    return 0;
}
```

&ndereço	Valor
X	0xA1
0xA2	
0xA3	
0xA4	
pA	0xA5
0xA6	
pB	0xA7
0xA8	
0xA9	0xA1
0xAA	

Funções Passagem de parâmetros por valor

Ponteiros

Ponteiros e funções

Passagem por **VALOR**:

```
void somar(int x);

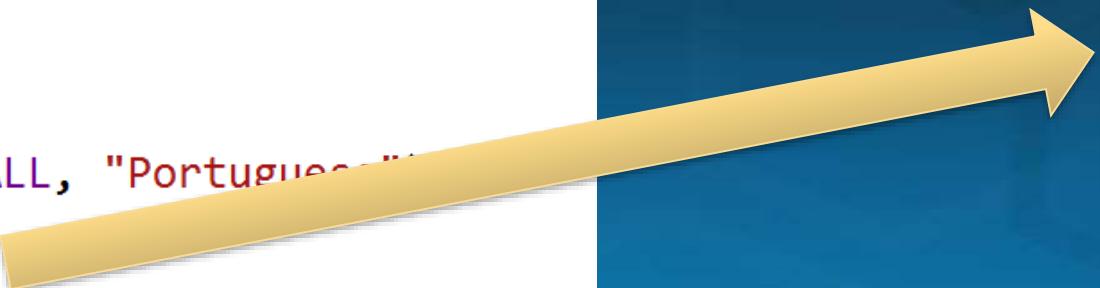
int main() {
    setlocale(LC_ALL, "Portuguese");
    int x = 100;

    somar(x);

    cout << "Valor de X: " << x << endl;

    system("pause");
    return 0;
}

void somar(int x) {
    x = x + 400;
}
```



&ndereço	Valor
0xA1	100
0xA2	
0xA3	
0xA4	
0xA5	
0xA6	
0xA7	
0xA8	
0xA9	
0xAA	

Ponteiros e funções

Passagem por **VALOR**:

```
void somar(int x);

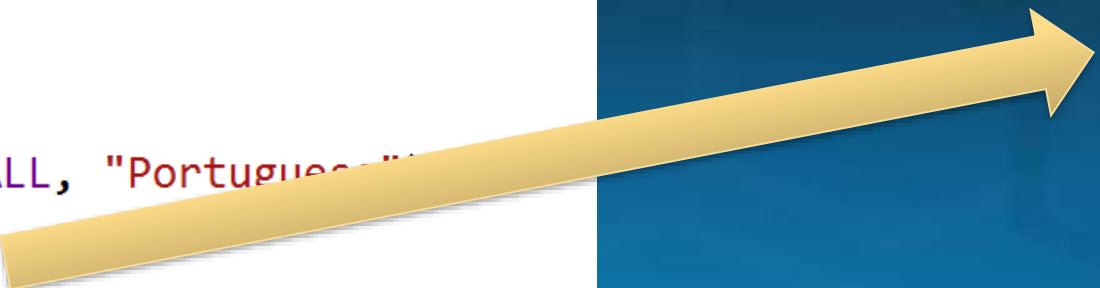
int main() {
    setlocale(LC_ALL, "Portuguese");
    int x = 100;

    somar(x);

    cout << "Valor de X: " << x << endl;

    system("pause");
    return 0;
}

void somar(int x) {
    x = x + 400;
}
```



&ndereço	Valor
0xA1	100
0xA2	
0xA3	
0xA4	
0xA5	
0xA6	
0xA7	
0xA8	
0xA9	
0xAA	

Ponteiros e funções

Passagem por **VALOR**:

```
void somar(int x);

int main() {
    setlocale(LC_ALL, "Portuguese");

    int x = 100;

    somar(x);

    cout << "Valor de X: " << x << endl;

    system("pause");
    return 0;
}

void somar(int x) {
    x = x + 400;
}
```

&ndereço	Valor
0xA1	100
0xA2	
0xA3	
0xA4	
0xA5	
0xA6	
0xA7	
0xA8	
0xA9	
0xAA	

Ponteiros e funções

Passagem por **VALOR**:

```
void somar(int x);

int main() {
    setlocale(LC_ALL, "Portuguese");

    int x = 100;

    somar(x);

    cout << "Valor de X: " << x << endl;

    system("pause");
    return 0;
}

void somar(int x) {
    x = x + 400;
}
```

&ndereço	Valor
0xA1	100
0xA2	
0xA3	
0xA4	
0xA5	100
0xA6	
0xA7	
0xA8	
0xA9	
0xAA	

Ponteiros e funções

Passagem por **VALOR**:

```
void somar(int x);

int main() {
    setlocale(LC_ALL, "Portuguese");

    int x = 100;

    somar(x);

    cout << "Valor de X: " << x << endl;

    system("pause");
    return 0;
}

void somar(int x) {
    x = x + 400;
}
```

&ndereço	Valor
0xA1	100
0xA2	
0xA3	
0xA4	
0xA5	500
0xA6	
0xA7	
0xA8	
0xA9	
0xAA	

Ponteiros e funções

Passagem por **VALOR**:

```
void somar(int x);

int main() {
    setlocale(LC_ALL, "Portuguese");

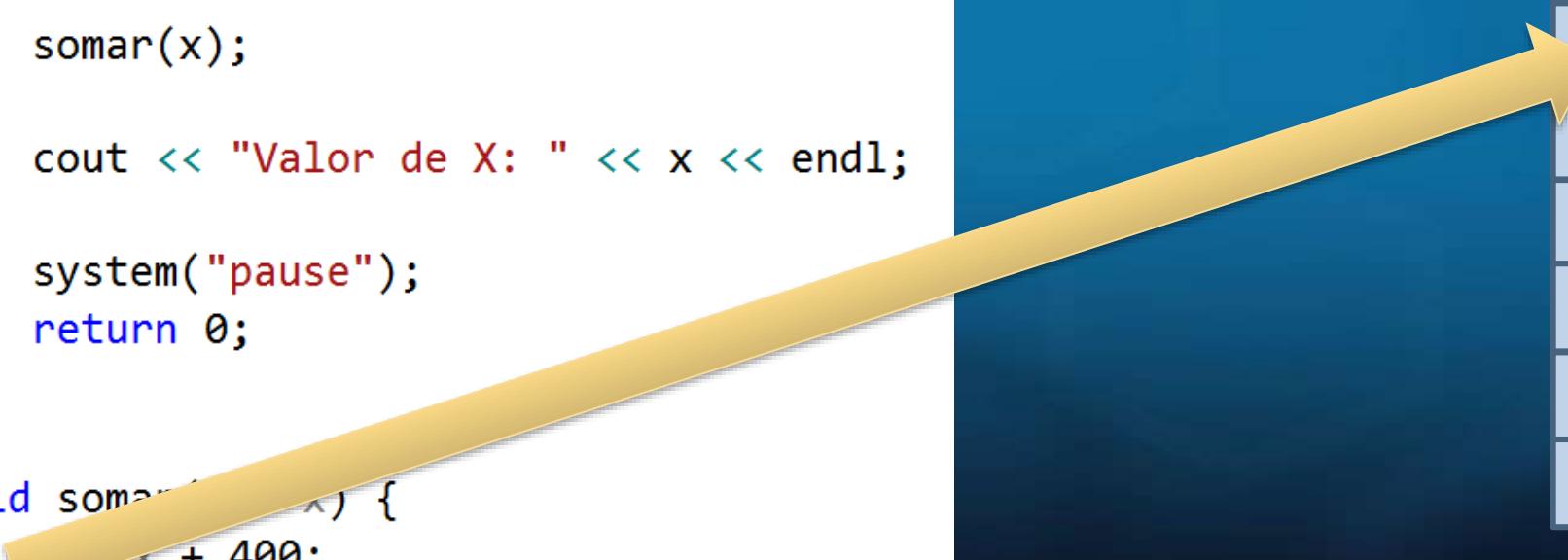
    int x = 100;

    somar(x);

    cout << "Valor de X: " << x << endl;

    system("pause");
    return 0;
}

void somar(int x) {
    x + 400;
}
```



A yellow arrow originates from the variable 'x' in the main function's code and points towards a memory dump table. The table has two columns: '&ndereço' (Address) and 'Valor' (Value). The address column lists memory locations from 0xA1 to 0xAA. The value column shows the initial value of x (100) at address 0xA1, and subsequent values (140, 180, 220, 260, 300, 340, 380, 420) at addresses 0xA2 through 0xAA respectively, indicating that each call to somar() creates a new copy of the variable x.

&ndereço	Valor
0xA1	100
0xA2	140
0xA3	180
0xA4	220
0xA5	260
0xA6	300
0xA7	340
0xA8	380
0xA9	420
0xAA	

Ponteiros e funções

Passagem por **VALOR**:

```
void somar(int x);

int main() {
    setlocale(LC_ALL, "Portuguese");

    int x = 100;

    somar(x);

    cout << "Valor de X: " << x << endl;

    system("pause");
    return 0;
}

void somar(int x) {
    x = x + 400;
}
```



&ndereço	Valor
0xA1	100
0xA2	
0xA3	
0xA4	
0xA5	
0xA6	
0xA7	
0xA8	
0xA9	
0xAA	

Funções Passagem de parâmetros por referência

Ponteiros

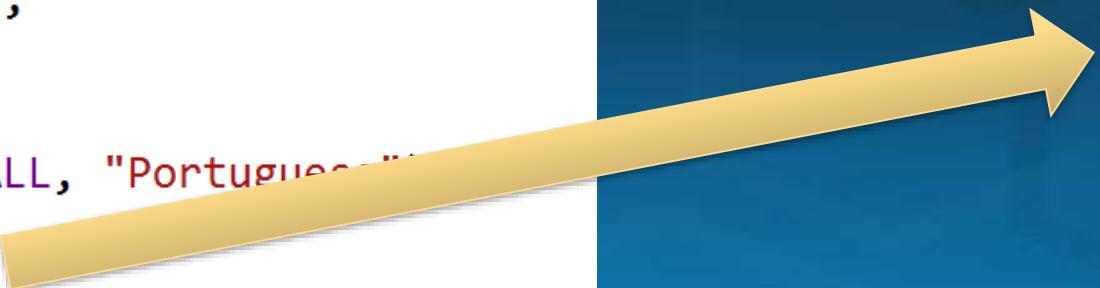
Ponteiros e funções

Passagem por REFERÊNCIA:

```
void somar(int *x);

int main() {
    setlocale(LC_ALL, "Portuguese");
    int x = 100; int x = 100;
    somar(&x);
    cout << "Valor de X: " << x << endl;
    system("pause");
    return 0;
}

void somar(int *x) {
    *x = *x + 400;
}
```



&ndereço	Valor
0xA1	100
0xA2	
0xA3	
0xA4	
0xA5	
0xA6	
0xA7	
0xA8	
0xA9	
0xAA	

Ponteiros e funções

Passagem por REFERÊNCIA:

```
void somar(int *x);

int main() {
    setlocale(LC_ALL, "Portuguese");

    int x = 100;

    somar(&x);

    cout << "Valor de X: " << x << endl;

    system("pause");
    return 0;
}

void somar(int *x) {
    *x = *x + 400;
}
```

&ndereço	Valor
0xA1	100
0xA2	
0xA3	
0xA4	
0xA5	
0xA6	
0xA7	
0xA8	
0xA9	
0xAA	

Ponteiros e funções

Passagem por REFERÊNCIA:

```
void somar(int *x);

int main() {
    setlocale(LC_ALL, "Portuguese");

    int x = 100;

    somar(&x);

    cout << "Valor de X: " << x << endl;

    system("pause");
    return 0;
}

void somar int *x {
    *x = *x + 400;
}
```

&ndereço	Valor
X	0xA1
0xA2	
0xA3	
0xA4	
X	0xA1
0xA5	
0xA6	
0xA7	
0xA8	
0xA9	
0xAA	

Ponteiros e funções

Passagem por REFERÊNCIA:

```
void somar(int *x);

int main() {
    setlocale(LC_ALL, "Portuguese");

    int x = 100;

    somar(&x);

    cout << "Valor de X: " << x << endl;

    system("pause");
    return 0;
}

void somar(int *x) {
    *x = *x + 400;
}
```

&ndereço	Valor
0xA1	100
0xA2	
0xA3	
0xA4	
0xA5	0xA1
0xA6	
0xA7	
0xA8	
0xA9	
0xAA	

Ponteiros e funções

Passagem por REFERÊNCIA:

```
void somar(int *x);

int main() {
    setlocale(LC_ALL, "Portuguese");

    int x = 100;

    somar(&x);

    cout << "Valor de X: " << x << endl;

    system("pause");
    return 0;
}

void somar(int *x) {
    *x = *x + 400
}
```

A diagram illustrating pointer semantics. On the left, a variable `x` is shown with a blue circle containing a white 'X'. A large yellow arrow points from this `x` to a table on the right. The table has two columns: `&ndereço` (Address) and `Valor` (Value). The address column lists memory locations from `0xA1` to `0xAA`. The value column shows the value at `0xA1` as `500`, while other addresses show `0xA1` as their value.

&ndereço	Valor
0xA1	500
0xA2	
0xA3	
0xA4	
0xA5	0xA1
0xA6	
0xA7	
0xA8	
0xA9	
0xAA	

Ponteiros e funções

Passagem por REFERÊNCIA:

```
void somar(int *x);

int main() {
    setlocale(LC_ALL, "Portuguese");

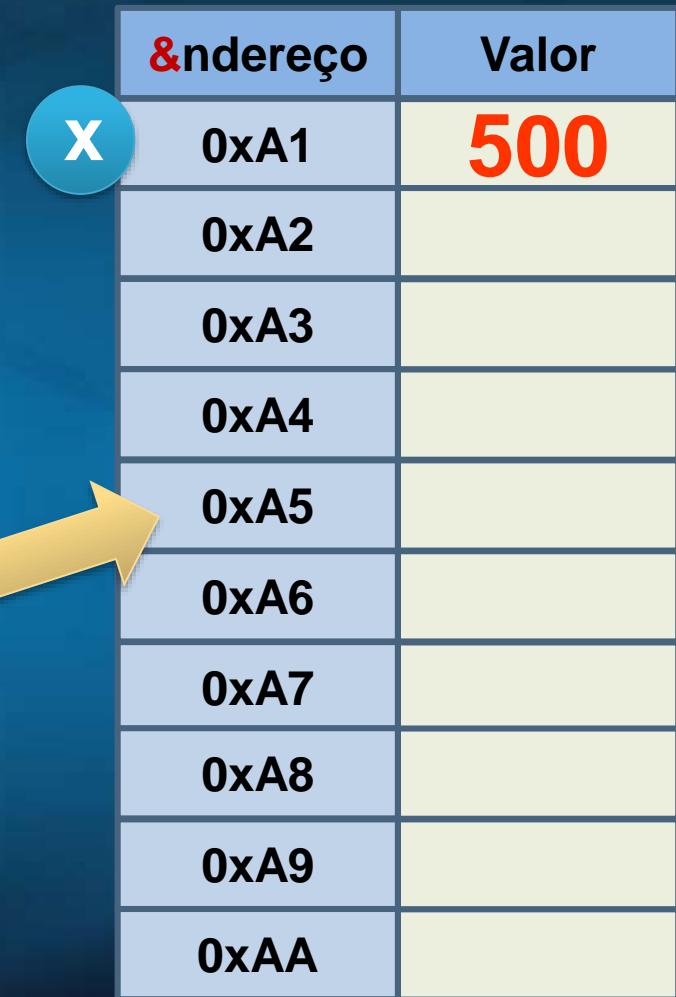
    int x = 100;

    somar(&x);

    cout << "Valor de X: " << x << endl;

    system("pause");
    return 0;
}

void somar(int *x) {
    *x + 400;
}
```



A diagram illustrating pointer semantics. A blue circle containing the letter 'X' is positioned above a yellow arrow. The arrow points from this circle to a table on the right. The table has two columns: 'Endereço' (Address) and 'Valor' (Value). The first row shows address 0xA1 with value 500, highlighted in red. The other rows (0xA2 to 0xAA) are empty.

Endereço	Valor
0xA1	500
0xA2	
0xA3	
0xA4	
0xA5	
0xA6	
0xA7	
0xA8	
0xA9	
0xAA	

Ponteiros e funções

Passagem por REFERÊNCIA:

```
void somar(int *x);

int main() {
    setlocale(LC_ALL, "Portuguese");

    int x = 100;

    somar(&x);

    cout << "Valor de X: " << x << endl;

    system("pause");
    return 0;
}

void somar(int *x) {
    *x = *x + 400;
}
```



&ndereço	Valor
0xA1	500
0xA2	
0xA3	
0xA4	
0xA5	
0xA6	
0xA7	
0xA8	
0xA9	
0xAA	

Problema - Exercício

- Como obter o retorno das três variáveis utilizando uma única função?

```
void obterXYZ(int x, int y, int z);

int main() {
    int x, y, z;

    x = 0;
    y = 0;
    z = 0;

    obterXYZ(x, y, z);

    cout << "Valor de X: " << x << endl;
    cout << "Valor de Y: " << y << endl;
    cout << "Valor de Z: " << z << endl;

    system("pause");
    return 0;
}

void obterXYZ(int x, int y, int z) {
    x = 100;
    y = 200;
    z = 300;
}
```

```

void obterXYZ(int *x, int *y, int *z);

int main() {
    setlocale(LC_ALL, "Portuguese");

    int x, y, z;

    x = 0;
    y = 0;
    z = 0;

    obterXYZ(&x, &y, &z);

    cout << "Valor de X: " << x << endl;
    cout << "Valor de Y: " << y << endl;
    cout << "Valor de Z: " << z << endl;

    system("pause");
    return 0;
}

void obterXYZ(int *x, int *y, int *z) {
    *x = 100;
    *y = 200;
    *z = 300;
}

```

Solução

&ndereço	Valor
0xA1	
0xA2	
0xA3	
0xA4	
0xA5	
0xA6	
0xA7	
0xA8	
0xA9	
0xAA	

```
void obterXYZ(int *x, int *y, int *z);

int main() {
    setlocale(LC_ALL, "Portuguese");

    int x, y, z;

    x = 0;
    y = 0;
    z = 0;

    obterXYZ(&x, &y, &z);

    cout << "Valor de X: " << x << endl;
    cout << "Valor de Y: " << y << endl;
    cout << "Valor de Z: " << z << endl;

    system("pause");
    return 0;
}

void obterXYZ(int *x, int *y, int *z) {
    *x = 100;
    *y = 200;
    *z = 300;
}
```

Solução



&Endereço	Valor
X	0xA1
y	0xA2
Z	0xA3
0xA4	
0xA5	
0xA6	
0xA7	
0xA8	
0xA9	
0xAA	

```
void obterXYZ(int *x, int *y, int *z);
```

```
int main() {  
    setlocale(LC_ALL, "Portuguese");
```

```
    int x, y, z;
```

```
    x = 0;  
    y = 0;  
    z = 0;
```

```
    obterXYZ(&x, &y, &z);
```

```
    cout << "Valor de X: " << x << endl;  
    cout << "Valor de Y: " << y << endl;  
    cout << "Valor de Z: " << z << endl;
```

```
    system("pause");
```

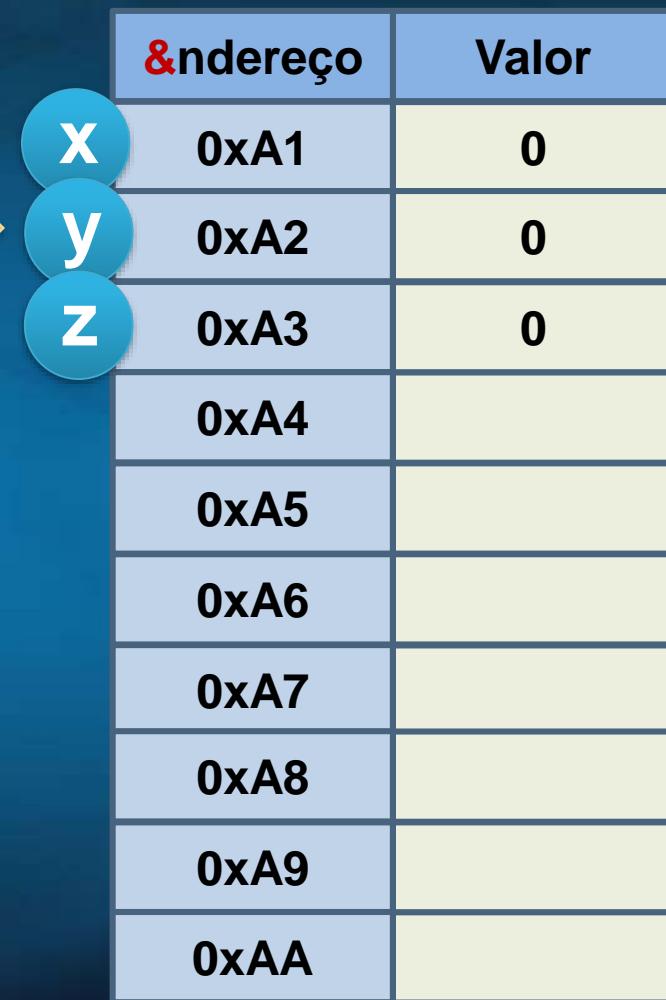
```
    return 0;
```

```
}
```

```
void obterXYZ(int *x, int *y, int *z) {  
    *x = 100;  
    *y = 200;  
    *z = 300;
```

```
}
```

Solução



&ndereço	Valor
X	0xA1
y	0xA2
Z	0xA3
0xA4	
0xA5	
0xA6	
0xA7	
0xA8	
0xA9	
0xAA	

```
void obterXYZ(int *x, int *y, int *z);

int main() {
    setlocale(LC_ALL, "Portuguese");

    int x, y, z;

    x = 0;
    y = 0;
    z = 0;

    obterXYZ(&x, &y, &z);

    cout << "Valor de X: " << x << endl;
    cout << "Valor de Y: " << y << endl;
    cout << "Valor de Z: " << z << endl;

    system("pause");
    return 0;
}

void obterXYZ(int *x, int *y, int *z) {
    *x = 100,
    *y = 200;
    *z = 300;
}
```

Solução

&ndereço	Valor
x	0xA1
y	0xA2
z	0xA3
0xA4	
0xA5	
0xA6	
x	0xA1
y	0xA2
z	0xA3
0xAA	

```

void obterXYZ(int *x, int *y, int *z);

int main() {
    setlocale(LC_ALL, "Portuguese");

    int x, y, z;

    x = 0;
    y = 0;
    z = 0;

    obterXYZ(&x, &y, &z);

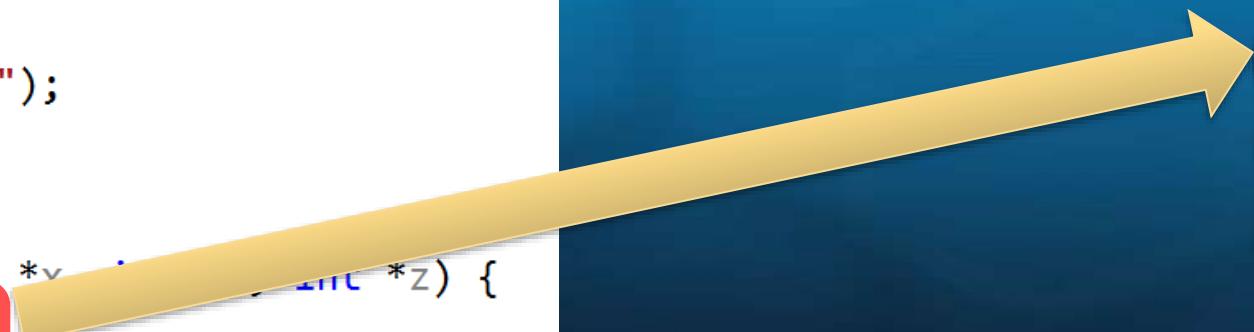
    cout << "Valor de X: " << x << endl;
    cout << "Valor de Y: " << y << endl;
    cout << "Valor de Z: " << z << endl;

    system("pause");
    return 0;
}

void obterXYZ(int *x, int *y, int *z) {
    *x = 100;
    *y = 200;
    *z = 300;
}

```

Solução



&ndereço	Valor
X	100
y	0
Z	0
0xA4	
0xA5	
0xA6	
0xA7	0xA1
0xA8	0xA2
0xA9	0xA3
0xAA	

```

void obterXYZ(int *x, int *y, int *z);

int main() {
    setlocale(LC_ALL, "Portuguese");

    int x, y, z;

    x = 0;
    y = 0;
    z = 0;

    obterXYZ(&x, &y, &z);

    cout << "Valor de X: " << x << endl;
    cout << "Valor de Y: " << y << endl;
    cout << "Valor de Z: " << z << endl;

    system("pause");
    return 0;
}

void obterXYZ(int *x, int *y, int *z)
{
    *x = 100;
    *y = 200;
    *z = 300;
}

```

Solução

The diagram illustrates the state of memory after the execution of the provided C++ code. A yellow arrow points from the variable declarations in the code to the memory dump table, specifically highlighting the addresses of x, y, and z.

&ndereço	Valor	
X	0xA1	100
y	0xA2	200
Z	0xA3	0
0xA4		*
0xA5		
0xA6		
0xA7	0xA1	
0xA8	0xA2	
0xA9	0xA3	
0xAA		

```
void obterXYZ(int *x, int *y, int *z);

int main() {
    setlocale(LC_ALL, "Portuguese");

    int x, y, z;

    x = 0;
    y = 0;
    z = 0;

    obterXYZ(&x, &y, &z);

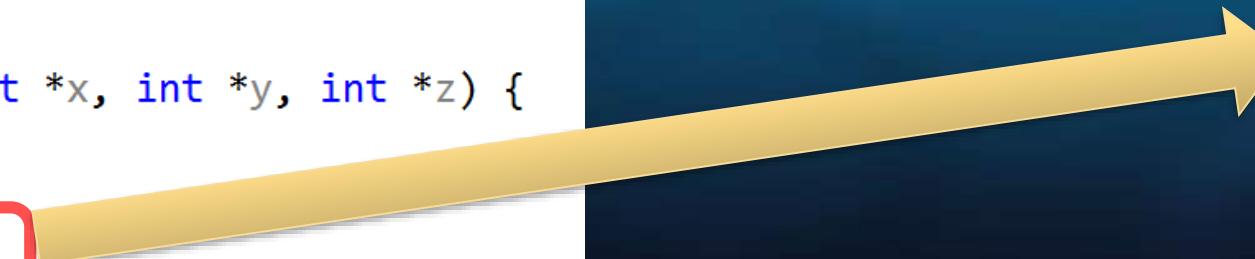
    cout << "Valor de X: " << x << endl;
    cout << "Valor de Y: " << y << endl;
    cout << "Valor de Z: " << z << endl;

    system("pause");
    return 0;
}

void obterXYZ(int *x, int *y, int *z) {
    *x = 100;
    *y = 200;
    *z = 300;
}
```

Solução

&ndereço	Valor
X	0xA1
y	0xA2
Z	300
0xA4	
0xA5	
0xA6	
0xA7	0xA1
0xA8	0xA2
0xA9	0xA3
0xAA	



```
void obterXYZ(int *x, int *y, int *z);

int main() {
    setlocale(LC_ALL, "Portuguese");

    int x, y, z;

    x = 0;
    y = 0;
    z = 0;

    obterXYZ(&x, &y, &z);

    cout << "Valor de X: " << x << endl;
    cout << "Valor de Y: " << y << endl;
    cout << "Valor de Z: " << z << endl;

    system("pause");
    return 0;
}

void obterXYZ(int *x, int *y, int *z) {
    *x = 100;
    *y = 200;
    *z =
```

Solução

&ndereço	Valor
X	0xA1
y	0xA2
Z	0xA3
0xA4	
0xA5	
0xA6	
0xA7	
0xA8	
0xA9	
0xAA	

}

```
void obterXYZ(int *x, int *y, int *z);
```

```
int main() {  
    setlocale(LC_ALL, "Portuguese");
```

```
    int x, y, z;
```

```
    x = 0;
```

```
    y = 0;
```

```
    z = 0;
```

```
    obterXYZ(&x, &y, &z);
```

```
    cout << "Valor de X: " << x << endl;  
    cout << "Valor de Y: " << y << endl;  
    cout << "Valor de Z: " << z << endl;
```

```
    system("pause");
```

```
    return 0;
```

```
}
```

```
void obterXYZ(int *x, int *y, int *z) {  
    *x = 100;  
    *y = 200;  
    *z = 300;
```

```
}
```

Solução

&ndereço	Valor
X	0xA1
y	0xA2
Z	0xA3
0xA4	
0xA5	
0xA6	
0xA7	
0xA8	
0xA9	
0xAA	

Passando ponteiros para funções

```
void somar(int *x);

int main() {
    setlocale(LC_ALL, "Portuguese");

    int x = 100;
    int *px;

    px = &x;

    somar(px);

    cout << "Valor de X: " << x << endl;

    system("pause");
    return 0;
}

void somar(int *x) {
    *x = *x + 400;
}
```

A diagram illustrating the state of memory. On the left, a white box contains C++ code. A red rectangle highlights the declaration of variable `x`. A large yellow arrow points from this highlighted area to a table on the right. The table has two columns: `&ndereço` (Address) and `Valor` (Value). The address `0xA1` is highlighted in blue. The value at `0xA1` is `100`, which corresponds to the value of `x` in the code.

&ndereço	Valor
0xA1	100
0xA2	
0xA3	
0xA4	
0xA5	
0xA6	
0xA7	
0xA8	
0xA9	
0xAA	

Passando ponteiros para funções

```
void somar(int *x);

int main() {
    setlocale(LC_ALL, "Portuguese");

    int x = 100;
    int *pX;

    pX = &x;

    somar(pX);

    cout << "Valor de X: " << x << endl;

    system("pause");
    return 0;
}

void somar(int *x) {
    *x = *x + 400;
}
```

A diagram illustrating pointer passing. On the left, a code snippet shows a variable `pX` being assigned the address of variable `x`. A large yellow arrow points from the variable `pX` in the code to a memory dump on the right. The memory dump is a table with two columns: `&ndereço` (Address) and `Valor` (Value). The table shows memory starting at address `0xA1`, where the value `100` is stored. Subsequent memory locations `0xA2` through `0xAA` are shown as empty cells.

&ndereço	Valor
0xA1	100
0xA2	
0xA3	
0xA4	
0xA5	
0xA6	
0xA7	
0xA8	
0xA9	
0xAA	

Passando ponteiros para funções

```
void somar(int *x);

int main() {
    setlocale(LC_ALL, "Portuguese");

    int x = 100;
    int *pX;

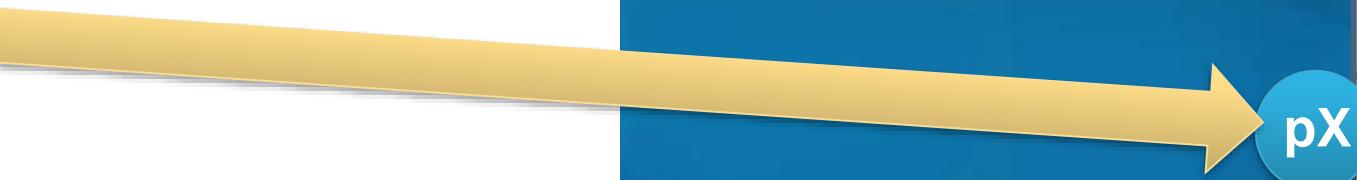
    pX = &x; pX = &x;

    somar(pX);

    cout << "Valor de X: " << x << endl;

    system("pause");
    return 0;
}

void somar(int *x) {
    *x = *x + 400;
}
```



&ndereço	Valor
0xA1	100
0xA2	
0xA3	
0xA4	
0xA5	0xA1
0xA6	
0xA7	
0xA8	
0xA9	
0xAA	

Passando ponteiros para funções

```
void somar(int *x);

int main() {
    setlocale(LC_ALL, "Portuguese");

    int x = 100;
    int *pX;

    pX = &x;

    somar(pX);

    cout << "Valor de X: " << x << endl;

    system("pause");
    return 0;
}

void somar(int *x) {
    *x = *x + 400;
}
```

&ndereço	Valor
X	0xA1
0xA2	
0xA3	
0xA4	
pX	0xA1
0xA6	
0xA7	
0xA8	
0xA9	0xA1
0xAA	

Passando ponteiros para funções

```
void somar(int *x);

int main() {
    setlocale(LC_ALL, "Portuguese");

    int x = 100;
    int *pX;

    pX = &x;

    somar(pX);

    cout << "Valor de X: " << x << endl;

    system("pause");
    return 0;
}

void somar(int *x) {
    *x = *x + 400;
}
```

&ndereço	Valor
X	0xA1
0xA2	
0xA3	
0xA4	
pX	0xA1
0xA6	
0xA7	
0xA8	
X	0xA1
0xAA	

Passando ponteiros para funções

```
void somar(int *x);

int main() {
    setlocale(LC_ALL, "Portuguese");

    int x = 100;
    int *pX;

    pX = &x;

    somar(pX);

    cout << "Valor de X: " << x << endl;

    system("pause");
    return 0;
}

void somar(int *x) {
    *x = *
```

&ndereço	Valor
X	0xA1
0xA2	
0xA3	
0xA4	
pX	0xA1
0xA6	
0xA7	
0xA8	
0xA9	
0xAA	

Exercício

O método `misterio(&i,&j)` tem um problema. Qual é?
Antes da chamada do método,
temos a seguinte linha de comando: `int i=6, j=10;`

```
void misterio(int *p, int *q){  
    int *temp;  
    *temp = *p;  
    *p = *q;  
    *q = *temp;  
}
```

Exercício

1) Fazer uma função que retorna a soma e a subtração entre dois números. Após a função, imprima os dados na tela.

Ex: void calculo(int nro1, int nro2, int *soma, int *sub)

2) Fazer uma função para ler e retornar o valor das 4 notas de um aluno e sua média. Após a função, imprima os dados na tela.

3) Fazer uma função que recebe um número, calcule e retorne se o número é positivo ou negativo e se é múltiplo ou não de três. Após a função, imprima os dados na tela.

Ex: void validarNro(int nro1, bool *positivo, bool *multiploTres)

Exercício 1

```
void calculadora(int nro1, int nro2, int *sm, int *sb);

int main() {
    setlocale(LC_ALL, "Portuguese");
    int soma, subtracao;

    calculadora(20, 10, &soma, &subtracao);

    cout << "Soma: " << soma << endl;
    cout << "Subtração: " << subtracao << endl;

    system("pause");
    return 0;
}

void calculadora(int nro1, int nro2, int *sm, int *sb) {
    *sm = nro1 + nro2;
    *sb = nro1 - nro2;
}
```

Exercício 2 – Parte 1

```
void notas(float *n1, float *n2, float *n3, float *n4, float *media);

int main() {
    setlocale(LC_ALL, "Portuguese");
    float nota1, nota2, nota3, nota4, media;

    notas(&nota1, &nota2, &nota3, &nota4, &media);

    cout << "Nota 1: " << nota1 << endl;
    cout << "Nota 2: " << nota2 << endl;
    cout << "Nota 3: " << nota3 << endl;
    cout << "Nota 4: " << nota4 << endl;
    cout << "Média: " << media << endl;

    system("pause");
    return 0;
}
```

Exercício 2 – Parte 2

```
void notas(float *n1, float *n2, float *n3, float *n4, float *media) {  
  
    cout << "Informe a nota 1: ";  
    cin >> *n1;  
  
    cout << "Informe a nota 2: ";  
    cin >> *n2;  
  
    cout << "Informe a nota 3: ";  
    cin >> *n3;  
  
    cout << "Informe a nota 4: ";  
    cin >> *n4;  
  
    *media = (*n1 + *n2 + *n3 + *n4) / 4;  
}
```

Exercício 3 – Parte 1

```
void validarNro(int nro1, bool *positivo, bool *multiploTres);

int main() {
    setlocale(LC_ALL, "Portuguese");

    int nro1;
    bool pos, multTres;

    nro1 = 9;

    validarNro(nro1, &pos, &multTres);

    if (pos) {
        cout << "O número é positivo." << endl;
    }
    else {
        cout << "O número é negativo." << endl;
    }

    if (multTres) {
        cout << "O número é múltiplo de três." << endl;
    }
    else {
        cout << "O número não é múltiplo de três." << endl;
    }

    system("pause");
    return 0;
}
```

Exercício 3 – Parte 2

```
void validarNro(int nro1, bool *positivo, bool *multiploTres)
{
    if (nro1 >= 0) {
        *positivo = true;
    }
    else {
        *positivo = false;
    }

    if ((nro1 % 3) == 0) {
        *multiploTres = true;
    }
    else {
        *multiploTres = false;
    }
}
```

FIM

“Nossa história, depositada nas mãos de Deus, pode ser reescrita a qualquer momento”.

(Pe. Luís Erlin - 9 Meses com Maria)

Prof. Dr. Ricardo Luis Balieiro
ricbalieiro@gmail.com