Tecnologia Em Analise e Desenv. de Sistemas

Estrutura de Dados

Aula 2 – Recursividade

Prof. Dr. Ricardo Luis Balieiro

Recursividade

- Para cada chamada de uma função, recursiva ou não, os parâmetros e as variáveis locais são empilhados na pilha de execução.

Recursividade

- Internamente, quando qualquer chamada de função é feita dentro de um programa, é criado um Registro de Ativação na Pilha de Execução do programa
- O registro de ativação armazena os parâmetros e variáveis locais da função bem como o “ponto de retorno” no programa ou subprograma que chamou essa função.
- Ao final da execução dessa função, o registro é desempilhado e a execução volta ao subprograma que chamou a função

Recursividade

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

    int x, y;

    x = 100;
    y = 200;

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

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

Dados

Instruções

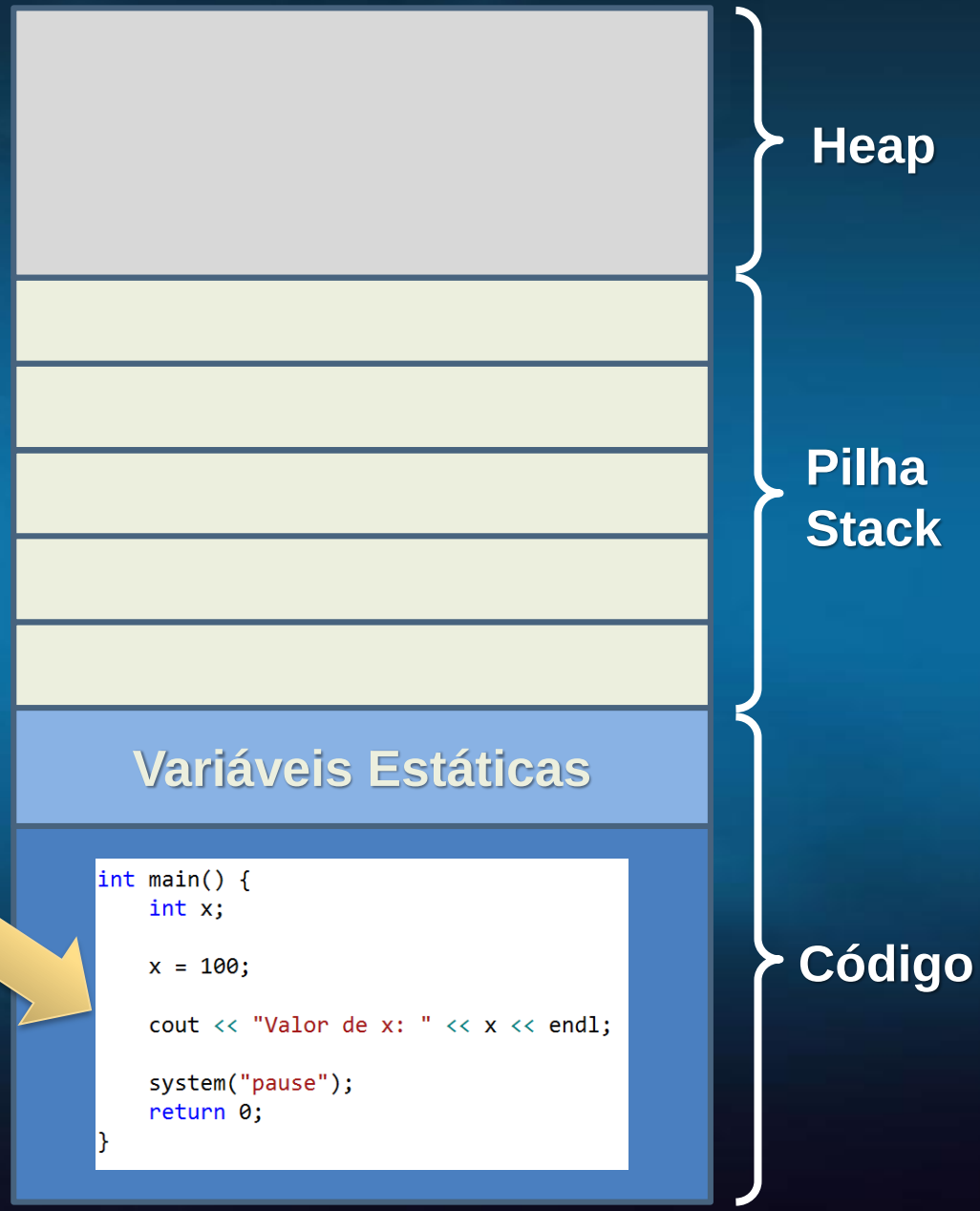
Recursividade

Solicita a execução o código

```
int main() {
    int x;

    x = 100;

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

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

Recursividade

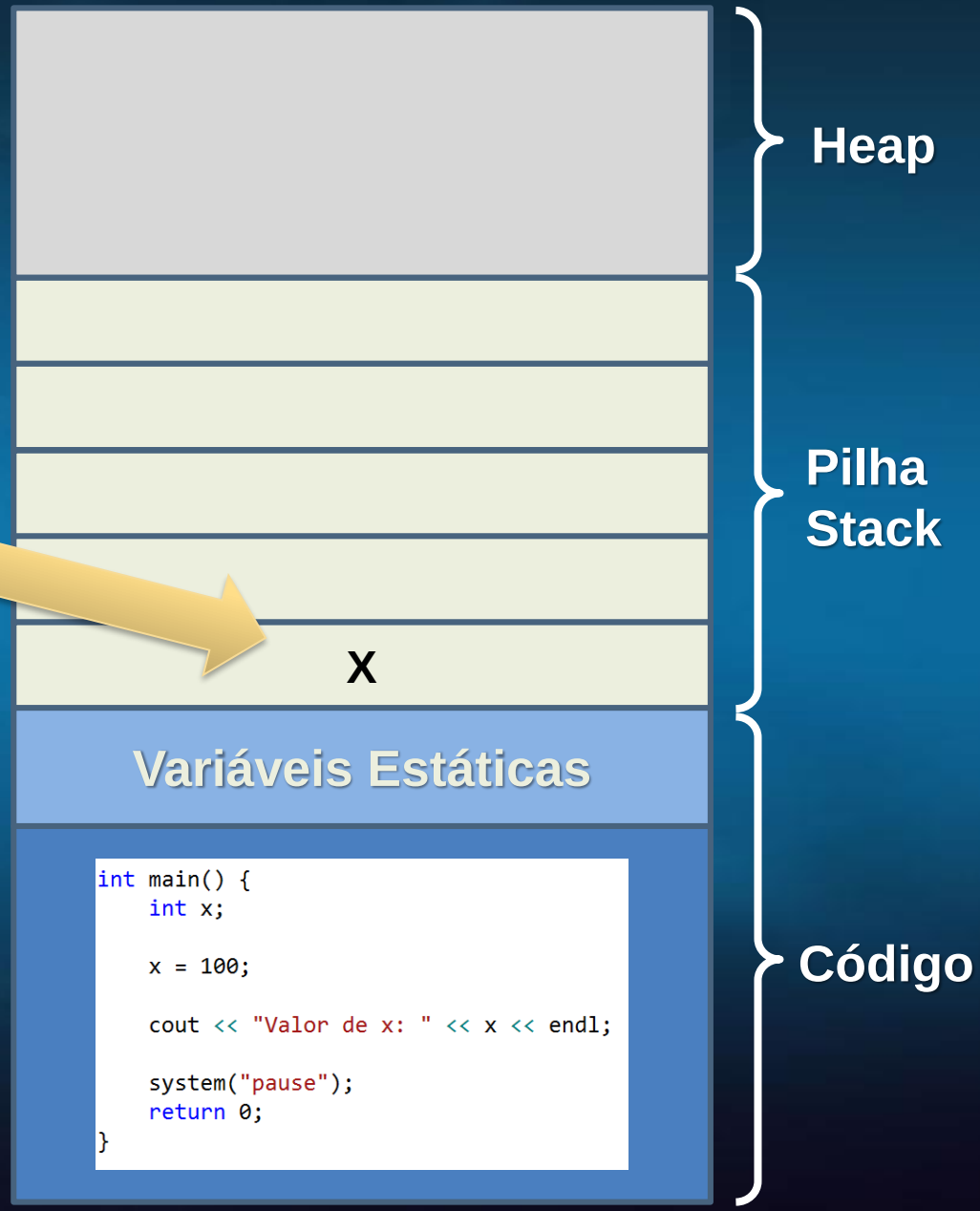
```
int main() {
    int x;

    x = 100;

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

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



LOAD ➔ Carrega o código na memória para a EXECUÇÃO.

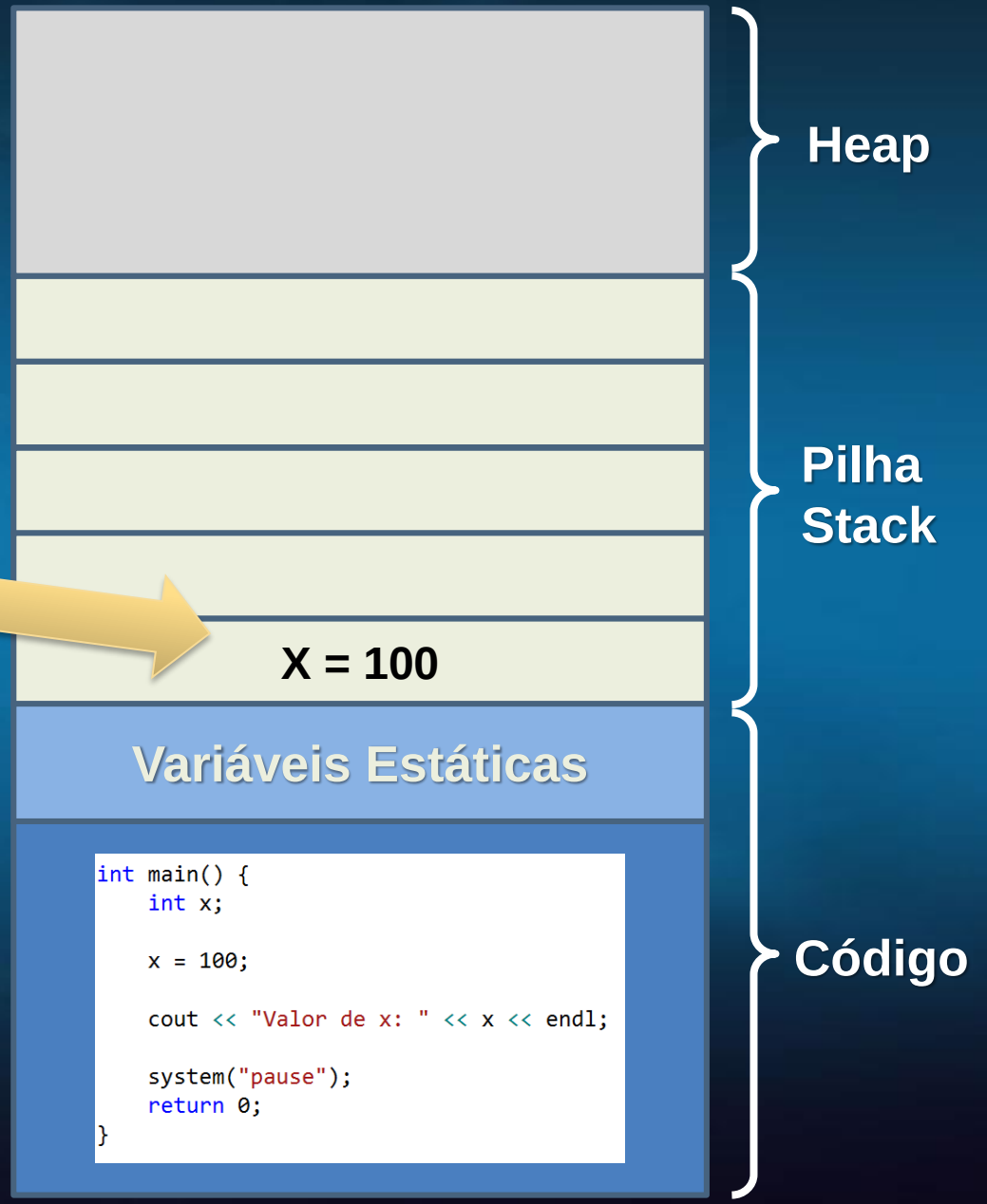
Recursividade

```
int main() {
    int x;

    x = 100;

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

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



Recursividade

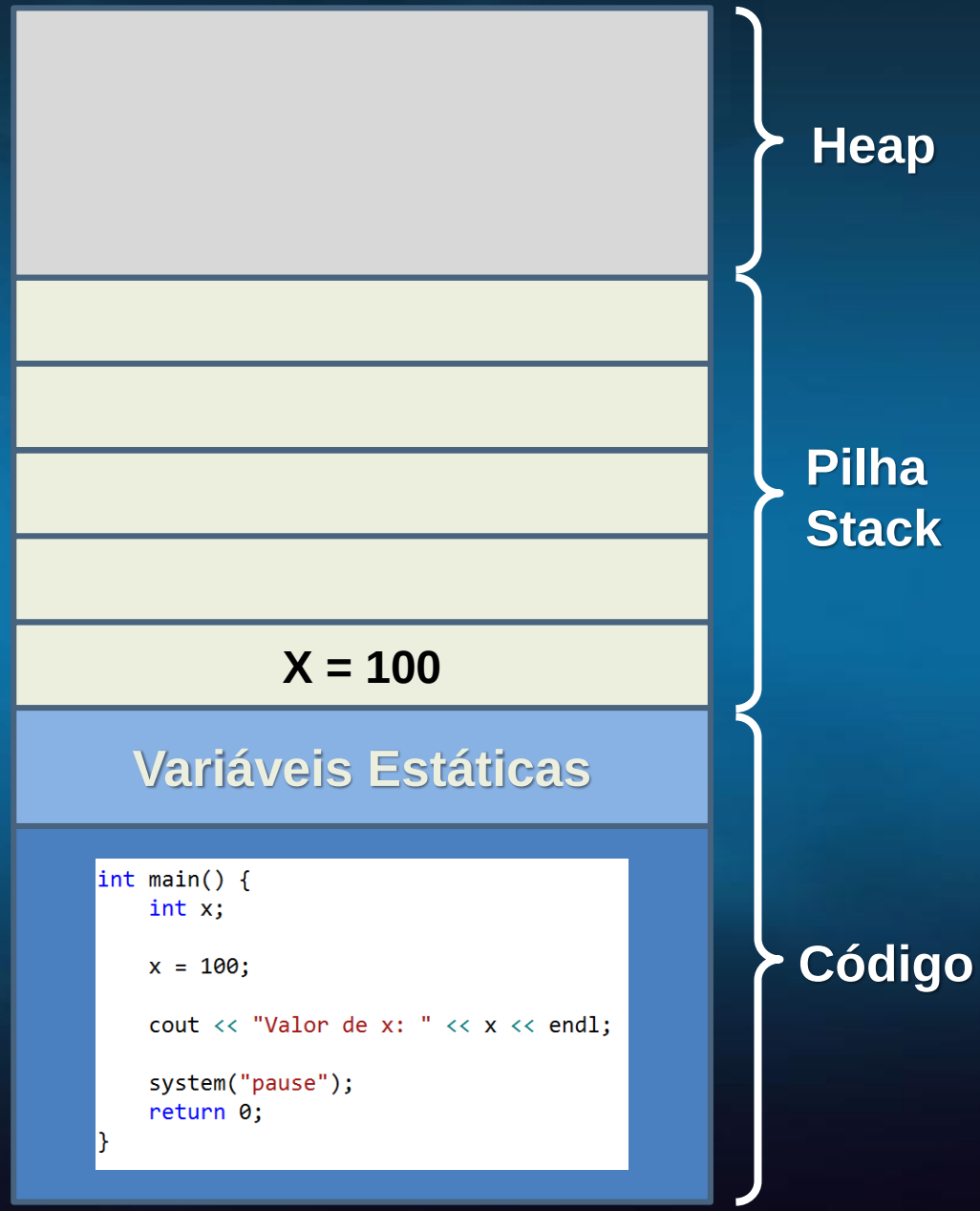
```
int main() {
    int x;

    x = 100;

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

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



Recursividade

```
int main() {
    int x;

    x = 100;

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

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

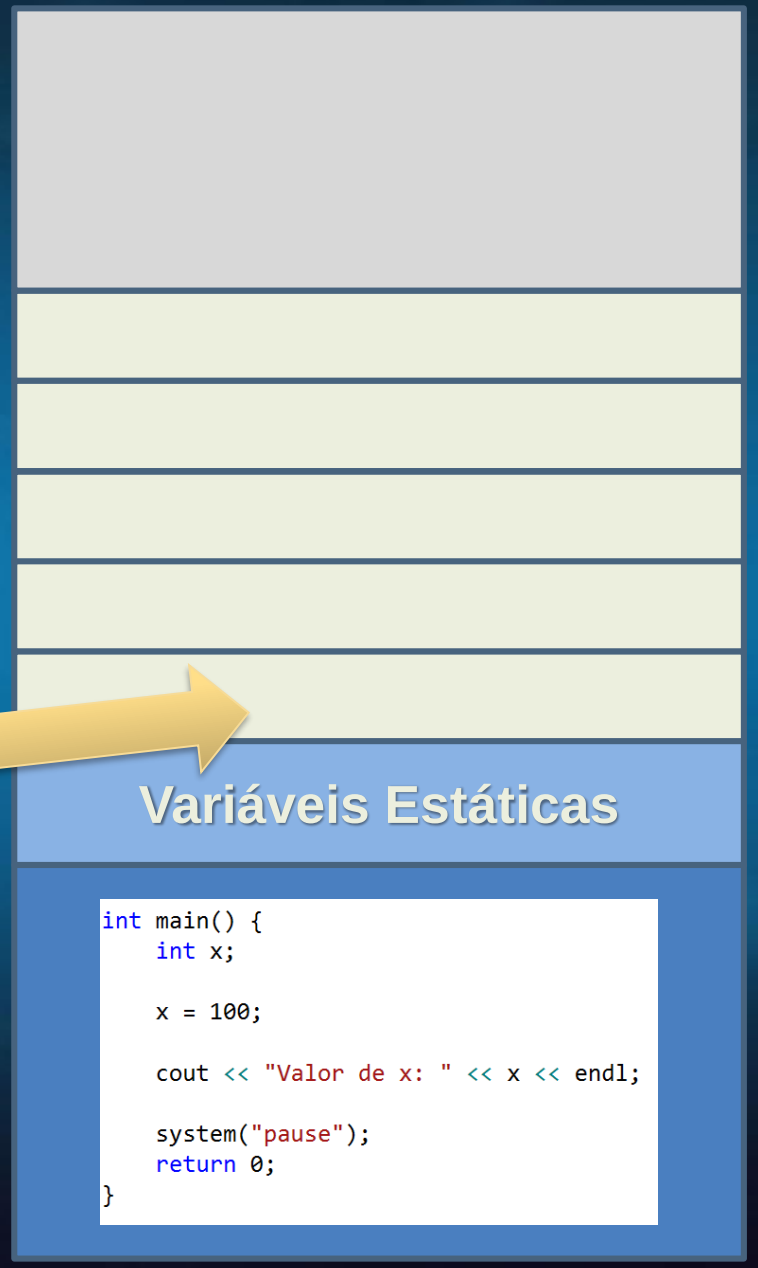
Saída
100



Recursividade

```
int main() {
    int x;

    x = 100;

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

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

Saída
100



Recursividade

```
int main() {
    int x;

    x = 100;

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

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

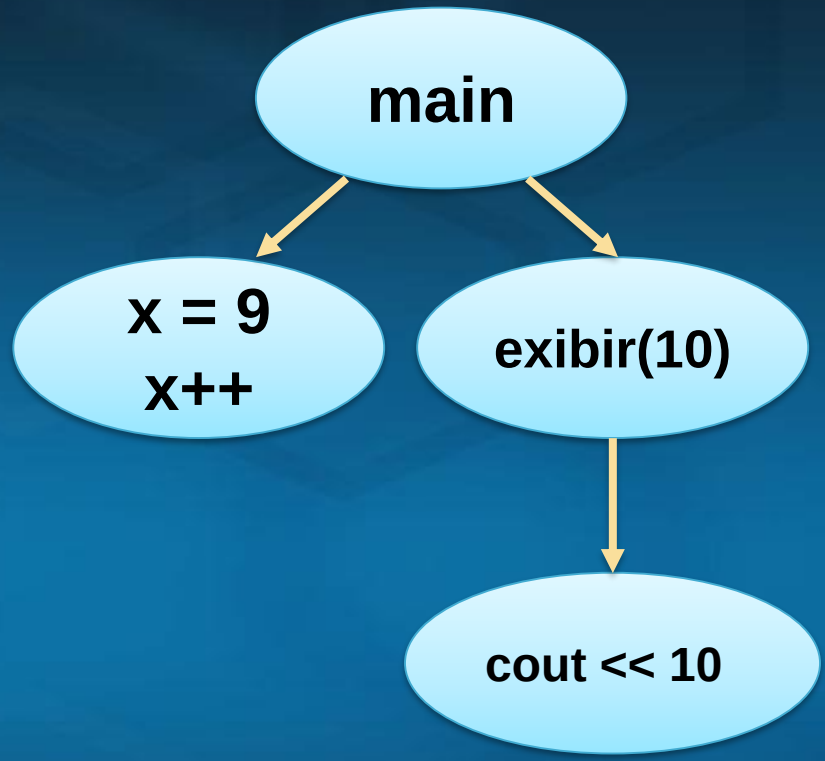
Saída
100

Chamada de Função

Recursividade

Chamada de função

```
void exibir(int vlr);

int main() {
    int x;

    x = 9;
    x++;

    exibir(x);

    system("pause");
    return 0;
}

void exibir(int vlr) {

    cout << "Valor de vlr: " << vlr << endl;
}
```

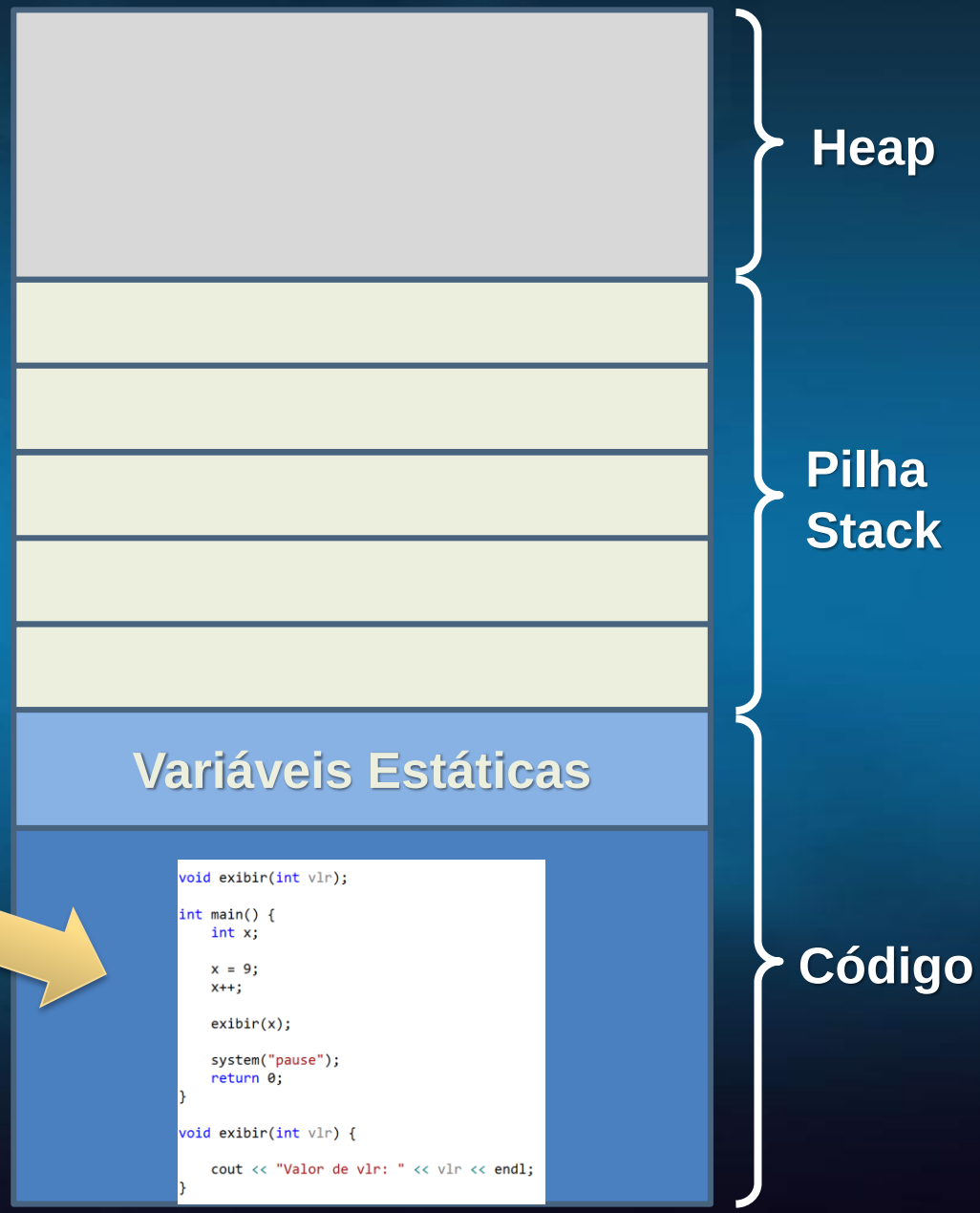


Saída
10

Chamada de função

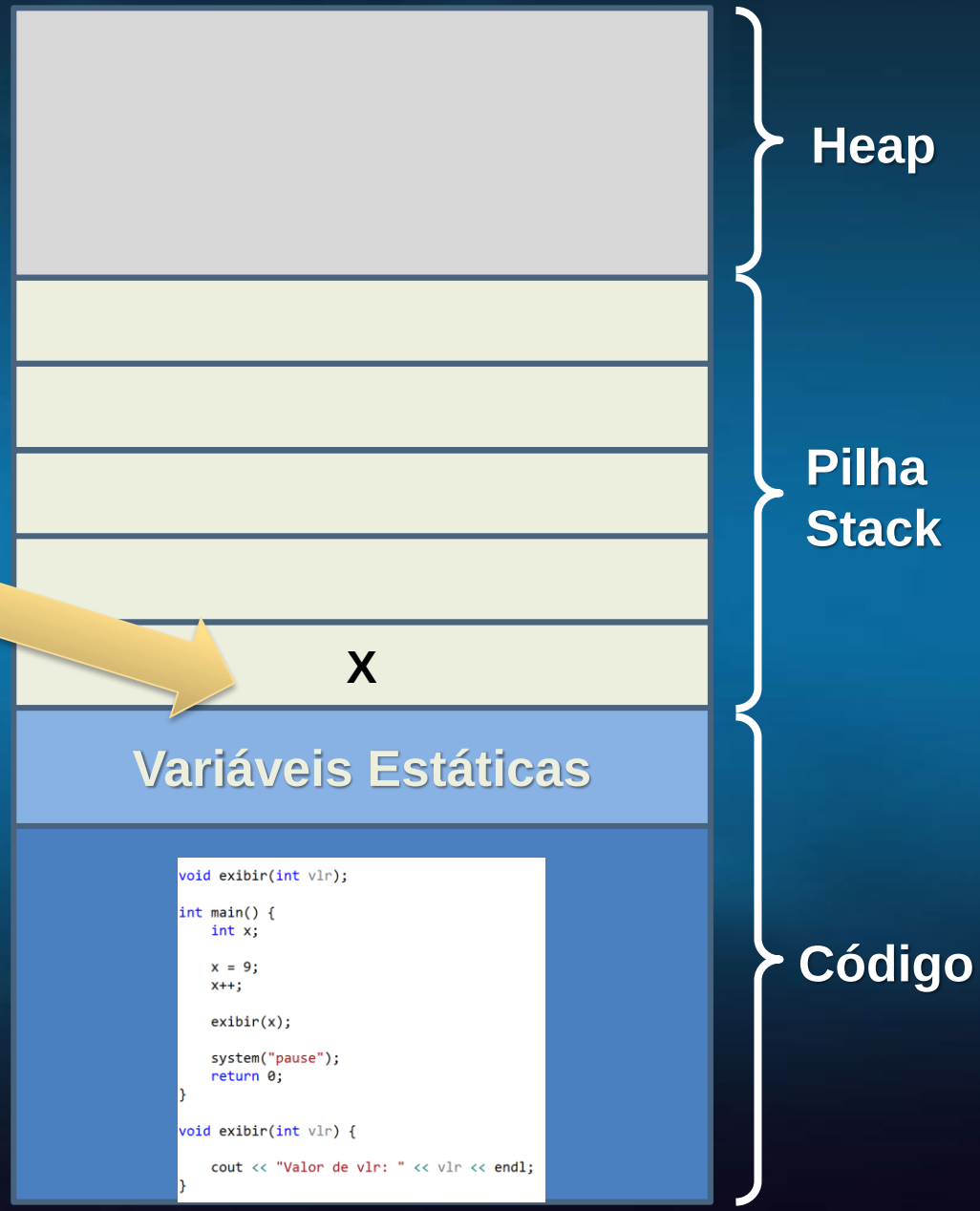
```
void exibir(int vlr);

int main() {
    int x;

    x = 9;
    x++;

    exibir(x);

    system("pause");
    return 0;
}

void exibir(int vlr) {

    cout << "Valor de vlr: " << vlr << endl;
}
```

LOAD



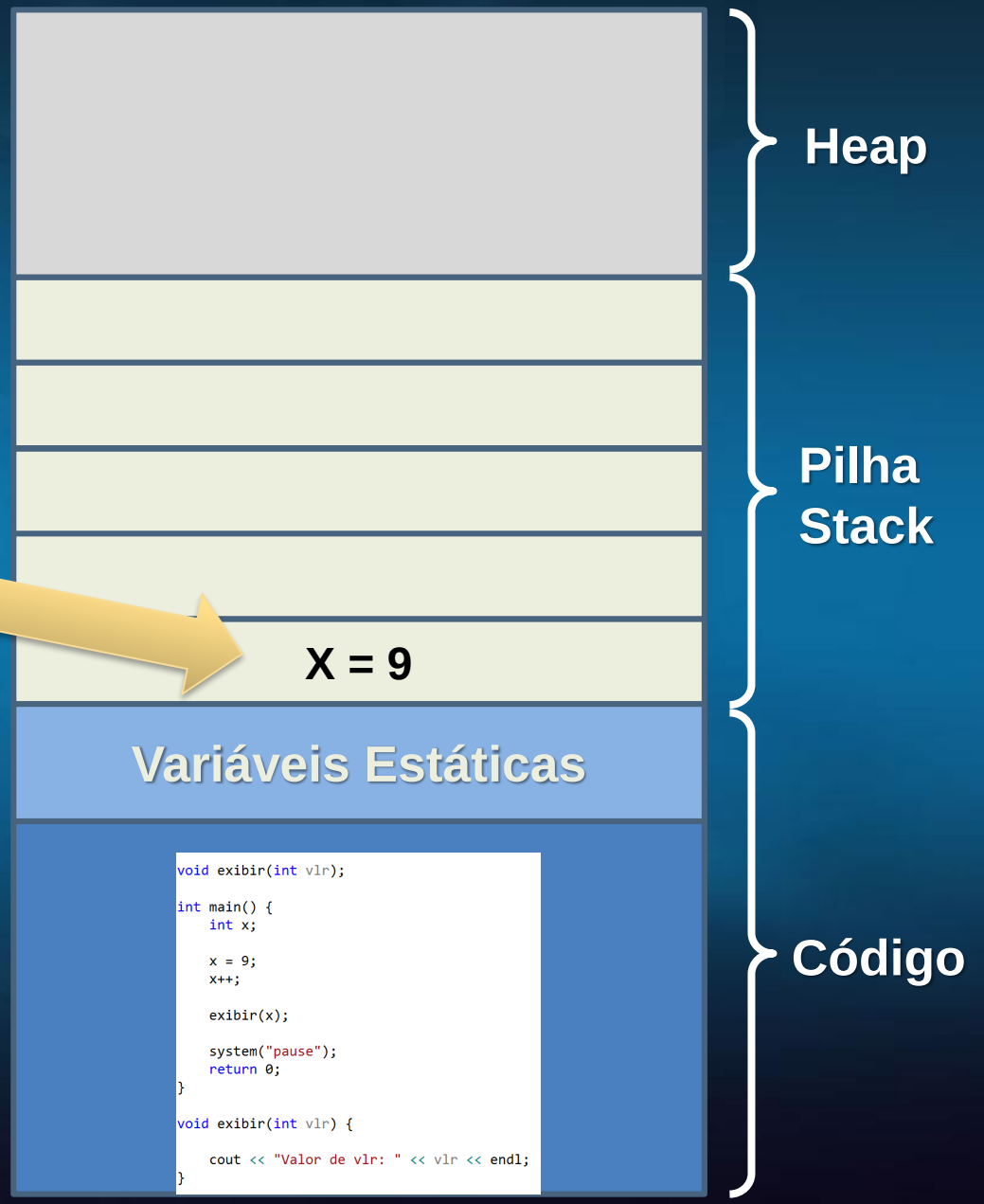
Chamada de função

```
void exibir(int vlr);

int main() {
    int x;

    x = 9;
    x++;

    exibir(x);

    system("pause");
    return 0;
}

void exibir(int vlr) {

    cout << "Valor de vlr: " << vlr << endl;
}
```



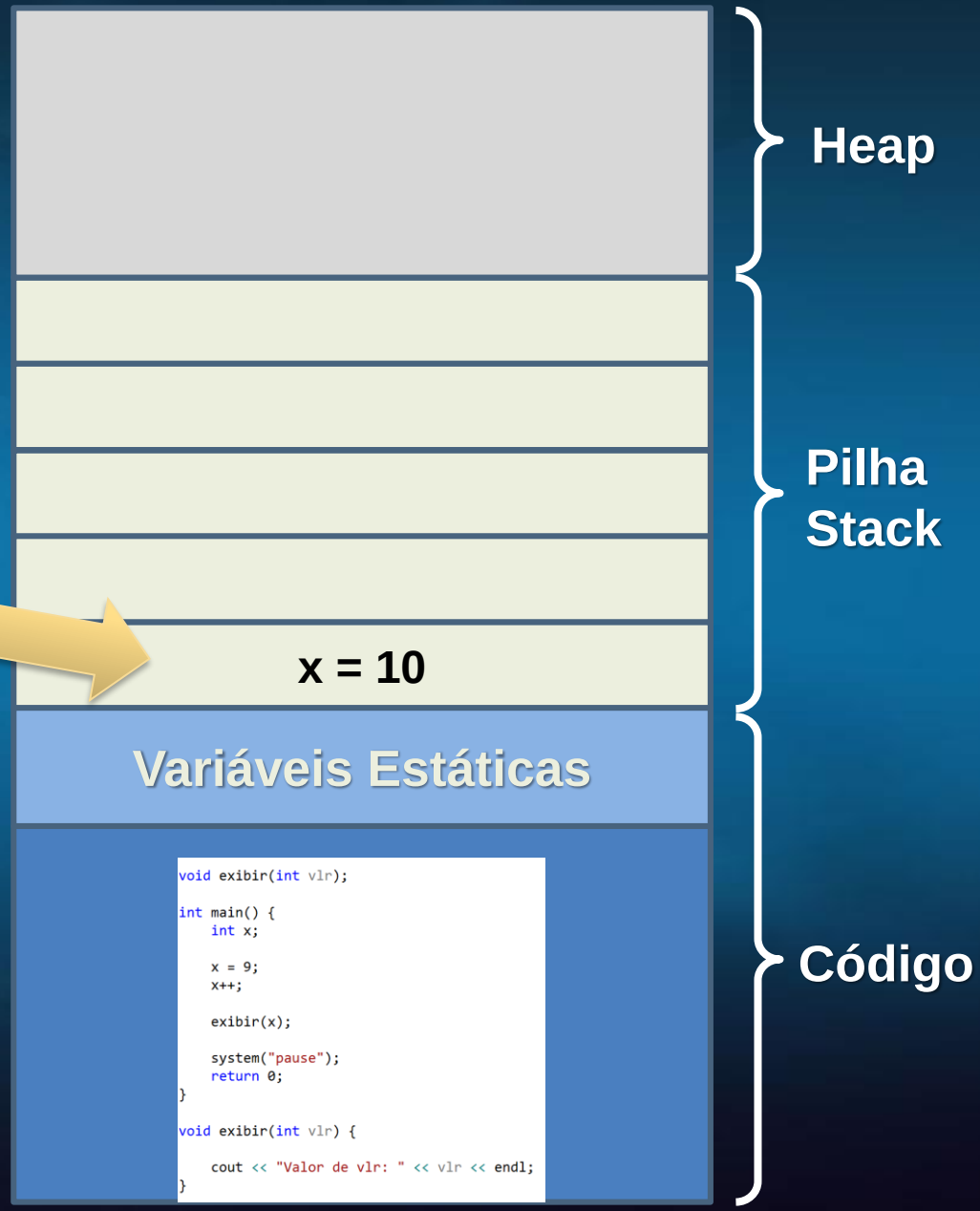
Chamada de função

```
void exibir(int vlr);

int main() {
    int x;

    x = 9;
    x++;

    exibir(x);

    system("pause");
    return 0;
}

void exibir(int vlr) {

    cout << "Valor de vlr: " << vlr << endl;
}
```



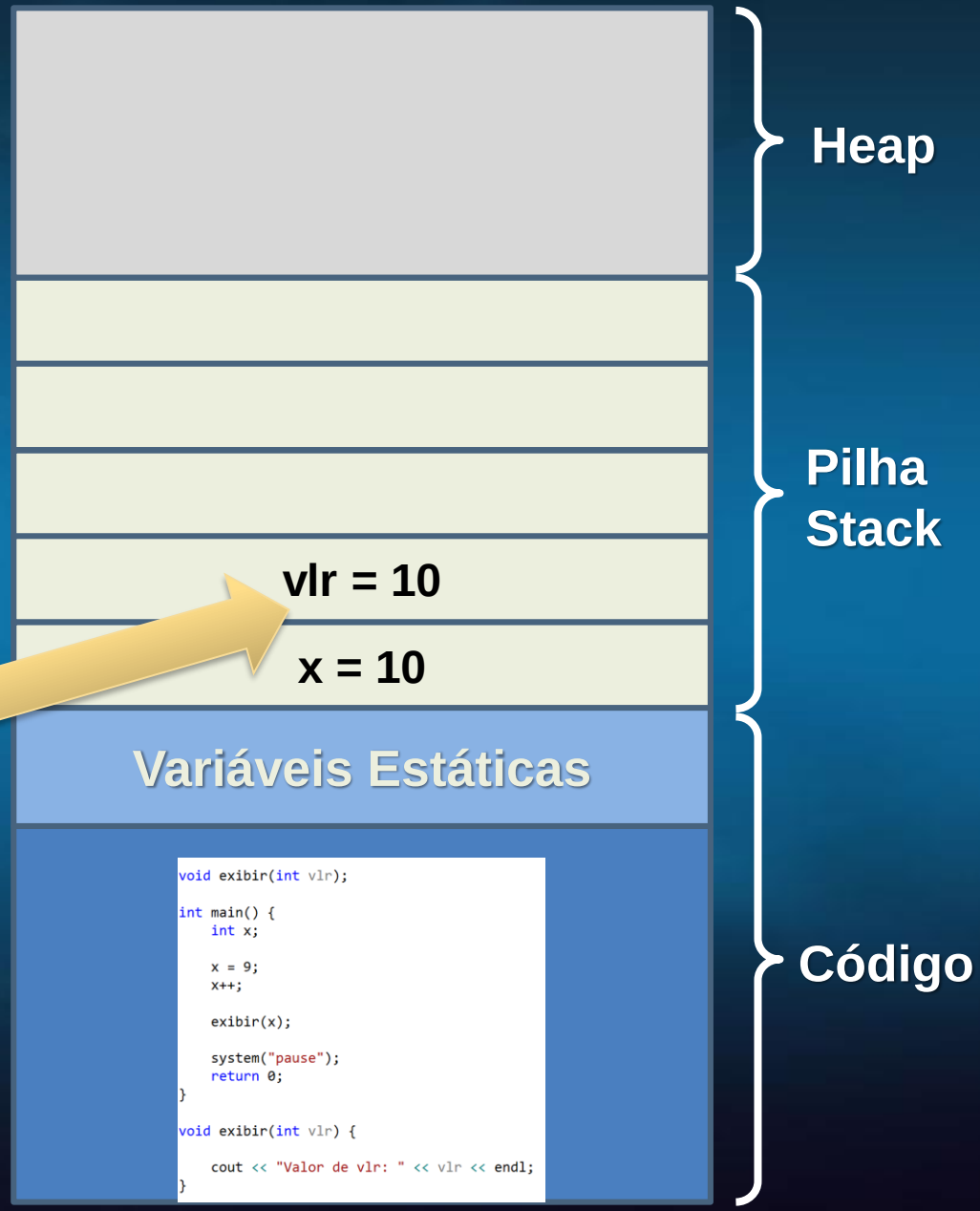
Chamada de função

```
void exibir(int vlr);

int main() {
    int x;

    x = 9;
    x++;

    exibir(x);

    system("pause");
    return 0;
}

void exibir(int vlr) {

    cout << "Valor de vlr: " << vlr << endl;
}
```



Chamada de função

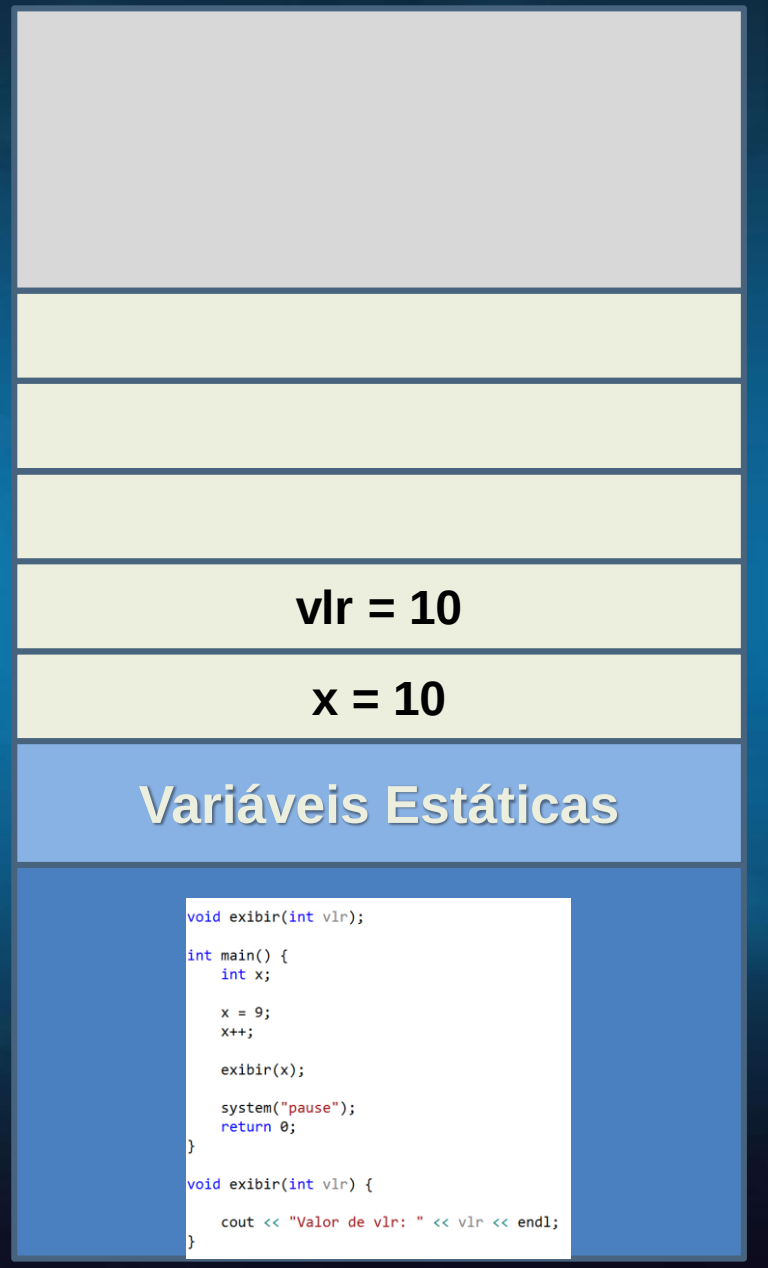
```
void exibir(int vlr);

int main() {
    int x;

    x = 9;
    x++;

    exibir(x);

    system("pause");
    return 0;
}

void exibir(int vlr) {

    cout << "Valor de vlr: " << vlr << endl;
}
```



Chamada de função

```
void exibir(int vlr);

int main() {
    int x;

    x = 9;
    x++;

    exibir(x);

    system("pause");
    return 0;
}

void exibir(int vlr) {

    cout << "Valor de vlr: " << vlr << endl;
}
```

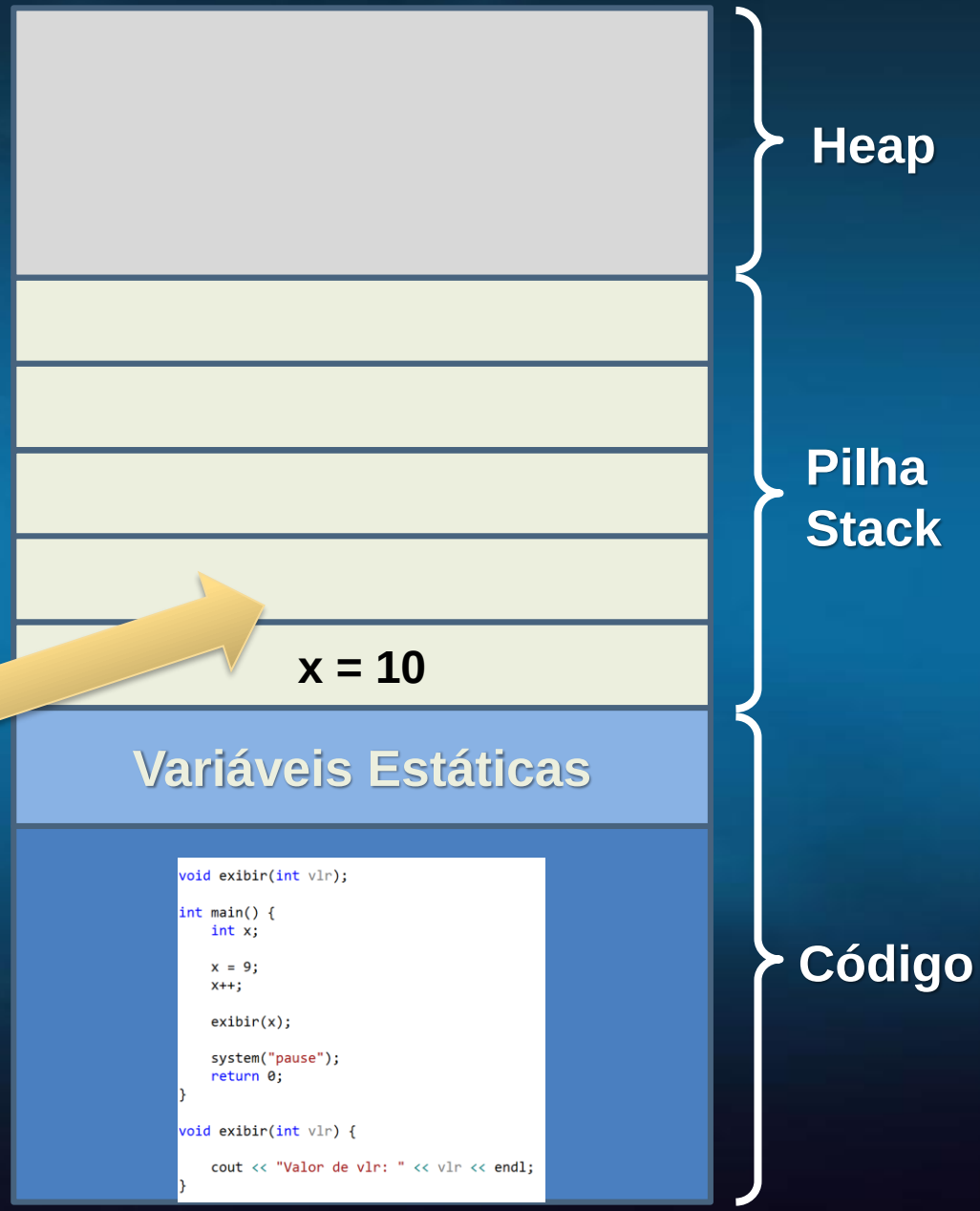
Saída
10



Chamada de função

```
void exibir(int vlr);

int main() {
    int x;

    x = 9;
    x++;

    exibir(x);

    system("pause");
    return 0;
}

void exibir(int vlr) {

    cout << "Valor de vlr: " << vlr << endl;
}
```

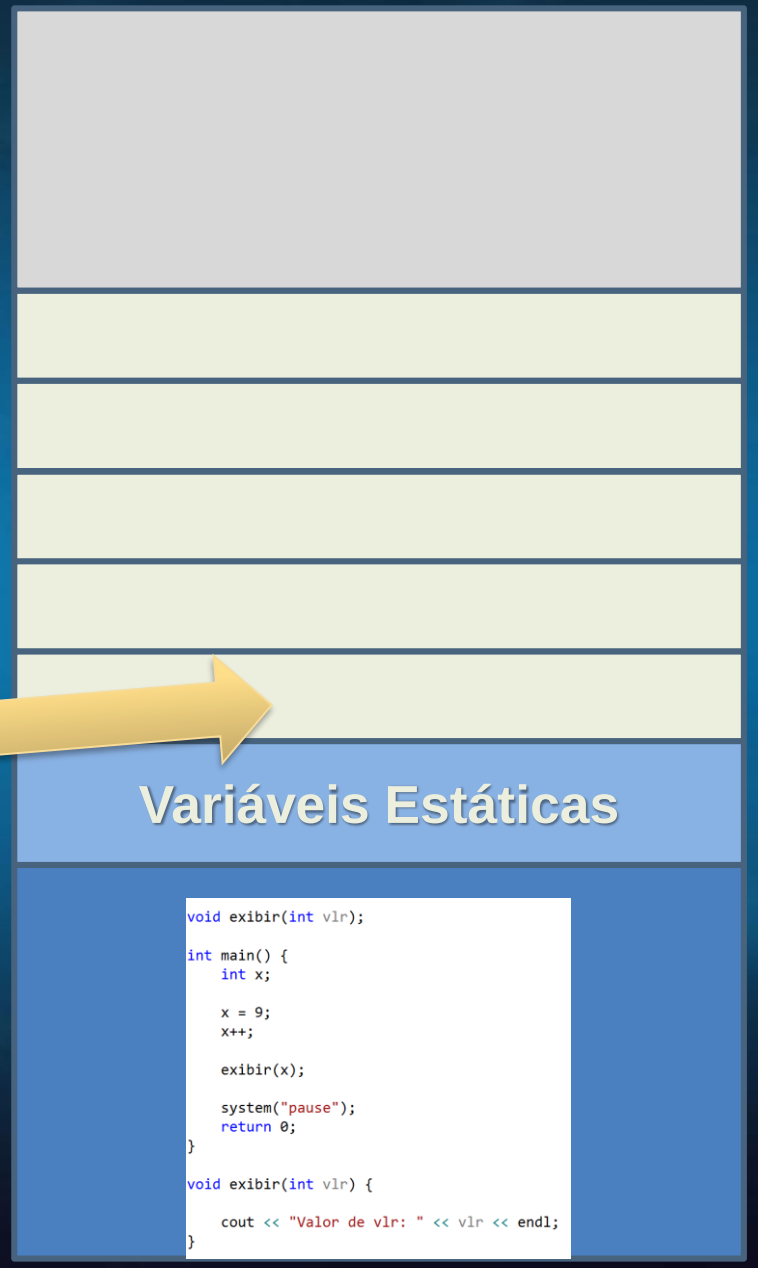
Saída
10



Chamada de função

```
void exibir(int vlr);

int main() {
    int x;

    x = 9;
    x++;

    exibir(x);

    system("pause");
    return 0;
}

void exibir(int vlr) {

    cout << "Valor de vlr: " << vlr << endl;
}
```

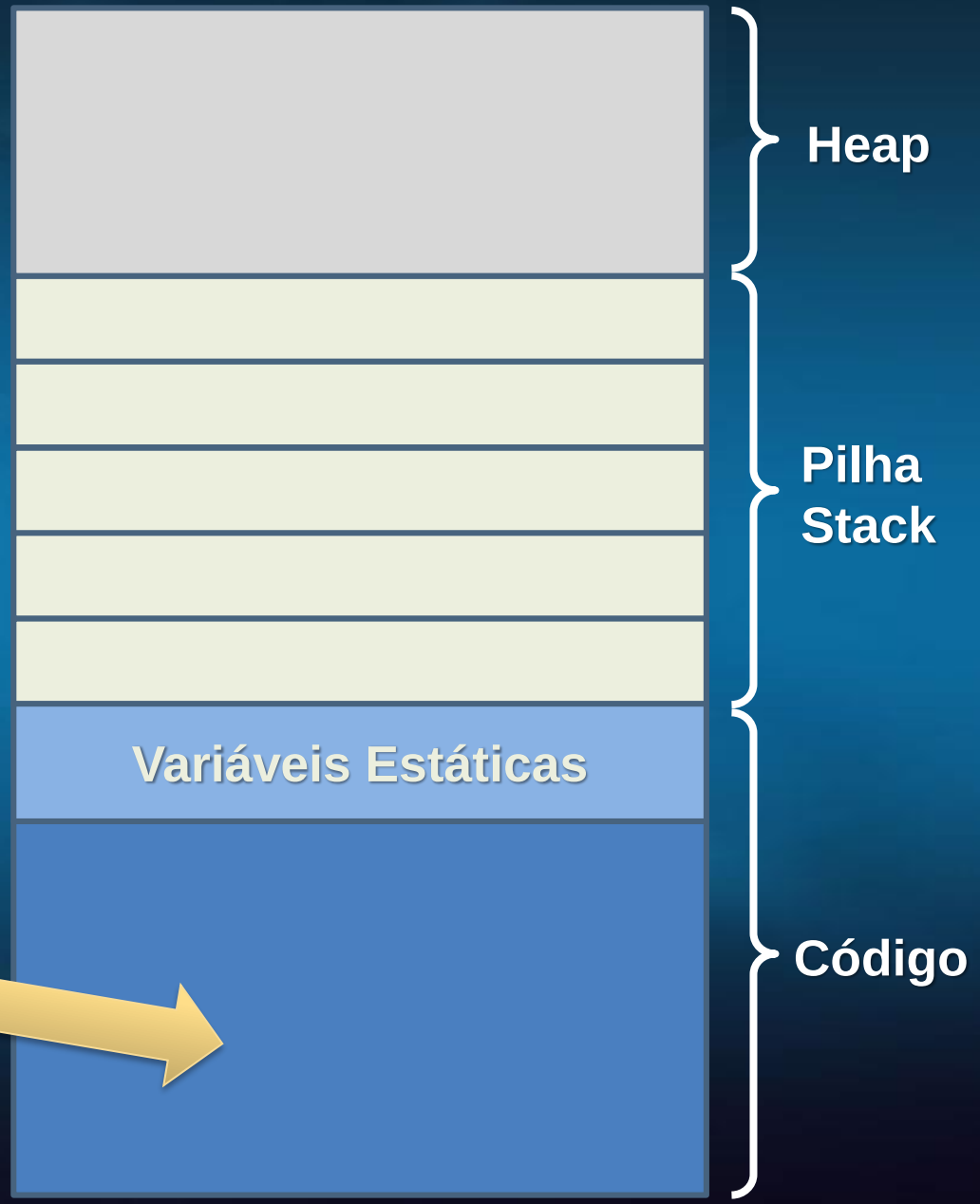
Saída
10



Chamada de função

```
void exibir(int vlr);

int main() {
    int x;

    x = 9;
    x++;

    exibir(x);

    system("pause");
    return 0;
}

void exibir(int vlr) {

    cout << "Valor de vlr: " << vlr << endl;
}
```

Saída
10



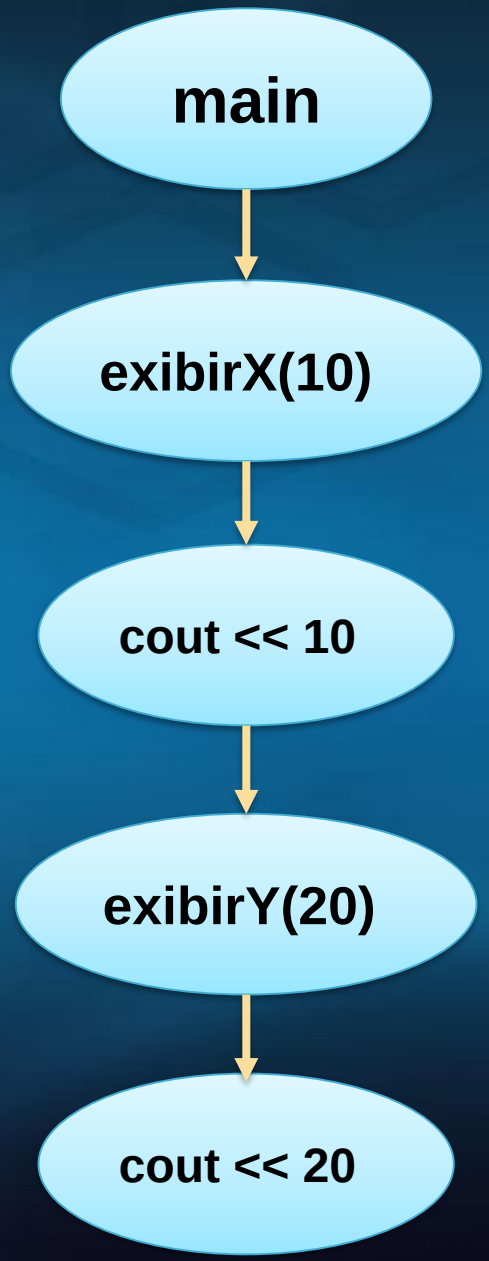
Funções Aninhadas

Recursividade

Funções aninhadas

```
void exibirX(int vlrX);
void exibirY(int vlrY);

int main() {
    int x = 10;

    exibirX(x);

    system("pause");
    return 0;
}

void exibirX(int vlrX) {
    int y;

    cout << "Valor de vlrX: " << vlrX << endl;

    y = vlrX * 2;

    exibirY(y);
}

void exibirY(int vlrY) {

    cout << "Valor de vlrY: " << vlrY << endl;
}
```

Saída
10
20



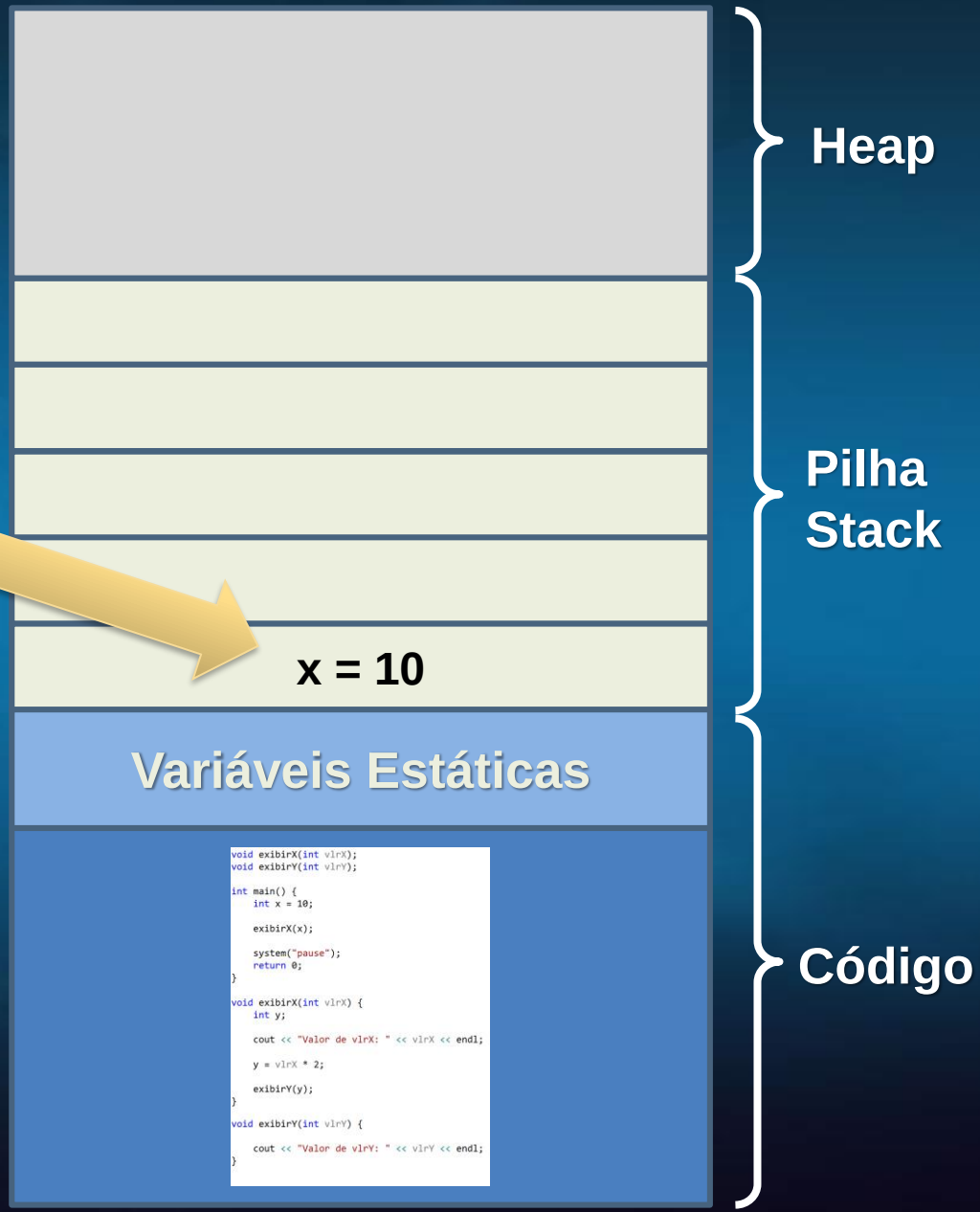
Funções aninhadas

```
void exibirX(int vlrX);
void exibirY(int vlrY);

int main() {
    int x = 10;

    exibirX(x);

    system("pause");
    return 0;
}

void exibirX(int vlrX) {
    int y;

    cout << "Valor de vlrX: " << vlrX << endl;

    y = vlrX * 2;

    exibirY(y);
}

void exibirY(int vlrY) {

    cout << "Valor de vlrY: " << vlrY << endl;
}
```

LOAD

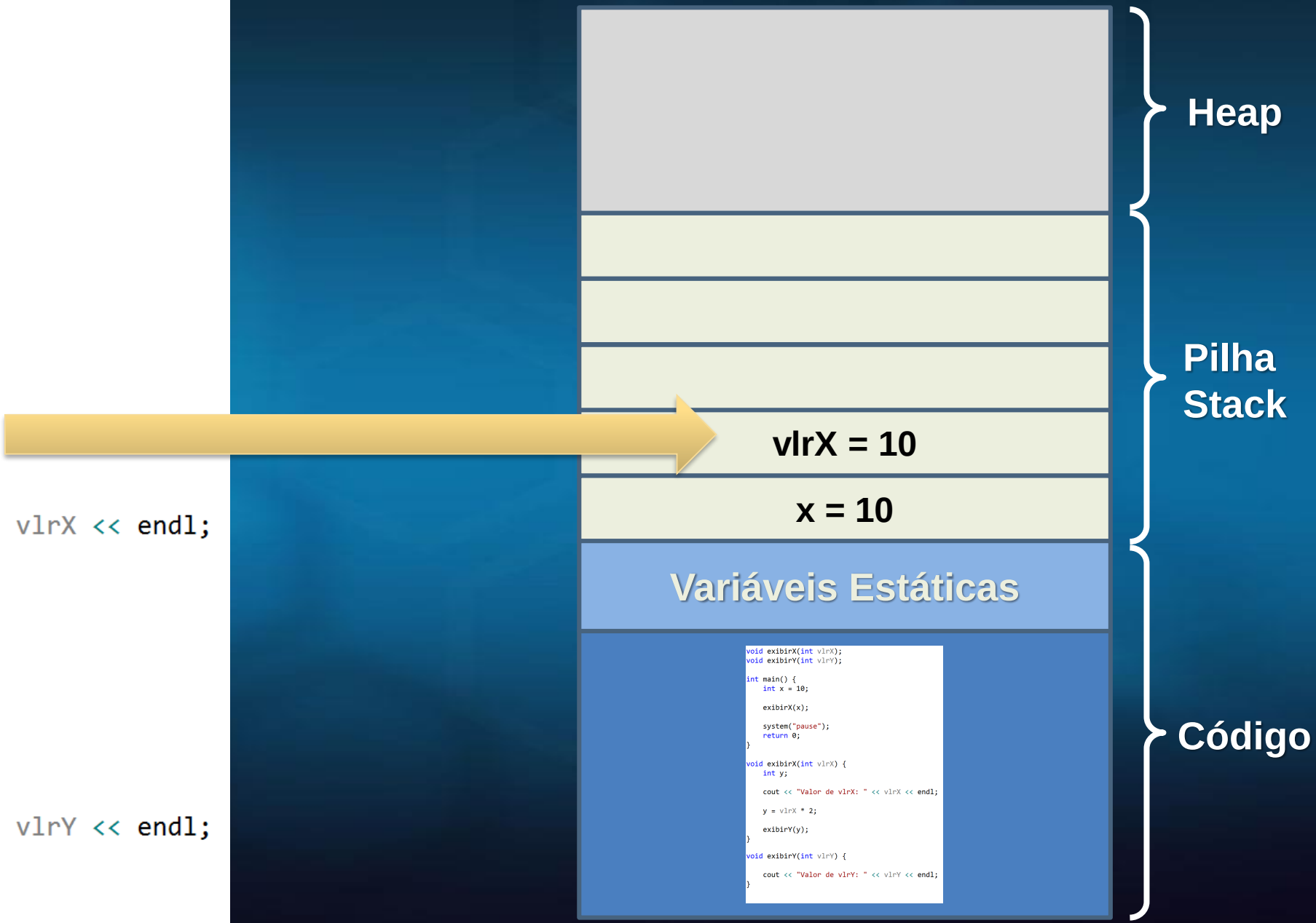
Funções aninhadas

```
void exibirX(int vlrX);
void exibirY(int vlrY);

int main() {
    int x = 10;

    exibirX(x);

    system("pause");
    return 0;
}

void exibirX(int vlrX) {
    int y;

    cout << "Valor de vlrX: " << vlrX << endl;

    y = vlrX * 2;

    exibirY(y);
}

void exibirY(int vlrY) {

    cout << "Valor de vlrY: " << vlrY << endl;
}
```



Funções aninhadas

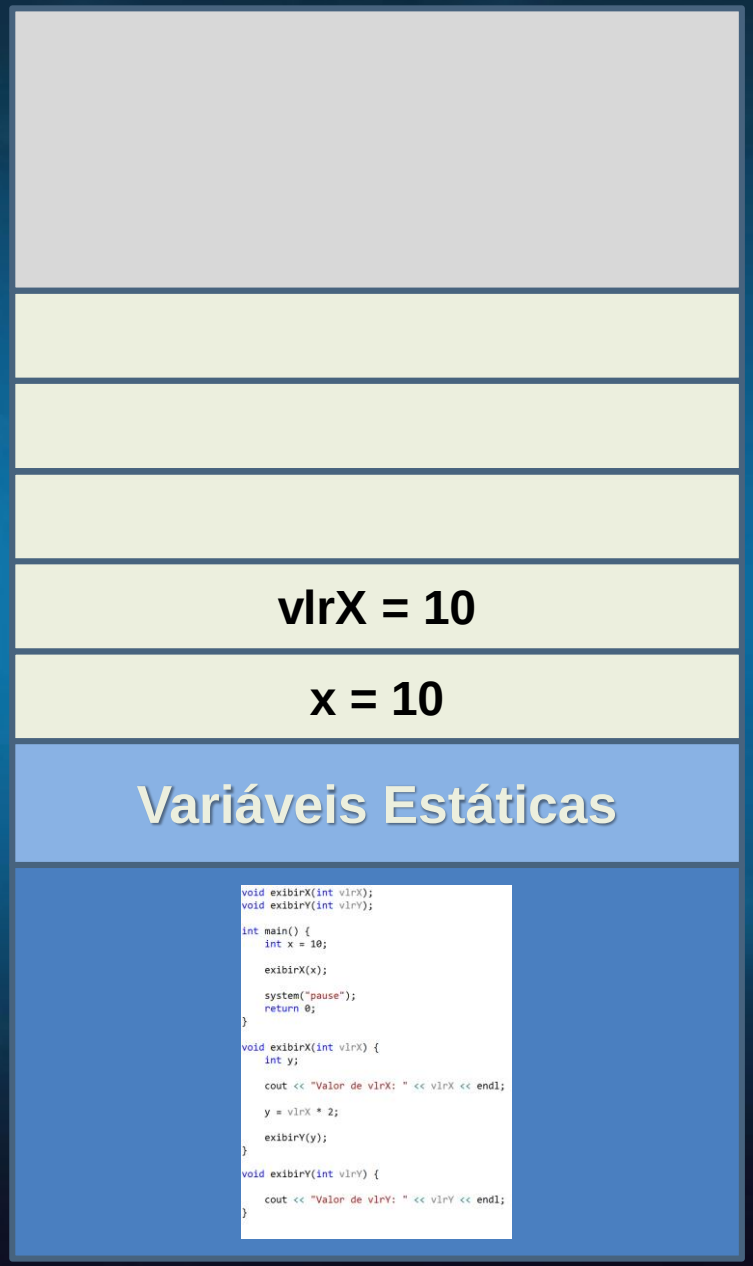
```
void exibirX(int vlrX);
void exibirY(int vlrY);

int main() {
    int x = 10;

    exibirX(x);

    system("pause");
    return 0;
}

void exibirX(int vlrX) {
    int y;

    cout << "Valor de vlrX: " << vlrX << endl;

    y = vlrX * 2;

    exibirY(y);
}

void exibirY(int vlrY) {

    cout << "Valor de vlrY: " << vlrY << endl;
}
```



Funções aninhadas

```
void exibirX(int vlrX);
void exibirY(int vlrY);

int main() {
    int x = 10;

    exibirX(x);

    system("pause");
    return 0;
}

void exibirX(int vlrX) {
    int y;

    cout << "Valor de vlrX: " << vlrX << endl;

    y = vlrX * 2;

    exibirY(y);
}

void exibirY(int vlrY) {

    cout << "Valor de vlrY: " << vlrY << endl;
}
```

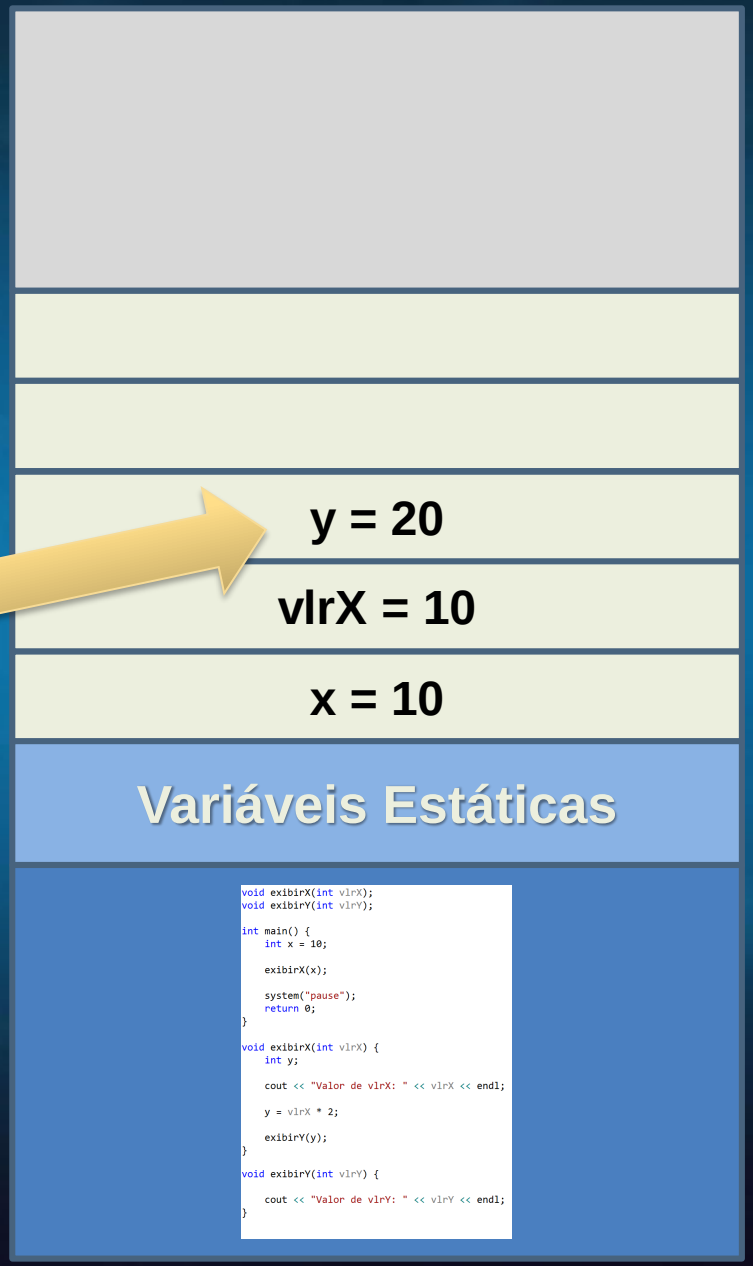
Saída
10



Funções aninhadas

```
void exibirX(int vlrX);
void exibirY(int vlrY);

int main() {
    int x = 10;

    exibirX(x);

    system("pause");
    return 0;
}

void exibirX(int vlrX) {
    int y;

    cout << "Valor de vlrX: " <<

    y = vlrX * 2;

    exibirY(y);
}

void exibirY(int vlrY) {

    cout << "Valor de vlrY: " << vlrY << endl;
}
```

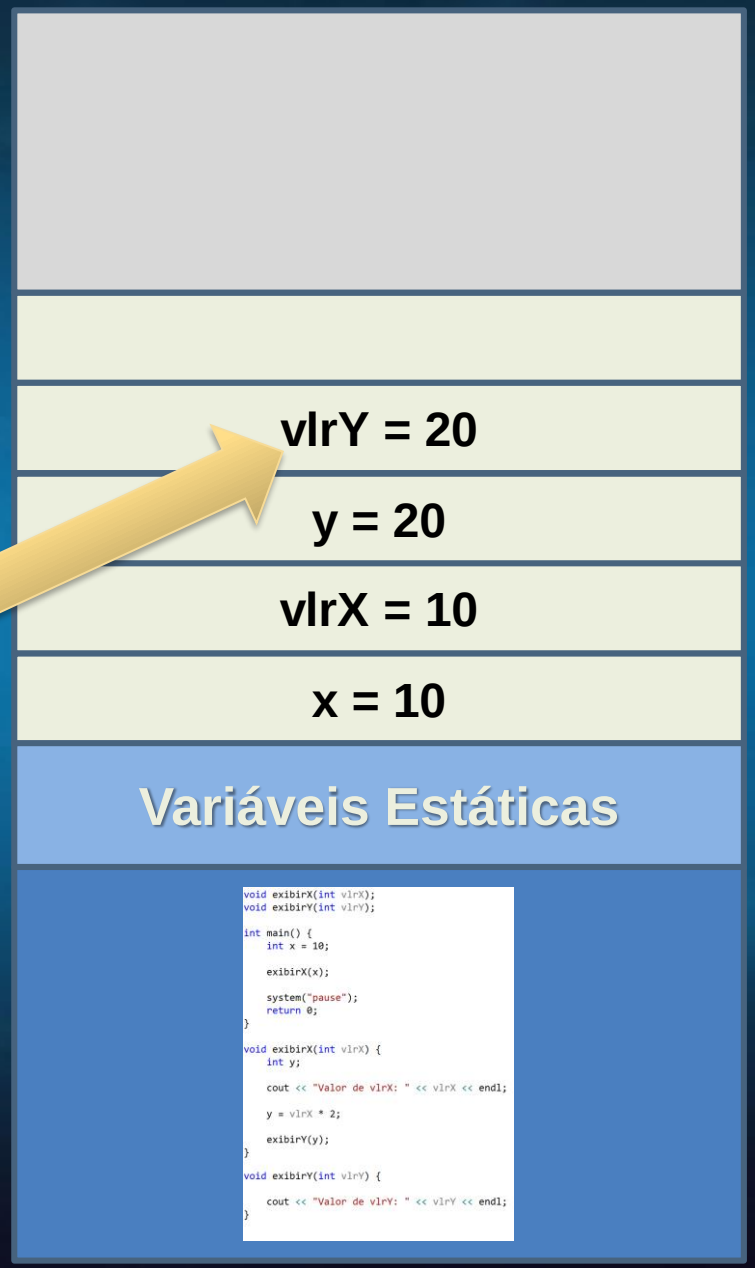
Saída
10



Funções aninhadas

```
void exibirX(int vlrX);
void exibirY(int vlrY);

int main() {
    int x = 10;

    exibirX(x);

    system("pause");
    return 0;
}

void exibirX(int vlrX) {
    int y;

    cout << "Valor de vlrX: " << vlrX << endl;

    y = vlrX * 2;

    exibirY(y);
}

void exibirY(int vlrY) {

    cout << "Valor de vlrY: " << vlrY << endl;
}
```

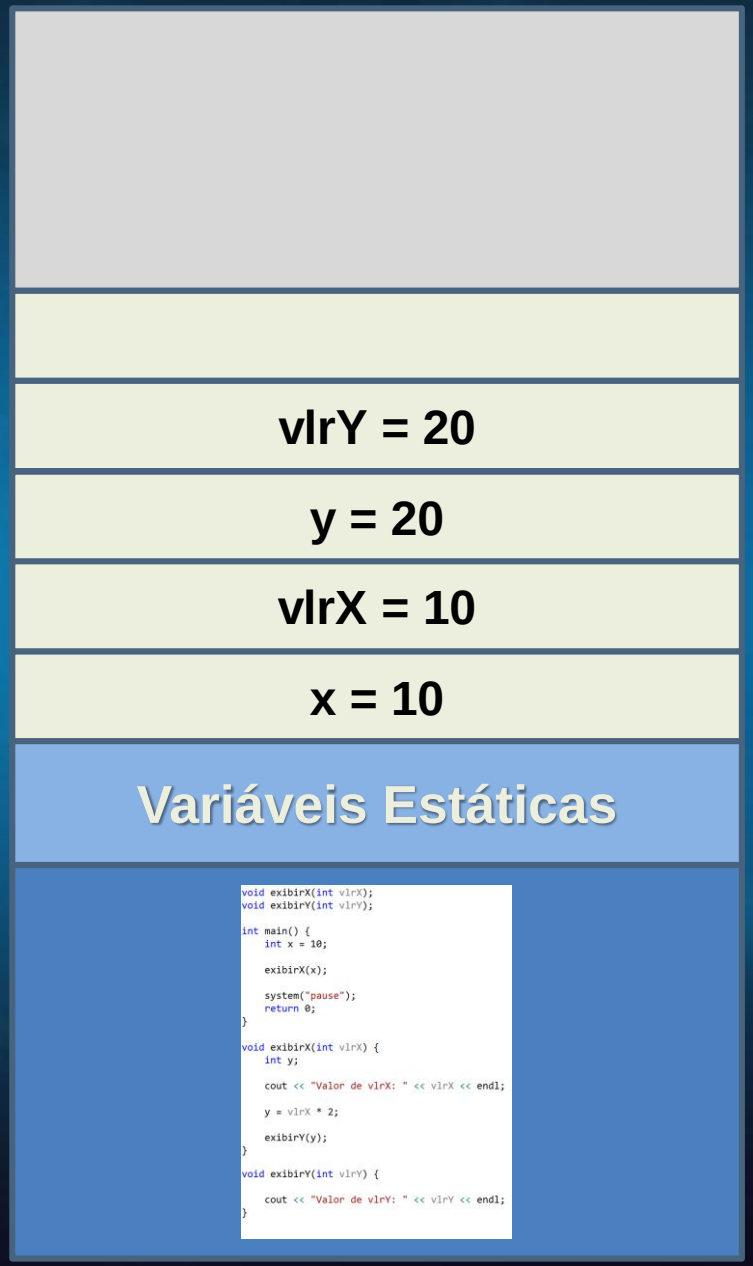
Saída
10



Funções aninhadas

```
void exibirX(int vlrX);
void exibirY(int vlrY);

int main() {
    int x = 10;

    exibirX(x);

    system("pause");
    return 0;
}

void exibirX(int vlrX) {
    int y;

    cout << "Valor de vlrX: " << vlrX << endl;

    y = vlrX * 2;

    exibirY(y);
}

void exibirY(int vlrY) {

    cout << "Valor de vlrY: " << vlrY << endl;
}
```

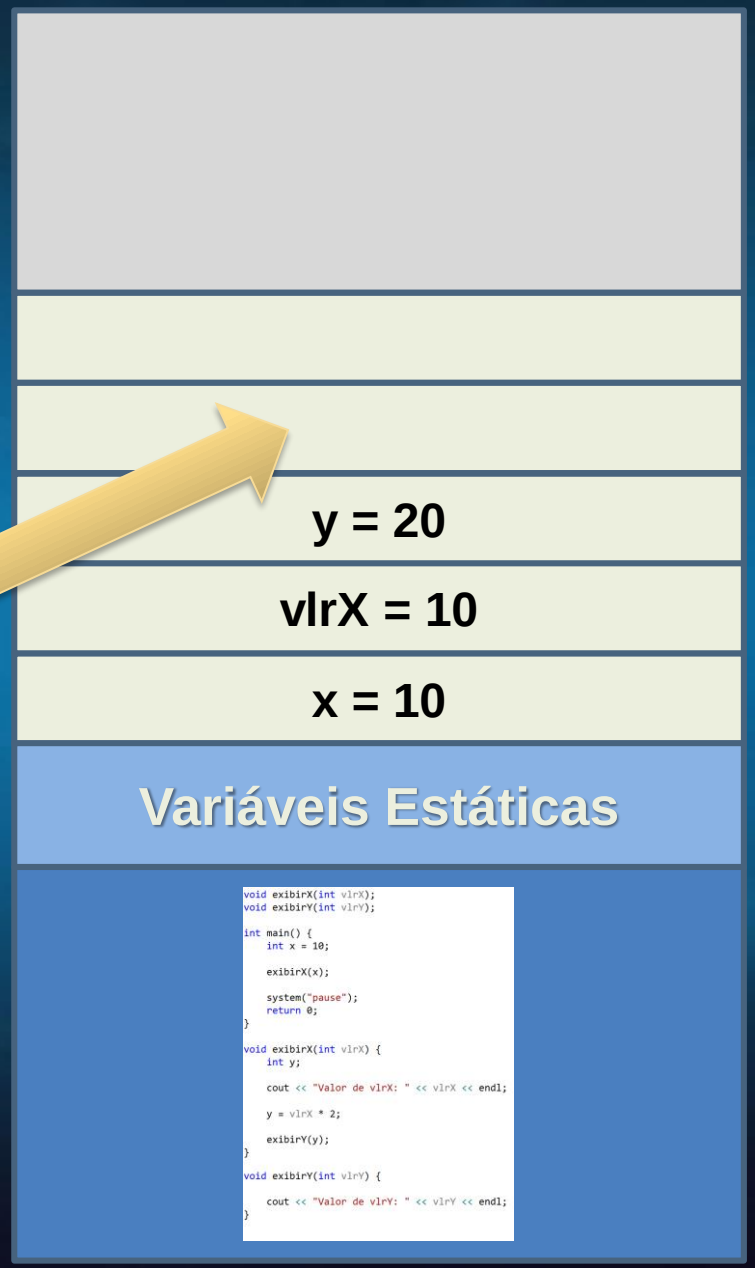
Saída
10
20



Funções aninhadas

```
void exibirX(int vlrX);
void exibirY(int vlrY);

int main() {
    int x = 10;

    exibirX(x);

    system("pause");
    return 0;
}

void exibirX(int vlrX) {
    int y;

    cout << "Valor de vlrX: " << vlrX << endl;

    y = vlrX * 2;

    exibirY(y);
}

void exibirY(int
                {

    cou      valor de vlrY: " << vlrY << endl;
}
```

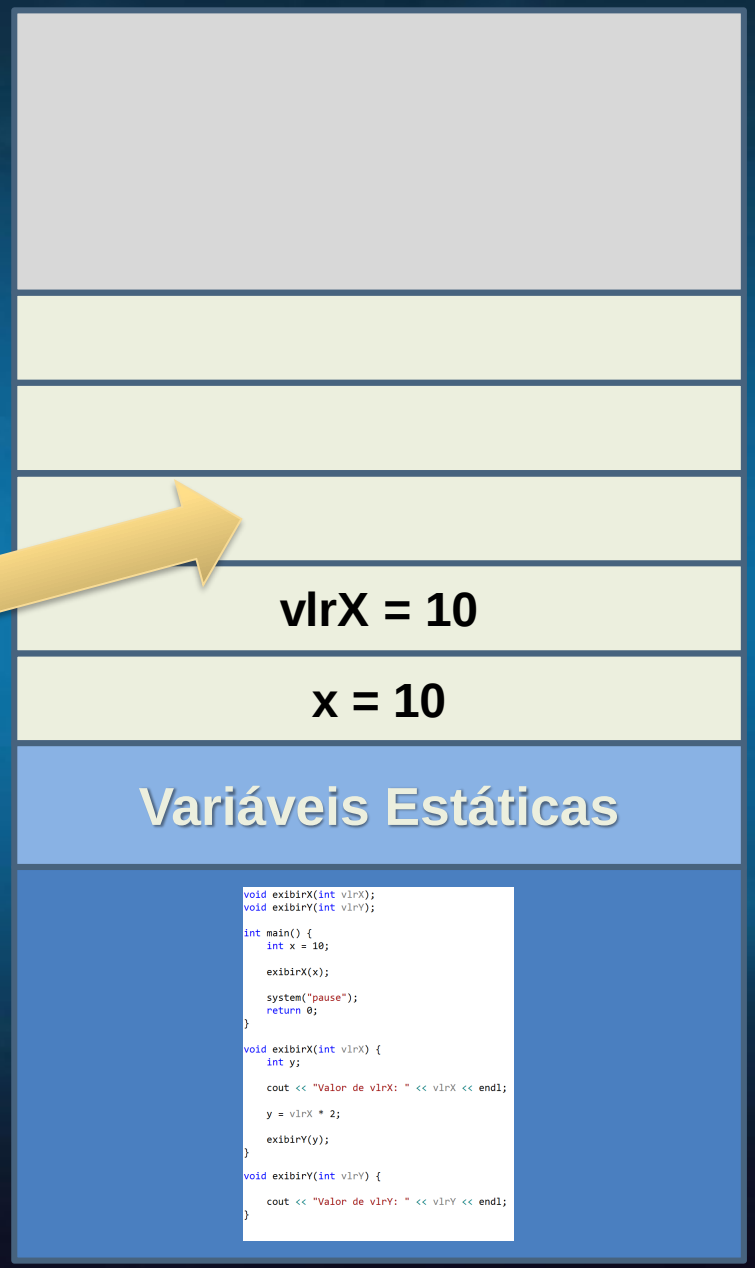
Saída
10
20



Funções aninhadas

```
void exibirX(int vlrX);
void exibirY(int vlrY);

int main() {
    int x = 10;

    exibirX(x);

    system("pause");
    return 0;
}

void exibirX(int vlrX) {
    int y;

    cout << "Valor de vlrX: " << vlrX <<

    y = vlrX * 2;

    exi
}
void exibirY(int vlrY) {

    cout << "Valor de vlrY: " << vlrY << endl;
}
```

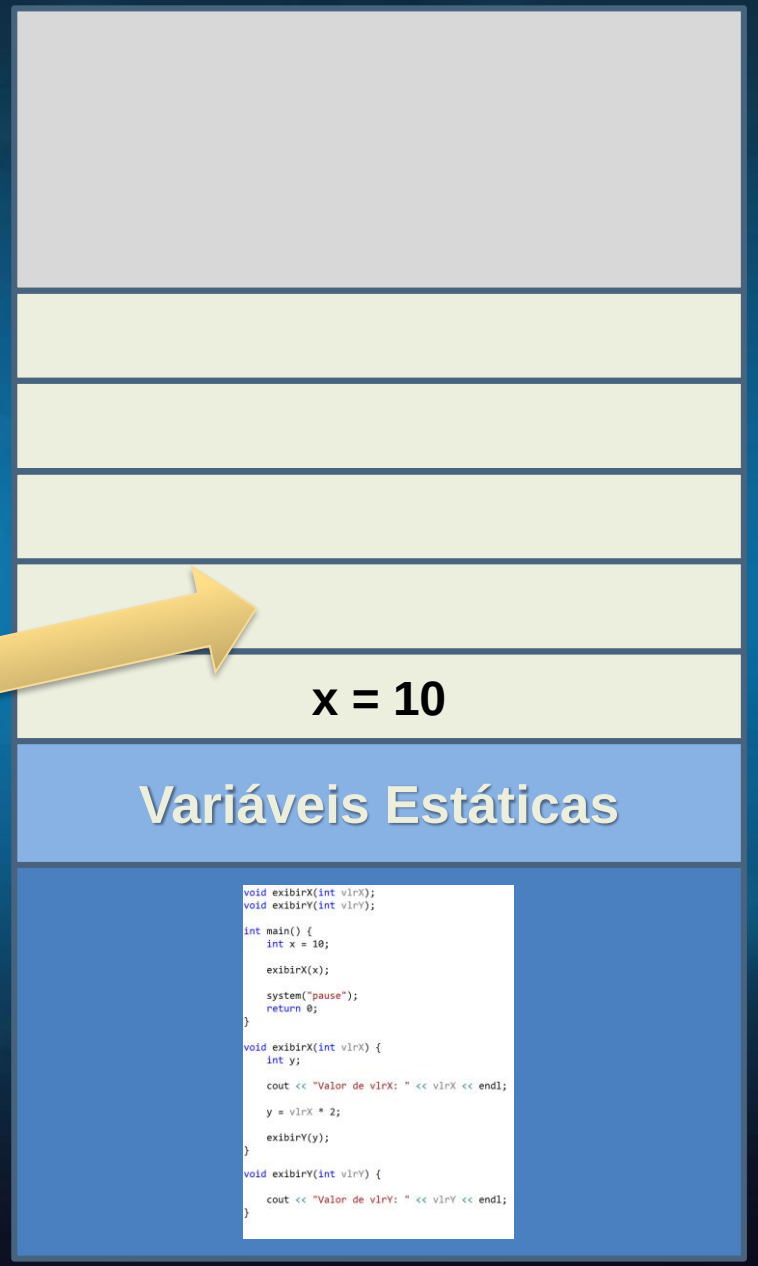
Saída
10
20



Funções aninhadas

```
void exibirX(int vlrX);
void exibirY(int vlrY);

int main() {
    int x = 10;

    exibirX(x);

    system("pause");
    return 0;
}

void exibirX(int vlrX) {
    int y;

    cout << "Valor de vlrX: " << vlrX << endl;

    y = vlrX * 2;

    exibirY(
}
void exibirY(int vlrY) {

    cout << "Valor de vlrY: " << vlrY << endl;
}
```

Saída
10
20



Funções aninhadas

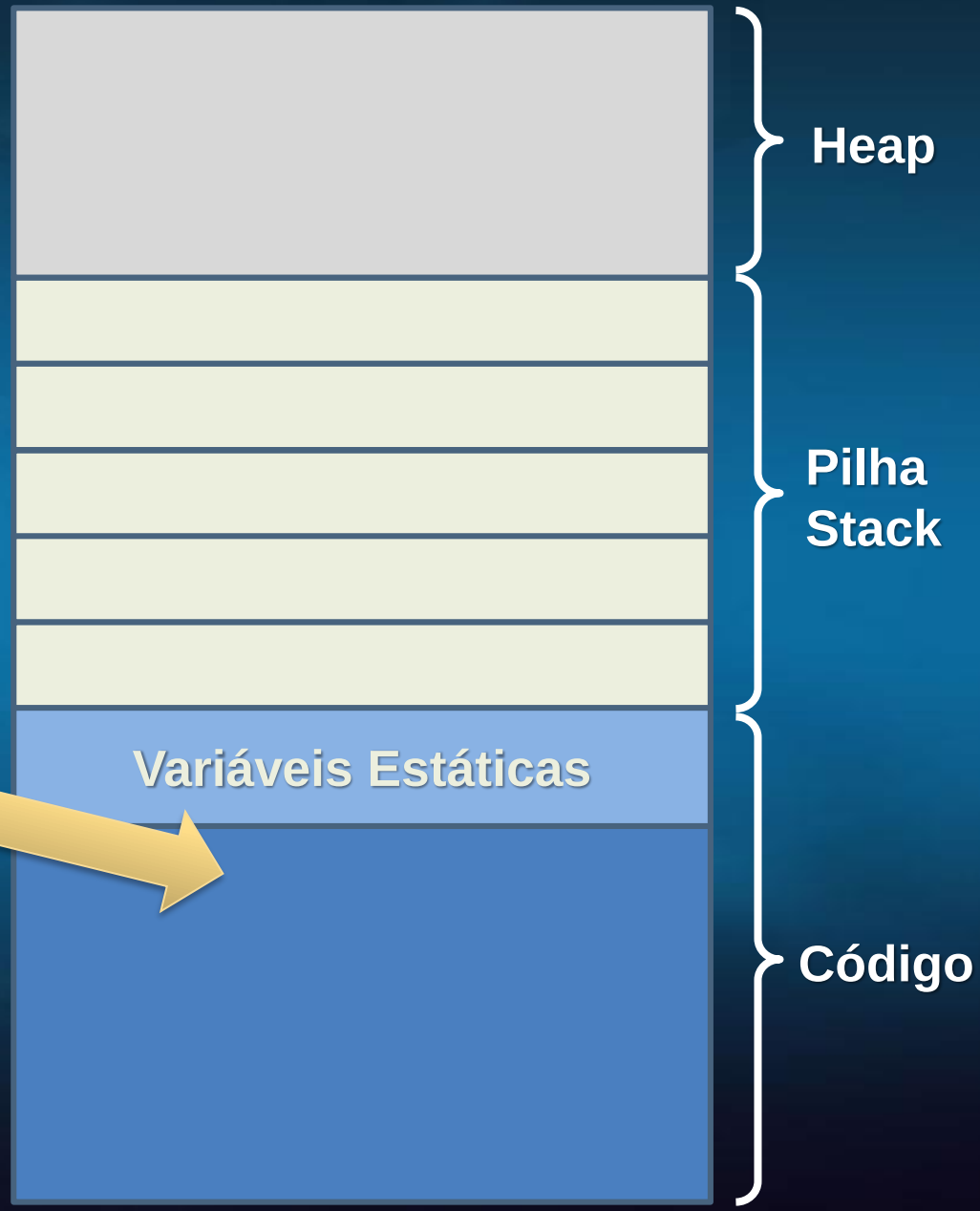
```
void exibirX(int vlrX);
void exibirY(int vlrY);

int main() {
    int x = 10;

    exibirX(x);

    system("pause");
    return 0;
}

void exibirX(int vlrX) {
    int y;

    cout << "Valor de vlrX: " << vlrX << endl;

    y = vlrX * 2;

    exibirY(y);
}

void exibirY(int vlrY) {

    cout << "Valor de vlrY: " << vlrY << endl;
}
```

Saída
10
20

Funções aninhadas

```
void exibirX(int vlrX);
void exibirY(int vlrY);

int main() {
    int x = 10;

    exibirX(x);

    system("pause");
    return 0;
}

void exibi
    int y;

    cout << "Valor de vlrX: " << vlrX

    y = vlrX * 2;

    exibirY(y);
}

void exibirY(int vlrY) {

    cout << "Valor de vlrY: " << vlrY << endl;
}
```

Saída
10
20



Recursividade – Versão 1

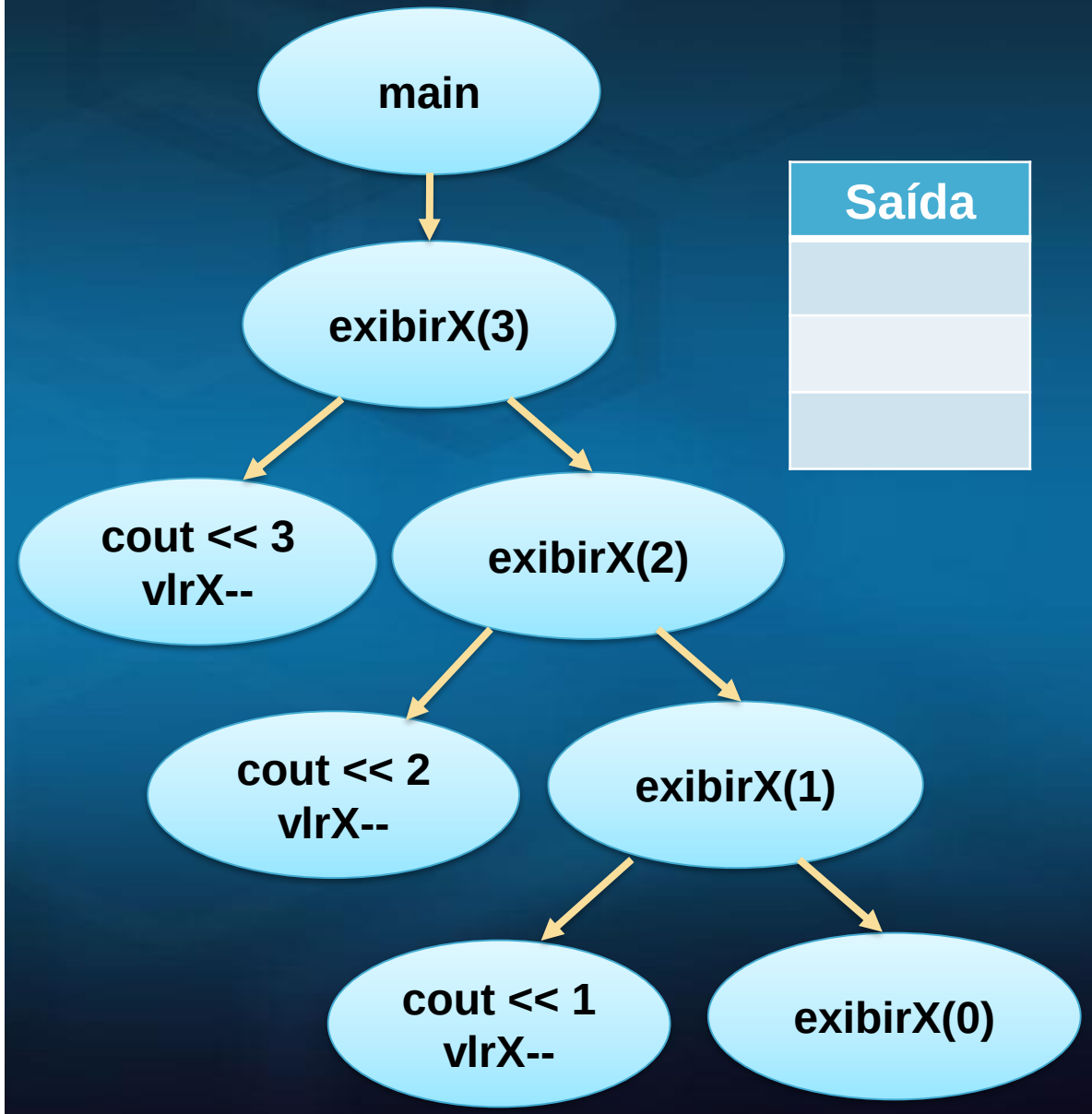
Recursividade

Função Recursiva

- Função que chama ela mesma
- Precisa de um ponto de parada.

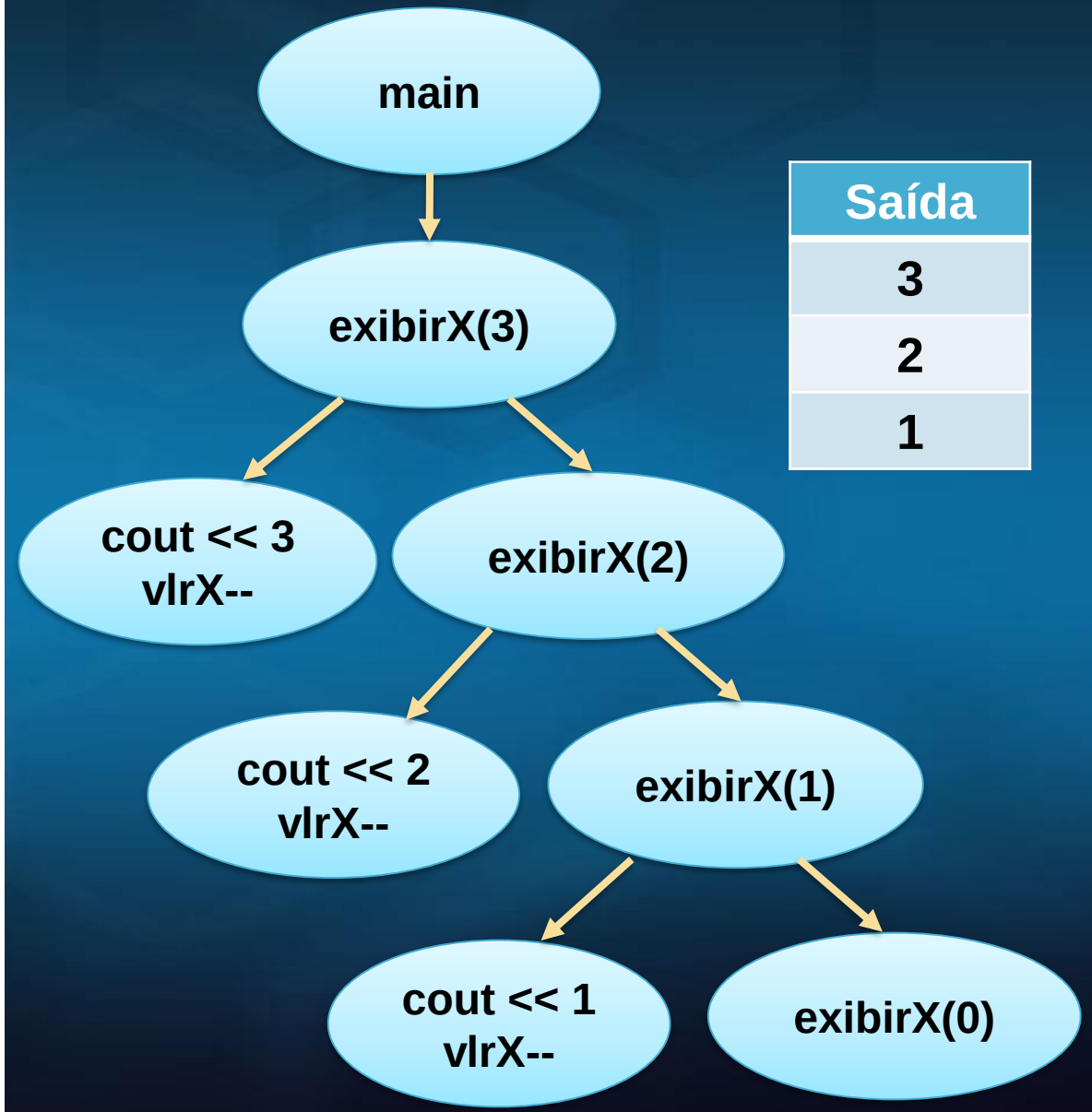
Recursividade

```
void exibirX(int vlrX);

int main() {
    int x = 3;

    exibirX(x);

    system("pause");
    return 0;
}

void exibirX(int vlrX) {

    if (vlrX > 0)
    {
        cout << "Valor de vlrX: " << vlrX << endl;

        vlrX--;

        exibirX(vlrX);
    }
}
```



Recursividade

```
void exibirX(int vlrX);

int main() {
    int x = 3;

    exibirX(x);

    system("pause");
    return 0;
}

void exibirX(int vlrX) {

    if (vlrX > 0)
    {
        cout << "Valor de vlrX: " << vlrX << endl;

        vlrX--;

        exibirX(vlrX);
    }
}
```



Saída
3
2
1

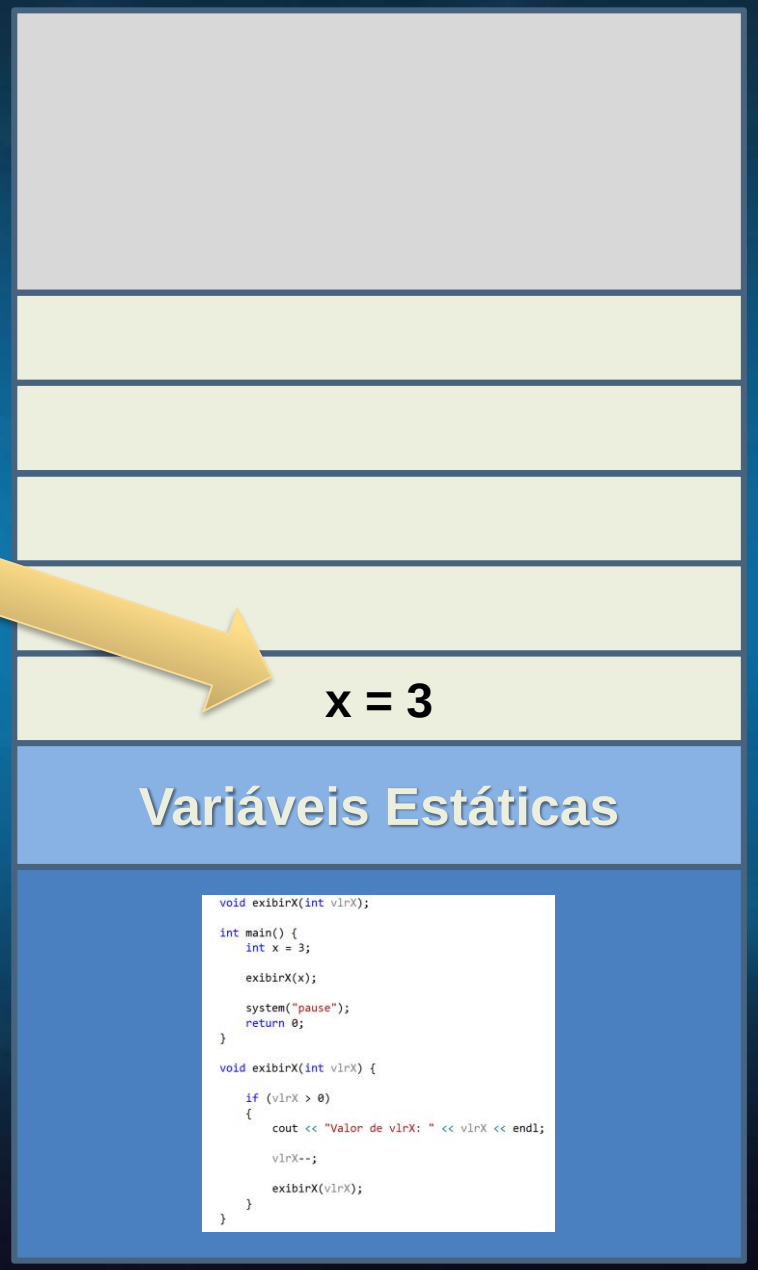
Funções aninhadas

```
void exibirX(int vlrX);

int main() {
    int x = 3;

    exibirX(x);

    system("pause");
    return 0;
}

void exibirX(int vlrX) {

    if (vlrX > 0)
    {
        cout << "Valor de vlrX: " << vlrX << endl;

        vlrX--;

        exibirX(vlrX);
    }
}
```

LOAD

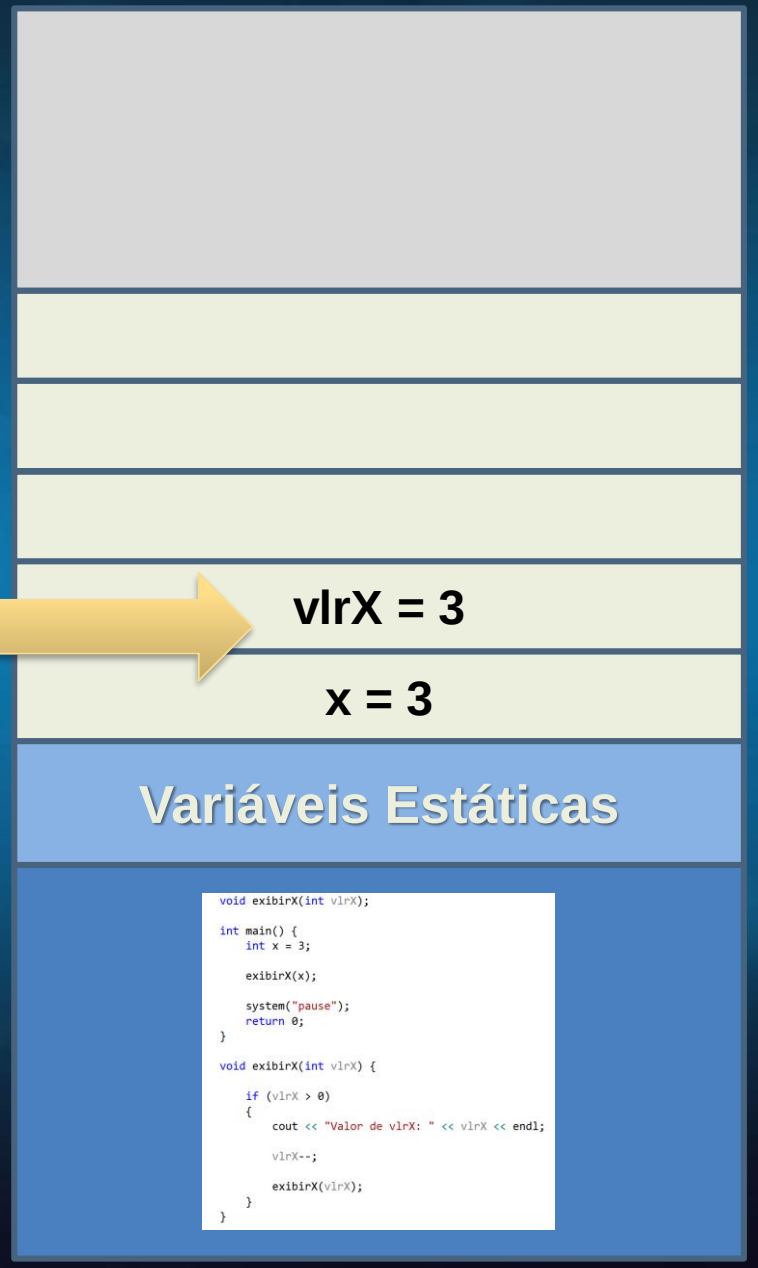
Funções aninhadas

```
void exibirX(int vlrX);

int main() {
    int x = 3;

    exibirX(x);

    system("pause");
    return 0;
}

void exibirX(int vlrX) {

    if (vlrX > 0)
    {
        cout << "Valor de vlrX: " << vlrX << endl;

        vlrX--;

        exibirX(vlrX);
    }
}
```



Funções aninhadas

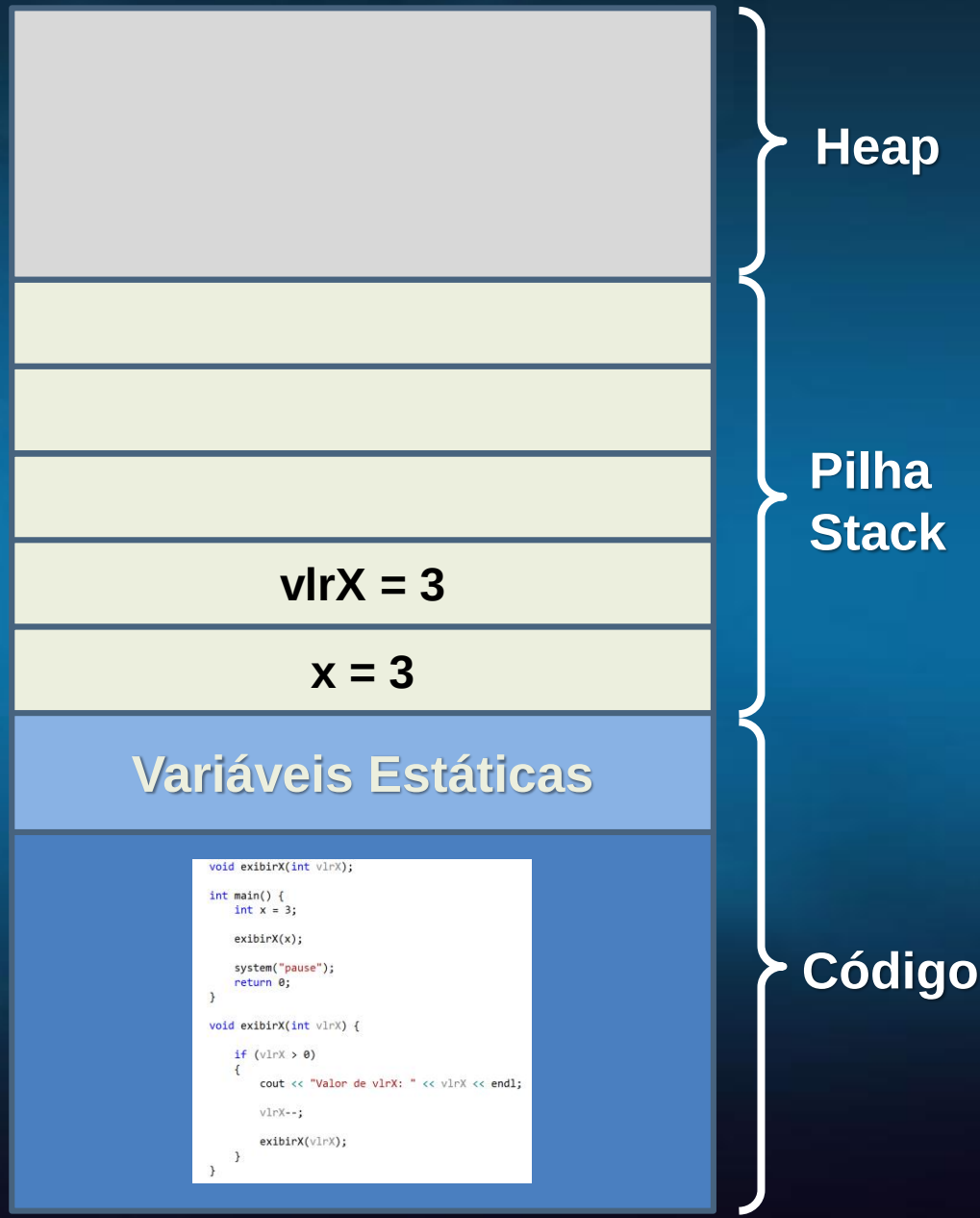
```
void exibirX(int vlrX);

int main() {
    int x = 3;

    exibirX(x);

    system("pause");
    return 0;
}

void exibirX(int vlrX) {

    if (vlrX > 0)
    {
        cout << "Valor de vlrX: " << vlrX << endl;

        vlrX--;

        exibirX(vlrX);
    }
}
```



Funções aninhadas

```
void exibirX(int vlrX);

int main() {
    int x = 3;

    exibirX(x);

    system("pause");
    return 0;
}

void exibirX(int vlrX) {

    if (vlrX > 0)
    {
        cout << "Valor de vlrX: " << vlrX << endl;

        vlrX--;

        exibirX(vlrX);
    }
}
```

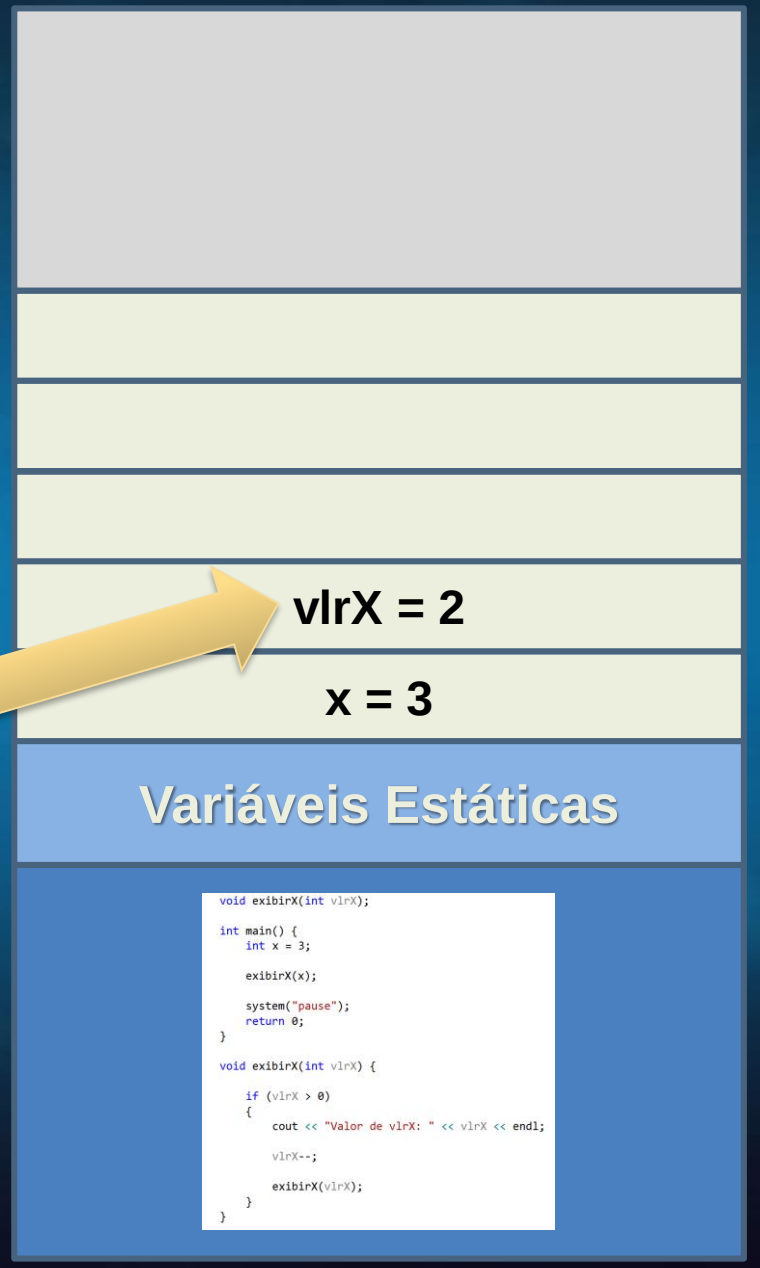
Saída
3



Funções aninhadas

```
void exibirX(int vlrX);

int main() {
    int x = 3;

    exibirX(x);

    system("pause");
    return 0;
}

void exibirX(int vlrX) {

    if (vlrX > 0)
    {
        cout << "Valor de v                << vlrX << endl;

        vlrX--;

        exibirX(vlrX);
    }
}
```

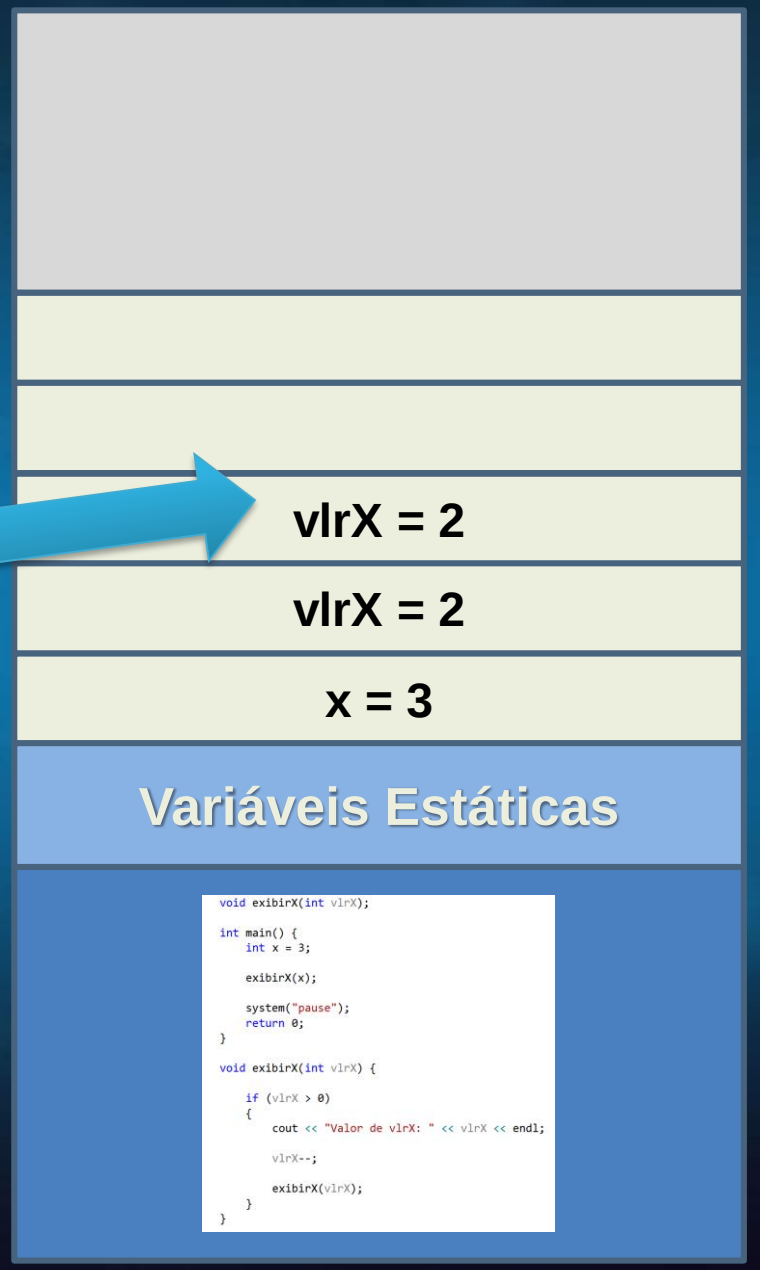
Saída
3



Funções aninhadas

```
void exibirX(int vlrX);

int main() {
    int x = 3;

    exibirX(x);

    system("pause");
    return 0;
}

void exibirX(int vlrX) {

    if (vlrX > 0)
    {
        cout << "Valor de vlrX: " << vlrX << endl;

        vlrX--;

        exibirX(vlrX);
    }
}
```

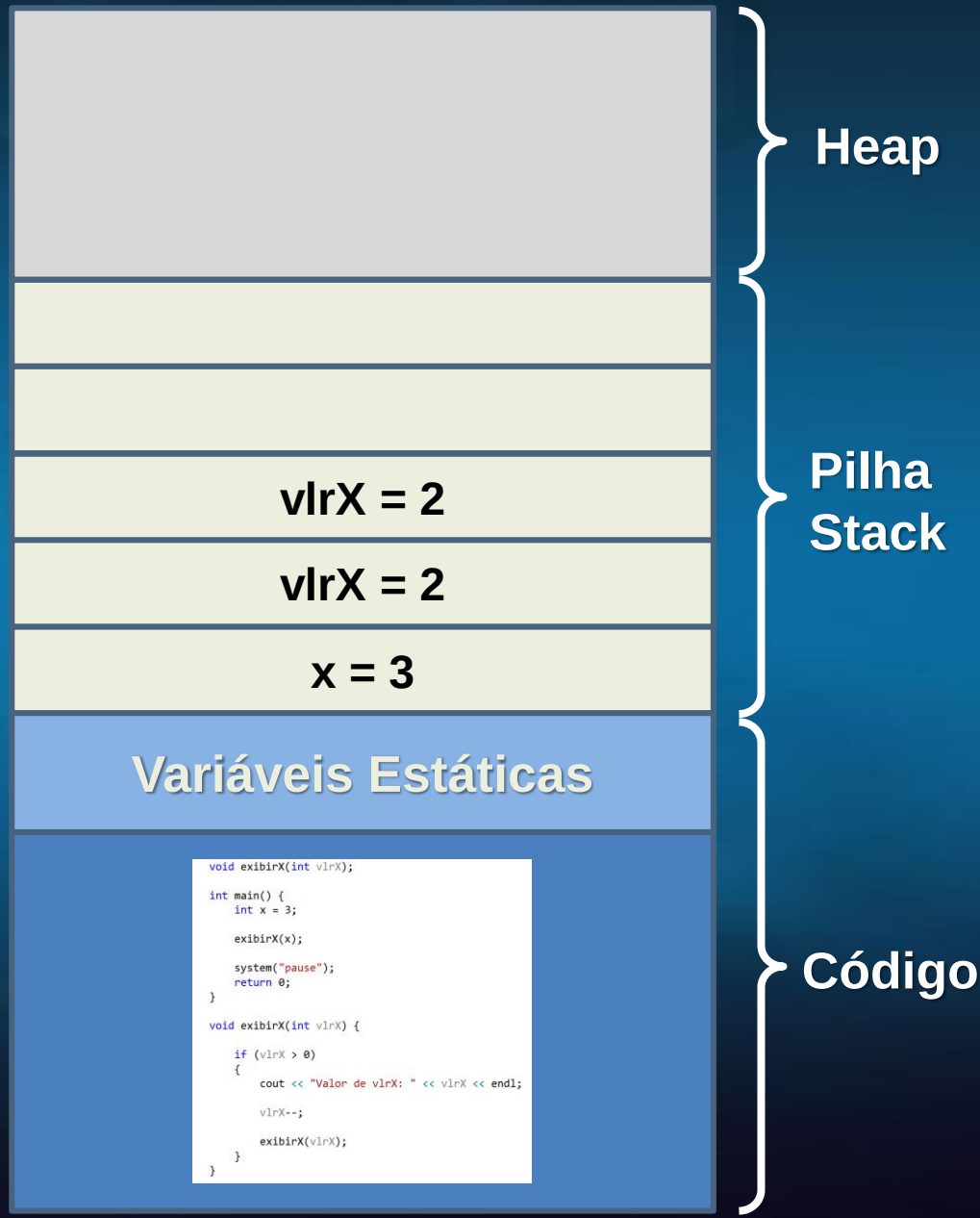
Saída
3



Funções aninhadas

```
void exibirX(int vlrX);

int main() {
    int x = 3;

    exibirX(x);

    system("pause");
    return 0;
}

void exibirX(int vlrX) {

    if (vlrX > 0)
    {
        cout << "Valor de vlrX: " << vlrX << endl;

        vlrX--;

        exibirX(vlrX);
    }
}
```

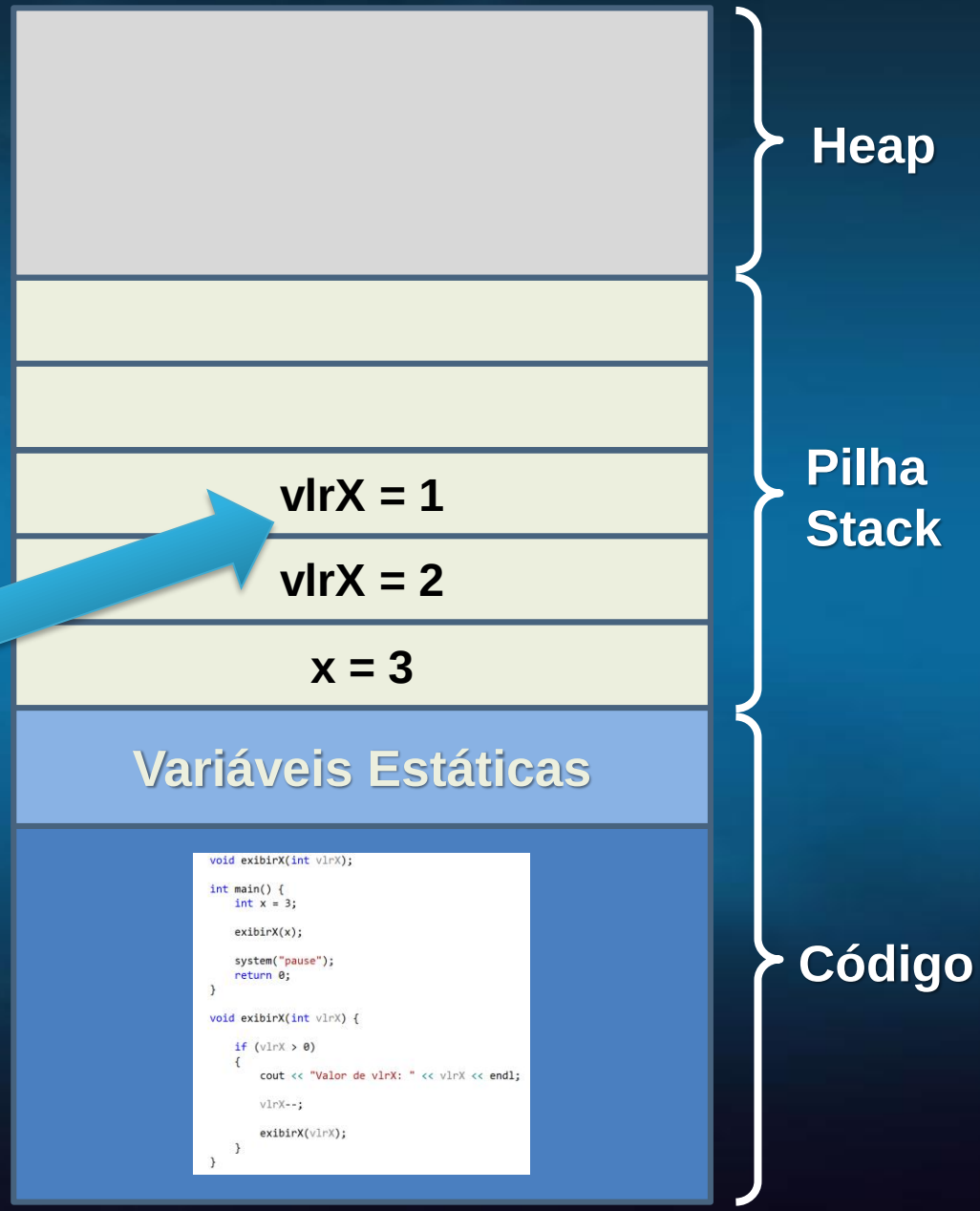
Saída
3
2



Funções aninhadas

```
void exibirX(int vlrX);

int main() {
    int x = 3;

    exibirX(x);

    system("pause");
    return 0;
}

void exibirX(int vlrX) {

    if (vlrX > 0)
    {
        cout << "Valor de                << vlrX << endl;

        vlrX--;

        exibirX(vlrX);
    }
}
```

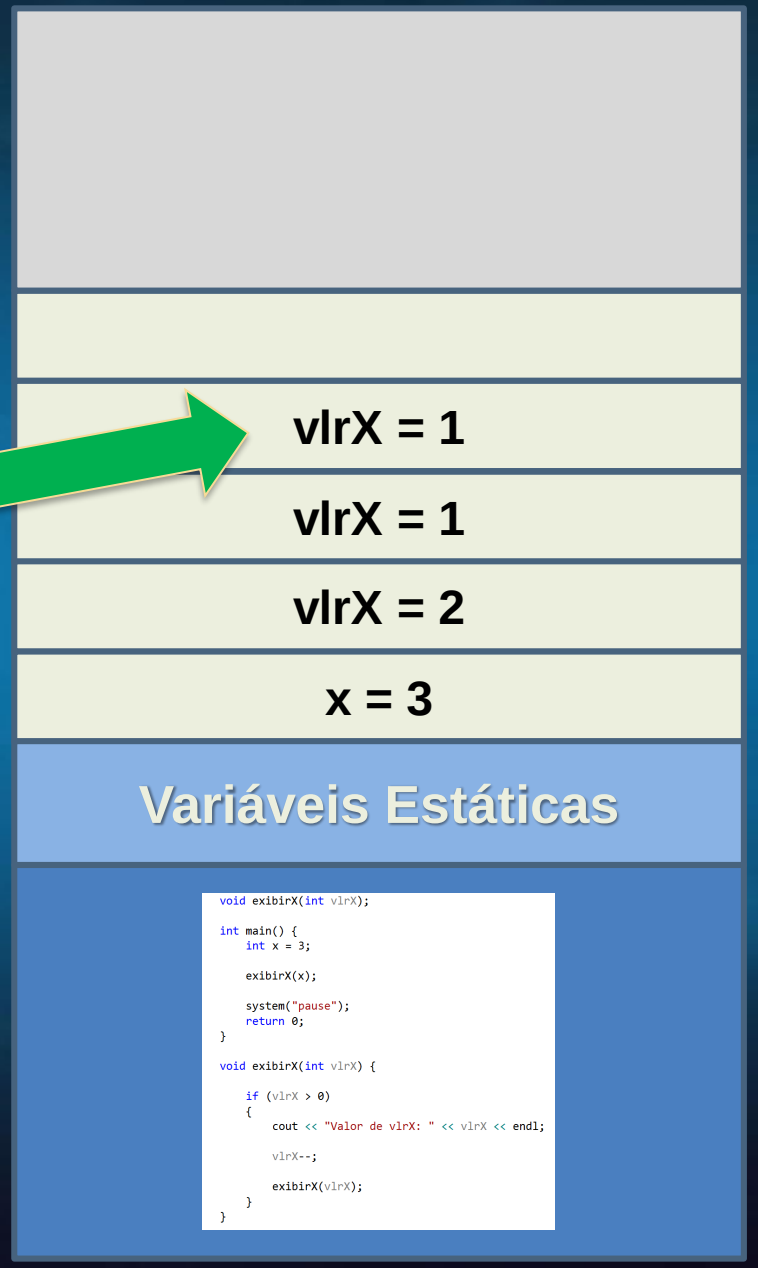
Saída
3
2



Funções aninhadas

```
void exibirX(int vlrX);

int main() {
    int x = 3;

    exibirX(x);

    system("pause");
    return 0;
}

void exibirX(int vlrX) {

    if (vlrX > 0)
    {
        cout << "Valor de vlrX: " << vlrX << endl;

        vlrX--;

        exibirX(vlrX);
    }
}
```

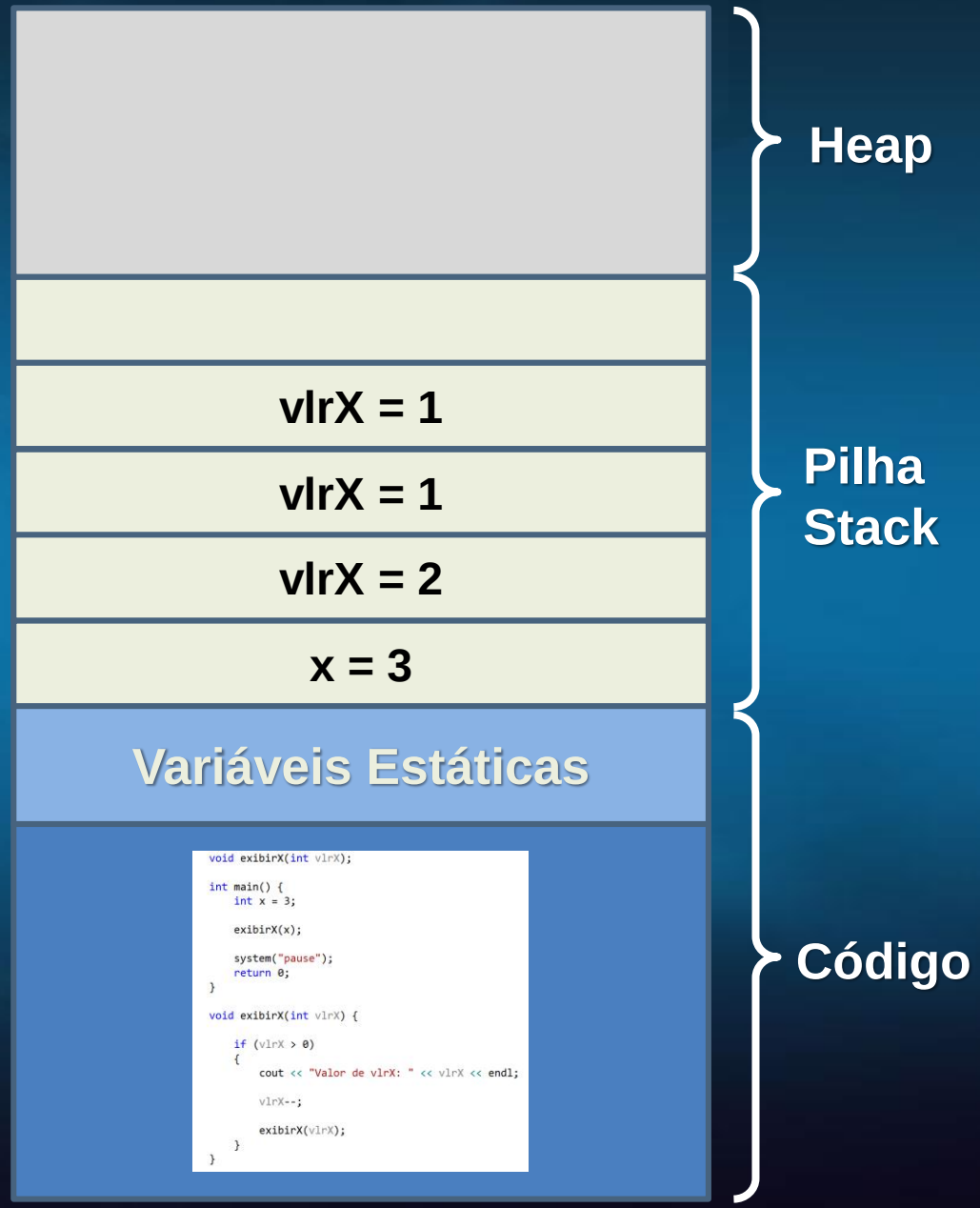
Saída
3
2



Funções aninhadas

```
void exibirX(int vlrX);

int main() {
    int x = 3;

    exibirX(x);

    system("pause");
    return 0;
}

void exibirX(int vlrX) {

    if (vlrX > 0)
    {
        cout << "Valor de vlrX: " << vlrX << endl;

        vlrX--;

        exibirX(vlrX);
    }
}
```

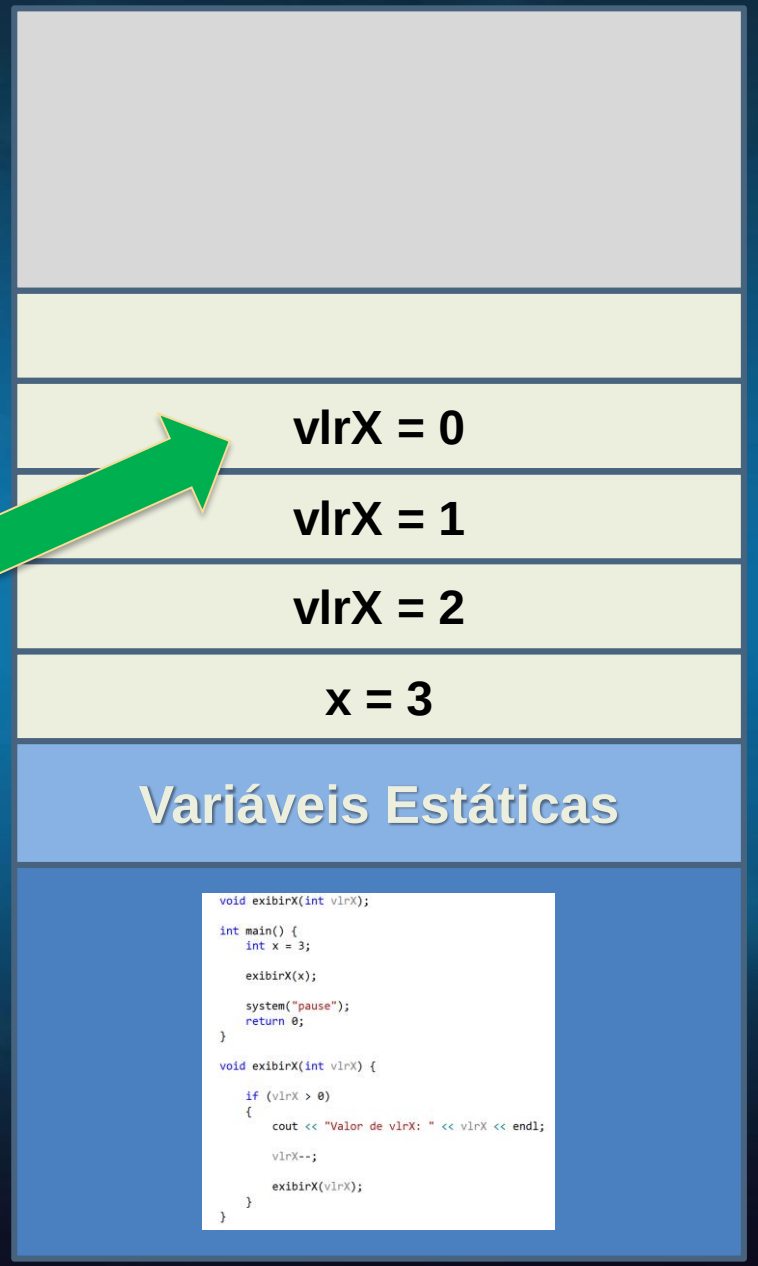
Saída
3
2
1



Funções aninhadas

```
void exibirX(int vlrX);

int main() {
    int x = 3;

    exibirX(x);

    system("pause");
    return 0;
}

void exibirX(int vlrX) {

    if (vlrX > 0)
    {
        cout << "Valor d    X: " << vlrX << endl;

        vlrX--;

        exibirX(vlrX);
    }
}
```

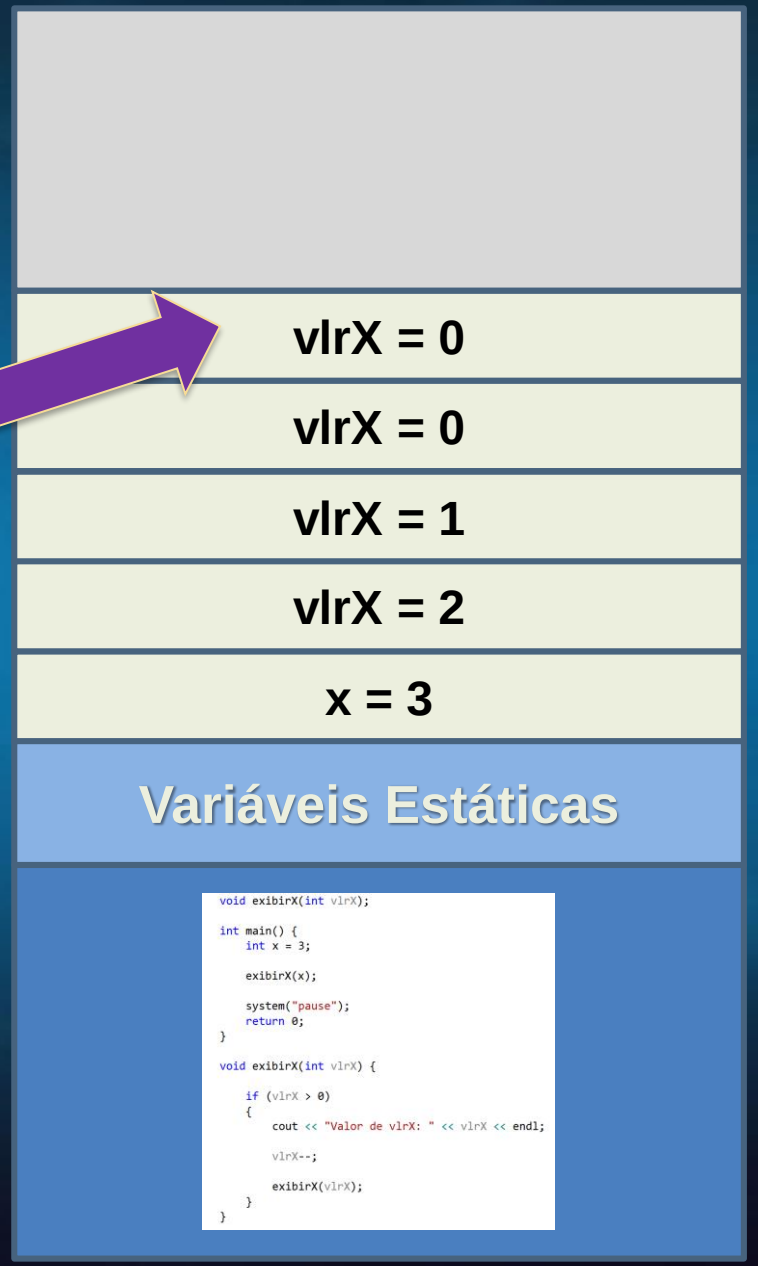
Saída
3
2
1



Funções aninhadas

```
void exibirX(int vlrX);

int main() {
    int x = 3;

    exibirX(x);

    system("pause");
    return 0;
}

void exibirX(int vlrX) {

    if (vlrX > 0)
    {
        cout << "Valor de vlrX: " << vlrX << endl;

        vlrX--;

        exibirX(vlrX);
    }
}
```

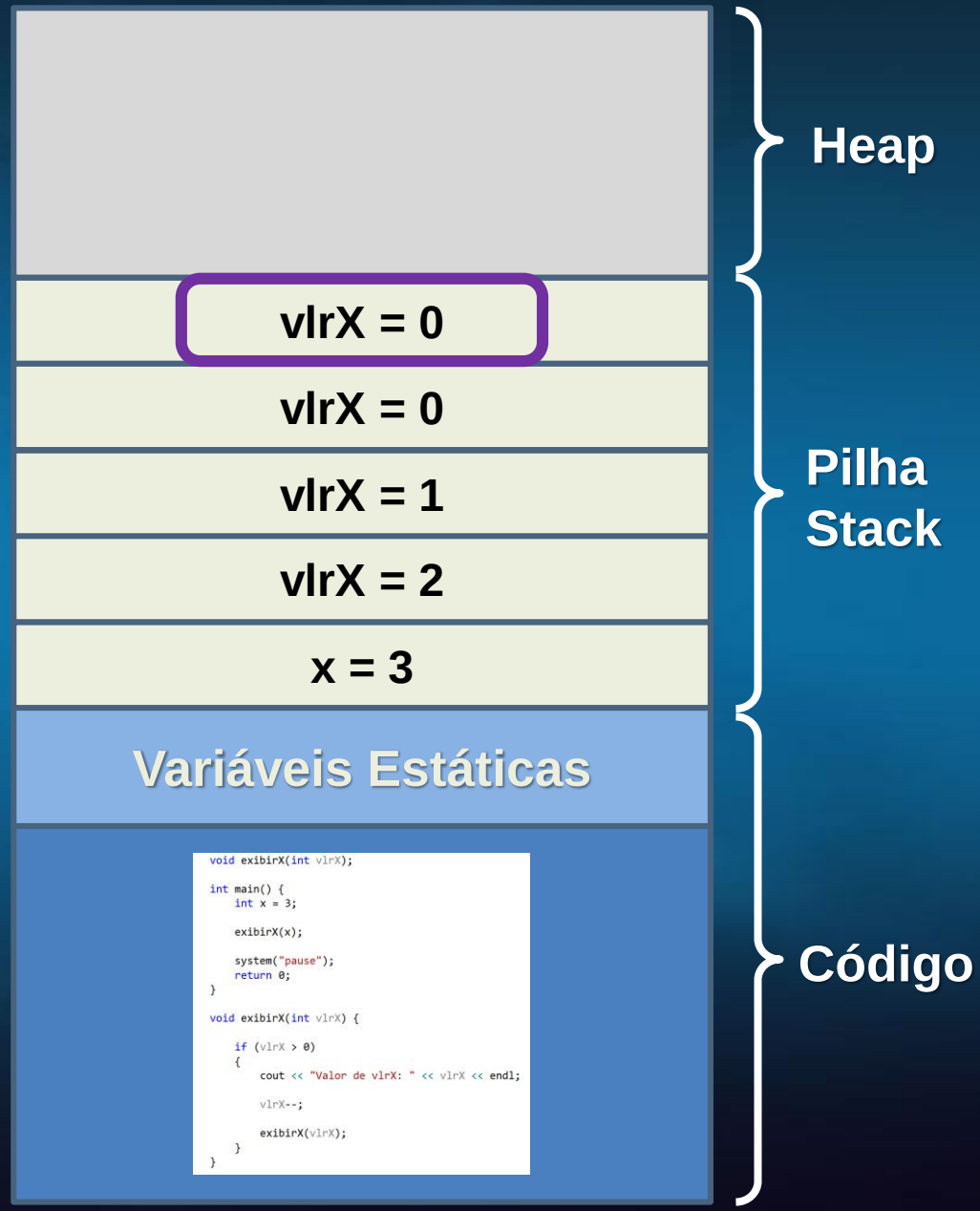
Saída
3
2
1



Funções aninhadas

```
void exibirX(int vlrX);

int main() {
    int x = 3;

    exibirX(x);

    system("pause");
    return 0;
}

void exibirX(int vlrX) {

    if (vlrX > 0)
    {
        cout << "Valor de vlrX: " << vlrX << endl;

        vlrX--;

        exibirX(vlrX);
    }
}
```

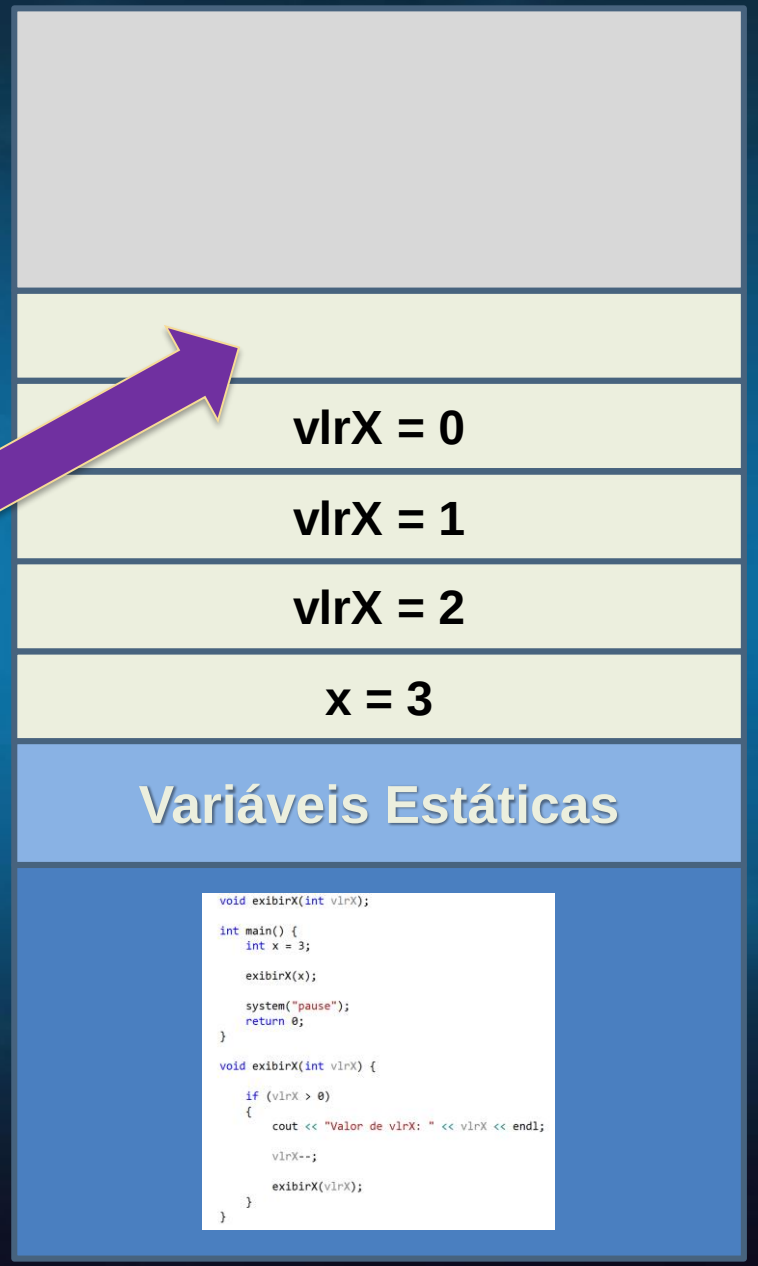
Saída
3
2
1



Funções aninhadas

```
void exibirX(int vlrX);

int main() {
    int x = 3;

    exibirX(x);

    system("pause");
    return 0;
}

void exibirX(int vlrX) {

    if (vlrX > 0)
    {
        cout << "Valor de vlrX: " << vlrX << endl;

        vlrX--;

        exibirX(vlrX);
    }
}
```

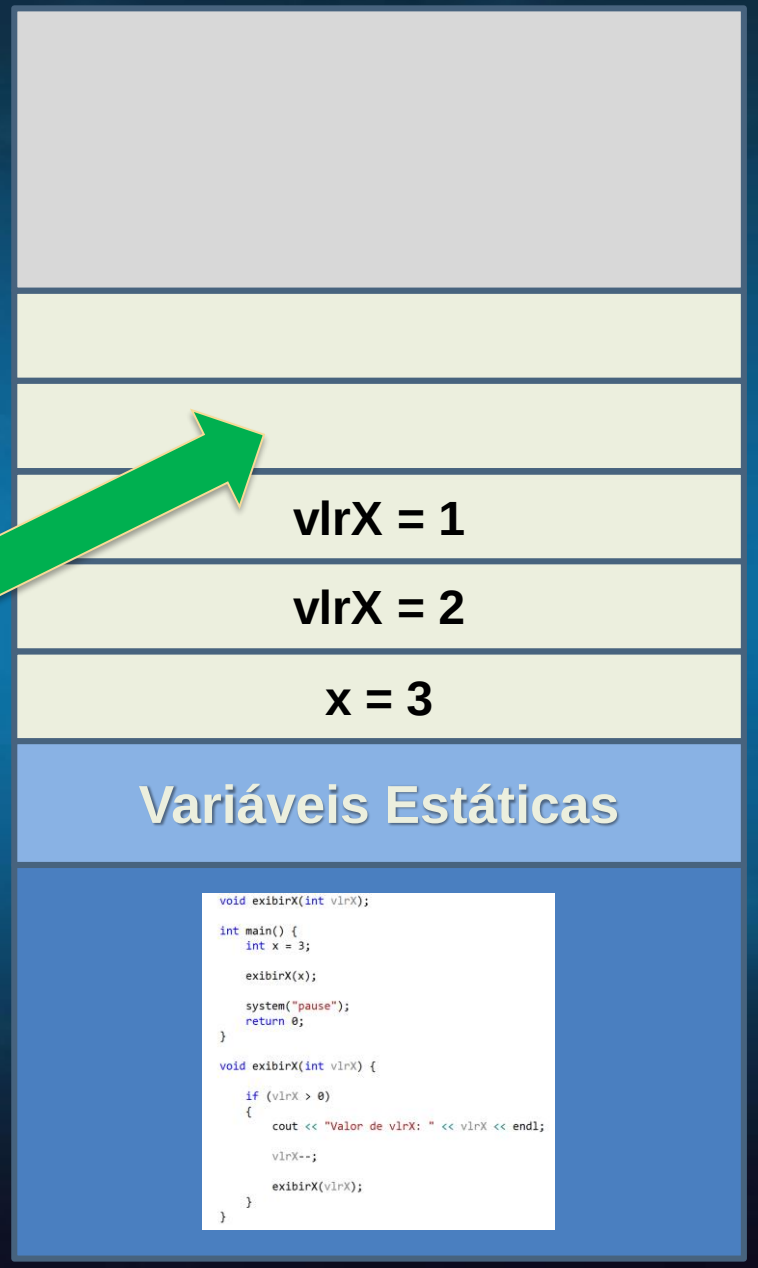
Saída
3
2
1



Funções aninhadas

```
void exibirX(int vlrX);

int main() {
    int x = 3;

    exibirX(x);

    system("pause");
    return 0;
}

void exibirX(int vlrX) {

    if (vlrX > 0)
    {
        cout << "Valor de vlrX: " << vlrX << endl;

        vlrX--;

        exibirX(vlrX);
    }
}
```

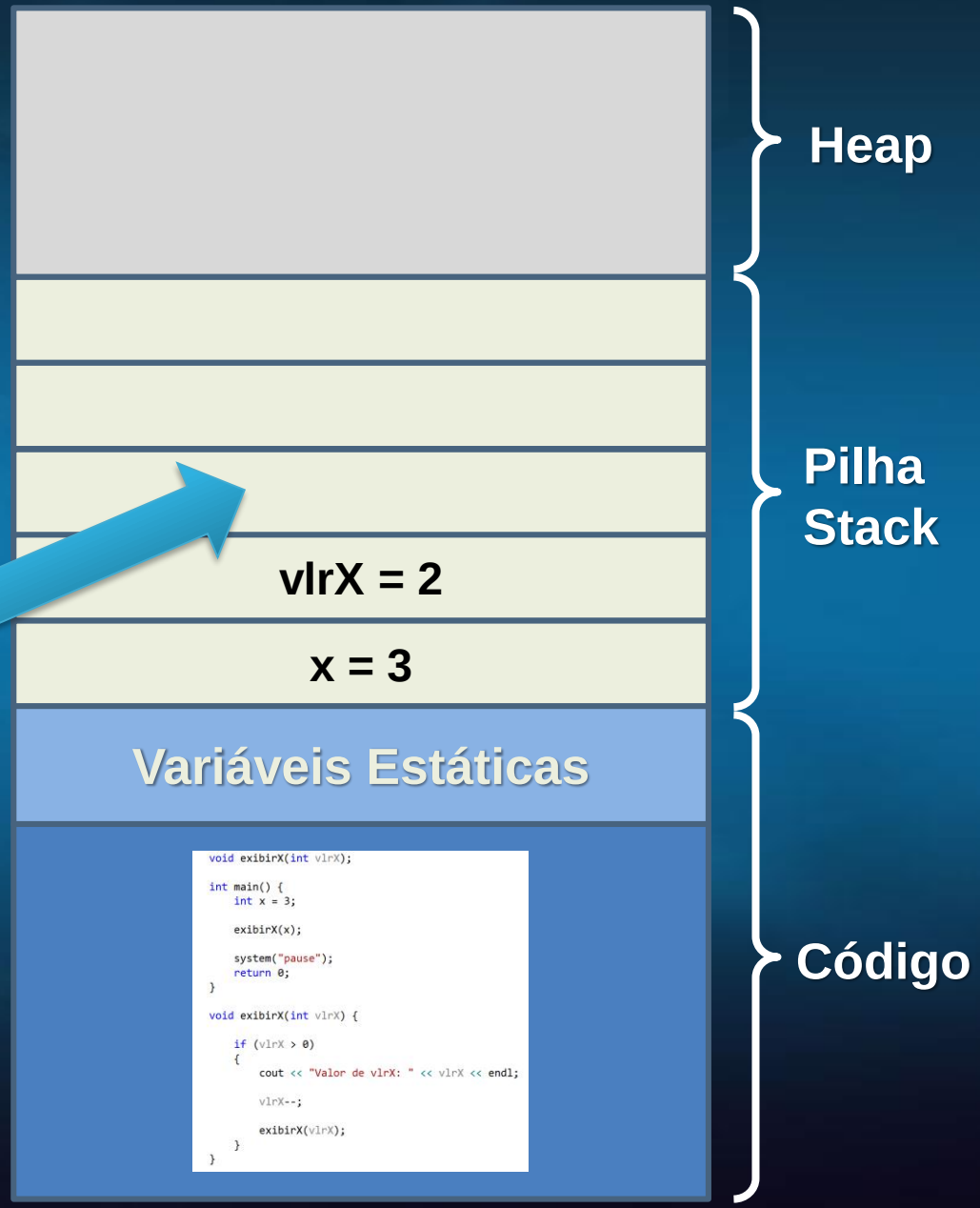
Saída
3
2
1



Funções aninhadas

```
void exibirX(int vlrX);

int main() {
    int x = 3;

    exibirX(x);

    system("pause");
    return 0;
}

void exibirX(int vlrX) {

    if (vlrX > 0)
    {
        cout << "Valor de vlrX: " << vlrX << endl;

        vlrX--;

        exibirX(vlrX);
    }
}
```

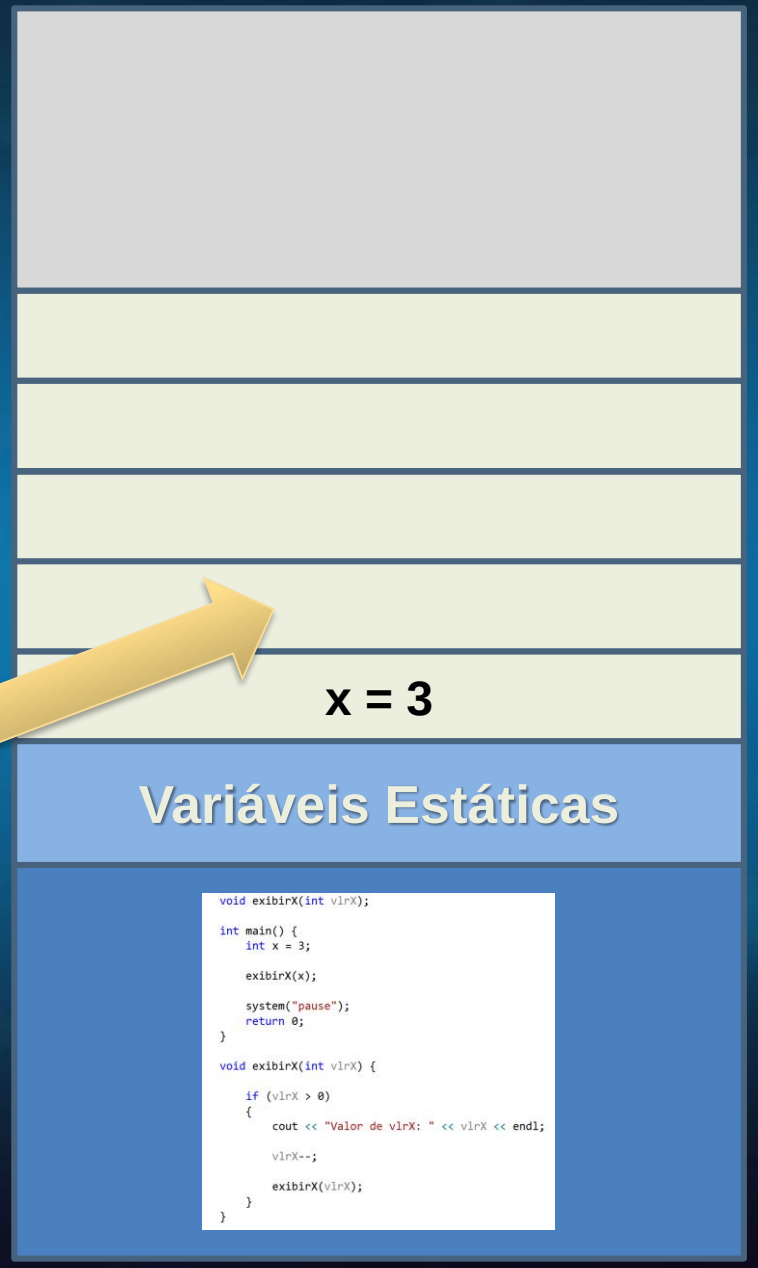
Saída
3
2
1



Funções aninhadas

```
void exibirX(int vlrX);

int main() {
    int x = 3;

    exibirX(x);

    system("pause");
    return 0;
}

void exibirX(int vlrX) {

    if (vlrX > 0)
    {
        cout << "Valor de vlrX: " << [illegible] << endl;

        vlrX--;

        exibi[illegible]rX);
    }
}
```

Saída
3
2
1



Funções aninhadas

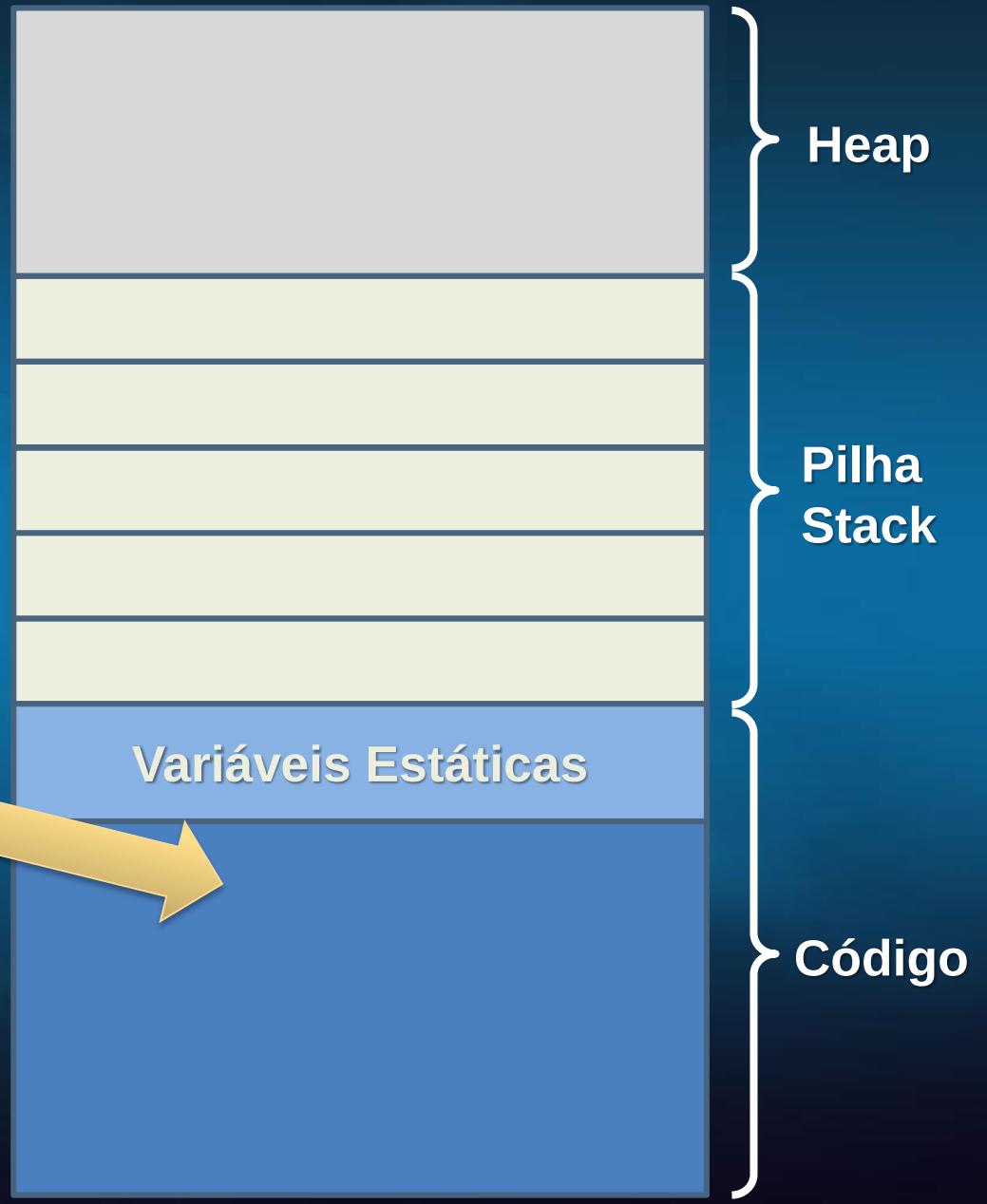
```
void exibirX(int vlrX);

int main() {
    int x = 3;

    exibirX(x);

    system("pause");
    return 0;
}

void exibirX(int vlrX) {

    if (vlrX > 0)
    {
        cout << "Valor de vlrX: " << vlrX << endl;

        vlrX--;

        exibirX(vlrX);
    }
}
```

Saída
3
2
1

Funções aninhadas

```
void exibirX(int vlrX);

int main() {
    int x = 3;

    exibirX(x);

    system("pause");
    return 0;
}

void exibirX(int

    if (vlrX > 0)
    {
        cout << "Valor de vlrX: " << vlrX << endl;

        vlrX--;

        exibirX(vlrX);
    }
}
```

Saída
3
2
1



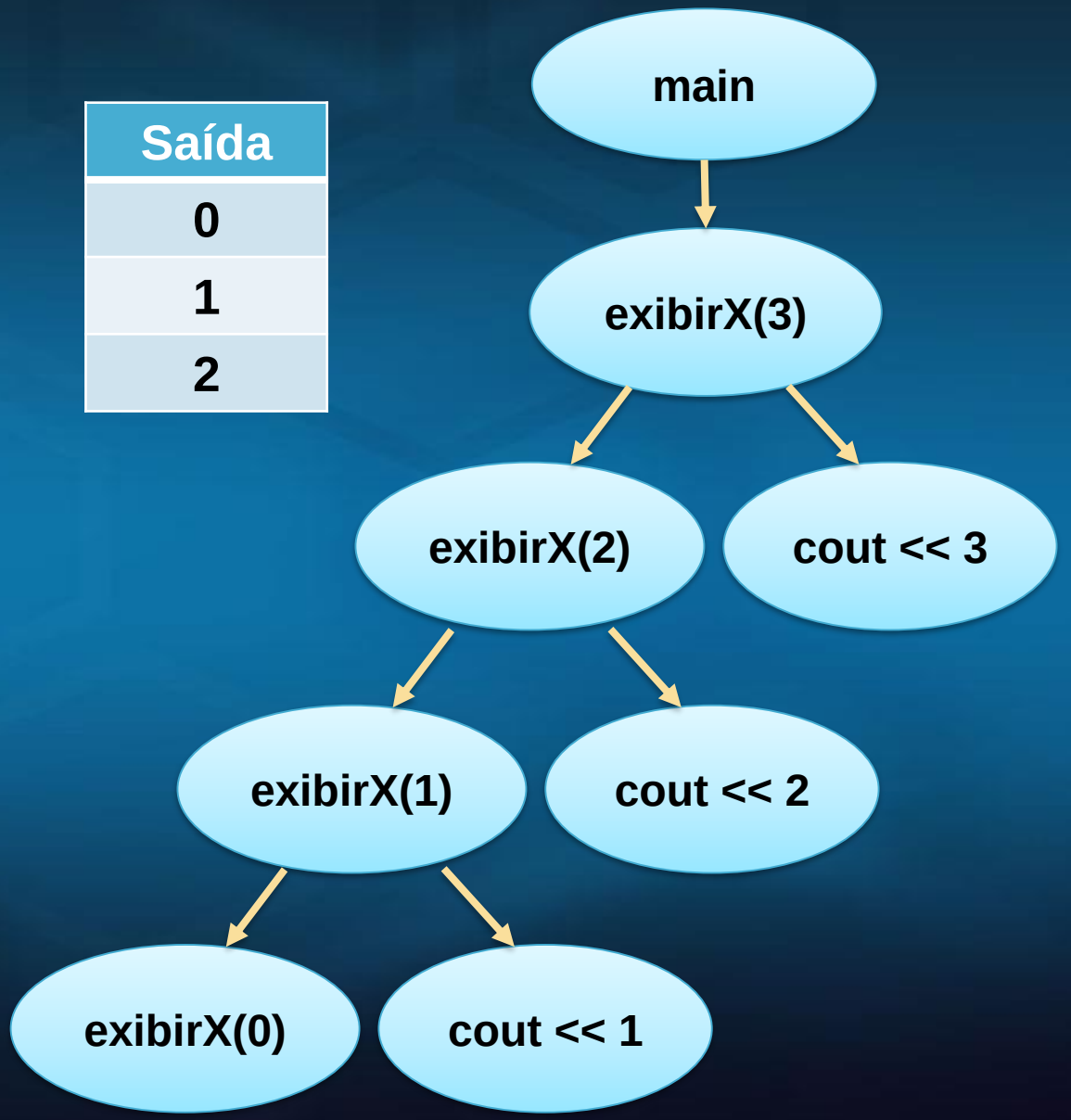
Recursividade – Versão 2

Recursividade

Recursividade

```
int main() {
    int x = 3;

    exibirX(x);

    system("pause");
    return 0;
}

void exibirX(int vlrX) {

    if (vlrX > 0)
    {
        vlrX--;

        exibirX(vlrX);

        cout << "Valor de vlrX: " << vlrX << endl;
    }
}
```

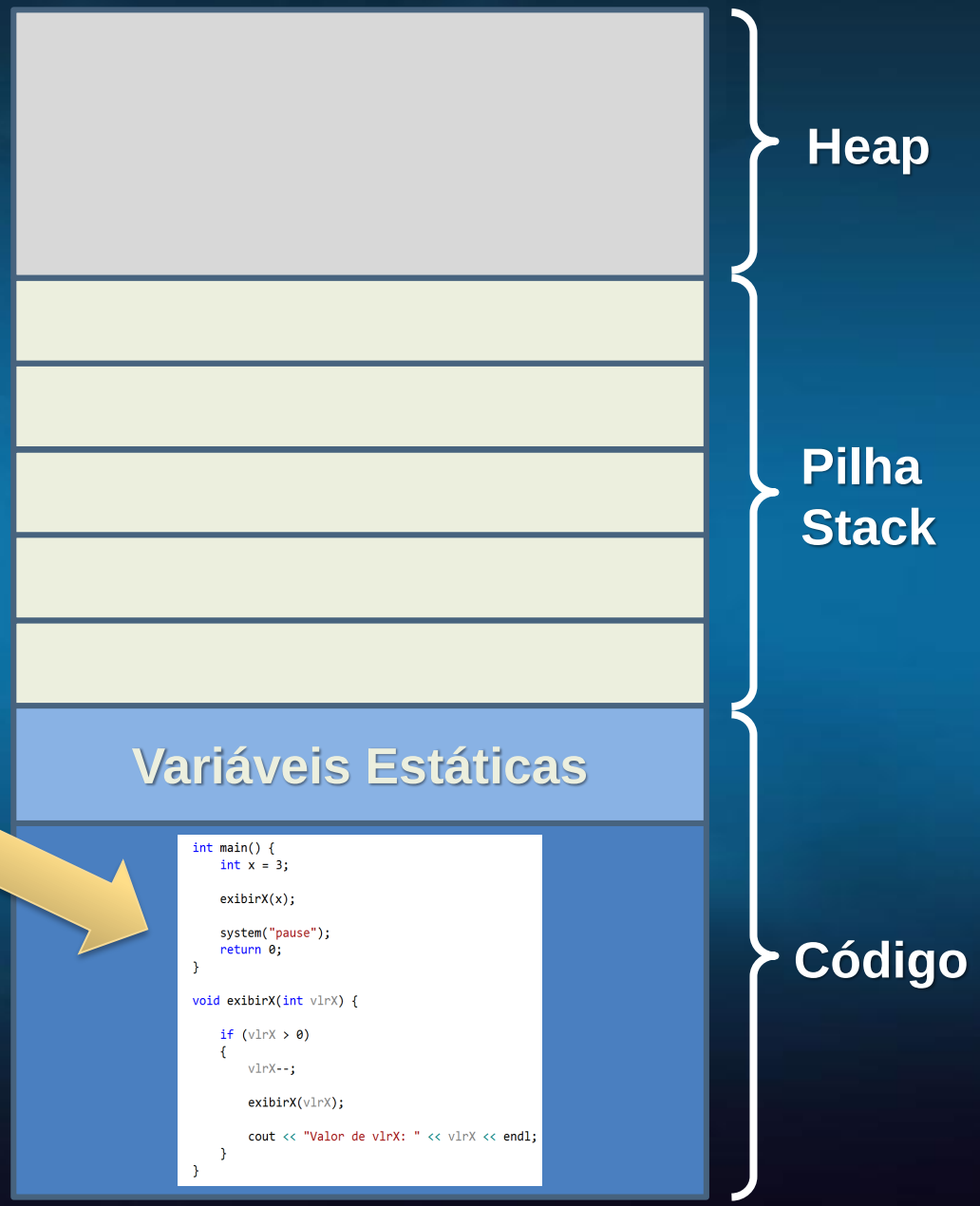
Saída
0
1
2



Funções aninhadas

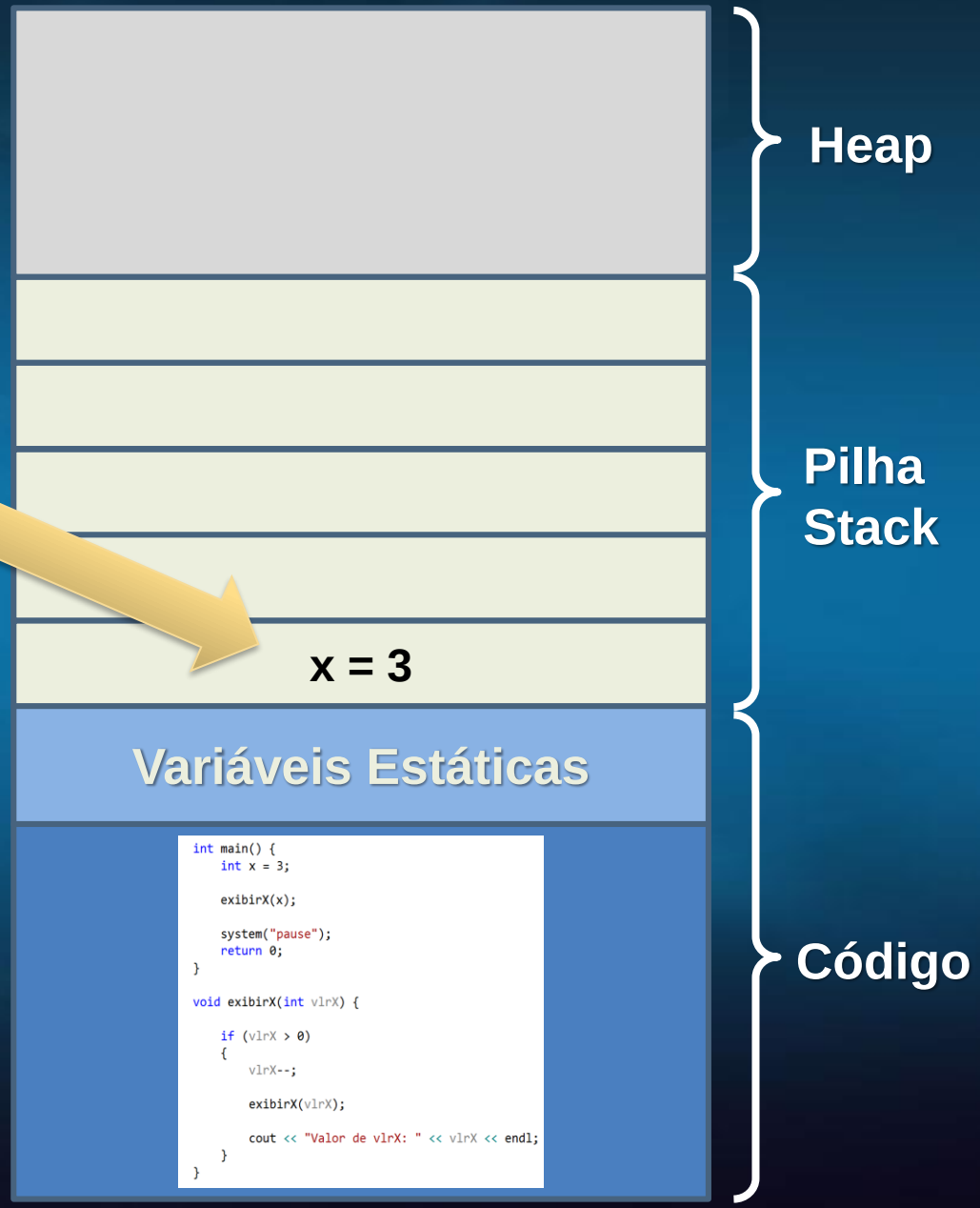
```
int main() {
    int x = 3;

    exibirX(x);

    system("pause");
    return 0;
}

void exibirX(int vlrX) {

    if (vlrX > 0)
    {
        vlrX--;

        exibirX(vlrX);

        cout << "Valor de vlrX: " << vlrX << endl;
    }
}
```

LOAD



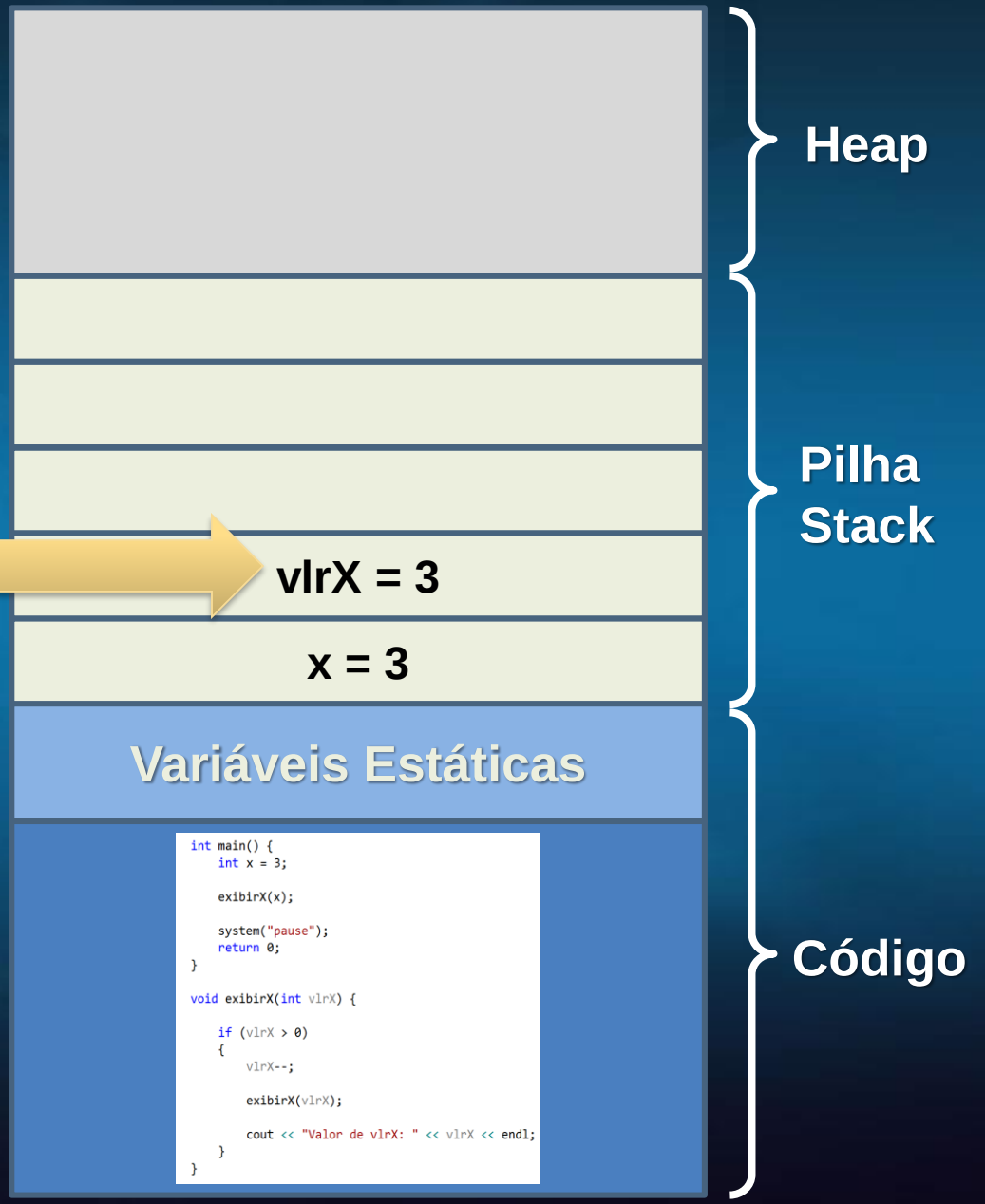
Funções aninhadas

```
int main() {
    int x = 3;

    exibirX(x);

    system("pause");
    return 0;
}

void exibirX(int vlrX) {

    if (vlrX > 0)
    {
        vlrX--;

        exibirX(vlrX);

        cout << "Valor de vlrX: " << vlrX << endl;
    }
}
```



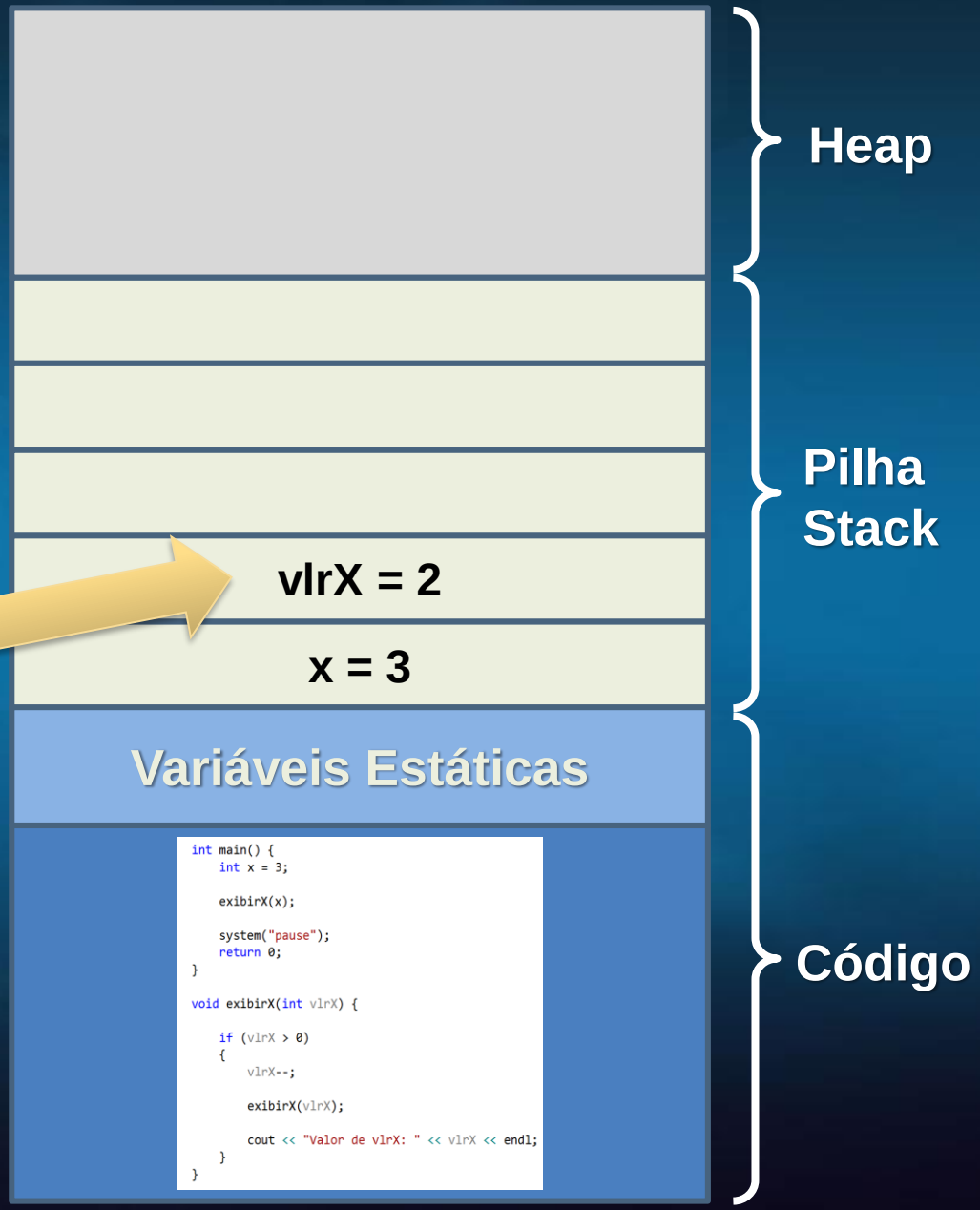
Funções aninhadas

```
int main() {
    int x = 3;

    exibirX(x);

    system("pause");
    return 0;
}

void exibirX(int vlrX) {

    if (vlrX > 0)
    {
        vlrX--;

        exibirX(vlrX);

        cout << "Valor de vlrX: " << vlrX << endl;
    }
}
```



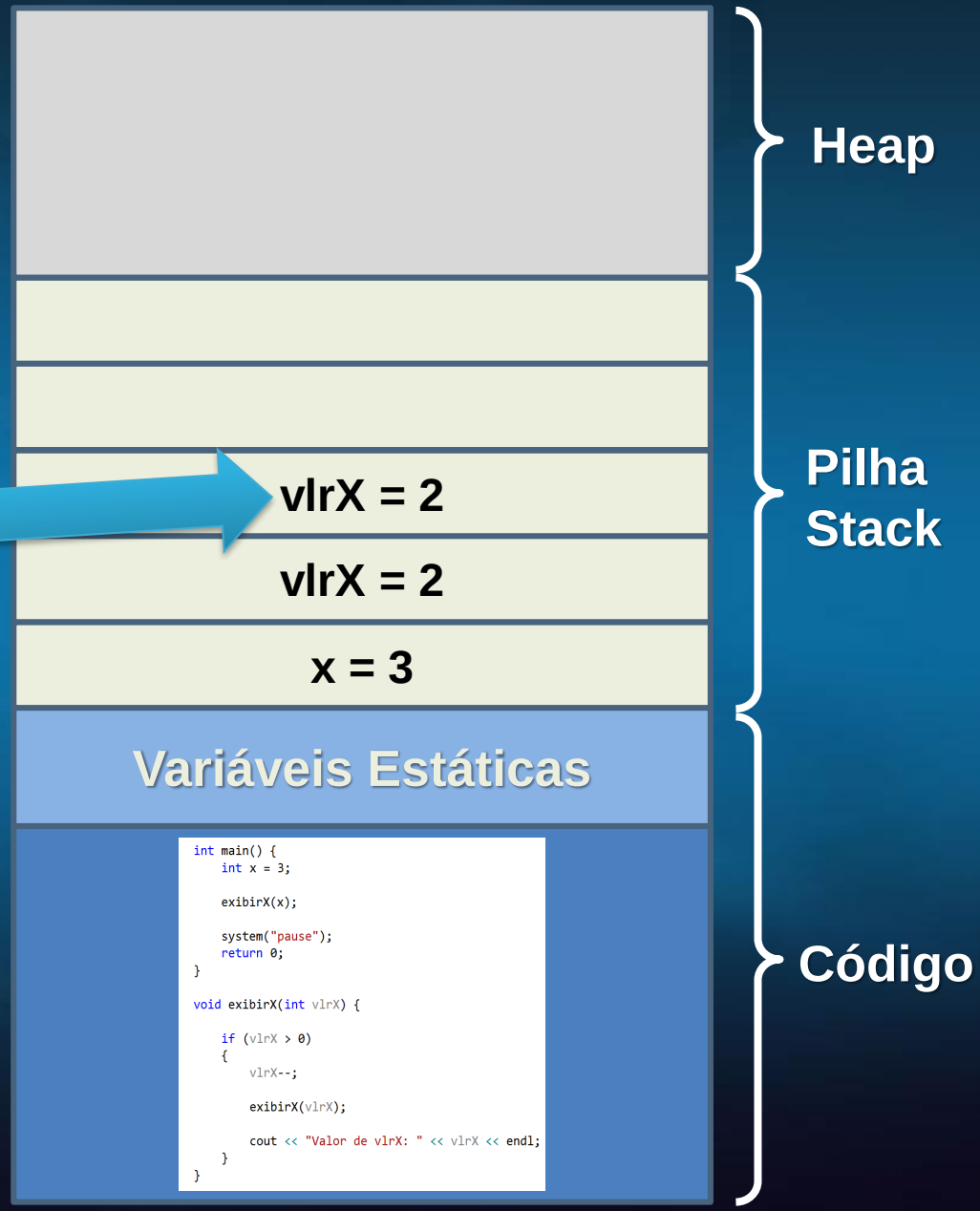
Funções aninhadas

```
int main() {
    int x = 3;

    exibirX(x);

    system("pause");
    return 0;
}

void exibirX(int vlrX) {

    if (vlrX > 0)
    {
        vlrX--;

        exibirX(vlrX);

        cout << "Valor de vlrX: " << vlrX << endl;
    }
}
```



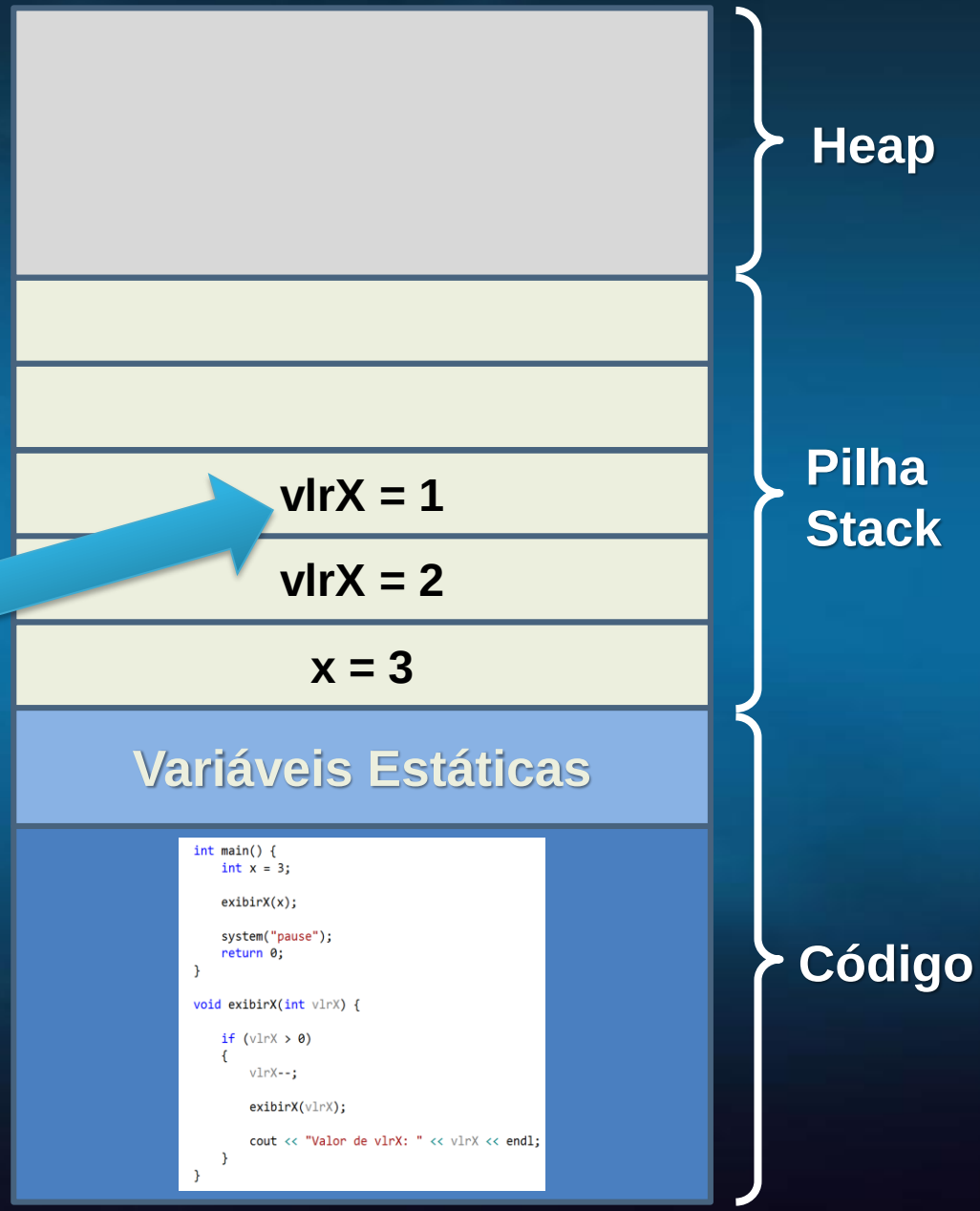
Funções aninhadas

```
int main() {
    int x = 3;

    exibirX(x);

    system("pause");
    return 0;
}

void exibirX(int vlrX) {

    if (vlrX > 0)
    {
        vlrX--;

        exibirX(vlrX);

        cout << "Valor de vlrX: " << vlrX << endl;
    }
}
```



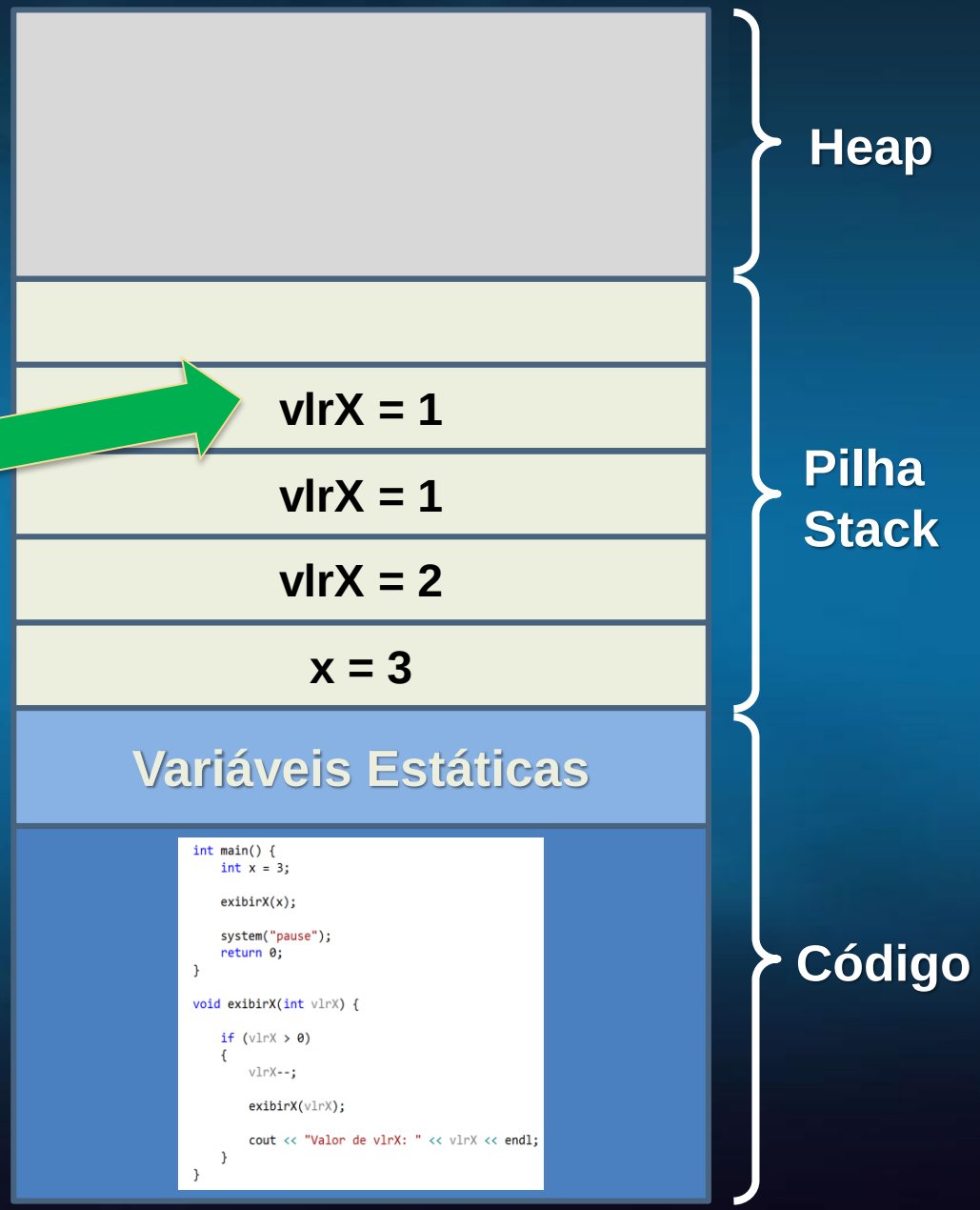
Funções aninhadas

```
int main() {
    int x = 3;

    exibirX(x);

    system("pause");
    return 0;
}

void exibirX(int vlrX) {

    if (vlrX > 0)
    {
        vlrX--;

        exibirX(vlrX);

        cout << "Valor de vlrX: " << vlrX << endl;
    }
}
```



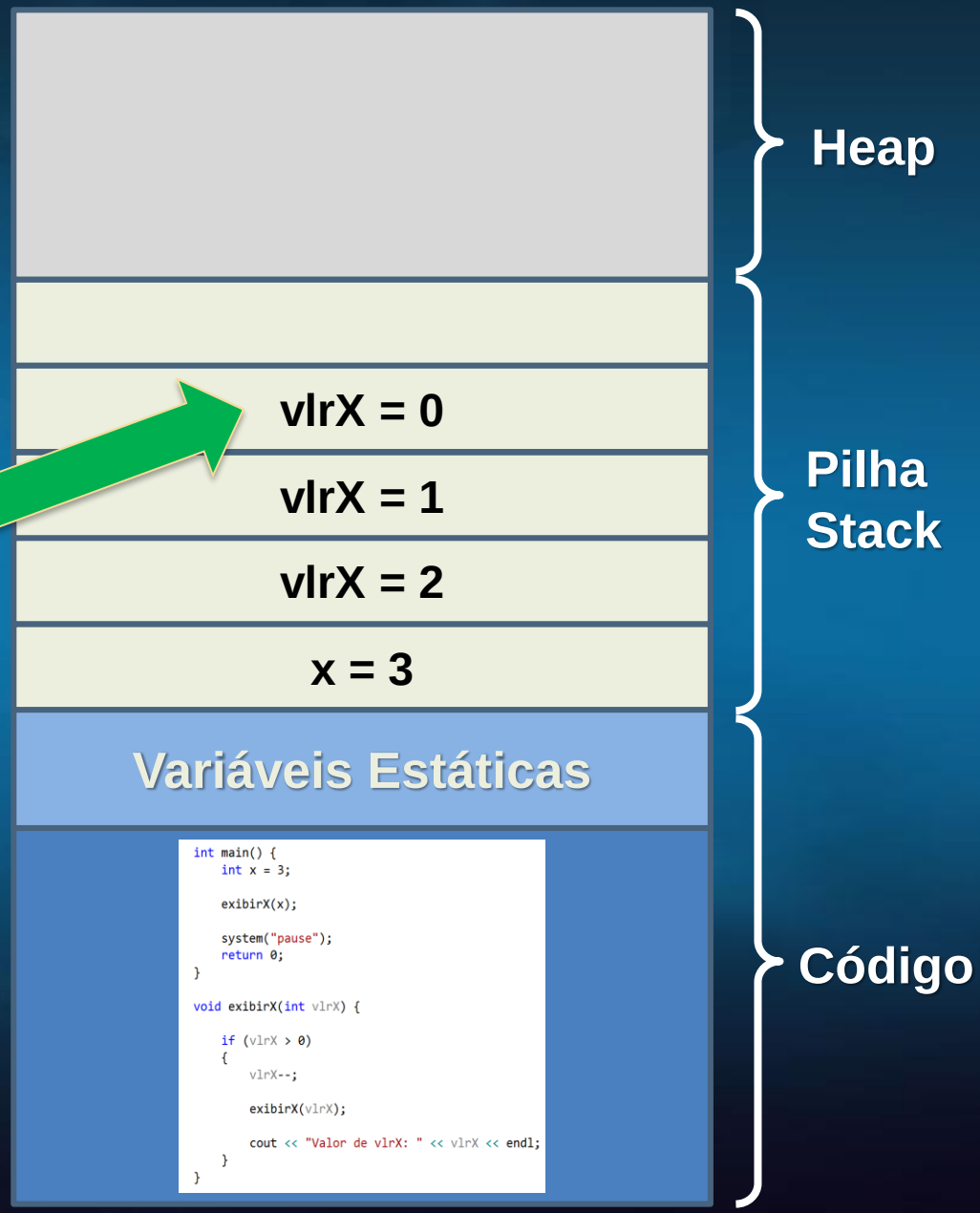
Funções aninhadas

```
int main() {
    int x = 3;

    exibirX(x);

    system("pause");
    return 0;
}

void exibirX(int vlrX) {

    if (vlrX > 0)
    {
        vlrX--;

        exibirX(vlrX);

        cout << "Valor de vlrX: " << vlrX << endl;
    }
}
```



Funções aninhadas

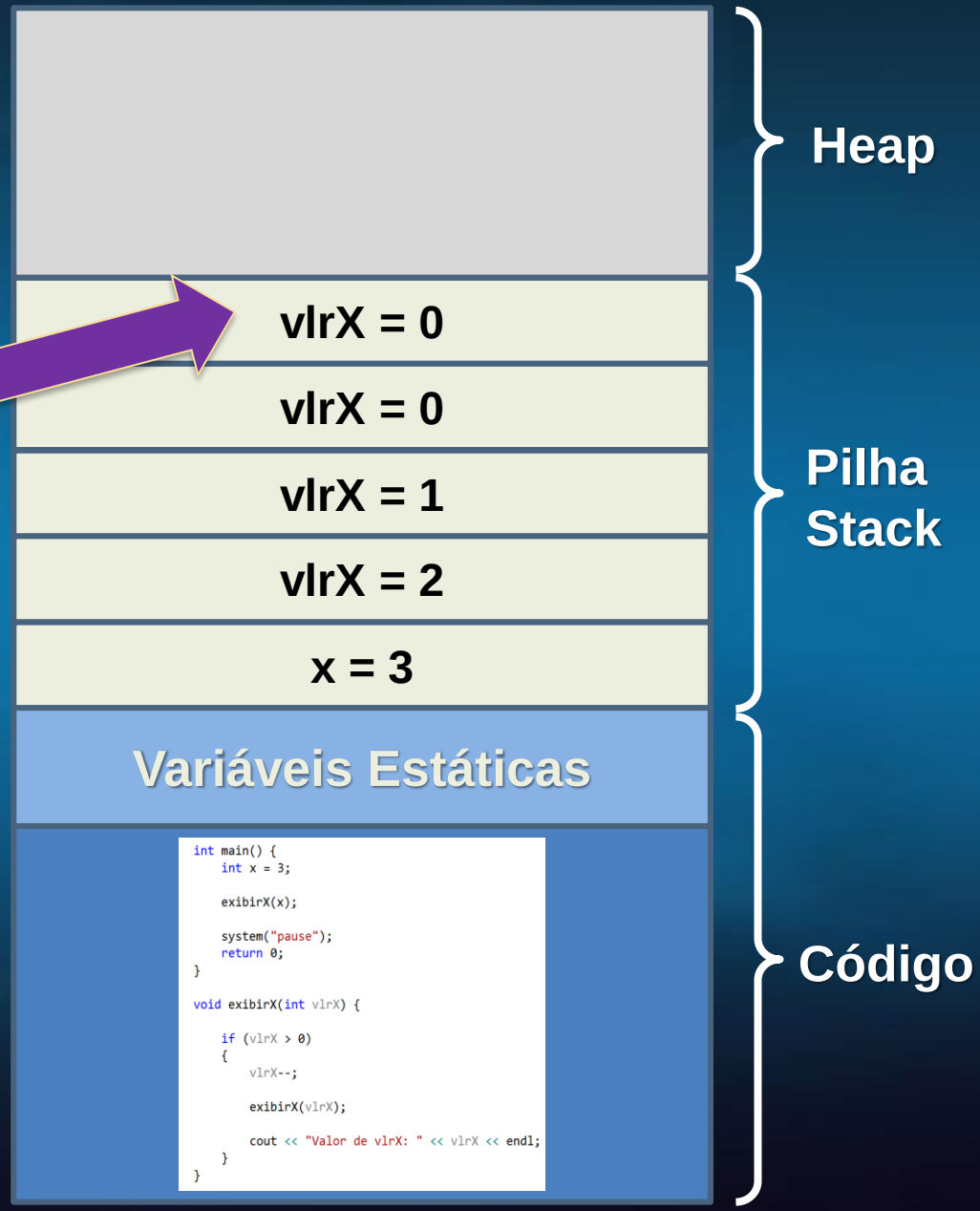
```
int main() {
    int x = 3;

    exibirX(x);

    system("pause");
    return 0;
}

void exibirX(int vlrX) {

    if (vlrX > 0)
    {
        vlrX--;

        exibirX(vlrX);

        cout << "Valor de vlrX: " << vlrX << endl;
    }
}
```



Funções aninhadas

```
int main() {
    int x = 3;

    exibirX(x);

    system("pause");
    return 0;
}

void exibirX(int vlrX) {

    if (vlrX > 0)
    {
        vlrX--;

        exibirX(vlrX);

        cout << "Valor de vlrX: " << vlrX << endl;
    }
}
```

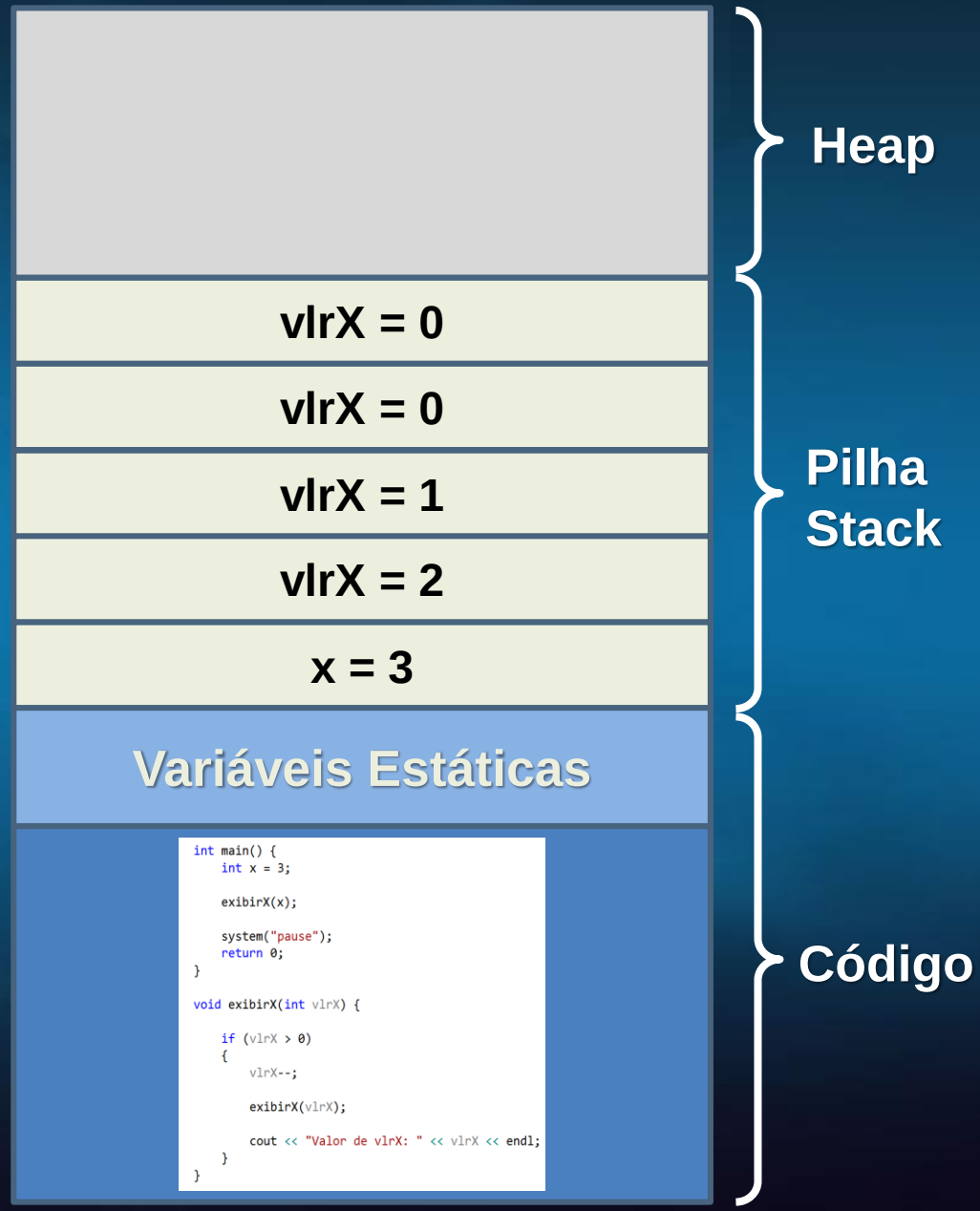
Saída



Funções aninhadas

```
int main() {
    int x = 3;

    exibirX(x);

    system("pause");
    return 0;
}

void exibirX(int vlrX) {

    if (vlrX > 0)
    {
        vlrX--;

        exibirX(vlrX);

        cout << "Valor de vlrX: " << vlrX << endl;
    }
}
```

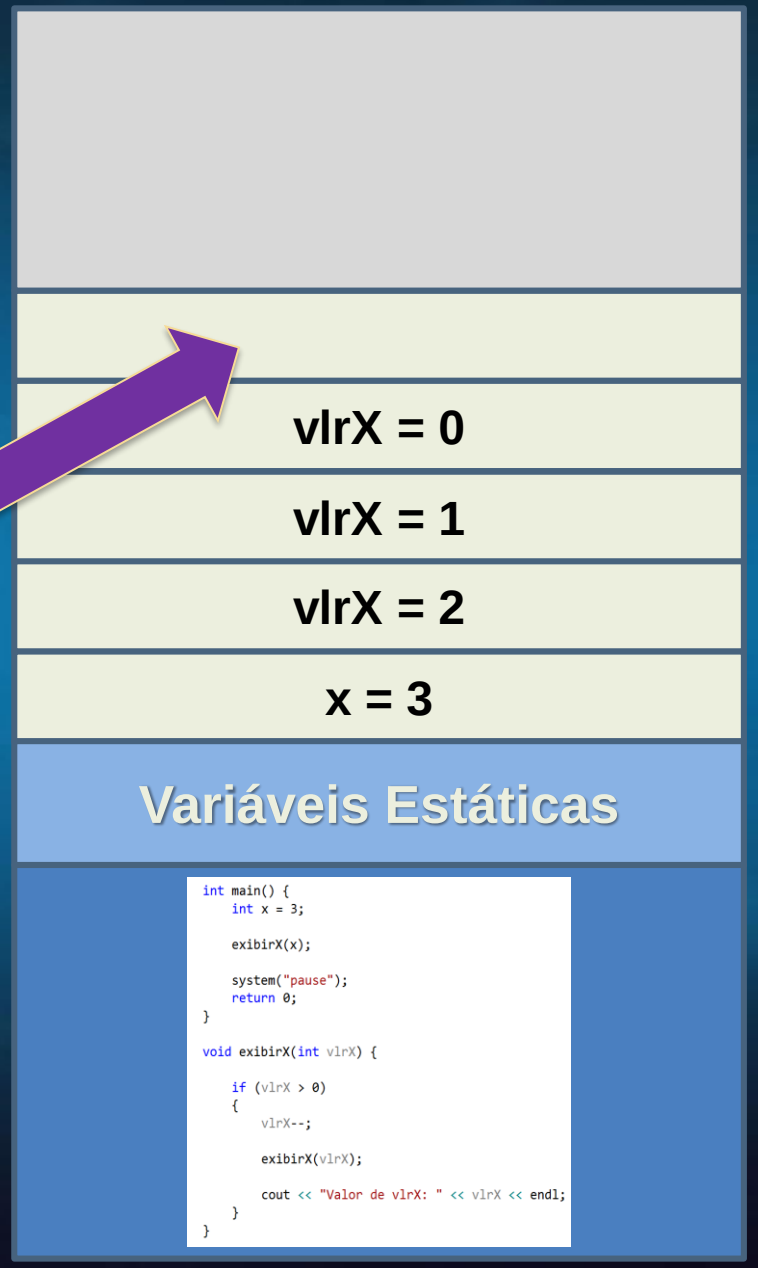
Saída



Funções aninhadas

```
int main() {
    int x = 3;

    exibirX(x);

    system("pause");
    return 0;
}

void exibirX(int vlrX) {

    if (vlrX > 0)
    {
        vlrX--;

        exibirX(vlr

        co  << "Valor de vlrX: " << vlrX << endl;
    }
}
```

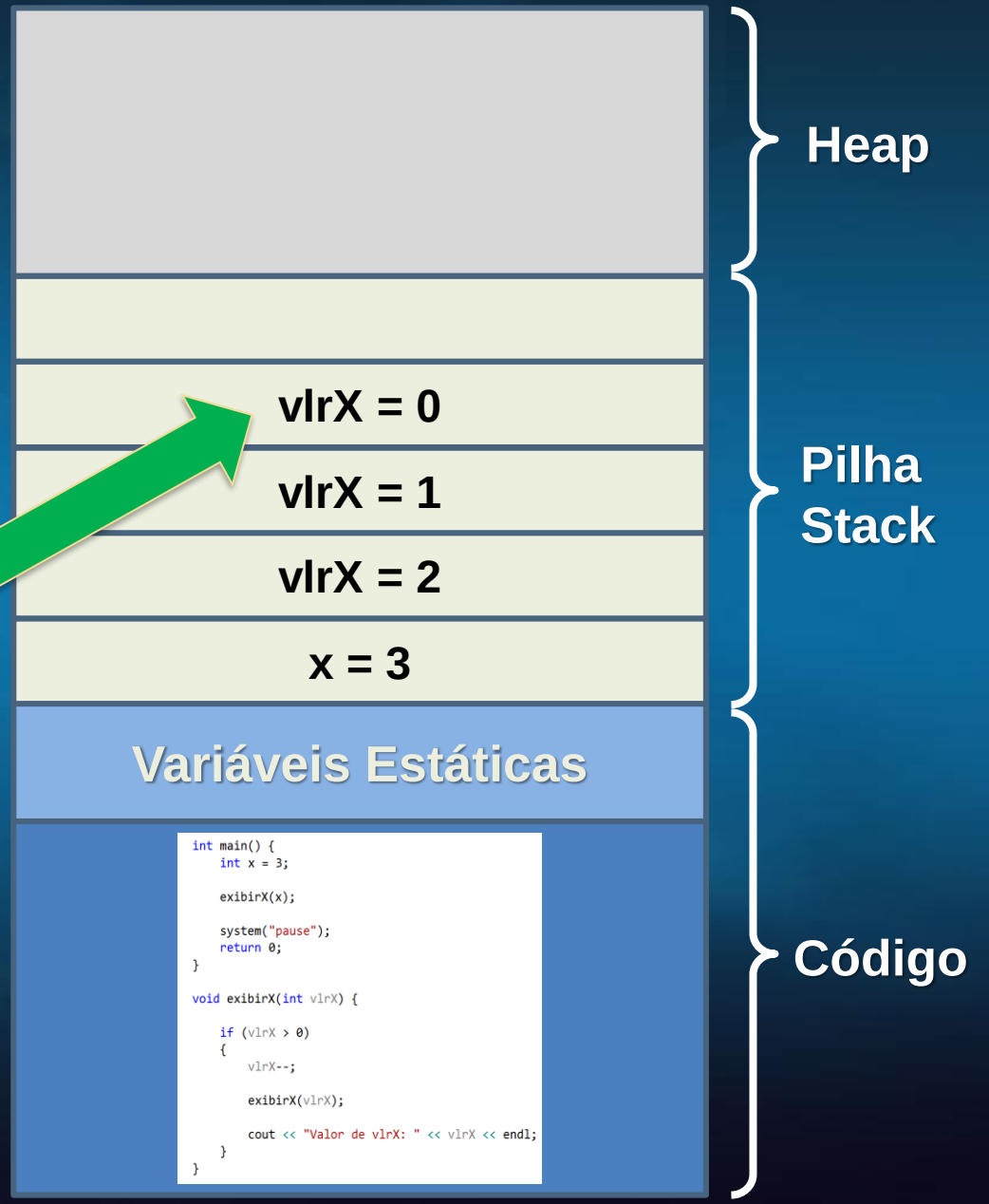
Saída



Funções aninhadas

```
int main() {
    int x = 3;

    exibirX(x);

    system("pause");
    return 0;
}

void exibirX(int vlrX) {

    if (vlrX > 0)
    {
        vlrX--;

        exibirX(vlrX);

        cout << "Valor de vlrX: " << vlrX << endl;
    }
}
```

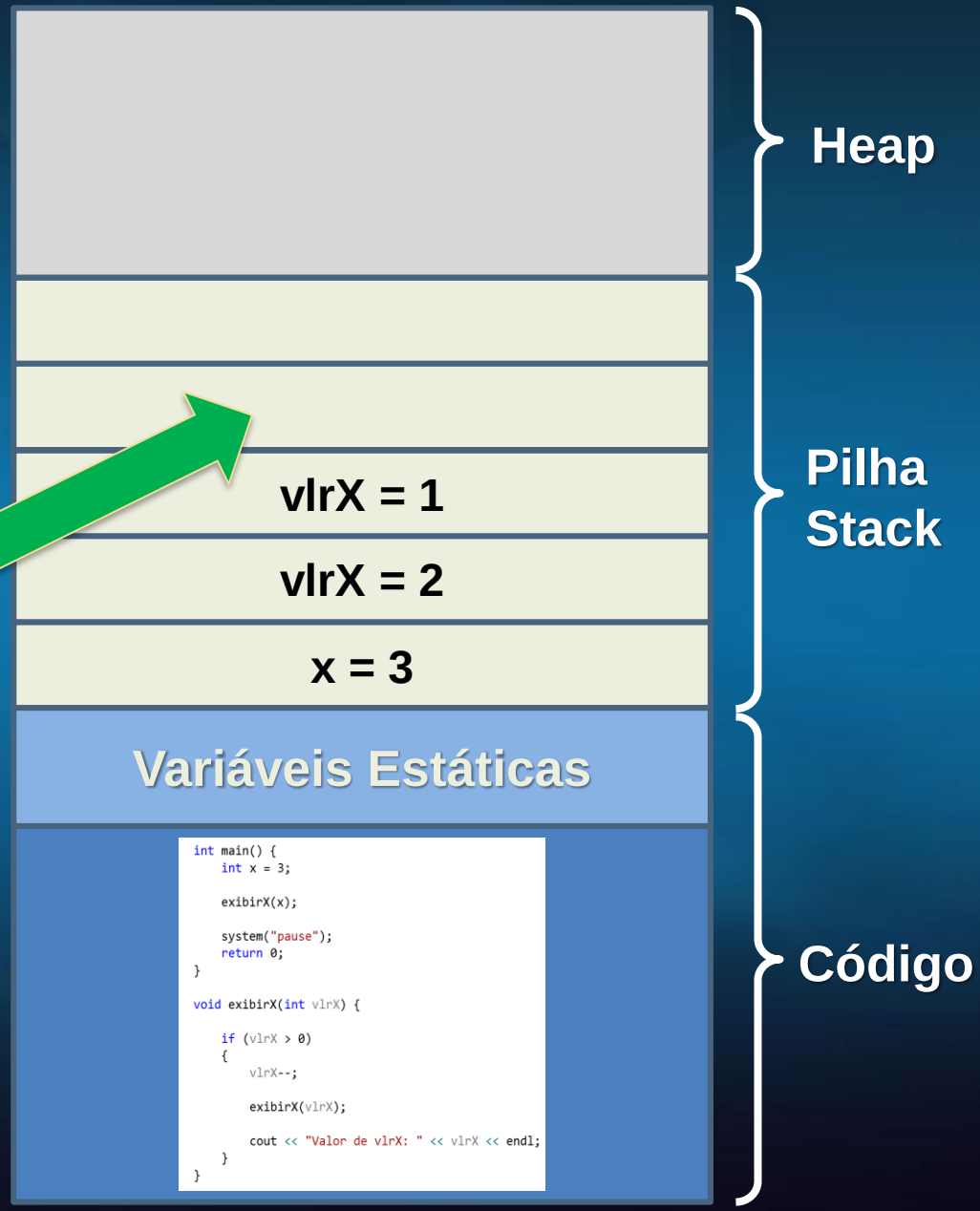
Saída
0



Funções aninhadas

```
int main() {
    int x = 3;

    exibirX(x);

    system("pause");
    return 0;
}

void exibirX(int vlrX) {

    if (vlrX > 0)
    {
        vlrX--;

        exibirX(vlrX);

        cout << "Valor de vlrX: " << vlrX << endl;
    }
}
```

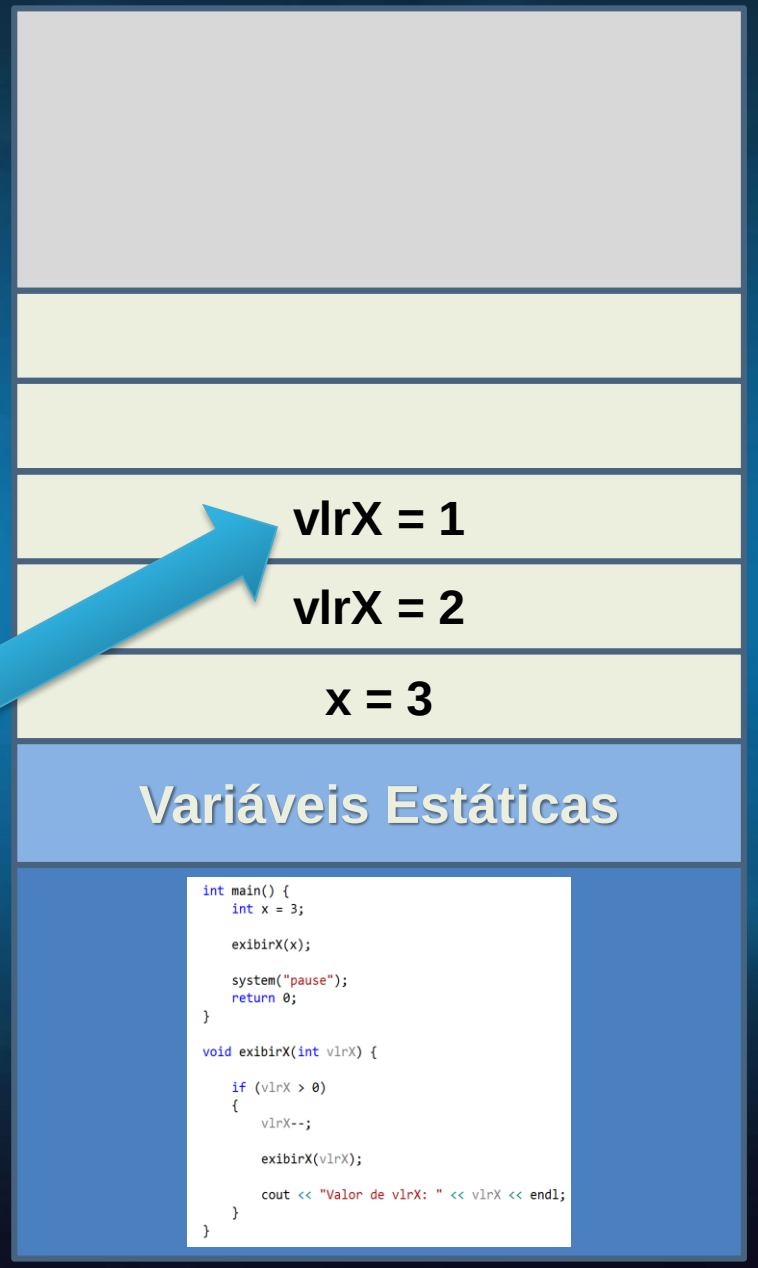
Saída
0



Funções aninhadas

```
int main() {
    int x = 3;

    exibirX(x);

    system("pause");
    return 0;
}

void exibirX(int vlrX) {

    if (vlrX > 0)
    {
        vlrX--;

        exibirX(vlrX);

        cout << "Valor de vlrX: " << vlrX << endl;
    }
}
```

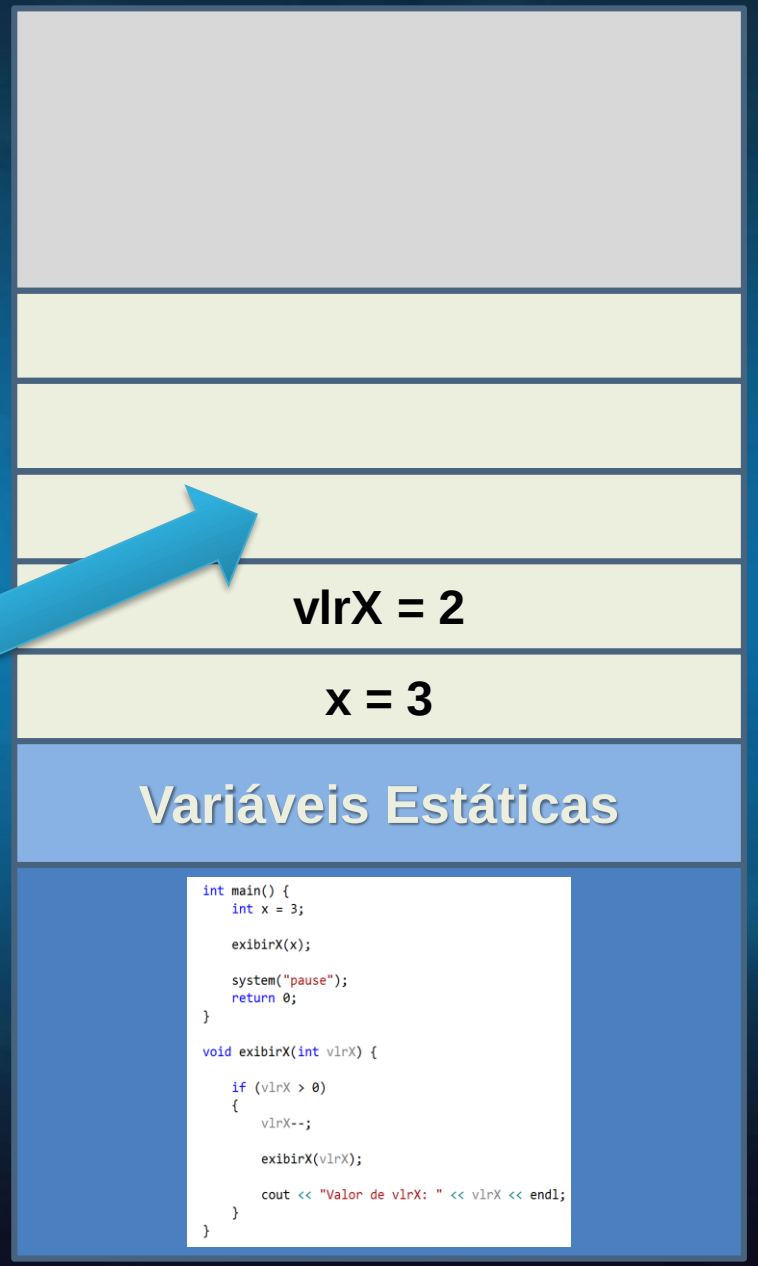
Saída
0
1



Funções aninhadas

```
int main() {
    int x = 3;

    exibirX(x);

    system("pause");
    return 0;
}

void exibirX(int vlrX) {

    if (vlrX > 0)
    {
        vlrX--;

        exibirX(vlrX);

        cout << "Valor de vlrX: " << vlrX << endl;
    }
}
```

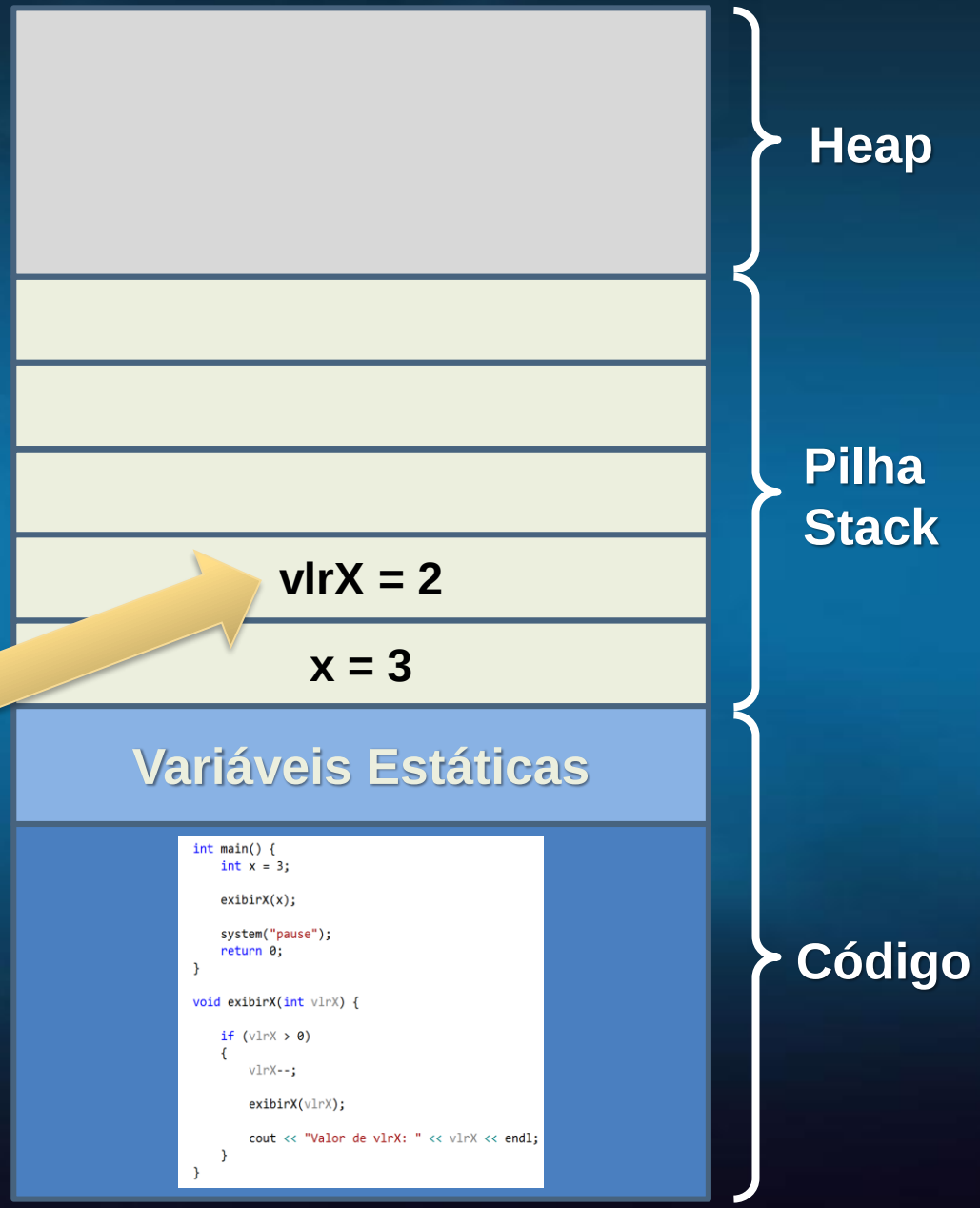
Saída
0
1



Funções aninhadas

```
int main() {
    int x = 3;

    exibirX(x);

    system("pause");
    return 0;
}

void exibirX(int vlrX) {

    if (vlrX > 0)
    {
        vlrX--;

        exibirX(vlrX);

        cout << "Valor de vlrX: " << vlrX << endl;
    }
}
```

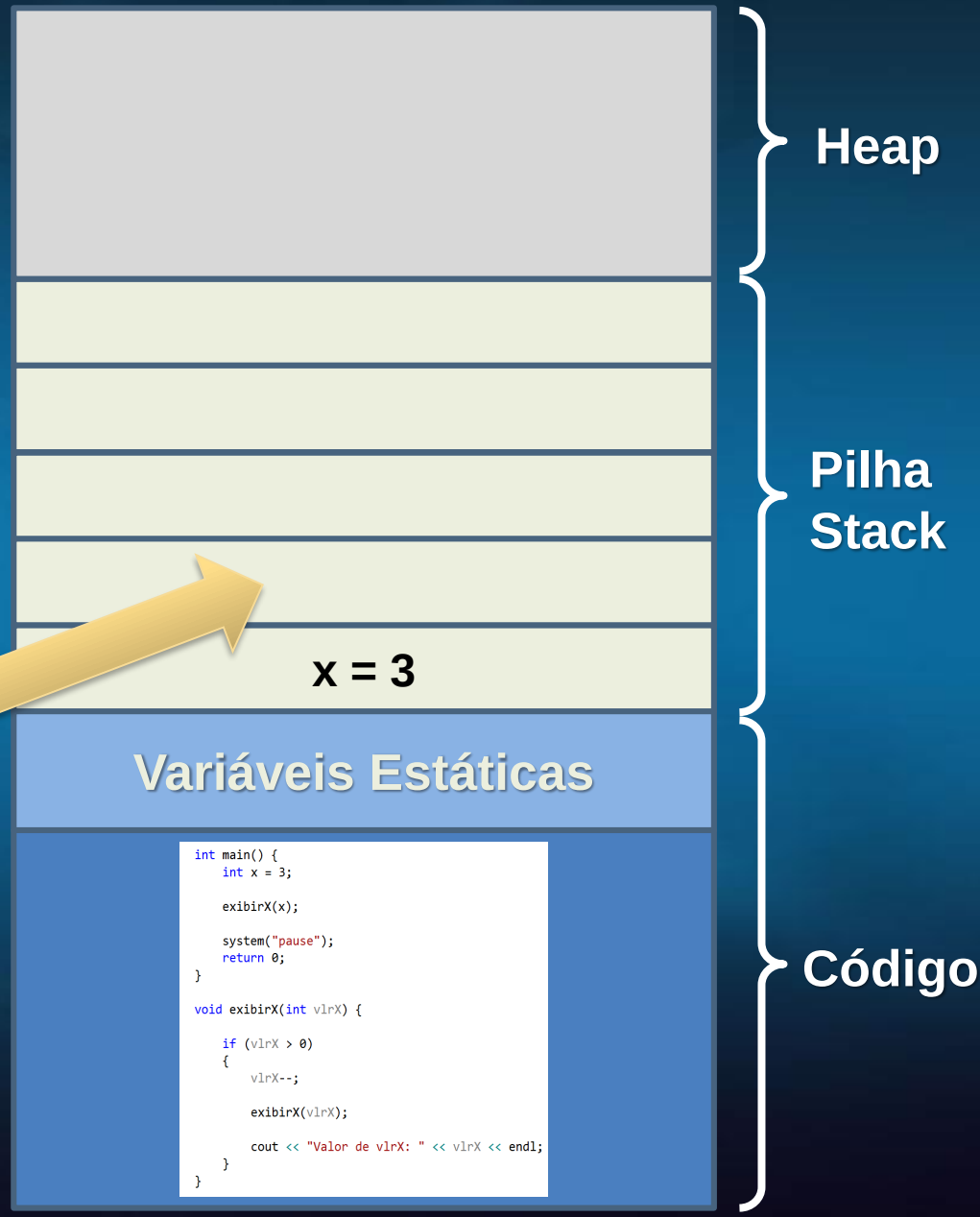
Saída
0
1
2



Funções aninhadas

```
int main() {
    int x = 3;

    exibirX(x);

    system("pause");
    return 0;
}

void exibirX(int vlrX) {

    if (vlrX > 0)
    {
        vlrX--;

        exibirX(vlrX);

        cout << "Valor de vlrX: " << vlrX << endl;
    }
}
```

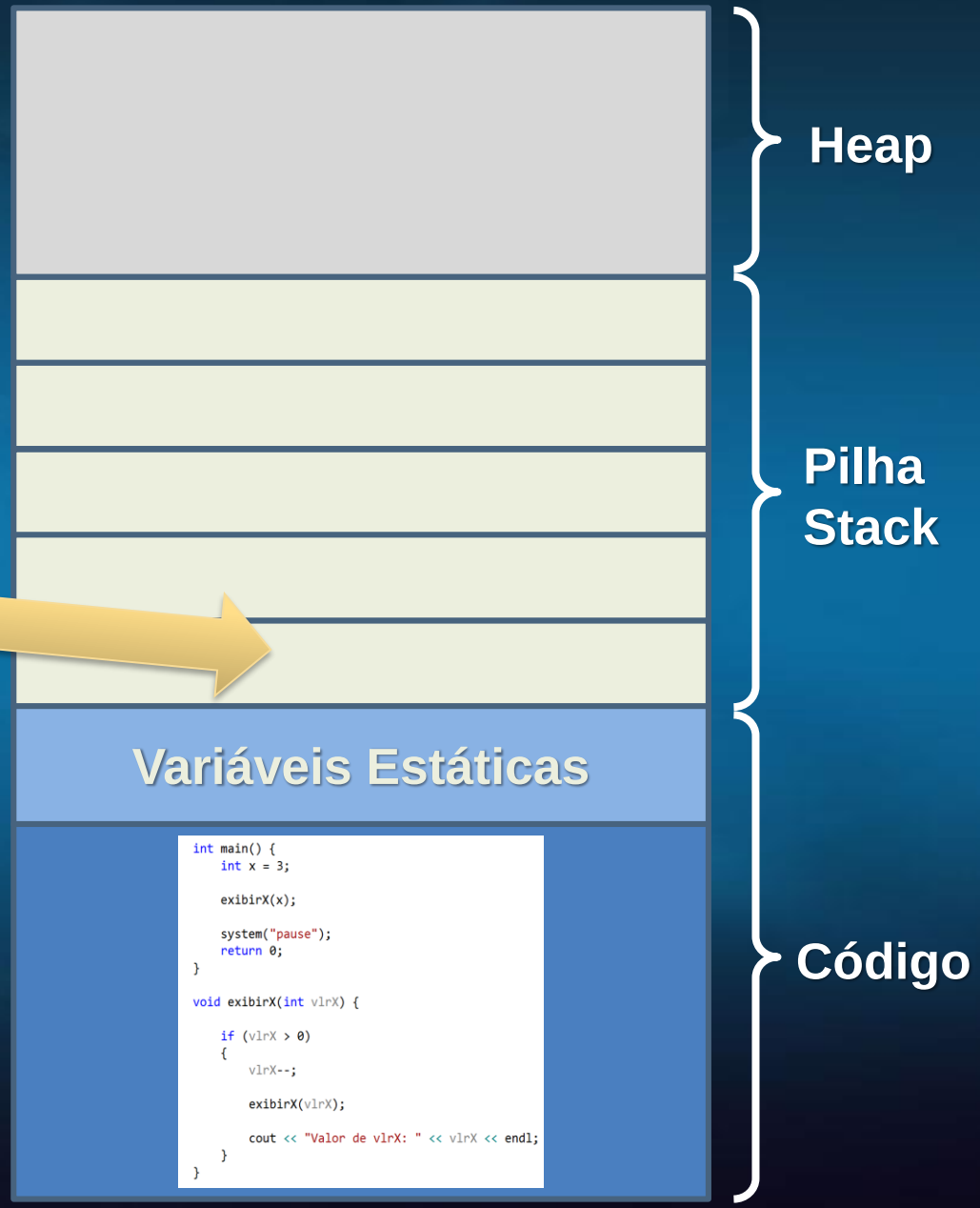
Saída
0
1
2



Funções aninhadas

```
int main() {
    int x = 3;

    exibirX(x);

    system("pause");
    return 0;
}

void exibirX(int vlrX) {

    if (vlrX > 0)
    {
        vlrX--;

        exibirX(vlrX);

        cout << "Valor de vlrX: " << vlrX << endl;
    }
}
```

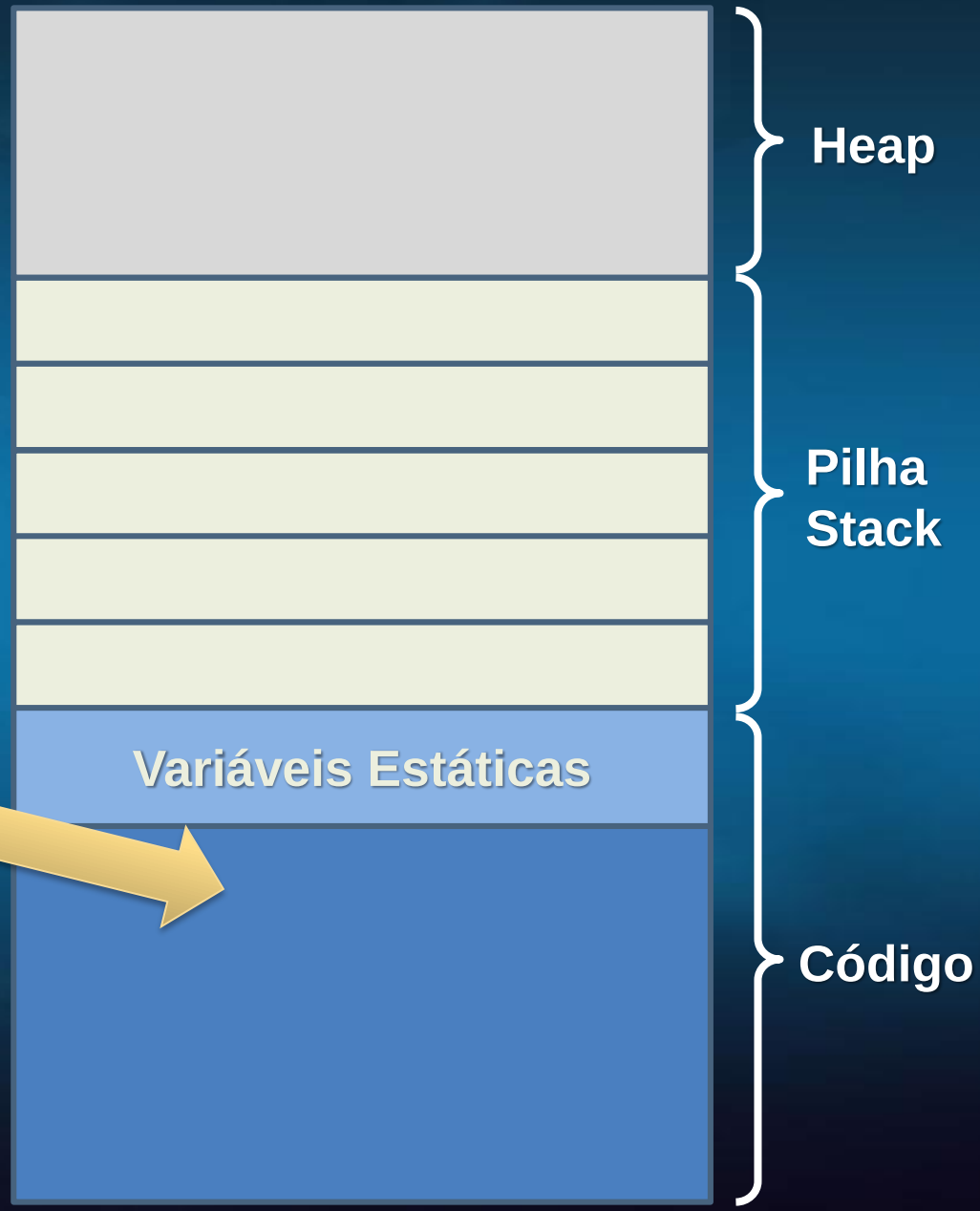
Saída
0
1
2



Funções aninhadas

```
int main() {
    int x = 3;

    exibirX(x);

    system("pause");
    return 0;
}

void exi    t vlrX) {

    if (vlrX > 0)
    {
        vlrX--;

        exibirX(vlrX);

        cout << "Valor de vlrX: " << vlrX << endl;
    }
}
```

Saída
0
1
2



Função recursiva

- Comparação de estrutura de repetição e função recursiva.

Sem condição de parada do while e da função recursiva.
Repetição infinita.

```
while(true) {

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

}
```

```
void exibirX(int vlrX) {

    cout << "Valor de vlrX: " << vlrX << endl;

    exibirX(vlrX);
    ...
}
```

Estrutura de Repetição

Função recursiva

- Adicionar algum critério **(if)** que evite passar pela função recursiva

Colocar critério de parada

```
void exibirX(int vlrX) {

    cout << "Valor de vlrX: " << vlrX << endl;

    exibirX(vlrX);

}
```

Exemplo 1: Critério de Parada

- Função recursiva que imprime números de 0 a 10.

```
int main() {
    int x = 0;

    exibirX(x);

    system("pause");
    return 0;
}

void exibirX(int vlrX) {

    if (vlrX <= 10) {

        cout << "Valor de vlrX: " << vlrX << endl;

        vlrX++;

        exibirX(vlrX);
    }
}
```

Exemplo 1: Critério de Parada

- Função recursiva que imprime números de 0 a 10.

```
int main() {
    int x = 0;

    while(x <= 10) {

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

        x++;
    }

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

```
int main() {
    int x = 0;

    exibirX(x);

    system("pause");
    return 0;
}

void exibirX(int vlrX) {

    if (vlrX <= 10) {

        cout << "Valor de vlrX: " << vlrX << endl;

        vlrX++;

        exibirX(vlrX);
    }
}
```

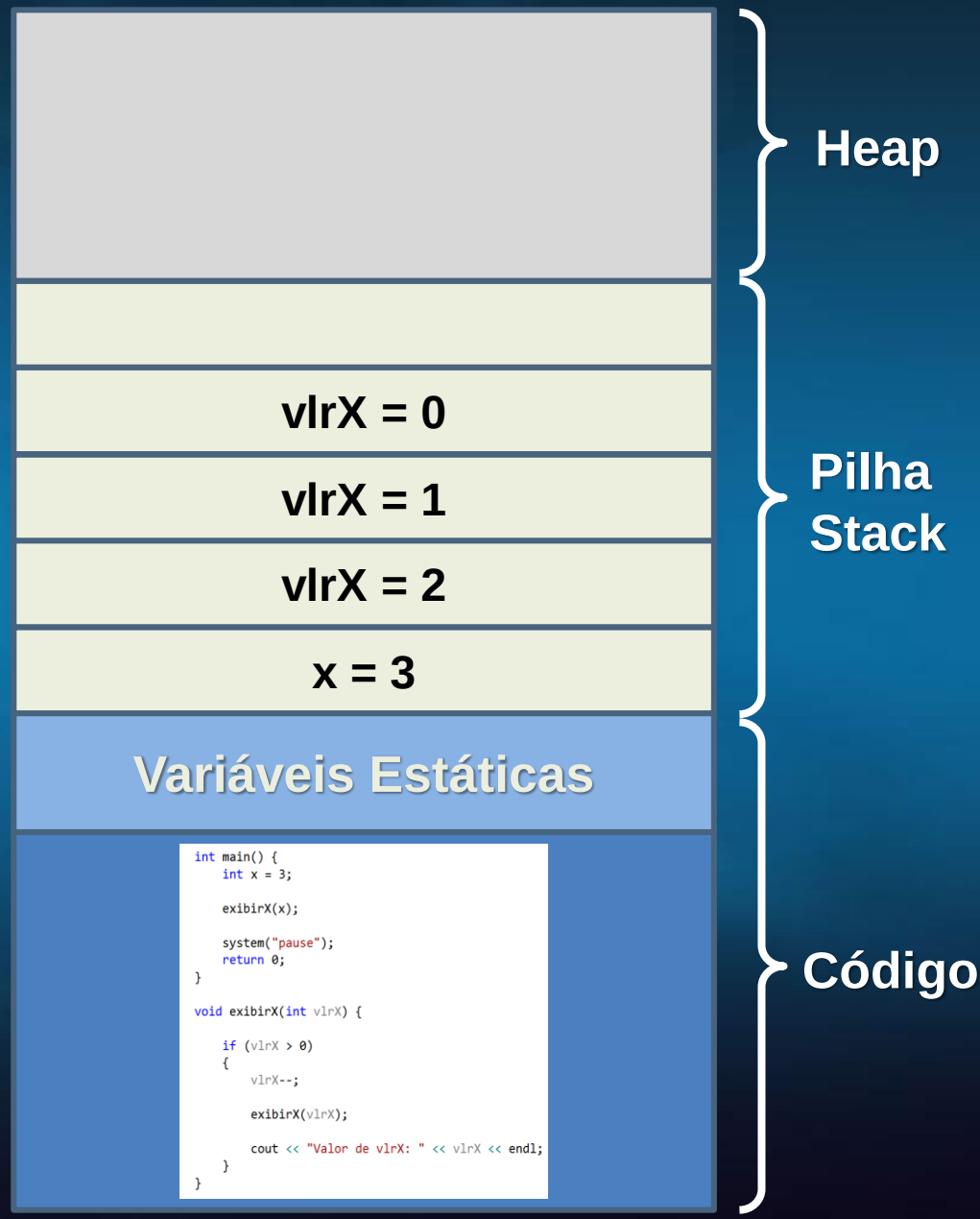
Exemplo 2: Critério de Parada

- Função recursiva que imprime números de 0 a 10.

```
int main() {
    int x = 0;

    exibirX(x);

    system("pause");
    return 0;
}

void exibirX(int vlrX) {

    if (vlrX > 10) {
        return;
    }

    cout << "Valor de vlrX: " << vlrX << endl;

    vlrX++;

    exibirX(vlrX);
}
```

Posição dos dados

```
int main() {
    int x = 3;

    exibirX(x);

    system("pause");
    return 0;
}

void exibirX(int vlrX) {

    if (vlrX > 0)
    {
        vlrX--;



        exibirX(vlrX);


    }
}
```

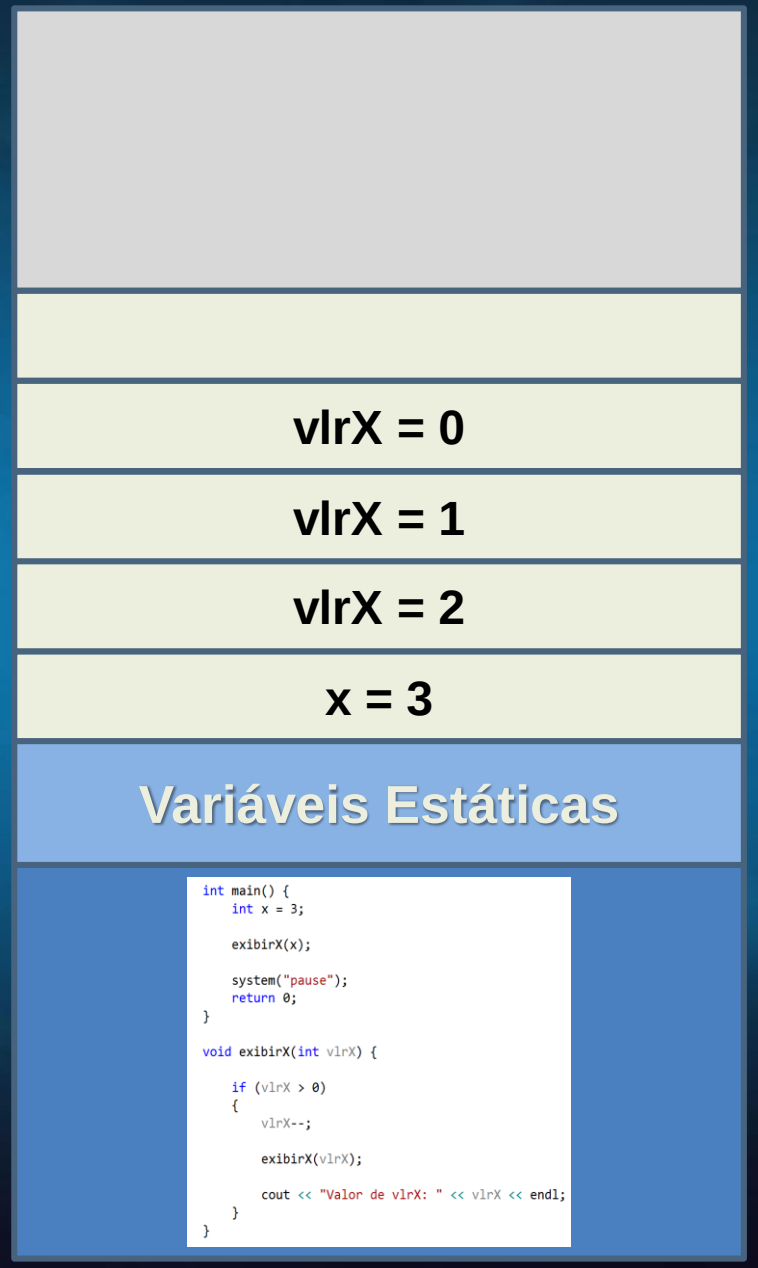


Posição dos dados

```
int main() {
    int x = 3;

    exibirX(x);

    system("pause");
    return 0;
}

void exibirX(int vlrX) {

    if (vlrX > 0)
    {
        vlrX--;

        cout << "Valor de vlrX: " << vlrX << endl;

        exibirX(vlrX);


    }
}
```

Saída
2
1
0

Posição possível para exibir os dados

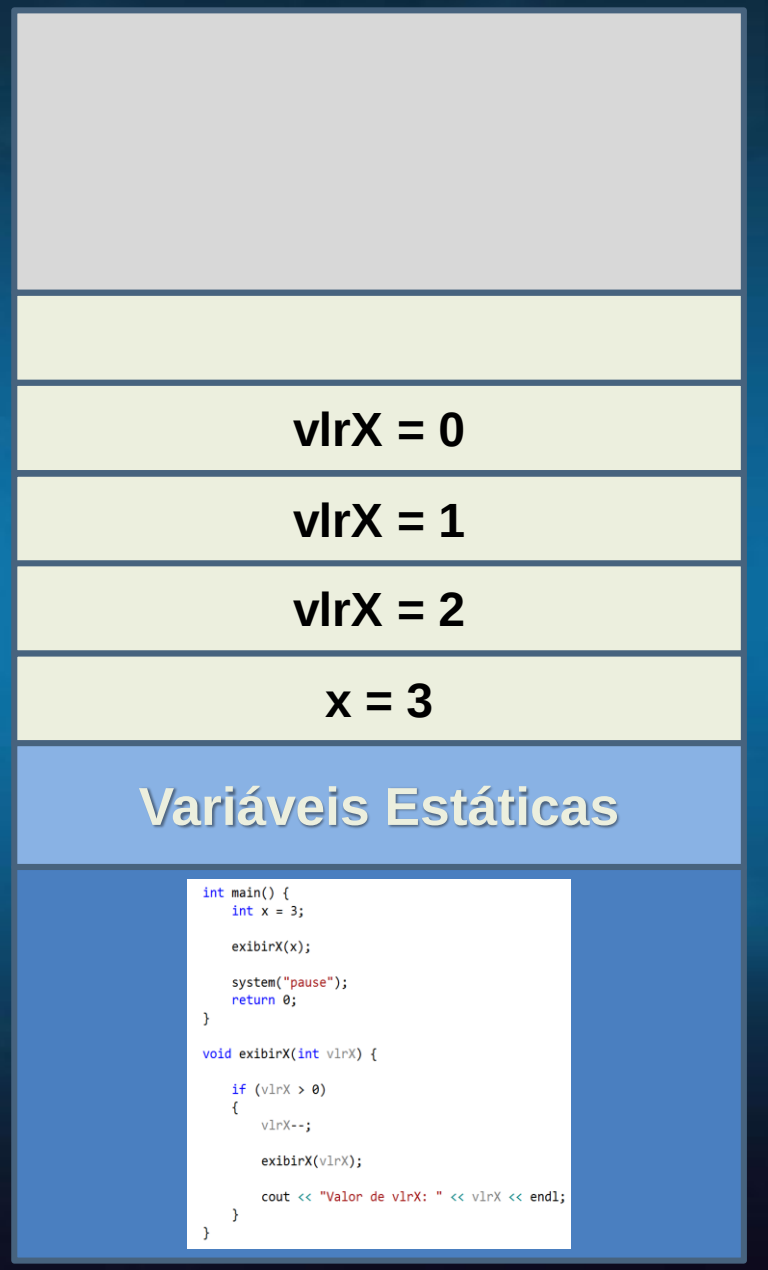


Posição dos dados

```
int main() {
    int x = 3;

    exibirX(x);

    system("pause");
    return 0;
}

void exibirX(int vlrX) {

    if (vlrX > 0)
    {
        vlrX--;



        exibirX(vlrX);
        cout << "Valor de vlrX: " << vlrX << endl;
    }
}
```

Saída
0
1
2

Posição possível para exibir os dados



Exercício

- Crie uma função recursiva que imprima os número pares entre 0 a 20.

Exercício

```
int main() {
    int x = 0;

    exibirX(x);

    system("pause");
    return 0;
}

void exibirX(int vlrX) {

    if (vlrX > 20) {
        return;
    }

    cout << "Valor de vlrX: " << vlrX << endl;

    vlrX = vlrX + 2;

    exibirX(vlrX);
}
```

Exercício

- Faça um programa que peça ao usuário que digite um número. Em seguida chame uma função recursiva que exiba a contagem regressiva a partir do número informado até 0.

Exercício

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

    int x;

    cout << "Informe um número: ";
    cin >> x;

    exibirX(x);

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

```
void exibirX(int vlrX) {

    if (vlrX <= -1) {
        return;
    }

    cout << "Valor de vlrX: " << vlrX << endl;

    vlrX--;

    exibirX(vlrX);
}
```

F I M

“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