

Tecnologia Em Analise e Desenv. de Sistemas

# Estrutura de Dados

Aula 5 – Ponteiros – Parte 2

Prof. Dr. Ricardo Luis Balieiro

# Problema

## • Retorno por **VALOR**

```
int retornaNumero();

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

    int A;

    A = retornaNumero();

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

    system("pause");
    return 0;
}

int retornaNumero() {
    int x = 100;

    return x;
}
```



# Problema

```
int retornaNumero();

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

    int A;

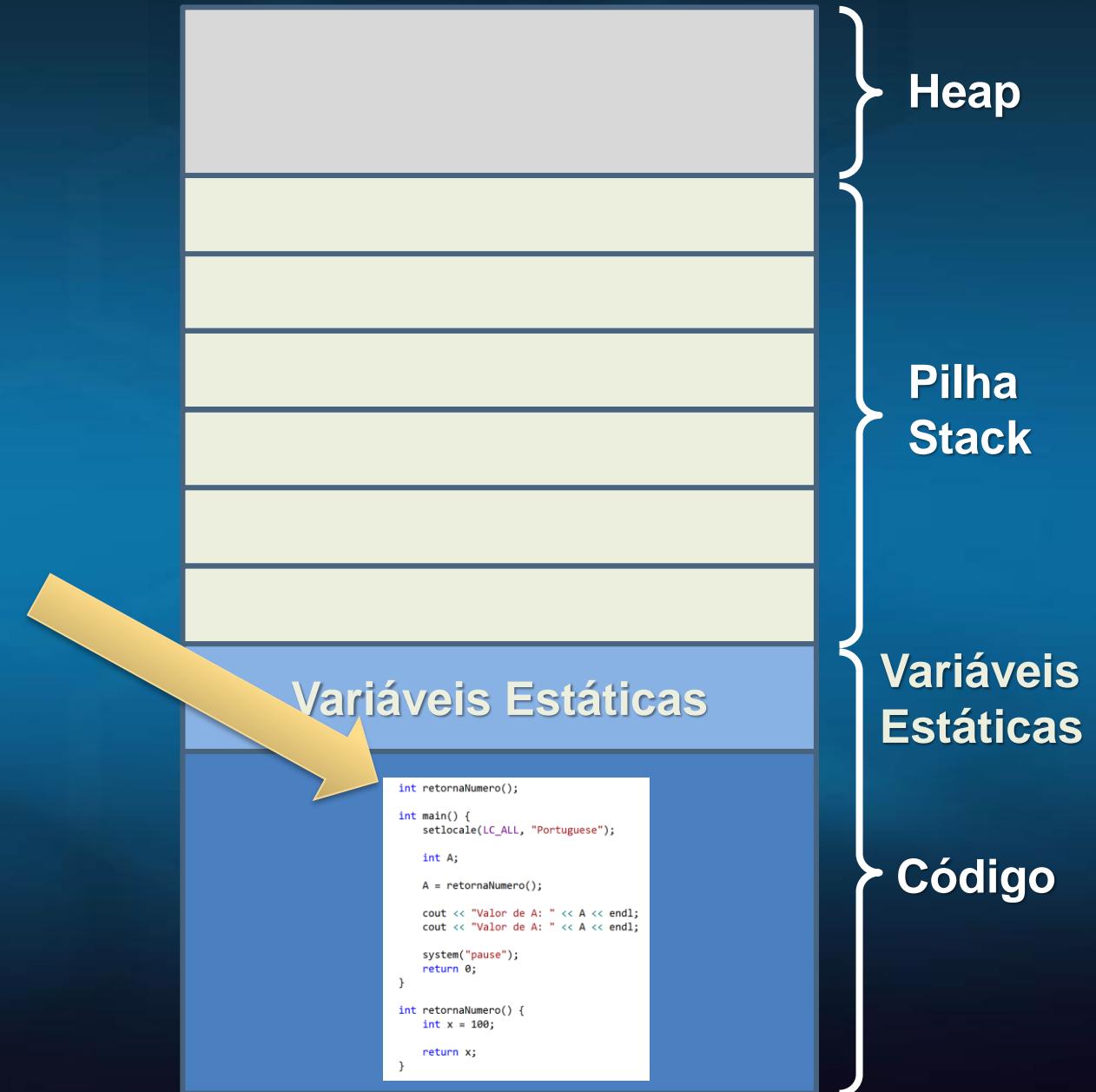
    A = retornaNumero();

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

    system("pause");
    return 0;
}

int retornaNumero() {
    int x = 100;

    return x;
}
```



# Problema

```
int retornaNumero();

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

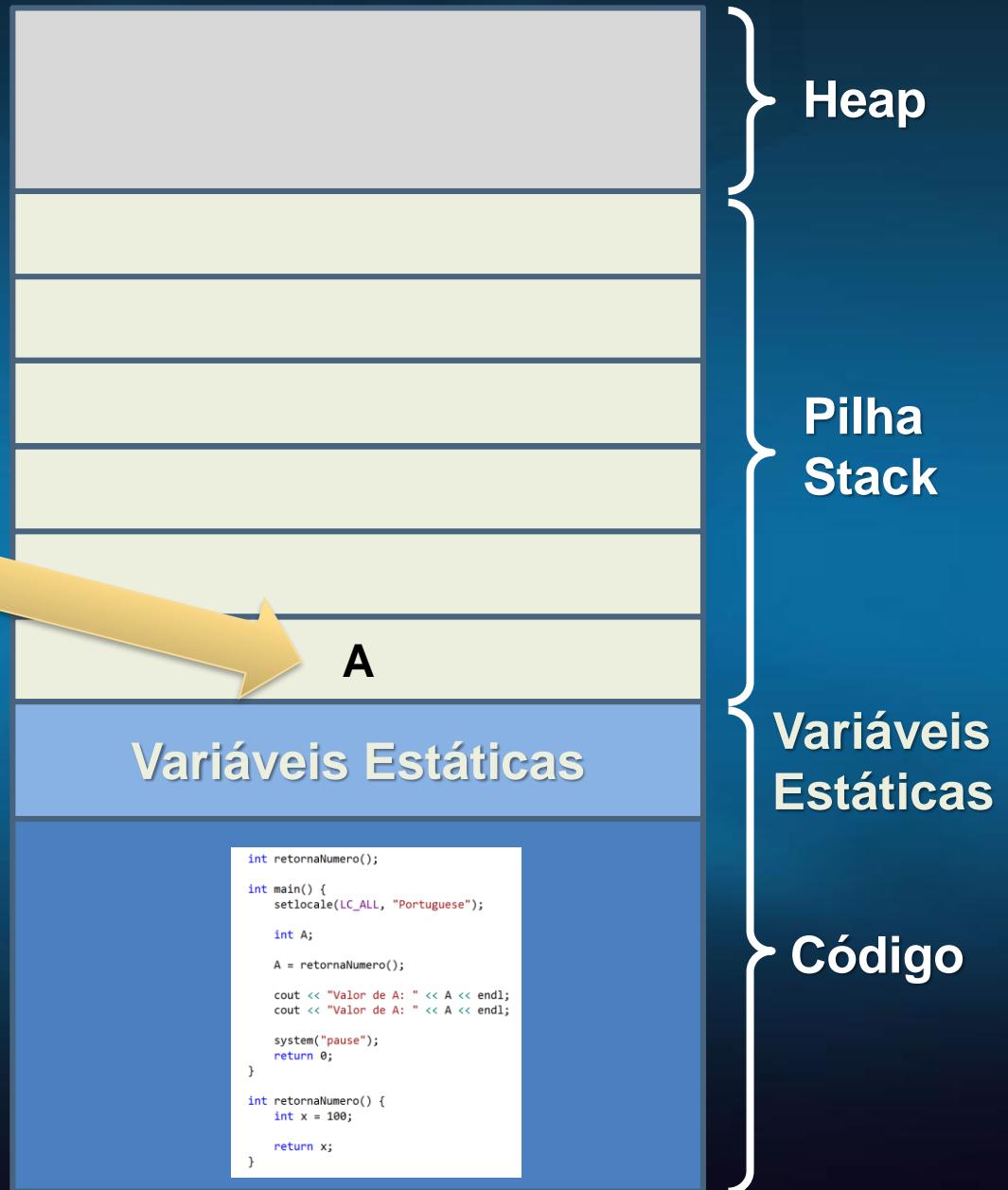
    int A;
    A = retornaNumero();

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

    system("pause");
    return 0;
}

int retornaNumero() {
    int x = 100;

    return x;
}
```



# Problema

```
int retornaNumero();

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

    int A;

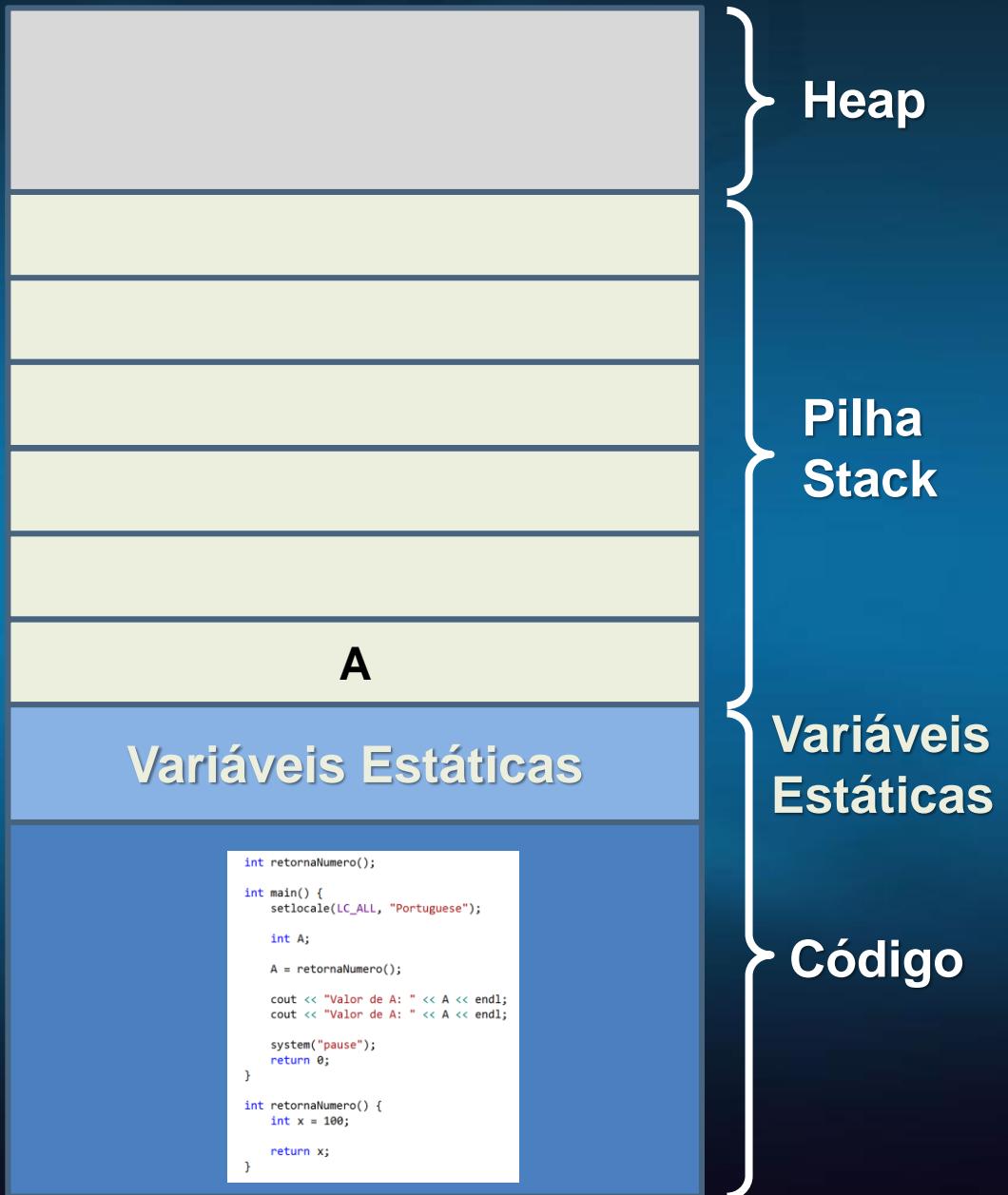
    A = retornaNumero();

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

    system("pause");
    return 0;
}

int retornaNumero() {
    int x = 100;

    return x;
}
```



# Problema

```
int retornaNumero();

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

    int A;

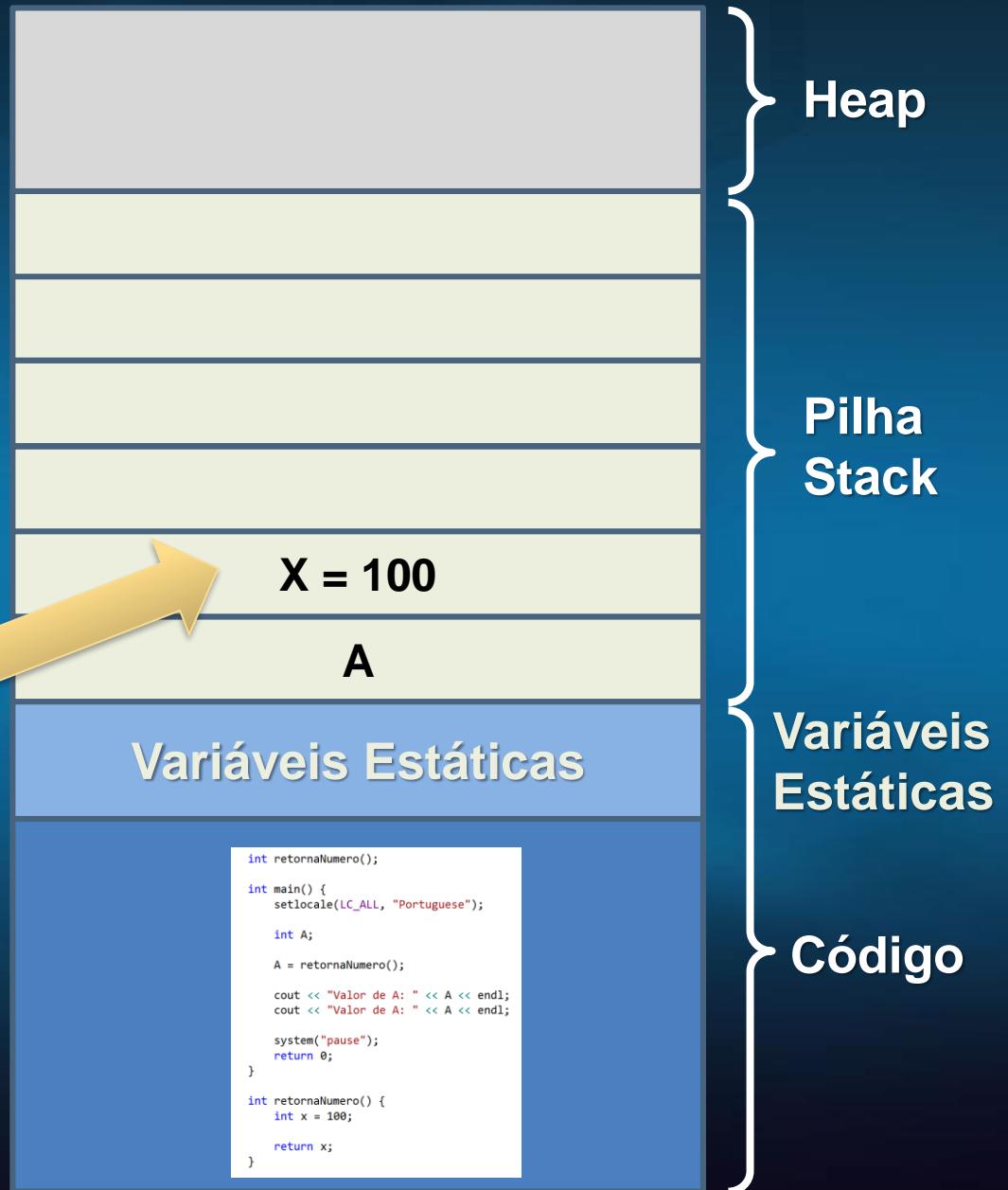
    A = retornaNumero();

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

    system("pause");
    return 0;
}

int retornaNumero() {
    int x = 100;

    return x;
}
```



# Problema

```
int retornaNumero();

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

    int A;

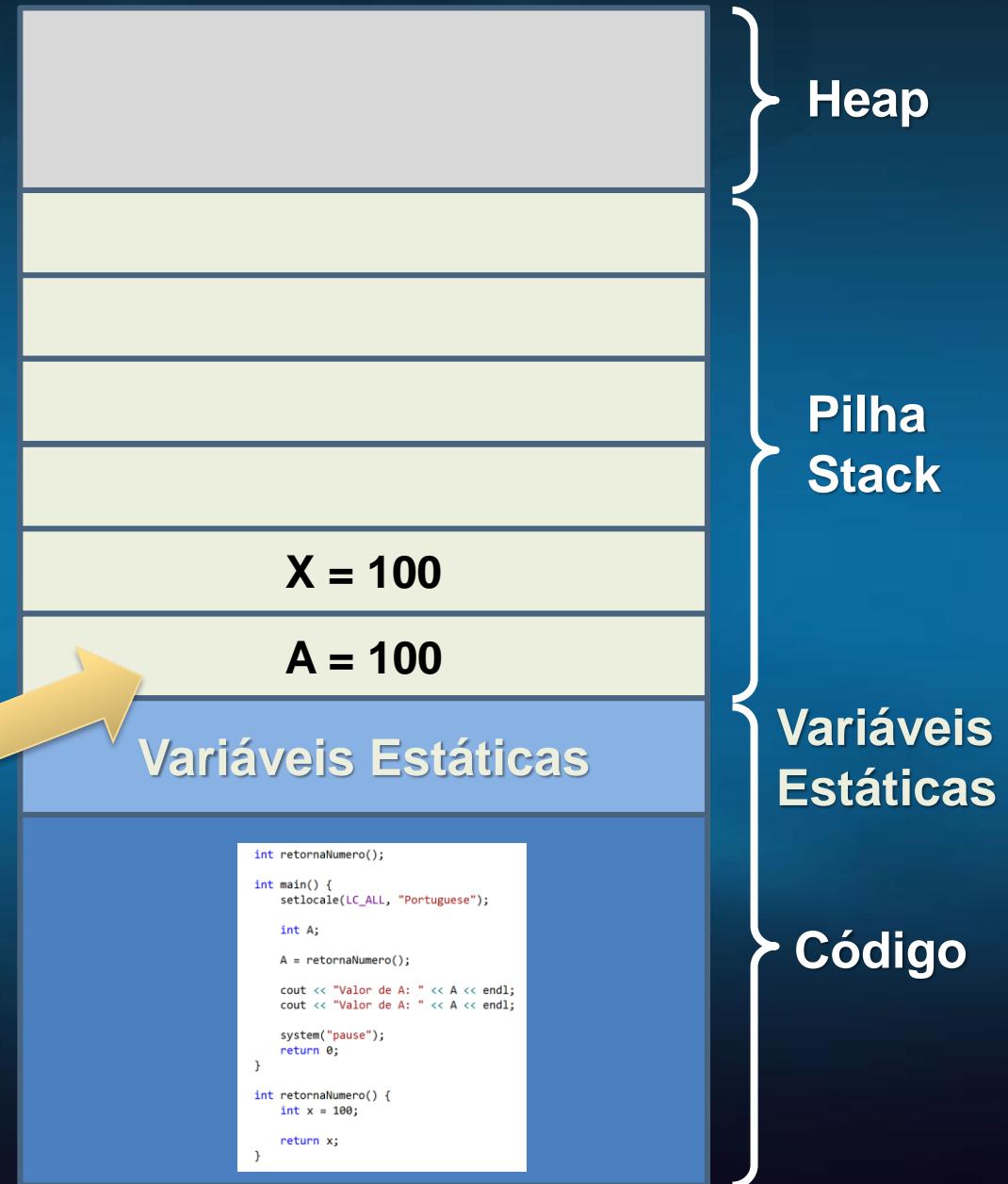
    A = retornaNumero();

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

    system("pause");
    return 0;
}

int retornaNumero() {
    int x = 100;

    return x;
}
```



# Problema

```
int retornaNumero();

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

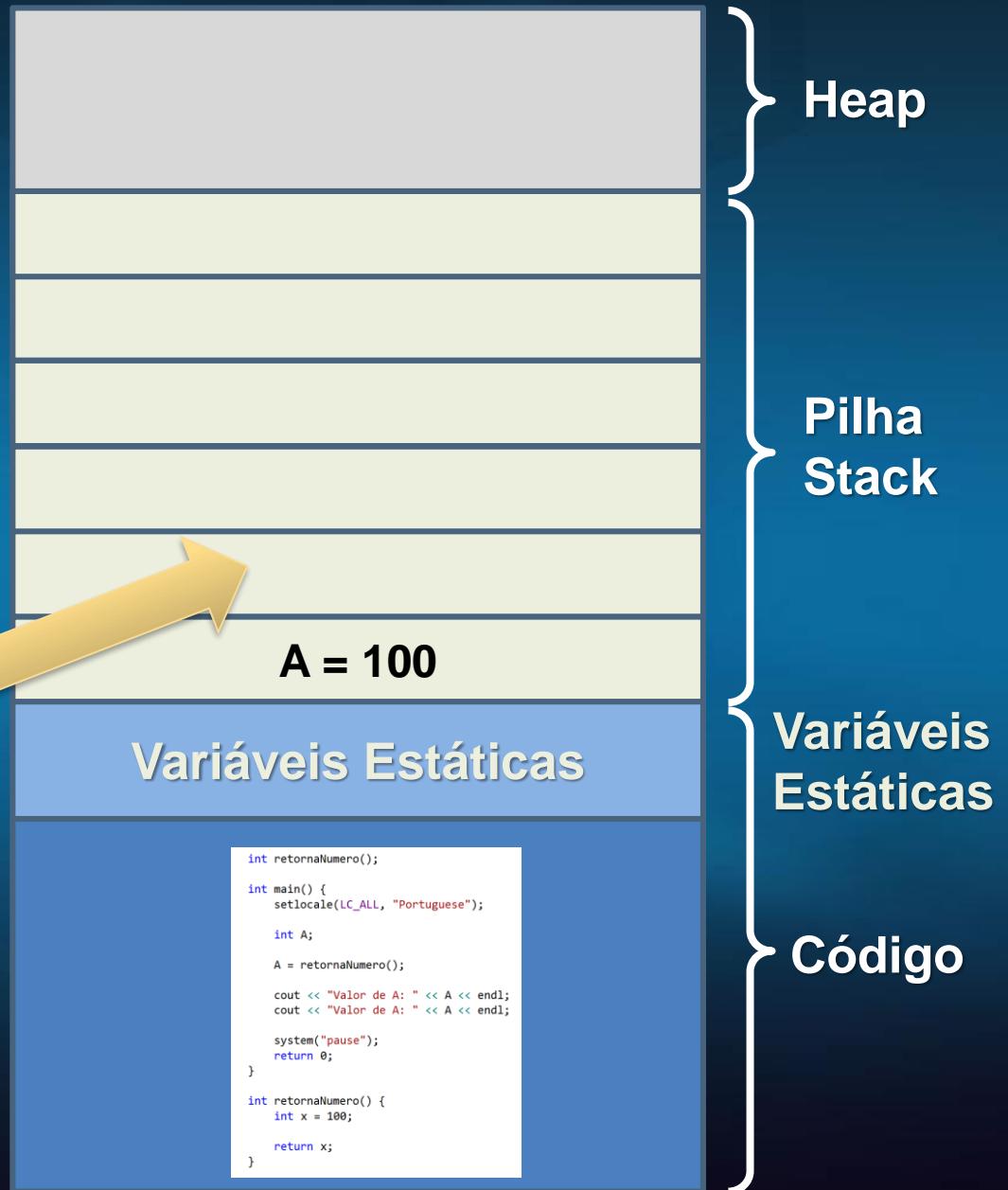
    int A;

    A = retornaNumero();

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

    system("pause");
    return 0;
}

int retornaNumero() {
    int x = 100;
    return x;
}
```



# Problema

```
int retornaNumero();

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

    int A;

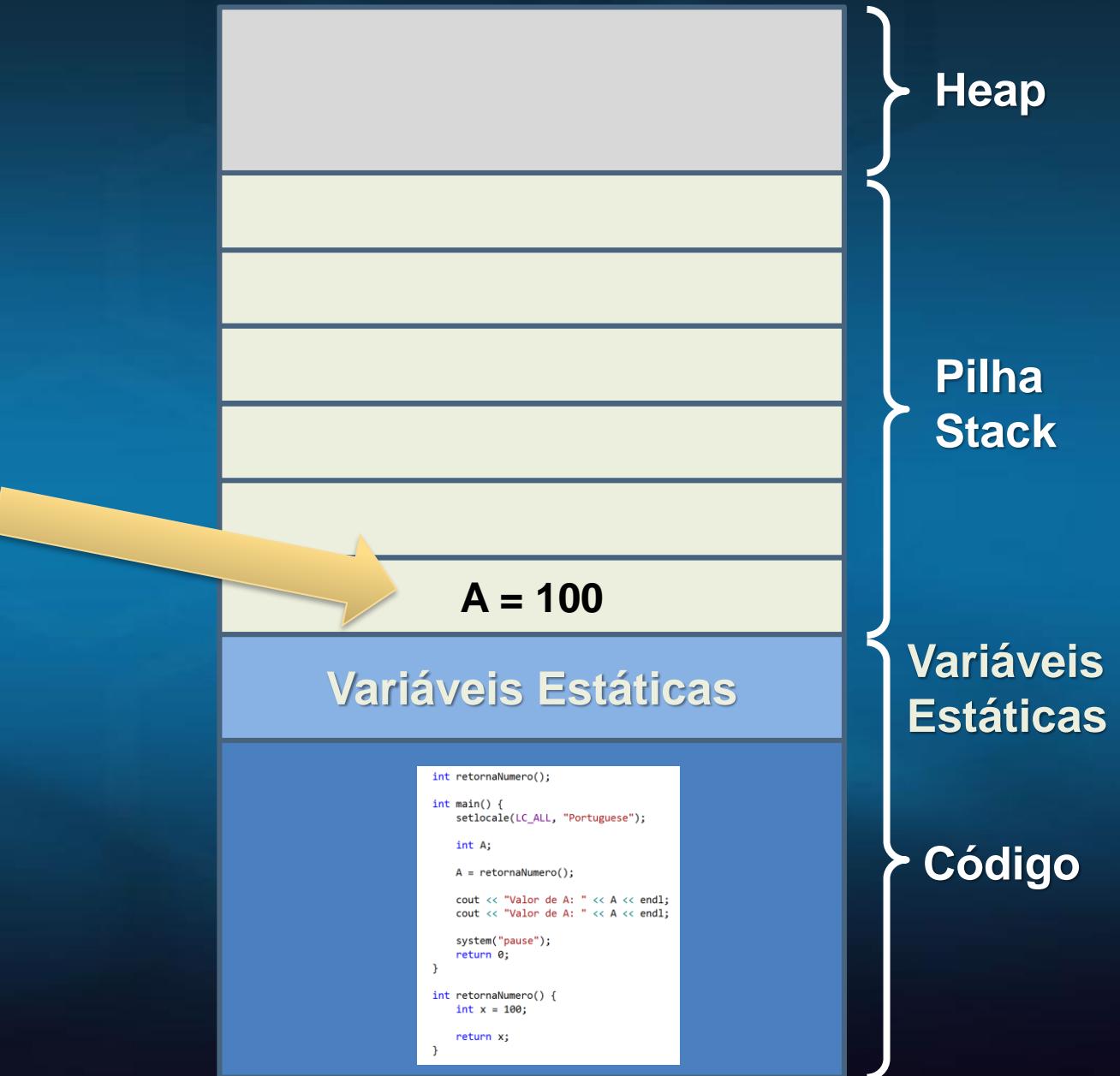
    A = retornaNumero();

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

    system("pause");
    return 0;
}

int retornaNumero() {
    int x = 100;

    return x;
}
```



# Variáveis estáticas

## • Retorno por REFERÊNCIA

```
int* retornaNumero();

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

    int *pA;

    pA = retornaNumero();

    cout << "Valor de *pA: " << *pA << endl;
    cout << "Valor de *pA: " << *pA << endl;

    system("pause");
    return 0;
}

int* retornaNumero() {
    int x = 100;

    return &x;
}
```



# Variáveis estáticas

```
int* retornaNumero();

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

    int *pA;

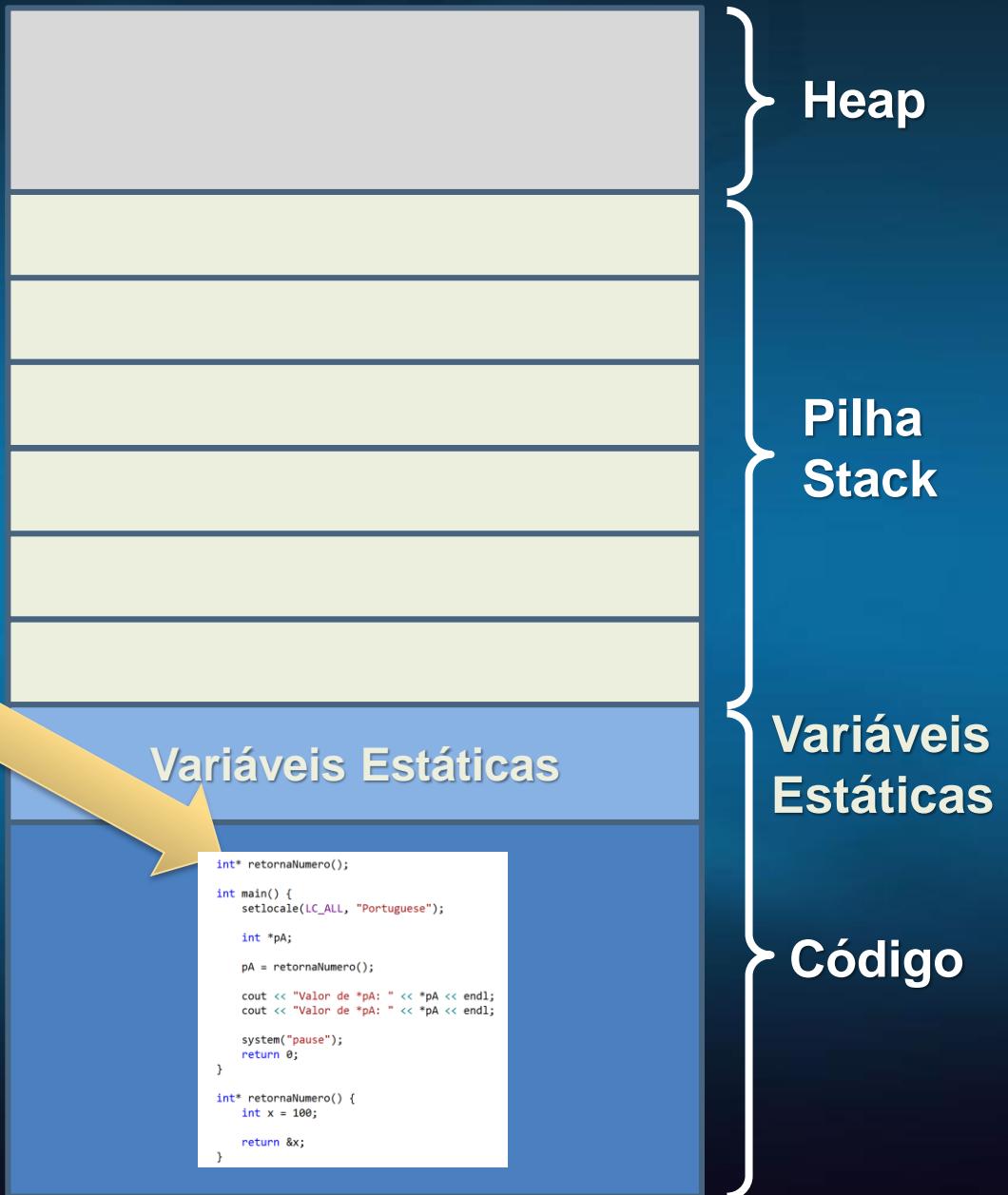
    pA = retornaNumero();

    cout << "Valor de *pA: " << *pA << endl;
    cout << "Valor de *pA: " << *pA << endl;

    system("pause");
    return 0;
}

int* retornaNumero() {
    int x = 100;

    return &x;
}
```



# Variáveis estáticas

```
int* retornaNumero();

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

    int *pA;

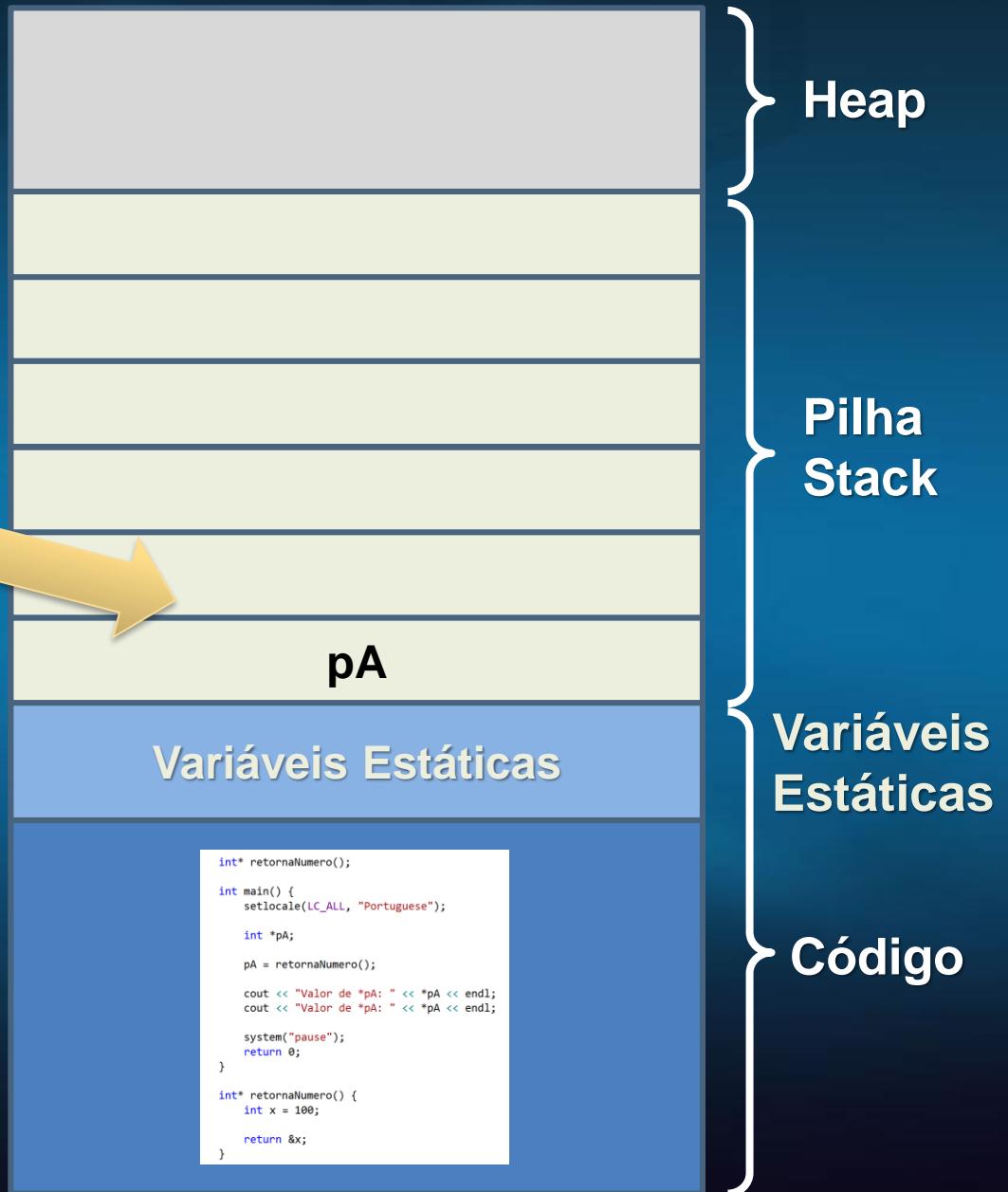
    pA = retornaNumero();

    cout << "Valor de *pA: " << *pA << endl;
    cout << "Valor de *pA: " << *pA << endl;

    system("pause");
    return 0;
}

int* retornaNumero() {
    int x = 100;

    return &x;
}
```



# Variáveis estáticas

```
int* retornaNumero();

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

    int *pA;

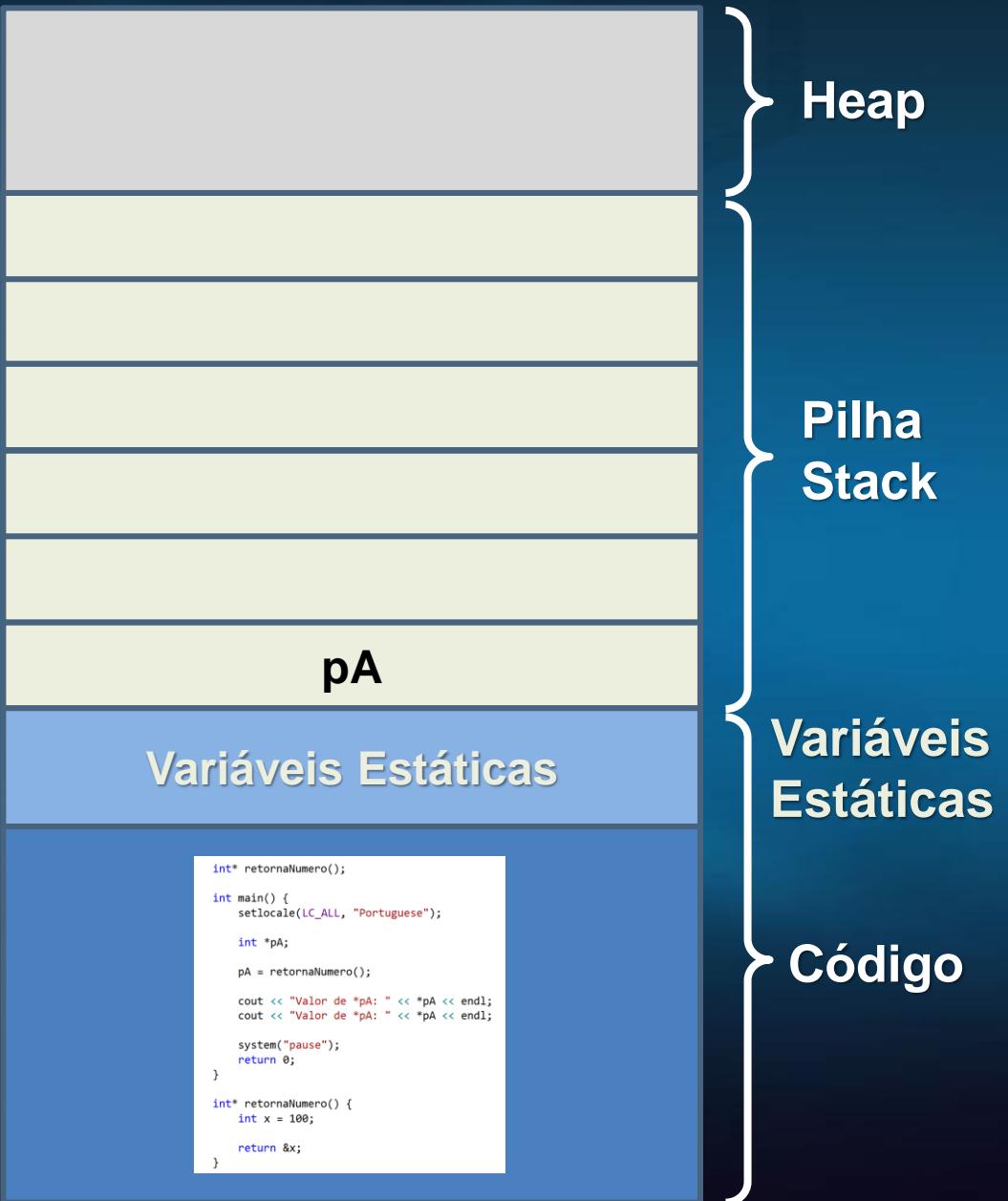
    pA = retornaNumero();  

    cout << "Valor de *pA: " << *pA << endl;
    cout << "Valor de *pA: " << *pA << endl;

    system("pause");
    return 0;
}

int* retornaNumero() {  
    int x = 100;

    return &x;
}
```



# Variáveis estáticas

```
int* retornaNumero();

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

    int *pA;

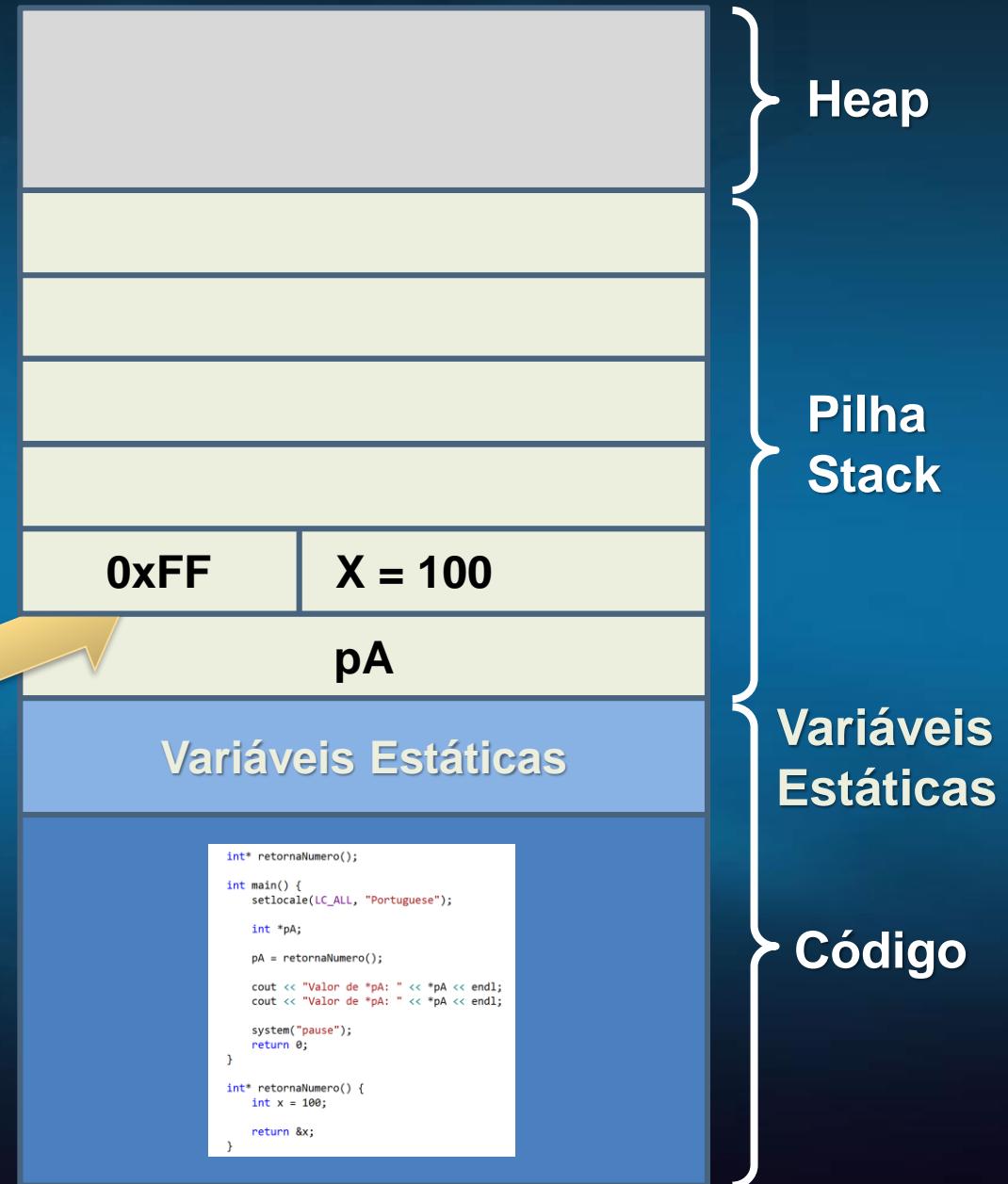
    pA = retornaNumero();

    cout << "Valor de *pA: " << *pA << endl;
    cout << "Valor de *pA: " << *pA << endl;

    system("pause");
    return 0;
}

int* retornaNumero()
{
    int x = 100;

    return &x;
}
```



# Variáveis estáticas

```
int* retornaNumero();

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

    int *pA;

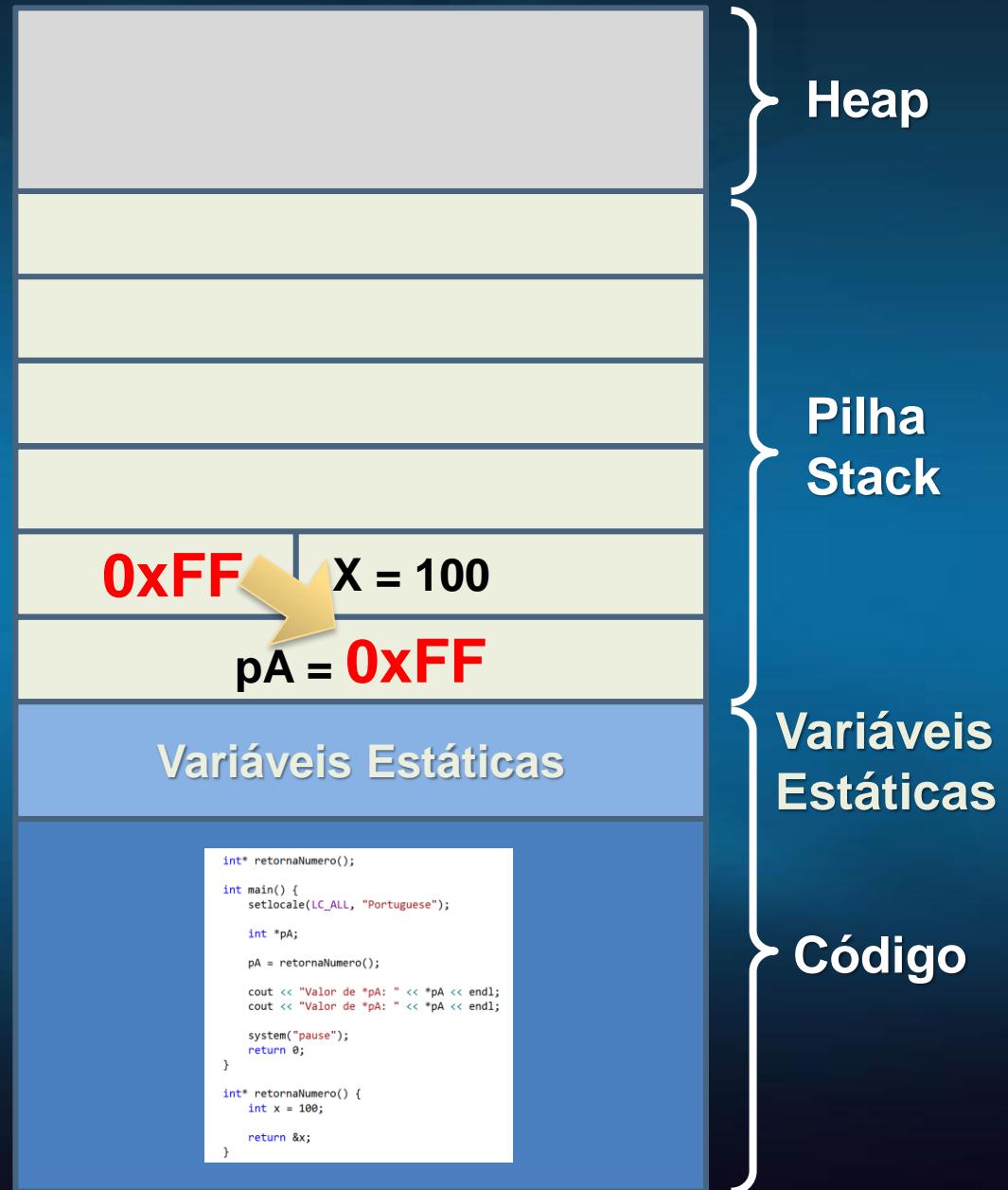
    pA = retornaNumero();

    cout << "Valor de *pA: " << *pA << endl;
    cout << "Valor de *pA: " << *pA << endl;

    system("pause");
    return 0;
}

int* retornaNumero() {
    int x = 100;

    return &x;
}
```



# Variáveis estáticas

```
int* retornaNumero();

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

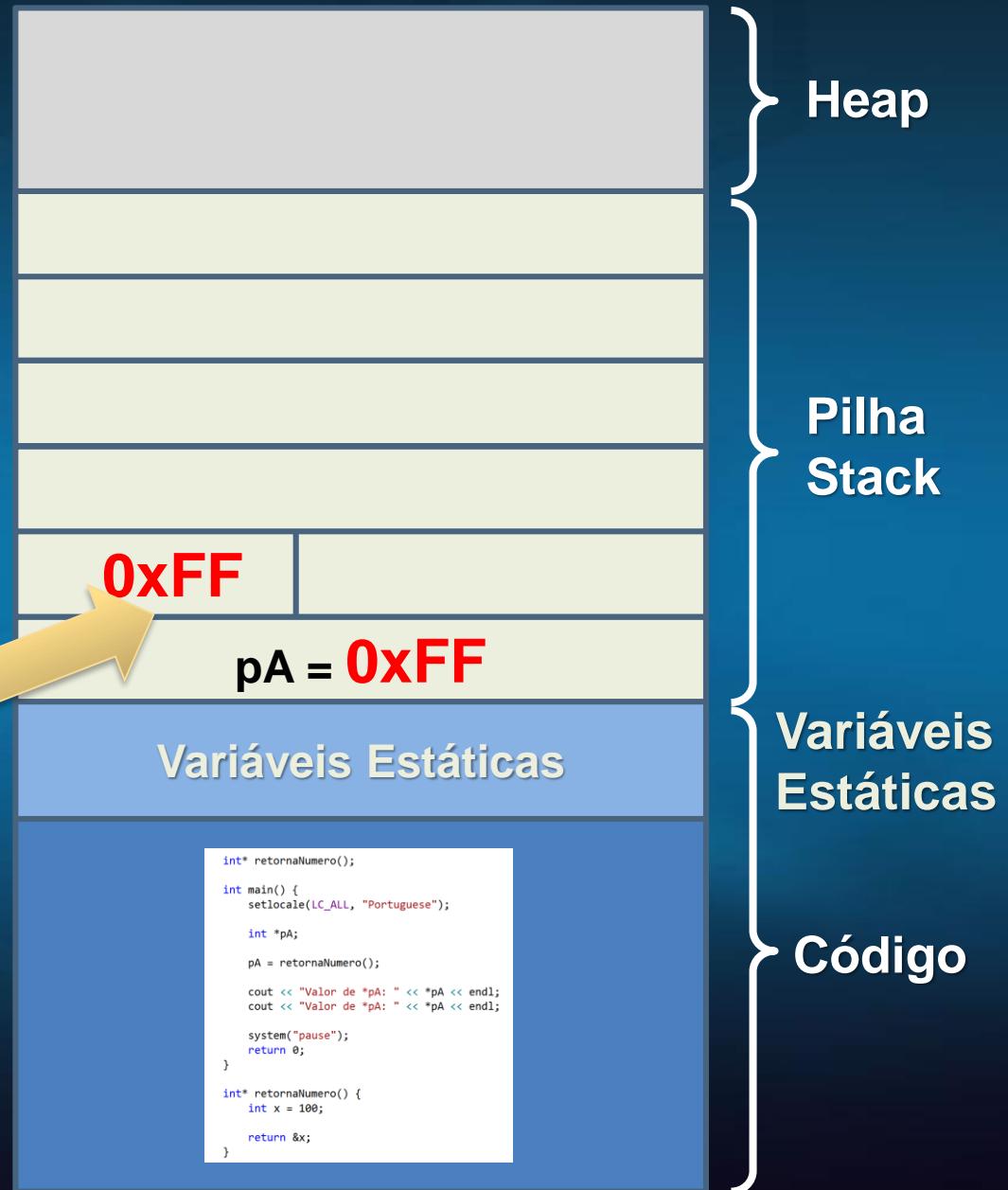
    int *pA;

    pA = retornaNumero();

    cout << "Valor de *pA: " << *pA << endl;
    cout << "Valor de *pA: " << *pA << endl;

    system("pause");
    return 0;
}

int* retornaNumero() {
    int x = 100;
    return &x;
}
```



# Variáveis estáticas

```
int* retornaNumero();

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

    int *pA;

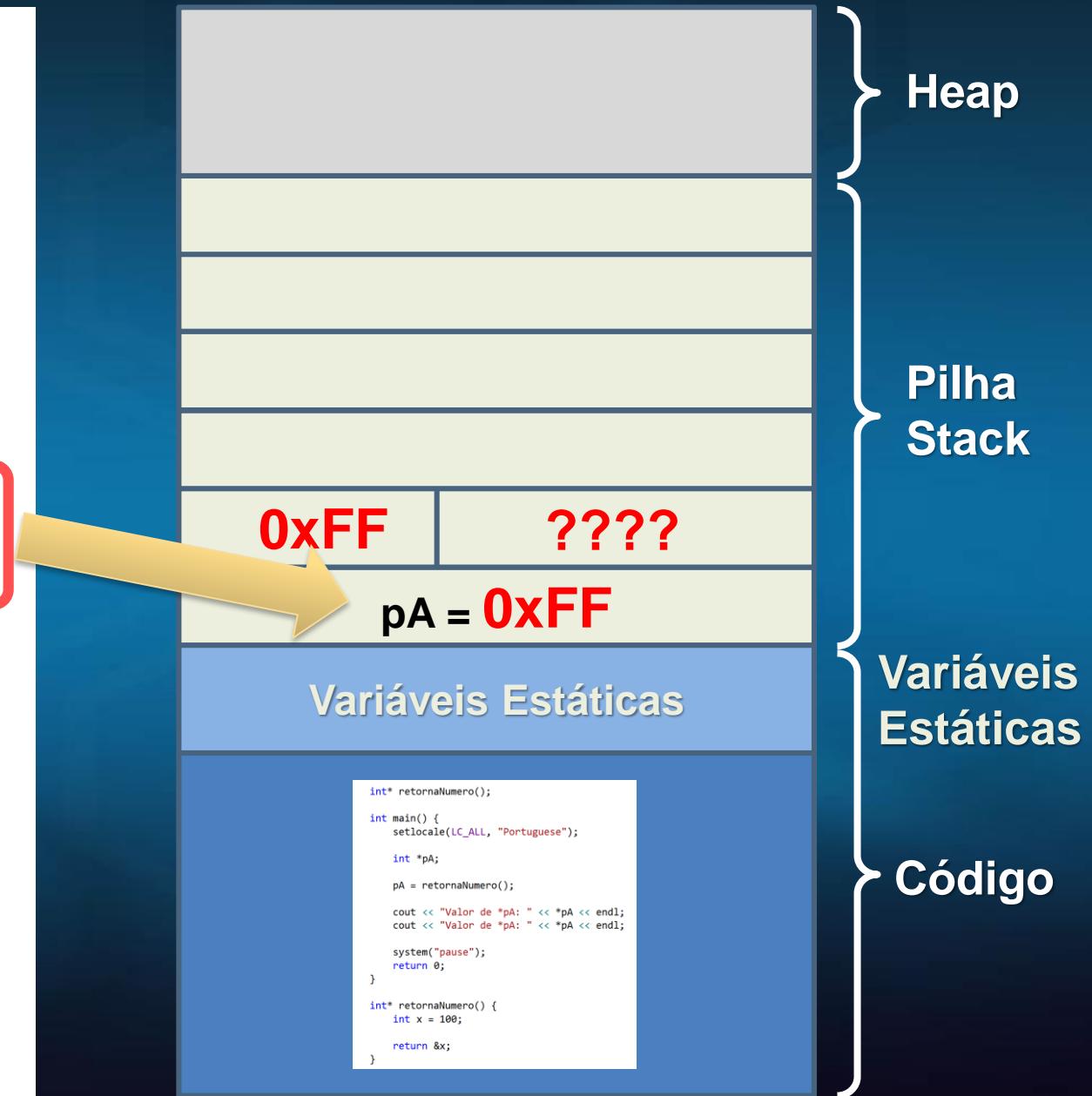
    pA = retornaNumero();

    cout << "Valor de *pA: " << *pA << endl;
    cout << "Valor de *pA: " << *pA << endl;

    system("pause");
    return 0;
}

int* retornaNumero() {
    int x = 100;

    return &x;
}
```



# Variáveis estáticas

- Variáveis locais **estáticas** são variáveis cujo valor é mantido na memória de uma chamada da função para a outra.

```
int* retornaNumero();

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

    int *pA;

    pA = retornaNumero();

    cout << "Valor de *pA: " << *pA << endl;
    cout << "Valor de *pA: " << *pA << endl;

    system("pause");
    return 0;
}

int* retornaNumero() {
    static int x = 100;

    return &x;
}
```

# Variáveis estáticas

```
int* retornaNumero();

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

    int *pA;

    pA = retornaNumero();

    cout << "Valor de *pA: " << *pA << endl;
    cout << "Valor de *pA: " << *pA << endl;

    system("pause");
    return 0;
}

int* retornaNumero() {
    static int x = 100;

    return &x;
}
```



# Variáveis estáticas

```
int* retornaNumero();

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

    int *pA;

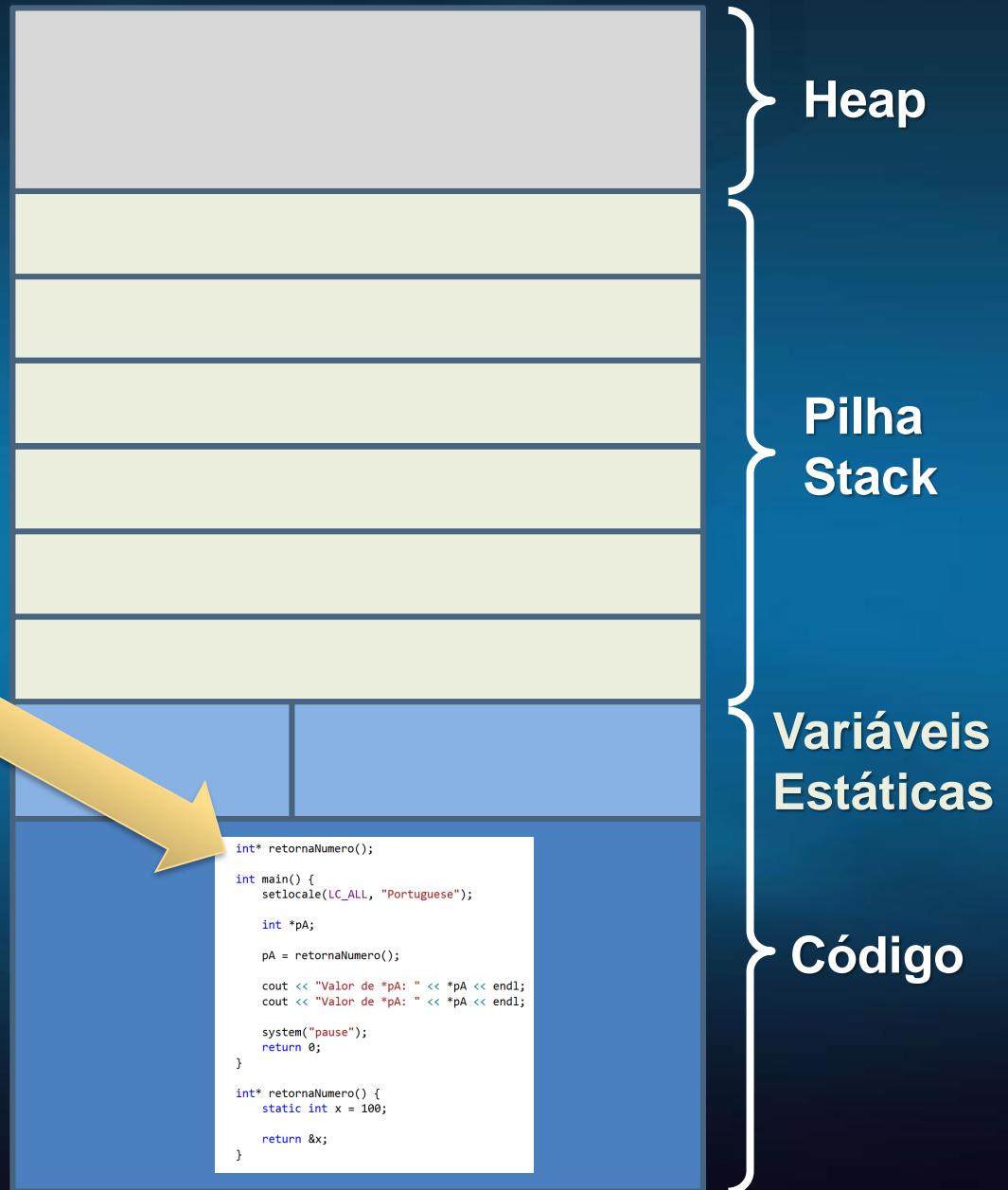
    pA = retornaNumero();

    cout << "Valor de *pA: " << *pA << endl;
    cout << "Valor de *pA: " << *pA << endl;

    system("pause");
    return 0;
}

int* retornaNumero() {
    static int x = 100;

    return &x;
}
```



# Variáveis estáticas

```
int* retornaNumero();

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

    int *pA;

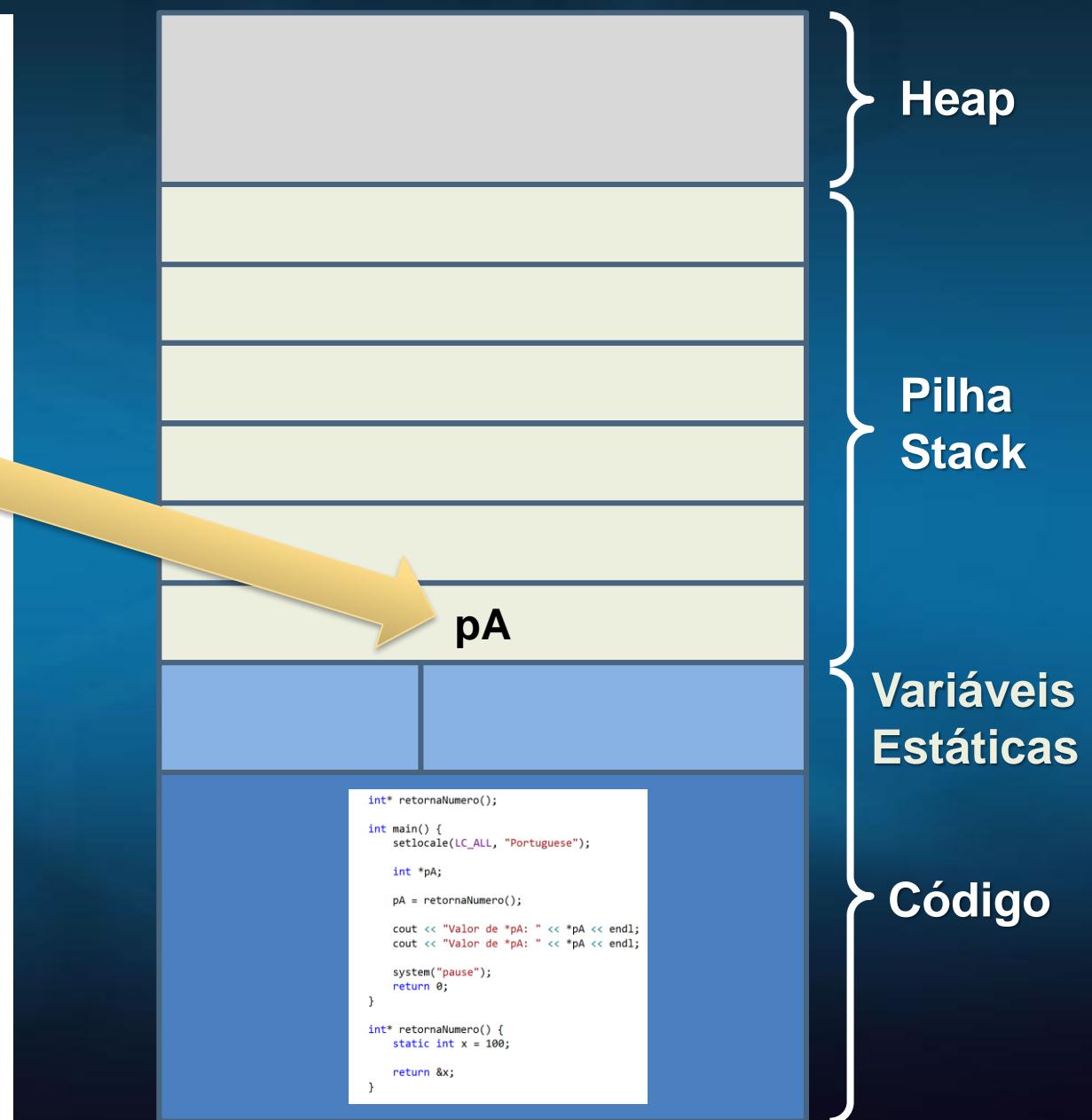
    pA = retornaNumero();

    cout << "Valor de *pA: " << *pA << endl;
    cout << "Valor de *pA: " << *pA << endl;

    system("pause");
    return 0;
}

int* retornaNumero() {
    static int x = 100;

    return &x;
}
```



# Variáveis estáticas

```
int* retornaNumero();

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

    int *pA;

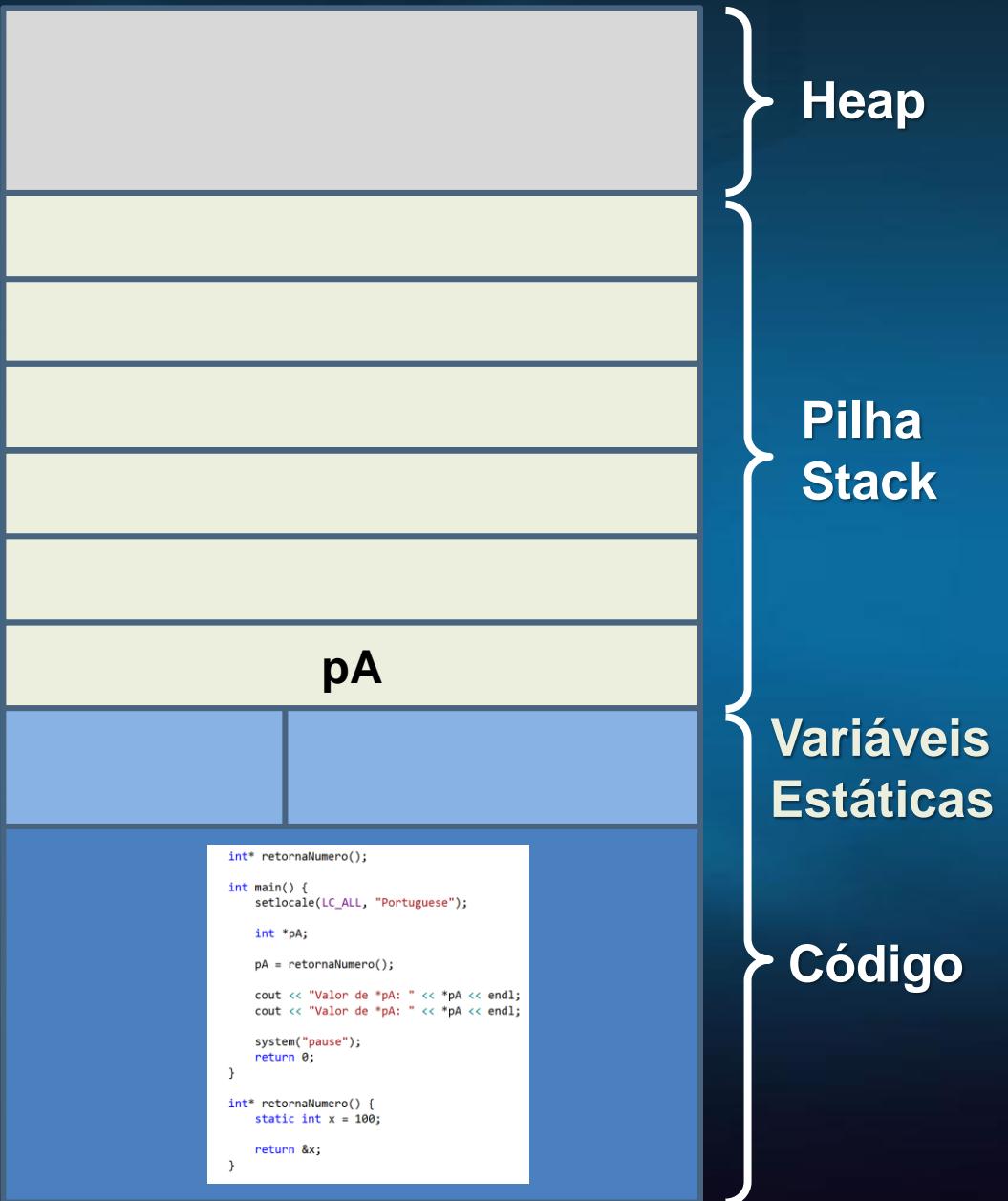
    pA = retornaNumero(); // A

    cout << "Valor de *pA: " << *pA << endl;
    cout << "Valor de *pA: " << *pA << endl;

    system("pause");
    return 0;
}

int* retornaNumero() { // B
    static int x = 100;

    return &x;
}
```



# Variáveis estáticas

```
int* retornaNumero();

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

    int *pA;

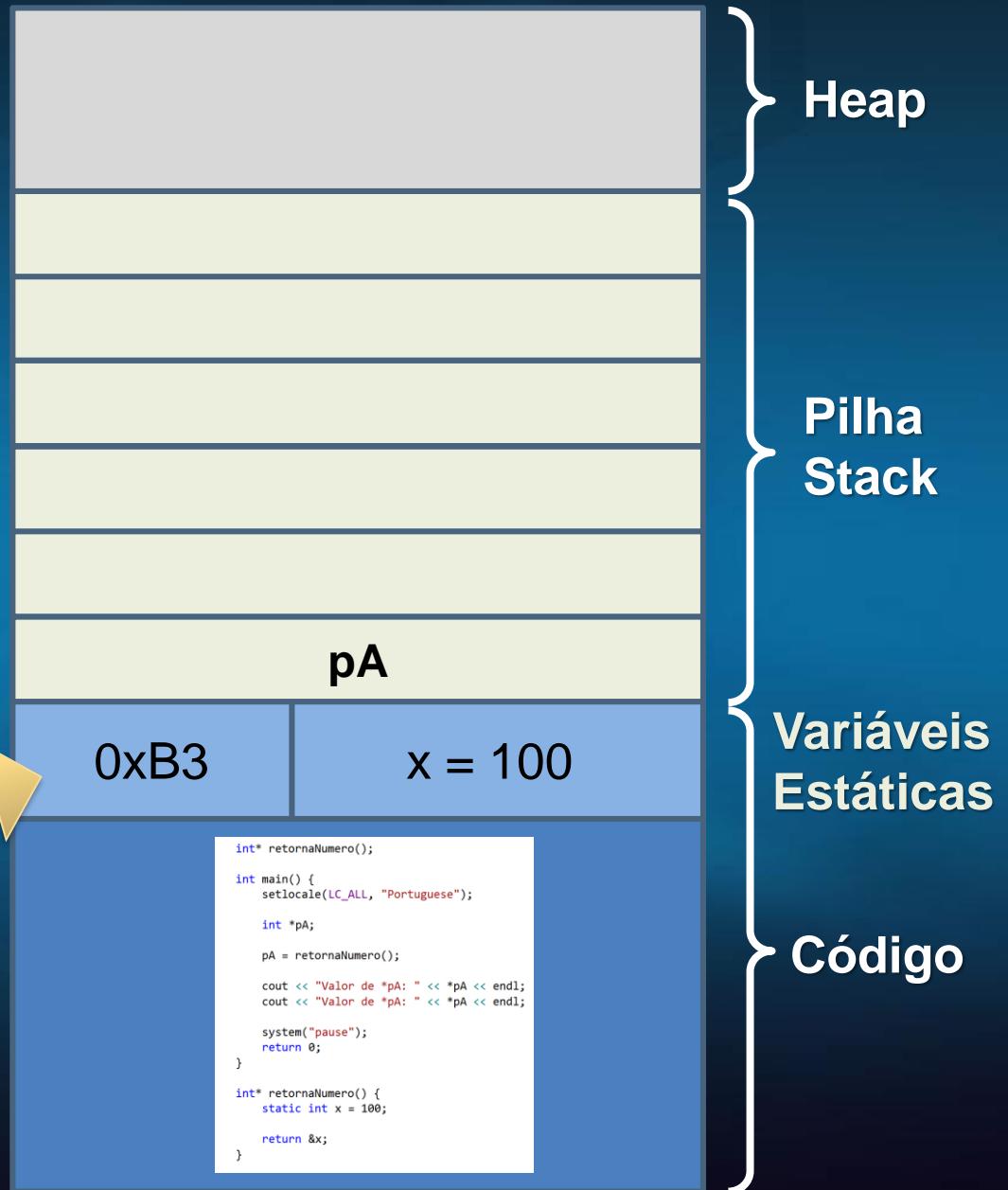
    pA = retornaNumero();

    cout << "Valor de *pA: " << *pA << endl;
    cout << "Valor de *pA: " << *pA << endl;

    system("pause");
    return 0;
}

int* retornaNumero() {
    static int x = 100;

    return &x;
}
```



# Variáveis estáticas

```
int* retornaNumero();

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

    int *pA;

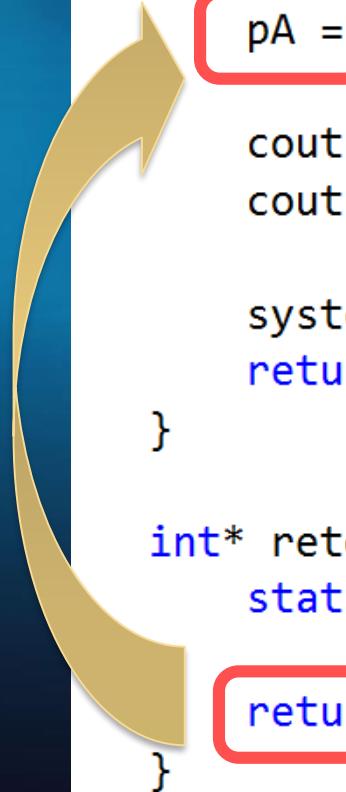
    pA = retornaNumero();

    cout << "Valor de *pA: " << *pA << endl;
    cout << "Valor de *pA: " << *pA << endl;

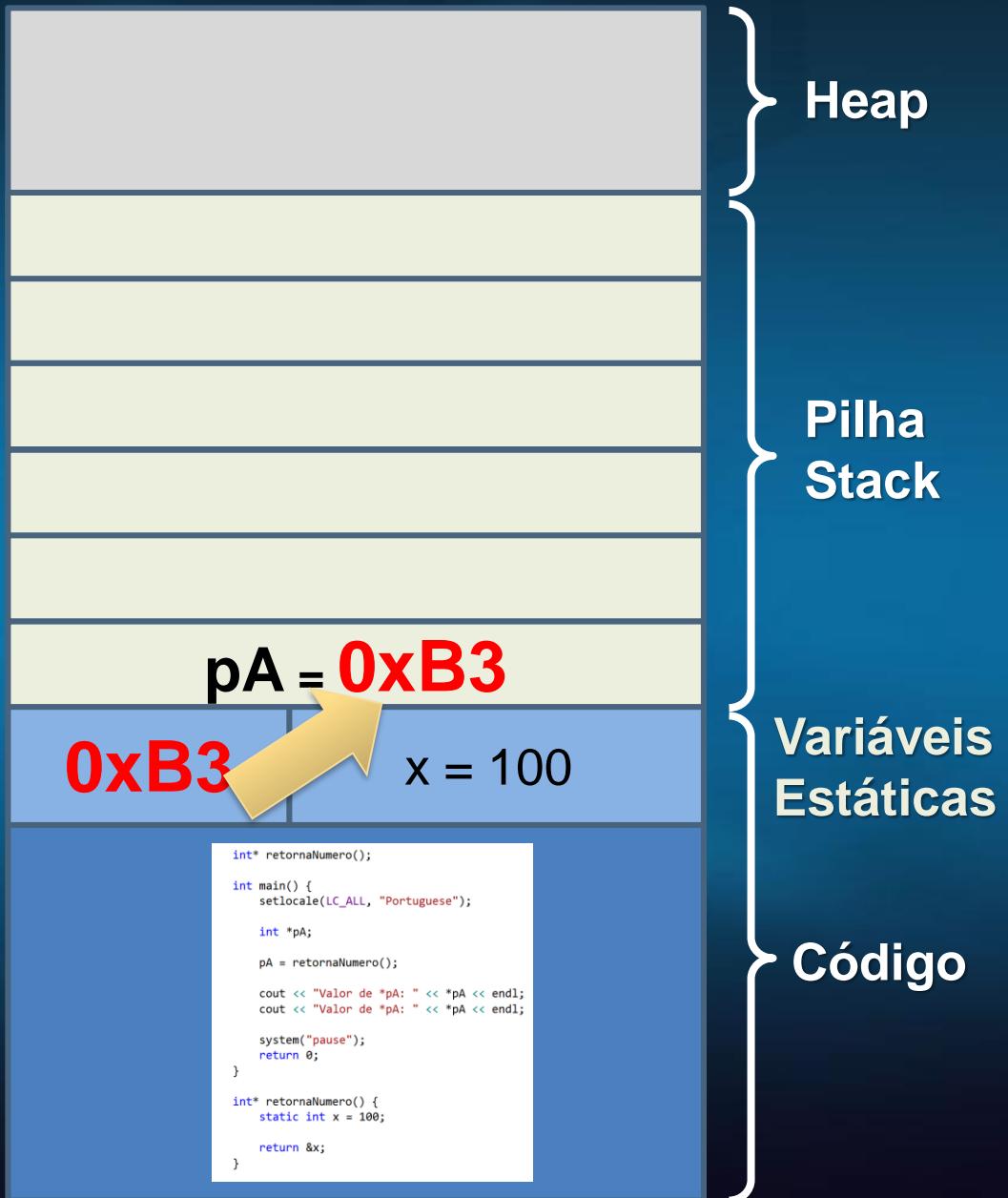
    system("pause");
    return 0;
}

int* retornaNumero() {
    static int x = 100;

    return &x;
}
```



A large yellow arrow on the left side of the slide points from the bottom right towards the code block, highlighting the flow of control.



# Variáveis estáticas

```
int* retornaNumero();

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

    int *pA;

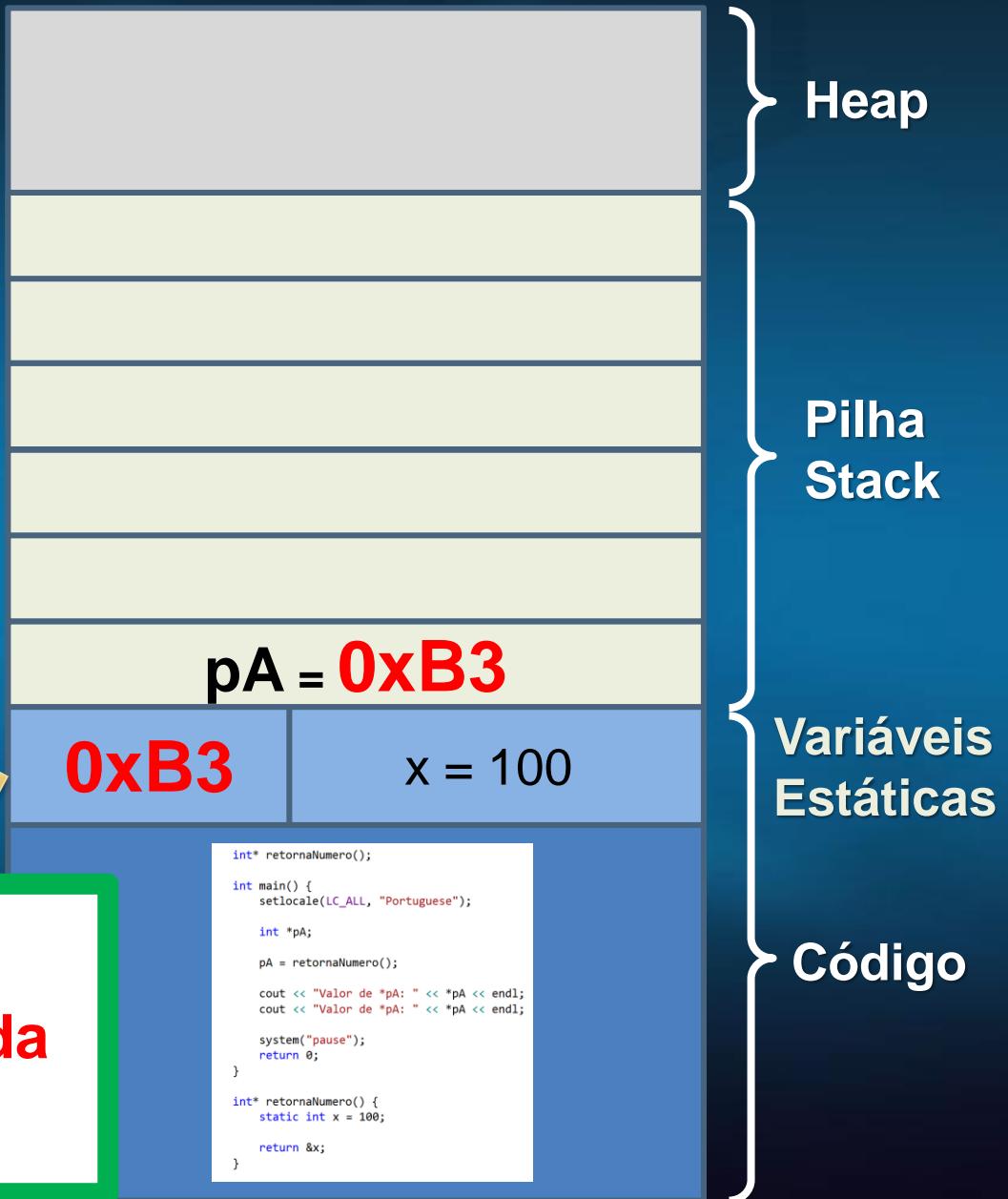
    pA = retornaNumero();

    cout << "Valor de *pA: " << *pA << endl;
    cout << "Valor de *pA: " << *pA << endl;

    system("pause");
    return 0;
}

int* retornaNumero() {
    static int x = 100;
    return &x;
}
```

Variáveis estáticas  
não são apagadas da  
memória



# Variáveis estáticas

```
int* retornaNumero();

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

    int *pA;

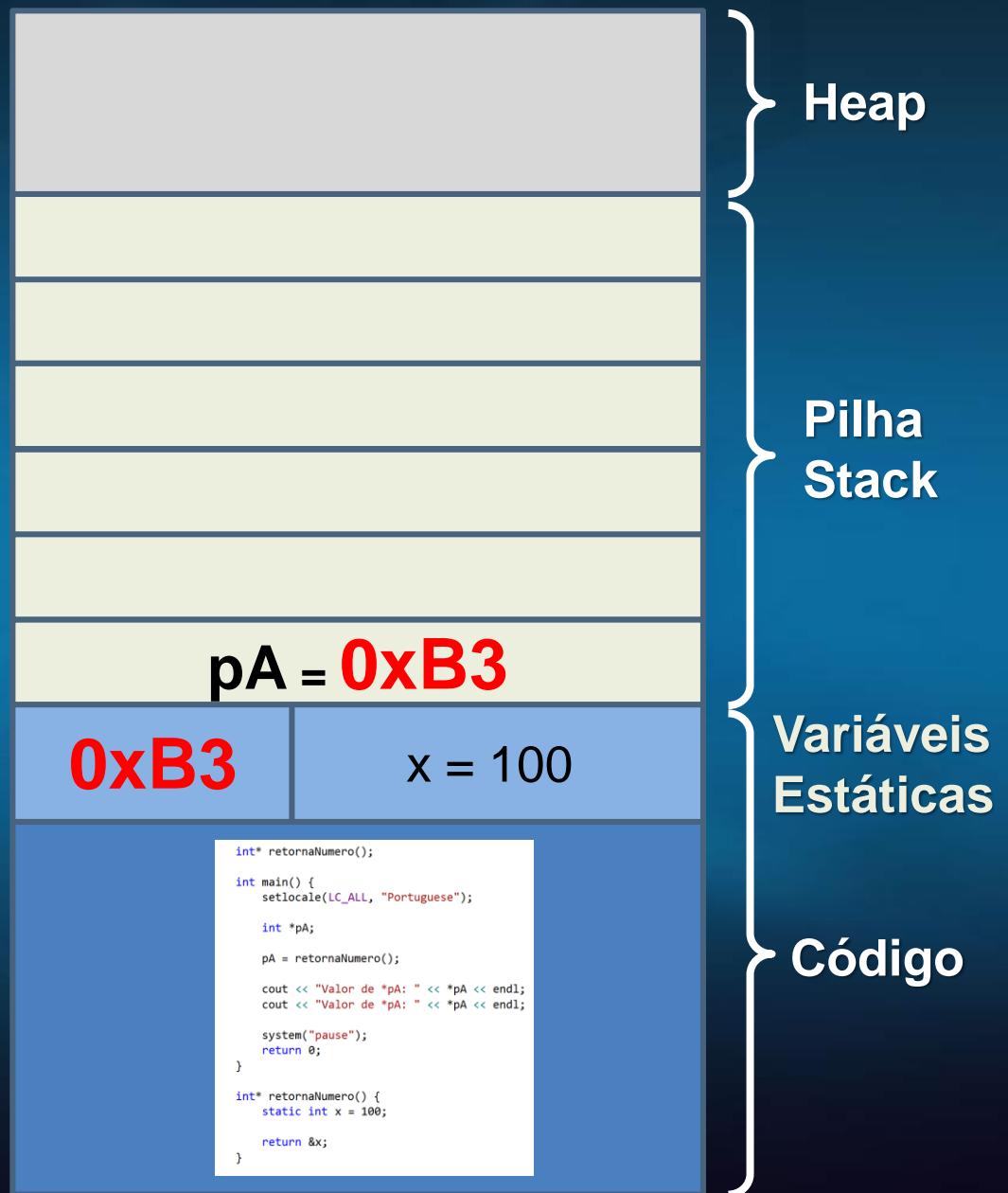
    pA = retornaNumero();

    cout << "Valor de *pA: " << *pA << endl;
    cout << "Valor de *pA: " << *pA << endl;

    system("pause");
    return 0;
}

int* retornaNumero() {
    static int x = 100;

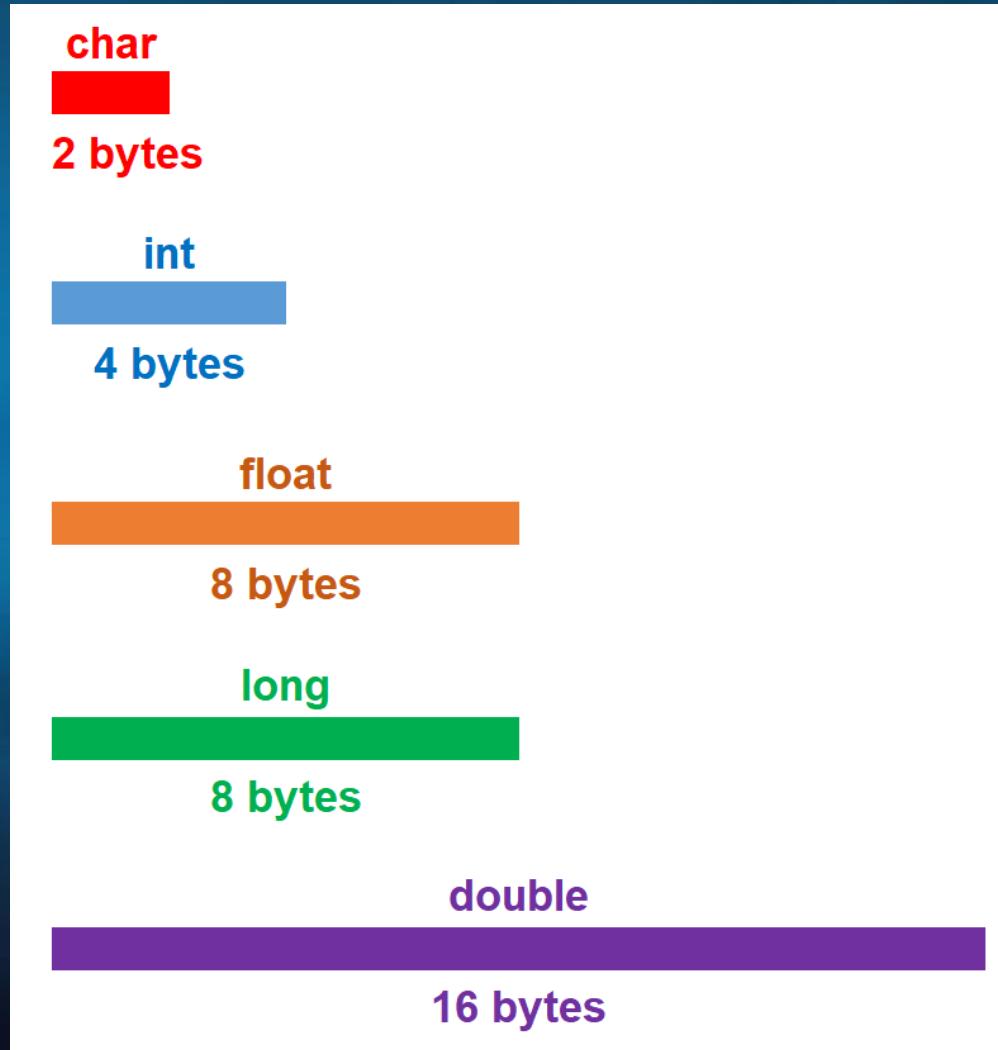
    return &x;
}
```



# Aritmética Com Ponteiros

Apenas duas operações aritméticas podem ser utilizadas nos endereços armazenados pelos ponteiros:

- Adição
- Subtração



# Aritmética Com Ponteiros

```
int main() {
    int a;
    int *pA;

    pA = &a;

    cout << "Conteúdo de pA: " << pA << endl;

    pA = pA + 1;
    cout << "Conteúdo de pA: " << pA << endl;

    pA = pA + 1;
    cout << "Conteúdo de pA: " << pA << endl;

    pA = pA - 1;
    cout << "Conteúdo de pA: " << pA << endl;

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

# Ponteiros e Vetores

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

    float nota[4] = { 10.0, 8.5, 9.0, 9.5 };
    float *pNota;
    float media;

    pNota = nota;

    cout << "Nota 1: " << *(pNota + 0) << endl;
    cout << "Nota 2: " << *(pNota + 1) << endl;
    cout << "Nota 3: " << *(pNota + 2) << endl;
    cout << "Nota 4: " << *(pNota + 3) << endl << endl << endl;

    // Alterar valor
    *(pNota + 0) = 10.0;
    *(pNota + 1) = 10.0;
    *(pNota + 2) = 10.0;
    *(pNota + 3) = 10.0;

    media = (*(pNota + 0) + *(pNota + 1) + *(pNota + 2) + *(pNota + 3)) / 4;
    cout << "Média: " << media << endl;

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

# AULA E PROVA

**PARA TODOS OS EXERCÍCIOS  
(EM AULA OU PROVA)**

**SEMPRE QUE MANUSEAR PONTEIROS,  
UTILIZE APENAS E SOMENTE  
NOTAÇÃO DE PONTEIROS.**

# Exercício

- Faça um programa que declare um vetor do tipo inteiro de 5 posições e atribua valores a ele utilizando o cin.
- Em seguida crie e inicialize o ponteiro ptr com o vetor.
- Exiba os valores do ponteiro ptr.

# Exercício

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

    int valores[5];
    int *pValores;

    cout << "Digite o valor 1: ";
    cin >> valores[0];

    cout << "Digite o valor 2: ";
    cin >> valores[1];

    cout << "Digite o valor 3: ";
    cin >> valores[2];

    cout << "Digite o valor 4: ";
    cin >> valores[3];

    cout << "Digite o valor 5: ";
    cin >> valores[4];
```

```
    pValores = valores;

    cout << "Valor 1: " << *(pValores + 0) << endl;
    cout << "Valor 2: " << *(pValores + 1) << endl;
    cout << "Valor 3: " << *(pValores + 2) << endl;
    cout << "Valor 4: " << *(pValores + 3) << endl;
    cout << "Valor 5: " << *(pValores + 4) << endl << endl;

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

# Exercício

- Crie um programa que tenha um vetor para armazenar três temperaturas de paciente de uma clínica.
- O vetor deve ser atribuído a um ponteiro.
- Utilizando CIN, atribua valores ao ponteiro e em seguida exiba os dados.

# Exercício

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

    float temp[3];
    float *pTemp;

    pTemp = temp;

    cout << "Digite a temperatura 1: ";
    cin >> *(pTemp + 0);

    cout << "Digite a temperatura 2: ";
    cin >> *(pTemp + 1);

    cout << "Digite a temperatura 3: ";
    cin >> *(pTemp + 2);

    cout << "Valor 1: " << *(pTemp + 0) << endl;
    cout << "Valor 2: " << *(pTemp + 1) << endl;
    cout << "Valor 3: " << *(pTemp + 2) << endl;
    system("pause");
    return 0;
}
```

# Exercício

- Crie uma função que retorne um ponteiro para um vetor de três posições. O vetor deve guardar a altura de três irmão. Solicite que o usuário informe as alturas na tela. Após o retorno da função exiba os dados do ponteiro.

Ex.: float\* altura( );

# Exercício – Parte 1

```
float* altura();

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

    float *pAltura;
    int ind;

    pAltura = altura();

    for (ind = 0; ind < 3; ind++) {
        cout << "Altura " << ind +1 << ":" << *(pAltura + ind) << endl;
    }

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

# Exercício – Parte 2

```
float* altura() {  
    static float altura[3];  
  
    cout << "Digite o altura da pessoa 1: ";  
    cin >> altura[0];  
  
    cout << "Digite o altura da pessoa 2: ";  
    cin >> altura[1];  
  
    cout << "Digite o altura da pessoa 3: ";  
    cin >> altura[2];  
  
    return altura;  
}
```

# Ponteiros e Estruturas

```
struct Pessoa
{
    string nome;
    int idade;
    float salario;
};
```

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

    Pessoa aluno;
    Pessoa *pAluno;

    pAluno = &aluno;

    pAluno->nome = "Maria José";
    pAluno->idade = 25;
    pAluno->salario = 3000.00f;

    cout << "Nome: " << pAluno->nome << endl;
    cout << "Idade: " << pAluno->idade << endl;
    cout << "Salário: " << pAluno->salario << endl << endl;

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

# Passando estruturas para funções

```
struct Pessoa
{
    string nome;
    int idade;
    float salario;
};

void exibirDados(Pessoa *pA);
```

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

    Pessoa aluno;
    Pessoa *pAluno;

    pAluno = &aluno;

    pAluno->nome = "Maria José";
    pAluno->idade = 25;
    pAluno->salario = 3000.00f;

    exibirDados(pAluno);

    system("pause");
    return 0;
}

void exibirDados(Pessoa *pA) {

    cout << endl << endl;
    cout << "Nome: " << pA->nome << endl;
    cout << "Idade: " << pA->idade << endl;
    cout << "Salário: " << pA->salario << endl << endl;
}
```

# Exercício

- Crie um programa que tenha uma estrutura para guardar dados de veículos: modelo, ano de fabricação e número de portas.
- Crie uma variável desta estrutura e passe-a para uma função.
- Utilizando CIN, a função deverá preencher os valores da estrutura.
- Ao retornar da função, imprima os dados fornecidos pelo usuário.

# Exercício – Parte 1

```
struct Veiculo  
{  
    string modelo;  
    int anoFabricacao;  
    int nroPortas;  
};
```

```
int main() {  
    setlocale(LC_ALL, "Portuguese");  
  
    Veiculo carro;  
  
    obterDados(&carro);  
  
    cout << endl << endl;  
    cout << "Modelo: " << carro.modelo << endl;  
    cout << "Ano de fabricação: " << carro.anoFabricacao << endl;  
    cout << "Nro Portas.: " << carro.nroPortas << endl;  
  
    system("pause");  
    return 0;  
}
```

# Exercício – Parte 2

```
void obterDados(Veiculo *pCar) {
    cout << "Digite o modelo do carro: ";
    cin >> pCar->modelo;

    cout << "Digite o ano de fabricação do carro: ";
    cin >> pCar->anoFabricacao;

    cout << "Digite o número de portas do carro: ";
    cin >> pCar->nroPortas;
}
```

# Exercício

- Altere o exercício anterior, passando o estrutura, após receber os dados de retorno da função, para uma função que imprima o seu conteúdo.

# Exercício

```
struct Veiculo
{
    string modelo;
    int anoFabricacao;
    int nroPortas;
};

void obterDados(Veiculo *pCar);
void imprimirDados(Veiculo *pCar);

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

    Veiculo carro;

    obterDados(&carro);

    imprimirDados(&carro);

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

```
void obterDados(Veiculo *pCar) {
    cout << "Digite o modelo do carro: ";
    cin >> pCar->modelo;

    cout << "Digite o ano de fabricação do carro: ";
    cin >> pCar->anoFabricacao;

    cout << "Digite o número de portas do carro: ";
    cin >> pCar->nroPortas;
}

void imprimirDados(Veiculo *pCar) {
    cout << endl << endl;
    cout << "Modelo: " << pCar->modelo << endl;
    cout << "Ano de fabricação: " << pCar->anoFabricacao << endl;
    cout << "Nro Portas.: " << pCar->nroPortas << endl;
}
```

# Exercício

- Crie um programa que tenha uma função cujo retorno seja um ponteiro para uma estrutura produto.  
Ex.: Produto\* obterDados();
- A estrutura deve conter: código do produto, descrição e preço.
- A função deve solicitar ao usuários os dados da estrutura.
- Após o retorno da função, imprima os dados do ponteiro.

# Exercício

```
struct Produto {  
    int codProd;  
    string desc;  
    float preco;  
};  
  
Produto* obterDados();  
  
int main() {  
    setlocale(LC_ALL, "Portuguese");  
  
    Produto *pProd;  
  
    pProd = obterDados();  
  
    cout << "Código:" << pProd->codProd << endl;  
    cout << "Descrição:" << pProd->desc << endl;  
    cout << "Preço:" << pProd->preco << endl;  
  
    system("pause");  
    return 0;  
}
```

```
Produto* obterDados() {  
    static Produto prod;  
  
    cout << "Digite o código do produto: ";  
    cin >> prod.codProd;  
  
    cout << "Digite a descrição do produto: ";  
    cin >> prod.desc;  
  
    cout << "Digite o preço do produto: ";  
    cin >> prod.preco;  
  
    return &prod;  
}
```

# Exercício

- Altere o exercício anterior, passando o ponteiro, após receber os dados de retorno da função, para uma função que imprima o seu conteúdo.

# Exercício

```
struct Produto {  
    int codProd;  
    string desc;  
    float preco;  
};  
  
Produto* obterDados();  
void imprimirDados(Produto* prod);  
  
int main() {  
    setlocale(LC_ALL, "Portuguese");  
  
    Produto *pProd;  
  
    pProd = obterDados();  
  
    imprimirDados(pProd);  
  
    system("pause");  
    return 0;  
}
```

```
Produto* obterDados() {  
    static Produto prod;  
  
    cout << "Digite o código do produto: ";  
    cin >> prod.codProd;  
  
    cout << "Digite a descrição do produto: ";  
    cin >> prod.desc;  
  
    cout << "Digite o preço do produto: ";  
    cin >> prod.preco;  
  
    return &prod;  
}  
  
void imprimirDados(Produto* prod) {  
  
    cout << "Código:" << prod->codProd << endl;  
    cout << "Descrição:" << prod->desc << endl;  
    cout << "Preço:" << prod->preco << endl;  
}
```

# 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](mailto:ricbalieiro@gmail.com)