BCC702 - Programação de Computadores II

Escopo de variáveis, funções e passagem de parâmetros

Prof José Romildo Malaquias
Profª Valéria de Carvalho Santos

UFOP

decom
departamento de computação

Escopo de variável

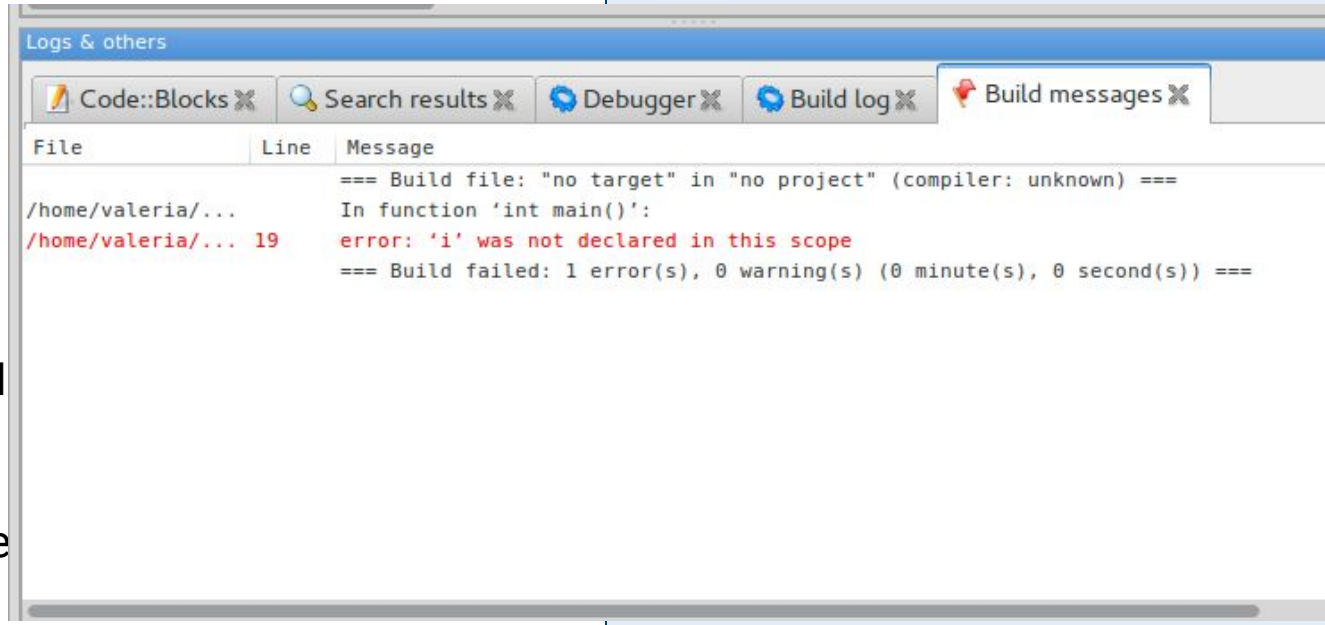
- Em C++, um bloco de comandos é delimitado por { }

- Faça um programa para ler N valores reais e verificar qual é o maior.

```
#include <iostream>
#include <math.h>
using namespace std;

int main () {
int n;
double num, maior = INFINITY;
cout << “Informe o valor de N: ”;
cin >> n;
for(int i = 0; i < n; i++){
	cout << “Digite um número real: ”;
		cin >> num;
		if(num > maior){
			maior = num;
		}
	}
	cout << “Maior: ” << maior;
return 0;
}
```

```
#include <iostream>
#include <math.h>
using namespace std;

int main () {
int n;
double num, maior = INFINITY;
cout << “Informe o valor de N: ”;
cin >> n;
for(int i = 0; i < n; i++){
        cout << “Digite um número real: ”;
                cin >> num;
                if(num > maior){
                        maior = num;
                }
        }
        cout << “Maior: ” << maior;
return 0;
}
```

```
#include <iostream>
#include <math.h>
using namespace std;

int main () {
int n;
float num, maior = INFINITY;
cout << “Informe o valor de N
cin >> n;
for(int i = 0; i < n; i++){
	cout << “Digite um núme
		cin >> num;
		if(num > maior){
			maior = num;
		}
	}
	cout << “i: ” << i << endl;
	cout << “Maior: ” << maior;
return 0;
}
```



Funções

- Exemplo: Faça um programa para calcular o calor de $2x^a+4y^b+z^c$, onde os valores de x, y, z, a, b e c são dados de entrada.

- Como você resolveria esse problema se não conhecesse a função *pow*?

Funções

- Podemos criar nossas próprias funções

- Sintaxe:

```
tipoRetorno nomeFuncao(tipo parametro1, tipo parametro2...) {

//corpo da função

return expressãoTipoRetorno;
}
```

Funções

- Exemplo: função para calcular potência

```
int potencia(int x, int a) {

        int pot = 1;
        for (int i = 0; i < a; i++)
                pot = pot * x;

return pot;
}
```

```
#include <iostream>
using namespace std;

int potencia(int x, int a) {
        int pot = 1;
        for (int i = 0; i < a; i++)
                pot = pot*x;
        return pot;
}

int main() {
        int x, y, z, a, b, c, p1, p2, p3;
        cin >> x >> y >> z;
        cin >> a >> b >> c;
        p1 = potencia(x,a);
        p2 = potencia(y,b);
        p3 = potencia(z,c);
        cout << 2*p1+4*p2+p3 << endl;
        return 0;
}
```

```
#include <iostream>
using namespace std;

int potencia(int x, int a) {
        int pot = 1;
        for (int i = 0; i < a; i++)
                pot = pot*x;
return pot;
}

int main() {
        int x, y, z, a, b, c, p1, p2, p3;
        cin >> x >> y >> z;
        cin >> a >> b >> c;
        p1 = potencia(x, a);
        p2 = potencia(y, b);
        p3 = potencia(z, c);
        cout << 2*p1 + 4*p2 + p3 << endl;
        return 0;
}
```

Definição da função

Chamada da função

Funções

- Observações:
  - O início de qualquer programa em C++ é a execução da função main
  - As outras funções só são executadas quando acontece uma chamada
  - Uma função pode ser chamada no corpo de outras funções, não apenas da função main
  - Após o **return**, a função é encerrada retornando o valor indicado. Nenhum comando é executado após o **return**

```
#include <iostream>
using namespace std;

void imprimir() {
        cout << “Potencia calculada”;
}

int potencia(int x, int a) {
        int pot = 1;
        for(int i = 0; i < a; i++)
                pot = pot*x;
        imprimir();
return pot;
}
.
.
.
```

```
.
.
.

int main(){
        int x, y, z, a, b, c, p1, p2, p3;
        cin >> x >> y >> z;
        cin >> a >> b >> c;
        p1 = potencia(x, a);
        p2 = potencia(y, b);
        p3 = potencia(z, c);
        cout << 2*p1 + 4*p2 + p3 << endl;
        return 0;
}
```

```
#include <iostream>
using namespace std;

void imprimir(){
        cout << “Potencia calculada”;
}

int potencia(int x, int a) {
        int pot = 1;
        for (int i = 0; i < a; i++)
                pot = pot*x;
        imprimir();
return pot;
}
.
.
.
```

Sem retorno

```
.
.
.

int main(){
        int x, y, z, a, b, c, p1, p2, p3;
        cin >> x >> y >> z;
        cin >> a >> b >> c;
        p1 = potencia(x, a);
        p2 = potencia(y, b);
        p3 = potencia(z, c);
        cout << 2*p1 + 4*p2 + p3 << endl;
        return 0;
}
```

Protótipo

- A definição de uma função

  deve ser feita antes da sua chamada

```
#include <iostream>
using namespace std;

void imprimir() {
        cout << “Potencia calculada”;
}

int potencia(int x, int a) {
        int pot = 1;
        for (int i = 0; i < a; i++)
                pot = pot*x;

        imprimir();
return pot;
}
```

Protótipo

- E se eu quiser/precisar definir a função

Depois da sua chamada?

- Faço a declaração da função (**protótipo**) antes, e a definição depois.

```
#include <iostream>
using namespace std;

void imprimir();

int potencia(int x, int a){
        int pot = 1;
        for(int i = 0; i < a; i++){
                pot = pot*x;
        }
        imprimir();
return pot;
}

void imprimir(){
        cout << “Potencia calculada”;
}
```

Funções - escopo de variáveis

- Observe que uma variável declarada em uma função, só existe naquela função.

```
int potencia(int x, int a){
        int pot = 1;
        for(int i = 0; i < a; i++)
                pot = pot*x;
return pot;
}
```

```
int main() {
int b, e;
cout << “Digite a base:”;
cin >> b;
cout << “Digite o expoente:”
cin >> e;
potencia(b, e);
cout << “Potencia: ” << pot;
return 0;
}
```

Funções - escopo de variáveis

- Observe que uma variável declarada em uma função, só existe naquela função.

```
int potencia(int x, int a) {
        int pot = 1;
        for(int i = 0; i < a; i++)
                pot = pot*x;

return pot;
}
```

```
int main() {
int b, e;
cout << “Digite a base:”;
cin >> b;
cout << “Digite o expoente:”
cin >> e;
potencia(b, e);
cout << “Potencia: ” << pot;
return 0;
}
```

pot não está definida no escopo da função main()

Funções - escopo de variáveis

- Observe que uma variável declarada em uma função, só existe naquela função.

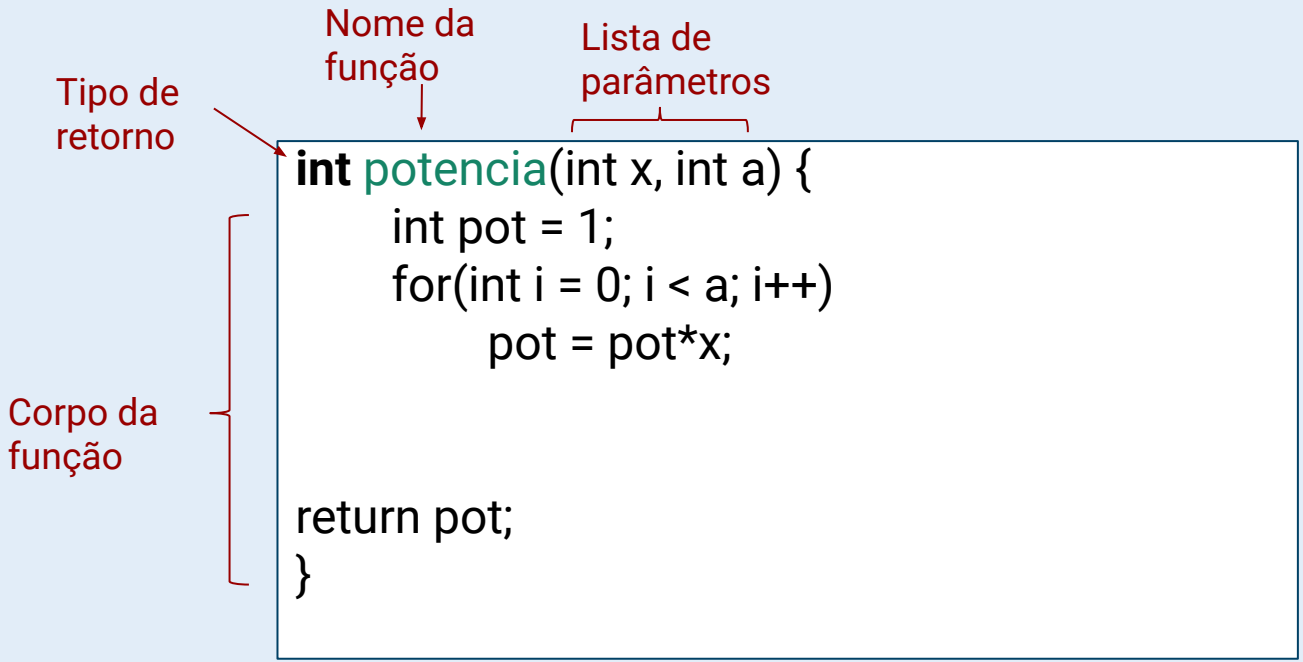
```
int potencia(int x, int a){
        int pot = 1;
        for(int i = 0; i < a; i++)
                pot = pot*x;

return pot;
}
```

```
int main(){
int b, e, p;
cout << “Digite a base:”;
cin >> b;
cout << “Digite o expoente:”
cin >> e;
p = potencia(b, e);
cout << “Potencia: ” << p;
return 0;
}
```

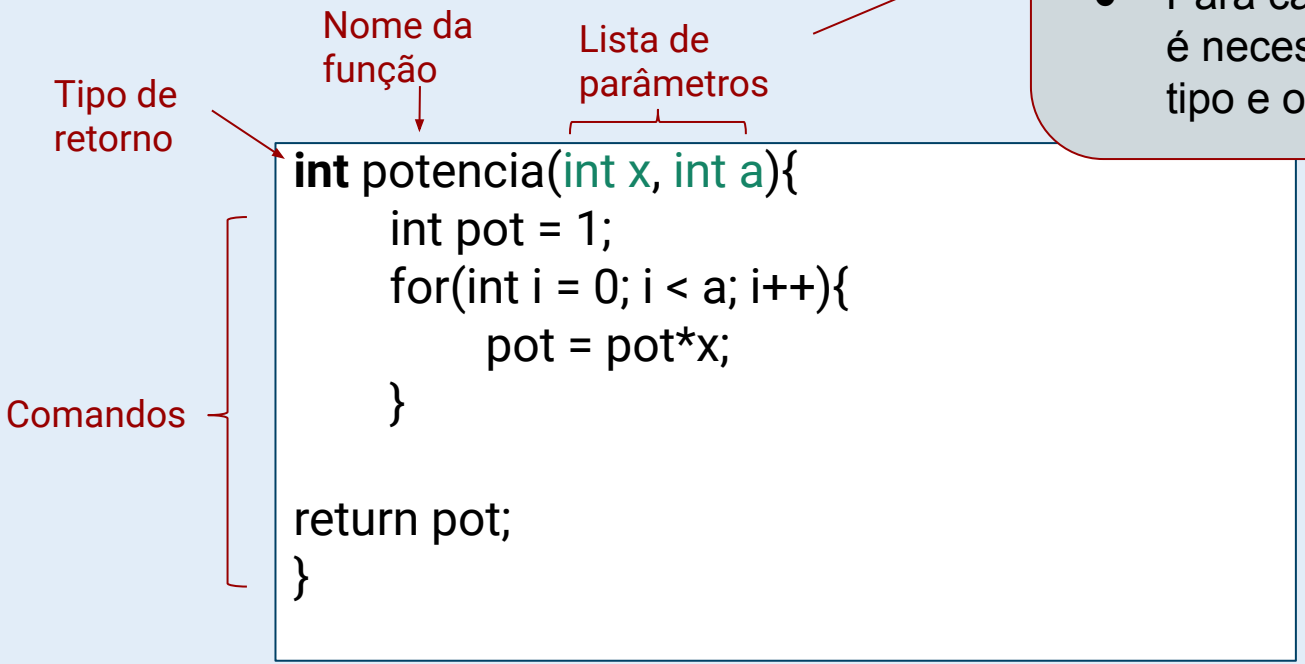
Exercício

- Escreva um programa com uma função que recebe dois parâmetros inteiros, calcula e retorna sua soma.
- O programa deve pedir ao usuário que digite dois números e então usar esta função para calcular a soma.
- As entradas e a saída do programa deve ser efetuada no programa principal, não na função.

Parâmetros


Nome da função
Lista de parâmetros
Tipo de retorno
int potencia(int x, int a) {
int pot = 1;
for(int i = 0; i < a; i++)
pot = pot*x;
return pot;
}
Corpo da função

Parâmetros



- Pode ser vazia
- Pode ter vários parâmetros (separados por vírgula)
- Para cada parâmetro, é necessário definir o tipo e o nome

Parâmetros

- Para chamar uma função, usa-se o nome da função e, caso tenha argumentos, passa-se valores como argumentos para a função

```
int p = potencia(2, 3);
```

Exemplo

- Qual a saída deste programa?

```
#include <iostream>
using namespace std;

void somar_um(int a) {
        a = a+1;
}

int main() {
        int x;
cout << “Digite um valor inteiro: ”;
cin >> x;
soma_um(x);
cout << x << endl;
return 0;
}
```

Exemplo

```
#include <iostream>
using namespace std;

void somar_um(int a){
        a = a+1;
}

int main() {
        int x;
cout << “Digite um valor inteiro: ”;
cin >> x;
soma_um(x);
cout << x << endl;
return 0;
}
```

O programa exibe o valor digitado pelo usuário sem somar um, parecendo que a função soma_um não foi executada. Por quê?

Exemplo

```
#include <iostream>
using namespace std;

void somar_um(int a) {
        a = a+1;
}

int main() {
        int x;
cout << “Digite um valor inteiro: ”;
cin >> x;
soma_um(x);
cout << x << endl;
return 0;
}
```

O programa exibe o valor digitado pelo usuário sem somar um, parecendo que a função *soma_um* não foi executada. Por quê?

A função *somar_um* é executada. Porém, quando a variável *x* é passada como argumento para a função, é feita uma **cópia do seu valor** para a variável *a*.

Passagem de parâmetro por valor

- Os parâmetros da função são **variáveis locais** ao corpo da função.
- Quando é feita uma chamada da função com seus argumentos
  - os argumentos são avaliados
  - os parâmetros da função são alocados na memória e inicializados com os **valores dos argumentos** correspondentes (ou seja, são **copiados** para os parâmetros da função)
  - o corpo da função é executado
  - a expressão de retorno é avaliada

Exemplo

- Se fosse necessário alterar o valor de x na função?
  - Passagem de parâmetro por referência
  - Coloca-se um & na frente do nome do parâmetro que deve ter o valor alterado

Exemplo

```
#include <iostream>
using namespace std;

void somar_um(int &a) {
        a = a+1;
}

int main() {
        int x;
cout << “Digite um valor inteiro: ”;
cin >> x;
soma_um(x);
cout << x << endl;
return 0;
}
```

Exemplo

```
#include <iostream>
using namespace std;

void somar_um(int &a){
        a = a+1;
}

int main(){
        int x;
cout << “Digite um valor inteiro: ”;
cin >> x;
soma_um(x);
cout << “x:” << x << endl;
return 0;
}
```

```
Digite um valor inteiro: 5

x: 6
```

Exercício

- Faça um programa que tenha uma função chamada ***troca***. Essa função deve receber dois parâmetros reais por referência e trocar o valor dos parâmetros.