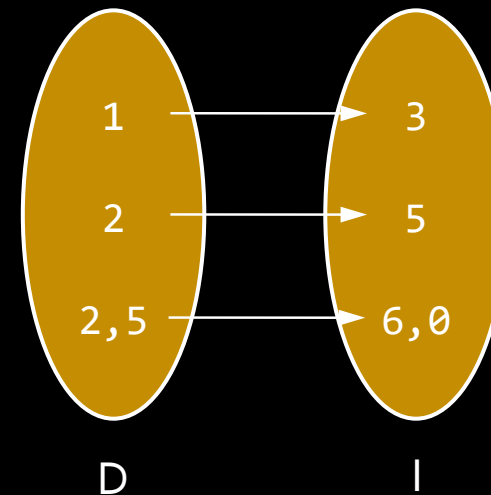


Programação de Computadores

MODULARIDADE E FUNÇÕES

Introdução

- O que é uma **função**?



Visão matemática

$$f: D \rightarrow I$$

- É uma lei que **associa elementos** do conjunto D (domínio) a elementos do conjunto I (imagem)

$$f(x) = 2x+1$$

f: \rightarrow

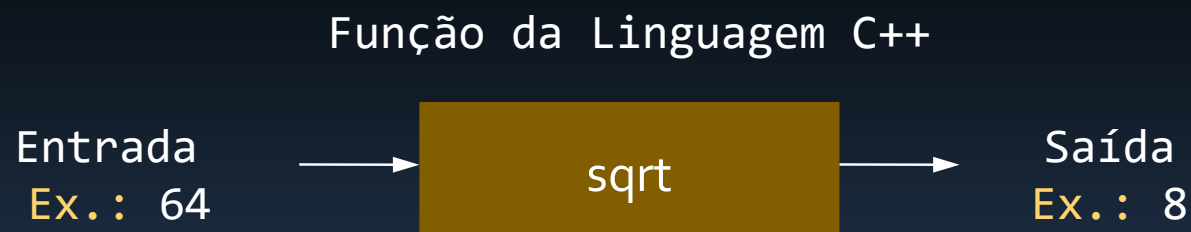
O domínio e a imagem definem uma **interface para a função**

Introdução

- O que é uma **função**?

Visão computacional

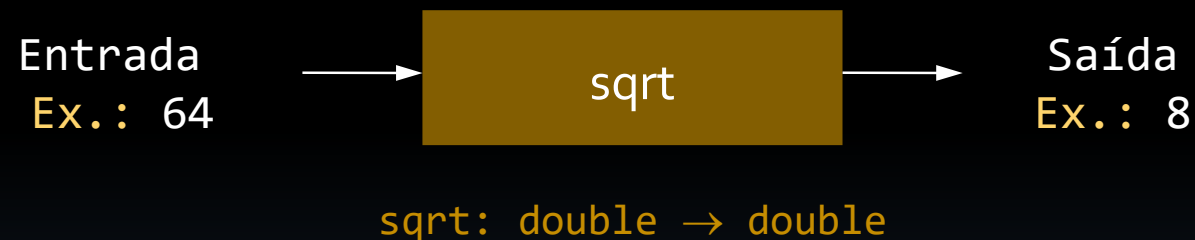
- Uma caixa preta que recebe uma entrada e retorna uma saída



`sqrt` calcula a raiz quadrada de um número

Introdução

- Como na matemática, **toda função em C++ tem uma interface**



A interface da função computacional é definida através de **tipos computacionais** ao invés de conjuntos numéricos

Funções

```
// exemplo do uso de funções
#include <iostream>
#include <cmath>
using namespace std;

int main()
{
    cout << "Digite a área da sua casa em metros quadrados: ";
    double area;
    cin >> area;

    double lado;
    lado = sqrt(area); // chamada da função sqrt

    cout << "Isso é o equivalente a um quadrado com " << lado
         << "metros de lado." << endl;

    system("pause"); // chamada da função system
}
```

Funções

- A interface de uma função é chamada de **protótipo da função**
 - Em C++ toda função deve ter seu protótipo definido por uma instrução de declaração

```
double sqrt(double); // protótipo da função sqrt
```

- O protótipo da função sqrt está definido no **arquivo de inclusão** `cmath`
`#include <cmath>`

Funções

- Para **utilizar a função sqrt** foi preciso:
 - Incluir o arquivo de cabeçalho cmath
 - Chamar a função dentro do programa:

Valor de retorno
é atribuído à variável

Argumento (valor)
passado para a função

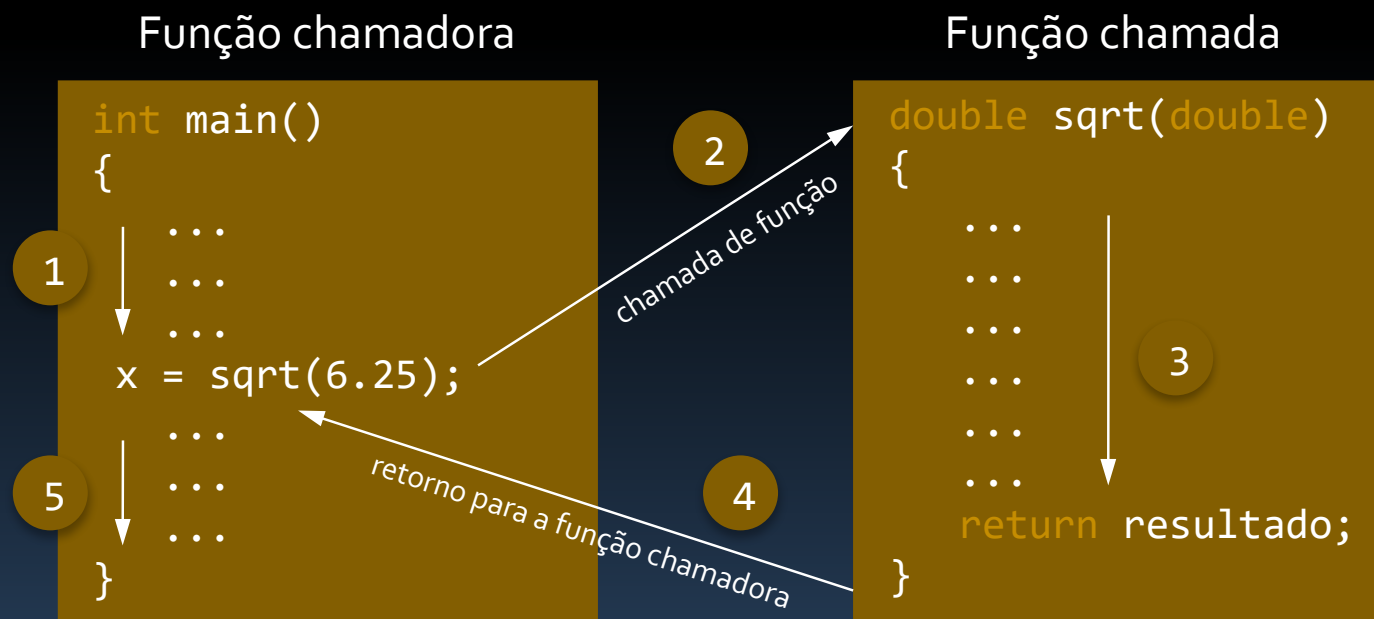
```
lado = sqrt(area);
```

Nome da função

Funções

- Uma **chamada de função** provoca um desvio no programa

```
// retorna o valor 2.5 e o atribui para x  
x = sqrt(6.25);
```



Funções

- Além de um **protótipo** toda função precisa ter uma **definição**
 - Para a função sqrt:
 - O protótipo da função está em um arquivo de inclusão
`#include <cmath>`
 - A definição da função está em um arquivo objeto (DLL), previamente compilado e instalado* no sistema
`msvcr100d.dll`

Funções

- Ao contrário das funções matemáticas, em C++ as funções podem **não retornar valor**

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

Nome da função

Argumento (valor)
passado para a função

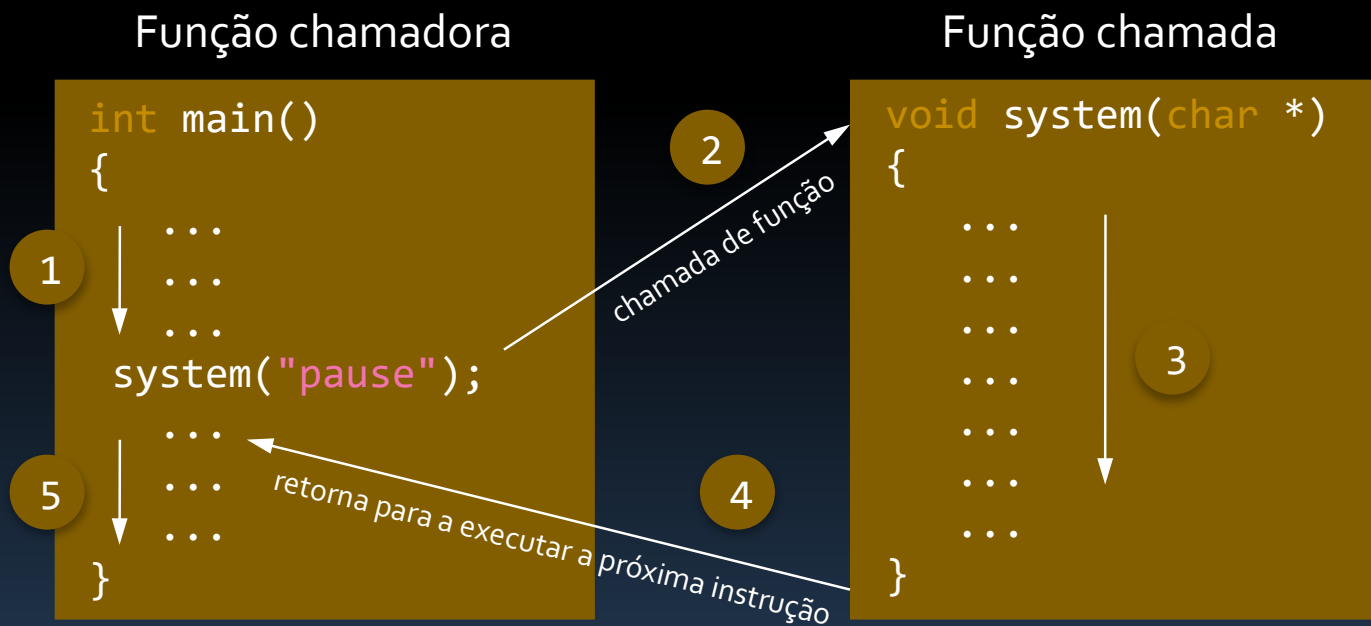
Entrada
"pause"



system

Funções

- Uma **função que não retorna valor** executa uma tarefa sem retornar resultado



Funções

- Uma **função que não retorna valor** possui tipo de retorno **void** no seu protótipo

```
void system(const char*); // protótipo da função system
```

- Uma função com tipo de retorno **void não pode ser atribuída** a uma variável

```
// atribuição inválida  
x = system("pause"); ❌
```

Entrada
"pause"



system

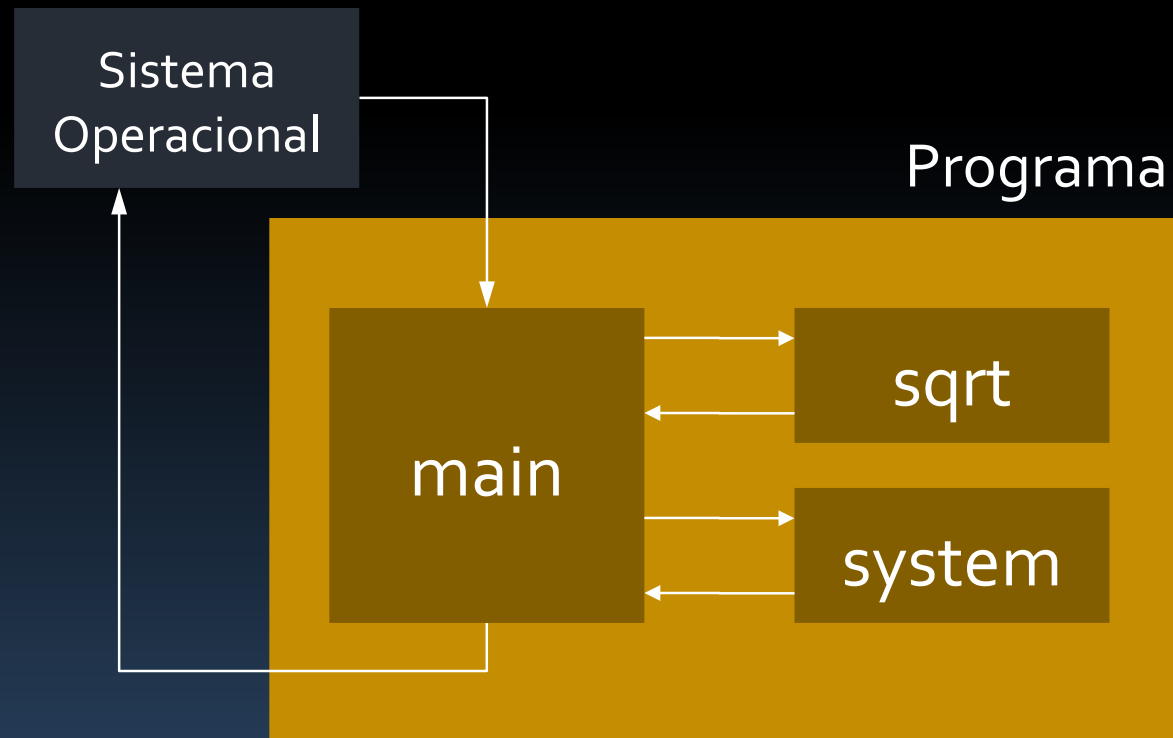
Modularidade

- Para que servem as funções?
 - **Dividir** o código em blocos
 - **Reaproveitar** código existente
- A modularização de programas é a principal característica da **programação estruturada**
 - Facilita a manutenção
 - Encapsula a solução
 - Cria uma interface

Saída ← `double` sqrt(`double`); → Entrada

Modularidade

- Os **programas** em C++ são construídos a partir desses blocos, chamados de funções

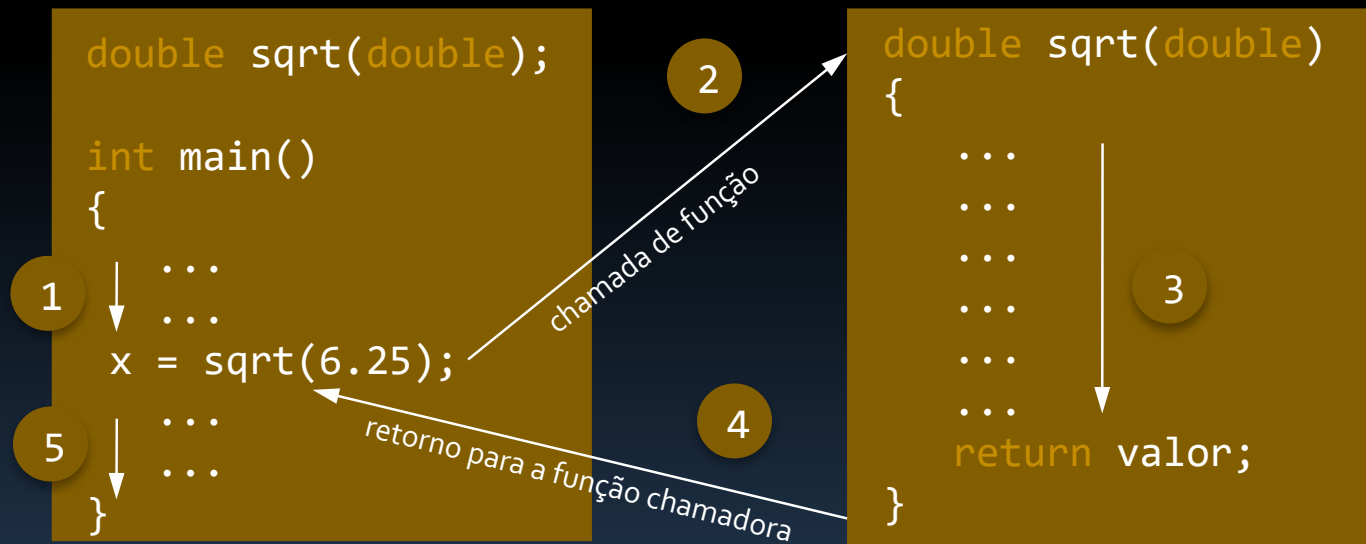


Modularidade

- Muitas funções já vem **prontas em bibliotecas** fornecidas pela linguagem
 - **iostream**: contém funções relacionadas a entrada e saída de dados como cout, cin e system*
 - **cmath**: contém funções matemáticas como sqrt (raiz quadrada), pow (potência), sin (seno), cos (cosseno), tan (tangente), etc.
- Outras funções podem ser **criadas pelo programador**

Modularidade

- De uma forma geral, para **usar uma função** é preciso:
 - Fornecer um **protótipo**, uma **definição** e **chamar** a função



Bibliotecas de Funções

- Ao usar uma **função de biblioteca**:
 - O **protótipo** é obtido incluindo-se um arquivo de cabeçalho

```
#include <iostream>
#include <cmath>
```

- A **definição** já foi pré-compilada em um arquivo objeto/DLL

```
msvcr100d.dll
```

- Resta apenas **chamar a função**

```
lado = sqrt(area);
system("pause");
```

Bibliotecas de Funções

- A biblioteca padrão da linguagem C/C++ é extensa
 - Possui mais de **140 funções** predefinidas
 - Prefira utilizar as funções existentes
- Mesmo assim, o programador precisa escrever suas funções
 - Uma **função definida pelo programador** funciona da mesma forma que uma função de biblioteca
 - A diferença:
 - É preciso **escrever o protótipo**
 - É preciso **escrever a definição** da função

Criando Funções

- Para **criar uma função** é preciso:
 - ✓ Fornecer um **protótipo** para a função
 - **Definir** a função (fornecer um corpo)
 - **Chamar** a função
- **Protótipo**: define que informações a função recebe e que informações a função retorna

```
void simples(); // protótipo da função
```

Criando Funções

- Para **criar uma função** é preciso:
 - Fornecer um **protótipo** para a função
 - ✓ **Definir** a função (fornecer um corpo)
 - **Chamar** a função
- **Definição:** contém um conjunto de instruções que realiza a tarefa para a qual a função foi criada

```
// definição da função simples
void simples()
{
    cout << "Eu sou uma função simples" << endl;
}
```

Criando Funções

- Para **criar uma função** é preciso:
 - Fornecer um **protótipo** para a função
 - **Definir** a função (fornecer um corpo)
 - ✓ **Chamar** a função
- **Chamada**: invoca a função a partir do programa principal, ou a partir de uma outra função

```
simples(); // chamada da função
```

Criando Funções

```
// declarando, definindo e chamando uma função
#include <iostream>
using namespace std;

void simples(); // protótipo da função

int main()
{
    cout << "main() vai chamar a função simples():\n";
    simples(); // chamada da função

    return 0;
}

void simples() // definição da função
{
    cout << "Eu sou uma função simples" << endl;
}
```

Depurando Funções

- A depuração de um programa com funções pode ser feita:
 - Menu **Depurar** > **Pular método** (Step Over - F10)
 - Executa a função em um único passo
 - O erro não está na função
 - Menu **Depurar** > **Intervir** (Step Into - F11)
 - Entra na função para executar cada instrução
 - Inspeção das instruções da função

Depurando Funções

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

void feliz(void);
void natal(void);
void anonovo(void);

int main()
{
    cout << "Eu desejo a todos um ";
    feliz();
    natal();
    cout << "e um ";
    feliz();
    anonovo();
    cout << endl;

    return 0;
}
```

```
void feliz(void)
{
    cout << "Feliz ";
}

void natal(void)
{
    cout << "Natal ";
}

void anonovo(void)
{
    cout << "Ano Novo ";
}
```


Depurando Funções

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

void auxiliar(void);
void outra(void);

int main(void)
{
    cout << "Um programa em C++, "
          << "sempre começa pela "
          << "função main.\n";

    outra();
    auxiliar();

    cout << "mais de uma vez.\n";
    return 0;
}
```

```
void auxiliar(void)
{
    cout << "Funções também podem "
          << "ser invocadas ";
}

void outra(void)
{
    cout << "Funções podem ser"
          << "invocadas a partir "
          << "da função main.\n";

    auxiliar();

    cout << "a partir de "
          << "outras funções.\n";
}
```

Resumo

- Um **programa C++** consiste de uma ou mais funções
- A execução inicia a partir da função **main()**
- Uma função consiste em:
 - Um **protótipo**: define os tipos de valores recebidos e o tipo do valor retornado pela função
 - Uma **definição**: consiste em uma série de instruções entre um par de chaves ({ })

Resumo

- Em C++ o **programador** pode:
 - Usar funções pré-definidas de bibliotecas
Ex.: sqrt e system
 - Criar suas próprias funções
Ex.: simples
- O uso de funções
 - Facilita a **manutenção** de programas grandes
 - **Encapsula** a solução
 - Cria uma **interface**