

Programação de Computadores

# INTRODUÇÃO AO C++

# Iniciação ao C++

```
// primeiro.cpp - mostra uma mensagem
```

Comentário

```
#include <iostream>
```

Uma diretiva de pré-processamento

```
int main()
```

Cabeçalho da função

```
{
```

```
using namespace std;
```

Dispensa uso do nome longo std::cout, std::endl

```
cout << "Bem vindo ao C++.";
```

Exibe mensagem na tela

```
cout << endl;
```

Inicia uma nova linha

```
cout << "Meu primeiro programa!" << endl;
```

Outra mensagem

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

Espera pelo pressionamento de uma tecla

```
return 0;
```

Finaliza a função main()

```
}
```

Corpo da função

# Iniciação ao C++

- A linguagem C++ faz diferença entre letras **maiúsculas e minúsculas**
  - Só existe uma forma de escrever **cout**:

cout ✓	kout ✗
Cout ✗	cour ✗
COUT ✗	coot ✗
- O compilador retorna um erro de **identificador não declarado** se for usada uma palavra inválida

# Iniciação ao C++

```
// programa de boas vindas
#include <iostream>
using namespace std;

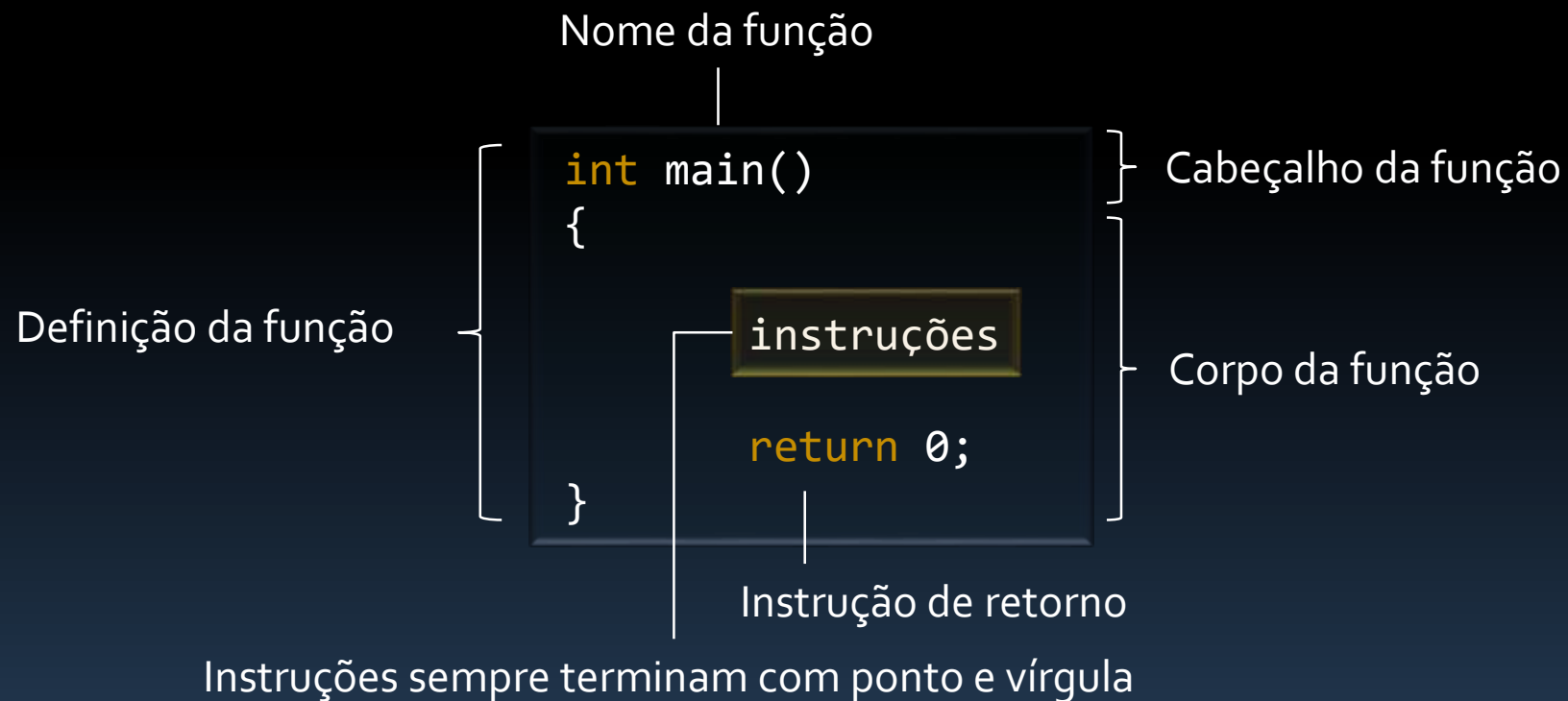
int main()
{
    Cout << "Bem vindo ao C++." << ENDL;
    return 0;
}
```

## ■ Resultado da Compilação:

```
>BemVindo.cpp
>BemVindo.cpp(7): error C2065: 'Cout': identificador não declarado
>BemVindo.cpp(7): error C2065: 'ENDL': identificador não declarado
>Projeto de compilação pronto "ProgComp.vcxproj" -- FALHA.
```

# A Função `main()`

- A estrutura da função `main()` é:



# Instruções

- Para traduzir corretamente as instruções, o compilador precisa saber exatamente **onde uma instrução termina**:
  - FORTRAN: uma instrução por linha
  - PASCAL: separa uma instrução da próxima com ;
  - C/C++: cada instrução termina por um ;

```
cout << "Bem vindo ao C++.";
cout << "Meu primeiro programa!" << endl;
return 0;
```

# Instruções

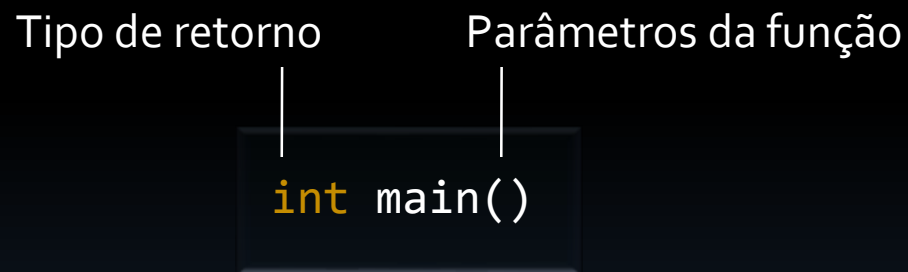
- Ao contrário de PASCAL, na linguagem C++ o **ponto e vírgula** faz parte da instrução

```
// primeiro.cpp - mostra uma mensagem
#include <iostream>
using namespace std;

int main()
{
    cout << "Bem vindo ao C++.";
    cout << endl;
    cout << "Meu primeiro programa!" << endl;
    return 0;
}
```

# O Cabeçalho da Função

- O cabeçalho de uma função descreve a **interface** da função



- A função main é a interface entre o **sistema operacional** e o seu programa



# O Cabeçalho da Função

- A função main também **pode ser escrita** assim

Tipo de retorno

Parâmetros da função

```
int main(int argc, char ** argv)
```

- `argc` é a quantidade de comandos passados na linha de comando
- `argv` são os comandos em si

# O Cabeçalho da Função

- Utilizando **argumentos** passados na **linha de comando**

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

int main(int argc, char ** argv)
{
    cout << "Programa: " << argv[0] << endl;

    if (argc > 1)
        cout << "Arg: " << argv[1] << endl;

    return 0;
}
```

# O Cabeçalho da Função

- Muitos **programas existentes** usam o cabeçalho clássico da linguagem C
- Uma **alternativa válida** em C++ é dizer explicitamente que a função **não recebe argumentos**

```
main() // estilo original C
{
}
```



```
int main(void) // estilo explícito
{
}
```



# O Cabeçalho da Função

- Alguns programadores usam este cabeçalho para omitir a instrução de retorno
- O padrão ANSI/ISO C++ permite que o programador omita a instrução de retorno (somente na função main)

```
// não está no padrão C++  
void main()  
{  
}
```



```
// sem retorno explícito  
int main()  
{  
}
```



# Comentários

- Comentários são introduzidos com o uso de **barras duplas**

```
// comentários acabam no final da linha
```

- O compilador ignora os comentários
- **Comentários devem ser usados** para:
  - Documentar os programas
    - Ajudam outros a entenderem o programa
    - Permitem que o programador lembre o que fez

# Comentários

- C++ também aceita comentários no **estilo da linguagem C**

- Iniciando com `/*`

- Finalizando com `*/`

```
/*  
    comentários na linguagem C  
    podem ocupar várias  
    linhas  
*/
```

```
/*  
=====  
comentário em destaque  
=====  
*/
```

```
/* comentários em C permitem  
vários estilos de  
organização */
```

```
/** comentário válido **/
```

```
/* porém é preciso ficar  
atento para não  
cometer erros */
```

# 0 Pré-processador de C++

- Um **pré-processador** é um programa que processa (modifica) o código fonte antes da compilação
  - O pré-processador trata as **diretivas** que iniciam com **#**
- A diretiva **#include** adiciona o conteúdo de um arquivo ao código fonte do programa

```
// adiciona o arquivo iostream ao programa  
#include <iostream>
```

# 0 Arquivo `iostream`

- O arquivo `iostream` contém as definições das funções de entrada e saída de dados
  - `i` = input (entrada)
  - `o` = output (saída)
  - `stream` = fluxo ou canal

```
// necessário para usar cin e cout  
#include <iostream>
```

Programas que usam `cin` e `cout` para `entrada e saída` de dados devem incluir o arquivo `iostream`



# Arquivos de Cabeçalho

- Arquivos como **iostream** são chamados:
  - Arquivos de cabeçalho (**header files**) ou
  - Arquivos de inclusão (**include files**)

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

int main()
{
}
```

Os includes são feitos  
no início (cabeçalho)  
do programa.

# Arquivos de Cabeçalho

- A tradição sempre foi usar a **extensão .h** para estes tipos de arquivos:
  - `iostream.h` – funções de entrada/saída
  - `math.h` – funções matemáticas
- A convenção atual é:
  - Não usar extensão nos **arquivos padrão**
  - Usar nos arquivos criados pelo programador

# Arquivos de Cabeçalho

- C++ aceita arquivos de cabeçalho:
  - Na nova convenção sem extensão (Ex.: `iostream`)
  - No antigo formato do C (Ex.: `math.h`)
  - Convertidos do C para C++ (Ex.: `cmath`)

Cabeçalho	Convenção	Exemplo	Usado por
C++ estilo antigo	.h	iostream.h	Programas C++
C estilo antigo	.h	math.h	Programas C e C++
C++ novo estilo	Sem extensão	iostream	Programas C++
C convertido	Prefixo c	cmath	Programas C++

# Namespaces

- Ao incluir `iostream` em um programa é preciso acrescentar a seguinte diretiva `using`:

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

- Isto permite usar os objetos `cin` e `cout` sem a designação do espaço de nome ao qual eles pertencem

```
// utilização sem a diretiva using  
std::cout << "Bem vindo ao C++.";
```

# Namespaces

- É um recurso que permite combinar códigos existentes de diferentes fornecedores:
  - Se duas empresas desenvolverem um objeto `cout`, elas o farão sob diferentes namespaces:
    - `Mioflop::cout` – objeto `cout` da Mioflop
    - `Wanda::cout` – objeto `cout` da Wanda
- Para usar todos os objetos da Mioflop:  
`using namespace Mioflop;`

# Namespaces

- As funções, classes e objetos **padrões da linguagem C++** foram colocados no espaço de nomes **std**

- Para ter acesso a tudo definido em std:

```
using namespace std;
```

- Para ter acesso apenas a itens selecionados:

```
using std::cout;  
using std::cin;  
using std::endl;
```

# Saída de Dados com cout

- O programa `primeiro.cpp` usa `cout` para exibir uma mensagem na tela:

Um objeto predefinido que sabe  
como mostrar números e caracteres

conjunto de caracteres (string)

```
cout << "Bem vindo ao C++.";
```

Operador de inserção: indica a  
direção do fluxo de informações

# O Manipulador endl

- O programa `primeiro.cpp` usa `cout` também para pular linhas na tela:

Manipulador  
|  
`cout << endl;`

Enviar `endl` para a saída faz o cursor saltar para o início da próxima linha

- Assim como `cout`, `endl` é definido no arquivo `iostream`



# O Manipulador endl

- cout **não pula linha automaticamente:**

```
cout << "O Bom, o";  
cout << "ruim, ";  
cout << "e o desconhecido."  
cout << endl;
```

- Produzirá a saída:

O Bom, ruim, e o desconhecido.

# 0 Caractere de Nova Linha

- Existe outra forma de produzir um salto de linha:

```
// \n significa comece uma nova linha  
cout << "E agora?\n";
```

- A combinação \n é considerada como um único caractere

```
// os pares de instruções abaixo são equivalentes  
cout << "Júpiter é um planeta grande.\n";  
cout << "Júpiter é um planeta grande." << endl;  
  
cout << "\n";  
cout << endl;
```

# Formatação do Código Fonte

- Ao editar um código C++ o programador tem muita **flexibilidade na formatação do código**

```
#include <iostream>
    int
main
() { using
    namespace
        std;    cout
        <<
    "Bem vindo ao C++."
    ;cout <<
    endl; cout << "Meu primeiro programa!"
    << endl; return 0; }
```

# Estilo de Código C++

- A **leitura do código fonte** é facilitada se o programador seguir algumas regras básicas:

```
// primeiro.cpp - mostra uma mensagem
```

```
#include <iostream>
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
    cout << "Bem vindo ao C++.";
```

```
    cout << endl;
```

```
    cout << "Meu primeiro programa!" << endl;
```

```
    return 0;
```

```
}
```

Uso de espaços para separar blocos

Abra e feche chaves em linhas separadas

Uma instrução por linha

Instruções indentadas

# Resumo

- Programas iniciam a execução a partir de uma função principal chamada **main()**
- Uma **função** consiste em:
  - Um **cabeçalho**: define a interface da função, o tipo de valor recebido e o tipo de valor retornado como resultado
  - Um **corpo**: consiste em uma série de **instruções** dentro de um par de chaves ({ }) e finalizadas por ponto-e-vírgula

# Resumo

- Diretivas de pré-processamento modificam o programa
  - São executadas antes de iniciar a compilação
  - `#include` insere o conteúdo de um arquivo no programa
- A instrução de saída de dados (cout)
  - Está definida no `arquivo de cabeçalho iostream`
  - Não salta linhas automaticamente
    - Use o manipulador `endl`
    - Ou o `caractere '\n'`