

Estruturas de Dados I

Introdução

Prof. Bruno Azevedo

Instituto Federal de São Paulo



INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
SÃO PAULO
Campus Catanduva

Hello World em C++

```
#include <iostream>
int main() {
    std::cout << "Olá Mundo!\n";
    return 0;
}
```

- Vamos começar olhando o código mais famoso que existe escrito em C++.

Hello World em C++

```
#include <iostream>
int main() {
    std::cout << "Olá Mundo!\n";
    return 0;
}
```

- Em C++, `#include <iostream>`. é uma diretiva de pré-processador que informa ao compilador para incluir o arquivo de cabeçalho `iostream` no código-fonte.
- Um arquivo de cabeçalho é um arquivo que contém declarações de funções, classes, estruturas, constantes, macros, etc.
- Podemos usar estes arquivos para acessar bibliotecas e utilizar seus recursos.
- Uma biblioteca é um conjunto de códigos pré-compilados que contém funções, classes, etc, que podem ser utilizados por nossos programas.

Hello World em C++

```
#include <iostream>
int main() {
    std::cout << "Olá Mundo!\n";
    return 0;
}
```

- Temos então a função `main()`. Todo programa em C++ possui uma função `main()`.
- Uma função é um bloco de código que executa uma, ou mais, instruções.
- Essa função é especial; assim que seu programa inicia, a função `main()` é chamada automaticamente.
- O tipo do valor de retorno é `int` (inteiro – logo veremos isso em detalhe).
- Notem, que ao contrário de outras linguagens (e.g. Python), precisamos usar chaves `{ e }` para delimitar blocos de código (existem exceções, mas não precisam se preocupar com isso agora).
- Muitas das instruções em C++ precisam ser terminadas com o ponto-e-vírgula `(;)`. Existem exceções, como diretivas de pré-processador e estruturas de controle de fluxo.

Hello World em C++

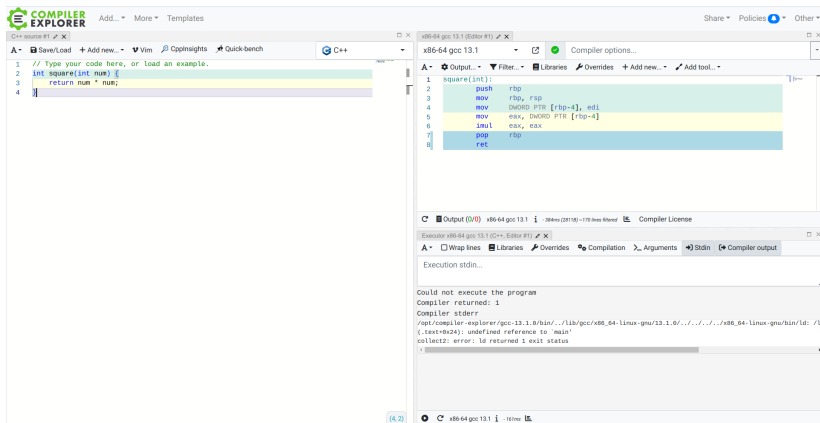
```
#include <iostream>
int main() {
    std::cout << "Olá Mundo!\n";
    return 0;
}
```

- Em seguida temos a parte principal do código: a impressão do texto.
- Usamos `cout` para enviar informações para saída padrão, no caso, nossa tela.
- `std` avisa ao compilador que o objeto `cout` que usaremos faz parte da biblioteca padrão, utilizando o especificador `std` (namespace standard).
- Todos devem conhecer o caractere especial `\n` para quebra de linha. Comum em muitas linguagens.
- Finalmente, retornamos zero usando o comando *return*. Ou seja, informamos que o programa executou com sucesso.

Compilando o nosso Código

- Não é tão interessante olharmos código em slides.
- Sempre é mais interessante executarmos nós mesmos.
- Existem diversas opções. Vocês podem usar o compilador que desejarem: GCC, Visual Studio, etc.
- Eu recomendarei uma opção online que poderão usar durante toda a disciplina.

Compilando o nosso Código



- O Compiler Explorer está em <https://godbolt.org/>.
- É bem poderoso e robusto, apresentando diversas opções.
- Ele executa a todo momento, não é necessário buscar um botão de “Executar”.

Compilando o nosso Código

The screenshot shows the Compiler Explorer web application. On the left, the C++ source code is displayed: `int square(int num) { return num * num; }`. An arrow points from the text 'Escrevemos aqui nosso código em C++' to this code. In the center, the assembly output for the x86_64 gcc 13.1 compiler is shown, with instructions like `push rbp`, `mov rbp, rsp`, `mov DWORD PTR [rbp-4], edi`, `mov eax, DWORD PTR [rbp-4]`, `imul eax, eax`, `add rbp, 4`, and `ret`. An arrow points from the text 'Aqui temos o código em Assembly. Podemos mudar o compilador, adicionar opções, é bem flexível.' to this assembly output. On the right, the 'Compiler options...' panel is open, showing various configuration options. An arrow points from the text 'Aqui podemos ver a saída e adicionar itens à entrada (vocês precisarão adicionar essa janela usando add new - execution only - e arrastando para esta posição.' to the 'Execution stdin...' field in the 'Compiler output' tab. Below this, an error message is visible: 'Could not execute the program. Compiler returned: 1. Compiler stderr: /opt/compiler-explorer/gcc-13.1.0/bin/.../lib/gcc/x86_64-linux-gnu/13.1.0/.../x86_64-linux-gnu/bin/ld: (.text+0x24): undefined reference to 'main' collect2: error: returned 1 exit status'.

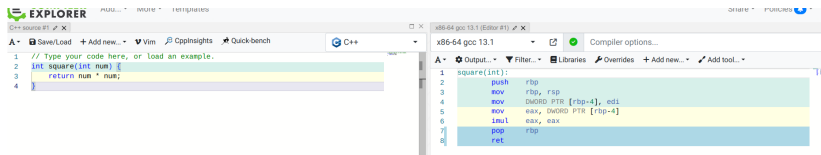
Escrevemos aqui nosso código em C++

Aqui temos o código em Assembly. Podemos mudar o compilador, adicionar opções, é bem flexível.

Aqui podemos ver a saída e adicionar itens à entrada (vocês precisarão adicionar essa janela usando add new - execution only - e arrastando para esta posição.

- O Compiler Explorer está em <https://godbolt.org/>.
- É bem poderoso e robusto, apresentando diversas opções.
- Ele executa a todo momento, não é necessário buscar um botão de “Executar”.

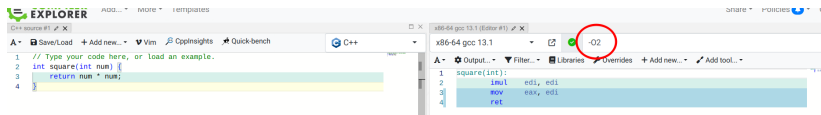
Compilando o nosso Código



```
1 // Type your code here, or load an example.
2 int square(int num) {
3     return num * num;
4 }
```

```
1 square(int):
2     push    rbp
3     mov     rbp, rsp
4     mov     DWORD PTR [rbp+4], edi
5     mov     eax, DWORD PTR [rbp-4]
6     imul    eax, eax
7     pop     rbp
8     ret
```

- Usar a opção `-O2` no compilador GCC é uma forma de otimizar o código C++ durante o processo de compilação.
- É um dos níveis de otimização disponíveis no GCC, sendo um dos mais comuns para melhorar o desempenho do programa gerado.
- Podemos ver a otimização no código em Assembly.



```
1 // Type your code here, or load an example.
2 int square(int num) {
3     return num * num;
4 }
```

```
1 square(int):
2     imul    edi, edi
3     mov     eax, edi
4     ret
```

Conhecendo o objeto *cout*

- `cout` \Rightarrow C++ Output.
- O objeto `cout`, juntamente com o operador `<<` (geralmente chamado de enviar ou inserir), é usado para exibir valores, imprimir texto.
- Vamos olhar alguns exemplos de uso do `cout`.
- Abram o site do Compiler Explorer, ou o compilador de sua preferência, e copiem o código abaixo.

```
#include <iostream>
int main() {
    std::cout << "Olá! Aqui está um número:\t" << 42 << "\n";
    std::cout << "O manipulador endl efetua a quebra de linha.\t" << std::endl;
    std::cout << "Aqui está uma fração:\t\t" << (float) 5/8 << std::endl;
    std::cout << "Um número bem grande:\t\t" << (double)7000*7000 << std::endl;
    std::cout << "Estou aprendendo a programar em C++!\n";
    return 0;
}
```

- Vejam o resultado.
- PS: $4.9e+07$ é notação científica representando 49 milhões.

Conhecendo o objeto *cout*

```
#include <iostream>
int main() {
    std::cout << "Olá! Aqui está um número:\t" << 42 << "\n";
    std::cout << "O manipulador endl efetua a quebra de linha.\t" << std::endl;
    std::cout << "Aqui está uma fração:\t\t" << (float) 5/8 << std::endl;
    std::cout << "Um número bem grande:\t\t" << (double)7000*7000 << std::endl;
    std::cout << "Estou aprendendo a programar em C++!\n";
    return 0;
}
```

- Na primeira linha, exibimos uma *string*, em seguida um inteiro, e o caractere `\n` para quebrar a linha.
- Notem como utilizamos o operador `<<` para separar cada elemento.
- Como alternativa ao `\n`, podemos usar o manipulador `endl`. Existem algumas diferenças, mas não são particularmente relevantes neste momento da disciplina.
- Neste momento, podem escolher qual usar, fica a seu critério.
- O caractere especial `\t` insere um tab, pode ser usado para organizar a impressão como desejar.

Conhecendo o objeto *cout*

- Pode ser um pouco cansativo escrever `std::` a todo momento quando usando `cout`, `endl`, etc.

```
#include <iostream>
using namespace std;
int main() {
    cout << "Olá! Aqui está um número:\t" << 42 << "\n";
    cout << "O manipulador endl efetua a quebra de linha.\t" << endl;
    cout << "Aqui está uma fração:\t\t" << (float) 5/8 << endl;
    cout << "Um número bem grande:\t\t" << (double)7000*7000 << endl;
    cout << "Estou aprendendo a programar em C++!\n";
    return 0;
}
```

- Uma alternativa é indicar que estaremos utilizando o namespace `std` para todos os objetos em nosso código.

Conhecendo o objeto *cout*

- Vamos tentar algo um pouco mais complexo.
- Testem o código abaixo.

```
#include <iostream>
```

```
using namespace std;
```

```
int main() {  
    int n = 19, n2 = 42, result, result2;  
    result = n * n;  
    result2 = n2 * n2;  
    cout << " Resultado: " << result << " e " << result2 << endl;  
    return 0;  
}
```

- Como podem ver, podemos usar variáveis com *cout*.
- Discutiremos variáveis um pouco mais pra frente.

Comentando o seu Programa

- Todos aqui devem saber da importância de comentar seu código.
- Não existem novidades, já que muitas linguagens utilizam os mesmos elementos.
- Linha simples: `//Comentário`.
- Bloco: `/* Comentário */` (para comentários de múltiplas linhas).

```
// Esse é um comentário inline
```

```
cout << "Hello World!\n";
```

```
cout << "Hello World!\n"; // Esse também é um comentário inline
```

```
/* Esse é um
```

```
comentário
```

```
em bloco*/
```

```
cout << "Hello World!\n";
```

Tipos Primitivos e Variáveis

Tipos Primitivos

- Tipos primitivos, ou tipos de dados primitivos, são os tipos de dados básicos suportados diretamente pela linguagem de programação.
- Esses tipos de dados são usados para representar valores, como números inteiros, números reais (de ponto flutuante), caracteres, valores booleanos, entre outros.
- Os tipos primitivos fornecem a base para a construção de estruturas de dados mais complexas.
- Cada linguagem de programação tem seus próprios tipos primitivos.
- Exemplos: 5,5 é um dado do tipo real. 7 é um dado do tipo inteiro. 'S' é um dado do tipo caractere. True é um dado do tipo booleano.

Variáveis

- Uma variável é um local para armazenar informações.
- Ou seja, é uma localização na memória do seu computador na qual você pode armazenar um valor, modificar esse valor, e recuperar esse valor.
- Este valor será de um tipo primitivo, ou um tipo criado pelo programador.
- Observe que as variáveis são usadas para armazenamento temporário.

Variáveis

- C++ é uma linguagem de programação fortemente tipada.
- Isso significa que é necessário definir explicitamente o tipo (primitivo) de dados de cada variável antes de utilizá-la.
- C++ é uma linguagem estaticamente tipada
- Isso significa que o tipo das variáveis é determinado em tempo de compilação e não pode ser alterado durante a execução do programa.
- Em C++, existem diferentes tipos de variáveis (tipos primitivos), que são definidos com diferentes palavras-chave.

Tipo	Descrição
int	Armazena números inteiros, sem casas decimais, como 123 ou -123.
double	Armazena números de ponto flutuante, com casas decimais, como 19.99 ou -19.99.
char	Armazena caracteres individuais, como 'a' ou 'B'. Os valores char são delimitados por aspas simples.
string	Armazena texto, como "Hello World". Os valores <i>string</i> são delimitados por aspas duplas.
bool	Armazena valores com dois estados: verdadeiro (<i>true</i>) ou falso (<i>false</i>).

Variáveis

- Para criar uma variável, especifique o tipo, defina um nome, e atribua um valor a ela.

`tipo nomeVariável = valor;`

- Alternativamente, especifique o tipo e defina um nome. Podemos atribuir um valor posteriormente, se desejarmos.

`tipo nomeVariável;`

- Não esqueça do ponto-e-vírgula (;) ao fim.

Variáveis

- Exemplo 1:

```
/* Aqui estamos criando uma variável do tipo inteiro  
e atribuindo o valor 15.*/
```

```
int minhaVariavel = 15;  
cout << minhaVariavel;
```

- Exemplo 2:

```
/* Aqui estamos criando uma variável do tipo inteiro,  
mas não atribuímos valor a ela.*/
```

```
int minhaVariavel;  
minhaVariavel = 15; // Atribuímos o valor aqui.  
cout << minhaVariavel;
```

- Exemplo 3:

```
/* Aqui estamos criando uma variável do tipo inteiro  
e atribuindo o valor 15.*/
```

```
int minhaVariavel = 15;  
minhaVariavel = 10; // Decidimos mudar o valor para 10  
cout << minhaVariavel;
```

Variáveis

- Outros exemplos:

```
int meuInteiro = 5;  
float meuFloat = 5.99;  
char meuChar = 'D';  
string minhaString = "Olá";  
bool meuBooleano = true;
```

Variáveis – Tipos Primitivos

Tipo	Tamanho	Valores possíveis
bool	1 byte	True or false
unsigned short int	2 bytes	0 to 65,535
short int	2 bytes	-32,768 to 32,767
unsigned long int	4 bytes	0 to 4,294,967,295
long int	4 bytes	-2,147,483,648 to 2,147,483,647
int (16 bit)	2 bytes	-32,768 to 32,767
int (32 bit)	4 bytes	-2,147,483,648 to 2,147,483,647
unsigned int (16 bit)	2 bytes	0 to 65,535
unsigned int (32 bit)	4 bytes	0 to 4,294,967,295
char	1 byte	256 character values
float	4 bytes	1.2e-38 to 3.4e38
double	8 bytes	2.2e-308 to 1.8e308

Começando a Programar em C++

- Vocês entendem a estrutura básica de um programa em C++.
- Sabem exibir informações na tela.
- Para criarem um programa mais interessante falta apenas aprenderem a ler informações do usuário.

Conhecendo o objeto *cin*

- `cin` \Rightarrow C++ Input.
- O objeto `cin`, juntamente com o operador `>>` (extrair), é usado para ler valores do usuário.
- Especificamente, `cin` é uma variável pré-definida que lê dados do teclado usando o operador de extração .
- Abram o site do Compiler Explorer, ou o compilador de sua preferência, e copiem o código abaixo.
- Usem o campo para `stdin` (entrada padrão) no Compiler Explorer para digitar o número.

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

```
int main() {
    float x;
    cout << "Digite um número: ";
    cin >> x;
    cout << "Seu número é: " << x;
    return 0;
}
```

Conhecendo o objeto *cin*

- Vamos criar uma simples calculadora que apenas faz adições.
- Digite dois números com um espaço entre eles.

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

```
int main() {  
    int x, y;  
    int soma;  
    cout << "Digite dois números:\n";  
    cin >> x;  
    cin >> y;  
    soma = x + y;  
    cout << "A soma é: " << soma;  
    return 0;  
}
```

Conhecendo o objeto *cin*

- `cin` permite lermos *strings*.
- Declararemos o arquivo de cabeçalho para podermos usar o tipo `string` (que é uma classe, mas veremos isso mais a frente).
- O tipo `string` é usado para armazenar e manipular *strings* de texto, como palavras, frases, ou qualquer sequência de caracteres.

```
#include <iostream>
```

```
#include <string>
```

```
using namespace std;
```

```
int main() {  
    string palavra;  
    cout << "Digite uma palavra:\n";  
    cin >> palavra;  
    cout << "Você digitou: " << palavra << endl;  
    return 0;  
}
```

- Isso já é interessante, não permite que leiam mais de uma palavra.
- Mesmo que digitem várias palavras, separadas por espaços em branco, apenas uma palavra será exibida.

Conhecendo o objeto *cin*

- Agora vamos ler uma linha de texto contendo várias palavras usando a função `getline` em combinação com o objeto `cin`.

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

```
int main() {
    string linha;
    cout << "Digite uma linha de texto:\n";
    getline(cin, linha);
    cout << "Você digitou: " << linha << endl;
    return 0;
}
```

- Agora conseguem imprimir uma frase inteira.

Strings e Concatenações

- Concatenações são operações comuns que fazemos em strings.
- Você pode concatenar strings utilizando o operador `+` ou o método `append()` da classe `std::string`.
- Não vamos nos adiantar no conteúdo, então, por enquanto, considerem apenas a primeira alternativa.

```
#include <iostream>
```

```
#include <string>
```

```
using namespace std;
```

```
int main() {
```

```
    string str1 = "Olá, ";
```

```
    string str2 = "Mundo!";
```

```
    string resultado = str1 + str2; // Concatenação usando o operador +
```

```
    cout << resultado << endl; // Imprime "Olá, Mundo!"
```

```
    return 0;
```

```
}
```

Começando a Programar em C++

Exercícios (1).

- ⇒ Cada programa deve ser exibido as informações solicitadas igual o exemplo dado abaixo do exercício.
- Escreva um programa que solicite ao usuário que digite seu nome e, em seguida, imprima uma mensagem de boas-vindas com o nome digitado.

Exemplo:

Digite seu nome: Maria
Olá, Maria! Seja bem-vinda!

- Escreva um programa que converta uma temperatura em graus Celsius para Fahrenheit. O programa deve solicitar ao usuário que digite a temperatura em Celsius e, em seguida, imprimir o resultado da conversão (descubram como!).

Exemplo:

Digite a temperatura em Celsius: 25
25 graus Celsius equivalem a 77 graus Fahrenheit.

- Escreva um programa que calcule a média de três números inseridos pelo usuário. O programa deve solicitar ao usuário que digite os três números e, em seguida, imprimir a média dos números.

Exemplo:

Digite o primeiro número: 5
Digite o segundo número: 8
Digite o terceiro número: 12
A média dos números é: 8.33333

Começando a Programar em C++

Exercícios (2).

- ⇒ Cada programa deve ser exibido com as informações solicitadas igual o exemplo dado abaixo do exercício.
- Escreva um programa que peça ao usuário para digitar a base e a altura de um triângulo e, em seguida, calcule e imprima a área do triângulo (descubram como!).

Exemplo:

Digite a base do triângulo: 6
Digite a altura do triângulo: 4
A área do triângulo é: 12

- Escreva um programa que solicite ao usuário que digite o primeiro termo (a), a razão (r) e o número de termos (n) de uma progressão aritmética. O programa deve calcular e imprimir a soma dos n termos da P.A (descubram como!).

Exemplo:

Digite o primeiro termo (a): 3
Digite a razão (r): 2
Digite o número de termos (n): 5
A soma dos 5 termos da P.A. é: 35