



PCS311

**Laboratório de Programação
Orientada a Objetos para
Engenharia Elétrica**

Aula 1: Introdução

Escola Politécnica da Universidade de São Paulo

Agenda

1. Informações gerais sobre a disciplina
2. Visão geral da OO
3. Visão geral da linguagem C++
4. Programa básico em C++

Informações gerais sobre a disciplina

Objetivos

- Conceitos de Orientação a Objetos (OO)
- Aspectos Básicos de Programação
 - Estilo de código
 - Programação defensiva e tratamento de erros
 - Manipulação de arquivos
- Apresentação da Linguagem C++

Programa

Dia	Aula	Assunto
01-07/08	1	Introdução
08-14/08	2	Ponteiros, Testes e Depuração
15-21/08	3	Conceitos Básicos de OO
22-28/08	4	Encapsulamento
11/09		Sem aula (Semana de provas)
12-18/09	5	Construtor e Destrutor
19-25/09	6	Herança e Polimorfismo I
26-02/10	7	Herança e Polimorfismo II
03-09/10	8	Classe Abstratas e Herança Múltipla
16/10	P1	Semana de provas
17-23/10	9	Programação Defensiva
24-04/11	10	Persistência em Arquivos
31-11/11	11	Namespace e STL
07-18/11	12	Finalização do EP2
27/11	P2	Semana de provas
04/12	SUB	Semana de provas
11/12	REC	Recuperação

Organização

- Apresentações e material no e-Disciplinas
 - <https://edisciplinas.usp.br/course/view.php?id=69732>
 - Disponíveis na 2ª feira
- Professores
 - Jorge Rady de Almeida Júnior T2M
 - Ricardo Luis de Azevedo da Rocha T2T
 - Lucia Vilela Leite Filgueiras T3T
 - Solange Nice Alves de Souza T4M / T4T
 - Kechi Hirama T5M
 - Fabio Levy Siqueira T5T
 - Pedro Luiz Pizzigatti Corrêa T6T

Organização

- Monitores / Técnicos
 - Bernardo Martins Ferreira
 - Felipe Desiglo Ferrare
 - Michelet del Carpio Chávez

Bibliografia

■ Básica

- BUDD, T. **An Introduction to Object-Oriented Programming**. 3rd Edition. Addison-Wesley. 2001.
- LAFORE, R. **Object-Oriented Programming in C++**. 4th Edition. SAMS. 2002.
- SAVITCH, W. **C++ Absoluto**. Addison-Wesley. 2004.

■ Complementar

- STROUSTRUP, B. **The C++ Programming Language**. 4th Edition. Addison-Wesley, 2013.

Atividades

- Em sala de aula
 - Teoria \approx 50min
 - Prática = 100min
 - Exercícios individuais
 - Correção automática (Judge)
 - No **Judge**, limite de 3 submissões sem penalização
 - Para cada submissão subsequente, a nota máxima do exercício será decrementada em 2 pontos:
 - 4ª. Submissão, nota máxima 8
 - 5ª. Submissão, nota máxima 6
 - ...

Atividades

- Fora de sala de aula
 - 2 EPs
 - Desenvolvimento incremental
 - Realizado **em duplas**
 - Alunos de uma mesma turma
 - Correção Automática e **nota adicional**

Observações

- Não será permitida a entrada no laboratório após um atraso de 20 minutos
- Não são autorizadas trocas de turma (casos excepcionais devem ser encaminhados por e-mail ao coordenador)

Avaliação

- $MF = (2*ME + 3*MEP + 5*MP) / 10;$
if $(MP < 5)$ $MF = MP;$
 - $ME = (E1 + \dots + E11 - \min(E1, \dots, E11)) / 10$
 - $MEP = (EP1 + 2*EP2) / 3$
 - onde $EPI = 0.3 * Qi + 0.7 * AUT,$
 - Qi = nota de especificação / qualidade / interface
 - $AUTi$ = nota da correção automática
 - $MP = (P1 + P2) / 2$
- Prova Substitutiva é fechada

Código de ética da USP

- Disponível em:
[http://www.mp.usp.br/sites/default/files/arquivosanexos/codigo de etica da usp.pdf](http://www.mp.usp.br/sites/default/files/arquivosanexos/codigo_de_etica_da_usp.pdf)
- Especificamente na disciplina
 - Alunos não devem submeter programas por outros alunos
 - Alunos não submeter programas feitos por outros!
 - Alunos devem fazer a prova **individualmente**
 - Código do EP não pode ser copiado de outro grupo
 - Nem partes do código: **plágio**
- **Será cobrado!**

Visão Geral de OO

Desenvolvimento de Software

- Desenvolver software não envolve só a linguagem de programação
 - Métodos, arcabouços (*frameworks*), bibliotecas, ferramentas, etc.
- Um aspecto importante é o *paradigma de programação*

“Forma de conceituar o que significa realizar computação e como tarefas executadas no computador devem ser estruturadas e organizadas.” (Budd, 2001)

- A solução de um problema computacional é influenciada pelo paradigma seguido
 - Facilidade / dificuldade de representação

Paradigmas de Programação

- Existem diversos paradigmas de programação
 - *Exemplo de paradigmas e linguagens*
 - Imperativo: Pascal e Cobol
 - Funcional: Lisp, Haskell e Scala
 - Lógico: Prolog e Datalog
 - **Orientado a Objetos: C++, C#, Java e Python**
 - Orientado a Eventos: bastante usado para interfaces gráficas
 - Declarativo: SQL e HTML
- Algumas linguagens são *multiparadigma*
 - *Linguagens: C++, Python*

Histórico da OO

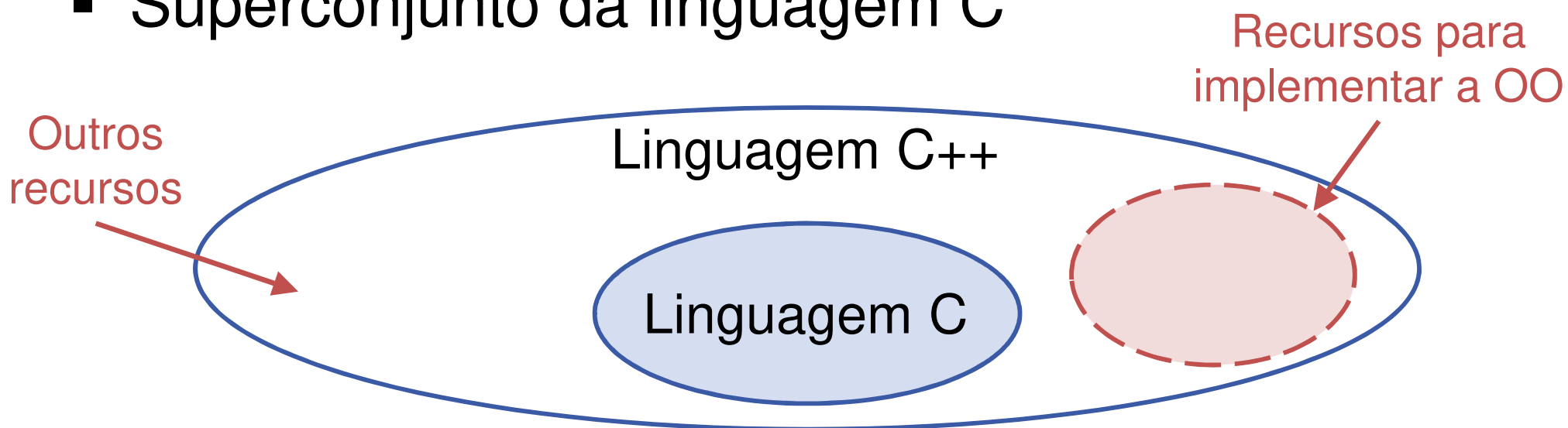
- *Centro de Computação Norueguês*
 - Simula: 1ª Linguagem OO (1967)
 - <http://www.uio.no/om/aktuelt/rektorbloggen/2017/50-years-anniversary-of-simula-the-first-object-or.html>
 - Ideia motivou outras linguagens
- Alan Kay (*Xerox PARC*)
 - Linguagem que fosse fácil de entender por usuários
 - Smalltalk (disponibilizada em 1980)
- Bjarne Stroustrup (*Bell Labs*)
 - Extensão de C para usar os conceitos de *Simula*
 - C++ (1983)
- Popularização na década de 1990

C++

- Linguagem de propósito geral
- Ênfase em software básico (software de sistemas)
 - Nível do hardware
 - Controle do programador
 - Permite a geração de códigos eficientes
- Orientado a Objetos
 - Chamado originalmente de "C com classes"
 - Na realidade é *multiparadigma*
 - Paradigma Imperativo
 - Paradigma Orientado a Objetos
 - Programação genérica (*templates*)

C++

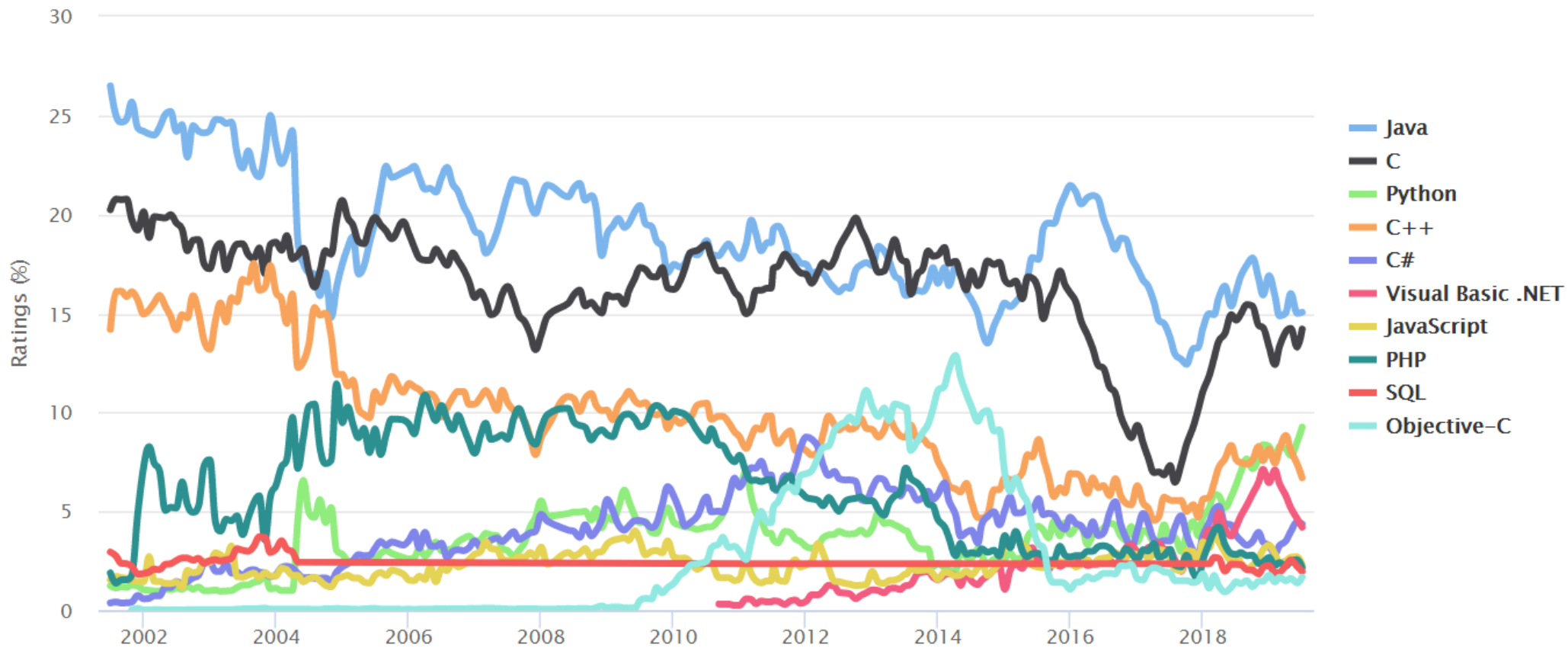
- Superconjunto da linguagem C



- Foco da disciplina: recursos para OO
 - Veremos alguns dos outros recursos

- Versão atual: C++17

Popularidade de C++



C++: 6,75%
(jul. 2019)

Fonte: <https://www.tiobe.com/tiobe-index/>

Compiladores e Ambientes

- Alguns compiladores
 - GCC (Linux)
 - *Windows*: MinGW (<http://www.mingw.org>)
 - Intel C++ Compiler
- Alguns ambientes de programação (IDE)

- **Code::Blocks**

- <http://www.codeblocks.org>

- Netbeans (Oracle)

- <https://netbeans.org>

- Eclipse

- <http://eclipse.org>

- Visual Studio (Microsoft)

- <https://www.visualstudio.com>



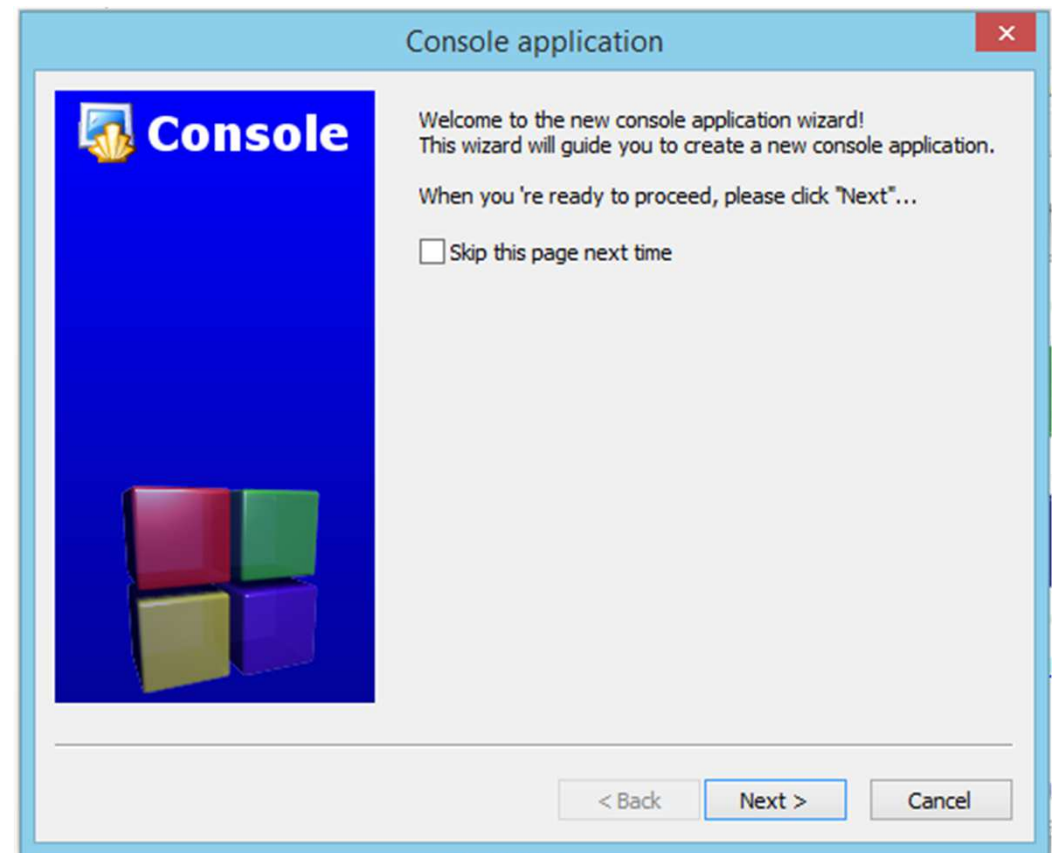
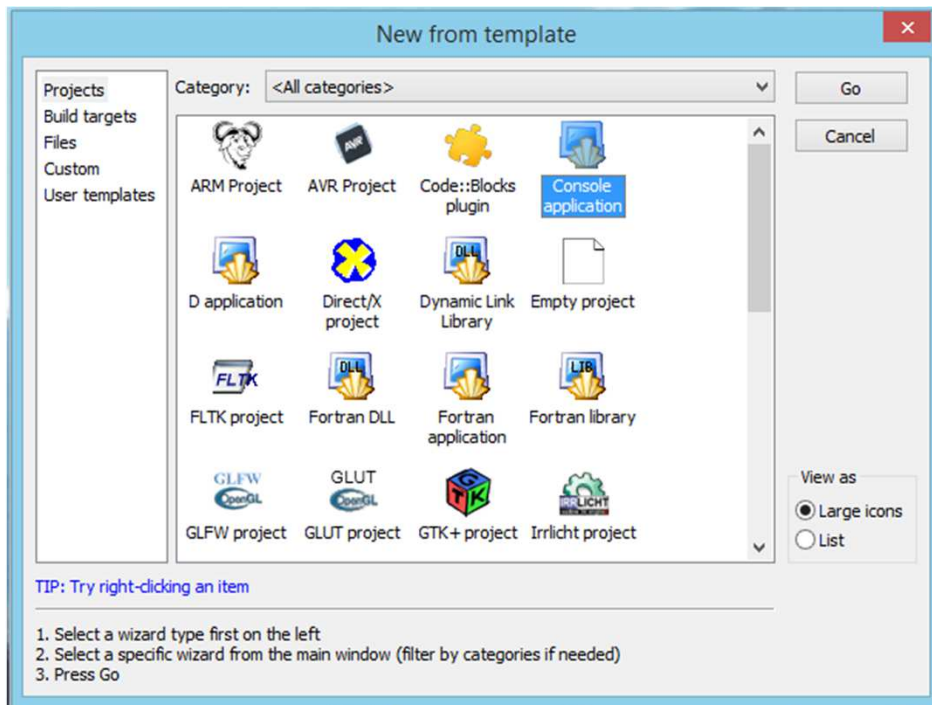
Primeiro Exemplo

```
1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {
7      cout << "Hello world!" << endl;
8      return 0;
9  }
```

- Crie um projeto no Code::Blocks
 - File → New → Project
 - Ou atalho “*Create a new project*”

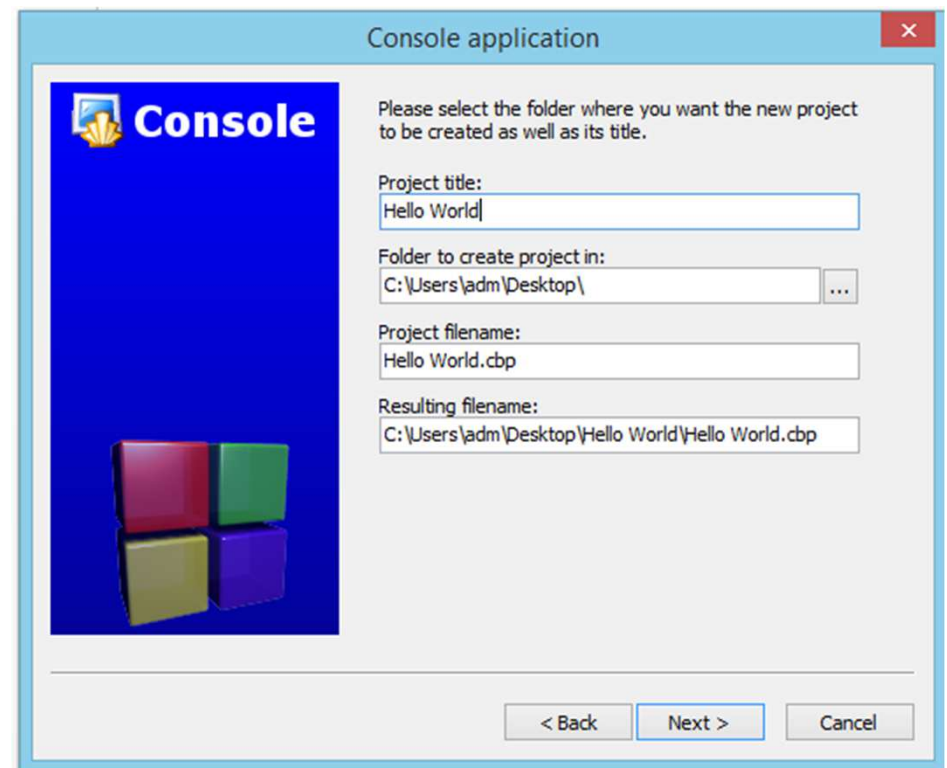
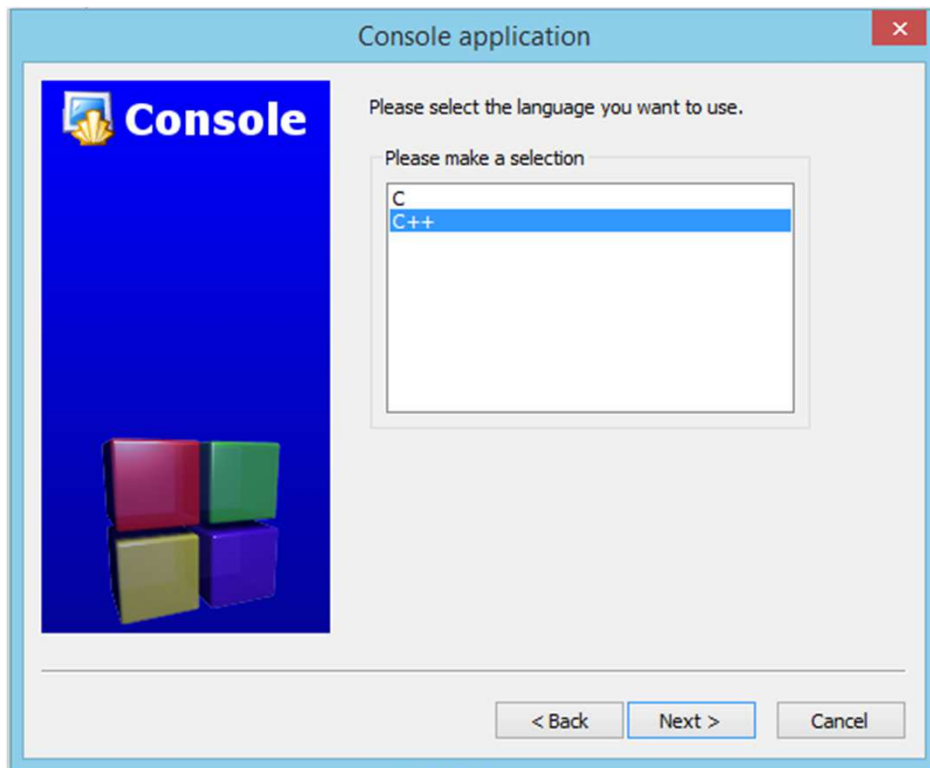
Primeiro Exemplo

- Escolha a categoria *Console Application*
 - Ação Go → Next



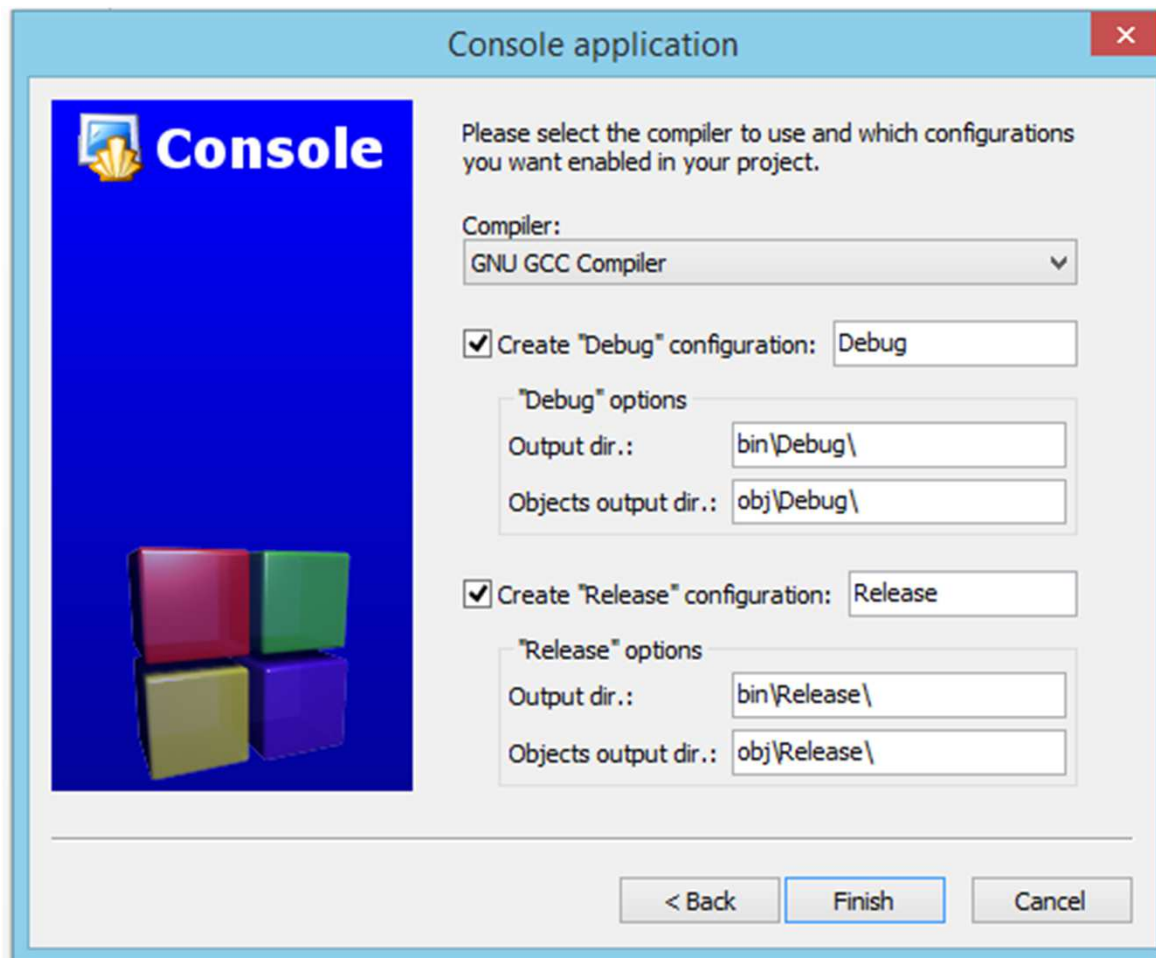
Primeiro Exemplo

- Escolha a linguagem C++ → Next
- Escolha o nome e a pasta do projeto → Next



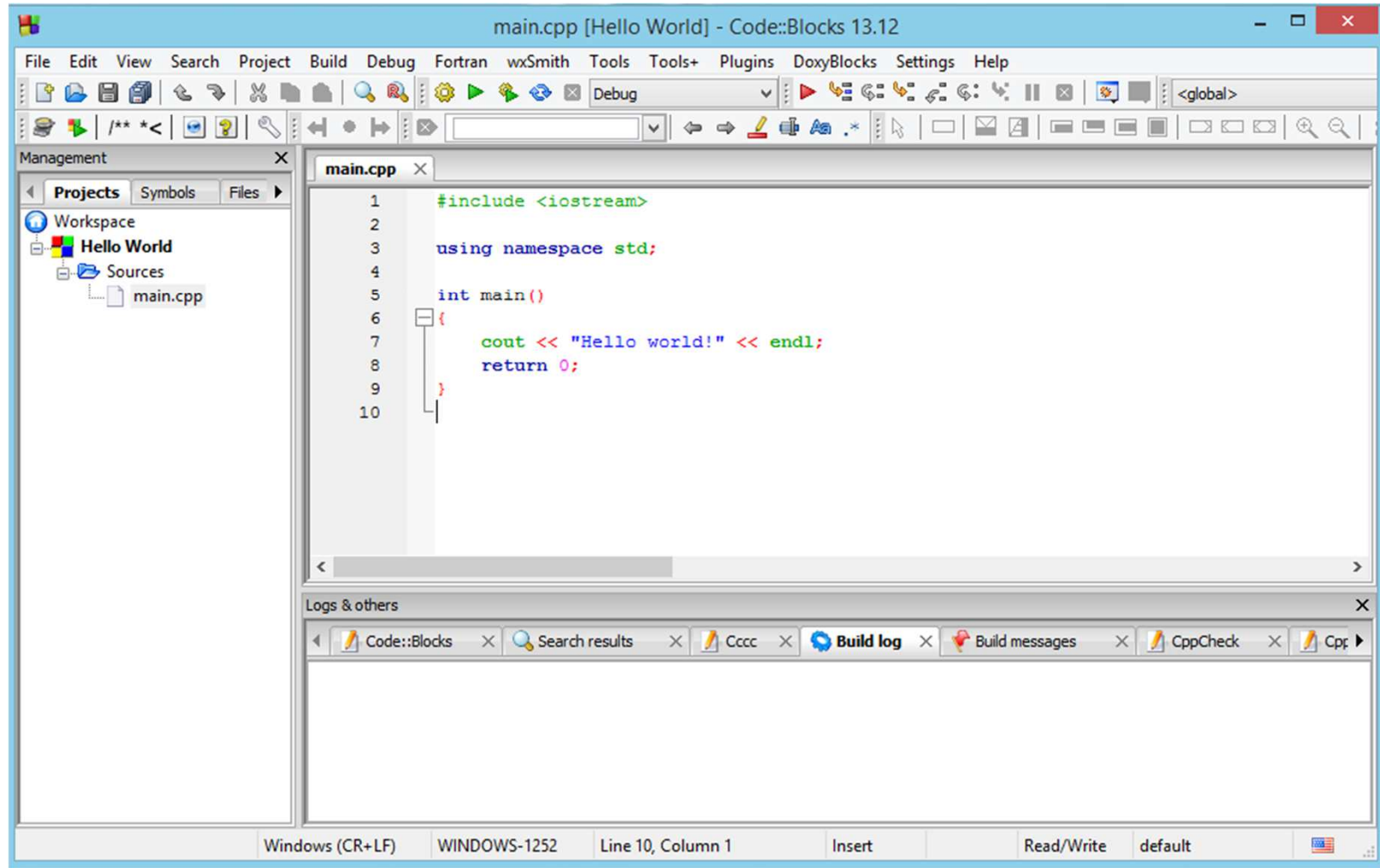
Primeiro Exemplo

- Escolha GNU GCC Compiler (não altere as configurações *default*) → Finish



Primeiro Exemplo

- Projeto Hello World e arquivo main.cpp

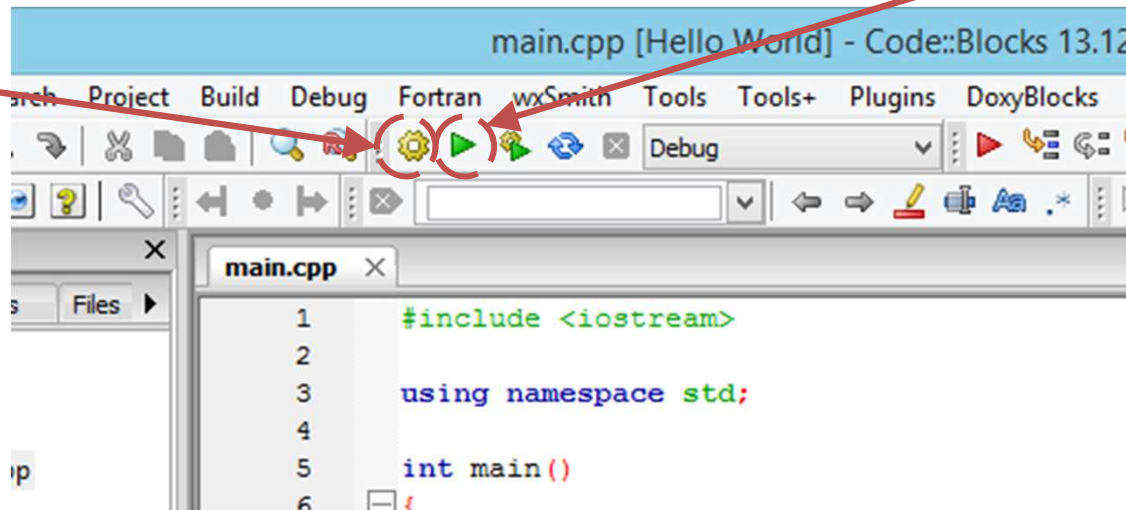


Primeiro Exemplo

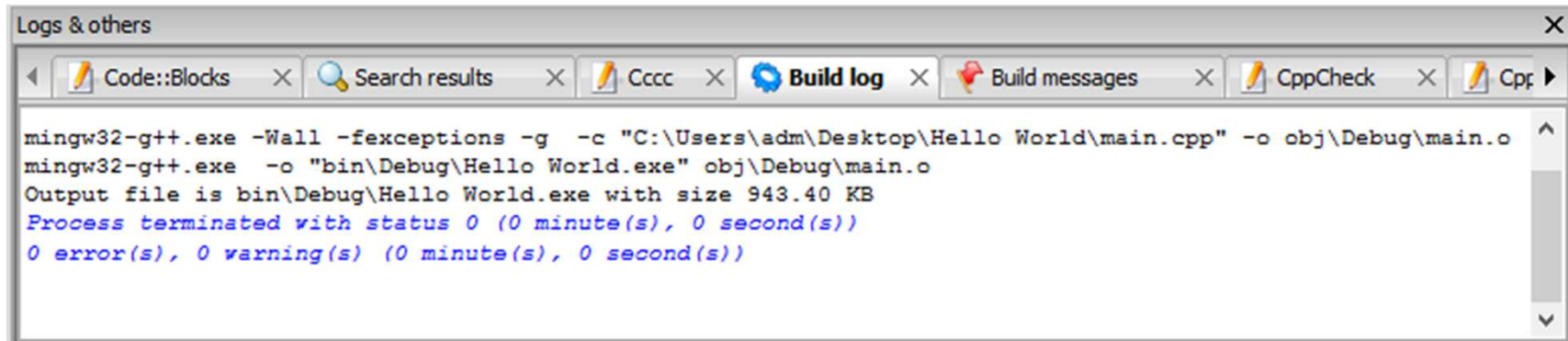
- Compile e execute o programa

Compilar
(CTRL-F9)

Executar
(CTRL-F10)

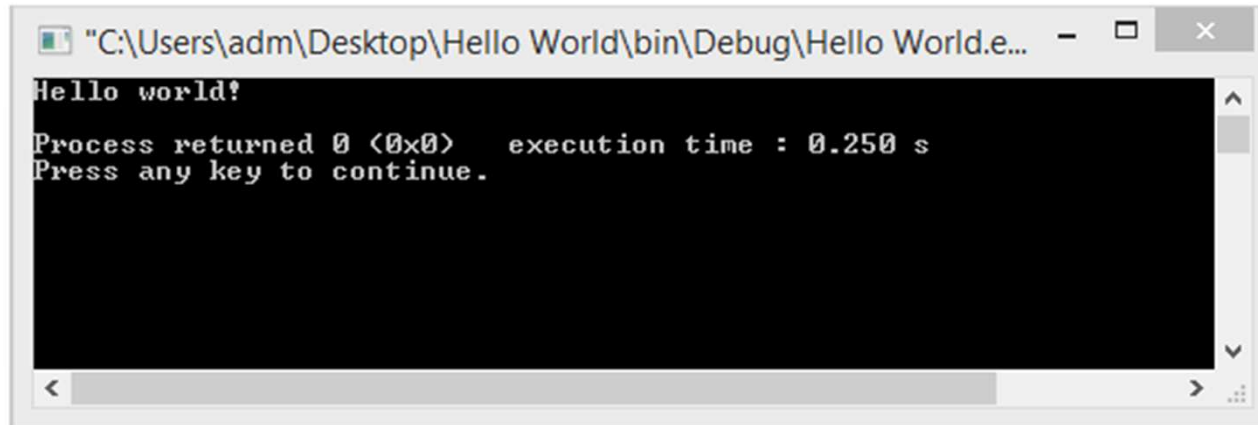


- Resultado da compilação



Primeiro Exemplo

- Saída do console



A screenshot of a Windows command prompt window. The title bar shows the file path: "C:\Users\adm\Desktop\Hello World\bin\Debug\Hello World.e...". The window has standard minimize, maximize, and close buttons. The main area is black with white text. The text displayed is: "Hello world!", "Process returned 0 (0x0) execution time : 0.250 s", and "Press any key to continue.". There is a scrollbar on the right side of the text area.


```
"C:\Users\adm\Desktop\Hello World\bin\Debug\Hello World.e...  
Hello world!  
Process returned 0 (0x0) execution time : 0.250 s  
Press any key to continue.
```

Instalação do Code::Blocks


- (Para instalar na sua casa)
- Site do Code::Blocks: <http://www.codeblocks.org/downloads>
 - Escolha *Download the binary release*
 - Escolha **codeblocks-17.12mingw-setup.exe**
 - (Esse build já vem com o compilador C++)

Quick links

- FAQ
- Wiki
- Forums
- Forums (mobile)
- Nightlies
- Ticket System
- Browse SVN
- Browse SVN log

 **Windows XP / Vista / 7 / 8.x / 10:**

File	Date	Download from
codeblocks-17.12-setup.exe	30 Dec 2017	Sourceforge.net
codeblocks-17.12-setup-nonadmin.exe	30 Dec 2017	Sourceforge.net
codeblocks-17.12-nosetup.zip	30 Dec 2017	Sourceforge.net
codeblocks-17.12mingw-setup.exe	30 Dec 2017	Sourceforge.net



Visão Geral do C++

Variáveis

■ Declaração

- Tipo, identificador e valor (*opcional*)

```
int numeroDePessoas;  
bool confirmado = true;  
int maior = 100, menor = 0;  
double x, y = 50.0;
```

- Variáveis podem ser declaradas em qualquer parte do bloco
 - **Bloco**: conjunto de comandos entre "{" e "}"

Tipos Primitivos

- Principais tipos (alguns podem ser *unsigned*)
 - (O tamanho em bytes exato depende do compilador)

Tipo	Valores	Bytes	Exemplo
bool	Booleano	1	true, false, 1, 0
char	Caractere	1	'a', ';', 125
short	Número	2	0, -1, 15000
int	Número	4	0, -1, 15000
long	Número	4	0, -1, 1E10
float	Ponto flutuante	4	-1.45E-30
double	Ponto flutuante	8	1.9E100

Condição e Laços

■ Condição

```
if (x == 0) {  
    // ...  
} else if (x > 0) {  
    // ...  
} else {  
    // ...  
}
```

■ Laços

- While

```
while (x > 0) {  
    // ...  
}
```

- Do-while

```
do {  
    // ...  
} while (x > 0);
```

- For

```
for (int i = 0; i < 10 ; i++) {  
    // ...  
}
```


Operadores Lógicos

- Principais operadores lógicos

Operador	Descrição
&&	E lógico
	Ou lógico
!	Negação

- Exemplo*

```
bool encontrado = false;
int x = 0, y = 0;
...

if (!encontrado && (x > 0 || y > 5)) {
    ...
}
```

Funções

■ Definição

Tipo de retorno Nome da função Parâmetros (separados por vírgula)

Corpo da função (bloco)

```
int processaElementos(int elementos[], int tamanho) {  
    ...  
}
```

■ Chamada de uma *função*

```
retorno = processaElementos(vetor, 10);
```

■ Retorno de valores

```
void f() {  
    ...  
    return;  
    ...  
}
```

Sem retorno

```
int g() {  
    ...  
    return 1;  
    ...  
}
```

Com retorno (inteiro)

Comentários

- Dois tipos de comentários

- //

- Comenta do “//” em diante até o fim da linha

```
x++; // O resto da linha é comentado
```

- /* e */

- Comenta o texto entre os /* e */
 - Permite comentar várias linhas

```
/*  
    Exercício 1  
    Autor: Meu nome  
    Data: 01/08/2019  
*/
```

```
/* Comentário */ x++;
```

Vetor (ou arranjo)

- É um conjunto ordenado de variáveis de um mesmo tipo

- Exemplo*

```
int y[5];
```

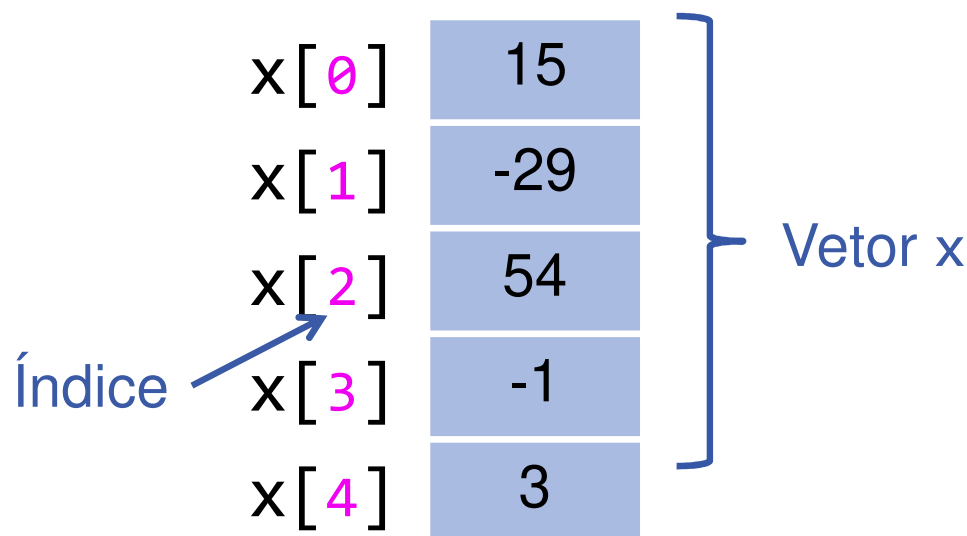
Declara um vetor y (os valores não estão inicializados)

```
int x[] = {15, -29, 54, -1, 3};
```

Declara um vetor x, inicializando os valores

```
int m[5][5];
```

Declara uma matriz 5x5



Vetor (ou arranjo)

- Acesso aos elementos do vetor

```
numeros[0] = 10;
```



Atribui o valor 10 à posição 0 do vetor (1ª posição)

```
x = numeros[5];
```



Atribui o valor da posição 5 do vetor à variável x

Programa Básico em C++

cin e cout

- Entrada e saída padrão estão em **iostream**
 - Necessário o `#include` e o `using namespace`

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

- Entrada padrão: `cin`
 - Texto e variáveis devem ser separados por `>>`
 - Chamado de "obter de"
 - Funciona com os principais tipos
 - *Exemplo*

```
7  int a = 0;
8  cin >> a;
```

EX01

- O inteiro digitado pelo usuário é colocado na variável `a`

cin e cout

■ Saída padrão: cout

- Texto e variáveis devem ser separados por <<
 - Chamado de "colocar em"
- endl é *equivalente* a "\n"
- *Exemplos*

```
13 int i = 5;  
14 cout << "Ola" << endl;  
15 cout << i;
```

EX01



Saída

Ola
5

```
19 int x = 5, y = 6;  
20 cout << "x vale " << x << " e y vale " << y << endl;
```

EX01

texto

x

texto

y

Pula linha

Saída

x vale 5 e y vale 6

Programa Básico

```
1  #include <iostream>
2  using namespace std;
3
4  int multiplicar (int x, int y) {
5      return x * y;
6  }
7
8  int main() {
9      int x = 5, y = 3;
10     cout << multiplicar (x, y) << endl;
11     return 0;
12 }
```

Inclusões e outras diretivas

Funções

EX03

- **main**: ponto de entrada do programa
 - Sempre coloque um **return**, **0** indica sucesso
 - Um projeto só pode ter um único main

Programa Básico

- Se a função for usada antes de ser definida, é necessário criar um **protótipo**
 - Apenas *assinatura* da função

```
1  #include <iostream>
```

```
2  using namespace std;
```

```
3
```

```
4  int multiplicar (int x, int y);
```

EX04

← Protótipo (declaração)

```
5
```

```
6  int main() {
```

```
7      int x = 5, y = 3;
```

```
8      cout << multiplicar (x, y) << endl;
```

← Uso

```
9      return 0;
```

```
10 }
```

```
11
```

```
12 int multiplicar (int x, int y) {
```

← Definição da função

```
13     return x * y;
```

```
14 }
```

string

- Não é *somente* um vetor de caracteres...
- Necessário o `#include` e o `using namespace`
- *Exemplo*

```
1  #include <iostream>
2  #include <string>
3  using namespace std;
4
5  int main() {
6      string nome = "Jose";
7      nome = "Pedro";
8      char inicial = nome[0];
9      cout << inicial << endl;
10     return 0;
11 }
```

(Necessário para o *cout*)

Necessário para usar *string*

Valor inicial

Atribuindo novo valor

Obtendo um caractere

EX02

- Existem diversas “funções” auxiliares (*métodos*)

Bibliografia

- BUDD, T. **An Introduction to Object-Oriented Programming**. 3rd Edition. Addison-Wesley. 2001. Cap. 1.
- LAFORE, R. **Object-Oriented Programming in C++**. 4th Edition. SAMS. 2002. Cap. 2, 3, 4 e 5.