



**PCS3111**

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

**Aula 1: Introdução**

Escola Politécnica da Universidade de São Paulo

# Agenda

1. Visão geral da OO
2. Visão geral da linguagem C++
3. Programa básico em C++

# 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, Haskel 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

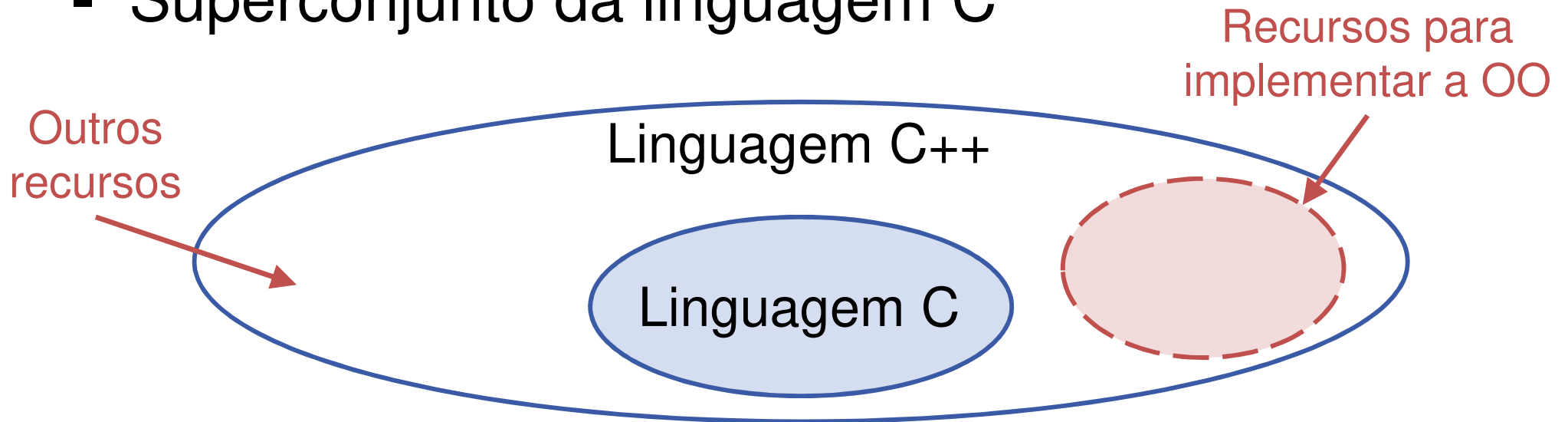
- *Centro de Computação Norueguês*
  - Simula: 1ª Linguagem OO (1967)
    - <http://www.uio.no/om/aktuelt/rektorbloggen/2017/50-years-anniversary-of-simula-the-first-object-or.html>
  - Ideia motivou outras linguagens
- Alan Kay (*Xerox PARC*)
  - Linguagem que fosse fácil de entender por usuários
  - 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

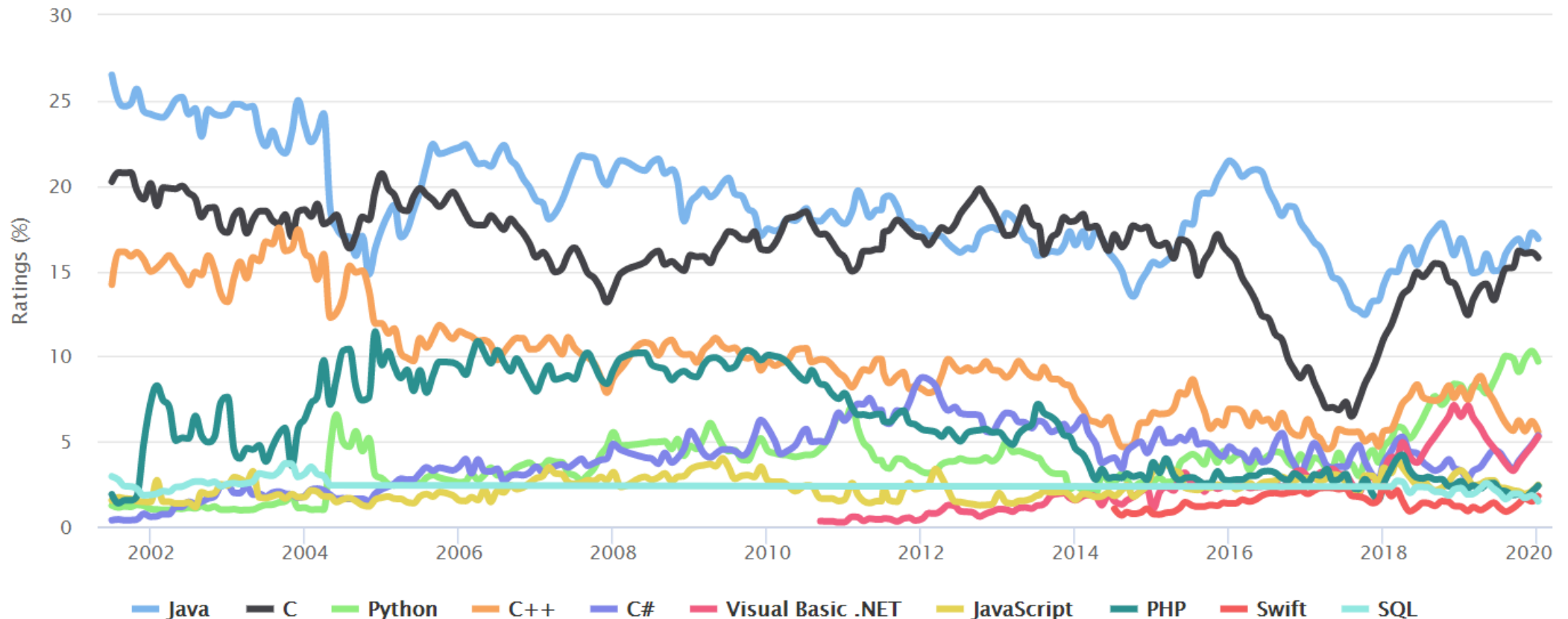
- Versão atual: C++17



# Popularidade de C++

## TIOBE Programming Community Index

Source: [www.tiobe.com](http://www.tiobe.com)



**C++: 5,57%**

(jan. 2020)

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>



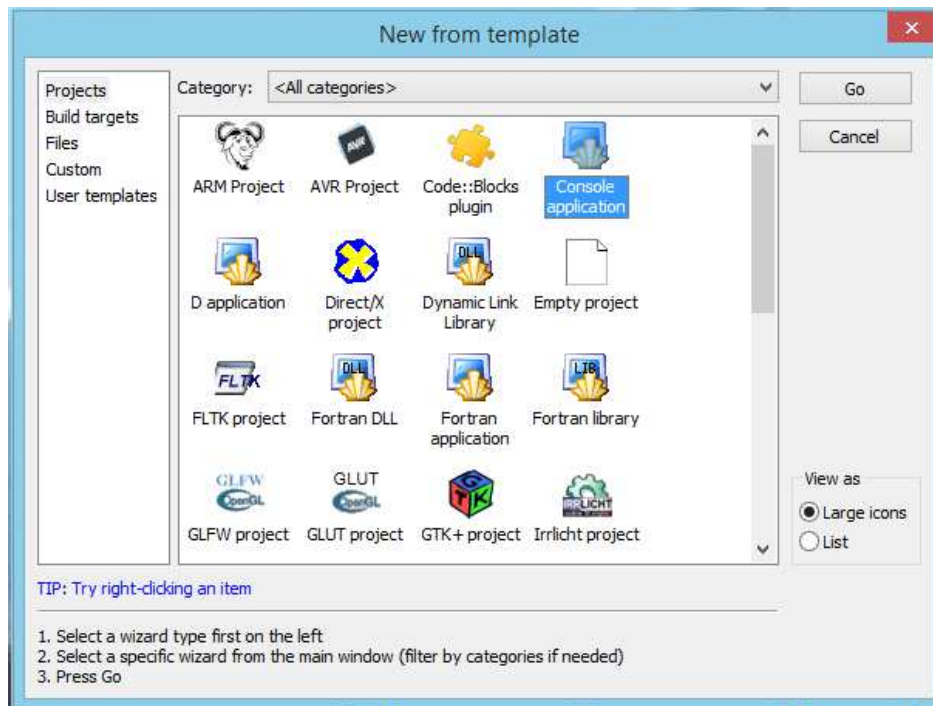
# Primeiro Exemplo

```
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*”

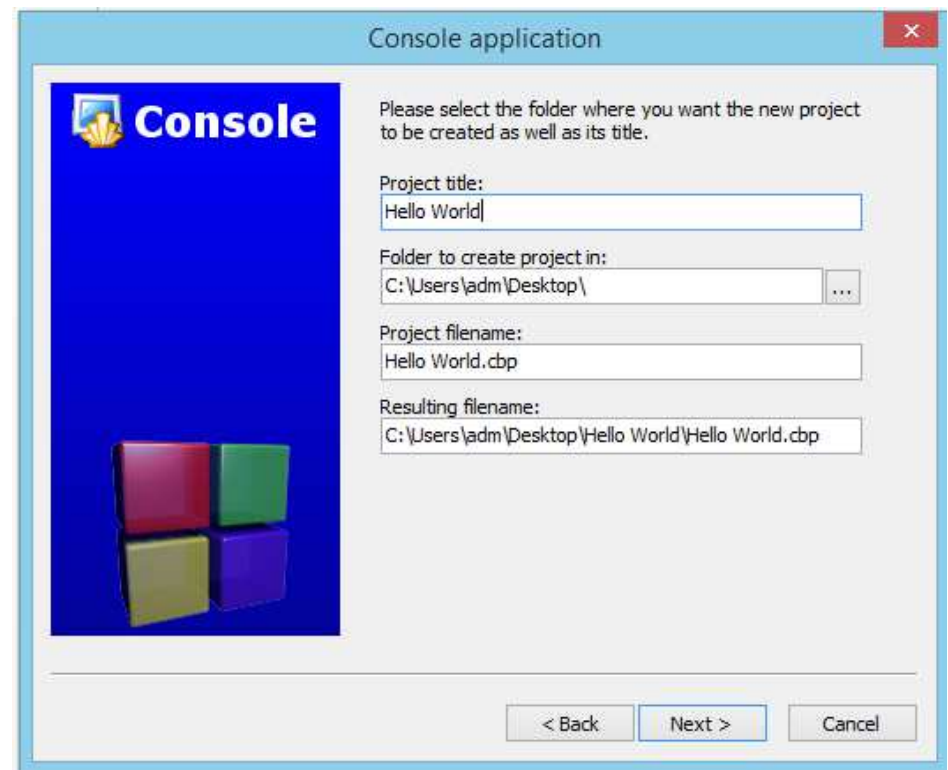
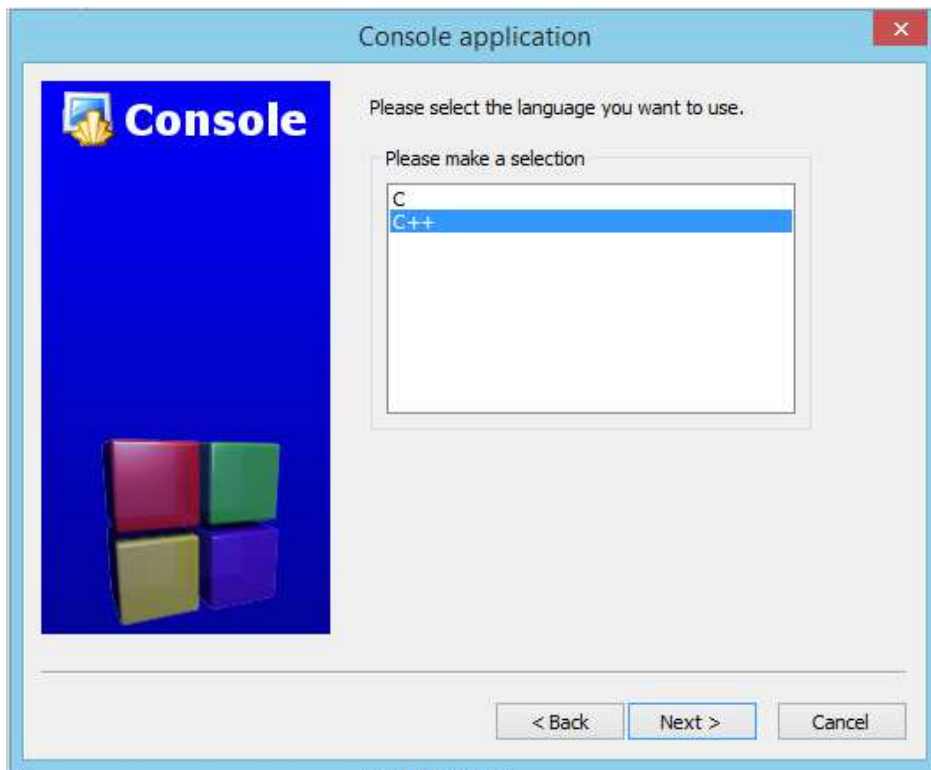
# Primeiro Exemplo

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



# Primeiro Exemplo

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



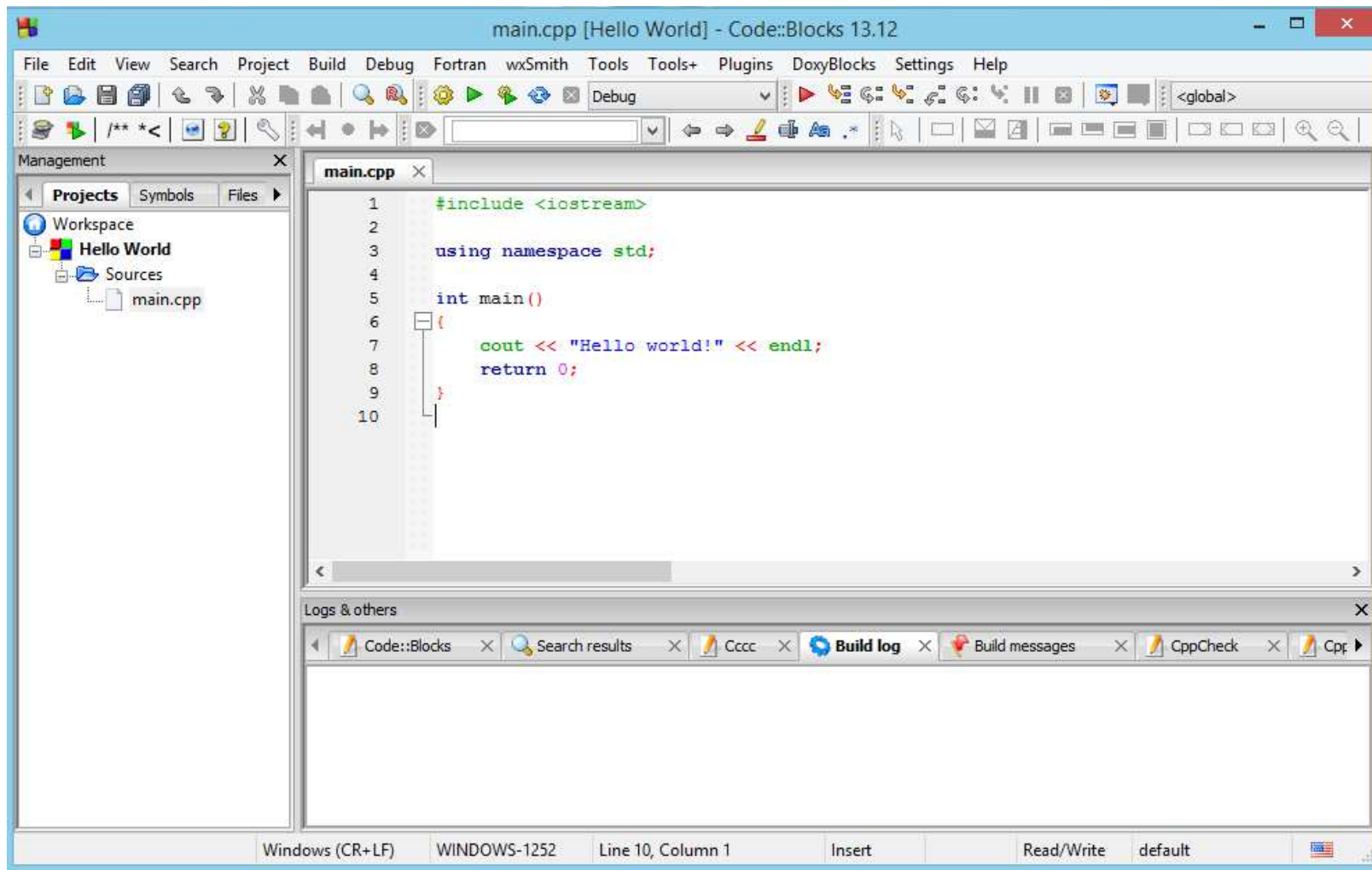
# Primeiro Exemplo

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



# Primeiro Exemplo

- Projeto Hello World e arquivo main.cpp

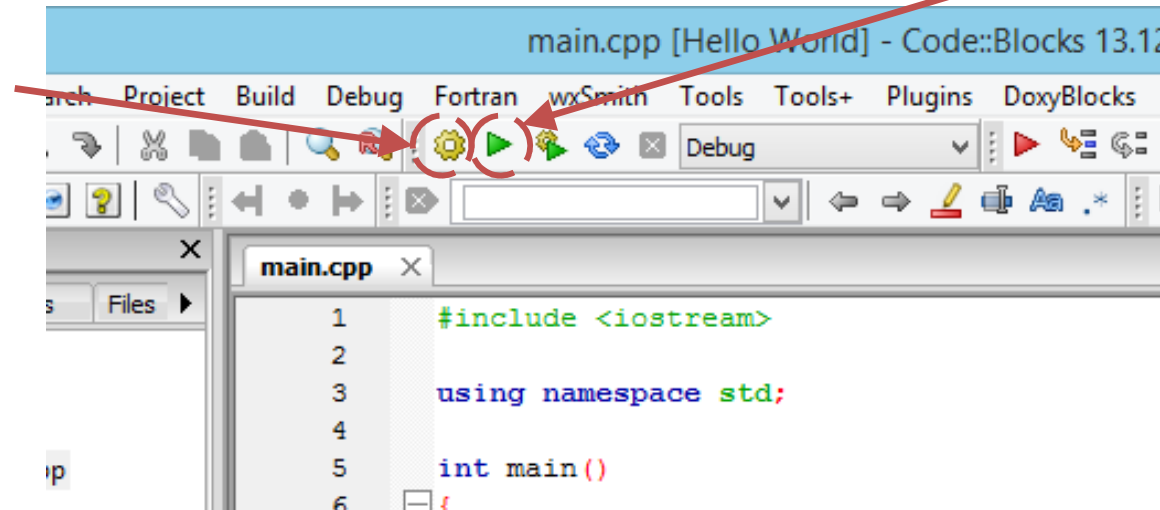


# Primeiro Exemplo

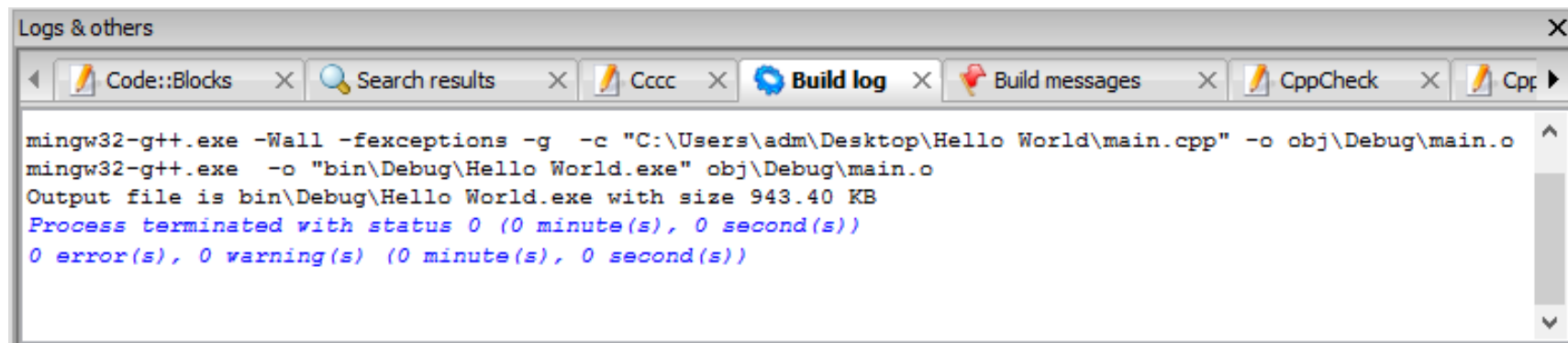
- Compile e execute o programa

Compilar  
(CTRL-F9)

Executar  
(CTRL-F10)



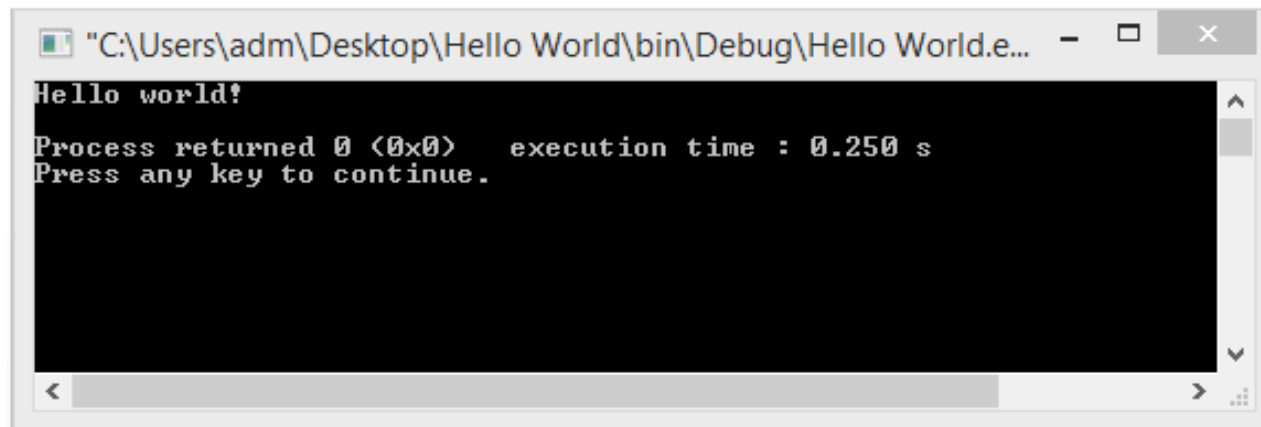
- Resultado da compilação





# 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 a black background with white text. The text displayed is: "Hello world!", "Process returned 0 (0x0) execution time : 0.250 s", and "Press any key to continue.". The window includes standard Windows window controls (minimize, maximize, close) and a scrollbar on the right side.

```
"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++)



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

File	Date	Download from
codeblocks-17.12-setup.exe	30 Dec 2017	FossHUB or Sourceforge.net
codeblocks-17.12-setup-nonadmin.exe	30 Dec 2017	FossHUB or Sourceforge.net
codeblocks-17.12-nosetup.zip	30 Dec 2017	FossHUB or Sourceforge.net
codeblocks-17.12mingw-setup.exe	30 Dec 2017	FossHUB or Sourceforge.net
codeblocks-17.12mingw-nosetup.zip	30 Dec 2017	FossHUB or Sourceforge.net
codeblocks-17.12mingw_fortran-setup.exe	30 Dec 2017	FossHUB or 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
==	Igual
!=	Diferente

- 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: 01/02/2020  
*/
```

```
/* Comentário */ x++;
```

# Vetor (ou arranjo)

- É um conjunto ordenado de variáveis de um mesmo tipo

- Exemplo*

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

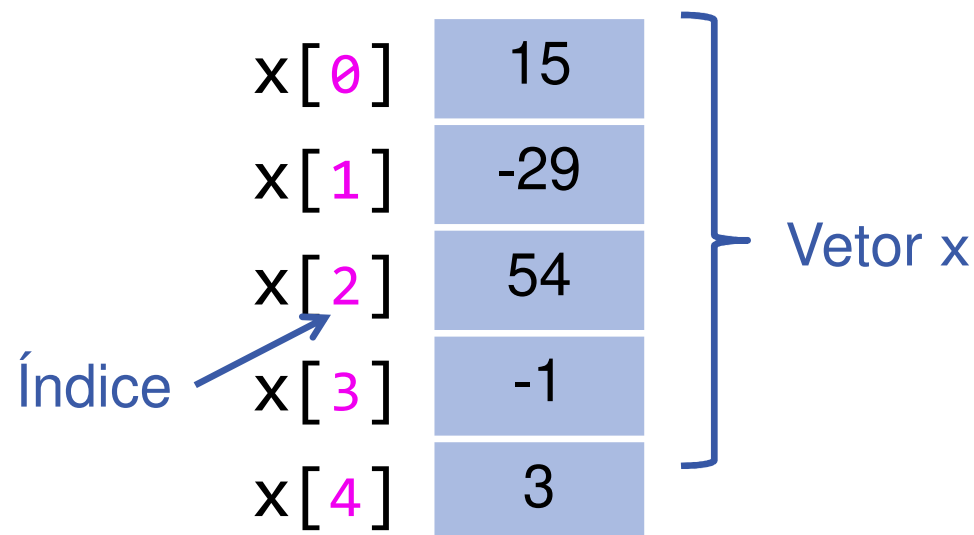
Declara um vetor y (os valores não estão inicializados)

```
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;
```

EX04

```
3
4  int multiplicar (int x, int y);
```

← 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**. 3<sup>rd</sup> Edition. Addison-Wesley. 2001. Cap. 1.
- LAFORE, R. **Object-Oriented Programming in C++**. 4<sup>th</sup> Edition. SAMS. 2002. Cap. 2, 3, 4 e 5.