



PCS3111

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

Aula 1: Introdução

Escola Politécnica da Universidade de São Paulo

Agenda

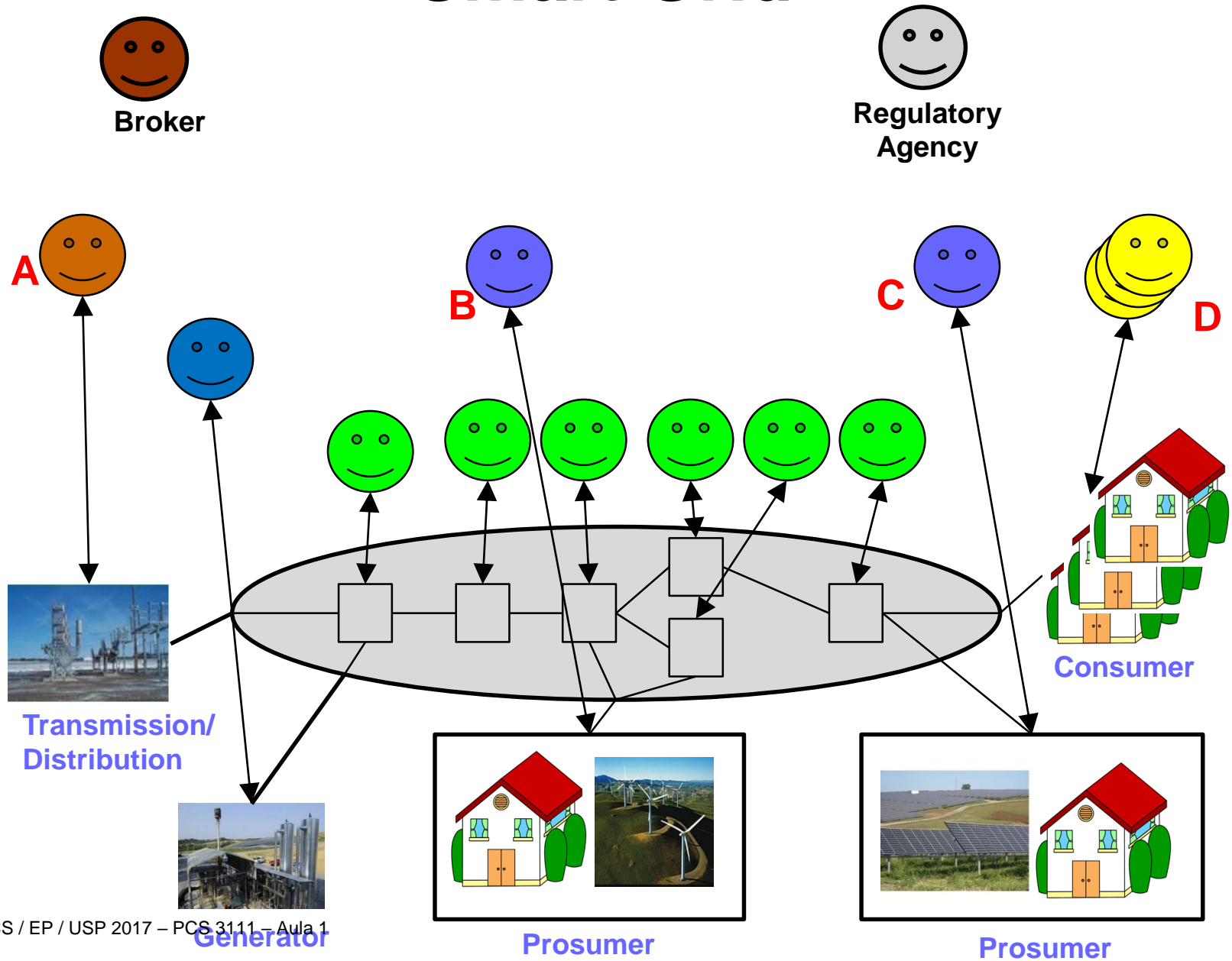
1. Informações gerais sobre a disciplina
2. Visão geral da OO
3. Visão geral da linguagem C++
4. cin e cout
5. string
6. Programa básico em C++
7. Apresentação do ambiente de programação
8. Ferramentas e processo de compilação

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++
- Problemas de Engenharia Elétrica
 - Smart Grid (Redes Inteligentes)
 - https://www.smartgrid.gov/files/sg_introduction.pdf

Smart Grid



Programa

Semana	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
30/08		P1
04-08/09		Semana da Pátria
14-20/09	5	Construtor e Destrutor
21-27/09	6	Herança e Polimorfismo I
28/09-04/10	7	Herança e Polimorfismo II
05-11/10	8	Classe Abstratas e Herança Múltipla
18/10		P2
23-27/10	9	Programação Defensiva
06/11-10/11	10	Persistência em Arquivos
11/11+21/11-24/11	11	Namespace e STL
27/11-01/12	12	Tópicos Especiais
06/12		P3
13/12		PSUB

Organização

- Apresentações e material no Moodle do Stoa
 - <http://disciplinas.stoa.usp.br>

- Professores

- | | |
|-------------------------------------|---------|
| • Fábio Levy Siqueira | T4T |
| • Jaime Simão Sichman (coordenador) | T4M |
| • Kechi Hirama | T3T |
| • Lucia Vilela Leite Filgueiras | T2M/T2T |
| • Maria Alice Grigas Ferreira | T5M |
| • Pedro Luiz Pizzigatti Corrêa | T6T |
| • Reginaldo Arakaki | T5T |
| • Solange Nice Alves da Silva | T6M |

Organização

- **Monitores / Técnicos**
 - Igor Conrado Alves de Lima
 - Márcio Fernando Stabile Jr.
 - Michelet del Carpio Chavez
 - Victor Alberto Romero

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

As atividades realizadas no curso serão as seguintes:

- Em sala de aula

- **Teoria** \approx 50min
- **Prática** = 100min
 - Exercícios individuais
 - Entregues no final da aula
- Correção Automática (Sharif Judge) e **nota de acompanhamento pelo professor**

- Fora de sala de aula

- 3 EPs
- Realizado em duplas
- Correção Automática (Sharif Judge) e **nota adicional pelos monitores**

Exercícios em Aula

- A submissão dos exercícios será realizada do seguinte modo:
 - Abertura da submissão:
 - 1º bloco (aulas 1 a 4): últimos 50 minutos da aula
 - 2º e 3º blocos (aulas 5 a 12): últimos 20 minutos da aula
 - Limite de 3 (três) submissões sem penalização
 - Para cada submissão subsequente, a nota máxima do exercício será 2/3 da anterior:
 - Exemplo: 4ª. Submissão, nota máxima 6.7
 - 5ª. Submissão, nota máxima 4.4
 - ...

Exercícios Programa

- Realizado em duplas
- Haverá 3 exercícios
 - Desenvolvimento incremental
- Integração com a disciplina Introdução à Engenharia Elétrica

Avaliação

- $MF = (2*ME + 3*MEP + 5*MP) / 10$
 - $ME = (E1 + \dots + E11) / 11$
 - onde $E_i = k_i * AUT_i$,
 k_i = nota de acompanhamento, $k_i \in \{0, 0.8, 0.9, 1.0\}$
 AUT_i = nota da correção automática
 - $MEP = (EP1 + EP2 + 2*EP3) / 4$
 - onde $EP_i = 0.3 * q_i + 0.7 * AUT_i$,
 q_i = nota de especificação/qualidade/interface
 AUT_i = nota da correção automática
 - $MP = (P1 + P2 + 2*P3) / 4$
- Prova Substitutiva é fechada: 13/12/17

Código de ética da USP

- Disponível em

http://www.mp.usp.br/sites/default/files/arquivosanexos/codigo_de_etica_da_usp.pdf

- Art. 23 - É vedado aos membros do corpo docente e demais alunos da Universidade:
 - I – [...]
 - II - lançar mão de meios e artifícios que possam fraudar a avaliação do desempenho, seu ou de outrem, em atividades acadêmicas, culturais, artísticas, desportivas e sociais, no âmbito da Universidade, e acobertar a eventual utilização desses meios.

Código de ética da USP

- Alunos não devem fazer upload de programas por outros alunos
- Alunos não devem fazer upload fora dos seus horários de aula
- Alunos não devem vir à aula com programas prontos sem nem ter idéia do que eles fazem!
- Alunos devem fazer a prova individualmente
 - Sem uso de conhecimento de colegas
 - Sem uso de whatsapp!
- Será cobrado na disciplina!

Visão Geral de OO

Conceito de Objeto

- Dificuldade: identificação de objetos de um domínio



Mr. Potato Head, um brinquedo orientado a objetos.
(fonte: Budd, 2002)



Desenvolvimento de Software

- Desenvolver software não envolve só uma 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

- **Imperativo:** estado global e comandos de mudanças de estado
 - *Linguagens:* Pascal, C e Cobol
- **Funcional:** funções matemáticas (não afetam o estado)
 - *Linguagens:* Lisp, Haskell, ML e Scala
- **Lógico:** lógica formal (ex.: lógica de 1ª ordem)
 - *Linguagens:* Prolog e Datalog
- **Orientação a objetos:** abstração do mundo em objetos
 - *Linguagens:* C++, Java, C#, Objective C e Python

Paradigmas de Programação

- **Orientação a Eventos:** incorporam eventos e sua manipulação
 - Bastante usado para a criação de interfaces gráficas (inclusive em C++)
- **Declarativo:** especificam relações entre entidades que o programa “deve satisfazer”, sem dizer “como” deve fazer.
 - *Linguagens:* SQL e HTML
- Algumas linguagens são *multiparadigma*
 - *Linguagens:* C++, Python

Histórico da OO

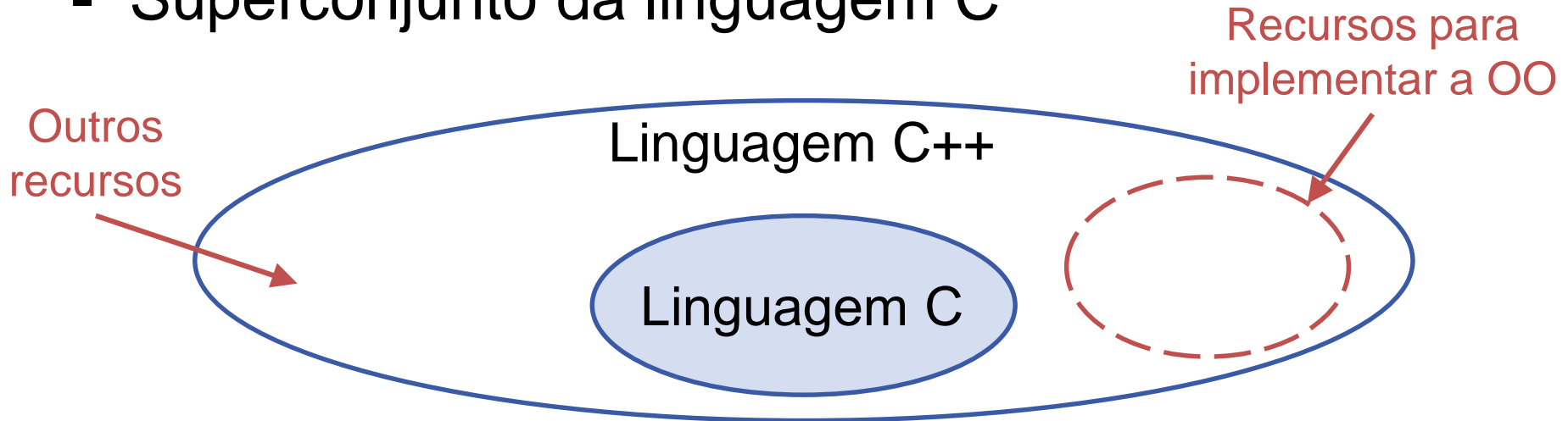
- *Centro de Computação Norueguês*
 - Simula: 1ª Linguagem OO (1967)
 - Ideia motivou outras linguagens
- Alan Kay (*Xerox PARC*)
 - Linguagem que fosse fácil de entender por usuários
 - Smalltalk (disponibilizada em 1980)
- Bjarne Stroustrup (*Bell Labs*)
 - Extensão de C para usar os conceitos de *Simula*
 - 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

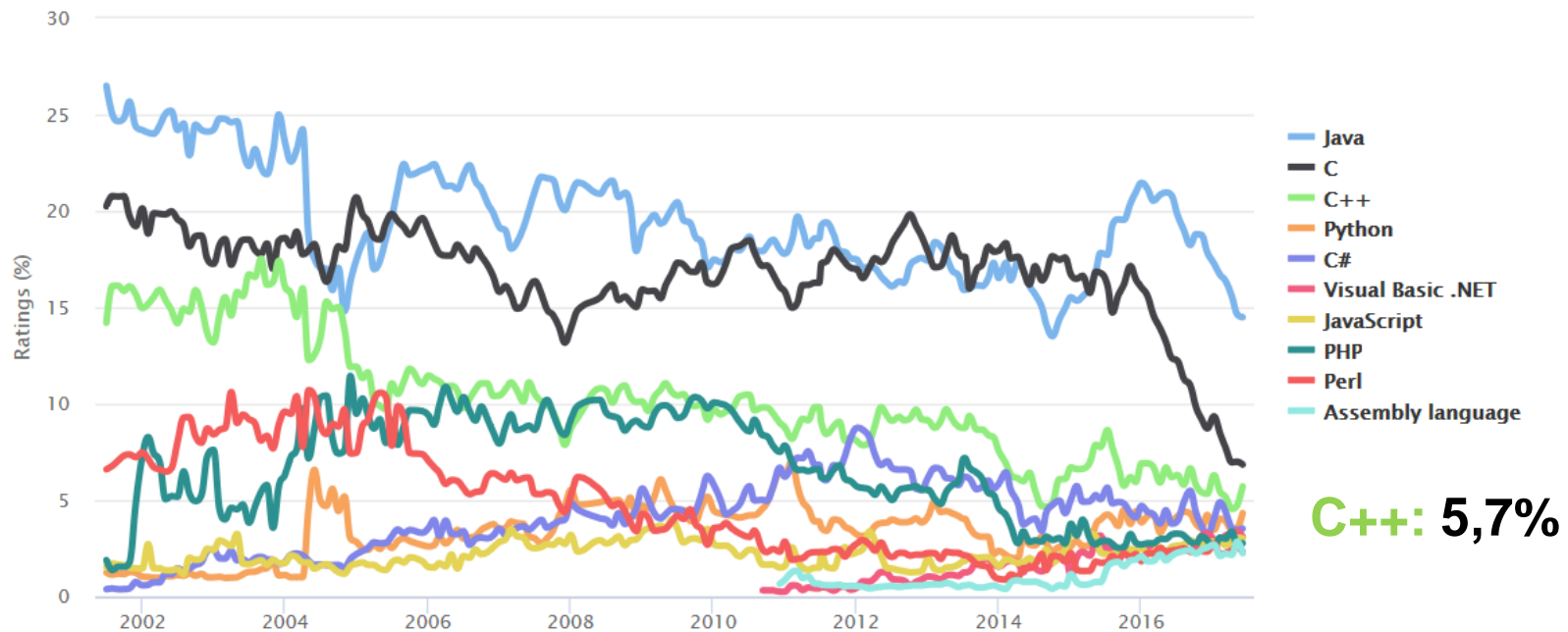
- Padrão ISO (a partir de 1998)

- Versão atual: C++11

Popularidade de C++

TIOBE Programming Community Index

Source: www.tiobe.com



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

Compiladores e Ambientes

- Alguns compiladores
 - GCC (Linux) Windows: MinGW <http://www.mingw.org> e Cygwin (<http://www.cygwin.com>)
 - 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)
 - <http://msdn.microsoft.com/en-us/vstudio>



Primeiro Exemplo

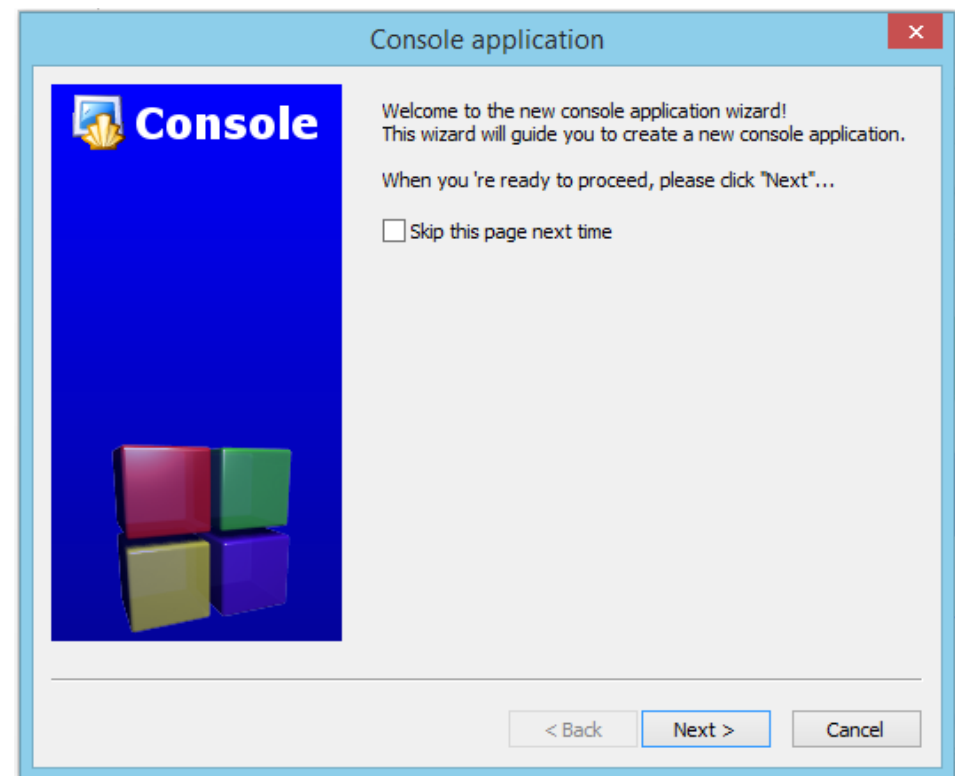
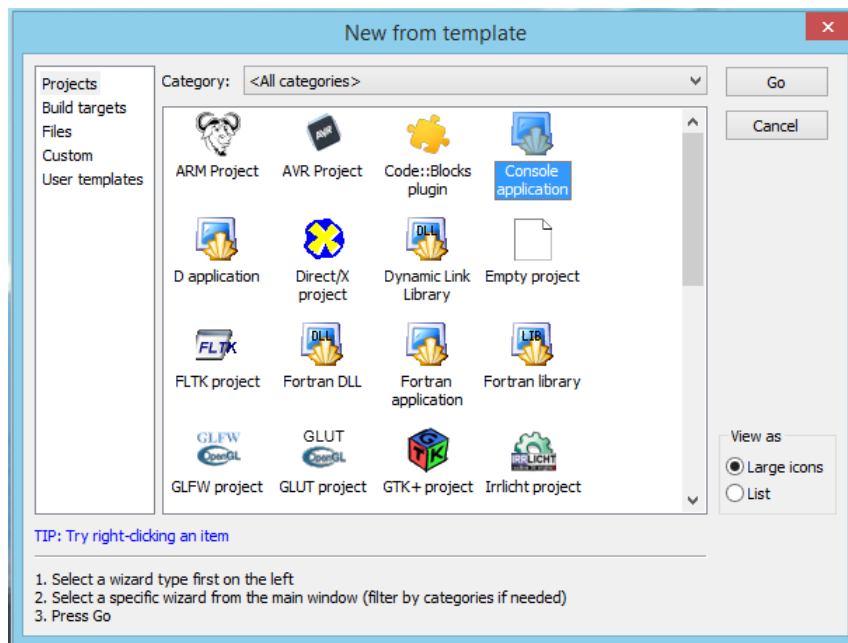
```
1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {
7      cout << "Hello world!" << endl;
8      return 0;
9  }
```

EX01

- Crie um projeto no CodeBlocks
 - File → New → Project (ou atalho “Create a new project”)

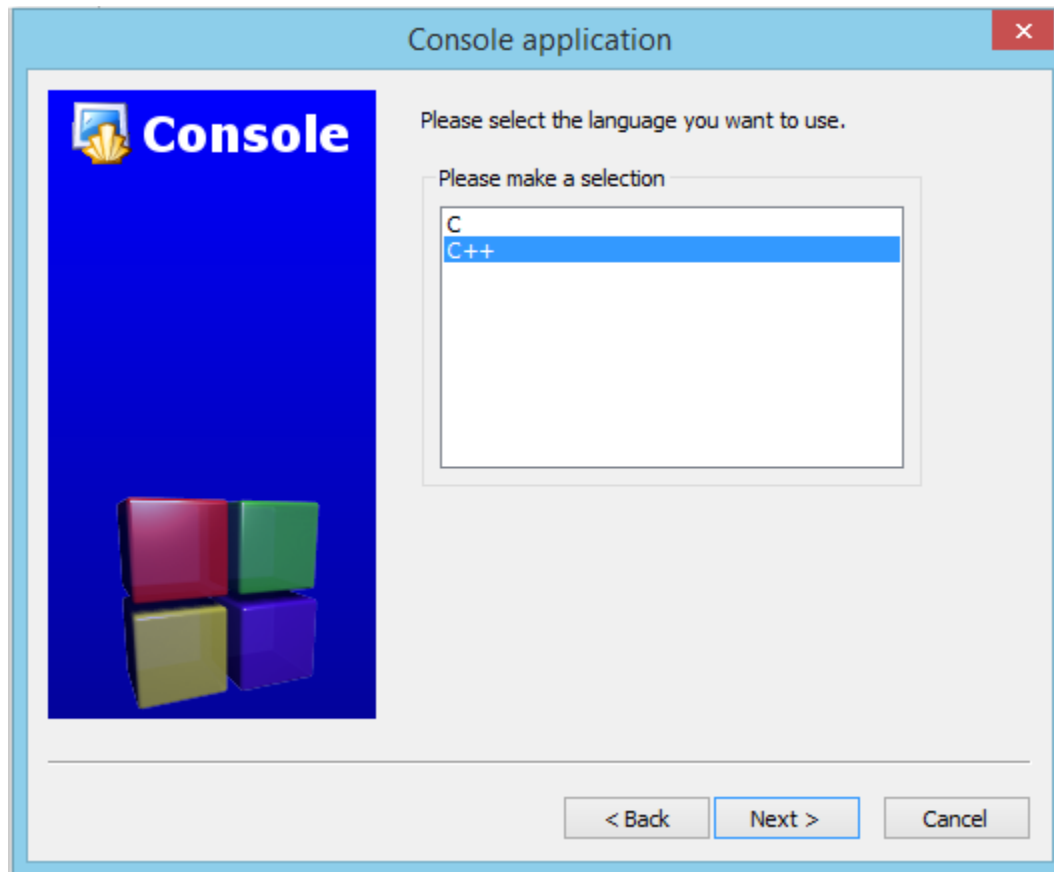
Primeiro Exemplo

- Escolha a categoria Console Application
 - Acione Go → Next



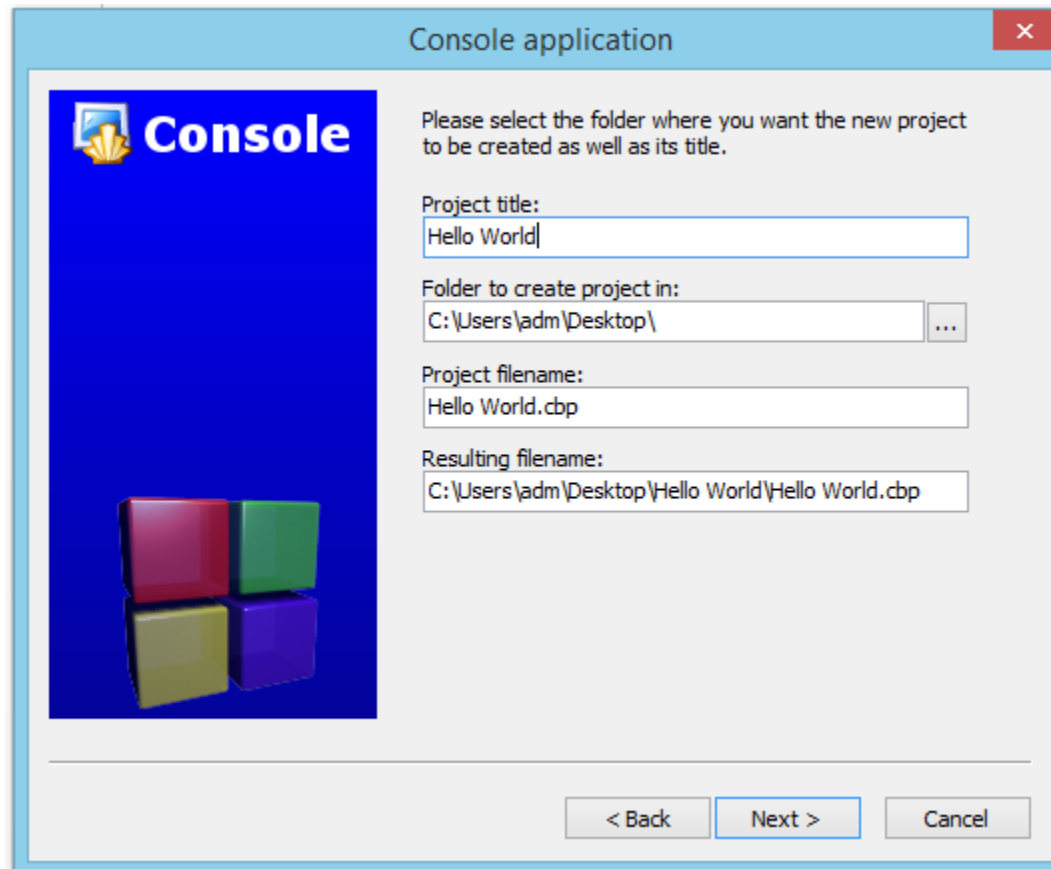
Primeiro Exemplo

- Escolha a linguagem C++ → Next



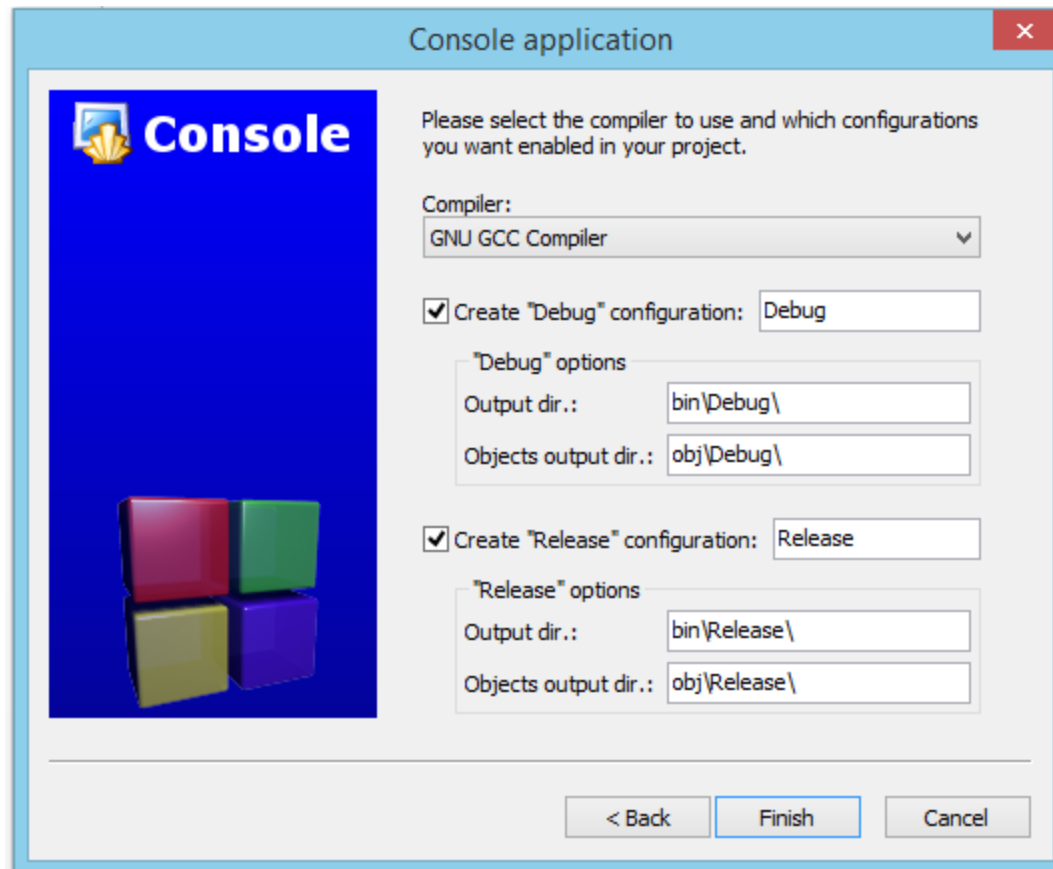
Primeiro Exemplo

- Escolha o nome e a pasta do projeto → Next



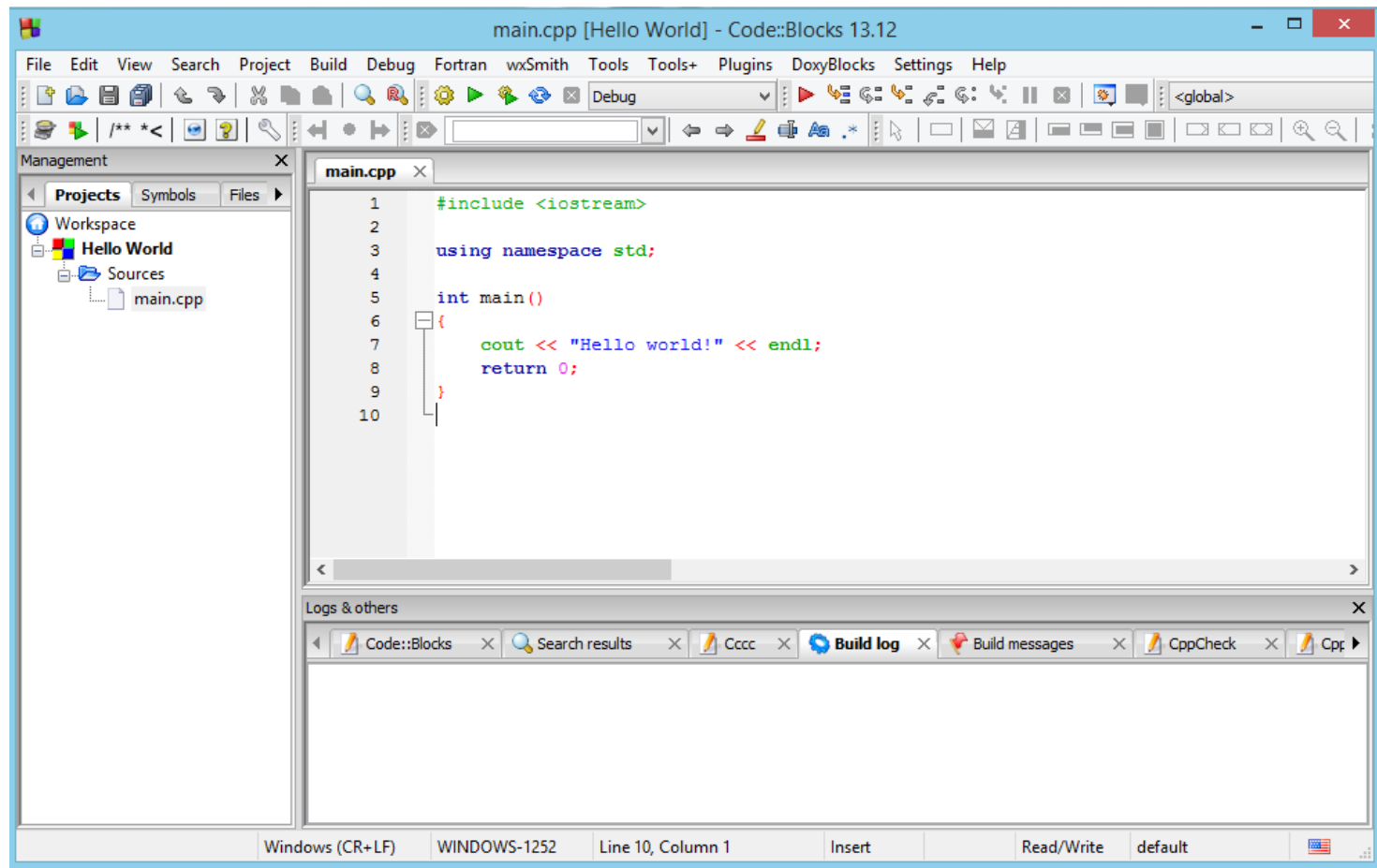
Primeiro Exemplo

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



Primeiro Exemplo

- Projeto Hello World e arquivo main.cpp

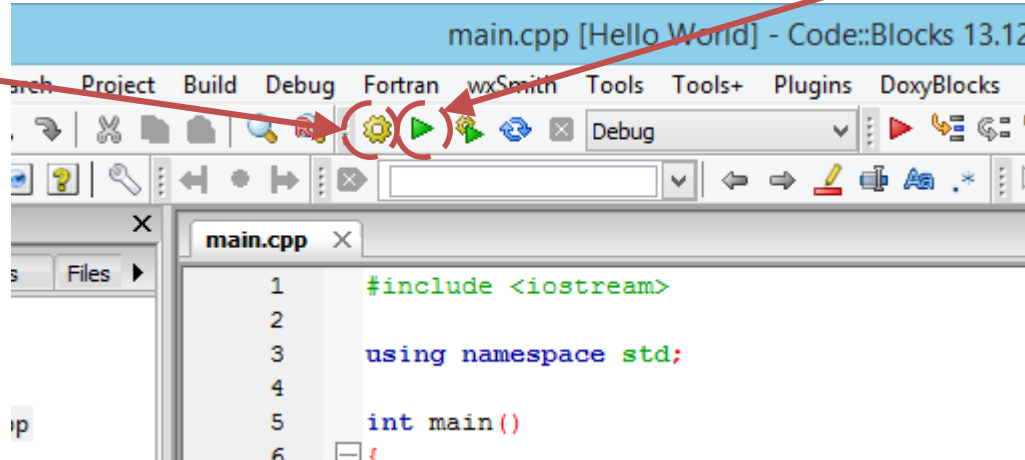


Primeiro Exemplo

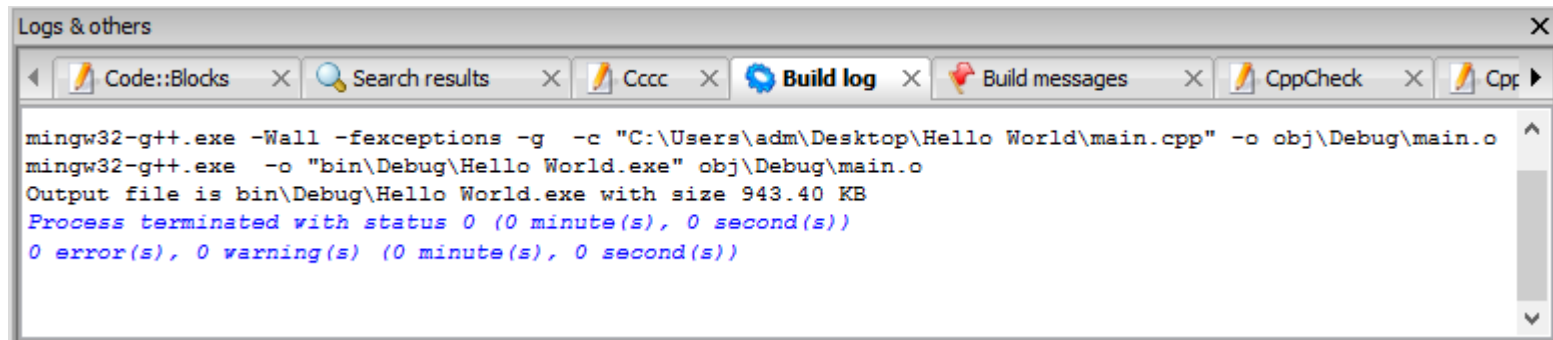
- Compile e execute o programa

Compilar
(CTRL-F9)

Executar
(CTRL-F10)

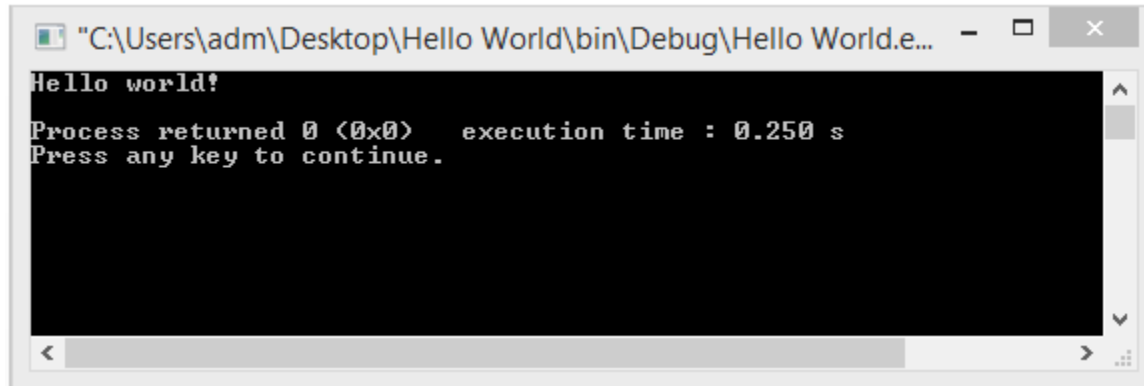


- Resultado da compilação



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 command prompt area is black with white text. It displays "Hello world!" on the first line. The second line shows "Process returned 0 (0x0) execution time : 0.250 s". The third line says "Press any key to continue.". There is a scrollbar on the right side of the window.

```
"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.
```

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: 01/09/2014  
*/
```

```
/* Comentário */ x++;
```


Vetor

- É um conjunto ordenado de variáveis de um mesmo tipo

- Exemplo*

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

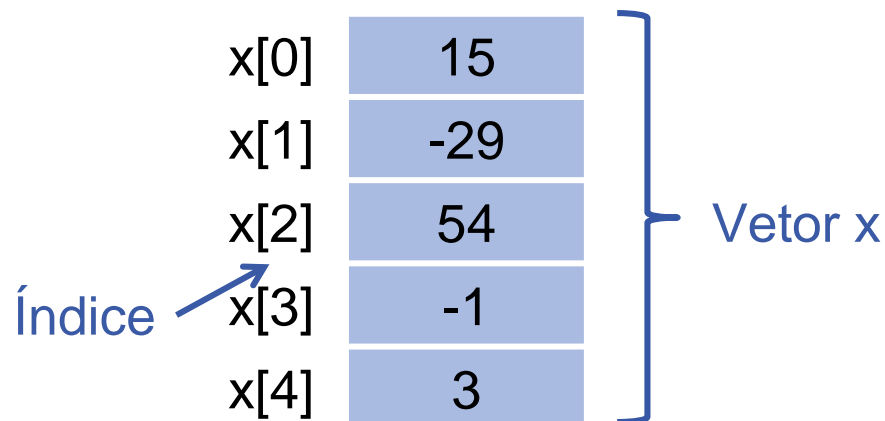
Declara um vetor y (os valores não estão inicializados)

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

Declara um vetor x, inicializando os valores

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

Declara uma matriz 5x5



Vetor

- 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

- *Observação:* o tamanho do vetor deve ser uma constante (não pode ser uma variável)
 - (Veremos futuramente como ser uma variável)

```
int tamanho = 5;  
int numeros[tamanho];
```



```
int numeros[5];
```



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*

```
int x = 0;
cin >> x;
```

digitado pelo usuário é colocado na variável x

cin e cout

■ Saída padrão: cout

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

```
int i = 5;  
cout << "Olá\n";  
cout << i;
```



Saída

```
Olá  
5
```

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

Diagram illustrating the output construction for the second example. Brackets and labels are placed below the code to show how the output is built:

- "x vale " is labeled **texto**.
- x is labeled **x**.
- " e y vale " is labeled **texto**.
- y is labeled **y**.
- endl is labeled **Pula linha**.

Saída

```
x vale 5 e y vale 6
```

string

■ Tipo string

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

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

← (Necessário para o *cout*)

} Necessário para usar a string

```
int main() {
    string nome = "Jose";
    ...
    nome = "Pedro";
    char inicial = nome[0];
    cout << nome << endl;
}
```

← Valor inicial

← Novo valor

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

Programa Básico

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

int multiplicar(int x, int y) {
    return x * y;
}

int main() {
    int x = 5, y = 3;
    cout << multiplicar(x, y) << endl;
    return 0;
}
```

Inclusões e outras diretivas

Funções

- **Main:** ponto de entrada do programa
 - Sempre coloque um `return, 0` indica sucesso
 - Um projeto só pode ter 1 main

Programa Básico

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

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

```
int multiplicar(int x, int y); ← Protótipo (declaração)
```

```
int main() {
    int x = 5, y = 3;
    cout << multiplicar(x, y) << endl; ← Uso
    return 0;
}
```

```
int multiplicar(int x, int y) { ← Definição da função
    return x * y;
}
```


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.
- Indentação
 - FEOFILOFF, P. **Algoritmos em linguagem C**. Editora Campus, 2009. Apêndice A.