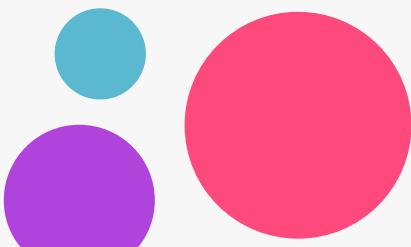


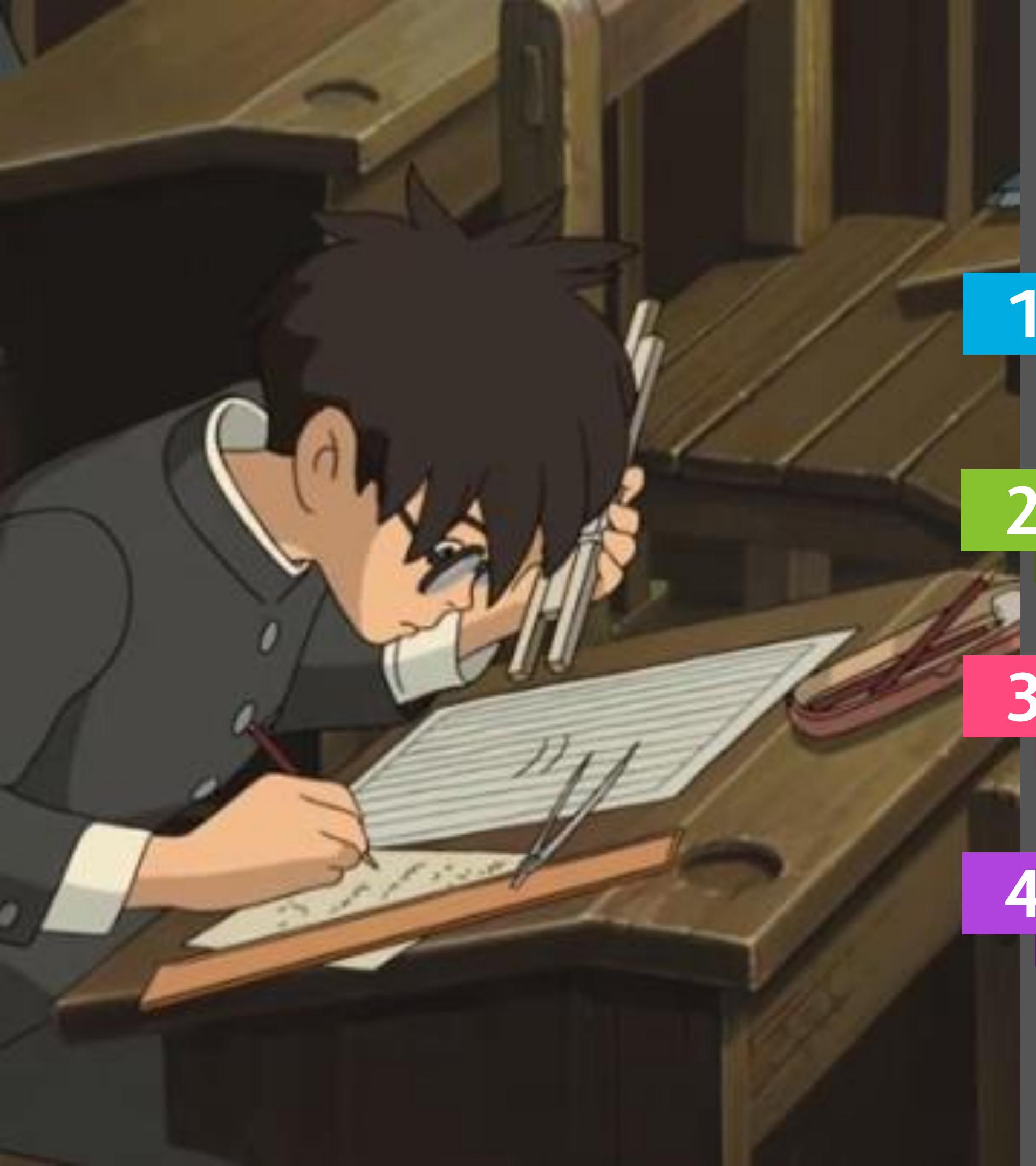
Fundamentos de C++

Aula 01 - Programação Orientada à Objetos

• • •

Professor Daniel Marques





Conteúdo da aula

1

Introdução

2

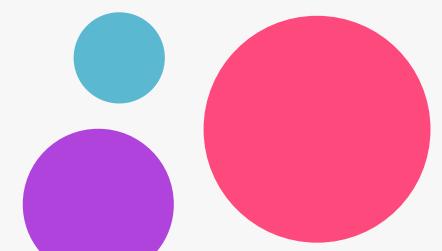
Conhecendo C++

3

Tipos de dados

4

Expressões lógicas



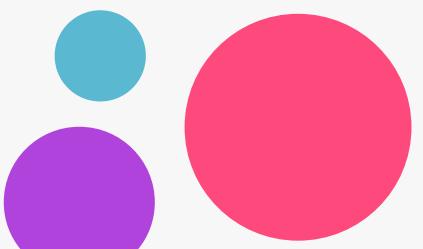
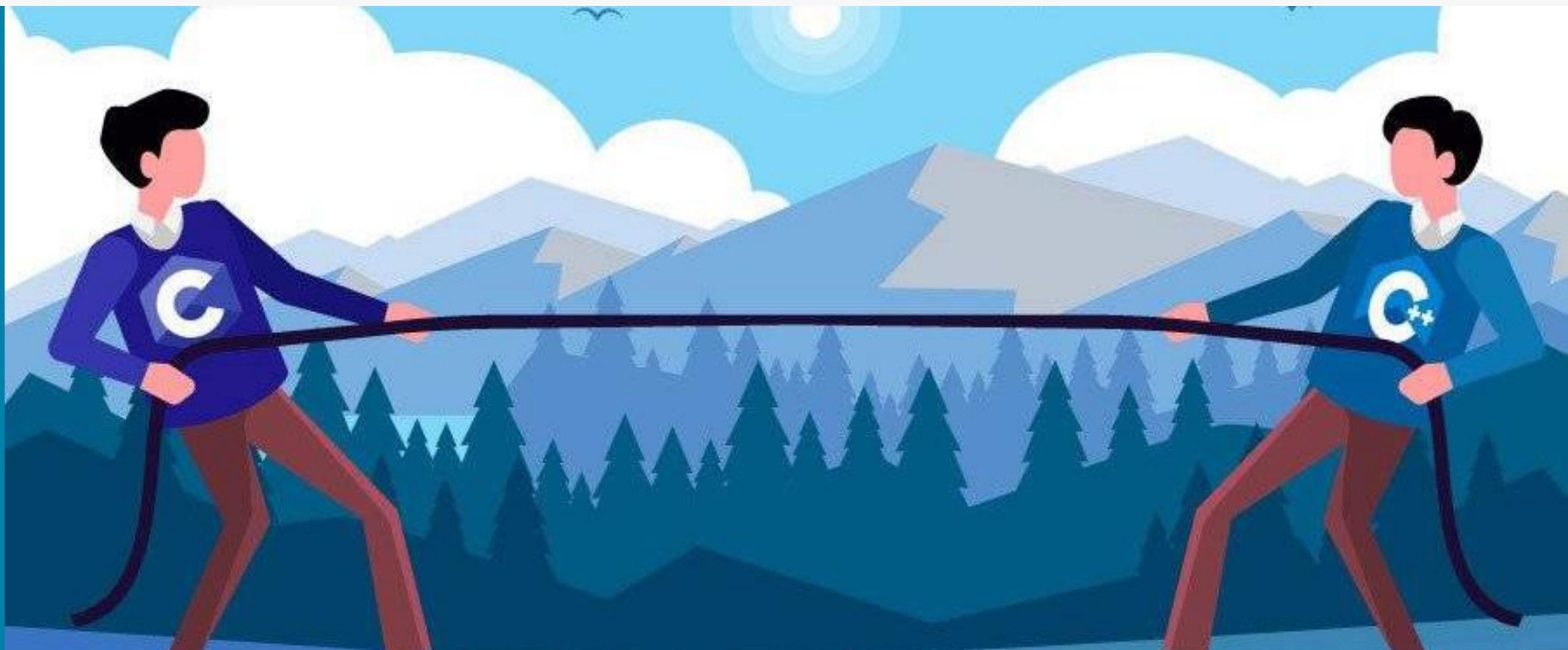
1

Introdução

C e C++

Vamos esclarecer algumas coisas:

- C e C++ não são diferentes;
- C++ não é mais complexo do que C;



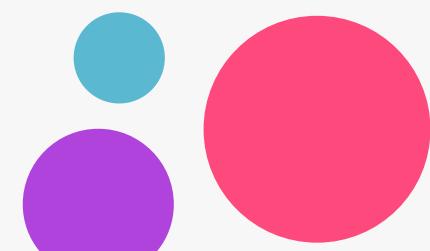
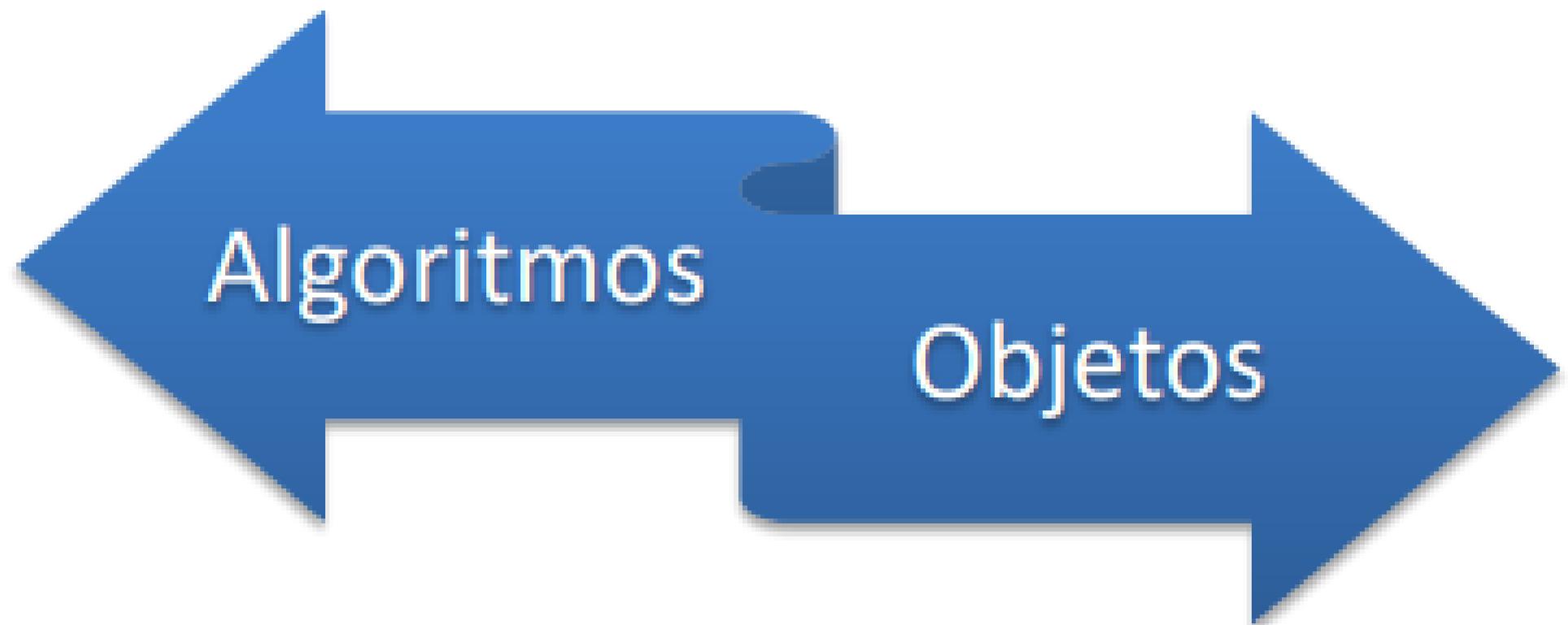
C++ e programação orientada a objetos

C com programação
orientada a objetos, resulta
em C++



Paradigma Procedimental X Orientado à Objetos

- O paradigma procedural organiza o programa em termos de algoritmos
- O paradigma orientado à objetos organiza o programa em termos de objetos



Algoritmos X Objetos

Podemos criar programas pensando em termos de objetos ao invés de algoritmos?



O mundo é composto por objetos

- Uma loja tem produtos, pedidos, estoque;
- Um restaurante tem mesas, garçons, comida;
- Uma rodoviária tem ônibus, passageiros, bagagens

**E se criarmos programas
basicamente criando objetos?**



- Objetos equivalentes ao mundo real;
- Fazer com que esses objetos se comuniquem

Princípios da orientação a objetos

Abstração: a representação computacional do objeto real deve concentrar nas características que são relevantes para o problema

Modularidade: O sistema deve ser composto de objetos altamente coesos e fracamente acoplados

Encapsulamento: o objeto deve esconder seus dados e os detalhes de sua implementação

Hierarquia: Os objetos devem ser organizados no sistema de forma hierárquica





Mas o que é um paradigma?

Um paradigma define a forma (e os recursos) para se resolver um problema

Prof. Victor André (IFPB)

- Não estruturado;
- Estruturado;
- Orientado a objetos;
- Funcional;
- Imperativo

Conceito

Exemplos

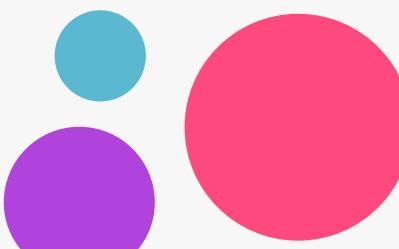
Quem criou C++?

Bjarne Stroustrup é um cientista da computação dinamarquês e professor catedrático da Universidade do Texas A&M. É conhecido como o pai da linguagem de programação C++.



Stroustrup, nas suas próprias palavras, “inventou a C++, escreveu as suas definições iniciais e produziu a sua primeira implementação [...]”.

Site: <https://www.stroustrup.com/>



Um pouco da história

Extensão da linguagem C é desenvolvida por Bjarne Stroustrup nos Bell Laboratories

1979

Stroustrup publica o livro referência The C++ Programming Language. (1a Edição)

1985

1983

Nome é modificado para C++;

1985

C++ é implementada como um produto comercial, mas ainda não estava padronizada

Um pouco da história

Compilador Borland's Turbo C++ é lançado.
Adição de várias bibliotecas que trouxeram
significativo impacto para a linguagem C++.

1990

Comitê da ISO lança uma
revisão do padrão (C++ 03)

2003

C++ é padronizado pela ISO
(C++ 98)

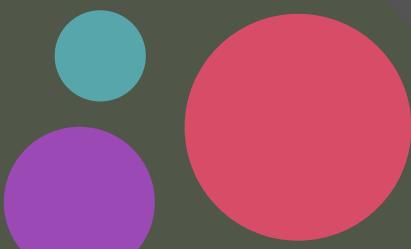
1998

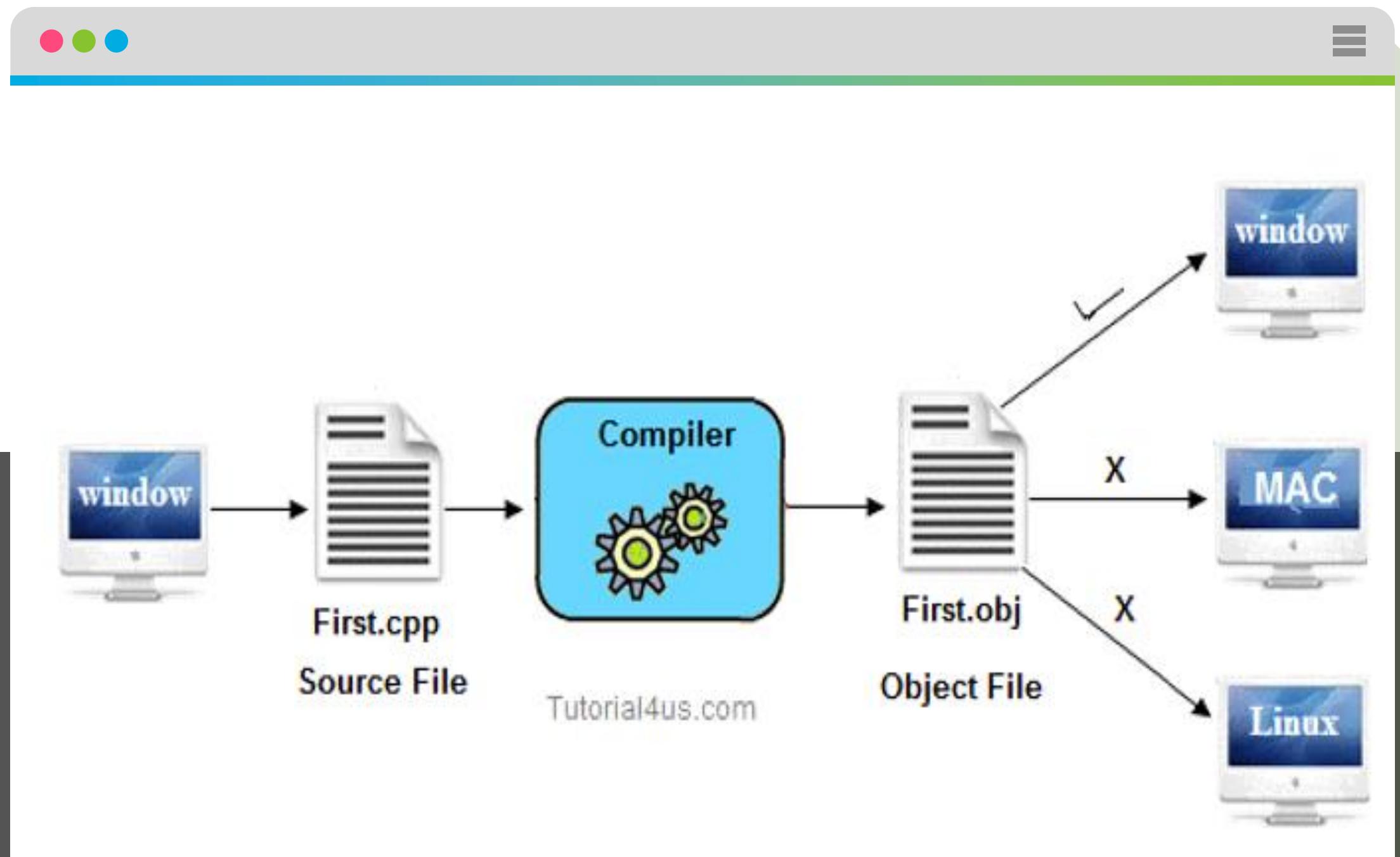
Padrão C++ 11 é lançado. Algumas das
novas funcionalidades: Threads,
biblioteca de tempo (time), suporte a
loops for-each

2011

Oct 2021	Oct 2020	Change	Programming Language	Ratings	Change
1	3	▲	 Python	11.27%	-0.00%
2	1	▼	 C	11.16%	-5.79%
3	2	▼	 Java	10.46%	-2.11%
4	4		 C++	7.50%	+0.57%
5	5		 C#	5.26%	+1.10%
6	6		 Visual Basic	5.24%	+1.27%
7	7		 JavaScript	2.19%	+0.05%
8	10	▲	 SQL	2.17%	+0.61%
9	8	▼	 PHP	2.10%	+0.01%

Criação de programas em C++





Criação de programas em C++

Inicia com a edição de um programa-fonte e termina com a geração de um programa executável

Ferramentas para programar em C++

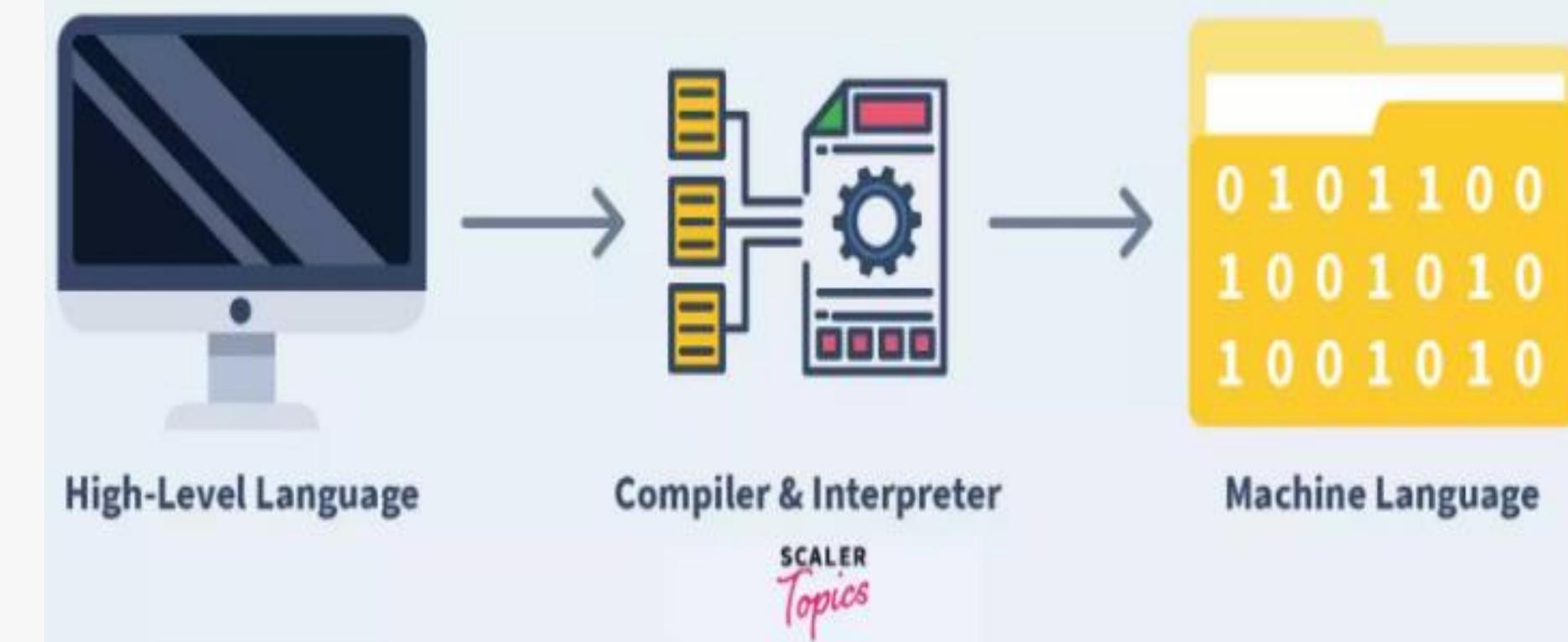
Tipicamente só é necessário: Editor de texto e um Compilador



Editores de texto

- Notepad++ (Windows)
- Sublime Text (Windows, Linux e Mac)
- Atom (Windows)
- Vim (Linux)
- Gedit (Linux)

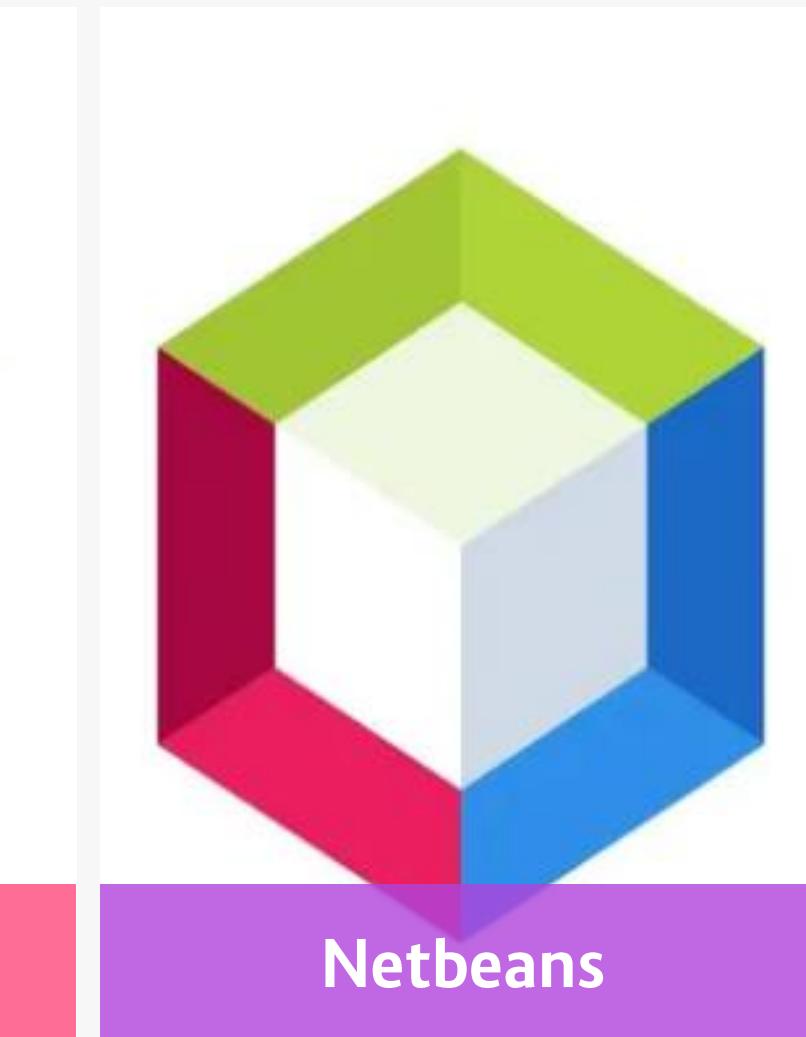
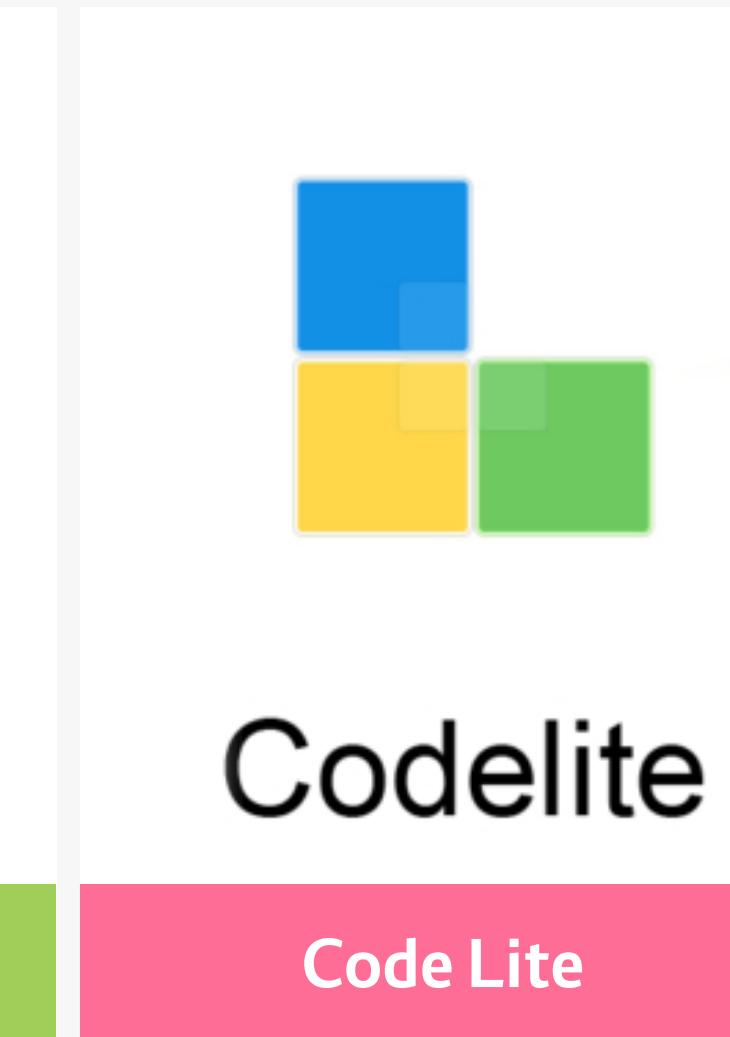
Compiladores C++



- G++ (Linux, OS X)
- MinGW (Win)
- Cygwin (Win)
- Apple C++ (OS X)

Ferramentas para programar em C++

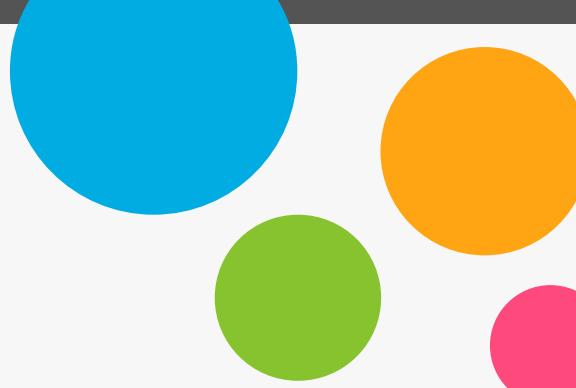
Ambientes de desenvolvimento





2

Conhecendo C++



Estrutura básica de um programa em C++



codigo_c.cpp



```
1 int main()
2 {
3     return 0;
4 }
```

-
- A função main inicia a execução do programa.
 - Todo programa deverá ter exatamente uma função main



Estrutura básica de um programa em C++

As linhas 3 e 4 começam com `//`, indicando que o restante de cada linha é um comentário

```
1 int main()
2 {
3     // Figura 2.1: fig02_01.cpp
4     // Programa de impressão de texto.
5
6     return 0;
7 }
```

C++ também podem utilizar o estilo C em que um comentário — possivelmente contendo muitas linhas — inicia com o par de caracteres `/*` e termina com `*/`



Entrada e saída de dados em C++

A entrada e saída de dados em C é feita através das funções `scanf()` e `printf()`.

Já em C++, vamos usar o novo sistema de entrada e saída de dados por fluxo

```
1 #include <iostream.h>
2
3 int main() {
4     cout << "Olá mundo! \n";
5     return 0;
6 }
7
```

Precisamos incluir (importar) o arquivo `iostream.h` em um programa C++

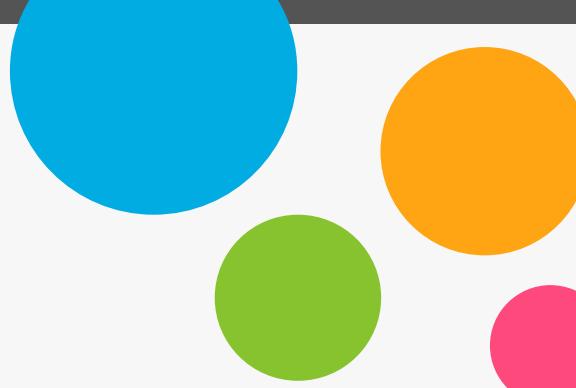


Entrada e saída de dados em C++

um programa C++ dispõe de três fluxos predefinidos que são abertos automaticamente pelo sistema

```
1 #include <iostream.h>
2
3 int main() {
4     char nome[80];
5     cout << "Qual o seu nome? ";
6     cin >> nome;
7     cout << "Olá " << nome << ", tudo bem?\n";
8     return 0;
9 }
```

-
- **cin**, que corresponde à entrada padrão;
 - **cout**, que corresponde à saída padrão;
 - **cerr**, que corresponde à saída padrão de erros;



Entrada e saída de dados em C++

- O operador `<<` permite inserir valores em um fluxo de saída
- o operador `>>` permite extrair valores de um fluxo de entrada

```
1 #include <iostream.h>
2
3 int main() {
4     char nome[80];
5     cout << "Qual o seu nome? ";
6     cin >> nome;
7     cout << "Olá " << nome << ", tudo bem?\n";
8     return 0;
9 }
```

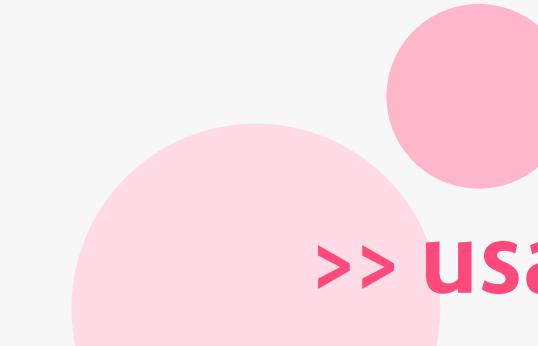


Entrada e saída de dados em C++

```
#include <iostream.h>

int main() {
    char nome[80];
    cout << "Qual o seu nome? ";
    cin >> nome;
    cout << "Olá " << nome << ", tudo bem?\n";
    return 0;
}
```

<< usado de forma encadeada



>> usado de forma encadeada

```
#include <iostream.h>

int main() {
    float comprimento, largura;
    cout << "Informe o comprimento e a largura do retângulo: ";
    cin >> comprimento >> largura;
    cout << "Área do retângulo: " << comprimento * largura << " m2\n";
    return 0;
}
```

3

Tipos de dados



Tipos primitivos de dados

Type	Meaning	Minimum Size
bool	boolean	NA
char	character	8 bits
wchar_t	wide character	16 bits
char16_t	Unicode character	16 bits
char32_t	Unicode character	32 bits
short	short integer	16 bits
int	integer	16 bits
long	long integer	32 bits
long long	long integer	64 bits
float	single-precision floating-point	6 significant digits
double	double-precision floating-point	10 significant digits
long double	extended-precision floating-point	10 significant digits



Tipos primitivos de dados

```
1 #include <iostream.h>
2
3 int main() {
4     std::cout << "Tamanho em bytes:" << std::endl;
5     std::cout << "bool " << sizeof(bool) << std::endl;
6     std::cout << "char " << sizeof(char) << std::endl;
7     std::cout << "short " << sizeof(short) << std::endl;
8     std::cout << "int " << sizeof(int) << std::endl;
9     std::cout << "long " << sizeof(Long) << std::endl;
10    std::cout << "long long " << sizeof(Long Long) << std::endl;
11    std::cout << "float " << sizeof(float) << std::endl;
12    std::cout << "double " << sizeof(double) << std::endl;
13    std::cout << "long double " << sizeof(Long double) << std::endl;
14    return 0;
15 }
```

Execute o código acima, para ver o tamanho em bytes de cada tipo de dado



Constantes literais

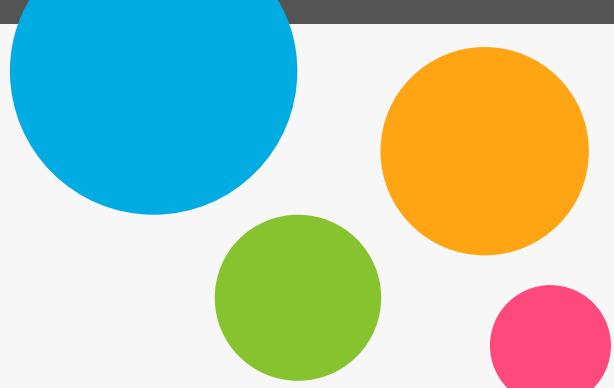
Inteiros

20 /* decimal */

024 /* octal */

0x14 /* hexadecimal */

```
1 #include <iostream.h>
2
3 int main() {
4
5     std::cout << 20 << std::endl;
6     std::cout << 024 << std::endl;
7     std::cout << 0x14 << std::endl;
8
9     return 0;
10 }
```



Constantes literais

Ponto flutuante

3.1415

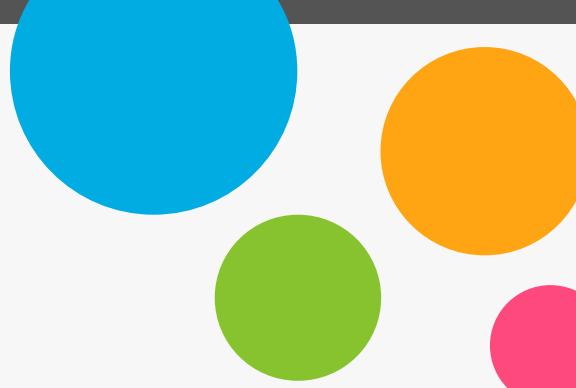
3.1415e0 ou 3.1415E0

0.

.0

0e0

```
1 #include <iostream.h>
2
3 int main() {
4
5     std::cout << 3.1415 << std::endl;
6     std::cout << 3.1415e0 << std::endl;
7     std::cout << 3.1415E0 << std::endl;
8     std::cout << 0. << std::endl;
9     std::cout << .0 << std::endl;
10    std::cout << 0e0 << std::endl;
11
12    return 0;
13 }
```



Constantes literais

Caractere e strings

‘a’

“

“uma string qualquer”

“outra string”

```
#include <iostream.h>

int main() {
    std::cout << 'a' << std::endl;
    std::cout << '' << std::endl;
    std::cout << "uma string qualquer" << std::endl;
    std::cout << "outra string" << std::endl;

    return 0;
}
```



Constantes literais

```
1 #include <iostream.h>
2
3 int main() {
4
5     std::cout << 100 << std::endl; //int
6     std::cout << 100u << std::endl; //unsigned
7     std::cout << 100L << std::endl; //long
8     std::cout << 100LL << std::endl; //long long
9
10    std::cout << 100.0 << std::endl; //double
11    std::cout << 100.0f << std::endl; //float
12    std::cout << 100.0L << std::endl; //long double
13
14    return 0;
15 }
```

Especificando o tipo da constante literal



Escape

Seqüência de escape	Descrição
\n	Nova linha. Posiciona o cursor de tela para o início da próxima linha.
\t	Tabulação horizontal. Move o cursor de tela para a próxima parada de tabulação.
\r	Retorno de carro. Posiciona o cursor da tela no início da linha atual; não avança para a próxima linha.
\a	Alerta. Aciona o aviso sonoro do sistema.
\\"	Barras invertidas. Utilizadas para imprimir um caractere de barra invertida.
\'	Aspas simples. Utilizadas para imprimir um único caractere de aspas simples.
\"	Aspas duplas. Utilizadas para imprimir um caractere de aspas duplas.



4

Variáveis



Tipos de dados

Tipos primitivos

- Incorporados na própria linguagem
- Representados por palavras chaves

Exemplos

- `int idade;`
- `double altura;`
- `string nome;`
- `char escolha;`

Tipos derivados

- Definidos pelo programador ou fornecidos pela biblioteca padrão do C++

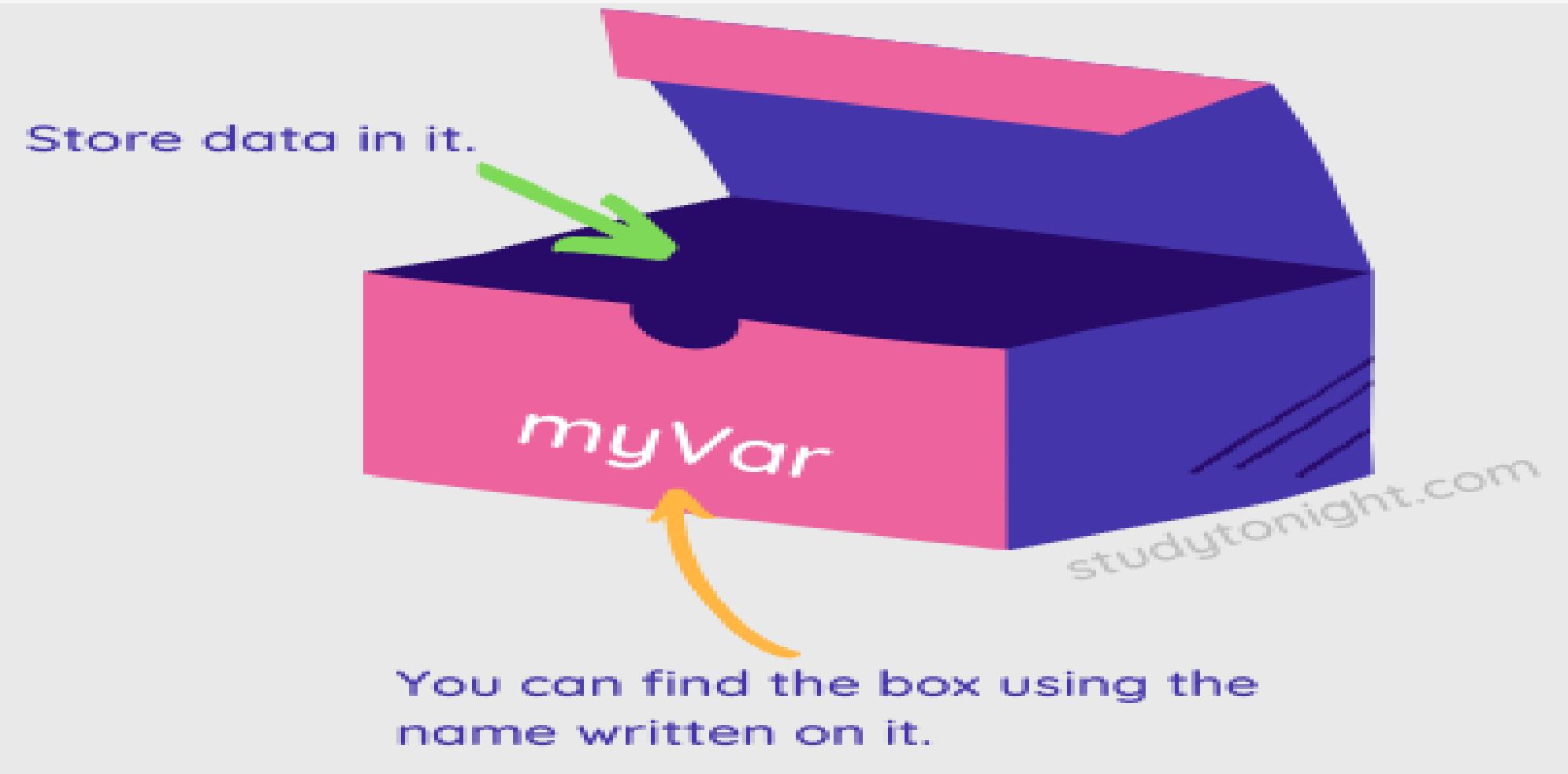
Exemplos

- Classes (estudaremos em breve)
- Classe pessoa;
- Classe endereço;

Variáveis

Identificadores

Em C++ podem conter letras, números e underscore (_), desde que não comecem por números



- um identificador deve indicar o seu significado
- nomes de variáveis normalmente em minúsculas
- classes começam com maiúsculas
- identificadores de muitas palavras devem ser visualmente distinguíveis



Declaração de variáveis

Segue a mesma sintaxe de C:

- Devem ser declaradas antes de ser utilizadas
- Mesma regras para definição de identificadores

```
1 #include <iostream.h>
2
3 int main() {
4
5     int idade;
6     double altura;
7     char turma;
8     bool especial;
9     string nome;
10
11    return 0;
12 }
```

Formato: <tipo> <nomeDaVariavel>;



Atribuição de variáveis

Note para o caso do tipo **bool**, que pode ser **true** ou **false**, ou também **0** ou **1**

```
#include <iostream.h>

int main() {

    int idade = 30;
    double altura1 = 1.75;
    float altura2 = 1.73;
    char turma = 'A';
    bool especial = true;
    bool especial = 1;

    return 0;
}
```

Mas, C++ reconhece o tipo **string**?



Atribuição de variáveis

Tipo string

Para trabalharmos com o tipo `string`, é necessário importar a biblioteca através do

`#include <string>`

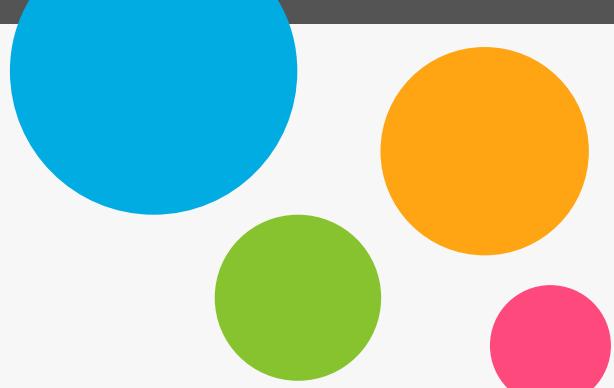
```
1 #include <iostream.h>
2 #include <string>
3
4 int main() {
5
6     string nome = "Daniel";
7     int idade = 30;
8     double altura1 = 1.75;
9     float altura2 = 1.73;
10    char turma = 'A';
11    bool especial = true;
12    bool especial = 1;
13
14    return 0;
15 }
```



Declaração de variáveis

Também podemos declarar várias variáveis de um único tipo na mesma linha

```
1 #include <iostream.h>
2 #include <string>
3
4 int main() {
5
6     string nome, cidade, estado;
7     int idade, id;
8     double altura, peso;
9
10    return 0;
11 }
12
```



Atribuição de variáveis

Atribuição de várias variáveis em uma única linha.

```
#include <iostream.h>
#include <string>

int main() {

    string nome = "João", cidade = "Campina Grande", estado = "Paraíba";
    int idade = 30, id = 0001;
    double altura = 1.80, peso = 60;

    return 0;
}
```

Note que o tipo da variável só aparece uma única vez na linha



Strings

Incluir a biblioteca

#include <string>

Usar também o namespace
std

```
1 #include <iostream>
2 #include <string>
3
4 using namespace std;
5
6 int main() {
7
8     string nome = "Daniel";
9     string sobrenome = "Marques";
10
11    std::cout << "Nome: " << nome << std::endl;
12    std::cout << "Abreviação: " << nome[0] << "." << sobrenome[0] << "." << std::endl;
13
14    return 0;
15 }
```

Operadores:

- [i] : acessa o i-ésimo caracter da string
- + : serve para concatenar a outra string

Strings: principais funções

`size()`

Retorna o tamanho da string

`empty()`

Verifica se a string está vazia

`at(int i)`

Retorna o caractere da posição i

`erase()`

Apaga o conteúdo da string

`replace(posição inicial, nº
caracteres, string)`

Substitui parte da string

Strings: principais funções

find()

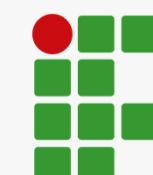
Procura por conteúdo na string

substr()

Gera substrings

compare()

Compara substrings





String e suas funções

```
1 #include <iostream>
2 #include <string>
3
4 using namespace std;
5
6 int main() {
7
8     string rua = "Floriano Peixoto, 1200, Dinamérica";
9     int tamanhoDaString = rua.size();
10    bool stringVazia = rua.empty();
11    string ruaModificada = rua.replace(18, 4, "1500");
12
13    std::cout << "Rua: " << rua << std::endl;
14    std::cout << "Quantidade de caracteres: " << tamanhoDaString << std::endl;
15    std::cout << "String está vazia? " << stringVazia << std::endl;
16    std::cout << "Rua modificada: " << ruaModificada << std::endl;
17
18    return 0;
19 }
```

5

Operadores aritméticos



Operadores aritméticos

Operação C++	Operador aritmético C++	Expressão algébrica	Expressão C++
Adição	+	$f + 7$	<code>f + 7</code>
Subtração	-	$p - c$	<code>p - c</code>
Multiplicação	*	bm ou $b \cdot m$	<code>b * m</code>
Divisão	/	x / y ou $\frac{x}{y}$ ou $x \div y$	<code>x / y</code>
Módulo	%	$r \bmod s$	<code>r % s</code>

Exemplo

```
1 #include <iostream>
2 #include <string>
3 #include <math.h>
4 using namespace std;
5
6 int main() {
7
8     int x, y;
9     x = 9;
10    y = 3;
11
12    double soma = x + y;
13    double subtracao = x - y;
14    double multiplicacao = x * y;
15    double divisao = x / y;
16
17    std::cout << "O valor de x: " << x << std::endl;
18    std::cout << "O valor de y: " << y << std::endl;
19
20    std::cout << "A soma entre x e y é: " << soma << std::endl;
21    std::cout << "A subtração entre x e y é: " << subtracao << std::endl;
22    std::cout << "A multiplicação entre x e y é: " << multiplicacao << std::endl;
23    std::cout << "A divisão entre x e y é: " << divisao << std::endl;
24
25    return 0;
26 }
27
28 }
```



Ordem de precedência

Operadores aritméticos

Operador(es)	Operação(ões)	Ordem de avaliação (precedência)
()	Parênteses	Avaliados primeiro. Se os parênteses estão aninhados, a expressão no par mais interno é avaliada primeiro. Se há vários pares de parênteses ‘no mesmo nível’ (isto é, não aninhados), eles são avaliados da esquerda para a direita.
*	Multiplicação	Avaliado em segundo lugar. Se houver vários, eles são avaliados da esquerda para a direita.
/	Divisão	
%	Módulo	
+	Adição	Avaliado por último. Se houver vários, eles são avaliados da esquerda para a direita.
-	Subtração	



Operadores aritméticos

Outras formas de realizar
alguns cálculos específicos

Operador	Exemplo	Significado
<code>+=</code>	<code>x += 2</code>	<code>x = x + 2</code>
<code>-=</code>	<code>x -= 1</code>	<code>x = x - 1</code>
<code>*=</code>	<code>x *= 1</code>	<code>x = x * 1</code>
<code>/=</code>	<code>x /= 3</code>	<code>x = x / 3</code>
<code>%=</code>	<code>x %= 5</code>	<code>x = x % 5</code>



Exponenciação e radiciação

Lembrar de importar a
biblioteca Math

#include <math.h>

```
1 #include <iostream>
2 #include <string>
3 #include <math.h>
4 using namespace std;
5
6 int main() {
7
8     int x, y;
9     x = 9;
10    y = 3;
11
12    double raizQuadrada = sqrt(x);
13    double exponenciacao = pow(x, y);
14
15    std::cout << "O valor de x: " << x << std::endl;
16    std::cout << "O valor de y: " << y << std::endl;
17
18    std::cout << "A raiz quadrada de x é: " << raizQuadrada << std::endl;
19    std::cout << "A exponenciação de x elevado a y é: " << exponenciacao << std::endl;
20
21    return 0;
22 }
```

- Exponenciação: pow (base, potência)
- Raiz quadrada: sqrt(número)

Alguma dúvida?

Não guardem dúvidas, perguntam

• • •

Referências

- 1 OLIVEIRA, Victor A. P. **Fundamentos de C++**. Instituto Federal da Paraíba. 2022.
- 2 DA COSTA, Anderson Fabiano F. **Fundamentos de C++**. Instituto Federal da Paraíba. 2022.
- 3 Materiais de aula dos professores Tiago Maritan e Guillermo Camara-Chavez
- 4 HORSTMANN, C. **Conceitos de Computação com o Essencial de C++**, 3^a edição, Bookman, 2005.
- 5 CPLUSPLUS. Disponível em: <<http://www.cplusplus.com/>>. Acesso em: 04 Abr 2022.