

IMD0030 LINGUAGEM DE PROGRAMAÇÃO I

Aula 01 – Apresentação da disciplina e Introdução à Linguagem de Programação C++

Objetivo da disciplina



Objetivo da disciplina

Capacitar o estudante a utilizar a linguagem de programação C++ para a implementação de programas visando a solução de problemas, aplicando boas práticas de programação

Competências e habilidades

- **Idealizar de forma algorítmica soluções para problemas**
- **Conhecer e fazer uso de importantes ferramentas de suporte ao programador**
- **Identificar e corrigir problemas de codificação e execução de programas**
- **Dominar o uso dos recursos básicos da linguagem de programação C++ e sua biblioteca padrão**
- **Implementar soluções para problemas utilizando a linguagem de programação C++**

Conteúdos

UNDER
REVIEW

1	31/07/2018 Apresentação do Plano de Aulas e Introdução à linguagem C++
2	02/08/2018 Introdução à Linguagem de Programação C++. Modularização e Compilação.
3	07/08/2018 Depuração (com o GDB) e Profiling (com o Gprof).
4	09/08/2018 Controle de Versões (Git) e Documentação (Doxygen).
5	14/08/2018 Laboratório 1: Uso das ferramentas de desenvolvimento de programas em C++.
6	16/08/2018 Introdução à orientação a objetos: Classes e objetos 21/08/2018 Não haverá aula 23/08/2018 Não haverá aula
7	28/08/2018 Construtores e destrutores
8	30/08/2018 Sobrecarga de funções e passagem de parâmetro
9	04/09/2018 Sobrecarga de operadores
10	06/09/2018 Laboratório 2: Classes e Objetos
11	11/09/2018 Avaliação I.

Conteúdos

UNDER
REVIEW

12	13/09/2018	Gerenciamento de memória.
13	18/09/2018	Ponteiros inteligentes. Revisão da Avaliação I.
14	20/09/2018	Programação genérica: templates de funções e templates de classes.
15	25/09/2018	Herança.
16	27/09/2018	Classes abstratas.
17	02/10/2018	Manipulação de arquivos.
18	04/10/2018	Laboratório 3: Herança, Classes Abstratas e Manipulação de arquivos
19	09/10/2018	Implementação de Tipos Abstratos de Dados (TADs) genéricos.
20	11/10/2018	Laboratório 4: Programação genérica e utilização de TADs genéricos.
21	16/10/2018	Standard Template Library (STL): Containers e iterators.
22	18/10/2018	Standard Template Library (STL): Biblioteca algorithms e type casting.
23	23/10/2018	Laboratório 5: Standard Template Library (STL)
24	25/10/2018	Avaliação II.

Conteúdos

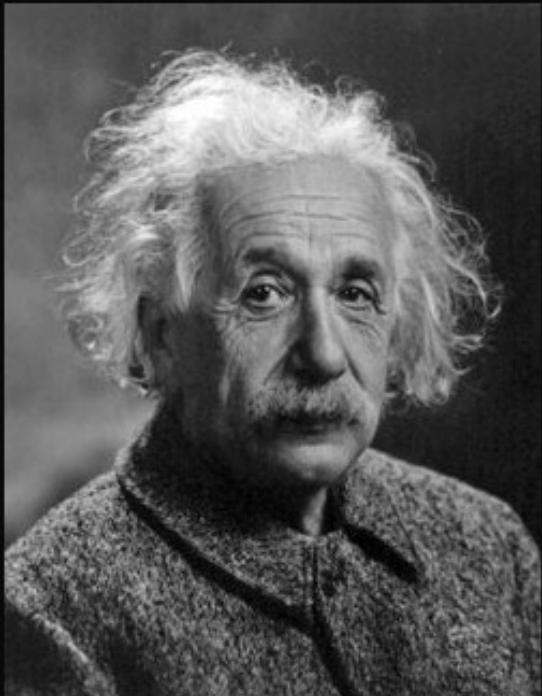
UNDER
REVIEW

- | | |
|----|---|
| 25 | 30/10/2018 Namespaces e bibliotecas |
| 26 | 01/11/2018 Laboratório 6: Namespaces e bibliotecas |
| 27 | 06/11/2018 Uso de bibliotecas externas |
| 28 | 08/11/2018 Laboratório 7: Uso de bibliotecas externas |
| 29 | 13/11/2018 Tratamento de exceções |
| 30 | 15/11/2018 Projeto III |
| 31 | 20/11/2018 Acompanhamento do Projeto III. |
| 32 | 22/11/2018 Avaliação III. |
| 33 | 27/11/2018 Acompanhamento de Projeto |
| 34 | 29/11/2018 Acompanhamento de Projeto |
| 35 | 04/12/2018 Acompanhamento de Projeto |
| 36 | 06/12/2018 Acompanhamento de Projeto |

Metodologia

- **Aulas teóricas** expositivas
- **Aulas práticas** voltados para a resolução de exercícios de programação e aplicação dos conceitos vistos
- **Laboratórios e Projeto Final** com o objetivo de solucionar de problemas por meio de programas implementados na linguagem C++

As regras do jogo



You have to learn the rules of the game. And then you have to play better than anyone else.

(Albert Einstein)

As regras do jogo

- **Não será aceito** nenhum código fonte desenvolvido utilizando recursos de C nem código fonte resultante de mescla entre C e C++
- **Não será adotada qualquer IDE**, privilegiando-se o uso de editores de texto simples e ferramentas em linha de comando
- **Será fortemente cobrada** a implementação de programas sem mensagens de aviso (*warnings*)

As regras do jogo

- Haverá **redução significativa de pontos** na avaliação de programas que não compilem ou que apresentem falha de segmentação (*segmentation fault*) na execução
- Haverá **redução significativa de pontos** na avaliação de programas que não produzam a saída esperada ou não estejam em conformidade com a especificação fornecida

As regras do jogo

Será ampla e fortemente estimulada a aplicação de boas práticas de programação

- Codificação de programas de maneira legível
(com indentação de código, nomes consistentes, etc.)
- Documentação adequada na forma de comentários (e mais tarde com *Doxygen*)
- Organização de programas complexos na forma de funções modulares e arquivos
- Teste sistemático de programas na forma de casos de teste

Importante lembrar...



- O conteúdo da disciplina é **incremental**
 - Os conceitos avançados somente podem ser compreendidos quando os básicos forem bem assimilados
- O conteúdo da disciplina é **abrangente** e requer um **esforço importante e permanente**
- Os exercícios propostos devem ser resolvidos para **melhor fixação** dos conceitos apresentados
 - **Inclusive fora do horário de aula!**

Avaliação

Instrumentos de avaliação – **Todos valem nota**

- Exercícios de programação (laboratórios)
 - Aplicação dos conceitos vistos nas aulas expositivas
 - Estímulo ao desenvolvimento das habilidades de programação em C++
- Três avaliações individuais e presenciais, uma em cada unidade
- Projeto final de programação
 - Solução de problemas por meio de programas implementados na linguagem C++
 - Realizado individualmente
 - **Tema LIVRE a ser definido pelo aluno – não deixem para o final**

Avaliação

Instrumentos de avaliação

Os laboratórios, avaliações ou projeto final poderão envolver conceitos vistos na disciplina

IMD0029 – Estruturas de Dados Básicas I ou equivalente



Avaliação

Critérios de avaliação dos programas desenvolvidos

- Utilização correta dos conteúdos vistos em aula
- Corretude da execução do programa implementado, com saída em conformidade com a especificação e as entradas de dados fornecidas
- Aplicação correta de boas práticas de programação (legibilidade, organização e documentação de código)
- Qualidade do relatório escrito (quando solicitado)

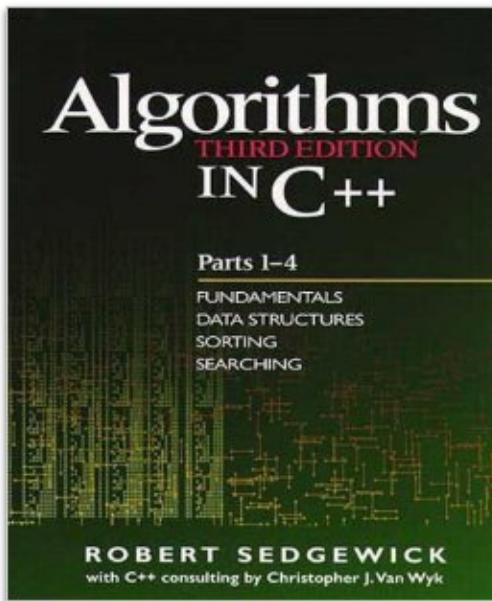
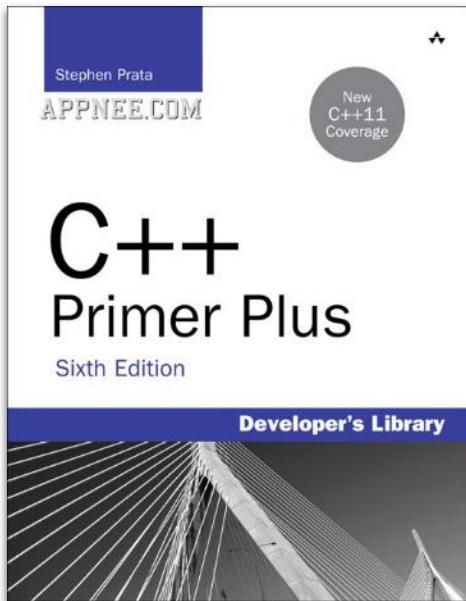
Avaliação

Rendimento acadêmico

- Ausência a alguma das avaliações ou não entrega de algum dos exercícios ou projeto de programação: **nota zero**
- Avaliação de reposição
 - Substituição do menor rendimento acadêmico nas unidades (Art. 107 e 110 do Regulamento dos Cursos de Graduação)
 - Avaliação individual e presencial realizada no fim do período letivo, cobrindo **todo o conteúdo ministrado**

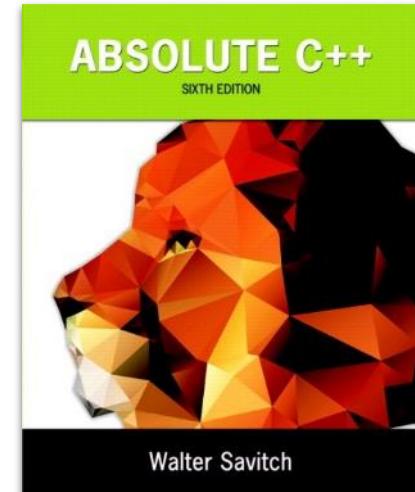
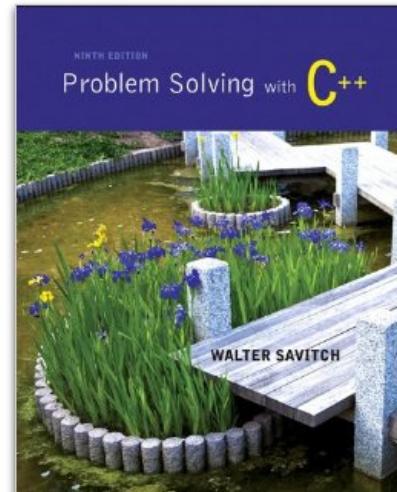
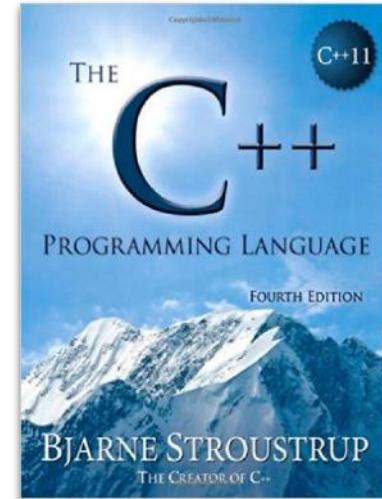
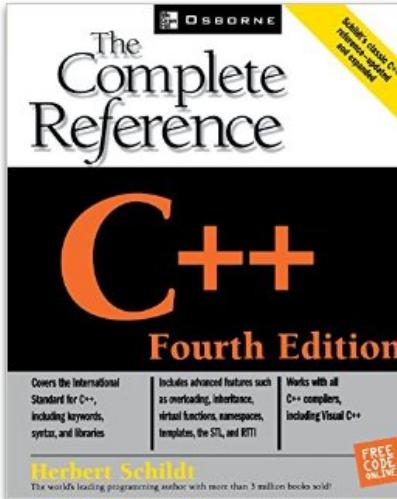
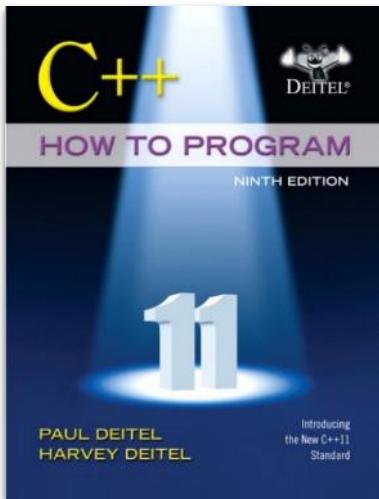
Bibliografia sugerida

Disponível na BCZM



Bibliografia sugerida

Disponível nas Livrarias ([há boatos que também na Internet!!](#))



Bibliografia sugerida

Links úteis

- cplusplus.com – The C++ Resources Network: <http://wwwcplusplus.com/>
- cppreference.com: <http://en.cppreference.com/w/>
- Stack Overflow: <http://stackoverflow.com/>



Observações gerais

Atendimento extraclasse

35N56

Preferencialmente, agendado previamente via e-mail

silviocs@imd.ufrn.br

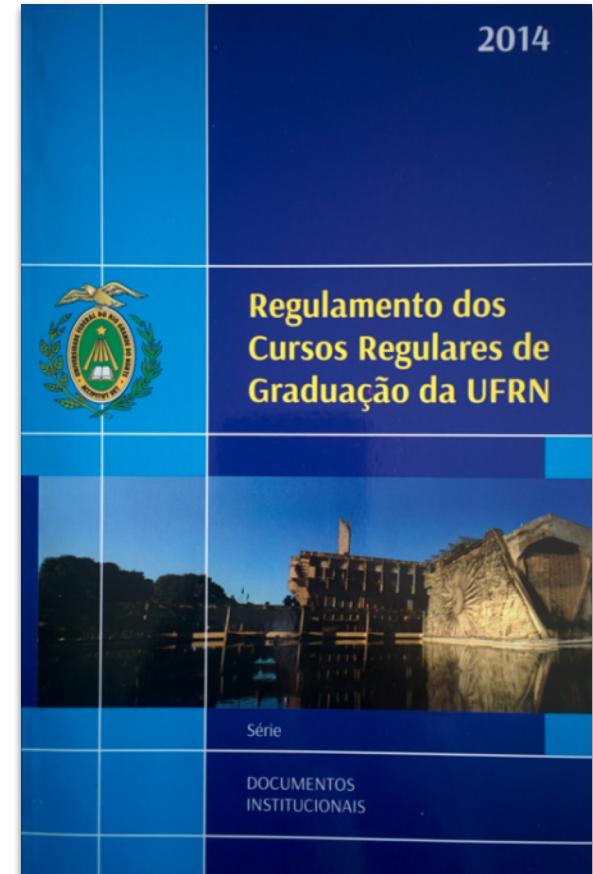
Sala A206

Observações gerais

Faltas às aulas presenciais

Não existe abono de faltas

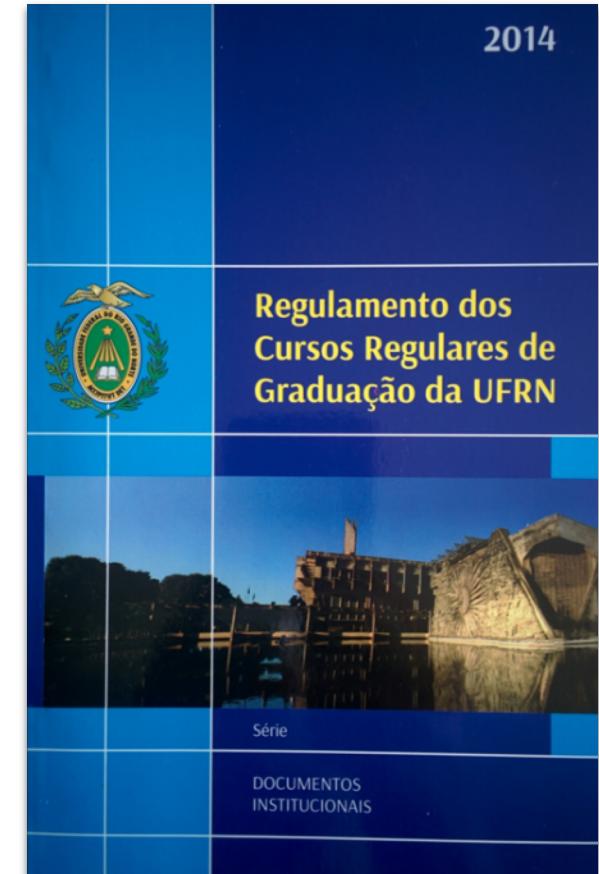
Art. 112 do Regulamento dos Cursos de Graduação



Observações gerais

Controle de presença

- Aprovação condicionada à presença mínima de 75% das aulas presenciais ministradas
Art. 94 e 113 do Regulamento dos Cursos de Graduação
- Frequência rigorosamente registrada via SIGAA e/ou lista de presença

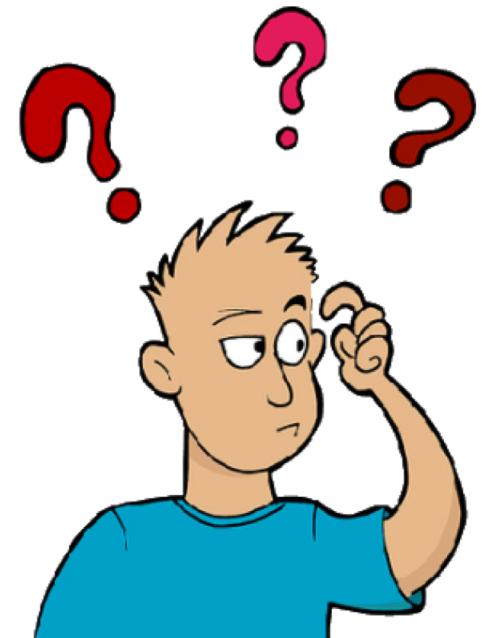


Observações gerais

Sobre plágio

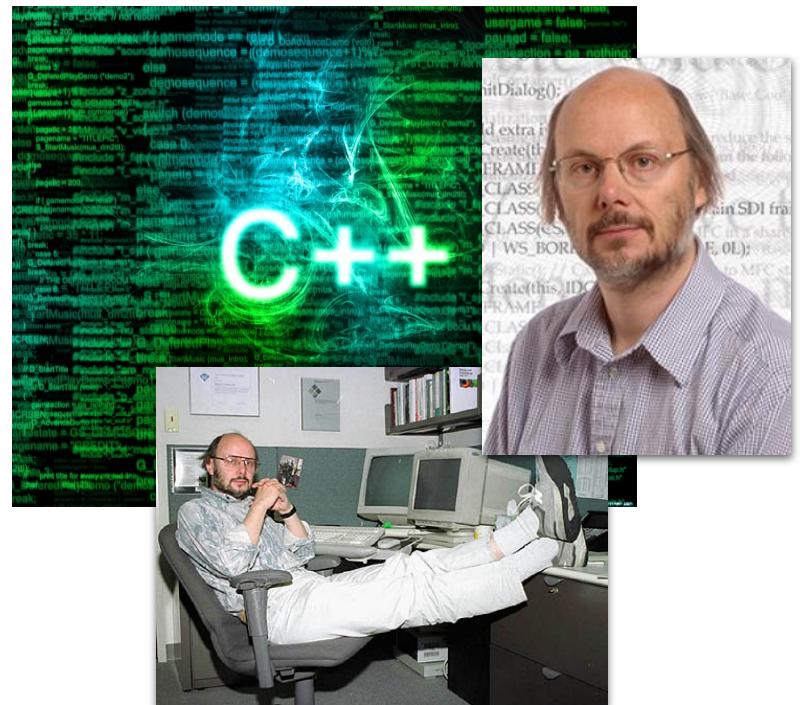
- O trabalho em cooperação é estimulado, sendo aceitável a discussão de ideias e estratégias
- Não será permitida a utilização de (parte de) códigos-fonte de outros estudantes
- **Trabalhos copiados em todo ou em parte de outros estudantes ou da Internet receberão automaticamente nota zero**

Dúvidas?
Perguntas?
Questionamentos?



A linguagem de programação C++ (1)

- Linguagem de programação multiparadigma de propósito geral padronizada pela ISO
- Considerada de médio nível, pois combina características de linguagens de alto e baixo níveis
- Criada por Bjarne Stroustrup no AT&T Bell Labs no início dos anos 1980
- Após a padronização ISO de 1998 e a posterior revisão de 2003, uma nova versão da especificação da linguagem, conhecida como C++11, foi lançada em 2011

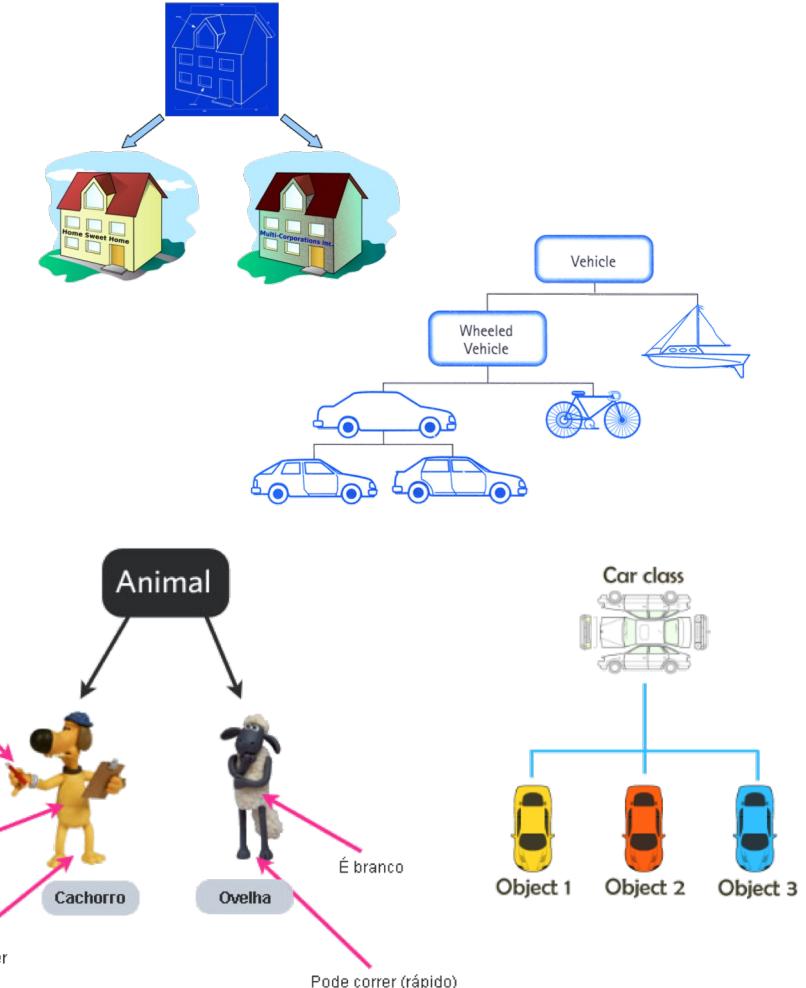


A linguagem de programação C++ (2)

- Principais objetivos
 - Inserir o paradigma de programação orientada a objetos em C
 - **Manter-se simultaneamente próxima da máquina e (da análise) do problema**
- Por quê?
 - A medida que os sistemas de software crescem, também cresce a **complexidade** associada a eles, tornando difícil satisfazer um grande número de requisitos
- **O paradigma de programação orientada a objetos** oferece uma nova forma para tratar essa complexidade
 - Organiza o código em componentes lógicos que facilitam a programação

Paradigma Orientado a Objetos

- Paradigma de programação que permite aos programadores raciocinar e solucionar problemas em termos de **objetos** diretamente associados às entidades reais
 - Mais próximo da forma como pensamos naturalmente!
- A programação orientada a objetos serve de **elo** entre os problemas existentes e as soluções computacionais
 - Grande importância na solução de problemas complexos



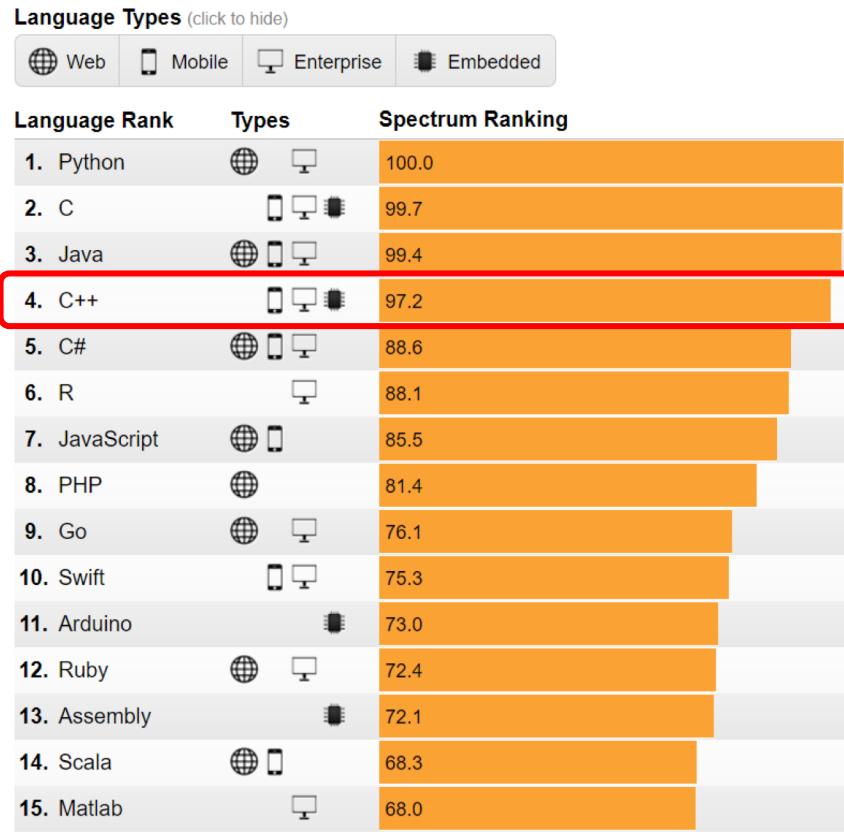
Comparativo entre C, C++ e Java

- C, C++ e Java estão entre as **linguagens de programação mais populares**
 - C segue o paradigma procedural, no qual as soluções são baseadas na decomposição de **tarefas** distintas
- Java segue o paradigma orientado a objetos
 - Soluções baseadas na decomposição de **objetos** distintos
- C++ é híbrida (ou multiparadigma), permitindo seguir os paradigmas procedural e orientado a objetos
 - Partes da solução baseadas em **tarefas** distintas e outras partes em **objetos** distintos



2016.2

Top Programming Languages 2017



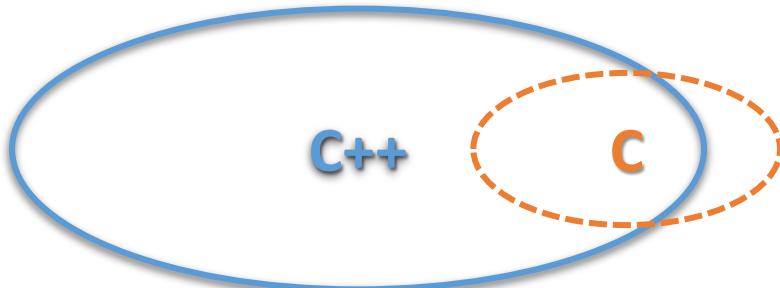
Jul 2017	Jul 2016	Change	Programming Language	Ratings	Change
1	1		Java	13.774%	-6.03%
2	2		C	7.321%	-4.92%
3	3		C++	5.576%	-0.73%
4	4		Python	3.543%	-0.62%
5	5		C#	3.518%	-0.40%
6	6		PHP	3.093%	-0.18%
7	8	▲	Visual Basic .NET	3.050%	+0.53%
8	7	▼	JavaScript	2.606%	-0.04%
9	12	▲	Delphi/Object Pascal	2.490%	+0.45%
10	55	▲	Go	2.363%	+2.20%
11	9	▼	Perl	2.334%	-0.09%
12	14	▲	Swift	2.253%	+0.29%
13	11	▼	Ruby	2.249%	+0.13%
14	10	▼	Assembly language	2.240%	-0.04%
15	17	▲	R	2.105%	+0.59%

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

Source: <http://spectrum.ieee.org/static/interactive-the-top-programming-languages-2017>

Comparativo entre C e C++ (1)

- C++ é uma extensão da linguagem C
 - Contém um **superconjunto de C**, no qual quase toda instrução correta em C é correta em C++



- Tem todas as vantagens de C, além de permitir abstração de dados e manipulação de objetos
- Ela também é bastante usada na academia devido ao seu **excelente desempenho e uma grande base de usuários**

Comparativo entre C e C++ (2)

Linguagem C	Linguagem C++
Paradigma procedural	Multiparadigma (procedural e orientado a objetos)
Inteiro como valor booleano	Tipo bool
Variáveis devem ser declaradas no início de um bloco	Variáveis podem ser declaradas em qualquer parte de um bloco
stdio.h define canais de entrada e saída (printf e scanf)	iostream define canais de entrada e saída (std::cout e std::cin)
String como vetor de caracteres	Tipo std::string
Casts simples	Novos tipos de cast
Não suporta tipos de dados abstratos	Suporta tipos de dados abstratos
Desprovida de suporte a estruturas genéricas	Suporta estruturas de código parametrizadas ou genéricas (<i>templates</i>)

Comparativo entre C e C++ (3)

Linguagem C	Linguagem C++
Duas funções não podem ter o mesmo nome	Duas funções não podem ter o mesmo protótipo
Parâmetros de funções somente podem ser passados por valor	Parâmetros de funções também podem ser passados por referência
Argumentos são sempre necessários nas chamadas de funções	Valores padrão podem ser definidos para os argumentos
Operadores de baixo nível para alocação e liberação dinâmica de memória (malloc e free)	Operadores de alto nível para alocação e liberação dinâmica de memória (new e delete)
Desprovida de mecanismo para manipulação de exceções	Dispõe de mecanismo para manipulação de exceções

Componentes da linguagem C++ (1)

- A inclusão de cabeçalhos com a diretiva `#include` diz ao compilador para inserir um outro arquivo no código fonte
 - Em C++, ela **não necessita mais da extensão do arquivo (.h)**
 - Na biblioteca padrão de C++, iostream substitui stdio.h de C
- Comentários iniciam com `//` e terminam no fim da linha

Exemplo de código em linguagem C

```
#include <stdio.h>
```

```
int main(void)
{
    /* Comentário no estilo
       de C */
    return 0;
}
```

Exemplo de código em linguagem C++

```
#include <iostream>
```

```
int main(void)
{
    // Comentário no estilo de C++
    /* Comentário no estilo de C
       também é aceito em C++ */
    return 0;
}
```

Componentes da linguagem C++ (2)

- Comando de fluxo de saída padrão
 - std::cout substitui printf, eliminando os identificadores %
 - << é um operador de inserção que direciona o valor a ser impresso para o dispositivo de saída
- O manipulador std::endl substitui o caractere '\n'

Exemplo de código em linguagem C

```
#include <stdio.h>

int main(void)
{
    int x = 10;
    printf("Iniciando...\n");
    printf("%i, %f\n", x, 20.5f);
    return 0;
}
```

Exemplo de código em linguagem C++

```
#include <iostream>

int main(void)
{
    int x = 10;
    std::cout << "Iniciando..." << std::endl;
    std::cout << x << ", " << 20.5f << std::endl;
    return 0;
}
```

Componentes da linguagem C++ (3)

- Comando de fluxo de entrada padrão
 - std::cin substitui scanf, no qual identificadores % e operador de endereçamento & não são mais necessários
 - >> é um operador de extração que recebe um valor digitado pelo usuário através do dispositivo de entrada

Exemplo de código em linguagem C

```
#include <stdio.h>

int main(void)
{
    int x;
    float y;
    scanf("%i %f", &x, &y);
    return 0;
}
```

Exemplo de código em linguagem C++

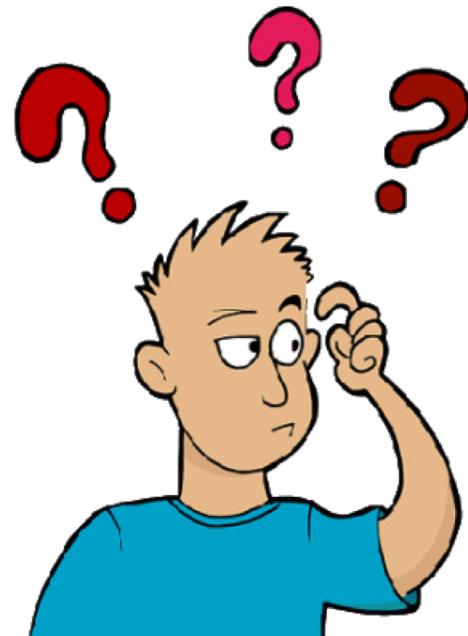
```
#include <iostream>

int main(void)
{
    int x;
    float y;
    std::cin >> x >> y;
    return 0;
}
```

Compilando tudo

- Para compilar todos os arquivos e gerar o nosso primeiro programa em C++, utilizaremos o compilador g++
- Processo de compilação:
`g++ -Wall -pedantic teste.cpp main.cpp –o programa`
 - Note que apenas os arquivos de corpo (.cpp) são passados para o compilador
 - Como resultado da compilação, será gerado o arquivo executável de nome programa
 - Os parâmetros -Wall -pedantic são aqui usados para indicar ao compilador que qualquer tipo de mensagem de aviso (*warning*) deve ser interpretada como um erro, devendo o programador corrigir o código que dá origem ao aviso
- Execução: basta executar o arquivo de nome programa

Alguma Questão?





KEEP
CALM
AND
CODE
C++;

```
while(alive)
{
    eat();
    sleep();
    code();
}
```