

IMD0030

LINGUAGEM DE PROGRAMAÇÃO I

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

(material baseado nas notas de aula do Prof. Silvio Sampaio e Prof. César Rennó-Costa)

Apresentação

- Professor responsável
 - Renan Cipriano Moioli
 - renan.moioli@imd.ufrn.br (use use no início do título [IMD0030])
 - Sala A208
- Dúvidas sobre a disciplina preferencialmente via fórum no SIGAA
- Atendimento extra-classe agendado via email

Objetivos 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

Ver no SIGAA



Aula	Dia	Tema
	1	12/02/2019 Apresentação do Plano de Aulas e Introdução à linguagem C++
	2	14/02/2019 Introdução à Linguagem de Programação C++. Modularização e Compilação
	3	19/02/2019 Depuração (com o GDB)
	4	21/02/2019 Profiling (com o Gprof).
	5	26/02/2019 Recursividade
	6	28/02/2019 Laboratório 1
	7	07/03/2019 Introdução à orientação a objetos: Classes e objetos
	8	09/03/2019 Revisão, exercícios e esclarecimento de dúvidas.
	9	12/03/2019 Construtores e destrutores
	10	14/03/2019 Sobrecarga de funções e passagem de parâmetro
	11	19/03/2019 Sobrecarga de operadores
	12	21/03/2019 Laboratório 2: Classes e Objetos
	13	23/03/2019 Revisão, exercícios e esclarecimento de dúvidas
	14	26/03/2019 Avaliação Teórica I.
	15	28/03/2019 Gerenciamento de memória.
		02/04/2019 Não haverá aula
		04/04/2019 Não haverá aula
	16	09/04/2019 Ponteiros inteligentes.
	17	11/04/2019 Programação genérica: templates de funções e templates de classes.
	18	13/04/2019 Discussão da Avaliação I.
	19	16/04/2019 Herança.
	20	23/04/2019 Classes abstratas.
	21	25/04/2019 Manipulação de arquivos.
	22	30/04/2019 Laboratório 3: Herança, Classes Abstratas e Manipulação de arquivos
	23	02/05/2019 Implementação de Tipos Abstratos de Dados (TADs) genéricos.
	24	04/05/2019 Dúvidas e acompanhamento do projeto
	25	07/05/2019 Laboratório 4: Programação genérica e utilização de TADs genéricos.
	26	09/05/2019 Standard Template Library (STL): Containers e iterators.
	27	14/05/2019 Standard Template Library (STL): Biblioteca algorithms e type casting.
	28	16/05/2019 Laboratório 5: Standard Template Library (STL)
	29	18/05/2019 Namespaces e bibliotecas
	30	21/05/2019 Avaliação Teórica II.
	31	23/05/2019 Laboratório 6: Namespaces e bibliotecas
	32	25/05/2019 Discussão da Avaliação Teórica II. Uso de bibliotecas externas.
	33	28/05/2019 Discussão do projeto
	34	30/05/2019 Laboratório 7: Uso de bibliotecas externas
		30/05/2019 Data limite para envio do projeto
	35	04/06/2019 Tratamento de exceções
	36	06/06/2019 Resumo do curso e fechamento da disciplina
		11/06/2019 Prova de reposição
		13/06/2019 Não haverá aula
		18/06/2019 Não haverá aula
		25/06/2019 Não haverá aula
		27/06/2019 Não haverá aula

Metodologia

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

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 (sugestão: *Doxygen*)
- Organização de programas complexos na forma de funções modulares e arquivos
- Teste sistemático de programas na forma de casos de teste

As regras do jogo

- 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

- Exercícios de programação (laboratórios - dupla)
 - Aplicação dos conceitos vistos nas aulas expositivas
 - Estímulo ao desenvolvimento das habilidades de programação em C++
- Duas avaliações individuais e presenciais (unidades I e II)
- Projeto final de programação
 - Solução de problemas por meio de programas implementados na linguagem C++
 - Realizado em duplas
 - Tema LIVRE a ser definido por grupo
- Os laboratórios, avaliações ou projeto final poderão envolver conceitos vistos na disciplina **IMD0029 – Estruturas de Dados Básicas I** ou equivalente

Avaliação

Média

$$Média = \frac{U_1 + U_2 + U_3}{3} \quad U_i = \frac{\frac{\sum_{j=1}^N L_j}{N} + 2A_i}{3}$$

L_j : laboratórios da unidade $L_j = \{0; 2,5; 5,0; 7,5; 10,0\}$

N : número de laboratórios da unidade

$A_{1,2}$: avaliação teórica

A_3 : projeto

Avaliação

Critérios de avaliação

- 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 da escrita

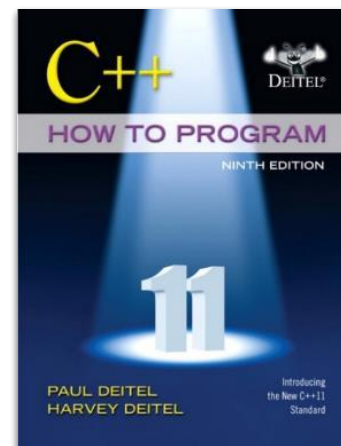
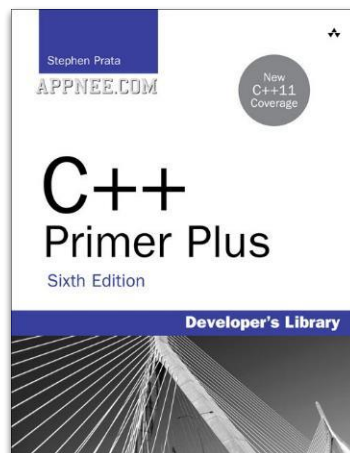
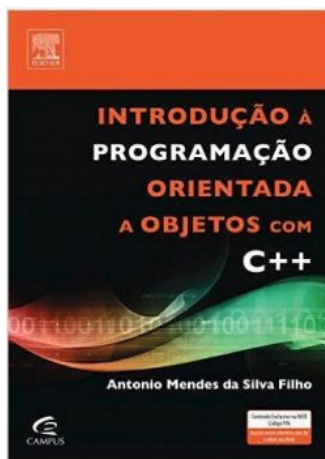
Avaliação

Rendimento acadêmico

- Ausência a alguma das avaliações ou não entrega de algum dos laboratórios 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

Links úteis

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



Observações gerais

Faltas às aulas presenciais

Não existe abono de faltas

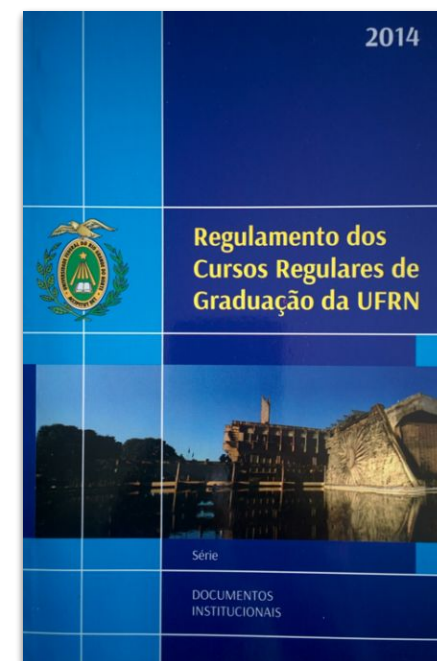
Art. 112 do Regulamento dos Cursos de Graduação



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



Regulamento dos Cursos Regulares de Graduação

Resolução 171/2013 – CONSEPE/UFRN

Principais Novidades e Modificações Perguntas e Respostas Frequentes



Autoria:
Assessoria Acadêmica do CCET

Revisão:
Pró-Reitoria de Graduação



[http://arquivos.info.ufrn.br/arquivos/2014230053a0961803064f981f2aba6d3/Cartilha do novo regulamento de graduao-pginaPROGRAD.pdf](http://arquivos.info.ufrn.br/arquivos/2014230053a0961803064f981f2aba6d3/Cartilha_do_novo_regulamento_de_graduao-pginaPROGRAD.pdf)

Observações gerais

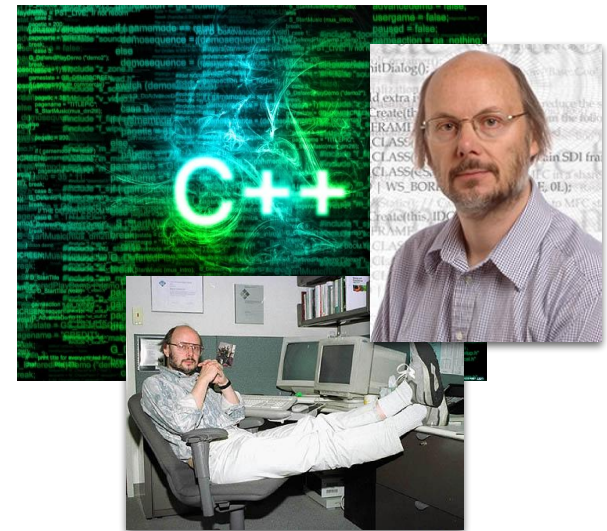
Sobre plágio

- 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****



A linguagem de programação C++

- **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

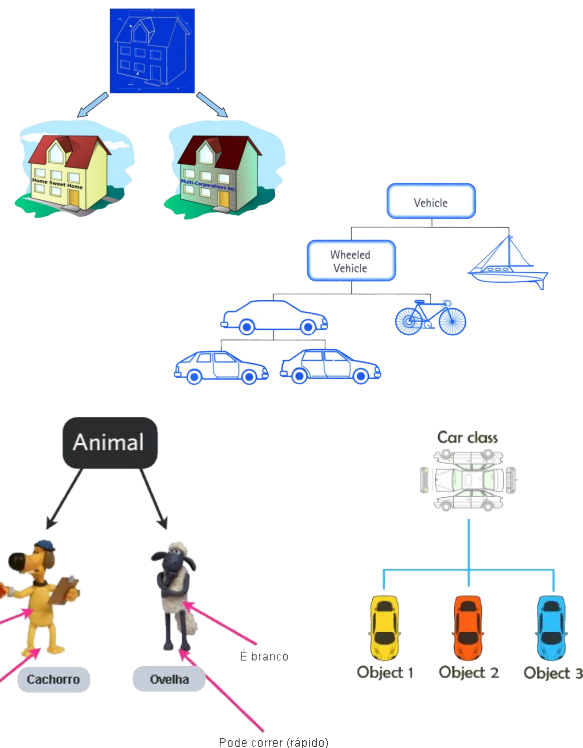


A linguagem de programação C++

- 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

Top Programming Languages 2017-2018

Language Types (click to hide)

☒ Web
 ☒ Mobile
 ☒ Enterprise
 ☒ Embedded

Language Rank	Types	Spectrum Ranking
1. Python	   	100.0
2. C	  	99.7
3. Java	  	99.4
4. C++	  	97.2
5. C#	  	88.6
6. R		88.1
7. JavaScript	 	85.5
8. PHP		81.4
9. Go	 	76.1
10. Swift	 	75.3
11. Arduino		73.0
12. Ruby	 	72.4
13. Assembly		72.1
14. Scala	 	68.3
15. Matlab		68.0

Language Rank	Types	Spectrum Ranking
1. Python	   	100.0
2. C++	  	99.7
3. Java	  	97.5
4. C	  	96.7
5. C#	  	89.4
6. PHP		84.9
7. R		82.9
8. JavaScript	 	82.6
9. Go	 	76.4
10. Assembly		74.1

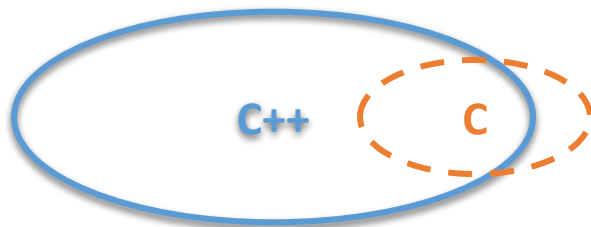
Source: <http://spectrum.ieee.org/static/interactive-the-top-programming-languages-2017>

<https://spectrum.ieee.org/at-work/innovation/the-2018-top-programming-languages>

Comparativo entre C e C++

- **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++

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

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++

- 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++

- 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++

- 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



Aula 01

Apresentação da disciplina e introdução ao C++

Aula 02

Modularização e compilação