

Programação e Desenvolvimento de Software 2

Luigi D. C. Soares (luigi.domenico@dcc.ufmg.br)

Material adaptado: prof. Douglas Macharet
prof. Flávio Figueiredo

Ementa

- Armazenamento de dados em memória
- Uso de tipos abstratos de dados
- Boas práticas de desenvolvimento
- Introdução à orientação a objetos
- Desenvolvimento de programas corretos
- Programação defensiva

Objetivos do curso

Apresentar técnicas básicas de desenvolvimento, teste e análise de programas de computador, para a resolução de problemas de forma eficaz. É esperado que nesta disciplina os alunos desenvolvam seus primeiros programas de tamanho moderado, motivando a necessidade de uso de boas práticas de desenvolvimento, fixando os conteúdos abordados através de atividades práticas. Concluindo o curso, os alunos deverão dominar as técnicas mais básicas utilizadas no processo de desenvolvimento de software.

Bibliografia

- **Clean Code: A Handbook of Agile Software Craftsmanship.**
Robert C. Martin.
Prentice Hall, 2008.
- **Code Complete: A Practical Handbook of Software Construction.**
Steve McConnell.
Microsoft Press, 2004. 2nd Edition.
- **Effective C++: 55 Specific Ways to Improve Your Programs and Designs.**
Scott Meyers.
Addison-Wesley Professional, 2005. 3rd Edition.
- **A Tour of C++.**
Bjarne Stroustrup.
Addison-Wesley Professional, 2013. 1st Edition.

Moodle

- Todas as informações relacionadas ao curso
 - Avisos
 - Notas de aulas
 - Atividades práticas
 - Projeto
 - Discussão de dúvidas

Critérios de avaliação

- Provas teóricas (2x20): 40 pontos
- Atividades práticas: 30 pontos
- Projeto: 30 pontos
 - Pontos extras (criatividade, extras, etc.)

Critérios de avaliação

- Provas
 - Material de referência (livros, slides, etc)
 - Conteúdo visto durante a aula
 - Exercícios de laboratório
- Revisão da correção
 - Durante o horário de atendimento
 - Em até uma semana após divulgação

Critérios de avaliação

- Projeto
 - Especificação (entrega parcial)
 - Código: conceitos, funcionamento, ...
 - Documentação: clareza e coesão, conteúdo, ...
- Em alguns casos pode ocorrer entrevista

Critérios de avaliação

- Haverá tolerância ZERO com cópia/cola
 - Nota será automaticamente anulada
 - Será aberto processo disciplinar



Notas e frequência

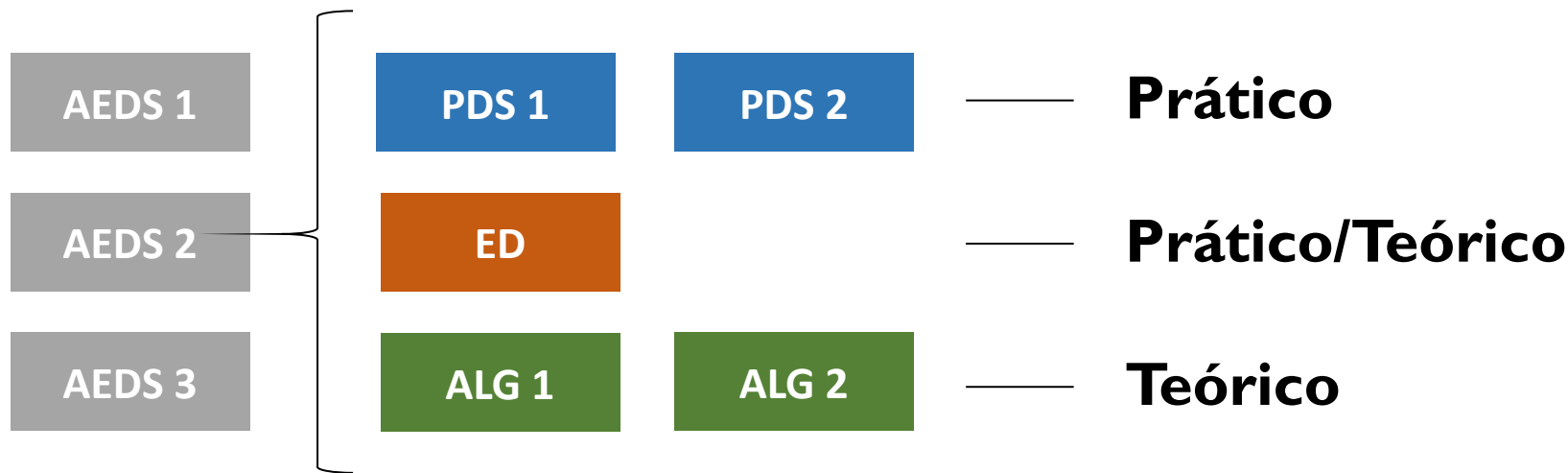
- Frequência é **obrigatória!** ($\geq 75\%$)
- Caso contrário, se for infrequente ($< 75\%$)
 - Conceito F
 - Não tem direito a exame especial

Contato

- Email:
 - luigi.domenico@dcc.ufmg.br
 - Adicionar [PDS 2] no assunto
- Sala: DCC – 4327 (combinar horário)

PDS 2

- O que você ouviu sobre PDS 2?
- Quais as suas expectativas para o curso?



PDS 2



PDS 2

Boas práticas de desenvolvimento

- O que pode acontecer se um engenheiro civil não aplicar boas práticas em seu trabalho?



PDS 2

Boas práticas de desenvolvimento

The New York Times

Self-Driving Uber Car Kills Pedestrian in Arizona, Where Robots Roam

By Daisuke Wakabayashi

March 19, 2018

The
Guardian

Self-driving Uber kills Arizona woman in first fatal crash involving pedestrian

Tempe police said car was in autonomous mode at the time of the crash and that the vehicle hit a woman who later died at a hospital

Sam Levin and Julia Carrie Wong in San Francisco

Mon 19 Mar 2018 22.48 GMT

PDS 2

Boas práticas de desenvolvimento

DRIVERLESS CAR SAFETY—

Report: Software bug led to death in Uber's self-driving crash

Sensors detected Elaine Herzberg, but software reportedly decided to ignore her.

TIMOTHY B. LEE - 5/7/2018, 7:12 PM



Enlarge / NTSB officials inspecting the vehicle that killed Elaine Herzberg in a March crash in Arizona.

PDS 2

Boas práticas de desenvolvimento

- O que é um sistema computacional complexo?
 - Quais características podem ser consideradas?
 - Algoritmo complexo, funcionalidades, tamanho, ...
 - Quais sistemas vocês julgam serem complexos?
 - Google, Battlefield, Whatsapp, Uber, ...

PDS 2

Boas práticas de desenvolvimento

- Programas de porte médio (a muito grande)
 - Como desenvolver esses tipos de sistema?
 - Quando acaba o desenvolvimento de um software?
 - É preciso saber desenvolver em equipe?

Como desenvolver um programa complexo tendo certeza que de fato funcionará e será de qualidade?

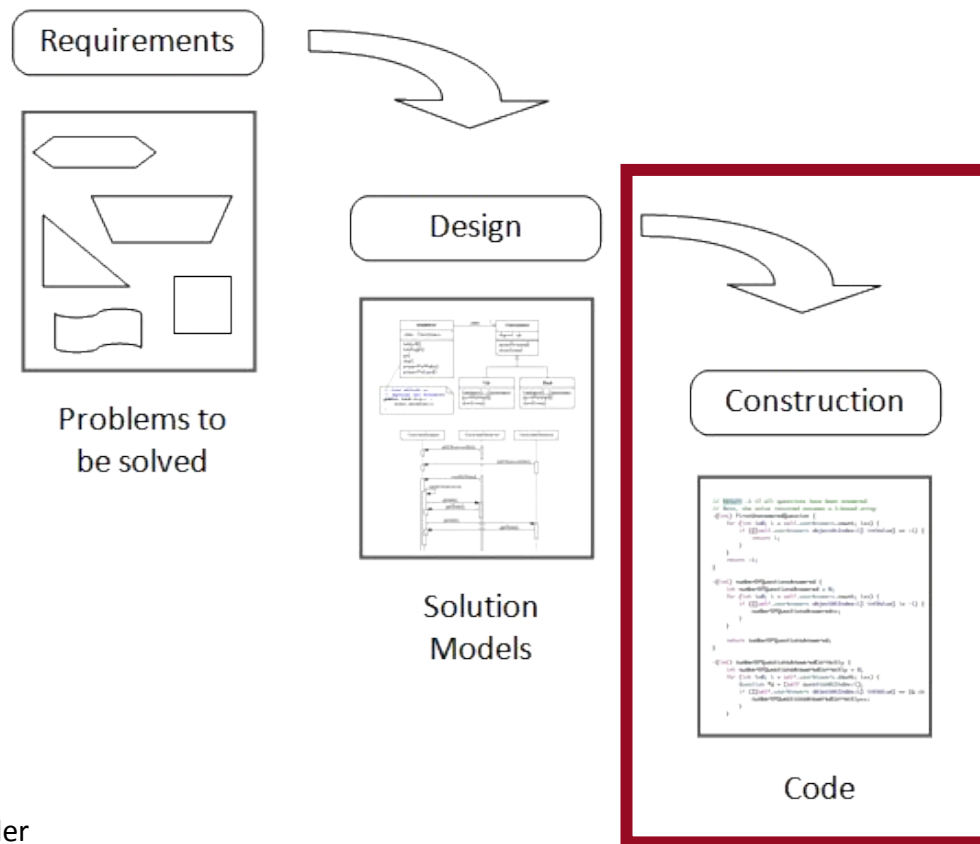
- Módulos: trabalho em equipe, controle, reuso, ...
- Desafios
 - Quais módulos devem existir? Como implementar?
 - Como assegurar a qualidade? É possível?

PDS 2

Programação Orientada a Objetos

- Paradigma de programação
 - Estruturação e execução do programa
 - Maneira de organizar o código
- Não é associado a uma linguagem específica
- Como escolher um paradigma?
 - Programa que soluciona equações do 2º grau
 - Sistema de gestão de uma universidade























PDS 2



Is Design Dead? – Martin Fowler

<https://www.martinfowler.com/articles/designDead.html>

- Utilizaremos a linguagem C++

Language Rank	Types	Spectrum Ranking
1. Python	  	100.0
2. C++	  	98.4
3. C	  	98.2
4. Java	  	97.5
5. C#	  	89.8
6. PHP		85.4
7. R		83.3
8. JavaScript	 	82.8
9. Go	 	76.7
10. Assembly		74.5

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

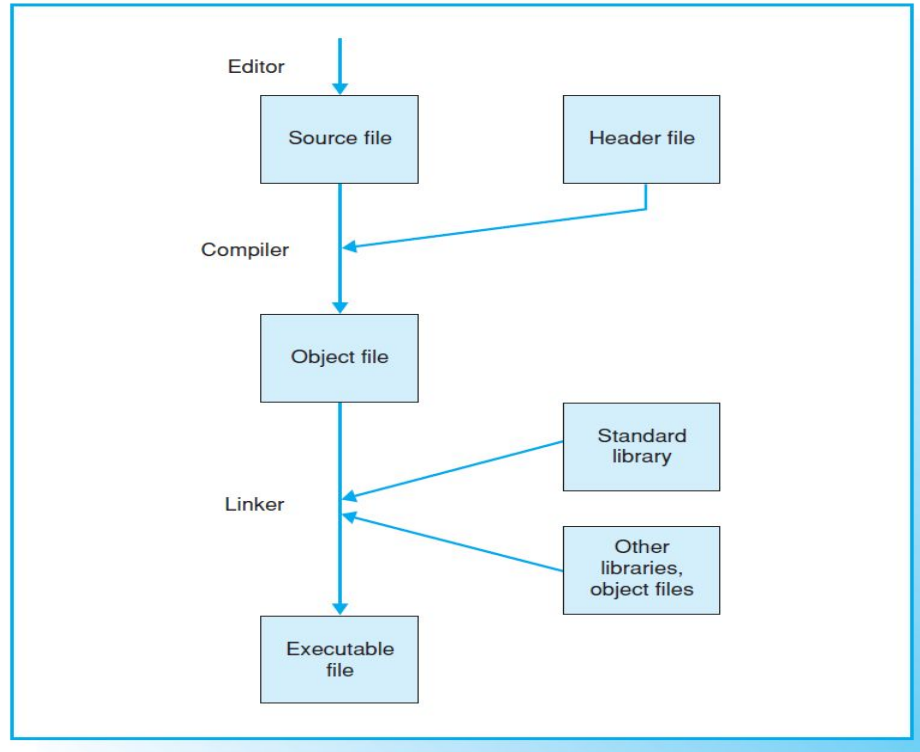
PDS 2

C++

- Extensão da linguagem C
 - Características de alto e baixo nível
 - *Quase* (mas não exatamente) superconjunto de C
- *C with Classes* □ C++
- Fonte de inspiração para Java, C#, ... (sintaxe)

PDS 2

C++



PDS 2

C++

C

```
#include <stdio.h>

int main() {
    printf("Hello World");
    return 0;
}
```

C++

```
#include <iostream>

using namespace std;
int main() {
    cout << "Hello world!" << endl;
    return 0;
}
```

```
$ g++ hello.cpp -o hello
$ ./hello
Hello world!
```

PDS 2

C++ – String

```
#include <iostream>
#include <string>

using namespace std;

int main() {
    string curto = "Hello World!";
    string longo = "Essa eh uma string grande para o exemplo!";
    cout << curto << endl;
    cout << longo << endl;
    cout << longo.length() << endl;
    return 0;
}
```

PDS 2

C++ – Entrada/Saída (console)

```
#include <iostream>
using namespace std;
int main() {
    string s;
    cin >> s; // Faz **uma** leitura apenas
    cout << "[" << s << "]" << endl;
    int i;
    // Faz leituras enquanto a entrada respeita o tipo (int)
    while(cin >> i) {
        cout << "[" << i << "]" << endl;
    }
    return 0;
}
```

PDS 2

C++ – Entrada/Saída (console)

```
#include <iostream>
#include <sstream>

using namespace std;

int main() {
    string line;
    // Leitura de uma linha inteira de entrada
    while (getline(cin, line)) {
        // Parsing da linha inteira usando string stream
        istringstream row(line);
        // Leitura de **cada** variavel na linha
        string s;
        row >> s;
        int i;
        row >> i;
        cout << "[" << s << "|" << i << "]" << endl;
    }
    return 0;
}
```

<http://www.cplusplus.com/reference/string/string/getline/>

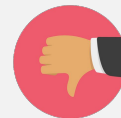
<http://www.cplusplus.com/reference/sstream/istringstream/istringstream/>

PDS 2

C++ – Entrada/Saída (arquivos)

```
#include <fstream>
#include <iostream>
#include <string>

using namespace std;
int main() {
    ifstream in("entrada.txt", fstream::in);
    ofstream out("saida.txt", fstream::out);
    string line;
    while (getline(in, line)) {
        cout << line << endl;
        out << line << endl;
    }
    in.close();
    out.close();
    return 0;
}
```



PDS 2

C++ – Entrada/Saída (arquivos)

```
#include <fstream>
#include <iostream>
#include <string>

using namespace std;
int main() {
    ifstream in("entrada.txt", fstream::in);
    if (!in.is_open()) { return 1; }
    ofstream out("saida.txt", fstream::out);
    if (!out.is_open()) { return 1; }
    string line;
    while (getline(in, line)) {
        cout << line << endl;
        out << line << endl;
    }
    in.close();
    out.close();
    return 0;
}
```



<http://www.cplusplus.com/doc/tutorial/files/>

PDS 2

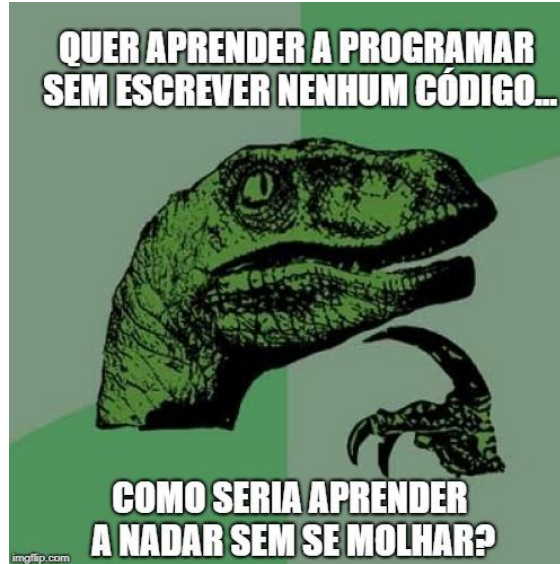
C++

Ano	Padrão C++	Nome Informal
1998	ISO/IEC 14882:1998	C++98
2003	ISO/IEC 14882:2003	C++03
2011	ISO/IEC 14882:2011	C++11
2014	ISO/IEC 14882:2014	C++14
2017	ISO/IEC 14882:2017	C++17
2020	ISO/IEC 14882:2020	C++20

```
$ g++ -std=c++17 -Wall hello.cpp -o hello
```

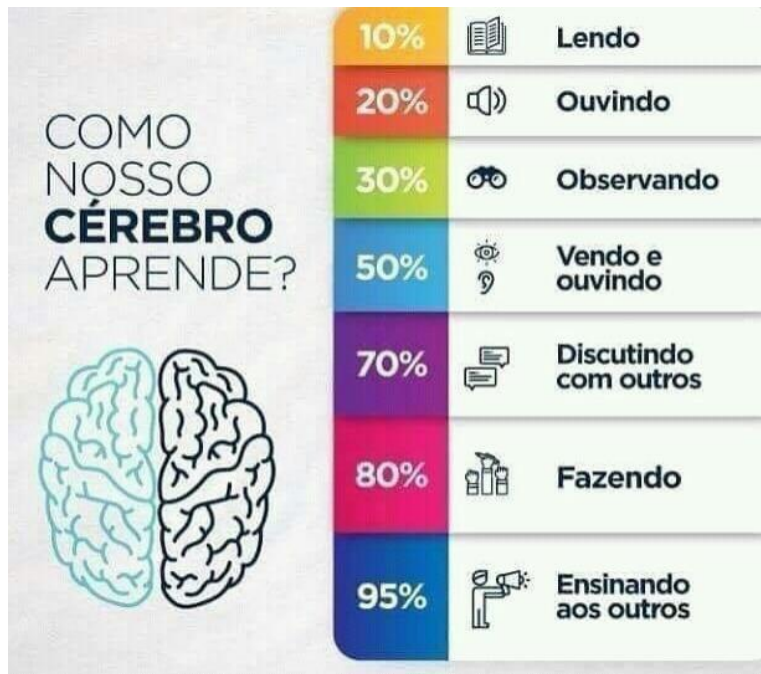
PDS 2

- Não é um curso técnico de programação!
 - Linguagem utilizada para aplicar os conceitos
 - O foco não é simplesmente a sintaxe!
- Objetivos
 - Desenvolvimento de bons programas
 - Prática e experiência em programação
 - Grandes projetos □ Grandes responsabilidades
 - Princípios e conceitos do paradigma OO



**É claro que também será necessário
familiaridade e experiência com a linguagem!**

PDS 2



Sim, você precisa gastar **muito** tempo extraclasse (provavelmente mais do que em sala)!

Tarefas

- Cadastro no GitHub + Tutoriais:
<https://docs.github.com/pt/get-started>
- Preencher a planilha de grupos e temas
- Preparar ambiente C++. Sugerimos:
 - Windows Subsystem for Linux (WSL) 2
 - `$ sudo apt update`
 - `$ sudo apt install g++ gdb git make`
 - Visual Studio Code
- Fazer primeiros VPLs