UFMG
UNIVERSIDADE FEDERAL DE MINAS GERAIS

Programação e Desenvolvimento de Software 2

Prof. Douglas G. Macharet
douglas.macharet@dcc.ufmg.br

DCC
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

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.

Ementa

- Armazenamento de dados em memória
  - Revisão de ponteiros, alocação dinâmica, ...
- Uso de tipos abstratos de dados
  - Conjuntos, dicionários, listas, vetores
- Boas práticas de desenvolvimento
  - Controle de versão, refatoração, bibliotecas, ...
- Introdução à orientação a objetos
  - Classes, encapsulamento, herança, polimorfismo
- Desenvolvimento de programas corretos
  - Tipos de erros, testes, revisão, depuração, ...
- Programação defensiva
  - Validações, tratamento de exceções, ...

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. 3nd 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 (15–20): 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
  - Código
    - Aplicação dos conceitos, funcionamento, ...
  - Documentação
    - Clareza e coesão, conteúdo, ...
  - Apresentação
- 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

- Se o aluno possuir aproveitamento >= 60 pts
  - Não reprovo por frequência
- Caso contrário, se for infrequente (< 75%)
  - Não altero conceito
  - Não ajudo a passar
  - Não tem direito a exame especial
  - Nota < 60: conceito F

Notas e frequência

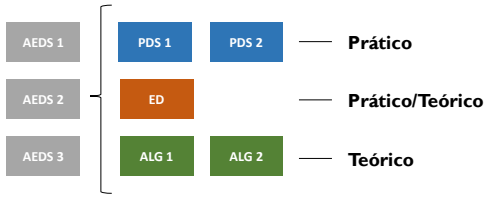
- Lista de presença em todas as aulas
- Mas não venha se não estiver interessado
- Por que você deve vir?
  - Tirar dúvidas
  - Participar de discussões em sala
  - Dicas sobre possíveis questões
  - Mencionar algo não contido nos livros/slides

Contato

- Email:
  - douglas.macharet@dcc.ufmg.br
  - Adicionar [DCC204] no assunto
- Sala:
  - ICEx – 4314 (Anexo U)

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

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

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

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, ...
- 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?

PDS 2
Modularidade

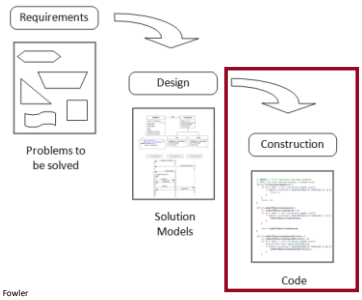
Como desenvolver um programa complexo tendo certeza que de fato funcionará e será de qualidade?

- Módulos (decomposição sistemática)
  - Trabalho em equipe, controle, reuso, ...
- Desafios
  - Quais módulos devem existir?
  - Como implementar cada um desses módulos?
  - 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

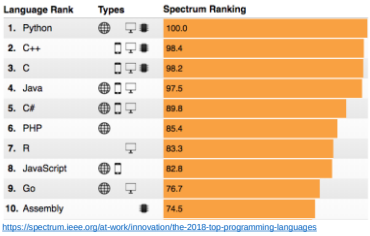
PDS 2



Is Design Dead? – Martin Fowler
https://www.martinfowler.com/articles/designDead.html

PDS 2

- Utilizaremos a linguagem C++



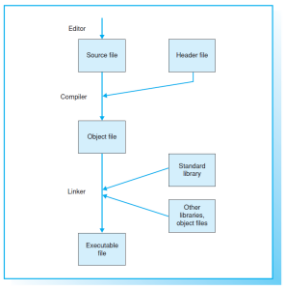
Language Rank	Types	Spectrum Ranking
1. Python		100.0
2. C++		98.4
3. C		98.2
4. Java		97.5
5. C#		89.8
6. PHP		85.4
7. R		83.3
8. JavaScript		82.8
9. Go		76.7
10. Assembly		74.5

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
  - Não pode ser considerado um super conjunto
    - Não implementa o C completamente!
- *C with Classes* → C++
- Fonte de inspiração para Java, C#, ... (sintaxe)

PDS 2
C++



PDS 2
C++

C

```c
#include <stdio.h>

int main() {
  printf("Hello World");

  return 0;
}
```

C++

```cpp
#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
  - Tipo de dado muito útil (biblioteca padrão)
  - Guarda uma sequência de caracteres

```cpp
#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)

- Biblioteca padrão: cin/cout

```cpp
#include <iostream>

using namespace std;

int mainx() {

  string s;
  cin >> s;  // Faz **uma** leitura apenas
  cout << "|" << s << "|" << endl;

  int i;

  // Faz leituras enquanto as variaveis de
  // entrada resultam o tipo especificado (int).
  while(cin >> i) {
    cout << "|" << i << "|" << endl;
  }

  return 0;
}
```

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

using namespace std;

int main() {

  string line;
  // Faz uma leitura de entrada
  while (getline(cin, line)) {
    // Insere os dados lidos 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/istream/istringstream/istringstream/

PDS 2
C++ – Entrada/Saída (arquivos)

```cpp
#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)

```cpp
#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++

- C++11

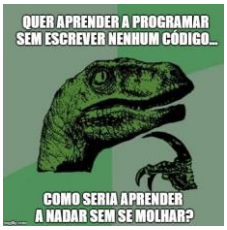
Ano	Padrão C++	Nome Informal
1998	ISO/IEC 14882:1998	C++98
2003	ISO/IEC 14882:2003	C++03
2011	ISO/IEC 14882:2011	C++11
2014	ISO/IEC 14882:2014	C++14
2017	ISO/IEC 14882:2017	C++20

```
$ g++ -std=c++11 -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

PDS 2



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

PDS 2

"deliberative, sustained practice; learning by doing; and collaborating with other programmers on projects"
Peter Norvig

Teach Yourself Programming in Ten Years – Peter Norvig
http://norvig.com/21-days.html

https://abstrusegoose.com/249

PDS 2

https://en.wikipedia.org/wiki/Outliers_(book)

10000 horas em **1** semestre!

PDS 2

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

Tarefas

- Cadastro no GitHub
  - Leitura de tutoriais[1,2]
- Preencher planilha de grupos
  - Data limite: 12/03
  - Entre 3 e 4 componentes
- Preparar ambiente C++ (ferramentas)
- Fazer primeiros exercícios VPL
  - Lab00Ex01, Lab00Ex02, Lab00Ex03

[1] https://guides.github.com/
[2] https://www.youtube.com/playlist?list=PLWjCJDeWfDdcjB_NSNFfsbHrjCfetYJWJ