UFV
Universidade Federal de Viçosa

INF 112 - Programação II

Apresentação da Disciplina

Pessoal

- Julio Cesar Soares dos Reis
  - Contato
    - jreis@ufv.br
  - Dúvidas?
    - Qualquer horário assumindo disponibilidade
    - Pessoalmente/PVA Moodle/Email

Sobre a matéria

- INF112 → Maior foco em desenvolvimento
  - Programação Orientada a Objetos
  - Uso de Estruturas de Dados
  - **Maior foco em desenvolvimento**
- INF213 → Maior foco em algoritmos
  - Implementação de Estruturas de Dados
  - Ordenação
  - **Maior foco em algoritmos**

INF112 - Programação II

A ideia é aprender como abstrair o mundo em software

- Entender o problema
- Modelar os dados
- Codificar a solução

Ementa

- Desenvolvimento de software
- **Programação orientada a objetos**
- Uso e aplicação de estruturas de dados
- Entendimento da memória
- Boas práticas
  - Tratamento de Exceções
- Vamos começar revisando alguns assuntos

Avaliação

- 2 provas
  - Cada uma valendo 25 pontos (50 pontos no total)
- Laboratórios + Trabalhos Práticos
  - 30 pontos no total
- I Trabalho Prático Final
  - Valendo 20 pontos

Trabalho Prático Final

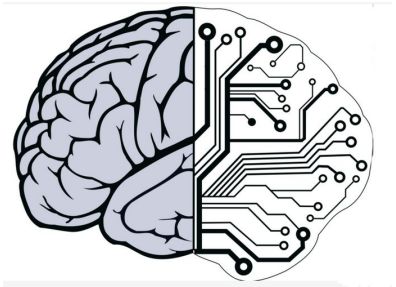
- Tema da sua escolha
  - Existe uma lista de temas possíveis
- Em Grupo
- Código no github
  - Repositório privado
  - Sua tarefa: criar conta no github

UFV
Universidade Federal de Viçosa

Programação Estruturada X POO

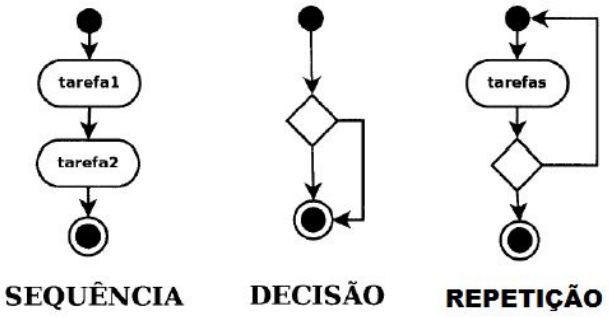
Programação Estruturada (PE)

- Paradigma de programação
  - As instruções mudam o estado do programa
  - Programas imperativos (ações)
  - "Pensamento de programação mais voltado ao pensamento da máquina"
  - É bastante eficiente para solucionar problemas simples e diretos



Programação Estruturada (PE)

- Os programas podem ser reduzidos à três estruturas:
  - Sequência;
  - Decisão (desvio);
  - e Iteração (repetição).



Programação Estruturada (PE)

Programação Estruturada (PE)

- Uso de variáveis (dados armazenados em memória) e funções que executam regras implementadas

Programação Estruturada (PE)

- Uso de variáveis (dados armazenados em memória) e funções que executam regras implementadas
  - Exemplo de variáveis

```
double pi = 3.1415;
int idade;
char sexo;
<tipo> var;
```

Programação Estruturada (PE)

- Uso de variáveis (dados armazenados em memória) e funções que executam regras implementadas
  - Exemplo de funções

```
int soma(int a, int b){
  return a + b;
}
```

Programação Estruturada (PE)

- Como resolver problemas muito grandes?

Programação Estruturada (PE)

- Como resolver problemas muito grandes?
  - Construí-lo a partir de partes menores

Programação Estruturada (PE)

- Módulos compiláveis
  - Solucionam uma parte do problema
  - Dados X Manipulação
    - Abstração fraca para problemas mais complexos

Saber programar é o passo inicial..

- O que já sabemos:
  - if, while, else, for, funções

- Como modelar um programa?
- Como representar um conceito?

Desenvolvimento de Software

Exemplos do mundo real

- Como desenvolver um sistema de banco?

Desenvolvimento de Software

Exemplos do mundo real

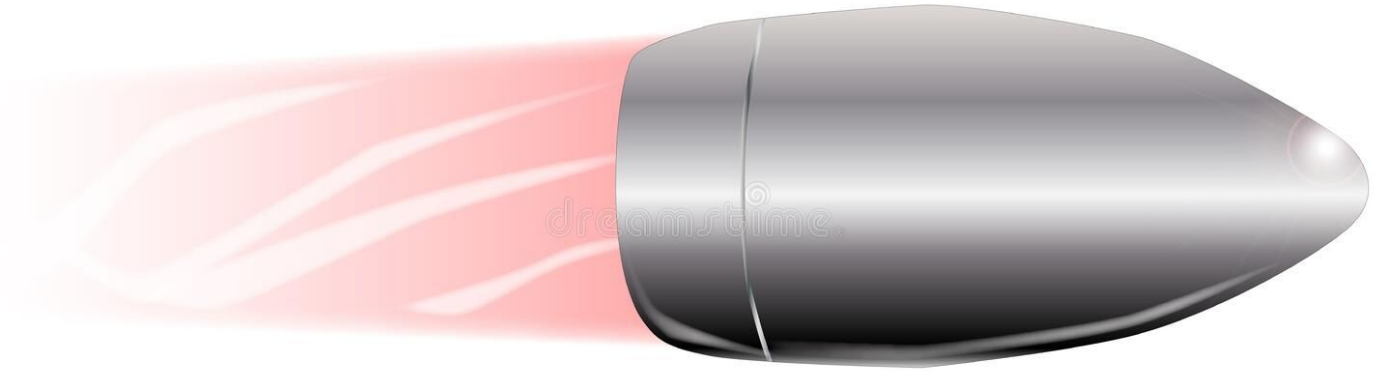
- Como desenvolver um sistema de banco?
- Clientes
- Transações
- Contas
- . . .

Programação Orientada a Objetos (POO)

- Sistemas maiores e mais complexos
  - Aumentar a produtividade no desenvolvimento
  - Diminuir a chance de problemas
  - Facilitar a manutenção/extensão

Programação Orientada a Objetos (POO)

- Programação Orientada a Objetos
  - Tem apresentado bons resultados
  - "Não é uma bala de prata!"



Programação Orientada a Objetos (POO)

- História
  - Desenvolvimento de Hardware
    - Pedaços simples de hardware (chips) unidos para se montar um hardware mais complexo
  - Amadurecimento dos conceitos
    - Simula (60's)
    - Smalltalk (70's)
    - C++ (80's)
    - Java (90's)


O que é
Programação
Orientada a
Objetos?

Programação Orientada a Objetos (POO)

- Paradigma de programação
  - "Pensamento de programação mais voltado ao pensamento da máquina"..."ensinar a máquina a pensar como os humanos..."

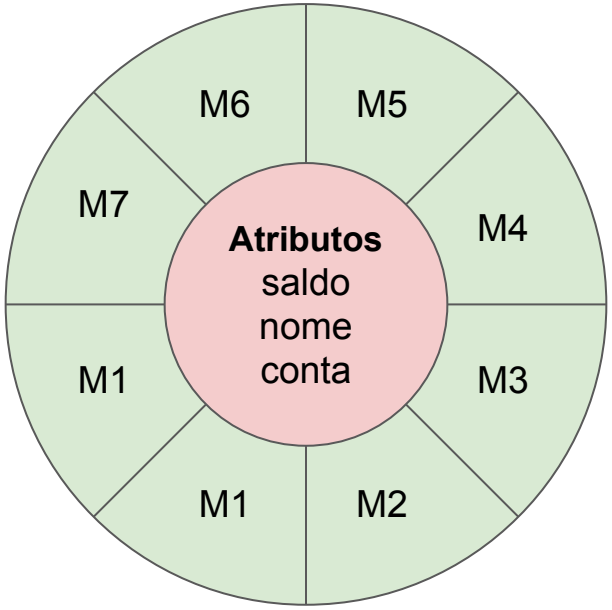
Programação Orientada a Objetos (POO)

- Paradigma de programação
  - "Pensamento de programação mais voltado ao pensamento da máquina"..."ensinar a máquina a pensar como os humanos..."

Programação Orientada a Objetos (POO)

Uma das formas de modelar o mundo

- Cada entidade do mundo real **pode** virar um objeto. Será que deve?



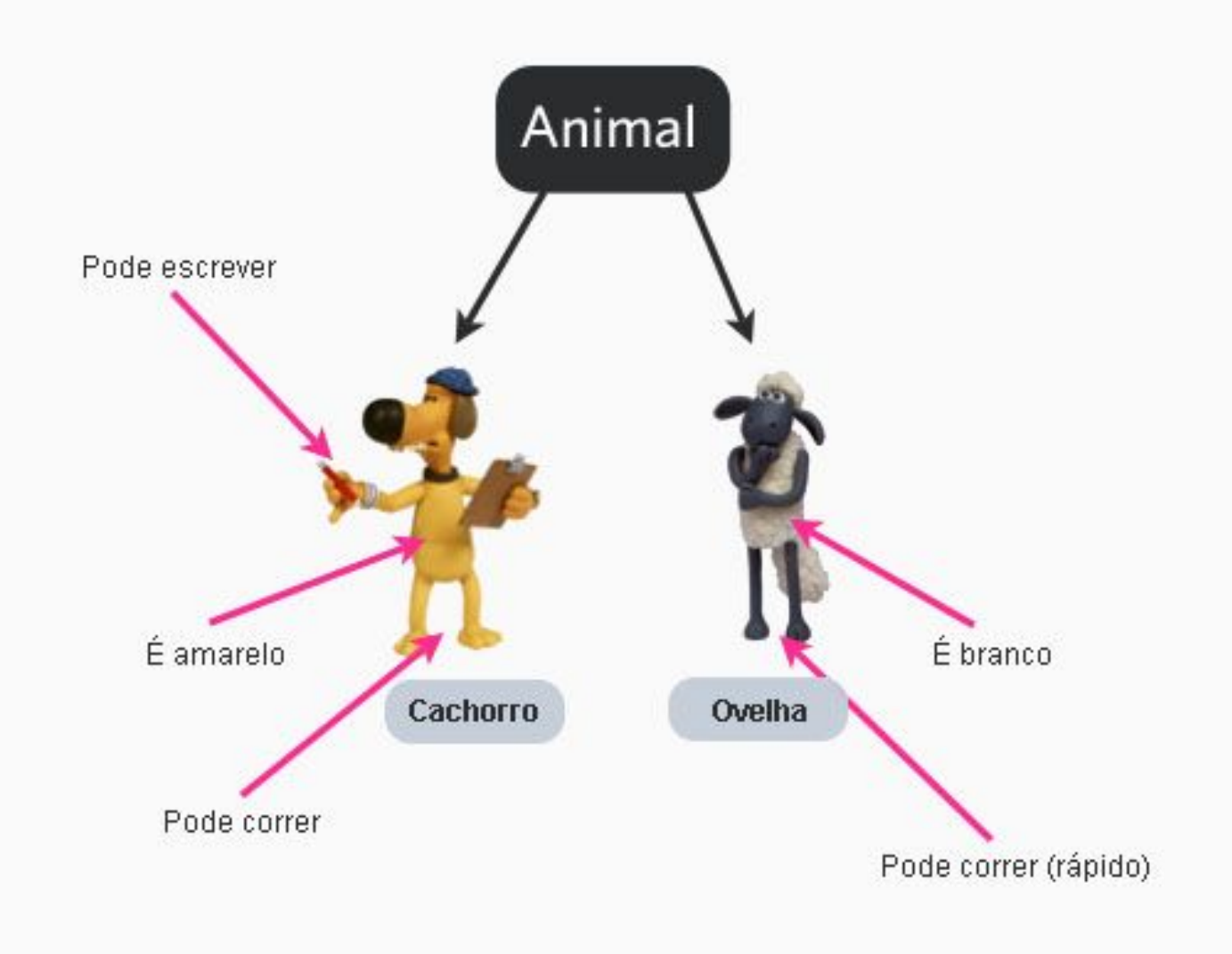
Programação Orientada a Objetos (POO)

- É necessário apresentar ao computador o funcionamento do nosso mundo...
- Para isso vamos explorar vários conceitos:
  - Classe
  - Objeto
  - Atributo
  - Método
  - Herança
  - Polimorfismo

Programação Orientada a Objetos (POO)

Programação Orientada a Objetos (POO)

Programação Orientada a Objetos (POO)



Programação Orientada a Objetos (POO)

- Como modelar um sistema bancário?

Programação Orientada a Objetos (POO)

- Como modelar um sistema bancário?
  - Temos que representar: **Clientes, Agências, Contas, Operações, Extratos, etc.**

Programação Orientada a Objetos (POO)

- Como modelar um sistema biológico?

Programação Orientada a Objetos (POO)

- Como modelar um sistema biológico?
  - Temos que representar: **Pacientes, Vírus e Conexões**. Em cada instante de **Tempo**, um **Vírus** pode infectar um **Paciente**.

Programação Orientada a Objetos (POO)

- Onde quer que você olhe no mundo real, você vê objetos
  - Pessoas, animais, plantas, carros, etc.

- Humanos pensam em termos de objetos
  - Orientação a objetos é alto nível i.e., mais próximo dos humanos que dos computadores

PE *versus* POO

- Programação Estruturada (PE)
  - Procedimentos implementados em blocos
  - Comunicação pela passagem de dados
  - Execução ➜ Acionamento de procedimentos
- Programação Orientada a Objetos (POO)
  - Dados e procedimentos encapsulados
  - Composto por diversos objetos
  - Execução ➜ Interação/Comunicação entre objetos

PE *versus* POO

- Programação Estruturada (PE)
  - Dados acessados via funções
  - Representação de tipos complexos
- Programação Orientada a Objetos (POO)
  - Dados são dotados de certa inteligência
  - Sabem realizar operações sobre si mesmos
  - É preciso conhecer a implementação?

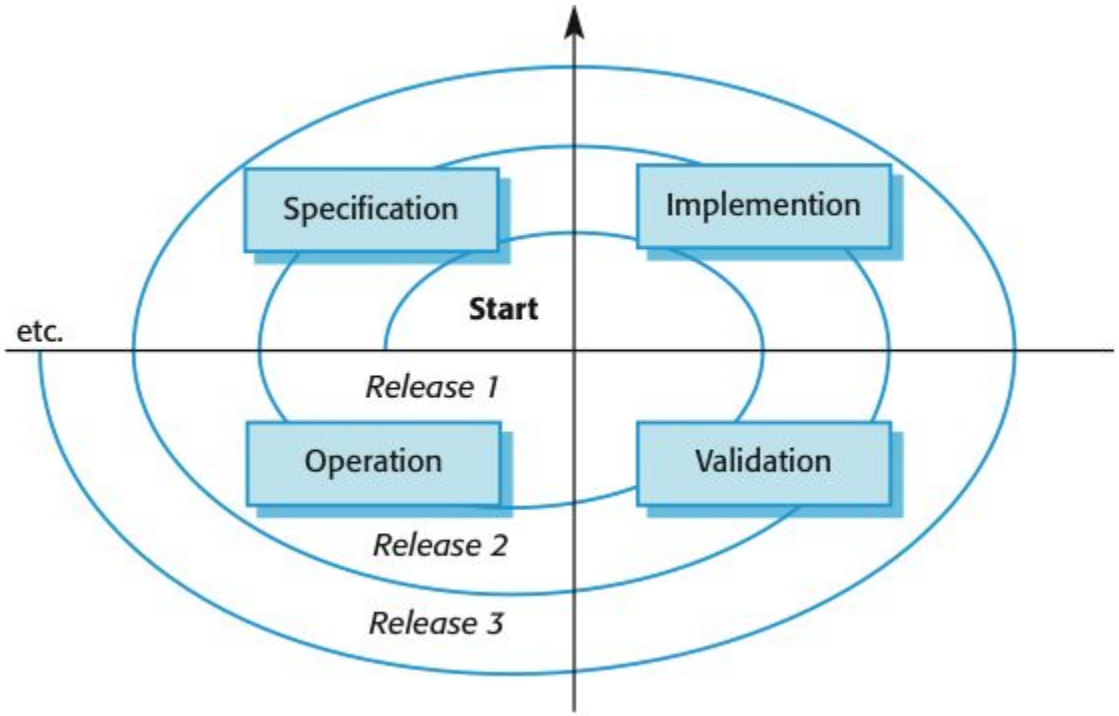
POO - Benefícios

- Maior confiabilidade
- Maior reaproveitamento de código
- Facilidade de manutenção
- Melhor gerenciamento
- Maior robustez
- ...

INF112 e o Desenvolvimento de Software...

Programação é uma atividade social (acreditem ou não)

- INF112 é um passo importante para aprendermos a desenvolver software



C++ Através de Exemplos

Olá Mundo!

```cpp
#include <iostream>

int main() {
  std::cout << "Hello World!" << std::endl;
  return 0;
}
```

Olá Mundo!

- Um programa C++ parece com C
- Porém C++ **não é** C
  - São compatíveis

```
#include <iostream>

int main() {
  std::cout << "Hello World!" << std::endl;
  return 0;
}
```

Incluir biblioteca externa (.h)

Procedimento main

"Atalho" para '\n'

Stream da saída padrão

Compilando

- Usamos o g++
  - Similar ao gcc

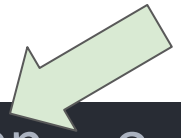
```
$ g++ hello.cpp -o hello
```

- Saída do programa

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

Nesta matéria

- Vale utilizar C++11/14
  - Favor não usem C++17



```
$ g++ -std=c++14 -Wall hello.cpp -o hello
```

- É comum usar a extensão .cpp

Padrões de C++

Pequeno histórico

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

Estamos aqui

Padrão 2017.

Usando Tipos e STDIN/OUT

```cpp
#include <iomanip>
#include <iostream>
#include <cmath>

using namespace std;

int main() {
  double pi = 3.1415;
  cout << "Olá DPI :) ";
  cout << "O valor de pi é? ";
  cout << pi;
  cout << endl;
  cout << "E se eu quiser uma precisão menor? ";
  cout << setprecision(1) << pi;
  cout << endl;
  cout << "Pi ao quadrado com 7 precisão: " << setprecision(7) << pow(pi, 2);
  return 0;
}
```

Usando Tipos e STDIN/OUT

```cpp
#include <iomanip>
#include <iostream>
#include <cmath>

using namespace std;

int main() {
  double pi = 3.1415;
  cout << "Olá DPI :) ";
  cout << "O valor de pi é? ";
  cout << pi;
  cout << endl;
  cout << "E se eu quiser uma precisão menor? ";
  cout << setprecision(1) << pi;
  cout << endl;
  cout << "Pi ao quadrado com 7 precisão: " << setprecision(7) << pow(pi, 2);
  return 0;
}
```

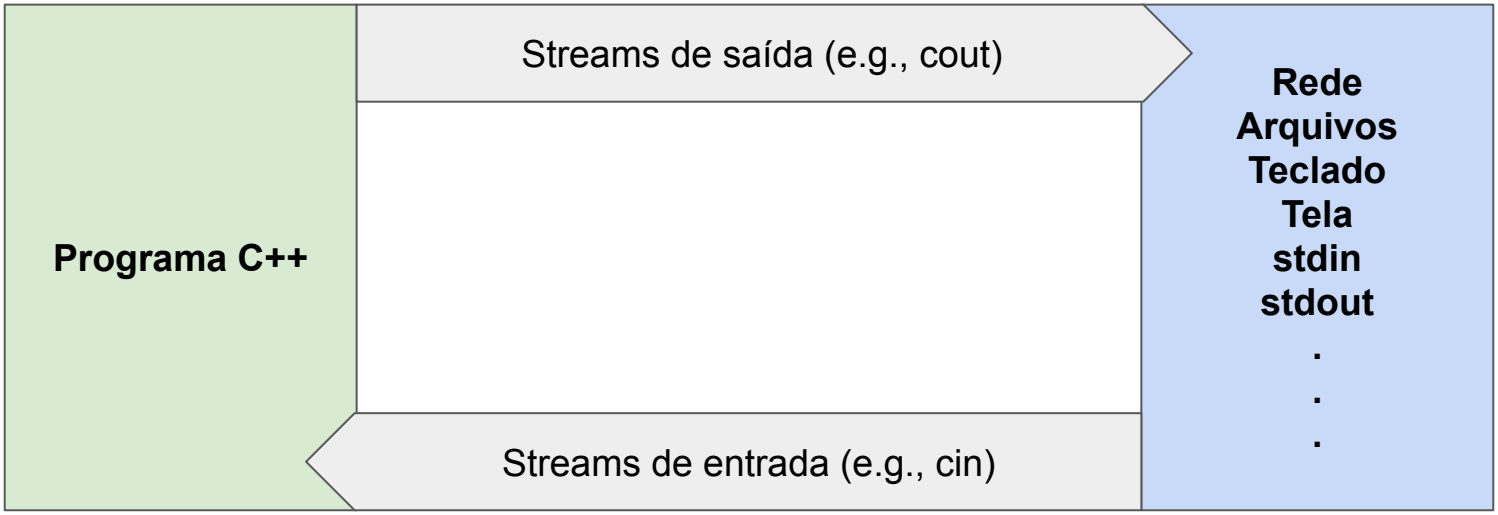
Manipulação de ES

Matemática

Note como o resultado de pow passa pelo filtro setprecision

Streams

- Streams são utilizados para comunicação
- Podemos usar `printf` também



Usando funções c

- C++ consegue fazer uso de C
- Vamos tentar manter o curso 100% C++

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

int main() {
  double pi = 3.1415;
  printf("Olá DPI :)\n");
  printf("O valor de pi é? %.2f", pi);
  printf("O valor de pi ao quadrado é? %.2f", pow(pi, 2));
  return 0;
}
```

Usando STDIN

- Com o cin vamos ler do teclado >>

```
#include <iostream>

int main()  {
  double num1 = 0.0;
  double num2 = 0.0;
  std::cout << "Digite o primeiro número: ";
  std::cin >> num1;
  std::cout << "Digite o segundo número: ";
  std::cin >> num2;
  std::cout << "A divisão de " << num1 << " e " << num2 << " é " \
     << num1/num2 << ".\n";
  return 0;
}
```

Note o std::, sem namespace

Operadores em C++

Na maioria dos casos, a semântica de C se mantém. Porém...

- Assim como em C, usamos operadores para atuar nos dados:
  `+, -, /, *, >>, >, <`

- Porém, o sentido pode mudar dependendo do tipo. Para números `>>` é shift, para streams é saída.

Streams em arquivos

Pouca mudança

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

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)) {
    out << line;
  }
  in.close();
  out.close();
}
```

Strings

Finalmente! Vamos esquecer o '\0' por um tempo

- C++ tem suporte nativo para strings

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

int main() {
  std::string hello("Olá mundo!\n");
  std::string inf112("Vamos iniciar INF112\n");
  std::cout << hello;
  std::cout << std::endl;
  std::cout << inf112;

  std::string maisuma = "Mais uma!";
  std::cout << maisuma.size();
  std::cout << std::endl;
  return 0;
}
```

Diferentes formas de declarar

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

int main() {
  std::string hello1("Olá mundo!\n");
  std::string hello2 = "Olá mundo!\n";
}
```

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

using namespace std;

int main() {
  string hello1("Olá mundo!\n");
  string hello2 = "Olá mundo!\n";
}
```

Strings

- Suporte nativo ajuda bastante
- Métodos como: .size
  - Tamanho da string
- Note a diferença:
  - `str.size() vs strlen(str)`
- Vamos explorar isso durante a disciplina

Comparando Strings

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

int main() {
  std::string hello("Olá mundo!\n");
  std::string hello2("Olá mundo!\n");
  if (hello == hello2) {
    std::cout << "c++ faz overload do == para strings!!!!.\n";
  }
  if (hello.compare(hello2) == 0) {
    std::cout << "Strings iguais.\n";
  }
  return 0;
}
```

Note o overload do operador ==. Comodidade.

Mesma coisa de antes

Vetores

```
#include <iostream>

int main()  {
  int n = 0.0;
  std::cout << "Digite o número de elementos: ";
  std::cin >> n;

  int dados[n];
  for (int i = 0; i < n; i++) {
    std::cout << "Digite o " << i+1 << "-ésimo número: ";
    std::cin >> dados[i];
  }

  int soma = 0;
  for (int i = 0; i < n; i++) {
    soma += dados[i];
  }
  std::cout << "A soma foi: " << soma << std::endl;
}
```

Saída

- Sem muita surpresa
- Comandos de repetição estilo C
- Porém, podemos incrementar.

```
$ g++ -Wall -std=c++14 acumulador.cpp -o acumulador
$ ./acumulador
Digite o número de elementos: 3
Digite o 1-ésimo número: 2
Digite o 2-ésimo número: 1
Digite o 3-ésimo número: 6
A soma foi: 9
```

Vectors

```
#include <iostream>
#include <vector>

int main()  {
  std::vector<int> dados = {};
  int v = 0;
  int i = 0;
  while (v >= 0) {
    std::cout << "Digite o " << i+1 << "-ésimo número (-1 para terminar): ";
    std::cin >> v;
    if (v < 0) break;
    dados.push_back(v);
  }

  for (int& x : dados)
    x *= 2;
  for (int x : dados)
    std::cout << x << std::endl;
}
```

Vetor redimensionável, uma lista com array por baixo.

Nova forma de iterar

Nova saída

- Duplicamos o valor dos elementos
- Como?

```
$ g++ -Wall -std=c++14 acumulador2.cpp -o acumulador2
$ ./acumulador2
Digite o 1-ésimo número (-1 para terminar): 3
Digite o 1-ésimo número (-1 para terminar): 4
Digite o 1-ésimo número (-1 para terminar): 7
Digite o 1-ésimo número (-1 para terminar): 8
Digite o 1-ésimo número (-1 para terminar): -1
6
8
14
16
```

Qual a saída em cada caso?

- Laço clássico

```
std::vector<int> dados = {0, 7, 8, 1, 3};
for (int i = 0; i < dados.size(); i++)
  std::cout << dados[i];
```

- Laço compacto

```
for (int x : dados)
  std::cout << x;
```

- Laço para a referência

```
for (int &x : dados)
  x *= 2;
```

Exemplo &

https://goo.gl/MXw83D

```
#include <iostream>

int& function(int& f) {
  f=f+3;
  return f;
}

int main() {
  int x = 7;
  int y;
  y = function(x);
  std::cout << "Input: " << x << std::endl;
  std::cout << "Output:" << y << std::endl;
  x++;
  y--;
  std::cout << "X: " << x << std::endl;
  std::cout << "Y:" << y << std::endl;
  return 0;
}
```

Até agora

Apresentação da disciplina + apresentação inicial da linguagem

- Todo o curso vai ser focado em exemplos
- Vamos explorar melhor os conceitos
  - Exemplos de hoje são motivadores iniciais

- Não é um curso de linguagem!
  - Não podemos focar nos detalhes de C++
  - C++ é uma ferramenta para nosso curso

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.

Por fim

- Criar conta no github
- Configurar um ambiente C++

- Fique à vontade para escolher uma IDE.

  Sugiro:
  - Visual code studio
  - Linux subsystem for windows
  - Tutoriais de configuração em breve