

Introdução ao Curso

Programação e Desenvolvimento de Software II

- Programação e Desenvolvimento de Software I
 - Conceitos básicos de programação
 - **for, if, while, else**
 - Definições de Funções
 - Recursividade
 - **Maior foco em conceitos base de como manipular o computador**

- Programação e Desenvolvimento de Software I
 - Conceitos básicos de programação
 - **for, if, while, else**
 - Definições de Funções
 - Recursividade
 - **Maior foco em conceitos base de como manipular o computador**
- Programação e Desenvolvimento de Software II
 - Programação Orientada a Objetos
 - Uso e entendimento de Estruturas de Dados
 - Boas práticas de programação
 - Ferramentas
 - **Maior foco em como desenvolver bons programas**

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

- Entender o problema
 - Qual o problema do meu cliente?
 - Qual o resultado esperado?
- Modelar os dados
- Codificar a solução

- Entender o problema
 - Qual o problema do meu cliente?
 - Qual o resultado esperado?
- Modelar os dados
 - Como abstrair as partes do sistema?
 - Como que as diferentes partes se ligam?
- Codificar a solução

- Entender o problema
 - Qual o problema do meu cliente?
 - Qual o resultado esperado?
- Modelar os dados
 - Como abstrair as partes do sistema?
 - Como que as diferentes partes se ligam?
- Codificar a solução
 - Quais partes serão classes quais serão interfaces?
 - Como testar o sistema?
 - Corretude.

- São desenvolvidos através de abstrações
 - **Sistemas Operacionais:** interface consistente em cima de um hardware

- São desenvolvidos através de abstrações
 - **Sistemas Operacionais:** interface consistente em cima de um hardware
 - **Redes:** comunicação confiável em cima de meios de transporte não confiáveis

- São desenvolvidos através de abstrações
 - **Sistemas Operacionais:** interface consistente em cima de um hardware
 - **Redes:** comunicação confiável em cima de meios de transporte não confiáveis
 - **Databases:** uma interface declarativa (SQL) para acesso de dados

- São desenvolvidos através de abstrações
 - **Sistemas Operacionais:** interface consistente em cima de um hardware
 - **Redes:** comunicação confiável em cima de meios de transporte não confiáveis
 - **Databases:** uma interface declarativa (SQL) para acesso de dados
 - **Sistemas Distribuídos:** uma interface de acesso para um conjunto de máquina em rede

- São desenvolvidos através de abstrações
 - **Sistemas Operacionais:** interface consistente em cima de um hardware
 - **Redes:** comunicação confiável em cima de meios de transporte não confiáveis
 - **Databases:** uma interface declarativa (SQL) para acesso de dados
 - **Sistemas Distribuídos:** uma interface de acesso para um conjunto de máquina em rede
- Qual o fator comum?

- São desenvolvidos através de abstrações
 - **Sistemas Operacionais:** interface consistente em cima de um hardware
 - **Redes:** comunicação confiável em cima de meios de transporte não confiáveis
 - **Databases:** uma interface declarativa (SQL) para acesso de dados
 - **Sistemas Distribuídos:** uma interface de acesso para um conjunto de máquina em rede
- Qual o fator comum?
Esconder a complexidade. Abstração. Abstração. Abstração.

- São desenvolvidos através de abstrações
 - **Sistemas Operacionais:** interface consistente em cima de um hardware
 - **Redes:** comunicação confiável em cima de meios de transporte não confiáveis
 - **Databases:** uma interface declarativa (SQL) para acesso de dados
 - **Sistemas Distribuídos:** uma interface de acesso para um conjunto de máquina em rede
- Qual o fator comum?
Esconder a complexidade. Abstração. Abstração. Abstração.
- Vamos ver uma formas principal de fazer isto:
 - **Orientação a Objetos**
 - **O curso ainda é focado em código!**

- Foca no essencial para passar a mensagem
- Aqui, vamos focar no essencial para resolver um problema

Abstração

- Foca no essencial para passar a mensagem
- Aqui, vamos focar no essencial para resolver um problema



vs.

Abstração

- Foca no essencial para passar a mensagem
- Aqui, vamos focar no essencial para resolver um problema



vs.



- Foca no essencial para passar a mensagem
- Aqui, vamos focar no essencial para resolver um problema
- Os dois representam a mesma coisa. **O mais simples pode ser suficiente!**



vs.



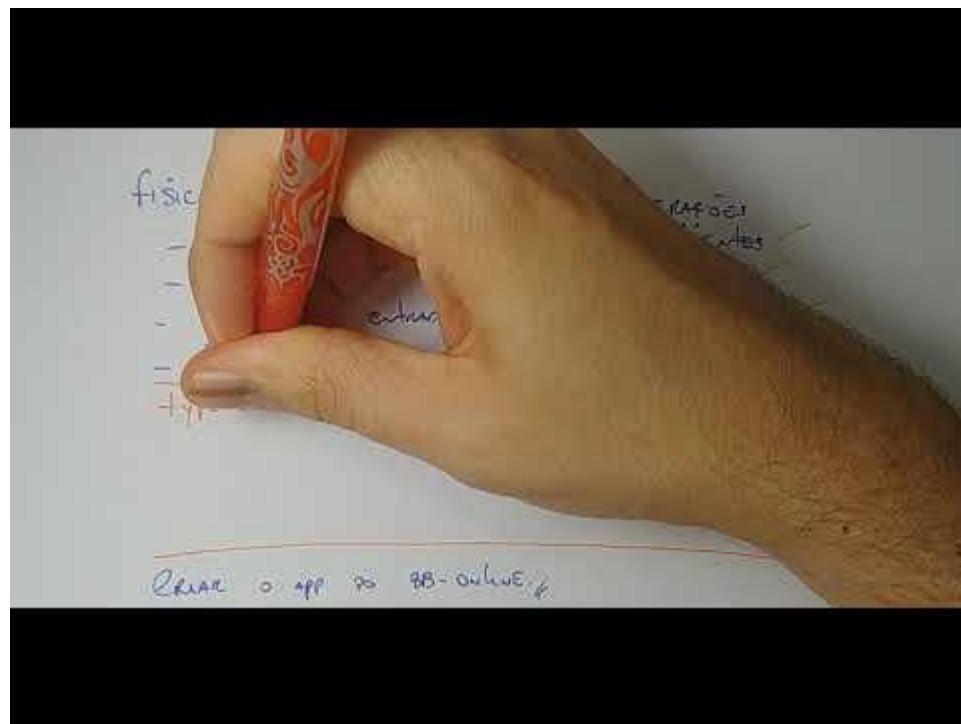
Flavio Vinicius Diniz de Figueiredo

flavio@dcc.ufmg.br

Programação e Desenvolvimento de Software II

Modelando um Sistema Bancário

Programação e Desenvolvimento de Software II



Flavio Vinicius Diniz de Figueiredo

flavio@dcc.ufmg.br

Programação e Desenvolvimento de Software II

Introdução ao Curso - Objetivos e Ementa

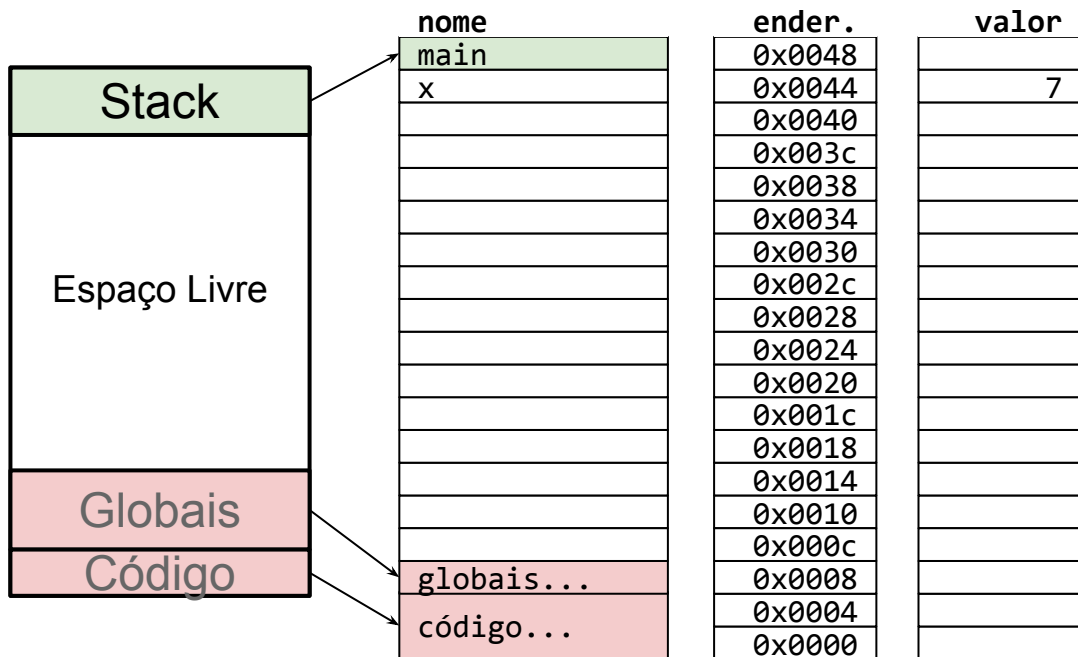
Programação e Desenvolvimento de Software II

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**.

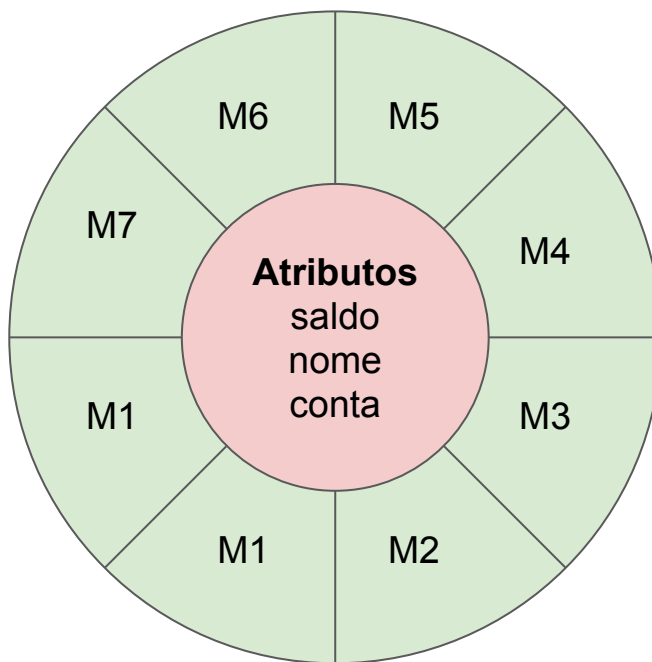
- Desenvolvimento de software
- Entendimento da memória
- Programação orientada a objetos
- Uso e aplicação de estruturas de dados
- Boas práticas
 - Testes
 - Programação defensiva

- Como desenvolver um sistema de banco? (Vídeo Anterior)
 - Clientes
 - Transações
 - Contas

- Onde que moras as variáveis do meu programa?
- Onde que mora a memória alocada dinamicamente?



- Cada entidade do exemplo anterior pode virar um **objeto**. Será que deve?



- Quando é o melhor momento de usar um mapa/dicionário? Uma lista?
- Qual a intuição por trás de tais estruturas?
 - Não vamos implementar estruturas complexas do zero!
 - Apenas listas encadeadas
 - Este é um assunto de ED

1120217	Nikhilesh
1120236	Navneet
1120250	Vikas
1120255	Doodrah

Keys

values



- Como testar programas? Como garantir uma execução correta?
- Depuração



Flavio Vinicius Diniz de Figueiredo

flavio@dcc.ufmg.br

Programação e Desenvolvimento de Software II

C++; **Alguns exemplos**

(este conjunto de slides não é um curso C++!)

Programação e Desenvolvimento de Software II


```
#include <iostream>

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

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

```
#include <iostream>
```

Incluir biblioteca externa (.h)

```
int main() {
```

Procedimento main

```
    std::cout << "Hello World!" << std::endl;
```

“Atalho” para ‘\n’

```
    return 0;
```

Stream da saída padrão

```
}
```

Compilando

USUÁRIOS WINDOWS, TEREMOS UM VÍDEO
EXPLICANDO COMO CONFIGURAR O
AMBIENTE!! NÃO SE PREOCUPEM!

- Usamos **g++**. Para C usamos o gcc.

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

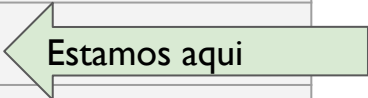
- Executamos da mesma forma de um programa C

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

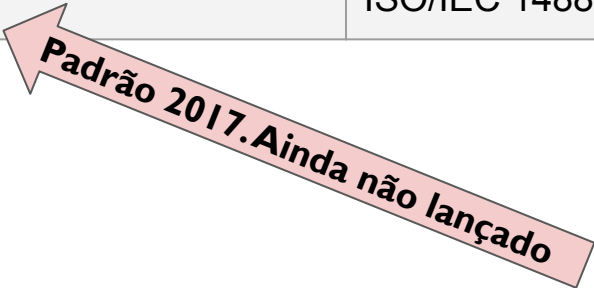
- Vale utilizar C++11/14
 - Favor não usem C++17/20
- É comum usar a extensão .cpp

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

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

A green arrow with a black outline points from the right towards the "C++14" entry in the table. The text "Estamos aqui" is written inside the arrow's body.

Estamos aqui

A pink arrow with a black outline points from the bottom-left towards the "C++20" entry in the table. The text "Padrão 2017. Ainda não lançado" is written inside the arrow's body.

Padrão 2017. Ainda não lançado

- Vale utilizar C++11/14
 - Favor não usar C++17/20
- É comum usar a extensão .cpp

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

COMPILADOR

USE C++14

INDIQUE
ERROS

NESTE ARQUIVO

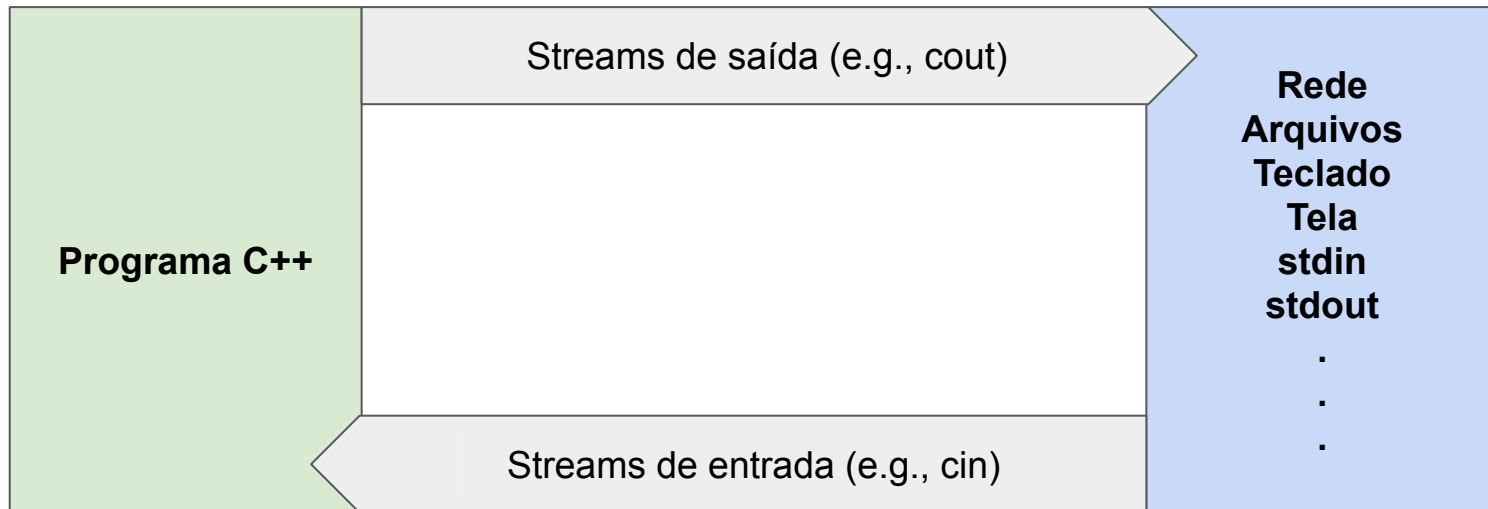
GERE

ESTE
PROGRAMA

Streams → std::cout vs printf

```
#include <iostream>

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



- C++ Suporta Strings

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

int main() {
    std::string ola = std::string("Olá!\n");
    std::string pds2("Vamos iniciar PDS2\n");
    std::cout << ola;
    std::cout << std::endl;
    std::cout << pds2;

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



Strings

- C++ Suporta Strings
- Várias formas de declarar

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

int main() {
    std::string ola = std::string("Olá!\n");
    std::string pds2("Vamos iniciar PDS2\n");
    std::cout << ola;
    std::cout << std::endl;
    std::cout << pds2;

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




Strings

- C++ Suporta Strings
- Várias formas de declarar
- Isto se chama de construir um objeto string.

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

int main() {
    std::string ola = std::string("Olá!\n");
    std::string pds2("Vamos iniciar PDS2\n");
    std::cout << ola;
    std::cout << std::endl;
    std::cout << pds2;

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



Forma 1

Strings

- C++ Suporta Strings
- Várias formas de declarar
- Isto se chama de construir um objeto string.
- A mesma coisa

```
#include <iostream>
```

```
#include <string>
```

```
int main() {
```

```
    std::string ola = std::string("Olá!");
```

```
    std::string pds2("Vamos iniciar PDS2\n");
```

```
    std::cout << ola;
```

```
    std::cout << std::endl;
```

```
    std::cout << pds2;
```

```
    std::string maisuma = "Mais uma!";
```

```
    std::cout << maisuma.size();
```

```
    std::cout << std::endl;
```

```
    return 0;
```

```
}
```


Forma 2

- C++ Suporta Strings
- Várias formas de declarar
- Isto se chama de construir um objeto string.
- A mesma coisa
- Mesma coisa novamente

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

int main() {
    std::string ola = std::string("Olá!\n");
    std::string pds2("Vamos iniciar PDS2\n");
    std::cout << ola;
    std::cout << std::endl;
    std::cout << pds2;

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




- C++ Suporta Strings
- Várias formas de declarar
- Isto se chama de construir um objeto string.
- A mesma coisa
- Mesma coisa novamente
- No curso vamos entender as diferenças

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

int main() {
    std::string ola = std::string("Olá!\n");
    std::string pds2("Vamos iniciar PDS2\n");
    std::cout << ola;
    std::cout << std::endl;
    std::cout << pds2;

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



C vs C++

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

C

```
int main() {  
    char s0[4] = {'d', 'c', 'c', '\0'};  
    char s1[] = "dcc";  
    printf("%s\n", s0);  
    printf("%s\n", s1);  
}
```

```
#include <iostream>
```

C++

```
#include <string>
```

```
int main() {  
    std::string string = "dcc";  
    std::cout << string.size();  
    std::cout << std::endl;  
    return 0;  
}
```

Famoso ‘\0’ na string em C. C trata strings como vetores com \0 no fim.

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

C

```
int main() {  
    char s0[4] = {'d', 'c', 'c', '\0'};  
    char s1[] = "dcc";  
    printf("%s\n", s0);  
    printf("%s\n", s1);  
}
```



```
#include <iostream>
```

C++

```
#include <string>
```

```
int main() {  
    std::string string = "dcc";  
    std::cout << string.size();  
    std::cout << std::endl;  
    return 0;  
}
```

C++ tem um permite a criação **OBJETOS** (tipos por hora) string

```
#include <stdio.h>

int main() {
    char s0[4] = {'d', 'c', 'c', '\0'};
    char s1[] = "dcc";
    printf("%s\n", s0);
    printf("%s\n", s1);
}
```

C

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

int main() {
    std::string string = "dcc";
    std::cout << string.size();
    std::cout << std::endl;
    return 0;
}
```

C++



Basta fazer o import

```
#include <stdio.h>

int main() {
    char s0[4] = {'d', 'c', 'c', '\0'};
    char s1[] = "dcc";
    printf("%s\n", s0);
    printf("%s\n", s1);
}
```

C

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

int main() {
    std::string string = "dcc";
    std::cout << string.size();
    std::cout << std::endl;
    return 0;
}
```

C++

Falamos com o tipo usando .

```
#include <stdio.h>

int main() {
    char s0[4] = {'d', 'c', 'c', '\0'};
    char s1[] = "dcc";
    printf("%s\n", s0);
    printf("%s\n", s1);
}
```



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



```
int main() {
    std::string string = "dcc";
    std::cout << string.size();
    std::cout << std::endl;
    return 0;
}
```



Flavio Vinicius Diniz de Figueiredo

flavio@dcc.ufmg.br

Programação e Desenvolvimento de Software II