

DCC004 - Algoritmos e Estruturas de Dados II

Renato Martins

Email: renato.martins@dcc.ufmg.br

<https://www.dcc.ufmg.br/~renato.martins/courses/DCC004>

Material adaptado de PDS2 - Douglas Macharet e Flávio Figueiredo

Pessoal

Renato J. Martins

- Contato
 - Prédio do ICEX (ao lado da diretoria) sala 3009B
 - <http://www.dcc.ufmg.br/~renato.martins/courses/DCC004>
 - Email: renato.martins@dcc.ufmg.br
- Conteúdo da disciplina:
 - Seguindo a ementa de DCC204 - Programação e Desenvolvimento de Software (PDS2)
 - Prof. Douglas Macharet (DCC/UFMG) e Prof. Flavio Figueiredo (DCC/UFMG)
- Dúvidas?
 - Qualquer horário assumindo disponibilidade
 - Pessoalmente/Email

Sobre a matéria

AEDS2 → Maior foco em algoritmos

Implementação de Estruturas de Dados

Ordenação

Maior foco em algoritmos

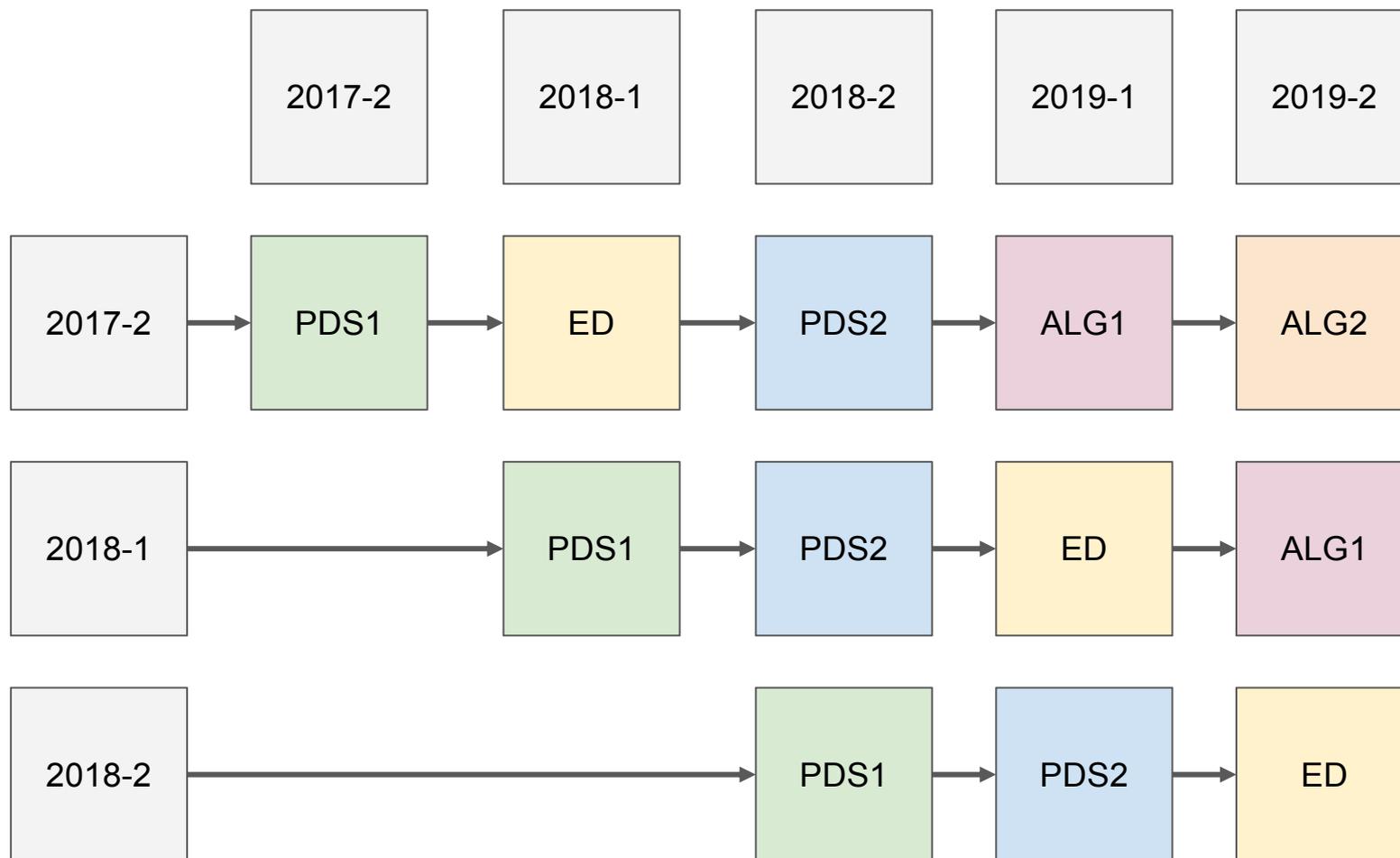
PDS2 → Maior foco em desenvolvimento

Programação Orientada a Objetos

Uso de Estruturas de Dados

Maior foco em desenvolvimento

Mudança Curricular



Programa da Disciplina

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
 - Testes
 - Programação defensiva

Avaliação

- 2 provas
 - Cada uma valendo 20 pontos
- 3 listas de exercícios/trabalhos
 - Cada uma valendo 10 pontos
- 1 Projeto
 - Valendo 30 pontos

Projeto Prático

- Tema da sua escolha
- Temas possíveis: rendering de imagens, realidade aumentada
- Individual (ou max 2 alunos)

Objetivos da Disciplina

Saber programar é apenas o passo inicial

- Em pouco tempo conseguimos:
 - if, while, else, for, funções
- Como modelar um programa?
- Como representar um conceito?

Objetivos da Disciplina

Exemplos do mundo real

- Como desenvolver um sistema de banco?

Objetivos da Disciplina

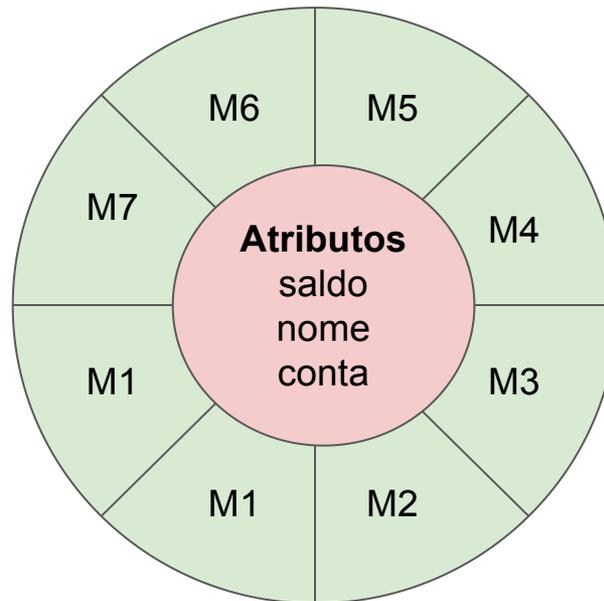
Exemplos do mundo real

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

Programação Orientada a Objetos

Uma das formas de modelar o mundo

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



Uso e Aplicação de Estruturas de Dados

A forma que você representa os dados guia seu programa

- Quando é o melhor momento de usar um mapa/dicionário?
- Uma lista?

Uso e Aplicação de Estruturas de Dados

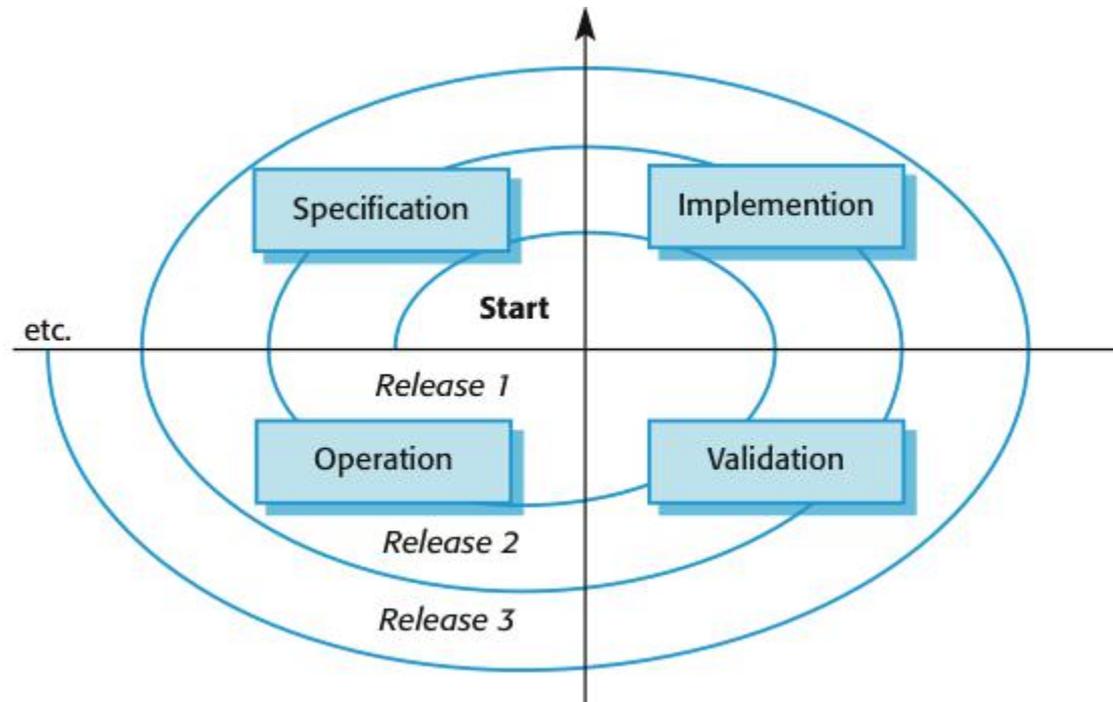
A forma que você representa os dados guia seu programa

- Antigamente AEDS2 focava na implementação de tais estruturas
- Assunto passou para ED
- Vamos prover uma visão top-down

Boas práticas

Programação é uma atividade social
(acreditem ou não)

- PDS2 é o passo 0 para desenvolver software



C++ A través de Ejemplos

Olá Mundo!

```
#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;
}
```

The diagram shows a code block with several green arrows pointing to specific parts of the code:

- An arrow points from the text "Incluir biblioteca externa (.h)" to the `<iostream>` part of the `#include` line.
- An arrow points from the text "Procedimento main" to the `int main() {` part of the code.
- An arrow points from the text "Stream da saída padrão" to the `std::cout` part of the `std::cout << "Hello World!" << std::endl;` line.
- An arrow points from the text "Atalho para '\n'" to the `std::endl;` part of the same line.

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++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. Ainda não lançado

Usando Tipos e STDIN/OUT

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

using namespace std;

int main() {
    double pi = 3.1415;
    cout << "Olá DCC :) ";
    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

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

using namespace std;

int main() {
    double pi = 3.1415;
    cout << "Olá DCC :) ";
    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;
}
```

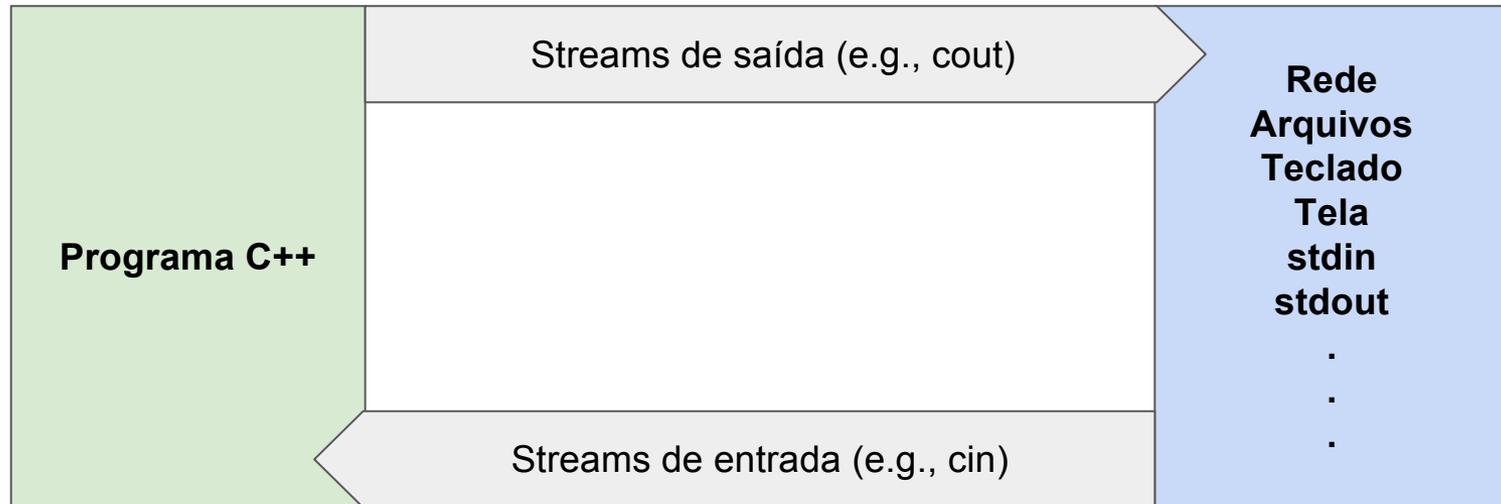
Manipulação de
ES

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++

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

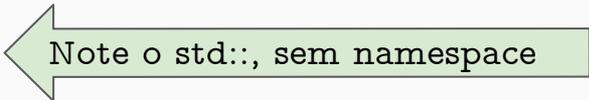
int main() {
    double pi = 3.1415;
    printf("Olá DCC :)\n");
    printf("O valor de pi é? %.2f", pi);
    printf("O valor de pi ao quadrado é? %.2f", pow(pi, 2));
    return 0;
}
```

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:
 $+$, $-$, $/$, $*$, \gg , $>$, $<$
- Porém, o sentido pode mudar dependendo do tipo. Para números \gg 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 pds2("Vamos iniciar PDS2\n");
    std::cout << hello;
    std::cout << std::endl;
    std::cout << pds2;

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

Diferentes formas de declarar

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

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

```
#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)`

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 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

Por fim

- Configurar um ambiente C++
- Escolher uma IDE
 - Qt-creator
 - Eclipse
 - Codeblocks
 - Visual studio