IMD0030
LINGUAGEM DE PROGRAMAÇÃO I

Aula 11 – Revisão, exercícios e esclarecimento de dúvidas

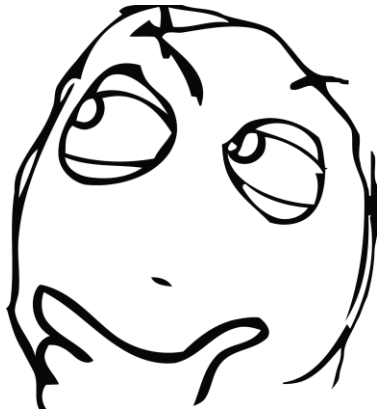
IMD INSTITUTO METRÓPOLE DIGITAL
UFRN UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE

Objetivos desta aula

- Revisar os principais conceitos de POO na linguagem C++, vistos até aqui
- Realizar exercícios para melhor fixação dos conceitos
- Esclarecer dúvidas sobre os conceitos e o Laboratório 2

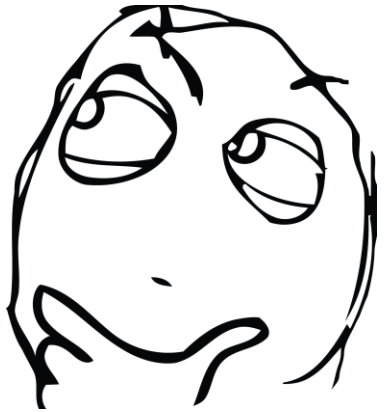
Definição da classe Dado

```
#ifndef _DADO_H_
#define _DADO_H_

#include <random>
#include <ostream>

class Dado {
  private:
    int valor;

  public:
    static std::random_device rd;
    static std::mt19937 gen;
    static std::uniform_int_distribution<> dis;

    Dado();
    Dado(int val);
    friend std::ostream& operator<< (std::ostream &o, Dado const &d);
    int jogar();
    int getValor();

    Dado operator+(const Dado &d) const;
    int operator+(const int &val) const;

    // Overloaded do typecast para int
    operator int();
};

#endif
```



Implementação da classe Dado (1)

```
#include <random>
#include <ostream>
#include <random>
#include "dado.h"

#include <iostream>


std::random_device Dado::rd{};
std::mt19937 Dado::gen(Dado::rd());
std::uniform_int_distribution<> Dado::dis(1,6);

Dado::Dado() {
    valor = std::round(dis(gen));
}

Dado::Dado(int val) {
    valor = val;
}
```



Implementação da classe Dado (2)

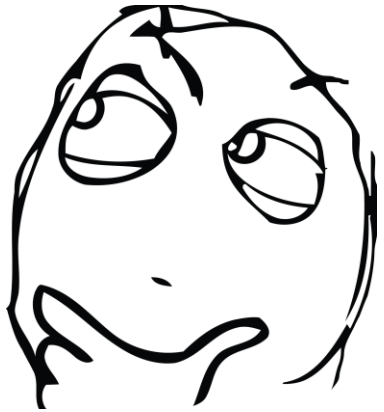
```
int
Dado::jogar() {
    valor = std::round(dis(gen));
    return valor;
}

int
Dado::getValor() {
    return valor;
}

std::ostream&
operator<< (std::ostream &o, Dado const &d) {
    o << d.valor;
    return o;
}
```

Implementação da classe Dado (3)

```
Dado
Dado::operator+(const Dado &d) const {
    return Dado(this->valor+d.valor);
}
```

```
int
Dado::operator+(const int &val) const {
    return (this->valor+val);
}
```

```
Dado::operator int() {
    return valor;
}
```



Testando a classe Dado

```
#include <sstream>
#include <iostream>
#include "dado.h"

int main(int argc, char const *argv[])
{
  Dado d1;
  Dado d2(5);
  Dado* d3 = new Dado();

  std::cout << "Valor d1: " << d1 << std::endl;
  std::cout << "Valor d2: " << d2 << std::endl;
  std::cout << "Valor d3: " << (*d3) << std::endl;

  int soma = d1+d2+(*d3);
  std::cout << "Soma: " << soma << std::endl;

  std::cout << "d1+5: " << d1+5 << std::endl;
  return 0;
}
```


IMD INSTITUTO METRÓPOLE DIGITAL
UFRN UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE

Programando com diferentes níveis de elegância...

```
#include <sstream>
#include <iostream>
#include "dado.h"

#include <vector>

int main(int argc, char const *argv[])
{
  // Totalmente C...
  Dado d[5];

  for(int i=0; i<5; ++i) {
    std::cout << d[i] << std::endl;
  }
```

Programando com diferentes níveis de elegância...

```
// Mistura mais proxima do C...
std::vector<Dado> dados(5);

for(int i=0; i<5; ++i) {
  std::cout << dados[i] << std::endl;
}

// Mistura mais proxima do C++...
std::vector<Dado*> muitosDados;

muitosDados.push_back(new Dado());
muitosDados.push_back(new Dado());
muitosDados.push_back(new Dado());

for (int i = 0; i < (int) muitosDados.size(); ++i)
  std::cout << (*muitosDados[i]) << std::endl;
```

Programando com diferentes níveis de elegância...

```
// Totalmente C++...
std::vector<Dado*> maisDados;

maisDados.push_back(new Dado());
maisDados.push_back(new Dado());
maisDados.push_back(new Dado());

// for (std::vector<Dado*>::iterator it = maisDados.begin(); it < maisDados.end(); ++it)
  std::cout << (**it) << std::endl;

  return 0;
}
```


IMD INSTITUTO METRÓPOLE DIGITAL UFRN UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE

Exercício

- Implemente em C++ as respectivas classes, atributos e métodos (incluindo construtores e destrutor) necessários para atender às seguintes abstrações:
  - Uma conta corrente possui uma agência, um número, um saldo, um status que informa se ela é especial ou não, um limite (no caso de conta especial) e um conjunto de movimentações (normalmente em grande número e variável entre contas).
  - Uma movimentação possui uma descrição, um valor e uma indicação se ela é uma movimentação de crédito ou débito.
- Usando as classes implementadas, escreva um programa em C++ para simular uma agência bancária que deve armazenar um conjunto de contas e permitir as seguintes operações básicas:
  - Criações de conta, exclusão de contas, saques (respeitando o saldo e o limite), depósitos, emissão de saldo e extrato, além de transferência entre contas.
  - Especificamente com relação à emissão de extrato, seu programa deverá exibir a lista de movimentações realizadas por meio da sobrecarga do operador de inserção em stream (<<).

Alguma Questão?