

Programação de Computadores

Programação Orientada a Objetos

Prof. Delano Beder

Roteiro 11 – Sobrecarga de operadores

1. Crie um projeto no Netbeans (Aplicação C/C++) denominado SobrecargaOperadores

1.1 Escolha: Criar arquivo principal main (C++)

2. Implementação da abstração Data

Crie uma Nova classe C++ denominada Data

Verifique se dois arquivos foram gerados: Data.h (Cabeçalho) e Data.cpp (Código-fonte)

2.1 Atualize o arquivo Data.h

```
#ifndef DATA_H
#define DATA_H

#include <iostream>
#include <iomanip>
using namespace std;

class Data {
public:

    // operadores relacionais

    bool operator>(const Data& right) const;
    bool operator>=(const Data& right) const;
    bool operator<(const Data& right) const;
    bool operator<=(const Data& right) const;
    bool operator==(const Data& right) const;
    bool operator!=(const Data& right) const;

    // funcao friend (impressão usando cout)
    friend ostream& operator<<(ostream& os, const Data& obj);

    // funcao friend (leitura usando cin)
    friend istream& operator>>(istream& is, Data& obj);

private:
    int compare(Data outra) const;
    int dia, mes, ano;
};

#endif /* DATA_H */
```

2.2 Atualize o arquivo denominado Data.cpp

```
#include "Data.h"

int Data::compare(Data outra) const {
    if (this->ano != outra.ano) {
        return this->ano - outra.ano;
    } else if (this->mes != outra.mes) {
        return this->mes - outra.mes;
    } else return this->dia - outra.dia;
}

bool Data::operator<(const Data& right) const {
    return compare(right) < 0;
}

bool Data::operator<=(const Data& right) const {
    return compare(right) <= 0;
}

bool Data::operator>(const Data& right) const {
    return compare(right) > 0;
}
```

```

bool Data::operator>=(const Data& right) const {
    return compare(right) >= 0;
}

bool Data::operator==(const Data& right) const {
    return compare(right) == 0;
}

bool Data::operator!=(const Data& right) const {
    return compare(right) != 0;
}

ostream& operator<<(ostream& os, const Data& obj) {
    os << setfill('0') << setw(2) << obj.dia << "/";
    os << setfill('0') << setw(2) << obj.mes << "/";
    os << setfill('0') << setw(4) << obj.ano;
    return os;
}

istream& operator>>(istream& is, Data& obj) {
    char c; // usado para ignorar a barra "/"
    is >> obj.dia >> c >> obj.mes >> c >> obj.ano;
    return is;
}

```

3. Atualize o arquivo main.cpp

```

#include "Data.h"

int main() {

    Data D1;
    Data D2;

    cout << "1a Data: ";
    cin >> D1;
    cout << "2a Data: ";
    cin >> D2;

    if (D1 > D2) {
        cout << D1 << " é mais recente que " << D2 << endl;
    } else if (D1 < D2) {
        cout << D2 << " é mais recente que " << D1 << endl;
    } else {
        cout << D1 << " é igual a " << D2 << endl;
    }

    return 0;
}

```

4. Compile e execute (verifique a saída impressa)

5. Fim