OOP laboratory_02

Terman Emil FAF161

September 22, 2017

PHD prof: M. Kulev

LaTeX

1 Subject

Constructor - initialization function

2 Objectives

- Studierea principiilor de definire şi utilizare a constructorilor
- Studierea principiilor de definire şi utilizare a destructorilor
- Studierea tipurilor de constructori

3 Task

a) Să se creeze clasa Date – dată cu cîmpurile: zi(1-28..31), lună(1-12), an (numere întregi). Să se definească constructorii; funcţiile membru de setare a zilei, lunii şi anului; funcţiile membru de returnare a zilei, lunii, anului; funcţiile de afişare: afişare tip "6 iunie 2004" şi afişare tip "6.06.2004". Funcţiile de setare a cîmpurilor clasei trebuie să verifice corectitudinea parametrilor primiţi.

b) Să se creeze clasa Matrix – matrice. Clasa conţine pointer spre int, numărul de rînduri şi de coloane şi o variabilă – codul erorii. Să se definească constructorul fără parametri (constructorul implicit), constructorul cu un parametru – matrice pătrată şi constructorul cu doi parametri – matrice dreptunghiulară ş. a. Să se definească funcţiile membru de acces: returnarea şi setarea valorii elementului (i, j). Să se definească funcţiile de adunare şi scădere a două matrice; înmulţirea unei matrice cu alta; înmulţirea unei matrice cu un număr. Să se testeze funcţionarea clasei. În caz de insuficienţă de memorie, necorespondenţă a dimensiunilor matricelor, depăşire a limitei memoriei utilizate să se stabilească codul erorii.

4 Main notions of theory and used methods

A class in C++ is a user defined type or data structure declared with keyword class that has data and functions (also called methods) as its members whose access is governed by the three access specifiers private, protected or public (by default access to members of a class is private).

5 Data analysis

5.1 Ex a

```hpp
#ifndef DATE_HPP
# define DATE_HPP

# include <iostream>
# include <string>
# include <stdexcept>

class   Date
{
public:

    //Exceptions
    class   InvalidDate : public std::exception {
    public:
        virtual const char * what() const throw();
    };

    enum        EMonth {
                    jan = 1, feb, mar,
                    apr, may, jun,
                    jul, aug, sept,
                    oct, nov, dec
    };

    //Getters
    int         getDay(void) const;
    int         getMonth(void) const;
    int         getYear(void) const;

    //Setters
    void        setDay(int day);
    void        setMonth(int month);
    void        setYear(int year);

    //Constr & destr
    Date(void);
    Date(int day, int month, int year);
    Date(Date const & target);
    ~Date(void);

    bool        isLeapYear(void) const;
    std::string toStrNamedMonth(void) const;
    std::string toStr(void) const;


    //Operators
    Date &      operator = (Date const & target);

private:
    static const std::string _monthNames[12];

    int         _day;
    int         _month;
    int         _year;

    int         _monthMaxDays(void) const;
};

std::ostream &  operator << (std::ostream & o, Date const & target);

#endif
```

5.2 Ex b

```
#ifndef MATRIX_HPP
# define MATRIX_HPP

# include <string>
# include <iostream>
# include <errno.h>

class Matrix
{
public:
    enum            EMatrixErrno {
                        boundsErr = 1,
                        invalidSize,
                        enomem = ENOMEM
    };

    mutable int     mErrno;

    int             getLines(void) const;
    int             getCols(void) const;

    Matrix(void);
    Matrix(int lines, int cols);
    Matrix(Matrix const & target);
    ~Matrix(void);

    //Utils
    void            assignAll(int value);

    //Operators
    Matrix &        operator = (Matrix const & target);

    int const *     operator [] (int i) const;
    int *           operator [] (int i);

    Matrix          operator + (Matrix const & target) const;
    Matrix          operator - (Matrix const & target) const;
    Matrix          operator * (Matrix const & target) const;
    Matrix          operator * (int nb) const;

private:
    int             **_tab;
    int             _lines;
    int             _cols;

    //Utils
    void            _delTab(void);
    int             _newTab(int lines, int cols);
};

std::ostream &      operator << (std::ostream & o, Matrix const & target);

#endif
```