****

**MINISTERUL EDUCAȚIEI ȘI CERCETĂRII AL REPUBLICII MOLDOVA**

**Universitatea Tehnică a Moldovei**

**Facultatea Calculatoare, Informatică şi Microelectronică**

**Departamentul Informatică şi Ingineria Sistemelor**

**Samarchin Maxim**

**Gr. MI-222**

**Raport**

**pentru lucrarea de laborator Nr.2**

***la cursul “Programarea orientată pe obiecte”***

***Tema:*** *: Clase și obiecte. Constructori și destructor pentru clasa*

Verificat:

**Kulev Mihail ,** *dr., conf. univ.*

Facultatea FCIM, UTM

**Chișinău – 2023**

**Scopul lucrării:**

Studierea principiilor de definire şi utilizare a constructorilor

Studierea principiilor de definire şi utilizare a destructorilor

Studierea tipurilor de constructori

Noţiuni de bază

Una din cele mai răspîndite erori de programare (în orice limbaj) constă în utilizarea obiectelor fără iniţializare anterioară, aşa precum nu toate limbajele asigură iniţializare automată. Desigur, poate fi definită funcţia de iniţializare şi de distrugere a obiectului:

Sarcina Nr.1:

Să se creeze clasa Time – timp cu cîmpurile: ore (0-23), minute (0-59), secunde (0-59). Să se definească constructorii, funcţiile membru de setare a timpului; de returnare a orei, minutelor şi secundelor; funcţiile de afişare conform şablonului: „ora 17 şi 18 minute şi 4 secunde” şi „5 p.m. 18 minute 4 secunde”. Funcţiile de setare a cîmpurilor clasei trebuie să verifice corectitudinea parametrilor primiţi.

#include <iostream>

#include <iomanip>

#include <stdexcept>

class Time {

private:

int hours;

int minutes;

int seconds;

public:

// Constructor implicit

Time() : hours(0), minutes(0), seconds(0) {}

// Constructor cu parametri

Time(int h, int m, int s) {

setTime(h, m, s);

}

// Funcție pentru setarea timpului

void setTime(int h, int m, int s) {

if (h >= 0 && h <= 23 && m >= 0 && m <= 59 && s >= 0 && s <= 59) {

hours = h;

minutes = m;

seconds = s;

} else {

throw std::invalid\_argument("Parametrii timpului sunt incorecți!");

}

}

// Funcții de returnare a orei, minutelor și secundelor

int getHours() const {

return hours;

}

int getMinutes() const {

return minutes;

}

int getSeconds() const {

return seconds;

}

// Funcție de afișare a timpului conform șablonului specificat

void displayTimeTemplate1() const {

std::cout << "Ora " << hours << " și " << minutes << " minute și " << seconds << " secunde\n";

}

void displayTimeTemplate2() const {

std::string period = (hours >= 12) ? "p.m." : "a.m.";

int displayHours = (hours > 12) ? hours - 12 : hours;

std::cout << displayHours << " " << period << " " << minutes << " minute " << seconds << " secunde\n";

}

};

int main() {

try {

Time time1; // Se folosește constructorul implicit

time1.setTime(17, 18, 4);

time1.displayTimeTemplate1(); // Afișează conform primului șablon

time1.displayTimeTemplate2(); // Afișează conform celui de-al doilea șablon

Time time2(15, 45, 30); // Se folosește constructorul cu parametri

time2.displayTimeTemplate1();

time2.displayTimeTemplate2();

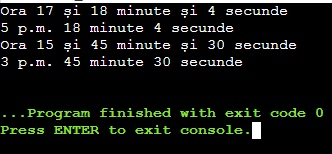
} catch (const std::invalid\_argument& e) {

std::cerr << "Eroare: " << e.what() << "\n";

}

return 0;

}



Sarcina Nr 2:

b) Să se creeze clasa Matrix – matrice. Clasa conţine pointer spre double, 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.

#include <iostream>

#include <vector>

using namespace std;

class Matrix {

private:

vector<vector<double>> data;

int rows;

int cols;

int error\_code; // Codul erorii

public:

// Constructor fără parametri

Matrix() : rows(0), cols(0), error\_code(0) {}

// Constructor cu un parametru pentru matrice pătrată

Matrix(int size) : rows(size), cols(size), error\_code(0) {

data.resize(size, vector<double>(size, 0.0));

}

// Constructor cu doi parametri pentru matrice dreptunghiulară

Matrix(int rows, int cols) : rows(rows), cols(cols), error\_code(0) {

data.resize(rows, vector<double>(cols, 0.0));

}

// Funcție membru de acces pentru setarea valorii elementului (i, j)

void set\_value(int i, int j, double value) {

if (i >= 0 && i < rows && j >= 0 && j < cols) {

data[i][j] = value;

} else {

error\_code = 1; // Codul erorii pentru depășirea limitelor matricei

}

}

// Funcție membru de acces pentru returnarea valorii elementului (i, j)

double get\_value(int i, int j) {

if (i >= 0 && i < rows && j >= 0 && j < cols) {

return data[i][j];

} else {

error\_code = 1; // Codul erorii pentru depășirea limitelor matricei

return -1; // sau orice altă valoare semnificativă pentru eroare

}

}

// Funcție pentru adunarea a două matrice

Matrix add(Matrix other) {

if (rows != other.rows || cols != other.cols) {

error\_code = 2; // Codul erorii pentru dimensiuni incompatibile

return \*this; // Întoarcem matricea curentă în caz de eroare

}

Matrix result(rows, cols);

for (int i = 0; i < rows; ++i) {

for (int j = 0; j < cols; ++j) {

result.data[i][j] = data[i][j] + other.data[i][j];

}

}

return result;

}

// Alte funcții pentru scădere, înmulțire între matrice și înmulțire cu un număr

// ...

// Funcție pentru afișarea matricei

void display() {

for (int i = 0; i < rows; ++i) {

for (int j = 0; j < cols; ++j) {

cout << data[i][j] << " ";

}

cout << endl;

}

}

};

int main() {

// Testare funcționare clasă Matrix

Matrix matrice\_patratica(3); // Matrice pătratică de dimensiune 3x3

matrice\_patratica.set\_value(0, 0, 1);

matrice\_patratica.set\_value(1, 1, 1);

matrice\_patratica.set\_value(2, 2, 1);

Matrix matrice\_rectangulara(3, 2); // Matrice rectangulară de dimensiune 3x2

matrice\_rectangulara.set\_value(0, 0, 2);

matrice\_rectangulara.set\_value(1, 1, 2);

matrice\_rectangulara.set\_value(2, 0, 3);

// Adunare matrice

Matrix rezultat\_adunare = matrice\_patratica.add(matrice\_rectangulara);

rezultat\_adunare.display(); // Afișare rezultat adunare

return 0;

}