****

**MINISTERUL EDUCAȚIEI, CULTURII ȘI CERCETĂRII**

**AL REPUBLICII MOLDOVA**

**Universitatea Tehnică a Moldovei**

**Facultatea Calculatoare, Informatică şi Microelectronică**

**Departamentul Ingineria Software și Automatică**

**Raport**

**pentru lucrarea de laborator Nr.3**

***la cursul de “Programarea orientata pe obiecte”***

Efectuat: Studentul gr. SI-191 **Pelin Bogdan**

Verificat:  *conf. Univ* **Mititelu Vitalii**

**Chișinău – 20****20**

***Tema:*** Constructorul – funcţie de iniţializare a obiectelor clasei

**Scopul lucrării:**

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

Varianta 7

а) Să se creeze clasa *Queue –* coadă. Cîmpurile – numărul de elemente şi un pointer pentru alocarea dinamică a memoriei. Să se definească constructorii: implicit, de copiere şi cu un parametru – numărul necesar de elemente; funcţiile *add* şi *get* pentru punerea unui element în coadă şi pentru scoaterea unui element din coadă respectiv; funcţiile: *isEmpty,* care returnează valoarea 1 dacă coada este vidă şi zero în caz contrar, şi *isFull* care returnează valoarea 1 dacă coada este plină şi zero în caz contrar.

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.

Codul programului(a):

#include <iostream>

using namespace std;

class Queue {

int \* init;

int num;

int first\_element;

int element;

int max\_elem;

public:

Queue();

Queue(const Queue & );

Queue(int num);

void create(int num);

bool isFull();

bool isEmpty();

void add(int new\_elem);

void get();

void display();

};

Queue::Queue(): init(NULL), num(0), first\_element(0), element(0), max\_elem(0) {}

void Queue::create(int num) {

init = new int[num];

max\_elem = num;

}

bool Queue::isFull() {

return (element >= max\_elem) ? true : false;

}

bool Queue::isEmpty() {

return (first\_element == element) ? true : false;

}

void Queue::add(int new\_elem) {

if (isFull()) {

cout << "Queue is full !!" << endl;

return;

}

init[element++] = new\_elem;

cout << "Success !!" << endl;

}

void Queue::get() {

if (isEmpty()) {

cout << "Queue is Empty !!" << endl;

return;

}

init[first\_element++];

max\_elem++;

cout << "Success !!" << endl;

}

void Queue::display() {

int i;

if (isEmpty()) {

cout << "Queue is empty" << endl;

return;

}

for (i = first\_element; i < element; i++)

cout << init[i] << " ";

cout << endl;

}

int main() {

Queue queue;

char menu\_choice;

int new\_elem;

int num;

while (1) {

system("cls");

cout << "Menu" << endl;

cout << "[1]. Create queue." << endl;

cout << "[2]. Add one element." << endl;

cout << "[3]. Get one element" << endl;

cout << "[4]. Show all elements" << endl;

cout << "[0]. Exit." << endl;

cout << "\n>>> ";

cin >> menu\_choice;

if (menu\_choice <= '4') {

if (menu\_choice == '0')

exit(0);

if (menu\_choice == '1') {

cout << "Input the length of queue !! >> " << endl;

cin >> num;

queue.create(num);

}

if (menu\_choice == '2') {

cout << "Input one element (integer) >>> ";

cin >> new\_elem;

queue.add(new\_elem);

}

if (menu\_choice == '3')

queue.get();

if (menu\_choice == '4')

queue.display();

} else

cout << "Bad choice" << endl;

system("pause");

}

return (0);

}

Codul programului(b):

#include <iostream>

using namespace std;

class Matrix {

public:

int Tab[50][50];

int line;

int column;

public:

Matrix();

Matrix(int);

Matrix(int, int);

Matrix(const Matrix & );

~Matrix();

void f\_set\_matr();

void f\_show();

void f\_addition(const Matrix & matrix);

void f\_subtract(const Matrix & matrix);

void f\_multipl\_matrix(const Matrix & matrix);

void f\_multipl\_num(int value);

};

int g\_size;

Matrix::Matrix(): line(0), column(0) {}

Matrix::Matrix(int line, int column): line(line), column(column) {

int i, j;

if ((line >= 0) && (column >= 0)) {

for (i = 0; i < line; i++)

for (j = 0; j < column; j++)

Tab[i][j] = 0;

} else

cout << "Error. Matrix are not positive !!" << endl;

}

Matrix::Matrix(const Matrix & Tab2): line(Tab2.line), column(Tab2.column) {

int i, j;

for (i = 0; i < line; i++)

for (j = 0; j < column; j++)

Tab[i][j] = Tab2.Tab[i][j];

}

Matrix::~Matrix() {

int i;

for (i = 0; i < line; ++i)

delete[] Tab;

line = 0;

column = 0;

}

void Matrix::f\_set\_matr() {

int i, j;

for (i = 0; i < line; i++)

for (j = 0; j < column; j++) {

cout << " Matrix[" << i << "][" << j << "]=";

cin >> Tab[i][j];

}

}

void Matrix::f\_show() {

int i, j;

for (i = 0; i < line; i++) {

for (j = 0; j < column; j++)

cout << "\t" << Tab[i][j] << "\t";

cout << endl;

}

}

void Matrix::f\_addition(const Matrix & matrix) {

Matrix matr;

int i, j;

if ((line == matrix.line) && (column == matrix.column)) {

for (i = 0; i < line; i++)

for (j = 0; j < column; j++)

Tab[i][j] += matrix.Tab[i][j];

} else

cout << "The dimensions do not match !!" << endl;

}

void Matrix::f\_subtract(const Matrix & matrix) {

int i, j;

if ((line == matrix.line) && (column == matrix.column)) {

for (i = 0; i < line; i++)

for (j = 0; j < column; j++)

Tab[i][j] -= matrix.Tab[i][j];

} else

cout << "The dimensions do not match !!" << endl;

}

void Matrix::f\_multipl\_matrix(const Matrix & matrix) {

int i, j, k;

int temp = 0;

int Tab2[50][50];

if (column == matrix.line) {

for (i = 0; i < line; i++) {

for (j = 0; j < column; j++) {

for (k = 0; k < line; k++) {

temp += Tab[i][k] \* matrix.Tab[k][j];

Tab2[i][j] = temp;

}

temp = 0;

}

temp = 0;

}

for (i = 0; i < line; i++)

for (j = 0; j < column; j++)

Tab[i][j] = Tab2[i][j];

} else

cout << "The matrix are not compatible for multiplying !!" << endl;

}

void Matrix::f\_multipl\_num(int value) {

int i, j;

for (i = 0; i < line; i++)

for (j = 0; j < column; j++)

Tab[i][j] \*= value;

}

void f\_menu() {

Matrix m1, m2;

int ml\_nr;

cout << "\tInput first matrix !!\n" << endl;

cout << "\*\*Lines = ";

cin >> m1.line;

cout << "\*\*Columns = ";

cin >> m1.column;

Matrix matrix1(m1.line, m1.column);

matrix1.f\_set\_matr();

system("pause");

system("cls");

cout << "\tInput second matrix !!\n" << endl;

cout << "\*\*Lines = ";

cin >> m2.line;

cout << "\*\*Columns = ";

cin >> m2.column;

Matrix matrix2(m2.line, m2.column);

matrix2.f\_set\_matr();

system("pause");

system("cls");

cout << "\*\*\*First Matrix:" << endl;

matrix1.f\_show();

cout << "\n\*\*\*Second Matrix:" << endl;

matrix2.f\_show();

Matrix matrix3 = matrix1;

Matrix matrix4 = matrix1;

Matrix matrix5 = matrix1;

cout << "\n\*\*\*Matrix addition:" << endl;

matrix1.f\_addition(matrix2);

matrix1.f\_show();

cout << "\n\*\*\*Matrix subtraction: (matrix1 - matrix2)" << endl;

matrix4.f\_subtract(matrix2);

matrix4.f\_show();

cout << "\n\*\*\*Matrix multiplication:\n" << endl;

matrix5.f\_multipl\_matrix(matrix2);

matrix5.f\_show();

cout << "\n\*\*\*Multiply first matrix with number" << endl;

cout << "\*\*Imput number >>> ";

cin >> ml\_nr;

cout << endl;

matrix3.f\_multipl\_num(ml\_nr);

matrix3.f\_show();

system("pause");

exit(0);

}

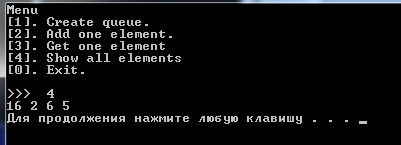
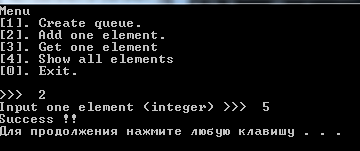
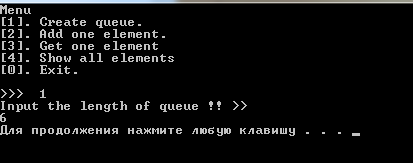
int main() {

f\_menu();

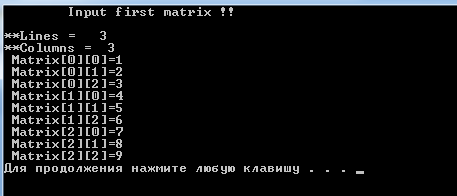
return (0);

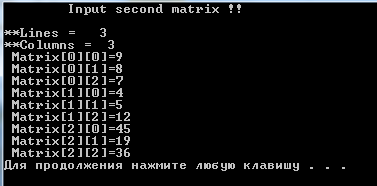
}

Output(a):

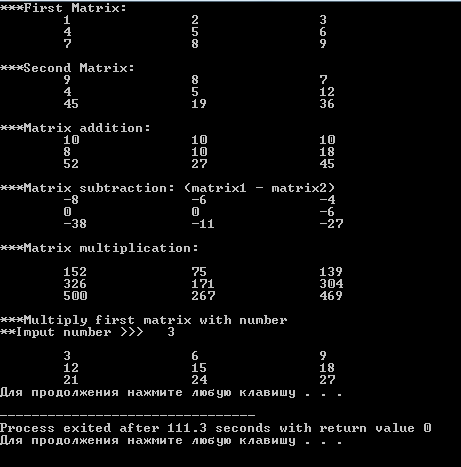


Output (b):





Rezolvarea sarcinilor:



Concluzie:

* In rezultatul elaborarii lucrarii date s-a pus baza aplicarii in practica a cunostintelor teoretice referitoare la utilizarea claselor, si constructorilor.
* Astfel se poate judeca despre posibilitatile largi acordate de limbajul C++ referitor la manipularea memoriei in directia dorita de utilizator.