ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ МОЛДОВЫ

Факультет Информатики, Вычислительной Техники и Микроэлектроники

Департамент Информатики и Системной Инженерии

# Отчёт

По лабораторной работе 2

|  |  |
| --- | --- |
| Выполнил: | ст. гр. SI-191 МайстренкоДанила |
| Проверил: | Митителу Виталий |

Кишинев – 2020

Лабораторная работа №2

Тема: Конструктор –функция инициализации объектов класса

**Цель работы:** изучение основ определения и использования конструкторов;

изучение основ определения и использования деструкторов;

изучение типов конструкторов; Задание (вариант 1):

а) Создать класс Date-дата с полями: день (1-28..31), месяц (1-12), год (целые числа). Определить конструкторы, функции-члены установки дня, месяца и года, функции-члены получения дня, месяца и года. Также две функции печати: печать по шаблону: "19 апреля 2003 года"и "19.04.2003". Функции установки полей класса должны проверять корректность задаваемых параметров.

b) Создать класс Matrix-матрица. Данный класс содержит указатель на int, количество строк и столбцов и переменную -код ошибки. Определить конструктор без параметров, конструктор с одним параметром –квадратная матрица и конструктор с двумя параметрами –прямоугольная матрица и др. Определить методы доступа: возвращение и определение значения элемента (i,j). Определить функции сложения и вычитания (матрицы с матрицей), умножение матрицы на матрицу. Определить умножение матрицы на число. Проверить работу этого класса. В случае нехватки памяти, несоответствия размерностей, выхода за пределы используемой памяти устанавливать код ошибки.

Решение:

*Исходный код программы:*

**a)** #include <bits/stdc++.h>

using namespace std;

static string months[12]={"january", "february", "march", "april", "may", "june", "july", "august", "september", "october", "november", "december"};

const char splash[34]="-------------------------------\n";

class date{

private:

int day;

int month;

int year;

public:

date(int day, int month, int year);

void set\_date();

string get\_date();

void show();

void show\_words();

friend bool valid();

~date();

};

void error(int num){

switch(num){

case 1:{

cout << "Invalid day" << endl;

break;

}

case 2:{

cout << "Invalid month" << endl;

break;

}

case 3:{

cout << "There's only 28 days in february" << endl;

break;

}

case 4:{

cout << "There's only 29 days in february" << endl;

break;

}

}

}

date::date(int day=1, int month=1, int year=1997){

this->day=day;

this->month=month;

this->year=year;

}

bool valid(int day, int month, int year){

if(day > 31 || day <1 ){

error(1);

return false;

}

else if(month > 12 || month < 1){

error(2);

return false;

}

else if(month == 2 && day > 28){

error(3);

return false;

}

else if (month == 2 && day > 29 && (year % 4) != 0){

error(4);

return false;

}

else

return true;

}

void date::set\_date(){

bool check=false;

while(!check){

cout << "Enter day: ";

cin >> day;

cout << "Enter month: ";

cin >> month;

cout << "Enter year: ";

cin >> year;

check=valid(day, month, year);

cout << splash;

}

}

date::~date(){}

string date::get\_date(){

string fin;

fin=to\_string(day) + '.' + to\_string(month) + '.' + to\_string(year) + '\n';

return fin;

}

void date::show(){

cout << "Entered date: " << day << "." << month << ".";

if(year < 0){

cout << abs(year) << "B.C." << endl;

}

else

cout << year << endl;

cout << splash;

return;

}

void date::show\_words(){

cout << "Entered date: " << day << " " << months[month-1] << ", ";

if(year < 0)

cout << abs(year) << " B.C." << endl;

else

cout << year << endl;

cout << splash;

return;

}

int main(){

date obj, obj1(2,12,1998);

obj.set\_date();

obj.show();

obj.show\_words();

cout << "Getter:";

cout << obj.get\_date();

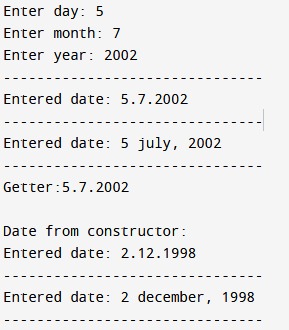
cout << "\nDate from constructor: " << endl;

obj1.show();

obj1.show\_words();

return 0;

}



**b)**

#include <bits/stdc++.h>

using namespace std;

const char splash[34]="--------------------\n";

class matrix{

private:

int\*\* elem;

int row;

int line;

int error=0;

public:

matrix(){};

matrix(int row);

matrix(int row, int line);

void set\_matrix(bool rand);

void get\_elem();

void handler();

void show\_matrix();

~matrix();

friend void plus\_matrix(matrix &a, matrix &b);

friend void substract(matrix &a, matrix &b);

friend void mult\_m(matrix &a, matrix &b);

friend void mult\_n(matrix &a);

};

matrix::matrix(int row){

this->row=row;

line = row;

elem = new int\*[row];

if(!elem){

error=4;

handler();

}

for(int i=0;i<row;i++)

elem[i] = new int[line];

if(!elem){

error=4;

handler();

}

}

matrix::matrix(int row, int line){

this->row = row;

this->line = line;

elem = new int\*[row];

for(int i=0;i<row;i++)

elem[i] = new int[line];

}

void matrix::set\_matrix(bool random = false){

if(random)

for(int i=0;i<row;i++)

for(int j=0;j<line;j++)

elem[i][j] = rand() % 50;

else{

for(int i=0;i<row;i++)

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

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

cin >> elem[i][j];

}

cout << splash;

}

return;

}

void matrix::show\_matrix(){

cout << splash;

cout << "\t";

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

cout << k << "\t";

cout << endl;

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

cout << i;

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

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

}

cout << endl;

}

cout << splash;

return;

}

void matrix::handler(){

switch(error){

case 1:{

cout << "Matrix's aren't the same size. Can't sum.\n";

break;

}

case 2:{

cout << "Matrix's aren't the same size. Cant't substract.\n";

break;

}

case 3:{

cout << "Matrix's amount of lines and rows aren't equal. Can't multiply.\n";

break;

}

case 4:{

cout << "Couldn't allocate memory.\n";

break;

}

case 5:{

cout << "Out of range.\n";

break;

}

}

error=0;

return;

}

void plus\_matrix(matrix &a, matrix &b){

if(a.row != b.row || a.line != b.line){

a.error=1;

a.handler();

return;

}

else

if(a.row == b.row && a.line == b.line){

matrix res\_sum(a.row,a.line);

for (int i = 0; i < a.row; i++){

for (int j = 0; j < a.line; j++){

res\_sum.elem[i][j] = a.elem[i][j] + b.elem[i][j];

}

}

cout << "Result of addition\n";

res\_sum.show\_matrix();

}

return;

}

void substract(matrix &a, matrix &b){

if(a.row != b.row || a.line != b.line){

a.error=2;

a.handler();

return;

}

else

if(a.row == b.row && a.line == b.line){

matrix res\_substract(a.row, a.line);

for (int i = 0; i < a.row; i++){

for (int j = 0; j < a.line; j++){

res\_substract.elem[i][j] = a.elem[i][j] - b.elem[i][j];

}

}

cout << "Result of substraction\n";

res\_substract.show\_matrix();

}

return;

}

void mult\_m(matrix &a, matrix &b){

if( a.line != b.row){

a.error = 3;

a.handler();

}

else{

matrix res\_mult\_m(a.row,b.line);

for (int i = 0; i < a.row; i++)

for (int j = 0; j < b.line; j++){

res\_mult\_m.elem[i][j] = 0;

for (int k = 0; k < a.line; k++)

res\_mult\_m.elem[i][j] += a.elem[i][k] \* b.elem[k][j];

}

cout << "Result of multiplying two matrix's\n";

res\_mult\_m.show\_matrix();

}

return;

}

void mult\_n(matrix &a){

matrix res\_mult\_n(a.row,a.line);

int temp;

cout << "Enter number: ";

cin >> temp;

for(int i=0;i<a.row;i++)

for(int j=0;j<a.line;j++)

res\_mult\_n.elem[i][j] = a.elem[i][j] \* temp;

cout << "Result of multiplication of the first matrix to " << temp << " equals:\n";

res\_mult\_n.show\_matrix();

return;

}

void default\_check(int num, int &row, int &line){

line = row = 2;

char check;

cout << "Do you want to enter size of " << num << " matrix on your own? (default - 2\*2 size) ////////// [Y/N] ////////////\n";

cin >> check;

if(check == 'y' || check == 'Y'){

cout << "Enter the size of matrix: ";

cin >> row >> line;

}

return;

}

bool set\_check(){

char rand;

cout << "Do you want to fill both matrix's with random elements? ////////// [Y/N] //////////";

cin >> rand;

if (rand == 'Y' || rand == 'y')

return true;

else

return false;

}

void matrix::get\_elem(){

show\_matrix();

int i,j;

int temp;

cout << "Enter which number you want to change[row][line]: ";

cin >> i;

cin >> j;

if(i>row-1 || j>line-1){

error=5;

handler();

return;

}

cout << "Enter new number: ";

cin >> temp;

elem[i][j] = temp;

show\_matrix();

return;

}

matrix::~matrix(){

for(int i=0;i<row;i++)

delete[] elem[i];

delete[] elem;

}

int main(){

bool rand;

srand(time(NULL));

int row, line;

default\_check(1, row, line);

matrix obj\_1(row, line);

default\_check(2, row, line);

matrix obj\_2(row, line);

rand = set\_check();

obj\_1.set\_matrix(rand);

if (rand)

obj\_1.show\_matrix();

obj\_2.set\_matrix(rand);

if (rand)

obj\_2.show\_matrix();

plus\_matrix(obj\_1, obj\_2);

substract(obj\_1,obj\_2);

mult\_m(obj\_1,obj\_2);

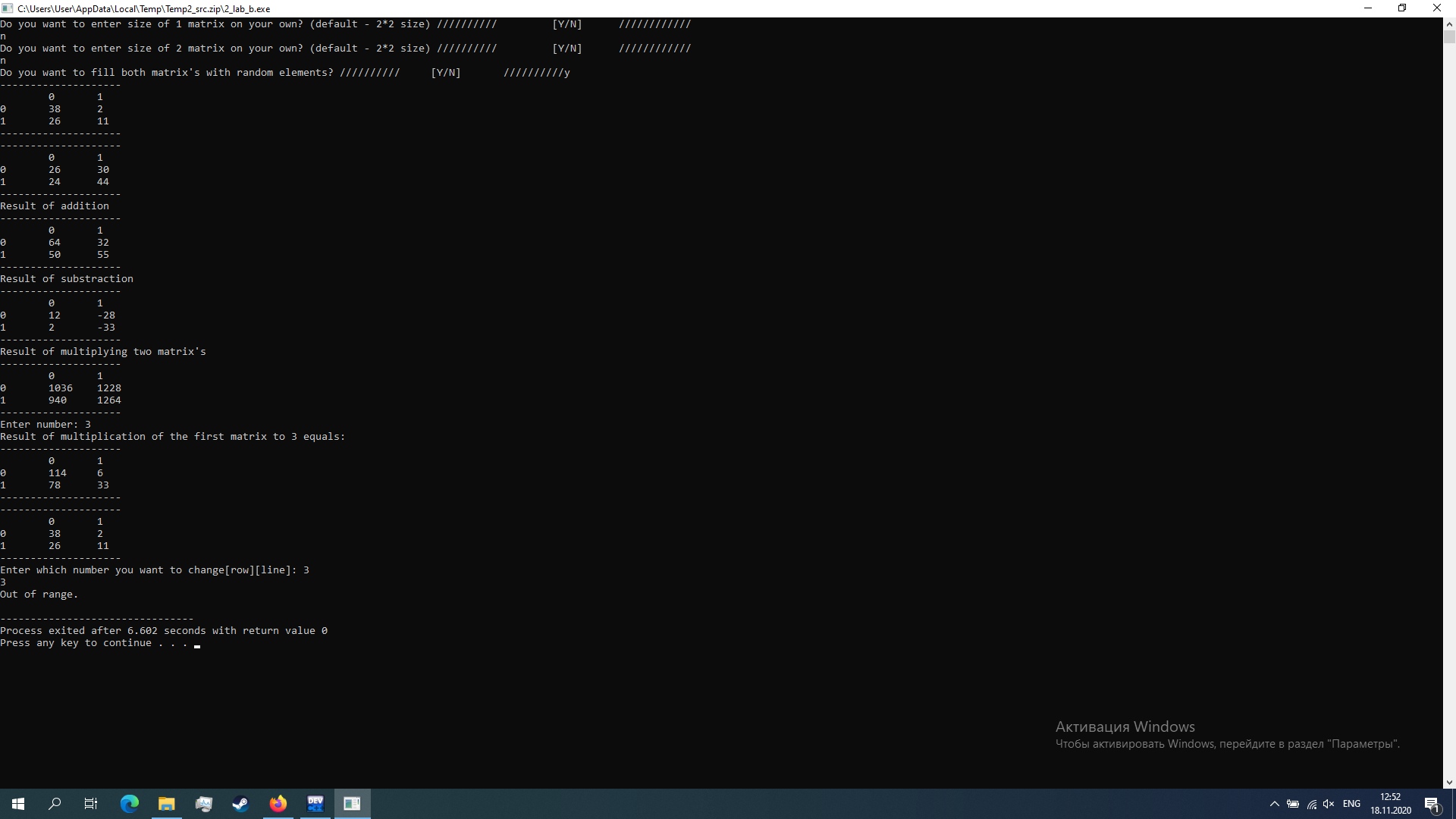
mult\_n(obj\_1);

obj\_1.get\_elem();

return 0;

}

*Результат:*



Вывод:

В ходе данной лабораторной работы были изучены возможности языка C++ для создания класса. В частности были изучены:

* Более углубленные понятия об ООП
* Типы конструкторов и их создание
* Применение различных типов конструкторов для различных целей
* Перегрузка конструкторов
* Создание деструктора
* Getter и setter для класса