# ЛАБОРАТОРНАЯ РАБОТА №4

# ИССЛЕДОВАНИЕ МЕХАНИЗМА ДРУЖЕСТВЕННОСТИ

Цель работы

Приобретение практических навыков при написании объектно-ориентированных программ с использованием механизма дружественности.

Задания

1. Изучить основы работы с “дружественными” классами и функциями;
2. Разработать программу по варианту (Рисунок 1), демонстрирующую использование дружественной функции. Описать заданные по варианту классы (содержащие private поля и методы). Для каждого класса описать конструктор по умолчанию и конструктор с параметрами, а также деструктор (по необходимости). Создать функцию, дружественную обоим классам, и в ней обратиться к их закрытым полям и методам;

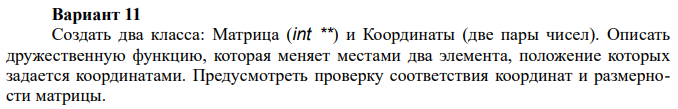


Рисунок 1 – Вариант задания

1. Разработать тестовые примеры и выполнить отладку программы;
2. Получить результаты работы программы и исследовать её свойства для различных режимов работы, сформулировать выводы;

Текст программы

#include <iostream>

using namespace std;

// Объявление функции и классов

class Matrix;

class Coords;

int swap(Matrix\* m, Coords c1, Coords c2);

// Класс для матрицы чисел

class Matrix

{

int\*\* matr; // указатель на матрицу

int lenX; // число столбцов

int lenY; // число строк

public:

friend int swap(Matrix\* m, Coords c1, Coords c2);

Matrix(int X, int Y) // конструктор c параметрами

{

matr = new int\* [Y];

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

matr[i] = new int[X];

lenX = X;

lenY = Y;

}

Matrix() // конструктор по умолчанию

{

cout << "Number of columns - ";

cin >> lenY;

cout << "Number of rows - ";

cin >> lenX;

matr = new int\* [lenY];

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

matr[i] = new int[lenX];

}

~Matrix() // деструктор

{

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

delete[] matr[i];

delete[] matr;

lenX = lenY = 0;

cout << "\nDestructor worked" << endl;

}

void put() // ввод матрицы

{

cout << "\nEnter matrix:" << endl;

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

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

{

cout << "Element (" << j << "; " << i << ") - ";

cin >> matr[i][j];

}

}

void print() // вывод матрицы

{

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

{

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

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

cout << endl;

}

}

};

// Класс для координат числа в матрице

class Coords

{

int x;

int y;

public:

friend int swap(Matrix\* m, Coords c1, Coords c2);

Coords(int \_x, int \_y) // конструктор с параметрами

{

x = \_x;

y = \_y;

}

Coords() // конструктор по умолчанию

{

cout << "Coordinate x - ";

cin >> x;

cout << "Coordinate y - ";

cin >> y;

}

};

// Функция, меняющая элементы матрицы местами по координатам

int swap(Matrix\* m, Coords c1, Coords c2)

{

if ((c1.x >= m->lenX) || (c1.y >= m->lenY) || (c2.x >= m->lenX) || (c2.y >= m->lenY))

return 0;

int forSwap = m->matr[c1.y][c1.x];

m->matr[c1.y][c1.x] = m->matr[c2.y][c2.x];

m->matr[c2.y][c2.x] = forSwap;

return 1;

}

// Главный класс

int main()

{

Matrix M; // создание и заполнение матрицы

M.put();

M.print();

cout << "\nEnter coords 1:" << endl; // изменение по координатам

Coords c1;

cout << "\nEnter coords 2:" << endl;

Coords c2;

cout << "\nSwap result: " << swap(&M, c1, c2) << endl;

M.print(); // вывод матрицы

}

Тестовые примеры

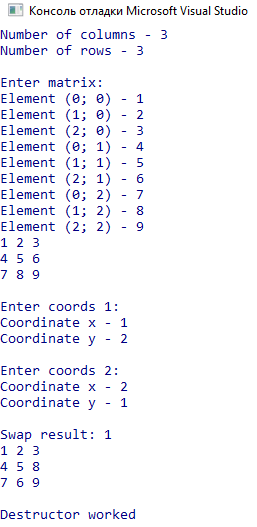


Рисунок 2 – Тестирование программы

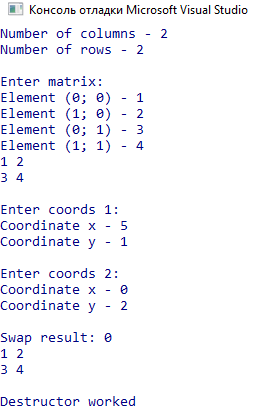


Рисунок 3 – Тестирование проверки выхода за пределы матрицы

Вывод

Ходе работы был исследован механизм дружественности в объектно-ориентированном программировании. Была написана программа, содержащая реализацию функции, дружественной двум классам.