**ЛАБОРАТОРНАЯ РАБОТА №4**

**«ИССЛЕДОВАНИЕ МЕХАНИЗМА ДРУЖЕСТВЕННОСТИ»**

**Цель работы**: Приобретение практических навыков при написании объектно-ориентированных программ с использованием механизма дружественности.

**Вариант задания**

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.

Вариант 11

Создать два класса: Матрица (int \*\*) и Координаты (две пары чисел). Описать

дружественную функцию, которая меняет местами два элемента, положение которых задается координатами. Предусмотреть проверку соответствия координат и размерности матрицы.

**2. Код программы на языке C++**

#include <iostream>

class Matrix;

class Cords;

bool swop(Cords& cord, Matrix& matrs);

class Cords

{

public:

Cords();

void enter\_cord();

Cords(int\* x, int\* y);

private:

int x[2];

int y[2];

friend bool swop(Cords& cord, Matrix& matrs);

};

Cords::Cords()

{

x[0] = 0; y[0] = 0;

x[1] = 0; y[1] = 0;

}

Cords::Cords(int \*x,int \*y)

{

this->x[0] = x[0];

this->x[1] = x[1];

this->y[0] = y[0];

this->y[1] = y[1];

}

void Cords::enter\_cord()

{

std::cout << "Введите x1--> ";

std::cin >> x[0];

std::cout << "Введите y1--> ";

std::cin >> y[0];

std::cout << "Введите x2--> ";

std::cin >> x[1];

std::cout << "Введите y2--> ";

std::cin >> y[1];

std::cout << "Ввод координат через дружественный класс Matrix успешен" << std::endl;

}

class Matrix

{

public:

Matrix();

Matrix(int n, int m);

void show();

~Matrix();

friend bool swop(Cords& cord, Matrix& matrs);

private:

int\*\* matr;

int n, m;

};

Matrix::Matrix()

{

matr = NULL;

}

Matrix::Matrix(int n, int m)

{

this->n = n; this->m = m;

matr = (int\*\*)calloc(1,sizeof(int\*) \* n);

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

matr[i] = (int\*)calloc(1,sizeof(int) \* m);

}

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

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

matr[i][j] = rand()%100;

}

Matrix::~Matrix()

{

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

free(matr[i]);

}

free(matr);

}

void Matrix::show() {

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

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

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

std::cout << std::endl;

}

}

bool swop(Cords &cord,Matrix &matrs)

{

int temp;

if (cord.x[0] > matrs.m || cord.x[0] < 0) {

std::cout << "\n Координаты выходят за пределы матрицы" << std::endl;

return false;

}

if (cord.x[1] > matrs.m || cord.x[1] < 0) {

std::cout << "\n Координаты выходят за пределы матрицы" << std::endl;

return false;

}

if (cord.y[0] > matrs.n || cord.y[0] < 0) {

std::cout << "\n Координаты выходят за пределы матрицы" << std::endl;

return false;

}

if (cord.y[1] > matrs.n || cord.y[1] < 0) {

std::cout << "\n Координаты выходят за пределы матрицы" << std::endl;

return false;

}

std::cout << "\n Элемент -->" << matrs.matr[cord.x[0]][cord.y[0]] <<" Будет заменен местом с -->" << matrs.matr[cord.x[1]][cord.y[1]] << std::endl;

temp = matrs.matr[cord.x[0]][cord.y[0]];

matrs.matr[cord.x[0]][cord.y[0]] = matrs.matr[cord.x[1]][cord.y[1]];

matrs.matr[cord.x[1]][cord.y[1]] = temp;

std::cout << "\n Элементы успешно заменены местами" << std::endl;

}

int main()

{

system("chcp 1251");

Matrix obj1(10, 10);

Cords obj2;

obj2.enter\_cord();

obj1.show();

swop(obj2, obj1);

obj1.show();

std::cin;

}

**3. Тестирование и отладка**

Для тестирования данной программы, в самом коде программы был создан объект класса Matrix, размером 10 на 10. После программа запрашивает у пользователя какие элементы поменять местами. После ввод данных происходит вызов функции, которая обращается к закрытым полям классов Matrix и Cord.ы

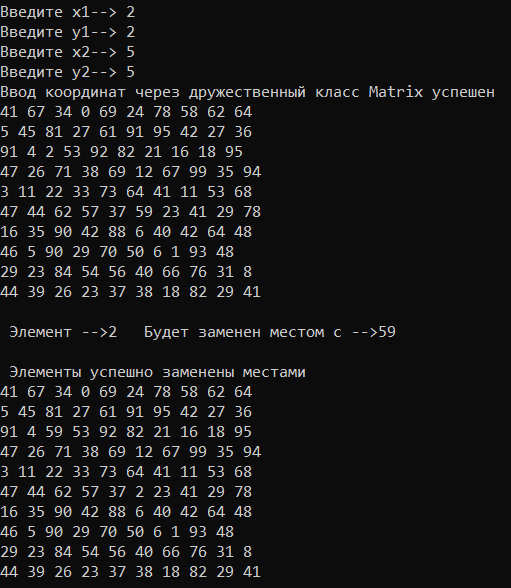


Рисунок 1 – Результат выполнения программы.

В результате тестирования, видно, что вызов происходит корректно, и функция выполняет свои действия. При этом функция обращается к закрытым(приватным) полям классов.

**Вывод**

При выполнении данной лабораторной работы были получены навыки работы с механизмом дружественности в языке программирования C++, а именно с дружественными классами и дружественными функциями. Также были повторно закреплены навыки создания объектов из класса и работы с ними.