

## 4 ЛАБОРАТОРНАЯ РАБОТА №4

### «ИССЛЕДОВАНИЕ МЕХАНИЗМА ДРУЖЕСТВЕННОСТИ»

#### 4.1 Цель работы

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

#### 4.2 Вариант задания – 8

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.

Требуется создать два класса: Вектор (`int *`) и Матрица (`double **`). Описать дружественную функцию, заменяющую все элементы главной диагонали матрицы на соответствующие элементы вектора. Учесть проверку соответствия размерностей.

#### 4.3 Ход работы

4.3.1 В программе были созданы 2 класса “Vector” и “Matrix” имеющие общую “функцию – друга”. Были описаны конструкторы и деструкторы, функции демонстрации вектора и матрицы. В функции `main` проведены действия с объектами этих двух классов, а также продемонстрирована работа дружественной функции.

4.3.2 Написана программа на C++ согласно вышеописанного алгоритма.

```
#include <iostream>

#include <windows.h>

using namespace std;

class Matrix; // Неполное объявление класса

// Класс - вектор из целых чисел

class Vector {
```

```

        int *vector;

        int arrLen;

public:
        Vector();

        Vector(int _arrLen);

        ~Vector();

        void showVector();

        friend void replacingDiagonal(Vector *ob1, Matrix *ob2); // Дружественная ф-ция
};

// Класс - матрица из чисел double
class Matrix {
        double **matrix;

        int rows, cols;

public:
        Matrix();

        Matrix(int _rows, int _cols);

        ~Matrix();

        void showMatrix();

        friend void replacingDiagonal(Vector *ob1, Matrix *ob2); // Дружественная ф-ция
};

// Конструктор по умолчанию класса Vector
Vector :: Vector() {
        arrLen = 1;

        vector = new int[arrLen]; //выделение памяти под массив целых чисел размером
arrLen

        vector[0] = 1;

        // Создали массив из одного элемента с числом "1"

        cout << "Конструктор Vector по умолчанию" << endl << endl;

```

```

}

// Конструктор с параметрами (длина массива) класса Vector
Vector :: Vector(int _arrLen) {

    arrLen = _arrLen;

    vector = new int[arrLen]; // выделяем память

    // заполняем массив

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

        cout << "Введите " << i+1 << " элемент массива: ";

        cin >> vector[i];

    }

    cout << "Конструктор Vector с параметрами" << endl << endl;

}

// Деструктор класса Vector
Vector :: ~Vector() {

    delete [] vector; // очистить память выделенную массиву

    cout << "Деструктор Vector" << endl;

}

// Показать вектор класса Vector
void Vector :: showVector() {

    cout << "Вектор:" << endl;

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

        cout << vector[i] << "\t";

    }

    cout << endl;

}

// Конструктор по умолчанию класса Matrix
Matrix :: Matrix() {

```

```

rows = 1;

cols = 1;

matrix = new double* [rows]; //выделение памяти под массив указателей на массивы

matrix[0] = new double [cols]; // выделение памяти под один массив

// Создали матрицу из одного элемента с числом "1"

cout << "Конструктор Matrix по умолчанию" << endl << endl;

}

```

// Конструктор с параметрами (кол-во строк, столбцов матрицы) класса Matrix

```

Matrix :: Matrix(int _rows, int _cols) {

    rows = _rows;

    cols = _cols;

    // выделяем память

    matrix = new double* [rows];

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

        matrix[i] = new double [cols];

    }

    // заполняем массив

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

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

            cout << "Введите [" << i+1 << "]"[" << j+1 << "]" элемент матрицы: ";

            cin >> matrix[i][j];

        }

    }

    cout << "Конструктор Matrix с параметрами" << endl << endl;

}

```

// Деструктор класса Matrix

```

Matrix :: ~Matrix() {

    // очистить память выделенную матрице

```

```

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

            delete [] matrix[i];

        }

        delete [] matrix;

        cout << "Деструктор Matrix" << endl;

    }

// Показать матрицу класса Matrix

void Matrix :: showMatrix() {

    cout << "Матрица:" << endl;

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

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

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

        }

        cout << endl;

    }

}

// Функция друг, меняющая главную диагональ матрицы на вектор

void replacingDiagonal(Vector *ob1, Matrix *ob2) {

    if ((ob1->arrLen <= ob2->rows) && (ob1->arrLen <= ob2->cols)) {

        for (int i = 0; i < ob1->arrLen; i++) {

            ob2->matrix[i][i] = ob1->vector[i];

        }

        cout << "Функция друг, меняющая главную диагональ матрицы на вектор была  
выполнена" << endl << endl;

    }

    else {

        cout << "К сожалению длина вектора больше, чем может вместить в себя главная  
диагональ матрицы" << endl

            << "поэтому операция замещения не будет произведена" << endl << endl;

    }

}

```

```
}
```

```
//=====MAIN=====
```

```
int main() {
```

```
    SetConsoleCP(1251);
```

```
    SetConsoleOutputCP(1251);
```

```
    system("color B0");
```

```
    cout << "Создать вектор и показать его" << endl;
```

```
    // Создать вектор
```

```
    Vector *vectorObj1 = new Vector(4);
```

```
    // Показать вектор
```

```
    vectorObj1->showVector();
```

```
    cout << endl << endl;
```

```
    cout << "Создать матрицу и показать её" << endl;
```

```
    // Создать матрицу
```

```
    Matrix *matrixObj1 = new Matrix(5,6);
```

```
    // Показать матрицу
```

```
    matrixObj1->showMatrix();
```

```
    cout << endl << endl;
```

```
    cout << "Заменить главную диагональ матрицы на вектор и показать её" << endl;
```

```
    // Заменить главную диагональ матрицы на вектор
```

```
    replacingDiagonal(vectorObj1, matrixObj1);
```

```
    // Показать матрицу
```

```
    matrixObj1->showMatrix();
```

```
    cout << endl << endl;
```

```
    // Удалить объекты (очистить память)
```

```
delete vectorObj1;

delete matrixObj1;


cout << endl << endl;

cout << "Создать вектор и показать его" << endl;

// Создать вектор
Vector *vectorObj2 = new Vector(3);

// Показать вектор
vectorObj1->showVector();

cout << endl << endl;


cout << "Создать матрицу и показать её" << endl;

// Создать матрицу
Matrix *matrixObj2 = new Matrix(2,2);

// Показать матрицу
matrixObj1->showMatrix();

cout << endl << endl;


cout << "Заменить главную диагональ матрицы на вектор и показать её" << endl;

// Заменить главную диагональ матрицы на вектор
replacingDiagonal(vectorObj2, matrixObj2);

// Показать матрицу
matrixObj1->showMatrix();

cout << endl << endl;


// Удалить объекты (очистить память)
delete vectorObj1;

delete matrixObj1;


system("pause");
```

```

return 0;

}

```

### 4.3.3 Выполнена отладка программы.

Результаты тестирования отображены на рисунке 4.1. На изображении видно как вводятся данные, а именно заполняется вручную вектор и матрица, выводятся вектор и матрица, затем заменяется главная диагональ матрицы вектором, далее итоговая матрица выводится на экран. На рисунке 4.2 производятся аналогичные действия, но размер вектора намеренно больше длины главной диагонали матрицы, поэтому функция-друг сказала, что у нас не совпадают размеры и матрица не меняется.

```

D:\SevSu\3_sem\OOP\4\program4.exe
Создать вектор и показать его
Введите 1 элемент массива: 1
Введите 2 элемент массива: 2
Введите 3 элемент массива: 3
Введите 4 элемент массива: 4
Конструктор Vector с параметрами

Вектор:
1      2      3      4

Создать матрицу и показать её
Введите [1][1] элемент матрицы: 1
Введите [1][2] элемент матрицы: 2
Введите [1][3] элемент матрицы: 3
Введите [1][4] элемент матрицы: 4
Введите [1][5] элемент матрицы: 5
Введите [1][6] элемент матрицы: 66
Введите [2][1] элемент матрицы: 7
Введите [2][2] элемент матрицы: 8
Введите [2][3] элемент матрицы: 9
Введите [2][4] элемент матрицы: 10
Введите [2][5] элемент матрицы: 11
Введите [2][6] элемент матрицы: 12
Введите [3][1] элемент матрицы: 13
Введите [3][2] элемент матрицы: 14
Введите [3][3] элемент матрицы: 15
Введите [3][4] элемент матрицы: 16
Введите [3][5] элемент матрицы: 17
Введите [3][6] элемент матрицы: 18
Введите [4][1] элемент матрицы: 19
Введите [4][2] элемент матрицы: 20
Введите [4][3] элемент матрицы: 21
Введите [4][4] элемент матрицы: 22
Введите [4][5] элемент матрицы: 23
Введите [4][6] элемент матрицы: 24
Введите [5][1] элемент матрицы: 25
Введите [5][2] элемент матрицы: 26
Введите [5][3] элемент матрицы: 27
Введите [5][4] элемент матрицы: 28.234
Введите [5][5] элемент матрицы: 29.613
Введите [5][6] элемент матрицы: 30.111
Конструктор Matrix с параметрами

Матрица:
1      2      3      4      5      66
7      8      9      10     11     12
13     14     15     16     17     18
19     20     21     22     23     24
25     26     37     28.234  29.613  30.111

Заменить главную диагональ матрицы на вектор и показать её
Функция друг, меняющая главную диагональ матрицы на вектор была выполнена

Матрица:
1      2      3      4      5      66
7      2      9      10     11     12
13     14     3      16     17     18
19     20     21     4      23     24
25     26     37     28.234  29.613  30.111

Деструктор Vector
Деструктор Matrix

```

Рисунок 4.1 – Выполнение программы при верных размерах матрицы и вектора



```
Создать вектор и показать его
Введите 1 элемент массива: 1
Введите 2 элемент массива: 2
Введите 3 элемент массива: 3
Конструктор Vector с параметрами

Вектор:
1      2      3

Создать матрицу и показать её
Введите [1][1] элемент матрицы: 8
Введите [1][2] элемент матрицы: 7
Введите [2][1] элемент матрицы: 6
Введите [2][2] элемент матрицы: 5
Конструктор Matrix с параметрами

Матрица:
8      7
6      5

Заменить главную диагональ матрицы на вектор и показать её
К сожалению длина вектора больше, чем может вместить в себя главная диагональ матрицы
поэтому операция замещения не будет произведена

Матрица:
8      7
6      5

Деструктор Vector
Деструктор Matrix
Для продолжения нажмите любую клавишу . . .
```

Рисунок 4.1 – Выполнение программы при неверных размерах матрицы и вектора

Результаты тестирования полностью соответствуют ожиданиям.

## Выводы

В ходе выполнения данной лабораторной работы были получены навыки разработки программ, использующих дружественные функции. Были закреплены навыки разработки и отладки программ, использующих классы и объекты. Полученные во время разработки навыки помогут разрабатывать более сложные программы с использованием классов и объектов, более эффективные по времени выполнения алгоритмы.