

## Спецификация лабораторной работы № 5

### Часть 1. Встроенные массивы.

В данной программе нужно создать функцию для работы со встроенным массивом. Значения для матрицы (встроенный массив) вводятся из файла input.txt и полученный ответ из функции выводится в поток cout.

Задание:

- 5) Количество столбцов, содержащих хотя бы один нулевой элемент.

### Часть 2. Массивы в динамической памяти.

Во второй части задания нужно также создать функцию для работы с квадратной( $N*N$ ) матрицей. Размерность вводится из файла input2.txt и полученный ответ из функции выводится в файл output.txt.

Задание:

- 5) Матрицу  $N*N$  заполнить натуральными числами от 1 до  $N*N$  по спирали, начинающейся в верхнем левом углу и закрученной по часовой стрелке.

## Тестовый набор с выходными данными

В input.txt и input2.txt лежат входные данные для работы с матрицей.

input.txt	input2.txt	output.txt
1	1 23 0 46 89	
2	37 27 5 29 43	
3	37 27 2 29 12	
4	0 27 80 29 11	

input2.txt	input.txt	output.txt
1	5	

Вывод:

```
Консоль отладки Microsoft Visual Studio
Ответ к первой части:
2
```

input2.txt	input.txt	output.txt	Labwork 5.cpp
1	Ответ к второй части:		
2	1 2 3 4 5		
3	16 17 18 19 6		
4	15 24 25 20 7		
5	14 23 22 21 8		
6	13 12 11 10 9		
7			

## Программа:

```
#include <iostream>
#include <fstream>
#include <iomanip>

const int ROW = 4;
const int COL = 5;

void decideFunctionOne(int arr[][COL], const int row, const int col);

void decideFunctionTwo(int** arr, const int size);

int** createDynamicArray(int size);

void outChapterOne();

void outChapterTwo();

void createRowsAndCols(int** arr, int size, int m);

int main()
{
    setlocale(LC_ALL, "ru");
    {
        outChapterOne();
    }
    {
        outChapterTwo();
    }
    return 0;
}

void decideFunctionOne(int arr[][COL], const int row, const int col)
{
    int count = 0;
    int numberCol = 0;

    for (int i = 0; i < row; i++)
    {
        for (int j = 0; j < col; j++)
        {
            if (!arr[i][j])
            {
                ++count;
                break;
            }
        }
    }
    std::cout << count;
}

void decideFunctionTwo(int** arr, const int size)
{
    std::ofstream fout;
    fout.open("output.txt");

    fout << "Ответ к второй части:" << std::endl;
    int m = 1;

    createRowsAndCols(arr, size, m);

    for (int i = 0; i < size; i++)
```

```

    {
        for (int j = 0; j < size - 1; j++)
        {
            fout << std::setw(3) << arr[i][j] << " ";
        }
        fout << std::setw(3) << arr[i][size - 1] << std::endl;
    }
    fout.close();
}

int** createDynamicArray(int size)
{
    int** dynamicArr = new int* [size];
    for (int i = 0; i < size; i++)
    {
        dynamicArr[i] = new int[size];
    }

    for (int i = 0; i < size; i++)
    {
        for (int j = 0; j < size; j++)
        {
            dynamicArr[i][j] = 0;
        }
    }
    return dynamicArr;
}

void outChapterOne()
{
    std::ifstream fin;
    fin.open("input.txt");

    int arr[ROW][COL];
    for (int i = 0; i < ROW; i++)
    {
        for (int j = 0; j < COL; j++)
        {
            fin >> arr[i][j];
            if (fin.fail())
            {
                std::cout << "Ошибка. Попробуйте еще раз";
                exit(1);
            }
        }
    }

    std::cout << "Ответ к первой части:" << std::endl;
    decideFunctionOne(arr, ROW, COL);
    fin.close();
}

void outChapterTwo()
{
    int size;

    std::ifstream fin;
    fin.open("input2.txt");
    std::cout << "\n";
    fin >> size;
    if (fin.fail() || size < 0)
    {
        std::cout << "Ошибка. Попробуйте еще раз";
        exit(1);
    }
}

```

```

    int** dynamicArr = createDynamicArray(size);
    decideFunctionTwo(dynamicArr, size);
    delete[] dynamicArr;
    fin.close();
}

void createRowsAndCols(int** arr, int size, int m)
{
    if (size % 2 != 0)
    {
        arr[(size / 2)][(size / 2)] = size * size;    // Если число нечетное, то
находим центр матрицы
    }
    for (int i = 0; i < (size / 2); i++)
    {
        for (int j = i; j < (size - i); j++)
        {
            arr[i][j] = m;
            m++;
        }
        for (int j = 1; j < (size - i - i); j++)
        {
            arr[(j + i)][(size - i) - 1] = m;
            m++;
        }
        for (int j = (size - 2) - i; j >= i; j--)
        {
            arr[(size - i) - 1][j] = m;
            m++;
        }
        for (int j = ((size - i) - 2); j > i; j--)
        {
            arr[j][i] = m;
            m++;
        }
    }
}

```