МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
УЧРЕЖДЕНИЯ ОБРАЗОВВАНИЯ
«БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»
Кафедра интеллектуальных информационных технологий

Отчёт по лабораторной работе №7
по дисциплине Объектно-Ориентированные Технологии Программирования и Стандарты Проектирования за I семестр
Тема: "Умные указатели"

Выполнил:
студент 2 курса
IV семестра
факультета ЭИС
группы ПО-4(1)
Галанин П. И.
«__» _____ 2021 г.

Проверил:
магистрант
кафедры ИИТ
Миндер А. В.
«__» _____ 2021 г.

Брест 2021

Отчёт по лабораторной работе №7

Тема: «Умные указатели»

Цель работы: Изучить применение умных указателей.

Ход работы:

Вариант V задание 1

Условие: Одномерный массив. Удалить элемент с заданным номером.

Решение:

Листинг: main.cpp

```cpp
#include <iostream>
#include <memory>

//функция варианта 5 задания 1
void option5_task1();
//функция напечатает массив
template <typename T>
void print_1d_array(T* arr, int size);
//функция сгенерирует массив из int элементов в промежутке от a до b
void generate_int_elements_in_1d_array(int* arr, int size, int a = 1, int b = 99);
//функция, которая удалит элемент из массива
template <typename T>
std::shared_ptr<T []> del_index_in_1d_array(
    std::shared_ptr<T []> arr,
    int *size,
    int deleted_index
);

int main()
{
    srand(100); //Seed рандома. Программа всегда будет давать один и тот же результат при рандоме
    option5_task1();

    return 0;
}

void option5_task1()
{
    int size = 7;                                       //размер массива
    std::shared_ptr<int[]> arr(new int[size]);          //выделяем память под массив

    generate_int_elements_in_1d_array(arr.get(), size); //генерируем числа в массив
    print_1d_array(arr.get(), size);                    //печатаем массив

    arr = move(                                         //передаём владение
        del_index_in_1d_array(move(arr), &size, 3)      //возвратит умный указатель
    );
    print_1d_array(arr.get(), size);                    //печатаем массив
```

```cpp
}

template <typename T>
void print_1d_array(T* arr, int size)
{
    std::cout << "[" << std::endl;
    for (int i = 0; i < size; i++)
    {
        std::cout << "\t" << i << ": \"" << arr[i] << "\",\n";
    }
    std::cout << "]" << std::endl;
}

void generate_int_elements_in_1d_array(int* arr, int size, int a, int b)
{
    for (int i = 0; i < size; i++)
    {
        arr[i] = rand() % (b - a + 1) + a;
    }
}

template <typename T>
std::shared_ptr<T []> del_index_in_1d_array(
    std::shared_ptr<T []> arr,
    int *size,
    int deleted_index
)
{
    if (deleted_index <= 0 || deleted_index >= *size)    //if index in (-00; 0] u [size; ++00]
    {
        std::cout << deleted_index << " not in (0; " << *size << ")\n";
        return arr;
    }

    *size -= 1;                                          //уменьшаем размер массива
    for (int i = deleted_index; i < *size; i++)          //от удаляемого элемента до конца
    {
        arr[i] = arr[i + 1];                             //смещаем элементы
    }

    std::shared_ptr<T[]> temp_arr(new T[*size]);         //выделяем память под новый массив
    for (int i = 0; i < *size; i++)
    {
        temp_arr[i] = arr[i];                            //копируем элементы
    }

    arr = move(temp_arr);                                //передаём права
    return arr;
}
```

Листинг: Out

```
[
        0: "12",
        1: "24",
        2: "45",
        3: "39",
        4: "71",
        5: "29",
        6: "47",
```

```
]
[
        0: "12",
        1: "24",
        2: "45",
        3: "71",
        4: "29",
        5: "47",
]
```

Вариант V задание 2

Условие: Двумерный массив. Добавить строку в начало матрицы

Решение:

Листинг: main.cpp

```cpp
#include <iostream>
#include <memory>

//функция варианта 5 задания 2
void option5_task2();
//функция сгенерирует элементы для int массива
void generate_elements_in_2d_array(
    const std::unique_ptr<std::unique_ptr<int[]>[]> &arr,
    const int rows,
    const int cols,
    int a = 1,
    int b = 99
);
//функция напечатает функцию
template <typename T>
void print_2d_array(
    const std::unique_ptr<std::unique_ptr<T[]>[]> &arr,
    const int rows,
    const int cols
);
//функция вставит строку в функцию
template <typename T>
void add_line_to_2d_array(
    std::unique_ptr<std::unique_ptr<T[]>[]> &arr,
    int *rows,
    const int cols
);

int main()
{
    srand(100); //Seed рандома. Программа всегда будет давать один и тот же результат при рандоме
    option5_task2();

    return 0;
}

void option5_task2()
{
    int rows = 5;                                   //количество строк
    int cols = 12;                                  //количество столбцов
    std::unique_ptr<
        std::unique_ptr<int[]>[]
```

```
    > arr( std::make_unique<std::unique_ptr<int[]>[]>(rows) );  //выделение памяти под 2d массив
    for (int i = 0; i < rows; i++)                //заполняем одномерный массив массивом
    {
        arr[i] = std::make_unique<int[]>(cols); //в ячейке массива массив
    }

    generate_elements_in_2d_array(arr, rows, cols);             //заполняем массив
    print_2d_array(arr, rows, cols);              //печатаем 2d массив

    add_line_to_2d_array(arr, &rows, cols);      //добавляем строку в массив
    print_2d_array(arr, rows, cols);              //печатаем 2d массив
}

void generate_elements_in_2d_array(
    const std::unique_ptr<std::unique_ptr<int[]>[]> &arr,
    const int rows,
    const int cols,
    int a,
    int b
)
{
    for (int i = 0; i < rows; i++)
    {
        for (int j = 0; j < cols; j++)
        {
            arr[i][j] = rand() % (b - a + 1) + a;//рандом от a до b
        }
    }
}

template <typename T>
void print_2d_array(
    const std::unique_ptr<std::unique_ptr<T[]>[]> &arr,
    const int rows,
    const int cols
)
{
    for (int i = 0; i < rows; i++)
    {
        for (int j = 0; j < cols; j++)
        {
            std::cout << arr[i][j] << "\t";      //печатаем элементы через табуляцию
        }
        std::cout << std::endl;                  //новая строка массива
    }
    std::cout << std::endl; //чтобы матрицы не слиплись, вызвав функцию два раза
}

template <typename T>
void add_line_to_2d_array(
    std::unique_ptr<std::unique_ptr<T[]>[]> &arr,
    int *rows,
    const int cols
)
{
    *rows += 1;                                  //увеличиваем количество строк
    std::unique_ptr<
        std::unique_ptr<T[]>[]
    > temp(std::make_unique<std::unique_ptr<T[]>[]>(*rows));     //память под 2d массив
    for (int i = 0; i < *rows; i++)              //заполняем массив массивами
```

```
    {
        temp[i] = std::make_unique<T[]>(cols);   //в ячейке массива массив
    }

    for (int i = 1; i < *rows; i++)
    {
        for (int j = 0; j < cols; j++)
        {
            temp[i][j] = arr[i - 1][j];          //переносим элементы из массива в новый массив
        }
    }

    arr = move(temp);                            //передаём права
}
```

Листинг: Out

```
12      24      45      39      71      29      47      74      95      8       10      14
27      18      82      71      92      25      4       7       83      3       11      12
37      17      44      28      54      21      93      66      42      38      3       13
66      48      85      59      55      94      72      79      11      52      51      3
77      54      7       58      54      17      68      90      31      10      18      85

0       0       0       0       0       0       0       0       0       0       0       0
12      24      45      39      71      29      47      74      95      8       10      14
27      18      82      71      92      25      4       7       83      3       11      12
37      17      44      28      54      21      93      66      42      38      3       13
66      48      85      59      55      94      72      79      11      52      51      3
77      54      7       58      54      17      68      90      31      10      18      85
```

Вариант V задание 3

Условие: Двумерный массив. Для матрицы размером NxM вывести на экран все седловые точки. Элемент матрицы называется седловой точкой, если он является наименьшим в своей строке и одновременно наибольшим в своем столбце или наоборот.

Решение:

Листинг: main.cpp

```
#include <iostream>
#include <memory>

//функция варианта 5 задания 3
void option5_task3();
//функция сгенерирует элементы для int массива
void generate_elements_in_2d_array(
    const std::unique_ptr<std::unique_ptr<int[]>[]> &arr,
    const int rows,
    const int cols,
    int a = 1,
    int b = 10
);
//функция напечатает функцию
template <typename T>
void print_2d_array(
    const std::unique_ptr<std::unique_ptr<T[]>[]> &arr,
    const int rows,
```

```
    const int cols
);
//печатает седловые точки
template <typename T>
void printf_saddle_points(
    const std::unique_ptr<std::unique_ptr<T[]>[]> &arr,
    const int rows,
    const int cols
);
//если value равен минимальному элементу по колонке, то функция возвращает true, иначе false
template <typename T>
bool is_min_in_col_2d_array(
    const std::unique_ptr<std::unique_ptr<T[]>[]> &arr,
    const int rows,
    const int col_number,
    const T value
);
//если value равен максимальному элементу по колонке, то функция возвращает true, иначе false
template <typename T>
bool is_max_in_col_2d_array(
    const std::unique_ptr<std::unique_ptr<T[]>[]> &arr,
    const int rows,
    const int col_number,
    const T value
);
//если value равен минимальному элементу по строке, то функция возвращает true, иначе false
template <typename T>
bool is_min_in_row_2d_array(
    const std::unique_ptr<std::unique_ptr<T[]>[]> &arr,
    const int row_number,
    const int cols,
    const T value
);
//если value равен минимальному элементу по строке, то функция возвращает true, иначе false
template <typename T>
bool is_max_in_row_2d_array(
    const std::unique_ptr<std::unique_ptr<T[]>[]> &arr,
    const int row_number,
    const int cols,
    const T value
);

int main()
{
    srand(1);   //Seed рандома. Программа всегда будет давать один и тот же результат при рандоме
    option5_task3();

    return 0;
}

void option5_task3()
{
    int rows = 3;                                   //количество строк
    int cols = 7;                                   //количество столбцов
    std::unique_ptr<
        std::unique_ptr<int[]>[]
    > arr( std::make_unique<std::unique_ptr<int[]>[]>(rows) );  //выделение памяти под 2d массив
    for (int i = 0; i < rows; i++)                  //заполняем одномерный массив массивом
    {
        arr[i] = std::make_unique<int[]>(cols); //в ячейке массива массив
```

```
    }

    generate_elements_in_2d_array(arr, rows, cols);//заполняем массив
    print_2d_array(arr, rows, cols);            //печатаем 2d массив
    printf_saddle_points(arr, rows, cols);
}

void generate_elements_in_2d_array(
    const std::unique_ptr<std::unique_ptr<int[]>[]> &arr,
    const int rows,
    const int cols,
    int a,
    int b
)
{
    for (int i = 0; i < rows; i++)
    {
        for (int j = 0; j < cols; j++)
        {
            arr[i][j] = rand() % (b - a + 1) + a;//рандом от a до b
        }
    }
}

template <typename T>
void print_2d_array(
    const std::unique_ptr<std::unique_ptr<T[]>[]> &arr,
    const int rows,
    const int cols
)
{
    for (int i = 0; i < rows; i++)
    {
        for (int j = 0; j < cols; j++)
        {
            std::cout << arr[i][j] << "\t";      //печатаем элементы через табуляцию
        }
        std::cout << std::endl;                  //новая строка массива
    }
    std::cout << std::endl; //чтобы матрицы не слиплись, вызвав функцию два раза
}

template <typename T>
void printf_saddle_points(
    const std::unique_ptr<std::unique_ptr<T[]>[]> &arr,
    const int rows,
    const int cols
)
{
    for (int i = 0; i < rows; i++)
    {
        for (int j = 0; j < cols; j++)
        {
            if (
                is_min_in_col_2d_array(arr, rows, j, arr[i][j])
                &&
                is_max_in_row_2d_array(arr, i, cols, arr[i][j])
                ||
                is_max_in_col_2d_array(arr, rows, j, arr[i][j])
                &&
```

```
                is_min_in_row_2d_array(arr, i, cols, arr[i][j])
            )
            {
                std::cout << "arr[" << i << "][" << j << "] = " << arr[i][j] << std::endl;
            }
        }
    }
}

template <typename T>
bool is_min_in_col_2d_array(
    const std::unique_ptr<std::unique_ptr<T[]>[]> &arr,
    const int rows,
    const int col_number,
    const T value
)
{
    T min = arr[0][col_number];
    for (int i = 0; i < rows; i++)              //поиск минимального в колонке
    {
        if(arr[i][col_number] < min)
        {
            min = arr[i][col_number];
        }
    }
    return (value == min);
}

template <typename T>
bool is_max_in_col_2d_array(
    const std::unique_ptr<std::unique_ptr<T[]>[]> &arr,
    const int rows,
    const int col_number,
    const T value
)
{
    T max = arr[0][col_number];
    for (int i = 0; i < rows; i++)              //поиск максимального в колонке
    {
        if(arr[i][col_number] > max)
        {
            max = arr[i][col_number];
        }
    }
    return (value == max);
}

template <typename T>
bool is_min_in_row_2d_array(
    const std::unique_ptr<std::unique_ptr<T[]>[]> &arr,
    const int row_number,
    const int cols,
    const T value
)
{
    T min = arr[row_number][0];
    for (int j = 0; j < cols; j++)              //поиск минимального в строке
    {
        if (arr[row_number][j] < min)
        {
```

```
            min = arr[row_number][j];
        }
    }
    return (value == min);
}

template <typename T>
bool is_max_in_row_2d_array(
    const std::unique_ptr<std::unique_ptr<T[]>[]> &arr,
    const int row_number,
    const int cols,
    const T value
)
{
    T max = arr[row_number][0];
    for (int j = 0; j < cols; j++)              //поиск максимального в строке
    {
        if (arr[row_number][j] > max)
        {
            max = arr[row_number][j];
        }
    }
    return (value == max);
}
```

Листинг: Out

```
4       7       8       6       4       6       7
3       10      2       3       8       1       10
4       7       1       7       3       7       2

arr[0][0] = 4
arr[2][1] = 7
```