Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования «Московский государственный технический университет имени Н. Э. Баумана (национальный исследовательский университет)

Факультет «Робототехника и комплексная автоматизация»

Кафедра «Системы автоматизированного проектирования»

**Отчет по лабораторной работе**

По курсу «Объектно-ориентированное программирование»

Выполнил: Студент Петраков С.А.

Группа РК6-26Б

Проверил: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Подпись\_\_\_\_\_\_\_\_\_\_\_\_\_

Москва, 2020 г.

**Вариант ПЯВУ М16**

**Задание:**

Разработать объектно-ориентированную программу для построения магического квадрата любого нечетного порядка из последовательных натуральных чисел по методу альфила в следующем варианте. Начальное значение 1 нужно записать под клеткой в правом верхнем углу квадрата. Запись чисел в остальные клетки должна происходить налево и вниз по диагонали через 1 клетку. Если эта клетка уже занята, то следующее число нужно записать на 1 клетку левее и на 3 клетки ниже предыдущего. Когда число оказывается за границами квадрата, его следует перенести внутрь квадрата, изменив заграничную координату на порядок квадрата. Результат построения магического квадрата по таким правилам должен отображаться в поток стандартного вывода. Значение порядка квадрата должно передаваться программе аргументом командной строки. При разработке программы следует реализовать базовый класс квадратной матрицы и производный от него класс магического квадрата. Конструктор базового класса должен обеспечивать динамическое распределение памяти по защищенному адресу для двумерного массива. Компонентные методы для его заполнения и отображения следует реализовать в производном классе.

**Алгоритм:**

Для реализации данной программы я создал класс, в котором будет хранится данный магический квадрат и методы вывода, заполнения.

Для вывода проходимся по всем элементам, когда новая строка переходим на след строку.

Для заполнения встаем на клетку под правым верхним углом и сдвигаемся по координатам на (x-1; y-1). В случае выхода за пределы квадрата одной из координат прибавляем к ней порядок квадрата. Для проверки выхода создан приватный метод, который будет это проверять. В случае если в данной клетке уже будет число сдвинемся на (x-1; y-3) от изначального значения.

В итоге выводим магический квадрат с помощью метода вывода.

**Входные данные:**

1 значение - Порядок квадрата.

**Выходные данные:**

Магический квадрат

**Текст программы:**

#include <iostream>

#include <stdlib.h>

#include <math.h>

using namespace std;

class Magic

{

private:

unsigned int\*\* \_table;

int \_degree;

int reflect(int n);

public:

Magic(int);

~Magic();

void print();

void fill();

};

Magic::Magic(int n)

{

\_degree = n;

\_table = new unsigned int\* [\_degree];

for (int i = 0; i < \_degree; i++)

\_table[i] = new unsigned int[\_degree];

for (int i = 0; i < \_degree; i++)

for (int j = 0; j < \_degree; j++)

\_table[i][j]=0;

}

Magic::~Magic()

{

for (int i = 0; i < \_degree; i++)

delete[] \_table[i];

delete[] \_table;

}

void Magic::print()

{

int degree2 = \_degree\*\_degree;

int len = 0;

while (degree2 > 0)

{

len++;

degree2 /= 10;

}

for (int i = 0; i < \_degree; i++, cout << endl)

for (int j = 0; j < \_degree; j++)

printf("%\*d ",len,\_table[i][j]);

cout << endl;

}

void Magic::fill()

{

int curr = 1;

int degree2 = \_degree \* \_degree;

int i, j;

int col = \_degree-1;

int row = 1;

\_table[row][col] = curr;

while (curr < degree2)

{

i = reflect(row + 2);

j = reflect(col - 2);

if (\_table[i][j] != 0)

{

i = reflect(row +3);

j = reflect(col -1);

}

row = i;

col = j;

\_table[row][col] = ++curr;

cout << "Iteration " << curr << ":\n";

print();

}

}

int Magic::reflect(int n)

{

if (n < 0)

return (n + \_degree);

if (n > (\_degree - 1))

return (n - \_degree);

return n;

}

int main(int argc, char\* argv[])

{

if (argc != 2)

{

cout << "Invalid count argument.\n";

exit(1);

}

int degree = atoi(argv[1]);

cout << "Usage degree of magic square: " << degree << ".\n";

Magic result(degree);

result.fill();

cout << "Final result.\n";

result.print();

return 0;

}

**Тесты:**

|  |  |  |
| --- | --- | --- |
| **Номер** | **Вход** | **Выход** |
| 1 | 5 | |  |  |  |  |  | | --- | --- | --- | --- | --- | | 3 | 19 | 10 | 21 | 12 | | 17 | 8 | 24 | 15 | 1 | | 6 | 22 | 13 | 4 | 20 | | 25 | 11 | 2 | 18 | 9 | | 14 | 5 | 16 | 7 | 23 | |
| 2 | 3 | |  |  |  | | --- | --- | --- | | 2 | 7 | 6 | | 9 | 5 | 1 | | 4 | 3 | 8 | |

**Список использованной литературы:**

* Волосатова Т.М., Родионов С.В. Лекции по курсу «Объектно-ориентированное программирование»
* bigor.bmstu.ru