Разработать ООП для построения магического квадрата из последовательных натуральных чисел по методу коня в следующем варианте. Начальное значение 1 нужно записать в среднюю клетку нижней строки квадрата. Запись чисел в остальные клетки должна происходить по ходу шахматного коня на 2 клетки вверх и на 1 клетку вправо. Если эта клетка уже занята, то следующее число нужно записать на 1 клетку ниже, чем предыдущее. Когда число оказывается за границей квадрата, его необходимо перенести внутрь квадрата, изменив заграничную координату на порядок квадрата. Результат построения квадрата должен быть отображен через стандартный вывод. Значение порядка квадрата должно передаваться программе через аргумент командной строки. При разработке программы нужно реализовать класс магического квадрата с приватными полями для матрицы и порядка квадрата, а также публичными методами его заполнения и отображения. Конструктор и деструктор класса должны обеспечивать динамическое распределение памяти квадрата.

1: #include <stdlib.h>

2: #include <stdio.h>

3:

4: class Magic

5: {

6: private:

7: unsigned\*\* tab;

8: int row;

9: int col;

10: int degree;

11: public:

12: Magic(int);

13: ~Magic();

14: void print();

15: void horse3();

16: int reflect(int);

17: };

18:

19: Magic::Magic(int n)

20: {

21: degree = n;

22: tab = new unsigned\* [degree];

23: for(row=0; row < degree; row++)

24: tab[row] = new unsigned [degree];

25: for(row=0; row < degree; row++)

26: for(col=0; col < degree; col++)

27: tab[row][col] = 0;

28: }

29:

30: Magic::~Magic()

31: {

32: for(row=0; row < degree; row++)

33: delete [] tab[row];

34: delete []tab;

35: }

36:

37: void Magic::print()

38: {

39: int degree2;

40: int len=0;

41: degree2 = (degree \* degree);

42: while(degree2 > 0)

43: {

44: degree2 /= 10;

45: len++;

46: }

47: for(row=0; row < degree; row++)

48: {

49: for(col=0; col < degree; col++)

50: printf("%0\*d ", len, tab[row][col]);

51: putchar('\n');

52: }

53: putchar('\n');

54: return;

55: }

56:

57: int Magic::reflect(int k)

58: {

59: if(k < 0)

60: return(k + degree);

61: if(k > (degree - 1))

62: return(k - degree);

63: return(k);

64: }

65:

66: void Magic::horse3()

67: {

68: int i;

69: int j;

70: int degree2 = (degree\*degree);

71: int z = 1;

72: row = (degree - 1);

73: col = (degree / 2);

74: tab[row][col] = z;

75: while(z < degree2)

76: {

77: i = reflect(row - 2);

78: j = reflect(col + 1);

79: if(tab[i][j] > 0)

80: {

81: i = reflect(row + 1);

82: j = reflect(col);

83: }

84: row = i;

85: col = j;

86: ++z;

87: tab[row][col] = z;

88: }

89: return;

90: }

91:

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

93: {

94: if(argc != 2)

95: return(puts("Usage: magic degree"));

96: int n = atoi(argv[1]);

97: if((n % 2) == 0)

98: {

99: puts("Usage: magic 5 (or 7, 11, 17, 19, 23, ...)");

100: return(n);

101: }

102: Magic mag(n);

103: mag.horse3();

104: mag.print();

105: return(n);

106: }