

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное автономное образовательное учреждение высшего образования
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

ИНСТИТУТ НЕПРЕРЫВНОГО И ДИСТАНЦИОННОГО ОБРАЗОВАНИЯ

КАФЕДРА КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ И ПРОГРАММНОЙ ИНЖЕНЕРИИ

ОЦЕНКА

ПРЕПОДАВАТЕЛЬ

старший преподаватель		Е.О. Шумова
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №4

Наследование классов, базовый класс, производный класс

по дисциплине: Объектно-ориентированное программирование

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ гр. №	Z1431		М.Д. Быстров
	номер группы	подпись, дата	инициалы, фамилия

Студенческий билет №	2021/3572
----------------------	-----------

Санкт-Петербург 2023

Условие

Цель работы: изучить механизм создания нового класса на основе уже существующего, варианты доступа к элементам базового класса из производного.

Закрепить знания по теме: классы, наследование классов, варианты доступа.

Описание работы: в работе необходимо реализовать базовый класс заданной структуры, на основе него создать производные классы. В нём предусмотреть конструктор для установки начальных значений полей. Создать объекты производных классов. Продемонстрировать работу всех методов, реализуемых в классах.

Индивидуальное задание (вариант 2):

2. Создать класс Points для хранения координат четырёх точек A, B, C и D на плоскости. В классе предусмотреть возможность распечатки координат каждой точки по отдельности и всех разом. На основе класса Points создать класс Quadrilateral для работы с четырёхугольником. Предусмотреть методы для проверки существования четырёхугольника, нахождения площади и диагоналей.

Полный текст (листинг) программы

1. Файл «main.cpp»

```
1. #include <iostream>
2. #include "Quadrilateral.h"
3. #include "main.h"
4.
5. #define QUADR_COORDS 1,0, 2,1, 5,5, 1,2
6. #define INVALID_COORDS 0,0, 1,1, 0,1, 1,0
7.
8. using namespace std;
9.
10. void CheckQuadrilateral(Quadrilateral* quadr)
11. {
12.     cout << "Quadrilateral " << quadr->getAllPointsStr() << endl << endl;
13.
14.     if (quadr->isReal())
15.     {
16.         cout << "Square: " << quadr->getSquare() << endl;
17.         cout << "Diagonal AC" << quadr->getAStr() << quadr->getCStr() << ":
18.         " << quadr->getACLength() << endl;
19.         cout << "Diagonal BD" << quadr->getBStr() << quadr->getDStr() << ":
20.         " << quadr->getBDLength() << endl;
21.     }
22.     else
23.     {
24.         cout << "Quadrilateral does not exist" << endl;
25.     }
26.     cout << endl;
27.     delete quadr;
28. }
29.
30. int main()
31. {
32.     Quadrilateral* quadr1 = new Quadrilateral(QUADR_COORDS);
33.     Quadrilateral* quadr2 = new Quadrilateral(INVALID_COORDS);
34.
35.     CheckQuadrilateral(quadr1);
36.     CheckQuadrilateral(quadr2);
37. }
```

2. Файл «Points.h»

```

1. #include <string>
2.
3. #pragma once
4. class Points
5. {
6. private:
7.
8.     int ax;
9.     int ay;
10.
11.     int bx;
12.     int by;
13.
14.     int cx;
15.     int cy;
16.
17.     int dx;
18.     int dy;
19.
20. public:
21.
22.     Points(int x1, int y1, int x2, int y2, int x3, int y3, int x4, int
        y4);
23.     ~Points();
24.
25.     int getAX() const;
26.     void setAX(int x1);
27.
28.     int getAY() const;
29.     void setAY(int y1);
30.
31.     int getBX() const;
32.     void setBX(int x2);
33.
34.     int getBY() const;
35.     void setBY(int y2);
36.
37.     int getCX() const;
38.     void setCX(int x3);
39.
40.     int getCY() const;
41.     void setCY(int y3);
42.
43.     int getDX() const;
44.     void setDX(int x4);
45.
46.     int getDY() const;
47.     void setDY(int y4);
48.
49.     std::string getAllPointsStr() const;

```

```

50.
51.         std::string getAStr() const;
52.         std::string getBStr() const;
53.         std::string getCStr() const;
54.         std::string getDStr() const;
55. };

```

3. Файл «Points.cpp»

```

1. #include <stdlib.h>
2. #include <utility>
3. #include <sstream>
4. #include <iostream>
5. #include "Points.h"
6.
7. using namespace std;
8.
9. Points::Points(int x1, int y1, int x2, int y2, int x3, int y3, int x4, int
    y4)
10. {
11.     this->ax = x1;
12.     this->ay = y1;
13.
14.     this->bx = x2;
15.     this->by = y2;
16.
17.     this->cx = x3;
18.     this->cy = y3;
19.
20.     this->dx = x4;
21.     this->dy = y4;
22. }
23.
24. Points::~Points()
25. {
26.     cout << "Desctruction of Points " << this->getAllPointsStr() << endl;
27. }
28.
29. int Points::getAX() const { return this->ax; }
30. void Points::setAX(int ax) { this->ax = ax; }
31. int Points::getAY() const { return this->ay; }
32. void Points::setAY(int ay) { this->ay = ay; }
33.
34. int Points::getBX() const { return this->bx; }
35. void Points::setBX(int bx) { this->bx = bx; }
36. int Points::getBY() const { return this->by; }

```

```

37. void Points::setBY(int by) { this->by = by; }
38.
39. int Points::getCX() const { return this->cx; }
40. void Points::setCX(int cx) { this->cx = cx; }
41. int Points::getCY() const { return this->cy; }
42. void Points::setCY(int cy) { this->cy = cy; }
43.
44. int Points::getDX() const { return this->dx; }
45. void Points::setDX(int dx) { this->dx = dx; }
46. int Points::getDY() const { return this->dy; }
47. void Points::setDY(int dy) { this->dy = dy; }
48.
49. string Points::getAllPointsStr() const
50. {
51.     return this->getAStr() + this->getBStr() + this->getCStr() + this-
        >getDStr();
52. }
53.
54. string coordsToStr(int a, int b)
55. {
56.     std::stringstream ss;
57.
58.     ss << "[" << a << ";" << b << "];"
59.
60.     return ss.str();
61. }
62.
63. string Points::getAStr() const
64. {
65.     return coordsToStr(this->ax, this->ay);
66. };
67.
68. string Points::getBStr() const
69. {
70.     return coordsToStr(this->bx, by);
71. };
72.
73. string Points::getCStr() const
74. {
75.     return coordsToStr(this->cx, this->cy);
76. };
77.
78. string Points::getDStr() const
79. {
80.     return coordsToStr(this->dx, this->dy);
81. }

```

4. Файл «Quadrilateral.h»

```

1. #pragma once
2. #include "Points.h"
3. class Quadrilateral :
4.     public Points
5. {
6. public:
7.
8.     Quadrilateral(int x1, int y1, int x2, int y2, int x3, int y3, int x4,
9.         int y4);
10.     ~Quadrilateral();
11.
12.     double getSquare() const;
13.
14.     double getACLength() const;
15.
16.     double getBDLength() const;
17.
18.     bool isReal() const;
19. };
20.

```

5. Файл «Quadrilateral.cpp»

```

1. #include <stdlib.h>
2. #include <utility>
3. #include <stdexcept>
4. #include <iostream>
5. #include "Quadrilateral.h"
6.
7. using namespace std;
8.
9. inline int area(int ax, int ay, int bx, int by, int cx, int cy)
10. {
11.     return (bx - ax) * (cy - ay) - (by - ay) * (cx - ax);
12. }
13.
14. inline bool intersect_1(int a, int b, int c, int d)
15. {
16.     if (a > b)
17.     {
18.         swap(a, b);
19.     }
20.
21.     if (c > d)
22.     {
23.         swap(c, d);

```

```

24.     }
25.
26.     return max(a, c) <= min(b, d);
27.}
28.
29.bool intersect(int ax, int ay, int bx, int by, int cx, int cy, int dx, int
    dy) {
30.    return intersect_1(ax, bx, cx, dx)
31.        && intersect_1(ay, by, cy, dy)
32.        && area(ax, ay, bx, by, cx, by) * area(ax, ay, bx, by, dx, dy)
    <= 0
33.        && area(cx, cy, dx, dy, ax, ay) * area(cx, cy, dx, dy, bx, by)
    <= 0;
34.}
35.
36.Quadrilateral::Quadrilateral(int x1, int y1, int x2, int y2, int x3, int
    y3, int x4, int y4) :
37.    Points(x1, y1, x2, y2, x3, y3, x4, y4)
38.{
39.}
40.
41.Quadrilateral::~~Quadrilateral()
42.{
43.    cout << "Desctruction of Quadrilateral " << this->getAllPointsStr()
    << endl;
44.}
45.
46.double Quadrilateral::getSquare() const
47.{
48.    const int topsCount = 4;
49.
50.    int xArray[topsCount] = {
51.        this->getAX(),
52.        this->getBX(),
53.        this->getCX(),
54.        this->getDX()};
55.
56.    int yArray[topsCount] = {
57.        this->getAY(),
58.        this->getBY(),
59.        this->getCY(),
60.        this->getDY() };
61.
62.    int sum = 0;
63.
64.    for (int i = 0; i < topsCount; i++)
65.    {
66.        sum += xArray[i] * yArray[(i + 1) % topsCount];
67.        sum -= xArray[(i + 1) % topsCount] * yArray[i];
68.    }

```

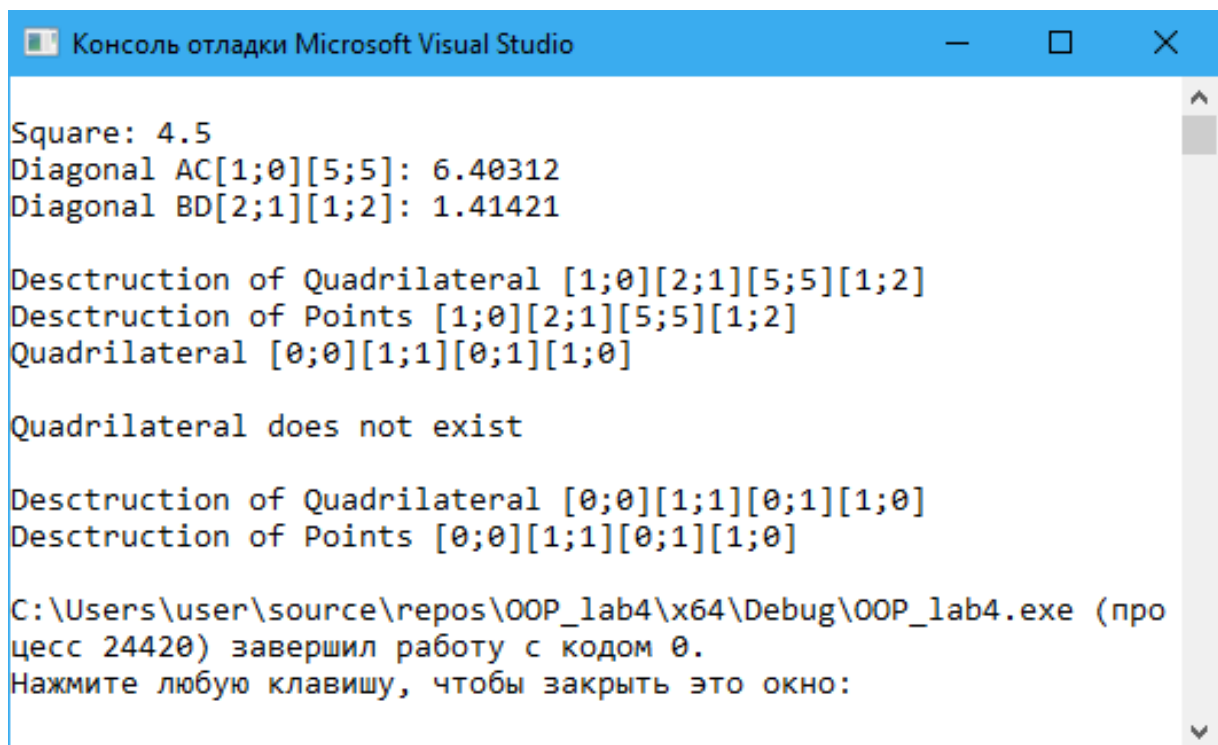


```

69.
70.     sum = abs(sum) ;
71.
72.     double square = (double)sum / (double)2;
73.
74.     return square;
75.}
76.
77.double hypo(int x, int y)
78.{
79.     double sumOfCatets = pow(x, 2) + pow(y, 2);
80.
81.     double ret = sqrt(sumOfCatets);
82.
83.     return ret;
84.}
85.
86.double Quadrilateral::getACLength() const
87.{
88.     return hypo(
89.         abs(this->getAX() - this->getCX()),
90.         abs(this->getAY() - this->getCY()));
91.}
92.
93.double Quadrilateral::getBDLength() const
94.{
95.     return hypo(
96.         abs(this->getBX() - this->getDX()),
97.         abs(this->getBY() - this->getDY()));
98.}
99.
100.    bool Quadrilateral::isReal() const
101.    {
102.        return !intersect(
103.            this->getAX(),
104.            this->getAY(),
105.            this->getBX(),
106.            this->getBY(),
107.            this->getCX(),
108.            this->getCY(),
109.            this->getDX(),
110.            this->getDY());
111.    }

```

Работа программы



Консоль отладки Microsoft Visual Studio

```
Square: 4.5  
Diagonal AC[1;0][5;5]: 6.40312  
Diagonal BD[2;1][1;2]: 1.41421  
  
Desctruction of Quadrilateral [1;0][2;1][5;5][1;2]  
Desctruction of Points [1;0][2;1][5;5][1;2]  
Quadrilateral [0;0][1;1][0;1][1;0]  
  
Quadrilateral does not exist  
  
Desctruction of Quadrilateral [0;0][1;1][0;1][1;0]  
Desctruction of Points [0;0][1;1][0;1][1;0]  
  
C:\Users\user\source\repos\OOP_lab4\x64\Debug\OOP_lab4.exe (про  
цесс 24420) завершил работу с кодом 0.  
Нажмите любую клавишу, чтобы закрыть это окно:
```

Рисунок 1 Работа программы

Выводы

В ходе выполнения лабораторной работы №4 получены навыки конструирования классов на основе базовых классов.

Создан производный класс. Продемонстрировано использование конструкторов базового класса в производном. Показано использование функциональности базового класса при работе с объектом производного класса. В консоль выведены сообщения о работе деструкторов, из которых можно определить порядок вызова деструкторов при освобождении памяти от объекта, участвующего в иерархии наследования.

Продемонстрирована работа всех методов, реализованных в классах.