Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)
Кафедра экономической математики, информатики и статистики (ЭМИС)

ВИРТУАЛЬНЫЕ ФУНКЦИИ И АБСТРАКТНЫЕ БАЗОВЫЕ КЛАССЫ

Отчет по лабораторной работе по дисциплине «Объектно-ориентированное программирование»

Студент группы 549

 Баулин С.К.

«__» _____ 2020 г.

Кандидат физико-математических наук, доцент кафедры ЭМИС

_____ _____ Шельмина Е. А.

оценка «__» _____ 2020 г.

Томск 2020

Лабораторная работа №8

Виртуальные функции и абстрактные базовые классы

Цель работы: освоить и применить на практике виртуальные функции и абстрактные базовые классы.

Теоритические сведения

Виртуальные функции

К механизму виртуальных функций обращаются в тех случаях, когда в каждом производном классе требуется свой вариант некоторой компонентной функции. Классы, включающие такие функции, называются полиморфными и играют особую роль в ООП.

Виртуальные функции предоставляют механизм позднего (отложенного) или динамического связывания. Любая нестатическая функция базового класса может быть сделана виртуальной, для чего используется ключевое слово virtual.

Таким образом, интерпретация каждого вызова виртуальной функции через указатель на базовый класс зависит от значения этого указателя, т.е. от типа объекта, для которого выполняется вызов.

Выбор того, какую виртуальную функцию вызвать, будет зависеть от типа объекта, на который фактически (в момент выполнения программы) направлен указатель, а не от типа указателя.

Виртуальными могут быть только нестатические функции-члены.

Виртуальность наследуется. После того как функция определена как виртуальная, ее повторное определение в производном классе (с тем же самым прототипом) создает в этом классе новую виртуальную функцию, причем спецификатор virtual может не использоваться.

Конструкторы не могут быть виртуальными, в отличие от деструкторов. Практически каждый класс, имеющий виртуальную функцию, должен иметь виртуальный деструктор.

Абстрактные классы

Абстрактным называется класс, в котором есть хотя бы одна чистая (пустая) виртуальная функция.

Чистой виртуальной функцией называется компонентная функция, которая имеет следующее определение:

virtual тип имя_функции (список_формальных_параметров) = 0;

Чистая виртуальная функция ничего не делает и недоступна для вызовов. Ее назначение – служить основой для подменяющих ее функций в производных классах. Абстрактный класс может использоваться только в качестве базового для производных классов.

Механизм абстрактных классов разработан для представления общих понятий, которые в дальнейшем предполагается конкретизировать. При этом построение иерархии классов выполняется по следующей схеме. Во главе иерархии стоит абстрактный базовый класс. Он используется для наследования интерфейса. Производные классы будут конкретизировать и реализовать этот интерфейс. В абстрактном классе объявлены чистые виртуальные функции, которые по сути есть абстрактные методы.

Объект абстрактного класса не может быть формальным параметром функции, однако формальным параметром может быть указатель на абстрактный класс. В этом случае появляется возможность передавать в вызываемую функцию в качестве фактического параметра значение указателя на производный объект, заменяя им указатель на абстрактный базовый класс. Таким образом, мы получаем полиморфные объекты.

Абстрактный метод может рассматриваться как обобщение переопределения. В обоих случаях поведение родительского класса изменяется для потомка. Для абстрактного метода, однако, поведение просто не определено. Любое поведение задается в производном классе.

Одно из преимуществ абстрактного метода является чисто концептуальным: программист может мысленно наделить нужным действием абстракцию сколь угодно высокого уровня. Например, для геометрических

фигур мы можем определить метод Draw, который их рисует: треугольник TTriangle, окружность TCircle, квадрат TSquare. Мы определим аналогичный метод и для абстрактного родительского класса TGraphObject. Однако такой метод не может выполнять полезную работу, поскольку в классе TGraphObject просто нет достаточной информации для рисования чего бы то ни было. Тем не менее присутствие метода Draw позволяет связать функциональность (рисование) только один раз с классом TGraphObject, а не вводить три независимые концепции для подклассов TTriangle, TCircle, TSquare.

Имеется и вторая, более актуальная причина использования абстрактного метода. В объектно-ориентированных языках программирования со статическими типами данных, к которым относится и C++, программист может вызвать метод класса, только если компилятор может определить, что класс действительно имеет такой метод. Предположим, что программист хочет определить полиморфную переменную типа TGraphObject, которая будет в различные моменты времени содержать фигуры различного типа. Это допустимо для полиморфных объектов. Тем не менее компилятор разрешит использовать метод Draw для переменной, только если он сможет гарантировать, что в классе переменной имеется этот метод. Присоединение метода Draw к классу TGraphObject обеспечивает такую гарантию, даже если метод Draw для класса TGraphObject никогда не выполняется. Естественно, для того чтобы каждая фигура рисовалась по-своему, метод Draw должен быть виртуальным.

Задания

Задание 1. Разработать программу с использованием наследования классов, реализующую классы: железнодорожный вагон, вагон для перевозки автомобилей, цистерна.

Используя виртуальные функции, выведите на экран вес железнодорожного вагона и количество единиц товара в вагоне.

Скриншоты кода программы и результат представлены на рисунках 1.1 – 1.3.

```
6   class Train
7   {
8   protected:
9       string type;
10      long weight;
11      int number;
12
13  public:
14      Train()
15      {
16          type = "";
17          weight = 0;
18          number = 0;
19      }
20      Train(int w, int n)
21      {
22          weight = w;
23          number = n;
24      }
25      virtual void show()
26      {
27          cout << "-----------------------" << endl
28               << type << endl
29               << "Вес: " << weight << endl
30               << "Кол-во единиц товара: " << number << endl
31               << "-----------------------" << endl;
32      }
33  };
```

Рисунок 1.1 – Скриншот кода программы

```
35  class CarTrain : public Train
36  {
37  public:
38      CarTrain() : Train()
39      {
40          type = "Вагон для перевозки автомобилей";
41          weight = 3000000;
42          number = 0;
43      }
44      CarTrain(long w, int n) : Train(w, n)
45      {
46          type = "Вагон для перевозки автомобилей";
47      }
48      void show() override
49      {
50          cout << "------------------------" << endl
51               << type << endl
52               << "Вес с автомобилями: " << weight << endl
53               << "Кол-во автомобилей: " << number << endl
54               << "------------------------" << endl;
55      }
56  };
57
    You, 13 days ago | 1 author (You)
58  class TankerTrain : public Train
59  {
60  public:
61      TankerTrain() : Train()
62      {
63          type = "Вагон - цистерна";
64          weight = 800000;
65          number = 0;
66      }
67      TankerTrain(long w, int n) : Train(w, n)
68      {
69          type = "Вагон - цистерна";
70      }
71      void show() override
72      {
73          cout << "------------------------" << endl
74               << type << endl
75               << "Вес с содержимым: " << weight << endl
76               << "Объем: " << number << " л." << endl
77               << "------------------------" << endl;
78      }
79  };
```

Рисунок 1.2 – Скриншот кода программы

```
81  int main()
82  {
83      setlocale(LC_ALL, "65001");
84      system("chcp 65001");
85      system("cls");
86      Train *n1 = new CarTrain(5000000, 3);
87      Train *n2 = new TankerTrain(2500000, 12);
88      n1->show();
89      n2->show();
90
91      return 0;
92  }
```

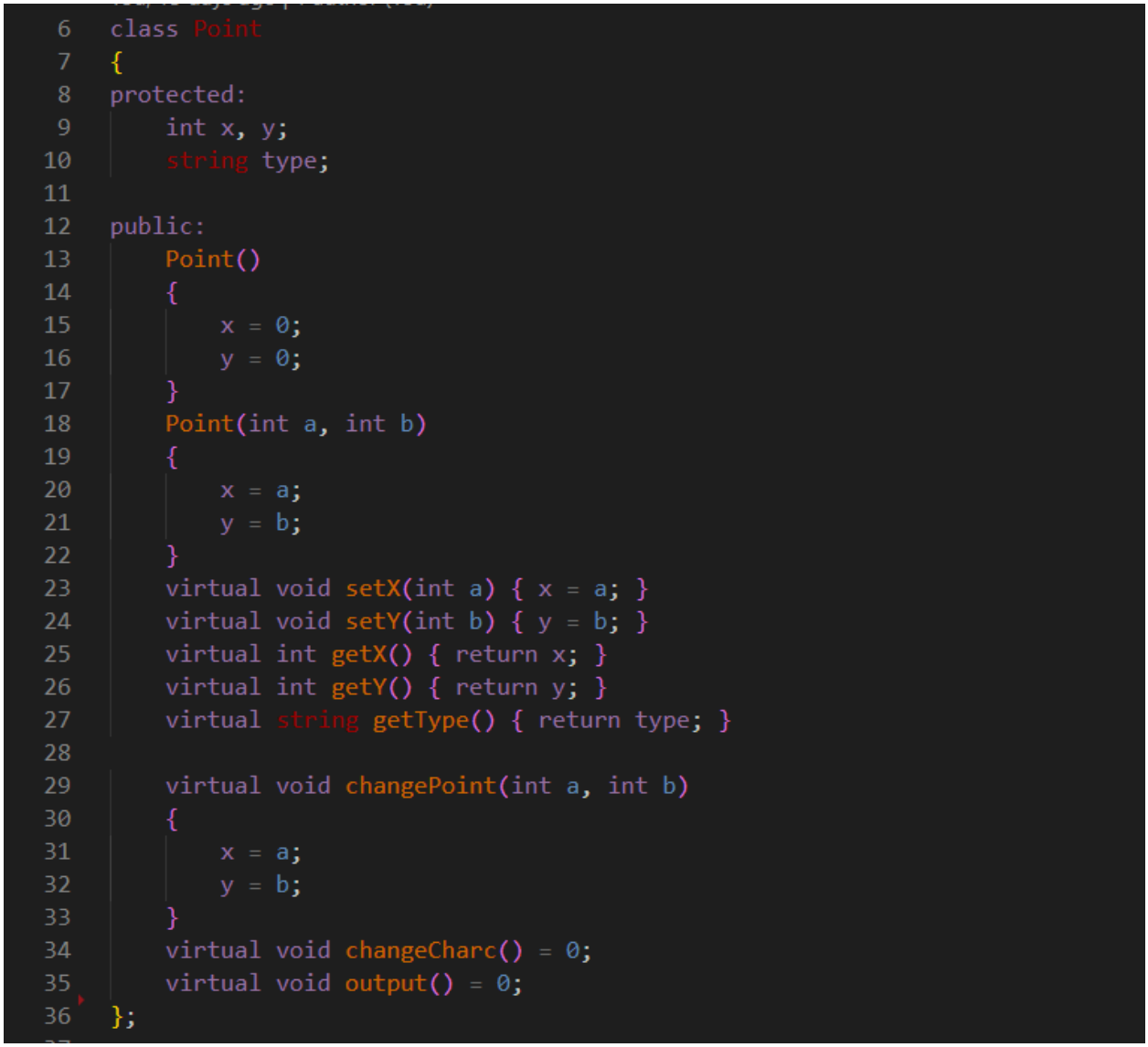
```
ПРОБЛЕМЫ  ВЫХОДНЫЕ ДАННЫЕ  ТЕРМИНАЛ  КОНСОЛЬ ОТЛАДКИ

----------------------
Вагон для перевозки автомобилей
Вес с автомобилями: 5000000
Кол-во автомобилей: 3
----------------------
----------------------
Вагон - цистерна
Вес с содержимым: 2500000
Объем: 12 л.
----------------------
```

Рисунок 1.3 – Скриншот кода и результата работы программы

Задание 2. Создать абстрактный класс Point (точка). На его основе создать классы ColoredPoint и Line. На основе класса Line создать класс ColoredLine и класс PolyLine (многоугольник). Все классы должны иметь виртуальные методы установки и получения значений всех координат, а также изменения цвета и получения текущего цвета. Создать класс Picture, содержащий массив объектов этих классов в динамической памяти. Предусмотреть возможность вывода характеристик объектов списка. Написать демонстрационную программу, в которой будут использоваться все методы классов.

Скриншоты кода программы и результат представлены на рисунках 2.1 – 2.6.



```
6   class Point
7   {
8   protected:
9       int x, y;
10      string type;
11
12  public:
13      Point()
14      {
15          x = 0;
16          y = 0;
17      }
18      Point(int a, int b)
19      {
20          x = a;
21          y = b;
22      }
23      virtual void setX(int a) { x = a; }
24      virtual void setY(int b) { y = b; }
25      virtual int getX() { return x; }
26      virtual int getY() { return y; }
27      virtual string getType() { return type; }
28
29      virtual void changePoint(int a, int b)
30      {
31          x = a;
32          y = b;
33      }
34      virtual void changeCharc() = 0;
35      virtual void output() = 0;
36  };
```

Рисунок 2.1 – Скриншот кода программы

```
38  class ColoredPoint : public Point
39  {
40  protected:
41      string color;
42
43  public:
44      ColoredPoint() : Point()
45      {
46          type = "Цветная точка";
47          color = "Нет цвета";
48      }
49      ColoredPoint(int a, int b, string c) : Point(a, b)
50      {
51          type = "Цветная точка";
52          color = c;
53      }
54      string getPointColor() { return color; }
55
56      void setColor(string c) { color = c; }
57      virtual void setCharacter(int a, int b, string c)
58      {
59          this->changePoint(a, b);
60          color = c;
61      }
62
63      void output() override
64      {
65          cout << endl
66               << type << endl;
67          cout << "X = " << x << "; Y = " << y << ";";
68          cout << "\nЦвет: " << color << endl;
69      }
70      void changeCharc() override
71      {
72          cout << "X = ";
73          cin >> x;
74          cout << "Y = ";
75          cin >> y;
76          cout << "Цвет: ";
77          cin >> color;
78      }
79  };
```

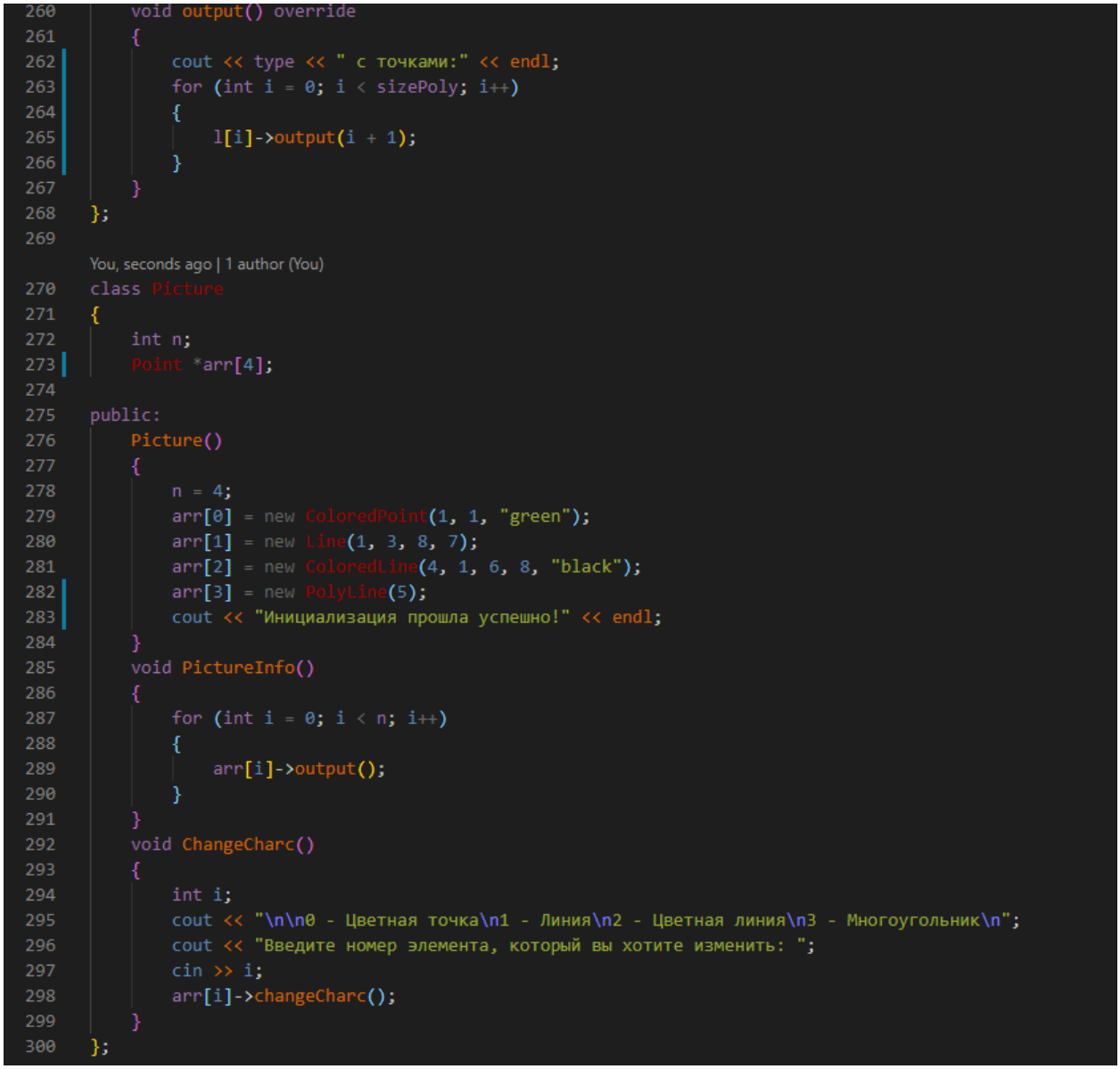
Рисунок 2.2 – скриншот кода программы

```
81  class Line : public Point
82  {
83  protected:
84      int endx, endy;
85
86  public:
87      Line()
88      {
89          x = y = endx = endy = 0;
90          type = "Линия";
91      }
92      Line(int x1, int y1, int x2, int y2)
93      {
94          x = x1;
95          y = y1;
96          endx = x2;
97          endy = y2;
98          type = "Линия";
99      }
100     int getEndX()
101     {
102         return endx;
103     }
104     int getEndY()
105     {
106         return endy;
107     }
108
109     void setLine(int x1, int y1, int x2, int y2)
110     {
111         x = x1;
112         y = y1;
113         endx = x2;
114         endy = y2;
115     }
116     void output() override
117     {
118         cout << endl
119              << type << endl;
120         cout << "Первая точка: " << endl;
121         cout << "X = " << x << "; Y = " << y << ";";
122         cout << endl;
123         cout << "Вторая точка: " << endl;
124         cout << "X = " << endx << "; Y = " << endy << ";\n";
125     }
```

Рисунок 2.3 – Скриншот кода программы

```
212             l[2] = new Line(3, 8, 2, 1);
213             l[3] = new Line(2, 1, 4, 6);
214             l[4] = new Line(4, 6, 1, 1);
215 
216             sizePoly = 5;
217         }
218         PolyLine(int sizeN)
219         {
220             sizePoly = sizeN;
221             type = "\nМногоугольник";
222             changeCharc();
223         }
224         void changeCharc()
225         {
226             int tempX1, tempY1, tempX2, tempY2;
227             cout << "Введите точки для многоугольника: " << endl;
228             for (int i = 0; i < sizePoly; i++)
229             {
230                 if (i == 0)
231                 {
232                     cout << "X(" << i + 1 << ") = ";
233                     cin >> tempX1;
234                     cout << "Y(" << i + 1 << ") = ";
235                     cin >> tempY1;
236                 }
237                 else
238                 {
239                     tempX1 = tempX2;
240                     tempY1 = tempY2;
241                 }
242 
243                 if (i == sizePoly - 1)
244                 {
245                     tempX1 = l[i - 1]->getEndX();
246                     tempY1 = l[i - 1]->getEndY();
247                     tempX2 = l[0]->getX();
248                     tempY2 = l[0]->getY();
249                 }
250                 else
251                 {
252                     cout << "X(" << i + 2 << ") = ";
253                     cin >> tempX2;
254                     cout << "Y(" << i + 2 << ") = ";
255                     cin >> tempY2;
256                 }
257                 l[i] = new Line(tempX1, tempY1, tempX2, tempY2);
258             }
259         }
```

Рисунок 2.4 – Скриншот кода программы



```
260     void output() override
261     {
262         cout << type << " с точками:" << endl;
263         for (int i = 0; i < sizePoly; i++)
264         {
265             l[i]->output(i + 1);
266         }
267     }
268 };
269
    You, seconds ago | 1 author (You)
270 class Picture
271 {
272     int n;
273     Point *arr[4];
274
275 public:
276     Picture()
277     {
278         n = 4;
279         arr[0] = new ColoredPoint(1, 1, "green");
280         arr[1] = new Line(1, 3, 8, 7);
281         arr[2] = new ColoredLine(4, 1, 6, 8, "black");
282         arr[3] = new PolyLine(5);
283         cout << "Инициализация прошла успешно!" << endl;
284     }
285     void PictureInfo()
286     {
287         for (int i = 0; i < n; i++)
288         {
289             arr[i]->output();
290         }
291     }
292     void ChangeCharc()
293     {
294         int i;
295         cout << "\n\n0 - Цветная точка\n1 - Линия\n2 - Цветная линия\n3 - Многоугольник\n";
296         cout << "Введите номер элемента, который вы хотите изменить: ";
297         cin >> i;
298         arr[i]->changeCharc();
299     }
300 };
```

Рисунок 2.5 – Скриншот кода программы

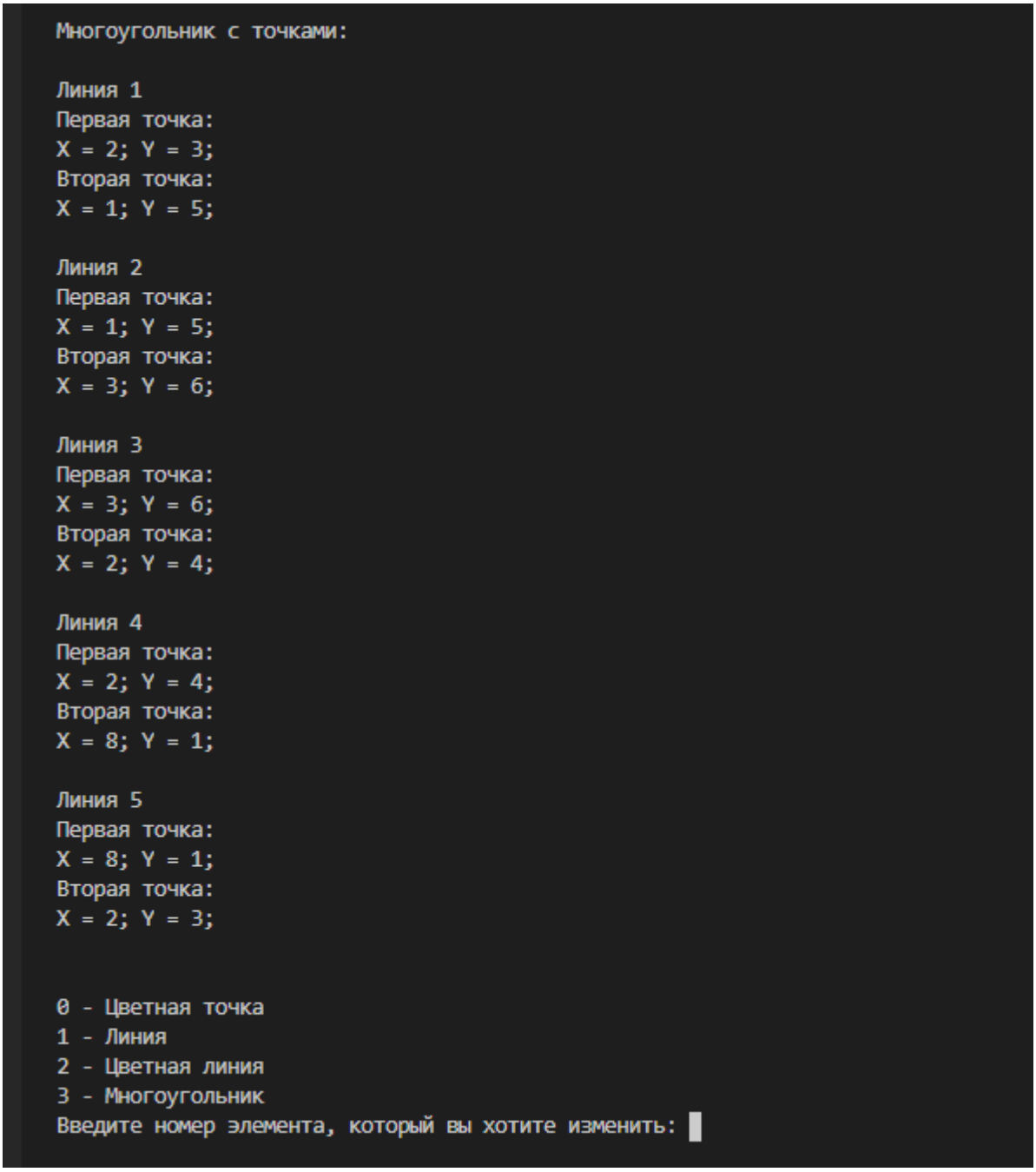
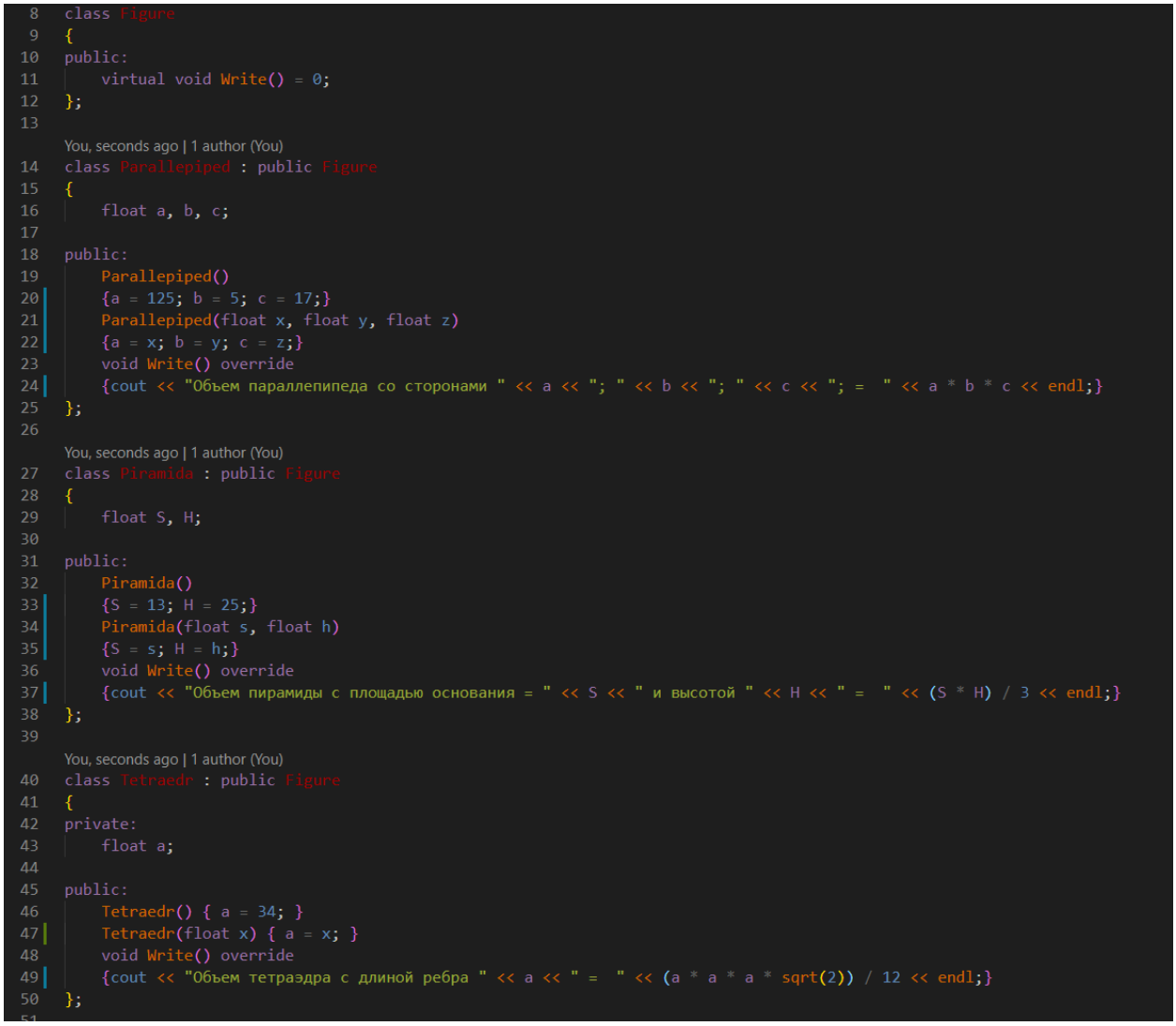


Рисунок 2.6 – Скриншот результата работы программы

Задание 3. Создать абстрактный базовый класс фигура с виртуальной функцией – печать объёма фигуры. Создать производные классы: параллелепипед, пирамида, тетраэдр, шар со своими функциями печати объема. Для проверки определить массив указателей на абстрактный класс, которым присваиваются адреса различных объектов.

Скриншоты кода программы и результат представлены на рисунках 3.1 – 3.2.



```
class Figure
{
public:
    virtual void Write() = 0;
};

You, seconds ago | 1 author (You)
class Parallepiped : public Figure
{
    float a, b, c;

public:
    Parallepiped()
    {a = 125; b = 5; c = 17;}
    Parallepiped(float x, float y, float z)
    {a = x; b = y; c = z;}
    void Write() override
    {cout << "Объем параллепипеда со сторонами " << a << "; " << b << "; " << c << "; =  " << a * b * c << endl;}
};

You, seconds ago | 1 author (You)
class Piramida : public Figure
{
    float S, H;

public:
    Piramida()
    {S = 13; H = 25;}
    Piramida(float s, float h)
    {S = s; H = h;}
    void Write() override
    {cout << "Объем пирамиды с площадью основания = " << S << " и высотой " << H << " =  " << (S * H) / 3 << endl;}
};

You, seconds ago | 1 author (You)
class Tetraedr : public Figure
{
private:
    float a;

public:
    Tetraedr() { a = 34; }
    Tetraedr(float x) { a = x; }
    void Write() override
    {cout << "Объем тетраэдра с длиной ребра " << a << " =  " << (a * a * a * sqrt(2)) / 12 << endl;}
};
```

Рисунок 3.1 – Скриншот кода программы

```cpp
class Shar : public Figure
{
private:
    float r;

public:
    Shar() { r = 23; }
    Shar(float R) { r = R; }
    void Write() override
    {
        cout << "Объем шара с радиусом " << r << " = " << (4 * PI * r * r * r) / 3 << endl;
    }
};

int main()
{
    setlocale(LC_ALL, "65001");
    system("chcp 65001");
    system("cls");
    Parallepiped parallepiped(13, 15, 17);
    Piramida piramida(12, 21);
    Tetraedr tetraedr(10);
    Shar shar(8);

    Figure *f[4]{
        f[0] = &parallepiped,
        f[1] = &piramida,
        f[2] = &tetraedr,
        f[3] = &shar};

    for (int i = 0; i < 4; i++)
    {
        f[i]->Write();
    }

    return 0;
}
```

```
ПРОБЛЕМЫ  ВЫХОДНЫЕ ДАННЫЕ  ТЕРМИНАЛ  КОНСОЛЬ ОТЛАДКИ
Объем параллепипеда со сторонами 13; 15; 17; =  3315
Объем пирамиды с площадью основания = 12 и высотой 21 =  84
Объем тетраэдра с длиной ребра 10 =  117.851
Объем шара с радиусом 8 = 2144.66
```

Рисунок 3.2 – Скриншот результата работы программы

Вывод: освоены и применены на практике виртуальные функции и абстрактные базовые классы.