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 «Программирование»

Студентка гр.548-1 К.Е.Анисимова

«\_\_» \_\_\_\_\_\_ 2019 г.

Доцент кафедры ЭМИС

Шельмина Е. А.

«\_\_»\_\_\_\_\_\_\_ 2019г.

Томск 2019

**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 С++, 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: точка, круг, квадрат.

Используя виртуальные функции, выведите на экран размер и координаты графического объекта.

Скриншоты кода программы и результат представлены на рисунках 1.1 – 1.4.

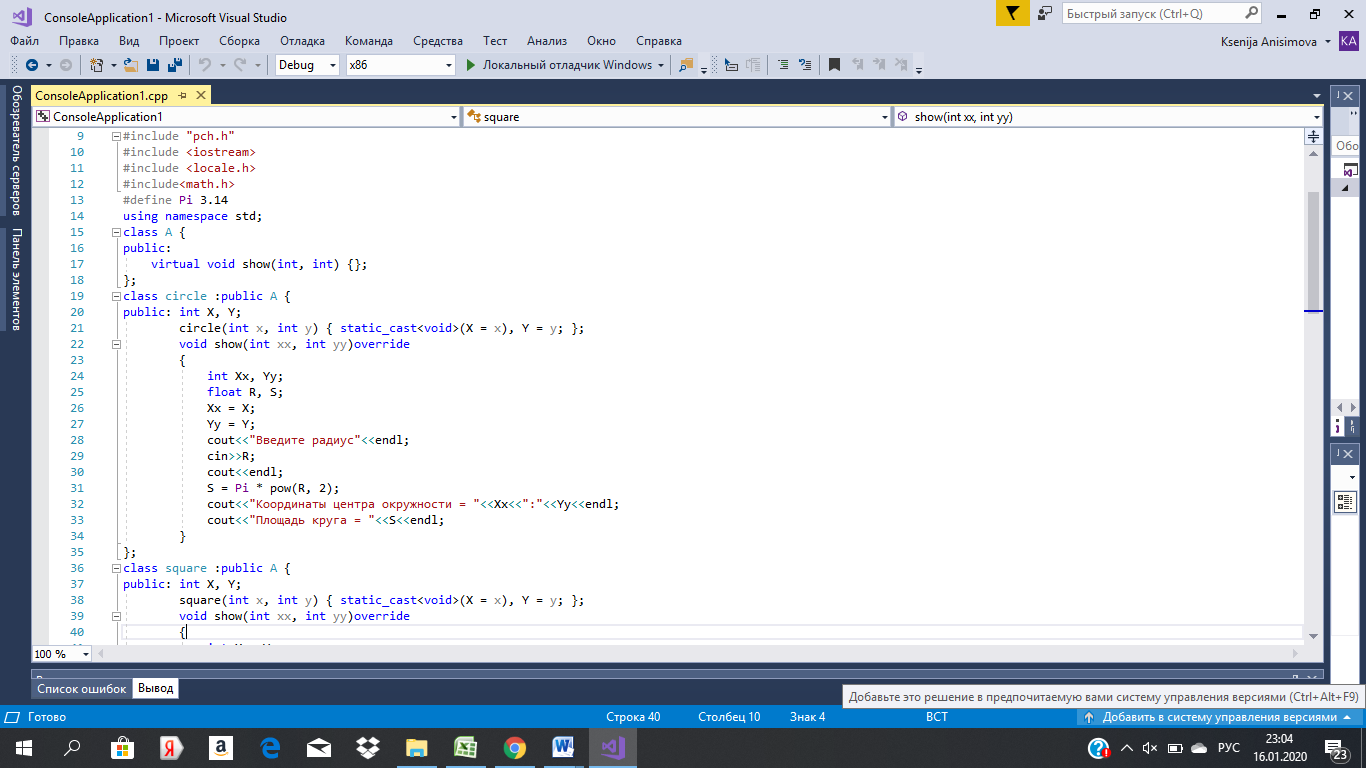


Рисунок 1.1 – скриншот кода программы

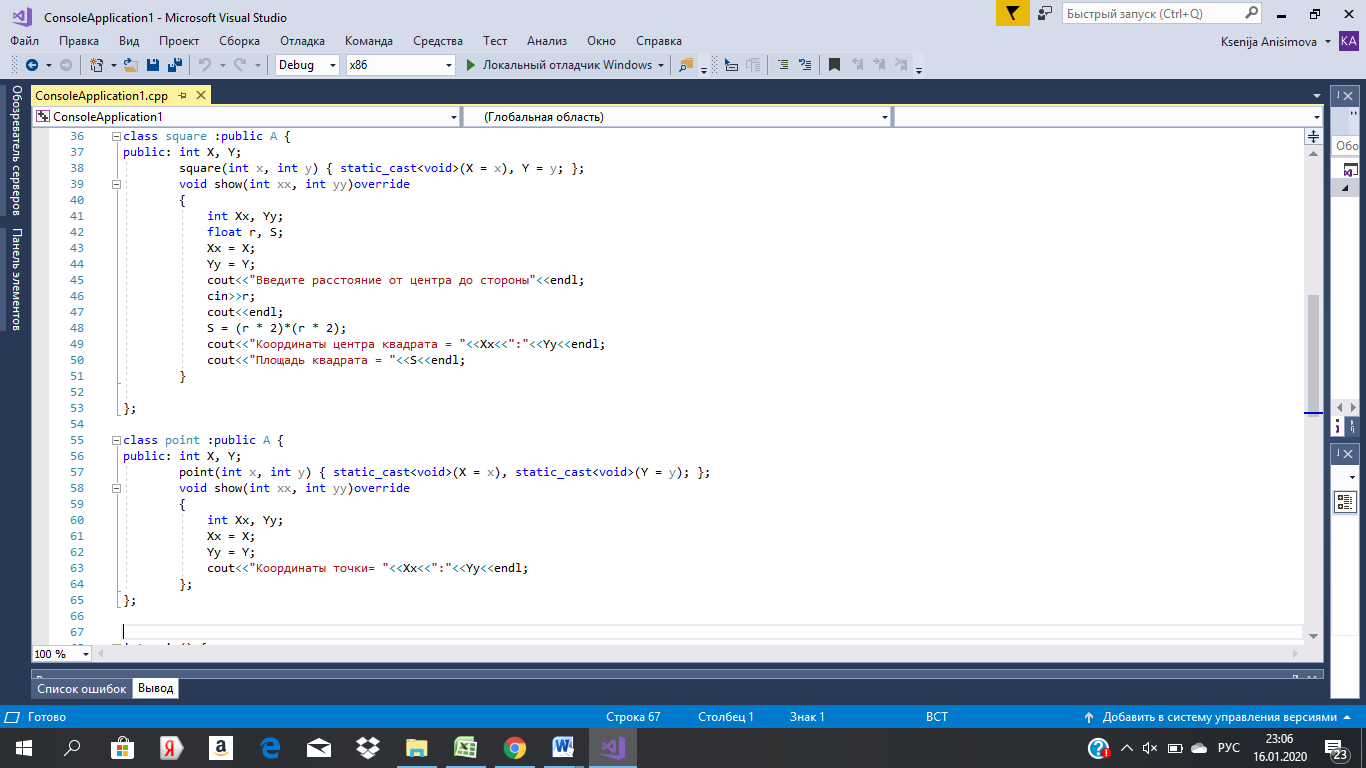


Рисунок 1.2 – скриншот кода программы

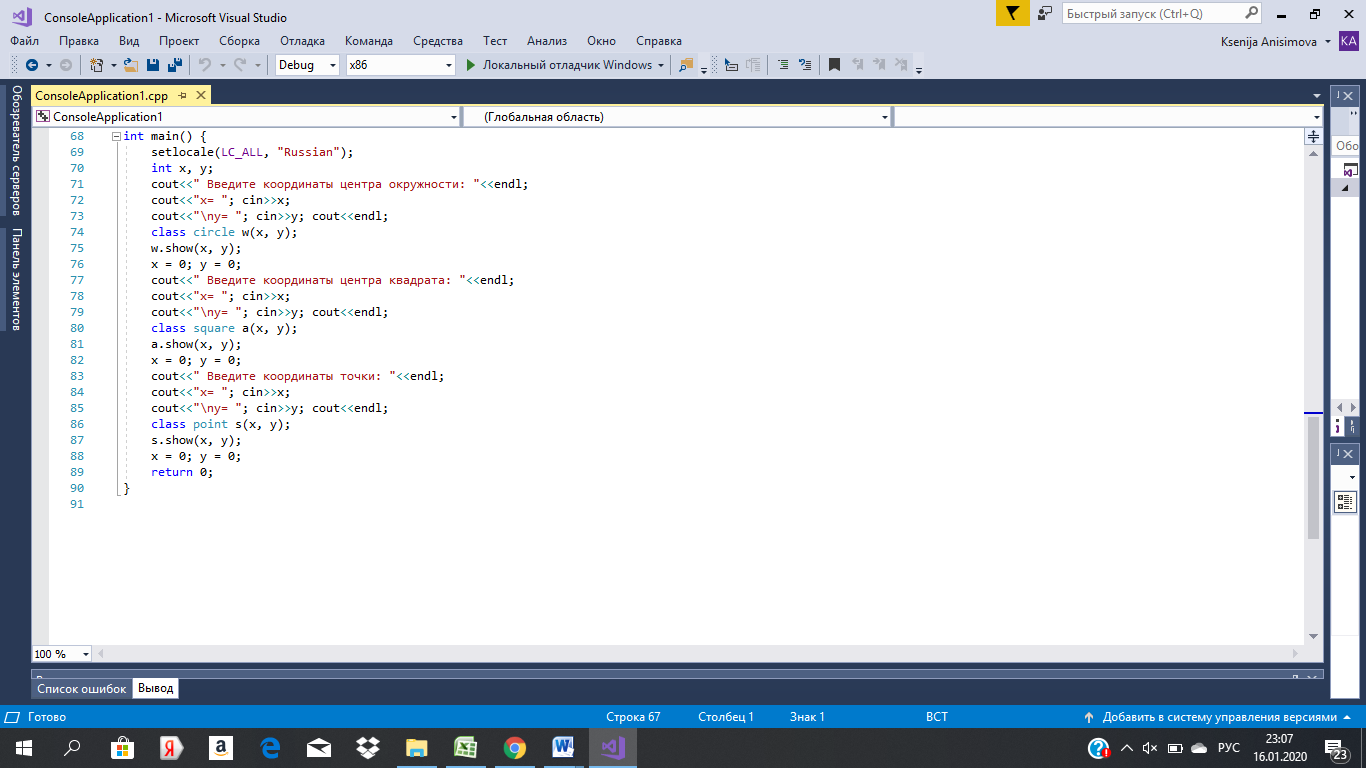


Рисунок 1.3 – скриншот кода программы

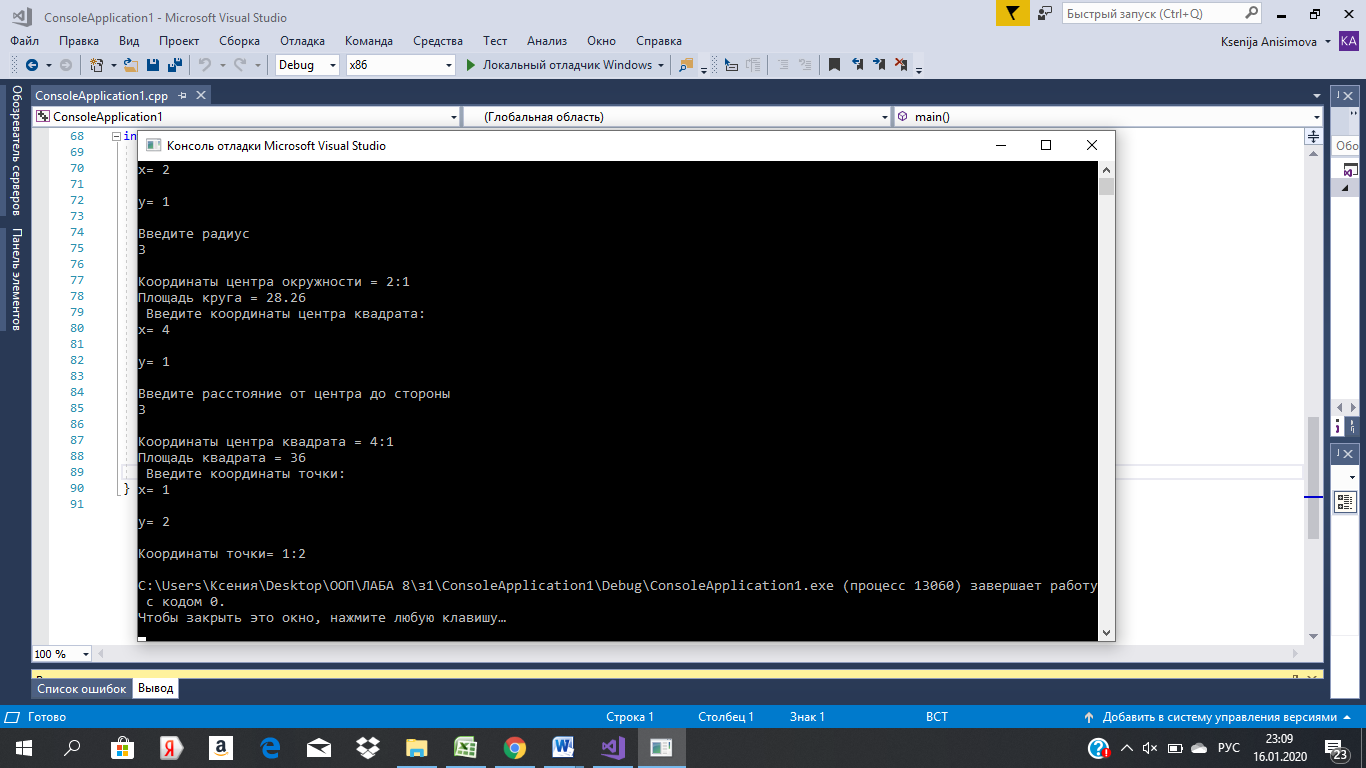


Рисунок 1.4 – скриншот кода программы

Задание 2. Создать абстрактный класс Figure. На его основе реализовать классы Rectangle (прямоугольник), Circle (круг) и Trapezium (трапеция) с возможностью вычисления площади и периметра. 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 – 2.4.

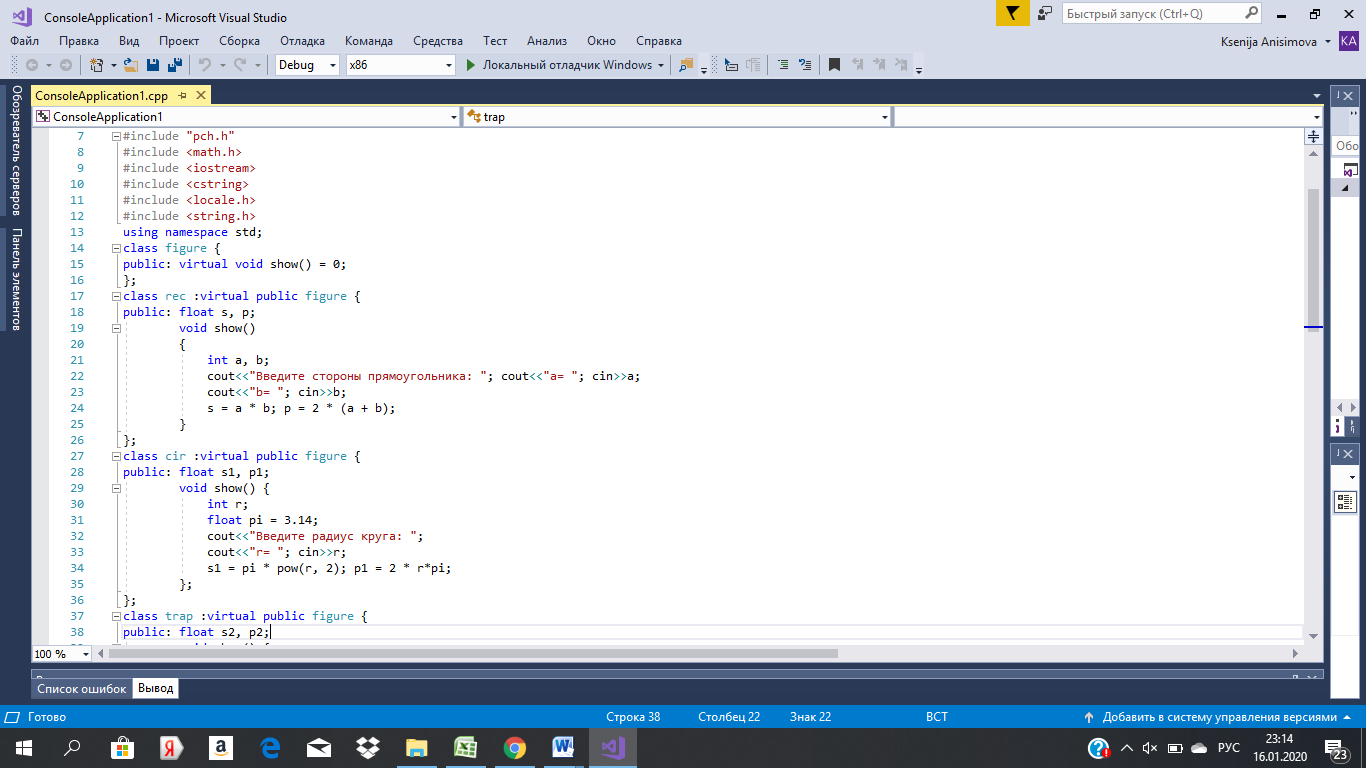


Рисунок 2.1 – скриншот кода программы

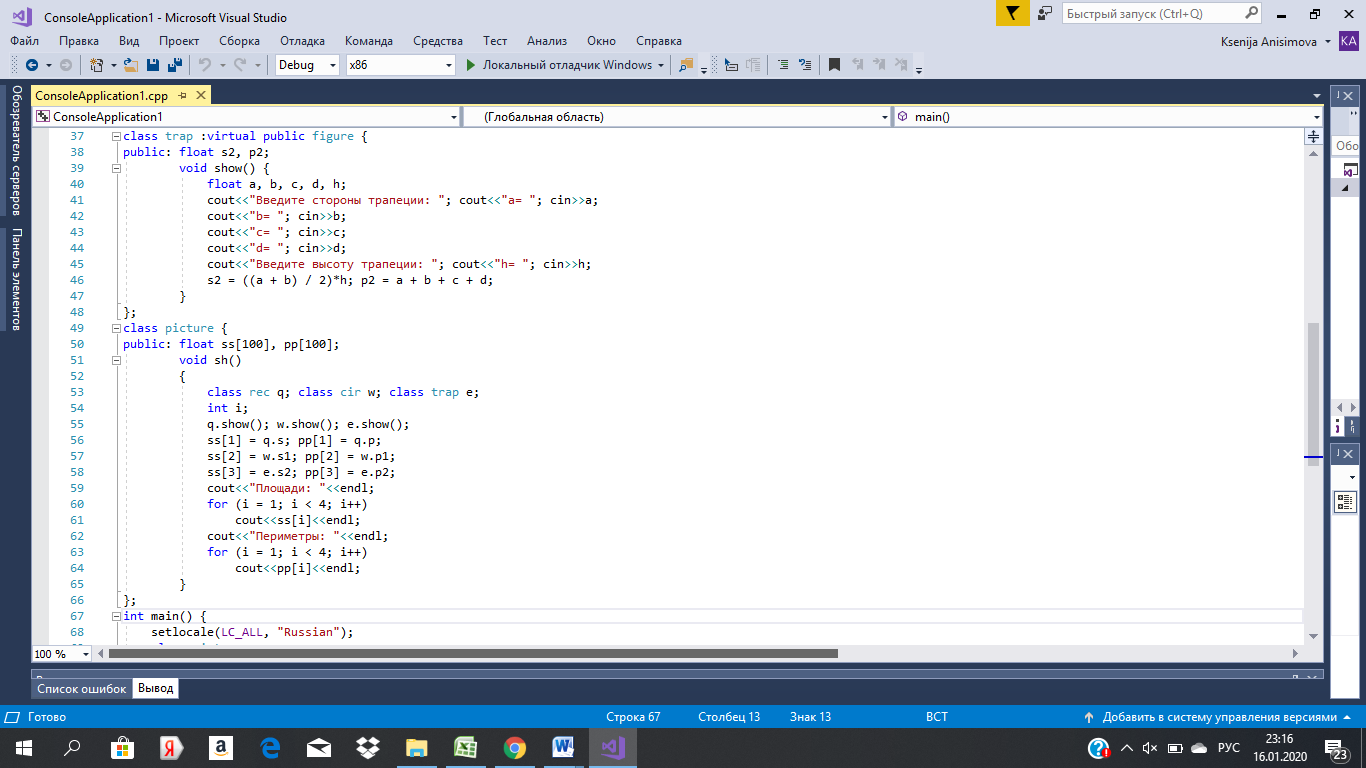


Рисунок 2.2 – скриншот кода программы

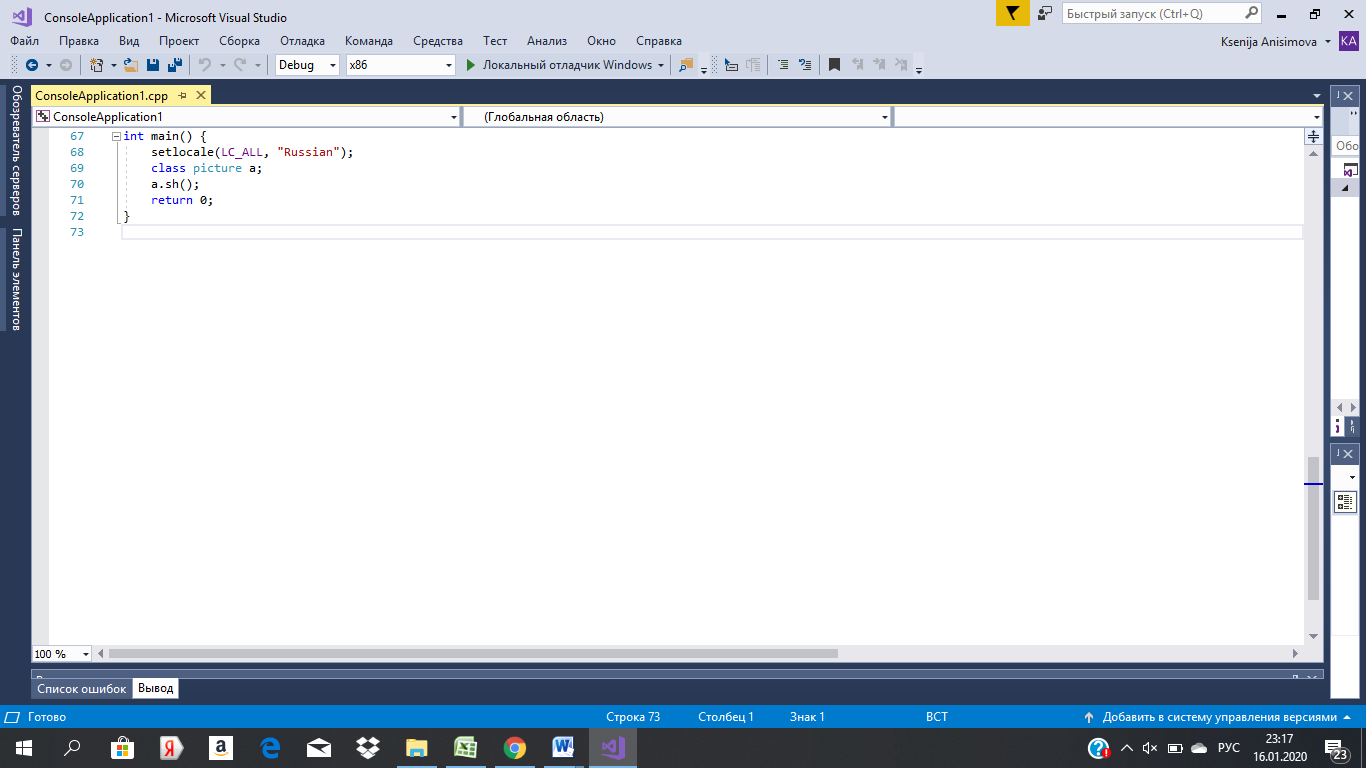


Рисунок 2.3 – скриншот кода программы

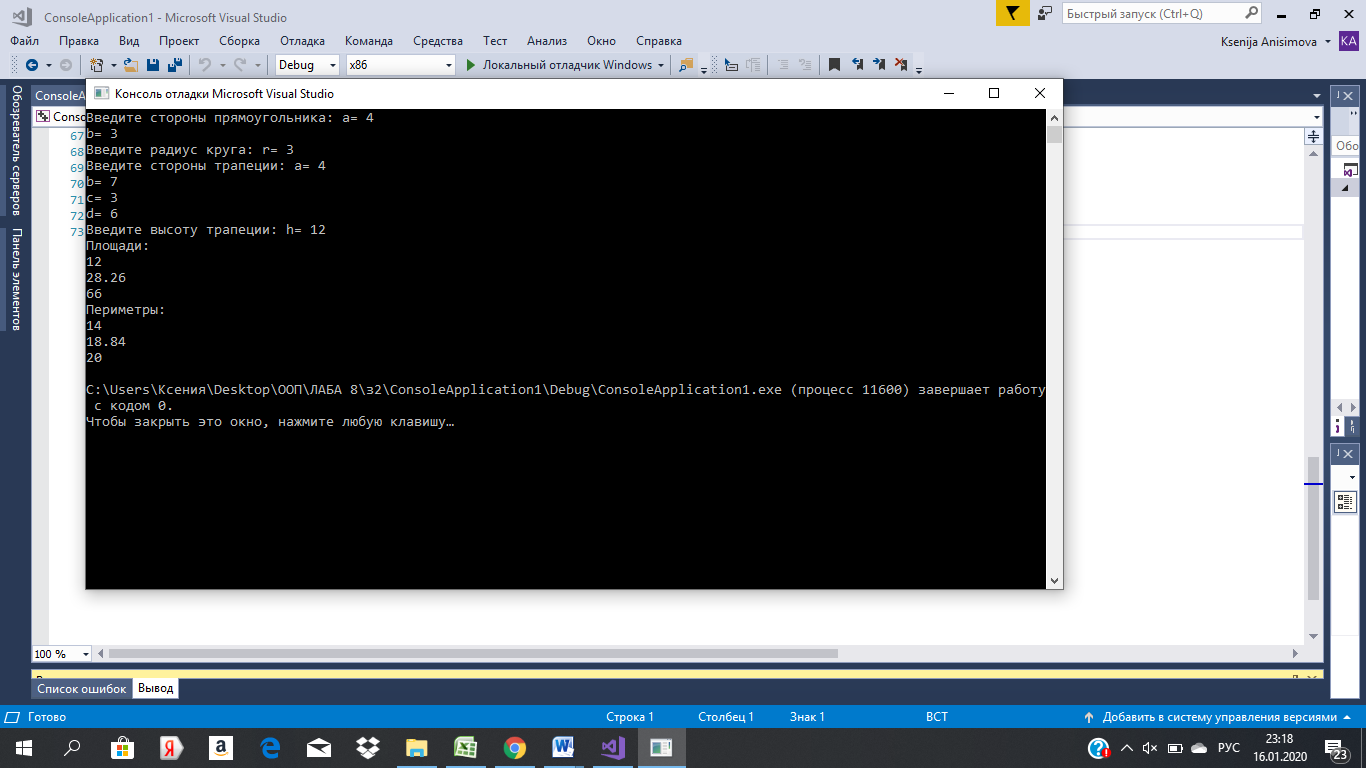


Рисунок 2.4 – скриншот результата программы

Задание 3. Создать абстрактный базовый класс уравнения с виртуальной функцией печать корней уравнения. Создать производные классы: линейное уравнение и квадратное уравнение. Определить функцию вычисления корней уравнений. Для проверки определить массив указателей на абстрактный класс, которым присваиваются адреса различных объектов

Скриншоты кода программы и результат представлены на рисунках 3.1 – 3.3.

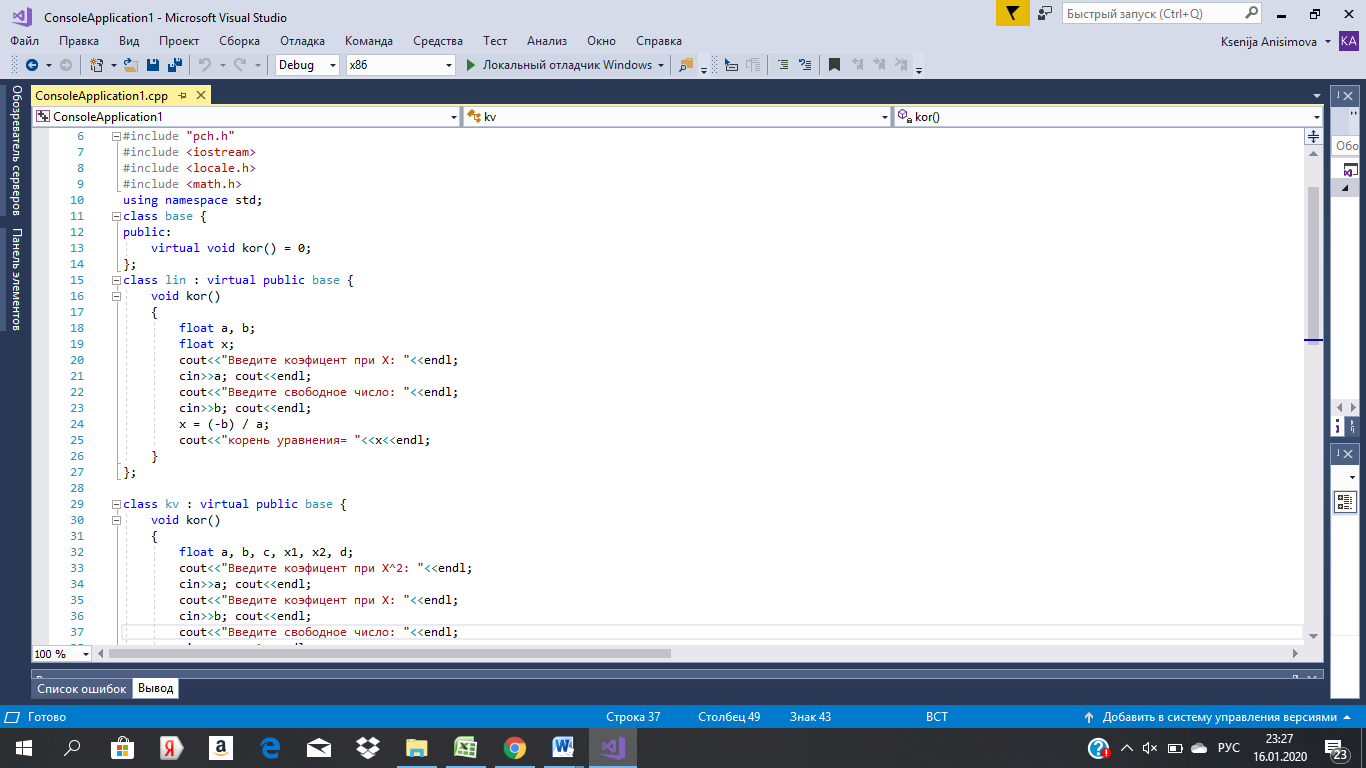


Рисунок 3.1 – скриншот кода программы

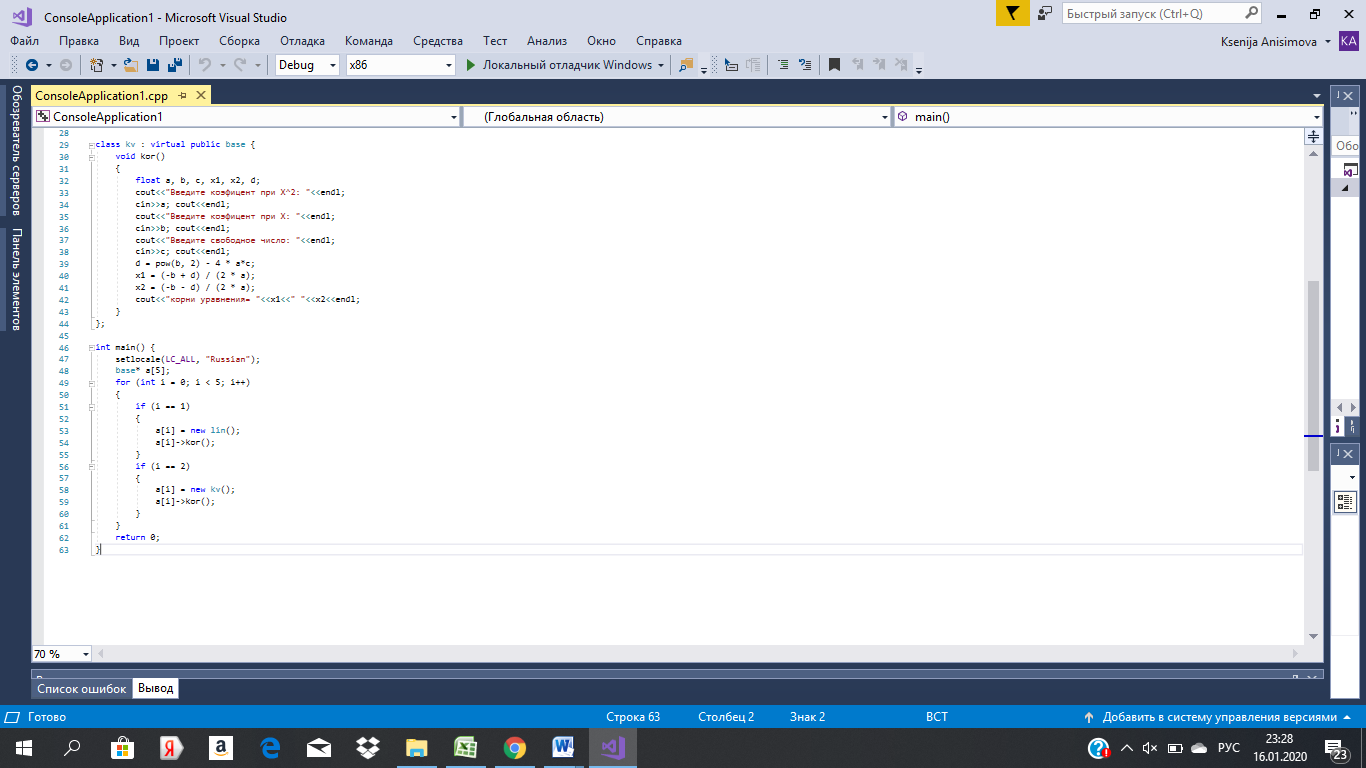


Рисунок 3.2 – скриншот кода программы

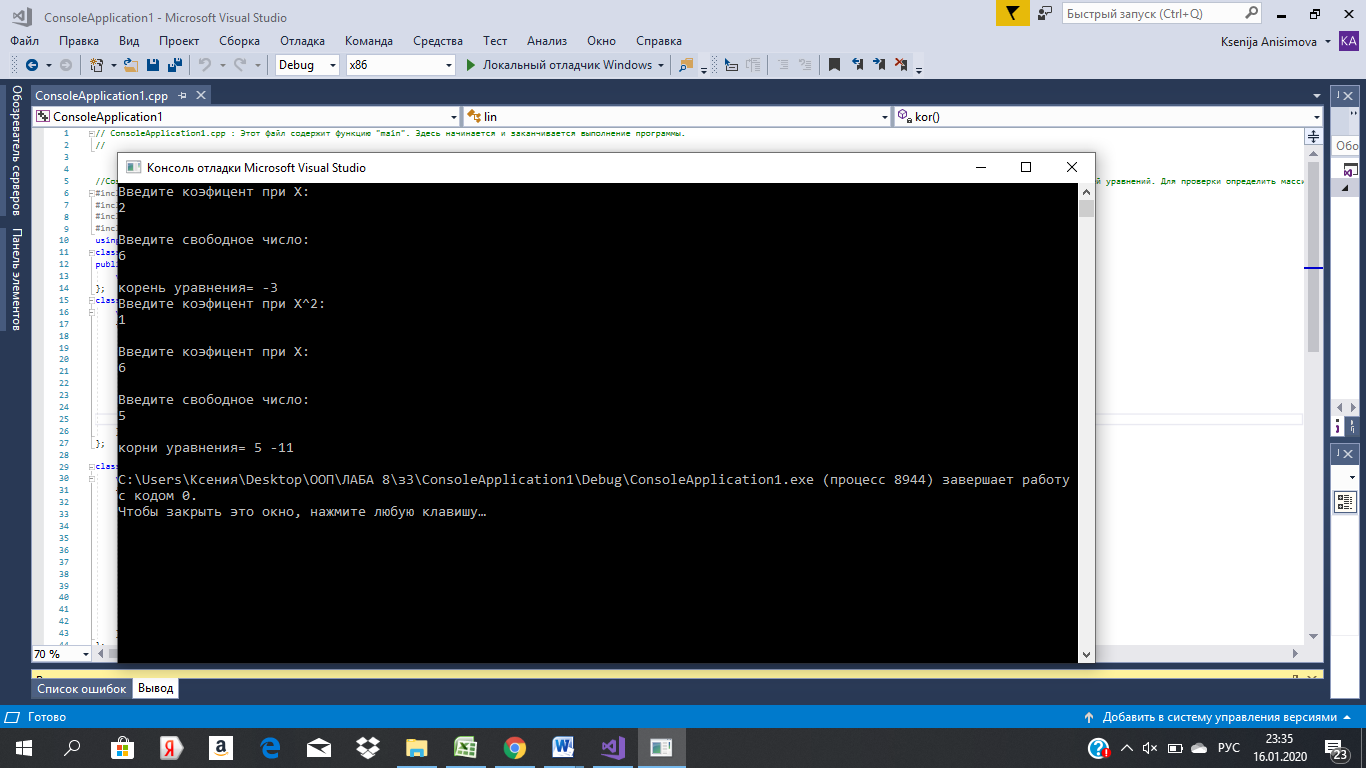


Рисунок 3.3 – скриншот результата программы

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