

Министерство науки и высшего образования Российской Федерации  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«Оренбургский государственный университет»

Кафедра управления и информатики в технических системах

В.А. Трипкош

# **ПРОГРАММНО-МЕТОДИЧЕСКИЕ КОМПЛЕКСЫ СИСТЕМ АВТОМАТИЗИРОВАННОГО ПРОЕКТИРОВАНИЯ**

Методические указания

Рекомендовано к изданию редакционно-издательским советом федерального государственного бюджетного образовательного учреждения высшего образования «Оренбургский государственный университет» для обучающихся по образовательным программам высшего образования по направлению подготовки 27.04.03 Системный анализ и управление

Оренбург  
2019

УДК 004.89(076.5)

ББК 32.971.3я7

Т 67

Рецензент – профессор, доктор технических наук А.М. Пищухин

**Трипкош, В.А.**

Т67 Программно-методические комплексы систем автоматизированного проектирования: методические указания / В.А. Трипкош; Оренбургский гос. ун-т. – Оренбург : ОГУ, 2019. – 73 с.

Методические указания направлены на формирование профессиональных компетенций, связанных с выполнением лабораторных работ по направлению подготовки 27.04.03 Системный анализ и управление. В методических указаниях приведены основные положения по лабораторным работам, формируемые компетенции, а также описания лабораторных работ и указания по выполнению всех их компонентов, включая допуск к работе и защиту отчетов.

Методические указания предназначены для выполнения лабораторных работ по дисциплине «Программно-методические комплексы систем автоматизированного проектирования» для обучающихся очной и заочной форм обучения по образовательным программам высшего образования по направлению подготовки 27.04.03 Системный анализ и управление.

УДК 004.89(076.5)

ББК 32.971.3я7

© Трипкош В.А., 2019

© ОГУ, 2019

## Содержание

Введение .....	4
1 Формируемые компетенции .....	7
2 Общие правила выполнения лабораторных работ .....	8
3 Описания лабораторных работ и методические указания по их выполнению .....	13
3.1 Лабораторная работа № 1. Установка и изучение интегрированной среды быстрой разработки приложений Microsoft Visual Studio, библиотек двумерной и трёхмерной компьютерной графики OpenGL .....	13
3.2 Лабораторная работа № 2. Начало работы в интегрированной среде Microsoft Visual Studio, разработка простого приложения .....	16
3.3 Лабораторная работа № 3. Разработка простого приложения, использующего графические библиотеки OpenGL в среде Microsoft Visual Studio .....	23
3.4 Лабораторная работа № 4. Разработка приложений, использующих графические библиотеки OpenGL в среде Microsoft Visual Studio (моделирование линий) .....	36
3.5 Лабораторная работа № 5. Разработка приложений, использующих графические библиотеки OpenGL в среде Microsoft Visual Studio (построение поверхностей и сложных объектов, наложение текстуры на примитивы) .....	40
3.6 Лабораторная работа № 6. Разработка приложений, использующих графические библиотеки OpenGL в среде Microsoft Visual Studio (создание анимации) .....	53
3.7 Лабораторная работа № 7. Разработка программного приложения в среде Visual Studio для выполнения инженерных расчётов .....	57
3.8 Лабораторная работа № 8. Разработка программного приложения в среде Visual Studio для выполнения инженерных расчётов и их представления в виде графиков .....	69
Список использованных источников .....	73

## Введение

Современный этап развития проектных работ, как важной стадии промышленного производства, характеризуется всеобъемлющими процессами компьютеризации и автоматизации.

В результате компьютеризации осуществляется оснащение проектных организаций необходимыми средствами компьютерной техники и программным обеспечением для комплексной автоматизации технологических и производственных процессов.

Автоматизация позволяет снизить степень участия человека в производственном процессе в ходе проектирования, управления, контроля и производства готовых изделий.

Процессы компьютеризации и автоматизации проектных работ обусловили создание и применение систем автоматизированного проектирования (САПР).

САПР – это автоматизированная система, реализующая информационную технологию выполнения функций проектирования [1]. САПР представляет собой организационно-техническую систему, предназначенную для автоматизации процесса проектирования, состоящую из персонала и комплекса технических, программных и других средств автоматизации его деятельности [2, 3].

Составными структурными частями САПР, жестко связанными с организационной структурой проектной организации, являются подсистемы САПР, в которых при помощи специализированных комплексов средств решается функционально законченная последовательность задач САПР с получением соответствующих проектных решений и проектных документов. В качестве специализированных комплексов выступают программно-методические и программно-технические комплексы.

Программно-методический комплекс (ПМК) – это взаимосвязанная совокупность некоторых частей программного, математического, лингвистического, методического и информационного обеспечения, необходимая для получения закон-

ченного проектного решения по объекту проектирования или для выполнения определенных унифицированных процедур.

Поэтому учебная дисциплина «Программно-методические комплексы систем автоматизированного проектирования» посвящена вопросам, связанным с решением задач разработки и использования ПМК САПР в организациях и на предприятиях отраслей народного хозяйства.

Изучение дисциплины «Программно-методические комплексы систем автоматизированного проектирования» является важным этапом подготовки магистров по направлению 27.04.03 Системный анализ и управление и в рамках этой дисциплины предусмотрено выполнение цикла лабораторных работ.

Лабораторные работы относятся к основным видам учебных занятий. Они обеспечивают экспериментальное подтверждение теоретических положений дисциплины и формирование учебных и профессиональных практических умений обучающихся. Лабораторные работы составляют важную часть теоретической и профессиональной практической подготовки обучающихся.

Лабораторные работы являются неотъемлемой частью изучения дисциплины «Программно-методические комплексы систем автоматизированного проектирования» и имеют целью закрепление навыков по использованию программных приложений и библиотек, применяемых в ПМК САПР различного назначения, при решении проектных задач.

Лабораторные работы по дисциплине «Программно-методические комплексы систем автоматизированного проектирования» относятся к средствам, обеспечивающим решение следующих основных задач:

- закрепление, развитие и детализация теоретических знаний, полученных на лекциях;
- получение новой информации по изучаемой дисциплине;
- приобретение навыков научного исследования процессов, явлений и объектов, изучаемых в рамках данной дисциплины, навыков аналитического мышления при обработке полученных результатов;

– приобретение навыков самостоятельной работы с программно-аппаратными средствами, обеспечивающими разработку и использование ПМК САПР.

Лабораторные работы выполняются по следующим разделам дисциплины:

- общие сведения о системах автоматизированного проектирования и их программно-методических комплексах;
- методы и средства геометрического моделирования технических объектов;
- программно-информационное обеспечение систем автоматизированного проектирования.

Целью настоящих методических указаний является формирование профессиональных компетенций, связанных с выполнением лабораторных работ магистранта по дисциплине «Программно-методические комплексы систем автоматизированного проектирования».

Выполнение цикла лабораторных работ позволит расширить и углубить знания, умения, навыки и компетенции у обучающихся в области разработки ПМК САПР и их использования в профессиональной деятельности.

## 1 Формируемые компетенции

В ходе выполнения цикла лабораторных работ по дисциплине «Программно-методические комплексы систем автоматизированного проектирования» обучающийся должен сформировать профессиональную компетенцию ПК-1, предполагающую овладение способностью применять адекватные методы математического и системного анализа и теории принятия решений для исследования функциональных задач управления техническими объектами на основе отечественных и мировых тенденций развития методов, управления, информационных и интеллектуальных технологий.

В результате формирования ПК-1 обучающиеся должны:

– знать основы построения и развития систем автоматизированного проектирования и их программно-методических комплексов; методы и средства геометрического моделирования технических объектов; программно-информационное обеспечение систем автоматизированного проектирования;

– уметь работать с программными средами быстрой разработки приложений и разрабатывать программные приложения для выполнения инженерных расчётов; работать с библиотеками двумерной и трёхмерной компьютерной графики; выбирать методы проведения экспериментов с ПМК САПР и системно анализировать результаты исследований; определять и учитывать отечественные и мировые тенденции развития методов управления, информационных и интеллектуальных технологий;

– владеть способностью применять адекватные методы математического и системного анализа и теории принятия решений для исследования функциональных задач в области ПМК САПР.

## 2 Общие правила выполнения лабораторных работ

Лабораторная работа – это вид учебного занятия, направленный на углубление и закрепление знаний, практических навыков, овладение современной методикой и техникой эксперимента в соответствии с требованиями образовательной программы высшего образования по соответствующему направлению подготовки магистров. Лабораторная работа состоит из экспериментально-практической, расчетно-аналитической частей и контрольных мероприятий.

Для проведения лабораторных занятий по дисциплине используется аудитория, оснащенная компьютерной техникой. Рабочим местом для выполнения лабораторных работ является лабораторная установка на базе персонального компьютера, оснащенного специальным программным обеспечением:

- Open Office/LibreOffice – свободный офисный пакет программ, включающий в себя текстовый и табличный редакторы, редактор презентаций и другие офисные приложения;

- Microsoft Visual Studio – средство для разработки и проектирования.

Для успешного выполнения лабораторных работ по дисциплине «Программно-методические комплексы систем автоматизированного проектирования» обучающийся должен выполнять следующие правила:

- строго выполнять весь объем домашней подготовки, указанный в описаниях соответствующих лабораторных работ;

- знать, что выполнению каждой работы предшествует проверка готовности обучающегося (допуск к работе), которая проводится преподавателем;

- знать, что после выполнения работы, обучающиеся должны представить отчет с защитой полученных результатов и выводов.

При подготовке к очередной лабораторной работе необходимо помнить, что на лекциях обычно рассматривается не весь материал, а только его часть. Поэтому основной объем учебного материала должен быть усвоен обучающимся в процессе самостоятельной работы с рекомендованной литературой. Особое внимание при этом необходимо обратить на содержание основных положений и выводов, объяс-



нение явлений и фактов, уяснение практического приложения рассматриваемых теоретических вопросов.

В процессе этой работы обучаемый должен стремиться понять и запомнить основные положения рассматриваемого материала и сопоставить изученный материал с конкретными заданиями, изложенными в описании лабораторной работы.

При необходимости следует обратиться за консультацией к преподавателю. При этом перед консультацией требуется хорошо продумать вопросы, которые должны быть разъяснены.

Выполнение лабораторной работы осуществляется под руководством преподавателя при участии ведущего инженера кафедры.

В начале каждого лабораторного занятия проводится инструктаж и проверка знаний обучающихся (допуск к работе) – их теоретической готовности к выполнению задания.

Форму такой проверки (устный опрос, письменная «летучка» или тестирование), перечень контрольных вопросов, а также критерии принятия решения о допуске или недопуске обучающегося к выполнению работы преподаватель доводит до обучающихся на предшествующем лабораторном занятии. Содержание контрольных вопросов (тестов) позволяет оценить знания теоретических положений, которые предстоит экспериментально проверить обучающимся в ходе лабораторной работы, а также знания и умения использовать для этого аппаратные и программные средства лабораторной установки.

Не допущенные к выполнению лабораторной работы обучающиеся, в ходе текущего занятия устраняют указанные преподавателем пробелы. Повторная процедура допуска к работе и выполнение заданий лабораторной работы организуется в указанное преподавателем время, как правило, под руководством ведущего инженера кафедры.

Лабораторная работа выполняется каждым обучающимся индивидуально. Однако преподавателем может быть организована работа обучающихся в составе подгрупп или даже всей учебной группы, в соответствии с порядком и требованиями, изложенными в методических указаниях к выполнению конкретной лабора-

торной работы. Осуществление обучаемыми других (посторонних) действий, не связанных с выполнением лабораторной работы не допускается.

Ответственность за соблюдение правил техники безопасности всеми участниками лабораторной работы возлагается на преподавателя кафедры, проводящего занятие.

Ответственность за порядок на рабочем месте, сохранность полученных для работы оборудования, приборов и материалов несет обучающийся.

Отчет по результатам лабораторной работы выполняется в текстовом редакторе. Отчет должен содержать следующие структурные элементы:

- титульный лист;
- цель и задание на лабораторную работу;
- основную часть, включающую описание хода выполнения задания и полученные результаты (таблицы, графики, скриншоты экрана компьютера и др.);
- основные выводы по работе;
- приложение (при необходимости).

Отчет по результатам лабораторной работы выполняется каждым обучающимся лично вне зависимости от того, осуществлялась ли работа индивидуально или в составе группы. В последнем случае в отчете обучающегося должны быть отражены результаты выполнения индивидуального задания или элементы лично выполненной им работы, в зависимости от порученной ему роли в группе.

Содержание и оформление всех структурных элементов отчета по лабораторной работе должно соответствовать требованиям, предъявляемым к студенческим работам [4].

Отчет в распечатанном виде предъявляется преподавателю на этапе защиты лабораторной работы и/или выгружается в электронном курсе дисциплины в системе Moodle [5].

Результаты выполнения каждой лабораторной работы защищаются каждым обучающимся индивидуально (или группой при выполнении работы в группе) перед преподавателем, ведущим дисциплину. В ходе защиты обучающиеся должны продемонстрировать работу выполненного программного средства, обосновать

сформулированные в отчете выводы и ответить на контрольные вопросы преподавателя (примерный перечень вопросов приведен в описании каждой лабораторной работы).

Обучающиеся, выполнившие лабораторную работу, но не защитившие ее, имеют право на повторную защиту.

Результаты защиты лабораторной работы определяются оценками «отлично», «хорошо», «удовлетворительно», «неудовлетворительно».

Оценка «отлично» выставляется за работу, в которой: полностью выполнено задание (в том числе индивидуальное), в отчете дано всестороннее и глубокое освещение практически исследуемых вопросов; исполнитель показал умение работать с лабораторной установкой и нормативными документами по проведению экспериментов, проводить научные исследования, делать теоретические и практические выводы; обучающийся дает исчерпывающие и аргументированные ответы на контрольные вопросы.

Оценка «хорошо» выставляется за работу, в которой: полностью выполнено задание (в том числе индивидуальное), отчет отвечает, предъявляемым к нему требованиям; исполнитель умеет работать с лабораторной установкой и нормативными документами по проведению экспериментов, однако не на все вопросы дает исчерпывающие и аргументированные ответы.

Оценка «удовлетворительно» выставляется за работу, в которой: задание в основном выполнено, но имелись затруднения с выполнением индивидуального задания, отчет отвечает основным, предъявляемым к нему требованиям; исполнитель в целом умеет работать с лабораторной установкой, но не проявил должного умения самостоятельно проводить эксперименты; обучающийся поверхностно отвечает на контрольные вопросы и допускает существенные недочеты.

Оценка «неудовлетворительно» выставляется за работу, в которой: задание (в том числе индивидуальное) не выполнено и/или отчет не отвечает основным, предъявляемым к нему требованиям; исполнитель не умеет работать с лабораторной установкой и не проявил должного умения самостоятельно проводить экспе-

рименты; обучающийся не отвечает на контрольные вопросы или допускает в ответах грубые ошибки.

Обучающийся, не выполнивший в установленные сроки все лабораторные работы по дисциплине или не защитивший их, кроме случая неявки по уважительной причине, считается имеющим академическую задолженность и не допускается к сдаче экзамена по данной учебной дисциплине.

### **3 Описания лабораторных работ и методические указания по их выполнению**

#### **3.1 Лабораторная работа № 1. Установка и изучение интегрированной среды быстрой разработки приложений Microsoft Visual Studio, библиотек двумерной и трёхмерной компьютерной графики OpenGL**

Цели работы: приобрести практические навыки по установке на персональный компьютер программы Visual Studio Community и библиотек OpenGL; углубить и закрепить знания по использованию этого программного обеспечения (ПО) при решении проектных задач.

Время: 2 часа.

Задание:

- установить на персональный компьютер программное средство для разработки и проектирования Microsoft Visual Studio Community и загрузить библиотеки OpenGL;
- изучить краткие руководства по работе с этими продуктами;
- подготовить и защитить отчет по лабораторной работе.

Методические указания к выполнению задания.

3.1.1 Инструктаж и проверка готовности обучающихся к выполнению лабораторной работы № 1.

В начале занятия получите инструктаж преподавателя и допуск к работе.

3.1.2 Установка программного средства для разработки и проектирования Microsoft Visual Studio Community.

По указанной преподавателем ссылке с официального сайта Microsoft Visual Studio [6] скачайте инсталляцию Visual Studio Community – бесплатной полнофункциональной интегрированной среды разработки для учащихся, разработчиков открытого ПО и отдельных разработчиков.

Запустите установку ПО и в начале установки выберите следующие компоненты (рисунок 1.1):

- разработка приложений для универсальной платформы Windows;

- разработка классических приложений .NET;
- разработка классических приложений на C++;
- проверьте наличие галочки на «отладчике графики».

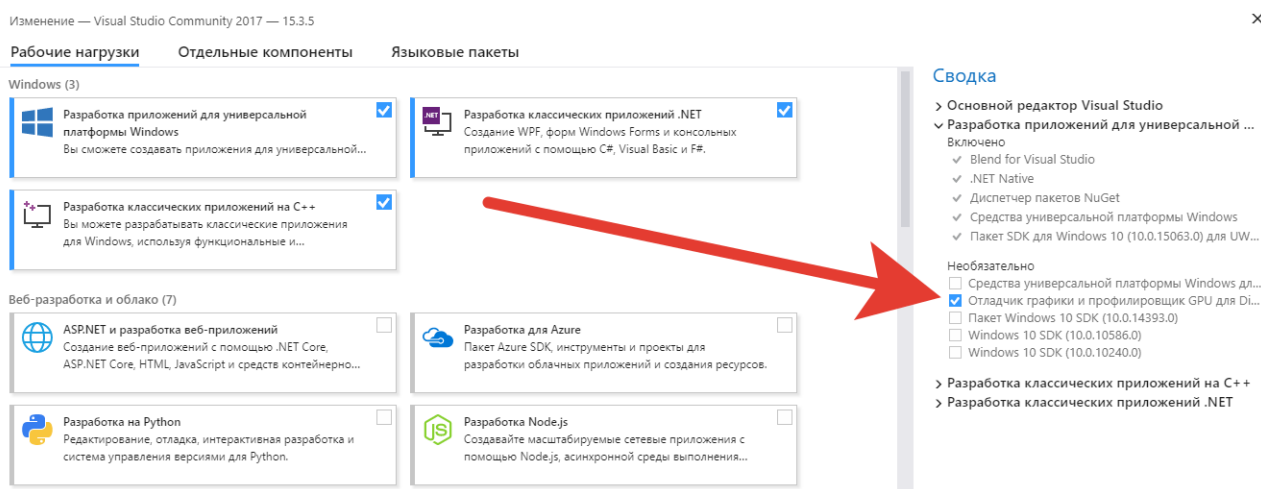


Рисунок 1.1 – Выбор компонентов Visual Studio Community

### 3.1.3 Загрузка библиотек OpenGL.

Загрузите библиотеку Glut по указанной преподавателем ссылке с сайта [7] или откройте архивный файл с именем glutdlls37beta.zip, расположенный в указанной преподавателем папке персонального компьютера лабораторной установки. В архиве находятся пять файлов: glut.dll; glut32.dll; glut.h; glut.lib; glut32.lib.

Скопируйте файлы glut.dll и glut32.dll в папку C:\\Windows\\System32 своего компьютера.

Откройте папку с установленной программой Visual Studio и скопируйте файл glut.h по следующим адресам:

- ...\\Microsoft Visual Studio\\2019\\Community\\VC\\Auxiliary\\VS\\include;
- ...\\Microsoft Visual Studio\\2019\\Community\\VC\\Tools\\MSVC\\ 14.11.25503\\include.

Два оставшихся файла (glut.lib и glut32.lib) скопируйте в папки:

- ...\\Microsoft Visual Studio\\2017\\Community\\VC\\Auxiliary\\VS\\lib;
- ...\\Microsoft Visual Studio\\2017\\Community\\VC\\Tools\\MSVC\\ 14.11.25503\\lib.

### 3.1.4 Изучение руководств по работе с программным приложением Microsoft Visual Studio и библиотекой OpenGL.

По указанной преподавателем ссылке зайдите на сайт Microsoft и изучите описание Visual Studio Community 2019 в части следующих разделов:

- «Начать работу», где приведены руководства, примеры и документация по программному продукту, которые помогут начать писать код (рисунок 1.2);
- «Дополнительная информация», где приведены дополнительные видеоролики, документация и руководства.

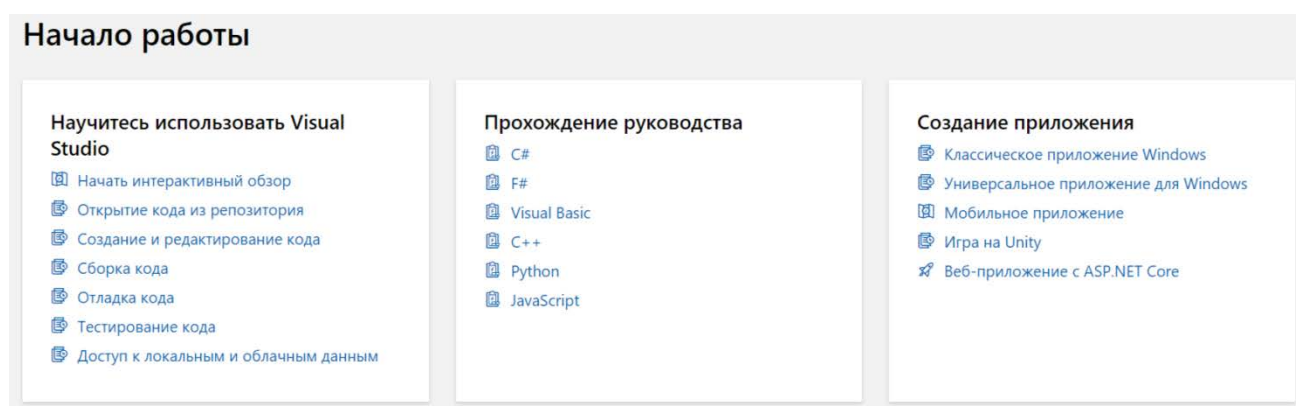


Рисунок 1.2 – Содержание раздела «Начать работу»

### 3.1.5 Подготовка и защита отчета по лабораторной работе.

По результатам выполнения всех заданий подготовьте письменный отчет и представьте его для защиты лабораторной работы № 1. В ходе защиты отчета необходимо быть готовым к демонстрации установленных программных продуктов и к ответам на контрольные вопросы преподавателя.

Примерный перечень контрольных вопросов:

- 1) Назначение интегрированной среды Microsoft Visual Studio.
- 2) В чем состоят преимущества использования программной среды Microsoft Visual Studio при решении проектных задач.
- 3) Основные режимы и этапы установки на компьютер программного средства Microsoft Visual Studio Visual Studio Community.
- 4) Содержание этапа подготовка компьютера к установке Visual Studio.

- 5) Содержание этапов скачивания и установки установщика Visual Studio.
- 6) Содержание этапов выбора рабочих нагрузок и отдельных компонентов Visual Studio.
- 7) Содержание этапов выбора расположения установки и начала разработки проекта Visual Studio.
- 8) Что такое OpenGL? Назначение и преимущества использования библиотек OpenGL при решении проектных задач.
- 9) Назовите основные задачи, на которые ориентируется OpenGL.
- 10) Порядок подключения OpenGL в Visual Studio.

### **3.2 Лабораторная работа № 2. Начало работы в интегрированной среде Microsoft Visual Studio, разработка простого приложения**

Цели работы: приобрести базовые навыки работы в интегрированной среде Microsoft Visual Studio при разработке простого консольного приложения; углубить и закрепить знания по использованию Visual Studio при решении проектных задач.

Время: 2 часа.

Задание:

- создать простое консольное приложение в интегрированной среде Microsoft Visual Studio;
- добавить программный код в приложение с учетом индивидуального задания;
- отладить и протестировать приложение;
- подготовить и защитить отчет по лабораторной работе.

Методические указания к выполнению задания.

3.2.1 Инструктаж и проверка готовности обучающихся к выполнению лабораторной работы № 2.

В начале занятия получите инструктаж преподавателя и допуск к работе. После чего получите индивидуальное задание на работу № 2.



### 3.2.2 Создание простого консольного приложения в среде Microsoft Visual Studio.

При разработке любого приложения в Visual Studio необходимо сначала создать проект и решение. В данном случае создается консольное приложение Windows [8].

Поэтому в строке меню выберите Файл, Создать, Проект (рисунок 2.1).

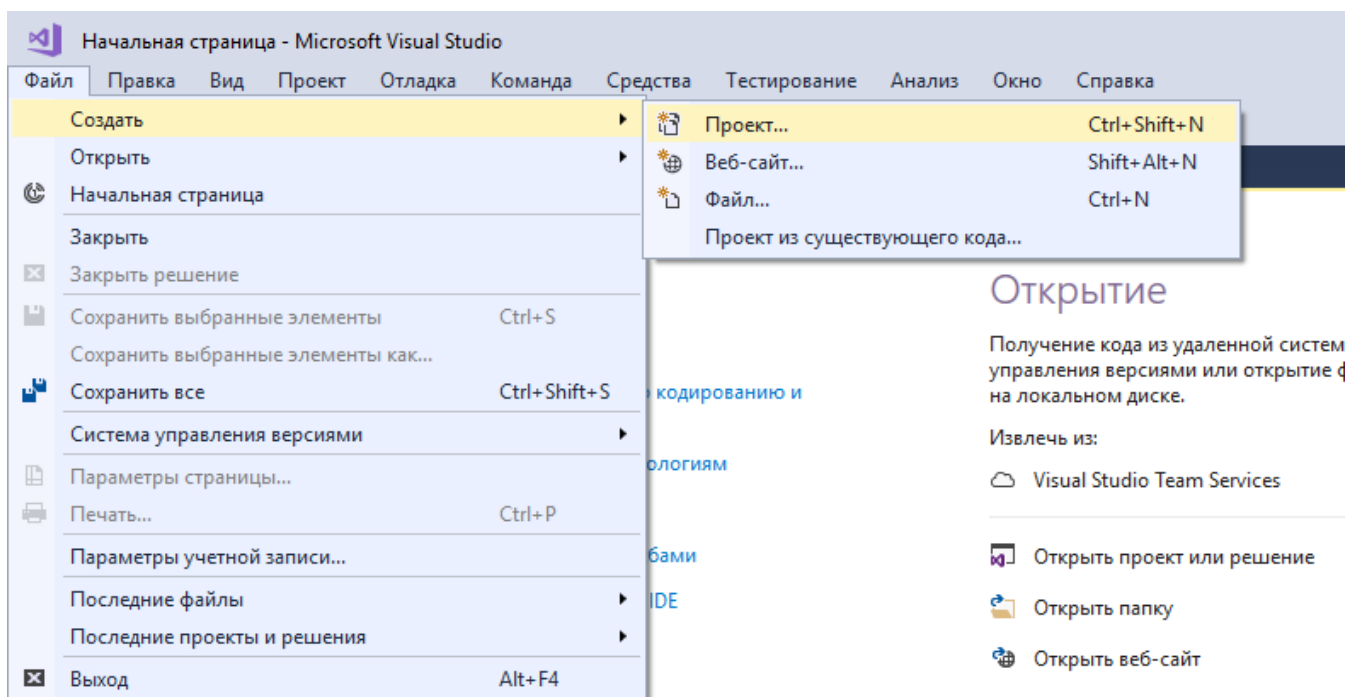


Рисунок 2.1 – Создание проекта

В категории Visual C++ выберите шаблон Консольное приложение Windows с названием ConsoleApplication (рисунок 2.2).

Проект ConsoleApplication и решение с основными файлами для консольного приложения будут автоматически созданы и загружены в окно «Обозреватель решений». Исходный файл ConsoleApplication откроется в редакторе кода (рисунок 2.3).

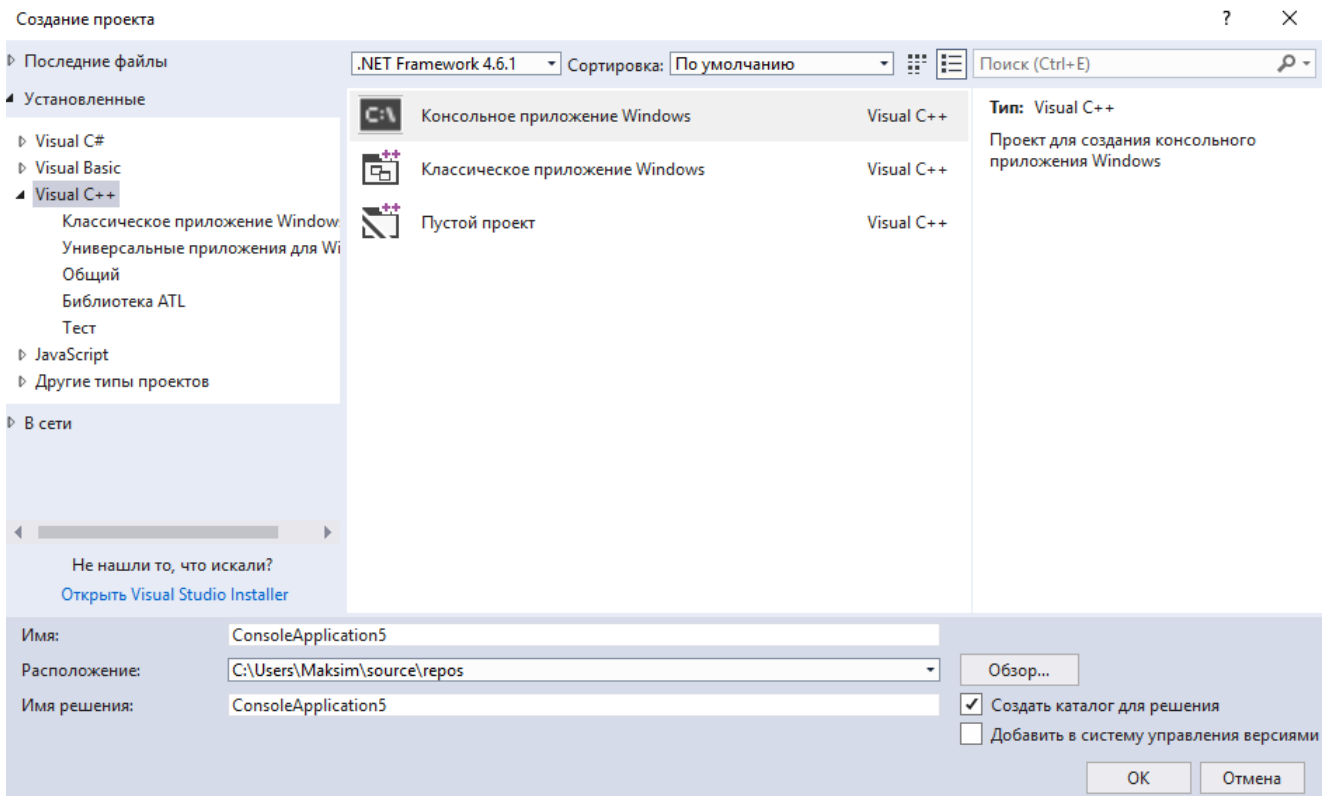


Рисунок 2.2 – Создание консольного приложения

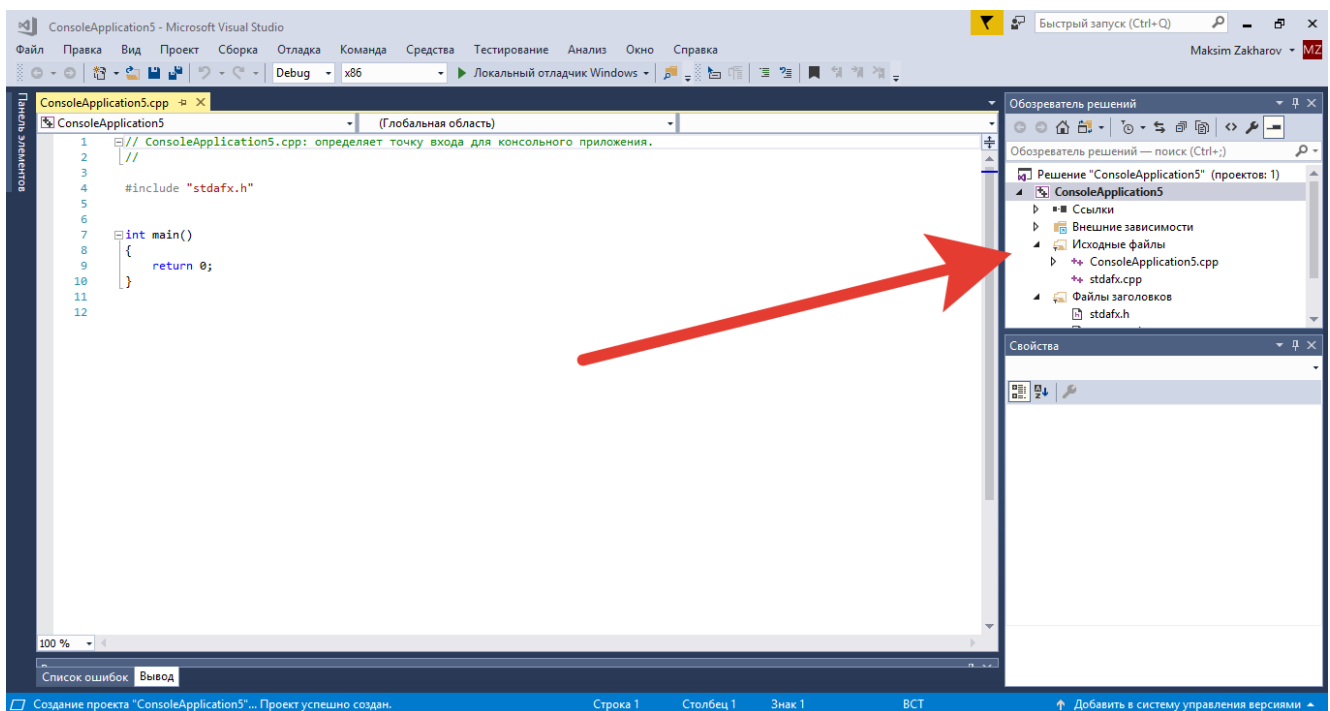


Рисунок 2.3 – Окно «Обозреватель решений»

### 3.2.3 Добавление программного кода в приложение.

Теперь необходимо добавить код для отображения, например, приветствия «Hello, world!\n» в окне консоли [3].

В исходном файле `ConsoleApplication.cpp` введите пустую строку перед строкой «`return 0;`», а затем введите в нее следующий код:

```
cout << "Hello, world!\n";
```

В тексте программы под «`cout`» появится красная волнистая линия. При наведении на нее отобразится сообщение об ошибке. Сообщение об ошибке также отобразится в окне «Список ошибок». Отобразить это окно можно, выбрав в строке меню Вид, Список ошибок.

Устраняется ошибка включением «`cout`» в файл заголовка `<iostream>`. Для включения заголовка `<iostream>` введите следующий код после `#include "stdafx.h"`:

```
#include <iostream>  
using namespace std;
```

После вставки этого кода появляется поле, предлагающее рекомендации для символов, которые были введены. Это поле является частью технологии C++ IntelliSense, обеспечивающей подсказки по коду, в том числе отображение членов класса или интерфейса и сведения о параметрах. Кроме того, можно использовать фрагменты кода в виде предопределенных блоков кода.

Красная волнистая линия под «`cout`» исчезнет после исправления ошибки (рисунок 2.4).

Сохраните изменения в файле.

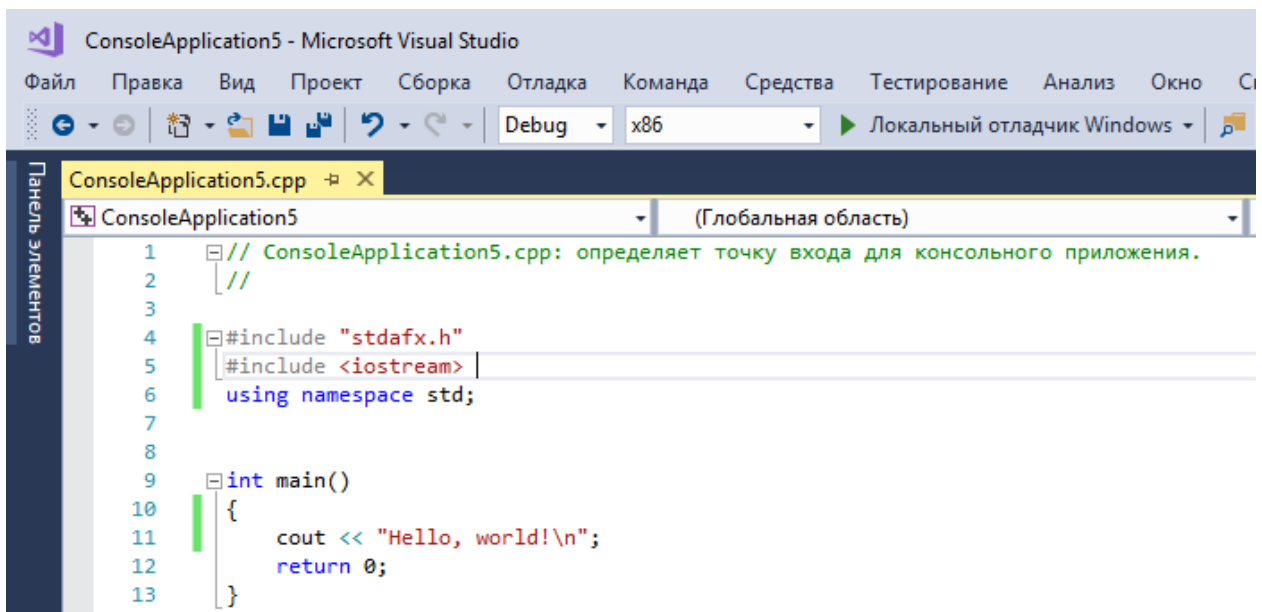


Рисунок 2.4 – Полный текст программы

### 3.2.4 Отладка и тестирование приложения.

С помощью отладки можно посмотреть, отображается ли заданная фраза в окне консоли. Для этого запустите отладчик нажав Отладчик / Начать отладку (рисунок 2.5).

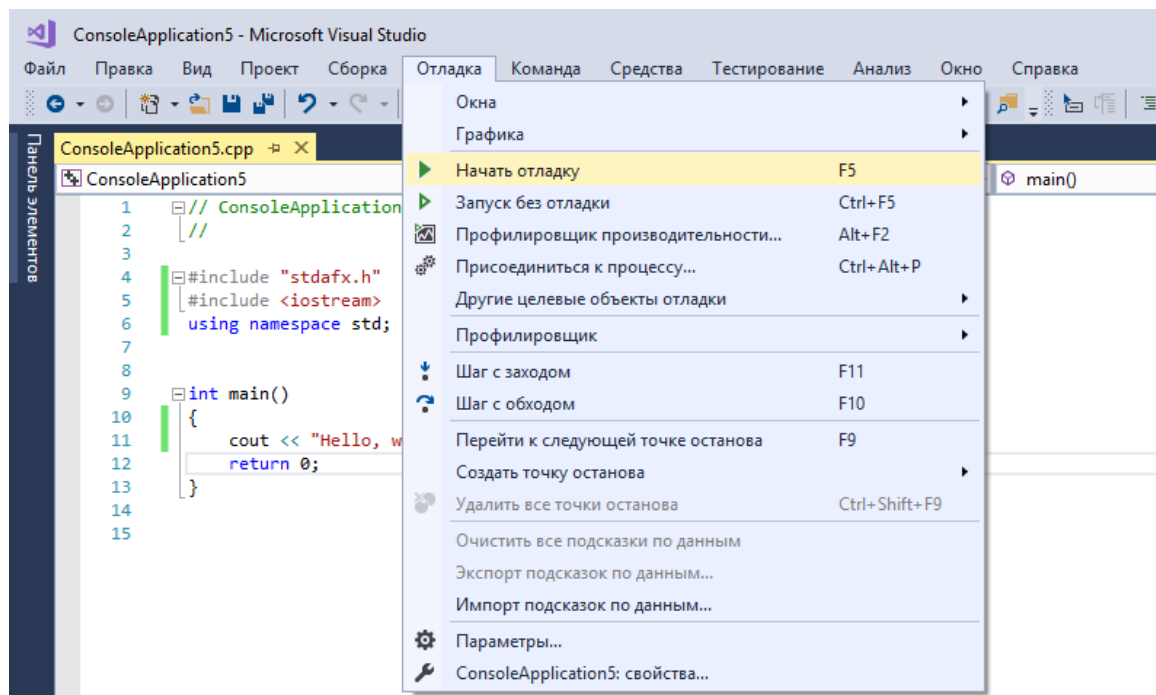


Рисунок 2.5 – Отладка программы

Отладчик запустится и выполнит код. Окно консоли (отдельное окно, подобное командной строке) отображается в течение нескольких секунд, но при остановке отладчика быстро закрывается. Чтобы просмотреть текст, необходимо установить точку останова выполнения программы.

Добавьте точку останова из меню в строке «return 0;». Для установки точки останова можно также просто щелкнуть область слева (рисунок 2.6).

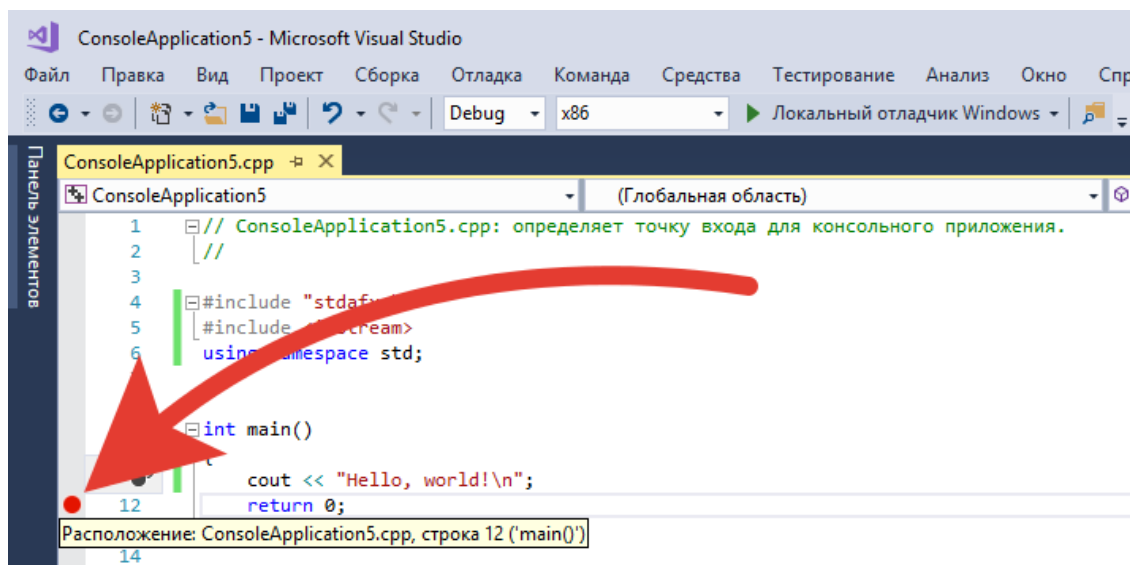


Рисунок 2.6 – Полный текст программы с точкой останова

Рядом со строкой кода в крайнем левом поле окна редактора появится красный кружок.

Нажмите клавишу F5, чтобы начать отладку. Запускается отладчик, появляется окно консоли, в котором выводится фраза «Hello, world!» (рисунок 2.7).

Внесите в программный код изменения в соответствии с индивидуальным заданием и повторно проведите отладку разработанного приложения.

Сохраните изменения в файле.

Следует отметить, что созданный проект находится по адресу: C:\Users\User\source\repos, где «User» – имя пользователя.

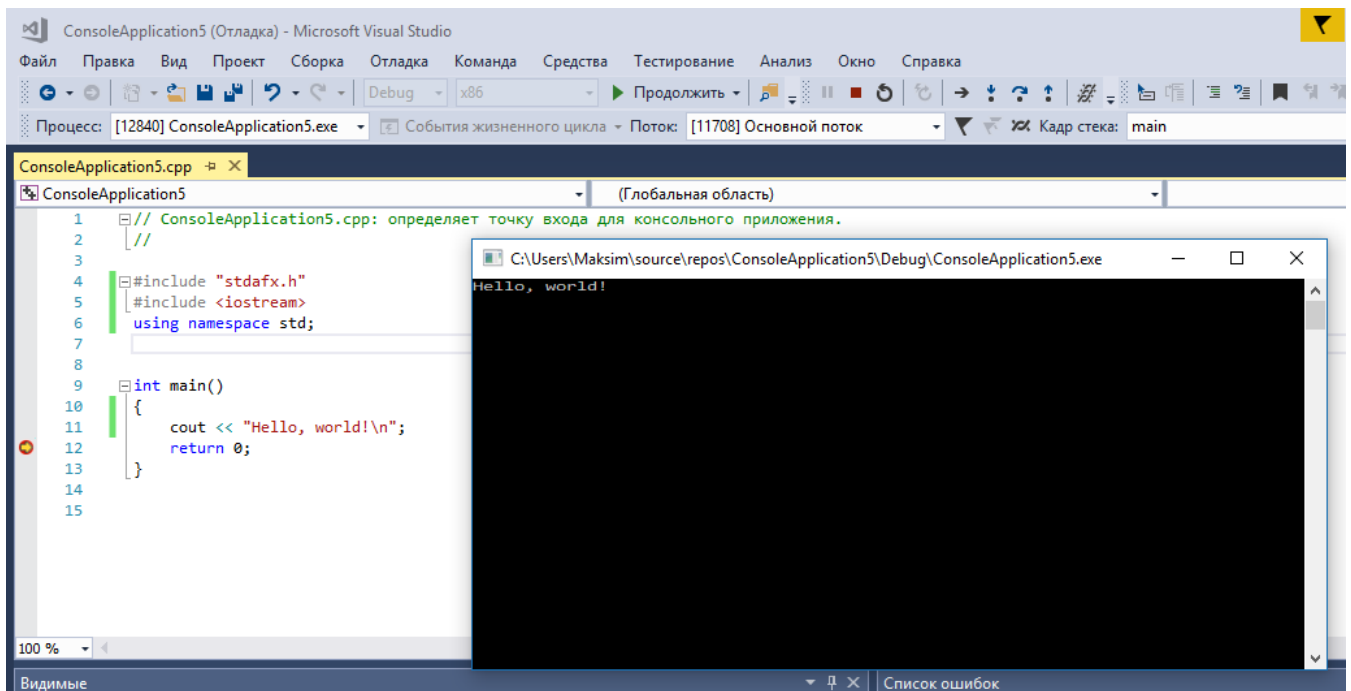


Рисунок 2.7 – Результат выполнения программы

### 3.2.5 Подготовка и защита отчета по лабораторной работе.

По результатам выполнения всех заданий подготовьте письменный отчет и представьте его для защиты лабораторной работы № 2. В ходе защиты отчета необходимо быть готовыми к демонстрации разработанного программного приложения и к ответам на контрольные вопросы преподавателя.

Примерный перечень контрольных вопросов:

- 1) Как создать консольное приложение Windows в Visual Studio?
- 2) Поясните порядок добавления программного кода в приложение Visual Studio?
- 3) Какие средства отображения сообщений об ошибках имеются в Visual Studio?
- 4) Поясните порядок отладки и тестирования приложения.
- 5) Как установить точку останова выполнения программы?

### **3.3 Лабораторная работа № 3. Разработка простого приложения, использующего графические библиотеки OpenGL в среде Microsoft Visual Studio**

Цели работы: приобрести практические навыки по созданию простого приложения, использующего библиотеки `opengl32.lib`, `glut32.lib`, `glu32.lib`, в Microsoft Visual Studio и закрепить знания по использованию программной среды при решении проектных задач.

Время: 2 часа.

Задание:

- подключить библиотеки OpenGL в Visual Studio;
- создать простое приложение в Microsoft Visual Studio для рисования фигуры белого квадрата с помощью средств и библиотек OpenGL;
- создать простое приложение в Microsoft Visual Studio, содержащее поверхность формы для отображения графики с помощью средств и библиотек OpenGL;
- внести изменения в разработанные приложения с учетом индивидуального задания;
- подготовить и защитить отчет по лабораторной работе.

Методические указания к выполнению задания.

3.3.1 Инструктаж и проверка готовности обучающихся к выполнению лабораторной работы № 3.

В начале занятия получите инструктаж преподавателя и допуск к работе. После чего получите индивидуальное задание на работу № 3.

3.3.2 Подключение библиотеки OpenGL в Visual Studio.

Запустите Visual Studio и создайте консольный проект (консольное приложение Windows) как в предыдущей лабораторной работе.

Подключите библиотеки OpenGL. Для этого нажмите кнопкой мыши на проект, вкладка «Свойства», как это показано на рисунке 3.1.

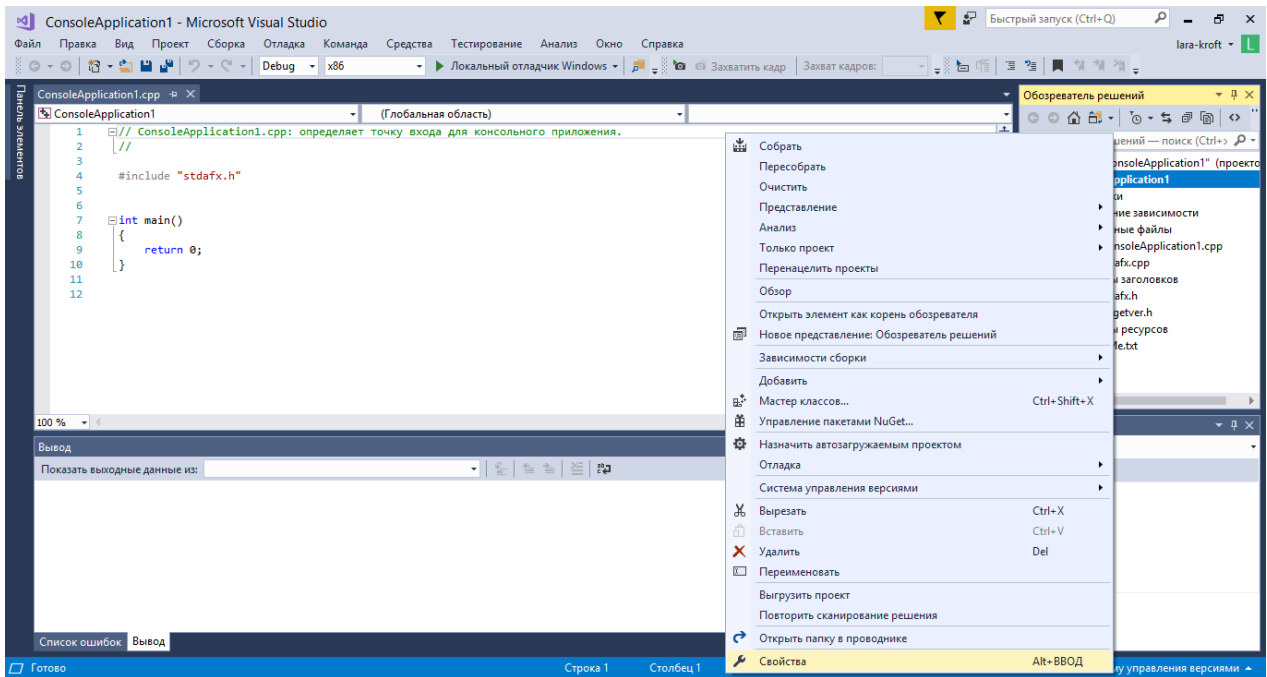


Рисунок 3.1 – Свойства проекта

В появившемся окне во вкладке «Конфигурации» выберите «Все конфигурации», как это показано на рисунке 3.2.

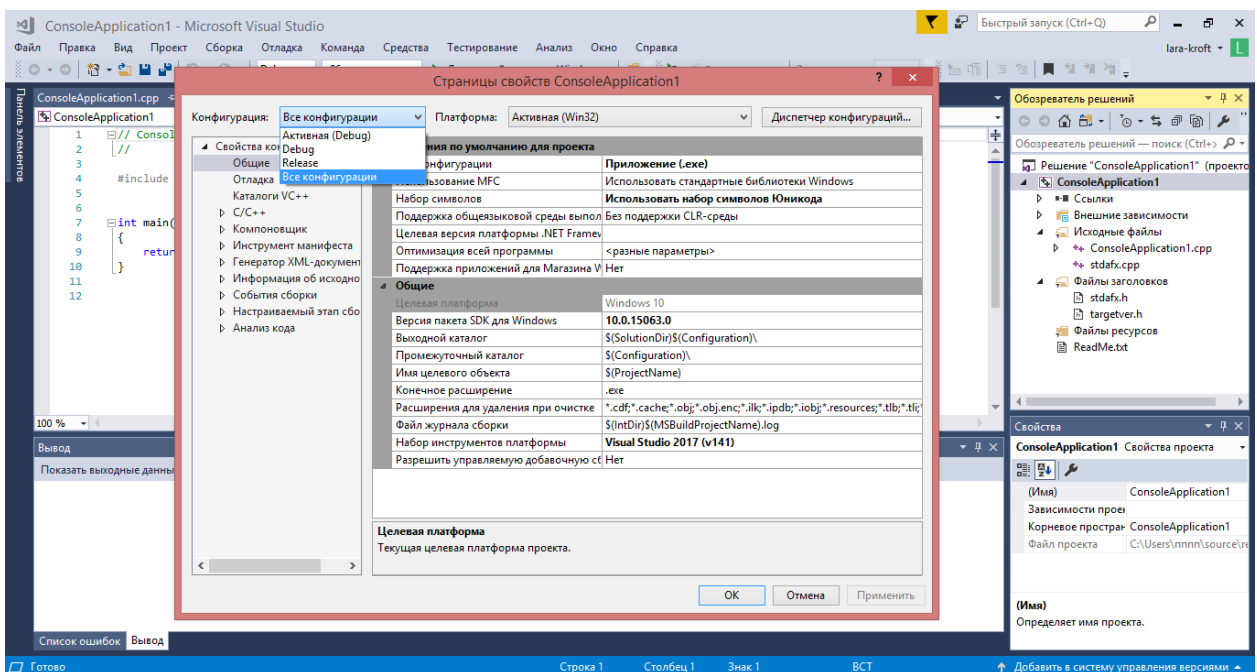


Рисунок 3.2 – Выбор конфигурации

Во вкладке «Компоновщик» выберите вкладку «Ввод» (рисунок 3.3).



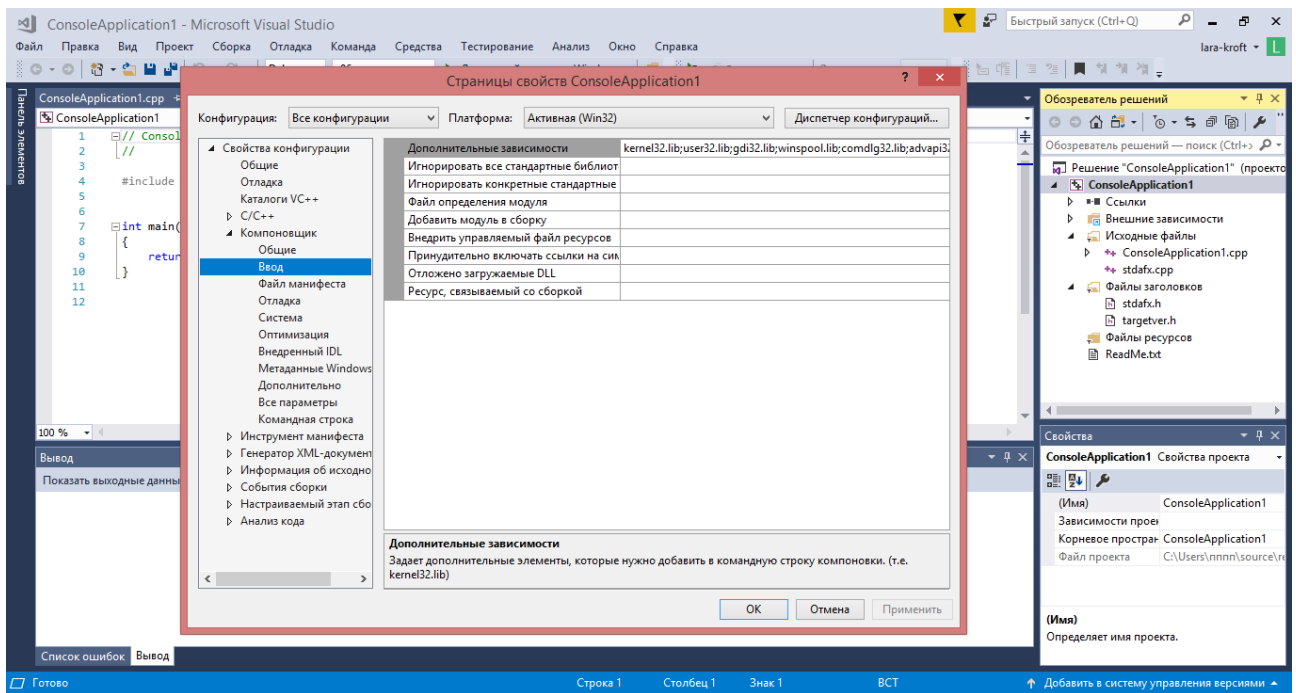


Рисунок 3.3 – Выбор вкладки «Ввод»

Справа в окне «Дополнительные зависимости» нажмите «Изменить». В появившемся окне вставьте библиотеки `opengl32.lib`, `glut32.lib`, `glu32.lib` и нажмите «ОК» и «Применить», как это показано на рисунке 3.4.

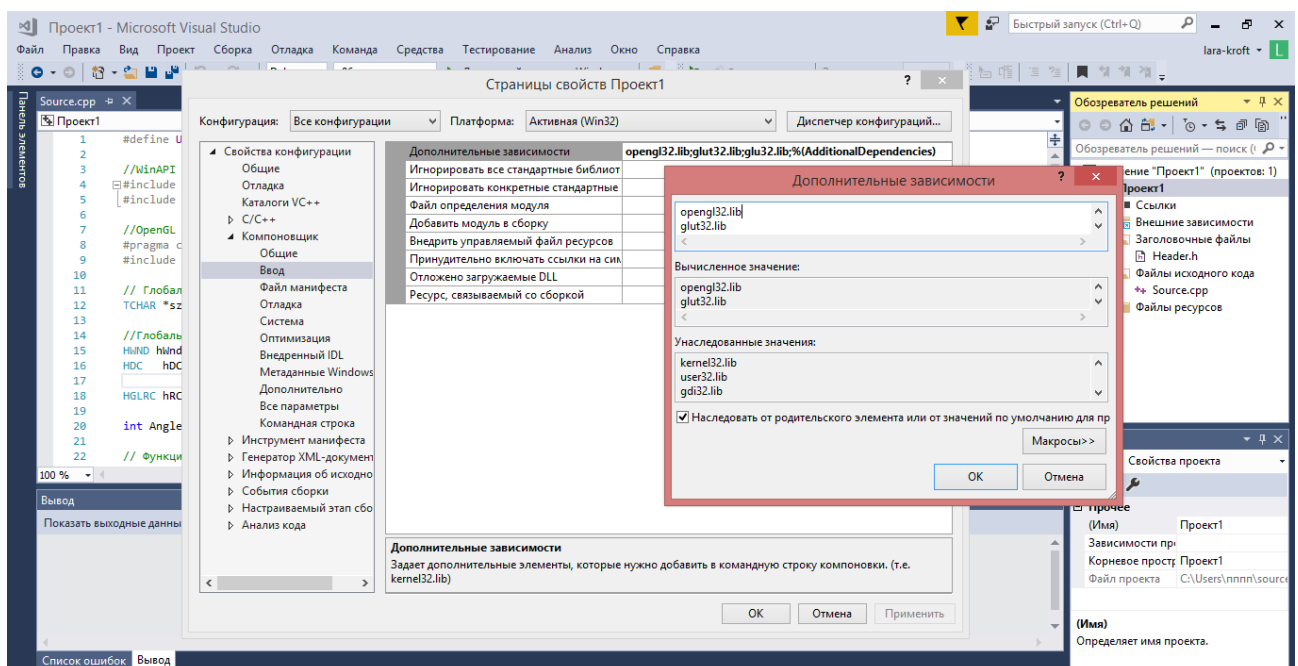


Рисунок 3.4 – Подключение библиотек OpenGL

В том же окне свойств выберите вкладку «Компоновщик», затем «Общие» (рисунок 3.5).

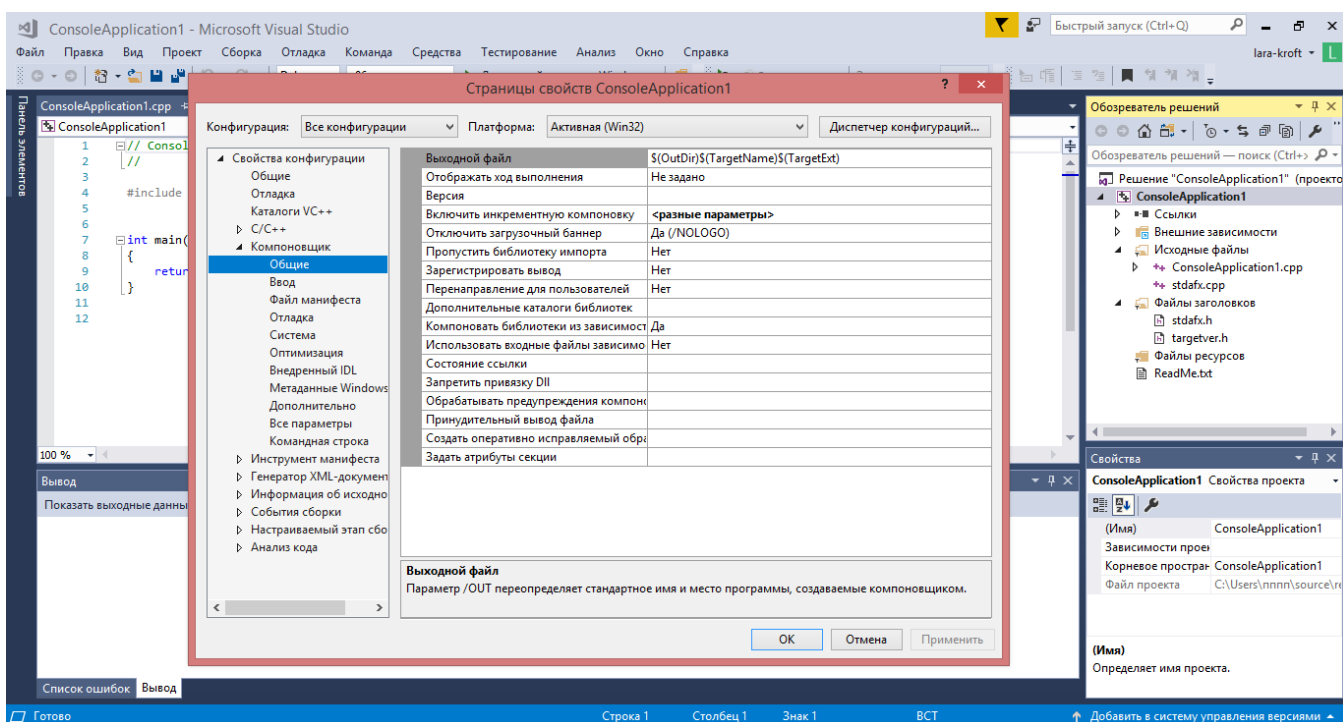


Рисунок 3.5 – Выбор вкладки «Общие»

В окне справа в строке «Дополнительные каталоги библиотек» укажите путь к библиотеке `glut32.lib`. Для этого введите строку: `C:\Program Files (x86)\Microsoft Visual Studio\2017\Community\VC\Tools\MSVC\14.11.25503\lib`, как показано на рисунке 3.6.

Таким образом, библиотеки OpenGL в Visual Studio подключены.

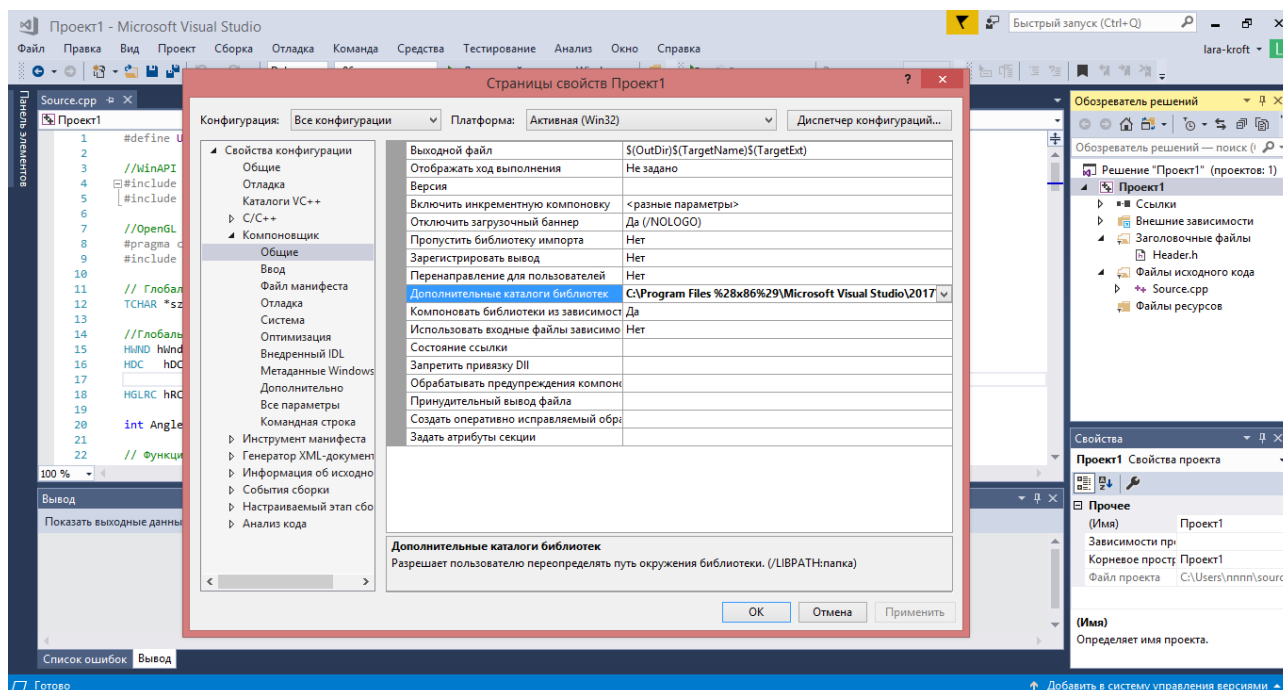


Рисунок 3.6 – Путь к библиотеке glut32.lib

### 3.3.3 Создание простого приложения в Microsoft Visual Studio для рисования фигуры белого квадрата с помощью средств и библиотек OpenGL.

Добавьте в приложение следующий программный код:

```
#include "stdafx.h"
#include <Windows.h>
#include <glut.h>

void Initialize()
{
    glClearColor(1.0, 0.0, 1.0, 1.0);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(0.0, 1.0, 0.0, 1.0, -1.0, 1.0);
}

void Draw()
{
    glColor3f(1.0, 1.0, 1.0);
    glBegin(GL_POLYGON);
    glVertex3f(0.25, 0.25, 0.0);
    glVertex3f(0.75, 0.25, 0.0);
    glVertex3f(0.75, 0.75, 0.0);
    glVertex3f(0.25, 0.75, 0.0);
    glEnd();
    glFlush();
}
```

```

int main(int argc, char **argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(400, 400);
    glutInitWindowPosition(100, 100);
    glutCreateWindow("Polygon");
    Initialize();
    glutDisplayFunc(Draw);
    glutMainLoop();
    return 0;
}

```

Запустите проект как в предыдущей лабораторной работе. В результате на экране получится белый квадрат как показано на рисунке 3.7.

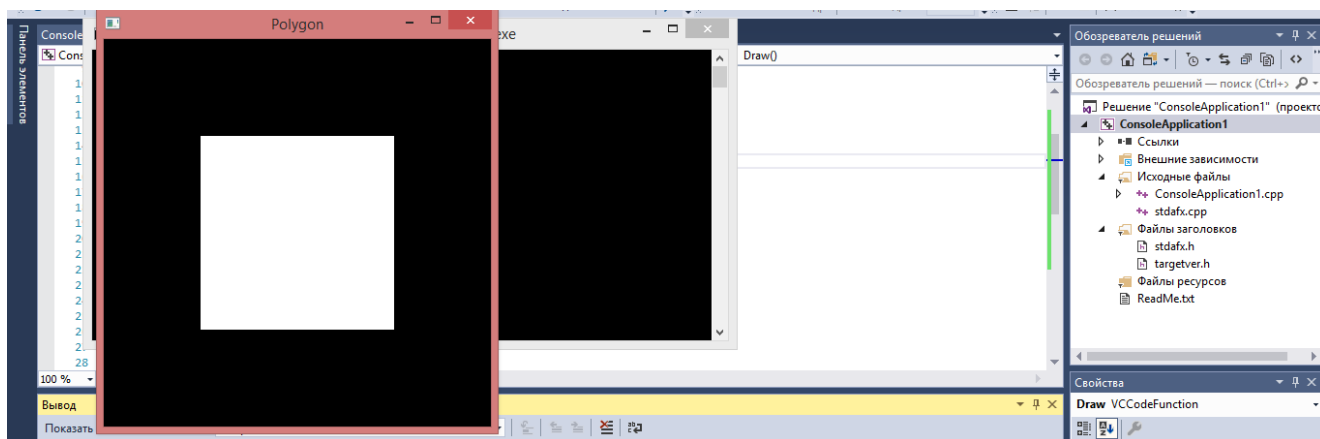


Рисунок 3.7 – Результат выполнения программы (белый квадрат)

Внесите в разработанный проект изменения в соответствии с индивидуальным заданием и проведите отладку приложения.

3.3.4 Создание простого приложения в Microsoft Visual Studio, содержащего поверхность формы для отображения графики с помощью средств и библиотек OpenGL.

Создайте проект (рисунок 3.8).

Выберите вкладку Шаблоны, Visual C++, Win32, Проект Win32 (классическое приложение Windows). Ниже в окне задайте имя и путь, где будет сохранен проект.

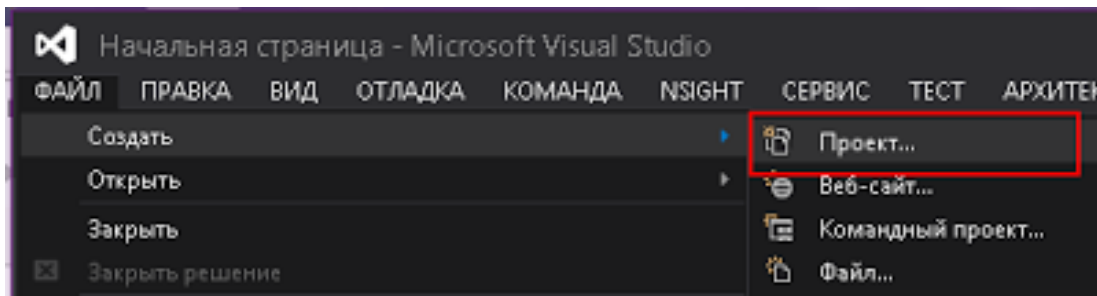


Рисунок 3.8 – Создание проекта

В окне «Мастер приложений Win32» нажмите Готово (рисунок 3.9).

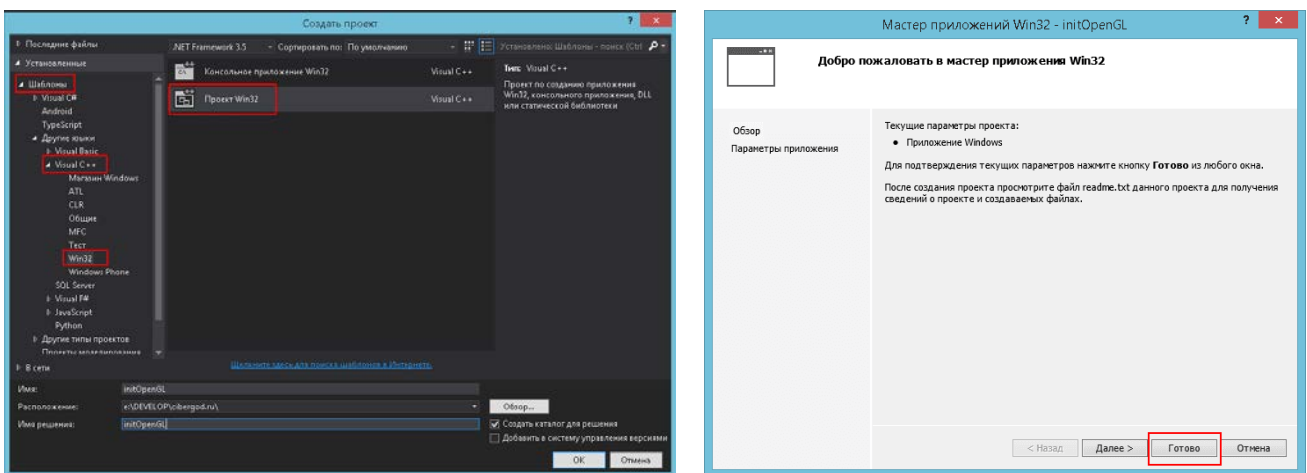


Рисунок 3.9 – Окно создания проекта и мастер приложений

Система сгенерирует огромный набор строк кода, в которых нелегко сразу разобраться. Поэтому для удобной навигации по проекту пользуйтесь Навигатором функций, который расположен над исходным текстом (рисунок 3.10).

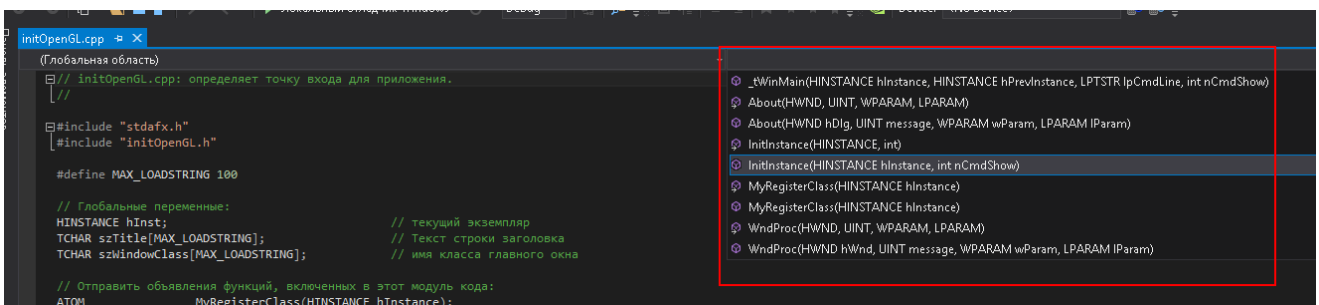


Рисунок 3.10 – Навигатор функций

Вначале добавьте нужные к проекту для работы с OpenGL ссылки. Для этого в «Обзревателе решений» выберите файл InitOpenGL и двойным щелчком мыши откройте его (рисунок 3.11).

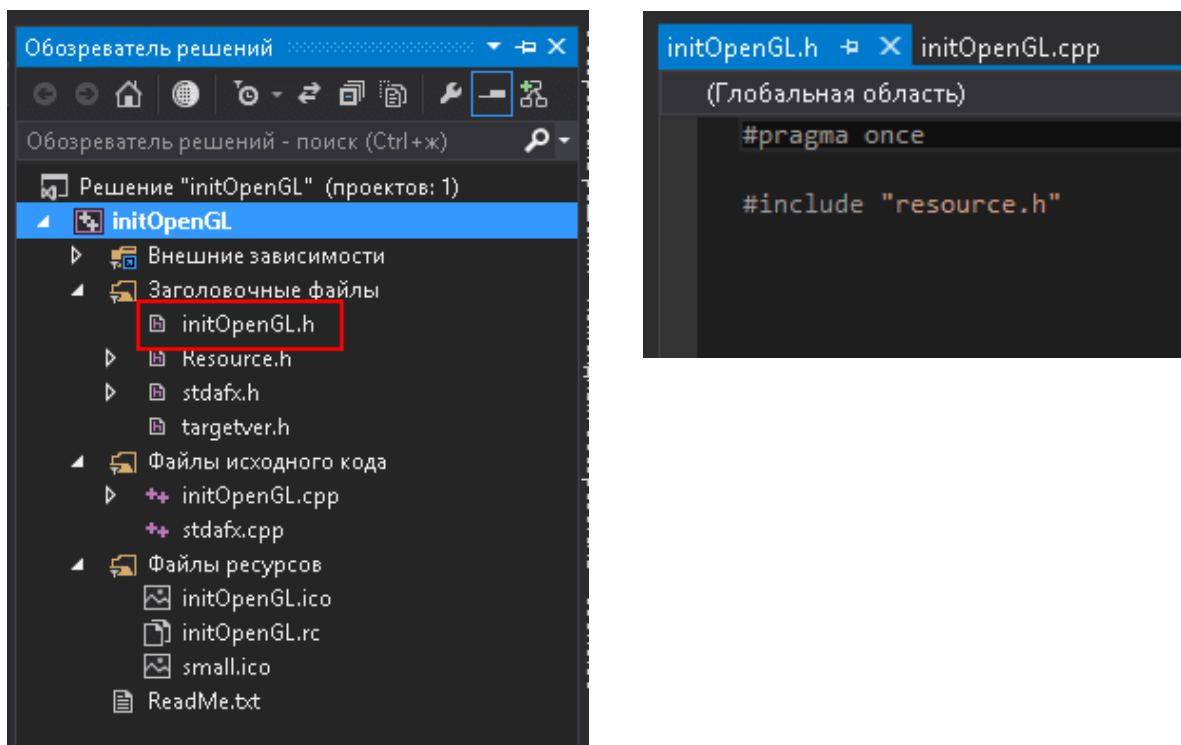


Рисунок 3.11 – Обзреватель решений

Модифицируйте файл, добавляя в него подключение библиотек и заголовочных файлов (рисунок 3.12).

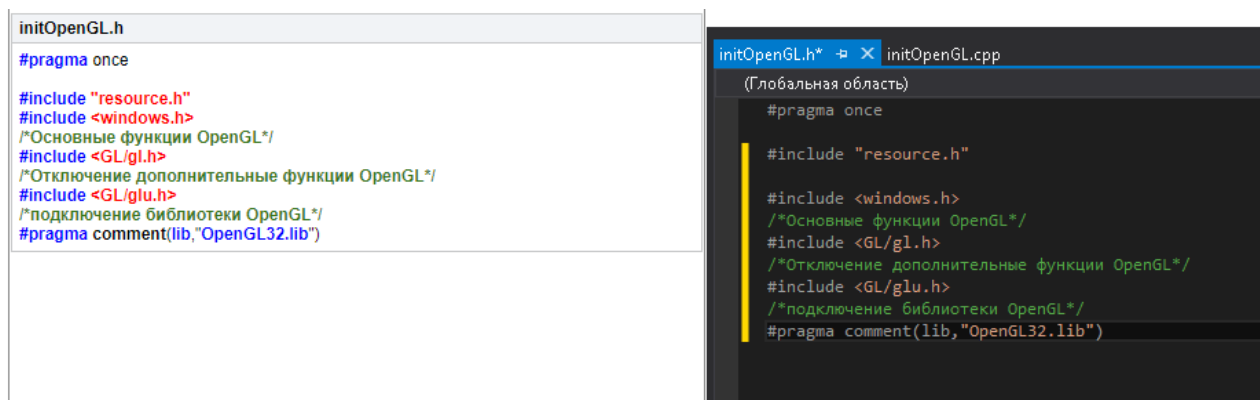


Рисунок 3.12 – Подключение библиотек и заголовочных файлов

Далее необходимо объявить набор переменных для инициализации графики на основе OpenGL. Перейдите обратно в файл `initOpenGL.cpp` и после списка подключаемых заголовочных файлов добавьте список переменных (рисунок 3.13).

```
переменные
/* Добавим свои переменные *****/
/*ссылка на поверхность рисования на которой будет все
рисование *****/
HDC hdc; /* *****/
/*ссылка на OpenGL через которую будем передавать параметр
*****/
HGLRC hGLRC; /* *****/
/*ссылка на окно в котором будет происходить рисование
*****/
HWND hWnd; /* *****/

initOpenGL.h*   initOpenGL.cpp*
(Глобальная область)
// initOpenGL.cpp: определяет точку входа для приложения.
//
#include "stdafx.h"
#include "initOpenGL.h"
/* Добавим свои переменные *****/
/*ссылка на поверхность рисования на которой будет все рисование *****/
HDC hdc; /* *****/
/*ссылка на OpenGL через которую будем передавать параметр *****/
HGLRC hGLRC; /* *****/
/*ссылка на окно в котором будет происходить рисование *****/
HWND hWnd; /* *****/
```

Рисунок 3.13 – Добавление списка переменных

Далее выберите в «Навигаторе функций» следующую строку:

`BOOL InitInstance(HINSTANCE hInstance, int nCmdShow).`

Она поможет быстро найти функцию, которая создает окно. В ней опишите дальнейшую инициализацию OpenGL. Модифицируйте функцию как показано на рисунке 3.14.

```
initOpenGL.h*   initOpenGL.cpp*
(Глобальная область)
wcecx.nIconSm = LoadIcon(wcecx.hInstance, MAKEINTRESOURCE(IDI_SMALL));

return RegisterClassEx(&wcecx);
}

// ФУНКЦИЯ: InitInstance(HINSTANCE, int)
// НАЗНАЧЕНИЕ: сохраняет обработку экземпляра и создает главное окно.
// КОММЕНТАРИИ:
//
// В данной функции дескриптор экземпляра сохраняется в глобальной переменной, а также
// создается и выводится на экран главное окно программы.
//
BOOL InitInstance(HINSTANCE hInstance, int nCmdShow)
{
    HWND hWnd;

    hInst = hInstance; // Сохранить дескриптор экземпляра в глобальной переменной

    hWnd = CreateWindow(szWindowClass, szTitle, WS_OVERLAPPEDWINDOW,
        CW_USEDEFAULT, 0, CW_USEDEFAULT, 0, NULL, NULL, hInstance, NULL);

    if (!hWnd)
    {
        return FALSE;
    }

    ShowWindow(hWnd, nCmdShow);
    UpdateWindow(hWnd);

    return TRUE;
}

@ _WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPTSTR lpCmdLine, int nCmdShow)
@ About(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)
@ InitInstance(HINSTANCE, int)
@ InitInstance(HINSTANCE hInstance, int nCmdShow)
@ MyRegisterClass(HINSTANCE hInstance)
@ MyRegisterClass(HINSTANCE hInstance)
@ WndProc(HWND, UINT, WPARAM, LPARAM)
@ WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)
```

Рисунок 3.14 – Модификация функции создания окна

Из этой функции уберите первую строку, так как переменную HWND hWnd уже объявили как глобальную (рисунок 3.15).

```
BOOL InitInstance(HINSTANCE hInstance, int nCmdShow)
{
    /*закомментируем строку
    HWND hWnd;
    */

    hInst = hInstance; // Сохранить дескриптор экземпляра в глобальной переменной

    hWnd = CreateWindow(szWindowClass, szTitle, WS_OVERLAPPEDWINDOW,
        CW_USEDEFAULT, 0, CW_USEDEFAULT, 0, NULL, NULL, hInstance, NULL);

    /*если функция не сможет создать окно то и создавать графику не за чем*/
    if (!hWnd)
    {
        return FALSE;
    }
    /*Весь код инициализации будет находиться здесь *****/

    /******/
    ShowWindow(hWnd, nCmdShow);
    UpdateWindow(hWnd);

    return TRUE;
}
```

Рисунок 3.15 – Удаление строки HWND hWnd

При создании окна система сама проверит помещается ли программный код создаваемой формы в память. Эта проверка происходит с помощью условия:

```
/*если функция не сможет создать окно, то и создавать графику незачем*/
if (!hWnd)
{
    return FALSE;
}
```

Если все хорошо, то функция продолжит работу, а если нет, то закончит ее. Поэтому после этой строчки продолжите задание параметров для создаваемого приложения.

Сначала добавьте описание формата точки, чтобы система определяла в каком виде отдавать их видеокарте и объяснить видеокарте в каком виде ее получать. Делается это заполнением структуры PIXELFORMATDESCRIPTOR.



Для этого добавьте описание структуры, представленное на рисунке 3.16.

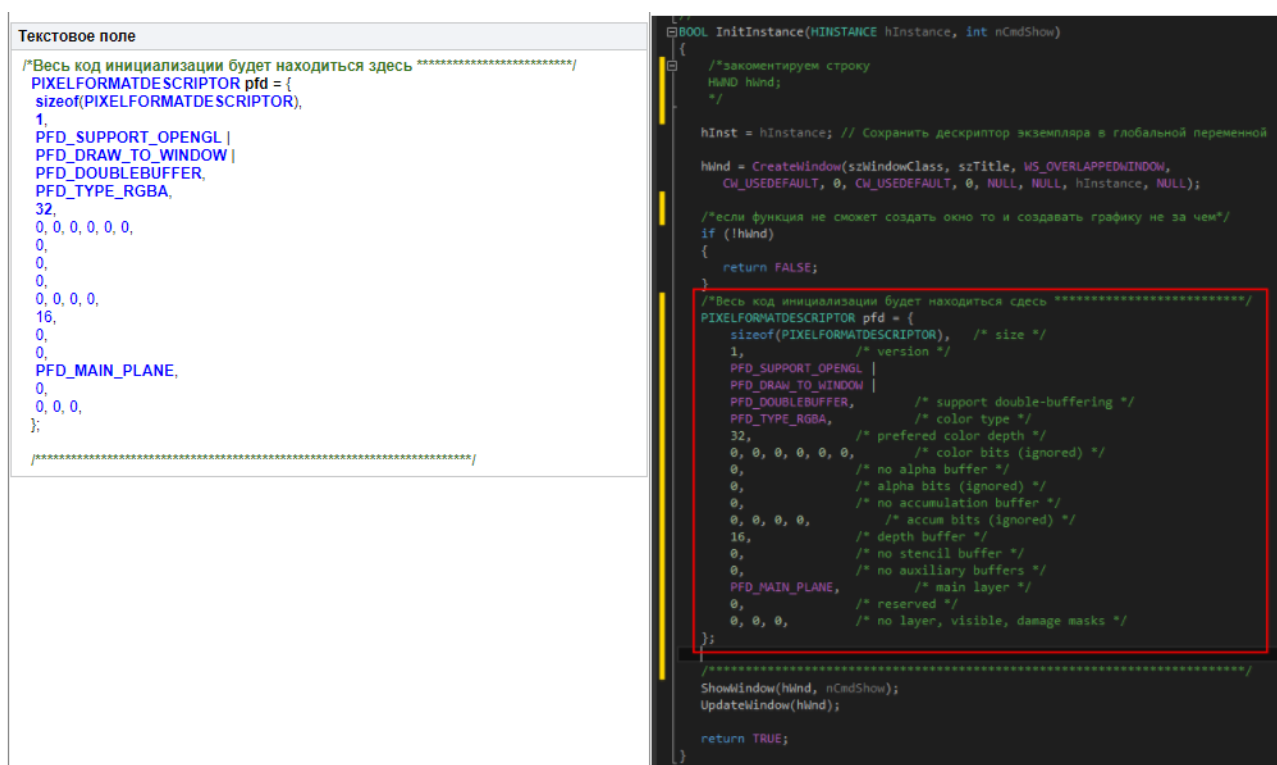


Рисунок 3.16 – Описание структуры PIXELFORMATDESCRIPTOR

Далее получите ссылку на окно, задайте формат точек рисования, укажите OpenGL, где будет отображаться графика. Затем создайте саму поверхность рисования в памяти (рисунок 3.17).

Следующим шагом выберите функцию `LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)`. Она отвечает за возникающие события в системе.

В этой функции описываются любые события связанные с приложением (рисунок 3.18).

<pre> <b>инициализация OpenGL</b>  /*получаем ссылку на поверхность где будем рисовать */ hDC = GetDC(hWnd); /*Задаем параметры точек*/ int pixelFormat = ChoosePixelFormat(hDC, &amp;pfid); /*задаем параметры точки*/ SetPixelFormat(hDC, pixelFormat, &amp;pfid); /*сообщаем OpenGL где будем рисовать*/ hGLRC = wglCreateContext(hDC); /*создаем контекст рисования*/ wglMakeCurrent(hDC, hGLRC); /***** </pre>	<pre> BOOL InitInstance(HINSTANCE hInstance, int nCmdShow) {     /*комментируем строку     HWnd hWnd;     */     hInst = hInstance; // Сохранить дескриптор экземпляра в глобальной переменной     hWnd = CreateWindow(szWindowClass, szTitle, WS_OVERLAPPEDWINDOW,         CW_USEDEFAULT, 0, CW_USEDEFAULT, 0, NULL, NULL, hInstance, NULL);     /*если функция не сможет создать окно то и создавать графику не за чем*/     if (!hWnd)     {         return FALSE;     }     /*Весь код инициализации будет находиться здесь *****/     PIXELFORMATDESCRIPTOR pfd = {         sizeof(PIXELFORMATDESCRIPTOR),         1,         PFD_SUPPORT_OPENGL           PFD_DRAW_TO_WINDOW           PFD_DOUBLEBUFFER,         PFD_TYPE_RGBA,         32,         0, 0, 0, 0, 0, 0,         0,         0, 0, 0, 0,         16,         0,         PFD_MAIN_PLANE,         0,         0, 0, 0,     };     /*получаем ссылку на поверхность где будем рисовать */     hDC = GetDC(hWnd);     /*Задаем параметры точек*/     int pixelFormat = ChoosePixelFormat(hDC, &amp;pfid);     /*задаем параметры точки*/     SetPixelFormat(hDC, pixelFormat, &amp;pfid);     /*сообщаем OpenGL где будем рисовать*/     hGLRC = wglCreateContext(hDC);     /*создаем контекст рисования*/     wglMakeCurrent(hDC, hGLRC);     /*****     ShowWindow(hWnd, nCmdShow);     UpdateWindow(hWnd);     return TRUE; </pre>
---	--

Рисунок 3.17 – Получение ссылки на поверхность, задание параметров точек

```

// ФУНКЦИЯ: WndProc(HWND, UINT, WPARAM, LPARAM)
// НАЗНАЧЕНИЕ: обрабатывает сообщения в главном окне.
// WM_COMMAND - обработка меню приложения
// WM_PAINT - Закрасить главное окно
// WM_DESTROY - ввести сообщение о выходе и вернуться.
//

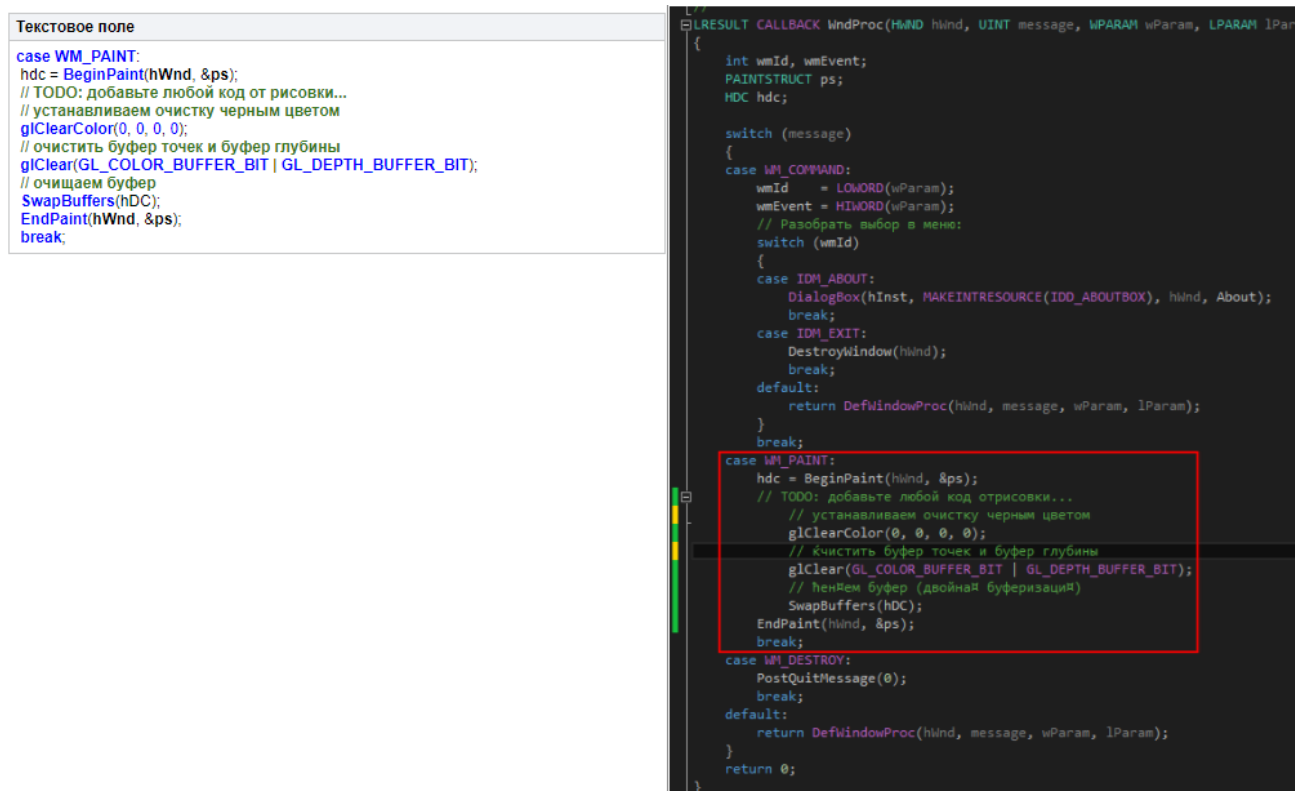
LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)
{
    int wmId, wmEvent;
    PAINTSTRUCT ps;
    HDC hdc;

    switch (message)
    {
    case WM_COMMAND:
        wmId = LOWORD(wParam);
        wmEvent = HIWORD(wParam);
        // Разобрать выбор в меню:
        switch (wmId)
        {
        case IDM_ABOUT:
            DialogBox(hInst, MAKEINTRESOURCE(IDD_ABOUTBOX), hWnd, About);
            break;
        case IDM_EXIT:
            DestroyWindow(hWnd);
            break;
        default:
            return DefWindowProc(hWnd, message, wParam, lParam);
        }
        break;
    case WM_PAINT:
        hdc = BeginPaint(hWnd, &ps);
        // TODO: добавьте любой код отрисовки...
        DrawGLScene();
        EndPaint(hWnd, &ps);
        break;
    case WM_DESTROY:
        PostQuitMessage(0);
        break;
    default:
        return DefWindowProc(hWnd, message, wParam, lParam);
    }
    return 0;
}

```

Рисунок 3.18 – Выбор функции LRESULT CALLBACK WndProc

В момент рисования окна происходит событие WM\_PAINT и окно обрабатывает всю логику, описанную в ветке case WM\_PAINT. Добавьте сюда вызов команд OpenGL. Для этого введем следующий код (рисунок 3.19).



```
Текстовое поле
case WM_PAINT:
hdc = BeginPaint(hWnd, &ps);
// TODO: добавьте любой код от рисовки...
// устанавливаем очистку черным цветом
glClearColor(0, 0, 0, 0);
// очистить буфер точек и буфер глубины
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
// очищаем буфер
SwapBuffers(hdc);
EndPaint(hWnd, &ps);
break;

LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)
{
    int wmId, wmEvent;
    PAINTSTRUCT ps;
    HDC hdc;

    switch (message)
    {
        case WM_COMMAND:
            wmId = LOWORD(wParam);
            wmEvent = HIWORD(wParam);
            // Разобрать выбор в меню:
            switch (wmId)
            {
                case IDM_ABOUT:
                    DialogBox(hInst, MAKEINTRESOURCE(IDD_ABOUTBOX), hWnd, About);
                    break;
                case IDM_EXIT:
                    DestroyWindow(hWnd);
                    break;
                default:
                    return DefWindowProc(hWnd, message, wParam, lParam);
            }
            break;
        case WM_PAINT:
            hdc = BeginPaint(hWnd, &ps);
            // TODO: добавьте любой код отрисовки...
            // устанавливаем очистку черным цветом
            glClearColor(0, 0, 0, 0);
            // Кистить буфер точек и буфер глубины
            glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
            // Меняем буфер (двойная буферизация)
            SwapBuffers(hdc);
            EndPaint(hWnd, &ps);
            break;
        case WM_DESTROY:
            PostQuitMessage(0);
            break;
        default:
            return DefWindowProc(hWnd, message, wParam, lParam);
    }
    return 0;
}
```

Рисунок 3.19 – Добавление вызова команд OpenGL

Теперь запустите приложение и убедитесь, что форма окрасится в черный цвет.

Внесите в разработанный проект изменения в соответствии с индивидуальным заданием и проведите отладку приложения.

### 3.3.5 Подготовка и защита отчета по лабораторной работе.

По результатам выполнения всех заданий подготовьте письменный отчет и представьте его для защиты лабораторной работы № 3. В ходе защиты отчета необходимо быть готовыми к демонстрации разработанных программных приложений и к ответам на контрольные вопросы преподавателя.

Примерный перечень контрольных вопросов:

1) Порядок подключения библиотек OpenGL в Visual Studio.

2) Порядок создания консольного проекта (консольного приложения Windows) в Microsoft Visual Studio.

3) Порядок создания приложения в Microsoft Visual Studio, содержащего поверхность формы для отображения графики с помощью средств и библиотек OpenGL.

4) Назовите средство в Visual Studio, обеспечивающее удобную навигацию по проекту, перечислите его возможности.

5) Какая функция отвечает за возникающие события в системе и описывающая любые события, связанные с приложением?

### **3.4 Лабораторная работа № 4. Разработка приложений, использующих графические библиотеки OpenGL в среде Microsoft Visual Studio (моделирование линий)**

Цели работы: приобрести практические навыки по созданию различных приложений, использующих графические библиотеки OpenGL в среде Microsoft Visual Studio и закрепить знания по использованию программной среды при решении проектных задач.

Время: 2 часа.

Задание:

- подключить библиотеки OpenGL в Visual Studio;
- создать приложение, использующее OpenGL в Microsoft Visual Studio и рисующее точку, линию, ломаную линию, кривую и замкнутую кривую;
- внести изменения в разработанное приложение с учетом индивидуального задания;
- подготовить и защитить отчет по лабораторной работе.

Методические указания к выполнению задания.

3.4.1 Инструктаж и проверка готовности обучающихся к выполнению лабораторной работы № 4.

В начале занятия получите инструктаж преподавателя и допуск к работе. После чего получите индивидуальное задание на работу № 4.

3.4.2 Подключение библиотеки OpenGL в Visual Studio.

Запустите Visual Studio, создайте пустой проект. Для этого нажмите правой кнопкой мыши на проект и добавьте файл C++ и файл заголовка.

По аналогии с предыдущей лабораторной работой подключите библиотеки OpenGL: opengl32.lib; glut32.lib; glu32.lib. А также укажите путь к библиотеке glut32.lib: C:\Program Files (x86)\Microsoft Visual Studio\2017\Community\VC\Tools\MSVC\14.11.25503\lib.

3.4.3 Создание приложения, использующего OpenGL в Microsoft Visual Studio и рисующего точку, линию, ломаную линию, кривую и замкнутую кривую.

Нарисуйте красную линию, красную точку, белую ломанную линию, толщиной 4 пикселя, и кривую линию (формула кривой:  $y = x^2$ ). Для этого введите код программы:

```
#define UNICODE
// WinAPI
#include <Windows.h>
#include <tchar.h>
// OpenGL
#pragma comment(lib, "opengl32.lib")
// GLUT
#pragma comment(lib, "glut32.lib")
#include "glut.h"
// Other
#include <math.h>
int Angle = 0;
// Функции для работы с OpenGL/GLUT
void GLInit();
void GLRenderScene();
void GLKeyDown(unsigned char key, int x, int y);
int WINAPI wWinMain(HINSTANCE hInst, HINSTANCE hPrevInst, LPWSTR lpCmdLine,
int nShowCmd)
{
    // Аргументы командной строки (путь к EXE)
    char *argv0 = new char[512];
    GetModuleFileNameA(0, argv0, 512);
```

```

int argc = 1;
// Инициализация GLUT
glutInit(&argc, &argv0);
glutInitDisplayMode(GLUT_DOUBLE | GLUT_SINGLE | GLUT_RGB);
// Координаты и размер окна GLUT
glutInitWindowPosition(200, 200);
glutInitWindowSize(400, 400);
// Создание окна
glutCreateWindow("");
// Обработчик рендеринга
glutDisplayFunc(GLRenderScene);
// Обработчик клавиатуры
glutKeyboardFunc(GLKeyDown);
// Инициализация OpenGL
GLInit();

// Главный цикл
glutMainLoop();
return 0;
}
void GLInit()
{
    // Цвет фона - черный
    glClearColor(0.0, 0.0, 0.0, 1.0);
}
void GLRenderScene()
{
    // Очищаем буферы
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glLoadIdentity();
    // Поворот
    glRotatef(Angle, 1.0, 1.0, 0.0);
    // --- Линия ---
    glColor3f(1.0, 0.0, 0.0);
    glBegin(GL_LINES);
    glVertex3f(0.3, -0.2, 0.0);
    glVertex3f(-0.9, -0.22, 0.0);
    glEnd();
    // --- Точка ---
    glBegin(GL_POINTS);
    glVertex3f(0.5, 0.5, 0.0);
    glEnd();
    // --- Ломаная ---
    glColor3f(1.0, 1.0, 1.0);
    glLineWidth(4); // Толщина линии - 4 пкс
    glBegin(GL_LINES); // Или GL_LINE_STRIP
    // Фрагмент 1
    glVertex3f(0.0, 0.0, 0.0); // Начало фрагмента 1
    glVertex3f(0.5, 0.0, 0.0); // Конец фрагмента 1
    // Фрагмент 2
    glVertex3f(0.5, 0.0, 0.0); // Начало 2. При GL_LINE_STRIP можно не задавать
    glVertex3f(0.3, 0.5, 0.0); // Конец 2
}

```

```

glEnd();
// --- Кривая ---
glColor3f(0.0, 1.0, 0.0);
glLineWidth(1); // Толщина линии - снова 1 пкс
glBegin(GL_LINE_STRIP); // Или GL_LINE_LOOP - для замкнутой фигуры, или
GL_POLYGON - для замкнутой закрашенной фигуры
float x = -0.1;
float y;
for (x; x > -0.5; x-=0.001)
{
    y = x * x;
    glVertex3f(x, y, 0.0);
}
glEnd();
glutSwapBuffers();
}
void GLKeyDown(unsigned char key, int x, int y)
{
    Angle += 5; // Увеличиваем угол поворота на 5 градусов
    glutPostRedisplay(); // Перерисовываем окно
}

```

Запустите проект и убедитесь в результате выполнения программы (рисунок 4.1).

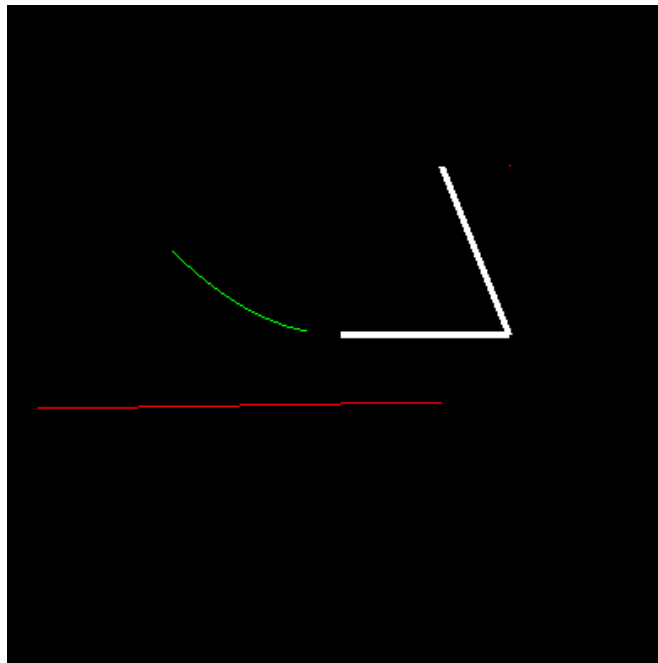


Рисунок 4.1 – Результат выполнения программы

Внесите в разработанный проект изменения в соответствие с индивидуальным заданием и проведите отладку приложения.

#### 3.4.4 Подготовка и защита отчета по лабораторной работе.

По результатам выполнения всех заданий подготовьте письменный отчет и представьте его для защиты лабораторной работы № 4. В ходе защиты отчета необходимо быть готовыми к демонстрации разработанного программного приложения и к ответам на контрольные вопросы преподавателя.

Примерный перечень контрольных вопросов:

- 1) Как подключить библиотеки OpenGL в Visual Studio?
- 2) Порядок создания приложения, рисующего точку и линию в Microsoft Visual Studio.
- 3) Порядок создания приложения, рисующего ломаную линию в Microsoft Visual Studio.
- 4) Порядок создания приложения, рисующего кривую и замкнутую кривую в Microsoft Visual Studio.

### **3.5 Лабораторная работа № 5. Разработка приложений, использующих графические библиотеки OpenGL в среде Microsoft Visual Studio (построение поверхностей и сложных объектов, наложение текстуры на примитивы)**

Цели работы: приобрести практические навыки по созданию различных приложений, использующих графические библиотеки OpenGL в среде Microsoft Visual Studio и закрепить знания по использованию программной среды при решении проектных задач.

Время: 2 часа.

Задание:

- подключить библиотеки OpenGL в Visual Studio;
- создать приложение, использующее OpenGL в Microsoft Visual Studio и осуществляющее построение поверхностей и сложных объектов;



– внести изменения в разработанное приложение с учетом индивидуального задания;

– подготовить и защитить отчет по лабораторной работе.

Методические указания к выполнению задания.

3.5.1 Инструктаж и проверка готовности обучающихся к выполнению лабораторной работы № 5.

В начале занятия получите инструктаж преподавателя и допуск к работе. После чего получите индивидуальное задание на работу № 5.

3.5.2 Подключение библиотек OpenGL в Visual Studio.

По аналогии с предыдущей лабораторной работой запустите Visual Studio, создайте пустой проект и подключите библиотеки OpenGL.

3.5.3 Создание приложения, использующего OpenGL в Microsoft Visual Studio и осуществляющего построение поверхностей и сложных объектов.

Нарисуйте параллелепипед с разноцветными гранями в формациях квадрата. Для этого введите код программы:

```
#define UNICODE
// WinAPI
#include <Windows.h>
#include <tchar.h>
// OpenGL
#pragma comment(lib, "opengl32.lib")
// GLUT
#pragma comment(lib, "glut32.lib")
#include "glut.h"
// Other
#include <math.h>
int Angle = 0;
// Функции для работы с OpenGL/GLUT
void GLInit();
void GLRenderScene();
void GLKeyDown(unsigned char key, int x, int y);
int WINAPI wWinMain(HINSTANCE hInst, HINSTANCE hPrevInst, LPWSTR lpCmdLine,
int nShowCmd)
{
    // Аргументы командной строки (путь к EXE)
    char *argv0 = new char[512];
    GetModuleFileNameA(0, argv0, 512);
    int argc = 1;
    // Инициализация GLUT
    glutInit(&argc, &argv0);
```

```

glutInitDisplayMode(GLUT_DOUBLE | GLUT_SINGLE | GLUT_RGB);
// Координаты и размер окна GLUT
glutInitWindowPosition(200, 200);
glutInitWindowSize(400, 400);
// Создание окна
glutCreateWindow("");
// Обработчик рендеринга
glutDisplayFunc(GLRenderScene);
// Обработчик клавиатуры
glutKeyboardFunc(GLKeyDown);
// Инициализация OpenGL
GLInit();
// Главный цикл
glutMainLoop();
return 0;
}
void GLInit()
{
    // Цвет фона - черный
    glClearColor(0.0, 0.0, 0.0, 1.0);
}
void GLRenderScene()
{
    // Очищаем буферы
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glLoadIdentity();
    // Включаем тест глубины,
    // чтобы грани 3D-фигур - рисовались непрозрачными, и не просвечивали
    glEnable(GL_DEPTH_TEST);
    // Поворот
    glRotatef(Angle, 1.0, 1.0, 0.0);
    // ----- Рисование параллелепипеда -----
    // Длина x высота x глубина:
    // 0.5  1.0  0.25
    // Передняя грань
    glColor3f(0.0, 1.0, 0.0); // зеленый
    glBegin(GL_POLYGON);
    glVertex3f(0, 0, 0); // ЛН
    glVertex3f(0, 1.0, 0); // ЛВ
    glVertex3f(0.5, 1.0, 0); // ПВ
    glVertex3f(0.5, 0, 0); // ПН
    glEnd();
    // Задняя грань
    // Та же передняя, только, Z = 0.25, а, не 0
    glColor3f(1.0, 0.0, 0.0); // красный
    glBegin(GL_POLYGON);
    glVertex3f(0, 0, 0.25); // ЛН
    glVertex3f(0, 1.0, 0.25); // ЛВ
    glVertex3f(0.5, 1.0, 0.25); // ПВ
    glVertex3f(0.5, 0, 0.25); // ПН
    glEnd();
}

```

```

// Левая грань
glColor3f(1.0, 1.0, 0.0); // желтый
glBegin(GL_POLYGON);
glVertex3f(0, 0, 0.0); // Перед-низ
glVertex3f(0, 1.0, 0.0); // Перед-верх
glVertex3f(0, 1.0, 0.25); // Зад-верх
glVertex3f(0, 0, 0.25); // Зад-низ
glEnd();
// Правая грань
// Та же левая, только, X = 0.5
glColor3f(0.0, 1.0, 1.0); // Голубой
glBegin(GL_POLYGON);
glVertex3f(0.5, 0, 0.0); // Перед-низ
glVertex3f(0.5, 1.0, 0.0); // Перед-верх
glVertex3f(0.5, 1.0, 0.25); // Зад-верх
glVertex3f(0.5, 0, 0.25); // Зад-низ
glEnd();
// Верхняя грань
glColor3f(1.0, 0.0, 1.0); // Розовый
glBegin(GL_POLYGON);
glVertex3f(0, 1.0, 0.0); // Перед-лево
glVertex3f(0.5, 1.0, 0.0); // Перед-право
glVertex3f(0.5, 1.0, 0.25); // Зад-право
glVertex3f(0, 1.0, 0.25); // Зад-лево
glEnd();
// Нижняя грань
// Та же верхняя, только, Y = 0
glColor3f(1.0, 1.0, 1.0); // Белый
glBegin(GL_POLYGON);
glVertex3f(0, 0.0, 0.0); // Перед-лево
glVertex3f(0.5, 0.0, 0.0); // Перед-право
glVertex3f(0.5, 0.0, 0.25); // Зад-право
glVertex3f(0, 0.0, 0.25); // Зад-лево
glEnd();
glutSwapBuffers();
}
void GLKeyDown(unsigned char key, int x, int y)
{
    Angle += 5; // Увеличиваем угол поворота на 5 градусов
    glutPostRedisplay(); // Перерисовываем окно
}

```

Запустите проект. Результат выполнения программы показан на рисунке 5.1.

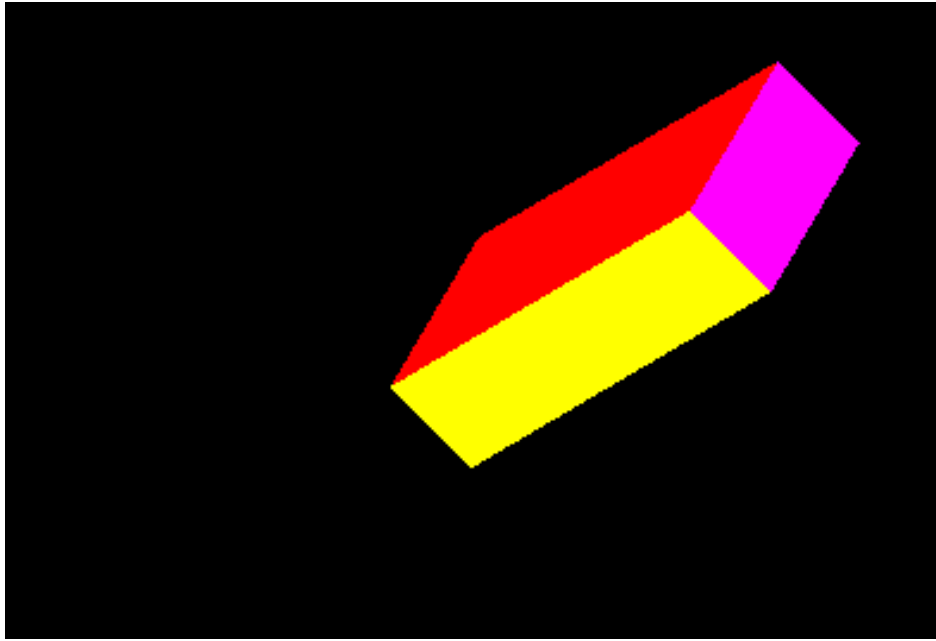


Рисунок 5.1 – Результат выполнения программы

Измените предыдущую программу. Грани параллелепипеда залейте разными градиентами, вместо монотонной заливки. Рисуйте в полноэкранном режиме, вместо оконного. Введите код программы:

```
#define UNICODE
// WinAPI
#include <Windows.h>
#include <tchar.h>
// OpenGL
#pragma comment(lib, "opengl32.lib")
// GLUT
#pragma comment(lib, "glut32.lib")
#include "glut.h"
// Other
#include <math.h>
int Angle = 0;
// Функции для работы с OpenGL/GLUT
void GLInit();
void GLRenderScene();
void GLKeyDown(unsigned char key, int x, int y);
int WINAPI wWinMain(HINSTANCE hInst, HINSTANCE hPrevInst, LPWSTR lpCmdLine,
int nShowCmd)
{
    // Аргументы командной строки (путь к EXE)
    char *argv0 = new char[512];
    GetModuleFileNameA(0, argv0, 512);
    int argc = 1;
```

```

// Инициализация GLUT
glutInit(&argc, &argv0);
glutInitDisplayMode(GLUT_DOUBLE | GLUT_SINGLE | GLUT_RGB);
// Создание окна
glutCreateWindow("");
// Развертывание созданного окна в полный экран
glutFullScreen();
// Обработчик рендеринга
glutDisplayFunc(GLRenderScene);
// Обработчик клавиатуры
glutKeyboardFunc(GLKeyDown);
// Инициализация OpenGL
GLInit();
// Главный цикл
glutMainLoop();
return 0;
}
void GLInit()
{
    // Цвет фона - черный
    glClearColor(0.0, 0.0, 0.0, 1.0);
}
void GLRenderScene()
{
    // Очищаем буферы
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glLoadIdentity();
    // Включаем тест глубины,
    // чтобы грани 3D-фигур - рисовались непрозрачными
    glEnable(GL_DEPTH_TEST);
    // Поворот
    glRotatef(Angle, 1.0, 1.0, 0.0);
    // ----- Рисование параллелепипеда -----
    // Длина x высота x глубина:
    // 0.5  1.0  0.25
    // Передняя грань
    glBegin(GL_POLYGON);
    glColor3f(0.0, 1.0, 0.0); // зеленый
    glVertex3f(0, 0, 0); // ЛН
    glColor3f(1.0, 0.0, 0.0); // красный
    glVertex3f(0, 1.0, 0); // ЛВ
    glColor3f(1.0, 0.0, 0.0); // красный
    glVertex3f(0.5, 1.0, 0); // ПВ
    glColor3f(0.0, 1.0, 0.0); // зеленый
    glVertex3f(0.5, 0, 0); // ПН
    glEnd();
    // Задняя грань
    // Та же передняя, только, Z = 0.25, а, не 0
    glBegin(GL_POLYGON);
    glColor3f(0.0, 1.0, 0.0); // зеленый
    glVertex3f(0, 0, 0.25); // ЛН
    glColor3f(1.0, 0.0, 0.0); // красный

```

```

glVertex3f(0, 1.0, 0.25); // ЛВ
glColor3f(1.0, 0.0, 0.0); // красный
glVertex3f(0.5, 1.0, 0.25); // ПВ
glColor3f(0.0, 1.0, 0.0); // зеленый
glVertex3f(0.5, 0, 0.25); // ПН
glEnd();
// Левая грань
glBegin(GL_POLYGON);
glColor3f(0.0, 1.0, 0.0); // зеленый
glVertex3f(0, 0, 0.0); // Перед-низ
glColor3f(1.0, 0.0, 0.0); // красный
glVertex3f(0, 1.0, 0.0); // Перед-верх
glColor3f(1.0, 0.0, 0.0); // красный
glVertex3f(0, 1.0, 0.25); // Зад-верх
glColor3f(0.0, 1.0, 0.0); // зеленый
glVertex3f(0, 0, 0.25); // Зад-низ
glEnd();
// Правая грань
// Та же левая, только, X = 0.5
glBegin(GL_POLYGON);
glColor3f(0.0, 1.0, 0.0); // зеленый
glVertex3f(0.5, 0, 0.0); // Перед-низ
glColor3f(1.0, 0.0, 0.0); // красный
glVertex3f(0.5, 1.0, 0.0); // Перед-верх
glColor3f(1.0, 0.0, 0.0); // красный
glVertex3f(0.5, 1.0, 0.25); // Зад-верх
glColor3f(0.0, 1.0, 0.0); // зеленый
glVertex3f(0.5, 0, 0.25); // Зад-низ
glEnd();
// Верхняя грань
glBegin(GL_POLYGON);
glColor3f(0.0, 1.0, 0.0); // зеленый
glVertex3f(0, 1.0, 0.0); // Перед-лево
glColor3f(1.0, 0.0, 0.0); // красный
glVertex3f(0.5, 1.0, 0.0); // Перед-право
glColor3f(1.0, 0.0, 0.0); // красный
glVertex3f(0.5, 1.0, 0.25); // Зад-право
glColor3f(0.0, 1.0, 0.0); // зеленый
glVertex3f(0, 1.0, 0.25); // Зад-лево
glEnd();
// Нижняя грань
// Та же верхняя, только, Y = 0
glBegin(GL_POLYGON);
glColor3f(0.0, 1.0, 0.0); // зеленый
glVertex3f(0, 0.0, 0.0); // Перед-лево
glColor3f(1.0, 0.0, 0.0); // красный
glVertex3f(0.5, 0.0, 0.0); // Перед-право
glColor3f(1.0, 0.0, 0.0); // красный
glVertex3f(0.5, 0.0, 0.25); // Зад-право
glColor3f(0.0, 1.0, 0.0); // зеленый
glVertex3f(0, 0.0, 0.25); // Зад-лево
glEnd();

```

```

    glutSwapBuffers();
}
void GLKeyDown(unsigned char key, int x, int y)
{
    if (key == VK_ESCAPE) // Если нажата клавиша Escape
    {
        exit(0); // Выход
    }
    else // Если нажата любая другая клавиша
    {
        Angle += 5; // Увеличиваем угол поворота на 5 градусов
        glutPostRedisplay(); // Перерисовываем окно
    }
}
}

```

Запустите проект. Результат выполнения программы показан на рисунке 5.2.

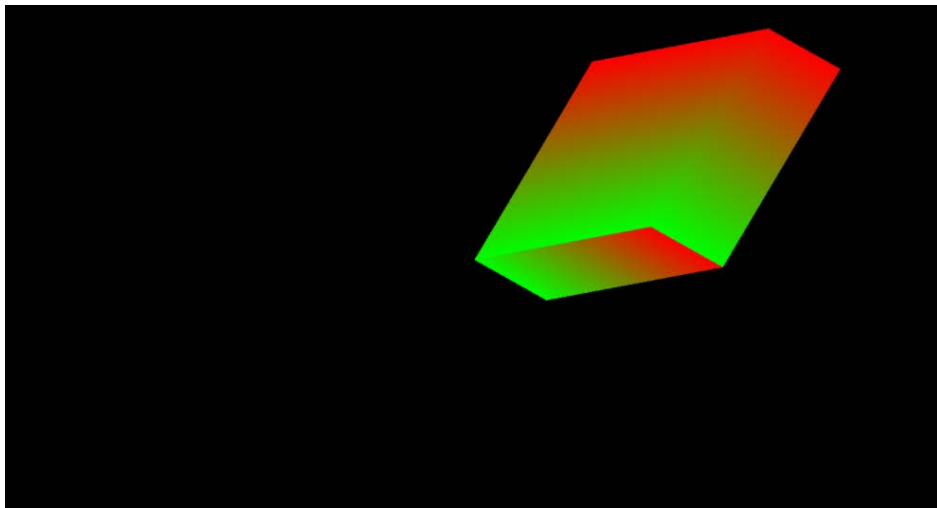


Рисунок 5.2 – Результат выполнения программы

Освойте текстурную заливку объектов. Сначала необходимо загрузить текстуры формата BMP. Для работы программы необходимо, помимо подключения библиотек, выполнить следующее:

- открыть свойства проекта;
- на вкладке «С/С++» выбрать вкладку «Перепроцессор», в строке «Определение перепроцессора» указать: `_CRT_SECURE_NO_WARNINGS` как это показано на рисунке 5.3.

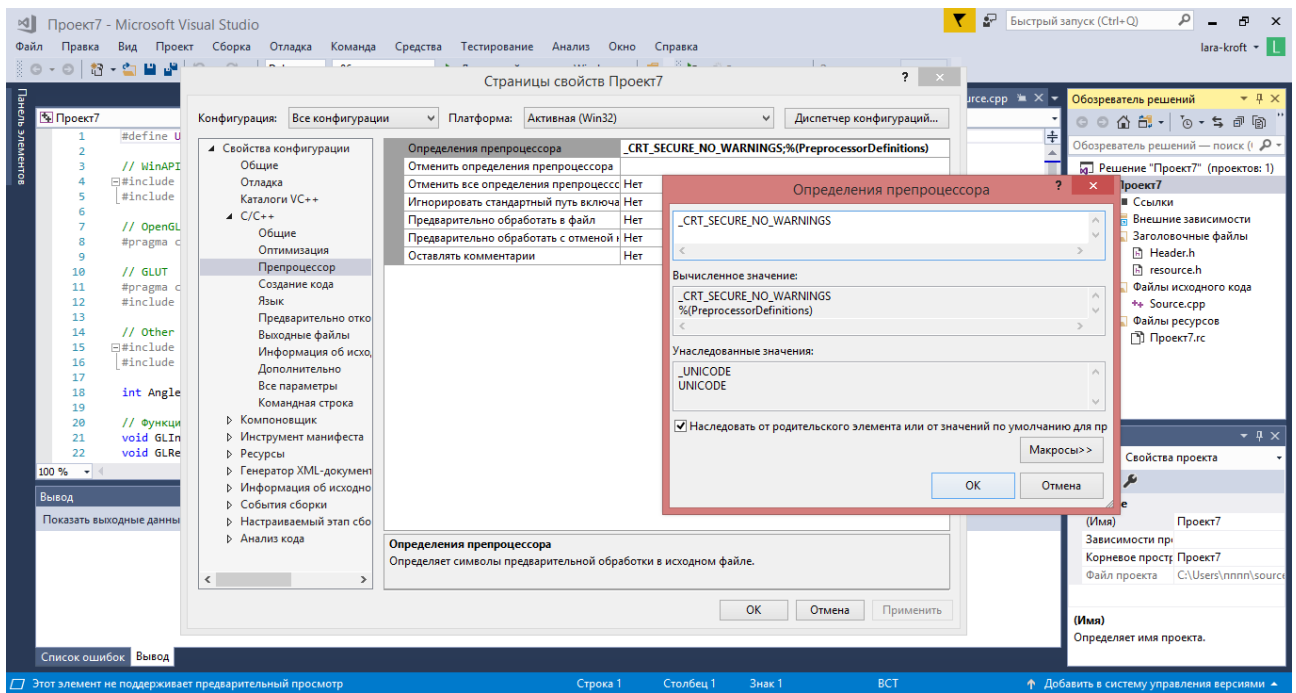


Рисунок 5.3 – Редактирование свойств

Переместите файл Texture.bmp в папку с проектом. Введите код программы:

```
#define UNICODE
// WinAPI
#include <Windows.h>
#include <tchar.h>
// OpenGL
#pragma comment(lib, "opengl32.lib")
// GLUT
#pragma comment(lib, "glut32.lib")
#include "glut.h"
// Other
#include <math.h>
#include <stdio.h>
int Angle = 0;
// Функции для работы с OpenGL/GLUT
void GLInit();
void GLRenderScene();
void GLKeyDown(unsigned char key, int x, int y);
// Прочие функции
GLuint InitTextureFromBMP(const char * imagepath);
int WINAPI wWinMain(HINSTANCE hInst, HINSTANCE hPrevInst, LPWSTR lpCmdLine,
int nShowCmd)
{
    // Аргументы командной строки (путь к EXE)
    char *argv0 = new char[512];
    GetModuleFileNameA(0, argv0, 512);
    int argc = 1;
```



```

// Инициализация GLUT
glutInit(&argc, &argv0);
glutInitDisplayMode(GLUT_DOUBLE | GLUT_SINGLE | GLUT_RGB);
// Координаты и размер окна GLUT
glutInitWindowPosition(200, 200);
glutInitWindowSize(400, 400);
// Создание окна
glutCreateWindow("");
// Обработчик рендеринга
glutDisplayFunc(GLRenderScene);
// Обработчик клавиатуры
glutKeyboardFunc(GLKeyDown);
// Инициализация OpenGL
GLInit();
// Главный цикл
glutMainLoop();
return 0;
}
void GLInit()
{
    // Цвет фона - черный
    glClearColor(0.0, 0.0, 0.0, 1.0);
    // Загрузка текстуры
    InitTextureFromBMP("Texture.bmp");
}
void GLRenderScene()
{
    // Очищаем буферы
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glLoadIdentity();
    // Включаем тест глубины,
    // чтобы грани 3D-фигур - рисовались непрозрачными
    glEnable(GL_DEPTH_TEST);
    // Поворот
    glRotatef(Angle, 1.0, 1.0, 0.0);
    // ----- Рисование параллелепипеда -----
    // Длина x высота x глубина:
    // 0.5  1.0  0.25
    ::glEnable(GL_TEXTURE_2D);
    // Передняя грань
    glBegin(GL_POLYGON);
    glTexCoord2f(0.0, 0.0);
    glVertex3f(0, 0, 0); // ЛН
    glTexCoord2f(0.0, 2.0);
    glVertex3f(0, 1.0, 0); // ЛВ
    glTexCoord2f(2.0, 2.0);
    glVertex3f(0.5, 1.0, 0); // ПВ
    glTexCoord2f(2.0, 0.0);
    glVertex3f(0.5, 0, 0); // ПН
    glEnd();
    ::glDisable(GL_TEXTURE_2D);
}

```

```

// Задняя грань
// Та же передняя, только, Z = 0.25, а, не 0
glColor3f(1.0, 0.0, 0.0); // красный
glBegin(GL_POLYGON);
glVertex3f(0, 0, 0.25); // ЛН
glVertex3f(0, 1.0, 0.25); // ЛВ
glVertex3f(0.5, 1.0, 0.25); // ПВ
glVertex3f(0.5, 0, 0.25); // ПН
glEnd();
// Левая грань
glColor3f(1.0, 1.0, 0.0); // желтый
glBegin(GL_POLYGON);
glVertex3f(0, 0, 0.0); // Перед-низ
glVertex3f(0, 1.0, 0.0); // Перед-верх
glVertex3f(0, 1.0, 0.25); // Зад-верх
glVertex3f(0, 0, 0.25); // Зад-низ
glEnd();
// Правая грань
// Та же левая, только, X = 0.5
glColor3f(0.0, 1.0, 1.0); // Голубой
glBegin(GL_POLYGON);
glVertex3f(0.5, 0, 0.0); // Перед-низ
glVertex3f(0.5, 1.0, 0.0); // Перед-верх
glVertex3f(0.5, 1.0, 0.25); // Зад-верх
glVertex3f(0.5, 0, 0.25); // Зад-низ
glEnd();
// Верхняя грань
glColor3f(1.0, 0.0, 1.0); // Розовый
glBegin(GL_POLYGON);
glVertex3f(0, 1.0, 0.0); // Перед-лево
glVertex3f(0.5, 1.0, 0.0); // Перед-право
glVertex3f(0.5, 1.0, 0.25); // Зад-право
glVertex3f(0, 1.0, 0.25); // Зад-лево
glEnd();
// Нижняя грань
// Та же верхняя, только, Y = 0
glColor3f(1.0, 1.0, 1.0); // Белый
glBegin(GL_POLYGON);
glVertex3f(0, 0.0, 0.0); // Перед-лево
glVertex3f(0.5, 0.0, 0.0); // Перед-право
glVertex3f(0.5, 0.0, 0.25); // Зад-право
glVertex3f(0, 0.0, 0.25); // Зад-лево
glEnd();
glutSwapBuffers();
}
void GLKeyDown(unsigned char key, int x, int y)
{
    Angle += 5; // Увеличиваем угол поворота на 5 градусов
    glutPostRedisplay(); // Перерисовываем окно
}
GLuint InitTextureFromBMP(const char* filename)
{

```

```

// Заголовок файла BMP
char header[54];
int dataPos; // Позиция в файле BMP, где заканчивается заголовок и начинаются
пиксели
int width, height;
int imageSize; // width * height * 3 (т.к., каждый пиксель занимает 3 байта - В (си-
ний), G (зеленый), R (красный))
char* data; // Сюда будут читаться пиксели из BMP
// Открываем файл
FILE * file = fopen(filename, "rb");
if (!file)
{
    MessageBox(0, L"Ошибка: Не удастся открыть файл с текстурой.", 0, 0);
    return 0;
}
if (fread(header, 1, 54, file) != 54)
{
    MessageBox(0, L"Ошибка: Файл с текстурой поврежден. Не удастся считать
заголовок файла.", 0, 0);
    return false;
}
if (header[0] != 'B' || header[1] != 'M') {
    MessageBox(0, L"Ошибка: Файл с текстурой поврежден. Имеет неверный за-
головок.", 0, 0);
    return 0;
}
dataPos = *(int*)&(header[0x0A]);
imageSize = *(int*)&(header[0x22]);
width = *(int*)&(header[0x12]);
height = *(int*)&(header[0x16]);
if (imageSize == 0)
    imageSize = width * height * 3;
if (dataPos == 0)
    dataPos = 54;
data = new char[imageSize];
fread(data, 1, imageSize, file);
fclose(file);
GLuint textureID;
glGenTextures(1, &textureID);
// ----- Привязка текстуры и настройка -----
glBindTexture(GL_TEXTURE_2D, textureID);
glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, width, height, 0,
    GL_BGR_EXT, // BGR означает, что в считанных пикселях следует B и R
(красный и синий) поменять местами (в формате BMP они перепутаны)
    GL_UNSIGNED_BYTE, data);
glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
// ----- Настройки масштабирования текстуры (под нужный размер) -----
glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
// При уменьшении - сглаживать

```

```
        glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR); //  
При увеличении - сглаживать  
        return textureID;  
    }
```

Запустите проект. Результат выполнения программы показан на рисунке 5.4.

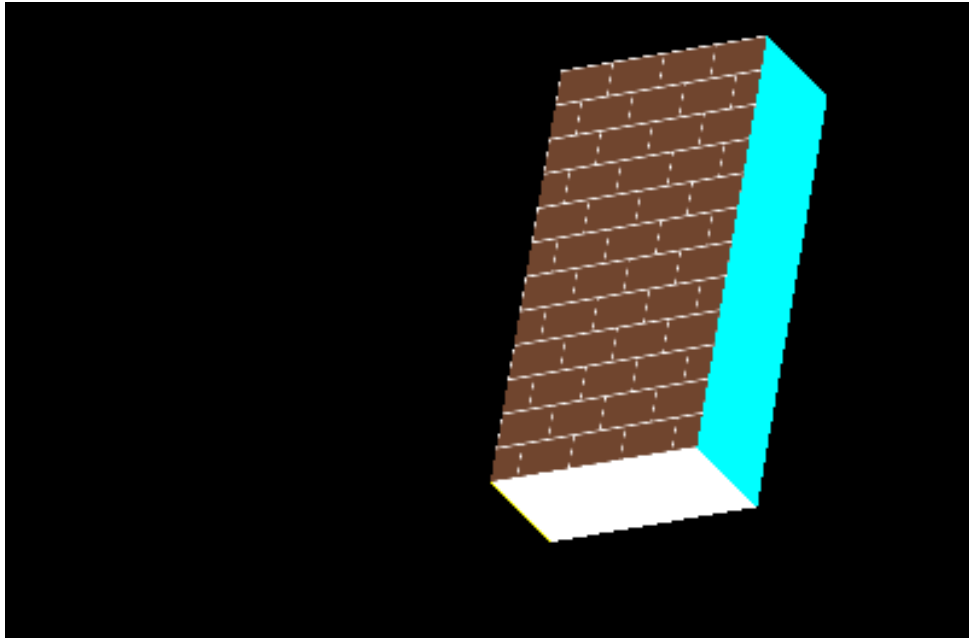


Рисунок 5.4 – Результат выполнения программы

Внесите в разработанный проект изменения в соответствии с индивидуальным заданием и проведите отладку приложения.

#### 3.5.4 Подготовка и защита отчета по лабораторной работе.

По результатам выполнения всех заданий подготовьте письменный отчет и представьте его для защиты лабораторной работы № 5. В ходе защиты отчета необходимо быть готовыми к демонстрации разработанных программных приложений и к ответам на контрольные вопросы преподавателя.

Примерный перечень контрольных вопросов:

- 1) Как подключить библиотеки OpenGL в Visual Studio?
- 2) Порядок создания приложения, рисующего параллелепипед с разноцветными гранями в Microsoft Visual Studio.

3) Порядок создания приложения, рисующего параллелепипед с гранями залитыми разными градиентами, вместо монотонной заливки в Microsoft Visual Studio.

4) Порядок создания приложения, рисующего параллелепипед с текстурной заливкой в Microsoft Visual Studio.

### **3.6 Лабораторная работа № 6. Разработка приложений, использующих графические библиотеки OpenGL в среде Microsoft Visual Studio (создание анимации)**

Цели работы: приобрести практические навыки по созданию различных приложений, использующих графические библиотеки OpenGL в среде Microsoft Visual Studio и закрепить знания по использованию программной среды при решении проектных задач.

Время: 2 часа.

Задание:

- подключить библиотеки OpenGL в Visual Studio;
- создать приложение, использующее OpenGL в Microsoft Visual Studio и осуществляющее анимацию секундной стрелки часов;
- внести изменения в разработанное приложение с учетом индивидуального задания;
- подготовить и защитить отчет по лабораторной работе.

Методические указания к выполнению задания.

3.6.1 Инструктаж и проверка готовности обучающихся к выполнению лабораторной работы № 6.

В начале занятия получите инструктаж преподавателя и допуск к работе. После чего получите индивидуальное задание на работу № 6.

3.6.2 Подключение библиотек OpenGL в Visual Studio.

По аналогии с предыдущими лабораторными работами запустите Visual Studio, создайте пустой проект и подключите библиотеки OpenGL.

### 3.6.3 Создание приложения, использующего OpenGL в Microsoft Visual Studio и осуществляющее анимацию секундной стрелки часов.

Для создания простой анимации достаточно использовать таймер, при тике которого будут изъясняться параметры изображения (например, координаты концов отрезка) и вызываться обработчик события Paint для рисования по новым параметрам. При таком подходе не надо заботиться об удалении старого изображения, ведь оно создается в окне заново.

Создайте новый проект, для этого в списке «Установленные шаблоны» выберите Visual C#. В списке шаблонов выберите значок «Приложение Windows Forms». Дайте подходящее название новой форме и нажмите кнопку ОК (рисунок 6.1).

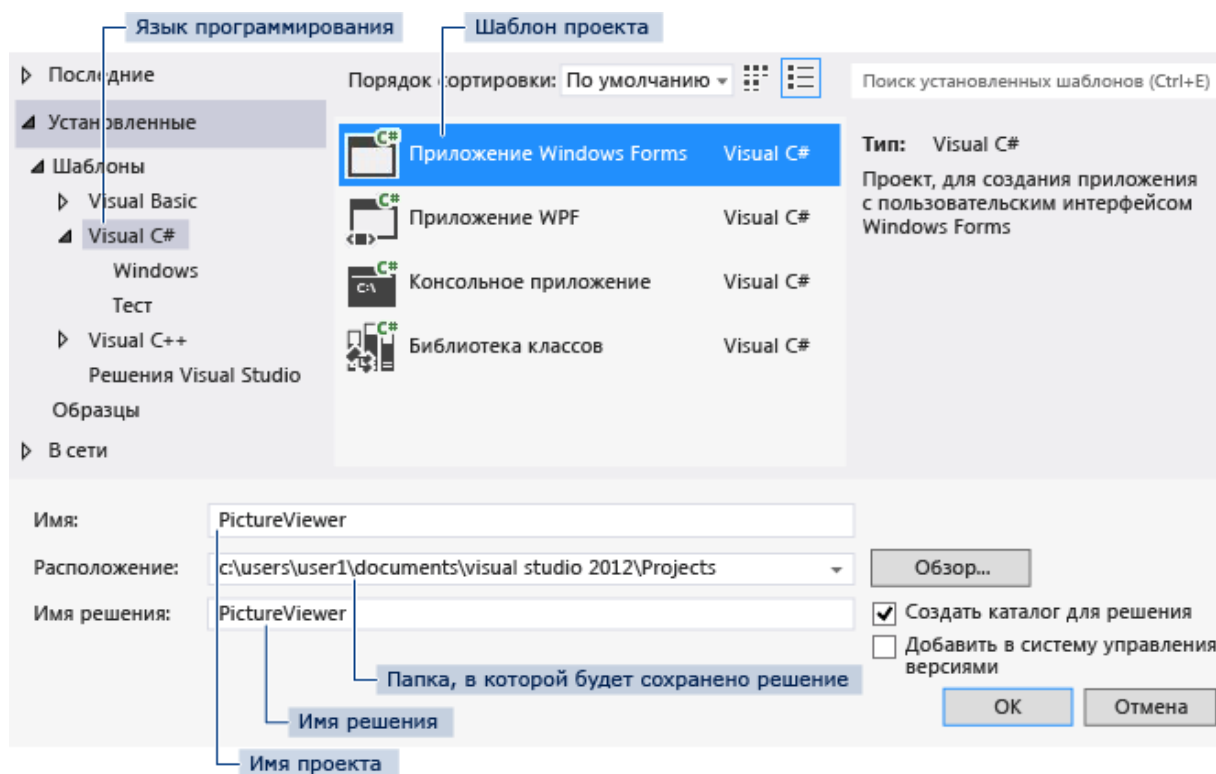


Рисунок 6.1 – Создание проекта

Обратите внимание на папку, в которой сохраняется проект.

Откроется конструктор форм. На форму следует поместить элемент Timer (при перетаскивании на форму он окажется в поле под ней). Далее установите зна-

чение свойства Interval = 1000 (1000 миллисекунд = 1 секунда), а также установите свойство Enabled в значение true (рисунок 6.2).

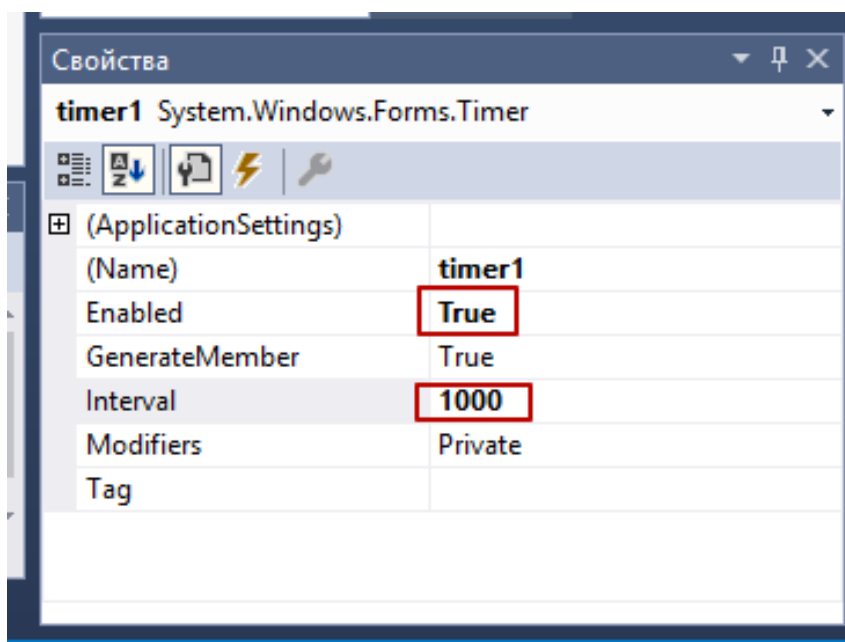


Рисунок 6.2 – Настройка элемента Timer

Далее следует объявить глобальные переменные, введя следующий код:

```
// Глобальные переменные
private int x1, y1, x2, y2, r;
private double a;
private Pen pen = new Pen(Color.DarkRed, 2);
```

Для события Paint разрабатываемой формы создайте обработчик, используя

код:

```
// Перерисовка формы
private void Form1_Paint(object sender, PaintEventArgs e)
{
    Graphics g = e.Graphics;
    // Рисуем секундную стрелку
    g.DrawLine(pen, x1, y1, x2, y2);
}
```

Для события Load также создайте обработчик:

```
// Действия при загрузке формы
private void Form1_Load(object sender, EventArgs e)
{
    r = 150; // радиус стрелки
```

```

a = 0; // угол поворота стрелки
// Определяем центр формы - начало стрелки
x1 = ClientSize.Width / 2;
y1 = ClientSize.Height / 2;
// Конец стрелки
x2 = x1 + (int)(r * Math.Cos(a));
y2 = y1 - (int)(r * Math.Sin(a));
}

```

Далее в списке событий элемента Timer дважды щелкните мышью по полю события Tick и перейдите к написанию кода обработчика этого события:

```

// Действия при очередном "тике" таймера
private void timer1_Tick(object sender, EventArgs e)
{
    a -= 0.1; // уменьшаем угол на 0,1 радиану
    // Новые координаты конца стрелки
    x2 = x1 + (int)(r * Math.Cos(a));
    y2 = y1 - (int)(r * Math.Sin(a));
    // Принудительный вызов события Paint
    Invalidate();
}

```

Результат работы программы – вращающаяся по часовой стрелке линия (рисунок 6.3).

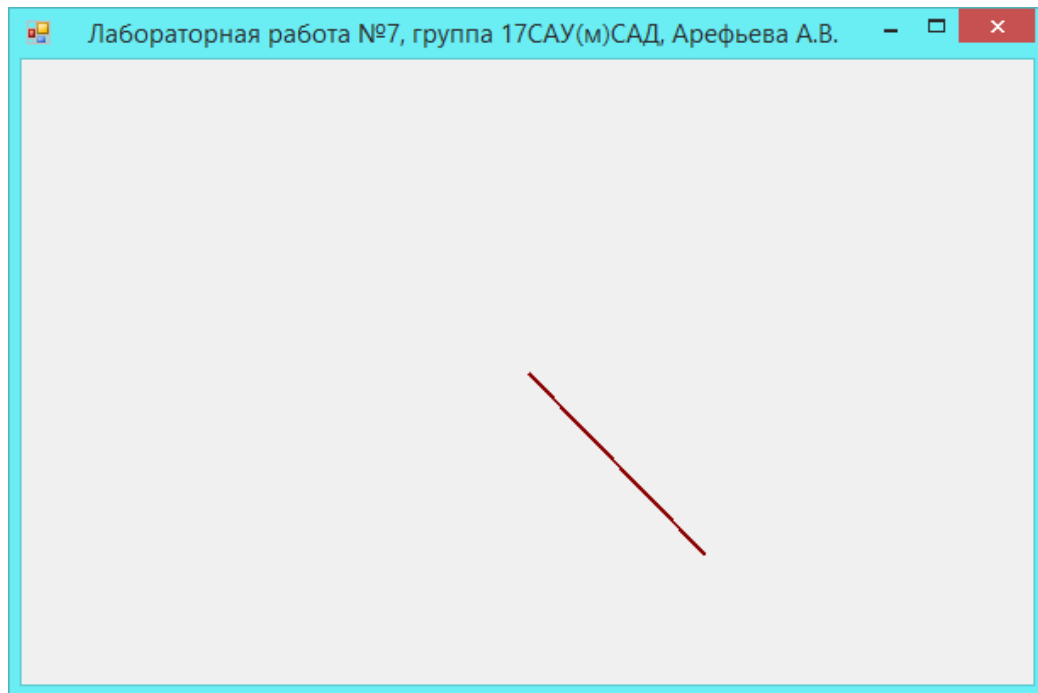


Рисунок 6.3 – Результат работы программы



Внесите в разработанный проект изменения в соответствии с индивидуальным заданием и проведите отладку приложения.

### 3.6.4 Подготовка и защита отчета по лабораторной работе.

По результатам выполнения всех заданий подготовьте письменный отчет и представьте его для защиты лабораторной работы № 6. В ходе защиты отчета необходимо быть готовыми к демонстрации разработанного программного приложения и к ответам на контрольные вопросы преподавателя.

Примерный перечень контрольных вопросов:

- 1) Порядок подключения библиотек OpenGL в Visual Studio.
- 2) Назовите условия для создания простой анимации в Microsoft Visual Studio.
- 3) Порядок создания приложения, реализующего анимацию секундной стрелки часов в Microsoft Visual Studio.

## **3.7 Лабораторная работа № 7. Разработка программного приложения в среде Visual Studio для выполнения инженерных расчётов**

Цели работы: приобрести практические навыки по созданию простейшей программы в среде Visual Studio для выполнения инженерных расчётов и закрепить знания по использованию программной среды при решении проектных задач.

Время: 2 часа.

Задание:

– составить программу вычисления арифметического выражения

$$u = \operatorname{tg}^2(x + y) - e^{y-z} \sqrt{\cos x^2 + \sin z^2} \quad (1)$$

для заданных значений  $x$ ,  $y$ ,  $z$  и организовать панель диалога программы в виде, представленном на рисунке 7.1 [9];

– внести изменения в разработанную программу с учетом индивидуального задания;

– подготовить и защитить отчет по лабораторной работе.

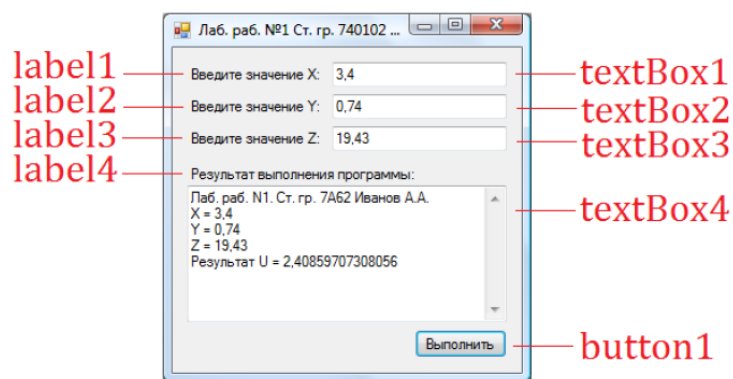


Рисунок 7.1 – Панель диалога программы

Методические указания к выполнению задания.

### 3.7.1 Инструктаж и проверка готовности обучающихся к выполнению лабораторной работы № 7.

В начале занятия получите инструктаж преподавателя и допуск к работе. После чего получите индивидуальное задание на работу № 7.

### 3.7.2 Создание программы вычисления арифметического выражения по формуле (1) для заданных значений x, y, z и организация требуемого вида панели диалога программы.

Запустите Visual Studio и создайте приложение Windows Forms (рисунок 7.2).

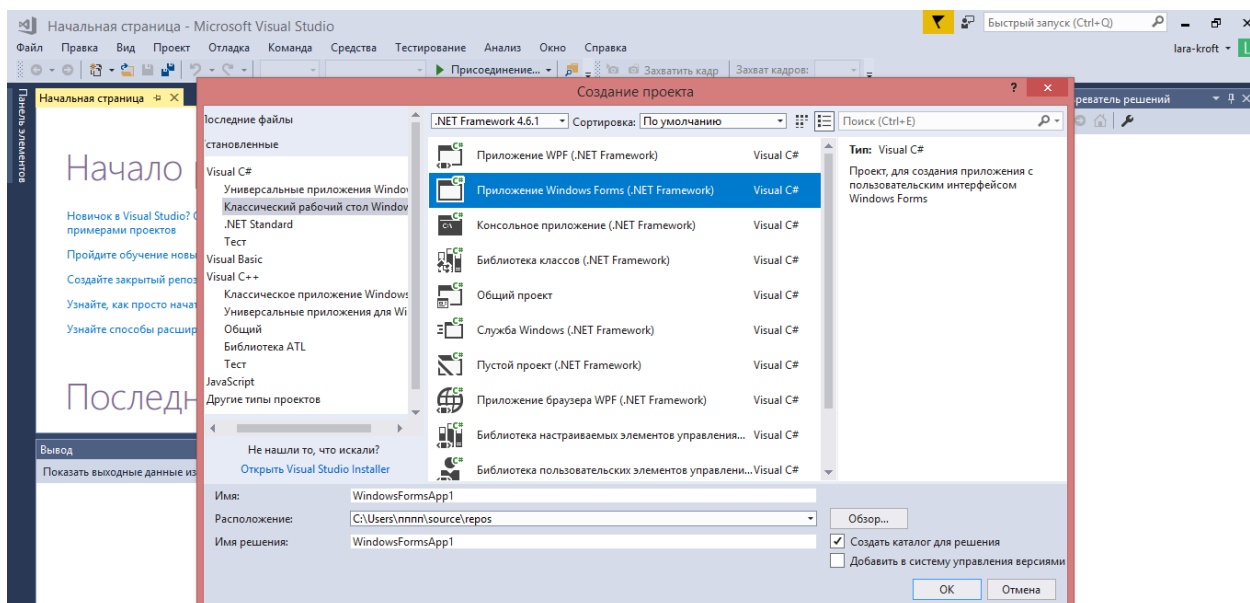


Рисунок 7.2 – Создание приложения Windows Forms

После загрузки проекта появится окно формы (рисунок 7.3).

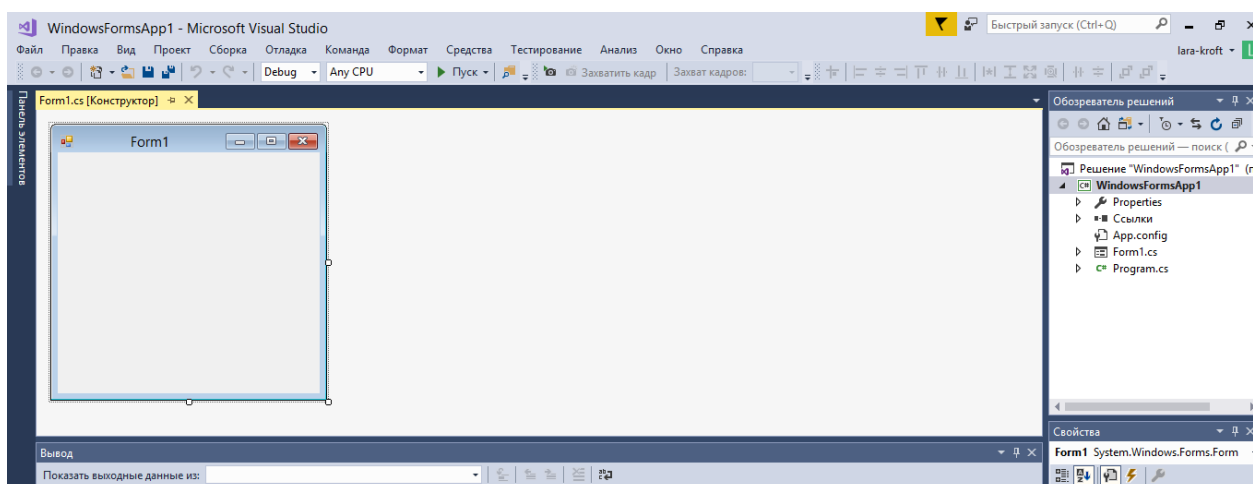


Рисунок 7.3 – Окно формы

Произведите настройку формы.

Для этого однократным щелчком мыши по кромке формы измените (растяните) ее размер. Измените свойства формы (окно «Свойства»), щелкнув левой кнопкой мыши на строке Text (если данное окно отсутствует, то необходимо щелкнуть правой кнопкой мыши по элементу «Форма», как это показано на рисунке 7.4).

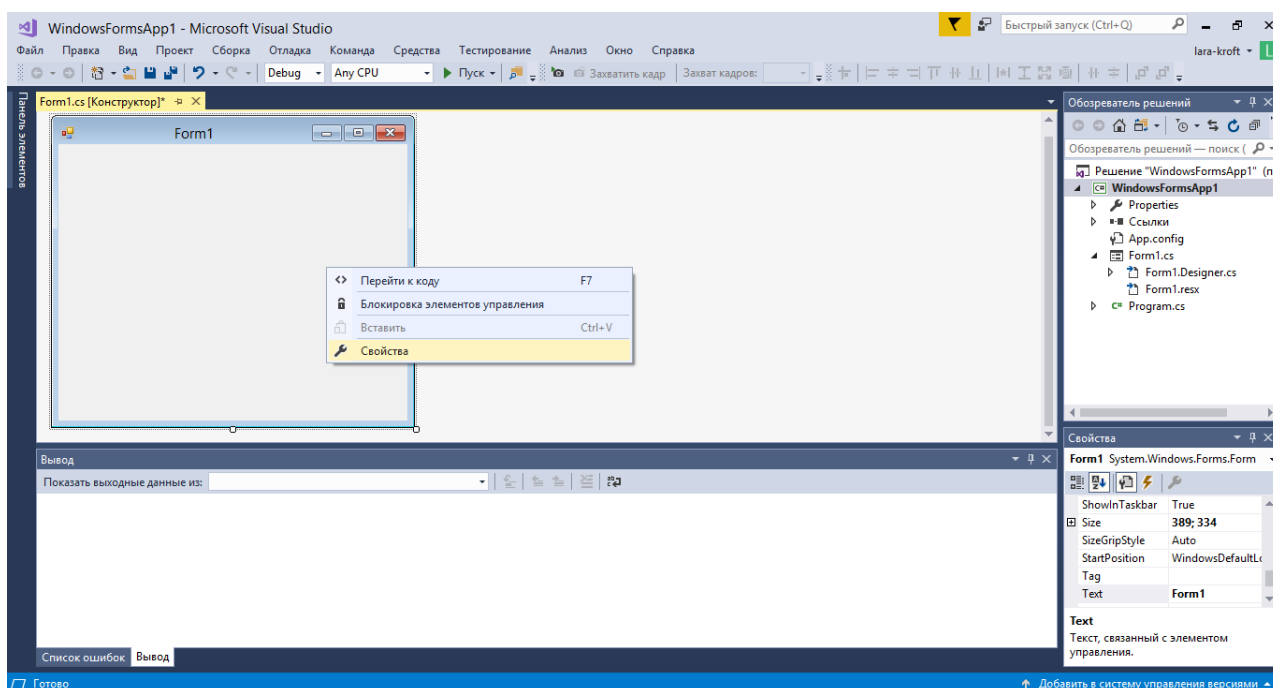


Рисунок 7.4 – Вызов окна свойств

В выделенном окне введите название, например, «ЛР-7. Вычисление выражения» (для изменения цвета окна можно использовать свойство BackColor).

Разместите на форме элементы управления, для этого используйте панель элементов (рисунок 7.5).

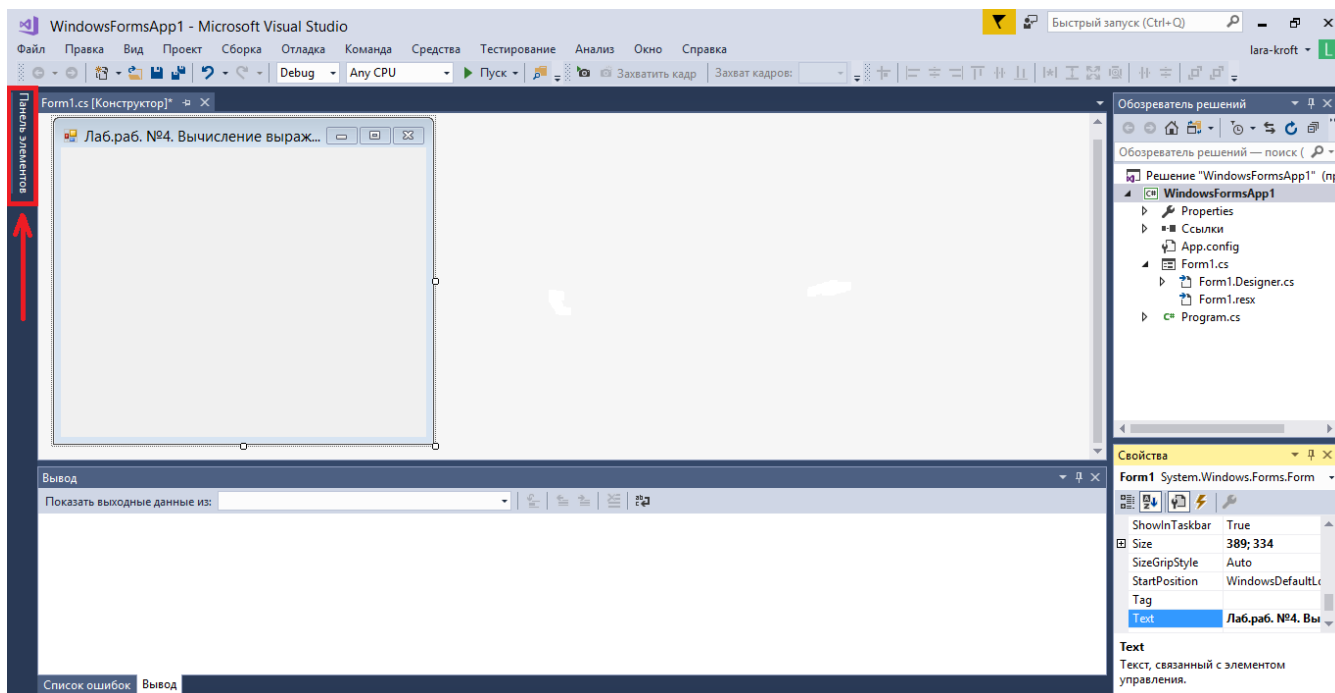


Рисунок 7.5 – Панель элементов

Разместите на форме строки для ввода значений переменных  $x$ ,  $y$ ,  $z$ , а также строку для вывода результата выражения –  $u$ .

Для этого используйте окно однострочного редактора текста, представляемого элементом управления `TextBox`. Выберите на панели элементов пиктограмму с названием `TextBox` (рисунок 7.6) и щелкните мышью в том месте формы, где ее необходимо поставить.

Захватив полученный элемент управления левой кнопкой мыши, можно отрегулировать его размеры и положение.

На рисунке 7.7 представлена полученная структура формы.

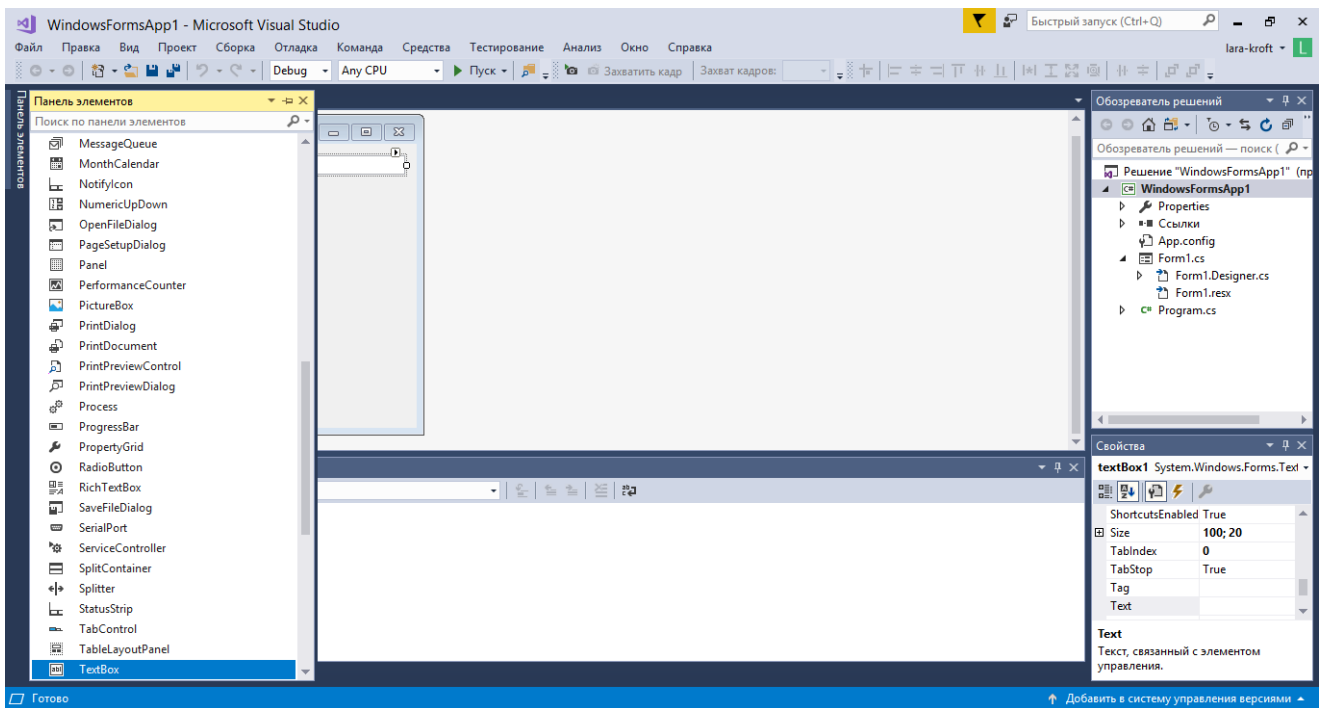


Рисунок 7.6 – Элемент управления TextBox

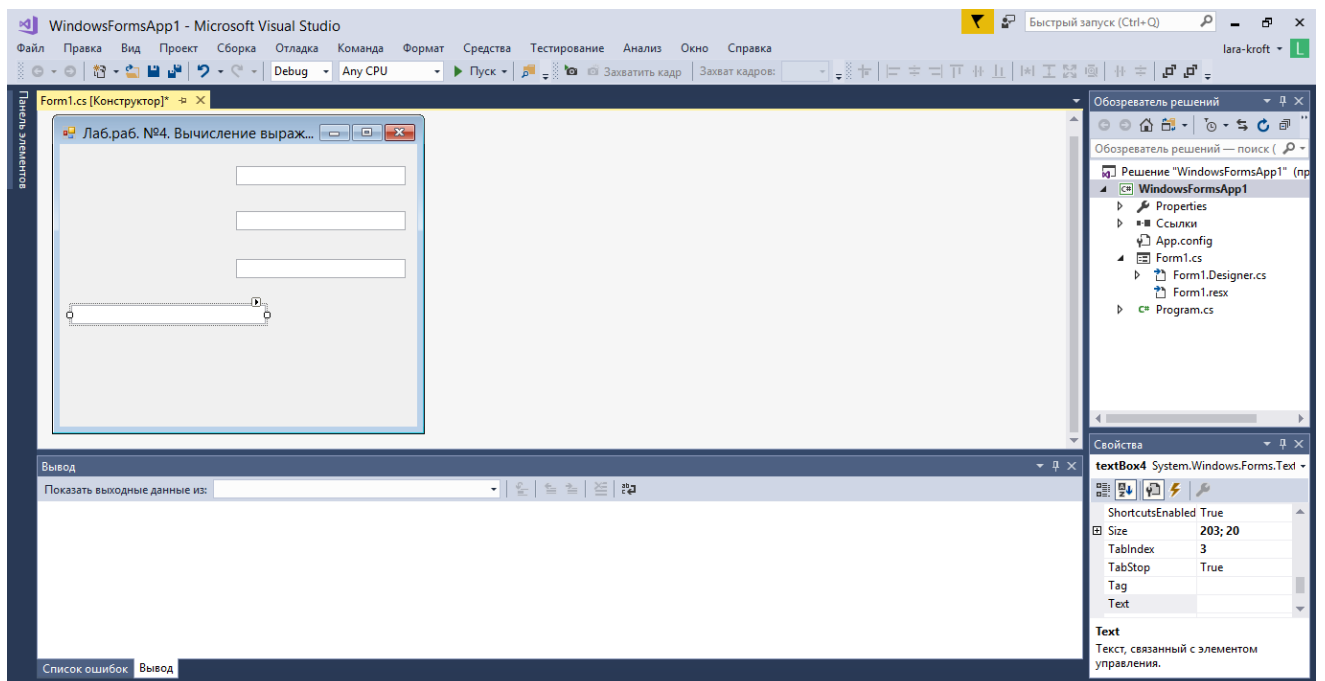


Рисунок 7.7 – Структура формы

В тексте программы теперь можно использовать переменные TextBox 1, TextBox 2, TextBox 3, TextBox 4. С помощью окна свойств установите шрифт и размер символов, отражаемых в строках TextBox (свойство Font).

Четвертое окно предназначено для вывода результата программы. Растяните это окно по вертикали, для этого щелкните мышкой на треугольник и установите свойство Multiline в True, как это показано на рисунке 7.8.

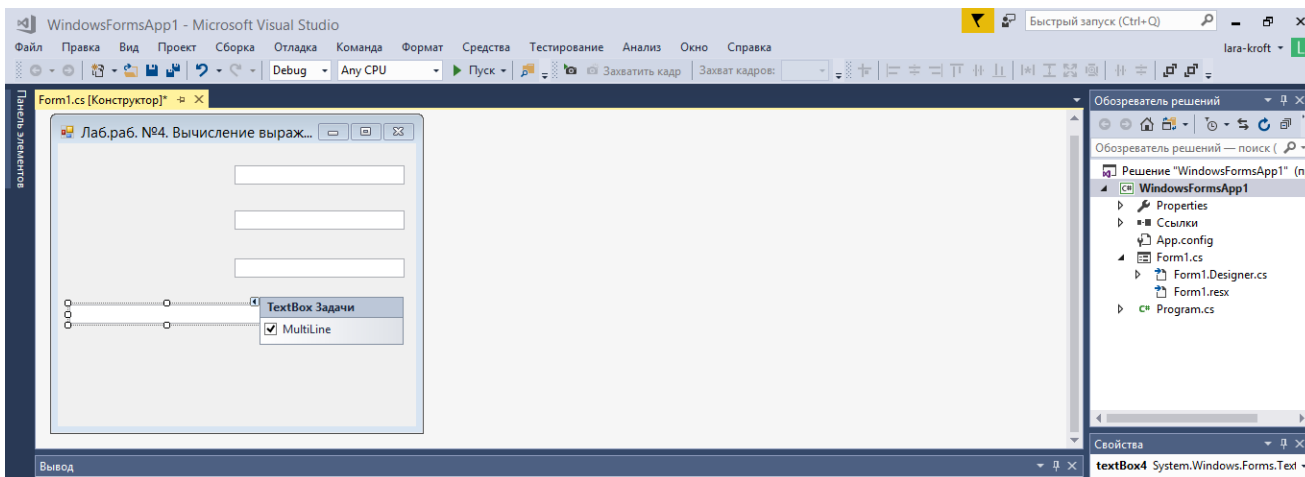


Рисунок 7.8 – Свойство Multiline

Для добавления в окно вертикальной и горизонтальной прокрутки установите свойство ScrollBars в значение Both, как это показано на рисунке 7.9.

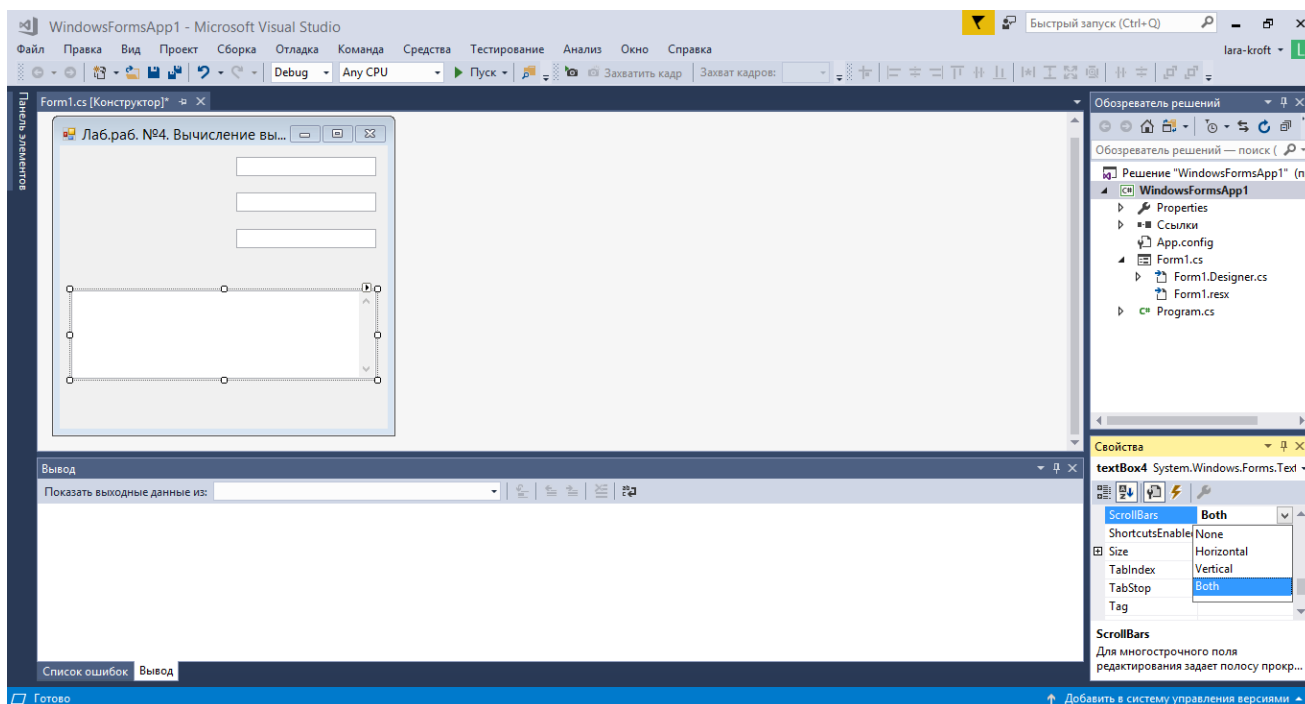


Рисунок 7.9 – Добавление в окно вертикальной и горизонтальной прокрутки

Разместите на форме надписи. Для этого используется элементы управления Label. Щелкнув на них, отрегулируйте размер и настройте свойство Text в окне свойств. Также выберите шрифт и размер символов (свойство Font). На рисунке 7.10 представлена вновь полученная структура.

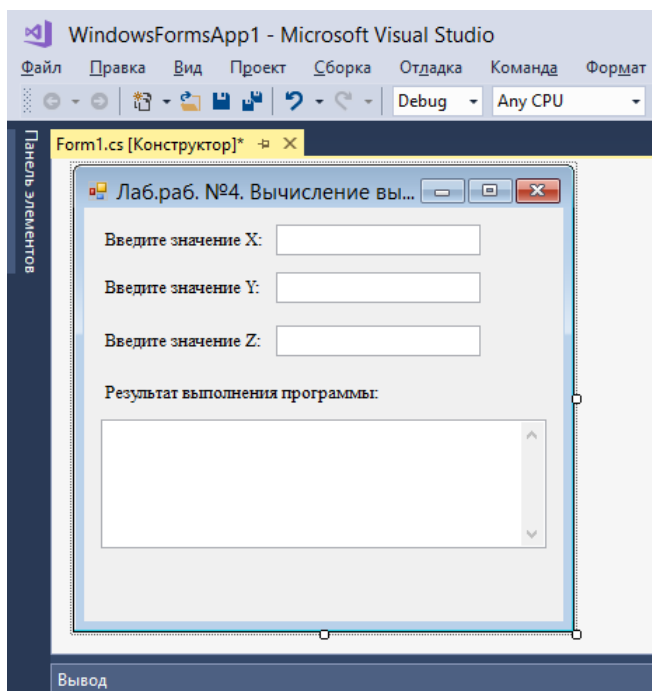


Рисунок 7.10 – Размещение на форме надписей

Разместите на форме кнопку. Для этого используется элемент управления Button. Щелкнув на нем, отрегулируйте размер и настройте свойство Text в окне свойств. Также выберите шрифт и размер символов (свойство Font). На рисунке 7.11 представлена панель диалога программы, которая была задана.

Чтобы программа вычисления выражения (1) работала, необходимо создать обработчики событий Load у формы и Click у кнопки, а затем в них вставить код программы.

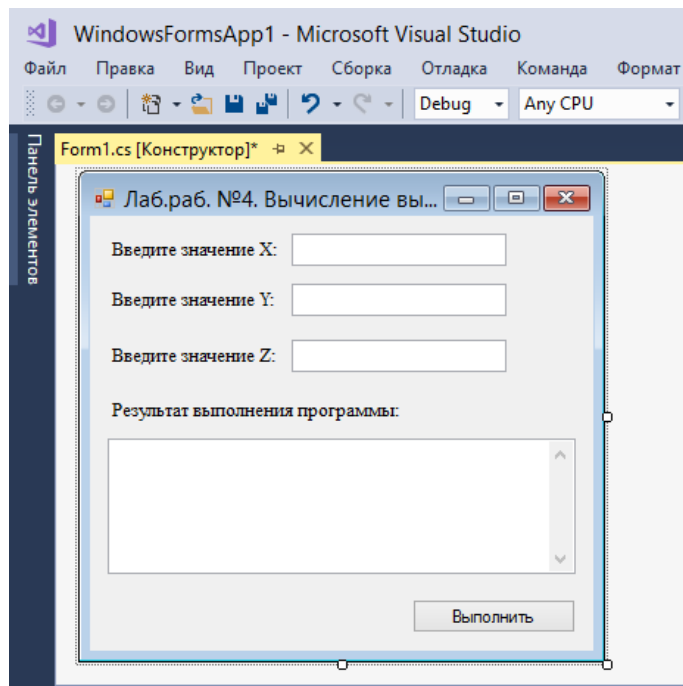


Рисунок 7.11 – Панель диалога программы

Для этого дважды щелкните мышкой на форму, появится текст программы – это и есть обработчик событий Load у формы (рисунок 7.12).

```
19  
20 private void Form1_Load(object sender, EventArgs e)  
21 {  
22 }  
23
```

Рисунок 7.12 – Обработчик событий Load

Между фигурными скобками необходимо вставить код программы.

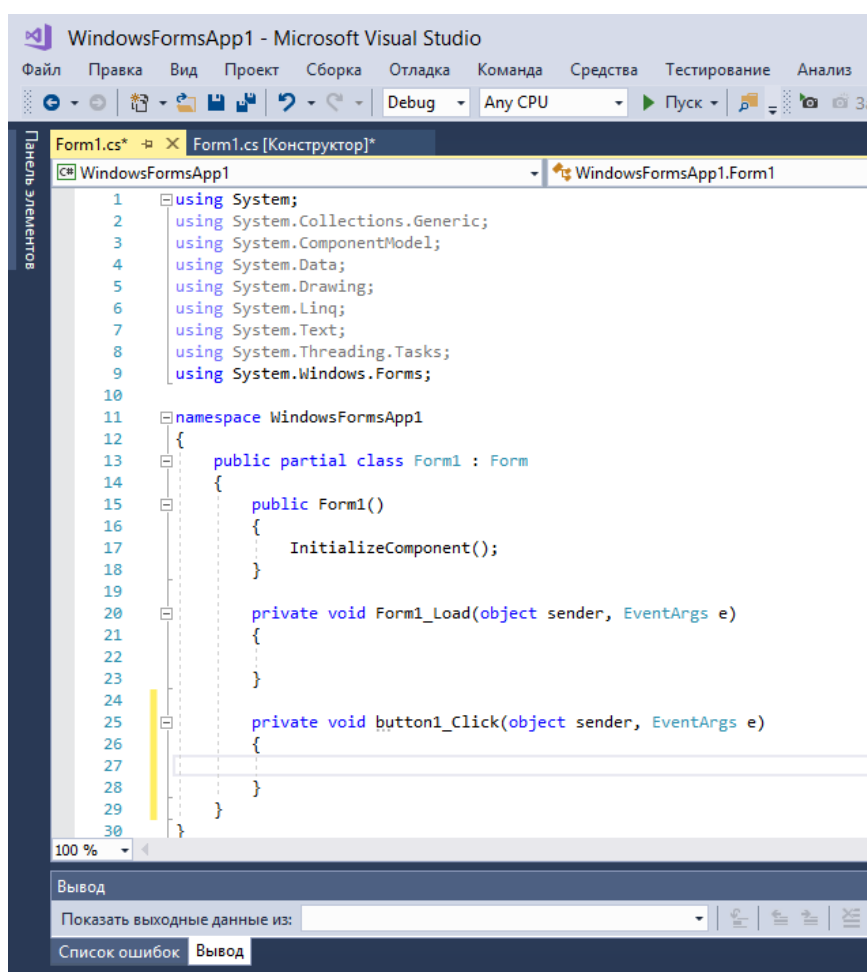
Возвратесь в конструктор и создайте обработчик событий Click у кнопки двойным щелчком мыши по кнопке. Появится текст программы, представленный на рисунке 7.13.

```
private void button1_Click(object sender, EventArgs e)  
{  
}  
}
```

Рисунок 7.13 – Обработчик событий Click



Между фигурными скобками также необходимо вставить код программы.  
На рисунке 7.14 представлен фрагмент текста программы.



```
1 using System;
2 using System.Collections.Generic;
3 using System.ComponentModel;
4 using System.Data;
5 using System.Drawing;
6 using System.Linq;
7 using System.Text;
8 using System.Threading.Tasks;
9 using System.Windows.Forms;
10
11 namespace WindowsFormsApp1
12 {
13     public partial class Form1 : Form
14     {
15         public Form1()
16         {
17             InitializeComponent();
18         }
19
20         private void Form1_Load(object sender, EventArgs e)
21         {
22
23         }
24
25         private void button1_Click(object sender, EventArgs e)
26         {
27
28         }
29     }
30 }
```

Рисунок 7.14 – Фрагмент текста программы

Допишите текст программы. На рисунке 7.15 представлен полный текст программы.

```

1  g System;
2  g System.Collections.Generic;
3  g System.ComponentModel;
4  g System.Data;
5  g System.Drawing;
6  g System.Linq;
7  g System.Text;
8  g System.Threading.Tasks;
9  g System.Windows.Forms;
10
11 space WindowsFormsApp1
12
13 public partial class Form1 : Form
14 {
15     public Form1()
16     {
17         InitializeComponent();
18     }
19
20     private void Form1_Load(object sender, EventArgs e)
21     {
22         // Начальное значение X
23         textBox1.Text = "3,4";
24         // Начальное значение Y
25         textBox2.Text = "0,74";
26         // Начальное значение Z
27         textBox3.Text = "19,43";
28     }
29
30     private void button1_Click(object sender, EventArgs e)
31     {
32         // Считывание значения X
33         double x = double.Parse(textBox1.Text);
34         // Вывод значения X в окно
35         textBox4.Text += Environment.NewLine + "X=" + x.ToString();
36         // Считывание значения Y
37         double y = double.Parse(textBox2.Text);
38         // Вывод значения Y в окно
39         textBox4.Text += Environment.NewLine + "Y=" + y.ToString();
40         // Считывание значения Z
41         double z = double.Parse(textBox3.Text);
42         // Вывод значения Z в окно
43         textBox4.Text += Environment.NewLine + "Z=" + z.ToString();
44         // Вычисляем арифметическое выражение
45         double a = Math.Tan(x + y) * Math.Tan(x + y);
46         double b = Math.Exp(y - z);
47         double c = Math.Sqrt(Math.Cos(x * x) + Math.Sin(z * z));
48         double u = a - b * c;
49         // Выводим результат в окно
50         textBox4.Text += Environment.NewLine + "Результат U =" + u.ToString();
51     }
52 }
53

```

Рисунок 7.15 – Полный текст программы

Запустите проект. В появившемся окне введите любые значения переменных  $x$ ,  $y$ ,  $z$  и нажмите кнопку «Выполнить», в результате чего будет выведено значение выражения, как это показано на рисунке 7.16.

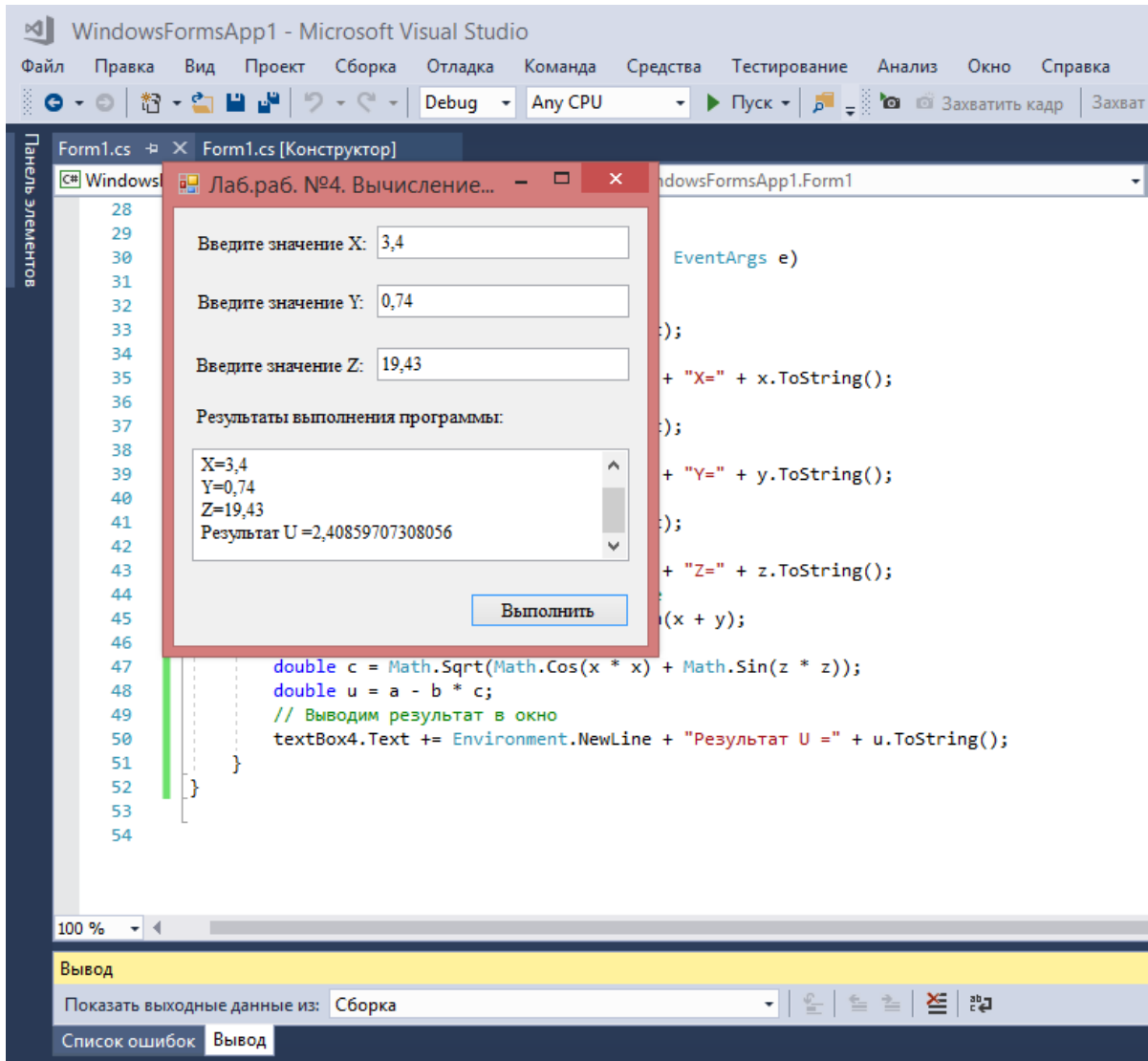


Рисунок 7.16 – Результат выполнения программы

В случае если после запуска отладки, программа выдает ошибки, то необходимо ввести фрагмент кода, приведенный на рисунке 7.17.

```
53
54     internal class textBox4
55     {
56         internal static string Text;
57     }
58
59     internal class textBox3
60     {
61         internal static string Text;
62     }
63
64     internal class textBox2
65     {
66         internal static string Text;
67     }
68
69     internal class textBox1
70     {
71         internal static string Text;
72     }
73 }
74
```

Рисунок 7.17 – Фрагмент программы

Внесите в разработанную программу изменения в соответствии с индивидуальным заданием и проведите отладку приложения.

### 3.7.3 Подготовка и защита отчета по лабораторной работе.

По результатам выполнения всех заданий подготовьте письменный отчет и представьте его для защиты лабораторной работы № 7. В ходе защиты отчета необходимо быть готовыми к демонстрации разработанного программного приложения и к ответам на контрольные вопросы преподавателя.

Примерный перечень контрольных вопросов:

- 1) Порядок настройки формы Windows Forms.
- 2) Как добавить в окно формы вертикальную и горизонтальную прокрутку?
- 3) Как разместить на форме надписи?
- 4) Что необходимо, чтобы программа вычисляла выражения?

### **3.8 Лабораторная работа № 8. Разработка программного приложения в среде Visual Studio для выполнения инженерных расчётов и их представления в виде графиков**

Цели работы: приобрести практические навыки по созданию программы в среде Visual Studio для выполнения инженерных расчётов, вывода результата в виде графиков и закрепить знания по использованию программной среды при решении проектных задач.

Время: 2 часа.

Задание:

– составить программу, отображающую графики функций  $\sin(x)$  и  $\cos(x)$  на интервале  $[x_{\min}, x_{\max}]$  с возможностью изменения разметки координатных осей, а также шага построения таблицы [9];

– внести изменения в разработанную программу с учетом индивидуального задания;

– подготовить и защитить отчет по лабораторной работе.

Методические указания к выполнению задания.

3.8.1 Инструктаж и проверка готовности обучающихся к выполнению лабораторной работы № 8.

В начале занятия получите инструктаж преподавателя и допуск к работе. После чего получите индивидуальное задание на работу № 8.

3.8.2 Создание программы, отображающей графики функций  $\sin(x)$  и  $\cos(x)$  на интервале  $[x_{\min}, x_{\max}]$  с возможностью изменения разметки координатных осей, а также шага построения таблицы.

Создайте новый проект, для этого в списке «Установленные шаблоны» выберите Visual C#. В списке шаблонов выберите значок «Приложение Windows Forms». Дайте подходящее имя новой форме и нажмите кнопку ОК (рисунок 8.2). Обратите внимание на папку, в которой сохраняется проект.

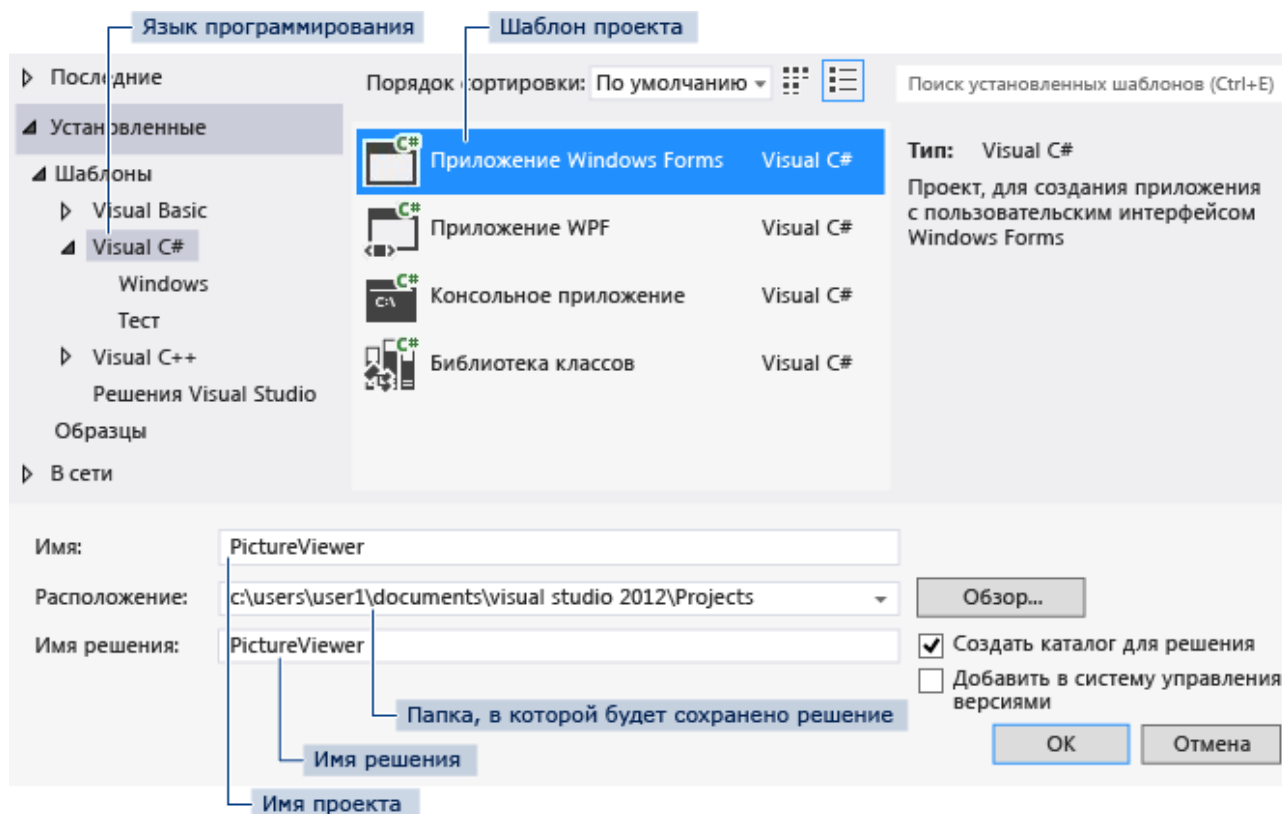


Рисунок 8.2 – Создание проекта

Откроется конструктор форм. Прежде всего, следует поместить на форму все необходимые элементы: Chart (раздел «Данные»); три элемента Textbox; три элемента Label, Button.

В свойствах элементов Label и Button, а также формы измените параметр Text согласно рисунку 8.3.

Далее следует настроить элемент управления Chart. Для этого в свойствах элемента выберите пункт Series. В открывшемся окне измените два параметра – Name (имя линии на графике) и ChartType (тип диаграммы). Для Name задайте имя графика, который необходимо выводить, в ChartType выберите тип линии Spline.

Затем нажмите кнопку «Добавить», которая добавит на график еще одну диаграмму (для второй функции). Для нее настройте параметры аналогично первой. Дополнительно можно изменить такие свойства, как BorderWidth (определяет толщину линии на графике) и Color (меняет цвет линии на графике). После этого окно можно закрыть.

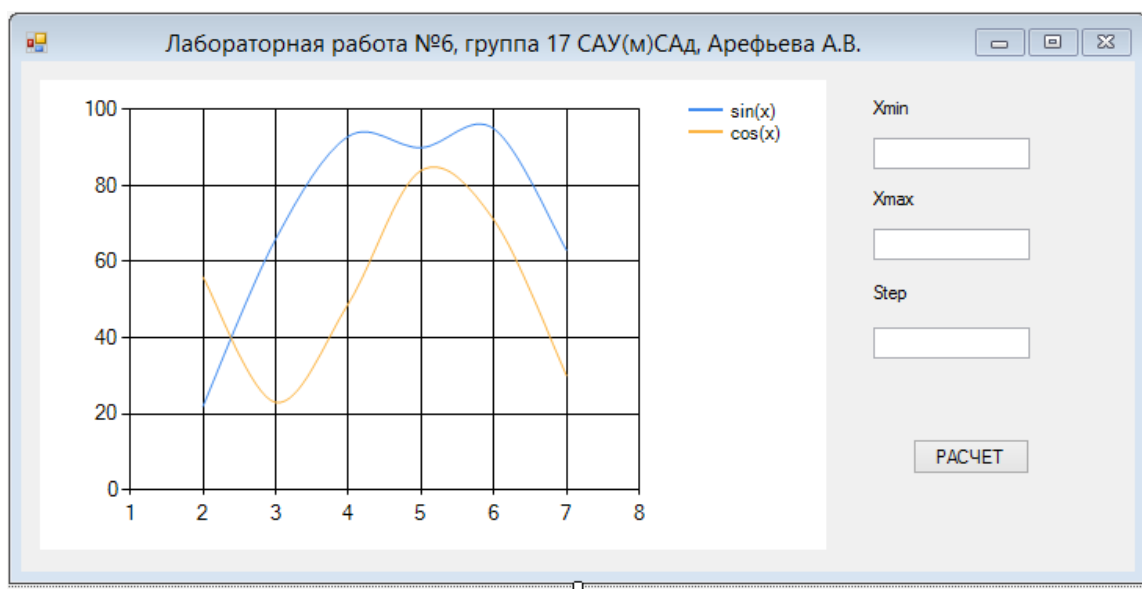


Рисунок 8.3 – Вариант расположения элементов в конструкторе форм

Далее двойным кликом по элементу Button создается обработчик события нажатия этой кнопки, который выполняет все требуемые настройки и расчеты, а также отображает графики функций.

Код обработчика событий:

```
private void button1_Click(object sender, EventArgs e)
{
    // Считываем с формы требуемые значения
    double xmin = double.Parse(textBox1.Text);
    double xmax = double.Parse(textBox2.Text);
    double step = double.Parse(textBox3.Text);
    // Количество точек графика
    int count = (int)Math.Ceiling((xmax - xmin) / step) + 1;
    // Массив значений X - общий для обоих графиков
    double[] x = new double[count];
    // Два массива Y - по одному для каждого графика
    double[] y1 = new double[count];
    double[] y2 = new double[count];
    // Рассчитываем точки для графиков функций
    for (int i = 0; i < count; i++)
    {
        // Вычисляем значение X
        x[i] = xmin + step * i;
        // Вычисляем значение функций в точке X
        y1[i] = Math.Sin(x[i]);
        y2[i] = Math.Cos(x[i]);
    }
    // Настраиваем оси графика
    chart1.ChartAreas[0].AxisX.Minimum = xmin;
```

```

chart1.ChartAreas[0].AxisX.Maximum = xmax;
// Определяем шаг сетки
chart1.ChartAreas[0].AxisX.MajorGrid.Interval = step;
// Добавляем вычисленные значения в графики
chart1.Series[0].Points.DataBindXY(x, y1);
chart1.Series[1].Points.DataBindXY(x, y2);
}

```

Результат работы программы представлен на рисунке 8.4.

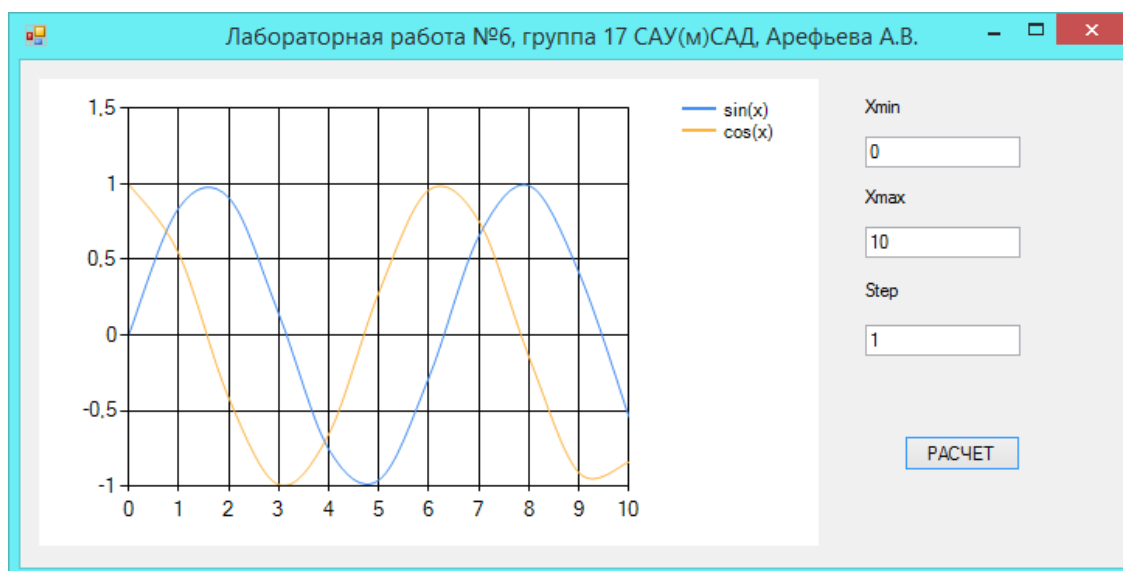


Рисунок 8.4 – Результат работы программы

Внесите в разработанную программу изменения в соответствии с индивидуальным заданием и проведите отладку приложения.

### 3.8.3 Подготовка и защита отчета по лабораторной работе.

По результатам выполнения всех заданий подготовьте письменный отчет и представьте его для защиты лабораторной работы № 8. В ходе защиты отчета необходимо быть готовыми к демонстрации разработанного программного приложения и к ответам на контрольные вопросы преподавателя.

Примерный перечень контрольных вопросов:

- 1) Порядок создания и настройки формы Windows Forms.
- 2) Как настроить элемент управления Chart?
- 3) Как добавить на график еще одну диаграмму (для второй функции)?
- 4) Как можно изменить такие свойства, как BorderWidth и Color?



## Список использованных источников

- 1 Норенков, И.П. Основы автоматизированного проектирования: учеб. для вузов / И.П. Норенков; – М.: Изд-во МГТУ им. Н.Э. Баумана, 2-е изд., перераб. и доп., 2002. – 336 с.
- 2 Малюх, В.Н. Введение в современные САПР: курс лекций / В.Н. Малюх; – М.: ДМК Пресс, 2010. – 192 с. – ISBN 978-5-94074-551-8.
- 3 Ушаков, Д.М. Введение в математические основы САПР: курс лекций / Д.М. Ушаков; – М.: ДМК Пресс, 2011. – 208 с.
- 4 Стандарт организации. СТО 02069024. 101 – 2015. Работы студенческие. Общие требования и правила оформления [Электронный ресурс]. – Режим доступа: [http://www.osu.ru/docs/official/standart/standart\\_101-2015\\_.pdf](http://www.osu.ru/docs/official/standart/standart_101-2015_.pdf). – 85 с.
- 5 Ахмедьянова, Г.Ф. Программно-методические комплексы систем автоматизированного проектирования [Электронный ресурс] : электронный курс в системе Moodle / Г.Ф. Ахмедьянова, В.А. Трипкош; М-во науки и высш. образования Рос. Федерации, Федер. гос. бюджет. образоват. учреждение высш. образования "Оренбург. гос. ун-т". – Оренбург : ОГУ. – 2019. – 8 с. – Режим доступа: <https://moodle.osu.ru/course/view.php?id=2005>.
- 6 <https://visualstudio.microsoft.com/ru/> – официальный сайт Microsoft Visual Studio (дата обращения: 8.09.2019 г.).
- 7 <https://www.khronos.org/opengl> – сайт «Обзор OpenGL» (дата обращения: 8.09.2019 г.).
- 8 <https://prog-cpp.ru/vs-console> – сайт «Создание консольного приложения Visual Studio» (дата обращения: 8.09.2019 г.).
- 9 Демин, А.Ю. Лабораторный практикум по информатике: учебное пособие / А.Ю. Демин, В.А. Дорофеев; Томский политехнический университет. – Томск: Изд-во Томского политехнического университета, 2014. – 132 с.