Министерство науки и высшего образования Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«Санкт-Петербургский государственный университет промышленных технологий и дизайна»
Высшая школа технологии и энергетики
Кафедра прикладной математики и информатики

ВИЗУАЛЬНЫЕ СРЕДЫ ПРОГРАММИРОВАНИЯ

Выполнение курсовой работы

Методические указания для студентов всех форм обучения по направлению подготовки
01.03.02 — Прикладная математика и информатика

Составитель
А. Н. Маслобоев

Санкт-Петербург
2024

Утверждено
на заседании кафедры ПМ и И
22.12.2021 г., протокол № 5

Рецензент В. И. Сидельников

Методические указания соответствуют программам и учебным планам дисциплины «Визуальные среды программирования» для студентов, обучающихся по направлению подготовки: 01.03.02 «Прикладная математика и информатика». В указаниях представлен порядок выполнения и оформления курсовой работы. В приложении приведен рекомендуемый список тем для курсовых работ.

Методические указания предназначены для бакалавров очной формы обучения.

Утверждено Редакционно-издательским советом ВШТЭ СПбГУПТД в качестве методических указаний

Редактор и корректор Д. А. Романова
Техн. редактор Д. А. Романова Темплан 2020 г., поз. 101

Подписано к печати 14.03.2024. Формат 60x84/16. Бумага тип № 1.
Печать офсетная. Печ.л. 2,8. Уч.-изд. л. 2,8.
Тираж 30 экз. Изд. № 101. Цена «С». Заказ №

Ризограф Высшей школы технологии и энергетики СПбГУПТД,

198095, Санкт-Петербург, ул. Ивана Черных, 4.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Цель дисциплины «Визуальные среды программирования» заключается в формировании компетенции обучающегося в области современных технологий визуального программирования и обучении студента применению полученных знаний и навыков в профессиональной деятельности. В практическом аспекте данной дисциплины студент должен:

- изучить основные принципы работы в интегрированной среде разработки программного обеспечения Microsoft Visual Studio;
- освоить основные приемы быстрой разработки приложений для ОС Windows при помощи программных средств визуального программирования;
- получить практические навыки визуальной разработки приложений в средах Visual C++ и Visual C#.

Методические указания разработаны в соответствии с программой курса «Визуальные среды программирования» Федерального государственного образовательного стандарта для бакалавров по направлению подготовки 01.03.02 «Прикладная математика и информатика».

Методические указания состоят из четырех разделов, введения, заключения, списка литературы и двух приложений.

1. ЦЕЛЬ И ОСНОВНЫЕ ЭТАПЫ СОЗДАНИЯ КУРСОВОЙ РАБОТЫ

Целью курсовой работы является приобретение базовых навыков по разработке приложения с графическим пользовательским интерфейсом, работающего на платформе операционной системы Microsoft Windows. Курсовая работа состоит из следующих этапов:

- выбор темы для разработки приложения;
- подбор необходимой для выполнения работы литературы и разработка плана курсовой работы;
- утверждение плана курсовой работы руководителем;
- разработка графического интерфейса приложения и написание его программного кода
- тестирование и отладка приложения при различных наборах исходных данных;
- создание пояснительной записки к курсовой работе;
- доработка курсовой работы с учетом замечаний и предложений руководителя;
- сдача курсовой работы руководителю для оформления ее допуска к защите;
- защита курсовой работы.

Выбор студентом темы для разработки приложения производится на основе списка рекомендуемых тем, который приводится далее в настоящей работе. Кроме того, допускается самостоятельный выбор темы студентом при условии согласования ее с руководителем курсового проекта (преподавателем) и утверждения данной темы заведующим кафедрой.

Ниже приводится рекомендуемый список тем для выполнения курсовой работы:

1. Разработка простого текстового редактора.
2. Разработка программы автоматического рерайта русского текста.
3. Разработка инженерного калькулятора.
4. Разработка калькулятора для программиста.
5. Разработка финансового калькулятора.
6. Разработка матричного калькулятора.
7. Разработка калькулятора для решения системы линейных уравнений различными методами.
8. Разработка калькулятора расчета потребления электроэнергии.
9. Разработка статистического калькулятора.
10. Разработка приложения, выполняющего аппроксимацию функции
    а) методом наименьших квадратов;
    б) методом выбранных точек;
    в) методом средних.
11. Разработка приложения, выполняющего интерполяцию функции
    а) по формуле Лагранжа;
    б) по формуле Ньютона;

в) по формуле Бесселя.

12. Разработка приложения, выполняющего решение системы линейных уравнений

а) по формуле Крамера;

б) методом Гаусса;

в) методом Зейделя.

13. Разработка приложения, выполняющего решение обыкновенных дифференциальных уравнений

а) методом Эйлера-Коши;

б) методом Рунге-Кутта;

в) методом Адамса.

14. Разработка приложения, выполняющего численное интегрирование

а) по формуле трапеций;

б) по формуле Симпсона;

в) по формуле Гаусса.

15. Разработка приложения, выполняющего решение нелинейных алгебраических уравнений

а) методом половинного деления;

б) методом хорд;

в) методом касательных.

16. Разработка приложения, выполняющего численное дифференцирование

а) на основе интерполяционной формулы Лагранжа;

б) на основе интерполяционной формулы Ньютона;

в) по безразностным формулам численного дифференцирования.

17. Разработка приложения, выполняющего решение системы нелинейных уравнений

а) методом Пикара;

б) методом Зейделя;

в) методом Якоби.

18. Разработка приложения, осуществляющего определение собственных значений и собственных векторов матрицы

а) методом Крылова;

б) методом Леверье-Фадеева;

в) методом Данилевского.

19. Разработка клавиатурного тренажера.

20. Разработка менеджера паролей.

21. Разработка программы для работы с базой данных «Записная книжка».

22. Разработка программы для работы с базой данных «Ежедневник».

23. Разработка плейера для воспроизведения аудиофайлов.

24. Разработка программы статистического анализа текста.

25. Разработка векторного графического редактора.

26. Разработка растрового графического редактора.

27. Разработка приложения для учета рабочего времени сотрудников подразделения.
28. Разработка приложения для работы с базой данных по сотрудникам подразделения.
29. Разработка программы учета занятости аудиторий в вузах.
30. Разработка обучающей программы по формальному языку декорирования и описания внешнего вида документа CSS.
31. Разработка обучающей программы по языку программирования Javascript.
32. Разработка обучающей программы по языку структурированных запросов SQL.
33. Разработка обучающей программы по объектно-ориентированному языку программирования C#.
34. Разработка обучающей программы по скриптовому языку программирования PHP.
35. Разработка обучающей программы по языку программирования TypeScript.
36. Разработка обучающей программы по языку программирования Python.
37. Разработка обучающей программы по языку программирования Java.
38. Разработка обучающей программы по языку программирования Ruby.
39. Разработка обучающей программы по языку гипертекстовой разметки документов HTML.
40. Разработка обучающей программы по языку программирования Kotlin.

О выборе темы студент должен сообщить преподавателю, после чего тема утверждается на заседании кафедры. Не допускается дублирование тем в рамках одного потока. На разработку и оформление курсовой работы студенту отводится 2 – 3 месяца.

После выбора темы следует начинать разработку приложения в соответствии с заданием, выданным преподавателем. На этом этапе обучающийся должен сформулировать основные требования к приложению, определить его функциональные возможности, описать входную и выходную информацию, необходимую для данного приложения. При этом рекомендуется основную задачу разделить на более мелкие, частные задачи, которые в дальнейшем можно будет реализовать в виде отдельных подпрограмм. Таким образом можно определиться с внутренней структурой создаваемого приложения, которую в дальнейшем в случае необходимости можно будет детализировать.

Далее необходимо выбрать оптимальный для решения поставленной задачи язык программирования и соответствующую среду программирования. При этом, безусловно, следует отдавать приоритет свободно распространяемому программному обеспечению, что в дальнейшем облегчит все следующие этапы разработки приложения.

Затем следует спроектировать его пользовательский интерфейс. Качественный интерфейс должен быть интуитивно понятным, дружественным по отношению к пользователю, он должен способствовать удобной и эффективной работе с приложением. При создании интерфейса необходимо выбрать те управляющие элементы (экранные кнопки, текстовые окна, диалоговые окна, переключатели, флажки, списки, элементы меню и другие), которые нужны для нормальной работы приложения. Затем следует разместить их на основной экранной форме. При этом необходимо стремиться к тому, чтобы количество таких элементов было минимальным, чтобы не перегружать форму и снабдить ее только теми элементами, которые действительно нужны пользователю. Также нужно продумать взаимное расположение элементов интерфейса. Это расположение должно быть таким, чтобы пользователю было максимально удобно и комфортно пользоваться этими элементами, то есть приложение должно отвечать требованиям эргономики. Для того, чтобы пользователю было доступно и понятно назначение всех элементов интерфейса их можно снабжать поясняющими надписями и иллюстрациями, не относящимися непосредственно к управляющим элементам. После создания всех элементов интерфейса необходимо произвести начальную настройку свойств для каждого имеющегося в проекте элемента.

Следующим этапом будет написание программного кода приложения. Для этого нужно предварительно продумать логику работы приложения, в основе которой должны находиться современные принципы объектно-ориентированного и событийно-ориентированного программирования. Каждое программное событие, значимое для работы приложения (нажатие определенной клавиши, наведение мыши на объект, щелчок левой или правой клавишей мыши, ввод значения в текстовое поле), должно оформляться в виде соответствующей процедуры (подпрограммы). Строки программного кода должны быть хорошо читаемыми и понимаемыми (это важно, в первую очередь, для самого разработчика приложения). Для этого необходимо не забывать сопровождать программный код соответствующими комментариями, поясняющими выполняемые в программе действия. Если есть такая возможность, то следует для выполнения разного рода рутинных операций, вычисления стандартных математических функций и т.д. использовать имеющиеся в большинстве современных языков и сред программирования стандартные библиотеки.

После написания программного кода приложения следует приступить к его тестированию и отладке. На этом этапе прежде всего необходимо выявить и устранить так называемые ошибки времени выполнения программы. Если синтаксические ошибки сравнительно легко можно найти и исправить еще на этапе разработки программного кода приложения (как правило, компилятор сразу сообщает о таких ошибках с указанием их типа, а часто также и строки, содержащей эту ошибку), то ошибки времени выполнения обнаруживаются только в процессе работы программы. Поэтому программу нужно проверить на предмет ее работоспособности, вводя различные наборы исходных данных. При этом особое внимание следует уделить так называемой «защите от дурака»,

когда пользователь по невнимательности или небрежности вводит в программу явно ошибочные данные. Программа должна адекватно реагировать на подобные ситуации и выдавать четкое и понятное пользователю сообщение об ошибке для того, чтобы он в дальнейшем смог скорректировать вводимую в программу информацию и получить требуемый результат. Только в результате выполнения всех вышеперечисленных действий можно получить работоспособный и эффективный программный продукт.

Когда работа над программой практически доведена до своего логического завершения (хотя ее определенная доработка, исправления и улучшения могут производиться и далее), можно приступать к работе над пояснительной запиской к курсовому проекту, содержание которой будет рассмотрено в следующем разделе данного пособия.

2. СТРУКТУРА ПОЯСНИТЕЛЬНОЙ ЗАПИСКИ К КУРСОВОЙ РАБОТЕ

Пояснительная записка к курсовой работе должна включать в себя следующие основные структурные элементы: титульный лист, задание, оглавление, введение, основную часть (включающую разделы и, в случае необходимости, подразделы), заключение, список литературы.

Титульный лист курсовой работы (образец его оформления приведен в приложении 1) должен включать в себя следующие сведения:

- полное наименование учебного заведения, отделение;
- название темы курсовой работы;
- название вида документа;
- сведения об исполнителе (ФИО студента, номер группы, подпись) и сведения о преподавателе (руководителе) (ФИО, подпись);
- место и год выполнения.

В задании (образец приведен в приложении 2) указываются:

- тема курсового проекта;
- перечень основных вопросов, подлежащих изучению и разработке;
- срок сдачи курсового проекта.

Оглавление должно включать перечень структурных элементов курсового проекта с указанием номеров страниц, с которых начинается их местоположение в тексте, в том числе:

- введение;
- главы, параграфы;
- заключение;
- список литературы;
- приложения (если в них есть необходимость).

Текст введения должен кратко раскрывать актуальность и значение темы курсовой работы. Во введении должна быть сформулирована цель работы и поставлены те конкретные задачи, которые обучающийся должен решить для ее достижения. Здесь же должно быть кратко изложено содержание основных разделов работы.

Основная часть должна содержать обзор литературы по изучаемому вопросу, подробные сведения о выполнении предложенного задания, а также дополнительные сведения. В этой части должны быть описаны технологии, использованные при разработке компьютерного приложения, описана внутренняя структура приложения, а также для наглядности должны быть приведены копии экранов, показывающие разработку пользовательского интерфейса приложения и демонстрирующие работу приложения при различных исходных данных, вводимых пользователем. Также в работе должны быть приведены исходные коды модуля (модулей), входящих в состав разработанного пользователем приложения.

В заключении должны быть приведены выводы о проделанной студентом работе, намечены дальнейшие перспективы деятельности в данной предметной области.

В приложение к курсовой работе рекомендуется вынести информацию справочного характера, которую из-за ее большого объема и относительно малой информативности нецелесообразно помещать в основной текст работы.

Список литературы должен включать библиографический перечень источников (включая web-ресурсы), информация из которых использовалась при выполнении курсовой работы. Все источники должны быть оформлены в соответствии с общепринятыми правилами оформления библиографического аппарата. Эти правила установлены Государственным стандартом (ГОСТ) 7.1-2003 «Библиографическая запись. Библиографическое описание. Общие требования и практика составления».

3. ТРЕБОВАНИЯ К КУРСОВОЙ РАБОТЕ И ПОРЯДОК ЗАЩИТЫ КУРСОВОЙ РАБОТЫ

Текст курсовой работы должен быть распечатан на принтере на одной стороне стандартного листа формата А4. Также допускается представление работы в электронном виде, но при этом все требования, касающиеся ее оформления, должны быть соблюдены точно таким же образом, как и для печатного варианта.

Приведем далее основные требования к оформлению. Текст пояснительной записки должен быть набран в текстовом процессоре Microsoft Word шрифтом с кеглем 14 пунктов с полуторным межстрочным интервалом. Заголовки и подзаголовки разделов пояснительной записки должны быть набраны полужирным шрифтом. Основные заголовки (заголовки первого уровня) должны быть набраны прописными буквами, а подзаголовки (заголовки второго уровня) – строчными. Для основного текста записки рекомендуется использовать шрифт с засечками (например, Times New Roman), а для заголовков лучше применить шрифт без засечек (например, Arial). Для программного кода, приводимого в записке) следует использовать один из моноширинных шрифтов (например, Courier New). Шрифт не должен содержать различные спецэффекты такие, как использование тени и дополнительного контура, имитация объемного текста и т.п. Цвет всех букв, цифр и прочих символов должен быть черным.

Между соседними словами нужно ставить только один пробел. Между знаками препинания (точкой, запятой, точкой с запятой, двоеточием) и предшествующим им словом не должно быть пробела. Пробел должен находится после знака препинания. В тексте следует различать дефис и тире. Дефис используется при образовании сложных слов, обозначении диапазонов чисел и в телефонных номерах. В других случаях (например, для отделения друг от друга частей предложения) используется тире. Для его набора можно использовать комбинацию клавши Ctrl + Alt + «минус» на дополнительной цифровой клавиатуре. В отличие от других знаков препинания, при использовании тире пробел должен стоять и до, и после этого знака.

Заголовки должны быть выровнены по центру, а основной текст – по ширине. В начале каждого абзаца (кроме заголовков) должна быть красная строка размером 1,25 см. Не допускается формирование красной строки с помощью пробелов и табуляции. Для ее создания следует использовать в Word на вкладке «Главная» раздел «Абзац». Необходимо не допускать наличия в документе так называемых висячих строк. Это означает, что в конце страницы не должно быть одной строки, оторванной от абзаца, находящегося на следующей странице, а также нельзя допускать, чтобы на следующую страницу переносилась только одна строка от абзаца, который не помещается на данной странице целиком. Если какой-либо абзац целиком не помещается на текущей странице, то на следующую страницу должны переноситься, как минимум, две строки из этого абзаца. Так же, если на текущей странице помещается только начало абзаца, то на этой странице должно оставаться не менее двух строк данного абзаца. Если выполнить данное условие не получается, то лучше перенести абзац целиком на следующую страницу.

Каждый раздел курсовой работы должен начинаться с новой страницы. Поля страницы курсовой работы должны иметь следующие размеры: левое поле – 30 мм, правое – 10 мм, верхнее и нижнее – 20 мм. Ориентация страницы должна быть книжной. В курсовой работе следует использовать сквозную нумерацию страниц. Номера проставляются, начиная со страницы №3 – оглавления (на титульном листе и задании номера проставлять не надо). Далее нумеруются все страницы, включая приложение. Номер страницы должен находиться внизу страницы (в ее нижнем колонтитуле) и должен быть выровнен по центру.

На все иллюстрации, имеющиеся в пояснительной записке, должны быть сделаны ссылки в тексте пояснительной записки. Сами иллюстрации должны располагаться на той же странице, что и ссылки на них или, если это невозможно, максимально близко к ссылкам. Ко всем иллюстрациям (а также схемам, графикам, диаграммам), имеющимся в пояснительной записке, должны быть сделаны подписи. Подпись отделяется одной строкой от рисунка и выравнивается по центру. Подпись должна начинаться со слова «рисунок», затем указывается его номер и далее, после тире, поясняется содержание рисунка. Все иллюстрации должны быть пронумерованы. Нумерация рисунков может быть сквозной или отдельной для каждого раздела работы. В последнем случае сначала указывается номер раздела, а затем, после точки номер самого рисунка. Например: «Рисунок 3.2 – графический интерфейс приложения». В отличие от текста пояснительной записки иллюстрацию можно выполнить в цвете. Это рекомендуется делать в тех случаях, когда использование разных цветов позволяет сделать иллюстрацию более наглядной и ясной для понимания.

Таблицы в пояснительной записке используют для представления имеющейся в записке числовой информации в наглядном и упорядоченном виде. Таблицы создаются средствами текстового процессора Microsoft Word с помощью вкладки «Вставка», а редактируются с помощью вкладки «Макет». Ширина таблицы должна соответствовать ширине текстового блока пояснительной записки. Все таблицы так же, как и иллюстрации, должны быть пронумерованы. На каждую таблицу так же должна быть сделана ссылка в тексте записки, причем таблица должна быть расположена максимально близко к этой ссылке. Таблица должна быть снабжена надписью, которую располагают сверху от самой таблицы и выравнивают по левому краю текста. Формат надписи такой же, как и для иллюстрации, то есть она начинается со слова «таблица», затем указывается ее номер и далее, после тире, поясняется содержание таблицы. Текст, расположенный в ячейках таблиц, может быть меньшего размера, чем основной текст документа (например, он может быть набран кеглем 12). Не следует оставлять в таблице пустые ячейки. Если какая-либо ячейка не должна содержать никакой информации, то в этой ячейке следует ставить прочерк (то есть тире).

Имеющиеся в тексте пояснительной записки уравнения и формулы следует помещать на отдельную строку. Для формул используется сквозная нумерация, а сам номер заключается в круглые скобки и выравнивается по правому краю листа. Нумеровать необходимо только те формулы, ссылки на которые имеются в тексте пояснительной записки. Расшифровка значений

символов, используемых в формуле, приводится сразу после формулы, причем в том же порядке, как они написаны в самой формуле. Первая строка этой расшифровки начинается со слова «где» без двоеточия.

После завершения работы над пояснительной запиской студент представляет разработанное им в соответствии с заданием приложение вместе с пояснительной запиской на проверку преподавателю. По результатам проверки преподаватель составляет отзыв, который передается студенту для подготовки к защите курсового проекта. В отзыве преподаватель дает общую характеристику работы, отмечает имеющиеся в работе недостатки, которые необходимо устранить до начала защиты, а также указывает предварительную оценку курсовой работы, которая в дальнейшем может измениться в ту или иную сторону, в зависимости от результатов защиты курсовой работы.

На защите курсовой работы студент должен представить электронную презентацию, в которой должны быть отражены цель курсовой работы, основные этапы ее выполнения, полученные студентом результаты и основные выводы по итогам работы. В ходе защиты студент должен показать результаты, полученные им в процессе работы над курсовым проектом, доказать актуальность проделанной им работы, аргументированно изложить выводы, логически следующие из данной работы, ответить на вопросы членов комиссии, принимающей курсовую работу.

По итогам защиты курсового проекта комиссия выставляет студенту окончательную оценку. При выставлении итоговой оценки учитываются глубина усвоения студентом материалов как всей дисциплине в целом, так и по конкретной теме курсовой работы, степень самостоятельности, проявленная при выполнении работы, качество оформления пояснительной записки, качество доклада, представленного в ходе защиты, уровень компетентности, продемонстрированный при ответе на вопросы членов комиссии, предварительная оценка, выставленная преподавателем, осуществлявшим руководство студентом.

По результатам выполнения и защиты курсовой работы комиссией может быть поставлена одна из следующих оценок: «отлично», «хорошо», «удовлетворительно» и «неудовлетворительно». Итоговая оценка складывается из двух оценок − оценки за качество выполненной работы и оценки за качество ее защиты.

При неудовлетворительной оценке курсовой работы студент должен полностью переработать курсовой проект с учетом замечаний, высказанных членами комиссии. В то же время студент, представивший курсовой проект, но получивший неудовлетворительную оценку, имеет право на его повторную защиту, время и место проведения которой определяется комиссией. При этом комиссия также решает вопрос о том, можно ли допустить студента к повторной защите курсовой работы по той же теме, или же он должен разработать проект по новой теме.

4. ПРИМЕР ВЫПОЛНЕНИЯ КУРСОВОЙ РАБОТЫ

Целью данной курсовой работы в соответствии с выданным заданием является разработка на языке C# в среде программирования Microsoft Visual Studio приложения, которое выполняет численное интегрирование следующими методами:

а) по формуле прямоугольников;

б) по формуле трапеций;

в) по формуле парабол (Симпсона);

г) по формуле Гаусса.

Приложение должно содержать графический интерфейс, который позволяет пользователю выбрать любой из вышеперечисленных методов, задать основные параметры интегрирования и получить результат вычислений в наглядном виде.

4.1. Теоретические основы курсовой работы

Работу над курсовым проектом начнем с рассмотрения основных теоретических положений, лежащих в основе методов численного интегрирования. Именно эти методы будут в дальнейшем реализованы в виде программы, написанной на языке C#.

При решении ряда задач возникает необходимость вычисления определенного интеграла от некоторой функции:

$$I = \int_a^b f(x) \cdot dx \,, \tag{4.1}$$

где $f(x)$ – подынтегральная функция на участке $[a, b]$.

Геометрический смысл интеграла заключается в том, что если $f(x) \geq 0$ на отрезке $[a, b]$, то интеграл численно равен площади фигуры, ограниченной графиком функции $y = f(x)$, отрезком оси абсцисс, прямой $x = a$ и прямой $x = b$ (см. рис. 4.1). Таким образом, вычисление интеграла равносильно вычислению площади криволинейной трапеции.

Классическим способом вычисления определенного интеграла является использование формулы Ньютона-Лейбница, согласно которой, если функция *f(x)* непрерывна на отрезке $[a,b]$, и *F(x)* – любая ее первообразная на этом отрезке, то имеет место равенство:

$$\int_a^b f(x)dx = F(b) - F(a) \,. \tag{4.2}$$

После нахождения первообразных несложно вычислить и сам интеграл. Но приведенный выше способ вычисления определенного интеграла по формуле 4.2 не всегда возможен и целесообразен. Тогда для вычисления интеграла используются другие способы, к которым относится выражение интеграла через хорошо изученные и затабулированные неэлементарные специальные функции, разложение подынтегральной функции в ряды специального вида, графическое интегрирование и численное интегрирование.

Последнее является наиболее универсальным методом, пригодным к интегралам, заданным любым способом.

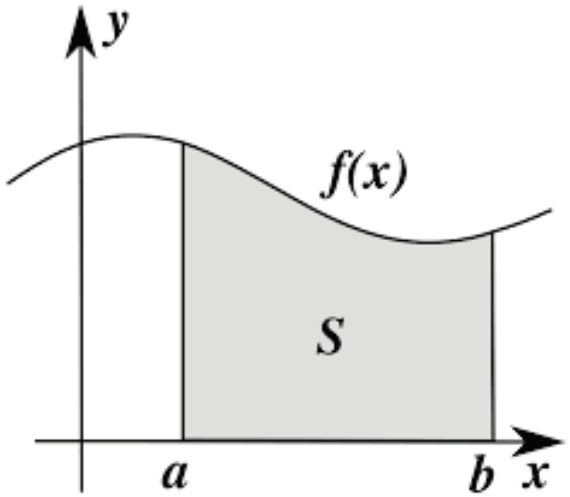

Рисунок 4.1 – Геометрический смысл определенного интеграла

Численное интегрирование представляет собой набор численных методов, применяемых для вычисления значения определенного интеграла. Численное интегрирование, как правило, используется в тех случаях, когда вычисление определенного интеграла по формуле Ньютона-Лейбница либо в принципе невозможно, либо оказывается слишком длительным и сложным процессом. Первая ситуация возникает, например, когда подынтегральная функция вообще не задана аналитически, а представляет собой просто набор значений, заданных в виде таблицы или массива. Во втором случае вид первообразной может быть настолько сложен, что быстрее будет вычислить значение определенного интеграла численным методом.

Формулы численного интегрирования дают приближенные значения определенного интеграла, если известны подынтегральной функции в некоторых точках интервала интегрирования, называемых узлами.

Процесс численного интегрирования начинается с того, что интервал интегрирования от a до b разбивается на некоторое число равных частей, количество которых составляет n. При этом заданы или вычислены значения подынтегральной функции в точках деления, то есть узлах.

Длина каждой части, на которую делится отрезок составляет:

$$h = \frac{b-a}{n}$$

Способы численного интегрирования для определенных интегралов основаны на замене интеграла конечной суммой:

$$\int_a^b f(x)dx \approx \sum_{i=1}^{n} c_i f(x_i), \qquad (4.3)$$

где c_i – числовые коэффициенты, выбор которых зависит от выбранного метода численного интегрирования; x_i – узлы интегрирования; $x_i \in [a,b], i = 1,2,\dots n$. Выражение 4.3 называют квадратурной формулой.

Исходя из вышеизложенного, значение определенного интеграла можно представить в следующем виде:

$$\int_a^b f(x)dx \approx \sum_{i=1}^{n} \int_{x_{i-1}}^{x_i} f(x)dx. \quad (4.4)$$

Из данного выражения следует, что для численного интегрирования на отрезке *[a,b]* достаточно построить квадратурную формулу на каждом частичном отрезке *[xi-1,xi]*.

Одним из наиболее простых методов численного интегрирования является метод трапеций. Задача в данном случае состоит в том, чтобы вычислить интеграл следующего вида:

$$\int_a^b y\,dx, \text{ где } y = f(x). \quad (4.5)$$

Для дальнейших расчетов используем следующие обозначения:

$$f(a) = y_0,\ f(a+h) = y_1,\quad f(a+2h) = y_2,\ \dots.$$
$$f(a+nh) = f(b) = y_n$$

Если провести ординаты для каждого из полученных узлов, то криволинейная трапеция, площадь которой равна интегралу, разбивается на n столбцов, каждый из которых также представляет собой криволинейную трапецию. Эти столбцы можно заменить прямолинейными трапециями, построенными на крайних ординатах, как это показано на рисунке 4.2.

Площади этих трапеций соответственно равны:

$$\frac{y_0+y_1}{2}h, \qquad \frac{y_1+y_2}{2}h, \dots, \frac{y_{n-1}+y_n}{2}h.$$

Сложив эти площади, мы получим площадь многоугольной фигуры, вписанной в исходную криволинейную трапецию. Если *n* достаточно велико (то есть *h* достаточно мало), то эта площадь приближенно равна площади криволинейной трапеции, то есть интегралу, и мы получаем следующую формулу:

$$\int_a^b y\,dx \approx \frac{y_0+y_1}{2}h + \frac{y_1+y_2}{2}h + \dots + \frac{y_{n-1}+y_n}{2}h$$

или, после преобразования:

$$\int_a^b y\,dx \approx h\left(\frac{y_0+y_n}{2} + y_1 + y_2 + \dots + y_{n-1}\right). \quad (4.6)$$

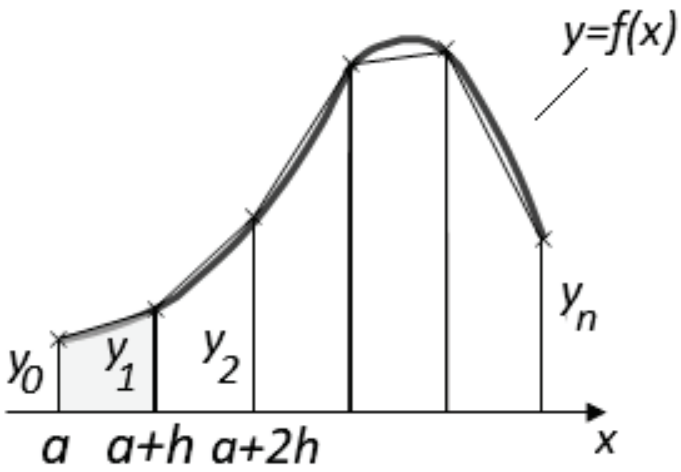

Рисунок 4.2 – Численное интегрирование методом трапеций

Другим достаточно простым и удобным способом численного интегрирования является метод прямоугольников. На частичном отрезке $[x_{i-1}, x_i]$ подынтегральная функция заменяется полиномом Лагранжа нулевого порядка, построенном в одной точке. В качестве этой точки можно выбрать середину частичного отрезка $x_{i-0.5}=x_i - 0.5h$, поэтому данный метод часто также называют методом средних прямоугольников. Тогда значение интеграла на частичном отрезке будет равно:

$$\int_{x_{i-1}}^{x_i} f(x)dx \approx f(x_{i-0.5})h\,. \qquad (4.7)$$

Подставив это выражение в формулу 4.4 получаем составную формулу средних прямоугольников:

$$\int_{x_{i-1}}^{x_i} f(x)dx \approx \sum_{i=1}^{n} f(x_{i-0.5})h\,. \qquad (4.8)$$

Графическое изображение метода средних прямоугольников приведено на рисунке 4.3. Из данной иллюстрации видно, что площадь криволинейной трапеции приближенно равна площади прямоугольника, который составлен из n многоугольников. Таким образом, вычисление определенного интеграла сводится к вычислению суммы n элементарных прямоугольников.

Наряду с методом среднего прямоугольника также используются методы левого и правого прямоугольников, но они дают гораздо большую погрешность и поэтому далее рассматриваться не будут.

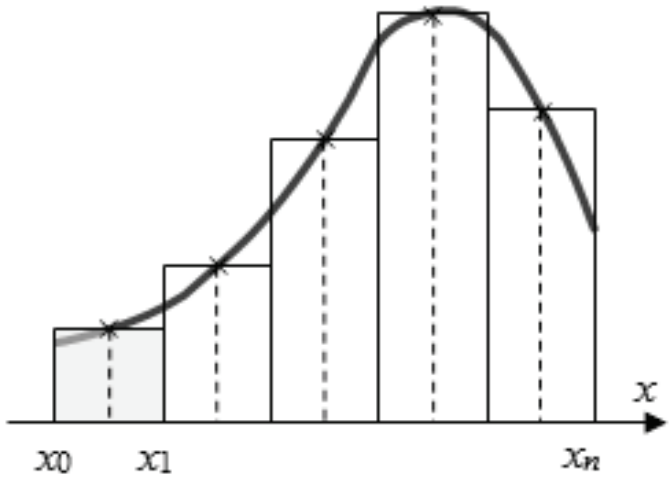

Рисунок 4.3 – Численное интегрирование методом средних прямоугольников

Если для численного интегрирования применить интерполяционный полином второй степени, то получится метод Симпсона, названный по имени английского математика XVIII века Т. Симпсона, который вывел соответствующую формулу. Сначала предположим, что значения интегрируемой функции даны в трех точках:

$$y(x_0)=y_0, \quad y(x_0+h)=y_1, \; y(x_0+2h)=y_2 .$$

Тогда интерполяционный полином второй степени, принимающий в этих точках такие же значения, определяется по формуле Лагранжа:

$$P(x) = y_0 + \Delta y_0 \frac{s}{h} + \frac{\Delta^2 y_o}{2}\frac{s}{h}\left(\frac{s}{h} - 1\right), \text{ где } s = x - x_0.$$

Исходя из этого получим для интеграла следующее выражение:

$$\int_{x_0}^{x_0+2h} P(x)dx = \int_0^{2h}[y_0 + \Delta y \frac{s}{h} + \frac{\Delta^2 y_o}{2}\left(\frac{s^2}{h^2} - \frac{s}{h}\right)]ds = y_0 2h + \Delta y_0 2h + \frac{\Delta^2 y_0}{2}\frac{2}{3}h.$$

Если далее подставить:

$$\Delta y_0 = y_1 - y_0, \quad \Delta^2 y_0 = \Delta y_1 - \Delta y_0 = (y_2 - y_1) - (y_1 - y_0) = y_2 - 2y_1 + y_0,$$

то после приведения подобных членов получим;

$$\int_{x_0}^{x_0+2h} f(x)dx \approx \int_{x_0}^{x_0+2h} P(x)dx = h\frac{y_0+4y_1+y_2}{3}. \qquad (4.9)$$

Если теперь отрезок интегрирования $a \le x \le b$ разбить на 2n равных частей с помощью точек деления:

$$x_0 = a, x_1 = a + h, x_2 = a + 2h, \quad \ldots, x_{2n} = a + 2nh = b, \text{ где } h = \frac{b-a}{2n},$$

то к каждой паре из этих частей можно применить формулу:

$$\int_{x_0}^{x_0+2h} y\,dx \approx h\frac{y_0+4y_1+y_2}{3}, \int_{x_0+2h}^{x_0+4h} y\,dx \approx h\frac{y_2+4y_3+y_4}{3}, \ldots,$$

$$\int_{x_0+2(n-2)h}^{x_0+2nh} y\,dx \approx h\frac{y_{2n-2}+4y_{2n-1}+y_{2n}}{3}.$$

Сложив эти формулы и приведя подобные члены, получим формулу Симпсона:

$$\int_a^b ydx \approx \frac{h}{3}[(y_0 + y_{2n}) + 2(y_2 + y_4 + \cdots + y_{2n-2}) + \\ +4(y_1 + y_3 + \cdots + y_{2n-1})]. \quad (4.10)$$

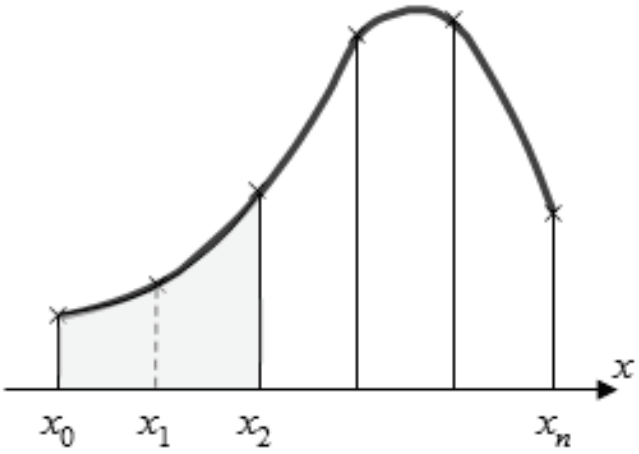

Рисунок 4.4 – Численное интегрирование методом Симпсона (методом парабол)

На рисунке 4.4 приведено графическое представление метода Симпсона. На каждом из частичных отрезков дуга кривой заменяется параболой, поэтому данный метод часто также называют методом парабол.

Все рассмотренные выше методы численного интегрирования основаны на одном общем алгоритме: интегрируемая функция интерполируется на отрезке интегрирования по равноотстоящим узлам с помощью полинома Лагранжа, для которого аналитически вычисляется значение интеграла. Группа методов, которые основаны на таком подходе, называется методами Ньютона-Котеса.

Следующий рассматриваемый в данной работе метод называется методом Гаусса. Его основное отличие от методов Ньютона-Котеса состоит в том, что узлы интегрирования x_i на отрезке $[x_{j-1},x_j]$ располагаются не равномерно, а выбираются таким способом, чтобы при наименьшем возможном числе узлов точно интегрировать многочлены наивысшей возможной степени. Интегрирование выполняется по следующей формуле:

$$\int_a^b f(x)dx \approx \frac{b-a}{2n} \cdot \sum_{j=1}^{n} \sum_{i=0}^{n} c_{in} \cdot f(x_i). \quad (4.11)$$

Узлы x_i являются корнями полинома Лежандра степени n, а веса вычисляются интегрированием полиномов Лежандра по формуле:

$$a_i = \frac{2}{(1-x_i^2)\cdot[P_n'(x_i)]^2}, \quad (4.12)$$

где P_n' – первая производная полинома Лежандра.

Для интегрирования на произвольном частичном отрезке необходимо пересчитать значения узлов для данного частичного отрезка [x_{j-1},xj] по следующей формуле:

$$x_i = x_{j-1} + \frac{(x_{i[-1;1]}+1)(x_{j-1}-x_j)}{2} \quad (4.13)$$

В приведенной ниже таблице 4.1 указаны весовые коэффициенты метода Гаусса для количества узлов от 1 до 5.

Таблица 4.1 – Весовые коэффициенты метода Гаусса

n	i	x_i[-1;1]	c_i
1	1	0	2
2	1	-0.5773503	1
	2	0.5773503	1
3	1	-0.7745967	0.5555556
	2	0	0.8888889
	3	0.7745967	0.5555556
4	1	-0.8611363	0.3478548
	2	-0.3399810	0.6521451
	3	0.3399810	0.6521451
	4	0.8611363	0.3478548
5	4	-0.9061798	0.4786287
	2	-0.5384693	0.2369269
	3	0	0.5688888
	4	0.5384693	0.2369269
	5	0.9061798	0.4786287
6	1	-0.9324700	0.1713245
	2	-0.6612094	0.3607616
	3	-0.2386142	0.4679140
	4	0.2386142	0.4679140
	5	0.6612094	0.3607616
	6	0.9324700	0.1713245

Квадратура Гаусса относится к квадратурам открытого типа. Это означает, что ни один из узлов не совпадает ни с одним из концов отрезка интегрирования *a* или *b*. Веса квадратур Гаусса всегда положительны, и при

увеличении числа узлов точность приближения почти всегда возрастает. Как правило, для получения удовлетворительного результата оказывается достаточным использование пяти узлов.

4.2. Создание пользовательского интерфейса приложения

Описанные выше методы численного интегрирования должны быть реализованы в виде программы, составленной на одном из языков программирования высокого уровня, и имеющей полноценный графический пользовательский интерфейс. Поэтому следующим шагом в решении задачи, поставленной в курсовой работе, должен быть выбор языка программирования и среды программирования.

В качестве языка программирования для решения поставленной задачи был выбран объектно-ориентированный язык программирования общего назначения C#. Выбор данного языка обусловлен тем, что он оптимизирован для разработки Windows-приложений и изначально разрабатывался как язык для решения прикладных задач, к числу которых относятся и численные методы решения математических задач. Одной из разновидностей этих методов и является численное интегрирование.

В качестве среды программирования была использована интегрированная среда разработки Microsoft Visual Studio, которая поддерживает современные технологии объектно-ориентированного программирования и визуального проектирования приложений. Эта среда программирования позволяет создавать как консольные приложения, так и приложения с графическим пользовательским интерфейсом, что является основной целью данной работы.

Работу над созданием приложения начнем с запуска пакета Microsoft Visual Studio и создания нового проекта. После запуска программы Microsoft Visual C#, содержащейся в данном пакете, любым удобным для пользователя способом (это может быть обращение к соответствующему пункту, который появляется в главном меню операционной системы Windows после установки Visual Studio, или использование ярлыка программы, вынесенного на рабочий стол Windows) на экране компьютера появляется начальное окно, в котором предлагается выбрать дальнейший вариант действий.

В этом окне предлагается возможность открыть уже существующий проект или создать новый (рис. 4.5). В самом начале работы над проектом следует выбрать второй вариант. В дальнейшем во время следующих сеансов работы нужно выбирать первый вариант, то есть открытие проекта.

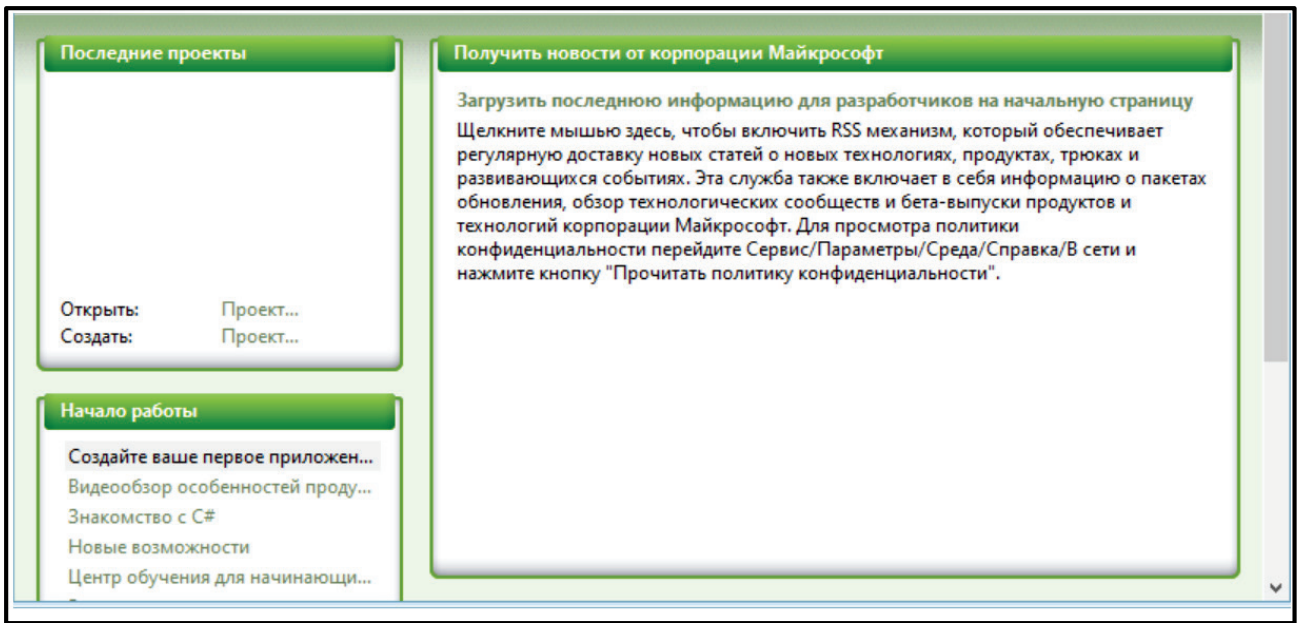

Рисунок 4.5 – Начальное окно среды программирования Visual C#

После выбора варианта «Создать» на экране компьютера появляется еще одно дополнительное диалоговое окно в котором предлагается выбрать тип создаваемого приложения (рис. 4.6). Для создания приложения, имеющего полноценный графический интерфейс пользователя, нужно в этом окне выбрать вариант «Приложение Windows Forms», затем следует выбрать в строке, расположенной в нижней части окна, имя проекта и подтвердить сделанный выбор щелчком на экранной кнопке «Ok».

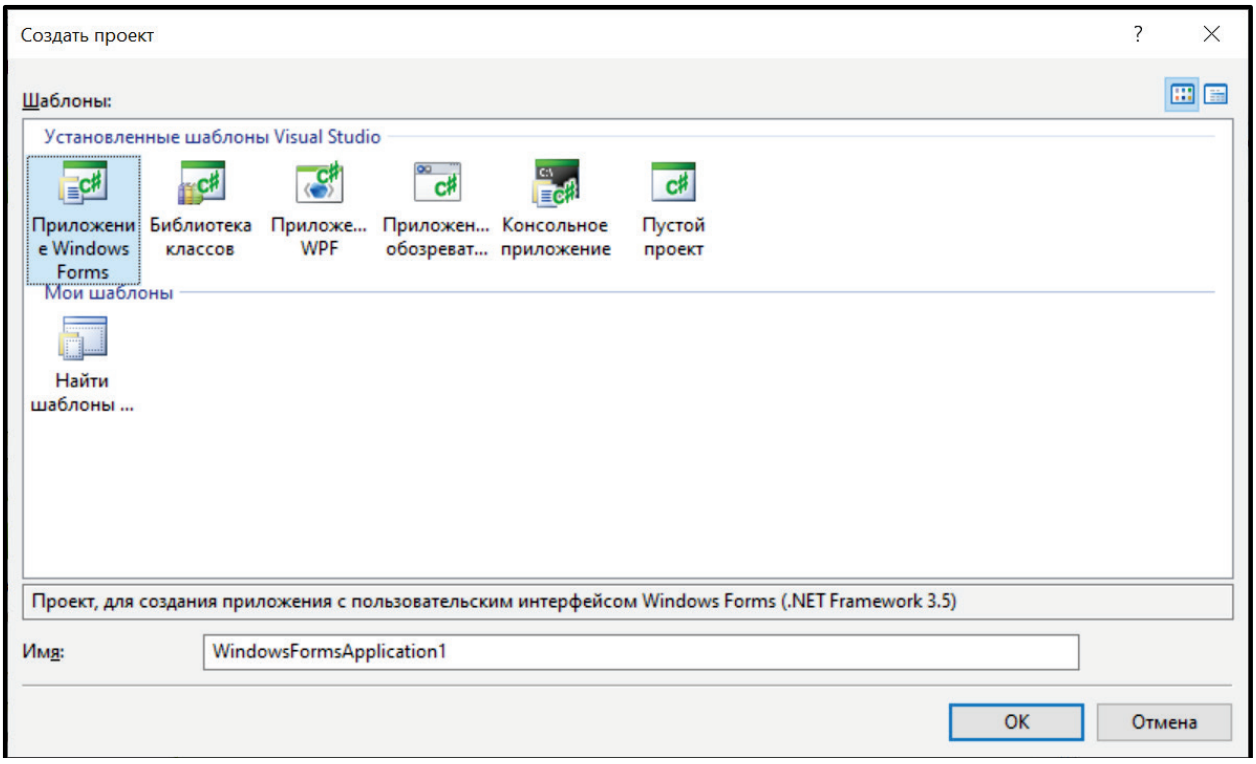

Рисунок 4.6 – Окно выбора типа создаваемого приложения в среде Visual C#

Среда Visual Studio создаст для проекта новую папку с таким же именем, как и у проекта. Далее в результате вышеуказанных действий на экране компьютера появляется окно для конструирования пользовательского интерфейса приложения, имеющее вид, приведенный на рисунке 4.7. Центральное место в этом окне занимает экранная форма с именем Form1. Режим, в котором сейчас работает среда программирования, называется конструкторским.

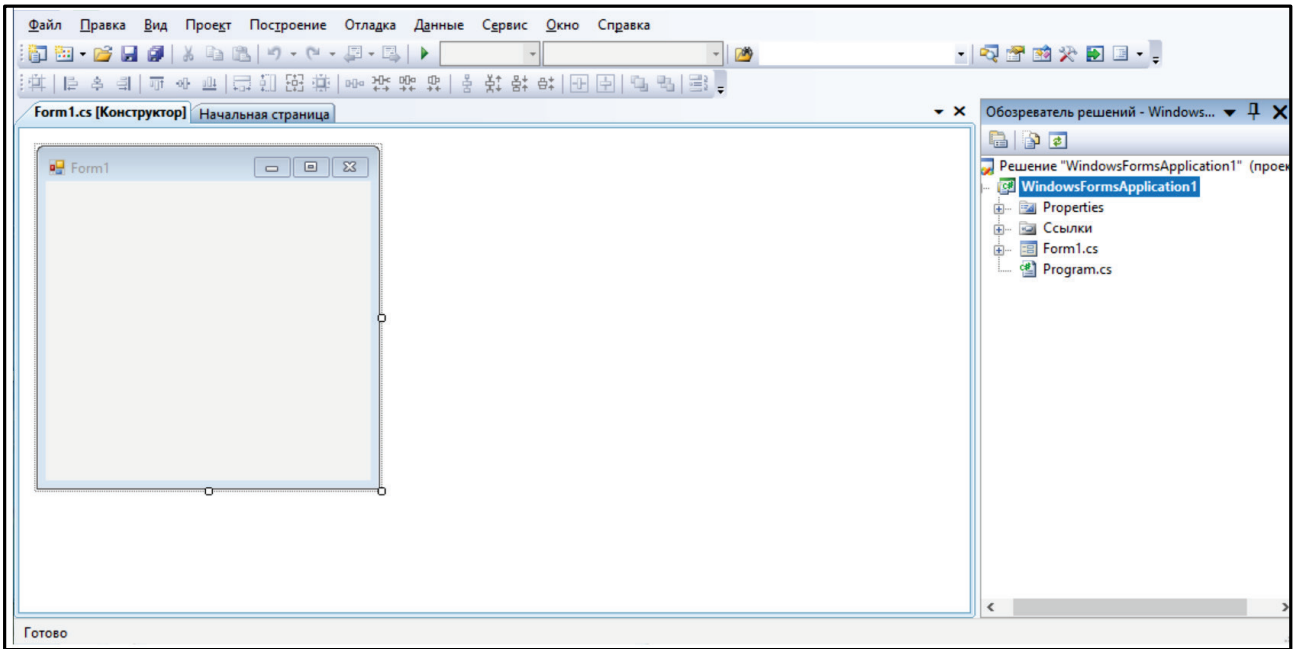

Рисунок 4.7 – Начальный экран конструктора приложения

Форма, созданная в конструкторском режиме, представляет собой визуальное представление окна, которое появляется при запуске приложения на выполнение. Именно на этой форме Form1 и создается пользовательский интерфейс приложения, который должен включать в себя следующие элементы:

а) элементы для ввода исходных данных, которые включают в себя два текстовых окна для ввода пределов интегрирования и текстовое окно для ввода количества участков, на которые разбивается область интегрирования; эти окна должны быть также снабжены поясняющими надписями для обеспечения дружественного интерфейса пользователя, то есть удобной и комфортной работы с приложением, когда пользователю всегда понятно, что именно ему нужно делать в данный момент;

б) элементы для выбора метода интегрирования, представляющие собой 4 экранные кнопки: первая для метода прямоугольников, вторая для метода трапеций, третья для метода парабол (метода Симпсона), четвертая для метода Гаусса; на каждой кнопке также должна быть помещена надпись, поясняющая ее назначение;

в) 4 окна, расположенных под экранными кнопками, в каждом из которых находится результат вычисления интеграла соответствующим методом, также сопровождаемые поясняющими надписями;

г) две вспомогательные экранные кнопки; первая – это кнопка «Сброс», которая позволяет очистить все окна с исходными данными и с результатами вычислений; вторая – кнопка «Выход», с помощью которой приложение закрывается.

Для дальнейшей работы над интерфейсом необходимо, чтобы в окне конструктора отображались панель элементов, расположенная слева от основной формы и окно свойств, находящееся справа от нее (рис. 4.8). Если изначально они отсутствуют, то их можно вывести на экран с помощью соответствующих пунктов, имеющихся в разделе меню «Вид».

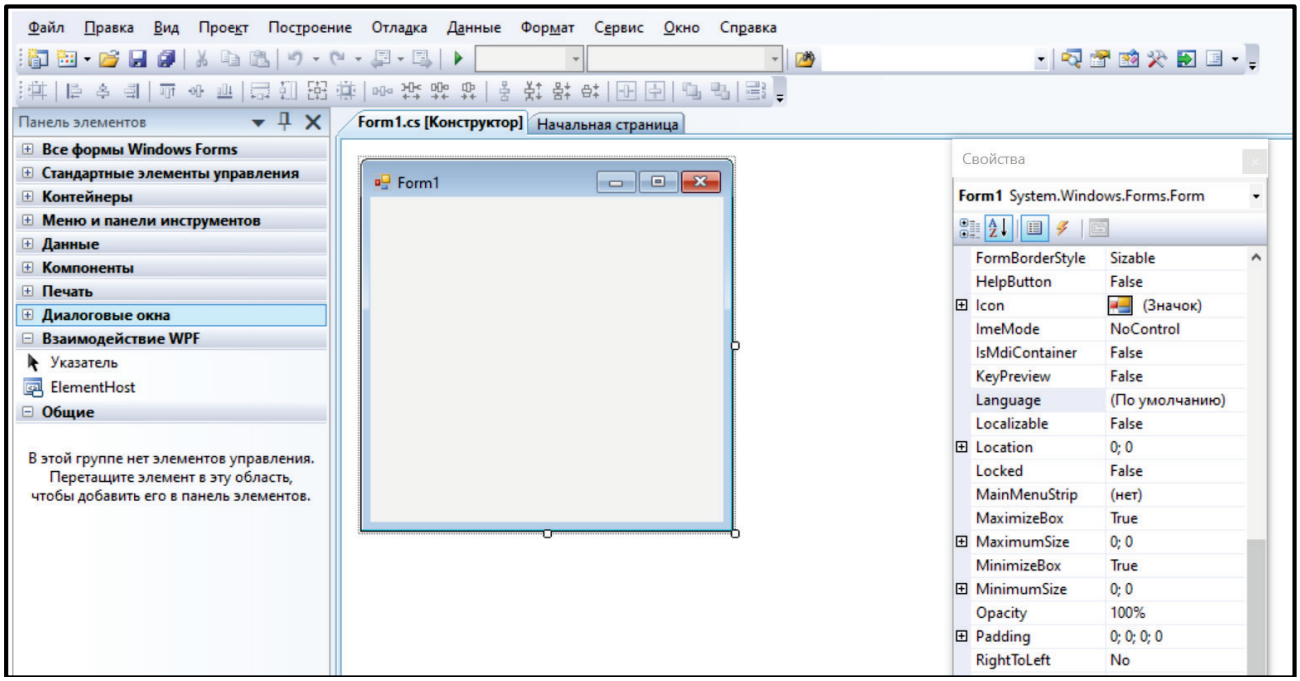

Рисунок 4.8 – Экран конструктора приложения с выведенными на него панелью элементов и окном свойств

Панель элементов содержит заготовки для тех элементов интерфейса, которые в дальнейшем можно будет разместить на основной экранной форме. Для удобства пользователя эти элементы объединены в несколько групп: все формы, стандартные элементы управления, контейнеры, меню и панели инструментов, данные, компоненты, печать, диалоговые окна и другие. Слева от названия каждой группы находится небольшой квадрат со знаком «плюс». Щелкнув по этому квадратику, можно открыть любую группу и ознакомиться с содержащимися в ней элементами (рис. 4.9). Например, открыв группу «Стандартные элементы управления», можно увидеть, что она содержит такие элементы, как Button (экранная кнопка), CheckBox (флажок, который можно устанавливать или сбрасывать), Label (надпись), MonthCalendar (календарь), PictureBox (окно для вставки изображения), RadioButton (переключатель, называемый также радиокнопкой), TextBox (текстовое окно) и целый ряд

других. Именно элементы, входящие в эту группу, чаще всего применяются для разработки пользовательского интерфейса.

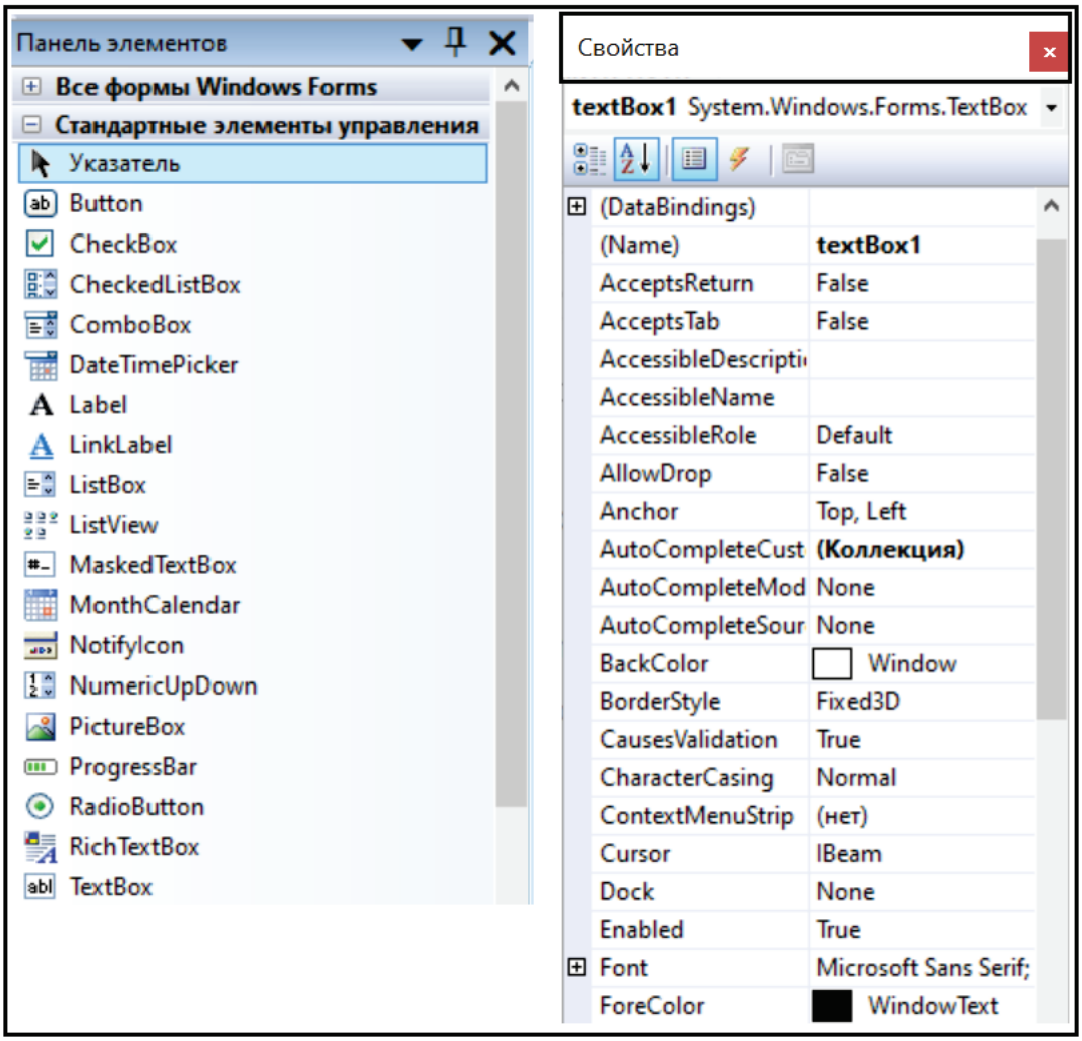

Рисунок 4.9 – Панель элементов (раскрыта группа свойств «Стандартные элементы управления») и окно свойств (показаны свойства объекта «Button»)

Когда пользователь выбирает на панели элементов какой-либо элемент управления, в окне свойств отображается список свойств выбранного элемента, приведенный в алфавитном порядке. В качестве примера на рисунке 4.9 приведен список свойств объекта Button, то есть экранной кнопки. У этого объекта имеются такие свойства как Name (имя), BackColor (фоновый цвет объекта), Font (параметры шрифта), ForeColor (цвет надписи на объекте), Text (содержание надписи на объекте), Visible (видимость объекта) и ряд других. Большинство этих свойств имеются и у други расположенных на панели элементов пользовательского интерфейса.

Необходимо отметить, что часть приведенных выше свойств являются составными, то есть при выборе подобного свойства появляется дополнительное диалоговое окно, позволяющее осуществить его окончательную настройку. В частности, это относится к таким свойствам, как BackColor, ForeColor и Font. Рассмотрим их более детально.

Если щелкнуть левой клавишей мыши на свойстве BackColor, то справа от названия свойства появляется квадратик с направленной вниз стрелкой, который и открывает новое диалоговое окно (рис. 4.10). В этом диалоговом окне содержится три вкладки: пользовательская, Web и «Система».

Пользовательская вкладка содержит палитру, из которой разработчик может выбрать нужный ему цвет. Вкладка Web содержит раскрывающийся список с перечнем тех цветов, которые распознают практически все современные web-браузеры. В этой вкладке каждому цвету, находящемуся в списке, соответствует прямоугольник, закрашенный этим цветом, и находящееся справа от него название цвета на английском языке. Вкладка «Система» по своей структуре аналогична вкладке Web, но перечень цветов там меньше, поскольку она содержит только те цвета, в которые окрашены элементы интерфейса Windows.

Свойство ForeColor включает в себя те же три вкладки, что и BackColor, но она задает цвета не для фона объекта, а для символов, содержащихся в надписи, расположенной на данном объекте.

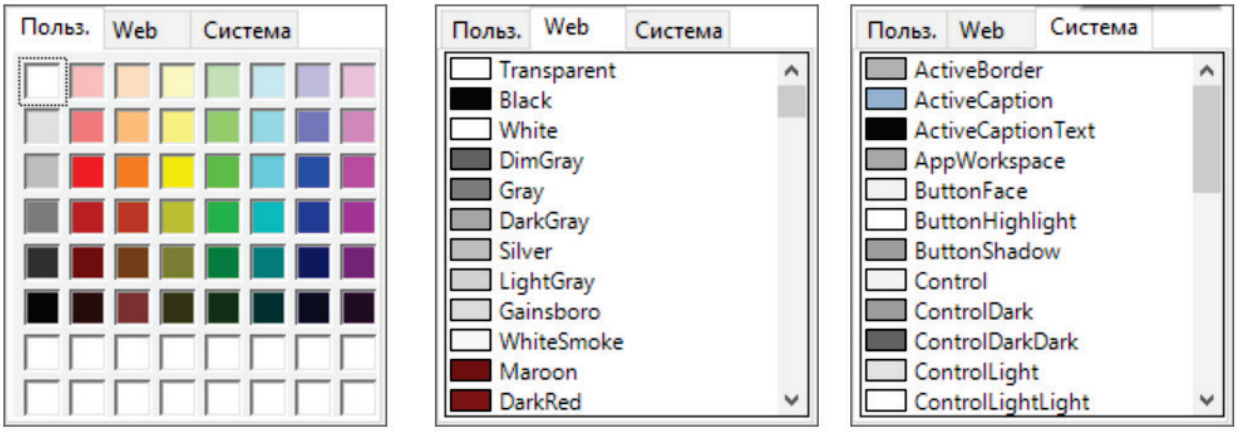

Рисунок 4.10 – Вкладки свойств BackColor и ForeColor: «Пользовательская», Web и «Система»

Также составным является свойство Font, содержащее набор параметров шрифта. Слева от названия свойства имеется небольшой квадрат с плюсом. Если щелкнуть по нему левой клавишей мыши, то справа от названия появляется квадратная кнопка с многоточием. После щелчка на этой кнопке появляется дополнительное диалоговое окно, в котором можно настроить различные свойства шрифта (рис. 4.11). В этом окне можно установить гарнитуру (то есть вид) шрифта, его начертание (обычное, полужирное или курсивное), размер, выраженный в пунктах. Кроме того, можно использовать такие дополнительные возможности оформления шрифта, как его подчеркивание или зачеркивание. Для этого нужно установить один из флажков, находящихся внизу окна. После завершения настроек нужно щелкнуть мышью на экранной кнопке «Ok», и все выполненные изменения вступят в силу.

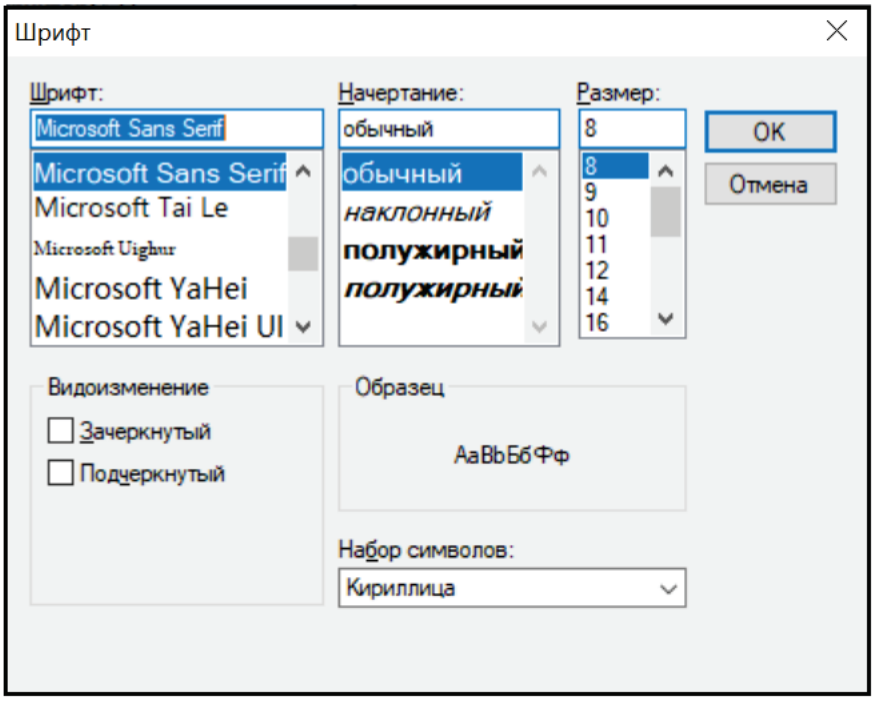

Рисунок 4.11 – Окно настройки шрифта, открывающееся в свойстве Font

Для остальных (не составных) свойств любого объекта в строке справа от названия свойства нужно ввести текстовое значение (например, для свойства Text), численное значение (например, для свойства Size, определяющего размеры объекта) или выбрать значение из открывающегося списка (например, для свойства Visible). Последнее из перечисленных определяет видимость объекта и относится к числу логических, поскольку в раскрывающемся списке для него имеется всего два значения: True (в этом случае объект виден) и False (при выборе этого значения объект становится невидимым).

После ознакомления с интерфейсом среды программирования Visual C# можно приступать непосредственно к конструированию пользовательского интерфейса будущего приложения. Работу начнем с настройки свойств основной экранной формы Form1. Эта форма является таким же объектом, как и другие элементы интерфейса, которые впоследствии будут на ней размещены. Щелкнув мышью по заголовку формы, активизируем расположенное справа окно ее свойств. В этом окне для свойства Text вводит значение «Численное интегрирование тригонометрической функции sin(x)». Для свойства BackColor открываем пользовательскую палитру и в этой палитре выбираем светло-зеленый цвет (вместо серого, в который форма окрашена по умолчанию).

Если размеры формы, предлагаемые системой программирования, слишком малы, для того чтобы разместить на ней все необходимые элементы интерфейса (как в нашем случае), то можно эти размеры увеличить. Для этого в середине нижней границы формы, в середине правой границы и в ее правом нижнем углу имеются специальные маркеры (небольшие прозрачные квадратики). «Ухватившись» мышью за один из этих маркеров и перетащив его в нужном направлении, можно изменить размеры формы соответственно по вертикали, по горизонтали или по диагонали.

Когда первоначальная настройка свойств формы произведена, можно приступать к размещению на ней элементов пользовательского интерфейса приложения. Первой группой элементов являются текстовые окна для ввода исходных данных и надписи, поясняющие их назначение. По условиям задания нам необходимо иметь на форме три текстовых окна: в первом окне вводится начальное значение отрезка интегрирования *a*, во втором окне – конечное значение этого отрезка *b*, а в третьем окне – количество участков, на которые разбивается отрезок интегрирования *n*. Вначале нужно выбрать необходимый элемент на панели элементов (в данном случае это элемент Textbox, расположенный в группе «Стандартные элементы управления»), щелкнув по нему мышью, а затем еще раз щелкнуть мышью в том месте основной экранной формы, где его необходимо поместить. Таким образом на экранной форме мы создаем текстовые окна textBox1, textBox2 и textBox3 (имена этих объектов, как и прочих, создаваемых в данном приложении, специально вводить в окно свойств не надо, так как они даются средой программирования автоматически).

Для создания пояснений к текстовым окнам выбираем на панели элементов в той же группе «Стандартные элементы управления» элемент Label и переносим его на экранную форму. На форме создается объект label1. В окне свойств для свойства Text этого объекта вводим значение «Введите границы интегрирования». После нажатия на клавиатуре компьютера клавши Enter это значение отобразится и на экранной форме. Повторив эту операцию, создаем на форме объекты label2, label3 и label4. Для свойства Text этих объектов вводим соответственно значения «a», «b» и «Введите количество участков интегрирования». В случае необходимости изменения свойств этих объектов (например, увеличения размера шрифта для объектов) обращаемся к рассмотренному ранее окну свойств и вводим там необходимые значения.

Затем создаем вторую группу объектов, которые будут использоваться непосредственно для вычислений искомых значений. Вверху этой группы находится надпись label5, содержание которой – «Выберите метод интегрирования». Под ней находятся четыре экранные кнопки button1, button2, button3 и button4, соответствующие четырем используемым в программе методам численного интегрирования. На каждой кнопке (с помощью свойства Text) должна быть помещена надпись, поясняющая какой именно метод используется в данном случае. Для button1 – это «прямоугольников», для button2 – «трапеций», для button3 – «Симпсон (парабол)», для button4 – «Гаусса». Под каждой из перечисленных экранных кнопок создаем текстовое окно, в которое будет выводиться вычисленное значение определенного интеграла. Этим четырем текстовым окнам среда программирования присвоила названия textbox4, textbox5, textbox6 и textbox7. Слева от каждого текстового окна помещаем надпись, которая поясняет, что в нем содержится значение интеграла, вычисленное данным способом. Эти надписи получают имена label6, label7, label8 и label9.

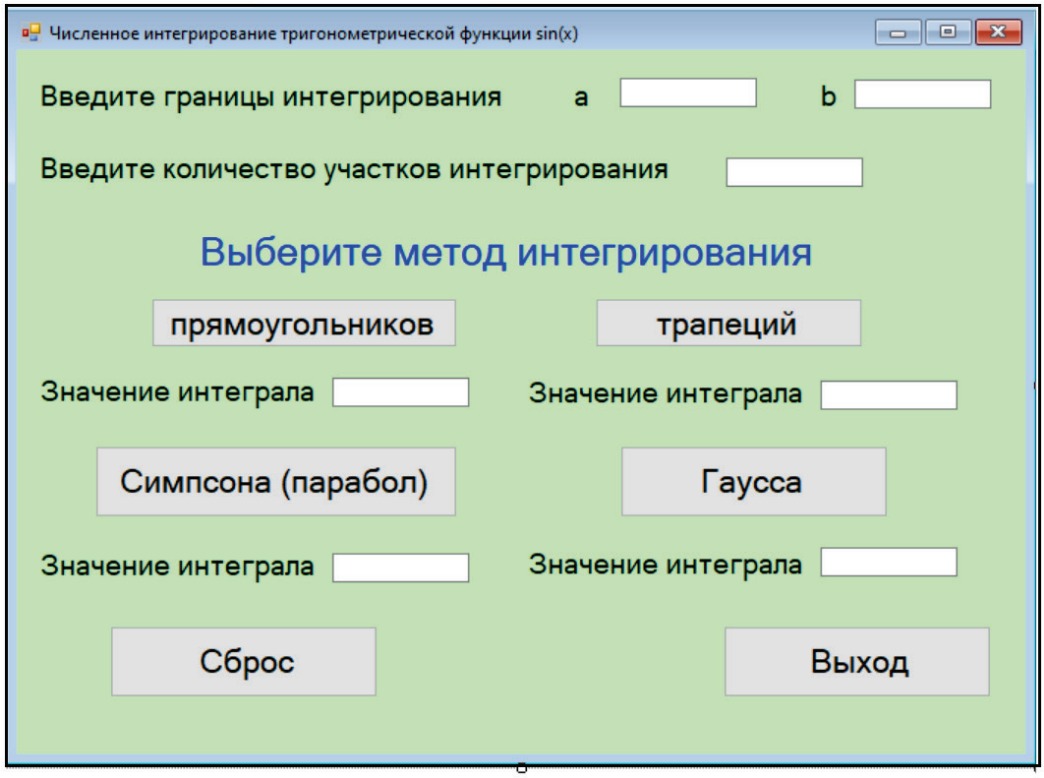

Рисунок 4.12 – Графический пользовательский интерфейс приложения «Численное интегрирование тригонометрической функции»

Третья и последняя группа объектов содержит две экранные кнопки, которым система присвоила имена button5 и button6. На кнопке button5 помещаем надпись «Выход». Щелчок мышью на этой кнопке закрывает окно приложения и завершает его работу. На кнопке button6 находится надпись «Сброс». Щелчок на этой кнопке очищает все имеющиеся в приложении текстовые окна (как те, в которые вводятся исходные данные программы, так и те, в которых выводятся результаты вычислений). После такой очистки текстовых окон можно вводить новые исходные данные и начинать вычисления заново. На этом работу по созданию пользовательского интерфейса приложения можно считать завершенной. Внешний вид созданного для данного приложения графического интерфейса пользователя приведен на рисунке 4.12.

4.3. Разработка программного кода приложения

После завершения создания графического интерфейса приложения можно приступать к следующему этапу работы над приложением – написанием программного кода на языке C#. Данный язык программирования является объектно-ориентированным, а это означает, что программа фактически представляет собой набор процедур (функций), описывающих реакцию объектов, имеющихся в приложении, на различные программные события. Таких объектов в разрабатываемом приложении шесть – это все экранные кнопки, а событием, на которое они должны реагировать, является щелчок на кнопке левой клавишей мыши.

Для того чтобы создать в программе новую функцию, достаточно дважды щелкнуть левой клавишей мыши на соответствующем объекте. После этого на экране среды программирования появляется окно программного кода, содержащее заготовку будущей программы на языке C# (рис. 4.13).

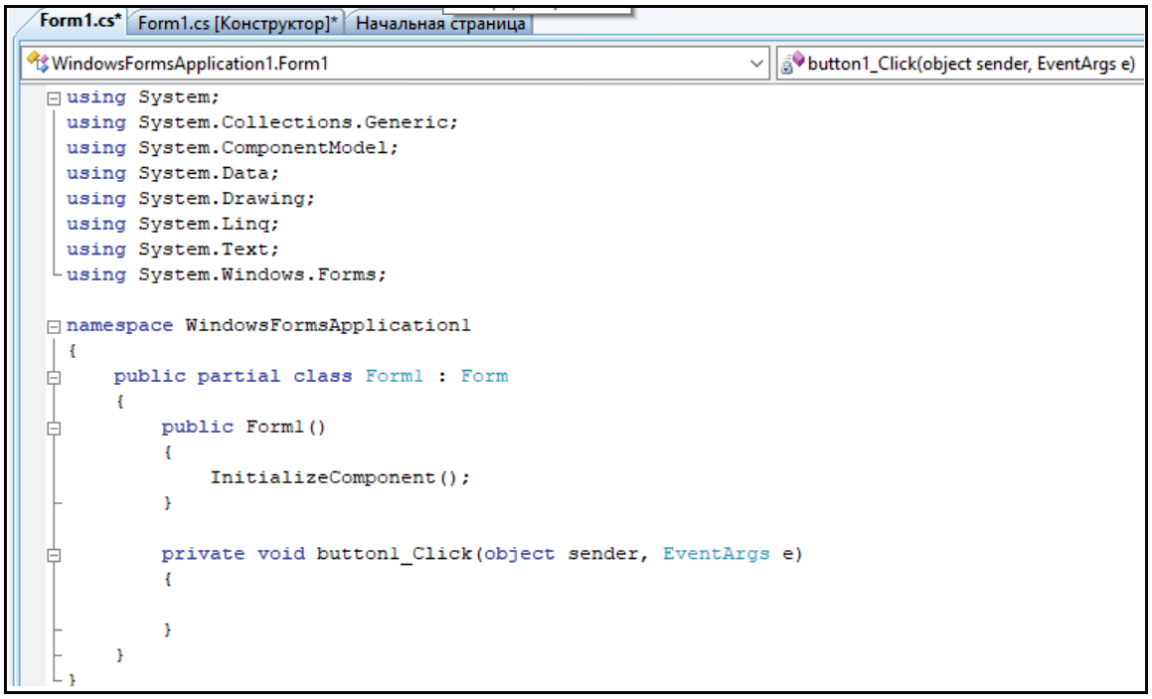

Рисунок 4.13 – Окно программного кода объектно-ориентированной среды программирования Visual C#

В дальнейшем в ходе работы над приложением понадобится неоднократно переключаться между окном кода и окном основной формы интерфейса. Для этого нужно щелкнуть правой клавишей мыши в окне кода и из открывшегося контекстного меню выбрать команду «Открыть в конструкторе». Если необходимо снова переключиться в обратном направлении, то нужно щелкнуть правой клавишей мыши на форме и в контекстном меню найти команду «Перейти к коду».

Вид окна кода (рис. 4.13) наглядно демонстрирует важное достоинство среды объектно-ориентированного и визуального программирования. Это достоинство заключается в том, что значительную часть программного кода система программирования генерирует автоматически и программисту не нужно заниматься написанием этой части. Так система самостоятельно подключает все программные модули, которые нужны для работы приложения, создает заголовок и общую структуру программы, а также создает заготовку для той функции, которая описывает реакцию объекта на программное событие, которым в данном случае является щелчок левой клавишей мыши.

Начнем работы над программным кодом с создания функции для вычисления интеграла методом прямоугольников. Эта функция должна выполняться по щелчку на экранной кнопке button1. Заготовка для данной функции будет выглядеть следующим образом:

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

Как видно, здесь уже имеется заголовок функции button1_Click, автоматически сгенерированный системой, и пара фигурных скобок, ограничивающих тело функции. Далее заполнять это тело функции программист должен самостоятельно. Первое, что необходимо сделать в начале тела функции – это описать используемые в ней локальные переменные. Для данной функции это целочисленные переменные *i* и *n*, а также вещественные переменные *a, b, h*, *sum, s* и *x*.

Данное описание выглядит следующим образом:

```
int i, n;
  double a, b, h, sum, s, x;
```

Затем нужно обеспечить правильный ввод исходных данных. Значение начала отрезка интегрирования *a* и конца отрезка *b* вводятся из текстовых окон textbox1 и textbox2, а количество участков *n* – из окна textbox3. Но необходимо учитывать, что данные вводимые в окна, являются строковыми величинами, а для расчетов нам необходимы числовые значения. Поэтому необходимо выполнить преобразование строк в числа, которое производится с помощью метода Parse. Синтаксис этого метода следующий: в начале указывается числовой тип, к которому нужно преобразовать строковое значение, затем после точки пишется ключевое слово Parse, а далее в круглых скобках – преобразуемая строка. Результат, возвращаемый методом, можно присваивать целочисленной или вещественной переменной. В нашей программе это преобразование и присваивание значений переменным производится следующими операторами:

```
a = double.Parse(textBox1.Text);
b = double.Parse(textBox2.Text);
n = int.Parse(textBox3.Text);
```

Далее производится вычисление искомой величины *s* (приближенного значения определенного интеграла) методом прямоугольников в соответствии с алгоритмом, описанным в разделе 4.2. Для этого вычисляется длина участка интегрирования *h=(b-a)/n*, а затем в цикле с заранее известным числом повторений находится *sum* – сумма всех значений подынтегральной функции в середине участков, на которые разбит диапазон интегрирования. Умножив эту сумму на величину *h*, находим искомое приближенное значение интеграла *s*.

Последнее, что осталось сделать в данной функции – это вывести полученный результат (значение переменной *s*) в текстовое окно textBox4. Это означает, что нужно выполнить действие обратное тому, что было произведено в начале функции – преобразовать численную величину в строковую. Для этого можно воспользоваться имеющейся в языке C# классом String. Для этого нужно указать название класса, а затем, после разделительной точки, название метода Format, который и преобразует числовую величину в строковую с указанием

формата вывода. Полученное значение присваиваем свойству Text объекта textBox4, а в целом оператор вывода будет выглядеть так:

```
textBox4.Text = String.Format("{0:F8}",s);
```

Далее приводится весь программный код функции button1_Click.

```
// метод прямоугольников
        private void button1_Click(object sender,
EventArgs e)
        {
          int i, n;
          double a, b, h, sum, s, x;
          a = double.Parse(textBox1.Text);
          b = double.Parse(textBox2.Text);
          n = int.Parse(textBox3.Text);
          h = (b - a) / n;
          sum = (Math.Sin(a) + Math.Sin(b)) / 2;
          for (i = 1; i <= n; i++)
          {
            x = a + h * i;
            sum = sum + Math.Sin(x);
          }
          s = h * sum;
         textBox4.Text = String.Format("{0:F8}",s);
        }
```

Следующая функция button2Click, которая выполняется после щелчка на кнопке button2, осуществляет вычисление определенного интеграла методом трапеций. Описание переменных и ввод исходных данных производится так же, как в функции button1Click.

Что же касается алгоритма вычисления, то он состоит в следующем. В начале определяется сумма промежуточных значений функции *y*, то есть сумма всех значений функции в точках x_i кроме первого и последнего. Эта величина обозначается в программе *yp*. Начальное ее значение берется равным нулю, а итоговое значение этой величины вычисляется в цикле с заданным числом повторений. В теле цикла определяется значение аргумента в каждой из точек x_i. Это значение вычисляется путем прибавления к предыдущему значению аргумента величины *h* (начальное значение аргумента берется равным *a*). Затем вычисляется соответствующее аргументу значение функции, и это значение прибавляется к величине *yp*.

По завершении работы цикла вычисляются величины *yn* и *yk* – значения функции в начальной и конечной точках отрезка интегрирования. Полусумма этих величин прибавляется к *yp* и полученная сумма умножается на *h*. Таким образом, мы получаем по формуле трапеций искомую величину *s* – значение определенного интеграла для функции *sin(x)* на заданном пользователем отрезке. Остается только вывести полученный результат в текстовое окно textbox5 аналогично тому, как это было сделано в предыдущей функции. Далее приводится весь программный код функции button2_Click.

```
    // метод трапеций
        private void button2_Click(object sender,
EventArgs e)
        {
          int i, n;
          double a, b, h, s, x, yp, yn, yk;

          a = double.Parse(textBox1.Text);
          b = double.Parse(textBox2.Text);
          n = int.Parse(textBox3.Text);
          h = (b - a) / n;
          yp = 0;
          x = a;
          for (i = 1; i <= (n - 1); i++)
          {
            x = x + h;
            yp = yp + Math.Sin(x);
          }
          yn = Math.Sin(a);
          yk = Math.Sin(b);
          s = ((yk + yn) / 2 + yp) * h;
          textBox5.Text = string.Format("{0:F8}",s);
        }
```

С экранной кнопкой button3 связана функция, которая вычисляет приближенное значение определенного интеграла методом парабол (Симпсона). Здесь применен следующий алгоритм. После ввода исходных данных производятся начальные присваивания. Аргументу x присваивается значение начальной точки отрезка a. Переменной s, которая представляет собой вычисляемое в программе значение интеграла, будет присвоено начальное значение 0. Переменной c присваивается начальное значение 1.

Затем в программе вычисляется значение суммы:

$$\sum_{i=1}^{2n-1}(3+c_i)\,y_i\,.$$

Для этого используется цикл с заранее заданным числом повторений. Начальное значение переменной цикла i равно единице, а конечное значение равно $2n$-1. В теле цикла производятся следующие действия. Текущее значение аргумента увеличивается на h, к текущему значение переменной s прибавляется новая величина согласно приведенной выше формуле, текущее значение переменной c изменяется на противоположное. После завершения работы цикла находим величину $s1$ – сумму начального и конечного значений подынтегральной функции. Теперь осталось только подставить вычисленные величины в формулу Симпсона и получить искомое значение, которое нужно вывести в текстовое окно textbox6. Далее приводится программный код функции button3_Click.

```
    // метод Симпсона (парабол)
        private void button3_Click(object sender,
EventArgs e)
        {
          int i, n, c;
          double a, b, h, s, s1, x;
          a = double.Parse(textBox1.Text);
          b = double.Parse(textBox2.Text);
          n = int.Parse(textBox3.Text);
          x=a;
          c=1;
          s=0;
          h = (b - a) / n;
          for (i = 1; i <= (n - 1); i++)
          {
            x = x + h;
            s = s + (3 + c) * Math.Sin(x);
            c=-c;
          }
         s1 = Math.Sin(a) + Math.Sin(b);
         s = h * (s1 + s) / 3;
         textBox6.Text = String.Format("{0:F8}", s);
        }
```

С экранной кнопкой button4 связана функция, которая вычисляет значение определенного интеграла по методу Гаусса. Отличие данной функции от предыдущих заключается в том, что в функции button4_Click производится только ввод исходных данных для интегрирования, обращение к вспомогательной функции fgauss, которая собственно и производит все необходимые вычисления. Функция fgauss использует вычислительный алгоритм, который подробно описан в разделе 4.2.

При вызове из основной функции button4_click вспомогательной функции fgauss в эту вспомогательную функцию вместо формального параметра f передается фактический параметр fsinus. Это название еще одной вспомогательной функции, которая, в свою очередь, возвращает значение подынтегральной функции *sin(x)*. Таким образом в этой части программы помимо основной функции задействованы еще две дополнительных. Далее приводится текст рассмотренного выше фрагмента программы.

```
    // вспомогательная функция для метода Гаусса
        public static double fgauss(Function f, double a,
double b, int n)
        {
          // Число точек в квадратуре - 5
          //   весовые коэффициентыdouble[]   w   =   {
0.236926885,   0.478628670,   0.568888889,   0.478628670,
0.236926885 };
```

```
          // значения аргументаdouble[] x = { -
0.906179846, -0.538469310, 0.000000000, +0.538469310,
+0.906179846 };
          double sum = 0.0;
          for (int i = 0; i < n; i++)
          {
    sum = sum +0.5 * (b – a) * w[i] * f(0.5 * (a + b) +
0.5 * (b – a) * x[i]);
          }
          return sum;
        }
        public delegate double Function(double x);

        // Интегрируемая функция y = sin(x)
        static double fsinus(double x)
        {
          return Math.Sin(x);
        }

    // основная функция для метода Гаусса
        private void button4_Click(object sender,
EventArgs e)
        {
         double a, b, s;
         a = double.Parse(textBox1.Text);
         b = double.Parse(textBox2.Text);
         s=fgauss(fsinus,a,b,5);
         textBox7.Text = String.Format("{0:F8}", s);
        }
```

С экранной кнопкой button5 связана функция, которая закрывает основную экранную форму Form1 и все приложение в целом. Эта функция button5_Click является самой простой во всем приложении, так как её тело состоит из единственного оператора Close, который и выполняет вышеописанное действие.

Экранная кнопка button6 вызывает функцию, которая очищает все текстовые окна, имеющиеся в приложении (как с входными данными, так с результатами), чтобы пользователь имел возможность заново начать работу с приложением «с чистого листа». Тело функции button6_Click состоит из группы однородных операторов, которые для всех окон их свойству Text присваивают «пустое» значение:

```
    textBox1.Text = "";
          textBox2.Text = "";
          textBox3.Text = "";
          textBox4.Text = "";
          textBox5.Text = "";
          textBox6.Text = "";
          textBox7.Text = "";
```

Ниже приводится полный текст программного кода приложения, разработанного для данного курсового проекта.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
namespace WindowsFormsApplication1
{
  public partial class Form1 : Form
  {
    public Form1()
    {
      InitializeComponent();
    }
// метод прямоугольников
        private void button1_Click(object sender,
EventArgs e)
    {
      int i, n;
      double a, b, h, sum, s, x;
      a = double.Parse(textBox1.Text);
      b = double.Parse(textBox2.Text);
      n = int.Parse(textBox3.Text);
      h = (b - a) / n;
      sum = (Math.Sin(a) + Math.Sin(b)) / 2;
      for (i = 1; i <= n; i++)
      {
        x = a + h * i;
        sum = sum + Math.Sin(x);
      }
      s = h * sum;
     textBox4.Text = String.Format("{0:F8}",s);
    }

// метод трапеций
      private void button2_Click(object sender,
EventArgs e)
    {
      int i, n;
      double a, b, h, s, x, yp, yn, yk;

      a = double.Parse(textBox1.Text);
      b = double.Parse(textBox2.Text);
      n = int.Parse(textBox3.Text);
```

```
            h = (b - a) / n;
            yp = 0;
            x = a;
            for (i = 1; i <= (n - 1); i++)
            {
              x = x + h;
              yp = yp + Math.Sin(x);
            }
            yn = Math.Sin(a);
            yk = Math.Sin(b);
            s = ((yk + yn) / 2 + yp) * h;
            textBox5.Text = string.Format("{0:F8}",s);
          }

    // метод Симпсона (парабол)
         private   void   button3_Click(object   sender,
EventArgs e)
          {
            int i, n, c;
            double a, b, h, s, s1, x;
            a = double.Parse(textBox1.Text);
            b = double.Parse(textBox2.Text);
            n = int.Parse(textBox3.Text);
            x=a;
            c=1;
            s=0;
            h = (b - a) / n;
            for (i = 1; i <= (n - 1); i++)
            {
              x = x + h;
              s = s + (3 + c) * Math.Sin(x);
              c=-c;
            }
           s1 = Math.Sin(a) + Math.Sin(b);
           s = h * (s1 + s) / 3;
           textBox6.Text = String.Format("{0:F8}", s);
          }

    // вспомогательная функция для метода Гаусса
         public static double fgauss(Function f, double a,
double b, int n)
          {
            // Число точек в квадратуре - 5
            //   весовые   коэффициентыdouble[]   w   =   {
0.236926885,   0.478628670,   0.568888889,   0.478628670,
0.236926885 };
```

```
          //  значения  аргументаdouble[]  x  =  {  -
0.906179846,  -0.538469310,  0.000000000,  +0.538469310,
+0.906179846 };
          double sum = 0.0;
          for (int i = 0; i < n; i++)
          {
    sum = sum +0.5 * (b - a) * w[i] * f(0.5 * (a + b) +
0.5 * (b - a) * x[i]);
          }
          return sum;
        }
        public delegate double Function(double x);

        // Интегрируемая функция y = sin(x)
        static double fsinus(double x)
        {
          return Math.Sin(x);
        }

    // основная функция для метода Гаусса
        private   void   button4_Click(object   sender,
EventArgs e)
        {
         double a, b, s;
         a = double.Parse(textBox1.Text);
         b = double.Parse(textBox2.Text);
         s=fgauss(fsinus,a,b,5);
         textBox7.Text = String.Format("{0:F8}", s);
        }

    // закрытие экранной формы и приложения
     private void button5_Click(object sender, EventArgs
e)
        {
          Close();
        }

    // очистка всех окон приложения
       private   void   button6_Click(object   sender,
EventArgs e)
        {
          textBox1.Text = "";
          textBox2.Text = "";
          textBox3.Text = "";
          textBox4.Text = "";
          textBox5.Text = "";
          textBox6.Text = "";
          textBox7.Text = "";
```

```
        }
    }
```

После завершения работы над программным кодом можно запустить созданное приложение на выполнение. Для этого следует либо на панели инструментов среды программирования щелкнуть мышью экранную кнопку «Начать отладку» (она имеет вид зеленой треугольной стрелки), либо на клавиатуре компьютера нажать функциональную клавишу F5. После запуска программы любым из указанных способов приложение переходит в рабочий режим. В этом режим можно вводить в окна приложения различные исходные данные и протестировать его работоспособность. На рис. 14.4 показана работа приложения для границ интегрирования от 0 до π.

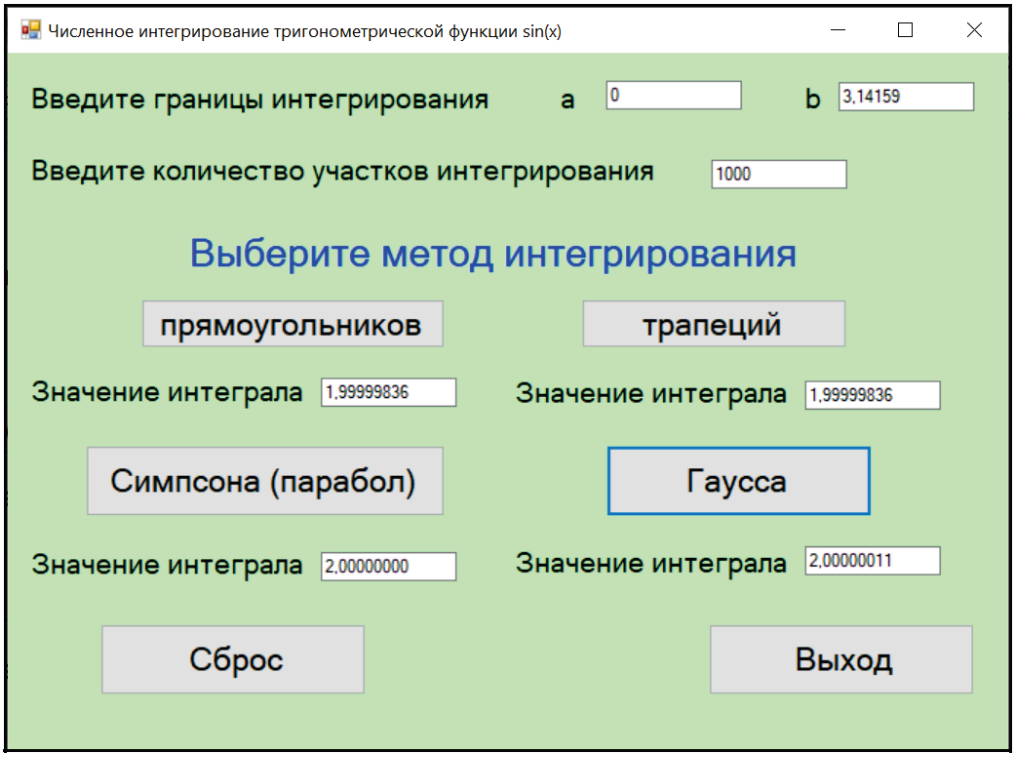

Рисунок 4.14 – Приложение «Численное интегрирование тригонометрической функции» в рабочем режиме

Результаты работы приложения показывают, что для данной тригонометрической функции все используемые методы дают хорошие результаты, но наивысшую точность вычислений обеспечивает метод Симпсона.

Таким образом, поставленная в курсовой работе задача выполнена, то есть создано работоспособное приложение, выполняющее численное интегрирование различными методами.

ЗАКЛЮЧЕНИЕ

В данной работе были сформулированы методические указания для разработки курсовой работы по дисциплине «Визуальные среды программирования».

Была определена цель выполнения курсовой работы и рассмотрены основные этапы ее разработки, определена структура курсовой работы, приведены основные требования к ее оформлению, описана процедура защиты курсовой работы и указаны основные критерии ее оценки. Также в работе имеется список тем, из которого студенты могут выбрать ту, которую в дальнейшем они должны раскрыть в своей курсовой работе. Приведен пример выполнения курсовой работы по одной из перечисленных в списке тем.

Создание курсовой работы, приобретенные при этом теоретические знания и практические навыки, опыт защиты курсовой работы должны стать для студентов важным и необходимым этапом для подготовки к разработке и защите выпускной квалификационной работы.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Алексеев, Ю. Е. Введение в информационные технологии и программирование на языке С в среде VS C++. Модуль 1 дисциплины «Информатика»: учебное пособие / Алексеев Ю. Е. – М.: Московский государственный технический университет имени Н. Э. Баумана, 2018. – 100 с. – ISBN 978-5-7038-4891-3. – Текст: непосредственный.
2. Борисенко, В. В. Основы программирования / В. В. Борисенко. – М.: Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. – 323 с. – ISBN 978-5-4497-0678-2. – URL: http://www.iprbookshop.ru/52206 (дата обращения 10.01.2024). – Режим доступа: ЭБС «IPRbooks». – Текст: электронный.
3. Визуальное программирование на основе библиотеки MFC : методические указания / сост. А. Я. Лахов, Р. Е. Борщиков. – Саратов: Вузовское образование, 2016. – 57 с. – URL: http://www.iprbookshop.ru/28324 (дата обращения 10.01.2024). – Режим доступа: ЭБС «IPRbooks». – Текст: электронный.
4. Горелов, С. В. Современные технологии программирования: разработка Windows-приложений на языке C#. В 2 томах. Т. I : учебник / С. В. Горелов. – М.: Прометей, 2019. – 362 с. – ISBN 978-5-907100-09-1. – Текст: непосредственный.
5. Горелов, С. В. Современные технологии программирования: разработка Windows-приложений на языке C#. В 2 томах. Т. II : учебник / С. В. Горелов. – М.: Прометей, 2019. – 378 с. – ISBN 978-5-907100-18-3. – Текст: непосредственный.
6. Достовалов, Д. Н. Объектно-ориентированный анализ и проектирование. Задачи и примеры на C++ : учебное пособие / Д. Н. Достовалов, О. В. Лауферман. – Новосибирск : Новосибирский государственный технический университет, 2022. – 74 с. – ISBN 978-5-7782-4708-6. – Текст: непосредственный.
7. Информационная система «Единое окно доступа к образовательным ресурсам» : [сайт]. – 2022. – URL: http://window.edu.ru (дата обращения 10.01.2024). – Текст: электронный.
8. Кивран, В. К. Программирование в среде Visual C++ 6 [Электронный ресурс]: учебное пособие/ Кивран В. К. – Самара: Самарский государственный архитектурно-строительный университет, ЭБС АСВ, 2014. – 118 с. – ISBN 978-5-9585-0601-9. – URL: http://www.iprbookshop.ru/43185 (дата обращения 10.01.2024). – Режим доступа: ЭБС «IPRbooks». – Текст: электронный.
9. Мартыненко, Т. В. Основы визуального программирования в среде Visual Studio на базе C# : учебное пособие / Т. В. Мартыненко, В. В. Турупалов, Н. К. Андриевская ; под редакцией В. В. Турупалова. – М., Вологда : Инфра-Инженерия, 2023. – 232 с. – ISBN 978-5-9729-1225-4. – Текст: непосредственный.

10. Оформление текстовой части курсовой работы и курсового проекта. Краткая выписка из ГОСТ 7.32-2017 «Отчет о научно- исследовательской работе. Структура и правила оформления»: методические рекомендации / сост.: М. Д. Баранова, А. Ю. Котова. – СПб.: ВШТЭ СПбГУПТД, 2023. – 20 с. – URL: https://nizrp.narod.ru/recomedation.pdf (дата обращения 10.01.2024). — Текст: электронный.

11. Пестриков, В. М. Решение математических задач в Turbo Pascal : учебное пособие / В. М. Пестриков, А. Н. Маслобоев. – СПб.: ГОУВПО СПб ГТУ РП, 2009. – 110с. – Текст: непосредственный.

12. Подготовка, оформление и защита курсовой работы : методические указания / сост. В. П. Яковлев, П. Е. Антонюк. – СПб.: СПбГТУРП, 2015. – 32 с. – Текст: непосредственный.

13. Самохвалов, Э. Н. Введение в проектирование и разработку приложений на языке программирования C# : учебное пособие / Э. Н. Самохвалов, Г. И. Ревунков, Ю. Е. Гапанюк. – М.: Московский государственный технический университет имени Н. Э. Баумана, 2018. – 248 с. – Текст: непосредственный.

ПРИЛОЖЕНИЕ 1. Образец титульного листа курсовой работы

Министерство науки и высшего образования Российской Федерации
федеральное государственное бюджетное образовательное учреждение высшего образования
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ПРОМЫШЛЕННЫХ ТЕХНОЛОГИЙ И ДИЗАЙНА»
ВЫСШАЯ ШКОЛА ТЕХНОЛОГИИ И ЭНЕРГЕТИКИ
Институт Энергетики и автоматизации
Кафедра прикладной математики и информатики

КУРСОВАЯ РАБОТА

по дисциплине «Визуальные среды программирования»

на тему:

Выполнил студент учебной группы №

_____
(фамилия, имя, отчество)

Проверил _____
(должность, фамилия, имя, отчество)

Санкт-Петербург
2024

ПРИЛОЖЕНИЕ 2. Образец задания на курсовую работу

Министерство науки и высшего образования Российской Федерации
федеральное государственное бюджетное образовательное учреждение высшего образования
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ПРОМЫШЛЕННЫХ ТЕХНОЛОГИЙ И ДИЗАЙНА»
ВЫСШАЯ ШКОЛА ТЕХНОЛОГИИ И ЭНЕРГЕТИКИ
Институт Энергетики и автоматизации
Кафедра прикладной математики и информатики

ЗАДАНИЕ НА КУРСОВУЮ РАБОТУ
по дисциплине «Визуальные среды программирования»

Студенту_____ группа_____
Тема проекта_____

Содержание пояснительной записки

Реферат.
Введение.
1. Название раздела
2. Название раздела.
3. Название раздела.
Заключение.
Список использованных источников
Приложение. Название приложения

Исходные данные

Руководитель _____ _____
(должность/ звание, ученая степень, Ф.И.О. *(подпись)*

Задание на курсовую работу выдано «__» _____202_ г.
Срок предоставления курсовой работы к защите «__» _____202_ г.

Исполнитель _____ _____
Ф.И.О. *(подпись)*