СОДЕРЖАНИЕ

1. [РЕФЕРАТ 2](#_Toc514285715)

1.1. [Общее число страниц, количество таблиц, рисунков, количество разделов 2](#_Toc514285716)

1.2. [Краткий обзор содержания разделов расчетно-пояснительной записки 2](#_Toc514285717)

2. [ОБЩАЯ ХАРАКТЕРИСТИКА КР 3](#_Toc514285718)

3. [ТЕОРЕТИЧЕСКОЕ ВВЕДЕНИЕ (раздел 1) 4](#_Toc514285719)

3.1. [ООП 4](#_Toc514285720)

3.2. [Описание UML 6](#_Toc514285721)

3.3. [Подходы к сбору и анализу требований к ПО 8](#_Toc514285722)

4. [ПРОЕКТНАЯ ЧАСТЬ (раздел 2) 12](#_Toc514285723)

5. [ЭКСПЕРИМЕНТАЛЬНАЯ ЧАСТЬ (раздел 3) 18](#_Toc514285724)

5.1. [Тестирование реализованной программной системы 23](#_Toc514285725)

6. [ЗАКЛЮЧЕНИЕ 27](#_Toc514285726)

7. [СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ 28](#_Toc514285727)

8. [ПРИЛОЖЕНИЕ 29](#_Toc514285728)

[Intergal.h 29](#_Toc514285729)

[MyForm.cpp 31](#_Toc514285730)

[MyForm.h 31](#_Toc514285731)

РЕФЕРАТ

**Общее число страниц, количество таблиц, рисунков, количество разделов**

Общее число страниц: 41

Количество таблиц: 3

Рисунков: 12

Количество разделов: 3

**Краткий обзор содержания разделов расчетно-пояснительной записки**

Язык программирования C++ является одним из самых популярных и широко используемых для разработки GUI приложений. В данной работе нам предстоит разработать несколько классов, логику их взаимодействия, разработать приложение с графическим интерфейсом, работать c различными функциями стандартных библиотек С++. Объем работы обеспечит закрепление и углубление практических навыков по программированию на языке С++.

ОБЩАЯ ХАРАКТЕРИСТИКА КР

Тема, выбранная для курсовой работы, является очень актуальной. Такая математическая операция как интегральное исчисление является очень важной во многих сферах, такие как физика, математика, и так далее, поэтому программа, которая получает значения определенного интеграла, актуальна. Это сэкономит время вычислений. Для того чтобы решить данную задачу, понадобится разделить ее на несколько разделов:

Теоретическая часть. В ней будут рассмотрены основы ООП, чтобы понять, как будет работать программа, и как в данном случае стоит воспринимать задачу. Также после этого будет разобрано, что такое UML, и для чего оно используется, и будут рассмотрены подходы к сбору и анализу требований, так как, безусловно, нужно знать взаимодействие будущей программы, но как подойти к тому что она будет делать? Для этого нужно обязательно проанализировать данные требования, поэтому этот пункт тоже важен. Затем идет проектная часть, где уже будет обсуждено определения интеграла, его численные методы, и как они вычисляются, и так же уже примерно я соединю всю информацию, которая была собрана и составлю макет программы. Следующий раздел это экспериментальная часть, где все будет описано детальнее, и будет реализовываться сама программа на языке C++. После этого пойдет этап тестирования, где будет проверено работоспособность программы, и после чего можно будет сделать выводы о проделанной работы.

ТЕОРЕТИЧЕСКОЕ ВВЕДЕНИЕ

**ООП**

Объектно-ориентированное программирование (ООП) – это методология программирования, основная на представлении программы как совокупность объектов. Каждый объект представляет собой экземпляром класса. А классы составляют иерархию наследования. Суть ООП составляет в том, что оно воспринимает программирование как абстракцию информационных объектов. Таким образом, оно решает задачу Структурного программирования на новом уровне, а именно структуризацию информации в плане управляемости. Также управляемость системы подразумевает минимизацию данных, то есть удаление избыточных данных. Что делает программу более чистую и понятную для понимания. Также у ООП есть свои собственные важные свойства:

- Важна абстракция предмета. Составление конкретной задачи над ним, само понимания предмета, и составление его в виде класса.

-Инкапсуляция. То есть составление системы в котором находятся и данные и методы, совершающие над ними. Данное свойство делает иерархическую систему очень быстрой, так как есть понимания того, что нужно делать, без дополнительный нюансов: как это будет сделано и так далее.

- Наследование. То есть отношение между системами как родитель-дочь. Есть основной класс, от которого идет дерево классов, которые имеют все данные и методы, находящиеся в основном классе, но также имеют свои данные и методы. Это свойство очень ускоряет взаимодействие между объектами, так как данные и методы не дублируются, а ссылаются, что делает процесс для компьютера менее затратным в плане времени и памяти.

-Полиморфизм для определения точки, в которой единое управление лучше распараллелить или наоборот — собрать воедино. Другими словами, полиморфизм указывает, какую версию метода текущего объекта запустить. Допустим, есть несколько наследуемых классов, с разной реализацией одного и того же метода. То есть, имеется несколько разных методов с одинаковым именем, реализованных в разных классах. Какую версию метода вызвать? Это зависит от типа переменной, в которой находится объект. Еще с помощью полиморфизма можно поместить объекты разных классов в один массив с типом базового класса.

То есть по сути идет речь о организации информации нужная/ненужная, одно/множество, родительское/дочерние, ключевое/подробности.

Как уже было сказано ООП представляет собой абстрактную систему взаимодействии объектов. Объект – представляет собой сущность, которое обрабатывает сообщения и реагирует на них. Объект – это экземпляр Класса. Сам класс – это универсальный комплексный тип данных состоящий из полей (переменные) и методов (функций), также может иметь конструктор (объявление всех переменных в классе, при запуске программы) и деструктор (уничтожение всех переменных в классе при завершении программы), но не во всех языках программирования они реализованы. Стоит сказать о важной особенности реализации: Контроль доступа. Безусловно, при создании класса, нам нужно использовать какие-то методы или переменные вне класса, какие в нем, именно для этого и был создан контроль доступа. Все, что находится в модификаторе private, доступно только внутри класса. Protected значит, что доступ есть только внутри класса и наследнику.Public означает, что методы и поля внутри него доступны открыто, то есть их можно менять вне класса и его наследников. ООП ориентируется на описании структуры, и поэтому при проектировании программа важно задать два вопроса: Из каких частей будет состоять программ? Какую функцию выполняет каждая часть? Также само ООП разделено на несколько методологий: Компонентное программирование (специальная надстройка, сводка правил и ограничений для компонентов программы. Все новые элементы могут наследовать лишь базовый интерфейс, что делает базовый класс менее хрупким. Используется для крупных проектов и продолжительным временем действия); Прототипное программирование (данная методология сохранила часть черт ООП, но полностью избавилась от понятия класса и наследования); Класс-ориентированное программирование (данная методология сфокусирована на данных. Тем более данные и поведение неразрывны между собой. Класс определяет структуру и поведение одинаковую для всех экземпляров класса. Экземпляр является носителем данных, то есть он меняется в зависимости от поведения, заданным классом. Получившийся экземпляр строго следует поведению и структуре, которую задал ей класс.).

**Описание UML**

UML – это графический язык, который описывает объектное моделирование в области разработки ПО. Используется для составления абстрактной модели системы, относительно которой потом реализуется программа. Системы, созданные на UML, используются на всех этапах жизненного цикла программного обеспечения, начиная с анализа требований и заканчивая сопровождением системы. UML, так же как и другие языки, имеет свой “словарь” и правила комбинирования “слов” для получения осмысленных конструкций. “Словарь” языка образует графические элементы. Каждому графическому символу соответствует определенная семантика, поэтому, модель, созданная одним программистом, будет понятна однозначно другим. Отсюда, к слову, следует, что модель программного обеспечения, сделанная на UML, можно перевести на объектно-ориентированный язык программирования, другими словами, при наличии опыта и хорошего инструментального средства моделирования, которое поддерживает UML, создав абстракцию, также будет получено и заготовку кода программы, которой соответствует этой модели. Также, стоит упомянуть важные термины. Сущность - -это абстракция, являющиеся основным элементом моделей. Существует 4 вида сущности: структурная (интерфейс, класс, узел, компонент, вариант использования), поведенческая (состояние, взаимодействие), группирующая (пакеты), аннотационные (комментарий). У каждого вида есть свое графическое представление. Также сущностями существуют разные связи. Зависимость – показывает связь сущностей, при которой одна сущность при изменении можно повлиять на другую сущность. Ассоциация –отношение, при котором объекты одной сущности связаны с объектами другой сущности. Обобщение - отношение между сущность-родителем и сущность-потомком (по сути, это наследование). Реализация –отношение между сущностью, определяющей спецификацию поведения (интерфейс) с сущностью, определяющей реализацию этого поведения (класс, компонент). Достоинство этого графического языка в том, что он достаточно ясно показывает все взаимосвязи и показывает работу проекта, что может сэкономить время не только программисту для реализации программы, но и помочь клиенту понять, как будет реализована программа, что поможет найти быстрее “общий язык” между разработчиком и клиентом.UML очень близок семантически к объектно-ориентированным языкам программирования. Также выделяют несколько недостатков, такие как: слишком большое количество не нужных конструкций; при составлении крупных проектов диаграмма становится очень крупной и сложной; также схемы показывают взаимодействие между компонентами программы, но не их внутреннее строение и алгоритм выполнения.

**Подходы к сбору и анализу требований к ПО**

Перед тем как создавать программу, программист, безусловно, должен знать конкретную задачу и требования к ней. И именно здесь часто возникают проблемы, так как не всегда разработчик и заказчик могут найти единое решение/требования. Поэтому для удаления такой проблемы должны быть какие-то методы, чтобы минимизировать “расплывчатость” требований и видеть эти требования к задачи насквозь.

**Совещание**

Совещание – это встреча, сфокусированная на обсуждение вопросов, которые были обсуждены и озвучены участниками заранее. На нем созываются люди с различными точками зрениями, чтобы создать требования, основываясь на разных мнениях. На них составляется список решенных, скрытых, и конфликтных требований.

**Достоинства:** Позволяет расширить и детализировать требования, выявить основные задачи

**Недостатки:** Не всегда совещания заканчиваются приходу к общему решению проблемы.

**Мозговой штурм**

Один из самых популярных вариантов получений требований. Он дает возможность собрать идеи заинтересованных лиц в кратчайшие время.

**Достоинства:** Позволяет получить необычные подходы к решению задачи и большой спектр вариантов решения задачи

**Недостатки:** Участники мозгового штурма должны быть мотивированы.

**Анкетирование**

Данный метод подразумевает собой создания опросника/тест, в котором содержатся открытые и закрытые вопросы, на которые должен ответить заказчик.

**Достоинства:**

1. Очень быстрый метод получения требований заказчика.
2. Метод требует небольшие материальные расходы.

**Недостатки:** Невозможно составить опросник, в котором имеются все вариации требований.

**Повторное использование спецификации**

Повторно использовать спецификации можно в том случае, если есть уже завершенные один или несколько подобных проектов.   
  
Техническое задание, сделанное для предыдущего проекта, может быть использовано для другого проекта с целью сократить продолжительность сбора, анализа и разработки требований, что позволит быстрее начать разработку.  
Например, ТЗ для интернет магазинов похожи друг на друга и содержат одинаковые требования.

В большинстве случаев только часть документации актуальна для нового проекта, поэтому потребуется тщательная проверка требований на соответствие текущим целям и задачам Заказчика.  
**Достоинства:** Сокращение времени на разработку документации.  
**Недостатки:**

1. Высокая стоимость первого проекта.

2. Излишняя детализация требований, может привести к их дорогостоящим изменениям в будущем.

**Представитель заказчика в компании разработчиков**

Один из самых эффективных методов, так как составляются и корректируются вовремя разработки, что позволяет сэкономить время.

**Достоинства:** Быстрая обратная связь от заказчика.

**Недостатки:** Высокая цена для заказчика. Время на адаптацию сотрудника заказчика.

**Обучение**

Суть данного метода в том, что разработчик обучается в организации заказчика. Метод полезен, когда заказчику сложно составить требования.

**Достоинства:** Позволяет понять процессы/объект для которых нужно создать ПО, что дает составить детальное решение.

**Недостатки:** Высокая длительность и стоимость.

Наилучший метод является комбинированный. То есть соединение вышеперечисленных методов. Так как, посмотрев вышеперечисленные методы, можно понять, что каждый метод имеет свои достоинства, но и также недостатки. И чтобы эти недостатки минимизировать, следует использовать разные методы, чтобы один убирал недостатки другого.

ПРОЕКТНАЯ ЧАСТЬ

Первым делом стоит понять, из чего состоит определенный интеграл, и как он считается численными методами, чтобы потом составить поля, методы класса. Существует большое количество определений интеграла, поэтому будет использовано такое определение: определенный интеграл представляет собой площадь криволинейной трапеции. То есть функция отделена двумя отрезками (верхний и нижний предел) и этот модуль площади участка равен определенному интегралу. Стоит уже обозначить из чего состоит интеграл, чтобы потом сделать из его составных частей поля. Он состоит из верхнего и нижнего предела, подынтегральной функции, и соответственно самого результата. Определенный интеграл находится разными методами, один из самых основных – это формула Ньютона-Лейбница, но в данной ситуации задачи он не будет использоваться, дело в том, что, во-первых, данная формула не является численным методом, и во-вторых ей нельзя найти неберущиеся интегралы, то есть те, у которых невозможно найти неопределенный интеграл. Именно поэтому определенный интеграл в данной задаче будет находиться численными методами. Один из недостатков, которые можно увидеть, используя их, заключается в том, что они не точные, всегда есть погрешность, и можно получиться только результат близкий к истинному. Общая идея численных методов, которые мы рассмотрим, заключается в том, что они делят функцию на небольшие кусочки и суммируют. Соответственно, чем больше разделяется на кусочки, тем больше результат близок к истинному, поэтому стоит взять переменную, которая будет отвечать за количество делений функции, пусть дальше будет указываться буквой n.

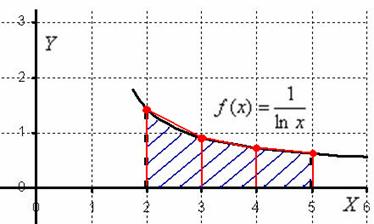


Рисунок 1. Визуальный пример метода трапеций

Также рассмотрим какие есть численные методы. Первый, прямоугольный метод, суть его в том, что он разделяет функцию на прямоугольники, и суммирует их.

Где

Небольшое пояснение: имеется ввиду, что берем i=0, что в итоге получается: и подставляем это значение в функцию, и так далее до n. Второй, метод трапеции, соответственно он делит функцию на трапеции и суммирует их.

Где

И последний, который будет рассмотрен, это метод Симпсона. Суть метода в том, что он приближает полином второй степени к функции на отрезке параболы. То есть он, иначе говоря, делит функцию на параболы и суммирует их.

Где

Методы, перечисленные выше, были расположены по возрастанию точности. То есть метод Симпсона самый точный. Говоря о точности, как уже было сказано все эти методы лишь приближены к истинному результату, поэтому должна быть какая-та погрешность. Для этого будет использовано правило Рунге.

Где для метода прямоугольников

для метода трапеций

для метода Симпсона

То есть нужно взять найти интеграл при 2n разбиений и n. Итак, разбив задачу, можно начинать составлять класс. В данном случае создается один класс в котором будет весь вышеперечисленный функционал. Итак, в полях будут переменные отвечающие за верхний и нижний предел. Поле, которое содержит введенную функцию, которую нужно интегрировать. Результат, полученный в итоге вычислений. И, конечно, количество делений на кусочки функции, то есть число итераций. И также погрешность при вычислении. Итак, далее идут методы, здесь, во-первых, создадим для каждого численного метода свою функцию. То есть метод прямоугольника, метод трапеции, и Симпсона. Также будет сделано отдельно вычисление погрешности по правилу Рунге. И в итоге получаем функционал нашего класса “Интеграл”. Но были уточнены только расчеты, поставленные для задачи, но не ввод или вывод значений. Для этого будет создан класс Интерфейс, где будет возможность вводить и выводить нужные значения. Но перед этим стоит разобрать, какой интерфейс лучше всего подойдет. В наше время появилось много вариантов созданий интерфейса, от классического, когда мы все делаем сами, до VR-интерфейса. Интерфейс даже можно создать на средах, которые, вроде бы, для этого не созданы. Например, игровой движок Unity, который предоставляет удобные UI инструменты и методы. Также стоит отметить, что в настоящее время создания удобного интерфейса для пользователя стало одним из важным аспектом программ, что нельзя сказать о прошлом, когда насчет интерфейса сильно не заморачивались. Одна из причин такой важности хорошего интерфейса – это массовость ПК и мобильных устройств, а это значит, что при создании обеспечения часто нужно учитывать, что неспециализированному пользователю будет сложно разбираться в непонятном для него интерфейсе, что может снизить популярность и актуальность программы. Программа для курсовой работы не будет иметь сложную систему выбора параметров и настроек, в ней будет всего лишь несколько кнопок ввода и вывода, поэтому для этого вполне подойдет Windows Forms, так как в нем реализации такого просто интерфейса будет очень легкой. Стоит сразу определить, что будет выводится и что будет вводится. Безусловно, функция, которую нужно интегрировать, будет вводится, также ее верхний и нижний предел. Количество итераций тоже будет вводится. И выводится у нас будет результат вычислений и его погрешность. Также стоит отметить, что пользователю надо будет выбрать метод, которым программа будет решать. Разобравшись с этим, нужно понять из чего состоит класс интерфейса. Поле у него будет всего одно, а именно конструктор класса Интеграл. Вызвав его, класс интерфейса получит доступ ко всем методам и полям, которые были разобраны выше. Теперь стоит разобрать то, как будут устроены методы. Пользователь должен будет выбирать каким методом ему считать, поэтому стоит создать 3 метода, которые начинают вычисление выбранным методом. Пусть выбор будет сделан из 3 кнопок, то есть нажав нужную кнопку, начинается вычисление выбранным методом. Внутри их будут запросы на ввод функции, верхнего, и нижнего предела, а также число итераций. В Windows Forms это можно сделать с помощью простого поля текста, куда можно будет ввести значения. То есть понадобится 4 таких поля, а также внутри метода присвоить этим переменным значения введенных в текстовых полях. В итоге вычислений, в специальном поле будет выведено значение и погрешность интеграла.

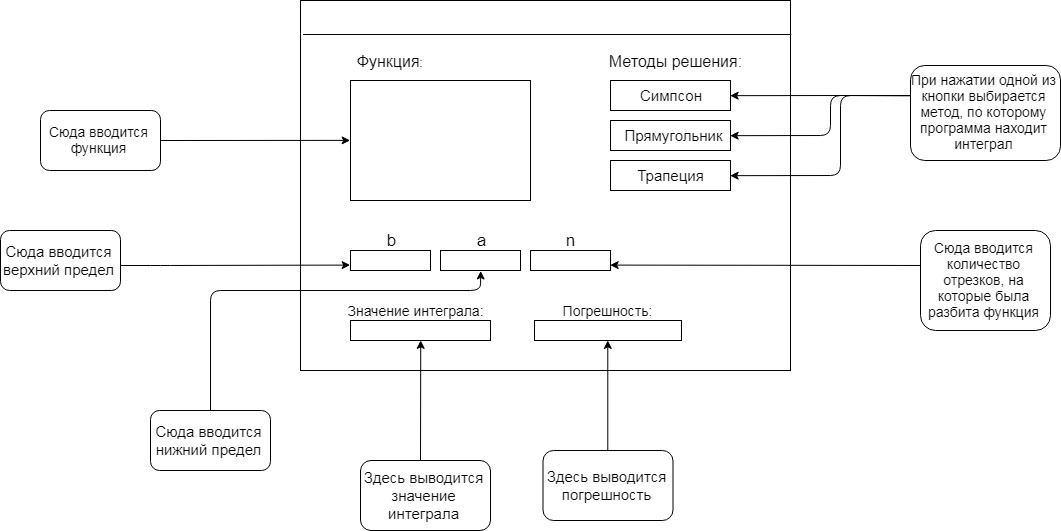


Рисунок 2. Макет интерфейса

Таким образом, было выяснено какие классы будут использованы в программе, и также какой у них будет функционал. Но также стоит уточнить их взаимодействие. Из вышеперечисленного, можно сделать вывод, что нужно связать класс интерфейса и интеграла, и данная связь не будет являться родительской, а просто объявление класса интеграла в классе интерфейсе, чтобы интерфейс имел доступ к функционалу класса Интеграл.

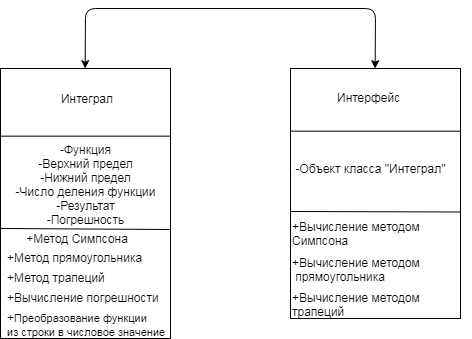


Рисунок 3. Диаграмма классов

ЭКСПЕРИМЕНТАЛЬНАЯ ЧАСТЬ

Было рассмотрено только примерно, как будет устроена программа, поэтому следует ее разобрать детальнее. Итак, стоит начать с полей класса Интеграла. Подынтегральная функция должна иметь тип данных string, так как функции могут иметь не только число, но и также sin, cos, log, и поэтому численные типы данных не подойдут. Но даже так, введя функцию, программа не сможет воспринимать введенную строку как функцию, в которую нужно подставить значения переменной x. То есть надо как-то перевести string, в, например, double. Для этого будет использоваться библиотеку C++ Mathematical Expression Toolkit Library. Итак, первым делом, нужно написать метод, который переводит введенную функцию из string в double. Пусть будет называться vvod, она ничего возвращать не будет, поэтому она будет void. Перед объяснением, стоит сказать, что библиотека является очень крупной, и полностью ее внутреннее устройство объяснить сложно, поэтому некоторые объяснения будут “поверхностными”. Для начало надо создать переменную класса symbol\_table\_t, данный класс помогает распознавать математические функции, и аргументы, в которые надо будет подставить числа. Symbol\_table\_t.add\_variable(“x”,x) – находит в строке аргумент x, и дает программе указания, что x нужно воспринимать как переменную численного типа данных. Symbol\_table\_t.add\_constants() – воспринимает все математические функции (sin,log, и так далее) в строке. Далее надо создать переменную класса parser\_t parser, и пишем команду parser.compile(введенная функция, expression\_t name), которая переводит строку в функцию. Разобравшись с подынтегральной функцией, рассмотрим остальные поля. Пусть верхний предел будет называться b, а нижний а. Они оба будут типа double, так как это самый обширный тип данных, и также не имеет смысла делать их целочисленные, так как иначе пропадет погрешность, и результат будет округлен и не будет равен тому, что было подсчитано. Вполне логично, сам результат будет тоже double, пусть называется res. Количество итераций будет целочисленное положительное число, поэтому его тип данных будет unsigned int, пусть будет называться n. Погрешность будет double, и назовем ее tochnost. Остается еще одно поле, которое обязательно нужно упомянуть. Дело в том, что правило Рунге в зависимости от выбранного метода, считается по-разному. То есть коэффициент при выборе метода Симпсона будет равен 1/15, а для прямоугольника 1, для трапеции 1/3. То есть, нужно сделать так чтобы, когда пользователь выбирал выбранный метод, то и программа считала погрешность с нужным коэффициентом, поэтому нужна переменная, которая ставится внутри каждого метода. При выборе численного метода этой переменной присваивается число, и в методе вычисления погрешности она выбирает подсчет по правилу Рунге тот, которому соответствует числу этой переменной, назовем ее checkmethod. Теперь рассмотрим наши методы, vvod переводит строку, метод для вычислений Симпсоном, вычисление прямоугольником, вычисление трапецией, метод вычисления погрешность по правилу Рунге. Vvod уже был разобран в начале этого раздела. Далее стоит рассмотреть численные методы. Их структура в принципе одна и та же, но чем точнее метод, тем больше вычислений. Сначала находится так называемая высота h=(a-b)/n (для Симпсона: h=(a-b)/2n), а дальше идет цикл, в течение которого подставляются значения в функцию и суммируется, и в итоге будет получен результат. Метод прямоугольника: находим h=(b-a)/n. Запускается цикл for от 0 до n-1, внутри функции должно быть подставлено вот такое выражение: a+i\*h, где i-счетчик цикла, и затем суммируется. Полученная сумма умножается на h, и выводится как результат.

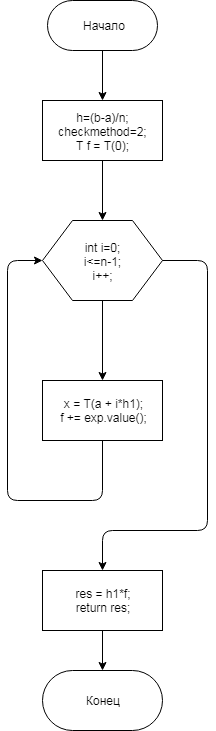


Рисунок 4. Блок-схема метода прямоугольников

Метод трапеции: опять же в начале надо найти h, опять запускается цикл for от 0 по n-1. Существенное отличие метода трапеции от прямоугольника в том, что подставлять надо в функцию два разных аргумента, а затем суммировать результату формулу, которая является площадью трапеции. В одну надо подставить a+i\*h, в другую a+(i+1)\*h, а дальше суммируется эти две функции и умножается на h/2,и затем суммируется в переменную для результата. После окончания цикла, выводится результат.

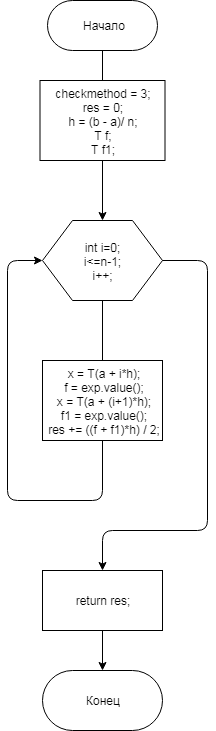


Рисунок 5. Блок-схема метода трапеций

Метод Симпсона: для начала опять надо найти h, но в этот раз надо разделить на 2n. Рассмотрев формулу, становится ясно, что перед циклом нам нужно сразу найти функцию при i=0 и i=2n. Вычисляем и начинаем цикл от 1 по 2n-1. Заметим в формуле, что имеются две суммы, одна умножается на 2, другая на 4. Та что на 2 ее i всегда четная, та что на 4 всегда нечетная. Следовательно, нужно сделать два условия: если четная, то суммируется в одну переменную, иначе в другую. Заканчивая цикл, суммируются первые два значения, которые были найдены, умножается сумма нечетных на 2, а сумма четных на 4, и суммируются с первыми двумя значениями. А затем все это умножается на h/3, и в конце выводится результат.

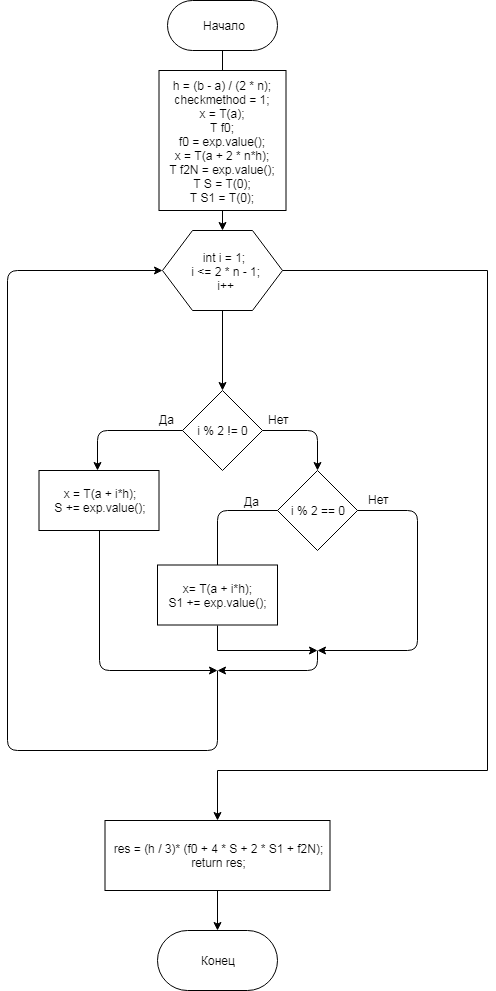


Рисунок 6. Блок-схема метода Симпсона

Отлично, все численные методы были разобраны, осталось создать метод вычисления погрешности. Здесь все просто, в каждом методе находится checkmethod, который присваивает определенное число. Допустим если checkmethod = 3, то программа считала интеграл по методу Симпсона, и по Правилу Рунге надо подставить два значений итерации в метод, который считал интеграл (в вышеперечисленном примере метод Симпсона), первый n, а второй 2n, затем надо вычитать полученные значение, и умножить на коэффициент, для Симпсона он равен 1/15. Выводим нашу погрешность. В итоге, мы создали наш класс Интеграл. Осталось создать интерфейс. Итак, внутри класса Интерфейса будет одно поле, а именно переменная класса интеграла, и будет 3 метода. Суть каждого метода одна и та же. В начале надо запросить то, что внутри полей, которые заполнил пользователь (функция, верхний, нижний предел, число делений функции на отрезки). Затем перевести введенную функцию из string в double с помощью vvod(). Дальше вызывается метод, который выбран при нажатии данной кнопки, и погрешность, и затем указываются поля в которые надо ввести результат и погрешность. В итоге получается программа с интерфейсом! Но перед тем как открывать шампанское, стоит обязательно проверить ее на работоспособность.

**Тестирование реализованной программной системы**

Программа будет тестироваться таким образом: в начале вписывается значения функции, верхнего и нижнего предела, а также число делений на отрезки функции, затем вычисления которые проводились вручную, и в конце следует вписать вычисления, которые выдает программа. Как уже было сказано, численные методы лишь близки к истинному результату, поэтому стоит ожидать, что они будут равны примерно к истинному результату, также надо обязательно обратить внимание на то, чтобы ближе всего к истинному результату был метод Симпсона, затем метод трапеций, и в конце метод прямоугольников. Если они находятся в другом порядке, значит где-то есть ошибка в программе.

Таблица 1. Тестирование функции метода Симпсона

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Номер теста | Исходные данные | | | | Эталон результата  (ожидаемый результат) | Результат программы (заполняется при выполнении программы) | Отметка о правильном выполнении теста |
| funct | b | a | n | res | res |
| 1 | sin(x) | 3,14 | 0 | 70 |  | 1,99999873453937 | Пройден |
| 2 | cos(x) | 1,57 | 0 | 60 |  | 0,999999683094618 | Пройден |
| 3 | x\*x+1 | 3 | 1 | 80 |  | 10,6666666666667 | Пройден |
| 4 | exp(x) | 4 | 5 | 50 |  | -93,8150090746442 | Пройден |

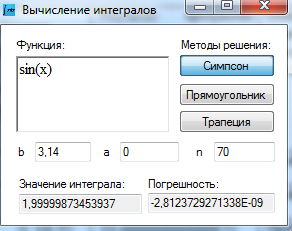


Рисунок 7. Результат программы при первом тесте

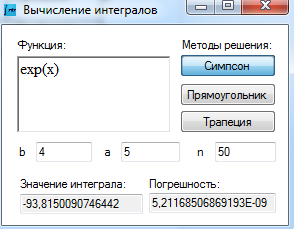


Рисунок 8. Результат программы при четвертом тесте

Таблица 2. Тестирование функции метода трапеций

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Номер теста | Исходные данные | | | | Эталон результата  (ожидаемый результат) | Результат программы (заполняется при выполнении программы) | Отметка о правильном выполнении теста |
| funct | b | a | n | res | res |
| 1 | cos(x)+  sin(x) | 3.14 | 0 | 70 |  | 2,00125574599984 | Пройден |
| 2 | sin(x) +x | 1.57 | 0 | 90 |  | 2,23162833430135 | Пройден |
| 3 | x+1 | 7 | 1 | 80 |  | 30,0533555555556 | Пройден |
| 4 | Exp(x) | 4 | 5 | 50 |  | -93,8181362155538 | Пройден |

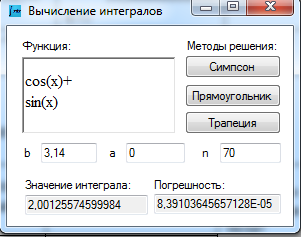


Рисунок 9. Результат программы при первом тесте

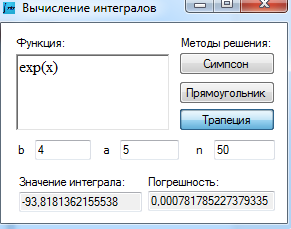


Рисунок 10. Результат программы при четвертом тесте

Таблица 3. Тестирование функции метода прямоугольников

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Номер теста | Исходные данные | | | | Эталон результата  (ожидаемый результат) | Результат программы (заполняется при выполнении программы) | Отметка о правильном выполнении теста |
| funct | b | a | n | res | res |
| 1 | x^2+x^3+1 | 6 | 0 | 500 |  | 400,48944 | Пройден |
| 2 | (cos(x))^2 | 1.57 | 0 | 60 |  | 0,798481397388307 | Пройден |
| 3 | x\*x+1 | 3 | 6 | 70 |  | -66,5794897959184 | Пройден |
| 4 | Exp(x) | 4 | 5 | 50 |  | -93,8150090746442 | Пройден |

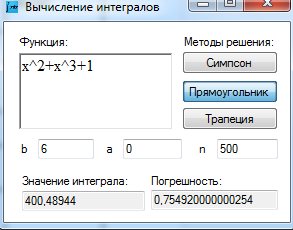


Рисунок 11. Результат программы при первом тесте

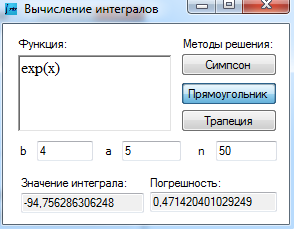


Рисунок 12. Результат программы при четвертом тесте

ЗАКЛЮЧЕНИЕ

В результате выполнения курсовой работы была получена программа, вычисляющая определенный интеграл 3 численными методами. Для нее был реализован рабочий интерфейс на базе Windows Forms. В процессе выполнения были созданы блок-схемы, макет интерфейса, диаграммы классов. Сам процесс выполнения показал, как создаются более серьезные проекты. Также были закреплены знания по C++ и математическому анализу. Было проведено тестирование программы, которое показало ее работоспособность. В итоге, все цели, перечисленные в начале работы, были достигнуты.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Каханер Д., Моулер К., [Нэш С.](https://ru.wikipedia.org/w/index.php?title=%D0%9D%D1%8D%D1%88,_%D0%A1%D1%82%D0%B8%D0%B2_(%D0%BC%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA)&action=edit&redlink=1" \o "Нэш, Стив (математик) (страница отсутствует)) Численные методы и программное обеспечение (пер. с англ.). — Изд. второе, стереотип. — М.: Мир, 2001. — 575 с.  
2. Герберт Шилдт. C++ Базовый курс, пер. Н. Ручко – 2-е изд. – Вильямс, 2014 г.

3. К. Вигерс, Д. Битти Разработка требований к программному обеспечению. 3-е изд, дополнительное, «Русская редакция», СПб. : БХВ-Петербург, 2014 г.

4. Microsoft Developer Network. <https://msdn.microsoft.com/> (Дата обращения: 06.05.18)

ПРИЛОЖЕНИЕ

**Intergal.h**

#pragma once

#include"iostream"

#include <ctime>

#include "math.h"

#include "string.h"

#include "exprtk.hpp"

#include "locale.h"

#include <cstdio>

using namespace std;

template <typename T> class

Intergral

{

typedef exprtk::symbol\_table<T> symbol\_table\_t;

typedef exprtk::expression<T> expression\_t;

typedef exprtk::parser<T> parser\_t;

private:

T x;

expression\_t exp;

int checkmethod;

public:

Intergral<T>(){

setlocale(LC\_ALL,"rus");

}

string funct;

double res;

double a;

double b;

double h;

unsigned int n;

double tochnost;

template <typename T>

void vvod()

{

symbol\_table\_t symbol\_table;

symbol\_table.add\_variable("x", x);

symbol\_table.add\_constants();

exp.register\_symbol\_table(symbol\_table);

parser\_t parser;

parser.compile(funct, exp);

}

template <typename T>

double findintergalsimpson(unsigned int n)

{

h = (b - a) / (2 \* n);

checkmethod = 1;

x = T(a);

T f0;

f0 = exp.value();

x = T(a + 2 \* n\*h);

T f2N = exp.value();

T S = T(0);

T S1 = T(0);

for (int i = 1; i <= 2 \* n - 1; i++)

{

x = T(a + i\*h);

if (i % 2 != 0)

{

S += exp.value();

}

else if (i % 2 == 0)

{

S1 += exp.value();

}

}

res = (h / 3)\* (f0 + 4 \* S + 2 \* S1 + f2N);

return res;

}

template <typename T>

double findintergralpryamoygol(unsigned int n)

{

checkmethod = 2;

h = (b - a)/ n;

T f = T(0);

for (double i = 0; i <= n - 1; i++)

{

x = T(a + i\*h);

f += exp.value();

}

res = h\*f;

return res;

}

template <typename T>

double findintegraltrap(unsigned int n)

{

checkmethod = 3;

res = 0;

h = (b - a)/ n;

T f;

T f1;

for (int i = 0; i <= n-1; i++)

{

x = T(a + i\*h);

f = exp.value();

x = T(a + (i+1)\*h);

f1 = exp.value();

res += ((f + f1)\*h) / 2;

}

return res;

}

template <typename T>

double findtochnost()

{

double f2n;

double fn;

if (checkmethod == 1)

{

f2n = findintergalsimpson<double>(n);

fn = findintergalsimpson<double>(n/2);

tochnost = (f2n - fn)/15;

}

else if (checkmethod == 2)

{

f2n = findintergralpryamoygol<double>(2\*n);

fn = findintergralpryamoygol<double>(n);

tochnost = f2n - fn;

}

else if (checkmethod == 3)

{

f2n = findintegraltrap<double>(2 \* n);

fn = findintegraltrap<double>(n);

tochnost = (f2n - fn)/3;

}

return tochnost;

}

};

**MyForm.cpp**

#include "MyForm.h"

using namespace System;

using namespace System::Windows::Forms;

[STAThread]

void main() {

Application::EnableVisualStyles();

Application::SetCompatibleTextRenderingDefault(false);

Project1::MyForm form;

Application::Run(%form);

}

**MyForm.h**

pragma once

#include"iostream"

#include <ctime>

#include "math.h"

#include "string.h"

#include "exprtk.hpp"

#include "locale.h"

#include <cstdio>

#include <msclr\marshal\_cppstd.h>

#include "Intergal.h"

namespace Project1 {

using namespace System;

using namespace System::ComponentModel;

using namespace System::Collections;

using namespace System::Windows::Forms;

using namespace System::Data;

using namespace System::Drawing;

/// <summary>

/// Сводка для MyForm

/// </summary>

public ref class MyForm : public System::Windows::Forms::Form

{

public:

MyForm(void)

{

InitializeComponent();

//

//TODO: добавьте код конструктора

//

}

protected:

/// <summary>

/// Освободить все используемые ресурсы.

/// </summary>

~MyForm()

{

if (components)

{

delete components;

}

}

private: System::Windows::Forms::Label^ label1;

protected:

private: System::Windows::Forms::TextBox^ textBox2;

private: System::Windows::Forms::TextBox^ textBox3;

private: System::Windows::Forms::TextBox^ textBox4;

private: System::Windows::Forms::Button^ button1;

private: System::Windows::Forms::Button^ button2;

private: System::Windows::Forms::Button^ button3;

private: System::Windows::Forms::Label^ label5;

private: System::Windows::Forms::RichTextBox^ richTextBox1;

private: System::Windows::Forms::Label^ label6;

private: System::Windows::Forms::Label^ label7;

private: System::Windows::Forms::TextBox^ textBox1;

private: System::Windows::Forms::TextBox^ textBox5;

private: System::Windows::Forms::Label^ label4;

private: System::Windows::Forms::Label^ label3;

private: System::Windows::Forms::Label^ label2;

protected:

private:

/// <summary>

/// Обязательная переменная конструктора.

/// </summary>

System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code

/// <summary>

/// Требуемый метод для поддержки конструктора — не изменяйте

/// содержимое этого метода с помощью редактора кода.

/// </summary>

void InitializeComponent(void)

{

System::ComponentModel::ComponentResourceManager^ resources = (gcnew System::ComponentModel::ComponentResourceManager(MyForm::typeid));

this->label1 = (gcnew System::Windows::Forms::Label());

this->textBox2 = (gcnew System::Windows::Forms::TextBox());

this->textBox3 = (gcnew System::Windows::Forms::TextBox());

this->textBox4 = (gcnew System::Windows::Forms::TextBox());

this->button1 = (gcnew System::Windows::Forms::Button());

this->button2 = (gcnew System::Windows::Forms::Button());

this->button3 = (gcnew System::Windows::Forms::Button());

this->label5 = (gcnew System::Windows::Forms::Label());

this->richTextBox1 = (gcnew System::Windows::Forms::RichTextBox());

this->label6 = (gcnew System::Windows::Forms::Label());

this->label7 = (gcnew System::Windows::Forms::Label());

this->textBox1 = (gcnew System::Windows::Forms::TextBox());

this->textBox5 = (gcnew System::Windows::Forms::TextBox());

this->label4 = (gcnew System::Windows::Forms::Label());

this->label3 = (gcnew System::Windows::Forms::Label());

this->label2 = (gcnew System::Windows::Forms::Label());

this->SuspendLayout();

//

// label1

//

this->label1->AutoSize = true;

this->label1->Location = System::Drawing::Point(12, 13);

this->label1->Name = L"label1";

this->label1->Size = System::Drawing::Size(56, 13);

this->label1->TabIndex = 0;

this->label1->Text = L"Функция:";

this->label1->Click += gcnew System::EventHandler(this, &MyForm::label1\_Click);

//

// textBox2

//

this->textBox2->Location = System::Drawing::Point(34, 117);

this->textBox2->Name = L"textBox2";

this->textBox2->Size = System::Drawing::Size(56, 20);

this->textBox2->TabIndex = 3;

this->textBox2->TextChanged += gcnew System::EventHandler(this, &MyForm::textBox2\_TextChanged);

//

// textBox3

//

this->textBox3->Location = System::Drawing::Point(119, 117);

this->textBox3->Name = L"textBox3";

this->textBox3->Size = System::Drawing::Size(59, 20);

this->textBox3->TabIndex = 5;

this->textBox3->TextChanged += gcnew System::EventHandler(this, &MyForm::textBox3\_TextChanged);

//

// textBox4

//

this->textBox4->Location = System::Drawing::Point(213, 117);

this->textBox4->Name = L"textBox4";

this->textBox4->Size = System::Drawing::Size(61, 20);

this->textBox4->TabIndex = 7;

this->textBox4->TextChanged += gcnew System::EventHandler(this, &MyForm::textBox4\_TextChanged);

//

// button1

//

this->button1->Location = System::Drawing::Point(178, 29);

this->button1->Name = L"button1";

this->button1->Size = System::Drawing::Size(96, 23);

this->button1->TabIndex = 8;

this->button1->Text = L"Симпсон";

this->button1->UseVisualStyleBackColor = true;

this->button1->Click += gcnew System::EventHandler(this, &MyForm::button1\_Click);

//

// button2

//

this->button2->Location = System::Drawing::Point(178, 58);

this->button2->Name = L"button2";

this->button2->Size = System::Drawing::Size(96, 23);

this->button2->TabIndex = 9;

this->button2->Text = L"Прямоугольник";

this->button2->UseVisualStyleBackColor = true;

this->button2->Click += gcnew System::EventHandler(this, &MyForm::button2\_Click);

//

// button3

//

this->button3->Location = System::Drawing::Point(178, 85);

this->button3->Name = L"button3";

this->button3->Size = System::Drawing::Size(96, 23);

this->button3->TabIndex = 10;

this->button3->Text = L"Трапеция";

this->button3->UseVisualStyleBackColor = true;

this->button3->Click += gcnew System::EventHandler(this, &MyForm::button3\_Click);

//

// label5

//

this->label5->AutoSize = true;

this->label5->Location = System::Drawing::Point(177, 13);

this->label5->Name = L"label5";

this->label5->Size = System::Drawing::Size(97, 13);

this->label5->TabIndex = 11;

this->label5->Text = L"Методы решения:";

//

// richTextBox1

//

this->richTextBox1->BackColor = System::Drawing::Color::White;

this->richTextBox1->Location = System::Drawing::Point(15, 31);

this->richTextBox1->Name = L"richTextBox1";

this->richTextBox1->Size = System::Drawing::Size(154, 77);

this->richTextBox1->TabIndex = 12;

this->richTextBox1->Text = L"";

this->richTextBox1->TextChanged += gcnew System::EventHandler(this, &MyForm::richTextBox1\_TextChanged);

//

// label6

//

this->label6->AutoSize = true;

this->label6->Location = System::Drawing::Point(15, 152);

this->label6->Name = L"label6";

this->label6->Size = System::Drawing::Size(113, 13);

this->label6->TabIndex = 13;

this->label6->Text = L"Значение интеграла:";

//

// label7

//

this->label7->AutoSize = true;

this->label7->Location = System::Drawing::Point(144, 152);

this->label7->Name = L"label7";

this->label7->Size = System::Drawing::Size(78, 13);

this->label7->TabIndex = 14;

this->label7->Text = L"Погрешность:";

this->label7->Click += gcnew System::EventHandler(this, &MyForm::label7\_Click);

//

// textBox1

//

this->textBox1->Location = System::Drawing::Point(18, 169);

this->textBox1->Name = L"textBox1";

this->textBox1->ReadOnly = true;

this->textBox1->Size = System::Drawing::Size(123, 20);

this->textBox1->TabIndex = 15;

this->textBox1->TextChanged += gcnew System::EventHandler(this, &MyForm::textBox1\_TextChanged);

//

// textBox5

//

this->textBox5->Location = System::Drawing::Point(147, 168);

this->textBox5->Name = L"textBox5";

this->textBox5->ReadOnly = true;

this->textBox5->Size = System::Drawing::Size(127, 20);

this->textBox5->TabIndex = 16;

this->textBox5->TextChanged += gcnew System::EventHandler(this, &MyForm::textBox5\_TextChanged);

//

// label4

//

this->label4->AutoSize = true;

this->label4->Location = System::Drawing::Point(193, 120);

this->label4->Name = L"label4";

this->label4->Size = System::Drawing::Size(13, 13);

this->label4->TabIndex = 6;

this->label4->Text = L"n";

this->label4->Click += gcnew System::EventHandler(this, &MyForm::label4\_Click);

//

// label3

//

this->label3->AutoSize = true;

this->label3->Location = System::Drawing::Point(100, 120);

this->label3->Name = L"label3";

this->label3->Size = System::Drawing::Size(13, 13);

this->label3->TabIndex = 4;

this->label3->Text = L"a";

//

// label2

//

this->label2->AutoSize = true;

this->label2->Location = System::Drawing::Point(15, 120);

this->label2->Name = L"label2";

this->label2->Size = System::Drawing::Size(13, 13);

this->label2->TabIndex = 2;

this->label2->Text = L"b";

this->label2->Click += gcnew System::EventHandler(this, &MyForm::label2\_Click);

//

// MyForm

//

this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);

this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;

this->BackColor = System::Drawing::Color::White;

this->ClientSize = System::Drawing::Size(286, 200);

this->Controls->Add(this->textBox5);

this->Controls->Add(this->textBox1);

this->Controls->Add(this->label7);

this->Controls->Add(this->label6);

this->Controls->Add(this->richTextBox1);

this->Controls->Add(this->label5);

this->Controls->Add(this->button3);

this->Controls->Add(this->button2);

this->Controls->Add(this->button1);

this->Controls->Add(this->textBox4);

this->Controls->Add(this->label4);

this->Controls->Add(this->textBox3);

this->Controls->Add(this->label3);

this->Controls->Add(this->textBox2);

this->Controls->Add(this->label2);

this->Controls->Add(this->label1);

this->Icon = (cli::safe\_cast<System::Drawing::Icon^>(resources->GetObject(L"$this.Icon")));

this->Name = L"MyForm";

this->Text = L"Вычисление интегралов";

this->ResumeLayout(false);

this->PerformLayout();

}

#pragma endregion

Intergral<double> \*l = new Intergral<double>();

private: System::Void label4\_Click(System::Object^ sender, System::EventArgs^ e) {

//Intergral<double> \*l = new Intergral<double>();

}

private: System::Void button1\_Click(System::Object^ sender, System::EventArgs^ e) {

//Метод Cимпсона

//l->findintergalsimpson<double>(l->n);

try {

String^ txt = richTextBox1->Text;

double b = Convert::ToDouble(textBox2->Text);

double a = Convert::ToDouble(textBox3->Text);

unsigned int n = Convert::ToInt32(textBox4->Text);

l->b = b;

l->a = a;

l->n = n;

l->funct = msclr::interop::marshal\_as<std::string>(txt);

l->vvod<double>();

l->findintergalsimpson<double>(l->n);

textBox1->Text = l->res.ToString();

textBox5->Text = l->findtochnost<double>().ToString();

}

catch (...) {

textBox1->Text = "Неверный формат !";

}

}

private: System::Void label1\_Click(System::Object^ sender, System::EventArgs^ e) {

}

private: System::Void label7\_Click(System::Object^ sender, System::EventArgs^ e) {

}

private: System::Void richTextBox1\_TextChanged(System::Object^ sender, System::EventArgs^ e) {

}

private: System::Void textBox2\_TextChanged(System::Object^ sender, System::EventArgs^ e) {

}

private: System::Void textBox3\_TextChanged(System::Object^ sender, System::EventArgs^ e) {

}

private: System::Void textBox4\_TextChanged(System::Object^ sender, System::EventArgs^ e) {

}

private: System::Void textBox1\_TextChanged(System::Object^ sender, System::EventArgs^ e) {

}

private: System::Void button2\_Click(System::Object^ sender, System::EventArgs^ e) {

//Метод прямоугольника

try {

String^ txt = richTextBox1->Text;

double b = Double::Parse(textBox2->Text);

double a = Double::Parse(textBox3->Text);

unsigned int n = Convert::ToInt32(textBox4->Text);

l->b = b;

l->a = a;

l->n = n;

l->funct = msclr::interop::marshal\_as<std::string>(txt);

l->vvod<double>();

l->findintergralpryamoygol<double>(l->n);

textBox1->Text = l->res.ToString();

textBox5->Text = l->findtochnost<double>().ToString();

}

catch (...) {

textBox1->Text = "Неверный формат !";

}

}

private: System::Void button3\_Click(System::Object^ sender, System::EventArgs^ e) {

//Метод Трапеции

try {

String^ txt = richTextBox1->Text;

double b = Double::Parse(textBox2->Text);

double a = Double::Parse(textBox3->Text);

unsigned int n = Convert::ToInt32(textBox4->Text);

l->b = b;

l->a = a;

l->n = n;

l->funct = msclr::interop::marshal\_as<std::string>(txt);

l->vvod<double>();

l->findintegraltrap<double>(l->n);

textBox1->Text = l->res.ToString();

textBox5->Text = l->findtochnost<double>().ToString();

}

catch (...) {

textBox1->Text = "Неверный формат !";

}

}

private: System::Void textBox5\_TextChanged(System::Object^ sender, System::EventArgs^ e) {

}

private: System::Void label2\_Click(System::Object^ sender, System::EventArgs^ e) {

} }; }