Аннотация

Настоящий документ является пояснительной запиской выпускной квалификационной работы на тему «Разработка программы для автоматизации работ при провидении олимпиады по программированию».

С помощью данного программного продукта организаторы олимпиады смогут повысить качество и скорость проверки решений участников.

Пояснительная записка включает в себя введение, составленные в соответствии с требованиями ГОСТ ЕСПД программные документы «Описание программы» и «Руководство оператора», «Расчет сметы затрат на разработку программного продукта», заключение, список литературы и приложения. В приложениях помещены программный код, схемы по ГОСТ ЕСПД, иллюстрации процесса разработки и функционирования программного продукта.

УТВЕРЖДАЮ:

Зам.директора по УМ и ВР

_____ /Иванова Л.Н.

«_____»_ _____2020

_____г.

ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ
на выпускную квалификационную работу

Обучающемуся группы 6901 *Харитонов Н.С.*
(№ группы) (И. О. Фамилия)

1. Тема выпускной квалификационной работы Разработка программы для автоматизации работ при проведении олимпиады по программированию (формирование тестов для задач, проверка тестов)

2. Форма выпускной квалификационной работы дипломный проект

3. Срок сдачи студентом законченной выпускной квалификационной работы
«5» июня 2020 г.

4. Исходные данные
Использовать ГОСТ «Единая система программной документации»; стандарт НовГу по правилам оформления документации; материалы, самостоятельно собранные в ходе преддипломной практики, в специальной литературе и сети Интернет.

5. Перечень подлежащих разработке задач/вопросов:
Должна быть выполнена разработка указанного программного продукта и составлена пояснительная записка к дипломному проекту, включающая в себя общую (теоретическую) часть, специальную (практическую, опытно-экспериментальную) часть и экономическую часть.
В общей части дается теоретическое освещение темы, включающее пункты «Постановка задачи», «Выбор инструментальных средств», «Технико-математическое описание задачи», «Обзор и анализ существующих программных систем»;
Специальная часть предполагает разработку продукта, который может быть представлен программным приложением или отдельным модулем приложения, информационной системой (desktop-приложение или веб-приложение) и включает в себя:
-анализ задачи, в котором описывается разработка и проектирование модели базы данных, разрабатываемый пользовательский интерфейс в виде warframe эскизов;
-алгоритмическое конструирование, т.е. обоснование, выбор и описание алгоритмов решения программы в целом и отдельных модулей системы, описание блок-схем алгоритмов;
-описание работы программы, которое должно включать организацию данных и внутреннего интерфейса; используемые методы; составные части программы и взаимосвязь между ними; тестирование программы – тест-кейсы;
-описание методов защиты данных;
-эксплуатационные документы - инструкции, руководства, которые включают описание требований к оборудованию; процесс установки программы; руководство пользователя

(оператора), в том числе при разработке многопользовательского приложения руководство для администратора, модератора; технику безопасности при работе на компьютере.
Экономическая часть должна содержать пункты «Расчет основной и дополнительной заработной платы», «Расчет стоимости материалов и программного обеспечения», «Расчет накладных расходов», «Составление и расчет цены реализации программного продукта», документы должны содержать график и структуры затрат.
Текст программы должен быть помещён в приложении.

6. Перечень графического/иллюстративного/ практического материала
В электронной форме на любом носителе предоставляются все исходные файлы программного продукта в виде, достаточном для его воспроизведения. В пояснительной записке должны быть схемы, соответствующие требованиями ГОСТ 19.701-90. Должны быть представлены следующие схемы:
-схема данных, схемы алгоритмов, схемы взаимодействия модулей программы, схема иерархии классов, диаграмма Ганта, функциональные схемы (IDEF0, UML) и т.п.
Должна быть разработана презентация для защиты дипломного проекта.

7 Прочие условия

8 Консультанты по разделам выпускной квалификационной работы

Раздел	Консультант
Раздел	Консультант
Специальная часть	Цымбалюк Лариса Николаевна
Экономическая часть	Лебедева Галина Вячеславовна
Нормоконтроль	Чернега Анна Михайловна

Задание выдал руководитель

_____	Цымбалюк	«20» апреля 2020 г.
подпись	фамилия	дата

Задание принял к исполнению обучающийся

_____	Харитонов	«20» апреля 2020 г.
подпись	фамилия	дата

№ строки	Формат	Обозначение	Наименование	Кол-во листов	Примечание
1					
2			Документация общая		
3					
4			Вновь разработанная		
5					
6	А4	ПТК. ДП 6901 04. 000ПЗ	Пояснительная записка	46	альбом
7			Программа		флеш-накопитель
8			Презентация		флеш-накопитель
9					
10					
11					
12					
13					
14					
15					
16					
17					
18					
19					
20					
21					
22					
23					

					ПТК. ДП 6901 04. 000ВД			
Изм.	Лист	№ докум.	Подпись	Дата				
Разраб.		Харитонов Н.С.				*Лит.*	*Лист*	*Листов*

Провер.	Цымбалюк Л.Н			РАЗРАБОТКА ПРОГРАММЫ ДЛЯ АВТОМАТИЗАЦИИ РАБОТ ПРИ ПРОВЕДЕНИИ ОЛИМПИАДЫ ПО ПРОГРАММИРОВАНИЮ (ФОРМИРОВАНИЕ ТЕСТОВ ДЛЯ ЗАДАЧ, ПРОВЕРКА ТЕСТОВ)		Д		1	1
Н. контр.	Чернега А.М.				09.02.03				
Утв.	Цымбалюк Л.Н.								

НОВГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ ЯРОСЛАВА МУДРОГО	Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования «Новгородский государственный университет имени Ярослава Мудрого» ПОЛИТЕХНИЧЕСКИЙ КОЛЛЕДЖ

Утверждаю:
Зам. директора по УМ и ВР
_____ Л.Н. Иванова
(подпись)
«____» _____ 2020 года

РАЗРАБОТКА ПРОГРАММЫ ДЛЯ АВТОМАТИЗАЦИИ РАБОТ ПРИ ПРОВЕДЕНИИ ОЛИМПИАДЫ ПО ПРОГРАММИРОВАНИЮ (ФОРМИРОВАНИЕ ТЕСТОВ ДЛЯ ЗАДАЧ, ПРОВЕРКА ТЕСТОВ)

Пояснительная записка к дипломному проекту по специальности
09.02.03 Программирование в компьютерных системах
ПТК. ДП 6901 00. 000ПЗ

Согласовано:
Консультант по спец. части
_____Л.Н. Цымбалюк
«___»_____2020 года
Консультант по экон. части
_____Г.В. Лебедева
«____»_____2020 года
Нормоконтроль
_____ А.М. Чернега
«____»_____2020 года

Руководитель
_____Л.Н. Цымбалюк
«___» _____ 2020 года
Выполнил:
обучающийся группы 6901
_____Н.С. Харитонов
«___» _____ 2020 года

Заместитель директора по УПР
_____ А.М. Чернега
(подпись)
«____» _____ 20___ года

Содержание

Введение

В соответствие с заданием на выпускную квалификационную работу требуется разработать приложение для автоматизации проведения тестирования при организации и проведении муниципального этапа всероссийской олимпиады школьников по информатике.

Актуальность и практический аспект данной работы связаны с тем, что при проверке и провидении итогов олимпиады в полуавтоматическом режиме нивелируется риск допущения ошибок по вине человеческого фактора, а о превосходстве машин в скорости выполнения рутинных задач и говорить не приходиться.

Задачей данной дипломной работы в связи с указанной целью является разработка приложения со следующим перечнем функционала:

- компиляция программы из исходного кода участника в исполняемый файл.
- оценка правильности решения, которая будет происходить путем исполнения программы с заранее подготовленными и неизвестными участникам входными файлами. После выполнения результат сравнивается с эталонным и итог записывается в локальную базу данных.
- подведение итогов в удобном, для восприятия человеком, формате csv.

Общая (теоретическая) часть

1.1 Постановка задачи

В соответствии с выбранной темой на дипломное проектирование требуется разработать программу для автоматизации процесса проведения олимпиады по информатике, а именно компиляция кода, проведение тестов, и формирование итогов.

1.1.1 Обоснование необходимости разработки

Конечно, можно было бы вручную заниматься проверкой решения от каждого участника, но использования автоматизированного подхода позволит избежать ошибок, связанных с человеческим фактором, а также время проверяющей комиссии на подведение результатов.

1.1.2 Технико-математическое описание задачи

1.1.2.1 Математические методы

Для расчёта баллов участника по каждой задаче используется формула подсчёта процента успешно пройдённых тестов.

Для проверки идентичности выходного файла решения участника и правильного ответа используются их контрольная сумма по алгоритму md5.

1.1.2.2 Описание технологий разработки

Приложение может быть разработано на базе следующих технологий:

- ADO.NET - технология, предоставляющая доступ и управление данными, хранящимся в базе данных или других источниках, основанных на платформе .NET Framework и входящая в состав .NET Framework 2.0, представляет собой набор библиотек;
- PHP - это распространенный язык программирования общего назначения с открытым исходным кодом. PHP специально сконструирован для веб-разработок и его код может внедряться непосредственно в HTML;

– C++ - компилируемый, статически типизированный язык программирования общего назначения;

– MySQL - свободная реляционная система управления базами данных;

– MmongoDB - документ ориентированная система управления базами данных с открытым исходным кодом, не требующая описания схемы таблиц. Классифицирована как NoSQL, использует JSON-подобные документы и схему базы данных;

– SQLite - компактная встраиваемая СУБД. Исходный код библиотеки передан в общественное достояние.

1.1.3 Характеристика бизнес-процессов

Необходимо разработать приложение с интуитивно понятным графическим интерфейсом. Перед запуском работы приложения пользователь должен загрузить решения участников и подготовленные тесты в определённые директории, после чего пользователь запускает программу и выбирает из списка задач и из списка участников, после чего выбирает действие «Скомпилировать» или «Проверить». Кнопка «Итог» формирует результаты. Функциональная модель и характеристика бизнес-процессов изображена на рисунках А.1 и А.2.

1.2 Обзор аналогов разрабатываемой системы

На данный момент можно выделить три аналога автоматической системы тестирования:

– TestSys – разработана и поддерживается Санкт-Петербургским государственным университетом, используется им же для проведения олимпиад по правилам ACM и IOI;

– Ejudge – разработана и поддерживается Московским государственным университетом, используется для проведение

четвертьфинала мира в Московском подрегионе Северо-Восточного Европейского региона, а так же в летней компьютерной школе для проведения Открытого кубка России по программированию;

– Codeforce – разработана и поддерживается университетом информационных технологий, механики и техники.

1.3 Обоснование проектных решений

1.3.1 Обоснование выбора среды программирования

Для создания приложения была выбрана среда программирования Microsoft Visual Studio 2019. Данная среда разработки распространяется бесплатно, а её функционала для разработки более чем достаточно. С помощью шаблона «Разработка приложений Windows Forms» очень удобно создавать приложения с графическим интерфейсом для Windows, что идеально подходит для решения поставленной задачи.

1.3.2 Выбор инструментальных средств

В качестве базы данных было принято решение использовать локальную SQL базу данных SQLite. Для создания блок-схем, презентации, таблиц и документации было решено использовать пакет офисных программ Microsoft Office. Программы в данном пакете отлично справляются с перечисленными задачами, а также предустановка этого пакета на большинстве компьютеров избавляет от поиска альтернатив данным программам. Для работы в Интернете было решено использовать браузер Google Chrome, являясь достаточно быстрым и простым в использовании.

1.3.3 Выбор языка программирования

Язык программирования C# также является продуктом компании Microsoft. При использовании языка в среде разработки Visual Studio 2019, C# раскрывается полностью, что позволяет создавать приложения максимально

эффективно. Так же средствами языка предоставляется удобный набор функций для работы с файлами, запуска и контроля новых процессов. Именно поэтому выбор языка программирования для решения поставленной задачи пал на C#.

1.3.4 Информационное обеспечение

Информационным обеспечением являются данные в виде списка участников олимпиады, представляющих собой набор директорий с решениями и списка задач, представляющих собой набор директорий с тестами. Для нахождения остальной информации было решено использовать поисковую систему Google, как одну из лучших на данный момент.

1.3.5 Техническое обеспечение

Для запуска приложения минимально необходимо иметь:

- Microsoft® Windows® 2000 и XP;
- Pentium III или AMD Athlon, с тактовой частотой 1 ГГц;
- 256 МБ Оперативной памяти.

1.4 Требования к программе

Для работы программы должен быть установлен Microsoft Build Tools и открыт доступ приложения к бат файлу “C:/Program Files (x86)/Microsoft Visual Studio/2019/BuildTools/Common7/Tools/VsDevCmd.bat”.

2 Специальная, практическая, опытно – экспериментальная часть

2.1 Анализ задачи

В соответствии с заданием для дипломного проекта требуется разработка программы для автоматизации процесса проведения олимпиады по информатике среди школьников. Программа должна компилировать решение конкурсантов, проводить тестирование с заранее подготовленными входными и выходными данными и подводить результаты.

2.1.1 Информационное моделирование предметной области

2.1.1.1 Построение информационной модели

Пользователями данной программы являются члены комиссии, которые занимаются проведением олимпиады. Реализация программы состоит из нескольких этапов:

- создание графического интерфейса;
- автоматическая загрузка списка задач и участников;
- компиляция исходного кода в исполняемый файл;
- проведение тестов;
- вывод результатов в csv файл.

2.1.1.2 Описание структуры базы данных

Как уже упоминалась ранее, для хранения данных в данном дипломном проекте была выбрана локальная база данных SQLite и данные хранятся в принятом для неё формате, а именно созданы две сущности, структура которых представлена на рисунке 1.

```
public class Contest
{
    public Guid Id { get; set; }
    public string Name { get; set; }
    public List<Contestant> Contestants { get; set; }
}
public class Contestant
{
    public string Name { get; set; }
    public string Task { get; set; }
    public string TestId { get; set; }
    public string Result { get; set; }
}
```

Рисунок 1 – Структура для отображения данных

2.1.2 Проектирование пользовательского интерфейса

Для проектирования интерфейса и дизайна приложения было решено использовать программу Photoshop CS6. Создаётся новый файл со следующими параметрами: имя файла «interface», ширина 800 пикселей, высота 600 пикселей, разрешение файла 72 пикселей на дюйм. Вид окна нового файла представлен на рисунке 2.

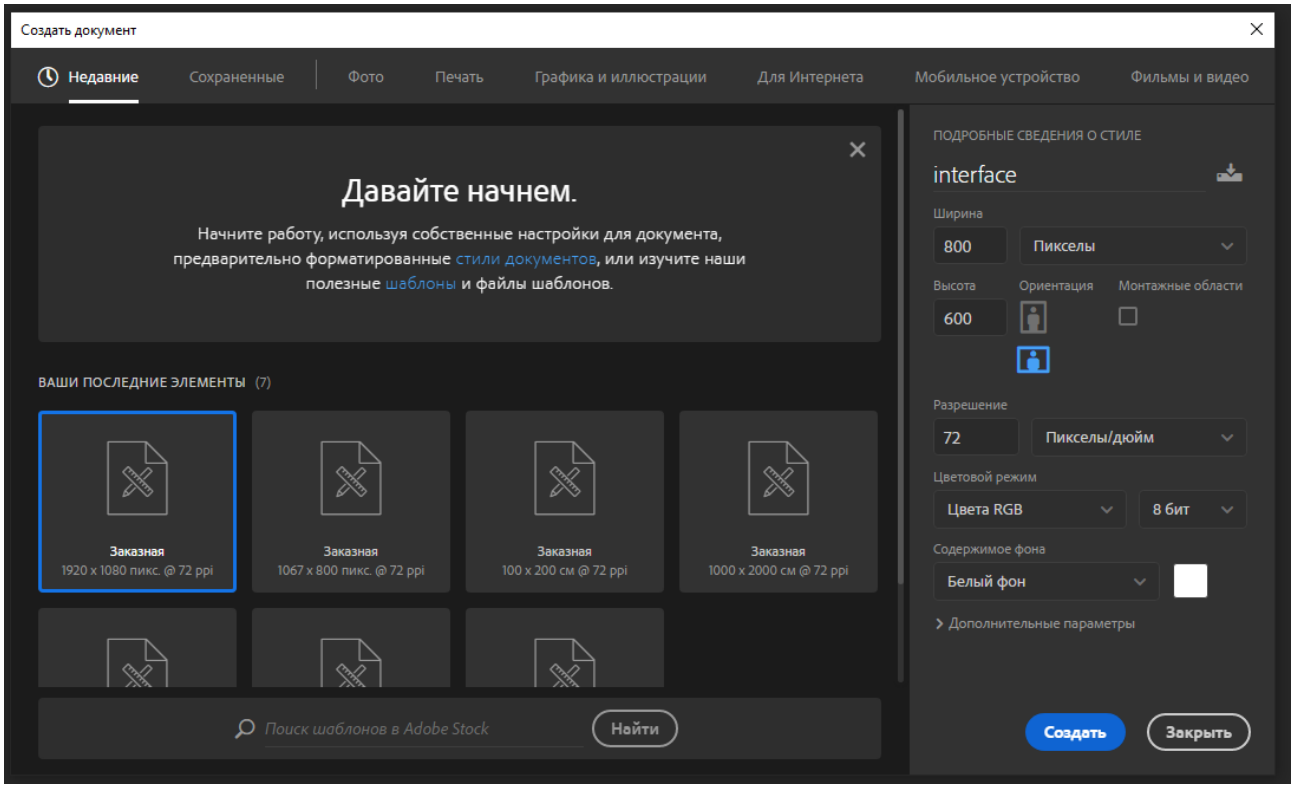

Рисунок 2 – Создание файла в Photoshop

С помощью стандартных инструментов программы был составлен макет интерфейса, который представлен на рисунке 3.

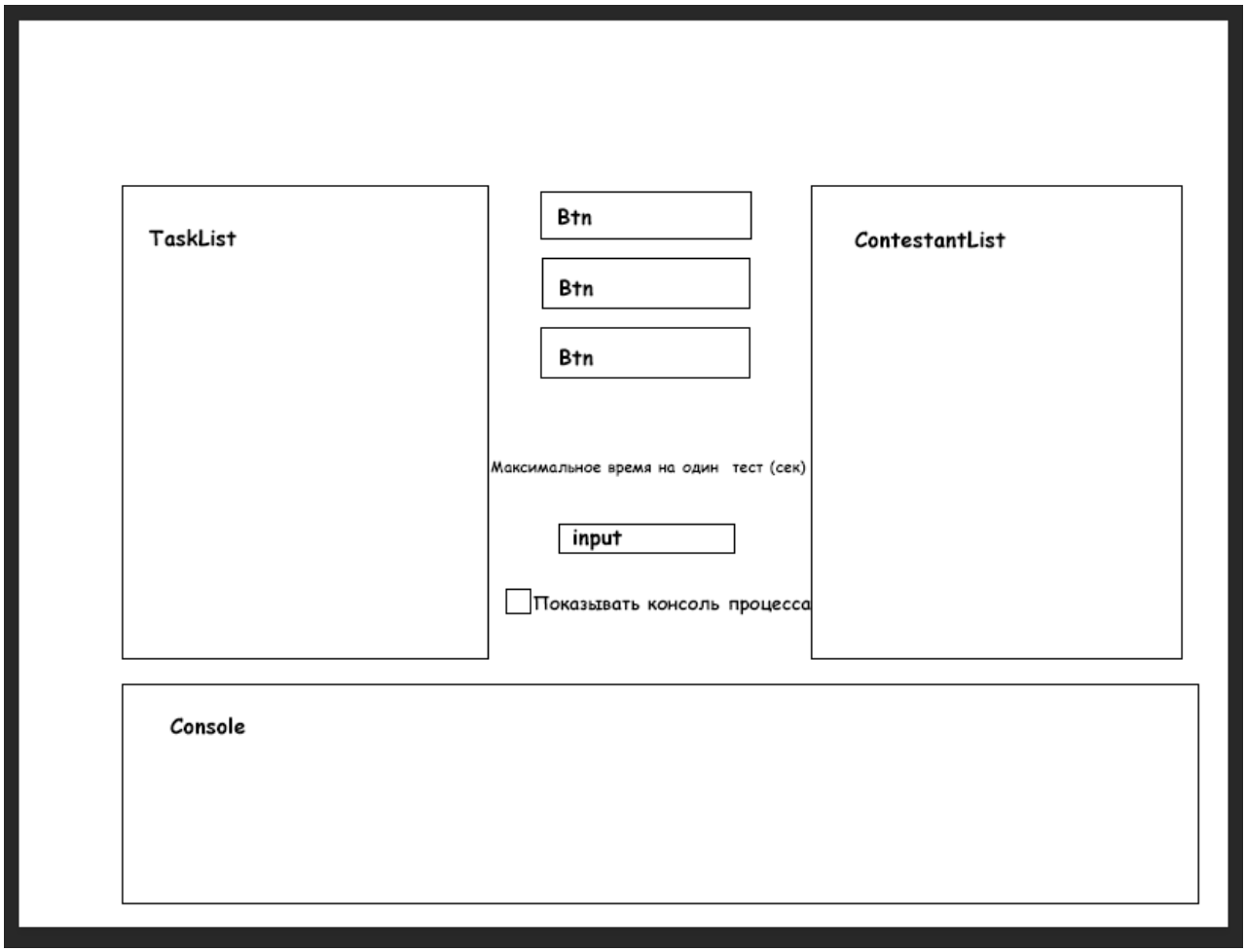

Рисунок 3 – Макет интерфейса программы

Теперь по данному макету предстоит составить интерфейс в Microsoft Visual Studio. Для этого было создано приложение Windows Forms (.NET Framework) и составлен интерфейс, представленный на рисунке 4.

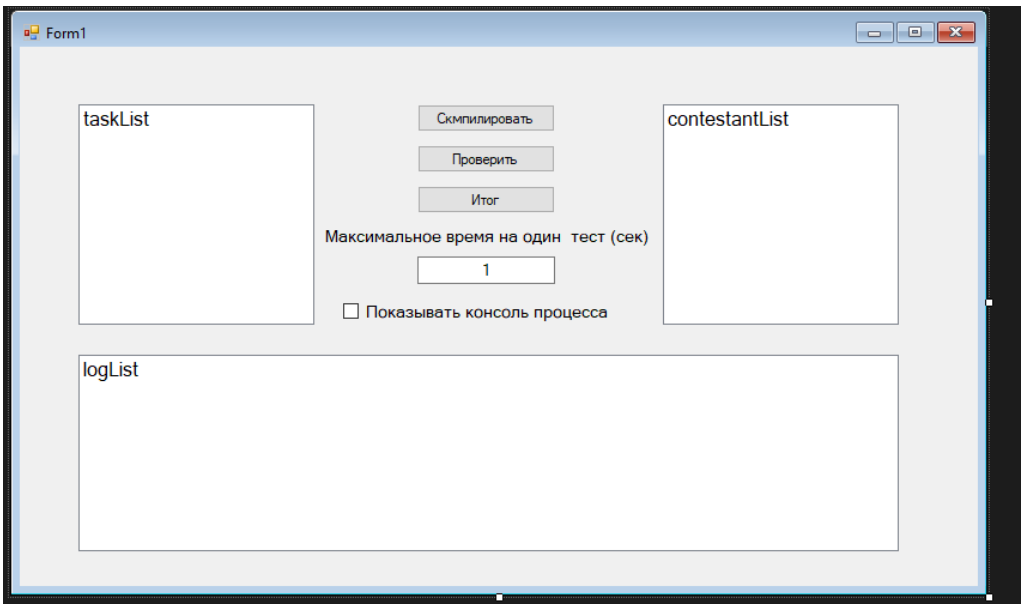

Рисунок 4 – Финальный интерфейс приложения

2.2 Описание логической структуры

2.2.1 Структура приложения

Структура приложения изображена на рисунке 5.

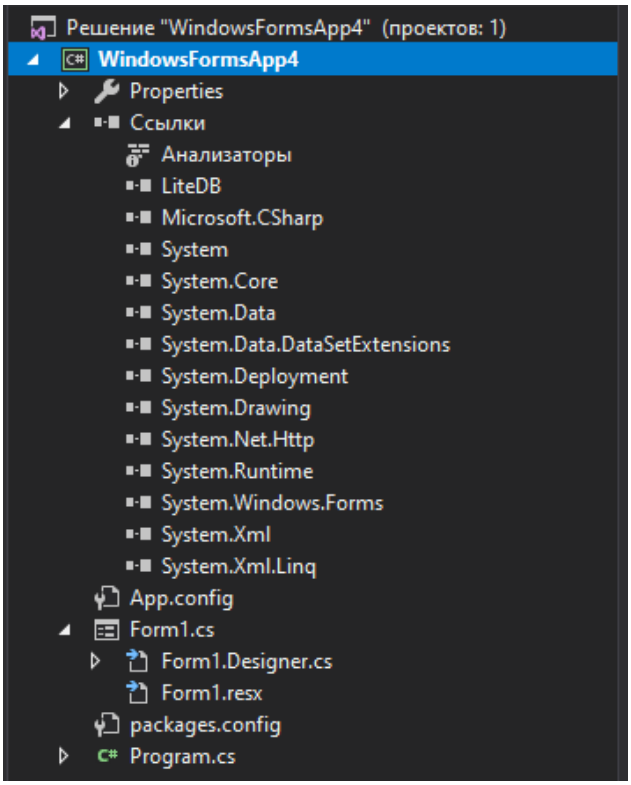

Рисунок 5 – Структура приложения

2.2.2 Алгоритм программы

После запуска и генерации интерфейса, приложение собирает информацию о всех доступных задачах и участников и вносит её в советующие для этой информации listBox’ы. Далее проверяет была ли уже создана база данных и если нет, то создаёт её. Схема алгоритма расположена на рисунке Б.1.

Алгоритм вывода в консоль представляет метод принимающий текст произвольной длины и выводящий его в импровизированную консоль строчками максимальной длиной в 80 символов с указанием текущего времени и прокрутки консоли в низ при добавлении нового элемента. Если в строке оказалось больше 80 символов, то метод будет рекурсивно вызывать себя с оставшейся частью строки, до тех пор, пока не выведет её полностью. Схема алгоритма вывода в консоль расположена на рисунке.

Во всех фрагментах программы, где происходит работа с файлами и процессами, потенциально опасные места обёрнуты в конструкцию try/catch. В случае возникновения критических ситуаций исполнение текущего кода прервётся, в консоль выведется информация об ошибке, но само приложение продолжит работать.

После загрузки программа ожидает действий пользователя, а именно нажатие выбор элементов и/или нажатие на одну из трёх кнопок.

При нажатии на кнопку «Скомпилировать» программа составит список из выделенных пользователем задач и конкурсантов. А именно методом перебора пропускает через алгоритм компиляции каждое выбранное решение.

Алгоритм компиляции заключается в том, что сначала на основании текущего участника строится путь к директории с его решениями. Затем в этой директории ищется текущая задача и по расширению определяется язык, на котором решение написано. Если файл написан на языке программирования python, то текущая итерация алгоритма компиляции прерывается с соответствующим выводом в консоль, так как данное решение не требует

компиляции. Если файл отсутствует, то текущая итерация алгоритма компиляции прерывается с соответствующим выводом в консоль. Далее по коду программы выводится сообщение в консоль, оповещающее о старте компиляции и о текущих параметрах. После чего директория конкурсанта отчищается от возможно уже ранее скомпилированных версий текущего решения, чтобы после завершения алгоритма иметь самую актуальную. Далее по коду приложения создаётся новый экземпляр класса Process, ему прописываются параметры запуска и вызывается метод для старта. Далее запускается таймер и цикл, в котором каждые 100 миллисекунд проверяется текущий статут процесса. Если компиляция не завершилась через минуту или произошла ошибка, алгоритм заканчивается с соответствующим выводом информации в консоль. Если компиляция прошла успешно, программа заканчивается с соответствующим выводом информации в консоль. Схема алгоритма расположена на рисунке Б.2.

При нажатии на кнопку «Проверить» программа составит список из выделенных пользователем задач и конкурсантов. А именно методом перебора пропускает через алгоритм тестирования каждое выбранное решение.

Алгоритм компиляции заключается в том, что сначала на основании текущего участника строится путь к исполняемому файлу с его решением. Если файл отсутствует, то текущая итерация алгоритма тестирования прерывается с соответствующим выводом в консоль. Затем текущее решение путём алгоритма перебора всех входных данных, для данной задачи, проходит тестирование. А именно в корневую директорию приложения копируются файлы с входными и выходными данными для текущего теста. Далее по коду приложения создаётся новый экземпляр класса Process, ему прописываются параметры запуска и вызывается метод для старта. Далее запускается таймер и приложение останавливается в ожидании завершения процесса. Если процесс не завершиться за заданное пользователем максимальное время, приложение возобновит работу и принудительно завершит процесс. Если процесс завершился удачно за отведённое ему на это время, то приложение

сравнит контрольную сумму файла результата работы приложения с эталонным по алгоритму md5. Если контрольные суммы файлов совпадают, то и сами файлы идентичны. При любом из исходов приложение остановит таймер после завершения процесса и запишет результат. После чего отчистит директорию от только что созданных файлов. Далее, если в базе данных уже имеется запись о данном тесте, то она обновляется, если же её нет, она создаётся. После чего выводится сообщение в консоль. Схема алгоритма расположена на рисунке Б.3.

При нажатии на кнопку «Итог» из базы данных выбирается информация о текущем соревновании, создаются и заполняются словари с количеством тестов по каждой задаче и количеством баллов за каждую задачу каждого участника. После чего полный результат записывается в файл «resultLog.csv», а краткий результат с итогами записывается в файл «result.csv».

2.2.3 Используемые методы

2.2.3.1 Стандартные методы

При разработке приложения была использована библиотека SQLite, необходимая для локальной базы данных. Данная библиотека распространяется с открытым кодом для всех желающих. Также были использованы стандартные библиотеки при создании приложения, а именно:

- System – содержит фундаментальный набор классов, такие как типы данных, события и т. д.;
- System.Collections.Generic – для работы со списками;
- System.Diagnostics – для работы точным временем исполнения и статистикой процессов;
- System.IO – для работы с файлами;
- System.Security.Cryptography - Для работы с криптографическими функциями;
- System.Linq – для работы с источниками данных, например список;

– System.Windows.Forms – для работы с графической частью приложения;

– System.Threading – для работы с потоками.

2.2.3.2 Пользовательские методы

В ходе разработки приложения были созданы следующие пользовательские методы:

– Form1_Load – обработчик события завершения загрузки формы;

– compilate – обработчик события нажатия по кнопке «Скомпилировать»;

– test – обработчик события нажатия по кнопке «Проверить»;

– show – обработчик события нажатия по кнопке «Итог»;

– ConsoleWrite – метод записи текста в консоль. В качестве параметров принимает текс для записи;

– ComputeMD5Checksum – метод получения контрольной суммы файла по алгоритму md5. В качестве параметров принимает путь к файлу.

2.2.3.3 Методы защиты данных

Данные хранятся в не зашифрованном виде, так как не представляют из себя никакой ценности.

2.2.4 Составные части программы и связи между ними

Для взаимодействия модулей программы используется набор стандартных библиотек, а также глобальные и локальные переменные.

Глобальные переменные:

– tasksPath – хранит в себе путь до заданий с тестами;

– contestantsPath – хранит в себе путь до конкурсантов с решениями;

– db – экземпляр класса LiteDatabase для работы с базой данных.

Локальные переменные:

– folder– переменная типа string, в методе Form1_Load, хранит в себе путь к директории;

– col– экземпляр класса ILiteCollection в методах Form1_Load, test, show, применяется для работы с базой данных ;

– ct– экземпляр класса Contest в методе Form1_Load, test, show, применяется для работы с данными из базы данных;

– tasks– переменная типа List<string>в методе compilate, show, хранит в себе список задач;

– contestants– переменная типа List<string>в методе compilate, show, хранит в себе список участников;

– fail– переменная типа bool в методе compilate, предназначается для маркировки неудачной компиляции;

– secToCompil – переменная типа int в методе compilate, хранит в себе количество секунд, отведённое на компиляцию;

– taskPath– переменная типа string в методе compilate, хранит в себе путь к директории текущего решения;

– extention– переменная типа string в методе compilate, хранит расширение текущего решения;

– compiler– экземпляр класса Process в методе compilate, предназначен для запуска cmd с заданными параметрами и компиляции решения;

– timmer– экземпляр класса Stopwatch в методе compilate, test, предназначен для расстановки точных временных меток начала и конца процесса;

– maxTestTime– переменная типа int в методе test, максимально допустимое время ожидание завершения решения в миллисекундах;

– exePath– переменная типа string в методе test, хранит в себе путь к исполняемому файлу решения;

– testId– переменная типа string в методе test, хранит в себе путь к текущему тесту;

– result– переменная типа string в методе test, хранит в себе результата работы теста;

– exe– экземпляр класса Process в методе test, предназначен для запуска решения;

– results– экземпляр класса Contest в методе test, применяется для работы с данными из базы данных;

– testId– переменная типа string в методе test, хранит в себе путь к текущему тесту;

– testCount– переменная типа Dictionary<string, int> в методе show, хранит в себе в методе show, количество тестов по каждому решению;

– result– переменная типа Dictionary<string, Dictionary<string, int>> в методе show, количество баллов по каждой задаче каждого участника;

– testCount– переменная типа Dictionary<string, int> в методе show, хранит в себе в методе show, количество тестов по каждому решению;

– swResLog– экземпляр класса StreamWriter в методе show, предназначен для работы с файлами;

– swRes– экземпляр класса StreamWriter в методе show, предназначен для работы с файлами;

– md5– экземпляр класса MD5 в методе ComputeMD5Checksum, предназначен для работы с криптографическими функциями;

– fs– экземпляр класса FileStream в методе ComputeMD5Checksum, предназначен для работы с файлами;

– fileData– переменная типа byte в методе ComputeMD5Checksum, хранит в себе файл в байтах;

– checkSum– переменная типа byte[] в методе ComputeMD5Checksum, хранит в себе хеш файла в байтах;

– result– переменная типа string в методе ComputeMD5Checksum, хранит контрольную сумму файла по алгоритму md5;

2.3 Описание работы программы

2.3.1 Общие сведения

Программа для автоматизации процесса проведения олимпиады по информатике представляет собой настольное приложение с графическим интерфейсом и написана на языке C# в среде программирования Visual Studio 2019.

2.3.2 Функциональное назначение

Настольная программа, предназначенная автоматизации процесса тестирования решений при проведении олимпиады по информатике без каких-либо трудностей.

2.3.3 Входные данные

Входными данными приложения являются:

– список файлов и директорий с тестами к задачам;

– список файлов и директорий с решениями конкурсантов;

– управляющее воздействие пользователя.

2.3.4 Выходные данные

Выходными данными в программе являются два csv файла, с подробным и кратким результатом тестирования.

Файл с подробным результатом содержит информацию о количестве баллов каждого участника за каждую задачу.

Файл с кратким результатом содержит информацию о каждом участнике и сумме его баллов за все задачи.

2.3.5 Тестирование

При работе с приложением стоит не забыть до запуска загрузить подготовленные тесты и решения в соответствующие директории, но если этого не сделать, приложение всё равно не сломается, просто при запуске программы таким способом в полях для данных будет выведено соответствующее предупреждение. Что показано на рисунках 6 и 7.

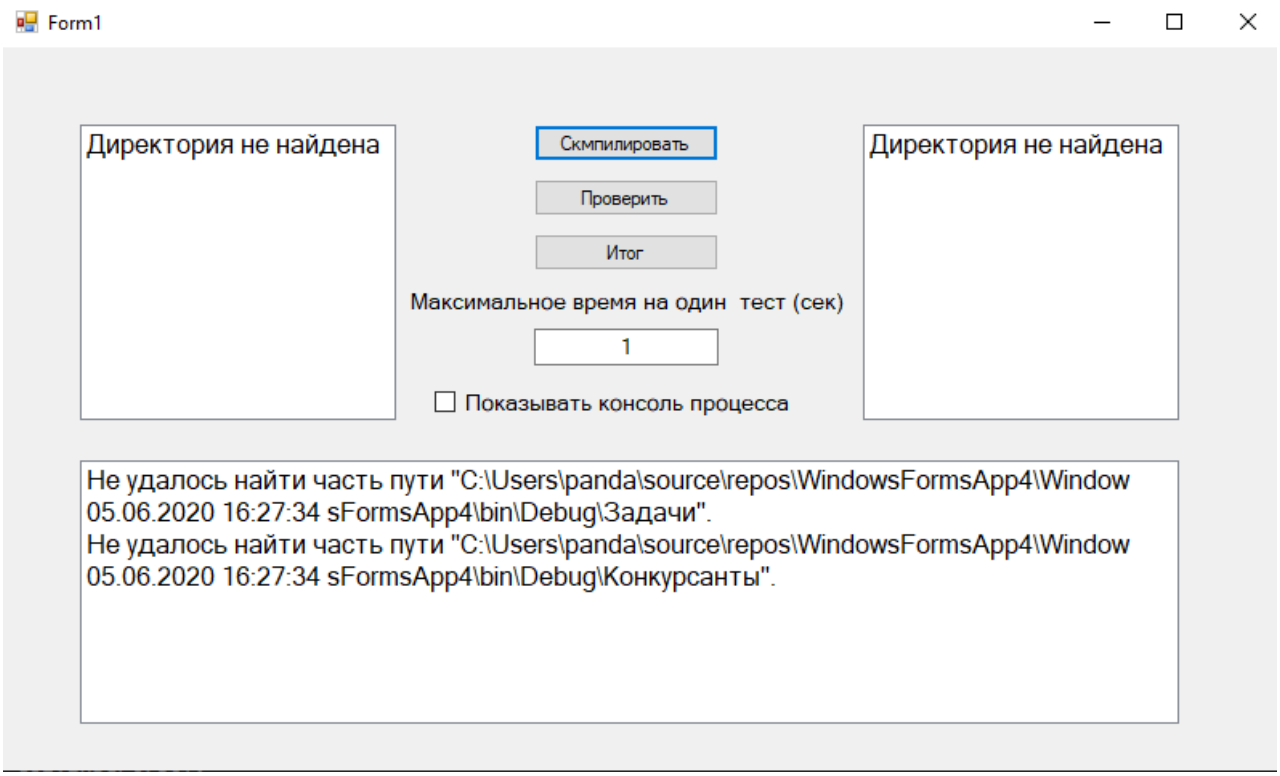

Рисунок 6 – Результат запуска без директорий с данными

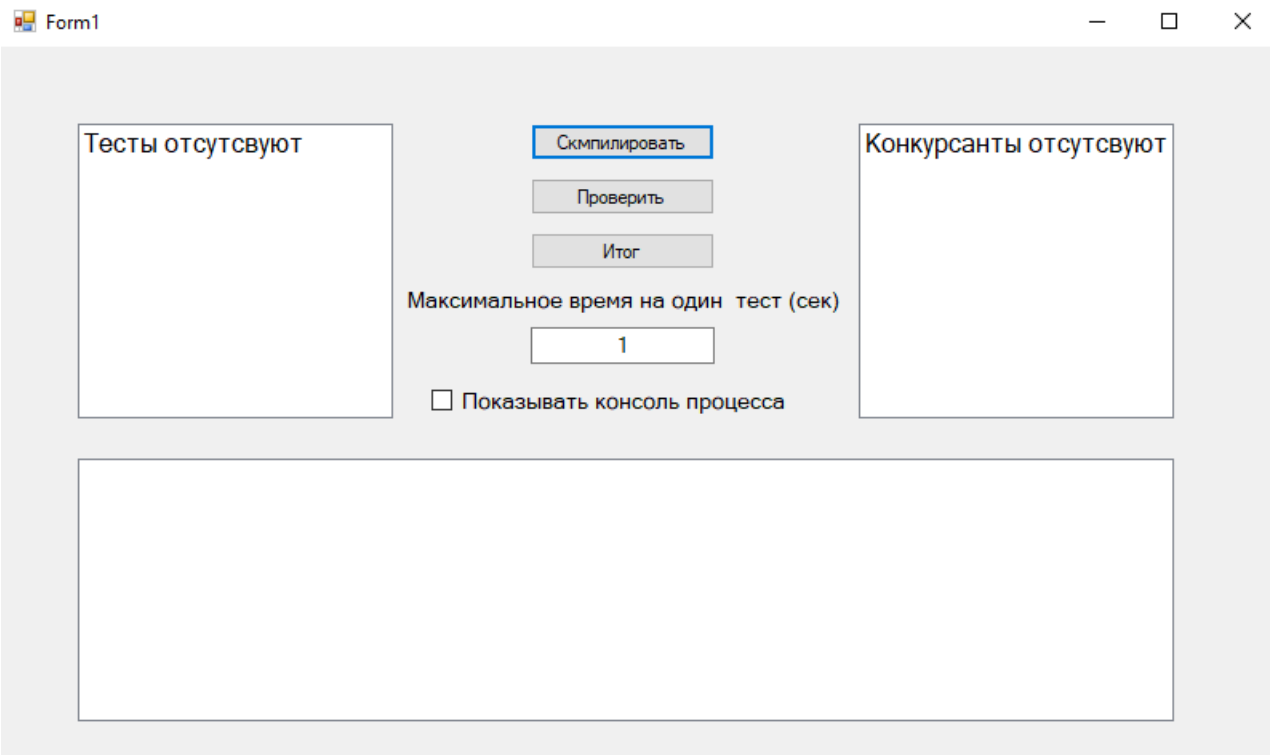

Рисунок 7 – Результат запуска с пустыми директориями данных

Также можно попытаться запустить проверку или компиляцию отсутствующего файла, результат приведён на рисунке 8.

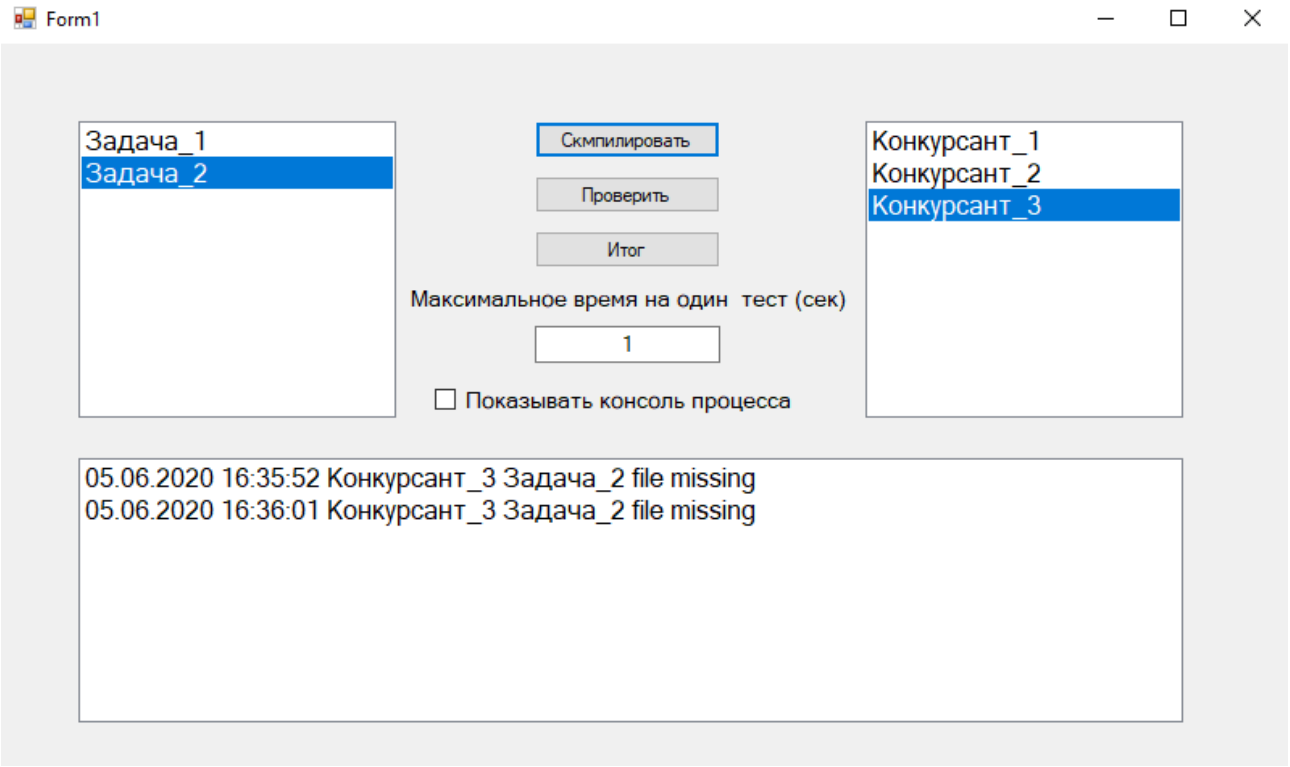

Рисунок 8 – Компиляция и проверка несуществующего файла

2.3.6 Вызов и загрузка

Запуск программы осуществляется стандартными для приложений Windows способами. Например, можно найти исполняемый файл программы ExcelApp.exe. В результате на экране появляется окно программы.

2.4 Руководство оператора

2.4.1 Назначение программы

Программа предназначена для упрощения проведения тестов решений олимпиады по информатике, а именно автоматизировать процессы, связанные с высокой затратой человеческих ресурсов и риска возникновения ошибок, связанных с человеческим фактором. Именно для этого был разработан программный продукт, который в форме приложения реализовывает данную возможность.

2.4.2 Условия выполнения

2.4.2.1 Требование к оборудованию

Программа «ExcelApp» работает на компьютере стандартной конфигурации под управлением операционной системы Windows.

2.4.2.2 Установка программы

Для работы программы необходимо предварительно установить систему сборки Microsoft Build Tools 2015 или новее, а также обеспечить приложение правами на запуск bat файла «VsDevCmd.bat» расположенного в директории этой системы. Устанавливать саму программу не нужно, достаточно скопировать корневую директорию и запустить исполняемый файл.

2.4.3 Выполнение программы

Первым делом, перед запуском программы необходимо загрузить решения для тестирования и сами тесты в соответствующие для этого директории.

После запуска программы необходимо выбрать из списка решения и задачи (множественное выделение происходит путём зажатия клавиши Ctrl или shift) с которыми планируется работа программы и нажать кнопку «скомпилировать», или сразу перейти к следующему пункту, если компиляция уже была проведена ранее.

Далее необходимо выбрать из списка решения и задачи, с которыми планируется работа программы, или оставить старое выделение, после чего ввести информацию о максимальном ожидании результата от решения и нажать на кнопку «Проверить».

После проведения всех запланированных тестов можно подвести итог, путём нажатия на кнопку «Итог». В результате в корневой директории приложения появится два файла со статистикой.

При завершении работы выход из приложения осуществляется кнопкой «Закрыть» в верхнем правом углу программы.

3 Экономическая часть

3.1 Расчёт основной и дополнительной заработной платы с отчислением на социальное страхование

На первом этапе разработки программного продукта необходимо рассчитать смету затрат, а также определить этапы разработки программного продукта и трудоёмкость выполнения каждого этапа. Все показатели представлены в таблице 3.1.

Таблица 3.1 – Этапы и трудоёмкость выполнения работ, в часах

Этапы разработки	Трудоемкость выполнения работ	
	Руководитель	Техник
Постановка задачи	1	1
Выбор инструментальных средств	1	2
Разработка мат. модели	2	10
Построение алгоритма	2	10
Программирование	1	30
Тестирование	1	10
Отладка	1	10
Написание пояснительной записки	1	20
Итого	10	93

Месячный оклад руководителя составляет 20000 руб., месячный оклад техника составляет 25000 руб.

При расчёте основной заработной платы учитывается заработная плата всех категорий работников, непосредственно занятых разработкой программного продукта. Размер заработной платы определяется исходя из количества исполнителей и их квалификационного уровня, а также

затраченного ими времени в целом на разработку программного продукта и отдельных его этапов. Так как разработка происходила в марте, то количество рабочих часов равно 103.

Основная заработная плата руководителя составляет 20000 * 10/ 168 = 1190 руб., основная заработная плата техника составляет 25000 * 93 / 168 = 13839 руб. Сумма основной заработной платы руководителя и техника расположена в таблице 3.2.

Таблица 3.2 – Основная заработная плата работников

Должность	Заработная плата, руб.
Руководитель проекта	1190
Техник	13839
Итого	15029

Дополнительная заработная плата включает различные виды доплат сверх основной заработной платы: премии, доплату за работу в сверхурочное время, надбавки за профессиональное мастерство, оплату очередного и учебных отпусков и прочие виды доплат.

Дополнительная заработная плата устанавливается на предприятии в процентах от суммы основной заработной платы и составляет от 8 до 10%.

Таким образом, дополнительная заработная плата определяется по следующей формуле:

$$ЗП_{доп} = ЗП_{осн} * 10\% / 100\%, \quad (3.1)$$

где $ЗП_{осн}$ - основная заработная плата, руб.

Значение $ЗП_{осн}$ взято из таблицы 3.1, и с помощью формулы 3.1 высчитывается дополнительная заработная плата.

$$ЗП_{доп} = 15029 * 10\% / 100\% = 1502 \text{ руб.}$$

На основании вышеприведенных расчетов можно сделать вывод, что сумма основной и дополнительной заработной платы работников по созданию программного продукта составляет 16531 руб.

Отчисления с заработной платы на социальное страхование (во внебюджетные фонды) включают следующие виды отчислений:

- пенсионный фонд - 22%;
- фонд социального страхования - 2,9%;
- страховые взносы на травматизм – 0,2%
- фонд обязательного медицинского страхования - 5,1%.

Всего отчисления на социальное страхование составляют 30.2% от суммы основной и дополнительной заработной платы. Отчисления на социальное страхование рассчитываются по формуле 3.2.

$$ЗП_{стр} = (ЗП_{осн} + ЗП_{доп}) * 30.2\% / 100\%, \quad (3.2)$$

где $ЗП_{осн}$ - основная заработная плата, руб.;

$ЗП_{доп}$ - дополнительная заработная плата, руб.

Значение $ЗП_{осн}$ взято из таблицы 3.2, а $ЗП_{доп}$ рассчитывается по формуле 3.1. Величина отчислений определяется с помощью формулы 3.2.

$$ЗП_{стр} = (16531) * 30.2\% / 100\% = 4992 \text{ руб.} \quad (3.2)$$

Таким образом, из расчетов видно, что отчисления на социальное страхование составляют 4992 руб.

Сумма основой и дополнительной заработной платы с отчислением на социальное страхование равна 21523 руб.

3.2 Расчёт стоимости материалов и лицензионного обеспечения

Расчёт стоимости необходимых материалов для разработки программного продукта рассчитывается на основе норм расхода материальных ресурсов и оптовых цен на их приобретение по формуле 3.3.

$$С_м = Н_р * Ц, \quad (3.3)$$

где $Н_р$ - норма расхода материальных ресурсов в натуральных единицах;

Ц - цена приобретения за единицу, руб.

Расчёт стоимости материалов представлен в таблице 3.3.

Таблица 3.3 – Расчёт стоимости материалов в рублях

Наименование материала	Норма расхода	Цена за единицу	Сумма
USB Flash Kingston DataTraveler DTIG4 16 ГБ, шт.	1	399	399
Бумага, уп.	1	180	180
Папка, шт.	1	50	100
Итого			679

Общая стоимость материалов составит 679 руб.

При создании программного продукта используется лицензионное программное обеспечение, поэтому необходимо включить стоимость лицензионных программ, используемых при разработке программного продукта, в смету затрат.

Перечень лицензионных программ, норма установки и цена расположены в таблице 3.4.

Таблица 3.4 – Перечень программного обеспечения для выполнения работ

Наименование лицензионных программ	Норма установки, шт.	Цена, руб.
Microsoft Windows 10	1	9197
Visual Studio	1	3223
Microsoft Office	1	716
Photoshop CS6	1	1932
Итого		15068

Стоимость лицензионного программного обеспечения для разработки программного продукта составит 15068 руб.

Расходы на эксплуатацию составляют:

- основная и дополнительная заработная плата;
- отчисления на социальное страхование;
- стоимость материалов;
- стоимость лицензионного обеспечения.

В итоге расходы на эксплуатацию равны 37270 руб.

3.3 Расчёт накладных расходов

Накладные расходы представляют собой дополнительные к основным расходам затраты на управление, организацию и обслуживание производства.

Не связаны напрямую с основным производством товаров или предоставлением услуг, не входят в стоимость материалов и оплату труда.

Накладные расходы закладываются в себестоимость товара, издержки его производства и обращения, но не прямо, а косвенно, пропорционально стоимости материалов и сырья, сумме заработной платы и так далее.

Накладные расходы составляют 10% от расходов на эксплуатацию, и рассчитываются по формуле 3.4.

$$Н_р = (Р_{эксп} * 10\%) / 100\%, \quad (3.4)$$

где $Р_{эксп}$ – величина расходов на эксплуатацию, руб.

$$Н_р = (37270 * 10\%) / 100\% = 3727 \text{ руб.}$$

3.4 Составление и расчёт цены реализации программного продукта

На основании выше рассчитанных данных составлена смета затрат на программный продукт. Результат расположен в таблице 3.5.

Таблица 3.5 – Смета затрат на программный продукт

Статьи затрат	Сумма, руб.	Структура затрат, %
Основная и дополнительная заработная плата	16531	40,3
Отчисления на социальное страхование	4992	12,2
Стоимость материалов	679	1,66
Стоимость лицензионного обеспечения	15068	36,75
Накладные расходы	3727	9,09
Итого	40997	100

Таким образом, общая сумма затрат на разработку программного продукта составляет 40997 руб.

Под структурой затрат понимается удельный вес (в процентах) каждой статьи в общей величине затрат. По результатам расчета построена диаграмма структуры затрат на программный продукт и представлена на рисунке Д.1.

Цена является одним из самых важных экономических показателей. Её основная функция состоит в обеспечении выручки от реализации программного продукта, поэтому цена определяет прибыль и финансовую стабильность предприятия, его жизнеспособность.

Так как целью предпринимательской деятельности является получение прибыли, то цена реализации программного продукта будет представлять собой сумму затрат на его разработку и запланированную величину прибыли. Коэффициент рентабельности проекта равен 30%. Величина общих затрат на разработку программного продукта взята из таблицы 3.5, а стоимость материалов из таблицы 3.3.

Расчёт оптовой цены производится по формуле 3.5.

$$Ц_{опт} = З * (1+Р), \quad (3.5)$$

где З – общие затраты на разработку программного продукта, руб.;

Р – уровень рентабельности проекта, коэффициент.

Произведём расчет оптовой цены используя формулу 3.5.

$$Ц_{опт} = 40997 * (1 + 0,3) = 53296 \text{ руб.}$$

Расчёт прибыли от продажи продукта производится по формуле 3.6.

$$П = Ц_{опт} - З, \quad (3.6)$$

$$П = 53296 – 40997 = 12299 \text{ руб.}$$

Таким образом, можно сделать вывод, что данный продукт прибыльный и может быть реализован. Прибыль от продажи программного продукта составит 12299 руб.

Заключение

В ходе дипломного проектирования разработан программный продукт, соответствующий требованиям индивидуального задания.

Работы, выполненные в ходе проектирования, и разработанный программный продукт описаны в настоящей пояснительной записке с соблюдением требований ГОСТ ЕСПД.

Таким образом, задание на выпускную квалификационную работу выполнено в полном объёме.

Список литературы

1 ГОСТ: Единая система программной документации. М,: Госстандарт России: Изд-во стандартов, 2011, или более позднее официальное издание.

2 Архитектура автоматизированной системы тестирования решений задач по программированию [Электронный ресурс] - https://cyberleninka.ru/article/n/arhitektura-avtomatizirovannoy-sistemy-testirovaniya-resheniy-zadach-po-programmirovaniyu (Дата обращения 02.05.2020).

3 В чем набрать и чем собрать C++ проект [Электронный ресурс] - https://habr.com/ru/post/442682/ (Дата обращения 02.05.2020).

4 Документация C# Класс Process [Электронный ресурс] - https://docs.microsoft.com/ru-ru/dotnet/api/system.diagnostics.process (Дата обращения 03.05.2020)

5 Документация SQLite [Электронный ресурс] - https://www.sqlite.org/docs.html (Дата обращения 03.05.2020).

6 Пошаговое руководство. Компиляция собственной программы на языке C++ из командной строки [Электронный ресурс] - https://docs.microsoft.com/ru-ru/cpp/build/walkthrough-compiling-a-native-cpp-program-on-the-command-line (Дата обращения 02.05.2020).

Приложение А

(обязательное)

Рисунок А.1 – Функциональная модель проведения олимпиады

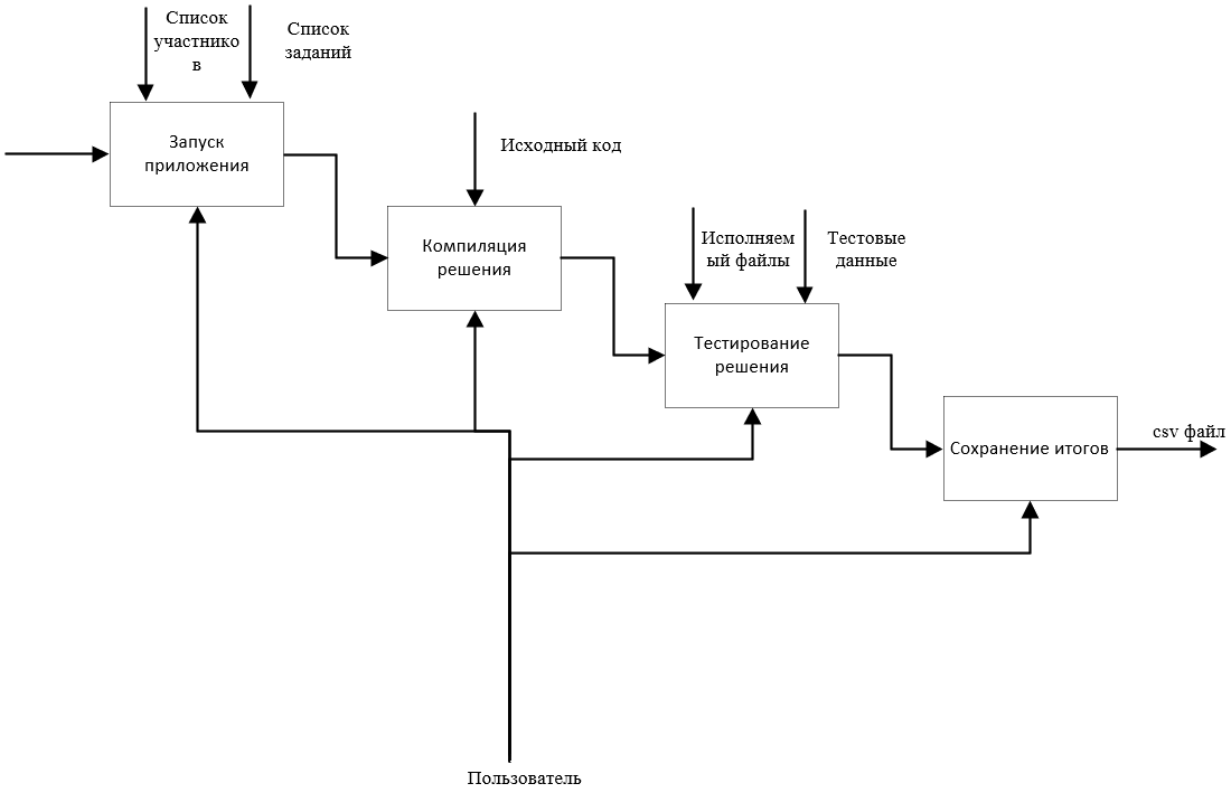

Рисунок А.2 – Характеристика бизнес-процессов

Приложение Б

(обязательное)

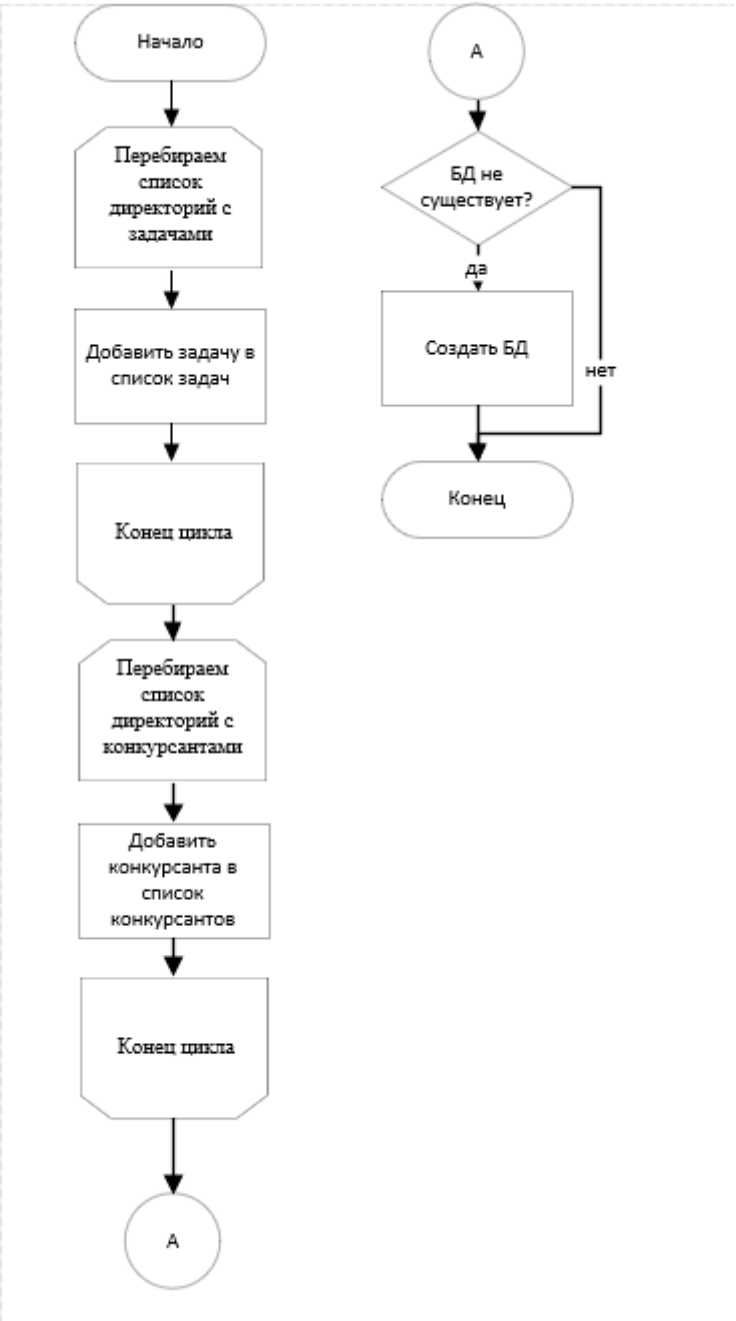

Рисунок Б.1 – Схема алгоритма запуска программы

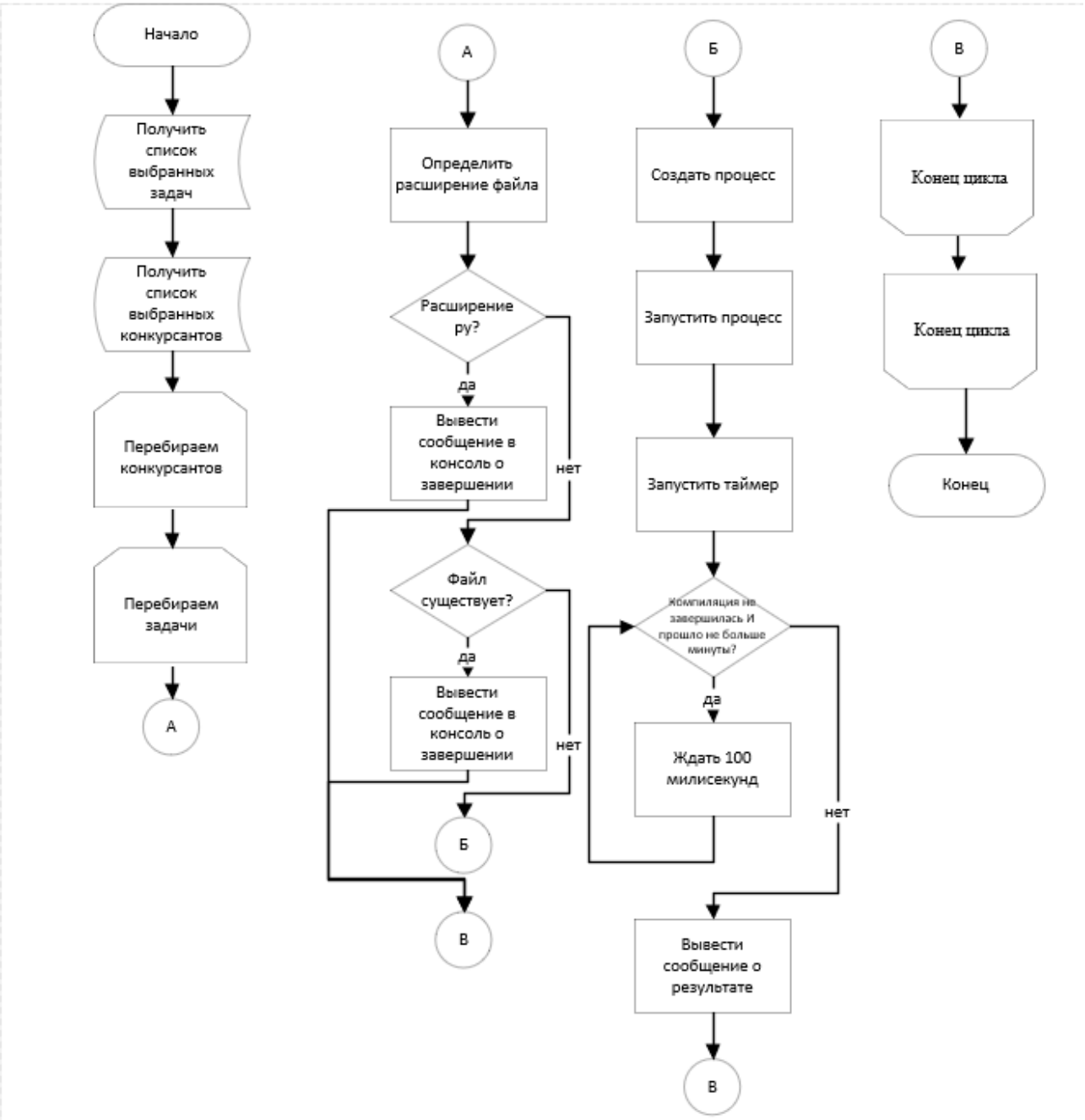

Рисунок Б.2 – Схема алгоритма компиляции

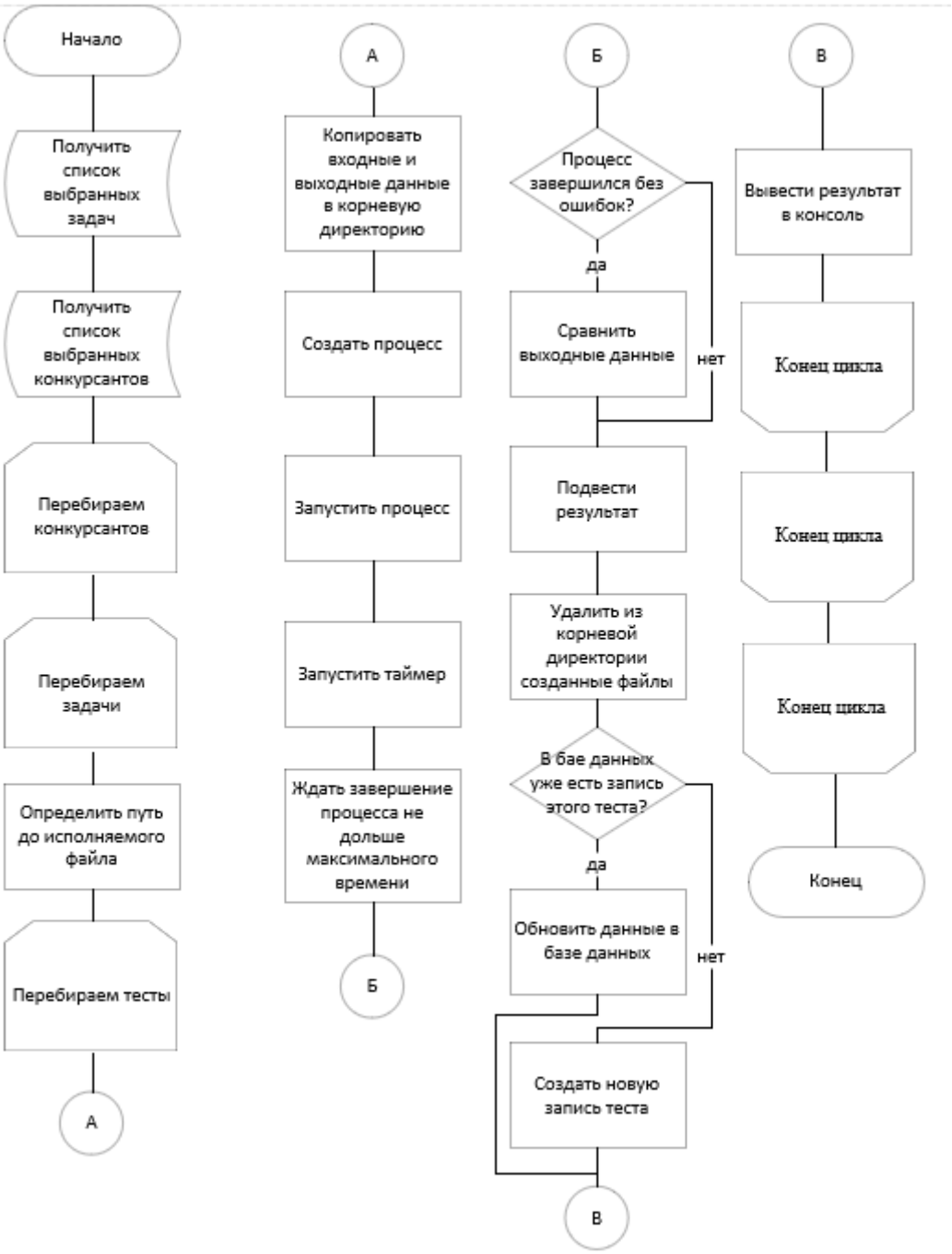

Рисунок Б.3 – Схема алгоритма тестирования

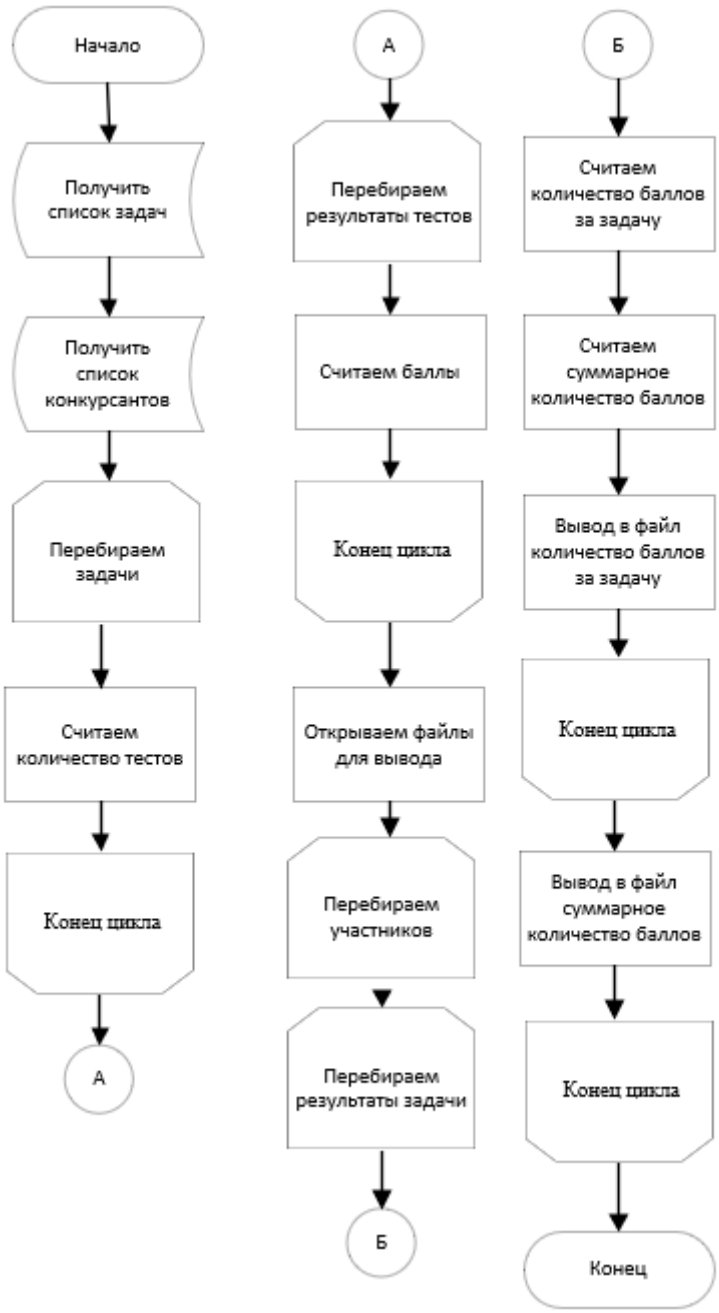

Рисунок Б.4 – Схема алгоритма вывода итогов

Приложение В

(обязательное)

В.1 – Обработчик события завершения загрузки формы

```
private void Form1_Load(object sender, EventArgs e)
        {
            try
            {
                foreach (var folder in Directory.GetDirectories(tasksPath))
                    taskList.Items.Add(Path.GetFileName(folder));
                if(taskList.Items.Count == 0)
                    taskList.Items.Add("Тесты отсутсвуют");
            }
            catch (Exception ex)
            {
                taskList.Items.Add("Директория не найдена");
                ConsoleWrite(ex.Message);
            }
            try
            {
                foreach (var folder in
Directory.GetDirectories(contestantsPath))
                    contestantList.Items.Add(Path.GetFileName(folder));
                if (contestantList.Items.Count == 0)
                    contestantList.Items.Add("Конкурсанты отсутсвуют");
            }
            catch (Exception ex)
            {
                contestantList.Items.Add("Директория не найдена");
                ConsoleWrite(ex.Message);
            }

            var col = db.GetCollection<Contest>("Contest");
                if (col.FindOne(Query.EQ("Name", "Contest")) == null)
                {
                    var ct = new Contest { Name = "Contest" };
                    ct.Contestants = new List<Contestant>();
                    col.Insert(ct);
                }
        }
```

В.2 – Обработчик события нажатия на кнопку «Скомпилировать»

```
private void compilate(object sender, EventArgs e)
        {
            try
            {
                List<string> tasks =
taskList.SelectedItems.OfType<string>().ToList();
                List<string> contestants =
contestantList.SelectedItems.OfType<string>().ToList();
                foreach (string contestant in contestants)
                {
                    foreach (string task in tasks)
                    {
                        bool fail = true;
                        int secToCompil = 60;
                        String taskPath = contestantsPath + "/" + contestant
+ "/";
                        String extention = "";
                        if (File.Exists(taskPath + "/" + task + ".cpp"))
                        {
                            extention = ".cpp";
                        }
                        else if (File.Exists(taskPath + "/" + task + ".c"))
                        {
                            extention = ".c";
                        }
                        else if (File.Exists(taskPath + "/" + task + ".py"))
                        {
                            ConsoleWrite(contestant + " " + task + " " +
"python file ready");
                            continue;
                        }
                        else
                        {
                            ConsoleWrite(contestant + " " + task + " " +
"file missing");
                            continue;
                        }
                        ConsoleWrite(contestant + " " + task + " " +
"compilation started");
                        File.Delete(taskPath + task + ".exe");
                        File.Delete(taskPath + task + ".obj");

                        Process compiler = new Process();
                        compiler.StartInfo.FileName = "cmd";
                        compiler.StartInfo.CreateNoWindow =
!checkBox1.Checked;
                        compiler.StartInfo.UseShellExecute = false;
                        compiler.StartInfo.Arguments = "/k \"C:/Program Files
(x86)/Microsoft Visual Studio/2019/BuildTools/Common7/Tools/VsDevCmd.bat\"" +
                            "&& cd " + taskPath +
                            "&& cl /EHsc " + task + extention;
                        compiler.Start();

                        Stopwatch timmer = new Stopwatch();
                        timmer.Start();
                        for (int i = 0; i < secToCompil * 10; i++)
                        {
                            if (File.Exists(taskPath + "/" + task + ".exe"))
{ fail = false; break; };
                            Thread.Sleep(100);
                        }
                        compiler.Kill();
```

```
                        if (fail) ConsoleWrite(contestant + " " + task + " "
+ timmer.ElapsedMilliseconds + "ms compilation error");
                        else ConsoleWrite(contestant + " " + task + " " +
timmer.ElapsedMilliseconds + "ms compilation finished");
                    }

                }
            }
            catch (Exception ex)
            {
                ConsoleWrite(ex.Message);
            }
        }
```

В.3 – Обработчик события нажатия на кнопку «Проверить»

```
private void test(object sender, EventArgs e)
        {
            List<string> tasks =
taskList.SelectedItems.OfType<string>().ToList();
            List<string> contestants =
contestantList.SelectedItems.OfType<string>().ToList();
            int maxTestTime = Int32.Parse(textBox1.Text) * 1000;
            var col = db.GetCollection<Contest>("Contest");
            try
            {
                foreach (string contestant in contestants)
                {
                    foreach (string task in tasks)
                    {
                        String exePath = "";
                        if (File.Exists(contestantsPath + "/" + contestant +
"/" + task + ".exe"))
                            exePath = contestantsPath + "/" + contestant +
"/" + task + ".exe";
                        else if(File.Exists(contestantsPath + "/" +
contestant + "/" + task + ".py"))
                            exePath = contestantsPath + "/" + contestant +
"/" + task + ".py";
                        else
                        {
                            ConsoleWrite(contestant + " " + task + " " +
"file missing");
                            continue;
                        }
                        foreach (String testId in
Directory.GetFiles(tasksPath + "/" + task, "*.in"))
                        {
                            String result;
                            File.Copy(testId, "input.txt", true);
                            File.Copy(testId.Replace("in", "out"),
"answer.txt", true);
                            Process exe = new Process();

                            exe.StartInfo.FileName = exePath;
                            exe.StartInfo.CreateNoWindow =
!checkBox1.Checked;
                            exe.StartInfo.UseShellExecute = false;
                            exe.Start();

                            Stopwatch timmer = new Stopwatch();
                            timmer.Start();
                            if (!exe.WaitForExit(maxTestTime))
                            {
                                timmer.Stop();
                                if (exe.HasExited)
                                {
                                    result = "Time error";
                                }
                                else
                                {
                                    result = "Runtime error";
                                }
                                exe.Kill();
                            }
```

```
                            else if (ComputeMD5Checksum("answer.txt") ==
ComputeMD5Checksum("output.txt"))
                            {
                                timmer.Stop();
                                result = "Done";
                            }
                            else
                            {
                                timmer.Stop();
                                result = "Wrong answer";
                            }

                            File.Delete("input.txt");
                            File.Delete("output.txt");
                            File.Delete("answer.txt");

                            var ct = col.FindOne(Query.EQ("Name",
"Contest"));
                            var results = ct.Contestants.Find(
                                x => x.Name.Equals(contestant) &&
x.Task.Equals(task) && x.TestId.Equals(Path.GetFileName(testId))
                                );
                            if (results == null)
                            {
                                ct.Contestants.Add(
                                new Contestant
                                {
                                    Name = contestant,
                                    Task = task,
                                    TestId = Path.GetFileName(testId),
                                    Result = result
                                });
                            }
                            else
                            {
                                results.Result = result;
                            }
                            col.Update(ct);
                            ConsoleWrite(contestant + " " + task + " " +
Path.GetFileName(testId) +
                                " " + timmer.ElapsedMilliseconds + "ms " +
result);
                        }
                    }
                }
            }
            catch (Exception ex)
            {
                ConsoleWrite(ex.Message);
            }
        }
```

В.4 – Обработчик события нажатия на кнопку «Итог»

```
private void show(object sender, EventArgs e)
        {
            try
            {
                var col = db.GetCollection<Contest>("Contest");
                var ct = col.FindOne(Query.EQ("Name", "Contest"));
                Dictionary<string, int> testCount = new Dictionary<string,
int>();
                Dictionary<string, Dictionary<string, int>> result = new
Dictionary<string, Dictionary<string, int>>();

                foreach (var folder in Directory.GetDirectories(tasksPath))
                {
                    testCount.Add(Path.GetFileName(folder),
Directory.GetFiles(folder).Length / 2);
                }
                foreach (string contestant in contestantList.Items)
                {
                    result.Add(contestant, new Dictionary<string, int>());
                    foreach(string task in taskList.Items)
                    {
                        result[contestant].Add(task, 0);
                    }
                }

                foreach (Contestant c in ct.Contestants)
                {
                    if (c.Result == "Done")
                    {
                        result[c.Name][c.Task]++;
                    }
                    ConsoleWrite(c.Name + " " + c.Task + " " + c.TestId + " "
+ c.Result);
                }
                StreamWriter swResLog = new StreamWriter("resutLog.csv",
false, System.Text.Encoding.Default);
                StreamWriter swRes = new StreamWriter("resut.csv", false,
System.Text.Encoding.Default);

                //result.OrderBy(x => x.Key);
                foreach (var contestant in result)
                {
                    float pointsSum = 0;
                    foreach(var task in contestant.Value)
                    {
                        float points = (float)task.Value /
(float)testCount[task.Key] * 100;
                        pointsSum += points;
                        swResLog.WriteLine(contestant.Key + ";" + task.Key +
";" + points);
                    }
                    swRes.WriteLine(contestant.Key + ";" + pointsSum);
                }
                swResLog.Close();
                swRes.Close();

            }
            catch (Exception ex)
            {
                ConsoleWrite(ex.Message);
            }
        }
```

В.5 – Метод вывода в консоль

```
private void ConsoleWrite(string text)
        {
            const int consoleLenght = 80;
            if (text.Length < consoleLenght)
            {
                logList.Items.Add(DateTime.Now + " " + text);
                logList.SelectedIndex = logList.Items.Count - 1;
                logList.ClearSelected();
            }
            else
            {
                logList.Items.Add(text.Substring(0, consoleLenght));
                ConsoleWrite(text.Substring(consoleLenght));
            }
        }
```

В.6 – Метод получения контрольной суммы файла по алгоритму md5

```
private string ComputeMD5Checksum(string path)
        {
            using (FileStream fs = File.OpenRead(path))
            {
                MD5 md5 = new MD5CryptoServiceProvider();
                byte[] fileData = new byte[fs.Length];
                fs.Read(fileData, 0, (int)fs.Length);
                byte[] checkSum = md5.ComputeHash(fileData);
                string result = BitConverter.ToString(checkSum).Replace("-",
String.Empty);
                return result;
            }
        }
```

Приложение Г

(обязательное)

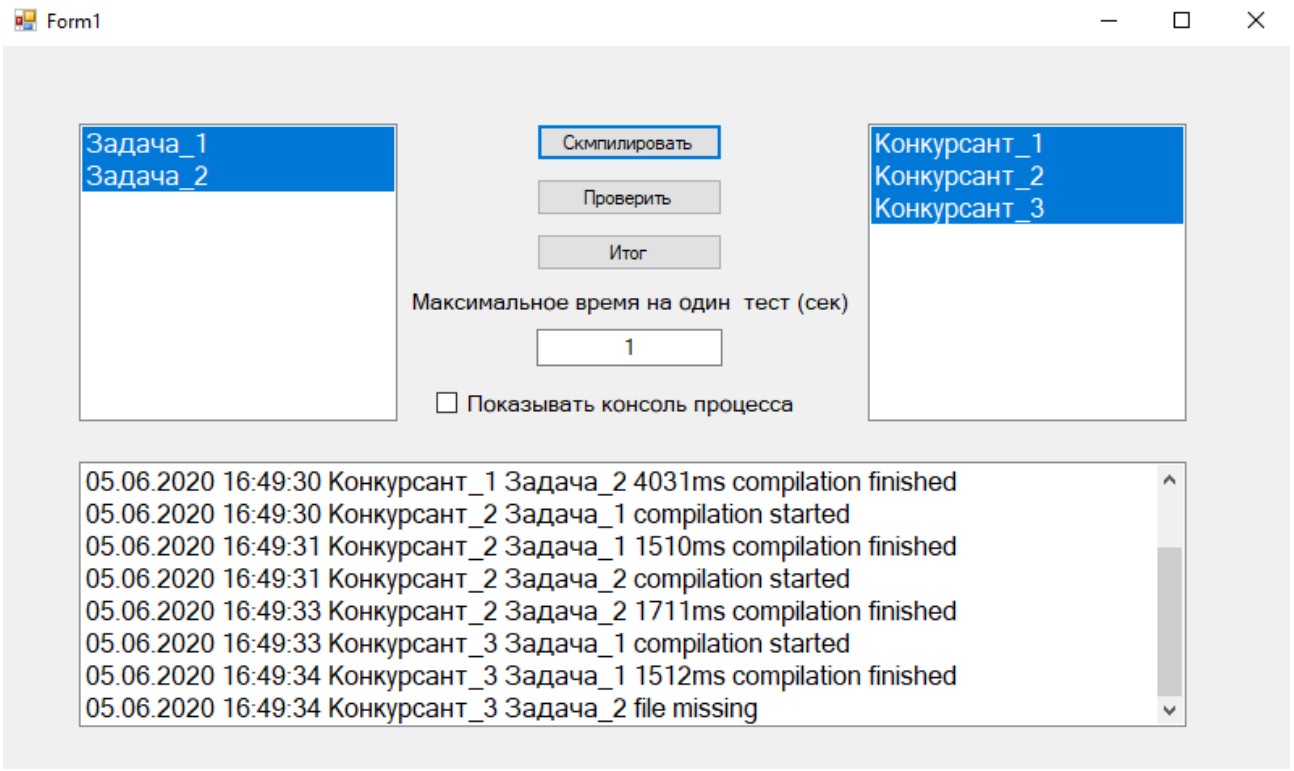

Рисунок Г.1 – Компиляция решений

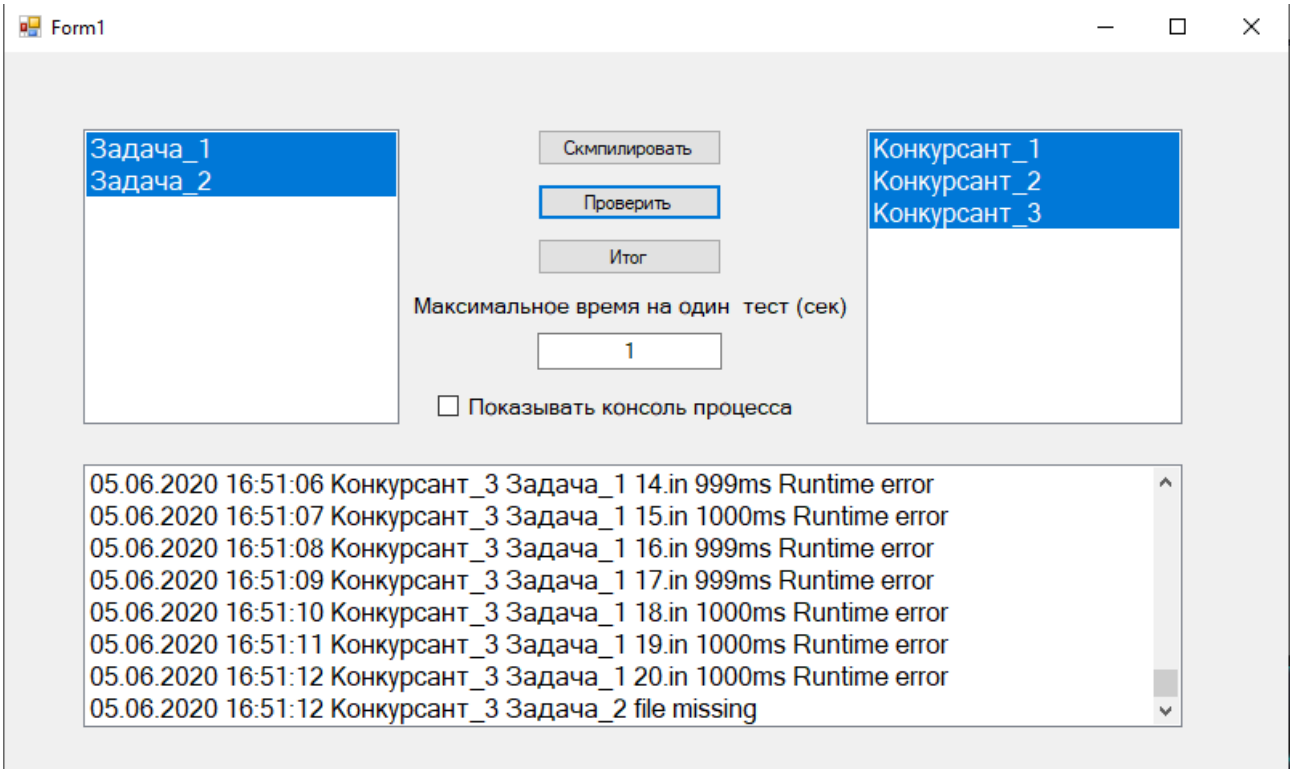

Рисунок Г.2 – Проверка решений по тестам

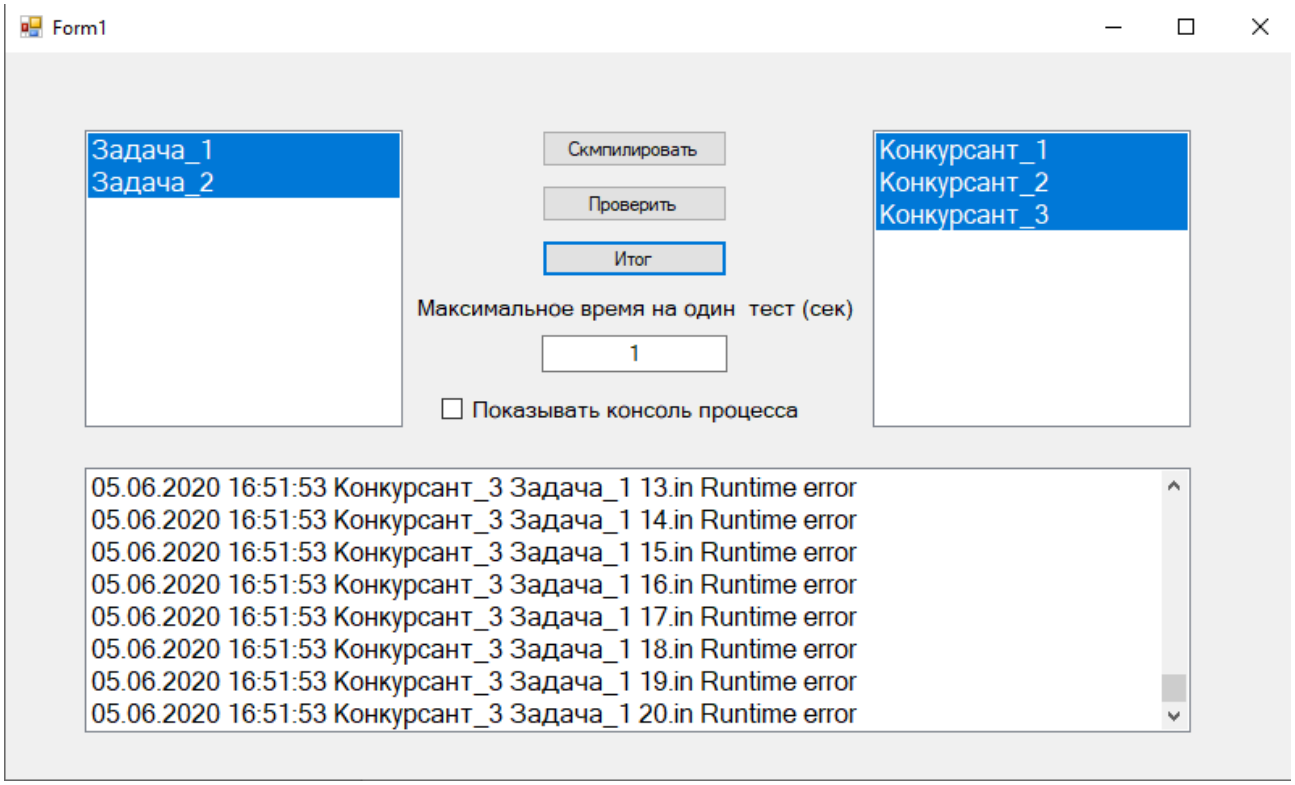

Рисунок Г.3 – Вывод итогов

	A	B	C
1	Конкурсант_1	100	
2	Конкурсант_2	100	
3	Конкурсант_3	0	
4			

Рисунок Г.4 – Данные файла result.csv

	A	B	C	D
1	Конкурсант_1	Задача_1	0	
2	Конкурсант_1	Задача_2	100	
3	Конкурсант_2	Задача_1	0	
4	Конкурсант_2	Задача_2	100	
5	Конкурсант_3	Задача_1	0	
6	Конкурсант_3	Задача_2	0	

Рисунок Г.5 – Результат файла resultLog.csv

Приложение Д

(обязательное)

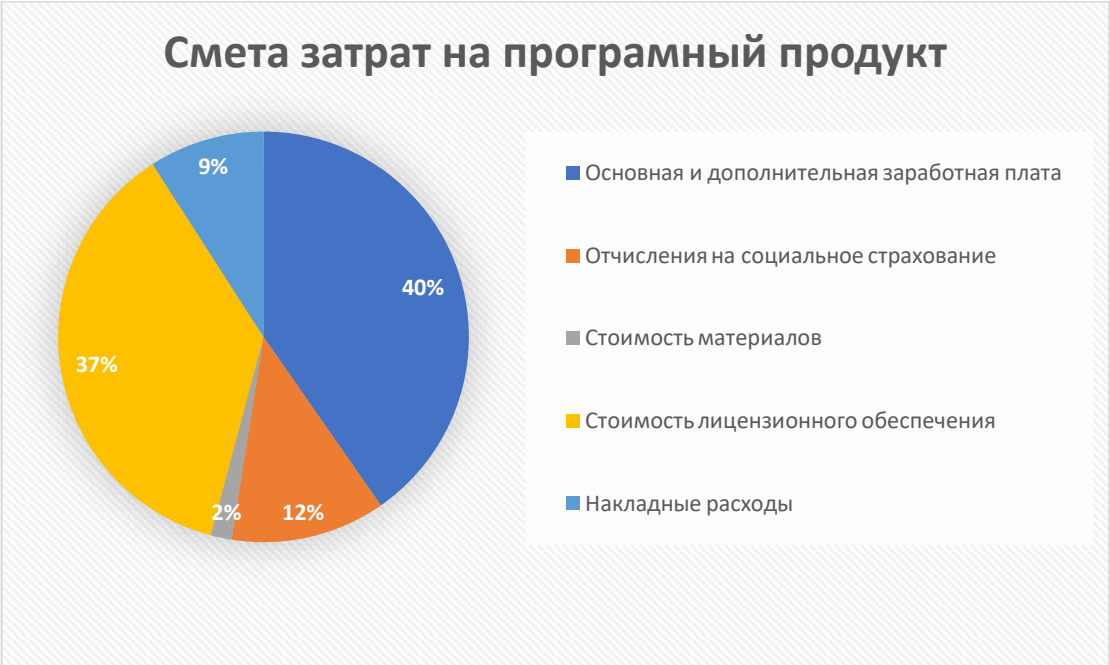

Рисунок Д.1 – Диаграмма сметы затрат на программный продукт