**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ**

**УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ ИМ. ПРОФ. М.А. БОНЧ-БРУЕВИЧА»**

**(СПбГУТ)**

**АРХАНГЕЛЬСКИЙ КОЛЛЕДЖ ТЕЛЕКОММУНИКАЦИЙ**

**ИМ. Б.Л. РОЗИНГА (ФИЛИАЛ) СПбГУТ**

**(АКТ (ф) СПбГУТ)**

**КУРСОВОЙ ПРОЕКТ**

### НА ТЕМУ

|  |  |
| --- | --- |
| РАЗРАБОТКА ОБУЧАЮЩЕЙ ПРОГРАММЫ | |
| «ОСНОВЫ РАБОТЫ | |
| В ПРОГРАММНОМ ОБЕСПЕЧЕНИИ | |
| АВТОГРАФ» | |
| Л309. 22КП02. 023 ПЗ | |

(Обозначение документа)

|  |
| --- |
| МДК.03.01 Технология разработки программного |
| обеспечения |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ПКС-91 | |  | Н.А. Шефов |
|  | (Группа) | (Подпись) | (Дата) | (И.О. Фамилия) |
| Преподаватель |  |  |  | С.В. Лукина |
|  |  | (Подпись) | (Дата) | (И.О. Фамилия) |

Архангельск 2022

Содержание

[Содержание 2](#_Toc122773368)

[Перечень сокращений и обозначений 3](#_Toc122773369)

[Введение 4](#_Toc122773370)

[1 Анализ и разработка требований 6](#_Toc122773371)

[1.1 Назначение и область применения 6](#_Toc122773372)

[1.2 Постановка задачи 6](#_Toc122773373)

[1.3 Описание алгоритма функционирования системы 7](#_Toc122773374)

[1.4 Выбор состава программных и технических средств 7](#_Toc122773375)

[2 Проектирование и разработка 9](#_Toc122773376)

[2.1 Проектирование причинно-следственной диаграммы 9](#_Toc122773377)

[2.2 Проектирование DFD-диаграммы первого и второго уровня 10](#_Toc122773378)

[2.3 Проектирование UML-диаграммы стереотипов и классов 12](#_Toc122773379)

[2.4 Проектирование прототипа разрабатываемой программы 14](#_Toc122773380)

[2.5 Создание обучающей программы «Основы работы в программном обеспечении Автограф» 16](#_Toc122773381)

[3 Тестирование разработанного продукта 22](#_Toc122773382)

[3.1 Тестирование методом «черного ящика» 22](#_Toc122773383)

[4 Руководство пользователя 24](#_Toc122773384)

[4.1 Установка приложения 24](#_Toc122773385)

[4.2 Инструкция по работе 24](#_Toc122773386)

[Заключение 29](#_Toc122773387)

[Список использованных источников 30](#_Toc122773388)

[Приложение А (обязательное) Программный документ «Техническое задание» 31](#_Toc122773389)

Перечень сокращений и обозначений

В настоящем курсовом проекте применяют следующие сокращения и обозначения:

ОС – операционная система

ПК – персональный компьютер

ПО – программное обеспечение

ПП – программный продукт

DFD – диаграмма потоков данных

HTML – язык гипертекстовой разметки

UML – унифицированный язык моделирования

Введение

Использование компьютерных технологий в процессе обучения становится актуальнее, позволяя сэкономить время и ресурсы, затраченные на обучение. Обучающие программы внедряются в различных сферах для работников на предприятиях и обучающихся в образовательных учреждениях.

Множеству специалистов от инженеров до программистов требуется составлять схемы и диаграммы. Для решения этих задач они могут использовать высокопроизводительный Российский графический редактор «Автограф». И для того, чтобы разобраться в данной программе и проверить свои навыки работы в ней, спроектирована обучающая программа «Основы работы в программном обеспечении Автограф».

Обучающая программа «Основы работы в программном обеспечении Автограф» – необходимое и эффективное решение для повышения навыков работы в программе «Автограф». Возможность одновременного самостоятельного обучения сотрудников, а также автоматизированной проверки знаний делает использование обучающей программы оптимальным решением.

Целью курсового проектирования является разработка однопользовательской клиентской обучающей программы «Основы работы в программном обеспечении Автограф».

Для достижения поставленной цели требуется решить следующие задачи:

* выполнить сбор требований к обучающей программе,
* проанализировать собранные требования,
* выполнить сбор теоретического материала по поставленной теме,
* выбрать и обосновать выбор средств разработки ПП,
* описать спецификацию разрабатываемого ПП,
* спроектировать причинно-следственную диаграмму,
* спроектировать UML-диаграмму стереотипов,
* спроектировать DFD-диаграммы первого и второго уровня,
* спроектировать прототип ПП,
* спроектировать ПП,
* разработать ПП,
* подобрать теоретический материал,
* подобрать вопросы для тестирования,
* разработать руководство по эксплуатации,
* провести тестирование разработанного ПП.

1. Анализ и разработка требований
   1. Назначение и область применения

Разрабатываемый ПП является обучающей программой, предназначенной для обучения и тестирования по работе в программе «Автограф».

Пользователями обучающей программы являются любые пользователи, желающие повысить свой навык работы в программе «Автограф».

* 1. Постановка задачи

Требуется разработать однопользовательскую клиентскую обучающую программу с оконным интерфейсом пользователя для организации обучения и тестирования по охране труда и пожарной безопасности.

ПП должен обеспечивать выполнение следующих задач:

* предоставление теоретического материала по заданной теме, поделенного на разделы,
* организация тестирования по предоставленному теоретическому материалу,
* вывод результатов тестирования.

При тестировании должны соблюдаться следующие требования:

* в тестах содержится три типа вопросов: с одиночным выбором, с множественным выбором и со свободным вводом,
* на каждый вопрос предоставляется определенное время на ответ,
* на разные типы вопросов предоставляется разное время,
* банк вопросов должен содержать набор вопросов, из которых только часть используется при тестировании.
  1. Описание алгоритма функционирования системы

После запуска приложения пользователю открывается страница с теоретическим материалом, на котором присутствуют кнопки перехода к окну «О программе», теоретическому материалу и к прохождению тестов.

При нажатии на кнопку «Теоретический материал» пользователь переходит к странице, содержащей теоретический материал. Нажимая на пункты содержания, пользователь может открывать и просматривать нужные ему разделы.

При нажатии на кнопку с наименованием теста пользователь переходит к странице с соответствующим тестом. Нажатием кнопки «Начать» пользователь запускает выбранный тест. Перемещение по вопросам происходит через кнопку «Далее» в нижней части окна вопроса, которая появляется после ответа на вопрос или по истечению времени, отведенному на ответ.

Время ответа на каждый вопрос теста ограничено. При истечении данного на ответ времени вопрос автоматически засчитывает текущий выбранный ответ, если ответ отсутствует, то баллы за вопрос не засчитываются.

После ответа на последний вопрос теста отображается количество набранных баллов за тест и полученная оценка.

* 1. Выбор состава программных и технических средств

Согласно цели проекта требуется разработать однопользовательскую клиентскую обучающую программу с оконным интерфейсом пользователя для организации обучения и тестирования по охране труда и пожарной безопасности.

Работа с ПП будет осуществляться на персональных компьютерах с установленной ОС Windows версии не ниже Windows 7.

Приложение будет написано на языке программирования C#, так как C# – это объектно-ориентированный язык программирования, который позволяет разработчикам создавать многофункциональные приложения и из-за возможности работать с платформой .Net.

Для создания интерфейса приложения будет использоваться платформа WPF, так как она предоставляет широкий спектр возможностей по настройке внешнего вида приложения.

Для разработки приложения будет использоваться интегрированная среда разработки программ Microsoft Visual Studio Community 2022, т.к. она позволяет работать с языком программирования C#. Microsoft Visual Studio Community 2022 имеет продвинутую систему IntelliCode и производительна.

Для функционирования системы на ПК пользователя достаточны следующие программные и технические средства:

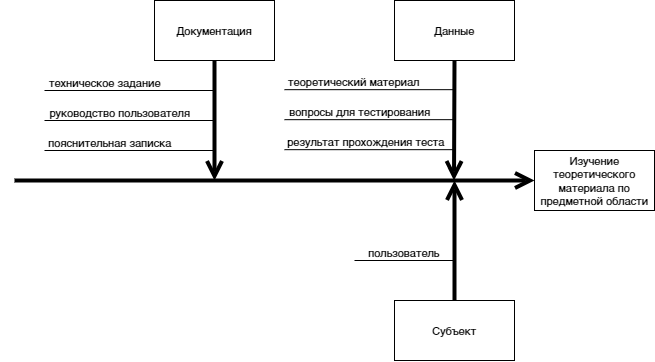
* процессор: 1 ГГц,
* ОЗУ: 512 МБ,
* жесткий диск: минимум 4,5 ГБ свободного места на диске,

- ОС не ниже Windows 10,

- .Net Framework 4.8.

1. Проектирование и разработка
   1. Проектирование причинно-следственной диаграммы

Причинно-следственная диаграмма – это графический способ, позволяющий выявить наиболее существенные причины, влияющие на конечный результат. Необходимо составить причинно-следственную диаграмму для наглядного отображения функционала разрабатываемого продукта и лучшего понимания исследуемого процесса. На рисунке 1 представлена причинно-следственная диаграмма обучающей программы «Основы работы в программном обеспечении Автограф».



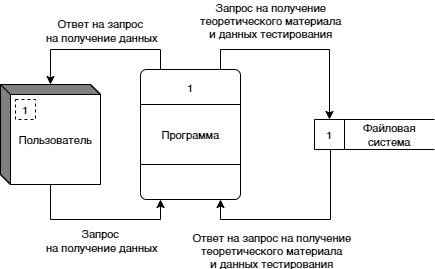
1. Причинно-следственная диаграмма

Диаграмма показывает, что разрабатываемая обучающая программа предусматривает только пользовательскую роль. Для описания программы предусмотрена пояснительная записка, техническое задание и руководство пользователя. Основной проблемой разрабатываемой обучающей программы является изучение теоретического материала по предметной области. Данная проблема изображена на диаграмме длинной стрелкой, а остальные стрелки, примыкающие к основной, усугубляют проблему.

* 1. Проектирование DFD-диаграммы первого и второго уровня

DFD-диаграммы представляют собой иерархию функциональных процессов, связанных потоками данных. Такая диаграмма демонстрирует, как каждый процесс преобразует свои входные данные в выходные, а также выявляет отношения между этими процессами[2].

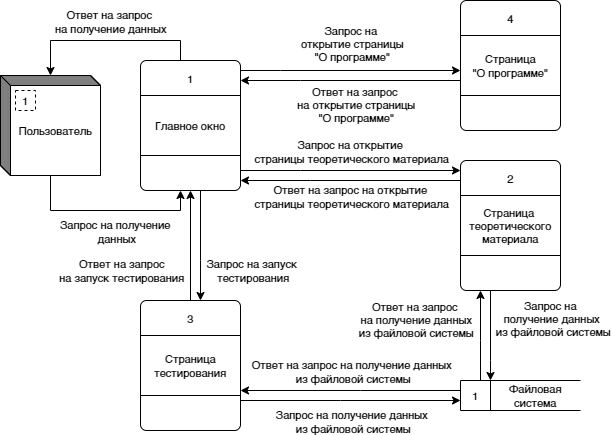
В ходе разработки были спроектированы DFD-диаграммы первого и второго уровня. DFD-диаграмма первого уровня обучающей программы «Основы работы в программном обеспечении Автограф» представлена на рисунке 2.



1. DFD-диаграмма первого уровня

На DFD-диаграмме представлено, как пользователь при взаимодействии с программой отправляет запрос на получение теоретического материала и данных тестирования из файловой системы. После этого программа передает этот запрос в файловую систему, где произойдет обработка запроса и сформирование ответа на запрос о получении теоретического материала и данных тестирования который будет выведен пользователю в приложении.

DFD-диаграмма второго уровня обучающей программы «Основы работы в программном обеспечении Автограф» представлена на рисунке 3.



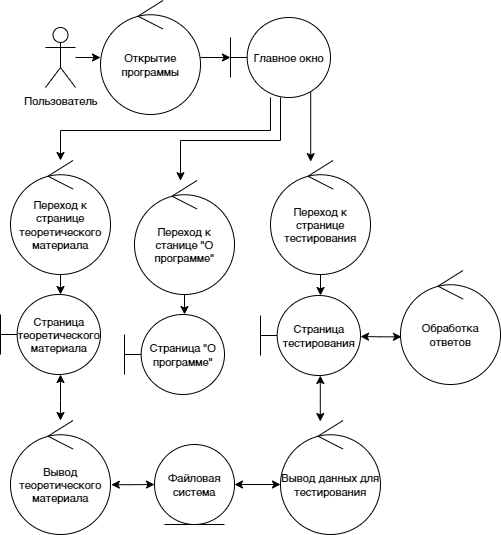
1. DFD-диаграмма второго уровня

На DFD-диаграмме второго уровня продемонстрировано взаимодействие пользователя с основными элементами интерфейса программы: главное окно, страница теоретического материала, тестирования и «О программе». Пользователь получает тесты и теоретический материал, передавая запрос к файловой системе через страницу теоретического материала и страницу тестирования соответственно.

* 1. Проектирование UML-диаграммы стереотипов и классов

UML**-**диаграмма – это система обозначений, которую можно применять для объектно-ориентированного анализа и проектирования. Его можно использовать для визуализации, спецификации, конструирования и документирования программных систем.

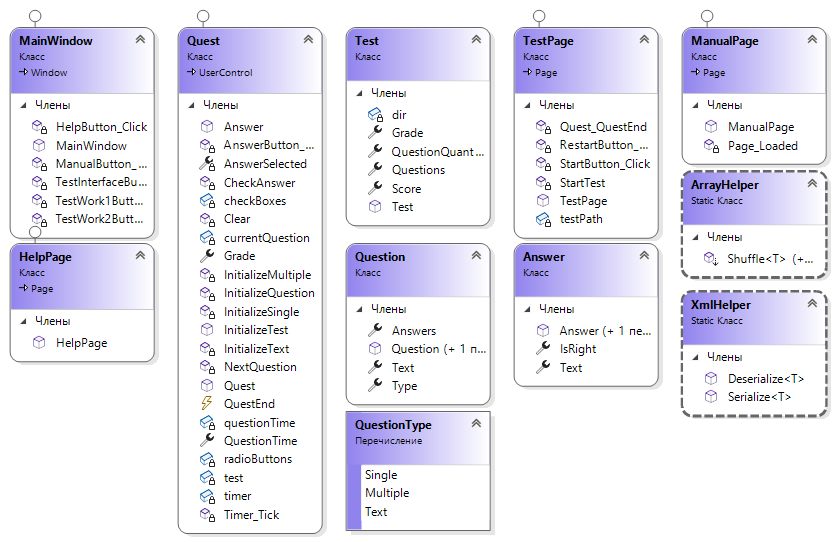
Во время работы спроектированы UML-диаграммы стереотипов и классов. UML-диаграмма стереотипов обучающей программы «Основы работы в программном обеспечении Автограф» представлена на рисунке 4.



1. Диаграмма стереотипов

На UML-диаграмме представлены последовательные действия, которые может совершить пользователь. При открытии программы пользователя встречает главное окно, с открытой страницей теоретического материала. В боковом меню он может открыть страницу «О программе», теоретического материала или тестирования. При переходе к странице теоретического материала или к странице тестирования происходит процесс вывода данных, который обращается к файловой системе для получения данных и отображения их пользователю.

Диаграмма классов – структурная диаграмма языка моделирования UML, демонстрирующая общую структуру иерархии классов, их коопераций, атрибутов (полей), методов, интерфейсов и взаимосвязей (отношений) между ними. Широко применяется не только для документирования и визуализации, но также для конструирования посредством прямого или обратного проектирования. Диаграмма классов обучающей программы «Основы работы в программном обеспечении Автограф» представлена на рисунке 5.

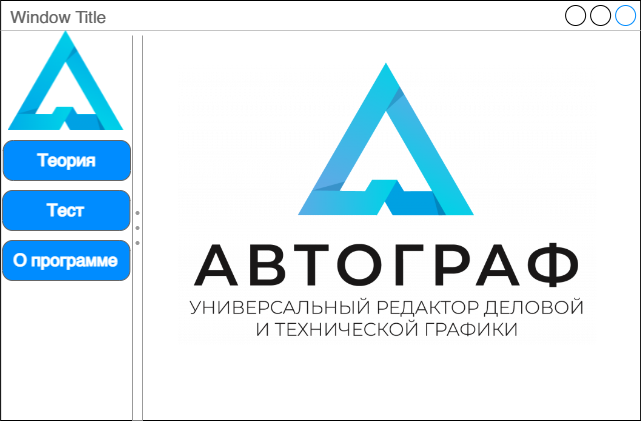


1. Диаграмма классов
   1. Проектирование прототипа разрабатываемой программы

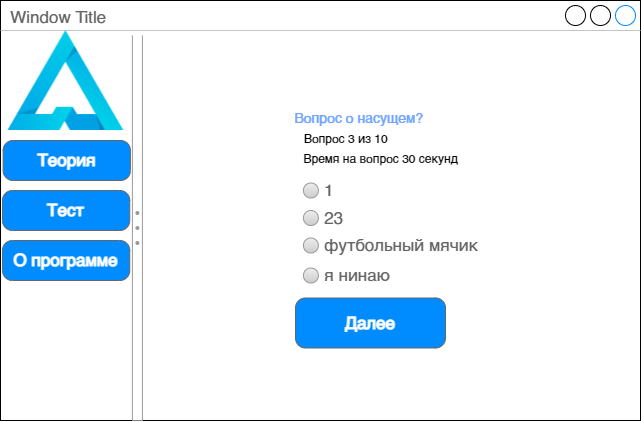
Прототип – это модель, прообраз конечного продукта. Прототипы различаются по степени точности и приближенности к реальному продукту. Кроме того, разные виды прототипов служат разным целям и способны решать разные задачи. По стадии готовности их можно условно поделить на 3 этапа:

* концептуальные,
* интерактивные,
* анимированные.

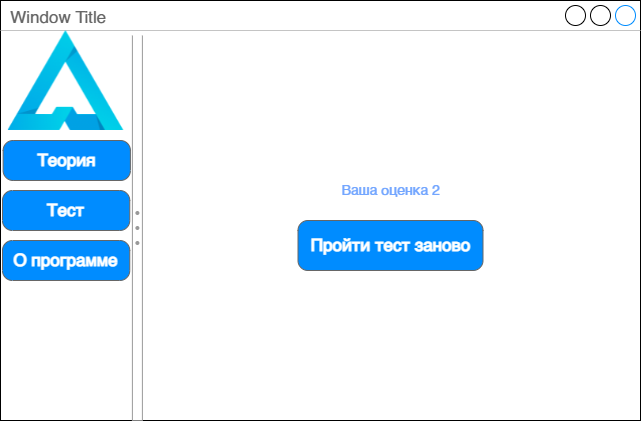
В ходе разработки системы были спроектированы концептуальные прототипы страниц обучающей программы «Основы работы в программном обеспечении Автограф». Концептуальный прототип представляет собой схематичное изображение будущих экранов и создается на ранних этапах разработки продукта. Концептуальное прототипирование отлично подходит для самого быстрого тестирования идей, потому что позволяет набросать основные элементы экранов за считанные минуты. На рисунках 6-9 изображены концептуальные модели обучающей программы «Основы работы в программном обеспечении Автограф».



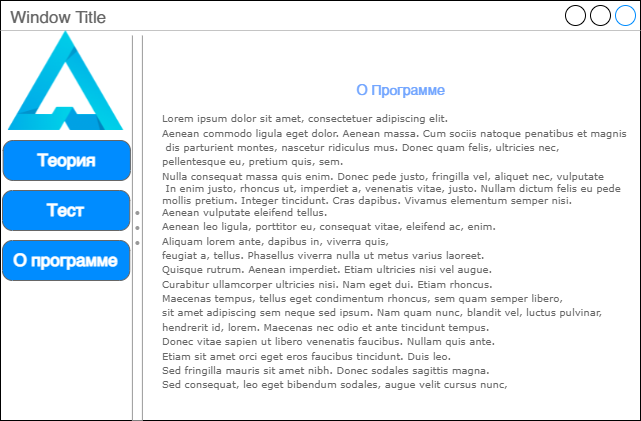
1. Прототип главного окна



1. Прототип страницы тестирования



1. Прототип страницы завершенного теста



1. Прототип страницы «О программе»
   1. Создание обучающей программы «Основы работы в программном обеспечении Автограф»

В процессе разработки обучающей программы создано несколько страниц, каждая из которых служит для выполнения функций, определенных в процессе проектирования.

Приложение организовано по принципам страничного интерфейса, т.е. каждая страница загружается в главное окно. Для организации навигации между страницами используется элемент Frame в главном окне приложения.

Для хранения теоретического материала используется HTML-документ, отображаемый в приложении при помощи элемента WebBrowser. Для навигации по теоретическому материалу в HTML-документе представлено содержание с ссылками на соответствующие разделы.

Для хранения вопросов для тестирования используются XML-файлы с особой разметкой, соответствующей листингу 1. Информация из них считывается и преобразуется с помощью метода Deserialize (листинг 2).

1. Разметка XML-файла с вопросами к тесту

|  |
| --- |
| <?xml version="1.0"?>  <ArrayOfQuestion xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">  <Question>  <Text>Какой раздел отсутствует в главном меню?</Text>  <Answers>  <Answer>  <Text>Диаграмма</Text>  <IsRight>true</IsRight>  </Answer>  <Answer>  <Text>Файл</Text>  <IsRight>false</IsRight>  </Answer>  <Answer>  <Text>Правка</Text>  <IsRight>false</IsRight>  </Answer>  <Answer>  <Text>Объект</Text>  <IsRight>false</IsRight>  </Answer>  </Answers>  <Type>Single</Type>  </Question> |

1. Код метода Deserialize

|  |
| --- |
| public static T Deserialize<T>(string path) where T : class  {  //объявление объекта для типа XmkSerializer  XmlSerializer formatter = new XmlSerializer(typeof(T));  using (FileStream fs =  new FileStream(path, FileMode.OpenOrCreate))  {  //присваивание объекта в ходе десериализации файла  T obj = formatter.Deserialize(fs) as T;  return obj;  }  } |

Для последующей работы с объектами вопросов разработан пользовательский элемент управления Quest[5]. Для элемента разработаны метод для генерации элементов для указания ответов – InitializeQuestion, представленный листингом 3, и метод проверки правильности ответов CheckAnswer[4], представленный листингом 4.

1. Код метода InitializeQuestion

|  |
| --- |
| private void InitializeQuestion(Question question)  {  currentQuestion = question;  QuestionTextBlock.Text = question.Text;  //очистка формы от элементов управления  Clear();  //перемешивание массива ответов в случайном порядке  question.Answers.Shuffle();  switch (question.Type)  {  case QuestionType.Single:  //инициализация вопроса с одним ответом  InitializeSingle(question);  break;  case QuestionType.Multiple:  //инициализация вопроса с несколькими ответами  InitializeMultiple(question);  break;  case QuestionType.Text:  //инициализация вопроса с открытым ответом  InitializeText(question);  break;  default:  break;  }  //запуск таймера на вопрос  timer.Start();  } |

1. Код метода CheckAnswer

|  |
| --- |
| private bool CheckAnswer()  {  switch (currentQuestion.Type)  {  case QuestionType.Single:  //проверка вопроса с одним ответом  foreach (var radioButton in radioButtons)  {  if ((bool)radioButton.Tag  && (bool)radioButton.IsChecked)  {  return true;  }  }  return false;  case QuestionType.Multiple:  //проверка вопроса с несколькими ответами  foreach (var checkBox in checkBoxes)  {  if (!(bool)checkBox.IsChecked && (bool)checkBox.Tag  || (bool)checkBox.IsChecked && !(bool)checkBox.Tag)  {  return false;  }  }  return true;  case QuestionType.Text:  var correctAnswers = currentQuestion.Answers  .Where(a => a.IsRight);  if (correctAnswers == null  || correctAnswers.Count() == 0)  {  return false;  }  //проверка текстового ответа  foreach (var answer in correctAnswers)  {  if (answer.Text.ToLower().Trim()  == AnswerTextBox.Text.ToLower().Trim())  {  return true;  }  }  return false;  default:  return false;  }  } |

В методе InititalizeQuestion используются метод Clear (листинг 5) для очистки элемента Quest от предыдущего вопроса и методы для генерации элементов управления конкретного типа вопроса. InitializeSingle для вопросов с одним вариантом ответа, представлен листингом 6. InitializeMultiple для вопросов с несколькими вариантами ответа, представлен листингом 7. InitializeText для вопросов с открытым вариантом ответа, представлен листингом 8.

1. Код метода Clear

|  |
| --- |
| private void Clear()  {  //перебираются кнопки для их удаления из списка  foreach (var item in radioButtons)  {  AnswerStackPanel.Children.Remove(item);  }  //перебираются флажки для их удаления из списка  foreach (var item in checkBoxes)  {  AnswerStackPanel.Children.Remove(item);  }  //скрываются панели для отображения сопутсвующей информации  AnswerTextBox.Visibility = CheckAnswerLabel.Visibility =  EmptyAnswerLabel.Visibility = AnswerStackPanel.Visibility =  Visibility.Collapsed;  //удаляется текстовое поля ответа  AnswerTextBox.Clear();  //удаляются кнопки выбора ответа  radioButtons.Clear();  //удаляются флажки выбора ответа  checkBoxes.Clear();  } |

1. Код метода InititalizeSingle

|  |
| --- |
| private void InitializeSingle(Question question)  {  //указывается время для таймера на 30 секунд  QuestionTime = 30;  //отображается панель с ответами  AnswerStackPanel.Visibility = Visibility.Visible;  foreach (var answer in question.Answers)  {  //создаются кнопки типа RadioButton  RadioButton radioButton = new RadioButton();  //задается текст у кнопки выбора ответа  radioButton.Content = answer.Text;  //указывается правильность ответа  radioButton.Tag = answer.IsRight;  //добавляются кнопки в группу “Answers” для их общей работы  radioButton.GroupName = "Answers";  radioButtons.Add(radioButton);  //добавляется кнопку в панель ответов  AnswerStackPanel.Children.Add(radioButton);  }  } |

1. Код метода InititalizeMultiple

|  |
| --- |
| private void InitializeMultiple(Question question)  {  //указывается время для таймера на 30 секунд  QuestionTime = 60;  //отображается панель с ответами  AnswerStackPanel.Visibility = Visibility.Visible;  foreach (var answer in question.Answers)  {  //создаем флажки  CheckBox checkBox = new CheckBox();  //задается текст у флажка  checkBox.Content = answer.Text;  //указывается правильность ответа  checkBox.Tag = answer.IsRight;  //добавляется флажок в список флажков  checkBoxes.Add(checkBox);  //добавляется флажок в панель ответов  AnswerStackPanel.Children.Add(checkBox);  } |

1. Код метода InititalizeText

|  |
| --- |
| private void InitializeText(Question question)  {  //указывается время для таймера на 30 секунд  QuestionTime = 90;  //отображается панель с ответом  AnswerTextBox.Visibility = Visibility.Visible;  } |

1. Тестирование разработанного продукта
   1. Тестирование методом «черного ящика»

Тестирование чёрного ящика – это метод тестирования функционального поведения программы с точки зрения внешнего мира, при котором не используется знание о внутреннем устройстве кода тестируемого объекта.

Для тестирования будет использоваться метод «черного ящика».

В таблице 1 показан набор тестов разработанного приложения.

Таблица 1 – Набор тестов разработанного приложения

| Действие | Ожидаемый результат | Полученный результат |
| --- | --- | --- |
| Нажать на кнопку «Теоретический материал» | Переход на страницу «Теоретический материал» | Соответствует ожидаемому |
| Нажать на кнопку «О программе» | Переход на страницу «О программе» | Соответствует ожидаемому |
| Нажать на кнопку тестирования «Интерфейс» | Переход на страницу теста по теме «Интерфейс» | Соответствует ожидаемому |
| Нажать на кнопку тестирования «Работа в программе ч. 1» | Переход на страницу теста по теме «Работа в программе ч. 1» | Соответствует ожидаемому |
| Нажать на кнопку тестирования «Работа в программе ч. 2» | Переход на страницу теста по теме «Работа в программе ч. 2» | Соответствует ожидаемому |
| Завершение таймера во время тестирования | Засчитывается выбранный ответ, так как будто вы нажали на кнопку «Ответить» | Соответствует ожидаемому |
| Нажать на кнопку «Ответить» не выбрав ответа. | Вывод сообщения на экран «Пожалуйста, ответьте на вопрос» | Соответствует ожидаемому |
| Нажать на кнопку «Ответить» выбрав правильный ответ | Вывод сообщения на экран «Вы ответили верно. Ваш ответ:» | Соответствует ожидаемому |
| Нажать на кнопку «Ответить» выбрав неправильный ответ | Вывод сообщения на экран «Вы ответили не верно. Правильный ответ:» | Соответствует ожидаемому |
| Нажать на кнопку «Далее» в тестировании не на последнем вопросе | Отображение следующего вопроса на странице тестирования | Соответствует ожидаемому |

Продолжение таблицы 1

| Действие | Ожидаемый результат | Полученный результат |
| --- | --- | --- |
| Нажать на кнопку «Далее» в тестировании на последнем вопросе | Отображение результатов тестирования | Соответствует ожидаемому |
| Нажать на кнопку «Пройти тест заново» | Отображение первого вопроса теста | Соответствует ожидаемому |
| Нажать на кнопку «Начать» на странице тестирования | Отображение первого вопроса теста | Соответствует ожидаемому |
| Завершить тестирование, ответив правильно на 5 вопросов из 10 | Отображение результатов на странице тестирования: «Ваша оценка 2» | Соответствует ожидаемому |
| Завершить тестирование, ответив правильно на 6 вопросов из 10 | Отображение результатов на странице тестирования: «Ваша оценка 3» | Соответствует ожидаемому |
| Завершить тестирование, ответив правильно на 8 вопросов из 10 | Отображение результатов на странице тестирования: «Ваша оценка 4» | Соответствует ожидаемому |
| Завершить тестирование, ответив правильно на 9 вопросов из 10 | Отображение результатов на странице тестирования: «Ваша оценка 5» | Соответствует ожидаемому |

С учетом критериев качества ПО, можно сделать вывод о довольно высоком уровне качества: ПП полноценно выполняет поставленные задачи, имеет высокую надежность, так как выполняет обработку большинства возможных исключений[1].

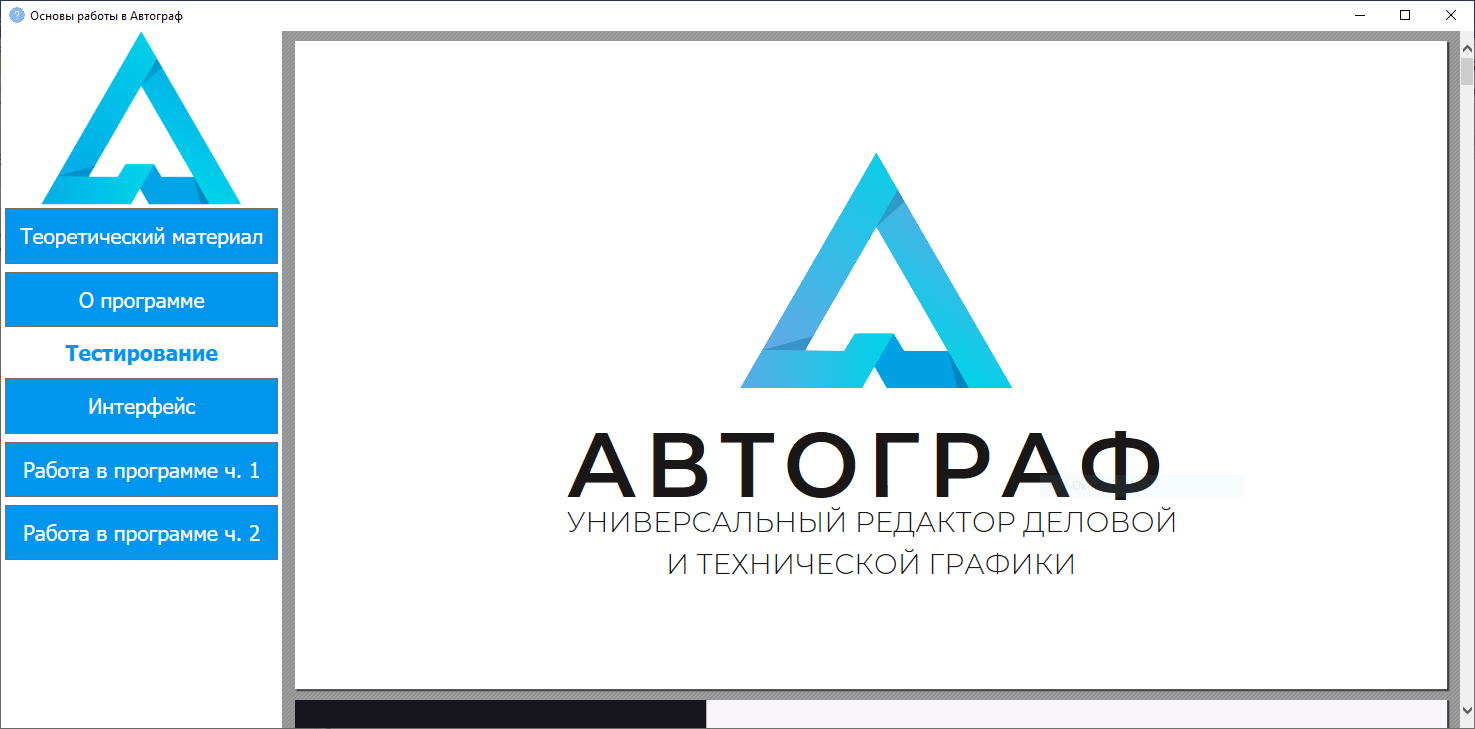
1. Руководство пользователя
   1. Установка приложения

Для установки ПП на ПК пользователя необходимо[3]:

* запустить файл установщика (setup.exe или AutographManualSetup.msi),
* выбрать папку для установки приложения,
* в папке с приложением или на рабочем столе запустить двойным щелчком мыши AutographManual.exe.
  1. Инструкция по работе

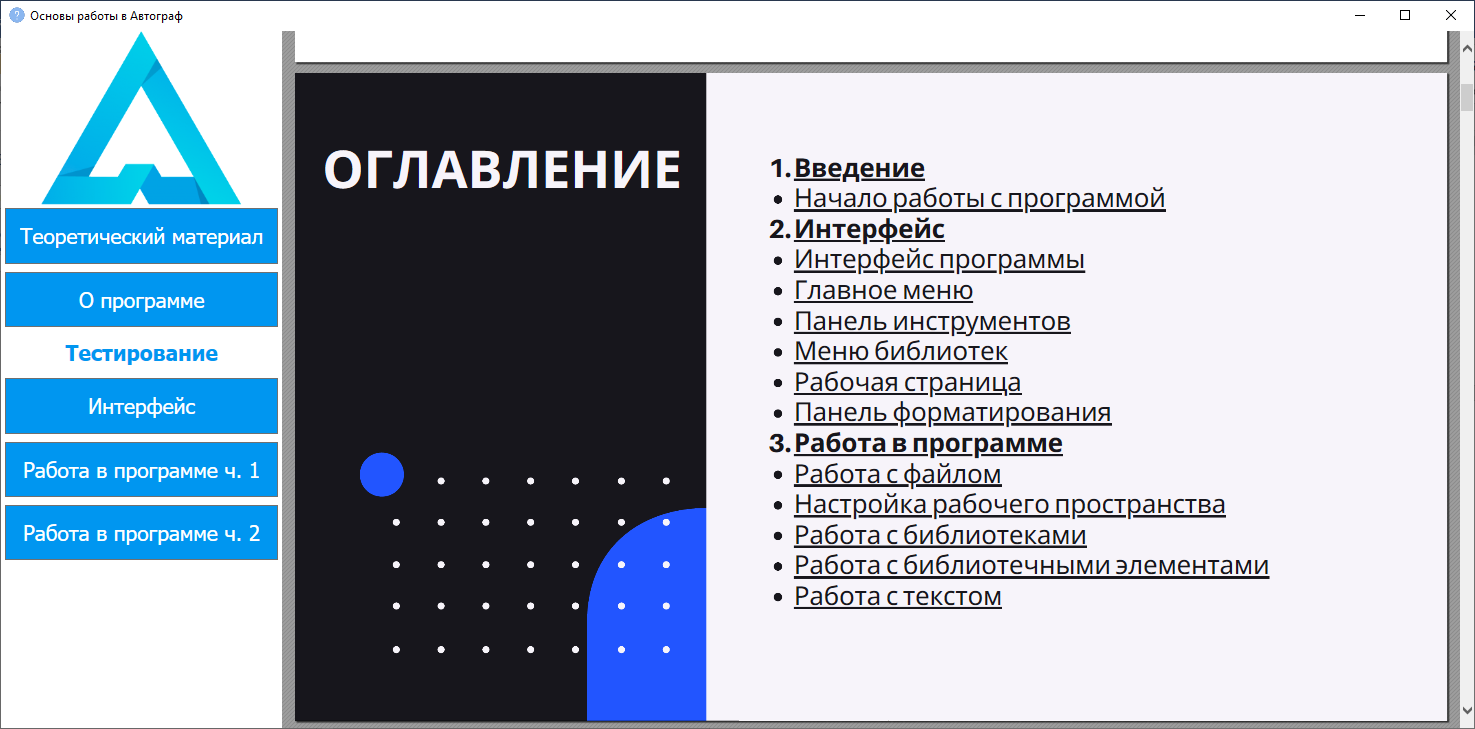
Для запуска приложения требуется дважды кликнуть на его ярлык.

После запуска приложения откроется главное окно с открытой страницей теоретического материала (рисунок 10).



1. Вид страницы теоретического материала

На странице теоретического материала пользователь может листать HTML-документ и использовать навигацию по оглавлению (рисунок 11) для переходов к нужным разделам.



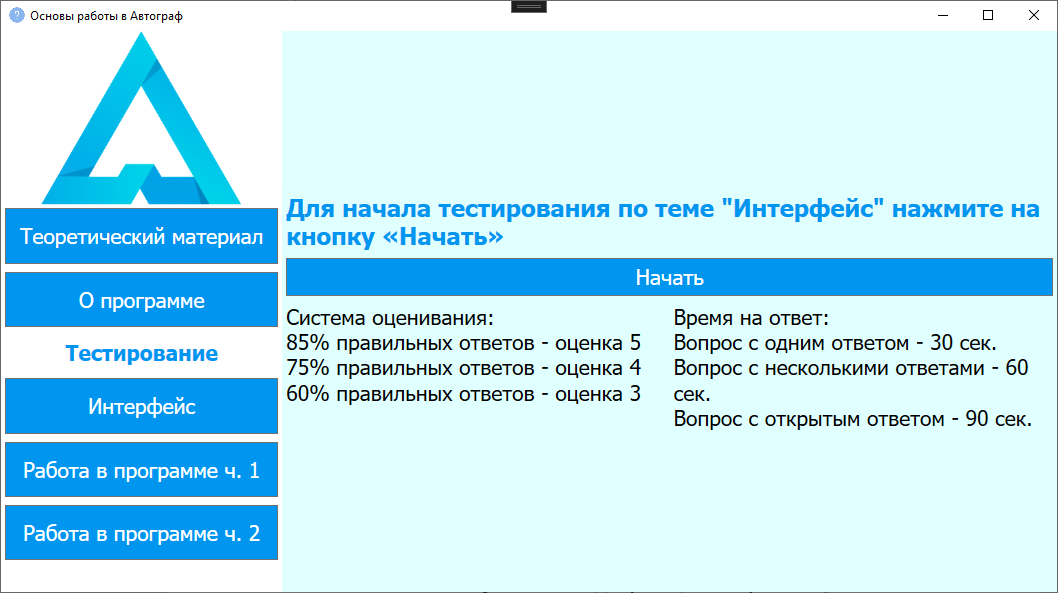
1. Вид страницы теоретического материала с оглавлением

После нажатия на кнопку тестирования (рисунок 12) «Интерфейс», «Работа в программе ч. 1» или «Работа в программе ч. 2» откроется страница тестирования с соответствующей темой.



1. Вид кнопок тестирования

На странице тестирования будут размещены подпись с названием теста, инструкции по прохождению теста и кнопка «Начать», которая запустит тест (рисунок 13).

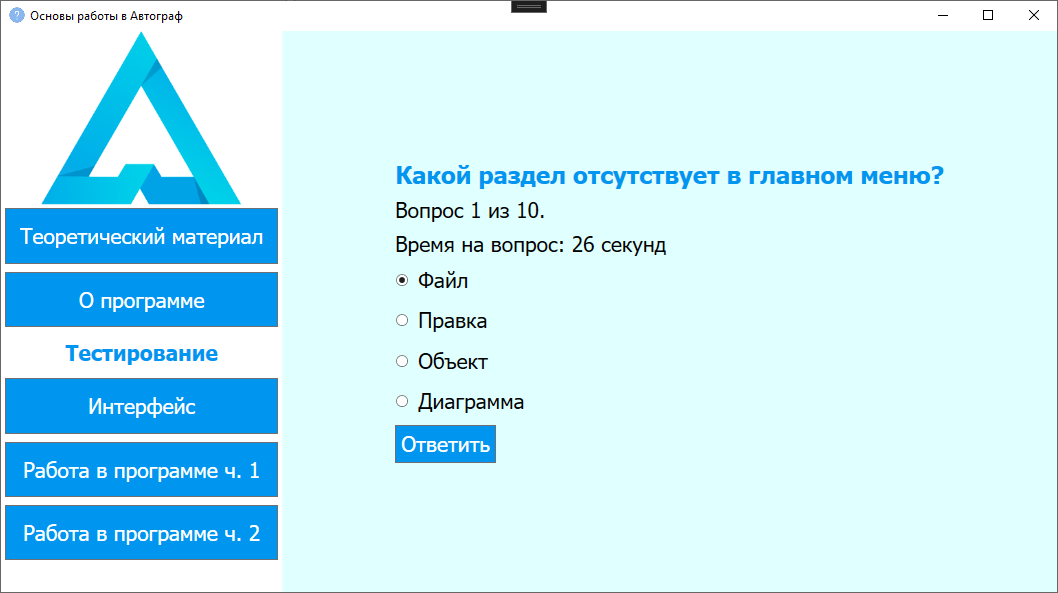


1. Вид страницы тестирования

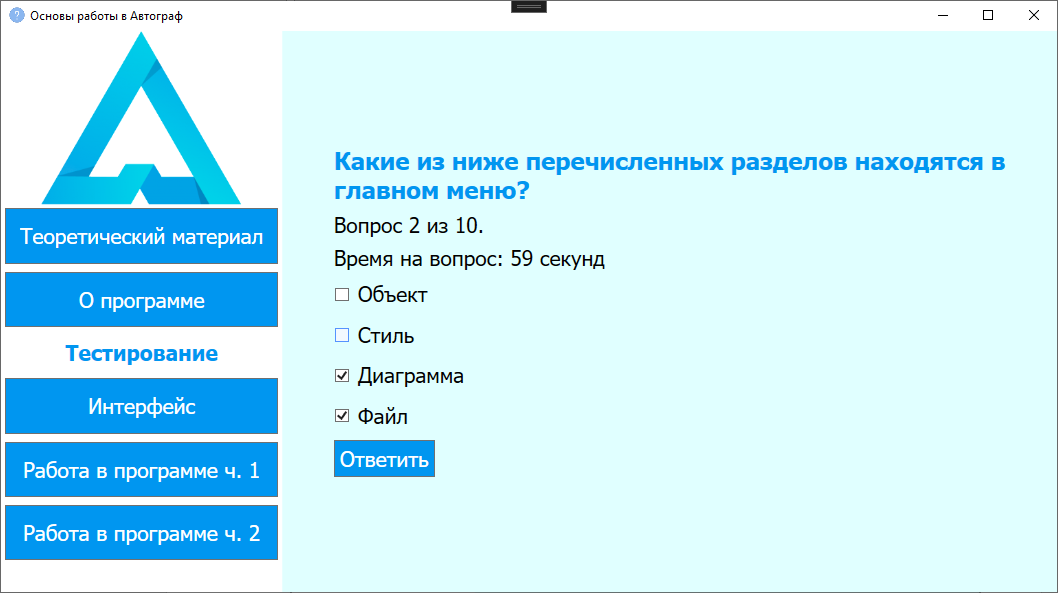
На странице отобразится первый вопрос, содержащий время на ответ, текст вопроса, варианты ответа (или строку для ответа на открытый вопрос), количество вопросов в тестировании и кнопку отправки ответа.

Во время прохождения теста будут появляться вопросы разных видов: вопрос с одним вариантом ответа (рисунок 14), вопрос с несколькими вариантами ответа (рисунок 15), и вопрос с открытым ответом (рисунок 16).

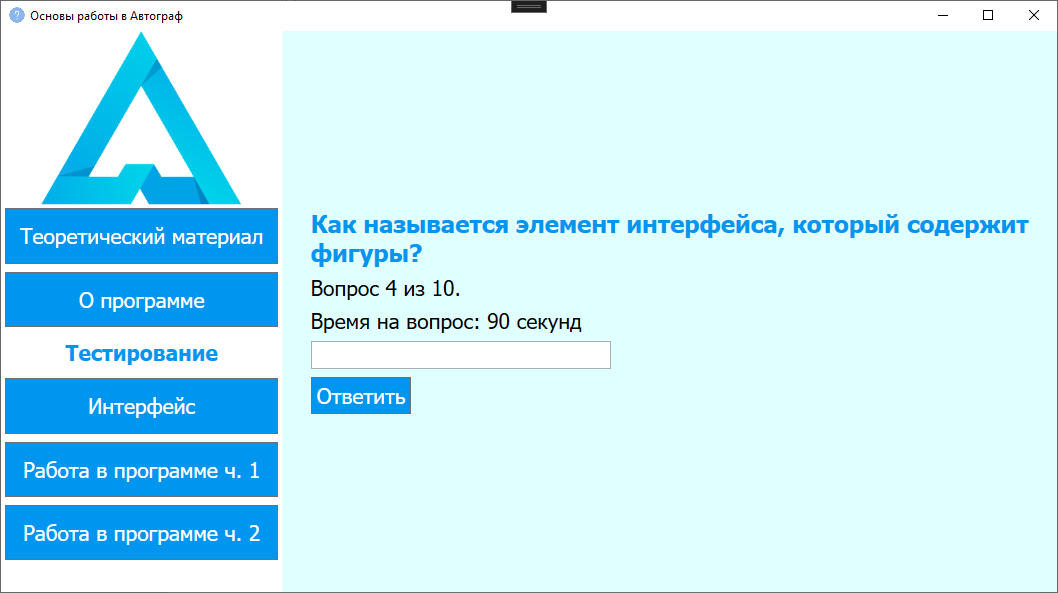
После ответа на последний вопрос при нажатии кнопки «Далее» открывается страница с результатами пройденного тестирования   
(рисунок 17), где отображается количество полученных баллов и оценка.



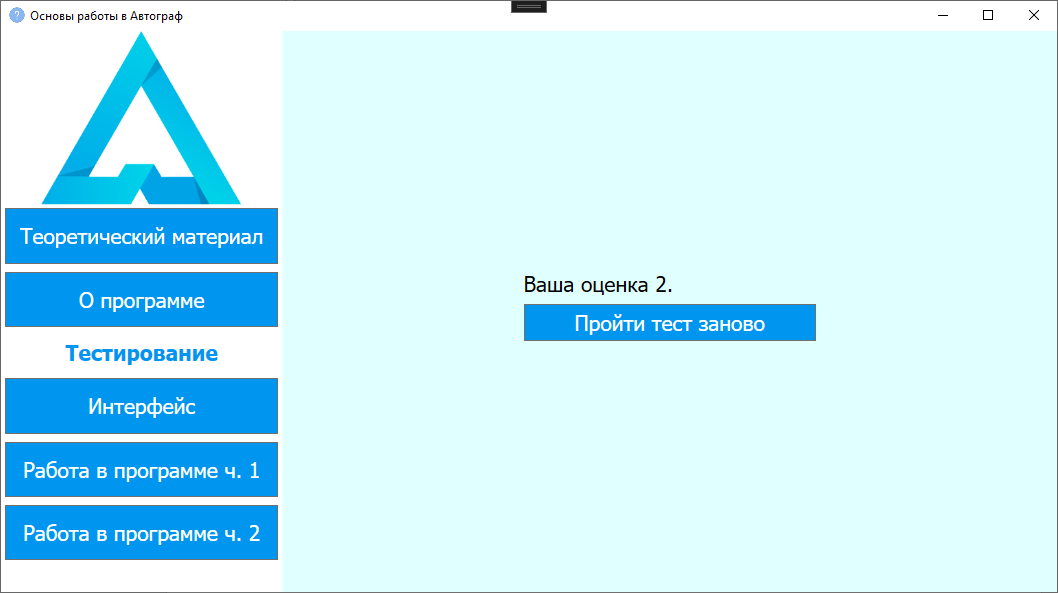
1. Вид страницы тестирования



1. Вид страницы тестирования



1. Вид страницы тестирования



1. Вид страницы тестирования

Заключение

В ходе курсового проектирования сформированы и закреплены навыки проектирования и разработки клиентских приложений; сформированы навыки по разработке спецификаций и технической документации разрабатываемого ПП.

Цель курсового проектирования достигнута: разработана однопользовательская клиентская обучающая программа «Основы работы в программном обеспечении Автограф».

Для достижения этой цели решены следующие задачи:

* выполнен сбор требований к обучающей программе,
* проанализированы собранные требования,
* выполнен сбор теоретического материала по поставленной теме,
* выбраны средства разработки ПП,
* обоснован выбор средств разработки ПП,
* описана спецификация разрабатываемого ПП,
* спроектирована причинно-следственная диаграмма,
* спроектирована UML-диаграмма стереотипов,
* спроектирована DFD-диаграмма первого и второго уровня,
* спроектирован прототип ПП,
* спроектирован ПП,
* разработан ПП,
* подобран теоретический материал,
* подобраны вопросы для тестирования,
* разработано руководство по эксплуатации,
* проведено тестирование разработанного ПП.

Результат проектирования можно считать успешным, так как поставленные цели и задачи выполнены

Список использованных источников

1. Ананьева, Т. Н. Стандартизация, сертификация и управление качеством программного обеспечения: учебное пособие / Т. Н. Ананьева, Н. Г. Новикова, Г. Н. Исаев – Москва : ИНФРА-М, 2021. – 232 с. – URL: https://znanium.com/catalog/document?id=376058. Режим доступа: для зарегистрир. пользователей. – Текст : электронный.
2. Гагарина, Л. Г. Технология разработки программного обеспечения: учебное пособие / Л. Г. Гагарина, Е. В. Кокорева, Б. Д. Виснадул / под редакцией Л. Г. Гагариной. – Москва : ИД «ФОРУМ» : ИНФРА-М, 2021. – URL: https://znanium.com/catalog/document?id=364479. – Режим доступа: для зарегистрир. пользователей. – Текст : электронный.
3. Зверева, В. П. Сопровождение и обслуживание программного обеспечения компьютерных систем (2–е изд., испр.) : учебник / Зверева, В.П. – Москва: Академия, 2020.
4. Павловская, Т. А. C#. Программирование на языке высокого уровня: Учебник для вузов / Т. А. Павловская. – Санкт-Петербург : Питер, 2021. – 432 с. – URL: https://ibooks.ru/bookshelf/377952/reading. – Режим доступа: для зарегистрир. пользователей. – Текст : электронный.
5. Фленов, М. Е. Библия C# / М. Е. Фленов. – 4-е изд., перераб. и доп. – Санкт-Петербург : БХВ-Петербург, 2019. – 512 с. – URL: https://ibooks.ru/bookshelf/366634/reading. – Режим доступа: для зарегистрир. пользователей. – Текст : электронный.

Приложение А

**(обязательное)**

**Программный документ «Техническое задание»**

Документ оформлен согласно ГОСТ 19.201 – 78 «ЕСПД. Техническое задание. Требование к содержанию и оформлению».

Стадии разработки описаны согласно ГОСТ 19.102 – 77 «ЕСПД. Стадии разработки».