Учреждение образования «Белорусский

государственный университет информатики и

радиоэлектронники»

Факультет комьютерный систем и сетей

Кафедра электронных вычислительных машин

К ЗАЩИТЕ ДОПУСТИТЬ

\_\_\_\_\_\_\_\_\_ А.В.Самусевич

Пояснительная записка

по курсовому проекту

«психологические тесты»

по дисциплине

«Системное программное обеспечение вычислительных машин»

БГУИР КР 1-40 02 01 118 ПЗ

Выполнил: Руководитель:

студент гр. 650501 Самусевич А.В.

Одод М.Ю.

МИНСК 2018

Учреждение образования

«Белорусский государственный университет информатики

и радиоэлектронники»

Факультет: Комьютерных систем и сетей

Кафедра: Электронных вычислительных машин

Специальность: 40 02 01 «Вычислительные машины, системы и сети»

Предмет: Системное и программноеобеспечение электронных вычислительных машин

УТВЕРЖДАЮ

Заведующий кафедрой ЭВМ

\_\_\_\_\_\_\_\_\_\_Д. И. Самаль

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_20\_\_г.

ЗАДАНИЕ

По курсовому проектированию студента

Одод Михаил Юрьевич

1. Тема проекта: «Психологические тесты с использованием базы данных»

2. Дата выдачи задания:20.02.2018

3 .Предоставление студентом готового проекта:08.06.2018

4. Решаемые задачи и функционал разрабатываемого ПО:

4.1 Разработка и создание таблиц для базы данных

4.1.1 Создание таблиц

4.1.2 Создание связей между таблицами

4.2 Создание простого интерфейса

4.3 Реализовать события в интерфейсе

4.3.1 Создание событий на кнопки при нажатии мыши

4.3.2 Реализация запросов

4.4 Создать изображения (иконки) для красочной визуализации

5. ОС и средства разработки:

5.1 *Visual Studio*

5.2 *Windows 7*

6. Содержание пояснительной записки:

Введение. 1. Постановка задачи. 2. Проектирование задачи. 3. Программная реализация.

4. Тестирование. 5. Применение. Заключение. Список источников. Приложения.

7.Перечень графического материала:

7.1 Психологические тесты. Диаграмма классов. Формат А4.

7.2 Психологические тесты. Диаграмма состояний. Формат А4.

7.3 Листинг программы

8. Контрольные точки готовности сдачи проекта:

8.1 01.03 – Введение, глава 1 (постановка задачи), лист задания.

8.2 01.04 – Предварительные диаграммы классов и состояний.

8.3 01.05 – предварительная демонстрация ПО.

8.4 10.05 - Диаграмма классов, Диаграмма состояний, главы 1-7

РУКОВАДИТЕЛЬ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Самусевич А.В.

Задание принял к исполнению \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Одод.М.Ю.

Минск 2017

Содержание

[Введение 5](#_Toc484688757)

[1 Постановка задачи 6](#_Toc484688758)

[1.2 Описание аналогов 7](#_Toc484688759)

[2 Проектирование задачи 10](#_Toc484688768)

[2.1 Алгоритм работы программы 10](#_Toc484688769)

[2.2 Логическое моделирование 10](#_Toc484688770)

[2.3 Обоснование выбора средств и среды разработки 13](#_Toc484688771)

[3 Программная реализация 15](#_Toc484688772)

[3.1 Описание разработанных классов и основных методов 15](#_Toc484688773)

[3.2 Компоненты Windows Forms 17](#_Toc484688774)

[3.2 Структуры данных (входные, выходные, постоянные) 17](#_Toc484688775)

[4 Тестирование 20](#_Toc484688776)

[5 Применение 23](#_Toc484688777)

[5.1 Установка приложения и требования к аппаратно-программным ресурсам ПК 23](#_Toc484688778)

[5.2 Демонстрационный пример работы программы 23](#_Toc484688779)

[Заключение 29](#_Toc484688780)

[Список использованных источников 30](#_Toc484688781)

[Приложение А Код программы 31](#_Toc484688782)

[Приложение Б Тест-кейс 40](#_Toc484688783)

[Приложение В Чек-лист 44](#_Toc484688784)

[Приложение Г Диаграмма состояний 46](#_Toc484688785)

[Приложение Д диаграмма вариантов использования 48](#_Toc484688787)

[Приложение Е Диаграмма деятельности 50](#_Toc484688788)

[Приложение Ж Диаграмма потоков данных 52](#_Toc484688789)

[Приложение З Диаграмма сущность - связь 54](#_Toc484688790)

[Приложение И Диаграмма классов 56](#_Toc484688791)

[Приложение К Диаграмма последовательностей 58](#_Toc484688792)

Приложение Л Структурная схема……………………………………………………………………………………………………………….60

# Введение

Любой человек, время от времени, в определенные жизненные моменты склонен заниматься «самокопанием», абсолютно каждому интересно заглянуть в собственную душу и раскрыть сущность окружающих. В таких ситуациях как нельзя лучше помогают психологические тесты, раскрывающие многие тайны тонкой натуры человека.

Психологические [тесты](http://psi-technology.net/psytest/) можно найти в современном мире везде: в модных известных журналах, в газетах и, конечно же, в глобальной сети. Стоит отметить, что правду о себе можно узнать только в одном случае, если на все вопросы теста отвечать самым откровенным образом и не утаивать ничего, что могло бы помочь раскрыть истинное положение дел.

Обычно люди при выборе психологического теста останавливают свой выбор на тех, которые в полной мере могут помочь ответить на интересующие вопросы. Одни при выборе профессии желают узнать на кого же пойти учиться, иные, какого спутника выбрать для жизни. Женский пол частенько руководствуется тестами при выборе парфюмерии, одежды и даже типа косметики.

Но правильно ли это, могут ли психологические методики тестирования дать достоверный и исчерпывающий ответ на мучающие вопросы?

В современном мире огромная масса информации и возможностей. И в этом хаосе человеку очень легко потеряться и начать пользоваться без разбора всем, что ему предложит рекламная индустрия. Но часто такая информация может оказаться бесполезной, и руководствуясь ею можно пойти в неправильном направлении. Вот в таких ситуациях как раз и приходят на выручку тесты, они помогают определиться в том или ином выборе среди многообразия предлагаемого жизнью.

Что же такое психологический тест? А это ничто другое, как один из способов заглянуть в себя и открыть грани души до сих пор неведомые. Однако к результатам тестов нельзя относиться слишком серьезно, так как тестирование лишь с одной стороны открывает Вашу сущность. Со временем жизнь каждого меняется, а значит, если ответить на тот же тест через время, то и результат теста не останется тем же. Поэтому тесты стоит использовать для расширения своего кругозора, что, безусловно, полезно и никогда не будет лишним.

# 1 Постановка задачи

Разработать программное средство, автоматизирующее процесс психологического тестирования. Данное ПС должно предоставлять пользователю психологические тесты в следующих тематиках:

* личность;
* семья;
* карьера;
* отношения.

Программное средство должно позволять сохранять результат пользователей, добавлять тесты в избранное. Необходимо предусмотреть регистрацию и авторизацию пользователей.

Вся информация должна храниться в базе данных пользователей со следующей структурой:

* Пользователь (id пользователя, имя пользователя, логин, пароль).
* Характеристика (характеристика, тест, балл).
* Вопрос (вопрос, номер вопроса, номер теста).
* Тест (номер теста, описание, название).
* Варианты ответов (номер варианта, номер вопроса, номер теста, вариант, балл).
* Результаты (номер теста, результат, id пользователя).
* Избранные тесты (id пользователя, номер теста).

Пользователи разрабатываемого ПС должны иметь возможность проходить тесты, получать информацию о тестах, добавлять их в избранное. Также программа должна обладать справкой, которая расскажет об основных возможностях.

ПС должен обладать понятным и красочным интерфейсом. Преимущественно должны использоваться следующие цвета:

* синий (голубой);
* фиолетовый;
* розовый;
* желтый;
* белый.

Необходимо, чтобы пользователь с первых же секунд понимал, что перед ним за программное средство. Оно должно погружать его в атмосферу таинственности и познания.

# 1.2 Описание аналогов

На данный момент, существует несколько достойных аналогов программ для психологического тестирования. В основном все они рассчитаны под мобильные устройства. Здесь приведены несколько лучших:

Соционический тест. Соционика - это не наука. Соционика - это субкультура. И только Богу известно, когда наконец страстный Гюго найдёт своего невозмутимого Робеспьера. Чтобы определить свой соционический тип, нужно пройти тест. И поможет в этом программа "Соционический тест".

## 

Рисунок 1.1 – Соционический тест

Программа предназначена для определения социотипа личности. Будет полезна руководителю при принятии решения по назначению сотрудника на ту или иную должность. Да и любой наёмный работник сможет узнать о себе много нового.

В программе имеется таблица взаимоотношений между социотипами. Используя её, можно подобрать коллектив.

### Достоинства Соционического теста

* В настройках интерфейса доступен русский язык
* Бесплатное распространение программы
* Большое количество тестов в узкой тематике, что позволит хорошо изучить себя в соционике

### Недостатки Соционического теста

* Сложный интерфейс
* Невозможность сохранения результатов

iTemperament – программа психологического тестирования, написанная под мобильную систему IOS, которая поможет определить генетический психотип, сильные и слабые стороны вашей личности, расскажет о ваших генах и особенностях

### 

Рисунок 1.2 - iTemperament

### Преимущества:

* Удобный интерфейс
* Большое количество тестов на различные тематики

### Недостатки:

* Иногда вылетает
* Имеются платные тесты

Познай себя – психологические тесты. Программа для психологического тестирования личности, разработанная под мобильную систему IOS, которая предоставляет огромное количество психологических тестов для людей всех возрастов.

Познай себя поделена на логические блоки (разделы), в каждом из которых имеются уникальные тесты, помогающие определить сильные и слабые стороны человека. Познай себя поможет узнать много нового про вашу личность, укажет что, возможно, стоит исправить в себе, а также расскажет о вашем темпераменте, характере и многом другом.

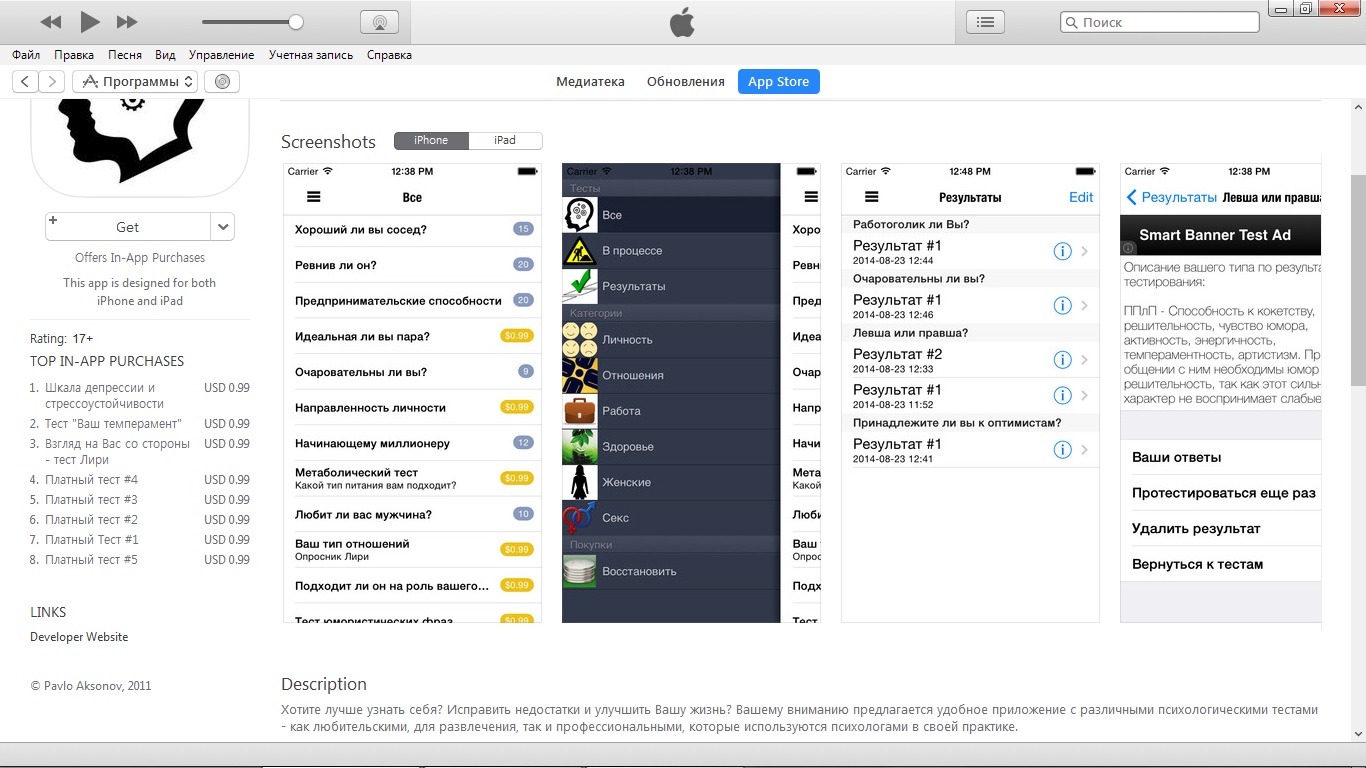


Рисунок 1.3 – Познай себя

### Преимущества

* Русскоязычная версия
* Портативная
* Огромный выбор тестов
* Бесплатная

### Недостатки

* Устаревший интерфейс
* Имеются платные тесты

Очень часто встречаются небольшие психолоческие опросники на интернет-ресурсах, однако они имеют очень скромный функционал и, в большинстве, являются узконаправленными.

# 2 Проектирование задачи

## 2.1 Алгоритм работы программы

Алгоритм — набор инструкций, описывающих порядок действий исполнителя для достижения некоторого результата. В старой трактовке вместо слова «порядок» использовалось слово «последовательность», но по мере развития параллельности в работе компьютеров слово «последовательность» стали заменять более общим словом «порядок». Независимые инструкции могут выполняться в произвольном порядке, параллельно, если это позволяют используемые исполнители.

Виды алгоритмов:

* линейный алгоритм — набор команд (указаний), выполняемых последовательно во времени друг за другом;
* разветвляющийся алгоритм — алгоритм, содержащий хотя бы одно условие, в результате проверки которого может осуществляться разделение на несколько альтернативных ветвей алгоритма;
* циклический алгоритм — алгоритм, предусматривающий многократное повторение одного и того же действия (одних и тех же операций) над новыми исходными данными. К циклическим алгоритмам сводится большинство методов вычислений, перебора вариантов.

Оптимальным вариантов демонстрации работы алгоритма программы считается диаграмма состояний, представленная в приложении Г.

# 2.2 Логическое моделирование

Одним из средств, используемых на этапе логического моделирования, являются средства, иллюстрирующие функции, выполняемые системой.

Эти средства соответствуют определенным видам моделей (диаграмм).

Наиболее распространенными являются IDEF0 – модели и соответствующие функциональные диаграммы (Приложение Д).

Наиболее известной реализацией IDEF0 является методология SADT, разработанная Дугласом Россом. Основная идея методологии SADT – это построение древовидной функциональной модели.

Сначала функциональность описывается в целом – это называется контекстной диаграммой. При создании контекстной диаграммы формулируется цель моделирования, область (т.е., что будет рассматриваться, как компонент системы, а что как внешнее воздействие) и позиция, в соответствии с которой будет строиться модель.

Методология SADT представляет собой совокупность методов правил и процедур, предназначенных для построения функциональной модели объекта какой-либо предметной области. Такая модель отображает функциональную структуру объекта, т.е. производимые им действия и связи между этими действиями. Основные элементы этой методологии основаны на следующих концепциях:

Функции отображаются в виде блоков, а интерфейсы представлены дугами (т.е. стрелками), входящими в блок и выходящими из него.

На этапе структурного анализа кроме средств, иллюстрирующих функции, выполняемые системой, существуют средства, иллюстрирующие отношения между данными.

Наиболее распространенными являются следующие:

* DFD – диаграммы потоков данных (Приложение З);

Разрабатываемая модель представляет собой иерархию диаграмм потоков данных (ДПД или DFD), описывающих процесс преобразования данных от ввода в систему до выдачи результата пользователю. Контекстные диаграммы (т.е. диаграммы верхних уровней) определяют основные процессы или подсистемы с внешними входами и выходами. Контекстные диаграммы (т.е. диаграммы верхних уровней) определяют основные процессы или подсистемы с внешними входами и выходами.

ERD - диаграммы “сущность-связь” (Приложение И).

Процесс логического моделирования осуществлялся с использованием следующих программных средств:

1. IBM Rational Rose Enterprise для моделирования диаграмм вариантов использования, последовательности, состояний и классов;
2. Dia для моделирования диаграмм сущность-связь и деятельности;
3. Allfusion Process Modeller для моделирования функциональной и потоков данных диаграмм.

IBM Rational Rose Enterprise предоставляет набор функций, управляемых моделью, для разработки целого ряда приложений, в том числе на языках Ada, ANSI C++, C++, CORBA, Java, Java EE, Visual C++ и Visual Basic. Это программное обеспечение позволяет ускорить разработку таких приложений благодаря созданию кода на основе визуальных моделей с использованием UML (Unified Modeling Language).

Rational Rose Enterprise предлагает общий инструмент моделирования и язык для упрощения рабочей среды и возможности более быстрого создания качественного программного обеспечения.

* Моделирование самых распространенных приложений — предоставляет функции визуального моделирования для разработки приложений многих типов.
* Разработка веб-приложений — предлагает инструменты XML и веб-моделирования веб-приложений.
* Интеграция проектирования и разработки приложений — унифицирует работу коллектив проекта путем предоставления общих средств выполнения и нотации модели UML.

Диаграмма вариантов использования представлена в приложении Е.

Диаграмма последовательности представлена в приложении Л.

Диаграмма состояний представлена в приложении Г.

Dia — свободный кроссплатформенный редактор диаграмм, часть GNOME Office, но может быть установлен независимо. Он может быть использован для создания различных видов диаграмм: блок-схем алгоритмов программ, древовидных схем, статических структур UML, баз данных, диаграмм сущность-связь, радиоэлектронных элементов, потоковых диаграмм, сетевых диаграмм и других.

Dia расширяема новыми наборами объектов, которые описываются с помощью файлов в формате, основанном на XML.

AllFusion ERwin Data Modeler (ранее ERwin) — CASE-средство для проектирования и документирования баз данных, которое позволяет создавать, документировать и сопровождать базы данных, хранилища и витрины данных. Модели данных помогают визуализировать структуру данных, обеспечивая эффективный процесс организации, управления и администрирования таких аспектов деятельности предприятия, как уровень сложности данных, технологий баз данных и среды развертывания.

AllFusion ERwin Data Modeler (ERwin) предназначен для всех компаний, разрабатывающих и использующих базы данных, для администраторов баз данных, системных аналитиков, проектировщиков баз данных, разработчиков, руководителей проектов, — и позволяет управлять данными в процессе корпоративных изменений, а также в условиях стремительно изменяющихся технологий.

AllFusion ERwin Data Modeler (ERwin) позволяет наглядно отображать сложные структуры данных. Удобная в использовании графическая среда системы упрощает разработку базы данных и автоматизирует множество трудоёмких задач, уменьшая сроки создания высококачественных и высокопроизводительных транзакционных баз данных и хранилищ данных. Продукт улучшает коммуникацию организации, обеспечивая совместную работу администраторов и разработчиков баз данных, многократное использование модели, а также наглядное представление комплексных активов данных в удобном для понимания и обслуживания формате.

## 2.3 Обоснование выбора средств и среды разработки

Выбор среды разработки и языка программирования всегда очень важен. Их нужно выбирать исходя из своих возможностей и возможностей среды разработки и языка, которые должны обеспечить соответствие программы с требованиями пользователя.

Для разработки приложения будет использована среда разработки Microsoft Visual Studio.

Microsoft Visual Studio – бесплатная, полнофункциональная и расширяемая интегрированная среда разработки программного обеспечения и ряд других инструментальных средств для создания современных приложений для Windows, Android и iOS, а также веб-приложений и облачных служб. Девиз данной среды разработки «Все, что вам нужно, в одном месте». А ведь что нужно пользователю?

* мощные инструменты программирования, которые обеспечивают легкость написания кода, перемещения по нему и устранение ошибок;
* усовершенствованная отладка - локальная или удаленная отладка для разных языков. Диагностика проблем производительности непосредственно из рабочего процесса отладчика;
* приложения для любой платформы - Windows Desktop, универсальная платформа Windows, iOS и Android;
* веб-инструменты, обеспечивающие создание и развертывание современных веб-приложений. Разработка веб-приложений с помощью ASP.NET, Node.js, Python и JavaScript. Использование таких мощных веб-платформ, как AngularJS, jQuery, Bootstrap, Django и Backbone.js;
* использование языка на свой выбор. MVS поддерживает нескольких языков, включая C#, Visual Basic, F#, C++, JavaScript, TypeScript, Python и множество других. Помогает писать, отлаживать и тестировать код независимо от того, какой язык выбран.
* доступ к тысячам расширений. Можно настроить Visual Studio с помощью инструментов, элементов управления и шаблонов, доступных у партнеров и в сообществе, или пойти дальше и создавать собственные модули.
* интеграция с Git – размещение кода где угодно.
* доступ к бесплатным инструментам и ресурсам.

Так же будет использован интерфейс – Windows Forms. Это интерфейс программирования приложений (API), отвечающий за графический интерфейс пользователя и являющийся частью Microsoft .NET Framework. Данный интерфейс упрощает доступ к элементам интерфейса Microsoft Windows. Приемуществом является и то, что управляемый код — классы, реализующие API для Windows Forms, не зависят от языка разработки.

Язык программирования – C#. Это полнофункциональный объектно-ориентированный язык, который поддерживает все три «столпа» объектно-ориентированного программирования: инкапсуляцию, наследование и полиморфизм. Он имеет прекрасную поддержку компонентов, надежен и устойчив благодаря использованию «сборки мусора», обработки исключений, безопасности типов.

Таким образом, разрабатывается приложение на языке программирования C# в Microsoft Visual Studio с использованием интерфейса Windows Forms.

# 3 Программная реализация

## 3.1 Описание разработанных классов и основных методов

В программе реализованы следующие классы (Приложение К):

* «Form1» – класс формы Окна приветствия. Данная форма содержит три кнопки «Авторизация», «Регистрация» и «Справка». По нажатии на данные кнопки, открываются новые формы. У каждой кнопки есть текстовая подсазка. На фон формы и кнопок установлены картинки.;
* «Логин» – класс формы авторизации. Данный клас предназначен для допуска зарегистрированного пользователя к главному меню. В данном классе делается запрос к базе данных, который либо предоставляет пользователю допуск, либо нет;
* «Регистрация» – класс формы регистрации. Данный клас служит для добавления новых пользователей в базу данных и, если новый пользователь добавился, допуска к главному меню. Данный класс обрабатывает информацию, введенную пользователем в textBoxs и проверяет ее на уникальность. Если логин уникален, то пользователь получит допуск. Иначе ему предложится ввести другой логин;
* «Form2» – класс формы справки. Данный класс хранит справочную информацию о программе;
* «Главное меню» – класс формы главного меню. Данный класс служит для навигации и предлагает пользователю переход на последующие формы;
* «Разделы» - класс формы разделов. Данный класс передает на форму 3 определенный параметр, в зависимости от выбранного раздела;
* «Form3» - класс формы списка тестов. Данный класс служит для отображение списка тестов, в зависимости от выбранного раздела меню. В данном классе делается запрос к базе данных, получаются имена тестов и заносятся в список;
* «Тестирование» - класс формы тестирования. Данный класс служит для отображения описания выбранного теста, процесса тестирования и результата. В нем обрабатываются ответы пользователя на тестовые вопросы и выводится ему резултат. Данный класс также служит для добавления теста в раздел «Избранное» и для сохранения результата в базу данных;
* «Forms» - данный класс содержит в себе формы, с которыми мы работаем во время работы программы. Он предназначен для более удобного обращения и передачи данных между формами;
* «Test» - Данный содержит в себе поля, связанные с тестом. Он служит для запоминания того, с каким мы тестом работаем, в каком разделе;
* «User» - Данный класс содержит в себе информацию о пользователе, который работает в программе.

В данной программе реализовано большое количество различных методов, далее приведены основные из них:

* private void listBox1\_Click(object sender, EventArgs e). Данный метод обрабатывет выбор пользователя в listBox при выборе вариантов ответов;
* void NewQuest(int num). Данный метод получает из базы данных и выводит на экран новые варианты ответов;
* void Rezult(int ball). Метод подсчитывает количество баллов, набранных пользователем и выводит ему результат, соответствующий данному баллу;
* private void button1\_Click(object sender, EventArgs e). Данный метод выводит первый тестовый вопрос. После окончания теста, он позволяет сохранить результата в БД.
* bool Is(string table). Данный метод проверяет имеетсля ли уже у пользователя результат по заданному тесту или добавлен ли этот тест уже в избранное;
* private void button3\_Click(object sender, EventArgs e). Метод выполняет добавление теста в избранное;
* private void button1\_Click(object sender, EventArgs e). Данный метод добавляет тестовые вопросы, варианты ответов и характеристики в БД;
* private void dataGridView1\_UserDeletingRow(object sender, DataGridViewRowCancelEventArgs e). Данный метод удаляет из базы данных вопрос, вариант ответа или характеристику;
* private void button2\_Click(object sender, EventArgs e). Данный метод добавляет новый тест в базу данных;
* private void button1\_Click(object sender, EventArgs e). Данный метод проводит аутентификацию пользователя;
* private void button1\_Click(object sender, EventArgs e). Данный метод проводит регистрацию пользователя.

Имеется БД «Users» со следующими таблицами: пользователи, тесты, вопросы, ответы, характеристики, понравившиеся тесты, результаты. Все данные получаются с помощью запросов. Редактирование и добавление также реализовано запросами в программном коде.

## 3.2 Компоненты Windows Forms

В программе были использованы следующие компоненты:

* button, при нажатии на неё происходит программно связанное с этим нажатием действие либо событие (рисунок 3.1);

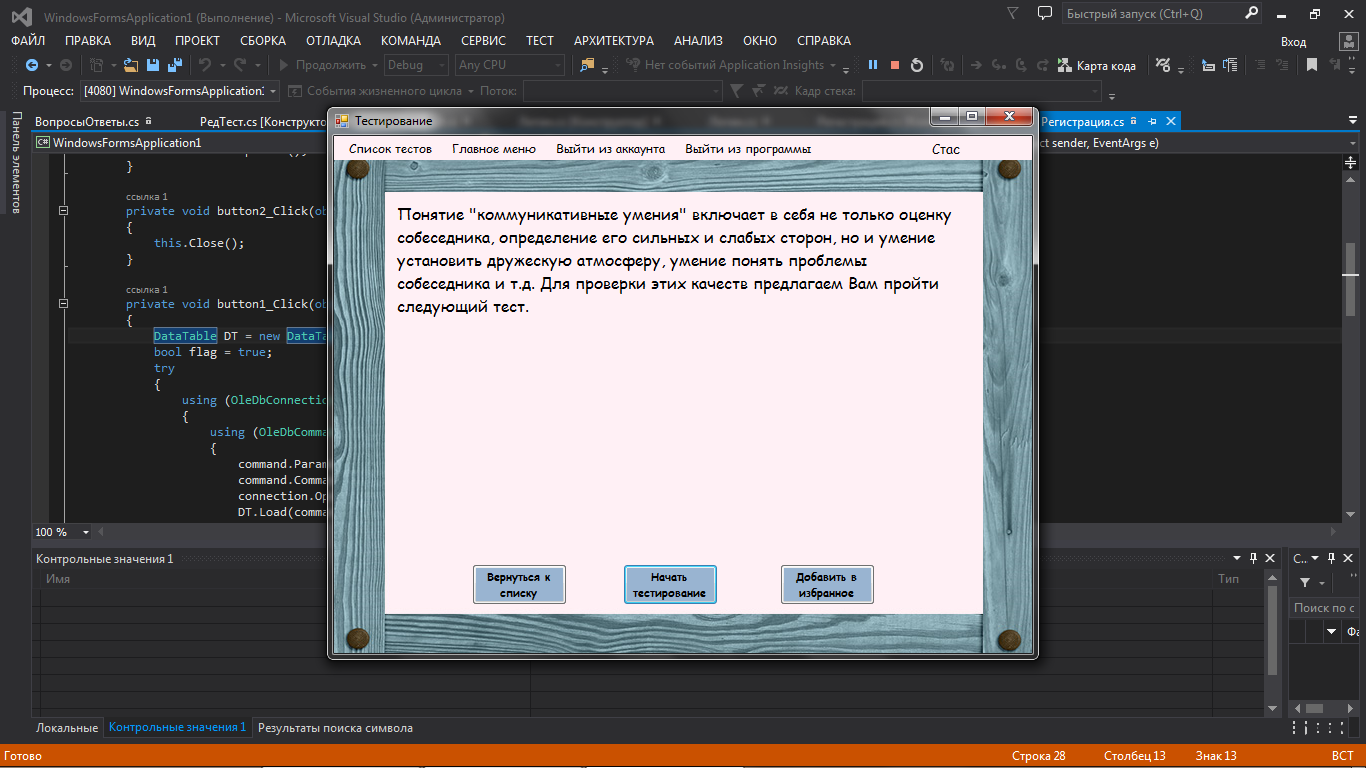


Рисунок 3.1 – Кнопка, расположенная в окне «Тестированя»

* menuStrip – элемент интерфейса пользователя, позволяющий выбрать одну из нескольких перечисленных функций программы (рисунок 3.2);



Рисунок 3.2 – Компонент menuStrip

* label, отображает текст на форме окна (рисунок 3.3);

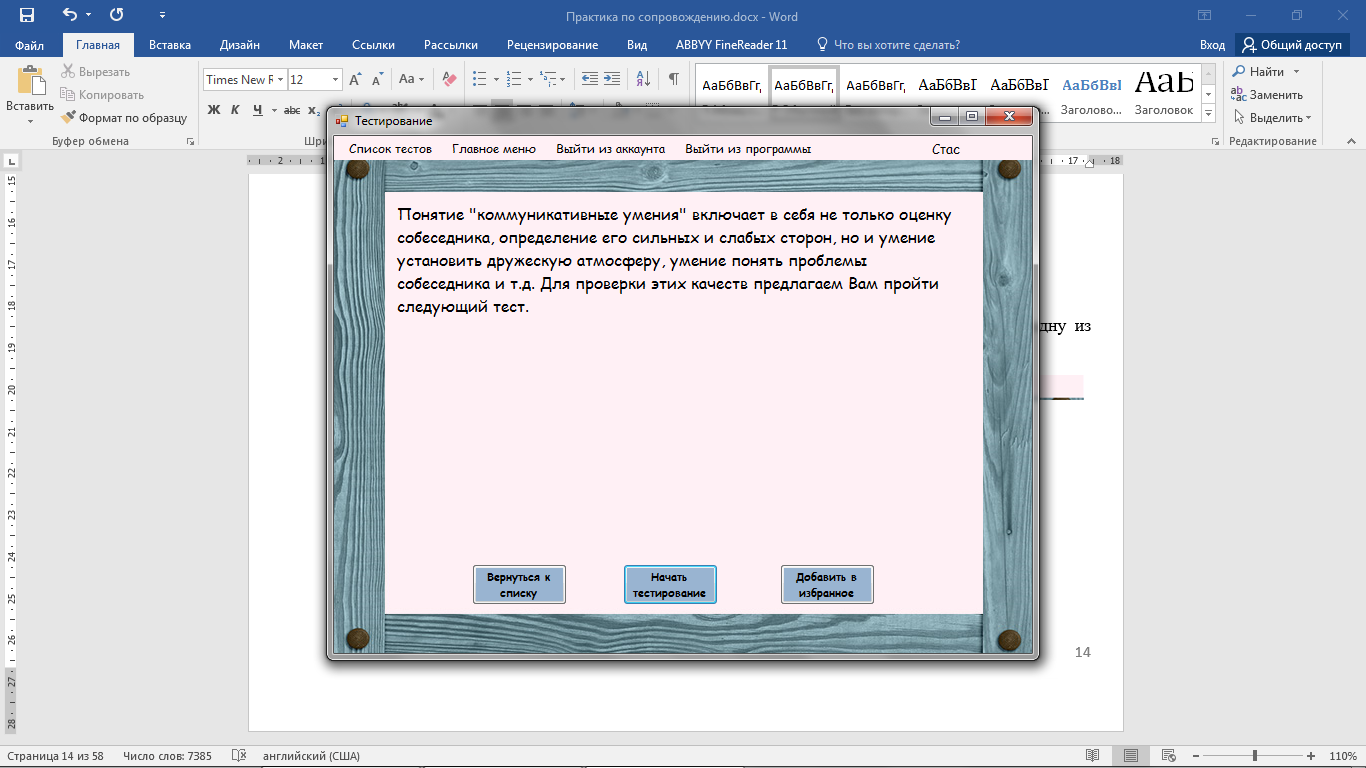


Рисунок 3.3 – Надпись, расположенная в окне «Тестирование»

* listBox -  элемент пользовательского интерфейса, который отображает определенные значения (рисунок 3.6);

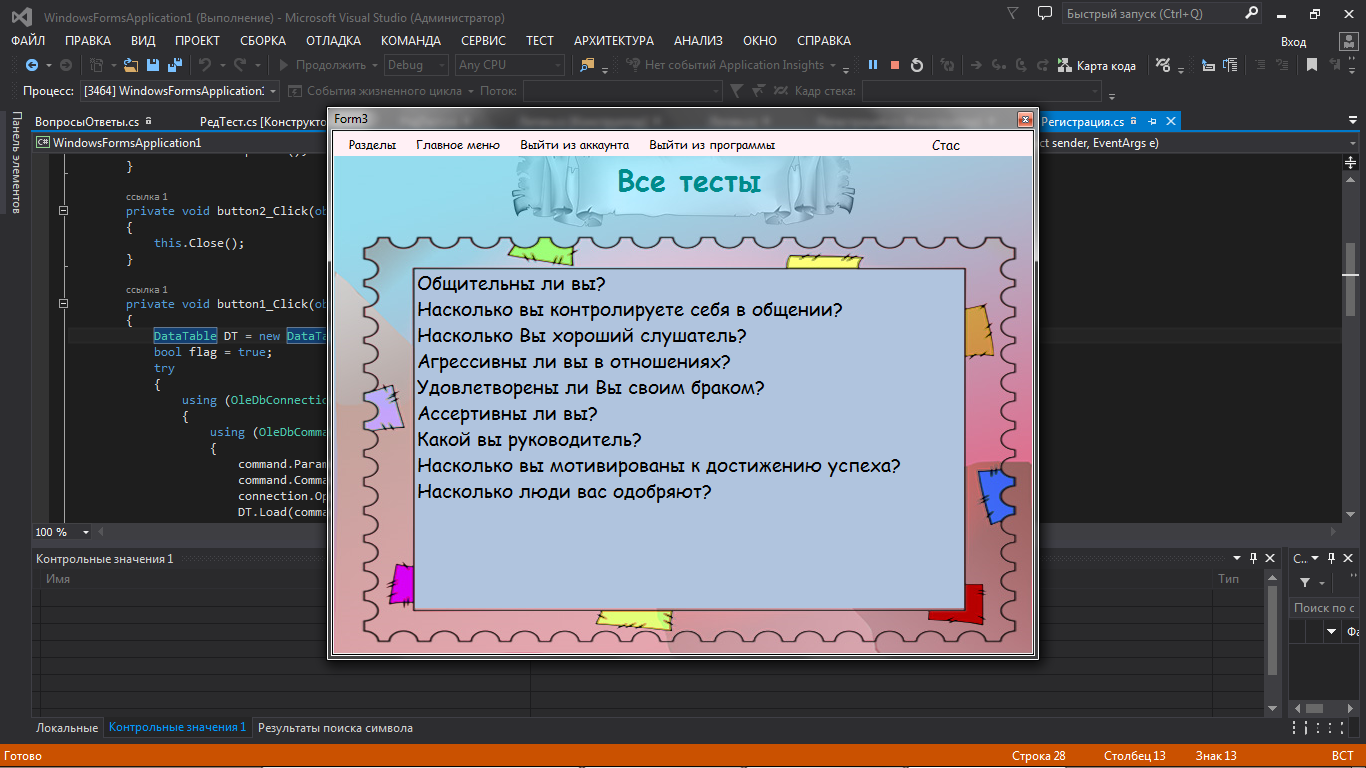


Рисунок 3.6 – listBox, выводящий список тестов

* textBox - Предоставляет элемент управления "текстовое поле" Windows. (рисунок 3.7);

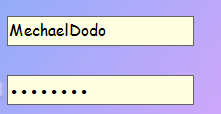


Рисунок 3.7 – Поля авторизации

* panel – Служит для группировки элементов управления. (рисунок 3.8);

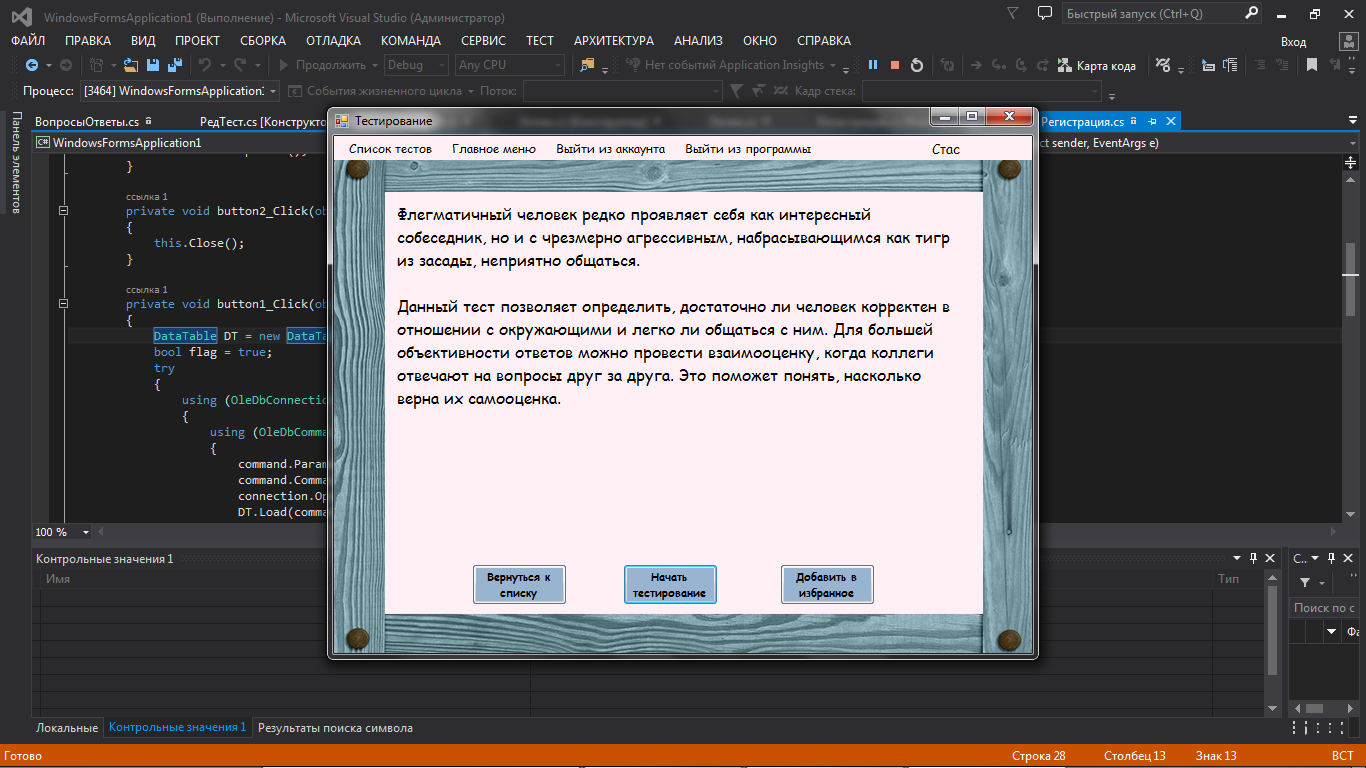


Рисунок 3.8 – Пример панели в окне «Тестирование»

## 3.3 Структуры данных (входные, выходные, постоянные)

Входные данные:

* логин и пароль пользователя,
* логин и пароль администратора,
* изменение в тесты
* имя выбранного теста
* загружаемый тест
* ответы на тестовые вопросы.

Выходные данные:

* результат прохождения теста,

Постоянные данные:

* База данных со следующими таблицами: пользователи, тесты, вопросы, ответы, характеристики, избранные тесты, результаты. Все данные получаются с помощью запросов. Редактирование и добавление также реализовано запросами в программном коде.

Постоянное поддержание базы данных обеспечит наиболее оптимальное взаимодействие с программным средством.

# Тестирование

Тестирование – процесс проверки программного обеспечения с целью выявления и предотвращения дефектов, оценки соответствия требованиям заказчика.

Баг (дефект) – ошибка в коде, которая может привести к тому, что программа будет работать не так, как ожидается.

Тестирование программного обеспечения – это креативная и интеллектуальная работа. Разработка правильных и эффективных тестов – достаточно непростое занятие. Принципы тестирования, представленные ниже, были разработаны в последние 40 лет и являются общим руководством для тестирования в целом.

* тестирование показывает наличие дефектов. Оно может показать наличие дефектов в программе, но не доказать их отсутствие. Тем не менее, важно составлять тест-кейсы, которые будут находить как можно больше багов. Таким образом, при должном тестовом покрытии, тестирование позволяет снизить вероятность наличия дефектов в программном обеспечении. В то же время, даже если дефекты не были найдены в процессе тестирования, нельзя утверждать, что их нет;
* исчерпывающее тестирование невозможно. Невозможно провести исчерпывающее тестирование, которое бы покрывало все комбинации пользовательского ввода и состояний системы, за исключениям совсем уж примитивных случаев. Вместо этого необходимо использовать анализ рисков и расстановку приоритетов, что позволит более эффективно распределять усилия по обеспечению качества ПО.
* раннее тестирование. Тестирование должно начинаться как можно раньше в жизненном цикле разработки программного обеспечения, и его усилия  должны быть сконцентрированы на определенных целях.
* скопление дефектов. Разные модули системы могут содержать разное количество дефектов – то есть, плотность скопления дефектов в разных элементах программы может отличаться. Усилия по тестированию должны распределяться пропорционально фактической плотности дефектов. В основном, большую часть критических дефектов находят в ограниченном количестве модулей. Это проявление принципа Парето: 80% проблем содержатся в 20% модулей.
* парадокс пестицида. Прогоняя одни и те же тесты вновь и вновь, Вы столкнетесь с тем, что они находят все меньше новых ошибок. Поскольку система эволюционирует, многие из ранее найденных дефектов исправляют и старые тест-кейсы больше не срабатывают. Чтобы преодолеть этот парадокс, необходимо периодически вносить изменения в используемые наборы тестов, рецензировать и корректировать их с тем, чтобы они отвечали новому состоянию системы и позволяли находить как можно большее количество дефектов.
* тестирование зависит от контекста. Выбор методологии, техники и типа тестирования будет напрямую зависеть от природы самой программы. Например, программное обеспечение для медицинских нужд требует гораздо более строгой и тщательной проверки, чем, скажем, компьютерная игра. Из тех же соображений, сайт с большой посещаемостью должен пройти через серьезное тестирование производительности, чтобы показать возможность работы в условиях высокой нагрузки.
* заблуждение об отсутствии ошибок. Тот факт, что тестирование не обнаружило дефектов, еще не значит, что программа готова к релизу. Нахождение и исправление дефектов будут не важны, если система окажется неудобной в использовании, и не будет удовлетворять ожиданиям и потребностям пользователя.

Исполнение тестов необходимо, но не менее важны и сопровождающие действия – планирование и документирование процесса. В обязанности тестировщика входит разработка тестовых сценариев, а также подготовка тестирования и оценка его результатов. Таким образом можно выделить следующие процессы тестирования:

* планирование и управление - определение целей тестирования и описание задач тестирования для достижения этих целей; сопоставление текущей ситуации в процессе тестирования с планом и составление отчетности;
* анализ и проектирование - это процесс написания тестовых сценариев и условий на основе общих целей тестирования;
* внедрение и реализация - написание тест-кейсов, на основе написанных ранее тестовых сценариев, собирается необходимая для проведения тестов информация, подготавливается тестовое окружение и запускаются тесты;
* оценка критериев выхода и написание отчетов – проверка, было ли проведено достаточное количество тестов, достигнута ли нужная степень обеспечения качества системы; убедится в том, что нет необходимости проводить дополнительные тесты. Если все же такая необходимость есть, возможно, потребуется изменить установленный критерий выхода;
* действия по завершению тестирования - сборка, систематизация и анализ информации о результатах тестирования.

Тестирование программного обеспечения имеет следующие уровни:

* компонентное - поиск дефектов и проверка функциональностей (модулей, программ, объектов, классов), которые можно протестировать изолированно;
* интеграционное - проверяет взаимодействие между компонентами, взаимодействие различных частей системы (ОС, файловая система, аппаратное обеспечение);
* системное – тестирование поведения тестового объекта как целостной системы или продукта;
* приемочное, в свою очередь, подразделяется на пользовательское, эксплуатационное, контрактное и правовое, альфа и бета. Заключается в проверке работоспособности системы, частей системы или отдельных нефункциональных характеристик системы.

Так же различают следующие типы тестирования:

* функциональное – проверяет, что система делает. Проводится тестирование внешнего поведения ПО (black-box testing);
* нефункциональное – проверяет, как система работает. Сюда входят нагрузочное тестирование, тестирование производительности, стресс-тестирование, тестирование удобства использования (в большинстве случаев black-box testing);
* структурное - тестирование, основанное на анализе внутренней структуры компонента или системы (white-box testing);
* тестирование изменений включает в себя подтверждение и регрессионное тестирование. Подтверждение – проверка того, что дефект действительно исправлен. Регрессионное тестирование – проверка, что ранее работающая корректно функциональность все еще работает.

Тестирование программы проходило в ходе и после ее разработки. При тестировании использовалось компонентное, системное, интеграционное и приёмочное тестирование. Меньше всего времени было затрачено на приёмочное тестирование, потому что ежедневно просматривались требования к данному программному обеспечению и продумывались варианты оптимальной реализации. Также использовались методы тестирования чёрным и белым ящиком, что весьма помогло в поиске дефектов.

При разработке в основном использовалось функциональное и нефункциональное тестирование. Были разработаны тест-кейсы и чек-лист, представленные в приложениях Б и В соответственно.

# 5 Применение

## 5.1 Установка приложения и требования к аппаратно-программным ресурсам ПК

Установка приложения не требуется.

Современные компьютеры становятся всё более производительными, и всё меньше программ могут полностью загрузить все аппаратные ресурсы системы.

Программа была разработана на компьютере со следующей конфигурацией:

* процессор: Intel Core i3-3110M;
* оперативная память – 4 GВ;
* винчестер – HDD –1 TB;
* операционная система: Windows 7 Профессиональная

Для оптимальной работы программы требуется персональный компьютер со следующими системными характеристиками и выше:

* процессор AMD C-60 APU with Radeou(tm) HD Graphics 1.00 GHz;
* 800 MБ ОЗУ;
* видеокарта на 64 МВ;
* 100 МВ свободного дискового пространства;
* монитор с разрешением 1024х768;
* операционная система MicrosoftWindows XP/Vista/Seven/8.

Соблюдение вышеперечисленных системных требований приведёт к оптимальной работе программы, не влияя на работу других приложений.

## 5.2 Демонстрационный пример работы программы

Данная программа предназначена для автоматизации процесса психологического тестирования.

Для использования данного ПО следует запустить файл WindowsFormsApplication1.exe.

После запуска приложения на экране появится окно приветствия, откуда пользователь может либо авторизироваться, либо зарегистрироваться, либо ознакомится со справочной информацией:

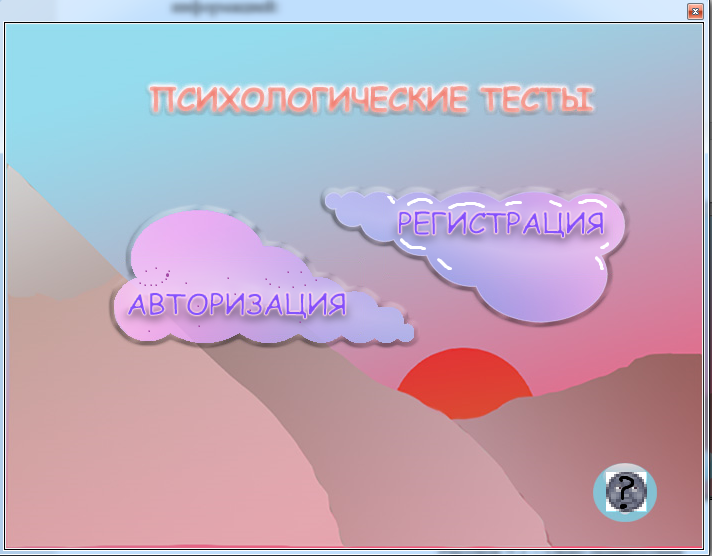


Рисунок 5.1 – Окно приветствия

Справка.

После нажатия на пиктограму в правом нижнем углу экрана откроется окно справки, где пользователь может ознакомиться с общей информацией о программном средстве (рисунок 5.2).

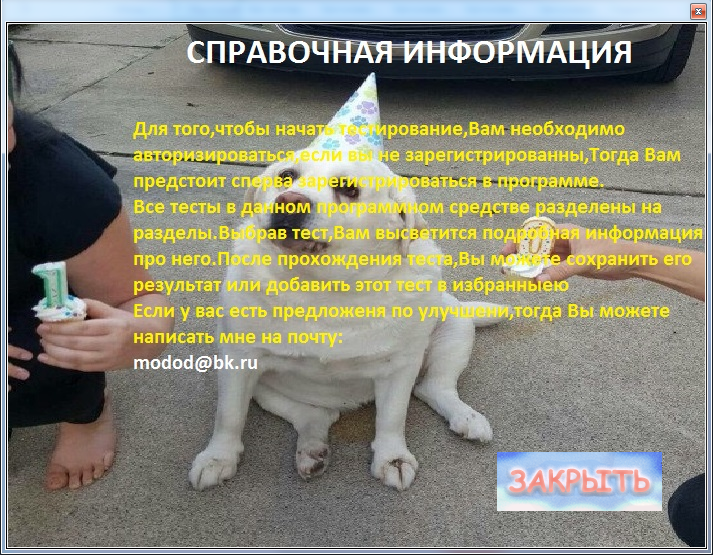


Рисунок 5.2 – Окно справки

После ознакомления со справочной информацией пользователь может зарегистрироваться, если у него нету аккаунта или авторизироваться (рисунки 5.3, 5.4).

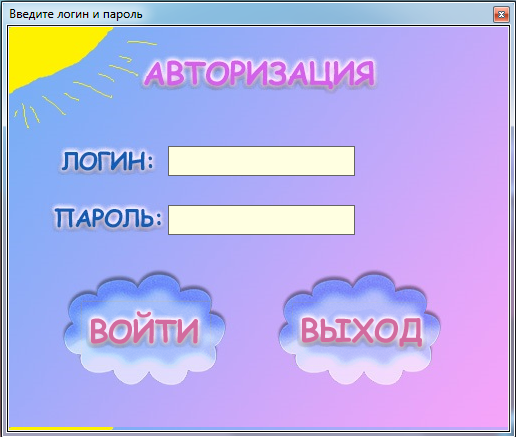


Рисунок 5.3 – Окно авторизации

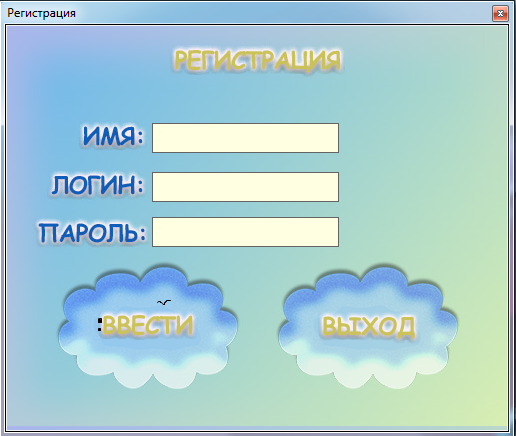


Рисунок 5.4 – Окно регистрации

После авторизации или регистрации, пользователю покажется главное меню. Если пользователь обладает правами администратора, то ему будет отображена кнопка «Редактировать тесты». В данном случае вход совершен под администратором (рисунок 5.5).



Рисунок 5.5 – Главное меню

После нажатия на кнопку «Тестирование», необходимо выбрать раздел. (рисунок 5.6)

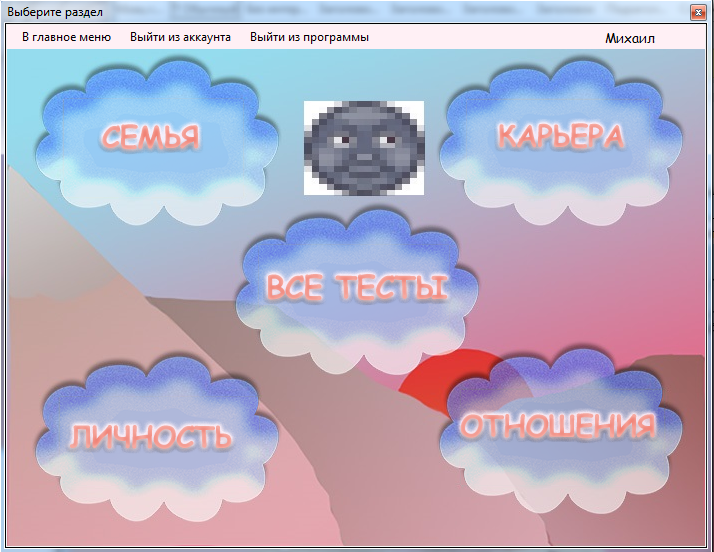


Рисунок 5.6 – Форма разделов

Выбрав раздел, откроется список всех тестов, расположенных в данном разделе. Для демонстрации был выбран раздел «Все тесты». (рисунок 5.7)

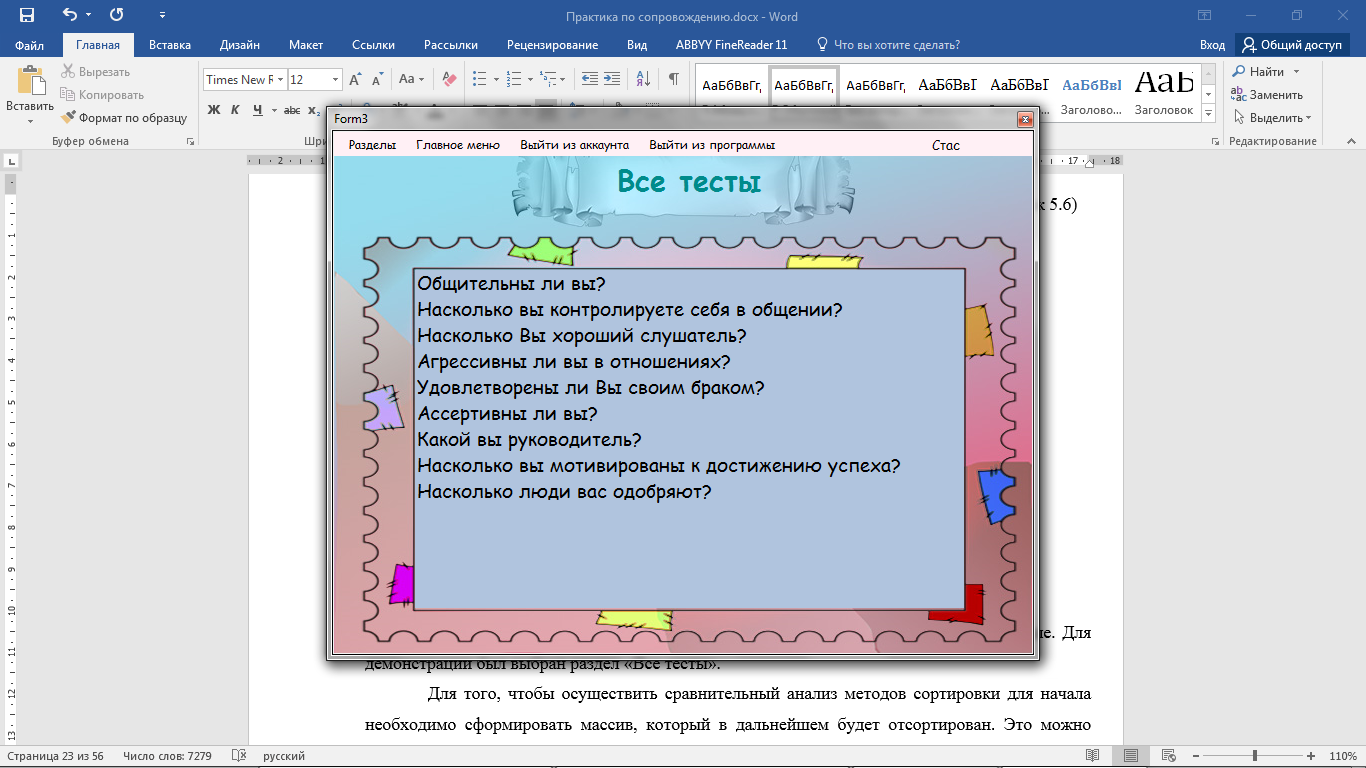


Рисунок 5.7 – Список тестов

Даллее из приведенного списка необходимо выбрать сам тест. После нажатия на элемент списка, нам откроется форма процесса тестирования (рисунок 5.8). Сперва нам выведется описание теста, после ознакомления с описанием, можно вернуться к списку, начать тестирование или добавить в избранное (чтобы пройти потом, например). После нажатия на кнопку «Начать тестирование», начинают выводиться тестовые вопросы и варианты ответов на них (рисунок 5.9). Когда на все вопросы есть ответы, выводится результат, который можно сохранить. Также можно добавитаь этот тест в избранное (рисунок 5.10).

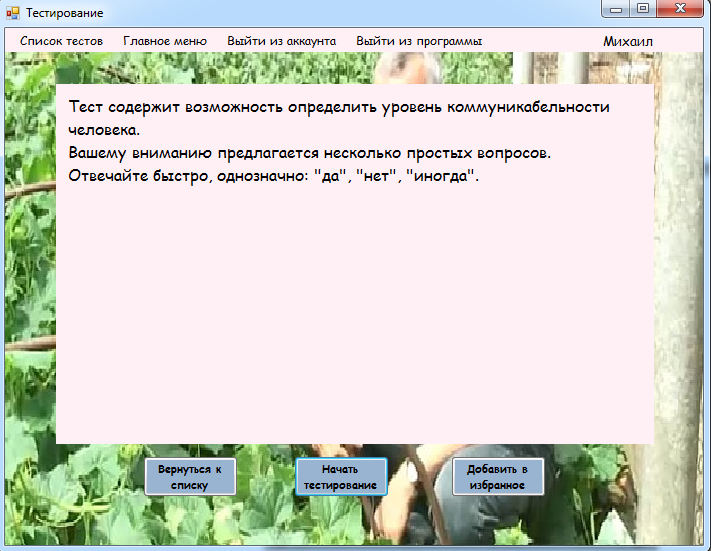


Рисунок 5.8 – Начальное окно тестирования

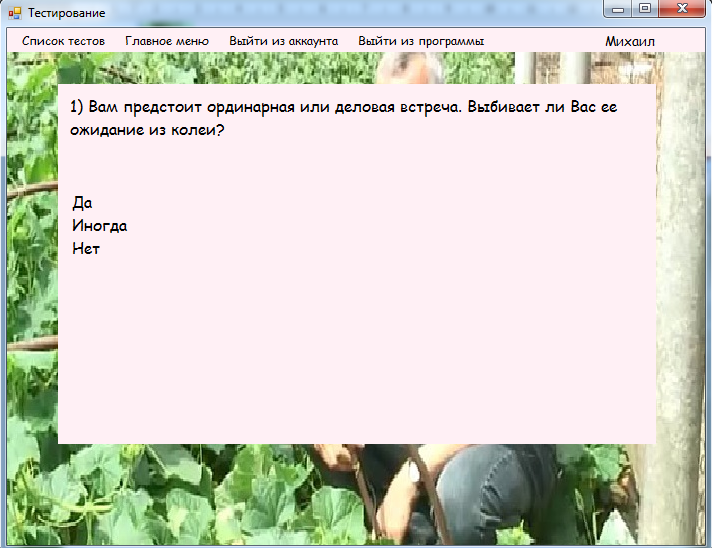


Рисунок 5.9 – Процесс тестирования

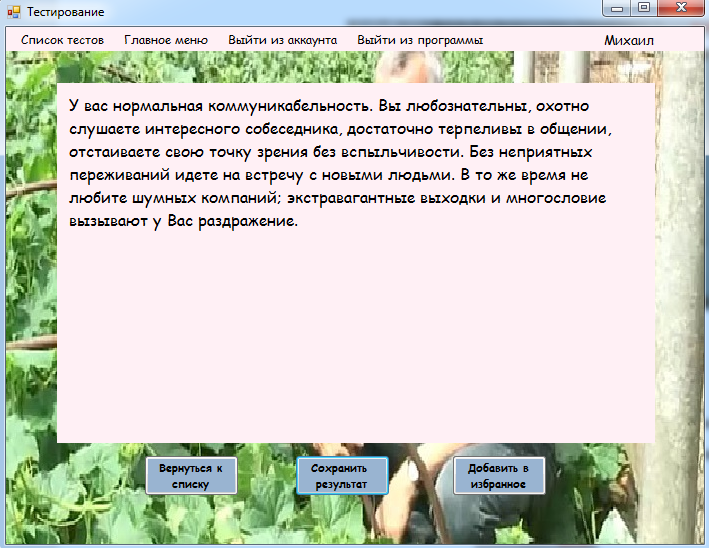


Рисунок 5.10 – Выведеный результат

При нажатии на кнопку «сохранить результат» высветится окно, что результат сохранен (рисунок 5.11). В случае, если по данному тесту уже имеется результат, нам предложат его перезаписать (рисунок 5.12).

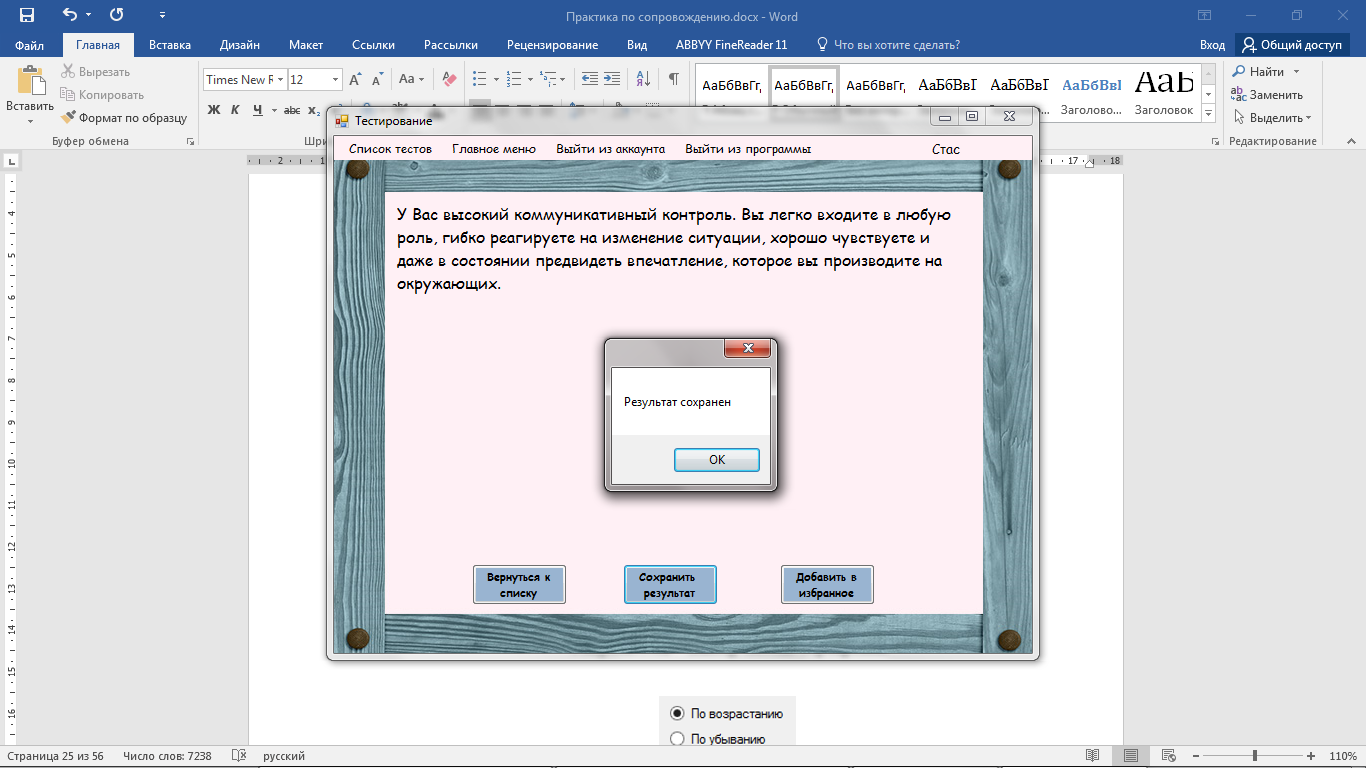


Рисунок 5.11 – Сообщениее о том, что результат сохранен

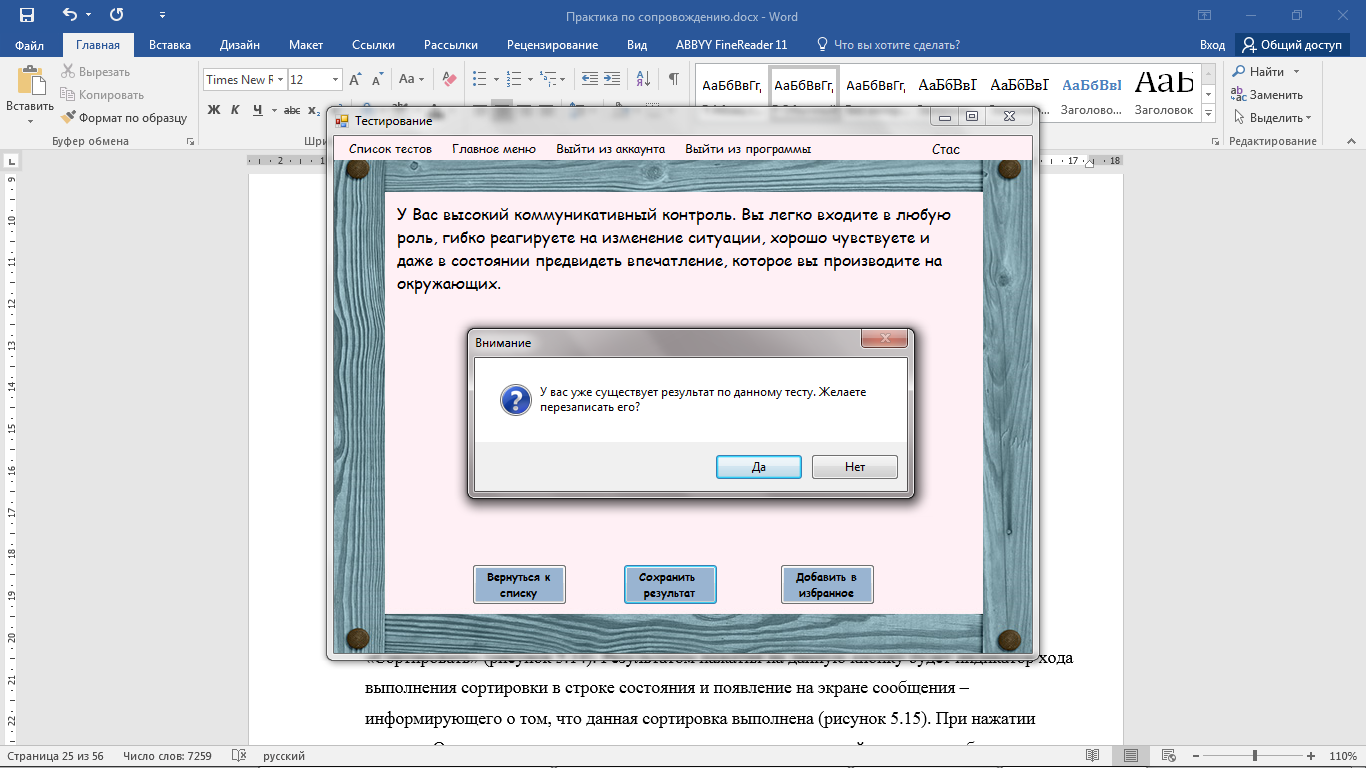


Рисунок 5.12 – Предложение перезаписать результат

Результат и избранные тесты можно просмотерть из соответствующих разделов главного меню.

# Заключение

Результатом данного курсового проекта стало разработанное программное средство «психологические тесты». Все предъявляемые требования были соблюдены в полном объеме:

* авторизация/регистрация;
* тестирования;
* добавление/редактирование/удавление тестов (режим администратора);
* сохранение результата;
* просмотр справки;
* добавление тестов в «избранное».

Был разработан оптимальный и удобный интерфейс, который позволяет быстро ознакомиться с программой и легко в ней ориентироваться. Программное средство имеет удобную и понятную систему меню, не имеет большого уровня вложенности окон, что позволяет комфортно взаимодействовать с ним.

Программное средство было протестированно на различные баги и дефекты. Результатом тестирования стал тест-кейс. Все тесты программа выполнила успешно.

Разрабатывалось программное средство по техническому заданию и диаграммам, которые представлены в приложениях.

В процессе разработки была спроектирована быза данных с использованием программного средства ACCESS. Работа с данной базой данных осуществляется запросами в программном коде программы.

В ходе исследовательской работы разработана функционирующая программа. Проектирование выполнено в системе Microsoft Visual Studio. В работе:

* + представлена информация о средствах и среде разработки;
  + представлен алгоритм работы программы;
  + приведен теоретический материал о компонентах, используемых в программе;
  + приведен анализ и тестирование программы с подробным описанием ее применения;
  + представлен листинг программы.

В ходе проделанной работы были улучшены навыки в создании MDI-приложений с использованием технологии Windows Forms на языке программирования C# в среде разработки Microsoft Visual Studio, написании технического задания к разрабатываемому ПС.

# Список использованных источников

1. Джеффри Рихтер. CLR via C#. Программирование на платформе Microsoft .NET Framework 4.5 на языке C#. — 4-е изд. — СПб.: Питер, 2013. — 896 с.
2. Медведев, В.И. M 42 Особенности объектно-ориентированного программирования на C++/CLI, C# и Java. 2-е изд., испр. и доп. - Казань: РИЦ «Школа», 2010.-444 c.
3. Основы программирования в Microsoft Visual C# 2010. Н.Б. Культин — СПб.: БХВ-Петербург, 2011. — 368 с.: ил.+ CD-ROM
4. Стиллмен, Э., Грин Дж.С80 Изучаем C#. 3-е изд. — СПб.: Питер, 2014. — 816 с.: ил. — (Серия «Head First O’Reilly»)
5. Технология разработки программного обеспечения: учеб. программа, метод. указания и контрол. задания для учащихся заочной формы получения образования специальности 2-40 01 01 «Программное обеспечение информационных технологий» / сост. Т.М.Тарасова. – Минск : МГВРК, 2013. – 64 с.
6. Э. Гамма, Р. Хелм, Р. Джонсон, Дж. Влиссидес. Приемы объектно-ориентированного проектирования. Паттерны проектирования  «Питер», 1994. — P. 395.
7. Beizer, Boris. (1995). Black-Box Testing: Techniques for Functional Testing of Software and Systems. New York: Wiley.

# Приложение А Код программы

Модуль «Тестирование»

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Data.OleDb;

namespace WindowsFormsApplication1

{

public partial class Тестирование : Form

{

public Тестирование()

{

InitializeComponent();

}

int num;

int ball;

int test;

DataTable Answers;

DataTable DT;

public Тестирование(DataTable DT,int test)

{

InitializeComponent();

label1.Text = User.name;

panel1.AutoScroll = true;

this.DT = DT;

this.test = test;

Answers = new DataTable();

if (Test.razd == 6){

button1.Text = "Назад";

button2.Visible = false;

button3.Visible = false;

try

{

using (OleDbConnection connection = new OleDbConnection("Provider=Microsoft.ACE.OLEDB.12.0;Data Source=Users.accdb"))

{

using (OleDbCommand command = new OleDbCommand("SELECT Characteristic FROM UsersResults where id\_user = " + User.id + " AND id\_test = "+test+"", connection))

{

command.CommandType = CommandType.Text;

connection.Open();

Answers.Load(command.ExecuteReader());

label2.Text = Convert.ToString(Answers.Rows[0]["Characteristic"]);

}

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

else

{

button1.Text = "Начать тестирование";

num = 0;

try

{

using (OleDbConnection connection = new OleDbConnection("Provider=Microsoft.ACE.OLEDB.12.0;Data Source=Users.accdb"))

{

using (OleDbCommand command = new OleDbCommand("SELECT Описание FROM Test Where ID\_test = @test", connection))

{

command.Parameters.AddWithValue("@test", test);

command.CommandType = CommandType.Text;

connection.Open();

Answers.Load(command.ExecuteReader());

label2.Text = Convert.ToString(Answers.Rows[0]["Описание"]);

}

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

}

private void выйтиИзПрограммыToolStripMenuItem2\_Click(object sender, EventArgs e)

{

Application.Exit();

}

private void listBox1\_Click(object sender, EventArgs e)

{

if (listBox1.SelectedIndex >= 0)

{

ball = ball + Convert.ToInt32(Answers.Rows[listBox1.SelectedIndex]["Ball"]);

listBox1.Items.Clear();

num++;

if (num < DT.Rows.Count)

{

NewQuest(num);

}

else

{

Rezult(ball);

}

}

}

void NewQuest(int num)

{

Answers.Clear();

label2.Text = Convert.ToString(num+1)+") "+ Convert.ToString(DT.Rows[num]["Question"]);

try

{

using (OleDbConnection connection = new OleDbConnection("Provider=Microsoft.ACE.OLEDB.12.0;Data Source=Users.accdb"))

{

using (OleDbCommand command = new OleDbCommand("SELECT Answer, Ball FROM Answers Where ID\_quest = @Num", connection))

{

command.Parameters.AddWithValue("@Num", DT.Rows[num]["ID\_quest"]);

command.CommandType = CommandType.Text;

connection.Open();

Answers.Load(command.ExecuteReader());

for (int i = 0; i < Answers.Rows.Count; i++)

{

listBox1.Items.Add(Convert.ToString(Answers.Rows[i]["Answer"]));

}

}

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

void Rezult(int ball)

{

Answers.Clear();

button1.Text = "Сохранить \nрезультат";

button1.Visible = true;

button2.Visible = true;

button3.Visible = true;

try

{

using (OleDbConnection connection = new OleDbConnection("Provider=Microsoft.ACE.OLEDB.12.0;Data Source=Users.accdb"))

{

using (OleDbCommand command = new OleDbCommand("SELECT Value, Balls FROM Characteristic Where ID\_test = @id", connection))

{

command.Parameters.AddWithValue("@id", test);

command.CommandType = CommandType.Text;

connection.Open();

Answers.Load(command.ExecuteReader());

BubbleSort(Answers);

if (Answers.Rows.Count == 1)

{

label2.Text = Convert.ToString(Answers.Rows[0]["value"]) + "\n\nВаш балл: " + Convert.ToString(ball);

listBox1.Visible = false;

}

for (int i = 0; i < Answers.Rows.Count-1; i++)

{

if (Convert.ToString(Answers.Rows[i]["Value"]) == Convert.ToString(Answers.Rows[i+1]["Value"]))

{

if ((ball >= Convert.ToInt32(Answers.Rows[i]["Balls"]) && ball <= Convert.ToInt32(Answers.Rows[i + 1]["Balls"])))

{

label2.Text = Convert.ToString(Answers.Rows[i]["value"]);

listBox1.Visible = false;

break;

}

}

}

}

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void button1\_Click(object sender, EventArgs e)

{

if (num == 0)

{

button1.Visible = false;

button2.Visible = false;

button3.Visible = false;

label2.Text = Convert.ToString(1) + ") " + Convert.ToString(DT.Rows[0]["Question"]);

Answers.Clear();

listBox1.Visible = true;

try

{

using (OleDbConnection connection = new OleDbConnection("Provider=Microsoft.ACE.OLEDB.12.0;Data Source=Users.accdb"))

{

using (OleDbCommand command = new OleDbCommand("SELECT Answer, Ball FROM Answers Where ID\_quest = @Num", connection))

{

command.Parameters.AddWithValue("@Num", DT.Rows[0]["ID\_quest"]);

command.CommandType = CommandType.Text;

connection.Open();

Answers.Load(command.ExecuteReader());

for (int i = 0; i < Answers.Rows.Count; i++)

{

listBox1.Items.Add(Convert.ToString(Answers.Rows[i]["Answer"]));

}

}

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

else

{

try

{

if (Is("UsersResults") == true){

using (OleDbConnection connection = new OleDbConnection("Provider=Microsoft.ACE.OLEDB.12.0;Data Source=Users.accdb"))

{

using (OleDbCommand command = new OleDbCommand("INSERT INTO UsersResults VALUES (@id\_us, @id\_test, @Char)", connection))

{

command.Parameters.AddWithValue("@id\_us", User.id);

command.Parameters.AddWithValue("@id\_test", test);

command.Parameters.AddWithValue("@Char", label2.Text);

connection.Open();

command.ExecuteReader();

MessageBox.Show("Результат сохранен");

}

}

}

else

{

DialogResult result = MessageBox.Show("У вас уже существует результат по данному тесту. Желаете перезаписать его?", "Внимание", MessageBoxButtons.YesNo, MessageBoxIcon.Question);

if (result == DialogResult.Yes)

{

using (OleDbConnection connection = new OleDbConnection("Provider=Microsoft.ACE.OLEDB.12.0;Data Source=Users.accdb"))

{

using (OleDbCommand command = new OleDbCommand("Update UsersResults set ID\_user = @id\_us, id\_test = @id\_test, Characteristic = @Char Where ID\_user = @id\_us AND id\_test = @id\_test", connection))

{

command.Parameters.AddWithValue("@id\_us", User.id);

command.Parameters.AddWithValue("@id\_test", test);

command.Parameters.AddWithValue("@Char", label2.Text);

connection.Open();

command.ExecuteReader();

MessageBox.Show("Результат перезаписан");

}

}

}

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

if (Test.razd == 6)

{

Forms.f3.Show();

this.Close();

}

}

private void button3\_Click(object sender, EventArgs e)

{

try

{

if (Is("UsersLikes") == true)

{

using (OleDbConnection connection = new OleDbConnection("Provider=Microsoft.ACE.OLEDB.12.0;Data Source=Users.accdb"))

{

using (OleDbCommand command = new OleDbCommand("INSERT INTO UsersLikes VALUES (@id\_us, @id\_test)", connection))

{

command.Parameters.AddWithValue("@id\_us", User.id);

command.Parameters.AddWithValue("@id\_test", test);

connection.Open();

command.ExecuteReader();

MessageBox.Show("Теперь данный тест будет отображаться в разделе \"Избранные тесты\" ");

}

}

}

else

{

DialogResult result = MessageBox.Show("Данный тест уже в отмечен, как избранный. Желаете удалить его из избранных тестов?", "Внимание", MessageBoxButtons.YesNo, MessageBoxIcon.Question);

if (result == DialogResult.Yes)

{

using (OleDbConnection connection = new OleDbConnection("Provider=Microsoft.ACE.OLEDB.12.0;Data Source=Users.accdb"))

{

using (OleDbCommand command = new OleDbCommand("DELETE FROM UsersLikes Where ID\_user = @id\_us AND ID\_test = @id\_test", connection))

{

if (Test.razd == 5)

{

Forms.f3.listBox1.Items.RemoveAt(Test.index);

Forms.f3.ind.RemoveAt(Test.index);

}

command.Parameters.AddWithValue("@id\_us", User.id);

command.Parameters.AddWithValue("@id\_test", test);

connection.Open();

command.ExecuteReader();

MessageBox.Show("Тест удален из раздела \"избранные тесты\" ");

}

}

}

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void button2\_Click(object sender, EventArgs e)

{

Forms.f3.Show();

this.Close();

}

private void выйтиИзАккаунтаToolStripMenuItem\_Click(object sender, EventArgs e)

{

this.Close();

Forms.f3.Show();

}

private void выйтиИзПрограммыToolStripMenuItem\_Click(object sender, EventArgs e)

{

this.Close();

Forms.MainMenu.Show();

Forms.f3 = null;

}

private void выйтиИзПрограммыToolStripMenuItem1\_Click(object sender, EventArgs e)

{

this.Close();

Forms.Form1.Show();

Forms.f3 = null;

User.admin = false;

}

void BubbleSort(DataTable A)

{

for (int i = 0; i < A.Rows.Count; i++)

{

for (int j = 0; j < A.Rows.Count - i - 1; j++)

{

if (Convert.ToInt32(A.Rows[j]["Balls"]) > Convert.ToInt32(A.Rows[j + 1]["Balls"]))

{

int temp = Convert.ToInt32(A.Rows[j]["Balls"]);

string temp1 = Convert.ToString(A.Rows[j]["Value"]);

A.Rows[j]["Balls"] = A.Rows[j+1]["Balls"];

A.Rows[j]["Value"] = A.Rows[j+1]["Value"];

A.Rows[j + 1]["Balls"] = temp;

A.Rows[j + 1]["Value"] = temp1;

}

}

}

}

bool Is(string table)

{

DataTable tab = new DataTable();

try

{

using (OleDbConnection connection = new OleDbConnection("Provider=Microsoft.ACE.OLEDB.12.0;Data Source=Users.accdb"))

{

using (OleDbCommand command = new OleDbCommand("Select ID\_user FROM "+table+" WHERE id\_user = @id\_us AND id\_test = @id\_test", connection))

{

command.Parameters.AddWithValue("@id\_us", User.id);

command.Parameters.AddWithValue("@id\_test", test);

connection.Open();

tab.Load(command.ExecuteReader());

if (tab.Rows.Count == 0){

return true;

}

else

{

return false;

}

}

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

return false;

}

}

}

}

Модуль «Регистрация»

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Data.OleDb;

namespace WindowsFormsApplication1

{

public partial class Регистрация : Form

{

public Регистрация()

{

InitializeComponent();

}

private void button2\_Click(object sender, EventArgs e)

{

this.Close();

}

private void button1\_Click(object sender, EventArgs e)

{

DataTable DT = new DataTable();

bool flag = true;

try

{

using (OleDbConnection connection = new OleDbConnection("Provider=Microsoft.ACE.OLEDB.12.0;Data Source=Users.accdb"))

{

using (OleDbCommand command = new OleDbCommand("SELECT Login FROM Users Where Login = @Login", connection))

{

command.Parameters.AddWithValue("@Login", textBox2.Text.ToLower());

command.CommandType = CommandType.Text;

connection.Open();

DT.Load(command.ExecuteReader());

if (DT.Rows.Count != 0)

{

MessageBox.Show("Такой логин уже существует");

flag = false;

}

if (textBox1.Text == "" || textBox2.Text == "" || textBox3.Text == "")

{

MessageBox.Show("Все поля обязательны для заполнения");

flag = false;

}

if (flag == true)

{

command.Parameters.Clear();

command.CommandText = "INSERT INTO Users ([Name], Login, [Password]) VALUES (@Name, @Login, @Password);";

command.Parameters.AddWithValue("@Name", textBox1.Text);

command.Parameters.AddWithValue("@Login", textBox2.Text);

command.Parameters.AddWithValue("@Password", textBox3.Text);

command.ExecuteNonQuery();

command.CommandText = "SELECT id\_user FROM Users Where Login = '" + textBox2.Text + "'";

DT.Load(command.ExecuteReader());

User.id = Convert.ToInt32(DT.Rows[0]["id\_user"]);

User.name = textBox1.Text;

this.Close();

User.flag = true;

}

}

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

}

}

# 

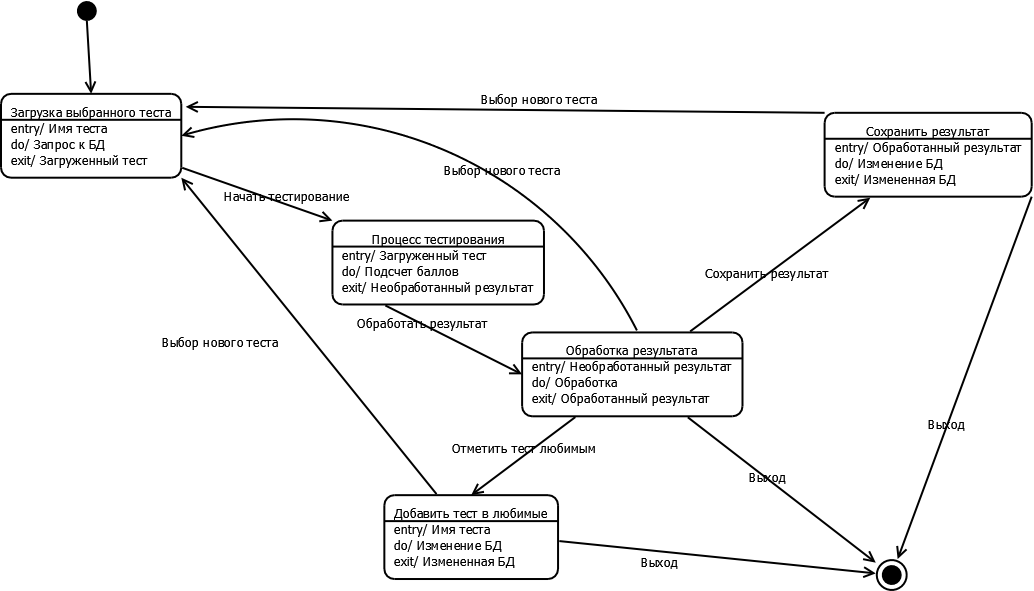
# Приложение Б Тест-кейс

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Модуль | Подмодуль/экран | Описание теста | Ожидание результата | Статус |
| ST\_001 | Приложение запущено | Окно приветствия/Форма авторизации | Авторизация обычного пользователя:  1) Нажать на кнопку авторизация  2) Ввести логин и пароль 3) Нажать на кнопку "Войти" | Если данные, введенные пользователем, существуют в БД, тогда пользователю откроется главное меню, иначе выскочит сообщение "Неверные данные" | Тестировано |
| ST\_002 | Прилжение запущено | Окно приветствия/Форма авторизации | Авторизация администратора:  1) Нажать на кнопку авторизация  2) Ввести логин и пароль 3) Нажать на кнопку "Войти" | Если данные, введенные пользователем, существуют в БД, тогда пользователю откроется главное меню, иначе выскочит сообщение "Неверные данные". Если у пользователя в БД установлен флаг администратора, тогда в главном меню появится кнопка "Редактировать тесты", которая не видна обычным пользователям | Тестировано |
| ST\_003 | Приложение запущено | Окно приветствия/Форма справки | Получение справочной информации:  1) Нажать на кнопку справки в правом нижнем углу окна приветствия 2) Получить справочную информацию | При нажатии на кнопку справки, откроется окно справки, где пользователь сможет ознакомиться с основной информацией о программе. | Тестировано |
| ST\_004 | Приложение запущено | Главное меню/Форма разделов/Форма списка | Отображение списка тестов, в зависимости от выбранного раздела: а) Открытие списка теста, из формы разделов:  1) В главном меню нажать на кнопку "Тестирование";  2) Выбрать один из приведенных разделов или же раздел "Все тесты";  3) Нажать на кнопку; б) Открытие списка избранных тестов:  1) В главном меню выбрать раздел "Избранные тесты"  2) Нажать на кнопку в) Открытие списка результатов:  1) В главном меню выбрать раздел "Результаты"  2) Нажать на кнопку | При нажатии на кнопку, произойдет обращение к БД, откуда, в зависимости от выбранного раздела, выберутся тесты и выведутся на экран | Тестировано |
| ST\_005 | Приложение запущено | Форма списка/Форма тестирования | Загрузка описания выбранного теста: 1) Нажать на имя теста, который вы хотите выбрать; 2) Получить описание | При нажатии на имя теста, сделается запрос к БД и пользователю выведется описание теста. | Тестировано |
| ST\_006 | Приложение запущено | Форма тестирования | Загрузка вопросов и вариантов ответов выбранного теста: 1) Нажать на кнопку "Начать тестирование" 2) Выбирать определенный вариант ответа, чтобы менялся вопрос и варианты ответов | При нажатии на кнопку "Начать тестирование", пользователю выведется первый вопрос и первые варианты ответов. При последующем выборе определенного варианта ответа, вопросы и варианты ответов будут меняться | Тестировано |
| ST\_007 | Приложение запущено | Форма тестирования | Обработка и вывод результата: 1) Ответить на все вопросы в тесте 2) Получить характеристику | Каждому варианту ответов, соответствует определенное количество баллов. При выборе того или иного варианта ответа, в специальную переменную накаплиается балл. После ответа на все вопросы, делается запрос к БД, после чего выбирается характеристика, соответствующая вашему баллу | Тестировано |
| ST\_008 | Приложение запущено | Форма тестирования | Сохранение результата в БД: 1) Пройти тест 2) Нажать на кнопку "Сохранить результат" 3) Если у вас уже имеется результат по данному тесту, нажать на кнопку "Перезаписать" или "Отмена" | После прохождения теста, на форме появится кнопку "Сохранить результат". При нажатии на нее, пользователю выведется сообщение, что результат сохранен. Если же в БД уже имеется результат по данному тесту, тогда вам высветится диалоговое окно с вопросом "Перезаписать результат". Если пользователь ответит "Да", тогда результат перезапишится, если "Отмена", тогда результат останется прежним. | Тестировано |
| ST\_09 | Приложение запущено | Форма тестирования | Добавление теста в избранное: 1) Нажать на кнопку "Добавить в избранное" 2) Если данный тест уже добавлен в избранное, нажать на кнопку "Удалить или "Отмена | Кнопка "Добавить в избранное" видна, когда показано описание теста и когда показан результат теста. При нажатии на нее, высветится сообщение "Тест добавлен в избранное". Если данный тест уже добавлен в избранное, то пользователю высветится диалог вопроса "Данный тест уже добавлен в избранное. Хотите удалить его". Если пользователь ответит "Да", тест удалится, иначе - ничего не изменится | Тестировано |
| ST\_010 | Приложение запущено | Главное меню/Форма списа/Форма тестирования | Просмотр сохраненных результатов: 1) Выбрать в главном меню раздел "Результаты" 2) Выбрать в списке тест 3) Просмотерть результат | Пользователю выведется его результат по запрашиваемому тесту | Тестировано |
| ST\_011 | Приложение запущено | Окно приветствия/Окно регистрации | Регистрация: 1) Нажать на кнопку "Регистрация" 2) Ввести имя, логин и пароль 3) Нажать на кнопку "Ввести" | Если введенный логин уже не занят, в БД занесутся данные о новом пользователе. После чего откроется главное меню | Тестировано |

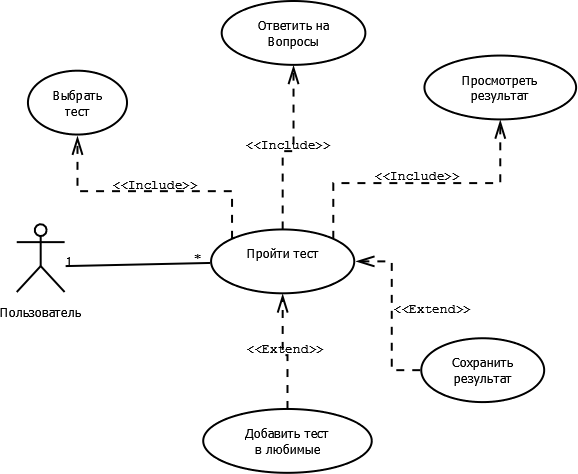
# Приложение В Чек-лист

|  |  |
| --- | --- |
| Проверка | Результат |
| Авторизация обычного пользователя | Ок |
| Авторизация администратора | Ок |
| Получение справочной информации | Ок |
| Отображение списка тестов, в зависимости от выбранного раздела | Ок |
| Загрузка описания выбранного теста | Ок |
| Загрузка вопросов и вариантов ответов выбранного теста | Ок |
| Обработка и вывод результата | Ок |
| Сохранение результата в БД | Ок |
| Добавление теста в избранное | Ок |
| Просмотр сохраненных результатов | Ок |
| Регистрация | Ок |

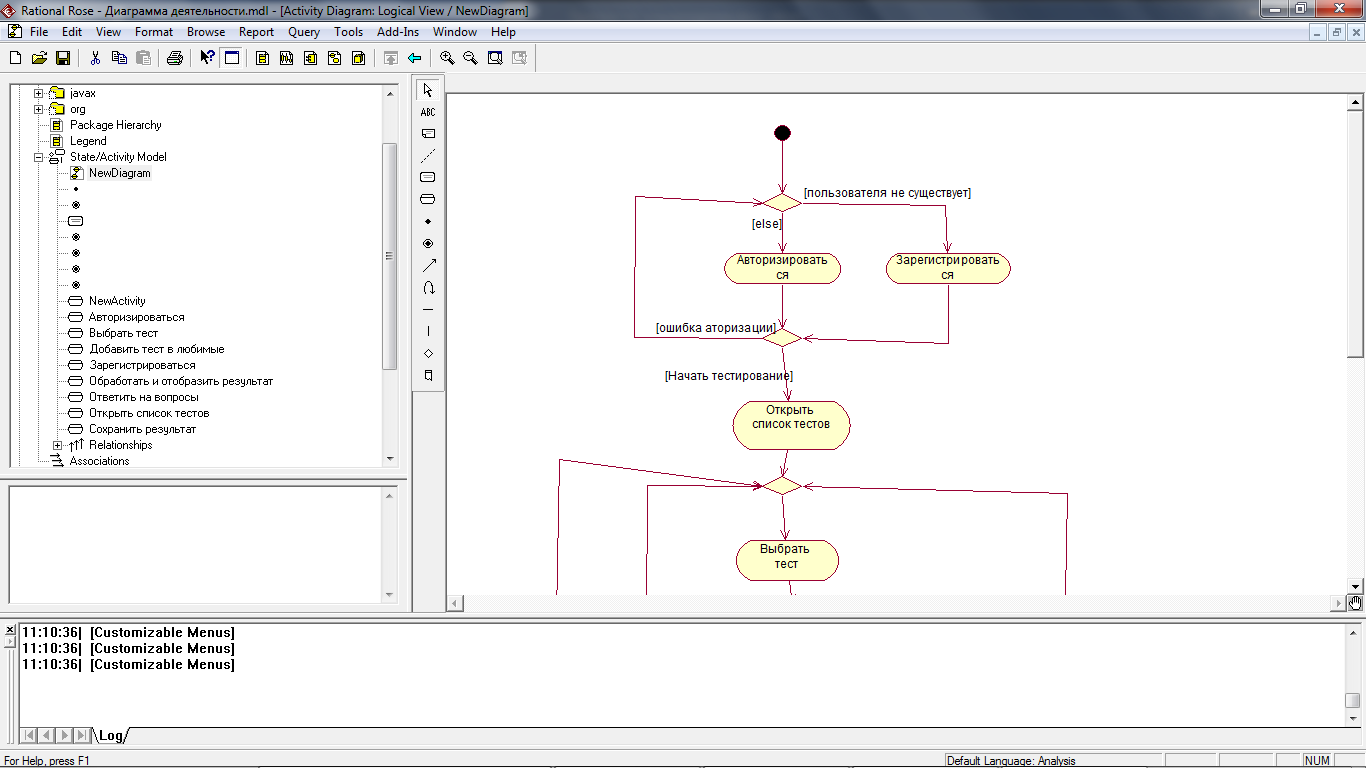
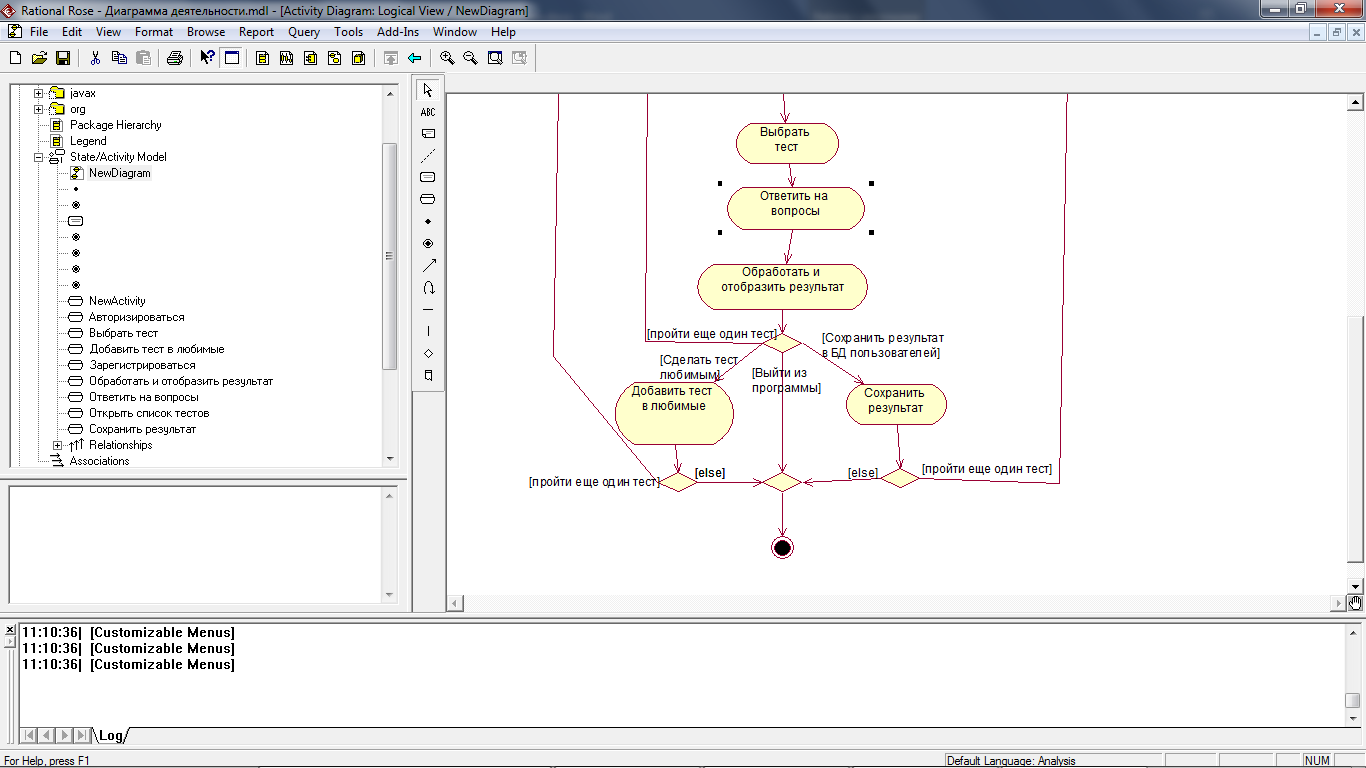
# Приложение Г Диаграмма состояний



# Приложение Д диаграмма вариантов использования



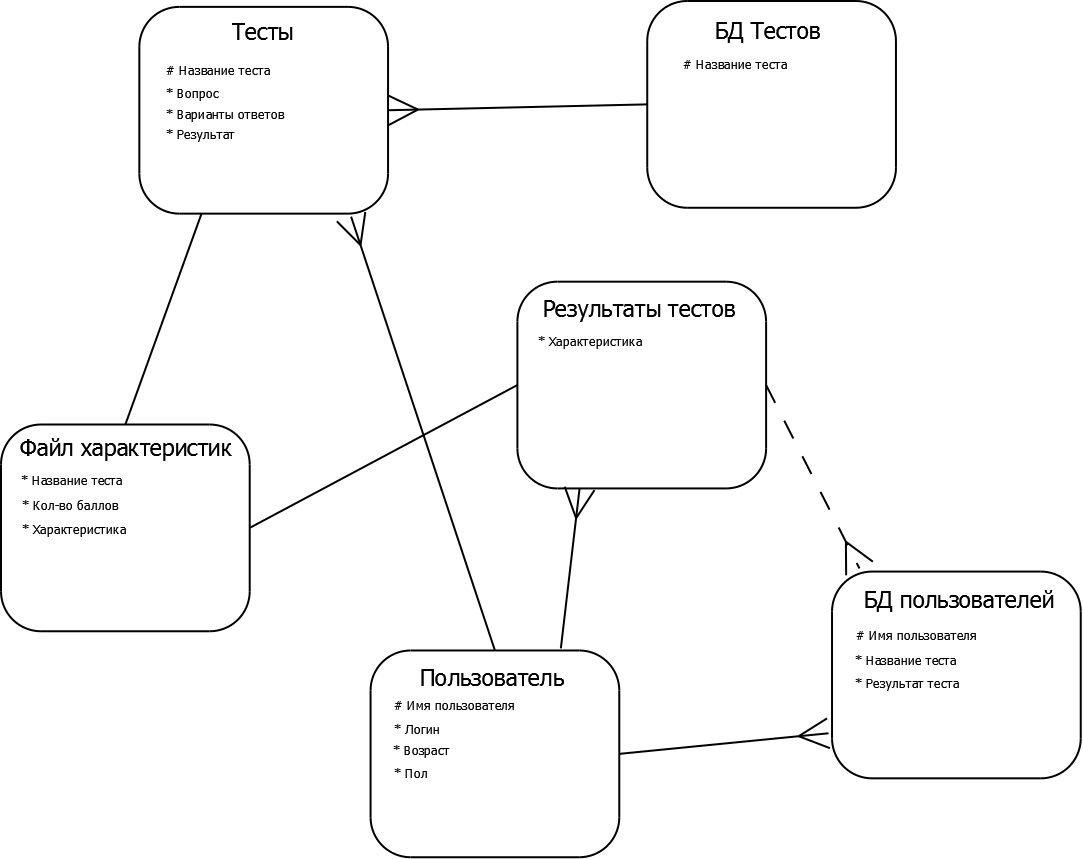
# Приложение Е Диаграмма деятельности



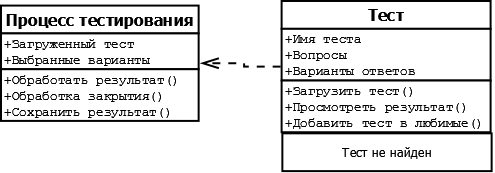
# Приложение Ж Диаграмма потоков данных



# Приложение З Диаграмма сущность - связь

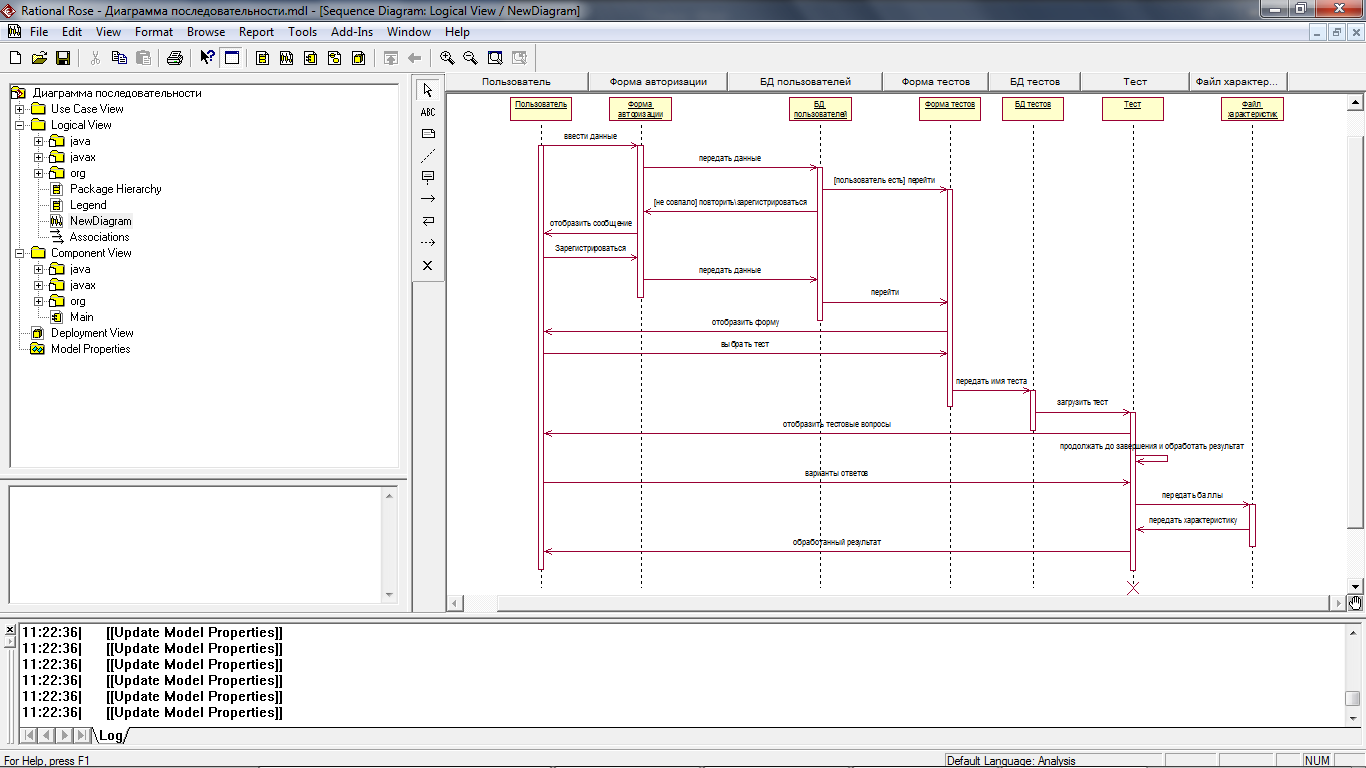


# Приложение И Диаграмма классов



# 

# Приложение К Диаграмма последовательностей



Приложение Л структурная схема