**3 РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ПРОГРАММНО – АППАРАТНОГО КОМПЛЕКСА ДЛЯ ИССЛЕДОВАНИЯ ПРОЦЕССОВ ПАМЯТИ**

**3.1 Обоснование выбора языка реализации и среды разработки**

Программный комплекс разрабатывался в среде программирования Microsoft Visual Studio 2015 на языке программирования C# с использованием XML.

Средой написания приложение является Microsoft Visual Studio – линейка продуктов компании Microsoft, включающих интегрированную среду разработки программного обеспечения и ряд других инструментальных средств, позволяющих разрабатывать как консольные приложения, так и приложения с графическим интерфейсом, в том числе с поддержкой технологии Windows Forms, а также веб-сайты, веб-приложения, веб-службы как в родном, так и в управляемом кодах для всех платформ, поддерживаемых Microsoft Windows, Windows Mobile, Windows CE, .NET Framework, Xbox, Windows Phone .NET Compact Framework и Microsoft Silverlight..

Visual Studio включает в себя редактор исходного кода с поддержкой технологии IntelliSense и возможностью простейшего рефакторинга кода. Встроенный отладчик может работать как отладчик уровня исходного кода, так и как отладчик машинного уровня. Остальные встраиваемые инструменты включают в себя редактор форм для упрощения создания графического интерфейса приложения, веб-редактор, дизайнер классов и дизайнер схемы базы данных. Visual Studio позволяет создавать и подключать сторонние дополнения (плагины) для расширения функциональности практически на каждом уровне, включая добавление поддержки систем контроля версий исходного кода, добавление новых наборов инструментов или инструментов для прочих аспектов процесса разработки программного обеспечения. [6]

Microsoft .Net Framework является так называемой программной платформой. Архитектура платформы состоит из двух частей. Основой является исполняющая среда Common Language Runtime (CLR), которая может выполнять как обычные программы, так и серверные приложения. Вторая, не менее важная часть, это библиотека классов Framework Class Library (FCL), содержащая в себе множество компонентов для работы с базами данных, сетью, вводом/выводом, файлами, пользовательским интерфейсом и т.д. Это позволяет разработчику не заниматься низкоуровневым программированием, а использовать уже готовые классы.

Важные части библиотеки классов:

* Windows Forms — отвечает за разработку графического интерфейса. Фактически является обёрткой над Win32 API.
* ADO.NET — предоставляет доступ данным. В основном используется для работы с базами данных.
* ASP.NET — технология разработки веб-сайтов, веб-приложений и веб-сервисов.
* Language Integrated Query (LINQ) — реализация языка запросов, напоминающего по синтаксису SQL в программах на .Net.
* Windows Presentation Foundation (WPF) — система создания графических интерфейсов, использующая язык разметки XAML. В отличие от Windows Forms использует графическую технологию DirectX, что обеспечивает более быструю работу за счет аппаратного ускорения графики.
* Windows Communication Foundation (WCF) — система обмена данными между приложениями .Net. Используется для создания распределённых приложений.

Одной из основных идей, заложенной в .Net, является совместимость различных частей приложения, которые могут быть разработаны на разных языках. Например программа, написанная на C# может обратиться к методу из библиотеки, написанной на Visual Basic .NET, или класс на Managed C++ может быть унаследован от класса на Delphi .Net. Языки, включённые в Visual Studio: C#, J#, Visual Basic .NET, JScript .NET, C++/CLI, F#.

Также существуют независимые проекты, позволяющие разрабатывать программы под .Net Framework на других языках [7].

На сегодняшний день XML является одним из наиболее распространенных стандартов документов, который позволяет в удобной форме сохранять сложные по структуре данные. Поэтому разработчики платформы .NET включили в фреймворк широкие возможности для работы с XML. XML — расширяемый язык разметки. Спецификация XML описывает XML-документы и частично описывает поведение XML-процессоров (программ, читающих XML-документы и обеспечивающих доступ к их содержимому). XML разрабатывался как язык с простым формальным синтаксисом, удобный для создания и обработки документов программами и одновременно удобный для чтения и создания документов человеком, с подчёркиванием нацеленности на использование в Интернете. Язык называется расширяемым, поскольку он не фиксирует разметку, используемую в документах: разработчик волен создать разметку в соответствии с потребностями к конкретной области, будучи ограниченным лишь синтаксическими правилами языка. Расширение XML — это конкретная грамматика, созданная на базе XML и представленная словарём тегов и их атрибутов, а также набором правил, определяющих какие атрибуты и элементы могут входить в состав других элементов. Сочетание простого формального синтаксиса, удобства для человека, расширяемости, а также базирование на кодировках Юникод для представления содержания документов привело к широкому использованию как собственно XML, так и множества производных специализированных языков на базе XML в самых разнообразных программных средствах.

Язык написания приложения – C#. C# – объекто-ориентированный язык программирования. C# относится к семье языков с C-подобным синтаксисом, из них его синтаксис наиболее близок к C++ и Java. Язык имеет статическую типизацию, поддерживает полиморфизм, перегрузку операторов (в том числе операторов явного и неявного приведения типа), делегаты, атрибуты, события, свойства, обобщённые типы и методы, итераторы, исключения, комментарии в формате XML. Основными преимуществами данного языка являются:

– компонентно-ориентированное программирование;

– безопасный (по сравнению с языками C и C++) код;

– унифицированная система типизации;

– поддержка событийно-ориентированного программирования;

– объединение лучших современных языков программирования: Java,C++, Visual Basic и др.

**4.2 Разработка структуры программного средства компьютерной системы**

При разработке было принято решение разбить приложение на следующие модули:

1. Модуль проведения эксперимента – модуль отвечает за проведение всех опытов в программе и вывод результатов на экран.
2. Модуль работы с файлами – осуществляет чтение, редактирование, удаление и создание файлов с базой слов, результатов и настроек.
3. Модуль администрирования – доступ и изменение настроек опыта, а также создание новых баз стимулов; просмотр и удаление результатов проведенных опытов.
4. Модуль регистрации – проводит идентификацию пользователя (студент/преподаватель), а также осуществляет проверку введенного пользователем пароля при входе в систему в качестве преподавателя.

Описание модулей продемонстрировано на рисунке 3.1.



Рисунок 3.1 – Описание модулей

Основной составляющей модулей являются методы. Метод — это блок кода, содержащий ряд инструкций. Программа инициирует выполнение инструкций, вызывая метод и указывая все аргументы, необходимые для этого метода. Далее будут описаны основные методы, содержащиеся в программных модулях (таблицы 3.1 – 3.4):

Таблица 3.1 – Описание методов модуля проведение эксперимента.

|  |  |
| --- | --- |
| Название метода | Описание метода |
| 1 | 2 |
| public void beginTestWords() | метод анализирует введенную информацию пользователем, и в зависимости от этого начинает выбранный тест. При выборе, первого опыта начинается эксперимент со словами, второго – опыт со слогами, третьего – с числовыми значениями, при выборе четвертого опыта, в качестве стимульного материала выступают буквы. Также данный метод предъявляет стимульный ряд испытуемому, в соответвии с выбранным экспериментом, и проводит первую часть опыта - осуществляет запись введенных испытуемых данных; |
| public void endTestWords() | метод, осуществляющий основную часть теста, а именно проводит вторую часть эксперимента по узнаванию стимулов из первой части опыта. По окончанию сохраняет результаты; |
| public void beginTestpic() | осуществляется начало эксперимента с графическими изображениями. Метод отвечает за тренировочную серию эксперимента, а также за проведение первой части опыта: происходит предъявление стимула и его воспроизведение. Формируются набор стимулов из случайных изображений, расположенных в директории. Затем следует отрисовка запомнившихся картинок в специальной форме; |
| public void endTestPic() | метод, описывающий заключительную часть эксперимента с изображениями. В нем пользователь пытается узнать стимулы, предъявленные в предыдущей части опыта; |
| public void res\_pic() | подведение итогов эксперимента с изображениями и получение результата. |

Таблица 3.2 – Описание методов модуля работы с файлами

|  |  |
| --- | --- |
| Название метода | Описание метода |
| 1 | 2 |
| public static Settings getSetting(string settingId) | метод считывает из XML файла данные и возвращает объект, содержащий настройки проведения определенного опыта; |
| public static void editSetting(Settings setting) | осуществляется изменение существующих настроек опыта; |
| public static int getWordCount() | считает количество стимулов в базе слов и возвращает это значение; |
| public static DBWords getWord(int wordId) | метод считывает из XML файла с базы слов по его Id и возвращает слово; |
| public static void saveResults(Result results) | создание структуры файла из переданного объекта результата и сохранение его в XML файл; |
| public static void saveResults(Result\_pic results) | перегрузка метода сохранения результатов в файл для эксперимента с изображениями; |
| public static List<Result> getResults() | считывание сохраненных результатов из XML файла и запись их в List; |
| public static List<Result\_pic> getResultsPic() | считывание сохраненных результатов для эксперимента с графическими стимулами из XML файла и запись их в List; |
| public static void deleteResult(Result result) | удаление результатов; |
| public static void deleteResult(Result\_pic result) | удаление результатов для эксперимента с графическими стимулами; |
| public static void saveDoc (Result res) | сохранение отчета о проведенных опытах в Word; |
| public static void createXML(string xmlName, string rootElement) | создание XML файла для сохранения результатов; |
| public static void createDataBase(List<string> DBword, string name) | создание новой базы стимулов при создании ее администратором. |

Таблица 3.3 – Описание методов модуля администрирования

|  |  |
| --- | --- |
| Название метода | Описание метода |
| 1 | 2 |
| private void button\_save\_admin(object sender, EventArgs e) | сохраняет все настройки, введенные администратором; |
| private void button\_deleteBase\_Click(object sender, EventArgs e) | удаление выбранной базы слов; |
| private void button25\_Click(object sender, EventArgs e) | открывает форму создания новой базы слов; |
| private void button\_save\_Click(object sender, EventArgs e) | в зависимости от выбора администратора сохраняет новую базу слов либо описание к опыту; |
| private void button\_showResults\_Click(object sender, EventArgs e) | открывает форму просмотра результатов выполненных испытуемыми опытов; |
| private void radioButton2\_CheckedChanged(object sender, EventArgs e) | идентификация пользователя: метод вызывается при входе в систему с учетной записи администратора и выполняет проверку пароля. |

Таблица 3.4 – Описание методов регистрации

|  |  |
| --- | --- |
| Название метода | Описание метода |
| 1 | 2 |
| private void buttonNext\_Click(object sender, EventArgs e) | метод осуществляющий контроль доступа к учетной записи администратора путем сверки введенного пользователем пароля, с паролем, прописанным администратором. |

Для осуществления перехода межу окнами формы были разработаны кнопки «Далее» и «Назад». Методами, описывающими данные кнопки приведены в таблице 3.5

Таблица 3.5 – Описание методов регистрации

|  |  |
| --- | --- |
| Название метода | Описание метода |
| 1 | 2 |
| private void buttonNext\_Click(object sender, EventArgs e) | метод для осуществления контроля доступа к элементам вкладок программы: переход на следующую вкладку; |
| private void buttonBack\_click(object sender, EventArgs e) | метод для осуществления контроля доступа к элементам вкладок программы: переход на предыдущую вкладку. |

Метод static void Main() – главная точка входа в программу и запуск основной формы приложения.

**3.3 Тестирование разработанного программно-аппаратного комплекса**

Тестирование программного обеспечения (Software Testing) – проверка соответствия между реальным и ожидаемым поведением программы, осуществляемая на конечном наборе тестов, выбранном определенным образом. В более широком смысле, **тестирование** – это одна из техник контроля качества, включающая в себя активности по планированию работ (Test Management), проектированию тестов (Test Design), выполнению тестирования (Test Execution) и анализу полученных результатов (Test Analysis). Основными целями проведения тестирования ПО являются демонстрация разработчикам и заказчикам, что программа соответствует требованиям и выявление ситуации, в которых поведение программы является неправильным, нежелательным или не соответствующим спецификации.

Все виды тестирования программного обеспечения, в зависимости от преследуемых целей, можно условно разделить на следующие группы:

1. Функциональные виды тестирования, которые базируются на функциях и особенностях, а также взаимодействии с другими системами, и могут быть представлены на всех уровнях тестирования. Функциональные виды тестирования рассматривают внешнее поведение системы. Самые распространенные виды функциональных тестов:

* функциональное тестирование (Functional testing);
* тестирование безопасности (Security and Access Control Testing);
* тестирование взаимодействия (Interoperability Testing).

1. Нефункциональные виды тестирования, описывающие тесты, необходимые для определения характеристик программного обеспечения, которые могут быть измерены различными величинами. В целом, это тестирование того, "Как" система работает. Основные виды нефункциональных тестов:

* все виды тестирования производительности;
* тестирование установки (Installation testing);
* тестирование удобства пользования (Usability Testing);
* тестирование на отказ и восстановление (Failover Testing);
* конфигурационное тестирование (Configuration Testing).

1. Связанные с изменениями виды тестирования, работающие после проведения необходимых изменений, таких как исправление бага, когда программное обеспечение должно быть перетестировано для подтверждения, что проблема была действительно решена. Ниже перечислены виды тестирования, которые необходимо проводить после установки программного обеспечения, для подтверждения работоспособности приложения или правильности осуществленного исправления дефекта:

* дымовое тестирование (Smoke Testing);
* регрессионное тестирование (Regression Testing);
* тестирование сборки (Build Verification Test);
* санитарное тестирование или проверка согласованности/исправности (Sanity Testing).

По степени изолированности компонентов можно выделить следующие виды тестирования:

1. Модульное тестирование – процесс в программировании, позволяющий проверить на корректность отдельные модули исходного кода программы. Идея состоит в том, чтобы писать тесты для каждой функции или метода. Это позволяет проверить, не привело ли очередное изменение кода к регрессии, а также облегчает обнаружение и устранение таких ошибок.
2. Интеграционное тестирование – одна из фаз тестирования программного обеспечения, при которой отдельные программные модули объединяются и тестируются в группе. Целью интеграционного тестирования является проверка соответствия проектируемых единиц функциональным, приёмным и требованиям надежности.
3. Системное тестирование программного обеспечения – это тестирование программного обеспечения, выполняемое на полной, интегрированной системе, с целью проверки соответствия системы исходным требованиям. Основной задачей системного тестирования является проверка как функциональных, так и не функциональных требований к системе в целом[9].

Тестирование программного обеспечения в рамках курсового проекта проводится в виде функционального ручного тестирования, так как оно имитирует фактическое использование системы и позволяет сразу же обнаружить дефекты в работе приложения. Также создание Test case обеспечивает возможность более детального рассмотрения работы основного функционала приложения. В таблице 3 приведены примеры составления Test case.

Таблица 3.6 – Пример Test case для разработанного приложения

|  |  |  |  |
| --- | --- | --- | --- |
| № | Функция | Функциональность | Тестовое покрытие |
| 1 | Инсталлятор | Возможность установки приложения на ПК. | да |
| 2 | Запуск приложения после установки | да |
| Интерфейс пользователя | | | |

[6] Microsoft Visual Studio [Электронный ресурс]. – Режим доступа <https://ru.wikipedia.org/wiki/Microsoft_Visual_Studio>  
[7] Microsoft .Net Framework [Электронный ресурс]. – Режим доступа <http://net-framework.ru/article/chto-takoe>  
[9] [9] Документация «Виды тестирования ПО» [Электронный ресурс]. – Электронные данные. – Режим доступа: http://www.protesting.ru/testing/testtypes.html