Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

филиал «Минский радиотехнический колледж»

**ПРОГРАММНОЕ СРЕДСТВО «АВТОМАТИЧЕСКОЕ РАБОЧЕЕ МЕСТО БИБЛИОТЕКАРЯ»**

Пояснительная записка

к курсовому проекту по учебной дисциплине

«Базы данных и системы управления базами данных»

КП62.493007.207.081ПЗ

Руководитель /О.Н. Виничук/

Учащийся /М.А. Дрозд/

Минск 2020

КП 62.493025.102.081ПЗ

Программное средство «АРМ Библиотекаря»

Пояснительная записка

МРК

Содержание

[Введение 3](#_Toc36157286)

[1 Постановка задачи 5](#_Toc36157287)

[1.1 Описание предметной области 5](#_Toc36157288)

[1.2 Анализ существующих аналогов 6](#_Toc36157289)

[1.3 Функциональное назначение 8](#_Toc36157290)

[2 Проектирование реляционной базы данных 9](#_Toc36157291)

[2.1 Описание инструментов разработки 9](#_Toc36157292)

[2.2 Концептуальное проектирование 11](#_Toc36157293)

[2.3 Логическое проектирование 13](#_Toc36157294)

[2.4 Физическое проектирование 15](#_Toc36157295)

[2.5 Проектирование диаграммы деятельности 19](#_Toc36157296)

[3 Программная реализация 20](#_Toc36157297)

[3.1 Описание разработанных компонентов базы данных 20](#_Toc36157298)

[3.2 Описание разработанных модулей приложения 21](#_Toc36157299)

[4 Тестирование 25](#_Toc36157300)

[5 Руководство пользователя 30](#_Toc36157301)

[5.1 Описание процесса установки и запуска приложения 30](#_Toc36157302)

[5.2 Демонстрационный пример работы приложения 30](#_Toc36157303)

[Заключение 34](#_Toc36157304)

[Список использованных источников 35](#_Toc36157305)

[Приложение А 36](#_Toc36157306)

Дата

Подпись

Лист

Изм.

Разраб.

№ докум.

Дрозд

Листов

Провер.

Н. Контр.

Реценз.

Виничук

Лист

44

2

Лит.

Утверд.

# Введение

Темой курсового проекта является программное средство «АРМ библиотекаря», которое будет помогать библиотекарям вести учет книг.

Главная цель, которая стоит перед написанием курсового проекта – это приобретение практических навыков проектирования базы данных с использованием программ системами управления базами данных. В ходе выполнения данной работы необходимо будет разработать информационную базу данных для автоматического рабочего места библиотекаря.

Поставленные следующие задачи для курсового проектирования:

* описание алгоритмов, работы программы. Кроме того, подробное описание логической модели и физической структуры программы, в частности, реализация классов и основных методов. Так же указание компонентов windows forms, которые были использованы в разработке;
* проведение тестирования программы, которое будет показывать, что программа работает корректно и готова к использованию;
* в пункте применение будет содержатся информация о функциональном назначении программы, о требованиях к установке и аппаратно-программным ресурсам. Также будет приведен пример работы с программой;
* разработка диаграммы деятельности и сущность связи.

Развитие современного общества вплотную связано с развитием вычислительной техники и широкой информатизацией всех сфер человеческой деятельности. Трудно представить нашему современнику, что буквально совсем недавно человек проводил всякое вычисление не при помощи машины, а вручную. Сейчас же практически в любой сфере жизни человека присутствуют настольные персональные компьютеры (ПК), ноутбуки, планшеты, смартфоны, которые заняли достойное место в жизни людей. Они улучшают и облегчают их деятельность.

С развитием компьютерных технологий стали появляться все новые и новые варианты их использования. Будет странным увидеть человека, который использует компьютер только для вычислений, ведь современные технологии позволяют использовать компьютер в абсолютно различных направлениях

Развитие компьютерной техники привело к её внедрению практически во все области жизнедеятельности человека, одной из которых является библиотекарское дело.

Для данного курсового проекта будет разработано программное средство «АРМ библиотекаря», которое будет облегчать труд библиотекарей и уменьшать количество бумажной работы.

Однако создание приложений требует особых знаний, умений и, конечно же, практики. Для того чтобы успешно практиковаться в написании программ, как и во всем, следует ставить перед собой цели и задачи. Они должны иметь четкую формулировку. Были поставлены следующие цели, которые должны быть достигнуты за период курсового проектирования:

* закрепить знания, полученные в процессе изучения дисциплины «Системы управления базами данных»;
* ознакомиться с современными средствами автоматизации процесса проектирования и разработки ПО;
* закрепить изученные методы и приемы в области инженерного проектирования и разработки ПО.

В приложениях требуется привести текст программы, диаграмму вариантов использования и диаграмму классов.

В завершении курсового проектирования все поставленные цели были достигнуты.

# 1 Постановка задачи

# Описание предметной области

Библиотека – это учреждение культуры, организующее сбор, хранение и общественное пользование произведениями печати и другими документами. Библиотеки систематически занимаются сбором, хранением и выдачей читателям произведений печати.

Основными направлениями работы любой библиотеки являются: комплектование и организация книжного фонда; обслуживание читателей.

Комплектование фондов библиотеки состоит из систематического выявления (путем просмотра библиографических источников и литературы) нужных для данной библиотеки изданий и приобретения их. От своевременности и полноты комплектования библиотеки в значительной мере зависит уровень обслуживания читателей.

Организация книжного фонда включает вопросы учета, расстановки, хранения литературы и доставки ее читателю.

Учет библиотечного фонда – это прием поступивших в библиотеку изданий, запись прибывших и исключение по тем или иным причинам непригодных для читателей изданий в документах библиотеки, которые дают точные сведения о наличном составе фонда и помогают обеспечить его сохранность.

Правильная организация фонда облегчает читателю пользование литературой, библиотекарю - быстрое выполнение читательских требований, а также обеспечивает сохранность фондов как общественной собственности.

В библиотеке ведется картотека читателей. О каждом читателе заносятся следующие сведения:

* фамилия;
* имя;
* отчество;
* дата рождения;
* адрес;
* мобильный телефон.

Обслуживание читателей библиотеки осуществляется различным путем выдачи литературы и помощи отдельным читателям в подборе необходимой им литературы.

Срок пользования литературой для различных категорий читателей и количество выдаваемых изданий на каждом абонементе определяется администрацией, исходя из вида литературы и категории читателя. Число выдаваемых книг не ограничивается.

Библиотечная деятельность связана с учетом большого количества операций, множество книг и читателей серьезно замедляют работу библиотекарей. Сложность поиска нужной книги в каталоге занимает длительное время, и целиком опирается на компетентность работников библиотеки.

Каждая книга может присутствовать в нескольких экземплярах.

Все книги различаются по своему уникальному номеру.

БД находится на специальном компьютере в электронном виде, постоянно редактируется и изменяется работниками библиотеки. С базой данных библиотеки могут работать как руководители (администрация) библиотеки, так и работники библиотеки, заинтересованные в получении необходимой информации. При работе с системой библиотекарь должен иметь возможность решать следующие задачи:

* Принимать новые книги и регистрировать их в библиотеке.
* Вести учет выданных книг читателям, при этом предполагается два режима работы: выдача книг читателю и прием от него возвращаемых им книг обратно в библиотеку. При выдаче книг фиксируется, когда и какая книга была выдана данному читателю и на какой срок выдается данная книга. При приеме книги, возвращаемой читателем, проверяется соответствие названия книги, и она ставится на свое старое место в библиотеке.
* Администрация библиотеки должна иметь возможность получать сведения о должниках - читателях библиотеки, которые не вернули вовремя взятые книги.

Таким образом

# Анализ существующих аналогов

На данный момент существует несколько аналогов, находящихся в свободном доступе.

Программа 1С:Библиотека обладает очень широким функционалом: ведение учета книг, ведение учета читателей и каталогизация, однако для того, чтобы научится этим всем пользоваться, необходимо проходимо долгое и недешевое обучение у компании, выпускающей данный продукт. Помимо этого, заказчика не устраивает то, что программа использует внутреннюю базу данных, а заказчик хочет использовать уже существующую базу данных MySQL. Рабочее окно программы 1С:Библиотека представлено на рисунке 1.1

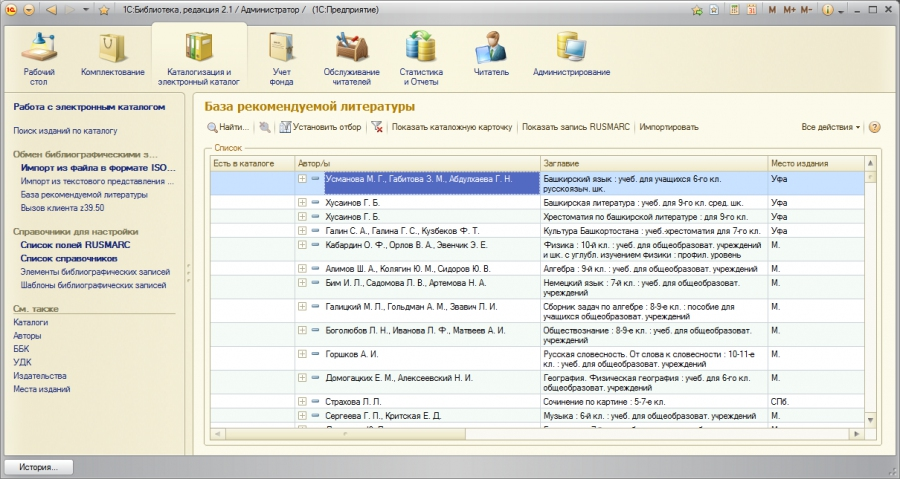


Рисунок 1.1 - Рабочее окно программы 1С:Библиотека

Программа OPAC-Global, рабочее окно которой представлено на рисунке 1.2.

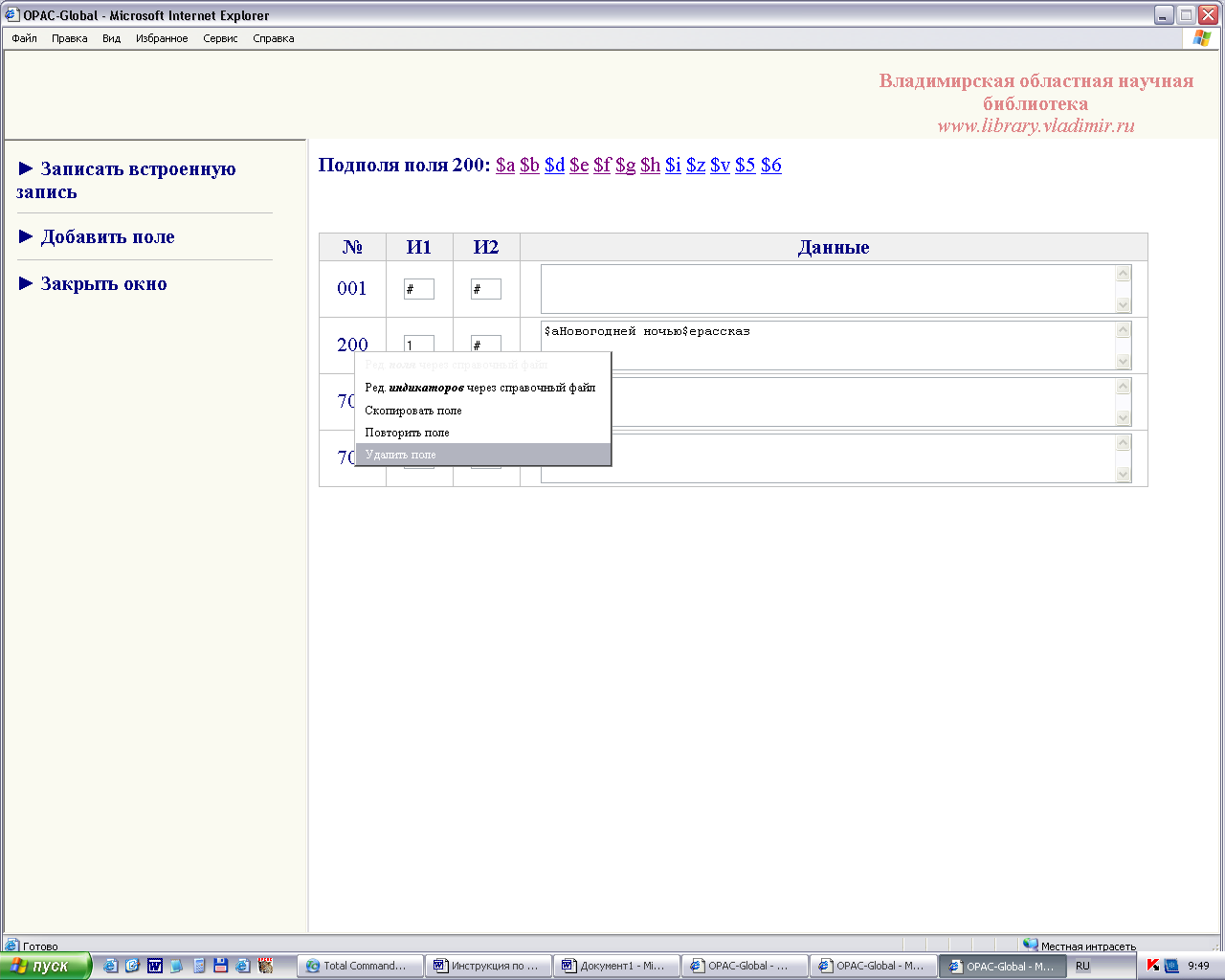


Рисунок 1.2 - Рабочее окно программы OPAC-Global

Программа OPAC-Global имеет схожий функционал программы 1С:Библиотека однако из-за более простого интерфейса научиться пользоваться ей становится намного легче. Но даже эта программа не умеет подключаться к уже существующим базам данных MySQL, что делает эту программу плохим вариантом для заказчика.

Таким образом на основе приведенных аналогов, можно сделать вывод о том, какими преимуществами аналогов должно обладать разрабатываемое средство и какие недостатки должны в нем отсутствовать. К достоинствам программного средства будут относится возможность работы с уже существующей базой данных MySQL и простой интерфейс.

# Функциональное назначение

АРМ библиотекаря реализовывает следующие функции:

* Добавление, редактирование и удаление читателей.
* Добавление, редактирование и удаление книг.
* Возможность выдачи книги читателю.
* Поиск по читателям по различным критериям.
* Поиск по книгам по различным критериям.
* Возможность продлить дату возврата книги читателя.
* Возможность вернуть книгу читателю.
* Контроль сроков возврата книги путем сравнения текущей даты и даты предположительной сдачи книги.

На этапе общесистемного проектирования одной из главных задач является задача проектирования структуры. Применительно к человеко-машинным системам, наиболее распространённым является случай, когда принципы построения системы, выполняемые функции и компоненты системы оказываются заданными.

Эффективным инструментом проектирования и рационализации организационных структур управления является моделирование, позволяющее находить оптимальные варианты их построения, прогнозировать их развитие, проводить диагностику состояния действующей структуры и устанавливать ее соответствие реальным производственно-технологическим условиям, оценивать различные варианты построения организационной структуры, когда прямые эксперименты невозможны или затруднительны, а также экономически невыгодны, а иногда невозможны.

# 2 Проектирование реляционной базы данных

# 2.1 Описание инструментов 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 программного средства с требованиями пользователя.

В качестве средства проектирования базы данных, была выбрана СУБД MySQL.

MySQL **–** система управления реляционными базами данных (СУБД), разработанная корпорацией Microsoft. Основной используемый язык запросов **–** Transact-SQL, создан совместно Microsoft и Sybase. Transact-SQL является реализацией стандарта ANSI/ISO по структурированному языку запросов (SQL) с расширениями. Используется для работы с базами данных размером от персональных до крупных баз данных масштаба предприятия; конкурирует с другими СУБД в этом сегменте рынка. Эта разработка корпорации Microsoft, является популярной среди малых и средних предприятий, используется во многих крупных компаниях.

В отличие от всех остальных наиболее известных СУБД, MySQL имеет ряд преимуществ. Главным из них является то, что он обеспечивает создание информационных систем с архитектурой «клиент-сервер», в которой он играет роль сервера баз данных. Одним из важных свойств MySQL является возможность управления целостностью данных.

Также можно сказать, что он удовлетворяет требованиям, предъявляемым к системам распределенной обработки информации. Эта СУБД поддерживает: тиражирование данных, параллельную обработку, создание и обработку больших баз данных на недорогих аппаратных платформах, отличается простотой управления и использования.

Средством, связывающим клиента с сервером, является язык SQL (Structured Query Language) – язык структурированных запросов. В данном курсовом проекте были использованы основные операторы этого языка, в частности для создания запросов. Этот язык является очень распространенным на данный момент и имеет ряд особенностей.

Масштабируемость и надежность. MySQL обеспечивает практически неограниченный рост объемов хранения данных за счет увеличения надежности и масштабируемости системы, используя все преимущества мультипроцессорной обработки данных. Это безопасная, надежная, масштабируемая платформа, защищающая информацию в приложениях и повышающая её доступность.

Скорость создания решений. MySQL в сочетании с.NET Framework уменьшает время разработки, внедрения и выхода на рынок современных приложений, ускоряет процесс поиска данных, упрощает управление, позволяет использовать создаваемые пользователем функции в других приложениях, предоставляет широкие возможности для создания Web-приложений. Среда ADO.NET Entity Framework повышает эффективность труда разработчиков, поскольку теперь они имеют дело не непосредственно с таблицами и полями, а с логическими информационными сущностями, согласованными с бизнес-требованиями устройства, а позже синхронизовать их с центральными серверами.

Удобство разработки приложений. Интеграция с Microsoft Visual Studio и.NET Framework дает возможность создавать хранимые процедуры на языках, отличных от T-SQL.

Ядро реляционной базы данных MySQL включает следующие возможности для создания и поддержки различных приложений с хранилищами данных:

– табличные секции, обеспечивающие быструю загрузку данных и упрощенную поддержку очень больших таблиц;

– выполнение онлайновых операций над индексами;

– быстрая инициализация файлов.

На основании этих преимуществ, по сравнению с другими СУБД, и возможностей СУБД была выбрана эта система управления реляционными базами данных.

Для разработки пользовательского интерфейса был выбран язык С# и среда разработки Visual Studio. Это было сделано потому, что на данный момент они набирают очень большой темп, и нет столь простого и многофункционального языка, как C#.

Visual Studio и .NET являются большим шагом вперед и предоставляют разработчикам новые инструменты, которые повышают их производительность и дают возможность создавать различные приложения для множества платформ. За последние несколько лет продуктовая линейка Visual Studio была значительно расширена множеством новых возможностей.

Самым главным компонентом является Visual Studio IDE которое используют миллионы разработчиков во всем мире и создают с помощью этого инструмента замечательные приложения. Сегодняшний релиз Visual Studio и .NET предоставляет сотни новых возможностей для разработчиков и позволяет создавать десктоп, веб, мобильные, облачные приложения и многое другое [1].

Visual Studio.NET предоставляет мощные и удобные средства написания, корректировки, компиляции, отладки и запуска приложений, использующих .NET-совместимые языки. Корпорация Microsoft включила в платформу средства разработки для четырех языков: С#, VB.NET, С++ и J#.

Язык написания программной части 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, строгой компонентной архитектуре и реализации передовых механизмов обеспечения безопасности кода.

Таким образом разрабатывается MDI – приложение на языке программирования C# в Microsoft Visual Studio с использованием интерфейса Windows Forms и базы данных My SQL.

# 2.2 Концептуальное проектирование

Основными задачами концептуального проектирования являются определение предметной области системы и формирование взгляда на предметную область с позиций сообщества будущих пользователей БД, т. е. инфологической модели предметной области.

Концептуальная модель предметной области представляет собой описание структуры и динамики предметной области, характера информационных потребностей пользователей в терминах, понятных пользователю и не зависимых от реализации БД. Это описание выражается в терминах не отдельных объектах предметной области и связей между ними, а их типов, связанных с ними ограничений целостности и тех процессов, которые приводят к переходу предметной области из одного состояния в другое.

Сущности описывают объекты, являющиеся предметом деятельности предметной области, и субъекты, осуществляющие деятельность в рамках предметной области. Свойства объектов и субъектов реального мира описываются с помощью атрибутов.

Основные требования к содержанию модели:

* концептуальная модель должна отображать все сущности и связи, значимые для той цели, ради которой мы ее рисуем;
* все объекты модели (и сущности, и связи) должны быть именованы. Именование сущностей и связей должно выполняться в терминах предметной области;
* для связей должна быть указана кратность (один — многие);
* для каждой связи должно быть указано направление чтения.

Для данного программного средства существует три главных сущности: читатель, книга и задолженность. Сущность «Читатель» содержит следующие атрибуты: фамилия, имя, отчество, адрес, номер телефона. Сущность «Книга» содержит следующие атрибуты: автор, название, язык, жанр, обложка, издательство. Сущность «Задолженность» содержит атрибуты: читатель, книга, дата выдачи книги и дата возврата книги. Диаграмма сущность связь представлена на рисунке 3.

Взаимоотношения между сущностями иллюстрируются с помощью связей. Правила и ограничения взаимоотношений описываются с помощью свойств связей. Обычно связи определяют либо зависимости между сущностями, либо влияние одной сущности на другую.

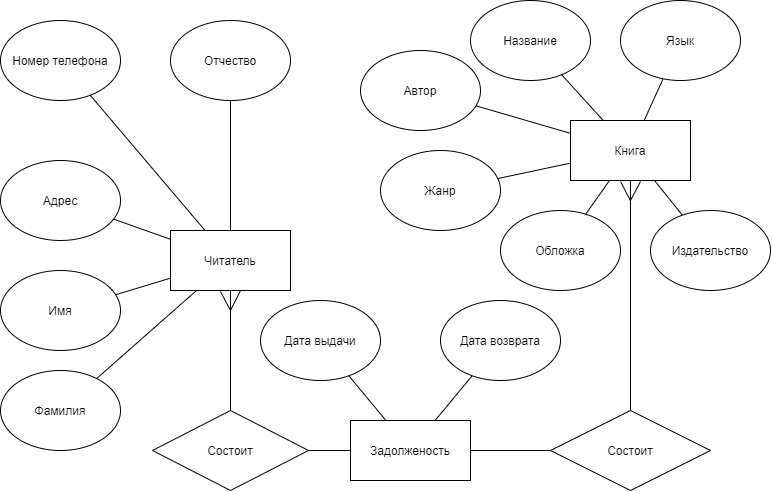


Рисунок 3 – Диаграмма «Сущность-связь»

Типы сущностей можно классифицировать как сильные и слабые. Сильные сущности существуют сами по себе, а существование слабых сущностей зависит от существования сильных. Слабые сущности называют подчинёнными (дочерними), а сильные – базовыми (родительскими).

# 2.3 Логическое проектирование

На этапе логического проектирования разрабатывается логическая структура БД, соответствующая логической модели предметной области. Решение этой задачи существенно зависит от модели данных, поддерживаемой выбранной СУБД. Будем рассматривать логическое проектирование БД для реляционной модели данных, так как современные СУБД – реляционные.

Проектирование реляционной базы данных проходит в том же порядке, что и проектирование БД других моделей данных, но имеет свои особенности.

Проектирование схемы БД должно решать задачи минимизации дублирования данных и упрощения процедур их обработки и обновления. При неправильно спроектированной схеме БД могут возникнуть аномалии модификации данных. Они обусловлены отсутствием средств явного представления типов множественных связей между объектами ПО и неразвитостью средств описания ограничений целостности на уровне модели данных.

* На этом этапе выполняются следующие действия:
* удаление связей M:N;
* удаление рекурсивных связей;
* удаление связей с атрибутами;
* удаление множественных атрибутов;
* перепроверка связей типа 1:1;
* удаление избыточных связей.

Далее выполняется нормализация отношений. В рамках реляционной модели данных Э. Ф. Коддом (E. F. Codd) был разработан аппарат нормализации отношений и предложен механизм, позволяющий любое отношение преобразовать к третьей нормальной форме.

Нормализация отношений выполняется на основе анализа первичных ключей и существования функциональных зависимостей между атрибутами. Как правило, нормализация выполняется в несколько этапов. Каждый этап соответствует определенной нормальной форме (НФ). При проектировании реляционных баз данных требование первой нормальной формы (1НФ) должны выполняться всегда, остальные по желанию проектировщика. Однако, чтобы исключить аномалии обновления и избыточность данных рекомендуется приводить отношение к третьей нормальной форме 3НФ [2].

Данная схема БД состоит из 9 таблиц:

* Таблица «reader»;
* Таблица «record»;
* Таблица «book»;
* Таблица «author»;
* Таблица «language»;
* Таблица «genre»;
* Таблица «publishing\_house»;
* Таблица «city»;
* Таблица «cover».

Таблица «Student» должна хранить информацию о фамилии, имени, отчестве студента, а также год его поступления и средний балл. Таблица «Student» связана с таблицей «Group» с помощью следующих атрибутов: таблица «Group» атрибут «groupID» (первичный ключ) с таблицей «Student» атрибут «groupID» (внешний ключ).

Таблицы «Group» и «Department» хранят данные о группе и отделении, к которому принадлежит студент и заведующего соответствующего отделения. Таблица «Group» связана с таблицей «Department» с помощью следующих атрибутов: таблица «Department» атрибут «depID» (первичный ключ) с таблицей «Group» атрибут «depID» (внешний ключ).

Так же в проектируемой базе данных присутствует таблица «TestingSession», которая хранит информацию о дате прохождения теста и его результате. Эта таблица связана с таблицами «Student», «SessionStatus», «Session\_question» с помощью атрибутов: «student» (внешний ключ), «executingState» (внешний ключ).

Таблица «SessionStatus» хранит информацию о статусе прохождения теста, например, пройден был тест данным студентом или нет. Таблица «Question» хранит данные о вопросе.

Таблица «AnswerVar» хранит данные о доступных вариантах ответа для определенного вопроса. Таблицы «SessionQuestion» и «QuestionAnswer» являются связующими сущностями для вопросов и ответов теста c помощью внешних ключей «questionID», «sessionID» и «AnswerID».

Схема базы данных представлена в плакате КП 62.493025.102.001ПЛ.

# 2.4 Физическое проектирование

Этап физического проектирования заключается в увязке логической структуры БД и физической среды хранения с целью наиболее эффективного размещения данных, т. е. отображении логической структуры БД в структуру хранения. Решается вопрос размещения хранимых данных в пространстве памяти, выбора эффективных методов доступа к различным компонентам «физической» БД. Результаты этого этапа документируются в форме схемы хранения на языке определения данных (DDL). Принятые на этом этапе решения оказывают определяющее влияние на производительность системы. Между физическим и логическим проектированием существует обратная связь, так как иногда с целью повышения эффективности приходится менять структуру БД. Это возникает, если полная нормализация данных не позволяет достичь требуемой эффективности обработки информации. Принимается решение о денормализации отношений. При этом следует учитывать следующее:

* денормализация усложняет реализацию БД;
* денормализация снижает гибкость системы;
* денормализация ускоряет выборку данных, но снижается скорость обновления данных.

Иногда денормализацию называют оптимизацией исполнения.

Одной из важнейших составляющих проекта базы данных является разработка средств защиты БД. Защита данных имеет два аспекта: защита от сбоев и защита от несанкционированного доступа. Для защиты от сбоев разрабатывается стратегия резервного копирования. Для защиты от несанкционированного доступа каждому пользователю доступ к данным предоставляется только в соответствии с его правами доступа [4].

Итогом физического проектирования должно быть представление метаданные всех таблиц. Создание базы данных начинается с самых нижних уровней, с таблиц у которых нету внешних ключей. Далее будет описано пошаговое создание базы данных.

Метаданные таблицы «reader» представлены ниже:

CREATE TABLE `reader` (

`id` int AUTO\_INCREMENT NOT NULL,

`name` varchar(255) NOT NULL,

`surname` varchar(255) NOT NULL,

`patronymic` varchar(255) NOT NULL,

`birthday` DATETIME NOT NULL,

`phone\_number` varchar(255) NOT NULL,

`address` varchar(255) NOT NULL,

PRIMARY KEY (`id`)

);

Метаданные таблицы «record» представлены ниже:

CREATE TABLE `record` (

`id` int AUTO\_INCREMENT NOT NULL,

`id\_reader` int NOT NULL,

`id\_book` int NOT NULL,

`issue\_date` DATETIME NOT NULL,

`return\_date` DATETIME NOT NULL,

PRIMARY KEY (`id`)

);

Метаданные таблицы «book» представлены ниже:

CREATE TABLE `book` (

`id` int AUTO\_INCREMENT NOT NULL,

`id\_author` int NOT NULL,

`id\_language` int NOT NULL,

`id\_genre` int NOT NULL,

`id\_publishing\_house` int NOT NULL,

`id\_cover` int NOT NULL,

`title` varchar(255) NOT NULL,

PRIMARY KEY (`id`)

);

Метаданные таблицы «author» представлены ниже:

CREATE TABLE `author` (

`id` int AUTO\_INCREMENT NOT NULL,

`name` varchar(255) NOT NULL,

`surname` varchar(255) NOT NULL,

`patronymic` varchar(255) NOT NULL,

PRIMARY KEY (`id`)

);

Метаданные таблицы «genre» представлены ниже:

CREATE TABLE `genre` (

`id` int AUTO\_INCREMENT NOT NULL,

`genre` varchar(255) NOT NULL,

PRIMARY KEY (`id`)

);

Метаданные таблицы «language» представлены ниже:

CREATE TABLE `language` (

`id` int AUTO\_INCREMENT NOT NULL,

`language` varchar(255) NOT NULL,

PRIMARY KEY (`id`)

);

Метаданные таблицы «publishing\_house» представлены ниже

CREATE TABLE `publishing\_house` (

`id` int AUTO\_INCREMENT NOT NULL,

`id\_city` int NOT NULL,

`title` varchar(255) NOT NULL,

PRIMARY KEY (`id`)

);

Метаданные таблицы «city» представлены ниже:

CREATE TABLE `city` (

`id` int AUTO\_INCREMENT NOT NULL,

`city\_name` varchar(255) NOT NULL,

PRIMARY KEY (`id`)

);

Метаданные таблицы «cover» представлены ниже:

CREATE TABLE `cover` (

`id` int AUTO\_INCREMENT NOT NULL,

`cover\_description` varchar(255) NOT NULL,

PRIMARY KEY (`id`)

);

Представленная база данных является нормализованной.

# 2.5 Проектирование диаграммы деятельности

Проектирование диаграммы деятельности является важным этапом при разработке приложения. Диаграмма деятельности (Activity Diagrams) является представлением алгоритмов неких действий (активностей), выполняющихся в системе.

После запуска приложения отобразиться окно с информацией и инструкцией о тесте. После перехода откроется новое окно, где будут находится два поля для ввода ФИО и номера группы пользователя. После ввода верной информации можно начать тест, который отобразиться в новом окне с доступными вариантами ответа. Для перехода к следующему вопросу необходимо выбрать вариант ответа. Если ответ не был выбран, то новое окно с новым тестом открыто не будет. Если ответ был выбран, то после выполнится переход к следующему окну с тестом. После того, как все тесты будут пройдены появиться окно с вычисленным результатом, который автоматически будет сохранен. Разработанная диаграмма деятельности представлена в плакате КП 62.493025.102.002ПЛ.

Диаграммы деятельности могут быть использованы не только для спецификации алгоритмов вычислений или потоков управления в программных системах. Не менее важная область их применения связана с моделированием бизнес-процессов. Действительно, деятельность любой организации также представляет собой не что иное, как совокупность отдельных действий, направленных на достижение требуемого результата.

# 3 Программная реализация

# 3.1 Описание разработанных компонентов базы данных

В ходе курсового проекта в СУБД MySQL были созданы следующие таблицы:

* таблица «Student» хранит личные данных о студенте. А именно его ФИО, средний балл;
* таблица «Group» хранит данные о номере группы каждого студента;
* таблица «Departament» хранит данные о названии отделения, кто им заведует, и какой студент принадлежит к какому отделению.
* таблица «TestingSession» хранит данные о статусе прохождения теста;
* таблица «Questions» хранит данные о количестве и тексте вопроса;
* таблица «AnswerVar» хранит данные о данных ответах пользователя;
* таблица «SessionStatus» и таблица «Session\_question» являются промежуточными таблицами для связи таблиц с ответами и вопросами;
* таблица «Question\_answer» хранит все варианты ответов для каждого вопроса.

Так же были разработаны процедура «InsertAnswerVar» и триггер «OnAnswerVarDelete».

Процедура: InsertAnswerVar(ansText varchar(45), answerPoints int, questionID int) - добавляет вариант ответа с текстом ansText и количество очков за него answerPoints к вопросу с id questionID, добавляя данные в таблицу вариантов ответа и таблицу, связывающую эти варианты ответа с вопросами.

delimiter &&

create procedure InsertAnswerVar(ansText varchar(45), answerPoints int, questionID int)

begin

declare lastAnswerVarID int default -1;

set lastAnswerVarID = (select max(number) from answervar);

start transaction;

if ((select questions.questionID from questions where questions.questionID = questionID) is not null)

then

insert into answervar values (lastAnswerVarID + 1, ansText, answerPoints);

insert into question\_answer values (questionID, lastAnswerVarID + 1);

end if;

commit;

end&& delimiter ;

Триггер: OnAnswerVarDelete - вызывается при попытке удаления вариантов ответа из таблицы, перед этим удаляя связи с вопросами

delimiter &&

create trigger OnAnswerVarDelete before delete on answervar

for each row

begin

if ((select count(questionID) from question\_answer where answerID = old.number) > 0)

then

delete from question\_answer where answerID = old.number;

end if;

end&& delimiter ;

# 3.2 Описание разработанных модулей приложения

При проектировании уделяется отдельное внимание вопросам предоставления информации пользователю, а также взаимодействию пользователя с системой, что в наибольшей степени определяется типом приложения и выдается заказчиком в виде требования. В настоящее время существует большое количество различных программных решений, которые позволяют разработчику выбрать наиболее оптимальное для конкретной проблемы.

Модули – это законченные программные блоки, состоящие из моделей, представлений, контроллеров и других вспомогательных компонентов. При установке модулей в приложение, конечный пользователь получает доступ к их контроллерам. По этой причине модули часто рассматриваются как миниатюрные приложения. В отличии от приложений, модули нельзя развертывать отдельно. Модули должны находиться внутри приложений.

Разрабатываемое программное средство будет иметь вид классического desktop-приложения. Такие приложения также называют «оконными». В них используется графический интерфейс и ввод информации осуществляется при помощи клавиатуры и мыши. Данные приложения позволяют создавать практически любые по сложности решения. При разработке был использован один из принципов SOLID (Single Responsibility, Open Closed, Liskov substitution, Interface Segregation and Dependency Inversion Principles). Принцип единственной ответственности означает, что каждый класс работает только над одной целью, ответственен только за неё и изменяется только по одной причине. В программном средстве реализовано три основных класса:

* Main.cs – класс формы главного меню. Часть кода главной формы представлена на рисунке 4. На главной форме происходит регистрация пользователя. На главной форме находится два поля для ввода личной информации пользователя: «Фамилия Имя Отчество» и «Группа», где пользователь должен ввести эти данные для последующего анализа его результата. Объекты этой формы содержат в себе две кнопки: «Начать тест», «Выйти»;

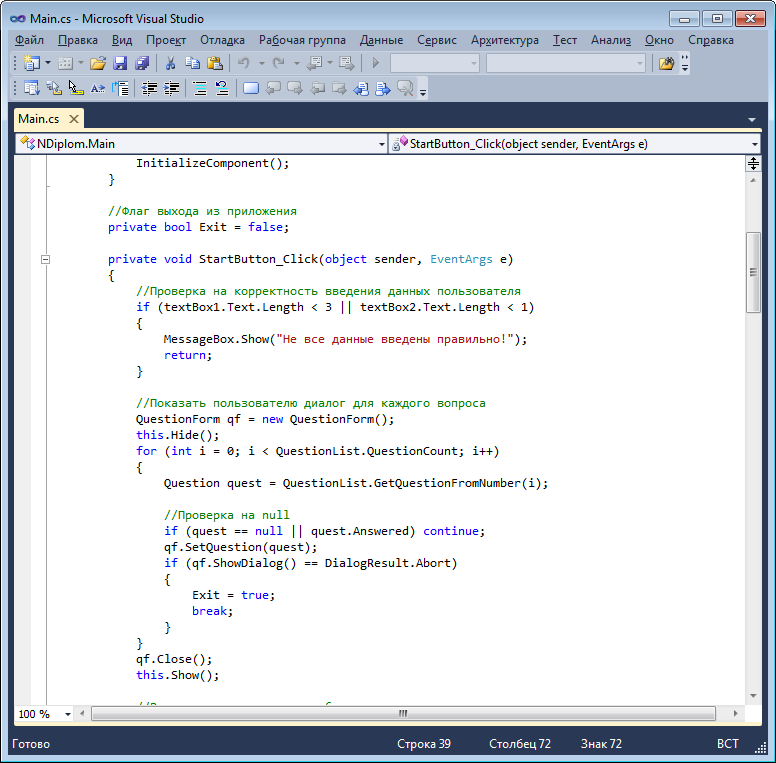


Рисунок 4 – Код главной формы

* Question Form.cs – класс формы «Вопрос №». Часть кода формы вопросов представлена на рисунке 5. Объекты которого представляют собой форму с отображением вопроса на который есть 3-4 варианта ответов и содержащей в себе две кнопки для перехода на следующий вопрос или предварительного выхода из формы прохождения тестирования;

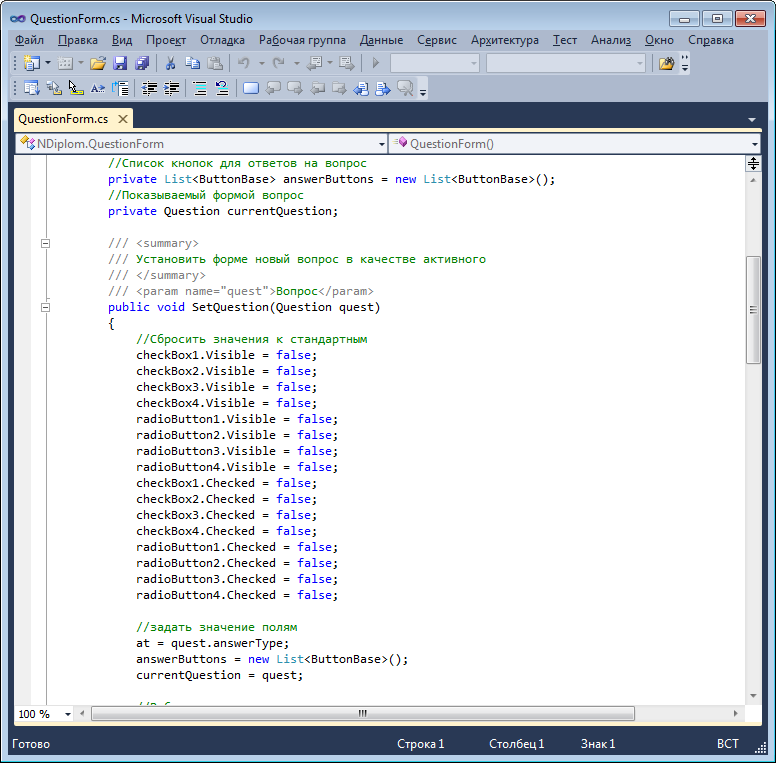


Рисунок 5 – Код формы с вопросами

* ResultShowingForm.cs – это класс, в котором реализовано отображение результата психоаналитики личности пользователя. Часть кода формы с результатом представлена на рисунке 6. Результат показывается постепенно, переходя от одного фактора к последующему. Для перехода к следующему фактору внизу диалогового окна расположена кнопка «Дальше».

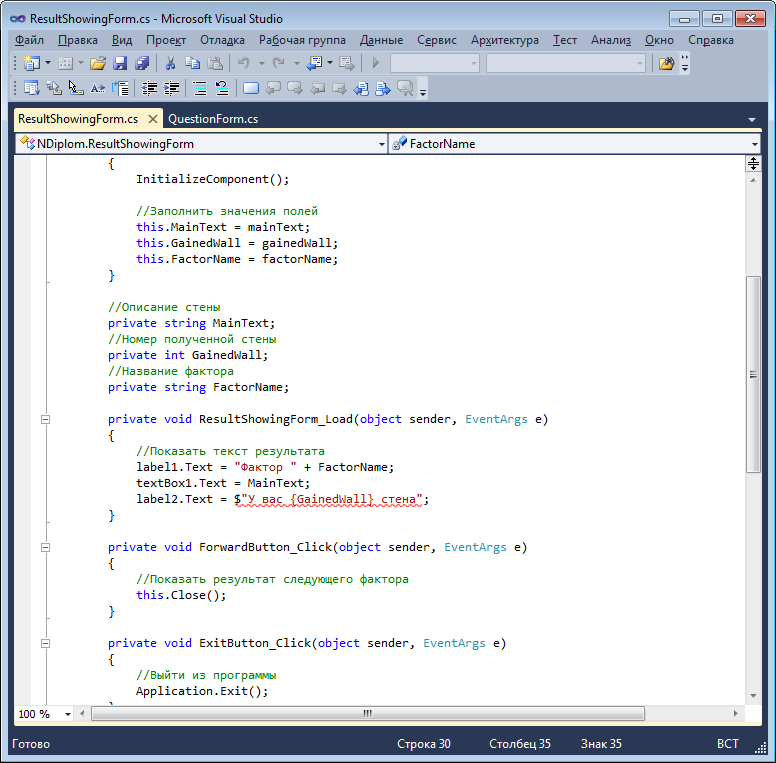


Рисунок 6 – Код формы с результатом

В программном средстве были использованы следующие компоненты:

* button – при нажатии происходит программно связанное с этим нажатием действие либо событие;
* label – отображает текст на форме окна;
* рanel – элемент управления, который позволяет перемещаться по веб-странице внутри формы;
* textBox – элемент управления, который позволяет перемещаться по веб-странице внутри формы.

Процесс разработки форм состоит в том, чтобы разместить компоненты на форме или других компонентах. Другими словами, сама форма, а также некоторые компонентами являются контейнерами для компонентов.

Так же был использован многоуровневый шаблон. Система разбивается на уровни, которые на диаграмме изображаются один над другим. Каждый уровень может вызывать только уровень на 1 ниже него. Таким образом разработку каждого уровня можно вести относительно независимо, что повышает модифицируемость системы. Недостатками данного подхода являются усложнение системы и снижение производительности.

Использование различных компонентов при построении форм облегчает взаимодействие пользователя и компьютера.

# 4 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 программное средство 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 программном средстве, 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 исключения 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 программного средства 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.
* тестирование зависит от контекста. Выбор методологии, техники и типа тестирования будет напрямую зависеть от природы самого программного средства. 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.

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, программных средств, 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 программного средства;
* 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.

Тестирование программного средства 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.

В процессе тестирования в основном использовалось функциональное и нефункциональное тестирование. Были разработаны тест-кейсы, представленные в таблице 2. Тест кейс – это набор входных значений, предусловий выполнения, шагов и ожидаемых результатов, определяемых для покрытия определенных тестовых условий.

Таблица 2 – Тест Кейс

|  |  |  |  |
| --- | --- | --- | --- |
| Название операции | Описание тестового случая | Ожидаемый результат | Фактический результат |
| 1 | 2 | 3 | 4 |
| Запуск | Запуск приложения 1.Нажать Тест Кетела.exe | 1.Откроется главная форма приложения | 1.Открылась главная форма приложения |
| Регистрация пользователя | Зарегистрировать пользователя  1.Нажать на окно ввода ФИО  2.Заполнить поле  4.Нажать на окно ввода группы  5.Заполнить поле  6.Нажать кнопку «Начать тест» | 1.Откроется форма прохождения теста | 1.Открылась форма прохождения теста |

Продолжение таблицы 2

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| Регистрация пользователя | Зарегистрировать пользователя  1.Нажать на окно ввода ФИО  2.Заполнить поле  4.Нажать на окно ввода группы  5.Оставить поле пустым  6.Нажать кнопку «Начать тест» | 1.Произойдет вывод сообщения "Не все данные введены правильно!" | 1.Вывод сообщение "Не все данные введены правильно!" |
| Регистрация пользователя | Зарегистрировать пользователя  1.Нажать на окно ввода ФИО  2.Оставить поле пустым  4.Нажать на окно ввода группы  5.Заполнить поле  6.Нажать кнопку «Начать тест» | 1.Произойдет вывод сообщения "Не все данные введены правильно!" | 1.Вывод сообщение "Не все данные введены правильно!" |
| Прохождение теста | Переход к следующему вопросу теста 1.Выбрать один из вариантов ответа 2.Нажать на кнопку «ответить» | 1.Откроется форма со следующем вопросом теста | 1.Открылась форма со следующем вопросом теста |
| Прохождение теста | Переход к следующему вопросу теста 1.Не выбирать ни один из вариантов ответа  2.Нажать на кнопку «ответить» | 1.Произойдет вывод сообщения "Ответ не был дан" | 1.Вывод сообщение "Ответ не был дан" |
| Прохождение теста | Выход во время прохождения теста  2.Нажать на кнопку «выход» во время прохождения теста | 1.Форма прохождения теста будет закрыта | 1.Форма прохождения теста была закрыта |

Продолжение таблицы 2

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| Вывод результата | Проверка вывода результата  1.Ответить на вопрос №104 и нажать кнопку «ответить»  2.Ответить на сообщение «Все вопросы пройдены» «да» | 1.Произойдет вывод сообщения «Все вопросы пройдены»  2.Откроется окно «Результаты» | 1.Вывод сообщения «Все вопросы пройдены»  2.Открылось окно «Результаты» |
| Просмотр результатов | Проверка просмотра всех результатом  1.Нажать на кнопку далее | 1.Выполнится переход на диалоговое окно со следующим фактором | 1.Выполнился переход на диалоговое окно со следующим фактором |
| Сохранение результатов | Проверка сохранения результатов  1.Ответить на 104 вопроса  2.Ответить на сообщение «Все вопросы пройдены» «да» | 1.Произойдет вывод сообщения «Все вопросы пройдены»  2.Файл с результатом появится в указанной папке | 1.Вывод сообщения «Все вопросы пройдены»  2.Файл с результатом появился в указанной папке |

Тестирование программного средства проходило в ходе и после его 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.

# 5 Руководство пользователя

# 5.1 Описание процесса установки и запуска приложения

Данное приложение нуждается в установке. Для этого необходимо иметь файл Test.exe. Необходимо два раза щёлкнуть левой кнопкой мыши по ярлыку и выбрать место сохранения программы на жестком диске.

ПО не нуждается в дополнительных файлах, но после сохранения будет создан файл ФИО.txt. Он хранит в себе информацию, которая была получена после работы с программой. Если эти данные нужны, то файл должен находиться рядом с файлом «Test».

# 5.2 Демонстрационный пример работы приложения

Для запуска приложения необходимо щелкнуть левой кнопкой мыши по ярлыку с названием «Test».

После запуска будет открыто окно, которое предоставляет инструкцию и информацию о тесте пример окна, предоставляемого эту информацию представлен на рисунке 7.

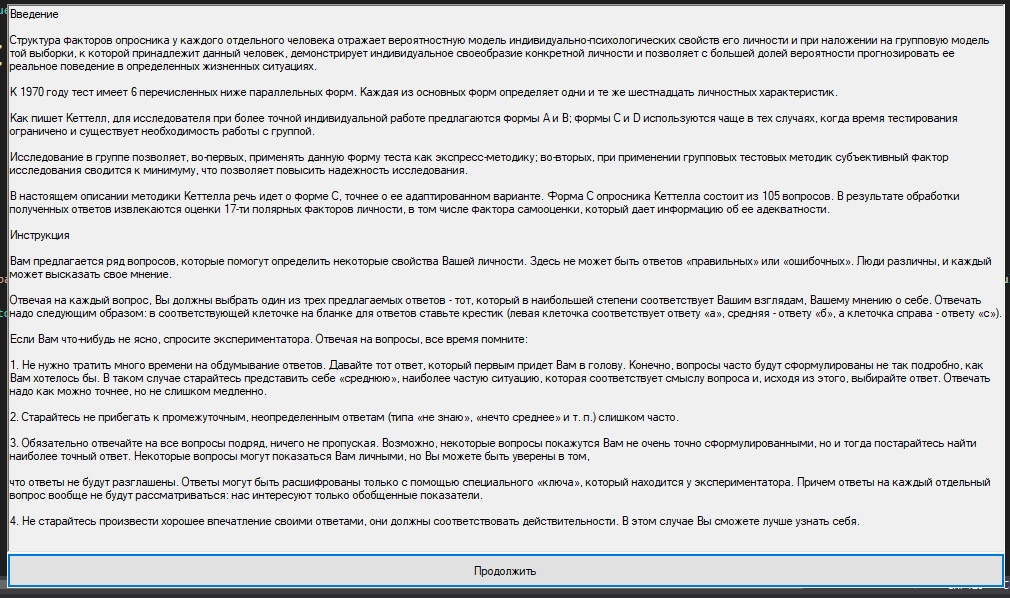


Рисунок 7 – Окно с ознакомительной информацией

После просмотра обучающей информации необходимо нажать далее ,после чего появятся два поля для ввода ФИО и номера группы пользователя, рабочее окно представлено на рисунке 8.

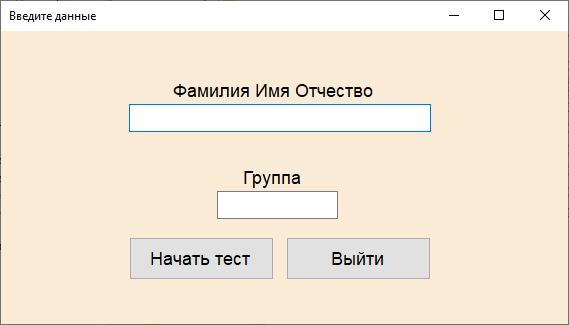


Рисунок 8 –Окно для ввода личных данных

Если информация была введена неверно или поля для ввода были оставлены пустыми появиться сообщение о неправильном вводе, пример сообщения представлен на рисунке 9.

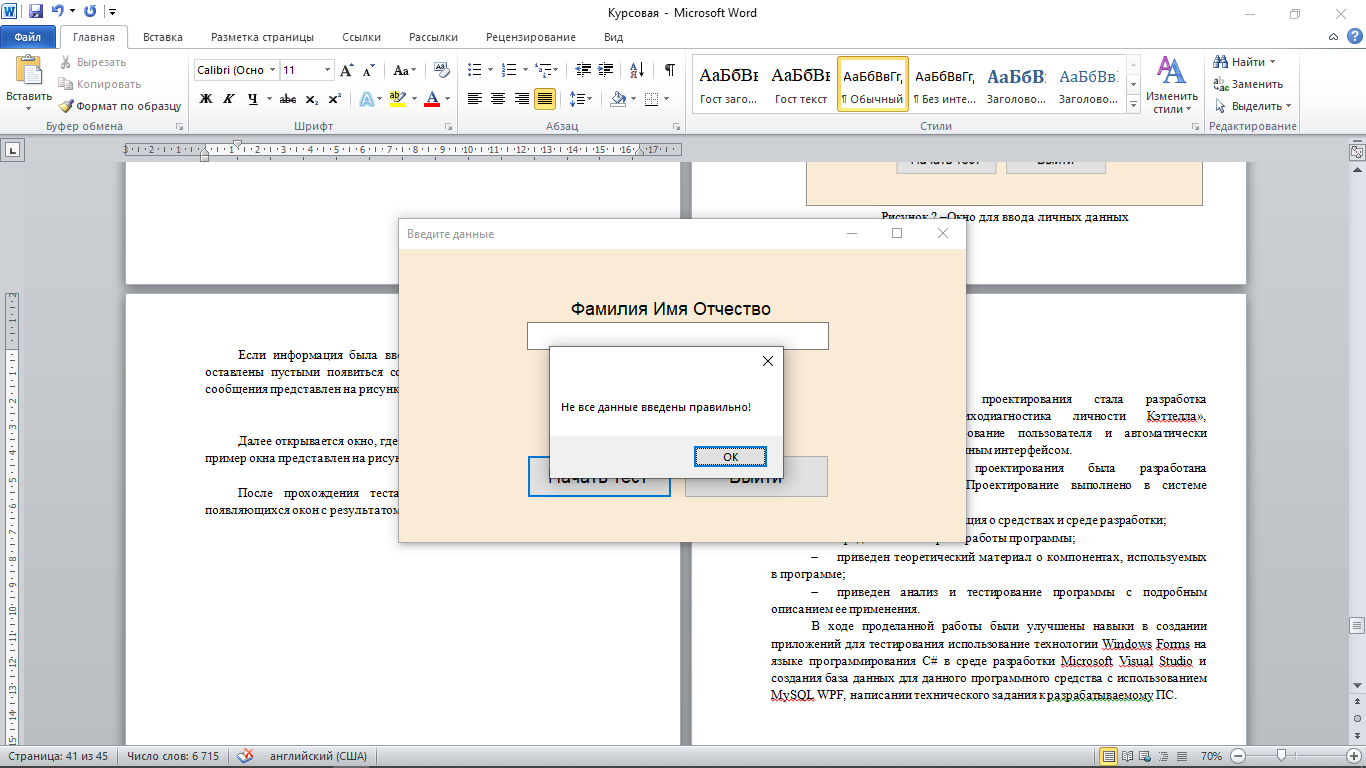


Рисунок 9 – Окно предупреждение о неправильности введенных данных

Далее открывается окно, где представлен вопрос и варианты ответа, пример окна представлен на рисунке 10.

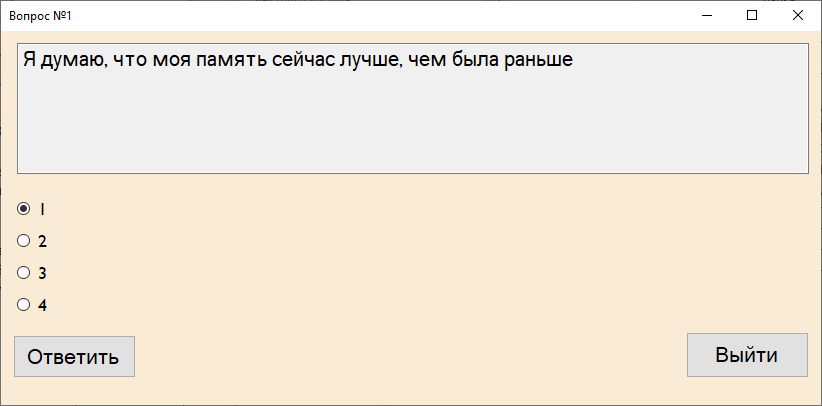


Рисунок 10 – Окно вопроса и ответов теста

Если ответ дан не был, но пользователь нажмет кнопку «ответить» будет выведено сообщение о том, что нужно дать ответ. Окно сообщения представлено на рисунке 11.

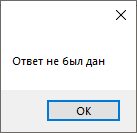


Рисунок 11 – Окно предупреждение о не данном ответе

После прохождения теста появиться окно о том, что все вопросы пройдены, а после этого будут представлено несколько последовательно появляющихся окон с результатом, пример представлен на рисунке 12.

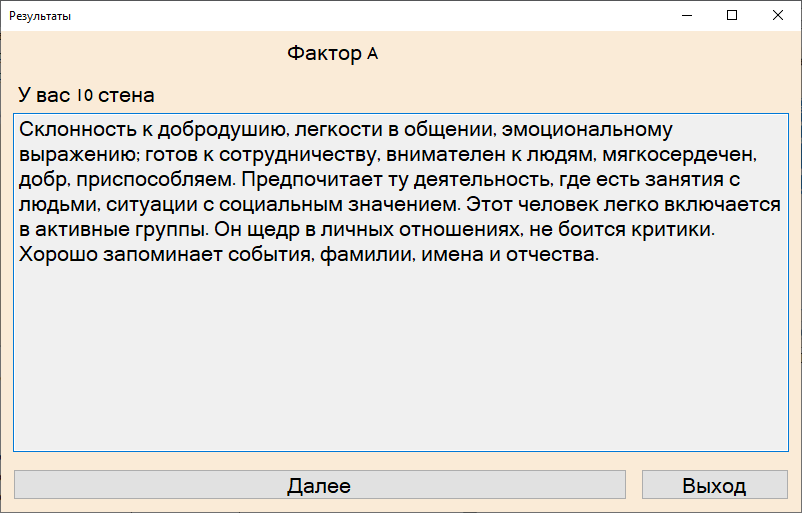


Рисунок 12 – Окно результата

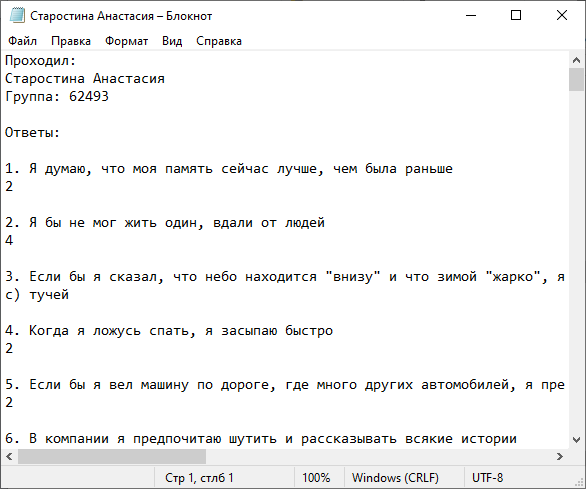


Рисунок 13 – Сохранение результата в файл

После просмотра всех своих характеристик, представленных в последовательности нескольких факторов ответ будет автоматически сохранен в файл.

# Заключение

Результатом курсового проектирования стала разработка программного средства «Психодиагностика личности Кэттелла», позволяющее проводить тестирование пользователя и автоматически вычислять результат, с проработанным интерфейсом.

В ходе курсового проектирования была 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.

В ходе проделанной работы были улучшены навыки в создании приложений для тестирования использование технологии Windows Forms на языке программирования C# в среде разработки Microsoft Visual Studio и создания база данных для данного программного средства с использованием MySQL WPF, написании технического задания к разрабатываемому ПС.

За период курсового проектирования была смоделирована информационная модель, которая была нормализована в схему базы данных. А также выполнена физическая реализация базы данных и разработано функционирующее приложение для тестирования личности методом Кеттелла, разработаны модели диаграммы деятельности и сущность-связи для лучшего понимая программы, проведено тестирование программы, которое показывает, что программа работает корректно и готова к использованию, а также разработана пояснительная записка, которая содержит всю необходимую информацию о разработанном программном средстве.

# Список использованных источников

Литература

1 Медведев, В.И. M 42 Особенности объектно-ориентированного программирования на C++/CLI, C# и Java/ В.И. Медведев, 2-е изд., испр. и доп. - Казань: РИЦ «Школа», 2010

2 Лазицкас, Е.А. Методические рекомендации по выполнению курсового проекта / Е.А. Лазицкас, О.Н. Винничук. -Мн: 2020

Электронный ресурс

3 Метод психодиагностики личности Кеттелла [Электронный ресурс]. – Режим доступа: <https://effecton.ru/260.html> - Дата доступа: 10.03.2020

4 Принципы тестирования [Электронный ресурс]. – Режим доступа: <https://intellect.icu/printsipy-testirovaniya-6186> - Дата доступа: 10.03.2020

# Приложение А

(обязательное)

Текст программы

namespace NDiplom

{

public partial class Main : Form

{

public Main()

{

InitializeComponent();

}

//Флаг выхода из приложения

private bool Exit = false;

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

{

//Проверка на корректность введения данных пользователя

if (textBox1.Text.Length < 3 || textBox2.Text.Length < 1)

{

MessageBox.Show("Не все данные введены правильно!");

return;

}

//Показать пользователю диалог для каждого вопроса

QuestionForm qf = new QuestionForm();

this.Hide();

for (int i = 0; i < QuestionList.QuestionCount; i++)

{

Question quest = QuestionList.GetQuestionFromNumber(i);

}

qf.Close();

this.Show();

//Выход из приложения если была нажата кнопка

if (Exit)

{

Application.Exit();

return;

}

//Высчитать значения всех факторов

FactorList.CalculateFactors();

//Найти стену для каждого фактора по таблице

PointsFormingTable.FormFactorWalls();

SaveResult();

MessageBox.Show("Все вопросы пройдены");

//Показать пользователю диалог с результатами теста

for (int i = 0; i < FactorList.factors.Count; i++)

{

Factor f = FactorList.factors[i];

ResultShowingForm rsf = new ResultShowingForm(f.GetResultText(), f.FactorWall + 1, f.Name);

rsf.ShowDialog();

}

textBox1.Text = "";

textBox2.Text = "";

QuestionList.Reset();

FactorList.Reset();

}

/// <summary>

/// Сохранение результатов прохождения теста в файл

/// </summary>

private void SaveResult()

{

//Перезаписать файл с результатами если такой имеется

if (File.Exists($"{textBox1.Text}.txt")) File.Delete($"{textBox1.Text}.txt");

//Начать запись в файл

FileStream fs = File.Create($"{textBox1.Text}.txt");

StreamWriter sw = new StreamWriter(fs);

sw.WriteLine($"Проходил: \r\n{textBox1.Text}\r\nГруппа: {textBox2.Text}\r\n\r\nОтветы:\r\n");

//Записать в файл ответ для каждого вопроса

for (int i = 0; i < QuestionList.questions.Count; i++)

{

sw.WriteLine((i + 1).ToString() + ". " + QuestionList.questions[i].QuestionText);

int[] questions = QuestionList.questions[i].ChoosenAnswers;

if (questions == null || questions.Length < 1)

{

sw.WriteLine("Ответ дан не был\r\n");

}

else

{

string toWrite = "";

for (int k = 0; k < questions.Length; k++)

{

toWrite += QuestionList.questions[i].Answers[questions[k]].Text + " ";

}

sw.WriteLine(toWrite+"\r\n");

}

}

//Записать в файл значения стен факторов

sw.WriteLine("\r\n" + new string('=', 40) + "\r\nФакторы: ");

for (int i = 0; i < FactorList.factors.Count; i++)

{

Factor f = FactorList.factors[i];

sw.WriteLine($"Фактор {f.Name} со стеной {f.FactorWall+1}");

}

//Завершить запись

sw.Close();

fs.Close();

}

private void Main\_Load(object sender, EventArgs e)

{

//Проверить наличие файла с вопросами

if (!QuestionList.Deserialize())

{

if (MessageBox.Show("Файла с вопросами не найдено! Программа не сможет работать корректно! Желаете создать пустой файл-шаблон?", "Ошибка", MessageBoxButtons.YesNo) == DialogResult.Yes)

{

List<Question> lq = new List<Question>()

{

new Question(0, "Вопрос 1", new Answer[]{ new Answer("1", 4), new Answer("2", 3), new Answer("3", 2), new Answer("4", 1) }, Question.AnswerType.One),

new Question(1, "Вопрос 2", new Answer[]{ new Answer("1", 1), new Answer("2", 3), new Answer("3", 2) }, Question.AnswerType.Multiple),

};

QuestionList.questions = lq;

QuestionList.Serialize();

}

StartButton.Enabled = false;

}

//Проверить наличие файла с факторами

if (!FactorList.Deserialize())

{

if (MessageBox.Show("Файла с факторами не найдено! Программа не сможет работать корректно! Желаете создать пустой файл-шаблон?", "Ошибка", MessageBoxButtons.YesNo) == DialogResult.Yes)

{

List<Factor> fl = new List<Factor>()

{

new Factor("A", 1),

new Factor("B", 3, 5, 10),

};

FactorList.factors = fl;

FactorList.Serialize();

}

StartButton.Enabled = false;

}

Intro intro = new Intro();

intro.ShowDialog();

}

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

{

//Закрыть программу

this.Close();

}

}

}

public static class QuestionList

{

/// <summary>

/// Путь для сериализации/десериализации

/// </summary>

private const string serializePath = "Questions.xml";

/// <summary>

/// Список всех вопросов после десиализации

/// </summary>

public static List<Question> questions = new List<Question>();

/// <summary>

/// Общее кол-во вопросов

/// </summary>

public static int QuestionCount { get { return questions.Count; } }

private const bool UseDB = false;

/// <summary>

/// Получить вопрос по номеру

/// </summary>

/// <param name="number">Номер вопроса</param>

/// <returns>Возвращает вопрос по его номеру. Если такового нет, то возвращает null</returns>

public static Question GetQuestionFromNumber(int number)

{

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

{

Question question = questions[i];

if (question.Number == number) return question;

}

return null;

}

/// <summary>

/// Сбросить все к изначальному варианту

/// </summary>

public static void Reset()

{

questions = new List<Question>();

Deserialize();

}

/// <summary>

/// Сериализовать список вопросов в файл

/// </summary>

public static void Serialize()

{

XmlSerializer formatter = new XmlSerializer(typeof(List<Question>));

using (FileStream fs = new FileStream(serializePath, FileMode.OpenOrCreate))

{

formatter.Serialize(fs, questions);

}

}

/// <summary>

/// Десериализовать список вопросов из файла

/// </summary>

public static bool Deserialize()

{

if (UseDB)

{

MySqlConnection connection = new MySqlConnection(Config.ConnectionString);

connection.Open();

MySqlCommand command = new MySqlCommand("select \* from questions;", connection);

MySqlDataReader reader = command.ExecuteReader();

List<(int number, string text)> questions = new List<(int number, string text)>();

while (reader.Read())

{

int qNumber = int.Parse(reader.GetString(0));

string qText = reader.GetString(1);

questions.Add((qNumber, qText));

}

reader.Close();

command = new MySqlCommand("select \* from answervar;", connection);

reader = command.ExecuteReader();

List<Answer> answers = new List<Answer>();

while (reader.Read())

{

int number = (int)reader.GetValue(0);

string answerText = reader.GetString(1);

int points = (int)reader.GetValue(2);

Answer a = new Answer(answerText, points);

answers.Add(a);

}

answers.RemoveAt(0);

foreach (var item in questions)

{

Question q = new Question(item.number, item.text, answers.ToArray(), Question.AnswerType.One);

QuestionList.questions.Add(q);

}

return true;

}

//Проверить наличие файла

if (!File.Exists(serializePath))

{

return false;

}

XmlSerializer formatter = new XmlSerializer(typeof(List<Question>));

using (FileStream fs = new FileStream(serializePath, FileMode.Open))

{

questions = (List<Question>)formatter.Deserialize(fs);

}

public partial class QuestionForm : Form

{

public QuestionForm()

{

InitializeComponent();

}

//Тип вопроса, показываемого формой

private Question.AnswerType at;

//Список кнопок для ответов на вопрос

private List<ButtonBase> answerButtons = new List<ButtonBase>();

//Показываемый формой вопрос

private Question currentQuestion;

/// <summary>

/// Установить форме новый вопрос в качестве активного

/// </summary>

/// <param name="quest">Вопрос</param>

public void SetQuestion(Question quest)

{

//Сбросить значения к стандартным

checkBox1.Visible = false;

checkBox2.Visible = false;

checkBox3.Visible = false;

checkBox4.Visible = false;

...

radioButton4.Checked = false;

//задать значение полям

at = quest.answerType;

answerButtons = new List<ButtonBase>();

currentQuestion = quest;

//Выбрать нужные для ответов кнопки в зависимости от типа ответа на вопрос

switch (at)

{

case Question.AnswerType.One:

answerButtons.Add(radioButton1);

…

answerButtons.Add(checkBox3);

answerButtons.Add(checkBox4);

break;

}

//Вывести вопрос и ответ на вопросы

this.Text = $"Вопрос №{currentQuestion.Number}";

textBox1.Text = currentQuestion.QuestionText;

for (int i = 0; i < currentQuestion.answerCount; i++)

{

answerButtons[i].Visible = true;

answerButtons[i].Text = currentQuestion.Answers[i].Text;

}

}

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

{

//Получить выбранные ответы

List<int> answer = new List<int>();

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

{

if (answerButtons[i] is RadioButton)

{

RadioButton rb = (RadioButton)answerButtons[i];

if (rb.Checked) answer.Add(i);

}

else if (answerButtons[i] is CheckBox)

{

CheckBox cb = (CheckBox)answerButtons[i];

if (cb.Checked) answer.Add(i);

}

}