МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ИНСТИТУТ ТЕХНОЛОГИЙ (ФИЛИАЛ) ФЕДЕРАЛЬНОГО ГОСУДАРСТВЕННОГО БЮДЖЕТНОГО ОБРАЗОВАТЕЛЬНОГО УЧРЕЖДЕНИЯ ВЫСШЕГО ОБРАЗОВАНИЯ «ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ» В Г. ВОЛГОДОНСКЕ РОСТОВСКОЙ ОБЛАСТИ

(Институт технологий (филиал) ДГТУ в г. Волгодонске)

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

для выполнения курсовой работы

по дисциплине
«Технологии программирования»
для обучающихся по направлению подготовки
09.03.02 Информационные системы и технологии
программа бакалавриата «Информационные системы»

г. Волгодонск
2021

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В архитектуре клиент-сервер для обработки данных выделяется специальное ядро - так называемый сервер баз данных, который принимает на себя функции обработки запросов пользователей, именуемых теперь клиентами. Сервер баз данных представляет собой программу, выполняющуюся, как правило, на мощном компьютере. Приложения-клиенты посылают с рабочих станций запросы на выборку (вставку, обновление, удаление) данных. При этом сервер выполняет всю “грязную” работу по отбору данных, отправляя клиенту только требуемую “выжимку”. Если приведенный выше пример перестроить с учетом клиент-серверной архитектуры, то приложение-клиент “получит” от сервера в качестве результата список только тех работников, которые участвуют в заданном проекте, и не более того!

Такой подход обеспечивает решение трех важных задач:

- уменьшение нагрузки на сеть
- уменьшение требований к компьютерам-клиентам
- повышение надежности и сохранение логической целостности базы данных.

Здесь приложения также выполняются, в основном, на рабочих станциях. Приложение включает модули для организации диалога с пользователем и бизнес-правила (транзакции). Ядро СУБД является общим для всех рабочих станций и функционирует на сервере. Операторы обращения к СУБД (SQL-операторы), закодированные в транзакции, не выполняются на рабочей станции, а пересылаются для обработки на сервер. Ядро СУБД транслирует запрос и выполняет его, обращаясь для этого к индексам и другим промежуточным данным. Обратно на рабочую станцию передаются только результаты обработки оператора.

В современных СУБД на сервере могут запускаться так называемые хранимые процедуры и триггеры, которые вместе с ядром СУБД образуют сервер базы данных. К хранимым процедурам можно обращаться из приложений на рабочих станциях. Это позволяет сократить размер кода прикладной программы и уменьшить поток SQL-операторов с рабочей станции, так как группу требуемых SQL-предложений можно закодировать в хранимой процедуре. Триггеры - это программы, которые выполняются ядром СУБД перед или после обновления (UPDATE, INSERT, DELETE) таблицы базы данных. Они позволяют автоматически поддерживать целостность базы данных

1 ОСНОВНЫЕ ПОЛОЖЕНИЯ

Обучающимся в процессе выполнения курсовой работы необходимо выполнить ряд требований:

Курсовая работа выполняется с учётом нормативных и законодательных актов Российской Федерации, современных теоретических исследований по проблемам организации и планирования деятельности, данных предприятия. В свою очередь результаты данной курсовой работы могут служить основой при выполнении последующих курсовых работ, а также при выполнении выпускных квалификационных работ.

Оформление курсовой работы должно соответствовать «Правилам оформления и требованиями к содержанию курсовых проектов (работ) и выпускных квалификационных работ», утвержденных Приказом ректора ДГТУ №227 от 30.12.2015г. Объём курсовой работы не должен превышать 30-40 страниц печатного текста.

Текст работы набирается на ПЭВМ на одной стороне листа формата А4. Титульный лист, лист задания, рецензия являются бланками определенного образца. Содержание работы оформляется на листе с рамкой и основной надписью формы 2, включает название всех разделов работы с указанием номера страниц, на которых помещены заголовки разделов. Текст работы оформляется на листе с рамкой и основной надписью формы 2а. Текстовый материал должен быть изложен лаконично, расчеты выполнены грамотно.

Каждая курсовая работа должна быть выполнена с учётом единых методических подходов и выстроена по общей структуре. В зависимости от специфических особенностей хозяйственной деятельности исследуемого предприятия структура работы может корректироваться.

Тема курсовой работы утверждается каждому студенту индивидуально. В качестве информационной базы используют данные действующего предприятия сферы сервиса.

По результатам выполнения курсовой работы обучающемуся выставляется оценка «зачтено» или « не незачтено».

Оценка «зачтено» выставляется обучающемуся который:

- выполнил в срок и на высоком уровне весь намеченный объем работы, определенный заданием к курсовой работе;

- продемонстрировал умение правильно определять и эффективно решать основные задачи курсовой работы;

- на дополнительные вопросы преподавателя обучающийся дал правильные ответы;

- продемонстрировал свободное владение концептуально-понятийным аппаратом, научным языком и терминологией соответствующей дисциплины.

Компетенция (и) или ее часть (и) сформированы на базовом уровне (уровень 1).

Оценка «не зачтено» ставится обучающемуся, который не выполнил поставленные в курсовой работе задачи. Оформление графической части представил на низком уровне или не представил; не исправил ошибки в ходе выполнения курсовой работы; не подготовил доклад.

Компетенция(и) или ее часть (и) не сформированы.

Ниже приведены основные технологические операции и приемы, необходимы для создания проекта информационной системы (ИС).

Общая характеристика процесса проектирования ИС

Исходные данные для проектирования ИС. Методы управления ресурсами, процессами, корпоративными знаниями (коммуникациями), как основа для проектирования ИС. Поддержка информационными технологиями методов управления: СУБД, стандарты ассоциации Workflow Management Coalition, Intranet. Риск проекта ИС. Компоненты проектирования. Стадии разработки, модели представления, уровни детализации

Информационно-логическая модель ИС

Общая схема информационно-логической модели, графовая основа

модели представления, определение структуры ИС. Модели представления ИС.

Графические средства описания различных моделей представления ИС. ЕСПД: типы документов для представления проектных решений; графическое представление элементов. Схемы классификации/детализации: схемы формирования вторичных элементов; схемы организационно-функциональной структуры; схемы требований; схемы потоков.

Функциональная модель ИС

Описание функциональной модели. Стратегии построения схем требований действий.

Основные схемы декомпозиции действий и данных функциональной модели: декомпозиция действий на основе состава выходных данных; декомпозиция действий на основе входных данных; декомпозиция действий на основе представлений о промежуточных результатах; декомпозиция действий на основе представлений о фазах обработки; декомпозиция действий на основе представлений об альтернативных действиях.

Функциональная модель существующей технологии обработки данных. Варианты преобразования функциональной модели. Общая схема разработки функциональной модели.

Модели данных

Иерархия моделей данных; определения модели данных; уровни представления (концептуальный, логический, физический); локальная (внешняя) модель; композиционная модель данных.

Некоторые концептуальные модели данных; реляционная модель данных; ER - модель; функциональная модель данных; модель данных КОДАСИЛ; модель с классификацией информационных объектов (модель Смиттов). Нормализация концептуальной модели данных и целостность данных: нормальные формы концептуальной модели данных; параметризация модели данных; ссылочная целостность. Агрегирование объектов в предметные базы данных.

Сравнение различных моделей данных концептуального уровня. Методики конструирования моделей данных: методика построения локальных моделей данных на основе выделения базовых действий; методика построения локальных моделей данных на основе выделения базовых объектов.

Методика разработки СУБД ориентированных схем данных на основе нормализованной модели данных; методика разработки типов данных на основе синтаксиса языка управления заданиями. Диаграммы потоков действий-данных (модель де-Марко).

Построение ER-модели данных.

Объектно-ориентированная модель предметной области (проект расширенной концептуальной модели)

Основные компоненты объектно-ориентированной модели предметной области: метаобъект, объект, определение атрибута, связи. Спецификация атрибутов: простые первичные показатели, ссылки, копии; категории; ключи; вычисляемые показатели; вычисляемые связи.

Траектории движения объекта в пространстве параметров. Специальные объекты: состояния, события, работы (операции).

Обеспечение синхронности данных. Регламент.

Программно-ориентированные модели представления ИС

Основные компоненты объектно-ориентированной модели предметной области: метаобъект, объект, определение атрибута, связи. Спецификация атрибутов: простые первичные показатели, ссылки, копии; категории; ключи; вычисляемые показатели; вычисляемые связи.

Платформа клиент-сервер. Согласованное управление: транзакции и серверы баз данных, уровни разграничения транзакций, переход от запросов к хранимым процедурам.

GUI (Graphical User Interface , Графический интерфейс пользователя), MS Windows.

Структура модулей

Типы данных. Описание структуры модулей. Процедуры. Пакеты.

Задачи. Обмены. Классы. Компоненты. Пользовательский интерфейс: объектный подход к организации пользовательского интерфейса.

Конструирование последовательных управляющих структур

Приемы структурирования для последовательных управляющих структур: метод дублирования кодов. Метод введения переменной состояния; метод введения флагов состояния.

Проектирование логики на основе асинхронных взаимодействий

Базовые варианты обработки точек входа. Логика асинхронных взаимодействий, доступ к переменным состояния и событиям.

Логический анализ структур ИС

Типизированные множества и отношения: основные операции. Прочность и сцепление компонентов ИС. Анализ информационной связности систем.

Анализ информационной связности действий. Анализ функциональной связности данных. Анализ функциональной связности систем.

Распределение обработки данных на основе анализа структур ИС

Формы распределенных данных. Организация синхронности данных. Компоновка распределенной обработки. Доступ к данным в локальной сети.

Анализ производительности ИС

Временной анализ блок-схем. Сравнение моделей с экспоненциальным и постоянным распределением временами обслуживания. Оценка вероятности превышения заданного времени ответа в ИС. Выбор емкости буферного накопителя.

Представление СМО в виде взаимодействующих задач. Модель M/G/1 - FIFO.

Замкнутая модель массового обслуживания с конечным числом источников. Циклическое обслуживание с квантованием. Оценка производительности и времени отклика. Субъективная производительность ИС.

Управление проектами ИС

Принятие решения руководителем (подготовка, принятие и реализация решения). Психологические аспекты принятия решений в процессе проектирования.

Организационные формы управления проектами: структуры управления проектами, функции участников проекта. Инвестиционный проект. Типы и основные группы инвестиций инвестиций.

Структура технико-экономических исследований проекта. Структура прединвестиционных исследований проектов.

Оценка инвестиционной привлекательности проекта. Источники и формы финансирования проектов. Отбор и сертификация проектов.

2 ПРИМЕРНОЕ СОДЕРЖАНИЕ ПОЯСНИТЕЛЬНОЙ ЗАПИСКИ К КУРСОВОМУ ПРОЕКТУ

При выполнении курсовой работы студент должен опираться на знания, полученные им при изучении теоретических разделов дисциплины, и на навыки и умения, приобретенные в ходе выполнения им лабораторного практикума по дисциплине, так как совокупность лабораторных работ является прототипом данной курсовой работы.

Пояснительная записка к курсовой работе должна быть оформлена в соответствие со стандартом предприятия и требованиями Единой Системы Программной Документации (ЕСПД) (смотри Приложение).

Пояснительная записка включает в себя:

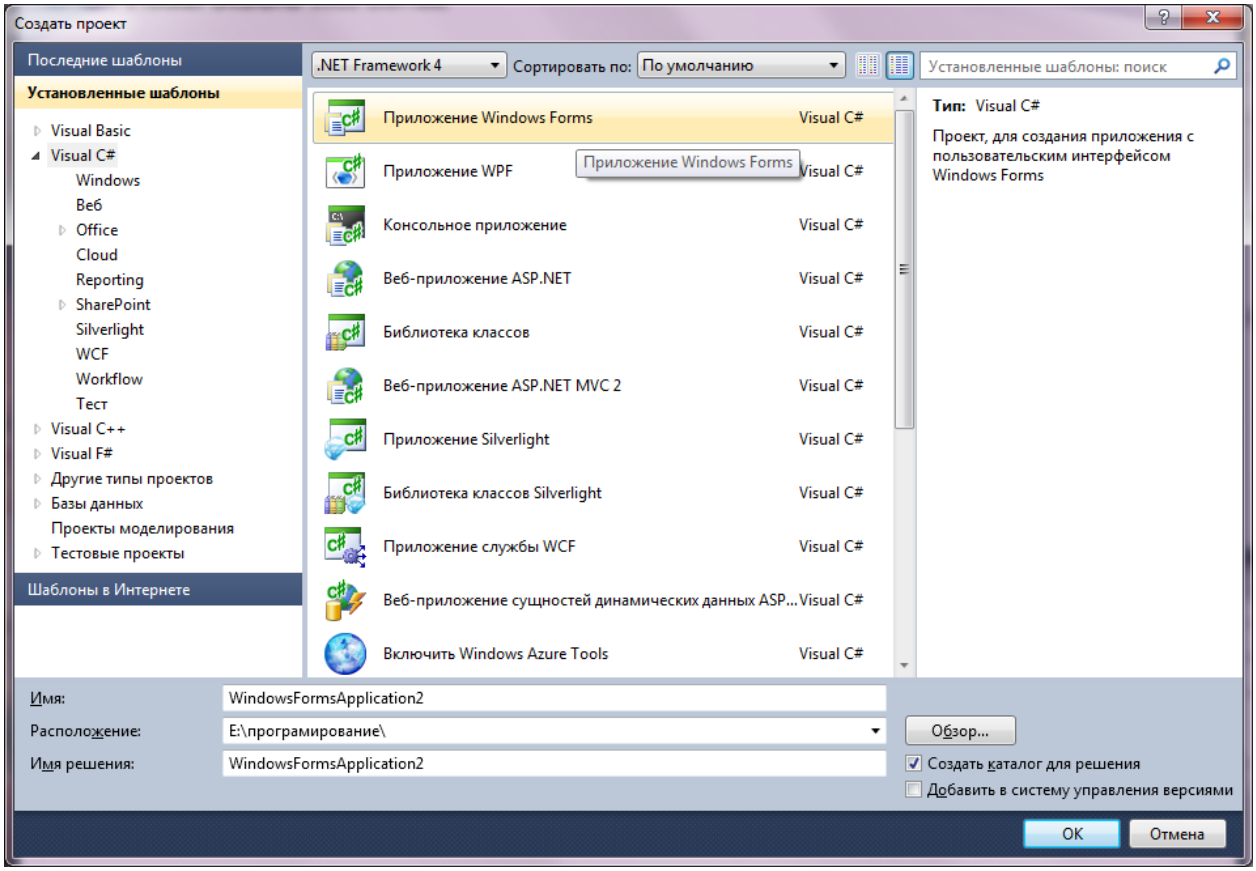
- Введение, в котором приводится краткий обзор современных технологий высокоуровневого программирования и обоснование выбора среды программирования, отладки и тестирования программы.
- Постановку задачи и описание на естественном языке обобщенного алгоритма программы.
- Описание подробного алгоритма решения задачи с учетом выбранного языка программирования и требований ЕСПД (смотри Приложение).
- Исходный текст программы на выбранном языке программирования с подробными комментариями.
- Описание процесса компиляции и тестирования программы.
- Заключение с выводами и рекомендациями по возможной модификации программы.
- Экранные формы.

Отдельно прилагаются файлы с исходным текстом программы, исполняемым модулем и текстовым файлом с исходными данными.

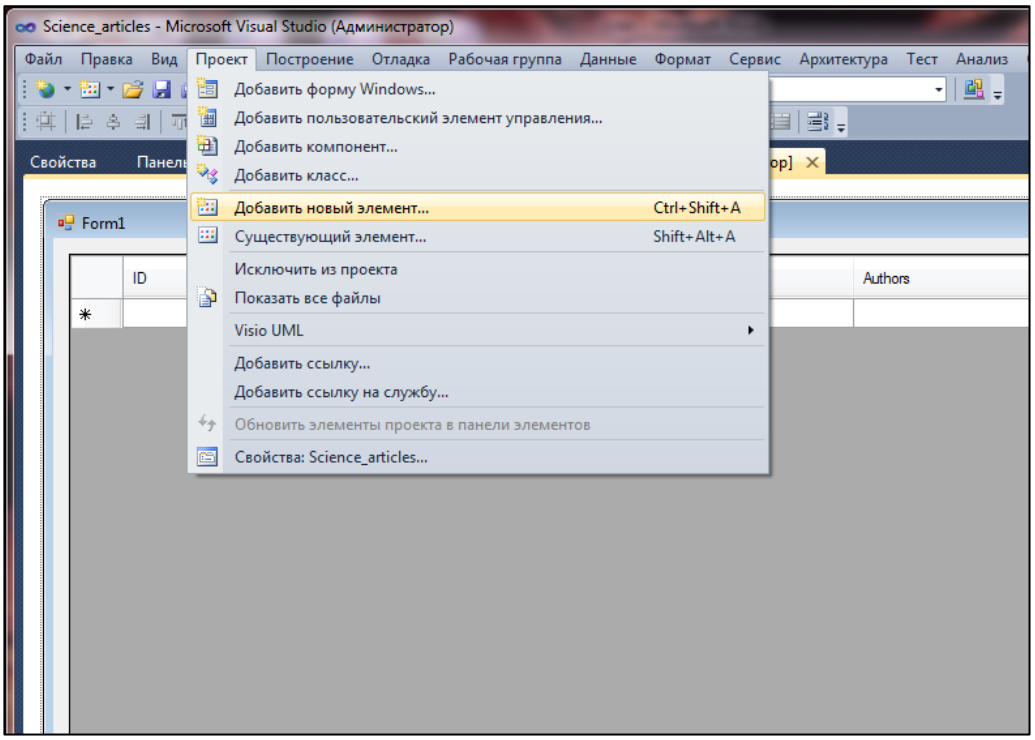
3 ПРОЕКТИРОВАНИЕ SQL БАЗЫ ДАННЫХ

В данном разделе приведен пример проектирования SQL базы данных для предметной области «Книги» в среде СУБД MS SQL и Microsoft Visual Studio 2010.

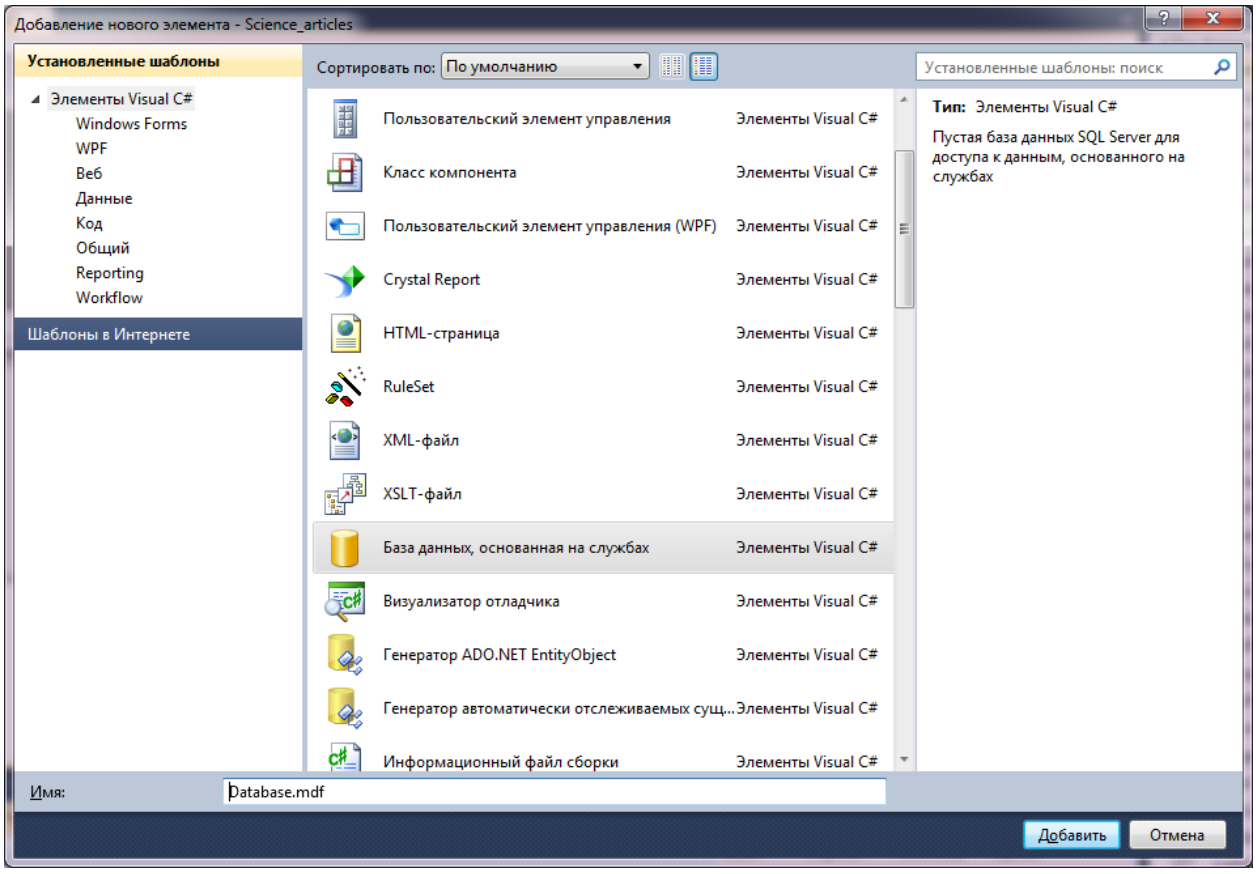
Открываем Microsoft Visual Studio 2010. Выбираем язык программирования C#, создаём проект (Windows Forms).



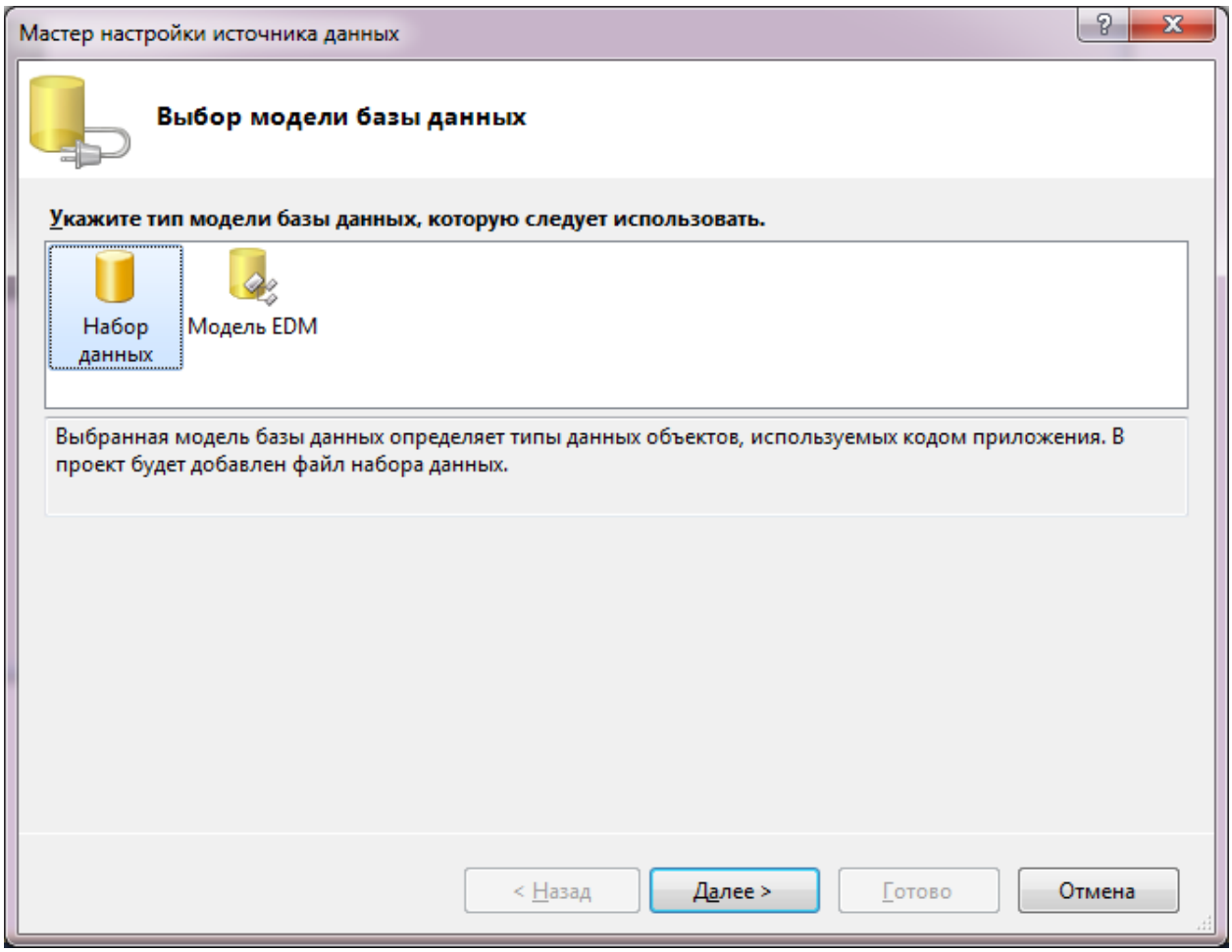
Переходим в «проект» и выбираем «Добавить новый элемент».



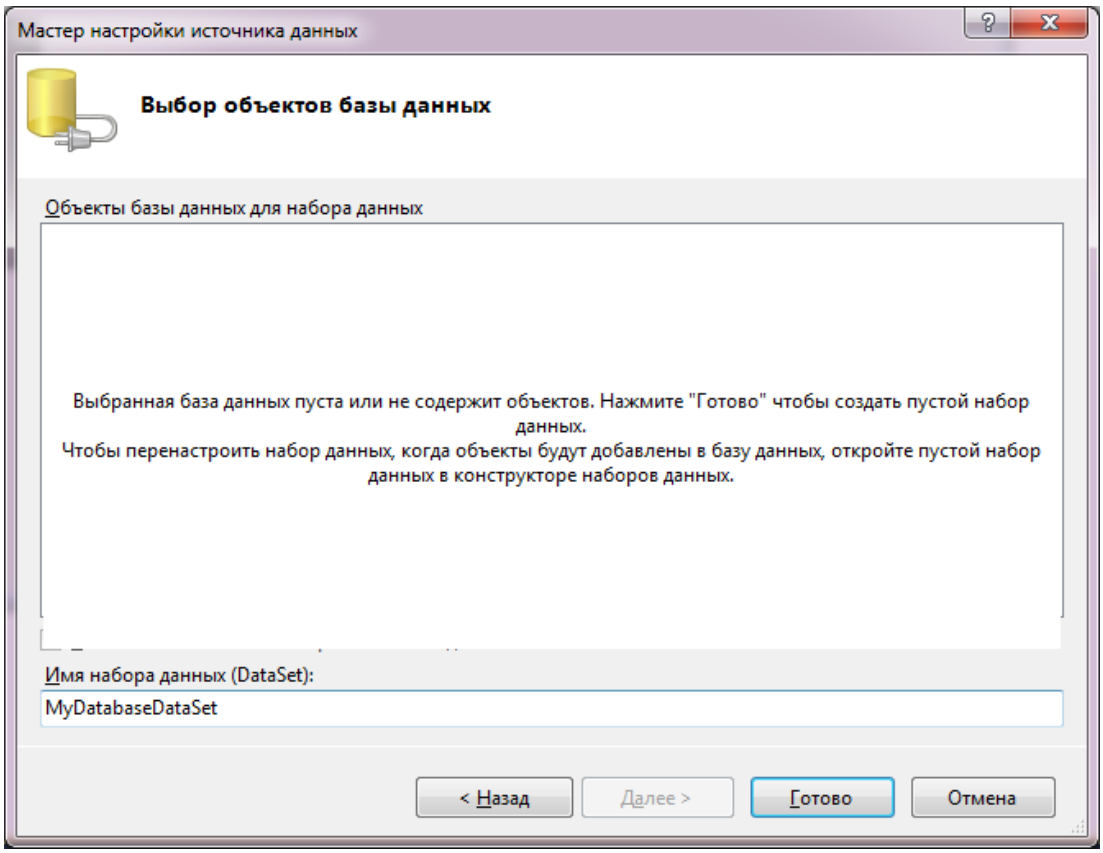
Появляется окно «Добавления нового элемента» для того чтобы создать SQL базу даных выбираем «Базы данных основаные на службах». Также выбираем имя будущей базы данных.



Появляется окно мастера настройки источника данных выбираем «набор данных» и далее

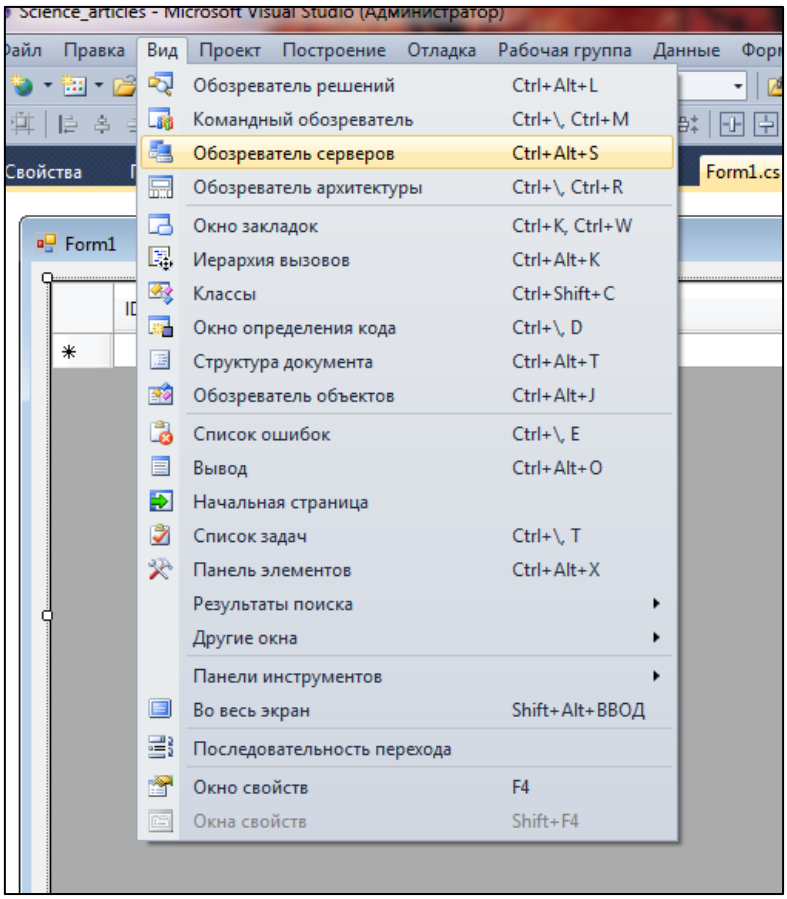


Далее выбираем объекты базы данных, здесь мы задаём имя набора данных и нажимаем готово.

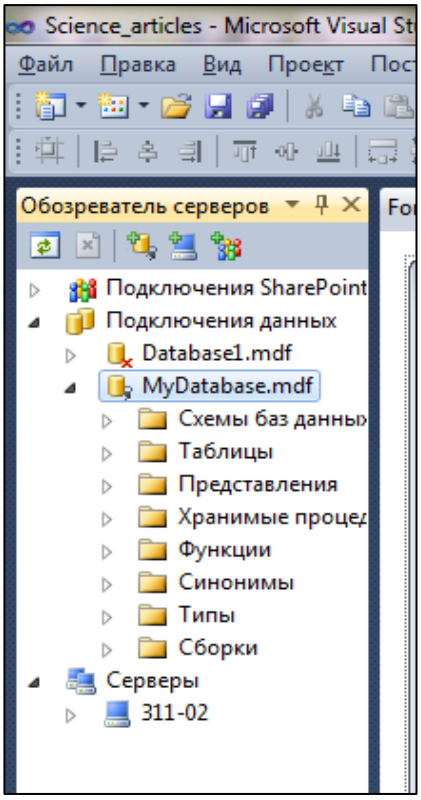


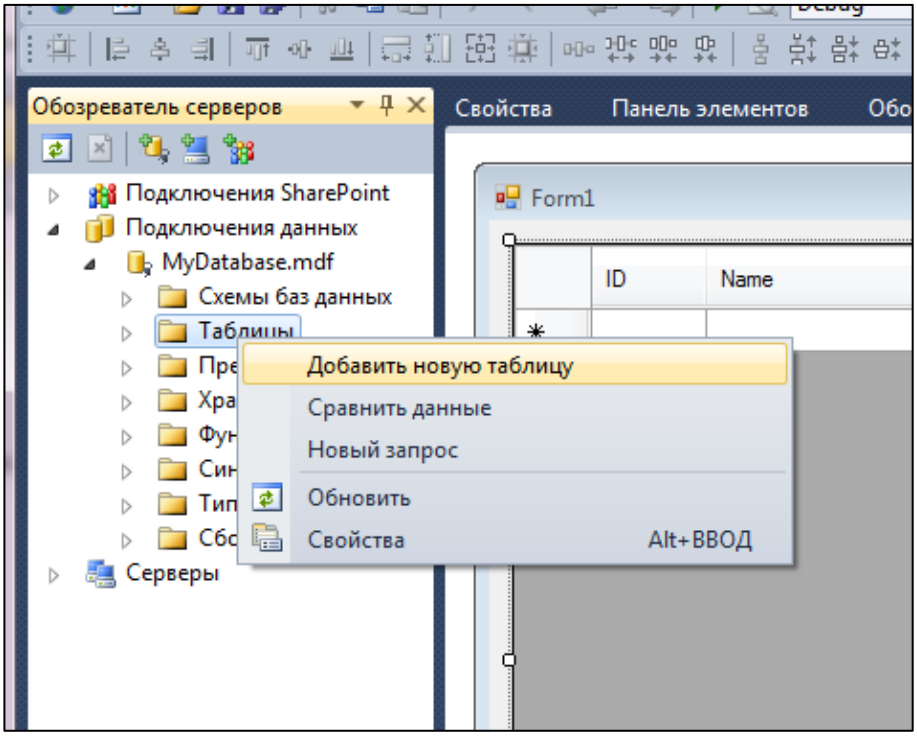
База данных подключена к проекту.

Переходим во вкладку «вид» и выбираем «обозреватель серверов».

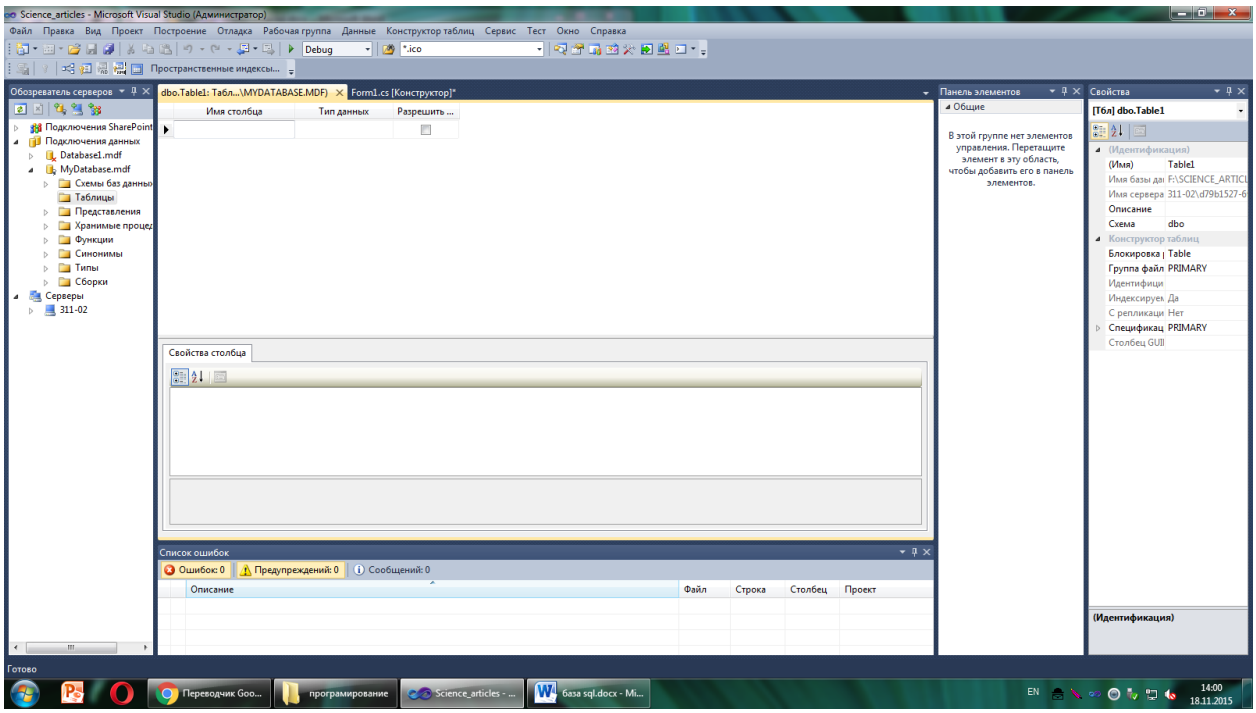


В обозревателе решений открываем папку нашей базы данных, щёлкаем правой кнопкой мыши по папке таблицы и создаём новую таблицу

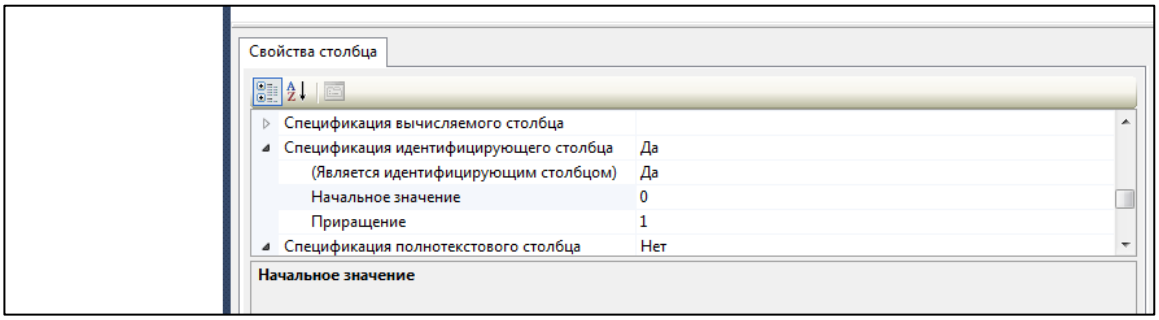




Открывается конструктор таблиц.

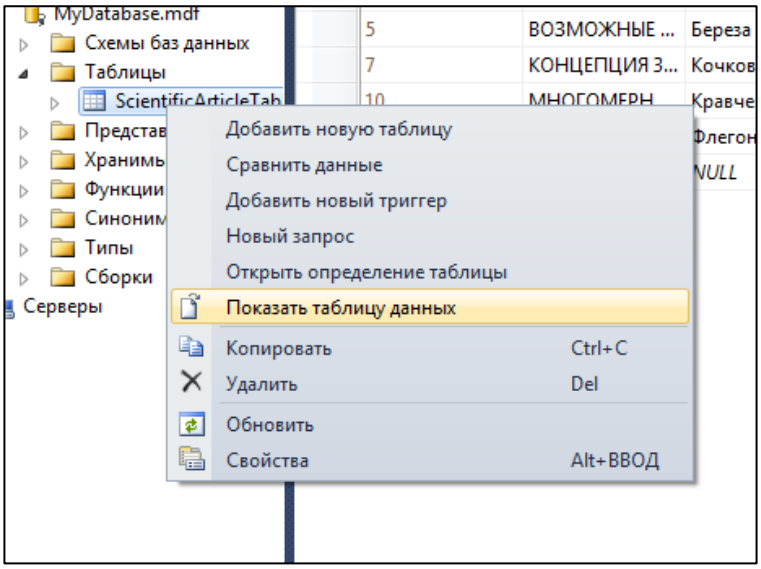


В конструкторе мы создаём столбцы (поля) нашей таблицы. Как правило, в базе данных первое поле – это поле первичного ключа ID. Это поле задаёт уникальное значение для каждой записи (строки) в таблице. Также устанавливаем на поле ID идентифицирующий столбец (автоинкримент) что - бы значения в этом поле автоматически изменялись.

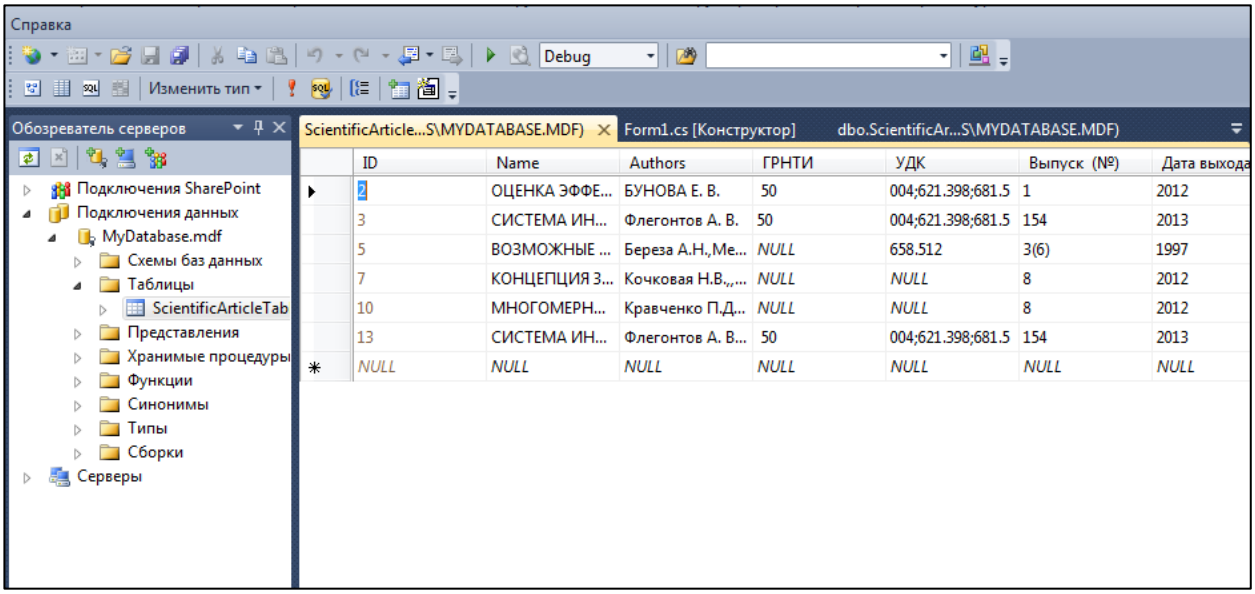


Далее создаём остальные поля таблицы, задавая им соответствующий формат данных. После того как мы создали все столбцы мы сохраняем нашу таблицу и она появляется в обозревателе решений.

Щёлкаем по таблице правой кнопкой мыши и выбираем показать таблицы данных.

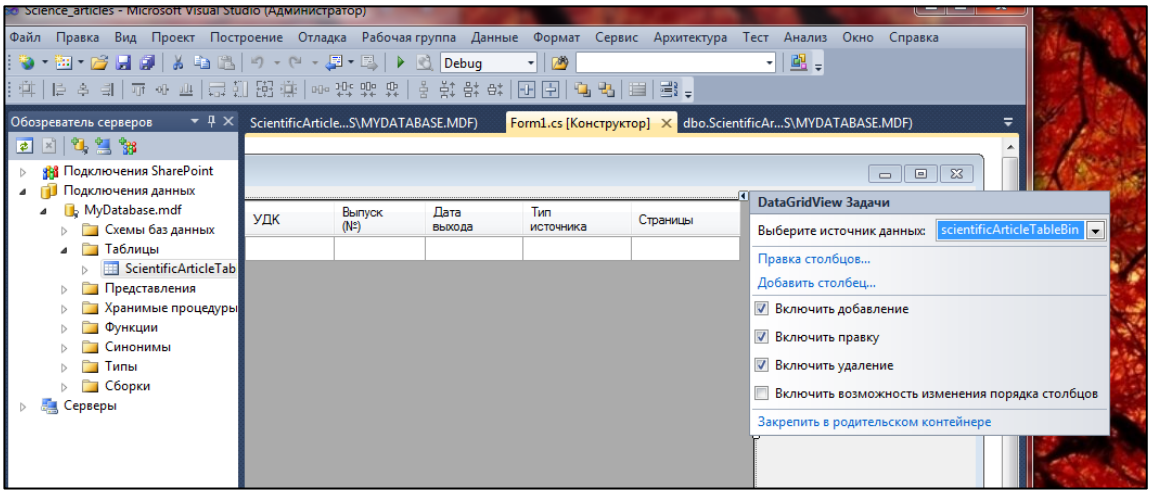


Далее заполняем нашу таблицу данными.

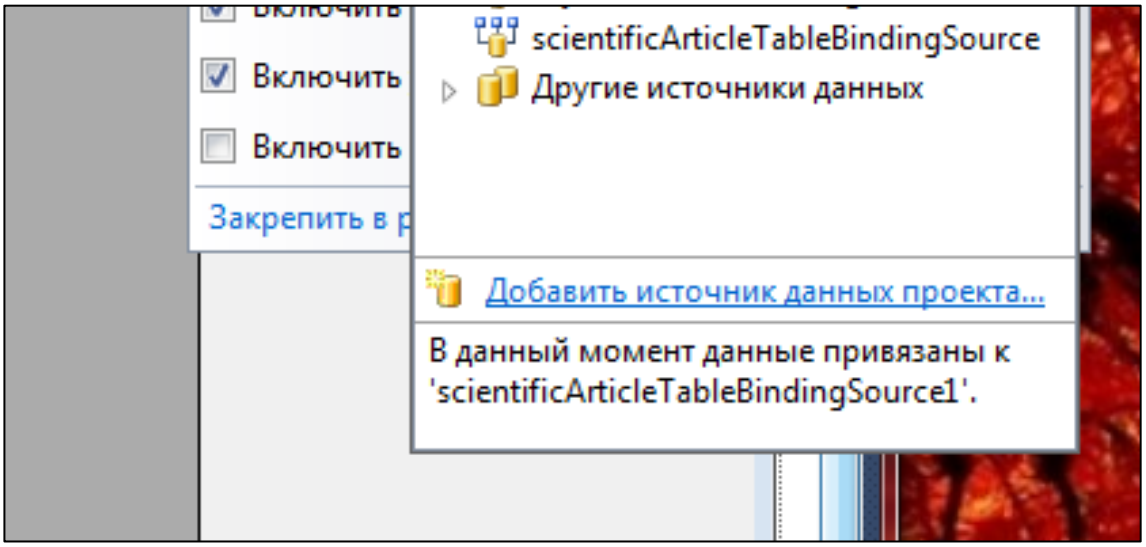


После заполнения сохраняем таблицу.

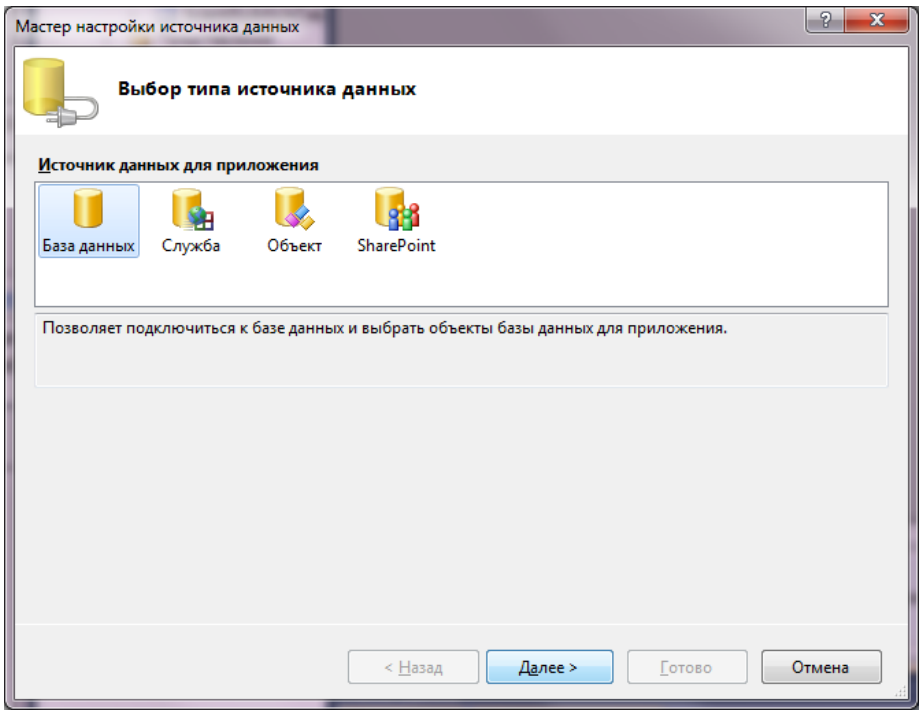
Переходим в Form1.cs. Добавляем в форму таблицу и выбираем для таблицы источник данных.



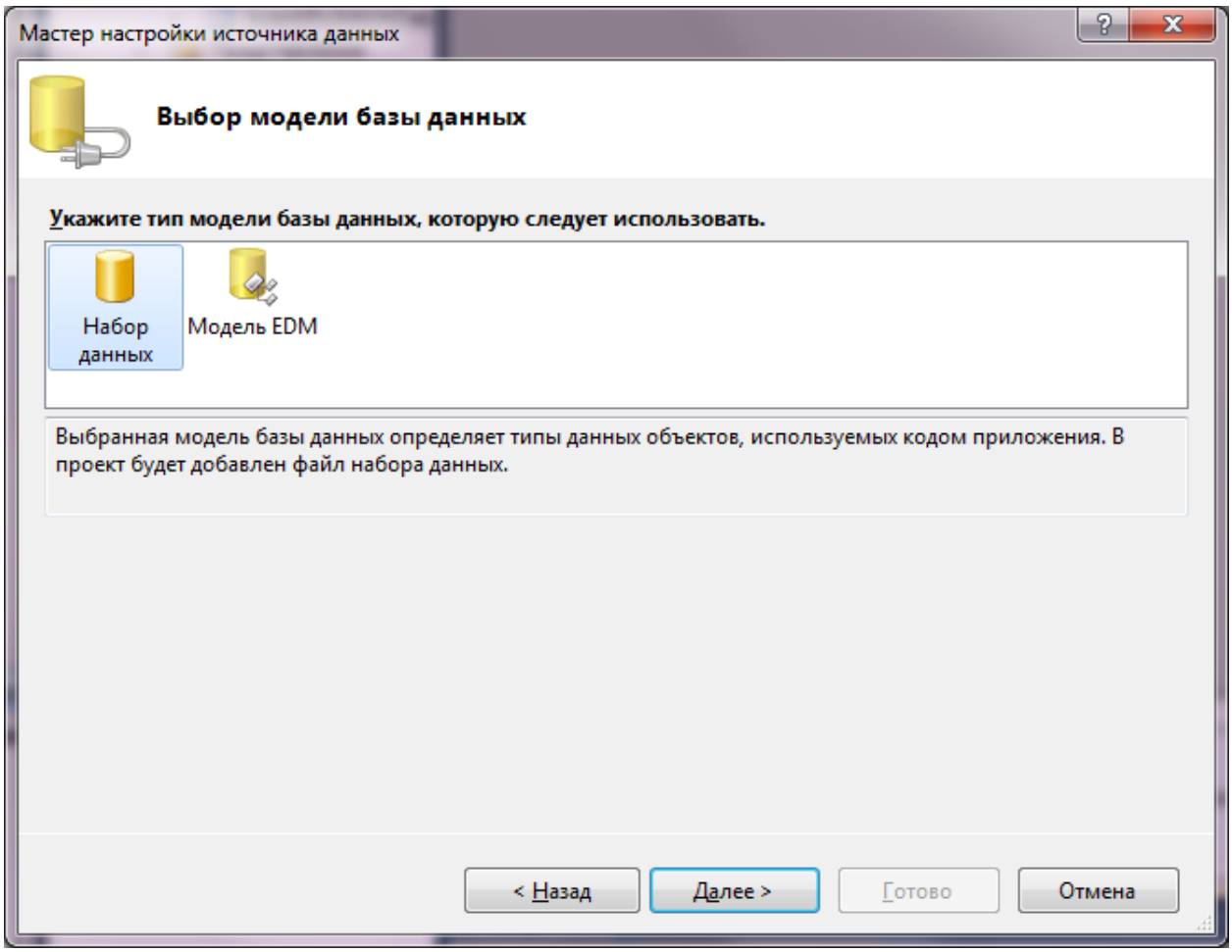
Добавить источник данных проекта.



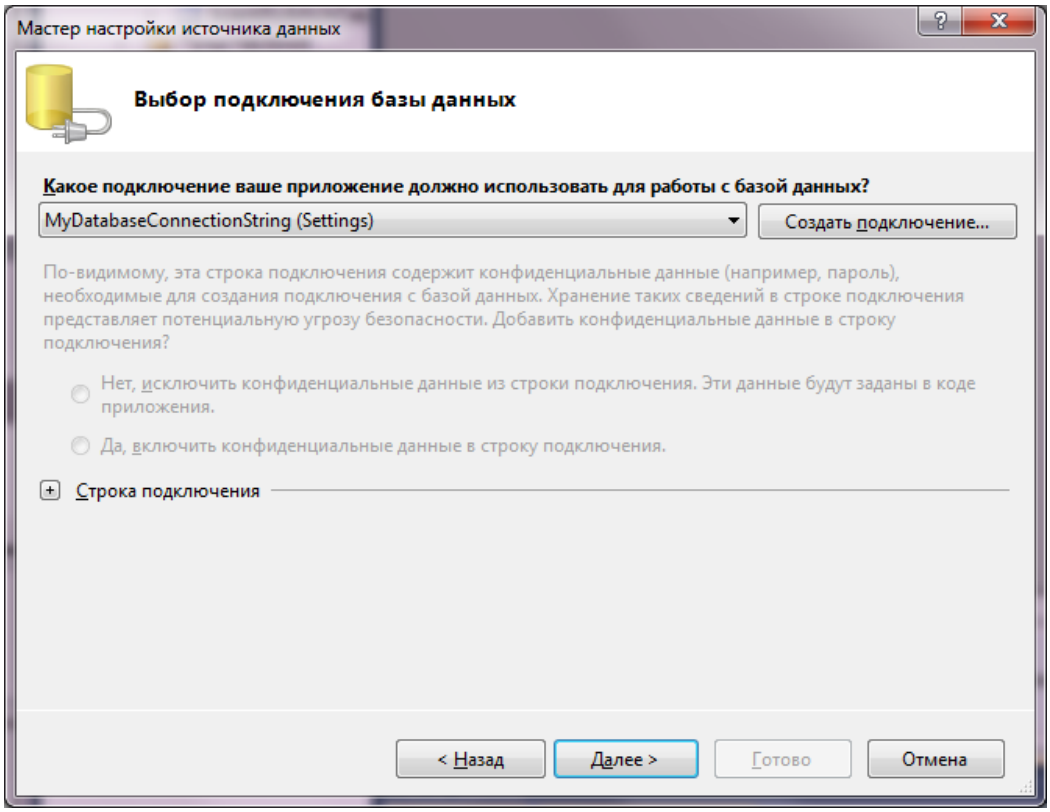
Нажимаем далее.



Нажимаем далее.



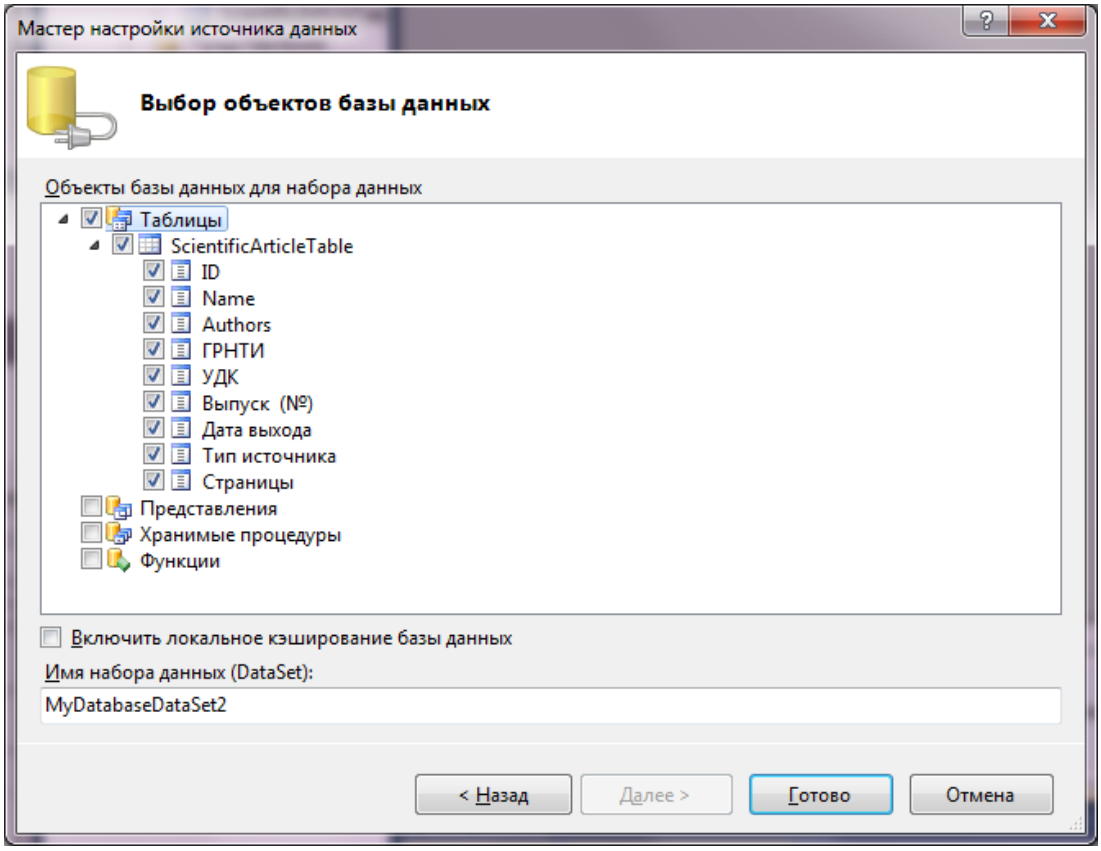
Выбираем подключение и нажимаем далее.



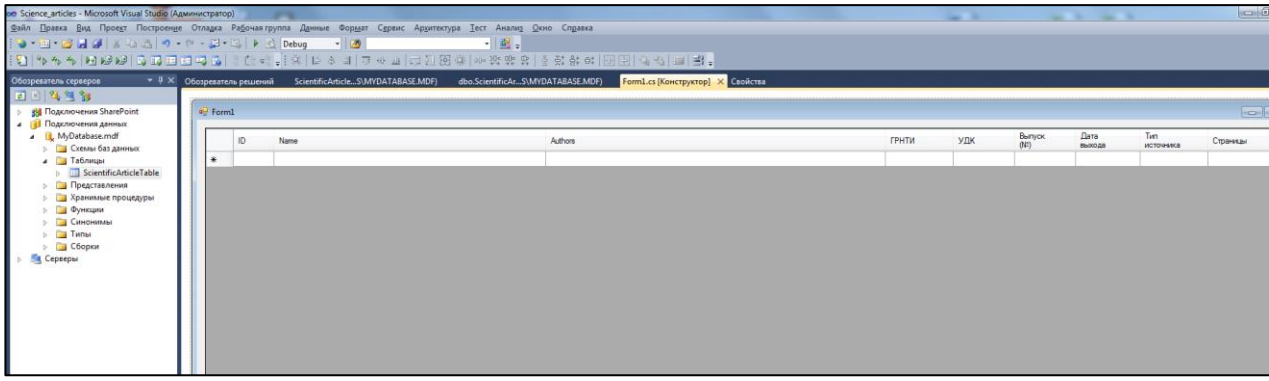
Выбираем объекты базы данных, которые мы хотим подключить,

отмечая их галочкой.

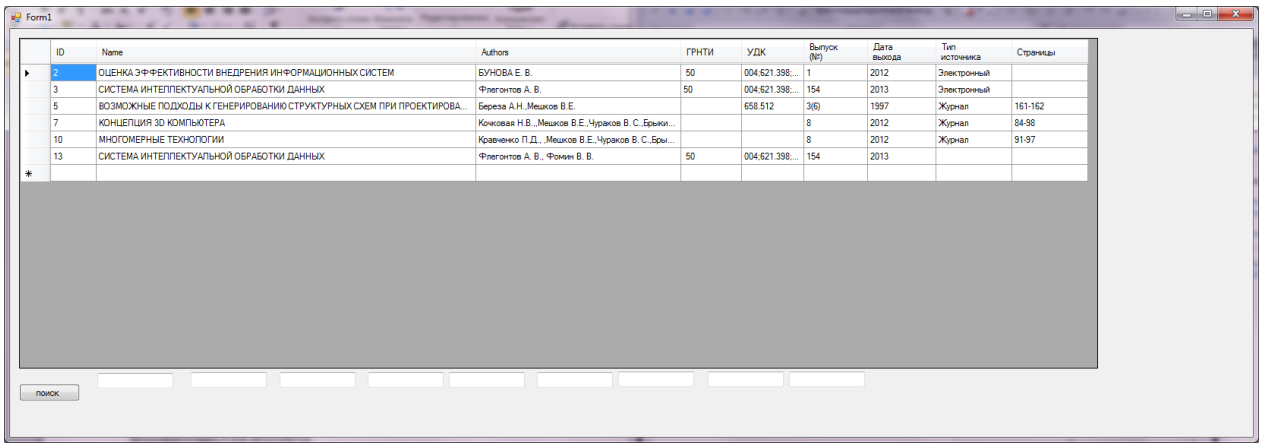
Нажимаем готово.



Наша база появляется в таблицы формы проекта.



Запускаем отладку и таблица заполняется записями из нашей базы.


Form1
ID
Name
Authors
ГРНТИ
УДК
Выпуск (№)
Дата выхода
Тип источника
Страницы
2
ОЦЕНКА ЭФФЕКТИВНОСТИ ВНЕДРЕНИЯ ИНФОРМАЦИОННЫХ СИСТЕМ
БУНОВА Е. В.
50
004.621.398;...
1
2012
Электронный
3
СИСТЕМА ИНТЕЛЛЕКТУАЛЬНОЙ ОБРАБОТКИ ДАННЫХ
Флегонтов А. В.
50
004.621.398;...
154
2013
Электронный
5
ВОЗМОЖНЫЕ ПОДХОДЫ К ГЕНЕРИРОВАНИЮ СТРУКТУРНЫХ СХЕМ ПРИ ПРОЕКТИРОВА...
Береза А.Н.,Мешков В.Е.
658.512
3(6)
1997
Журнал
161-162
7
КОНЦЕПЦИЯ 3D КОМПЬЮТЕРА
Кочковая Н.В.,,Мешков В.Е.,Чураков В. С.,Брыки...
8
2012
Журнал
84-98
10
МНОГОМЕРНЫЕ ТЕХНОЛОГИИ
Кравченко П.Д., ,Мешков В.Е.,Чураков В. С.,Бры...
8
2012
Журнал
91-97
13
СИСТЕМА ИНТЕЛЛЕКТУАЛЬНОЙ ОБРАБОТКИ ДАННЫХ
Флегонтов А. В., Фомин В. В.
50
004.621.398;...
154
2013
поиск

4 ЗАДАНИЯ НА ВЫПОЛНЕНИЕ КУРСОВОГО ПРОЕКТА

Программы разрабатываются студентом на языке высокого уровня программирования C#.

Среда разработки VS MS 2010 и выше.

Технология программирования – Windows Form.

Источник данных – база данных (СУБД MS SQL).

Формы титульного листа, листа задания и ведомости проекта даны в Приложении.

Пример выполнения курсового проекта (пояснительная записка) дан в Приложении.

Описание подробного алгоритма решения задачи приводится с учетом требований ЕСПД, приведенных в Приложении.

Тема курсового проекта утверждается приказом Директора ИТ (ф) ДГТУ.

Примерная тематика курсовых проектов:

1. Разработать программу обработки списка научных статей
2. Разработать программу обработки списка тематических сайтов в области ИТ-технологий
3. Разработать программу обработки списка тем дипломных проектов
4. Разработать программу обработки списка баз практики студентов
5. Разработать программу обработки списка онлайн ресурсов
6. Разработать программу обработки списка тем курсовых проектов по дисциплинам
7. Разработать программу обработки списка программного обеспечения
8. Разработать программу обработки списка учебно-методической литературы

РЕКОМЕНДУЕМАЯ ЛИТЕРАТУРА

Основная литература

1 Борисенко В.В. Основы программирования М.: Интернет-Университет Информационных Технологий (ИНТУИТ) 2016 http://www.iprbookshop.ru/52206 [Электронный ресурс]

2 Ю.Ю. Громов, О.Г. Иванова, М.П. Беляев, Ю.В. Минин Технология программирования Тамбов : Издательство ФГБОУ ВПО «ТГТУ» 2013 http://biblioclub.ru/index.php?page=book&id=277802 [Электронный ресурс]

Дополнительная литература

1 Благодатских, В. А. Стандартизация разработки программных средств М.: Финансы и статистика учеб. пособ. для вузов 2003

2 Грибанов В.П. Высокоуровневые методы информатики и программирования М.: Евразийский открытый институт учебно-практическое пособие 2011 http://www.iprbookshop.ru/14636 [Электронный ресурс]

3 Давыдов, В. Г. Технологии программирования C++ СПб.: БХВ-Петербург учеб. пособ. для студентов вузов 2005

4 Зиборов, В. В. Visual Basic 2010 на примерах СПб. : БХВ-Петербург, 2010 [Текст]

Периодические издания.

1 Информационные технологии и вычислительные системы 2015

2 Научно-техническая информация (НТИ). Серия 2. Информационные процессы и системы 2015

ПРИЛОЖЕНИЕ А. Пример выполнения курсового проекта.

Содержание:

Для решения задачи по СУБД-Книги, реализация клиент-серверной архитектуры наиболее выгодно в плане хранения и обработки по атрибутам списка нашей базы. Так же можно выделить простоту использования такой базы данных, а так же малыми системными требованиями к клиентской рабочей станции.

1. Реализация программной подсистемы. Концептуальное проектирование

1.1. Обзор и анализ предметной области

В настоящее время электронные библиотеки набирают все больше популярности. Причиной тому является возможность получить доступ к абсолютно любой интересующей вас книге. А так же возможность открыть данную книгу на мобильных устройствах.

В настоящее время у большинства пользователей электронных библиотек, имеются свои маленькие списки книг интересные или необходимые для пользователя.

Для обеспечения оперативности ведения и редактирования информации о имеющихся книг - необходима автоматизированная система, основанная на современной базе данных. Использование базы данных и автоматизированной системы для работы с базой данных существенно упростит задачу управления списком книг, а главное вся информация, касающаяся книг пользователя, будут храниться в одном месте

1.2. Определение обобщённой структуры данных

Основными конструктивными элементами инфологических моделей являются сущности, связи между ними и их свойства (атрибуты).

Сущность - любой объект, событие или концепция, имеющие существенное значение для предметной области, и информация о которых должна сохраняться. В нашей разрабатываемой базе данных присутствует только одна сущность: Книги.

У каждой сущности имеются свои свойства (атрибуты), которые важны для данной сущности.

Атрибут - любая характеристика сущности, значимая для рассматриваемой предметной области. Атрибут предназначен для

квалификации, идентификации, классификации, количественной характеристики или выражения состояния сущности.

Для того что бы инфицировать сущность нам необходимо иметь атрибут или группу атрибутов – первичный ключ.

Атрибуты сущности приведены в таблице 1:

Сущность	Атрибуты
Книги	Автор
	Название
	Жанр
	Перевод
	Издательство
	Язык оригинала
	Год издания
	Город издания
	ISBN
	Тип книги
	Размер книги
	Объем книги

Таблица 1. Атрибуты сущности

Наша база данных должна включать более полный список всех атрибутов нашей сущности. Это необходимо для того что бы мы могли более точно редактировать наш список. В предоставленной таблице 2, имеется расшифровка всех атрибутов которые мы используем:

Атрибут	Описание
Автор	Ф.И.О. Автора
Название	Полное название книги

Жанр	Жанр книги
Перевод	Ф.И.О. Кто переводил
Издательство	Полное название издательства
Язык оригинала	Язык на котором написана книга
Год издания	Год когда книга была издана
Город издания	Город в котором издали книгу
ISBN	Уникальный номер книжного издания
Тип книги	Электронная книга или в печатном виде
Размер книги	Размер электронной книги в мб.
Объем книги	Количество страниц в книге.

Таблица 2. Расшифровка всех атрибутов

1.3. Выбор концептуальной модели проектирования

Концептуальное проектирование технических систем — начальная стадия проектирования, на которой принимаются определяющие последующий облик решения, и проводится исследование и согласование параметров созданных технических решений с возможной их организацией. Таким образом, проектирование на концептуальном уровне — на уровне смысла или содержания понятия систем.

Основными задачами концептуального проектирования являются определение предметной области системы и формирование взгляда на ПО с позиций сообщества будущих пользователей.

В настоящее время наиболее распространены две концептуальные модели:

1) Каскадная модель

Основной характеристикой каскадной (водопадной) модели является разбиение всей разработки на этапы, причем переход с одного этапа на следующий происходит только после того, как будет полностью завершена работа над текущей. Каждый этап завершается выпуском полного комплекта документации, достаточной для того, чтобы разработка могла быть продолжена командой специалистов на следующем этапе.

Положительные стороны применения каскадного подхода заключаются в следующем:

- на каждом этапе формируется законченный набор проектной документации, отвечающий критериям полноты и согласованности;
- выполняемые в логичной последовательности этапы работ позволяют планировать сроки завершения всех работ и соответствующие затраты.

Пример каскадной модели можно посмотреть на рисунке 1:

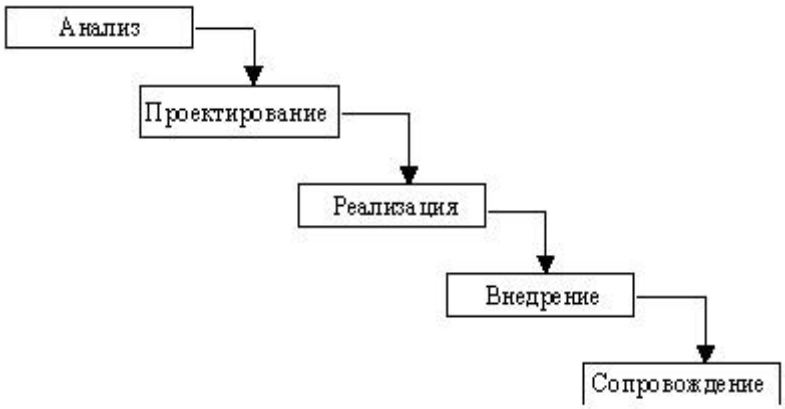


Рисунок 1- каскадная модель

2) Спиральная модель

Классический пример применения эволюционной стратегии конструирования. Модель (автор Б. Боэм, 1988) базируется на лучших свойствах классического жизненного цикла и макетирования, к которым

добавляется новый элемент – анализ риска, отсутствующий в этих парадигмах. Модель определяет четыре действия, представляемые четырьмя квадрантами спирали (Рисунок 2):

Рисунок 2- Спиральная модель

1. Планирование – определение целей, вариантов и ограничений.
2. Анализ риска – анализ вариантов и распознавание/выбор риска.
3. Конструирование – разработка продукта следующего уровня.
4. Оценивание – оценка заказчиком текущих результатов конструирования.

Для более оптимального моделирования нашей базы данных необходимо применить спиральное моделирование. Вид нашей модели можно увидеть на рисунке 3:

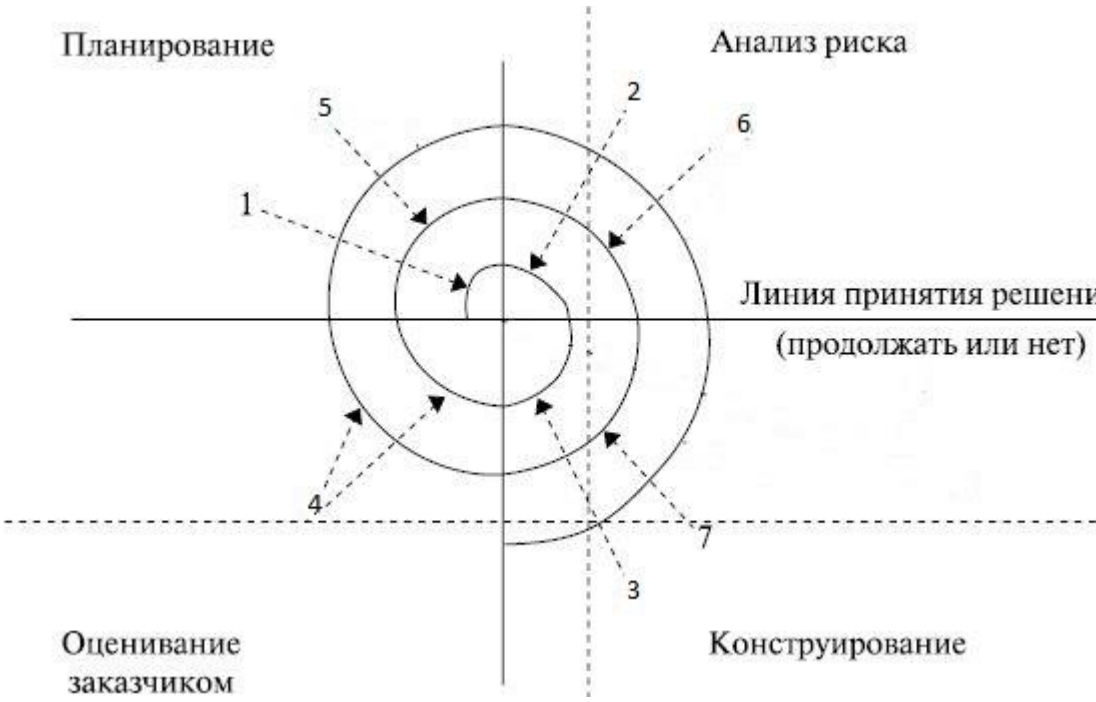


Рисунок 3- Спиральная модель (уточнение)

1 – Выбор архитектуры для решения данной задачи

2 - Обдумывание возможных недостатков архитектуры клиент-сервер

3 – Создание и подключение базы данных SQL

4 – Отладка программы и проверка ее работоспособности

5 – Выбор оптимальной для пользователя конфигурации интерфейса

6 – Обдумывание выбора той или иной конфигурации интерфейса

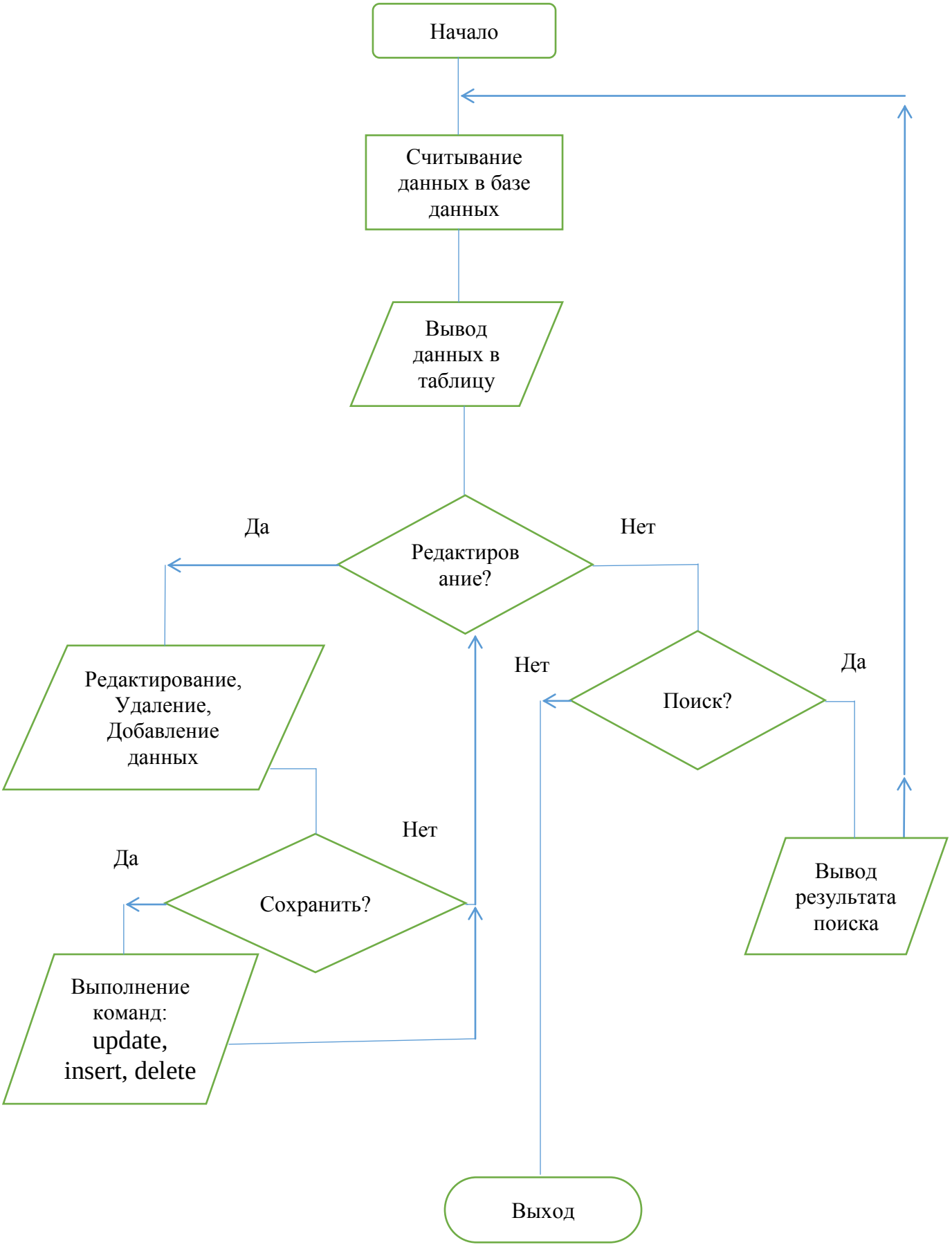
7 – Создание интерфейса

Таким образом мы видим все этапы построения нашей программы.

2. Реализация программной подсистемы

2.1. Обобщенный алгоритм функционирования

Для более наглядного представления алгоритма функционирования составим блок-схему .



2.2. Проектирование базы данных SQL

Для реализации архитектуры клиент-сервер нам необходимо подключить к нашему проекту базу данных SQL. Это одна из первых задач при построении нашего проекта.

При проектировании базы данных нам потребуются атрибуты нашей сущности. Так же каждому атрибуту необходимо указать тип данных которые мы будем использовать. В нашем случае мы выбираем тип данных nvarchar(50) – символьные данные, максимальная длина 50 символов.

Рассмотрим подключение базы данных поподробнее:

1) Создаем новый проект Windows Form, заходим во вкладку “Проект –Добавить новый элемент” , нам откроется окно выбора элементов (Рисунок 4):

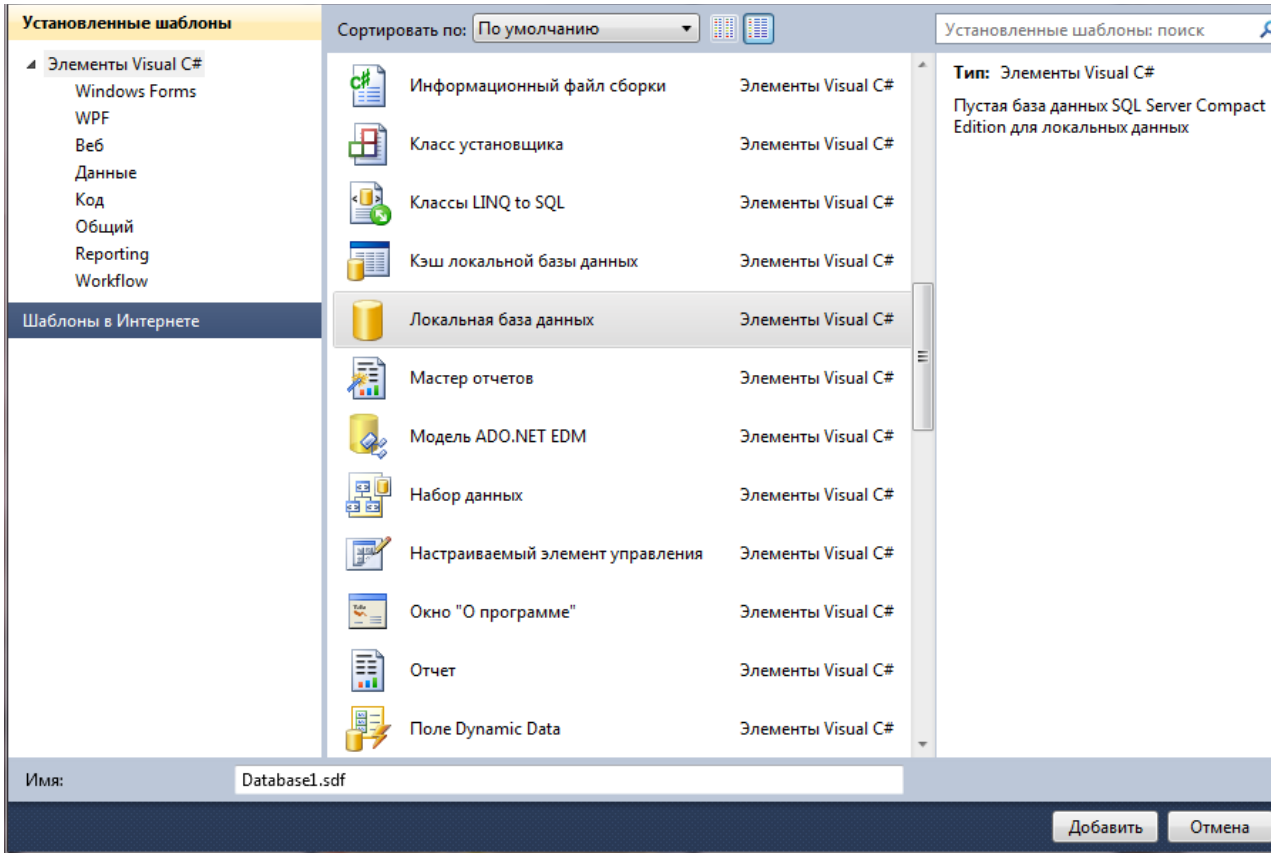


Рисунок 4

2) В списке мы находим «База данных, основанная на службах » и нажимаем кнопку добавить. Нам откроется «Мастер настройки источника данных (Рисунок 5):

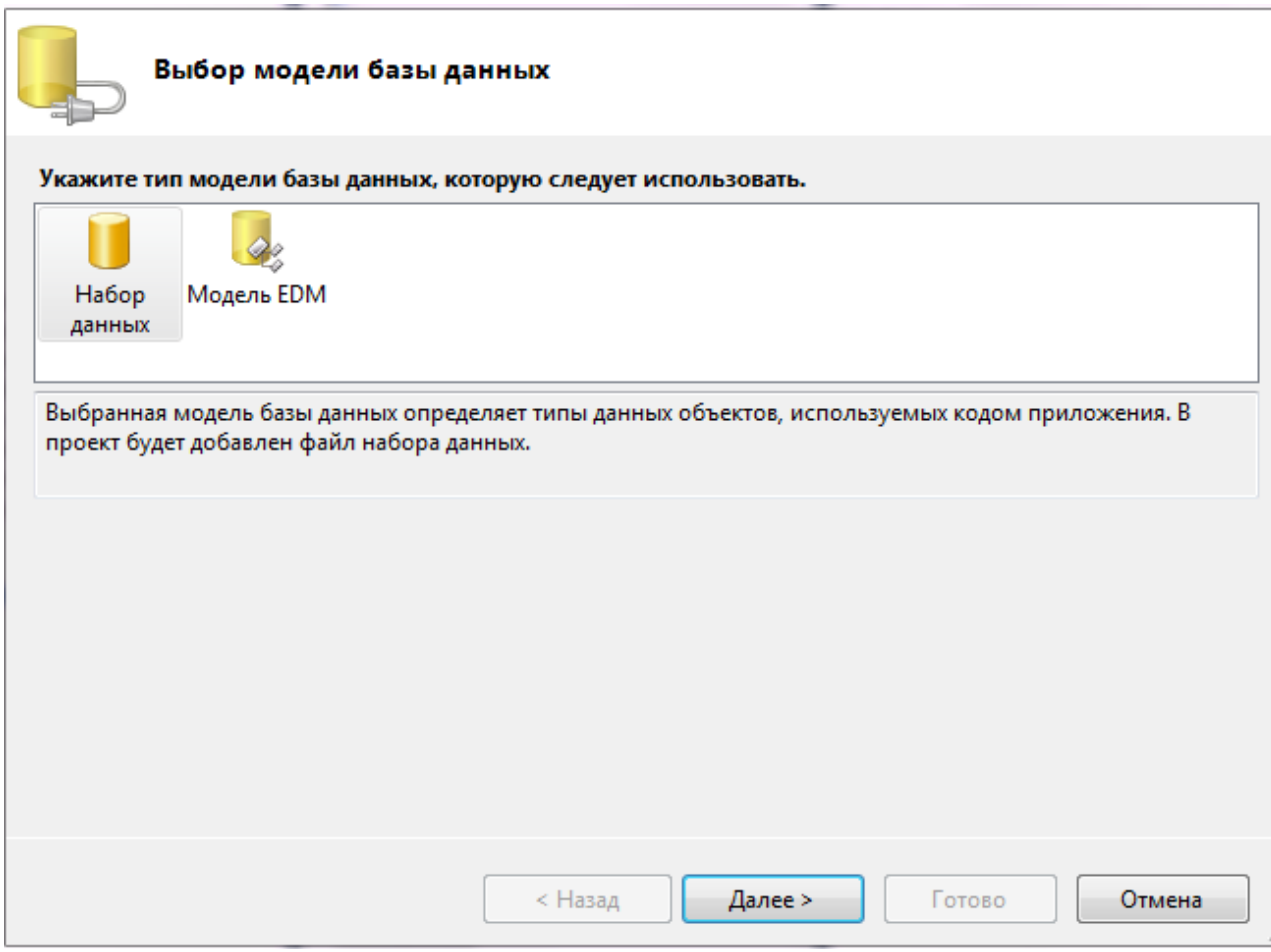


Рисунок 5

3) В данном окне выбираем набор данных и жмем далее. После чего нажимаем готово. Как мы можем увидеть в окне обозреватель серверов наша база успешно подключена (Рисунок 6):

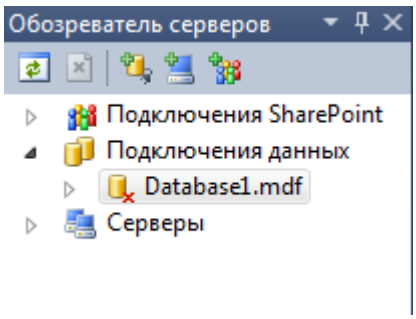


Рисунок 6

4) Сейчас наша база данных пуста, добавим в нее новую таблицу. В окне обозреватель серверов найдем нашу базу данных. Нажмем на строку таблицы правой кнопкой мыши и выберем добавить новую таблицу. После чего нам откроется вкладка для добавления столбцов в таблицу (Рисунок 7):

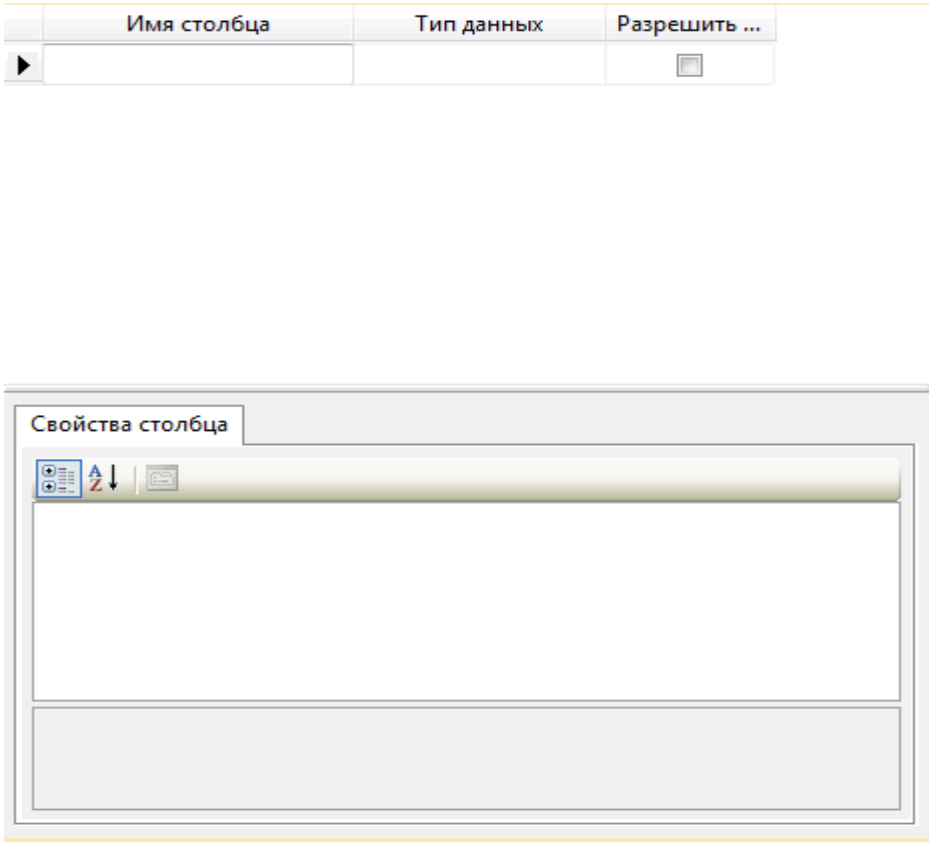
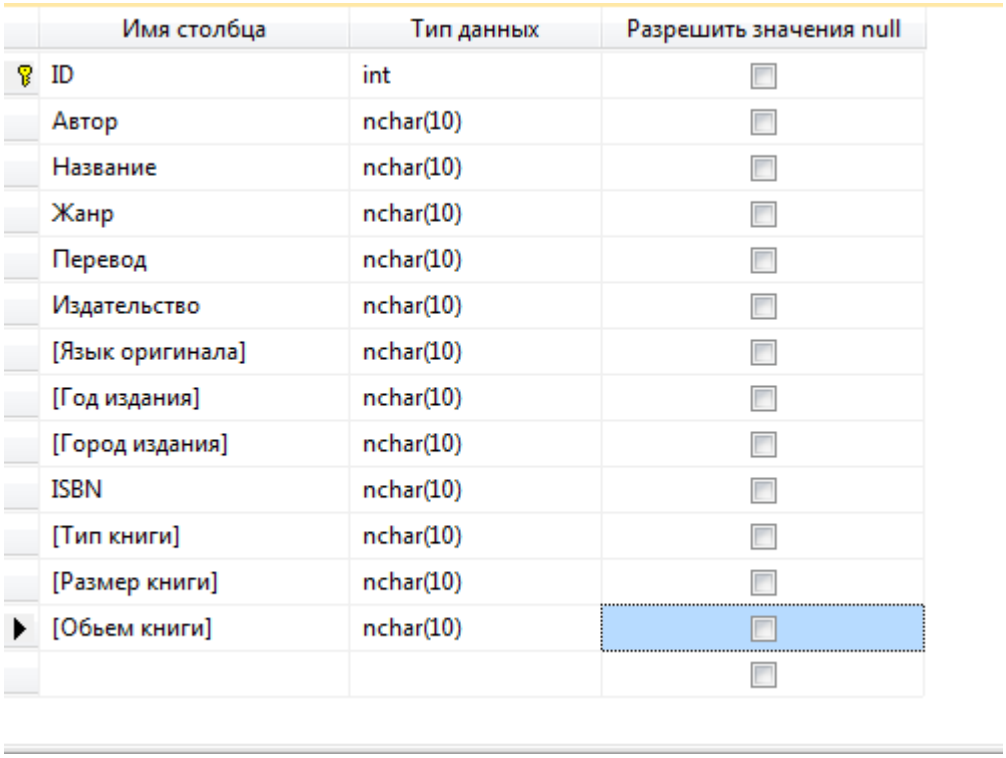


Рисунок 7

5) Теперь мы должны ввести: Имя столбца – это наши атрибуты, тип данных – мы выбираем nvarchar(50), о после убрать галочку «Разрешить значения null» - тем самым наши строки не должны иметь пустых ячеек. Готовая таблица выглядит так (Рисунок 8):



Имя столбца	Тип данных	Разрешить значения null
ID	int	☐
Автор	nchar(10)	☐
Название	nchar(10)	☐
Жанр	nchar(10)	☐
Перевод	nchar(10)	☐
Издательство	nchar(10)	☐
[Язык оригинала]	nchar(10)	☐
[Год издания]	nchar(10)	☐
[Город издания]	nchar(10)	☐
ISBN	nchar(10)	☐
[Тип книги]	nchar(10)	☐
[Размер книги]	nchar(10)	☐
[Обьем книги]	nchar(10)	☐
		☐

Рисунок 8

6) Нажимаем сохранить и вводим имя нашей таблицы. Теперь на вкладке Обозреватель серверов мы можем увидеть что наша таблица создана (Рисунок 9):

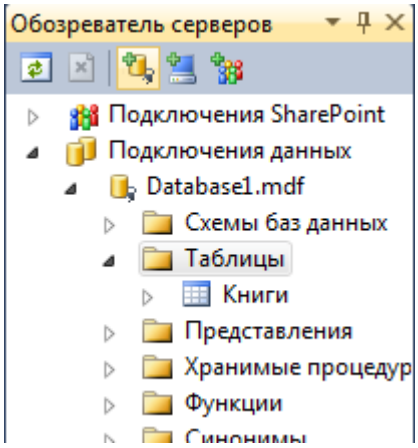


Рисунок 9

7) Теперь нам необходимо вставить таблицу в наш проект. Для этого переходим в «Источники данных» и нажимаем «Настроить источники данных». Там мы отмечаем нашу таблицу (Рисунок 10):

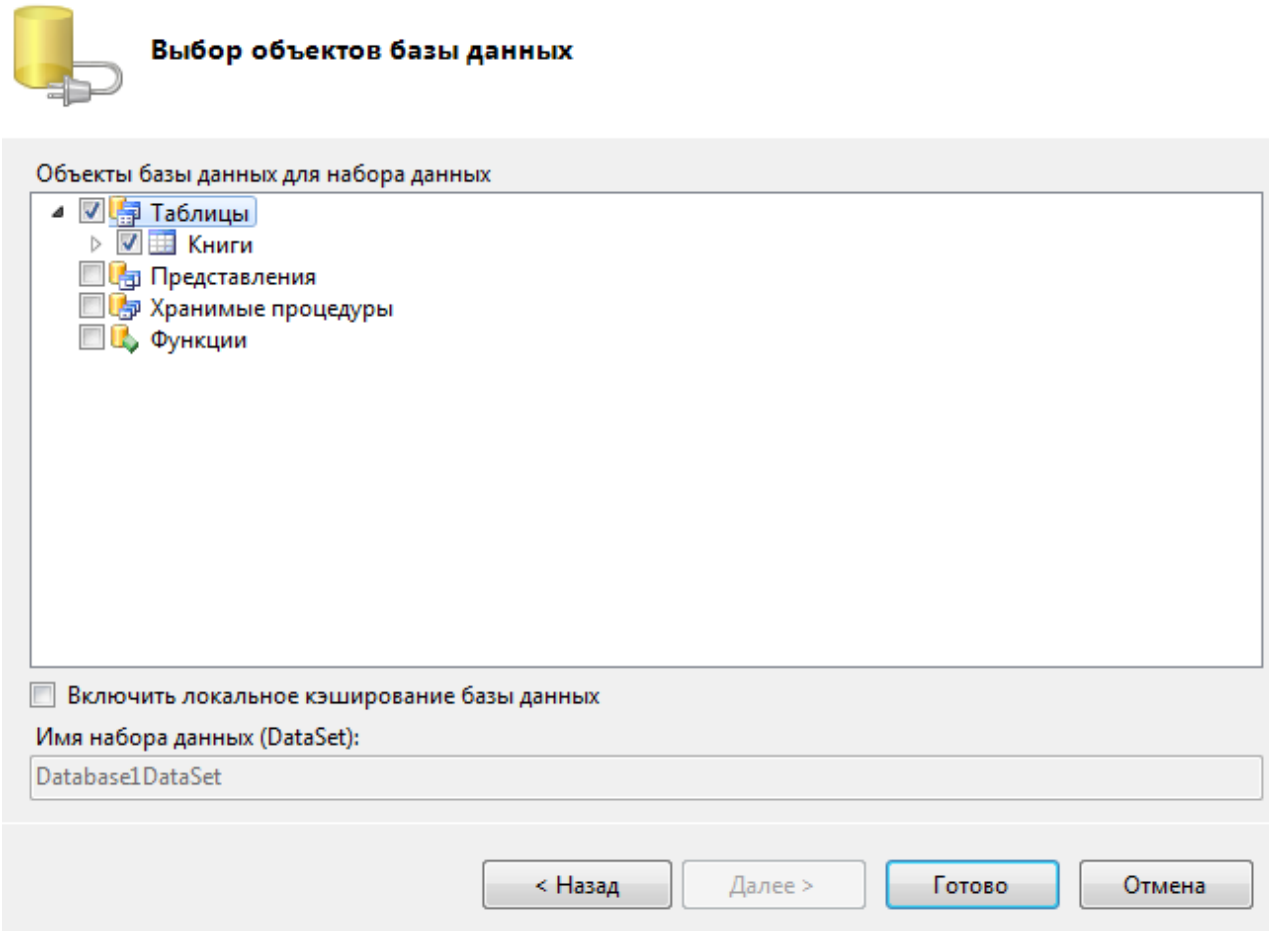


Рисунок 10

8) Нажимаем готово. Теперь перетаскиваем нашу таблицу в форму и настраиваем ее форму (Рисунок 11). Теперь мы можем переходить к созданию и реализации интерфейса.

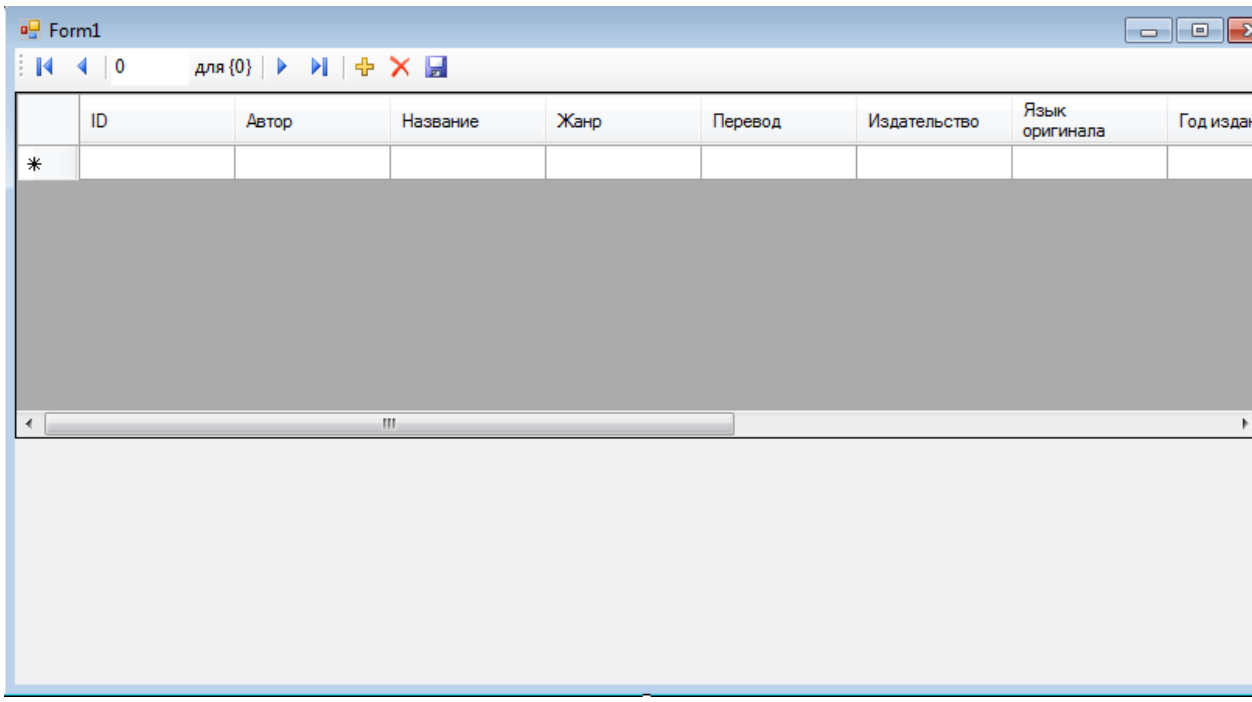


Рисунок 11

2.3. Реализация интерфейса

При реализации интерфейса основное внимание было обращено на основные функции нашей программы. Основной функцией нашей программы является поиск по нашему списку книг для более быстрого доступа к той или иной книге.

Для построения интерфейса под нашу основную функцию мы будем использовать такие элементы как: lebel – для подписи наших полей ввода, Textbox – для ввода наших критериев поиска, Button – для активации функции поиска, отчистки всех полей ввода и сброса поиска.

В конечном итоге мы получили интерфейс нашей программы (Рисунок 12):

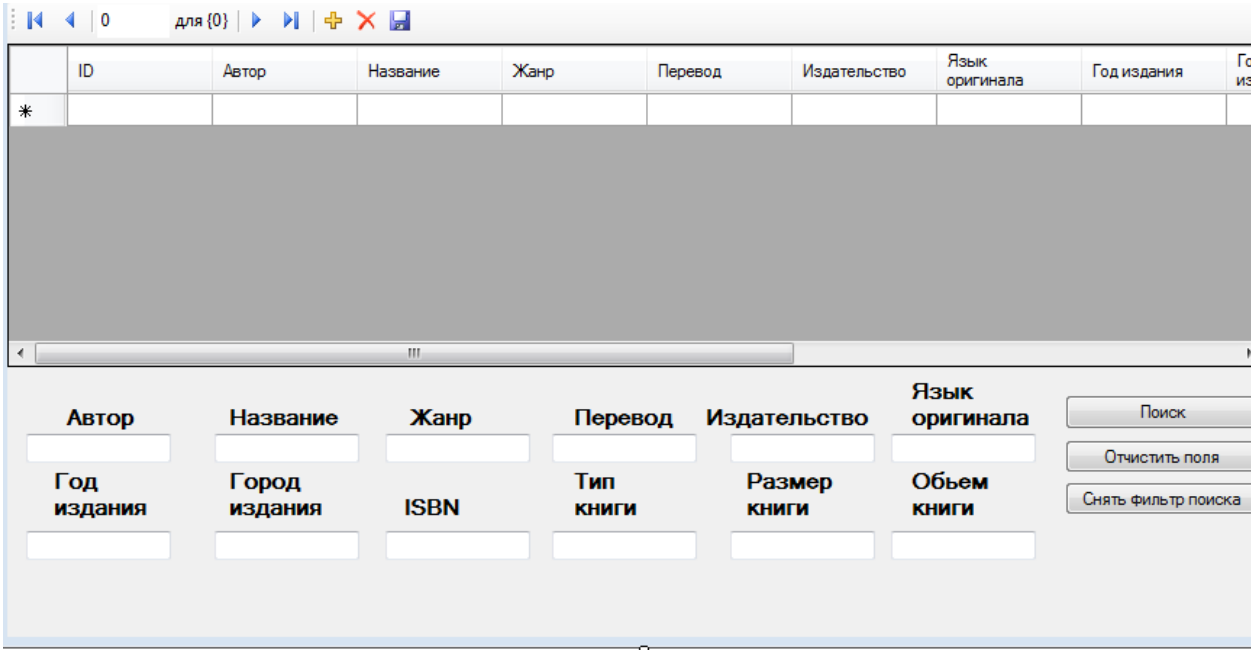


Рисунок 12

После создания интерфейса мы можем перейти к реализации программного кода.

2.4. Реализация программного кода

Для полной работы программы нам необходимо реализовать программный код под нашу основную задачу. Нашей основной задачей является поиск по всему списку книг. Данный поиск мы будем выполнять по нажатию на соответствующую кнопку. Поиск будет выполнятся по таблице в которой будет отображаться весь список. Критерии поиска задаются самим пользователем в соответствующем поле. Сам поиск мы реализуем таким образом:

```
for (int i = 0; i < книгиDataGridView.RowCount - 1; i++)
                {
                    if
(книгиDataGridView.Rows[i].Cells[0].Value.ToString().Contains(textBox1.Text) &&
книгиDataGridView.Rows[i].Cells[1].Value.ToString().Contains(textBox2.Text) &&
книгиDataGridView.Rows[i].Cells[2].Value.ToString().Contains(textBox3.Text) &&
книгиDataGridView.Rows[i].Cells[3].Value.ToString().Contains(textBox4.Text) &&
книгиDataGridView.Rows[i].Cells[4].Value.ToString().Contains(textBox5.Text) &&
книгиDataGridView.Rows[i].Cells[5].Value.ToString().Contains(textBox6.Text) &&
книгиDataGridView.Rows[i].Cells[6].Value.ToString().Contains(textBox7.Text) &&
```

```
книгиDataGridView.Rows[i].Cells[7].Value.ToString().Contains(textBox8.Text) &&
книгиDataGridView.Rows[i].Cells[8].Value.ToString().Contains(textBox9.Text) &&
книгиDataGridView.Rows[i].Cells[9].Value.ToString().Contains(textBox10.Text) &&
книгиDataGridView.Rows[i].Cells[10].Value.ToString().Contains(textBox11.Text) &&
книгиDataGridView.Rows[i].Cells[11].Value.ToString().Contains(textBox12.Text))
                        {
                            книгиDataGridView.Rows[i].Visible = true;
                        }
                        else
                        {
                            книгиDataGridView.Rows[i].Visible = false;
                        }
                    }
```

Так же на данном этапе мы должны предусмотреть такие ситуации когда у нас поиск не дал результатов по тем критериям которые ввел пользователь, а так же те случаи когда не один из критериев поиска не введен. Данные ошибки мы реализуем в помощью данного кода:

```
    for (int i = 0; i < книгиDataGridView.RowCount - 1; i++)
                    {
                        if (книгиDataGridView.Rows[i].Visible == false)
                        {
                            n = 1;
                        }
                        else
                        {
                            n = 0;
                            break;
                        }
                    }
                    if (n == 1)
                    {
                        MessageBox.Show("По данному запросу ничего не найдено",
"Результат", MessageBoxButtons.OK, MessageBoxIcon.Information);
                    }
                }
                catch (NullReferenceException)
                {
                    MessageBox.Show("Введите хотя бы один критерий для поиска.\n",
"Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);
                    return;
                }
```

После того как мы реализовали поиск в нашей программе мы добавляем некоторые дополнительные функции которые упростят нам работу со списком книг. Первой функцией мы добавим возможность отчистки полей критериев. Это упростит нам задачу при работе с поиском. Для реализации мы добавили отдельную кнопку и реализовали в ней данный код:

```
textBox1.Clear();
textBox2.Clear();
textBox3.Clear();
textBox4.Clear();
textBox5.Clear();
textBox6.Clear();
textBox7.Clear();
textBox8.Clear();
textBox9.Clear();
textBox10.Clear();
textBox11.Clear();
textBox12.Clear();
```

Третьей же функцией будет сброс фильтра поиска. Плюс данной функции очевиден. В случае когда нам необходимо искать не одну книгу а разные мы просто сбрасываем фильтр и производим поиск далее. Данную функцию мы реализуем на новой кнопке с помощью данного кода:

```
for (int i = 0; i < table1DataGridView.RowCount - 1; i++)
        {
           table1DataGridView.Rows[i].Visible = true;
        }
```

Таким образом мы реализовали весь программный код основанный из основной функции программы.

2.5. Тестирование программы

После реализации программного кода нам необходимо произвести отладку и полное тестирование программы. Начинаем мы с самого простого и заканчиваем проверкой всех возможных недочетов и ситуаций.

1) Первым будет ввод списка книг. (Рисунок 13)

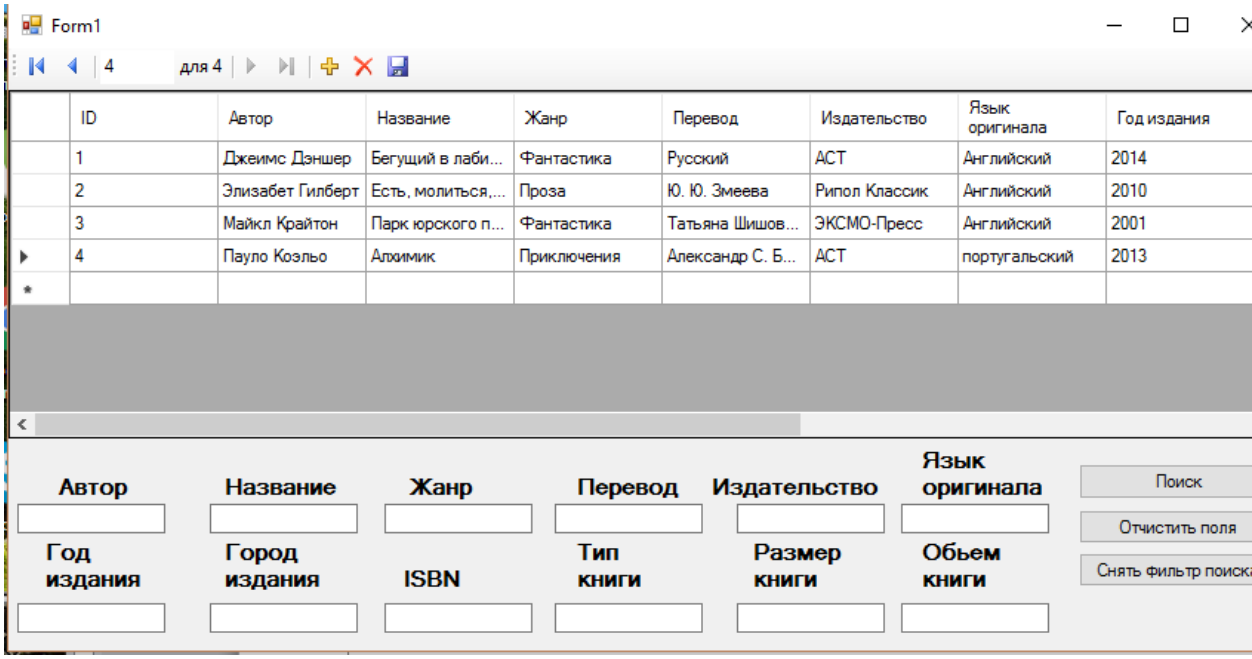


Рисунок 13

2) Далее пробуем сохранить внесенный список, после чего пробуем сортировать по одному из критериев (Рисунок 14):

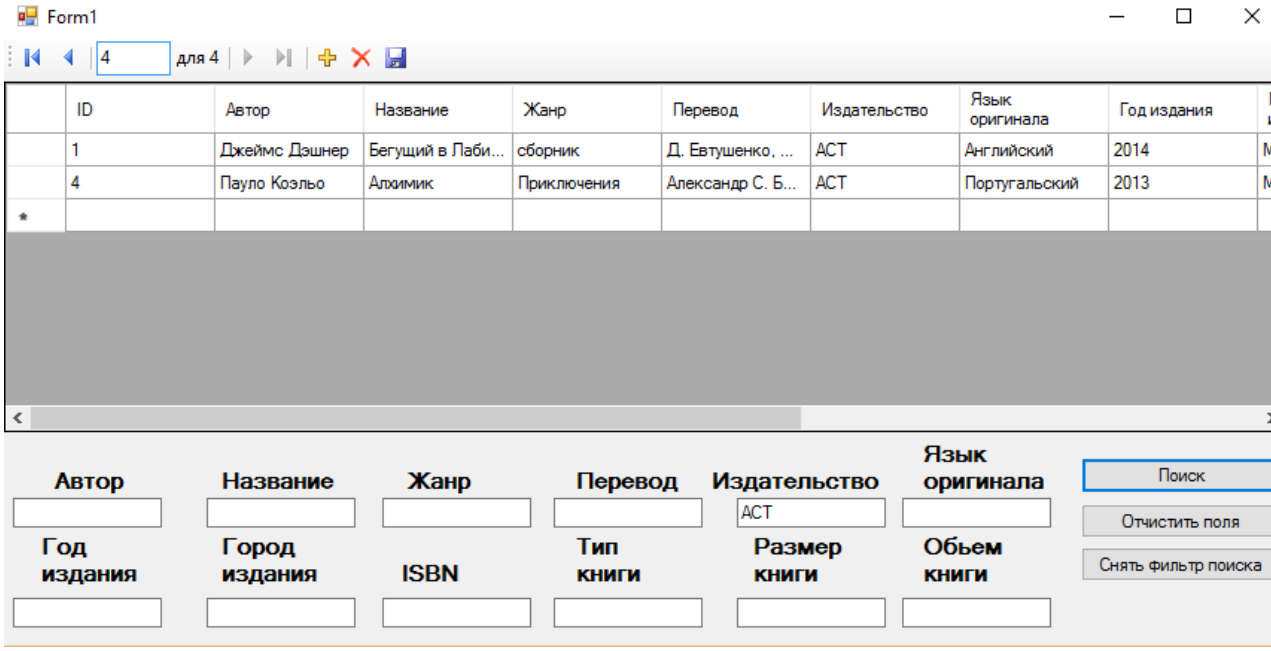


Рисунок 14

3) Так же попробуем отменить фильтрацию (Рисунок 15):

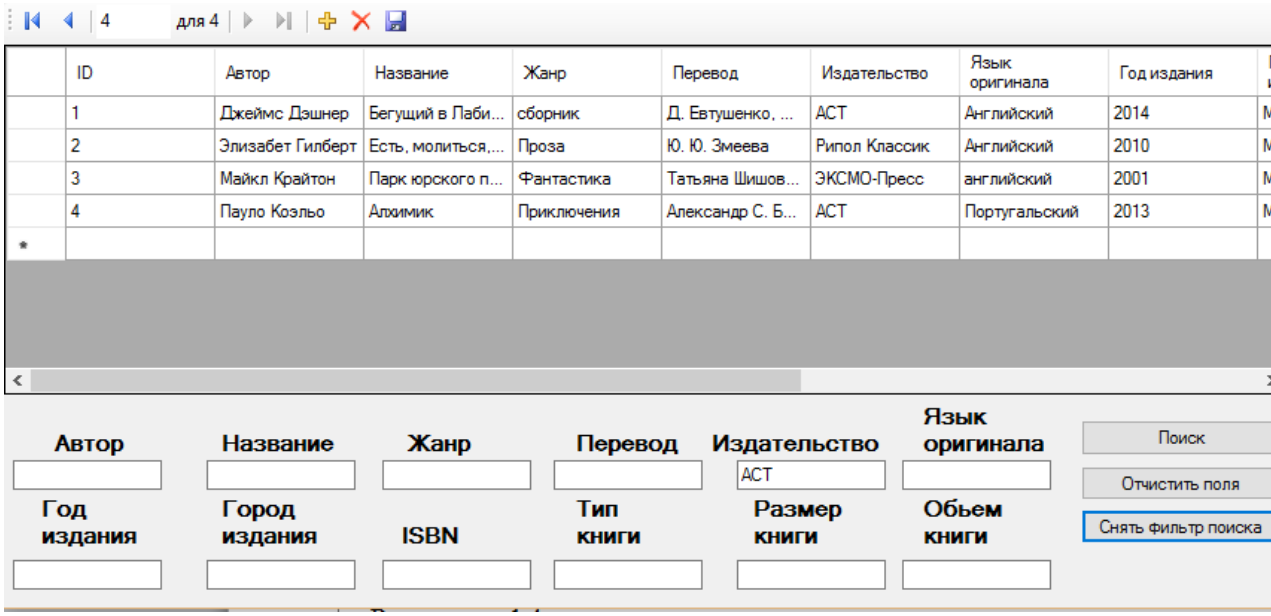


Рисунок 15

4) Далее попробуем отчистить поля (Рисунок 16, 17):

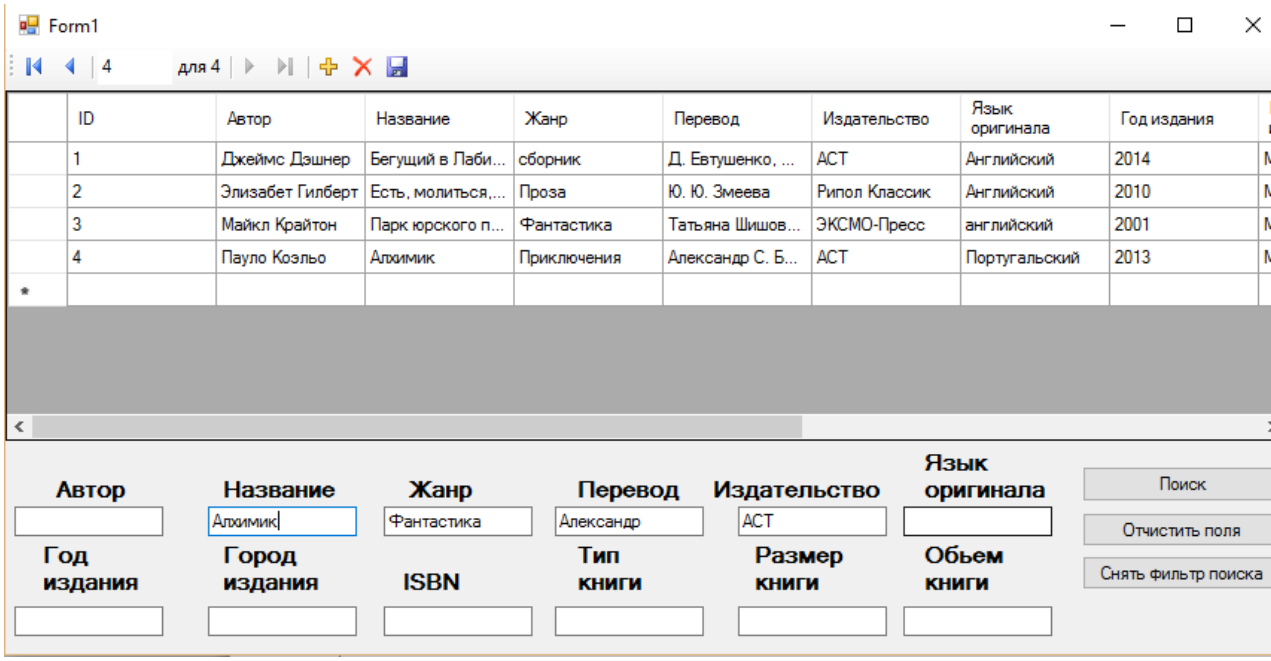


Рисунок 16

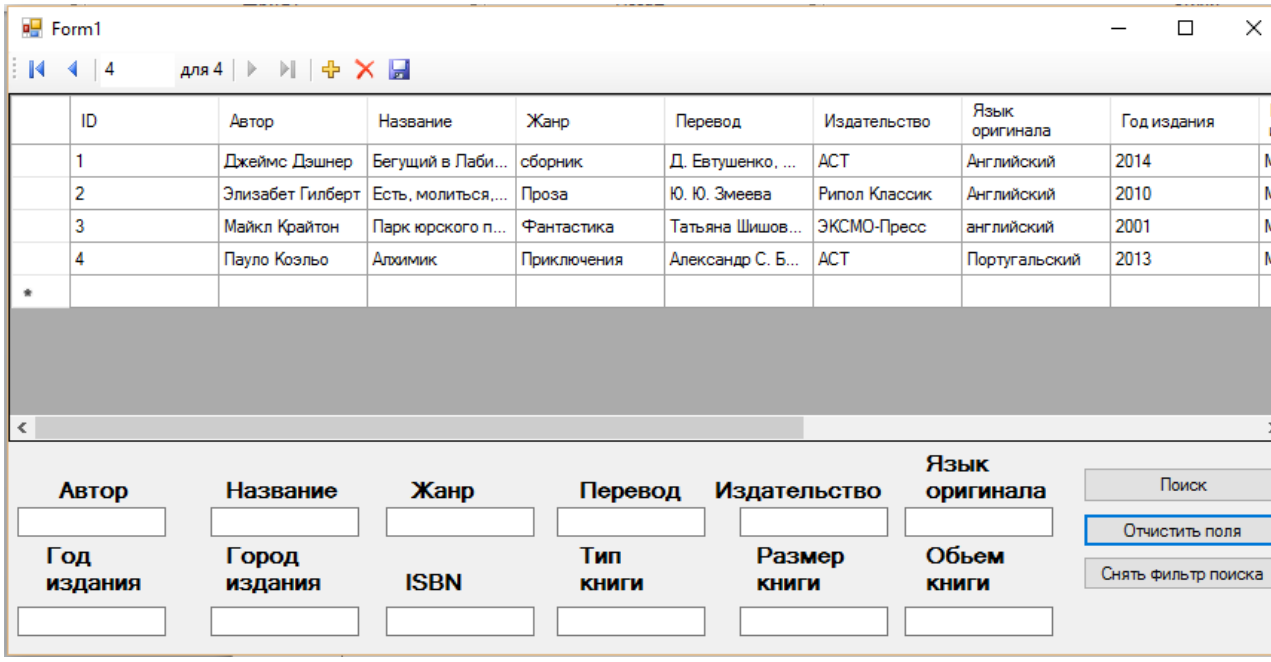
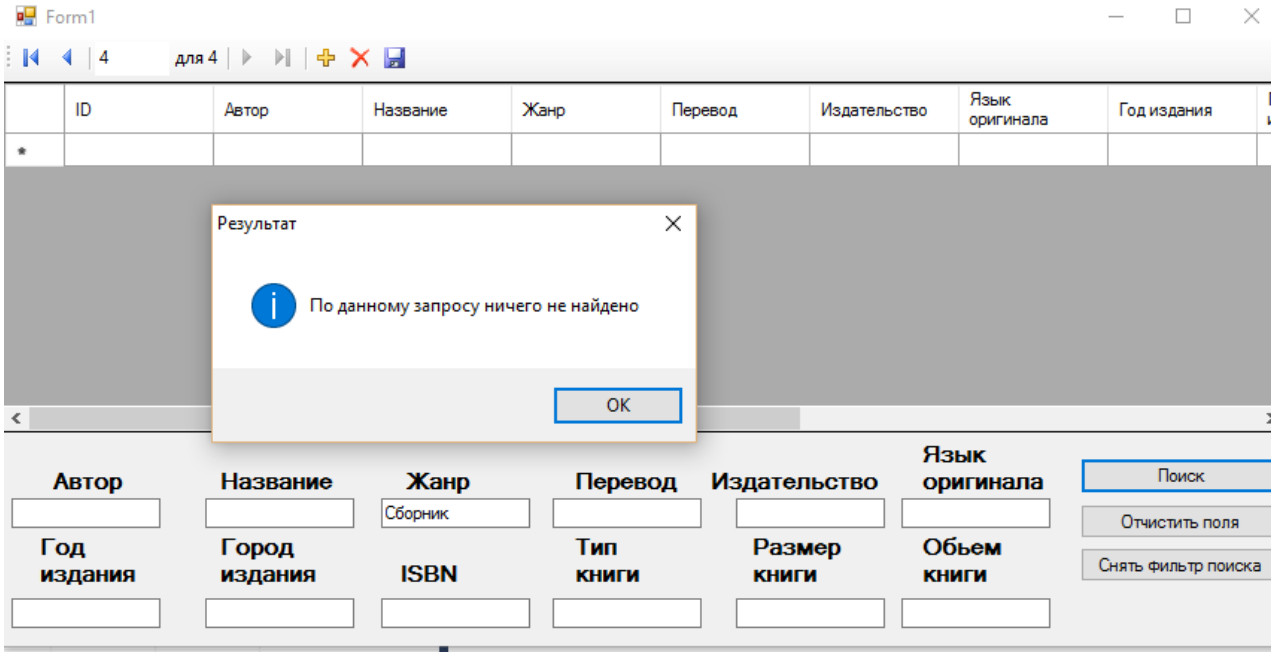


Рисунок 17

5) Попробуем задать неверный поисковой запрос (Рисунок 18):



Вывод и рекомендации:

При реализации курсового проекта мы добились оптимизации и удобство управления списком книг. Для достижения данной оптимизации и удобства мы рассмотрели алгоритм функционирования программы, основные SQL запросы а также реализация интерфейса и программного кода. Было проведено тестирование всех основных функций программы.

При дальнейшей разработке данной программы можно модифицировать элементы описания книги, при запросе той или иной книги выводить кратвое описание книги. Так же можно сразу же производить поиск в интернет библиотеках.

Источники:

Учебные пособия:

1. Трутнев Д. Р. Архитектуры информационных систем. Основы проектирования: Учебное пособие. – СПб.: НИУ ИТМО, 2012. – 66 с.

2. Проектирование архитектур информационных систем : методические указания к лабораторным работам/ сост. К. С. Беляев. – Ульяновск : УлГТУ, 2010. – 48 с.

Интернет ресурсы:

1. Электронная библиотека http://bookz.ru/

2. Электронная энциклопедия: http://gruzdoff.ru/wiki/%D0%A1%D0%BF%D0%B8%D1%80%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C

Книжное пособие:

1. Вендров А.М. Проектирование программного обеспечения экономических информационных систем. – М.: Финансы и статистика, 2000. – 352с

Приложение:

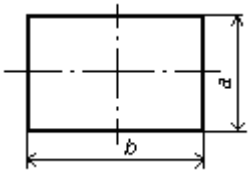
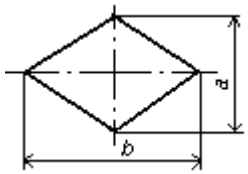
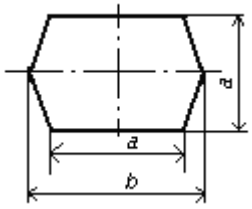
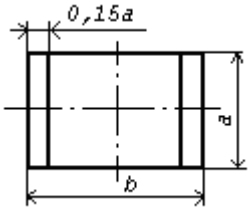
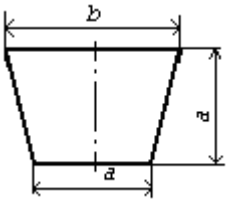
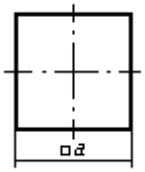
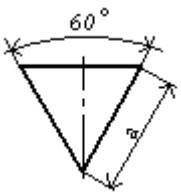
```
        private void button1_Click(object sender, EventArgs e)
                {
                    книгиDataGridView.CurrentCell = null;
                    try
                    {
                        for (int i = 0; i < книгиDataGridView.RowCount - 1; i++)
                        {
                            if
(книгиDataGridView.Rows[i].Cells[1].Value.ToString().Contains(textBox1.Text) &&

книгиDataGridView.Rows[i].Cells[2].Value.ToString().Contains(textBox2.Text) &&

книгиDataGridView.Rows[i].Cells[3].Value.ToString().Contains(textBox3.Text) &&

книгиDataGridView.Rows[i].Cells[4].Value.ToString().Contains(textBox4.Text) &&

книгиDataGridView.Rows[i].Cells[5].Value.ToString().Contains(textBox5.Text) &&

книгиDataGridView.Rows[i].Cells[6].Value.ToString().Contains(textBox6.Text) &&

книгиDataGridView.Rows[i].Cells[7].Value.ToString().Contains(textBox7.Text) &&

книгиDataGridView.Rows[i].Cells[8].Value.ToString().Contains(textBox8.Text) &&

книгиDataGridView.Rows[i].Cells[9].Value.ToString().Contains(textBox9.Text) &&

книгиDataGridView.Rows[i].Cells[10].Value.ToString().Contains(textBox10.Text) &&

книгиDataGridView.Rows[i].Cells[11].Value.ToString().Contains(textBox11.Text) &&

книгиDataGridView.Rows[i].Cells[12].Value.ToString().Contains(textBox12.Text))
                            {
                                книгиDataGridView.Rows[i].Visible = true;
                            }
                            else
                            {
                                книгиDataGridView.Rows[i].Visible = false;
                            }
                        }
                        for (int i = 0; i < книгиDataGridView.RowCount - 1; i++)
                        {
                            if (книгиDataGridView.Rows[i].Visible == false)
                            {
                                n = 1;
                            }
                            else
                            {
                                n = 0;
                                break;
                            }
                        }
                        if (n == 1)
                        {
                            MessageBox.Show("По данному запросу ничего не найдено",
"Результат", MessageBoxButtons.OK, MessageBoxIcon.Information);
                        }
                    }
                    catch (NullReferenceException)
                    {
                        MessageBox.Show("Введите хотя бы один критерий для поиска.\n",
"Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);
                        return;
                    }
```

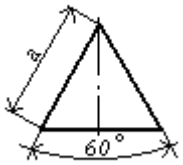
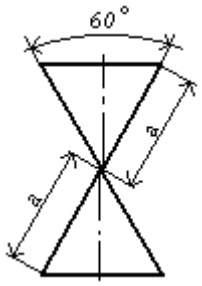
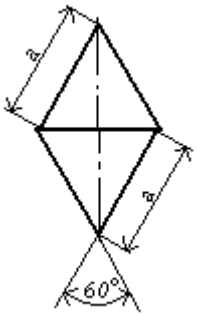
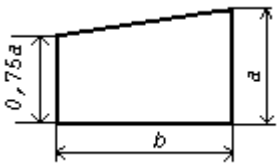
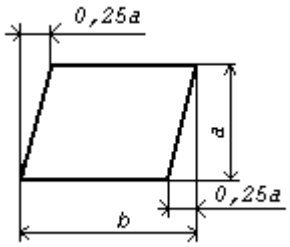
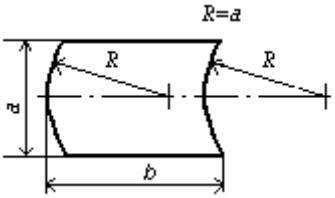
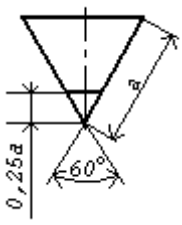
```
}

private void button2_Click(object sender, EventArgs e)
{
    textBox1.Clear();
    textBox2.Clear();
    textBox3.Clear();
    textBox4.Clear();
    textBox5.Clear();
    textBox6.Clear();
    textBox7.Clear();
    textBox8.Clear();
    textBox9.Clear();
    textBox10.Clear();
    textBox11.Clear();
    textBox12.Clear();
}

private void button3_Click(object sender, EventArgs e)
{
    for (int i = 0; i < книгиDataGridView.RowCount - 1; i++)
    {
        книгиDataGridView.Rows[i].Visible = true;
    }
}
```

ПРИЛОЖЕНИЕ Б. Выдержка из ЕСПД.

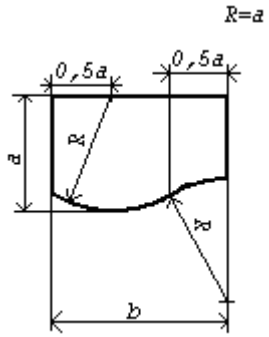
1.1. Перечень, наименование, обозначение и размеры обязательных символов и отображаемые ими функции в алгоритме и программе обработки данных должны соответствовать указанным в табл. 1.

Таблица 1.

Наименование	Обозначение и размеры в мм	Функция
1. Процесс		Выполнение операций или группы операций, в результате которых изменяется значение, форма представления или расположение данных
2. Решение		Выбор направления выполнения алгоритма или программы в зависимости от некоторых переменных условий
3. Модификация		Выполнение операций, меняющих команды или группу команд, изменяющих программу
4. Предопределенный процесс		Использование ранее созданных и отдельно описанных алгоритмов или программ
5. Ручная операция		Автономный процесс, выполняемый вручную или при помощи неавтоматически действующих средств
6. Вспомогательная операция		Автономный процесс, выполняемый устройством, не управляемым непосредственно процессором
7. Слияние		Объединение двух или более множеств в единое множество

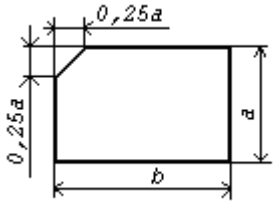
8. Выделение		Удаление одного или нескольких множеств из единого множества
9. Группировка		Объединение двух или более множеств с выделением нескольких других множеств
10. Сортировка		Упорядочение множества по заданным признакам
11. Ручной ввод		Ввод данных вручную при помощи неавтономных устройств с клавиатурой, набором переключателей, кнопок
12. Ввод-вывод		Преобразование данных в форму, пригодную для обработки (ввод) или отображения результатов обработки (вывод)
13. Неавтономная память		Ввод-вывод данных в случае использования запоминающего устройства, управляемого непосредственно процессором
14. Автономная память		Ввод-вывод данных в случае использования запоминающего устройства, не управляемого непосредственно процессором

15. Документ



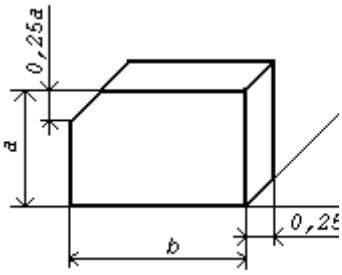
Ввод-вывод данных, носителем которых служит бумага

16. Перфокарта



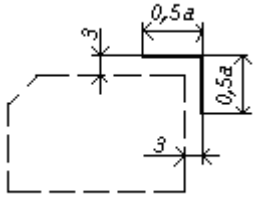
Ввод-вывод данных, носителем которых служит перфокарта

17. Колода перфокарт



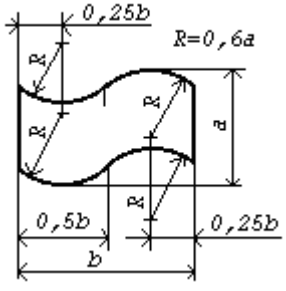
Отображение набора перфокарт

18. Файл

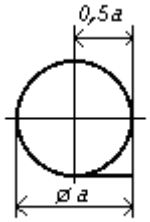
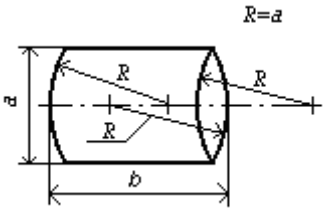
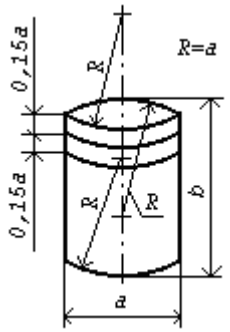
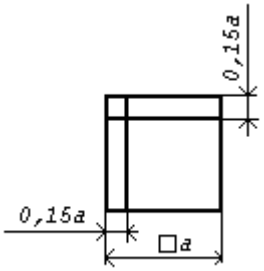
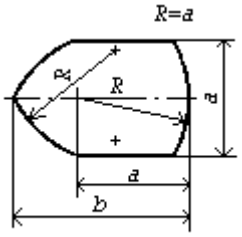
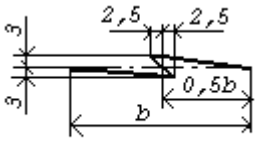


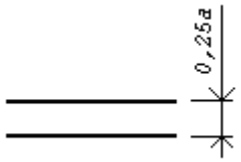
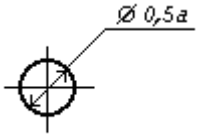
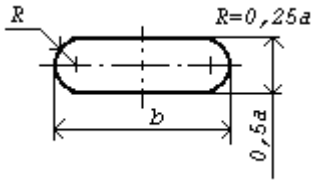
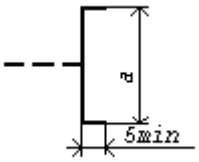
Представление организованных на основе общих признаков данных, характеризующих в совокупности некоторый объект обработки данных. Символ используется в сочетании с символами конкретных носителей данных, выполняющих функции ввода-вывода

19. Перфолента



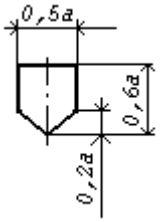
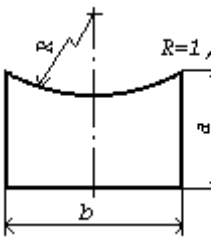
Ввод-вывод данных, носителем которых служит перфолента

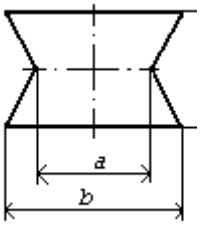
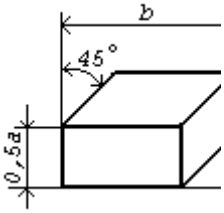
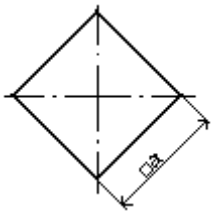
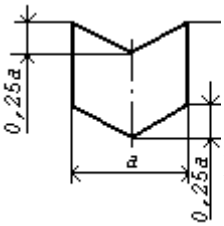
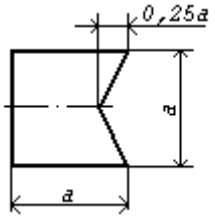
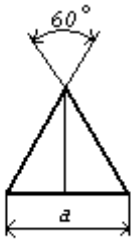
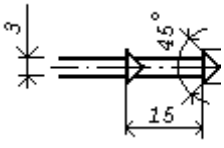
20. Магнитная лента		Ввод-вывод данных, носителем которых служит магнитная лента
21. Магнитный барабан		Ввод-вывод данных, носителем которых служит магнитный барабан
22. Магнитный диск		Ввод-вывод данных, носителем которых служит магнитный диск
23. Оперативная память		Ввод-вывод данных, носителем которых служит магнитный сердечник
24. Дисплей		Ввод-вывод данных, если непосредственно подключенное к процессу устройство воспроизводит данные и позволяет оператору ЭВМ вносить изменения в процессе их обработки
25. Канал связи		Передача данных по каналам связи
26. Линия потока		Указание последовательности между символами

27. Параллельные действия		Начало или окончание двух и более одновременно выполняемых операций
28. Соединитель		Указание связи между прерванными линиями потока, связывающими символами
29. Пуск - останов		Начало, конец, прерывание процесса обработки данных или выполнения программы
30. Комментарий		Связь между элементом схемы и пояснением

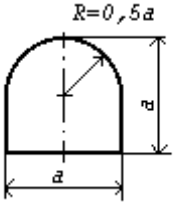
1.2. Перечень, наименование, обозначение и размеры рекомендуемых символов и отображаемые ими функции в алгоритме и программе обработки данных должны соответствовать указанным в табл. 2.

Таблица 2

Наименование	Обозначение и размеры в мм	Функция
1. Межстраничный соединитель		Указание связи между разъединенными частями схем алгоритмов и программ, расположенных на разных листах
2. Магнитная карта		Ввод-вывод данных, носителем которых служит магнитная карта

3. Ручной документ		Формирование документа в результате выполнения ручных операций
4. Архив		Хранение комплекта упорядоченных носителей данных в целях повторного применения
5. Автономная обработка		Преобразование исходных данных в результате выполнения автономных операций
6. Расшифровка		Считывание с носителя данных, перекодирование и печать на том же или другом носителе данных в результате выполнения автономной операции
7. Кодирование		Нанесение кодированной информации на носитель в результате выполнения автономной операции
8. Копирование		Образование копии носителя в результате выполнения автономной операции
9. Транспортирование носителей		Перемещение носителей данных при помощи транспортных средств или курьером
10. Материальный поток		Указание последовательности операций в технологическом процессе изготовления предметов труда, направление их перемещения

11. Источник (приемник) данных



Отправитель или получатель данных