Содержание:

ВЕДЕНИЕ

Актуальность темы исследования заключается в необходимости проведения обзора и анализа возможностей существующих языков программирования. В настоящее время использование языков программирования является востребованным в различных отраслях информационных технологий, в виду их гибкости, интеграции существующих парадигм и методов разработки программного обеспечения различного уровня сложности. Современные языки программирования часто применяются для разработки приложений с графическим пользовательским интерфейсом.

Объект исследования: возможности и средства использования современных языков программирования.

Предмет исследования: возможности создания приложений с графическим интерфейсом на базе языка программирования высокого уровня Delphi.

Основная **Цель работы** заключается в обзоре и анализе характеристик языков программирования. В качестве дополнительных целей можно выделить закрепление, расширение, обобщение и систематизацию знаний в рамках изучаемой предметной дисциплины, что реализуется вследствие проведения комплексного анализа особенностей и технических возможностей использования современных высокоуровневых языков программирования и реализации приложения, обладающего интуитивно понятным графическим интерфейсом на базе Borland Delphi.

Задачи исследования:

1. Обзор и анализ возможностей, популярности, достоинств и недостатков современных языков программирования.
2. Анализ специфики объектно-ориентированной парадигмы программирования.
3. Анализ и исследование возможностей, особенностей, достоинств и недостатков современных интегрированных сред программирования и используемых на практике языков программирования высокого уровня.

4. Реализация приложения с графическим пользовательским интерфейсом на базе использования Borland Delphi.
5. Описание результатов и возможностей разработанного программного приложения.

В рамках первого раздела написанной курсовой работы рассмотрены и приведены основные особенности языков программирования. Проведен анализ ряда литературных источников по существующим языкам программирования. В частности, даны основные определения, проведен анализ тенденций и причин востребованности современных языков программирования, рассмотрены преимущества и недостатки существующих и наиболее популярных на практике языков высокоуровневого программирования. Рассмотрена сущность и специфика объектно-ориентированной парадигмы программирования, даны определения основным принципам ООП: инкапсуляции, наследованию и полиморфизму.

Во втором разделе проведен анализ ряда литературных источников по актуальным языкам программирования. В рамках данной главы описана специфика функциональных возможностей современных объектно-ориентированных языков программирования высокого уровня. Рассмотрены программные средства, преимущества и недостатки интегрированных сред разработки Netbeans, Microsoft Visual Studio, описана языков Java и Delphi.

Третий раздел работы отражает специфику разработки графического приложения в среде Borland Delphi, которое представляет собой автоматизированную информационную систему учета складских операций. Приведены результаты разработанной модели базы данных для графического приложения с помощью СУБД MS Access, а также описаны основные функциональные формы, приведены краткие фрагменты программного кода.

В процессе написания данной работы было использовано 20 литературных источников. К наиболее значительным работам следует отнести публикации и книги следующих авторов: И.Ю. Баженива, Г.С. Иванова, Н.Вирта, Р. Себеста, Д. Скляра.

ГЛАВА 1 ХАРАКТЕРИСТИКИ И ОСОБЕННОСТИ ЯЗЫКОВ ПРОГРАММИРОВАНИЯ

Языки программирования - формальные знаковые системы, предназначенные для записи компьютерных программ. Языки программирования определяют набор лексических, синтаксических и семантических правил, определяющих внешний вид программы и действия, которые выполнит исполнитель (обычно — ЭВМ) под её управлением [15].

Высокоуровневые языки программирования разработаны для оперативности и удобства использования для разработки приложений программистами. Основная черта таких языков - абстракция, представляющая смысловую конструкцию, которая кратко описывает используемые структуры данных с возможными операциями, формализация которых на базе машинного кода является трудоемкой и длительной [6].

1.1 Анализ тенденций и причин востребованности современных языков программирования

На рис. 1 приведена диаграмма популярности и распространенности современных языков программирования за последний год.

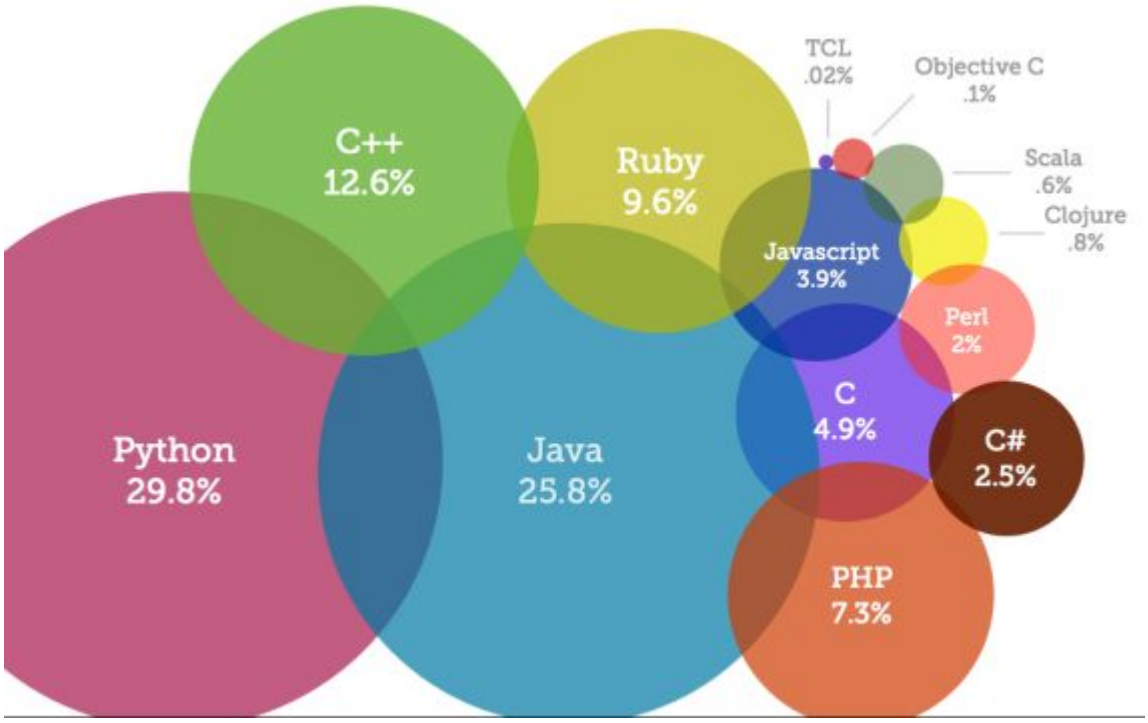

Рисунок 1 – Диаграмма распространенности языков программирования [8]

Спецификой современных языков программирования высокого уровня является отсутствие необходимости в учете особенностей отдельных архитектур и систем, что позволяет переносить и использовать разработанные приложения на разных компьютерах. Достаточно одной предварительной перекомпиляции программного продукта под конкретную операционную систему и архитектуру. Это позволяет сократить время, необходимое на разработку программного приложения, что является критичным параметром при работе над большими проектами [2].

Языки высокого уровня призваны к упрощению процессов решения трудоемких задач в программировании существующего и будущего программного обеспечения. Путем использования интерпретаторов и трансляторов обеспечивается взаимосвязь между программами, которые написаны с помощью языков высокого уровня, с разными операционными системами, средами и программируемыми устройствами. Примерами таких языков являются: C++, Java, C#, PHP, JavaScript, Python, Ruby, Delphi, Lisp, Groove и др. Рассмотрим некоторые из них подробнее [6].

1. C++. Основным преимуществом данного языка является полная универсальность, поэтому его чаще всего выбирают профессионалы. Следует заметить, что в данном языке существует достаточно простотой компилятор. Недостатки языка: не слишком удобный и простой синтаксис, длинный программный код, что осложняет дальнейшую работу с программой [15].

2. Python. Популярный, суть которого сводится к упрощению процесса разработки сложных программ. Основные достоинства компилятора: многофункциональный и простой. Недостатки: не слишком быстрый, содержит некоторые ошибки в системном коде [9].

3. PHP. Данный язык чаще всего применяется при создании веб-ресурсов и сайтов. В наше время он поддерживается подавляющим большинством провайдеров хостинга. Достоинства: гибкие возможности применения на любых ОС, высокая скорость выполнения, удобные механизмы создания пользовательского графического интерфейса. Недостатком языка является отсутствие поддержки функционирования приложений в несколько потоков[16].

4. Java. Данный язык является кроссплатформенным и широко востребованным на рынке труда. Большинство существующих операционных систем включают его в состав, в связи с тем, что функционирование ряда приложений без данного компилятора будет не эффективной. Недостатки: не высокое быстродействие разработанных приложений, необходимость в обеспечении большого количества

оперативной памяти [8].

5. JavaScript. Назначение данного языка заключается в обеспечении интерактивности веб-страницам сайта. Область применения языка JavaScript широка: браузеры и их расширения, веб-приложения, прикладное программное обеспечение, серверные решения. Преимущества языка: значительное количество специальных библиотек, которые позволяют достигать высокого уровня абстракции реализации кода. Недостатки: низкий уровень безопасности, большое количество недоработок и ошибок в расширениях [18].

6. C#. Данный язык программирования реализует компонентно-ориентированный подход в программировании приложений, что позволяет снизить машинно-архитектурную зависимость полученного кода, благодаря чему достигается высокая степень переносимости и повторного использования кода. Преимущества языка: включает расширенную и гибкую поддержку событийно-ориентированного программирования, эффективно интегрируется с существующими продуктами от Microsoft. Главными недостатками языка является довольно сложный синтаксис и не высокая, в сравнении с C++, производительность [12].

Существует мнение, что код современных программ, разработанный на высокоуровневых языках, пишется только один раз, а используется везде. Это является правдой лишь для тех программных продуктов, которые не взаимодействуют и не заточены под конкретную ОС (выполняют вычисления или производят обработку аудио-видео данных). При этом, на практике, большая часть интерактивных программных продуктов постоянно обращаются к системным библиотекам и модулям, которые значительно отличаются во всех ОС [15].

1.2. Сущность и специфика объектно-ориентированной парадигмы программирования

Объектно-ориентированное программирование (ООП) это современная парадигма программирования, опирающаяся на такие ключевые понятия, как объект и класс, позволяющая организовать гибкую структуру программного кода проекта.

ООП появилась посредством развития идеологии функционально-процедурного программирования, в которой не было четкой связи между данными, процедурами и функциями, их вызов осуществлялся вручную, а обработка передавалась посредством операторов типа goto [3].

На развитие парадигмы ООП существенное значение оказало концептуальные понятия события и компонента. Взаимодействие объектов приведено на рис.2.

Основными понятиями в ООП являются классы и объекты. Класс представляет собой формализованный, в рамках синтаксиса используемого языка программирования в составленном пространстве имен, исходный код абстрактного (неинициализированной при создании) объекта (сущности). Класс предназначен для детального описания структуры устройства объекта, т.е. это некий каркас. При этом, когда создается объект, принято считать, что он является экземпляром класса, который его описывает.

Следует отметить, что в некоторых системах, в случае выполнения программного кода интерпретатором или компилятором языка, посредством использования динамического определения типов данных, класс также может быть представлен в виде некоторого объекта. Чаще всего на практике классы разрабатываются так, чтобы их будущие объекты и их описание четко соответствовало специфике исследуемой предметной области [6].

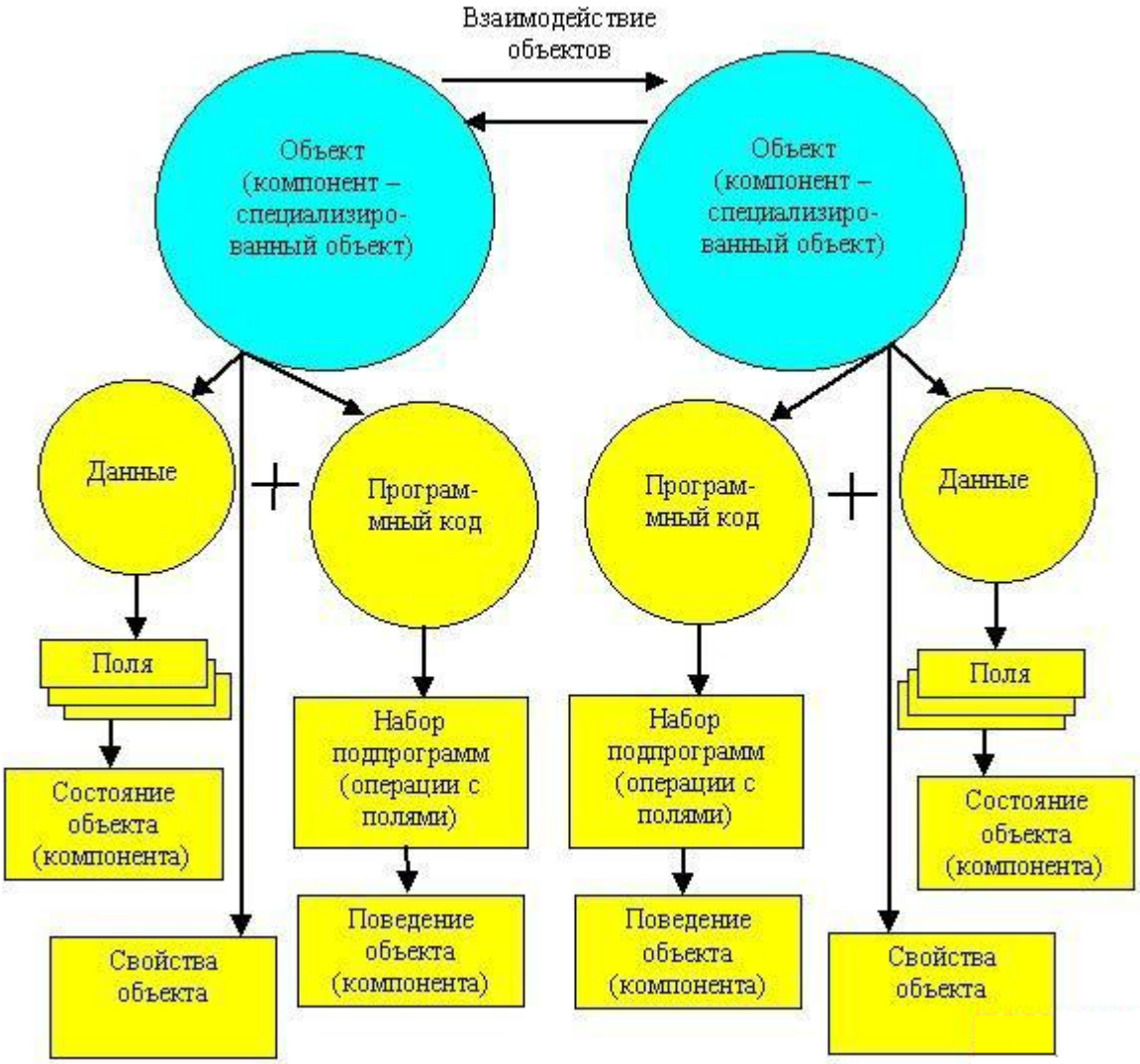

Рисунок 2 – Взаимодействие объектов и их структура в ООП

Таким образом, объект в выделенном адресном пространстве системы, создается при инициализации формирования отдельного экземпляра класса, а также в случае копирования существующего прототипа. Типичным примером является запуск итоговых результатов выполненного процесса компиляции, за которым следует связывание выполняемого кода.

Прототипом называют образцовый объект, на базе которого осуществляется создание и определение других объектов. Прототипы обладают возможностью сохранения имеющихся связей с родительскими объектами, от которых они созданы, поддерживая механизмы автоматического изменения своего состава в случае модификации структуры родительского объекта. Следует отметить, что данная особенность прототипов отлична в различных языках программирования высокого уровня.

Существует 3 главных принципа парадигмы ООП [16].

1. Инкапсуляция – принцип ООП, который заключается в осуществлении объединения данных и методов, которые находятся в рамках одного класса, за счет чего достигается сокрытие детали реализации логики класса от пользователя.

Пример схемы отображения принципа инкапсуляции парадигмы ООП приведена на рис.3.

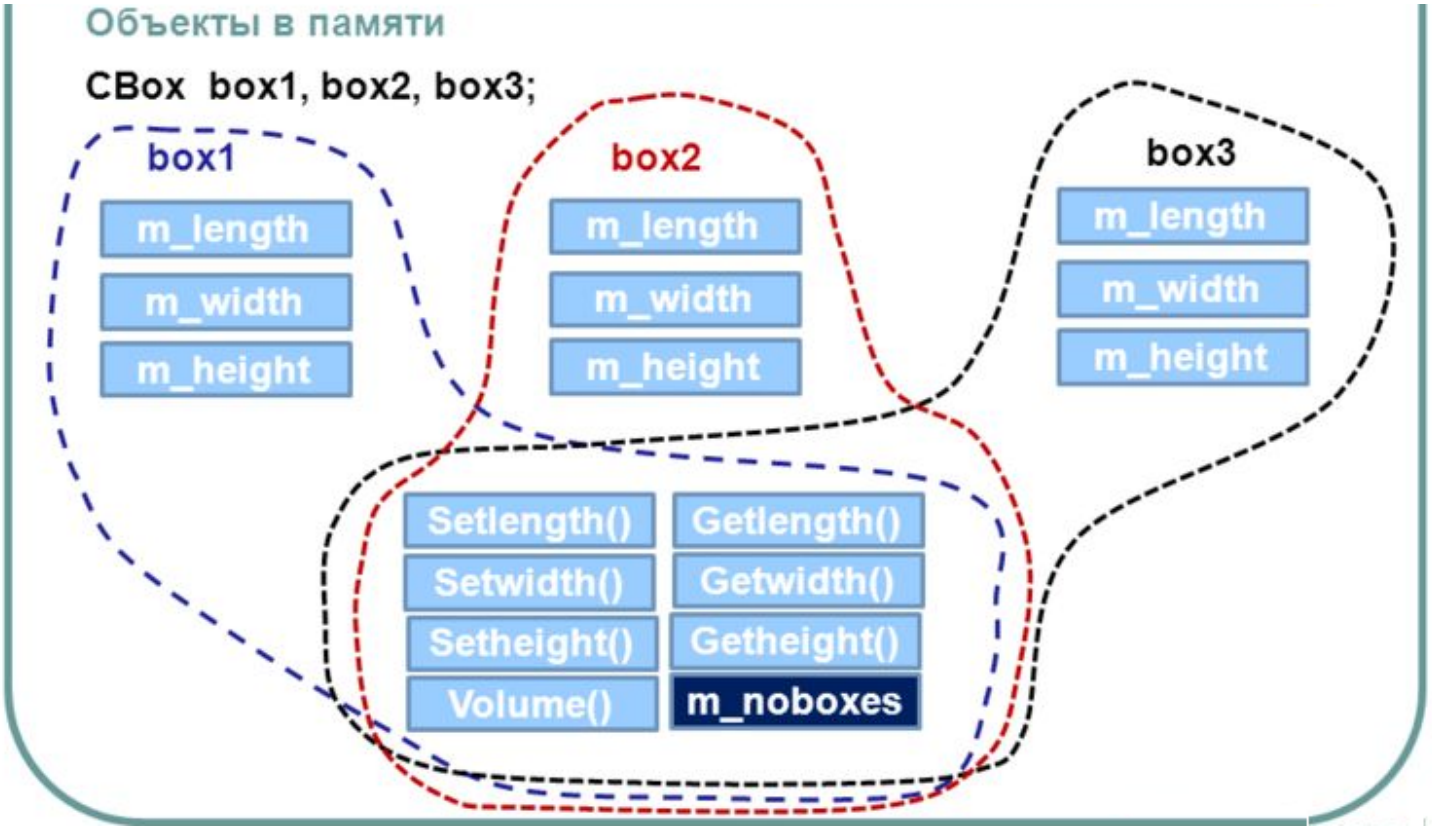

Рисунок 3 – Пример схемы отображения принципа инкапсуляции ООП

Абстрагирование в данном случае позволяет обеспечить описание комплекта наиболее значимых характеристик объекта, а незначимые и неприоритетные характеристики объекта отбрасываются. Таким образом, абстракция представляет собой формализованное описание набора всех значимых параметров и характеристик объекта.

2. Наследование - это принцип ООП, который позволяет осуществлять описание новых классов на базе использования существующего класса (или нескольких, если поддерживается механизмы множественного наследования), которое может быть дополнено новыми функциональными возможностями (новые методы, поля).

Класс, от которого производится наследование нового класса, называют родителем, базовым или суперклассом [17].

Пример иллюстрации принципа наследования в парадигме ООП приведен на рис.4 [19]. Класс Program является родителем для класса MyCar, который в свою очередь выступает родителем для класса MySportCar, все они могут дополнять описанные методы суперкласса Program новыми функциональными возможностями.

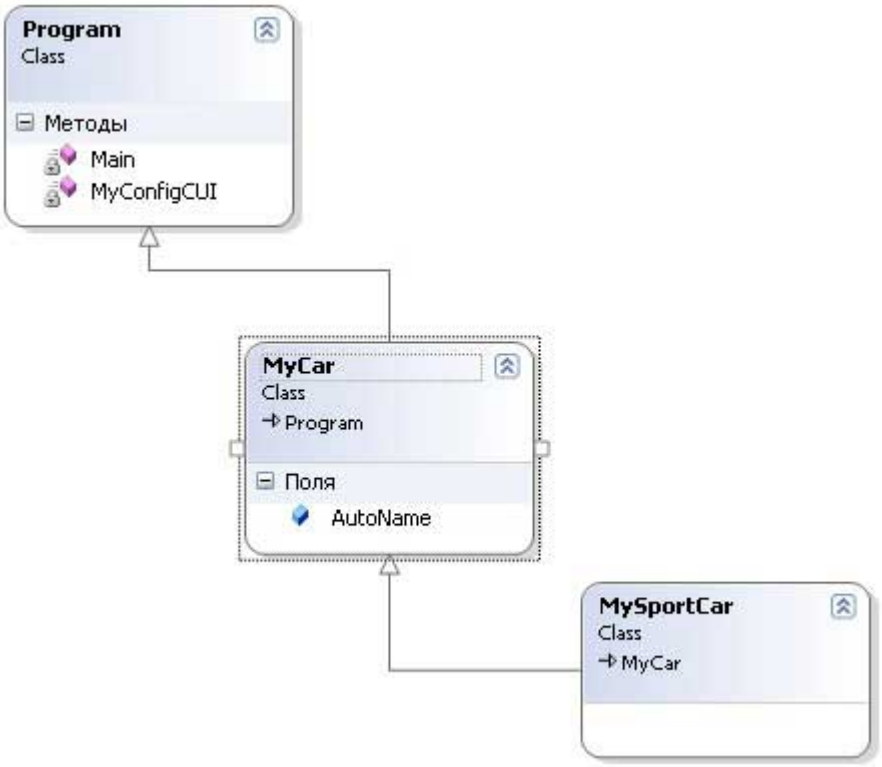

Рисунок 4 – Пример иллюстрации принципа наследования в парадигме ООП

3. Полиморфизм - это принцип ООП, заключающийся в использовании различных объектов, которые обладают идентичным интерфейсом без необходимости использования информации внутренней структуре объекта или его типе.

В настоящее время количество высокоуровневых прикладных языков программирования, которые поддерживают парадигму ООП, является наиболее широким. Каждый язык программирования реализует ООП по-разному, однако принципы ООП во всех языках, в целом, одинаковы [15].

В предметной области, которая касается непосредственно системного программирования, при этом, и в настоящее время активно применяется процедурная парадигма программирования. Чаще всего данная парадигма находит

свое воплощение в средствах языка С или С++, которые обладают максимальной гибкостью обработки данных для микроконтроллеров и других устройств низкого архитектурного уровня.

Выводы по главе 1

В рамках данной главы описаны основные особенности языков программирования. Проведен анализ ряда литературных источников по существующим высокоуровневым языкам программирования. В частности, даны основные определения, проведен анализ тенденций и причин востребованности современных языков программирования, рассмотрены преимущества и недостатки существующих и наиболее популярных на практике языков высокоуровневого программирования. Рассмотрена сущность и специфика объектно-ориентированной парадигмы программирования, даны определения основным принципам ООП: инкапсуляции, наследованию и полиморфизму.

ГЛАВА 2 СПЕЦИФИКА ФУНЦИОНАЛЬНЫХ ВОЗМОЖНОСТЕЙ СОВРЕМЕННЫХ ЯЗЫКОВ ПРОГРАММИРОВАНИЯ

2.1. Анализ функциональных возможностей языка C# и IDE Visual Studio

В настоящее время крайне популярен среди разработчиков программного обеспечения (ПО) набор продуктов компании Microsoft, включающих, в частности, интегрированную среду разработки (IDE) программ - Microsoft Visual Studio. В настоящее время актуальной версией является MVS 2015.

Интерфейс Visual Studio приведен на рис.5.

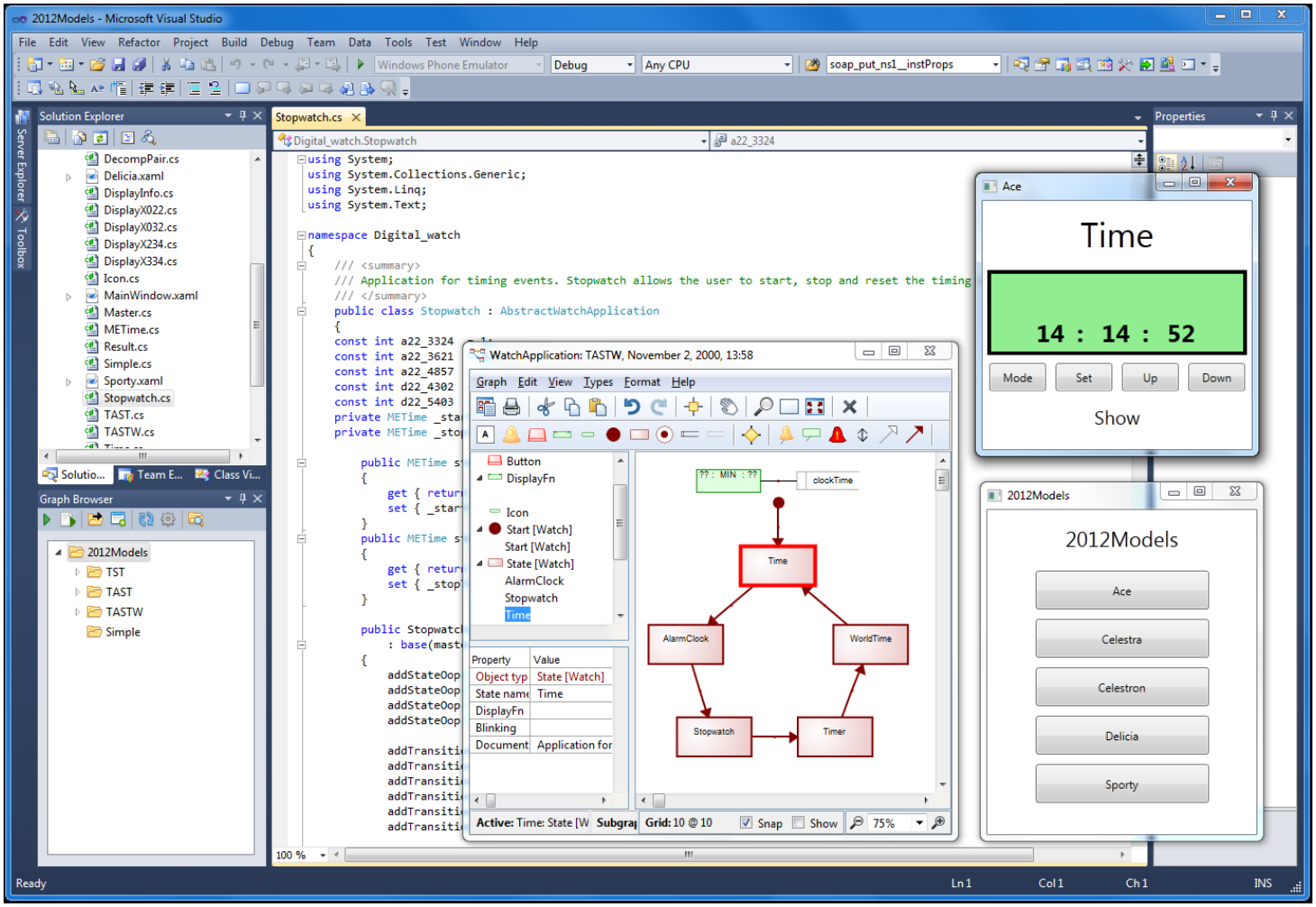

Рисунок 5 - Интерфейс среды Visual Studio

Предлагаемые средства разработки ПО продукты позволяют оперативно и гибко разрабатывать различные типы приложений, в частности, согласно [12]:

- консольные приложения;
- приложения с графическим интерфейсом, на базе использования популярной среди разработчиков десктопных решений технологии Windows Forms;
- веб-сайты, на базе использования ASP.net.

Данные решения реализуются для всех платформ, которые поддерживаются вендором средств разработки Microsoft. К таким платформам относятся операционные системе Windows 7, 8, 10, Windows Mobile, операционная система в Xbox, а также осуществляется интеграция Microsoft Silverlight и Azure.

Структура проекта на языке C# в Visual Studio приведена на рис. 6.

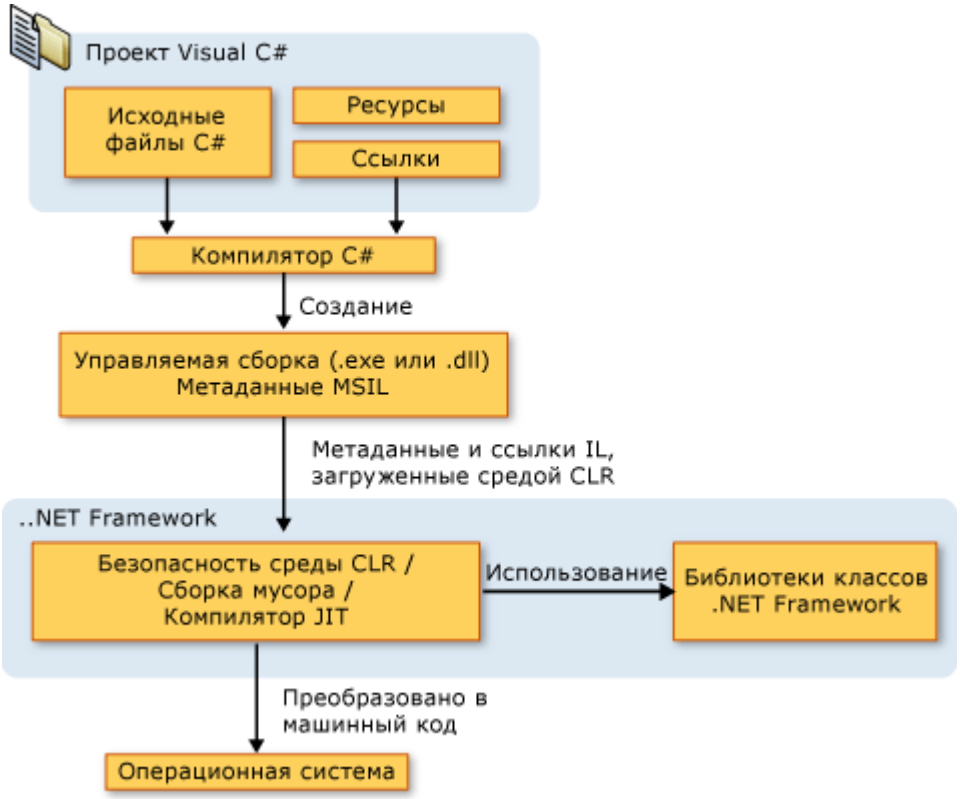

Рисунок 6 - Структура проекта на языке C# в Visual Studio

IDE Visual Studio включает в себя гибкий и современный редактор исходного программного кода, интегрируя поддержку технологии IntelliSense, а также средства оперативного профилирования и рефакторинга кода. Имеющийся в среде разработки отладчик способен функционировать в качестве отладчика на уровне исходного программного кода, а также имеются возможности его использования в качестве отладчика на машинном уровне. К другим встроенным средствам и инструментам среды следует отнести интуитивно понятный редактор форм, который способствует ускорению процесса создания и конфигурирования компонентов графического интерфейса программного приложения, дизайнеры классов, объектов и схем баз данных [11].

IDE Visual Studio, также, позволяет разрабатывать и интегрировать в проект сторонние плагины и функциональные расширения, которые обеспечивают новые возможности разработки приложений на различных уровнях. В частности, широко используются плагины добавления функций использования современных систем контроля версий (Subversion, Git), интеграции новых пакетов инструментов для визуального редактирования проектирования кода на UML-языке, создания диаграмм сценариев использования, разработки алгоритмов и др.

Интегрирования среда разработки Visual Studio поставляется в различных модификациях и может включать следующие основные компоненты [12]:

а) Visual Basic .NET;

б) Visual C ++;

в) Visual C #;

г) Visual F #.

2.2. Анализ функциональных возможностей языка Java и IDE Netbeans

Java это современный и гибкий объектно-ориентированный язык программирования высокого уровня, созданный компанией Sun Microsystems в 1995. В настоящее время разработкой и продвижением этого языка занимается компания Oracle, которая купила первоначального разработчика Java еще в 2009 году.

Синтаксис Java похож на C и C#. В официальном порте языка Java разрабатываемые приложения компилируются в некий байт-код, который при инициализации выполнения интерпретируется виртуальной машиной (JVM) под конкретную платформу (Windows, Linuz, iOS и др.) [8].

При создании высокоуровневого языка программирования Java закладывались такие цели:

- синтаксис языка должен быть понятным и простым, с поддержкой парадигмы ООП;

- разработка исполняемых приложений должна осуществляться безотказно, т.е. надежность и производительность работы программных приложений должны быть высокой;

- должна быть обеспечена максимально возможная независимость от аппаратной и программной архитектуры;

- синтаксис язык должен быть гибким и динамичным, поддерживающим возможности быстрой интерпретации обеспечение механизмов мультизадачности.

Т.е, программное приложение, которое написано с использованием средств языка Java, должно обеспечивать корректное функционирование на любой, поддерживаемой вендором, системной платформе без необходимости внесения изменений в исходный код приложения или его перекомпиляции [19].

Этого можно достичь путем компиляции исходного Java кода к виду байт-кода, который представляет собой упрощенные машинные команды. После чего программное приложение может быть использовано и запущено выполнить на любой платформе, имеющую в наличии установленную виртуальную машину, которая способна осуществлять оперативную интерпретацию байт-кода в исполняемый код.

Главное преимущество использования байт-кода - это его портативность. Однако, наличие дополнительных расходов на осуществление интерпретации во многом определяет такое свойство приложений, что они будут чаще всего работать более медленно, в сравнении с тем, которые скомпилированы непосредственно в машинный код. В связи с этим язык Java получил репутацию не оперативного, в плане быстродействия, языка. Однако, в последнее время данная проблема постоянно улучшается благодаря регулярной оптимизации программных средств и методов виртуального окружения Java, а также обновлением его версий [11].

Одна из особенностей концепции виртуальной машины заключается в том, что ошибки не приводят к полному краху работы приложения. Также, в настоящее время существуют эффективные инструменты, которые интегрируются в среду исполнения и каждый раз, в случае происшествия определенных исключительных ситуаций, происходит запись информации из памяти для осуществления корректной и последовательной отладки программы. Данные инструменты автоматизированной обработки исключений предоставляют основную информацию об исключениях в программах на Java.

Java использует автоматический сборщик мусора для управления памятью в жизненном цикле использования объекта. Разработчик самостоятельно решает и выбирает условия, по которым нужно создавать и инициализировать объекты, а JVM обеспечивает освобождение памяти уже после того, как объект не используется и становится ненужным. Это происходит тогда, когда на конкретный объект не существует ни одной ссылки. Идентифицируя данный аспект сборщик мусора автоматически удаляет его из памяти, обеспечивая тем самым рациональное ее использование [15].

Сбор мусора разрешен в любое время. Как правило, сборка мусора происходит во время бездействия программы, данный процесс автоматически форсируется при недостатке свободной памяти в куче для размещения нового объекта, что может приводить к нескольким секундам зависания приложения. Поэтому существуют реализации виртуальной машины Java со сборщиком мусора специально созданным для программирования систем реального времени.

При создании корпорацией Google оперативной системы Android в качестве языка разработки был использован Java.

При этом ядро самой операционная системы было реализовано на C, но разработчиками специально был создан пакет Android SDK, который использует именно язык Java для разработки различных программных приложений под разные версии Android [6].

Существует несколько сред разработки программного обеспечения на языке Java, таких как: NetBeans, Eclipse, IntelliJ IDEA, JDeveloper, BlueJ и некоторые другие.

NetBeans IDE – представляет собой интегрированную среду разработки программных приложений, поддерживая такие языки программирования как Java, JavaScript, Groove, C, PHP, C++, Python и др. Данный продукт является разработкой компании Oracle, но проект является открытым, поэтому развитие NetBeans осуществляется сообществом разработчиков-энтузиастов (NetBeans Community) [19].

Интерфейс данной IDE приведен на рис.7.

По качеству и функциональным возможностям актуальные версии сборки NetBeans IDE под разные операционные системы ничем не уступают коммерческим IDE, таким, как IntelliJ IDEA.

IDE NetBeans обладает удобными средствами и инструментами обеспечения рефакторинга разрабатываемых приложений, профилирования в несколько этапов с описанием исследуемых метрик, выделения синтаксических конструкций программного кода цветами, автоматическое дополнение разрабатываемых конструкций кода, оперативную интеграцию библиотек и зависимостей, а также большое количество специально разработанных плагинов и библиотек для ускорения процесса разработки ПО.

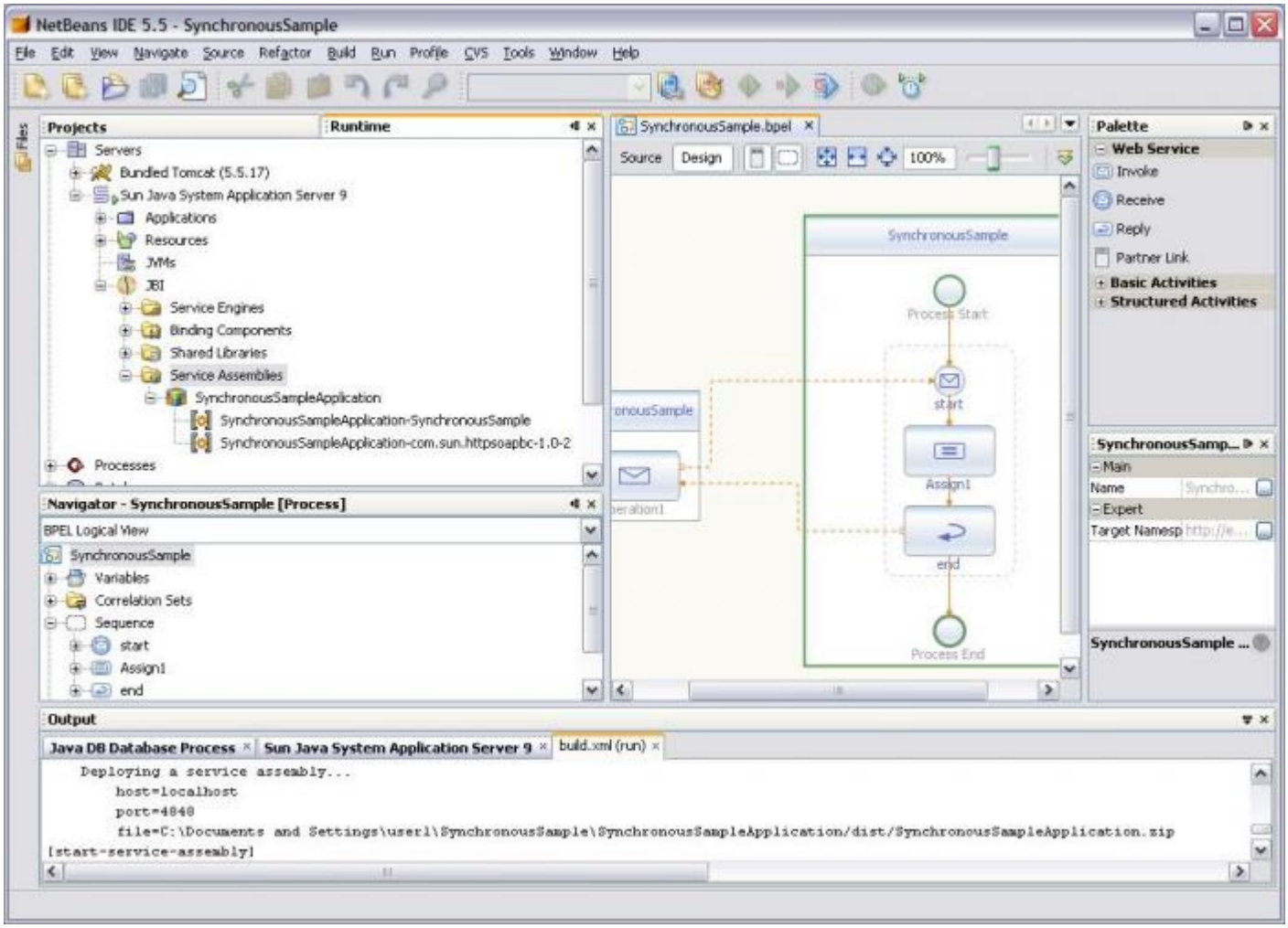

Рисунок 7 - Интерфейс среды NetBeans

2.3. Анализ функциональных возможностей языка Delphi

Delphi – это продукт Borland International для быстрого создания приложений. Процесс создания интерфейса будущей программы напоминает забаву с игровым компьютерным конструктором. Поэтому RAD-среды еще называют визуальными средами разработки, какими мы видим рабочие и диалоговые окна программы при проектировании, такими они и будут, когда программа заработает [1].

Высокопроизводительный инструмент визуального построения приложений включает в себя настоящий компилятор кода и предоставляет средства визуального программирования, несколько похожие на те, что можно обнаружить в Microsoft Visual Basic (она не является RAD-системой) или в других инструментах

визуального проектирования. В основе Delphi лежит язык Object Pascal, который является расширением объектно-ориентированного языка Pascal. В Delphi также входят локальный SQL-сервер, генераторы отчетов, библиотеки визуальных компонентов, и прочее, необходимое для того, чтобы чувствовать себя совершенно уверенным при профессиональной разработке информационных систем или просто программ для Windows-среды [3].

В первую очередь Delphi предназначен для учебных целей и оперативной разработки быстродействующих проприетарных приложений, т.к. позволяет разработчикам быстро создавать клиент-серверные приложения. Delphi генерирует компактные библиотеки (формат dll) и исполняемые файлы (формат exe), в связи с чем программные продукты, разработанные на Delphi гибко интегрируются с другими решениями. При этом, небольшие и оперативно исполняемые модули являются причиной снижения требований к архитектуре (железу) пользовательских компьютеров, что является несомненным преимуществом при эксплуатации программного приложения [6].

Преимущества Delphi по сравнению с другими конкурирующими программными средствами разработки:

- высокая скорость создания приложений;
- разработанные приложения являются высокопроизводительными и не требовательными к вычислительным ресурсам;
- возможности расширения функциональных модулей за счет нтеграции сторонних компонентов и средств в среду Delphi;
- целостная поддержка объектно-ориентированной парадигмы программирования.

Недостатки:

- статическое присоединение (linking) библиотеки VCL и компонентов к исполняемому файлу, что является не слишком рациональным распределением накапливаемого объема программы [17].
- используемая парадигме форм обуславливает не оптимальное хранение информация в исполняемом файле о форме, такой как свойства, настройки компонентов, значения по умолчанию.
- отсутствие гибких шаблонов, механизмов перегрузки операторов и C++ объектной модели, что ограничивает функции языка [20].

Выводы по главе 2

Проведен анализ ряда литературных источников по актуальным языкам программирования. В рамках данной главы описана специфика функциональных возможностей современных объектно-ориентированных языков программирования высокого уровня. Рассмотрены программные средства, преимущества и недостатки интегрированных сред разработки Netbeans, Microsoft Visual Studio, описана структура языков Java и Delphi.

ГЛАВА 3 РАЗРАБОТКА ГРАФИЧЕСКОГО ПРИЛОЖЕНИЯ В СРЕДЕ BORLAND DELPHI

3.1. Разработка сущностей и структуры базы данных для программного приложения

Сначала определим основные типы сущностей, исходя из специфики предметной области, а именно управления складом, выполнение различных учетных операций. На этапе проектирования базы данных сущности обычно представлены как существительные или выражения [4].

Модель данных сущностей, построенная средствами СУБД MS Access приведена на рис.8.

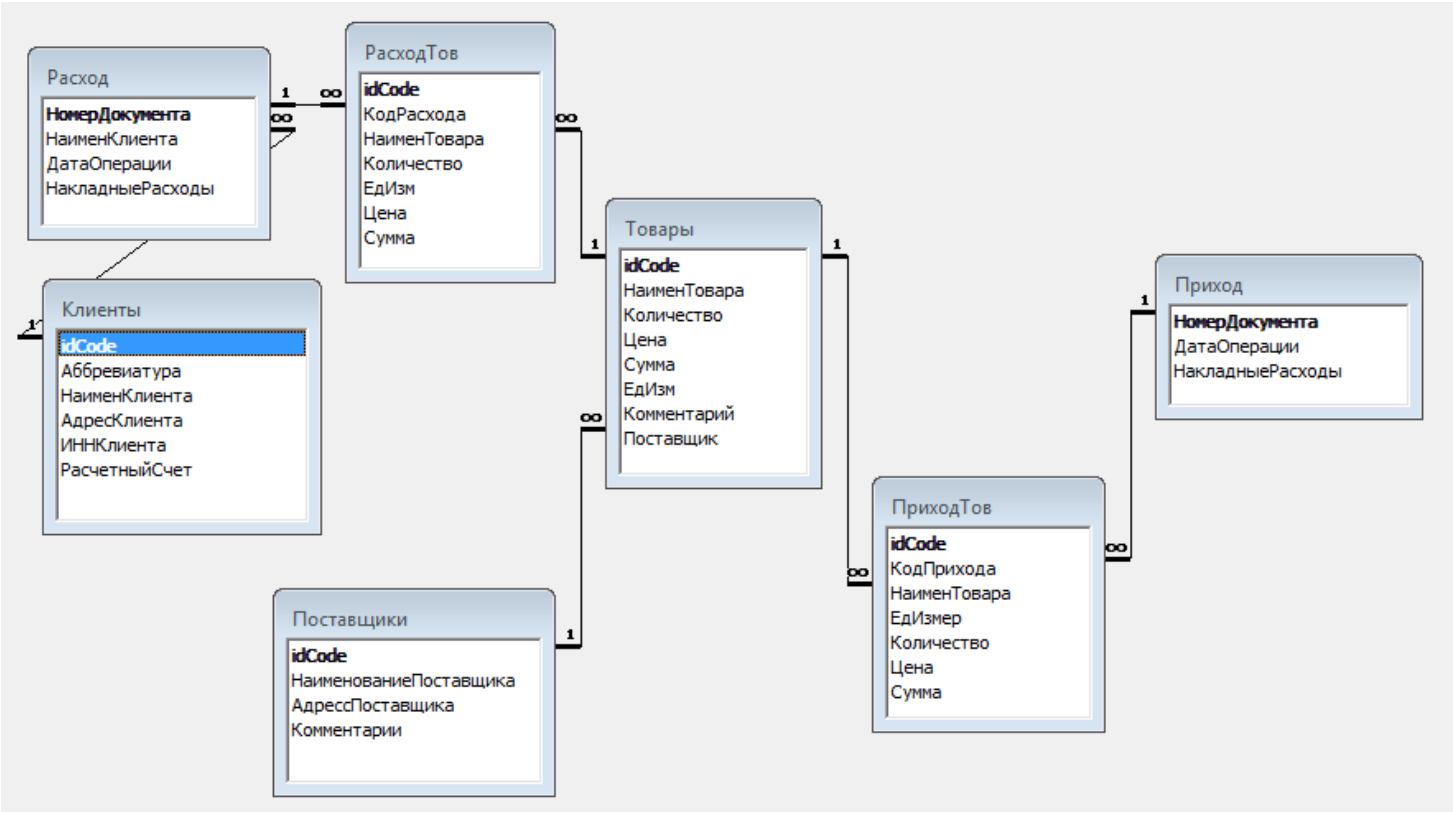

Рисунок 8 – Модель разработанной базы данных

Анализ специфики предметной области показывает, что основными типами сущностей являются:

- Клиенты;

- Товары;

- Расходы;

- Расходы товаров;

- Поставщики;

- Прибыль;

- Прибыль товаров.

3.2. Разработка форм интерфейса программного приложения

Разрабатываемое программное приложение представляет собой исполняемый программный код в виде отдельного exe-файла, обеспечивающее пользовательский интерфейс и функциональные возможности автоматизации учета процессов работы складского отдела выбранной организации.

Разработанная программа позволяет обеспечить корректность и унификацию всех бизнес-процессов организации, что позволяет значительно увеличить процент эффективности использования материальных ресурсов, что приводит к оптимизации временных затрат по отдельным складским операциям.

Разработанное программное приложение управления складскими операциями позволяет реализовать:

- учет трудовых ресурсов организации;

- управление и учет форм документации по операциям;

- визуализацию отчетов по складским операциям;

- поиск товаров в базе данных;

- отчетные операции с товаром на складе.

Разработанный интерфейс главного окна программного приложения приведен на рис.9.

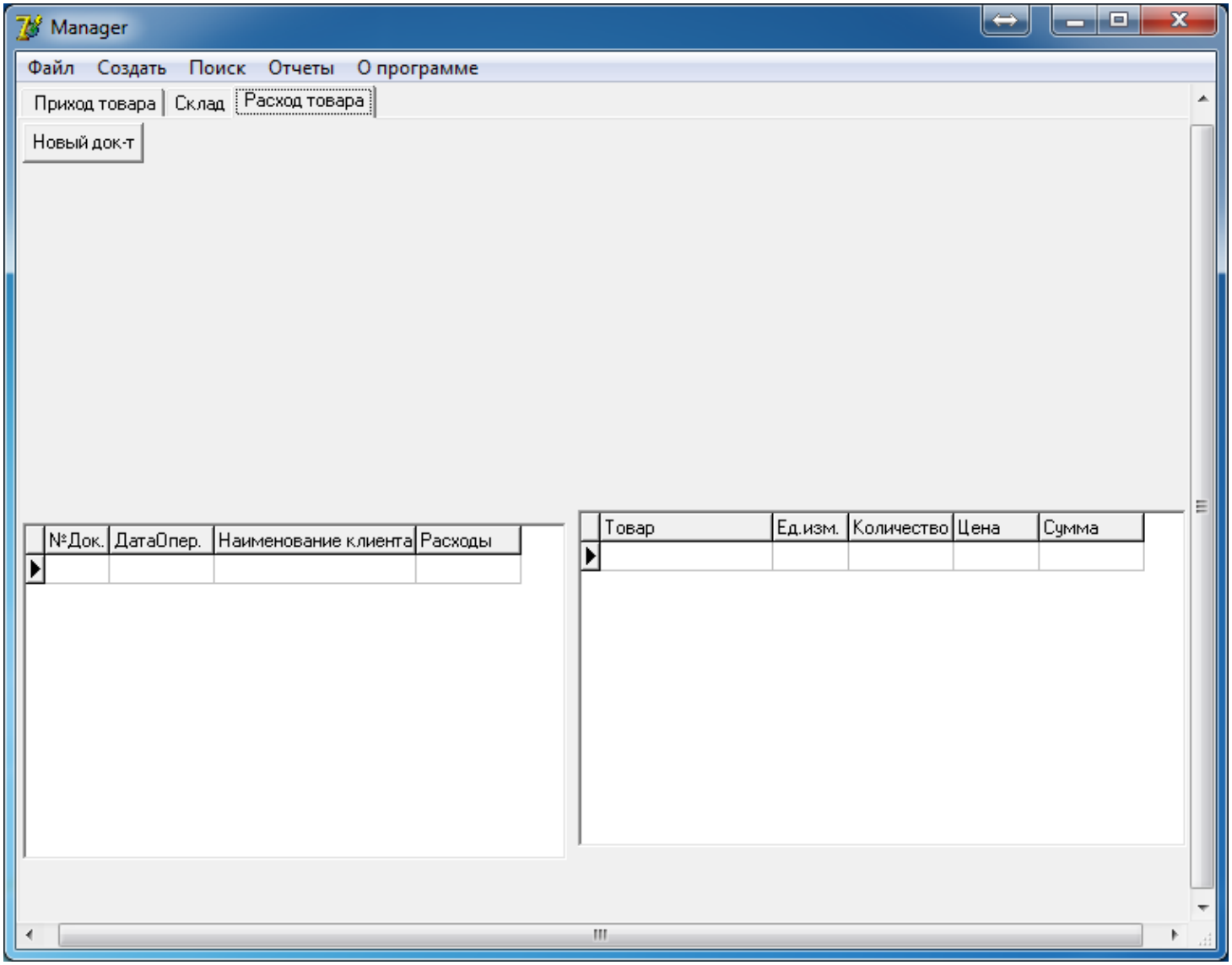

Рисунок 9 – Главное окно программы

Разработанная программа обеспечивает высокую оперативность поиска и обработки данных, которые хранятся в специально спроектированной в универсальном виде базе данных.

В следствие этого, данная программа может быть использована в составе различных промышленных компаний, занимающихся оптово-розничной торговлей, а также может быть интегрирована в рабочий процесс других предприятий в качестве отдельного унитарного модуля или скомпилирована в виде библиотеки.

Внешний вид формы учета расхода товара приведен на рис.10

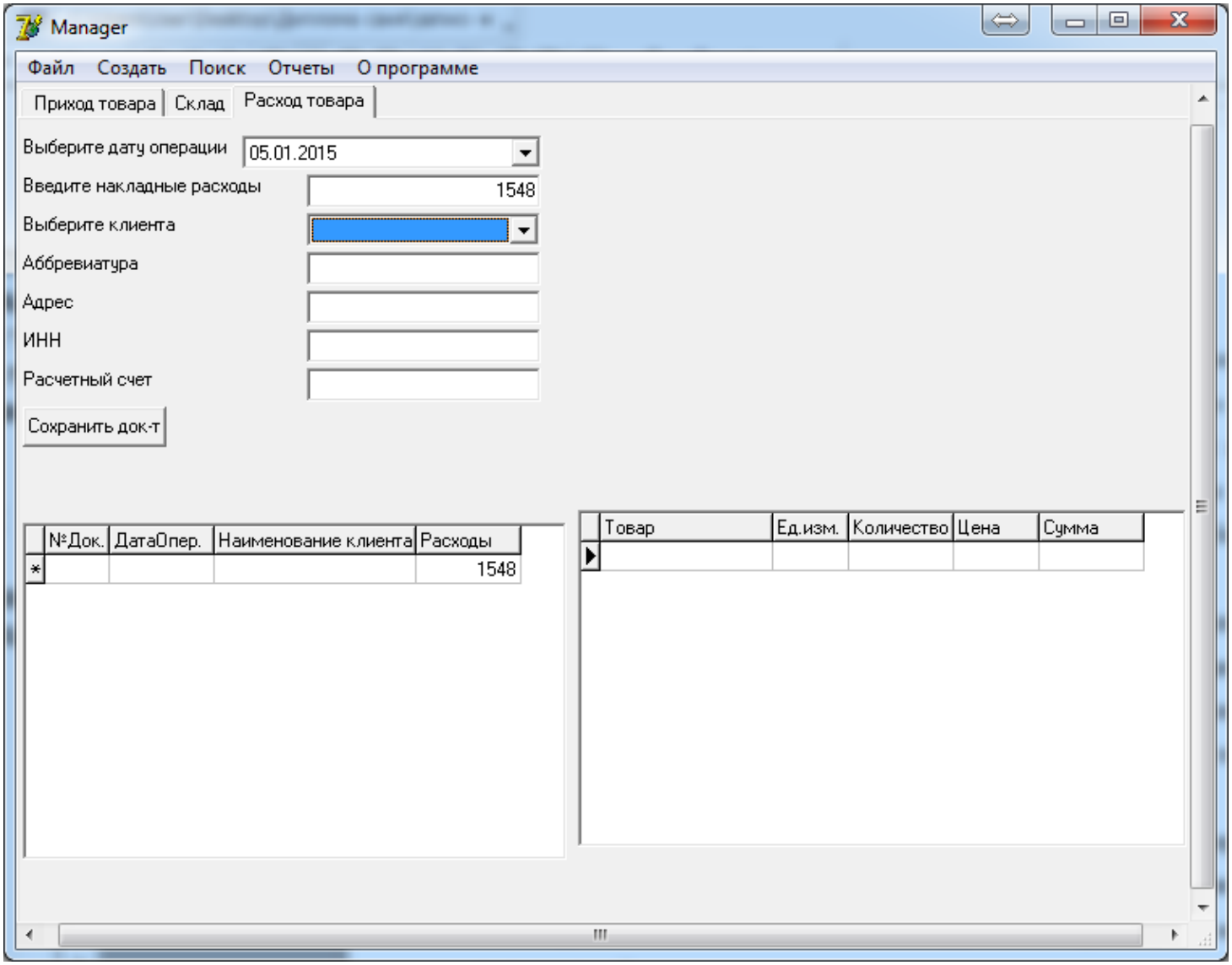

Рисунок 10 – Форма учета расхода товара

Благодаря реализованным программным компонентам возможна гибкая интеграция с другими существующими программными продуктами, что позволяет осуществлять оперативную обработку и проверку всего складского имущества.

Пример фрагмента программного кода реализации ряда функциональных возможностей данной формы приведен ниже.

```
procedure TORasxodTovarov.N1Click(Sender: TObject);

begin

close;

end;
```

```
procedure TORasxodTovarov.Button5Click(Sender: TObject);
var
summ, summ2:Currency;
begin
DataModule2.ViborDatiRasxodtov.Active:=False;
DataModule2.ViborRasxod.Active:=False;
DataModule2.ViborDatiRasxodtov.Parameters.ParamByName('[na4per]').Value:=DateTimePicke
DataModule2.ViborDatiRasxodtov.Parameters.ParamByName('[konper]').Value:=DateTimePicke
DataModule2.ViborRasxod.Parameters.ParamByName('[na4per]').Value:=DateTimePicker1.Date
DataModule2.ViborRasxod.Parameters.ParamByName('[konper]').Value:=DateTimePicker2.Date
DataModule2.ViborRasxod.Active:=true;
DataModule2.ViborDatiRasxodtov.Active:=true;
DataModule2.ViborDatiRasxodtov.First;
DataModule2.ViborRasxod.First;
while not DataModule2.ViborDatiRasxodtov.Eof do
begin
summ2:=summ2+DataModule2.ViborDatiRasxodtov.Fields.fieldbyname('').AsCurrency;
DataModule2.ViborDatiRasxodtov.next;
end;
```

Интерфейс формы ввода данных о товаре приведена на рис.11.

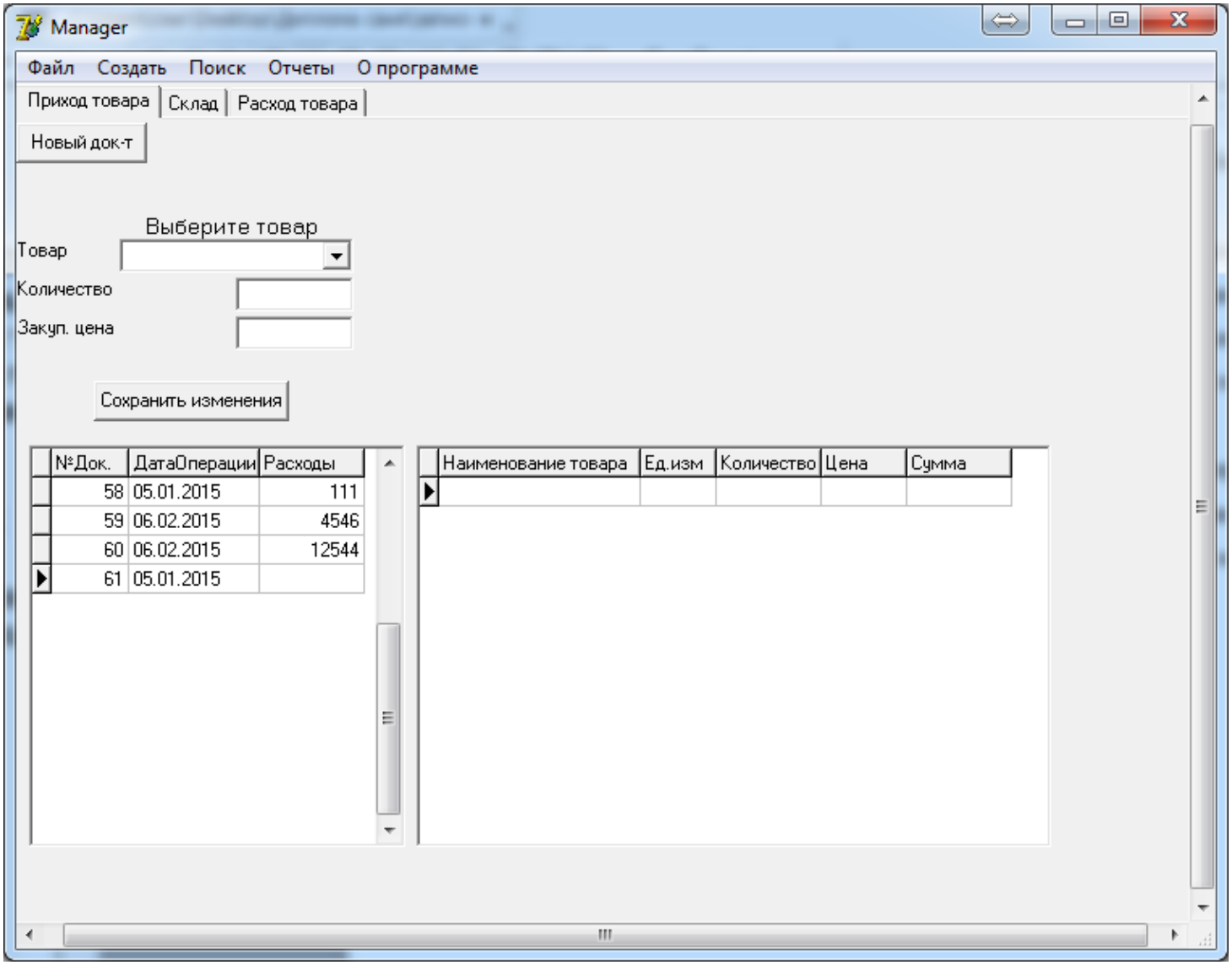

Рисунок 11 – Форма ввода данных о товаре

Оператор вводит название товара, количество и цену на данный товар.

Пример фрагмента программного кода реализации данной функциональной возможности приведен ниже. Форма ввода нового клиента приведена на рис.12.

```
procedure TGlavn.Button3Click(Sender: TObject);

begin

DataModule2.prixodtov.insert;

//появление полей для заполнения

button3.Visible:=false;button11.Visible:=true;DBLookupComboBox8.Visible:=true;
```

label7.Visible:=true;label11.Visible:=true;label9.Visible:=true;

label41.Visible:=true;label42.Visible:=false;dbedit8.Visible:=true;

dbedit9.Visible:=true;dbedit20.Visible:=false;

end;

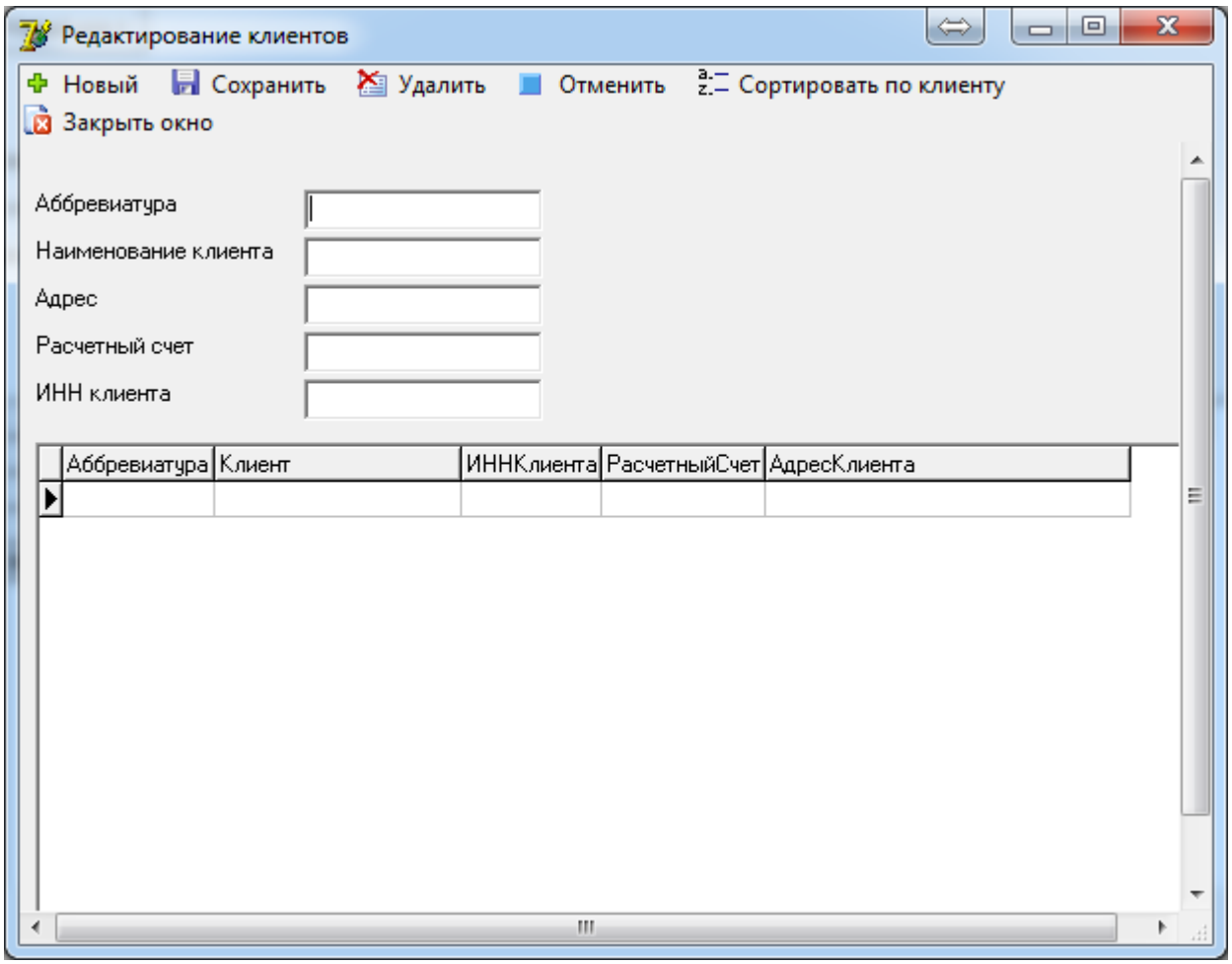

Рисунок 12 – Форма ввода нового клиента

Оператор вводит в базу данных название организации, адрес, счет клиента.

В окне реализована возможность добавления нового Клинта, сохранения введенных данных, удаление данных и сортировки.

Пример фрагмента программного кода реализации данной функциональной возможности приведен ниже. Форма ввода поставщиков приведена на рис.13.

//Меню создать -> клиент

```
procedure TGlavn.N8Click(Sender: TObject);

begin

newklient.ShowModal;

end;
```

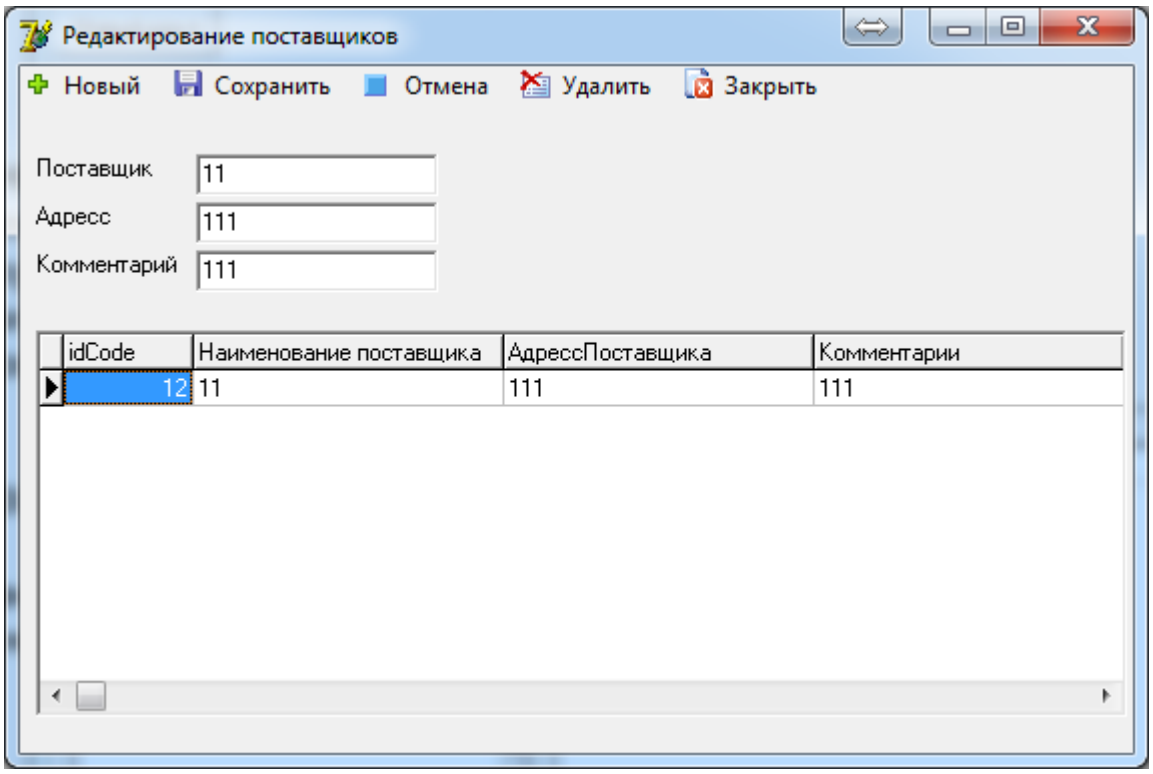

Рисунок 13 – Форма ввода поставщиков

Оператор вводит в базу данных новых поставщиков, их адрес и комментарии. Есть возможность сохранения, удаления и отмены введенной информации.

Пример фрагмента программного кода реализации данной функциональной возможности приведен ниже.

```
//Меню создать -> поставщик

procedure TGlavn.N10Click(Sender: TObject);

begin

NewPostav4ik.ShowModal;

end;
```

Интерфейс компонентов реализации поиска по базе даннях приведен на рис.14.

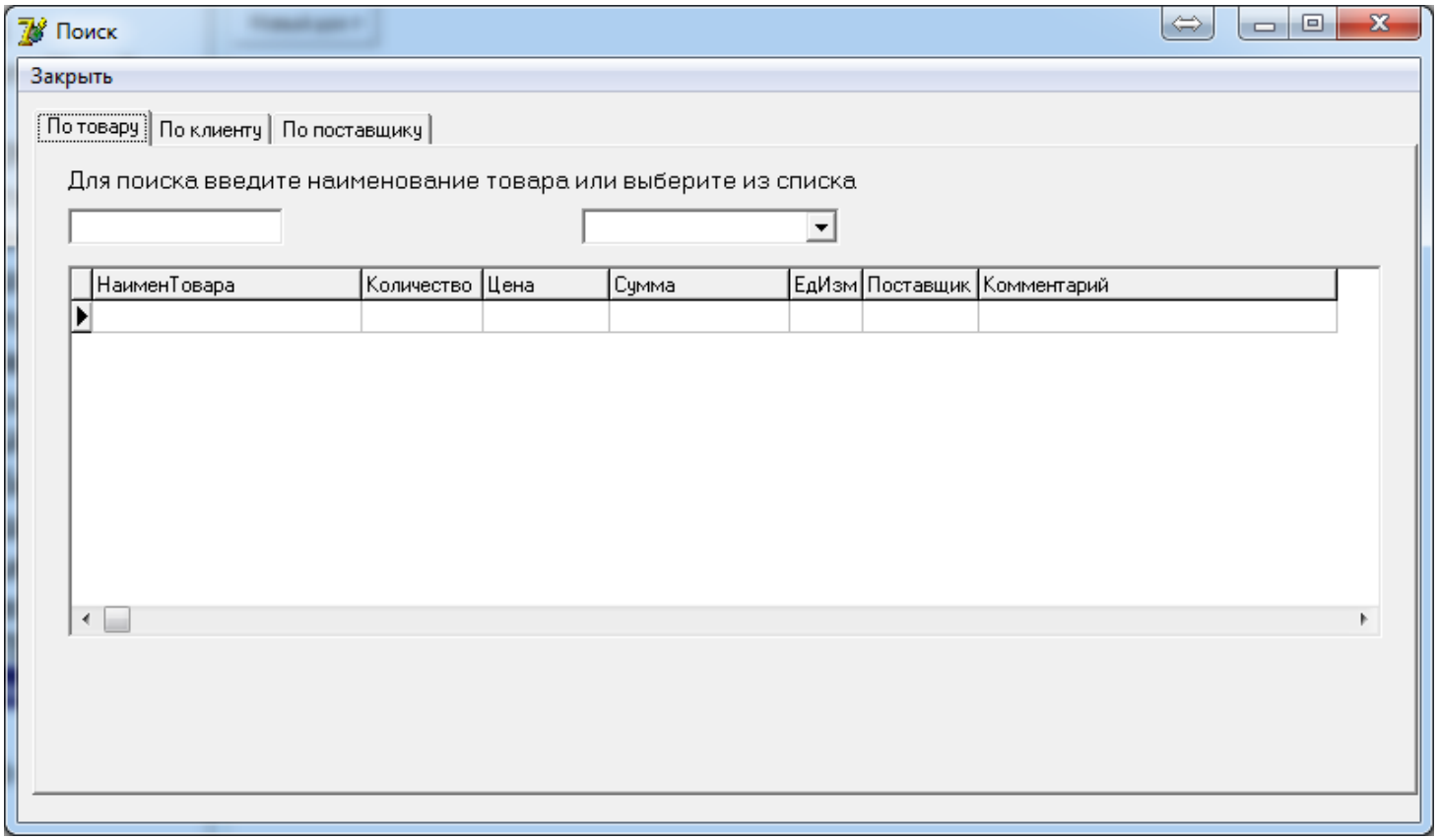

Рисунок 14 – Интерфейс компонентов реализации поиска по базе данных

В программе есть возможность поиска по клиентам, по поставщикам, и поиск по названию и штрих кода товара.

Пример фрагмента программного кода реализации данной функциональной возможности приведен ниже.

```
procedure TPoisk.Edit1Change(Sender: TObject);

begin

DataModule2.PoiskPoTovary.Active:=False;

DataModule2.PoiskPoTovary.Parameters.ParamByName('tovar').Value:=Edit1.Text;

DataModule2.PoiskPoTovary.Active:=active;

end;

procedure TPoisk.Edit2Change(Sender: TObject);
```

```
begin

DataModule2.PoiskPoKlienty.Active:=False;

DataModule2.PoiskPoKlienty.Parameters.ParamByName('Klient').Value:=Edit2.Text;

DataModule2.PoiskPoKlienty.Active:=active;

end;
```

Форма выдачи отчета по рас ходам приведена на рис.15. В данном окне оператор заполняет отчет по расходам и прибыли предприятия.

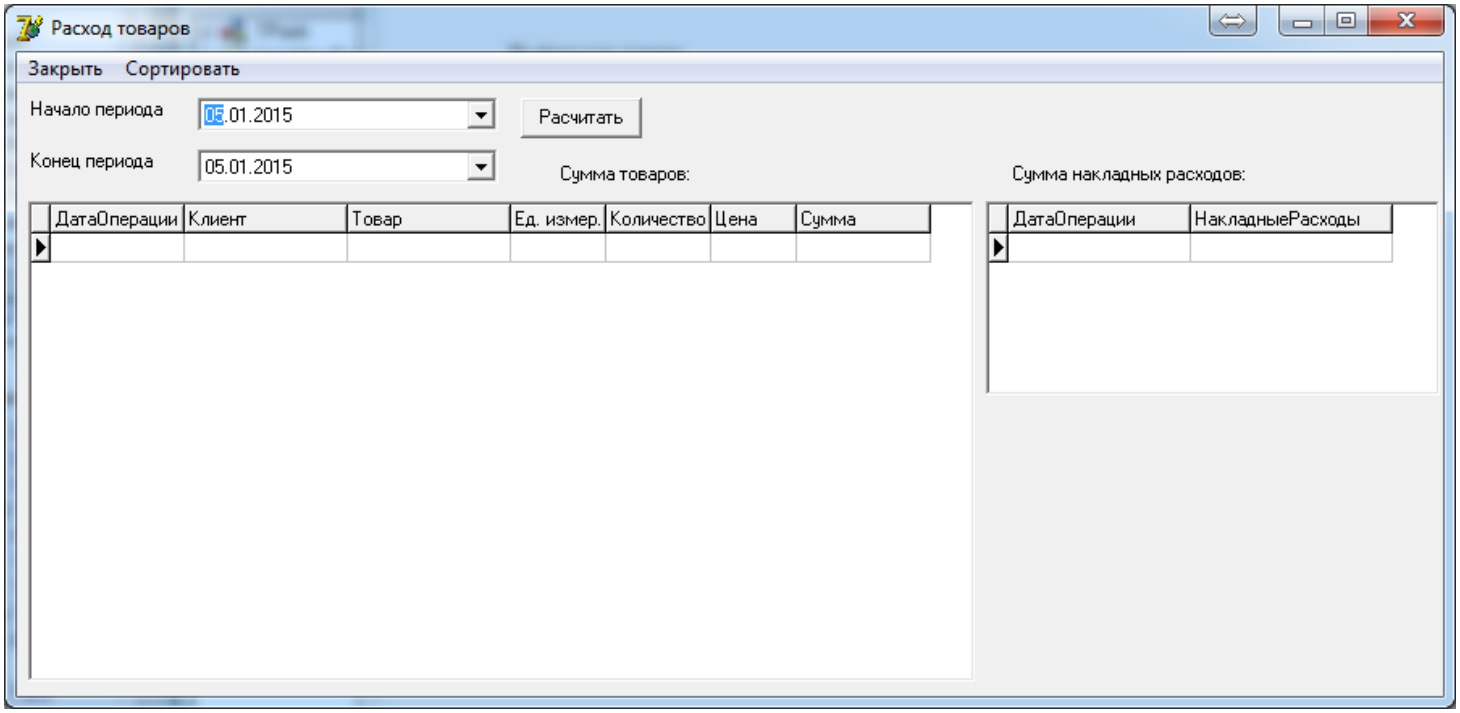

Рисунок 15 – Форма выдачи отчета по расходам

Сгенерированный по запросу прайс лист товаров приведена на рис.16.

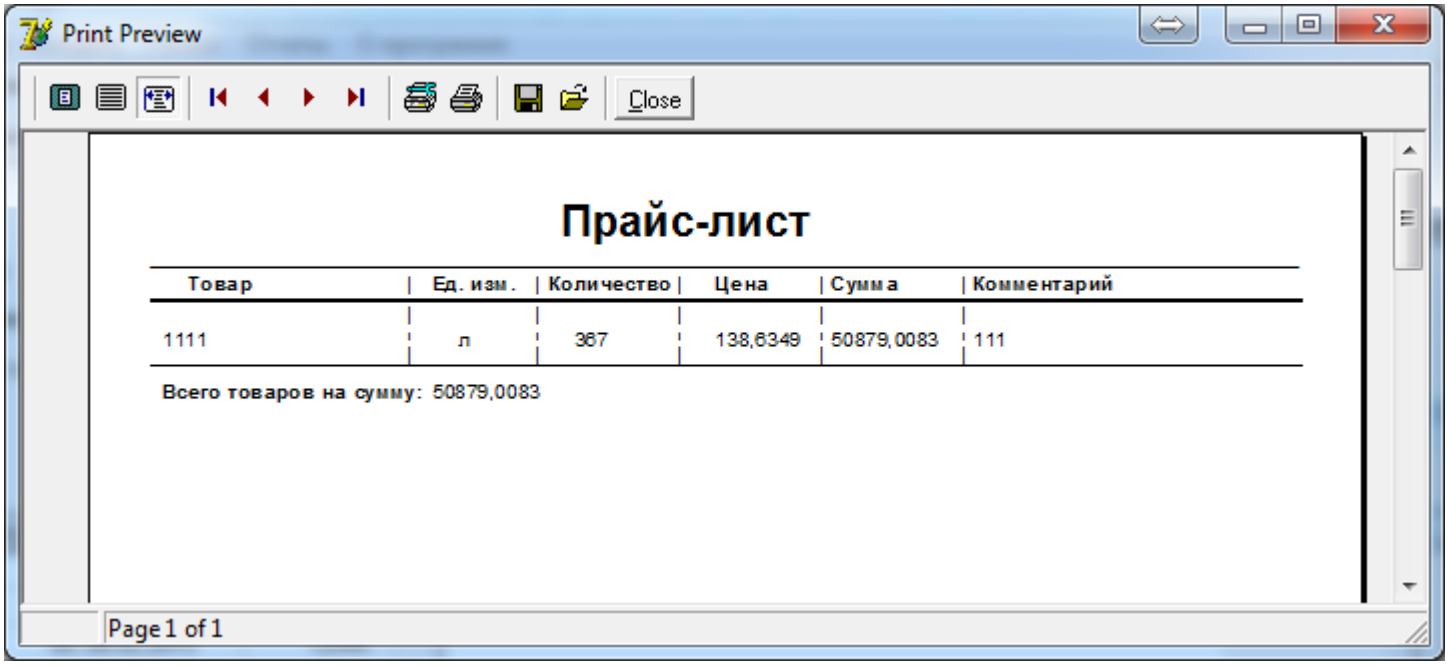

Рисунок 16 – Прайс лист товаров

Выводы по главе 3

Третий раздел работы посвящен разработке графического приложения в среде Borland Delphi, представляющего собой автоматизированную информационную систему учета складских операций. Приведены результаты разработанной модели базы данных для графического приложения с помощью СУБД MS Access, а также описаны основные функциональные формы.

ЗАКЛЮЧЕНИЕ

В рамках первого раздела написанной курсовой работы рассмотрены и приведены основные особенности языков программирования. Проведен анализ ряда литературных источников по существующим языкам программирования. В частности, даны основные определения, проведен анализ тенденций и причин востребованности современных языков программирования, рассмотрены преимущества и недостатки существующих и наиболее популярных на практике языков высокоуровневого программирования. Рассмотрена сущность и специфика объектно-ориентированной парадигмы программирования, даны определения основным базовым принципам ООП: инкапсуляции, наследованию и полиморфизму.

Во втором разделе проведен анализ ряда литературных источников по актуальным языкам программирования. В рамках данной главы описана специфика функциональных возможностей современных объектно-ориентированных языков программирования высокого уровня. Рассмотрены программные средства, преимущества и недостатки интегрированных сред разработки Netbeans, Microsoft Visual Studio, описана структура языков Java и Delphi.

Третий раздел работы отражает специфику разработки графического приложения в среде Borland Delphi, которое представляет собой автоматизированную информационную систему учета складских операций. Приведены результаты разработанной модели базы данных для графического приложения с помощью СУБД MS Access, а также описаны основные функциональные формы, приведены краткие фрагменты программного кода приложения.

В процессе выполнения курсовой работы решены следующие задачи:

1. Обзор и анализ возможностей, популярности, достоинств и недостатков современных языков программирования высокого уровня.
2. Анализ специфики объектно-ориентированной парадигмы программирования.
3. Анализ и исследование возможностей, особенностей, достоинств и недостатков современных интегрированных сред программирования и используемых на практике языков программирования высокого уровня.
4. Реализация приложения с графическим пользовательским интерфейсом на базе использования Borland Delphi.
5. Описание результатов и возможностей разработанного программного приложения.

Учитывая, что все поставленные задачи курсовой работы решены, можно обоснованно утверждать, что главная цель исследования – достигнута.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Баженива И.Ю. Самоучитель программиста Delphi. – М.: Кудиц образ, 2011. – 278 с.
2. Бобровский С.И. Delphi 7.Учебный курс. – СПб: Питер, 2012. – 736 с.
3. Вирт Н. Алгоритмы и структуры данных. – М.: Мир, 2011. – 651 с.
4. Дейт К. Дж. Введенния в системи баз даних. – М.: Вильямс, 2011. – 315 с.
5. Гофман В.З. Работа с базами данных в Delphi. – СПб.:БХВ–Петербург, 2012. – 656 с.

6. Иванова Г.С. Основы программирования Учебник для вузов. – М.: Изд-во МГТУ им. Н.Э. Баумана, 2012. –303 с.
7. Никаноров С.П. Мастера Delphi. – М.: Мир, 2013. – 415 с.
8. Нимейер П. Программирование на Java. – Москва: Эксмо, 2014. – 1216 с.
9. Орлов С.А. Теория и практика языков программирования. – СПб.: Питер, 2014. - 690 с.
10. Пестриков В. М. Delphi на примерах. – СПб.: БХВ-Петербург, 2012. – 496 с.
11. Раскин Д. Интерфейс: новые направления в проектировании компьютерных систем. – СПб.: Символ–Плюс, 2014. – 227 с.
12. Рихтер Д. Программирование на языке C#. – СПб.: Питер, 2013. – 412 с.
13. Себеста Р. Основные концепции языков программирования. – М.: Эксмо, 2013. – 316 с.
14. Семакин И.Г. Информатика и информационно-коммуникационные технологии. Базовый курс: - М: БИНОМ, Лаборатория знаний, 2012. – 314 с.
15. Синицын С. Программирование на языке высокого уровня. - СПб.: Академия, 2011. - 400 с.
16. Скляр Д. PHP. Принципы программирования. – СПб.: БХВ-Петербург, 2011. – 216 с.
17. Стивенс Р. Delphi. Готовые алгоритмы. – СПб.: Питер, 2011. – 381 с.
18. Сухов К. Node.js. Путеводитель по технологии. - М.:ДМК Пресс, 2015. - 416 с.
19. Сьерра К. Изучаем Java. – М.: Эксмо, 2013. – 414 с.
20. Федоров А. Г. Создание Windows-приложений в среде Delphi. – М.: ТОО «Компьютер Пресс», 2011. – 347 с.