МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕРМСКИЙ ГОСУДАРСТВЕННЫЙ ГУМАНИТАРНО-ПЕДАГОГИЧЕСКИЙ УНИВЕРСИТЕТ»

МАТЕМАТИЧЕСКИЙ ФАКУЛЬТЕТ

Кафедра информатики и вычислительной техники

Выпускная квалификационная работа

ДИДАКТИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ДЛЯ ИЗУЧЕНИЯ РАСШИРЕННЫХ ВОЗМОЖНОСТЕЙ ПРОГРАММИРОВАНИЯ В ВИЗУАЛЬНЫХ СРЕДАХ (НА ПРИМЕРЕ VISUAL BASIC)

Работу выполнил:
Студент 151 группы
направления подготовки 44.03.05
«Педагогическое образование»
(с двумя профилями подготовки),
профили «Математика и Информатика»,
Гайнутдинов Роман Русланович

_____
подпись

«Допущена к защите в ГЭК»
Зав. кафедрой

_____
подпись

«____» _____ 2018г.

Руководитель:
канд. пед. наук, доцент,
заведующий кафедрой
информатики и ВТ
Шестаков Александр Петрович

_____
подпись

ПЕРМЬ
2018

ОГЛАВЛЕНИЕ

ОГЛАВЛЕНИЕ 2

ВВЕДЕНИЕ 3

ГЛАВА 1 ПРОГРАММИРОВАНИЕ КАК ОДИН ИЗ РАЗДЕЛОВ ИНФОРМАТИКИ 5

1.1. Место программирования в информатике 5

1.2. Программирование в вузовском курсе информатики 10

1.3. Цели и задачи дисциплин 12

Глава 2 ДИДАКТИЧЕСКОЕ И МЕТОДИЧЕСКОЕ ОБЕСПЕЧЕНИЕ КУРСОВ ВИЗУАЛЬНОГО ПРОГРАММИРОВАНИЯ 16

2.1. Что такое Visual Basic? 16

2.2. Пользовательские типы данных 20

2.3. Создание классов и объектов 22

2.4. Основные компоненты 28

2.5. Графика 33

2.6. Файлы 37

2.7. База данных 40

2.8. Сетевые компоненты 43

2.9. Мультимедиа 51

ЗАКЛЮЧЕНИЕ 56

СПИСОК ЛИТЕРАТУРЫ 58

ВВЕДЕНИЕ

В связи с развитием информационных технологий, требуется большое количество высококвалифицированных специалистов, умеющих разрабатывать программные приложения любой сложности. Основную часть по подготовке специалистов в области разработки продуктов информационных технологий берут на себя ВУЗы.

Обучение программированию осложняется тем, что подготовка по данному направлению в школе очень слабая, даже в профильных классах. Для устранения этого пробела изучение программирования в вузах начинается с основ, на чем теряется достаточно много времени. Как правило, обучение начинается на простом языке (например, Pascal), если изучаются основы программирования и в среде VisualBasic, если изучается визуальное программирование.

Для проведения лабораторных занятий по программированию, которые, собственно, и позволяют практически овладеть азами программирования, требуются задачи и задания по каждому изучаемому разделу, желательно индивидуальные. В основной части литературы, посвященной визуальному программированию, присутствует существенный недостаток: немногочисленны или отсутствуют практические задания, позволяющие читателю в полной мере освоить объектно-ориентированное программирование. В большинстве пособий задачи разрозненны и не имеют определенной тематики, распределение задач по уровню сложности не равномерное, что усложняет практическое усвоение приемов программирования.

Таким образом, существует проблема в дидактическом наполнении дисциплин, связанных с визуальным программированием. По этой причине выполнение настоящей работы представляется ***актуальным***.

Указанная проблема определила ***объект*** исследования: процесс обучения программированию, и ***предмет*:** дидактическое обеспечение визуального программирования. В связи с указанным предметом и объектом была поставлена следующая ***цель*:** разработать учебно-методический комплект для изучения расширенных возможностей программирования в визуальных средах, предназначенный для обеспечения преподавания программирования в ВУЗе.

Исходя из цели исследования, выделены следующие ***задачи***, определившие структуру работы:

- проанализировать ФГОС ВО;
- проанализировать учебную и научную литературу по обучению визуальному программированию;
- разработать комплект дидактических материалов (теоретическая подборка материала, текстовые задачи, тестовые задания) в поддержку курса ООП;
- разработать методику применения комплекта дидактических материалов.

Работа состоит из Введения, двух глав (Глава 1 Программирование как один из разделов информатики, Глава 2 Дидактическое и методическое обеспечение курсов визуального программирования), Заключения и списка литературы.

ГЛАВА 1 ПРОГРАММИРОВАНИЕ КАК ОДИН ИЗ РАЗДЕЛОВ ИНФОРМАТИКИ

1.1. Место программирования в информатике

Для того чтобы подготовить специалистов информационных технологий, нужно четко представлять, какие знания им необходимы. Уже в школе человек получает самые основные знания об информатики. Но этого недостаточно, тем более учитывая современное состояние обучения информатики в школе. Поэтому подготовка специалистов осуществляется в высших учебных заведениях. Для подготовки уделяется достаточно большое внимание такому разделу информатики как программирование. Программирование пронизывает все разделы информатики, начиная с теоретических основ информатики и заканчивая социальной информатикой.

Рассмотрим современную концепцию предметной области информатики предложенной на II Международном конгрессе ЮНЕСКО «Образование и информатика» (Москва, июнь 1996 год).

Таблица 1

ФУНДАМЕНТАЛЬНЫЕ ОСНОВЫ ИНФОРМАТИКИ	
ТЕОРЕТИЧЕСКАЯ ИНФОРМАТИКА	Информация как семантическое свойство материи. Информация и эволюция в живой и неживой природе. Начало общей теории информации. Методы измерения информации. Макро– и микроинформация. Математические и информационные модели. Теория алгоритмов. Стохастические методы в информатике. Вычислительный эксперимент как методология научного исследования. Информация и знания. Семантические аспекты интеллектуальных процессов и информационных систем. Информационные системы искусственного интеллекта. Методы представления знаний. Познание и творчество как информационные процессы. Теория и методы разработки и проектирования информационных систем и технологий.

<table>
<tr><td rowspan="4">СРЕДСТВА АВТОМАТИЗАЦИИ</td><td>ТЕХНИЧЕСКИЕ</td><td colspan="2">ОБРАБОТКИ ОТОБРАЖЕНИЯ И ПЕРЕДАЧИ ДАННЫХ</td><td>Персональные компьютеры. Рабочие станции. Устройства ввода/вывода и отображения информации. Аудио– и видеосистемы, системы мультимедиа. Сети ЭВМ. Средства связи и компьютерные телекоммуникационные системы.</td></tr>
<tr><td rowspan="3">ПРОГРАММНЫЕ</td><td colspan="2">СИСТЕМНЫЕ</td><td>Операционные системы и среды. Системы и языки программирования. Сервисные оболочки, системы пользовательского интерфейса. Программные средства межкомпьютерной связи (системы теледоступа), вычислительные и информационные среды.</td></tr>
<tr><td rowspan="2">РЕАЛИЗАЦИЯ ТЕХНОЛОГИЙ</td><td>УНИВЕРСАЛЬНЫХ</td><td>Текстовые и графические редакторы. Системы управления базами данных. Процессоры электронных таблиц. Средства моделирования объектов, процессов, систем. Информационные языки и форматы представления данных и знаний; словари; классификаторы; тезаурусы. Средства зашиты информации от разрушения и несанкционированного доступа.</td></tr>
<tr><td>ПРОФЕССИОНАЛЬНО ОРИЕНТИРОВАННЫХ</td><td>Издательские системы. Системы реализации технологий автоматизации расчетов, проектирования, обработки данных (учета, планирования, управления, анализа, статистики и т.д.). Системы искусственного интеллекта (базы знаний, экспертные системы, диагностические, обучающие и др.).</td></tr>
<tr><td colspan="4">ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ</td><td>Ввода/вывода, сбора, хранения, передачи и обработки данных. Подготовки текстовых и графических документов, технической документации. Интеграции и коллективного использования разнородных информационных ресурсов. Защиты информации. Программирования, проектирования, моделирования, обучения, диагностики, управления (объектами, процессами, системами).</td></tr>
<tr><td colspan="4">СОЦИАЛЬНАЯ ИНФОРМАТИКА</td><td>Информационные ресурсы как фактор социально–экономического и культурного развития общества. Информационное общество — закономерности и проблемы становления и развития. Информационная инфраструктура общества. Проблемы информационной безопасности. Новые возможности развития личности в информационном обществе. Проблемы демократизации в информационном обществе и пути их решения. Информационная культура и информационная безопасность личности.</td></tr>
</table>

Как видно из схемы, вся предметная область разделена на четыре группы. Программирование как раздел относится к системно–программным средствам автоматизации. Но можно заметить, что все разделы в той или иной мере содержат в себе элементы программирования.

Теоретические основы информатики, теория алгоритмов, математические и информационные модели – во всех этих разделах можно заметить эти элементы: способы записи алгоритмов (последовательности действий), способы описания процессов (модели) на языке понятном ЭВМ.

Технические средства автоматизации, взаимодействие различных устройств ЭВМ друг с другом (создание «микропрограмм», «прошивок», драйверов) – это очередная задача программирования.

Программные средства автоматизации – содержание данного раздела основывается на программных продуктах, который создаются с помощью языков и систем программирования.

Информационные технологии, проектирование, моделирование – это разделы также в значительной степени основаны на программировании.

Для того чтобы обнаружить элементы программирования в разделе «Социальная информатика», достаточно рассмотреть проблемы информационной безопасности и информационной культуры (информационная безопасность, создание различных систем охраны авторских прав; информационная культура, принципы оформления программ, программного кода).

Итак, программирование связано со всеми разделами, составляющими информатику, напрямую или через какой–либо другой раздел.

Программирование является инструментом в руках специалиста информационных технологий, как молоток в руках слесаря. С помощью него можно правильно направить тот или иной информационный процесс, поток. Ведь именно программирование способствует развитию логического, абстрактного, пространственного, мышлений, выработки алгоритмического мышления. При его изучении становится понятным структура готовых программных комплексов, что помогает изучить их работу, понять принцип действия.

Одним из приложений программирования является моделирование. В особенности компьютерное математическое моделирование, оно открывает широкие возможности для осознания связи информатики с математикой и другими науками – естественными и социальными [7]. Моделирование – это мощное орудие, с его помощью можно получать новые знания, объекты, материалы, делать прогнозы, узнавать историю.

Общепринятое определение программирования звучит следующим образом:

Программирование — процесс и искусство создания компьютерных программ и/или программного обеспечения с помощью языков программирования. Программирование сочетает в себе элементы искусства, фундаментальных наук (прежде всего информатика и математика), инженерии, спорта и ремесла.

В узком смысле слова, программирование рассматривается как «Кодирование алгоритмов на заданном языке программирования». Под программированием также может пониматься разработка логической схемы для ПЛИС (программируемая логическая интегральная схема), а также процесс записи информации в ПЗУ. В более широком смысле *программирование*– процесс создания программ, то есть разработка программного обеспечения.

Постоянное совершенствование информационных технологий привело не только к появлению большого количества языковых средств кодирования алгоритмов, но и к довольно четкому выделению четырех основных способов разработки самих алгоритмов. Такие способы в специализированной литературе получили название парадигм. В настоящее время сложилось четыре парадигмы, которые диктуют как подходы к разработке программ на современных языках программирования, так и методы изучения информатики. Это процедурная, объектно-ориентированная, функциональная и логическая парадигмы. Разработаны методы интерпретации языков всех парадигм

друг в друга. Рядом авторов был сделан вывод о приоритете алгоритмической парадигмы, поскольку именно она, поддержанная на аппаратном уровне, является наиболее распространенным типом машин фон Неймана. Однако, благодаря совершенствованию информационных технологий, разработаны и реализованы различные типы машин, соответствующие любому из четырех методов программирования, что, в свою очередь, дает возможность классификации программного обеспечения по четырем ветвям.

Процедурное – (императивное) программирование является отражением архитектуры традиционных ЭВМ, которая была предложена фон Нейманом в 40–х годах. Теоретической моделью процедурного программирования служит алгоритмическая система под названием «машина Тьюринга».

Программа на процедурном языке программирования состоит из последовательности операторов (инструкций), задающих процедуру решения задачи. Основным является оператор присваивания, служащий для изменения содержимого областей памяти. Концепция памяти как хранилища значений, содержимое которого может обновляться операторами программы, является фундаментальной в императивном программировании.

Выполнение программы сводится к последовательному выполнению операторов с целью преобразования исходного состояния памяти, то есть значений исходных данных, в заключительное, то есть в результаты. Таким образом, с точки зрения программиста существуют программа и память, причем первая последовательно обновляет содержимое последней.

Объектно-ориентированное– парадигма программирования, в которой основными концепциями являются понятия объектов и классов. В объектно-ориентированном подходе исходная задача представляется как совокупность взаимодействующих объектов.

Функциональное – парадигма программирования, в которой процесс вычисления трактуется как вычисление значений функций в математическом понимании последних (в отличие от функций как подпрограмм в процедурном программировании). Противопоставляется парадигме императивного программирования, которая описывает процесс вычислений как последовательность изменения состояний (в значении, подобном таковому в теории автоматов). Функциональное программирование не предполагает изменяемость данных (в отличие от императивного, где одной из базовых концепций является переменная).

Логический (декларативный подход) в разработке компьютерных программ появился в начале 70–х годов. Он не получил столь широкого применения как процедурный, поскольку был направлен на относительно узкий круг задач искусственного интеллекта. При его применении программист описывает свойства исходных данных, их взаимосвязи, свойства, которыми должен обладать результат, а не алгоритм получения результата. Разумеется, для получения результата этот алгоритм все равно нужен, но он должен порождаться автоматически той системой, которая поддерживает декларативно–ориентированный язык программирования [9].

Это только основные парадигмы программирования, существуют другие, которые являются как самостоятельными подходами, так и надстройками над известными. Один из наиболее популярных подходов в современном мире объектно-ориентированный.

1.2. Программирование в вузовском курсе информатики

Информатика как наука о способах обработки и хранения информации переживает сейчас ни с чем не сравнимый скачок в своем развитии, следствием которого является постоянная актуальность проблемы совершенствования систем подготовки специалистов в области

обработки информации, смены технической оснащенности, методической и содержательной базы учебного процесса.

Существенным аспектом, влияющим на содержание и методику обучения информатике, является развитие подходов к построению алгоритмов и разработке компьютерных программ.

За прошедшие 40 лет в методологии написания программ для компьютеров произошла радикальная перемена. Это связано с повышением производительности компьютеров и перемены в составе используемого программного обеспечения. Языки программирования создавались для различных целей, что обусловило ряд фундаментальных различий между ними. Так, например, системные языки разрабатывались для построения структур данных и алгоритмов «с нуля», начиная от примитивных элементов. Объектно-ориентированные языки создавались для связывания готовых программ.

У студентов, обучающихся по технологическому или физико-математическому профилю, программирование развивает профессиональную компетенцию.

Как показывает практика, построение учебного процесса на базе только одной парадигмы малоперспективно, поскольку для каждого методического направления есть свой класс «приемлемых задач». Кроме того, понятие алгоритма, например, становится не единственным понятием, лежащим в основе содержания обучения программированию. Наряду с ним появляются такие понятия как «унификация», «управление», характерные для логической парадигмы, «объект», «действие над объектом» – для объектно-ориентированной парадигмы, и «функция», «декомпозиция функций» – для функциональной парадигмы. И, наконец, использование лишь некоторых парадигм в качестве базы для обучения накладывает свой весомый отпечаток на ход мыслительной деятельности студента по составлению алгоритмов. За счет этого, практически отсутствуют специалисты, владеющие всеми четырьмя методами и

способные варьировать подходы к программированию в зависимости от специфики решаемой задачи. Исходя из требований к знаниям и умениям будущих специалистов по информатике, а также с учетом вышесказанного должны быть определены цели и содержание учебного процесса и построена методическая система обучения программированию.

1.3. Цели и задачи дисциплин

В последнее время, основной задачей высшего образования является формирование определенных компетенций у выпускника. Обратимся, к примеру, ко ФГОС ВО, по направлению подготовки 05.03.03 Картография и геоинформатика.[21]. В этом ФГОС можно выделить 2 компетенции, которые можно сформировать, используя этот курс: владением современными геоинформационными и веб-технологиями создания карт, программнымобеспечением в области картографии, геоинформатики и обработки аэрокосмических снимков (ПК-9);владением современнымпрограммным обеспечением в области картографии, геоинформатики(ПК-14).

Существуют и другие направления, где данный курс можно задействовать: ФГОС ВО, по направлению подготовки09.03.02 Информационные системы и технологии.[21].Выпускник, освоивший программу бакалавриата, должен обладать профессиональными компетенциями, соответствующими виду (видам) профессиональной деятельности, на который (которые) ориентирована программа бакалавриата: проектно-конструкторская деятельность: способностью проводить предпроектное обследование объекта проектирования, системный анализ предметной области, их взаимосвязей (ПК-1); способностью проводить техническое проектирование (ПК-2); способностью проводить моделирование процессов и систем (ПК-5); проектно-технологическая деятельность: способностью к проектированию

базовых и прикладных информационных технологий (ПК-11); способностью разрабатывать средства реализации информационных технологий (методические, информационные, математические, алгоритмические, технические и программные) (ПК-12); способностью разрабатывать средства автоматизированного проектирования информационных технологий (ПК-13).[24].

В процессе развития информатики как прикладной науки появились разные подходы к программированию. Курсы для изучения визуального программирования призваны содействовать знакомству студентов с различными парадигмами проектирования и разработки программного обеспечения. Они важны с той точки зрения, что, являются составной частью подготовки специалиста и бакалавра соответствующего профиля, способствуют развитию алгоритмического мышления, навыков программирования.

Цель дисциплин: изучение методов программирования для овладения знаниями в области технологии программирования; подготовка к осознанному использованию как языков программирования, так и методов программирования, развитие таких умений, как решать образовательные и исследовательские задачи, анализ научной и учебно-научной литературы, использования современных технологий сбора и обработки информации, осуществлять обучение и воспитание обучающихся.

Воспитательной целью дисциплин является формирование у студентов научного, творческого подхода к освоению технологий, методов и средств производства программного обеспечения.

Основные задачи курса программирования на основе структурного и объектно-ориентированного подхода:

- знакомство с методами объектно-ориентированного программирования как наиболее распространенными и эффективными методами разработки программных продуктов;

- обучение разработке алгоритмов на основе объектно-ориентированного подхода;
- закрепление навыков алгоритмизации и программирования на основе изучения среды VisualBasic;
- знакомство с основными объектами, классами, возможностями объектно-ориентированного программирования на основе VisualBasic.

Отбор материала основывается на необходимости ознакомить студентов со следующей современной научной информацией:

- парадигмы программирования (императивной, функциональной, логической);
- технологии программирования (структурной, модульной, объектно-ориентированной);
- аспекты формализации синтаксиса и семантики языков программирования.

Изучение визуального программирования базируется на знании математических дисциплин и общего курса информатики.

Концепция дисциплины основана на том, что она имеет общеобразовательный и в определенной степени мировоззренческий характер и предназначена для формирования человека с широким научным кругозором.

В качестве подробного примера рассмотрим содержание курса Языки и методы программирования

В результате изучения дисциплины студент должен:

- иметь представление:
  - о теоретических основах объектно-ориентированного анализа, проектирования и программирования,
  - методах и технологиях программирования в объектно-ориентированных программных средах;

- формализации синтаксиса и семантики языков программирования

– уметь:

- разрабатывать новые компоненты на основе своих либо существующих классов,
- разрабатывать приложения, работающие под управлением Windows,
- разрабатывать приложения работы с базами данных с использованием технологий BDE и ADO;

– приобрести навыки:

- в разработке новых компонентов,
- разработке приложений, работающих под Windows,
- разработке приложений, для управления работой баз данных;
- владеть, иметь опыт:
- работы в средах программирования,
- в разработке новых компонентов,
- разработке приложений, работающих под Windows,
- разработке приложений, для управления работой баз данных.[22]

Глава 2 ДИДАКТИЧЕСКОЕ И МЕТОДИЧЕСКОЕ ОБЕСПЕЧЕНИЕ КУРСОВ ВИЗУАЛЬНОГО ПРОГРАММИРОВАНИЯ

2.1. Что такое Visual Basic?

Visual Basic – это последняя версия одного из популярных языков программирования. В настоящее время с помощью Visual Basic можно быстро создавать приложения, работающие в среде Windows для любой области компьютерных технологий: бизнес-приложения, мультимедиа, приложения типа клиент – сервер и приложения управления базами данных. Кроме того, Visual Basic является встроенным языком для приложений Microsoft Office. Многие разработчики приложений также используют Visual Basic в качестве внутреннего языка своих приложений. [18]

Visual Basic представляет собой интегрированную среду разработки, которая содержит набор инструментов, облегчающих и ускоряющих процесс разработки приложений. Причем процесс разработки заключается не в написании программы (программного кода), а в проектировании приложения. Приложение формируется средствами графического редактирования (компоновки), что позволяет свести процесс создания программного кода к минимуму. [11].

Как и во всех современных системах визуального проектирования, в Visual Basic применяется объектно-ориентированный подход к программированию. Любое приложение, написанное на Visual Basic, представляет собой совокупность объектов.

Объект – некая сущность, которая четко проявляет свое поведение и является представителем некоторого класса подобных себе объектов. Почти все, с чем производится работа в VB, является объектами. Например: Форма, Командная кнопка, Текстовое поле и т. д.

Каждый объект характеризуется:

- свойствами;
- методами;
- событиями.

Свойство – это имеющий имя атрибут объекта. Свойства определяют характеристики объекта (цвет, положение на экране, состояние объекта).

Методы – это действия или задачи, которые выполняет объект (то, что можно делать с объектами).

Классом объектов в объектно-ориентированных языках программирования называется общее описание таких объектов, для которых характерно наличие множества общих свойств и общих действий, которые способны выполнять эти объекты (например, класс Командная кнопка – общее описание кнопок в окнах приложений). Они должны иметь множество общих свойств и других характеристик (например, событий, одинаковых для всех этих объектов: щелчок мышью).

Приложение, создаваемое в среде Visual Basic, называется проектом. *Программный проект*– это совокупность частей, составляющих будущее WINDOWS-приложение. Любой проект должен обязательно состоять из экранных форм (хотя бы одной) и программных модулей (хотя бы одного). Visual Basic хранит каждый проект в отдельном файле с расширением vbp.

Экранная форма– это графическое представление WINDOWS-приложения вместе с содержанием этого окна. Содержание включает в себя:

- совокупность свойств этого окна с их значениями;
- совокупность, объектов, находящихся в этом окне;
- совокупность свойств этих объектов с их значениями.

В Visual Basic экранная форма хранится в отдельном файле с расширением frm.

Программный модуль– это хранящийся в отдельном файле программный код (текст некоторой программы). Он может использоваться

при решении чаще всего одной, а иногда и нескольких задач. Имя этого файла имеет расширение bas.

Программный код проекта существует не сам по себе, он привязан к отдельным объектам экранной формы. Часть кода, которая относится только к одному объекту, в свою очередь может состоять из нескольких фрагментов-процедур [7].

В Visual Basic программный код почти всегда привязывается к какому-либо событию, которое является сигналом к началу работы программы. Например, щелчок мыши по какому-либо объекту экранной формы; загрузка новой экранной формы; перемещение указателя мыши вдоль полосы прокрутки; нажатие какой-либо клавиши на клавиатуре.

Сначала проектируется экранная форма, затем устанавливаются события, которые будут происходить в работающем приложении, и только затем программируются действия, связанные с этими событиями.

Событие – это характеристика класса объекта, описывающая внешнее воздействие, на которое реагирует объект этого класса во время работы приложения.

Большинство процедур, из которых состоит программный код VB, привязаны к событиям и называются процедурами-событиями.

Создание любого приложения состоит из следующих этапов:

1. Постановка задачи. Описание принципа работы будущего приложения, видов экранных форм (окон) этого приложения.

2. Разработка интерфейса. Создание экранных форм приложения со всеми находящимися на этих формах объектами и свойствами этих объектов.

3. Программирование. Определение того, какие события будут происходить в процессе работы приложения, составление алгоритмов процедур для этих событий и написание программы (программных кодов) этих процедур.

4. Отладка программы. Устранение логических ошибок в процедурах и достижение того, чтобы приложение работало удовлетворительно в среде проектирования.

5. Сохранение проекта и при желании – компиляция (превращение проекта в исполняемое приложение, способное работать самостоятельно за пределами среды проектирования).

Приложение может работать в режиме компиляции или интерпретации. В режиме интерпретации все инструкции на языке БЭЙСИК, из которых состоит программа, будут выполняться системой Visual Basic непосредственно в процессе их чтения компьютером строка за строкой в среде разработки.

В режиме компиляции сначала производится отладка программы с помощью интерпретатора, затем она полностью транслируется (переводится) с языка высокого уровня (Бейсика) на язык низкого уровня (язык машинных команд компьютера), т. е. компилирует.

Скомпилированная программа помещается в файл с расширением exe. Этот файл может быть запущен на исполнение самостоятельно, без поддержки среды Visual Basic.

2.2. Пользовательские типы данных

Пользовательские типы данных можно представить, как разделенную на части переменную, неоднократно применяемую в программе. Для этого создают «пакет» переменных и присваивают ему подходящее имя, а затем формируют *экземпляры* этого пакета аналогично созданию переменных, указывая, вместо стандартного типа данных, пользовательский тип. [9]. Пакет описывают с помощью ключевого слова «*Type*» в разделе «*General*» модуля:

```
[Dim|Private|Public] Type TypeName
ElementName As DataType
   ...
End Type
```

- *Dim*|*Private*|*Public* – ключевое слово, определяющее область видимости пользовательского типа данных;
- *Type* – ключевое слово VisualBasic, означающее начало блока определения типа данных;
- *TypeName* – имя, назначаемое пользовательскому типу данных;
- *ElementAsDataType* – строки объявления элементов, из которых образован тип;
- *ElementName* – имя конкретного элемента в структуре типа данных;
- *DataType* – один из существующих стандартных типов;
- *EndType* – конец блока описания пользовательского типа данных.

Ссылки на элементы пользовательского типа данных осуществляются с помощью следующего синтаксиса:

- *VarName.ElementName*
- *VarName* – имя, которое было назначено переменной (экземпляру пользовательского типа данных);

- *ElementName* – имя конкретного элемента в структуре типа данных.

Visual Basic позволяет формировать на основе этих типов также и массивы, не отличающиеся от обычных. Ссылки на ячейки подобных массивов осуществляются следующим образом:

```
ArrayName(Number).ElementName
```

- *ArrayName* – имямассива.
- *Number* – номер (номера) нужной ячейки массива.
- *ElementName* – имя конкретного элемента в структуре типа данных.

Для подобных массивов справедливы все методы работы, что и для обычных: динамические и статические массивы, переопределение, реинициализация и др. [10]

2.3. Создание классов и объектов

На языке объектно-ориентированного программирования элемент данных называется *свойством*, а процедура – *методом*. Возможно, хотя и редко, использование соответственно терминов *атрибут* и *поведение*.

Класс подобен пользовательскому типу, но, кроме данных, класс содержит процедуры и функции, которые запускаются каждый раз, когда происходит запись или считывание информации в соответствующее свойство. *Класс* – это набор методов и свойств, а *объект* – это экземпляр класса.

Строительным блоком класса является *модуль класса*. Для его добавления в проект необходимо выбрать команду «*AddClassModule*» из меню «*Project*» и в появившемся окне щелкнуть на кнопку «*Ok*». Свойству «*Name*» модуля присвоить подходящее имя, которое и есть имя класса. [11]

Второй шаг – создание необходимых свойств класса. Для этого можно воспользоваться диалоговым окном «*AddProcedure*», но переключатель «*Type*» необходимо установить в положение «*Property*» (свойство). Имя, указанное в поле «*Name*», будет имя свойства. Visual Basic добавит в окно модуля класса две процедуры:

```
Public Property Get Name() DataType
        ...
   Name = Total
End Property
```

- *Public* – область видимости свойства; поскольку свойство вызывается из другого модуля, то оно должно всегда должно быть «Public».
- *Property* – зарезервированное слово, означающее описание свойства класса.
- *Get* – зарезервированное слово, описывающее событие считывания данных из свойства.

- *Name* – имясвойстваклacса.
- *DataType* – тип возвращаемых свойством данных.
- *Total* – некоторая заранее вычисленная переменная, итоговое значение.
- *EndProperty* – конец блока описания свойства.

```
Public Property Let Name(ByValvNewValue As DataType)
  ...
End Property
```

- *Let* – зарезервированное слово, описывающее событие записи данных в свойство.
- *Name* – имясвойстваклacса.
- *ByValvNewValueAsDataType* – описание переменной *«vNewValue»* в которую будет помещено записываемое в свойство значение.

Каждое свойство всегда имеет пару таких процедур с соответствующими именами. Принцип их работы состоит в следующем. При записи некоторой информации в свойство запускается процедура «*Let*», которая помещает это значение в предопределенную локальную переменную «*vNewValue*». Ее можно использовать в дальнейшем для необходимых вычислений. При считывании информации из свойства выполняется процедура «*Get*» (на самом деле функция), которая присваивает вычисленное заранее значение этому свойству. [11]

Свойство можно добавить в модуль, создав в нем простую Public переменную. В этом случае запись или считывание значения не будет запускать внутренние структуры объекта. [8]

Другим способом создания внешнего интерфейса модуля класса – создание *методов*. Создаются они путем добавления в класс «*Public*» процедуры. Их можно вызывать из внешней программы, как и обычные

свойства. Создается такая процедура обычным способом (см. раздел «Создание повторно используемого кода»), но только с помощью зарезервированного слова «*Public*»:

```
Public Sub NameMetod(<списокпараметров>)
```

- *Public* – зарезервированноеслово.
- *Sub* – зарезервированное слово, объявляющее процедуру.
- *NameMetod* – имясоздаваемогометода.
- *<список параметров>* – при необходимости добавляются нужные параметры, для передачи значений в модуль.

Принцип действия такого метода полностью аналогичен работе обычной процедуры. Благодаря наличию списка аргументов, можно передавать в модуль класса некоторые значения. Свойства аргументов (необязательные и именованные) полностью аналогичны таковым для обычных подпрограмм.[15]

Помимо их в модуле класса могут присутствовать и другие процедуры (*функции-члены*) и переменные (*переменные-члены*) типа «*Private*», которые и составляют внутренние механизмы по переработке информации, находящейся в свойствах. Такая «*Public/Private-схема*» и называется инкапсуляцией, поскольку программа никогда не обращается к «*Private*» ресурсам класса непосредственно, а использует для этого «*Public*» свойства и методы.

До появления *VisualBasic 5.0* не существовало возможности добавления новых событий. Чтобы добавить новое событие в необходимый модуль класса, необходимо выполнить следующие действия:

1. Объявить открытое (*Public*) событие в разделе объявлений (*General*) нужного модуля класса:

```
Public Event Name(ByVal<списокпеременных>)
```

- *Event* – зарезервированное слово, определяющее описание события.
- *Name* – имясобытия.
- *<список переменных>* – список аргументов, возвращаемых событием; необязательный параметр.

2. Внутри модуля класса в нужных его местах введите оператор «*RaiseEvent*» для вызова необходимого события:

```
RaiseEventName(<списокпеременных>)
```

- *RaiseEvent* – операторвызовасобытия.
- *Name* – имявызываемогособытия.
- *<список переменных>* – список значений, которые будут возвращены событием.

Экземпляр созданного класса можно использовать в программе в виде объекта. Непосредственно его использовать нельзя. Объект с необходимым именем может создаваться двумя способами.[2]

Первый способ, самый простой, создает модуль в соответствии со следующим синтаксисом:

```
Dim|Public|PrivateNameObject As New NameClass
```

- *Dim|Public|Private* – ключевые слова, определяющие область видимости объекта в приложении.
- *NameObject* – имяобъекта.
- *As* – ключевое слово для определения типа.
- *New* – ключевое слово, с помощью которого создается экземпляр объекта.
- *NameClass* – имя класса, из которого создается экземпляр объекта.

После этого объект можно использовать в программе, устанавливая или считывая значения его свойств соответственно:

- Запись значения в свойство:

```
NameObject.NameProperty = MyVar
```

- Считывание значения из свойства:

```
MyVar = NameObject.NameProperty
```

- *NameObject* – имяобъекта.
- *NameProperty* – имясвойстваобъекта.
- *MyVar* – промежуточнаяпеременная.

Аналогично можно использовать созданные вами методы. Их вызов похож на вызов обычных процедур:

```
NameObject.NameMetod<список значений аргументов>
```

- *NameObject* – имяобъекта.
- *NameMetod* – имясвойстваобъекта.
- *<список значений аргументов>* – некоторые значения, которые будут переданы методу (процедуре). [4]

Особенность такого способа состоит в том, что в приложении не будут доступны события этого модуля. Для его исправления создают модуль иным способом:

1. В нужной форме объявите переменную типа модуля класса, используя ключевое слово «*WithEvents*», чтобы сообщить, что событие будет происходить для данного модуля:

```
Private WithEventsobjVar As NameClass
```

- *Private* – областьвидимостипеременной.
- *objVar* – имя переменной модуля класса.
- *NameClass* – имямодулякласса.[6]

2. В событии «*Form_Load*» создайте экземпляр класса с помощью следующего оператора:

```
Set objVar = New NameClass
```

Модульбудетсоздан. Для того чтобы событие обрабатывалось в приложении, необходимо ввести соответствующую процедуру обработки этого события, синтаксис которой аналогичен общепринятым подобным процедурам:

```
Private Sub objVar_Name(ByVal<списокпеременных>)
```

- *Private* – область видимости процедуры обработки события.
- *objVar* – имя переменной модуля класса.
- *Name* – имясоответствующегособытия.
- *<список переменных>* – список возвращаемых процедурой значений (при наличии их в объявлении).

Использование свойств и методов в этом случае подобно первому случаю, но необходимо указывать в качестве имени модуля соответствующую переменную модуля класса:

```
objVar.NameProperty = MyVar
```
и т.д.

Главное достоинство подобного вида программирования – создание инкапсулированных объектов (т.е. переработка информации осуществляется исключительно за счет внутренних структур), что дает возможность из них формировать специальные «*DLL*» библиотеки, которые можно использовать в любых языках программирования, поддерживающих объектно-ориентированное программирование, поскольку программный код в ней уже откомпилирован в машинные коды [7].

2.4. Основные компоненты

Задачи данного раздела демонстрируют примеры оформления интерфейса, внешнего вида программы, они предназначены для изучения свойств компонентов, обеспечивающих ввод–вывод информации. Задачи первого, второго блоков сопровождаются картинками позволяющие наглядно продемонстрировать, варианты оформления интерфейса, принцип написания программ на VB.

Все задачи этого раздела содержат несколько основных компонентов, обеспечивающих интерфейс программы, что обеспечивает изучение компонентов одновременно, показывает их взаимодействие друг с другом.

Первоначально предлагается к рассмотрению задачи с готовым решением, для знакомства с программированием в Visual Basic. Данные задачи демонстрируют работу таких компонентов как Label, TextBox, OptionButton, CommandButton, CheckBox, ComboBox, Timer, PictureBox.

Задача 1. Создать приложение для расчета пройденного расстояния по известным величинам скорости и времени.

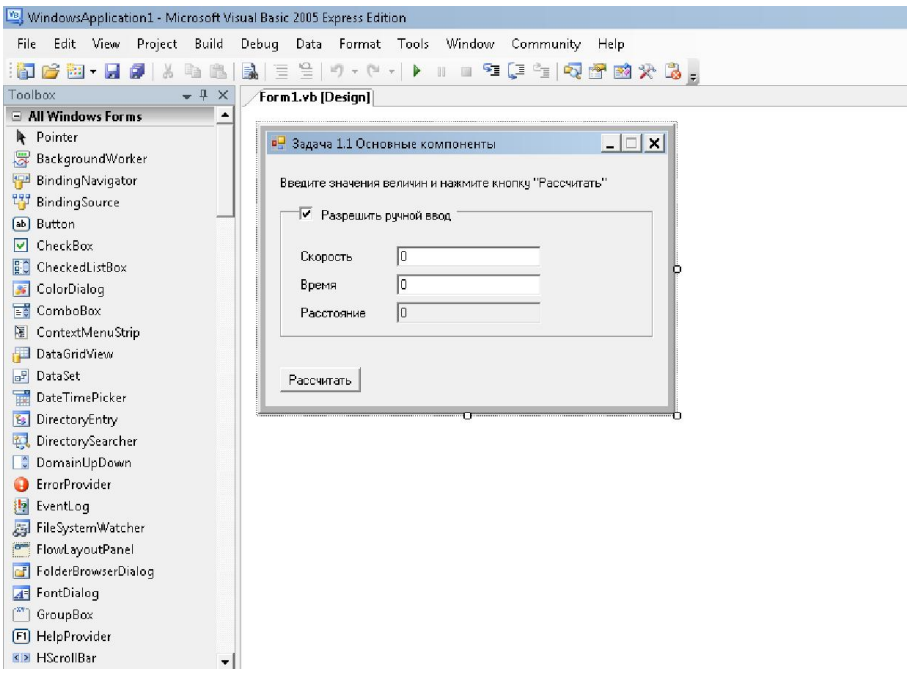

Рис.1. Внешний вид приложения

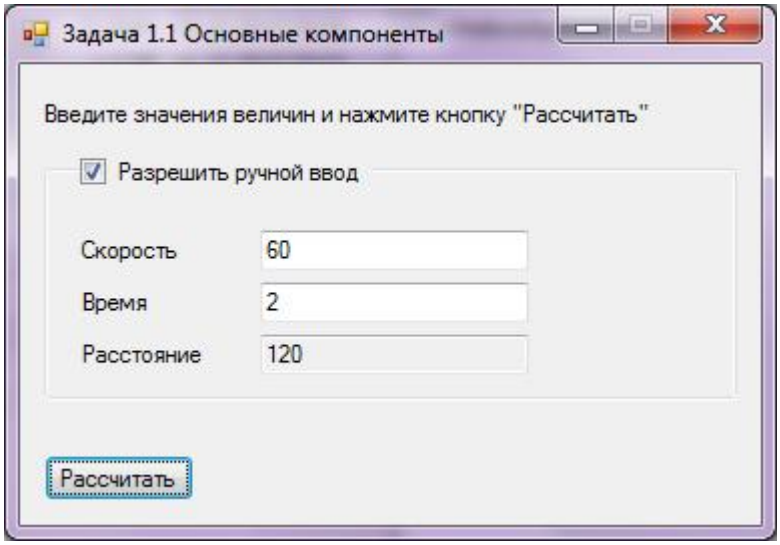

Рис. 2. Работающее приложение

Решение:

```
PublicClassForm1
Private Sub CheckBox1_CheckedChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles CheckBox1.CheckedChanged
'определяем доступность элементов в зависимости от флажка
TextBox1.Enabled = CheckBox1.Checked
        TextBox2.Enabled = CheckBox1.Checked
        TextBox3.Enabled = CheckBox1.Checked
    End Sub
```

```
        Private Sub Button1_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
Button1.Click
    'описываем переменные
            Dim S, V, T As Integer
    считываем значения текстовых полей в переменные
    V = Val(TextBox1.Text)
            T = Val(TextBox2.Text)
    'рассчитываем формулу
    S = V * T
    'устанавливаем в текстовое поле посчитанное значение из
переменной
    TextBox3.Text = S
        End Sub
    End Class
```

Задача 2. Создать приложение для приветствия пользователя по введенному имени.

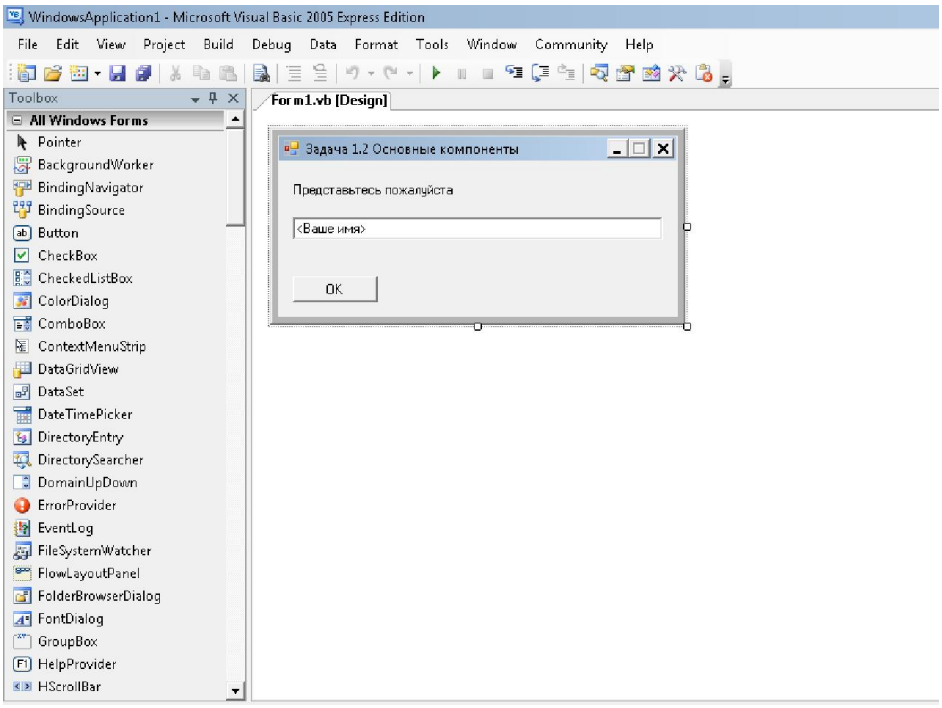

Рис. 3. Внешний вид приложения

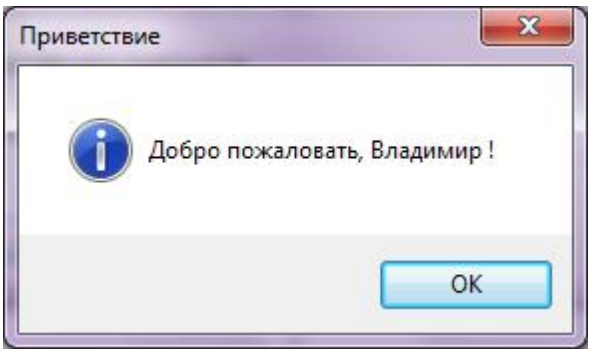

Рис.4. Информационное окно

Решение:

```
PublicClassForm1
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
'описываем переменные
DimSAsString
'считываем содержимое текстового поля в переменную
S = TextBox1.Text
'выводим сообщение с приветствием
MsgBox("Добро пожаловать, " &S& " !", MsgBoxStyle.Information, "Приветствие")
Close()
    End Sub
End Class
```

Задача 3.Создать приложение для тестирования. Пользователю предлагается вопрос и варианты ответа.

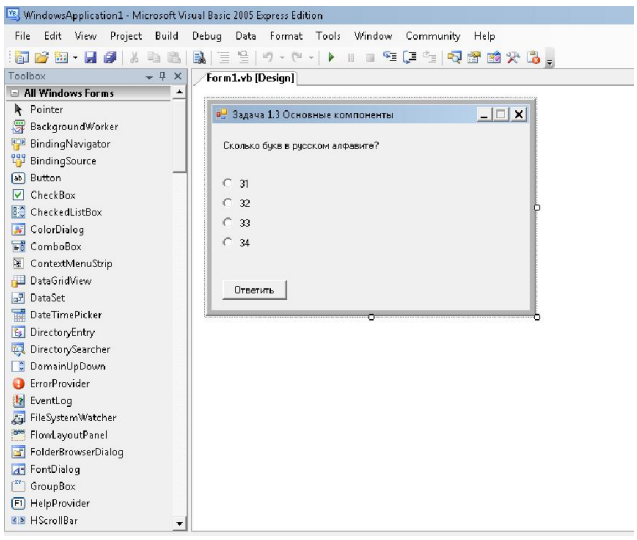

Рис.5. Внешний вид приложения

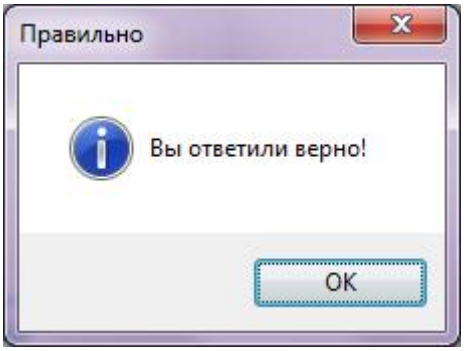

Рис.6. Информационное окно о правильном ответе

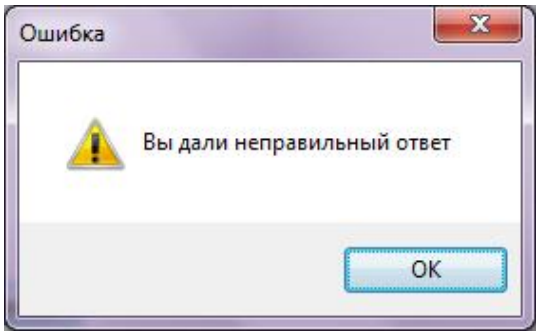

Рис.7. Информационное окно о неправильном ответе

Решение:

```
    Public Class Form1
        Private Sub Button1_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
Button1.Click
    'если установлен третий флажок, то...
    IfRadioButton3.CheckedThen
    'выводим сообщение о правильном ответе
    MsgBox("Вы ответили верно!", MsgBoxStyle.Information,
"Правильно")
    Else'...иначе
    'выводим сообщение о неправильном ответе
    MsgBox("Вы дали неправильный ответ",
MsgBoxStyle.Exclamation, "Ошибка")
    EndIf
    'закрыть форму и выйти из программы
    Close()
        End Sub
    End Class
```

2.5. Графика

Следующим разделом будет графика. Любое современное программное приложение уделяет большое место внешнему виду, графическому дизайну. Одним из способов украшения своего приложения, может быть создание на его форме каких-либо рисунков с помощью анимации. Рассмотрение данного раздела предлагается начать с примитивов графики: первая задача предлагает студенту нарисоватьпростейшее изображение. Составляющие этого изображения будут один прямоугольник, и нескольктреугольников, которые выводятся на форму при возникновении события MsPaint.

Задача 1. Создать приложение вывода изображения на форму.

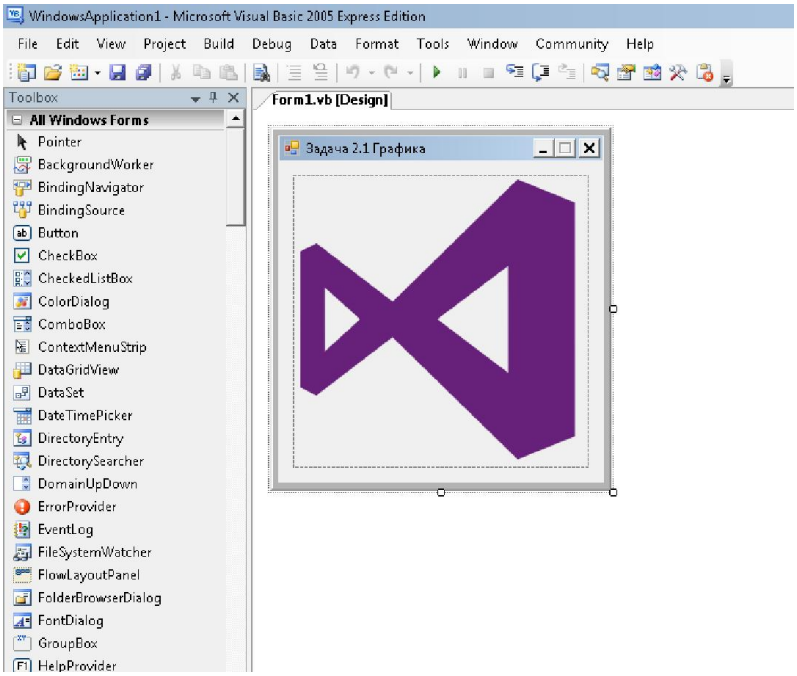

Рис. 8. Внешний вид приложения

Решение:

Компонент **`PictureBox`**

Задача 2. Создать приложения на форме которого можно рисовать указателем мыши как в MSPaint.

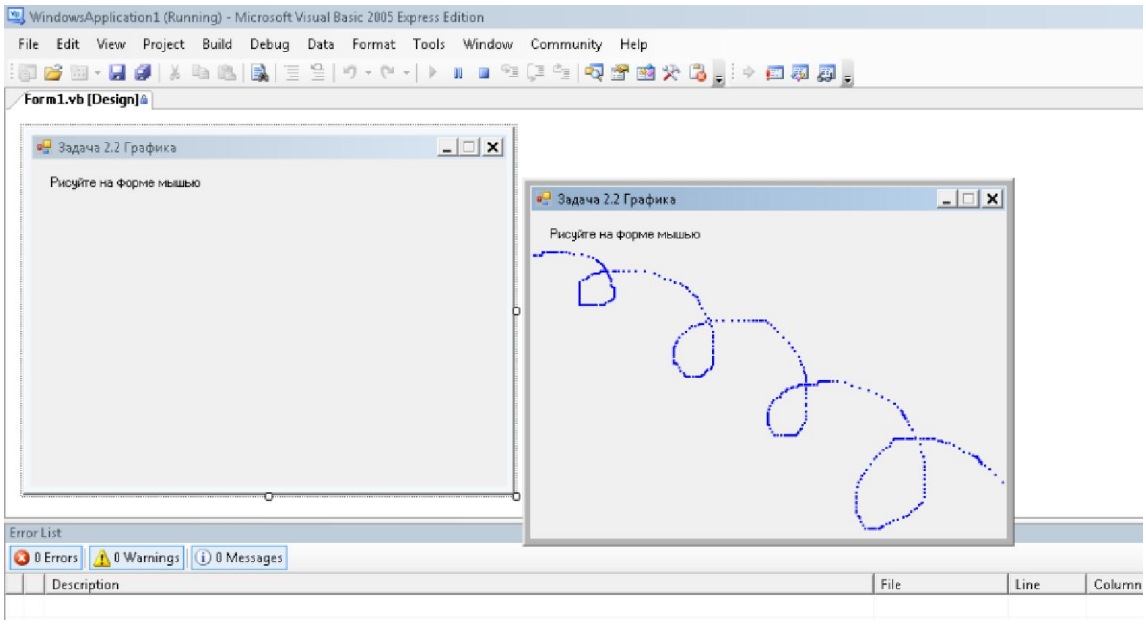

Рис.9. Внешний вид приложения

Решение:

```
Public Class Form1

    Private Sub Form1_MouseMove(ByVal sender As
System.Object, ByVal e As System.Windows.Forms.MouseEventArgs)
Handles MyBase.MouseMove
    'объявляем переменную
    DimGrapAsGraphics
    'создаем объект
    Grap = Me.CreateGraphics
    'рисуем маленький прямоугольник размером 1x1. В сумме
ширина будет 2x2
    Grap.DrawRectangle(Pens.Blue, e.X, e.Y, 1, 1)
    End Sub
    End Class
```

Задача 3.Создать приложение воспроизводящей покадровую анимацию.

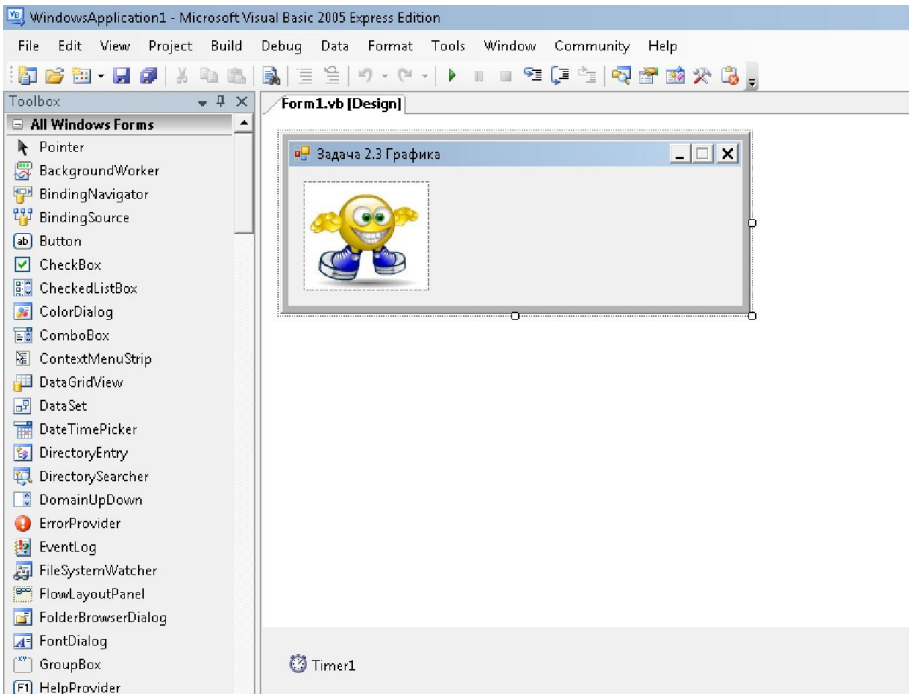

Рис. 10. Внешний вид приложения

Решение:

```
    Public Class Form1
        Private Sub Timer1_Tick(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Timer1.Tick
    'сначаласкрываемвсеспрайты
    PictureBox1.Hide()
    PictureBox2.Hide()
            PictureBox3.Hide()
    PictureBox4.Hide()
    PictureBox5.Hide()
    PictureBox6.Hide()
    'текущий спрайт увеличиваем на единицу и храним во
встроенной переменной Tag
    Tag = Tag + 1
    'если номер текущего спрайта больше максимального, то
установить значение на первый спрайт
    IfTag> 6 ThenTag = 1
    'в зависимости от номера текущего спрайта выводим
соответствующее изображение
    Select Case Tag
                Case 1
                    PictureBox1.Show()
```

```
            Case 2
                PictureBox2.Show()
            Case 3
                PictureBox3.Show()
            Case 4
                PictureBox4.Show()
            Case 5
                PictureBox5.Show()
            Case 6
                PictureBox6.Show()
        End Select
    End Sub
End Class
```

2.6. Файлы

Этот раздел, один из наиболее важных в программировании, так как рано или поздно придется столкнуться с необходимостью долговременного хранения полученной информации. Сохранение информации происходит в файлах. Этот раздел посвящен работе с файлами, каталогами и обработке исключительных ситуаций.

Что бы начать изучение файлов, мы предлагаем для начала изучить обработку исключительных ситуаций. Для этого студентам предложены задачи, которые помещены в отдельный раздел, потому что последующие задачи должны сопровождаться перехватом исключительных ситуаций, возникающих при написании программ.

Первыми предложенными действиями над файлами является создание простейшего файла с добавочной информацией.

Задача 1. Создать приложение, которое создает текстовый файл и записывает в него строку "HelloWorld".

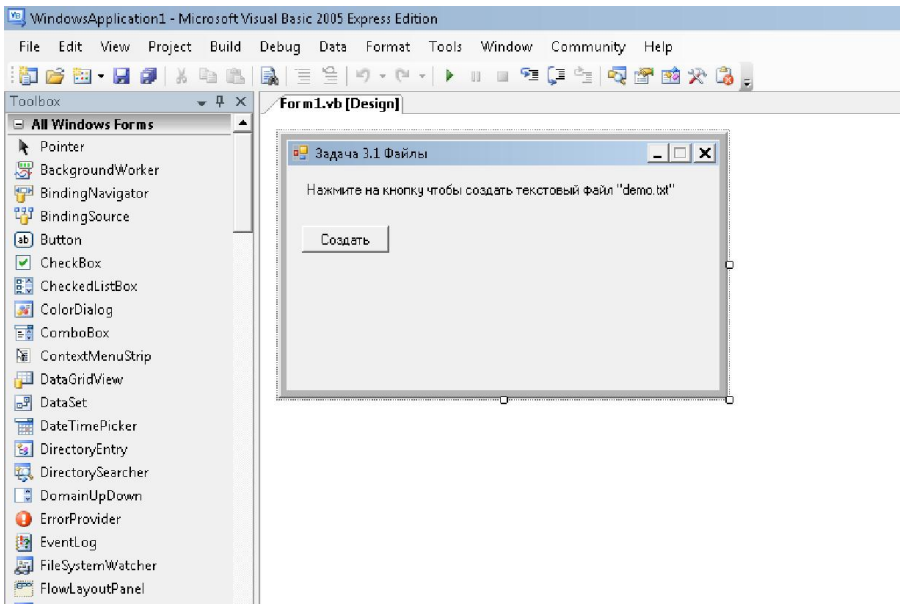

Рис.10. Внешний вид приложения

Решение:

```
Imports System.IO
Public Class Form1
    Private Sub Button1_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
Button1.Click
'объявляем переменные
DimFAsStreamWriter
'создаем файл
F = New StreamWriter("Demo.txt")
'записываем в него строку
F.WriteLine("HelloWorld !")
'закрываем файл
F.Close()
    End Sub
End Classъ
```

Задача 2. Создать приложение, удаляющая в текущем каталоге файлы с расширением "tmp".

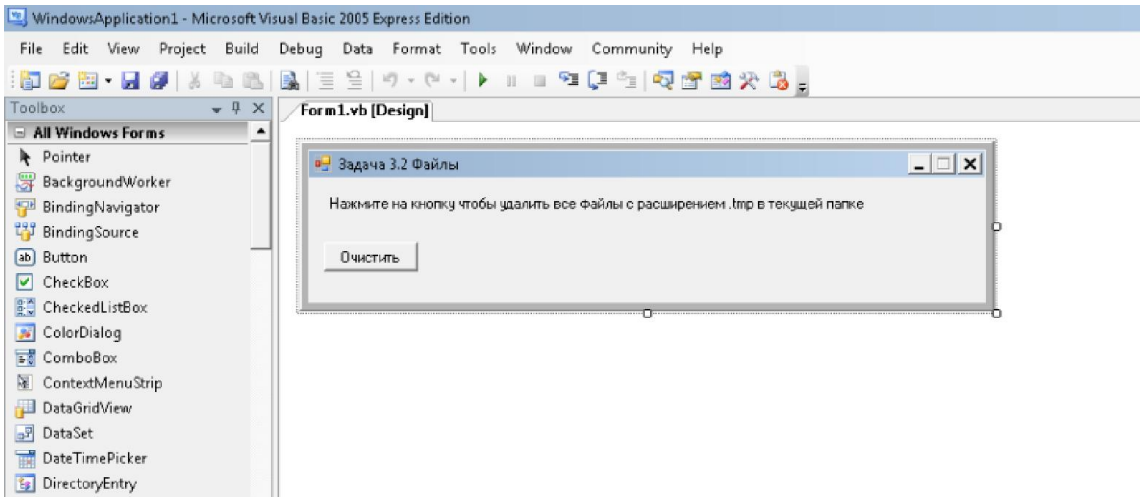

Рис.11. Внешний вид приложения

Решение:

```
Public Class Form1
    Private Sub Button1_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
Button1.Click
'удаляем файлы по заданной маске
Kill("*.tmp")
EndSub
End Class
```

Структура этого раздела аналогично предыдущему, и рассматривать его можно точно так же, правда с небольшим исключением. Все задачи на разделе «файлы» сопровождаются многочисленными ошибками записи, чтения файла, существования файла, поэтому первоначально необходимо полностью изучить блок исключительные ситуации, который позволит в дальнейшем уменьшить вероятность некорректного завершения приложения.

2.7. База данных

Хранение информации в файлах можно организовать по–разному, одним из способов является хранение в виде баз данных. VB предоставляет широкие возможности для создания и работы с базами данных. В изучении баз данных, задачи разделяют на следующие группы: основные компоненты для работы с базами данных; основы языка SQL; организация БД драйверами BDE; организация БД драйверами ADO.

Задача 1. Создать приложение отображающий на форме содержимое таблицы из файла Access.

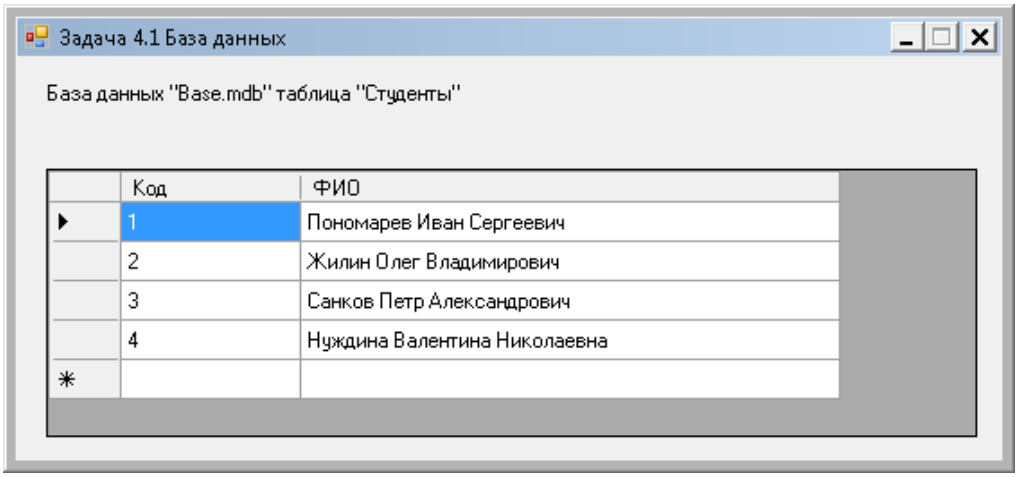

Рис.12. Внешний вид приложения

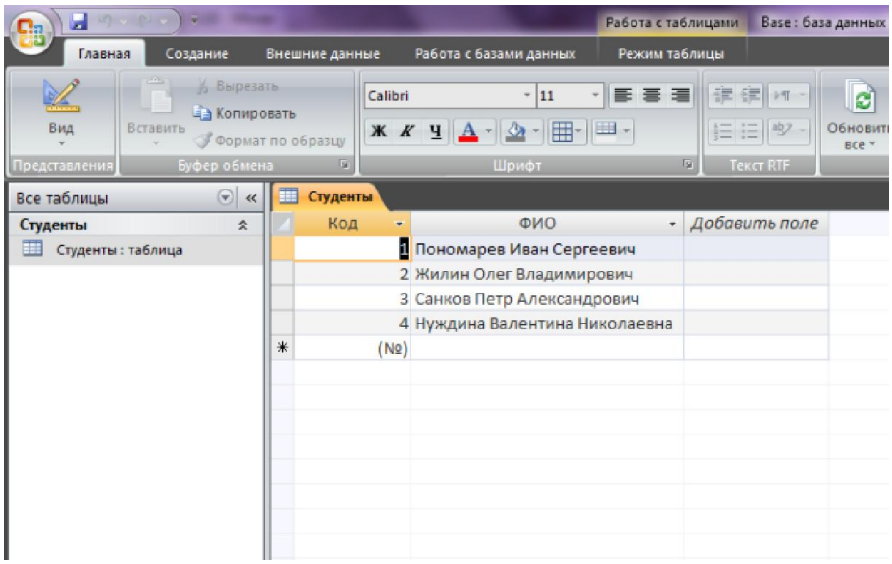

Рис.13. База данных

Решение:

Компонент **DataGridView.**

```
Option Explicit
Public KeyWord As String 'параметр запроса
    Private Sub Adodc1_Error(ByVal ErrorNumber As Long,
_Description As String, ByVal Scode As Long,
     ByVal Source As String, ByVal HelpFile As String, _ ByVal
HelpContext As Long, fCancelDisplay As Boolean)
Dim msg As String 'сообщение об ошибке
     msg = "Ошибка: " + Str(ErrorNumber) + vbCr + Description
MsgBox msg, vbCritical, "Студенты"
fCancelDisplay = True'не отображать стандартное сообщение об
ошибке
End Sub
    Private Sub Form_Initialize()
Form1.ScaleMode = vbPixels
установить ширину столбцов
DataGrid1.Columns(0).Width = 270
End Sub
```

Задача 2. Создать приложение отображающий на форме содержимое таблицы из файла Access.

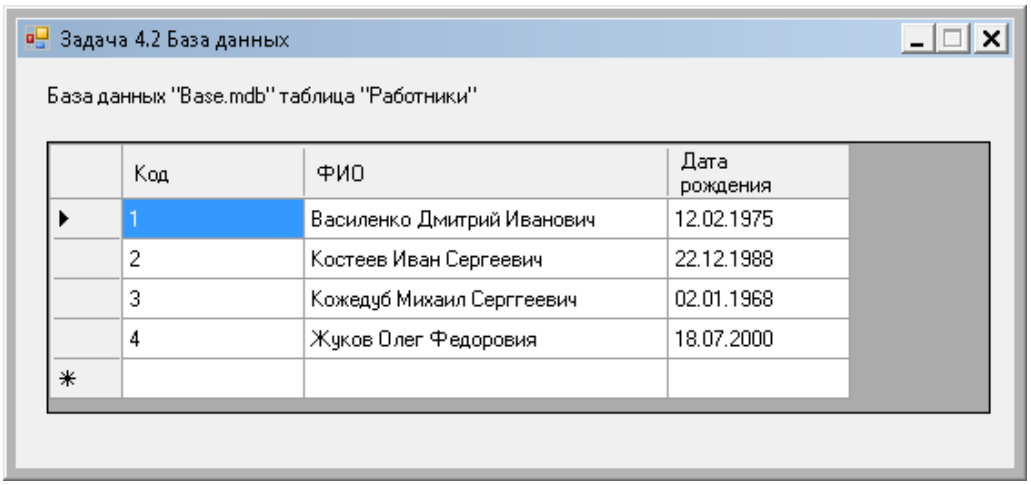

Рис.14. Внешний вид приложения

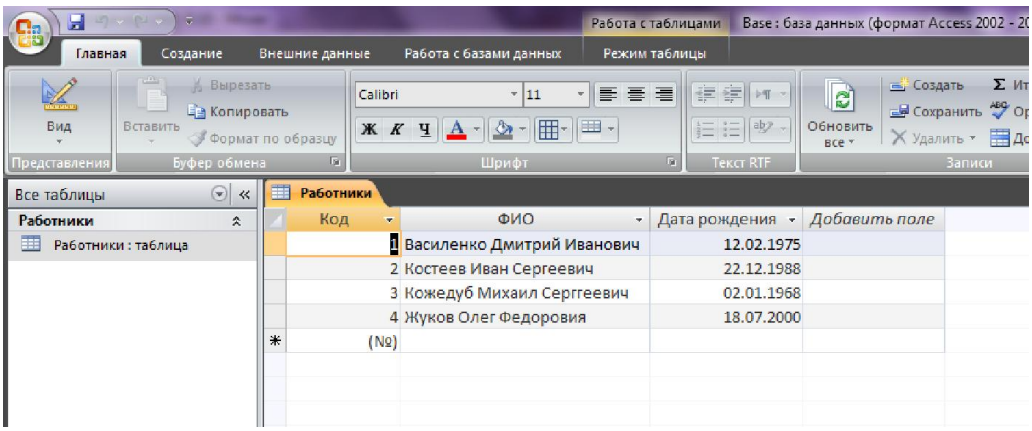

Рис.15. База данных

Решение:

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs)
Handles Button1.Click
        Dim z As New OleDb.OleDbConnection
("Provider=Microsoft.Jet.OLEDB.4.0;Data    Source=C:1Приложение
Microsoft  Office  Access.mdb")  'Устанавливаем  соединение  с
базой данных.
        z.Open() 'Открываем соединение.
        Dim t As New OleDb.OleDbCommand("UPDATE [Таблица1] SET
[Поле1] = ' " & " " & TextBox1.Text & " '  WHERE [Код] = 1",
z)  'Иначе  можно  было  записать  так     Dim  t  As  New
OleDb.OleDbCommand("UPDATE  [Таблица1]  SET  [Поле1]  =  'Нужное
значение'  WHERE [Код] = 1", z)
        t.ExecuteNonQuery()
        z.Close() 'Закрываем соединение
    End Sub
```

2.8. Сетевые компоненты

Компьютер не представлял бы большой ценности, если не было бы возможности обмена информации между компьютерами. Данную проблему можно легко решить, используя сетевые приложения. Данный раздел показывает, как быстро и легко создать небольшие сетевые приложения.

Задача 1. Создать приложение открывающий заданную веб-страницу

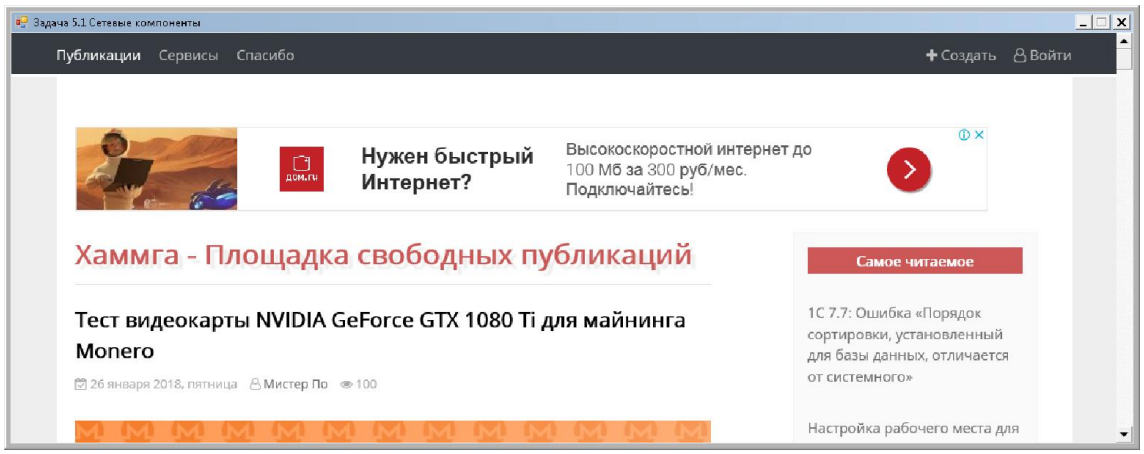

Рис.16. Внешний вид приложения

Решение:

```
Dim objShell
Set objShell = CreateObject("Shell.Application")

iURL = "www.google.com"

Call objShell.ShellExecute("C:\Program Files
(x86)\Google\Chrome\Application\chrome.exe", iURL, "", "", 1)
```

Задача 2. Создать приложение показывающий информацию о сетевых адаптерах, установленных в операционном системе.

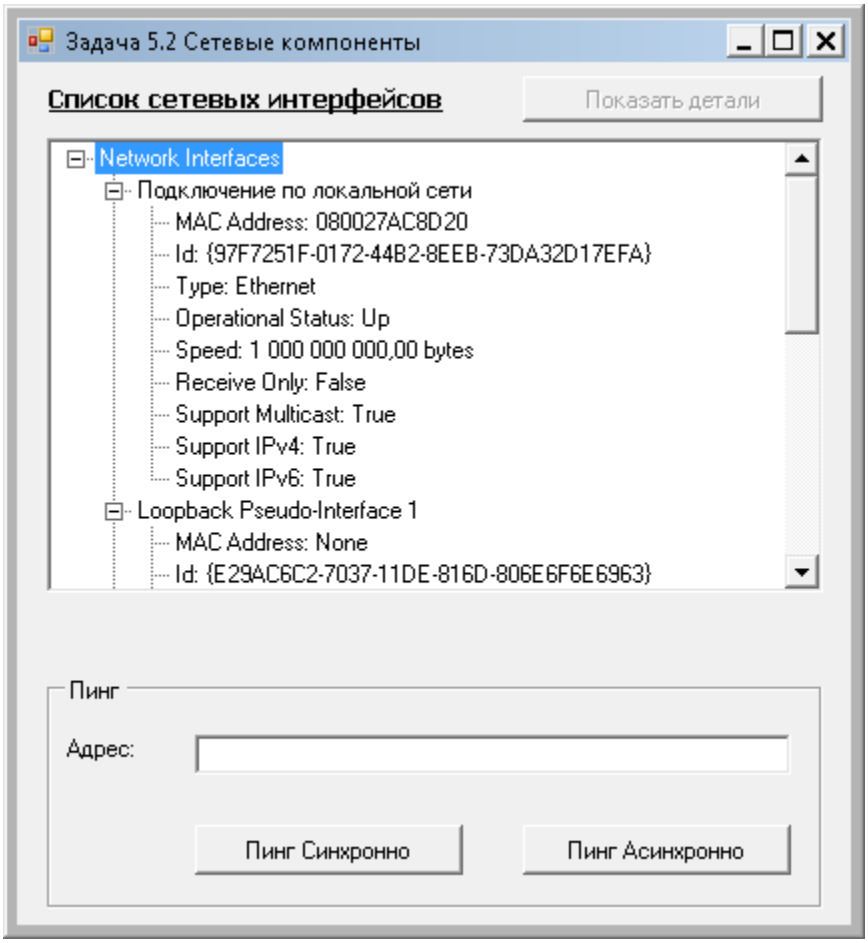

Рис.17. Внешний вид приложения

Решение:

```
    Imports System.Text
    Imports System.Net.NetworkInformation
    Imports Microsoft.VisualBasic.ApplicationServices
    Imports Microsoft.VisualBasic.Devices

    Public Class Form1
        Public Sub New()
            ' This call is required by the Windows Form
Designer.
    InitializeComponent()
            ' get a list of all network interfaces on this
computer
            If NetworkInterface.GetIsNetworkAvailable Then
                interfaces                                  =
NetworkInterface.GetAllNetworkInterfaces
            End If
            Me.toolTip1.IsBalloon = True
```

```
            Me.ToolTip1.ToolTipIcon = ToolTipIcon.Info
    LoadTreeView(interfaces)
            ping = New Ping
            ' register with the NetworkChange events
    AddHandlerNetworkChange.NetworkAddressChanged,
AddressOfNetworkChange_NetworkAddressChanged
    AddHandlerNetworkChange.NetworkAvailabilityChanged,
AddressOfNetworkChange_NetworkAvailabilityChanged
        End Sub

        Private Sub LoadTreeView(ByVal interfaces As
NetworkInterface())
            ' Load the details about each network interface
into the treeview
    Me.infoTree.BeginUpdate()
            Dim rootNode As TreeNode = New TreeNode
    rootNode.Text = "Network Interfaces"
            If interfaces.GetLength(0) > 0 Then
                For Each networkInterface As NetworkInterface
In interfaces
                    Dim address As PhysicalAddress =
networkInterface.GetPhysicalAddress
                    Dim bytes As Byte() =
address.GetAddressBytes
                    Dim networkInterfaceNode As TreeNode = New
TreeNode
    networkInterfaceNode.Text = networkInterface.Name
    networkInterfaceNode.Tag = networkInterface
                    Dim networkInterfaceDetailNode As TreeNode
= New TreeNode
    networkInterfaceDetailNode.Text = "MAC Address: " +
(Microsoft.VisualBasic.IIf(address.ToString = String.Empty,
"None", address.ToString))
    networkInterfaceNode.Nodes.Add(networkInterfaceDetailNode)
    networkInterfaceDetailNode = New TreeNode
    networkInterfaceDetailNode.Text = "Id: " +
networkInterface.Id.ToString
    networkInterfaceNode.Nodes.Add(networkInterfaceDetailNode)
    networkInterfaceDetailNode = New TreeNode
    networkInterfaceDetailNode.Text = "Type: " +
networkInterface.NetworkInterfaceType.ToString
    networkInterfaceNode.Nodes.Add(networkInterfaceDetailNode)
    networkInterfaceDetailNode = New TreeNode
    networkInterfaceDetailNode.Text = "Operational Status: " +
networkInterface.OperationalStatus.ToString
    networkInterfaceNode.Nodes.Add(networkInterfaceDetailNode)
```

```
    networkInterfaceDetailNode = New TreeNode
    networkInterfaceDetailNode.Text    =    "Speed:    "    +
networkInterface.Speed.ToString("N") + " bytes"
    networkInterfaceNode.Nodes.Add(networkInterfaceDetailNode)
    networkInterfaceDetailNode = New TreeNode
    networkInterfaceDetailNode.Text  =  "Receive  Only:  "  +
networkInterface.IsReceiveOnly.ToString
    networkInterfaceNode.Nodes.Add(networkInterfaceDetailNode)
    networkInterfaceDetailNode = New TreeNode
    networkInterfaceDetailNode.Text = "Support Multicast: " +
networkInterface.SupportsMulticast.ToString
    networkInterfaceNode.Nodes.Add(networkInterfaceDetailNode)
    networkInterfaceDetailNode = New TreeNode
    networkInterfaceDetailNode.Text  =  "Support  IPv4:  "  +
networkInterface.Supports(NetworkInterfaceComponent.IPv4).ToSt
ring
    networkInterfaceNode.Nodes.Add(networkInterfaceDetailNode)
    networkInterfaceDetailNode = New TreeNode
    networkInterfaceDetailNode.Text  =  "Support  IPv6:  "  +
networkInterface.Supports(NetworkInterfaceComponent.IPv6).ToSt
ring
    networkInterfaceNode.Nodes.Add(networkInterfaceDetailNode)
    rootNode.Nodes.Add(networkInterfaceNode)
                Next
            End If
    Me.infoTree.Nodes.Add(rootNode)
    rootNode.Expand()
    Me.infoTree.EndUpdate()
        End Sub

        Private  Sub  infoTree_NodeMouseHover(ByVal  sender  As
System.Object,              ByVal              e              As
System.Windows.Forms.TreeNodeMouseHoverEventArgs)       Handles
infoTree.NodeMouseHover
          Dim    networkInterface    As    NetworkInterface    =
CType(e.Node.Tag, NetworkInterface)
            If Not (networkInterface Is Nothing) Then
    e.Node.ToolTipText = networkInterface.Description
            End If
        End Sub

        Private   Sub   infoTree_AfterSelect(ByVal   sender   As
System.Object,              ByVal              e              As
System.Windows.Forms.TreeViewEventArgs)                  Handles
infoTree.AfterSelect
```

```
            'if the selected node is a network interface enable
the Show Interface Details button
            Dim   networkInterface   As   NetworkInterface   =
CType(e.Node.Tag, NetworkInterface)
            If Not (networkInterface Is Nothing) Then
    ShowInterfaceDetails.Enabled = True
                _currentInterface = networkInterface
            Else
    ShowInterfaceDetails.Enabled = False
                _currentInterface = Nothing
            End If
        End Sub

        Private Sub ShowInterfaceDetails_Click(ByVal sender As
System.Object,   ByVal   e   As   System.EventArgs)   Handles
ShowInterfaceDetails.Click
            Dim detailTree As DetailTree = New DetailTree

detailTree.LoadIPv4InterfaceStats(_currentInterface.GetIPv4Sta
tistics)
    detailTree.LoadIPProperties(_currentInterface.GetIPPropertie
s)
    detailTree.ShowDialog()
        End Sub

        Private Sub DoPing_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles DoPing.Click
            If Me.pingIPAddress.Text.Length> 0 Then
    Me.DoPing.Enabled = Me.doPingAsynch.Enabled = False
                Try
                    Dim     reply     As     PingReply     =
ping.Send(Me.pingIPAddress.Text)
                    'Show the ping results
    ShowPingResultInfo(reply)
                Catch ex As PingException
    MessageBox.Show("Произошласледующаяошибка:"
&Microsoft.VisualBasic.Chr(10)               &               ""
&Microsoft.VisualBasic.Chr(13)    &    ""    +    ex.Message,
"Примерсети", MessageBoxButtons.OK, MessageBoxIcon.Error)
                End Try
    Me.DoPing.Enabled = Me.doPingAsynch.Enabled = True
            Else
    MessageBox.Show("You must enter an address to ping first",
"Network Sample", MessageBoxButtons.OK, MessageBoxIcon.Error)
            End If
        End Sub
```

```
        Private Sub doPingAsynch_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
doPingAsynch.Click
            If Me.pingIPAddress.Text.Length> 0 Then
    Me.DoPing.Enabled = Me.doPingAsynch.Enabled = False
                Try
    ping.SendAsync(Me.pingIPAddress.Text, Me)
                Catch ex As PingException
    MessageBox.Show("Произошласледующаяошибка:"
&Microsoft.VisualBasic.Chr(10)                &              ""
&Microsoft.VisualBasic.Chr(13)    &    ""    +    ex.Message,
"Примерсети", MessageBoxButtons.OK, MessageBoxIcon.Error)
                End Try
            Else
    MessageBox.Show("Введитеip-адресилиdns-имяхоста",
"Примерсети", MessageBoxButtons.OK, MessageBoxIcon.Error)
            End If
    System.Threading.Thread.Sleep(1000)
        End Sub

        Sub ping_PingCompleted(ByVal sender As Object, ByVal e
As PingCompletedEventArgs) Handles ping.PingCompleted
    Me.DoPing.Enabled = Me.doPingAsynch.Enabled = True
            If e.Cancelled = True Then
    MessageBox.Show("Асинхронныйпингбылотменен ", "Примерсети",
MessageBoxButtons.OK, MessageBoxIcon.Error)
                Exit Sub
            End If
            If Not (e.ErrorIs Nothing) Then
    MessageBox.Show("Произошласледующаяошибка:"
&Microsoft.VisualBasic.Chr(10)                &              ""
&Microsoft.VisualBasic.Chr(13)   &   ""   +   e.Error.Message,
"Примерсети", MessageBoxButtons.OK, MessageBoxIcon.Error)
            End If
    ShowPingResultInfo(e.Reply)
        End Sub

        Private   Sub   ShowPingResultInfo(ByVal   reply   As
PingReply)
                  If reply.Status = IPStatus.Success Then
                Dim   builder   As   StringBuilder   =   New
StringBuilder
    builder.Append("You pinged: " &Microsoft.VisualBasic.Chr(9)
& "" &Microsoft.VisualBasic.Chr(9) & "")
    builder.AppendLine(reply.Address.ToString)
```

```
    builder.AppendLine()
    builder.Append("The     status     returned     was:     "
&Microsoft.VisualBasic.Chr(9) & "")
    builder.AppendLine(reply.Status.ToString)
    builder.AppendLine()
    builder.Append("The     roundtrip     time     was:     "
&Microsoft.VisualBasic.Chr(9) & "")
    builder.AppendLine(reply.RoundtripTime.ToString("N"))
    builder.AppendLine()
                If Not (reply.BufferIs Nothing) Then
    builder.Append("The     reply     message     was:     "
&Microsoft.VisualBasic.Chr(9) & "")
    builder.AppendLine(ConvertByteBufferToString(reply.Buffer))
    builder.AppendLine()
                End If
                Dim options As PingOptions = reply.Options
    builder.Append("Ttl:  "  &Microsoft.VisualBasic.Chr(9)  &  ""
&Microsoft.VisualBasic.Chr(9)               &               ""
&Microsoft.VisualBasic.Chr(9) & "")
    builder.AppendLine(options.Ttl.ToString)
    builder.AppendLine()
    builder.Append("Dont               Fragment:               "
&Microsoft.VisualBasic.Chr(9)               &               ""
&Microsoft.VisualBasic.Chr(9) & "")
    builder.AppendLine(options.DontFragment.ToString)
    builder.AppendLine()
    MessageBox.Show(builder.ToString,    "Результатыпинга    ",
MessageBoxButtons.OK, MessageBoxIcon.Information)
                Return
            End If
    MessageBox.Show("Ping      was      not      Successfull."
&Microsoft.VisualBasic.Chr(10)              &               ""
&Microsoft.VisualBasic.Chr(13)    &    "Status    Code:    "    +
reply.Status.ToString,  "Ping  Results",  MessageBoxButtons.OK,
MessageBoxIcon.Information)
        End Sub

        Private                                          Function
ConvertByteBufferToString(ByValinputBuffer   As   Byte())   As
String
            Return
System.Text.Encoding.ASCII.GetString(inputBuffer,           0,
inputBuffer.Length)
        End Function
        Sub      NetworkChange_NetworkAvailabilityChanged(ByVal
sender As Object, ByVal e As NetworkAvailabilityEventArgs)
```

```
            'this method is called by the
NetworkAvailabilityChangedEventHandler delegate
    MessageBox.Show("The network avialability has changed."
&Microsoft.VisualBasic.Chr(13)              &              ""
&Microsoft.VisualBasic.Chr(10) & "The network is now " +
(Microsoft.VisualBasic.IIf(e.IsAvailable = True, "on line",
"off line")), "Network Change Event", MessageBoxButtons.OK,
MessageBoxIcon.Exclamation)
        End Sub

        Sub NetworkChange_NetworkAddressChanged(ByVal sender As
Object, ByVal e As EventArgs)
    MessageBox.Show("The network address has changed.",
"Network      Change      Event",      MessageBoxButtons.OK,
MessageBoxIcon.Exclamation)
        End Sub
        Private interfaces As NetworkInterface()
        Private _currentInterface As NetworkInterface
        Private WithEvents ping As Ping
    End Class
```

2.9. Мультимедиа

Иногда возникает необходимость разнообразить программы, сделать более привлекательными. Один из способов – это мультимедиа, анимационные эффекты, сопровождаемые или не сопровождаемые звуком. В разделе мультимедиа рассматривается два типа анимации: простая анимация, не сопровождаемая звуком, и анимационные эффекты со звуком (AVI клипы). Соответственно, рассмотрены такие компоненты.

Задача 1. Создать приложение, воспроизводящее стандартные системные звуки.

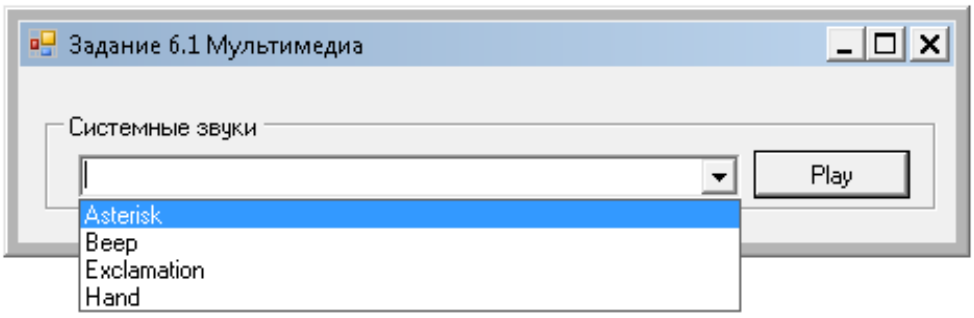

Рис.18. Внешний вид приложения

Решение:

```
    Imports System.Media
    Imports System.ComponentModel
    Public Class Form1
        Private fileName As String
        ' A SoundPlayer control is used for playing system
sounds
        ' and .wav files.
        Public Sub New()
            ' This call is required by the Windows Form
Designer.
    InitializeComponent()
            ' Display available system sounds on the form.
    LoadSystemSounds()
        End Sub

        Public Sub LoadSystemSounds()
```

```
            ' These are the system sounds currently supported.
            ' There is no list avaliable for enumerating,
    ' so load them manually
    systemSoundsComboBox.Items.AddRange( _
                New      String()      {"Asterisk",      "Beep",
"Exclamation", "Hand"})
        End Sub

        Private Sub playSystemSoundButton_Click(ByVal sender As
System.Object,   ByVal   e   As   System.EventArgs)   Handles
playSystemSoundButton.Click
            Select Case systemSoundsComboBox.Text
                Case "Asterisk"
    My.Computer.Audio.PlaySystemSound(System.Media.SystemSounds
.Asterisk)
                    Exit Select
                Case "Beep"
    My.Computer.Audio.PlaySystemSound(System.Media.SystemSounds
.Beep)
                    Exit Select
                Case "Exclamation"
    My.Computer.Audio.PlaySystemSound(System.Media.SystemSounds
.Exclamation)
                    Exit Select
                Case "Hand"
    My.Computer.Audio.PlaySystemSound(System.Media.SystemSounds
.Hand)
                    Exit Select
                Case Else
                    Throw   New   ApplicationException("Invalid
system sound.")
            End Select
        End Sub
    End Class
```

Задача 2. Создать приложение, воспроизводящее WAV-файлы.

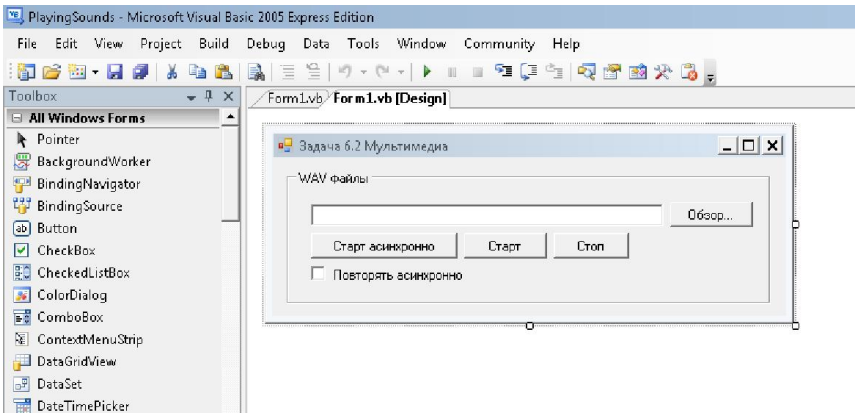

Рис.19. Внешний вид приложения

Решение:

```
Imports System.Media
Imports System.ComponentModel

Public Class Form1
    Private fileName As String
        Public Sub New()
InitializeComponent()
        End Sub

        Private Sub browseButton_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
browseButton.Click
            ' Provide a file dialog for browsing for the sound
file.
            Dim openFileDialog1 As New OpenFileDialog()
            openFileDialog1.InitialDirectory =
"c:\\windows\\media"
            openFileDialog1.Filter = "WAV files (*.wav)|*.wav"
            openFileDialog1.FilterIndex = 1
            openFileDialog1.RestoreDirectory = True
            If (openFileDialog1.ShowDialog() =
Windows.Forms.DialogResult.OK) Then
    fileName = openFileDialog1.FileName
    soundFileNameTextBox.Text = fileName
            End If
        End Sub
        Private Sub playSyncButton_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
playSyncButton.Click
            If fileNameIsNot Nothing Then
    My.Computer.Audio.Play(fileName)
            End If
```

```
        End Sub

        Private Sub playAsyncButton_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
playAsyncButton.Click
            If loopCheckBox.Checked Then
                'You can also play the sound asynchronously and
loop the sound using My.
    My.Computer.Audio.Play(fileName,
AudioPlayMode.BackgroundLoop)
            Else
                'You can play the sound synchronously but using
the WaitToComplete argument
    My.Computer.Audio.Play(fileName, AudioPlayMode.Background)
            End If
        End Sub

        Private Sub stopAsyncPlayButton_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
stopAsyncPlayButton.Click
            'You can stop the audio from playing by simply
using the Stop Method.
    My.Computer.Audio.Stop()
        End Sub
    EndClass
```

Подводя итоги, можно сказать, что изучение следует начать с теоретической части, где проводится первоначальное знакомство со средой Visual Basic. В главе 2 приведен внешний вид среды VB, рассмотрен простейший пример расположения компонента на форме, что обеспечивает наглядность изучения строения среды Visual Basic, далее целесообразно рассмотреть несколько простых свойств этого компонента (для простоты лучше брать компонент Button, и рассмотреть его свойства как Caption). После определения свойств, необходимо перейти к рассмотрению методов и обработки событий. Это обеспечит наглядное демонстрирование основных свойств ООП, таких как инкапсуляция (свойства и методы компонента объединены), полиморфизм (свойства, и методы имеют общее имя, но выполняются по–разному), наследственность (обнаружение общих свойств и методов компонента). После первоначального ознакомления можно переходить к изучению параграфа «основные компоненты», в котором пользователь познакомиться с назначением некоторых основных

компонентов. Изучение следует параллельно проводить на конкретных задачах.

После полного освоения теоретической части можно приступать к практическим заданиям, под четким руководством преподавателя, так как у студентов могут возникать много ошибок, это связанно с тем, что большую часть кода они должны прописывать руками. Когда студенты усвоят простые задачи, то можно переходить к более сложным задачам.

Данная последовательность изложения материала будет способствовать закреплению и контролю по строению среды Visual Basic. Практическое освоение применения основных компонентов следует начать со знакомством разобранных задач, которые приведены с кодом решения.

ЗАКЛЮЧЕНИЕ

В связи с быстрым развитием вычислительной техники, обществу требуется большое количество специалистов умеющих разрабатывать программные приложения любой сложности. Одним из наиболее популярных способов разработки программных приложений является визуальное программирование. Примером такой системы может являться среда Visual Basic. Для обучения визуальному программированию нами был разработан дидактический комплекс, предназначенный для обучения студентов ВУЗов обучающихся по профилю физико-математический, технологический. Основной акцент делается на практическую составляющую обучения, т.к. в большинстве литературы по программированию в Visual Basic описана только теоретическая часть, а практических заданий мало или они не имеют определенной тематики.

Таким образом, частично решается проблема дидактического наполнения дисциплин, связанных с визуальным программированием.

В данной работе рассматривается изучение объектно-ориентированного подхода к программированию в среде VisualBasicна практических примерах ориентированных на использование основных свойств ООП. Разработаны задания (текстовые задачи), позволяющие получить начальные навыки программирования в VisualBasic. Приведена краткая теория по каждому разделу среды.

В процессе разработки дидактического комплекса были поставлены и решены следующие задачи:

- анализ государственных стандартов высшего профессионального образования;
- анализ учебной и научной литературы по обучению визуальному программированию;

- разработан комплект дидактических материалов в поддержку курса ООП на основе среды VisualBasic;
- разработана методика применения разработанного комплекта дидактических материалов в поддержку визуального программирования;

Таким образом, дидактический комплект в поддержку изучения VisualBasic насчитывает 20 текстовых задач, что позволяет в полной мере изучить визуальное программирование на основе среды VisualBasic.

Данную работу можно переориентировать на другие визуальные среды, например Delphi 7, Visual C++. Все задачи придерживаются объектно-ориентированной методологии, и могут быть перенесены на различные платформы. Необходимо пересмотреть условия задач, решения задач с кодом.

СПИСОК ЛИТЕРАТУРЫ

1. *NET* Сетевое программирование; ЛОРИ – М.: , 2013. – 99 с.
2. *Eric A. Smith* Active Server® Pages Bible / Smith Eric – Высшая школа – М.: Высшая школа – 2016. – 65 с.
3. *Браун С.* Visual Basic 6; / С. Браун – СПб.: 2001. – 576 с.
4. *Гарнаев А.Д.* Visual Basic .NET. Разработка приложений / А.Д. Гарнаев – М.: БХВ-Петербург, 2017. - 88 с.
5. *Геворкян Г.Х.;* Семенов, В.Н.Бейсик – это просто / Г.Х. Геворкян, В.Н. Семенов – М.: Радио и связь, 1989. – 144 с.
6. *Глушаков С.В* Microsoft 2007. Краткий курс /С.В. Глушаков, А.С, Сурядный – М.: Харвест, 2008. – 352 с.
7. *Дукин А.А*. Самоучитель Visual Basic 2010 / А.А. Дукин, А.В. Пожидаев – М.: БХВ-Петербург, 2013. – 146 с.
8. *Жаров Дмитрий* Моделирование в Visual Basic / Дмитрий Жаров – М.: СИНТЕГ, 2008. – 176 с.
9. *Кашаев* С. Офисные решения с использованием Visual Basic 2007 и VBA / С. Кашаев –СПб.: 2009. – 352 с.
10. *Кетков Ю.Л.* Диалог на языке бейсик для мини– и микро–ЭВМ / Ю.Л. Кетков – М.: Наука, 1988. – 368 с.
11. *Коцюбинский, А.О.* VB в примерах / А.О. Коцюбинский, С.В. Грошев – М.: ГроссМедиа, 2004. - 304 с.
12. *Культин Н.Б.* Visual Basic для студентов и школьников / Н.Б. Культин, Л.Б. Цой – СПб.: БХВ-Петербург, 2010. – 416 с.
13. *Мак-Федрис Пол* Формулы и функции в MicrosoftExcel 2003 / Пол Мак-Федрис – М.: Высшая школа, 2006. – 576 с.
14. *Мур Дж.* Экономическое моделирование в MicrosoftExcel (+ CD-ROM) / Джон *Мур, Г.*Эппен, Л. Уэдерфорд и др. – М.: Огни, 2004. – 852 с.

15. *Могилев А.В. и др.* Информатика: учеб. Пособие для студ. пед. вузов / А.В. Могилев, Н.И. Пак, Е.К. Хеннер // Под ред. Е.К. Хеннера. – М.:Академия», 2000. – 816 с.

16. *Патрик Т.* Visual Basic 2005. Рецепты программирования / Т.Патрик – М.: БХВ-Петербург, 2013. – 57 с.

17. *Политика в области образования и новые информационные технологии:* Нац. доклад РФ на II Международном конгр. ЮНЕСКО «Образование и информатика» № 6. Москва, 1–5 июля 1996г. // ИНФО. – 1996. – 12 с.

18. *Половина И.П.,* УМК Языки и методы программирования / И.П. Половина – Пермь: 2008. – 85 с.

19. *Пикуза Владимир* Экономические и финансовые расчеты в Excel. Самоучитель / Владимир *Пикуза, Александр* Гаращенко – СПб.: Питер, 2003. – 400 с.

20. *Справка по SQL*(БД) [Электронный ресурс] – Режим доступа: http://www.sql (дата обращения: 1 июня 2018 г.).

21. *Трусов М. А.* Visual Basic .NET. Практическое руководство для начинающего программиста / М.А. Трусов – М,: НТ Пресс, 2015. – 32 с.

22. *Харвей Г.* Visual Basic 5.0 для "чайников"; Диалектика; Издание 2-е / Г. Харвей – М.: 2014. – 288 с.

23. *Юдин М.В.;* Куприянова, А.В. MicrosoftExcel 2007. Компьютерная шпаргалка / М.В. Юдин, А.В. Куприянова – СПб.: Наука и Техника, 2009. – 265 с.

24. *ФГОС ВО* Направление 090301«Информатика и вычислительная техника». – Введ. 09.02.2016г. [Электронный ресурс], – Режим доступа:http://fgosvo.ru/fgosvo/92/91/4/9 (дата обращения: 15 мая 2018 г.).

25. *ФГОС ВО* Направление 090302«Информационные системы и технологии» – Введ. 30.03.2015г. [Электронный ресурс], –

Режимдоступа:http://fgosvo.ru/fgosvo/92/91/4/9 (дата обращения: 15 мая 2018 г.).

26. *ФГОС ВО* Направление 090303«Прикладная информатика» – Введ. 27.03.2015г. [Электронный ресурс], – Режим доступа:http://fgosvo.ru/fgosvo/92/91/4/9 (дата обращения: 15 мая 2018 г.).

27. *ФГОС ВО* Специальность 090304 «Программная инженерия» – Введ. 01.04.2015г. [Электронный ресурс], – Режим доступа: http://fgosvo.ru/fgosvo/92/91/4/9 (дата обращения: 20 мая 2018 г.).

28. *ФГОС ВО* Специальность 030100 «Информатика». – Введ. 31.01.2005г. [Электронный ресурс], – Режим доступа: http://www.edu.ru/abitur/act.82/index.php. (дата обращения: 20 мая 2018 г.).

29. *ФГОС ВО*"Специальность 075200 «Компьютерная безопасность». – Введ. 5.04.2000г. [Электронный ресурс], – Режим доступа: http://www.edu.ru/abitur/act.82/index.php. (дата обращения: 20 мая 2018 г.).

30. *ФГОС ВО* Специальность 050303 «Картография и геоинформатика». – Введ. 14.03.2015г. [Электронный ресурс] – Режим доступа: http://fgosvo.ru/fgosvo/92/91/4/5 (дата обращения: 20 мая 2018 г.).

31. *Якушева Н.М.* Введение в программирование на языке Visual Basic. Net / Н.М. Якушева ; Финансы и статистика – М.: 2013. – 40 с.