ФГОС

9

Н. Д. Угринович

ИНФОРМАТИКА

ИЗДАТЕЛЬСТВО БИНОМ

ФГОС

Н. Д. Угринович

ИНФОРМАТИКА

Учебник для 9 класса

4-е издание

Рекомендовано
Министерством образования и науки Российской Федерации
к использованию при реализации имеющих государственную
аккредитацию образовательных программ начального общего,
основного общего, среднего общего образования

Москва
БИНОМ. Лаборатория знаний
2016

УДК 004.9
ББК 32.97
У27

Угринович Н. Д.

У27 Информатика : учебник для 9 класса / Н. Д. Угринович. — 4-е изд. — М. : БИНОМ. Лаборатория знаний, 2016. — 152 с. : ил.

ISBN 978-5-906812-88-9

Учебник предназначен для продолжения изучения курса «Информатика» в общеобразовательных школах, а также в классах предпрофильной подготовки по физико-математическому и информационно-технологическому профилям. Большое внимание в учебнике уделяется формированию у учащихся алгоритмического и системного мышления, а также практических умений и навыков в области информационных технологий. Учебник мультисистемный, так как практические работы компьютерного практикума могут выполняться в различных операционных системах: Windows или Linux. Соответствует Федеральному государственному образовательному стандарту основного общего образования (2010 г.).

УДК 004.9
ББК 32.97

Учебное издание

Угринович Николай Дмитриевич

ИНФОРМАТИКА

Учебник для 9 класса

Научный редактор *М. Цветкова*
Ведущий редактор *О. Полежаева*
Ведущий методист *И. Сретенская*
Художник *Н. Новак*
Технический редактор *Е. Денюкова*
Корректор *Е. Клитина*
Компьютерная верстка: *В. Носенко*

Подписано в печать 14.12.15. Формат 70x100/16. Усл. печ. л. 12,35.
Тираж 20 000 экз. Заказ №м2558 (K-Sm).

Издательство «БИНОМ. Лаборатория знаний»
127473, Москва, ул. Краснопролетарская, д. 16, стр. 1,
тел. (495) 181-5344, e-mail: binom@Lbz.ru
http://www.Lbz.ru, http://metodist.Lbz.ru

Отпечатано в филиале «Смоленский полиграфический комбинат»
ОАО «Издательство «Высшая школа».
214020, г. Смоленск, ул. Смольянинова, 1
Тел.: +7 (4812) 31-11-96. Факс: +7 (4812) 31-31-70
E-mail:spk@smolpk.ru, http://www.smolpk.ru

ISBN 978-5-906812-88-9

Оглавление

Рекомендации по использованию учебника 6

Глава 1. Основы алгоритмизации и объектно-ориентированного программирования 9

1.1. Алгоритм и его формальное исполнение 9
1.2. Кодирование основных типов алгоритмических структур на языках объектно-ориентированного и процедурного программирования 19
1.3. Переменные: тип, имя, значение 25
1.4. Арифметические, строковые и логические выражения 28
1.5. Функции в языках объектно-ориентированного и процедурного программирования 29
1.6. *Графические возможности объектно-ориентированного языка программирования Visual Basic 33

Практические работы компьютерного практикума к главе 1 «Основы алгоритмизации и объектно-ориентированного программирования» 37

Практическая работа 1.1
Знакомство с системами объектно-ориентированного и процедурного программирования 37
Практическая работа 1.2
Разработка проекта «Переменные» 43
Практическая работа 1.3
Разработка проекта «Калькулятор» 46
Практическая работа 1.4
Разработка проекта «Строковый калькулятор» 50
Практическая работа 1.5
Разработка проекта «Даты и время» 52
Практическая работа 1.6
Разработка проекта «Сравнение кодов символов» 55

Практическая работа 1.7
Разработка проекта «Отметка» 57
Практическая работа 1.8
Разработка проекта «Коды символов» 60
Практическая работа 1.9
Разработка проекта «Слово-перевертыш» 63
*Практическая работа 1.10
Разработка проекта «Графический редактор» 65
*Практическая работа 1.11
Разработка проекта «Системы координат» 69
*Практическая работа 1.12
Разработка проекта «Анимация» 71

Глава 2. Моделирование и формализация 74
2.1. Окружающий мир как иерархическая система 74
2.2. Моделирование, формализация, визуализация 78
2.3. Основные этапы разработки и исследования моделей на компьютере 87
2.4. Построение и исследование физических моделей 89
2.5. Приближенное решение уравнений 91
2.6. Компьютерное конструирование с использованием системы компьютерного черчения 92
2.7. Экспертные системы распознавания химических веществ 93
2.8. Информационные модели управления объектами 96
Практические работы компьютерного практикума к главе 2 «Моделирование и формализация» 99
*Практическая работа 2.1
Разработка проекта «Бросание мячика в площадку» . . 99
Практическая работа 2.2
Разработка проекта «Графическое решение уравнения» 105
Практическая работа 2.3
Выполнение геометрических построений в системе компьютерного черчения КОМПАС 108
Практическая работа 2.4
Разработка проекта «Распознавание удобрений» . . . 117
Практическая работа 2.5
Разработка проекта «Модели систем управления» . . . 120

Глава 3. Логика и логические основы компьютера 125
3.1. Алгебра логики 125
3.2. Логические основы устройства компьютера 129

Практические работы компьютерного практикума к главе 3 «Логика и логические основы компьютера» 135
Практическая работа 3.1
Таблицы истинности логических функций 135
Практическая работа 3.2
Модели электрических схем логических элементов «И», «ИЛИ» и «НЕ» 138

Глава 4. Информационное общество и информационная безопасность 140
4.1. Информационное общество 140
4.2. Информационная культура 144
4.3. Правовая охрана программ и данных. Защита информации 146

Рекомендации по использованию учебника

1. Учебник «Информатика-9» входит в состав законченной линейки учебников для основной школы: «Информатика-7», «Информатика-8» и «**Информатика-9**». Эти учебники обеспечивают изучение курса «Информатика» в соответствии с новым Федеральным государственным образовательным стандартом основного общего образования (2010).

   Линейка учебников для 7–9 классов является основой учебно-методического комплекта, в который также входят практикум и методическое пособие.

2. Компьютерный практикум может проводиться в операционных системах Windows  и Linux  . К каждой главе указано необходимое для выполнения работ компьютерного практикума программное обеспечение и его источники.

   Дистрибутивы программ, необходимых для выполнения практических работ компьютерного практикума, можно скачать из Интернета по указанным ссылкам в учебниках.

   Начало каждой работы компьютерного практикума обозначается значками операционной системы и приложений, для которых приведена подробная пошаговая инструкция выполнения работы.

3. В тексте учебника приняты следующие шрифтовые выделения:
   - шрифтом Arial выделены имена программ, файлов, папок и дисков;
   - шрифтом `Courier New` выделены программы на языках программирования;
   - **полужирным шрифтом** выделены важные термины и понятия;
   - *курсивом* выделены названия диалоговых окон, вкладок и управляющих элементов графического интерфейса операционных систем и приложений.

4. В работе с книгой вам помогут навигационные значки:

 — важное утверждение или определение;

 — ссылка на Интернет-ресурс;

 — задания для использования в подготовке к итоговой аттестации;

 — вопросы и задания к параграфу;

 — выполнение практической работы на компьютере;

 — дополнительная интересная информация;

 — задания для самостоятельного выполнения;

 — практические задания для самостоятельного выполнения на компьютере;

 — домашний эксперимент или проект;

 — указание на наличие примеров и файлов для выполнения работ компьютерного практикума на сайте www.metodist.lbz.ru в авторской мастерской Угриновича Н. Д.

Глава 1

ОСНОВЫ АЛГОРИТМИЗАЦИИ И ОБЪЕКТНО-ОРИЕНТИРОВАННОГО ПРОГРАММИРОВАНИЯ

1.1. Алгоритм и его формальное исполнение

1.1.1. Свойства алгоритма и его исполнители

Дискретность. Во многих отраслях человеческой деятельности для достижения требуемого результата используются **алгоритмы**, содержащие четкие описания последовательности действий. Примерами алгоритмов являются кулинарные рецепты, в которых подробно описана последовательность действий по приготовлению пищи.

Алгоритмы кулинарных рецептов состоят из отдельных действий, которые обычно нумеруются. Разделение алгоритма на последовательность шагов является важным свойством алгоритма и называется **дискретностью**.

Алгоритм приготовления блюда быстрого питания:

1. Высыпать в емкость содержимое пакетика.
2. Налить в емкость 200 мл горячей воды.
3. Тщательно перемешать.

Результативность. Алгоритмами являются известные из начальной школы правила сложения, вычитания, умножения и деления столбиком. Применение этих алгоритмов независимо от количества разрядов в числах и, соответственно, количества вычислительных шагов алгоритма всегда приводит к результату. Получение из исходных данных результата за конечное число шагов называется **результативностью** алгоритма.

Алгоритм сложения целых чисел в десятичной системе счисления:

1. Записать числа в столбик так, чтобы цифры самого младшего разряда чисел (единицы) расположились одна под другой (на одной вертикали).

2. Сложить цифры младшего разряда.
3. Записать результат под горизонтальной чертой на вертикали единиц, если при этом полученная сумма больше или равна величине основания системы счисления (в данном случае 10), перенести десятки в старший разряд десятков.
4. Повторить пункты 2 и 3 для всех разрядов с учетом переносов из младших разрядов.

	1	
2	5	6
1	2	8
3	8	4

Массовость. Алгоритмы сложения, вычитания, умножения и деления могут быть применены для любых чисел, причем не только в десятичной, но и в других позиционных системах счисления (двоичной, восьмеричной, шестнадцатеричной и др.). Возможность применения алгоритма к большому количеству различных исходных данных называется **массовостью**.

Само слово «алгоритм» происходит от «algorithmi» — латинской формы написания имени выдающегося математика IX века аль-Хорезми, который сформулировал правила выполнения арифметических операций.

Исполнители алгоритмов. Алгоритмы широко используются в технике в системах управления объектами. В любой системе управления существует управляющий объект, который является **исполнителем** алгоритма управления. Так, в системах терморегуляции для поддержания определенной температуры в помещении исполнителем алгоритма может являться как человек, так и микропроцессор.

Алгоритм терморегуляции:
1. Измерить температуру в помещении.
2. Если измеренная температура ниже заданной, включить обогреватель.

Детерминированность (определенность). При управлении самолетом используются сложные алгоритмы, исполнителями которых являются пилот или бортовой компьютер. При этом важно, чтобы каждая команда определяла однозначное действие исполнителя. Кроме того, последовательность выполнения действий, например, при взлете должна быть строго определенной (например, нельзя отрываться от взлетной полосы, пока самолет не набрал необходимую взлетную скорость). Исполнитель алгоритма, выполнив очередную команду, должен точно знать, какую команду необходимо исполнять следующей.

Понятность. После включения компьютера начинают выполняться алгоритмы тестирования компьютера и загрузки операцион-

ной системы. Исполнителем этих алгоритмов является компьютер, поэтому они должны быть записаны на понятном компьютеру машинном языке.

Каждый исполнитель обладает определенным набором, **системой команд**, которые он может выполнить. Алгоритм должен быть понятен исполнителю, т. е. должен содержать только те команды, которые входят в систему команд исполнителя.

Свойства алгоритма. Выше были приведены примеры алгоритмов из различных областей человеческой деятельности и знаний. В этих алгоритмах различные исполнители выполняли операции над объектами различной природы (материальными объектами и числами). При этом во всех примерах можно выделить следующие основные свойства алгоритма:

Результативность и дискретность. Алгоритм должен обеспечивать получение из исходных данных результата за конечное число дискретных шагов.

Массовость. Один и тот же алгоритм может применяться к большому количеству однотипных объектов.

Детерминированность (*определенность*). Исполнитель должен выполнять команды алгоритма в строго определенной последовательности.

Понятность. Алгоритм должен содержать команды, входящие в систему команд исполнителя и записанные на понятном исполнителю языке.

Алгоритм — это описание детерминированной последовательности действий, направленных на получение из исходных данных результата за конечное число дискретных шагов с помощью понятных исполнителю команд.

Формальное исполнение алгоритма. Из приведенных выше свойств алгоритма вытекает возможность его формального выполнения. Это означает, что алгоритм можно выполнять, не вникая в содержание поставленной задачи, а только строго выполняя последовательность действий, описанных в алгоритме.

Контрольные вопросы

1. Приведите примеры известных вам алгоритмов.
2. Перечислите основные свойства алгоритмов и проиллюстрируйте их примерами.
3. Как вы понимаете формальное исполнение алгоритма?

Задания для самостоятельного выполнения

1.1. *Задание с развернутым ответом.* Запишите алгоритм вычитания столбиком целых чисел в десятичной системе счисления.

Контрольные вопросы

1. Перечислите основные элементы блок-схем и их назначение.

1.1.2. Выполнение алгоритмов компьютером

Выше мы говорили о возможности формального исполнения алгоритма. Это означает, что выполнение алгоритма может быть автоматически реализовано техническими устройствами, среди которых особое место занимает компьютер. При этом говорят, что компьютер исполняет программу (последовательность команд), реализующую алгоритм.

Алгоритм, записанный на «понятном» компьютеру языке программирования, называется **программой**.

Машинный язык. На заре компьютерной эры, в 40–50-е годы XX века, программы писались на машинном языке и представляли собой очень длинные последовательности нулей и единиц. Составление и отладка таких программ являлись чрезвычайно трудоемким делом. Программы на машинных языках были машинно-зависимыми, т. е. для каждой ЭВМ необходимо было создавать свою собственную программу, так как в ней в явной форме учитывались аппаратные ресурсы ЭВМ.

Ассемблер. В начале 50-х годов XX века были созданы языки программирования, которые называются ассемблерами. Вместо одних только нулей и единиц программисты теперь могли пользоваться операторами (`MOV`, `ADD`, `SUB` и т. д.), которые были похожи на слова английского языка. Для преобразования текста программы на ассемблере в понятный компьютеру машинный код использовался компилятор. Программы на ассемблере были также машинно-зависимыми, т. е. ассемблеры для различных процессоров существенно различались между собой.

Языки программирования высокого уровня. С середины 50-х годов XX века начали создаваться первые языки программирования высокого уровня. Эти языки были машинно-независимыми, так как использовали универсальную компьютерную логику и не

были привязаны к типу ЭВМ. Однако для каждого языка и каждого типа ЭВМ должны были быть разработаны собственные компиляторы. Одним из первых языков программирования высокого уровня был созданный в 1964 году Бейсик (Basic).

С конца 50-х годов XX века начали создаваться языки программирования, которые позволили программистам перейти к структурному программированию. Отличительной чертой этих языков было использование операторов ветвления, выбора и цикла и отказ от хаотического использования оператора `goto`. Такие языки позволяют легко кодировать основные алгоритмические структуры. Наибольшее влияние на переход к структурному программированию оказал язык ALGOL (АЛГОЛ), а затем Pascal (назван его создателем Никлаусом Виртом в честь великого физика Блеза Паскаля). Компания Microsoft создала язык QBasic, а в настоящее время язык OpenOffice.org Basic встроен в мультиплатформенную (операционные системы Windows, Linux, Mac OS) интегрированную офисную систему OpenOffice.org.

Существуют различные стили программирования. Перечисленные выше языки поддерживают *процедурный стиль*. Программа, составленная в соответствии с этим стилем, представляет собой последовательность операторов (инструкций), задающих те или иные действия.

Объектно-ориентированные языки. С 70-х годов XX века начали создаваться объектно-ориентированные языки программирования, на которых было удобно программировать в *объектно-ориентированном стиле*. В основу этих языков были положены программные объекты, которые объединяли данные и методы их обработки. С течением времени для этих языков были созданы интегрированные среды разработки, позволяющие визуально конструировать графический интерфейс приложений:

- язык Object Pascal был разработан компанией Borland на основе языка Pascal. После создания интегрированной среды разработки система программирования получила название Delphi;
- язык Visual Basic был создан корпорацией Microsoft на основе языка QBasic для разработки приложений с графическим интерфейсом в среде операционной системы Windows;
- язык Gambas был создан по аналогии с языком Visual Basic для разработки приложений с графическим интерфейсом в среде операционной системы Linux.

Java. В 90-е годы XX века в связи с бурным развитием Интернета был создан язык Java, обеспечивающий межплатформенную

совместимость. На подключенных к Интернету компьютерах с различными операционными системами (Windows, Linux, Mac OS и др.) могли выполняться одни и те же программы. Исходная программа на языке Java компилируется в промежуточный код, который исполняется на компьютере встроенной в браузер виртуальной машиной.

Платформа .NET. Выпущена компанией Microsoft в 2002 году. Эта система предоставляет возможность создавать приложения в различных системах объектно-ориентированного программирования, в которых для составления программного кода используются объектно-ориентированные языки программирования (Visual Basic .NET, Delphi .NET и др.).

На рис. 1.1. показана краткая история развития языков программирования.

Рис. 1.1. История развития языков программирования

Программы-трансляторы. Для того чтобы программа, записанная на языке программирования, могла быть выполнена компьютером, она должна быть переведена на машинный язык. Эту функцию выполняют программы-трансляторы.

Программы-трансляторы с языков программирования бывают двух типов: **интерпретаторы** и **компиляторы.** Интерпретатор — это программа, которая обеспечивает последовательный «перевод» команд программы на машинный язык с одновременным их выполнением. Поэтому при каждом запуске программы на выполнение эта

процедура повторяется. Достоинством интерпретаторов является удобство отладки программы (поиска в ней ошибок), так как возможно «пошаговое» ее исполнение, а недостатком — сравнительно малая скорость выполнения.

Компилятор действует иначе, он переводит весь текст программы на машинный язык и сохраняет его в исполняемом файле (обычно с расширением exe). Затем этот уже готовый к исполнению файл, записанный на машинном языке, можно запускать на выполнение. Достоинством компиляторов является большая скорость выполнения программы, а недостатком большинства из них — трудоемкость отладки, так как невозможно пошаговое выполнение программы.

Системы объектно-ориентированного программирования Visual Basic и Gambas позволяют работать как в режиме интерпретатора, так и в режиме компилятора. На этапе разработки и отладки программы используется режим интерпретатора, а для получения готовой исполняемой программы — режим компилятора.

Система процедурного программирования OpenOffice.org Basic позволяет работать только в режиме интерпретатора.

Контрольные вопросы

1. Подготовьте сообщение о преимуществах машинно-независимых языков программирования перед машинно-зависимыми языками.
2. В чем состоят достоинства и недостатки интерпретаторов и компиляторов?

1.1.3. Основы объектно-ориентированного визуального программирования

Проект (Project). С одной стороны, системы объектно-ориентированного визуального программирования являются **системами программирования**, так как позволяют кодировать алгоритмы на этом языке. С другой стороны, системы объектно-ориентированного визуального программирования являются **средами проектирования**, так как позволяют осуществлять визуальное конструирование графического интерфейса.

Результатом процессов программирования и конструирования является **проект**, который объединяет в себе программный код и графический интерфейс. Системы объектно-ориентированного программирования Visual Basic и Gambas содержат и интерпретатор, и компилятор, поэтому проекты могут выполняться в самой системе, а также могут быть преобразованы в **приложения**, которые выполняются непосредственно в операционных системах Windows и Linux.

Графический интерфейс проекта. Графический интерфейс необходим для реализации интерактивного диалога пользователя с запущенным на выполнение готовым проектом. Основой для создания графического интерфейса разрабатываемого проекта является объект **форма**, которая представляет собой окно, на котором размещаются другие объекты — **элементы управления.**

Элементы управления (рис. 1.2) имеют различное назначение в графическом интерфейсе проекта. *Текстовые поля* (`TextBox`) используются для ввода и вывода данных, *метки* (`Label`) — для вывода данных и пояснительных текстов, *графические окна* (`PictureBox`) — для вывода графики, *кнопки* (`Button`) — для запуска обработчиков событий.

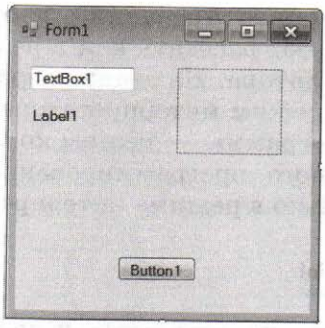


Рис. 1.2. Элементы управления

Графический интерфейс проекта представляет собой **форму,** на которой размещены **элементы управления.**

Визуальное конструирование графического интерфейса проекта состоит в том, что на форме с помощью мыши «рисуются» те или иные элементы управления. Выбрав щелчком мышью на *Панели элементов* нужный элемент, мы можем поместить его на форму разрабатываемого проекта. Процесс размещения на форме элементов управления аналогичен рисованию графических примитивов с использованием графического редактора.

Объекты (Objects). Как конструирование графического интерфейса, так и разработка программного кода базируется на использовании программных **объектов.** Каждый объект обладает определенным набором **свойств** и может использовать определенные **методы** обработки данных. Если говорить образно, то объекты — это «существительные», свойства объекта — «прилагательные», а методы объекта — «глаголы».

Программные **объекты** обладают **свойствами** и могут использовать **методы** обработки данных.

Объекты могут реагировать на внешние **события**.

Классы объектов являются «шаблонами», определяющими наборы свойств, методов и событий, по которым создаются объекты. Основными классами объектов являются объекты, реализующие графический интерфейс проектов.

Объект, созданный по «шаблону» класса объектов, является **экземпляром класса** и **наследует** весь набор свойств, методов и событий данного класса. Каждый экземпляр класса объектов имеет уникальное для данного класса имя.

Например, на основе класса объектов *форма* (`Form`), который является основой для создания графического интерфейса проекта, можно создавать экземпляры объектов *форма*, которые получают имена `Form1`, `Form2` и т. д.

Свойства объекта (Properties). Каждый класс объектов обладает определенным набором свойств. Так, например, класс объектов `Form` обладает несколькими десятками различных свойств, которые определяют размеры объекта *форма*, цвет формы, положение на экране монитора и т. д. (табл. 1.1).

Таблица 1.1. **Некоторые свойства объекта *форма***

Свойство	Значение по умолчанию	Комментарий
`Name`	`Form1`	Имя объекта, используется в программном коде для обращения к объекту
`Text`	`Form1`	Текст в левом верхнем углу формы
`BackColor`	`Control`	Серый цвет фона формы
`Font`	`MS Sans Serif, обычный, 8`	Шрифт, его начертание и размер

Различные экземпляры класса объектов обладают одинаковым набором свойств, однако значения свойств у них могут различаться. Первоначальные значения свойств объектов можно установить с использованием диалогового окна *Свойства* (*Properties*) системы программирования (см. практическую работу 1.1).

Так, для объекта *форма* `Form1` можно установить требуемое значение любого свойства. Для этого необходимо выбрать свойство из списка и изменить его значение.

Значения свойств объектов можно изменять в программном коде. Для присваивания свойству объекта нового значения в левой части строки программного кода необходимо указать имя объекта и затем название свойства, которые в соответствии с правилами то-

чечной нотации разделяются между собой точкой. В правой части строки необходимо записать конкретное значение свойства:


```
Объект.Свойство = ЗначениеСвойства
```

Например, новая надпись «Первый проект» в левом верхнем углу объекта `Form1` (значение свойства `Text`) появится в результате выполнения программного кода:

```
Form1.Text = "Первый проект"
```

Методы объекта (Methods). Объекты могут использовать различные методы обработки данных. Методы имеют аргументы, которые позволяют задать параметры выполняемых действий.

Для использования метода в строке программного кода необходимо указать имя объекта и затем метод, которые в соответствии с правилами точечной нотации разделяются между собой точкой. В скобках при необходимости записываются аргументы метода, разделяемые запятыми:


```
Объект.Метод(арг1, арг2)
```

Например, с помощью метода `SIZE(x,y)` можно изменить размеры формы или элемента управления.

Событие (Event). Событие (`Event`) представляет собой действие, распознаваемое элементом управления. Событие может создаваться пользователем (например, щелчок мышью или нажатие клавиши) или быть результатом воздействия других программных объектов.

Каждый объект реагирует на определенный набор событий. Например, кнопка реагирует на щелчок (`Click`), нажатие (`MouseDown`) и отпускание (`MouseUp`) кнопки мыши или нажатие определенной клавиши на клавиатуре (`KeyPress`).

Обработчик события. Для каждого события можно запрограммировать отклик, т. е. реакцию объекта на произошедшее событие. Если пользователь производит какое-либо воздействие на элемент графического интерфейса (например, щелчок), в качестве отклика выполняется обработчик события (событийная процедура), представляющий собой программу. Программа может включать основные алгоритмические структуры (линейная, ветвление, выбор, цикл).

Для того чтобы создать заготовку обработчика события, необходимо в режиме разработки проекта осуществить двойной щелчок мышью по объекту. Например, после щелчка по кнопке

Button1 в окне *Программный код* будет создана заготовка обработчика события:

```
Private Sub Button1_Click(…)

End Sub
```

Служебные слова **Private Sub** и **End Sub** обозначают начало и конец обработчика события. Имя обработчика события Button1_Click() включает в себя имя объекта и имя события.

Далее необходимо ввести в обработчик события программный код, который реализует определенный алгоритм.

Обработчик события представляет собой программу, которая начинает выполняться после наступления определенного события.

Контрольные вопросы

1. Что можно изменить в выбранном объекте: набор свойств, набор методов, значения свойств?
2. Сосоставьте перечень объектов, которые могут быть использованы при конструировании графического интерфейса проекта.
3. На какие события реагирует кнопка?

1.2. Кодирование основных типов алгоритмических структур на языках объектно-ориентированного и процедурного программирования

Синтаксис объектно-ориентированных языков программирования Visual Basic и Gambas и языка процедурного программирования OpenOffice.org Basic очень похож, а запись операторов иногда просто совпадает. В этом можно легко убедиться, последовательно рассмотрев изображение основных алгоритмических структур в виде блок-схем и запись соответствующих операторов на языках программирования.

1.2.1. Линейный алгоритм

Существует большое количество алгоритмов, в которых команды должны быть выполнены последовательно одна за другой. Такие последовательности команд будем называть **сериями**, а алгоритмы, состоящие из таких серий, **линейными**.

Алгоритм, в котором команды выполняются последовательно одна за другой, называется **линейным алгоритмом**.

Для того чтобы сделать алгоритм более наглядным, часто используют **блок-схемы**.

Блок-схема позволяет сделать алгоритм наглядным и выделяет в алгоритме основные алгоритмические структуры (линейная, ветвление, выбор и цикл). По блок-схеме человек может легко проследить выполнение алгоритма, так как элементы блок-схемы соединены стрелками, указывающими шаги выполнения алгоритма.

Элементы алгоритма изображаются на блок-схеме с помощью различных геометрических фигур, внутри которых записывается программный код.

На блок-схеме (рис. 1.3) хорошо видна структура линейного алгоритма.

Рис. 1.3. Линейный алгоритм

Контрольные вопросы

1. Как выполняются команды в линейном алгоритме?
2. Что вы понимаете под блок-схемой?

1.2.2. Алгоритмическая структура «ветвление»

В отличие от линейных алгоритмов, в которых команды выполняются последовательно одна за другой, в алгоритмическую структуру «ветвление» входит **условие**. В зависимости от выполнения (истинности) или невыполнения (ложности) условия реализуется одна или другая последовательность команд (серий) (рис. 1.4).

Блок-схема	Языки программирования Visual Basic, Gambas и OpenOffice.org Basic

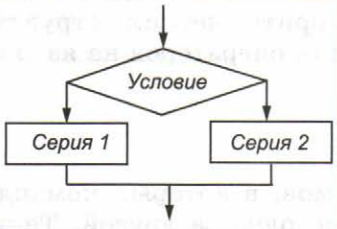
	`If` Условие `Then`
	  Серия 1
	`[Else`
	  Серия 2]
	`End If`

Рис. 1.4. Алгоритмическая структура «ветвление»

В алгоритмической структуре **«ветвление»** в зависимости от истинности или ложности **условия** выполняется одна или другая серия команд.

В условии два числа, две строки, две переменных, два арифметических или строковых выражения сравниваются между собой с использованием операций сравнения (>, <, =, >=, <=). Например: `5 > 3`, `"A" = "B"` и т. д.

Алгоритмическая структура «ветвление» может быть наглядно представлена с помощью блок-схемы. На языках Visual Basic и Gambas, а также на языке OpenOffice.org Basic ветвление кодируется с использованием **оператора условного перехода `If` ... `Then` ... `Else` ... `End If`** (Если ... То ... Иначе ... Конец Если) (см. рис. 1.4).

В операторе условного перехода после первого ключевого слова **`If`** должно быть размещено условие. Второе ключевое слово **`Then`** размещается на той же строке. Во второй строке размещается последовательность команд (*Серия 1*), которая должна выполняться, если условие истинно. На третьей строке — ключевое слово **`Else`**. На четвертой строке — последовательность команд (*Серия 2*), которая должна выполняться, если условие ложно. На пятой строке — конец инструкции ветвления **`End If`**.

В случае отсутствия серии команд, которую необходимо выполнить при ложности условия, используется сокращенная форма алгоритмической структуры «ветвление». В этом случае в операторе условного перехода отсутствует ключевое слово **`Else`** и, соответственно, последовательность команд *Серия 2* (на рис 1.4 и далее необязательные части оператора заключены в квадратные скобки). Тогда, если условие ложно, выполнение оператора условного перехода заканчивается и происходит переход на следующую строку программы.

Контрольные вопросы

1. В каком случае в алгоритмической структуре «ветвление» выполняется последовательность команд *Серия 1*? *Серия 2*?
2. В каком случае можно использовать сокращенную форму алгоритмической структуры «ветвление»?

Задания для самостоятельного выполнения

1.2. *Задание с развернутым ответом.* Начертите блок-схему алгоритмической структуры «ветвление».

1.2.3. Алгоритмическая структура «выбор»

Алгоритмическая структура «выбор» применяется для реализации ветвлений со многими вариантами серий команд. В структуру выбора входят несколько **условий**, которые последовательно проверяются. При истинности одного из условий *Условие 1*, *Условие 2* и т. д. выполняется соответствующая последовательность команд *Серия 1*, *Серия 2* и т. д. Если ни одно из условий не истинно, то выполняется последовательность команд *Серия* (рис. 1.5).

Блок-схема	Языки программирования Visual Basic, Gambas и OpenOffice.org Basic

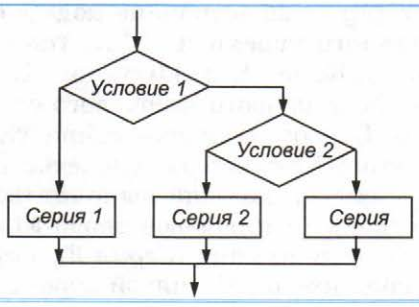
	**Select Case** Выражение
	Case Условие1
	  Серия1
	Case Условие2
	  Серия2
	[Case Else
	  Серия]
	End Select

Рис. 1.5. Алгоритмическая структура «выбор»

В алгоритмической структуре **«выбор»** выполняется одна из нескольких последовательностей команд при истинности соответствующего **условия**.

На языках Visual Basic и Gambas, а также на языке OpenOffice.org Basic оператор выбора начинается с ключевых слов **Select Case**, после которых записывается переменная или выражение. После ключевых слов **Case** записываются условия, в которых заданная переменная или выражение сравнивается с определенными значениями. При истинности одного из условий выполняется соответствующая серия команд. Если ни одно из условий не истинно, то выполняется серия команд после ключевого слова **Else**. Заканчивается оператор ключевыми словами **End Select** (см. рис. 1.5).

В случае отсутствия серии команд, которую необходимо выполнить при ложности всех условий, используется сокращенная форма алгоритмической структуры «выбор». В этом случае в операторе выбора отсутствуют ключевые слова **Case Else** и, соответственно, последовательность команд *Серия*. Тогда, если все условия ложны, выполнение оператора выбора заканчивается и происходит переход на следующую строку программы.

Контрольные вопросы

1. В каком случае в алгоритмической структуре «выбор» выполняется последовательность команд *Серия 1*? *Серия 2*? *Серия*?
2. В каком случае можно использовать сокращенную форму алгоритмической структуры «выбор»?

Задания для самостоятельного выполнения

1.3. *Задание с развернутым ответом.* Начертите блок-схему алгоритмической структуры «выбор».

1.2.4. Алгоритмическая структура «цикл»

В алгоритмическую структуру «цикл» входит серия команд, которая может выполняться многократно. Такая последовательность команд называется **телом цикла**.

Циклические алгоритмические структуры бывают двух типов:

- **цикл со счетчиком**, в котором тело цикла выполняется определенное количество раз;
- **цикл с условием**, в котором тело цикла выполняется, пока истинно условие.

В алгоритмической структуре **«цикл»** серия команд (тело цикла) может выполняться многократно.

Цикл со счетчиком. Алгоритмическая структура «цикл со счетчиком» используется, если заранее известно, какое число повторений тела цикла необходимо выполнить. Цикл со счетчиком может быть зафиксирован графически, с помощью блок-схемы, а также записан на языках Visual Basic и Gambas и на языке OpenOffice.org Basic с использованием оператора цикла **For ... Next** (рис. 1.6).

Блок-схема	Языки программирования Visual Basic, Gambas и OpenOffice.org Basic

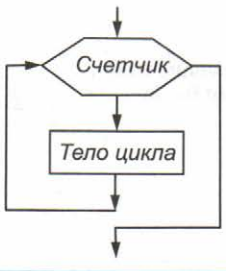
	`For Счетчик=НачЗнач To КонЗнач [Step шаг]`
	`  Тело цикла`
	`Next [Счетчик]`

Рис. 1.6. Алгоритмическая структура «цикл со счетчиком»

Синтаксис оператора **For...Next** следующий: строка, начинающаяся с ключевого слова **For**, является заголовком цикла, а строка с ключевым словом **Next** — концом цикла, между ними располагаются операторы, являющиеся телом цикла.

В начале выполнения цикла значение переменной Счетчик устанавливается равным НачЗнач. При каждом «проходе» цикла значение переменной Счетчик увеличивается на величину шага. Если оно достигает величины КонЗнач, то цикл завершается, и происходит переход на следующую строку программы.

Цикл с условием. Алгоритмическая структура «цикл с условием» используется, если заранее не известно, какое количество раз необходимо повторить тело цикла. В этом случае количество повторений тела цикла зависит от истинности условия. Цикл с условием можно отобразить с помощью блок-схемы и записать на языках Visual Basic и Gambas и на языке OpenOffice.org Basic с помощью оператора цикла **Do While...Loop** (рис. 1.7).

После ключевого слова **While** записывается условие продолжения цикла. Цикл выполняется, пока истинно условие. Как только условие примет значение «ложь», выполнение цикла закончится. Если условие продолжения цикла стоит перед телом цикла, то такой цикл называется **циклом с предусловием**.

Блок-схема	Языки программирования Visual Basic, Gambas и OpenOffice.org Basic

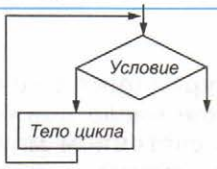
	`Do While Условие`
	`  Тело цикла`
	`Loop`

Рис. 1.7. Алгоритмическая структура «цикл с предусловием»

Если условие продолжения цикла стоит после тела цикла, то такой цикл называется **циклом с постусловием**. Цикл выполняется? пока условие ложно. Как только условие примет значение «истина», выполнение цикла закончится.

	Языки программирования Visual Basic, Gambas и OpenOffice.org Basic

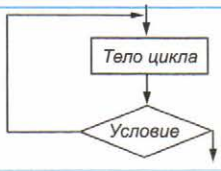
	`Do`
	`  Тело цикла`
	`Loop While Условие`

Рис. 1.8. Алгоритмическая структура «цикл с постусловием»

Контрольные вопросы

1. В каких случаях используется алгоритмическая структура «цикл со счетчиком», а в каких — алгоритмическая структура «цикл с условием»?

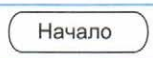
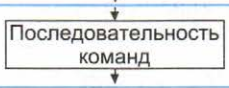
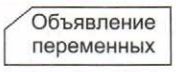
Задания для самостоятельного выполнения

1.4. *Задание с развернутым ответом.* Начертите блок-схемы алгоритмических структур «цикл со счетчиком» и «цикл с условием».

1.2.5. Блок-схемы алгоритмов

Сведем в таблицу (табл. 1.2) элементы блок-схемы (графические изображения в виде геометрических фигур) с пояснениями их назначения.

Таблица 1.2. **Элементы блок-схем**

Элемент блок-схемы	Назначение элемента блок-схемы
	Прямоугольник с закругленными углами, применяется для обозначения начала или конца алгоритма
	Параллелограмм, предназначен для описания ввода или вывода данных, имеет один вход вверху и один выход внизу
	Прямоугольник, применяется для описания линейной последовательности команд, имеет один вход вверху и один выход внизу
	Ромб, служит для обозначения условий в алгоритмических структурах «ветвление» и «выбор», имеет один вход вверху и два выхода (налево, если условие выполняется, и направо, если условие не выполняется)
	Прямоугольник со срезанным углом, применяется для объявления переменных или ввода комментариев

1.3. Переменные: тип, имя, значение

В языках Visual Basic и Gambas и в языке OpenOffice.org Basic **переменные** используются для хранения и обработки данных в программах.

Переменная задается **именем**, определяющим область оперативной памяти компьютера, в которых хранится **значение** переменной. Значениями переменных могут быть **данные** различных **типов** (целые или вещественные числа, последовательности символов, логические значения и т. д.).

Переменная в программе представлена **именем** и служит для обращения к **данным** определенного **типа**, конкретное **значение** которых хранится в ячейке оперативной памяти.

Тип переменной. Тип переменной определяется типом данных, которые могут быть значениями переменной. Значениями переменных числовых типов **Byte**, **Short**, **Integer**, **Long**, **Single**, **Double** являются числа, логического типа **Boolean** — значения «истина» (**True**) или «ложь» (**False**), строкового типа **String** — последовательности символов. Обозначения типов переменных являются ключевыми словами языка и поэтому выделяются.

Данные различных типов требуют для своего хранения в оперативной памяти компьютера различное количество ячеек (байтов) (табл. 1.3).

Таблица 1.3. **Типы переменных**

Visual Basic	Gambas	OpenOffice.org Basic	Занимаемая память	Диапазон значений
Целочисленные переменные				
`Byte`	`Byte`		1 байт	от 0 до 255
`Short`	`Short`	`Integer`	2 байта	от –32 768 до 32 767
`Integer`	`Integer`	`Long`	4 байта	от –2 147 483 648 до 2 147 483 647
`Long`	`Long`		8 байтов	от –9 223 372 036 854 775 808 до 9 223 372 036 854 775 807
Переменные с плавающей запятой				
`Single`	`Single`	`Single`	4 байта	от $-1{,}5 \cdot 10^{-45}$ до $3{,}4 \cdot 10^{38}$, 7–8 значащих цифр
`Double`	`Float`	`Double`	8 байтов	от $-5{,}0 \cdot 10^{-324}$ до $1{,}7 \cdot 10^{308}$, 15–16 значащих цифр
`Decimal`		`Float`	16 байтов	от $\pm 1{,}0 \cdot 10^{-28}$ до $\pm 7{,}9 \cdot 10^{28}$, 28–29 значащих цифр
Строковые переменные				
`String`	`String`	`String`	2 байта × количество символов	от 0 до 65 535 знаков в кодировке *Unicode*. (В языке Gambas 1 байт на символ в кодировке *ASCII*.)
Логические переменные				
`Boolean`	`Boolean`	`Boolean`	2 байта	`True` или `False`

Имя переменной. Имя переменной определяет область оперативной памяти компьютера, в которых хранится **значение** переменных. Имя каждой переменной (идентификатор) уникально и не может меняться в процессе выполнения программы. В рассматриваемых языках имя переменной может состоять из различных символов (латинские и русские буквы, цифры и т. д.), но должно обязательно начинаться с буквы и не должно включать знак точка «.». Количество

символов в имени не может быть более 1023, однако для удобства обычно ограничиваются несколькими символами.

Объявление переменных. Необходимо объявлять переменные, для того чтобы исполнитель программы (компьютер) «понимал», переменные какого типа используются в программе.

Для объявления переменной используется **оператор объявления переменных Dim.** С помощью одного оператора можно объявить сразу несколько переменных, например:

```
Dim A As Byte, B As Short, C As Single,
    D As String, E As Boolean
```

Присваивание переменным значений. Задать или изменить значение переменной можно с помощью **оператора присваивания.** При выполнении оператора присваивания переменная, имя которой указано слева от знака равенства, получает значение, которое находится справа от знака равенства. Например:

```
A = 255
B = -32768
C = 3.14
D = "информатика"
E = True
```

Значение переменной может быть задано числом, строкой или логическим значением, а также может быть представлено с помощью арифметического, строкового или логического выражения.

Проанализируем процесс выполнения программы компьютером (для определенности записанной на языке Visual Basic). После запуска проекта оператор объявления переменных **Dim** отведет в оперативной памяти для их хранения необходимое количество ячеек (табл. 1.4):

- для целой неотрицательной переменной A одну ячейку;
- для целочисленной переменной B две ячейки;
- для переменной одинарной точности C четыре ячейки;
- для строковой переменной D по две ячейки на символ;
- для логической переменной E две ячейки.

Таблица 1.4. **Значения переменных в оперативной памяти**

Имя переменной	Оперативная память	
	Номера ячеек	Значение переменной
A	1	255
B	2–3	-32768
C	4–7	3,14
D	8–29	информатика
E	30–31	True

Таким образом, в памяти для хранения значений переменных будет отведена 31 ячейка, например ячейки с 1-й по 31-ю.

Контрольные вопросы

1. В чем состоит разница между типом, именем и значением переменной?
2. Какие основные типы переменных используются в языке программирования Visual Basic? Gambas? OpenOffice.org Basic?
3. Почему рекомендуется объявлять переменные перед их использованием в программе?

Задания для самостоятельного выполнения

1.5. *Задание с кратким ответом.* Назовите количество ячеек оперативной памяти, необходимое для хранения значений переменных первых семи типов языка OpenOffice.org Basic, перечисленных в табл. 1.3.

1.4. Арифметические, строковые и логические выражения

Арифметические выражения. В состав арифметических выражений могут входить переменные числового типа, числа, знаки арифметических операций, а также математические функции.

Порядок вычисления арифметических выражений производится в соответствии с общеизвестным порядком выполнения арифметических операций (возведение в степень, умножение или деление, сложение или вычитание), который может изменяться с помощью скобок.

Строковые выражения. В состав строковых выражений могут входить переменные строкового типа, строки (последовательности символов) и строковые функции.

Над переменными строкового типа и строками может производиться операция **конкатенации**. Она объединяет строки или значения строковых переменных в единую строку. Операция конкатенации обозначается знаком «+» (который не следует путать со знаком сложения чисел в арифметических выражениях) или знаком «&».

***Логические выражения.** В состав логических выражений могут входить логические переменные, логические значения, операторы сравнения чисел и строк, а также логические операции. Логические выражения могут принимать лишь два значения: **`True`** (истина) и **`False`** (ложь).

Операторы сравнения =, <, >, <>, <= и >= сравнивают выражение в левой части оператора с выражением в правой части оператора и представляют результат в виде логического значения **True** или **False**. Например:

```
5>3 = True;    "A"="B" = False
```

Над элементами логических выражений могут производиться логические операции, которые на языках программирования обозначаются следующим образом: логическое умножение — **And**, логическое сложение — **Or** и логическое отрицание — **Not**. При записи сложных логических выражений используются скобки. Например:

```
(5>3) And ("A"="B") = False
(5>3) Or ("A"="B") = True
Not (5>3) = False
```

Контрольные вопросы

1. Какие элементы могут входить в состав арифметических, строковых и логических выражений?

1.5. Функции в языках объектно-ориентированного и процедурного программирования

Понятие функции в языках программирования близко к понятию функции в математике. Функция может иметь один или более аргументов или не иметь аргументов, в зависимости от решаемой задачи. Для каждого набора значений аргументов можно определить значение функции. В программировании говорят, что функция **возвращает** свое значение, если заданы значения ее аргументов. Функции обычно входят в состав выражений, значения которых присваиваются переменным.

Функции могут быть различных типов: математические, строковые, ввода и вывода, даты и времени и др. Тип функции определяется возможными значениями аргументов и значением функции. В математических функциях значениями как аргументов, так и функций являются числа.

Математические функции. Примеры математических функций в языке Gambas и языке OpenOffice.org Basic: синус `Sin()`, косинус `Cos()`, квадратный корень `Sqr()` и др.

В языке Visual Basic математические функции реализуются с помощью методов: синус `Math.Sin()`, косинус `Math.Cos()`, квадратный корень `Math.Sqrt()` и др.

Строковые функции (табл. 1.5). В строковых функциях аргументы и возвращаемые функциями значения могут иметь строковый тип или другой. В языке OpenOffice.org Basic и языке Visual Basic строковые функции оперируют данными в кодировке *Unicode*, а в языке Gambas — в кодировке *ASCII*.

Функция вырезания левой подстроки `Left()`. В функции вырезания подстроки (части строки) `Left(Строка, Длина)` значением функции является левая подстрока. Подстрока начинается от крайнего левого символа аргумента `Строка` и имеет количество символов, равное значению числового аргумента `Длина`.

Функция вырезания правой подстроки `Right()`. В функции вырезания подстроки `Right(Строка, Длина)` значением функции является правая подстрока. Подстрока начинается от крайнего правого символа аргумента `Строка` и имеет количество символов, равное значению числового аргумента `Длина`.

Функция вырезания произвольной подстроки `Mid()`. В функции вырезания подстроки `Mid(Строка, Позиция, Длина)` значением функции является подстрока. Подстрока начинается с символа аргумента `Строка`, позиция которого задана числовым аргументом `Позиция`, и имеет количество символов, равное значению числового аргумента `Длина`.

Функция определения длины строки `Len()`. В функции определения длины строки `Len(Строка)` аргументом является строка `Строка`, а возвращает функция числовое значение длины строки (количество символов в строке).

Функция `Asc()`. Функция `Asc(Строка)` осуществляет преобразование строки в числовой код ее первого символа. Аргументом функции является строка, а значением — число.

Функция `Chr()`. Функция `Chr(Число)` осуществляет преобразование числового кода в символ. Аргументом функции является число, а значением — символ.

Таблица 1.5. **Строковые функции и их значения**

Язык Visual Basic	Язык OpenOffice.org Basic	Значение функции
`Microsoft.VisualBasic.Len("бит")`	`Len("бит")`	3
`Microsoft.VisualBasic.Left("Килобайт",4)`	`Left("Килобайт",4)`	"Кило"
`Microsoft.VisualBasic.Right("Килобайт",4)`	`Right("Килобайт",4)`	"байт"
`Microsoft.VisualBasic.Mid("информатика",3,5)`	`Mid("информатика",3,5)`	"форма"
`Microsoft.VisualBasic.Asc("и")`	`Asc("и")`	232
`Microsoft.VisualBasic.Chr(255)`	`Chr(255)`	"я"

Функции ввода/вывода данных. В языке OpenOffice.org Basic и в языке Visual Basic для ввода данных может использоваться функция `InputBox()`, которая позволяет вводить данные с помощью диалогового окна ввода (рис. 1.9).

Рис. 1.9. Диалоговое окно ввода функции `InputBox()`

Аргументами этой функции являются две строки: `"Сообщение"` и `"Заголовок"`, а значением функции является строка, введенная пользователем в текстовое поле:

```
A = InputBox ("Сообщение","Заголовок")
```

Если пользователь введет строку в текстовое поле и щелкнет по кнопке *OK*, то значением функции станет строка, введенная в текстовое поле. Если пользователь щелкнет по кнопке *Отмена*, то значением функции станет пустая строка `""`.

В языке OpenOffice.org Basic и в языке Visual Basic для вывода данных может использоваться функция `MsgBox()`. Эта функция позволяет выводить сообщения с помощью окна сообщений, в котором можно разместить определенный набор кнопок и информационный значок о типе сообщения:

```
MsgBox("Сообщение" [,ЧисКод1+ЧисКод2][,"Заголовок"])
```

Аргумент `"Сообщение"` выводится в окне сообщений, аргумент `ЧисКод1+ЧисКод2` определяет внешний вид окна, а строка `"Заголовок"` выводится в строке заголовка окна. Последние два аргумента не являются обязательными. Необязательные части программного кода заключаются в квадратные скобки.

Например, для функции

```
MsgBox("Сообщение", 48+3, "Заголовок")
```

будет выведено следующее окно сообщений (рис. 1.10).

Рис. 1.10. Диалоговое окно сообщений функции `MsgBox()`

Значение, возвращаемое функцией `MsgBox()`, зависит от того, какая из кнопок в окне сообщений была нажата (табл. 1.6).

Таблица 1.6. **Значения функции `MsgBox()`**

Нажатая кнопка	Значения функции
OK	1
Отмена	2
Стоп	3
Повторить	4
Пропустить	5
Да	6
Нет	7

Однако в языке OpenOffice.org Basic для вывода данных часто до сих пор используется оператор **`Print`**, который выводит строки или числовые выражения, разделенные в программе запятой или точкой с запятой, в диалоговом окне.

В языках Visual Basic и Gambas для ввода и вывода данных чаще используются элементы управления графического интерфейса. Для ввода данных используется элемент управления *текстовое поле* `TextBox`, а для вывода данных — элемент управления *метка* `Label`.

Функции даты и времени. В языке OpenOffice.org Basic и в языках Visual Basic и Gambas и существуют функции даты и времени. Для определенности рассмотрим функции даты и времени, принятые в языке OpenOffice.org Basic.

Функция `Time` возвращает значение текущего времени в формате Часы:Минуты:Секунды.

Функция `DateDiff` возвращает количество дней между двумя датами, заданными в формате День/Месяц/Год.

Контрольные вопросы

1. Какой тип данных могут иметь аргументы и возвращаемые значения математических функций?
2. Какой тип данных могут иметь аргументы и возвращаемые значения строковых функций?
3. Какой тип данных могут иметь аргументы и возвращаемые значения функций ввода и вывода?
4. Какой тип данных могут иметь аргументы и возвращаемые значения функций даты и времени?

1.6. *Графические возможности объектно-ориентированного языка программирования Visual Basic

На форме и элементах управления можно рисовать линии, прямоугольники, окружности и другие графические фигуры. Для рисования необходимо определить объекты `Graphics` (*Область рисования*), `Pen` (*Перо*) и `Brush` (*Кисть*).

Область рисования. Объект `Graphics` (*Область рисования*) позволяет выбрать в качестве области рисования определенный элемент управления и обладает методами рисования графических фигур. Сначала необходимо в разделе объявления переменных определить имя объекта, например:

```
Dim Graph1 As Graphics
```

Затем в программном коде обработчика события необходимо указать определенный элемент управления в качестве области рисования. Обычно в качестве области рисования выбирается размещенное на форме графическое поле (например, `PictureBox1`):

```
Graph1 = Me.PictureBox1.CreateGraphics()
```

Имя **Me** в программном коде используется вместо имени формы (например, `Form1`), если производятся операции над самой формой.

Перо. Объект `Pen` (*Перо*) определяет цвет и ширину линии рисования. Сначала необходимо в разделе объявления переменных определить имя объекта (например, `Pen1`), установить цвет (например, красный `Color.Red`) и ширину линии в пикселях (например, 3):

```
Dim Pen1 As New Pen(Color.Red, 3)
```

Затем в программном коде обработчика события можно установить новые значения цвета и ширины линии, например:

```
Pen1.Color = Color.Green
Pen1.Width = 15
```

Кисть. Объект `Brush` (*Кисть*) определяет цвет и стиль закрашивания прямоугольников, окружностей и других замкнутых фигур. Сначала необходимо в разделе объявления переменных определить имя объекта (например, `Brush1`) и установить тип закраски и цвет (например, сплошная закраска синего цвета `SolidBrush(Color.Blue)`):

```
Dim Brush1 As New SolidBrush(Color.Blue)
```

Затем в программном коде обработчика события можно установить новый цвет закраски (например, пурпурный):

```
Brush1.Color = Color.Magenta
```

Графические методы. Графические фигуры рисуются с использованием графических методов. Замкнутые фигуры, такие как прямоугольники или эллипсы, состоят из двух частей — контура и внутренней области. Контур рисуется с использованием заданного пера, а внутренняя область закрашивается с использованием заданной кисти (табл. 1.7).

Цвет. Цвет устанавливается как значение свойства `Color`. Можно установить цвет с использованием нескольких десятков цветовых констант. Ниже приведены примеры установки зеленого цвета для объекта `Pen1` и желтого цвета для объекта `Brush1`:

```
Pen1.Color = Color.Green
Brush1.Color = Color.Yellow
```

Для установки цвета в 24-битовой палитре цветов RGB используется метод `Color.FromArgb(Red, Green, Blue)`, аргументами которого являются три числа в диапазоне от 0 до 255 (интенсивности красного, зеленого и синего цветов). Например, так можно установить пурпурный цвет для объекта `Brush1`:

```
Brush1.Color = Color.FromArgb(255,0,255)
```

Рисование текста. Метод `DrawString()` позволяет выводить текст в область рисования. Аргументами метода являются *Строка текста*, *Шрифт*, *Кисть* и координаты начала строки. Объекты *Шрифт* (например, `drawFont`) и *Кисть* (например, `drawBrush`) необходимо объявить:

```
Dim drawFont As New Font("Arial", 16)
Dim drawBrush As New SolidBrush(Color.Black)
```

Рисование текста в поле рисования можно осуществить так:

```
Graph1.DrawString("Текст", drawFont, drawBrush, 10, 10)
```

Системы координат. Рисование линий, прямоугольников и других фигур производится в компьютерной системе координат, начало которой расположено в верхнем левом углу формы или элемента управления. Ось X направлена **вправо**, а ось Y — **вниз**. Единицей измерения по умолчанию является точка (пиксель). Компьютерная система координат графического поля шириной 300 точек и высотой 200 точек приведена на рис. 1.11.

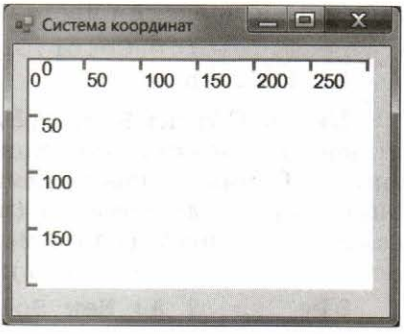


Рис. 1.11. Компьютерная система кооординат графического поля

Таблица 1.7. **Графические методы**

Фигура	Графический метод	Аргументы
Линия	`DrawLine(Pen1, X1, Y1, X2, Y2)`	*Перо* (например, `Pen1`), а также координаты концов линии `X1, Y1` и `X2, Y2`
Прямоугольник	`DrawRectangle(Pen1, X1, Y1, Width, Height)`	*Перо* (например, `Pen1`), а также координаты левого верхнего угла `X1, Y1`, ширина `Width` и высота `Height`
Закрашенный прямоугольник	`FillRectangle(Brush1, X1, Y1, Width, Height)`	*Кисть* (например, `Brush1`), а также координаты левого верхнего угла `X1, Y1`, ширина `Width` и высота `Height`
Эллипс (окружность)	`DrawEllipse(Pen1, X1, Y1, Width, Height)`	*Перо* (например, `Pen1`), а также координаты левого верхнего угла описанного прямоугольника `X1, Y1`, его ширина `Width` и высота `Height`
Закрашенный эллипс (окружность)	`FillEllipse(Brush1, X1, Y1, Width, Height)`	*Кисть* (например, `Brush1`), а также координаты левого верхнего угла описанного прямоугольника `X1, Y1`, его ширина `Width` и высота `Height`
Точка	`DrawRectangle(Pen1, X1, Y1, 1, 1)` `DrawEllipse(Pen1, X1, Y1, 1, 1)`	Аргументы `Width` и `Height` равны 1
Стирание	`Clear(Color.White)`	Аргумент – цвет

При геометрических построениях и построении графиков функций удобнее использовать математическую систему координат, начало которой обычно находится в центре области рисования. Ось X направлена вправо, а ось Y — вверх. Математическая система координат графического поля шириной 300 точек и высотой 200 точек приведена на рис. 1.12.

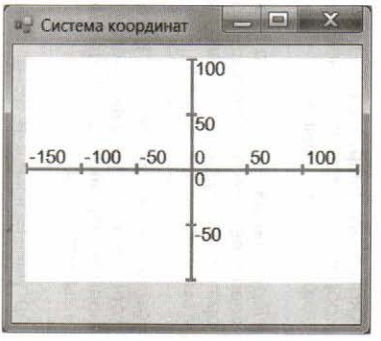


Рис. 1.12. Математическая система координат графического поля

Для преобразования компьютерной системы координат в математическую систему координат используется **метод масштабирования и поворота** осей `ScaleTransform()` и **метод сдвига** начала координат `TranslateTransform()`.

Метод

```
Graph1.ScaleTransform(1, -1)
```

обеспечивает поворот оси Y.

Метод

```
Graph1.TranslateTransform(150, -100)
```

обеспечивает сдвиг по оси X на 150 точек вправо и сдвиг по оси Y на 100 точек вниз.

Анимация. Для создания анимации (иллюзии движения на экране какого-либо объекта) применяется принцип смены кадров (изображений), как это делается в мультипликации. Для этого необходимо с определенной частотой рисовать объект в поле рисования, причем координаты объекта должны каждый раз изменяться на определенную величину.

Контрольные вопросы

1. Перечислите методы рисования графических фигур и их аргументы.
2. Каким образом можно изменить систему координат формы или графического поля?
3. Покажите на примере основные этапы создания анимации движения объекта.

Практические работы компьютерного практикума к главе 1 «Основы алгоритмизации и объектно-ориентированного программирования»

	Установить • систему программирования Basic, входящую в OpenOffice.org; • систему объектно-ориентированного программирования Visual Basic	http://ru.openoffice.org/  http://www.microsoft.com/visualstudio/ru-ru/products/2010-editions/express 
	Установить: • систему программирования Basic, входящую в OpenOffice.org; • систему объектно-ориентированного программирования Gambas	http://altlinux.org/Альт-Линукс-5.0.2-Школьный  

Практическая работа 1.1

Знакомство с системами объектно-ориентированного и процедурного программирования

Аппаратное и программное обеспечение. Компьютер с установленной операционной системой Windows или Linux.

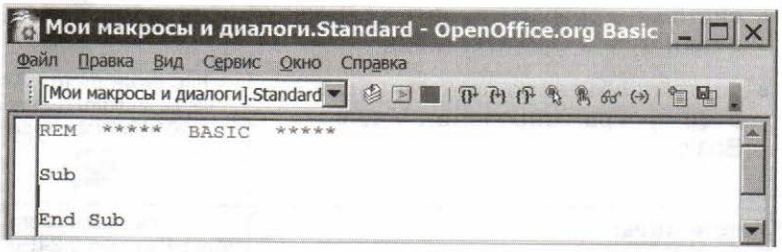
Цель работы. Знакомство с системами объектно-ориентированного и процедурного программирования.

Задание. Познакомиться с системой процедурного программирования Basic, входящей в интегрированный пакет OpenOffice.org, и с системами объектно-ориентированного программирования Visual Basic (Windows) и Gambas (Linux).

Задание. Знакомство с системой программирования OpenOffice.org Basic

1. В операционной системе Windows или Linux запустить один из компонентов интегрированного офисного приложения OpenOffice.org (например, OpenOffice.org Writer) командой [*Пуск-Все программы-OpenOffice-OpenOffice.org Writer*].

2. В меню приложения ввести команду [*Сервис-Макросы- Управление макросами-OpenOffice.org Basic*].
   Появится диалоговое окно *Макросы OpenOffice.org Basic*. Нажать кнопку *Создать*.
3. Ознакомиться с элементами диалоговых окон и командами сохранения, вставки и запуска программы на языке Basic.
   В появившемся диалоговом окне интерпретатора Basic ввести программный код на алгоритмическом языке Basic, между служебными словами **Sub** и **End Sub**.

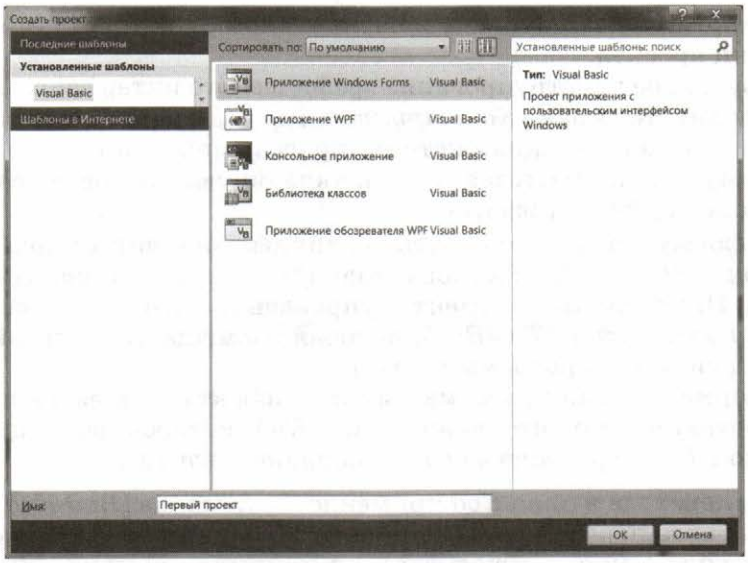


   Сохранение программы на языке Basic производится нажатием кнопки *Сохранить Basic* или командой [*Файл-Сохранить все*].
   Вставка программы на языке Basic производится нажатием кнопки *Вставить код на Basic*.
   Запуск на выполнение программы на языке Basic производится нажатием кнопки *Выполнить Basic* или командой [*Сервис-Макросы-Выполнить макрос...*].

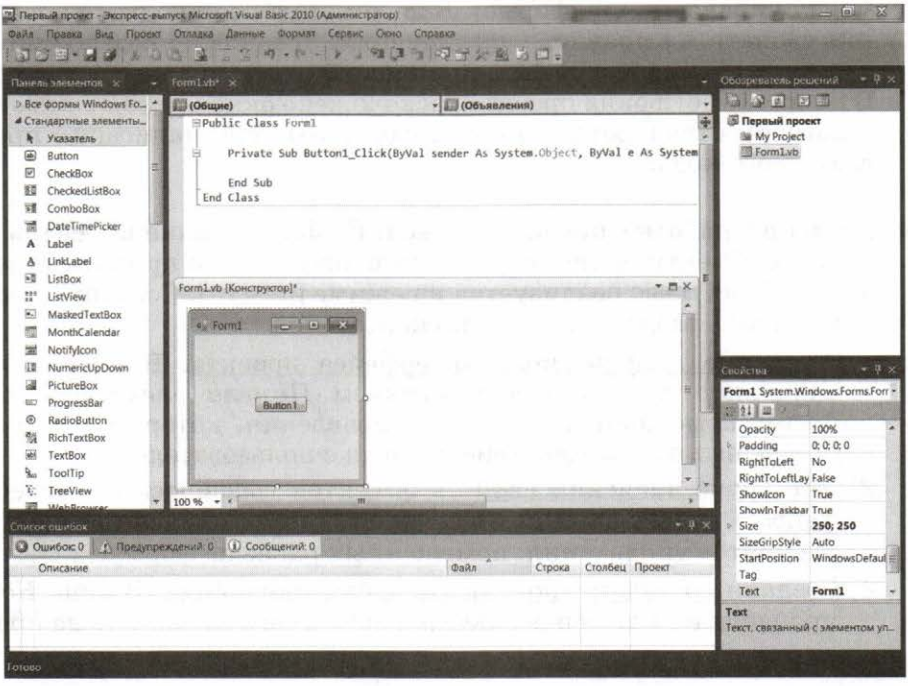
Задание. Знакомство с системой объектно-ориентированного программирования Visual Basic

Система объектно-ориентированного программирования Visual Basic имеет разные версии: Visual Basic 2003, Visual Basic Express Edition 2005, Visual Basic Express Edition 2007, Visual Basic Express Edition 2010. Ниже приводятся данные для Visual Basic Express Edition 2010.

1. Запустить систему объектно-ориентированного программирования Visual Basic командой [*Пуск-Все программы-Visual Basic 2010 Express*].
2. После запуска системы программирования ввести команду [*Файл-Создать проект...*]. В появившемся диалоговом окне *Создать проект* выбрать тип создаваемого проекта *Приложение Windows Forms* и в текстовое поле *Имя:* ввести название проекта, например *Первый проект*. Щелкнуть по кнопке *ОК*.



После этого появится окно системы программирования, включающее несколько окон.



3. Ознакомиться с составом и назначением окон и этапами разработки проекта.

Визуальное конструирование графического интерфейса проекта выполняется в окне *Конструктор форм* (*Designer*). Оно располагается в центре окна системы программирования и содержит **форму** (в данном случае `Form1`), являющуюся основой графического интерфейса проекта.

На форму можно поместить различные **элементы управления**: кнопки (`Button`), текстовые поля (`TextBox`), надписи (`Label`) и т. д. Пиктограммы элементов управления располагаются на *Панели элементов* (*ToolBox*), которая размещается в левой части окна системы программирования.

С формой связан программный код проекта, для ввода и редактирования которого служит окно *Код*, которое размещено над окном *Конструктор форм* на вкладке *Form1.vb**.

Для перехода в окно *Код* применяется команда [*Вид-Код*], а для обратного перехода в окно конструирования графического интерфейса *Конструктор форм* применяется команда [*Вид-Конструктор*].

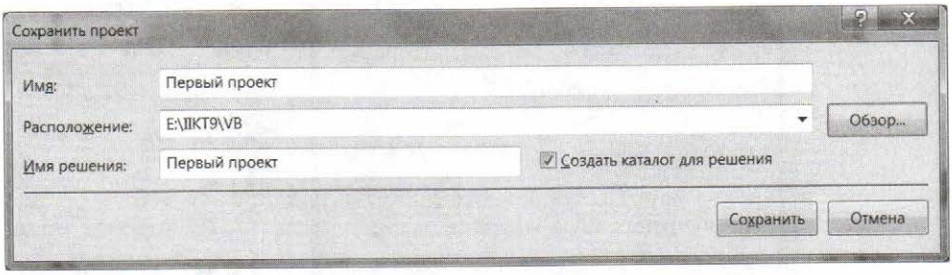
Справа располагается окно *Свойства*. Окно содержит список свойств, относящихся к выбранному объекту (форме или элементу управления на форме). В левом столбце находятся названия свойств, а в правом — их значения. Установленные по умолчанию значения могут быть изменены.

В нижней части окна проекта расположено окно *Список ошибок*, в котором отражаются ошибки, сделанные при написании программного кода.

4. **Этапы разработки проекта в Visual Basic.** Создание проектов в системе объектно-ориентированного визуального программирования Visual Basic реализуется в режиме [*design*]. Создание проекта можно разделить на несколько этапов.

1) Создание графического интерфейса проекта. В окне *Конструктор форм* с использованием *Панели элементов* на форму помещаются элементы управления, которые должны обеспечить взаимодействие проекта с пользователем.
2) Установка значений свойств объектов графического интерфейса. С помощью окна *Свойства* задаются значения свойств элементов управления, помещенных ранее на форму.
3) Создание и редактирование программного кода. В окне *Код* производится ввод и редактирование программного кода проекта.

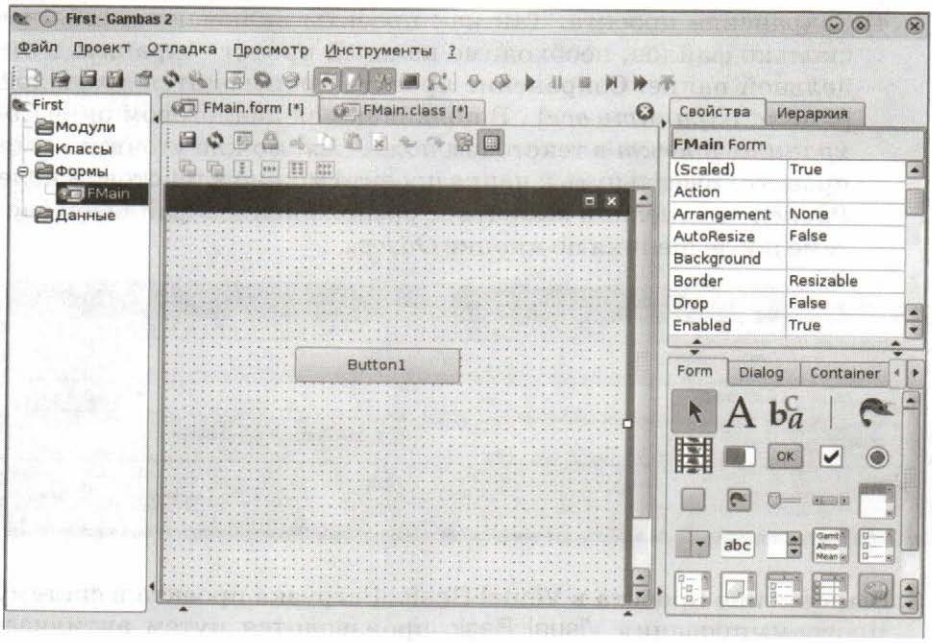
4) Сохранение проекта. Так как проекты включают в себя несколько файлов, необходимо каждый проект сохранять в отдельной папке. Сохранение проекта производится командой [*Файл-Сохранить все*] . В появившемся диалоговом окне *Сохранить проект* в текстовом поле *Имя:* можно уточнить имя проекта. Ввести путь к папке проекта можно в текстовом поле *Расположение:* или выбрать ее расположение в файловой системе после щелчка по кнопке *Обзор...* .



5. **Выполнение проекта в Visual Basic.** Загрузка проекта в систему программирования Visual Basic производится путем активизации в папке проекта основного файла проекта (файла с расширением vbproj).
Запуск проекта на выполнение производится командой [*Отладка-Начать отладку*] или щелчком по кнопке ▷ на панели инструментов окна системы программирования. После этого система программирования переходит в режим выполнения проекта [*run*], в котором редактирование графического интерфейса или программного кода невозможно. В процессе выполнения проекта производится его компиляция в приложение, которое имеет расширение exe и находится в папке ...\bin.
Для окончания выполнения проекта и перехода в режим конструирования проекта [*design*] необходимо ввести команду [*Отладка-Остановить отладку*] или щелкнуть по кнопке ■ на панели инструментов окна системы программирования.

Задание. Знакомство с системой объектно-ориентированного программирования Gambas

1. Запустить систему объектно-ориентированного программирования Gambas командой [*Пуск-Образование-Разработка-Gambas IDE (Интегрированная среда разработки Gambas)*].
После этого появится окно системы программирования, включающее несколько окон.



2. Ознакомиться с составом и назначением окон и этапами разработки проекта. Визуальное конструирование графического интерфейса проекта выполняется в центре окна системы программирования (на вкладке *FMain form*) и содержит **форму**, являющуюся основой графического интерфейса проекта.

   На форму можно поместить различные **элементы управления**: кнопки (`Button`), текстовые поля (`TextBox`), надписи (`Label`) и т. д. Пиктограммы элементов управления располагаются на *Панели объектов*, которая вызывается командой [*Просмотр-Палитра компонентов*] и размещается в правой нижней части окна системы программирования.

   С формой связан программный код проекта, для ввода и редактирования которого служит окно *Редактор кода* (размещено на вкладке *FMain class*).

   Для перехода от конструирования графического интерфейса к редактированию программного кода и обратно активизируется соответствующая вкладка.

   Справа располагается окно *Свойства*. Окно содержит список свойств, относящихся к выбранному объекту (форме или элементу управления на форме). В левом столбце находятся названия свойств, а в правом — их значения. Установленные по умолчанию значения могут быть изменены.

3. **Этапы разработки проекта в Visual Basic.** Создание проектов в системе объектно-ориентированного визуального программирования Gambas можно разделить на несколько этапов.
   1) Создание графического интерфейса проекта. В окне *Конструктор форм* с использованием *Панели объектов* на форму помещаются элементы управления, которые должны обеспечить взаимодействие проекта с пользователем.
   2) Установка значений свойств объектов графического интерфейса. С помощью окна *Свойства* задаются значения свойств элементов управления, помещенных ранее на форму.
   3) Создание и редактирование программного кода. В окне *Редактор кода* производится ввод и редактирование программного кода проекта.
   4) *Сохранение проекта.* Так как проекты включают в себя несколько файлов, необходимо каждый проект сохранять в отдельной папке. Сохранение проекта производится командой [*Файл-Сохранить проект*].
   5) *Компиляция проекта в приложение.* Компиляция проекта в приложение производится командой [*Проект-Компилировать*].

4. **Выполнение проекта в Visual Basic.** Открытие проекта в системе программирования Gambas производится путем активизации в папке проекта основного файла проекта.

   Запуск проекта на выполнение производится командой [*Отладка-Старт*] или щелчком по кнопке ▷ на панели инструментов окна системы программирования. После этого система программирования переходит в режим выполнения проекта, в котором редактирование графического интерфейса или программного кода невозможно.

   Для окончания выполнения проекта и перехода в режим конструирования проекта необходимо ввести команду [*Отладка-Стоп*] или щелкнуть по кнопке ■ на панели инструментов окна системы программирования.

Практическая работа 1.2

Разработка проекта «Переменные»

Аппаратное и программное обеспечение. Компьютер с установленной операционной системой Windows или Linux.

Цель работы. Научиться использовать переменные разных типов в системах процедурного и объектно-ориентированного программирования.

Задание. Создать проект, в котором объявить переменные различных типов, присвоить переменным A и B значения, переменным разных типов C, D и F присвоить значения арифметического выражения A/B, вывести значения переменных C, D и F.

Задание. Программа «Переменные» на языке программирования OpenOffice.org Basic

1. В операционной системе Windows или Linux запустить один из компонентов интегрированного офисного приложения OpenOffice.org (например, OpenOffice.org Writer) командой [*Пуск-Все программы-OpenOffice-OpenOffice.org Writer*].

2. В меню приложения ввести команду [*Сервис-Макросы-Управление макросами-OpenOffice-OpenOffice.org Basic*].
   Появится диалоговое окно *Макросы OpenOffice.org Basic*. Нажать кнопку *Создать*.

3. В появившемся диалоговом окне интерпретатора Basic ввести программный код на языке Basic, между служебными словами **Sub** и **End Sub**:

```
Dim A, B, C As Integer, D As Single, F As Double
Sub Variable
A=1
B=3
C=A/B
D=A/B
F=A/B
Print C, D, F
End Sub
```

4. Запустить на выполнение программу на языке Basic нажатием кнопки *Выполнить Basic*.
   Появится окно с результатами выполнения программы.

Задание. Проект «Переменные» на языках объектно-ориентированного программирования Visual Basic и Gambas

1. В операционной системе Windows запустить систему объектно-ориентированного программирования Visual Basic командой [*Пуск-Все программы-Visual Basic 2010 Express*].

Или:
в операционной системе Linux запустить систему объектно-ориентированного программирования Gambas командой [*Пуск-Образование-Разработка-Gambas IDE (Интегрированная среда разработки Gambas)*].

Создадим графический интерфейс проекта*. Основой графического интерфейса является форма (в данном случае Form1), которая отображается в окне *Конструктор форм*. На форму можно поместить различные элементы управления: метки (Label), кнопки (Button) и т. д. Пиктограммы элементов управления располагаются на *Панели элементов* (*ToolBox*), которая размещается в левой части окна системы программирования.

2. Поместить на форму:

- три метки Label1, Label2 и Label3 для вывода значений переменных;
- кнопку Button1 для запуска обработчика события.

С помощью окна *Свойства* (*Properties*) изменим установленные по умолчанию значения свойств управляющих элементов.

3. Изменить значение свойства Text:

- формы — Form1 на Переменные;
- кнопки — Button1 на Вычислить.

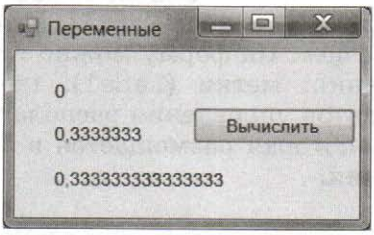
4. Объявим переменные. Щелкнем по кнопке и в окне *Код* создадим обработчик события, реализующий линейный алгоритм:

1) присвоить переменным A и B значения;
2) присвоить переменным разных типов C, D и F значения арифметического выражения A/B;
3) вывести значения переменных C, D и F на метки, присвоив их значения свойству Text.

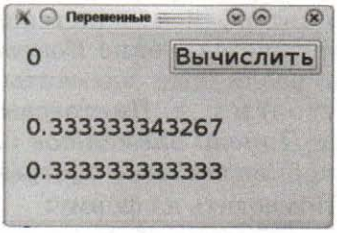
```
Dim A, B As Byte, C As Integer, D As Single,
    F As Double
Private Sub Button1_Click(…)
A = 1
B = 3
C = A/B
D = A/B
F = A/B
Label1.Text = C
Label2.Text = D
Label3.Text = F
End Sub
```

* Названия окон и панелей приведены для Visual Basic.

5. Запустить проект на выполнение. После щелчка по кнопке с надписью *Вычислить* начнет выполняться обработчик события. Результат парадоксален с точки зрения математики, на метки выводятся различные значения арифметического выражения, зависящие от типа используемой переменной.



Проект «Переменные» на языке Visual Basic



Проект «Переменные» на языке Gambas

Практическая работа 1.3

Разработка проекта «Калькулятор»

Аппаратное и программное обеспечение. Компьютер с установленной операционной системой Windows или Linux.

Цель работы. Научиться вычислять с использованием четырех арифметических действий и математических функций в системах процедурного и объектно-ориентированного программирования.

Задание. Разработать проект «Калькулятор», который позволит производить четыре арифметических действия над числами (сложение, вычитание, умножение и деление), находить синус и квадратный корень.

Задание. Программа «Калькулятор» на языке программирования OpenOffice.org Basic

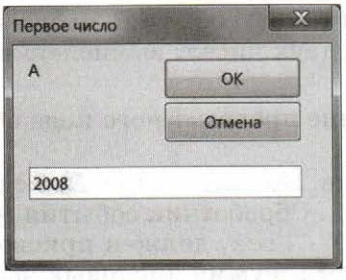
1. В операционной системе Windows или Linux запустить один из компонентов интегрированного офисного приложения OpenOffice.org (например, OpenOffice.org Writer) командой [*Пуск-Все программы-OpenOffice-OpenOffice.org Writer*].
2. В меню приложения ввести команду [*Сервис-Макросы-Управление макросами-OpenOffice.org Basic*].
   Появится диалоговое окно *Макросы OpenOffice.org Basic*. Нажать кнопку *Создать*.

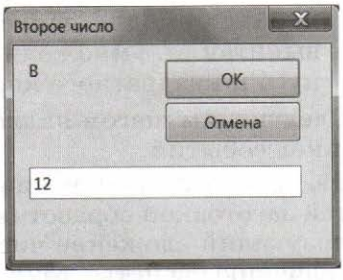
3. В появившемся диалоговом окне интерпретатора Basic ввести программный код на языке Basic, между служебными словами **Sub** и **End Sub**:

```
DIM A, B, C, D, F, K, M, N As Single
Sub Calculator
A = Val(InputBox ("A", "Первое число"))
B = Val(InputBox ("B", "Второе число"))
C = A+B
D = A-B
F = A*B
K = A/B
M = Sin(A)
N = Sqr(B)
Print "A+B= "; C, "A-B= "; D, "A*B= "; F, "A/B= "; K,
Print
Print "Sin(A)= "; M,
Print
Print "Sqr(B)= "; N
End Sub
```

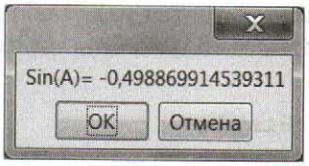
4. Запустить на выполнение программу на языке Basic нажатием кнопки *Выполнить Basic*.

В появившихся диалоговых окнах ввести первое (например, 2008) и второе (например, 12) числа и каждый раз нажимать кнопку *OK*.





Появятся окна с результатами выполнения программы.



Задание. Проект «Калькулятор» на языках объектно-ориентированного программирования Visual Basic и Gambas

1. В операционной системе Windows запустить систему объектно-ориентированного программирования Visual Basic командой [*Пуск-Все программы-Visual Basic 2010 Express*].
   Или:
   в операционной системе Linux запустить систему объектно-ориентированного программирования Gambas командой [*Пуск-Образование-Разработка-Gambas IDE (Интегрированная среда разработки Gambas)*].

Работа над проектом начинается с создания графического интерфейса, для этого на форму помещаются элементы управления.

2. Разместить на форме:
   - два текстовых поля `TextBox1` и `TextBox2` для ввода числовых данных;
   - метку `Label1` для вывода результата;
   - шесть кнопок `Button1`, `Button2`, `Button3`, `Button4`, `Button5` и `Button6` для запуска обработчиков событий: сложения, вычитания, умножения, деления двух чисел, вычисление синуса и квадратного корня числа.

Следующим шагом является создание программного кода обработчиков событий.

3. Дважды щелкнуть мышью по кнопке. Появится окно *Код* с пустой заготовкой обработчика событий. Обработчик события, реализующий сложение чисел `Button1_Click`, должен присвоить значению свойства `Text` метки `Label1` сумму числовых значений, введенных в текстовые поля `TextBox1` и `TextBox2`. Обработчики событий вычитания, умножения и деления создаются аналогично.

   Для преобразования строковых значений свойства `Text` текстовых полей в десятичные числа воспользоваться функцией `Val()`, аргументом которой является строка, а значением — число.

```
Private Sub Button1_Click(…)
Label1.Text = Val(TextBox1.Text) + Val(TextBox2.Text)
End Sub
```

4. Обработчик события, реализующий вычисление квадратного корня (аналогично — вычисления синуса), примет вид (в Gambas используется функция `Sqr()`):

```
Private Sub Button6_Click(...)
Label1.Text = Math.Sqrt(Val(TextBox1.Text))
End Sub
```

Графический интерфейс проекта можно сделать более понятным и привлекательным.

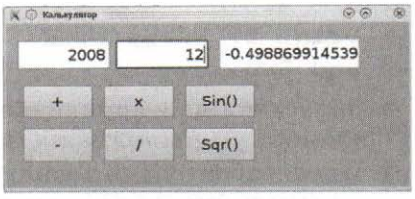
5. В режиме конструирования проекта последовательно выделить объекты графического интерфейса и с помощью диалогового окна *Свойства* установить новые значения некоторых свойств для каждого объекта:
   - для объекта *форма* `Form1` изменить значение свойства `Text` и цвет (значение свойства `BackColor`);
   - для объектов *кнопка* `Button1`, `Button2`, `Button3`, `Button4`, `Button5` и `Button6` изменить значение свойства `Text`;
   - для объектов *текстовое поле* `TextBox1` и `TextBox2` установить выравнивание текста по правому краю (значение свойства `TextAlign`) и шрифт (значение свойства `Font`);
   - для всех объектов увеличить шрифт (значение свойства `Font`).

Изменения значений свойств объектов с помощью диалогового окна *Свойства* могут производиться различными способами. В большинстве случаев нужно просто стереть старое значение свойства и ввести новое. Однако для ввода значений некоторых свойств используются **раскрывающиеся списки** или **диалоговые окна**. Так, автоматическое изменение размера метки (значение свойства `AutoSize`) устанавливается с использованием списка, а цвет фона (значение свойства `BackColor`) и шрифт (значение свойства `Font`) устанавливаются с использованием диалоговых окон.

6. Запустить проект на выполнение. Ввести число (например, 2008) в первое текстовое поле и щелкнуть по кнопке вычисления синуса.

На метку будет выведен результат.

Проект «Калькулятор» на языке Visual Basic



Проект «Калькулятор» на языке Gambas

Практическая работа 1.4
Разработка проекта «Строковый калькулятор»

Аппаратное и программное обеспечение. Компьютер с установленной операционной системой Windows или Linux.

Цель работы. Научиться применять строковые функции в системах процедурного и объектно-ориентированного программирования.

Задание. Создать проект, который позволит производить преобразования строк с использованием строковых функций.

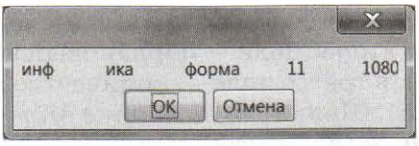
Задание. Программа «Строковый калькулятор» на языке программирования OpenOffice.org Basic

1. В операционной системе Windows или Linux запустить один из компонентов интегрированного офисного приложения OpenOffice.org (например, OpenOffice.org Writer) командой [*Пуск-Все программы-OpenOffice-OpenOffice.org Writer*].
2. В меню приложения ввести команду [*Сервис-Макросы-Управление макросами-OpenOffice.org Basic*].
   Появится диалоговое окно *Макросы OpenOffice.org Basic*. Нажать кнопку *Создать*.
3. В появившемся диалоговом окне интерпретатора Basic ввести программный код на языке Basic, между служебными словами **Sub** и **End Sub**:

```
DIM A, D, F, K As String, B, C, M, N As Integer
Sub StringCalculator
A = InputBox ("Слово", "Строка")
B = Val(InputBox ("Номер символа", "Позиция"))
C = Val(InputBox ("Количество символов", "Длина")
D = Left(A,B)
F = Right(A,B)
K = Mid (A,B,C)
M = Len(A)
N = Asc(A)
Print D, F, K, M, N
End Sub
```

4. Запустить на выполнение программу на языке Basic нажатием кнопки *Выполнить Basic*.
5. В последовательно появляющихся диалоговых окнах ввести слово (например, «информатика»), порядковый номер символа в слове (например, 3) и количество вырезаемых символов (например, 5).

Появится окно с результатами выполнения программы. Обратите внимание на то, что функция `Asc()` возвращает десятичный числовой код символа в кодировке *Unicode*.



Задание. Проект «Строковый калькулятор» на языках объектно-ориентированного программирования Visual Basic и Gambas

1. В операционной системе Windows запустить систему объектно-ориентированного программирования Visual Basic командой [*Пуск-Все программы-Visual Basic 2010 Express*].
   Или:
   в операционной системе Linux запустить систему объектно-ориентированного программирования Gambas командой [*Пуск-Образование-Разработка-Gambas IDE (Интегрированная среда разработки Gambas)*].

   Создадим графический интерфейс проекта.
2. Разместить на форме:
   - текстовое поле `TextBox1` для ввода строкового аргумента `Строка`;
   - текстовое поле `TextBox2` для ввода числового аргумента `Позиция`;
   - текстовое поле `TextBox3` для ввода числового аргумента `Длина`;
   - метку `Label1` для вывода результата;
   - шесть кнопок для запуска обработчиков событий.
3. Создадим для каждой кнопки обработчик события, реализующий одну из строковых функций. Событийная процедура, реализующая функцию `Mid()`, будет иметь следующий вид:

```
Private Sub Button1_Click(…)
Label1.Text = Microsoft.VisualBasic.Mid(TextBox1.Text,
              Val(TextBox2.Text), Val(TextBox3.Text))
End Sub
```

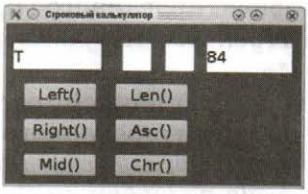
4. Событийная процедура, реализующая функцию `Asc()`, будет иметь следующий вид:

```
Private Sub Button5_Click(…)
Label1.Text = Asc(TextBox1.Text)
End Sub
```

Событийные процедуры вырезания левой и правой подстрок, а также определения длины строки и преобразования строки в символ и символа в строку создаются аналогично.

5. Запустить проект, в первое поле ввести слово (например, «информатика»), во второе поле — порядковый номер символа в слове (например, 3), в третье поле — количество вырезаемых символов (например, 5). Щелкнуть по кнопке *Mid( )*. На метке появится вырезанная подстрока «форма».
   Последовательно щелкнуть по другим кнопкам и проанализировать результаты.
   Обратите внимание на то, что в языке Gambas функция `Asc()` возвращает десятичный числовой код символа в кодировке *ASCII*.

Проект «Строковый калькулятор» на языке Visual Basic



Проект «Строковый калькулятор» на языке Gambas

Практическая работа 1.5

Разработка проекта «Даты и время»

Аппаратное и программное обеспечение. Компьютер с установленной операционной системой Windows или Linux.

Цель работы. Научиться применять функции даты и времени в системах процедурного и объектно-ориентированного программирования.

Задание. Разработать проект, в котором:

- на метку выводится текущее время;
- на метку выводится прошедшее (или оставшееся) количество дней с (до) какого-либо события.

Задание. Программа «Даты и время» на языке программирования OpenOffice.org Basic

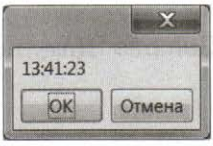
1. В операционной системе Windows или Linux запустить один из компонентов интегрированного офисного приложения OpenOffice.org (например, OpenOffice.org Writer) командой [*Пуск-Все программы-OpenOffice-OpenOffice-OpenOffice.org Writer*].

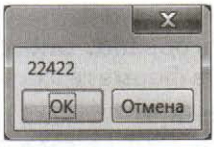
2. В меню приложения ввести команду [*Сервис-Макросы-Управление макросами-OpenOffice.org Basic*].
Появится диалоговое окно *Макросы OpenOffice.org Basic*. Нажать кнопку *Создать*.

3. В появившемся диалоговом окне интерпретатора Basic ввести программный код на языке Basic, между служебными словами **Sub** и **End Sub**.
Функция `Time` возвращает значение текущего времени в формате `Часы:Минуты:Секунды`.
Функция `DateDiff` возвращает количество дней между двумя датами, заданными в формате `День/Месяц/Год`.

```
Sub Day
Print Time
Print DateDiff("d", "3/1/1950","25/05/2011")
End Sub
```

4. Запустить на выполнение программу на языке Basic нажатием кнопки *Выполнить Basic*.
В диалоговых окнах последовательно появятся текущее время и количество дней, прошедших между двумя датами.





Задание. Проект «Даты и время» на языках объектно-ориентированного программирования Visual Basic и Gambas

1. В операционной системе Windows запустить систему объектно-ориентированного программирования Visual Basic командой [*Пуск-Все программы-Visual Basic 2010 Express*].
Или:
В оперативной системе Linux запустить систему объектно-ориентированного программирования Gambas командой [*Пуск-Образование-Разработка-Gambas IDE (Интегрированная среда разработки Gambas)*].

Создадим графический интерфейс проекта.

2. Разместить на форме:
   - метку `Label1` для вывода значений текущего времени;
   - метку `Label2` для вывода прошедшего (или оставшегося) количества дней с (до) какого-либо события;

- объект Timer1  для периодического обновления значения времени, который вызывает событие Tick через определенные пользователем интервалы времени;
- метки Label3 и Label4 для ввода поясняющего текста.

Периодичность события Tick может быть задана в свойстве Interval, измеряемом в миллисекундах (может изменяться от 0 до 65535). Для того чтобы событие Tick происходило каждую секунду, необходимо свойству Interval присвоить значение 1000.

3. Выделить объект Timer1 и с помощью диалогового окна *Свойства* присвоить свойству Interval значение 1000, а свойству Enabled — значение **True**.
4. Создать обработчик события на языке Visual Basic. Дату необходимо задать в формате День/Месяц/Год.

```
Dim Dat1, Dat2 As Date
Sub Timer1_Tick(...)
Label1.Text = TimeOfDay
Dat1 = #3/1/1950#
Dat2 = Today
Label2.Text=DateDiff(DateInterval.Day,Dat1,Dat2)
End Sub
```

5. Создать обработчик события на языке Gambas. Дату необходимо задать в формате Месяц/День/Год.

```
Public Sub Timer1_Timer()
Dim Dat1, Dat2 As Date
Label1.Text = Time
Dat1 = "1/3/1950"
Dat2 = Now
Label2.Text = DateDiff(Dat1, Dat2, gb.Day)
End
```

6. Запустить проект. На одну метку с интервалом в одну секунду будет выводиться системное время компьютера, а на другую метку — количество дней, прошедших со дня рождения до текущей даты.

Обратите внимание, что в режиме выполнения проекта объект Timer1 не виден.

Проект «Даты и время» на языке Visual Basic

Проект «Даты и время» на языке Gambas

Практическая работа 1.6
Разработка проекта «Сравнение кодов символов»

Аппаратное и программное обеспечение. Компьютер с установленной операционной системой Windows или Linux.

Цель работы. Научиться применять оператор условного перехода в системах процедурного и объектно-ориентированного программирования.

Задание. Создать проект, который позволит определять больший из числовых кодов двух символов.

Задание. Программа «Сравнение кодов символов» на языке программирования OpenOffice.org Basic

1. В операционной системе Windows или Linux запустить один из компонентов интегрированного офисного приложения OpenOffice.org (например, OpenOffice.org Writer) командой [*Пуск-Все программы-OpenOffice-OpenOffice.org Writer*].
2. В меню приложения ввести команду [*Сервис-Макросы-Управление макросами-OpenOffice.org Basic*].
   Появится диалоговое окно *Макросы OpenOffice.org Basic*. Нажать кнопку *Создать*.
3. В появившемся диалоговом окне интерпретатора Basic ввести программный код на языке Basic, между служебными словами **Sub** и **End Sub**:

```
DIM A,B As String
Sub Branch
A = InputBox ("Символ", "Ввод первого символа")
B = InputBox ("Символ", "Ввод второго символа")
Print A; Asc(A)
Print B; Asc(B)
If Asc(A)>Asc(B) Then
   Print A; Asc(A)
Else
   Print B; Asc(B)
End If
End Sub
```

4. Запустить на выполнение программу на языке Basic нажатием кнопки *Выполнить Basic*.
   В последовательно появляющихся диалоговых окнах ввести первый символ (например, «а») и второй символ (например, «Я»).

5. Появятся окна с промежуточными результатами выполнения программы.

Обратите внимание на то, что функция `Asc()` возвращает десятичный числовой код символа в кодировке *Unicode*.

6. В результате появится окно с символом, соответствующим большему числовому коду.

Задание. Проект «Сравнение кодов символов» на языках объектно-ориентированного программирования Visual Basic и Gambas

1. В операционной системе Windows запустить систему объектно-ориентированного программирования Visual Basic командой [*Пуск-Все программы-Visual Basic 2010 Express*].
Или:
в операционной системе Linux запустить систему объектно-ориентированного программирования Gambas командой [*Пуск-Образование-Разработка-Gambas IDE (Интегрированная среда разработки Gambas)*].

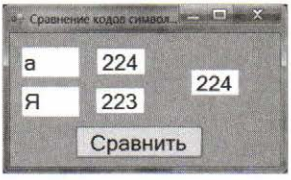
Создадим графический интерфейс проекта.

2. Разместить на форме:
   - текстовое поле `TextBox1` для ввода первого символа;
   - текстовое поле `TextBox2` для ввода второго символа;
   - метку `Label1` для вывода числового кода первого символа;
   - метку `Label2` для вывода числового кода второго символа;
   - метку `Label3` для вывода результата (большего числового кода);
   - кнопку `Button1` для запуска обработчика события.

3. Создать для кнопки обработчик события, реализующий определение числовых кодов символов и выбор большего из них:

```
Dim A, B As String
Private Sub Button1_Click(...)
A = TextBox1.Text
B = TextBox2.Text
Label1.Text = Asc(A)
Label2.Text = Asc(B)
If Asc(A)>Asc(B) Then
   Label3.Text = Asc(A)
Else
   Label3.Text = Asc(B)
End If
End Sub
```

4. Запустить проект, ввести символ в первое поле, затем другой символ — во второе.
Щелкнуть по кнопке *Сравнить*. На метках появятся числовые коды символов и больший числовой код.
Обратите внимание на то, что в языке Gambas на метки выводятся десятичные числовые коды символов в кодировке *ASCII*.



Проект «Сравнение кодов символов» на языке Visual Basic

Проект «Сравнение кодов символов» на языке Gambas

Практическая работа 1.7

Разработка проекта «Отметка»

Аппаратное и программное обеспечение. Компьютер с установленной операционной системой Windows или Linux.

Цель работы. Научиться применять оператор выбора в системах процедурного и объектно-ориентированного программирования.

Задание. Создать проект, который позволит выставлять отметку в зависимости от количества ошибок.

Задание. Программа «Отметка» на языке программирования OpenOffice.org Basic

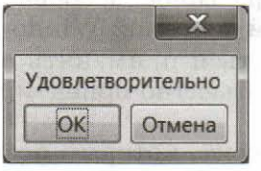
1. В операционной системе Windows или Linux запустить один из компонентов интегрированного офисного приложения OpenOffice.org (например, OpenOffice.org Writer) командой [*Пуск-Все программы-OpenOffice-OpenOffice.org Writer*].

2. В меню приложения ввести команду [*Сервис-Макросы- Управление макросами-OpenOffice.org Basic*].
   Появится диалоговое окно *Макросы OpenOffice.org Basic*. Нажать кнопку *Создать*.

3. В появившемся диалоговом окне интерпретатора Basic ввести программный код на языке Basic, между служебными словами **Sub** и **End Sub**.

```
DIM N As Byte
Sub Selection
N = Val(InputBox ("Ошибки", "Количество ошибок"))
Select Case N
Case 0
Print "Отлично"
Case 1
Print "Хорошо"
Case 2
Print "Удовлетворительно"
Case Else
Print "Плохо"
End Select
End Sub
```

4. Запустить на выполнение программу на языке Basic нажатием кнопки *Выполнить Basic*.
   В диалоговом окне ввести количество ошибок (например, 2).
   В результате появится окно с отметкой.



Задание. Проект «Отметка» на языке объектно-ориентированного программирования Visual Basic или Gambas

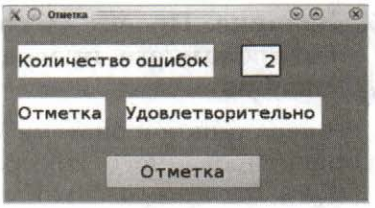
1. В операционной системе Windows запустить систему объектно-ориентированного программирования Visual Basic командой [*Пуск-Все программы-Visual Basic 2010 Express*].
   Или:
   в операционной системе Linux запустить систему объектно-ориентированного программирования Gambas командой [*Пуск-Образование-Разработка-Gambas IDE (Интегрированная среда разработки Gambas)*].

   Создадим графический интерфейс проекта.
2. Разместить на форме:
   - текстовое поле `TextBox1` для ввода количества ошибок;
   - метку `Label1` для вывода отметки;
   - кнопку `Button1` для запуска обработчика событий;
   - две метки для вывода поясняющих текстов.
3. В обработчике событий объявим переменную `N`, значением которой будет являться количество ошибок, как переменную типа **Byte** (количество ошибок не может быть отрицательным и вряд ли может быть больше 255). Присвоим переменной `N` значение свойства `Text` текстового поля `TextBox1`.
   В операторе **Select Case** в зависимости от значения переменной `N` будем присваивать значению свойства `Text` метки `Label1` определенные отметки:

```
Dim N As Byte
Private Sub Button1_Click(…)
N = TextBox1.Text
Select Case N
Case 0
  Label1.Text = "Отлично"
Case 1
  Label1.Text = "Хорошо"
Case 2
  Label1.Text = "Удовлетворительно"
Case Else
  Label1.Text = "Плохо"
End Select
End Sub
```

4. Запустить проект на выполнение, ввести в текстовое поле количество ошибок и щелкнуть по кнопке *Отметка*. На метку будет выведена соответствующая отметка.

Проект «Отметка»
на языке Visual Basic



Проект «Отметка»
на языке Gambas

Практическая работа 1.8

Разработка проекта «Коды символов»

Аппаратное и программное обеспечение. Компьютер с установленной операционной системой Windows или Linux.

Цель работы. Научиться применять оператор цикла со счетчиком в системах процедурного и объектно-ориентированного программирования.

Задание. Создать проект, который должен выводить в поле списка числовые коды символов и соответствующие им символы.

Задание. Программа «Коды символов» на языке программирования OpenOffice.org Basic

1. В операционной системе Windows или Linux запустить один из компонентов интегрированного офисного приложения OpenOffice.org (например, OpenOffice.org Writer) командой [*Пуск-Все программы-OpenOffice-OpenOffice.org Writer*].
2. В меню приложения ввести команду [*Сервис-Макросы-Управление макросами-OpenOffice.org Basic*].

   Появится диалоговое окно *Макросы OpenOffice.org Basic*. Нажать кнопку *Создать*.
3. В появившемся диалоговом окне интерпретатора Basic ввести программный код на языке Basic, между служебными словами **Sub** и **End Sub**:

```
DIM N As Integer
Sub Cycle1
For N = 1025 To 1279
  Print N; "-"; Chr(N)
Next N
End Sub
```

В кодировке *Unicode* знаки кириллицы начинаются с десятичного кода 1025, а заканчиваются десятичным кодом 1279 (в кириллицу входят не только буквы русского алфавита). Учтем это при присваивании начального и конечного значений счетчика цикла.

4. Запустить на выполнение программу на языке Basic нажатием кнопки *Выполнить Basic*.
   В диалоговых окнах последовательно появятся десятичные коды символов и соответствующие им символы в кодировке *Unicode*.

Задание. Проект «Коды символов» на языках объектно-ориентированного программирования Visual Basic и Gambas

1. В операционной системе Windows запустить систему объектно-ориентированного программирования Visual Basic командой [*Пуск-Все программы-Visual Basic 2010 Express*].
   Или:
   в операционной системе Linux запустить систему объектно-ориентированного программирования Gambas командой [*Пуск-Образование-Разработка-Gambas IDE (Интегрированная среда разработки Gambas)*].
   Создадим графический интерфейс проекта.
2. Разместить на форме:
   - поле списка `ListBox1` для вывода числовых кодов символов;
   - поле списка `ListBox2` для вывода соответствующих им символов в кодировке *Windows*;
   - кнопку `Button1` для запуска обработчика событий.
3. Создать обработчик события на языке Visual Basic, в котором в качестве счетчика цикла использовать целочисленную переменную `N`. В кодировке *Windows* первые 33 кода (десятичные коды с 0 по 32) соответствуют не знакам, а клавишам клавиатуры (клавиши управления курсором, *Пробел*, *Ввод* и др.). Поэтому воспользуемся циклом со счетчиком с шагом –1, для того чтобы выводить на форму символы, начиная с наибольшего числового кода 255.
   Для преобразования числового кода в символ использовать функцию `Chr()`, аргументом которой является число (от 33 до 255), а значением — символ. В теле цикла числовые коды сим-

волов и соответствующие им символы будут выводиться в поля списков с помощью метода `Items.Add()`.

```
Dim N As Integer
Sub Button1_Click(…)
For N = 255 To 33 Step -1
  ListBox1.Items.Add(N)
  ListBox2.Items.Add(Chr(N))
Next N
End Sub
```

4. Создать обработчик события на языке Gambas.
Для вывода символов в кодировке *ASCII* воспользуемся циклом со счетчиком с шагом –1, для того чтобы выводить на форму символы, начиная с наибольшего числового кода 126.
Для преобразования числового кода в символ на языке Gambas использовать функцию `Chr()`, аргументом которой является число (от 33 до 126), а значением — символ. В теле цикла числовые коды символов и соответствующие им символы будут выводиться в поля списков с помощью метода `Add()`.

```
Public Sub Button1_Click()
Dim N As Integer
For N = 126 To 33 Step -1
  ListBox1.Add(N)
  ListBox2.Add(Chr(N))
Next
End
```

5. Запустить проект на выполнение, щелкнув по кнопке *Пуск*.
В поля списка будут выведены последовательности числовых кодов символов и соответствующих им символов. С помощью полос прокрутки можно ознакомиться со всеми кодами и их символами.

Проект «Коды символов» на языке Visual Basic

Проект «Коды символов» на языке Gambas

Практическая работа 1.9

Разработка проекта «Слово-перевертыш»

Аппаратное и программное обеспечение. Компьютер с установленной операционной системой Windows или Linux.

Цель работы. Научиться применять оператор цикла с предусловием в системах процедурного и объектно-ориентированного программирования.

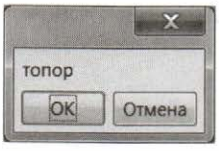
Задание. Создать проект преобразования введенного слова в слово-перевертыш, т. е. в слово с обратным порядком символов.

Задание. Программа «Слово-перевертыш» на языке программирования OpenOffice.org Basic

1. В операционной системе Windows или Linux запустить один из компонентов интегрированного офисного приложения OpenOffice.org (например, OpenOffice.org Writer) командой [*Пуск-Все программы-OpenOffice-OpenOffice.org Writer*].
2. В меню приложения ввести команду [*Сервис-Макросы- Управление макросами-OpenOffice.org Basic*].
   Появится диалоговое окно *Макросы OpenOffice.org Basic*. Нажать кнопку *Создать*.
3. В появившемся диалоговом окне интерпретатора Basic ввести программный код на языке Basic, между служебными словами **Sub** и **End Sub**:

```
DIM N As Integer, A,B,S As String
Sub Cycle2
A = InputBox("Слово", "Исходное слово")
N=1
Do While N<=Len(A)
  S = Mid(A, N, 1)
  B = S+B
  N = N+1
Loop
Print B
End Sub
```

4. Запустить на выполнение программу на языке Basic нажатием кнопки ▷ *Выполнить Basic*.
   В диалоговом окне появится слово-перевертыш.



Задание. Проект «Слово-перевертыш» на языках объектно-ориентированного программирования Visual Basic и Gambas

1. В операционной системе Windows запустить систему объектно-ориентированного программирования Visual Basic командой [*Пуск-Все программы-Visual Basic 2010 Express*].
   Или:
   в операционной системе Linux запустить систему объектно-ориентированного программирования Gambas командой [*Пуск-Образование-Разработка-Gambas IDE (Интегрированная среда разработки Gambas)*].

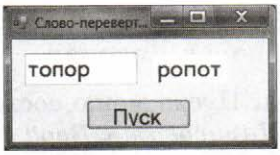
   Разработаем графический интерфейс проекта.

2. Разместить на форме:
   - текстовое поле `TextBox1` для ввода исходного слова;
   - метку `Label1` для вывода слова-перевертыша;
   - кнопку `Button1` для создания обработчика событий.

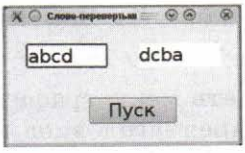
3. Создать обработчик события.
   В обработчике события цикл с предусловием будет выполняться, пока справедливо условие `N <= Len(TextBox1.Text)`, т. е. пока значение переменной `N` меньше или равно количеству символов в слове. (Количество символов во введенном слове является значением строковой функции `Len()`.)
   В цикле символы последовательно вырезаются из введенного слова в прямой последовательности (слева направо) с использованием функции вырезания подстроки из строки `Mid(TextBox1.Text, N, 1)` и присваиваются строковой переменной `S`. Затем вырезанные символы (значения переменной `S`) в обратной последовательности (справа налево) присваиваются свойству `Label1.Text`, значением которого после завершения цикла будет слово-перевертыш.

```
Dim S As String, N As Byte
Private Sub Button1_Click(…)
Label1.Text = ""
N = 1
Do While N<=Len(TextBox1.Text)
  S = Mid(TextBox1.Text, N, 1)
  Label1.Text = S+Label1.Text
  N = N+1
Loop
End Sub
```

4. Запустить проект на выполнение, ввести в текстовое поле исходное слово и щелкнуть по кнопке *Пуск*. На метку будет выведено слово-перевертыш.



Проект «Слово-перевертыш» на языке Visual Basic



Проект «Слово-перевертыш» на языке Gambas

*Практическая работа 1.10

Разработка проекта «Графический редактор»

Аппаратное и программное обеспечение. Компьютер с установленной операционной системой Windows.

Цель работы. Научиться применять графические методы в объектно-ориентированном программировании.

Задание. Создать проект, который позволит рисовать линии, прямоугольники и окружности, заданным цветом.

Задание. Проект «Графический редактор» на языке объектно-ориентированного программирования Visual Basic

1. В операционной системе Windows запустить систему объектно-ориентированного программирования Visual Basic командой [*Пуск-Все программы-Visual Basic 2010 Express*].

   Создадим графический интерфейс проекта.

2. Поместить на форму:
   - графическое поле `PictureBox1`, которое будет использоваться в качестве области рисования;
   - четыре текстовых поля `TextBox1`, `TextBox2`, `TextBox3` и `TextBox4` для ввода значений переменных `X1`, `Y1` и `X2`, `Y2`, содержащих координаты графических фигур;
   - три текстовых поля `TextBox5`, `TextBox6` и `TextBox7` для ввода значений переменных `Red`, `Green`, `Blue`, содержащих числовые коды цветов;
   - семь меток для вывода поясняющих текстов;
   - элемент управления `MenuStrip1` для создания меню проекта;
   - элемент управления `ColorDialog1` для выбора цвета в диалоговом окне.

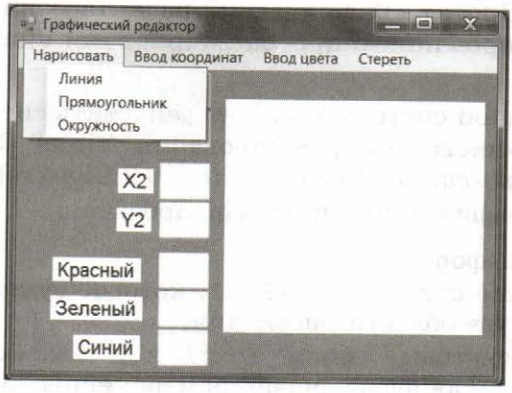
3. Объявим объекты `Graphics` (*Область рисования*) и `Pen` (*Перо*), а также переменные, которые будут использоваться в проекте:

```
Dim Graph1 As Graphics
Dim Pen1 As New Pen(Color.Red, 3)
Dim X1, X2, Y1, Y2, Red, Green, Blue As Integer
```

4. Создать меню графического редактора. Пусть меню состоит из четырех заголовков верхнего уровня: *Нарисовать*, *Ввод координат*, *Ввод цвета* и *Стереть*. В пункт *Нарисовать* входят команды *Линия*, *Прямоугольник* и *Окружность*.

Нарисовать ***Ввод координат*** ***Ввод цвета*** ***Стереть***
Линия
Прямоугольник
Окружность

Для создания меню используется управляющий элемент *MenuStrip1* (*Редактор меню*).
В редакторе меню, появившемся в левом верхнем углу формы, создать заголовки первого уровня (в поле *Type Here* внести пункты меню).
Для создания подпункта меню перейти на следующую строку в редакторе меню.



5. Создать программный код обработчика события, который реализует ввод координат графических фигур. Для этого щелкнуть по пункту меню *Ввод координат* и в появившуюся заготовку ввести программный код. Для преобразования строкового значения в число использовать функцию `Val()`. Для большей понятности программного кода ввести в него комментарии, которые начинаются с символа апостроф «`'`».

```
'Ввод координат
Private Sub ВводКоординатToolStripMenuItem_Click(...)
Graph1 = Me.PictureBox1.CreateGraphics()
X1 = Val(TextBox1.Text)
Y1 = Val(TextBox2.Text)
X2 = Val(TextBox3.Text)
Y2 = Val(TextBox4.Text)
End Sub
```

6. Создать программный код обработчика события для ввода интенсивностей базовых цветов в системе RGB. Для этого щелкнуть по пункту меню *Ввод цвета* и в появившуюся заготовку ввести программный код:

```
'Цвет
Private Sub ВводЦветаToolStripMenuItem_Click(...)
Red = Val(TextBox5.Text)
Green = Val(TextBox6.Text)
Blue = Val(TextBox7.Text)
End Sub
```

7. Создать программный код обработчика события, который реализует рисование линии. Для этого щелкнуть по пункту меню *Линия* и в появившуюся заготовку ввести программный код. Цвет линии задать с помощью цветовой константы (в данном случае `Red`):

```
'Линия
Private Sub ЛинияToolStripMenuItem_Click(...)
Pen1.Color = Color.Red
Graph1.DrawLine(Pen1, X1, Y1, X2, Y2)
End Sub
```

8. Создать программный код обработчика события, реализующего рисование прямоугольника. Для этого щелкнуть по пункту меню *Прямоугольник* и в появившуюся заготовку ввести программный код.

Цвет контура задать с помощью функции `FromArgb(Red, Green,Blue)`, аргументами которой являются интенсивности базовых цветов (красного, зеленого и синего).

```
'Прямоугольник, где аргумент Width=X2-X1,
'а аргумент Height=Y2-Y1
Private Sub ПрямоугольникToolStripMenuItem_Click(...)
Pen1.Color = Color.FromArgb(Red, Green, Blue)
Graph1.DrawRectangle(Pen1, X1, Y1, X2-X1, Y2-Y1)
End Sub
```

9. Создать программный код обработчика события, реализующего рисование окружности. Для этого щелкнуть по пункту меню *Окружность* и в появившуюся заготовку ввести программный код.

   Цвет контура задать с помощью диалога `ColorDialog1`, который предоставляет для выбора цвета диалоговое окно *Цвет*. Оно будет выведено с помощью метода `ShowDialog()`.


```
'Окружность
Private Sub ОкружностьToolStripMenuItem_Click(…)
ColorDialog1.ShowDialog()
Pen1.Color = ColorDialog1.Color
Graph1.DrawEllipse(Pen1, X1, Y1, X2-X1, Y2-Y1)
End Sub
```

10. Предусмотреть стирание неудачно нарисованного — создать программный код обработчика события, реализующего стирание. Для этого щелкнуть по пункту меню *Стереть* и в появившуюся заготовку ввести программный код:

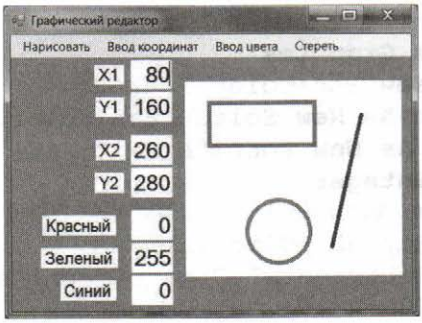
```
'Стирание
Private Sub СтеретьToolStripMenuItem_Click(…)
Graph1.Clear(Color.White)
End Sub
```

11. Запустить проект. Ввести в текстовые поля значения координат графической фигуры и щелкнуть по пункту меню *Ввод координат*.

    Ввести в текстовые поля интенсивности базовых цветов и щелкнуть по кнопке *Ввод цвета*.

    Нарисовать графическую фигуру, щелкнув по пункту меню *Линия*. Такие же действия повторить для прямоугольника и окружности.

    При необходимости очищать поле рисования, щелкая по пункту меню *Стереть*.



Проект «Графический редактор» на языке Visual Basic

*Практическая работа 1.11

Разработка проекта «Системы координат»

Аппаратное и программное обеспечение. Компьютер с установленной операционной системой Windows.

Цель работы. Научиться создавать различные системы координат в системах объектно-ориентированного программирования.

Задание. Создать проект, который обеспечит рисование осей и вывод шкалы в компьютерной системе координат (см. рис. 1.10) и математической системе координат (см. рис. 1.11).

Задание. Проект «Системы координат» на языке объектно-ориентированного программирования Visual Basic

1. В операционной системе Windows запустить систему объектно-ориентированного программирования Visual Basic командой [*Пуск-Все программы-Visual Basic 2010 Express*].

Создадим графический интерфейс проекта.

2. Поместить на форму:
   - графическое поле `PictureBox1`, которое будет использоваться в качестве области рисования;
   - кнопки `Button1` и `Button2` для запуска обработчиков событий.

Установим размеры графического поля `PictureBox1`.

3. Присвоить свойству `Size` значение `300;200`.

4. Создать обработчик события, реализующий рисование осей и печать шкал в компьютерной системе координат:

```
'Объявление переменных
Dim Graph1 As Graphics
Dim Pen1 As New Pen(Color.Red, 3)
Dim drawBrush As New SolidBrush(Color.Black)
Dim drawFont As New Font("Arial", 10)
Dim X, Y As Integer
'Обработчик события
Private Sub Button1_Click(…)
Graph1 = Me.PictureBox1.CreateGraphics()
Graph1.Clear(Color.White)
'Рисование шкал компьютерной системы координат
Graph1.DrawLine(Pen1, 0, 0, 300, 0) 'Ось X
Graph1.DrawLine(Pen1, 0, 0, 0, 200) 'Ось Y
For X = 0 To 300 Step 50 'Засечки на оси X
  Graph1.DrawLine(Pen1, X, 0, X, 10)
Next X
For Y = 0 To 200 Step 50 'Засечки на оси Y
  Graph1.DrawLine(Pen1, 0, Y, 10, Y)
Next Y
'Вывод шкалы оси X
For X = 0 To 300 Step 50
  Graph1.DrawString(X, drawFont, drawBrush, X, 10)
Next X
'Вывод шкалы оси Y
For Y = 0 To 200 Step 50
  Graph1.DrawString(Y, drawFont, drawBrush, 10, Y)
Next Y
End Sub
```

5. Создать обработчик события, реализующий рисование осей и вывод шкал в математической системе координат:

```
Private Sub Button2_Click(…)
Graph1 = Me.PictureBox1.CreateGraphics()
Graph1.Clear(Color.White)
'Вывод шкал математической системы координат
'в компьютерной системе координат
For X = -150 To 150 Step 50
  Graph1.DrawString(X, drawFont, drawBrush,
                    X + 150, 80)
Next X
For Y = 0 To 200 Step 50
  Graph1.DrawString(Y - 100, drawFont, drawBrush, 150,
                    200 - Y)
Next Y
```

```
'Преобразование компьютерной системы координат
'в математическую систему координат
Graph1.ScaleTransform(1, -1) 'Поворот оси Y
Graph1.TranslateTransform(150, -100) 'Сдвиг по осям X и Y
'Рисование осей с засечками в математической
'системе координат
Graph1.DrawLine(Pen1, -150, 0, 150, 0) 'Ось X
Graph1.DrawLine(Pen1, 0, -100, 0, 100) 'Ось Y
For X = -150 To 150 Step 50 'Засечки на оси X
  Graph1.DrawLine(Pen1, X, -5, X, 5)
Next X
For Y = -100 To 100 Step 50 'Засечки на оси Y
  Graph1.DrawLine(Pen1, -5, Y, 5, Y)
Next Y
End Sub
```

6. Запустить проект.

*Практическая работа 1.12
Разработка проекта «Анимация»

Аппаратное и программное обеспечение. Компьютер с установленной операционной системой Windows.

Цель работы. Научиться создавать анимацию в системах объектно-ориентированного программирования.

Задание. Разработать проект, в котором реализуется «полет бабочки». Для создания иллюзии взмаха крыльями два изображения бабочки («с развернутыми крыльями» и «со свернутыми крыльями») с определенной частотой выводятся в поле рисования. Для создания иллюзии движения при каждом взмахе координаты изображения изменяются на определенную величину.

Задание. Проект «Анимация» на языке объектно-ориентированного программирования Visual Basic

1. Запустить систему объектно-ориентированного программирования Visual Basic командой [*Пуск-Все программы-Visual Basic 2010 Express*].

Создадим графический интерфейс проекта.

2. Поместить на форму:
   - графическое поле `PictureBox1`, которое будет использоваться в качестве области рисования;
   - объект `Timer1`, для периодического вывода изображения на графическое поле, который вызывает событие `Tick` через определенные пользователем интервалы времени.

Периодичность события Tick может быть задана в свойстве Interval, измеряемом в миллисекундах (может изменяться от 0 до 65535).

3. Выделить объект Timer1 и с помощью диалогового окна *Свойства* присвоить свойству Interval значение 100, а свойству Enabled — значение **True**.

4. Создать заготовку обработчика события щелчком по объекту Timer1:

```
Private Sub Timer1_Tick(…)

End Sub
```

5. Выбрать в качестве области рисования графическое поле PictureBox1.

Использовать в качестве изображения бабочки «с развернутыми крыльями» графический файл bfly1.bmp, а изображения «со свернутыми крыльями» — графический файл bfly2.bmp.

Объявить графические объекты Image1 и Image2 и создать их из файлов с помощью функции Image.FromFile(). Аргументом функции является путь к графическому файлу, а также его имя.

Для того чтобы при каждом событии Tick выводить попеременно то одно, то другое изображение бабочки, использовать логическую переменную flg1. При каждом событии с помощью оператора **If...Then...Else** в однострочной форме будем менять ее значение с **True** на **False** или, наоборот, с **False** на **True**. Затем с использованием оператора **If...Then...Else...End If** в многострочной форме выводить в зависимости от значения логической переменной flg1 то или иное изображение бабочки.

Вывод изображения на область рисования осуществлять с помощью метода DrawImage(Image1, X, Y), аргументами которого являются изображение и координаты его левого верхнего угла на области рисования.

Для реализации перемещения из нижнего левого угла графического поля в правый верхний угол сместить ось координат *Y* вниз на 200 точек и при каждом событии увеличивать координату *X* и уменьшать координату *Y*.

В результате обработчик события примет следующий вид:

```
Dim Graph1 As Graphics
Dim Image1, Image2 As Image
Dim X, Y As Single
Dim flg1 As Boolean
```

```
Private Sub Timer1_Tick(…)
Graph1 = Me.PictureBox1.CreateGraphics()
Image1 = Image.FromFile("\INF9\VB\Анимация\bfly1.bmp")
Image2 = Image.FromFile("\INF9\VB\Анимация\bfly2.bmp")
If flg1 Then flg1 = False Else flg1 = True
Graph1.TranslateTransform(0, 200)
If flg1 Then
   Graph1.DrawImage(Image1, X, Y)
Else
   Graph1.DrawImage(Image2, X, Y)
End If
X = X+5
Y = Y-5
End Sub
```

6. Запустить проект. В графическом поле начнется «порхание» бабочки из нижнего левого угла графического поля в правый верхний угол.
Для изменения частоты «порхания» необходимо изменить у объекта `Timer1` значение свойства `Interval`. Для изменения скорости полета необходимо изменить шаг изменения координат.

Проект «Анимация» на языке Visual Basic

Глава 2
МОДЕЛИРОВАНИЕ И ФОРМАЛИЗАЦИЯ

2.1. Окружающий мир как иерархическая система

Микро-, макро- и мегамир. Мы живем в **макромире**, т. е. в мире, который состоит из объектов, по своим размерам сравнимых с человеком. Обычно макрообъекты разделяют на неживые (камень, льдина, бревно и т. д.), живые (растения, животные, человек) и искусственные (здания, средства транспорта, станки и механизмы, компьютеры и т. д.).

Макрообъекты состоят из молекул и атомов, которые, в свою очередь, состоят из элементарных частиц, чьи размеры чрезвычайно малы. Этот мир называется **микромиром**.

Мы живем на планете Земля, которая входит в Солнечную систему, Солнце вместе с сотнями миллионов других звезд образует нашу галактику Млечный Путь, а галактики образуют Вселенную. Все эти объекты имеют громадные размеры и образуют **мегамир**.

Все многообразие объектов мега-, макро- и микромира состоит из **вещества**, при этом все материальные объекты взаимодействуют друг с другом и поэтому обладают **энергией**. Поднятое над поверхностью Земли тело обладает механической энергией, нагретый чайник — тепловой, заряженный проводник — электрической, а ядра атомов — атомной.

Окружающий мир можно представить в виде иерархического ряда объектов: элементарных частиц, атомов, молекул, макротел, звезд и галактик. При этом на уровнях молекул и макротел в этом иерархическом ряду образуется ответвление — другой ряд, связанный с живой природой.

В живой природе также существует иерархия: одноклеточные — растения и животные — популяции животных.

Вершиной эволюции жизни на Земле является человек, а он не может жить вне общества.

Каждый человек в отдельности и общество в целом изучают окружающий мир и накапливают знания, на основании которых создаются искусственные объекты.

Всё вышесказанное можно отобразить в виде схемы на рис. 2.1.

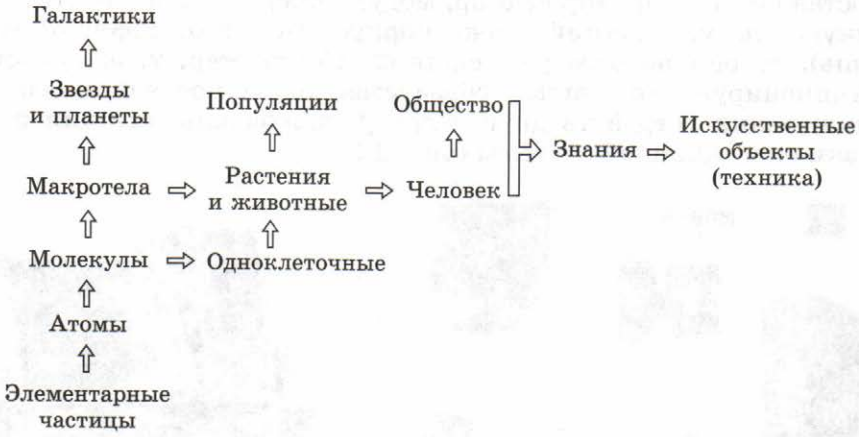


Рис. 2.1. Иерархическая система объектов окружающего мира

Системы и элементы. Каждый объект состоит из других объектов, т. е. представляет собой **систему**. Вместе с тем каждый объект может входить в качестве **элемента** в систему более высокого структурного уровня. Является ли объект системой или элементом системы, зависит от точки зрения (целей исследования).

Система состоит из объектов, которые называются **элементами системы.**

Например, атом водорода можно рассматривать как систему, так как он состоит из элементов — положительно заряженного протона и отрицательно заряженного электрона.

Вместе с тем атом водорода входит в молекулу воды, т. е. является элементом системы более высокого структурного уровня (рис. 2.2).

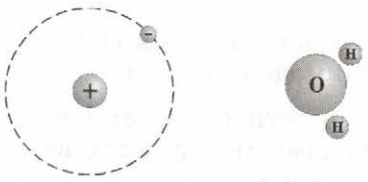


Рис. 2.2. Атом водорода и молекула воды

Целостность системы. Необходимым условием существования системы является ее **целостное функционирование.** Система является не набором отдельных объектов, а совокупностью взаимосвязанных элементов.

Например, если сложить в кучу устройства, которые входят в состав компьютера (процессор, модули оперативной памяти, системную плату, жесткий диск, корпус, монитор, клавиатуру и мышь), то они не образуют систему. Компьютер, т. е. целостно функционирующая система, образуется только после физического подключения устройств друг к другу, включения питания и загрузки операционной системы (рис. 2.3).

Рис. 2.3. Отдельные объекты (устройства) и целостная система (компьютер)

Если из системы удалить хотя бы один элемент, то она может перестать функционировать. Так, если удалить одно из устройств компьютера (например, процессор), то он выйдет из строя, т. е. прекратит свое существование как система.

Взаимосвязь элементов в системах может иметь различную природу. В неживой природе взаимосвязь элементов осуществляется с помощью физических взаимодействий:

- в системах мегамира (например, в Солнечной системе) элементы взаимодействуют между собой посредством сил всемирного тяготения;
- в макротелах происходит электромагнитное взаимодействие между атомами;
- в атомах элементарные частицы связаны ядерными и электромагнитными взаимодействиями.

В живой природе целостность организмов обеспечивается химическими взаимодействиями между клетками, в обществе — социальными связями и отношениями между людьми, в технике — функциональными связями между устройствами и т. д.

Свойства системы. Каждая система обладает определенными свойствами, которые в первую очередь зависят от набора составляющих ее элементов. Так, свойства химических элементов зависят от строения их атомов.

Атом водорода состоит из двух элементарных частиц (протона и электрона), и соответствующий химический элемент является газом.

Атом лития состоит из трех протонов, четырех нейтронов и трех электронов, и соответствующий химический элемент является щелочным металлом (рис. 2.4).

Рис. 2.4. Атом водорода и атом лития

Свойства системы зависят также от **структуры** системы, т. е. от типа отношений и связей элементов системы между собой. Если системы состоят из одинаковых элементов, но обладают разными структурами, то их свойства могут существенно различаться. Например, алмаз, графит и углеродная нанотрубка состоят из одинаковых атомов (атомов углерода), однако способы связей между атомами (кристаллические решетки) существенно различаются (рис. 2.5).

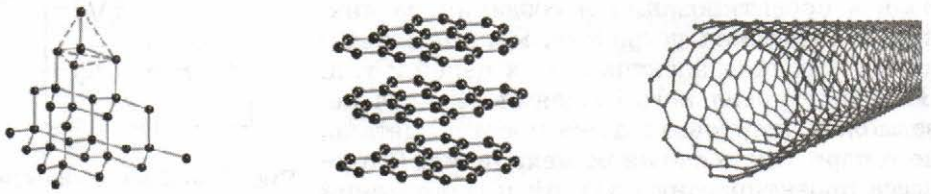


Рис. 2.5. Кристаллические решетки алмаза, графита и углеродная нанотрубка

В кристаллической решетке алмаза взаимодействие между атомами очень сильное по всем направлениям, поэтому он является самым твердым веществом на планете и существует в форме кристаллов.

В кристаллической решетке графита атомы размещены слоями, между которыми взаимодействие слабое, поэтому он легко крошится и используется в грифелях карандашей.

Углеродная нанотрубка представляет собой свернутую в цилиндр плоскость кристаллической решетки графита. Нанотрубки очень прочные на разрыв (хотя имеют толщину стенки в один атом углерода). Сделанная из нанотрубок нить толщиной с человеческий волос способна удерживать груз в сотни килограммов. Электрические свойства нанотрубок могут меняться, что делает их одним из основных материалов наноэлектроники.

Контрольные вопросы

1. Приведите примеры систем в окружающем мире.
2. Объясните, образуют ли систему устройства, из которых состоит компьютер: до сборки; после сборки; после включения компьютера.
3. От чего зависят свойства системы? Приведите примеры систем, состоящих из одних и тех же элементов, но обладающих различными свойствами.

2.2. Моделирование, формализация, визуализация

2.2.1. Моделирование как метод познания

Моделирование. Человечество в своей деятельности (научной, образовательной, технологической, художественной и др.) постоянно создает и использует модели объектов окружающего мира. Строгие правила построения моделей сформулировать невозможно, однако человечество накопило богатый опыт моделирования различных объектов и процессов.

Модели играют чрезвычайно важную роль в проектировании и создании различных технических устройств, машин и механизмов, зданий, электрических цепей и т. д. Без предварительного создания чертежа невозможно изготовить даже простую деталь, не говоря уже о сложном механизме. В процессе проектирования зданий и сооружений кроме чертежей часто изготавливают их макеты. Разработка электрической схемы обязательно предшествует созданию электрических цепей (рис. 2.6) и т. д.

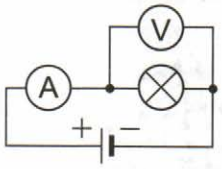


Рис. 2.6. Электрическая схема

Развитие науки невозможно без создания теоретических моделей (теорий, законов, гипотез и т. д.), отражающих строение, свойства и поведение реальных объектов. Создание новых теоретических моделей иногда коренным образом меняет представление человечества об окружающем мире (например, такую роль сыграла гелиоцентрическая система мира Коперника). Истинность теоретических моделей, т. е. их соответствие законам реального мира, проверяется с помощью опытов и экспериментов.

Всё художественное творчество фактически является процессом создания моделей. Например, такой литературный жанр, как басня, переносит реальные отношения между людьми на отношения между животными и фактически создает модели человеческих отношений. Более того, практически любое литературное произведение может рассматриваться как модель реальной человеческой жиз-

ни. Моделями, в художественной форме отражающими реальную действительность, являются также живописные полотна, скульптуры, театральные постановки и т. д.

Моделирование — это метод познания, состоящий в создании и исследовании моделей.

Модель. Каждый объект имеет большое количество различных свойств. В процессе построения модели выделяются главные, **наиболее существенные** для проводимого исследования свойства. В процессе исследования аэродинамических качеств модели самолета в аэродинамической трубе важно, чтобы модель имела геометрическое подобие оригиналу, но не важен, например, ее цвет. При построении электрических схем — моделей электрических цепей необходимо учитывать порядок подключения элементов цепи друг к другу, но не важно их геометрическое расположение друг относительно друга и т. д.

Модель создается человеком в процессе познания окружающего мира и отражает **существенные** с точки зрения цели проводимого исследования (цели моделирования) **свойства** изучаемого объекта, явления или процесса.

Разные науки исследуют объекты под разными углами зрения и строят различные типы моделей. В физике изучаются процессы взаимодействия и изменения объектов, в химии — химический состав объектов, в биологии — строение и поведение живых организмов и т. д.

Рассмотрим в качестве примера человека, в разных науках он исследуется в рамках различных моделей. В рамках механики его можно рассматривать как материальную точку, в химии — как объект, состоящий из различных химических веществ, в биологии — как систему, стремящуюся к самосохранению и т. д.

География, военное дело, судоходство и другие области человеческой деятельности невозможны без информационных моделей поверхности Земли в виде карт. Различные типы географических карт (политические, физические и т. д.) представляют собой информационные модели, отражающие различные особенности земной поверхности, т. е. один и тот же объект отражают несколько моделей.

Для описания и исследования **одного и того же объекта** могут использоваться **несколько моделей.**

Вместе с тем разные объекты могут описываться одной моделью. Например, для описания движения планет, движения автомобиля или движения мяча в определенных условиях (размеры объекта гораздо меньше его перемещений) можно использовать одну и ту же модель движения материальной точки.

Для описания и исследования **разных объектов** может использоваться **одна и та же модель.**

Никакая модель не может заменить сам объект. Но при решении конкретной задачи, когда нас интересуют определенные свойства изучаемого объекта, модель оказывается полезным, а подчас и единственным инструментом исследования.

Контрольные вопросы

1. Подготовьте сообщение об использовании моделирования в различных областях деятельности.
2. Может ли объект иметь несколько моделей? Приведите примеры.
3. Могут ли разные объекты описываться одной и той же моделью? Если да, то приведите примеры.

2.2.2. Материальные и информационные модели

Все модели можно разбить на два больших класса: **материальные** модели и **информационные** модели.

Материальные модели. Материальные модели позволяют представить в материальной наглядной форме объекты, недоступные для непосредственного исследования (очень большие или очень маленькие объекты, очень быстрые или очень медленные процессы и др.). Например, макеты зданий и сооружений позволяют архитекторам выбрать наилучшие градостроительные решения, модели самолетов и кораблей позволяют инженерам выбрать оптимальную форму этих транспортных средств.

Материальные модели часто используются в процессе обучения. В курсе географии первые представления о нашей планете Земля мы получаем, изучая ее модель — глобус (рис. 2.7), в курсе физики изучаем работу двигателя внутреннего сгорания по его модели, в химии при изучении строения вещества используем модели молекул и кристаллических решеток, в биологии изучаем строение человека по анатомическим муляжам.

Рис. 2.7. Материальная модель — глобус Земли

Информационные модели. Информационные модели представляют объекты и процессы в **образной** или **знаковой** форме, а также в форме **таблиц, блок-схем, графов** и т. д.

Образные модели (рисунки, фотографии и др.) представляют собой зрительные образы объектов, зафиксированные на каком-либо носителе информации (бумаге, фото- и кинопленке и др.). Широко используются образные информационные модели в обучении, где требуется классификация объектов по их внешним признакам (вспомните учебные плакаты по ботанике, биологии и физике).

Знаковые информационные модели строятся с использованием различных языков (знаковых систем). Знаковая информационная модель может быть представлена в форме текста (например, программы на языке программирования) или формулы (например, второго закона Ньютона $F = ma$).

Широко распространены информационные модели в форме таблиц. В периодической системе химических элементов Д. И. Менделеева (рис. 2.8) химические элементы располагаются в ячейках таблицы по возрастанию атомных весов, а в столбцах — по количеству валентных электронов. Важно, что по положению в таблице можно определить некоторые физические и химические свойства элементов.

ПЕРИОДЫ	РЯДЫ	I	ПЕРИОДИЧЕСКАЯ СИСТЕМА ЭЛЕМЕНТОВ Д.И. МЕНДЕЛЕЕВА					VII	VIII		
1	I	(H)	II	III	IV	V	VI	ВОДОРОД 1 H 1,0079	ГЕЛИЙ 2 He 4,00260		
2	II	3 Li ЛИТИЙ 6,941	4 Be БЕРИЛЛИЙ 9,01218	БОР 5 B 10,81	УГЛЕРОД 6 C 12,011	АЗОТ 7 N 14,0067	КИСЛОРОД 8 O 15,9994	ФТОР 9 F 18,998403	НЕОН 10 Ne 20,179		
3	III	11 Na НАТРИЙ 22,98977	12 Mg МАГНИЙ 24,305	13 Al АЛЮМИНИЙ 26,98154	14 Si КРЕМНИЙ 28,086	15 P ФОСФОР 30,97376	СЕРА 16 S 32,06	ХЛОР 17 Cl 35,453	АРГОН 18 Ar 39,948		
4	IV	19 K КАЛИЙ 39,0983	20 Ca КАЛЬЦИЙ 40,08	21 Sc СКАНДИЙ 44,9559	22 Ti ТИТАН 47,90	23 V ВАНАДИЙ 50,9415	24 Cr ХРОМ 51,996	25 Mn МАРГАНЕЦ 54,9380	26 Fe ЖЕЛЕЗО 55,847	27 Co КОБАЛЬТ 58,9332	28 Ni НИКЕЛЬ 58,71
	V	29 Cu МЕДЬ 63,546	30 Zn ЦИНК 65,38	31 Ga ГАЛЛИЙ 69,735	32 Ge ГЕРМАНИЙ 72,59	МЫШЬЯК 33 As 74,9216	СЕЛЕН 34 Se 78,96	БРОМ 35 Br 79,904	КРИПТОН 36 Kr 83,80		
5	VI	37 Rb 85,467 РУБИДИЙ	38 Sr 87,62 СТРОНЦИЙ	39 Y 88,9059 ИТТРИЙ	40 Zr 91,22 ЦИРКОНИЙ	41 Nb 92,906 НИОБИЙ	42 Mo 95,94 МОЛИБДЕН	43 Tc 98,906 ТЕХНЕЦИЙ	44 Ru 101,07 РУТЕНИЙ	45 Rh 102,9055 РОДИЙ	46 Pd 106,4 ПАЛЛАДИЙ
	VII	47 Ag СЕРЕБРО 107,868	48 Cd КАДМИЙ 112,41	49 In ИНДИЙ 114,82	50 Sn ОЛОВО 118,69	51 Sb СУРЬМА 121,75	52 Te ТЕЛЛУР 127,60	ИОД 53 I 126,90451	54 Xe КСЕНОН 131,30		
6	VIII	55 Cs ЦЕЗИЙ 132,905	56 Ba БАРИЙ 137,33	57 La ЛАНТАН 138,9055	72 Hf ГАФНИЙ 178,49	73 Ta ТАНТАЛ 180,947	74 W ВОЛЬФРАМ 183,85	75 Re РЕНИЙ 186,207	76 Os ОСМИЙ 190,2	77 Ir ИРИДИЙ 192,22	78 Pt ПЛАТИНА 195,09
	IX	79 Au ЗОЛОТО 196,9665	80 Hg РТУТЬ 200,59	81 Tl ТАЛЛИЙ 204,37	82 Pb СВИНЕЦ 207,2	83 Bi ВИСМУТ 208,9804	84 Po ПОЛОНИЙ [209]	85 At АСТАТ [210]	86 Rn РАДОН [222]		
7	X	87 Fr [223] ФРАНЦИЙ	88 Ra 226,0254 РАДИЙ	89 Ac [227] АКТИНИЙ	104 Ku [260] КУРЧАТОВИЙ	105 [260]	106 [263]				

Рис. 2.8. Информационная модель — таблица элементов Менделеева

При построении информационных моделей некоторых типов одновременно используются система графических элементов и знаковая система. Так, в блок-схемах алгоритмов используются различ-

ные геометрические фигуры для обозначения элементов алгоритма и формальный язык для записи команд программы (рис. 2.9).

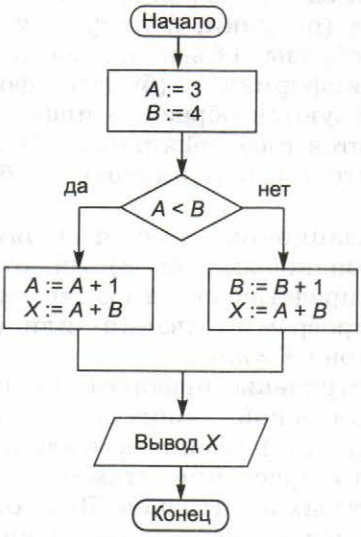


Рис. 2.9. Информационная модель — блок-схема алгоритма

Важную роль играют информационные модели, которые отображают **иерархические системы.** В биологии весь животный мир рассматривается как иерархическая система (тип, класс, отряд, семейство, род, вид), в информатике используется иерархическая файловая система и т. д.

В иерархической информационной модели объекты распределяются по уровням, от первого (верхнего) уровня до нижнего (последнего) уровня. На первом уровне может располагаться только один элемент, который является вершиной иерархической структуры. Основное отношение между уровнями состоит в том, что элемент более высокого уровня может состоять из нескольких элементов нижнего уровня, при этом каждый элемент нижнего уровня может входить в состав только одного элемента верхнего уровня.

Удобным способом наглядного представления иерархических информационных моделей являются **графы.** Элементы иерархической модели отображаются в графе овалами (**вершинами графа**).

Элементы верхнего уровня находятся в отношении «состоять из» к элементам более низкого уровня. Такая связь между элементами отображается в форме **дуги графа** (направленной линии в форме стрелки).

Графы, имеющие одну вершину верхнего уровня, напоминают деревья, которые растут сверху вниз, поэтому называются **деревьями. Ветви дерева** могут связывать объекты только соседних иерархических уровней, причем каждый объект нижнего уровня может быть связан ветвью только с одним объектом верхнего уровня.

Для описания исторического процесса смены поколений семьи используются информационные модели в форме **генеалогического дерева.** В качестве примера можно рассмотреть фрагмент генеалогического дерева династии Рюриковичей (рис. 2.10).

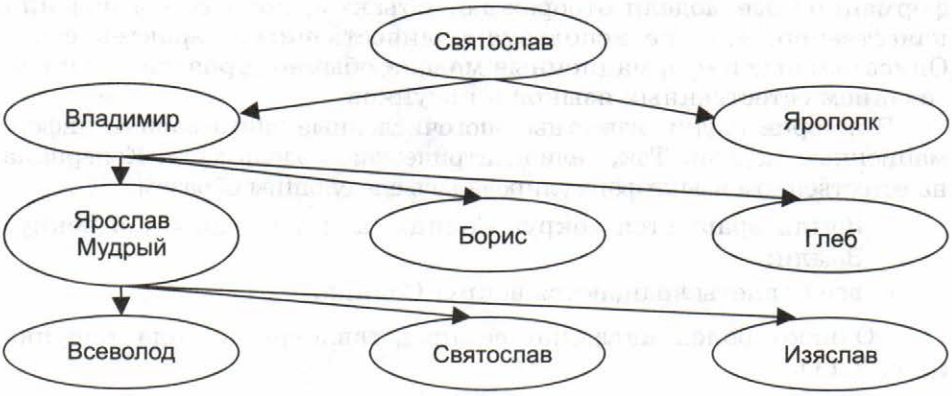


Рис. 2.10. Информационная модель — генеалогическое дерево Рюриковичей (X–XI века)

Контрольные вопросы

1. Приведите примеры материальных моделей.
2. Приведите примеры информационных моделей различных видов.

Задания для самостоятельного выполнения

2.1. *Задание с развернутым ответом.* Постройте фрагмент иерархической модели животного мира.

2.2. *Задание с развернутым ответом.* Постройте фрагмент модели генеалогического дерева вашей семьи.

2.2.3. Формализация и визуализация информационных моделей

На протяжении своей истории человечество использовало различные способы и инструменты для создания информационных моделей. Эти способы постоянно совершенствовались. Так, первые информационные модели создавались в форме наскальных рисунков. В настоящее время информационные модели обычно строятся и исследуются с использованием современных компьютерных технологий.

Описательные информационные модели. Описательные информационные модели отображают объекты, процессы и явления качественно, т. е. не используя количественных характеристик. Описательные информационные модели обычно строятся с использованием естественных языков и рисунков.

В истории науки известны многочисленные описательные информационные модели. Так, гелиоцентрическая модель мира Коперника на естественном языке формулировалась следующим образом:

- Земля вращается вокруг Солнца, а Луна вращается вокруг Земли;
- все планеты вращаются вокруг Солнца.

Однако более наглядно ее представление в виде рисунка (рис. 2.11).

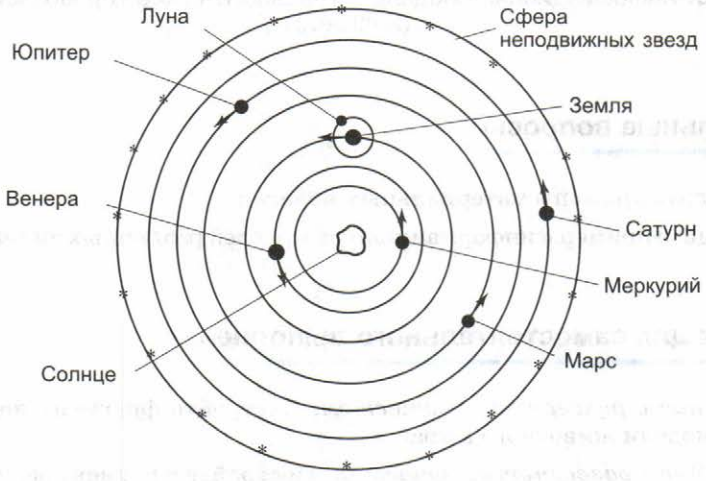


Рис. 2.11. Описательная модель гелиоцентрической системы мира Коперника

В физике явление электростатического взаимодействия двух зарядов описывается на естественном языке так: «Два одноименных заряда отталкиваются, а два разноименных притягиваются».

Для наглядности можно нарисовать линии напряженности электростатического поля и эквипотенциальные поверхности (рис. 2.12).

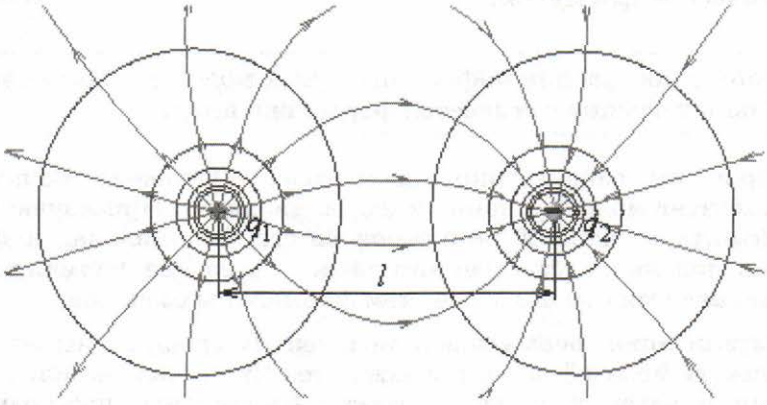


Рис. 2.12. Описательная модель взаимодействия электрических зарядов

В химии строение молекулы воды можно качественно описать на естественном языке: «Молекула воды состоит из атома кислорода и двух атомов водорода».

Для наглядности строение молекулы можно нарисовать (рис. 2.13).

Рис. 2.13. Описательная модель молекулы воды

Формализация информационных моделей. С помощью формальных языков строятся формальные информационные модели. Математика является широко используемым формальным языком. С использованием математических понятий и формул строятся **математические модели**. Математика включает различные формальные языки, с некоторыми из которых (алгебра и геометрия) вы знакомитесь в школе.

В естественных науках (физике, химии и др.) строятся формальные модели явлений и процессов. В большинстве случаев для этого применяется универсальный математический язык алгебраических формул. Однако в некоторых случаях используются специализированные формальные языки (в химии — язык химических формул, в музыке — нотная грамота и т. д.).

Ньютон формализовал гелиоцентрическую систему мира, открыв закон всемирного тяготения и законы механики и записав их в виде формул.

$$F = \gamma \cdot \frac{m_1 \cdot m_2}{r^2}$$

$$\vec{F} = m \cdot \vec{a}$$

В электростатике взаимодействие электрических зарядов описывается формулой закона Кулона.

$$F = k \cdot \frac{q_1 \cdot q_2}{r^2}$$

В химии строение молекулы воды описывается химической формулой.

$$H_2O$$

Процесс построения информационных моделей с помощью формальных языков называется **формализацией**.

В процессе познания окружающего мира человечество постоянно использует моделирование и формализацию. При изучении нового объекта сначала обычно строится его описательная информационная модель на естественном языке, затем она формализуется, т. е. выражается с использованием формальных языков.

Визуализация формальных моделей. В процессе исследования формальных моделей часто производится их визуализация. Для визуализации алгоритмов используются блок-схемы, пространственных соотношений между объектами — чертежи, моделей электрических цепей — электрические схемы. При визуализации формальных моделей с помощью анимации может отображаться динамика процесса, производиться построение графиков изменения величин и т. д.

В настоящее время широкое распространение получили **компьютерные интерактивные визуальные модели**. В таких моделях исследователь может менять начальные условия и параметры протекания процессов и наблюдать изменения в поведении модели.

В качестве примера визуализации формальной модели можно привести компьютерную визуальную интерактивную модель гидравлической машины (рис. 2.14).

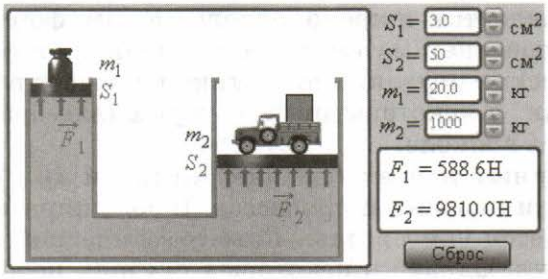


Рис. 2.14. Компьютерная визуальная интерактивная модель гидравлической машины

В компьютерном эксперименте можно изменять площади поршней S_1 и S_2 и массы грузов m_1 и m_2 и вывести формулу соотношения между площадями поршней и действующими на них силами.

Контрольные вопросы

1. Приведите примеры описательных информационных моделей.
2. Приведите примеры формализованных информационных моделей.

Задания для самостоятельного выполнения

2.3. *Практическое задание.* Ознакомьтесь с визуальными интерактивными моделями из различных предметных областей в Интернете по адресу http://www.college.ru

2.3. Основные этапы разработки и исследования моделей на компьютере

Использование компьютера для исследования информационных моделей различных объектов и систем позволяет изучить их изменения в зависимости от значений тех или иных свойств. Процесс разработки моделей и их исследования на компьютере можно разделить на несколько основных этапов.

Описательная информационная модель. На первом этапе исследования объекта или процесса обычно строится описательная информационная модель. Такая модель выделяет существенные, с точки зрения целей проводимого исследования, свойства объекта, а несущественными свойствами пренебрегает.

Формализованная модель. На втором этапе создается формализованная модель, т. е. описательная информационная модель записывается с помощью какого-либо формального языка. В такой модели с помощью формул, уравнений или неравенств фиксируются формальные соотношения между начальными и конечными значениями свойств объектов, а также накладываются ограничения на допустимые значения этих свойств.

Однако далеко не всегда удается найти формулы, явно выражающие искомые величины через исходные данные. В таких случаях используются приближенные математические методы, позволяющие получать результаты с заданной точностью.

Компьютерная модель. На третьем этапе необходимо формализованную информационную модель преобразовать в компьютерную модель, т. е. выразить ее на понятном для компьютера языке. Существуют различные пути построения компьютерных моделей, в том числе:

- создание компьютерной модели в форме проекта на одном из языков программирования;
- построение компьютерной модели с использованием электронных таблиц или других приложений: систем компьютерного черчения, систем управления базами данных, геоинформационных систем и т. д.

В процессе создания компьютерной модели полезно разработать удобный графический интерфейс, который позволит визуализировать формальную модель, а также реализовать интерактивный диалог человека с компьютером на этапе исследования модели.

Компьютерный эксперимент. Четвертый этап исследования информационной модели состоит в проведении компьютерного эксперимента. Если компьютерная модель существует в виде проекта на одном из языков программирования, ее нужно запустить на выполнение, ввести исходные данные и получить результаты.

Если компьютерная модель исследуется в приложении, например в электронных таблицах, то можно построить диаграмму или график, провести сортировку и поиск данных или использовать другие специализированные методы обработки данных.

При использовании готовой компьютерной визуальной интерактивной модели необходимо ввести исходные данные, запустить модель на выполнение и наблюдать изменение объекта и характеризующих его величин.

В виртуальных компьютерных лабораториях можно проводить эксперименты с реальными объектами. Для этого к компьютеру присоединяются датчики измерения физических параметров (температуры, давления, силы и др.), данные измерений передаются в компьютер и обрабатываются специальной программой. Результаты эксперимента в виде таблиц, графиков и диаграмм отображаются на экране монитора и могут быть распечатаны.

Анализ полученных результатов и корректировка исследуемой модели. Пятый этап состоит в анализе полученных результатов и корректировке исследуемой модели. В случае расхождения результатов, полученных при исследовании информационной модели, с измеряемыми параметрами реальных объектов можно сделать вывод, что на предыдущих этапах построения модели были допущены ошибки или неточности.

Например, при построении описательной качественной модели могут быть неправильно отобраны существенные свойства объектов, в процессе формализации могут быть допущены ошибки в формулах и т. д. В этих случаях необходимо провести корректировку модели, причем уточнение модели может проводиться многократно, пока анализ результатов не покажет их соответствие изучаемому объекту.

Контрольные вопросы

1. Перечислите и опишите основные этапы разработки и исследования моделей на компьютере.
2. Какие программные средства обычно используются для создания компьютерных моделей?

2.4. Построение и исследование физических моделей

Рассмотрим процесс построения и исследования модели на конкретном примере движения тела, брошенного под углом к горизонту.

Содержательная постановка задачи «Бросание мячика в площадку». В процессе тренировок теннисистов используются автоматы по бросанию мячика в определенное место площадки. Требуется задать автомату необходимую скорость и угол бросания мячика для попадания в площадку определенной длины, находящуюся на известном расстоянии (рис. 2.15).

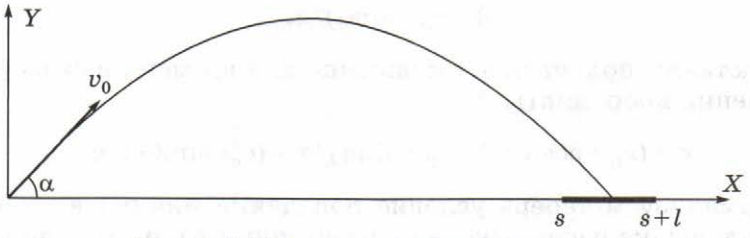


Рис. 2.15. Бросание мячика в площадку

Качественная описательная модель. Сначала построим качественную описательную модель процесса движения тела с использованием физических объектов, понятий и законов, т. е. в данном случае идеализированную модель движения объекта. Из условия

задачи можно сформулировать следующие основные предположения:

- мячик мал по сравнению с Землей, поэтому его можно считать материальной точкой;
- изменение высоты мячика мало, поэтому ускорение свободного падения можно считать постоянной величиной $g = 9{,}8$ м/с^2 и движение по оси Y можно считать равноускоренным;
- скорость бросания тела мала, поэтому сопротивлением воздуха можно пренебречь и движение по оси X можно считать равномерным.

Формальная модель. Для формализации модели используем известные из курса физики формулы равномерного и равноускоренного движения. При заданных начальной скорости v_0 и угле бросания α значения координат дальности полета x и высоты y от времени можно описать следующими формулами:

$$\begin{aligned} x &= v_0 \cdot \cos\alpha \cdot t, \\ y &= v_0 \cdot \sin\alpha \cdot t - g \cdot t^2/2. \end{aligned} \qquad (2.1)$$

Площадка расположена на поверхности Земли, поэтому из второй формулы (2.1) можно выразить время, которое понадобится мячику, чтобы достичь площадки:

$$\begin{aligned} v_0 \cdot \sin\alpha \cdot t - g \cdot t^2/2 &= 0, \\ t \cdot (v_0 \cdot \sin\alpha - g \cdot t/2) &= 0. \end{aligned}$$

Значение времени $t = 0$ не имеет физического смысла, поэтому:

$$\begin{aligned} v_0 \cdot \sin\alpha - g \cdot t/2 &= 0, \\ t &= (2 \cdot v_0 \cdot \sin\alpha)/g. \end{aligned}$$

Подставим полученное выражение для времени в формулу для вычисления координаты x:

$$x = (v_0 \cdot \cos\alpha \cdot 2 \cdot v_0 \cdot \sin\alpha)/g = (v_0^2 \cdot \sin 2\alpha)/g. \qquad (2.2)$$

Формализуем теперь условие попадание мячика в площадку. Пусть площадка расположена на расстоянии s и имеет длину l (см. рис. 2.16). Тогда попадание произойдет, если значение координаты x мячика будет удовлетворять условию в форме неравенства:

$$s \le x \le s + l.$$

Если $x < s$, то это означает «недолет», а если $x > s + l$, то это означает «перелет».

Компьютерная модель движения тела. На основе формальной модели, описывающей движение тела, брошенного под углом к горизонту, можно создать компьютерную модель с использованием системы программирования, например системы объектно-ориентированного программирования Visual Basic, так как она позволяет визуализировать траектории движения тела.

На основе данной формальной модели можно создать компьютерную модель также с использованием электронных таблиц Microsoft Excel или OpenOffice.org Calc. Траектория движения тела визуализируется с использованием диаграммы типа *График*.

Контрольные вопросы

1. Какие основные предположения можно сделать при построении качественной описательной модели бросания мячика под углом к горизонту?
2. Чем отличается компьютерная модель от формальной модели?

2.5. Приближенное решение уравнений

На языке алгебры формальные модели записываются с помощью уравнений, точное решение которых основывается на поиске равносильных преобразований алгебраических выражений, позволяющих выразить переменную величину с помощью формулы.

Точные решения существуют только для некоторых уравнений определенного вида (линейные, квадратные, тригонометрические и др.), поэтому для большинства уравнений приходится использовать методы приближенного решения с заданной точностью (графические или численные).

Например, нельзя найти корень уравнения $x^3 - \sin x = 0$ путем равносильных алгебраических преобразований. Однако такие уравнения можно решать приближенно графическими и численными методами.

Построение графиков функций может использоваться для грубо приближенного решения уравнений. Для уравнений вида $f(x) = 0$, где $f(x)$ — некоторая непрерывная функция, корень (или корни) этого уравнения является точкой (или точками) пересечения графика функции с осью X.

Графическое решение таких уравнений можно осуществить путем построения компьютерных моделей:

- построением графика функции в системе объектно-ориентированного программирования Visual Basic;
- в электронных таблицах Microsoft Excel или OpenOffice.org Calc путем построения диаграммы типа *График*.

Контрольные вопросы

1. В каких случаях используются приближенные (графические) методы решения уравнений?

2.6. Компьютерное конструирование с использованием системы компьютерного черчения

Одним из средств компьютерного конструирования являются системы компьютерного черчения, которые представляют собой векторные графические редакторы, предназначенные для создания чертежей. При классическом черчении с помощью карандаша, линейки и циркуля производится построение элементов чертежа (отрезков, окружностей и прямоугольников) с точностью, которую предоставляют чертежные инструменты. Использование систем компьютерного черчения позволяет создавать чертежи с гораздо большей точностью. Кроме того, системы компьютерного черчения позволяют измерять расстояния, углы, периметры и площади начерченных объектов.

Пространственные соотношения между реальными объектами (положение и ориентация объектов в пространстве и их размеры) изучаются в курсе геометрии. Важное место в школьном курсе геометрии занимают геометрические построения с использованием линейки и циркуля. Для создания геометрических моделей на компьютере удобно использовать системы компьютерного черчения.

Системы компьютерного черчения могут использоваться в школьном курсе технологии, так как позволяют создавать чертежи деталей, в том числе трехмерных. Такие системы позволяют грамотно оформить чертеж: обозначить на чертеже размеры деталей и сделать надписи в соответствии с существующими стандартами.

Системы компьютерного черчения используются в качестве инструмента автоматического проектирования на производстве, так как обеспечивают возможность реализации сквозной технологии проектирования и изготовления деталей. На основе компьютерных чертежей генерируются управляющие программы для станков с числовым программным управлением (ЧПУ), в результате по компьютерным чертежам могут изготавливаться высоко-

точные детали из металла, пластмассы, дерева и других материалов.

Система компьютерного черчения КОМПАС специально предназначена для обучения компьютерному черчению в школах. КОМПАС можно использовать для выполнения геометрических построений с помощью циркуля и линейки, а также при создании чертежей деталей.

Система компьютерного черчения КОМПАС использует оригинальный формат файлов с расширением FRW, который распознается только самой создающей программой.

Контрольные вопросы

1. Какие вы знаете форматы файлов векторных графических редакторов? Подготовьте сообщение.

2.7. Экспертные системы распознавания химических веществ

Экспертные системы. Профессиональные экспертные системы достаточно широко используются в различных областях науки и техники. Такие системы позволяют автоматически выявлять причины сбоев в работе сложных технических систем (например, космических кораблей), распознавать личность человека по его отпечаткам пальцев или радужной оболочке глаза и т. д.

Основная задача экспертных систем — распознавать объекты или состояния объекта. В процессе обучения встречается достаточно много учебных ситуаций, когда вам приходится выступать в роли эксперта и необходимо распознать тот или иной объект. Обычно такие задачи выполняются методом проб и ошибок, без осознания и фиксации стратегии поиска.

Создание учебной экспертной системы позволяет осознать и зафиксировать последовательность рассуждений или действий, которая приводит к распознаванию того или иного объекта среди некоторой совокупности.

Лабораторная работа по неорганической химии «Распознавание химических удобрений». Даны удобрения, химические реактивы и справочная таблица по взаимодействию удобрений с некоторыми реактивами (табл. 2.1) и предлагается распознать каждое из удобрений.

Таблица 2.1. **Свойства удобрений**

№	Внешний вид	Взаимодействие раствора удобрения с			Удобрение (результат распознавания)
		H_2SO_4	$BaCl_2$	раствором щелочи	
1	Мелкокристаллическая масса или гранулы белого цвета	Выделяется бурый газ	—	Запах аммиака	Аммиачная селитра
2	Крупные белые кристаллы	Выделяется бурый газ	—	—	Натриевая селитра
3	Мелкие белые кристаллы	—	Белый осадок	Запах аммиака	Сульфат аммония
4	Светло-серый порошок или гранулы	—	Белый осадок	—	Суперфосфат
5	Розовые кристаллы (от примеси)	—	—	—	Сильвинит
6	Белые кристаллы	—	—	—	Калийная соль

Формальная модель экспертной системы «Распознавание удобрений». Экспертная система может быть представлена в виде алгоритма, состоящего из последовательности шагов с использованием алгоритмической структуры «ветвление». Можно построить различные алгоритмы поиска, однако необходимо стремиться к выбору оптимальной стратегии распознавания (достижения цели за минимальное число шагов). Такая стратегия будет реализована, если каждый шаг будет максимально уменьшать неопределенность (нести максимальное количество информации).

Построим алгоритм (рис. 2.16), в котором на первом шаге разделим шесть веществ на две группы по условию «*При взаимодействии с H_2SO_4 выделяется бурый газ*». Если условие:

- выполняется, то это вещества первой группы под номерами 1 и 2;
- не выполняется, то это вещества второй группы под номерами 3, 4, 5 и 6.

Для идентификации веществ первой группы достаточно проверить справедливость условия «*При взаимодействии с раствором щелочи ощущается запах аммиака*». Если условие:

- выполняется, то это вещество «1. Аммиачная селитра»;
- не выполняется, то это вещество «2. Натриевая селитра».

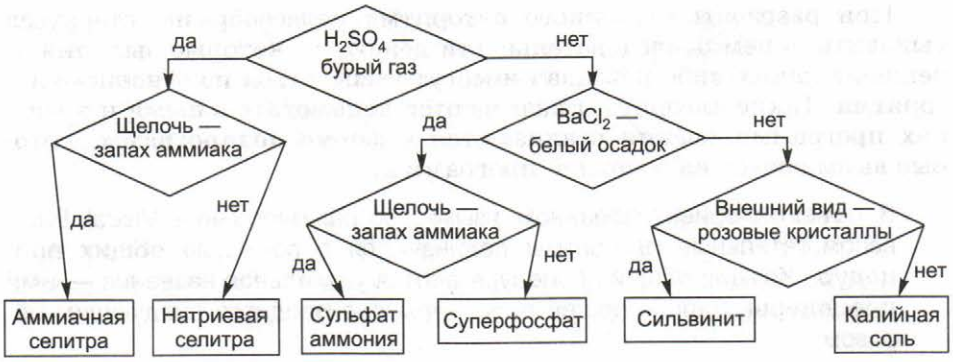


Рис. 2.16. Блок-схема учебной экспертной системы «Распознавание удобрений»

Для идентификации веществ второй группы сначала необходимо проверить справедливость условия «*При взаимодействии с $BaCl_2$ выпадает белый осадок*». Если условие:

- выполняется, то это вещества 3 и 4;
- не выполняется, то это вещества 5 и 6.

Для идентификации веществ 3 и 4 достаточно проверить справедливость условия «*При взаимодействии с раствором щелочи ощущается запах аммиака*». Если условие:

- выполняется, то это вещество «3. Сульфат аммония»;
- не выполняется, то это вещество «4. Суперфосфат».

Для идентификации веществ 5 и 6 достаточно проверить справедливость условия «*Внешний вид — розовые кристаллы*». Если условие:

- выполняется, то это вещество «5. Сильвинит»;
- не выполняется, то это вещество «6. Калийная соль».

Целесообразно представить иерархическую модель экспертной системы в виде блок-схемы (см. рис. 2.17).

Компьютерная модель экспертной системы на языке Visual Basic. Создать экспертную систему распознавания удобрений можно с использованием языка Visual Basic. Экспертная система будет задавать пользователю серии вопросов о результатах взаимодействия вещества с кислотой, щелочью и солью или о внешнем виде удобрений. Пользователь будет отвечать «*Да*» или «*Нет*» (на основании опытов или теоретических знаний). В результате нескольких серий вопросов будут определены названия всех удобрений.

При разработке сложного алгоритма целесообразно стараться выделить в нем последовательности действий, которые выполняют решение каких-либо подзадач и могут вызываться из основного алгоритма. Такие алгоритмы называются **вспомогательными** и в языках программирования реализуются в форме **подпрограмм**, которые вызываются из основной программы.

В объектно-ориентированном языке программирования Visual Basic вспомогательные алгоритмы реализуются с помощью **общих процедур**. Каждой общей процедуре дается уникальное название — **имя процедуры**. Запись общей процедуры производится следующим образом:

```
Sub ИмяПроцедуры(...)
программный код
End Sub
```

Запуск общих процедур не связывается с какими-либо событиями, а реализуется путем вызова по имени из других процедур.

Контрольные вопросы

1. Является ли единственным приведенный алгоритм учебной экспертной системы распознания удобрений? Какие еще варианты алгоритма экспертной системы вы можете предложить?

2.8. Информационные модели управления объектами

В процессе функционирования сложных систем (биологических, технических и т. д.) важную роль играют информационные процессы управления.

Для поддержания своей жизнедеятельности любой живой организм постоянно получает информацию из внешнего мира с помощью органов чувств, обрабатывает ее и управляет своим поведением (например, перемещаясь в пространстве, избегает опасности).

В процессе управления полетом самолета в режиме автопилота бортовой компьютер получает информацию от датчиков (скорости, высоты и т. д.), обрабатывает ее и передает команды на исполнительные механизмы, изменяющие режим полета (закрылки, клапаны, регулирующие работу двигателей, и т. д.).

В любом процессе управления всегда происходит взаимодействие двух объектов — **управляющего** и **управляемого**, которые соединены каналами **прямой** и **обратной связи**. По каналу прямой связи передаются управляющие сигналы, а по каналу обратной связи — информация о состоянии управляемого объекта.

Системы управления без обратной связи. В системах управления без обратной связи не учитывается состояние управляемого объекта и обеспечивается управление только по прямому каналу (от управляющего объекта к управляемому объекту). Информационную модель системы управления без обратной связи можно наглядно представить с помощью схемы на рис. 2.17.

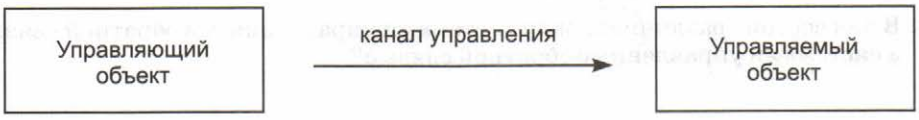


Рис. 2.17. Система управления без обратной связи

В качестве примера системы управления без обратной связи можно привести процесс записи информации на оптический диск.

Системы управления с обратной связью. В системах управления с обратной связью управляющий объект по прямому каналу управления производит необходимые действия над объектом управления, а по каналу обратной связи получает информацию о его реальных параметрах. Это позволяет осуществлять управление с гораздо большей точностью.

Информационную модель системы управления с обратной связью можно наглядно представить с помощью схемы на рис. 2.18.

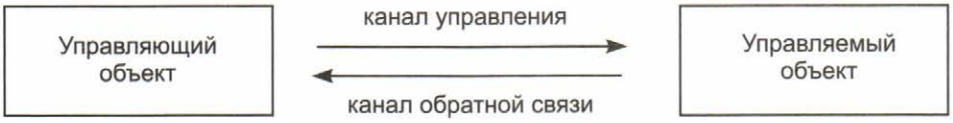


Рис. 2.18. Система управления с обратной связью

Примером системы управления с автоматической обратной связью является запись информации на жесткий диск. При записи информации на жесткий диск требуется особая точность установки магнитных головок, так как на рабочей поверхности пластин име-

ются десятки тысяч дорожек и необходимо учитывать их механические деформации (например, в результате изменения температуры). Контроллер жесткого диска (управляющий объект) по каналу обратной связи постоянно получает информацию о реальном положении магнитных головок (управляемый объект), а по каналу управления выставляет головки над поверхностью пластин с большой точностью.

Контрольные вопросы

1. Приведите примеры систем управления без обратной связи и с обратной связью. Подготовьте реферат.
2. В чем состоит различие между системами управления без обратной связи и системами управления с обратной связью?

Практические работы компьютерного практикума к главе 2 «Моделирование и формализация»

	Установить: • систему объектно-ориентированного программирования Visual Basic; • электронные таблицы OpenOffice.org Calc; • систему компьютерного черчения КОМПАС; • электронные таблицы Microsoft Excel	http://www.microsoft.com/visualstudio/ru-ru/products/2010-editions/express  http://ru.openoffice.org/  http://shkola.softline.ru/catalog/37  http://www.shkolaedu.ru/products/43 
	Установить: • электронные таблицы OpenOffice.org Calc	http://altlinux.org/Альт-Линукс-5.0.2-Школьный 

*Практическая работа 2.1

Разработка проекта «Бросание мячика в площадку»

Аппаратное и программное обеспечение. Компьютер с установленной операционной системой Windows или Linux.

Цель работы. Научиться создавать компьютерные модели движения в электронных таблицах и на языке объектно-ориентированного программирования Visual Basic.

Задание. Разработать проект, в котором визуализируется траектория движения тела, брошенного под углом к горизонту, и выясняется, попадет ли тело в площадку определенной длины, находящуюся на заданном расстоянии.

Задание. Проект «Бросание мячика в площадку» в электронных таблицах Microsoft Excel и OpenOffice.org Calc

1. В операционной системе Windows запустить электронные таблицы Microsoft Excel командой [*Пуск-Все программы-Microsoft Office-Microsoft Excel*] или электронные таблицы OpenOffice.org Calc командой [*Пуск-Все программы-OpenOffice-OpenOffice Calc*].

Или:

в операционной системе Linux запустить электронные таблицы OpenOffice.org Calc командой [*Пуск-Офис-OpenOffice Calc*].

Построение траектории движения мячика. Для ввода начальной скорости v_0 бросания мячика будем использовать ячейку B1, а для ввода угла бросания — ячейку B2.

Введем в ячейки A5:A18 значения времени t с интервалом 0,2 с, и вычислим по формулам (2.1) значения координат тела x и y для заданных значений времени.

2. Ввести:
   - в ячейку B5 формулу
     =$B$1*COS(РАДИАНЫ($B$2))*A5;
   - в ячейку C5 формулу
     =$B$1*SIN(РАДИАНЫ($B$2))*A5–4,9*A5*A5

В электронных таблицах аргументы функций COS() и SIN() задаются в радианах, поэтому необходимо преобразовать значения углов из градусов в радианы с помощью функции РАДИАНЫ().

3. Скопировать введенные формулы в ячейки B6:B18 и C6:C18 соответственно.

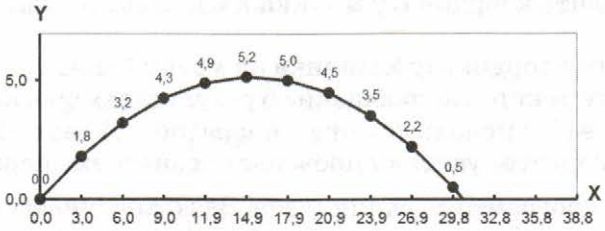
Получим в столбце B значения координаты мячика по оси X, а в столбце C — координаты по оси Y, вычисленные для определенных моментов времени.

	A	B	C
1	V0 =	18,0 м/с	
2	α =	34,0 град	
3			
4	t	*x = v0 ·cosα· t*	*y = v0 ·sinα· t – g·t2/2*
5	0,0	0,0	0,0
6	0,2	3,0	1,8
7	0,4	6,0	3,2
8	0,6	9,0	4,3
9	0,8	11,9	4,9
10	1,0	14,9	5,2
11	1,2	17,9	5,0
12	1,4	20,9	4,5
13	1,6	23,9	3,5
14	1,8	26,9	2,2
15	2,0	29,8	0,5
16	2,2	32,8	-1,6
17	2,4	35,8	-4,1
18	2,6	38,8	-7,0

Координаты мячика в заданные моменты времени

Визуализируем модель, построив график зависимости координаты y от координаты x (траекторию движения тела). Для построения траектории движения мячика используем диаграмму типа *График*.

4. При построении графика в качестве категорий использовать диапазон ячеек B5:B18, а в качестве значений — диапазон ячеек C5:C18.



Траектория движения мячика

5. По полученному графику (траектории движения мячика) можно качественно судить, попадет ли он в площадку при заданных начальных условиях (расстоянии до площадки и ее длины).

Задание. Проект «Бросание мячика в площадку» на языке объектно-ориентированного программирования Visual Basic

1. В операционной системе Windows запустить систему объектно-ориентированного программирования Visual Basic командой [*Пуск-Все программы-Visual Basic 2010 Express*].

Создадим сначала графический интерфейс проекта.

2. Разместить на форме:
   - четыре текстовых поля `TextBox` для ввода значений начальной скорости и угла бросания мячика, расстояния до площадки и ее длины;
   - две метки `Label` для вывода координаты `X` мячика в момент падения на поверхность и текстового сообщения о результатах броска;
   - десять меток `Label` для обозначения назначения текстовых полей (имен переменных и единиц измерения).

3. Создать программный код обработчика события, который определяет попадание мячика в площадку:
   - объявить вещественные константы одинарной точности `G` (ускорение свободного падения g) и `Pi` (число π);
   - объявить вещественные переменные одинарной точности `V0` (начальная скорость v_0), `A` (угол бросания α), `S` (расстояние до площадки s), `L` (длина площадки l), `X` и `Y` (координаты мячика), `T` (время);
   - присвоить переменным `V0`, `A`, `S`, `L` значения, введенные в текстовые поля, с использованием функции преобразования строки в вещественное число `Val()`;

- вычислить координату мячика X в момент падения на поверхность;
- вывести координату X мячика на метку Label1;
- вывести текстовое сообщение о результатах броска в поле метки Label2 с использованием оператора Select Case, в котором в качестве условия проверяется значение переменной X.

В языке программирования Visual Basic аргументы тригонометрических функций Sin(), Cos() и Tan() задаются в радианах, а угол бросания мячика мы будем вводить в градусах. Поэтому необходимо преобразовать значение угла из градусов в радианы с использованием константы Pi.

4. Поместить на форму кнопку Button1 и создать для нее обработчик событий Button1_Click():

```
Const G As Single = 9.81
Const Pi As Single = 3.14
Dim V0, A, S, L, X As Single
Private Sub Button1_Click(…)
'Ввод начальных значений
V0 = Val(TextBox1.Text)
A = Val(TextBox2.Text)
S = Val(TextBox3.Text)
L = Val(TextBox4.Text)
'Вычисление координаты мячика в момент падения на площадку
X = V0^2*Math.Sin(2*A*Pi/180)/G
Label1.Text = X
'Попадание в площадку
Select Case X
Case Is < S
  Label2.Text = "Недолет"
Case Is > S + L
  Label2.Text = "Перелет"
Case Else
  Label2.Text = "Попадание"
End Select
End Sub
```

Для визуализации формальной модели построим траекторию движения тела (график зависимости высоты мячика над поверхностью земли от дальности полета).

5. Поместить дополнительно на форму графическое поле PictureBox1. С помощью диалогового окна *Свойства* установить с использованием свойства Size размер поля, например 400;220.

В обработчике события осуществим преобразование компьютерной системы координат графического поля в математическую систему координат, удобную для построения траектории движения. Нарисуем оси координат и нанесем на них шкалы.

В математической системе координаты находятся в диапазонах $0 \le X \le 400$ и $-20 \le Y \le 200$. Траектория движения мячика, скорее всего, будет в диапазоне координат $0 \le X \le 40$ м и $0 \le Y \le 20$ м. Следовательно, необходимо увеличить масштаб графика в 10 раз:

- координаты точек графика необходимо умножить на 10;
- значения шкал осей разделить на 10.

Построение траектории осуществим в цикле со счетчиком (координата `X`) с использованием метода рисования точки `DrawEllipse(Pen1, X*10, Y*10, 1, 1)`, в котором координатами точки являются координаты мячика.

6. Поместить на форму кнопку `Button2` и создать для нее обработчик события.

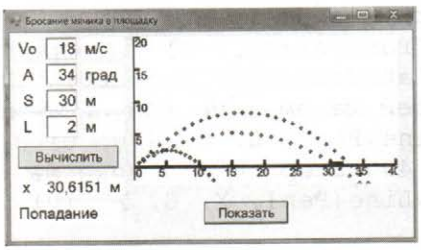
```
'Объявление переменных
Dim X, Y, T As Single
Dim Graph1 As Graphics
Dim Pen1 As New Pen(Color.Black, 4)
Dim drawBrush As New SolidBrush(Color.Black)
Dim drawFont As New Font("Arial", 10)
'Обработчик события
Private Sub Button2_Click(…)
Graph1 = Me.PictureBox1.CreateGraphics()
'Вывод шкал математической системы координат
'в компьютерной системе координат
For X = 0 To 400 Step 50
  Graph1.DrawString(X/10, drawFont, drawBrush, X-15, 200)
Next X
For Y = 0 To 200 Step 50
Graph1.DrawString(Y/10, drawFont, drawBrush, 0, 200-Y)
Next Y
'Преобразование компьютерной системы координат
в математическую систему координат
Graph1.ScaleTransform(1, -1) 'Поворот оси Y
Graph1.TranslateTransform(0, -200) 'Сдвиг по оси Y
'Рисование осей математической системы координат
Graph1.DrawLine(Pen1, 0, 0, 400, 0) 'Ось X
For X = 0 To 400 Step 50 'Засечки на оси X
  Graph1.DrawLine(Pen1, X, 0, X, 10)
Next X
Graph1.DrawLine(Pen1, 0, -20, 0, 200) 'Ось Y
```

```
For Y = 0 To 200 Step 50 'Засечки на оси Y
  Graph1.DrawLine(Pen1, 0, Y, 10, Y)
Next Y
'Площадка
Graph1.DrawLine(Pen1, S*10, 4, (S+L)*10, 4)
'Построение траектории движения мячика
For T = 0 To 10 Step 0.1
  Y = V0*Math.Sin(A*Pi/180)*T - G*T * T/2
  X = V0*Math.Cos(A*Pi/180)*T
  Graph1.DrawEllipse(Pen1, X*10, Y*10, 1, 1)
Next T
End Sub
```

Компьютерный эксперимент. Введем произвольные значения начальной скорости и угла бросания мячика; скорее всего, при этих значениях его попадания в площадку не будет. Меняя один из параметров, например угол, произведем пристрелку, используя известный артиллерийский прием «взятие в вилку», в котором применяется эффективный метод «деление пополам». Сначала найдем угол, при котором мячик перелетит площадку, затем угол, при котором мячик не долетит до нее. Вычислим среднее значение углов, составляющих «вилку», и проверим, попадет ли мячик в площадку. Если он попадет в площадку, то задача выполнена, если не попадет, то рассматривается новая «вилка» и т. д.

7. Запустить проект и ввести значения начальной скорости, угла, расстояния до площадки и ее длины.
   Щелкнуть по кнопкам *Вычислить* и *Показать*.
   На метки будут выведены результаты, а в графическом поле появится траектория движения тела.

8. Подобрать значения начальной скорости и угла бросания мячика, обеспечивающие его попадание в площадку.
   Например, при скорости бросания мячика $v_0 = 18\ м/с$ и угле бросания $\alpha = 34\ град$ мячик попадет в площадку длиной $L = 2\ м$, находящуюся на расстоянии $S = 30\ м$, на расстоянии $x = 30{,}61514\ м$.



Проект «Бросание мячика в площадку» на языке Visual Basic

Анализ результатов. Полученная точность расстояния попадания мячика в площадку $x = 30{,}61514$ *м* не имеет физического смысла и определяется типом переменной. Так как X является переменной одинарной точности, то ее значение вычисляется с точностью семи значащих цифр. Исходные данные заданы с точностью двух значащих цифр, поэтому целесообразно результат округлить до трех значащих цифр $x = 30{,}6$ *м*.

Практическая работа 2.2

Разработка проекта «Графическое решение уравнения»

Аппаратное и программное обеспечение. Компьютер с установленной операционной системой Windows или Linux.

Цель работы. Научиться создавать компьютерные модели графического решения уравнений в электронных таблицах и на языке объектно-ориентированного программирования Visual Basic.

Задание. Разработать проект, в котором приближенно графически решается уравнение $x^3 - \sin x = 0$.

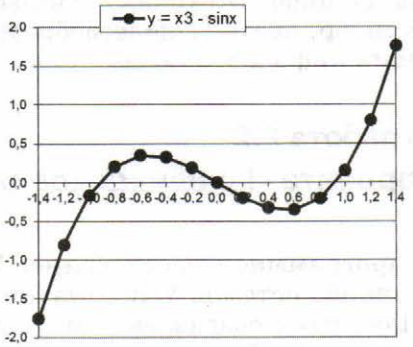
Задание. Проект «Графическое решение уравнения» в электронных таблицах Microsoft Excel и OpenOffice.org Calc

1. В операционной системе Windows запустить электронные таблицы Microsoft Excel командой [*Пуск-Все программы-Microsoft Office-Microsoft Excel*] или электронные таблицы OpenOffice.org Calc командой [*Пуск-Все программы-OpenOffice-OpenOffice. Calc*].
   Или:
   в операционной системе Linux запустить электронные таблицы OpenOffice.org Calc командой [*Пуск-Офис-OpenOffice Calc*].
   Для графического решения уравнения представим функцию $y = x^3 - \sin x$ в табличной форме.
2. В диапазон ячеек B1:P1 ввести значения аргумента функции от −1,4 до 1,4 с шагом 0,2.
3. В ячейку B2 ввести формулу вычисления значений функции =B1^3–SIN(B1) и скопировать ее в диапазон ячеек C2:P2.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	**x**	-1,4	-1,2	-1,0	-0,8	-0,6	-0,4	-0,2	0,0	0,2	0,4	0,6	0,8	1,0	1,2	1,4
2	**y = x3 - sinx**	-1,8	-0,8	-0,2	0,2	0,3	0,3	0,2	0,0	-0,2	-0,3	-0,3	-0,2	0,2	0,8	1,8

4. Для грубо приближенного определения корней уравнения построить диаграмму типа *График*.

График функции пересекает ось X три раза и, следовательно, уравнение имеет три корня.
По графику приближенно можно определить, что $x_1 \approx -0{,}9$, $x_2 \approx 0$ и $x_3 \approx 0{,}9$.



Решение уравнения путем построения диаграммы типа *График*

Задание. Проект «Графическое решение уравнения» на языке объектно-ориентированного программирования Visual Basic

1. В операционной системе Windows запустить систему объектно-ориентированного программирования Visual Basic командой [*Пуск-Все программы-Visual Basic 2010 Express*].
2. Разместить на форме:
   - графическое поле `PictureBox1`, в котором будет осуществляться построение графика функции $y = x^3 - \sin x$;
   - кнопку `Button1` для запуска обработчика события, реализующего построение графика.

В обработчике события осуществим преобразование компьютерной системы координат графического поля в математическую систему координат, удобную для построения графика функции. Нарисуем оси координат и нанесем на них шкалу.

В полученной математической системе координаты находятся в диапазонах $-150 \le X \le 150$ и $-100 \le Y \le 100$. Однако для поиска корней уравнения необходимо построить график функции в диапазоне аргумента $-1{,}5 \le X \le 1{,}5$, на котором функция принимает значения примерно в диапазоне $-1 \le Y \le 1$. Следовательно, необходимо увеличить масштаб графика в 100 раз:

- координаты точек графика необходимо умножить на 100;
- значения шкал осей разделить на 100.

Построение графика функции осуществим в цикле со счетчиком (аргумент X) с использованием метода рисования точки DrawEllipse(Pen1, X * 100, Y * 100, 1, 1), в котором координатами точки являются аргумент функции и значение функции.

3. Создать для кнопки Button1 обработчик события:

```
Dim Graph1 As Graphics
Dim Pen1 As New Pen(Color.Black, 2)
Dim drawBrush As New SolidBrush(Color.Black)
Dim drawFont As New Font("Arial", 10)
Dim X, Y As Single
Private Sub Button1_Click(…)
Graph1 = Me.PictureBox1.CreateGraphics()
Graph1.Clear(Color.White)
'Вывод шкал математической системы координат
'в компьютерной системе координат
For X = -150 To 150 Step 50
  Graph1.DrawString(X/100, drawFont, drawBrush,
                    X+150, 80)
Next X
For Y = 0 To 200 Step 50
  Graph1.DrawString((Y-100)/100, drawFont,
                    drawBrush, 150, 180-Y)
Next Y
'Преобразование компьютерной системы координат
'в математическую систему координат
Graph1.ScaleTransform(1, -1) 'Поворот оси Y
'Сдвиг по осям X и Y
Graph1.TranslateTransform(150, -100)
'Рисование осей математической системы координат
Graph1.DrawLine(Pen1, -150, 0, 300, 0) 'Ось X
Graph1.DrawLine(Pen1, 0, -100, 0, 100) 'Ось Y
For X = -150 To 150 Step 50 'Засечки на оси X
  Graph1.DrawLine(Pen1, X, -5, X, 5)
Next X
For Y = -100 To 100 Step 50 'Засечки на оси Y
  Graph1.DrawLine(Pen1, -5, Y, 5, Y)
Next Y
'График функции
For X = -1.5 To 1.5 Step 0.01
  Y = X^3 - Math.Sin(X)
  Graph1.DrawEllipse(Pen1, X*100, Y*100, 1, 1)
Next X
End Sub
```

4. Запустить проект на выполнение и щелкнуть по кнопке *График*. График функции пересекает ось X три раза, и следовательно, уравнение имеет три корня. По графику грубо приближенно можно определить, что $x_1 \approx \approx -0{,}9$, $x_2 \approx 0$ и $x_3 \approx 0{,}9$.

Проект «Графическое решение уравнения» на языке Visual Basic

Практическая работа 2.3

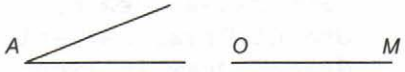
Выполнение геометрических построений в системе компьютерного черчения КОМПАС

Аппаратное и программное обеспечение. Компьютер с операционной системой Windows, система компьютерного черчения КОМПАС.

Цель работы. Научиться выполнять геометрические построения в системе компьютерного черчения КОМПАС.

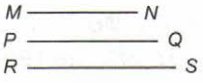
Задания. Выполнить в системе компьютерного черчения следующие геометрические построения.

2.3.1. Отложить от луча OM угол, равный заданному углу A.

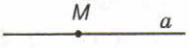


2.3.2. Построить треугольник по двум сторонам и углу между ними.

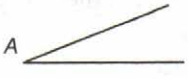
2.3.3. Построить треугольник по трем сторонам.



2.3.4. Даны прямая и точка на ней. Построить прямую, проходящую через данную точку и перпендикулярную к данной прямой.



2.3.5. Дан неразвернутый угол A. Построить его биссектрису.



Пояснение к рисованию объектов. Можно рисовать объект (отрезок, окружность, прямоугольник и др.) с помощью мыши или вводить данные об объекте вручную, введя их в соответствующие поля на *Панели свойств*, которая находится внизу системы компьютерного черчения. Например, для отрезка можно задать координаты его концов, длину, угол наклона и стиль линии.

Однако можно снять значения параметров с чертежа. Для подобного снятия параметров используется *Геометрический калькулятор*. Если в процессе рисования объекта установить курсор над каким-либо из полей и щелкнуть правой кнопкой мыши, на экране появляется меню команд *Геометрического калькулятора*, причем набор команд зависит от типа параметра.

Задание 2.3.1. Геометрическое построение угла, равного заданному

Формальная модель. Построим формальную модель процесса геометрического построения, зафиксировав его в форме алгоритма:

1) Построить угол A и отрезок OM.
2) Построить окружность произвольного радиуса с центром в точке A, обозначить точки пересечения окружности с отрезками угла буквами B и C.
3) Построить окружность того же радиуса с центром в точке O, обозначить точку пересечения окружности с отрезком угла буквой D.
4) Построить окружность с радиусом, равным длине отрезка BC, с центром в точке D, обозначить одну из точек пересечения окружностей буквой E.
5) Соединить отрезком точки O и E, угол EOD, равный углу A, построен.

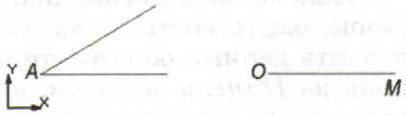
Начертим геометрические объекты, заданные в условии задачи: произвольный угол и отрезок.

1. С помощью *Компактной панели* вызвать панель *Геометрия*.

Выбрать объект *Отрезок* и построить сначала произвольный угол A (начертить два отрезка, выходящих из одной точки), а затем построить произвольный луч OM (начертить отрезок).

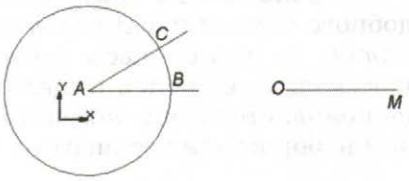
Введем обозначения точек на чертеже с помощью панели *Обозначения*.

2. С помощью *Компактной панели* вызвать панель *Обозначения*. Щелкнуть по кнопке *Ввод текста* и последовательно ввести обозначения угла и концов отрезка.



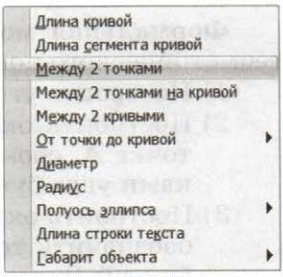
Построим окружность произвольного радиуса с центром в вершине заданного угла A, которая пересечет стороны угла в точках B и C.

3. На панели *Геометрия* выбрать объект *Окружность* и построить окружность с центром в точке A.
На панели *Обозначения* щелкнуть по кнопке *Ввод текста* и обозначить точки пересечения окружности со сторонами угла буквами B и C.

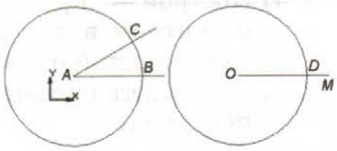


Построим окружность того же радиуса с центром в начале заданного луча OM, которая пересечет отрезок в точке D.

4. На панели *Геометрия* выбрать объект *Окружность*.
На *Панели свойств* щелкнуть правой кнопкой мыши по полю *Радиус* и в контекстном меню выбрать пункт *Между 2 точками*.
На чертеже навести курсор сначала на точку A, а затем на точку B.
Центр появившейся окружности заданного радиуса переместить в точку O.



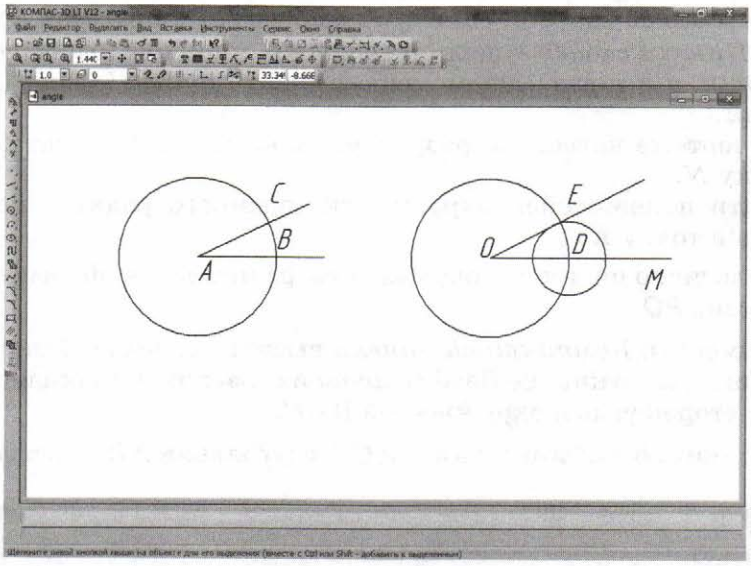
5. С помощью *Компактной панели* вызвать панель *Обозначения*. Щелкнуть по кнопке *Ввод текста* и обозначить точку пересечения окружности с отрезком OM буквой D.



Построим окружность с центром в точке D заданного радиуса BC.

6. На панели *Геометрия* выбрать объект *Окружность*.
На Панели свойств щелкнуть правой кнопкой мыши по полю *Радиус* и в контекстном меню выбрать пункт *Между 2 точками*.
На чертеже навести курсор сначала на точку C, а затем — на точку B.
Центр появившейся окружности заданного радиуса переместить в точку D.

7. С помощью *Компактной панели* вызвать панель *Обозначения*. Щелкнуть по кнопке *Ввод текста* и обозначить точку пересечения окружностей буквой E.
8. Соединить отрезком точки O и E, угол EOD, равный углу A, построен.



Задание 2.3.2. Построение треугольника по двум сторонам и углу между ними

Формальная модель. Построим формальную модель процесса геометрического построения, зафиксировав его в форме алгоритма:

1) Построить угол A и два отрезка MN и PQ.
2) Построить угол K, равный заданному углу A.
3) Отложить на сторонах угла K отрезки, равные заданным отрезкам MN и PQ путем построения двух окружностей соответствующих радиусов с центром в точке K. Обозначить точки пересечения окружностей со сторонами угла буквами B и C.
4) Соединить отрезком точки B и C. Треугольник KBC построен.

Начертим геометрические объекты, заданные в условии задачи: произвольный угол и два отрезка.

1. Построить произвольный угол A (начертить два отрезка, выходящих из одной точки). Построить два отрезка MN и PQ. Ввести обозначения точек на чертеже.

Построим угол, равный заданному.

2. Построить угол K, равный заданному углу A.

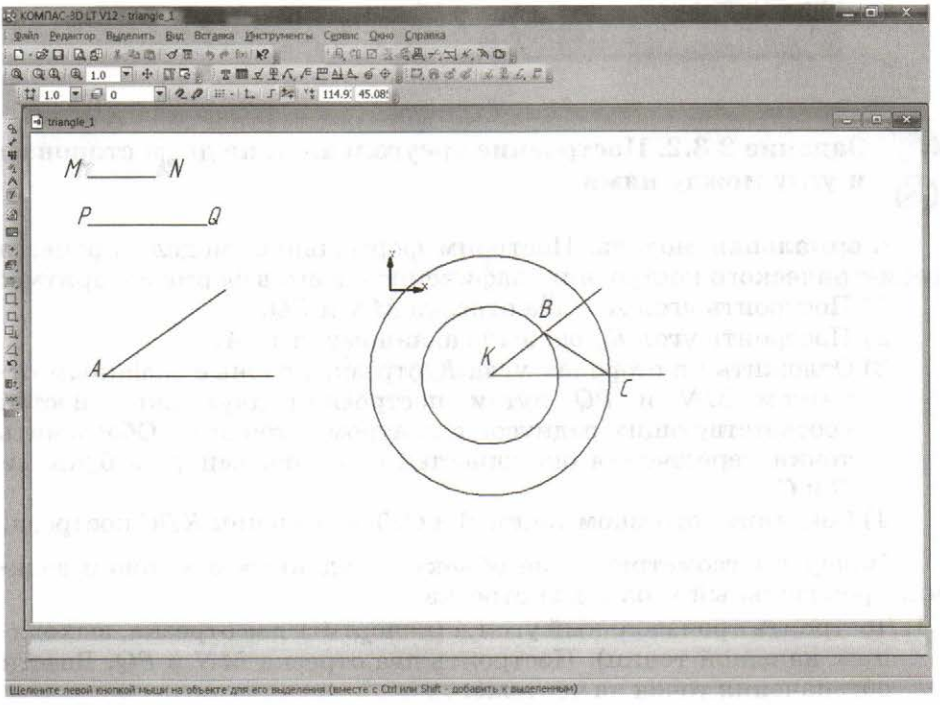
Отложим на сторонах угла отрезки, равные заданным отрезкам MN и PQ.

3. Построить окружность, радиус которой равен длине отрезка MN.
Для этого на панели *Геометрия* выбрать объект *Окружность*.
На *Панели свойств* щелкнуть правой кнопкой мыши по полю *Радиус* и в контекстном меню выбрать пункт *Между 2 точками*.
На чертеже навести курсор сначала на точку M, а затем — на точку N.
Центр появившейся окружности заданного радиуса переместить в точку K.

4. Аналогично построить окружность, радиус которой равен длине отрезка PQ.

5. С помощью *Компактной панели* вызвать панель *Обозначения*. Щелкнуть по кнопке *Ввод текста* и обозначить точки пересечения сторон угла и окружностей B и C.

6. Соединить отрезком точки B и C. Треугольник KBC построен.



Задание 2.3.3. Построение треугольника по трем сторонам

Формальная модель. Построим формальную модель процесса геометрического построения, зафиксировав его в форме алгоритма.

1) Построить три отрезка MN, PQ и RS, причем длина ни одного из них не должна превышать сумму длин двух других.
2) Провести прямую и отложить на ней отрезок, равный по длине заданному отрезку RS, обозначить его концы буквами A и B.
3) Построить две окружности, радиусы которых равны длинам заданных отрезков MN и PQ, с центрами в точках A и B. Обозначить точку пересечения окружностей буквой C.
4) Построить отрезки AC и BC. Треугольник ABC построен.

Начертим геометрические объекты, заданные в условии задачи: три отрезка (длина ни одного из них не должна превышать сумму длин двух других).

1. Построить три отрезка MN, PQ и RS и ввести обозначение точек на чертеже.

Проведем прямую и отложим на ней отрезок, равный по длине заданному отрезку RS.

2. Выбрать объект *Вспомогательная прямая* на панели *Геометрия* и начертить горизонтальную прямую.

3. Скопировать отрезок RS командой [*Редактор-Копировать*] и поместить его на горизонтальную прямую щелчком мышью. Обозначить концы отрезка буквами A и B.

Построим две другие стороны треугольника, равные заданным отрезкам MN и PQ.

Для этого построим окружность, радиус которой равен длине отрезка MN.

4. На панели *Геометрия* выбрать объект *Окружность*.

На *Панели свойств* щелкнуть правой кнопкой мыши по полю *Радиус* и в контекстном меню выбрать пункт *Между 2 точками*.

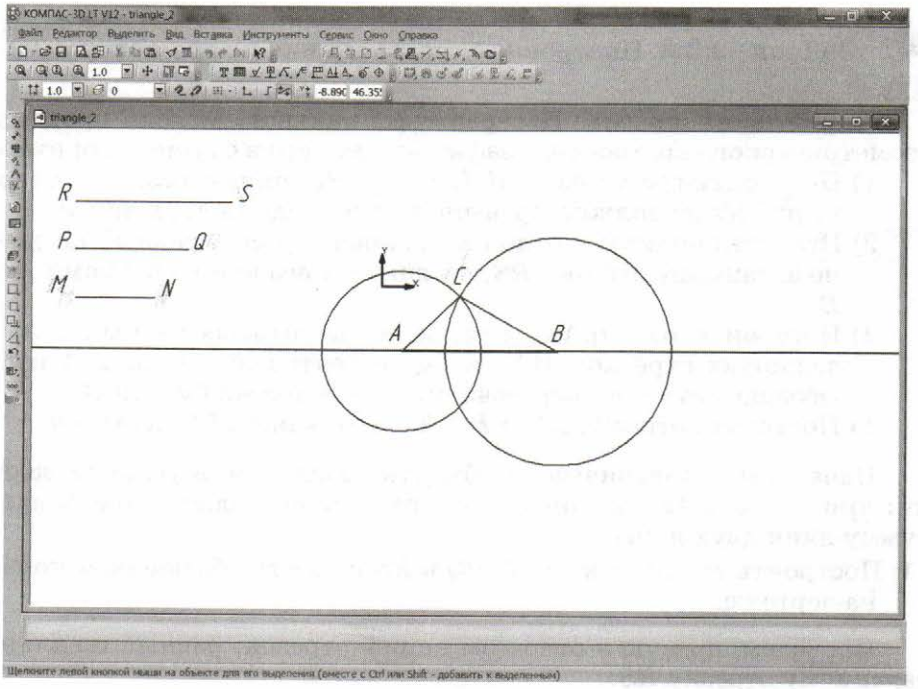
На чертеже навести курсор сначала на точку M, а затем —на точку N.

Центр появившейся окружности заданного радиуса переместить в точку A.

5. Аналогично построить окружность, радиус которой равен длине отрезка PQ, и поместить ее центр в точку B.

6. Пересечение окружностей обозначить буквой C.

7. Построить отрезки AC и BC. Треугольник ABC построен.



Задание 2.3.4. Построение перпендикуляра к заданной прямой

Формальная модель. Построим формальную модель процесса геометрического построения, зафиксировав его в форме алгоритма.

1) Построить прямую a и точку M на ней.
2) На равных расстояниях от точки M построить на прямой точки A и B путем рисования окружности с центром в точке M.
3) Построить две окружности равного радиуса с центрами в точках A и B.
4) Через точки пересечения окружностей P и Q провести прямую. Данная прямая пройдет через точку M и будет являться перпендикуляром к прямой a.

Начертим геометрические объекты, заданные в условии задачи: проведем прямую a и установим точку M на ней.

1. Выбрать объект *Вспомогательная прямая* на панели *Геометрия* и начертить горизонтальную прямую. Ввести обозначение «*a*».
2. Выбрать объект *Точка* на панели *Геометрия* и установить точку M на прямой a.

На прямой на равных расстояниях от точки M построим точки A и B.

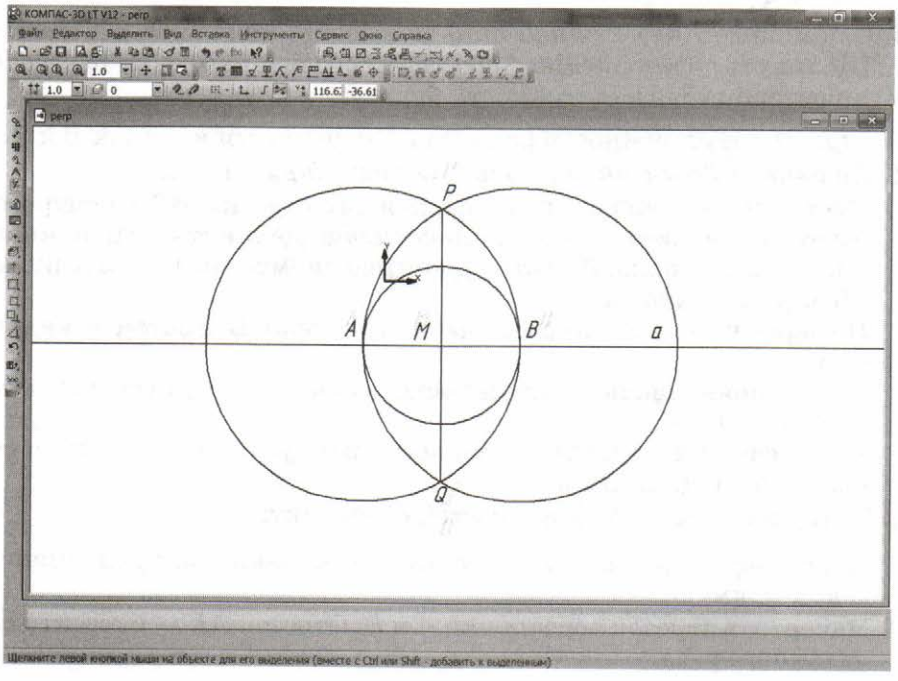
3. Выбрать объект *Окружность* и построить окружность произвольного радиуса. Обозначить пересечение окружности с прямой точками A и B.

Построим две окружности одинакового радиуса с центрами в точках A и B.

4. На панели *Геометрия* выбрать объект *Окружность*.
Построить окружность произвольного радиуса с центром в точке A.
На *Панели свойств* щелкнуть правой кнопкой мыши по полю *Радиус* и в контекстном меню выбрать пункт *Между 2 точками*.
На чертеже навести курсор сначала на точку A, а затем на точку пересечения окружности с линией.
Центр появившейся окружности заданного радиуса переместить в точку B.

Через точки пересечения окружностей P и Q проведем прямую.

5. Обозначить точки пересечения окружностей буквами P и Q. Соединить точки пересечения окружностей отрезком.
Данная прямая пройдет через точку M и будет являться перпендикуляром к прямой a.



Задание 2.3.5. Построение биссектрисы неразвернутого угла

Формальная модель. Построим формальную модель процесса геометрического построения, зафиксировав его в форме алгоритма.

1) Построить окружность произвольного радиуса с центром в вершине заданного угла A, которая пересечет стороны угла в точках B и C.
2) Построить две окружности радиуса BC с центрами в точках B и C. Точку пересечения окружностей внутри угла обозначить буквой E.
3) Через вершину угла A и точку пересечения окружностей E провести прямую. Отрезок AE — биссектриса заданного угла.

Начертим геометрические объекты, заданные в условии задачи: два отрезка, исходящих из одной точки под произвольным неразвернутым углом.

1. Построить два отрезка, исходящие из одной точки. Ввести обозначение угла на чертеже «A».

Построим окружность произвольного радиуса с центром в вершине заданного угла A.

2. Построить окружность произвольного радиуса с центром в точке A.

3. С помощью *Компактной панели* вызвать панель *Обозначения*. Щелкнуть по кнопке *Ввод текста* и обозначить точки пересечения сторон угла и окружности буквами B и C.

Построим две окружности радиуса BC с центрами в точках B и C.

4. На панели *Геометрия* выбрать объект *Окружность*.

Построим окружность радиуса, равного отрезку BC, с центром в точке B. Для этого на *Панели свойств* щелкнуть правой кнопкой мыши по полю *Радиус* и в контекстном меню выбрать пункт *Между 2 точками*.

На чертеже навести курсор сначала на точку B, а затем — на точку C.

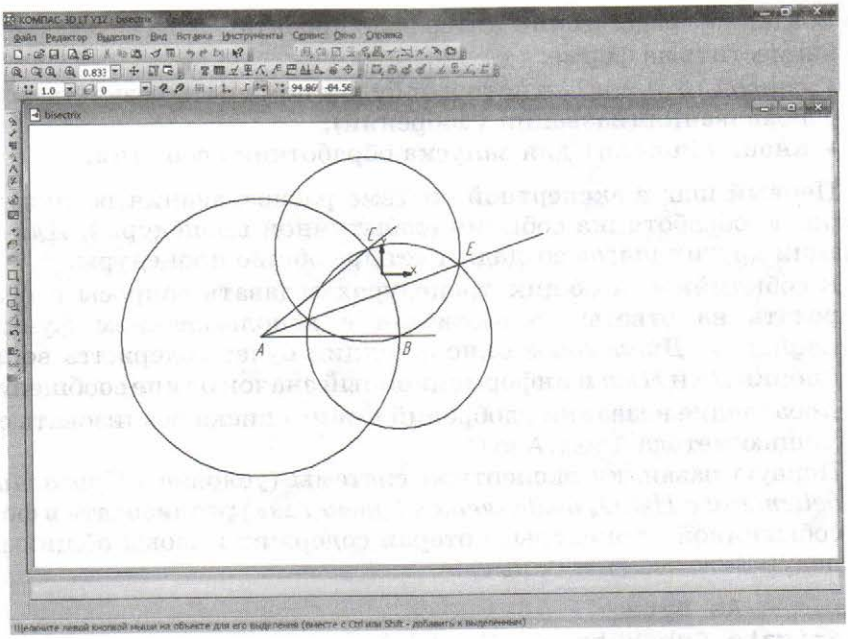
Центр появившейся окружности заданного радиуса переместить в точку B.

Аналогично построить окружность радиуса, равного длине отрезка BC, с центром в точке C.

5. Точку пересечения окружностей обозначить E.

Через вершину угла A и точку пересечения окружностей E проведем прямую.

6. Начертить отрезок через точки A и E. Отрезок AE — биссектриса заданного угла.



Практическая работа 2.4

Разработка проекта «Распознавание удобрений»

Аппаратное и программное обеспечение. Компьютер с установленной операционной системой Windows.

Цель работы. Научиться создавать компьютерные модели экспертных систем на языке объектно-ориентированного программирования Visual Basic.

Задание. Разработать проект, в котором необходимо создать экспертную систему распознавания удобрений. Даны удобрения, химические реактивы и справочная таблица по взаимодействию удобрений с некоторыми реактивами и предлагается распознать каждое из удобрений.

Задание. Проект «Распознавание удобрений» на языке объектно-ориентированного программирования Visual Basic

1. В операционной системе Windows запустить систему объектно-ориентированного программирования Visual Basic командой [*Пуск-Все программы-Visual Basic 2010 Express*].

Создадим графический интерфейс проекта.

2. Разместить на форме:
   - список ListBox1, в который будем помещать результаты распознавания (названия удобрений);
   - кнопку Button1 для запуска обработчика события.

Первый шаг в экспертной системе распознавания реализуем с помощью обработчика события (событийной процедуры). Для реализации других шагов создадим четыре общие процедуры.

3. В событийной и общих процедурах задавать вопросы и реагировать на ответы пользователя с использованием функции MsgBox(). Диалоговое окно функции будет содержать вопрос, кнопки *Да* и *Нет* и информационный значок о типе сообщения.
   Добавление названий удобрений в окно списка реализовать с помощью метода Item.Add().
   Первую развилку экспертной системы (условие «*При взаимодействии с H_2SO_4 выделяется бурый газ*») реализовать в форме событийной процедуры, которая содержит вызовы общих процедур Щелочь1() и Соль():

```
Dim A As Byte
Private Sub Button1_Click(…)
A = MsgBox("При взаимодействии с серной кислотой
           выделяется бурый газ?", 36,
           "Первый вопрос")
If A = 6 Then Щелочь1() Else Соль()
End Sub
```

4. Для распознавания удобрений первой группы (1-го и 2-го) создать общую процедуру Щелочь1() (условие «*При взаимодействии с раствором щелочи ощущается запах аммиака*»):

```
Sub Щелочь1()
A = MsgBox("При взаимодействии со щелочью ощущается
           запах аммиака?", 36,
           "Второй вопрос")
If A = 6 Then ListBox1.Items.Add
              ("1.Аммиачная селитра")
         Else ListBox1.Items.Add
              ("2.Натриевая селитра")
End Sub
```

5. Для распознавания удобрений второй группы создать общую процедуру Соль() (условие «*При взаимодействии с $BaCl_2$ выпадает белый осадок*»), которая содержит вызовы общих процедур Щелочь2() и Внешний_вид():

```
Sub Соль()
A = MsgBox("При взаимодействии с солью выпадает белый
      осадок?", 36, "Второй вопрос")
If A = 6 Then Щелочь2() Else Внешний_вид()
End Sub
```

6. Для распознавания 3-го и 4-го удобрений создать общую процедуру `Щелочь2()` (условие «*При взаимодействии с раствором щелочи ощущается запах аммиака*»):

```
Sub Щелочь2()
A = MsgBox("При взаимодействии со щелочью ощущается
      запах аммиака?", 36, "Третий вопрос")
If A = 6 Then
        ListBox1.Items.Add("3.Сульфат аммония")
     Else
        ListBox1.Items.Add("4.Суперфосфат")
End Sub
```

7. Для распознавания 5-го и 6-го удобрений создать общую процедуру `Внешний_вид()` (условие «*Внешний вид — розовые кристаллы*»):

```
Sub Внешний_вид()
A = MsgBox("Розовые кристаллы?", 36, "Третий вопрос")
If A = 6 Then ListBox1.Items.Add("5.Сильвинит")
Else ListBox1.Items.Add("6.Калийная соль")
End Sub
```

Компьютерный эксперимент. Работа с экспертной системой позволит более эффективно спланировать и провести распознавание удобрений в процессе выполнения лабораторной работы по химии.

8. Запустить проект щелчком по кнопке *Распознать*.

   Экспертная система начнет задавать вопросы, на которые надо отвечать на основе проводимых химических опытов. Проделать процедуру распознавания для каждого вещества.

   В результате в окно списка будут выведены названия всех удобрений.

Проект «Распознавание удобрений» на языке Visual Basic

Практическая работа 2.5

Разработка проекта «Модели систем управления»

Аппаратное и программное обеспечение. Компьютер с установленной операционной системой Windows.

Цель работы. Научиться создавать компьютерные модели систем управления без обратной связи, с обратной связью и автоматической обратной связью на языке объектно-ориентированного программирования Visual Basic.

Задание. Разработать проект, в котором управляемым объектом будет точка, которую управляющий объект (пользователь) должен переместить в центр мишени (окружности). Прямое управление положением точки будем производить путем нажатия на кнопки, которые перемещают объект влево и вправо, вверх и вниз. Рассмотрим три варианта:

1) обратная связь отсутствует, так как текущие положения точки в процессе управления невидимы;
2) обратная связь присутствует, так как текущие положения точки в процессе управления видимы;
3) присутствует автоматическая обратная связь .

Задание. Проект «Модели систем управления» на языке объектно-ориентированного программирования Visual Basic

1. В операционной системе Windows запустить систему объектно-ориентированного программирования Visual Basic командой [*Пуск-Все программы-Visual Basic 2010 Express*].

Вариант 1 «Модель системы управления без обратной связи»

2. Для создания графического интерфейса проекта разместить на форме:
   - графическое поле `PictureBox1`, по которому будет перемещаться точка;
   - кнопку `Button1` для запуска обработчика события, реализующего вывод первоначального положения точки и мишени (окружности);
   - четыре кнопки `Button2`, `Button3`, `Button4` и `Button5` для управления движением точки;
   - кнопку `Button6` для запуска обработчика события, реализующего вывод конечного положения точки.
3. Выделить объект `PictureBox1` и с помощью диалогового окна *Свойства* установить для свойства `Size` значение, например, `200;200`.

Создадим обработчик события, реализующего рисование в графическом поле точки (управляемого объекта) и окружности (мишени).

4. При каждом запуске обработчика события будем рисовать точку с различными координатами. Случайные значения координат получим с помощью оператора `Randomize()` и функции `Rnd()`, которая генерирует случайные числа в интервале $0 \le X < 1$. Начальные координаты точки должны быть заданы целыми числами и соответствовать размерам графического поля, т. е. $0 \le X < 200$ и $0 \le Y < 200$. Поэтому значения функции `Rnd()` необходимо увеличить в 200 раз и выделить целую часть числа с использованием функции `Int()`.

   Для рисования в графическом поле точки и окружности использовать графические методы.

```
Dim Graph1 As Graphics
Dim Pen1 As New Pen(Color.Black, 3)
Dim Brush1 As New SolidBrush(Color.Black)
Dim X, Y As Integer
Private Sub Button1_Click(…)
Graph1 = Me.PictureBox1.CreateGraphics()
Graph1.Clear(Color.White)
Randomize()
X = Int(Rnd() * 200)
Y = Int(Rnd() * 200)
Graph1.DrawEllipse(Pen1, X, Y, 2, 2)
Graph1.FillEllipse(Brush1, X, Y, 2, 2)
Graph1.DrawEllipse(Pen1, 90, 90, 20, 20)
End Sub
```

5. Четыре обработчика событий перемещения точки влево и вправо, вверх и вниз должны обеспечивать соответствующие изменения координат точки. В компьютерной системе координат для перемещения влево координата точки X должна уменьшаться, а для перемещения вправо — увеличиваться. Для перемещения вниз координата точки Y должна увеличиваться, а для перемещения вверх — уменьшаться.

   Обратная связь должна отсутствовать, поэтому текущие положения точки не будут отображаться в графическом поле.

   Например, обработчик события перемещения точки влево будет следующим:

```
Private Sub Button3_Click(…)
X = X - 1
End Sub
```

6. Возможность увидеть результаты управления, т. е. попала точка в центр мишени или нет, обеспечивает обработчик события вывода конечного положения точки:

```
Private Sub Button6_Click(…)
Graph1.DrawEllipse(Pen1, X, Y, 2, 2)
Graph1.FillEllipse(Brush1, X, Y, 2, 2)
End Sub
```

7. Запустить проект и для вывода первоначального положения точки и мишени щелкнуть по кнопке *Мишень и точка*.
Щелчками по кнопкам управления перемещением точки (кнопки со стрелками) постараться переместить точку в центр окружности.
Щелкнуть по кнопке *Результат*. Управление перемещением точки производится без обратной связи, поэтому попасть в центр окружности довольно трудно.

Проект «Модель системы управления без обратной связи» на языке Visual Basic

Вариант 2 «Модель системы управления с обратной связью»

За основу возьмем проект «Модель системы управления без обратной связи». Для осуществления обратной связи будем рисовать текущее положение точки в графическом поле, а также выводить текущие координаты точки на метки.

8. Разместить на форме две метки `Label1` и `Label2` для вывода текущих координат точки.

9. В программные коды обработчиков событий перемещения точки добавить строки рисования точки и вывода на метки ее текущих координат. Обработчик события перемещения точки влево примет вид:

```
Private Sub Button3_Click(…)
X = X - 1
Graph1.DrawEllipse(Pen1, X, Y, 2, 2)
Graph1.FillEllipse(Brush1, X, Y, 2, 2)
Label1.Text = X
Label2.Text = Y
End Sub
```

10. Запустить проект и осуществить попадание точки в мишень, имеющую координаты центра окружности (100,100).
Легко убедиться, что использование обратной связи обеспечивает уверенное попадание точки в центр мишени.

Проект «Модель системы управления с обратной связью» на языке Visual Basic

Вариант 3 «Модель системы управления с автоматической обратной связью»

За основу возьмем проект «Модель системы управления с обратной связью».

11. Уберем из графического интерфейса:
четыре кнопки `Button2`, `Button3`, `Button4` и `Button5` для управления движением точки и кнопку `Button6`, выводящую результат.

12. Поместим на графический интерфейс:
кнопку `Button2` для создания автоматической обратной связи.

Для осуществления автоматической обратной связи используем корректировку координат с использованием инструкции выбора **`Select Case`**.

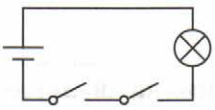
13. Создать обработчик события, реализующий корректировку положения точки:

```
Private Sub Button2_Click_1(…)
X2 = 100
Y2 = 100
'Автоматическая корректировка координаты X
Select Case X2 - X1
Case Is > 0
  X1 = X1 + 1
Case Is < 0
  X1 = X1 - 1
```

```
Case Is = 0
  X1 = X1
End Select
'Автоматическая корректировка координаты Y
Select Case Y2 - Y1
Case Is > 0
  Y1 = Y1 + 1
Case Is < 0
  Y1 = Y1 - 1
Case Is = 0
  Y1 = Y1
End Select
Label1.Text = X1
Label2.Text = Y1
Graph1.DrawEllipse(Pen1, X1, Y1, 2, 2)
Graph1.FillEllipse(Brush1, X1, Y1, 2, 2)
End Sub
```

14. Запустить проект и нажатием кнопки *Шаг* осуществить попадание точки в мишень, имеющую координаты центра окружности (100,100).

Проект «Модель системы управления с автоматической обратной связью» на языке Visual Basic



Глава 3
ЛОГИКА И ЛОГИЧЕСКИЕ ОСНОВЫ КОМПЬЮТЕРА

3.1. Алгебра логики

Первые учения о формах и способах рассуждений возникли в странах Древнего Востока (Китай, Индия), но в основе современной логики лежат учения, созданные древнегреческими мыслителями. Основы формальной логики заложил Аристотель, который впервые отделил логические формы мышления (речи) от его содержания.

Логика — это наука о формах и способах мышления.

Законы логики отражают в сознании человека свойства, связи и отношения объектов окружающего мира. Логика позволяет строить формальные модели окружающего мира, отвлекаясь от содержательной стороны.

Алгебра в широком смысле этого слова — наука об общих операциях, аналогичных сложению и умножению, которые могут выполняться над различными математическими объектами (алгебра переменных и функций, алгебра векторов, алгебра множеств и т. д.). Объектами алгебры логики являются высказывания.

Высказывания — это повествовательные предложения, о которых можно однозначно сказать, истинны они или ложны.

Алгебра логики отвлекается от смысловой содержательности высказываний. Ее интересует только один факт — истинно или ложно данное высказывание, что дает возможность определять истинность или ложность составных (составленных из простых) высказываний алгебраическими методами.

Логические переменные. Простые высказывания в алгебре логики обозначаются заглавными латинскими буквами — именами **логических переменных.** Высказывания могут быть истинными

или ложными. Истинному высказыванию соответствует значение логической переменной 1, а ложному — значение 0.

В алгебре логики высказывания обозначаются **именами логических переменных**, которые могут принимать лишь два значения: «истина» (1) и «ложь» (0).

Рассмотрим два простых высказывания:
A — {Два умножить на два равно четырем}.
B — {Два умножить на два равно пяти}.
Первое высказывание истинно ($A = 1$), а второе ложно ($B = 0$).

Составные высказывания на естественном языке образуются с помощью связок «и», «или», «не», которые в алгебре логики заменяются на **логические операции**. Рассмотрим базовые логические операции.

Логическое умножение (конъюнкция). Объединение двух (или нескольких) высказываний в одно с помощью союза «и» называется операцией логического умножения или конъюнкцией.

Составное высказывание, образованное в результате **операции логического умножения (конъюнкции)**, истинно тогда и только тогда, когда истинны входящие в него простые высказывания.

Из приведенных ниже четырех составных высказываний, образованных с помощью операции логического умножения, истинно только четвертое, так как в первых трех составных высказываниях хотя бы одно из простых высказываний ложно:

(1) «$2 \times 2 = 5$ и $3 \times 3 = 10$»
(2) «$2 \times 2 = 5$ и $3 \times 3 = 9$»
(3) «$2 \times 2 = 4$ и $3 \times 3 = 10$»
(4) «$2 \times 2 = 4$ и $3 \times 3 = 9$»

Перейдем теперь от записи высказываний на естественном языке к их записи на формальном языке алгебры логики. Операцию логического умножения (конъюнкцию) принято обозначать значком &. Операция логического умножения, аргументами которой являются логические переменные A и B, записывается следующей формулой

$$A \,\&\, B. \qquad (3.1)$$

Значение операции логического умножения задается с помощью таблицы истинности. **Таблица истинности** показывает, ка-

кие значения дает логическая операция при всех возможных наборах ее аргументов. Результатом операции логического умножения является значение «истина» (1) тогда и только тогда, когда оба аргумента принимают значения «истина» (1) (табл. 3.1).

Таблица 3.1. **Таблица истинности конъюнкции (логического умножения)**

A	B	A & B
0	0	0
0	1	0
1	0	0
1	1	1

Таблица истинности задает **логическую функцию**, аргументы которой — логические переменные (принимают значение либо 0, либо 1), а значение функции — результат логической операции над этими переменными (тоже значение либо 0, либо 1).

По таблице истинности легко определить истинность составного высказывания, образованного с помощью операции логического умножения. Рассмотрим, например, составное высказывание

«2 × 2 = 4 и 3 × 3 = 10».

Первое простое высказывание истинно ($A = 1$), а второе высказывание ложно ($B = 0$), по таблице истинности логического умножения определяем, что данное составное высказывание ложно.

Логическое сложение (дизъюнкция). Объединение двух (или нескольких) высказываний с помощью союза «или» называется операцией логического сложения или дизъюнкцией.

Составное высказывание, образованное в результате **логического сложения (дизъюнкции)**, истинно тогда и только тогда, когда истинно хотя бы одно из входящих в него простых высказываний.

Так, из приведенных ниже четырех составных высказываний, образованных с помощью операции логического сложения, ложно только первое, так как в последних трех составных высказываниях хотя бы одно из простых высказываний истинно:

(1) «2 × 2 = 5 или 3 × 3 = 10»
(2) «2 × 2 = 5 или 3 × 3 = 9»
(3) «2 × 2 = 4 или 3 × 3 = 10»
(4) «2 × 2 = 4 или 3 × 3 = 9»

Запишем теперь операцию логического сложения на формальном языке алгебры логики. Операцию логического сложения (дизъюнкцию) принято обозначать значком $\vee$. Операция логического сложения, аргументами которой являются логические переменные A и B, записывается следующей формулой:

$$A \vee B. \qquad (3.2)$$

Значение операции логического сложения задается с помощью таблицы истинности. Результатом операции логического сложения является значение «ложь» (0) тогда и только тогда, когда оба аргумента принимают значения «ложь» (0) (табл. 3.2).

Таблица 3.2. **Таблица истинности дизъюнкции (логического сложения)**

A	B	$A \vee B$
0	0	0
0	1	1
1	0	1
1	1	1

По таблице истинности легко определить истинность составного высказывания, образованного с помощью операции логического сложения. Рассмотрим, например, составное высказывание

«$2 \times 2 = 4$ или $3 \times 3 = 10$».

Первое простое высказывание истинно ($A = 1$), а второе высказывание ложно ($B = 0$); с помощью таблицы истинности логического сложения определяем, что данное составное высказывание истинно.

Логическое отрицание (инверсия). Присоединение частицы «не» к высказыванию называется **операцией логического отрицания** или **инверсией.**

Логическое отрицание (инверсия) делает истинное высказывание ложным и, наоборот, ложное — истинным.

Высказывание «Два умножить на два равно четырем» истинно, а высказывание, образованное с помощью операции логического отрицания, «Два умножить на два не равно четырем» — ложно.

Запишем теперь операцию логического отрицания на формальном языке алгебры логики. Операцию логического отрицания (инверсию) над логическим высказыванием A принято обозначать $\overline{A}$

или $\neg A$. Операция логического отрицания, аргументом которой является логическая переменная A, записывается следующей формулой:

$$\overline{A}. \qquad (3.3)$$

Значение логической операции отрицания задается с помощью таблицы истинности. Результатом операции логического отрицания является значение «истина» (1), когда аргумент принимает значение «ложь» (0), и значение «ложь» (0), когда аргумент принимает значение «истина» (1) (табл. 3.3).

Таблица 3.3. **Таблица истинности инверсии (логического отрицания)**

A	$\overline{A}$
0	1
1	0

Истинность высказывания, образованного с помощью операции логического отрицания, можно легко определить с помощью таблицы истинности. Например, высказывание «Два умножить на два не равно четырем» ложно ($A = 0$), а полученное из него в результате логического отрицания высказывание «Два умножить на два равно четырем» истинно ($\overline{A} = 1$).

Контрольные вопросы

1. Что изучает наука логика?
2. Что такое высказывание?
3. Что такое логические переменные и какие значения они принимают?
4. Какие логические операции вы знаете? Как обозначаются логические операции в высказываниях на естественном языке и на языке алгебры логики?
5. Что такое таблица истинности?

3.2. Логические основы устройства компьютера

3.2.1. Базовые логические элементы

Дискретный преобразователь, который после обработки входных двоичных сигналов выдает на выходе сигнал, являющийся значением одной из логических операций, называется **логическим элементом**. Базовые логические элементы реализуют три базовые логические операции:

- логический элемент «И» (конъюнктор) — логическое умножение;
- логический элемент «ИЛИ» (дизъюнктор) — логическое сложение;
- логический элемент «НЕ» (инвертор) — инверсию.

Любая логическая операция может быть представлена в виде комбинации трех базовых, поэтому любые устройства компьютера, производящие обработку или хранение информации (сумматоры в процессоре, ячейки памяти в оперативной памяти и др.), могут быть собраны из базовых логических элементов.

Логические элементы компьютера оперируют с сигналами, представляющими собой электрические импульсы. Есть импульс — логическое значение сигнала 1, нет импульса — значение 0. На входы логических элементов поступают сигналы-аргументы, на выходе появляется сигнал — значение функции.

Преобразование сигнала логическим элементом задается таблицей состояния, которая фактически является таблицей истинности, соответствующей логической функции.

Логический элемент «И» — конъюнктор (рис. 3.1). На входы *A* и *B* логического элемента последовательно подадим четыре пары сигналов, на выходе получим последовательность из четырех сигналов, значения которых определяются в соответствии с таблицей истинности операции логического умножения.

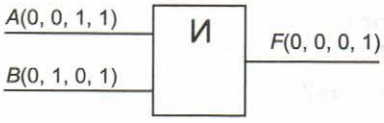
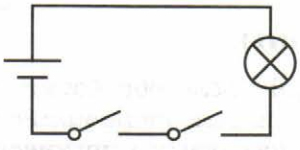


Рис. 3.1. Логический элемент «И» — конъюнктор

Простейшей моделью логического элемента «И» может быть электрическая схема, состоящая из источника тока, лампочки и двух выключателей (рис. 3.2). Данную схему можно собрать из реальных электрических элементов или с использованием компьютерного конструктора «Начала электроники».

Из схемы видно, что если оба выключателя замкнуты (на обоих входах 1), по цепи идет ток и лампочка горит (на выходе 1).



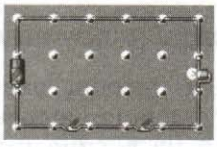


Рис. 3.2. Электрическая схема модели логического элемента «И» и ее реализация в компьютерном конструкторе «Начала электроники»

Логический элемент «ИЛИ» — дизъюнктор (рис. 3.3). На входы *A* и *B* логического элемента последовательно подадим четыре пары сигналов, а на выходе получим последовательность из четырех сигналов, значения которых определяются в соответствии с таблицей истинности операции логического сложения.

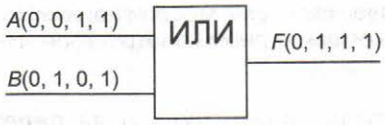


Рис. 3.3. Логический элемент «ИЛИ» — дизъюнктор

Простейшей моделью логического элемента «ИЛИ» может быть электрическая схема, которую можно собрать из реальных электрических элементов или с использованием компьютерного конструктора «Начала электроники» (рис. 3.4).

Рис. 3.4. Электрическая схема модели логического элемента «ИЛИ» и ее реализация в компьютерном конструкторе «Начала электроники»

Из схемы видно, что если хотя бы один выключатель замкнут (на входе 1), по цепи идет ток и лампочка горит (на выходе 1).

Логический элемент «НЕ» — инвертор (рис. 3.5). На вход *A* логического элемента последовательно подадим два сигнала, на выходе получим последовательность из двух сигналов, значения которых определяются в соответствии с таблицей истинности логической инверсии.

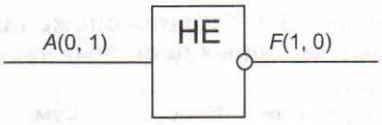


Рис. 3.5. Логический элемент «НЕ»

Простейшей моделью логического элемента «НЕ» может быть электрическая схема, которую можно собрать из реальных электрических элементов или с использованием компьютерного конструктора «Начала электроники» (рис. 3.6).

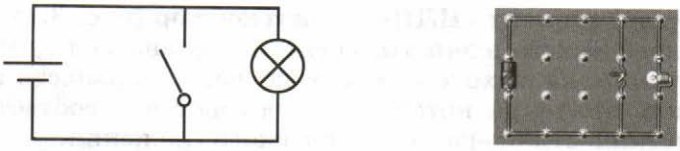


Рис. 3.6. Электрическая схема модели логического элемента «НЕ» и ее реализация в компьютерном конструкторе «Начала электроники»

Из схемы инвертора видно, что когда переключатель не замкнут (на входе 0), лампочка горит (на выходе 1). Наоборот, когда кнопку переключателя замыкают (на входе 1), лампочка гаснет (на выходе 0).

Контрольные вопросы

1. Объясните действие электрических схем, реализующих модели логических элементов, с точки зрения законов постоянного тока.

3.2.2. Сумматор двоичных чисел

В целях максимального упрощения работы компьютера всё многообразие математических операций в процессоре сводится к сложению двоичных чисел. Поэтому главной частью процессора является сумматор, который как раз и обеспечивает такое сложение. Сумматор составляется из базовых логических элементов.

Полусумматор. При сложении двоичных чисел образуется сумма в данном разряде, при этом возможен перенос в старший разряд. Обозначим слагаемые — A и B, перенос — P и сумму — S. Таблица сложения одноразрядных двоичных чисел с учетом переноса в старший разряд выглядит следующим образом (табл. 3.4).

Таблица 3.4. **Таблица сложения одноразрядных двоичных чисел**

Слагаемые		Перенос	Сумма
A	*B*	*P*	*S*
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Из этой таблицы видно, что перенос можно реализовать с помощью операции логического умножения:

$$P = A \,\&\, B.$$

Получим теперь формулу для вычисления суммы. Значения суммы более всего совпадают с результатом операции логического сложения (кроме случая, когда на входы подаются две единицы, а на выходе должен получиться ноль).

Нужный результат достигается, если результат логического сложения умножить на инвертированный перенос. Таким образом, для определения суммы можно применить следующую логическую функцию:

$$S = (A \vee B) \,\&\, \overline{(A \,\&\, B)}.$$

Построим таблицу истинности для данной логической функции и убедимся в правильности нашего предположения (табл. 3.5).

Таблица 3.5. **Таблица истинности логической функции $S = (A \vee B) \,\&\, \overline{(A \,\&\, B)}$**

A	B	$A \vee B$	$A \,\&\, B$	$\overline{(A \,\&\, B)}$	$(A \vee B) \,\&\, \overline{(A \,\&\, B)}$
0	0	0	0	1	0
0	1	1	0	1	1
1	0	1	0	1	1
1	1	1	1	0	0

Теперь на основе полученных логических формул можно построить из базовых логических элементов схему сложения одноразрядных двоичных чисел.

По логической формуле переноса можно определить, что для получения переноса необходимо использовать логический элемент «И».

Анализ логической формулы для суммы показывает, что на выходе должен стоять элемент логического умножения «И», который имеет два входа. На один из входов подается результат логического сложения исходных величин $A \vee B$, т. е. на него должен подаваться сигнал с элемента логического сложения «ИЛИ».

На второй вход требуется подать результат инвертированного логического умножения исходных сигналов $\overline{A \,\&\, B}$, т. е. на второй вход подается сигнал с элемента «НЕ», на вход которого поступает сигнал с элемента логического умножения «И» (рис. 3.7).

Данная схема называется **полусумматором**, так как реализует суммирование одноразрядных двоичных чисел A и B без учета переноса из младшего разряда.

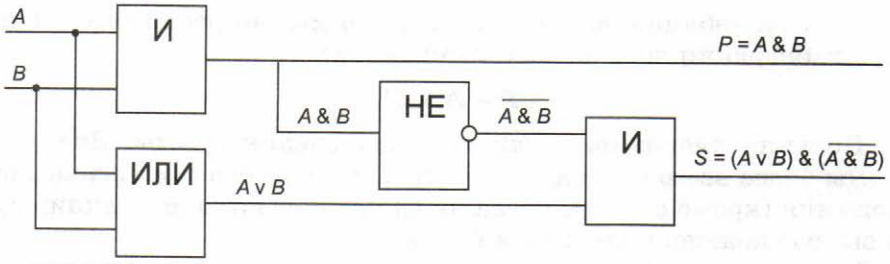


Рис. 3.7. Полусумматор двоичных чисел

Для сложения многоразрядных двоичных чисел служит **сумматор**, который составляется из полусумматоров.

Контрольные вопросы

1. Какие значения будут иметь перенос и сумма при суммировании одноразрядных двоичных чисел, равных 1, и переноса из младшего разряда?

Практические работы компьютерного практикума к главе 3 «Логика и логические основы компьютера»

	Установить: • электронные таблицы OpenOffice.org Calc; • компьютерный конструктор «Начала электроники»; • электронные таблицы Microsoft Excel	http://ru.openoffice.org  http://www.edsoft.ru/fizika/294.html  http://www.shkolaedu.ru/products/43 
	Установить: • электронные таблицы OpenOffice.org Calc	http://altlinux.org/Альт-Линукс-5.0.2-Школьный 

Практическая работа 3.1

Таблицы истинности логических функций

Аппаратное и программное обеспечение. Компьютер с установленной операционной системой Windows или Linux.

Цель работы. Научиться создавать таблицы истинности базовых логических функций с использованием электронных таблиц.

Задание. Получить таблицы истинности операций логического умножения, логического сложения и логического отрицания с использованием электронных таблиц.

В электронных таблицах логические операции осуществляются с помощью встроенных логических функций.

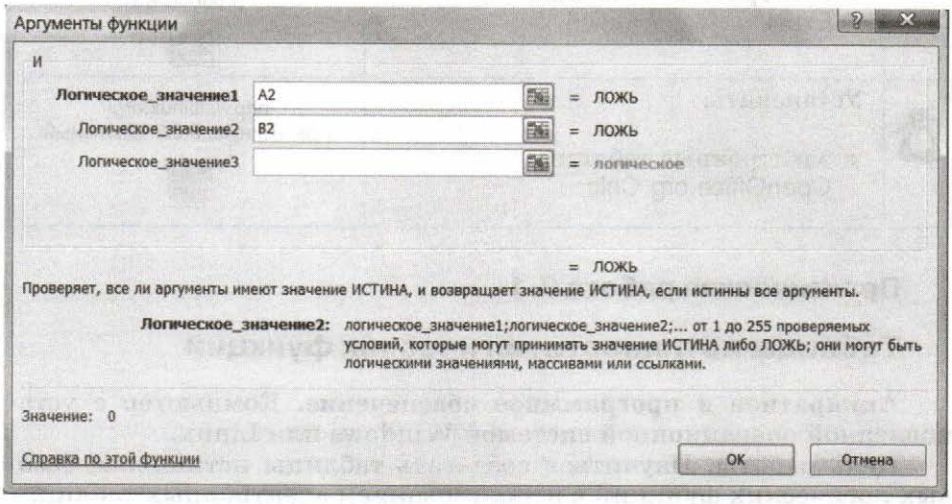
Задание. Определение значений логических функций с использованием электронных таблиц Microsoft Excel

В электронных таблицах Microsoft Excel функция логического умножения И(логическое значение1;логическое значение2;...) дает значение ИСТИНА (1) тогда и только тогда, когда все логические аргументы имеют значение ИСТИНА (1).

Функция логического сложения ИЛИ(логическое значение1; логическое значение2;...) дает значение ИСТИНА (1) тогда и только тогда, когда хотя бы один логический аргумент имеет значение ИСТИНА (1).

Функция логического отрицания НЕ(логическое значение) дает значение ИСТИНА (1), когда логический аргумент имеет значение ЛОЖЬ (0) и, наоборот, значение ЛОЖЬ (0), когда логический аргумент имеет значение ИСТИНА (1).

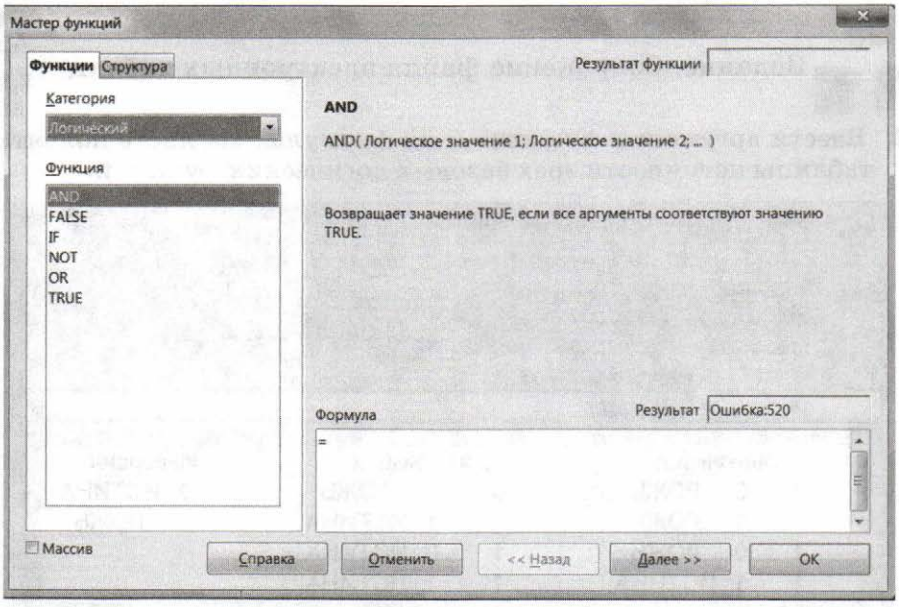
1. Для ввода логических функций воспользоваться командой [*Формулы-Логические-И*].
2. В диалоговом окне *Аргументы функции* в текстовых полях *Логическое значение 1* и *Логическое значение 2* выбрать имена ячеек, в которых хранятся аргументы логической функции. Щелкнуть по кнопке *ОК*.



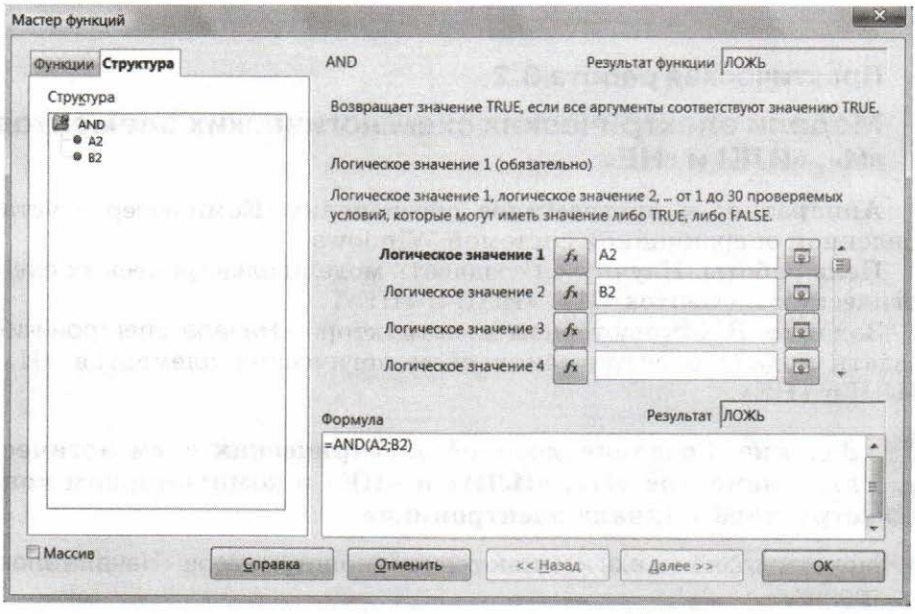
Задание. Ввод логических функций с использованием электронных таблиц OpenOffice.org Calc

В электронных таблицах OpenOffice.org Calc функция логического умножения обозначается AND(), функция логического сложения — OR() и функция логического отрицания — NOT().

1. Для ввода логических функций воспользоваться командой [*Вставка-Функция...*].
   В появившемся диалоговом окне *Мастер функций* в раскрывающемся списке *Категория* выбрать *Логический*, а в окне *Функция* — функцию *AND*. Щелкнуть по кнопке *Далее*.

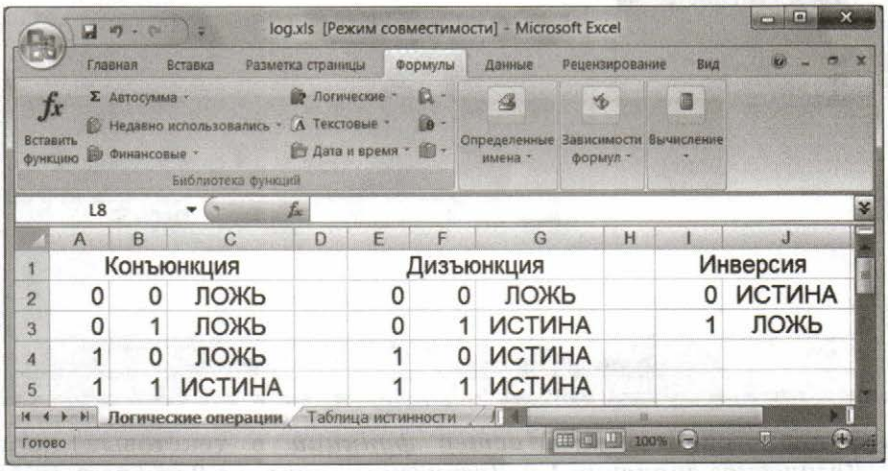


2. В диалоговом окне *Мастер функций* в текстовых полях *Логическое значение 1* и *Логическое значение 2* выбрать имена ячеек, в которых хранятся аргументы логической функции. Щелкнуть по кнопке *ОК*.



Задание. Сохранение файла электронных таблиц

3. Ввести аргументы функций и их формулы. На листе появятся таблицы истинности трех базовых логических функций.



4. Переименовать лист *Лист1* в *Логические операции*.
5. Сохранить файл электронных таблиц.

Практическая работа 3.2

Модели электрических схем логических элементов «И», «ИЛИ и «НЕ»

Аппаратное и программное обеспечение. Компьютер с установленной операционной системой Windows.

Цель работы. Научиться создавать модели электрических схем логических элементов «И», «ИЛИ и «НЕ».

Задание. В компьютерном конструкторе «Начала электроники» создать модели электрических схем логических элементов «И», «ИЛИ и «НЕ».

Задание. Создание моделей электрических схем логических элементов «И», «ИЛИ» и «НЕ» в компьютерном конструкторе «Начала электроники»

1. Запустить файл e.exe компьютерного конструктора «Начала электроники».

Создадим электрическую схему логического элемента «И».

2. Поместить на монтажный стол последовательно соединенные элемент питания, два выключателя и лампочку (см. рис. 3.2).

Создадим электрическую схему логического элемента «ИЛИ».

3. Поместить на монтажный стол два параллельно соединенных выключателя, элемент питания и лампочку (см. рис. 3.4).

Создадим электрическую схему логического элемента «НЕ».

4. Поместить на монтажный стол параллельно соединенные элемент питания, выключатель и лампочку (см. рис. 3.6).

Глава 4
ИНФОРМАЦИОННОЕ ОБЩЕСТВО И ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ

4.1. Информационное общество

Доиндустриальное общество

Человеческое общество по мере своего развития прошло этапы овладения веществом, затем энергией, и, наконец, информацией.

В первобытно-общинном, рабовладельческом и феодальном обществах деятельность общества в целом и каждого человека в отдельности была направлена в первую очередь на овладение веществом.

На заре цивилизации (десятки тысяч лет до нашей эры) люди научились изготавливать простые орудия труда и охоты (каменный топор, стрелы и т. д.), в античности появились первые механизмы (рычаг и др.) и средства передвижения (колесницы, корабли), в средние века были изобретены первые сложные орудия труда и механизмы (ткацкий станок, часы).

Овладение энергией находилось в этот период на начальной ступени, в качестве источников энергии использовались Солнце, вода, огонь, ветер и мускульная сила человека.

С самого начала человеческой истории возникла потребность передачи и хранения информации. Для передачи информации сначала использовался язык жестов, а затем человеческая речь. Для хранения информации стали использоваться наскальные рисунки, а в IV тысячелетии до нашей эры появились письменность и первые носители информации (шумерские глиняные таблички и египетские папирусы). История создания устройств для обработки числовой информации начиналась также еще в древности — с абака (счетной доски, являющейся прообразом счетов).

Индустриальное общество

Начиная примерно с XVII века, в процессе становления машинного производства, на первый план выходит проблема овладения энергией (машины и станки необходимо было приводить в движе-

ние). Сначала совершенствовались способы овладения энергией ветра и воды (ветряные мельницы и водяные колеса), а затем человечество овладело тепловой энергией (в середине XVIII века была изобретена паровая машина, а в конце XIX века — двигатель внутреннего сгорания).

В конце XIX века началось овладение электрической энергией, были изобретены электрогенератор и электродвигатель. И наконец, в середине XX века человечество овладело атомной энергией. В 1954 году в СССР была пущена в эксплуатацию первая атомная электростанция.

Овладение энергией позволило перейти к массовому машинному производству потребительских товаров, было создано индустриальное общество. Основными показателями развитости индустриального общества являлись количественные показатели, т. е. сколько было добыто угля и нефти, сколько произведено станков и т. д.

В этот период происходили также существенные изменения в способах хранения и передачи информации. В середине XV века было изобретено книгопечатание, что позволило сделать информацию доступной для гораздо большего количества людей. С конца XIX века для передачи информации на дальние расстояния по проводам стали широко использоваться телеграф и телефон, а в XX веке — электромагнитные волны (радио, телевидение).

Информационное общество

Первой попыткой автоматизированной обработки информации стало создание Чарльзом Бэббиджем в середине XIX века механической цифровой Аналитической машины. Однако лишь с середины XX века, с момента появления электронных устройств обработки и хранения информации (ЭВМ, а затем персонального компьютера), начался постепенный переход от индустриального общества к информационному.

В информационном обществе главным ресурсом является информация, именно на основе владения информацией о самых различных процессах и явлениях можно эффективно и оптимально строить любую деятельность.

Важно не только произвести большое количество продукции, но и сделать нужную продукцию в определенное время и с определенными затратами. В информационном обществе повышается не только качество потребления, но и качество производства. Человек, использующий информационные технологии, имеет лучшие условия труда, а труд становится творческим и интеллектуальным.

В качестве **критериев развитости** информационного общества можно выбрать три: наличие компьютеров, уровень развития компьютерных сетей и количество населения, занятого в информационной сфере, а также использующего информационные и коммуникационные технологии в своей повседневной деятельности.

Производство компьютеров. Первые электронные вычислительные машины (ЭВМ), которые могли автоматически по заданной программе обрабатывать большие объемы информации, были созданы в 1946 году в США (ЭНИАК) и в 1950 году в СССР (МЭСМ). В 40–60-х годах XX века производство ЭВМ измерялось единицами, десятками и, в лучшем случае, сотнями экземпляров. ЭВМ были очень дорогими и очень большими (занимали громадные залы), поэтому оставались недоступными для массового потребителя.

Массовое производство сравнительно недорогих персональных компьютеров началось с середины 1970-х годов с компьютера Apple II (с этого компьютера начала свое существование фирма Apple). Количество произведенных персональных компьютеров начало составлять десятки тысяч в год, что по тем временам было колоссальным достижением.

В начале 1980-х годов приступила к массовому производству персональных компьютеров корпорация IBM (компьютеры так и назывались IBM Personal Computer — IBM PC). Достаточно скоро IBM-совместимые компьютеры стали выпускать многие фирмы, и их производство достигло сотен тысяч в год. Ежегодное производство персональных компьютеров постоянно росло и в 2011 году превысило 300 миллионов.

Персональный компьютер постоянно совершенствовался, его производительность возросла на три порядка, при этом, что очень важно, цена даже снизилась. Персональный компьютер стал доступен массовому потребителю, и теперь в развитых странах мира компьютер имеется на большинстве рабочих мест и в большинстве семей.

Компьютерные сети. В настоящее время существенной тенденцией в информатизации общества является переход от использования компьютеров в автономном режиме к использованию их в информационных сетях.

Информационные сети создают реальную возможность быстрого и удобного доступа пользователя ко всей информации, накопленной человечеством за свою историю. Электронная почта и телеконференции, поиск информации во Всемирной паутине и в файловых архивах, интерактивное общение, прослушивание радиостанций и просмотр телевизионных программ, покупки в Интернет-магазинах

стали повседневной практикой многих пользователей компьютеров в развитых странах.

Количество постоянных пользователей ресурсов и услуг Интернета во всех странах мира составляет примерно один миллиард человек. В России количество пользователей растет быстрыми темпами и в 2011 году составило примерно 59 миллионов человек.

Население, занятое в информационной сфере. По данным ООН, начиная с 90-х годов XX века, количество работников, занятых в информационной сфере (для которых обработка информации является основной производственной функцией), возросло примерно на 50%.

Компьютеры и информационные технологии интенсивно проникают и в сферу материального производства. Инженер, фермер, специалисты других традиционных профессий все чаще имеют на своем рабочем месте компьютер и используют информационные и коммуникационные технологии в своей профессиональной деятельности.

С развитием коммуникационных технологий и мобильной связи всё больше людей осуществляют свою производственную деятельность дистанционно, т. е. работая дома, а не в офисе. Всё большее распространение получает дистанционное образование и поиск работы через Интернет. Оборот мирового рынка информационных и коммуникационных технологий составляет триллионы долларов. При этом на закупку аппаратных средств тратят менее половины полученной суммы денег, бóльшая часть вкладывается в разработку программного обеспечения, проектирование компьютерных сетей и т. д.

Информационное общество — это общество, в котором бóльшая часть населения занята получением, переработкой, передачей и хранением информации.

Курс информатики играет особую роль в эпоху перехода от индустриального общества к информационному, так как готовит выпускников школы к жизни и деятельности в информационном обществе.

Контрольные вопросы

1. Какую роль играли вещество, энергия и информация на различных этапах развития общества? Подготовьте доклад.

2. По каким основным параметрам можно судить о степени развитости информационного общества и почему?

3. Опишите изменения, происходящие в жизни и деятельности людей в процессе перехода от индустриального общества к информационному.

Задания для самостоятельного выполнения

4.1. Найдите в Интернете данные о росте количества его пользователей и серверов.

4.2. Информационная культура

Количество информации в современном обществе стремительно нарастает, человек оказывается погруженным в море информации. Для того чтобы в этом море «не утонуть», необходимо обладать **информационной культурой**, т. е. знаниями и умениями в области информационных и коммуникационных технологий, а также быть знакомым с юридическими и этическими нормами в этой сфере.

Процесс информатизации общества меняет традиционные взгляды на перечень умений и навыков, необходимых для социальной адаптации. Возьмем традиционный навык письма. На заре цивилизации (Шумеры, Египет), в античном мире (Эллада, Римская империя и др.) и в средние века (до изобретения книгопечатания) навык каллиграфического письма был залогом успешного продвижения по социальной лестнице. В индустриальном обществе (до изобретения персональных компьютеров) навыки письма ручкой также были необходимы для любого члена общества.

В настоящее время, на пороге информационного общества, социальная значимость навыка письма ручкой снижается, и наоборот, социальная значимость навыков ввода информации с помощью клавиатуры и работы с графическим интерфейсом приложений с помощью мыши возрастает.

Создание и редактирование документов с помощью компьютера, т. е. овладение **офисными информационными технологиями**, становятся в информационном обществе социально необходимым умением — достаточно просмотреть объявления о приеме на работу.

Современные информационные технологии позволяют включать в состав документа любые мультимедийные объекты (графику, звук, анимацию, видео). Дома вы можете привести в порядок фотоархив семьи, отсканировав старые фотографии и поместив их в упорядоченном виде в компьютерный фотоальбом, в процессе обучения

вы можете подготовить реферат с иллюстрациями, в процессе профессиональной деятельности — подготовить компьютерную презентацию о деятельности вашей фирмы. **Умение работать с мультимедиа документами, создавать компьютерные презентации** становится важным в информационном обществе.

В современном информационном обществе вряд ли необходимы навыки традиционного черчения на ватмане. Вместо этого полезно получить первоначальное представление о назначении и возможностях **компьютерных систем автоматизированного проектирования (САПР)**. Такие системы позволят вам быстро рассмотреть различные варианты планировки интерьера дома или квартиры, создать чертеж или схему.

Использование **электронных таблиц** сделает более простым и наглядным планирование и ведение домашнего бюджета, исследование и построение графиков функций в математике, автоматизирует заполнение многочисленных квитанций оплаты за квартиру и пр.

Необходимость упорядочить информацию, например, о людях, с которыми вы контактируете, требует использования записной книжки. Однако часто удобнее использовать для хранения такой информации компьютерную базу данных «Записная книжка». В информационном обществе очень полезным является умение создавать **базы данных**, а также вести в них поиск данных.

Квалифицированный пользователь компьютера может на основе использования **языков визуального объектно-ориентированного программирования** создавать необходимые ему специализированные приложения, создавать и исследовать модели различных объектов и процессов, разрабатывать динамические Web-сайты и т. д.

Современному человеку необходимо овладеть **коммуникативной культурой**, т. е. умениями создавать и посылать электронные письма, находить нужную информацию во Всемирной паутине или в файловых архивах, участвовать в чатах и т. д. Необходимым условием успешной профессиональной деятельности становится создание и публикация в Интернете Web-сайтов с информацией о деятельности организации или предприятия.

Информационная культура состоит не только в овладении определенным комплексом знаний и умений в области информационных и коммуникационных технологий, но и предполагает **знание и соблюдение юридических и этических норм и правил**. Законы запрещают использование пиратского компьютерного обеспечения и пропаганду насилия, наркотиков и порнографии в Интернете. Общение с помощью электронной почты или в чатах, участие в телеконференциях предполагают соблюдение определенных правил:

отвечать на письма и не рассылать знакомым и незнакомым людям многочисленные рекламные сообщения (спам), не отклоняться от темы обсуждения в телеконференциях и чатах и т. д.

Контрольные вопросы

1. Опишите основные компоненты информационной культуры, которые необходимы человеку для жизни в информационном обществе. Подготовьте сообщение.

4.3. Правовая охрана программ и данных. Защита информации

4.3.1. Правовая охрана информации

Правовая охрана программ и баз данных. Охрана интеллектуальных прав, а также прав собственности распространяется на все виды программ для компьютера, которые могут быть выражены на любом языке и в любой форме, включая исходный текст на языке программирования и машинный код. Однако правовая охрана не распространяется на идеи и принципы, лежащие в основе программы, в том числе на идеи и принципы организации интерфейса и алгоритма.

Для признания авторского права на программу для компьютера не требуется ее регистрации в какой-либо организации. Авторское право на программу возникает автоматически при ее создании. Для оповещения о своих правах разработчик программы может, начиная с первого выпуска в свет программы, использовать знак охраны авторского права, состоящий из трех элементов:

- буквы С в окружности © или круглых скобках (с);
- наименования (имени) правообладателя;
- года первого выпуска программы в свет.

Автору программы принадлежит исключительное право осуществлять воспроизведение и распространение программы любыми способами, а также выполнять модификацию программы. Организация или пользователь, правомерно владеющие экземпляром программы (купившие лицензию на ее использование), могут осуществлять любые действия, связанные с функционированием программы, в том числе ее запись и хранение в памяти компьютера.

Необходимо знать и выполнять существующие законы, запрещающие нелегальное копирование и использование лицензионно-

го программного обеспечения. В отношении организаций или пользователей, которые нарушают авторские права, разработчик может потребовать через суд возмещения причиненных убытков и выплаты нарушителем компенсации.

Электронная подпись. Электронная цифровая подпись в электронном документе признается юридически равнозначной подписи в документе на бумажном носителе.

В 2002 году был принят Закон «Об электронно-цифровой подписи», который стал законодательной основой электронного документооборота в России.

При регистрации электронно-цифровой подписи в специализированных центрах корреспондент получает два ключа: **секретный** и **открытый**. Секретный ключ хранится на flash-диске или смарт-карте и должен быть известен только самому корреспонденту. Открытый ключ должен быть у всех потенциальных получателей документов и обычно рассылается по электронной почте.

Процесс электронного подписания документа состоит в обработке с помощью секретного ключа текста сообщения. Далее зашифрованное сообщение посылается по электронной почте абоненту. Для проверки подлинности сообщения и электронной подписи абонент использует открытый ключ.

Контрольные вопросы

1. Как можно зафиксировать свое авторское право на программу?

Задания для самостоятельного выполнения

4.2. *Практическое задание.* Ознакомьтесь с законом «Об электронной подписи» и частью 4 Гражданского кодекса Российской Федерации.

4.3.2. Лицензионные, условно бесплатные и свободно распространяемые программы

Программы по их правовому статусу можно разделить на три большие группы: лицензионные, условно бесплатные и свободно распространяемые программы.

Лицензионные программы. В соответствии с лицензионным соглашением разработчики программы гарантируют ее нормальное функционирование в определенной операционной системе и несут за это ответственность.

Лицензионные программы разработчики могут продавать пользователям в форме коробочных дистрибутивов (рис. 4.1). В коробке находятся CD или DVD, с которых производится установка программы на компьютеры пользователей, и руководство пользователя по работе с программой. Лицензионные программы можно также приобретать по Интернету с сайта разработчика.

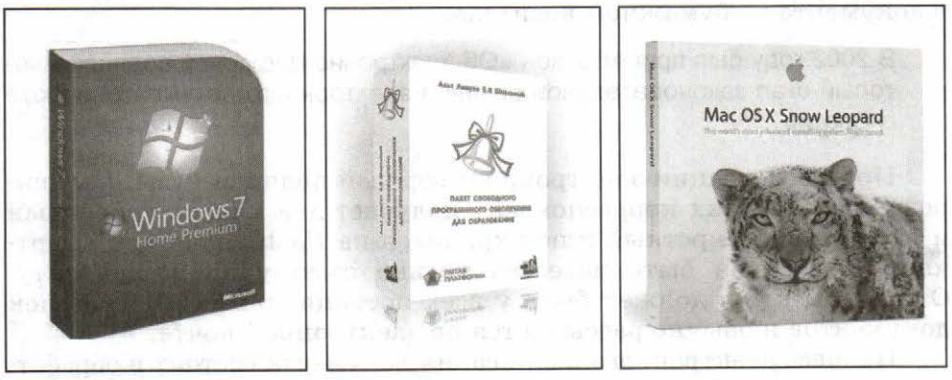


Рис. 4.1. Коробочные дистрибутивы операционных систем Windows 7, Alt Linux и Mac OS X Snow Leopard

Довольно часто разработчики предоставляют существенные скидки при покупке лицензий на использование программы на большом количестве компьютеров или на использование программы в учебных заведениях.

Условно бесплатные программы. Некоторые фирмы — разработчики программного обеспечения — предлагают пользователям условно бесплатные программы в целях их рекламы и продвижения на рынок. Пользователю предоставляется версия программы с ограниченным сроком действия (после истечения указанного срока программа перестает работать, если за нее не произведена оплата) или версия программы с ограниченными функциональными возможностями (в случае оплаты пользователю сообщается код, включающий все функции).

Свободно распространяемые программы. Многие производители программного обеспечения и компьютерного оборудования заинтересованы в широком бесплатном распространении программного обеспечения. К таким программным средствам можно отнести следующие:

- программы, поставляемые в учебные заведения в соответствии с государственными проектами;
- новые недоработанные (бета) версии программных продуктов (это позволяет провести их широкое тестирование);

- программные продукты, являющиеся частью принципиально новых технологий (это позволяет завоевать рынок);
- дополнения к ранее выпущенным программам, исправляющие найденные ошибки или расширяющие возможности;
- драйверы к новым или улучшенные драйверы к уже существующим устройствам.

Контрольные вопросы

1. В чем состоит различие между лицензионными, условно бесплатными и свободно распространяемыми программами?
2. Какие типы программ обычно распространяются бесплатно?

4.3.3. Защита информации

Защита от несанкционированного доступа к информации. Для защиты от несанкционированного доступа к данным, хранящимся на компьютере, используются **пароли**. Компьютер разрешает доступ к своим ресурсам только тем пользователям, которые зарегистрированы и ввели правильный пароль. Каждому конкретному пользователю может быть разрешен доступ только к определенным информационным ресурсам. При этом может производиться регистрация всех попыток несанкционированного доступа.

Защита с паролем используется при загрузке операционной системы (при загрузке системы пользователь должен ввести свой пароль). Вход по паролю может быть установлен в программе BIOS Setup; компьютер не начнет загрузку операционной системы, если не введен правильный пароль. Преодолеть такую защиту нелегко, более того, возникнут серьезные проблемы доступа к данным, если пользователь забудет этот пароль.

От несанкционированного доступа может быть защищен каждый диск, папка и файл локального компьютера. Для них могут быть установлены определенные права доступа (полный, только чтение, по паролю), причем права могут быть различными для различных пользователей.

В настоящее время для защиты от несанкционированного доступа к информации все чаще используются **биометрические системы идентификации.** Используемые в этих системах характеристики являются уникальными неотъемлемыми качествами личности человека и поэтому не могут быть утерянными и подделанными. К биометрическим системам защиты информации относятся системы идентификации **по отпечаткам пальцев, системы распознавания речи,** а также **системы идентификации по радужной оболочке глаза.**

Защита программ от нелегального копирования и использования. Компьютерные пираты, нелегально тиражируя программное обеспечение, обесценивают труд программистов, делают разработку программ экономически невыгодным бизнесом. Кроме того, компьютерные пираты нередко предлагают пользователям недоработанные программы, программы с ошибками или демонстрационные версии программ.

Для того чтобы программное обеспечение компьютера могло функционировать, оно должно быть установлено (инсталлировано). Программное обеспечение распространяется фирмами-производителями в форме **дистрибутивов** на CD или DVD, а также в виде дистрибутивов на сайте разработчика. Каждый дистрибутив имеет свой серийный номер, что препятствует незаконному копированию и установке программ. Серийный номер может быть получен в электронном виде или на бумажном носителе.

Для предотвращения нелегального копирования программ и данных, хранящихся на оптических дисках, может использоваться специальная защита. На CD или DVD может быть размещен закодированный **программный ключ**, который теряется при копировании и без которого программа не может быть установлена.

Защита от нелегального использования программ может быть реализована с помощью **аппаратного ключа**, который присоединяется обычно к порту компьютера. Защищаемая программа обращается к порту и запрашивает секретный код; если аппаратный ключ к компьютеру не присоединен, то защищаемая программа определяет ситуацию нарушения защиты и прекращает свое выполнение.

Физическая защита данных на дисках. Для обеспечения большей надежности хранения данных на жестких дисках используются RAID-массивы (Redundant Arrays of Independent Disks — избыточный массив независимых дисков). Несколько жестких дисков подключаются к **RAID-контроллеру**, который рассматривает их как единый логический носитель информации. При записи информации она дублируется и сохраняется на нескольких дисках одновременно, поэтому при выходе из строя одного из дисков данные не теряются.

Защита информации в Интернете. Если компьютер подключен к Интернету, то в принципе любой злоумышленник, также подключенный к Интернету, может получить доступ к информационным ресурсам этого компьютера. Если сервер, имеющий соединение с Интернетом, одновременно является сервером локальной сети, то возможно несанкционированное проникновение из Интернета в локальную сеть.

Для доступа к данным на компьютере, подключенном к Интернету, часто используется особо опасная разновидность компьютерных вирусов — **троянцы**. Троянцы распространяются по компьютерным сетям и встраиваются в операционную систему компьютера. В течение долгого времени они могут незаметно для

пользователя пересылать важные данные (пароли доступа к Интернету, номера банковских карточек и т. д.) злоумышленнику.

Такие компьютерные вирусы были названы троянцами по аналогии с троянским конем. В поэме Гомера описана осада древними греками города Троя (около 1250 года до н. э.). Греки построили громадного коня, поместили в нем воинов и оставили его около ворот города в качестве дара. Ничего не подозревающие троянцы втащили коня в город, а ночью греки вышли из коня и захватили город.

Для защиты от троянцев и других компьютерных вирусов используются антивирусные программы.

Большую опасность для серверов Интернета представляют **хакерские атаки**. Пример хакерской атаки: на определенный сервер Интернета посылаются многочисленные запросы со многих Интернет-адресов, что может привести к перегрузке сервера запросами пользователей, в результате чего сервер перестает отвечать на запросы.

Для защиты компьютера, подключенного к Интернету, от сетевых вирусов и хакерских атак между Интернетом и компьютером устанавливается аппаратный или программный **межсетевой экран**. Межсетевой экран отслеживает передачу данных между Интернетом и локальным компьютером, выявляет подозрительные действия и предотвращает несанкционированный доступ к данным.

Контрольные вопросы

1. Какие способы идентификации личности используются при предоставлении доступа к информации?
2. Объясните, почему компьютерное пиратство наносит ущерб обществу.
3. Чем отличается копирование файлов от инсталляции программ? Для чего каждый дистрибутив имеет серийный номер?
4. Какие существуют программные и аппаратные способы защиты информации?

Для заметок