

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ
Федеральное государственное бюджетное
образовательное учреждение
высшего профессионального образования
«Пензенский государственный университет» (ПГУ)

Р. А. Малеев, В. В. Регода, О. Н. Регода

Компьютерные технологии

*Рекомендовано федеральным государственным бюджетным
образовательным учреждением высшего профессионального образования
«Национальный исследовательский университет «МЭИ»
в качестве учебного пособия для студентов высших учебных заведений,
обучающихся по направлению подготовки 140400
«Электроэнергетика и электротехника»
(регистрационный номер рецензии 2518 от 25.09.2013)*

Пенза
Издательство ПГУ
2014

УДК 001.8 (083.95)

М18

Р е ц е н з е н т ы:

доктор технических наук, профессор кафедры
«Тракторы, автомобили и теплоэнергетика»
Пензенской государственной сельскохозяйственной академии
С. В. Тимохин;

кандидат технических наук, доцент,
заведующий кафедрой «Прикладная информатика»
Пензенского государственного технологического университета
А. П. Ремонтов

Малеев, Р. А.

М18 Компьютерные технологии : учеб. пособие / Р. А. Малеев, В. В. Регеда, О. Н. Регеда. – Пенза : Изд-во ПГУ, 2014. – 230 с.

ISBN 978-5-94170-837-6

Изложены виды алгоритмических процессов, а также основные правила выполнения схем алгоритмов по ГОСТ 19.701–90. Приводятся примеры задач на составление схем алгоритмов. Рассмотрены основные возможности языка программирования VBA для Word и Excel по созданию автоматизированных приложений по созданию и редактированию объектов, а также по применению эффектов в среде CorelDraw, основные приемы решения возникающих в инженерной практике математических задач средствами пакета Mathcad.

Издание подготовлено на кафедре «Электротехника и транспортное электрооборудование» Пензенского государственного университета и предназначено для лабораторных и индивидуальных занятий студентов, обучающихся по направлению 140400 «Электроэнергетика и электротехника», изучающих курс «Компьютерные технологии».

УДК 001.8 (083.95)

ISBN978-5-94170-837-6

© Пензенский государственный
университет, 2014

Введение

Формирование профессиональных компетенций, позволяющих применять компьютерные технологии в различных областях профессиональной деятельности, является одним из основных требований ФГОС для бакалавров по направлению 13.03.02 «Электроэнергетика и электротехника».

Важную роль в реализации поставленной задачи играет дисциплина «Компьютерные технологии», позволяющая обеспечить изучение студентами основ алгоритмизации и программирования, а также воспитание у них готовности использовать современные средства компьютерной графики и автоматизации инженерных расчетов в своей предметной области.

Решение многих инженерных задач невозможно без их алгоритмизации, т.е. предварительного представления последовательных шагов-подзадач. Поэтому освоение методов построения алгоритмов и их формализация, необходимая для решения инженерных задач на современных программно-аппаратных системах, являются важными этапами в подготовке студента к последующей профессиональной деятельности.

Одним из наиболее широко распространенных универсальных средств программного обеспечения является пакет программ Microsoft Office. Естественно, эта пользовательская среда была выбрана для изучения студентами в курсе «Компьютерные технологии». При необходимости создания новых надстроек в этих программах самым простым, быстрым в разработке и экономичным средством является встроенный в них редактор Visual Basic for Application (VBA). Кроме того, в большинстве программ Microsoft Office есть средство для записи макросов (макрорекордеров), которое позволяет автоматически создавать программы, обладая минимальными навыками в программировании.

VBA – значительно облегченная версия языка программирования Visual Basic (VB), встроенная в оболочку Microsoft Office (Excel, Word, Access, PowerPoint и т.д.), а также во многие другие программные пакеты, такие как CorelDRAW, WordPerfect и т.д. VBA расширяет функциональные возможности этих приложений и заменяет ранее использовавшиеся специализированные макроязыки, такие как WordBasic. VBA является интерпретируемым языком и он близок к Visual Basic, но может выполняться лишь в рамках приложения, в которое встроен. VBA может использоваться для

управления одним приложением из другого. В настоящее время появился и другой инструментарий расширения функциональности офисных приложений Microsoft – это Visual Studio Tools for Applications. Однако он значительно сложнее, чем VBA, поэтому компания Microsoft в своих новых продуктах оставляет возможность использования и VBA [1].

Для обработки изображений на компьютере используются специальные программы – графические редакторы. Векторные графические изображения являются оптимальным средством для хранения высокоточных графических объектов (чертежей, схем и т.д.). Достоинством векторной графики является также сравнительно небольшой объем получаемых файлов и возможность масштабирования без потери качества. На протяжении многих лет одним из популярных векторных графических редакторов является программа CorelDraw.

Общепризнанный лидер для автоматизации, документирования и повторного использования инженерных вычислений в процессе разработки изделия – это система Mathcad. Пакет Mathcad сочетает в себе мощнейший расчетный блок, полноценный текстовый редактор, графический модуль, экспертную документацию и контекстно-зависимый Help. К его достоинствам можно отнести высокую универсальность, полное соответствие используемых в пакете функций и операторов традициям оформления в математике, высокую степень интеграции с другими Windows-приложениями и минимальные требования к вычислительным и системным ресурсам. В данном учебном пособии изложены основы работы с продуктом корпорации PTC Mathcad Prime 2.0, который относится к новому поколению передового программного обеспечения для инженерных расчетов. В нем сочетаются уникальное удобство использования, обновленная документация, традиционный интерфейс Mathcad, широкие функциональные возможности и возможности управления единицами измерения.

1. Алгоритм и его свойства

1.1. Общие требования к алгоритмам

Алгоритмом называется точное предписание, которое определяет процесс, ведущий от исходных данных к требуемому конечному результату.

Можно дать и другие определения алгоритма [2]:

- 1) описание последовательности действий для решения задачи или достижения поставленной цели;
- 2) правила выполнения основных операций обработки данных;
- 3) описание вычислений по математическим формулам.

Алгоритмами, например, являются правила сложения, умножения, решения алгебраических уравнений, умножения матриц и т.п. Слово «алгоритм» происходит от «algoritmi», являющегося латинской транслитерацией арабского имени хорезмийского математика IX в. аль-Хорезми. Благодаря латинскому переводу трактата аль-Хорезми европейцы в XII в. познакомились с позиционной системой счисления, и в средневековой Европе алгоритмом стали называть десятичную позиционную систему счисления и правила счета в ней.

Применительно к ЭВМ алгоритм определяет *вычислительный процесс*, заключающийся в обработке некоторой совокупности возможных исходных данных (числовых, символьных, графических или звуковых и т.д.), направленный на получение соответствующих исходным данным результатов.

Если вычислительный процесс заканчивается получением результатов, то говорят, что соответствующий алгоритм применим к рассматриваемой совокупности исходных данных. В противном случае говорят, что алгоритм неприменим к совокупности исходных данных. Любой применимый алгоритм обладает следующими основными свойствами:

- результативностью;
- определенностью;
- массовостью.

Результативность означает возможность получения результата после выполнения конечного количества операций.

Определенность состоит в совпадении получаемых результатов независимо от пользователя и применяемых технических средств.

Массовость заключается в возможности применения алгоритма к целому классу однотипных задач, различающихся конкретными значениями исходных данных.

Для задания алгоритма необходимо описать следующие его элементы:

- набор объектов, составляющих совокупность возможных исходных данных, промежуточных и конечных результатов;
- правило начала;
- правило непосредственной переработки информации (описание последовательности действий);
- правило окончания;
- правило извлечения результатов.

Алгоритм всегда рассчитан на конкретного исполнителя. В настоящее время традиционным исполнителем является персональный компьютер (ПК). Для обеспечения возможности реализации на ПК алгоритм должен быть описан на языке, понятном компьютеру, т.е. на языке программирования.

Таким образом, можно дать следующее определение *программы для ПК* – это описание алгоритма и данных на некотором языке программирования, предназначенное для последующего автоматического выполнения.

1.2. Способы описания алгоритмов

На практике наиболее распространены следующие формы представления алгоритмов:

- словесно-формульная (записи на естественном языке);
- графическая, или схемная (изображения с помощью графических символов);
- с помощью псевдокодов (полуформализованные описания алгоритмов на условном алгоритмическом языке, включающие в себя как элементы языка программирования, так и фразы естественного языка, общепринятые математические обозначения и др.);
- программная (тексты на языках программирования).

Перед составлением программ чаще всего используются словесно-формульный и схемный способы. Иногда перед составлением программ на низкоуровневых языках программирования типа языка

Ассемблер алгоритм программы записывают, пользуясь конструкциями некоторого высокоуровневого языка программирования. Схемное описание алгоритмов функционирования удобно использовать наряду с программным для сложных программных систем.

Словесный способ записи алгоритмов представляет собой последовательное описание этапов обработки данных. Алгоритм задается в произвольном изложении на естественном языке.

Например, словесный алгоритм определения четности числа n может быть следующим:

- задать число n ;
- разделить число n на 2;
- если остаток от деления равен 0, то число n – четное, иначе – нечетное.

Описанный алгоритм применим к любым натуральным числам и должен приводить к решению поставленной задачи.

Словесный способ неудобен, так как его трудно формализовать, и, кроме того, он допускает неоднозначность толкования отдельных предписаний.

При *схемном* описании алгоритм изображается отдельными геометрическими фигурами (символами), связанными между собой с помощью линий потоков с однозначно заданным направлением со стрелками. Внутри символов записывается выполняемая ими последовательность действий.

Данный способ по сравнению с другими способами записи алгоритма имеет ряд преимуществ. Он наиболее нагляден: каждая операция вычислительного процесса изображается отдельной геометрической фигурой. Кроме того, графическое изображение алгоритма наглядно показывает разветвления путей решения задачи в зависимости от различных условий, повторение отдельных этапов вычислительного процесса и другие детали.

Псевдокод представляет собой систему обозначений и правил, цель которой – единообразная запись алгоритмов. Он занимает промежуточное место между естественным и формальным языками. В псевдокоде используются некоторые формальные конструкции и математическая символика, что приближает запись алгоритма к общепринятой математической записи. В псевдокоде не приняты строгие синтаксические правила для записи команд, присущие формальным языкам, что облегчает запись алгоритма на стадии его проектирования и дает возможность использовать более широкий набор команд, рассчитанный на абстрактного исполнителя. Однако в псев-

докоде обычно имеются некоторые конструкции, присущие формальным языкам, что облегчает переход от записи на псевдокоде к записи алгоритма на формальном языке.

Например, псевдокод циклического алгоритма для определения произведения чисел от одного до пяти может быть записан следующим образом:

$p = 1, k = 1$

пока $k \leq 5$

нц

$p = p * k$

$k = k + 1$

кц

П р и м е ч а н и е. Приведенный алгоритм обозначает, что пока выполняется условие, стоящее после ключевого слова «пока», будет выполняться тело цикла, расположенное между ключевыми словами «нц» (начало цикла) и «кц» (конец цикла).

1.3. Основные правила выполнения схем алгоритма

Графическое представление алгоритма называют *схемой алгоритма* – последовательность связанных между собой функциональных блоков, каждый из которых соответствует выполнению одного или нескольких действий.

В схеме алгоритмов каждому типу действий (вводу исходных данных, вычислению значений выражений, проверке условий, управлению повторением действий, окончанию обработки и т.п.) соответствует геометрическая фигура, представленная в виде *символа*. Символы соединяются линиями потоков данных или управления, определяющими очередность выполнения действий.

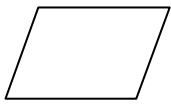
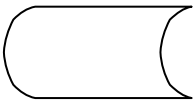
Правила выполнения схем алгоритмов устанавливаются ГОСТ 19.701–90 «Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения» [3]. Они строятся из имеющих заданное значение символов, краткого пояснительного текста и соединительных линий. Схемы алгоритмов могут использоваться на разных уровнях детализации, причем число уровней зависит от размеров и сложности задачи обработки данных.

В ГОСТ 19.701–90 определены символы, предназначенные для использования в документации по обработке данных, и приведено руководство по условным обозначениям для применения их в схе-

мах данных, в схемах программ, в схемах работы системы, в схемах взаимодействия программ и в схемах ресурсов системы.

В табл. 1.1 приведены основные символы данных, используемые на схемах алгоритма [3].

Таблица 1.1

Название символа	Обозначение	Назначение символа
Данные		Отображает данные; носитель данных не определен
Запоминаемые данные		Отображает данные, хранящиеся в виде, пригодном для обработки; носитель данных не определен
Документ		Отображает данные, представленные на носителе в удобочитаемой форме (распечатка принтера, документ для оптического или магнитного считывания и т.п.)
Ручной ввод		Отображает данные, вводимые вручную во время обработки с устройств любого типа
Дисплей		Отображает данные, представленные в удобочитаемой форме на носителе в виде отображающего устройства (экран для визуального наблюдения и т.п.)

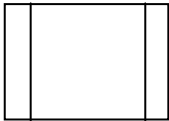
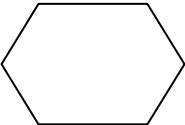
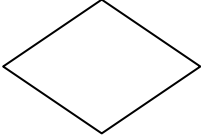
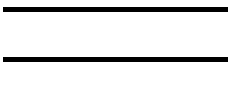
Операции обработки данных и носители информации изображаются на схеме соответствующими *символами*.

В ГОСТ 19.701–90 представляются рекомендуемые формы символов на схеме алгоритма, а также указывается, что не должны меняться углы и другие параметры, влияющие на соответствующую форму символов. По возможности символы должны быть одного размера.

В сложившейся практике дополнительно установились габариты символов. Большая часть символов по построению условно вписана в прямоугольник со сторонами a и b . Минимальное значение a должно быть не менее 10 мм, увеличение a производится на число, кратное 5 мм. Размер b рекомендуется устанавливать равным $1,5 a$. Для отдельных блоков допускается соотношение между a и b , равное 1 : 2.

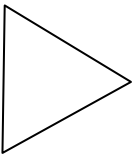
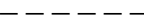
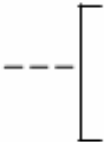
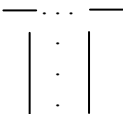
В табл. 1.2 приведены символы процессов, используемые на схемах алгоритма [3].

Таблица 1.2

Название символа	Обозначение	Назначение символа
Процесс		Отображает функцию обработки данных любого вида (выполнение определенной операции или группы операций, приводящее к изменению значения, формы или размещения информации)
Предопределенный процесс		Отображает предопределенный процесс, состоящий из одной или нескольких операций (шагов) программы, которые определены в подпрограмме, модуле и т.п.
Подготовка		Отображает модификацию команды (группы команд) для воздействия на последующую функцию (установка переключателя, модификация индексного регистра или инициализация программы)
Ручная операция		Отображает любой процесс, выполняемый человеком
Решение		Отображает решение или функцию переключательного типа, имеющую один вход и ряд альтернативных выходов, один и только один из которых может быть активизирован после вычисления условий, определенных внутри этого символа
Параллельные действия		Отображает синхронизацию двух или более параллельных операций
Граница цикла		Состоит из двух частей, отображает начало и конец цикла. Обе части символа имеют один и тот же идентификатор

Символы линий и специальные символы показаны в табл. 1.3 [3].

Таблица 1.3

Название символа	Обозначение	Назначение символа
Линия		Отображает поток данных или управления
Передача управления		Отображает непосредственную передачу управления от одного процесса к другому. Тип передачи управления должен быть назван внутри символа (запрос, вызов, событие)
Пунктирная линия		Отображает альтернативную связь между двумя или более символами. Кроме того, символ используют для обведения выбранного участка
Соединитель		Отображает выход в часть схемы и вход из другой части этой схемы и используется для обрыва линии и продолжения ее в другом месте
Терминатор		Отображает выход во внешнюю среду и вход из внешней среды (начало или конец схемы программы)
Комментарий		Используют для добавления комментариев. Пунктирные линии в символе комментария связаны с соответствующим символом или группой символов. Текст комментариев должен быть помещен около ограничивающей скобки
Пропуск		Используют в схемах для отображения пропуска символа или группы символов

Примеры применения некоторых символов линий и специальных символов приведены на рис. 1.1, а, б [3].

Символы в схеме должны быть расположены равномерно. Следует придерживаться разумной длины соединений и минимального числа длинных линий.

В пределах одной схемы рекомендуется изображать блоки одинаковых размеров. При возникновении необходимости описать схему алгоритма блоки нумеруются.

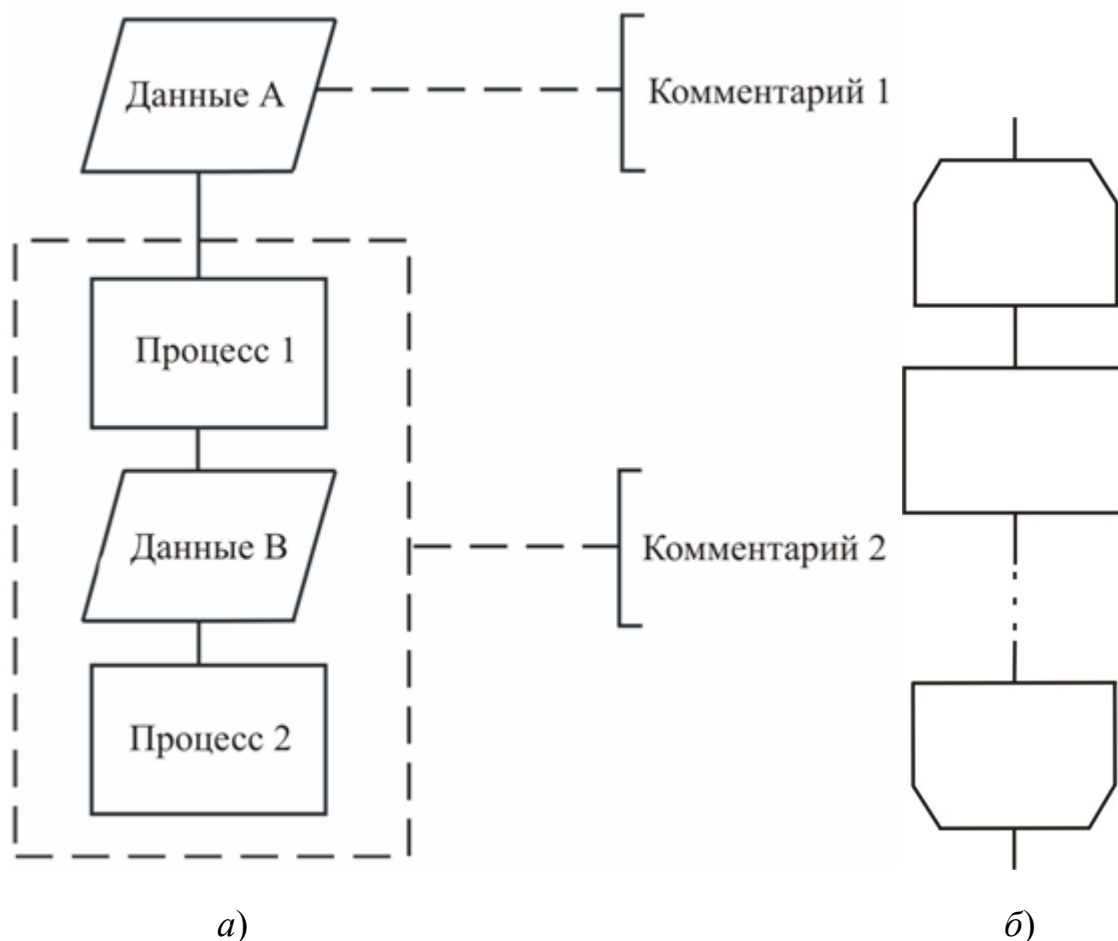


Рис. 1.1

Символы могут быть вычерчены в любой ориентации, но, по возможности, предпочтительной является горизонтальная ориентация.

Зеркальное изображение формы символа обозначает одну и ту же функцию, но не является предпочтительным.

Минимальное количество текста, необходимого для понимания функции данного символа, следует помещать внутри данного символа слева направо и сверху вниз независимо от направления потока. Если объем текста, помещаемого внутри символа, превышает его размеры, следует использовать символ комментария.

В схемах слева над символом может использоваться *идентификатор* символов, который определяет символ при использовании его в других элементах документации (например в листинге программы).

В схемах может использоваться описание символов – любая другая информация, например, для отображения специального при-

менения символа с перекрестной ссылкой или для улучшения понимания функции как части схемы. Описание символа должно быть расположено справа над символом.

Потоки данных или потоки управления в схемах показываются линиями. Их стандартное направление – слева направо и сверху вниз.

В случаях, когда необходимо внести большую ясность в схему, на линиях используются стрелки. Если поток имеет направление, отличное от стандартного направления, то это должно указываться стрелками.

В схемах следует избегать пересечения линий. Пересекающиеся линии не имеют логической связи между собой, поэтому изменения направления в точках пересечения не допускаются.

Две или более входящие линии могут объединяться в одну исходящую линию. Если две или более линии объединяются в одну линию, место объединения должно быть смещено (рис. 1.2).

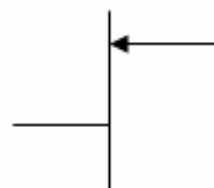


Рис. 1.2

Линии в схемах должны подходить к символу либо слева, либо сверху, а исходить либо справа, либо снизу. Линии должны быть направлены к центру символа.

При необходимости линии в схемах следует разрывать во избежание излишних пересечений или слишком длинных линий, а также, если схема состоит из нескольких страниц. Соединитель в начале разрыва называется внешним, а соединитель в конце разрыва – внутренним.

Ссылки к страницам могут быть приведены совместно с символом комментария для их соединителей (рис. 1.3, а, б).

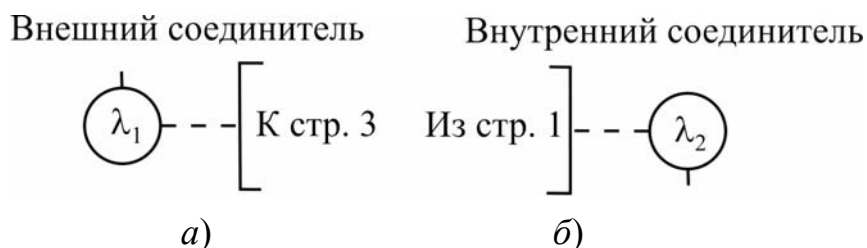


Рис. 1.3

Несколько выходов из символа следует показывать:

– несколькими линиями от данного символа к другим символам (рис. 1.4, а);

– одной линией от данного символа, которая затем разветвляется в соответствующее число линий (рис. 1.4,б).

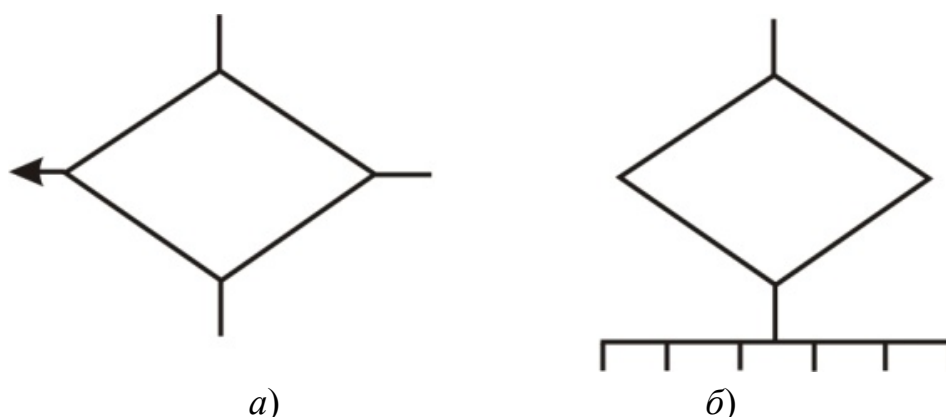


Рис. 1.4

Каждый выход из символа должен сопровождаться соответствующими значениями условий, чтобы показать логический путь, который он представляет.

Вместо одного символа с соответствующим текстом могут быть использованы несколько символов с перекрытием изображения от переднего (первого) к заднему (последнему), каждый из которых содержит описательный текст (использование или формирование нескольких носителей данных или файлов, производство множества копий печатных отчетов или форматов перфокарт). При этом линии могут входить или исходить из любой точки перекрытых символов.

1.4. Виды алгоритмических процессов

Одним из свойств алгоритма является дискретность – возможность разделения процесса вычислений, предписанного алгоритмом, на отдельные этапы, возможность выделения участков алгоритма с определенной структурой. Можно выделить и наглядно представить графически три простейшие структуры: последовательность двух или более операций, выбор направления, повторение.

Любой вычислительный процесс может быть представлен как комбинация этих элементарных алгоритмических структур. Вычислительные процессы, выполняемые на ЭВМ по заданной программе, можно разделить на три основных вида: *линейные, разветвляющиеся, циклические* [4].

1.4.1. Линейные алгоритмические процессы

Линейным принято называть вычислительный процесс, в котором операции выполняются последовательно, в порядке их записи. Каждая операция является самостоятельной, не зависимой от каких-либо условий. На схеме блоки, отображающие эти операции, располагаются в линейной последовательности. Линейные вычислительные процессы имеют место, например, при вычислении арифметических выражений, когда имеются конкретные числовые данные и над ними выполняются соответствующие условию задачи действия.

На рисунке 1.5 показан пример линейного алгоритма, определяющего процесс вычисления арифметического выражения

$$y = (b^2 - ac) : (a + c)$$

для различных значений переменных a , b и c .

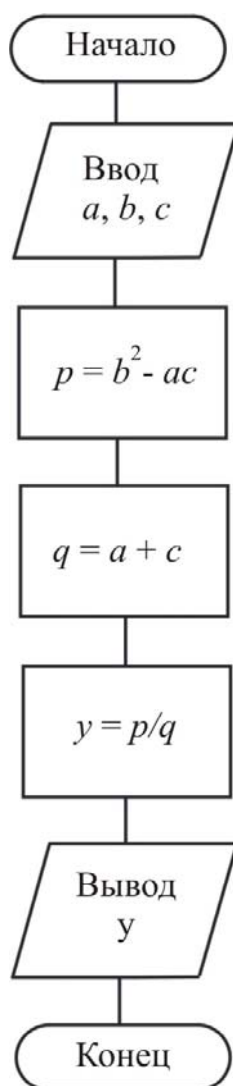


Рис. 1.5

1.4.2. Разветвляющиеся алгоритмические процессы

Вычислительный процесс называется *разветвляющимся*, если для его реализации предусмотрено несколько направлений (ветвей). Каждое конкретное направление процесса обработки данных является отдельной ветвью вычислений. Ветвление в программе – это выбор одной из нескольких последовательностей команд при выполнении программы. Выбор направления зависит от заранее определенного признака, который может относиться к исходным данным, к промежуточным или конечным результатам. Признак характеризует свойство данных и имеет два или более значения.

Разветвляющийся процесс, включающий в себя две ветви, называется простым, более двух ветвей – сложным. Сложный разветвляющийся процесс можно представить с помощью комбинации простых разветвляющихся процессов. Направление ветвления выбирается логической проверкой, в результате которой возможны два ответа: «да» («+») – условие выполнено и «нет» («–») – условие не выполнено.

Следует иметь в виду, что, хотя на схеме алгоритма должны быть показаны все возможные направления вычислений в зависимости от выполнения определенного условия (или условий), при однократном прохождении программы процесс реализуется только по одной ветви, а остальные исключаются. Любая ветвь, по которой осуществляются вычисления, должна приводить к завершению вычислительного процесса.

На рисунке 1.6 показан пример алгоритма с разветвлением для вычисления следующего выражения:

$$y = \begin{cases} (a + b), & \text{при } x \leq 0; \\ \frac{c}{b}, & \text{при } x > 0. \end{cases}$$

В зависимости от истинности или ложности условия, указанного в символе *решение*, значение функции вычисляется по одному из двух альтернативных выражений.

На рисунке 1.7 показан еще один пример алгоритма с разветвлением для вычисления значения ступенчатой функции:

$$y = \begin{cases} 0,5x^2 + 1 & \text{при } x \leq a; \\ \cos x & \text{при } a < x < b; \\ (\operatorname{tg} x)^2 & \text{при } x \geq b. \end{cases}$$

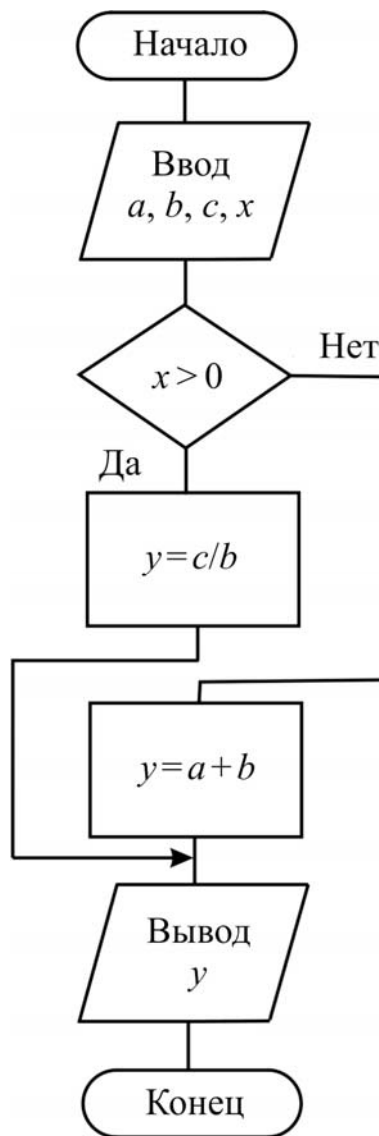


Рис. 1.6

В алгоритме после ввода любых значений x , a и b и проверки условия в символе *решение 3* значение функции z может вычисляться по одной из трех альтернативных ветвей: в символе *процесс 4* или 5, или 6. При этом, так как количество альтернативных ветвей больше двух, около соответствующего вывода символа решение 3 указывается условие перехода на соответствующую ветвь, а внутри самого символа *решение* указывается лишь имя переменной x .

Результаты вычисления значения функции по любой из альтернативных ветвей выводятся в символе *вывод-вывод 7*.

Рассмотрим алгоритм, определяющий принадлежность точки с координатами (X, Y) кругу радиуса R (рис. 1.8).

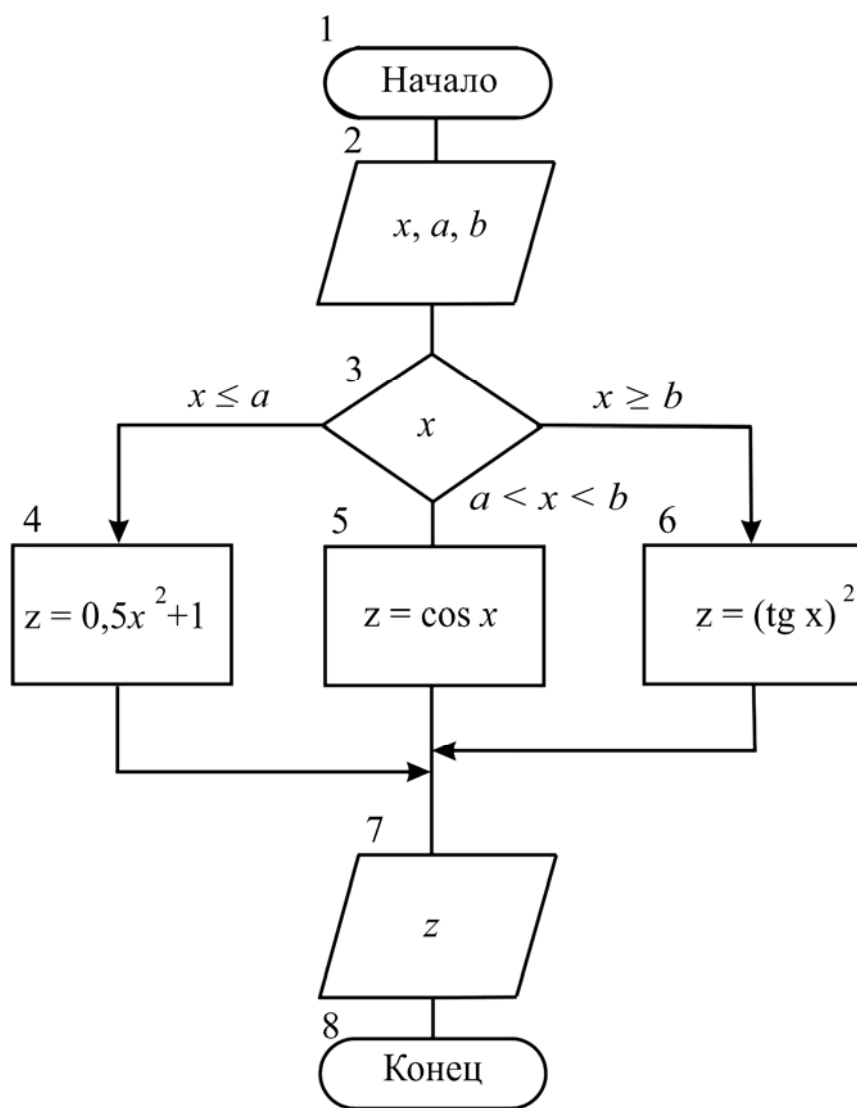


Рис. 1.7

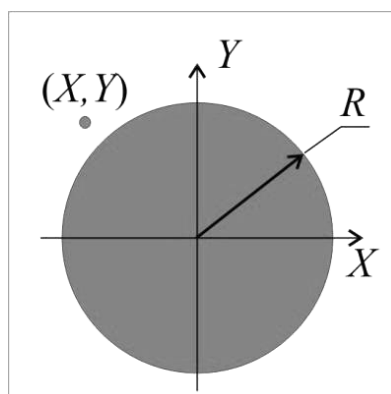


Рис. 1.8

На рис. 1.9 приведена соответствующая схема алгоритма.

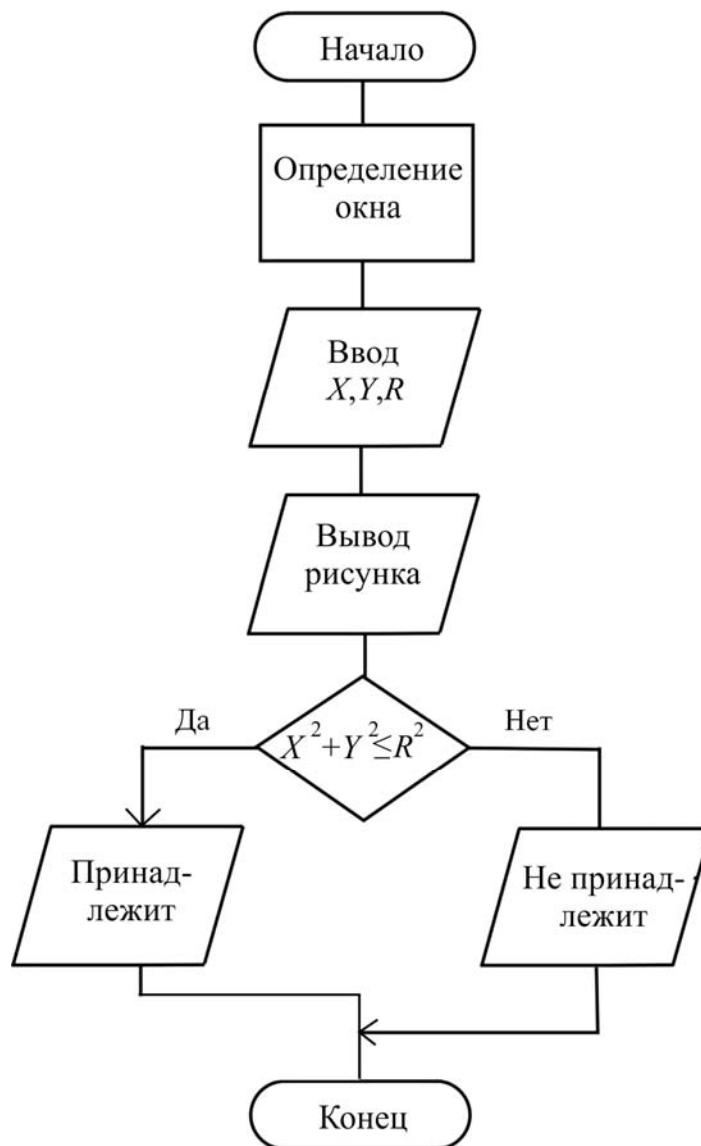


Рис. 1.9

Сначала в символе процесса *Определение окна* задаются размеры окна, в котором будет выводиться рисунок (см. рис. 1.8), например: $(-100, -100) - (100, 100)$.

Алгоритм предусматривает ввод любых значений X , Y , R и вывод рис. 1.8.

В символе *решение* проверяется выполнение условия

$$X^2 + Y^2 \leq R^2.$$

В случае истинности условия выводится сообщение «Принадлежит», иначе – «Не принадлежит».

Среди разветвляющихся алгоритмических процессов в отдельную группу можно выделить *переключательные* алгоритмические процессы.

Переключательные алгоритмические процессы позволяют выбрать один вариант из нескольких возможных вариантов. Их можно представить с помощью простых разветвляющихся процессов, но удобнее их представлять с помощью структуры *Выбор*.

На рис. 1.10 приведен переключательный алгоритм, выдающий словесное значение числа a .

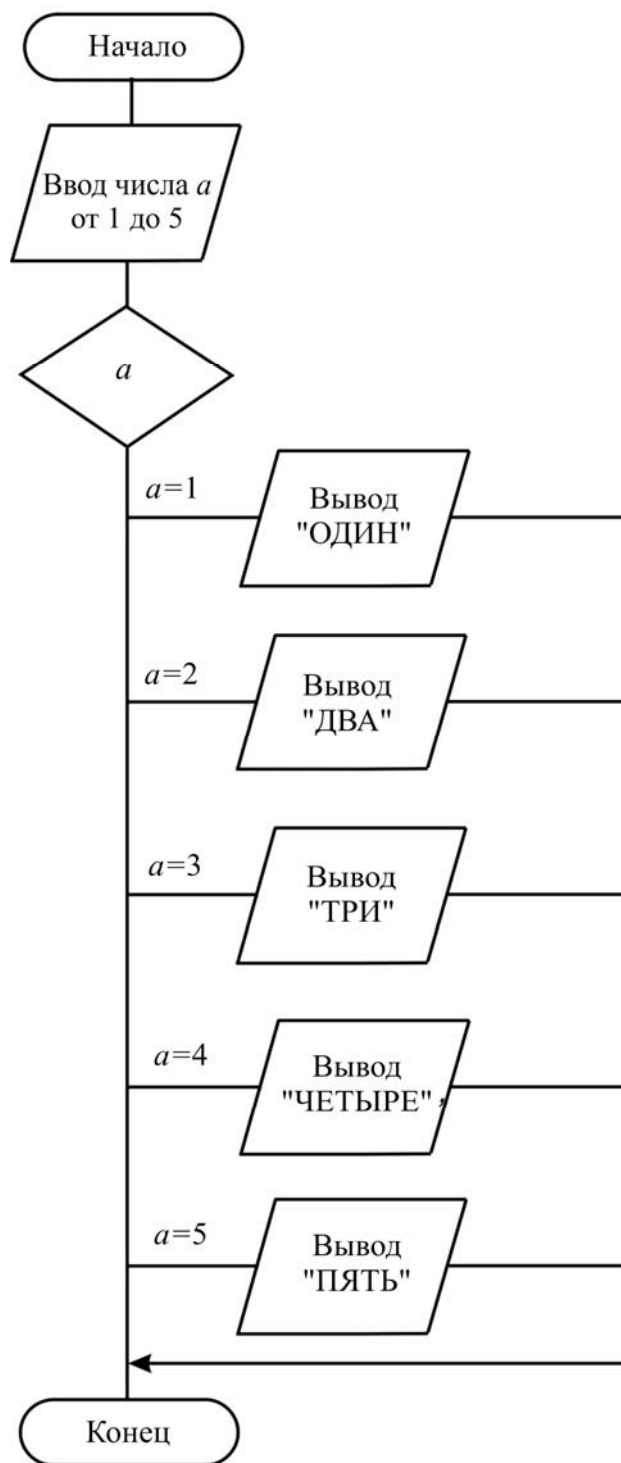


Рис. 1.10

В данном алгоритме вводится число a и сравнивается с числами от одного до пяти. Если a равно одному из этих чисел, то на экран выводится словесное значение числа.

Каждый из альтернативных выходов символа *решение* должен сопровождаться соответствующим условием, в случае выполнения одного из них алгоритм пойдет по соответствующей ветви.

Выход из любой альтернативной ветви алгоритма приводит к его завершению, при этом остальные условия не проверяются.

1.4.3. Циклические алгоритмические процессы

Циклическими называются алгоритмы, содержащие циклы.

Цикл – это многократно повторяемый участок алгоритма.

В организации цикла можно выделить следующие этапы:

- подготовка (инициализация) цикла (И);
- выполнение вычислений цикла (тело цикла) (Т);
- модификация параметров (М);
- проверка условия окончания цикла (У);
- продолжение (П).

На рис. 1.11, *а,б* показаны возможные варианты последовательности этих этапов.

На этапе инициализации присваиваются начальные значения переменной цикла X , тем самым обеспечивается выполнение тела цикла при первом выполнении.

Тело цикла в общем случае может содержать любое количество операций (в частности, может вообще отсутствовать).

На этапе модификации параметров в случае, если существует закономерность изменения значений переменной цикла, а также если эти значения используются строго последовательно, то к текущему значению переменной цикла прибавляется шаг ΔX с учетом его знака $X = X + \Delta X$.

Если не существует никакой закономерности изменения значений переменной цикла, то эти значения могут храниться в одномерном массиве в виде $X(i)$. В этом случае в качестве переменной цикла будет выступать i , модификация которой будет производиться путем изменения ее значения на единицу $i = i + 1$.

Проверка условия окончания цикла чаще всего предполагает сравнение текущего значения переменной цикла с конечным значением переменной цикла.

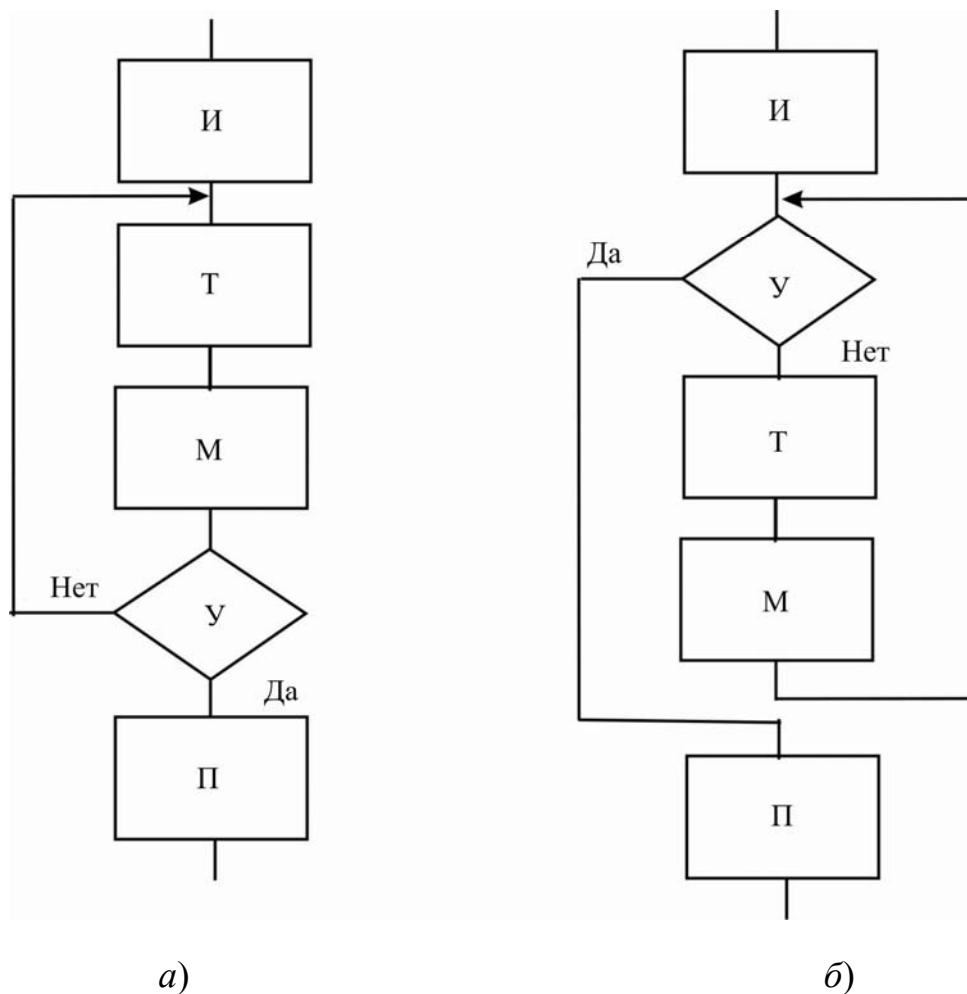


Рис. 1.11

В общем случае условие окончания цикла может быть явно не связано ни со значением переменной цикла, ни с ее конечным значением.

Для цикла с нижним окончанием (рис. 1.11,а) тело цикла выполняется как минимум один раз, так как сначала производятся вычисления, а затем проверяется условие выхода из цикла. В случае цикла с верхним окончанием (рис. 1.11,б) тело цикла может не выполниться ни разу в случае, если сразу соблюдается условие выхода.

После выхода из цикла алгоритм переходит на этап продолжения программы (этап П на рис. 1.11).

На рис. 1.12 показан пример циклического алгоритма вычисления суммы десяти чисел. Здесь в качестве переменной цикла используется переменная i с начальным значением, равным единице, и конечным значением, равным 10. Шаг изменения i равен единице.

В этом цикле проверка условия выхода из цикла выполняется в конце цикла. При этом тело цикла повторится десять раз.



Рис. 1.12

Для удобства на схеме алгоритма (см. рис. 1.12) с помощью символов *комментарий* поясняется, какие действия выполняются на каждом шаге алгоритма.

Цикл называется *детерминированным*, если число повторений тела цикла заранее известно или может быть определено из выражений. Цикл на рис. 1.13,а предписывает выполнять тело цикла для всех значений переменной цикла X в заданном диапазоне от 0 до 20 с шагом 2.

Инициализация выполняется в символе 1, а к телу цикла относятся символы 2 и 3 на схеме алгоритма. Модификация параметров выполняется в символе 4, а проверка условия окончания цикла — в символе 5.

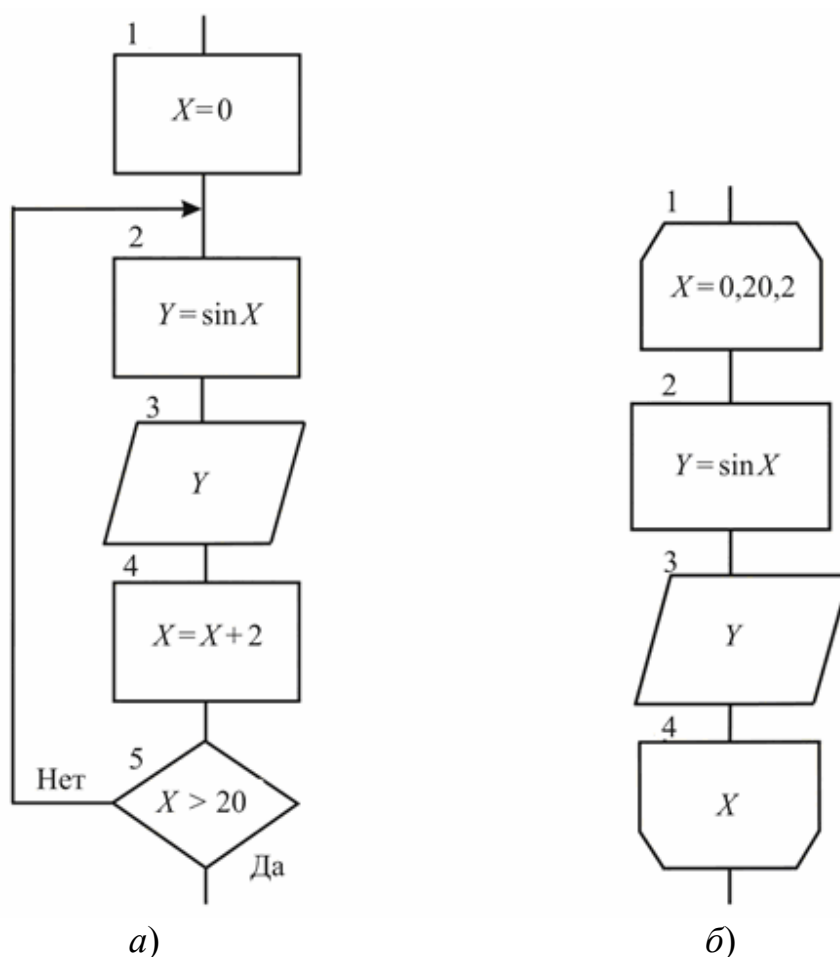


Рис. 1.13

Число повторений цикла, определяемое как отношение разности конечного (20) и начального (0) значений переменной цикла (X) к шагу (2) плюс один, в данном примере равно 11.

На рис. 1.13,б приведен другой вариант реализации этого же цикла с применением специальных символов начала и конца цикла, соответственно 1 и 4. В символе 1 задаются начальные и конечные значения переменной цикла X , а также шаг изменения X , равный двум. К телу цикла относятся символы 2 и 3 на рис. 1.13,б. Модификация параметров цикла и проверка условия окончания цикла выполняются в символе 4.

Вычисление суммы и произведения нескольких чисел, а также подсчет количества в большинстве языков программирования выполняется постепенно, путем накопления.

Сумма вычисляется по формуле

$$\text{сумма} = \text{сумма} + \text{слагаемое}.$$

Перед вычислениями начальному значению суммы нужно присвоить значение нуля.

Произведение вычисляется по формуле

произведение = произведение $\times$ сомножитель.

Перед вычислениями начальному значению произведения нужно присвоить значение единицы.

Подсчет количества выполняется по формуле, подобной формуле для вычисления суммы:

количество = количество + 1.

Начальному значению количества присваивается значение нуля.

Алгоритмы с накоплением часто применяются при расчете конечных рядов и произведений.

На рисунке 1.14, а, б приводятся алгоритмы вычисления конечного ряда $\sum_{i=2}^{100} \frac{i+1}{i+2}$ и конечного произведения $\prod_{i=1}^{25} \frac{i^2}{i^2+2i+3}$ соответственно.

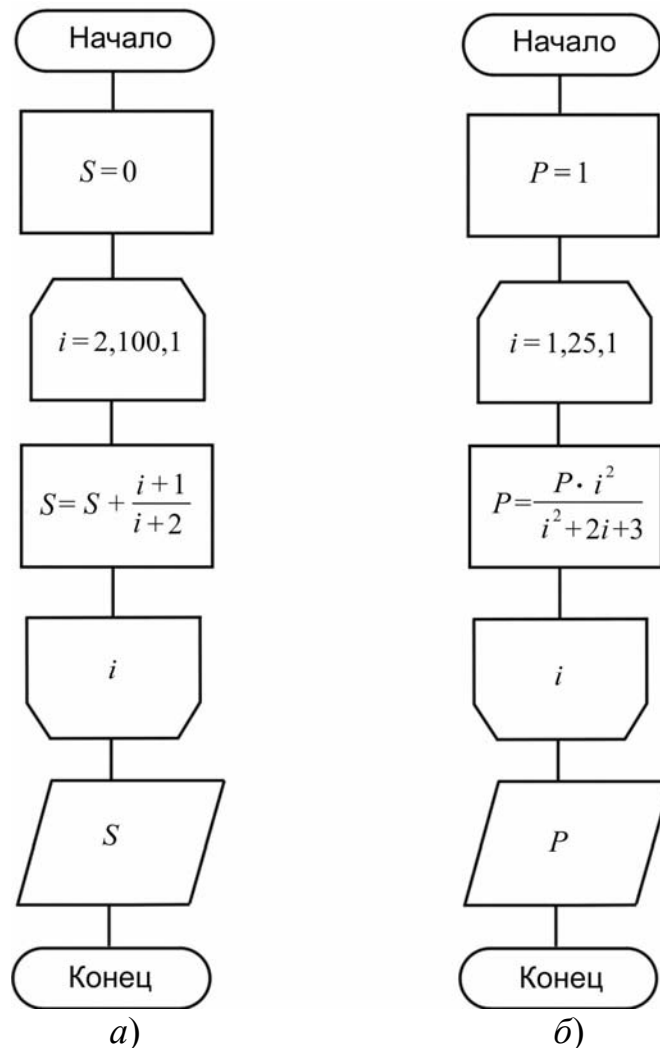


Рис. 1.14

1.4.4. Задачи по обработке одномерных массивов

Массив – это совокупность элементов, имеющих одинаковое имя и отличающихся номером (индексом) их расположения в последовательности.

Одномерный массив – последовательность чисел $A(1)$, $A(2)$, $A(3)$, ..., $A(n)$. Размерность массива равна количеству элементов в нем.

Пример 1

Дан одномерный массив $A(i)$ длиной 20, упорядоченный по убыванию. Переупорядочить массив $A(i)$ по возрастанию его элементов,

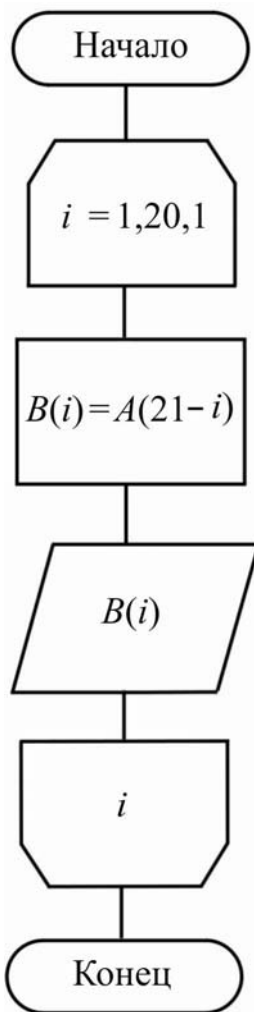


Рис. 1.15

используя первоначальную упорядоченность массива, и сохранить его значения в массиве $B(i)$. Вывести полученный массив $B(i)$ на дисплей.

Сначала рассмотрим словесный алгоритм решения задачи:

1) В цикле последовательно перебрать все элементы массива $B(i)$ (начиная с первого по двадцатый).

2) Присвоить им значения соответствующих элементов массива $A(21 - i)$ (начиная с конца).

3) Вывести полученное значение на дисплей.

4) Повторить тело цикла 20 раз.

На рис. 1.15 приводится соответствующая схема алгоритма.

Пример 2

Одномерный массив A длиной 20 заполнить случайными числами. Составить алгоритм определения первого минимального элемента массива A и определить его индекс.

Рассмотрим словесный алгоритм:

1. В первом цикле присвоить значения элементам массива $A(i)$.

2. Переменной $\min$ присвоить значение первого элемента массива A , а переменной N , в которой будет храниться значение индекса минимального элемента массива, присвоить значение 1.

3. Во втором цикле, который будет последовательно выбирать элементы массива $A(i)$, начиная со второго, сравнить значение те-

кущего элемента массива $A(i)$ со значением переменной $\min$, и если выполняется условие $A(i) < \min$, то на место $\min$ записать текущее значение элемента массива $A(i)$, а в переменной N запомнить соответствующий номер индекса.

4. Повторить тело цикла 20 раз.

5. Вывести полученные значения переменных $\min$ и N .

На рис. 1.16 приводится соответствующая схема алгоритма.

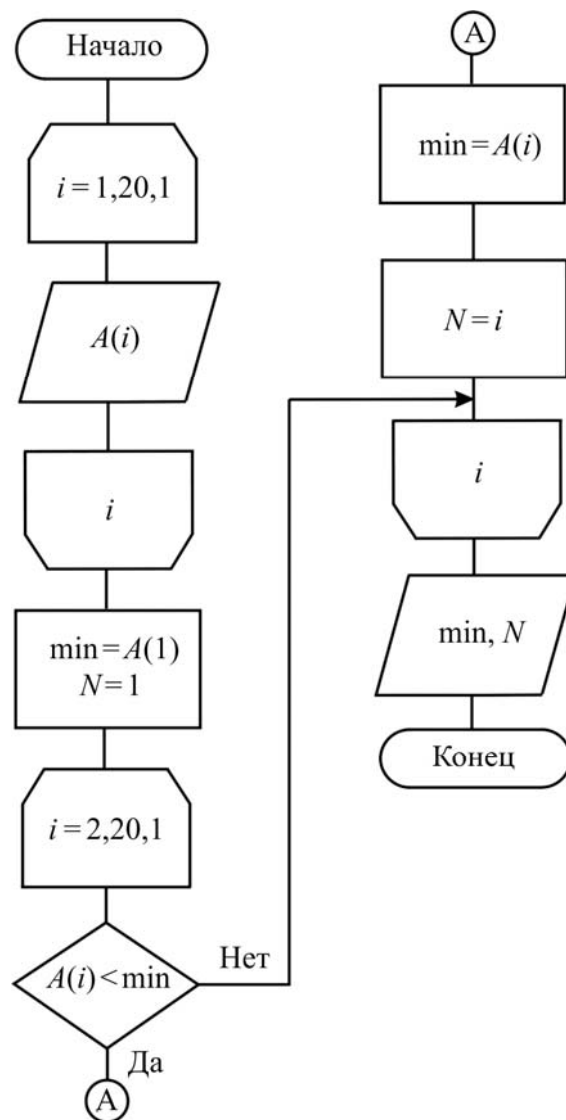


Рис. 1.16

Пример 3

Даны одномерные массивы A и B длиной 20, упорядоченные по возрастанию (убыванию). Сформировать из них общий одномерный массив C , упорядоченный по убыванию (возрастанию).

Рассмотрим словесный алгоритм:

1. Выбрать последние элементы массива $A(k)$ и $B(m)$.
2. Организовать цикл по переменной i с начальным значением, равным 1, конечным значением, равным 40, и с шагом, равным 1.
3. Сравнить между собой значения $A(k)$ и $B(m)$, и если $A(k) \geq B(m)$ ($A(k) \leq B(m)$), то текущему элементу $C(i)$ присвоить значение $A(k)$ и уменьшить значение k на 1, а если иначе, то текущему элементу $C(i)$ присвоить значение $B(m)$ и уменьшить значение m на 1.
4. Повторять тело цикла 40 раз.
5. Закончить алгоритм.

На рис. 1.17 приводится соответствующая схема алгоритма.

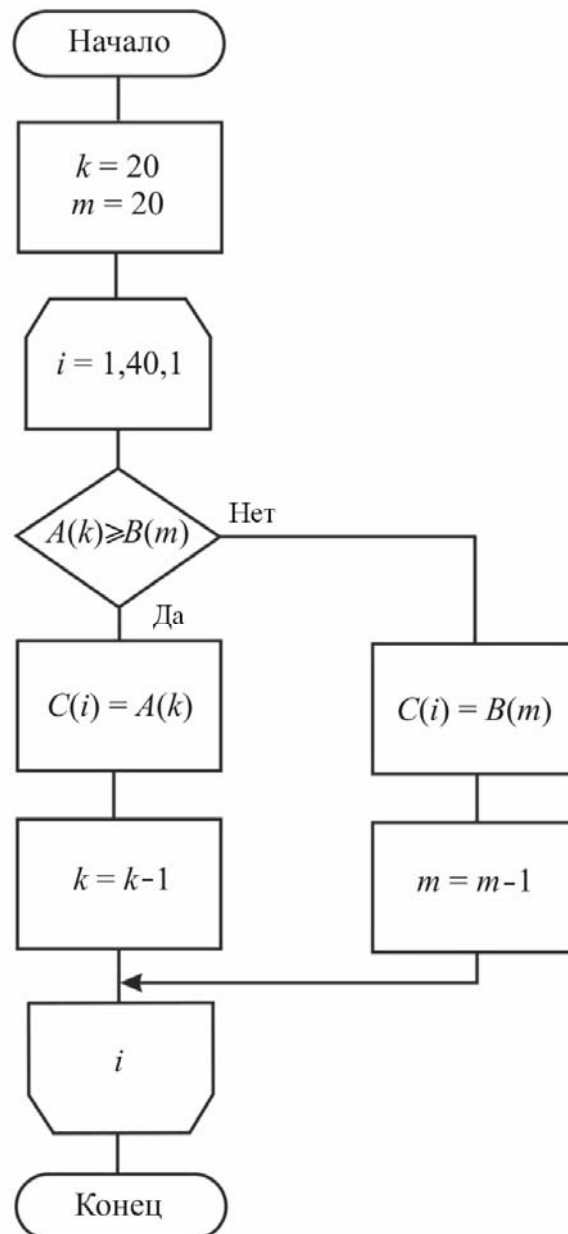


Рис. 1.17

Пример 4

Одномерный массив A длиной 30 заполнить случайными числами. Переместить нулевые элементы массива в конец, сдвинув остальные элементы влево, и вывести массив.

Рассмотрим словесный алгоритм:

1. В первом цикле присвоить значения элементам массива.
2. Во втором цикле выбрать ненулевые элементы массива $A(i)$ и присвоить их значения текущему элементу массива $B(k)$.
3. В третьем цикле вывести первые k элементов массива $B(i)$, а последующим $(30 - k)$ элементам массива $B(i)$ сначала присвоить нулевое значение, а потом вывести их.

На рис.1.18 приводится соответствующая схема алгоритма.

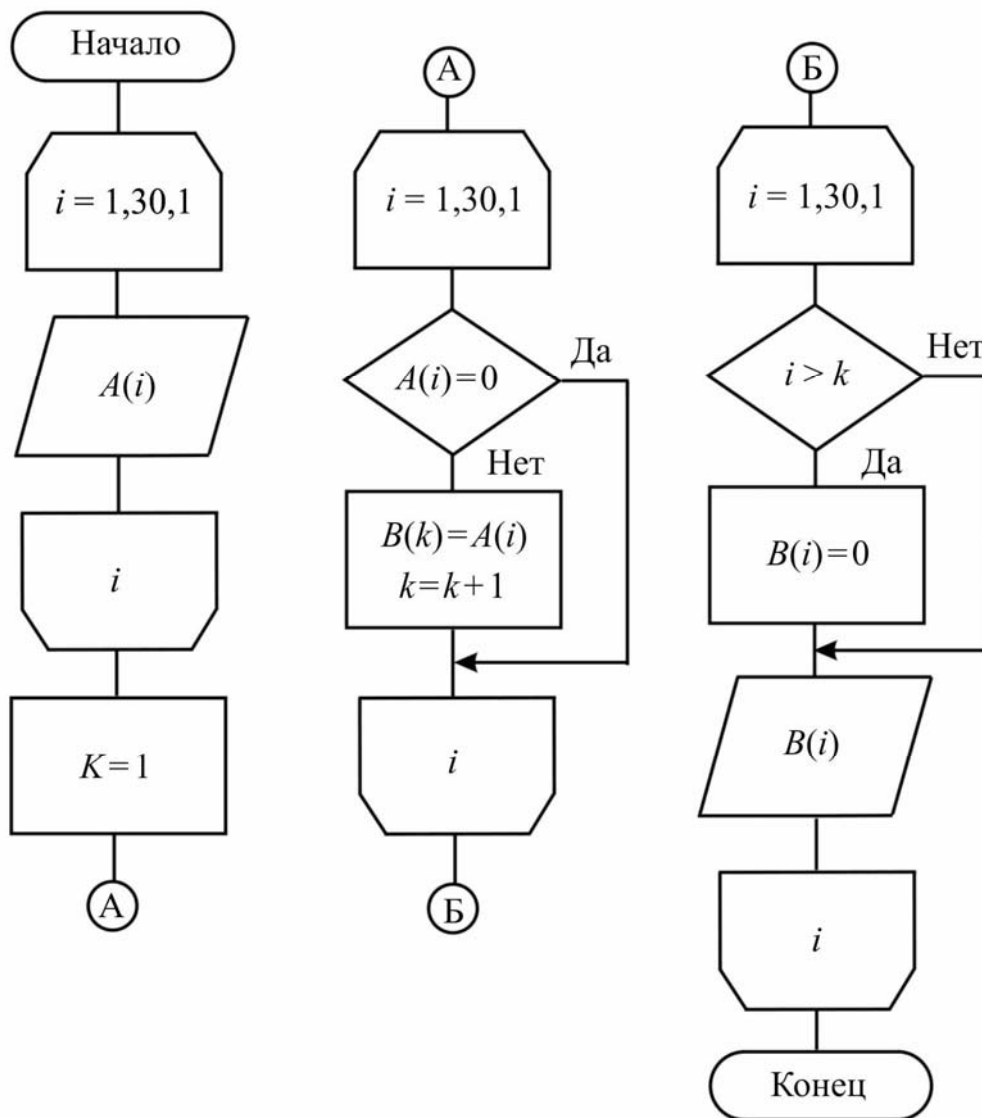


Рис. 1.18

1.4.5. Задачи по поиску и обработке текста

Символы, образующие текст, можно рассматривать как элементы одномерного массива. При этом номер символа в тексте совпадает с индексом элемента в массиве. В большинстве языков программирования для работы с текстом используют строковые константы, доступ к символам которых организуется с помощью специальных функций по номеру символа в константе, аналогично, как это делается для элементов массива.

Пример 5

Дано слово, хранящееся в массиве $A(N)$. Определить, есть ли в этом слове символ «а». Если есть, то каков номер первого символа «а» в слове?

На рис. 1.19 приводится соответствующий алгоритм.

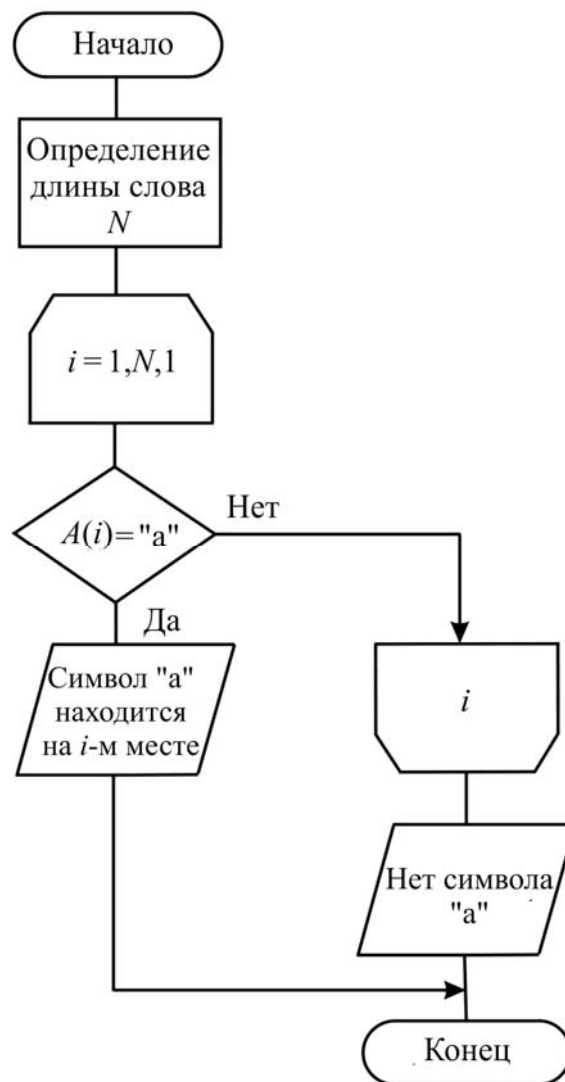


Рис. 1.19

Определив количество символов в слове, в цикле организуем их последовательное сравнение с символом «а». Если такой символ встретится, то выйдем из цикла и выведем сообщение «Символ "а" хранится на i -м месте». Иначе после проверки всех символов в слове выведем сообщение «Нет символа "а"».

Пример 6

Дан текст из N символов, хранящийся в массиве $A(N)$. Заменить пробелы между словами на запятую.

На схеме алгоритма (рис. 1.20) сначала определяется количество символов в тексте N . Затем организуется цикл с числом повторений N , в котором последовательно перебираются все символы текста и если встретился символ пробела, то он заменяется на символ запятая «,».

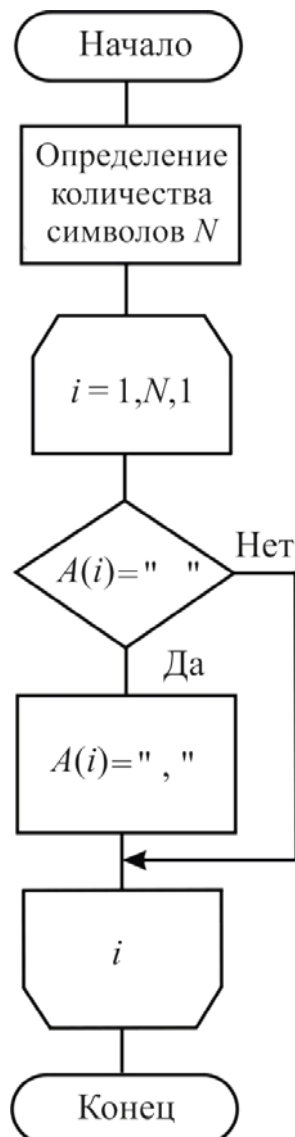


Рис. 1.20

Пример 7

Дан текст из N символов. Подсчитать:

- 1) количество слов в данном тексте;
- 2) количество слов, у которых первый и последний символы совпадают.

На рис. 1.21 и 1.22 приводятся данные алгоритмы.

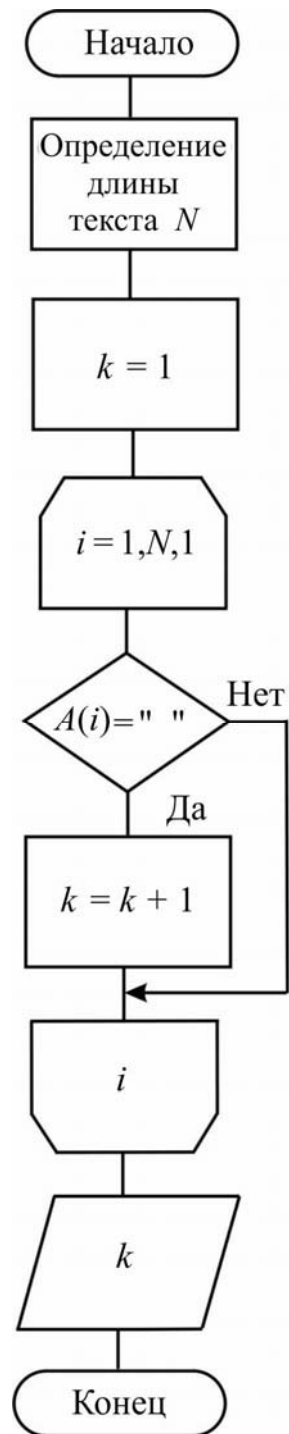


Рис. 1.21

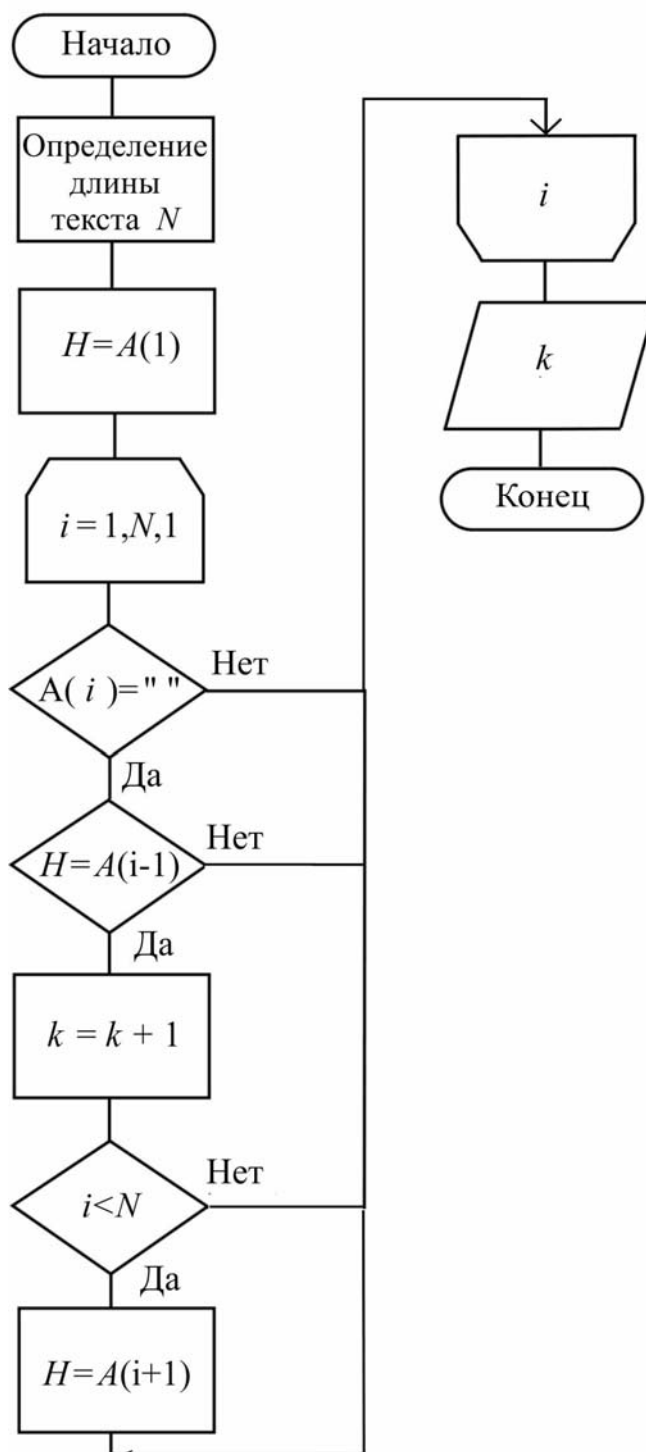


Рис. 1.22

Группы символов, расположенные между пробелами и не содержащие пробелов внутри себя, называем *словами*.

Перебирая в цикле символы текста (см. рис. 1.21), будем изменять значение переменной k на единицу в случае, если встретился символ пробела.

После завершения цикла в переменной k будет храниться искомое значение, которое выведем в качестве результата работы алгоритма.

Для определения количества слов, у которых первый и последний символы совпадают, запомним в переменной H первый символ текста (см. рис. 1.22).

Организуем цикл, в котором будем последовательно перебирать символы, и, как только встретится символ пробела, это будет означать, что закончилось очередное слово. В этом случае сравним предыдущий символ текста со значением, хранящимся в переменной H , если они совпадут, то увеличим значение переменной k на единицу и, если текст еще не закончился, в переменной H сохраним значение первого символа следующего слова.

Таким образом, после выхода из цикла в переменной k будет храниться искомое значение, которое выведем в качестве результата работы алгоритма.

Пример 8

Дан текст из N символов. Исключить из текста группу символов, расположенную между скобками, включая сами скобки. Предполагается, что внутри скобок нет других скобок. Вывести полученный текст.

На рис. 1.23 приводится соответствующий алгоритм.

В начале алгоритма присвоим признаку $3 = 0$ (нет открывающей скобки). Последовательно перебирая в цикле символы текста, пока $3 = 0$, увеличиваем значение переменной k на единицу и сохраняем в одномерном массиве $B(k)$ текущее значение элемента $A(i)$.

Если в тексте встретится символ «открывающая скобка», то присвоим признаку 3 значение, равное единице, и, пока в тексте не встретится символ закрывающей скобки, алгоритм будет обходить ветвь, в которой происходит увеличение значения k на единицу. Таким образом, символы, заключенные в скобки, не сохраняются в массиве $B(k)$.

Как только встретится закрывающая скобка, значение признака 3 снова станет равным нулю и на следующем шаге цикла текущее значение элемента $A(i)$ сохранится в соответствующем элементе массива $B(k)$.

Для вывода полученного текста организуем цикл с числом повторений, равным k .

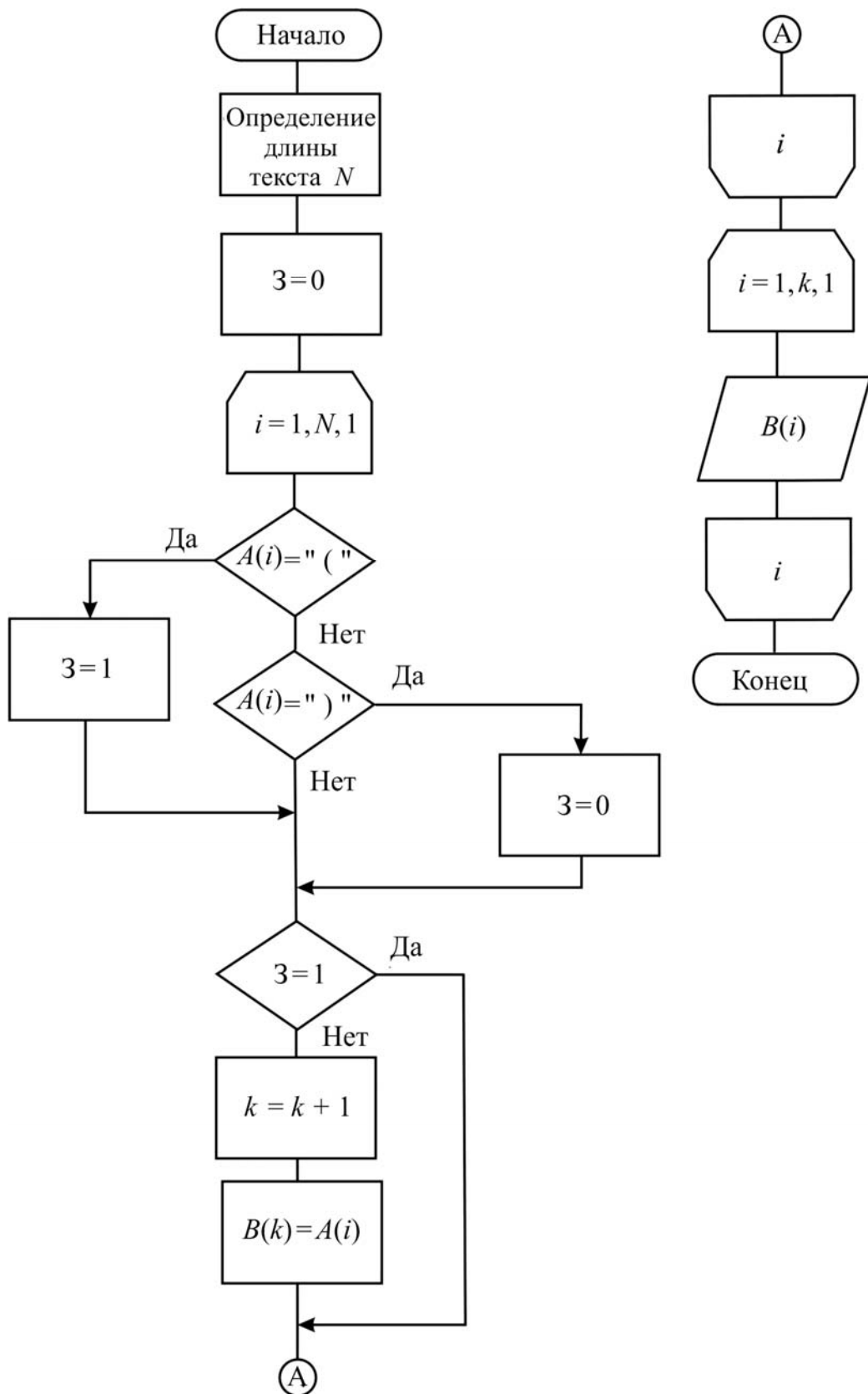


Рис. 1.23

1.4.6. Вложенные циклы

Возможны случаи, когда внутри тела цикла необходимо повторять некоторую последовательность операторов, т.е. организовывать внутренний цикл. Такая структура получила название *цикла в цикле*, или *вложенных циклов*. Вложенные циклы не могут пересекаться: пока не закончился внутренний цикл, внешний не может завершиться. Глубина вложения циклов, т.е. количество вложенных друг в друга циклов, может быть различной. При использовании такой структуры для экономии машинного времени необходимо выносить из внутреннего цикла во внешний цикл все операторы, которые не зависят от параметра внутреннего цикла.

Если число повторений вложенного цикла известно или зависит от значения переменной, модифицируемой в цикле, то в схеме алгоритма можно использовать символ границы цикла.

Пример 9

Рассмотрим фрагмент алгоритма, вычисляющего значения функции

$$A = \prod_{k=1}^n \sqrt{x^2 + k^2}$$

при x , изменяющемся от 0 до 5 с шагом 0,5 для любого значения n .

Вычисление произведения организуется в теле цикла путем накопления – последовательного перебора и умножения отдельных сомножителей $A = A * f(x, k)$ (рис. 1.24). При этом первое произведение для каждого значения x получается путем умножения первого сомножителя на единицу (т.е. перед внутренним циклом необходимо присвоить A значение, равное единице).

Если необходимо организовать вычисление суммы, то в теле цикла нужно последовательно перебирать и складывать отдельные слагаемые по формуле $A = A + f(x, k)$, а первая сумма получится путем сложения первого слагаемого с нулем, т. е. перед внутренним циклом необходимо присвоить A значение, равное нулю.

С помощью вложенных циклов выполняется и задача сортировки. *Сортировка* – это расположение чисел в порядке возрастания или убывания.

Наиболее распространенный и простой метод сортировки – метод «пузырька». Он требует минимального объема памяти для данных, но затраты времени на этот метод велики.

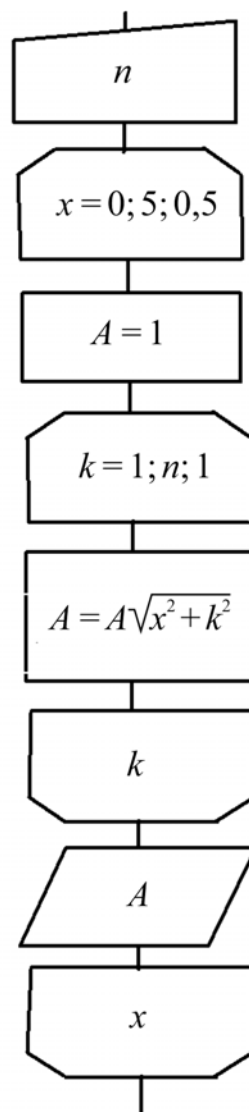


Рис. 1.24

Суть метода «пузырька» состоит в следующем.

Пусть дано n чисел и их необходимо расположить (для определенности) в порядке возрастания. При упорядочении можно использовать следующий алгоритм:

- 1) числа сравниваются попарно: первое со вторым; второе с третьим; ... , i -е с $(i + 1)$ -м;
- 2) если меньшее стоит в паре на втором месте, то числа меняются местами.

За один такой просмотр массива минимальное число перемещается, по крайней мере, на одно место вверх (вперед), а максимальное – в самый конец, т. е. минимальное число как легкий пузырек воздуха в жидкости постепенно «всплывает» в начале последовательности. Отсюда название метода.

За $K = (n-1)$ просмотров произойдет полное упорядочение массива при любом исходном расположении чисел в нем.

Рассмотрим алгоритм на примере шести чисел: 2, 1, 4, 5, 6, 3 (рис. 1.25).

2	1	4	5	6	3	– исходный массив
2	1	4	5	6	3	
						(Перестановка)
1	2	4	5	6	3	– после 1-го шага
1	2	4	5	6	3	– после 2-го шага
1	2	4	5	6	3	– после 3-го шага
1	2	4	5	6	3	– после 4-го шага
						(Перестановка)
1	2	4	5	3	6	Результат после 5-го шага

Рис. 1.25

Итак, после 5-го шага на последнем месте максимальное число «6», а минимальное число передвинулось на одно место вперед, и произошли две перестановки.

Если повторить указанный алгоритм еще четыре раза ($n - 2$, где n – число элементов массива), то будет выполнено полное упорядочение массива в порядке возрастания.

На рисунке 1.26 приведена соответствующая схема алгоритма.

Глубину просмотра можно уменьшить, основываясь на том, что большие числа «опускаются» вниз (в конец последовательности) и затем не переставляются, поэтому их можно не анализировать при последующих просмотрах.

При этом на схеме алгоритма конечное значение переменной внутреннего цикла i должно зависеть от номера просмотра и равняться $n - K$.

Сокращение количества просмотров улучшает временные характеристики алгоритма. Этого можно достичь, проводя анализ того, были ли перестановки на предыдущем просмотре. Если их не было, то можно заканчивать сортировку – данные уже отсортированы. При большой размерности массивов это дает значительную экономию времени.

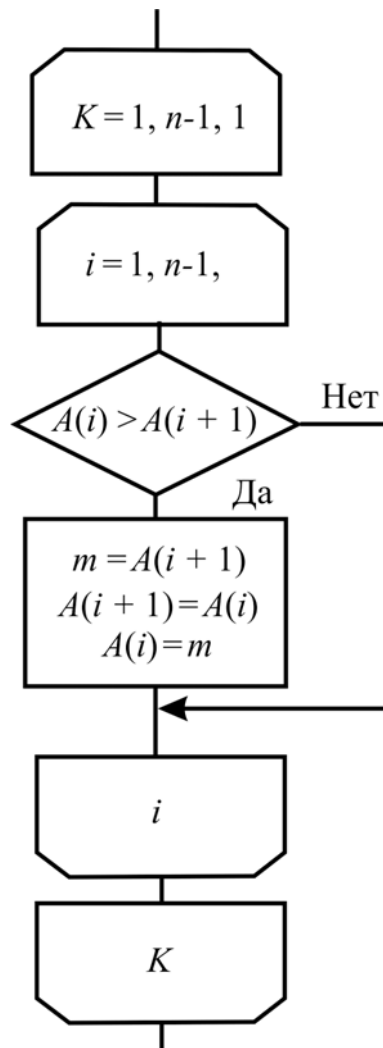


Рис. 1.26

На рис. 1.27 приведен алгоритм, в котором для анализа наличия перестановок при очередном просмотре введен признак:

$P = 1$ – перестановка есть;

$P = 0$ – перестановки нет.

Двумерный массив, или *матрица*, часто используется для описания данных одного типа. Например, таблицы сложения и умножения, коэффициенты в системе линейных уравнений и т. п. Элементы матрицы задают с помощью индексов: i – номер строки и j – номер столбца:

$$A(i, j) = \begin{bmatrix} A(1,1) & A(1,2) & \dots & A(1,j) \\ A(2,1) & A(2,2) & \dots & A(2,j) \\ \vdots & \vdots & \ddots & \vdots \\ A(i,1) & A(i,2) & \dots & A(i,j) \end{bmatrix}.$$

Пример 10

Присвоить значения элементам матрицы $A(5,3)$. Вычислить сумму элементов заданной матрицы и вывести матрицу и полученную сумму на дисплей.

На рис. 1.28 приводится соответствующий алгоритм. После ввода значения каждого элемента матрицы $A(i, j)$ его значение выводится на дисплей в строку и прибавляется к текущему значению переменной S . После завершения внутреннего цикла выполняется перевод строки, алгоритм переходит на конец внешнего цикла и, пока $i \leq 5$, внутренний цикл снова повторяется, последовательно выводя строки матрицы. После выхода из внешнего цикла сформированная в переменной S искомая сумма выводится на дисплей.

Пример 11

Ввести матрицу $A(i, j)$ размерностью ($i = 5, j = 8$). Подсчитать количество отрицательных чисел в ней и заменить эти числа нулями. Вывести новую матрицу. На рис. 1.29 приведен соответствующий алгоритм.

В данном алгоритме можно совместить в двух общих вложенных циклах присвоение значений элементам матрицы, анализ их знака, подсчет количества отрицательных элементов матрицы, их замену на нуль и вывод матрицы.

Пример 12

Подсчитать и вывести на дисплей суммы элементов той же матрицы по строкам. Схема алгоритма приведена на рис. 1.30.

Во внутреннем цикле для каждой строки матрицы формируются промежуточные суммы элементов по строкам, а после выхода из внутреннего цикла они выводятся на дисплей.

Прежде чем произойдет переход на следующую строку матрицы $A(i, j)$, переменную S необходимо обнулить.

В некоторых случаях при обработке матриц достаточно организовать только один цикл. Это, например, задачи, в которых обрабатываются диагонали матриц.

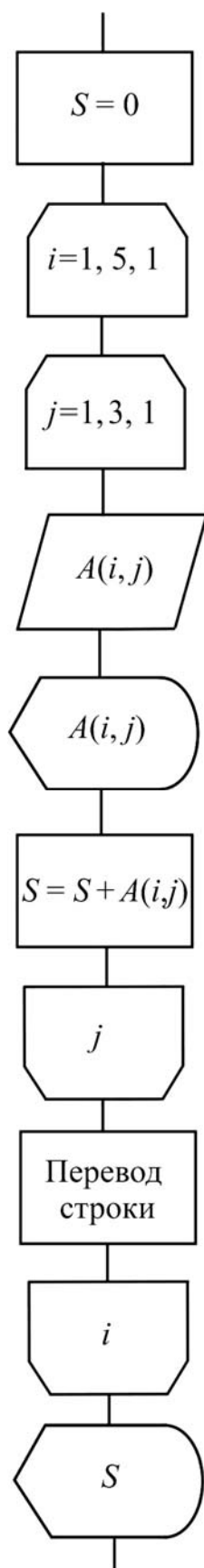


Рис. 1.28

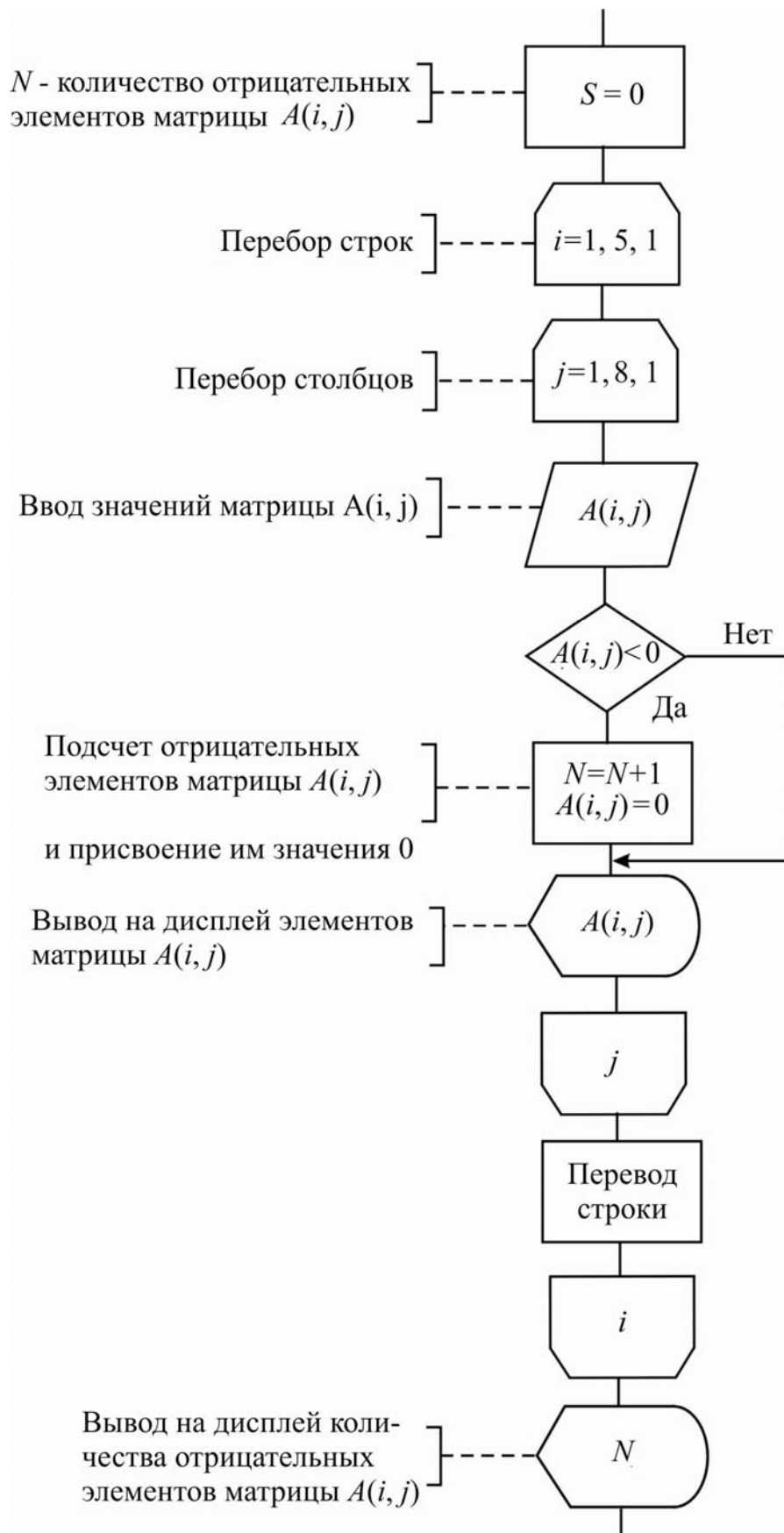


Рис. 1.29

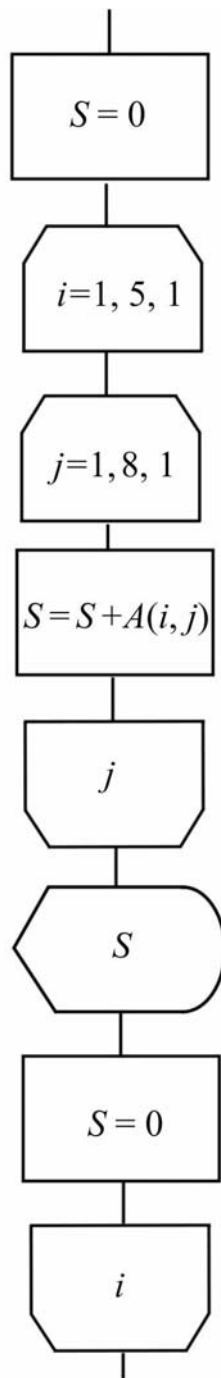


Рис. 1.30

Пример 13

Для матрицы $A(i, j)$ размерностью 8×8 подсчитать сумму элементов ее главной и побочной диагоналей и вывести их на дисплей.

При составлении схемы алгоритма (рис. 1.31) учтем, что элементы главной диагонали имеют одинаковые индексы, а у побочной номер строки и столбца связаны между собой следующим соотношением: $j = 9 - i$.

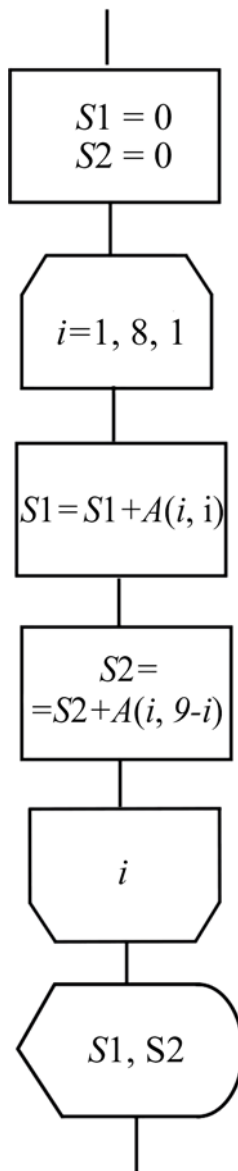


Рис. 1.31

1.4.7. Итерационные циклы

Особенностью *итерационного цикла* является то, что число повторений операторов тела цикла заранее неизвестно, а зависит от значений параметров (некоторых переменных), участвующих в вычислениях. Для его организации используется цикл типа «пока». Выход из итерационного цикла осуществляется в случае выполнения заданного условия.

На каждом шаге вычислений происходит последовательное приближение и проверка условия достижения искомого результата.

Алгоритм, в состав которого входит итерационный цикл, называется итерационным алгоритмом. Итерационные алгоритмы ис-

пользуются при реализации итерационных численных методов. В итерационных алгоритмах необходимо обеспечить обязательное достижение условия выхода из цикла (сходимость итерационного процесса). В противном случае произойдет заикливание алгоритма, т.е. не будет выполняться основное свойство алгоритма – результативность (или конечность). Это свойство состоит в том, что алгоритм должен приводить к решению задачи за конечное число шагов.

Рассмотрим применение итерационного метода для приближенного вычисления функций.

Пример 14

Вычислить значение функции $\sin(X)$ по приближенной формуле

$$F = \sin X = X - X^3/3! + X^5/5! - \dots$$

с погрешностью $E = 10^{-5}$.

Функция F представляет собой сумму членов степенного ряда:

$$F = U_1 + U_2 + \dots + U_n.$$

Условие окончания вычисления функции F

$$|U_n - U_{n+1}| \leq E,$$

где U_n – общий член ряда; U_{n+1} – следующее слагаемое.

Слагаемые можно определить:

1) непосредственно по формуле (но при большом значении n это неэффективно);

2) используя предыдущее значение.

Запишем формулы для вычисления U_n , учитывая, что в выражении для $\sin(X)$ присутствуют только нечетные степени X и факториалы нечетных чисел. Если общий член ряда равен

$$U_n = \frac{X^{2n-1}}{(2n-1)!},$$

то следующее слагаемое ряда равно

$$U_{n+1} = -\frac{X^{2n+1}}{(2n+1)!},$$

откуда U_{n+1} через U_n можно определить как

$$U_{n+1} = -\frac{X^2}{2n(2n+1)} \cdot U_n.$$

Рассмотрим сначала словесный алгоритм.

1. Ввести значения аргумента X и точность вычисления E .
2. Положить $F = 0$, $n = 1$, $U_{n+1} = X$.
3. Заменить $U_n = U_{n+1}$.
4. Вычислить $F = F + U_n$.
5. Увеличить $n = n + 2$.
6. Вычислить

$$U_{n+1} = -\frac{X^2}{2n(2n+1)} \cdot U_n.$$

7. Пока $|U_{n+1} - U_n| > E$, повторять пп. 3–6.
8. Вывести значение F .

Схема для рассмотренного алгоритма приведена на рис. 1.32.

Пример 15

Составить алгоритм вычисления суммы ряда

$$S = X - \frac{X^2}{2} + \frac{X^3}{3} - \frac{X^4}{4} + \dots (-1)^{i-1} \frac{X^i}{i} + \dots$$

с заданной точностью E (для данного знакочередующегося степенного ряда требуемая точность будет достигнута, когда очередное слагаемое станет по абсолютной величине меньше E).

При составлении алгоритма нужно учесть, что знаки слагаемых чередуются, а степень числа x в числителях слагаемых возрастает.

Решая эту задачу «в лоб» путем вычисления на каждом i -м шаге частичной суммы

$$S = S + (-1)^{i-1} \frac{X^i}{i},$$

мы получим неэффективный алгоритм, требующий выполнения большого числа операций.

Гораздо лучше организовать вычисления следующим образом: если обозначить числитель какого-либо слагаемого буквой p , то у следующего слагаемого числитель будет равен $-p \cdot X$ (знак минус обеспечивает чередование знаков слагаемых), а само слагаемое m будет равно p/i , где i – номер слагаемого.

На рис. 1.33 приведена соответствующая схема алгоритма.

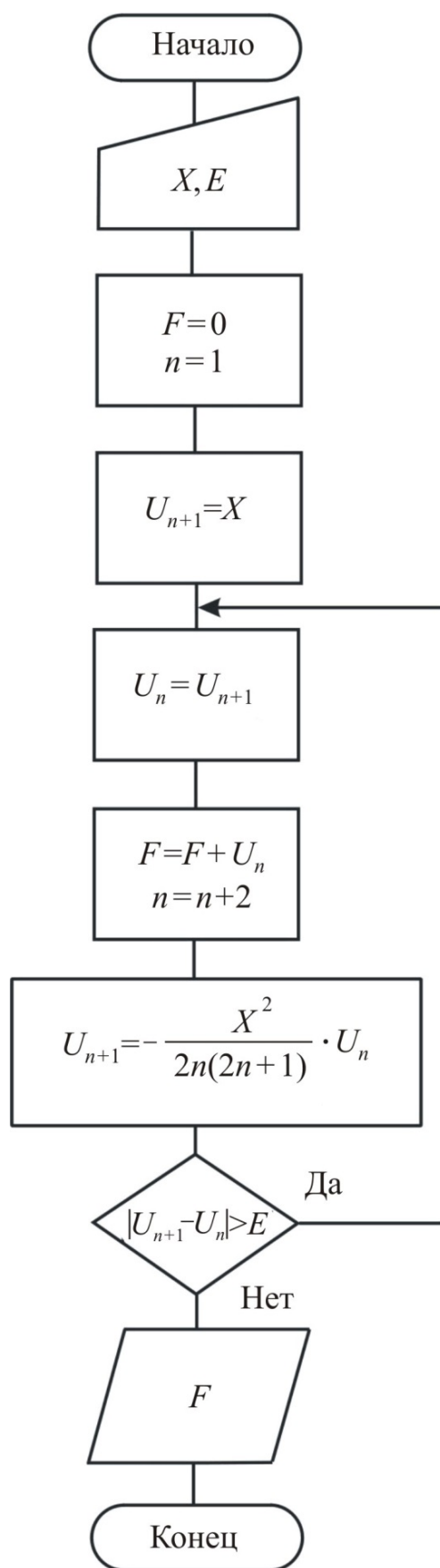


Рис. 1.32

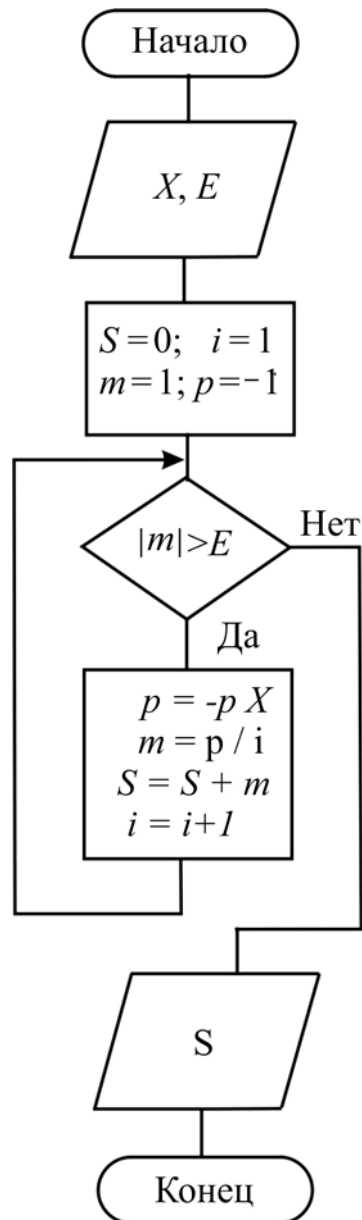


Рис. 1.33

1.4.8. Задания для самостоятельного выполнения

1. Найти расстояние от заданной точки A с координатами (X, Y) до начала координат.
2. Найти площадь заданной заштрихованной области.
3. Найти радиус окружности, вписанной в квадрат со стороной A .
4. Вычислить периметр квадрата, если известна его площадь.
5. Определить целую и дробную часть среднего арифметического двух чисел.

6. Дано число 20 в десятичной системе счисления. Составить программу перевода этого числа в восьмеричную систему счисления.

7. Дано число AAA в шестнадцатеричной системе счисления. Составить программу перевода этого числа в десятичную систему счисления.

8. Заданы координаты точек A и B . Написать программу вычисления координат точки C , симметричной точке A относительно B .

9. Вычислить координаты центра экрана размером $A \times B$ (координаты могут быть только целыми числами).

10. Составить схему алгоритма для нахождения наименьшего (наибольшего) значения из трех чисел a, b, c . В алгоритме предусмотреть ввод с клавиатуры трех чисел, а также вывод на экран монитора наименьшего (наибольшего) значения.

11. Составить схему алгоритма для расстановки трех чисел a, b, c в порядке убывания (возрастания). В алгоритме предусмотреть ввод с клавиатуры трех чисел, а также вывод на принтер чисел в порядке убывания (возрастания).

12. Даны действительные числа x, y, z . Удвоить эти числа, если $x \geq y \geq z$, и заменить их абсолютными значениями, если это не так.

13. Если сумма трех попарно различных действительных чисел x, y, z меньше единицы, то наименьшее из этих трех чисел заменить полусуммой двух других; в противном случае – заменить меньшее из x и y полусуммой двух оставшихся значений.

14. Даны действительные числа a, b . Если они отрицательны, то каждое значение заменить его модулем. Если отрицательно только одно из них, то оба значения увеличить на 0,5. Если оба значения неотрицательны и ни одно из них не принадлежит отрезку $[0,5; 2,0]$, то оба значения уменьшить в 10 раз. В остальных случаях a и b оставить без изменения.

15. Даны действительные положительные числа x, y, z . Выяснить, существует ли треугольник с длинами сторон x, y, z .

16. Даны целые числа k, m и действительные числа x, y, z . При $k \leq m^2$ или $k > m^2$ заменить модулем значение x , а значения y и z уменьшить на 0,5.

17. Дан одномерный массив A длиной $M \leq 20$. Найти первый минимальный (максимальный) элемент и поменять его местами с первым (последним) элементом массива.

18. Одномерный массив A длиной $M \leq 25$ заполнить случайными числами из диапазона $[x1... x2]$. Определить количество элементов, значения которых лежат в диапазоне $[y1... y2]$.

19. Одномерный массив A заполнить случайными числами из диапазона $[-20...20]$. Переместить:

- нулевые элементы массива в начало, сдвинув остальные элементы вправо;

- минимальные элементы в начало, сдвинув остальные элементы вправо;

- максимальные элементы в конец, сдвинув остальные элементы влево.

20. Дан текст из N символов. Подсчитать количество слов, начинающихся с буквы «б».

21. Дан текст из N символов. Определить, есть ли в тексте оператор присваивания «:=».

22. Дан текст из 30 символов. Поменять местами первый и последний символы. Удалить из полученного текста все пробелы.

23. Дана действительная матрица порядка $M \times N$. Найти:

- сумму наибольших (наименьших) значений ее строк (столбцов);

- сумму элементов строки (столбца), в которой расположен элемент с наименьшим (наибольшим) значением (предполагается, что такой элемент единственный).

24. В данной действительной квадратной матрице порядка $N \times N$:

- поменять местами строку, в которой расположен элемент с наибольшим значением, со строкой, содержащей элемент с наименьшим значением (предполагается, что такие элементы единственны);

- указать индексы всех элементов с наибольшим (наименьшим) значением;

- найти строки (столбцы), элементы которых упорядочены по возрастанию (убыванию);

- сумму положительных (отрицательных) элементов на главной (побочной) диагонали;

- сумму элементов на обеих диагоналях;

- сумму положительных (отрицательных) элементов над/под главной (побочной) диагональю;

– минимальный (максимальный) элемент на/над/под главной (побочной) диагональю.

25. Для элементов главной диагонали, меньших/больших нуля, вывести сумму/максимум/минимум элементов строки/столбца, где этот элемент расположен.

26. Определить последовательность $B(1)...B(N)$ из нулей и единиц, такую, что $B(i) = 1$, если:

– в строке/столбце матрицы есть хотя бы один отрицательный/положительный/нулевой элемент;

– строка/столбец образует возрастающую/убывающую последовательность;

– вывести индексы ненулевых (или с другим признаком) элементов.

27. Транспонировать данную целочисленную квадратную матрицу порядка 20×20 .

28. Определить, является ли квадратная матрица симметричной относительно главной (побочной) диагонали.

29. Даны две целочисленные квадратные матрицы порядка 20. Найти последовательность из нулей и единиц $B_1, \dots, B_{20}$, такую, что $B(i) = 1$, когда:

– все элементы i -х строк первой и второй матриц отрицательны;

– каждый элемент i -й строки первой матрицы больше соответствующего элемента i -й строки второй матрицы;

– количество отрицательных элементов i -й строки первой матрицы равно количеству отрицательных элементов той же строки второй матрицы.

30. Вычислить бесконечную сумму (произведение) с заданной точностью E :

$$S = \sum_{i=1}^{\infty} \frac{1}{i^2};$$

$$S = \sum_{i=1}^{\infty} \frac{(-1)^{i+1}}{i(i+1)(i+2)};$$

$$S = \prod_{i=1}^{\infty} \left(1 - \frac{1}{(2i+1)^2} \right);$$

$$S = \prod_{i=1}^{\infty} \left(1 - \frac{4x^2}{(2i-1)^2 \pi^2} \right).$$

Считать, что требуемая точность достигнута, если очередное слагаемое (сомножитель) стало по модулю меньше, чем E , тогда это и все последующие слагаемые (сомножители) можно не учитывать.

Глава 2. Программирование на языке VBA в Microsoft Office

2.1. Общие сведения о языках программирования

Различные языки программирования от самых примитивных языков до языков, близких к естественному языку человека, позволяют создавать совершенные и эффективные программы с продолжительным жизненным циклом. Постоянное повышение сложности задач, решаемых с помощью ЭВМ, а также совершенствование самих ЭВМ требует разработки новых специализированных, ориентированных на проблемную область и более мощных языков программирования.

В развитии инструментального программного обеспечения рассматривают пять поколений языков программирования. Первые три поколения характеризовались более сложным набором зарезервированных слов и синтаксисом. Языки четвертого поколения все еще требуют соблюдения определенного синтаксиса при написании программ, но он значительно легче для освоения. Разрабатываемые в настоящее время естественные языки программирования составляют пятое поколение. В табл. 2.1 приводится общая характеристика языков программирования [5].

Таблица 2.1

Поколения языков	Языки программирования	Характеристика
Первое	Машинные	Ориентированы на использование в конкретной ЭВМ, сложны в освоении, требуют хорошего знания архитектуры ЭВМ
Второе	Ассемблеры, макроассемблеры	Более удобны для использования, но машинно-зависимы
Третье	Языки высокого уровня	Мобильные, человеко-ориентированные, проще в освоении
Четвертое	Непроцедурные, объектно-ориентированные, языки запросов, параллельные	Ориентированы на непрофессионального пользователя и на ЭВМ с параллельной архитектурой
Пятое	Языки искусственного интеллекта, экспертных систем и баз знаний, естественные языки	Ориентированы на повышение интеллектуального уровня ЭВМ и интерфейса с языками

Краткий обзор современных языков программирования можно посмотреть в [6].

В [7] проводится краткая характеристика алгоритмического, структурного, событийно-ориентированного и объектно-ориентированного программирования.

Алгоритмические языки программирования представляют алгоритм в виде последовательности основных алгоритмических структур:

- линейных;
- разветвляющихся;
- циклических.

Алгоритм на выбранном языке программирования описывается с помощью команд описания данных, вычисления значений и управления последовательностью выполнения программы.

Основными элементами алгоритмического языка программирования являются:

- переменные и константы;
- арифметические операции, используемые для создания арифметических выражений;
- логические операции и выражения;
- указатели. В некоторых языках программирования возможно явное указание адреса физической памяти как в Ассемблере;
- сложные данные (списки, деревья, ...);
- массивы;
- описания переменных;
- операторы присвоения;
- комментарии;
- условные операторы;
- операторы цикла;
- операторы ввода-вывода.

Структурное программирование предполагает дополнительное использование подпрограмм, процедур и функций. Наличие подпрограмм позволяет вести проектирование и разработку приложений сверху–вниз – такой подход получил название – нисходящее проектирование. Сначала общая задача разбивается на глобальные подзадачи (модули). Затем каждый модуль разбивается на подпрограммы, которые, в свою очередь, могут содержать другие подпро-

граммы. Небольшие подпрограммы значительно проще отлаживать, а это повышает общую надежность всей программы. Кроме того, подпрограммы могут повторно использоваться, а это повышает производительность труда программистов.

Развитием идеи нисходящего проектирования стало *событийно-ориентированное* программирование. Оно возникло в связи с широким распространением визуальных систем типа Windows, идеология которых основана на событиях. Программы при *событийно-ориентированном* программировании имеют следующую структуру: главная часть программы – один бесконечный цикл, который опрашивает Windows, следя за появлением новых сообщений. При обнаружении сообщения вызывается подпрограмма, отвечающая за обработку соответствующего события, а цикл опроса продолжается до получения сообщения «конец работы».

Применение структурного и событийно-ориентированного программирования существенно повысило производительность труда программистов. Но это уже был предел возможностей человека.

Как развитие технологии структурного программирования в середине 1980-х гг. возникло понятие *объекта* и *объектно-ориентированного программирования* (ООП). ООП предполагает создание приложений из объектов, которые можно позаимствовать в готовом виде из разнообразных библиотек, либо при необходимости разрабатывать самостоятельно.

Важное место в технологии ООП занимает *событие* (щелчок клавиши мыши на объекте, нажатие определенной клавиши, открытие документа и т.д.). В качестве реакции на событие может вызываться определенная *процедура*, которая может изменять *свойства* объекта, вызывать его *методы*.

В настоящее время в ООП обычно используется графический интерфейс, позволяющий визуализировать процесс программирования. Появилась возможность создавать объекты, задавать им свойства с помощью мыши.

В среде Microsoft Office в качестве объектов выступают приложения, документы, окна и т.п. Каждый из этих объектов является исполнителем алгоритма, который задается с помощью команд на соответствующем формальном языке, либо другими объектами, функционирующими в данной системе, либо пользователем персонального компьютера. Запись алгоритма для объектов в среде Microsoft Office осуществляется с помощью языка программирования Visual Basic for Application (VBA), который является ядром ООП языка программирования Visual Basic.

Системы программирования содержат разнообразные средства создания программ:

- текстовый редактор;
- компилятор;
- редактор связей;
- библиотека функций;
- отладчик программы.

Из наиболее популярных, использовавшихся в течение длительных периодов времени, можно назвать следующие универсальные языки программирования: Basic, Pascal, C ++.

Для каждого из этих языков сегодня имеется немало систем программирования, ориентированных на разные персональные компьютеры и операционные системы разных поколений.

Наиболее популярные системы проектирования программ для непрофессиональных программистов Windows – это Microsoft Visual Basic, Borland Delphi, Borland C++ Bulider и т.д.

2.2. Объектная структура языка VBA

В качестве основного понятия объектно-ориентированного метода программирования используется понятие *объекта*. Данные и программный код, обслуживающий их, заключаются в единый объект. Для отображения этих данных необходимо послать объекту соответствующее сообщение. При внедрении объекта в приложение должно быть указано место его отображения на экране.

Любой *объект* обладает соответствующим набором свойств (характеристиками данного объекта) и набором *методов* (команд для обработки свойств) [8].

Объекты, объединяющие (инкапсулирующие) одинаковый перечень свойств и методов, образуют *классы*. При этом каждый отдельный объект является *экземпляром класса*. Однако экземпляры класса могут иметь отличные *значения* свойств.

Объекты VBA в Microsoft Office – это кнопки, элементы меню, документы, фрагменты документа, символы, интервалы ячеек рабочего листа и даже сам рабочий лист и т.п.

В табл. 2.2 приведены примеры классов объектов VBA для приложений Microsoft Office Word и Excel.

Объекты в приложениях образуют некоторую *иерархию*, на вершине которой находится приложение (*Application*).

Иерархия четырех уровней объектов в приложениях Microsoft Office Word и Excel приведена в табл. 2.3.

Таблица 2.2

Приложение	Класс объектов	Свойства	Методы
<i>Word</i>	<i>Documents</i> (документ)	<i>Name</i> (имя) <i>FileName</i> (полное имя файла) <i>Add</i> (создает новый объект в семействе) ...	<i>Open</i> (открыть) <i>Close</i> (закрыть) <i>Save</i> (сохранить) ...
<i>Excel</i>	<i>Workbooks</i> (книга)	<i>ActiveSheet</i> (активный лист книги) <i>Name</i> (имя) <i>Path</i> (полное имя папки, где находится книга) <i>Add</i> (создает новый объект в семействе) ...	<i>Open</i> (открыть) <i>Close</i> (закрыть) <i>Save</i> (сохранить) ...

Таблица 2.3

Word	Excel
Активное приложение (<i>Application</i>)	Активное приложение (<i>Application</i>)
Документ (<i>Documents</i>)	Книга (<i>Workbook</i>)
Фрагмент документа (<i>Selection</i>)	Лист (<i>Worksheet</i>)
Символ (<i>Character</i>)	Диапазон ячеек (<i>Range</i>)

Чтобы получить *доступ к объекту* в языке VBA, необходимо составить *ссылку* на него. Ссылка на требуемый объект содержит перечень всех объектов по иерархии, разделенных точками. Например, в программе Word ссылка на документ LP1.docx выглядит следующим образом:

Application.Documents ("LP1.docx")

Однако, если приложение Word активно, достаточно сделать относительную ссылку на сам документ:

Documents ("LP1.docx")

или, используя общее обращение к активному в данный момент документу без указания его имени:

ActiveDocument

В приложении Excel общее обращение к активной в данный момент рабочей книге выглядит следующим образом:

ActiveWorkbook

Для того чтобы объект выполнил какую-либо *операцию*, необходимо задать *метод*. Многие методы имеют *аргументы*, задающие параметры выполняемых действий.

Синтаксис команды применения метода объекта:

Объект.Метод arg1:=значение, arg2:=значение, ...

Например, операция открытия в приложении Word документа Проба.docx, находящегося на диске H: в папке ФИО, содержит не только название метода *Open*, но и указание пути к открываемому файлу, как значения аргумента *FileName*:

Documents.Open FileName:=(“H:\ФИО\Проба.docx”)

Для открытия нового документа в Word используется метод *Add* с указанием значения аргумента *DocumentType*:

Documents.Add DocumentType:=wdNewBlankDocument

Для вывода на печать двух первых страниц документа Проба.docx необходимо задать для метода *PrintOut* значения аргументов *Range* (задает формат диапазона печати), *From* и *To* (задают номер начальной и конечной страниц печати):

*Documents(“Проба.docx”).PrintOut _
Range:=wdPrintFromTo, From:=”1”, To:=”2”*

Обратите внимание, что в последнем выражении программная строка кода не поместилась в одной экранной строке, поэтому она перенесена на следующую экранную строку с помощью символа подчеркивания (_) с предшествующим символом пробела.

Для сохранения документа Проба.doc на диске в директории по умолчанию используется метод *Save*:

Documents(“Проба.docx”).Save

Для изменения *состояния* объекта необходимо задать новые значения его *свойств*.

Синтаксис команды изменения свойств:

Объект.Свойство = ЗначениеСвойства

Например, для установки во фрагменте текста (объект *Selection*) для восьмого символа (объект *Characters(8)*) начертания – полужирный (свойство *Bold*, которое имеет два значения *True* или *False*), введем следующий код:

Selection.Characters(8).Bold = True

Объект может иметь множество свойств. Для сокращения программного кода можно задавать значения сразу нескольким свойствам объекта с помощью инструкции *With*, не указывая явно его имя:

With Объект

.Свойство1 = ЗначениеСвойства1

.Свойство2 = ЗначениеСвойства2

...

End With

2.3. Основные этапы создания VBA-программ

Процесс создания VBA-программ включает в себя пять этапов.

На первом этапе разработки VBA-программ необходимо проанализировать все требования, которым должна удовлетворять разрабатываемая программа, и прежде всего – что должна делать программа и что представляют собой входные данные, какими ресурсами располагает проектировщик и какими должны быть выходные данные.

На втором этапе проектируется общая структура программы, т.е. определяются те объекты, из которых будет состоять программа (формы, диалоговые окна, элементы управления и т.д.), а также определяются общие принципы управления и взаимодействия между различными компонентами программы. Уже на этом этапе необходимо продумать интерфейс будущей программы.

На третьем этапе осуществляется реализация проекта на языке программирования VBA: проектируется внешний вид форм и связанных с ней элементов управления, задаются их свойства и пишутся процедуры, отвечающие за обработку связанных с ними событий.

На четвертом этапе производится проверка работы VBA-программ. При этом оценивается правильность, вычислительная сложность, а также эффективность ее реализации.

Последний этап – это сопровождение. На этапе эксплуатации программы устраняются обнаруженные при эксплуатации ошибки и находятся «узкие места» или неудачные проектные решения.

2.4. Макросы в Microsoft Office.

Редактор Visual Basic

Макросы являются удобным средством для автоматизации выполнения часто повторяемых действий, что позволяет повысить

точность и скорость работы в программах Office. Язык VBA был разработан Microsoft Office для унификации макроязыков в своих приложениях под Windows.

При создании макроса как в Word, так и в Excel макрорекодер записывает все действия пользователя в виде программного кода на языке VBA. При этом макросы сохраняются в файлах документов Word и в файлах рабочих книг Excel в специальной части файла, называемой модулем (Module). Модули, сохраняемые в одном документе Word, называют проектами Project, а в одной рабочей книге Excel называют проектами VBAProject.

Если при создании макроса были допущены ошибочные действия, то его можно отредактировать. Для повышения гибкости, расширения возможностей и оптимизации макросов при их редактировании может добавляться требуемый программный код на языке VBA. Кроме того, VBA позволяет организовать взаимодействие нескольких макросов между собой.

Внимание! Последующие разделы учебного пособия целесообразно изучать с одновременным выполнением соответствующих заданий на компьютере в изучаемой программной среде.

2.4.1. Создание макросов в Microsoft Word

Создадим первый макрос в текущем документе Word (версия 2010) «Глава2.docx» под именем *Макрос1*, который выполняет следующие действия:

- 1) устанавливает с помощью кнопок на панели инструментов начертание шрифта *Курсив* и цвет шрифта *Красный*;
- 2) выводит в строку, где находится курсор, текст «Я учусь писать макрос».

Для записи макроса в Word необходимо выбрать в разделе меню *Вид* команду *Макросы* ⇒ *Запись макроса*. В появившемся окне (рис. 2.1) будет указано имя макроса по умолчанию *Макрос1*, которое при необходимости можно изменить.

Так как в нашем случае макрос должен быть доступен только в текущем файле Word, то в списке *Макрос доступен для:* (см. рис. 2.1) требуется выбрать имя текущего документа. При сохранении макроса в шаблоне *Normal.dotm* он будет доступен для всех документов, созданных на его основе.

После нажатия на клавишу *ОК* необходимо выполнить все действия, описанные в пп. 1–2 данного раздела, а затем выбрать на панели *Макросы* команду *Остановить запись* (рис. 2.2).

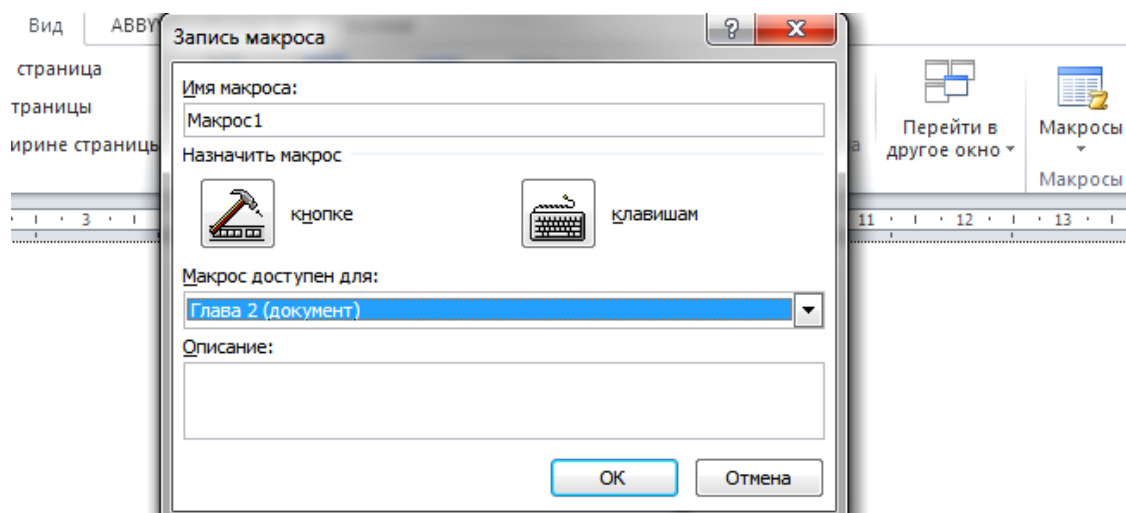


Рис. 2.1

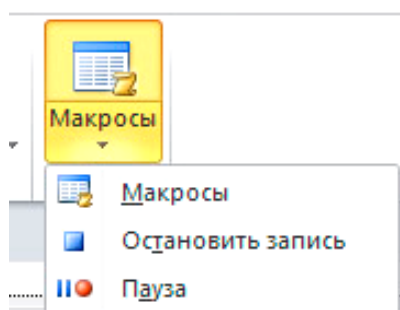


Рис. 2.2

После завершения записи макроса в документе Word добавится модуль под названием *NewMacros*, в котором помещается сгенерированный код на языке VBA.

Для запуска созданного макроса в Word переведем курсор в начало следующей строки и выберем в разделе меню *Вид* команду *Макросы* $\Rightarrow$ *Макрос1* $\Rightarrow$ *Выполнить*.

Создадим второй макрос в текущем документе Word под именем *Макрос2*, который в отличие от первого макроса устанавливает начертание шрифта *Курсив* и цвет шрифта – *Красный* с помощью диалогового окна *Шрифт*. Программный код второго макроса добавится в тот же модуль *NewMacros* текущего документа.

2.4.2. Интегрированная среда проектирования VBA

Для просмотра сгенерированного кода требуется выбрать команду *Макросы* $\Rightarrow$ *Макрос1* $\Rightarrow$ *Изменить* или нажать комбинацию клавиш *Alt+F11*. В результате откроется окно редактора Visual Basic в интерпретации Microsoft (рис. 2.3), которое называется *Интегрированной средой проектирования (IDE)*.

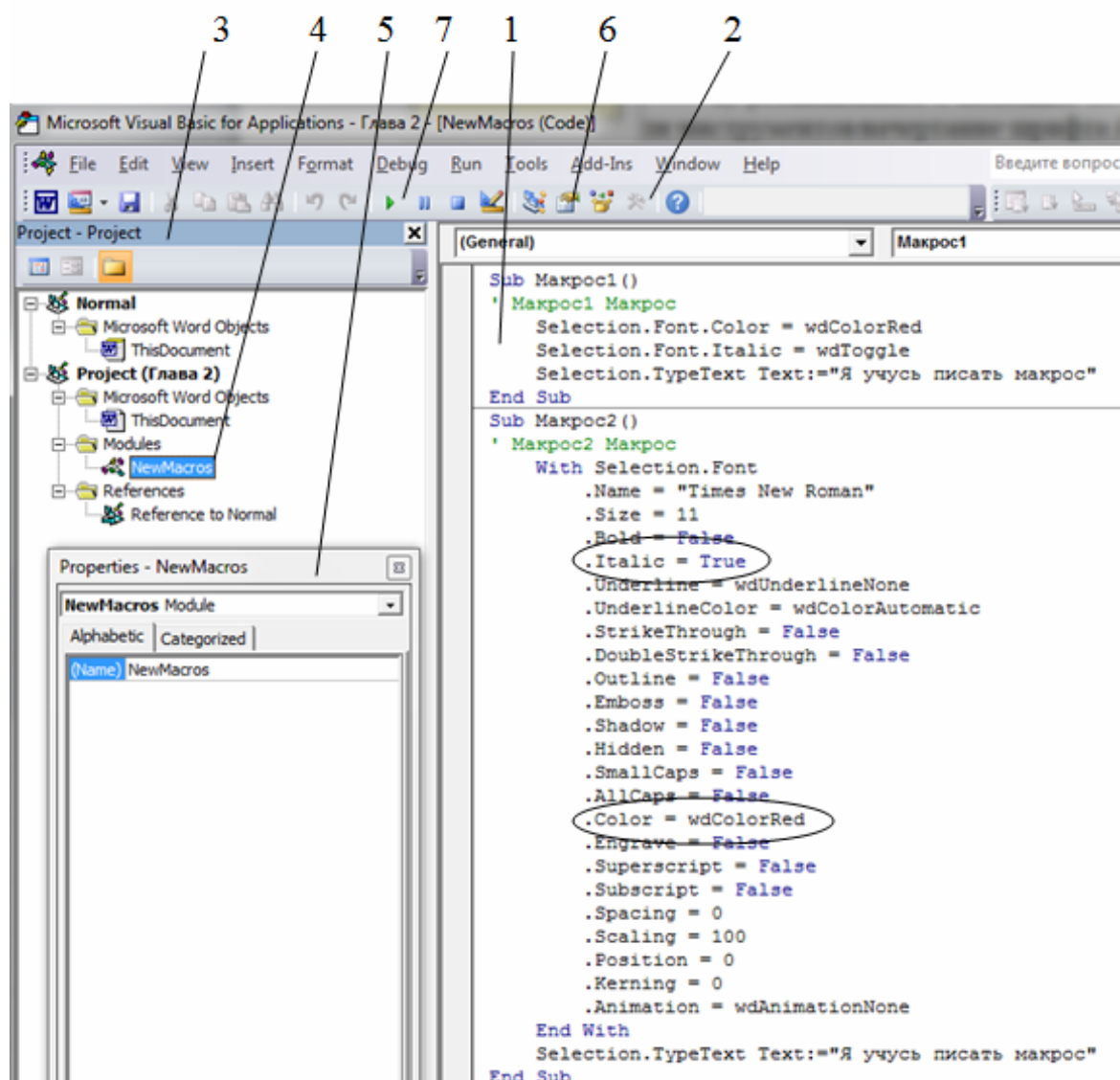


Рис. 2.3

IDE позволяет писать и редактировать программы, проектировать диалоговые окна с различными элементами управления, облегчающими взаимодействие с пользователем, запускать программы, тестировать и выполнять их отладку, контролировать взаимодействие различных VBA-программ.

Самое большое окно в IDE – окно редактирования кода (1), в котором можно посмотреть доступный для редактирования сгенерированный код макроса.

Над окном редактирования кода располагается стандартная панель инструментов (2), которую можно отобразить на экране выбором команды *View⇒Toolbars⇒Standard*.

На рис. 2.3 в окне проектов *Project* (3) содержатся два проекта: первый соответствует общему шаблону *Normal*, второй соответствует открытому документу *Глава 2*.

Каждый проект содержит модуль документа с именем *ThisDocument*.

Проект *Project* (Глава 2) дополнительно содержит модуль *NewMacros* (4), в котором хранятся созданные нами макросы, а также компонент *References*, который содержит ссылку на присоединенный к проекту шаблон (по умолчанию – это шаблон *Normal*).

Окно свойств *Properties* (5) предназначено для установки свойств форм и элементов управления. Окно отображается на экране нажатием кнопки (6)  на стандартной панели инструментов. Список свойств в окне разделен на две колонки.

В левой колонке окна свойств находятся имена свойств, а в правой – значения этих свойств. Установленные по умолчанию значения могут быть изменены. Для некоторых свойств объектов предусмотрена возможность выбора их значений из раскрывающегося списка. Ряд свойств объектов можно задавать и программно.

Для запуска на исполнение созданного макроса из среды IDE VBA необходимо нажать клавишу *F5* или кнопку *Run* (7) на стандартной панели инструментов. При этом в окне проектов должен быть выделен модуль *NewMacros*.

Программный код макроса при необходимости может быть отредактирован в окне редактирования. При этом применяются те же правила редактирования, что и в самом текстовом редакторе Word.

При написании программного кода его целесообразно снабжать *комментариями* – это неисполняемые строки, начинающиеся со знака апострофа (*'*).

Кроме того, во время отладки программы апостроф часто используют для временного отключения некоторых программных строк. Апостроф можно ставить вручную, а если надо добавить или удалить сразу несколько комментариев, то удобнее воспользоваться кнопками *CommenBlock* (закомментировать блок)  и *UncommenBlock* (раскомментировать блок)  панели инструментов *Edit*, которую выбирают в разделе меню *View* ⇒ *Toolbars*.

При написании кода в окне для редактирования редактор автоматически предлагает пользователю список компонентов, логически завершающих вводимую им инструкцию.

Например, при наборе кода для объекта *Selection* (определяет выделенный участок текста в Word, или если ничего не выделено, то место, где находится указатель вставки) после ввода точки на экране отобразится список компонентов (свойств или методов данного объекта), которые завершают данную конструкцию. Двойной щелчок на выбранном компоненте вставляет его имя в код программы.

2.4.3. Анализ полученного программного кода макроса

Сопоставим действия, которые были выполнены для написания первого макроса и полученный программный код из окна редактирования (см. рис. 2.3).

Первая и шестая строки *Макроса1* являются началом (*Sub*) и окончанием (*End Sub*) процедуры, соответственно. Вторая строка – это комментарий. В третьей строке устанавливается красный цвет шрифта – свойству *Color* присваивается значение константы *wdColorRed*. В четвертой строке задается начертание шрифта курсив – свойству *Italic* присваивается значение константы *wdToggle*, которая обозначает переключение текущей установки. При установке можно также использовать константы **True** (истина) и **False** (ложь). Пятая строка выводит в документе в строку, где находится курсор (объект *Selection*), текст «Я учусь писать макрос».

Сравнив текст первого и второго макросов, можно сделать вывод, что вместо третьей и четвертой строк добавилось 25 строк, в которых с помощью инструкции *With* задается 23 свойства объекта *Selection.Font* из диалогового окна *Шрифт*. Среди перечисленных свойств есть и свойства *Italic* и *Color* (на рис. 2.3 обведены овалами), значения которых менялись. Хотя все остальные настройки шрифта, включенные в текст *Макроса2*, не менялись, но в код макроса были включены все свойства шрифта. Значение этих свойств берется по умолчанию из диалогового окна *Шрифт*.

2.4.4. Сохранение макроса на диске

Для сохранения одного выбранного модуля в текущем файле из среды IDE достаточно нажать клавиши *Ctrl+S* или щелкнуть по кнопке  стандартной панели инструментов, или выбрать команду *File ⇒ Save*.

Для сохранения текста программы макроса в отдельном файле с расширением *.bas* необходимо нажать клавиши *Ctrl+E* или выполнить команду *File ⇒ ExportFile*. А чтобы загрузить в IDE VBA текст программы из файла с расширением *.bas*, необходимо нажать клавиши *Ctrl+M* или выполнить команду *File ⇒ ImportFile*.

Чтобы сохранить сразу все изменения в текущем проекте, включая все связанные с ним модули, необходимо вернуться в активное приложение Word и выбрать команду *Сохранить как:*.

Следует иметь в виду, что при сохранении на диске документа вместе с макросом в окне сохранения документа необходимо выбрать тип документа *Документ Word с поддержкой макросов*. При этом файл будет сохранен с расширением *.docm*.

Сохранить файл с макросами с расширением *.docm*. Файл называть по своей фамилии.

2.4.5. Открытие файлов с макросами

При открытии файлов с макросами появляется предупреждение системы безопасности (рис. 2.4). Чтобы иметь возможность работать с макросами в открываемом файле, необходимо нажать на клавишу *Включить содержимое*. Иначе макросы в данном файле будут не доступны.

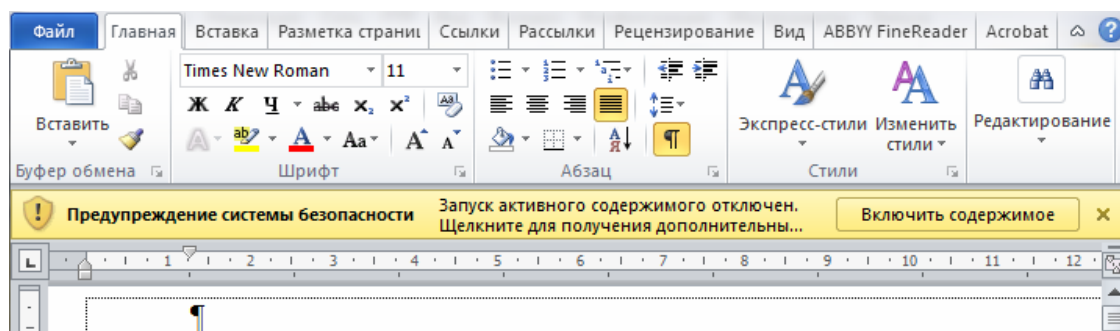


Рис. 2.4

Для настройки параметров макросов в центре управления безопасностью в Microsoft Word необходимо выполнить команду *Файл⇒Параметры⇒Центр управления безопасностью*. В результате откроется окно (рис. 2.5), в котором необходимо выбрать требуемые параметры и нажать клавишу ОК.

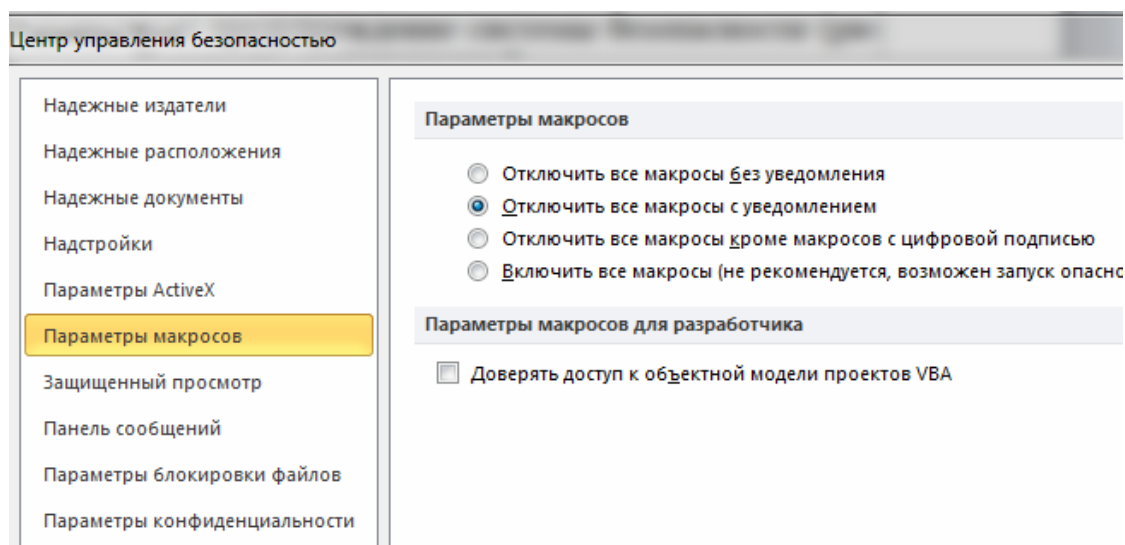


Рис. 2.5

2.4.6. Создание макросов в Excel

Создадим первый макрос в программе Microsoft Excel (версия 2010) в файле *Книга1.xlsm* под именем *Макрос1*, который вычисляет содержимое текущей ячейки как сумму двух ячеек, расположенных левее текущей ячейки и устанавливает денежный формат ячейки с точностью два знака после запятой (рис. 2.6).

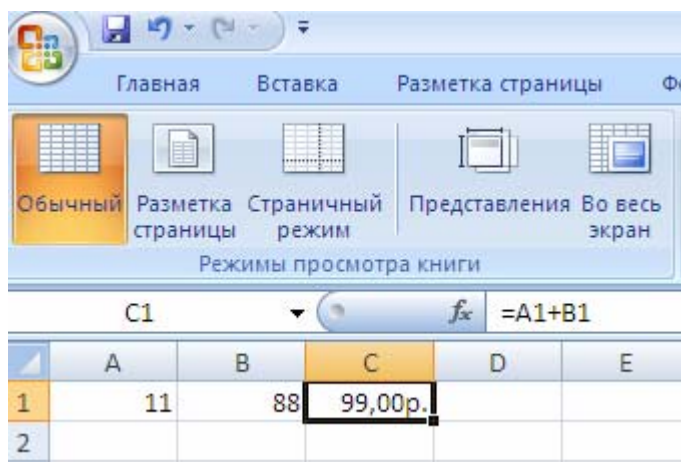


Рис. 2.6

Для записи макроса в Microsoft Excel необходимо выбрать в разделе меню *Вид* команду *Макросы* $\Rightarrow$ *Запись макроса*. В появившемся окне (рис. 2.7) будет указано имя макроса по умолчанию *Макрос1*, которое при необходимости можно изменить.

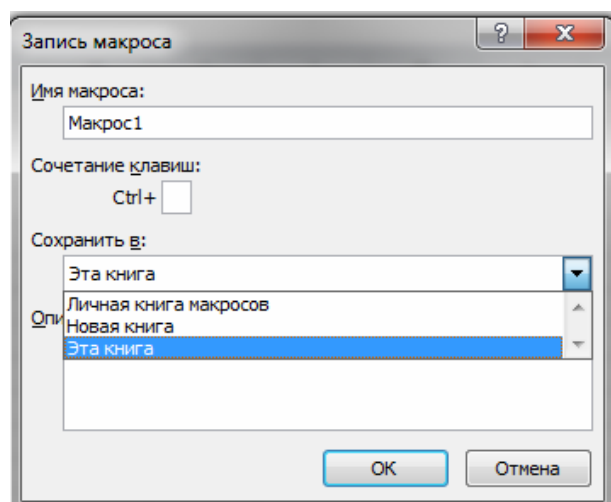


Рис. 2.7

Если макрос должен быть сохранен только в текущем файле Excel, то в списке *Сохранить в:* (см. рис. 2.7) требуется выбрать *Эта книга*. При сохранении макроса с настройкой *Личная книга макросов* он будет доступен для всех книг, так как запишется в спе-

циальный скрытый файл *Personal.xlsb*, который загружается при каждом запуске Excel. При выборе в списке строки *Новая книга* макрос запишется во вновь открытую книгу Excel.

Запуск макроса и просмотр его кода выполняется аналогично, как и в Word.

Созданный макрос помещается в модуле *Modules1* активного проекта *VBAProject (Книга1.xlsm)* (рис. 2.8).

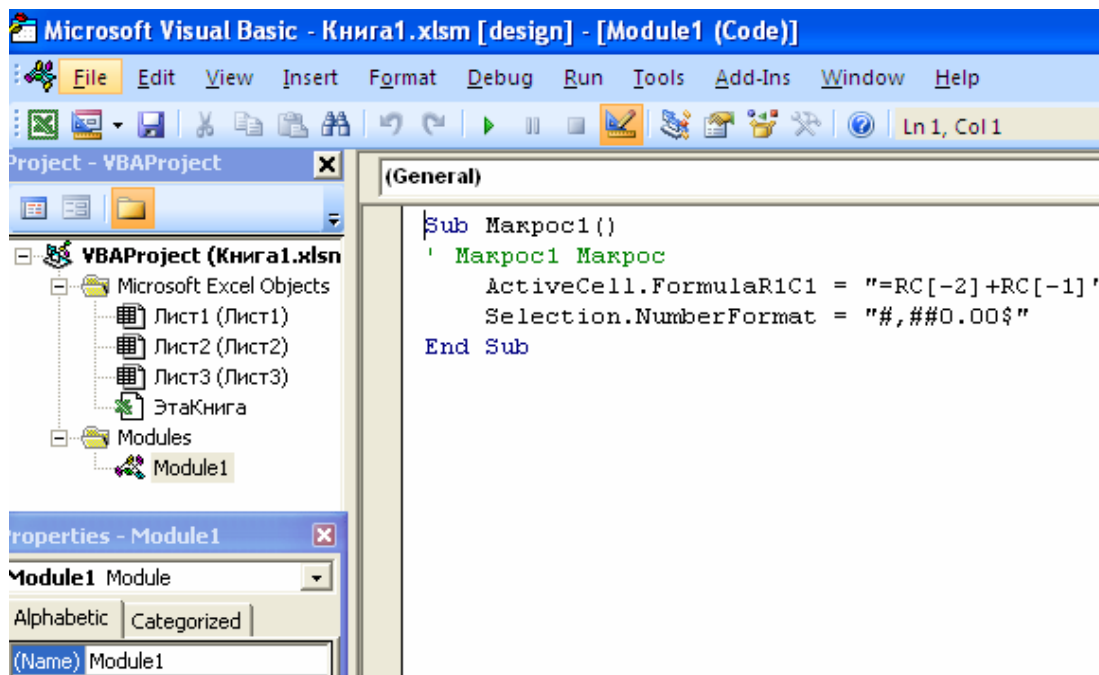


Рис. 2.8

Сопоставим действия, которые были выполнены для написания макроса, и полученный программный код макроса из окна редактирования (см. рис. 2.8).

В третьей строке формируется содержимое активной ячейки таблицы по формуле. При этом используется не прямая ссылка на ячейку, а относительная адресация через приращения по отношению к текущей строке (*Row*) и текущему столбцу (*Column*).

Символьная константа “=RC[-2]+RC[-1]”, задающая значение свойства *FormulaR1C1* активной ячейки *ActiveCell*, обозначает, что содержимое активной ячейки будет рассчитано как сумма содержимого ячеек в той же строке, с номерами столбцов на две и на одну единицу меньше номера столбца активной ячейки соответственно.

В четвертой строке устанавливается денежный формат активной ячейки с точностью два знака после запятой.

Сохранение листинга программы в отдельном файле, а также всего проекта выполняется аналогично, как и в разд. 2.4.4. Сохранить созданный макрос в текущем файле Excel.

2.4.7. Организация проекта в среде VBA

Программа на языке VBA состоит из одного или нескольких модулей. В VBA проект, помимо макросов, которые хранятся в соответствующем модуле, может состоять из нескольких форм, связанных с соответствующими программными модулями. Кроме того, в проекте могут присутствовать стандартные модули и модули классов.

Обычно модуль начинается с опций, которые управляют описанием переменных.

Инструкция *Option Explicit* используется для принудительного объявления всех переменных в области *General Declarations* (Общее описание) модуля, которая располагается в верхней части модуля.

Эта инструкция исключает возможность случайного создания новых переменных. Ее использование полезно при отладке программы, так как если в программе допущена ошибка в имени переменной, то будет выдано соответствующее сообщение об ошибке.

Затем в модуле располагается область объявления переменных и констант уровня модуля или проекта, которые могут быть использованы во всех процедурах модуля либо проекта, например:

```
Dim bytA As Byte
Dim strA1 As String
Private i As Integer
Private Среднее As Double
```

Далее располагается код процедур или функций, составляющий саму программу.

Процедура представляет собой поименованную часть кода, выполняющую определенные действия. Процедура может иметь параметры, которые в результате выполнения последовательности инструкций могут менять свои значения.

При общем описании формата (синтаксиса) команд принято использовать следующие обозначения:

[] (квадратные скобки) обозначают, что заключенная в них часть конструкции является необязательной, но если она есть, то присутствует без самих квадратных скобок;

| (вертикальная черта) разделяет возможные варианты выполнения команды.

Процедура имеет следующий синтаксис:

```
[PublicPrivate / Friend][Static] Sub name[(arglist)]
    [statements]
```

[Exit Sub]
[statements]

End Sub

где *Public* – указывает, что процедура доступна для всех других процедур во всех модулях;

Private – указывает, что процедура доступна для других процедур только того модуля, в котором она объявлена;

Friend – используется только в модулях классов. Позволяет вызывать процедуру из другого модуля проекта;

Static – указывает, что локальные переменные процедуры сохраняются в промежутках времени между вызовами этой процедуры;

arglist – список аргументов, разделенных запятой, значения которых передаются в процедуру или возвращаются из процедуры при ее вызове;

statements – любая группа инструкций, выполняемых в процедуре Sub;

Exit Sub – инструкция, которая приводит к немедленному выходу из процедуры Sub.

Программные модули могут включать в себя процедуры двух типов: *общие* и *событийные*.

Общая процедура представляет собой подпрограмму, которая начинает выполняться после ее вызова из другой процедуры. Выполнение общих процедур не связывается с какими-либо событиями, они вызываются на выполнение с помощью оператора *Call* и имеют следующий формат:

Call ИмяПроцедуры (Список параметров)

где *Список параметров* представляет собой набор переменных, значения которых устанавливаются до начала выполнения процедуры.

Событийная процедура представляет собой подпрограмму, которая начинает выполняться после реализации определенного события. В рассмотренной в следующем разделе программе Калькулятор содержатся лишь событийные процедуры, большинство из которых обрабатывает событие (*Click*), произошедшее при щелчке левой клавиши мыши на соответствующем элементе управления.

Кроме процедуры *Sub* в VBA используется и процедура *Function*, которая имеет следующий формат:

[Public /Private /Friend][Static] Function name[(arglist)]

```

statements
[name=expression]
[Exit Function]
statements
[name=expression]
End Function

```

где *name* – имя процедуры *Function*;

expression – любое выражение, возвращающее значение того же типа, что и процедура *Function*.

Инструкция *Exit Function* приводит к немедленному выходу из процедуры *Function*.

Если требуется использовать возвращаемое *Function* значение, в отличие от процедуры *Sub* процедура *Function* может применяться в правой части выражения, как и любая другая встроенная функция

```
Переменная = name()
```

В свою очередь, процедуре *Function* может присваиваться значение какого-либо выражения

```
name() = expression
```

Для вызова процедуры *Function* может использоваться так же, как и для процедуры *Sub* инструкция *Call*, когда интересует не возвращаемое процедурой значение, а осуществляемое ей действие.

В VBA возможно создание рекурсивных процедур *Function* и *Sub*, т.е. процедур вызывающих самих себя.

2.5. Основы программирования на VBA

2.5.1. Основные элементы языка программирования VBA

Программа, написанная в среде VBA, представляет собой совокупность программных строк, имеющих следующий формат:

```
[номер строки] оператор [:оператор] ... ['комментарий]
```

или

```
метка:
```

или

```
$метаоператор
```

Операторы представляют собой действия, выполняемые программой. В строке может быть один или несколько операторов, разделенных двоеточием. Длина программной строки в VBA может

быть до 249 символов. Если оператор не помещается в одной экранной строке, то его можно разбить на несколько экранных строк, поставив в конце строки символ подчеркивания (_). При этом следующая строка будет рассматриваться как продолжение предыдущей. Следует иметь в виду, что перед символом подчеркивания должен стоять пробел, если только строка не разрывается внутри слова.

Номер строки – это целое десятичное число от 0 до 65535. Номера необязательно использовать в программе, но если они есть, то используются для идентификации (распознавания) строк программы. Никаких ограничений, за исключением того, что все строки должны иметь разные номера, на взаимное расположение номеров строк не накладывается.

Метка должна быть в строке без операторов (хотя комментариев в этой строке может быть). Она служит для идентификации оператора, следующего непосредственно за ней. Метка должна начинаться с буквы и может содержать произвольное число букв и цифр. Заглавные и строчные буквы в метке не различаются. За меткой должно следовать двоеточие.

Основными элементами языка VBA являются символы, с помощью которых записываются константы, переменные, массивы, выражения, встроенные функции и операторы.

При написании программы на VBA можно использовать символы из определенного набора, называемого алфавитом языка. Он включает в себя заглавные и строчные латинские буквы от A до Z, цифры от 0 до 9, пробел, а также специальные символы: = (знак равенства), + (плюс), – (минус), * (звездочка, знак умножения), / (слэш, знак деления), \ (знак целочисленного деления), ^ (крышка, знак возведения в степень), % (знак процента), & (амперсанд), ! (восклицательный знак), # (знак номера), \$ (знак доллара), () (круглые скобки), [] (квадратные скобки), , (запятая), . (точка), ' (апостроф), ; (точка с запятой), : (двоеточие), ? (вопросительный знак), > (знак больше), < (знак меньше), " (кавычки), _ (подчеркивание). Кроме того, в комментариях, символьных константах и именах переменных могут использоваться заглавные и строчные буквы кириллицы (русские буквы).

2.5.2. Типы данных, константы и переменные

Программы на VBA используют различные типы данных, которые определяют область возможных значений данного типа, структуру организации данных и разрешенные операции для данных этого типа.

VBA разделяет обрабатываемые данные на числа, даты, строки, логические значения и объекты. Подробный список типов данных в VBA можно посмотреть, например, в табл. 4.1 самоучителя [9]. В данном пособии рассмотрены наиболее распространенные типы данных, используемых в VBA.

Среди *числовых* данных выделим следующие типы:

– *целое Integer* (число без десятичной точки в диапазоне от –32768 до 32767);

– *длинное целое Long* (в диапазоне от –2 147 483 648 до 2 147 483 647);

– *вещественное Single* (число с плавающей точкой одинарной точности):

для отрицательных чисел в диапазоне от $-3,402823 \cdot 10^{38}$ до $-1,401298 \cdot 10^{-45}$;

для положительных чисел в диапазоне от $1,401298 \cdot 10^{-45}$ до $3,402823 \cdot 10^{38}$.

Внимание. В VBA целая часть числа отделяется от дробной точкой!

Любые *строковые (текстовые) данные* в языке VBA используют тип данных *String*. Они всегда заключаются в кавычки (“ ”).

По умолчанию все строковые данные относятся к строкам переменной длины. Для объявления этого типа используют оператор *Dim*:

Dim имя_переменной As String

Если в программе требуется использовать строки фиксированной длины (от 1 до 65 400 символов), то необходимо использовать оператор *Dim* следующей конструкции:

*Dim имя_переменной As String * длина_строки*

Если в VBA переменная была создана без указания типа, то ей назначается тип *Variant*. Однако при этом переменная будет занимать много места в памяти компьютера, и программа будет выполняться дольше.

Константы – это элементы языка VBA, значения которых определены и неизменны. Они бывают *числовыми* и *символьными*.

Числовые константы, в свою очередь, могут быть целыми и вещественными. Они используют цифры, знак минус и необязательный знак плюс. *Целые* числа, например число 257, допускается записывать в двоичном (&B100000001), восьмеричном (&401) и в шестнадцатеричном (&H101) представлении.

Вещественные константы можно задавать как в десятичном формате (0.000235988), так и в формате с плавающей точкой (235.988E-6).

Символьные константы в VBA могут быть *литеральные* и *символические*. Значение литеральной символьной константы задается прямо в тексте программы в кавычках (“Пример”). Изменить его можно только при редактировании программы.

Символические константы, так же как и переменные, должны быть объявлены, прежде чем их использовать в любом месте программы. Тип данных константы при ее объявлении указывать не обязательно:

```
Dim НомерЗадачи As String
Dim Задача As String
Const НомерЗадачи = 2
Const Задача = “Решение системы уравнений”
```

Константы, объявленные в процедуре, могут использоваться только в ней. При определении константы в области определений модуля она будет доступна всем процедурам данного модуля.

Переменные – это элементы языка для хранения данных указанного типа в поименованной области компьютера. Значения переменных заранее не определены, но в процессе выполнения программы им можно присваивать значения констант, других переменных либо выражений. Переменные разделяются на *числовые* переменные и *символьные* (строковые). Числовой переменной можно присвоить значение любого числа, а символьной – значение строки символов.

Имя переменной должно быть уникальным и начинаться с буквы, хотя может содержать и цифры. Заглавные и строчные буквы не различаются. Длина имени переменной не должна превышать 255 символов. Зарезервированные слова не могут применяться в качестве имен переменных. Кроме того, имя переменной не может начинаться с букв *FN*, которые используются в начале имени пользовательской функции.

Для объявления переменной используется оператор *Dim*, который резервирует в памяти компьютера поименованную область для хранения данных указанного типа. При этом тип переменной определяется типом данных, хранящихся в ней:

```
Dim имя_переменной As тип_данных
```

Переменные, объявленные в процедуре, могут использоваться только в ней. При определении переменных в области определений

модуля с помощью ключевых слов *Private* и *Dim* они будут доступны всем процедурам данного модуля. Если при объявлении переменных в области определений модуля использовалось ключевое слово *Public*, то она будет доступна для всех модулей во всех проектах.

Массив – это последовательность данных одинакового типа, имеющих общее имя. Элемент массива является отдельной переменной. Он идентифицируется путем указания в скобках после имени для одномерного массива индекса его элемента, а для многомерного массива разделенных запятой индексов его элемента.

Для объявления массива необходимо указать его размерность. При этом в VBA по умолчанию индексы элементов массива начинаются с нуля. Для удобства можно изменить начальный индекс элементов массива с помощью оператора *Option Base*. Определим, например, матрицу A(i, j) размерностью пять на пять, элементы которой имеют тип данных *Integer*:

Dim A (1 to 5, 1 to 5) As Integer

2.5.3. Арифметические выражения и функции VBA

Выражения в VB состоят из элементов и операций, связывающих их. В одном выражении нельзя использовать несовместимые типы данных. Элементами выражений могут быть константы, переменные и функции.

Функции в VBA реализуют стандартные математические функции, функции преобразования данных, функции даты и времени, строковые функции, а также некоторые функции самих host-приложений Word и Excel [8, с. 151].

Элементы числовых выражений могут связываться между собой с помощью арифметических и логических операций, важнейшие из которых приведен в табл. 2.4 в порядке убывания их приоритета (устанавливающего порядок выполнения операций в выражении).

Операции с одинаковым приоритетом выполняются в порядке просмотра выражения слева направо. Для изменения порядка выполнения операций в выражении можно использовать скобки.

Все арифметические операции имеют традиционную форму записи для математики.

Запись $31 \text{ MOD } 4$ обозначает остаток от целочисленного деления 31 на 4 и возвращает в результате число 3.

Таблица 2.4

Обозначение операции	Наименование операции
$\wedge$	Возведение в степень
$-$	Унитарная операция минус (для обозначения отрицательных чисел или изменения знака)
$*$, $/$	Умножение, деление
$\backslash$	Целочисленное деление
<i>MOD</i>	Вычисление остатка от деления
$+$, $-$	Сложение, вычитание
$<$, $<=$, $=$, $>=$, $>$	Операции сравнения
$<>$	Операция не равно
<i>NOT</i>	Логическое НЕ
<i>AND</i>	Логическое И
<i>OR</i>	Логическое ИЛИ

Результатом операции сравнения является *True* (истина или 1) либо *False* (ложь или 0).

В VBA есть ряд встроенных функций, упрощающих вычисления и операции, которые включают в себя следующие группы [11]:

- математические функции;
- функции преобразования форматов;
- функции проверки типов;
- функции обработки строк;
- функции даты и времени.

Все функции подробно рассмотрены в справочной системе VBA, а также в [8, 11].

Чтобы получить список функций VBA при написании кода в окне редактирования, введите *VBA* и точку (.).

Особое место среди встроенных функций VBA занимают функции, которые не только возвращают одно значение, но и отображают диалоговое окно, в котором пользователь может выполнять определенные действия либо выводить сообщения.

В VBA также имеется встроенное диалоговое окно ввода *InputBox*, которое позволяет ввести данные в программу через *текстовое поле на диалоговой панели*. Соответствующая функция имеет следующий синтаксис:

InputBox (*Приглашение* \$, *Заголовок* \$, [*По Умолчанию* \$])

В процессе выполнения этой функции появляется диалоговая панель с текстовым полем. В строке заголовка панели будет печат-

таться значение второго аргумента *Заголовок\$*, на самой панели печатается значение аргумента *Приглашение\$*, а в текстовом поле печатается значение аргумента *По Умолчанию\$* (если это значение отсутствует, то содержимое текстового окна также отсутствует).

Введенная пользователем в текстовое поле строка становится значением функции.

На рис. 2.9 показано диалоговое окно, которое появится на экране в результате выполнения следующего программного кода:

$$x = \text{InputBox}("x = ", "Введите значение")$$

и введения в текстовое поле значения «1» с клавиатуры.

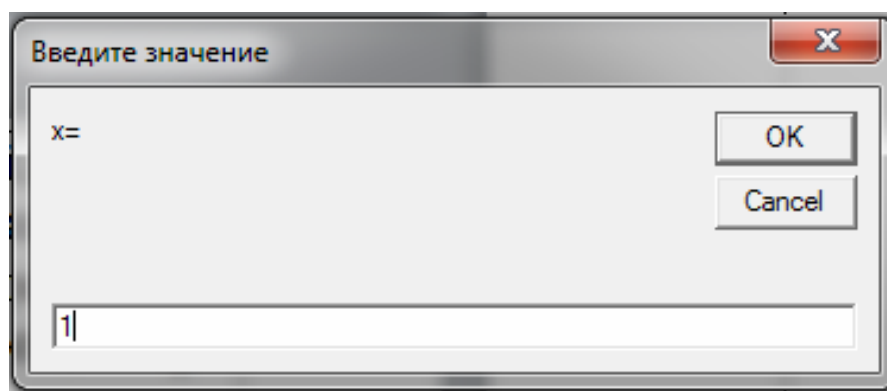


Рис. 2.9

После нажатия клавишу *OK* переменной *x* присвоится значение, равное строковой константе «1».

Для преобразования строковых значений (*x*), вводимых в текстовые поля, в десятичное число необходимо использовать функцию *Val(x)*. При этом строка *x* должна содержать только цифры и одну десятичную точку, иначе VBA не может преобразовать ее в число и возвращает значение 0.

С помощью функции *MsgBox* можно добавлять в проект *панели сообщений*. В простейшем случае функция *MsgBox* работает в режиме оператора и используется для вывода сообщений на специальной панели сообщений. Синтаксис функции в режиме оператора

$$\text{MsgBox } \text{Сообщение\$} [, \text{ЧислоКод1} + \text{ЧислоКод2}] [, \text{Заголовок\$}]$$

В результате выполнения данной команды на экране появится панель сообщений со строкой *Сообщение\$*, в строке заголовка будет выведено *Заголовок\$*, а аргумент *ЧислоКод1 + ЧислоКод2* определяет внешний вид панели.

С помощью одного числа, являющегося суммой чисел *ЧислоКод1* и *ЧислоКод2*, можно одновременно установить определенную пиктограмму и определенную комбинацию кнопок, размещенных на панели сообщений. Значения чисел *ЧислоКод1* и *ЧислоКод2* приведены в табл. 2.5 и 2.6.

Таблица 2.5

<i>ЧислоКод1</i>	Пиктограмма
16	
32	
48	
64	

Таблица 2.6

<i>ЧислоКод2</i>	Набор кнопок
0	<i>OK</i>
1	<i>OK, Cancel</i>
2	<i>Abort, Retry, Ignore</i>
3	<i>Yes, No, Cancel</i>
4	<i>Yes, No</i>
5	<i>Retry, Cancel</i>

Например, число 36 можно рассматривать как сумму числа 4 (код комбинации кнопок Yes, No) и числа 32 (код пиктограммы  «Вопрос»).

Для вывода на панели сообщений с заголовком "*MsgBox в режиме оператора*" сообщения "*Ввод закончен*" (рис. 2.10) требуется набрать следующий программный код:

MsgBox "Ввод закончен", 48, "MsgBox в режиме оператора"

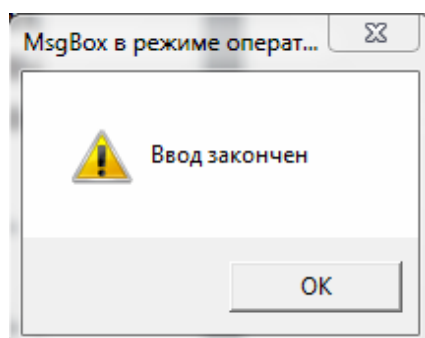


Рис. 2.10

В случае записи аргументов функции *MsgBox* в скобках функция возвращает, в зависимости от нажатой клавиши, одно из значений из табл. 2.7.

Возвращаемое функцией *MsgBox* значение может быть использовано в программе, что позволяет организовывать интерактивный режим работы программы.

Таблица 2.7

Нажатая кнопка	Возвращаемое значение
<i>OK</i>	1
<i>Cancel</i>	2
<i>Abort</i>	3
<i>Retry</i>	4
<i>Ignore</i>	5
<i>Yes</i>	6
<i>No</i>	7

2.5.4. Основные операторы языка программирования VBA для создания линейных и разветвляющихся программ

Наименьшей единицей программного кода VBA является *оператор*.

Оператор присваивания значения имеет следующий формат:

$$a = b,$$

где *a* – имя числовой, символьной или индексной переменной (должно стоять слева от знака равенства), которой присваивается значение;

b – выражение, значение которого присваивается переменной *a*, причем тип выражения должен соответствовать типу переменной (для числовой переменной числовое выражение, а для символьной – символьное выражение).

Для переменных типа объект при ссылке на него необходимо добавить ключевое слово *Set*, например

$$\text{Set obj1} = \text{obj2}$$

При необходимости в VBA можно использовать оператор безусловного перехода *GoTo*, который изменяет последовательность выполнения операторов программы.

Он имеет следующий формат:

$$\text{GoTo Метка}$$

Для организации разветвляющихся программ в VBA используют, в частности, оператор условного перехода *If...Then*. Существуют два его варианта: *строчный* и *блочный*.

Строчный оператор *If...Then* имеет следующий формат:

If условие1 [And условие2 ...] Then операторы [Else операторы]

где *условие1* (*условие2*) – это условное выражение;

операторы – оператор или последовательность разделенных двоеточием выполняемых операторов.

Если *условие1* истинно (*True*), то выполняются операторы, следующие за *Then*, после чего выполнение строчного оператора *If...Then* прекращается и программа переходит к следующему оператору.

Если *условие1* ложно (*False*), то операторы за *Then* пропускаются и выполняются операторы за *Else*, если они есть, затем программа также переходит к следующему оператору.

После *If* может проверяться выполнение нескольких условий:

условие1 And условие2

Открыть файл с макросами, созданный в Word, и перейти в среду IDE, нажав клавиши *Alt+F11*. Войти в модуль *NewMacros* проекта *Project* текущего файла. Перейти в окно редактирования и записать в нем следующий программный код:

```
Sub Макрос3()  
x = InputBox("Введите 1 – если строчный, 2 – если Блочный ", _  
"Введите значение")  
If x = 1 Then x = "Строчный If...Then" Else x = "Блочный If...Then"  
MsgBox x  
End Sub
```

Обратить внимание на то, что строчный оператор *If...Then* и все связанные с ним операторы, включая следующие за *Else*, должны быть в одной программной строке.

При запуске на исполнение макроса *Макрос3* на экране появится диалоговое окно. На рис. 2.11 показано это окно, после введения в текстовое поле значения «1» с клавиатуры.

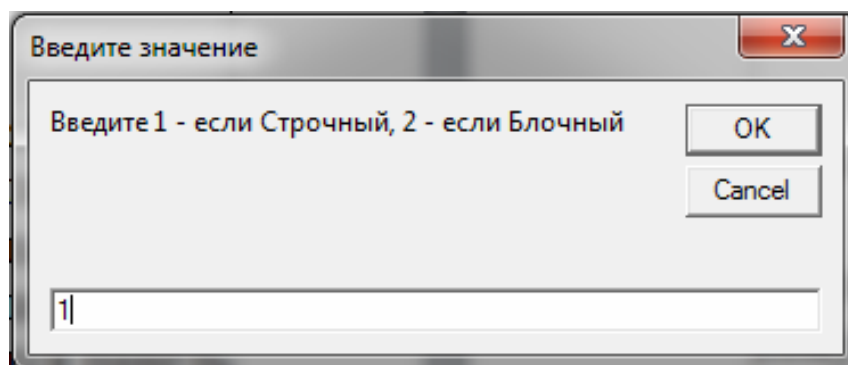


Рис. 2.11

Если нажать на клавишу *OK*, то на экране появится панель сообщений (рис. 2.12).

Если же в текстовое поле ввести с клавиатуры цифру 2 и нажать на клавишу *OK*, то на экране появится панель сообщений (рис. 2.13).

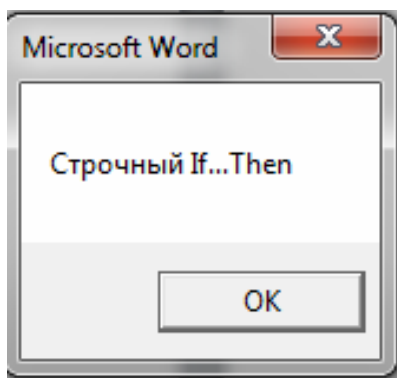


Рис. 2.12

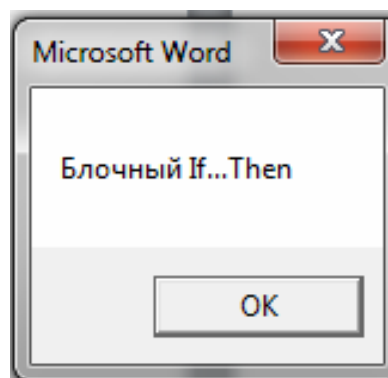


Рис. 2.13

В VBA допускается вложенность операторов *If...Then*, ограниченная только одной строкой, при этом оператор *Else* соответствует последнему не закрытому оператору *Then*.

Если в операторе *If...Then* после слова *Then* помещаются несколько операторов, то удобнее использовать *блочный* оператор *If...Then*, имеющий следующий формат:

```
If условие1 Then  
    ... операторы  
    [ElseIf условие2 [ ... ] Then  
    ... операторы ]  
    ...  
    [Else  
    ... операторы ]  
End If
```

При выполнении этого блока сначала проверяется истинность *условие1*. Если оно ложно, то по порядку проверяются условия в каждом из следующих за ним операторов *ElseIf* (их может быть сколько угодно).

Как только находится истинное условие, выполняются операторы, следующие за соответствующим оператором *Then*, после чего выполнение блока *If...Then* прекращается и управление передается на оператор, следующий за оператором *End If*.

Если никакое из проверенных условий *ElseIf* несправедливо, то выполняются операторы за *Else*, если они есть, после чего выполнение блока *If...Then* прекращается.

В блочном операторе *If...Then* после ключевых слов *Then* и *Else* не должно быть других операторов. *If*-блок может быть вложенным, т.е. любой из операторов, следующий за всяким *Then* или *Else*, может содержать другой *If*-блок.

Рассмотрим примеры использования строчного и блочного операторов *If...Then*. Определить, чему будет равна переменная *A* после выполнения оператора *If...Then*.

При использовании строчного оператора *If...Then*, программный код будет иметь следующий вид:

```
If x > y THEN A = A + 1 Else If z < y AND z > x Then A = A + 2 Else A = A * 2
```

Для реализации того же самого алгоритма, но с помощью блочного оператора *If...Then* требуется записать следующий программный код:

```
If x > y THEN  
    A = A + 1  
    ElseIf z < y AND z > x Then  
        A = A + 2  
    Else  
        A = A * 2  
End If
```

Если $A = 4$, $z = 1$, $x = 2$, $y = 3$, то в результате выполнения как первого, так и второго программного кода переменная *A* будет равна 8.

2.5.5. Операторы VBA для создания циклических программ

Часто при составлении программ приходится повторять некоторую часть программы *заданное число раз*. Пример такого циклического алгоритма приведен на рис. 1.13,б. Он предназначен для вычисления и вывода на экран значения функции $Y(x) = \sin x$ для значений аргумента x , изменяющегося в интервале от 0 до 20 с шагом 2.

В VBA для организации цикла с заданным числом повторений используется оператор *For...Next*, имеющий следующий формат:

```
For переменная = x To y [Step z]  
    ... операторы  
Next [переменная [, переменная] ...]
```

где *переменная* – числовая переменная, используемая как счетчик повторов; x , y , z – числовые выражения, определяющие соответственно начальное, конечное значение и приращение счетчика.

Оператор *For* задает начальное *x* и конечное *y* значение *переменной* цикла, а также шаг *z* (по умолчанию *1*), с которым она будет изменяться.

Конструкция выполняет операторы, следующие за оператором *For*, пока в программе не встретится оператор *Next*. Тогда к текущему значению *переменной* цикла прибавляется значение шага *z* и полученное значение сравнивается с конечным значением *y*.

Если значение *переменной* цикла больше конечного значения *y*, то выполняется оператор, следующий за оператором *Next*, иначе управление снова передается оператору, находящемуся за оператором *For*.

Например, процедура, реализующая алгоритм, приведенный на рис. 1.13,б, с помощью оператора *For...Next* имеет следующий вид:

```
Sub Макрос6()  
  For x = 0 To 20 Step 2  
    y = Sin(x)  
    y = Round(y, 2)  
    Ответ = Ответ + CStr(y) + " "  
  Next x  
  MsgBox Ответ  
End Sub
```

Результат работы программы показан на рис. 2.14.

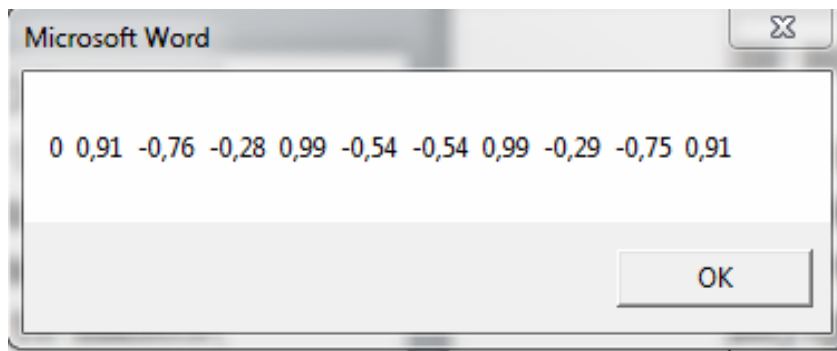


Рис. 2.14

В четвертой строке процедуры после вычисления значения функции *y* для текущего значения *переменной* цикла *x* происходит округление полученного значения до двух знаков после запятой с помощью функции *Round*.

В пятой строке путем накапливания формируется строковая константа *Ответ*, включающая в себя 11 значений функции *y*, разделенных пробелами.

Циклы с неопределенным количеством повторений позволяют задавать конструкцию *While...Wend*, в которой указывается условие продолжения цикла:

```
While x
    ...[операторы]
Wend
```

где x – числовое выражение.

Цикл повторяется до тех пор, пока выражение x не станет равным нулю (*False*). При $x = 0$ все операторы до оператора *Wend* включительно пропускаются, и программа выполняется дальше.

Например, если в процедуре *Макрос6* для организации цикла использовать оператор *While...Wend*, то программа будет иметь вид

```
Sub Макрос7()
    x = 0
    While x <= 20
        y = Sin(x)
        y = Round(y, 2)
        Ответ = Ответ + CStr(y) + " "
        x = x + 2
    Wend
    MsgBox Ответ
End Sub
```



Изучите самостоятельно наиболее общую форму организации циклов с помощью операторов *Do...Loop*, которая позволяет проверять условие окончания цикла в начале или в конце цикла [8, 10].

Циклы могут быть вложенными. Всякий оператор *For (Wend)* относится к ближайшему оператору *Next (While)*.

Рассмотрим пример программы, вычисляющей и выводящей на экран монитора значения функции

$$A = \prod_{k=1}^n \sqrt{x^2 + k^2}$$

при x , изменяющемся от 0 до 5 с шагом 0,5 и для любого значения n . Соответствующая схема алгоритма приведена на рис. 1.24. Она может быть реализована с помощью макроса *Макрос8*:

```

Sub Макрос8()
n = InputBox("Введите значение n")
n = Val(n)
  For x = 0 To 5 Step 0.5
    A = 1
    For k = 1 To n
      A = A * Sqr(x ^ 2 + k ^ 2)
      A = Round(A, 2)
    Next k
    MsgBox "x = " + CStr(x) + " A = " + CStr(A)
  Next x
End Sub

```

В результате выполнения данного макроса на экране сначала появится диалоговое окно, куда необходимо ввести требуемое значение переменной n , например равное «2», а затем панель сообщений (рис. 2.15,а) с результатом вычисления функции A для аргумента $x = 0$. После нажатия на клавишу ОК появится следующая панель сообщений (рис. 2.15,б) и т.д., всего 11 панелей.

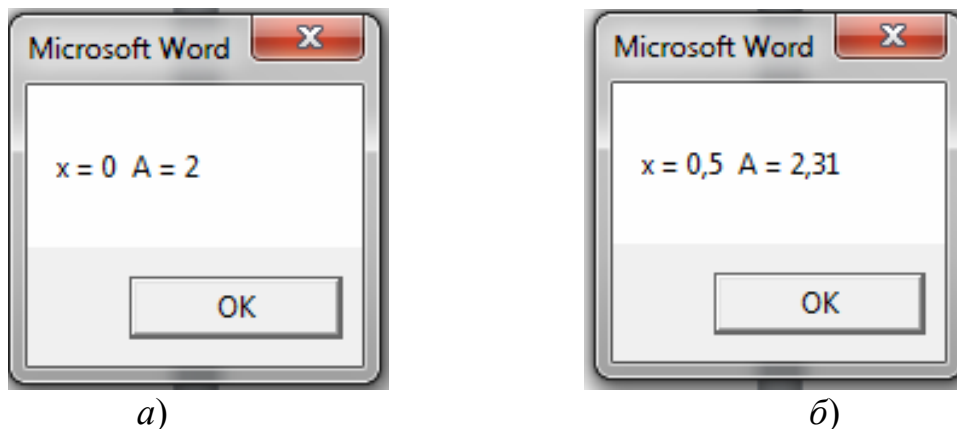


Рис. 2.15

Обратите внимание, что вычисление произведения A организуется в цикле путем последовательного перебора и умножения отдельных сомножителей как $A = A * f(x, k)$. При этом, когда вычисляется первое произведение, переменная A должна равняться «1», т.е. перед внутренним циклом необходимо присвоить переменной A значение, равное единице.

Если необходимо организовать вычисление суммы, $A = \sum_{k=1}^n f(x, k)$, то в цикле необходимо последовательно перебирать и складывать отдельные слагаемые по формуле $A = A + f(x, k)$.

При этом первая сумма получается путем сложения первого слагаемого с нулем, т.е. перед внутренним циклом необходимо присвоить переменной *A* значение, равное нулю.

Если несколько циклов имеют общий конечный оператор, можно указать для них один оператор *Next*, перечислив в нем переменные циклов в порядке, обратном порядку операторов *For*.

2.5.6. Циклические программы по обработке массивов

Для работы с массивами также удобно использовать циклические программы. Например, процедура *Макрос9*, присваивающая с помощью окна ввода значения элементам матрицы *A* размерностью 2 на 3 и выводящая затем матрицу на панель сообщения, имеет вид

```
Sub Макрос9()  
    Dim i As Integer: Dim j As Integer  
    Dim A(2, 3) As Integer  
    Dim Матрица As String  
    Матрица = ""  
    For i = 1 To 2  
        For j = 1 To 3  
            Aij = "A(" + CStr(i) + "," + CStr(j) + ")"  
            A(i, j) = InputBox (Aij, "Введите значения элементов матрицы")  
            Матрица = Матрица + CStr(A(i, j)) + vbTab  
        Next j  
        Матрица = Матрица + vbLf  
    Next i  
    MsgBox Матрица  
End Sub
```

В результате выполнения данного макроса на экране последовательно выводятся шесть диалоговых окон (рис. 2.16), в которых указываются индексы элементов матрицы $A(i, j)$, для которых необходимо ввести требуемое значение. Затем появляется панель сообщений (рис. 2.17), в которой помещаются все введенные значения элементов матрицы $A(i, j)$ в виде матрицы.

Во внутреннем цикле процедуры *Макрос9* для вывода элементов по столбцам (через равные интервалы) при формировании текущего значения строковой константы *Матрица* в конце добавляет-

ся системная переменная *vbTab* (знак табуляции), которая добавляет в конце константы недостающие пробелы. Что позволяет вывести все значения в строке матрицы через равные интервалы.

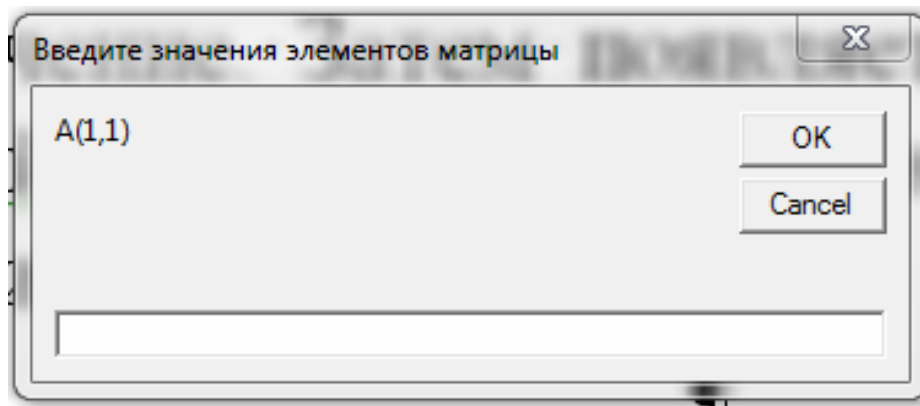


Рис. 2.16

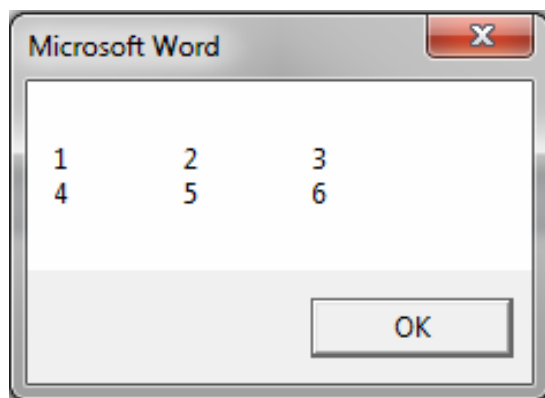


Рис. 2.17

Во внешнем цикле процедуры *Макрос9* для вывода элементов по строкам при формировании текущего значения строковой константы *Матрица* в конце добавляется системная переменная *vbLf*, которая выполняет перевод строки.

Для присвоения произвольных значений элементам матрицы можно использовать функцию генератора случайных чисел *RND*, которая возвращает случайные положительные числа из диапазона от 0 до 1. *Макрос9а* автоматически формирует и выводит в панель сообщений матрицу, значения элементов которой – случайные числа из диапазона от –50 до 50:

```
Sub Макрос9a()  
Dim i As Integer: Dim j As Integer: Dim A(2, 3) As Integer  
Dim Матрица As String: Матрица = ""  
For i = 1 To 2  
    For j = 1 To 3
```

```

    A(i, j) = (Rnd - 0.5) * 100
    Матрица = Матрица + CStr(A(i, j)) + vbTab
    Next j
    Матрица = Матрица + vbLf
    Next i
    MsgBox Матрица
End Sub

```

В результате выполнения данного макроса в панели сообщений (рис. 2.18) выводятся значения элементов матрицы $A(i, j)$ в виде матрицы.

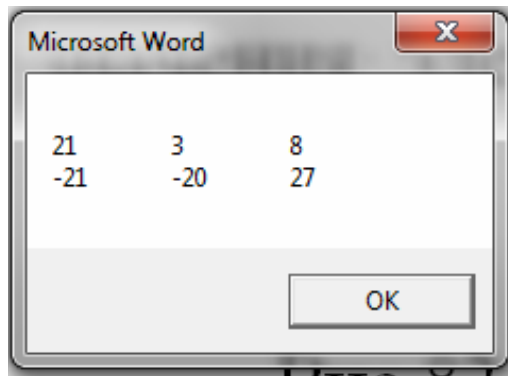


Рис. 2.18

2.6. Использование форм и элементов управления VBA для организации интерфейса программ

2.6.1. Форма и элементы управления VBA

Основным объектом в VBA является *форма*, представляющая собой окно приложения, в котором можно разместить различные элементы управления. Форма может быть добавлена в проект выбором команды *Insert*→*UserForm* или с помощью кнопки *UserForm* на стандартной панели инструментов в редакторе VBA (рис. 2.19). В результате на экране появляется сама форма, которая добавляется в проект с именем *UserForm1* по умолчанию (см. рис. 2.19). Если в проект добавить следующую форму, то она будет иметь имя *UserForm2* и т.д.

Размеры формы можно менять, перетаскивая мышкой ее правую или нижнюю границу. Как и любой объект в VBA, форма обладает свойствами, методами и событиями. Наиболее часто используемые свойства, методы и инструкции форм приведены в табл. П.А.1–П.А.3.

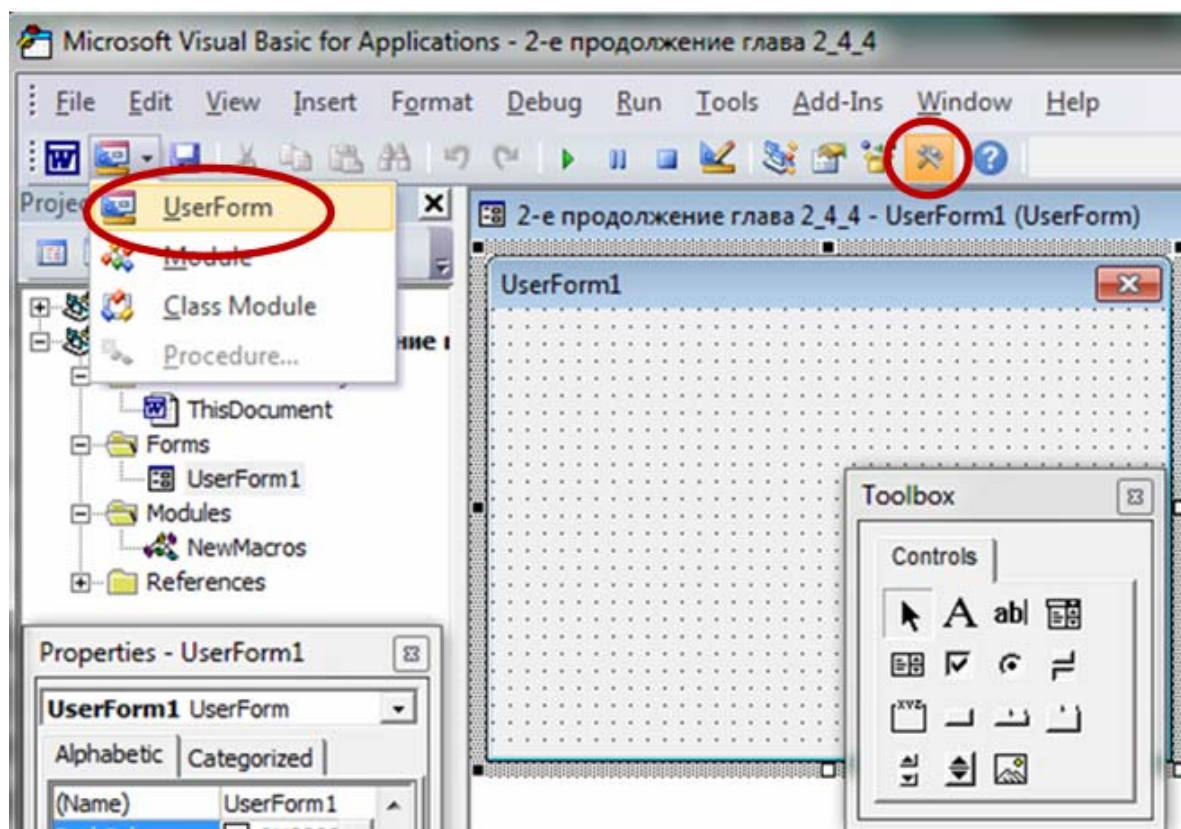


Рис. 2.19

Первоначально форма пуста, в дальнейшем в процессе создания графического интерфейса приложения в ней размещаются *элементы управления*. Используя эти элементы, нетрудно создавать любой пользовательский интерфейс в среде Windows. Пиктограммы элементов управления помещаются на *Панели элементов управления Toolbox*, которая либо появляется одновременно с появлением формы, либо может быть вызвана с помощью команды *View→Toolbox→Edit* или с помощью кнопки  на стандартной панели инструментов в редакторе VBA (см. рис. 2.19).

Список основных элементов управления VBA приведен в табл. П.А.4. Все кнопки панели инструментов, за исключением первой, служат для создания элементов управления. Первая кнопка называется *Указатель (Point)*; щелкнув по нему, можно выбрать уже созданный в форме элемент управления, изменить его размер или переместить. Данный режим конструирования формы включается автоматически после размещения элементов управления в форме. Для размещения элементов управления в форме необходимо, выбрав щелчком мыши элемент, поместить его в нужное место на форму проектируемого приложения. После этого элемент управления можно перемещать, изменять его размеры, копировать в буфер обмена и вставлять из буфера обмена. Для установки свойств эле-

ментов управления вручную при его конструировании необходимо его выделить и нажать кнопку *Properties*.

Все элементы управления формы образуют семейство *Controls*. Общий способ обращения к любому элементу управления в семействе *Controls* осуществляется путем указания его индекса:

Controls (номер элемента управления в проекте)

Индексы объектов *Control* начинаются с 0, затем 1 и т.д.

С формой связывается программный модуль, содержащий событийные и общие процедуры.

У каждого элемента управления есть целый набор событий, связанных с теми или иными действиями. Некоторые общие для форм и элементов управления события приведены в табл. П.А.5.

Для вывода программного модуля на экран можно воспользоваться контекстным меню, выбрав там команду *View→Code* или просто выполнить двойной щелчок по форме клавишей указания мыши, а для возвращения в окно форм необходимо выбрать в контекстном меню команду *Hide*, либо выбрать требуемую форму в окне проектов. Кроме того, любое из открытых в данном проекте окон может переключаться и изменяться средствами Windows.

2.6.2. Пример создания приложения в Microsoft Word

Зайти в среду VBA и создать приложение *Калькулятор*. Виртуальный калькулятор должен выполнять четыре арифметических действия – сложение, вычитание, умножение и деление.

Создание приложения включает в себя три этапа:

- виртуальное программирование графического интерфейса калькулятора;

- задание значений свойствам объектов графического интерфейса калькулятора;

- создание программного кода, связанного с элементами управления на форме.

1. Вставить в проект форму и разместить на ней необходимые для функционирования приложения элементы управления (рис. 2.20):

- первых три текстовых поля (объекты *TextBox1*, *TextBox2*, *TextBox3*): два поля для ввода числовых данных и третье – для вывода результатов вычислений;

- четыре кнопки (объекты *CommandButton1*, *CommandButton2*, *CommandButton3*, *CommandButton4*).

Расположить управляющие элементы в форме, таким образом, как это показано на рис. 2.20.

2. Для изменения свойств объектов вызвать окно свойств *Properties*. Последовательно выделяя каждый объект, включая саму форму, задать для них значения свойства *Caption* (Надпись) в соответствии с табл. 2.8.

В первом столбце таблицы в скобках указаны имена объектов по умолчанию – свойство *Name* (Имя).

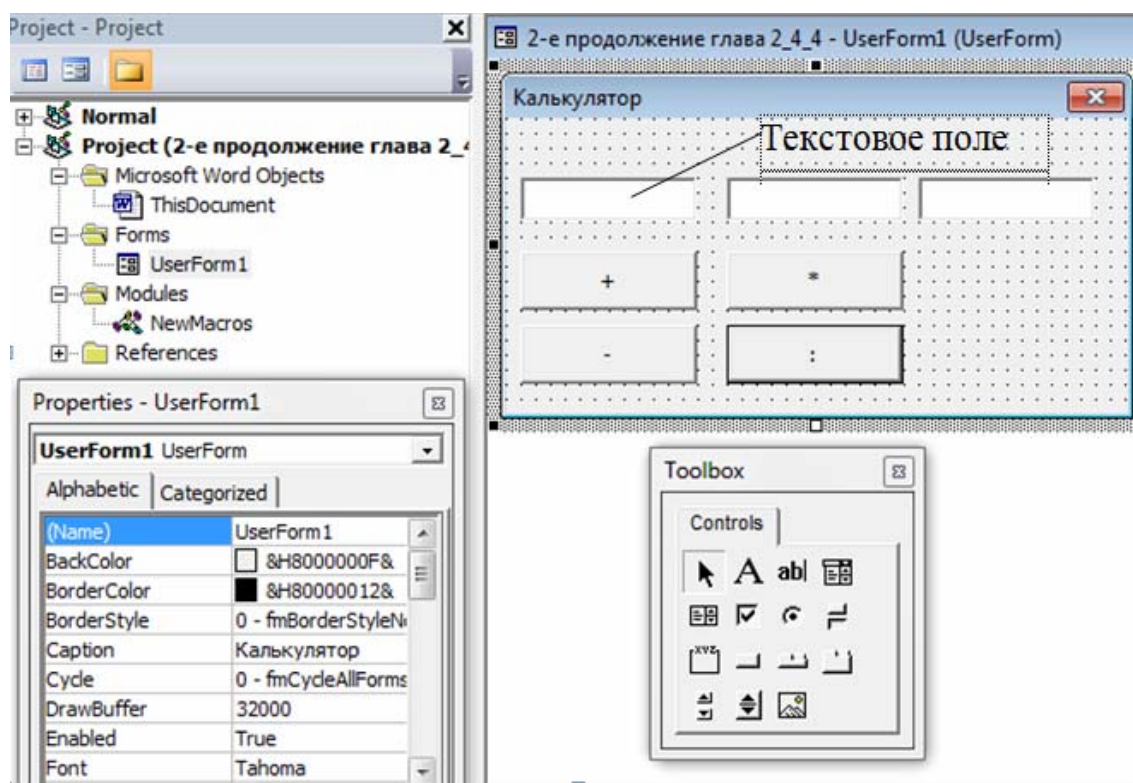


Рис. 2.20

Таблица 2.8

Класс объектов (свойство <i>Name</i>)	Значение свойства <i>Caption</i> по умолчанию	Новое значение свойства <i>Caption</i>
Форма (<i>UserForm1</i>)	<i>UserForm1</i>	<i>Калькулятор</i>
Кнопка (<i>CommandButton1</i>)	<i>CommandButton1</i>	+
Кнопка (<i>CommandButton2</i>)	<i>CommandButton2</i>	–
Кнопка (<i>CommandButton3</i>)	<i>CommandButton3</i>	*
Кнопка (<i>CommandButton4</i>)	<i>CommandButton4</i>	/

Для объектов *TextBox1* – *TextBox3* изменить значения свойства *Font*, которое задает нужный шрифт, начертание и размер шрифта, размещаемого в текстовом поле. Для этого выделить каждое текстовое окно, найти для него в окне свойств *Properties* свойство *Font* и

навести курсор на правую часть соответствующей ячейки второго столбца таблицы (рис. 2.21). В результате раскроется диалоговое окно *Шрифт*, в котором установить требуемые параметры шрифта.

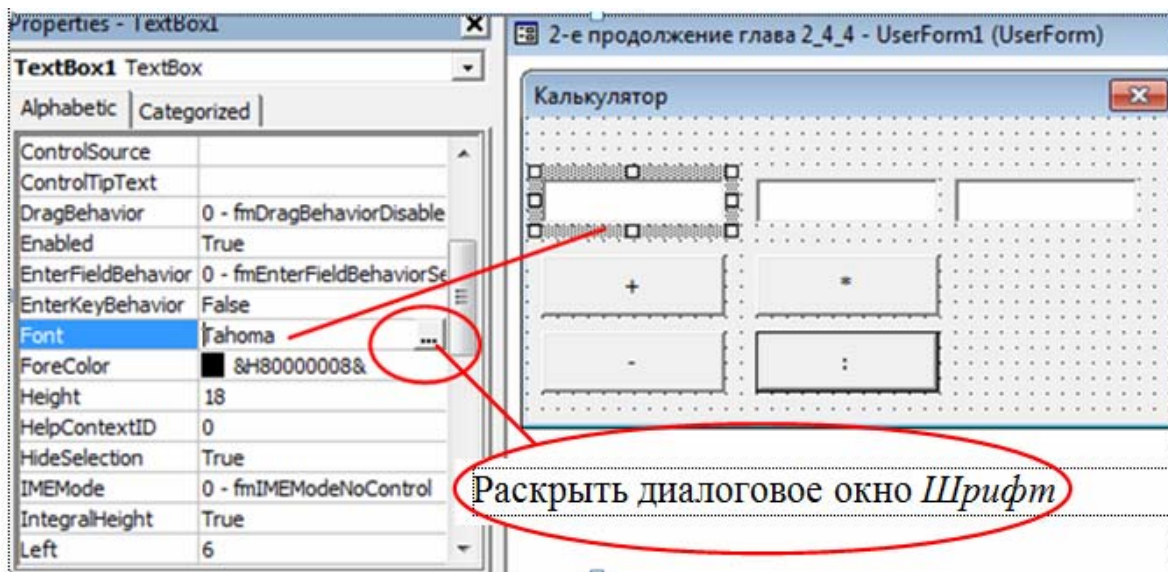


Рис. 2.21

3. На этапе создания программного кода происходит кодирование событийных процедур в приложении. В рассмотренном приложении *Калькулятор* содержатся лишь событийные процедуры, обрабатывающие событие (*Click*), произошедшее при щелчке левой клавиши мыши на соответствующем элементе управления.

Процедура сложения реализует сложение двух чисел. Эта процедура должна изменить свойство *Text* объекта *TextBox3* так, чтобы оно явилось суммой числовых значений свойства *Text* объектов *TextBox1* и *TextBox2*. Эта процедура должна вызываться, если пользователь щелкнет по кнопке с надписью «+». Для ее написания необходимо выполнить двойной щелчок по кнопке «+», в результате откроется окно редактора кода, в котором будет помещена заготовка для процедуры обработки события *Click* объекта *CommandButton1*:

```
Private Sub CommandButton1_Click()
End Sub
```

Внутри процедуры необходимо с клавиатуры добавить команду:

```
TextBox3.Text=Val(TextBox1.Text)+Val(TextBox2.Text)
```

Функция *Val* используется в процедуре для преобразования строковых значений, вводимых в текстовые поля, в десятичное число.

При написании программ целесообразно проверять работоспособность процедур по мере их написания. Для этого надо нажать клавишу *F5* или кнопку  или выбрать команду *Run→Run Sub*.

В результате на экране появится форма (рис. 2.22). Ввести любые числовые значения в первое и второе текстовые окна, например «100» и «22» на рис. 2.22. После нажатия на кнопку «+» в третьем окне появится числовое значение, равное сумме чисел из первых двух текстовых окон – «122» на рис.2.22.

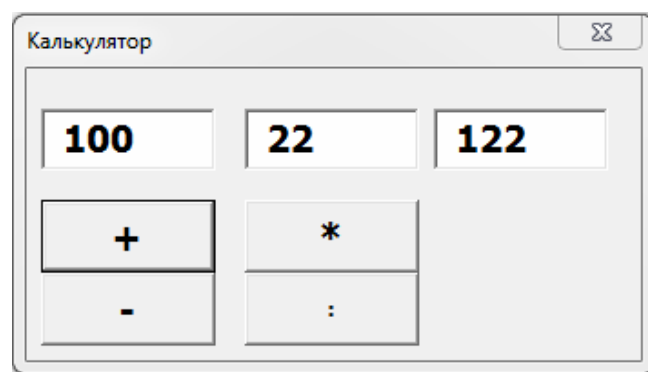


Рис. 2.22

Для прекращения работы приложения нажать стандартную кнопку Windows для закрытия окон  в правом верхнем углу формы. В результате приложение *Калькулятор* будет закрыто, а VBA вернется обратно в режим редактирования (ввода кода.)

Убедившись в правильности работы процедуры сложения *Private Sub CommandButton1_Click()*, использовать ее при написании процедуры вычитания.

Для этого выполнить двойной щелчок по кнопке «-», расположенной в форме.

В результате откроется окно редактора кода, в котором будет помещена заготовка процедуры *CommandButton2_Click()*. Скопировать внутрь процедуры вычитания программный код из процедуры сложения и заменить в нем символ «+» на символ «-». В результате получим следующий программный код:

```
Private Sub CommandButton2_Click()  
    TextBox3.Text = Val(TextBox1.Text) - Val(TextBox2.Text)  
End Sub
```

Аналогично запишем процедуры умножения *CommandButton3_Click()* и деления *CommandButton4_Click()*.

Проверить работу вновь созданных процедур.

На этом создание приложения *Калькулятор* завершено.

4. Записать в процедуре сложения имя функции *Val* с синтаксической ошибкой, например, *Vala*. Запустить приложение *Калькулятор* и нажать на кнопку «+». В результате окно с кодом программы станет активным и появится диалоговое окно с указанием ошибки, а также выделится то место в программе, где есть ошибка (рис. 2.23).

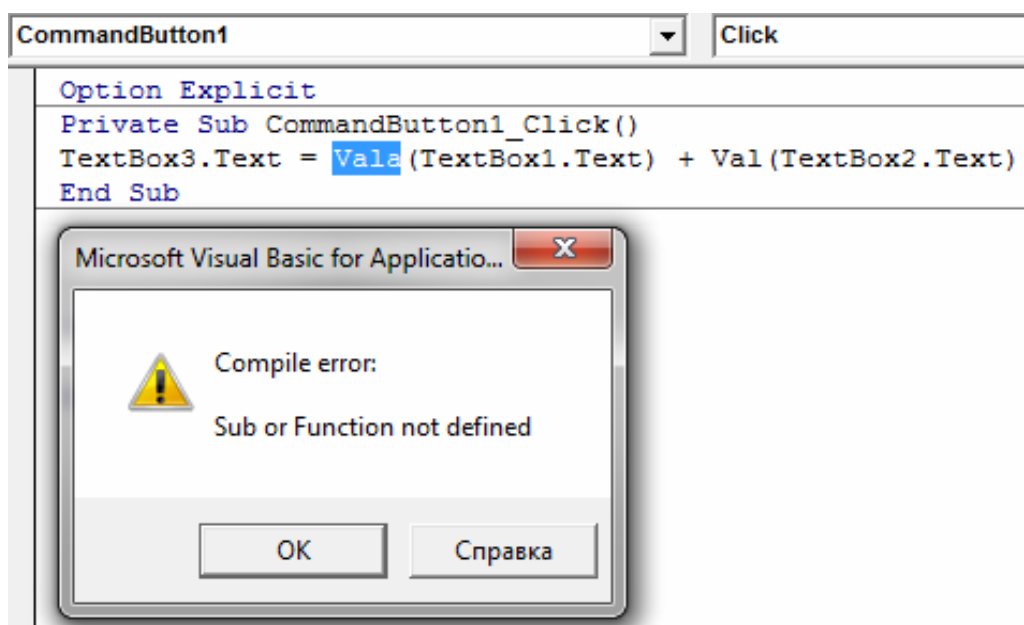


Рис. 2.23

Синтаксические ошибки отслеживаются VBA не только при трансляции кода, но и при его вводе.

При попытке запуска процедуры, содержащей ошибки *компиляции*, которые он не в состоянии правильно откомпилировать, появляется окно с указанием ошибки.

Если не понятна причина ошибки, то можно вызвать контекстную справку (кнопка *Справка*).

При нажатии на кнопку *OK* окно с сообщением об ошибке закрывается и автоматически включится режим отладки *Debug*. При этом выделится (по умолчанию желтым цветом) начало процедуры, на которой произошло прерывание выполнения программы.

В программах VBA могут возникать и *ошибки времени исполнения* или *Run-time error-ошибки*, которые возникают в случае несоответствия типов переменных в выражениях присвоения значения либо в аргументах подпрограммы, а также при попытке деления на нуль либо при переполнении и т.д. Они проявляются в ходе выполнения программного кода. Например, на рис. 2.24 приведено диалоговое окно с указанием кода ошибки, которая соответствует делению на нуль.

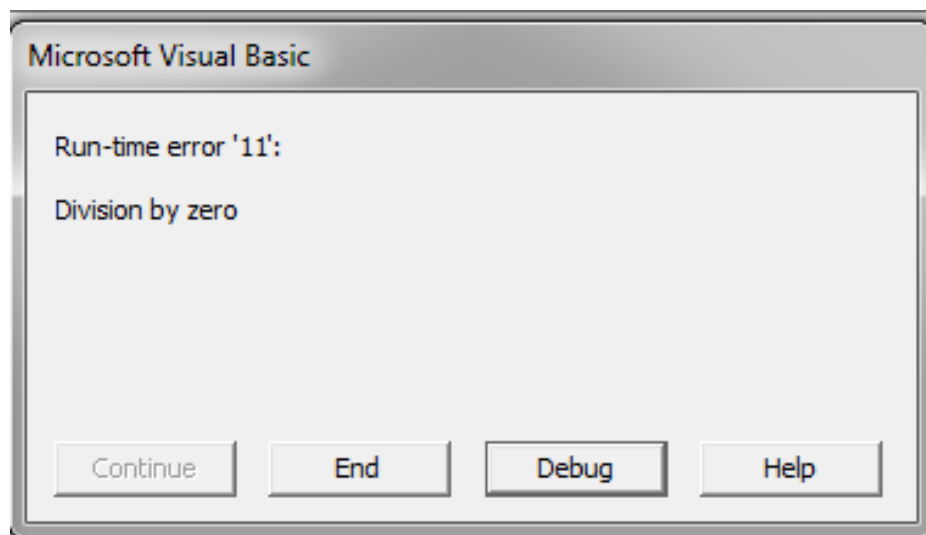


Рис. 2.24

При нажатии на кнопку *Debug* окно с сообщением об ошибке закроется и автоматически включится режим отладки *Debug*. При этом выделится (по умолчанию желтым цветом) начало процедуры, на которой произошло прерывание выполнения программы. Для того, чтобы вновь можно было бы запустить программу необходимо выйти из режима отладки, нажав кнопку *Reset* (рис. 2.25).

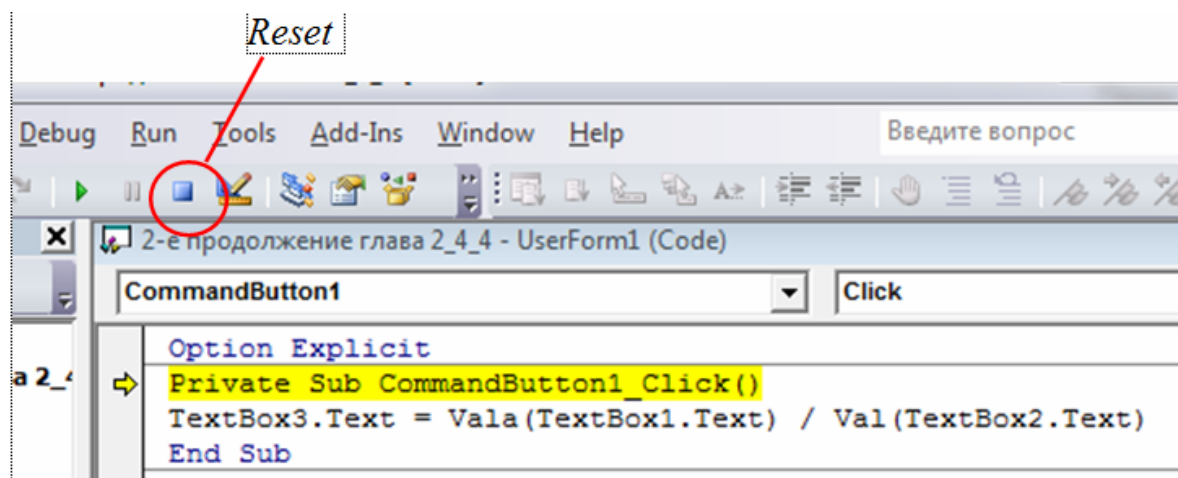


Рис. 2.25

При нажатии на кнопку *End* в окне с сообщением об ошибке (см. рис. 2.24) программа прерывается.

Исправить *синтаксическую ошибку* в процедуре сложения и убедиться, что процедура работает по-прежнему.

5. Расширить форму и добавить на нее с панели *Toolbox* следующие элементы управления: *Label* (надпись), *ComboBox* (комбинированное поле) и *CommandButton* (кнопку). Разместить элементы в форме в соответствии с рис. 2.26.

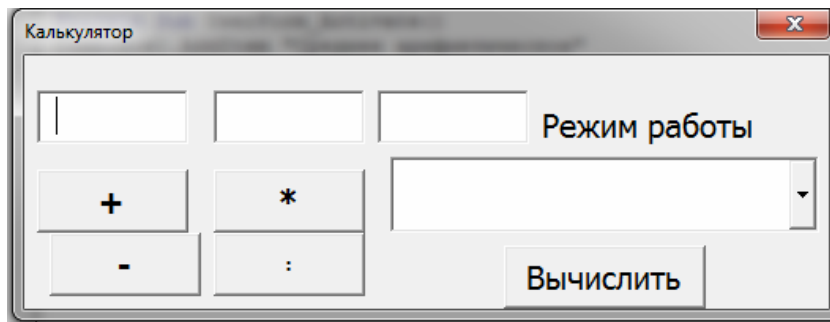


Рис. 2.26

Значения свойства *Name* элементов *Label*, *ComboBox* и *CommandButton* оставить по умолчанию: *Label1*, *ComboBox1* и *CommandButton5*.

Задать значение свойства *Caption* для объекта *Label1* – «Режим работы», а для объекта *CommandButton5* – «Вычислить».

6. Открыть окно кода, относящееся к *UserForm1* и добавить в его начале инструкцию *Option Explicit*, а также описание типа переменных модуля:

```
Переменные уровня модуля  
Private CpАрифм As Double  
Private CpГеом As Double  
Dim regim As String
```

7. Элемент управления *ComboBox1* будет использоваться в программе для выбора режима работы калькулятора. Он хранит список значений, из которых пользователь может выбрать только одно значение, а также позволяет вводить значение через поле ввода, также как и в элементе управления *TextBox*.

Для автоматического формирования списка значений в элементе *ComboBox* при появлении формы *UserForm1* на экране, включим в программу процедуру активизации формы:

```
Private Sub UserForm1_Activate()  
ComboBox1.AddItem «Среднее арифметическое»  
ComboBox1.AddItem «Среднее геометрическое»  
End Sub
```

8. Записать процедуру *Private Sub CommandButton5_Click()*, которая будет запускаться при нажатии на клавишу *Вычислить* и в зависимости от выбранного в списке *ComboBox1* режима работы вычислять среднее арифметическое или среднее геометрическое значение чисел, введенных в текстовое поле объектов *TextBox1* и *TextBox2*:

```

Private Sub CommandButton5_Click()
    regim = ComboBox1.Text
    If regim = "Среднее арифметическое" Then
        CpАрифм = (Val(TextBox1.Text) + Val(TextBox2.Text)) / 2
        CpАрифм = Round(CpАрифм, 2)
        TextBox3.Text = CpАрифм
    ElseIf regim = "Среднее геометрическое" Then
        CpГеом = (Val(TextBox1.Text)^2 + Val(TextBox2.Text)^2)^0.5
        CpГеом = Round(CpГеом, 2) : TextBox3.Text = CpГеом
    End If
End Sub

```

9. Добавить в форму элемент управления *Image* (Рисунок), который используется для отображения в форме графических файлов в формате BMP, CUR, GIF, ICO, JPG и WMF. Для этого найдем в окне свойств *Properties* для объекта *Image* свойство *Picture* и наведем курсор на правую часть соответствующей ячейки второго столбца таблицы. В результате откроется диалоговое окно *Load Picture* (рис. 2.27), в котором необходимо выбрать требуемый рисунок и нажать кнопку *Открыть*.

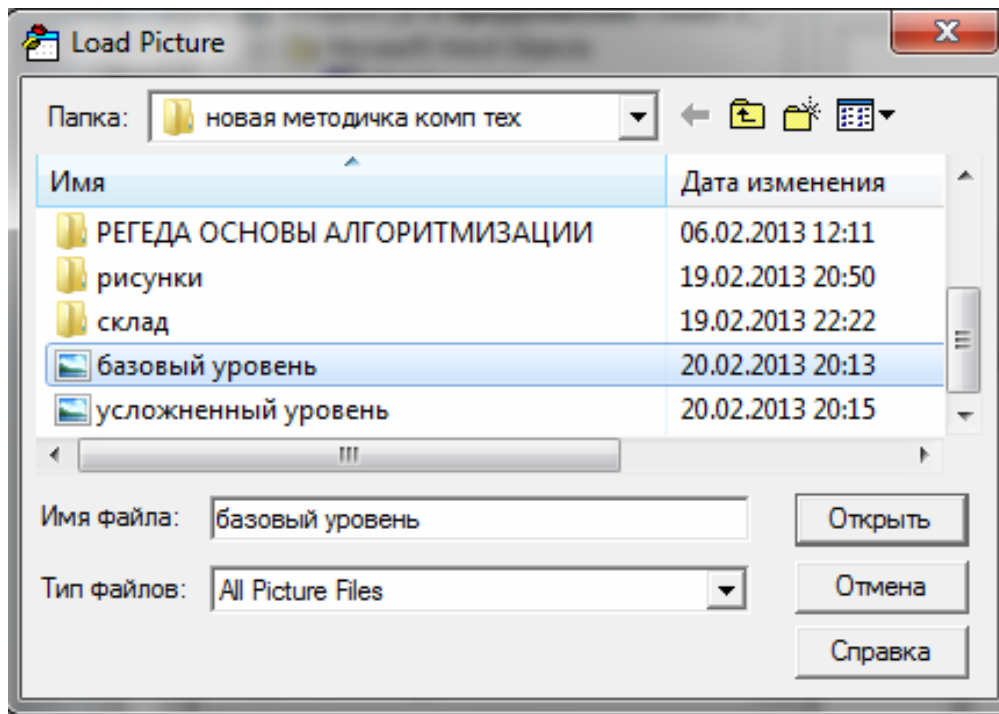


Рис. 2.27

Установить в окне *Properties* для свойства *AutoSize* значение *False*, что позволяет вписать рисунок в выбранное окно *Image*, зна-

чение *True* выводит часть рисунка, которая помещается в окне *Image*, либо уменьшает размеры окна до размеров рисунка, если рисунок оказался меньше окна.

10. Для доступа к созданному проекту из среды Microsoft Word удобно поместить в документе Word соответствующую кнопку на панели инструментов. Для этого необходимо создать пустой макрос *Макрос10()* и назначить ему кнопку на требуемой панели инструментов, а затем добавить в макросе всего одну строку, отображающую форму *UserForm1* на экране:

```
Sub Макрос10()  
UserForm1.Show  
End Sub
```

11. Проверить работу макроса и посмотреть в окне проектов *Project* структуру созданного проекта.

2.6.3. Пример создания приложения в Microsoft Excel

С помощью Microsoft Excel можно решить множество разнообразных задач. Использование средств VBA позволяет дополнительно создавать максимально удобный и гибкий интерфейс, приспособленный для решения конкретной задачи, предусмотреть средства защиты от несанкционированных действий, а также автоматизировать процесс создания электронных таблиц с помощью написания и использования собственных макрокоманд.

Создать в Microsoft Excel программу для построения графика функции $y = x^k$.

1. Войти в программу Microsoft Excel и сохранить книгу в файле с поддержкой макросов под именем *Построение графиков.xlsm*.

В случае, если на ленте меню нет раздела *Разработчик*, выбрать в меню команду *Файл* $\Rightarrow$ *Параметры*.

В появившемся диалоговом окне *Параметры Excel* (рис. 2.28) выбрать раздел *Настройка ленты* и поставить галочку в элементе флажок слева от раздела *Разработчик*.

2. Заполнить ячейки на Листе1 текущей книги в соответствии с рис. 2.29.

3. Выделить ячейки *A2:B23*, выбрать в контекстном меню команду *Формат ячеек* $\Rightarrow$ *Граница* и задать внешние и внутренние границы для выбранных ячеек.

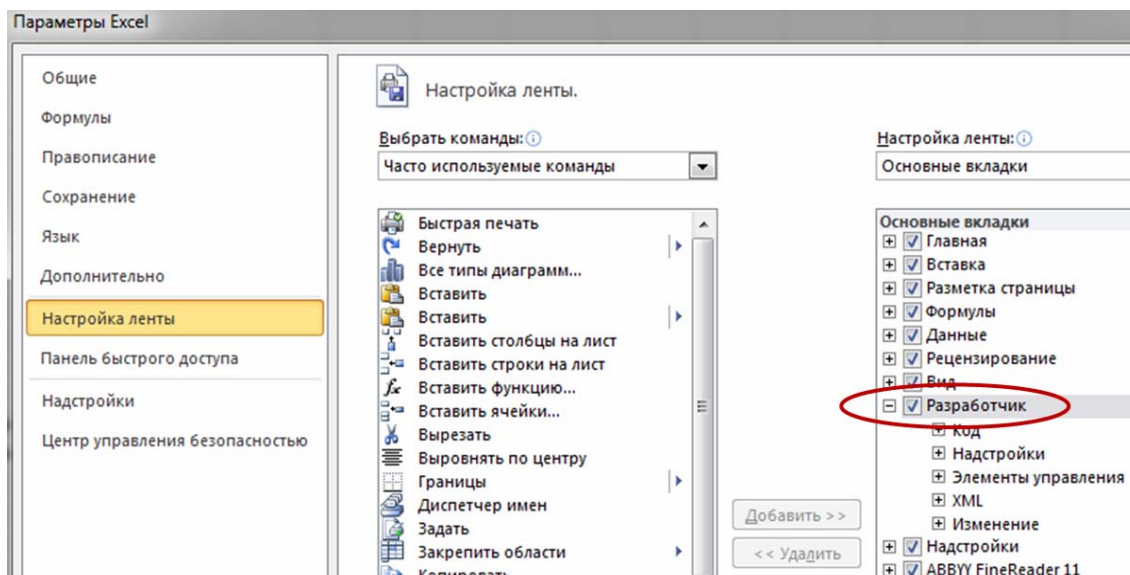


Рис. 2.28

	A	B	C	D	E	F	G	H
1	x	y(x)		Степень k =	2			
2		y		Начальное значение x =	-10			
3				Шаг изменения Δx =	1			
4								
5				Очистить таблицу	Заполнить таблицу			
6								
7								
8								
9								
10								
11								
12								
13								
14								
15								
16								
17								
18								
19								
20								
21								
22								
23								

Рис. 2.29

4. Перейти на закладку меню *Разработчик* и включить *Режим конструктора* , нажав на соответствующую кнопку (рис. 2.30). Добавить на лист две кнопки *CommandButton1* и *CommandButton2* с помощью панели *Элементы ActiveX* и назначить в соответствующем окне *Properties* свойству *Caption* для левой кнопки значение *Очистить таблицу*, а для правой кнопки значение *Заполнить таблицу*.

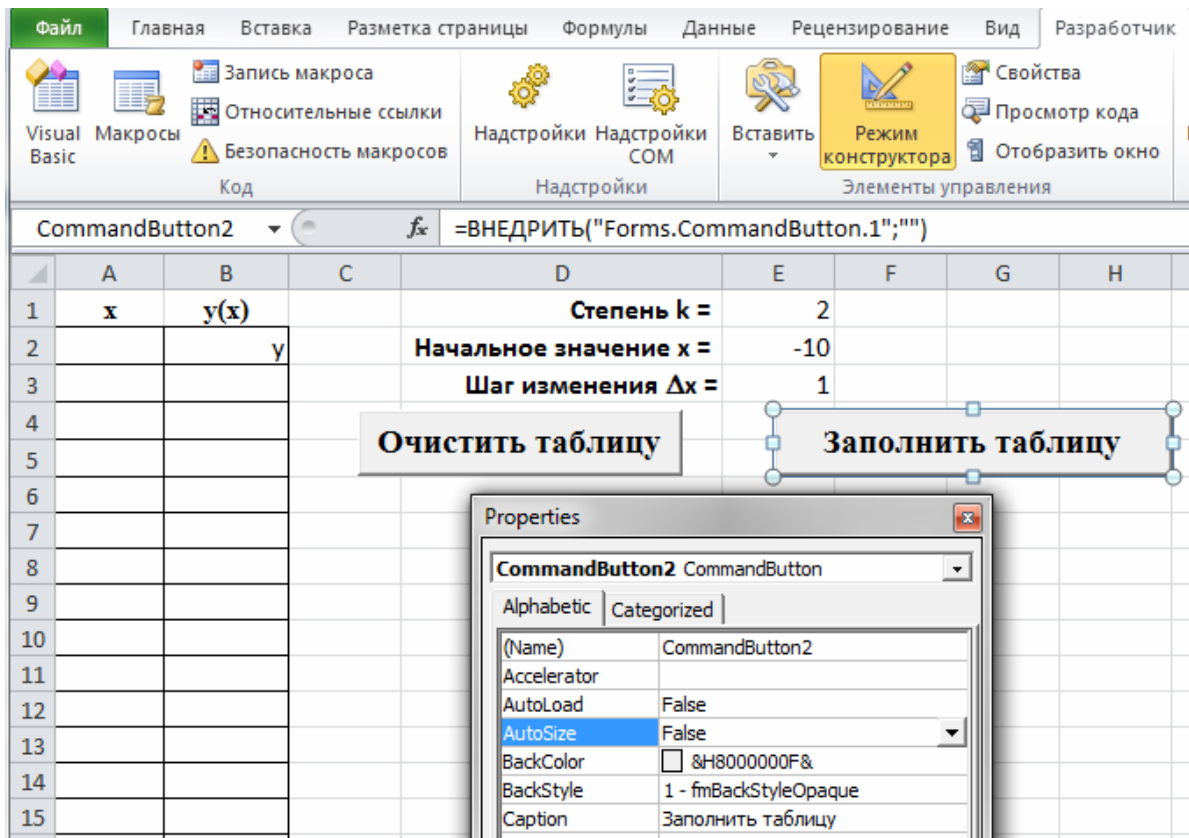


Рис. 2.30

5. Выполнить двойной щелчок на клавише *CommandButton1*, в результате откроется окно редактора кода (рис. 2.31), в котором будет помещена заготовка для процедуры обработки события *Click* объекта *CommandButton1*.

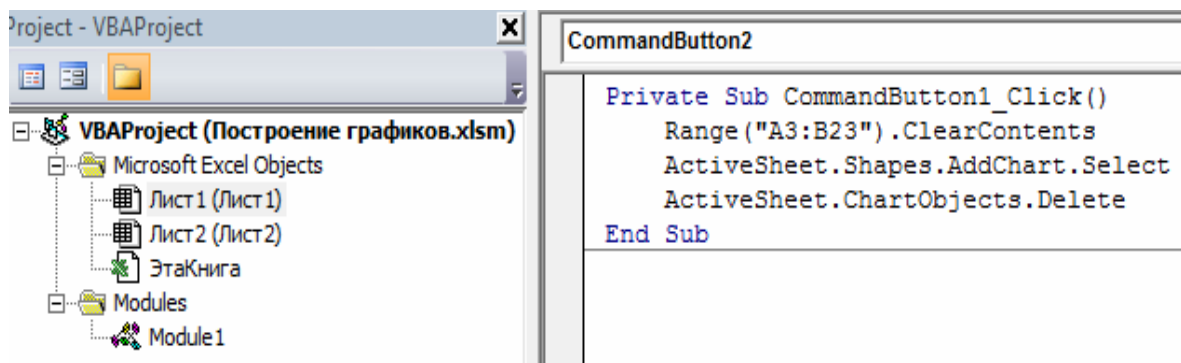


Рис. 2.31

Внутри процедуры необходимо дописать следующий код:

```
Private Sub CommandButton1_Click()
    Range("A3:B23").ClearContents
    ActiveSheet.Shapes.AddChart.Select
    ActiveSheet.ChartObjects.Delete
End Sub
```

Вторая строка записанной процедуры очищает ячейки таблицы в диапазоне ячеек *A3:B23*, а четвертая строка удаляет диаграмму *ChartObjects* из рабочего листа. Если при нажатии на кнопку *Очистить таблицу* диаграммы не было на рабочем листе, то в этой строке появится сообщение об ошибке. Поэтому в процедуру для этого случая добавлена третья строка, в которой сначала создается объект диаграмма *Chart*, а затем в четвертой строке он удаляется. Объект *Chart* является членом семейства всех графических объектов рабочего листа *Shapes*.

6) Дважды щелкнуть по клавише *CommandButton2*. В появившемся окне редактора кода необходимо дописать внутри заготовки процедуры *Sub CommandButton2_Click()* код, с помощью которого происходит заполнение ячеек таблицы в диапазоне ячеек *A3:B23*, а затем строится соответствующий график.

```
Private Sub CommandButton2_Click()
    x = Range("E2").Value
    k = Range("E1").Value
    For i = 3 To 23
        Cells(i, 1).Value = x
        Cells(i, 2).Value = x ^ k
        x = x + Range("E3").Value
    Next i
    ActiveSheet.Shapes.AddChart.Select
    With ActiveChart
        .ChartArea.Left = 110
        .ChartArea.Top = 100
    End With
    ActiveChart.ChartType = xlLine
    ActiveChart.SetSourceData Source:=Range("A2:B23")
    Name = "y(x^" & CStr(k) & ")"
    ActiveChart.ChartTitle.Text = Name
    Range("A24").Select
End Sub
```

7. Отключить *Режим конструктора* и проверить работу программы для различных значений параметров.

Результаты работы программы приведены на рис. 2.32.

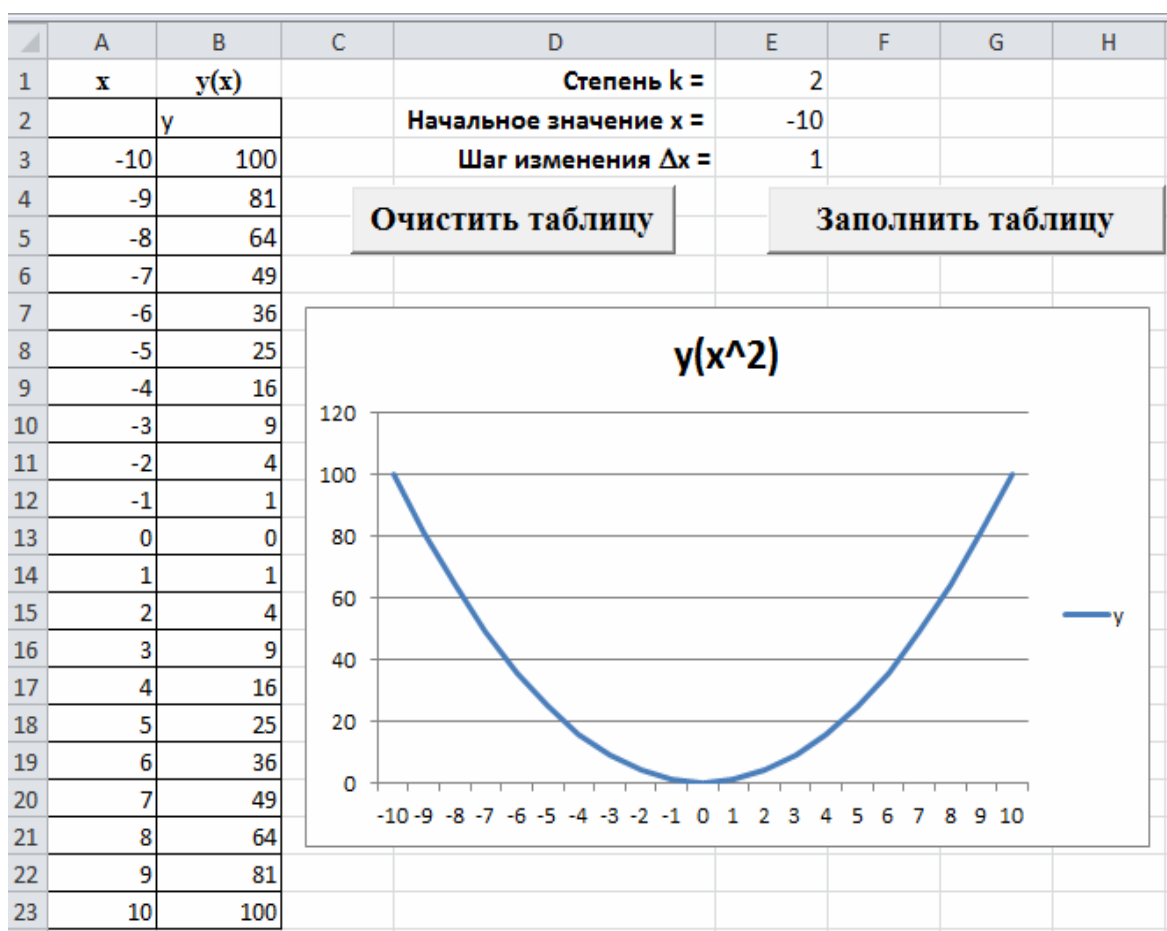


Рис. 2.32

2.7. Дополнительные возможности по созданию автоматизированных приложений в Microsoft Word

Прежде чем начать создавать автоматизированное приложение с использованием VBA, следует определить степень необходимой автоматизации и выработать определенный подход к дизайну элементов проекта.

Если документ должен быть определенным образом отформатирован или иметь заданные параметры страницы, то это удобнее осуществить средствами самого Word.

Если требования к вводимым в документ данным достаточно строги (должны выбираться из определенного списка, располагаться в определенном месте на листе документа и т.д.), то удобно применять пользовательские формы и соответствующие им VBA-программы.

После формулировки требований к приложению приступают к созданию *шаблона*, который определит требуемое форматирование, параметры страницы, а также будет содержать коды макрокоманд

и необходимые пользовательские формы. Начав с обычного документа или документа, основанного на похожем шаблоне, вставить в новый шаблон необходимые рисунки, таблицы и другие элементы.

Рассмотрим пример разработки документа-шаблона в среде Microsoft Word, предназначенного для добавления в документ рамки со штампом (для первого листа текстовых технических документов) и заполнения штампа из специально разработанной пользовательской формы с соответствующими элементами управления [10]. После заполнения формы файл требуется переименовать, чтобы оставить файл документа-шаблона без изменений.

Среда проектирования VBA предлагает удобный способ разработки собственных диалоговых окон, которые позволяют ввести текст в документ сразу в необходимом формате. Это дает возможность пользователю не заботиться о правильном положении текста в документе или использовать переменные документа для сохранения текста. Используя различные списки, переключатели, кнопки и другие управляющие элементы пользовательских форм, можно оказывать также информационную помощь пользователю.

Можно написать свою программу так, что прежде чем допустить пользователя к следующему этапу работы, будет проверяться, например, правильность введенных данных и использованных параметров. Благодаря наличию хорошо разработанных пользовательских форм, ввод данных будет происходить гораздо быстрее.

1. Создать в текстовом редакторе Microsoft Word файл с поддержкой макросов под именем «Формат A4.docm». Установить в диалоговом окне *Параметры страницы* следующие значения полей:

- левое – 2 см;
- правое – 0,5 см;
- верхнее и нижние – минимально возможные.

Затем с помощью таблицы создать в Microsoft Word рамку со штампом заданных размеров и вписать в соответствующие ячейки неизменяемый текст (рис. 2.33).

2. Пометить ключевые области документа, установив *закладки*. Закладки дают возможность VBA-программе быстро и безошибочно переходить к данной части документа. Такой подход обеспечивает быстрое выполнение различных действий в выделенной области, например, ввод текста, применение различных команд форматирования или перемещение точки вставки или курсора в то место, где пользователь должен ввести текст.

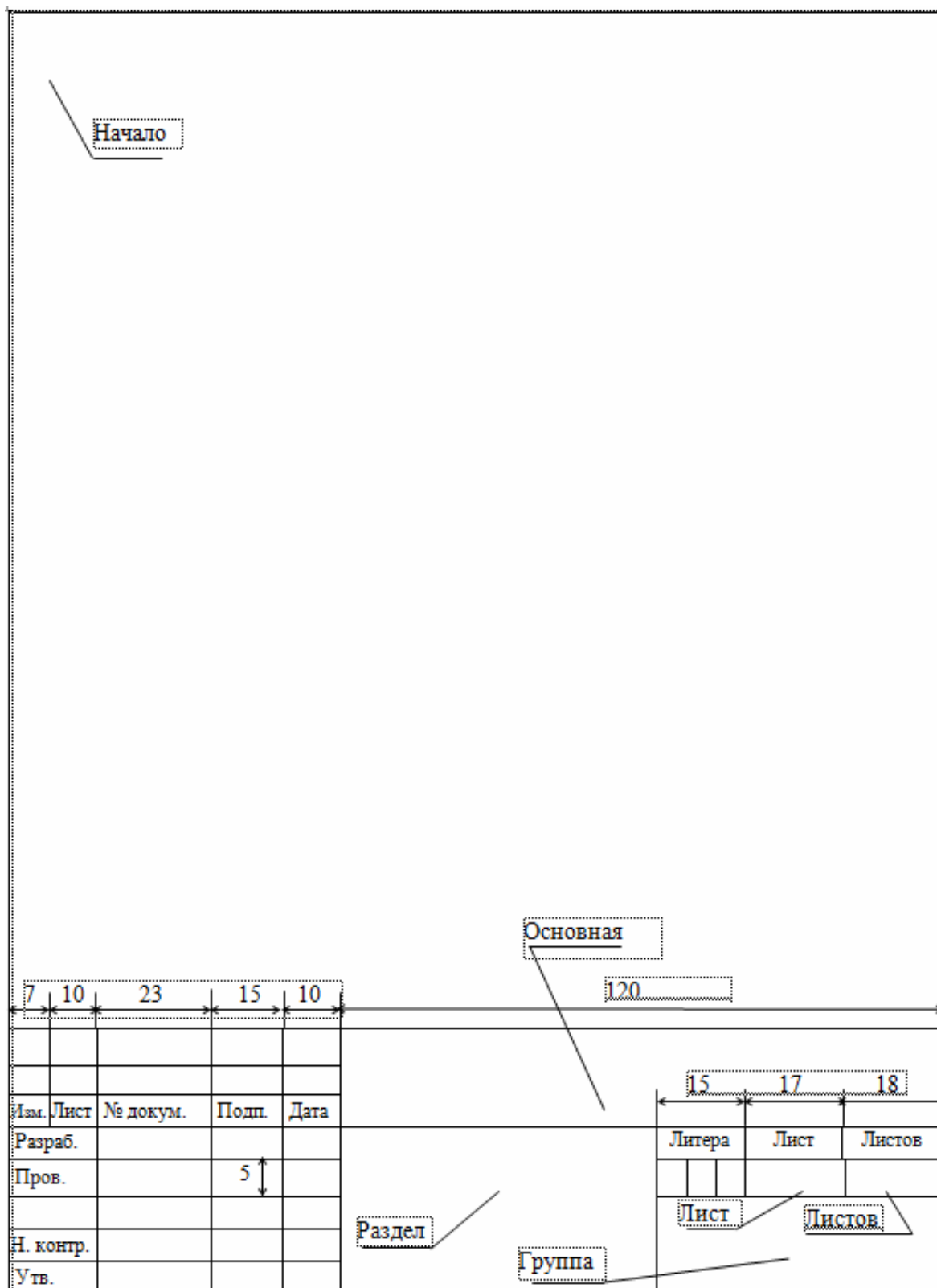


Рис. 2.33

Для создания закладки в шаблоне необходимо выполнить следующие действия:

- установить курсор в место для закладки;
- в меню *Вставка* выбрать команду *Закладка*;

– ввести имя закладки в поле *Имя закладки* и щелкнуть на кнопке *Добавить* (рис. 2.34).

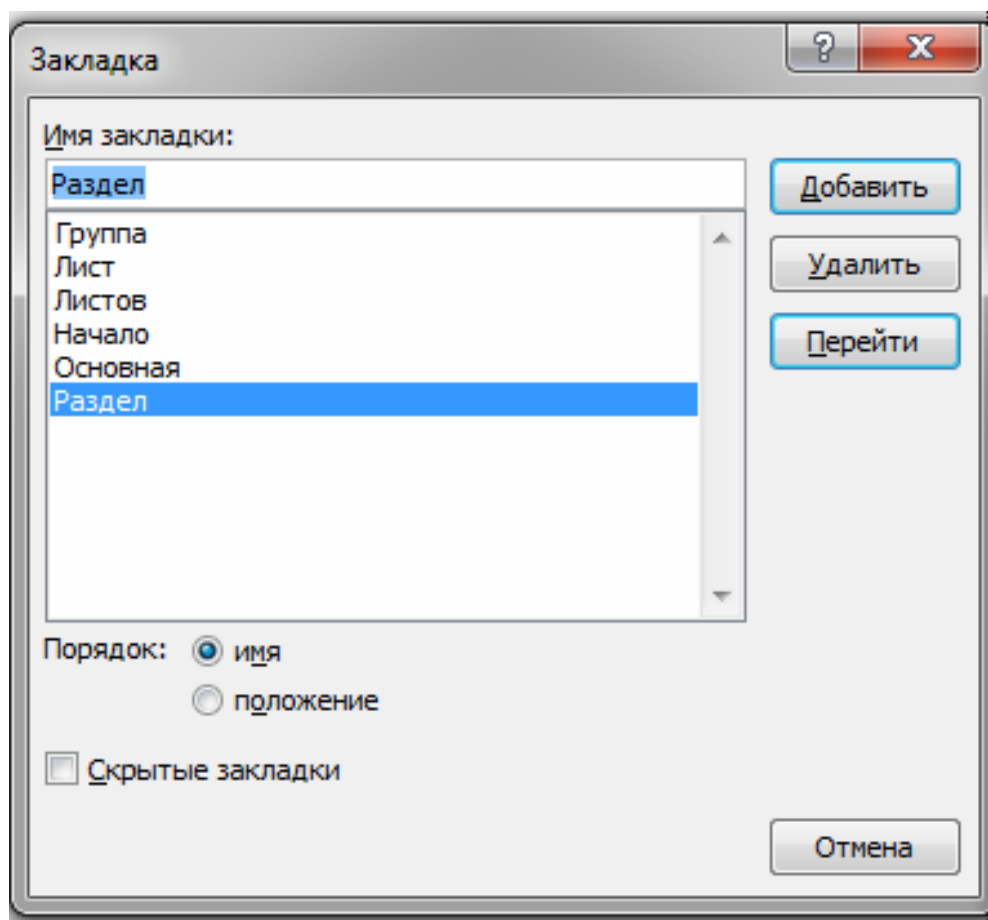


Рис. 2.34

Добавить в шаблон следующие закладки:

Основная – указывает место, куда вводится основная надпись;

Раздел – указывает место, куда вводится название раздела;

Лист – указывает место, куда вводится номер листа;

Листов – указывает место, куда вводится количество листов;

Группа – указывает место, куда вводится номер группы;

Начало – указывает место, куда по завершении макроса будет помещена точка вставки. Именно здесь должен начать вводиться текст технического документа.

Установить на месте всех закладок параметры шрифта и абзаца, указанные в табл. 2.9.

3. Прежде чем приступать к разработке пользовательской формы, которая будет определять интерфейс нашего проекта, необходимо продумать, какого рода информация должна быть выведена на

нее и определить, какие элементы управления процессом создания документа можно предоставить пользователю для заполнения штампа.

Таблица 2.9

Закладка	Размер шрифта	Вариант выравнивания
<i>Основная</i>	<i>20</i>	<i>По центру</i>
<i>Раздел</i>	<i>15</i>	То же
<i>Лист</i>	<i>10</i>	—"—
<i>Листов</i>	<i>10</i>	—"—
<i>Группа</i>	<i>15</i>	—"—
<i>Начало</i>	<i>15</i>	<i>По ширине</i>

Перейти в редактор IBA, вставить в проект *Project (Формат А4)* форму *UserForm1* и разместить на ней четыре надписи *Label1 – Label4* и одну кнопку *CommandButton1* (рис. 2.35).

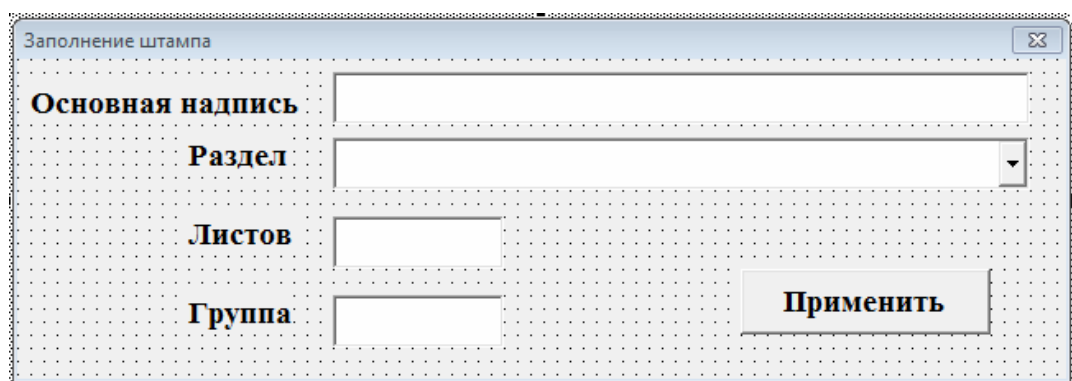


Рис. 2.35

Задать для вставленных в проект объектов значения свойства *Caption* в соответствии с табл. 2.10.

Таблица 2.10

Класс объектов (свойство <i>Name</i>)	Новое значение свойства <i>Caption</i>
Форма (<i>UserForm1</i>)	<i>Заполнение штампа</i>
Надпись (<i>Label1</i>)	<i>Основная надпись</i>
Надпись (<i>Label2</i>)	<i>Раздел</i>
Надпись (<i>Label3</i>)	<i>Листов</i>
Надпись (<i>Label4</i>)	<i>Группа</i>
Кнопка (<i>CommandButton1</i>)	<i>Применить</i>

Напротив надписи «Раздел» (см. рис. 2.31) вставить комбинированное поле *ComboBox1*.

Добавить в форму три *текстовых поля* (объекты *TextBox1*, *TextBox2*, *TextBox3*) и задать для них значения свойства *Name* в соответствии с табл. 2.11.

Таблица 2.11

Класс объектов (свойство <i>Name</i>)	Новое значение свойства <i>Name</i>
Текстовое поле <i>TextBox1</i>	<i>Основная</i>
Текстовое поле <i>TextBox2</i>	<i>Листов</i>
Текстовое поле <i>TextBox3</i>	<i>Группа</i>

4. После разработки документа и формы необходимо создать несколько процедур VBA, связанных с формой.

В модуле *ThisDocument* проекта *Project* (Формат A4) запишем процедуру *Document_Open*, которая выполняется каждый раз, когда открывается документ с заданным шаблоном.

```
Private Sub Document_Open()
' Формирование списка ComboBox1
UserForm1.ComboBox1.AddItem "Схема структурная"
UserForm1.ComboBox1.AddItem _
    "Схема электрическая функциональная"
UserForm1.ComboBox1.AddItem _
    "Схема электрическая принципиальная"
UserForm1.Show ' Отображение формы на экране
'Выбор закладки Основная
ActiveDocument.Bookmarks("Основная").Select
Selection.TypeText Text:=UserForm1.Основная.Text
ActiveDocument.Bookmarks("Раздел").Select
Selection.TypeText Text:=UserForm1.ComboBox1.Text
ActiveDocument.Bookmarks("Листов").Select
Selection.TypeText Text:=UserForm1.Листов.Text
ActiveDocument.Bookmarks("Группа").Select
Selection.TypeText Text:=UserForm1.Группа.Text
'Выбор закладки Лист
ActiveDocument.Bookmarks("Лист").Select
'Вставка номера страницы из поля PAGE
Selection.Fields.Add Range:=Selection.Range, _
    Type:=wdFieldEmpty, Text:="PAGE\*Arabic ", _
    PreserveFormatting:=True
'Переход на закладку Начало
```

```
ActiveDocument.Bookmarks("Начало").Select  
End Sub
```

Для ячейки таблицы «Лист» удобно использовать поле *Page*, которое в Microsoft Word вставляет номер текущей страницы. Для переключения между отображением результата кодов полей и отображением самих кодов нажмите комбинацию клавиш *Alt+F9*; чтобы включить отображение кодов полей нажмите клавиши *Alt+F9* еще раз.

В модуле *UserForm1* проекта *Project* (Формат А4) записать процедуру *CommandButton1_Click*, которая будет выполняться при нажатии на клавишу *Применить*, расположенную в форме:

```
Private Sub CommandButton1_Click()  
Dim vvod As Boolean : vvod = True  
If Len (Me.Основная.Text) = 0 Then vvod = False  
If Len (Me.ComboBox1.Text) = 0 Then vvod = False  
If Len (Me.Листов.Text) = 0 Then vvod = False  
If Len (Me.Грунна.Text) = 0 Then vvod = False  
If vvod = True Then  
Me.Hide  
Else  
MsgBox "Введите информацию во все поля формы", 48, _  
"Ошибка ввода данных"  
End If  
MsgBox "Не забудьте сохранить файл под требуемым именем"  
End Sub
```

Процедура начинается с объявления переменной *vvod* с начальным значением *True*. Далее функция *Len* проверяет длину введенных строк в текстовых полях формы. Если хотя бы одно из полей будет пустым, то на экран выводится сообщение, предупреждающее пользователя, что необходимо заполнить форму до конца.

Обратите внимание, что в процедуре используется краткое обращение к форме по умолчанию как «*Me*».

5. На этом процесс создания документа-шаблона завершен. Сохранить окончательный проект полностью в том же файле «Формат А4.docx».

Выйти из Microsoft Word и повторно открыть тот же файл, разрешив загрузку макросов при открытии, и проверить его работу.

Глава 3. Применение графического редактора CorelDraw для создания иллюстраций

3.1. Общие сведения о графических редакторах

Программы, позволяющие создавать и редактировать изображения в программной среде, получили название *графический редактор*. Хотя для работы с компьютерной графикой существует множество классов программного обеспечения, принято выделять три вида компьютерной графики: растровую графику, векторную и фрактальную. Они отличаются принципами формирования изображения при отображении на экране монитора или при печати на бумаге.

Растровую графику применяют при разработке электронных (мультимедийных) и полиграфических изданий. Для создания растровой графики чаще всего используют сканированные иллюстрации, подготовленные художником на бумаге, или фотографии, реже она создается вручную с помощью компьютерных программ. Для ввода растровых изображений в компьютер широкое применение нашли цифровые фото- и видеокамеры. Большинство графических растровых редакторов ориентированы не столько на создание изображений, сколько на их обработку.

Программные средства для работы с *векторной* графикой прежде всего ориентированы на создание иллюстраций и в меньшей степени на их обработку. Оформительские работы, основанные на применении шрифтов и простейших геометрических элементов средствами векторной графики, решаются намного проще. Поэтому большинство современных растровых редакторов содержат в качестве вспомогательных векторные инструменты редактирования.

Программные средства для работы с *фрактальной* графикой автоматически генерируют изображения, запрограммированные с помощью математических уравнений (или систем уравнений). Изменив коэффициенты в уравнении, можно получить совершенно другую фрактальную картину. Простейший фрактальный объект (треугольник) путем многократного масштабирования образует треугольники последующих поколений, которые наследуют свойства своих родительских фрактальных структур. Так рождается фрактальная фигура. Фрактальными свойствами обладают многие объек-

ты живой и неживой природы, например обычная снежинка. Фрактальные алгоритмы лежат в основе роста кристаллов и растений. Фрактальную графику в основном используют для создания развлекательных программ.

В настоящее время одним из самых популярных растровых редакторов является AdobePhotoshop. К данному типу также относится один из самых простых и потому востребованных редакторов Paint. Самые популярные векторные графические редакторы – это CorelDRAW, AdobeIllustrator, Xara.

3.2. Знакомство с графической системой CorelDRAWX6

Основным понятием в редакторе CorelDRAW является понятие объекта. *Объектом* называется элемент изображения: прямая, круг, прямоугольник, кривая, замкнутая кривая, многоугольник и т.п. С помощью комбинации нескольких объектов можно создавать новые довольно сложные объекты. Любой объект имеет некоторое количество точек или узлов, соединенных прямыми или кривыми линиями – сегментами, которые также имеют свой цвет и толщину. Координаты узлов и параметры сегментов определяют внешний вид объекта. Сегменты объекта образуют контуры (замкнутые или разомкнутые). Область внутри объекта можно закрасить или залить одним цветом, смесью цветов или узором.

Графический редактор CorelDRAW позволяет создавать простые геометрические фигуры или произвольные кривые и ломаные, вставлять и форматировать текст, редактировать любой объект, вставлять готовые картинки в документ, применять разнообразные художественные эффекты, размещать объекты в нужных местах и определять порядок их взаимного перекрытия [12].

3.2.1. Интерфейс программы

На рис. 3.1 показаны основные элементы рабочего окна редактора CorelDRAWX6.

В центре окна программы расположен рисунок листа бумаги, называемый *рабочей областью*. Хотя можно рисовать как внутри рабочей области, так и вне ее, но при выводе на печать по умолчанию будет напечатано только содержимое рабочей области.

Полосы прокрутки позволяют передвигаться по изображению, а измерительные линейки – точно позиционировать элементы рисунка и измерять их размеры.

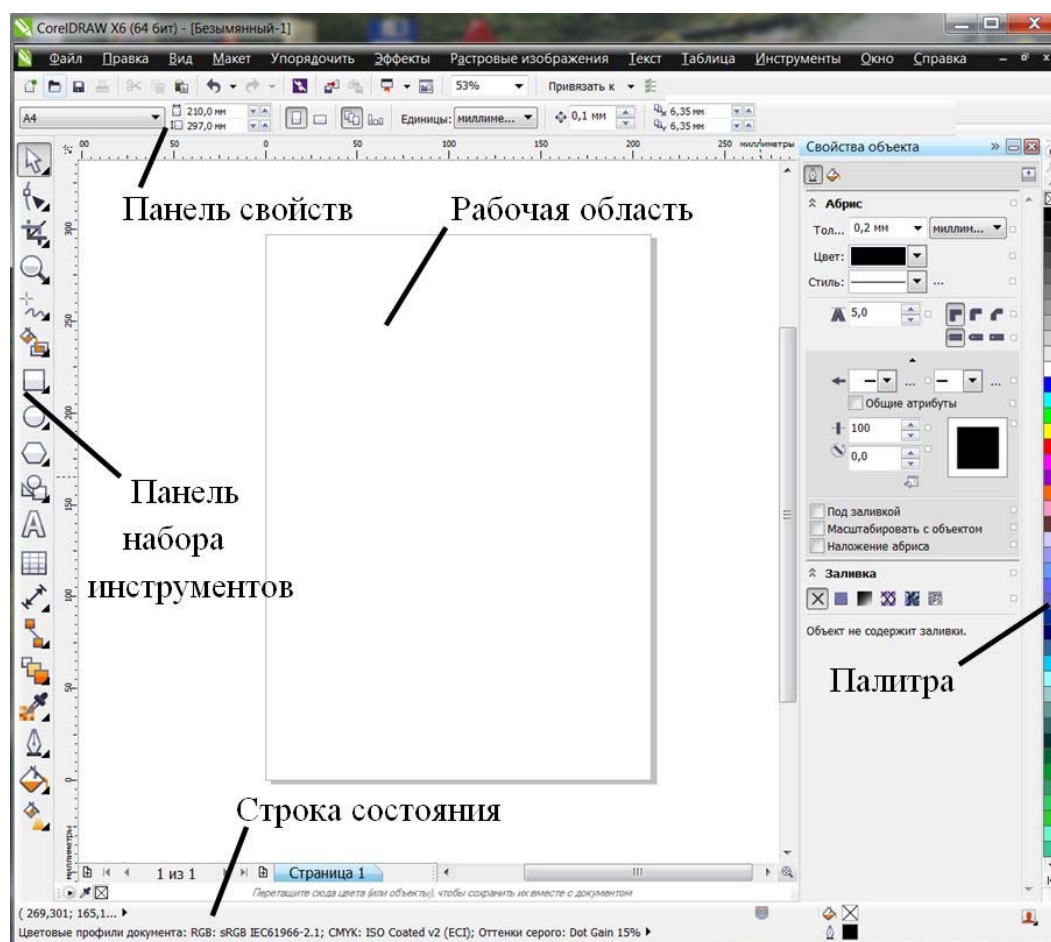


Рис. 3.1

Для работы с цветом в правой части окна расположена *Палитра цветов*.

В *Строке состояния* выводится различная информация, существенно облегчающая работу с редактором.

Кнопки в *панелях набора инструментов (Toolbox)* позволяют быстро выполнять любые операции в редакторе. При щелчке правой кнопкой мыши на свободном месте в любой панели инструментов появляется вспомогательное меню (рис. 3.2), на котором галочками отмечены видимые панели.

Слева располагается вертикальная *Панель набора инструментов Toolbox* (см. рис. 3.1), в каждой ячейке которой находится один или несколько инструментов.

Если инструментов несколько, то рядом отображается треугольник, при щелчке по которому раскрывается дополнительная панель с инструментами, относящимися к данной ячейке блока.

Особенно интересна *Панель свойств*, на которой располагаются инструменты, относящиеся к выбранной команде.

Создание новых файлов, открытие существующих файлов и сохранение их на диске осуществляется стандартным способом, как

и для любой программы под Windows. По умолчанию файл сохраняется с расширением *.cdr*.

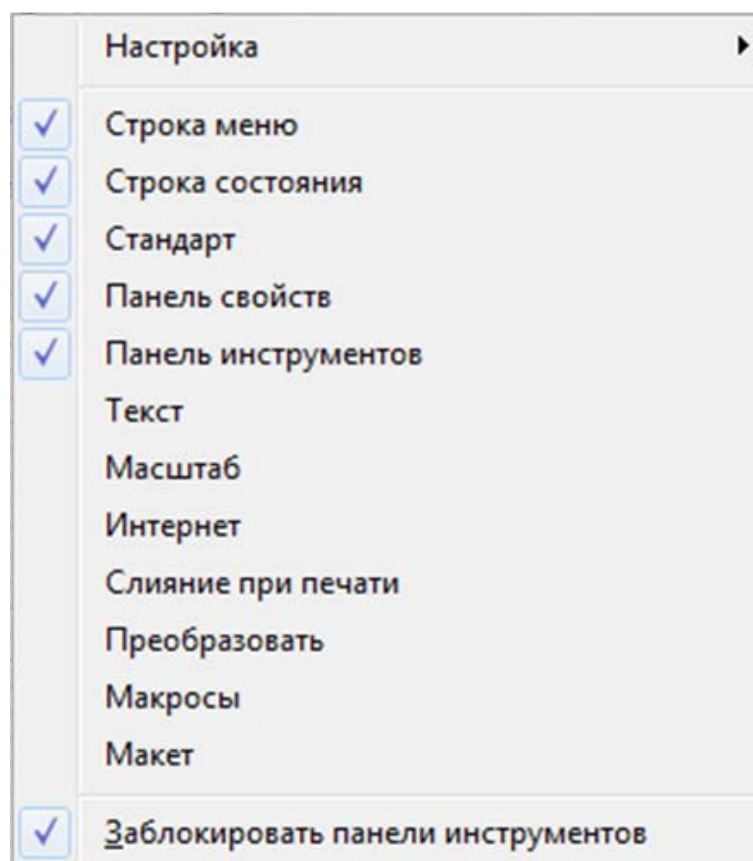


Рис. 3.2

Можно сохранить файл в формате как предыдущих версий CorelDRAW, а также и во многих других векторных графических форматах. При необходимости имеется возможность сохранить в файле только выделенные объекты.

3.2.2. Выделение объектов, изменение масштаба просмотра изображения и макет страницы

Чтобы выделить любой объект в CorelDRAW, надо выбрать инструмент  *Pick* (*Инструмент выбора*). При наличии множества перекрывающихся друг друга объектов, проще их выделять, щелкая на контуре объекта. Чтобы отменить выделение объектов, надо щелкнуть мышью на свободном месте в рабочей области, на котором нет объектов.

Для выделения сразу всех объектов дважды щелкните мышью на кнопке . Для выделения группы объектов сначала выделите первый объект, а затем нажмите клавишу *Shift* и, не отпуская ее, вы-

делите остальные объекты, щелкая мышью последовательно на контуре каждого объекта. При этом в строке состояния выводится информация о выделенных объектах. Отпустите клавишу *Shift*. Повторное нажатие на объект при нажатой клавише *Shift* отменяет выделение этого объекта.

Можно выделить требуемые объекты с помощью рамки выделения, указав с помощью мыши ее противоположные углы. Объекты, целиком оказавшиеся внутри пунктирного прямоугольника, будут выделены. Если во время выделения держать нажатой клавишу *Alt*, то выделены будут не только объекты, оказавшиеся внутри пунктирного прямоугольника целиком, но и частично. После выделения нескольких объектов с ними можно работать как с одним объектом.

Нажать кнопку  *Zoom Tool (Масштаб (Z))* панели инструментов. В результате откроется *Панель масштабов* (рис. 3.3). Поместить указатель мыши в центр рисунка, щелкнуть мышью, и масштаб будет увеличен. Щелкнуть еще раз, и масштаб снова увеличится. Щелкнуть правой кнопкой мыши, чтобы уменьшить масштаб.



Рис. 3.3

Можно установить указатель мыши в любой угол площади, которую вы хотите просмотреть в крупном масштабе. Нажать левую кнопку мыши, не отпуская ее, перевести указатель к противоположной границе выбранной площади и отпустить кнопку мыши. При этом объекты, попавшие в пунктирный прямоугольник, будут увеличены.

Можно выбрать масштаб изображения из списка масштабов *Zoom Levels*, расположенного в левой части *Панели масштабов* или на стандартной панели.



Изучить назначение остальных кнопок на *Панели масштабов*. Попробовать изменить масштаб изображения различными способами.

Рабочая область в CorelDraw определяет, что будет выводиться на печать, хотя можно рисовать и за ее пределами. Для создания макета страницы используются команды из раздела меню *Макет*.



Изучить команды из раздела меню *Макет*. Создать рисунок и установить для него следующие *Установки страницы*: полная страница с форматом *A4*, ориентация *Книжная*.

3.2.3. Использование линеек, сетки и объектной привязки

Измерительные линейки, расположенные сверху и слева, позволяют определить размеры и местонахождение объектов, а направляющие линии и сетка позволят более точно позиционировать объекты. Единицы измерения на линейках соответствуют единицам измерения, принятым в редактируемом документе.

При выбранном инструменте  и отсутствии выбранных объектов можно установить единицы измерения с помощью списка *Units (Единицы)* панели свойств. Двойной щелчок по любой из линеек открывает диалоговое окно *Параметры*, в котором также можно изменить эти установки.

Щелкнуть правой кнопкой мыши на кнопке, находящейся на пересечении двух линеек. В появившемся вспомогательном меню выбрать команду *Grid Setup (Настройка сетки)*. На экране появится диалоговое окно *Параметры* в разделе настройки сетки.

Задать для своего рисунка установки, аналогичные установкам в окне (рис. 3.4).

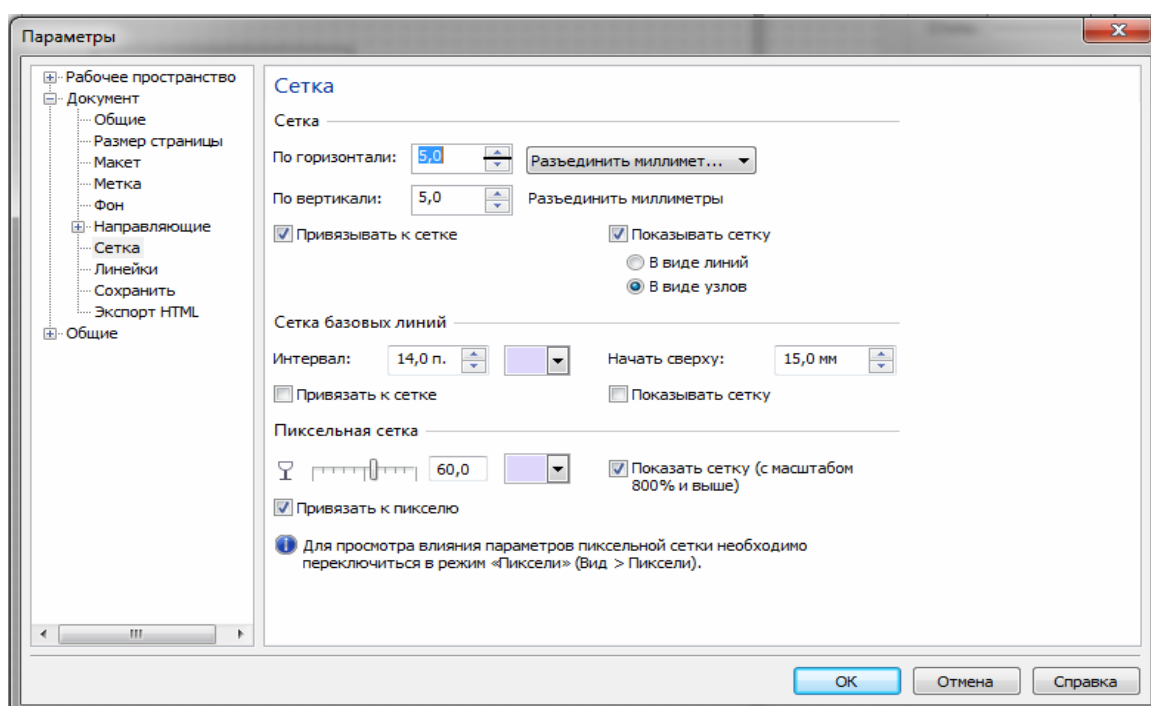


Рис. 3.4

Перейти к другим диалоговым окнам настройки можно, щелкнув мышью на нужной строке в левой части диалогового окна.

При перемещении или рисовании объекта с большей точностью, помимо привязки к сетке, его можно привязывать к пиксельной или базовой сетке, к направляющим, к элементам страницы (таким, как центр страницы), а также к другому объекту рисунка. При приближении указателя к точке привязки она выделяется, указывая на то, что указатель перейдет к этой точке.

Для установки требуемого режима привязки можно воспользоваться диалоговым окном *Параметры* либо выбрать раздел меню *Вид*⇒*Привязать к*. В табл. 3.1 приведены все доступные режимы привязки.

Таблица 3.1

Режим привязки	Описание	Индикатор режима привязки
Узел	Привязка к узлу объекта	
Пересечение	Привязка к геометрическому пересечению объектов	
Средняя точка	Привязка к средней точке сегмента линии	
Квадрант	Привязка к следующим точкам окружности, эллипса или дуги: 0°, 90°, 180° и 270°.	
По касательной	Привязка к точке на внешнем крае объекта (дуги, окружности или эллипса), в которой объект соприкасается с линией, но не пересекает ее	
Поперечный	Привязка к точке внешнего края сегмента, в которой линия будет перпендикулярна объекту	
Край	Привязка к точке касания края объекта	
По центру	Привязка к центру ближайшего объекта	
Базовая линия текста	Привязка к базовой линии фигурного или простого текста	

3.3. Создание простых векторных объектов

Самые сложные рисунки создаются в CorelDRAW из множества простых объектов. Для вставки большинства из них (прямоугольник, эллипс, многоугольник, звезда, разлинованная бумага,

спираль, стандартные фигуры) в рисунок необходимо выполнить следующие действия:

1) установить указатель мыши в любом месте на рабочем поле. При этом внешний вид указателя изменится на крестик вместе с изображением ниже и правее выбранной фигуры;

2) нажать левую кнопку мыши и, не отпуская ее, начать передвигать мышь. На экране появится вставляемая фигура, размеры и форма которой будут меняться вместе с передвижением мыши;

3) при отпускании левой кнопки мыши фигура останется на экране. Вокруг созданного объекта по периметру располагаются черные прямоугольники, в центре – перекрестие, а в вершинах – контурные прямоугольники. Все эти управляющие элементы предназначены для редактирования объекта.

Если при создании указанных выше фигур использовать клавишу *Ctrl*, то автоматически будут создаваться правильные фигуры (квадрат, круг, симметричные многоугольник, звезда и спираль, пропорциональные стандартные фигуры).

Если при создании всех рассмотренных фигур использовать клавишу *Shift*, то объекты будут создаваться относительно центра, совпадающего с начальной точкой.

Можно воспользоваться и обеими клавишами, чтобы создавать правильные объекты из центра.

3.3.1. Прямоугольник и эллипс

Выберать на панели инструментов инструмент  *Rectangle* (*Прямоугольник*) и, указав мышкой один из углов прямоугольника, передвинуть мышь в направлении его противоположного угла до получения требуемого изображения (рис. 3.5,а).

Убедиться, что при нажатой клавише *Ctrl* с помощью инструмента *Прямоугольник* можно нарисовать только квадрат (рис. 3.5,б).

Нарисовать прямоугольник под углом, выбрав на панели инструментов инструмент  (*Прямоугольник через 3 точки*) (рис. 3.5,в). Определить, чем он отличается от предыдущего.

Аналогично нарисовать эллипс и круг, выбрав инструмент  *Ellipse* на панели инструментов, а также эллипс под углом  (рис. 3.5,г,д,е).

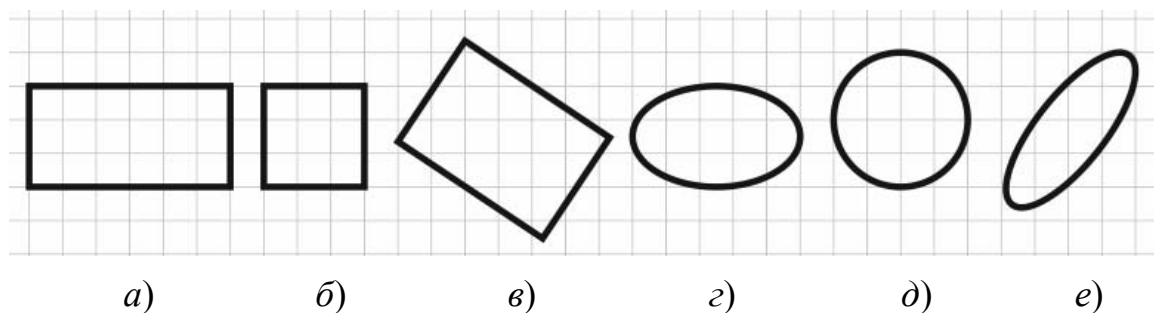


Рис. 3.5

Каждый объект в CorelDRAW обладает определенным набором свойств, которые можно посмотреть, если объект выделен, на соответствующей панели свойств (рис. 3.6).

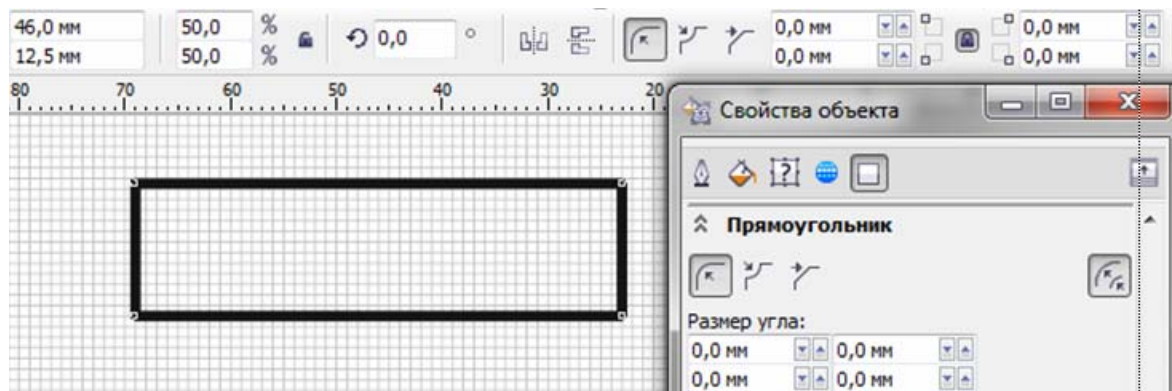


Рис. 3.6

Кроме того, свойства выделенного объекта можно посмотреть в окне *Свойства объекта*, которое открывается из раздела меню *Окно* $\Rightarrow$ *Окна настройки* (см. рис. 3.6).

3.3.2. Многоугольник, звезда, сложная звезда, разлинованная бумага и спираль

На панели (рис. 3.7) расположены команды для рисования многоугольника, звезды, сложной звезды, разлинованной бумаги и спирали.

При рисовании многоугольника (рис. 3.8,а) с помощью инструмента *Многоугольник* на панели свойств можно изменить число его вершин, повернуть его на требуемый угол, установить требуемую толщину линии и т.д.

При рисовании звезды (рис. 3.8,б) с помощью инструмента *Звезда* перетащить указатель в пределах нужной области. На панели свойств можно изменить число ее вершин, повернуть ее на требуемый угол, задать остроту лучей звезды, установить требуемую толщину линии и т.д.

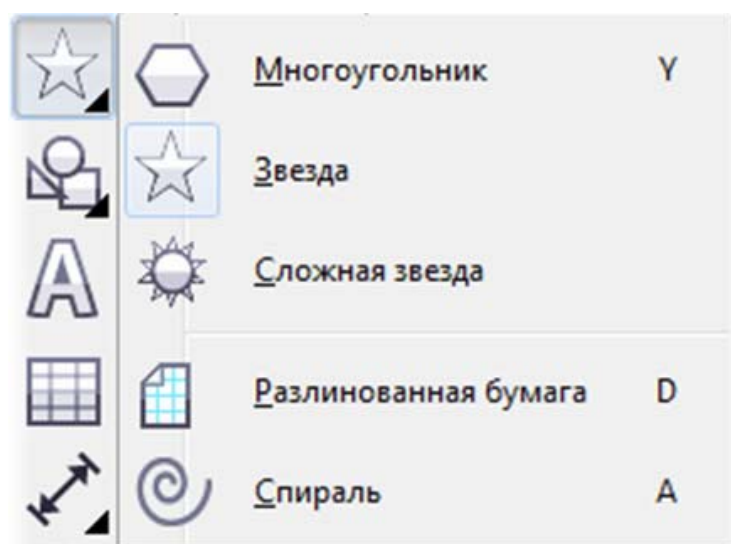


Рис. 3.7

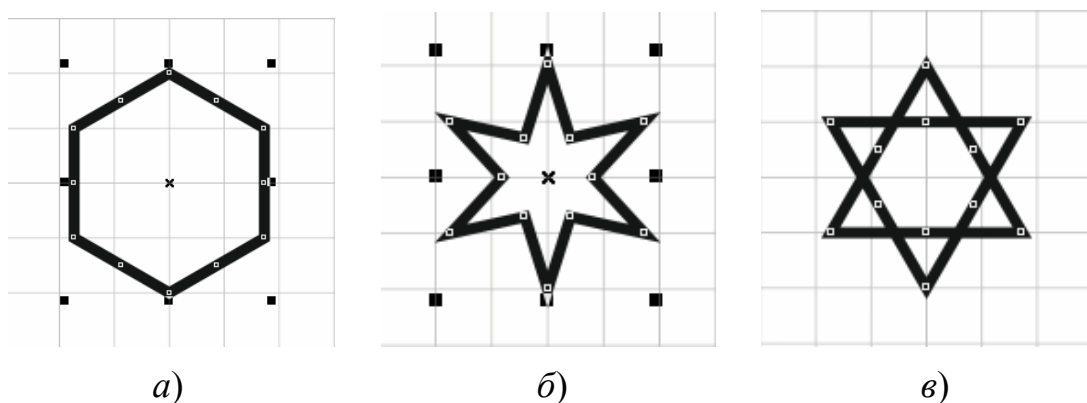


Рис. 3.8

Инструмент *Сложная звезда* работает аналогично, как и инструмент *Звезда*. Однако, выбирая требуемую остроту лучей, можно изменять ее вид. На рис. 3.8,в для звезды с шестью лучами каждая вторая вершина соединена друг с другом.

Инструмент *Разлинованная бумага* позволяет нарисовать сетку (рис. 3.9,а) и задать для нее число столбцов и строк, используя элементы управления на панели свойств. Если внешнюю границу сетки требуется сделать квадратной, удерживайте нажатой клавишу *Ctrl*.

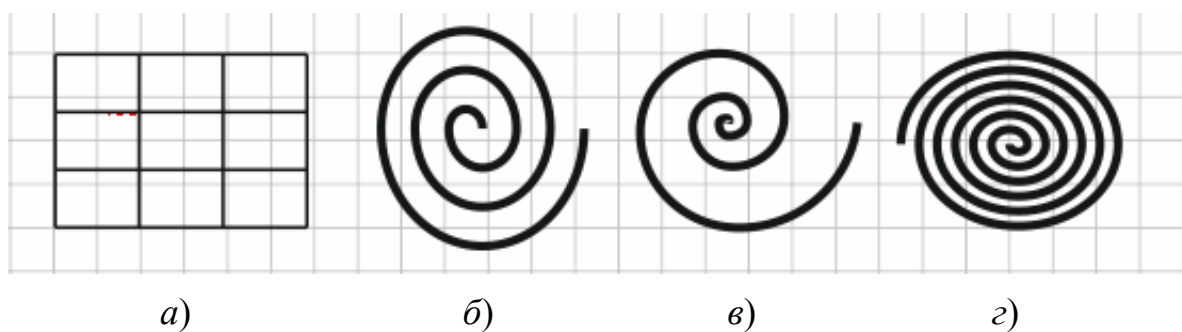


Рис. 3.9

Чтобы нарисовать спираль, выбрать инструмент *Спираль*, задать на панели свойств тип спирали, нажав кнопку *Симметрическая спираль*  (рис. 3.9,б) или *Логарифмическая спираль*  (рис. 3.9,в), а затем в окне рисования перетащить указатель в пределах области, в которой требуется ее разместить.

На панели свойств можно также установить требуемое число витков спирали. На рис. 3.9,г установлено шесть витков спирали.

В режиме логарифмической спирали имеется возможность с помощью ползунка на панели свойств установить требуемое значение коэффициента расширения спирали (от 1 до 100), т.е. насколько сильно она будет расширяться с каждым витком.

3.3.3. Стандартные фигуры

В CorelDRAW имеется большой набор стандартных фигур, которые сгруппированы на панели рис. 3.10.

Чтобы нарисовать стандартную фигуру, выбрать соответствующую группу на панели (см. рис. 3.10), а затем выбрать фигуру на *Панели свойств* в меню выбора *Правильные фигуры*. Выбрать указателем точку вставки фигуры и перетащить его в окне рисования, чтобы задать нужный размер фигуры.

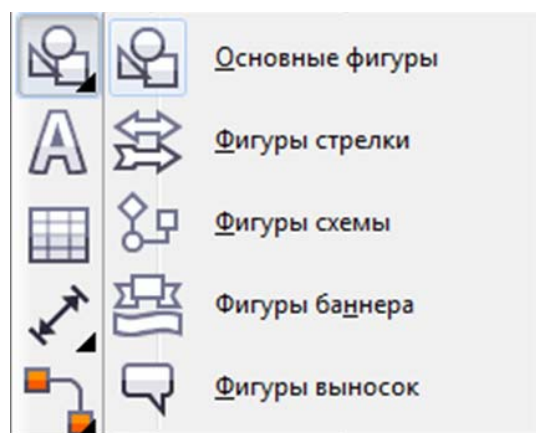


Рис. 3.10

Для настройки формы многих стандартных фигур можно использовать ее *глиф* (в виде ромбика красного цвета), перетащив его (рис. 3.11).

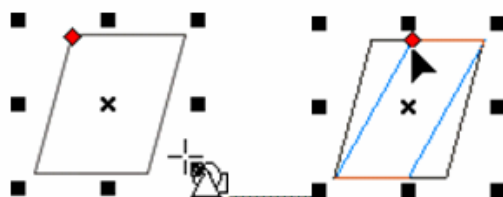


Рис. 3.11

При выделении большинства стандартных фигур, помимо черных квадратных маркеров, можно увидеть один или несколько красных ромбов, расположенных в разных частях объектов. Перемещая их с помощью мыши, можно менять форму фигур.

Если в фигуре есть один или два ромбика, то можно менять расположение и кривизну линий, перетаскивая мышью связанные с ними ромбики.

Кроме изменения формы, можно изменить размер автофигуры, выполнить перекося или повернуть ее. Все эти операции выполняются так же, как и для любого другого объекта.

Для создания произвольного контура может потребоваться полноценное редактирование узлов контура объекта. При этом автофигуры, эллипсы, прямоугольники и другие стандартные фигуры должны быть предварительно преобразованы в произвольные кривые линии.

Внутри или снаружи стандартных фигур можно добавлять текст.

3.4. Свободная форма, прямая через две точки, кривая Безье, художественное оформление, перо, В-сплайн, ломаная линия, кривая через три точки

На панели (рис. 3.12) расположены команды: *Свободная форма*, *Прямая через две точки*, *Кривая Безье*, *Художественное оформление*, *Перо*, *В-сплайн*, *Ломаная линия*, *Кривая через три точки*.

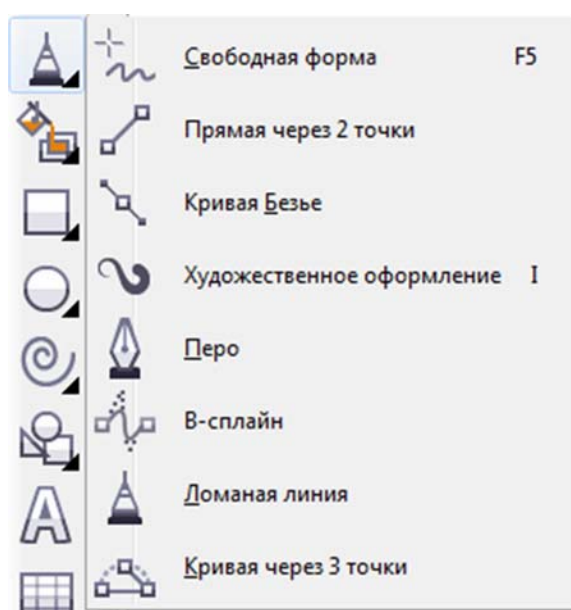


Рис. 3.12

Инструменты *Свободная форма* и *Ломаная линия* служат для построения линии в режиме произвольных кривых (прямых и изогнутых).

Инструмент *Свободная форма* позволяет рисовать с помощью мыши, как это делается карандашом на бумаге. Поместить указатель мыши на рабочее поле, нажать левую кнопку мыши и, не отпуская ее, начать передвигать мышь. При отпускании левой кнопки мыши на экране будет нарисована кривая линия, повторяющая передвижения мыши. Для стирания нарисованных сегментов следует перетащить указатель мыши назад при нажатой клавише *Shift*.

Если требуется быстро построить сложную ломаную линию, легче использовать инструмент *Ломаная линия*. При рисовании ломаных линий, состоящих из прямых сегментов, конечную точку каждого сегмента следует отмечать одинарным щелчком. Чтобы нарисовать изогнутый сегмент, щелкните точку начала сегмента, а затем, не отпуская клавиши, перетасщите указатель по странице рисования. Для завершения рисования, необходимо выполнить двойной щелчок.

Инструмент *Прямая через 2 точки* требует указать мышкой первую точку прямой, а затем, не отпуская кнопки, указать ее вторую точку. При перетаскивании в *Строке состояния* выводятся длина и угол сегмента. При добавлении нового сегмента к линии также отображается общая длина всех сегментов. Можно нарисовать прямую через две точки, перпендикулярную либо по касательной к существующему объекту или другой прямой, а также использовать любые другие режимы привязки.

Кривая Безье является гладкой кривой и проходит через две опорные точки (узлы). Две другие контрольные точки не лежат на кривой, но их расположение определяет кривизну кривой, поэтому эти точки иначе называются управляющими точками, а линия, соединяющая управляющую и опорную точки, – *управляющей линией*.

Кривая начинается в первой опорной точке (точка А на рис. 3.13,а), касаясь отрезка своей управляющей линией, и заканчивается в последней опорной точке (точка В на рис. 3.13,а). Это позволяет гладко соединять две кривые Безье друг с другом. Кривая Безье сохраняет свою форму при масштабировании.

Чтобы начать рисование, щелкнуть в окне рисования.

Чтобы нарисовать прямую линию, указать точку, в которой требуется закончить линию, а затем еще раз щелкнуть кнопкой мыши. Чтобы закончить рисование, нажать клавишу *Пробела*.

Чтобы нарисовать кривую, указать точку, в которой требуется ее закончить и, удерживая указатель, передвинуть мышь. На экране

появится пунктирная линия, длина и направление которой меняются с передвижением мыши. Эта пунктирная линия называется манипулятором кривизны, который определяет степень кривизны кривой в точке.

При рисовании кривой рис. 3.13,а,б после указания левой клавишей мыши первой точки А необходимо отпустить мышь и указать вторую точку, а затем, не отпуская кнопки мыши, вытянуть *управляющую линию* в направлении обхода по касательной к дуге в этой точке до тех пор, пока не будет относительно точно получена требуемая кривая.

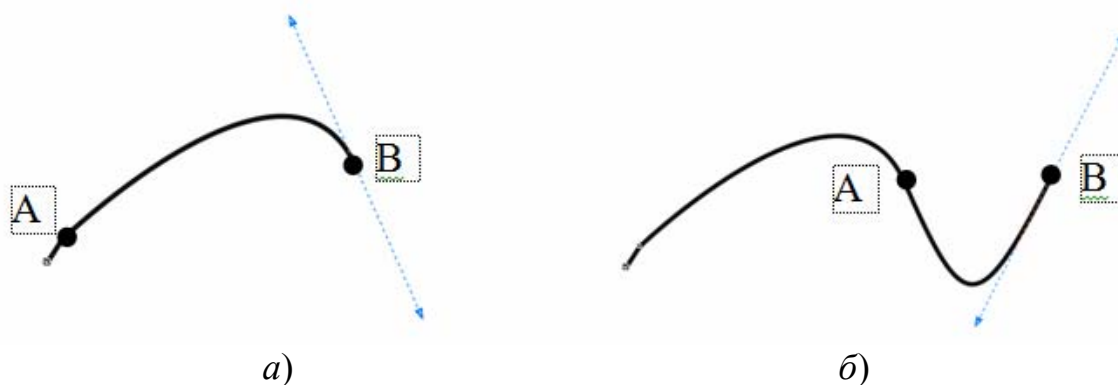


Рис. 3.13

Аналогично рисуются другие фрагменты кривой. Чтобы закончить рисование, нажать клавишу *Пробел*.

Данный инструмент удобно использовать для воспроизведения контура картинki с гладкими контурами.

Инструмент *Художественное оформление* позволяет имитировать эффект каллиграфического пера при рисовании линий, различающихся по толщине (в зависимости от направления линии и угла наклона кончика пера). По умолчанию каллиграфические линии отображаются как замкнутые фигуры, нарисованные карандашом. Можно управлять толщиной каллиграфической линии, изменяя соотношение угла рисования линии и выбранного угла наклона каллиграфического пера.

В CorelDRAW существуют различные заготовки линий, с помощью которых можно создавать широкие мазки разнообразных форм. После того как нарисована каллиграфическая линия или заготовка линии, для нее так же, как и для любого другого объекта, можно применить заливку.

Инструмент *Перо* позволяет рисовать линии по сегментам, точно размещая каждый узел и контролируя форму каждого изогнутого сегмента. Чтобы нарисовать изогнутый сегмент, щелкнуть точку размещения первого узла, а затем перетащить ручку управления

в точку размещения следующего узла. Отпустить кнопку мыши, а затем перетаскивайте ручку управления для создания нужной кривой. Чтобы нарисовать прямой сегмент, щелкнуть в точке, где требуется начать линейный сегмент, а затем – в точке, где его требуется закончить. Чтобы завершить рисование линии, дважды щелкнуть кнопкой мыши.

Инструмент *В-сплайн* позволяет рисовать гладкие, непрерывные изогнутые линии относительно начальной и конечной управляющих точек. В отличие от кривых Безье управляющие точки внутри сплайна являются «плавающими» (рис. 3.14) и не находятся на самой линии, но определяют вид кривой В-сплайна. Для завершения рисования линии дважды щелкнуть кнопкой мыши.

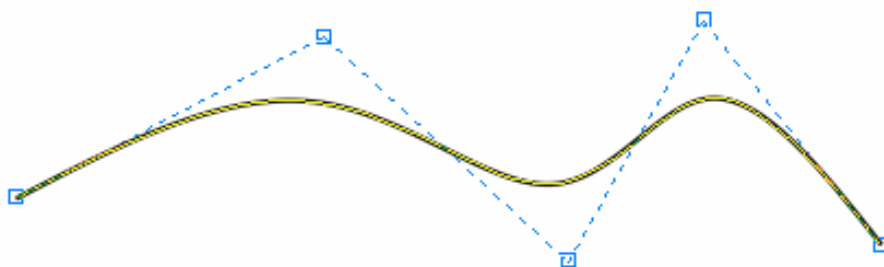


Рис. 3.14

Инструмент *Кривая через 3 точки* позволяет рисовать простые изогнутые линии, указав их ширину и высоту. Щелкнуть в начальной точке кривой (на рис. 3.15 точка *A*) и, не отпуская клавиши, перетаскать указатель мыши в конечную точку кривой (точка *B*). Отпустить кнопку мыши и щелкнуть в том месте, где должен быть центр кривой (точка *C*).

Если положение точки *C* показывать при нажатой клавише *Ctrl*, то будет получена округлая кривая, если при нажатой клавише *Shift*, то будет получена симметричная кривая.

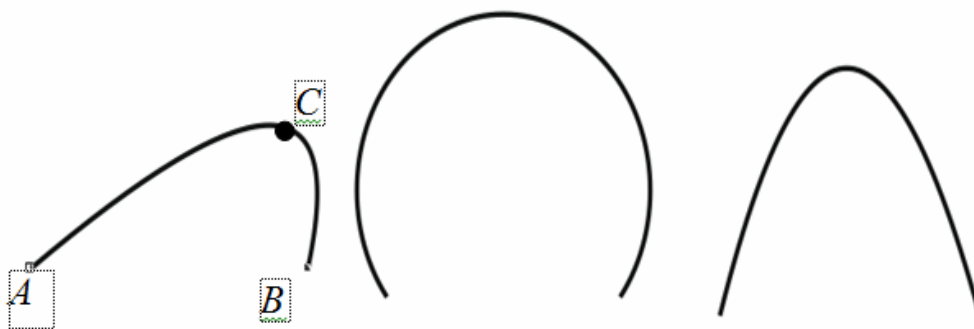


Рис. 3.15

3.5. Работа с текстом в программе CorelDRAW

По возможностям работы с текстом CorelDRAW не уступает многим текстовым редакторам. Дополнительно он позволяет выполнять сложное оформление текста и создавать оригинальные рисунки из символов.

В CorelDRAW можно создать два типа текста: *Простой текст* (текстовый блок), представляющий собой массив текста, вставленный в *текстовые фреймы*, и *Фигурный текст*, представляющий собой графический объект, с которым можно работать как с любым другим объектом.

Фигурный текст используется для ввода небольшого текста, а простой текст – для ввода больших объемов текстовой информации. Для упорядочения текстов и графических объектов в CorelDRAW можно использовать таблицы, создаваемые с помощью инструмента *Таблица*.

Для вставки фигурного текста выбрать инструмент *Текст* , щелкнуть в любом месте окна рисования и ввести текст «Фигурный». Нажать на клавишу *Enter* и ввести вторую строку текста «текст».

У фигурного текста можно форматировать как параметры шрифта, так и параметры абзаца с помощью панели свойств, с помощью диалогового окна *Редактировать текст* (рис. 3.16) и с помощью окна *Свойства объекта*. Выбрать тип шрифта *Times New Roman* и размер шрифта *14 п.*

Для ввода простого текста необходимо выбрать инструмент  и с помощью мыши определить область, в которой он будет размещен (*текстовый фрейм*).

На экране появится пунктирная рамка, а в ее начале появится текстовый курсор. Ввести в нее текст «Простой», нажать на клавишу *Enter* и ввести вторую строку текста «текст».

По умолчанию размеры текстовых фреймов постоянные независимо от добавленного в них объема текста. Если размер фрейма оказался меньше объема введенного текста (при этом цвет фрейма станет красным), то его можно увеличить вручную.

Простой текст, состоящий из одной строки, можно поместить как внутри, так и по контуру графического объекта. Нарисовать спираль и, не завершая команды, выбрать инструмент  и подвести указатель мыши к внутренней точке спирали. При этом указатель

изменит свой вид. Щелкнуть мышью, и, увидев текстовый курсор, начать вводить текст. Текст автоматически будет располагаться вдоль спирали (рис. 3.17).

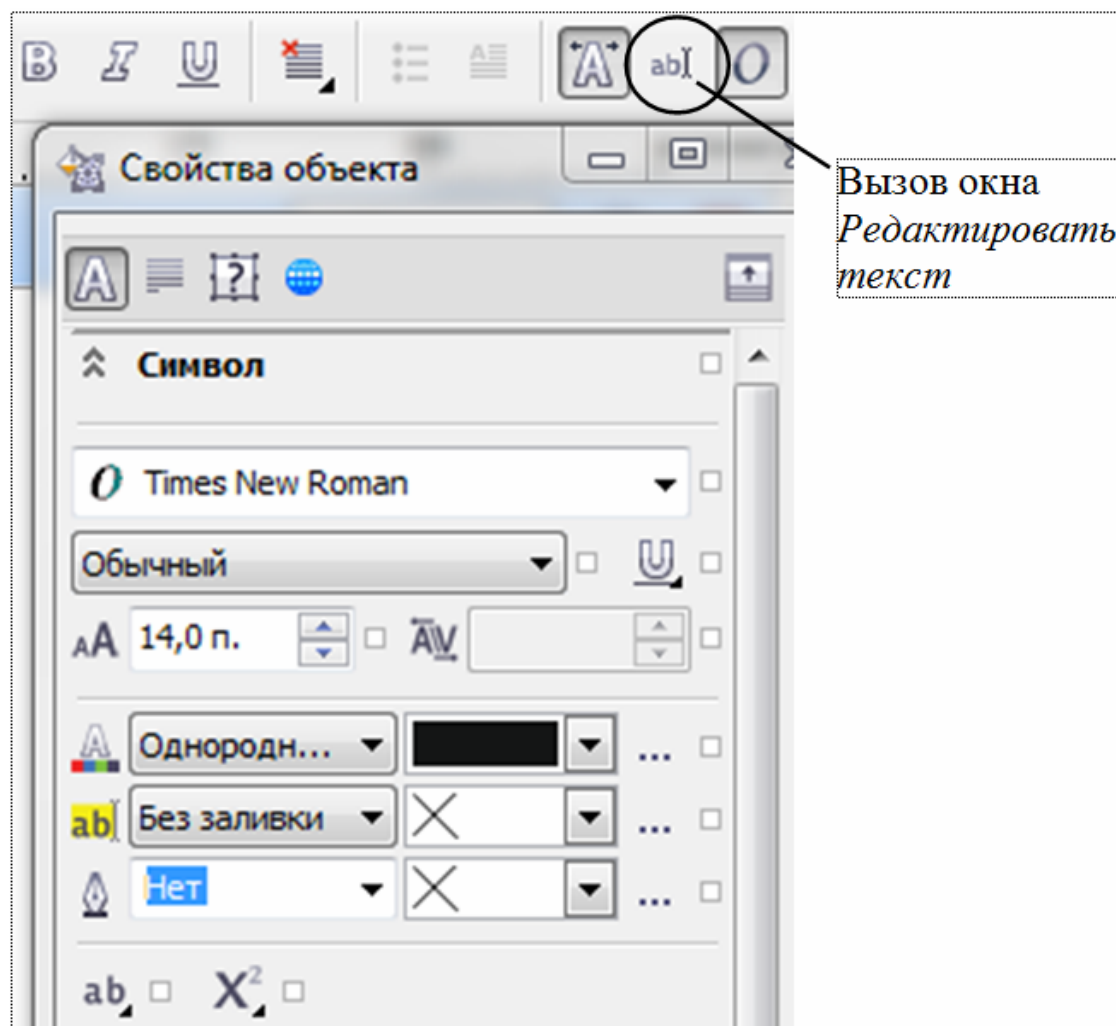


Рис. 3.16



Рис. 3.17

Нарисовать эллипс и рядом с ним на свободном месте строку фигурного текста. Для размещения уже существующего текста

вдоль кривой, например эллипса, следует выделить фигурный текст с помощью инструмента  и выбрать команду меню *Текст* ⇒ *Текст вдоль пути*. Текст будет расположен вдоль кривой, как и в предыдущем случае (рис. 3.18).



Рис. 3.18

3.6. Редактирование объектов, контура и заливки

Очень часто возникает потребность в изменении формы, контура и цвета созданных ранее объектов. В CorelDRAW можно изменить не только размеры и пропорции объектов, но и контур объекта, отредактировав узлы и сегменты, составляющие его. Изменение формы объектов в основном связано с изменением взаимного расположения и количества узлов, а также кривизны сегментов. В зависимости от способа редактирования объекты в редакторе можно разделить на две группы. Первая группа – это все объекты, которые состоят из кривых линий. Объекты этой группы можно редактировать произвольно, достигая самых замысловатых форм. Вторая группа – это стандартные объекты: прямоугольники, эллипсы и многоугольники, а также автофигуры, форму которых можно изменять в соответствии с предопределенным алгоритмом. Если же их предварительно преобразовать в произвольные кривые линии, то появляются еще и новые возможности по редактированию их узлов.

3.6.1. Редактирование объектов

Нарисовать пятиугольник и выделить его инструментом выбора . Для изменения его размера подвести указатель мыши к любому маркеру выделения, расположенному в одном из углов рамки

обрамления (рис. 3.19,а). Указатель мыши при этом примет вид двунаправленной стрелки. Нажать левую кнопку мыши, и, не отпуская ее, начать передвигать мышь. При этом у объекта появится тонкая голубая контурная копия. Передвинуть мышь правее и ниже, после чего отпустить кнопку мыши. Объект будет увеличен, при этом будут сохранены его пропорции.

Изменить размер объекта при нажатой клавише *Shift*. Убедитесь, что изменение размера будет происходить относительно центра объекта.

Изменить размер объекта только по горизонтали или только по вертикали, выполнив аналогичные действия с маркерами выделения, расположенными посередине сторон обрамляющей рамки.

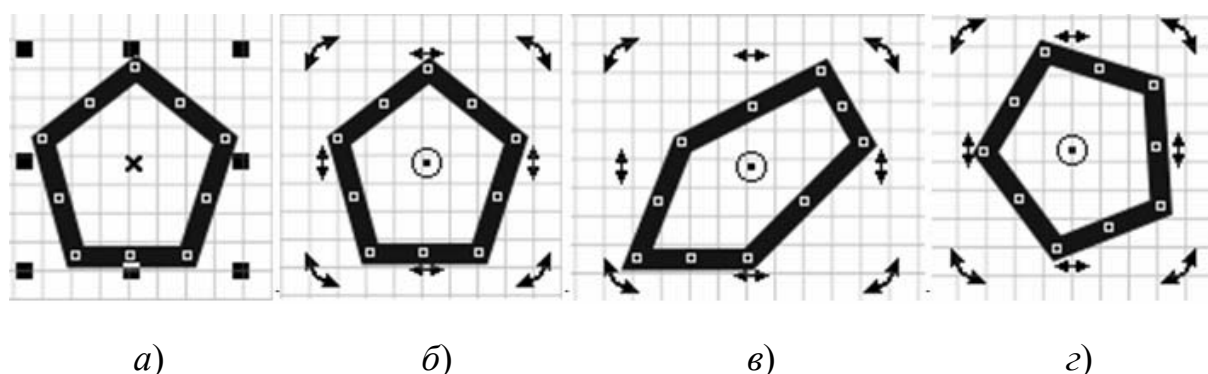


Рис. 3.19

Убедитесь, что если при уменьшении размера объекта до нуля не останавливаться и передвигать мышь дальше, через границу объекта, как бы выворачивая его, то можно получить его *зеркальное отображение*. Чтобы при зеркальном отображении сохранить размеры исходного объекта, нажать и не отпускать клавишу *Ctrl* во время операции. Получить зеркальное отображение любого объекта.

Если выполнить двойной щелчок инструментом выбора, то маркеры вокруг объекта изменятся (рис. 3.19,б).

Для *перекоса* объекта в горизонтальном направлении подвести курсор, например к горизонтальной верхней стрелке и сместить ее вправо. При этом одна из сторон объекта остается на месте, а другие стороны перемещаются (рис. 3.19,в). Последовательное выполнение нескольких перекосов может сильно исказить любой объект.

Для *поворота* объекта относительно его центра подвести курсор к любой из стрелок, расположенных по углам (в виде дуги) перетащить ее по часовой стрелке или против часовой стрелки (рис. 3.19,г). Для более сложного поворота можно изменить расположение центра вращения.

Выделить объект инструментом *Форма (Фигура)* , которая предназначена для работы с узлами. В результате по его периметру равномерно расположатся узлы, перетаскивая которые, можно формировать различные фигуры (рис. 3.20).



Рис. 3.20

Как и в других операциях CorelDRAW, перекося, вращение, изменение размеров и зеркальное отображение можно применить к группе объектов.

Нарисовать прямоугольник в рабочей области окна программы и выделить его инструментом *Форма*. Подвести указатель мыши к нижнему правому маркеру прямоугольника. Указатель примет вид стрелки (слева на рис. 3.21). Нажав левую кнопку мыши, не отпуская ее, начните передвигать мышью. Вместе с указателем мыши начнут передвигаться узлы, причем каждый из них раздвоится. Переместив узлы на нужное расстояние, отпустить кнопку мыши, углы прямоугольника останутся закругленными (в центре на рис. 3.21).

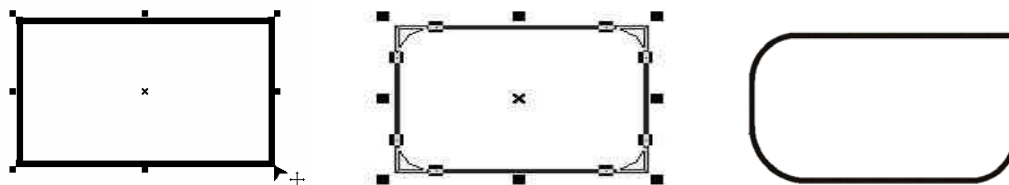


Рис. 3.21

Нарисовать еще один прямоугольник и выделить его инструментом *Форма*. Подвести указатель мыши к правому нижнему маркеру прямоугольника. Указатель примет вид стрелки. Нажать на маркер, в результате его размеры не изменятся, а другие маркеры уменьшатся. В этот момент, не сдвигая курсора, нажать левую кнопку мыши, не отпуская ее, начать передвигать мышью. Вместе с указателем мыши начнут передвигаться узлы только в нижнем правом углу прямоугольника. Переместив узлы на нужное расстояние, отпустить кнопку мыши, правый нижний угол прямоугольника останется закругленным. Прodelать то же самое для двух других углов

прямоугольника для получения фигуры, изображенной справа на рис. 3.21.

Нарисовать два одинаковых эллипса и выделить первый из них инструментом *Форма* (слева на рис. 3.22). Подвести указатель мыши к узлу, расположенному в верхней части первого эллипса. Нажав левую кнопку мыши, и не отпуская ее, начните передвигать мышью, не выводя указатель мыши за границу эллипса. Вместо одного узла появятся два, причем один начнет передвигаться вместе с мышью. Оба узла будут соединены с центром эллипса. Отпустите кнопку мыши, и из эллипса получится сектор (в центре на рис. 3.22).



Рис. 3.22

Подвести указатель мыши к узлу, расположенному в верхней части второго эллипса. Снова перетащить узел, но при этом указатель мыши должен находиться за пределами эллипса. Отпустить кнопку мыши и вместо эллипса вы увидите дугу (справа на рис. 3.22).

Следует отметить, что описанный выше способ не позволяет изменить форму эллипса произвольным образом, так как для этого сначала его следует преобразовать в кривые.

Нарисовать пятиугольник и добавить новые узлы в контур пятиугольника, для этого выбрать инструмент *Форма*. Дважды щелкнуть мышью на одной из сторон пятиугольника, примерно там, где показано стрелкой слева на рис. 3.23.



Рис. 3.23

На месте щелчка, а также в аналогичном месте на всех остальных сторонах пятиугольника, появятся новые узлы. Перетащить ближе к центру добавленный узел, чтобы получить фигуру, показанную справа на рис. 3.23.

3.6.2. Редактирование контура и заливки

При создании различных иллюстраций часто требуется настраивать контур объектов и использовать самую разнообразную заливку.

Для этого нарисовать шесть одинаковых прямоугольников, выбрать первый прямоугольник и задать требуемую толщину линий для его контура. Для этого нажать кнопку *Перо абриса*  (*F12*) либо на панели инструментов (рис. 3.24), либо на панели свойств, либо в окне *Свойства объекта*.

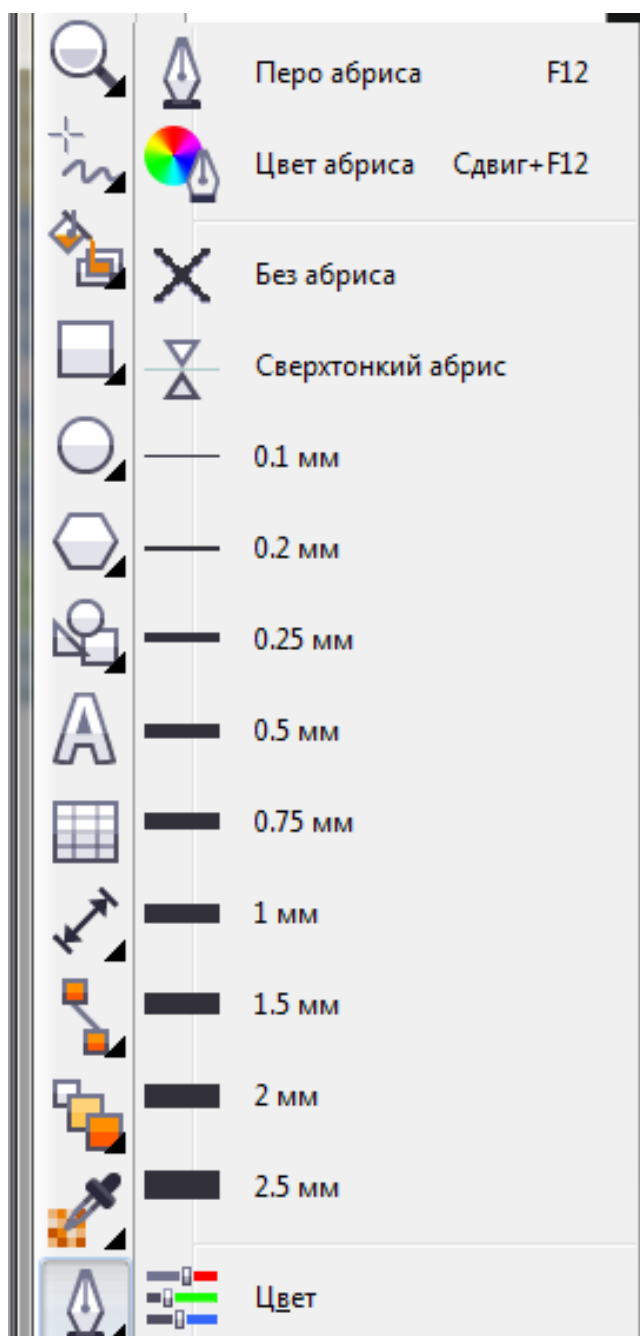


Рис. 3.24

Установить толщину контура равной 2 мм.

В CorelDRAW имеется возможность устанавливать не только толщину контура, но и много других его характеристик как с помощью окна *Свойства объекта*, так и с помощью диалогового окна *Перо абриса*, показанного на рис. 3.25.

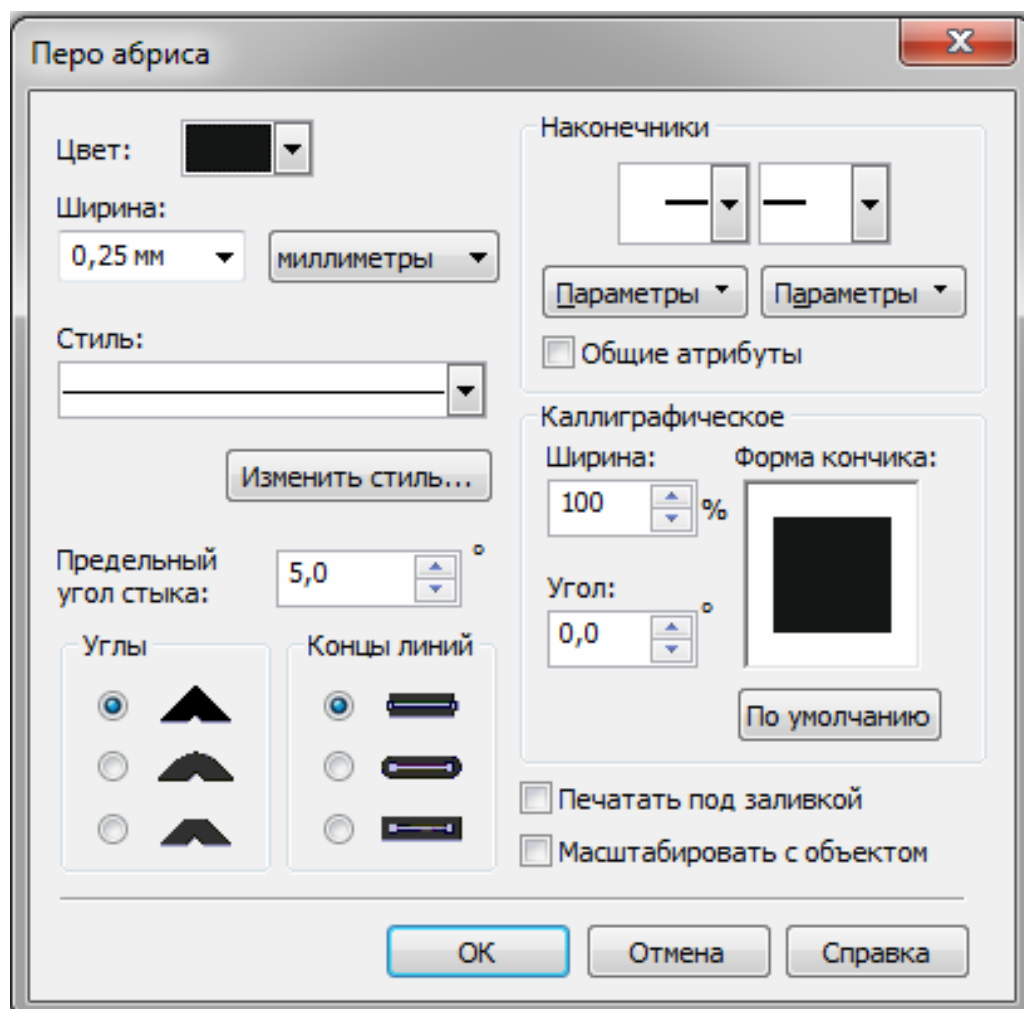


Рис. 3.25

Цвет контура можно выбрать, щелкнув правой кнопкой мыши на палитре цветов. Однако в CorelDRAW имеет более мощные средства работы с цветом.

Цвета абриса так же, как и заливки, можно выбирать из палитры документа, программ просмотра цветов или смесителей, настраиваемых палитр и т.д. Если требуется использовать цвет, уже существующий в объекте или документе, можно взять образец этого цвета с помощью цветовой пипетки, чтобы добиться полного совпадения.

На рис. 3.26 в качестве примера приведено диалоговое окно *Выбор цвета* с тремя вкладками, которое работает одинаково как при выборе цвета для абриса, так и для заливки.

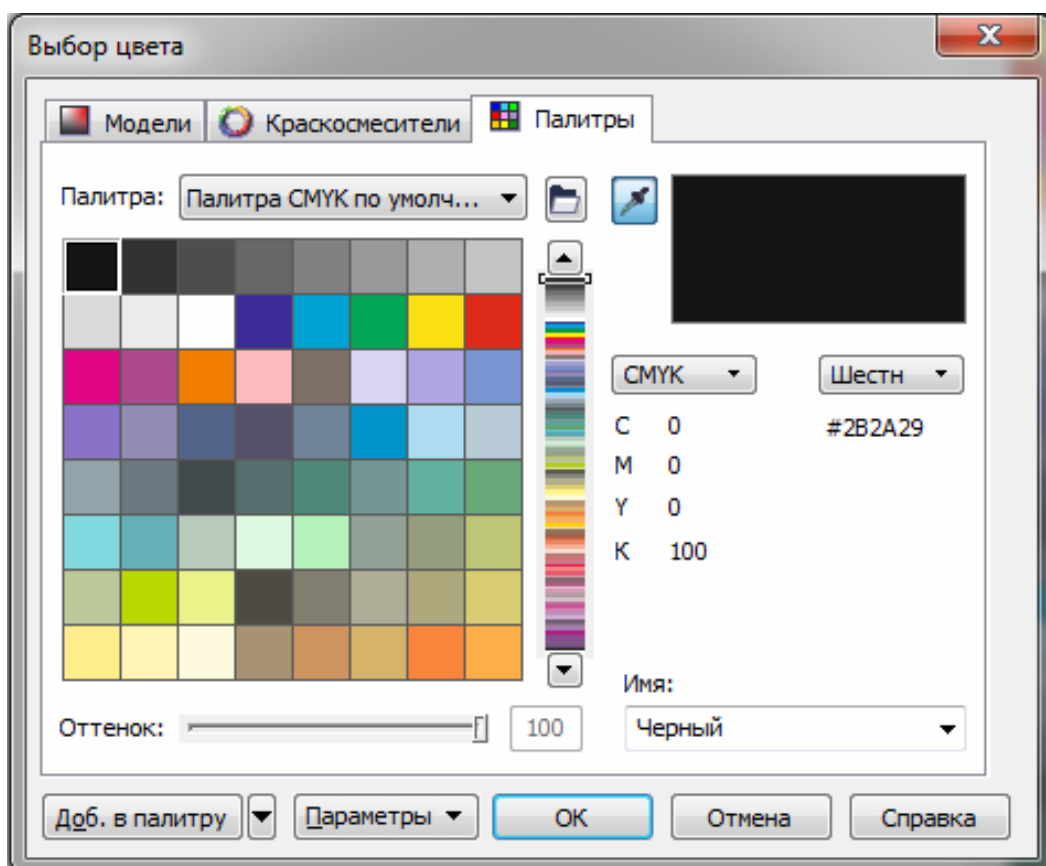


Рис. 3.26

Для заливки объектов разными цветами и узорами можно также использовать кнопку *Заливка*, расположенную на панели инструментов (рис. 3.27).

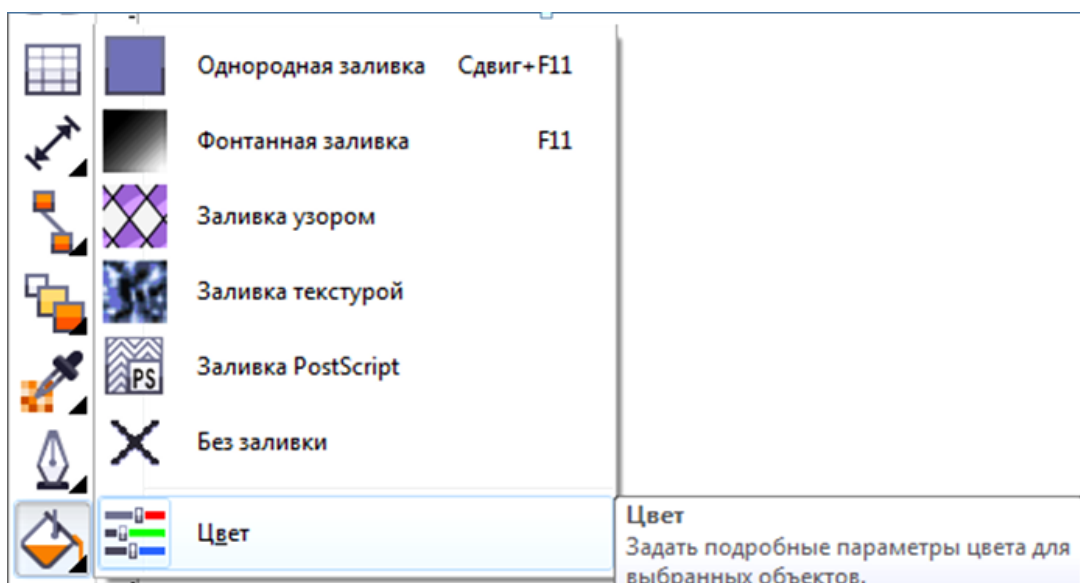


Рис. 3.27

Рядом с кнопкой появится вспомогательная панель с инструментами, предназначенными для различной заливки объектов (*Однородной*, *Фонтовой* (градиентной), *Узором*, *Текстурой* или *За-*

ливка *PostScript*). Кнопка *Без заливки* во вспомогательной панели инструментов убирает заливку объекта, делая внутреннюю область объекта прозрачной. Последний инструмент в списке вызывается окно настройки цвета.

Закрасить каждый из шести прямоугольников различными вариантами заливки, указанными на рис. 3.27. Самостоятельно поэкспериментировать с различными видами узоров.

3.7. Работа с растровыми изображениями в программе CorelDRAW

В CorelDRAW включены мощные средства для работы с растровыми изображениями. Можно импортировать растровые рисунки, преобразовывать векторный объект в растровый и трассировать растровое изображение, чтобы получить на его основе векторный рисунок, а также выполнять различные другие преобразования растровых рисунков.

Нарисовать пятиугольник, выделить его (рис. 3.28,а), и выбрать в разделе меню *Растровые изображения* команду *Преобразуйте в растровое изображение...*

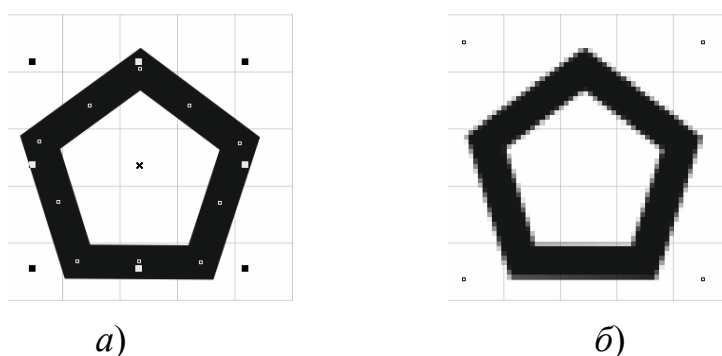


Рис. 3.28

На экране появится диалоговое окно настройки, показанное на рис. 3.29.

В группе полей *Цвет* задается требуемое количество цветов. При большем количестве цветов качество рисунка увеличивается, но возрастает и размер объекта растровой графики.

В поле *Разрешение*: задается разрешение рисунка в точках на дюйм.

Флажок *Сглаживание* предназначен для создания более гладких изображений путем удаления изломов из исходного рисунка.

Флажок *Прозрачный фон* создает прозрачный фон рисунка.

Ожидаемый размер объекта при установленных параметрах будет показан в нижней части диалогового окна.

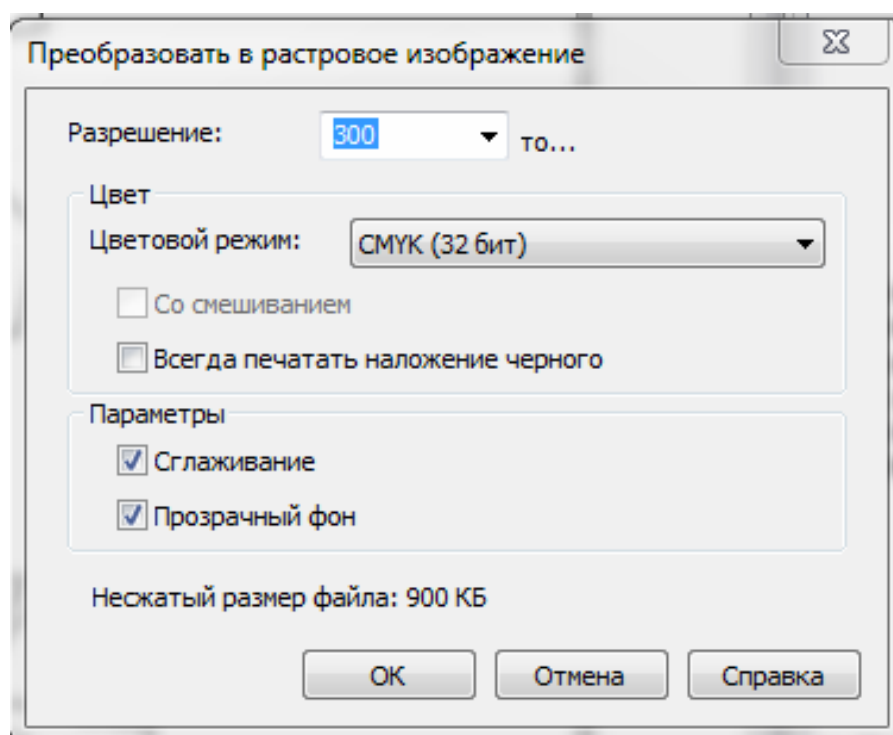


Рис. 3.29

Оставить все настройки в окне по умолчанию и нажать кнопку *ОК*. Диалоговое окно закроется, и объект будет преобразован в растровое изображение.

Выделить полученное растровое изображение (см. рис. 3.28,б) и обратить внимание, что в отличие от оригинала на его контуре отсутствуют узлы для редактирования, а также при определенном увеличении видны ступеньки по контуру, присущие всякому растровому изображению.

Растровые изображения могут быть также получены из векторных при их экспорте в такие форматы файлов растровых изображений, как JPG, CPT, PSD и т.п. При выполнении команды *Экспорт*, доступны аналогичные параметры преобразования, как и в окне на рис. 3.29.

В CorelDRAW можно импортировать растровые изображения с помощью команды *Импорт*. За один прием можно импортировать в документ CorelDRAW несколько изображений, выбрав их в диалоговом окне *Импорт*, при нажатой клавише *Shift* или *Ctrl*.

Можно импортировать и изображение не полностью, выбрав в раскрывающемся списке клавиши *Импорт* команду *Обрезать и загрузить* (рис. 3.30). В этом случае будет вызвано дополнительное диалоговое окно, в котором можно задать требуемую область рисунка.

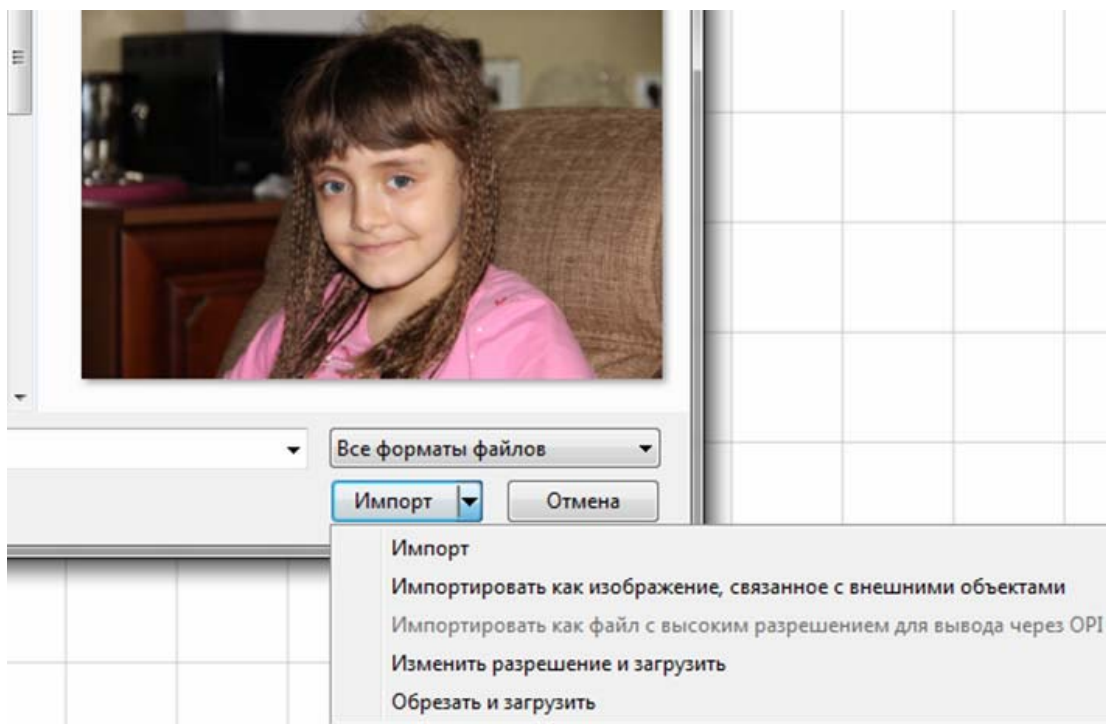


Рис. 3.30

3.8. Применение эффектов

В CorelDRAW можно использовать ряд оригинальных художественных эффектов, облегчающих создание профессиональных иллюстраций. Проще всего для создания эффектов использовать интерактивные средства CorelDRAW, сгруппированные во вспомогательной панели инструмента (рис. 3.31), но можно выбрать те же инструменты из раздела меню *Эффекты*.

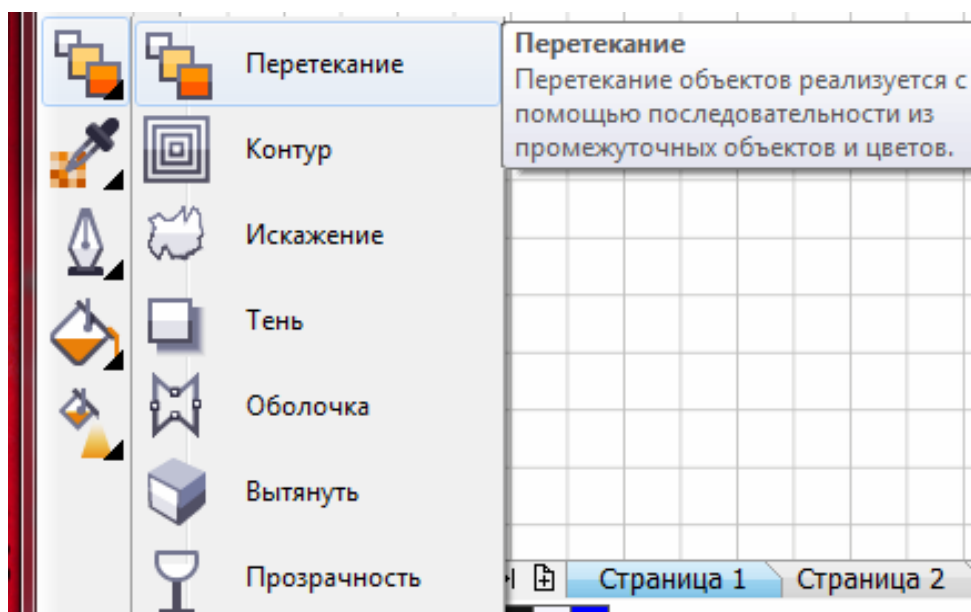


Рис. 3.31

3.8.1. Инструмент Перетекание

Эффект *Перетекание*  позволяет проследить плавный переход от одного объекта к другому. Перетекание может быть выполнено как кратчайшим путем, так и по заданной кривой. Перетекание может быть задано между объектами с разными контурами и заливкой, а также между открытой кривой и замкнутым объектом.

Нарисовать пятиугольник, а затем круг и залить его черным цветом. Выбрав инструмент *Перетекание*, перетащить указатель с первого объекта на второй. При этом форма указателя изменяется при его наведении на объект, который можно использовать для перетекания.

В результате в документе CorelDRAW появится ряд объектов, постепенно меняющих свою форму от формы начального объекта до формы конечного объекта (рис. 3.32,а).

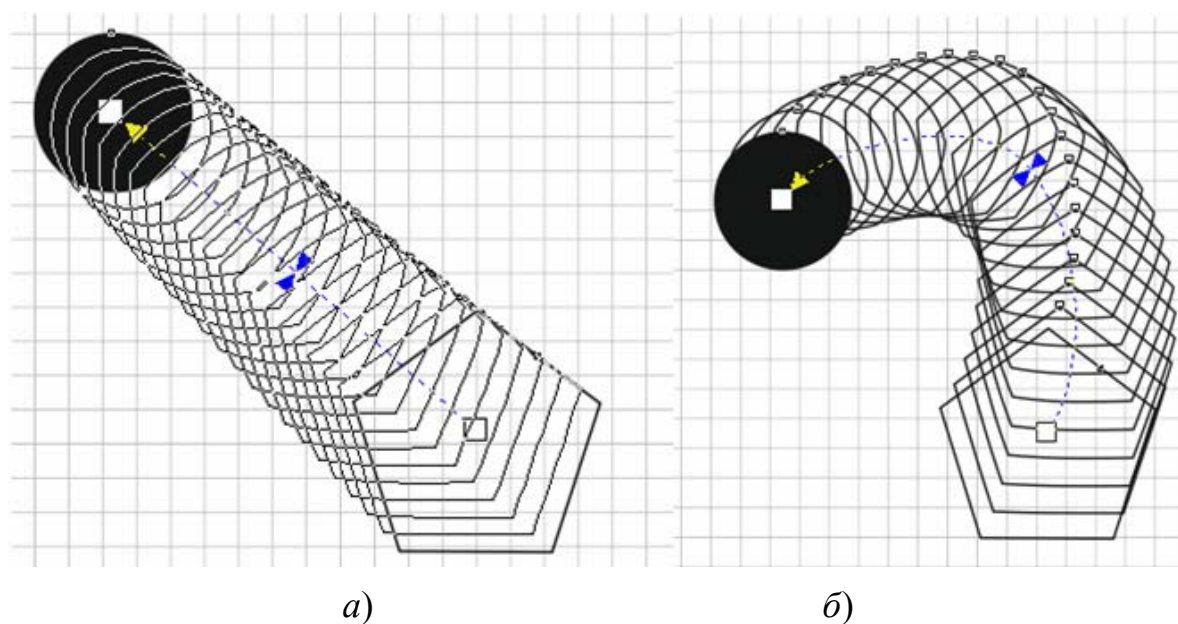


Рис. 3.32

Если для перетекания требуется нарисовать путь свободной формы, при перетаскивании указателя с первого объекта на второй необходимо удерживать нажатой клавишу *Alt* и показать нужную траекторию для преобразования (рис. 3.32,б). Расположить перетекание можно также вдоль произвольной кривой.

Параметры перетекания можно настроить с помощью элементов управления на панели свойств, задав там, например, нужное число шагов перетекания. Настройку можно также выполнить с помощью регуляторов в виде синих треугольников, расположенных на

линии, соединяющей два объекта. Верхний треугольник отвечает за скорость изменения формы, а нижний – за скорость изменения цвета. Чем ближе треугольник к конечному объекту, тем быстрее изменяются форма и цвет объекта.

По умолчанию оба треугольника связаны друг с другом и передвигаются по соединительной линии вместе. Чтобы появилась возможность передвинуть треугольники по-разному, надо дважды щелкнуть мышью на одном из треугольников. Повторный двойной щелчок вернет режим связанного передвижения.

3.8.2. Инструмент Контур

Инструмент *Контур* похож на *Перетекание*, но этот инструмент применяется к одному объекту, для которого строится его уменьшенная или увеличенная копия и происходит перетекание объекта в копию. Создать пятиконечную звезду. Выбрать инстру-

мент *Контур* , подвести указатель мыши к контуру звезды, щелкнуть объект, а затем перетащить начальный маркер в направлении к центру объекта (для создания внутреннего контура) или от центра объекта (для создания внешнего контура).

С помощью управляющих элементов панели свойств можно установить количество, ширину и цвет промежуточных форм. Можно настроить эффект и с помощью маркеров в виде ромба или квадрата (на рис. 3.33 маркеры показаны под звездой).

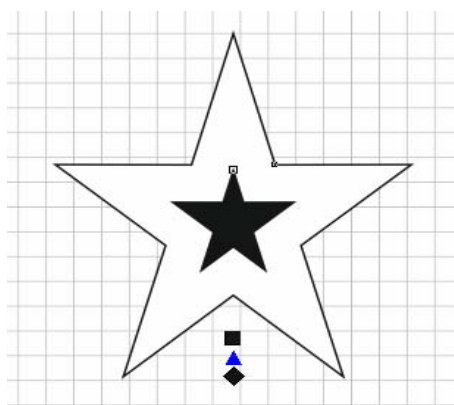


Рис. 3.33

3.8.3. Инструменты Искажение, Оболочка и Вытянуть

Инструмент *Искажение* очень сильно меняет форму объекта. Создать стандартную фигуру, например улыбающееся лицо. Вы-

брать инструмент  и установить указатель мыши на созданном

объекте, нажать кнопку мыши и, не отпуская ее, передвинуть мышь в сторону, после чего отпустить кнопку мыши. Автофигура будет искажена. Вариант искажения зависит от местоположения и длины вектора настройки искажения и от выбранного на панели свойств эффекта искажения.

На рис. 3.34 показано несколько вариантов результатов искажения.

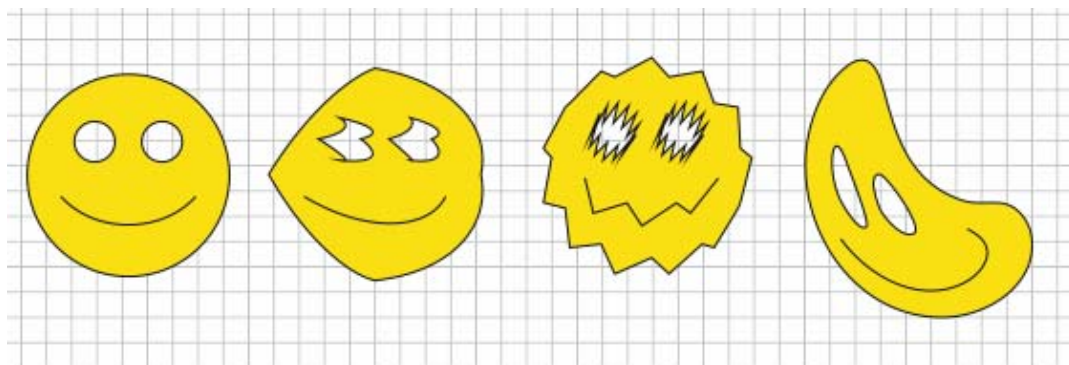


Рис. 3.34

Другим способом искажения объектов является использование инструментов *Оболочка*  и *Вытянуть* .

3.8.4. Инструмент Тень

Хотя тень объекта можно имитировать приданием ему объема, удобнее воспользоваться специальным инструментом для создания падающей от объекта тени. Так можно создавать полупрозрачные тени, размытые по краям и падающие в разных направлениях.

Создать произвольный прямоугольник, закрасить его серым цветом. Выбрать инструмент *Тень*  и установить указатель мыши в центре созданного объекта, нажать кнопку мыши и, не отпуская ее, передвинуть мышь в сторону. На экране появятся контуры тени и вектор настройки. Отпустить кнопку мыши. В результате у прямоугольника появится тень (рис. 3.35).

Перетащить конец вектора, чтобы изменить направление падения тени. Контурный прямоугольник, расположенный в середине вектора настройки, перетащить ближе к концу вектора, чтобы тень стала менее прозрачной.

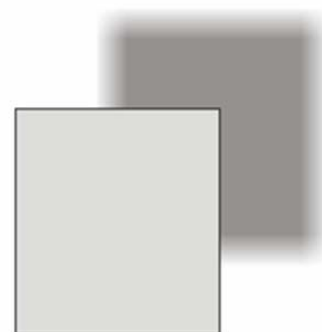


Рис. 3.35

3.9. Некоторые приемы по созданию иллюстраций

При создании иллюстраций часто бывает удобно использовать в качестве *подложки* готовые растровые рисунки (с цифровых фотоаппаратов, отсканированные или созданные средствами растровой графики). Для этого необходимо вставить оригинал картин- ки с помощью команды *Импорт*. Выделить картинку инструментом *Выбора* и понизить ее прозрачность с помощью команды *Прозрач-*

ность .

Эффекты прозрачности можно настроить с помощью соответствующих маркеров вручную либо выбрать из списков на панели свойств (на рис. 3.36 – *Однородный*, *Обычный*). В результате картинка становится равномерно прозрачной. Степень прозрачности регулируется находящимся на панели свойств ползунком либо вводом точного значения в пределах от 0 до 100 в поле рядом с ним (0 – отсутствие прозрачности, 100 – полностью прозрачный объект). Установить прозрачность равной 50.

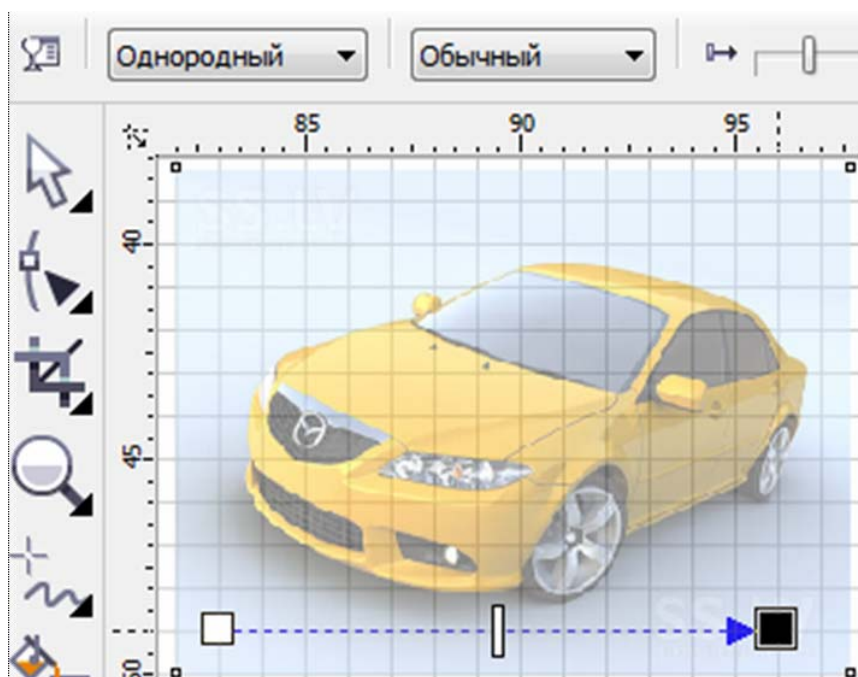


Рис. 3.36

Выделить картинку, нажать правую клавишу мышки и выбрать из контекстного меню команду *Блокировать объект*. По краям картинки появятся «замочки», говорящие о том, что она заблокирована, т.е. ее нельзя выбрать, захватить и сдвинуть с места.

Обвести контур картинки, например кузов автомобиля на рис. 3.37, используя инструмент *Безье*.

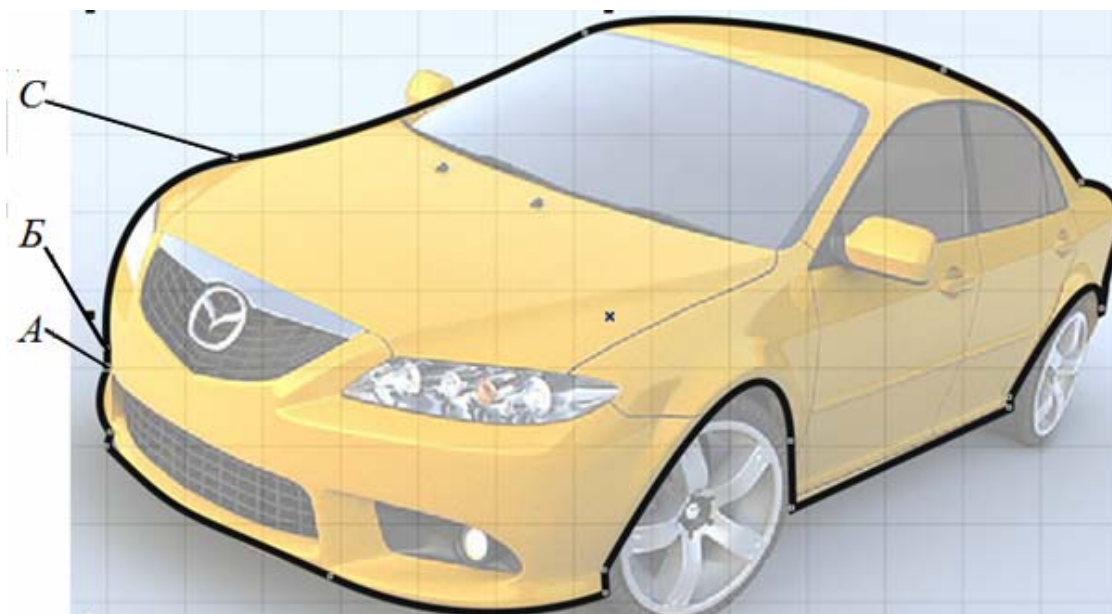


Рис. 3.37

Выбрать инструмент Безье, нажать мышкой в начальной точке (А на рис. 3.37) картинки, затем щелкнуть мышкой в конце линейного участка контура, либо рядом с начальной точкой, если рядом с ней нет линейных участков (точка В на рис. 3.37).

Затем щелкнуть в точке С и, не отпуская кнопку мыши, немного вытянуть управляющую линию по касательной к дуге в этой точке в направлении обхода контура до тех пор, пока появившаяся кривая не будет относительно точно воспроизводить прилегающий контур картинки и аналогично в окрестностях к каждой точке, где имеется излом или большой изгиб.

Если кривая в какой-то точке не должна идти по касательной к предыдущему участку, то выполнить двойной щелчок в начале этого участка, затем щелкнуть в его конце и продолжить аналогично как для точки С.

Закончить обрисовку в начальной точке, чтобы получить замкнутый контур.

Переключиться на инструмент *Выбор* и снять выделение с кривой *Безье*, нажав левой клавишей в любом свободном месте.

Для получения контура, более точно совпадающего с оригиналом, выполнить его редактирование. Перейти в режим редактирования кривой (выделить стрелкой кривую и выбрать инструмент редактирования *Форма* на панели инструментов, либо выполнить двойной щелчок мышкой по линии).

Самый быстрый метод редактирования контуров, полученных в CorelDRAW с помощью инструмента Безье, это искривление сегментов полученной кривой без использования манипуляторов кас-

тельных. Более точно подогнать контур можно, добавляя и передвигая узлы и управляющие линии кривой *Безье*.

Разблокировать картинку оригинала, наведя на нее курсор и выбрав из контекстного меню команду *Разблокировать объект*.

Все рисунки, созданные в CorelDRAW, состоят из наложенных друг на друга объектов. Порядок расположения объектов по вертикали оказывает влияние на вид рисунка.

Эффективным способом организации этих объектов является использование невидимых плоскостей, называемых *слоями*. Рисунки можно составлять, размещая объекты на различных уровнях или слоях. Например, можно символы на схеме алгоритма нарисовать в одном слое, а текст расположить в другом.

Для работы со слоями удобно использовать *Менеджер объекта* в разделе меню *Инструменты*.



Самостоятельно изучить, какие настройки могут меняться для слоев в окне *Менеджер объекта*.

Глава 4. Использование системы Mathcad в инженерных расчетах

4.1. Решения простейших задач средствами Mathcad

Математическая система Mathcad во всем мире признана одной из наиболее совершенных программных систем, позволяющих решать математические задачи в объеме программы технического вуза [13]. Начиная с версии Mathcad 2002 и до последней, доступной версии Mathcad Prime 2.0, система обеспечивает удобный интерфейс и широкий набор решаемых задач. В среде Mathcad задачи выполнения, документирования и совместного использования расчетов интегрированы в единый процесс, что существенно повышает производительность работы.

4.1.1. Основные правила задания и решения математических задач в системе Mathcad

При запуске Mathcad открывается пустой документ, в который можно добавить текст, уравнения, графики, таблицы Excel и изображения. Количество одновременно открытых документов ограничивается только ресурсами операционной системы.

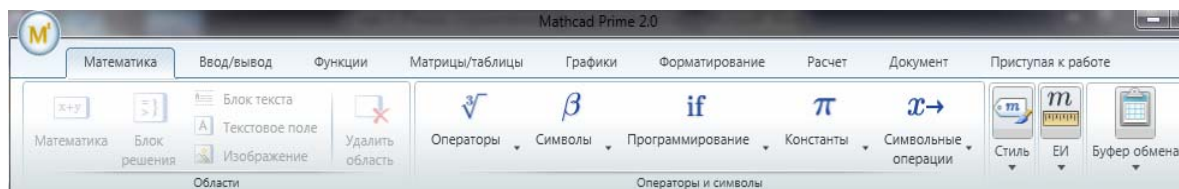
Рабочая область Mathcad содержит *окно редактирования* и *ленту*, на *вкладках* которой сгруппированы связанные команды. Окно редактирования можно настроить, уменьшив или увеличив ленту. Сведения о командах ленты можно получить из всплывающих подсказок и контекстной справки.

Вкладки служат для быстрого выполнения наиболее часто применяемых команд. На рис. 4.1,а–и показаны команды, помещенные на различных вкладках ленты.

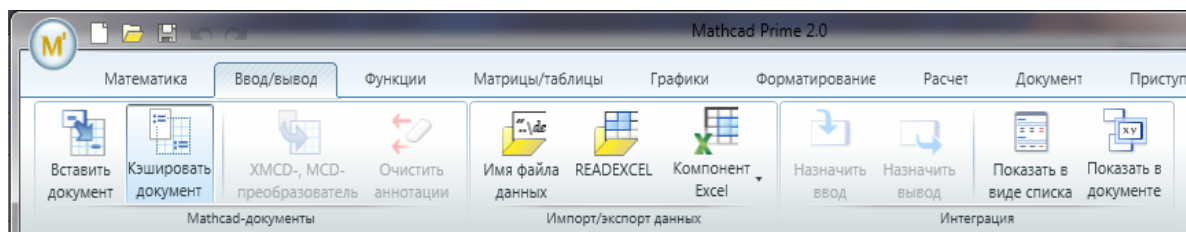
Кнопка Mathcad  содержит команды для работы с файлами, например *Открыть* (*Open*), *Сохранить* (*Save*) и *Печать* (*Print*).

Панель быстрого доступа содержит часто используемые команды. Команды можно добавлять и удалять с панели. Панель быстрого доступа может находиться как под лентой, так и над лентой.

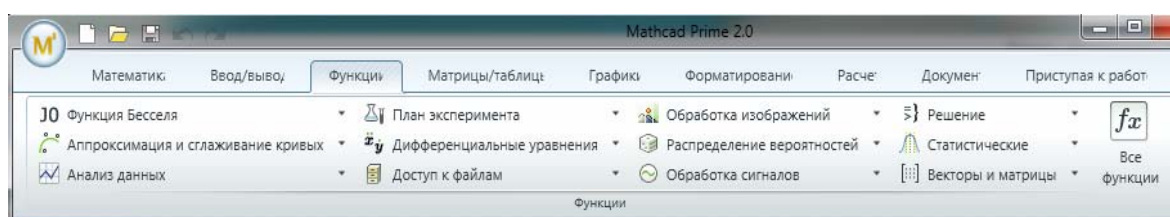
Строка состояния содержит номера страниц документа, цветной кружок, показывающий состояние кнопки «Автоматический расчет», окна «Найти и заменить», «Регулятор масштабирования», переключатели режимов «Просмотр страницы» и «Режим черновика».



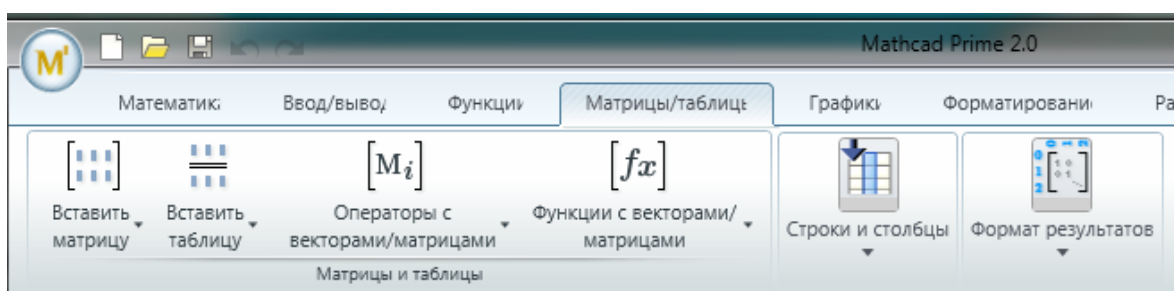
а)



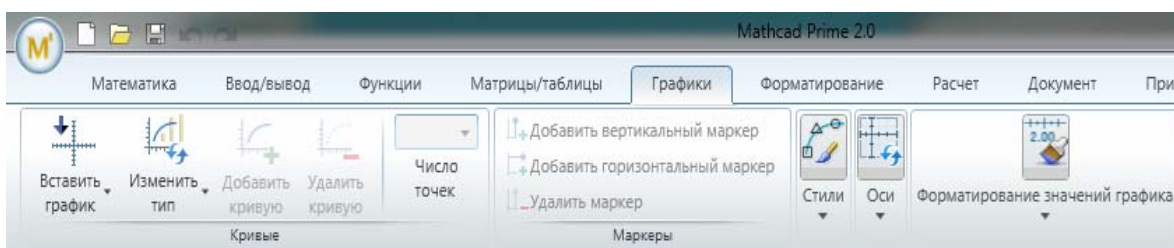
б)



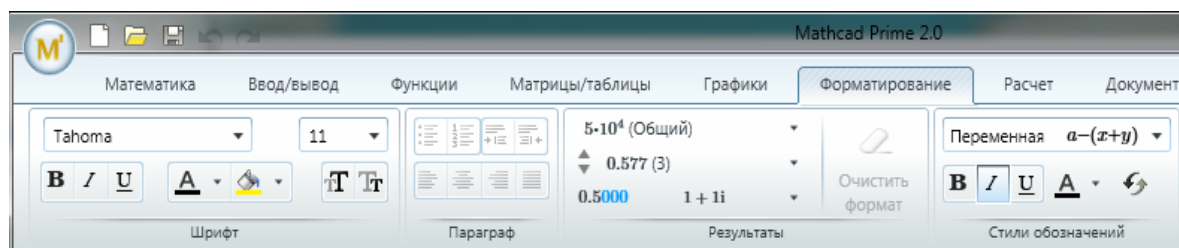
в)



г)

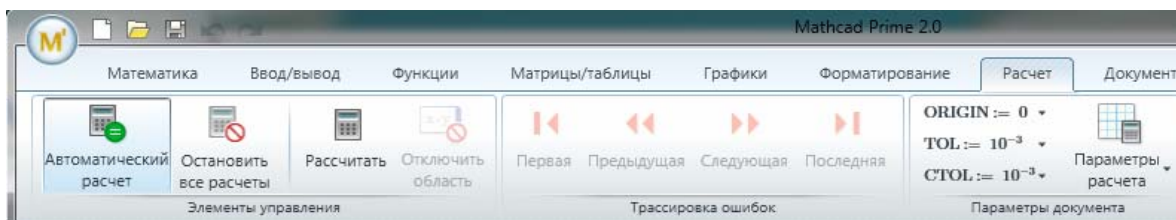


д)

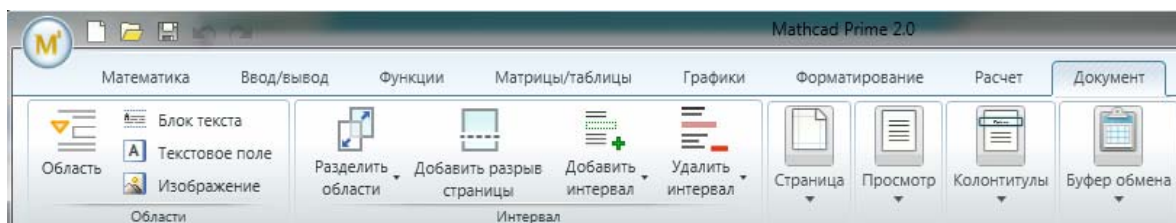


е)

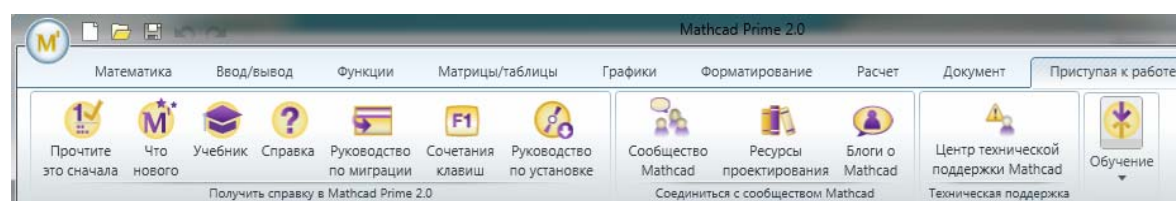
Рис. 4.1 (начало)



ж)



з)



и)

Рис. 4.1 (окончание)

Работа с системой Mathcad сводится к подготовке в окне редактирования задания на вычисления и к установке форматов для их результатов. При этом математические выражения вводятся в том виде, в котором они обычно записываются инженерами на листе бумаги. Входным языком системы является язык визуального программирования, многие записи вводятся просто путем заполнения шаблонов соответствующих операторов.

Mathcad интегрирует в себе три редактора: формульный, текстовый и графический.

Текстовые блоки позволяют создавать в документе пояснения, т.е. делать документ Mathcad более понятным. Для ввода текста необходимо щелкнуть в документе в место вставки текста, которое пометится синим перекрестием . Текст можно вставить в виде:

– *текстового блока*  **Блок текста**, который растягивается по ширине страницы, и его область не перекрывается с другими областями документа (комбинация клавиш *Ctrl+T*);

– *текстового поля* с изменяемой шириной  **Текстовое поле**, которое можно перемещать по документу, и оно может перекрываться другими областями (комбинация клавиш *Ctrl+Shift+T*).

Обе команды находятся на вкладке *Математика* (*Math*) в группе *Области* (*Regions*). После ввода или вставки через буфер обмена требуемого текста нажмите клавишу *Ввод* для перемещения на новую строку либо щелкните за пределами текстовой области для прекращения набора текста. Если изменить настройки стиля шрифта и абзаца, заданные на вкладке *Форматирование* (*Formatting*) в группах *Шрифт* (*Font*) и *Параграф* (*Paragraph*), то они будут применяться либо к выделенным, либо ко всем новым текстовым областям в документе.

Для вставки в документ изображения необходимо выбрать команду  *Изображение* на вкладке *Математика* в группе *Области* и выбрать изображение в окне *Открыть* (комбинация клавиш *Ctrl+4*).

Любая введенная область может быть удалена из документа с помощью команды *Удалить область*, также расположенной на вкладке *Математика* в группе *Области*.

Щелкнув в документе в любом месте, можно ввести *математические выражения* (цифры, буквы и математические операторы). В результате создается *Область формул*. Для ввода формулы можно



также выбрать команду *Математика* на вкладке *Математика* в группе *Области* (комбинация клавиш *Ctrl+Shift+M*).

В Mathcad можно вводить математические выражения следующих типов: алгебраические; дифференциальные и интегральные; логические; матрицы и векторы; строковые.

Щелкнуть в верхней части документа и ввести следующее математическое выражение для «определения» его значения:

$$12 + 3,2 \frac{3,8}{2,6}.$$

Обратить внимание, что в Mathcad для разделения целой и дробной части используется точка, а не запятая. При наборе выражений операторы, символы и константы можно вставить с вкладки *Математика*.

По мере набора выражения на экране отображается область формул с синей вертикальной линией в точке вставки, этот курсор можно использовать для правки математических выражений. Оператор *Вычисление* (символ « $\Rightarrow$ ») возвращает результат вычисления выражения, находящегося слева от него (рис. 4.2).

$$12 + 3.2 \cdot \frac{3.8}{2.6} = 16.677$$

Рис. 4.2

При изменении математического выражения его результат автоматически обновляется. Все связанные математические области, находящиеся справа или снизу от измененных выражений, соответственно обновляются. Если отключить расчет области на вкладке *Расчет* в группе *Элементы управления*, то результаты не будут обновляться.

Все, что начинается с цифры, Mathcad интерпретирует как число. При вводе комплексного (мнимого) числа в алгебраической форме записи за ним должны следовать символы i или j (рис. 4.3).

$$1i + 2.5i = 3.5i$$

Рис. 4.3

Для определения переменных и функций используется оператор «определения» $:=$, который можно выбрать на вкладке *Математика* в панели *Операторы* либо ввести с клавиатуры при нажатии клавиши «двоеточие». Справа от знака определения может стоять либо *числовое значение*, либо *вычисляемое выражение*.

На рисунке 4.4 приведен пример программы, определяющей, вычисляющей и выводящей на экран значение переменной f .

$$a := 2.6 \quad b := 2.5 \quad c := 3$$

$$f := a + 2 \cdot b \cdot \sqrt{c} = 11.26$$

$$f = 11.26$$

Рис. 4.4

Обратить внимание на то, что оператор определения $:=$ не возвращает значение переменной f в отличие от оператора вычисления « $=$ ». Однако в одной строке можно совмещать сначала оператор определения, а затем справа от вычисляемого выражения – оператор вычисления.

Кроме того, переменные, которые входят в вычисляемое выражение, должны быть ранее определены (либо выше, либо левее в окне редактирования). Иначе появится сообщение об ошибке (рис. 4.5).

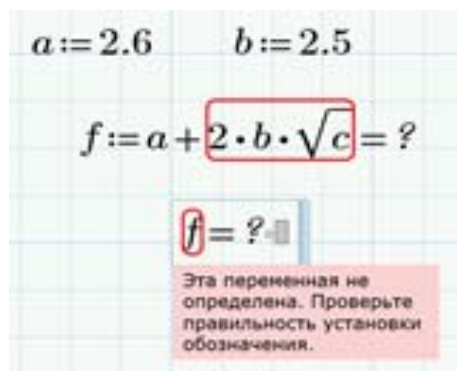


Рис. 4.5

Следует иметь в виду, что в Mathcad имена переменных и функций не могут начинаться с цифры, могут быть любой длины и содержать как русские, так и латинские буквы, кроме того, различаются строчные и прописные буквы.

В логических выражениях или ограничениях для блоков решений используется оператор «равно» ($=$), который сравнивает переменные, векторы или выражения (рис. 4.6). Обратите внимание, что в отличие от оператора вычисления он имеет более жирное начертание.

$$2 \cdot 3 = 1 + 5$$

Рис. 4.6

В Mathcad существует и *локальный оператор определения переменной-диапазона* (символ « $\equiv$ »), который определяет переменную-диапазон для оператора суммы или произведения.

Например, для нахождения произведения членов ряда следует на вкладке ленты *Математика* группы *Операторы и символы* (*Operators and Symbols*) раскрыть панель *Операторы*, найти соответствующую пиктограмму, и нажать левую клавишу мыши. В результате на поле документа выведется шаблон данной операции (рис. 4.7).

После заполнения соответствующих позиций шаблона и ввода символа « $\equiv$ » получим результат (рис. 4.8).

В данном выражении константа $e \approx 2,7$ в числителе дроби вводится из списка *Константы* (*Constants*) группы *Операторы и символы* на вкладке *Математика*.

На вкладке *Форматирование* (*Formatting*) в группе *Результаты* (*Results*) можно настроить отображение результатов для повышения точности или изменения числового формата (рис. 4.9).

Для форматирования отображения результатов необходимо выбрать одну или несколько областей для форматирования либо

щелкнуть за пределами области формул, чтобы применить форматирование результатов ко всем областям формул в документе.

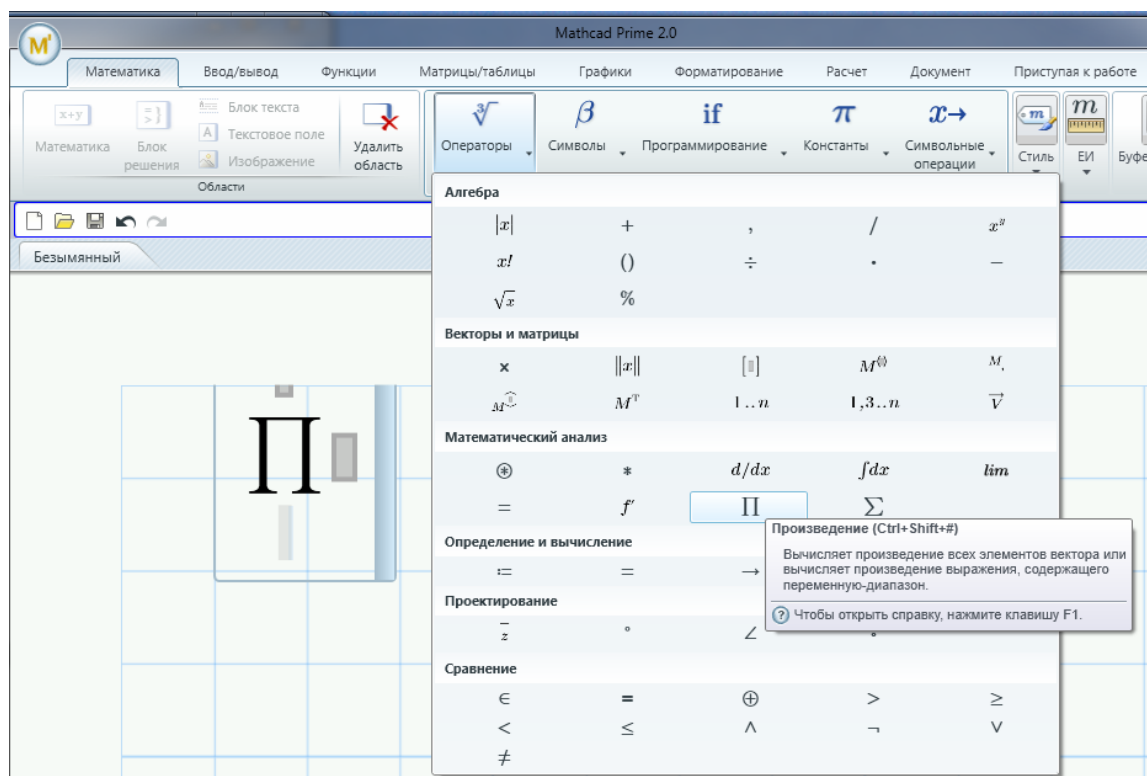


Рис. 4.7

$$\prod_{x=1}^{20} \left(\sin(x) \cdot \frac{e}{x} \right) = -2.628 \cdot 10^{-15}$$

Рис. 4.8

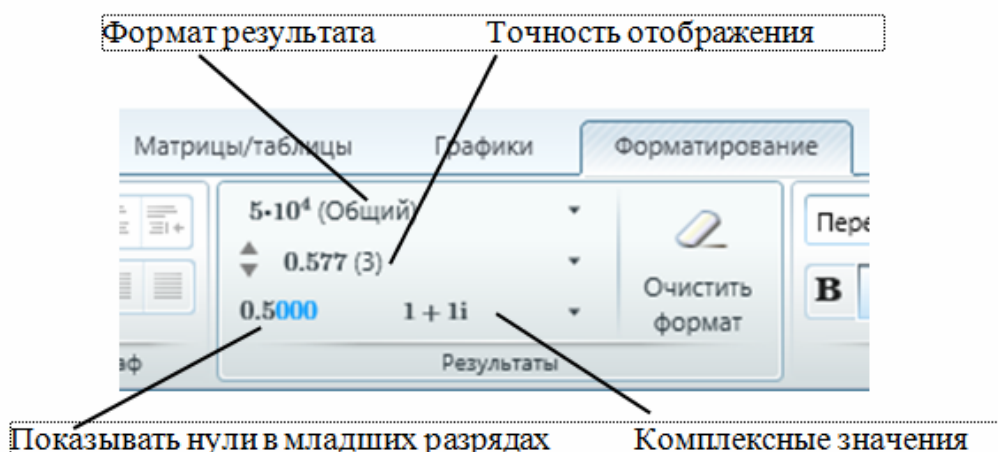


Рис. 4.9



Изучить самостоятельно все возможности по форматированию результатов вычисления.

4.1.2. Определение комплексных чисел

Результатом вычисления, а также решения уравнений могут являться *комплексные числа*, даже если все исходные значения вещественные числа. Независимо от того, в какой форме записи были заданы комплексные числа (алгебраической или показательной), Mathcad всегда возвращает их в алгебраической форме записи:

$$\underline{Z} = a + b i,$$

где $\underline{Z}$ – комплексное число (в Mathcad подчеркивание не указывается); a и b – вещественные числа, обозначающие вещественную и мнимую часть комплексного числа, соответственно; $i = \sqrt{-1}$ – мнимая единица.

Для работы с комплексными числами можно использовать следующие функции комплексного числа Z :

- функция $Re(Z)$ возвращает вещественную часть комплексного числа Z ;
- функция $Im(Z)$ возвращает мнимую часть комплексного числа Z ;
- функция $arg(Z)$ возвращает главное значение аргумента комплексного числа Z (между $-\pi$ и π , включая π). Аргумент является значением угла φ , когда Z записывается в показательной форме записи:

$$Z = |Z|e^{i\varphi}.$$

Некоторые операции над комплексными числами показаны на рис. 4.10.

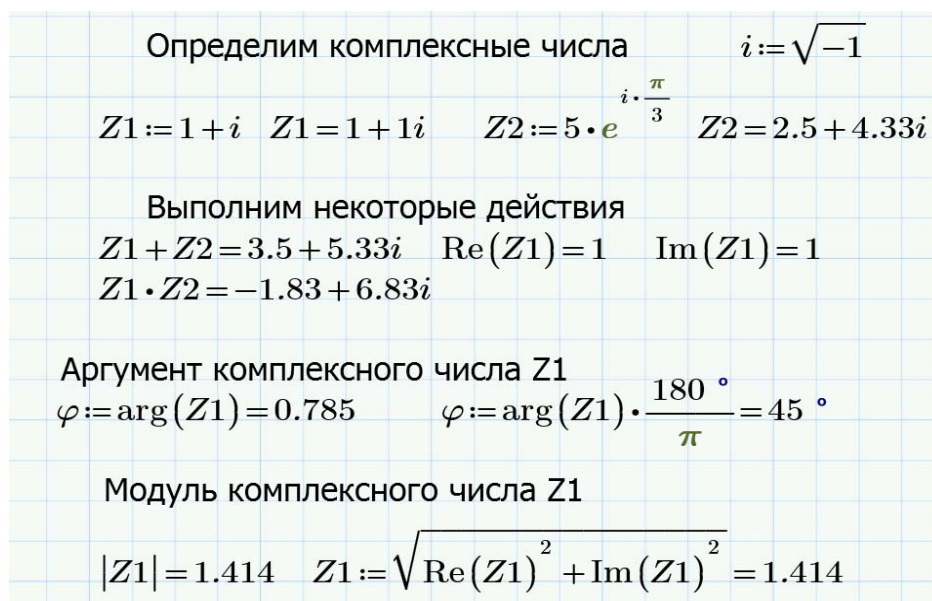


Рис. 4.10

4.1.3. Переменные диапазона

Уже в первых версиях Mathcad была предусмотрена поддержка *переменных диапазона* (ранжированных переменных), которые определяют область дискретных значений и позволяют организовывать циклические вычисления без применения операторов программирования.

Переменные диапазона используют для решения целого ряда задач: определения массивов и диапазонов суммирования или произведения, поиска решений методом последовательных приближений, построения графиков функций и таблиц.

Оператор диапазона  может быть введен в документ при нажатии два раза на клавишу «точка», а также выбран на вкладке ленты *Математика* в группе *Операторы и символы* на панели *Операторы*.

Для задания циклических вычислений с целочисленной управляющей переменной цикла (шаг изменяется на единицу) используется следующая конструкция:

$$\text{Имя переменной диапазона} := N_{\text{нач}} .. N_{\text{кон}}$$

где $N_{\text{нач}}$ – начальное значение переменной;

$N_{\text{кон}}$ – конечное значение переменной.

Если $N_{\text{нач}} < N_{\text{кон}}$, то шаг изменения переменной равен +1, а если $N_{\text{нач}} > N_{\text{кон}}$, то шаг изменения переменной равен –1.

Шаг изменения можно задать любым, используя другую конструкцию задания таких переменных:

$$\text{Имя переменной диапазона} := N_{\text{нач}}, N_{\text{след}} .. N_{\text{кон}}$$

где $N_{\text{след}}$ – следующее за $N_{\text{нач}}$ значение переменной. Шаг в этом случае равен $N_{\text{след}} - N_{\text{нач}}$.

На рис. 4.11 приведены примеры реализации циклов с заданными пределами измерения.

$a := 1 .. 5$	$b := -5 .. -1$	$c := -1, -0.5 .. 1$	$d := 1, 0.5 .. -1$
$a = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{bmatrix}$	$b = \begin{bmatrix} -5 \\ -4 \\ -3 \\ -2 \\ -1 \end{bmatrix}$	$c = \begin{bmatrix} -1 \\ -0.5 \\ 0 \\ 0.5 \\ 1 \end{bmatrix}$	$d = \begin{bmatrix} 1 \\ 0.5 \\ 0 \\ -0.5 \\ -1 \end{bmatrix}$

Рис. 4.11

4.1.4. Использование векторов, матриц и таблиц

В переменных диапазона невозможно осуществить доступ к произвольному элементу представляемого ими ряда. Этой цели служат *массивы*. Наиболее распространены одномерные массивы – *векторы* и двумерные – *матрицы*.

В Mathcad массив задается именем, как и любая переменная. Вектор имеет ряд элементов с определенным порядком расположения. Порядковый номер элемента задается индексом.

Нижняя граница индексации определяется значением системной переменной *ORIGIN*, которая по умолчанию равна 0. Влияние значения этой системной переменной показано на примере рис. 4.12.

$\text{ORIGIN} := 0$	$M := \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$	$M_2 = 3$
$\text{ORIGIN} := 1$	$M := \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$	$M_2 = 2$

Рис. 4.12

Для вставки шаблона вектора в документ необходимо на вкладке ленты *Матрицы/таблицы* выбрать команду *Вставить матрицу*, указав для нее один столбец и требуемое число строк.

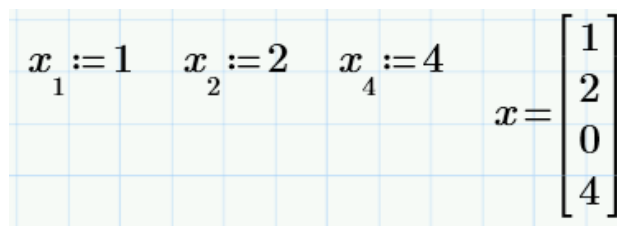
Для указания подстрочных индексов после имени переменной вводится знак открывающей квадратной скобки либо можно выбрать соответствующий шаблон на вкладке *Математика* в панели *Операторы*.

Элементы матриц также являются индексированными переменными, имена которых совпадают с именами матриц. В этом случае для каждой индексированной переменной указываются два индекса: первый – для номера строки, второй – для номера столбца.

Как векторы, так и матрицы можно создать и без введения шаблона путем определения их отдельных элементов. На рис. 4.13 приведен пример создания вектора, в котором были определены только первый, второй и четвертый элементы, а третий элемент по умолчанию равен нулю.

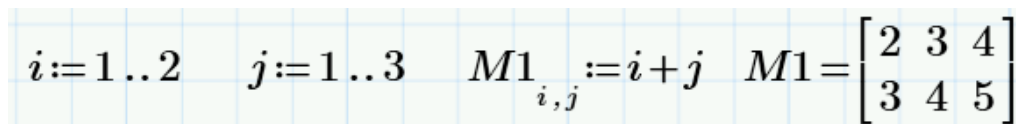
На рис. 4.14 приведен пример создания массива, где индексы элементов массива задаются с помощью переменных диапазона i и j , которые формируют два вложенных цикла: у внешнего цикла пере-

менная цикла изменяется от одного до двух, а у внутреннего цикла – от одного до трех.



The screenshot shows a Mathcad worksheet with the following content: $x_1 := 1$, $x_2 := 2$, $x_4 := 4$, and a vector $x = \begin{bmatrix} 1 \\ 2 \\ 0 \\ 4 \end{bmatrix}$.

Рис. 4.13



The screenshot shows a Mathcad worksheet with the following content: $i := 1..2$, $j := 1..3$, $M1_{i,j} := i + j$, and a matrix $M1 = \begin{bmatrix} 2 & 3 & 4 \\ 3 & 4 & 5 \end{bmatrix}$.

Рис. 4.14

Еще один способ создания векторов – использование таблиц. В Mathcad *таблица* представляет собой область, в которой определяются наборы данных. Заголовок таблицы включает в себя имя для каждого набора данных в первой строке и под ним поле для задания соответствующей единицы измерения.

Для вставки таблицы в документ необходимо щелкнуть в месте вставки и выбрать пункт *Вставить таблицу* на вкладке *Матрицы/таблицы* в группе *Матрицы и таблицы*, а затем задать указателем нужные размеры таблицы.

Следует иметь в виду, что максимальный размер таблицы, которую можно вставить из ленты, составляет десять строк по десять столбцов. Чтобы добавить новую строку, нажмите *Shift* + «Ввод»; чтобы добавить новый столбец, нажмите клавишу *Shift* + «Пробел». Строки и столбцы можно удалять клавишей «Delete», предварительно выделив их.

Создадим таблицу, состоящую из двух столбцов: первый столбец – это напряжение U , а второй столбец – это ток I (рис. 4.14).

Выберем поле для задания единицы измерения для переменной U и введем имя единицы измерения «V». Имя единицы измерения можно выбрать на вкладке *Математика* в группе *Единицы измерения (Units)*, раскрыв пункт *Единицы измерения (Units)*. Откроется список единиц измерения (рис. 4.15), отображающий все имена встроенных единиц измерения. Щелкните требуемую единицу измерения, чтобы вставить ее.

В Mathcad имеется полный набор единиц измерения по международной системе СИ, американской системе единиц (USCS) и

системе СГС. В Mathcad все переменные отображаются в системе единиц измерения, определенной для документа.

Выберем поле для задания единицы измерения для переменной I и введем имя единицы измерения «А».

Mathcad позволяет вставить масштабный множитель перед именем единицы измерения. Если поставить перед именем единицы измерения «А» множитель «0.001», то данные в этот столбец нужно будет заносить в миллиамперах.

Введем в первый столбец таблицы данные для напряжений, приложенных к сопротивлению 100 Ом (0, 5 и 10 В), рассчитаем по закону Ома соответствующие значения тока в миллиамперах и занесем их во второй столбец таблицы (рис. 4.16).

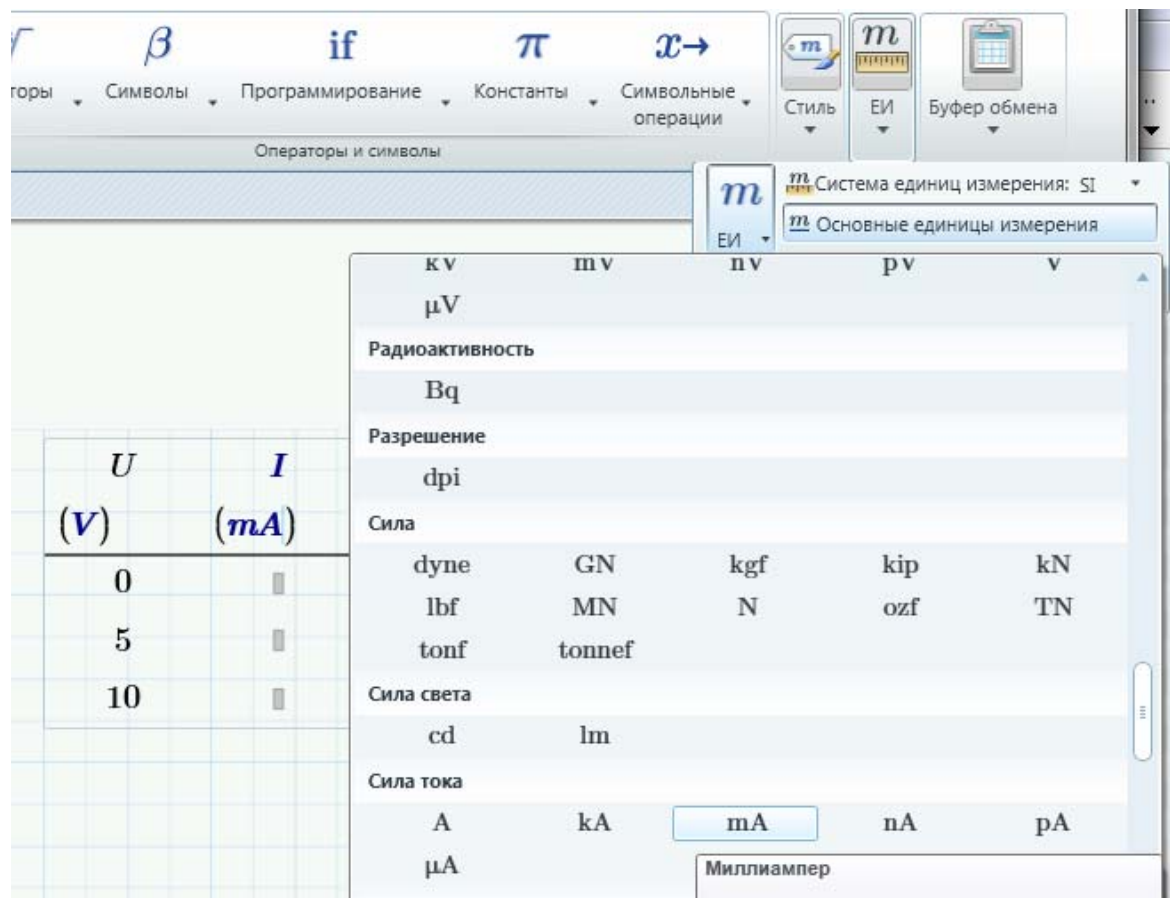


Рис. 4.15

Таблицы предназначены только для ввода данных, но не для вывода. Однако после задания наборов данных в таблице их можно использовать аналогично *векторам-столбцам*.

Вектор может возвращать свои значения с помощью оператора *Вычисление* (см. рис. 4.16), а также может использоваться в качестве аргументов функций (рис. 4.17).

U (V)	I (0.001 A)		
0	0	$I = \begin{bmatrix} 0 \\ 0.05 \\ 0.1 \end{bmatrix} \text{ A}$	Значения второго столбца в виде вектора
5	50		
10	100		
$R := 100 \text{ } \Omega$		$I := \frac{U}{R} = \begin{bmatrix} 0 \\ 50 \\ 100 \end{bmatrix} \text{ mA}$	Расчет по закону Ома

Рис. 4.16

t (s)		
0	$i := 1 \cdot e^{\frac{-t}{\tau}} = \begin{bmatrix} 1 \\ 0.368 \\ 0.135 \\ 0.05 \end{bmatrix}$	$\tau := 1 \text{ s}$
1		
2		
3		

Рис. 4.17

4.2. Построение графиков в системе Mathcad

По значениям таблицы можно также строить графики.

Для вставки графика в документ требуется выбрать место вставки графика либо ниже, либо правее соответствующей таблицы. Затем выбрать пункт *Вставить график* на вкладке *Графики* в группе *Кривые*, раскрыть список с вариантами графиков и выбрать, например, *График XY*.

В результате в документ вставится шаблон выбранного графика, в котором необходимо заполнить соответствующие позиции для X и Y .

Графики можно масштабировать. Можно изменять расположение названия осей графика, а также можно править и настраивать деления и значения делений осей. Оси графика можно задать в линейном или логарифмическом масштабе. По умолчанию масштаб осей является линейным.

На рис. 4.18 приведен график зависимости $U(I)$, построенный для таблицы из программы на рис. 4.16.

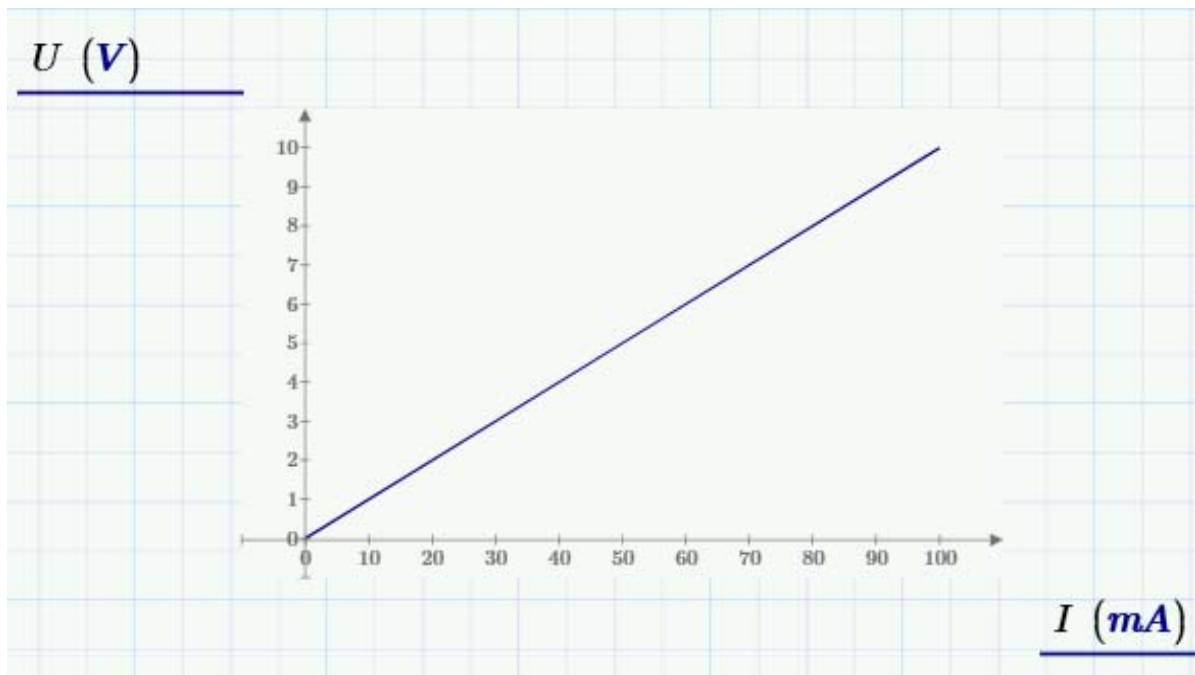


Рис. 4.18

На рис. 4.19 приведен график зависимости вектора мгновенного значения тока $i(t)$ из программы на рис. 4.17.

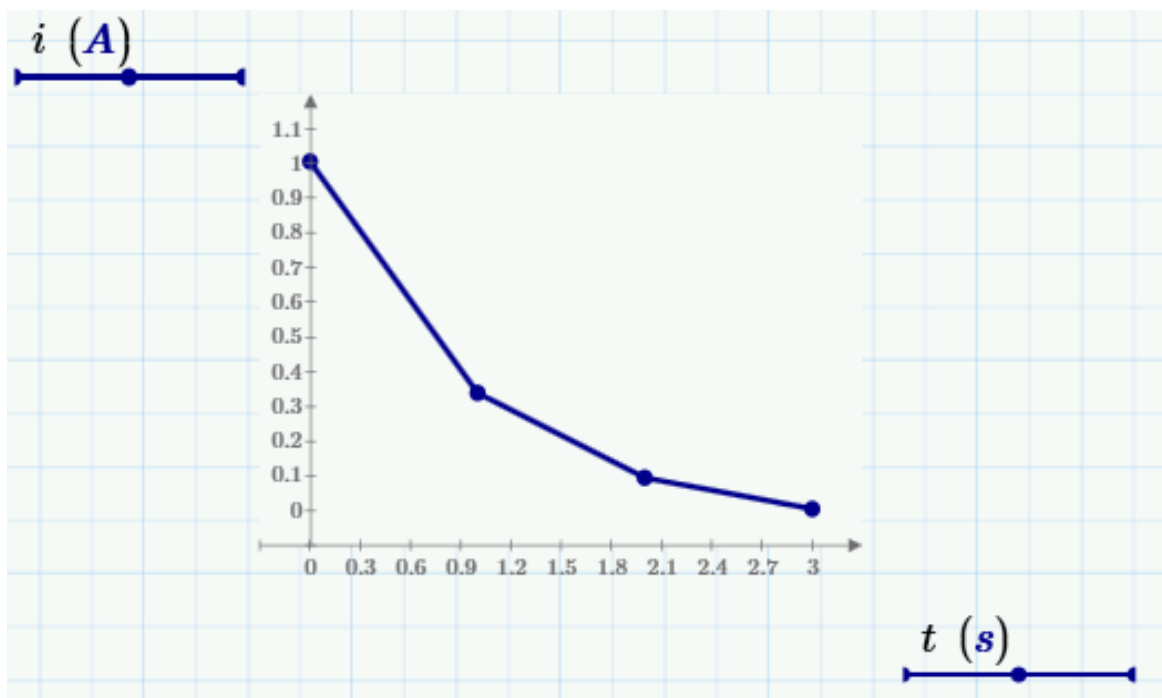


Рис. 4.19

При построении графиков удобно использовать переменную-диапазона, с помощью которой задавать не только диапазон изменения аргумента, но и его шаг, который, в свою очередь, определяет количество точек на графике (рис. 4.20).

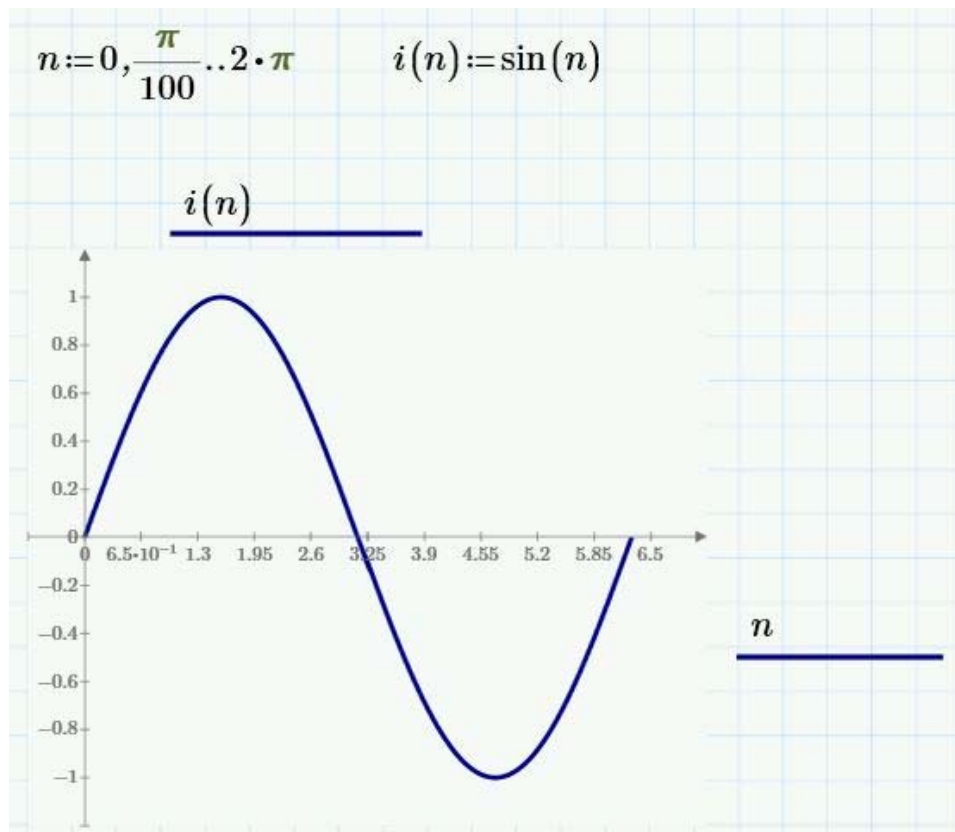


Рис. 4.20



Изучить самостоятельно другие типы графиков (полярный, контурный и 3D-график), другие варианты задания кривой, а также все возможности Mathcad по изменению настроек для графика.

4.3. Численные методы решения уравнений и систем уравнений

4.3.1. Решение систем линейных уравнений

Система линейных уравнений $M \cdot X = V$ может быть решена в Mathcad аналитически в матричной форме через обращение матрицы (рис. 4.21).

В Mathcad многие задачи решаются численными методами. Так, для решения систем линейных уравнений введена функция $lsolve(M, V)$, которая работает быстрее и в некоторых случаях дает более высокую точность. На рис. 4.21 показан пример использования функции $lsolve$, к аргументам которой предъявляются следующие требования:

$$M := \begin{bmatrix} -2 & 1 & 3 \\ 4 & 3 & -7 \\ 5 & 2 & 4 \end{bmatrix} \quad V := \begin{bmatrix} 3 \\ 6 \\ 1 \end{bmatrix} \quad X := M^{-1} \cdot V = \begin{bmatrix} -0.597 \\ 2.363 \\ -0.185 \end{bmatrix}$$

$$X := \text{lsolve}(M, V) = \begin{bmatrix} -0.597 \\ 2.363 \\ -0.185 \end{bmatrix}$$

Рис. 4.21

M – матрица коэффициентов системы должна состоять из действительных или комплексных значений. Если матрица квадратная, то ее определитель не должен равняться нулю.

V – вектор или матрица свободных членов должна состоять из действительных или комплексных значений, имеющая такое же число строк, как и матрица M .

4.3.2. Решение нелинейных уравнений

Многие *нелинейные уравнения*, например трансцендентные, и системы из них вообще не имеют аналитических решений. Однако они могут решаться *численными методами* с заданной погрешностью, определяемой переменной TOL , значение которой можно задать на вкладке *Расчет* в группе *Параметры документа*. Чтобы найти более точные значения корней, нужно уменьшить значение переменной TOL .

Если уравнение является полиномом, можно одновременно найти все корни с помощью функции $\text{polyroots}(V)$, которая возвращает вектор, содержащий корни полинома, где коэффициенты заданы параметром V .

На рис. 4.22 приведен пример использования функции polyroots для нахождения корней полинома третьей степени

$$f(x) = 2x^3 - 5x^2 + 43x - 92. \quad (4.1)$$

Функция $\text{root}(f(x1, x2, \dots), x1, [a, b])$ возвращает значение параметра $x1$, при котором функция f равна нулю. Если указаны параметры a и b , функция root находит параметр $x1$ в интервале $[a, b]$. В противном случае должно быть определено начальное приближение параметра $x1$ перед вызовом функции root .

$$V := \begin{bmatrix} -92 \\ 43 \\ -5 \\ 2 \end{bmatrix} \quad \text{polyroots}(V) = \begin{bmatrix} 0.147 + 4.564i \\ 0.147 - 4.564i \\ 2.206 \end{bmatrix}$$

Рис. 4.22

На рис. 4.23 приведен пример использования функции *root* для нахождения корней полинома (4.1). Сначала в программе задается полином $f(x)$. Далее строится график зависимости $f(x)$ для того, чтобы найти значение x для начального приближения, достаточно близкое к корню x_1 .

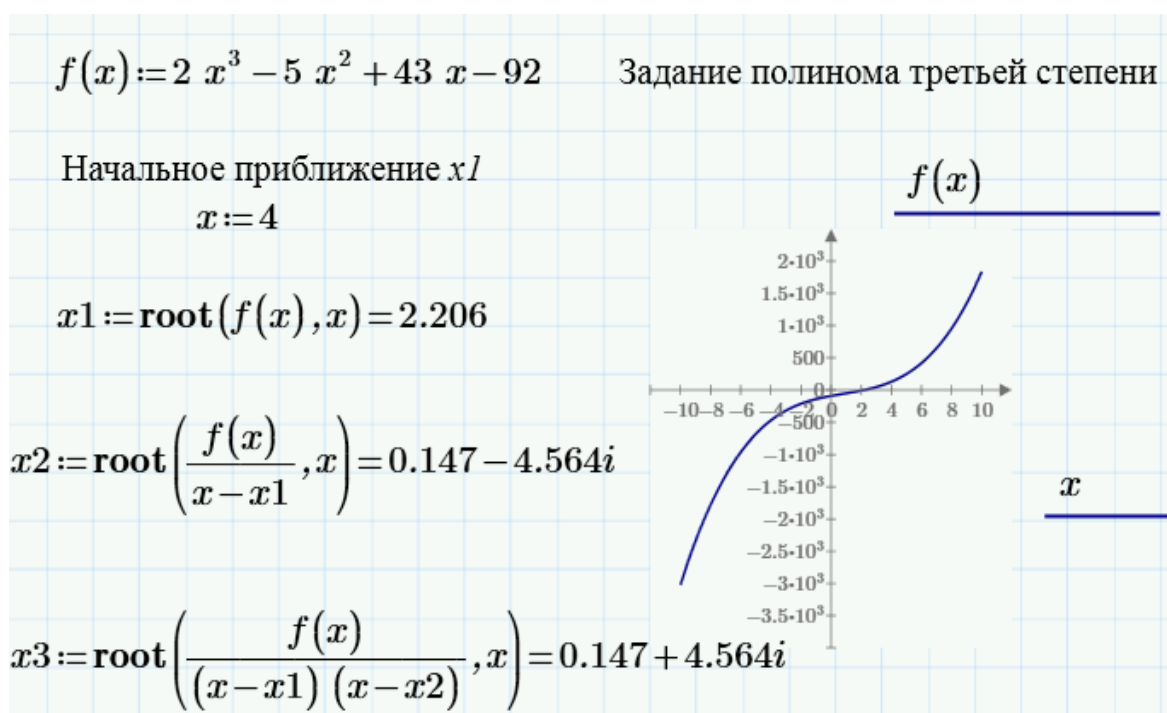


Рис. 4.23

Из графика видно, что функция пересекает ось x лишь в одной точке в интервале от 0 до 4. Зададим начальное приближение x равным 4 и найдем с помощью функции *root* корень x_1 . В случае комплексных корней на графике отображается только вещественная часть корня.

Для поиска второго корня исключим первый путем деления $f(x)$ на $(x - x_1)$ и найдем с помощью функции *root* корень x_2 . Соответственно для поиска третьего корня $f(x)$ разделим еще и на $(x - x_2)$.

Сравнивая результаты решения уравнения (4.1) в двух последних программах, можно сделать вывод, что они совпали.

На рис. 4.24 приведен пример программы для решения нелинейного уравнения $f(x) = 2 - e^{-x}$.

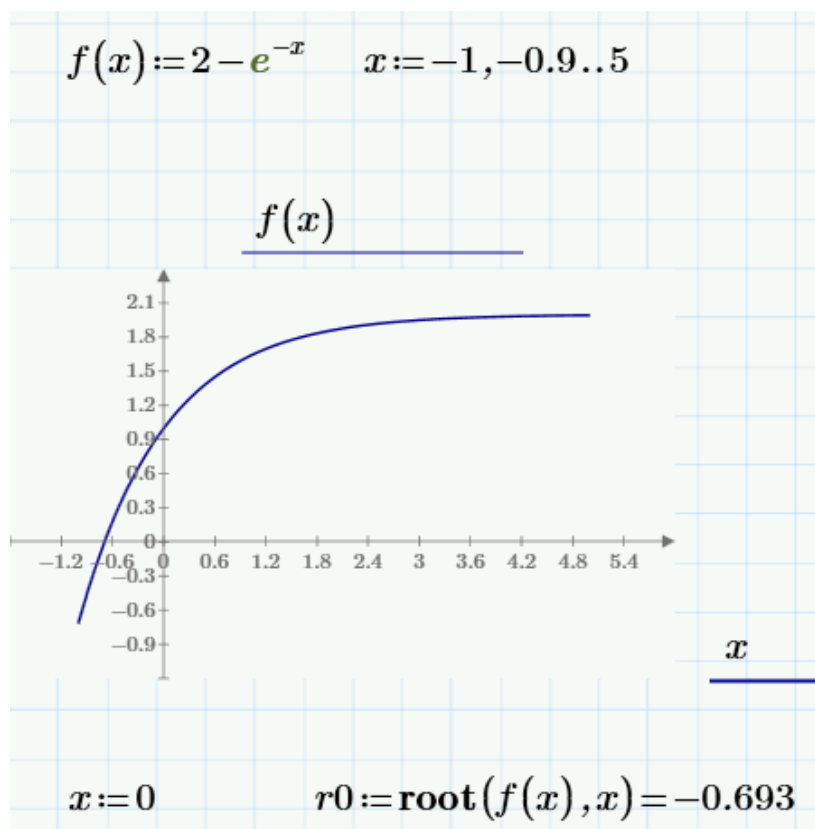


Рис. 4.24

Сначала в программе задается функция. Далее задается диапазон изменения с помощью ранжированной переменной и строится график зависимости $f(x)$ для того, чтобы найти для начального приближения значение x , достаточно близкое к корню.

Из графика видно, что функция пересекает ось x лишь в одной точке в интервале от 0 до -1 . Зададим начальное приближение x , равным 0 и найдем единственный корень с помощью функции *root*.

4.3.3. Решение систем нелинейных уравнений

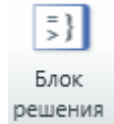
Функция *root* может решить только одно уравнение с одним неизвестным. Для решения системы из нескольких уравнений используйте функцию *find* или *minerr*.

Блок решения – это контейнер, который может содержать любые другие области, кроме блоков текста. Блоки решения используются для решения систем линейных, нелинейных или дифференциальных уравнений, а также для решения задач оптимизации. Блоки решения осуществляют поиск решения методом итераций. Часто найденное решение является аппроксимацией фактического

решения с заданной точностью. При этом решаемые уравнения выражены в явном виде.

Чаще всего Блок решения содержит только одну функцию решения, для которой необходимо задать начальные приближения, а также начальные или граничные условия. Если задача решается для n переменных, блок решения должен содержать n уравнений, в том числе в матричном представлении.

Для вставки в документ *Блока решения* необходимо щелкнуть в месте вставки блока в документе и выбрать команду *Блок решения*



(*Solve Block*) (комбинация клавиш *Ctrl+I*) на вкладке *Математика* в группе *Области*.

В результате на экране появляется область блока решения, которая содержит три раздела: *Начальные приближения*, *Ограничения* и *Решатель*.

В блоке решения используются следующие функции:

- *find*, *minerr* – решают системы линейных или нелинейных уравнений. Функция *minerr* возвращает приближенное решение в тех случаях, когда функция *find* возвращает ошибку;

- *minimize*, *maximize* – выполняют соответственно минимизацию и максимизацию заданной функции.

- *Odesolve* – решает однородные дифференциальные уравнения в блоках решения.

Функции *Блока решения* можно использовать только в его пределах. Первый символ в имени функции блока решения можно вводить без учета регистра. Переменные диапазона и аналитические преобразования функций внутри блоков решения не поддерживаются.

На рис. 4.25 приведен пример использования блока решения для решения системы нелинейных уравнений

$$\begin{cases} y = x^2 - 45; \\ y = x - 18. \end{cases}$$

Вставить *Блок решения* и ввести начальные приближения для переменных x и y (в разделе *Начальные приближения*). Для их определения удобно построить графики для уравнений, входящих в систему. Полученные точки пересечения можно использовать для дальнейшего поиска корней.

Используя логические операторы, ввести в разделе *Ограничения* уравнения, входящие в систему.

Вызвать функцию *find* блока решения, раскрыв на закладке *Функции* список *Решение*. Вставить ее в раздел *Решатель* и заполнить требуемые позиции шаблона функции *find*.

После введения оператора вычисление Mathcad отображает на экране первое решение уравнения в виде вектора *XY1*.

Если система имеет еще одно решение (что можно увидеть на соответствующих графиках), то необходимо вставить следующей блок решения, задав в нем другие значения для начальных приближений, записав еще раз решаемые уравнения и вставив функцию *Find* (см. рис. 4.25).

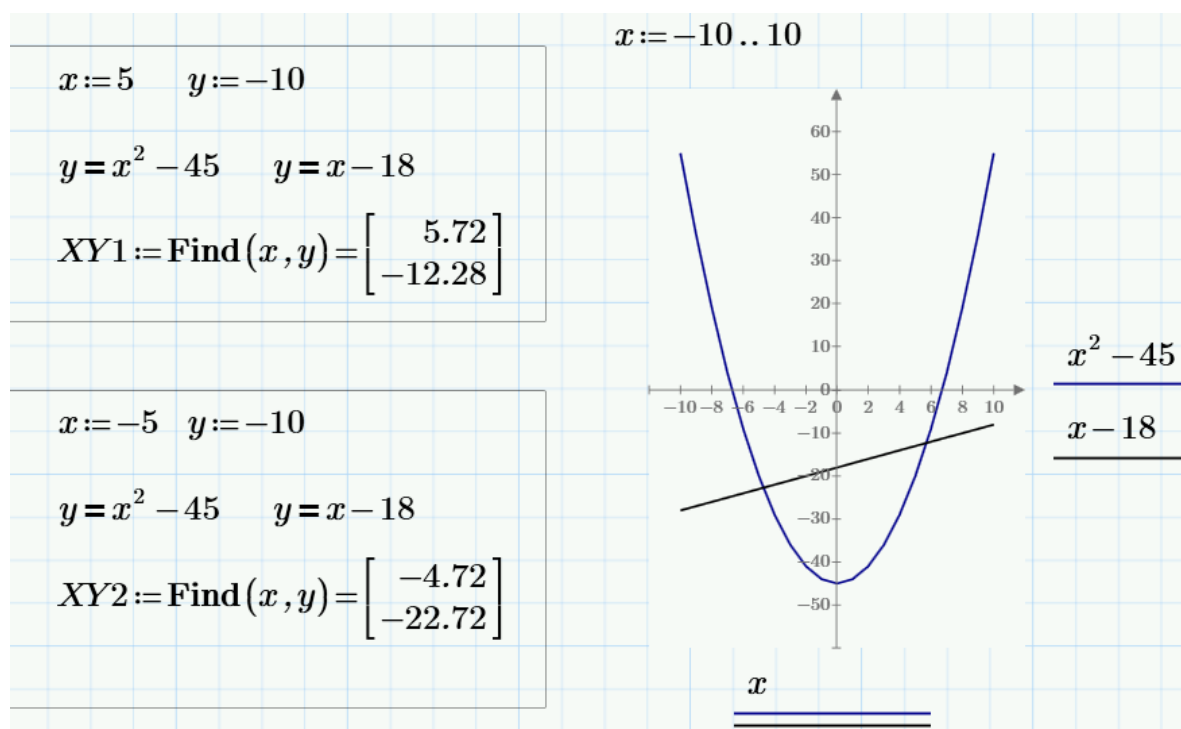


Рис. 4.25

Если система не сходится, можно найти приближенное решение с помощью функции *minerr* вместо функции *find*.

4.4. Программирование в системе Mathcad

Изначально в Mathcad для программирования использовались только несколько специфичных встроенных функций и ранжированные переменные.

В более поздних версиях Mathcad появился собственный просто и интуитивно понятный язык, который дает возможность программисту эффективно применять программный код в документах Mathcad в виде программных модулей.

Программирование в Mathcad позволяет решать самые различные, в том числе и довольно сложные, задачи: определение условий сравнения и ветвления, организация циклических расчетов, проведение итерационных расчетов и т.д.

Чтобы вставить новую *программу* (программный модуль) в документ Mathcad, необходимо выбрать команду *Программирование* (*Programming*) на вкладке *Математика* в группе *Операторы и символы*. В результате откроется список операторов программирования (рис. 4.26).

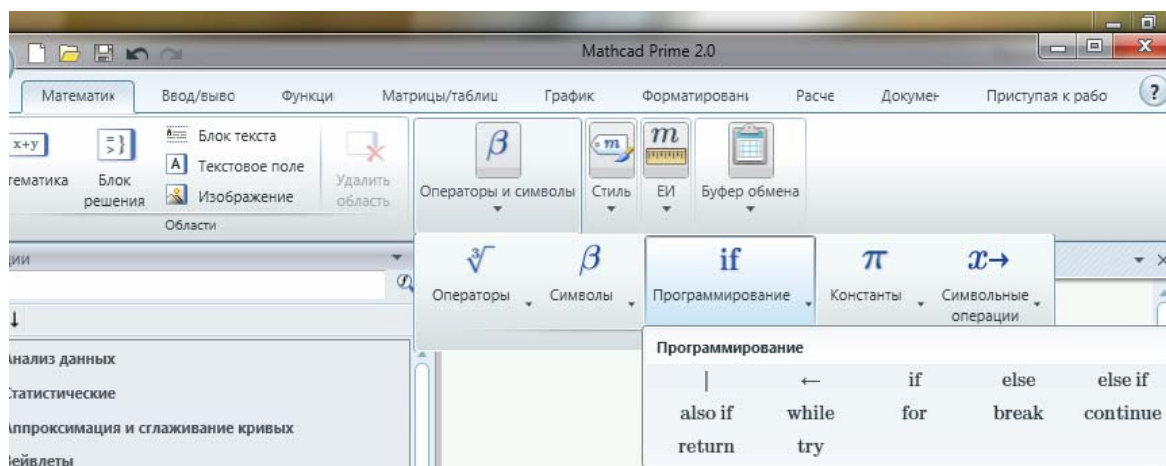


Рис. 4.26

Набор программных элементов в Mathcad ограничен:

символ $|$ – оператор *Программа* (*Program*) создает и при необходимости расширяет жирную вертикальную линию, справа от которой в шаблонах задается запись программного блока;

символ $\leftarrow$ – оператор *Локальное назначение* (*Local Assignment*) присваивает значение в теле программного модуля;

if – оператор условного выражения;

else if – оператор иного выбора, доступен только сразу после оператора *if* или *else if*;

also if – оператор иного выбора, доступен только сразу после оператора *if* или *also if*;

else – оператор иначе;

for – оператор задания цикла с фиксированным числом повторений;

while – оператор задания цикла выполняется, пока выполняется некоторое условие;

break – оператор прерывания;

continue – оператор продолжения;

return – оператор-функция возврата;

try/on error – оператор обработки ошибок.

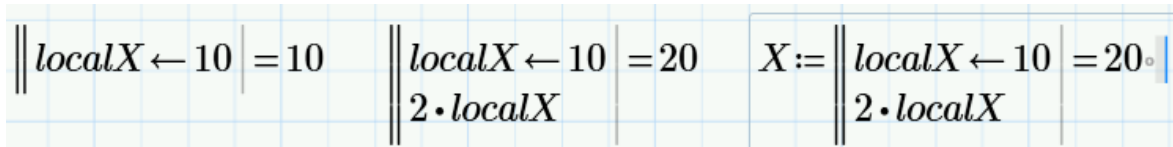
Щелкнуть значок | Программа (*Program*). Откроется структура

программы .

Определить внутри программы локальную переменную, например *X*. Ссылки на нее вне программы невозможны, а если в документе Mathcad уже есть переменная с таким же именем, то они будут считаться двумя разными переменными.

Присвоить ей с помощью оператора ($\leftarrow$) значение, равное 10.

При вводе оператора вычисление ($=$) в конце программного блока Mathcad отображает результат расчета последней строки программы (рис. 4.27).



$\text{local } X \leftarrow 10$	$= 10$	$\text{local } X \leftarrow 10$	$= 20$	$X := \text{local } X \leftarrow 10$	$= 20$
		$2 \cdot \text{local } X$		$2 \cdot \text{local } X$	

Рис. 4.27

Для добавления в программу новой строки поместить курсор в конец выражения присвоения (справа от «10»), нажать клавишу «Ввод» и ввести « $2 \cdot \text{local } X$ ».

После нажатия на клавишу *F5* Mathcad пересчитает программу и отобразит на экране результат вычисления последней строки программного блока, равный 20 (см. рис. 4.27).

В документе Mathcad значение, возвращаемое программой, может быть присвоено переменной, например *X* (см. рис. 4.27).

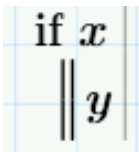
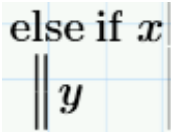
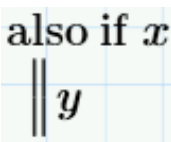
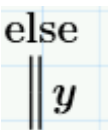
С помощью программных модулей можно определять выражения, переменные и функции в несколько строк, часто с применением специфических программных операторов.

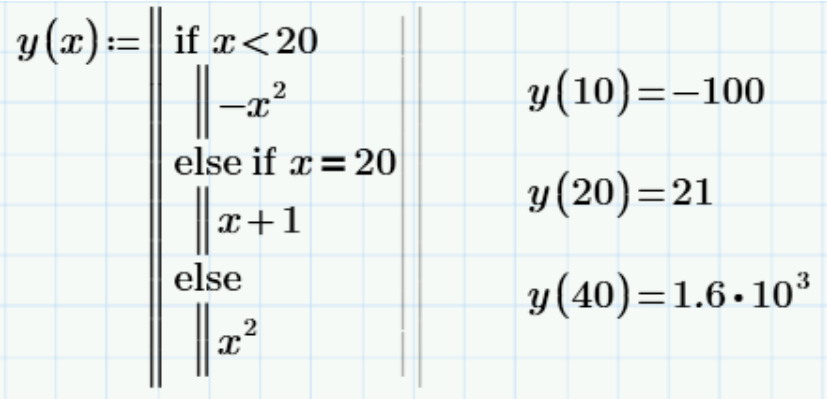
Для создания в программных модулях разветвляющихся алгоритмов целесообразно использовать условные операторы. В табл. 4.1 приведено описание условных операторов, используемых в Mathcad.

На рис. 4.28 представлен программный модуль для вычисления значений ступенчатой функции

$$y(x) = \begin{cases} -x^2 & \text{при } x < 20; \\ x + 1 & \text{при } x = 20; \\ x^2 & \text{при } x > 20. \end{cases}$$

Таблица 4.1

Оператор	Описание оператора
	С оператора <i>if</i> открываются все условные операторы в Mathcad. Вычисляется <i>y</i> при условии, что <i>x</i> не равно нулю (истина) Операнды: <i>x</i> – любое допустимое математическое выражение, результатом которого может быть значение нуль. Учитывается только возвращаемое значение выражения <i>x</i> . Допускается, например, локальное назначение или последовательность шагов программы. <i>Y</i> – любое допустимое математическое выражение
	Может использоваться только внутри оператора <i>if</i> . Вычисляется <i>y</i> при условии, что <i>x</i> не равно нулю (истина) только в том случае, когда предыдущий оператор <i>if</i> или <i>else if</i> возвращает значение нуль (ложь)
	Может использоваться только внутри оператора <i>if</i> . Вычисляется <i>y</i> при условии, что <i>x</i> не равно нулю (истина). Это происходит независимо от того, возвращают предыдущие операторы <i>if</i> или <i>also if</i> – ложь или истина.
	Может использоваться только внутри оператора <i>if</i> . Вычисляется <i>y</i> при условии, что предыдущие операции возвращают значение нуль. Оператором <i>else</i> заканчиваются условные операторы



$$y(x) := \begin{cases} \text{if } x < 20 \\ \quad \parallel \\ \quad -x^2 \\ \text{else if } x = 20 \\ \quad \parallel \\ \quad x + 1 \\ \text{else} \\ \quad \parallel \\ \quad x^2 \end{cases}$$

$$\begin{aligned} y(10) &= -100 \\ y(20) &= 21 \\ y(40) &= 1.6 \cdot 10^3 \end{aligned}$$

Рис. 4.28

Справа от программного модуля, где определяется ступенчатая функция $y(x)$, вычисляются ее значения для трех значений аргумента.

На рис. 4.29 приведен пример определения ступенчатой функции

$$y(x) = \begin{cases} \sin(x) & \text{при } \sin(x) \geq 0; \\ -\sin(x) & \text{при } \sin(x) < 0. \end{cases}$$

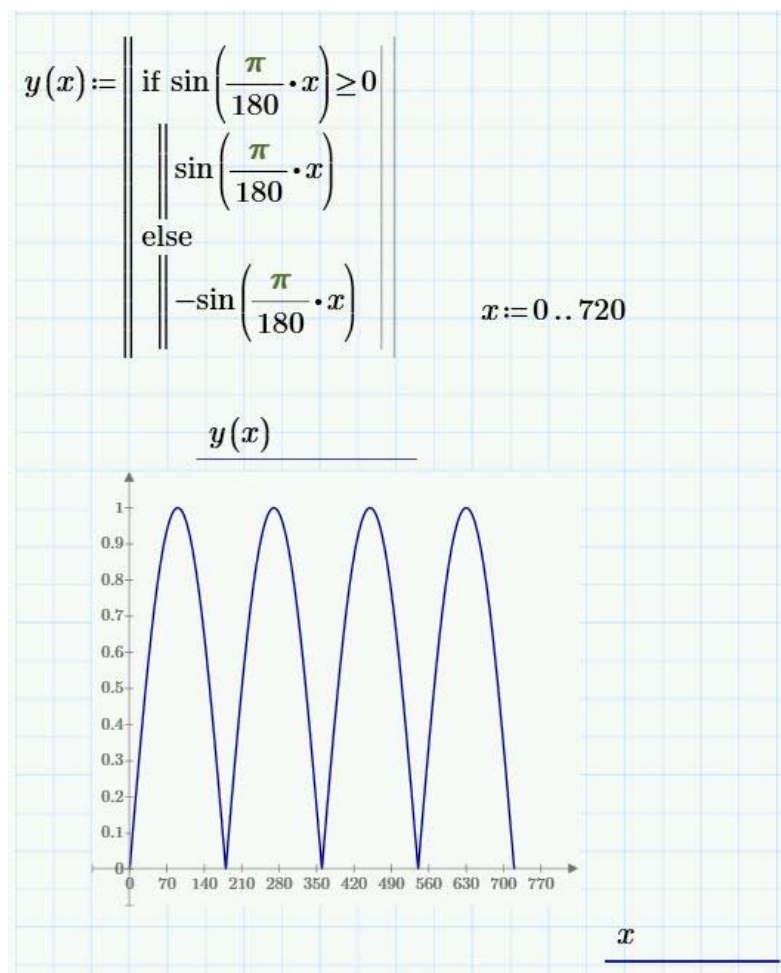


Рис. 4.29

Для полученной функции $y(x)$ построен график в диапазоне от 0 до 720° .

В Mathcad существуют и специальные операторы циклов следующих типов:

- оператор *for* используется в тех случаях, когда заранее известно количество повторений тела цикла;
- оператор *while* используется тогда, когда тело цикла должно выполняться до достижения некоторого условия.

В табл. 4.2 приведено описание циклических операторов, используемых в Mathcad.

На рис. 4.30 приведены два программных модуля, использующие для организации цикла операторы *for* и *while* соответственно, которые возвращают значение суммы чисел от 1 до n .

В первом программном модуле текущее значение переменной цикла изменяется самим оператором цикла *for*, а во втором программном модуле в теле цикла необходимо предусмотреть изменение текущего значения переменной цикла n с шагом один, иначе получится бесконечный цикл.

Таблица 4.2

Оператор	Описание оператора
$\text{for } x \in y \mid \parallel z \mid$	Выполняет тело цикла z для каждого значения переменной цикла x, меняющейся в пределах диапазона y. Операнды: x – любое допустимое имя переменной; y – значение или последовательность значений; z – любое допустимое выражение или последовательность выражений (локальное назначение или последовательность шагов программы)
$\text{while } x \mid \parallel y \mid$	Вычисляется y до тех пор, пока x не равно нулю (истина). Выражение условия вычисляется в начале цикла <i>while</i>. Выполнение цикла прекращается сразу, как только условие перестает выполняться, и возвращается результат, полученный в теле цикла на предыдущем шаге. Операнды: x – любое допустимое выражение или последовательность выражений, результатом вычисления которого может быть 0; y – любое допустимое выражение или последовательность выражений. По крайней мере, одно выражение в теле цикла должно изменять выражение условия x таким образом, что его значением станет «ложь». Иначе цикл будет выполняться бесконечно

$$\begin{array}{lll}
 \text{sum}(n) := \parallel \text{for } n \in 1..n \mid & \text{sum1}(n) := \parallel \text{while } n > 0 \mid & \text{sum}(4) = 10 \\
 \parallel S \leftarrow S + n \mid & \parallel S \leftarrow S + n \mid & \\
 \parallel S \mid & \parallel n \leftarrow n - 1 \mid & \text{sum1}(4) = 10 \\
 & \parallel S \mid &
 \end{array}$$

Рис. 4.30

На рис. 4.31 показаны примеры программных модулей, использующие для организации циклов операторы *for* и *while*, которые создают двумерный массив $M1$ размерностью 2×3 . Значения элементов массива $M1$ вычисляются по формуле

$$M1_{i,j} = i + j.$$

В примере на рис. 4.14 рассматривалось применение ранжированных переменных для создания такого же массива $M1$. Чаще всего намного удобнее организовать циклы (в том числе вложенные) с помощью ранжированных переменных, чем заниматься для этого программированием.

$$\begin{array}{l}
 \text{ORIGIN} := 1 \\
 M1 := \left[\begin{array}{c} \text{for } i \in 1..2 \\ \quad \left[\begin{array}{c} \text{for } j \in 1..3 \\ \quad M1_{i,j} \leftarrow i+j \end{array} \right] \\ M1 \end{array} \right] = \begin{bmatrix} 2 & 3 & 4 \\ 3 & 4 & 5 \end{bmatrix}
 \end{array}$$

$$\begin{array}{l}
 M1 := \left[\begin{array}{c} i \leftarrow 1 \\ \text{while } i \leq 2 \\ \quad \left[\begin{array}{c} j \leftarrow 1 \\ \text{while } j \leq 3 \\ \quad M1_{i,j} \leftarrow i+j \\ \quad j \leftarrow j+1 \\ \quad i \leftarrow i+1 \end{array} \right] \\ M1 \end{array} \right] = \begin{bmatrix} 2 & 3 & 4 \\ 3 & 4 & 5 \end{bmatrix}
 \end{array}$$

Рис. 4.31



Изучите самостоятельно другие программные модули Mathcad и возможности Mathcad по применению *рекурсии* при вычислении функции.

4.5. Символьные вычисления в системе Mathcad

Помимо выполнения числовых расчетов в Mathcad, можно также выполнять операции с *символьными выражениями*, содержащими переменные или математические символы.

При этом результат символьных операций Mathcad возвращает также в символьной форме, что позволяет легче выявить взаимосвязи переменных, которые не всегда очевидны из численных результатов. Кроме того, в отличие от числовых вычислений, символьные операции позволяют вычислять выражения без присвоения значений переменным, входящим в эти выражения, а возвращают результаты через символы и переменные исходного выражения.

Выражения в символьных расчетах вычисляются с использованием оператора *аналитического преобразования* ($\rightarrow$) вместо оператора численного вычисления ($=$). Все стандартные операторы Mathcad и многие встроенные функции можно вычислять аналитически. На рис. 4.32 приведены некоторые примеры.

При вычислении выражения с помощью оператора аналитического преобразования Mathcad упрощает результаты, выполняя арифметические действия и приводя подобные члены.

$$5x - 3(13y - 2x) - 3y \rightarrow 11 \cdot x - 42 \cdot y$$

$$\frac{d}{dt} \sin(\omega \cdot t) \rightarrow \omega \cdot \cos(t \cdot \omega)$$

$$\lim_{K \rightarrow \infty} \left(\frac{K}{K - \beta \cdot K} \right) \rightarrow -\frac{1}{\beta - 1}$$

Рис. 4.32

При выполнении более сложных символьных операций можно использовать *ключевые слова* и *модификаторы*, указывающие требуемую операцию, полный список которых и их описание можно посмотреть в справке Mathcad.

Рассмотрим для примера, ключевое слово *simplify*, которое упрощает с помощью тождеств выражения для алгебраических, тригонометрических и экспоненциальных функций и сокращает общие множители (рис. 4.33).

$$\sin(x)^2 + \cos(x)^2 \xrightarrow{\text{simplify}} 1$$

$$\frac{1}{x} + \sin(2x)^2 \xrightarrow{\text{simplify}} \frac{x \cdot \sin(2 \cdot x)^2 + 1}{x}$$

$$\frac{1}{x} + \sin(2x)^2 \xrightarrow{\text{simplify, max}} \frac{1}{x} + \sin(2 \cdot x)^2$$

Рис. 4.33

Для вставки ключевых слов и модификаторов необходимо сначала выбрать выражение, которое требуется вычислить аналитически, выбрать *Символьные операции* (Symbolics) на вкладке *Математика* в группе *Операторы и символы*, а затем щелкнуть оператор.

В результате справа от выражения появится оператором аналитического преобразования. При необходимости ввести ключевое слово или модификатор над стрелкой требуется щелкнуть пустое поле над ней и ввести его. Ключевые слова отображаются над оператором аналитического преобразования.

Можно сразу же выбрать нужное ключевое слово или модификатор на вкладке *Математика* в группе *Операторы и символы* в окне *Символьные операции*, а затем ввести выражение, которое требуется вычислить.

Модификаторами называются специальные элементы, которые используются для изменения поведения ключевых слов. Например, в третьем примере на рис. 4.33 для увеличения числа шагов, выполняемых алгоритмом упрощения, добавлен модификатор *max*. Однако данное выражение Mathcad все равно не смог упростить.

Символьные расчеты свободны от ошибок округления, при-
сущих численным расчетам.

Например, на рис. 4.34 показан результат сложения дробей с использованием оператора вычисления, в котором количество отображаемых цифр зависит от текущей настройки параметра *Точность отображения*.

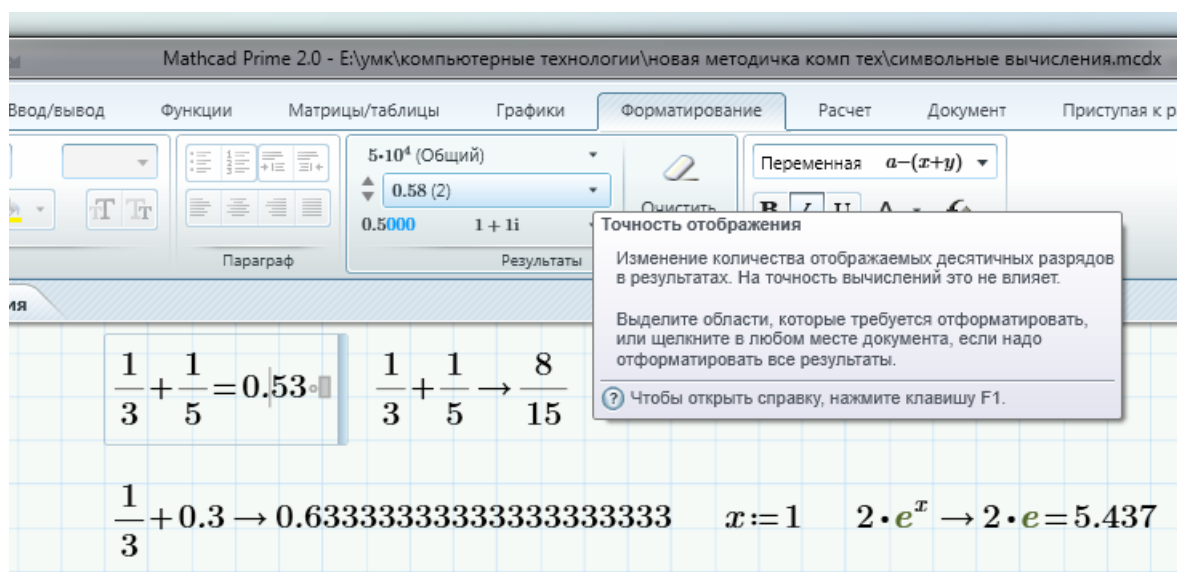


Рис. 4.34

При символьном вычислении того же самого арифметического выражения результат получается без округления.

Однако если выражение содержит десятичное число, Mathcad возвращает результат в виде десятичного числа (см. рис. 4.34).

Большинство символьных результатов можно рассчитать численно, вставив численный знак равенства после символьного результата (рис. 4.35).

При вычислении функции с использованием оператора аналитического преобразования Mathcad выполняет подстановку аргумента в функцию и вычисляет ее аналитически (рис. 4.36).

Чтобы указать Mathcad, что необходимо временно игнорировать заданное ранее в программе значение переменной, используйте ключевое слово *explicit*, например, для решения уравнения в символьной форме с помощью операции *solve* или для разложения выра-

жения на множители с помощью операции *factor* в том случае, если ранее в программе *x* уже было присвоено значение (рис. 4.36).

Задание функции	Вычисление функции	
	аналитически	численно
$f(R,X) := \sqrt{R^2 + X^2}$	$f(10,10) \rightarrow 10 \cdot \sqrt{2}$	$f(10,10) = 14.142$

Рис. 4.35

$$\begin{aligned}
 & x^2 - 27 \xrightarrow{\text{explicit, solve}} \begin{bmatrix} 3 \cdot \sqrt{3} \\ -3 \cdot \sqrt{3} \end{bmatrix} \\
 & x^2 - \frac{27}{x} \xrightarrow{\text{explicit, factor}} \frac{(x-3) \cdot (x^2 + 3 \cdot x + 9)}{x} \\
 & \frac{(x-3) \cdot (x^2 + 3 \cdot x + 9)}{x} \xrightarrow{\text{expand}} x^2 - \frac{27}{x}
 \end{aligned}$$

Рис. 4.36

Для разложения на сомножители символьных выражений в Mathcad используются следующие ключевые слова:

factor – выполняет разложение целых чисел, полиномов и рациональных функций в произведение более простых элементов;

expand – разлагает выражения путем перемножения элементов произведения и возведения их в степень и применения тождеств для общих функций.

На рис. 4.36 операция *expand* преобразует результат разложения выражения с помощью операции *factor* в исходное выражение.

На рис. 4.37 приведены примеры работы операции *collect*, которая собирает элементы в полиномы, содержащие одинаковые степени указанных переменных *x* или *y*.

$$\begin{aligned}
 & 5x^2 + 10x - 10xy + 7y^2 \xrightarrow{\text{collect, } x} 5 \cdot x^2 + (10 - 10 \cdot y) \cdot x + 7 \cdot y^2 \\
 & 5x^2 + 10x - 10xy + 7y^2 \xrightarrow{\text{collect, } y} 7 \cdot y^2 + -10 \cdot x \cdot y + 5 \cdot x^2 + 10 \cdot x
 \end{aligned}$$

Рис. 4.37

Глава 5. Практикум по применению среды VBA в Microsoft Office

5.1. Создание макросов в Microsoft Office

1) Создать в среде Word макрос под именем *Макрос1*, выполняющий в зависимости от варианта следующие действия:

– вводит текст «Студент *Фамилия* учится писать макрос» (указать свою фамилию);

– устанавливает с помощью кнопок на панели инструментов параметры шрифта и абзаца, указанные в табл. 5.1.

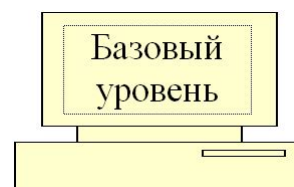


Таблица 5.1

Номер варианта	Параметры шрифта			Параметры абзаца	
	Начертание	Цвет	Размер	Выравнивание	Отступ, см
1	Обычный	Черный	11	По левому краю	0,5
2	Полужирный	Синий	12	По ширине	0,6
3	Курсив	Красный	13	По правому краю	0,7
4	Обычный	Синий	14	По центру	0,8
5	Полужирный	Красный	15	По левому краю	0,9
6	Курсив	Черный	16	По ширине	1
7	Обычный	Красный	17	По правому краю	1,1
8	Полужирный	Черный	18	По центру	1,2
9	Курсив	Синий	19	По левому краю	1,3
10	Обычный	Черный	20	По ширине	1,4
11	Полужирный	Синий	11	По правому краю	0,5
12	Курсив	Красный	12	По центру	0,6
13	Обычный	Синий	13	По левому краю	0,7
14	Полужирный	Красный	14	По ширине	0,8
15	Курсив	Черный	15	По правому краю	0,9
16	Обычный	Красный	16	По центру	1
17	Полужирный	Черный	17	По левому краю	1,1
18	Курсив	Синий	18	По ширине	1,2
19	Обычный	Черный	19	По правому краю	1,3
20	Полужирный	Синий	20	По центру	1,4

2) Сохранить на диске созданный макрос в текущем файле Microsoft Word под своей фамилией.

3) Сохранить на диске созданный макрос в отдельном файле под своей фамилией с расширением *.bas*.

4) Создать второй макрос в текущем документе Word под именем *Макрос2*, который выполняет те же самые действия, что и *Макрос1*, но форматирование шрифта и абзаца выполняется с помощью диалоговых окон *Шрифт* и *Абзац*.

5) Создать в среде Microsoft Excel макрос под именем *Макрос3*, выполняющий следующие действия:

- вводит в ячейку *A1* электронной таблицы Вашу фамилию;
- устанавливает в зависимости от варианта следующие параметры для активной ячейки, указанные в табл. 5.2.

Таблица 5.2

Номер варианта	Граница	Заливка	Шрифт		Выравнивание по горизонтали
			Размер	Начертание	
1	Слева	Голубой	11	Обычный	По левому краю
2	Справа	Желтый	12	Полужирный	По ширине
3	Сверху	Голубой	13	Курсив	По правому краю
4	Снизу	Желтый	14	Обычный	По центру
5	Слева	Голубой	15	Полужирный	По левому краю
6	Справа	Желтый	16	Курсив	По ширине
7	Сверху	Голубой	17	Обычный	По правому краю
8	Снизу	Желтый	18	Полужирный	По центру
9	Слева	Голубой	19	Курсив	По левому краю
10	Справа	Желтый	20	Обычный	По ширине
11	Сверху	Голубой	11	Полужирный	По правому краю
12	Снизу	Желтый	12	Курсив	По центру
13	Слева	Голубой	13	Обычный	По левому краю
14	Справа	Желтый	14	Полужирный	По ширине
15	Сверху	Голубой	15	Курсив	По правому краю
16	Снизу	Желтый	16	Обычный	По центру
17	Слева	Голубой	17	Полужирный	По левому краю
18	Справа	Желтый	18	Курсив	По ширине
19	Сверху	Голубой	19	Обычный	По правому краю
20	Снизу	Желтый	20	Полужирный	По центру

6) Сохранить на диске созданный макрос в текущей книге Microsoft Excel под своей фамилией.

7) Составить отчет, включив в него титульный лист, краткое описание среды VBA и текст записанных макросов с подробными комментариями, а также результаты работы макросов.

Контрольные вопросы

1. Что включает в себя понятие объекта?
2. Каким образом осуществляется доступ к объектам в VBA?
3. Для чего предназначены свойства объектов и каков синтаксис команды изменения свойства?
4. Для чего предназначены методы объектов и каков синтаксис команды изменения метода?
5. Как создать, отредактировать, запустить на выполнение и сохранить на диске макрос?
6. Какие способы активизации редактора VBA существуют?
7. Назвать основные компоненты редактора VBA.
8. Каковы назначение и возможности окна проекта?
9. Назовите способы открытия и основные возможности окна редактирования кода.
10. Какие команды по редактированию программного кода располагаются на панели инструментов Edit?
11. Каковы назначение и возможности окна свойств?
12. Назовите способы открытия окна свойств для просмотра свойств объекта.
13. Каково назначение инструкции Option Explicit?
14. Каким образом организован проект в VBA?
15. Какие существуют способы вызова процедуры в VBA?

5.2. Разработка программ в среде VBA для Word

1. Записать процедуру для вычисления значения ступенчатой функции Z для заданного варианта из табл. 5.3. Предусмотреть запуск программы из среды Word и возможность ввода значения переменной x из диалогового окна *InputBox*, а также вывод значения Z в панель сообщения *MsgBox*.



Таблица 5.3

Номер варианта	Функция
1	$Z = \begin{cases} 10,5x^2 + 1, & x \leq 0; \\ 5(x+1)^3, & 0 < x < 1; \\ \sqrt{x^2 + 1}, & x \geq 1 \end{cases}$
2	$Z = \begin{cases} \cos(x-1), & -2\pi < x \leq 0; \\ \cos x + 1, & 0 < x < \pi; \\ \sin(x+0,5), & \pi \leq x \leq 2\pi \end{cases}$
3	$Z = \begin{cases} x^4 - 12, & x \leq -1; \\ (2x+1)^3, & -1 < x < 1; \\ \sqrt{\frac{x+11}{x}}, & x \geq 1 \end{cases}$
4	$Z = \begin{cases} x^2 + 5x, & x < 1; \\ (x^2 + 3x)^2, & x = 1; \\ (x - 2x^2), & x > 1 \end{cases}$
5	$Z = \begin{cases} 0,5x^3 + 1, & x \leq -10; \\ (x+1)^2, & -10 < x < 10; \\ \sqrt{x+10}, & x \geq 10 \end{cases}$
6	$Z = \begin{cases} x^4 - 12, & x \leq -1; \\ (2x+1)^3, & -1 < x < 1; \\ \sqrt{\frac{x+11}{x}}, & x \geq 1 \end{cases}$
7	$Z = \begin{cases} 15x + 1, & x \leq 0; \\ 5(x+1)^2, & 0 < x < 1; \\ \sqrt{x^3 - 1}, & x \geq 1 \end{cases}$
8	$Z = \begin{cases} x^2 + 1, & x < 1; \\ (x+2)^2, & x = 1; \\ (x^3 + 2x), & x > 1 \end{cases}$

Номер варианта	Функция
9	$Z = \begin{cases} \sin 3x, & -\pi < x \leq 0; \\ \cos(x+1), & 0 < x < \pi/2; \\ \sin 2x, & \pi/2 \leq x \leq \pi \end{cases}$
10	$Z = \begin{cases} 0,5x^3 + 1, & x \leq -10; \\ (x+1)^2, & -10 < x < 10; \\ \sqrt{x+10}, & x \geq 10 \end{cases}$
11	$Z = \begin{cases} \frac{1}{x+1} & \text{при } x < 0; \\ -x^2 & \text{при } x \geq 0 \end{cases}$
12	$Z = \begin{cases} \frac{1}{x^2} & \text{при } x < -1; \\ x^2 & \text{при } -1 \leq x \leq 2; \\ 4 & \text{при } x > 2 \end{cases}$
13	$Z = \begin{cases} x^2 & \text{при } x < 0; \\ -x^4 & \text{при } x \geq 0 \end{cases}$
14	$Z = \begin{cases} -x-1 & \text{при } x < -1; \\ x+1 & \text{при } -1 \leq x \leq -0; \\ -x+1 & \text{при } -0 \leq x \leq 1; \\ x-1 & \text{при } x > 1 \end{cases}$
15	$Z = \begin{cases} -3x-1 & \text{при } x \leq -2; \\ -x+6 & \text{при } -2 \leq x \leq 1; \\ x+4 & \text{при } 1 \leq x \leq 3; \\ 3x-2 & \text{при } x > 3 \end{cases}$
16	$Z = \begin{cases} -1 & \text{при } x < -1; \\ x & \text{при } -1 \leq x \leq 1; \\ 1 & \text{при } x > 1 \end{cases}$

Номер варианта	Функция
17	$Z = \begin{cases} 1 - x^2 & \text{при } x < 0; \\ 2 - x^3 & \text{при } x \geq 0 \end{cases}$
18	$Z = \begin{cases} -x & \text{при } x < -1; \\ 1 & \text{при } -1 \leq x \leq 1; \\ x & \text{при } x > 1 \end{cases}$
19	$Z = \begin{cases} -\frac{1}{x^2} + 1 & \text{при } x < -1; \\ x^2 + 1 & \text{при } -1 \leq x \leq 2; \\ 5 & \text{при } x > 2 \end{cases}$
20	$Z = \begin{cases} x + 1 & \text{при } x < -1; \\ -x - 1 & \text{при } -1 \leq x \leq -0; \\ x - 1 & \text{при } -0 \leq x \leq 1; \\ -x + 1 & \text{при } x > 1 \end{cases}$

2. Записать процедуру, запускаемую из Word, для вычисления и вывода на панель сообщений *MsgBox* значений функции для заданного варианта из табл. 5.4.

Значения x меняются от -20 до 20 с шагом 2 .

Для организации циклов в одной процедуре использовать оператор *Fof...Next*, а в другой процедуре – оператор *While...Wend*.

Вывести полученные результаты с точностью один знак после запятой на панель сообщений в одну строку для первой процедуры и в отдельные панели сообщений для второй процедуры.

Таблица 5.4

Номер варианта	Функция	Номер варианта	Функция
1	$A = \sum_{n=1}^5 \sqrt[4]{1000n + x^4}$	3	$A = \prod_{k=1}^4 \sin(x + 3k)$
2	$A = \prod_{k=1}^3 (x^2 - k)$	4	$A = \sum_{n=1}^5 \sqrt{n^2 + 2x}$

Номер варианта	Функция	Номер варианта	Функция
5	$A = \prod_{k=1}^6 \frac{\cos x}{2k}$	13	$A = \prod_{k=1}^3 \sqrt{(xk + 2)^3}$
6	$A = \sum_{k=1}^4 (x^4 + 4k)$	14	$A = \sum_{n=1}^6 \frac{x^n}{n}$
7	$A = \prod_{k=1}^5 e^x + kx$	15	$A = \sum_{k=0}^5 \operatorname{tg}(x + k)$
8	$A = \sum_{i=1}^{100} \frac{1-i}{i^2}$	16	$A = \sum_{i=1}^{120} \frac{1+2i}{i+1}$
9	$A = \sum_{i=1}^{50} \frac{1+i}{i^3}$	17	$A = \prod_{k=1}^3 \sqrt{(xk + 1)^2}$
10	$A = \prod_{k=1}^6 \frac{\sin x}{2k + 1}$	18	$A = \sum_{i=1}^{12} \frac{(-1)^{i+1}}{i(i+1)}$
11	$A = \sum_{i=1}^{39} \frac{1}{(2i+1)^2}$	19	$A = \sum_{i=1}^{25} \frac{(-1)^i}{i(2i+1)}$
12	$A = \sum_{i=1}^{90} \frac{(-1)^{i+1}}{(i+1)(i+2)}$	20	$A = \sum_{i=1}^5 \frac{(-1)^{i+1}}{4^i + 5^{i+2}}$

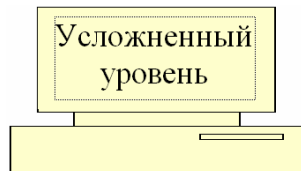
3. Составить процедуру, запускаемую из Word, для вычисления и вывода на панель сообщений *MsgBox* наибольшего и наименьшего значений функции для заданного варианта из табл. 5.5 и значения аргумента, при котором оно получено.

Таблица 5.5

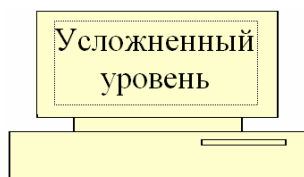
Номер варианта	Функция	Диапазон изменения x	Шаг изменения x
1	$Y = -\sin(x/2)$	0–360°	10°
2	$Y = 13x^2$	0–24	4
3	$Y = \cos(4x - 60^\circ)$	0–180°	5°
4	$Y = \frac{1}{x+4}$	–1–3	0,5
5	$Y = \frac{x+44}{x-23}$	–8–8	2
6	$Y = (3x^2 + 43)$	0–40	4

Окончание табл. 5.5.

Номер варианта	Функция	Диапазон изменения x	Шаг изменения x
7	$Y = \operatorname{tg}(x) - 1$	0–90°	2°
8	$Y = \frac{x}{x - 31}$	–10–30	5
9	$Y = (x^3 - x)$	0–200	20
10	$Y = \sin(4x - 45^\circ)$	0–180°	2°
11	$Y = \sin(3x + 60^\circ)$	0–360°	4°
12	$Y = 1 - e^{5x}$	0–5	0,5
13	$Y = \sin(5x)$	0–360°	8°
14	$Y = \operatorname{tg}(x + 30^\circ)$	0–90°	2°
15	$Y = \cos(5x - 15^\circ)$	0–180°	4°
16	$Y = \frac{2x}{x + 100}$	–80–80	10
17	$Y = 1 - e^{2x}$	0–5	0,1
18	$Y = 1 - x^2$	0–1	0,01
19	$Y = 1 + e^{4x}$	0–5	0,1
20	$Y = \cos(4x)$	0–360°	12°



4. Выполнить предыдущее задание, используя для организации цикла оператор *Do...Loop*. Вывести полученные результаты с точностью два знака после запятой в форму.



5. Записать процедуру, запускаемую из Word, по обработке заданной матрицы в соответствии с вариантом задания (табл. 5.6). Предусмотреть автоматическое заполнение матрицы с помощью генератора случайных чисел и ее вывод в форму, а также вывод результатов работы процедуры на панель сообщений *MsgBox*.

Таблица 5.6

Номер варианта	Задание
1	Дана действительная матрица порядка 6×3. Найти сумму наибольших значений ее строк
2	Для отрицательных элементов главной диагонали действительной квадратной матрицы порядка 4×4 вывести суммы элементов столбцов, где эти элементы расположены

Номер варианта	Задание
3	Определить, является ли квадратная матрица порядка 6×6 симметричной относительно побочной диагонали
4	Из данной действительной матрицы порядка 5×6 сформировать матрицу столбец, значение элементов строк которой равны среднеквадратическому значению элементов соответствующих строк исходной матрицы
5	Дана действительная матрица порядка 3×6 . Найти сумму элементов строки, в которой расположен единственный элемент с наименьшим значением
6	В данной действительной квадратной матрице порядка 5×5 найти строки, элементы которых упорядочены по убыванию
7	В данной действительной квадратной матрице порядка 4×4 найти сумму положительных элементов над главной диагональю
8	В данной действительной квадратной матрице порядка 3×3 поменять местами строку, в которой расположен единственный элемент с наибольшим значением, со строкой, содержащей единственный элемент с наименьшим значением
9	В данной действительной квадратной матрице порядка 4×4 указать индексы всех элементов с наибольшим значением
10	Для положительных элементов главной диагонали действительной квадратной матрицы порядка 4×4 вывести значения максимальных элементов строк, где эти элементы расположены
11	Дана действительная матрица порядка 5×3 . Найти сумму наименьших значений ее столбцов
12	В данной действительной квадратной матрице порядка 4×4 найти столбцы, элементы которых упорядочены по убыванию
13	В данной действительной квадратной матрице порядка 5×5 указать индексы всех элементов с наименьшим значением
14	Для действительной матрицы порядка 4×5 найти сумму наибольших значений столбцов
15	Для отрицательных элементов главной диагонали действительной квадратной матрицы порядка 6×6 вывести значения максимальных элементов столбцов, где эти элементы расположены
16	В данной действительной квадратной матрице порядка 3×3 найти строки, элементы которых упорядочены по возрастанию

Номер варианта	Задание
17	В данной действительной квадратной матрице порядка 6×6 найти сумму положительных элементов над побочной диагональю
18	Дана действительная матрица порядка 4×4 . Найти сумму элементов столбца, в котором расположен единственный элемент с наименьшим значением
19	В данной действительной квадратной матрице порядка 5×5 найти сумму положительных элементов на главной диагонали и сумму элементов по столбцам
20	Для положительных элементов главной диагонали действительной квадратной матрицы порядка 5×5 вывести суммы элементов строк, где эти элементы расположены
21	В данной действительной квадратной матрице порядка 5×5 найти сумму отрицательных элементов на побочной диагонали и сумму элементов по строкам
22	Для положительных элементов главной диагонали действительной квадратной матрицы порядка 4×4 вывести максимумы элементов столбцов, где эти элементы расположены
23	В данной действительной квадратной матрице порядка 4×4 найти произведение элементов на главной диагонали и сумму элементов на побочной диагонали
24	Для отрицательных элементов главной диагонали действительной квадратной матрицы порядка 5×5 вывести минимумы элементов строк, где эти элементы расположены
25	В данной действительной квадратной матрице порядка 6×6 найти сумму отрицательных элементов на главной диагонали
26	Найти сумму наименьших значений строк действительной матрицы порядка 5×5
27	В данной действительной квадратной матрице порядка 5×5 найти сумму элементов на главной диагонали и произведение элементов на побочной диагонали
28	Для отрицательных элементов главной диагонали действительной квадратной матрицы порядка 6×6 вывести максимумы элементов строк, где эти элементы расположены
29	Дана действительная матрица порядка 5×3 . Найти сумму элементов столбца, в котором расположен единственный элемент с наибольшим значением

Номер варианта	Задание
30	В данной действительной квадратной матрице порядка 3×3 найти столбцы, элементы которых упорядочены по возрастанию
31	Для отрицательных элементов главной диагонали действительной квадратной матрицы порядка 4×4 вывести суммы элементов строк, где эти элементы расположены
32	В данной действительной квадратной матрице порядка 4×4 поменять местами столбец, в котором расположен единственный элемент с наибольшим значением, со столбцом, содержащим единственный элемент с наименьшим значением
33	Дана действительная матрица порядка 4×5 . Найти сумму элементов строки, в которой расположен единственный элемент с наибольшим значением
34	Для положительных элементов главной диагонали действительной квадратной матрицы порядка 6×6 вывести минимальные значения элементов строк, где эти элементы расположены
35	Транспонировать целочисленную квадратную матрицу порядка 6×6
36	Для положительных элементов главной диагонали действительной квадратной матрицы порядка 5×5 вывести суммы элементов столбцов, где эти элементы расположены
37	В данной действительной квадратной матрице порядка 4×4 найти сумму положительных элементов на побочной диагонали
38	Для отрицательных элементов главной диагонали действительной квадратной матрицы порядка 5×5 вывести минимумы элементов столбцов, где эти элементы расположены
39	Определить, является ли квадратная матрица порядка 5×5 симметричной относительно главной диагонали
40	В данной действительной квадратной матрице порядка 5×5 найти произведение отрицательных элементов под главной диагональю

6. Сохранить разработанные процедуры в формате документов Word с поддержкой макросов, указав в качестве имени файла свою фамилию плюс номер задания.

7. Составить отчет, включив в него титульный лист, текст разработанных процедур с подробными комментариями и результаты их работы.

Контрольные вопросы

1. Назовите основные правила записи имен переменных и констант.
2. Каковы виды переменных и констант?
3. Назовите способы объявления (описания) переменных и констант.
4. Назовите типы данных, используемых при описании переменных и констант.
5. Какие группы функции используются в VBA?
6. Каковы назначение и формат функции *MsgBox* и *InputBox*?
7. В чем отличие между строчным и блочным вариантом оператора ветвления?
8. Можно ли вставлять инструкцию *Else* перед инструкцией *ElseIf* в блочном варианте оператора ветвления?
9. Что такое цикл и для чего он нужен?
10. Какие операторы могут использоваться для организации циклов с заданным числом повторений?
11. В чем основное отличие между циклами с проверкой условия в начале цикла и в конце цикла?
12. В каких случаях целесообразно использовать циклы с предусловием, циклы с постусловием и циклы по счетчику?
13. Какие операторы могут использоваться для циклов с неизвестным числом повторений?
14. Что такое вложенный цикл и каковы основные правила его конструирования?
15. Каково назначение функций *CStr*, *Round*, *Len*?
16. Каково назначение системных переменных *vbLf*, *vbTab*?
17. Назовите наиболее часто используемые свойства форм.
18. Назовите общие методы и инструкции форм.
19. Назовите общие для форм и элементов управления события.
20. Приведите список основных элементов управления.
21. Каким образом можно обратиться к любому элементу управления в VBA?
22. Каким образом можно перенести текст из одного текстового окна в другое?
23. Для чего используется функция *Val*?
24. Какие виды ошибок отслеживаются IDE VBA?
25. Как запускается режим отладки в IDE VBA?

26. Каким образом формируется список в элементе *ComboBox*?

27. Каким образом можно вставить рисунок в форму?

28. Для чего предназначены закладки и как в Microsoft Word организовано обращение к ним?

29. Каким образом информация из текстовых полей, расположенных в форме, переносится в место закладки?

5.3. Разработка программ в среде VBA для Excel

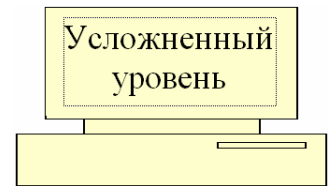
1. Составить процедуру, запускаемую из Excel, для вычисления и вывода на экран монитора наибольшего и наименьшего значений функции для заданного варианта из табл. 5.5 и значения аргумента, при котором оно получено.

Предусмотреть возможность ввода значения переменной x и значения функции $Y(x)$ в соответствующие ячейки таблицы. Вставить в текущую книгу Excel график функции $y(x)$ в заданном диапазоне изменения x .



2. Записать процедуру по обработке заданной матрицы в соответствии с вариантом задания (см. табл. 5.6) в среде VBA для Excel.

Предусмотреть автоматическое заполнение матрицы с помощью генератора случайных чисел и вывод исходной матрицы и результатов работы процедуры в соответствующие ячейки таблицы.



3. Сохранить разработанные процедуры в формате таблицы Excel с поддержкой макросов, указав в качестве имени файла свою фамилию плюс номер задания.

4. Составить отчет, включив в него титульный лист, текст разработанных процедур с подробными комментариями и результаты их работы.

Контрольные вопросы

1. На что влияет подключение режима конструктора?

2. Каким образом добавляются элементы управления в таблицу Excel?

3. Как в VBA программно перенести содержимое одной ячейки Excel в другую ячейку?

4. Каким образом в VBA задается диапазон ячеек?

5. Каким образом в VBA формируется заголовок графика?

Ключевое слово *parfrac* выполняет разложение рационального выражения в сумму дробей с линейными или квадратичными знаменателями. Если выражение содержит несколько переменных, то для его разложения на рациональные дроби введите запятую после ключевого слова *parfrac* и укажите имя одной из переменных (рис. 4.38).

$$\frac{3-a}{x^2-8x+15} \xrightarrow{\text{parfrac}, x} \frac{a-3}{2 \cdot (x-3)} - \frac{a-3}{2 \cdot (x-5)}$$

Рис. 4.38



Изучите самостоятельно по Справке Mathcad описание операций *coeffs* и *rewrite* и соответствующих им модификаторов.

Глава 6. Практикум по применению программы CorelDraw для создания иллюстраций

6.1. Создание простых векторных объектов в CorelDRAW

Для создания простых векторных объектов необходимо:



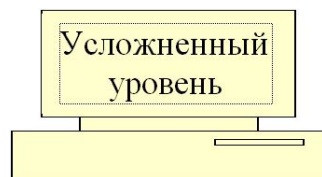
1. Изучить назначение кнопок на *Панели масштабов*. Попробовать изменить масштаб изображения различными способами и включить в отчет по лабораторной работе их описание.

2. Изучить команды из раздела меню *Макет*. Создать рисунок и установить для него следующие *Установки страницы*: полная страница с форматом *A4*, ориентация *Книжная*.

3. Нарисовать, используя рассмотренные векторные объекты CorelDRAW, заданную электрическую схему в соответствии с вариантом задания из приложения Б.

4. Нарисовать одну из стандартных фигур CorelDRAW в соответствии с вариантом задания из приложения В. Вписать внутри фигуры свою фамилию. Выполнить все возможные варианты редактирования заданной фигуры.

5. Составить отчет в электронном виде и на бумажном носителе, включив в него титульный лист, все нарисованные в соответствии с заданиями пп. 2–5 графические объекты и результаты выполнения двух индивидуальных заданий.



6. Изучить самостоятельно дополнительные возможности команды *Художественное оформление*. Включить в отчет по работе описание дополнительных возможностей команды *Художественное оформление* с соответствующими графическими примерами.

Контрольные вопросы

1. Какие области можно выделить в рабочем окне редактора CorelDRAW X6?

2. Каким образом можно выделить сразу несколько объектов в CorelDRAW?

3. Как в CorelDRAW могут быть выделены объекты, не полностью попавшие в рамку выделения, и в каком случае это бывает полезно?

4. Как существуют способы по выбору масштаба в CorelDRAW?

5. Как изменить единицы измерения и параметры сетки в CorelDRAW?

6. Как изменится внешний вид прямоугольника, эллипса, звезды, спирали и решетки, если при их создании в CorelDRAW использовать клавиши Ctrl и Shift?

7. Какие возможности имеются в CorelDRAW для рисования линий и кривых?

8. Какие типы текста можно вводить в CorelDRAW?

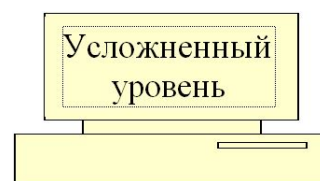
9. Каким образом выполняется форматирование текстов в CorelDRAW?

6.2. Редактирование объектов, контура и заливки в CorelDRAW

1. С помощью команды **Import (Импорт)** поместить в документ CorelDRAW картинку из файла, заданного преподавателем в соответствии с вариантом задания из приложения Г. С помощью инструмента *Безье* получить упрощенную копию оригинала в цвете и сохранить ее в формате .cdr. Вставить в отчет оригинал и копию заданной картинки.



2. Включить в отчет по лабораторной работе описание характеристик из окна *Перо абриса* с соответствующими графическими примерами.



3. Включить в отчет по лабораторной работе описание различных видов заливки с соответствующими графическими примерами.

4. Вставить в документ CorelDRAW свою фамилию с помощью объекта фигурного текста и, используя инструменты *Оболочка* и *Вытянуть*, изменить ее. Включить в отчет по лабораторной работе несколько вариантов результатов применения этих инструментов.

5. Самостоятельно изучить дополнительные возможности по настройке параметров инструмента *Тень*, в том числе и применительно к тексту, содержащему Вашу фамилию. Включить в отчет по работе их описание с соответствующими графическими примерами.

6. Самостоятельно изучить, какие настройки могут меняться для слоев в окне *Менеджер объекта*. Включить в отчет по работе их описание.

Контрольные вопросы

1. Какие опции имеются у команды *Import (Импорт)* в CorelDRAW?

2. Каким образом в CorelDRAW рисуются и редактируются объекты, состоящие из кривых Безье?

3. Каким образом можно менять размеры объектов в CorelDRAW с сохранением и без сохранения пропорций?

4. Как влияют клавиши *Ctrl* и *Shift* на редактирование объектов в CorelDRAW?

5. Как выполняются вращение и перекося объектов в CorelDRAW?

6. Какие возможности существуют в CorelDRAW по изменению контура?

7. Каким образом в CorelDRAW можно установить режим, когда при изменении размера объекта пропорционально меняется толщина контура?

8. Как в CorelDRAW могут выполняться различные виды заливки?

9. Какие типы текста существуют в CorelDRAW и как они создаются и редактируются?

10. Какие возможности существуют в CorelDRAW по работе с растровыми изображениями?

11. Поясните, каким образом выполняются и редактируются в CorelDRAW эффекты перетекания одного объекта в другой, подobia, интерактивного искажения, оболочек, перспективы и тени.

12. Для чего предназначен инструмент *Прозрачность* в CorelDRAW?

13. Как в CorelDRAW можно заблокировать объекты, и в каком случае это бывает полезно?

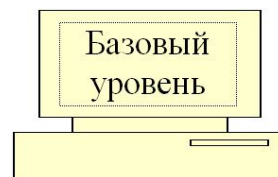
14. Какие возможности существуют в CorelDRAW по использованию готовых рисунков?

15. Для чего предназначены слои, и какие для них существуют настройки в CorelDRAW?

Глава 7. Практикум по использованию системы Mathcad в инженерных расчетах

7.1. Решение простейших задач и построение графиков средствами Mathcad

1. Внимательно ознакомиться с разделами 4.1–4.2 данного учебного пособия. Загрузить программу Mathcad и выполнить все рассмотренные в разд. 4.1–4.2 примеры, показать преподавателю полученные результаты.



2. Средствами Mathcad выполнить индивидуальные задания для варианта, указанного преподавателем.

2.1. Вычислить значение числового выражения

$$\left\{ \frac{8,8077}{20 - [28,2 : (13,333 \cdot 0,3 + 0,0001)] \cdot 2,004} + 4,9 \right\} \cdot \frac{5}{32}.$$

2.2. Решить квадратное уравнение $ax^2 + bx + c = 0$ для значений a , b и c , указанных в табл. 7.1, используя выражение для дискриминанта $\Delta = b^2 - 4ac$.

Таблица 7.1

Номер варианта	a	b	c	Номер варианта	a	b	c
1	2	–8	–3	11	–3	8	5
2	–1,1	–7	1	12	1,5	3	11
3	1	5	–1,18	13	10	15	5
4	–1,3	–6	1,2	14	4,5	–22	12
5	–7	15	2	15	–9	2	32
6	–5	–10	3,3	16	1,5	–4	–4,3
7	3	11	–3,1	17	6	21	–9
8	–2,3	–9	2,5	18	–31	–19	21
9	2,5	8	–1,9	19	6,6	8,8	4,5
10	3,8	–12	–4,5	20	33	–19	–44

2.3. Даны $x = 1,5$; $y = 2$; $z = 3$. Вычислить a , b из табл. 7.2.

Таблица 7.2

Номер варианта	a	b
1	$a = \frac{\sqrt{ x-1 } - \sqrt[3]{ z }}{1 + \frac{x^2}{2,5} + \frac{y^2}{4}}$	$b = \frac{x}{y} \operatorname{arctg}(z + \pi/6)$
2	$a = \frac{3,5 + e^{y-1}}{1 + x^2 y - \operatorname{tg} z }$	$b = y - x + \frac{(y - x)^2}{2} + \frac{ y - x ^3}{3}$
3	$a = \frac{z + y/(x^2 + 4)}{e^{-x-2}/(x^2 + 4)}$	$b = \frac{1 + \cos(y - 2)}{x^2/2 + \sin^2 z}$
4	$a = z + \frac{x}{y^2 + \left \frac{x^2}{y + x^3/1,3} \right }$	$b = \left(1 + \operatorname{tg}^3 \frac{z}{2x + 2y} \right)$
5	$a = \frac{2,3 \cos(x - \pi/6)}{1/2 + \sin^2 y} + z$	$b = x^y + \frac{z^2}{3 + z^2/5}$
6	$a = \frac{1,5 + \sin^2 z}{2 + \left x - 2x/(1 + x^2 y^2) \right }$	$b = \cos^2 \left(\operatorname{arctg} \frac{y}{z} \right) + \sin^2 x$
7	$a = \ln \left \left(y - \sqrt{ x } \right) \frac{1,5y}{z + x^2/4} \right $	$b = x - \frac{x^2}{3!} + \frac{x^5}{5!}$
8	$a = \frac{\sin(x^2 - 2y + z)}{2,6x^y}$	$b = \cos^2 \left(x^2 + \frac{y}{z} \right)$
9	$a = \frac{5 \cos(x - \pi/6)}{0,5 + \sin^2 x}$	$b = 2 + \frac{y^2}{3 + z^2/7} - 3x$
10	$a = \frac{3,5 + \operatorname{tg}(x^2 + y)}{2 + \left x - 4x/(1 + xy^2) \right }$	$b = \sin^2 \left(\operatorname{arctg} \frac{1}{z} \right)$

Номер варианта	a	b
11	$a = \frac{2,6 + \operatorname{tg}(x - y)}{2 + x - 2x/(x^2 + y^2) }$	$b = x(\operatorname{tg} z + e^{x-3})$
12	$a = \ln \left \left(2y + \sqrt{x} \right) \left(\frac{y + x^2/4}{5z} \right) \right $	$b = 1,5 + \frac{(y-x)^3}{2} + \frac{ y-x }{x}$
13	$a = \frac{\cos(x^3 + 2y - 2z)}{\operatorname{tgy} - 1,5}$	$b = \frac{1 + \sin(y-2)}{x^2 \frac{1}{2 + \sin^2 x}}$
14	$a = \frac{5 \sin(x + \pi/3)}{1/2 \cos x + 1}$	$b = \left(1 + \operatorname{tg}^2 \frac{z}{y+2} \right)$
15	$a = \frac{3 + \sin^3(x^2 + y)}{2,5 + x - 4x/(1 + x^2 y^2) }$	$b = 1 + \frac{z^2}{3 + z^2/5}$
16	$a = \frac{\sqrt{ x-1 } - \sqrt[3]{ y }}{1,5 + x^2 + y^2}$	$b = \cos^2 \left(\operatorname{arctg} \frac{1}{z+1} \right)$
17	$a = \frac{1,5 - e^{2-y}}{2x - \sqrt{y - \operatorname{tg} z}}$	$b = x + \frac{x^2}{3+z} + \frac{x^2}{5+z}$
18	$a = \frac{3,3 + y^2 + (x^2 + 2)}{e^{-0,5} + 1/(x^2 + 4)}$	$b = \cos^2 \left(x^2 + \frac{y}{1+z} \right)$
19	$a = y + \frac{3,5x}{y^2 - \sqrt{\frac{x^2}{2y + x^2}}}$	$b = 2 + \frac{y^2}{3 + \frac{z^2}{1+x}} - 3x$
20	$a = \frac{2 \cos(x - \pi/6)}{1/2 + \sin^2 y}$	$b = \sqrt{\sin \left(\operatorname{arctg} \frac{1}{x+z} \right)}$

2.4. Сформировать матрицу $B(i, j)$ (i – номер строки; j – номер столбца) по заданному алгоритму (табл. 7.3).

Таблица 7.3

Номер варианта	i	j	$B(i, j)$	Номер варианта	i	j	$B(i, j)$
1	3	6	$b_{ij} = \frac{1}{i+j-1}$	11	6	7	$b_{ij} = \frac{1+i}{2i-1}$
2	4	5	$b_{ij} = \frac{1}{i+j}$	12	7	6	$b_{ij} = \frac{i}{i+3j}$
3	5	5	$b_{ij} = \frac{i-1}{i+j}$	13	3	7	$b_{ij} = \frac{i}{1+3j}$
4	6	3	$b_{ij} = \frac{10i}{i+j}$	14	7	4	$b_{ij} = \frac{5i}{i+j}$
5	5	4	$b_{ij} = \frac{(i-j)^2}{(i+j)^2}$	15	6	5	$b_{ij} = \frac{i}{(i+j)^2}$
6	6	7	$b_{ij} = \frac{i^2}{2^{i+j}}$	16	3	6	$b_{ij} = \frac{1}{2^{i+j}}$
7	7	6	$b_{ij} = \frac{2^i}{j}$	17	4	5	$b_{ij} = \frac{2}{j^i}$
8	3	7	$b_{ij} = \frac{5i}{i+j}$	18	5	5	$b_{ij} = \frac{3+i}{i+j}$
9	7	4	$b_{ij} = \frac{i^{-2}}{3^{i+j}}$	19	6	3	$b_{ij} = \frac{i+1}{2^{i+j}}$
10	6	5	$b_{ij} = \frac{2^{i+1}}{j+2}$	20	5	4	$b_{ij} = \frac{2^i}{j+i}$

2.5. Найти произведение двух элементов матрицы $B(i, j)$, указанных в табл. 7.4, полученной при выполнении задания 7.3.

Таблица 7.4

Номер варианта	Элемент 1		Элемент 2		Номер варианта	Элемент 1		Элемент 2	
	i	j	i	j		i	j	i	j
1	3	5	1	3	11	1	6	1	6
2	7	2	2	7	12	2	4	2	4
3	2	6	3	2	13	4	6	4	6
4	5	1	4	5	14	2	7	2	7
5	4	7	5	4	15	2	5	2	5
6	3	4	6	3	16	7	5	7	5
7	2	5	7	2	17	5	3	5	3
8	5	1	8	5	18	6	4	6	4
9	1	6	9	1	19	3	4	3	4
10	6	2	10	6	20	2	7	2	7

2.6. Определите значения определенного интеграла для вариантов, указанных в табл. 7.5.

Таблица 7.5

Номер варианта	Интеграл	Номер варианта	Интеграл
1	$\int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \frac{dx}{1 + \cos x}$	5	$\int_{-\frac{\pi}{2}}^{\frac{\pi}{4}} \frac{\cos^2 x dx}{\sqrt[3]{\sin x}}$
2	$\int_0^{\frac{\pi}{2}} \cos^5 x \sin 2x dx$	6	$\int_{\frac{1}{\pi}}^{\frac{2}{\pi}} \frac{\sin \frac{1}{x} dx}{x^2}$
3	$\int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \sqrt{\cos x - \cos^2 x} dx$	7	$\int_0^1 x e^{-x} dx$
4	$\int_1^e \ln^2 x dx$	8	$\int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \cos\left(2x - \frac{\pi}{4}\right) dx$

Номер варианта	Интеграл	Номер варианта	Интеграл
9	$\int_0^{\pi} x^3 \sin x dx$	15	$\int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \cos t \sin\left(2t - \frac{\pi}{4}\right) dt$
10	$\int_0^{e-1} \ln(x+1) dx$	16	$\int_0^1 (x-1)(1-e^{-x}) dx$
11	$\int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \sqrt{\cos x - \cos^2 x} dx$	17	$\int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \frac{\cos x dx}{\sqrt{\sin x}}$
12	$\int_1^e \ln^2 x dx$	18	$\int_{\frac{1}{\pi}}^{\frac{2}{\pi}} \frac{\sin x}{x^3} dx$
13	$\int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \frac{dx}{1 + \cos x}$	19	$\int_0^{\pi} x^2 \sin 2x dx$
14	$\int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \frac{\cos x dx}{x-1}$	20	$\int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \sin\left(t + \frac{\pi}{6}\right) dt$

2.7. Постройте график функции в соответствии с вариантом задания (табл. 7.6).

Таблица 7.6

Номер варианта	Функция	Диапазон изменения x	Шаг изменения x
1	$Y = -\sin(x/2)$	$0-360^\circ$	10°
2	$Y = 13x^2$	$0-24$	8
3	$Y = \cos(4x - 60^\circ)$	$0-180^\circ$	5°
4	$Y = \frac{1}{x+4}$	$-1-3$	0,5

Номер варианта	Функция	Диапазон изменения x	Шаг изменения x
5	$Y = \frac{x+44}{x-23}$	$-8-8$	2
6	$Y = (3x^2 + 43)$	$0-40$	4
7	$Y = \operatorname{tg}(x) - 1$	$0-90^\circ$	2°
8	$Y = \frac{x}{x-31}$	$-10-30$	5
9	$Y = (x^3 - x)$	$0-200$	20
10	$\sin(4x - 45^\circ)$	$0-180^\circ$	2°
11	$\sin(3x + 60^\circ)$	$0-360^\circ$	4°
12	$1 - e^{5x}$	$0-5$	0,1
13	$\sin(5x)$	$0-360^\circ$	4°
14	$Y = \operatorname{tg}(x + 30^\circ)$	$0-90^\circ$	2°
15	$\cos(5x-15^\circ)$	$0-180^\circ$	1°
16	$Y = \frac{2x}{x+100}$	$-80-80$	10
17	$1 - e^{2x}$	$0-5$	0,1
18	$1 - x^2$	$0-1$	0,01
19	$1 + e^{4x}$	$0-5$	0,1
20	$\cos(4x)$	$0-360^\circ$	2°

3. Оформить отчет по работе со средствами Mathcad. Включить в него титульный лист и решения для всех примеров из п. 2 в виде единого документа.

4. Вычислите след (сумму диагональных элементов) для своего варианта матрицы из табл. 7.3.

Усложненный
уровень

5. Включите в отчет по работе различные варианты форматирования результатов вычислений, принятых в Mathcad, для своего задания из табл. 7.5.

6. Включите в отчет по работе для заданного графика из табл. 7.6 различные варианты настроек для графика, принятых в Mathcad.

Контрольные вопросы

1. Назовите основные элементы интерфейса системы Mathcad Prime 2.0.
2. Каким образом осуществляется формулировка математических задач средствами Mathcad Prime 2.0?
3. Какие виды блоков можно использовать в Mathcad Prime 2.0?
4. Какие типы математических выражений используются в Mathcad Prime 2.0?
5. Чем отличаются операторы вычисления, определения и равно в Mathcad Prime 2.0?
6. Какие возможности по форматированию результатов вычисления существуют в Mathcad Prime 2.0?
7. Какие функции для работы с комплексными числами существуют в Mathcad?
8. Для каких целей в Mathcad используются переменные диапазона?
9. Что представляют собой вектор, матрица и таблица Mathcad?
10. Какие средства существуют в Mathcad для задания единиц измерения?
11. Какие типы графиков, варианты задания кривой, а также возможности по изменению настроек для графика существуют в Mathcad Prime 2.0?

7.2. Численные методы решения уравнений и систем уравнений.

Программирование и символьные вычисления в системе Mathcad



1. Внимательно ознакомиться с разд. 4.3–4.5 данного учебного пособия. Загрузить программу Mathcad и выполнить все рассмотренные в разд. 4.3–4.5 примеры, показать преподавателю полученные результаты.

2. Средствами Mathcad выполнить индивидуальные задания для варианта, указанного преподавателем:

2.1. Составить программу для решения системы линейных уравнений, используя функцию *lsolve*:

$$\begin{cases} A_1x_1 + B_1x_2 + C_1x_3 = D_1; \\ A_2x_1 + B_2x_2 + C_2x_3 = D_2; \\ A_3x_1 + B_3x_2 + C_3x_3 = D_3. \end{cases}$$

Значения коэффициентов A_i , B_i , C_i и свободных членов D_i для заданного варианта взять из табл. 7.7. Выполнить проверку решения.

Таблица 7.7

Номер варианта	A_1	A_2	A_3	B_1	B_2	B_3	C_1	C_2	C_3	D_1	D_2	D_3
1	1	2	3	4	5	6	7	8	9	-10	11	12
2	2	-3	4	5	6	7	8	-9	10	11	12	13
3	3	4	-5	6	7	8	9	10	11	12	13	14
4	4	5	6	-7	8	9	10	11	12	13	14	15
5	5	6	7	8	-9	10	11	12	13	14	15	16
6	6	7	8	9	10	-11	12	13	14	15	16	17
7	7	8	9	10	11	12	-13	14	15	16	17	18
8	8	9	10	11	12	13	14	-15	16	17	18	19
9	9	10	11	12	13	14	15	16	-17	18	19	20
10	10	11	12	13	14	15	16	17	18	-19	20	21
11	11	12	13	14	15	16	17	18	19	20	-21	22
12	12	13	14	15	16	17	18	19	20	21	22	-23
13	-13	14	15	16	17	18	19	20	21	22	23	24
14	14	-15	16	17	18	19	20	21	22	23	24	25
15	15	16	-17	18	19	20	21	22	23	24	25	26
16	16	17	18	-19	20	21	22	23	24	25	26	27
17	17	18	19	20	-21	22	23	24	25	26	27	28
18	18	19	20	21	22	-23	24	25	26	27	28	29
19	19	20	21	22	23	24	-25	26	27	28	29	30
20	20	21	22	23	24	25	26	-27	28	29	30	31

2.2. Определить для заданного варианта задания все корни нелинейного уравнения с помощью функции *root*:

1. $x + \ln(x + 2) = 0$.
2. $x^4 + 2x^3 - x - 1 = 0$.
3. $\frac{2\sin^2 x}{3} - \frac{3\cos^2 x}{4} = 0$.
4. $x^2 - \sin 5x = 0$.
5. $1,8x^4 - \sin 10x = 0$.
6. $x - 0,2\sin(x + 0,5)$.
7. $4\cos\frac{x}{2} + \cos 4x = 0$.
8. $2\cos x \cdot \sin(\frac{\pi}{4} + 2x) = 0$.
9. $0,4x^2 - 1,4x - 1,2 = 0$.
10. $\ln(2x - 2) - 5x = 0$.
11. $x^5 - x - 0,2 = 0$.
12. $x^3 - 0,2x^2 - 0,2x - 1,2 = 0$.
13. $4x + 8x^3 - 4x^2 - 14x = 0$.
14. $x^4 - 4x^3 + x^2 - 5x + 4 = 0$.
15. $x^3 - 2x^2 + x - 3 = 0$.
16. $2\sin^2 - 3x = 0$.
17. $1 - \frac{1}{4}\sin^2 2x + \cos 2x = 0$.
18. $(\cos 4x + 4\cos 2x + 3) = 0$.
19. $5x^3 + 12x^2 + 3x + 4 = 0$.
20. $4x - \sin 5x = 0$.

2.3. Составить для заданного варианта задания программу для решения системы нелинейных уравнений, используя функцию *find* или *minerr*:

1. $\begin{cases} 10,5x^2 + 1 = 0 \\ 5(x + 1)^3 = 0 \end{cases}$
2. $\begin{cases} \cos x + 1 = 0 \\ \sin(x + 0,5) = 0 \end{cases}$
3. $\begin{cases} x^4 + 12 = 0 \\ (2x + 1)^3 = 0 \end{cases}$
4. $\begin{cases} x^2 + 5x = 0 \\ (x^2 + 3x)^2 = 0 \end{cases}$
5. $\begin{cases} (x + 1)^2 = 0 \\ \sqrt{x + 10} = 0 \end{cases}$
6. $\begin{cases} (2x + 1)^3 = 0 \\ \sqrt{\frac{x+11}{x}} = 0 \end{cases}$
7. $\begin{cases} 5(x + 1)^2 = 0 \\ \sqrt{x^3 - 1} = 0 \end{cases}$
8. $\begin{cases} (x + 2)^2 = 0 \\ (x^3 + 2x) = 0 \end{cases}$
9. $\begin{cases} \cos(x + 1) = 0 \\ \sin 2x = 0 \end{cases}$
10. $\begin{cases} (x + 1)^2 = 0 \\ \sqrt{x + 10} = 0 \end{cases}$
11. $\begin{cases} 5(x + 1)^3 = 0 \\ \sqrt{x^2 + 1} - x = 0 \end{cases}$
12. $\begin{cases} \cos(x - 1) = 0 \\ \cos x + 1 = 0 \end{cases}$
13. $\begin{cases} x^3 + 25 = 0 \\ (3x + 1)^2 = 0 \end{cases}$
14. $\begin{cases} (x^2 + 3x)^2 = 0 \\ (x - 2x^2) = 0 \end{cases}$

$$15. \begin{cases} 0,5x^3 + 1 = 0 \\ (x + 1)^2 = 0 \end{cases}$$

$$18. \begin{cases} x^2 + 1 = 0 \\ (x + 2)^2 = 0 \end{cases}$$

$$16. \begin{cases} x^4 - 15 = 0 \\ (2x + 1)^3 = 0 \end{cases}$$

$$19. \begin{cases} \sin 3x = 0 \\ \cos(x + 1) = 0 \end{cases}$$

$$17. \begin{cases} 15x + 1 = 0 \\ 5(x + 1)^2 = 0 \end{cases}$$

$$20. \begin{cases} 0,5x^3 + 1 = 0 \\ (x + 1)^2 = 0 \end{cases}$$

2.4. Используя условную функцию *if*, постройте график ступенчатой функции для заданного варианта из табл. 2.8.

2.5. Используя программный модуль, вычислите значение функции для заданного варианта из табл. 2.9. Значения x меняются от минус 0 до 20 с шагом 2.

2.6. Используя программный модуль, вычислить выражения для заданного варианта из табл. 7.8

Таблица 7.8

Номер варианта	Выражение	Номер варианта	Выражение
1	$\sum_{i=1}^{100} \sum_{j=1}^{50} \frac{1}{i + j^2}$	7	$\sum_{i=1}^{50} \sum_{j=1}^{10} \cos(i^2 + j^3)$
2	$\sum_{i=1}^{100} \sum_{j=1}^{60} \sin(i^3 + j^4)$	8	$\sum_{i=1}^{100} \sum_{j=1}^{100} \frac{j + i - 1}{2i + j}$
3	$\sum_{i=1}^{100} \sum_{j=1}^{100} \frac{j - i + 1}{i + j}$	9	$\sum_{i=1}^{50} \sum_{j=1}^{60} \sin(i^2 + j^3)$
4	$\sum_{i=1}^{100} \sum_{j=1}^i \frac{1}{i + 2j}$	10	$\sum_{i=1}^{20} \sum_{j=1}^{20} 5 \cos(i^3 + j^2)$
5	$\sum_{i=1}^{10} i^3 \sum_{j=1}^{15} (i - j)^2$	11	$\sum_{i=1}^{30} \sum_{j=i}^{20} \frac{2x + i}{0,5j}$ для $x=3,2$
6	$\sum_{i=1}^{20} \sum_{j=1}^{20} \frac{j - i + 2}{i + 2j}$	12	$\sum_{i=1}^{20} \sum_{j=i}^{60} \frac{x + 3i}{3,5j}$ для $x=4,3$

Номер варианта	Выражение	Номер варианта	Выражение
13	$\sum_{i=1}^{30} \sum_{j=i}^{50} \frac{x+3j}{i+j}$ для $x=1,5$	17	$\sum_{i=1}^{20} \sum_{j=i}^{60} \frac{2x-i}{j+1}$ для $x=1,3$
14	$\sum_{i=1}^{25} \sum_{j=i}^{30} \frac{x^2+j}{2i}$ для $x=1,2$	18	$\sum_{i=1}^{40} \sum_{j=i}^{100} \frac{x+i}{j+3}$ для $x=2,3$
15	$\sum_{i=1}^{50} \sum_{j=i}^{50} \frac{x+i}{j}$ для $x=1,23$	19	$\sum_{i=1}^{10} \sum_{j=i}^{70} \frac{3x-i}{j-6}$ для $x=3,3$
16	$\sum_{i=1}^{10} \sum_{j=i}^{40} \frac{x+i^2}{j-10}$ для $x=2,2$	20	$\sum_{i=1}^{15} \sum_{j=i}^{35} \frac{x-2}{j+2i}$ для $x=1,7$

2.7. Для заданного варианта из табл. 7.9 упростите или разложите на множители выражение.

Таблица 7.9

Номер варианта	Операция	Выражение
1	<i>simplify</i>	$\sqrt{x^2+4x+4} + \sqrt{x^2-2x+1} + \sqrt{x^2-6x+9}$
2	<i>factor</i>	a^4+4b^4
3	<i>simplify</i>	$\frac{x^2-1}{x^2+y^2} - \frac{(x-y)^2}{4x^4-y^4}$
4	<i>factor</i>	$xa-xb^2-ay+zb^2-za+yb^2$
5	<i>simplify</i>	$y^3(a-x)-x^3(a-y)+a^3(x-y)$
6	<i>factor</i>	x^2+4x+4
7	<i>simplify</i>	$\sqrt[3]{b^2c^4} + \sqrt[3]{a^4c^2} - \left(\sqrt[3]{c^2b^4} + \sqrt[3]{a^2c^4} + \sqrt[3]{a^4b^2} \right)$
8	<i>factor</i>	$a^3x^2-a^2+a^2x^2-16$
9	<i>simplify</i>	$(x+1)(x+3)(x+5)(x+7)+15$
10	<i>factor</i>	$25x^2-10xy+y^2$

Номер варианта	Операция	Выражение
11	<i>simplify</i>	$\sqrt{\frac{x}{x-a^2}} \cdot \left(\frac{\sqrt{x}-\sqrt{x-a^2}}{\sqrt{x}+\sqrt{x-a^2}} - \frac{\sqrt{x}+\sqrt{x-a^2}}{\sqrt{x}-\sqrt{x-a^2}} \right)$
12	<i>factor</i>	$x^2 - 3x + 2$
13	<i>simplify</i>	$\left(\frac{\sqrt{x}-\sqrt{x-a^2}}{\sqrt{x}+\sqrt{x-a^2}} - \frac{\sqrt{x}+\sqrt{x-a^2}}{\sqrt{x}-\sqrt{x-a^2}} \right)$
14	<i>factor</i>	$ab - ac + yb - yc$
15	<i>simplify</i>	$\frac{\frac{1}{x^2} + 1}{x + \frac{1}{x^2} + 1} \cdot \frac{1}{x^{1,5} - 1}$
16	<i>factor</i>	$ab - ac + yb - yc$
17	<i>simplify</i>	$\sqrt{\frac{x}{x-a^2}} \cdot \left(\frac{\sqrt{x}-\sqrt{x-a^2}}{1+\sqrt{x-a^2}} - \frac{\sqrt{x}}{\sqrt{x}-\sqrt{x-a^2}} \right)$
18	<i>factor</i>	$ab - ac + yb - yc$
19	<i>simplify</i>	$\frac{n+2+\sqrt{n^2-4}}{n+2-\sqrt{n^2-4}} + \frac{n+2-\sqrt{n^2-4}}{n+2+\sqrt{n^2-4}}$
20	<i>factor</i>	$ab - ac + yb - yc$

2.8. Найти пределы для заданного варианта задания.

1. $\lim_{x \rightarrow 0} \frac{1 - \cos^2 x}{x \sin 2x} 0.$

2. $\lim_{x \rightarrow -1} \frac{(x+1)^2}{\sqrt{x+1}}.$

3. $\lim_{x \rightarrow 0} \frac{\cos^2 x}{x \sin 2x}.$

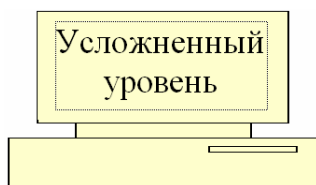
4. $\lim_{x \rightarrow 0} \frac{\sqrt{1+2\sin x} - \sqrt{1-2\sin x}}{2\sin x}.$

5. $\lim_{x \rightarrow \pi} \frac{1 - \sin \frac{x}{2}}{\cos \frac{x}{2} \left(\cos \frac{x}{4} - \sin \frac{x}{4} \right)}.$
6. $\lim_{x \rightarrow \pi} \frac{\sin x}{\cos x (\cos x - \sin x)}.$
7. $\lim_{x \rightarrow 0} \frac{e^x - e^{-x}}{\sin x}.$
8. $\lim_{x \rightarrow 0} \left(\frac{\sin x}{x} \right)^{\frac{\sin x}{x - \sin x}}.$
9. $\lim_{x \rightarrow 0} \frac{1 - \cos x}{2x \sin 2x}.$
10. $\lim_{x \rightarrow 0} \frac{\sqrt{1 + \sin x} - \sqrt{1 - \sin x}}{\operatorname{tg} x}$
11. $\lim_{x \rightarrow \infty} x^{\frac{3}{2}} \left(\sqrt{x^2 + 1} - \sqrt{x^2 - 1} \right).$
12. $\lim_{x \rightarrow \infty} \left(\frac{x^2 - 2x + 1}{x^2 - 4x + 2} \right)^x.$
13. $\lim_{x \rightarrow \infty} x^2 \left(1 - \cos \frac{1}{x} \right).$
14. $\lim_{x \rightarrow \pi/6} \frac{1 - \sin 4x}{x}.$
15. $\lim_{x \rightarrow \pi} \frac{\sin 3x}{3x}.$
16. $\lim_{x \rightarrow 0} \frac{\operatorname{tg} 2x}{2x^2}.$
17. $\lim_{x \rightarrow 0} \frac{\operatorname{ctg}(x) (1 - \cos(3x)^2)}{x^2 + 5x}.$
18. $\lim_{x \rightarrow 0} \frac{1 - \cos(4x)}{5x}.$

$$19. \lim_{x \rightarrow 0} \frac{\ln \cos x}{x^2}.$$

$$20. \lim_{x \rightarrow \infty} \left(\frac{x-2}{x+1} \right)^{2x+3}.$$

3. Оформить отчет по работе средствами Mathcad. Включить в него титульный лист и решения для всех примеров из п. 2 в виде единого документа.



4. Разработать в *Mathcad* программный модуль для вычисления и вывода на экран монитора наибольшего и наименьшего значения функции (табл. 2.10) и значения аргумента, при котором оно получено. Вывести полученные результаты с точностью два знака после запятой. Включите в отчет по работе полученный программный модуль.

5. Разработать в *Mathcad* программный модуль по обработке заданной матрицы в соответствии с вариантом задания (табл. 2.11). Включить в отчет по лабораторной работе полученный программный модуль.

Контрольные вопросы

1. Какие функции Mathcad предназначены для решения систем линейных и нелинейных уравнений?
2. Какие функции могут использоваться в блоке решения и для чего они предназначены?
3. Назовите программные операторы Mathcad.
4. Чем отличается локальная переменная программного модуля от переменных в общем документе Mathcad?
5. Чем отличается оператор *else if* от оператора *also if*?
6. Какими средствами можно организовать в Mathcad циклы с неизвестным количеством повторений и с заданным количеством повторений?
7. Каким образом можно организовать вложенные циклы?
8. Перечислите основные возможности символьной математики в Mathcad.
9. Каким образом можно задать упрощение выражения?
10. Какие вычисления в Mathcad выполняются точнее: числовые или символьные?

11. Каким образом можно получить значение числа π с точностью 20 знаков после запятой?

12. Каким образом можно выполнить разложение целых чисел, полиномов и рациональных функций в произведение более простых элементов?

13. Каким образом можно разложить выражения путем перемножения элементов произведения и возведения их в степень?

14. Каким образом можно разложить выражения путем применения тождеств для общих функций?

15. Для чего в Mathcad предназначены ключевые слова *collect* и *parfrac*?

Заключение

В настоящее время при решении любых научных, конструкторских и технологических задач невозможно обойтись без знания современных компьютерных технологий. Последние в данном учебном пособии рассматриваются на примере среды программирования VBA, универсальной для Microsoft Office, графического пакета CorelDraw и среды инженерных расчетов Mathcad. Освоение указанных пакетов сопровождается практическими разделами, изучение которых позволит закрепить теоретические знания.

Рассмотренные разделы пособия, по мнению авторов, позволят читателям получить достаточный базис знаний, умений и навыков, позволяющий при необходимости уверенно осваивать различные CAD-CAM пакеты в профессиональной специализированной работе.

Безусловно, авторы при написании данного пособия не ставили своей задачей создать универсальный методический материал, который бы позволил заместить всю рекомендованную при изучении дисциплины «Компьютерные технологии» литературу. Вместе с тем материал, включенный в данное пособие, охватывает главные направления использования современных средств компьютерного проектирования в инженерной практике.

К сожалению, ограниченный объем пособия не позволил в должной мере рассмотреть ряд тем, не менее важных для подготовки выпускников бакалавриата. Знания представленных в учебном пособии аспектов использования компьютерных технологий важны для специалистов в области электроэнергетики и электротехники разных профилей, но прежде всего ими должны овладеть проектировщики изделий электротехники и электроэнергетики, работники производственной сферы, особенно разработчики современных технологий производства, которые по-новому, комплексно, могут использовать возможности и преимущества современных компьютерных технологий в качестве весомых составляющих конкурентоспособности продукции.

Предлагаемое учебное пособие адресовано обучающимся, студентам и широкому кругу специалистов, изучающих современные компьютерные технологии, современные методы автоматизации инженерных расчетов и визуализации проектных задач.

Авторы надеются, что студенты и другие группы читателей, освоив материалы данного учебного пособия, используя дополнительную литературу и приобретя соответствующие навыки практической работы, смогут самостоятельно решать возникающие проектные задачи и на основе полученной базовой подготовки освоить профессиональные пакеты инженерного проектирования.

Список литературы

1. Справочник VBA. – URL: <http://vba-help.ru>
2. Регеда, В. В. Основы алгоритмизации : учеб. пособие / В. В. Регеда, О. Н. Регеда. – Пенза : Изд-во ПГУ, 2008.
3. ГОСТ 19.701–90. Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения. – М. : Изд-во стандартов, 1991.
4. Окулов, С. М. Программирование в алгоритмах / С. М. Окулов. – 2-е изд., испр. – М. : БИНОМ. Лаборатория знаний, 2006.
5. Эволюция языков программирования. Классификация языков программирования. – URL: http://www.urtt.ru/bib/dataindex/oaip/lection/_html/lect_05.htm
6. Обзор языков программирования. – URL: <http://bourabai.kz/alg/classification04.htm>
7. Информатика. Базовый курс / С. В. Симонович [и др.]. – СПб., 2000.
8. Кузьменко, В. Г. VBA 2000 / В. Г. Кузьменко. – М. : БИНОМ, 2000.
9. Слепцова, Л. Д. Программирование на VBA в Microsoft Office: самоучитель / Л. Д. Слепцова. – М. : Диалектика, 2010.
10. Регеда, В. В. Основы программирования на VBA : учеб. пособие / В. В. Регеда, О. Н. Регеда. – Пенза : Изд-во ПГУ, 2004.
11. Язык визуального программирования приложений Visual Basic for Application. – URL: <http://bourabai.kz/einf/vba/index.htm>
12. Комолова, Н. В. Самоучитель CorelDRAW X6 / Н. В. Комолова. – СПб. : БХВ-Петербург, 2012.
13. Гурский, Д. Вычисления в MathCAD 12 / Д. Гурский, Е. Турбина. – СПб. : Питер, 2006.

Приложение А

Справочная информация об объектах VBA

Таблица П.А.1

Наиболее часто используемые свойства форм

Свойство	Описание
<i>Name</i>	Возвращает или устанавливает имя пользовательской формы
<i>Caption</i>	Возвращает или устанавливает текст, отображаемый в строке заголовка формы
<i>Appearance</i>	Возвращает или устанавливает вид формы в период выполнения кода. Допустимые значения: • 0 (плоский); 1 (объемный)
<i>BorderStyle</i>	Возвращает или устанавливает только на этапе конструирования тип границы. Допустимые значения: • VbBSNone или 0 (нет границы, причем заголовок формы не отображается); • VbFixedSingle или 1 (пользователь не может изменять размер окна); • VbSizable или 2 (используется по умолчанию, пользователь может изменять размер окна); • VbFixedDouble или 3 (рамка формы имеет двойную толщину); • VbFixedToolWindow или 4 (используется для вывода окна с кнопкой Close, уменьшается шрифт в заголовке окна, пользователь не может изменять размер окна); • VbSizableToolWindow или 5 (как в предыдущем случае, но пользователь может изменять размер окна)
<i>MaxButton</i> , <i>MinButton</i>	Логические свойства, доступные только для чтения, которые при значении, равном True, указывают на присутствие в форме кнопок Maximize и Minimize соответственно
<i>ControlBox</i>	Логическое свойство, доступное только для чтения, которое при значении, равном True, указывает на присутствие в форме раскрывающегося меню, содержащее команды восстановления, перемещения, изменения размеров и закрытия формы
<i>BackColor</i> , <i>ForeColor</i>	Возвращают и устанавливают цвет фона и текста формы. Цвета зашифровываются в шестнадцатеричной системе счисления. Для удобства работы с цветами часто вместо их шестнадцатеричного представления используются константы, встроенные в Visual Basic

Свойство	Описание
<i>Icon</i>	Устанавливает или возвращает значок, отображаемый при минимизации окна формы. При программной установке данного свойства используется функция <i>LoadPicture</i> , в параметре которой приводится ссылка на полное имя файла, используемого в качестве значка. В Visual Basic имеется богатая коллекция стандартных значков, располагаемая в папке <i>GraphicsMcons</i> . В следующем примере в качестве значка отображается российский флаг: <i>Icon = LoadPicture ("\\Common\\Graphics\\Icons\\Flags_Flgrus.ico")</i>
<i>Moveable</i>	Логическое свойство, которое определяет, может ли пользователь передвигать форму
<i>WindowState</i>	Возвращает или устанавливает вид формы во время выполнения кода. Допустимые значения: • <i>VbNormal</i> или 0 (нормальный); • <i>VbMinimized</i> или 1 (в виде значка); • <i>VbMaximized</i> или 2 (полноэкранный).
<i>FillColor</i>	Возвращает или устанавливает цвет заполнения графического изображения, созданного методами <i>Line</i> и <i>Circle</i>

Таблица П.А.2

Наиболее часто используемые методы формы

Метод	Описание
<i>Show</i>	Отображает форму на экране
<i>Hide</i>	Закрывает форму
<i>Move</i>	Изменяет местоположение и размер формы
<i>PrintForm</i>	Печатает изображение формы

Таблица П.А.3

Наиболее часто используемые инструкции формы

Инструкция	Описание
<i>Load</i>	Загружает форму или элемент управления в память. Синтаксис: <i>Load object</i>
<i>Unload</i>	Выгружает форму или элемент управления с экрана и из памяти. Синтаксис: <i>Unload object</i>
<i>End</i>	Завершает выполнение кода без генерации событий

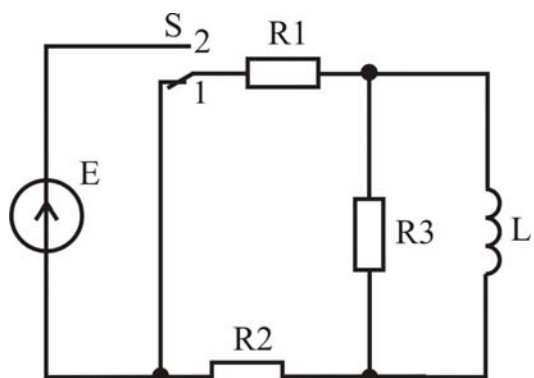
Список основных элементов управления

Элементы управления	Кнопка
<i>TextBox</i> – Поле ввода	
<i>Label</i> – Надпись	
<i>CommandButton</i> – Кнопка	
<i>ListBox</i> – Список	
<i>ComboBox</i> – Комбинированное поле	
<i>OptionButton</i> – Переключатель	
<i>CheckBox</i> – Флажок	
<i>Frame</i> – Рамка	
<i>HScrollBar</i> – Горизонтальная полоса прокрутки	
<i>VScrollBar</i> – Вертикальная полоса прокрутки	
<i>Image</i> – Рисунок	
<i>SpinButton</i> – Счетчик	

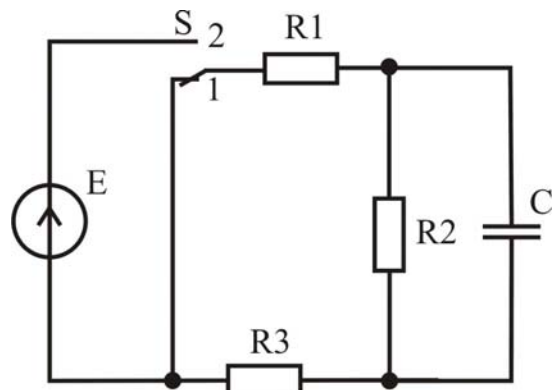
Некоторые общие для форм и элементов управления события

Событие	Описание
<i>Initilize</i>	Происходит во время конфигурирования формы, но до ее загрузки
<i>Load</i>	Происходит после инициализации формы, но до ее отображения на экран
<i>Unload</i>	Событие противоположное <i>Load</i> . Обычно используется для того, чтобы уточнить, действительно ли пользователь желает закрыть форму
<i>QueryUnload</i>	Событие, происходящее перед событием <i>Unload</i> . Также используется для того, чтобы создать запрос у пользователя, действительно ли он хочет закрыть окно
<i>Terminate</i>	Происходит при закрытии формы
<i>Click</i> <i>DbClick</i>	Происходят при щелчке и двойном щелчке в форме
<i>MouseDown</i> <i>MouseUp</i>	Происходят при нажатии кнопки мыши и ее отпуске
<i>KeyDown</i> <i>KeyUp</i>	Происходят при нажатии и отпуске клавиши
<i>Activate</i> <i>Deactivate</i>	Происходят при активизации и деактивизации формы
<i>Resize</i>	Происходит при изменении размеров формы. Обычно используется для переопределения размеров элементов управления при изменении размеров формы так, чтобы сохранить пропорцию их размеров и местоположение

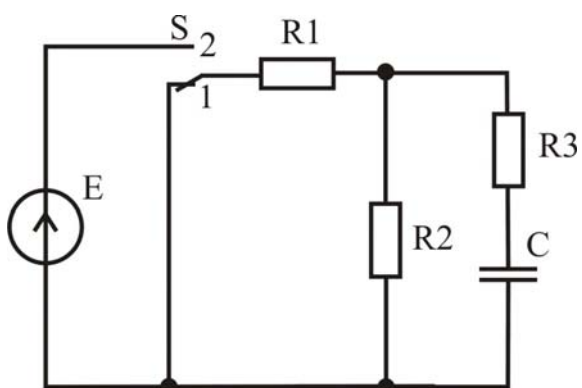
Приложение Б **Индивидуальные задания** **по CorelDRAW № 1**



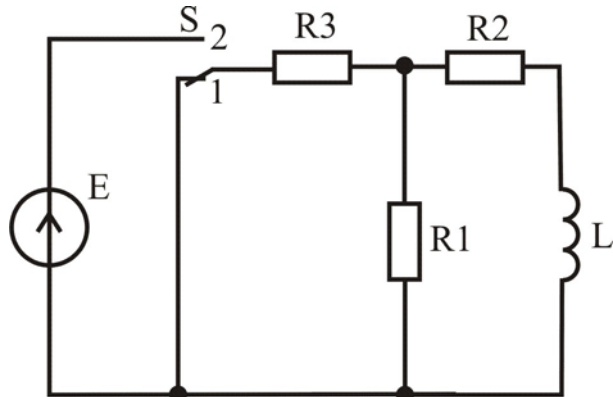
Вариант 1



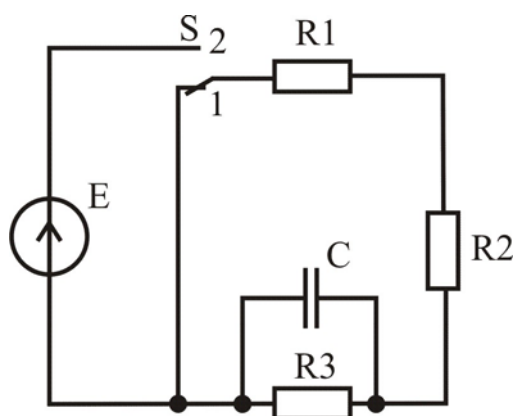
Вариант 2



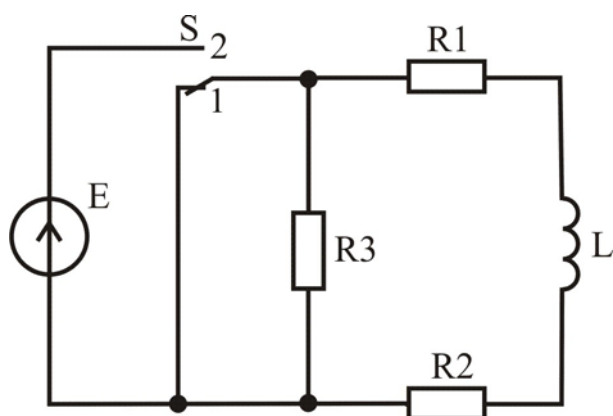
Вариант 3



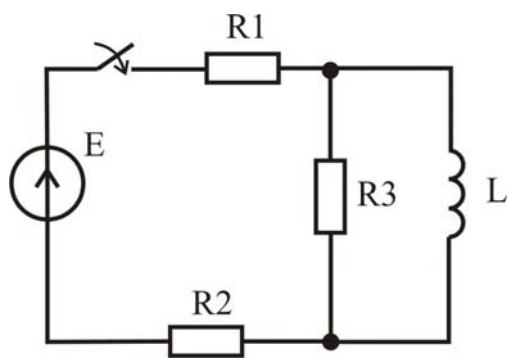
Вариант 4



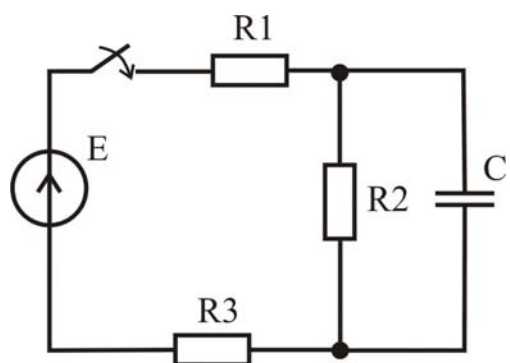
Вариант 5



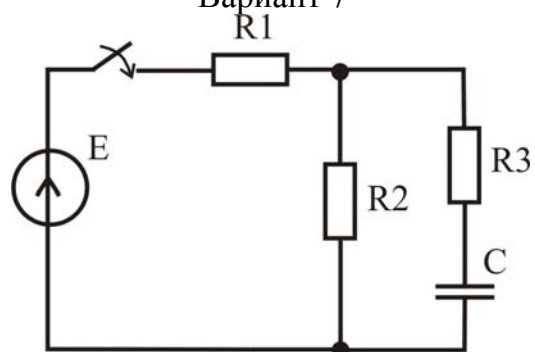
Вариант 6



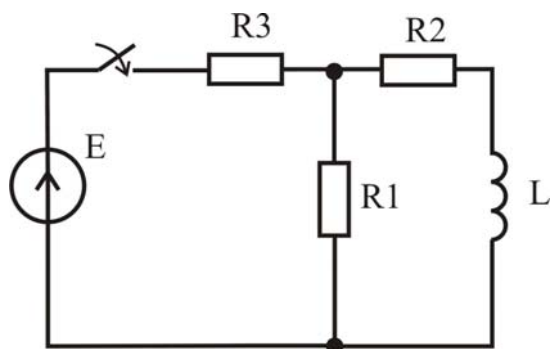
Вариант 7



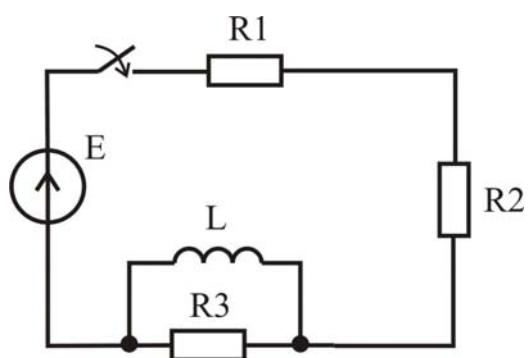
Вариант 8



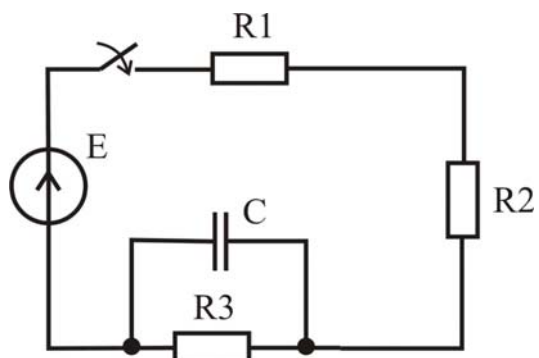
Вариант 9



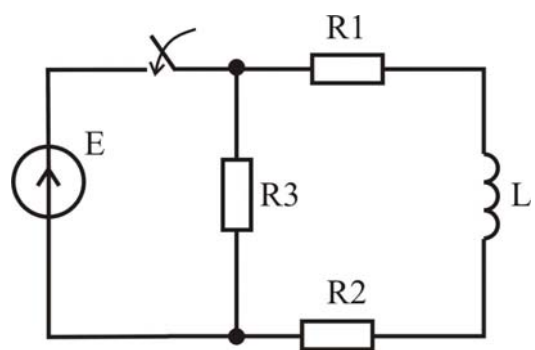
Вариант 10



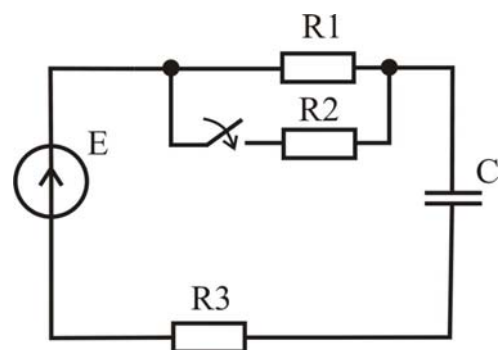
Вариант 11



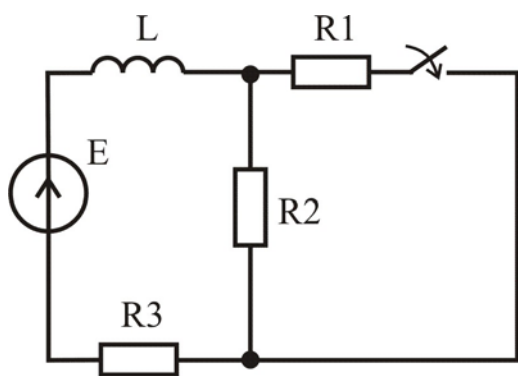
Вариант 12



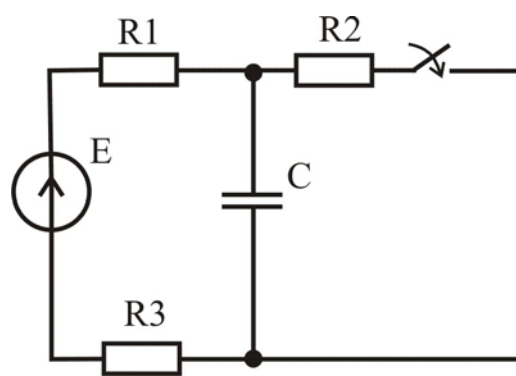
Вариант 13



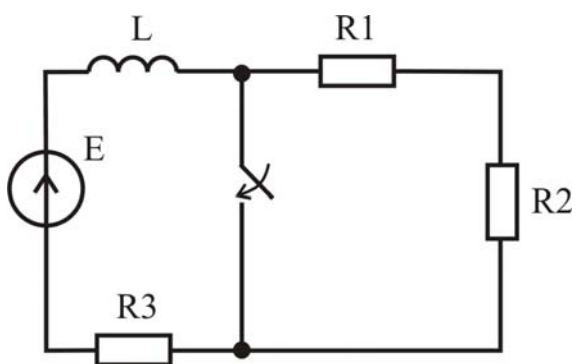
Вариант 14



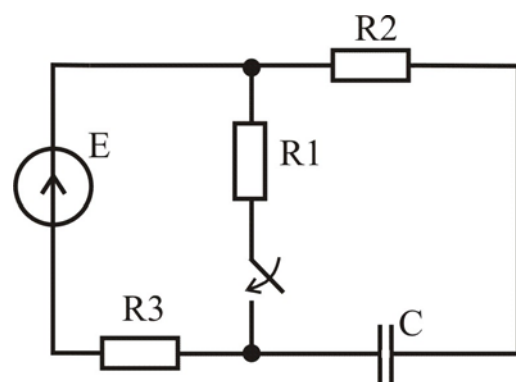
Вариант 15



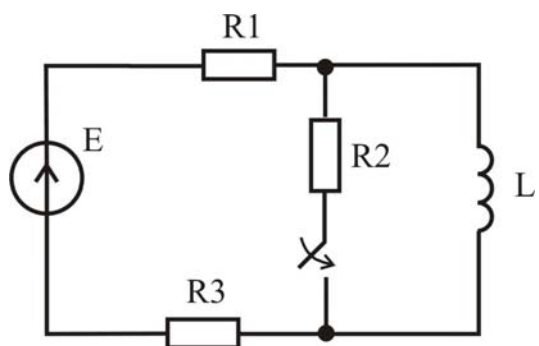
Вариант 16



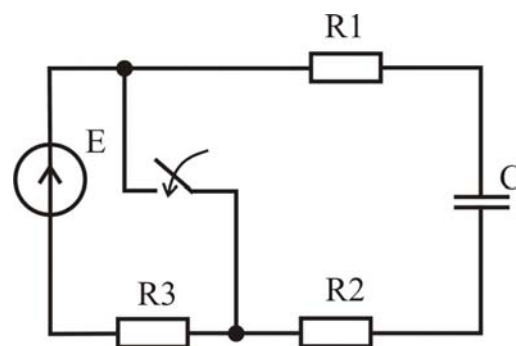
Вариант 17



Вариант 18



Вариант 19



Вариант 20

Приложение Г

Индивидуальные задания по CorelDRAW № 3



Вариант 1



Вариант 2



Вариант 3



Вариант 4



Вариант 5



Вариант 6



Вариант 7



Вариант 8



Вариант 9



Вариант 10



Вариант 11



Вариант 12



Вариант 13



Вариант 14



Вариант 15



Вариант 16



Вариант 17



Вариант 18



Вариант 19



Вариант 20



Вариант 21



Вариант 22



Вариант 23



Вариант 24



Вариант 25



Вариант 26



Вариант 27



Вариант 28



Вариант 29



Вариант 30



Вариант 31



Вариант 32



Вариант 33



Вариант 34



Вариант 35



Вариант 36



Вариант 37



Вариант 38



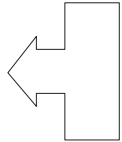
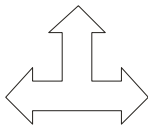
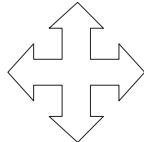
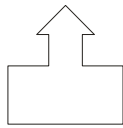
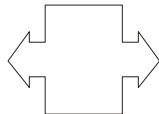
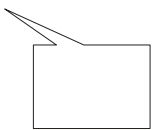
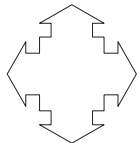
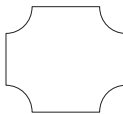
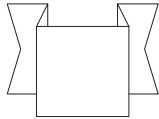
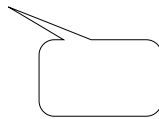
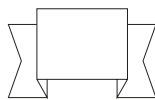
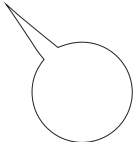
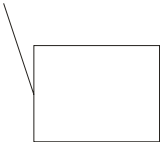
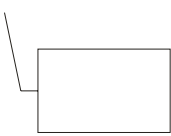
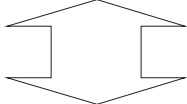
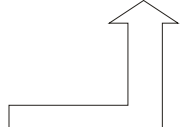
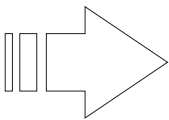
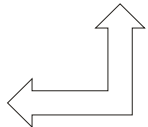
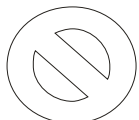
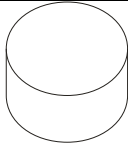
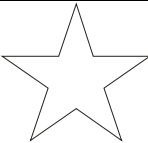
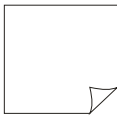
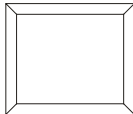
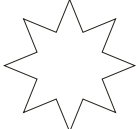
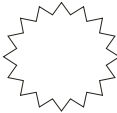
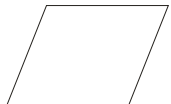
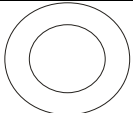
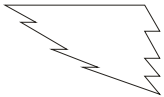
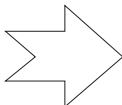
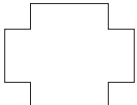
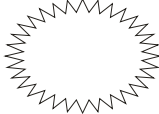
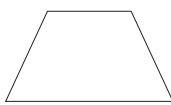
Вариант 39



Вариант 40

Приложение В

Индивидуальные задания по CorelDRAW № 2

Номер варианта	Стан- дартная фигура	Номер варианта	Стан- дартная фигура	Номер варианта	Стан- дартная фигура	Номер варианта	Стан- дартная фигура
1		2		3		4	
5		6		7		8	
9		10		11		12	
13		14		15		16	
17		18		19		20	
21		22		23		24	
25		26		27		28	
29		30		31		32	
33		34		35		36	
37		38		39		40	

Приложение Д

Методические указания

к выполнению курсовой работы

по дисциплине Компьютерные технологии

(часть 1)

1 Задание на курсовую работу предусматривает:

- разработку алгоритма заполнения и обработки массива;
- разработку интерфейса программы;
- разработку программного кода в среде VBA (в программе необходимо предусмотреть два варианта заполнения исходной матрицы: из таблицы Excel и с помощью генератора случайных чисел, а также вывод инструкции пользователя с помощью панели сообщений);
- оформление пояснительной записки в текстовом редакторе Microsoft Word с применением графического редактора CorelDRAW.

2 На защиту курсовой работы необходимо представить разработанную в соответствии с вариантом задания программу и продемонстрировать ее работу, а также пояснительную записку.

Во время защиты студент должен продемонстрировать понимание материала, изложенного в пояснительной записке и ответить на теоретические вопросы по теме курсовой работы.

3 Пояснительная записка должна включать в себя следующие разделы (см. образцы ниже):

- титульный лист;
- задание на курсовую работу;
- содержание;
- общие сведения о программе (обозначение и наименование программы, программное обеспечение, необходимое для функционирования программы, языки программирования, на которых она написана);
- функциональное назначение программы (классы решаемых задач и (или) назначение программы и сведения о функциональных ограничениях на применение);
- описание логической структуры (алгоритм программы, используемые методы, структура программы с описанием функций основных частей и связи между ними, связи программы с другими программами);

- используемые технические средства (обеспечивающие работу программы);
- вызов и загрузка программы (способы вызова программы с соответствующего носителя данных, входные точки в программу, при необходимости сведения об использовании оперативной памяти, объем программы);
- входные и выходные данные (характер и организация, формат, описание и способ кодирования входных и выходных данных);
- руководство оператора (сведения для обеспечения процедуры общения оператора с вычислительной системой в процессе выполнения программы).

В приложениях необходимо поместить текст разработанной программы; нарисованную по ГОСТ 19.701–90 схему алгоритма работы программы и результаты работы программы.

4 В данном разделе приводится примерное содержание разделов пояснительной записки.

1 Общие сведения о программе

Программа по обработке матрицы находится непосредственно в файле документа «Пример КР.doc». Программа написана на языке Visual Basic for Application для приложения MS Word, входящего в комплект пакета MS Office XP.

2 Функциональное назначение программы

Программа (приложение А) демонстрирует алгоритм заполнения матрицы размерностью 4×5 одним из двух способов: с помощью генератора случайных чисел и из таблицы Excel. Программа позволяет посчитать количество отрицательных чисел в матрице, заменить эти числа нулями, а также вывести преобразованную матрицу.

3 Описание логической структуры

Логическую структуру программы поясняет схема алгоритма программы, приведенная в приложении Б.

Структура проекта включает в себя форму UserForm1, с размещенными на ней элементами управления и связанными с ней процедурами обработки событий.

После запуска программы выполняется процедура активизации формы (Private Sub UserForm_Activate), с помощью которой

в текстовом окне со списком автоматически формируются варианты заполнения матрицы: с помощью генератора случайных чисел «RND» и с помощью таблицы «Excel».

При нажатии на клавишу «Инструкция пользователя» запускается процедура (Private Sub CommandButton3_Click), которая выводит на экран в соответствующие текстовые окна необходимые пояснения по работе с программой.

При нажатии на клавишу «Заполнить исходную матрицу» запускается процедура (Private Sub CommandButton1_Click), в которой в зависимости от выбранного варианта заполнения матрицы срабатывает одна из ветвей разветвляющегося алгоритма. В каждой из ветвей с помощью двух вложенных циклов поочередно выбираются элементы исходной матрицы и им присваиваются значения либо с помощью генератора случайных чисел, либо с помощью таблицы Excel. Затем с помощью двух вложенных циклов исходная матрица выводится в соответствующие элементы управления на форме.

При нажатии на клавишу «Заменить отрицательные элементы матрицы на нуль» запускается процедура (Private Sub CommandButton2_Click), в которой с помощью двух вложенных циклов отрицательным элементам матрицы присваивается значение, равное нулю, а также подсчитывается количество отрицательных элементов. Значения преобразованной матрицы выводятся в соответствующие элементы управления на форме.

4 Используемые технические средства

Разработка программы осуществлялась с использованием персонального компьютера, совместимого с IBM PC AT (указываются минимальные требования к персональному компьютеру).

5 Вызов и загрузка программы

Для доступа к программе необходимо загрузить текстовый редактор MS Word и открыть файл «Пример KP.doc». Перейти в редактор Visual Basic, нажав клавиши ALT+F11, и выбрать в окне проектов форму UserForm1. Для запуска программы необходимо нажать клавишу F5 или кнопку  на стандартной панели инструментов.

6 Входные и выходные данные

Входными данными программы являются:

- выбранный с помощью соответствующих элементов управления на форме способ заполнения матрицы;
- значения элементов исходной матрицы, вводимые либо из книги Excel, либо с помощью генератора случайных чисел. Кроме того, значения могут вводиться вручную в соответствующие текстовые окна.

Выходные данные программы (преобразованная матрица и количество отрицательных элементов матрицы) выводятся в соответствующие элементы управления на форме.

7 Руководство оператора

Для доступа к программе необходимо открыть файл «Пример КР.doc» (при этом необходимо, чтобы в среде MS Word было разрешено выполнение макросов) и запустить программу в соответствии с п. 5.

В результате появится окно «Курсовая работа студента группы ...» (рисунок П.Д.1)*, которое содержит 48 объектов:

- текстовое окно со списком, с помощью которого выбирается вариант заполнения матрицы;
- 40 текстовых окон для вывода значений исходной и преобразованной матрицы;
- 4 текстовых поля:
 - «Исходная матрица»;
 - «Преобразованная матрица»;
 - «Матрицу заполнить из:»;
 - «Label4», в которое будет выводиться количество отрицательных элементов матрицы.;
- 3 кнопки:
 - «Инструкция пользователя»;
 - «Заполнить исходную матрицу», при нажатии на которую, в зависимости от выбранного варианта заполнения матрицы, заполняются 20 текстовых окон исходной матрицы;
 - «Заменить отрицательные элементы матрицы на нуль», при нажатии на которую заполняются 20 текстовых окон преобразованной матрицы.

* Курсовая работа оформляется в соответствии с ЕСКД.

Курсовая работа студента группы ...

Исходная матрица

Преобразованная матрица

Матрицу заполнить из:

Заполнить исходную матрицу

Заменить отрицательные элементы матрицы на ноль

[Инструкция пользователя](#)

Рисунок П.Д.1

Результат работы программы показан на рисунке П.Д.2.

При нажатии на клавишу «Инструкция пользователя» на экране появляются текстовые окна, поясняющие следующее действие оператора.

Для завершения работы программы необходимо нажать на кнопку в виде креста в правом верхнем углу формы.

Курсовая работа студента группы ...

Исходная матрица

71	53	-58	29	-30
77	1	-76	81	71
5	41	-86	79	37
-96	87	-6	95	-36

Преобразованная матрица

71	53	0	29	0
77	1	0	81	71
5	41	0	79	37
0	87	0	95	0

Матрицу заполнить из:

Количество отрицательных элементов в матрице=7

Заполнить исходную матрицу

Заменить отрицательные элементы матрицы на ноль

[Инструкция пользователя](#)

Рисунок П.Д.2

В приложении А следует привести текст программы, например

Option Explicit

Dim i As Integer: Dim j As Integer

Dim S As Integer: Dim P As Integer

Dim k As Integer: Dim N As Integer

Dim v As String: Dim Aij As String

Dim A(4, 5) As Integer: Dim objExcel As Object

'Формирование списка

Private Sub UserForm_Activate()

ComboBox1.AddItem "RND"

ComboBox1.AddItem "Excel"

End Sub

'Заполнение исходного массива

Private Sub CommandButton1_Click()

v = ComboBox1.Text

If v = "Excel" Then

Set objExcel = CreateObject("Excel.Application")

'Открытие файла "*Книга1.xls*"

objExcel.Workbooks.Open FileName:=ThisDocument.Path _
& "Книга1.xls"

'Формирование исходной матрицы из таблицы *Excel*

For i = 1 To 4

For j = 1 To 5

A(i, j) = objExcel.Cells(i, j)

Next j

Next i

objExcel.Quit

Set objExcel = Nothing

Else

'Формирование исходной матрицы

'с помощью генератора случайных чисел

For i = 1 To 4

For j = 1 To 5

*A(i, j) = Rnd(1) * 100*

```

        If  $A(i, j) \bmod 2 = 0$  Then  $A(i, j) = -A(i, j)$ 
    Next j
Next i
End If

```

```

'Вывод исходной матрицы
    k = 0
    For i = 1 To 4
        For j = 1 To 5
            Controls(k).Text = A(i, j): k = k + 1
        Next j
    Next i
End Sub

```

```

Private Sub CommandButton2_Click()
'Вывод преобразованной матрицы
    k = 20: N = 0
    For i = 1 To 4
        For j = 1 To 5
            If  $A(i, j) < 0$  Then  $A(i, j) = 0$ : N = N + 1
            Controls(k).Text = A(i, j): k = k + 1
        Next j
    Next i

```

```

    Label4.Caption = "Количество отрицательных _
    " элементов в матрице = " + CStr(N)
End Sub

```

'Вызов инструкции пользователя

```

Private Sub CommandButton3_Click()
MsgBox "1.Выбрать вариант заполнения матрицы.", 0
MsgBox "2.Нажать клавишу Заполнить исходную матрицу.", 0
MsgBox "3.Нажать клавишу Заменить отрицательные" _
"элементы матрицы на нуль", 0
End Sub

```

В приложении Б следует привести схему алгоритма программы.

МИНОБРНАУКИ РОССИИ
ФГБОУ ВПО «Пензенский государственный университет»
Кафедра «Электротехника и транспортное электрооборудование»

Пояснительная записка
к курсовой работе
по дисциплине «Компьютерные технологии (часть 1)»
на тему «Алгоритмизация и программирование»

ПГУ 1.1-КР.140400.XX.001 ПЗ

Студент : _____ **Иванов И.И.**
Группа:
11МЭ1

Руководитель:
к.т.н., доцент _____ **Регеда О.Н.**

Работа защищена с оценкой _____

Преподаватель _____

Дата защиты _____

Пенза 2014

СОДЕРЖАНИЕ

Задание	
Содержание	
1 Общие сведения о программе	
2 Функциональное назначение программы	
3 Описание логической структуры	
4 Используемые технические средства	
5 Вызов и загрузка программы	
6 Входные и выходные данные	
7 Руководство оператора	
Список литературы	
Приложение А	
Приложение В.	

					ПГУ 1.1-КР.140400.XX.001 ПЗ			
Изм	Лист	№ документа	Подпись	Дата	Содержание			
Разраб.	Иванов И.И.							
Пров.	Регеда О.Н.							
Н. конт.								
УТВ.								
					Литера	Лист	Листов	
						3		
					Каф. ЭиТЭ гр. 12МЭ1			

СОДЕРЖАНИЕ

Введение	3
Глава 1. Алгоритм и его свойства	5
1.1. Общие требования к алгоритмам.....	5
1.2. Способы описания алгоритмов	6
1.3. Основные правила выполнения схем алгоритма.....	8
1.4. Виды алгоритмических процессов	14
1.4.1. Линейные алгоритмические процессы	15
1.4.2. Разветвляющиеся алгоритмические процессы	16
1.4.3. Циклические алгоритмические процессы.....	21
1.4.4. Задачи по обработке одномерных массивов.....	26
1.4.5. Задачи по поиску и обработке текста.....	30
1.4.6. Вложенные циклы	36
1.4.7. Итерационные циклы	45
1.4.8. Задания для самостоятельного выполнения	49
Глава 2. Программирование на языке VBA в Microsoft Office	53
2.1. Общие сведения о языках программирования	53
2.2. Объектная структура языка VBA	56
2.3. Основные этапы создания VBA-программ	59
2.4. Макросы в Microsoft Office. Редактор Visual Basic.....	59
2.4.1. Создание макросов в Microsoft Word	60
2.4.2. Интегрированная среда проектирования VBA	61
2.4.3. Анализ полученного программного кода макроса	64
2.4.4. Сохранение макроса на диске	64
2.4.5. Открытие файлов с макросами	65
2.4.6. Создание макросов в Excel	66
2.4.7. Организация проекта в среде VBA	68
2.5. Основы программирования на VBA	70
2.5.1. Основные элементы языка программирования VBA	70
2.5.2. Типы данных, константы и переменные	71
2.5.3. Арифметические выражения и функции VBA	74
2.5.4. Основные операторы языка программирования VBA для создания линейных и разветвляющихся программ.....	78
2.5.5. Операторы VBA для создания циклических программ	81
2.5.6. Циклические программы по обработке массивов	85
2.6. Использование форм и элементов управления VBA для организации интерфейса программ	87
2.6.1. Формы и элементы управления VBA.....	87
2.6.2. Пример создания приложения в Microsoft Word	89
2.6.3. Пример создания приложения в Microsoft Excel	97

2.7. Дополнительные возможности по созданию автоматизированных приложений в Microsoft Word	101
Глава 3. Применение графического редактора CorelDraw для создания иллюстраций	108
3.1. Общие сведения о графических редакторах.....	108
3.2. Знакомство с графической системой CorelDRAWX6.....	109
3.2.1. Интерфейс программы	109
3.2.2. Выделение объектов, изменение масштаба просмотра изображения и макет страницы.....	111
3.2.3. Использование линейек, сетки и объектной привязки	113
3.3. Создание простых векторных объектов	114
3.3.1. Прямоугольник и эллипс	115
3.3.2. Многоугольник, звезда, сложная звезда, разлинованная бумага и спираль	116
3.3.3. Стандартные фигуры.....	118
3.4. Свободная форма, прямая через две точки, кривая Безье, художественное оформление, перо, В-сплайн, ломаная линия, кривая через три точки	119
3.5. Работа с текстом в программе CorelDRAW.....	123
3.6. Редактирование объектов, контура и заливки	125
3.6.1. Редактирование объектов	125
3.6.2. Редактирование контура и заливки.....	129
3.7. Работа с растровыми изображениями в программе CorelDRAW	132
3.8. Применение эффектов	134
3.8.1. Инструмент Перетекание	135
3.8.2. Инструмент Контур.....	136
3.8.3. Инструменты Искажение, Оболочка и Вытянуть	136
3.8.4. Инструменты Тень	137
3.9. Некоторые приемы по созданию иллюстраций	138
Глава 4. Использование системы Mathcad в инженерных расчетах	141
4.1. Решения простейших задач средствами Mathcad	141
4.1.1. Основные правила задания и решения математических задач в системе Mathcad.....	141
4.1.2. Определение комплексных чисел	148
4.1.3. Переменные диапазона	149
4.1.4. Использование векторов, матриц и таблиц	150
4.2. Построение графиков в системе Mathcad	153
4.3. Численные методы решения уравнений и систем уравнений	155
4.3.1. Решение систем линейных уравнений	155
4.3.2. Решение нелинейных уравнений	156
4.3.3. Решение систем нелинейных уравнений	158
4.4. Программирование в системе Mathcad	160

4.5. Символьные вычисления в системе Mathcad	166
Глава 5. Практикум по применению среды VBA в Microsoft Office	171
5.1. Создание макросов в Microsoft Office	171
5.2. Разработка программ в среде VBA для Word	173
5.3. Разработка программ в среде VBA для Excel	183
Глава 6. Практикум по применению программы CorelDraw для создания иллюстраций	184
6.1. Создание простых векторных объектов в CorelDRAW	184
6.2. Редактирование объектов, контура и заливки в CorelDRAW	185
Глава 7. Практикум по использованию системы Mathcad в инженерных расчетах	187
7.1. Решение простейших задач и построение графиков средствами Mathcad	187
7.2. Численные методы решения уравнений и систем уравнений. Программирование и символьные вычисления в системе Mathcad	194
Заключение.....	203
Список литературы.....	204
Приложение А. Справочная информация об объектах VBA.....	205
Приложение Б. Индивидуальные задания по CorelDRAW № 1	209
Приложение В. Индивидуальные задания по CorelDRAW № 2.....	212
Приложение Г. Индивидуальные задания по CorelDRAW № 3.....	213
Приложение Д. Методические указания к выполнению курсовой работы по дисциплине Компьютерные технологии (часть 1)	218

Учебное издание

**Малеев Руслан Алексеевич,
Регеда Владимир Викторович,
Регеда Ольга Николаевна**

Компьютерные технологии

Редактор *Т. В. Веденеева*
Компьютерная верстка *М. Б. Жучковой*

Подписано в печать 09.06.2014.
Формат 60×84¹/₁₆. Усл. печ. л. 13,37.
Тираж 150. Заказ № 547.

Издательство ПГУ.
440026, Пенза, Красная, 40.
Тел./факс: (8412) 56-47-33; e-mail: iic@pnzgu.ru