САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Н. Н. Дмитриев, В. Ю. Сахаров

ВВЕДЕНИЕ В ИНФОРМАТИКУ

Учебное пособие

Санкт-Петербург
ВВМ
2008

ББК 32.81
Д53

Печатается по решению Ученого Совета
Математико-механического факультета
Санкт-Петербургского государственного университета

Рецензенты:

Профессор Санкт-Петербургского государственного университета
О. Н. Граничин

Доцент Балтийского государственного технического университета
М. Н. Охочинский

Дмитриев Н.Н., Сахаров В.Ю.

Д53 Введение в информатику.: Учеб. пособие. — СПб.: BBM, 2008. — 156 с.

Учебное пособие посвящено изучению основ Microsoft Office, VBA и Turbo Delphi.

Предназначено для студентов гуманитарных вузов.

ББК 32.81

ISBN 978-5-9651-0293-8

Введение

Данное учебное пособие предназначено для студентов вузов, которым в ходе обучения и в дальнейшей деятельности предстоит оформлять официальные документы, курсовые работы, рефераты, производить несложные расчеты и т.п. Использование данного пособия подразумевает непосредственный контакт с преподавателем на практических занятиях, на которых разъясняются различные действия при работе с MS Office и проверяется выполнение студентами контрольных мероприятий.

Для углубленного изучения возможностей Microsoft Word, Microsoft Excel, Microsoft Access и Visual Basic for Application следует обратиться к специализированным книгам, например, к таким как

- *Власенко С.Ю.* Microsoft Word 2002. – СПб.: БХВ – Петербург, 2002. – 992 с.
- *Долженков В.А., Колесников Ю.В.* Microsoft Excel 2002. – СПб.: БХВ – Петербург, 2002. – 1072 с.
- *Евсеев Г., Мураховский В., Симонович С.* Новейший самоучитель работы на компьютере. Эффективный курс – Москва: «ТехБук», 2004. — 688 с.
- *Кузьменко В. Г.* VBA 2002., М. 2002.
- *Хэлворсон М., Янг М.* Эффективная работа: Office XP. СПб., 2003.

Кроме того, в конце пособия приведена литература, которую следует изучить для понимания взаимодействия программ, входящих в пакет MS Office.

В 2007 году корпорация Microsoft выпустила Windows Vista и Office 2007. Это не должно смущать читателя данного пособия. Описанные ниже действия поддерживаются всеми версиями программ.

В пособии сознательно практически отсутствуют примеры, так как они резко увеличивают объем издания. Примеры приводятся студентам на практических занятиях в электронном виде и при выдаче индивидуальных заданий. Кроме того, студенту важно иметь под рукой некоторое краткое наставление для самостоятельной работы, которое отличается от книг «для начинающих».

Основные понятия

(вводные замечания)

При работе на персональном компьютере (ПК) пользователь должен сам справляться со всеми задачами, возникающими при непосредственном решении профессиональных задач и обслуживании ПК. Поэтому минимальный объем знаний по эксплуатации должен быть у любого человека, использующего современную технику.

Аппаратные средства (hardware) – это совокупность всех устройств, которые составляют ПК или могут быть к нему добавлены. К важнейшим составляющим персонального компьютера относятся: процессорный блок, клавиатура, монитор, принтер, мышь, сканер, накопитель на лазерном диске CD-ROM (в том числе пишущий и DVD), модем, сетевая плата, звуковая карта, web-камера, TV-тюнер и другие устройства, которые могут появиться со временем.

Под процессорным блоком в наиболее узком смысле понимают сам процессор (CPU – Central Processing Unit – центральный обрабатывающий блок). Развитие технологий и архитектуры процессоров позволяют решать все более сложные задачи. Первый персональный компьютер IBM PC (1981 год) имел тактовую частоту 4,77 МГц. В 2004 году интеловский процессор Prescott достиг рубежа в 4 ГГц. Но следует подчеркнуть, что не все решает частота процессора. Многое зависит от архитектуры, т.е. от того, как выполняются команды внутри процессора. Справедливость этого замечания подтверждают процессоры фирмы AMD, которые при меньшей тактовой частоте в начале 2006 года некоторые задачи решали быстрее, чем процессоры фирмы Intel. Но в четвертом квартале 2006 года на прилавках магазинов появились процессоры Intel Core Duo и затем Core 2 Duo, которые очень быстро решают задачи при частоте процессора от 1.6 МГц. Сейчас AMD стре-

мится ликвидировать «отставание» от Intel. Кроме того, начали появляться в продаже четырехядерные процессоры (Intel, AMD), а также объявлено о трехядерных процессорах (AMD).

С другой стороны, процессор сам по себе решить задачи не может. Он должен функционировать совместно с другими компонентами ПК. Этому взаимодействию помогает набор микросхем (Chipset), который обеспечивает обмен данными CPU и других устройств. В 2004 году для процессора Prescott (P4) chipset от фирмы Intel назывался 915 и 925, сейчас для Core 2 Duo выпущен chipset 965.

Аппаратные средства сами по себе человеку ничего не дают. Hardware становится интеллектуальным инструментом только при наличии программного обеспечения (Software), под которым понимают совокупность всех программ и служебных данных, предназначенных для управления ПК.

При включении питания ПК происходит загрузка операционной системы. В подавляющем большинстве в нашей стране в настоящее время персональные компьютеры находятся под управлением операционной системы Windows – разработки фирмы Microsoft. Другая операционная система Linux пока не получила сопоставимого распространения. Но в связи с тем, что Linux распространяется бесплатно и имеет открытый код, у этой системы имеются все предпосылки получить достаточно большое применение. В некоторых европейских странах государственные учреждения переходят именно на операционную систему Linux. Все современные программы и операционные системы требуют достаточно нового оборудования. Поэтому нужно весьма аккуратно подходить к подбору hardware и software. Кроме того, следует постоянно обновлять системное программное обеспечение, так как операционная система очень сложна и в ней могут быть различные недочеты, которые фирма-изготовитель исправляет, улучшая исходный код и оптимизируя для более корректной и быстрой работы, исключающей различные неожиданности.

Часть I.

Введение в Microsoft Word

Современные требования к специалистам с высшим образованием подразумевают умение ими печатать документы, не прибегая к помощи других людей.

Существуют программы, которые позволяют создавать очень сложные, на профессиональном уровне печатные издания. К таким программам, несомненно, относится пакет TeX, созданный выдающимся специалистом в области информатики Дональдом Кнутом (Donald E. Knuth). Пакет TeX был изначально предназначен для подготовки к печати математических текстов, и никакая другая издательская система до сих пор не может сравниться с TeX в полиграфическом качестве текстов с математическими формулами.

С другой стороны, не для каждого документа или публикации необходима возможность использования математических, физических, химических или других символов. Поэтому при наборе текстов с малым количеством тех или иных формул целесообразно пользоваться программой Microsoft Word, которая также позволяет создавать профессионально оформленные документы и публикации. Кроме того, Word является частью MS Office, что позволяет взаимодействовать документам, созданным в Word, с другими компонентами MS Office, а также обмениваться информацией и выполнять коллективную обработку данных.

Начало работы с Microsoft Word

Запуск Word можно осуществить следующими способами:

- в командной строке (**Пуск → Командная строка**) набрать Winword.exe и нажать клавишу <Enter>,

- дважды щелкнуть на ярлыке Word, который может располагаться на рабочем столе или **Пуск → Программы → Microsoft Office → Microsoft Office Word**,
- в проводнике найти документ, созданный с помощью Word (файл с расширением *doc*), и дважды щелкнуть на имени этого документа или нажать клавишу <Enter> (Word запускается для открытия выбранного документа).

Существуют и другие способы запуска, изучаемой нами программы.

Завершение работы в Word осуществляется следующими способами:

- **Файл → Выход** или нажать в верхнем правом углу кнопку **Закрыть**,
- Нажать комбинацию клавиш <Alt>+<F4>.

Если в документе были сделаны изменения, то появится запрос, сохранить ли эти изменения, на который нужно ответить нажатием на кнопку «Да» или «Нет». В случае первого сохранения документа или при сохранении его под другим именем следует выполнит действия **Файл → Сохранить как →** выбрать место, где файл должен быть сохранен **→** задать имя файла **→ Сохранить**.

Выполняя действия, указанные выше, следует обратить внимание на то, что взаимодействие пользователя с программой происходит с помощью интерфейсных конструкций:

- окон (окна приложения, окна документа, диалогового окна, окна формы);
- панелей инструментов, содержащих различные кнопки;
- меню (раскрывающегося по щелчку или всплывающего), состоящего из команд.

Для успешной работы в Word, как и с любой другой программой, нужна постоянная практика, в ходе которой приобретаются навыки и все действия становятся автоматическими и интуитивными.

Единица хранения данных в операционной системе называется файлом. Файл с точки зрения ОС – неделимый объект, который обладает определенными свойствами. Пользователь чаще всего обращает внимание на имя файла и время его создания. Имя файла состоит из двух частей, разделенных точкой. Правая часть называется типом файла (расширением). Файлы Word имеют расширение *doc (в Word 2007 docx)*. Левая часть – имя файла может содержать до 255 символов, включая пробелы. В имени файла можно использовать латинские буквы от A до Z и русские буквы от А до Я (строчные и прописные) и цифры от 0 до 9, символы переноса (-), подчер-

кивания (_), !, а также @, #, $, %, &, (,), {, }. Важно подчеркнуть, что в имени нельзя использовать символы: \, ?, :, *, , (запятая) , ", <, >, |.

При запуске Word автоматически создается пустой документ, готовый для ввода текста. С другой стороны, Word позволяет создавать документы из встроенных шаблонов, шаблона web-страницы или открыть уже существующий документ. Для открытия существующего документа следует выполнить действия:

Файл → Открыть → в окне открытия документа выделить тип файла, соответствующий документу, который будет редактироваться → открыть папку где он находится → выделить имя файла → нажать кнопку **Открыть**.

Замечание. Окно открытия документа Office можно вызвать комбинацией клавиш <Ctrl> + <O>. Word позволяет открывать файлы, не являющиеся документами Word, а также сохранять их в том же формате (если установлен конвертор).

Выделение текста

Для преобразования текста в другой формат, для его перемещения или для других каких-либо действий над ним текст следует выделить и затем выполнить нужную команду. Выделение можно производить с помощью мыши или клавиатуры.

Для выделения небольшого объема текста следует:

а) указатель мыши установить в начало выделяемого текста и нажать левую кнопку мыши;
б) перетащить указатель мыши до конца выделяемой области, не отпуская кнопку (в случае достижения указателем границы окна автоматически начнется прокрутка);
в) отпустить кнопку мыши.

Текст выделен. Для выделения большего объема (или если недостаточно ловкости при работе с мышью) целесообразно поступить иначе:

а) поместить курсор в начало выделяемой области;
б) нажать и удерживать клавишу <Shift> или нажать кнопку <F8> (режим расширенного выделения, в строке состояния появится индикатор ВДЛ);
в) довести курсор до конца выделяемой области и щелкнуть мышью;
г) клавишу <Shift> отпустить, если была нажата <F8>, то нажать <Esc> для выключения расширенного выделения.

Нужный результат достигнут.

Для отмены выделения нужно щелкнуть мышью в любом месте текста.

Областью выделения называется пустая вертикальная полоса слева от основного текста. В области выделения указатель мыши принимает форму стрелки, направленной вправо вверх.

Представим в виде таблицы действия, позволяющие выделять элементы текста:

Выделяемый объект	**Действие**
Слово	Двойной щелчок по слову
Предложение	<Ctrl> + Одинарный щелчок по предложению
Строка	Одинарный щелчок в области выделения
Несколько строк	Перетаскивание указателя мыши по области выделения в соответствующих строчках при нажатой левой кнопке
Абзац	Двойной щелчок в области выделения соответствующего абзаца
Документ	<Ctrl> + Одинарный щелчок в области выделения или тройной щелчок в области выделения
Прямоугольный блок текста	<Alt> + перетаскивание указателя мыши из левого верхнего угла выделяемого блока в правый нижний угол

Для тех, кто не хочет отрывать рук от клавиатуры, полезно знать следующие комбинации:

Выделяемый объект	**Действия**
Слово	<Shift> + <Ctrl> + <←> или <Shift> + <Ctrl> + <→>
От точки вставки до начала строки	<Shift> + <Home>
От точки вставки до конца строки	<Shift> + <End>
Строка	<Shift> + <↓> или <Shift> + <↑>
От точки вставки до начала документа	<Shift> + <Ctrl> + <Home>
От точки вставки до конца документа	<Shift> + <Ctrl> + <End>
Посимвольное выделение	<Shift> + <→> или <Shift> + <←>

После выделения текста его можно переформатировать, удалить, заменить и многое другое, о чем будет сказано далее. Отметим здесь, что удаление выделенного текста происходит при нажатии клавиши <Delete> или <Backspace>. Это действие важно в силу того, что окончательный текст хорошего документа получается после нескольких корректировок.

Перемещение и копирование

Для перемещения текста или изображения нужно выполнить:

а) Выделить нужный текст или графику.

б) **Правка → Вырезать** или <Ctrl> + <X>, или щелкнуть правой кнопкой мыши и выбрать команду **Вырезать**. При этом выделенный фрагмент удаляется.

в) Переместить курсор в точку вставки.

г) **Правка → Вставить** или <Ctrl> + <V>, или вызвать правой кнопкой мыши контекстное меню и выбрать команду **Вставить**. Фрагмент появится в нужном месте.

При вставке фрагмента в несколько мест следует выполнить пункты в) и г) инструкции по перемещению текста или изображения.

Копирование объектов происходит аналогично их перемещению. Исключение состоит в замене команды **Вырезать** на команду **Копировать** и <Ctrl> + <X> на <Ctrl> + <C>.

Отметим, что на панели инструментов есть кнопки **Вырезать**, **Копировать**, **Вставить**. Естественно, их тоже можно использовать.

В Microsoft Office в версиях 2000 и XP реализована возможность копирования до 24 фрагментов текста, графики и других объектов в *Буфер обмена*. Из *Буфера обмена* объекты могут вставляться в различные документы.

Перенос слов

В Word текст может быть выровнен по ширине, по центру, прижат к левому или правому краю. Если не происходит выравнивание по центру или выравнивания по левому (правому) краю, то отсутствие переносов в словах приводит к увеличению расстояний между словами в абзацах. Эти расстояния в соседних строчках чаще всего будут различными, что обычно приводит к плохому виду документа в целом.

Для автоматизации расстановки переносов надо провести настройку программы:

Сервис → Язык → Расстановка переносов → установить флажок **Автоматическая расстановка переносов → OK**.

Форматирование текста

Форматирование текста состоит в возможности по желанию пользователя выбрать вид символов, из которых состоит документ, форму абзацев и страницы в целом.

Форматирование символов

Символ – это отдельная буква, знак пунктуации или специальный знак, например, ®, €, ™.

Форматом символа называют параметры его начертания – шрифт, его атрибуты (курсив, полужирный, подчеркнутый, верхний или нижний регистр), цвет и др.

Шрифт – это способ начертания символов. Символы, относящиеся к конкретному шрифту, имеют характерный вид. Размер шрифта измеряется в пунктах (1 пункт равен 1/72 дюйма). Текст в книгах обычно равен 10 – 12 пунктам.

Стиль – это определенный набор команд форматирования. По умолчанию Word использует стиль «Обычный» (Normal).

Форматирование символов текста можно произвести с помощью панели инструментов

Если этой панели на экране нет, то нужно сделать следующее: **Вид → Панели инструментов →** установить флажок около элемента **Форматирование**.

Для изменения стиля, шрифта, размера шрифта и т.п. необходимо выделить нужный фрагмент и выбрать действие на панели **Форматирование** с помощью мыши.

Для использования большего числа параметров настройки следует воспользоваться помощью команд меню:

Выделить текст → **Формат** → **Шрифт** → выбрать нужное действие → **ОК**.

Замечание. Диалоговое окно **Шрифт** также вызывается комбинацией клавиш <Ctrl> + <D>, а перемещение внутри него может осуществляться с помощью клавиш перемещения курсора и <Tab>, <Enter> и <Пробел>.

Приведем комбинации клавиш, используемые для форматирования символов:

Формат	**Комбинация клавиш**
Полужирный	<Ctrl> + <B>
Курсив	<Ctrl> + <I>
Обычное подчеркивание	<Ctrl> + <U>
Подчеркивание только слов	<Ctrl> + <Shift> +<W>
Двойное подчеркивание	<Ctrl> + <Shift> + <D>
Малые прописные	<Ctrl> + <Shift> + <K>
Все прописные	<Ctrl> + <Shift> + <A>
Скрытый текст	<Ctrl> + <Shift> + <H>
Верхний индекс	<Ctrl> + <Shift> + <=>
Нижний индекс	<Ctrl> + <=>
Копирование формата	<Ctrl> + <Shift> + <C>
Вставка формата	<Ctrl> + <Shift> + <V>
Удаление формата	<Ctrl> + <Пробел>
Изменение регистра символов	<Shift> + <F3>
Шрифт	<Ctrl> + <Shift> + <F>
Шрифт symbol	<Ctrl> + <Shift> + <Q>
Размер в пунктах	<Ctrl> + <Shift> + <P>
Следующий по возрастанию размер в пунктах, доступный для данного шрифта	<Ctrl> + <Shift> + <>>
Предыдущий размер в пунктах, доступный для данного шрифта	<Ctrl> + <Shift> + <<>
Размер на один пункт больше	<Ctrl> + <]>
Размер на один пункт меньше	<Ctrl> + <[>

Форматирование абзацев

В Word абзацем называется фрагмент текста, заканчивающийся маркером конца абзаца ¶, который вводится нажатием клавиши <Enter>. Чтобы этот символ постоянно выводился на мониторе, нужно установить флажок в пункте знаки абзацев:

Сервис → Параметры → вкладка **Вид →** в группе **Знаки форматирования** установить флажок **Знаки абзацев → ОК**.

С другой стороны, этот маркер чаще нужен при окончательном форматировании документа. Поэтому для отображения или выключения символов абзаца следует нажать <Ctrl> + <Shift> + <8> или кнопку на панели инструментов **Непечатаемые знаки** (¶).

Наибольший выбор параметров форматирования абзацев предоставляет диалоговое окно **Абзац**, вызываемое из меню: **Формат → Абзац**. Выравнивание, изменение межстрочных интервалов и интервалов между абзацами, изменение отступов производится в выделенных абзацах по схеме:

Формат → Абзац → установка параметров на вкладке **Отступы и интервалы → ОК**.

Аналогично можно поступить с параметрами на вкладке **Положение на странице**.

Некоторые команды форматирования можно произвести с помощью клавиатуры. Для этого нужно выделить нужный абзац, после чего нажать комбинацию клавиш, приведенных в следующей таблице:

Действие	Комбинация клавиш
Выравнивание текста по левому краю	<Ctrl>+<L>
Центрирование текста	<Ctrl>+<E>
Выравнивание текста по правому краю	<Ctrl>+<R>
Выравнивание текста по ширине	<Ctrl>+<J>
Добавление отступа от левого края	<Ctrl>+<M>
Создание выступа на одну позицию табуляции	<Ctrl>+<T>
Сокращение выступа на одну позицию табуляции	<Ctrl>+<Shift>+<T>
Текст с одинарным интервалом	<Ctrl>+<1>
Переход к интервалу в 1,5 строки	<Ctrl>+<5>
Текст с двойным интервалом	<Ctrl>+<2>
Добавление/удаление 12 пунктов	<Ctrl>+<0>(ноль)

перед абзацем	
Удаление форматирования, не являющегося компонентом стиля, наложенного на абзац	<Ctrl>+<Q>
Восстановление форматирования по умолчанию (из стиля «Обычный»)	<Ctrl>+<Shift>+<N>

Форматирование страниц

Для установки параметров страницы набранного текста необходимо выделить текст или поместить точку вставки в раздел (о том, что такое «раздел» см. ниже), для которого задаются параметры, после чего выполнить процедуру:

Файл → Параметры страницы → на вкладке **Размер бумаги** установить необходимые данные **→** на вкладке **Поля** выбрать ориентацию бумаги **→ ОК**.

В приведенной процедуре упомянута вкладка **Поля**, которая позволяет установить еще одну важную характеристику страницы – *поле*. Под *полем* понимается область между краем листа бумаги и основным текстом. В полях могут располагаться колонтитулы, номера страниц, сноски, какой-либо текст и даже рисунки. Размеры полей устанавливаются в счетчиках **Верхнее, Нижнее, Левое, Правое**. Здесь же задается положение переплета (пространство, выделенное для переплета, не влияет на размер полей, но сокращает область основного текста).

Вид страницы зависит от того, как должен быть представлен *разворот*, т.е. пара страниц (левая и правая), которые видны при открытии книги, журнала, курсовой или дипломной работы и т.п. В случае печати документа на двух сторонах бумаги следует правильно установить параметры оформления разворотов. Это достигается с помощью выбора элемента **Зеркальные поля** либо элемента **2 страницы на листе** в раскрывающемся списке **Несколько страниц** на вкладке **Поля**. В первом случае разворот реализуется на двух различных листах, во втором случае – на одном (элемент 2 страницы на листе целесообразно выбирать, если ориентация листа альбомная).

Положение абзаца на странице

Word автоматически разрывает страницу при достижении границы нижнего поля, и текст продолжается на следующей странице. Такой процесс не всегда приводит к хорошему виду документа. Поэтому в Word имеется средство управления разрывами абзаца:

Поместить точку вставки внутрь абзаца, для которого надо задать параметры разрыва → **Формат** → **Абзац** → на вкладке **Положение на странице** установить нужные флажки → **OK**.

Флажок **Запрет висячих строк** означает, что для выделенных абзацев не допускается появление одиночной строки внизу или вверху страницы; **Не разрывать абзац** – для выделенных абзацев не допускается разрыв страницы внутри абзаца, т.е. если абзац не помещается целиком на текущей странице, то он перемещается на следующую страницу; **Не отрывать от следующего** – выделенный абзац всегда находится на одной странице со следующем абзацем; **С новой страницы** – выделенный абзац помещается наверху следующей страницы.

Принудительный разрыв страницы производится по схеме:

Поместить точку вставки в положение, где должна находиться новая страница → **Вставка** → **Разрыв** → в окне **Разрыв** в группе **Начать** выбрать параметр **новую страницу** → **OK**.

Разрыв страницы получается также при нажатии комбинации клавиш <Ctrl> + <Enter>.

Разделы

Текст документа часто делится на части, которые могут форматироваться независимо друг от друга. Такие части документа называются *разделами*.

Точка разрыва раздела вставляется следующими действиями:

Поместить точку вставки в то место, где должен начаться новый раздел → **Вставка** → **Разрыв** → в диалоговом окне **Разрыв** в группе **Новый раздел** отметить нужный переключатель → **OK**.

Следует отметить, что символ разрыва раздела хранит всю информацию о форматировании раздела. Для удаления разрыва раздела нужно поместить курсор перед разрывом раздела и нажать клавишу <Delete>.

Деление документа на разделы следует делать, как будет показано дальше, для различного форматирования этих разделов, например для вставки разных колонтитулов на страницах, для набора текста в несколько

колонок и т.п. Кроме того, оно весьма полезно при сложной структуре документа и/или большом количестве глав.

Колонтитулы

Колонтитулы – это информация, представленная на каждой странице документа. Чаще всего верхний колонтитул располагается в верхнем поле, а нижний – в нижнем поле. По умолчанию верхний колонтитул располагается на расстоянии 1,25 см от края бумаги. Для изменения этого параметра выполняется следующая процедура:

Вид → Колонтитулы → перейти к колонтитулу нужного раздела **→ Файл → Параметры страницы →** войти на вкладку **Источник бумаги →** в группе **От края** установить желаемые значения до верхнего колонтитула и до нижнего колонтитула **→** нажать **ОК** в окне **Параметры страницы →** нажать кнопку **Закрыть** для возвращения к основному тексту документа.

Для вставки текста в колонтитул или для его корректировки производятся следующие действия:

Вид → Колонтитулы → ввести и отформатировать верхний или нижний колонтитул (переход от верхнего колонтитула к нижнему колонтитулу производится с помощью кнопки **верхний/нижний колонтитул**) **→** нажать кнопку **Закрыть** на панели инструментов **Колонтитулы**.

По умолчанию в документе все колонтитулы одинаковы. Для того чтобы разные части документа имели свои колонтитулы, его следует разделить на разделы. Процедура вставки колонтитула в разделе имеет вид:

Поместить точку вставки в нужном разделе **→ Вид → Колонтитулы →** в открывшемся верхнем колонтитуле произвести корректировку текста (если нужен нижний колонтитул, следует нажать кнопку **Верхний/Нижний колонтитул**) **→** для отмены связи колонтитула с предыдущим разделом надо отжать кнопку **Как в предыдущем →** набрать новый колонтитул **→** нажать кнопку **Закрыть**.

Еще раз подчеркнем, что если не разрывать связь с другими колонтитулами, то изменяются колонтитулы во всем документе.

Разные колонтитулы для четных и нечетных страниц страницы задаются следующим образом:

Вид → Колонтитулы → перейти к колонтитулу нужного раздела (можно использовать соответствующие кнопки) **→ Файл → Параметры страницы → Источник бумаги →** установить флажок в группе **Различать колонтитулы** напротив четных и нечетных страниц **→** после нажатия кнопки **ОК** перейти к верхнему или нижнему колонтитулу нечетной страницы **→** кнопка **Перейти к следующему** позволяет приступить к редактированию колонтитула четной страницы **→** нажатие кнопки **Закрыть** возвращает к редактированию основного текста.

Нумерация страниц

Нумерацию страниц можно осуществить, используя диалоговое окно **Номера страниц**:

Вставка → Номера страниц → в открывшемся диалоговом окне установить нужные значения **→ OK**.

Формат номера страницы изменяется следующей процедурой:

Вставка → Номера страниц → нажать кнопку **Формат →** в диалоговом окне **Формат** номера страницы определить желаемый вид нумерации **→ OK**.

Таблицы

Таблица – это одна из форм представления данных, имеющих одинаковую структуру. Состоит таблица из строк и столбцов, на пересечении которых располагаются информационные ячейки.

Создание таблицы

Для вставки таблицы в документ требуется выполнить следующие действия:

Установить точку вставки в предполагаемом месте расположения таблицы **→ Таблица → Вставить → Таблица →** в диалоговом окне **Вставка таблицы** установить нужные параметры (число строк и столбцов, ширину столбцов) **→ OK**.

Если таблица имеет сложную структуру, то ее целесообразно нарисовать:

Таблица → Нарисовать таблицу → в появившейся панели инструментов **Таблицы и границы** нажать кнопку **Нарисовать таблицу** (указатель мыши примет форму пера) **→** нажать левую кнопку мыши в точке, где должен располагаться один из углов таблицы, и, не отпуская ее, переместить мышь так, чтобы получить внешнюю границу таблицы, отпустить левую кнопку мыши **→** используя мышь и ее левую кнопку, нарисовать внутренние линии таблицы, проходящие от одной разделительной линии до другой.

Таким образом, может быть создана таблица, имеющая весьма сложную структуру. Следует отметить, что на панели инструментов **Таблицы и границы** имеются кнопки, применяя которые, нарисовать таблицу можно более эффективно, а результат становится более изящным.

В ходе редактирования документа может оказаться, что созданная ранее таблица не нужна. Для ее удаления следует поместить точку вставки в любую из ее ячеек и выполнить действия:

Таблица → Удалить → Таблица.

Для добавления строки в конце таблицы следует поместить точку вставки в конец последней ячейки и нажать клавишу <Tab>. Для вставки строки в середине таблицы нужно поместить точку вставки в конец последней ячейки строки, после которой должна находиться новая строка, и последовательно нажать клавиши <→>, <Enter>.

Если нужно добавить столбцы или строки, то можно поступить следующим образом:

Выделить область или, в случае добавления одного столбца (строки), поместить курсор в ячейку, относительно которых будет происходить вставка → **Таблица** → **Вставить** → в появившемся меню выбрать нужный пункт.

Для удаления строки или столбца можно применить следующую инструкцию:

Выделить область, подлежащую удалению → **Таблица** → **Удалить** → в появившемся меню выбрать пункт **Столбцы** или **Строки**.

Ввод данных в таблицу

Содержимое ячеек таблицы может быть как текстовым, так и графическим. Текст вводится с клавиатуры, при этом высота и ширина ячейки может автоматически меняться. Ячейка для ввода информации может быть выбрана щелчком левой кнопки мыши. Но более эффективным способом перемещения по таблице является применение клавиатуры:

Комбинация клавиш	Описание
<Tab>	Перейти в соседнюю ячейку. Если точка вставки находится в последней ячейке таблицы, то добавляется новая строка.
<Shift>+<Tab>	Перейти в предыдущую ячейку.
<↑> или <↓>	Перейти в предыдущую или следующую строку.
<Alt>+<Home>	Перейти в первую ячейку в строке
<Alt>+<End>	Перейти в последнюю ячейку в строке
<Alt>+<Page up>	Перейти в первую ячейку в столбце
<Alt>+<Page Down>	Перейти в последнюю ячейку в столбце
<Enter> в начале первой ячейки	Добавить текст перед таблицей в начале документа

Расположение текста в ячейке

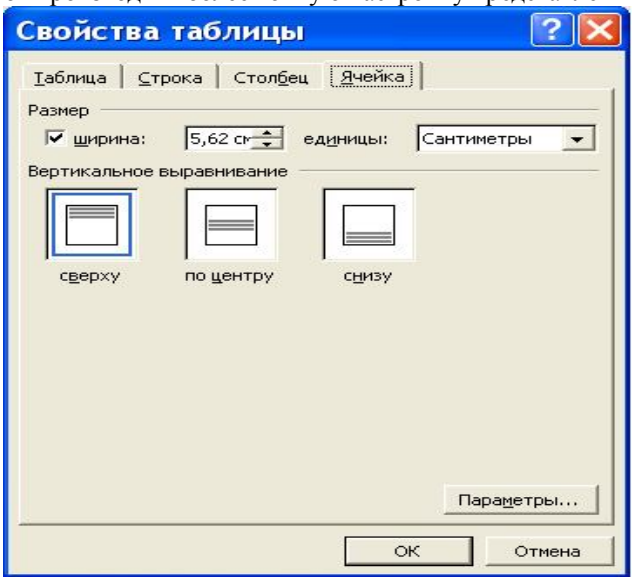
Часто при оформлении таблиц текст нужно расположить не горизонтально, а перпендикулярно к основанию таблицы. Делается это следующим образом:

Выделить ячейки, в которых нужно изменить ориентацию содержимого → **Формат → Направление текста** → в диалоговом окне **Направление текста** выбрать нужную ориентацию → **ОК**.

Полезные параметры таблицы в целом и ячеек в частности определяются в диалоговом окне **Свойства таблицы**, вызов которого происходит так:

Выделить группу ячеек таблицы → **Таблица → Свойства таблицы** → установить параметры на соответствующих вкладках → **ОК**.

Вид диалогового окна **Свойства таблицы** представлен ниже. В этом окне имеются вкладки **Таблица**, **Строка**, **Столбец**, **Ячейка**, которые позволяют производить более тонкую настройку представления таблицы:



При редактировании таблицы иногда появляется необходимость объединить ячейки:

Выделить объединяемую группу ячеек → **Таблица → Объединить ячейки**.

Почти аналогично происходит разбиение ячеек. Отличие заключается в необходимости установить в диалоговом окне Разбиение ячеек число строк и столбцов, на которые разбивается ячейка.

Умение получать хороший вид таблицы может прийти только при настойчивой и достаточной практике. Word позволяет настраивать гигантское количество параметров, разукрашивать таблицу в разные цвета, вставлять формулы и т.п. Экспериментируйте и достигнете успеха!

Слияние документов

При большой переписке (например, рассылка приглашений на конференцию 250 участникам) часто возникает потребность напечатать однотипные документы, которые различаются лишь конечным числом похожих позиций. Поэтому целесообразно каким-либо способом организовать базу данных, содержащую структурированную информацию и документ, в который будут вноситься данные из этой базы. В документе специальным образом организуются *поля слияния*, в которые с помощью некоторой процедуры вносятся *записи данных*.

Обычно базу данных организуют в виде таблицы, столбцы которой являются полями данных, а строки – записями данных. Первая строка таблицы состоит из названий полей данных.

Поля слияния, которые организуются в основном документе, имеют те же имена, что и *поля данных* в источнике (таблице). Процедура слияния заносит в *поля слияния* значения соответствующие *полям данных*.

Для выполнения процедуры **Слияние** необходима панель инструментов **Слияние**, которая вызывается так:

Вид → Панели инструментов → Слияние.

Составим в отдельном документе следующую таблицу:

Организация	Руководитель	Телефон	Факс	Адрес
Кот	Матроскин	12345	77777	деревня Простоквашино, д. 1
Пес	Шарик	76543	88888	деревня Простоквашино, д. 1/2

Напомним, что первая строка этой таблицы — названия полей данных. Файл с таблицей закрываем.

Теперь составим основной документ. Пусть текст его будет таким:

Глубокоуважаемый <Руководитель>!

Сообщаем Вам, что офис Вашей организации <Организация> зарегистрирован по адресу <Адрес>. Напоминаем, что выделенный Вам телефон <Телефон> и факс <Факс> нужно своевременно оплачивать!

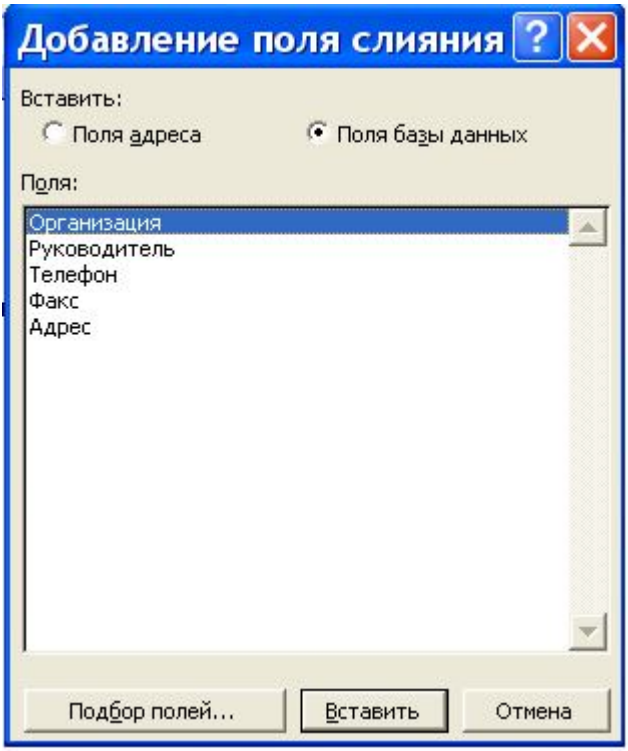
С уважением,

Администратор деревни Простоквашино.

Наша задача — подставить вместо слов, стоящих в треугольных скобках информацию из таблицы, которая находится в другом файле. Последовательность действий для этого будет следующей:

а) На панели инструментов **Слияние** нажать кнопку **Открыть источник данных**. В диалоговом окне **Выбор источника данных** находим файл, содержащий таблицу. Отмечаем его и нажимаем кнопку **Открыть**.

б) Устанавливаем точку вставки в основном документе в то место, куда должны вставить поле слияния. На панели инструментов нажимаем кнопку **Вставить поля слияния**. В открывшемся окне **Добавление поля слияния** выбираем нужное и нажимаем **Вставить**.



в) При необходимости повторяем пункт б) нужное количество раз.
г) Нажатие кнопки **Слияние** в новый документ дает команду Word для вставки всей информации из базы данных, что означает создание документов в количестве равном количеству строк в таблице без учета первой в новом файле.

Word 2002 имеет **Мастер слияния**, который помогает производить вышеописанную процедуру. Но вышеописанные действия следует научиться делать вручную для того, чтобы понять смысл процедуры и принцип ее работы.

Маркированные и нумерованные списки

Список – это совокупность абзацев, отформатированных специальным образом и снабженных номерами или маркерами.

Маркированный список – это список, в котором перед каждым элементом стоит некий символ или рисунок, отмечающий элемент списка.

Нумерованный список – это список, в котором последовательность составляющих имеет смысл и в качестве маркеров стоит порядковое число или символ, порядок которого известен.

Многоуровневые списки – иерархическая структура, сочетающая в себе маркированные и нумерованные списки.

Для создания списка выполняются следующие команды меню: **Формат → Список →** в диалоговом окне **Список** выбрать вкладку **Маркированный**, **Нумерованный** или **Многоуровневый** для создания соответствующего списка **→ OK**.

Настройка того или иного списка производится нажатием кнопки **Изменить**, которая открывает диалоговое окно для детальной настройки списка. Отметим также наличие переключателей **Начать заново** и **Продолжить** на вкладке **Нумерованный**, который позволяет начать заново нумерацию списка или продолжить его соответственно.

После того как закончен ввод одного из пунктов списка, нажатие клавиши <Enter> вводит следующий пункт списка со следующим номером или соответствующим маркером. Преобразование абзацев в список происходит путем их выделения и повторением процедуры для создания списка.

Многоуровневый список подразумевает наличие подчиненных элементов в некоторых абзацах. Для создания таких пунктов следует нажать клавишу <Enter>, что создает элемент текущего списка, и затем клавишу <Tab>, которая повысит уровень вложенности элемента. Для понижения уровня вложенности на единицу следует нажать комбинацию

<Shift> + <Tab>. Следует подчеркнуть, что многоуровневый список может содержать до девяти уровней вложенности.

Для завершения списка следует нажать <Enter>, что приведет к созданию следующего элемента, и затем клавишу <Backspace>, которая уберет маркер.

Ввод математических формул

Математическая формула вставляется в текст посредством специального объекта Microsoft Equation 3.0. Для этого следует выполнить следующие действия:

Расположить точку вставки в месте, где должна быть формула → **Вставка** → **Объект** → в диалоговом окне **Вставка объекта** на вкладке **Создание** выбрать **Тип** объекта **Microsoft Equation 3.0** → **OK**.

После нажатия кнопки OK появится панель инструментов **Формула** и *слот* – поле, в котором записывается математическое выражение. Элементы на панели **Формула** выбираются мышью, при этом в слоте появляются соответствующие символы, из которых при правильной последовательности нажатий появляется осмысленная математическая запись.

Для выхода из математического редактора достаточно щелкнуть мышью в произвольном месте вне формулы или нажать <Esc>.

Использование мыши при выборе шаблонов, символов или расстановке математических акцентов достаточно утомительно. Поэтому наиболее часто используемый ввод целесообразно производить при помощи клавиатуры:

Нажать и отпустить <Ctrl> + <T>, после чего выбрать клавишу из третьего столбца		
Пример	**Шаблон**	**Клавиша**
(a)	Скобки	<(> или <)>
$[b]$	Квадратные скобки	<[> или <]>
$\{C\}$	Фигурные скобки	<{> или <}>
$\lvert D\rvert$	Абсолютная величина	<|>
$\frac{a}{b}$	Дробь	<F>
a/b	Дробь с косой чертой	</>
A^2	Верхний индекс	<H>
B_2	Нижний индекс	<L>

G_1^2	Индексы	<J>
$\sqrt{Q}$	Квадратный корень	<R>
$\sqrt[3]{H}$	Корень *n*-й степени	<N>
$\sum_{i=1}^{N} A_i$	Сумма	<S>
$\prod_{i=9}^{15} Y_i$	Произведение	<P>
$\int_0^9 x^2 dx$	Интеграл	<I>
$\begin{matrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{matrix}$	Матрица 3×3	<M>
$R_{j \to \infty}$	Нижний предел	<U>

Для вставки математических символов применяются следующие комбинации клавиш:

Нажать и отпустить <Ctrl> + <K>, после чего выбрать клавишу из третьего столбца		
Символ	**Описание**	**Клавиша**
∞	Бесконечность	<I>
$\rightarrow$	Стрелка	<A>
∂	Частная производная	<D>
$\leq$	Меньше или равно	<<>
$\geq$	Больше или равно	<>>
$\times$	Произведение	<E>
$\in$	Принадлежит	<E>
$\notin$	Не принадлежит	<Shift> + <E>
$\subset$	Включается	<C>
$\not\subset$	Не включается	<Shift> + <C>

Математическим акцентом называются надстрочные значки вектора, средней величины, производной и т.п. Для их простановки курсор следует поместить справа от элемента, над которым проставляется акцент,

и затем воспользоваться панелью инструментов **Формула** или комбинацией клавиш из таблицы:

Пример	Акцент	Клавиши
$\bar{x}$	Черта	<Ctrl> + <->
$\tilde{y}$	Тильда	<Ctrl> + <~>
$\vec{a}$	Стрелка (вектор)	<Ctrl>+<Alt>+<->
z'	Штрих	<Ctrl>+<Alt>+<'>
z''	Два штриха	<Ctrl> + <">
$\dot{y}$	Точка	<Ctrl>+<Alt>+<.>

Особенности форматирования математических формул

Для хорошего вида формул часто требуется увеличить или уменьшить пробел между символами. Клавиша <Пробел> в Редакторе формул (Microsoft Equation) не работает. На панели инструментов **Формула** в палитре **Пробелы и многоточия** можно выбрать пять видов пробелов. С помощью клавиатуры большинство из них вставляется комбинациями клавиш, приведенными в следующей таблице:

Вид пробела	Комбинация клавиш
Нулевой пробел	<Shift> + <Пробел>
Пробел в 1 пункт	<Ctrl>+<Alt>+<Пробел>
Пробел в 1/6 длинного пробела	<Ctrl>+<Пробел>
Пробел в 1/3 длинного пробела	<Ctrl>+<Shift>+<Пробел>
Длинный пробел	Вставка только с помощью панели инструментов **Редактора формул**

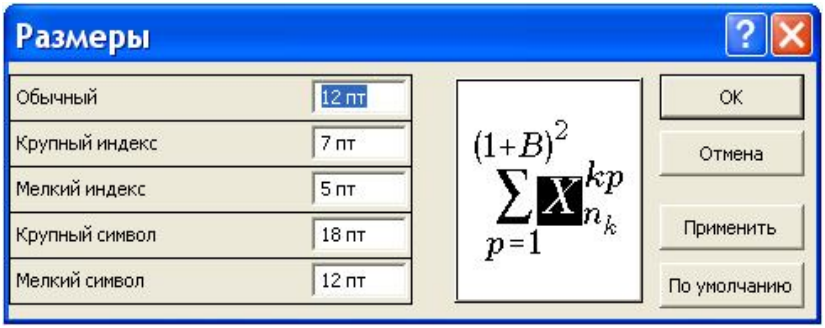
Следующей особенностью рассматриваемого форматирования является выравнивание формул:

- по левому краю;
- по центру;
- по правому краю;
- по знаку равенства;
- по знакам «точка» или «запятая».

Для выравнивания следует в **Редакторе формул** выбрать команду **Формат** и в открывшемся меню выбрать нужный тип выравнивания.

При наборе математических текстов для лучшего вида формул и текста в целом встречается необходимость изменить размеры символов, которые установлены по умолчанию. Делается это в **Редакторе формул** так:

Размер → Определить → в диалоговом окне **Размеры** ввести нужные значения **→ OK**.



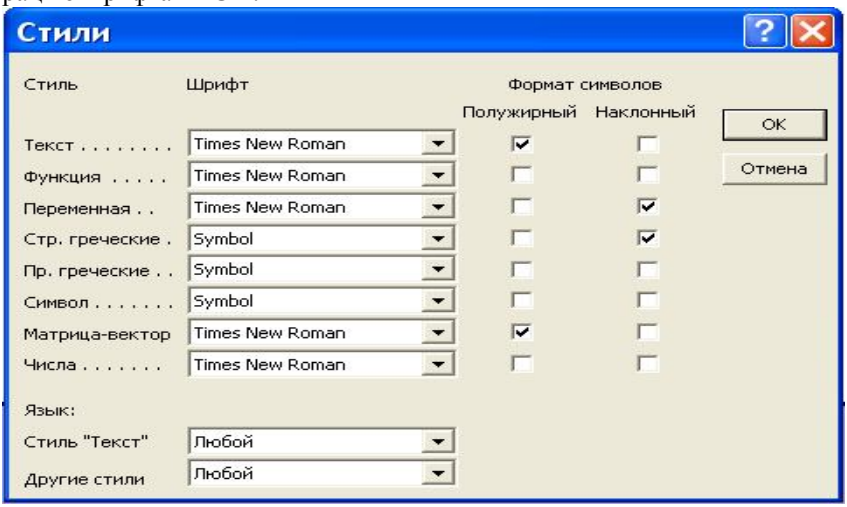
Для возвращения к значениям размеров символов по умолчанию в окне **Размеры** следует нажать кнопку **По умолчанию**.

Следует также подчеркнуть, различные элементы в формуле могут иметь свои собственные стили:

Математический,	Текст,	Функция,
Переменная,	Греческий,	Матрица-вектор.

Настройка этих стилей происходит в окне **Стили**:

Стиль → Определить → в открывающемся окне **Стили** задать конфигурацию шрифта **→ OK**.



Замечание 1. В Word 2007 набор математических текстов изменился. Но если редактируется файл, в котором формулы созданы в Word 97-2003, то все сказанное остается в силе, то есть **Microsoft Equation 3.0**

доступен для корректировки математического текста. Кроме того, **Microsoft Equation 3.0** можно вставить как объект в Word 2007.

Замечание 2. При наборе математических символов в Word 2007 стали доступны команды очень похожие на кодировку TEX (LATEX) . Эти команды сведены в следующую таблицу.

Символ	Команда	Символ	Команда
⊥	\above	∪	\cup
ℵ	\aleph	ℸ	\dalet
α	\alpha	⊣	\dashv
Α	\Alpha	ⅆ	\dd
⨿	\amalg	ⅅ	\Dd
∠	\angle	⃜	\ddddot
≈	\approx	⃛	\dddot
⬆	\asmash	̈	\ddot
∗	\ast	⋱	\ddots
≍	\asymp	°	\degree
┆	\atop	δ	\delta
̄	\bar	Δ	\Delta
▪	\Bar	⋄	\diamond
[	\begin	♢	\diamondsuit
⊤	\below	÷	\div
ℶ	\bet	̇	\dot
β	\beta	≑	\doteq
Β	\Beta	…	\dots
⊥	\bot	↓	\downarrow
⋈	\bowtie	⇓	\Downarrow
▫	\box	⬇	\dsmash
⟨	\bra	ⅇ	\ee

˘	\breve	ℓ	\ell
∙	\bullet	∅	\emptyset
∩	\cap	∎	\end
∛	\cbrt	ϵ	\epsilon
⋅	\cdot	Ε	\Epsilon
⋯	\cdots	▒	\eqarray
ˇ	\check	≡	\equiv
χ	\chi	η	\eta
Χ	\Chi	Η	\Eta
∘	\circ	∃	\exists
⏤	\close	∀	\forall
♣	\clubsuit	⁣	\funcapply
∲	\coint	γ	\gamma
≅	\cong	Γ	\Gamma
≥	\ge	⊂	\Leftarrow
≥	\geq	↽	\leftharpoondown
←	\gets	↼	\leftharpoonup
≫	\gg	↔	\leftrightarrow
ℷ	\gimel	⇔	\Leftrightarrow
̂	\hat	≤	\leq
ℏ	\hbar	⌊	\lfloor
♡	\heartsuit	≪	\ll
↩	\hookleftarrow	↦	\mapsto
↪	\hookrightarrow	▪	\matrix
⟡	\hphantom	∣	\mid
⃐	\hvec	⊨	\models
ⅈ	\ii	∓	\mp

∭	\iiint	μ	\mu
∬	\iint	Μ	\Mu
ℑ	\Im	∇	\nabla
∈	\in	⨟	\naryand
∆	\inc	≠	\ne
∞	\infty	↗	\nearrow
∫	\int	≠	\neq
ι	\iota	∋	\ni
Ι	\Iota	‖	\norm
ⅉ	\jj	ν	\nu
κ	\kappa	Ν	\Nu
Κ	\Kappa	↖	\nwarrow
⟩	\ket	ο	\o
λ	\lambda	Ο	\O
Λ	\Lambda	⊙	\odot
⟨	\langle	∰	\oiiint
{	\lbrace	∯	\oiint
[	\lbrack	∮	\oint
⌈	\lceil	ω	\omega
∕	\ldivide	Ω	\Omega
…	\ldots	⊖	\ominus
≤	\le	⊦	\open
←	\leftarrow	⊕	\oplus
⊗	\otimes	→	\rightarrow
∕	\over	⇒	\Rightarrow
‾	\overbar	⇁	\rightharpoondown
⏞	\overbrace	⇀	\rightharpoonup

⏜	\overparen	∕	\sdivide
∥	\parallel	↘	\searrow
∂	\partial	∖	\setminus
⟡	\phantom	σ	\sigma
ϕ	\phi	Σ	\Sigma
Φ	\Phi	∼	\sim
π	\pi	≃	\simeq
Π	\Pi	∕	\slashedfrac
±	\pm	∎	\smash
⁗	\pppprime	♠	\spadesuit
‴	\ppprime	⊓	\sqcap
″	\pprime	⊔	\sqcup
≺	\prec	√	\sqrt
≼	\preceq	⊑	\sqsubseteq
′	\prime	⊒	\sqsuperseteq
∏	\prod	⋆	\star
∝	\propto	⊂	\subset
ψ	\psi	⊆	\subseteq
Ψ	\Psi	≻	\succ
∜	\qdrt	≽	\succeq
x=(-b±√(b^2-4ac)	\quadratic	∑	\sum
⟩	\rangle	⊃	\superset
∶	\ratio	⊇	\superseteq
}	\rbrace	↙	\swarrow
]	\rbrack	τ	\tau
⌉	\rceil	Τ	\Tau
⋰	\rddots	θ	\theta

ℜ	\Re	Θ	\Theta
▭	\rect	×	\times
⌋	\rfloor	→	\to
ρ	\rho	⊤	\top
Ρ	\Rho	↦	\tvec
̲	\ubar	←	<-
̳	\Ubar	≤	<=
‾	\underbar	→	->
⏟	\underbrace	≥	>=
⏝	\underparen	⇑	\Uparrow
↑	\uparrow	↕	\updownarrow
⇕	\Updownarrow	∨	\vee
⊎	\uplus	|	\vert
υ	\upsilon	‖	\Vert
Υ	\Upsilon	⟡	\vphantom
ϵ	\varepsilon	∧	\wedge
φ	\varphi	℘	\wp
ϖ	\varpi	≀	\wr
ϱ	\varrho	ξ	\xi
ς	\varsigma	Ξ	\Xi
ϑ	\vartheta	ζ	\zeta
|	\vbar	Ζ	\Zeta
⊢	\vdash	(пробел нулевой ширины)	\zwsp
⋮	\vdots	∓	-+
⃗	\vec	±	+-

Сноски

Сноска – это размещенное внизу полосы подстрочное замечание или подстрочная библиографическая ссылка. Сноска отмечается специальным символом, который легко отличается от основного текста.

Word поддерживает два вида сносок: обычные сноски и концевые. *Обычная сноска* располагается на странице, на которой находится символ сноски. *Концевая сноска* размещена в конце раздела или всего документа. Оба типа сносок отделяются от основного текста горизонтальной чертой.

Для создания одной сноски следует:

Установить точку вставки в месте, где должен появиться значок сноски → **Вставка → Ссылка → Сноска →** в диалоговом окне **Сноски** выбрать положение сноски и другие ее характеристики → **Вставить →** в открывшейся **Панели сносок** ввести текст → нажать кнопку **Закрыть** для закрытия панели сносок.

Нумерация сносок может быть своя для каждой страницы, для каждого раздела и сквозная для всего документа. Порядок нумерации устанавливается так:

Вставка → Ссылка → Сноска → в окне **Сноски** в поле **Начать с** установить номер сноски → выбрать параметр **Продолжить**, **В каждом разделе** или **На каждой странице →** нажать кнопку **Применить**.

Для просмотра и редактирования текста сносок обычно используется панель сносок, в которой доступны практически все средства Word. Панель сносок открывается при двойном щелчке на символе сноски.

При оформлении больших документов (например, книг) встречается необходимость сделать ссылку на источник, который уже был упомянут. Такие вставки делаются в окне **Перекрестные ссылки**:

Установить точку вставки в позицию, где устанавливается сноска → **Вставка → Ссылка → Перекрестная ссылка →** в окне **Перекрестные ссылки** в списке **Тип ссылки** выбрать тип нужной ссылки → в списке **Вставить ссылку на** выбрать **Номер сноски** или **Номер концевой сноски →** в списке **Для какой сноски** (**Для какой концевой сноски**) выбрать нужный вариант → сбросить флажки **Вставить как гиперссылку**, **Добавить слово «выше» или «ниже» → Вставить → Закрыть**.

Создание оглавления

Оглавление – это перечень разделов, глав, параграфов произведения с указанием начальной страницы каждого. Оглавление показывает также соотношение заголовков по значимости, их соподчиненность, ориентируя читателя в основных связях и взаимоотношениях составных частей темы. Оглавление может располагаться в конце или в начале книги, дипломной или курсовой работы и т. д.

Приведем наиболее простой способ сборки оглавления, который состоит в обработке абзацев отформатированных соответствующими стилями заголовков. Для присвоения абзацу встроенного стиля заголовка следует:

Поместить курсор в этот абзац → выбрать нужный стиль в списке Стиль на панели инструментов Форматирование или <Ctrl> + <Alt> + <1>, <Ctrl> + <Alt> + <2>, <Ctrl> + <Alt> + <3> для назначения абзацу стиля «Заголовок 1», «Заголовок 2», «Заголовок 3», соответственно.

Непосредственно само оглавление собирается так:

Поместить точку вставки в то место где будет располагаться оглавление → Отключить отображение скрытого текста и кодов полей (для правильной разбивки на страницы: **Сервис → Параметры →** вкладка **Вид →** в группе **Показывать** убрать флажок **коды полей**.) **→ Вставка → Ссылка → Оглавление и указатели →** в диалоговом окне **Оглавление и указатели** раскрыть вкладку **Оглавление →** установить один из форматов оглавления в списке **Форматы →** установить флажки **Показать номера страниц** и **номера страниц по правому краю →** в списке **Заполнитель** установить желаемый заполнитель пространства между заголовком и номером страницы в оглавлении → установить количество уровней заголовка в счетчике **Уровни → OK**.

В ходе редактирования документа могут измениться номера страниц и сами разделы. Поэтому для обновления оглавления нужно выполнить процедуру:

Отключить отображение скрытого текста и кодов полей → поместить точку вставки в оглавление → нажать <F9> или щелкнуть правой кнопкой мыши и выбрать команду **Обновить поле →** в появившемся диалоговом окне **Обновление оглавления** выбрать нужное действие **→ OK**.

Оглавление обновлено. Более гибкое, но более трудоемкое создание оглавления с помощью кодов полей изложено в книге [1].

На этом первое и очень краткое знакомство с программой Word закончим и перейдем к изучению электронных таблиц Excel.

Часть II. Введение в Microsoft Excel

Современная жизнь не может обойтись без тех или иных расчетов. Результаты вычислений обычно оформляются таким образом, чтобы было видно, каким способом они получены. В этой связи особую роль приобретает программа, созданная корпорацией Microsoft – Microsoft Excel. Excel входит в комплект поставки Microsoft Office и предназначен, в первую очередь, для оформления сложных финансовых и экономических расчетов. С другой стороны, при большом желании и умении программировать в VBA, Excel позволяет решать и другие задачи, для которых он изначально не предназначен. Следует подчеркнуть, что Word, Excel, Access и другие программы, входящие в Microsoft Office, могут взаимодействовать друг с другом, что позволяет вести весь документооборот какой-либо организации.

Предварительные замечания

Документ, который создает Excel, называется *рабочей книгой*. Каждая рабочая книга имеет свое уникальное имя и хранится в отдельном файле с расширением xls.

Рабочая книга состоит из *листов*, которые содержат прямоугольные таблицы данных или диаграммы. Каждый лист имеет имя, которое отображается на ярлычке внизу листа. Для изменения имени текущего листа можно выполнить последовательность: **Формат → Лист → Переименовать** (или воспользоваться контекстным меню, нажав правую кнопку мыши, или дважды щелкнуть на ярлычке с названием листа) и ввести желаемое имя.

Рабочий лист состоит из ячеек. Каждая ячейка имеет свой буквенно-цифровой адрес: латинские буквы определяют столбец, цифры –

строку. Например, верхняя левая ячейка имеет адрес A1, ячейка справа от нее – B1, снизу – A2, по диагонали – B2.

В случае, когда нужно указать адрес ячейки (например, в формуле), который не должен меняться при копировании (но не при переносе!) или другом каком-либо действии с ней, следует использовать значок «$»: A9. Такая запись называется *абсолютным адресом*. Если не должен меняться только столбец или только строка (смешанный адрес), то знак «$» ставится перед соответствующим символом: $A9 – для неизменного столбца, A$9 – для неизменной строки. Адрес, не содержащий символа «$», называется *относительным адресом*.

Обращение к ячейке, находящейся на другом рабочем листе, происходит с помощью явного указания имени листа, знака «!» и адреса ячейки на этом листе:

Лист1!D9.

Если ячейка расположена в другой книге, то следует воспользоваться записью:

[Книга15.xls]Лист1!D9,

т.е. название книги заключается в квадратные скобки, пишется имя рабочего листа и ставится восклицательный знак, и затем записывается адрес ячейки.

Примечание. Если в пути адреса есть пробел, то следует производить запись с помощью одинарных кавычек:

'[Книга15.xls]Лист 95'!D9

Выделение ячеек

Будем называть *диапазоном* любой набор ячеек рабочего листа. Чаще всего в роли диапазона выступает прямоугольная область – пересечение нескольких строк и столбцов. Если надо указать адрес прямоугольного диапазона, то указывают адрес левой верхней ячейки, ставят двоеточие «:» и записывают адрес правой нижней ячейки. Адрес непрямоугольного диапазона представляется записью прямоугольных поддиапазонов, которые разделены точкой с запятой «;».

Для какого-либо преобразования электронной таблицы часто следует вначале выделить ячейки и только после этого выполнить ту или иную команду. Наиболее надежный способ выделения прямоугольного диапазона состоит в выделении одной из угловых ячеек этого диапазона, после чего нажимается и удерживается клавиша <Shift> и с помощью клавиш <←>, <→>, <↓> и <↑> достигается нужный результат. Еще одна возможность – нажать клавишу <F8> (перейти в режим расширенного выделения), применив клавиши со стрелками, вы-

делить нужную область, после чего следует опять нажать <F8> для отключения режима расширенного выделения.

При выделении большого диапазона, имеющего сложную структуру, следует выполнить действия:
Правка → Перейти → в диалоговом окне **Переход** в поле **Ссылка** указать диапазон **→ OK**.

Отметим также, что нажатие клавиш <Shift> + <Пробел> выделяет строку, содержащую активную ячейку, <Ctrl> + <Пробел> выделяет столбец с активной ячейкой, <Shift> + <Ctrl> + <Пробел> выделяет весь активный рабочий лист.

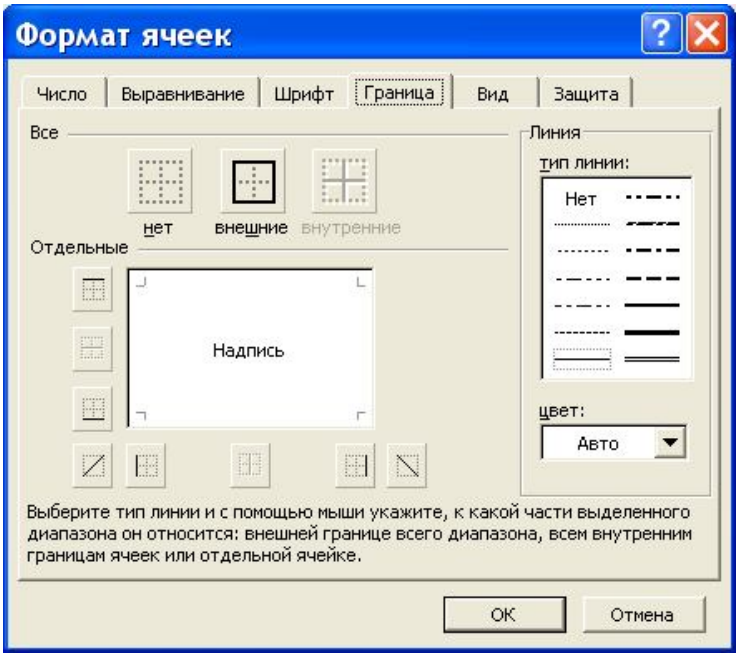
Ввод данных и оформление таблиц

По всей видимости, проблем с вводом данных в ячейку таблицы быть не должно. Хотя следует подчеркнуть, что пользователь должен отдавать отчет, в каком формате вводится или выводится результат в ячейке. При необходимости следует установить проверку данных при вводе: **Данные → Проверка →** в окне **Проверка вводимых значений** перейти на вкладку **Параметры →** выбрать **тип данных** и установить значение, характеризующее вводимую информацию **→** на вкладках **Сообщение для ввода** и **Сообщение об ошибке** сделать необходимые записи **→ OK**.

Данная процедура весьма полезна при создании финансовых отчетов или экономических планов. Какие данные вводятся – таков и результат!

Шрифт для активной ячейки или выделенного диапазона устанавливается выбором значения в поле **Шрифт** на панели инструментов **Форматирование** или процедурой: **Формат → Ячейки →** в диалоговом окне **Формат ячеек** выбрать нужный из списка **Шрифт**. Диалоговое окно **Формат ячеек** хорошо тем, что оно предоставляет большие возможности точной настройки по сравнению с панелью инструментов.

Это же диалоговое окно на вкладке **Выравнивание** предоставляет возможность произвести настройку расположения записи в ячейке. Кроме того, для получения при печати таблицы с разделительными линиями следует на вкладке **Граница** диалогового окна **Формат ячеек** установить нужные линии. Следует помнить, что надо выделять ячейки к которым применяется то или иное действие. Например, для получения особой рамки вокруг таблицы следует выделить все ячейки таблицы, установить тип линии и нажать кнопку внешние.



Эксперименты с вкладкой **Границы** приводят к хорошему виду окончательного документа.

Ввод формул и сообщения об ошибках

Ввод формул в Excel осуществляется непосредственно в ячейку или в строке формул. *Запись формулы начинается со знака «равно»*, после которого следует ее функциональная часть. Ввод формулы завершается нажатием клавиши <Enter> или кнопки с зеленой «галочкой» в строке формул. Режим ввода включается также нажатием функциональной клавиши <F2> или после двойного щелчка по ячейке. Отменить ввод можно нажатием клавиши <Esc> или соответствующей кнопки в строке формул.

Содержательная часть формулы вводится непосредственно с клавиатуры, либо с помощью мастера формул. Например, записи «=A1+B1» (использование клавиатуры) и «=СУММ(A1;B1)» (применили мастер формул) эквивалентны. Подчеркнем, что весьма целесообразно знать, какие имеются встроенные функции и как они работают. Это помогает вводу формул и облегчает многие действия.

В набираемых формулах обычно присутствуют ссылки на другие ячейки рабочего листа. Эти ссылки, как мы уже видели, можно на-

брать вручную с клавиатуры, а можно щелкнуть по соответствующей ячейке, и в формуле появится нужная запись.

Здесь следует напомнить, что существует так называемый *абсолютный адрес* ячейки, который используется для того, чтобы при каком-либо действии с ячейкой, содержащей формулу, данный адрес оставался бы неизменным. Для изменения относительной ссылки на абсолютную или смешанную следует: перейти в режим редактирования формул (например, нажать клавишу <F2>) → установить точку ввода на исправляемую ссылку → нажать несколько раз клавишу <F4> → завершить корректировку нажатием <Enter>.

При необходимости изменить одинаковым образом ссылки во всей формуле или ее части, следует их выделить и нажать нужное количество раз клавишу <F4>.

Встречается ситуация, когда нужно просмотреть формулы, записанные в ячейках, например, для их изменения (по умолчанию Excel отображает результаты вычислений). Для этого следует выполнить процедуру:

Сервис → Параметры → на вкладке **Вид** окна **Параметры** установить флажок **формулы → ОК**,

после чего вместо чисел будут отображаться формулы, в ячейках, которые их содержат. Для быстрого переключения в режим отображения формул и обратно можно воспользоваться комбинацией клавиш <Ctrl> + <`> (обратная кавычка).

Excel может производить вычисления по формулам в соответствии с формулами сразу после их введения (момент проведения расчетов зависит от настроек программы). Если формула введена в ячейку неправильно, то возникает ошибка, о которой Excel сообщает следующими кодами:

######	#ИМЯ?	#ЧИСЛО!
#ЗНАЧ!	#Н/Д	#ПУСТО!
#ДЕЛ/0!	#ССЫЛКА!	

Сообщение вида ###### возникает по следующим причинам:

- результат, вычисленный по формуле, не помещается в ячейке (следует увеличить ширину столбца);
- отрицательное число дней между двумя датами или отрицательная разность между двумя моментами времени (лучшим вариантом исправления может служить изменение формулы и установка контроля на вводимые даты в ячейки).

Сообщение #ЗНАЧ! – следствие использования недопустимого типа аргумента, операнда или неверно записанной формулы:

- для оператора или функции, требующих одного значения, возвращается диапазон;
- неправильно используется функция, аргументом которой является матрица;
- ошибка в макросе;
- после ввода *формулы массива*[1] была нажата клавиша <Enter> вместо <Ctrl> + <Sift> + <Enter>.

Сообщение #ДЕЛ/0! соответствует попытке деления на ноль.

Сообщение об ошибке #ИМЯ? возникает, когда Excel не может найти именованную ячейку или диапазон:

- при наборе имени произошла опечатка;
- текст не был заключен в кавычки;
- в ссылке на диапазон пропущен знак двоеточия.

Появление сообщения #Н/Д соответствует понятию «неопределенные данные»:

- в формуле массива диапазон аргумента или операнда не соответствует диапазону массива;
- не заданы один или несколько аргументов функции;
- используется не описанная в данной книге функция;
- ошибка в макросе.

Ошибка вида #ССЫЛКА! соответствует неправильно введенному адресу (ссылка на несуществующую ячейку может образоваться при копировании ячейки, ссылающейся на ячейку, находящуюся левее, влево или ячейки, ссылающейся на ячейку, находящуюся выше, вверх).

Сообщение #ЧИСЛО! Возникает в случаях:

- в функции с числовым аргументом встречается неприемлемый аргумент;
- функция, основанная на итерационном методе, расходится;
- функция возвращает число больше 10^{307}.

Сообщение #ПУСТО! Генерируется при ошибочной ссылке на ячейку или диапазон.

Excel способен и на инверсные вычисления. То есть, можно попросить его подобрать значение в некоторой ячейке, так чтобы значение в некоторой другой ячейке было равно заданной Вами величине. Для получения этого эффекта нужно войти в пункт меню **Сервис → Подбор параметра**, и в открывшемся окне ввести желаемые установки.

---

[1] Под *формулой массива* понимается единственная формула, связанная со всеми ячейками массива.

Создание диаграмм

Диаграммы предоставляют возможность наглядно отобразить зависимость некоторых данных от задаваемых параметров. В Excel данные, зависящие от параметров, по которым строится диаграмма, называются *рядом*. *Точкой* называется единичный элемент этого ряда.

Для построения диаграммы удобно воспользоваться мастером диаграмм:

I. Выделить данные, по которым строится диаграмма.

II. **Вставка → Диаграмма**. Открывается окно мастера диаграмм.

III. Выбрать **Тип** и **Вид** диаграммы и нажать кнопку **Далее**, которая осуществляет переход ко второму шагу мастера диаграмм (заметим, что для построения *графика*, то есть зависимости одной величины от другой, а не от номера точки, необходимо выбрать **Тип Точечная**).

IV. Второе диалоговое окно дает возможность откорректировать диапазон данных для построения диаграммы, и с помощью переключателя **Ряды в строках** или **столбцах** поменять местами оси X и Y.

На вкладке **Ряд** можно изменить подписи оси категорий (X), добавить или удалить ряды данных. Для добавления рядов данных следует нажать кнопку **Добавить →** сделать активным поле ввода **Имя** и выбрать на рабочем листе ячейку, содержащую имя ряда данных → активизировать поле **Значения** и выделить ячейки со значениями нового ряда данных. Для того чтобы удалить ряд данных, следует выделить в списке **Ряд** удаляемый ряд и нажать кнопку **Удалить**. Здесь же следует отрегулировать подписи оси X.

Нажать кнопку **Далее** и перейти к третьему окну мастера диаграмм.

V. Третье окно мастера позволяет добавить название диаграммы, дать имена осям X и Y, изменить ось категорий, линии сетки, внести изменение в легенду и ее расположение. После нажатия кнопки **Далее** попадаем в четвертое окно мастера диаграмм.

VI. Четвертое окно мастера диаграмм определяет местоположение диаграммы.

После того как диаграмма создана, ее можно редактировать и форматировать. Так, например, текстовые элементы можно изменять в окне **Параметры диаграммы**, которое вызывается командой **Диаграмма → Параметры диаграммы**. После чего на соответствующих вкладках выполняются необходимые для редактирования действия.

Корректировка данных, по которым строится диаграмма, производится следующим образом: Активизируется диаграмма → **Диаграмма → Исходные данные →** в окне **Исходные данные** активизировать поле данных **Диапазон →** выделить новый диапазон данных для диаграммы → **OK**.

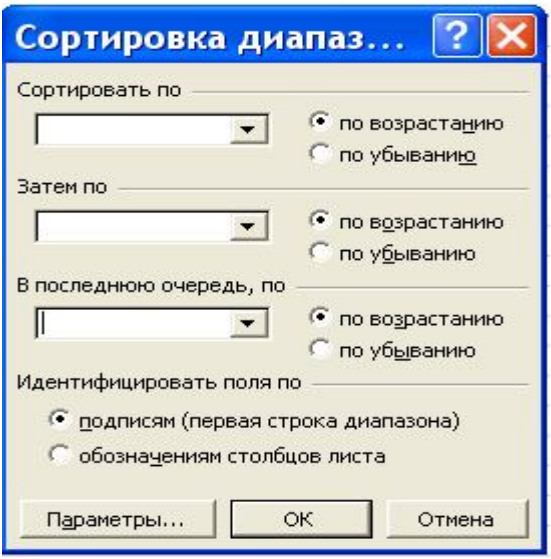
Добавление данных происходит несколько иначе: Активизировать диаграмму → **Диаграмма → Добавить данные →** при открытии окна **Новые данные** активизировать нужный рабочий лист и выделить в нем нужный диапазон → **OK**.

Замечание. Значение диапазона можно записать с клавиатуры.

Сортировка данных

При большом объеме информации данные для улучшения восприятия следует каким-либо способом упорядочить. Для этого в Excel встроена команда **Данные → Сортировка**, при выполнении которой открывается окно **Сортировка диапазона**. Изучение этого окна выявляет возможность произвести сортировку по трем полям, определить порядок сортировки для каждого поля, возможность сортировки строк или столбцов.

В окне параметры сортировки (открывается при нажатии кнопки **Параметры**) можно установить флажок **Учитывать регистр**, что позволяет при сортировке по возрастанию отделять символы верхнего и нижнего регистров. Отсутствие этого флажка означает, что символы этих двух регистров совпадают и записи при сортировке будут следовать в той последовательности, в которой они изначально располагались.



Для сортировки числовых данных в алфавитном порядке каждое число следует снабдить апострофом (`579) или вводить числа в виде формул (="579").

При необходимости произвести сортировку по более чем трем полям следует провести процедуру сортировки несколько раз, до тех пор, пока нужные поля не будут упорядочены.

Фильтрация данных

После создания большого списка часто пользователь желает увидеть только нужные в данный момент сведения. Данное желание можно удовлетворить, применяя процедуру фильтрации данных, которая может осуществляться при помощи *формы данных*: Активизировать ячейку или диапазон в списке → **Данные** → **Форма** → в открывшемся диалоговом окне формы нажать кнопку **Критерии** → в полях ввода задать критерий поиска → нажать клавишу <Enter> или одну из кнопок **Назад**, **Далее** для поиска записи по критерию (записей, удовлетворяющих критерию может быть несколько) → при желании запись можно отредактировать в соответствующих полях на форме → нажать кнопку **Закрыть** для завершения работы с формой.

Замечание. Если запись, соответствующая критерию, отсутствует, то форма переходит в режим ввода новой записи.

В Excel предусмотрены следующие критерии фильтрации: критерий сравнения, критерий в виде образца шаблона, критерий на

основе множественных условий и критерий на основе логических формул.

В критерии сравнения используются только операторы сравнения. Если запись критерия имеет вид = или <>, то выбирается запись с пустым или непустым полем. При определении критерия в виде =(данные), <>(данные), <(данные), >(данные), <=(данные), >=(данные) происходит выбор записи, в соответствующем поле которых выполняются эти равенства или неравенства.

Критерий в виде образца-шаблона позволяет отбирать текстовые значения, подходящие под шаблон. При составлении шаблона могут использоваться символы «?» и «*». Знак вопроса применяется при замене символа, в позиции которого сравнение при поиске/фильтрации не производится. Символ «звездочка» позволяет задать неограниченную последовательность символов, сравнение по которым не производится, начиная с позиции, в которой стоит указанный символ.

Замечание. Критерий в виде образца-шаблона работает только с текстом. При необходимости использования этого критерия при фильтрации чисел их следует задать в виде формул (="579") или ставить значок апострофа перед числом (`579).

Критерий на основе множественных условий используется при поиске/фильтрации по нескольким условиям. При этом используются логические операторы И (AND) и ИЛИ (OR).

Замечание. При фильтрации с помощью формы данных может использоваться только логическая операция AND.

Критерий на основе логических формул позволяет произвести поиск/фильтрацию с помощью логических выражений, возвращающих значение ИСТИНА или ЛОЖЬ. Но этот критерий может использоваться только в расширенном фильтре.

Автофильтр

Применение автофильтра происходит следующим образом:
Активировать список → **Данные** → **Фильтр** → **Автофильтр** (в строке заголовков базы данных появляются кнопки, характеризующие раскрывающийся список) → открыть раскрывающийся список в заголовке поля и выбрать необходимый способ фильтрации.

В раскрывающемся списке **Автофильтра** можно выбрать следующие пункты: **Все** (фильтрация по полю отключена), **Первые 10** (отбор не более 10 вариантов), **Условие** (пользователь задает условие), <значение одного из элементов базы данных> (производится фильтрация всех записей с заданными значениями), **Пустые** (выводятся строки с пустыми полями), **Не пустые** (выводятся строки с непустыми полями).

Замечание. Фильтрация по команде **Первые 10** работает только с числовыми данными.

Пользовательский автофильтр

При выборе команды **Условие** в раскрывающемся списке поля заголовка открывается диалоговое окно **Пользовательский автофильтр**. Условие сравнения задается с помощью раскрывающихся списков. В левых списках задаются способы сравнения, в правых – значения для сравнений. Переключатели «И» и «ИЛИ» служат для объединения условий. После задания условий следует нажать кнопку **ОК** для проведения фильтрации.

Расширенный фильтр

Более мощным средством поиска/фильтрации в Excel служит **Расширенный фильтр**. Данный способ извлечения информации несколько сложнее, так как требует создания списка, содержащего критерии. Но именно этот список позволяет задавать достаточно сложные и в то же время гибкие условия фильтрации.

Применение **Расширенного фильтра** происходит в соответствии с процедурой:
Создается критерий для фильтрации и область извлечения данных → **Данные** → **Фильтр** → **Расширенный фильтр** (открывается окно **Расширенный фильтр**) → переключатель **Обработка** позволяет *фильтровать список на месте* или *скопировать результат в другое место* → в поле **Исходный диапазон** указывается местоположение

фильтруемого списка → в поле **Диапазон условий** указывается местоположение критерия → в поле **Поместить результат в диапазон** (поле активно если установлен переключатель **скопировать результат в другое место**) указать диапазон извлечения → флажок **Только уникальные записи** позволяет исключить одинаковые записи → **OK**.

Замечание. Следует отметить, что строки в расширенном фильтре объединяются по правилу логической операции «ИЛИ» (OR), а поля – по правилу операции «И» (AND).

Пример.

Имя	**Любимый цвет**	**Автомобиль**
Пётр	белый	BMW
Николай	белый	LADA
Василий	синий	BMW
Сергей	белый	LADA
Андрей	синий	LADA

КРИТЕРИЙ 1

Имя	**Любимый цвет**	**Автомобиль**
		LADA
	синий	

КРИТЕРИЙ 2

Имя	**Любимый цвет**	**Автомобиль**
	синий	LADA

Результат 1

Имя	**Любимый цвет**	**Автомобиль**
Николай	белый	LADA
Василий	синий	BMW
Сергей	белый	LADA

Андрей	синий	LADA

Результат 2

Имя	**Любимый цвет**	**Автомобиль**
Андрей	синий	LADA

Взаимосвязь данных

Работа с объемными таблицами и базами данных часто заставляет вносить в некоторые ячейки одни и те же данные, производить вычисления, опираясь на содержимое других ячеек и создавать более сложные зависимости одних данных от других. Для выполнения этих действий на простейшем уровне надо уметь связывать ячейки.

Связывание ячеек

Для внесения содержимого одной ячейки в другую ячейку можно произвести записав формулу, содержащую ссылку, в ячейку-приемник. Например из ячейки А15 листа1 произвести запись в какую либо ячейку листа2:

=Лист1!A15 или =exp(Лист1!A15).

Если нужно связать данные из другой книги, то следует указать полный путь к файлу книги:

'D:\STUDENT\[mybook.xls]Лист1'!A15.

Еще раз напомним, что в апострофах указывается путь к листу, имя ячейки стоит вне апострофов после восклицательного знака.

С другой стороны, ячейки можно связать с помощью команд меню: Выделить диапазон ячеек (источник) → **Правка** → **Копировать** → открыть книгу, в которой находятся ячейки-приемники → **Правка** → **Специальная вставка** → в открывшемся окне **Специальная вставка** нажать кнопку **Вставить связь**, которая устанавливает ссылку на ячейки-источники.

Изменить связь означает отредактировать ссылку в строке формул. Но когда надо вносить много изменений, то проще и быстрее создать связи заново. Исключение составляет случай изменения имени книги или пути к ней: **Правка** → **Связи** → в открывшемся окне **Изменение связей** выделить связь, для которой следует внести изменения → нажать кнопку **Изменить** (откроется окно **Изменить источник**) → выделить нужную книгу-источник → **ОК**.

Консолидация

Обобщение однородных данных из различных источников в Excel называется *консолидацией*. В ходе обобщения над содержимым исходных ячеек производятся некоторые действия и результат записывается в новую ячейку или диапазон. Можно установить связь между исходными ячейками и консолидированными данными, что позволяет при смене значений в исходных данных наблюдать изменения в обобщении. В Excel встроены два основных способа консолидации – *консолидация по физическому расположению ячеек* и *консолидация по заголовкам*.

Консолидация по физическому расположению ячеек

При консолидации по физическому расположению указывается местоположение ячеек с исходными данными. Поэтому в целях недопущения ошибочных ссылок таблицы должны иметь одинаковую структуру.

Произвести процедуру консолидации можно так: Выделить верхнюю левую ячейку диапазона, в который будут помещены обобщенные данные → **Данные** → **Консолидация** (откроется диалоговое окно **Консолидация**)→ в списке **Функция** выбрать нужную для обобщения данных → ввести ссылку в поле **Ссылка** на первый исходный диапазон (можно с помощью мыши; если данные в закрытой книге, то следует нажать кнопку **Обзор** и выбрать книгу, закончить запись ссылки придется вручную) → повторить действия над ссылками для каждого исходного диапазона → **ОК**.

В ходе консолидации доступны следующие функции: сумма, количество значений, среднее, максимум, минимум, произведение, количество чисел, смещенное отклонение, несмещенное отклонение, смещенная дисперсия, несмещенная дисперсия.

Очевидно, что приведенную выше процедуру консолидации по физическому расположению можно провести вручную с помощью ссылок.

Консолидация по заголовкам строк и столбцов

Данный вид консолидации позволяет идентифицировать ячейки по заголовкам, что позволяет им быть в различных местах рабочих листов. Другими словами, конкретная ячейка характеризуется именем строки и именем столбца. В случае если имя строки не определено, то столбцы консолидируются по имени, а ячейки – по расположению.

Процедура консолидации по заголовкам производится по схеме: Выделить диапазон, в котором будут помещаться консолидированные данные (строки и столбцы должны иметь имена) → **Данные** → **Консолидация** → в окне **Консолидация** в списке **Функция** выбрать функцию для консолидации → в поле **Ссылка** ввести ссылку на первый исходный диапазон с заголовками (при консолидации будут выбраны только те столбцы и строки, которые содержатся в результирующем диапазоне) → нажать кнопку **Добавить** для добавления к списку исходных диапазонов → повторить действия с диапазонами исходных данных для добавления и учета в консолидации → в группе **Использовать в качестве имен** указать с помощью флажков где расположены заголовки → **ОК**.

Замечание. Если в окне **Консолидация** установить флажок **Создавать связи с исходными данными**, то при выполнении консолидации на листе, где располагаются обобщенные результаты, будет создана структура, которая позволяет видеть исходные данные и результат их обработки.

Сводные таблицы

Сводная таблица – это таблица, содержащая сводную информацию из нескольких источников данных, с которыми она связана и предназначена для изучения и анализа информации, в ней содержащейся.

Для создания сводной таблицы используется **Мастер сводных таблиц и диаграмм: Данные → Сводная таблица** (открывается окно **Мастер сводных таблиц и диаграмм**)→ установить нужный переключатель в группе **Создать таблицу на основе данных находящихся:** → в группе **Вид создаваемого отчета** установить переключатель в положение **сводная таблица** → нажать кнопку **Далее** (переход ко второму шагу мастера) → на втором шаге мастера следует выделить таблицу с исходными данными → **Далее** (переход к третьему шагу мастера) → выбрать вариант размещения таблицы в группе **Поместить таблицу в** → нажать кнопку **Параметры** для установки параметров сводной таблицы → нажать кнопку **Макет** для открытия окна **Мастер сводных таблиц и диаграмм – макет**, в котором с помощью мыши размещаем поля в сводной таблице → **ОК** (для закрытия макета) → нажать кнопку **Готово** для завершения процесса создания сводной таблицы.

Если не нажимать кнопку **Макет**, то на рабочем листе появится панель инструментов **Сводные таблицы** и разметка поля будущей таблицы. В этом случае структуру сводной таблицы следует создавать

непосредственно на рабочем листе. Небольшое экспериментирование позволяет пользователю разобраться в процессе создания нужной структуры сводной таблицы.

После того как сводная таблица создана, можно улучшить ее восприятие путем группировки некоторых элементов. Для этого нужно выделить ячейки, которые должны быть сгруппированы → в контекстном меню выбрать команду **Группа и структура → Группировать →** при желании переименовать группу. Кроме того, двойной щелчок в названии группы позволяет отразить суммарные данные.

Для детализации сводной таблицы можно воспользоваться диалоговым окном **Показать детали**, которое открывается при двойном щелчке по полю строк сводной таблицы. В этом окне следует выбрать поле, которое нужно отразить и нажать кнопку **OK**.

Отметим, что Excel может группировать элементы так же по временным диапазонам или числовым с некоторым шагом.

Кроме того, имеется возможность произвести сортировку в полученной сводной таблице:

Данные → Сортировка (откроется окно **Сортировка**) → в сводной таблице выделить поле, которое следует сортировать → переключатель **Сортировать** установить в одно из положений **Значения** или **Подписи →** установить порядок сортировки **по возрастанию** или **по убыванию → OK**.

Изменение значения итоговых функций

По умолчанию Excel подводит итоги в сводных таблицах при помощи функции суммирования (в текстовых полях вычисляется количество элементов). Но этот метод не всегда отражает результат, который нужен пользователю. Для изменения функции для всех итогов нужно выполнить действия: Щелкнуть правой кнопкой мыши, когда ее указатель находится в поле данных сводной таблицы → в контекстном меню выбрать пункт **Параметры поля →** В открывшемся диалоговом окне выбрать нужную функцию → **OK.**

Если нужно использовать свою функцию для каждого поля, то следует выполнить для каждого из них процедуру: Установить указатель мыши в одну из ячеек поля данных → вызвать контекстное меню и выбрать команду **Параметры поля →** выбрать итоговую функцию **→ OK**.

Вставка вычисляемого поля

Достаточно часто возникает потребность по данным сводной таблицы произвести некоторые вычисления. Для этого можно воспользоваться вставкой вычисляемого поля: Переместить указатель мыши в область данных и вызвать контекстное меню → **Формулы → Вычисляемое поле →** в диалоговом окне **Вставка вычисляемого поля** ввести имя поля → ввести формулу, по которой будет происходить вычисление (настоятельно рекомендуется при наборе формул пользоваться списком **Поля**) → **Добавить → ОК**.

Замечание. В формулах, которые записываются в окне **Вставка вычисляемого поля**, нельзя использовать ссылки не ячейки электронной таблицы.

Некоторые замечания по печати

Печать документов Word не вызывает у начинающего пользователя никаких вопросов, поскольку осуществляется постранично. При печати таблиц Excel обычно возникает желание печатать не абстрактные страницы, на которые компьютер разбивает рабочую книгу, а конкретный диапазон ячеек, называющийся в Excel массивом. Для этого нужно выделить желаемый диапазон, а перед печатью, в панели «Вывести на печать» поставить точку около фразы «выделенный диапазон».

Если печатаемая таблица не помещается на одном листе, но при этом имеет «шапку», которую желательно распечатать на каждом из листов бумаги, то следует использовать пункт меню **Файл → Параметры страницы →** вкладка **Лист → Печатать на каждой странице → Сквозные строки**, и установить строки «шапки».

На этом краткое руководство по работе с программой Microsoft Excel завершаем. При появлении трудностей следует обращаться к справочной системе, книгам или к преподавателю.

Часть III. Введение в MS Access

Вводные замечания

В состав Microsoft Office входит программа MS Access, позволяющая создавать базы данных и управлять ими. При этом большинству пользователей не нужно прибегать к программированию. Средства MS Access достаточно наглядные и понятные при наличии некоторого опыта работы с данной программой.

В MS Access под **базой данных (БД)** понимается файл, в котором хранятся данные и настройки системы управления базами данных.

Хранилищем информации БД выступает таблица, состоящая из **полей** (столбцов) и **записей** (строк). Запись (строка в таблице) – стандартный блок для хранения данных в таблице и выборке данных при запросе.

Еще одним важным элементом, которым обладает таблица, является **ключевое поле**, служащее для однозначного определения записи в таблице.

Одно или несколько ключевых полей, позволяющих идентифицировать запись таблицы и организовать связь между таблицами, называется **ключом**.

Во многих приложениях встречается аббревиатура **SQL** (structured query language). Этот язык встроен во многие программные продукты различных фирм. Поэтому с базой данных созданной в MS Access можно работать, применяя другие специализированные приложения.

Корпорация Microsoft, создав пакет Microsoft Office, позаботилась еще и о том, чтобы составные части этого программного продукта могли взаимодействовать друг с другом. Таким образом, появляется возможность, например, средствами Excel отобразить данные из базы

данных, которая создана с помощью MS Access, или таблицу из Excel импортировать в БД MS Access и т.д.

База данных MS Access

Информация в базе данных, созданной в программе MS Access, хранится в таблицах, которые связаны между собой. Поэтому первый шаг в создании БД – это формирование таблиц.

Создание таблиц

Таблицу можно создать в режиме конструктора, с помощью мастера или путем ввода данных (режим таблицы). Для этого следует выполнить процедуру: **Файл → Создать → Новая база данных →** в открывшемся окне **Файл новой базы данных** указать, в какой папке будет располагаться таблица → нажать кнопку **Создать.** В результате откроется окно, в котором выбирается способ создания таблицы (рис.1).

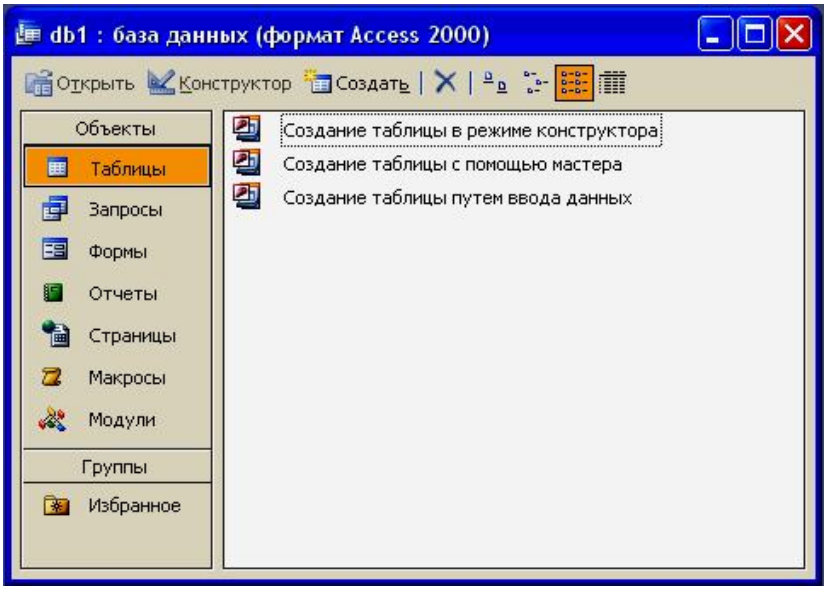


Рис. 1

Режим таблицы путем ввода данных

Выбрав пункт **Создание таблицы путем ввода данных**, пользователь получает доступ к заготовке, представленной на рис. 2.

В этой абстрактной таблице поля следует переименовать, чтобы они несли смысловую нагрузку. Тип полей программа определит автоматически, в зависимости от внесенной в них информации (чем больше данных в таблицу будет введено, тем точнее программа определит тип данных и размеры полей). Сохранение таблицы производится обычным способом: **Файл → Сохранить →** в окне **Сохранение** ввести имя таблицы **→** нажать кнопку **OK** (при сохранении таблицы все пустые столбцы с неизмененными именами будут удалены).

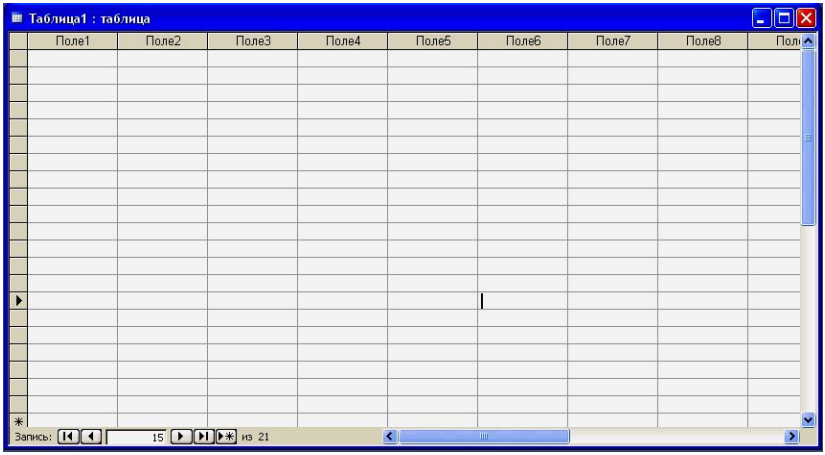


Рис. 2

Замечание. Если MS Access неверно определит какие-либо свойства полей, то исправления можно внести в режиме конструктора.

При работе с программами Microsoft постоянно приходится выделять тот или иной объект. Для выделения ячейки следует передвинуть указатель мыши в ее левую часть так, чтобы он превратился в жирный белый крестик, и нажать левую кнопку мыши. При двойном щелчке в ячейке выделяются только данные. Столбцы выделяются щелчком левой кнопкой мыши на его заголовке. Для выделения смежных столбцов используется клавиша <Shift> и мышь с нажатой левой кнопкой, указатель которой перемещается по нужным заголовкам.

Нажатие правой кнопки мыши вызывает контекстное меню, соответствующее ячейке в которой находится ее указатель. Назначение команд контекстного меню следует из их названия.

Если в таблице не заданы ключевые поля, то эта таблица не готовая база данных. Поэтому при закрытии таблицы MS Access предложит автоматически создать ключевое поле. Имя этого поля по умолчанию будет **Код**, и хранить оно будет номер записи в таблице. Поле-счетчик заполняется автоматически при появлении в таблице новой записи.

Режим конструктора

Более гибкое создание БД возможно в **режиме конструктора**, который, кроме того, позволяет полнее понять, что такое таблица MS Access (рис. 3). В этом случае структура таблицы создается пользователем «с нуля».

В верхней части окна конструктора расположен бланк с перечнем всех полей, заголовков, их типов и описаний. Графа **Описание** не является обязательной и предназначена для текста подсказки, которая появляется во время работы с таблицей.

Тип поля выбирается в ячейке столбца **Тип данных** из списка, который появляется при щелчке в соответствующей ячейке.

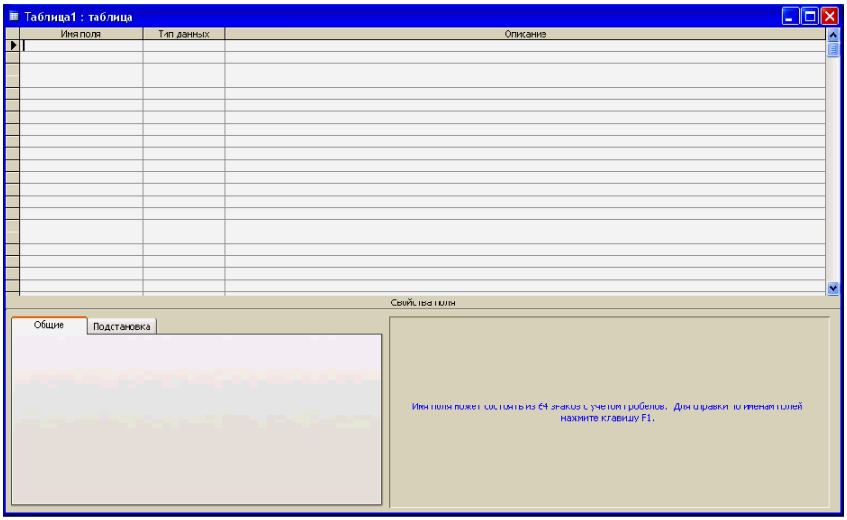


Рис. 3

Одно из полей таблицы должно быть определено как ключевое. Для этого следует установить курсор на соответствующую строку бланка и нажать кнопку **Ключевое поле** (или выбрать эту команду из контекстного меню). Если идентифицировать запись в таблице можно только по нескольким полям, то следует выделить несколько строк в

бланке и выбрать команду **Ключевое поле**. Ключевые поля выделяются изображением ключа в области выделения строки. Не может быть более одного ключа. Поэтому изображение символа ключа в нескольких строчках бланка свидетельствует о составном ключе.

Типы (форматы) полей

При щелчке мышью в колонке **Тип данных**, соответствующей некоторому полю, появляется возможность выбрать один из следующих форматов: **текстовый**, **числовой**, **дата/время**, **денежный**, **счетчик**, **логический**, **поле объект OLE**. Кроме того, в этом списке находится **мастер подстановок**.

Текстовый формат

На рис. 4 представлена вкладка *Общие* со свойствами текстовых полей.

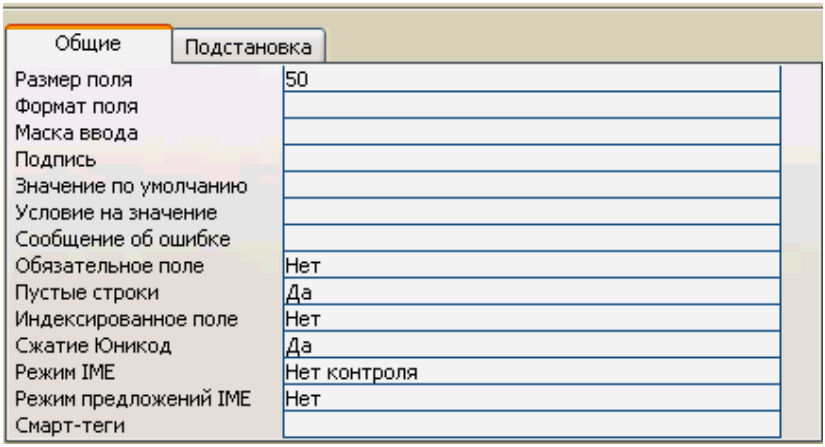


Рис. 4

Замечание. При конструировании базы данных нет необходимости явно указывать все свойства, но следует знать эту возможность и при необходимости ею пользоваться.

В первой строке свойств задается размер текстового поля, который может быть от 1 до 255 символов.

Формат поля позволяет жестко задать вид и размер вводимых строк. Кодовые символы формата имеют вид:

@ – должен быть текстовый символ или пробел;

& – текстовый символ;

> – преобразование символа в верхний регистр;

< – преобразование символа в нижний регистр.

Формат поля может состоять из двух частей, разделенных знаком «**;**». Правая часть – формат ввода, а вторая – определяет значение поля, если данные в него не были введены.

Маска ввода – последовательность кодовых символов:

0 – должна быть цифра от 0 до 9;

9 – цифра или пробел;

– цифра, пробел, плюс или минус;

L – должна быть буква (A,…,Z, A,...,Я);

? – буква;

A – должна быть буква или цифра;

a – может быть буква или цифра;

& – должен быть любой символ или пробел;

C – произвольный символ;

«.», «,», «;», «:»,«-», «/» – разделители, которые сохраняют свой вид в строке данных;

> – преобразование символа в верхний регистр;

< – преобразование символа в нижний регистр;

! – заполнение маски справа налево;

\ – ввод следующего за обратной косой чертой символа как символьной константы.

Подпись – это второй (первый имя) идентификатор поля, который используется программой вместо имени поля при работе с данными в табличной форме для создания заголовка столбца.

Свойство **Значение по умолчанию** позволяет автоматически подставлять заданное значение во все вновь создаваемые поля.

Условие на значение – это фильтр, который разрешает вводить в поле только то, что удовлетворяет заданному условию.

Обязательное поле – логическое значение «Да» означает, что в поле обязательно должны быть введены данные.

Свойство **Пустые строки** – определяет разрешены или нет в данном поле пустые строки.

Свойство **Индексированное поле** может принимать значения *поле неиндексированно*, *индексировано*, *но допускает повторяющиеся значения, индексировано с запретом повторяющихся значений*.

Свойства **Сжатие Юникод**, **Режим IME**, **Режим предложений IME**, **Смарт-теги** обсуждать здесь не будем ввиду их специфичности и ограниченности объема пособия. Сведения о них можно найти в справочной системе программы MS Access (вызывается нажатием <F1>).

Поле **MEMO** – это текстовый формат очень большой длины (сохраняет 65536 знаков), служащий для ввода комментариев, примечаний и т. п.

Числовой формат

Вкладка для определения свойств числового поля представлена на рис. 5.

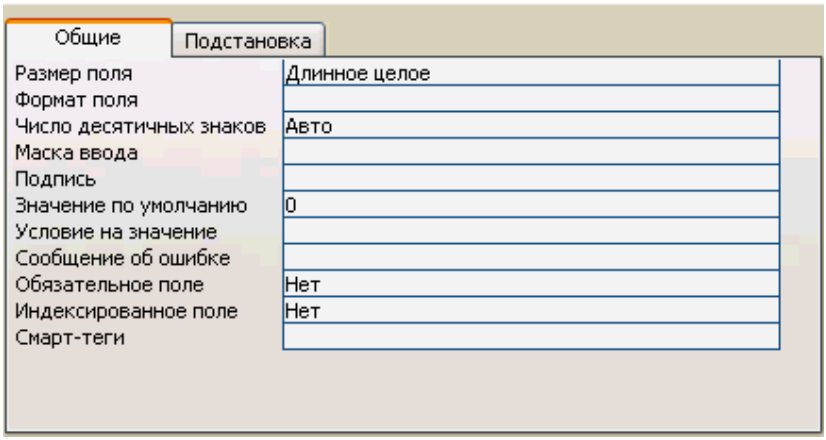


Рис. 5

Свойство **Размер поля** определяет представление числа и может принимать значения:

Байт – целые числа от 0 до 255;

Целое – целые числа от –32768 до 32767;

Длинное целое – целые числа от –2147483648 до 2147483647;

Одинарное с плавающей точкой – числа с семью знаками после запятой от –3.402823 E38 до 3.402823E38;

Двойное с плавающей точкой – числа с 15 знаками после запятой от –1,79769313486231E308 до 1,79769313486231E308;

Код репликации – при конструировании таблицы не используется;

Действительное – числа, содержащие 28 десятичных знаков и располагающиеся в диапазоне от –1E–28 до 1E28–1.

Формат поля определяет вид отражаемых в таблице чисел. Для этого следует выбрать какой-либо формат из списка или использовать следующие кодовые символы для создания нового формата:

. – точка используется в качестве десятичного разделителя;

, – запятая применяется как разделитель групп разрядов;
0 – вывод цифры или нуля, если разряд незначащий;
– вывод цифры;
$ – знак доллара;
% – вывод числа в процентном формате;
E или e – вывод числа в экспоненциальной форме.

При этом могут быть заданы четыре группы кодов: первая для ввода положительных чисел, вторая – для отрицательных, третья – для отражения нулевого значения и четвертая – для пустых полей. Группы разделяются знаком точка с запятой.

Пример. Число 123456,789 при задании формата поля конструкцией # ###,# будет выглядеть следующим образом: 123 456,78900 (при указании числа десятичных знаков, равном 5), Кроме того, в формате можно задать цвет выводимых символов:

#[Красный];-[Синий];0[Зеленый];”Нет данных”

При этом обязательно следует создать группу для нулевого значения, иначе ноль не будет выводиться на экран.

Формат дата/время

Вкладка со свойствами полей **дата/время** представлена на рис. 6.

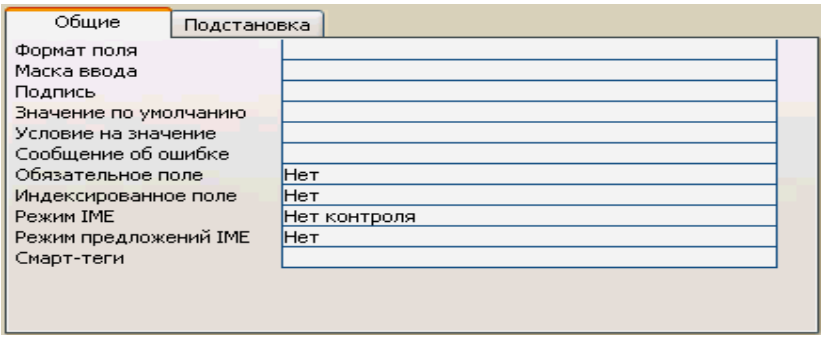


Рис. 6

Следует подчеркнуть, что в случае смены формата поля дата/время часть данных может измениться. Это произойдет, если записи в таблице были сделаны в разных форматах (например, время 15:45 и дата 15/09/05). Поэтому поле данного типа должно быть определено в самом начале работы. В свойствах целесообразно указать *Условие на значение* и *Сообщение об ошибке*.

Денежный формат

Свойства вкладки *денежного* формата (рис. 7) аналогичны описанным выше.

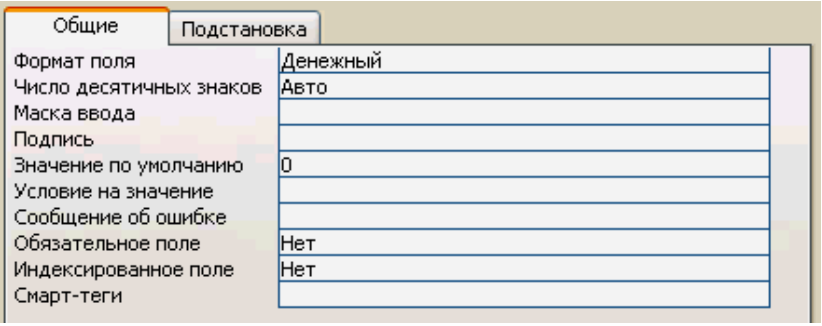


Рис. 7

Денежный тип поля используют для предотвращения округления во время вычислений. В денежных полях обеспечивается 15 знаков слева от десятичной запятой и 4 знака справа. Денежное поле занимает 8 байт на диске.

Формат Счетчик

Свойства поля типа **Счетчик** отображены на рис. 8.

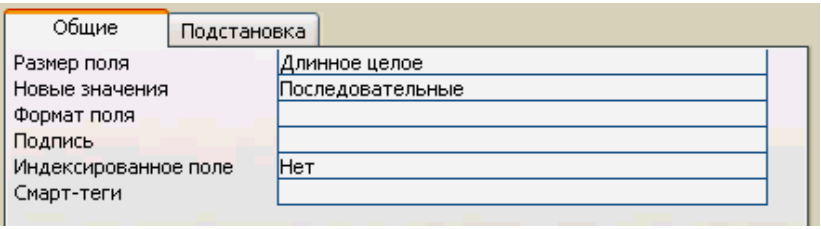


Рис. 8

Следует отметить, что **Счетчик** – это всегда число, предназначенное для идентификации записей в таблице. Изменение этого числа происходит автоматически последовательно или случайным образом. Первый вариант позволяет нумеровать записи, второй, когда числа выбираются случайным образом, уменьшает вероятность замены одного кода другим.

Логический формат

Ячейка в поле логического типа может содержать только одно из двух значений «Да» или «Нет». Свойства полей логического типа отображены на рис. 9.

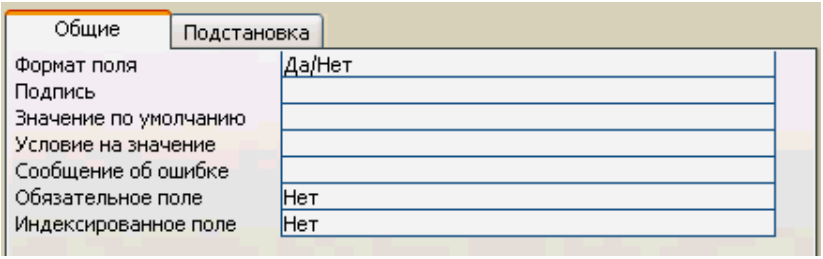


Рис. 9

Два других названия поля логического типа – **Истина/Ложь** и **Включено/Выключено**. Логическое поле представляется в таблице в виде набора флажков. Изображение галочки в квадрате соответствует логическому «Да», отсутствие галочки присваивает ячейке поля логическое значение «Нет» (рис. 10).

Рис. 10

Поле объекта OLE

Поле объекта OLE (Object Linking and Embedding) имеет два свойства (рис. 11).

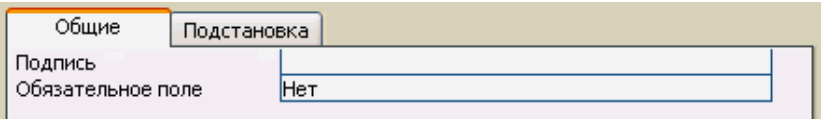


Рис. 11

Это поле не хранит информацию, а содержит ссылку на объект, который включается в базу данных с помощью *OLE-протокола* обмена данными.

Для внедрения объекта в ячейку следует установить в ней курсор → нажать правую кнопку мыши → в появившемся контекстном меню выбрать команду **Вставить объект** → в открывшемся диалоговом окне **Вставка объекта** (рис. 12) при установке переключателя в положение **Создать новый** встраиваемое приложение создается «с нуля»; если встраиваемый файл уже существует, то переключатель надо установить в положение **Создать из файла** → MS Access откроет **окно открытия документа** (рис. 12)→ выбирается документ → **ОК** → MS Access автоматически открывает приложение, соответствующее этому файлу → можно произвести редактирование встраиваемого файла → при закрытии программы объект оказывается внедренным.

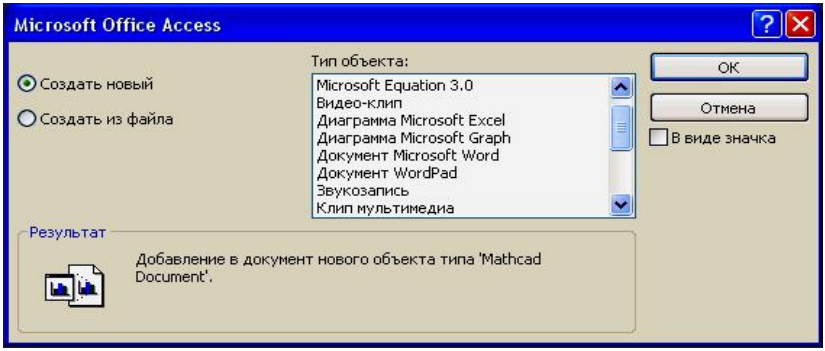


Рис. 12

Замечание. Если нужно внедрить существующий файл без редактирования, то можно воспользоваться процедурой: в проводнике Win-

dows найти и выделить файл → <Ctrl + C> (копирование файла в буфер обмена) → в MS Access выбрать ячейку в поле объекта OLE → активизировать контекстное меню (щелчок правой кнопкой мыши) → выполнить команду **Вставить**.

Активизация работы со встроенными объектами осуществляется двойным щелчком мыши.

Мастер подстановок

В списке типов полей имеется элемент являющийся не названием типа, а командой. Эта команда называется **Мастер подстановок**, создающая связь между таблицами. Действие этой команды рассмотрим на примере. Пусть даны две таблицы (рис.13 и 14), в которых присутствуют поля типа «**Счетчик», числовой и текстовый**.

Таблица12 : таблица

	Код1	Функция	Значение в точке x=2
▶	1	x+1	3
	2	x^2+2	6
	3	x^3+3	11
	4	x^4+4	20
	5	x^5+5	37
*	(Счетчик)		0

Рис. 13

Таблица22 : таблица

	Код2	Вариант	Производные
	1	15	1
	2	24	2x
	3	33	3x^2
	4	42	4x^3
▶	5	51	5x^4
*	(Счетчик)	0	

Рис. 14

Подставим поле **Вариант** из *Таблицы22* в *Таблицу12*, используя следующую процедуру: перейдем в режим конструктора для *Таблицы12* → введем имя нового поля, например, *Номер задания* →в графе **Тип данных** выбираем команду **Мастер подстановок** → на первом шаге этого мастера выбираем пункт соответствующий использованию данных из другой таблицы → нажмем кнопку **Далее** → на втором шаге выбираем таблицу, из которой осуществляется подстановка (*Таблица22*) → **Далее** → третий шаг предназначен ддля выбора подставляемого поля или нескольких полей (порядок следования полей важен, так как первое указанное поле является источником данных в операции подстановки) → **Далее** → выбираем порядок сортировки списка (по возрастанию или убыванию) → регулируем ширину, подставляемой колонки → **Далее** → задаем подпись столбца → очередное нажатие кнопки **Готово** завершает работу **Мастера подстановки**.

Мастер подстановки произвел следующие действия (рис.15 и рис.16):

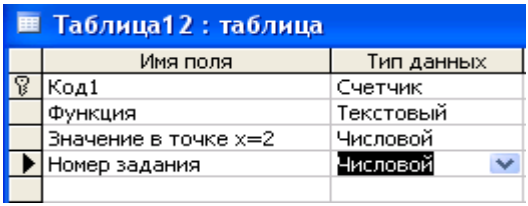


Рис. 15

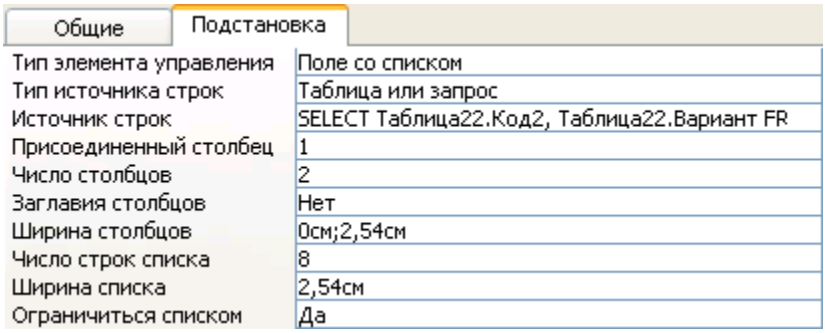


Рис. 16

1. Новому полю присвоен числовой тип (числовой тип относится к ключевому полю **Таблица22.Код2**, а не данным подстановки);

2. В *Таблице12* появился новый столбец, в котором каждая ячейка может быть заполнена данными из списка, содержащие данные выбранного поля *Таблицы22* (заполнение происходит при выделении значения в списке);

3. Если на третьем шаге **Мастера постановки** выбрать два поля (*Вариант* и *Производная*), то список подстановки формируется на основе выбранного первого поля, но при открытии списка будут отражены значения обоих полей, разделенных вертикальной линией. При этом вторая колонка выступает в качестве некоторого комментария и не подставляется в таблицу (рис.17 и рис. 18).

Рис. 17

Рис. 18

При выполнении команды **Сервис → Схема данных** раскрывается схема базы данных (рис. 19), в которой отражается связь между *Таблицей12* и *Таблицей22*. Подстановка на этой схеме обозначена линией без всяких обозначений.

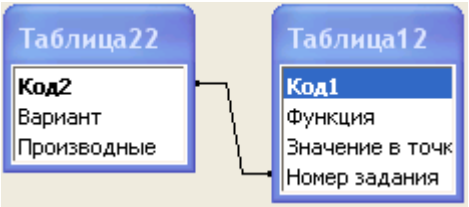


Рис. 19

Замечание 1. Для удаления поля подстановки следует вначале удалить связь между таблицами. Для этого в **Схеме данных** нужно выделить линию, соединяющую таблицы и в контекстном меню выбрать команду **Удалить связь**, и в режиме конструктора убрать поле подстановки.

Замечание 2. Для сохранения структуры таблиц перед выходом из режима конструктора необходимо нажать на панели инструментов кнопку **Сохранить** или выполнить команду **Файл → Сохранить** и нажать кнопку **Закрыть** в верхнем правом углу окна конструктора.

Замечание 3. Отметим, что в MS Access имеются заготовки таблиц**,** доступные через **Мастер таблиц**. Если эти шаблоны не вполне устраивают требованию пользователя, то можно их корректировать в режиме конструктора.

Межтабличные связи

База данных обычно состоит из нескольких таблиц, описания связей между ними, форм, отчетов и других объектов, помогающих пользователю работать с ней. При эксплуатации БД часто возникает необходимость редактирования данных**.** Всякое изменение любого значения удобно делать только в одном месте БД. С другой стороны, в таблицах следует группировать данные по определенной тематике. И, наконец, следует выделять блоки, которые можно совершенствовать по отдельности, а таблицы создавать таким образом, чтобы они могли быть использованы в другой БД. Некоторые из перечисленных замечаний удается удовлетворить с помощью **Мастера по анализу таблиц**.

Мастер по анализу таблиц

Процедура исключения дублирования данных в таблице называется *нормализацией*. Эту процедуру помогает осуществить **Мастер по анализу таблиц**, который запускается из меню таким образом: **Сервис**

→ **Анализ** → **Таблица**. Рассмотрим работу этого мастера на примере преобразований следующей таблицы (рис. 20).

Таблица3 : таблица

	Код	Вариант	Функция	Число	Буква
▶	1	Первый	x^2+1	2	А
	2	Второй	x+11	12	Б
	3	Третий	5x	5	В
	4	Первый	5x^2+2	7	А
	5	Второй	15x	15	Б
	6	Третий	x^3	1	В
	7	Третий	x^2+5	6	В
	8	Второй	2x+2	4	Б
	9	Первый	5x+5	10	А
	10	Первый	x^2+9x+1	11	А
*	(Счетчик)				

Рис. 20

Первые два шага **Мастера** содержат информацию о преобразовании таблицы, которое возможно будет произведено.

Замечание. Исходная таблица при работе **Мастера по анализу таблиц** не меняется, а создаются новые таблицы, содержащие те же данные. В дальнейшем пользователь должен решить какие таблицы оставить в базе данных.

На третьем шаге мастера выбирается таблица, над которой будет произведена работа. Кроме того, третий шаг позволяет отключить показ вводных страниц (первых двух шагов мастера).

На четвертом шаге запрашивается подтверждение распределения данных по двум или более таблицам в случае обнаружения необходимости разбиения таблицы.

Пятый шаг мастера демонстрирует первый результат своей работы. Исходная таблица разбита на две и дублирование данных исключено. Эти таблицы связаны с помощью операции подстановки, при этом связь называется «многие-к-одному». Графически это отображается на рис. 21.

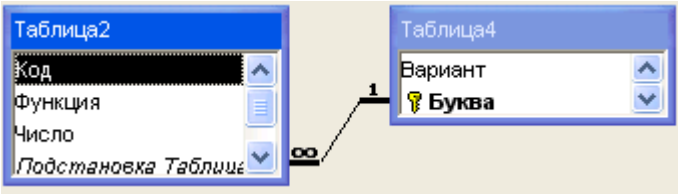


Рис. 21

Здесь видно, что в *Таблице4* поле **Буква** сделано ключевым автоматически. Этот вид связи нас может не устраивать. Поэтому существуют другие виды связи между таблицами, которые будут рассмотрены позже.

На шестом шаге мастера происходит проверка уникальности поля подстановки, как ключевого поля. В случае повторяющихся записей программа предлагает создать еще одно поле, которое станет ключевым, и будет содержать уникальные числовые коды. Для этого следует нажать кнопку **Добавить ключ**  .

Замечание. Если ключевое поле не создавать, хотя программа этого требует, то откроется окно, в котором можно исправить обнаруженный недочет.

Таблица2 : таблица

Код	Функция	Число	Подстановка Таблица4
1	x^2+1	2	А, Первый
4	5x^2+2	7	А, Первый
9	5x+5	10	А, Первый
10	x^2+9x+1	11	А, Первый
2	x+11	12	Б, Второй
5	15x	15	Б, Второй
8	2x+2	4	Б, Второй
3	5x	5	В, Третий
6	x^3	1	В, Третий
7	x^2+5	6	В, Третий
(Счетчик)			

Запись: 11 из 11

Рис. 22

Седьмой шаг мастера является завершающим. Остается ответить на вопрос о необходимости создания запроса. При выполнении запроса таблица-запрос будет доступна на вкладке **Запросы**.

Итак, мастер по анализу таблиц создал связанные таблицы (рис. 22 и рис. 23), эквивалентные исходной, и таблицу-запрос (рис. 24).

Рис. 23

Рис. 24

Замечание. В *Таблице2* имеется поле *Подстановка Таблица4* – это вычисляемое поле и дублирование в нем не противоречит принципам нормализации данных.

Связь типа «один-ко-многим»

Выше уже было отмечено, что связь таблиц с помощью подстановки не всегда является хорошим решением при создании БД. Поэто-

му опишем как поэтапно создается связь других типов на примере уже созданных таблиц.

Замечание. Связываемые поля должны быть одинакового типа. Числовые поля при этом должны иметь одинаковые значения свойства **Размер поля**. Поле счетчика можно связать с числовым полем типа **Длинное целое**.

Произведем подготовительные действия.

1. В *Таблице2* вставим столбец (**Вставка → Столбец**) с названием *Код Варианта* текстового типа.
2. В режиме конструктора для *Таблицы2* для поля с подписью *Подстановка Таблица4* **установить тип элемента управления** как **Поле** и сохранить изменения.
3. В *Таблице2* выделить поле *Подстановка Таблица4* → **Правка → Копировать** → выделить поле *Код Варианта* → **Правка → Вставить.**
4. Удалить связь между таблицами: **Сервис → Схема данных** → в окне **Схема данных** выделить существующую связь → нажать клавишу <Delete> → закрыть окно **Схема данных**.
5. В *Таблице2* выделить поле *Подстановка Таблица4* → **Правка → Удалить столбец**.

В результате пяти указанных шагов получаем результат, изображенный на рис. 25.

Таблица2 : таблица

	Код	Код_Варианта	Функция	Число
▶	1	А	x^2+1	2
	2	Б	x+11	12
	3	В	5x	5
	4	А	5x^2+2	7
	5	Б	15x	15
	6	В	x^3	1
	7	В	x^2+5	6
	8	Б	2x+2	4
	9	А	5x+5	10
	10	А	x^2+9x+1	11
*	(Счетчик)			

Запись: 1 из 10

Рис. 25

Установим теперь связь *Таблицы4* и *Таблицы2*:

6. **Сервис → Схема данных →** выделить мышью в *Таблице4* поле *Буква* и не отпуская левую кнопку мыши перетащить указатель на поле *Код Варианта Таблицы2* → откроется окно **Связи →** в окне **Связи** установить флажок **Обеспечение целостности данных →** нажать кнопку **Объединение →** в раскрывшемся окне **Параметры объединения** установить переключатель в положение 2 → нажать кнопку **Создать**.

Замечание. Польза от связывания таблиц проявляется при создании форм или отчетов. В зависимости от выбора принципа объединения в окне **Параметры объединения** меняется объем выводимых данных при совместном использовании таблиц.

В рассмотренном примере выбран второй тип объединения. Это означает, что *Таблица4* признана главной и все данные из нее доступны при использовании в форме или отчете. *Таблица2* – это связанная таблица и из нее могут выбираться только данные, для которых установлено соответствие.

В *Таблице4* коды, соответствующие записям, не повторяются дважды. В связанной таблице один код может встречаться несколько раз. Поэтому созданная связь называется «один-ко-многим». Иными словами, одному коду в главной таблице соответствует много записей подчиненной таблицы. На **Схеме данных** эта связь выглядит, как показано на рис. 26, пример формы представлен на рис. 27.

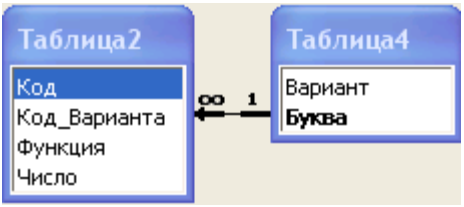


Рис. 26

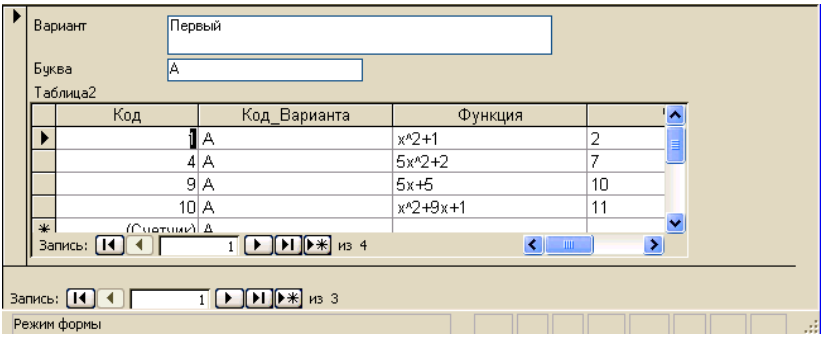


Рис. 27

Связь типа «один-к-одному»

Связь типа «один-к-одному» ставит в соответствие каждой строке одной таблицы единственную строку другой таблицы. При таком сопоставлении данные можно объединить в одну таблицу без ущерба для БД. Но с другой стороны, некоторые данные могут быть взяты из другой БД или модифицированы, а для этого их удобнее разместить в отдельной таблице. Такое размещение данных соответствует принципу модульности СУБД.

Для демонстрации организации связи «один-к-одному» создадим таблицу *Дополнительные задания* (рис. 28):

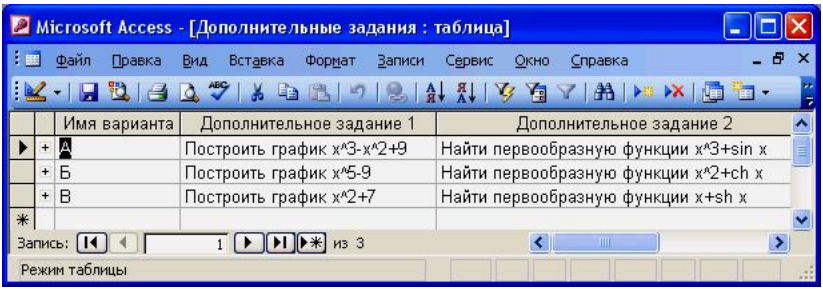


Рис. 28

Связь типа «один-к-одному» между *Таблицей4* и *Дополнительные задания* строится по алгоритму: открыть окно **Схема данных** → нажать кнопку **Отобразить таблицу** (или **Связи** → **Добавить таблицу**)→ выбрать таблицу *Дополнительные задания* → выделить поле **Код** в *Таблице4* и, не отпуская левую кнопку мыши, переместить

указатель на поле **Имя варианта** в таблице *Дополнительные задания* → в открывшемся окне **Связи** установить флажок **Обеспечение целостности данных** (программа определит связь типа «один-к-одному»), нажать кнопку **Объединение** → в раскрывшемся окне **Параметры объединения** установить переключатель в положение 2 → нажать кнопку **Создать**. Схема данных примет вид показанный на рис. 29.

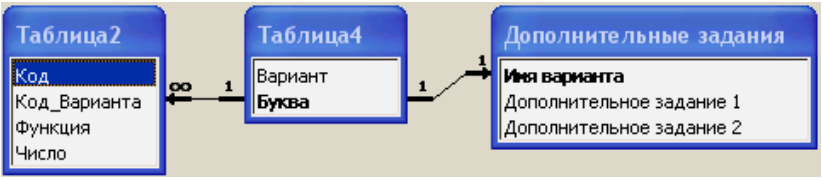


Рис. 29

Связь типа «многие-ко-многим»

При увеличении объема информации в БД количество таблиц возрастает, некоторые поля их могут содержать повторяющиеся значения. Ситуация часто складывается так, что именно по этим полям нужно связывать таблицы. Но попытка осуществит связь на прямую не увенчается успехом, так как тип связи «многие-ко-многим» программно не определен. К счастью, эта связь может быть представлена в виде двух связей типа «один-ко-многим» и некоторой промежуточной таблицей.

В качестве примера рассмотрим таблицу (рис. 30):

	Код1	Код варианта	Теоретический вопрос	Номер теста
▶	1	А	Теорема синусов	101
	2	А	Теорема косинусов	102
	3	Б	Теорема Пифагора	201
	4	Б	Бином Ньютона	202
	5	В	Теорема Фалеса	301
	6	В	Объем конуса	302
*	(Счетчик)			

Рис. 30

Промежуточная таблица должна содержать ключевое поле, которое используется для определения связи. Поэтому в качестве промежуточной составим следующую таблицу, в которой поле *Код варианта* является ключевым (рис. 31).

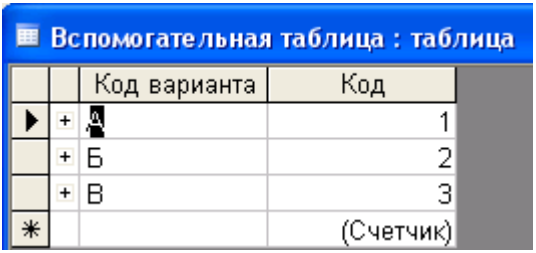


Рис. 31

Схема данных (рис. 32) показывает, как связь «многие-ко-многим» заменена двумя связями «один-ко-многим».

Замечание. Вспомогательная таблица может содержать два поля, через которые осуществляется связь.

Использование мастера форм позволяет быстро составить сводную таблицу, содержащую всевозможные комбинации записей (рис.33).

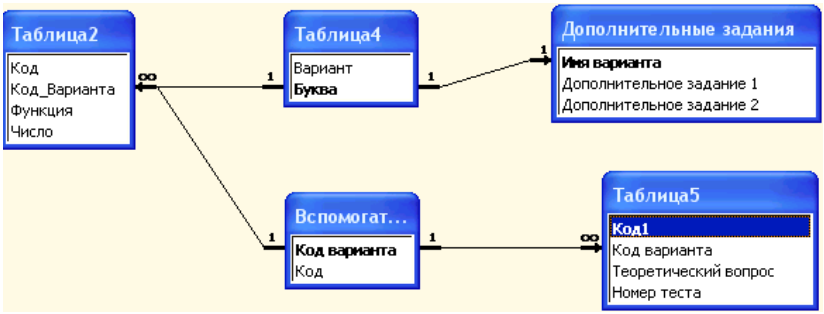


Рис. 32

Построение форм

Часто представление данных в виде таблиц может не устраивать пользователя. Кроме того, многие документы имеют определенный оригинальный вид, отличный от табличного. Поэтому придуман специальный механизм, названный ***формой***, позволяющий удовлетворить изысканным запросам пользователя.

На рис. 34 представлено окно, которое используется при построении формы.

Рис. 33

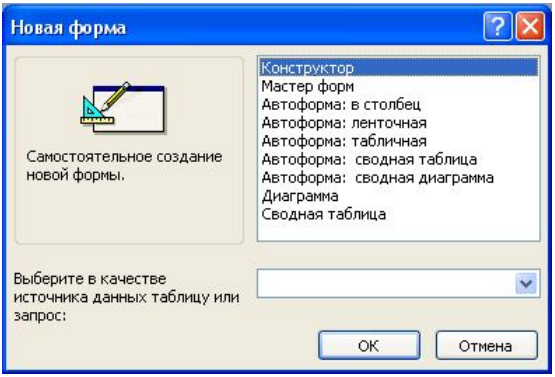


Рис. 34

Видно, что существует три стандартных вида автоформ: в столбец, ленточная и табличная. На рис. 35 и 36 последовательно показаны изображения первых двух автоформ на примере *Таблицы5*.

При просмотре этих форм в меню **Окно** появляется команда **По размеру формы**, позволяющая автоматически настраивать окно формы.

Табличная форма (рис. 37) имеет два режима: режим таблицы и режим формы.

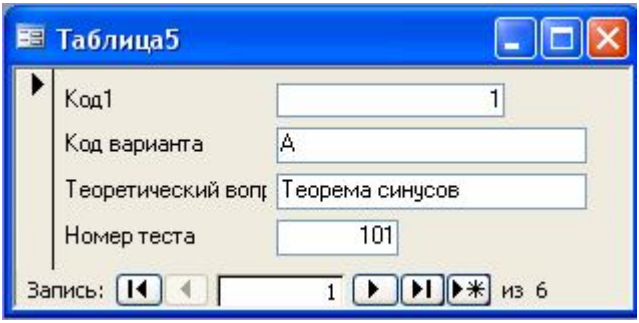


Рис. 35

Рис. 36

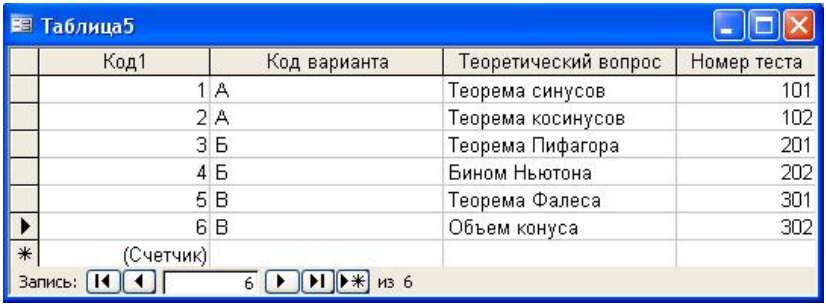


Рис. 37

Переключение между режимами происходит по команде: **Вид** → **Режим таблицы** (или **Режим формы**). Подчеркнем, что в режиме формы применяется команда **По размеру формы**.

Мастер форм

Форму можно также создать с помощью **Мастера форм**. Это делается в четыре шага: на первом шаге выбираются поля необходимые для построения формы → второй шаг предназначен для выбора вида

формы (в один столбец, ленточный, табличный) → третий шаг определяет стиль формы → четвертый шаг задает имя формы.

Замечание. Мастер позволяет быстро создать форму, но ее качество может не удовлетворять пользователя. В этом случае следует создать или доработать форму в режиме конструктора.

Конструктор формы

Для создания новой формы в режиме конструктора можно воспользоваться процедурой: в окне БД (см. рис. 1) нажать кнопку **Формы** панели **Объекты** → нажать кнопку **Создать** → в окне **Новая форма** выбрать пункт **Конструктор** → выбрать имя таблицы, для которой создается форма → **OK**.

Замечание. Если не выбирать пункт **Новая форма**, а сразу нажать кнопку конструктор, то список полей не появится. Для его отображения следует выполнит процедуру: **Вид → Свойства** → в открывшемся окне выбрать вкладку **Все** → в пункте **Источник записей** раскрыть список и выбрать имя нужной таблицы → после появления в окне списка полей заголовка и его заполнения закрыть окно свойств формы.

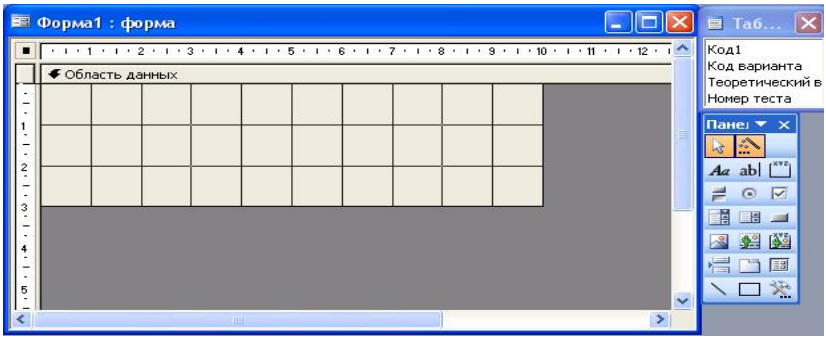


Рис. 38

С открытием окна конструктора разворачивается панель элементов и окно списка полей (рис. 38).

Замечание. Если в режиме конструктора не отобразилась панель элементов или список полей, то следует нажать кнопку **Панель элементов** и **Список полей** панели инструментов **Конструктор форм**.

Замечание. В форме целесообразно вначале разместить поля таблицы, в которые заносятся данные.

В MS Access *элементами управления* называют отдельные конструктивные элементы, которые размещаются в окне формы. При установке элемента в форму ему присваивается уникальное *имя элемента*

управления и он обладает набором свойств, просмотр и редактирование которых возможно через диалоговое окно свойств (Открывается при выделении элемента и выборе команды в контекстном меню).

Элементы управления делятся на три типа: *присоединенные*, *свободные* и *вычисляемые*.

Присоединенные элементы связаны с полем исходной таблицы и используются для отображения данных, ввода записей в таблицу и их обновления. Обычно полям таблицы соответствуют присоединенные элементы управления типа **Поле**. Для отображения полей логического типа применяют **переключатели**, **флажки** и **выключатели**. Все вышеуказанные элементы управления обычно применяются с соответствующими им подписями, которые соответствуют значению свойства элемента **Подпись**.

Свободные элементы управления независимы от источников данных и позволяют выводить на экран текст, линии, объекты OLE, которые содержатся не в таблице, а в самой форме.

Вычисляемые элементы не обновляют поля таблицы, но используют из нее данные для вычисления записей по некоторым формулам.

Элементы управления вставляются в форму путем выбора соответствующих кнопок на панели элементов. Описание некоторых из них представлено в табл. 1.

Таблица 1

Имя кнопки	Назначение
Выбор объектов	Позволяет использовать указатель мыши для выделения элемента управления, раздела или формы
Мастера	Включает и отключает мастера по созданию элементов управления
Надпись	Создает элемент управления, который отображает в форме некоторый поясняющий текст
Поле	Создает элемент управления **Поле** (вместе с присоединенной к нему надписью)
Группа переключателей	Создает элемент управления *Группа переключателей*. Элементы группы могут быть флажки, переключатели или выключатели
Выключатель	Создает элемент управления *Выключатель*, присоединяемый к логическому полю таблицы. Отображается в виде кнопки, которая в состоянии «включено» (утопленное положение) соответствует логическому ***Да***, а в состоянии «выключено» (приподнята) – логическому ***Нет***
Переключатель	Создает отдельный элемент управления *Переключатель*, аналогичный по своим функциям выключа-

	телю. Отображается в виде круглой кнопки и строки с поясняющим текстом. Если переключатель активизирован, то внутри кнопки появляется черная точка
Флажок	Создает элемент управления *Флажок*, аналогичный по своим функциям выключателю. Отображается в виде строки текста и маленького квадрата, в котором ставится галочка, если результат истинный, в случае ложного результата – квадрат пуст
Поле со списком	Создает элемент управления *Поле со списком*, в котором объединены поле для ввода значений и раскрывающийся список с заранее определенными значениями
Список	Создает раскрывающийся список допустимых опций выбора. Ввод других значений невозможен
Кнопка	Создает кнопку, нажатие которой активизирует выполнение определенных действий
Рисунок	Позволяет добавить в форму рисунок, который не является объектом OLE
Линия	Создает прямую линию, которая помогает визуально отделить родственные данные или разделы формы
Прямоугольник	Создает прямоугольник произвольного размера, который используется для создания графических эффектов

Элемент управления поле

Элементы управления, эквивалентные указанным в списке полям, помещаются в разделе формы, которая называется **Область данных**. Для добавления в форму элемента управления, соответствующего полю, следует выполнить следующие действия: в окне **Списка полей** выделить нужное поле и, не отпуская левую кнопку мыши, перетащить поле в раздел **Область данных формы** → отпустить кнопку мыши. После этой процедуры в форме появится элемент управления типа **Поле** с присоединенным заголовком. Указанную процедуру следует проделать для переноса в форму всех необходимых полей.

Когда все элементы управления размещены в форме необходимо на панели инструментов **Конструктор форм** нажать кнопку **Сохранить** → в открывшемся диалоговом окне **Сохранение** указать имя формы → нажать кнопку **OK.** Указанные действия сохраняют структуру формы.

Специальные элементы управления

С помощью *специальных элементов управления* обеспечивается возможность выбора вводимых значений из ограниченного набора данных. К *специальным элементам управления* относятся: *список, поле со списком, переключатели, флажки, выключатели, группа переключателей и кнопки.*

Список и поле со списком используется, когда заранее известно, что поле будет содержать значение из определенного множества вариантов. **Список** всегда отображается уже открытым и некоторая его часть уже видна на экране. Пользователь может выбрать только значения из представленного списка. **Поле со списком** выглядит как обычное поле формы, в правой части которого расположена кнопка со стрелкой. **Поле со списком** позволяет вводить значения отличные от предложенных вариантов.

Замечание. Преобразовать элемент **Поле со списком** на **Список**, и наоборот, можно, выделив в режиме конструктора нужный элемент и в контекстном меню выбрав команду **Преобразовать элемент в**, после чего указать новый тип элемента управления.

Создание указанных элементов управления происходит в соответствии с процедурой: активизировать кнопку **Мастера** на **панели элементов** → нажать кнопку **Поле со списком (Список) панели элементов** → поместить указатель в то место формы, где будет находиться выбранный элемент → нажать левую кнопку мыши и вычертить прямоугольник, определяющий границы элемента → отпустить левую кнопку мыши → в открывшемся окне мастера указать способ получения значений для нового элемента управления → нажать кнопку **Далее** → в третьем окне мастера следует указать, какие действия нужно предпринять программе после выбора значения из списка (например, **запомнить значение** или **Сохранить в поле**) → **Далее** → указать текст надписи для нового **поля со списком** или **списка** → **Готово**.

Группа переключателей

Для данных логического типа имеется три элемента управления: *переключатели*, *флажки* и *выключатели*. Каждый по отдельности из них возвращает значение типа *Да/Нет*. Если эти элементы поместить в **группу переключателей**, то они будут действовать согласованно. Иными словами, в группе может быть выбран один только элемент, числовое значение которого присваивается всей группе, и затем передается в поле таблицы.

Процедура создания **группы переключателей** состоит в следующем: активизировать кнопку **Мастера** на **панели элементов** → нажать

кнопку **Группа переключателей** → переместить указатель мыши в то место, где будет размещаться элемент управления → нажать левую кнопку мыши и, не отпуская ее, нарисовать прямоугольник, в котором будут размещаться элементы **группы переключателей** (не забудьте отпустить кнопку мыши после определения границ группы!) → мастер создания группы, запущен и в первом окне следует указать надписи для каждого переключателя → **Далее** → во втором окне мастера указывается переключатель, используемый по умолчанию → **Далее** → третье окно мастера предназначено для присвоения значений каждому элементу группы при их выборе (эти числовые значения присваиваются всей **Группе переключателей**, и следовательно, связанного с ним поля) → **Далее** → четвертое окно предназначено для указания действия, которое будет выполнено программой MS Access при выборе одного из элементов группы (при необходимости связать группу переключателей с полем таблицы следует отметить пункт **сохранить значение в поле**, при использовании значения присвоенного надписи, следует выбрать пункт **сохранить значение для дальнейшего применения**) → **Далее** → пятое окно мастера предназначено для выбора стиля оформления **группы переключателей** → **Далее** → в заключительном окне мастера вводится текст надписи для группы → **Готово**.

Подкорректировать свойства **группы переключателей** можно, открыв окно свойств этого элемента.

Элемент управления Кнопка

Для запуска конкретного процесса из формы применяется **элемент управления Кнопка**. Категории таких процессов представлены в табл.2.

Таблица 2

Категории	Действия
Переход по записям	Найти далее, найти запись, первая запись, последняя запись, предыдущая запись, последующая запись
Обработка записей	Позволяет восстановить, добавить дублировать, печатать, удалить или сохранить запись
Работа с формой	Можно закрыть форму, изменить фильтр формы, обновить данные, открыть страницу или форму, напечатать форму и применить фильтр формы
Работа с отчетом	Отправить отчет в файл или по почте, печатать или просмотреть отчет
Приложение	Можно выйти из MS Access, запустить Word или Excel или другое приложение

Разное	Запускается автонабор номера, выполняется запрос или макрос, печатается таблица

Создание ***элемента управления Кнопка*** происходит по следующей процедуре: активизировать кнопку **Мастера** на **панели элементов** → активизировать элемент **Кнопка** → отметить указателем место формы, где будет располагаться **Кнопка** (запустится **Мастер**) → в первом окне **Мастера создание кнопок** выбирается **Категория** и **Действие**, которое будет выполняться при нажатии кнопки → **Далее** во втором окне мастера определяется внешний вид кнопки → **Далее** → в заключительном окне указывается имя кнопки → **Готово**.

Подчиненная форма

При существовании между таблицами связей типа «один-к-одному» или «один-ко-многим» в форме могут одновременно размещаться зависимые данные из этих таблиц.

Для встраивания подчиненной формы следует запустить соответствующий мастер и выполнить ряд действий: активизировать кнопку **Мастера** на **панели элементов** → там же нажать кнопку **Подчиненная форма/Отчет** → в режиме конструктора нарисовать мышью область, в которой будет находиться подчиненная форма (запустится **Мастер подчиненных форм**) → первый шаг мастера служит для указания источника данных для новой формы (если форма уже есть, то следует выбрать ее имя, если это таблица, ее имя указывается на втором шаге мастера) → **Далее** → второй шаг служит для указания таблицы и полей подчиненной формы → **Далее** → третий шаг посвящен определению характеристик межтабличных связей → **Далее** → четвертый шаг задает имя для подчиненной формы → **Готово**.

На этом мы завершаем краткое введение в Microsoft MS Access. Нерассмотренными остались вопросы, связанные с поиском данных в БД, запросами, отчетами и многими другими важными аспектами СУБД. Но понятие СУБД у читателя данного пособия теперь имеется, как и возможность строить, вначале несложные, свои собственные базы данных.

Часть IV. Введение в программирование

Структурное программирование

С момента зарождения программирования было создано множество языков общения человека с ЭВМ. Сейчас, по видимости, наибольшее распространение имеют языки программирования C++, Delphi, Visual Basic (Visual Basic for Application) и некоторые другие. При этом в каждом из этих языков есть поддержка всех классических управляющих конструкций.

К 70-м годам XX века стало ясно, что программные проекты стали слишком сложными для успешного проектирования, кодирования и отладки в приемлемые сроки. Размер программ достиг величин, при которых программисты не могли с уверенностью сказать, что созданный программный продукт всегда выполняет то, что требуется, и что он не выполняет ничего такого, что не требуется. Назрела проблема изменения подходов к созданию больших программных продуктов.

В 1969 году Э. Дейкстра на международной конференции по программированию впервые использовал термин «структурное программирование» и предложил принципиально новый способ создания программ. Программа рассматривалась им как совокупность иерархических абстрактных уровней, которые позволяют четко структурировать программу, что позволяет лучше ее понимать, доказывать корректность ее работы и тем самым повышать надежность функционирования программы и сократить сроки ее разработки.

Правила структурной методологии разрабатывались такими учеными как Вирт, Дейкстра, Дал, Хоар, Иордан и другими. В этой связи следует отметить знаменитые книги Вирта [18, 19] и сборник [11].

Цели структурного программирования

I. *Обеспечить дисциплину программирования* в процессе создания программных комплексов. Дейкстра дал следующее определение:

«Структурное программирование – это дисциплина, которую программист навязывает сам себе».

II. *Улучшить читабельность программы*. Для этого следует избегать использования языковых конструкций с неочевидной семантикой; стремиться к локализации действия управляющих конструкций и использования структур данных; разрабатывать программу так, чтобы ее можно было читать от начала до конца без управляющих переходов на другую страницу.

III. *Повышать эффективность программы*. Данное положение достигается при структурировании программы, разбивании ее на модули так, чтобы можно было легко находить и корректировать ошибки, а также чтобы текст любого модуля с целью увеличения эффективности можно было переделать независимо от других.

IV. *Повышать надежность программы.* Надежность обеспечивается хорошим структурированием программы при разбивке ее на модули и выполнением правил написания читабельных программ, что ведет к возможности сквозного тестирования и не создает проблем для организации процесса отладки.

V. *Уменьшать время и стоимость программной разработки*. Этот пункт выполняется, когда повышается производительность труда программиста, чему способствуют правила структурного программирования.

Основные принципы структурной методологии

Принцип абстракции. Абстракция позволяет придумать решение задачи без сиюминутного учета множества деталей. Поэтому проблема может рассматриваться по уровням: верхний уровень показывает большую абстракцию, упрощает взгляд на проект, нижний – показывает мелкие детали реализации задачи.

Принцип формальности. Формальность позволяет перейти от импровизации к строгому инженерному подходу при написании программ. Более того, этот принцип дает основания для доказательства правильности программ, так как позволяет изучать алгоритмы как математические объекты.

Принцип «разделяй и властвуй». Этот принцип означает возможность разделения программы на отдельные модули, которые просты по управлению и допускают независимую отладку и тестирование.

Принцип иерархического упорядочения. Данный принцип тесно связан с принципом «разделяй и властвуй» и выдвигает требования иерархического структурирования взаимосвязей между модулями программного комплекса, что облегчает достижение структурного программирования.

Модульное программирование

Модульное программирование – это организация программы как совокупности небольших независимых блоков, называемых модулями, структура и поведение которых подчиняется определенным правилам. Первым основные свойства программного модуля сформулировал Парнас (Parnas): «Для написания одного модуля должно быть достаточно *минимальных* знаний о тексте другого». Но только в 1975 году Н. Виртом впервые была предложена специализированная синтаксическая конструкция модуля и включена в новый язык программирования **Modula.** Сейчас аналогичные конструкции имеются во многих языках.

Практические рекомендации

В [4] обсуждаются различные этапы и приемы программирования и разработки алгоритмов. Некоторые из них, применительно к обсуждаемым здесь вопросам сформулированы ниже.

1. *Никаких трюков и заумного программирования*. Не нужно применять сложных методов там, где можно использовать простые.
2. *Как можно меньше переходов*. Программирование без **go to** – это еще не структурное программирование. В некоторых ситуациях переход по **go to** является лаконичным, простым и ясным средством. Но разумное применение конструкций **if-then-else** и **for-do** способствует большей ясности, чем использование оператора **go to.**
3. Выбор с использованием конструкции **if-then-else**. Цель данного положения – обеспечение простоты хода вычисления, без каких то ни было переходов извне внутрь рассматриваемой структуры.
4. *Простота циклов.* Переходы по программе назад почти всегда можно представить как некоторую форму цикла. Существуют формы цикла **for-do**, **while-do**, **repeat-until**.

Можно написать

k:=0; **while** k<1000 **do begin** операторы; k:=k+1 **end**.

Но проще и более ясно записать так

for k:=0 **step** 1 **until** 999 **do** операторы.

Таким образом, каждое средство языка программирования следует применять по назначению.

5. *Сегментация*. Громоздкие программы, состоящие из одной основной программы, как правило, невозможно читать. В отсутствие конструкций **if-then-else** она будет перегружена метками и переходами и ее структура будет неясна. Если имеется возможность использования **if-then-else**, то глубина вложенности циклов может стать настолько большой, что отыскание для каждого **if-then** соответствующего **else** будет затруднительно.

Чтобы обойти указанные трудности, каждую большую программу можно разбить на множество модулей или процедур (подпрограмм или функций), спроектированных так, что цель для каждой из них определена (логическая часть исходной задачи), и в каждой по возможности используются собственные локальные переменные. Не нужно стремиться разделить программу на равные куски («куски» в этом случае – самое подходящее слово для того, что получится при делении программы на равные части), а следует выделять логические фрагменты. Некоторые из них окажутся достаточно малыми для того, чтобы их можно было в таком виде оставить. Другие же, в свою очередь, потребуют разбиения на процедуры следующего уровня.

6. *Рекурсия*. Все, что можно запрограммировать при помощи простого цикла, как правило, так и следует программировать. Но в некоторых ситуациях рекурсия оказывается естественной и понятной. Примером может служить программа вычисления факториала:

```
Function Nfact(j As Integer)
  Dim ii As Integer
  Dim fff As Long
    If j = 1 Then fff = 1 Else fff = j * Nfact(j - 1)
  Nfact = fff
End Function

Sub factorial()
Dim i, N As Integer
Dim F, FF As Long
F = 1
N = Worksheets("Factorial").Cells(1, 1).Value
For i = 1 To N
```

```
    F = F * i
Next i
Worksheets("Factorial").Cells(5, 1).Value = F
Worksheets("Factorial").Cells(5,5).Value= Nfact(N)
End Sub
```

7. *Содержательные обозначения.* Может показаться, что краткие имена ускоряют и упрощают написание программы. Но, когда через некоторое время программу потребуется модифицировать, простота понимания, обеспечиваемая содержательными обозначениями, полностью окупит затраченные при программировании усилия.

Очевидно, что к приведенным семи правилам можно добавить некоторые другие. Но, если придерживаться перечисленных правил, то можно писать программы, которые легко понимать и сопровождать.

Некоторые понятия об объектно-ориентированной методологии программирования

Следующим этапом развития науки программирования стало создание объектно-ориентированной методологии.

Объектно-ориентированная методология программирования преследует те же цели, что и структурная, но решает их с другой отправной точки. По определению Г. Буча [1], «объектно-ориентированное программирование – это методология программирования, которая основана на представлении программы в виде совокупности объектов, каждый из которых является реализацией определенного класса (типа), а классы (типы) образуют иерархию на принципах наследуемости».

Одним из принципов управления сложностью проекта является декомпозиция. Г. Буч выделяет две разновидности декомпозиции: алгоритмическую, которую поддерживают структурные методы, и объектно-ориентированную, отличие которой состоит в следующем: «Разделение по алгоритмам концентрирует внимание на порядке происходящих событий, а разделение по объектам придает особое значение факторам, либо вызывающим действия, либо являющимся объектами приложения этих действий». Другими словами, *алгоритмическая декомпозиция учитывает в большей степени структуру взаимосвязей между частями сложной проблемы, а объектно-ориентированная уделяет большее внимание характеру взаимодействий.*

На практике следует использовать обе разновидности декомпозиции. При создании крупных проектов сначала следует применить объ-

ектно-ориентированный подход для конструирования общей иерархии объектов, отражающих сущность программной задачи, а затем использовать алгоритмическую декомпозицию на модули для упрощения разработки, отладки и сопровождения программного комплекса.

Следует подчеркнуть [3], что в некоторых областях, таких как интерактивная графика, объектно-ориентированное программирование весьма полезно. В других задачах, таких как классические арифметические типы и вычисления, основанные на них, похоже трудно найти применение чему-то большему, чем абстракция данных, а средства, необходимые для поддержки объектно-ориентированного программирования, выглядят бесполезными.

Visual Basic for Application

Применение языка программирования VBA для решения некоторых задач диктуется желанием научить студентов создавать приложения для Microsoft Office, а VBA встроен в Word, Excel, MS Access и некоторые другие приложения. Отметим, что с появлением net – технологий эта задача может быть упрощена. Для углубленного изучения VBA можно порекомендовать книгу [7].

Сразу отметим, что ниже не дается всестороннее описание конструкций VBA, а рассматриваются примеры и даются небольшие комментарии к программам. Выбор примеров продиктован интересами авторов данного пособия и курсами, которые они читают на различных факультетах Санкт-Петербургского государственного университета.

Конструкции VBA

Описание переменных

Явное описание переменных позволяет избежать многих ошибок в коде программы. Поэтому предусмотрена команда **Option Explicit**, которая запрещает неявное объявление переменных, но действует только в модуле, в котором она появляется. Для того чтобы эта команда была в каждом модуле, следует выполнить действия: **Tools → Options →** вкладка **Editor →** установить флажок **Require Variable Declaration → OK**. Теперь требование явного объявления переменных действует во всех приложениях Microsoft Office, использующих VBA.

Для явного описания переменных используется оператор **Dim** со следующим синтаксисом: **Dim** *имя переменной* **As** *тип переменной*, [*имя переменной* **As** *тип переменной*]

Пример.

```
Dim n As Integer
Dim s As Double, R As Double
```

Следует помнить, что

1. Переменная, объявленная в процедуре, доступна только в этой конкретной процедуре.
2. Для того чтобы переменная была доступна во всех процедурах модуля, следует поместить объявление переменной в начале модуля перед объявлением процедур.

Во многих приложениях огромную роль играют совокупности переменных имеющий одинаковый математический или физический смысл. Такие переменные часто записывают в виде массивов, под которыми подразумеваются коллекции переменных с общим именем и одинаковым базовым типом. Объявление массива происходит с помощью оператора **Dim**:

Dim имя массива ([измерение массива]) **As** тип компонентов массива

Измерение массива представляет собой верхний и нижний индексы элементов массива, разделенные зарезервированным словом **to**.

Пример.

Объявление одномерного массива из 150 целых чисел

Dim A(1 **to** 150) **As** Integer

Объявление двухмерного массива, эквивалентного матрице 16x21

Dim B(0 **to** 15, 0 to 20) **As** Integer

Обращение к элементам массива происходит по имени, например, A(5) или B(8,7). VBA позволяет создавать массивы, имеющие до 60 измерений.

Арифметические операции

В VBA выполняются все обычные арифметические операции (см. табл. 3).

Таблица 3

Знак	Синтаксис	Описание
+	C+D	Сложение. A=C+D
-	C-D	Вычитание. A=C-D
*	C*D	Умножение. A=C*D
/	C/D	Деление. A=C/D
\	C\D	Целочисленное деление. Делит C на D и отбрасывает любую дробную часть так, чтобы результат был целым числом. A=C\D
Mod	C Mod D	Деление по модулю. Делит C на D, возвращая только остаток операции деления. A=C Mod D. Например, 8 Mod 2 возвращает 0, 5 Mod 3 возвращает 2
^	C^D	Возведение в степень. A=C^D

Операции сравнения

Операции сравнения используются для сравнения переменных значений любого сходного типа и возвращающие логическое значение **Thru** или **False** (табл. 4).

Таблица 4

Операция	**Синтаксис**	**Описание**
=	A=B	Равенство. **True.** Если A равно B, иначе **False**
<	A<B	Меньше, чем. **True**, если A меньше B, иначе **False**
<=	A<=B	Меньше, чем или равно. **True**, если A меньше или равно B, иначе **False**

>	A>B	Больше, чем. **True**, если A больше B, иначе **False**
>=	A>=B	Больше, чем или равно. **True**, если A больше или равно B, иначе **False**
<>	A<>B	Не равно. **True**, если A не равно B, иначе **False**

Логические операторы

Результат логической операции является значение типа Boolean (табл. 5).

Таблица 5

Оператор	Синтаксис	Название операции и ее описание
And	A and B	Конъюнкция. **True**, если A и B имеют значения **True**, иначе **False**
Or	A Or B	Дизъюнкция. **True**, если A, B или оба имеют значения **True**, иначе **False**
Not	Not A	Отрицание. **True**, если A имеет значение **False** и **False**, если A имеет значение **True**
Xor	A Xor B	Исключение. **True**, если A равно **Thru** или B равно **True**, иначе **False**
Eqv	A Eqv B	Эквивалентность. **True**, если A имеет то же значение, что и B, иначе **False**
Imp	A Imp B	Импликация. **False**, когда A является равным **True** и B равно **False**, иначе **True**

Приоритеты выполнения операций

При выполнении сложных выражений VBA следует правилам:

1. Части выражения, заключенные в круглые скобки, всегда выполняются в первую очередь.
2. Конкретные операции выполняются в зависимости от иерархии операторов (табл. 6).
3. Когда операторы имеют равный уровень приоритета, они вычисляются в порядке слева направо.

VBA вычисляет выражения в следующем порядке:

1. Знаки арифметических операций.
2. Знаки конкатенации строк.
3. Операторы сравнения.
4. Логические операторы.

Таблица 6

Оператор	Комментарии
^	Возведение в степень наивысший приоритет
-	Унарный минус
*,/	Умножение и деление имеют равные приоритеты
\	
Mod	
+,-	Сложение и вычитание имеют равный приоритет
&	Всякая конкатенация строк выполняется после любых арифметических операций в выражении и перед любыми операциями сравнения или логическими операциями
<,<=,>,>=,Like=,<>,is	Все операции сравнения имеют равные приоритеты
Not	
And	
Or	
Xor	
Eqv	
Imp	

Условный блок if-then-else

Часто решения математических, физических, экономических и других задач содержат ветвления в вычислительном алгоритме, которые происходят при некоторых условиях. Проверка этих условий осуществляется при помощи логических выражений. VBA в своем составе имеет несколько операторов для организации этих действий. Наиболее общим, по всей видимости, является *условный блок* со следующим синтаксисом:

If <*условие*> **then**
<*Операторы 1*>
else
<*Операторы 2*>
End If

Здесь <*условие*> – любое логическое выражение, <*Операторы1*>, <*Операторы 2*> – совокупности действий, первая из которых выполняется при принятии условием значения true, вторая – при выработке условием значения false. Ключевые слова **End If** указывают на закрытие условного оператора.

Организация циклов

В современных языках программирования обычно присутствуют три оператора для выполнения повторяющихся фрагментов программы. VBA их имеет несколько больше, но наиболее важными являются операторы циклов: **For, While** и **Do … Until**.

Оператор цикла ***For*** имеет синтаксис:

For <*параметр цикла*> = <*начальное значение*> **To** <*конечное значение*>

<*операторы*>

Next <*параметр цикла*>

Здесь <*параметр цикла*> – переменная типа integer, <*начальное значение*> и <*конечное значение*> – выражения того же типа, <*операторы*> – действия, которые повторяются в цикле.

Оператор цикла ***While*** имеет синтаксис:

Do While <*условие*>

<*операторы*>

Loop

Здесь <*условие*> – логическое выражение, <*операторы*> – действия, которые выполняются, если <*условие*> принимает значение **True**. Как только <*условие*> принимает значение **False**, происходит выход из цикла. Операторы, которые стоят внутри цикла, могут вообще не выполняться. Это произойдет, если <*условие*> примет сразу значение **False**.

Оператор цикла ***Do…Until*** с постпроверкой условия имеет синтаксис:

Do

<*тело цикла*>

Loop Until <условие>

Здесь **Do**, **Loop**, **Until** зарезервированные слова, <*тело цикла*> – последовательность операторов, которая выполняется при работе цикла, <условие> – логическое выражение. Если <условие> принимает значение **False**, то операторы, из которых состоит <*тело цикла*>, повторяются. В случае, когда

<условие> принимает значение **True**, цикл завершает свою работу. Подчеркнем, что <*тело цикла*> выполняется хотя бы один раз, и только потом проверяется условие прекращения работы цикла.

Замечание. В операторах циклов VBA можно заменить ключевое слово **While** на **Until** и наоборот. Этот выбор диктуется правилами:

1. Следует использовать **While**, если цикл продолжает выполняться, пока условие равно **True**.
2. Следует использовать **Until**, если цикл продолжает выполняться, пока условие равно **False**.

Реализация некоторых алгоритмов средствами VBA

Вычисление площади круга

Вход в редактор VBA осуществляется нажатием комбинации клавиш <Alt> + <F11>. Далее следует, исходя из сказанного выше об описании переменных, записать команду **Option Explicit.** Заметим, что всякая вычислительная задача в VBA оформляется как процедура, поэтому следует написать ключевое слово **Sub**, нажать клавишу пробел и ввести имя подпрограммы, после которого в круглых скобках разместить параметры, передающиеся в процедуру извне. Если никакие параметры не передаются, то в круглых скобках ничего не пишется. Нажатие клавиши <Enter> приводит к появлению строчки со словами **End Sub**, которые закрывают процедуру. Текст программы следует вводить между словами **Sub** и **End Sub**.

Замечание. Программы, приведенные ниже, написаны в VBA для Excel.

Первая задача будет наипростейшей. Необходимо вычислить площадь круга по формуле $S = \pi R^2$. Программа, приведенная ниже, запрашивает радиус круга и выдает значение площади.

```
Option Explicit
Sub my_program1()
Const pi = 3.1415926
Dim s, R As Double
Dim BoxTitle As String
BoxTitle = "input date"
R = InputBox("Input Radius", BoxTitle)
s = pi * R ^ 2
MsgBox  s, , "результат"
End Sub
```

Здесь вначале описана постоянная π (правила программирования требуют вначале описать постоянные, так как от них может зависеть описание переменных), затем вещественные s и R, BoxTitle – строковая переменная. Следующий оператор присваивает строковой переменной значение input date (строковые записи заключаются в кавычки). Переменной R мы присваиваем значение радиуса круга с помощью оператора **InputBox**. Параметрами этой функции являются обязательная подсказка, заключенное в кавычки строковое выражение, и необязательная строковая переменная, значение которой отображается в заголовке окна (рис. 39).

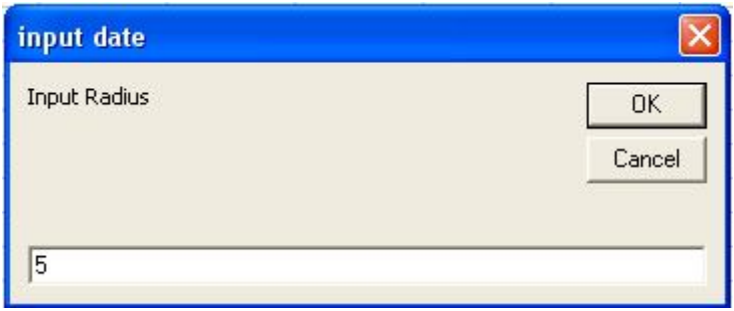


Рис. 39

После задания R, присваиваем переменной s значение площади. Вывод результата производится с помощью диалогового окна, вызываемого функцией **MsgBox** (рис. 40).

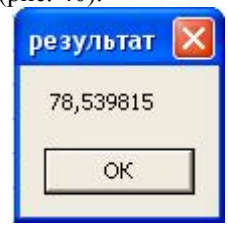


Рис. 40

Замечание. Полное описание функций, доступных пользователю, содержится в справке VBA. Там же присутствуют многочисленные примеры, показывающие применение этих функций.

Произведение матриц

Определение [8]. Пусть даны две матрицы

$$A=\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix} \text{ и } B=\begin{pmatrix} b_{11} & b_{12} & \dots & b_{1p} \\ b_{21} & b_{22} & \dots & b_{2p} \\ \dots & \dots & \dots & \dots \\ b_{n1} & b_{n2} & \dots & a_{np} \end{pmatrix},$$

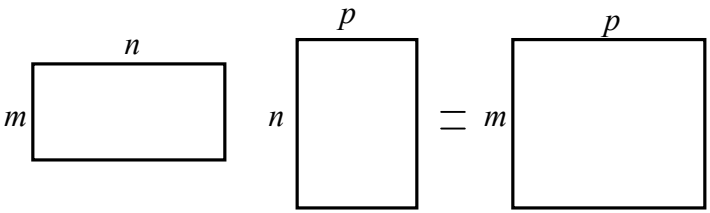
причем число столбцов матрицы A равно числу строк матрицы B. Произведением матрицы A на матрицу B называется матрица C:

$$C=\begin{pmatrix} c_{11} & c_{12} & \dots & c_{1p} \\ c_{21} & c_{22} & \dots & c_{2p} \\ \dots & \dots & \dots & \dots \\ c_{m1} & c_{m2} & \dots & c_{mp} \end{pmatrix},$$

обозначаемая через AB, элементы которой вычисляются по формулам

$$c_{ij}=\sum_{s=1}^{n} a_{is}b_{sj}\text{, где } i=1,2,\dots,m,\ j=1,2,\dots,p.$$

Еще раз отметим, что произведение двух матриц имеет смысл только при условии, которое графически выглядит так:



Программа перемножающая две матрицы представленные на листе Excel записывается так:

```
Option Explicit
Sub multiplcation_matrix()
Dim m, n, p As Integer
Dim A(1 To 20, 1 To 20) As Double
Dim B(1 To 20, 1 To 20) As Double
```

```
Dim C(1 To 20, 1 To 20) As Double
Dim i, j, k, ii, jj As Integer
Dim dd, ddddd, s As Double
m = Worksheets("произведение матриц").Cells(1, 1).Value
n = Worksheets("произведение матриц").Cells(1, 2).Value
p = Worksheets("произведение матриц").Cells(1, 3).Value

For i = 1 To m
  For j = 1 To n
    A(i, j) = Worksheets("произведение матриц").Cells(i + 2, j).Value
  Next j
Next i

For ii = 1 To n
  For jj = 1 To p
    B(ii, jj) = Worksheets("произведение матриц").Cells(m + 3 + ii,
jj).Value
  Next jj
Next ii

For i = 1 To m
  For j = 1 To p
    C(i, j) = 0
    For s = 1 To n
      C(i, j) = C(i, j) + A(i, s) * B(s, j)
    Next s
   Next j
Next i
For i = 1 To m
  For j = 1 To p
    Worksheets("произведение матриц").Cells(m + n + 5 + i, j).Value
= C(i, j)
  Next j
Next i
End Sub
```

Вычисление определителя квадратной матрицы

Во многих математических дисциплинах используется понятие определителя квадратной матрицы n-го порядка, под которым понимается *сумма всех* $n!$ *произведений элементов этой матрицы, взятых по одному из каждой строчки и по одному из каждого столбца; при этом каждое произведение снабжено знаком плюс или минус по некоторому правилу*.

Произведения, о которых говорится в определении, можно представить в виде

$$P = a_{1\alpha} a_{2\beta} \ldots a_{n\omega}. \qquad (1)$$

Первый индекс у сомножителя a_{ij} в произведении (1) соответствует номеру строки, второй – номеру столбца. Номера столбцов дают перестановку $(\alpha, \beta, \ldots, \omega)$. Если эта перестановка четная, то величина P, входящая в определение определителя, берется со знаком плюс, если нечетная – то со знаком минус.

Замечание. Непосредственное вычисление определителя по определению представляет собой трудоемкий процесс, так как нужно находить $n!$ произведений и определять знак каждого из них. Но можно заметить, что если все элементы, стоящие выше или ниже главной диагонали равны нулю, то определитель равен произведению элементов стоящих на этой диагонали: $a_{11} a_{22} \ldots a_{nn}$.

Полезными при вычислении определителей оказываются следующие их свойства:

- Определитель матрицы *n*-го порядка не изменится, если к элементам одной ее строки прибавить соответствующие элементы другой строчки, умноженной на одно и то же произвольное число.
- Если все элементы какой-нибудь строчки матрицы *n*-го порядка умножить на число c, то ее определитель также умножится на число c.

Из сделанного замечания и перечисленных свойств следует, что определитель целесообразно привести к треугольному виду и найти произведение элементов, стоящих на главной диагонали.

Программа, реализующая сказанное, имеет следующую запись:

Option Explicit

```
Sub determinant()
Dim n As Integer
Dim A(1 To 20, 1 To 20) As Double
Dim B(1 To 20, 1 To 20) As Double
Dim i, j, k, ii, jj As Integer
Dim dd, ddddd As Double

dd = 1

n = Worksheets("определитель").Cells(1, 1).Value
For i = 1 To n
  For j = 1 To n
    A(i, j) = Worksheets("определитель").Cells(i + 1, j).Value
  Next j
Next i

For i = 1 To n
  For j = 1 To n
    Worksheets("определитель").Cells(n + 2 + i + 1, j).Value = A(i, j)
  Next j
Next i

For ii = 1 To n
  For jj = 1 To n
    Worksheets("определитель").Cells(n + 2 + ii + 1, jj).Value = A(ii,
jj)
  Next jj
Next ii

dd = 1

For k = 1 To n - 1
  For ii = 1 To k
  For jj = 1 To n
    B(ii, jj) = A(ii, jj)
    Next jj
   Next ii

  For i = k + 1 To n
    For j = k To n
```

```
        B(i, j) = A(k, j) * A(i, k) - A(i, j) * A(k, k)
      Next j
      dd = dd * A(k, k)
    Next i

For ii = 1 To n
  For jj = 1 To n
    A(ii, jj) = B(ii, jj)
  Next jj
Next ii

Next k

For ii = 1 To n
  For jj = 1 To n
    Worksheets("определитель").Cells(n + 2 + ii + 1, jj).Value = B(ii,
jj)
  Next jj
Next ii

ddddd = 1
For k = 1 To n
ddddd = ddddd * B(k, k)
Next k
Worksheets("определитель").Cells(1, 9).Value = ddddd / dd
End Sub
```

Метод Гаусса решения системы линейных алгебраических уравнений

Пусть дана система линейных алгебраических уравнений, которая в матричной записи имеет вид

$$\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}. \qquad (2)$$

Составим расширенную матрицу

$$\begin{matrix} a_{11} & a_{12} & \dots & a_{1n} & b_1 \\ a_{21} & a_{22} & \dots & a_{2n} & b_2 \\ \dots & \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} & b_n \end{matrix}. \quad (3)$$

Определение. Под элементарными преобразованиями системы алгебраических уравнений понимаются следующие операции:

1) Умножение какого-либо уравнения системы на число, отличное от нуля.

2) Прибавление к одному уравнению другого уравнения, умноженного на произвольное число.

3) Перемена местами двух уравнений в системе.

В курсе алгебры доказывается теорема: при элементарных преобразованиях система переходит в равносильную систему (две системы линейных уравнений называются равносильными, если каждое решение одной из них является решением другой и наоборот).

Метод Гаусса решения системы линейных уравнений состоит в том, что при помощи элементарных преобразований систему приводят к такому виду, чтобы ее расширенная матрица из коэффициентов оказалась трапециевидной. После чего решение системы становится значительно проще.

Программа, решающая систему линейных алгебраических уравнений записывается так:

```
Sub Gaus()
Dim A(1 To 20, 1 To 21) As Double, X(1 To 20) As Double, _
    N As Integer, i As Integer, j As Integer, _
    n1 As Integer

N = Worksheets(1).Cells(1, 1).Value

For i = 1 To N
  For j = 1 To N + 1
    A(i, j) = Worksheets(1).Cells(i + 1, j).Value
  Next j
Next i

n1 = N + 1
For k = 1 To N
  k1 = k + 1
```

```
 s = A(k, k)
 j = k
  For i = k1 To N
   r = A(i, k)
   If Abs(r) > Abs(s) Then s = r:  j = i
  Next i
  If s = 0 Then MsgBox ("det=0 Решений нет")
  If j = k Then GoTo 1
    For i = k To n1
     r = A(k, i):  A(k, i) = A(j, i): A(j, i) = r
    Next i

For j = k1 To n1
    A(k, j) = A(k, j) / s
  Next j
  For i = k1 To N
   r = A(i, k)
    For j = k1 To n1
     A(i, j) = A(i, j) - A(k, j) * r
    Next j
  Next i
Next k
For i = N To 1 Step -1
 s = A(i, n1)
 For j = i + 1 To N
  s = s - A(i, j) * X(j)
 Next j
X(i) = s
Next i

For i = 1 To N
 Worksheets(1).Cells(i + 1, 8).Value = X(i)
Next i
  For j = 1 To N
  Worksheets(1).Cells(j + 1, 10).Value = 0

    For k = 1 To N
      Worksheets(1).Cells(j + 1, 10).Value = _
      Worksheets(1).Cells(j + 1, 10).Value + _
      Worksheets(1).Cells(j + 1, k).Value * X(k)
```

```
    Next k
  Next j

End Sub
```

Сортировка

Задача сортировки множества формулируется следующим образом [9]: задано конечное множество A, состоящее из n элементов, а на нем задано отношение линейного порядка P. Требуется перенумеровать элементы A числами от 1 до n таким образом, чтобы из неравенства $i < j$ следовало $(a_i, a_j) \in P$.

Обычно задача сортировки сводится к упорядочению чисел по значению.

Сортировка вставкой

Один из простейших способов сортировки состоит в следующем:

1) последовательность из одного элемента множества уже «отсортирована»;

2) берем следующий элемент и сравниваем его с предыдущим, при этом переставляем его вперед до тех пор, пока он не займет свое место.

```
Option Explicit

Sub Inset()
Dim i, j, k, N As Integer
Dim b As Double
N = Worksheets("сортировка вставкой").Cells(1, 1).Value
ReDim A(1 To N) As Double
'read array
For i = 1 To N
  A(i) = Worksheets("сортировка вставкой").Cells(2, i).Value
Next i

For i = 2 To N
  b = A(i)
 j = 1
  Do While b > A(j)
    j = j + 1
```

```
  Loop
  For k = i - 1 To j Step -1
    A(k + 1) = A(k)
  Next k
  A(j) = b
Next i
For i = 1 To N
  Worksheets("сортировка вставкой").Cells(5, i).Value = A(i)
Next i
End Sub
```

Сортировка выбором

Принцип метода:

1) выбираем в массиве элемент с минимальным значением на интервале от 1-го до n-го элемента и меняем его местами с первым элементом;

2) находим элемент с минимальным значением на интервале от 2-го до n-го элемента и меняем его местами со вторым элементом и так далее до $(n-1)$-го элемента.

```
Option Explicit
Sub sort2()
Dim i, N, imin, s As Integer
Dim min As Double
N = Worksheets("сортировка выбором").Cells(1, 1).Value
ReDim A(1 To N) As Double
'read array
For i = 1 To N
  A(i) = Worksheets("сортировка выбором").Cells(2, i).Value
Next i
For s = 1 To N - 1
  min = A(s)
  imin = s
  For i = s + 1 To N
    If A(i) < min Then min = A(i): imin = i
  Next i
  A(imin) = A(s)
  A(s) = min
Next s
For i = 1 To N
  Worksheets("сортировка выбором").Cells(5, i).Value = A(i)
```

```
Next i
End Sub
```

Сортировка обменом («пузырьковая» сортировка)

Содержание метода:

1) слева направо до конца массива поочередно сравниваются два соседних элемента, и если их взаиморасположение не соответствует условию упорядоченности, то они меняются местами;

2) после первого прохода на n-м месте стоит элемент с максимальным значением («всплыл» первый пузырек). Второй проход выполняется до $(n-1)$-го элемента, так как n-й уже на своем месте. Всего требуется $n-1$ проходов.

```
Option Explicit

Sub sort2()
Dim i, N, imin, s As Integer
Dim min As Double
N = Worksheets("сортировка выбором").Cells(1, 1).Value
ReDim A(1 To N) As Double
'read array
For i = 1 To N
  A(i) = Worksheets("сортировка выбором").Cells(2, i).Value
Next i
For s = 1 To N - 1
  min = A(s)
  imin = s
  For i = s + 1 To N
    If A(i) < min Then min = A(i): imin = i
  Next i
  A(imin) = A(s)
  A(s) = min
Next s
For i = 1 To N
  Worksheets("сортировка выбором").Cells(5, i).Value = A(i)
Next i
End Sub
```

Метод фон Неймана

Пусть даны два упорядоченных массива. Возьмем из них первые элементы и выберем тот, который должен следовать вначале в соответствии с некоторым условием, и запишем его в массив результата. Изменим рассмотрение индексов в массиве, из которого взят элемент. Применим изложенную процедуру к оставшимся массивам. Если один из массивов исчерпан, то остаток другого просто дописывается в результирующий массив. Описанная процедура называется *слиянием массивов*.

Дж. фон Нейман предложил метод сортировки, основанный на операции слияния:

1) исходный массив из n элементов вначале представляется в виде n «отсортированных» массивов, имеющих длину равную 1;

2) сольем эти массивы попарно, что приведет к появлению массивов длины 2;

3) повторим процедуру слияния, получая массивы длиной 4 и так далее. В итоге получается отсортированный массив длиной n.

Замечание. Недостатком этого метода является то, что для почти отсортированного массива тратится столько же времени, как и на самый «перемешанный» массив.

Еще один метод сортировки будет изложен в разделе Рекурсия.

Рекурсия

Рекурсивным называется объект, который частично определяется через самого себя. В программировании рекурсивная функция или процедура – это функция или процедура, которая вызывает сама себя.

Классическим примером сказанного является определения функции факториала. С одной стороны, факториал определяется следующей формулой:

$$n! = 1 \cdot 2 \cdot \ldots \cdot n,$$

с другой – рекуррентными соотношениями

I. $0! = 1$

II. для $\forall n > 0 \quad n! = n \cdot (n-1)!$

Другим примером может служить определение чисел Фибоначчи

I. $F(1) = 1$

II. $F(2) = 1$

III. для $\forall n > 2 \quad F(n) = F(n-1) + F(n-2)$

Программы, содержащие рекурсивные процедуры, наглядны и содержат компактный код. Однако использование рекурсивных процедур требует большого размера оперативной памяти при выполнении кода, чем нерекурсивные процедуры. Это вызвано тем, что при каждом рекурсивном вызове для параметров процедуры и локальных переменных выделяются новые ячейки памяти.

Определение. Глубиной рекурсии называется максимальное число рекурсивных вызовов, которое происходит во время выполнения программы.

Определение. Текущим уровнем рекурсии называется число рекурсивных вызовов в каждый конкретный момент времени.

Формы рекурсивных процедур

Любая рекурсивная процедура включает в себя некоторое множество операторов S и один или несколько операторов рекурсивного вызова P. Следует особо подчеркнуть, что безусловные рекурсивные процедуры приводят к бесконечным процессам. Еще раз напомним, что каждая копия рекурсивной процедуры требует выделения дополнительной памяти, но память в любом устройстве конечна!

Таким образом, вызов рекурсивной процедуры должен выполняться по условию, которое на каком-то уровне рекурсии станет ложным.

Если условие истинно, то рекурсивный спуск продолжается. В момент принятием условием ложного значения спуск прекращается и начинается поочередный рекурсивный возврат из всех вызванных на данный момент копий рекурсивной процедуры.

Форма с выполнением действий до рекурсивного вызова (действия выполняются на рекурсивном спуске)

```
Sub P()
  S
  if <условие> then P()
End Sub
```

Форма с выполнением действий после рекурсивного вызова (действия выполняются на рекурсивном возврате)

```
Sub P ()
  if <условие> then P()
```

```
  S
End Sub
```

Форма с выполнением действий как до, так и после рекурсивного вызова

```
Sub P ()
  S1
 if <условие> then P()
  S2
End Sub
```
или
```
Sub P ()
 if <условие> then
            S1
            P()
            S2
End Sub
```

Многие задачи безразличны к выбору формы рекурсивной процедуры. Но существуют целые классы задач, при решении которых требуется сознательно управлять ходом работы рекурсивных процедур и функций. К таким задачам, в частности, относится обработка списков и древовидных структур данных.

Выполнение действий на рекурсивном спуске

Для реализации алгоритма вычисления факториала, работающего на спуске, вводим параметры: *mult* для выполнения на спуске операции умножения накапливаемого значения факториала на очередной множитель, *m* для обеспечения независимости рекурсивной функции от имени конкретной глобальной переменной.

```
Function Fact1(mult As Long, i, m As Integer) As Long
  mult = mult * i
  If i = m Then Fact1 = mult Else Fact1 = Fact1(mult, i + 1, m)
End Function
Sub FF1()
Dim N As Integer
```

```
N = Worksheets("Factorial").Cells(1, 1).Value
Worksheets("Factorial").Cells(15, 1).Value = Fact1(1, 1, N)
End Sub
```

Выполнение действий на рекурсивном возврате

Ниже представлена программа, в которой вычисление факториала происходит на рекурсивном возврате, и при этом рекурсивный вызов и оператор накопления факториала разделены явным образом.

```
Function Fact2(i As Integer) As Long
Dim mult As Long
If i = 1 Then mult = 1 Else mult = Fact2(i - 1)
Fact2 = mult * i
End Function
Sub FF2()
Dim N As Integer
N = Worksheets("Factorial").Cells(1, 1).Value
Worksheets("Factorial").Cells(16, 1).Value = Fact2(N)
End Sub
```

Быстрая сортировка (метод Quicksort)

Английский математик, профессор Оксфордского университета Хоар (Charles Antony Richard Hoare), предложил в 1962 году метод основанный на следующей идее: взять один из элементов массива и поставить его таким образом, что элементы предшествующие ему в упорядочении, размещались до него, а следующие за ним – после. После этой операции исходный массив разбит на две части, которые можно упорядочить тем же способом.

Реализуем сказанное с помощью рекурсии [2]:

1. Выбираем центральный элемент массива и записываем его в переменную B. Затем элементы массива просматриваются поочередно слева направо и справа налево. При движении слева направо ищем элемент $A(i)$, который будет больше или равен B, и запоминаем его позицию. При движении справа налево ищем элемент $A(j)$, который меньше или равен B, и запоминаем его позицию. Найденные элементы меняем местами и продолжаем встречный поиск по указанным условиям. Этот процесс продолжаем, пока при очередной итерации поиска встречные индексы i и j не пересекутся. Таким образом, относи-

тельно значения B массив получается отсортированным. Остается упорядочить левую и правую части массива.

2. Описанную в предыдущем пункте процедуру повторяем для левой и правой частей по отдельности. После чего массив оказывается разбитым уже на четыре непересекающихся по сортировке части, которые можно упорядочить по отдельности. Процесс продолжается пока длина сортируемых частей не станет равной одному элементу и, следовательно, все элементы массива упорядочены.

Так как на каждом этапе выполняемые действия одни и те же, но в разных индексных промежутках, то удобно воспользоваться рекурсией. Программа Quicksort имеет вид:

```
Option Base 1
Option Explicit
Sub QS(A() As Double, L, R As Integer)
Dim i, j, B, tmp, LR As Integer

LR = (L + R) \ 2
B = A(LR)
i = L: j = R
Do While i <= j
  Do While A(i) < B: i = i + 1: Loop
  Do While A(j) > B: j = j - 1: Loop
    If i <= j Then
      tmp = A(i): A(i) = A(j): A(j) = tmp: i = i + 1: j = j - 1
    End If
Loop
If L < j Then Call QS(A, L, j + 1)
If i < R Then Call QS(A, i, R)
End Sub

Sub qqss()
Dim k, N As Integer
Dim A(1 To 20) As Double
N = 10
For k = 1 To N
 A(k) = Worksheets("Быстрая сортировка").Cells(1, k).Value
Next k
Call QS(A, 1, N)
For k = 1 To N
```

```
 Worksheets("Быстрая сортировка").Cells(2, k).Value = A(k)
Next k
End Sub
```

Решение задач механики

В этом разделе покажем, что VBA может быть применен для решения задач механического характера. При этом, размещая результаты расчетов в таблице MS Excel, мы получаем возможность графического отображения, полученных зависимостей.

Движение точки по инерции под действием силы трения

В соответствии со вторым законом Ньютона

$$m\mathbf{w} = \mathbf{F}$$

и законом Кулона

$$\mathrm{T} = -\mu N \frac{\mathbf{v}}{v}$$

уравнение движения точки по горизонтальной плоскости будет иметь вид (здесь μ – коэффициент трения, $\mathbf{v}$ – вектор скорости, v – величина скорости)

$$m\mathbf{w} = -\mu N \frac{\mathbf{v}}{v},$$

и после некоторых преобразований получаем дифференциальное уравнение

$$\ddot{x} = -\mu g. \qquad (1)$$

Аналитическое решение этого уравнения при начальных условиях $x(0) = 0$, $\dot{x}(0) = v_0$ имеет вид

$$x(t) = v_0 t - \frac{1}{2}\mu g t^2,\ v = v_0 - \mu g t. \qquad (2)$$

Откуда следует, что скольжение будет продолжаться в течении $t_* = \dfrac{v_0}{\mu g}$ с.

Получим численное решение уравнения (1) с помощью метода Рунге-Кутта 4-го порядка с постоянным шагом интегрирования. Реализация этого метода происходит по формулам [10]

$$k_1 = f(t_n, y_n), \qquad k_2 = f\left(t_n + \frac{\tau}{2}, y_n + \frac{\tau k_1}{2}\right),$$

$$k_3 = f\left(t_n + \frac{\tau}{2}, y_n + \frac{\tau k_2}{2}\right), \; k_2 = f\left(t_n + \tau, y_n + \tau k_3\right) \qquad (3)$$

$$y_{n+1} = y_n + \frac{\tau}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right).$$

Следует подчеркнуть, что величины $k_1, k_2, k_3, k_4, f, y_n$ и y_{n+1} являются векторами, а τ -скалярная величина, характеризующая шаг интегрирования по времени.

Для применения формул (3) уравнение (1) следует привести к нормальному виду:

$$\begin{aligned} \dot{x} &= v \\ \dot{v} &= -\mu g \end{aligned} \quad \text{или} \quad \begin{pmatrix} \dot{x} \\ \dot{v} \end{pmatrix} = \begin{pmatrix} v \\ -\mu g \end{pmatrix}.$$

Программа, выводящая в первый столбец Excel листа время, во второй – значение скорости, а в третий – координату, записывается так:

```
Option Explicit
Const NN = 2
Sub prav(Time As Double, Y() As Double, YY() As Double)
Dim x1 As Double, F As Double

x1 = Y(2)
F = -9.81 * 0.2
YY(1) = x1
YY(2) = F
End Sub
Sub RK4(T As Double, Tau As Double, Yrk4() As Double)
'Tau-STEP OF INTEGRATION;
 'T-TIME;
 'Y-vector begin value for one step
'YY - resalt prav
Dim i As Integer
Dim K(1 To NN) As Double, K1(1 To NN) As Double, K2(1 To NN) As
Double
Dim K3(1 To NN) As Double, K4(1 To NN) As Double
```

```
Call prav(T, Yrk4, K1)
For i = 1 To NN
  K(i) = Yrk4(i) + K1(i) * Tau / 2
Next i
T = T + Tau / 2
Call prav(T, K, K2)
For i = 1 To NN
  K(i) = Yrk4(i) + K2(i) * Tau / 2
Next i
Call prav(T, K, K3)
For i = 1 To NN
  K(i) = Yrk4(i) + K3(i) * Tau
Next i
T = T + Tau / 2
Call prav(T, K, K4)
For i = 1 To NN
  Yrk4(i) = Yrk4(i) + Tau / 6 * (K1(i) + 2 * (K2(i) + K3(i)) + K4(i))
Next i

End Sub
Sub rung(N As Integer, A As Double, B As Double, H As Double, TH
As Double, T As Double, Y() As Double)

Dim THP As Double
Dim i As Integer, j As Integer

T = A
THP = A + TH
j = 2

Worksheets("RK4").Cells(j, 1).Value = T
For i = 1 To NN
   Worksheets("RK4").Cells(j, i + 1).Value = Y(i)
Next i

Do While (T <= B)

Call RK4(T, H, Y)

  If T >= THP Then
```

```
        THP = THP + TH
          j = j + 1
            Worksheets("RK4").Cells(j, 1).Value = T
        For i = 1 To NN
            Worksheets("RK4").Cells(j, i + 1).Value = Y(i)
        Next i
      End If

    T = T + H
    Loop
    j = j + 1
            Worksheets("RK4").Cells(j, 1).Value = T
        For i = 1 To NN
            Worksheets("RK4").Cells(j, i + 1).Value = Y(i)
        Next i

    End Sub
    Sub solv()
    Dim Y(1 To 2) As Double
    Y(1) = 0: Y(2) = 1
    Call rung(NN, 0, 1.0, 0.0001, 0.1, 0, Y)
    End Sub
```

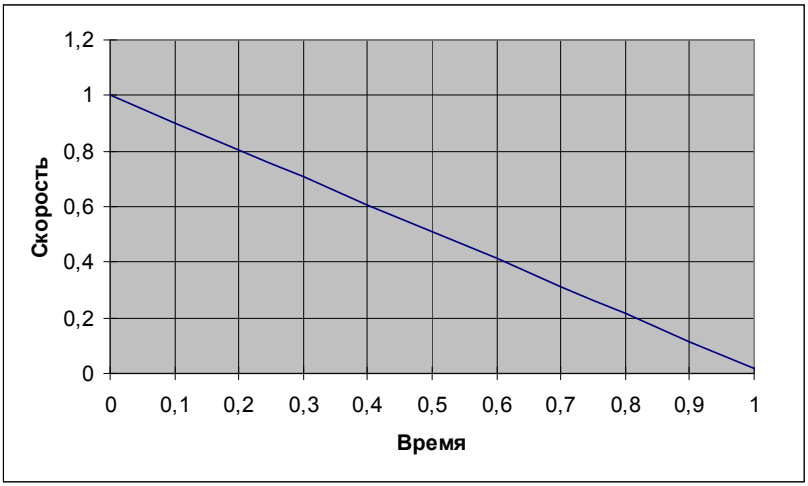
Отметим, что интегрирование происходит с шагом τ , но вывод в ячейки листа с шагом HT.

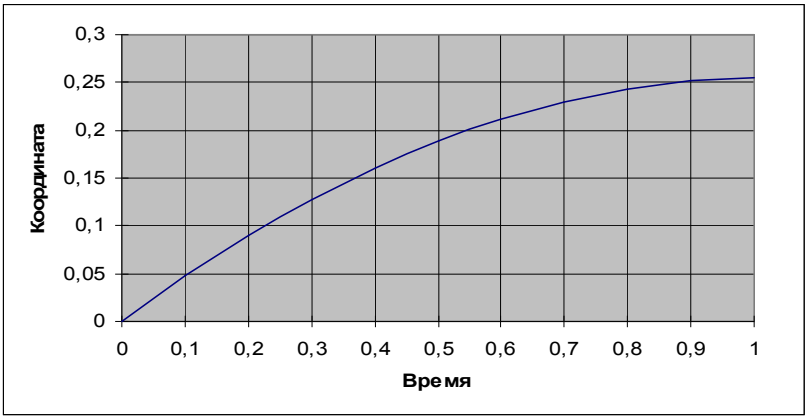
После получения таблицы с результатами интегрирования

0	0	1
	0,04	0,90
0,1	7638	1704
	0,09	0,80
0,2	027	3604
	0,12	0,70
0,3	7998	5504
	0,16	0,60
0,4	0821	7404
	0,18	0,50
0,5	8738	9304
	0,21	0,41
0,6	1751	1204

	0,22	0,31
0,7	9859	3104
	0,24	0,21
0,8	3062	5004
	0,25	0,11
0,9	1359	6904
	0,25	0,01
1	4752	8804

строим графики $v(t)$, $x(t)$:





Движение точки под действием восстанавливающей силы

Пусть точка массой m может перемещаться вдоль оси X. Предположим, что на точку действует восстанавливающая сила $-cx$.В соответствии с вышеизложенным имеем

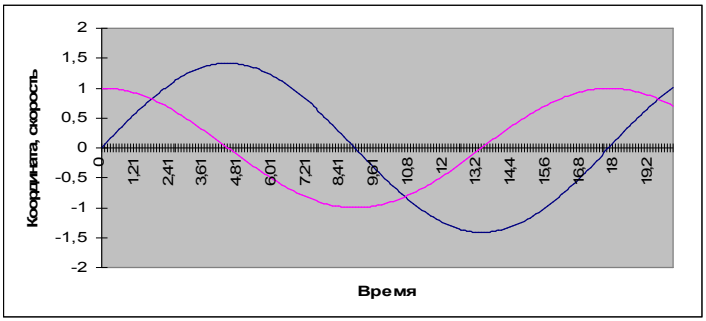
$$m\ddot{x} = -cx$$

или

$$\ddot{x} + k^2 x = 0 ,$$

где $k^2 = \frac{c}{m}$.

Решение данного уравнения для $m = 1$, $c = 0.5$ представлено на следующем графике (начальные условия $x(0) = 0$, $\dot{x}(0) = 1$):



На этом краткое описание реализации некоторых алгоритмов с помощью средств VBA мы завершаем.

Часть V. Программирование в Turbo Delphi и Free Pascal

Введение

В настоящее время существуют среды разработки программ, которые распространяются свободно (бесплатно). Для некоторых из них требуется регистрация, другие после закачки с соответствующего сайта сразу могут быть установлены на компьютер. Отметим интересующие нас источники: Free Pascal (), Turbo Delphi ().

Ниже приведены некоторые начальные сведения о программировании на языке Pascal, являющимся основой Free Pascal и Turbo Delphi. Особенности этих сред разработки здесь затрагиваться не будут. Но изучение нижеизложенного позволяет приступить к написанию своих собственных, вначале несложных, приложений.

Словарь языка Pascal

В любом языке программирования имеются символы и слова, которые компьютер понимает изначально. **Эти значки и слова нельзя употреблять как имена переменных**. Имена переменных часто называют идентификаторами. **Идентификатор –** это последовательность букв и цифр, начинающаяся с буквы.

Замечание. Пробелы в идентификаторах не допускаются!

Ограничители и специальные символы:

+	:=	:	<	(	..
-	.	‘	<=	)	{
*	,	=	>	[	}
/	;	<>	>=	]	^

Ключевые слова:

Absolute	And	Array	Asm
Begin	Case	Const	Constructor
Destructor	Div	Do	Downto
Else	End	External	File
For	Forward	Function	Goto
If	Implementation	In	Inline
Interface	Interrupt	Label	Mod
Nil	Not	Object	Of
Or	Packed	Procedure	Program
Record	Repeat	Set	Shl
Shr	String	Then	To
Type	Unit	Until	Uses
Var	Virtual	While	with
Xor			

Данные

Всякая программа предназначена для обработки данных. Эти данные могут быть различной природы (числа, тексты, последовательности двоичных разрядов или битов и т.д.). В зависимости от способа их хранения и обработки, данные можно разбить на две группы: *константы* и *переменные*.

Константы – это данные, которые не изменяются в процессе работы программы. *Переменные* – это данные, которые могут изменяться во время выполнения программы.

В Pascal константы могут быть трех видов: числовые, булевские (логические) и символьные.

Описание констант:

Const a=5; d=’ABC’; L=True;

Каждая переменная должна быть объявлена до своего первого применения. Тип переменной определяет множество допустимых для нее значений. Приведем некоторые стандартные типы данных:

Тип данных	**Диапазон**
Integer	-32768..32767
Byte	0..255
Word	0..65535
Real	По абсолютной величине 2.9E-39..1.7E+38
Double	5E-324..1.7E+308

Boolean	False, True
Char	Множество символов ASCII
String	Строка символов (до 255 символов)

Имеются и другие типы данных.

Пример описания переменных:

Var
A,B**:real;**
S**:string;**
S1,S2**:string;**
I,j,k**:integer;**
AA, BB**: double;**

Структура программы

Программы, написанные на языке программирования Pascal и в основанных на нем средах разработки, имеют четкую структуру: программа состоит из заголовка и блока. Заголовок — это ключевое слово **Program** и имя программы, после которых ставится точка с запятой. Например:

Program P;

Замечание. Заголовок программы не исполняемый оператор, то есть рассматривается компьютером как комментарий.

Блок программы состоит из шести разделов:

1) раздел меток (**Label**);
2) раздел констант (**Const**);
3) раздел типов (**Type**);
4) раздел переменных (**Var**);
5) раздел процедур и функций (**Procedure**, **Function**);
6) раздел действий.

Замечание. Раздел действий должен присутствовать всегда, остальные могут отсутствовать.

Первые четыре раздела начинаются с соответствующих ключевых слов (**Label, Const, Type, Var**).

Раздел меток (Label)

Любой выполняемый оператор может быть снабжен меткой – целочисленным числом от 0 до 9999 или идентификатором. Каждая описанная метка должна появиться в программе, причем ровно один раз. Метка отделяется от оператора двоеточием.

Пример:

```
Label L1, 25;
……
  L1: log:=true;
  25: a:=b;
……
```

Раздел констант (Const)

Если в программе используются константы (не обязательно числа), то удобно их обозначить каким-либо именем. И далее обращаться к этой постоянной величине по имени. Отметим, что при построении выражения для определения значения констант можно использовать только ранее описанные константы, соединенные знаками операций и функциями (Abs, Chr, Hi, Length, Odd, Ord, Round, Trunc и некоторые другие). Имеются предопределенные константы

Константа	Тип
Maxint=32767	**Integer**
MaxLongInt=2147483647	**LongInt**
False, True	**Boolean**
Pi=3.1415926	**Real**

Раздел типов (Type)

Если в программе вводится тип, отличный от стандартного, то этот тип описывается в разделе **Type**. Синтаксис этого действия состоит из ключевого слова **Type**, идентификаторов определяемых типов, значка равно «=» и описания вида типа:

Type T=<вид типа>;
TT=<вид типа>;

Пример.

Type Index=1..20;
A=array[index] of real;

Раздел переменных (Var)

Каждая переменная, встречающаяся в программе, должна быть описана до ее использования и отнесена к одному и только одному типу.

Пример.

Var r1, r2:real;
I,j,k:integer;
B:array[1..25] of real;

Область действия переменных определяется по правилу: **переменные локальны в блоке, где описаны, а также во всех вложенных блоках, если в них они не описаны повторно**.

Раздел действий

Эта часть программы начинается с ключевого слова **Begin** и заканчивается словом **End**, после которого ставится точка. Раздел действий – это выполняемая часть программы, состоящая из операторов.

Операторы

В языке программирования Pascal под операторами подразумеваются только действия. Операторы отделяются друг от друга точкой с запятой. Точку с запятой можно не ставить перед словами End и Until, поскольку такая запись будет означать наличие пустого оператора между этой точкой с запятой и указанным служебным словом. Категорически нельзя ставить точку с запятой перед словом Else, поскольку между Then и Else должен стоять ровно один оператор, а не два (тот, который написан программистом и пустой).

Оператор присваивания.

Пусть V – переменная, A – выражение. Тогда операция присваивания записывается в виде:
V:=A;
Выражение A может содержать константы, переменные, знаки операций, функции, скобки. Любое выражение в скобках вычисляется раньше, чем операция, предшествующая скобкам.

Замечание 1. Присваивание допускается для всех типов, за исключением типа файл.

Замечание 2. Переменной типа **real** (**double**) можно присвоить выражение типа **integer**, но переменной имеющей тип **integer** присвоить выражение типа **real** нельзя.

Составной оператор.

Составной оператор начинается ключевым словом **Begin** и заканчивается словом **End**. Между этими словами (их еще называют операторными скобками) помещается последовательность операторов, которые выполняются в порядке следования.

Замечание. Нельзя извне составного оператора передавать управление внутрь его.
Пример.

```
Begin
  R:=5;
  B:=R+2*pi;
End;
```

Условный оператор IF.

Условный оператор указывает, какие действия выполняются при истинности или ложности некоторого условия. Синтаксис этого оператора следующий:

If <Условие> **Then** <Оператор 1> **Else**<Оператор 2>;

Если <Условие> принимает значение **True**, то выполняется <Оператор 1>, иначе (при наличии **Else**) <Оператор 2>, расположенный за ключевым словом **Else**.

Замечание. Если при выполнении условия или его нарушении должно выполняться более одного действия, то эти действия заключают в операторные скобки **Begin** и **End** и <Оператор 1> и <Оператор 2> представляют собой составной оператор.
Пример.

If x>=0 **Then begin** a:=x*pi; B:=A*sqrt(x) **end**
Else begin A:=0; B:=exp(A); Writeln('A=0') **end**;
Существует и короткий вариант условного оператора: без использования служебного слова **Else**.
Пример.
If s>1 **Then** s:=1;

Оператор перехода Goto.

Синтаксис оператора перехода следующий:

Goto <Метка>;

Этот оператор (его еще называют оператором безусловного перехода) передает управление оператору, непосредственно следующему за меткой.
Пример.

```
Program Enter_one;
{$APPTYPE CONSOLE}
Uses  SysUtils;
Label back;
Var a:integer;
Begin
back:writeln('Enter one');readln(a);
If a<>1 Then Begin writeln('Mistake');Goto back End;
writeln('Correctly');readln
End.
```

Операторы цикла.

В языке программирования Pascal имеется три вида оператора цикла.

I. **Оператор For**.

For <переменная>**:=** <начальное значение> **To** <конечное значение> **Do** <оператор>**;**
Здесь переменная цикла увеличивается на единицу.
For <переменная>**:=** <начальное значение> **Downto** <конечное значение> **Do** <оператор>**;**
Здесь переменная цикла уменьшается на единицу.
Начальное и конечное значения имеют тип **Integer**. В теле цикла **For** его параметр меняться не должен. Циклы такого типа называют финитными, имея в виду то, что они всегда завершат свою работу и поэтому не могут служить причиной «зацикливания» программы.
Пример.

```
For i:=1 To 15 Do Begin
```

```
                    a:=a*I;
                    writeln(a:15)
                  End;
For i:=15 To 1 Do Begin
                    a:=I*sqrt(i);
                    writeln(a:15)
                  End;
```

II. **Оператор While.**

Синтаксис этого оператора следующий:

While <Выражение> **Do** <Оператор>;

Выражение имеет тип **Boolean,** оператор выполняется пока значение Выражения остается истинным (True) (проверка истинности осуществляется перед выполнением тела цикла).

III. **Оператор Repeat.**

Общий вид этого оператора следующий:

Repeat

<группа выполняемых операторов>

Until <Выражение>;

Тело цикла (операторы находящиеся между ключевыми словами **Repeat** и **Until**) выполняется пока значение <Выражения> - False. Проверка истинности осуществляется после выполнения тела цикла. Поэтому один раз <группа выполняемых операторов> отрабатывает в любом случае.

Массивы

Базируясь на простых типах, в языке Pascal можно строить более сложные типы переменных. Одним из таких типов является массив – структура состоящая из фиксированного числа компонент одного типа.
Синтаксис:
Var <идентификатор> : **array**[<тип индекса>,< тип индекса >] **of** <тип элемента>;
Пример. Описание двумерного массива с элементами целого типа:
Var M : **array**[1..3, 1..4] **of integer**;
В этом примере первый индекс (от 1 до 3) – номер строки, второй индекс (от 1 до 4) - номер столбца.

$$M = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \end{pmatrix}$$

Обращаться к элементу массива следует так: A:=M[i,j], где $i \in [1..3]$, $j \in [1..4]$. Переменной A присвоено значение M[i,j]. Если записать M[i,j]:=A, то элементу массива M[i,j] присвоено значение переменной A.

Функции и процедуры

Очень часто, почти всегда, программу следует разбивать на несколько блоков, где каждый блок может восприниматься как отдельная программа. Такое разбиение позволяет придать всей программе понятную структуру. В языке программирования Pascal (Delphi) такое разбиение осуществляется с помощью процедур и функций.

Процедуры

Процедура в рассматриваемом нами языке программирования имеет ту же структуру, что и главная программа (**Program**): разделов **Label, Const, Type, Var** и выполняемую часть от **Begin** до **End.** После слова **End** ставится точка с запятой в отличии от основной программы, где ставится точка. Процедура помещается в главной программе после раздела описания переменных **Var** перед **Begin** главной программы.

Важное замечание. Переменные, описанные внутри процедуры, действуют только в самой процедуре и называются локальными переменными. Если программа содержит много переменных и много процедур, то может случиться, что локальная и глобальная переменные будут иметь один и тот же идентификатор (переменные описаны в главной программе и процедуре). При вызове процедуры глобальный параметр временно «забывается». Когда же процедура выполнится, то глобальная переменная принимает снова первоначальное значение (значение, которое она имела перед выполнением процедуры). В результате путаницы не происходит.

Следующее важное понятие в описании процедур – это формальные параметры. Под формальными параметрами понимаются переменные, через которые передаются данные из программы в процедуру, либо из процедуры в программу. В качестве примера опишем программу, вычисляющую значение функции $y = a^x$ по формуле $y = e^{x \ln a}$.

Пример.

```
program degree;

{$APPTYPE CONSOLE}
uses
  SysUtils;

var p,q,z:real;
    fdeg:text;
procedure deg(a,x:real; var y:real);
   begin
     y:=exp(x*ln(a));
   end;

begin
  assignfile(fdeg,'fdeg.txt'); rewrite(fdeg);
  read(p,q);
  deg(p,q,z);
  write(fdeg,'возводим число ',p,' в степень ',q, ' получаем ',z:15);
  close(fdeg)
end.
```

В этой программе вызов процедуры производится оператором deg(p,q,z), где deg – имя процедуры, p, q, z – фактические параметры. При этом устанавливается взаимно-однозначное соответствие между фактическими и формальными параметрами, после чего управление передается процедуре. Когда процедура закончит свою работу, управление передается оператору следующим за вызовом процедуры (в нашем случае — оператору write).

Важно подчеркнуть:

1) число фактических параметров должно быть равно числу формальных параметров;
2) соответствующие фактические и формальные параметры должны совпадать по порядку и по типу.

Замечание. Если имеется запись

```
Var A:array [1..25] of integer;
    B:array [1..25] of integer;
```

то A и B считаются переменными разных типов! Поэтому при использовании массива в качестве параметра процедуры, для обеспечения совместимости следует сначала ввести тип нужной структуры.
Пример.

```
Type MM=array [1..25] of integer;
Var A:MM;
Procedure P(B:MM); // Описание процедуры P
Begin
{Текст процедуры}
…
End;
Begin
{Текст основной программы}
…
P(A); // Вызов процедуры P
…
End;
```

Параметры процедур могут быть параметрами-значениями и параметрами-переменными.

Параметры-значения

Если в качестве формального параметра указана переменная (*только параметр и его тип*), то такой параметр называется параметром - значения.

Для параметров-значений машина при вызове процедур выделяет место в памяти для каждого формального параметра, вычисляет значение фактического параметра и засылает его в ячейку, соответствующего формальному параметру. Если фактический параметр есть имя переменной, то значение этой переменной пересылается в ячейку соответствующую формальному параметру и далее эти параметры никак не связаны, то есть в памяти компьютера эти параметры занимают разные ячейки. Параметр-значение может быть константой, переменной, выражением.

Параметры-переменные

Если перед именем формального параметра стоит ключевое слово **Var,** то такой параметр есть параметр-переменная. В вышеприведенном примере это y:

procedure deg(a,x:**real**; **var** y:**real**);

Фактический параметр, соответствующий параметру-переменной, может быть только переменной (не константой, не выражением). Для параметра-переменной используется именно та ячейка, которая содержит соответствующий фактический параметр.

Замечание. Под формальные и фактические параметры-значения Pascal (Delphi) отводит разные области памяти. Поэтому результат выполнения процедуры может быть передан только через параметры-переменные.

Функции

Функция может аналогично процедуре восприниматься как отдельная подпрограмма. Но результат выполнения функции — это всегда определенное значение.

Описание функции начинается со слова **function**, затем идет имя описываемой функции, далее в скобках параметры функции по правилам, что и для процедуры, после закрывающей скобки ставится двоеточие и пишется тип результата функции. Вышеописанный пример может быть переписан так:

program degree;

```
{$APPTYPE CONSOLE}
uses
  SysUtils;

var p,q,z:real;
    fdeg:text;
function deg1(a,x:real):real;
   begin
     deg1:=exp(x*ln(a));
   end;

begin
  assignfile(fdeg,'fdeg.txt'); rewrite(fdeg);
  read(p,q);
  deg(p,q,z);
  write(fdeg,'возводим число ',p,' в степень ',q, ' получаем ',deg1(p,q):15);
  close(fdeg)
end.
```

После заголовка, так же как и у процедур, описываются метки, постоянные, типы, переменные и так далее. Затем между словами **begin** и **end** следуют действия, которые вычисляют функцию. В блоке функции обязательно должен присутствовать оператор присваивания имени функции значения:

```
deg1:=exp(x*ln(a))
```

Сходство между процедурой и функцией заключается в том, что в обоих случаях, используется блок-структура, при которой могут быть объявлены новые переменные, метки и т.д.

Отличие состоит в том, что значение у функции всегда должно вычисляться. Тип этого значения должен объявляться в заголовке. Кроме того, у функции, по крайней мере, один раз должно встречаться имя функции, стоящее слева от оператора присваивания и выражения, из которого находится конечное значение.

Замечание. В качестве параметра процедуры или функции может быть использована другая процедура или функция. Приведем программу, вычисляющую определенный интеграл методом средних прямоугольников.

Пример.

```
program integral;
{$APPTYPE CONSOLE}
```

```
uses SysUtils;
type ff=function(x:real):real;//описываем тип функции
{описание функции, которая будет фактическим параметром}
function f(x:real):real;
begin f:=sqr(x) end;
{описание функции, в которой параметром является функция}
function integr(a,b:real;g:ff;n:integer):real;
var s,d,d2:real; i:integer;
begin d:=(b-a)/n; d2:=d/2; s:=0;
for i:=1 to n do s:=s+g(a+(2*i-1)*d2);
integr:=s*d end;
begin
{вызов функции с параметром-функцией}
writeln(integr(0,1,f,100)); readln
end.
```

Замечание. У функций, так же как и у процедур, в качестве формальных параметров могут быть параметры-переменные.

Комментарии

Текст программы, заключенный между фигурными скобками, не воспринимается компьютером и там можно писать все, что угодно на любом языке. Также, не воспринимается текст в правой части строки после двух слеш. В комментариях рекомендуется указывать назначение процедур, функций, исполняемых операторов, назначение идентификаторов. Кроме того, сами идентификаторы используемых констант, переменных и меток могут быть словами, что-то говорящими программисту, и таким образом быть неявными комментариями. При сложной структуре программы, когда одни блоки вложены в другие, разумно начало и конец каждого из блоков помечать одинаковым комментарием и тогда будет видно, какой именно **End** соответствует какому **Begin**.

Ниже приведен довольно большой пример, изучение которого позволит понять работу многих программ.

Архиватор

Алгоритм архивации и разархивации описан ниже и далее идет его реализация.

1) В исходном файле пересчитывается количество различных встречающихся символов.

2) Если исходный файл пуст (не содержит ни одного символа), то файл-архив тоже будет записан пустым.

3) Определяются те частоты, с которыми каждый из присутствующих символов встречаются в файле.

4) Символы упорядочиваются по убыванию этих частот.

5) Символам ставятся в соответствие двоичные шифры по следующему принципу (рекурсивная процедура DELENIE):

5.a) Все символы разбиваются на две группы так, что суммы частот в группах примерно равны.

5.b) В первый разряд двоичного шифра в одной из групп пишется 1, а в другой группе 0.

5.c) Каждая из двух новых подгрупп делится снова на две подгруппы по тому же принципу, то есть по примерному равенству сумм частот в подгруппах.

5.d) Во второй разряд двоичного шифра в одной из подгрупп снова пишется 1, а в другой подгруппе 0.

5.e) И так далее, пока в каждой из под-под-под-...-подгрупп не будет по одному символу, и делить ее станет уже невозможно.

Поскольку у наиболее часто встречающихся символов частоты выше, то они получат более короткие двоичные шифры, а редко встречающиеся — более длинные.

Несмотря на то, что количества знаков в двоичных шифрах различных символов различное, такая кодировка не допускает разночтения при обратной расшифровке даже, если при записи шифры не отделены друг от друга пробелами (в отличие от азбуки Морзе, где между последовательностями точек и тире, соответствующими буквам, делаются паузы). В Теории вероятностей (см., например, Вентцель Е.С. Теория Вероятностей. Глава 18. Основные понятия теории информации. §18.8. Задачи кодирования сообщений. Код Шеннона — Фэно. М.: 1962) доказывается, что такая кодировка текста (из побуквенных кодировок) будет оптимальной. Более оптимальную кодировку можно получить, если целиком кодировать часто встречающиеся фразы.

6) Из-за неравномерности частот появления символов в исходном файле, возможен следующий казус: число символов в младшей под-подгруппе старшей подгруппы может оказаться больше числа символов в старшей под-подгруппе младшей подгруппы. В этом случае двоичные шифры более часто встречающихся символов окажутся несколько длиннее двоичных шифров несколько реже встречающихся символов. Исправить эту несуразность, возможно простой сортировкой двоичных шифров по их длине не изменяя порядка символов.

7) В файле-архиве первым байтом записывается число встречающихся символов (запись этого и других чисел осуществляется записью символа, компьютерный код которого, равен этому числу).

8) Потом записывается «алфавит», то есть символы, присутствующие в исходном файле в порядке убывания их частот и длины, соответствующих им двоичных шифров (на запись каждого числа, соответствующего длине двоичного шифра гарантированно хватает одного байта, поскольку эта длина никак не может превзойти 255).

9) Далее записываются двоичные шифры символов составленного «алфавита», а потом и самого текста исходного файла.

10) При этом двоичные шифры символов вплотную (без пробелов) приставляются друг другу, и получается длинная последовательность нулей и единиц.

11) Эта последовательность разбивается на восьмёрки (столько бит в одном байте) и каждое из таких восьмизначных чисел в двоичной системе счисления переделывается в число в десятичной системе, а затем и в символ (как и любое другое десятичное число при выводе в файл-архив) и записывается в файл-архив.

12) Для корректности завершения процесса будущей разархивации необходимо знать ещё одно число - это число оставшихся символов двоичного кода после записи последней полноценной восьмёрки. Это число нужно знать, чтобы при обратной разархивации исключить оставшиеся лишние (ничего не значащие) символы, которые вынужденно будут добавлены, поскольку в компьютере нельзя записать в файл не целое число байтов. Это число в файле-архиве записано сразу после «алфавита» и длин двоичных шифров и перед зашифрованными символами «алфавита». В программе этот пункт реализован между восьмым и девятым.

Эффективность архиватора основана на:

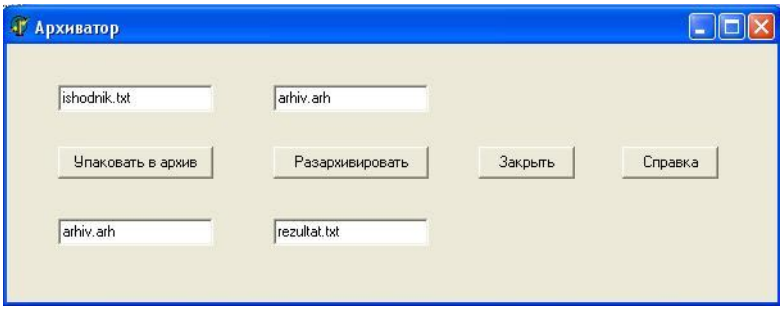
1) отсутствии в исходном файле некоторых из 256 символов;

2) том, что некоторые символы встречаются в исходном файле чаще остальных.

Чем больше отсутствующих символов и чем больше разность частот отдельных символов, тем эффективнее работа архиватора (например, абсолютно бесполезно архивировать фотографию в формате JPG, поскольку она уже в таком виде представляет специализированный архив).

Также, поскольку в начале файла-архива содержится "шапка" (то есть алфавит и некоторые необходимые числа), то при очень коротком исходном файле использование архиватора просто нецелесообразно.

На рисунке показан внешний вид основного окна программы-архиватора. Авторы надеются, что интерфейс этой программы интуитивно понятен.



Ниже приводится текст на Delphi основного модуля этой программы с подробными комментариями.

```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Di-
alogs,
  StdCtrls;

type
  TForm1 = class(TForm)
    Button1: TButton; //кнопка "Упаковать в Архив"
    Button2: TButton; //кнопка "Разархивировать"
    Button3: TButton; //кнопка "Закрыть"
    Button4: TButton; //кнопка "Справка"
 //Однострочные редакторы, из которых считывается имена файлов:
    Edit1: TEdit; //ахривируемого
    Edit2: TEdit; //куда будет записан полученный архив
    Edit3: TEdit; //разархивируемого
    Edit4: TEdit; //куда будет записан результат разархивации
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure Button4Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;



var
  Form1: TForm1;

implementation

uses Unit2;

type odinbyte=array[1..8]of integer;
```

```
var
nomer,dlina:array[0..255]of integer;
kod:array[0..255,1..255]of integer;
frikt:array[0..255]of real;
chislo:integer;

{$R *.DFM}

//Подпрограмма, переводящая из двоичной системы счисления в десятичную
Function decizdv(d:odinbyte):integer;
var i,s,t:integer;
begin
s:=0;t:=1;
for i:=8 downto 1 do begin s:=s+d[i]*t;t:=t*2 end;
decizdv:=s;
end;

//Подпрограмма, переводящая из десятичной системы счисления в двоичную
Procedure dvizdec(c:integer;var b:odinbyte);
var i,s:integer;
begin
{Знаем, что в байте не более 8 разрядов двоичной системы и пользуемся этим}
s:=128;
for i:=1 to 8 do
begin
if c<s then b[i]:=0 else begin b[i]:=1;c:=c-s end;
s:=s div 2;
end end;

//ЭТА ПОДПРОГРАММА САМАЯ ВАЖНАЯ
//Она создает двоичные шифры символов
//Так названа, потому что группа символов делится на две подгруппы
//Подпрограмма рекурсивная, то есть обращается сама к себе, что и
//обеспечивает деление подгрупп на под-подгруппы и так далее (см. справку)
Procedure Delenie(a,b,c:integer);
//a и b - номера начального и конечного символов в группе
```

```
//Это количество символов и делится на две подгруппы
//c - номер разряда шифра
var
x,y,z,nakop,polovina:real;
k,i:integer;
Begin
{Если в тексте ровно один символ несколько раз повторяющийся}
if (a=b) and (c=1) then begin dlina[a]:=1;kod[a,c]:=1;exit end;
{Если в подгруппе окажется только один символ,  то нет необходимо-
сти
добавлять в его шифр ещё разряд, поскольку его шифр и так уникален}
{Если группа состоит ровно из двух символов, то
в младший разряд шифра одного из них пишем единицу, а второго
нуль}
if b-a=1 then begin dlina[a]:=c;dlina[b]:=c;kod[a,c]:=1;kod[b,c]:=0;exit end;
{Если группа состоит более чем из двух символов}
if a<b then Begin polovina:=0;
{Вычисление суммы частот в группе}
for k:=a to b do polovina:=polovina+frikt[k];
{А это половина суммы частот в группе}
polovina:=polovina/2;nakop:=0;i:=a;
{Вычисление количества символов, суммы частот которых будут рав-
ны}
{половине суммы частот в группе}
repeat nakop:=nakop+frikt[i];
{nakop - накопленная сумма частот}
{inc(i) - это то же самое, что i:=i+1}
inc(i);
until (nakop>polovina) or (i>=b);
{dec(i) - это то же самое, что i:=i-1}
dec(i);
{А это попытка поделить точнее раз ровно половина суммы частот не
получится}
x:=frikt[i];y:=nakop-polovina;z:=polovina-nakop+x;
if y>z then dec(i);
{Теперь длина шифра стала другой, а именно c}
for k:=a to b do dlina[k]:=c;
{В шифры символов старшей группы добавим по единице}
for k:=a to i do kod[k,c]:=1;
{А младшей - нуль}
```

```
for k:=i+1 to b do kod[k,c]:=0;
{А вот и рекурсия, но на новом уровне (c+1) вместо c}
{Каждый из двух полученных промежутков снова делится на два промежутка}
delenie(a,i,c+1);delenie(i+1,b,c+1)
End End;

//Подпрограмма, вычисляющая номер присвоенный символу после упорядочивания
//частот (смотри справку)
Function antinomer(a:char):integer;
var n:integer;
b:boolean;
begin
b:=true;antinomer:=256;
{Проверяем все числа подряд, пока какое-нибудь не подойдёт}
for n:=0 to chislo do
{функция chr вычисляет символ по его коду}
if chr(nomer[n])=a then
begin antinomer:=n; b:=false;break end;
if b then begin
showmessage
('Обнаружен незашифрованный ранее символ. Обратитесь к авторам программы.');
form1.close end;
end;

//Это и есть собственно архиватор
Procedure Arhivart;
label b,conec,snova,posle,opjat;
var v,g:file of char;
sohr:array[1..264]of integer;
h:text;
massa:array[0..255]of integer;
nnn,dlsohr,dlbuf,ost,a1,a2,a3,y,l,k,i,j:integer;
ss,s:char;
dv:odinbyte;
bufer:array[1..264]of integer;
begin
{Проверим нет ли ошибки при вводе имени архива}
```

```
if form1.Edit1.text=form1.Edit2.text then begin
showmessage('Архивная копия не может называться также, как и ис-
ходный файл');
exit end;
{massa - это количество раз, которое встретился данный символ в ис-
ходном файле}
for j:=0 to 255 do begin massa[j]:=0;nomer[j]:=j end;
{Проверим наличие файла исходника}
if not fileexists(form1.Edit1.text) then
begin showmessage('Файл '+form1.Edit1.text+' не найден');exit end;
assignfile(g,form1.Edit1.text);reset(g);
{Файл shifrsmb.txt для работы архиватора не нужен и
создаётся исключительно в учебных целях}
assignfile(h,'shifrsmb.txt');rewrite(h);
i:=0;k:=0;
{eof - это аббревиатура End Of File}
while not eof(g) do begin
{Исходный файл будет просматриваться дважды}
{При первом просмотре идёт сбор информации}
{Архивация будет произведена при втором просмотре}
{i - это количество символов (всех) в исходном файле}
read(g,s);
{Считаем количества каждого из символов}
inc(i);
{ord - это функция, обратная функции chr}
{ord - вычисляет код для заданного символа}
inc(massa[ord(s)]); end;
for j:=0 to 255 do k:=k+massa[j];
writeln(h,'Всего байтов в исходном файле ',i,'=',k);
{Упорядочим символы в порядке убывания частот}
{методом всплывающего пузырька}
for l:=254 downto 0 do for j:=254 downto 254-l do
if massa[nomer[j]]<massa[nomer[j+1]] then begin
y:=nomer[j];nomer[j]:=nomer[j+1];nomer[j+1]:=y end;
writeln(h,'Упорядочили символы в порядке убывания частот');
{Определим сколько из различных 256 символов встречается в файле
(chislo)}
chislo:=0;
b:
```

```
if massa[nomer[chislo]]>0 then begin inc(chislo);if chislo<256 then goto b
end;
{Счёт начинается с нуля, а не с единицы - поэтому надо вычесть один}
dec(chislo);
writeln(h,'Различных символов в исходном файле ',chislo+1);
{Создаём файл-архив}
assignfile(v,form1.Edit2.text);rewrite(v);
{Если в исходном файле нет ни одного символа - файл-архив тоже
должен быть пустым}
if chislo<0 then goto conec;
{Это запись первого байта в файл-архив}
{В этом байте будет зашифровано число различных символов в файле-
исходнике}
s:=chr(chislo);write(v,s);
{Теперь вычисляем частоты (frikt)}
{dlina - это длины двоичных шифров, создаваемых рекурсивной под-
программой delenie}
for j:=0 to chislo do begin frikt[j]:=massa[nomer[j]]/k;dlina[j]:=0 end;
writeln(h,'Код Символ Встретился раз  Частота');
for j:=0 to 255 do
writeln(h,nomer[j]:3,chr(nomer[j]):5,massa[nomer[j]]:17,frikt[j]:12:8);
{Создание двоичных шифров - это обращение к рекурсивной процеду-
ре Delenie}
{0 и chislo - это номера начального и конечного символов}
{1 - эта единица - номер первого разряда создаваемого шифра}
delenie(0,chislo,1);
{Если вдруг более часто встречающийся символ имеет более длинный
шифр,}
{то надо поменять шифры местами}
{Это снова метод всплывающего пузырька}
for l:=chislo-1 downto 0 do for j:=chislo-1 downto chislo-1-l do
if dlina[j]>dlina[j+1] then begin
a1:=dlina[j];a2:=dlina[j+1];
for a3:=1 to a1 do sohr[a3]:=kod[j,a3];
for a3:=1 to a2 do kod[j,a3]:=kod[j+1,a3];
for a3:=1 to a1 do kod[j+1,a3]:=sohr[a3];
dlina[j]:=a2;dlina[j+1]:=a1 end;
writeln(h,'Символ Созданный двоичный шифр ');
for j:=0 to chislo do begin write(h,chr(nomer[j]):2,'     ');
for y:=1 to dlina[j] do write(h,kod[j,y]);writeln(h) end;
```

```
{Теперь запишем в файл-архив сами символы ("алфавит") и длины их
двоичных шифров}
for j:=0 to chislo do begin
s:=chr(nomer[j]);write(v,s);a3:=dlina[j];s:=chr(a3);write(v,s) end;
{Для корректного завершения процесса будущей разархивации}
{необходимо знать сколько будет лишних битов после записи}
{предпоследнего байта в файл-архив}
ost:=0;
for j:=0 to chislo do ost:=(ost+dlina[j]*(1+massa[nomer[j]])) mod 8;
{Это и есть байт, в котором зашифровано число лишних битов}
ss:=chr(ost);
writeln(h,'остаток ',ost);
write(v,ss);
{"Алфавит" виртуально ставим в начало файла-исходника и}
{составляем из двоичных шифров последовательность нулей и еди-
ниц}
{Их записываем в bufer и разбивая на восьмёрки записываем в виде
байтов}
{в файл-архив}
dlbuf:=0;
for j:=0 to chislo do begin {A}
for l:=1 to dlina[j] do bufer[dlbuf+l]:=kod[j,l];
dlbuf:=dlbuf+dlina[j];
snova:
if dlbuf>7 then begin
{если длина буфера достигла 8 бит или более - образуем байт из пер-
вых 8 битов}
for l:=1 to 8 do dv[l]:=bufer[l];
a2:=decizdv(dv);s:=chr(a2);write(v,s);
{Сдвигаем данные в буфере на 8 позиций}
dlbuf:=dlbuf-8;
for l:=1 to dlbuf do bufer[l]:=bufer[l+8];
goto snova end end; {A}
{Исходный файл просматривается второй раз, теперь для шифровки}
reset(g);
nnn:=0;
{Запись "шапки" завершена, пора шифровать и сам исходный файл}
{шифровка самого текста идёт также как и "алфавита"}
while not eof(g) do begin {C}
inc(nnn);read(g,s);j:=antinomer(s);
```

```
for l:=1 to dlina[j] do bufer[dlbuf+l]:=kod[j,l];
dlbuf:=dlbuf+dlina[j];
opjat:
{если длина буфера достигла 8 бит или более - образуем байт из первых 8 битов}
if dlbuf>7 then begin {B}
for l:=1 to 8 do dv[l]:=bufer[l];
a2:=decizdv(dv);s:=chr(a2);write(v,s);
{Сдвигаем данные в буфере на 8 позиций}
dlbuf:=dlbuf-8;
for l:=1 to dlbuf do bufer[l]:=bufer[l+8];
goto opjat end  {B} end; {C}
{пришло время шифровки последнего неполного байта}
if dlbuf>0 then
{добавим незначащие единицы до полного байта}
for l:=dlbuf+1 to 8 do bufer[l]:=1;
for l:=1 to 8 do dv[l]:=bufer[l];
{переводим его в двоичный код}
a2:=decizdv(dv);
{потом в символ}
s:=chr(a2);
{и записываем в файл-архив}
write(v,s);
conec:closefile(v);closefile(h);closefile(g);
showmessage('Архивация завершена')
end;
```

```
//Архиватор бесполезен без разархиватора
//Это разархиватор
procedure razarhivatr;
label ewe,iewe,bez,conec;
var otvet,v:file of char;
s:char;
hvost,l,a1,a3,dlsohr,j,simbvost,chsimb:integer;
dlinakoda:array[0..255]of integer;
sohr:array[1..264]of integer;
dva:odinbyte;
kody:array[0..255,1..255]of integer;
alfavit:array[0..255]of char;
buka:boolean;
begin
{Проверим нет ли ошибки при вводе имени файла-результата}
if form1.Edit3.text=form1.Edit4.text then
begin
showmessage('Разахивированный файл не может называться также, как
и архив');
exit end;
{Проверим наличие файла-архива}
if not fileexists(form1.Edit3.text) then
begin showmessage('Файл '+form1.Edit3.text+' не найден');exit end;
assignfile(v,form1.Edit3.text);reset(v);
assignfile(otvet,form1.Edit4.text);rewrite(otvet);
{Если файл-архив пуст, то пустым был и оригинал}
if eof(v) then goto conec;
reset(v);read(v,s);
{Начинаем расшифровку: chsimb - это число различных символов в
тексте}
chsimb:=ord(s);
for j:=0 to chsimb do begin
{Это - "буквы" "алфавита"}
read(v,s);alfavit[j]:=s;
{Считываем длины их кодов}
read(v,s);dlinakoda[j]:=ord(s);
end;
{А это - число битов в последнем неполном байте}
read(v,s);
```

```
simbvost:=ord(s);
{Небольшой пересчёт: нужно знать число незначащих символов+1}
if simbvost=0 then hvost:=1 else hvost:=9-simbvost;
a1:=-1; { a1 Это количество найденных кодов}
dlsohr:=0;
{Двоичные коды тоже помещаются в буфер (sohr), как и при архива-
ции}
{dlsohr - количество двоичных символов в буфере}
{Сначала расшифровываем "алфавит"}
ewe: if (dlsohr<255) and (not eof(v)) then begin
read(v,s);a3:=ord(s);dvizdec(a3,dva);
for l:=1 to 8 do sohr[dlsohr+l]:=dva[l];
dlsohr:=dlsohr+8;
goto ewe end;
{нужно найти коды зная их длину}
if a1<chsimb then begin
inc(a1);
for l:=1 to dlinakoda[a1] do kody[a1,l]:=sohr[l];
dlsohr:=dlsohr-dlinakoda[a1];
for l:=1 to dlsohr do sohr[l]:=sohr[l+dlinakoda[a1]];
goto ewe end;
{Теперь расшифровываем бывший текст}
{нужно искать какая это буква}
iewe: if (dlsohr<255) and (not eof(v)) then begin
read(v,s);a3:=ord(s);dvizdec(a3,dva);
for l:=1 to 8 do sohr[dlsohr+l]:=dva[l];
dlsohr:=dlsohr+8;
goto iewe end;
{какая это буква ищется прямым перебором кодов}
bez:
{Определим не последний ли это неполный байт}
if dlsohr<hvost then goto conec;
for j:=0 to chsimb do begin
buka:=true;
for l:=1 to dlinakoda[j] do
if sohr[l]<>kody[j,l]then begin buka:=false;break end;
if buka then begin
write(otvet,alfavit[j]);
dlsohr:=dlsohr-dlinakoda[j];
for l:=1 to dlsohr do sohr[l]:=sohr[l+dlinakoda[j]];
```

```
if eof(v) then goto bez;
goto iewe end; end;
conec:
closefile(v);closefile(otvet);
showmessage('Разархивация завершена')
end;

//реакция на нажатие кнопки "Упаковать в Архив"
procedure TForm1.Button1Click(Sender: TObject);
begin
Arhivart;
end;

//реакция на нажатие кнопки "Разархивировать"
procedure TForm1.Button2Click(Sender: TObject);
begin
razarhivatr;
end;


//реакция на нажатие кнопки "Закрыть"
procedure TForm1.Button3Click(Sender: TObject);
begin
form1.close
end;

//реакция на нажатие кнопки "Справка"
procedure TForm1.Button4Click(Sender: TObject);
begin
form2.showmodal
end;

begin

end.
```

Далее, в качестве примера, приведем текстовый файл (формат TXT), подготовленный для шуточного теста, который заархивируем с помощью нашей программы. Если Вы читали комментарии в программе, то заметили, что при архивации программа в учебных целях создает текстовый файл SHIFRSMB.TXT. В этом файле будут записаны:

1) Количество байт в исходном файле (с проверкой);
2) Количество различных символов в исходном файле;
3) После упорядочивания по убыванию частот компьютерные коды символов, сами символы, сколько раз они встретились, частоты;
4) Символы и созданные для них двоичные шифры;
5) Количество бит в последнем не полном байте.

Архивируемый файл

Что такое Эстония?
Государство в Европе
Целебная трава
Мазь для ног
Имя волшебницы в сказке про Изумрудный город

Что такое элерон?
Руль высоты в самолёте
Мясное блюдо
Химический элемент
Остров в Финском заливе

Что такое солнце?
Звезда
Планета
Комета
Астероид

Что такое Луна?
Планета
Звезда
Комета
Астероид

Из чего делают бензин?
Из нефти
Из керосина
Из машинного масла
Из молока

Как ещё называют жёсткий диск компьютера?
Винчестер
Маузер
Наган
Берданка

Что такое цугцванг?
Шахматный термин
Фамилия автора сборника задач по аналитической геометрии
Название вида верблюдов
Деталь автомобиля

Что такое астрономия?
Наука о космических телах
Название тихоокеанской рыбы
Название далёкой планеты
Наука о земле

Земля вокруг солнца совершает полный оборот за
Год
Четыре года
День
Месяц

Какая планета солнечной системы самая большая?
Юпитер
Земля
Цербер
Уран

Сколько известно планет солнечной системы в настоящее время?
9
10
5

7

Чем планеты отличаются от звёзд?
Планеты не испускают собственного света
У планет нет спутников
На планетах есть жизнь
Планеты вращаются вокруг своей оси

Сколько спутников у Земли?
1
2
17
9

Кто такой Юрий Гагарин?
Первый космонавт
Конструктор ракеты
Математик, рассчитавший полёт первого спутника вокруг Земли
Солист известной рок группы

Кто такой Нил Армстронг?
Первый человек, высадившийся на Луну
Конструктор Лунохода
Автор книги "Незнайка на Луне"
Автор известной песни "На обратной стороне Луны"

Что такое Луна?
Спутник земли
Спутник Солнца
Отражение Солнца на ночном небе
Космическая станция, созданная американцами

Шесть умножить на шесть это
36
31
33
37

Основателями славянкой письменности были
Кирилл и Мефодий

Борис и Глеб
Князь Игорь и княгиня Ольга
Нестор летописец

Что такое экватор?
Линия, делящая земной шар на северное и южное полушария
Линия, делящая земной шар на западное и восточное полушария
Самая южная точка земного шара
Самая высокая гора на земном шаре

Смена дня и ночи происходит из-за
Вращения Земли вокруг своей оси
Вращения Земли вокруг Солнца
Вращения Луны вокруг Земли
Вращения Солнца вокруг своей оси

Кто такой Тур Хейердал?
Знаменитый путешественник на папирусной лодке "Ра"
Знаменитый художник
Первооткрыватель материка "Антарктида"
Автор книги про Карлсона

Теперь продемонстрируем файл SHIFRSMB.TXT для приведенного в качестве примера архивируемого файла. Заметим, что не все символы окажутся печатаемыми. Вместо них будут стоять пробел или белый квадрат. Такие символы встречаются и в простейших текстовых файлах. Это, например, символы пробела, перехода на другую строку, и другие служебные.

Информация, находящаяся в файле SHIFRSMB.TXT.

Всего байтов в исходном файле 2376=2376
Упорядочили символы в порядке убывания частот
Различных символов в исходном файле 72

Код	Символ	Встре-тился раз	Частота	Код	Символ	Встре-тился раз	Частота
				209	С	12	0.0050
32		243	0.1022	199	З	11	0.0046
224	а	175	0.0736	205	Н	11	0.0046
238	о	175	0.0736	215	Ч	10	0.0042
229	е	153	0.0643	254	ю	10	0.0042
237	н	136	0.0572	203	Л	9	0.0037
10		124	0.0521	249	щ	9	0.0037
13		124	0.0521	34	"	8	0.0033
242	т	123	0.0517	200	И	8	0.0033
232	и	113	0.0475	245	х	8	0.0033
240	р	87	0.0366	192	А	7	0.0029
241	с	86	0.0361	207	П	7	0.0029
234	к	72	0.0303	230	ж	7	0.0029
235	л	71	0.0298	184	ё	6	0.0025
226	в	65	0.0273	204	М	6	0.0025
236	м	54	0.0227	44	,	5	0.0021
255	я	44	0.0185	51	3	5	0.0021
243	у	40	0.0168	194	В	5	0.0021
231	з	37	0.0155	49	1	4	0.0016
233	й	33	0.0138	195	Г	4	0.0016
239	п	33	0.0138	206	О	4	0.0016
228	д	32	0.0134	253	э	4	0.0016
227	г	28	0.0117	55	7	3	0.0012
251	ы	28	0.0117	57	9	2	0.0008
252	ь	18	0.0075	193	Б	2	0.0008
63	?	17	0.0071	196	Д	2	0.0008
225	б	16	0.0067	208	Р	2	0.0008
247	ч	16	0.0067	211	У	2	0.0008
248	ш	14	0.0058	212	Ф	2	0.0008
202	К	13	0.0054	213	Х	2	0.0008
246	ц	13	0.0054	214	Ц	2	0.0008

216	Ш	2	0.0008
222	Ю	2	0.0008
244	ф	2	0.0008
45	-	1	0.0004
48	0	1	0.0004
50	2	1	0.0004
53	5	1	0.0004
54	6	1	0.0004
197	Е	1	0.0004
210	Т	1	0.0004
221	Э	1	0.0004
Остальные символы		0	0.0000

Символ	Созданный двоичный шифр
	111
а	1101
о	1100
е	1011
н	1010
	1001
	1000
т	0111
и	01101
р	01100
с	01011
к	01010
л	01001
в	01000
м	00111
я	001101
у	001100
з	001011
й	001010
п	001001
д	0010001
г	0010000
ы	0001111
ь	0001110
?	0001101
б	0001100
ч	0001011
ш	00010101
К	00010100

Символ	Созданный двоичный шифр
ц	00010011
С	00010010
З	00010001
Н	00010000
Ч	00001111
ю	00001110
Л	00001101
щ	00001100
"	00001011
И	00001001
х	00000111
А	000010101
П	000010100
ж	000010001
ё	000010000
М	000001101
,	000001100
З	000001011
В	000001010
1	000001001
Г	000000111
О	0000010001
э	0000010000
7	0000001101
9	0000001100
Б	0000001011
Д	0000001001
Р	0000001000
У	0000000111

```
 Ф      0000000110              0      00000000101
 Х      0000000101              2      00000000100
 Ц      0000000100              5      00000000011
 Ш      00000010101             6      00000000001
 Ю      00000010100             Е      00000000000
 Ф      00000000111             Т      000000000101
 -      00000000110             Э      000000000100
остаток 2
```

Литература

1. *Власенко С.Ю.* Microsoft Word 2002. – СПб.: БХВ – Петербург, 2002 – 992 с.
2. *Додж М., Стинсон К.* Эффективная работа с Microsoft Excel 2000. — СПб: Издательство «Питер», 2000. — 1056 с.
3. *Долженков В.А.*, Колесников Ю.В. Microsoft Excel 2002. – СПб.: БХВ – Петербург, 2002. – 1072 с.
4. *Кнут Д.Е.* Все про TeX. – Протвино: РДТеХ, 1993. – 592 с.
5. *Львовский С.М.* Набор и верстка в системе LaTeX. – М.: МЦНМО, 2003. – 448 с.
6. *Нортон П., Андерсен В.* Разработка приложений в Access 97 в подлиннике: пер. с англ. – СПб.: BHV – Санкт – Петербург, 1998. 656 с.
7. *Хэлворсон М., Янг М.* Эффективная работа: Office XP. — СПб: Издательство «Питер», 2003. — 1072 с.
8. *Буч Г.* Объектно-ориентированное проектирование с примерами применения. Киев; М., 1992.
9. *Марченко А. И.* Программирование в среде Borland Pascal 7.0. Киев, 1996.
10. *Страуструп Б.* Язык программирования C++. Специальное издание: Пер. с англ. М., 2004.
11. Практическое руководство по программированию. /Под ред. Б. Мика, П. Хит, Н. Рашби. М., 1986.
12. *Додж М., Стинсон К.* Эффективная работа с Microsoft Excel 2000. СПб., 2000.
13. *Евсеев Г., Мураховский В., Симонович С.* Новейший самоучитель работы на компьютере. Эффективный курс – Москва: «ТехБук», 2004. — 688 с.
14. *Кузьменко В. Г.* VBA 2002., М. 2002.
15. *Боревич З. И.* Определители и матрицы. СПб., 2004.
16. *Романовский И. В.* Дискретный анализ. СПб., 2003.
17. *Самарский А. А, Гулин А. В.* Численные методы. М., 1989.
18. *Вирт Н.* Алгоритмы и структуры данных. М., Мир, 1989.
19. *Вирт Н.* Программирование на языке Модула-2. М., Мир, 1987.

Содержание

Учебное издание

Дмитриев Никита Николаевич
Сахаров Вадим Юрьевич

ВВЕДЕНИЕ В ИНФОРМАТИКУ

У ч е б н о е п о с о б и е

Издательство «ВВМ»
E-mail: vvmpub@yandex.ru

Вёрстка: Олег Шакиров

Подписано к печати 09.05.08. Формат $60 \times 84^{1}/_{16}$.
Бумага офсетная. Гарнитуры Таймс, Ариал. Печать офсетная. Печ. л. 9,07
Тираж 100 экз. Заказ 4181

Отпечатано в Отделе оперативной полиграфии химического факультета СПбГУ
198504, Санкт-Петербург, Старый Петергоф, Университетский пр., 26
Тел.: (812) 428-40-43