Федеральное государственное бюджетное образовательное учреждение
высшего образования «Белгородский государственный технологический
университет им. В.Г. Шухова»

Д.Н.Старченко, А.И. Рыбакова

Информатика

Методические указания к выполнению лабораторных работ
часть 2

Белгород
2018

Федеральное государственное бюджетное образовательное учреждение
высшего образования «Белгородский государственный технологический
университет им. В.Г. Шухова»

Кафедра информационных технологий

Д.Н.Старченко, А.И Рыбакова

Утверждено
научно-методическим
советом университета

Информатика

Методические указания к выполнению лабораторных работ
часть 2

Белгород
2018

УДК 004(07)
ББК 32.81я7
И 74

Составители:
Д.Н.Старченко, А.И.Рыбакова

Рецензент канд. техн. наук,доцент. Поспелова М.А.

И74 **Информатика**: методические указания к выполнению лабораторных работ / сост: Д.Н.Старченко, А.И. Рыбакова,
- Белгород: Изд-во БГТУ, 2018. – с.

Методические указания подготовлены в соответствии с рабочей программой дисциплины «Информатика» (часть 2) и требований Федерального государственного образовательного стандарта специальности. Предназначены для приобретения студентами навыков работы.

Методические указания предназначены для студентов специальности 23.03.01 – «Технология транспортных процессов», 23.03.03-02 «Эксплуатация транспортно-технологических машин и комплексов»; 23.03.03-01 «Автомобильный сервис».

Данное издание публикуется в авторской редакции.

УДК 004(07)
ББК 32.81я7

©Белгородский государственный технологический университет (БГТУ) им. В. Г. Шухова, 2018

Правила выполнения лабораторных работ

По курсу информатики предусмотрено выполнение лабораторных работ. Студент обязан перед выполнением каждой лабораторной работы самостоятельно ознакомиться с теоретическим материалом и по результатам ее выполнения предоставить отчет. Все отчеты о выполнении лабораторных работ оформляются в одной отдельной тетради. Каждый отчет (см. ниже пример оформления лабораторной работы) должен содержать:

1. Заголовок лабораторной работы – номер лабораторной работы, данные о студенте, слова «Выполнение» и «Защита», название и цель работы.
2. Содержание работы и индивидуальные задания.
3. Краткие теоретические сведения.
4. Ход работы – краткое описание последовательности действий, произведенных при выполнении работы.
5. Результаты выполнения лабораторной работы в электронном виде.
6. Вывод.

Пример оформления лабораторной работы

Лабораторная работа №1
студента группы ЭТА-11
Гоцай Евгений Сергеевич
Выполнение:_____ Защита:_____

Устройство персонального компьютера

Цель работы: ознакомиться с основными и дополнительными устройствами персонального компьютера и их основными характеристиками.

Содержание работы

1. *Ознакомьтесь с теоретическим материалом .*
2. *Занесите в отчет описание устройств* ***ввода****, входящих в состав закрепленного за Вами или Вашего домашнего компьютера.*
3. *Занесите в отчет описание устройств* ***вывода****, входящих в состав закрепленного за Вами или Вашего домашнего компьютера.*

4. *Занесите в отчет сведения о системе и комплектации компьютера.*
5. *Сделайте вывод о производительности компьютера*.

Краткие теоретические сведения

К основным устройствам относятся…

Ход работы

1. *Прочел страницы …учебника под редакцией…Ознакомился с материалом методического пособия …*
2. *Устройства ввода:*
   – *клавиатура имеет … клавиш , …;*
   – *манипулятор мышь …, имеет …;*
   – *…*
3. *Устройства вывода:*
   – *монитор ……., диагональ …, разрешение…,*
   – *…*
4. *Подвел указатель мыши к объекту рабочего стола Мой компьютер и нажал правую клавишу мыши. В появившемся контекстном меню выбирал команду Свойства. Информация о компьютере: процессор … работает на частоте …, объем ОЗУ …. Установлена операционная система …*

Вывод: из пунктов 2 и 3 следует, что оснащение компьютера периферийным оборудованием …; из пункта 4 можно заключить, что производительность компьютера…

Лабораторная работа № 1

УСТРОЙСТВО ПЕРСОНАЛЬНОГО КОМПЬЮТЕРА

Цель работы: ознакомиться с основными и дополнительными устройствами персонального компьютера и их основными характеристиками.

Краткие теоретические сведения

Устройства персонального компьютера подразделяются на два вида: основные (без них невозможно работать на компьютере) и дополнительные.

К основным устройствам относятся: системный блок, клавиатура, видеомонитор (дисплей).

К дополнительным устройствам относятся: мышь, принтер, сканер, модем, плоттер и т.д.

Системный блок

В системном блоке обязательно размещаются блок питания, который обеспечивает все части компьютера электрическим питанием и материнская плата, на которой размещается процессор, постоянное запоминающее устройство (ПЗУ), оперативное запоминающее устройство (ОЗУ), дополнительные платы расширения (видеоадаптер, сетевой адаптер и т.д.) и порты для подключения дополнительных внешних устройств. Кроме того, в системном блоке размещается долговременная память на магнитных дисках ("винчестер" или жесткий диск), DVD привод.

Материнская плата является одним из основных составляющих системного блока и представляет собой прямоугольник фольгированного текстолита, на котором распаяны микросхемы, дискретные элементы (резисторы, конденсаторы и т.д.), разнообразные разъемы (рис. 1.1). Их выводы соединены между собой дорожками из фольги, покрывающей текстолит.

Все современные компьютеры основаны на *шинной архитектуре*. Это означает, что для обмена данными между элементами компьютера используется одна или несколько шин, представляющие собой

совокупность проводников, по которым проходят электрические сигналы. Шины отличаются друг от друга пропускной способностью (количеством информации, передаваемой за единицу времени) и разрядностью (шириной) – количеством информации, передаваемой за один такт. На современных материнских платах имеется системная шина, через которую процессор обменивается данными с памятью и другими устройствами, а также локальные шины для подключения дополнительных плат расширения. Шины управляются специализированными устройствами – *контроллерами*.

На материнских платах находятся специальные перемычки – **джамперы**, позволяющие подстроить ее под тип процессора и других устройств, устанавливаемых на ней.

На материнской плате устанавливаются разъемы для установки дополнительных устройств – **слоты расширения**.

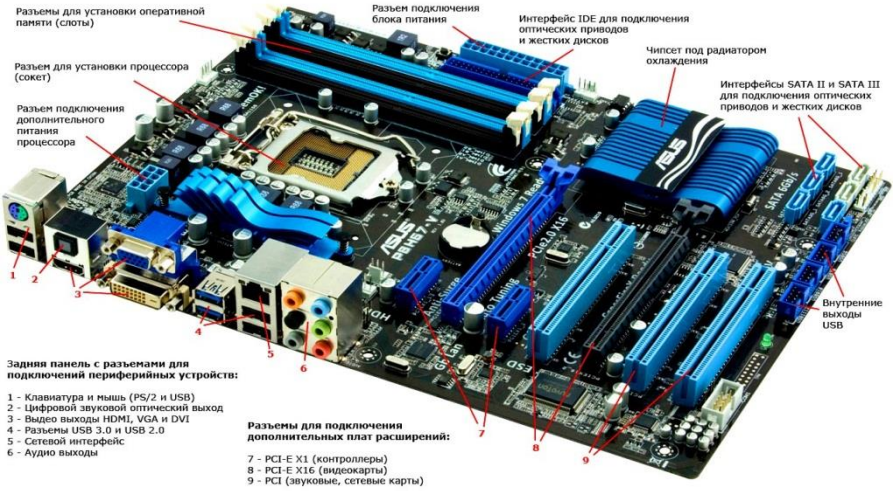


Рис. 1.1. Материнская плата

Материнская плата – это печатная плата, которая предназначена для подключения основных комплектующих компьютера. Часть из них, например, процессор или видеокарта устанавливается непосредственно на саму материнскую плату в предназначенный для этого разъем. А другая часть комплектующих, к примеру, жесткий диск или блок питания, подключается к материнской плате с помощью специальных кабелей.

Процессор – это микросхема и одновременно «мозг» компьютера. Он отвечает за выполнение всех операций. Чем лучше процессор тем быстрее он будет выполнять эти самые операции, соответственно компьютер будет работать быстрее. Процессор конечно влияет на скорость работы компьютера, и даже очень сильно, но от вашего жесткого диска, видеокарты и оперативной памяти также будет зависеть скорость работы ПК. Так что самый мощный процессор не гарантирует большую скорость работы компьютера, если остальные комплектующие уже давно устарели.

Основными характеристиками процессора являются:

тактовая частота – количество тактов, производимых процессором за 1 секунду. Операции, производимые процессором, не являются непрерывными, они разделены на такты. Эта характеристика определяет скорость выполнения операций и непосредственно влияет на производительность процессора. Процессор Pentium и его модификации имеют тактовые частоты от 60 МГц до 1,5 ГГц (выполнять 1,5 миллиарда операций в секунду);

разрядность – количество двоичных разрядов, которые процессор обрабатывает за один такт. Указывая разрядность процессора 64, имеют в виду, что процессор имеет 64-разрядную шину данных, т.е. за один такт он обрабатывает 64 бита.

Процессор состоит из следующих компонентов:

Арифметико-логическое устройство – выполняет арифметические операции (сложение, вычитание, умножение, деление) и операции программной логики (сравнение, условный переход, повторение).

Арифметико-логическое устройство может работать только с регистрами процессора;

Устройство управления – контролирует последовательность выполнения команд программы размещенной в оперативной и кэш-памяти. Устройство управления может обмениваться информацией с различными устройствами компьютера;

Регистры процессора – специальный вид памяти, небольшого объема, необходимый для временного хранения данных с которым работает процессор;

Кэш-память процессора – оперативная память процессора, в которой хранятся часто вызываемые фрагменты программ и часто используемые ими данные.

Память предназначена для хранения данных и программ их обработки и делится на внутреннюю (расположенную непосредственно на материнской плате) и внешнюю.

В **ОЗУ** хранятся исполняемые в данный момент программы и необходимые для этого данные. Это энергозависимая память и ее содержимое после выключения питания теряется. Оперативная память конструктивно оформляется в виде модулей – небольших плат, на которых расположены микросхемы памяти. Модули могут иметь различный объем и устанавливаются в специальные разъемы на материнской плате (см. рис. 1.1). Общий объем оперативной памяти может достигать нескольких гигабайт.

Комплект программ, находящихся в **ПЗУ**, образует *базовую систему ввода-вывода (BIOS — Basic Input Output System).* Основное назначение программ этого пакета состоит в том, чтобы проверить состав и работоспособность компьютерной системы и обеспечить взаимодействие с клавиатурой, монитором, жестким диском и дисководом гибких дисков. Программы, входящие в *BIOS,* позволяют нам наблюдать на экране диагностические сообщения, сопровождающие запуск компьютера, а также вмешиваться в ход запуска с помощью клавиатуры.

Кэш-память является «посредником» между процессором и оперативной памятью. В ней хранятся наиболее часто используемые данные. Кэш-память расположена на одном кристалле с процессором, поэтому время доступа к ней во много раз меньше, чем к обычной оперативной памяти. В современных процессорах используется многоуровневая система кэширования. Кэш первого уровня (L1) самый быстрый, но имеет небольшой размер (несколько десятков килобайт). Кэш второго уровня (L2) имеет, как правило, существенно больший объем (до нескольких мегабайт), но меньшее быстродействие. Кэш третьего уровня (L3) встречается в процессорах крайне редко и имеет еще меньшее быстродействие, чем L2.

Винчестеры (накопители на жестких магнитных дисках) относятся к внешней памяти и предназначены для долговременного хранения информации. Они представляют собой малогабаритный пакет из жестких алюминиевых дисков с ферромагнитным покрытием, вращающихся с высокой скоростью на одной оси и размещенных в герметичном корпусе вместе с головками записи-чтения. Рабочие поверхности дисков разделены на концентрические кольца – *дорожки*, информация на которые записывается отрезками фиксированной

длины – *секторами*. Считывание и запись производится несколькими головками со всех дисков сразу. При этом совокупность совпадающих на разных поверхностях дорожек называется *цилиндром*. Таким образом, количество цилиндров, поверхностей и секторов на дорожке определяет емкость винчестера, которая может достигать нескольких терабайт.

Основными характеристиками этого устройства являются:

Емкость - количество данных, которые могут храниться на накопителе. Сейчас современные накопители, благодаря технологии перпендикулярной записи, имеют размеры в 3 Терабайта (3000 Гб).

Физический размер. Накопители, имеющие ширину 3,5 дюйма (редко 2,5 дюйма) используются в настольных компьютерах, а 2,5 или 1,8 дюйма – в мобильных устройствах (ноутбуки или нетбуки).

Скорость вращения шпинделя. Важная характеристика, от которой зависят время доступа и средняя скорость передачи данных. Чем больше скорость вращения, тем быстрее жесткий диск. Измеряется в оборотах в минуту и в основном имеет значения: 5400 об/мин (в основном ноутбуки или высокоемкостные диски шириной 3,5”), 7200 об/мин (настольные ПК, реже ноутбуки), 10000 и 15000 об/мин (высокопроизводительные ПК или серверы). Уровень шума накопителя сильно возрастает на высоких оборотах и при сборке тихой системы выбирать диск со скоростью выше 7200 об/мин не рекомендуется.

Интерфейс подключения – тип разъема и шины, которые используются для подключения и обмена данными с жестким диском. Долгое время, самым распространённым интерфейсом в настольных и мобильных компьютерах являлся Parallel ATA (он же IDE, ATA, Ultra ATA, UDMA 133) с максимальной пропускной способностью 133 Мбайт/сек, в котором использовался принцип параллельной передачи данных. Из-за этого разъем подключения был достаточно широким и имел 40 контактов, а громоздкие 80-жильные кабели подключения всегда мешались в корпусе и мешали нормальному охлаждению. Пропускная способность современной 3-ей ревизии SATA III составляет 600 Мбайт/сек и превышает возможности PATA в 4,5 раза. Более того, SATA использует миниатюрный 7-контактный разъем, и соответственно, кабель гораздо меньшей площади, чем IDE, за счёт чего уменьшается сопротивление воздуху, обдувающему

комплектующие компьютера и упрощается разводка проводов внутри системного блока.

Оптические (лазерные) диски, так же как и дискеты являются сменными носителями информации. Из них наиболее распространены в настоящее время компакт-диски (***CD-ROM)***. Их емкость достигает 800 Мбайт, и предназначены они только для чтения. Для оптической записи применяются как специальные носители (*CD-R* для однократной записи и *CD-RW* для многократной), так и специальное устройство – пишущий *CD* привод (*CD-ReWriter*).

Другим популярным форматом лазерных дисков является ***DVD-ROM***. Диски этого формата, за счет более высокой плотности записи, могут содержать до 17 Гбайт информации. Так же как и для компакт-дисков имеются диски однократной и многократной записи, а также пишущие приводы.

Для подключения к материнской плате *CD*/*DVD* приводы используют те же интерфейсы, что и жесткие диски.

Кроме перечисленных выше устройств, непосредственно на материнской плате можно устанавливать дополнительные платы, расширяющие функциональность компьютера: звуковая карта, внутренний модем и др. Все платы расширения имеют в нижней части ряд контактов, строго соответствующих слоту, в который они могут устанавливаться.

Наряду с платами расширения, в составе персонального компьютера имеются и внешние устройства, которые должны быть подключены к системному блоку. Для этого на материнской плате имеются соответствующие порты. Набор портов может отличаться в зависимости от модели и фирмы-производителя, но на любой современной материнской плате, как правило, имеются следующие:

- порты клавиатуры и мыши *PS/2*;
- последовательный *COM*-порт;
- параллельный *LPT*-порт;
- порты универсальной последовательной шины *USB*.

Порты *COM* и *LPT* в настоящее время используются достаточно редко и устанавливаются для совместимости с устаревшим оборудованием. Основная масса оборудования, которое раннее подключалось через эти порты (принтеры, сканеры и т.д.), теперь выпускается с высокоскоростным интерфейсом *USB*. Клавиатура и мышь также могут подсоединяться через этот интерфейс. Кроме того, через *USB* могут подключаться и устройства внешней памяти. В основном это касается устройств *флэш-памяти*, получивших широкое

распространение в последнее время. Существуют также внешние жесткие диски и *CD*/*DVD* приводы.

В современные материнские платы производитель часто встраивает (интегрирует) дополнительные устройства, чтобы избавить пользователя от необходимости их дальнейшей покупки. Это могут быть сетевой адаптер, модем, звуковая карта и другие устройства. В этом случае разъемы интегрированных устройств располагаются рядом с перечисленными выше портами. Однако если пользователя персонального компьютера не устраивают характеристики интегрированных устройств, он может дополнительно установить их аналоги, используя соответствующие слоты или порты.

Клавиатура

Клавиатура компьютера является одним из основных устройств ввода информации и команд на ее обработку. Группы клавиш приведены на рис. 1.2.

Алфавитно-цифровые клавиши. Буквенные клавиши в нижнем регистре печатают строчные буквы, а в верхнем регистре (при нажатой клавише ***Shift***) печатают прописные (заглавные) буквы. Цифровые клавиши в нижнем регистре печатают цифры, а в верхнем – символы. Если возникает необходимость напечатать несколько прописных букв подряд, то удобнее включить режим фиксации прописных букв (нажать клавишу ***Caps Lock***). Переход в нижний регистр – повторное нажатие этой же клавиши. Действие клавиши ***Caps Lock*** на цифровые клавиши не распространяется. Переход в русский алфавит и обратно осуществляется нажатием двух клавиш ***Shift***, или какой-нибудь комбинацией из клавиш ***Shift, Alt, Ctrl*** в зависимости от установки программы-русификатора.

Функциональные клавиши свое назначение меняют в зависимости от текущей прикладной программы. Клавиша ***F1*** почти всегда подсказка, а клавиша ***F10*** – выход.

Esc (escape) используется для прекращения выполнения команды.

Курсор – указатель места на экране, где будет происходить действие. Он имеет вид мерцающей черточки или прямоугольника.

Tab передвигает курсор вправо на несколько позиций.

Ctrl включает управляющий режим, при котором клавиши выполняют не ввод литер, а команды или функции.

Alt всегда используется с другими клавишами для выполнения команд или функций.

Функциональные клавиши свое назначение меняют в зависимости от текущей прикладной программы. Клавиша ***F1*** почти всегда подсказка, а клавиша ***F10*** – выход.

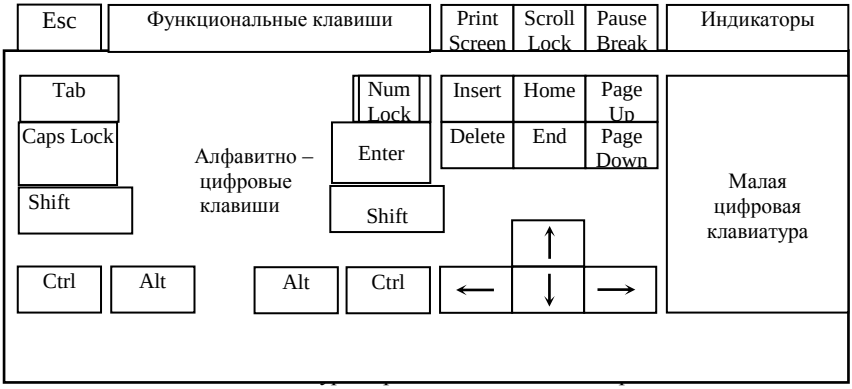
Esc (escape) используется для прекращения выполнения команды.

Курсор – указатель места на экране, где будет происходить действие. Он имеет вид мерцающей черточки или прямоугольника.

Tab передвигает курсор вправо на несколько позиций.

Ctrl включает управляющий режим, при котором клавиши выполняют не ввод литер, а команды или функции.

Alt всегда используется с другими клавишами для выполнения команд или функций.



Функциональные клавиши свое назначение меняют в зависимости от текущей прикладной программы. Клавиша ***F1*** почти всегда подсказка, а клавиша ***F10*** – выход.

Esc (escape) используется для прекращения выполнения команды.

Курсор – указатель места на экране, где будет происходить действие. Он имеет вид мерцающей черточки или прямоугольника.

Tab передвигает курсор вправо на несколько позиций.

Ctrl включает управляющий режим, при котором клавиши выполняют не ввод литер, а команды или функции.

Alt всегда используется с другими клавишами для выполнения команд или функций.

Enter используется для завершения ввода команды.

BackSpace (bs, в некоторых клавиатурах←) расположена выше клавиши ***Enter***. При нажатии на эту клавишу курсор перемещается на позицию влево, стирая находящийся там символ.

Print Screen позволяет вывести на принтер содержимое экрана.

Insert меняет режим ввода символов в ранее набранный текст: ввод с раздвижкой символов (вставка) или ввод с заменой символов (замена).

Delete удаляет символ, находящийся над курсором, со сдвигом остального текста на позицию влево.

Клавиши со стрелками предназначены для перемещения курсора в указанном направлении.

Scroll Lock переключает клавиши со стрелками в такой режим, при котором они перемещают вниз или вверх не курсор, а сам текст. Обычно эта клавиша программируется для выполнения других функций и редко соответствует первоначальному назначению.

Pause/Break временно приостанавливает выполнение программы. Для продолжения следует нажать любую клавишу. ***Ctrl+Break*** прерывает выполнение программы.

Home перемещает курсор в начало строки.

End перемещает курсор в конец строки.

Page Up перемещает курсор на страницу вверх.

Page Down перемещает курсор на страницу вниз.

Малая цифровая клавиатура является дублирующей и удобна при вводе чисел.

NumLock переключает режимы малой цифровой клавиатуры. При включенном соответствующем световом индикаторе будут печататься цифры. При выключенном индикаторе клавиши малой цифровой клавиатуры управляют курсором.

Видеомонитор

Видеомонитор является устройством вывода информации. На его экран выводятся результаты работы компьютера.

Для генерации видеосигнала монитору в системном блоке имеется специальное устройство – видеоадаптер (видеокарта). Информация об изображении на экране хранится в видеопамяти – специальных микросхемах видеоадаптера (рис.1.3).

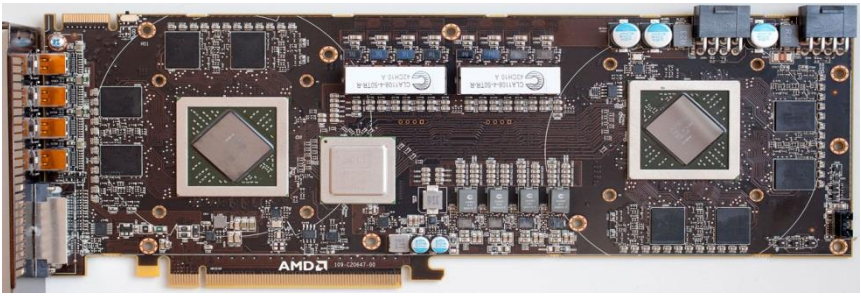


Рис. 1.3. Двухъядерная видеокарта без системы охлаждения

Все современные видеокарты совмещают в себе функции собственно видеоадаптера для вывода двухмерного изображения и видеоускорителя для работы с трехмерной графикой. Для этого видеокарта содержит специальный графический процессор, который выполняет большую часть расчетов для выводимой картинки, тем самым, разгружая центральный процессор.

Основными характеристиками видеокарты являются тактовые частоты видеопроцессора и видеопамяти, количество работающих исполнительных блоков внутри графического процессора, ширина шины видеопамяти (влияет на количество передаваемых памятью данных за один такт) и объем видеопамяти. Как правило, современные графические адаптеры имеют несколько выходов с одинаковыми или разными графическими интерфейсами для подключения разнообразных мониторов и телевизоров. Сейчас наиболее распространенными являются аналоговый интерфейс VGA и цифровые: DVI, HDMI (miniHDMI), DisplayPort (miniDP). Последние два, помимо видеоряда передают и звук.

Поскольку при работе видеоадаптеры требуют интенсивного обмена данными с оперативной памятью, для их подключения используют высокоскоростные шины *AGP* или *PCI-Express*, имеющие специальные разъемы на материнской плате. При этом видеоадаптер, так же как и другие устройства, может быть интегрированным. Однако в этом случае, как правило, его производительность ниже, чем у внешнего устройства.

Качество изображения видеомонитора определяется его разрешающей способностью – количеством точек (пикселей), изображаемых на экране по горизонтали и вертикали, а также минимальным размером точки.

Все современные мониторы относятся к типу ***SVGA*** и в зависимости от графического режима поддерживают разрешение: 640

× 480, 800 × 600, 1024 × 768, 1280 × 1024, 1600 × 1200 точек и более при воспроизведении от 16 до 16 млн. цветов.

Любой видеомонитор может работать в двух режимах: текстовом и графическом. В текстовом режиме на экране помещается чаще всего 80 столбцов и 25 строк. В каждой ячейке (знакоместе) может размещаться только один из 256 определенных символов. В графическом режиме изображение может быть произвольным, так как оно составляется непосредственно из точек (пикселей).

Дополнительные устройства компьютера

Мышь является дополнительным устройством ввода информации. В последнее время широкое распространение получили оптические мыши. В них специальный оптический сенсор делает с высокой частотой снимки участка поверхности под мышью, подсвечиваемого светодиодом или лазером. Полученные снимки обрабатывает специализированный процессор, который и определяет направление движения и скорость перемещения мыши.

С помощью мыши объект на экране (при поддержке программного обеспечения) может быть указан, выделен, открыт или перемещен.

Принтер служит для вывода информации на бумагу. Существует несколько основных типов принтеров.

Матричный принтер формирует изображение на бумаге с помощью точечной матрицы. Печатающая головка матричного принтера содержит набор тонких металлических стержней (иголок). Головка движется вдоль печатаемой строки, а стержни в нужный момент ударяют по бумаге через красящую ленту, образуя черные точки. Из отдельных точек формируются буквы, символы и элементы графического изображения.

Струйный принтер формирует изображение микрокаплями специальных чернил. Этот способ обеспечивает более высокое качество печати, удобен для цветной печати.

Лазерные принтеры в настоящее время обеспечивают самое высокое качество печати. В этих принтерах изображение переносится на бумагу со специального барабана, к которому за счет облучения лазером электрически притягиваются частички порошка (полимерной краски – тонера). После переноса на бумагу тонер закрепляется (вплавляется в лист).

Сканер – устройство для считывания в компьютер графической и текстовой информации. С помощью специального программного обеспечения компьютер распознает во введенной картинке символы и

преобразует их в компьютерные коды. Это позволяет во много раз увеличить скорость ввода информации.

Модем представляет собой плату системного блока или отдельное устройство, с помощью которого цифровой сигнал компьютера преобразуется в телефонный электромагнитный сигнал и наоборот. Это позволяет обмениваться информацией с другими компьютерами, объединенными в локальную сеть или подключенными к глобальной сети Интернет. Такой тип доступа называется коммутируемым (*Dial-Up*).

Одной из основных характеристик модема является скорость передачи данных, которая измеряется в бит/секунду. Современные модемы имеют максимальную скорость 56 Кбит/с.

Устройства, сочетающие возможности модема и средства для обмена факсимильными изображениями, называются факс-модемами.

Наряду с устройствами, обеспечивающими аналоговую передачу данных по телефонной линии, существуют другие типы модемов, использующие иные принципы работы. Одной из их разновидностей являются модемы, поддерживающие технологию *xDSL* (Digital Subscriber Line – цифровая абонентская линия). Такой модем также подключается к телефонной линии, но передает данные в цифровом виде. Это достигается путем частотного уплотнения линии. Данные передаются и принимаются в диапазоне частот, не используемом телефонной связью (свыше 4000 Гц).

Существует несколько разновидностей технологии *xDSL*. Наиболее распространенной в нашей стране является *ADSL* (Asymmetric *DSL*). При ее использовании максимальная скорость получения данных пользователем достигает 26 Мбит/с, отправки – 1 Мбит/с.

Плоттер (графопостроитель) выводит чертежи на бумагу. Как правило, плоттеры используют в системах проектирования (САПР).

Содержание работы

1. Ознакомьтесь с теоретическим материалом.
2. Занесите в отчет описание устройств **ввода**, входящих в состав закрепленного за Вами или Вашего домашнего компьютера.
3. Занесите в отчет описание устройств **вывода**, входящих в состав закрепленного за Вами или Вашего домашнего компьютера.
4. Занесите в отчет сведения о системе и комплектации компьютера.
5. Сделайте вывод о производительности компьютера.

Контрольные вопросы

1. Какие устройства относятся к основным?
2. Какие устройства относятся к дополнительным?
3. Назовите функциональные возможности основных устройств.
4. Какие функциональные возможности дополнительных устройств?
5. Что входит в состав системного блока?
6. Какие основные элементы материнской платы?
7. Каково назначение и основные характеристики процессора?
8. Какие вы знаете виды памяти?
9. Каково назначение BIOS?
10. Каково назначение кэш-памяти?
11. Назовите разновидности внешних носителей информации.
12. Как организовано хранение информации на жестких дисках?
13. Какие типы оптических дисков вам известны?
14. Охарактеризуйте группы клавиш клавиатуры.
15. Перечислите основные характеристики монитора.
16. Опишите устройство и принципы функционирования сканеров.
17. Какие разновидности модемов вы знаете?

Лабораторная работа № 2

ИЗУЧЕНИЕ IP-АДРЕСАЦИИ В ЛОКАЛЬНЫХ СЕТЯХ. МАСКИ. ПОДСЕТИ И НАДСЕТИ.

Цель работы: научиться переводить числа из десятично-точечной нотации в двоичную и обратно, разбивать адресное пространство на подсети, строить надсети.

Краткие теоретические сведения:

Для обмена данными в частной TCP/IP-сети или через Интернет, каждый сетевой узел должен обладать уникальным 32-битным IP-адресом. IP-адреса делятся на общие и частные. Первые уникальны в глобальном масштабе и используются для адресации в Интернете. Вторые ограничены диапазонами, которые обычно используются в частной сети, но не видны из Интернета.

Общие IP-адреса. Каждый IP-адрес в Интернете уникален. Для обеспечения такой уникальности адресов сетей в Интернете организация IANA (Internet Assigned Numbers Authority) разделила

незанятую часть пространства IP-адресов и делегировала полномочия по их распределению региональным реестрам, среди которых Asia-Pacific Network Information Center (APNIC), American Registry for Internet Numbers (ARIN) и Reseaux IP Europeens (RIPE NCC). Региональные регистраторы выделяют блоки адресов небольшому количеству крупных поставщиков интернет-услуг (ISP), которые затем выдают более мелкие блоки своим клиентам и менее крупным провайдерам. Как правило, интернет-провайдер выдает по одному общему IP-адресу на каждый напрямую подключенный к провайдеру компьютер. Этот IP-адрес может назначаться динамически в момент подключения компьютера к ISP или статически закрепляться за выделенной линией или модемным подключением.

Частные IP-адреса. Часть IP-адресов никогда не используется в Интернете. Они называются частными и используются для организации адресации в сетях, которые «не видны» в общей сети. Например, пользователю, объединяющему компьютеры в домашнюю TCP/IP-сеть, не надо назначать общие IP-адреса каждому узлу — он использует частные адреса. Диапазоны частных адресов варьируются:

Начальный адрес	*Конечный адрес*
10.0.0.0	10.255.255.255
172.16.0.0	172.31.255.255
192.168.0.0	192.168.255.255

Узлы с частными IP-адресами могут подключаться к Интернету через прокси-сервер или компьютер с Windows, сконфигурированный в качестве NAT-сервера (Network Address Translation). Windows также поддерживает сервис общего доступа к Интернету (Internet Connection Sharing, ICS), предоставляющий клиентам частной сети упрощенные сервисы NAT.

Методы IP- адресации. IP-адреса могут назначаться вручную, динамически (DHCP-сервером) или автоматически [например, с помощью APIPA (Automatic Private IP Addressing)].

Ручная IP – адресация. Назначение IP-адресов вручную используется нечасто, но иногда без него не обойтись. Например, ручное конфигурирование потребуется в сети, состоящей из нескольких сегментов, при отсутствии DHCP-сервера, или если IP-адрес DHCP-сервера также назначается вручную. Наконец, важным сетевым серверам, например DNS- или WINS-серверу или контроллеру домена, обычно назначают статические IP-адреса. Статические IP-адреса можно выделить по механизму резервирования DHCP-адресов, но большинство администраторов предпочитает не перепоручать это

дело DHCP-серверу и назначают их вручную. Во всех остальных случаях ручное конфигурирование рекомендуется, только если невозможно использовать DHCP. Администрирование назначенных вручную IP-адресов отнимает много времени и чревато ошибками, особенно в средних и крупных сетях.

Протокол DHCP. DHCP-сервер автоматически выделяет DHCP-клиентам IP-адреса из заданных администратором диапазонов. DHCP-сервер можно настроить на конфигурирование других параметров TCP/IP, например адресов DNS- и WINS-серверов, основных шлюзов и т. п.

Автоматическое назначение частных IP-адресов

APIPA (Automatic Private IP Addressing) служит для автоматического назначения адресов и применяется в простых односегментных сетях без DHCP-сервера (см. главу 1).

Альтернативная конфигурация

Подобно APIPA, альтернативная конфигурация позволяет назначить IP-адрес компьютерам, которым недоступен DHCP-сервер. Однако в отсутствие такого сервера компьютер с альтернативной конфигурацией не сможет использовать APIPA, даже если этот протокол будет доступен в сети. Эта функция полезна, когда компьютер работает в нескольких сетях, в одной из которых нет DHCP-сервера. Например, портативный компьютер, используемый для работы в офисе и дома. В обеих сетях используется один и тот же адаптер и локальное подключение, настроенное на автоматическое получение IP-адреса. При подключении к корпоративной сети параметры TCP/IP настраиваются DHCP-сервером. Дома DHCP-сервера нет, поэтому используется определенная альтернативная конфигурация: IP-адрес, маска подсети и основной шлюз для домашней сети.

Структура IP-адреса. IP-адреса привычно представляется в форме четырех чисел, разделенных точкой, например 192.168.100.22. Однако это лишь одна из форм IP-адреса, которая называется *десятично-точечной нотацией* и используется для удобства запоминания адреса. В компьютере применяется двоичная нотация, в которой все числа представлены только цифрами 1 и 0. Это «родная» форма IP-адреса. Логика IP-адресации становится понятной при рассмотрении двоичной версии IP- адреса. Для конфигурирования, управления и устранения неполадок IP-адресации надо уметь работать с IP-адресами в двоичной форме, а также переводить их из двоичного в десятичное представление и обратно.

Преобразование двоичного и десятичного представлений. В

десятично-точечной нотации каждое 32-битное число IP-адреса представляется в виде четырех десятичных групп, значение каждой из которых лежит в диапазоне 0—255, например 192.168.0.225. Эти числа представляют четыре 8-битных значения, составляющих 32-битный адрес. В любой нотации каждая из четырех групп называется октет. Но только двоичная форма позволяет наглядно увидеть значение каждого бита. Например, IP-адрес 192.168.0.225 в двоичной форме выглядит так: 11000000 10101000 00000000 11100001. В IP-адресах октеты и биты считаются слева направо. Первый октет соответствует первому слева, а биты с 1 по 8 соответствуют первым восьми битам, начиная с самого левого. Второй октет— это следующие восемь битов (9—16), затем идет третий октет (биты 17—24), а замыкает последовательность четвертый октет (биты 25—32). В десятично - точечной нотации октеты отделяются точками, а в двоичной — пробелами.

В табл. 2.1 показаны экспоненциальное и десятичное представление битов в двоичном октете. Обратите внимание: если смотреть слева направо, то первый бит дает значение 128, а каждый последующий бит — половину значения предыдущего. И, наоборот, в направлении справа налево, начиная с восьмого бита (значение 1), цена каждого последующего бита в два раза больше, чем предыдущего.

Обратите внимание, что вклад бита в общую сумму ненулевой, только если он содержит 1. Например, если первый бит — 1, ему соответствует десятичное значений 128. Если же его значение — 0, то и десятичное значение равно нулю. Октету со всеми битами; равными 1, соответствует десятичное значение 255. Если все биты содержат 0, десятичное значение октета равно 0.

Таблица 2.1. Представление битов октета в двоичной нотации

Октет	**Первый бит**	**Второй бит**	**Третий бит**	**Четвер-тый бит**	**Пятый бит**	**Шестой бит**	**Седьмой бит**	**Восьмой бит**
Экспоненциальное представление	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**
Десятичное представление	128	**64**	**32**	**16**	**8**	**4**	**2**	**1**
Пример	**1**	**0**	**1**	**0**	**1**	**1**	**0**	**0**

Пример перевода из двоичной нотации в десятичную.

Пусть первый октет IP-адреса в двоичном представлении выглядит так: **10101100**

Первый, третий, пятый и шестой биты содержат 1, а остальные — 0. Для упрощения решения нарисуем таблицу перевода, в которой отобразим возможные веса битов октета (см. табл. 2.2.):

Таблица 2.2. Пример *перевода двоичного числа в десятичное*

128	**64**	**32**	**16**	**8**	**4**	**2**	**1**
1	**0**	**1**	**0**	**1**	**1**	**0**	**0**

Сложим десятичные эквиваленты каждого бита и найдем десятичную сумму октета: 1-й бит (128) + 3-й бит (32) + 5-й бит (8) + 6-й бит (4) = сумма октета (172) Поскольку сумма составляет 172, первый октет нашего IP-адреса в десятичной форме равен 172. Применив этот же метод, можно преобразовать полный IP-адрес вида 10101100 00010001 00000111 00011011 в десятично-точечное представление: 172.17.7.27.

Пример перевода из десятичной нотации в двоичную. Перевод октета из десятичной формы в двоичную осуществляется записью 1 или 0 в соответствующий бит октета слева направо, пока не будет получено искомое десятичное число. Если запись 1 в очередной бит приводит к тому, что полученная сумма превосходит десятичное число, просто запишите в этот бит 0 и перейдите к следующему. Допустим, надо перевести IP-адрес 172.31.230.218 в двоичный вид. Первым делом запишите последовательность возможных весов битов в таблицу 2.3:

Таблица 2.3. Таблица для перевода чисел в двоичную систему счисления

128	64	32	16	8	4	2	1
							—

Начнем с первого числа — 128. Поскольку 128 меньше 172, запишем 1 в первый бит, а наша промежуточная сумма будет 128. Затем посмотрим вес второго бита — 64. Так как 128 + 64 больше 172, второй бит установим в 0. Затем перейдем к третьему биту, вес которого — 32. 128 и 32 в сумме дают меньше 172, поэтому запишем в этот бит 1. Промежуточная сумма становится 128 + 0 + 32=160. Перейдем к четвертому биту, его вес — 16. 160 и 16 в сумме дают больше 172, поэтому пишем 0. Вес пятого бита — 8. Сумма 160 + 8 меньше 172, пишем в пятый бит 1, а промежуточная сумма становится 128 + 0 + 32 + 0 + 8 = 168. И, наконец, вес шестого бита — 4, сумма 168

и 4 равна 172, т. е. искомому числу. Поэтому пишем 1 в шестой бит, а оставшиеся седьмой и восьмой биты заполняем нулями.

Таким образом, первый октет в двоичной форме выглядит так: 10101100

Выполнив аналогичные операции с остальными октетами получим двоичное представление адреса 172.31.230.218: 10101100 00011111 11100110 11011010

Перевод между системами счисления с помощью калькулятора. При помощи *Калькулятора* эта операция выполняется намного быстрее. Чтобы воспользоваться функцией перевода между системами счисления, в меню **Вид** (View) выберите **Инженерный (Scientific)** и установите переключатель в положение **Dec** или **Bin** (в зависимости от того, из какой системы счисления необходимо перевести число). Например, для перевода двоичного числа 11001100 в десятичное представление, отметьте Bin, введите двоичное число. После ввода двоичного числа просто установите Dec и получите число в десятичном представлении.

Примечание: как и в десятичной нотации, калькулятор отбрасывает крайние левые нули, октет 00001110 отображается как 1110. Поэтому необходимо контролировать число отображаемых в окошке калькулятора битов, чтобы не перепутать значения. Например, легко спутать двоичное число 1100001 (десятичное 97) с 11000001 (десятичное значение 193). Если число бит меньше 8, для представления октета IP-адреса надо добавить необходимое количество нулей слева.

Идентификаторы сети и узла. Маршрутизаторы, переправляющие пакеты данных между TCP/IP-сетями не обязаны знать, какому именно узлу предназначен тот или иной IP-пакет. Вместо этого маршрутизатор считывает из IP-пакета только адрес сети, в которой находится узел — приемник пакета, а затем на основе своей таблицы маршрутизации определяет, каким образом доставить пакет в сеть, в которой расположен адресат. Точное местоположение узла определяется только после доставки пакета в нужный сегмент сети.

Такой механизм маршрутизации возможен благодаря делению IP-адреса на два компонента:

• *идентификатор сети* (network ID) — первая часть IP-адреса, представляющая конкретную сеть в более крупной TCP/IP-сети (например в Интернете);

• *идентификатор узла* (host ID) — вторая часть IP-адреса, определяющая узел TCP/IP (рабочую станцию, сервер, маршрутизатор или любое другое TCP/IP- устройство). Например, разбиение IP-адреса

(131.107.16.200) на идентификаторы сети (это первые два октета — 131.107) и узла (последние два октета — 16.200).

Маска подсети. Еще один необходимый для нормальной работы TCP/IP параметр — *маска подсети* (subnet mask), которая служит для определения, в какой сети находится приемник пакета — локальной или внешней. Маска подсети — это 32-битный адрес, представляющий собой последовательность битов со значением 1, который используется для выделения, или *маскировки,* идентификатора сети адреса назначения пакета и отделения идентификаторов сети и узла. Каждому узлу сети TCP/IP нужна маска подсети (если сеть не разбита на подсети, т. е. состоит из одной подсети) или маска по умолчанию (в случае разбиения сети на подсети).

Например, такое 32-битное число представляет маску подсети по умолчанию для узлов с адресами класса B (например 172.20.16.200):

11111111 11111111 00000000 00000000 (255.255.0.0)

Когда TCP/IP-узел с адресом 172.20.16.200 отправляет пакет по адресу 172.21.17.201, он сначала выполняет побитовую операцию **И** по отношению к локальным адресу и маске подсети. Поскольку эта логическая операция в результате дает 0 во всех битах кроме тех, в который в обоих операндах стояли 1, то 172.20.16.200 И 255.255.0.0 = 172.20.0.0.

Затем узел повторяет эту операцию, но вместо адреса отправителя подставляет адрес получателя. В результате получается 172.21.0.0. Затем TCP/IP сравнивает результаты этих операций. Если они совпадают, получатель расположен в этой же подсети. Иначе приемник и получатель расположены в разных подсетях.

Длина префикса сети в маске подсети. Поскольку биты идентификатора сети всегда идут последовательно и начинаются с самого левого, самый простой способ показать маску подсети — это указать количество битов идентификатора сети в виде *префикса сети.* Таким образом, маска подсети выражается в виде IP-адрес/префикс сети. Например, IP-адрес 131.107.16.200 и маску подсети 255.255.0.0 можно записать в виде 131.107.16.200/16. Число 16 после слеша обозначает количество единичных битов в маске подсети. Точно так же, /24 обозначает маску подсети 255.255.255.0 для адреса класса C, например 206.73.118.23/24 (см. табл. 2.4).

Примечание: нотация с префиксом сети также известна как *бесклассовая междоменная маршрутизация* (Classless Interdomain Routing, CIDR).

Таблица 2.4. Маски по умолчанию различных классов сетей

Класс адреса	Маска подсети по умолчанию в двоичном виде	Префикс сети и десятичный эквивалент
Класс А	11111111 00000000 00000000 00000000	/8 = 255.0.0.0
Класс В	11111111 1111111100000000 00000000	/16 = 255.255.0.0
Класс С	11111111 11111111 11111111 00000000	/24 =-255.255.255.0

Основной шлюз. Связь между TCP/IP-узлами разных сетей, как правило, выполняется через маршрутизаторы. *Маршрутизатор* — это устройство с несколькими интерфейсами, подключенными к разным сетям, а *маршрутизация* — процесс приема IP-пакетов на одном интерфейсе и пересылка их на другой интерфейс в направлении адресата. С точки зрения узла сети TCP/IP, *основной шлюз* — это IP-адрес маршрутизатора, сконфигурированного на пересылку IP-трафика в другие сети. Пытаясь передать информацию другому узлу IP-сети, компьютер определяет тип узла (локальный или удаленный) по маске подсети. Если узел-получатель расположен в локальном сегменте сети, пакет направляется в локальную сеть по методу широковещания. В противном случае компьютер пересылает пакет в основной шлюз, определенный в параметрах TCP/IP. Обязанность дальнейшей пересылки пакета в нужную сеть возлагается на маршрутизатор, адрес которого указан в качестве основного шлюза.

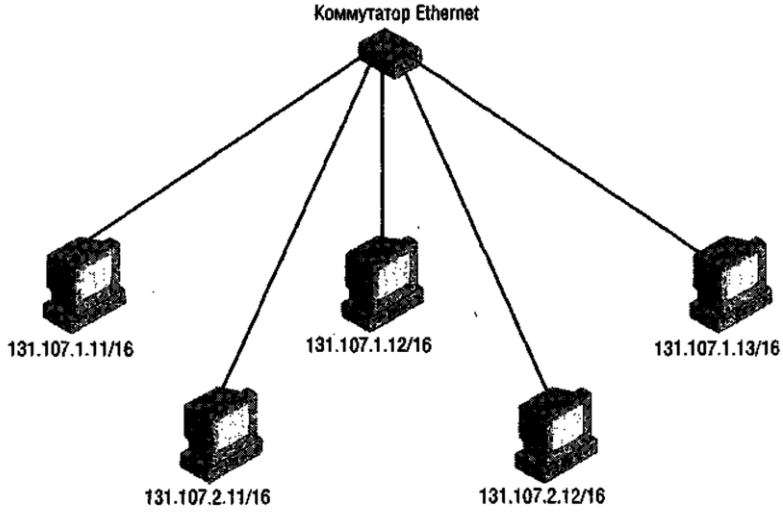
Разбиение IP-сетей на подсети и создание надсетей. Маски подсети позволяют настраивать адресное пространство в соответствии с требованиями к сети. Разбиение на подсети позволяет организовать иерархическую структуру сетей, а надсети и CIDR позволяют объединить разные сети в едином адресном пространстве.

Разбиение на подсети. Маски подсети помогают определить, как IP-адрес разбивается на идентификаторы сети и узла. В адресах классов А, В и С применяются стандартные маски подсети, занимающие соответственно первые 8, 16 и 24 бита 32-битового адреса. Подсетью называется логическая сеть, определяемая маской подсети.

Стандартные маски годятся для сетей, которые не предполагается разбивать. Например, в сети из 100 компьютеров, соединенных с помощью карт гигабитного Ethernet, кабелей и коммутаторов, все узлы могут обмениваться информацией по локальной сети. Сеть не нуждается в маршрутизаторах для защиты от чрезмерного широковещания или для связи с узлами, расположенными в отдельных физических сегментах. В таком простом случае вполне достаточно идентификатора сети класса С.

Механизм разбиения на подсети. *Разбиение на подсети* (subnetting) — это логическое разделение адресного пространства сети путем установки в «1» дополнительных битов маски подсети. Такое расширение позволяет создавать многие подсети в адресном пространстве сети.

Например, если маска подсети по умолчанию 255.255.0.0 используется для узлов сети класса B 131.107.0.0, IP-адреса 131.107.1.11 и 131.107.2.11 находятся в одной подсети и поддерживают взаимодействие посредством широковещания. Но если расширить маску подсети до 255.255.255.0, то эти адреса окажутся в разных подсетях и для обмена данными соответствующим узлам придется пересылать пакеты на основной шлюз, который перенаправит дейтаграммы в нужную подсеть. Внешние по отношению к сети узлы по-прежнему используют маску подсети по умолчанию для взаимодействия с узлами внутри сети. Обе версии показаны на рис. 2.1 и 2.2.



Единственный сегмент сети 131.107.0.0/16

Рис. 2.1. Неразбитое на подсети адресное пространство класса B

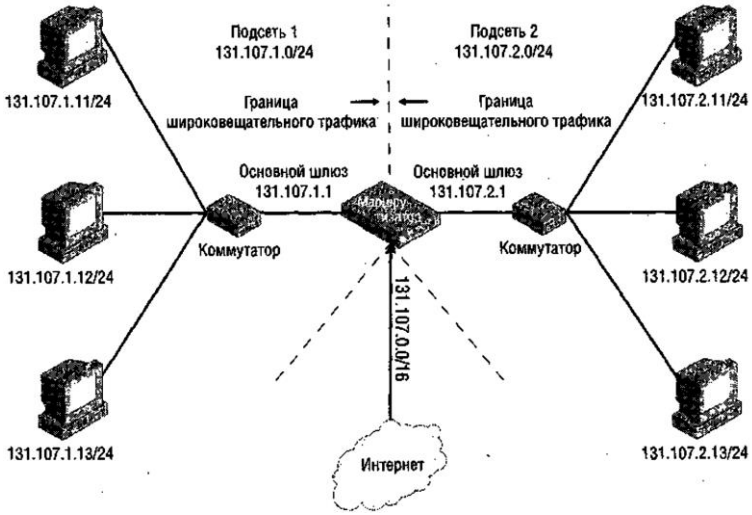


Рис. 2.2. Разбитое на подсети адресное пространство класса B

Показанное на рис. 2.1 исходное адресное пространство класса B, состоящее из единственной подсети, может содержать максимум 65 534 узлов, а новая маска подсети (рис. 2.2) позволяет разделить адресное пространство на 256 подсетей, в каждой из которых можно разместить до 254 узлов.

Преимущества разбиения на подсети. Разбиение на подсети часто используют для обеспечения соответствия физической и логической топологии сети или для ограничения широковещательного трафика. Другие несомненные преимущества: более высокий уровень защиты (благодаря ограничению неавторизованного трафика маршрутизаторами) и упрощение администрирования (благодаря передаче управления подсетями другим отделам или администраторам).

Соответствие физической топологии. Допустим, вам поручили спроектировать университетскую сеть, состоящую из 200 узлов, распределенных в четырех зданиях — Voter Hall, Twilight Hall, Monroe Hall и Sunderland Hall. В каждом здании планируется разместить по 50 узлов. Если интернет-провайдер выделил адрес 208.147.66.0 класса C, вам доступны адреса 208.147.66.1 - 208.147.66.254. Однако из-за размещения в четырех физически отделенных зданиях, узлы не могут обмениваться данными по локальной сети. Расширив маску подсети на 2 бита (т. е. позаимствовав их у идентификатора узла), сеть разбивают

на четыре логические подсети, а для связи устанавливается маршрутизатор (рис. 2.3).

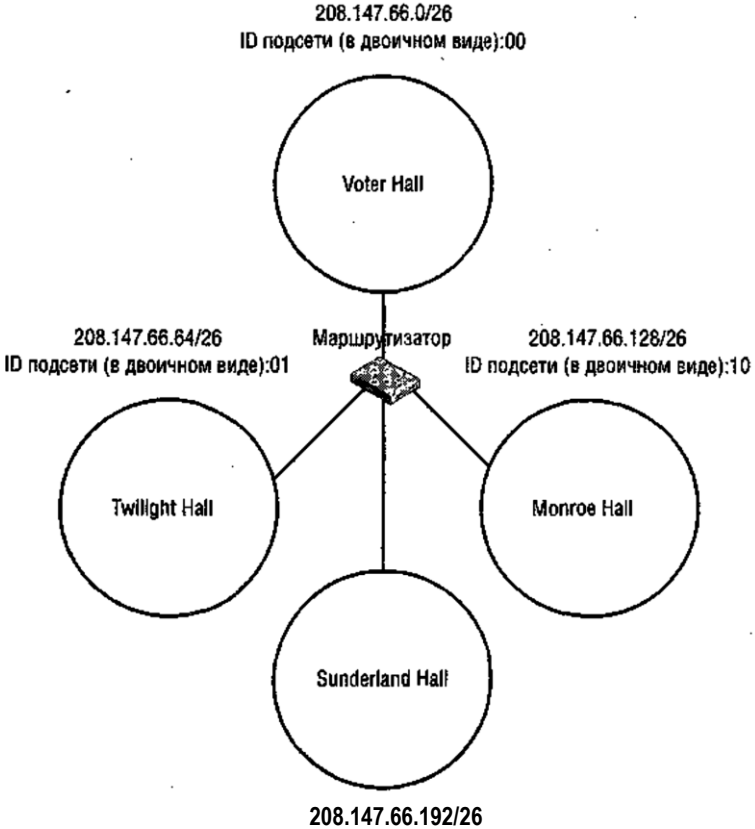


Рис. 2.3. Разбиение на подсети в соответствии с физической топологией

Ограничение широковещательного трафика. Широковещание — рассылка сообщений с одного компьютера на все расположенные в локальном сегменте устройства. Широковещание существенно нагружает ресурсы, поскольку занимает полосу пропускания и требует участия всех сетевых адаптеров и процессоров логического сегмента сети.

Маршрутизаторы блокируют широковещание и защищают сети от излишнего трафика. Поскольку маршрутизаторы также определяют логические ограничения подсетей, разбиение на подсети позволяет косвенно ограничивать широковещательный трафик в сети.

Определение максимального количества узлов в сети. Зная сетевой адрес, определить максимальное количество узлов в сети просто: надо возвести 2 в степень, равную количеству битов в идентификаторе узла, отведенных для идентификации узлов, то есть «нулей» - s, и вычесть 2. Например, в сетевом адресе 192.168.0.0/24 под идентификатор узла отведено 8 бит, поэтому возможное максимальное число узлов 2^8 — 2 = 254.

Исключение идентификаторов узла, состоящих из одних нулей или одних единиц. Значение 2x показывает общее количество комбинаций значений битов двоичного числа *x,* включая комбинации из одних нулей и одних единиц. Например, *V* дает 8, т. е. количество различных комбинаций из 3 битов *(dec.* означает десятичную систему счисления):

000 = 0 (дес.)
001 = 1 (дес.)
010 = 2 (дес.)
011 = 3 (дес.)
100 = 4 (дес.)
101 = 5 (дес.)
110 = 6 (дес.)
111 = 7 (дес.)

Однако узлам нельзя назначать адреса, состоящие из одних только нулей или единиц, поскольку они зарезервированы для других целей. Идентификатор узла, состоящий из одних нулей, на самом деле определяет сеть без указания конкретного узла. Идентификатор узла из одних единиц зарезервирован в протоколе IP для широковещания (передачи сообщения всем узлам сети). При подсчете максимального количества узлов в сети эти варианты надо исключить из рассмотрения (т. е. вычесть из него 2).

Определение емкости подсети. При увеличении количества битов в маске подсети для создания подсетей в адресном пространстве идентификатор узла укорачивается, и создается новое адресное пространство для идентификатора подсети (рис. 2.4).

Чтобы определить количество доступных в адресном пространстве подсетей, просто возведите 2 в степень *y,* где *y* — количество бит в идентификаторе подсети. Например, если в адресном пространстве 172.16.0.0/16 выделить 8 бит на адрес подсети (т. е. привести к виду 172.16.0.0/24), количество доступных подсетей станет 2s, или 256. Из него не надо вычитать 2, поскольку большинство современных маршрутизаторов принимают идентификаторы подсетей только из единиц или нулей.

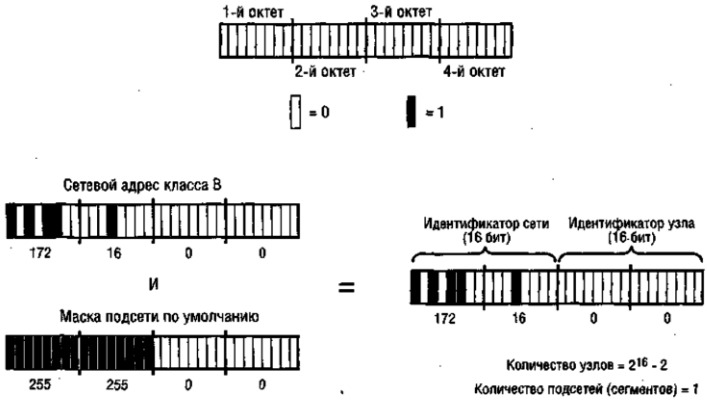


Рис. 2-10. Адресное пространство класса B по умолчанию

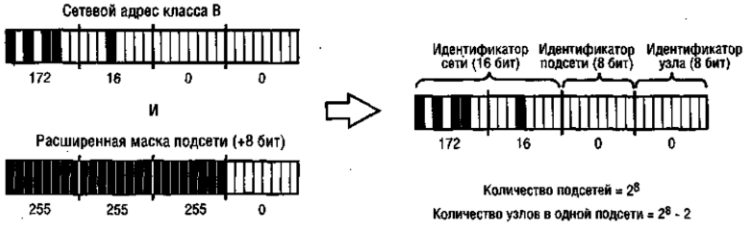


Рис. 2.4. Адресное пространство с идентификатором подсети

Планируя адресное пространство и маски подсети убедитесь, что отведенных на идентификатор подсети бит достаточно для размещения всех подсетей, а также обеспечен резерв для расширения сети в будущем. Помните, что любую физическую сеть надо рассматривать как подсеть.

Калькулятор позволяет быстро определить необходимое число бит для идентификатора подсети. Вычтите 1 из требуемого количества подсетей в десятичном формате, переведите результат в двоичный вид и посчитайте количество бит в нем. Например, если нужна 31 подсеть, введите 30 и установите переключатель **Bin.** Полученное число НПО говорит, что под идентификатор подсети нужно зарезервировать 5 бит.

Количество узлов в подсети. Количество идентификаторов узлов в подсети определяется так же, как и узлов в сети — оно равно 2x — 2, где *x* — количество бит в идентификаторе узла. Например, в адресе 172.16.0.0/24 резервируется 8 бит под идентификатор узла, поэтому число узлов в подсети равно 28 — 2, т. е. 254. Для вычисления

количества узлов во всей сети умножают полученный результат на количество подсетей.

В нашем примере адресное пространство 172.16.0.0/24 дает 254 сетей x=256, узлов = 65 024.

Конфигурируя адресное пространство и маски подсети в соответствии с требованиями сети убедитесь, что отвели на идентификатор узла достаточно бит с учетом возможного увеличения количества узлов в подсети в будущем.

Калькулятор позволяет быстро определить необходимое количество бит для идентификатора узла. Прибавьте 1 к требуемому количеству узлов в подсети в десятичном формате, переведите результат в двоичный вид и посчитайте количество бит в нем. Например, если нужно 33 узла в подсети, введите 34 и установите переключатель **Bin.** Результат 100010 говорит о том, что нужно зарезервировать 6 бит под идентификатор подсети.

Примеры подсетей. В предыдущем примере мы расширили маску подсети для адресного пространства 172.16.0.0/16 до 255.255.255.0, увеличив ее на целый октет. На практике маску расширяют более мелкими порциями, вплоть до отдельных битов.

Например, на рис. 2.5 показано адресное пространство 10.0.0.0/12. Поскольку адрес относится к классу А, количество единичных битов в маске подсети по умолчанию равно 8. Мы расширили его на 4 бита, т. е. 4 бита отдано под идентификатор подсети, а оставшиеся 20 служат идентификатором узла. В такой сети диапазон адресов первой подсети 10.0.0.1-10.15.255.254.

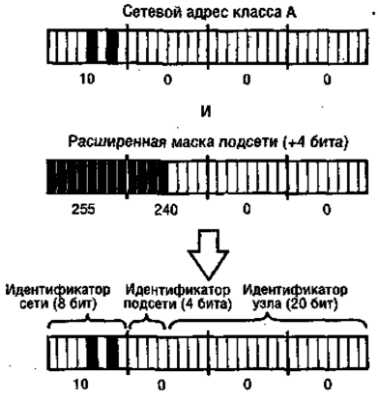
На рис. 2.6 адресу класса В 172.20.0.0 назначена нестандартная маска подсети 255.255.248.0, расширяющая маску по умолчанию на 5 бит. В такой сети диапазон адресов первой подсети 172.20.0.1-172.20.7.254.

На рис.2.7 показан адрес класса С 192.168.0.0/26. В этом примере 2 бита зарезервированы для идентификатора подсети и 6 — под идентификатор узла. В такой сети диапазон адресов первой подсети 192.168.0.1—192.168.0.62.

Определение диапазонов адресов подсети. Десятично-точечная форма маски подсети позволяет определить диапазоны IP-адресов в каждой подсети простым вычитанием из 256 числа в соответствующем октете маски.

Например, в сети класса С с адресом 207.209.68.0 с маской подсети 255.255.255.192 вычитание 192 из 256 даст 64. Таким образом, новый диапазон начинается после каждого 64 адреса: 207.209.68.0-207.209.68.63, 207.209.68.64-207.209.68.127 и т.д. В сети класса В

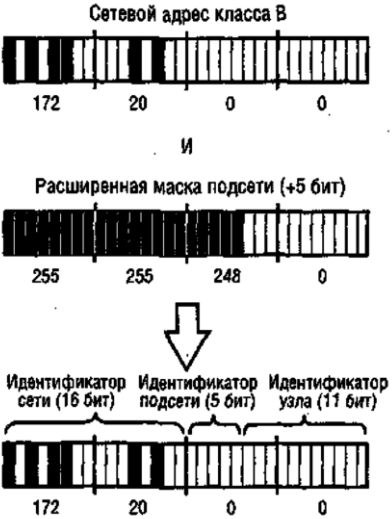
131.107.0.0 с маской подсети 255.255.240.0 вычитание 240 из 256 дает 16. Следовательно, диапазоны адресов подсетей группируются по 16 в третьем октете, а четвертый октет принимает значения из диапазона 0-255: 131.107.0.0-131.107.15.255, 131.107.16.0-131.107.31.255 и т. д.



Кол-во подсетей $2^4 = 16$

Кол-во узлов в подсети $2^{20} - 2 = 1048574$

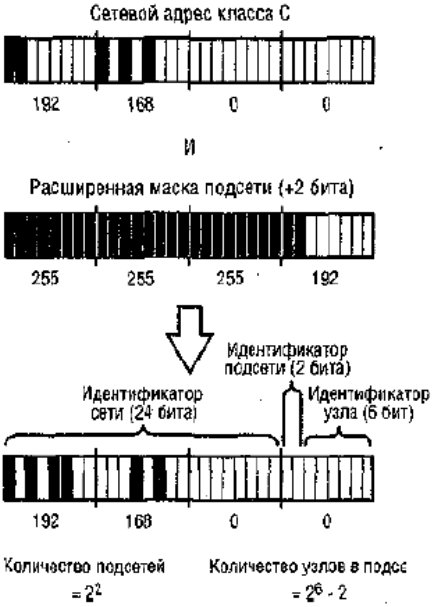
Рис. 2.5. Разделение на подсети адресного пространства класса А



Кол-во подсетей $2^5 = 32$

Кол-во узлов в подсети $2^{11} - 2 = 2046$

Рис. 2.6. Разделение на подсети адресного пространства класса В



Кол-во подсетей $2^2 = 4$

Кол-во узлов в подсети $2^6 - 2 = 62$

Рис. 2.7. Разделение на подсети адресного пространства класса С

Узлам нельзя назначать идентификаторы из одних нулей или единиц, так что исключаются первый и последний адрес каждого диапазона.

Сложение маршрутов путем создания надсетей. Чтобы предотвратить истощение доступных идентификаторов сетей старших классов, организации, ответственные за адресацию в Интернете, предложили схему, называемую *созданием надсетей* (supernetting), согласно которой несколько сетей (маршрутов) можно объединить (или сложить) в единую более крупную сеть. Надсети позволяют эффективнее управлять выделением участков адресного пространства.

Допустим, организации нужно объединить в сеть 2000 узлов. Это слишком много для одной сети класса С, которая поддерживает не более 254 узлов. Сеть класса В поддерживает 65 534 узла, но таких сетей возможно всего 16 383 и количество свободных стремительно сокращается. Интернет-провайдеру нет смысла (да и возможности) выделять сети класса В клиентам, которые будет использовать только 3% диапазона адресов.

Надсети позволяют Интернет-провайдеру выделить клиенту блок адресов класса С, который будет рассматриваться как единая сеть, представляющая собой нечто среднее между классами С и В. В нашем примере блок из 8 идентификаторов сети класса С даст возможность организации объединить в сеть до 2032 узлов.

Как работают надсети. Надсети отличаются от подсетей тем, что заимствуют биты идентификатора сети и маскируют их как идентификатор узла. Допустим, интернет-провайдер выделил блок из 8 адресов сети: 207.46.168.0—207.46.175.0. Если определить на маршрутизаторах провайдера и всех узлов сети маску подсети /21 (вместо /24 по умолчанию), все сети будут казаться единственной сетью из-за того, что их идентификаторы (урезанные до 21 бита) будут выглядеть одинаково (рис. 2.8).

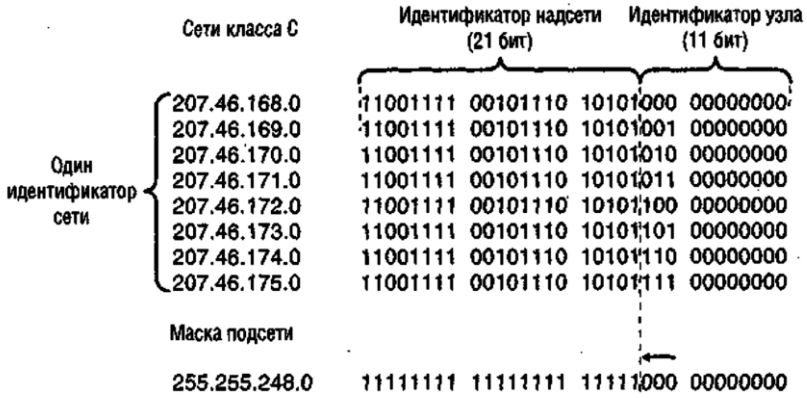


Рис. 2.8. Надсеть на основе блока адресов класса С

Использование бесклассовой междоменной маршрутизации. CIDR — это эффективный метод поддержки надсетей с помощью таблиц маршрутизации. Не будь CIDR, в таблицах маршрутизации следовало бы размещать отдельные записи для каждой сети в надсети, а так вся надсеть представляется одной записью (рис. 2.9).

Выделенные региональными регистраторами Интернета или Интернет-провайдером блоки адресов надсети часто называют CIDR-блоками, а термин CIDR часто используется для обозначения самих надсетей.

CIDR не совместим с устаревшим протоколом RIP (Routing Information Protocol) версии 1, который применялся в старых маршрутизаторах, и требует, чтобы маршрутизатор использовал

бесклассовый протокол маршрутизации, такой как RIP версии 2 или OSPF (Open Shortest Path First).

До применения CIDR

Таблица маршрутизации маршрутизатора B		
207.46.168.0	255.255.255.0	207.46.168.1
207.46.169.0	255.255.255.0	207.46.168.1
207.46.170.0	255.255.255.0	207.46.168.1
207.46.171.0	255.255.255.0	207.46.168.1
207.46.172.0	255.255.255.0	207.46.168.1
207.46.173.0	255.255.255.0	207.46.168.1
207.46.174.0	255.255.255.0	207.46.168.1
207.46.175.0	255.255.255.0	207.46.168.1

После применения CIDR

Таблица маршрутизации маршрутизатора B		
207.46.168.0	255.255.248.0	207.46.168.1

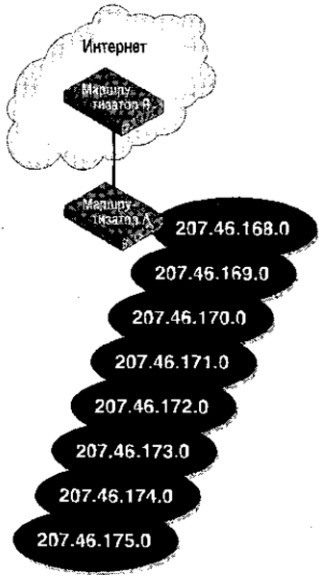


Рис. 2.9. Использование CIDR для упрощения создания надсетей

Будущее адресного пространства. Использование CIDR для выделения адресов дает новую жизнь идентификаторам сети. CIDR-блок из предыдущего примера (207.46.168.0, 255.255.248.0) можно рассматривать двояко:

- как блок 8 адресов сетей класса C;
- как адресное пространство, в котором зафиксирован 21 бит, а 11 битов доступны для изменения.

Во втором случае идентификаторы сети освобождаются от классовой наследственности и становятся частью бесклассового пространства IP-адресов. Каждый идентификатор сети независимо от длины представляет адресное пространство, в котом биты идентификатора сети зафиксированы, а биты узла можно менять. Биты узла можно использовать в качестве идентификаторов узлов или в других целях (допустим, для организации подсетей) и таким образом наилучшим образом удовлетворить потребности организации в поддержке сетей.

Маски подсети переменной длины. Традиционно все узлы и маршрутизаторы организации используют одну маску подсети. В этом

случае сеть может разбиваться на подсети, в которых максимальное количество идентификаторов узлов одинаковое.

Однако поддержка *масок подсети переменной длины* (variable-length subnet mask, VLSM) позволяет маршрутизаторам обслуживать разные маски. Чаще всего VLSM применяют для разбиения на подсети самих подсетей. Допустим, большой организации принадлежит большое адресное пространство 131.107.0.0/16. Внешние маршрутизаторы для определения идентификатора сети используют первые 16 бит адреса и в соответствии с этим осуществляют маршрутизацию. При получении данных из Интернета маршрутизаторы организации используют маску подсети /22 для перенаправления трафика в любой из 64 региональных отделений организации. А маршрутизаторы региональных офисов в свою очередь используют маску подсети /25 для маршрутизации трафика в 8 отделов в рамках отделения.

Как и **CIDR,** работа масок подсетей переменной длины основана на бесклассовых протоколах маршрутизации, таких как **RIP** версии **2** и **OSPF. VRLM** несовместим с более старыми протоколами маршрутизации (например с RIP версии 1).

Использование VLSM для поддержки подсетей разного размера. VLSM также позволяет разбивать сеть на подсети разных размеров на одном уровне иерархии и более эффективно использовать адресное пространство.

Например, если одна подсеть должна объединять 100 компьютеров, вторая - 50, а третья - 20, то не удастся обойтись традиционной маской по умолчанию для единственного идентификатора сети класса С. Как видно из табл. 2.5, никакая из масок подсети по умолчанию не обеспечивает одновременно достаточное число подсетей и узлов в подсети.

Таблица 2.5. Параметры масок подсети класса С (статические)

Сетевой адрес	Число подсетей	Число узлов в подсети
208.147.66.0/24	1	254
208.147.66.0/25	2	126
208.147.66.0/26	4	62
208.147.66.0/27	8	30

В таких ситуациях проблему решает VLSM. При этом не надо обращаться к Интернет-провайдеру за новым диапазоном адресов.

При разбиении на подсети различного размера нужно использовать специальный шаблон с завершающими нулями; сеть класса С

поддерживает до семи подсетей. Завершающие нули нужны для предотвращения пересечения адресных пространств подсетей.

Увеличение количества доступных узлов средствами VLSM. Обратите внимание, что в табл. 2.6 седьмая (и последняя) подсеть имеет такое же количество узлов, как и шестая, отличаются только идентификаторы подсети, да и то всего одним битом (в идентификаторе 7-й сети в отличие от остальных отсутствует завершающий нуль). Можно не использовать все семь подсетей — достаточно определить состоящий из одних единиц идентификатор подсети на любом уровне, который заменит все перечисленные в следующих строках таблицы подсети. Например, определить идентификатор подсети 1111, который заменит подсети 5—7 (см. табл. 2-6). Благодаря этому вы получите еще одну подсеть с 14 узлами вместо 3 подсетей, вместе содержащих только 10 узлов. Это позволит максимизировать количество узлов, которые вмещает сеть, состоящая из 5 подсетей.

Если сеть класса С разбита на 3, 5, 6 или 7 подсетей, VLSM позволяет максимизировать количество доступных узлов.

Если идентификатор подсети с маской переменной длины соответствует шаблону из табл.2.6, подсети не пересекутся, и адреса будут интерпретироваться однозначно.

Таблица 2.6. Идентификация подсети на основе VLSM

Номер подсети	Идентификатор подсети (двоичный)	Маска подсети	Количество узлов	Пример адреса подсети
1	**0**	255.255.255.128	126	208.147.66.0.0/25
2	**10**	255.255.255.192	62	208.147.66.0.128/26
3	**110**	255.255.255.224	30	208.147.66.0.192/27
4	**1110**	255.255.255.240	14	208.147.66.0.224/28
5	**11110**	255.255.255.248	6	208.147.66.0.240/29
6	**111110**	255.255.255.252	2	208.147.66.0.248/30
7	**1111111**	255.255.255.252	2	208.147.66.0.252/30

На рис. 2.10 показано, как с помощью VLSM построить 3 сети с 100, 50 и 20 узлами соответственно.

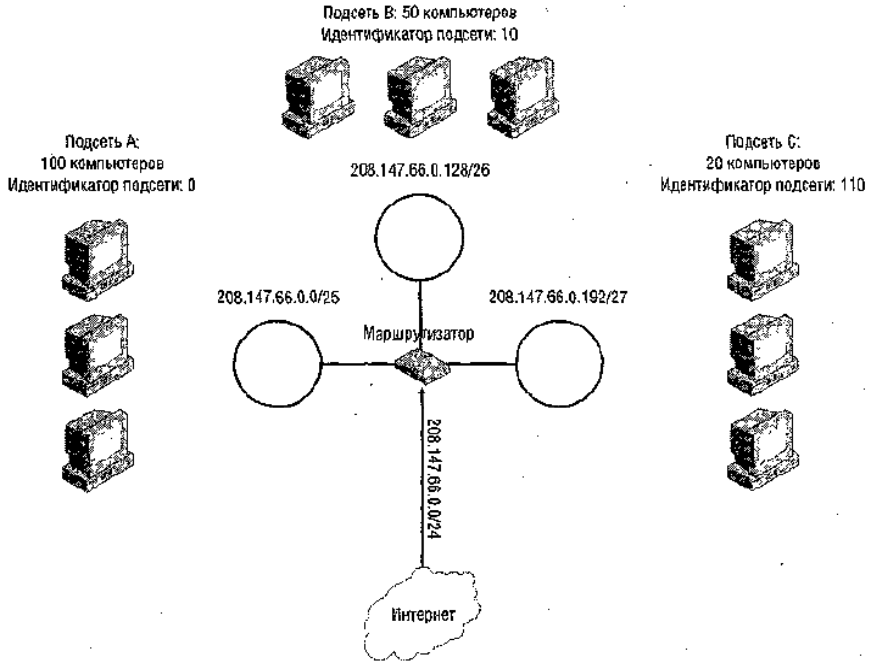


Рис. 2.10. VLSM дает дополнительную гибкость при разбиении на подсети

Задания к практической работе:

1. Перевод числа из десятичного представления в двоичное вручную посредством табл. 2.3. Исходные данные представлены в табл. 2.7.

Таблица 2.7. Исходные данные к заданию 1

Вар.	Числа	Вар.	Числа	Вар.	Числа
1.	105; 224	**11.**	51; 133	**21.**	135; 95
2.	97; 251	**12.**	30; 201	**22.**	44; 146
3.	163; 35	**13.**	198; 177	**23.**	88; 137
4.	24; 132	**14.**	122; 55	**24.**	62; 128
5.	154; 67	**15.**	65; 146	**25.**	241; 89
6.	201; 22	**16.**	233; 48	**26.**	79; 117
7.	11;156	**17.**	26; 157	**27.**	150; 71
8.	34; 41	**18.**	80; 139	**28.**	15; 149
9.	33;187	**19.**	140; 77	**29.**	196; 18
10.	90; 62	**20.**	83; 111	**30.**	27; 166

2. Перевод числа из двоичного представления в десятичное вручную посредством табл. 1.2. Исходные данные приведены в табл.2.8.

Таблица 2.8. Исходные данные к заданию 2

Вар.	Число	Вар.	Число	Вар.	Число
1.	10110111	**11.**	00110110	**21.**	00111011
2.	00101001	**12.**	10110110	**22.**	10001101
3.	11101101	**13.**	10110101	**23.**	10001101
4.	10111101	**14.**	00011011	**24.**	11001100
5.	11110101	**15.**	11101011	**25.**	10110000
6.	10110111	**16.**	11011101	**26.**	10010001
7.	11111011	**17.**	10110110	**27.**	10101010
8.	11001100	**18.**	00011000	**28.**	10110110
9.	10100011	**19.**	11011010	**29.**	10110101
10.	10100101	**20.**	10101010	**30.**	11011111

3. Преобразование маски подсети из десятично-точечной формы в форму с префиксом сети и обратно. Привести маску в двоичном виде, в форме префикса. Исходные данные приведены в табл. 2.9.

Таблица2.9. Исходные данные к заданию 3

Вар.	Маска; префикс	Вар.	Маска; префикс
1.	255.255.255.128; /15	**16.**	128.0.0.0; /26
2.	255.255.252.0; /12	**17.**	248.0.0.0; /27
3.	255.255.192.0; /13	**18.**	255.255.128.0; /28
4.	255.255.248.0; /14	**19.**	255.255.248.0; /29
5.	255.224.0.0; /15	**20.**	255.255.255.240; /30
6.	255.240.0.0; /19	**21.**	255.255.255.248; /31
7.	255.255.255.252; /17	**22.**	255.192.0.0; /7
8.	255.255.255.192; /18	**23.**	252.0.0.0; /12
9.	255.254.0.0; /19	**24.**	255.0.0.0; /9
10.	240.0.0.0; /20	**25.**	254.0.0.0; /10
11.	192.0.0.0; /21	**26.**	255.248.0.0; /11
12.	255.224.0.0; /22	**27.**	255.255.255.0; /5
13.	255.252.0.0; /23	**28.**	255.255.224.0; /3
14.	255.255.252.0; /18	**29.**	224.0.0.0; /6
15.	255.128.0.0; /25	**30.**	255.255.224.0; /4

4. Вычисление масок подсети. Выделен адрес сети. Рассчитайте маску сети и запишите её в десятично-точечной нотации, а также в

виде префикса, для каждой задачи, отталкиваясь, в первом случае, от требований к количеству подсетей, а во втором – к количеству узлов. Укажите класс рассчитанной сети в каждой из задач. Исходные данные для обеих задач приведены в табл. 2.10.

Таблица 2.10. Исходные данные к заданию 4

Вар.	**Исходный префикс сети**	**Требуемое кол-во подсетей**	**Требуемое кол-во узлов**
1.	10.54.2.0/24	20	1520
2.	192.168.0.0/25	7	418
3.	196.52.251.0/10	42	122
4.	207.83.100.0/24	9	541
5.	172.22.0.0/16	6	3200
6.	15.12.0.0/8	18	1932
7.	154.23.85.0/24	5	643
8.	10.23.2.0/16	20	2312
9.	192.167.0.0/16	14	567
10.	192.168.0.0/24	5	164
11.	191.168.0.0/16	8	2389
12.	100.124.0.0/8	31	12453
13.	172.3.85.0/24	2	142
14.	190.145.23.0/18	9	4856
15.	211.56.84.0/24	7	120
16.	175.96.41.0/17	56	780
17.	14.53.0.0/9	32	452
18.	164.8.0.0/16	4	2147
19.	174.123.0.0/16	6	8632
20.	153.145.0.0/16	9	452
21.	233.54.93.0/23	12	180
22.	14.59.73.0/17	4	270
23.	192.54.82.0/22	8	135
24.	14.152.0.0/18	5	9641
25.	18.23.4.0/19	3	1236
26.	238.124.0.0/19	11	578
27.	211.162.0.0/25	16	111
28.	198.63.2.0/16	25	368
29.	182.10.5.0/16	17	451
30.	14.2.8.0/10	16	97

5. Вычисление диапазонов адресов подсети. В этом упражнении нужно вычислить диапазоны адресов подсети, определив диапазоны

первых трех подсетей сети. Для каждого адреса сети и маски подсети (столбец А), вычтите из 256 значение соответствующего октета маски подсети. Запишите полученное значение в колонку В. Затем впишите в колонку С первые четыре кратные единицы (начните с 0) этого значения. С помощью этих значений заполните колонки D и E, как показано в примере (см. рис.1.11). Исходные данные представлены в табл.2.11.

(A) Адрес сети и маска подсети	(B) Число групп	(C) Первые четыре кратные единицы B (включая 0)	(D) Начальный адрес диапазонов адресов первых трех подсетей	(E) Конечный адрес диапазонов адресов первых трех подсетей
10.0.0.0 255.240.0.0	256 - 240 = 16	0, 16, 32, 48	10.0.0.0, 10.16.0.0, 10.32.0.0	10.15.255.255, 10.31.255.255, 10.47.255.255

Рис. 2.11. Пример выполнения задания 5

Таблица 2.11. Исходные данные к заданию 5

Вар.	**Исходный префикс сети**	**Вар.**	**Исходный префикс сети**
1.	10.54.2.0/24	**16.**	175.96.41.0/17
2.	192.168.0.0/25	**17.**	14.53.0.0/9
3.	196.52.251.0/10	**18.**	164.8.0.0/16
4.	207.83.100.0/24	**19.**	174.123.0.0/16
5.	172.22.0.0/16	**20.**	153.145.0.0/16
6.	15.12.0.0/8	**21.**	233.54.93.0/23
7.	154.23.85.0/24	**22.**	14.59.73.0/17
8.	10.23.2.0/16	**23.**	192.54.82.0/22
9.	192.167.0.0/16	**24.**	14.152.0.0/18
10.	192.168.0.0/24	**25.**	18.23.4.0/19
11.	191.168.0.0/16	**26.**	238.124.0.0/19
12.	100.124.0.0/8	**27.**	211.162.0.0/25
13.	172.3.85.0/24	**28.**	198.63.2.0/16
14.	190.145.23.0/18	**29.**	182.10.5.0/16
15.	211.56.84.0/24	**30.**	14.2.8.0/10

6. Проверка двух адресов на принадлежность одной подсети. С помощью логической функции “И” калькулятора можно определить, принадлежат ли два адреса одной и той же логической подсети. Необходимо выполнить две операции “И” над соответствующими октетами маски подсети и октетами заданного IP-адреса. Если

результаты совпадут, адреса принадлежат одной логической подсети. Например, при маске подсети 255.255.255.240 и IP-адресах 192.168.0.220 и 192.168.0.190, выражение “240 И 220” дает 208, а “240 И 190” - 176. Они отличаются, поэтому адреса принадлежат разным логическим подсетям. Исходные данные представлены в табл. 2.12.

Таблица 2.12. Исходные данные к заданию 6

Вар.	**IP-адрес 1**	**IP-адрес 2**	**Маска**
1.	10.54.2.24	10.54.2.48	255.255.255.128
2.	192.168.0.25	192.168.2.5	255.255.252.0
3.	196.52.251.10	196.52.240.97	255.255.192.0
4.	207.83.100.24	207.83.71.39	255.255.248.0
5.	172.22.0.16	172.1.3.8	255.224.0.0
6.	15.12.0.8	15.122.0.7	255.240.0.0
7.	154.23.85.24	154.23.54.1	255.255.255.252
8.	10.23.2.16	10.23.9.2	255.255.255.192
9.	192.167.0.6	192.154.0.78	255.254.0.0
10.	192.168.0.4	192.52.14.87	240.0.0.0
11.	191.168.0.1	171.168.0.85	192.0.0.0
12.	100.124.0.8	100.128.98.4	255.224.0.0
13.	172.3.85.24	183.54.78.6	255.252.0.0
14.	190.145.23.18	190.145.17.5	255.255.252.0
15.	211.56.84.24	211.128.5.4	255.128.0.0
16.	175.96.41.17	39.214.5.7	128.0.0.0
17.	14.53.0.9	10.5.87.5	248.0.0.0
18.	164.8.0.16	164.8.8.8	255.255.128.0
19.	174.123.0.16	174.123.95.7	255.255.248.0
20.	153.145.0.16	153.145.0.31	255.255.255.240
21.	233.54.93.23	233.251.8.5	255.255.255.248
22.	14.59.73.17	14.59.73.45	255.192.0.0
23.	192.54.82.22	198.0.56.21	252.0.0.0
24.	14.152.0.18	15.124.5.6	255.0.0.0
25.	18.23.4.19	19.34.52.6	254.0.0.0
26.	238.124.0.19	238.124.98.4	255.248.0.0
27.	211.162.0.25	211.162.89.5	255.255.255.0
28.	198.63.2.16	198.63.9.1	255.255.224.0
29.	182.10.5.6	111.0.0.3	224.0.0.0
30.	14.2.8.10	14.2.22.39	255.255.224.0

Контрольные вопросы:

1. Понятие IP-адреса, маски, шлюза.
2. Что такое префикс сети?
3. Что такое префикс маски?
4. Классы сетей. Свойства.
5. Механизм построения надсети.
6. Механизм разбиения на подсети.

Лабораторная работа № 3

ИЗУЧЕНИЕ СТРУКТУРЫ, СВОЙСТВ И СОЗДАНИЕ ТАБЛИЦ РЕЗЯЦИОННОЙ БАЗЫ ДАННЫХ

Цель лабораторной работы: закрепление теоретических знаний, полученных при изучении разделов курса «Информационные технологии», посвященных рассмотрению понятий: структура данных, реляционная таблица и ее параметры, ключи и виды упорядоченности баз данных (БД), приобретение навыков разработки и формирования таблиц реляционных БД.

Задание для самостоятельной работы: изучить краткие теоретические сведения; изучить состав, структуру и параметры реляционных таблиц БД; ознакомиться с программой MS Access.

Краткие теоретические сведения

База данных (БД) – это совокупность сведений, относящихся к определенной теме или задаче.

Реляционные БД – базы данных с табличной формой организации информации. Реляционная БД состоит из одной или нескольких взаимосвязанных двумерных таблиц.

СУБД (Система Управления Базами Данных) – программное обеспечение для работы с базами данных. Большинство современных СУБД предназначены для работы с реляционными базами данных.

Данные – это все, что вы хотите сохранить и к чему намерены обращаться неоднократно. В Microsoft Access данными могут быть тексты, числа, даты и картинки. Если, например, вы продаете книги, то можете хранить их названия, изображения обложек, координаты авторов, количество пачек на складе, цены на них, даты продаж.

Запись – строка таблицы. Одна запись содержит информацию об отдельном объекте, описываемом в БД.

База данных в Access представляет собой единый большой объект, который объединяет такие составляющие, как таблицы, отчеты, запросы, формы и так далее, и позволяет хранить их в едином дисковом файле с расширением *.mdb.

Достоинством системы управления базами данных (СУБД) MS Access является то, что она имеет очень простой графический интерфейс, который позволяет не только создавать собственную базу данных, но и разрабатывать простые и сложные приложения. Для анализа хранящейся в базе данных MS Access информации можно создавать различные отчеты, которые позволяют извлекать необходимые данные, группировать и сортировать их в нужном виде, вычислять итоговые значения по группам и в целом по всем отработанным записям. Отчеты могут быть дополнены рисунками, диаграммами, содержательными комментариями, могут быть выведены на печать, преобразованы в документ Word или опубликованы на Web.

Запуск MS Access и открытие баз данных. Для того чтобы запустить MS Access, необходимо нажать: кнопку «Пуск» на панели задач в нижней части рабочего стола;

открыть в главном меню пункт «Программы»;

выбрать программу MS Access.

Для того чтобы открыть существующую базу данных, можно воспользоваться одним из трех способов:

− выбрать в меню из списка ранее открывавшихся файлов требуемый файл.

- выбрать файл из списка в области задач, которая расположена в правой части окна приложения;

− выбрать команду «Открыть» в меню «Файл» и затем нужный файл в диалоговом окне «Открытие файла базы данных».

В последнем случае в диалоговом окне обычно по умолчанию отображается содержимое папки «Мои документы» или какой-либо другой (в зависимости от установленной на компьютере операционной системы). Найдя в списке необходимую базу данных, нужно выделить файл и нажать кнопку «Открыть» или дважды щелкнуть по элементу списка. В главном окне MS Access появится окно выбранной базы данных. Открыть файл можно в одном из четырех режимов:

в режиме коллективного доступа. При открытии файла базы данных в этом режиме доступ к нему будет разрешен другим пользователям, они могут открыть его и также работать с объектами

базы данных. Причем все пользователи будут иметь возможность как просматривать данные, так и изменять их;

в режиме монопольного доступа. В этом случае никто другой не сможет открыть этот файл, пока он используется;

в режиме коллективного доступа только для чтения. В этом случае никто не сможет изменять ни данные в таблицах, ни другие объекты базы данных;

в режиме монопольного доступа только для чтения. Открывается файл только для чтения и одновременно запрещается другим пользователям открывать этот файл. Выбрать нужный режим можно из списка, который раскрывается при нажатии стрелки справа от кнопки «Открыть». По умолчанию файл базы данных открывается в режиме коллективного доступа.

Создание новой базы данных. Для создания новой базы данных необходимо воспользоваться областью задач «Создание файла» окна приложения MS Access. Эта область появляется справа при первом запуске MS Access (рис.3.1). Если MS Access уже использовался для работы с каким- либо файлом и после его закрытия в главном окне этой области нет, можно:

выбрать команду «Создать» из меню «Файл»;

нажать кнопку «Создать» на панели инструментов.

Любое из этих действий приведет к открытию справа области задач «Создание файла».

После этого, чтобы создать базу данных, необходимо:

выбрать элемент «Новая база данных». Появится диалоговое окно «Файл новой базы данных» с содержимым папки «Мои документы»;

в раскрывающемся списке «Папка» открыть папку, в которой будет храниться создаваемая база данных;

в поле ввода «Имя файла» указать имя создаваемого файла и нажать кнопку «Создать». По умолчанию MS Access присваивает новой базе данных имя dbl, а если база с именем dbl уже существует, то db2 и т. д. После нажатия кнопки появится окно созданной базы данных.

Создание таблицы. Основными объектами MS Access являются таблицы. Список таблиц, составляющих базу данных приложения, появляется в окне базы данных при первом открытии приложения. Кроме этого, MS Access создает системные таблицы, в которых хранится информация обо всех объектах приложения, и эти таблицы при необходимости также могут отображаться в окне базы данных. Создать таблицу можно в режиме конструктора путем задания имен полей, их типов и свойств рис.3.2).

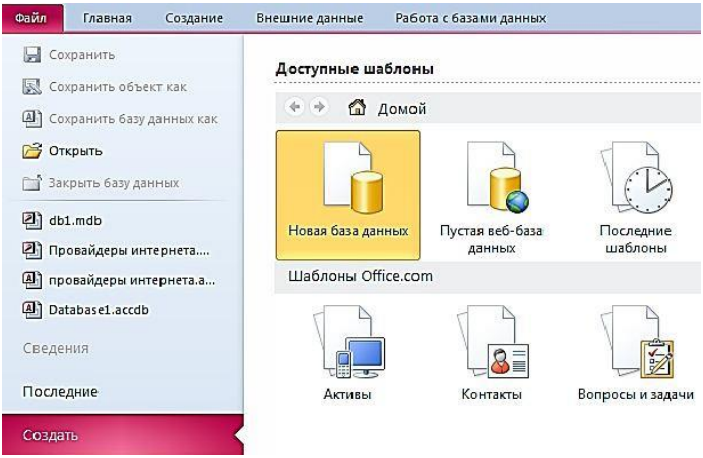


Рис.3.1. Создать новую базу данных

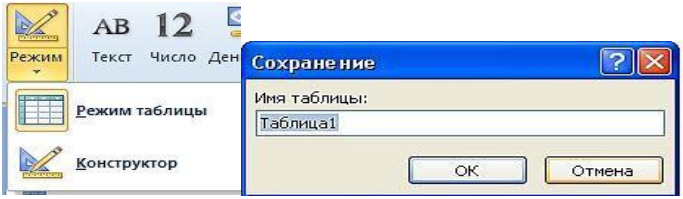


Рис.3.2. Создание новой таблицы в режиме конструктора

Для создания таблицы необходимо: дважды щелкнуть левой кнопкой мыши на ярлыке «Создание таблицы» с помощью конструктора или нажать на кнопку «Создать» в верхней части окна базы данных, выбрать из списка в окне «Новая таблица» элемент «Конструктор». В том и в другом случае откроется пустое окно конструктора таблиц; в окне конструктора таблиц в столбец «Имя поля» ввести имена полей создаваемой таблицы; в столбце «Тип данных» для каждого поля таблицы выбрать из раскрывающегося списка тип данных, которые будут содержаться в этом поле; в столбце «Описание» можно ввести описание данного поля (не обязательно); в нижней части окна конструктора таблиц на вкладках «Общие» и «Подстановка» ввести свойства каждого поля или оставить значения свойств, установленные по умолчанию; после описания всех полей будущей таблицы нажать кнопку

«Закрыть» (в верхнем правом углу окна таблицы); на вопрос «Сохранить изменения макета или структуры таблицы <имя таблицы>?» нажать кнопку
«Да»; в окне «Сохранить как» в поле «Имя таблицы» ввести имя создаваемой таблицы; в ответ на сообщение «Ключевые поля не заданы» и вопрос
«Создать ключевое поле сейчас?» нажать кнопку «Да», если ключевое поле необходимо, или кнопку «Нет», если такого не требуется.

При создании таблиц в режиме конструктора приходится задавать имена таблиц и полей в таблицах. При этом необходимо придерживаться ряда правил.

1. Имена полей в таблице не должны повторяться.

2. Имена полей могут содержать не более 64 символов, включая пробелы.

Имена полей могут содержать любые символы, включая буквы, цифры, пробелы, специальные символы, за исключением точки (.), восклицательного знака (!), апострофа (') и квадратных скобок ([ ]). Кроме того, имена полей не должны начинаться с пробела или управляющего символа.

После ввода имени поля необходимо задать тип данных, которые будут находиться в этом поле. Наиболее удобным способом является выбор типа из списка, но наименование типа можно ввести и вручную. Обычно при ручном вводе конструктор автоматически предлагает закончить вводимое название типа. В MS Access имеются следующие типы данных (рис.3.3):

текстовый – символьные или числовые данные, не требующие вычислений. Поле данного типа может содержать до 255 символов. Размер текстового поля задается с помощью свойства «Размер поля» (FieldSize), в котором указывается максимальное количество символов для данного поля. При необходимости ввести строку большего размера, чем указано в данном свойстве, надо значение свойства изменить, так как место под дополнительные символы не резервируется. Независимо от длины текстового поля оно хранится в базе данных MS Access в записи переменной длины.

поле MEMO – поле MEMO, предназначенное для ввода текстовой информации, по объему превышающей 255 символов. Такое поле может содержать до 65 535 символов. Этот тип данных отличается от типа «Текстовый» тем, что в таблице хранятся не сами данные, а ссылки на блоки данных, хранящиеся отдельно. За счет этого ускоряется обработка таблиц (сортировка, поиск и т.п.).

Поле типа MEMO не может быть ключевым или проиндексированным;

Рис.3.3.Типы данных полей таблицы

числовой – числовой тип, применяющийся для хранения числовых данных, используемых в математических расчетах, и имеющий несколько подтипов. От выбора подтипа (размера) данных числового типа зависит точность вычислений. Для задания подтипа числовых данных служит свойство «Размер поля». Данные этого типа могут содержаться в 1, 2, 4, 8 или 16 байтах. Обычно по умолчанию используется подтип «Длинное целое», который занимает 4 байта и представляет собой число в пределах от – 2 147 483 648 до +2 147 483 647. Но, кроме этого типа, можно указать «Байт» (1 байт), «Целое» (2 байта), «Одинарное с плавающей точкой» (4 байта), «Двойное с плавающей точкой» (8 байтов), «Десятичное» (12 байтов), «Код репликации» (16 байтов);

дата/время – тип для представления даты и времени. Он позволяет вводить даты с 100 по 9999 год. Размер поля – 8 байтов. Даты и время хранятся в специальном фиксированном числовом формате. Дата является целой частью значения поля этого типа, а время – его дробной частью. MS Access предоставляет большой выбор форматов отображения даты и времени;

денежный – тип данных, предназначенный для хранения данных, точность представления которых колеблется от 1 до 4 десятичных знаков. Целая часть данного типа может содержать до 15 десятичных знаков;

счетчик – поле, содержащее 4-байтный уникальный номер, определяемый MS Access автоматически для каждой новой записи либо случайным образом, либо путем увеличения предыдущего значения на 1. Значения полей, имеющих тип «Счетчик», обновлять нельзя. Максимальное число записей в таблице, имеющей поле «Счетчик», не должно превышать двух миллиардов;

логический – логическое поле, которое может содержать только два значения, интерпретируемых как «Истина»/«Ложь». Поля логического типа не могут быть ключевыми, но их можно индексировать;

поле объекта OLE, содержащее ссылку на OLE-объект (лист Excel, документ Word, звук, рисунок и т. п.). Объем объекта ограничивается имеющимся в наличии дисковым пространством. Тип объекта OLE не указывается в свойствах поля объекта OLE. Он зависит от приложения, использованного для создания объектов OLE, которые хранятся в этом поле. Упаковщик объектов позволяет внедрять файлы, созданные приложениями, которые не являются серверами объектов OLE. Например, он позволяет внедрить текстовые файлы в качестве объектов в таблицы MS Access. В поле объекта OLE могут храниться произвольные данные, в том числе и данные нескольких типов;

гиперссылка, дающая возможность хранить в поле ссылку на произвольный фрагмент данных внутри файла или Web-страницы на том же компьютере, в интранете или в Интернете. Гиперссылка состоит из четырех частей: отображаемого текста, адреса (путь к файлу или странице), дополнительного адреса (положение внутри файла или страницы) и текста всплывающей подсказки. Каждая часть гиперссылки может содержать до 2048 символов. Поле этого типа не может быть ключевым или индексированным.

Для большинства типов данных характерно свойство «Подпись». С помощью этого свойства можно задать названия полей таблицы, которые выводятся в различных режимах (в надписях, присоединенных к элементам управления формы, в заголовке столбца режима «Таблицы»; в строке заголовка в режиме «Формы», заголовке отчета, выводящегося в режиме предварительного просмотра). Поле может содержать до 2048 символов. Кроме того, для большинства типов данных существует свойство

«Обязательное поле», которое определяет необходимость ввода данных в это поле.

Свойство «Формат поля» указывает формат отображения данных из поля в режиме «Таблицы». Для определения формата полей текстового типа используются специальные символы форматирования.

Для числовых полей значение формата можно выбрать из раскрывающегося списка.

С помощью свойства «Маска ввода» указывается маска, позволяющая автоматизировать проверку ввода символов в поле. Она применяется к таким полям, как номер телефона, дата и т. д. Задавать маску ввода можно вручную или с помощью мастера.

Свойство «Индексированное поле» определяет, является ли данное поле индексированным, и если является, то в каком режиме. Существуют два режима индексирования: «Совпадения допускаются» и «Совпадения не допускаются». В первом случае поле может содержать повторяющиеся значения, во втором – нет.

Для большинства типов полей определено свойство «Значение по умолчанию». В этом свойстве указывается значение, автоматически добавляемое в поле для каждой новой записи, если это значение не введено пользователем.

Два свойства, которые тоже определены для большинства полей, позволяют выполнять проверку данных, вводимых в поле: «Условие на значение». Свойство определяет условие (ограничение), накладываемое на вводимые в это поле данные. При несоответствии вводимых данных указанному условию выдается сообщение об ошибке; «Сообщение об ошибке». Свойство определяет то сообщение, которое будет выдаваться пользователю, если при вводе данных не соблюдается условие, указанное в свойстве «Условие на значение». Если это свойство не указано, будет выдано стандартное сообщение об ошибке. Для таблицы можно, хотя и необязательно, ввести ключевые поля.

Ключевое поле – это одно или несколько полей, комбинация значений которых однозначно определяет каждую запись в таблице. Если для таблицы определены ключевые поля, то MS Access предотвращает дублирование или ввод пустых значений в ключевое поле. Ключевые поля используются для быстрого поиска данных и для установления связи между таблицами. В MS Access можно выделить три типа ключевых полей: счетчик, простой ключ и составной ключ.

Для создания ключевого поля первого типа необходимо в режиме конструктора таблиц:

- включить в таблицу поле счетчика;
- задать для него автоматическое увеличение на 1;
- указать это поле в качестве ключевого путем нажатия на кнопку

«Ключевое поле» на панели инструментов конструктора таблиц.

Если до сохранения созданной таблицы ключевые поля не были определены, то при сохранении будет выдано сообщение о создании ключевого поля. Для создания простого ключа достаточно иметь поле, которое содержит уникальные значения (например, коды или номера). Если выбранное поле содержит повторяющиеся или пустые значения, его нельзя определить как ключевое. Для определения записей, содержащих повторяющиеся данные, можно выполнить запрос на поиск повторяющихся записей. Если устранить повторы путем изменения значений невозможно, следует либо добавить в таблицу поле счетчика и сделать его ключевым, либо определить составной ключ. Составной ключ необходим в случае, если невозможно гарантировать уникальность записи с помощью одного поля. Он представляет собой комбинацию нескольких полей. Для определения составного ключа необходимо:

открыть таблицу в режиме конструктора;

выделить поля, которые необходимо определить как ключевые;

нажать кнопку «Ключевое поле» на панели инструментов конструктора таблиц.

Создание схемы данных. Связи между таблицами. Для установления связей между таблицами необходимо:

выполнить команду «Сервис» / Схема данных, откроется диалоговое окно «Добавление таблицы»; в диалоговом окне «Добавление таблицы» выделить таблицы, для которых необходимо установить связи, и нажать кнопку Добавить. Для установления связи между парой таблиц в схеме данных по простому ключу необходимо в главной таблице выделить ключевое поле, по которому устанавливается связь, и при нажатой кнопке мыши протащить курсор в соответствующее поле подчиненной таблицы.

После установления связи откроется диалоговое окно «Связи». При этом в окне «Тип отношения» автоматически установится тип «Один ко многим»; для установления связи по составному ключу необходимо выделить все поля, входящие в ключ главной таблицы, и перетащить их на одно из полей связи в подчиненной таблице. В окне «Связи» для каждого ключевого поля главной таблицы (таблица / запрос) выбрать соответствующее поле подчиненной таблицы (связанная таблица / запрос); задать параметр Обеспечение целостности данных; установить флажки Каскадное обновление связанных полей и Каскадное удаление связанных записей; нажать на кнопку «ОК».

Порядок выполнения лабораторной работы

В результате выполнения лабораторной работы каждый студент создает базу данных согласно его варианту, состоящую из таблиц с заполненными полями данных и установленными между каждой парой таблиц связями.

1. Создать рабочий каталог <disk>:\student\ТТП_№_\<FIO>.
2. Вызвать программу MS Access и создать новую базу данных с именем согласно варианту.
3. В режиме конструктора таблиц создать структуру таблиц, описание свойств которых приведено в табл. Звёздочкой отмечены ключевые поля. Размер и формат поля, подобрать самостоятельно.
4. В зависимость от варианта, предусмотреть маску ввода для поля Телефон, НомерПаспорта, НомерВодительскогоУдостоверения и т.д. При необходимости ввести условия на значения полей и задать соответствующие сообщения об ошибках в случае нарушения условий при вводе
5. Заполнить созданные ранее таблицы данными.
6. Создать схему данных путем установления групповых отношений (связей) между информационными объектами, представленными таблицами базы данных, используя данные таблиц.

Варианты 1, 13, 25. База данных «Спорт»

Таблица Спортсмен

Имя поля	**Тип данных**
*ID	Счетчик
Фамилия	Текстовый
Имя	Текстовый
Отчество	Текстовый
ГодРождения	Дата/время
Разряд	Числовой

Таблица Судья

Имя поля	**Тип данных**
*ID	Счетчик
Фамилия	Текстовый
Имя	Текстовый
Отчество	Текстовый
Категория	Текстовый

Таблица Соревнование

Имя поля	**Тип данных**
*ID	Счетчик
Название	Текстовый
ДатаНачала	Дата/время
Уровень	Текстовый
IDГлавногоСудьи	Числовой

Таблица Результат

Имя поля	**Тип данных**
*ID	Счетчик
IDСпортсмена	Числовой
IDДисциплины	Числовой
Результат	Дата/время
Стартовый Номер	Числовой

Таблица Дисциплины

Имя поля	**Тип данных**
*ID	Счетчик
Наименование	Текстовый
IDСудьи	Числовой
IDСоревнований	Числовой

Связи информационных объектов

Главный объект	**Подчиненный объект**	**Тип связи**
Судья	Соревнования	1:M
Спортсмен	Результат	1:M
Дисциплина	Результат	1:M
Судья	Дисциплина	1:M
Соревнование	Дисциплина	1:M

Варианты 2, 14, 26. База данных «Поликлиника»

Таблица Врач

Имя поля	**Тип данных**
*ID	Счетчик
Фамилия	Текстовый
Имя	Текстовый
Отчество	Текстовый
Специализация	Текстовый

Таблица Пациент

Имя поля	**Тип данных**
*ID	Счетчик
Фамилия	Текстовый
Имя	Текстовый
Отчество	Текстовый
ГодРождения	Дата/время

Таблица Больничный

Имя поля	**Тип данных**
*ID	Счетчик
IDПациента	Числовой
IDВрача	Числовой
ДатаНачала	Дата/время
ДатаОкончания	Дата/время

Таблица Приём

Имя поля	**Тип данных**
*ID	Счетчик
IDПациента	Числовой
IDВрача	Числовой
Дата	Дата/время
Время	Дата/время

Таблица Анализы

Имя поля	Тип данных
*ID	Счетчик
IDПациента	Числовой
IDИсследования	Числовой
IDВрача	Числовой
Дата	Дата/время
Результат	Числовой

Таблица Исследования

Имя поля	Тип данных
*ID	Счетчик
Название	Текстовый
Срок	Дата/время

Связи информационных объектов

Главный объект	Подчиненный объект	Тип Связи
Пациент	Больничный	1:M
Врач	Больничный	1:M
Пациент	Приём	1:M
Врач	Приём	1:M
Пациент	Анализы	1:M
Исследования	Анализы	1:M

Варианты 3, 15, 27. База данных «Университет»

Таблица Преподаватель

Имя поля	Тип данных
*ТабельныйНомер	Числовой
Фамилия	Текстовый
Имя	Текстовый
Отчество	Текстовый
ГодРождения	Дата/время

Таблица Студент

Имя поля	Тип данных
*НомерЗачётн Книжки	Числовой
Фамилия	Текстовый
Имя	Текстовый
Отчество	Текстовый
IDГруппы	Числовой

Таблица Группа

Имя поля	Тип данных
*ID	Счетчик
Название	Текстовый
ГодПоступления	Дата/время

Таблица Предмет

Имя поля	Тип данных
*ID	Счетчик
Название	Текстовый

Таблица Занятие

Имя поля	Тип данных
*ID	Счетчик
IDПредмета	Числовой
IDГруппы	Числовой
ТабельныйНомер	Числовой
Год	Дата/время
Семестр	Числовой

Связи информационных объектов

Главный объект	Подчиненный объект	Тип Связи
Группа	Студент	1:M
Предмет	Занятие	1:M
Группа	Занятие	1:M
Преподаватель	Занятие	1:M

Варианты 4, 16, 28. База данных «Грузоперевозки»

Таблица Автомобиль

Имя поля	Тип данных
*ID	Счетчик
Модель	Текстовый
ГосНомер	Текстовый
КатегорияАвто	Текстовый
Гркзоподъёмность	Числовой
ГодВыпуска	Дата/время

Таблица Перемещение

Имя поля	Тип данных
*ID	Счетчик
IDКлиента	Числовой
ПунктОтправления	Текстовый
ПунктПрибытия	Текстовый
Масса	Числовой
IDРейса	Числовой

Таблица Клиент

Имя поля	Тип данных
*ID	Счетчик
Фамилия	Текстовый
Имя	Текстовый
Отчество	Текстовый
Телефон	Числовой

Таблица Водитель

Имя поля	Тип данных
*НомерВодит Удостоверения	Числовой
Фамилия	Текстовый
Имя	Текстовый
Отчество	Текстовый
Категория	Текстовый

Таблица Рейс

Имя поля	Тип данных
*ID	Счетчик
НомерВодит-Удостов	Числовой
IDАвтомобиля	Числовой
ДатаОтправления	Дата/время

Связи информационных объектов

Главный объект	Подчиненный объект	Тип Связи
Клиент	Перемещение	1:M
Рейс	Перемещение	1:M
Водитель	Рейс	1:M
Автомобиль	Рейс	1:M

Варианты5, 17, 29 База данных « Библиотека»

Таблица Автор

Имя поля	Тип данных
*ID	Счетчик
Фамилия	Текстовый
Имя	Текстовый
Отчество	Текстовый

Таблица Выдача

Имя поля	Тип данных
*ID	Счетчик
IDЧитателя	Числовой
IDКниги	Числовой
ДатаВозврата	Дата/время

Таблица Читатель

Имя поля	Тип данных
*ID	Счетчик
Фамилия	Текстовый
Имя	Текстовый
Отчество	Текстовый
Телефон	Числовой
Адрес	Текстовый

Таблица Книга

Имя поля	Тип данных
*ID	Счетчик
Название	Текстовый
ГодИздания	Дата/время
IDАвтора	Числовой
IDАбонемента	Числовой

Таблица Абонемент

Имя поля	**Тип данных**
*ID	Счетчик
Название	Текстовый

Связи информационных объектов

Главный объект	**Подчиненный объект**	**Тип Связи**
Читатель	Выдача	1:M
Книга	Выдача	1:M
Автор	Книга	1:M
Абонемент	Книга	1:M

Варианты 6, 18, 30 База данных «Кинотеатр»

Таблица Фильм

Имя поля	**Тип данных**
*ID	Счетчик
Название	Текстовый
Год	Дата/время
Жанр	Текстовый

Таблица Места

Имя поля	**Тип данных**
*ID	Счетчик
IDЗала	Числовой
Ряд	Числовой
Место	Числовой

Таблица Сеансы

Имя поля	**Тип данных**
*ID	Счетчик
IDЗала	Числовой
IDФильма	Числовой
Дата	Дата/время
ВремяНачала	Дата/время

Таблица Билеты

Имя поля	**Тип данных**
*ID	Счетчик
IDСеанса	Числовой
IDМеста	Числовой
Цена	Финансовый
Продажа	Логический

Таблица Кинозал

Имя поля	**Тип данных**
*ID	Счетчик
Название	Текстовый

Связи информационных объектов

Главный объект	Подчиненный объект	Тип Связи
Зал	Место	1:M
Зал	Сеанс	1:M
Фильм	Сеанс	1:M
Сеанс	Билет	1:M
Место	Билет	1:M

Варианты База данных 7, 19 База данных «Производство»

Таблица Мастер

Имя поля	Тип данных
*ТабельныйНомер	Числовой
Фамилия	Текстовый
Имя	Текстовый
Отчество	Текстовый
Разряд	Числовой
Профессия	Текстовый
IDУчастка	Числовой

Таблица Операция

Имя поля	Тип данных
*ID	Счетчик
IDИзделия	Числовой
Название Операции	Текстовый
Профессия	Текстовый
Минимальный Разряд	Числовой
Трудоёмкость	Числовой

Таблица Подряд

Имя поля	Тип данных
*ID	Счетчик
ТабельныйНомер	Числовой
IDОперации	Числовой
Количество	Числовой

Таблица Участок

Имя поля	Тип данных
*ID	Счетчик
Название	Текстовый
IDЦеха	Числовой

Таблица Изделие

Имя поля	Тип данных
*ID	Счетчик
Шифр	Текстовый

Таблица Цех

Имя поля	Тип данных
*ID	Счетчик
Название	Текстовый

Связи информационных объектов

Главный объект	Подчиненный объект	Тип Связи
Участок	Рабочий	1:M
Изделие	Операция	1:M
Рабочий	Подряд	1:M
Операция	Подряд	1:M
Цех	Участок	1:M

Варианты 8, 20 База данных «Такси»

Таблица Автомобиль

Имя поля	Тип данных
*ID	Счетчик
Модель	Текстовый
ГосНомер	Текстовый
ЧислоПассажиров	Числовой
ДетскоеКресло	Логический

Таблица Водитель

Имя поля	Тип данных
*IDВоди-тельского Удостоверения	Числовой
Фамилия	Текстовый
Имя	Текстовый
Отчество	Текстовый
Телефон	Числовой

Таблица Заявка

Имя поля	Тип данных
*ID	Счетчик
IDВодительского Удостоверения	Числовой
IDКлиента	Числовой
IDАвто	Числовой
Дата	Дата/время
Время	Дата/время
ПунктОтправления	Текстовый
ПунктНазначения	Текстовый

Таблица Клиент

Имя поля	Тип данных
*ID	Счетчик
Телефон	Числовой
Фамилия	Текстовый
Имя	Текстовый
Отчество	Текстовый

Связи информационных объектов

Главный объект	Подчиненный объект	Тип Связи
Водитель	Заявка	1:M
Клиент	Заявка	1:M
Авто	Заявка	1:M

Варианты 9, 21 База данных «Кафе »

Таблица Клиент

Имя поля	Тип данных
*ID	Счетчик
Фамилия	Текстовый
Имя	Текстовый
Отчество	Текстовый
НомерКарты	Числовой

Таблица Блюдо

Имя поля	Тип данных
*ID	Счетчик
Название	Текстовый
Стоимость	Денежный
ЕдиницаИзмерения	Текстовый
Количемтво	Числовой

Таблица Заказ

Имя поля	Тип данных
*ID	Счетчик
IDЧека	Числовой
IDБлюда	Числовой
Количество	Числовой

Таблица Официант

Имя поля	Тип данных
*ТабельныйНомер	Числовой
Фамилия	Текстовый
Имя	Текстовый
Отчество	Текстовый

Таблица Чек

Имя поля	Тип данных
*ID	Счетчик
ТабельныйНомер	Числовой
IDСтола	Числовой
IDКлиента	Числовой
Дата	Дата/время
Время	Дата/время

Таблица Стол

Имя поля	Тип данных
*ID	Счетчик
Название	Текстовый

Связи информационных объектов

Главный объект	Подчиненный объект	Тип Связи
Чек	Заказ	1:M
Блюдо	Заказ	1:M
Официант	Чек	1:M
Стол	Чек	1:M
Клиент	Чек	1:M

Варианты 10, 22. База данных «Театр»

Таблица Актеры

Имя поля	Тип данных
*ID	Счетчик
Фамилия	Текстовый
Имя	Текстовый
Отчество	Текстовый
Звание	Текстовый

Таблица Сеансы

Имя поля	Тип данных
*ID	Счетчик
IDСпектакля	Числовой
НомерЗала	Числовой
Дата	Дата/время
Время	Дата/время

Таблица Места

Имя поля	Тип данных
*ID	Счетчик
НомерЗала	Числовой
Ряд	Числовой
Место	Числовой

Таблица Спектакли

Имя поля	Тип данных
*ID	Счетчик
Название	Числовой
Режиссёр	Текстовый
Год	Дата/время

Таблица Роли

Имя поля	Тип данных
*ID	Счетчик
IDСпектакля	Числовой
IDАктера	Числовой
Роль	Текстовый

Таблица Билеты

Имя поля	Тип данных
*ID	Счетчик
IDСеанса	Числовой
IDМеста	Числовой
Стоимость	Денежный

Связи информационных объектов

Главный объект	Подчиненный объект	Тип Связи
Спектакль	Сеанс	1:M
Спектакль	Роль	1:M
Актер	Роль	1:M
Сеанс	Билеты	1:M
Место	Билеты	1:M

Варианты 11, 23. База данных «Автостанция»

Таблица Водитель

Имя поля	Тип данных
*IDВодитель-скогоУдосто-верения	Числовой
Фамилия	Текстовый
Имя	Текстовый
Отчество	Текстовый
Телефон	Числовой

Таблица Пассажиры

Имя поля	Тип данных
*ID	Счетчик
Фамилия	Текстовый
Имя	Текстовый
Отчество	Текстовый
НомерПаспорта	Текстовый

Таблица Автобус

Имя поля	Тип данных
*ID	Счетчик
Модель	Текстовый
ГосНомер	Текстовый
ЧислоПасажиров	Числовой

Таблица Билеты

Имя поля	Тип данных
*ID	Счетчик
IDПассажира	Числовой
IDРейса	Числовой
Место	Числовой

Таблица Рейс

Имя поля	Тип данных
*ID	Счетчик
IDАвтобуса	Числовой
IDВодительского-Удостоверения	Числовой
Дата	Дата/время
ВремяОтпр	Дата/время
ПунктНазначения	Текстовый

Связи информационных объектов

Главный объект	Подчиненный объект	Тип Связи
Пассажиры	Билеты	1:M
Рейс	Билеты	1:M
Автобус	Рейс	1:M
Водитель	Рейс	1:M

Варианты 12, 24. База данных «Сервисный центр»

Таблица Мастер

Имя поля	**Тип данных**
*ID	Счетчик
Фамилия	Текстовый
Имя	Текстовый
Отчество	Текстовый
ДатаРождения	Дата/время

Таблица Клиент

Имя поля	**Тип данных**
*ID	Счетчик
Фамилия	Текстовый
Имя	Текстовый
Отчество	Текстовый
Телефон	Числовой

Таблица Работы

Имя поля	**Тип данных**
*ID	Счетчик
IDОборудования	Числовой
Срок	Дата/время
Стоимость	Числовой
НаименованиеРаботы	Текстовый

Таблица Заявка

Имя поля	**Тип данных**
*ID	Счетчик
IDКлиента	Числовой
IDРаботы	Числовой
IDМастера	Числовой
ДатаПриёма	Дата/время

Таблица Оборудование

Имя поля	**Тип данных**
*ID	Счетчик
Название	Текстовый
Категория	Текстовый

Связи информационных объектов

Главный объект	**Подчиненный объект**	**Тип связи**
Оборудование	РРаботы	1:M
Клиент	Заявка	1:M
Работа	Заявка	1:M
Мастер	Заявка	1:M

Лабораторная работа № 4

ИЗУЧЕНИЕ ТИПОВ, ПОРЯДКА ПОСТРОЕНИЯ И РЕАЛИЗАЦИИ ЗАПРОСОВ РЕЗЯЦИОННОЙ БД MS ACCESS

Цель лабораторной работы: закрепление теоретических знаний, полученных в лекционном курсе.

Задание для самостоятельной работы: изучить порядок создания и работы с запросами в БД MS Access; приобрести навыки разработки запросов в реляционных БД.

Краткие теоретические сведения

Одним из стандартных объектов MS Access является запрос. Запросы используются для просмотра, анализа и изменения данных в одной или нескольких таблицах. Например, можно использовать запрос для отображения данных из одной или нескольких таблиц и отсортировать их в определенном порядке, выполнить вычисления над группой записей, осуществить выборку из таблицы по определенным условиям. Запросы могут служить источником данных для форм и отчетов MS Access.

Сам запрос не содержит данных, но позволяет выбирать данные из таблиц и других запросов и выполнять над ними ряд операций.

В MS Access существует несколько видов запросов. Одним из наиболее часто применяемых типов запросов является запрос на выборку. Он содержит условия отбора данных и возвращает выборку, соответствующую указанным условиям, без изменения возвращаемых данных. Источником для запроса могут служить таблицы или другие запросы.

Для изменения уже существующих запросов и создания новых используется конструктор запросов. Чтобы создать новый запрос с помощью конструктора запросов, необходимо: в окне базы данных на панели объектов выбрать ярлык «Запросы»; в списке запросов выбрать ярлык «Создание запроса» в режиме конструктора или нажать кнопку «Создать», в появившемся окне «Новый запрос» выбрать пункт «Конструктор»; в окне «Добавление таблицы» выбрать одну или несколько таблиц или запросов для построения нового запроса и нажать кнопку «Добавить». Для удобства выбора таблиц и запросов в окне существуют следующие вкладки:

«Таблицы», на которой отображается список таблиц;

«Запросы», на которой отображается список запросов;

после добавления всех необходимых таблиц нажать кнопку «Закрыть» в окне «Добавление таблицы».

Создание перекрестного запроса: на вкладке Запросы нажать кнопку Создать, выбрать из списка инструментов «Перекрестный запрос». Выбрать источник данных – таблицу или запрос; выбрать необходимое поле в качестве заголовков строк, нажать кнопку Далее; выбрать необходимое поле в качестве заголовков столбцов, нажать кнопку Далее; для значения ячейки выбрать из списка Поля значение соответсвующее названию данных, из списка Функция выбрать необходимую групповую операцию – суммирование, минимальное, максимальное, первое и т.д., нажать кнопку Далее; указать имя.

Содержание работы:

1. Создать простой запрос с использованием данных из нескольких таблиц используемых в БД лабораторной работы №3.
2. Создать перекрестный запрос, позволяющий представить данные из таблиц и запросов БД в виде электронной таблицы (с группировкой по строкам и столбцам и вычисляемым значением на их пересечении).

Варианты	Простой запрос	Перекрёстный запрос
1, 13, 25	Все результаты спортсмена в хронологическом порядке	Рекорды соревнований по дисциплинам
2, 14, 26	Результаты анализов для врача, отсортированные по пациентам и в хронологическом порядке	Расписание врача на неделю. Столбцы – дата, строки – время, на пересечении Пациенты
3, 15, 27	Вывести предметы, преподаваемые студенту в семестре	Столбцы – предметы, строки – группы, на пересечении - преподаватель
4, 16, 28	Вывести всех водителей автомобиля за предыдущий год	Вывести суммарный товарооборот между пунктами отправления и прибытия
5, 17, 29	Вывести книги определённого автора прочитанные определённым Читателем	Вывести распределение количество книг авторов по абонементам

Варианты	Простой запрос	Перекрёстный запрос
6, 18, 30	Вывести фильмы, показываемые в этом месяце. Использовать «Построитель выражений» - «Встроенные функции» - функции Month, Year, Now	Вывести распределение стоимости билетов по местам в зале на конкретный сеанс
7, 19	Вывести список изделий, обрабатываемых определённым мастером	Вывести распределение количества подрядов между цехами в зависимости от названия операции
8, 20	Вывести список телефонных номеров клиентов, перевезённых определённым водителем за конкретный день	Вывести график количества заявок автомобилей по часам. Использовать формулы, Использовать «Построитель выражений» - «Встроенные функции» - функция Hour
9, 21	Вывести «Заказ» из определённого чека (названия блюд, количество, стоимость)	Распределение количества заказов по столам и официантам
10, 22	Вывести спектакли, показываемые в этом месяце. Использовать «Построитель выражений» - «Встроенные функции» - функции Month, Year, Now	Вывести распределение стоимости билетов в зависимости от ряда и места для определённого сеанса
11, 23	Вывести расписание рейсов (Время отправления и Пункт назначения) за определённую дату	Вывести количество проданных билетов в зависимости от пункта назначения и часа отправления. Использовать «Построитель выражений» - «Встроенные функции» - функции Hour
12, 24	Вывести «Прейскурант» работ для определённой категории оборудования	Вывести количество заявок исполненных мастерами в зависимости от категории оборудования

Контрольные вопросы

1. Какое определение объекта Запрос Вам известно?
2. В чем заключается назначение запроса?
3. Какие управляющие запросы или запросы действия Вам известны?
4. Когда используются перекрестные запросы?
5. Каков порядок создания запросов?

Лабораторная работа № 5

ИНТЕГРИРОВАННАЯ СРЕДА PascalABC.NET

Цель работы: приобрести навыки работы в системе программирования на примере интегрированной среды PascalABC.

Основные понятия

Этапы подготовки и решения задач на компьютере

Практика программирования показывает, что решение прикладных, инженерных, экономических и научных задач на ЭВМ сложный и трудоемкий процесс, состоящий из следующих этапов (Рис. 5.1).

1. ***Постановка задачи*** состоит в четком изложении условия задачи и определении подзадач.

2. *Физический* и *математический анализ.* Анализируется, существует ли вообще решение данной задачи и единственно ли оно. Подбирается математический аппарат, и строится математическая модель для решения задачи. Выбирается метод или методика решения (составляются формулы, определяются правила, связывающие эти формулы).

3. **Этап *алгоритмизации***. На основании выбранного метода и конкретных методик с учетом возможностей ПК разрабатывается алгоритм и строится его схема. Этот этап заключается в разложении вычислительного процесса на возможные составные части, описании содержания каждой такой части, установлении порядка их следования, которые определят структуру программы, т. е. разрабатывается укрупнённый алгоритм решения задачи и проверяется возможность реализации выбранного метода. Подобное расчленение алгоритма на составные части называется **структуризацией**.

4. ***Этап программирования***. Выбирается язык и (или) система программирования, и в соответствии с алгоритмом разрабатывается программа на конкретном языке программирования.

5. ***Отладка программы*** и ***тестирование***. Отладка программы состоит в обнаружении и исправлении ошибок, допущенных на всех

этапах проектирования программы. Синтаксические ошибки обнаруживаются компилятором, который выдаёт сообщение об ошибке и её месте (в основном это ошибки в написании операторов) при компиляции. Алгоритмические ошибки или смысловые обнаруживаются в результате тестирования.

6. ***Решение*** задач на компьютере.

7. ***Обработка результатов*** решения задач. Производится анализ результатов, строятся таблицы, графики, делаются выводы.

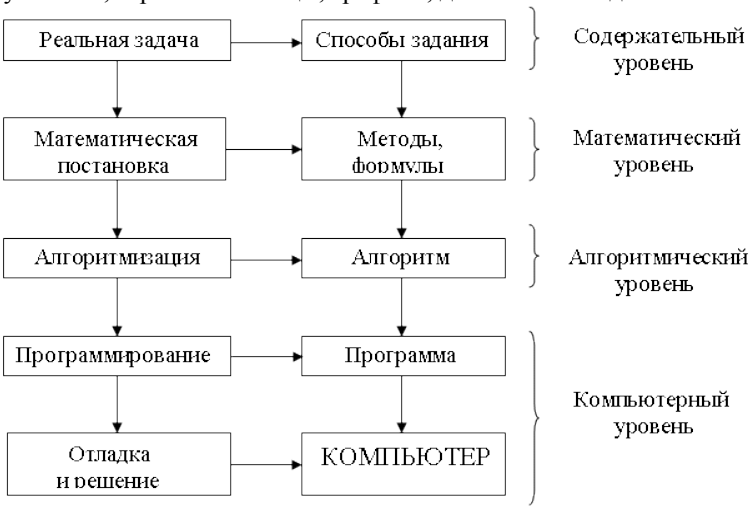


Рис.5.1. Наглядная схема решения задачи

Рассмотрим этапы программирования, отладки и решения задач в интегрированной среде Pascal ABC.NET

Интерфейс среды Pascal ABC.NET и работа с файлами

Окно приложения Pascal ABC.NET состоит из следующих частей:

- главное меню;
- панель инструментов;
- текстовый редактор;
- нижняя панель.

После входа в среду в верхней строке экрана появляется главное меню состоящее из следующих разделов (рис 5.2):

1. **Файл** – работа с файлами и выход из системы.
2. **Правка** – операции редактора текстов и поиска.

3. **Вид** – работа с окнами в системе.
4. **Программа** – выполнение и отладка программы.
5. **Сервис** – вызов вспомогательных функций.
6. **Модули** – сообщение об ошибке.
7. **Помощь** – обращение к справочнику

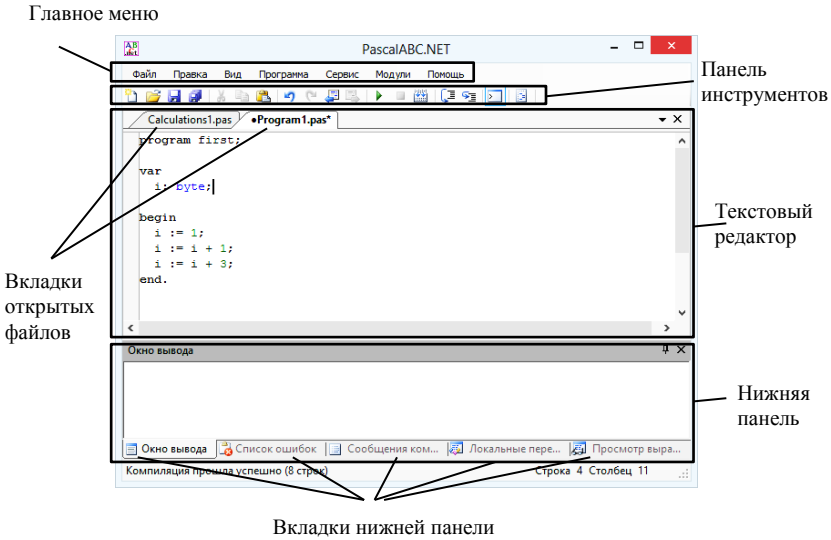


Рис.5.2. Интерфейс среды Pascal ABC.Net

Выбор раздела главного меню может быть осуществлен либо указателем мыши, либо нажатием клавиши *F10* и перемещением по разделам с помощью горизонтальных стрелок на клавиатуре ПК. Выбор раздела так же может быть ускорен одновременным нажатием клавиш *Alt* и первой буквы выбранного пункта. Например, одновременное нажатие *Alt-Ф* приводит к тому же результату, что и последовательное нажатие *F10* и *Файл*. В случае ошибочного выбора опции для возвращения на высший уровень меню необходимо нажать клавишу *Esc* или кнопку *Отмена.*

Для создания новой программы можно в меню **Файл** выбрать подменю **Новый** и набрать текст программы. Это же можно сделать с помощью одновременно нажатых клавиш *Ctrl-N.*

Введенный текст можно сохранить на диск достаточно нажать клавиши *Ctrl-S*, обратиться к разделу меню **Файл** и опции **Сохранить** или нажать значок дискеты на панели инструментов.

Одним из способом загрузки файлов является выбор раздела **Файл** (рис.1.3) и затем опции **Открыть** (или оперативных клавиш *Ctrl-O)* . После этих действий на экране появляется диалоговое окно, в котором выбрать открываемый файл и нажать клавишу *Enter*. Также файл можно открыть с помощью панели инструментов, нажав на соответствующий значок

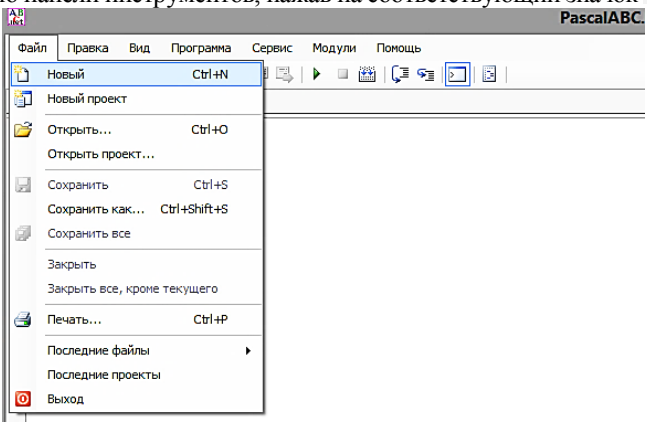


Рис.5.3. Меню Файл

Текстовый редактор

Исходный модуль – это текст программы, записанный на языке высокого уровня, в данном случае это одна из версий языка Pascal.

Для создания и редактирования текстов программ предназначен *текстовый редактор* системы Pascal ABC. Режим редактирования автоматически устанавливается сразу после загрузки PascalABC: в окне редактора появляется курсор.

Все команды редактора условно можно разделить на команды перемещения курсора, команды удаления/вставки, команды работы с выделенными блоками.

Среди команд перемещения курсора наиболее часто используются следующие:

на страницу вверх –*Pg Up*

в начало текущей строки – *Home*

на страницу вниз –*Pg Dn*

в конец текущей строки – *End*

в начало программы– *Ctrl-HOME*

в конец программы –*Ctrl-END*

Команды удаления и вставки:

включить (отключить) режим вставки – *INS*

стереть символ слева от курсора – *Backspace*

стереть символ справа от курсора или выделенный фрагмент – *Del*

удалить строку, на которой располагается курсор – *Ctrl-Y*

удалить слово справа от курсора –*Ctrl-*

вставить новую строку – *Enter*

удалить выбранный текст из окна и не помещать его в буфер обмена – *Ctrl-Del*

копирует выбранный текст в буфер обмена – *Ctrl- Ins* или *Ctrl-C*

помещает выбранный текст в буфер обмена и удаляет его из окна – *Shift-Del* или *Ctrl-X*

помещает текст из буфера в активное окно – *Shift- Ins* или *Ctrl-V*

Для выделения текста блока применяются следующие команды:

выделить фрагмент текста – *Shift - <стрелки>*

выделить всё –*Ctrl-A*

Редактор позволяет также обрабатывать выделенные фрагменты текста. Выделение текста выполняется обычными средствами: “мышью” или клавишами *Shift* со стрелками. Команды редактора находятся в меню **Правка** (рис.5.4).

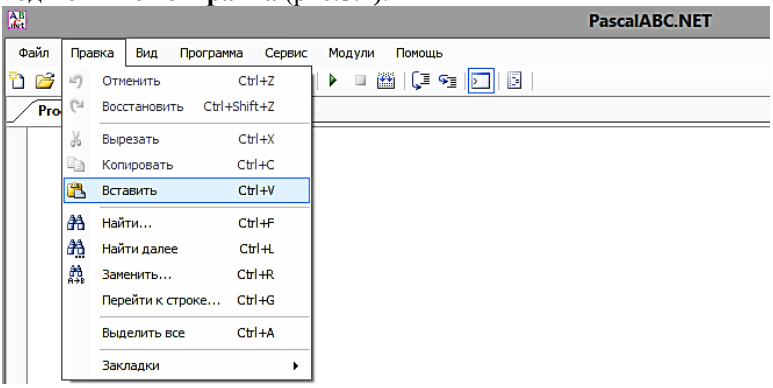


Рис.5.4. Меню Правка

Правила оформления программ

Набор текста программы производится с клавиатуры. После заполнения очередной строки следует нажать на клавишу *<Enter>*, чтобы перевести курсор на следующую строку.

Каждое предложение программы рекомендуется начинать с новой строки. Предложение в языке Паскаль может начинаться с любой позиции строки, однако при наборе программы следует придерживаться следующих правил:

Признаком хорошего стиля программирования является применение отступов в строках при оформлении текстов программ, поскольку они делают программу более наглядной (см. Program Primer3 в лабораторной работе №3).

Так, объявления (const, type, var и т.п.) и слова begin - end, определяющие тело главной программы, выравниваются по левому краю текста. Все предложения внутри любой пары операторных скобок begin - end записываются с отступом на определенное число позиций по отношению к первой букве слова begin (end).

В среде PascalABC.NET возможно автоматическое форматирование текста программы, после которого код принимает читаемый вид. Для автоматического форматирования необходимо выбрать раздел меню **Сервис**, и активировать опцию **Форматировать код** или нажать комбинацию клавиш *Ctrl-Shift-F*.

Кроме того, сделать программу более наглядной позволяет включение в ее текст комментариев, которые заключаются в фигурные скобки или комбинацию символов из открывающейся и закрывающейся скобок и символов звёздочек. Пример: «(* комментарий *)». Комментарии рекомендуется помещать в начало программы и в начало крупной программной единицы (подпрограммы, вложенных циклов и т.п.) и выделять пустыми строками или особо важные заключать в рамку.

Во время компиляции текст комментариев игнорируется, поэтому заключение строки программы в фигурные скобки исключает данную строку из конечной программы. Но, при необходимости, позволяет быстро её вернуть в работу, убрав фигурные скобки.

Компиляция и исправление синтаксических ошибок

После подготовки текста программы необходимо ее откомпилировать, связать с библиотекой стандартных процедур и функций, загрузить в оперативную память и передать ей управление.

Компиляция – это трансляция программы, записанной на исходном языке, в объектный модуль.

В среде PascalABC.NET программное средство, выполняющее компиляцию программы, называется ***Компилятор***

Этап компиляции также позволяет определить наличие или отсутствие синтаксических ошибок в программе. При их обнаружении среда автоматически загружает в окно редактора исходный файл и

помещает курсор в ту строку программы, при компиляции которой найдена ошибка (рис. 5.5).

Во многих случаях такие ошибки возникают из-за неправильного набора текста на клавиатуре, пропуска запятых, двоеточий и т.п. Найденная ошибка исправляется программистом, и производится повторная компиляция программы.

После удачной компиляции, в работу вступает Редактор связей.

Редактор связей – программа, предназначенная для создания исполняемого exe-файла программы из одного или более независимо транслируемых объектных или загрузочных модулей.

По умолчанию, в среде PascalABC.NET, исполняемый exe-файл удаляется после его выполнения. Для того, чтобы сохранить исполняемый файл после «прогона» программы, необходимо вызвать окно **Настройки**, с помощью меню **Сервис**, а затем в разделе опции компиляции снять галочку с пункта «Удалять EXE-файл после выполнения». Данный файл впоследствии можно использовать для решения прикладных задач автономно, без установки на компьютер среды PascalABC.NET

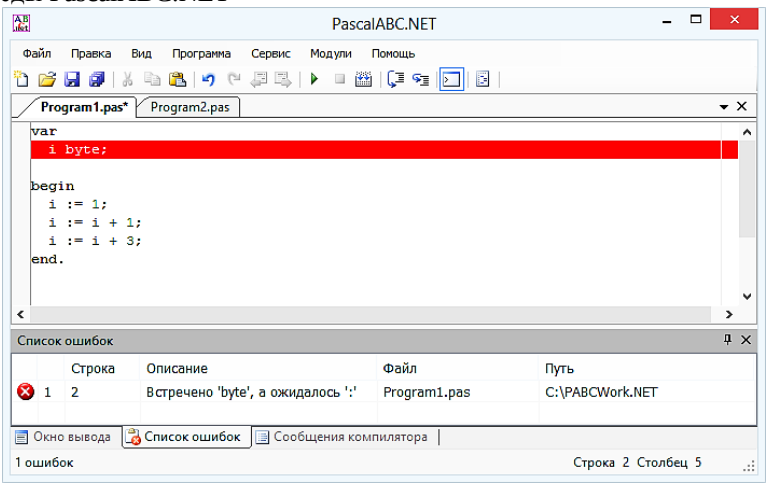


Рис.5.5. Ошибка выявленная в результате компиляции

Компиляция, связь с библиотекой стандартных процедур и функций, загрузка в оперативную память и передача ей управление называется *исполнением программы* и реализуется в разделе **Программа** командой **Выполнить** (*F9*). Вызов команды **Выполнить** приводит к компиляции и исполнению программного

модуля, находящегося непосредственно в редакционном окне (рис. 5.6).

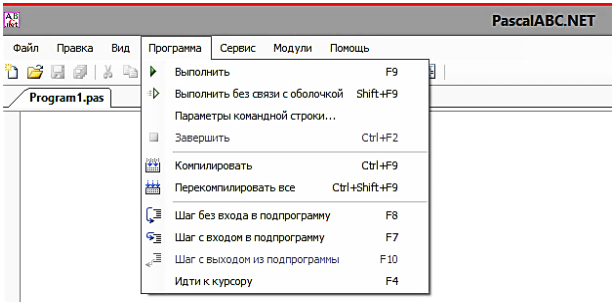


Рис.5. 6. Меню Программа

Отладка программы

Отладка – это процесс поиска и исправления ошибок в тексте программы, препятствующих её корректной работе.

Основной формой контроля правильности выполнения программы является получение и анализ промежуточных тестовых результатов. Встроенный *Отладчик* Паскаля позволяет легко выполнять программы по шагам, проверяя и модифицируя при этом переменные и ячейки памяти, устанавливая точки останов и прерывая выполнение программы с помощью специальных клавиш (Ctrl + Break). Нажатие на клавишу F1 выводит подробные сообщения с описанием опций и пунктов меню, поэтому приведём только общие сведения.

Без достаточного опыта отладки можно пользоваться одной клавишей F8 (шаг без входа в подпрограмму), после нажатия которой среда осуществит компиляцию, компоновку (связь с библиотекой стандартных процедур и функций) и загрузку программы, а затем остановит выполнение («прогон» программы) перед исполнением первого оператора, т.е. указатель остановится на слове Begin, открывающем раздел операторов основной программы. Строка будет выделена цветом (рис. 5.6). Теперь каждое следующее нажатие клавиши F8 будет вызывать исполнение операторов, запрограммированных в текущей строке, и смещение указателя к следующей строке текста программы.

В дальнейшем возможно использование клавиши F7 вместо клавиши F8. Отличие заключается в способе выполнения функций и подпрограмм. Клавиша F8 выполняет подпрограмму целиком, как один оператор, а клавиша F7 перемещает курсор внутрь текста подпрограммы и выполняет все её внутренние операторы пошагово.

Если возникла необходимость «прогнать» некоторую часть программы и остановить её выполнение в определённой строке, например, если известно что в «прогоняемой» части нет ошибок, тогда используют клавишу F4. В этом случае помещают курсор в строку, в которой необходимо остановить выполнение программы и нажимают клавишу F4. Все операторы находящиеся до курсора выполняются безостановочно, а на строке, в которой расположен курсор выполнение программы останавливается и возможно проверить правильность её выполнения.

В подозрительном (с точки зрения возможных ошибок) месте текста программы можно посмотреть значения любых переменных или выражений. Для этого, в меню **Вид** необходимо выбрать опцию **Локальные переменные**, после чего в нижней панели отобразится соответствующая вкладка (рис. 5.7).

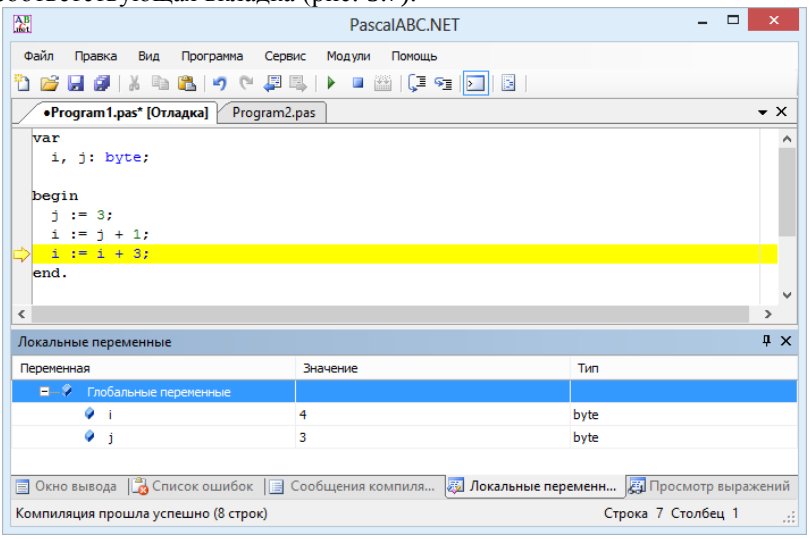


Рис.5. 7. Пошаговое выполнение программы с отслеживанием значений переменных

Для просмотра при «прогоне» программы сложных выражений, в которые входят сразу несколько значений переменных необходимо активизировать окно **Просмотр выражений** в разделе меню **Вид**. Добавление выражения для наблюдения осуществляется с помощью двойного клика курсором в первой свободной строке вкладки **Просмотр выражений** нижней панели, введения выражения и нажатия

клавиши *Enter*. Для удаления выражения необходимо активировать строку с его значением и нажать клавишу *Delete*.

Тестирование программы

Если в программе нет синтаксических ошибок, то далее производится тестирование программы. *Тестирование* включает подготовку тестов (специальным образом подобранных наборов входных данных, для которых заранее просчитан результат), его прогон и сравнение полученных результатов с эталонными (тестовыми). Число тестовых наборов должно быть достаточным для слежения за ходом выполнения программы по всем ее ветвям и проверки свойств, характеристик и функций (спецификаций) проектируемой программы. Кроме того, необходимы наборы, контролирующие допустимые значения для исходных данных и предельные значения для результатов.

Задание к лабораторной работе

1. Запустите загрузочный файл PascalABC.
2. Ознакомьтесь с видом основного экрана PascalABC. В отчете зарисуйте схему экрана.
3. Отработайте вызов меню различными способами

а) *F10, Enter;*

б) *Alt* – первая буква пункта меню.

4. Наберите текст программы, указав в треугольных скобках собственную группу.

```
Program lab1;
 begin
   writeln('<>');
 end.
```

5. Скопируйте строку *writeln('<>');*, используя оперативные клавиши. Вставьте в программу и уже в треугольных скобках укажите собственную фамилию, имя, отчество.
6. Отправьте файл на выполнение *F10*, **Программа**, **Выполнить** (или *F9*). Просмотрите результат работы программы в Окне вывода, которое расположено ниже Окна ввода.
7. Допишите в предпоследней строке программы оператор *readln;* (перед строкой «end.»)

8. Сохраните файл под именем Lab_1.pas, используя оперативные клавиши. Либо через *F10*, *Enter*, **Сохранить**, либо *Ctrl-S*.

9. Сохраните файл повторно под именем Lab_2.pas, используя оперативные клавиши. Либо через *F10*, *Enter*, **Сохранить как**, либо *Ctrl-Shift-S*.

10. Скомпилируйте исходный текст программы с помощью клавиш *Ctrl+F9*.

11. Выйдите из среды PascalABC, найдите на диске файлы Lab_1.pas, Lab_2.pas и Lab_2.exe. Занесите в отчёт полный путь к данным файлам и их размеры.

12. Запустите файл программы Lab_2.exe. Занесите в отчёт результат работы программы.

13. Загрузите оболочку PascalABC.

14. Откройте в первом окне файл Lab_1.pas, а во втором окне Lab_2.pas, используя либо *F10*, *Enter*, **Открыть**, либо *Ctrl-O*.

15. Перейдите в первое окно, используя **Панель инструментов** под строкой главного меню.

16. Наберите строку *writeln('Выполнил(а)');*. Скопируйте строку в буфер, используя функции **Панели инструментов.**

17. Перейдите во второе окно, вставьте из буфера строку, используя **Панель инструментов**.

18. Закройте поочередно окна, используя функции раздела **Файл.**

19. Создайте новое окно, используя оперативные клавиши (*Ctrl-N*).

20. Наберите текст программы:

```
program lab1;
 var i, k: byte;
  begin
   k := 3;
   i := k+5;
   i := i+1;
  end.
```

21. Выполните программу пошагово. Отследите изменения значений переменной *i* во вкладке **Локальные переменные**. Занесите в отчёт значение всех переменных на каждом из трёх шагов выполнения программы.

Содержание отчета:

1. Схема основного экрана PascalABC;
2. Содержание окна вывода, после выполнения пункта 6 задания к работе;
3. Полный путь и размеры файлов Lab_1.pas, Lab_2.pas и Lab_2.exe;
4. Значение переменных i и k на каждом из трёх шагов выполнения программы (пункты 20 и 21 задания к работе)

Контрольные вопросы

1. Сформулируйте определение PascalABC.
2. Какую функцию выполняет отладчик?
3. Кратко перечислите основные функции разделов главного меню.
4. Перечислите команды пункта меню Файл и опишите функции, которые они выполняют.
5. В каком пункте главного меню можно найти команды текстового редактора?
6. Существует ли буфер обмена в среде PascalABC?
7. Какие операции можно выполнять с буфером обмена?
8. Перечислите этапы подготовки и решения задачи.
9. Дайте определение редактору связей.
10. Дайте определение отладки.
11. Укажите функции клавиш *Ctrl-N, Ctrl-O, Ctrl-S?*
12. Укажите функции клавиш *Ctrl-Del, Ctrl- Ins, Shift-Del, Shift- Ins?*
13. Укажите функции клавиш *F7, F8, F9*?
14. С помощью какой команды можно выделить весь текст в окне?
15. Нарисуйте схему решения задачи с названиями уровней.

Лабораторная работа № 6

ПРОГРАММИРОВАНИЕ АЛГОРИТМОВ ЛИНЕЙНОЙ СТРУКТУРЫ. ВВОД И ВЫВОД ИНФОРМАЦИИ

Цель работы: освоение записи арифметических выражений в Паскале; ознакомление с вводом и выводом информации и со встроенным отладчиком Турбо Паскаля.

Основные понятия

Программа – это запись последовательности действий, которые необходимо выполнить компьютеру. Оператор в программе – это

единое неделимое предложение, выполняющее какое-либо действие. Простейшим и основным оператором является оператор присваивания, который имеет вид

a:=s;

где a – переменная; s – выражение; := - знак присваивания.

В результате выполнения этого оператора вычисляется выражение s и результат присваивается переменной a. Например, при выполнении оператора x := x + h; значение x увеличивается на величину h. Существенно, что s и a должны быть одного и того же типа (исключение сделано только для присваивания вещественному типу значений целого типа). Два последовательных оператора должны разделять точкой с запятой «;».

Арифметическое выражение имеет тот же смысл, что и в математике. Арифметические операции обозначаются символами: «+» (сложение), «-» (вычитание), «*» (умножение), «/» (деление). Для задания порядка выполнения операций можно использовать только круглые скобки «(«, «)». При этом вначале вычисляются функции (аргумент обязательно записывается в скобках), затем выражения в скобках. Арифметические операции имеют такое старшинство: «*» и «/», потом «+» и «-».

Следует отметить, что Паскаль является типизированным языком. Тип данных задает то множество значений, которые могут принимать данные рассматриваемого типа. Задание определенного типа данных устанавливает также ограниченный набор операций, которые могут выполняться над этими данными. Например, при выполнении операции деления «/» результат будет всегда вещественным, независимо от типов делителя и делимого. Для получения целочисленных значений при делении целых чисел предусмотрены операции div и mod, например: 13 div 4 = 3 (целая часть результата), 13 mod 4 = 1 (остаток от деления). В свою очередь, эти операции не могут применяться к вещественным числам типа Real. Типизация данных означает их специализацию, а это, с одной стороны облегчает выполнение операций над ними, а с другой стороны – значительно уменьшает возможно появления ошибок. Если ошибки все же возникают, то типизация данных облегчает их поиск и устранение.

В арифметических выражениях можно использовать только следующие математические функции, являющиеся «встроенными» в Паскаль (табл. 6.1). Если функции (необходимой для вычислений по разработанному алгоритму) в этом списке нет, то следует тождественными преобразованиями свести её к «встроенной.

Примеры записи выражений:

1. Выражение $e^{-|x-4|}/\cos x$ записывается в виде:

Exp(-Abs(x-4))/Cos(x)

2. Выражение $z = b + \sqrt[3]{\frac{y}{b}} \cdot c$ записывается в виде:
z=b+power((y/b),3)*c.

Таблица 6.1

Имя функции	Назначение	Ограничения аргументов	Тип	
			Аргумента	Результата
Abs(x)	$\|x\|$	нет	Вещественный	Вещественный
			Целый	Целый
Pi	число π	-	-	Вещественный
Sin(x)	sin X	x - в радианах	Вещественный	Вещественный
Cos(x)	cos X	x - в радианах	»	»
Tan(x0	tg(x)	/x = ±π/2		
Arctan(x)	arctg X (изменяется от $-\pi/2$ до $\pi/2$)	нет	»	»
ArcSin(x)	$\arcsin x$	-1 ≤ x ≤ 1	»	»
ArcCos(x)	$\arccos x$	-1 ≤ x ≤ 1	»	»
Sqrt(x)	$\sqrt{X}$	X≥0	Вещественный	Вещественный
Sqr(x)	x^2	нет	Вещественный	Вещественный
			Целый	Целый
Exp(x)	e^x	нет	Вещественный	Вещественный
Power(x,y)	x^y	нет	»	»
Ln(x)	$\ln x$	X>0	»	»
Log10(x)	$\lg x$	X>0	»	»
LogN(a,x)	$\log_a x$	нет	»	»
Int(x)	Целая часть от X	нет	»	»
Frac(x)	Дробная часть от X	нет	»	»
Round(x)	Округление X до ближайшего целого	нет	»	Целый
Trunc(x)	Целая часть значения X	нет	»	Целый
Random	Случайное число в интервале от 0 до 1	-	-	Вещественный
Random(x)	Случайное число в интервале от 0 до X	нет	Целый	Целый
Max(a,b)	Возвращает максимальное из чисел a, b	нет	Вещественный	Вещественный
			Целый	Целый
Min(a,b)	Возвращает минимальное из	нет	Вещественный	Вещественный
			Целый	Целый

Имя функции	Назначение	Ограничения аргументов	Тип	
			Аргумента	Результата
	чисел a, b			
Odd(x)	Проверка четности X, возвращает True, если X – нечетное число		»	Логический
Succ(x)	Возвращает следующее за X значение	нет	»	Целый
			Символьный	Символьный
Pred(x)	Возвращает предшествующее X значение	нет	Целый	Целый
			Символьный	Символьный

Ввод и вывод информации

Вывод информации на экран осуществляется с помощью оператора `write(…)` или `writeln(…)`. В круглых скобках при этом записывается список выводимых данных (через запятую). Можно указать количество позиций, отведенных на запись выводимых данных, только для вывода действительных чисел (количество знаков после запятой), а также выводить значения арифметических выражений, например `write(Pi*Sqr(n))`. Если элементом списка вывода является какой-либо текст, то он записывается внутри апострофов «’».

Пример. Для того, чтобы вывести на экран строку в виде
x1=3.1415926 x2=5.3 x3=14

Хранящихся в памяти компьютера чисел x1 = π, x2 = 5.31, x3 = 14, следует в тексте программы (там, где мы и хотим вывести эти значения) записать оператор:

```
Write(‘ x1 =’ ,x1:9:7, ’ x2=’ ,x2:3:1,
’ x3=’,x3:2);
```

где

‘ x1 =’ выводимый на экран текст «x1=»,

x1:9:7 – выводимое на экран значение переменной x1, при этом на вывод данного значения отводится всего 9 мест (символов), 7 из которых это цифры после запятой, сам символ разделителя целой и дробной части и один символ – цифра 3 из целой части числа π,

’ x2=’ – выводимый на экран текст « x2=», при этом перед символом «x» находится символ пробела. При его отсутствии надпись «x2=» «прилипнет» к предыдущим цифрам числа x1,

X2:3:1 – выводимое на экран значение переменной x2, при этом на вывод данного значения отводится всего 3 символа. Один на цифру из

целой части числа x2, один на символ разделителя целой и дробной части и один символ на вывод дробной части числа, хотя в памяти компьютера число x2 хранится с точностью до сотых,

’ x3=’ – аналогично, выводимый на экран текст « x3=», с пробелом перед символом «x»,

x3:2 – выводимое на экран значение переменной x3, на вывод которой отводится 2 символа, при этом количество знаков после запятой не указывается.

Оператор writeln отличается от оператора write только тем, что после его выполнения происходит перевод курсора в начало следующей строки экрана.

Если в списке вывода не указано количество позиций для чисел, то вывод будет осуществляться в соответствии с описанием переменных. Например, действительные числа выводятся в следующем виде:

S*.**********E#****,

где s – пробел для положительного или знак «-» для отрицательного числа; * - десятичная цифра; E – символ десятичного основания; # - знак «+» или «-» в зависимости от десятичного порядка, т.е. стоят ли нули в числе перед разделителем целой и дробной части или после него.

Ввод с клавиатуры осуществляется с помощью оператора Read(…) или Readln(…). В круглых скобках записывается список ввода. В списке ввода переменные записываются также через запятую, как в операторе write. Когда программа дойдёт до исполнения этого оператора, то её выполнение приостановится и необходимо набрать на клавиатуре вводимые числа. Если чисел несколько, то их можно вводить сразу, отделяя друг от друга пробелом. Сигналом окончания подготовки ввода является нажатие клавиши Enter. До этого момента можно исправлять любой ошибочно введенный символ. Оператор Readln может быть без параметров, при этом выполнение программы приостанавливается до нажатия клавиши Enter.

Пример: для ввода значений чисел x_1=2,57, x_2=-10^9, x_3=25 можно в программу поместить оператор

Readln(x1,x2,x3);

а на клавиатуре набирать так:

2.57_-1E9_25 Enter (числа здесь отделены друг от друга пробелами «_»).

Для того, чтобы знать, ввод какой информации ожидается компьютером, хорошим стилем программирования считается добавление оператора writeln (write) с сообщением-«подсказкой» перед каждым оператором ввода (как это сделано во всех приводимых

программах). В данном примере это может быть оператор Writeln(‘ Введите x1, x2, x3 ‘).

Задание для выполнения работы

1. Составить алгоритм программы, в которой предусмотреть ввод исходных данных с клавиатуры, вычисление значения функции по заданной формуле (табл. 6.2), поэтапный вывод результатов вычислений на экран монитора.

Пример: Для выражения $y = \left|\frac{\sin(x+a)}{\arcsin x}\right|$ вывод результатов вычислений должен выглядеть следующим образом:

```
Исходные данные: x=1, a=3.
1 + 3 = 4.
sin(4) = -0,756.
arcsin(1) = 1,570.
-0,756/1.570 = -0,481.
Модуль числа -0,481 равен 0,481.
Итого: y = 0,481
```

2. Ввести текст программы в компьютер.
3. Отладить программу.
4. Выполнить программу (получить результаты)

а) при вводе исходных данных из таблицы 2,

б) при вводе исходных данных отличающихся от табличных (показав тем самым универсальность программы)

Содержание отчета

5. Постановка задачи (конкретный вариант)
6. Текст программы, тестовые примеры и результаты вывода на экран.
7. Блок-схема алгоритма программы.

Варианты заданий для выполнения работы

Таблица 6.2

Вариант	Функция	Значение исходных данных		
		a	**b**	**x**
1	$y = \sqrt[3]{2a \cdot \arctan x + \ln^2 x}$	5	-	1.5
2	$y = a \cdot \lvert 1 - e^{4\cos x}\rvert + \sqrt[4]{a}$	2	-5	0.5
3	$y = \frac{\sin(x+a)}{\arcsin x}$	$\pi/2$	-	$\pi/4$
4	$y = \left\lvert\frac{x+a+3b}{2.5x}\right\rvert$	2	2	-15
5	$y = \left\lvert\arcsin(\sqrt{xab} + x^2)\right\rvert^5$	0.2	0.1	$-\pi/10$

Вариант	Функция	Значение исходных данных		
		a	b	x
6	$y = \frac{\lvert x - a^2 \rvert}{\sqrt[5]{\ln x} + b}$	1.2	2.1	5
7	$y = \left\lvert \frac{a + \sin b}{b + \sin a} \right\rvert + x$	$\pi/3$	$\pi/6$	1
8	$y = x^a + a^b + \mathrm{tg}(x + 2b)$	2	-3	3.4
9	$y = ab - 4a^2\lvert b \rvert + \sqrt[3]{\sin x}$	2	-3	$\pi/3$
10	$y = x + \frac{a \cdot \ln x}{\sqrt[3]{x} + b\sqrt{a}}$	9	-1	2
11	$y = \frac{3.25 + ab\lvert x \rvert}{\sqrt{\lvert x + 2ab + b^2 \rvert}}$	5	2	-3
12	$y = \frac{\ln^2 x + ab^4}{\cos^2(x + b)}$	10	0.2	0.3
13	$y = \left\lvert \frac{x + a}{b} \right\rvert^{\sin x}$	2	3	$\pi/4$
14	$y = \frac{\sin^2(xa + b)}{\sqrt[3]{\left\lvert \frac{x}{a} \right\rvert} - \ln x}$	-2	1	$\pi/3$
15	$y = \frac{2\sin^2 x + 3a}{\sqrt[3]{\lvert a + b \rvert}}$	2	75	0.85
16	$y = \sqrt[5]{\sin x + \cos^2(xa)}$	-2	-	$\pi/8$
17	$y = \mathrm{tg}^2 3x - \mathrm{ctg}^2 ab$	3	2	4
18	$y = \frac{a + b^x}{\sin\sqrt{a + b}}$	0.6	0.3	2
19	$y = \frac{\lvert x - a^2 \rvert}{\arcsin(\mathrm{a} - 2\mathrm{b}^2)}$	1.2	0.6	8
20	$y = \frac{a + \sqrt[3]{\lvert x - b \rvert}}{\cos^2(x + b)}$	10	0.2	0.3
21	$y = \frac{\mathrm{tg}^2(x - 2a)}{a + 3b^5}$	1	5	3
22	$y = \frac{\lvert \cos(x - 2b) \rvert}{\sqrt[3]{\ln x}}$	-	2.1	5
23	$y = \frac{a + 3b^5}{\arcsin(\mathrm{a} - 2\mathrm{b}^2)}$	0.6	0.3	2
24	$y = \frac{\mathrm{tg}^2(x + 2a)}{\sqrt{\lvert x + 2ab + b^2 \rvert}}$	5	2	-3
25	$y = \frac{a + b^x}{\sin\sqrt{a + b}}$	2	-6	$\pi/12$
26	$y = \frac{\ln^2 x + ab^4}{\sqrt[4]{\sin x}}$	3	2	0.1
27	$y = \frac{\log_2(x^2 + 3ab) + 0.5}{\sqrt[4]{\lvert \sin x + \cos x \rvert}}$	2	5	$\pi/4$

Вариант	Функция	Значение исходных данных		
		a	b	x
28	$y=\frac{\lvert x-a^2\rvert}{\sqrt[6]{\sin x}}$	3	2	0.1
29	$y=\mathrm{tg}^5 x\cdot\left(\sin x^2+2\sqrt{b}\right)+\frac{\ln x}{\lvert x\rvert}$	-	12	$\pi/3$
30	$y=\sqrt{x^3+2ab^5}+\frac{\mathrm{tg}^2 x+3}{\sin 2x}$	3	5	$\pi/4$

Практические задания для защиты

Выбрать алгоритм и составить его блок-схему для вычисления значения указанной в варианте функции y=y(x). Составить программу вычисления величины y(x)

Учесть возможность ввода пользователем значений аргумента x, при которых невозможно вычислить значение функции y (см. лабораторную работу № 3). Например: деление на ноль; подкоренное выражение корня чётной степени, меньше нуля; аргумент функции arcsin больше единицы и т.д.

Варианты практических заданий для защиты

1. $=\frac{x}{x^2-1}+log_3(x+2)$
2. $y=\frac{x^3}{(x+1)(x+2)}+\frac{\arcsin(1-x)}{\sqrt[3]{1-\ln x}}$
3. $y=\frac{\sin x}{1-\cos x}+\frac{\mathrm{tg}^3(\ln(1-x)}{\lvert 1+x\cdot e^{-x}\rvert}$
4. $y=\frac{-\arccos(1+x)}{\sqrt[4]{x^3-1}}++(2-x)\cos^2\lvert x\rvert$
5. $y=\frac{\cos^2 x}{1+\sin x}-\ln^2\left(\frac{x}{\sqrt[3]{x-1}}\right)$
6. $y=\frac{x^3e^{x-1}}{x^3-\lvert x\rvert}+log_2(\sqrt{x}-x)$
7. $y=\frac{\sqrt[3]{x+\sin x}}{x^2-x^4}\cdot\arcsin^2\sqrt[4]{3-x}$
8. $y=\frac{x^5+e^{-2\lvert x\rvert}}{\sqrt[4]{9-x^2}}\cdot\mathrm{tg}^3\lvert\cos^2 x\rvert$
9. $y=\frac{\sin x+{}^1/_x}{\sqrt[3]{tg^2x-\frac{x^3}{x^2-4}}}$
10. $y=\frac{\sin^2\frac{\lvert x\rvert}{2}+3^{\frac{1}{x-1}}}{\sqrt[6]{x^4-16}}\cdot\sqrt{1-\ln x}$
11. $y=\frac{\sqrt[4]{8x^2-6x+1}}{\mathrm{arctg}\sqrt{2x+1}}+2^{\frac{\sin x}{\lvert x\rvert}}$

12. $y=\frac{\ln^3\frac{(x-1)^2}{x}+\cos^2(2x)}{\sqrt[6]{x^2-5x+6}}\cdot\cdot\sin\frac{3^{x^2-1}}{2}$

13. $y=\sqrt[3]{\log_2(1-x)}+\frac{\mathrm{tg}\left(1+\frac{1}{x}\right)}{\sqrt{|x|}-2}$

14. $y=\frac{1}{x}\log_3(4-x^2)+\frac{\sin(\cos x)}{e^{|x|}-1}$

15. $y=\frac{\sqrt[4]{|x|}+1}{\sin^2\frac{x}{2}-1}+2^{\sqrt{x+1}}$

16. $y=\sqrt{\frac{1}{x}(x^2-1)}\cdot\cos^2\frac{|x|}{3}++\lg\frac{1}{x+1}$

17. $y=\frac{x^3+\sin(3|x|-1)}{1-\cos^2 x}--\log_2(3^x-9)$

18. $y=\frac{\sin^2\sqrt[3]{x}}{x}+e^{-\sqrt{x^2-6x+8}}$

19. $y=(1+x)^{\sin\sqrt{x}}\cdot 2^{\cos^2\left(\frac{x}{x+2}\right)}$

20. $y=\frac{-x^2}{(2x+2)(2x-3)}+\frac{\log_2(\sqrt{x}-1)}{\sin 2x}$

21. $y=2^{|x|}\cdot\ln|\sin x^4|--\cos^2\sqrt{4-x^2}$

22. $y=\left[\cos\left(e^{\sqrt{|x|-2}}+x^3\right)\right]^{2x}--\frac{|x|}{x-\sqrt{x}}$

23. $y=e^{x^2-1}+\frac{x\cdot\sin\frac{1}{x}}{\sqrt[4]{9-\sqrt{x}}}$

24. $y=\frac{x}{\cos\left(x-\frac{\pi}{2}\right)\sin^2\left(x-\frac{\pi}{2}\right)}++e^{\sqrt{x}-|x|}$

25. $y=\frac{2^{x^2}+\sqrt{16-x^2}}{\sqrt[3]{x-2}}+\mathrm{tg}^2\left(\frac{x}{x+2}\right)$

26. $y=\frac{1+\log_2(\sin 2x)}{1-2x}+\frac{\sqrt{|x|-1}}{x^3-27}$

27. $y=\frac{\lg_2(\sqrt{x}-1)}{x-\sqrt{x}}$

28. $y=\frac{\sqrt{x}-2}{2^{\sqrt{x-1}}}$

29. $y=\frac{\arccos(1+x)}{x-\sqrt{x}}$

30. $y=$

$$\frac{\arcsin\sqrt{2-x}}{x^2-x}$$

Контрольные вопросы

1. Зачем нужен блок описания переменных?
2. Какие математические функции имеются в Паскале?
3. Опишите действия, которые выполняются операторами Write, Writeln.
4. Почему при записи выражений запрещается использовать квадратные и фигурные скобки?
5. Как осуществляется вывод текстовой информации?
6. Чему равно выражение 3*7 div 2 mod 6/3 – trunk(sin(1))?
7. Какой смысл следующих предложений:

(*Вывод результатов*)

Write(‘ Вывод результатов ’)

8. Правильны ли операторы присваивания:

а. k:=k mod 13 + k*Cos(0);

б. x:=x*12 div 13 +x/4?

Ответ обосновать

9. Запишите на Паскале операторы для вычисления функций:

а. $y = \sqrt[3]{x}$;

б. $y = \sqrt[5]{\sin x}$;

в. $y = \lg \sqrt[3]{x}$;

г. $y = \operatorname{arcctg} x$;

д. $y = \sin 20°$;

е. $z = a^b$

10. Составьте программу, которая по заданным величинам углов треугольник и радиусу описанной окружности вычисляет длины его сторон.

11. Составьте программу, которая вычисляет площадь равнобочной трапеции по заданным длинам a и b и углу α при большем основании.

12. Составьте программу, которая по заданным длинам катета и гипотенузы прямоугольного треугольника вычисляет длину второго катета и радиус вписанной окружности.

13. Что будет выведено на экран в результате выполнения программы:

```
Program Bad: Var k,
I,j: integer
; fi:real;begin write(‘nmbr’
); read(
Fi);i:=trunk(fi+1/2);j:=round(fi);fi:=((int(fi)-((i-j)
mod -2-frac(fi
)))/1*2);writeln(fi,
k)end.
```

Сначала запишите текст программы в удобочитаемом виде.

Лабораторная работа № 7

ПРОГРАММИРОВАНИЕ ФОРМУЛ С РАЗВЕТВЛЕННОЙ СТРУКТУРОЙ.

Цель работы: изучение работы операторов перехода – безусловного, условного, выбора; программирование формул с разветвлённой структурой.

Основные понятия

Операторы выполняются, как правило, последовательно в том порядке, в каком они расположены в программе. Иногда возникает необходимость изменить естественный (линейный) порядок выполнения в зависимости от результата (исхода) каких-либо действий (некоторый аналог размышления). В Паскале для этой цели введены следующие операторы перехода: if, case, goto.

Условный оператор if имеет такую структуру:

If Y then S1 else S2;

где if (если) – зарезервированное слово; Y – некоторое условие (логическое выражение); then (то) – зарезервированное слово; S1, S2 – простые или составные операторы; else (иначе) – зарезервированное слово.

При выполнении оператора if вначале проверяется условие Y; если это условие истинно, то выполняется S1, если условие ложно – выполняется S2. Условие Y записывается с помощью арифметических выражений, логических констант True (истина), False (ложь), знаков операций сравнения «<», «<=», «=», «<>», «>», «>=», логических операций not (нет), and (и), or (или).

Пример. Для вычисления величины Z по формулам

$$Z = \begin{cases} \sin(x+y), & \text{если } y > x^2 \\ \cos(x-y), & \text{если } y \le x^2 \end{cases}$$

достаточно в программу вставить оператор

if y > x*x then z := sin(x+y) else z := cos(x-y);

Условные операторы могут быть вложенными друг в друга. В компилятор заложено правило: альтернатива else считается принадлежащей ближайшему условному оператору, не имеющему ветви else. При вложениях условных операторов самое главное – не запутаться в вариантах сочетаний условий. Частично этому может помочь ступенчатая форма записи операторов (и это является хорошим стилем записи). Более радикальным является использование составных операторов (расстановка операторных скобок begin … end;) для внутренних операторов if. Составной оператор – это последовательность (группа) операторов, записанных между двумя зарезервированными словами begin и end.

Пример. Вычислить значение функции y, если x вводится с клавиатуры, а функция задана формулами

$$y = \begin{cases} 2x, & \text{если } x \le 2, \\ 8 - 2x, & \text{если } 2 < x < 4, \\ (4 - x)^2, & \text{если } x \ge 4. \end{cases}$$

При составлении и записи алгоритма решения задачи можно использовать блок-схемы, а можно алгоритм записывать на псевдоязыке с помощью операторов Паскаля и комментариев, используя метод пошаговой детализации («нисходящего» программирования). При этом получается достаточно комментированная программа.

Общую схему программы представим так:

```
Program Primer2;
        (*вводим в употребление переменные
(описываем их)*)
begin
        (*- этап I – ввод значения x *)
        (*- этап II – нахождение y при заданном x *)
        (*- этап III – вывод результатов *)
End.
```

Реализация каждого этапа, кроме второго, достаточно тривиальна и ясна из окончательного текста программы (см. ниже). Детализируем (уточняем) второй этап, например, таким образом:

```
                (* нахождение y при заданном x *)
if x<=2 then {проверка условия x<=2}
                 {для y есть формула - 1}
        else    {т.е. для x>2}
                {необходимо уточнение значения x:
                больше оно или меньше 4 т.е. проверка
                следующего условия}
```

Теперь детализируем дальше:

```
                (* нахождение y при заданном x *)
if x<=2 then {проверка условия x<=2}
                y := 2*x
        else    {т.е. для x>2}
                begin
                  if x<4 then {проверка следующего условия}
                                {для y есть формула - 2}
                        else
                                {для y есть формула - 3}
                end;
```

Окончательный текст программы (с подробными комментариями) примет такой вид:

```
Program Primer3;
        (*вводим в употребление переменные (описываем их)*)
```

```
Var
        x, y : Real;        {x, y вещественные}
begin
        (*- этап I – ввод значения x *)
write(‘Введите x‘);         {вывод на экран «подсказки»}
readln(x);                  {ввод с клавиатуры значения x}
        (*- этап II – нахождение y при заданном x *)
if x<=2             {проверка условия x<=2}
then
        y := 2*x  {действие в случае истинности x<=2}
else
        begin     {действие в случае ложности x<=2}          if x<4 {проверка
следующего условия}
          then
                y:=8-2*x  {действие в случае его истинности}
          else
                y:=sqr(4-x)          {действие в случае его ложности
                                          т.е. при x>=4}
        end;
        (*- этап III – вывод результатов *)
writeln(‘x=’,x:7:3,’ y=’,y:7:3); {вывод на экран результатов}
End.
```

Отметим, что допускается также сокращённая форма условного оператора:

if Y then S1;

При выполнении этого оператора, если Y истинно, то S1 выполняется, в противном случае S1 пропускается.

В случае сложного разветвления более оптимальным (чем использование условных операторов) может быть применение оператора case. Ограничением в использовании оператора является то, что ключ выбора может быть типа Real или String.

Замечание. При составлении тестов необходимо подготовить такие исходные данные и в таком количестве, чтобы программой были выполнены действия по каждому разветвлению. В частности, в рассмотренном примере необходимо подготовить тесты со значениями x (хотя бы по одному) из каждого интервала $x \le 2$, $0<x<4$, $x \ge 4$. В противном случае гарантировать правильность выполнения программы для произвольных исходных данных не представляется возможным.

Задание к работе

1. Составить алгоритм вычисления значений функции, соответствующей конкретному варианту (табл. 7.1).

2. Составить программу, в которой предусмотреть ввод исходных данных с клавиатуры, вычисления значений функции, вывод на экран результатов выполнения условия и значения функции.

Пример: Для исходных данных $x = 0$, $y = 5$ и выражения

$$Z = \begin{cases} |\sin(x)|, & \text{если } 5 - |x - y| < 10 \\ \mathrm{ctg}^2(x - y), & \text{если } 5 - (x - y) = 10 \\ \frac{\cos(x)}{|x|+2}, & \text{в остальных случаях} \end{cases},$$

программа должна вывести на экран следующий текст:

Исходные данные: x = 0, y = 5.

5 - (0 - 5) = 10, значит, считаем функцию Z по второй формуле. В результате Z = 0,087.

3. Ввести текст программы в компьютер и отредактировать его.
4. Подготовить тестовые исходные данные, предусматривающие выполнение каждого из условий задания.
5. Отладить программу.
6. Выполнить программу (получить результаты) для каждого из условий задания.

Содержание отчета

1. Постановка задачи (конкретный вариант)
2. Текст программы, тестовые примеры и результаты вывода на экран.
3. Блок-схема алгоритма программы.

Варианты заданий

Таблица 7.1

Вариант	Функция
1	$Z = \begin{cases} e^{x^2-y}, & \text{если } y > x^2 \\ x + y, & \text{если } y < x^2 \\ y, & \text{если } y = x^2 \end{cases}$
2	$Z = \begin{cases} \sin(x + y) + x, & \text{если } x = y \\ \lvert x\rvert, & \text{если } x < y \\ y, & \text{если } x > y \end{cases}$
3	$Z = \begin{cases} \min(\cos(x), y), & \text{если } x > 0, y > 0 \\ \mathrm{tg}(\ln(xy)), & \text{если } x < 0, y < 0 \\ 0, & \text{в остальных случаях} \end{cases}$
4	$Z = \begin{cases} e^{-x}, & \text{если } \lvert x\rvert = y \\ \mathrm{tg}(x), & \text{если } \lvert x\rvert < y \\ x - y, & \text{если } \lvert x\rvert > y \end{cases}$
5	$Z = \begin{cases} \lvert x\rvert - y^3, & \text{если } x < -3 \\ \cos(y), & \text{если } x > 5 \\ 5, & \text{в остальных случаях} \end{cases}$

Вариант	Функция
6	$Z=\begin{cases} e^{\lvert x\rvert}+\cos(x), & \text{если } x\cdot y>15 \\ x^3+x+5, & \text{если } x\cdot y\le 10 \\ \sqrt{\lvert y\rvert}+\mathrm{tg}(x), & \text{в остальных случаях} \end{cases}$
7	$Z=\begin{cases} x^y+2, & \text{если } \sin(x)>0.5 \\ \mathrm{tg}(x)-y, & \text{если } \sin(x)=0.5 \\ \sqrt{\lvert x\rvert}, & \text{в остальных случаях} \end{cases}$
8	$Z=\begin{cases} cos\lvert x\rvert\cdot y, & \text{если } y\cdot\lvert x\rvert=5 \\ \sqrt{\lvert x-y\rvert}, & \text{если } y\cdot\lvert x\rvert=-5 \\ \sqrt{\lvert x\rvert}+1, & \text{в остальных случаях} \end{cases}$
9	$Z=\begin{cases} \min(x^2,y^3), & \text{если } x+y>5 \\ e^{x-y}, & \text{если } 0<x+y\le 5 \\ 2, & \text{в остальных случаях} \end{cases}$
10	$Z=\begin{cases} \max(x,y)+\min(x^2,y^2), & \text{если } x>0, y>0 \\ e^{\cos(x)+\sin(y)}, & \text{в остальных случаях} \end{cases}$
11	$Z=\begin{cases} \lvert\sin(\mathrm{x})\rvert, & \text{если } y\cdot\lvert x\rvert=8 \\ \sqrt{\lvert x-y\rvert}, & \text{если } y\cdot\lvert x\rvert>10 \\ \sin(xy), & \text{в остальных случаях} \end{cases}$
12	$Z=\begin{cases} \sin(\mathrm{xy}), & \text{если } x\ge 0, y<x \\ \max(x^2,y^2,xy), & \text{если } x^2+y^2\le 5, x>0 \\ \cos(xy), & \text{в остальных случаях} \end{cases}$
13	$Z=\begin{cases} \lvert\sin(\mathrm{x})\rvert, & \text{если } x+y>5 \\ \mathrm{ctg}^2(x-y), & \text{если } 0<x+y\le 5 \\ \dfrac{\cos(x)}{\lvert x\rvert+2}, & \text{в остальных случаях} \end{cases}$
14	$Z=\begin{cases} \cos(x)+\min(x,y), & \text{если } x^2>y^2, x>0 \\ \cos^2(x)+\mathrm{tg}^2(\mathrm{xy}), & \text{в остальных случаях} \end{cases}$
15	$Z=\begin{cases} 0, & \text{если } x>0, y>0 \\ \mathrm{tg}(xy), & \text{если } x<0, y<0 \\ \min(\cos(x,y), & \text{в остальных случаях} \end{cases}$
16	$Z=\begin{cases} \sin^3(\mathrm{x}^2)-\mathrm{y}, & \text{если } 5-(x-y)<10 \\ \lvert x-y\rvert+5, & \text{если } 5-(x-y)=10 \\ \mathrm{ctg}^{1/3}\,x, & \text{в остальных случаях} \end{cases}$
17	$Z=\begin{cases} \cos\lvert\mathrm{x}\rvert\cdot\mathrm{y}, & \text{если } y\cdot\lvert x\rvert=5 \\ \sqrt{\lvert x-y\rvert}, & \text{если } y\cdot\lvert x\rvert>5 \\ \sqrt{\lvert x\rvert}+1, & \text{в остальных случаях} \end{cases}$
18	$Z=\begin{cases} \lvert x\rvert^y-y^2, & \text{если } x<-4 \\ \cos\lvert y\rvert, & \text{если } x>7 \\ 5, & \text{в остальных случаях} \end{cases}$

Вариант	Функция
19	$Z = \begin{cases} \lvert\sin(x)\rvert, & \text{если } e^x < 6 \\ \mathrm{ctg}^2(x-y), & \text{если } e^x > 12 \\ \dfrac{\cos(x)}{\lvert x\rvert + 2}, & \text{в остальных случаях} \end{cases}$
20	$\left.\begin{array}{r} z_1 = \sqrt[3]{\mathrm{ctg}(x)} - x^2, \\ z_2 = e^{2x-y}, \end{array}\right\}$ если $x^2 - y < -2$ $\left.\begin{array}{r} z_1 = \sin(y - \lvert x\rvert), \\ z_2 = \sqrt{\lvert x - y\rvert}, \end{array}\right\}$ если $x^2 - y \geq -2$
21	$Z = \begin{cases} \ln\lvert x-y\rvert, & \text{если } y > x, x > 0 \\ \sqrt{\lvert x\rvert + 4}, & \text{если } \lvert xy\rvert > 5 \\ \max(x^3, y^3), & \text{в остальных случаях} \end{cases}$
22	$Z = \begin{cases} x^y + 10, & \text{если } \sin(x) > 0.8 \\ \cos\lvert x\rvert \cdot y, & \text{если } \sin(x) = 0.8 \\ \sqrt{\lvert y\rvert}, & \text{в остальных случаях} \end{cases}$
23	$\left.\begin{array}{r} z_1 = \lvert\sin(x)\rvert + y, \\ z_2 = \sqrt{\lvert x\rvert}, \end{array}\right\}$ если $x - y > 2$ $\left.\begin{array}{r} z_1 = e^y, \\ z_2 = x^2 + y - x, \end{array}\right\}$ если $x - y \leq 2$
24	$\left.\begin{array}{r} z_1 = \lvert x\rvert + \lvert y\rvert, \\ z_2 = e^{\lvert x+y\rvert}, \end{array}\right\}$ если $\sqrt{\lvert xy\rvert} < 50$ $\left.\begin{array}{r} z_1 = \sin(y - \lvert x\rvert), \\ z_2 = \sqrt{\lvert x - y\rvert}, \end{array}\right\}$ если $\sqrt{\lvert xy\rvert} \geq 50$
25	$\left.\begin{array}{r} z_1 = \sqrt{e^x} - 5, \\ z_2 = \sin(x^2 - y) - y^2, \end{array}\right\}$ если $x^2 - y < -2, x > 0$ $z_1 = x^{2/3} - (\sin(x))^y$, в остальных случаях
26	$Z = \begin{cases} \ln\lvert x-y\rvert, & \text{если } y > x, x > 0 \\ \cos^2(x) - y, & \text{если } x > y^3 + 1, x > 0 \\ \max(x^3, y^3), & \text{в остальных случаях} \end{cases}$
27	$\left.\begin{array}{r} z_1 = \mathrm{ctg}(xy) + 2, \\ z_2 = \lvert x\rvert - y, \end{array}\right\}$ если $x \leq 8.5$ $\left.\begin{array}{r} z_1 = \sqrt{\lvert x\rvert + 4}, \\ z_2 = \sin(x^3), \end{array}\right\}$ если $x > 8.5$
28	$\left.\begin{array}{r} z_1 = \sqrt{x^2 + y^2}, \\ z_2 = \mathrm{ctg}(x + y), \end{array}\right\}$ если $\lvert xy\rvert > 5$ $\left.\begin{array}{r} z_1 = e^{x-y}, \\ z_2 = \sin(xy), \end{array}\right\}$ если $\lvert xy\rvert \leq 5$
29	$\left.\begin{array}{r} z_1 = e^{-\lvert x\rvert + y}, \\ z_2 = \sin(x) - y, \end{array}\right\}$ если $x\lvert y\rvert < 5$

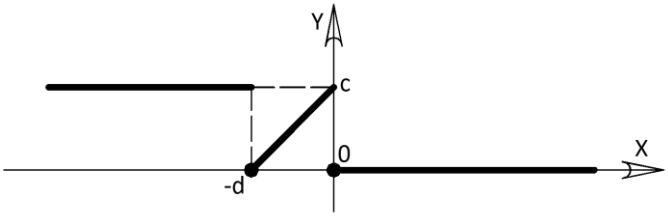
Вариант	Функция
	$\left.\begin{array}{l} z_1 = \sqrt{\lvert xy \rvert}, \\ z_2 = e^{\sin(x)}, \end{array}\right\}$ в остальных случаях
30	$Z = \begin{cases} \min(\cos(x), \sin(y), \mathrm{tg}(y)), & \text{если } x < 0, y > 0 \\ \max(x^2 + y^2, \ln\lvert xy \rvert), & \text{в остальных случаях} \end{cases}$

Практические задания для защиты

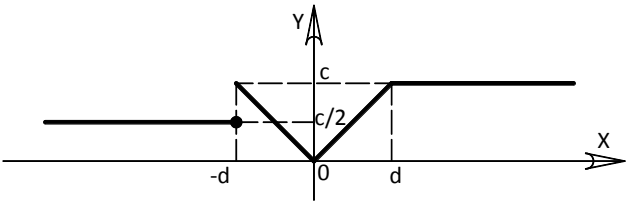
Для функции y=y(x), заданной графически, построить аналитическую зависимость. Составить блок-схему и программу вычисления величины y, при введённом пользователем значении аргумента x.

Варианты заданий для практической защиты

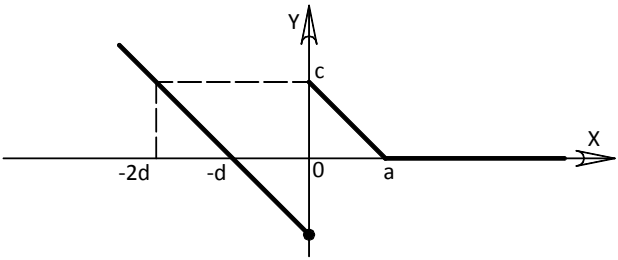
1



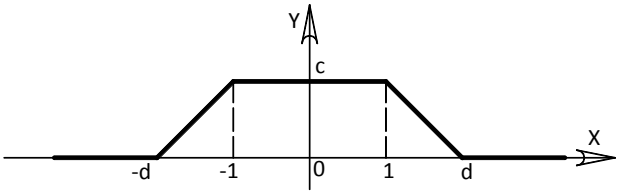
2



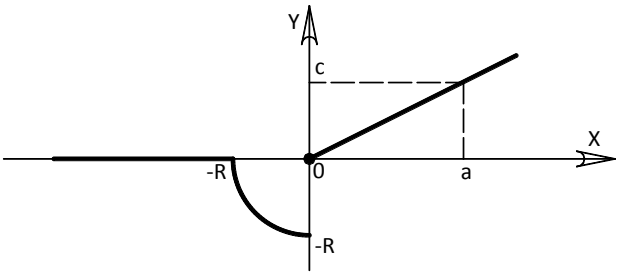
3



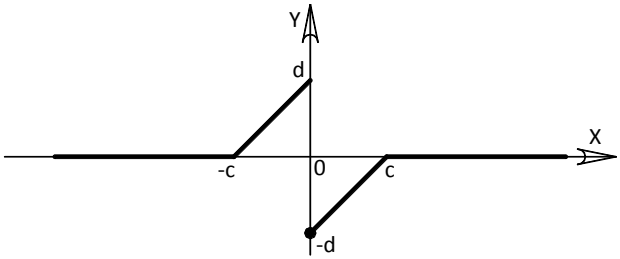
4



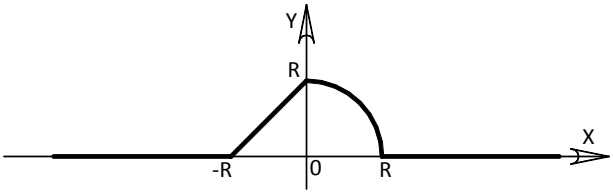
5



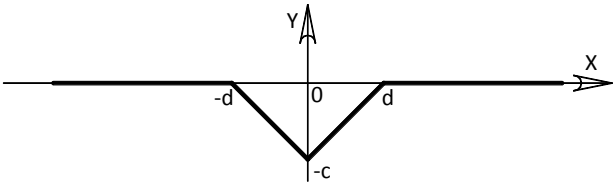
6



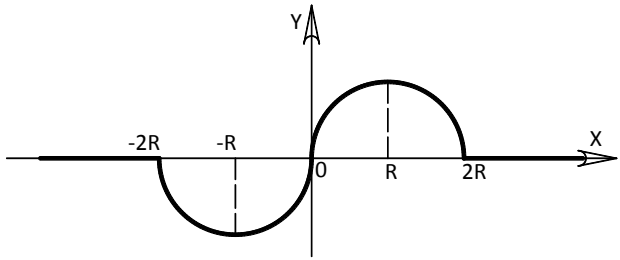
7



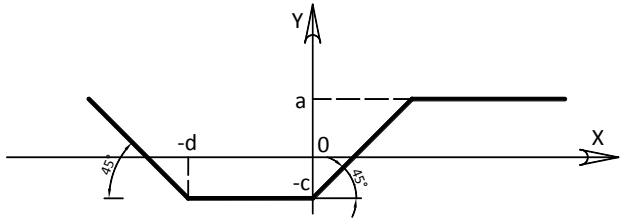
8



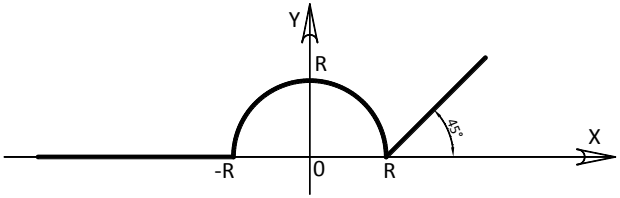
9



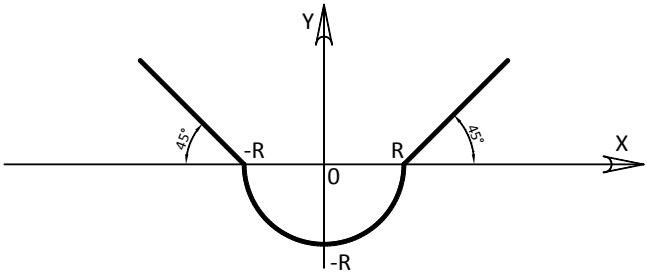
10



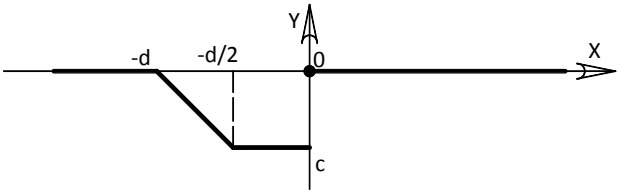
11



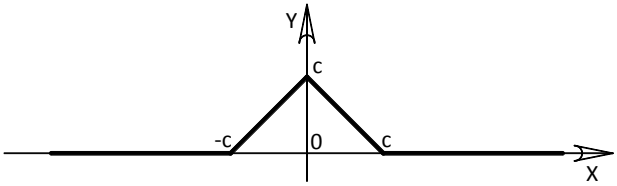
12



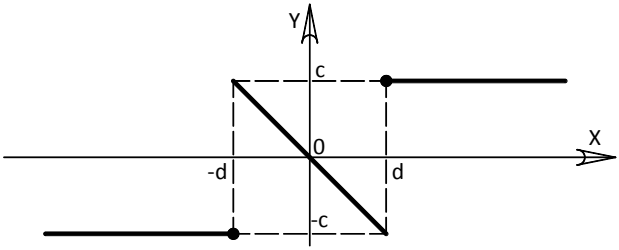
13



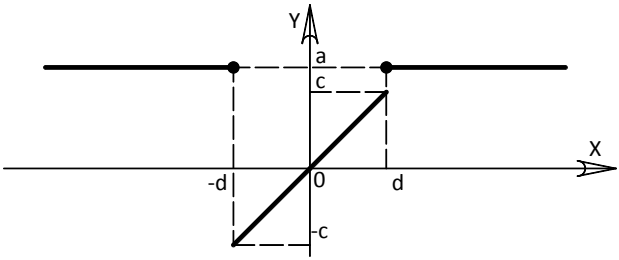
14



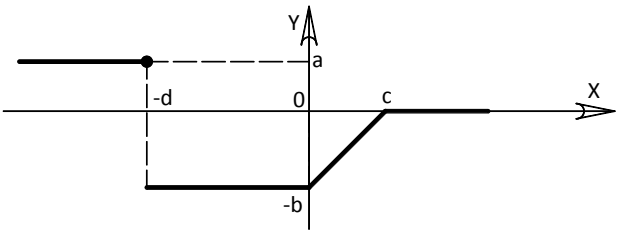
15



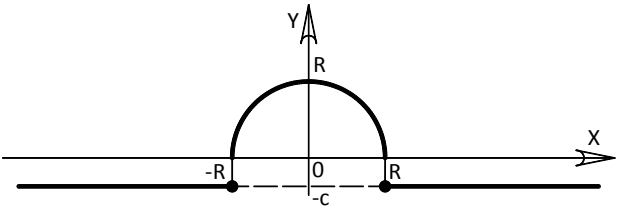
16



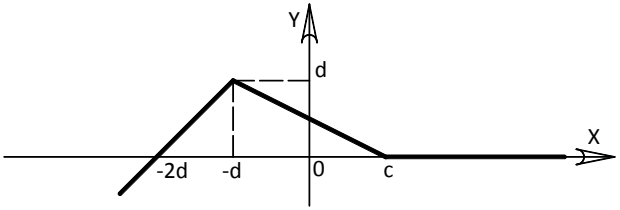
17



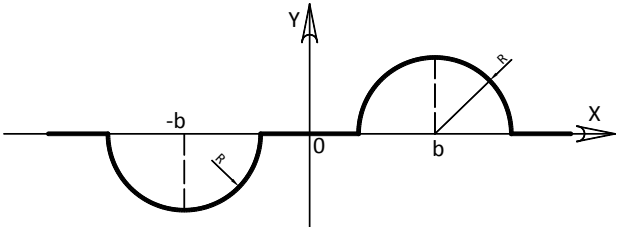
18



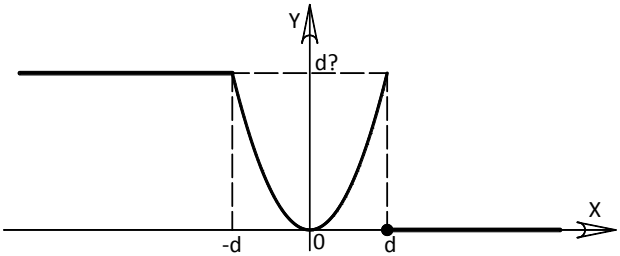
19



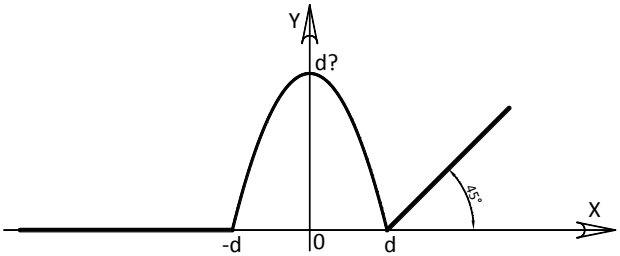
20



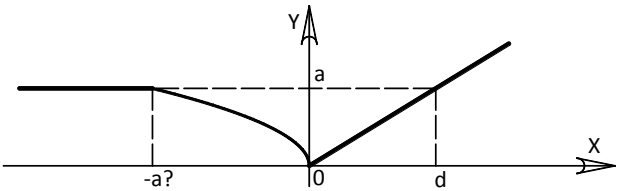
21



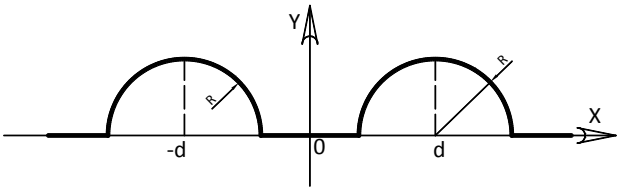
22



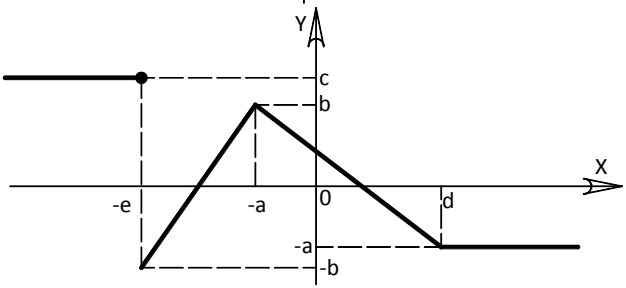
23



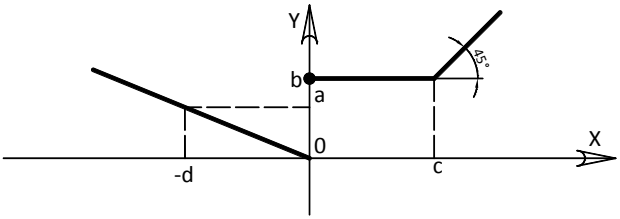
24



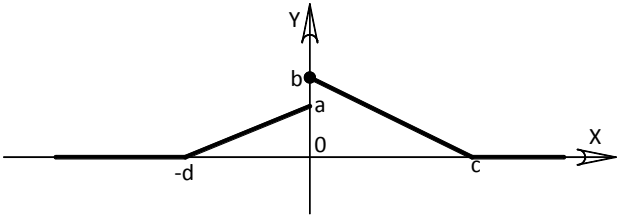
25



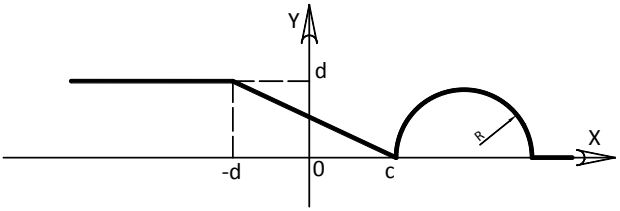
26



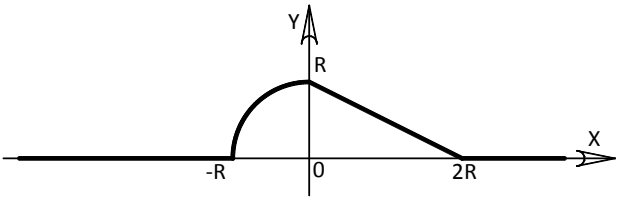
27



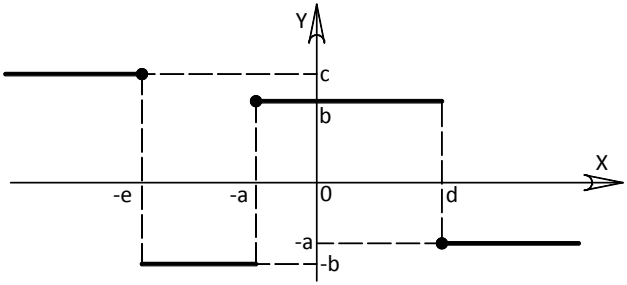
28



29



30



Контрольные вопросы

1. В какой форме может быть записан алгоритм программы?
2. Как выполняется оператор if?
3. Составить алгоритм и программу вычислений значений функций $y = \sqrt[3]{x}$, $y = \sqrt[5]{\sin x}$. Предусмотреть случаи, когда подкоренные выражения принимают отрицательные значения.
4. Составьте алгоритм и программу для нахождения наибольшего значения из трёх чисел max(a, b, c).
5. Может ли условный оператор содержать оператор перехода (условный оператор)?
6. Составьте алгоритм и программу, которая поменять местами числовые значения переменных X и Y в случае, если $X > Y$.
7. Составьте программу, которая посчитает величину $y = \lg x$, если $x > 0$, и выдаст текстовое сообщение об ошибке, если $x \leq 0$.
8. Составьте алгоритм и программу, которая по заданным положительным действительным числам x, y, z определит: а) существует ли треугольник с длинами сторон x, y, z; б) ответит является ли он остроугольным (тупоугольным) в случае его существования.
9. Составьте алгоритм и программу, которая выяснит для заданных положительных действительных чисел a, b, c, x, y, пройдет ли кирпич с ребрами a, b, c в прямоугольное отверстие со сторонами x, y. Просовывать кирпич в отверстие разрешается только так, чтобы каждое из его ребер было параллельно или перпендикулярно каждой из сторон отверстия.

Лабораторная работа № 8

ОДНОМЕРНЫЕ МАССИВЫ. ЦИКЛЫ С ЗАДАННЫМ ЧИСЛОМ ПОВТОРЕНИЙ

Цель работы: получение навыков в использовании оператора цикла For; овладение практическими навыками работы с массивами.

Основные понятия

При решении многих задач вычислительный процесс имеет циклический характер, т.е. одни и те же действия повторяются при различных значениях переменных. Многократно повторяющееся выполнение одних и тех же операторов называют циклом. Для организации циклов с заранее известным числом повторений предусмотрен оператор For, который имеет две модификации:

For i:= S1 to S2 do P;
или
For i:= S1 downto S2 do P;

где For (переводится с английского как «для») – зарезервированное слово; i – параметр цикла – переменная любого порядкового типа (но не Real); S1 – начальное значение параметра цикла (выражение того же типа); to (увеличивать до) – зарезервированное слово; downto (уменьшать до) – зарезервированное слово; S2 – конечное значение параметра цикла (выражение того же типа); P – простой или составной оператор.

В качестве параметра цикла может быть только переменная, а в качестве S1 и S2 могут быть константы, переменные или выражения того же типа (только целочисленного или перечисляемого). Шаг цикла всегда постоянный и равен «интервалу» между двумя ближайшими значениями параметров цикла.

Выполнение цикла начинается с присвоения параметру цикла i стартового значения S1. Затем проверяется, не превосходит ли оно значение S2 (случай to) или не является ли оно меньше S2 (случай downto). Если это утверждение истинно, то цикл считается завершенным и выполняется следующий (за оператором цикла) оператор программы. В противном случае выполняется оператор P (тело цикла) и после этого параметр цикла меняет своё значение на следующее (согласно заголовку цикла). Далее алгоритм повторяется. Запрещается изменять параметр i, а также S1 и S2 внутри тела цикла.

Формально после завершения цикла значение его параметра считается неопределённым, но на практике, как правило, оно оказывается равным конечному значению своего диапазона (однако полагаться на это в программах не рекомендуется, так как результаты могут быть не предсказуемыми).

Оператором For чаще всего обрабатываются массивы (Array). Массивом называют конечный (фиксированный) набор элементов одного и того же типа, к каждому из которых можно обращаться по его номеру (индексу). Индивидуальное имя получает только весь набор (вся структура данных), а для компонент определяется лишь порядок следования и общее их количество.

Как и все другие переменные, массив (до того, как он будет использоваться) должен быть описан в программе, что может быть выполнено (в разделе Var объявления переменных) следующим образом:

<имя> : Array [<диапазон индексов>] of <тип элементов>;

Например, для хранения двадцати элементов с индексами $a_1, a_2, \dots, a_{20}$ можно использовать массив, описанный в начале программы так:

Var A : Array [1..20] of Real;

Для обращения (ссылки) к отдельному элементу массива используется запись вида <имя массива> [<индекс>] – так вызывается переменная с индексами. Другими словами, индексы массива записываются внутри квадратных скобок, и в рассматриваемом примере элементы A[1], A[2], …, A[20] будут переменными типа Real. В общем случае ничто не обязывает объявлять диапазон индексов массива числами. В качестве индекса может быть использовано выражение любого порядкового (перечислимого) типа или его диапазона, значение которого определяет номер компоненты массива. Количество компонент массива, их упорядоченность и тип должны задаваться явно, т.е. быть определёнными до начала выполнения программы (не допускается динамическое определение). При этом сумма размеров всех компонентов массива не должна превосходить 64К.

Массивы (наряду с понятием записей Record, множеств Set, файлов File)- представляют собой одно из средств расширения стандартных (создания новых) типов Паскаля. Введение новых типов производится в разделе Type. Например, тот же массив A может быть описан так:

```
Type Vector = Array [1..20] of Real;   {в разделе типов}
Var A : Vector;                        {в разделе переменных}
```

Такой способ становится единственно возможным для описания формальных параметров структурированного типа в процедурах и функциях.

Рассмотрим пример обработки одномерного массива. Составим программу которая для заданного массива $b_1, b_2, \dots, b_{40}$ найдёт количество положительных элементов и наименьший положительный элемент.

Текст программы (с пояснением алгоритма) может быть следующим:

```
Program Rab5;
Var
```

```
        i      : integer;                {номер элемента}
        k      : integer;                {количество положительных
                                         элементов}
        minB: real;                      {наименьший положительный
                                         элемент}
        b      : array[1..40] of Real;   {массив для хранения заданных
                                         элементов}
Begin
(* установка начальных значений *)
K := 0;            {обнуление: положительных элементов пока нет}
minB := 1E19;      {искомый элемент временно = большому числу
                   10^19, полагаем, что вводимые элементы будут
                   не больше этого числа}
For i= 1 to 40 do  { = в цикле перебираем все элементы = }
  begin
(* организуем ввод элементов*)
      write('b[' , i , '] = ');{вывод «подсказки» какой номер вводится}
      readln(b[i]);        {ввод элемента}
(* вычисления *)
      if b[i] > 0 then     {проверка: положительный ли элемент?}
          begin
            k : k + 1;               {кол-во положит. увеличиваем на 1}
                                     {ищем минимальный}
            if b[i] < minB           {текущий элемент оказался меньше}
              then minB := b[i];     {искомый элемент равен текущему}
          end;
  end;  { = конец цикла = }
(* вывод результатов *)
if k > 0      then    {Проверка: а есть ли среди введенных положит.?}
        writeln(' k = ' , k , ' minB = ' , minB:7:3);     {выводим ответ 1}
              else
        writeln('Нет положительных элементов');     {выводим ответ 2}
end.
```

Задание к работе

1. Для введённого массива чисел (x_1, x_2, …, x_{10}) определить значение величины z, а также элементы массива (y_1, y_2, …, y_{20}) (табл. 5.1)

2. На печать вывести результаты вычислений в виде:

Входной массив X: x_1 x_2 x_3 x_4 x_5
X_6 x_7 x_8 x_9 x_{10}

промежуточная величина Z = ***.***

выходной массив Y: y_1 y_2 y_3 y_4 y_5
y_6 y_7 y_8 y_9 y_{10}

y_{11} y_{12} y_{13} y_{14} y_{15}
y_{16} y_{17} y_{18} y_{19} y_{20}

3. Ввести программу в компьютер и отладить её.

Содержание отчета

1. Постановка задачи (конкретный вариант)
2. Текст программы, тестовые примеры и результаты вывода на экран.
3. Блок-схема алгоритма программы.
4. «Ручной» счёт для проверки правильности работы программы.

Варианты заданий

Таблица 8.1

Вар.	**Формулы для определения промежуточной величины Z**	**Формулы для определения элементов массива ($y_1, y_2, \ldots, y_{20}$)**
1	$Z = \max\limits_i(x_i), (i = 1, 2, \ldots, 10)$	$y_i = e^{\cos(zi)}$
2	$Z = \min\limits_i(x_i), (i = 1, 3, 5, 7, 9)$	$y_i = \sin(i + z)$
3	$Z = \min\limits_i(\cos x_i), (i = 1, 2, \ldots, 7)$	$y_i = \sqrt[3]{i + z}$
4	$Z = \sum\limits_{i=1}^{10} x_i$	$y_i = \lvert i + z \rvert$
5	Z – первое отрицательное число среди элементов $x_1, x_2, \ldots, x_{10}$	$y_i = \mathrm{tg}\,(i - z)$
6	$Z = \prod\limits_{i=1}^{10} x_i$	$y_i = \sin(i) \cdot z$
7	Z – первое число среди элементов $x_1, x_2, \ldots, x_{10}$, для которого $\lvert x_i \rvert < 2$	$y_i = e^{\sin z} - i$
8	$Z = \sum\limits_{i=3}^{8} x_i \cos x_{i+1}$	$y_i = \mathrm{tg}\, z - i$
9	$Z = \prod\limits_{i=1}^{9} (x_i - \cos x_{i+1})$	$y_i = \dfrac{1}{z^2 + i}$
10	$Z = \max\limits_i \left(\dfrac{x_i}{\sin x_i + 5}\right), (i = 1, 2, \ldots, 10)$	$y_i = i - \sin z$
11	Z – первое отрицательное число среди $a_i = x_i \sin x_i$, (i = 1, 2, …, 10)	$y_i = \dfrac{z + i}{2}$
12	Z – первое число $a_i = \sin x_i - e^{\sin x_i}$, для которого $\lvert a_i \rvert < 3$, (i = 1, 2, …, 10)	$y_i = e^{\sin i} + z$

Вар.	Формулы для определения промежуточной величины Z	Формулы для определения элементов массива ($y_1, y_2, \ldots, y_{20}$)
13	$Z = \sum_{i=1}^{3} x_{3i}$	$y_i = z^3 - \sin i$
14	$Z = \prod_{i=1}^{5} x_{2i}$	$y_i = \cos z + \sqrt[3]{i}$
15	Z – первое число $a_i = \cos x_i$, для которого $\sqrt[3]{\lvert a_i \rvert} < 0.5$, ($i$ = 1, 2, …, 10)	$y_i = \cos(iz) - \sqrt[5]{i}$
16	Z – первое число x_i для которого $0 < x_i < 5$, (i = 1, 2, …, 10)	$y_i = \sqrt[3]{z^2} + \cos i$
17	Z – первое число $a_i = e^{\cos x_i}$, для которого $\sqrt[3]{\lvert a_i - 5 \rvert} < 2$, ($i$ = 1, 2, …, 10)	$y_i = \ln\lvert z \rvert + i$
18	$Z = \prod_{i=1}^{3} x_{3i}$	$y_i = \sin z \cdot \cos i$
19	$Z = \sum_{i=1}^{2} (\lvert x_{4i} \rvert + x_{2i})$	$y_i = \operatorname{ctg} i - \sqrt[3]{\lvert z \rvert}$
20	$Z = \sum_{i=1}^{8} (\cos x_{i+2} + 3)$	$y_i = \operatorname{tg} z - i^2$
21	$Z = \prod_{i=1}^{8} (x_i \cdot x_{i+2} - x_{i+1})$	$y_i = \frac{\ln i}{\lvert z \rvert + 2}$
22	$Z = \max_i (x_i^2 - x_{i+2}), (i = 1, 2, \ldots, 7)$	$y_i = \lvert \sin i \rvert^z$
23	$Z = \sum_{i=1}^{4} (x_i + x_{i+1} + x_{2i})$	$y_i = \sqrt{\lvert \cos z \rvert + i}$
24	Z – первое число из $a_i = 0.1 \cdot x_i$ для которого $0 < a_i < 3$, (i = 1, 2, …, 10)	$y_i = \frac{i + (i + z)^2}{\lvert \sin z \rvert + 2}$

Вар.	Формулы для определения промежуточной величины Z	Формулы для определения элементов массива ($y_1, y_2, \ldots, y_{20}$)
25	$Z = \sum_{i=1}^{5}(x_i + \sin x_i)$	$y_i = \ln^2 i + z \cdot i$
26	$Z = \prod_{i=1}^{5} 3x_{2i}$	$y_i = \sqrt{\lvert\sin z\rvert + i}$
27	$Z = \sum_{i=4}^{9} x_i \, sin \, x_{i+1}$	$y_i = \frac{z+i}{\sin i}$
28	$Z = \prod_{i=1}^{5} x_{2i-1}$	$y_i = \sin z \cdot \mathrm{tg}\, i$
29	$Z = \sum_{i=2}^{8}(x_{i-1} - 2x_i + x_{i+1})$	$y_i = \frac{\lvert\cos z\rvert}{i+3}$
30	$Z = \sum_{i=1}^{5} x_{2i-1}$	$y_i = i - \cos z$

Практическое задание для защиты

Составить блок-схему и написать текст программы для решения приведенной в варианте задания задачи

Варианты заданий для практической защиты

1. Вычислить сумму и количество элементов массива X больших числа a, где a – среднее арифметическое элементов массива X. Элементы массива вычисляются по формуле $X_i = \sqrt{i-1} \cdot \sin(i)$, где i = 1, 2, …, 50.

2. Вычислить среднее арифметическое отрицательных элементов массива X и их количество. Элементы массива вычисляются по формуле $X_i = \sqrt[3]{i-10} \cdot \sin(i)$, где i = 1, 2, …, 20.

3. массива X Элементы массива вычисляются по формуле $X_i = \sqrt[3]{i-10} \cdot \cos(i)$, где i = 1, 2, …, 20.

4. Переписать положительные элементы массива X в массив Y и подсчитать их количество. Элементы массива X вычисляются по формуле $X_i = \sqrt[3]{i-1} \cdot \sin(i)$, где i = 1, 2, …, 20.

5. Переписать отрицательные элементы массива X в массив Y и подсчитать их количество. Элементы массива X вычисляются по формуле $X_i = \sqrt[3]{i-10} \cdot \sin(i-0{,}5)$, где i = 1, 2, …, 20.

6. Определить максимальный элемент массива X и его порядковый номер. Элементы массива вычисляются по формуле $X_i = \sqrt{|i-10|} \cdot \mathrm{tg}(i)$, где i = 1, 2, …, 10.

7. Определить минимальный элемент массива X и его порядковый номер. Элементы массива вычисляются по формуле $X_i = (-1)^i \cdot (i^2 - 2i)$, где i = 1, 2, …, 10.

8. Найти максимальный и минимальный элементы массива X и поменять их в массиве местами. Элементы массива вычисляются по формуле $X_i = (-1)^i \cdot (i^2 - 3i)$, где i = 1, 2, …, 10.

9. Найти количество положительных и отрицательных лементов в массиве X. Элементы массива вычисляются по формуле $X_i = (-1)^i \cdot (i^2 - 5i + 6)$, , где i = 1, 2, …, 10.

10. Расположить в массиве Y сначала положительные, а затем отрицательные элементы массива X. Элементы массива вычисляются по формуле $X_i = i^2 - 9i + 8$, где i = 1, 2, …, 10.

11. Определить сумму элементов массива X, значения которых кратны трём. Элементы массива вычисляются по формуле $X_i = i^2 - 2i - 3$, где i = 1, 2, …, 10.

12. Определить сумму элементов массива X, значения которых кратны двум. Элементы массива вычисляются по формуле $X_i = i^2 + 3i + 2$, где i = 1, 2, …, 10.

13. Вычислить сумму и количество элементов массива X больших по абсолютной величине заданного числа a, где $a>0$. Элементы массива вычисляются по формуле $X_i = i^2 - 10i + 1$, где i = 1, 2, …, 10.

14. Найти числа, встречающиеся среди элементов массива X более одного раза. Элементы массива вычисляются по формуле $X_i = i^2 - 5i + 4$, где i = 1, 2, …, 10.

15. Среди элементов массива X найти максимальный по модулю элемент и его порядковый номер. Элементы массива вычисляются по формуле $X_i = \sqrt[3]{i-5} \cdot (\sin(i) + \cos(i))$, где i = 1, 2, …, 15.

16. Переписать чётные элементы массива X в массив Y и подсчитать их количество. Элементы массива вычисляются по формуле $X_i = i^2 + 4i - 12$, где i = 1, 2, …, 15.

17. Переписать элементы массива X в массив Y в обратном порядке и найти порядковый номер максимального элемента в массиве Y. Элементы массива вычисляются по формуле $X_i = i^2 - 4i + 3$, где i = 1, 2, …, 10.

18. Среди элементов массива X найти числа, встречающиеся более двух раз. Элементы массива вычисляются по формуле $X_i = i^3 - 6i^2 + 11i - 6$, где i = 1, 2, …, 10.

19. Переписать нечётные элементы массива X в массив Y и подсчитать их количество. Элементы массива вычисляются по формуле $X_i = i^2 + 4i - 12$, где i = 1, 2, …, 10.

20. Переписать элементы массива X в массив Y в обратном порядке и найти порядковый номер минимального элемента в массиве Y. Элементы массива вычисляются по формуле $X_i = i^3 - 7i - 3$, где i = 1, 2, …, 10.

21. Найти число чётных положительных элементов в массиве X. Элементы массива вычисляются по формуле $X_i = (-1)^i \cdot (i^2 - 4i + 3)$, где i = 1, 2, …, 10.

22. Найти число нечётных положительных элементов в массиве X. Элементы массива вычисляются по формуле $X_i = (-1)^i \cdot (i^2 + 6i + 5)$, где i = 1, 2, …, 10.

23. Найти число нечётных отрицательных элементов в массиве X. Элементы массива вычисляются по формуле $X_i = (-1)^i \cdot (i^3 - 4(i + 3))$, где i = 1, 2, …, 10.

24. Найти число чётных отрицательных элементов в массиве X. Элементы массива вычисляются по формуле $X_i = (-1)^i \cdot (i^3 - 5i - 10)$, где i = 1, 2, …, 10.

25. Расположить элементы массива X в порядке возрастания. Элементы массива вычисляются по формуле $X_i = (-1)^i \cdot (i^3 - 6i^2 - 6)$, где i = 1, 2, …, 10.

26. Среди элементов массива целых чисел X найти сумму всех двузначных чисел. Элементы массива вычисляются по формуле $X_i = (i + 1)^2$, где i = 1, 2, …, 20.

27. Среди элементов массива целых чисел X найти сумму и количество всех нечётных двузначных чисел. Элементы массива вычисляются по формуле $X_i = 10i + i^2$, где i = 1, 2, …, 20.

28. Среди элементов массива целых чисел X найти среднее арифметическое всех чётных двузначных чисел. Элементы массива вычисляются по формуле $X_i = i^3 + i^2 + 10$, где i = 1, 2, …, 20.

29. Среди элементов массива целых чисел X найти сумму квадратов всех чётных двузначных чисел. Элементы массива вычисляются по формуле $X_i = 2i^2 + 3i - 5$, где i = 1, 2, …, 20.

30. Среди элементов массива целых чисел X найти сумму квадратов всех нечётных двузначных чисел. Элементы массива вычисляются по формуле $X_i = 3i^2 - 2i + 10$, где i = 1, 2, …, 20.

Контрольные вопросы

1. Дайте определение цикла в программе.
2. Дайте определение переменной с индексом.
3. Как в программе описываются переменные с индексами?
4. Что понимается под массивами?

5. Как описываются массивы?
6. Как организуется в программе ввод (вывод) массива?
7. Какие операции разрешены над массивами и их элементами?
8. Что будет посчитано в результате выполнения оператора

```
S:=0;
For k:=5 to 10 do S:=S+2*k-1; ?
```

9. Составьте алгоритм и программу для определения суммы отрицательных элементов массива z1, z2, … , z15.

10. Составьте алгоритм и программу, которая определяет номера первого отрицательного и первого положительного элемента массива z1, z2, … , z25.

11. Составьте алгоритм и программу, которая поменяет местами наибольший и наименьшей элементы массива z1, z2, … , z13.

12. Составьте алгоритм и программу, которая запишет элементы массива z1, z2, … , z17 в обратном порядке.

13. Составьте алгоритм и программу, которая вычислит среднее арифметическое элементов массива z1, z2, … , z123.

14. Составьте программу для вычисления $y = \sqrt{3 + \sqrt{6 + \ldots + \sqrt{96 + \sqrt{99}}}}$

15. Составьте программу для вычисления $y = \frac{1}{1^2} + \frac{1}{2^2} + \frac{1}{3^2} + \ldots + \frac{1}{100^2}$

Библиографический список

1. Основы локальных компьютерных сетей / Сергеев А.Н.– СПб.: Лань, 2016. – 184с.: ISBN 978-5-8114-2185-5
2. Справка PascalABC.NET [Электронный ресурс]: http://pascalabc.net/downloads/pabcnethelp/index.htm (дата обращения: 19.05.2018)
3. Информатика. Программирование в среде PascalABC.NET: методическое пособие. / Чернова С.Б., Старченко Д.Н. – Белгород: Изд-во БГТУ, 2015.– 90 с
4. ЭВМ, сети и периферийное оборудование: практикум для студентов направления бакалавриата 140400.62 - Электроэнергетика и электротехника / сост. В. В. Михайлов, Е. П. Коломыцева. – Белгород: Изд-во БГТУ, 2013. – 45 с.
5. Информационные технологии. Базовый курс / Костюк А.В., Бобонец С.А., Флегонтов А.В., Черных А.К. – СПб.: Лань, 2018. – 604с.: ISBN 978-5-8114-2906-6

Лабораторная работа № 1 7
УСТРОЙСТВО ПЕРСОНАЛЬНОГО КОМПЬЮТЕРА..... 7
Лабораторная работа № 2 19
ИЗУЧЕНИЕ IP-АДРЕСАЦИИ В ЛОКАЛЬНЫХ СЕТЯХ. 19
МАСКИ. ПОДСЕТИ И НАДСЕТИ. 19
Лабораторная работа № 3..... 44
ИЗУЧЕНИЕ СТРУКТУРЫ, СВОЙСТВ И СОЗДАНИЕ ТАБЛИЦ
РЕЗЯЦИОННОЙ БАЗЫ ДАННЫХ 44
Лабораторная работа № 4 65
ИЗУЧЕНИЕ ТИПОВ, ПОРЯДКА ПОСТРОЕНИЯ И РЕАЛИЗАЦИИ ЗАПРОСОВ
РЕЗЯЦИОННОЙ БД MS ACCESS..... 65
Лабораторная работа № 5 68
ИНТЕГРИРОВАННАЯ СРЕДА PascalABC.NET 68
Лабораторная работа № 6 79
ПРОГРАММИРОВАНИЕ АЛГОРИТМОВ ЛИНЕЙНОЙ СТРУКТУРЫ. ВВОД И
ВЫВОД ИНФОРМАЦИИ..... 79
Лабораторная работа № 7 89
ПРОГРАММИРОВАНИЕ ФОРМУЛ С РАЗВЕТВЛЕННОЙ СТРУКТУРОЙ. ... 89
Лабораторная работа № 8 102
ОДНОМЕРНЫЕ МАССИВЫ. ЦИКЛЫ С ЗАДАННЫМ ЧИСЛОМ
ПОВТОРЕНИЙ..... 102