Министерство образования и науки
Российской Федерации

Белгородский государственный технологический университет
им. В.Г. Шухова
Кафедра информационных технологий

Утверждено
научно-методическим советом
университета

ИНФОРМАТИКА

Методические указания к выполнению
лабораторных работ для студентов очной формы обучения
направления бакалавриата 230400 – Информационные
системы и технологии

Белгород
2013

УДК 007(07)
ББК 32.81я7
И74

Составители: ст. преп. С.Н. Рога
ст. преп. А.Г. Смышляев
доц. Ю.И. Солопов

Рецензент канд. техн. наук, проф. А.И. Полунин

Информатика: методические указания к выполнению
И74 лабораторных работ для студентов очной формы обучения направления бакалавриата 230400 – Информационные системы и технологии / сост: С.Н. Рога, А.Г. Смышляев, Ю.И. Солопов. – Белгород: Изд-во БГТУ, 2013. – 124 с.

Методические указания составлены в соответствии с учебным планом и рабочей программой, предназначены для приобретения студентами базовых навыков в работе с персональным компьютером, алгоритмизации и программирования на языке С/С++ и содержат теоретический материал и задания к выполнению десяти лабораторных работ.

Методические указания предназначены для студентов очной формы обучения направления 230400 – Информационные системы и технологии.

Данное издание публикуется в авторской редакции.

УДК 007(07)
ББК 32.81я7

СОДЕРЖАНИЕ

ПРАВИЛА ВЫПОЛНЕНИЯ ЛАБОРАТОРНЫХ РАБОТ

По курсу информатики предусмотрено выполнение ряда лабораторных работ. Студент обязан перед выполнением каждой лабораторной работы самостоятельно ознакомиться с теоретическим материалом и по ее результатам предоставить отчет. Все отчеты о выполнении лабораторных работ оформляются в отдельной тетради. Допускается оформлять отчеты в печатном виде на листах формата А4.
Отчет к лабораторным работам №№ 1–5 должен содержать:

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

Отчет к лабораторным работам №№ 6–10 должен содержать:

1. Заголовок лабораторной работы – номер работы, данные о студенте, слова «Выполнение» и «Защита», название и цель работы.
2. Содержание работы и индивидуальные задания.
3. Блок-схемы разработанных алгоритмов (при оформлении отчета в печатном виде рекомендуется использовать Microsoft Visio).
4. Тексты программ на языке C/C++.
5. Результаты тестирования программ.
6. Вывод о выполненной работе.

Пример оформления лабораторной работы

Лабораторная работа №1
студента группы ИТ-11
Петрова Ильи Александровича

Выполнение:_____ Защита:_____

Устройство персонального компьютера

Цель работы: ознакомиться с классификацией и основными характеристиками персональных компьютеров (ПК), компонентами, входящими в их состав; научиться определять основные параметры ПК.

Содержание работы

1. *Ознакомьтесь с теоретическим материалом.*
2. *Занесите в отчет описание устройств ввода информации, входящих в состав вашего компьютера.*
3. *Занесите в отчет описание устройств вывода информации, входящих в состав вашего компьютера.*
4. *Занесите в отчет сведения об установленной операционной системе и конфигурации компьютера. Для их получения щелкните правой кнопкой мыши на объекте Компьютер на рабочем столе или в меню Пуск и выберите команду Свойства.*
5. *Сделайте вывод о производительности компьютера.*

Краткие теоретические сведения

Персональный компьютер …

Ход работы

1. *Прочел страницы …учебника под редакцией…Ознакомился с материалом методического пособия …*
2. *Устройства ввода:*
   - *клавиатура произведена фирмой…, имеет … клавиш, …;*
   - *манипулятор мышь произведен фирмой…, имеет …;*
   - *…*
3. *Устройства вывода:*
   - *монитор произведен фирмой…, диагональ …, разрешение…, … ;*
   - *…*
4. *Подвел указатель мыши к объекту рабочего стола Компьютер и нажал правую кнопку мыши. В появившемся контекстном меню*

выбрал 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 классификацией и основными характеристиками персональных компьютеров (ПК), компонентами, входящими в их состав; научиться определять основные параметры ПК.

КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Персональный компьютер обычно используется в каждый конкретный момент времени только одним пользователем. Причем области применения ПК могут быть самые разные: от научных расчетов до компьютерных игр. Значительно расширяют сферу применения компьютеров современные коммуникационные технологии, позволяя создавать вычислительные сети, существенно превосходящие по своим возможностям отдельные рабочие станции. Немаловажным является и выпуск компьютерной индустрией огромного количества разнообразного периферийного оборудования, которым может комплектоваться ПК для решения различных задач.

В принципе, любой ПК представляет собой набор соединенных между собой сменных компонентов. Однако конструктивное исполнение компьютера может быть различным, в зависимости от его основного назначения. Основным признаком, по которому ПК делятся на два больших класса, это возможность перемещения в рабочем состоянии. Различают стационарные и мобильные компьютеры. К последним относятся ноутбуки, планшеты, карманные ПК, а также устройства мобильной связи, оснащенные функциями компьютера (смартфоны).

Стационарные ПК в свою очередь по исполнению могут быть моноблочными и с раздельной компоновкой. В моноблоках основные элементы компьютера (обычно за исключением средств ввода информации) объединены в одно устройство. Достоинством такого подхода является компактность конструкции, недостатком – трудность модернизации.

Исторически самой первой появилась раздельная схема компоновки персонального компьютера. Она же является и самой распространенной. Несмотря на бóльшую громоздкость такой конструкции, она наиболее приспособлена к дальнейшей модернизации. Далее мы будем рассматривать ПК именно такой компоновки.

Наиболее типичная конфигурация персонального компьютера включает в себя системный блок, а также такие периферийные устройства, как монитор (устройство вывода информации), клавиатура, манипулятор типа «мышь» (устройства ввода информации), которые подключаются к системному блоку. При необходимости, к системному блоку могут подключаться и другие периферийные устройства, которые можно разделить на несколько типов:

- устройства вывода информации. Кроме монитора к таким устройствам относятся принтеры, многофункциональные устройства, плоттеры, колонки, наушники, проекторы и т.д.;
- устройства ввода информации. Кроме клавиатуры и мыши сюда относятся сканеры, игровые манипуляторы, графические планшеты (дигитайзеры), микрофоны, веб-камеры и т.д.;
- устройства хранения информации. К ним относятся разнообразные внешние накопители информации, такие как внешние жесткие диски, *USB* флеш-накопители, карты памяти и картридеры для работы с ними и т.д.
- устройства обмена информацией: модемы, а также различное сетевое оборудование.

Некоторые из периферийных устройств позднее мы рассмотрим более подробно, а пока определим, что может входить в состав такого важного компонента ПК, как системный блок.

Системный блок

По сути, системный блок состоит из корпуса с блоком питания, в котором размещены такие элементы, как материнская плата, жесткий диск («винчестер»), оптический привод, дополнительные платы расширения (видеоадаптер, сетевой адаптер и т.д.). Сзади (частично и спереди) находятся порты (разъемы) для подключения внешних периферийных устройств.

Материнская плата является одним из основных составляющих системного блока и представляет собой прямоугольник многослойного фольгированного текстолита, на котором распаяны микросхемы, дискретные элементы (резисторы, конденсаторы и т.д.), разнообразные разъемы (рис. 1.1). Их выводы соединяются несколькими слоями дорожек из фольги, размещенных на поверхности и внутри платы.

Непосредственно на материнскую плату устанавливаются процессор, постоянное запоминающее устройство (ПЗУ), оперативное запоминающее устройство (ОЗУ), а также необходимый набор плат расширения функциональных возможностей компьютера.

Все современные ПК построены по принципам *шинной архитектуры*. 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 типом (последовательные или параллельные), 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 оперативной памятью и другими высокоскоростными устройствами, а также локальные шины для подключения накопителей информации и периферийных устройств. Шины управляются специализированными устройствами – *контроллерами*.

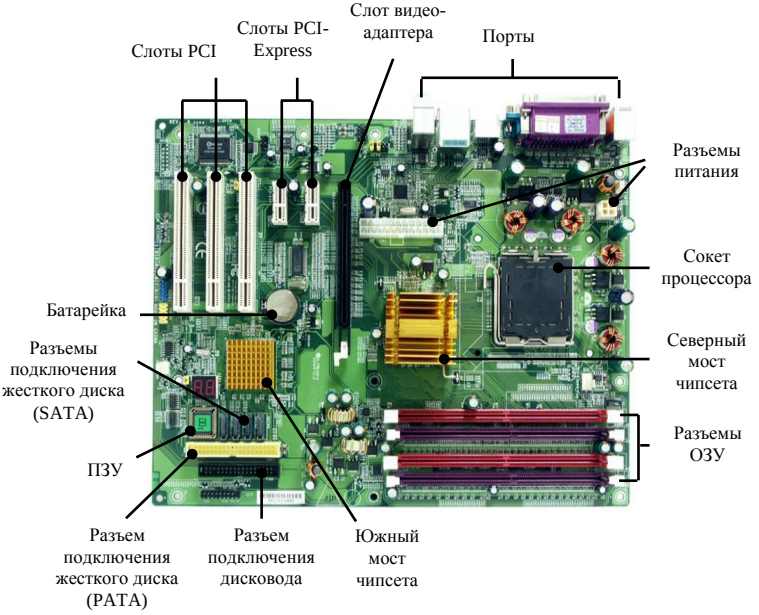


Рис. 1.1. Материнская плата

Основным элементом материнской платы, определяющим ее характеристики, является *чипсет* – набор микросхем системной логики. Назначением чипсета является обеспечение работы процессора, а также управление обменом данными между элементами материнской платы. Конструктивно, как правило, чипсет разделен на две микросхемы:

«северный» и «южный» мосты (на рис. 1.1 на микросхемах установлены радиаторы). Основным элементом северного моста является контроллер памяти, управляющий обменом информацией между процессором, оперативной памятью и локальной шиной видеоадаптера, т.е. наиболее высокоскоростными элементами компьютера. Южный мост отвечает за поддержку низкоскоростного периферийного оборудования: жесткие диски, платы расширения, клавиатура и т.д. Северный и южный мосты связаны между собой отдельной, высокоскоростной шиной.

Рассмотрим более подробно элементы, устанавливаемые на материнской плате или подключаемые к ней.

Процессор (центральный процессор) выполняет арифметические и логические операции, а также управляет всеми устройствами компьютера. При этом процессор может выполнять только ограниченный набор инструкций, и обработка информации любой сложности сводится именно к их выполнению.

Большинство современных процессоров для персональных компьютеров относятся к семействам x86 или x86-64. Первые из них представляют собой 32-разрядные процессоры, вторые – 64-разрядные. Разрядность определяет размер целочисленных данных, обрабатываемых процессором как одно целое, а также максимальный объем адресуемой им оперативной памяти. Все эти процессоры поддерживают выполнение тех же инструкций, что и первый 32-разрядный процессор фирмы Intel i80386. Также современные процессоры поддерживают многие расширенные наборы инструкций (*MMX*, *SSE* и другие), которые ускоряют процесс обработки информации.

Важнейшей характеристикой процессора является тактовая частота – число элементарных операций (тактов), выполняемых за секунду. Тактовая частота измеряется в мегагерцах (МГц) (1 МГц = 1 млн тактов в секунду). Тактовая частота современных процессоров может превышать 3000 МГц. Однако реальное быстродействие процессора зависит не только от тактовой частоты, но и от других факторов: особенностей внутренней архитектуры, частоты системной шины, размера кэш-памяти (см. ниже) и т.д.

Конструктивно процессор представляет собой пластиковый или керамический корпус с выводами для подключения к материнской плате. На корпусе размещается кристалл кремния, содержащий непосредственно *ядро* процессора, которое может содержать до нескольких сотен миллионов транзисторов. Процессор устанавливается в специальный разъем на материнской плате – так называемый *сокет* (см. рис. 1.1). Тип сокета бывает разный в зависимости от модели

процессора и фирмы-производителя. Сверху процессора для охлаждения его кристалла устанавливается охлаждающая система, которая, как правило, состоит из медного или алюминиевого радиатора и обдувающего его вентилятора. Необходимость такой системы вызвана тем, что все современные высокопроизводительные процессоры отличаются достаточно большим энергопотреблением и, следовательно, тепловыделением.

В настоящее время большинство процессоров являются многоядерными. Это означает, что на одном кристалле кремния размещается два, четыре, шесть или более ядер, которые работают практически независимо друг от друга. Такая конструкция процессора позволяет значительно увеличить его быстродействие при одновременном выполнении нескольких программ.

В течение нескольких последних лет наметилась тенденция включения в состав процессора тех устройств, которые ранее размещались на материнской плате. В частности в процессор встраивают контроллер памяти, что увеличивает быстродействие и позволяет размещать на материнской плате только южный мост чипсета. Также в состав процессора часто включают видеоадаптер, который пользователь может использовать или игнорировать по своему усмотрению.

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 (рис. 1.1). Общий объем оперативной памяти может достигать нескольких гигабайт.

В **ПЗУ** хранится базовая система ввода – вывода (*BIOS*), которая состоит из программы тестирования оперативной памяти и периферийного оборудования компьютера, а также программы запуска операционной системы. Кроме осуществления процедуры загрузки компьютера, *BIOS* осуществляет низкоуровневые операции ввода-вывода. Само ПЗУ является энергонезависимой памятью и реализовано в виде микросхемы, которая устанавливается в соответствующий разъем материнской платы на заводе-изготовителе (рис. 1.1). Поскольку информация в ПЗУ является неизменяемой, для хранения некоторых

изменяемых параметров *BIOS* имеется специальная программа настройки (*Setup*), хранящаяся в отдельной микросхеме памяти, питающейся после выключения ПК от батарейки на материнской плате.

Кэш-память – это память с большей, чем у обычного ОЗУ скоростью доступа и предназначенная для хранения наиболее часто используемых данных. 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 или сотен килобайт). Кэш второго уровня (L2) имеет, как правило, существенно больший объем (до нескольких мегабайт), но меньшее быстродействие. Если в процессоре применяется трехуровневое кэширование, то кэш третьего уровня (L3) имеет меньшее, чем L2, быстродействие, но в несколько раз больший объем.

Накопители на жестких магнитных дисках (НЖМД, жесткие диски) относятся к внешней памяти и предназначены для долговременного хранения информации. Они представляют собой малогабаритный пакет из жестких алюминиевых или стеклянных 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.

Для подключения к материнской плате, жесткие диски имеют соответствующий интерфейс: параллельный или последовательный. Параллельный (*Parallel ATA*) имеет пропускную способность до 100 Мбайт/с, последовательный (*Serial ATA*) – до 600 Мбайт/с (в зависимости от версии). Подключение винчестера к соответствующему разъему на материнской плате (см. рис. 1.1) производится с помощью специального интерфейсного кабеля. В настоящее время параллельный интерфейс практически не применяется и на многих материнских платах его разъемы отсутствуют.

Оптический привод предназначен для чтения и записи таких сменных носителей информации, как *оптические (лазерные) диски*. Информация на такие диски может записываться при изготовлении на

заводе (обычно это фильмы, музыка или игры) или непосредственно пользователем ПК с помощью привода. В последнем случае используются специальные носители для однократной или многократной записи. Что касается форматов лазерных дисков, то в настоящее время наиболее распространенными являются следующие:

- Compact Disc (компакт-диски, *CD*). Типичная емкость составляет 700 Мбайт, максимальная – 900 Мбайт. Обычно используются для хранения аудиозаписей или произвольных двоичных данных;
- Digital Video Disc (*DVD*). Имеют более высокую плотность записи. Типичная емкость составляет 4,7 Гбайт, максимальная – 17 Гбайт (двухслойный двухсторонний диск). По своей структуре диски могут содержать видеозаписи, аудиозаписи или произвольные данные пользователя.
- Blu-ray Disc (*BD*). Еще более высокая плотность записи. Типичная емкость однослойного диска составляет 25 Гбайт, двухслойного – 50 Гбайт. Максимальная емкость четырехслойного диска составляет 128 Гбайт.

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

Кроме перечисленных выше устройств, непосредственно на материнской плате можно устанавливать дополнительные платы (карты) расширения, позволяющие расширить функциональность компьютера. Для их подключения на материнской плате имеются специальные разъемы – *слоты* локальной шины *PCI* или *PCI-Express* (см. рис. 1.1). Все платы расширения имеют в нижней части ряд контактов, строго соответствующих слоту, в который они могут устанавливаться. Как правило, на материнскую плату устанавливается минимум одна такая карта – видеоадаптер (видеокарта), используемая для непосредственной генерации сигнала, подаваемого на монитор (см. ниже).

Все современные видеокарты совмещают в себе функции собственно видеоадаптера для вывода двухмерного изображения и видеоускорителя для работы с трехмерной графикой. Видеокарта содержит специальный графический процессор, который выполняет большую часть расчетов для выводимой картинки, тем самым, разгружая центральный процессор. Это особенно актуально в компьютерных играх и приложениях для работы с трехмерной компьютерной графикой. Информация об изображении на экране хранится в видеопамяти – специальных микросхемах видеоадаптера.

Поскольку при работе видеоадаптеры требуют интенсивного обмена данными с оперативной памятью, для их подключения используется

высокоскоростная шина *PCI-Express*, имеющая специальный слот на материнской плате (рис. 1.1).

Также в слоты материнской платы могут устанавливаться звуковая карта (служит для ввода/вывода аудиоинформации), сетевая карта (предназначена для обмена информацией в локальных сетях – *Local Area Network, LAN*), внутренний модем и другие устройства.

В современные материнские платы производитель, как правило, встраивает (интегрирует) дополнительные устройства, чтобы избавить пользователя от необходимости их дальнейшей покупки. Это могут быть видеокарта, сетевая карта, звуковая карта. Причем последние два устройства встраивают практически в сто процентов материнских плат. Однако если пользователя персонального компьютера не устраивают характеристики интегрированных устройств, он может всегда дополнительно установить их аналоги, используя соответствующие интерфейсы материнской платы.

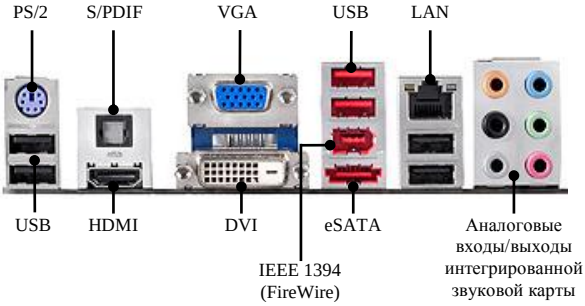


Рис. 1.2. Порты, размещаемые на материнских платах

Наряду с платами расширения, в составе персонального компьютера имеются и внешние периферийные устройства, которые должны быть подключены к системному блоку. Для этого на материнской плате имеются соответствующие порты (рис. 1.2), доступные на задней стороне системного блока. Их набор может отличаться в зависимости от модели и фирмы-производителя, но на любой современной материнской плате имеются следующие порты:

– *USB* (*Universal Serial Bus*) – универсальная последовательная шина, используемая для подключения самых разнообразных периферийных устройств: клавиатуры, мыши, принтера, плоттера, сканера, модема, внешних носителей информации и т.д.

Существуют разные спецификации с пропускной способностью от 12 (*USB* 1.0) до 4800 (*USB* 3.0) Мбит/с;

- *PS/2* – порты, служащие для подключения мыши или клавиатуры. В связи с широким распространением интерфейса *USB*, в настоящее время используются все реже;
- *LAN*-порт – разъем типа *8P8C* встроенного сетевого адаптера;
- аналоговые и иногда цифровые (*S/PDIF*) входы/выходы интегрированной звуковой карты. Позволяют подключать наушники, колонки, микрофоны и другие устройства.

Кроме вышеперечисленных портов, на некоторых материнских платах могут устанавливаться:

- при наличии интегрированной видеокарты интерфейсы для подключения монитора: аналоговый (*Video Graphic Array*, *VGA*) и/или цифровой (*Digital Visual Interface*, *DVI*). В большинстве случаев видеоадаптеры имеют также разъем *HDMI* (*High-Definition Multimedia Interface*), используемый для подключения жидкокристаллических или плазменных телевизионных панелей и позволяющий передавать видеосигнал высокого разрешения вместе со звуковым сопровождением;
- *IEEE 1394* (*FireWire*) – высокоскоростная шина с пропускной способностью до 400 Мбит/с. Может использоваться для подключения к ПК видеокамер, принтеров, сканеров, организации локальных сетей. В настоящее время используется достаточно редко в связи с широким распространением скоростных спецификаций шины *USB* (2.0 и 3.0);
- *eSATA* (*External SATA*) – интерфейс, используемый для обмена информацией с внешними жесткими дисками. В виду того, что разъем используется только для передачи информации, для подачи к жесткому диску питающего напряжения должен использоваться другой источник (например, *USB*-порт). Этот факт способствовал более широкому распространению внешних накопителей с *USB*-интерфейсом, у которых единственный разъем выполняет обе функции.

Порты могут располагаться не только непосредственно на материнской плате, но и на дополнительных картах расширения. Размещение слотов на материнской плате таково, что при установке в них карт расширения их интерфейсная часть также располагается на задней стороне системного блока.

Как уже было сказано, порты используются для подключения к системному блоку внешних периферийных устройств. Рассмотрим некоторые из них.

Устройства ввода информации

Клавиатура компьютера является одним из основных устройств ввода информации и команд на ее обработку. Для ее подключения можно использовать порты *USB* или *PS*/2. Группы клавиш приведены на рис. 1.3.

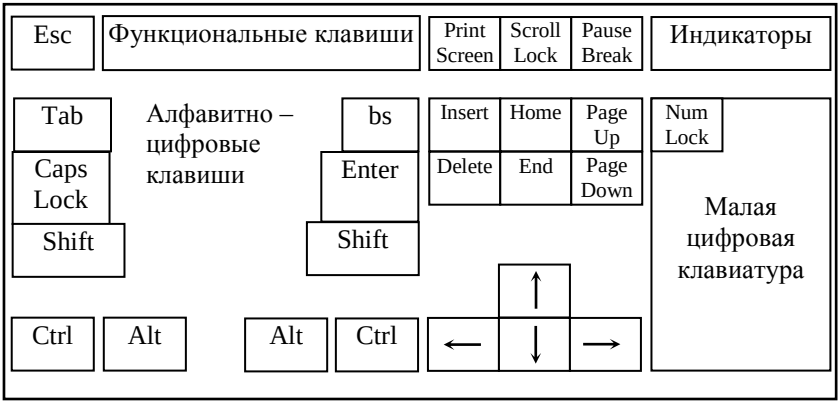


Рис. 1.3. Клавиатура персонального компьютера

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 на русскую раскладку клавиатуры и обратно осуществляется нажатием комбинации клавиш **Ctrl+Shift** или **Alt+Shift**, в зависимости от настроек операционной системы.

Позицию на экране, в которую будет вводиться символ, указывает ***курсор***, имеющий вид вертикальной мерцающей черты.

Клавиши со стрелками предназначены для перемещения курсора в указанном направлении.

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.

Scroll Lock переключает клавиши со стрелками в такой режим, при котором они перемещают вниз-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.

Функциональные клавиши свое назначение меняют в зависимости от текущей прикладной программы. Клавиша **F1** почти всегда служит для вызова справочной системы.

Esc (escape) используется для прекращения выполнения команды.

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 копирует содержимое экрана и помещает его в виде графического изображения (скриншота) в буфер обмена (область памяти, используемая для обмена информацией между разными приложениями).

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

Delete удаляет символ, находящийся справа от курсора, со сдвигом остального текста на позицию влево.

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.

Мышь является устройством ввода информации, используемым в основном в программах с графическим интерфейсом.

В настоящее время наиболее распространены 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.

Для подключения к системному блоку используются интерфейсы *USB* (преобладают) или *PS*/2. По технологии подключения мыши делятся на проводные и беспроводные. Последние работают в паре со специальным приемником сигнала, устанавливаемым в *USB*-порт.

Управление приложением производится путем перемещения курсора мыши и нажатием на ее кнопки, которых в самом минимальном варианте две: левая и правая. Если имеются другие кнопки, то обычно их функции могут настраиваться через специальное программное обеспечение. Также на мыши обычно расположено колесо для вертикальной прокрутки содержимого окна (скроллинга).

Игровые манипуляторы – устройства ввода информации, используемые преимущественно в компьютерных играх. Они бывают разных конструкций и типов, которые используются в играх определенной направленности: игровые пульты (геймпады) в аркадных играх, джойстики и штурвалы в авиасимуляторах, автомобильные и мотоциклетные рули в гоночных симуляторах и т.д. Сейчас активно развиваются технологии «захвата» движений человека, основанные на использовании инфракрасных датчиков и распознавании образов.

Сканер – устройство, предназначенное для преобразования изображения, размещенного на бумаге или пленке в цифровой формат. Позволяют во много раз увеличить скорость ввода информации. Наиболее распространены планшетные сканеры, которые часто используются совместно со специальным программным обеспечением распознавания в изображении рукописного или печатного текста.

Устройства вывода информации

Монитор является наиболее распространенным устройством вывода информации. На его экран выводится большинство результатов работы компьютера. Монитор подключается к видеоадаптеру, расположенному в системном блоке, с использованием интерфейсов *VGA* или *DVI*.

Качество изображения видеомонитора определяется его разрешающей способностью – количеством точек (пикселей), изображаемых на экране по горизонтали и вертикали, а также минимальным размером точки. Конкретные варианты разрешения монитора зависят от его формата, то есть от соотношения длин сторон:

стандартный (4:3) или широкоформатный (16:9, 16:10). Количество цветов, отображаемых современным монитором, достигает 16 миллионов.

В составе персональных компьютеров, как правило, используются мониторы двух типов: на базе электронно-лучевой трубки (*CRT*) и жидкокристаллические (*LCD*). Второй тип, несмотря на наличие некоторых недостатков, в настоящее время доминирует и практически полностью вытеснил с рынка *CRT*-мониторы.

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 пикселей.

Принтер служит для вывода информации на бумагу. Различают следующие основные типы 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).

Разновидностью принтеров являются многофункциональные устройства (МФУ). Они представляют собой комбинацию планшетного сканера и лазерного или струйного принтера в одном корпусе. Могут использоваться и как копиры, и как отдельные принтер и сканер.

Плоттер (графопостроитель) выводит информацию на бумагу большого формата (A0–A2). Как правило, плоттеры используют в системах автоматизированного проектирования (САПР).

Устройства обмена информацией

Модем представляет собой плату системного блока или отдельное устройство (подключаемое, например, через *USB*-порт), с помощью которого цифровой сигнал компьютера преобразуется в телефонный электромагнитный сигнал и наоборот. Это позволяет, используя обычную телефонную линию, обмениваться информацией с другими компьютерами, объединенными в локальную сеть или подключенными к глобальной сети Интернет. Основной характеристикой модема является скорость передачи данных, которая измеряется в бит/с.

В данный момент наиболее распространенными являются модемы, поддерживающие технологию *xDSL* (*Digital Subscriber Line* – цифровая абонентская линия). Такой модем подключается к телефонной линии и передает данные в цифровом виде. Поскольку линия предназначена прежде всего для передачи аналоговых сигналов телефонной связи, то необходимо производить ее частотное уплотнение. 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 24 Мбит/с, отправки – 3,5 Мбит/с. Появилась также более скоростная технология *VDSL* (*Very-high data rate DSL*), в которой скорость получения достигает 65 Мбит/с, отправки – 35 Мбит/с.

Также существуют модемы, не требующие подключения к телефонной линии. Это беспроводные модемы, использующие мобильную связь (*GSM*). На ПК они устанавливаются, как правило, в *USB*-порт. В мобильных сетях третьего поколения (*3G*) скорость получения данных модемом может достигать 7 Мбит/с.

Сетевое оборудование служит для поддержки работы проводных и беспроводных (*Wi-Fi*) вычислительных сетей, позволяющих организовать передачу данных между компьютерами. По масштабу охвата сети делятся на ряд классов, например: локальные, городские, глобальные. Как правило, ПК подключаются к локальным проводным сетям (*LAN*), используя для этого уже упомянутый выше сетевой адаптер (сетевую карту). В качестве среды передачи информации обычно используются специальные медные кабели (например, витая пара, в которой имеется несколько свитых пар изолированных медных проводников). Для соединения между собой нескольких компьютеров

или сегментов сети применяются коммутаторы (*switch*), а нескольких сетей – маршрутизаторы (*router*). В настоящее время наиболее распространены устройства, поддерживающие технологии передачи данных *Fast Ethernet* (со скоростью до 100 Мбит/с) и *Gigabit Ethernet* (до 1 Гбит/с). Максимальная длина сегмента в таких сетях – 100 м.

Беспроводные сети позволяют организовать передачу информации без использования кабельной проводки с помощью радиоволн в СВЧ-диапазоне. Такая технология особенно предпочтительна для мобильных устройств, а также при эксплуатации ПК в заданиях, не оборудованных проводными коммуникациями. Для подключения к сети в ПК должен устанавливаться специальный сетевой адаптер *Wi-Fi*, который может быть внешним (подключается в *USB*-порт) или внутренним (*PCI*-слот). В ноутбуках и других мобильных устройствах такие адаптеры являются встроенными. Во время работы адаптер Wi-Fi устанавливает связь со специальным устройством, которое в свою очередь подключено к проводной локальной сети. Такими устройствами могут быть так называемые точки доступа, коммутаторы и маршрутизаторы *Wi-Fi*. Скорость передачи данных зависит от используемого стандарта и может достигать 300 Мбит/с. Дальность покрытия обычно не превышает нескольких сотен метров.

Устройства хранения информации

Внешние жесткие диски среди подобного типа устройств обладают наибольшей емкостью (до нескольких терабайт), но при этом и самыми большими габаритами и массой. В разное время для их подключения использовали различные интерфейсы: *IEEE* 1394 (FireWire), *USB*, *eSATA*. В данный момент подавляющее большинство выпускаемых внешних жестких дисков имеют интерфейс *USB* (спецификации 3.0);

USB флеш-накопители подключаются к порту *USB* и используют в качестве носителя информации флеш-память. По сути она представляет собой электрически перепрограммируемое ПЗУ. Флеш-накопители характеризуются компактностью, легкостью и достаточно большой емкостью (до нескольких десятков Гбайт), хотя и уступают жестким дискам и по этому показателю и по скорости передачи данных.

Карты памяти применяются преимущественно в цифровой фото, аудио, видеоаппаратуре, в устройствах мобильной связи и т.д. Здесь также используется флеш-память. Для подключения карты памяти к ПК

необходимо специальное устройство сопряжения – картридер. Он может быть встроенным (например, в ноутбук или корпус системного блока) или внешним, подключаемым через *USB*-порт. Существуют разные форматы карт памяти, различающиеся габаритами, емкостью, спецификациями подключения к устройству и т.д. Наибольшее распространение получили карты формата *Secure Digital* (*SD*) и их более миниатюрный вариант – *microSD*. Карты объемом более 2 Гбайт и до 32 Гбайт включительно относятся к спецификации *SDHC* (*Secure Digital High Capacity*), а свыше 32 Гбайт к спецификации *SDXC* (*Secure Digital eXtended Capacity*). По скорости записи информации *SD*-карты делятся на ряд классов: 2, 4, 6, 10, 16. Число это означает минимально допустимую скорость записи информации на карту в Мбайт/с.

СОДЕРЖАНИЕ РАБОТЫ

1. Ознакомьтесь с теоретическим материалом.
2. Занесите в отчет описание устройств ввода информации, входящих в состав вашего компьютера.
3. Занесите в отчет описание устройств вывода информации, входящих в состав вашего компьютера.
4. Занесите в отчет сведения об установленной операционной системе и конфигурации компьютера. Для их получения щелкните правой кнопкой мыши на объекте *Компьютер* на рабочем столе или в меню *Пуск* и выберите команду *Свойства*.
5. Сделайте вывод о производительности компьютера.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Назовите основные элементы системного блока.
2. Основные принципы шинной архитектуры ЭВМ.
3. Какие устройства устанавливаются непосредственно на материнскую плату?
4. Каковы назначение и основные характеристики процессора?
5. Назначение чипсета материнской платы.
6. Назовите виды памяти ПК.
7. Назначение и принципы организации кэш-памяти.
8. Какие внешние носители информации вы знаете?
9. Охарактеризуйте группы клавиш клавиатуры.
10. Назначение и основные характеристики видеоадаптеров.
11. Перечислите основные характеристики мониторов.
12. Какие устройства обмена информацией вам известны?

ЛАБОРАТОРНАЯ РАБОТА № 2

ОПЕРАЦИОННЫЕ СИСТЕМЫ

Цель работы: ознакомиться с понятиями операционной системы, файла, каталога; получить представление об организации файловой системы; приобрести навыки выполнения операций с файлами и папками встроенными средствами Windows и с помощью дополнительно устанавливаемых файловых менеджеров.

КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Операционная система (ОС) – это комплекс программ, который обеспечивает диалог компьютера с пользователем, управляет всеми элементами компьютера, запускает другие (прикладные) программы на выполнение. Общим свойством всех видов ОС является обеспечение взаимодействия (*интерфейса*).

Интерфейс – это система правил, определяющих взаимодействие:

- пользователя и программно-аппаратных средств компьютера *(интерфейс пользователя);*
- программ и аппаратного обеспечения *(аппаратно-программный интерфейс);*
- между разными видами программного обеспечения *(программный интерфейс).*

Виды интерфейсов пользователя

По реализации интерфейса пользователя различают *неграфические* и *графические операционные системы.* Неграфические операционные системы реализуют *интерфейс командной строки.* Основным устройством управления в данном случае является клавиатура. Управляющие команды вводят в поле командной строки. Исполнение команды начинается после нажатия клавиши **Enter**. Для компьютеров платформы *IBM PC* в свое время было создано целое семейство неграфических операционных систем под общим названием *MS DOS* (версии от *MS DOS* 1.0 до *MS DOS* 6.22).

Графические операционные системы реализуют более сложный тип интерфейса, в котором в качестве органа управления кроме клавиатуры может использоваться мышь или другое устройство позиционирования. Работа с графической операционной системой основана на

взаимодействии активных и пассивных экранных элементов управления. В качестве активного элемента управления выступает *указатель мыши* – графический объект, перемещение которого на экране синхронизировано с перемещением мыши. В качестве пассивных элементов управления выступают графические *элементы управления приложений* (экранные кнопки, значки, переключатели, флажки, раскрывающиеся списки, строки меню и многие другие).

Кроме того, графические ОС являются, как правило, *многозадачными*. Это подразумевает возможность одновременного выполнения нескольких приложений и динамического обмена данными между ними.

Одними из наиболее популярных графических ОС для персональных компьютеров являются системы *Microsoft Windows* различных версий.

Файлы, каталоги, диски

Любая информация во внешних запоминающих устройствах хранится в файлах. Файл – это поименованная область диска или другого внешнего носителя информации. В файлах могут быть размещены некоторые данные, тексты, программы. Имя файла состоит из двух частей – корневого имени и расширения, которые отделяются друг от друга точкой. При этом расширение в отличие от корневого имени не является обязательным и указывает тип файла.

В разных операционных системах может различаться максимальная длина имени файла, а также набор допустимых символов, из которых оно может состоять. В *MS DOS* корневое имя может содержать до восьми символов, а расширение до трех. Допускаются прописные и строчные латинские буквы, цифры и символы:

– _ $ # & @ ! % ( ) { } ‘ ^ ˜

В операционных системах Windows правила именования файлов являются менее жесткими. Максимальная длина имени файла составляет 255 символов. Можно использовать кириллицу, пробелы, точки, запятые и другие символы, запрещенные в *MS DOS*. Недопустимыми в Windows являются только символы:

\ / : * ? " < > |

Файлы объединяются по каким-либо признакам в каталоги (синонимы каталогов – директории, папки). В любом каталоге могут быть вложенные каталоги или подкаталоги. Имена файлов и подкаталогов в одном каталоге должны быть уникальными (т.е. разными, несовпадающими). В разных каталогах могут быть файлы и подкаталоги с одинаковыми именами.

Каталоги, как и файлы, хранятся на дисках. Диски обозначаются латинскими буквами *A:*, *B:*, *C:*, *D:* и т.д. Буквами *A:* и *B:* обозначаются гибкие диски (дисководы для работы с гибкими дисками). Даже, если дисководы отсутствуют (как это характерно для большинства современных компьютеров), эти буквы остаются зарезервированными и не могут использоваться для обозначения других носителей информации. Диски *C:*, *D:* и т.д. представляют собой, как правило, участки (разделы) жесткого диска («винчестера»). Поэтому их обычно называют *логическими* дисками для того, чтобы подчеркнуть различие с физическим устройством. Оптическим приводам обычно назначаются первые из свободных букв после именования разделов жесткого диска. Дополнительно могут подключаться и другие, внешние носители информации (например, устройства флеш-памяти или внешние жесткие диски). В системе они также представляются в виде логических дисков, имеющих собственные имена.

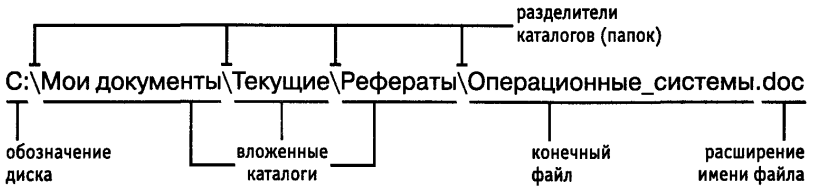


Рис. 2.1. Пример полного имени файла.

Структура расположения каталогов на диске – иерархическая, древовидная. На каждом диске на вершине этой иерархии располагается корневой каталог, обозначаемый символом «\» (обратный слэш). В корневом каталоге располагаются файлы и каталоги 1-го уровня, в каталогах 1-го уровня – каталоги и файлы 2-го уровня и т.д. Для доступа к файлу операционной системе необходимо его *полное имя*, которое является комбинацией имени диска, пути к файлу (т.е. перечисления имен каталогов, в которые последовательно входит файл, разделенные обратным слэшем) и собственно имени файла (рис. 2.1). Если файл находится на текущем диске и/или в текущем каталоге, то имя диска и/или путь к файлу можно не указывать.

Шаблон имени

Многие команды могут применяться не только к одному, но и к нескольким файлам. Для этого необходимо либо предварительно выделить группу файлов в каком-либо каталоге, либо непосредственно указать их имена в составе команды. Если имена имеют сходную (подобную) структуру, то можно использовать для их обозначения шаблон (маску) – обобщенное имя файла, определяющее множество объектов с похожими именами. В отличие от стандартных имен, шаблон содержит некоторое количество специальных подстановочных символов: звездочка (*), заменяющая любое количество символов и (или) вопросительный знак (?), заменяющий один символ. Подстановочные символы могут быть как в корневом имени, так и в расширении и действуют в той части полного имени, где они указаны.

Примеры выделения по шаблону: ***.*** – все файлы независимо от имени и расширения; ***.txt** – все файлы с расширением *.txt*; **a???.*** – все файлы, имена которых начинаются на букву «а», состоят не более чем из 4 символов с произвольным расширением.

Очень часто шаблоны применяются при поиске файлов, когда известна только часть имени и/или расширение. Как правило, требуется в заданном каталоге найти все файлы, имена которых подходят под указанный шаблон. После выполнения процедуры поиска пользователь может выбрать искомый файл из предложенного списка.

Организация файловой системы

Все современные операционные системы обеспечивают создание и функционирование файловых систем, предназначенных для организации хранения данных на дисках и других устройствах внешней памяти и обеспечения доступа к ним. Рассмотрим общие принципы организации файловой системы на жестком диске или его разделе.

Наименьшей физической единицей хранения данных на диске является сектор. Размер сектора равен 512 байт. Как правило, операционная система не в состоянии обеспечить адресацию к каждому отдельному сектору. В связи с этим группы секторов условно объединяются в *кластеры*. Кластер является *наименьшей единицей адресации к данным* с точки зрения операционной системы. Размер кластера, в отличие от размера сектора, не фиксирован и может зависеть от типа файловой системы и пожеланий пользователя, производящего форматирование раздела.

Для хранения файла может быть выделено целое число кластеров. Даже если файл имеет размер несколько байт, а кластер несколько килобайт (как правило, 4, 8, 16 или 32 Кбайт), то все равно другой файл не сможет занимать какую-либо его часть. То есть, если в разделе хранится большое количество маленьких файлов, то мы как будто «теряем» часть дискового пространства. А слишком маленький размер кластера будет приводить к снижению скорости файловых операций.

Принцип организации файловых систем, поддерживаемых ОС Windows – табличный. Первые версии поддерживали только файловые системы семейства *FAT*: *FAT12* (применялась для дискет), *FAT16* (применялась для дисков объемом не более 2 Гбайт), *FAT32* (появилась начиная с версии Windows 95 *OSR2*). Главный элемент организации хранения данных в этих файловых системах указан в самом названии. *FAT*-таблица (File Allocation Table, таблица размещения файлов), расположенная в системной области диска, содержит данные о размещении того или иного файла. Поскольку повреждение таблицы приводит к невозможности воспользоваться хранящимися данными, она существует в двух экземплярах, идентичность которых регулярно контролируется средствами операционной системы.

Число, указанное в названии файловой системы, означает количество двоичных разрядов, отводимое для хранения в *FAT*-таблице номера кластера. По сути, в таблице для каждого кластера имеется отдельная запись, показывающая его текущее состояние. В связи с этим, имеется очевидное ограничение на размер логического диска. К примеру, в *FAT16* при максимальном количестве кластеров 65 524 и размере кластера 32 Кбайт объем логического диска составит 2 Гбайта. Для *FAT32* с максимальным количеством кластеров 268 435 445 теоретический предел составляет уже 8 терабайт. То есть для дисков, чей объем составляет десятки или даже сотни Гбайт, размер кластера будет существенно меньше, чем в *FAT16*.

Однако на практике, система *FAT32* редко используется для логических дисков, объемы которых превышают 32 Гбайт. Это связано с тем, что сильно вырастает размер таблицы размещения файлов и может значительно замедлиться выполнение файловых операций. Кроме того, есть ограничение на максимальный размер файла – 4 Гб, что является серьезным недостатком этой системы. Поэтому, начиная с версии Windows NT, в качестве основной файловой системы используется *NTFS* – New Technology File System, файловая система новой технологии. Эта система, помимо всего, позволяет организовать многопользовательский режим работы и при необходимости

устанавливать ограничения на доступ к файлам и папкам или используемый объем дискового пространства для разных групп пользователей.

NTFS хранит информацию о файлах в главной файловой таблице (Master File Table, *MFT*), каждая строка которой (размером около 1 Кбайта) содержит сведения об одном файле. Первые 16 записей таблицы описывают так называемые *метафайлы*, необходимые для работы файловой системы. Это, к примеру, файл журнала системы, записи о правах пользователей, битовая карта свободного пространства и т.д. Данные о метафайлах дублируются ровно посередине диска. Под остальную часть *MFT* изначально резервируется 12,5% всего дискового пространства. При заполнении диска зона таблицы может сокращаться, а в дальнейшем снова расширяться. Максимальный размер логического диска в системе составляет около 16 эксабайт (2^{64} байт). Максимальный размер файла – 16 терабайт. При этом размер раздела не оказывает существенного влияния на скорость работы системы. Единственное, что может привести к существенному замедлению файловых операций, так это почти полное заполнение раздела, когда системе приходится фрагментировать *MFT* и распределять ее части по диску.

Для оптических дисков операционные системы Windows поддерживают такие файловые системы, как *ISO 9660, UDF*. Носители информации, использующие флеш-память, могут быть отформатированы в файловых системах *FAT32, NTFS, exFAT* (вариант, специально разработанный Microsoft для флеш-накопителей). Однако надо учитывать, что многие бытовые устройства, поддерживающие работу с такими накопителями, распознают только систему *FAT32*.

Интерфейс командной строки Windows

В таких операционных системах, как *MS DOS* команды, вводимые с клавиатуры, были единственным средством диалога пользователя с компьютером. В дальнейшем в графических операционных системах появились намного более развитые средства интерфейса и роль команд, вводимых с клавиатуры, была сведена к минимуму. Тем не менее, в Windows сохранилась возможность применения интерфейса командной строки. Большое количество команд дублирует функции, выполняемые системой с помощью графического интерфейса. Кроме того, некоторые программы не имеют графического интерфейса и работа с ними возможна только через командную строку. Для этого используется интерпретатор команд **Cmd.exe**. Запустить его можно выполнив команду *Пуск* | *Все программы* | *Стандартные* | *Командная строка*.

Когда интерпретатор готов к приему команд, на экране появляется *приглашение*. Приглашение, как правило, содержит информацию о текущем логическом диске и текущем каталоге, например:

A:\> – диск *A:*, корневой каталог.

C:\Windows> – диск *C:*, каталог 1-го уровня *\Windows*.

D:\BP\BIN> – диск *D:*, каталог 2-го уровня *\BP\BIN*.

Пользователь вводит команду после приглашения прописными или строчными латинскими буквами и подтверждает ее нажатием клавиши **Enter**. В случае возникновения ошибки на экран выводится соответствующее сообщение. После обработки введенной команды вновь появляется приглашение, что свидетельствует о готовности продолжения работы.

При вводе команды указывается ее имя, после которого при необходимости указываются параметры, отделяемые пробелом. Наиболее часто в качестве параметра команды указывается имя файла, которое указывается в полном формате с указанием имени диска и каталога. Если имя диска и путь к файлу не указаны, то он ищется в текущем каталоге, а затем, если не найден, в списке каталогов, заданных командой `Path`. В тех случаях, когда одна команда должна выполняться над группой файлов, используют шаблон с подстановочными символами «*» или «?» (см. выше **шаблон имени**). Также в некоторых командах допускается использовать вместо имени файла зарезервированное имя устройства ввода или вывода информации. Например:

CON – консоль (при вводе – клавиатура, при выводе – экран монитора).

PRN – принтер.

NUL – «пустое» устройство, для которого игнорируются все операции ввода-вывода.

Рассмотрим несколько наиболее часто используемых команд. Большинство из них могут иметь дополнительные ключи, начинающиеся с символа «/» (слэш). Здесь будут приведены только некоторые из ключей. Для получения полной информации можно выполнить команду:

`help` *имя_команды*

Некоторые команды:

1. **Запуск программы (исполняемого файла) на выполнение**. Для запуска в командной строке необходимо набрать имя файла и нажать **Enter**. Кроме исполняемых файлов (с расширением *.com* или *.exe*) можно запустить командный файл с расширением *.bat*.

При наборе имени файла расширение указывать необязательно. Например:

```
c:\autoexec
notepad.exe
```

2. **Смена текущего диска**. Для этого необходимо набрать имя диска, который должен стать текущим и нажать **Enter**. Например:
   `A:` – переход на дисковод *A:*.
   `d:` – переход на диск *D:*.
3. **Смена текущего каталога (команда cd).** Формат команды:
   `cd [/d] [диск:]путь`

   Здесь и далее при описании синтаксиса команды в квадратных скобках указываются необязательные параметры. Примеры:

   `cd \ –` переход в корневой каталог текущего диска.

   `cd /d d:\bp –` переход в подкаталог *bp* корневого каталога диска *D:* (ключ */d* обеспечивает одновременную смену и диска и каталога).

   `cd system –` переход в каталог *system* – подкаталог текущего каталога.

   `cd .. –` переход в надкаталог.
4. **Создание каталога (команда md).** Формат команды:
   `md [диск:]путь`

   Примеры:

   `md student –` создание подкаталога *student* в текущем каталоге.

   `md e:\work –` создание подкаталога *work* в корневом каталоге диска *E:*.
5. **Удаление каталога (команда rd).** По умолчанию можно удалять *только пустые каталоги*. Формат команды:
   `rd [ключи] [диск:]путь`

   Примеры:

   `rd student –` удаление подкаталога *student* в текущем каталоге.

   `rd /s e:\work –` удаление подкаталога *work* в корневом каталоге диска *E:* вместе с его подкаталогами и вложенными файлами.
6. **Просмотр каталога (команда dir).** Формат команды:
   `dir [диск:][путь\][имя_файла][ключи]`

   Если имя файла не задано, то выводится все оглавление каталога, иначе выводятся только сведения об указанном файле или группе файлов. Примеры:

   `dir c:\` – просмотр корневого каталога диска *C:*.

`dir \windows` – просмотр подкаталога *windows* корневого каталога текущего диска.

`dir /p` – просмотр текущего каталога в режиме поэкранного вывода.

7. **Копирование файла (команда copy).** Формат команды:

`copy` *имя_файла имя_файла*

или

`copy` *имя_файла [имя_каталога]*

Из каталога, указанного в первом параметре команды, файлы копируются в каталог, определяемый вторым параметром. Если во втором параметре имя файла отсутствует, имена копируемых файлов не изменяются. Символы «*» и «?» в имени файла во втором параметре указывают, что соответствующие символы в именах копируемых файлов не изменятся. Примеры:

`copy d:\f1.txt c:\dos\f2.txt` – копирование файла *f1.txt* из корневого каталога диска *D:* в подкаталог *dos* корневого каталога диска *C:* с изменением имени на *f2.txt.*

`copy c:\*.* d:\bp` – копирование всех файлов из корневого каталога диска *C:* в подкаталог *bp* корневого каталога диска *D:*.

`copy e:\f1.txt` – копирование файла *f1.txt* из корневого каталога диска *E:* в текущий каталог.

`copy \f1.txt system` – копирование файла *f1.txt* из корневого каталога текущего диска в подкаталог *system* текущего каталога.

Команда `copy` может применяться не только для копирования файлов из одного каталога в другой. Наряду с именами файлов в качестве параметров команды можно использовать имена устройств ввода-вывода информации. Например:

`copy con 1.txt` – создание текстового файла *1.txt* в текущем каталоге. После выполнения команды необходимо набрать строки текстового файла. Для прекращения ввода текста необходимо нажать комбинацию клавиш **Ctrl+Z**, а затем клавишу **Enter**.

Еще одной возможностью команды `copy` является объединение содержимого нескольких файлов в один файл. Формат команды в этом случае выглядит следующим образом:

`copy` *файл_1+файл_2+…+файл_n результирующий_файл*

Например:

`copy 1.txt+d:\2.txt c:\result.txt` – объединение файлов *1.txt* из текущего каталога и файла *2.txt* из корневого каталога диска *D:* в файл *result.txt*, расположенный в корневом каталоге диска *C:*.

8. **Переименование файла (команда ren).** Формат команды:
   `ren` *[диск:][путь\]имя_файла имя_файла*
   Первый параметр задает имя файла, подлежащего переименованию, второй – новое имя. Примеры:
   `ren c:\1.txt 2.doc` – переименование файла *1.txt* из корневого каталога диска *C:* в файл *2.doc*.
   `ren *.txt *.doc` – переименование файлов с расширением *.txt* из текущего каталога в файлы с расширением *.doc*.
9. **Удаление файла (команда del).** Формат команды:
   `del` *[диск:][путь\]имя_файла*
   Примеры:
   `del e:\f.txt` – удаление файла *f.txt* из корневого каталога диска *E:*.
   `del *.txt` – удаление файлов с расширением *.txt* из текущего каталога.
10. **Просмотр текстового файла (команда type).** Формат команды:
    `type` *[диск:][путь\]имя_файла*
    Примеры:
    `type d:\f1.txt` – просмотр текстового файла *f1.txt* из корневого каталога диска *D:*.
    `type f2.txt` – просмотр текстового файла *f2.txt* из текущего каталога.

Графический интерфейс Windows

Главным отличием, с точки зрения пользователя, операционных систем Windows от их предшественниц *MS DOS* является наличие полноценного графического интерфейса, который больше ориентирован на работу с координатными устройствами ввода, среди которых основным является мышь. Интерфейс разных версий может несколько отличаться друг от друга. Далее в этом разделе будут рассмотрены основные элементы интерфейса на примере версии Windows 7.

Данная версия характеризуется поддержкой интерфейса Aero (начиная с редакции *Домашняя расширенная*), использующего анимацию и различные визуальные эффекты. Название представляет собой аббревиатуру английских слов: Authentic, Energetic, Reflective, Open (подлинный, энергичный, отражающий и открытый). Интерфейс Aero объединяет в себе целый ряд инструментов и технологий: Aero Glass, Aero Peek, Aero Shake, Aero Snap, Windows Flip, Windows Flip 3D и других. Подробнее некоторые их этих функций будут рассмотрены ниже.

В Windows работа пользователя в основном производится в *окнах*. Окно – это прямоугольная область экрана, в которой отображается некоторая информация. Одно из окон (хотя и несколько специфичное) отображается сразу после загрузки операционной системы. Это так называемый *рабочий стол*. В нижней части его расположена полоса, называемая *панель задач*. Слева на ней расположена кнопка *Пуск*, вызывающая ***главное меню*** Windows (рис. 2.1).

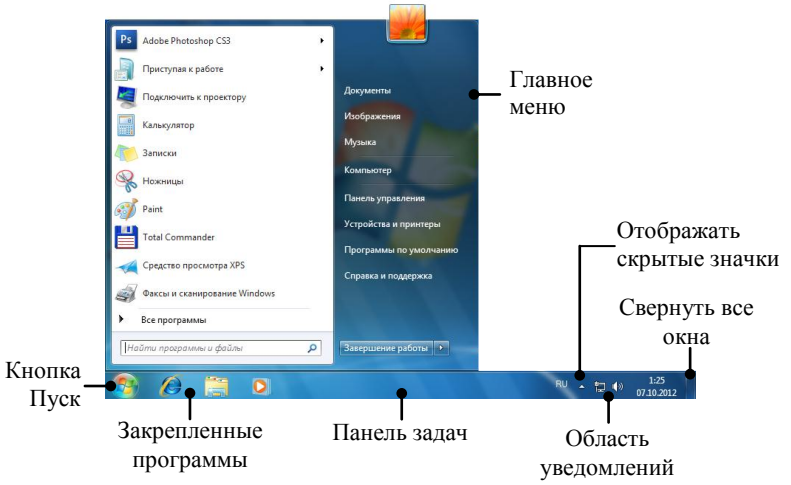


Рис. 2.1. Панель задач и главное меню Windows 7

В главном меню слева находится список приложений, которые использовались в последнее время или которые могут, по мнению разработчиков, понадобиться пользователю. Полный перечень установленных приложений можно увидеть после щелчка по пункту *Все программы*. Под ним находится поле поиска, позволяющее найти нужные пользователю файлы или папки. В правой части главного меню находятся ссылки для доступа к библиотекам пользователя (*Документы*, *Изображения*, *Музыка*), запуска окна дисков и папок Проводника (*Компьютер*), отображения средств настройки параметров Windows (*Панель управления* и другие) и справочной системы. Внизу находится кнопка, позволяющая завершить работу системы или сеанс ее текущего пользователя, а также перезагрузить компьютер.

Кроме главного, в Windows существуют и другие виды меню: контекстное (или всплывающее) и выпадающее. ***Контекстное меню*** появляется при щелчке правой кнопкой мыши. Его команды относятся к тому объекту, на котором находился при щелчке курсор мыши.

Выпадающее меню обычно представляется в виде строки меню окна. Перечень и состав пунктов меню зависит от приложения, в котором работает пользователь.

Правее кнопки *Пуск* располагаются закрепленные на панели задач значки (пиктограммы) наиболее часто используемых приложений (рис. 2.1). После запуска такого приложения значок преобразуется в кнопку окна, расположенную на том же месте. По умолчанию на панели задач закреплены значки браузера Internet Explorer, Проводника и проигрывателя Windows Media. Закрепить или изъять другую программу можно с помощью контекстного меню кнопки окна на панели задач.

Находящаяся в правой части панели задач *область уведомлений* содержит часы, указатель языка, а также значки, отображающие состояние и предоставляющие доступ к настройкам некоторых служб и приложений (рис. 2.1). Некоторые значки могут быть скрыты. Для их отображения можно щелкнуть по кнопке с треугольником (*Отображать скрытые значки*). Сформировать набор значков, отображаемых постоянно можно, выбрав в появившемся всплывающем окне команду *Настроить*.

Основное пространство панели задач используется для размещения кнопок открытых окон. По умолчанию включен режим группировки окон, когда для нескольких окон одного и того же приложения на панели отображается только одна кнопка с его значком. Функция Aero Peek позволяет при наведении курсора на кнопку увидеть миниатюры открытых окон (рис. 2.2). При наведении на миниатюру на ней появляется кнопка закрытия, а само окно временно восстанавливается. Активным окно становится после щелчка левой кнопкой мыши по миниатюре. Если нужно сразу сделать активным то из окон приложения, которое было свернуто последним, то производится щелчок по кнопке на панели задач при нажатой клавише **Ctrl**.

Рис. 2.2. Переключение между открытыми окнами с помощью технологии Aero Peek

Переключение между окнами можно осуществить и с помощью клавиатуры. Технология Windows Flip позволяет при использовании комбинации **Alt+Tab** увидеть такие же миниатюры открытых окон. Причем, удерживая клавишу **Alt** можно осуществлять переключение не только последовательным нажатием **Tab**, но и с помощью курсора мыши. Другим вариантом является использование технологии трехмерной прокрутки открытых окон Windows Flip 3D. Для этого совместно с клавишей **Tab** используется клавиша с логотипом Microsoft (⊞) – **Win**. Эта же клавиша в комбинации с цифрами позволит быстро запустить закрепленные на панели задач программы (например, **Win+1**, **Win+2** и т.д.).

Технология Aero Peek позволяет также временно сделать все открытые окна прозрачными, чтобы увидеть рабочий стол. Для этого курсор мыши нужно навести на кнопку *Свернуть все окна* в области уведомлений (или нажать комбинацию **Win+Пробел**). Щелчок по кнопке (**Win+D**) приведет к сворачиванию окон на панель задач.

На рабочем столе размещаются значки объектов и ярлыков, под каждым из которых находится поясняющая надпись. Объекты в Windows это программы, папки, документы. Ярлык – это значок быстрого доступа к какому-либо объекту. Признаком ярлыка служит стрелочка в левом нижнем углу значка.

Набор пиктограмм на рабочем столе определяет пользователь, создавая ярлыки тех программ, которые используются достаточно часто. Как правило, на рабочем столе находятся пиктограммы *Компьютер* и *Корзина*. Они являются элементами *Проводника* – встроенного в Windows файлового менеджера, предоставляющего графический интерфейс для доступа к файловой системе. Специальная папка Компьютер содержит значки имеющихся логических дисков и папок, также предоставляет доступ к общим ресурсам локальной сети. Корзина содержит перечень всех удаленных файлов за время, прошедшее после последней чистки корзины, и такие файлы могут быть восстановлены.

Если осуществить двойной щелчок по какому-нибудь ярлыку или объекту, то перед нами открывается соответствующее окно. В Windows существует несколько видов окон:

- *окна дисков и папок* – позволяют работать с Проводником Windows и получать доступ к файловой системе, а также к общим ресурсам в локальной сети (рис. 2.3);
- *окна приложений* – служат для управления работой каждого экземпляра приложения, имеющего оконный интерфейс;

– *вторичные окна документов* – содержат документы, созданные приложениями, поддерживающими многодокументный интерфейс (например, окна с содержимым тестовых файлов, открытых в текстовом редакторе);
– *диалоговые окна* – служат для запроса у пользователя дополнительных параметров или выдачи информационного сообщения.

В качестве примера рассмотрим структуру окна дисков и папок (рис. 2.3). Большинство окон приложений имеют в своем составе такие же элементы, возможно с небольшими дополнениями. Рассмотрим эти элементы подробнее.

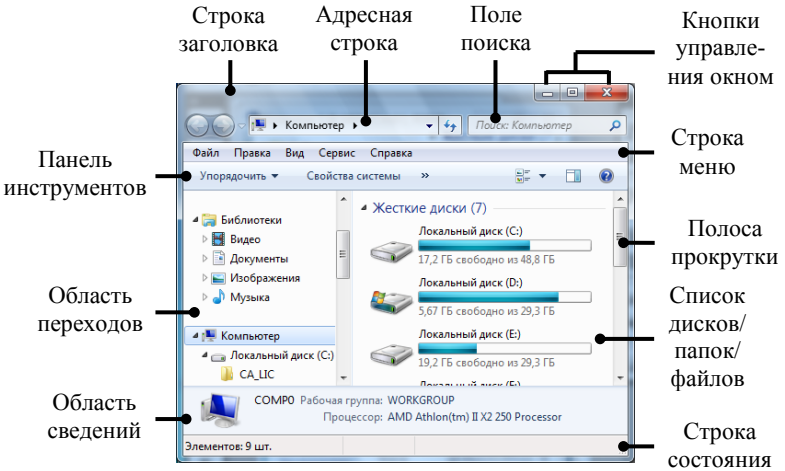


Рис. 2.3. Элементы окна дисков и папок Windows 7

В ***строке заголовка*** справа размещаются три кнопки управления окном: *Свернуть*, *Развернуть*/*Восстановить*, *Закрыть*. Такие же команды обычно дублируются в системном меню, вызвать которое можно щелчком по левому краю строки. В оформлении окна, как и панели задач, активно используется технология Aero Glass, которая делает заголовок и границы окна полупрозрачными (с эффектом матового стекла) и сквозь них можно видеть очертания следующего окна или Рабочего стола.

С помощью мыши с окном можно проделывать ряд манипуляций. Ухватив мышью заголовок окна (т.е. подведя к нему курсор и нажав, не отпуская левую кнопку) можно перемещать окно по экрану. Двойной щелчок по строке заголовка эквивалентен нажатию кнопки

Развернуть/*Восстановить*. Для плавного изменения размеров окна можно потянуть за его границу при нажатой левой кнопке мыши.

Другими интересными технологиями управления окнами являются Aero Shake и Aero Snap. Первая позволяет быстро свернуть все окна, кроме активного. Для этого нужно просто «потрясти» окно, ухватив его за заголовок (или нажать **Win+Home**). Технология Aero Snap «привязывает» окно к краям экрана. Если ухватить окно за заголовок и подвести его к верхнему краю, то оно развернется на весь экран. Повторное движение вниз восстанавливает исходный размер окна. Если подвести окно к левой или правой части экрана, то оно станет занимать ровно его половину. Аналогичного эффекта можно добиться комбинациями клавиш **Win+↑**, **Win+↓**, **Win+←**, **Win+→**.

Адресная строка показывает текущее расположение в иерархии файловой системы, а также позволяет перейти на другой ее уровень. Содержимое строки разделяется на несколько секций, заканчивающихся символом ▸. Щелчок по нему отображает возможные варианты перехода на этом уровне иерархии. Это позволяет быстро сформировать новый адрес отдельно по уровням. Перемещаться по уровням вложенности папок можно и с помощью кнопок *Назад* и *Вперед*, расположенных левее адресной строки. А для того, чтобы увидеть привычный путь к файлам, необходимо щелкнуть левой кнопкой на пустом месте адресной строки.

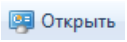
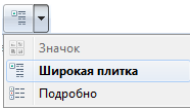
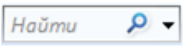
Поле поиска, расположенное справа от адресной строки, позволит найти файлы или папки, расположенные на логических дисках компьютера. Для поиска в поле вводится часть имени файла или папки или шаблон, содержащий подстановочные файлы. Область поиска определяет открытый в списке дисков/папок/файлов уровень файловой системы. Например, для поиска в пределах логического диска *C:* необходимо отобразить содержимое его корневой папки в рабочей области окна. При вводе поисковой информации можно дополнительно задать фильтр по виду искомого объекта, дате изменения, расширению файла.

Строка меню содержит пункты выпадающего меню Windows. Эта строка разработчиками по умолчанию скрыта. Предполагается, что пользователю для выполнения основных операций достаточно средств панели инструментов (см. ниже). Если пользователю все-таки понадобилось временно отобразить строку меню, то он должен нажать клавишу **Alt**.

Панель инструментов является обязательной для окна дисков и папок, но может быть не обязательной для окон других приложений. Состоит она из кнопок и других элементов управления (см. ниже),

щелчок по которым заменяет обращение к соответствующим командам меню. Если подвести курсор мыши к какой-нибудь кнопке и подождать 2–3 секунды, то появится надпись, поясняющая назначение кнопки (всплывающая подсказка).

На панели инструментов могут располагаться следующие виды элементов управления:

Вид	Наименование и назначение
	**Кнопка.** Основной элемент панели инструментов. Нажатие на нее соответствует выполнению какой-либо команды.
	**Кнопка со списком.** Позволяет выбрать один из предлагаемых вариантов действия.
	**Поле со списком.** В отличие то кнопки со списком позволяет, как правило, вводить и собственный вариант.

Состав панели инструментов в окне дисков и папок может меняться в зависимости от содержимого окна. Как правило, всегда отображается кнопка со списком *Упорядочить*, содержащая все необходимые команды для работы с файлами и папками, а также для включения/отключения некоторых элементов окна. Также на панели обычно присутствуют кнопки изменения представления содержимого списка дисков/папок/файлов (крупные или мелкие значки, таблица и т.д.) и вызова справочной системы.

Область переходов содержит четыре независимые иерархии для быстрого перемещения по файловой системе компьютера и общим ресурсам локальной сети: *Избранное*, *Библиотеки*, *Компьютер*, *Сеть*. Для раскрытия очередного уровня иерархии необходимо щелкнуть по значку ▷, расположенному слева от названия. После раскрытия уровня значок изменяет свой вид (◢) и щелчком по нему структуру этого уровня можно свернуть. Щелчок по самому названию уровня отображает его содержимое справа в рабочей области.

Список *Избранное* содержит ссылки для доступа к наиболее часто используемым папкам. Пользователь, при желании, может удалить ссылки, размещенные в списке по умолчанию и добавить туда свои.

Список библиотек по умолчанию содержит четыре наименования: *Видео*, *Документы*, *Изображения*, *Музыка*. Библиотеки представляют

собой коллекции файлов различных типов и служат для более быстрого и централизованного доступа к ним. Добавление папки в библиотеку можно произвести через команду *Свойства* ее контекстного меню или методом перетаскивания. При этом сама папка остается на месте, а в библиотеку просто добавляется информация о ней. Также пользователь может создать свою библиотеку, используя, например, кнопку *Создать библиотеку* панели инструментов или контекстное меню раздела.

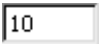
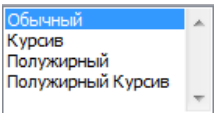
Раскрыв структуру иерархии *Компьютер*, пользователь может увидеть список логических дисков и папок компьютера и перейти на нужный уровень файловой системы. А список *Сеть* позволит отобразить компьютеры, подключенные к локальной сети и получить доступ к предоставляемым ими общим ресурсам.

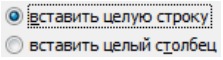
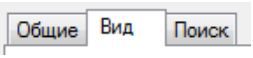
Список дисков/папок/файлов является рабочей областью окна, в которой размещается содержимое выбранного в области переходов уровня файловой системы или локальной сети. Если информация не помещается в рабочей области, окно снабжается вертикальной (справа) и/или горизонтальной (внизу) ***полосами прокрутки***.

Строка состояния отражает информацию о текущем состоянии окна. Эта строка в Windows 7 почти не используется и по умолчанию не отображается. Включить ее можно через пункт выпадающего меню *Вид*.

Область сведений является более информативным аналогом строки состояния. Для выделенной папки или файла показывается тип и дата последнего изменения. Для файла или группы файлов дополнительно отображается размер.

В диалоговых окнах, которые служат для запроса дополнительных параметров при выполнении каких-либо команд меню, могут располагаться те же элементы управления, что и на панелях инструментов и, кроме того:

Вид	Наименование и назначение
	**Текстовое поле ввода.** Позволяет пользователю вводить с клавиатуры текст или числовые значения.
	**Поле со счетчиком.** Позволяет не только ввести с клавиатуры нужное значение, но и увеличивать или уменьшать его с помощью кнопок.
	**Список.** Содержит список объектов, доступных для выбора.

Вид	Наименование и назначение
	**Зависимый переключатель.** Предназначен для выбора одного из взаимоисключающих режимов.
	**Независимый переключатель (флажок).** Используется для включения/ выключения режима, имя которого написано рядом.
	**Ползунок.** Служит для увеличения/уменьшения числового значения поля путем перемещения ползунка.
	**Вкладки** – расположенные под строкой заголовка окна страницы, объединяющие однотипные группы запросов по установке параметров той или иной команды.

Применение графического интерфейса Windows для выполнения операций с файлами и папками

Рассмотрим выполнение основных операций с папками и файлами с помощью встроенных средств Windows, т.е. Проводника.

Для создания новой папки нужно открыть окно, в котором она создается. На свободном месте щелчком правой кнопки мыши вызывается контекстное меню. Выбирается *Создать*, затем *Папку*. В окне появляется изображение папки с запросом об имени. На клавиатуре необходимо набрать имя папки и нажать клавишу **Enter**.

Файл создается в каком-нибудь приложении. После этого в окне Проводника рядом с именем файла появляется значок приложения, его создавшего. Имя файлу дается при его сохранении и в дальнейшем может быть изменено (см. ниже).

Иногда возникает необходимость переместить или скопировать папку или файл в другую папку. Для этого можно открыть папки (источник и приемник) в разных окнах и разместить их рядом. Затем левой кнопкой мыши взять нужный объект (папку или файл) и переместить его в окно папки-приемника. Если папки находятся на разных логических дисках, то произойдет копирование, иначе – перемещение. Для копирования объекта в пределах одного диска, необходимо при его переносе удерживать нажатой клавишу **Ctrl**.

Такой метод перемещения и копирования файлов особенно удобно использовать при работе с деревом папок, отображаемым в области переходов окна дисков и папок Проводника. Дерево позволяет наглядно отобразить иерархию файловой системы. Если раскрыть структуру

диска, то можно увидеть содержимое его корневой папки. Далее, при необходимости, можно последовательно раскрывать папки 1, 2 и последующих уровней. Если щелкнуть по самому значку нужной папки, то справа в рабочей области окна будут отображены входящие в нее файлы и папки. Обычно при перемещении или копировании файла (папки) его значок перетягивается с правой панели на левую и накладывается на значок папки-приемника.

Другим вариантом выполнения операций перемещения и копирования является использование буфера обмена – специальной области оперативной памяти, применяемой для промежуточного хранения перемещаемой или копируемой информации. Для перемещения или копирования файла или папки используются команды контекстного меню *Вырезать* или *Копировать* соответственно. Далее открывается окно папки-приемника и выполняется команда *Вставить*. Причем вставлять объект из буфера обмена можно многократно до тех пор, пока в него не будет помещена другая информация.

Для переименования файла или папки можно вызвать контекстное меню и выполнить команду *Переименовать*.

Удаление какого-либо объекта можно произвести командой *Удалить* из контекстного меню или выделив его щелчком левой кнопки мыши и нажав клавишу **Delete**.

Для выполнения вышеперечисленных операций с группой файлов или папок, их необходимо предварительно выделить. Для этого можно щелкнуть на пустом месте окна и при нажатой левой кнопке мыши обвести появившейся рамкой требуемые объекты. Для выделения несмежных объектов, можно щелкнуть по каждому из них левой кнопкой, удерживая нажатой клавишу **Ctrl**. Дальнейшие действия с выделенной группой ничем не отличаются от действий с одиночными объектами.

Файловые менеджеры

Изначально появление такого класса программ, как файловые менеджеры (также их называют оболочки операционных систем), было призвано облегчить работу пользователя в неграфических операционных системах. Эти программы избавляли от необходимости запоминать команды ОС и позволяли производить основные операции с файловой системой с помощью определенных комбинаций клавиш. И что самое главное, на экране наглядно отображалось содержимое выбранного логического диска. По сравнению с использованием

встроенных команд ОС, переход из одного каталога в другой, запуск программ на выполнение выполнялись более просто.

Наиболее популярной оболочкой для операционных систем *MS DOS* была программа *Norton Commander* (*NC*). Окно программы содержало две *панели* – левую и правую, на которых могло независимо отображаться содержимое двух разных каталогов. Это упрощало выполнение ряда операций.

После появления графических операционных систем применение оболочек, подобных *NC* перестало быть такой необходимостью, как ранее, при работе в системах с интерфейсом командной строки. У графических ОС появились собственные файловые менеджеры (например, Проводник в Windows). Однако наряду со встроенными средствами, по-прежнему применяются оболочки сторонних производителей. Это связано с тем, что в некоторых моментах они являются более удобными для пользователя, нежели стандартные файловые менеджеры. Кроме того, в современные файловые менеджеры встроены различные дополнительные программы, такие, как дисковые утилиты, *FTP*-клиент, архиваторы и т.д.

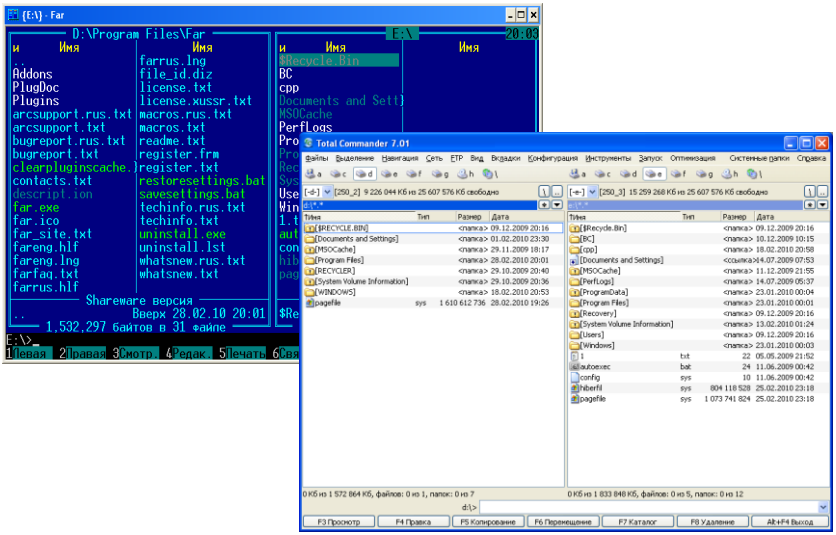


Рис. 2.4. Файловые менеджеры: FAR (слева) и Total Commander

В данный момент одними из наиболее популярных файловых менеджеров являются *FAR* и *Total Commander*. В целом они повторяют интерфейс *NC*, но есть и некоторые отличия (рис. 2.4). *FAR* по

внешнему виду во многом повторяет *NC*, эмулируя вид приложения *MS DOS*. Total Commander напротив имеет графический интерфейс наподобие Проводника. Это дает некоторые дополнительные возможности. Например, Total Commander позволяет производить операции копирования и перемещения перетаскиванием значка файла или папки, в том числе, и в окна Проводника.

Оба файловых менеджера, также как и *NC*, имеют две панели. Та из панелей, которая содержит курсор, является активной, другая соответственно – неактивной. Перемещение курсора на противоположную панель производится клавишей **Tab** или щелчком левой кнопки мыши. В заголовке каждой панели отображается имя открытого в ней каталога, а ниже располагается его содержимое в виде списка имен входящих в него файлов и каталогов. Внизу панели находится сводная строка, содержащая информацию о выделенном курсором файле или каталоге. Ниже панелей находится командная строка. Еще ниже располагается строка подсказок, содержащая сведения о назначении функциональных клавиш.

Все операции в обоих файловых менеджерах производятся с тем файлом или каталогом, который в данный момент выделен курсором или с группой выделенных файлов/каталогов. Для того чтобы войти в выделенный курсором каталог необходимо нажать клавишу **Enter** (при действиях мышью – двойной щелчок левой кнопкой). Для выхода из каталога курсор устанавливается на две точки, расположенные в начале его оглавления, и также нажимается клавиша **Enter**. Если при нажатии **Enter** оказался выделенным исполнимый файл (с расширением *.exe* или *.com*), то происходит его запуск на выполнение.

Рассмотрим те основные операции, которые производятся одинаковым образом в *FAR* и Total Commander.

1. **Операции с панелями.**
   Alt+F1 – смена диска на левой панели;
   Alt+F2 – смена диска на правой панели;
   Alt+F7 – поиск файла;
   Alt+F10 – дерево папок;
   Ctrl+U – обмен панелей местами;
   Ctrl+R – обновление активной панели.
2. **Операции выделения файлов и папок.**
   Insert – выделение (добавление к выделенной группе) отмеченного курсором файла или каталога. Повторное нажатие клавиши снимает выделение с файла или каталога;
   «**Серый +**» (на малой цифровой клавиатуре) – выделение группы файлов по маске (шаблону);

«**Серый –**» – снимает выделение с группы, выделенной клавишей «**Серый +**».

3. **Операции с файлами и каталогами.**

F3 – просмотр файла;

F4 – редактирование файла. В Total Commander для этого используется стандартный редактор *Блокнот*, в *FAR* – встроенный редактор;

Shift+F4 – создание текстового файла с помощью используемого в соответствующем менеджере редактора;

F5 – копирование файла/каталога. Для выполнения операции копирования обычно задействуют обе панели файлового менеджера: на одной из них открывают каталог, содержащий копируемый файл или папку, на другой – каталог в который будет производиться копирование. Далее необходимо выделить нужный файл или каталог, нажать **F5** и в появившемся диалоговом окне подтвердить выполнение операции. Копирование группы выделенных файлов и/или каталогов производится аналогично.

F6 – переименование/перенос файла или каталога. Операция переноса файла/каталога выполняется аналогично операции копирования. Для переименования файла или каталога в диалоговом окне необходимо набрать его новое имя и подтвердить выполнение операции.

F7 – создание каталога;

F8 – удаление файла/каталога;

После запуска некоторых команд на экране появляются диалоговые окна для запроса дополнительных параметров. После их установки выполнение операции необходимо подтвердить нажатием клавиши **Enter**. Для отмены выполнения операции нажимается клавиша **Esc**.

Большинство перечисленных выше команд можно выполнить также через меню файлового менеджера, которое активизируется нажатием клавиши **F9**. Там же можно найти полный перечень операций, выполняемых конкретным файловым менеджером.

СОДЕРЖАНИЕ РАБОТЫ

1. Ознакомиться с рабочим столом. Занести в отчет примеры объектов и ярлыков, находящихся на рабочем столе.
2. Используя главное меню войти в справочную систему Windows. С помощью поля поиска последовательно перейти и

ознакомиться с содержанием пунктов *Создание ярлыков* и *Поиск файлов*. Занести полученные сведения в отчет.

3. Закрыть справочную систему.
4. Используя команду *Пуск* | *Все программы* | *Стандартные* | *Командная строка* произвести запуск интерпретатора команд Cmd.exe.
5. Используя командную строку в папке, доступной для записи (определяется администратором компьютерного зала) создать папку с именем своей группы. Перейти в созданную папку.
6. В папке с именем группы создать папку *Folder*. Перейти в созданную папку.
7. В папке *Folder*, используя командную строку, создать текстовый файл *text1.txt*. В файле указать формат команды, использованный при его создании. Просмотреть содержимое папки *Folder* и файла *text1.txt*.
8. Скопировать текстовый файл *text1.txt* в папку с именем группы и переименовать его в *text2.txt*. Удалить папку *Folder* вместе с содержимым. В отчет обязательно внести **полную** последовательность команд, использованных при работе с командной строкой (можно в виде скриншота). Закрыть интерпретатор команд.
9. На рабочем столе найти ярлык *FAR* (в случае отсутствия – Total Commander) и запустить программу.
10. Используя соответствующие средства файлового менеджера (**выполнение операций с файлами в пунктах 10-14 производить только с помощью клавиатуры!**), произвести поиск ранее созданного файла *text2.txt*. Из диалогового окна поиска перейти в папку, содержащую найденный файл.
11. В папке с именем группы создать папку со своей фамилией. Войти в созданный каталог.
12. Создать текстовый файл с именем *lab.txt*. В файле указать группу и фамилию студента, выполняющего лабораторную работу.
13. Скопировать файл *lab.txt* в папку с именем группы. Переименовать скопированный файл в *lab2.txt*.
14. Удалить файл *lab.txt* из каталога со своей фамилией. Закрыть окно файлового менеджера.
15. Открыть окно дисков и папок (объект Компьютер). Перейти в папку с именем группы.
16. Переместить файл *lab2.txt* в папку со своей фамилией и переименовать его в *Лабораторная работа 2.txt*.

17. После демонстрации результатов работы преподавателю выделить и удалить папку со своей фамилией и файл *text2.txt*.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Понятие операционной системы. Виды интерфейсов пользователя операционных систем.
2. Что такое файл? Каталог? Логический диск?
3. Какие символы допустимо использовать в именах файлов?
4. Что такое путь к файлу, его полное имя?
5. Организация файловой системы. Какие файловые системы могут использоваться в операционных системах Windows?
6. Каков формат команд интерпретатора Cmd.exe для копирования и переименования файлов? Можно ли с помощью команды копирования произвести переименование файла?
7. С помощью каких команд можно создать файл и каталог?
8. Что размещается на рабочем столе?
9. Для чего служит панель задач?
10. Что представляют собой технологии: Aero Glass, Aero Peek, Aero Shake, Aero Snap, Windows Flip, Windows Flip 3D?
11. Какие бывают виды окон?
12. Какова структура окна?
13. Что располагается в строке заголовка?
14. Что располагается в адресной строке?
15. Как выполнить поиск файла в окне дисков и папок?
16. Зачем нужно меню? Какие виды меню вам известны?
17. Каково назначение панели инструментов? Какие элементы управления могут размещаться на панели инструментов?
18. Какие элементы управления могут размещаться в диалоговых окнах?
19. Как в ОС Windows создать файл или папку?
20. Как средствами ОС Windows произвести копирование, переименование, удаление файла или папки?
21. Назначение файловых менеджеров. Какие файловые менеджеры наиболее популярны в настоящее время?
22. Какие основные операции выполняются с помощью файловых менеджеров?

ЛАБОРАТОРНАЯ РАБОТА № 3

СТАНДАРТНЫЕ ПРИЛОЖЕНИЯ WINDOWS: PAINT, WORDPAD, КАЛЬКУЛЯТОР

Цель работы: приобрести навыки работы с приложениями Windows.

КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

В состав семейства операционных систем Windows входит ограниченный набор простейших прикладных программ, позволяющих решать повседневные задачи. Программы, поставляемые вместе с операционной системой Windows, называют ***стандартные приложения***.

Графический редактор Paint

Графический редактор Paint предназначен для создания и редактирования рисунков. Для открытия редактора нужно в главном меню выбрать *Все программы* | *Стандартные* | *Paint*. На рисунке 3.1 представлен внешний вид окна Paint.

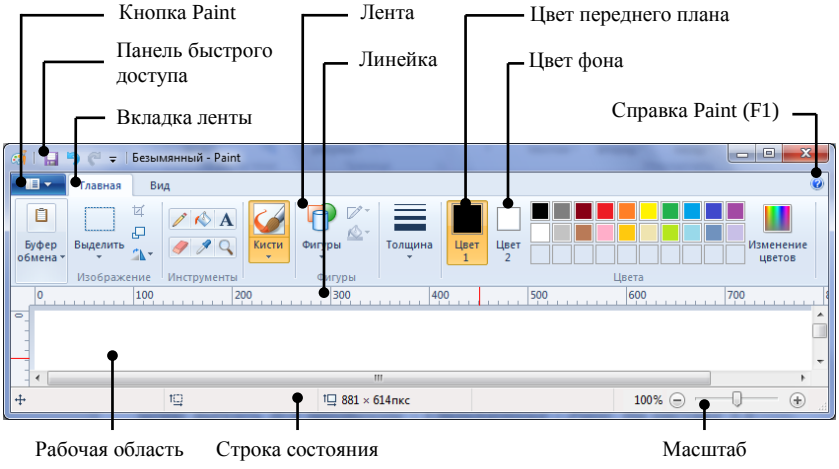


Рис. 3.1. Окно приложения Paint

В верхней части окна находится панель быстрого доступа, имя файла (при создании нового файла присваивается название – Безымянный) и кнопки управления окном. Для работы с файлом (команды *Создать*, *Сохранить*, *Открыть*, *Печатать* и др.) предназначена кнопка Paint. Инструменты объединены в **группы**, располагающиеся на **ленте**. «Прокрутить» ленту можно с помощью колеса прокрутки мыши или с помощью выбора вкладки. Справа от вкладок находится кнопка со знаком вопроса, вызывающая справку. Используя справочную систему редактора, можно выяснить назначение интересующих инструментов. Часто используемые инструменты можно с помощью контекстного меню добавить на панель быстрого доступа.

Текстовый редактор WordPad

Для открытия текстового редактора WordPad нужно выбрать *Пуск* | *Все программы* | *Стандартные* | *WordPad*.

На рисунке 3.2 представлен внешний вид окна WordPad. Приложение предназначено для ввода, редактирования и форматирования текстовых документов, поэтому набор инструментов на ленте отличается от инструментов графического редактора Paint.

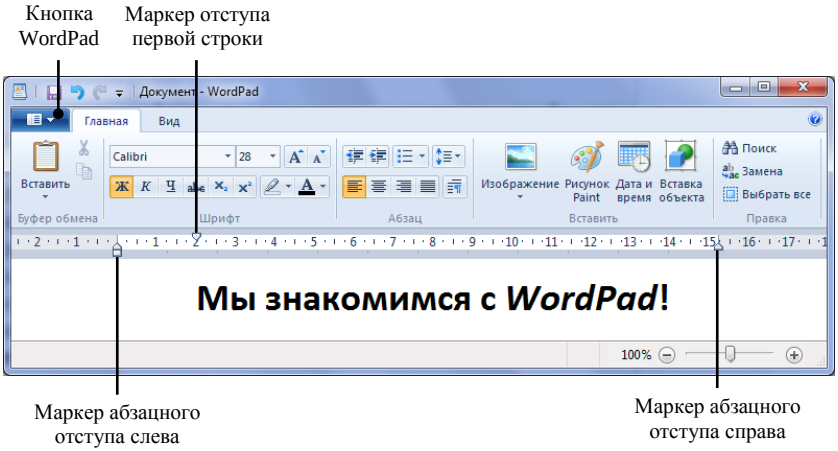


Рис. 3.2. Окно приложения WordPad

Перед началом работы в текстовом редакторе необходимо выполнить команду *Параметры страницы* в меню кнопки WordPad: установить размеры полей и ориентацию страницы. При возникновении

трудностей с понимание пользовательского интерфейса можно обратиться к справке, нажав клавишу **F1**.

Текст должен быть разбит на абзацы. Переход от одного абзаца к другому осуществляется нажатием клавиши ввода (**Enter**). Переход с одной строки на другую осуществляется автоматически. Если возникает необходимость разорвать строку, не делая абзаца, то следует нажать **Shift+Enter**.

Абзац текста может иметь отступы слева, справа, а также отступ первой строки (красная строка). Абзацные отступы можно установить, передвигая соответствующие маркеры (бегунки) на линейке (см. рис. 3.2). Верхний маркер устанавливает отступ первой строки абзаца, нижние – отступы слева и справа соответственно. Параметры форматирования абзаца можно задать в группе *Абзац* вкладки *Главная*.

Набранный текст может быть **отформатирован** (изменен по сравнению с начальными установками) полностью или частично. Форматируемый фрагмент должен быть предварительно выделен. **Выделение** фрагмента текста может быть осуществлено перемещением указателя мыши при нажатой левой кнопке от начала выделяемого фрагмента к его концу. При этом меняется цвет фона фрагмента. Отмена выделения осуществляется щелчком вне этого фрагмента.

Параметры форматирования шрифта выделенного фрагмента можно изменить, используя инструменты группы *Шрифт* вкладки *Главная*.

Калькулятор

Приложение Калькулятор (рис. 3.3) предназначено для вычислений. Открытие приложения осуществляется выбором меню *Пуск* | *Все программы* | *Стандартные* | *Калькулятор*.

Различают четыре вида калькуляторов: Обычный, Инженерный, Программист, Статистика. Переход от одного вида к другому осуществляется через меню *Вид*. Также в меню *Вид* можно выбрать команду *Журнал*, которая отображает выше поля ввода область с произведенными при вычислениях действиями и их результатами.

При вычислениях следует иметь в виду, что приоритет операций умножения и деления в Инженерном калькуляторе соблюдается, а в Обычном – нет. Так, например, нажатие последовательности клавиш 2+3*4= в Инженерном калькуляторе приведет к результату 14, а в Обычном – к результату 20.

Результаты вычислений могут быть переданы в другое приложение через буфер обмена.

Более подробные сведения о работе с калькулятором можно получить с помощью меню *Справка*.

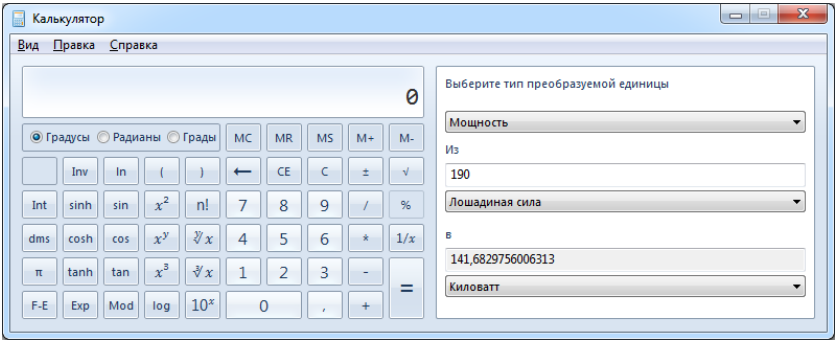


Рис. 3.3. Окно приложения Калькулятор

Рассмотрим пример вычисления значения выражения:

$$\frac{\operatorname{ctg}(\lg 2 + \ln 3{,}8) \cdot (4! - 2{,}7^3)}{\sqrt[7]{\arccos 0{,}8 + e^5} + \sin 50^\circ}$$

Программа вычислений:

*2 log + 3 , 8 ln = Радианы tan 1/x * (4 n!*

- 2 , 7 x^y 3) = MS 0 , 8 Inv $\cos^{-1}$ + 5 Inv e^x = $\sqrt[y]{x}$

*7 = + 5 0 Градусы sin = 1/x * MR =*

Ответ: числитель ≈-0,28; знаменатель ≈2,81; общий ответ ≈-0,10.

СОДЕРЖАНИЕ РАБОТЫ

1. Откройте приложение Paint. Используя справочную систему, выясните и занесите в отчет назначение инструментов на вкладке ленты Главная. Используя не менее семи инструментов, создайте рисунок в соответствии с вариантом. Вставьте название рисунка. Сохраните рисунок в личной папке.
2. Откройте приложение WordPad. Скопируйте, используя буфер обмена, созданный рисунок на открытую страницу. Измените его размеры.
3. На этой же странице (и в отчете) кратко опишите процесс создания (какими инструментами пользовались), сохранения и копирования рисунка. Сохраните файл в личной папке.

4. Откройте приложение Калькулятор. Ознакомьтесь со справкой этого приложения. В соответствии с вариантом (см. таблицу с вариантами заданий на следующей странице) произведите вычисления. В отчет занесите задание, программу вычислений (последовательность нажатия кнопок калькулятора) и результаты вычисления числителя, знаменателя, общий ответ.
5. Сделайте вывод о качестве изученных стандартных приложений Windows.

ВАРИАНТЫ ЗАДАНИЙ:

Номер студента в журнале	**Рисунок**	**Выражение для вычисления с помощью калькулятора**
1	2	3
1, 16	автомобиль	$\frac{tg(e^5 \cdot 1/8 + \sqrt[7]{\pi \cdot (\arccos 0{,}8 + \sin 50^\circ)})}{(\ln 8 - \lg 3) \cdot ctg(2^4 - 7!)}$
2, 17	паровоз	$\frac{tg(e^4 \cdot 1/7 + \sqrt[4]{\pi \cdot (\arccos 0{,}65 + \sin 45^\circ)})}{(\ln 3 - \lg 2) \cdot ctg(3^4 - 6!)}$
3, 18	пароход	$\frac{tg(e^3 \cdot 1/6 + \sqrt[8]{\pi \cdot (\arccos 0{,}35 + \sin 25^\circ)})}{(\ln 12 - \lg 3) \cdot ctg(3^6 - 5!)}$
4, 19	самолет	$\frac{tg(e^2 \cdot 1/3 - \sqrt[5]{\pi \cdot (\arccos 0{,}8 + \sin 50^\circ)})}{(\ln 8 + \lg 3) \cdot ctg(4^5 + 7!)}$
5, 20	ракета	$\frac{tg(e^7 \cdot 1/8 + \sqrt[4]{\pi \cdot (\arccos 0{,}85 + \sin 75^\circ)})}{(\ln 4 + \lg 3) \cdot ctg(6^4 - 4!)}$
6, 21	автобус	$\frac{tg(e^5 \cdot 1/9 - \sqrt[7]{\pi \cdot (\arccos 0{,}95 + \sin 85^\circ)})}{(\ln 5 - \lg 7) \cdot ctg(14^4 - 8!)}$

1	2	3
7, 22	луноход	$\frac{tg\left(e^{8}\cdot 1/6+\sqrt[5]{\pi\cdot(\arcsin 0{,}8+\cos 50^{\circ})}\right)}{(\ln 4-\lg 2)\cdot ctg\left(7^{5}-8!\right)}$
8, 23	трактор	$\frac{tg\left(e^{4}\cdot 1/7+\sqrt[6]{\pi\cdot(\arcsin 0{,}27-\cos 75^{\circ})}\right)}{(\ln 5+\lg 7)\cdot ctg\left(5^{4}-6!\right)}$
9, 24	грузовик	$\frac{tg\left(e^{5}\cdot 1/6+\sqrt[5]{\pi\cdot(\arcsin 0{,}35+\cos 56^{\circ})}\right)}{(\ln 7-\lg 12)\cdot ctg\left(3^{5}+5!\right)}$
10, 25	катер	$\frac{tg\left(e^{9}\cdot 1/8+\sqrt[4]{\pi\cdot(\arcsin 0{,}95+\cos 67^{\circ})}\right)}{(\ln 11-\lg 9)\cdot ctg\left(7^{5}-6!\right)}$
11, 26	вертолет	$\frac{tg\left(e^{2}\cdot 1/3-\sqrt[3]{\pi\cdot(\arccos 0{,}68+\sin 14^{\circ})}\right)}{(\ln 3-\lg 5)\cdot ctg\left(12^{4}-7!\right)}$
12, 27	комбайн	$\frac{tg\left(e^{3}\cdot 1/4+\sqrt[5]{\pi\cdot(\arcsin 0{,}69-\cos 87^{\circ})}\right)}{(\ln 6+\lg 2)\cdot ctg\left(5^{4}-6!\right)}$
13, 28	мотоцикл	$\frac{tg\left(e^{4}\cdot 1/7+\sqrt[4]{\pi\cdot(\arccos 0{,}35+\sin 27^{\circ})}\right)}{(\ln 8-\lg 32)\cdot ctg\left(3{,}5^{5}-10!\right)}$
14, 29	парусник	$\frac{tg\left(e^{5}\cdot 1/6+\sqrt[5]{\pi\cdot(\arcsin 0{,}57+\cos 68^{\circ})}\right)}{(\ln 7-\lg 4)\cdot ctg\left(2^{5}-7!\right)}$
15, 30	звездолет	$\frac{tg\left(e^{6}\cdot 1/8-\sqrt[7]{\pi\cdot(\arcsin 0{,}79-\cos 47^{\circ})}\right)}{(\ln 5+\lg 6)\cdot ctg\left(3^{6}-7!\right)}$

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Как нарисовать прямую и кривые линии, многоугольник, прямоугольник?
2. Каким образом можно нарисовать окружность, квадрат?
3. Как изменить цвет рисунка и фона?
4. Каким образом можно сделать надпись на рисунке?
5. Как переместить рисунок из одного приложения в другое?
6. Как изменить масштаб рисунка?
7. Что такое группа ленты?
8. Как в текстовом редакторе WordPad изменить размеры полей?
9. Какие параметры форматирования абзацев вам известны?
10. Как при наборе текста разорвать строку?
11. Как в текстовом редакторе WordPad изменить размер и тип шрифта?
12. Как в текстовом редакторе WordPad выполняются операции с файлами?
13. Какие существуют виды Калькулятора?
14. Каков приоритет операций в различных видах калькуляторов?
15. Какие тригонометрические функции можно вычислить с помощью приложения Калькулятор?

ЛАБОРАТОРНАЯ РАБОТА № 4

АБСТРАКТНАЯ ВЫЧИСЛИТЕЛЬНАЯ МАШИНА ПОСТА

Цель работы: получить навыки составления алгоритмов для машины Поста, их отладки и исполнения с помощью программы-интерпретатора.

КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Машина Поста (МП) представляет собой один из вариантов уточнения понятия «алгоритм». Являясь абстрактным устройством, МП существует лишь в воображении составителя алгоритма или на бумаге. Однако для упрощения процессов отладки и исполнения алгоритмов для МП можно использовать специальные программы-интерпретаторы. В данной лабораторной работе для этой цели будем использовать интерпретатор ***Algo2000*** (автор – Зартдинов Радик).

Запускает интерпретатор файл *ALGO2000.EXE*, расположенный в папке программы. Интерфейс представлен на рис. 4.1.

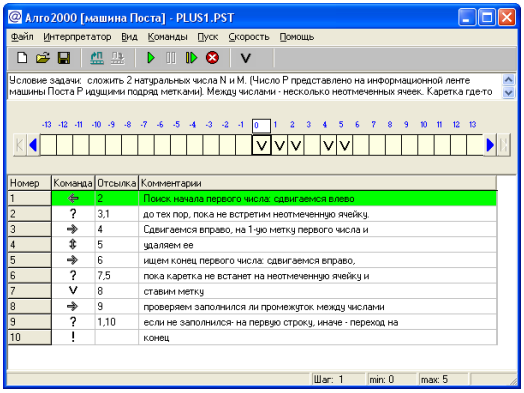


Рис. 4.1. Окно программы *Algo2000*

Окно интерпретатора содержит стандартные элементы: строку заголовка, в которой отображается имя программы, режим работы и имя файла, строку меню, панель инструментов, строку состояния. *Algo2000* является интерпретатором алгоритмов составленных не только для

машины Поста, но и для машины Тьюринга. Переключение режима работы производится с помощью команд пункта меню *Интерпретатор*. На рис. 4.1 показано окно приложения именно в режиме интерпретации программы машины Поста.

Ниже панели инструментов расположено текстовое поле *Условие задачи*, в котором по желанию пользователя можно размещать произвольный текст. Это поле, а также панель инструментов и строка состояния могут отключаться с помощью команд пункта меню *Вид*.

Далее в окне располагается информационная лента (рис. 4.2). Ячейки ленты нумеруются в диапазоне от -999 до 999. Жирной черной рамкой вокруг одной из ячеек обозначается положение каретки. Слева и справа от ленты расположены кнопки ее прокрутки на одну клетку в соответствующем направлении, а также кнопки перемещения каретки к крайним левым и правым меткам.

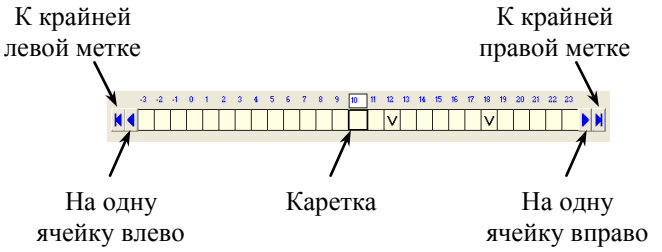


Рис. 4.2. Информационная лента

В большинстве случаев, перед началом выполнения программы для МП на ленте необходимо расставить метки. Для того чтобы установить или удалить метку в какой-либо ячейке ее можно выделить щелчком левой кнопки мыши, а затем нажать клавишу пробела или щелкнуть по кнопке V на панели инструментов. Также можно использовать команду *Поставить/ удалить метки* из пункта выпадающего меню *Команды* или контекстного меню ленты. Для заполнения нескольких соседних ячеек их необходимо выделить с помощью мыши или клавиатуры (**Shift+←/→**) и воспользоваться одним из вышеперечисленных способов. Кроме того, состояние одиночных ячеек удобно изменять с помощью двойного щелчка левой кнопкой мыши.

Для того чтобы при отладке программы МП каждый раз после ее запуска не восстанавливать вручную первоначальное состояние ленты, его можно запомнить. Это делается командой контекстного меню *Запомнить ленту*. Для восстановления состояния ленты выбирается

команда *Восстановить ленту*. Также для этой цели можно использовать кнопки панели инструментов  и  .

Ниже информационной ленты расположена таблица, в которую непосредственно вводятся команды МП. В *Algo2000* используются следующие обозначения команд:

1. **n ← m** – перемещение каретки влево.
2. **n → m** – перемещение каретки вправо.
3. **n ▼ m** – установка метки в ячейку, обозреваемую кареткой.
4. **n ↕ m** – стирание метки в ячейке, обозреваемую кареткой.
5. **n ? m1,m2** – проверка наличия метки в ячейке, обозреваемой кареткой.
6. **n!** – останов МП,

где *n* – номер текущей команды, а *m*, *m*1, *m*2 – отсылки.

Столбец таблицы *Номер* содержит номера команд, которые формируются автоматически. В столбце *Команда* пользователь вводит команды МП, выбирая их из раскрывающегося списка. В столбце *Отсылка* указывается номер следующей команды, подлежащей выполнению. Если выбрана команда ветвления, то верхняя и нижняя отсылки вводятся через запятую. В столбец *Комментарии* можно вносить какие-либо пояснения. При выполнении программы он игнорируется.

Добавлять или удалять строки, очищать какие-либо столбец, строку или таблицу целиком можно используя соответствующие команды контекстного меню таблицы.

После ввода команды в таблице видны только ее номер и тип, а ячейки с отсылкой и комментариями залиты черным цветом и отображаются только при непосредственном выборе их мышью. Полностью же программа отображается только после запуска программы МП на выполнение (чтобы видеть всю программу при наборе можно вводить ее в режиме пошагового выполнения).

Для запуска можно использовать команды пункта меню *Пуск* или соответствующие кнопки панели инструментов. Программу можно запускать в обычном режиме или пошагово. Выполняемая в данный момент команда выделяется зеленым цветом (рис. 4.1). При обычном запуске программы с помощью команд пункта меню *Скорость* можно изменять скорость ее выполнения. Также можно приостановить работу программы или остановить ее полностью. После того, как программа выполнилась до конца, на экране появляется соответствующее сообщение. Кроме того, на экране появится сообщение и при возникновении ошибки. При этом выполнение программы останавливается.

Набранная программа МП может быть сохранена в файле с расширением *.pst* и в дальнейшем при необходимости загружена. Это делается с помощью команд пункта меню *Файл*. В файле сохраняется не только программа МП, но и текущее состояние ленты и условие задачи.

СОДЕРЖАНИЕ РАБОТЫ

1. Запустить программу *Algo2000*. Ознакомиться с интерфейсом программы-интерпретатора и примерами алгоритмов из каталога *EXAMPLES*, расположенного в папке с программой. Изучить справочную систему программы.
2. Составить, ввести, отладить и выполнить программу машины Поста для решения задачи согласно своего варианта (см. ниже).
3. Занести в отчет программу, а также состояние ленты до и после выполнения тестовых запусков (можно в виде скриншотов).
4. Занести в отчет описание ошибок, встречавшихся при отладке программы.

ВАРИАНТЫ ЗАДАНИЯ

Номер студента в журнале	**Формулировка задачи**
1	2
1, 16	На ленте задан массив. Необходимо перенести две левые метки, так чтобы они образовали массив, отстоящий на две позиции левее оставшейся части исходного массива. Если исходный массив состоит из двух или одной меток, оставить его без изменения. Начальное положение каретки – над одной из меток массива.
2, 17	На ленте задан массив. Если он состоит из трех или менее меток, то удвоить его длину. В противном случае оставить массив без изменения. Начальное положение каретки – справа от массива.
3, 18	На ленте задан массив. Определить является ли длина массива четным значением. Если да, то оставить на ленте одну метку, или ни одной в противном случае. Начальное положение каретки – справа от массива или непосредственно над массивом.

1	2
4, 19	На ленте задано двоичное число, содержащее минимум два разряда и каждая цифра (одна или две метки) которого, отделена от другой пустой ячейкой. Если в начале числа расположены две единицы подряд, то в конце его дописать два нуля. Начальное положение каретки – над крайней левой меткой числа.
5, 20	На ленте задано двоичное число, содержащее минимум два разряда и каждая цифра (одна или две метки) которого, отделена от другой пустой ячейкой. Если число четное, то приписать единицу справа от него, в противном случае – слева. Начальное положение каретки – над крайней правой меткой числа.
6, 21	На ленте задано несколько массивов, отделенных друг от друга пустой ячейкой. Если количество массивов больше двух, то объединить два самых правых из них. Начальное положение каретки – над крайней левой меткой самого левого массива.
7, 22	На ленте задано двоичное число, содержащее три разряда и каждая цифра (одна или две метки) которого, отделена от другой пустой ячейкой. Считаем, что старшая двоичная цифра – 1. Проинвертировать цифры числа. Начальное положение каретки – над крайней правой меткой числа.
8, 23	На ленте задано несколько массивов, отделенных друг от друга пустой ячейкой и содержащих более трех меток. В каждом массиве, за исключением самого правого, стереть все метки, кроме крайних. Начальное положение каретки – над крайней левой меткой самого левого массива.
9, 24	На ленте задано три массива, отделенные друг от друга несколькими пустыми ячейками. Удалить те из крайних массивов, длина которых менее трех меток. Начальное положение каретки – над крайней левой меткой самого левого массива.
10, 25	На ленте задано два массива, отделенные друг от друга пустой ячейкой. Определить, имеют ли массивы одинаковую длину. Если да, то оставить на ленте одну метку, или ни одной в противном случае. Начальное положение каретки – над крайней левой меткой левого массива.
11, 26	На ленте задан массив длиной $2n$ меток. Раздвинуть на расстояние в одну ячейку две половины этого массива. Начальное положение каретки – над крайней левой меткой массива.
12, 27	На ленте задано два массива, отделенные друг от друга пустой ячейкой. Начальное положение каретки – над одним из массивов. Стереть тот массив, над которым в начальный момент не находится каретка.

1	2
13, 28	На ленте задано несколько массивов, отделенных друг от друга пустой ячейкой. В каждом массиве, длина которого более двух удалить часть меток, расположенных через одну позицию. Начальное положение каретки – над крайней левой меткой самого левого массива.
14, 29	На ленте задан массив. Необходимо справа от исходного разместить другой массив, втрое больший по длине. Начальное положение каретки – над одной из меток массива.
15, 30	На ленте задано три массива, отделенные друг от друга несколькими пустыми ячейками. Объединить два крайних массива, удалив средний. Начальное положение каретки – над крайней левой меткой самого левого массива.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Для чего предназначена машина Поста?
2. Из каких элементов состоит машина Поста?
3. Перечислите команды машины Поста.
4. Как в машине Поста представляются числа?
5. Какие бывают виды останова машины Поста?
6. Каким образом в машине Поста реализуется ветвление?
7. Каким образом в машине Поста реализуются циклические процессы?
8. Назовите основные возможности интерпретатора *Algo2000*.
9. Каким образом в интерпретаторе *Algo2000* можно изменять состояние ленты?
10. Каким образом в интерпретаторе *Algo2000* производится запуск программы? Как изменяются параметры запуска программы?

ЛАБОРАТОРНАЯ РАБОТА № 5

АБСТРАКТНАЯ ВЫЧИСЛИТЕЛЬНАЯ МАШИНА ТЬЮРИНГА

Цель работы: получить навыки составления алгоритмов для машины Тьюринга, их отладки и исполнения с помощью программы-интерпретатора.

КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Машина Тьюринга (МТ) представляет собой еще один подход к формализации понятия алгоритма. В данной лабораторной работе для исполнения алгоритмов МТ также как и в предыдущей работе будет использоваться интерпретатор *Algo2000*. Для переключения в режим интерпретации программ МТ выбирается соответствующая команда в пункте меню *Интерпретатор*. На рис. 5.1 представлено окно приложения в режиме интерпретации программы машины Тьюринга.

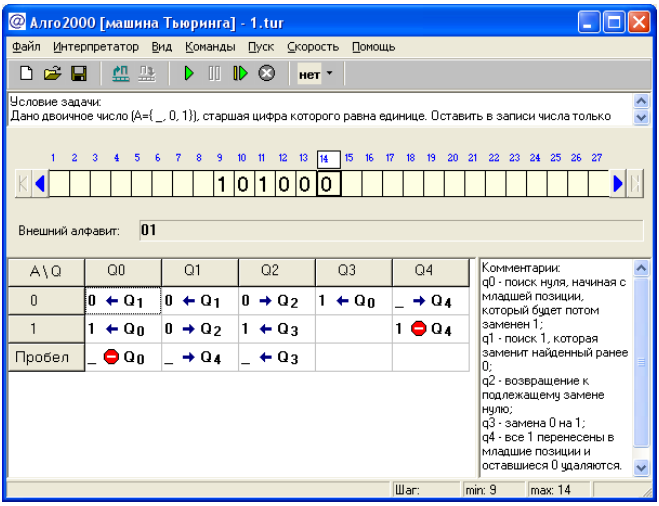


Рис. 5.1. Окно программы *Algo2000* в режиме машины Тьюринга

В этом режиме интерфейс программы в целом аналогичен режиму машины Поста, но есть и некоторые различия. В режиме МТ

информационная лента является «полубесконечной» (ограниченной слева) и ее ячейки имеют номера от 1 до 999. Положение каретки также обозначается черной рамкой вокруг одной из ячеек. Прокрутка содержимого ленты осуществляется аналогично режиму машины Поста.

Ячейки могут содержать в себе один из символов внешнего алфавита, который задается в поле, расположенном ниже ленты. Символы алфавита вводятся без каких-либо разделителей, а символ пробела добавляется автоматически. Пробелы, введенные пользователем, игнорируются. По окончании ввода можно щелкнуть мышью на другом элементе окна или нажать клавишу **Enter**.

Для ввода символа внешнего алфавита в ячейку ленты ее необходимо выделить, а затем нажать на клавиатуре соответствующую клавишу. Также можно использовать кнопку со списком панели инструментов *Отметить символ на ленте* () или команды контекстного меню *Отметить символ на ленте* и *Изменить символ*. Можно заполнять одинаковым символом смежную группу ячеек, если ее предварительно выделить.

В нижней части окна располагается таблица, в которую непосредственно вводится программа для МТ. В заголовках строк автоматически размещаются символы внешнего алфавита. То есть количество строк в таблице равно количеству символов, введенных ранее в поле *Внешний алфавит* плюс один (строка с символом пробела появляется автоматически). При добавлении в дальнейшем во внешний алфавит новых символов, соответствующие им строки также автоматически добавляются в таблицу.

В заголовках столбцов размещаются символы внутреннего алфавита Q_0, Q_1, и т.д. Начальным состоянием управляющего устройства (УУ) считается Q_0. Заключительное состояние МТ в данном интерпретаторе явным образом не указывается, а останов происходит тогда, когда каретка остается на месте. Количество столбцов в таблице соответствует количеству символов внутреннего алфавита и может быть изменено с помощью команд контекстного меню таблицы.

В ячейки таблицы непосредственно вводятся правые части команд МТ. Указанные действия будут выполнены в том случае, когда на ленте кареткой обозревается символ, стоящий в заголовке строки, а УУ находится в состоянии, указанном в заголовке столбца.

Для ввода команды в ячейку таблицы ее необходимо выделить и ввести три значения (безразлично, разделенные пробелами или нет):

1. Символ внешнего алфавита, который будет занесен в клетку ленты, обозреваемую кареткой (для ввода символа алфавита «пробел» используется знак подчеркивания '_').

2. Символ направления перемещения каретки ('<' – влево, '>' – вправо, '!' – каретка остается на месте, останов).
3. Номер нового состояния УУ (необходимо ввести только номер без буквы «*Q*»).

Так, например, для того чтобы ввести в ячейку ленты символ «1», передвинуть каретку влево и перевести УУ в состояние *Q*3 задается следующая команда: **1 < 3**. Закончить ввод команды можно щелчком мыши по другой ячейке таблицы или нажатием клавиши **Enter**. После ввода команда в ячейке будет выглядеть следующим образом:  . В дальнейшем для редактирования команды надо будет выделить ячейку таблицы, а затем щелкнуть на ней мышью или нажать клавишу **Enter**.

Если вычислительный процесс является не всюду определенным, то некоторые ячейки таблицы остаются незаполненными. Однако разработчик алгоритма должен быть четко уверен, что такая комбинация символов внешнего и внутреннего алфавитов невозможна. Если все же при интерпретации программы произойдет переход в пустую ячейку таблицы, это приведет к аварийному останову МТ.

Справа от таблицы находится поле для ввода комментариев к программе. Размер поля комментариев изменяется перемещением мышью разделителя между ним и таблицей.

Запуск программы МТ осуществляется аналогично режиму машины Поста. В случае корректного завершения программы на экране появляется соответствующее сообщение.

СОДЕРЖАНИЕ РАБОТЫ

1. Запустить программу *Algo2000*. Ознакомиться с интерфейсом программы-интерпретатора в режиме машины Тьюринга и имеющимися примерами алгоритмов.
2. Составить, ввести, отладить и выполнить программу машины Тьюринга для решения задачи согласно своего варианта (см. ниже). В результате работы программы исходные данные (если в задании явно не указано иное) **должны оставаться неизменными**.
3. Занести в отчет программу, а также состояние ленты до и после выполнения тестовых запусков (можно в виде скриншотов).
4. Занести в отчет описание ошибок, встречавшихся при отладке программы.

ВАРИАНТЫ ЗАДАНИЯ

Номер студента в журнале	Формулировка задачи
1	2
1, 16	Дано двоичное число, содержащее 2*n* разрядов. Заменить в нем все комбинации двух соседних разрядов «00» на «11» и наоборот. Начальное положение каретки – над крайней правой цифрой числа. После замены вернуть каретку в первоначальное положение.
2, 17	Дано двоичное число. Справа через пробел записать цифры числа в обратном порядке. Начальное положение каретки – над крайней правой цифрой числа.
3, 18	Дано двоичное число. Слева через пробел записать двоичное число, в котором будет продублирована каждая цифра исходного числа. Начальное положение каретки – над крайней правой цифрой числа.
4, 19	Дано двоичное число. Слева через пробел, записать десятичное значение, соответствующее количеству единиц в исходном числе, а справа – значение, соответствующее количеству нулей. Начальное положение каретки – над крайней правой цифрой числа.
5, 20	Дано два двоичных *n*-разрядных числа, отделенные на ленте символом '&'. Слева записать результат их поразрядного «И». Начальное положение каретки – над крайней правой цифрой правого числа.
6, 21	Дано двоичное число, содержащее 2*n* разрядов. Подсчитать в нем количество двух соседних разрядов, равных единице и записать это значение в десятичном виде слева от числа. Начальное положение каретки – над крайней правой цифрой числа.
7, 22	Дано два двоичных *n*-разрядных числа, отделенные на ленте символом '|'. Слева записать результат их поразрядного «ИЛИ». Начальное положение каретки – над крайней правой цифрой правого числа.

1	2
8, 23	Дано двоичное число, содержащее 2*n* разрядов. Зеркально отразить цифры числа относительно середины. Начальное положение каретки – над крайней правой цифрой числа.
9, 24	Дано два двоичных числа, отделенные на ленте пробелом. Поменять числа местами. Начальное положение каретки – над крайней правой цифрой правого числа.
10, 25	Дано двоичное число. Слева через пробел записать двоичное число, в котором будут содержаться нечетные (начиная с младшего) разряды исходного числа. Начальное положение каретки – над крайней правой цифрой числа.
11, 26	Дано два двоичных *n*-разрядных числа, отделенные на ленте символом 'X'. Слева записать результат их поразрядного «исключающего ИЛИ». Начальное положение каретки – над крайней левой цифрой левого числа.
12, 27	Дано четырехразрядное двоичное число. Если его запись является палиндромом, то на ленте оставить одну единицу, в противном случае – один ноль. Начальное положение каретки – над крайней правой цифрой числа.
13, 28	Дано двоичное неотрицательное число, состоящее не более чем из семи значащих разрядов. Получить восьмиразрядный дополнительный код отрицательного числа, с модулем, равным исходному числу и записать его слева от исходного числа. Начальное положение каретки – над крайней правой цифрой числа.
14, 29	Дано двоичное число. Произвести его циклический сдвиг на один разряд влево. Необходимо, чтобы результирующее значение занимало ячейки с теми же номерами, что и исходное. Начальное положение каретки – над крайней левой цифрой числа.
15, 30	Дано двоичное число, содержащее 3*n* разрядов. Слева записать десятичное значение, соответствующее количеству нечетных триад. Начальное положение каретки – над крайней правой цифрой числа.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Для чего предназначена машина Тьюринга?
2. Какие разновидности машин Тьюринга вам известны?
3. Из каких элементов состоит машина Тьюринга?
4. Как в машине Тьюринга задаются команды?
5. Для чего служит внешний алфавит машины Тьюринга?

6. Для чего служит внутренний алфавит машины Тьюринга?
7. Можно ли на машине Тьюринга обрабатывать числа в позиционной системе счисления?
8. В каких случаях может произойти аварийный останов машины Тьюринга?
9. Каким образом задается команда многоленточной машины Тьюринга?
10. Как осуществляется композиция машин Тьюринга?
11. Каким образом в интерпретаторе *Algo2000* можно заполнять ячейки ленты?
12. Каким образом в интерпретаторе *Algo2000* производится ввод и редактирование команд программы?

ЛАБОРАТОРНАЯ РАБОТА №6

РАБОТА В СРЕДЕ MICROSOFT VISUAL STUDIO 2010. РЕАЛИЗАЦИЯ ЦИКЛИЧЕСКИХ АЛГОРИТМОВ СРЕДСТВАМИ ЯЗЫКА C/C++

Цель работы: получить навыки в создании, настройке и отладке консольных приложений на языке программирования C/C++ в среде Visual Studio; ознакомиться с основными библиотечными функциями ввода-вывода; получить навыки в составлении простейших циклических алгоритмов и реализации их средствами языка C/C++.

КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Среда разработки программ Microsoft Visual Studio 2010

Среда Visual Studio 2010 позволяет работать с несколькими языками программирования, в т.ч. и с C/C++. Создаваемые приложения могут разрабатываться как в виде неуправляемого кода для платформы *Win32*, так и управляемого для платформы *.Net*. Далее рассмотрим процесс создания и настройки консольного приложения Win32 на языке C/C++.

Создание решения

После запуска среды на экране появляется ее главное окно, в котором обычно отображается стартовая страница, как на рис. 1.1.

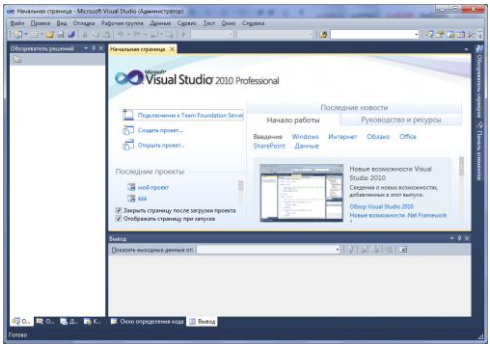


Рис. 1.1. Окно среды разработки программ Visual Studio 2010

Окно содержит заголовок, меню, панель инструментов, строку состояния, а также несколько дочерних окон, таких как *Обозреватель решений*, расположенный с левой стороны окна (рис. 1.1). В зависимости от настроек среды и пожеланий пользователя расположение и состав дочерних окон может отличаться от представленного на рисунке варианта. Стартовая страница, которую можно отключить в настройках среды (*Сервис* |*Параметры*), содержит ссылки создания и открытия проектов, открытия последних созданных проектов, а также запуска обучающих и справочных ресурсов Microsoft.

Для создания консольного приложения можно щелкнуть по ссылке *Создать проект* стартовой страницы или выполнить команду меню *Файл* | *Создать* | *Проект*. На экране появится диалоговое окно (рис. 1.2), в котором можно выбрать один из предлагаемых шаблонов для будущего приложения. В левой панели развернем узел *Visual C++*, а затем в средней части окна выберем пункт *Консольное приложение Win32*. Выбор в поле со списком версии платформы .Net Framework не имеет значения, т.к. приложение создается для платформы Win32. Далее в поле *Имя* в нижней части окна необходимо ввести имя нового проекта, в составе которого можно использовать как латиницу, так и кириллицу (например, «мой проект»). Одновременно автоматически заполняется поле *Имя решения*. В поле со списком *Расположение* можно указать каталог размещения проекта. Если установить флажок *Создать каталог для решения*, то в указанном месте будет создана новая папка с именем, совпадающим с именем решения. Нажатие *ОК* закрывает это окно и запускает окно *Мастера приложений Win32*.

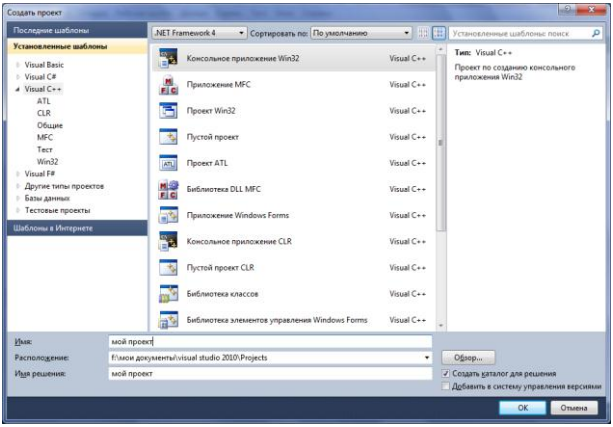


Рис. 1.2. Диалоговое окно *Создать проект*

В первом окне Мастера нажмем кнопку *Далее*. Во втором окне (рис. 1.3) устанавливаются параметры будущего приложения. Переключатель *Тип приложения* оставим в положении *Консольное приложение*. Далее снимем флажок *Предварительно скомпилированный заголовок* и установим флажок *Пустой проект*. Такие параметры означают, что приложение полностью будет создаваться самим программистом. Теперь нажмем кнопку *Готово*. Проект с именем *мой проект* создан.

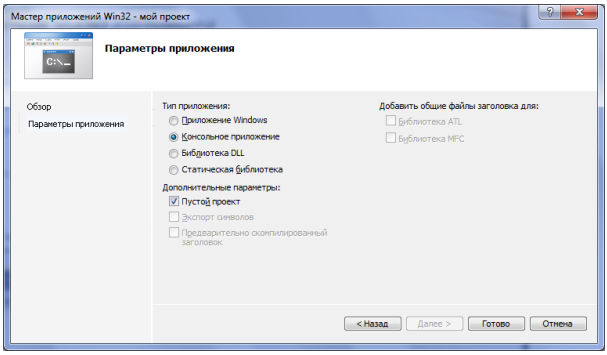


Рис. 1.3. Диалоговое окно Мастера приложений Win32

В среде Visual Studio проект всегда создается в составе *решения*, которое может содержать несколько проектов (в нашем случае один). В свою очередь проект может объединять несколько логически связанных файлов с исходным кодом, настройками конфигурации, ресурсами и т.д. Если перейти в папку, содержащую созданный проект, то мы увидим там ряд файлов:

- *мой проект.sln* – файл решения для созданной программы. Если выполнить двойной щелчок на этом файле, то произойдет запуск среды с открытием данного решения.
- *мой проект.suo* – файл настроек среды при работе с данным решением.
- *мой проект.sdf* – содержит данные, необходимые для работы компонента среды *Intellisence*.

Также в составе каталога с решением имеется подкаталог проекта, который также называется *мой проект* и содержит такие файлы:

- *мой проект.vcxproj* – файл проекта. Запуск этого файла также приведет к старту среды и открытию решения.

- *мой проект.vcxproj.filters* – файл с описанием фильтров, используемых *Обозревателем решения* для отображения разных групп файлов решения.
- *мой проект.vcxproj.user* – файл пользовательских настроек.

В дальнейшем в этом же каталоге будет размещаться файл с исходным текстом программы. Кроме того, в папке с решением будут созданы другие подкаталоги, о назначении которых будет рассказано ниже.

Состав созданного в виде пустого проекта решения отображается в *Обозревателе решения* (рис. 1.4) в виде набора папок.

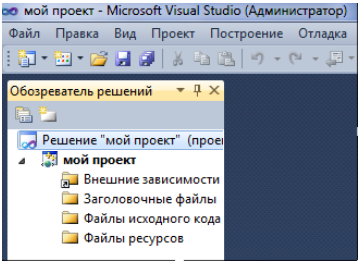


Рис. 1.4. Структура решения

Папка *Внешние зависимости* отображает файлы, не добавленные явно в проект, но использующиеся в файлах исходного кода, например включенные при помощи директивы *#include*. Остальные папки содержат соответственно имеющиеся в проекте заголовочные файлы, файлы исходного кода и ресурсов.

Созданный проект пока не содержит ни одного файла с исходным кодом. Для его добавления можно щелкнуть правой кнопкой мыши на папке *Файлы исходного кода* и выбрать команды *Добавить*, *Создать элемент* (рис. 1.5).

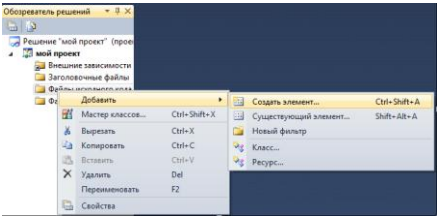


Рис. 1.5. Добавление в проект файла с исходным кодом

В диалоговом окне (рис. 1.6) выберем тип элемента – *Файл C++ (.cpp)* и в поле *Имя* введем название будущего файла, например *main*. Если не указывать расширение, то автоматически к имени файла добавится *.cpp*. Можно также вручную добавлять расширение *.c*, которое будет показывать компилятору, что файл содержит код на «чистом С». Однако если такая программа будет содержать элементы языка С++, то компилятор будет выдавать сообщение о наличии синтаксических ошибок. Поскольку наши программы потенциально могут содержать некоторые средства языка С++, то будем оставлять здесь и в дальнейшем расширение, предлагаемое по умолчанию, то есть *.cpp*.

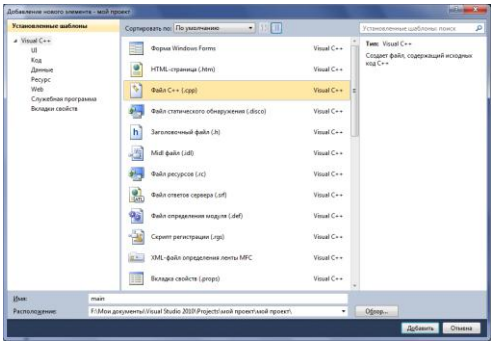


Рис. 1.6. Диалоговое окно *Добавление нового элемента*

После нажатия кнопки *Добавить* появляется окно редактора кода с единственной вкладкой *main.cpp* (рис. 1.7). В дальнейшем в зависимости от действий пользователя и режимов работы среды в редакторе могут открываться в отдельных вкладках и другие файлы. Закрыть вкладку с файлом можно щелкнув по кнопке закрытия на ярлыке вкладки. Для открытия в редакторе кода какого-либо файла проекта достаточно дважды щелкнуть на нем в окне *Обозревателя решений*.

В состав решения можно добавлять и новые проекты. Для этого можно щелкнуть в *Обозревателе решений* правой кнопкой на строке с названием решения и выбрать команды *Добавить*, *Создать проект*. Дальнейшие действия с созданным проектом аналогичны тем, что рассматривались ранее. Из нескольких проектов в составе решения по умолчанию только один в данный момент будет запускаемым. Чтобы назначить запускаемым другой проект нужно в *Обозревателе решений*

правой кнопкой щелкнуть на его имени и выбрать *Назначить запускаемым проектом*.

Если необходимо прекратить работу с текущим решением, то можно выполнить команду *Файл* | *Закрыть решение* или открыть другое решение командой *Файл* | *Открыть*.

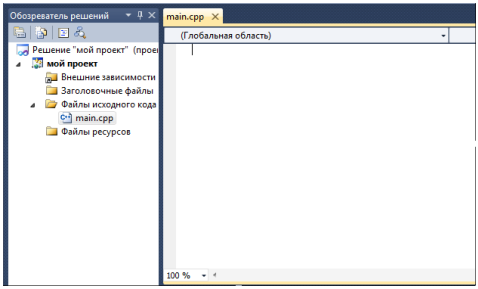


Рис. 1.7. Окно редактора кода

Теперь произведем настройку свойств созданного проекта. Для этого можно выполнить команду меню *Проект* | *Свойства* или нажать **Alt+F7**. В левой части диалогового окна (рис. 1.8) раскроем узел *Свойства конфигурации*. Конфигурации проекта определяют параметры компоновки приложения и свойства устанавливаются для каждой из них отдельно. Изначально каждый проект в решении Visual Studio имеет две конфигурации — *Debug* (Отладка) и *Release* (Выпуск). Каждая из них создается (автоматически) в отдельном каталоге. При использовании конфигурации *Debug* будет создаваться отладочная версия проекта, с помощью которой можно осуществлять отладку на уровне исходного кода. Конфигурация *Release* предназначена для окончательной сборки приложения. В ней отсутствует отладочная информация и при создании выходного файла (с расширением *.exe*) компиляция происходит с включенным режимом оптимизации кода, что уменьшает его объем (по сравнению с конфигурацией *Debug*). Для примера настроим некоторые свойства активной конфигурации *Debug*.

После раскрытия узла *Свойства конфигурации* выберем пункт *Общие*. Установим свойство *Набор символов* в значение *Использовать многобайтовую кодировку*. Данное свойство означает, что по умолчанию в программе будут использоваться строки и строковые константы в однобайтовой кодировке ANSI и соответствующие функции их обработки.

У проекта и его конфигураций имеется много других свойств. Например, раскрыв узел *C/C++* и выбрав пункт *Дополнительно*, можно установить значение свойства *Компилировать как*. Это свойство позволяет принудительно установить параметры компиляции независимо от расширения файла исходного кода. Свойство *Командная строка* позволяет увидеть с какими параметрами будет запускаться компилятор среды (*cl.exe*). Однако мы оставим эти свойства без изменения и нажмем *ОК*.

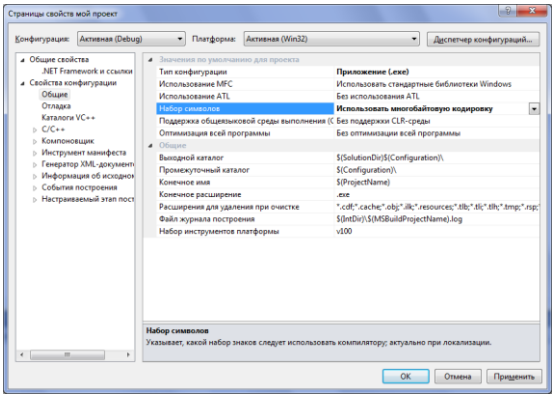


Рис. 1.8. Диалоговое окно определения свойств проекта

Редактор кода

Среда Visual Studio включает в себя удобный редактор для набора исходного текста программы. Перечислим некоторые полезные возможности редактора кода:

- подсветка служебных слов языка и комментариев;
- разбиение кода на логические группы. Редактор автоматически объединяет фрагменты кода в группы, которые можно свернуть или развернуть воспользовавшись значками с символами «-» и «+» слева от кода (рис. 1.9). Примером таких логических групп кода может быть блок комментариев или составной оператор тела функции;
- автоматическое создание отступов в коде. Для улучшения читабельности текста программы редактор автоматически добавляет отступы при вводе содержимого составного и некоторых других операторов. Также автоматически

устанавливаются позиции самих открывающей и закрывающей фигурных скобок (с одинаковым отступом);
- удобное перемещение между началом и концом составного оператора. Если курсор расположен на открывающей фигурной скобке, то после нажатия **Ctrl+]** он автоматически перемещается на закрывающую скобку;
- демонстрация отладочной информации: выделение в тексте ошибок, размещение точек прерывания, закладок и т.п.
- поиск и замена фрагментов кода;
- вывод информации об определении элементов кода;
- использование технологии автодополнения *IntelliSense*.

Рассмотрим более подробно две последние возможности.

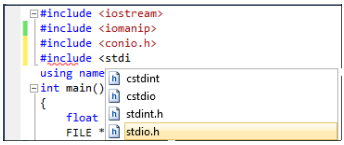


Рис. 1.9. Пример применения технологии *IntelliSense*

При работе с текстами программ (в особенности больших и состоящих из нескольких модулей) часто требуется получить напоминание о типе той или иной переменной или о том, как выглядит прототип (заголовок) какой-либо функции. Самый простой способ получения такой информации – подвести курсор мыши к нужному элементу и дождаться появления всплывающей подсказки. Если же требуется информация о том, в каком файле, и в каком месте этого файла определен элемент программы, то можно воспользоваться вкладкой *Окно определения кода* информационного окна, расположенного по умолчанию в нижней части окна среды программирования. Если установить курсор на элементе программы, то через несколько секунд в *Окне определения кода* появится текст файла и будет выделена строка, содержащая нужное определение (рис. 1.10).

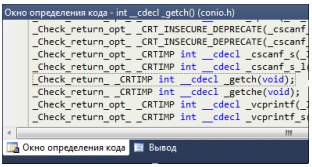


Рис. 1.10. Окно определения кода

По-другому отобразить определение элемента можно, если установить на него курсор и нажать комбинацию клавиш **Ctrl+F12.** Если определение находится в этом же файле, то оно будет выделено. Если в другом, то он будет открыт в отдельной вкладке редактора кода и название элемента будет также выделено.

Технология автодополнения кода *IntelliSense* позволяет программисту не вводить полностью название функции или заголовочного файла, а выбрать их из списка, сформированного по первым введенным буквам. Например, если при наборе директивы препроцессора ввести `#include <` , то раскроется список всех доступных заголовочных файлов (рис. 1.9). Если вводить первые символы имени, то курсор начнет перемещаться по списку, сужая круг возможных вариантов. Для ввода выбранного варианта следует нажать **Tab** или **Enter.**

Для того, чтобы полностью не вводить название какой-либо функции или символической константы можно ввести первые несколько символов имени и нажать **Ctrl+Пробел**. Будет показан список возможных вариантов. После выбора одного из них и ввода открывающей круглой скобки появится всплывающая подсказка о параметрах функции.

Средства ввода-вывода данных

Прежде, чем рассмотреть примеры простейших программ, вкратце остановимся на тех средствах ввода-вывода, которые можно использовать в консольных приложениях.

Языки C и C++ лишены встроенных средств ввода-вывода, который осуществляется с помощью библиотечных функций. Языку C++ в наследство от языка C досталась стандартная библиотека ввода-вывода, прототипы функций которой содержатся в заголовочном файле *stdio.h*. Для того чтобы использовать эти функции, в начале программы необходимо разместить директиву препроцессора `#include <stdio.h>`.

Вывести информацию в стандартный поток вывода (на экран) можно с помощью функции форматированного вывода *printf*, которая может иметь переменное число параметров. Первым параметром является управляющая строка, которая содержит компоненты трех типов: обычные символы, которые просто копируются в стандартный выходной поток; спецификации преобразования, каждая из которых вызывает вывод на экран очередного аргумента из последующего списка; управляющие символьные константы. После управляющей

строки могут размещаться через запятую аргументы, представляющие собой какие-либо выражения. Например, для вывода значения целочисленной переменной можно использовать следующий фрагмент:

```
int a=5;
printf(“a= %d\n”,a);
```

В результате такого вызова функции первые три символа управляющей строки (включая и пробел) без изменений выводятся на экран. Затем вместо спецификации преобразования *%d* выводится значение переменной *a*. Далее в выходной поток выводится управляющий символ *‘\n’*, что обеспечивает перевод курсора в начало следующей строки.

Для ввода данных можно использовать функцию *scanf*, которая имеет сходный набор параметров. В управляющей строке могут размещаться спецификации преобразования, а также и пробелы и символы табуляции. Аргументы должны быть указателями на переменные соответствующих типов. Для этого перед именем переменной записывается символ *&* (операция взятия адреса переменной). Например, для ввода целочисленной и вещественной переменных можно использовать следующий фрагмент:

```
int x;
double y;
scanf(“%d%f”,&x,&y);
```

Кроме стандартной библиотеки ввода-вывода языка С можно использовать инструменты стандартной библиотеки языка С++. Средства ввода-вывода для консольных приложений описаны в заголовочном файле *iostream*. Для их использования в начале текста программы размещаются следующие инструкции:

```
#include <iostream>
using namespace std;
```

В отличие от заголовочных файлов языка С, заголовочные файлы стандартной библиотеки С++ не имеют расширения. Все идентификаторы стандартной библиотеки определены в пространстве имен *std*. Такой подход позволяет использовать одни и те же названия в разных библиотеках. Для доступа к объектам необходимо перед их именами размещать квалификатор *std::* (например, *std::cout*) или указывать оператор *using*. Во втором случае ниже указанной строки можно использовать все имена без квалификаторов.

Заголовочный файл *iostream* содержит описание классов для управления вводом и выводом данных, а также определения

стандартных объектов-потоков ввода с клавиатуры (*cin*) и вывода на экран (*cout*). Также в файле определены (перегружены) для разных типов данных операции помещения в поток << и чтения из потока >>. Например, для ввода с клавиатуры значений двух целых переменных и вывода их суммы можно использовать следующий фрагмент:

```
int x,y;
cout<<”Введите два числа”<<’\n’;
cin>>x>>y;
cout<<”Сумма чисел= “<<x+y<<endl;
```

В приведенном примере для перевода строки при выводе используются два альтернативных варианта: вывод символьной константы с управляющим символом *‘\n’* или применение манипулятора *endl*.

Пример составления и отладки простейшей программы

Для примера в созданном ранее файле *main.cpp* разместим программу, которая выводит на экран фразу *Здравствуй, мир* (рис. 1.11). Первая строка программы содержит директиву препроцессора, включающую в текст программы содержимое заголовочного файла *stdio.h*. Далее идет определение функции *main*, которая обязательно должна быть в программе и с которой начинается ее выполнение. Согласно стандарта C++ функция *main* должна быть типа *int* и возвращать системе значения нуля в случае успешного завершения с помощью оператора:

```
return 0;
```

Фактически же достижение при выполнении программы закрывающей фигурной скобки эквивалентно применению данного оператора, поэтому его можно не указывать.

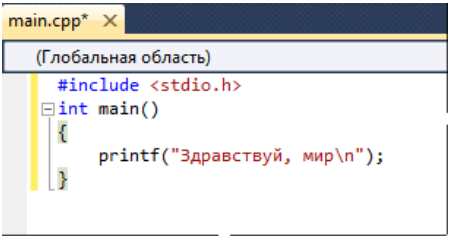


Рис. 1.11. Пример программы

Для того, чтобы запустить набранную программу на выполнение можно использовать команду *Отладка* | *Начать отладку* или нажать **F5**. На экране появится окно, в котором будет сказано, что проект устарел и нужно выполнить его построение (т.е. компиляцию и компоновку). После нажатия кнопки *Да* можно увидеть появление сведений о ходе построения во вкладке *Вывод* информационного окна. Если в тексте будут обнаружены синтаксические ошибки, то в это окно о них выводятся сообщения с указанием номера ошибки. Если вызвать контекстное меню сообщения и выполнить команду *Найти в коде*, то курсор переместится в место, где ошибка обнаружена.

В нашем случае ошибок нет и произойдет запуск программы в конфигурации, установленной по умолчанию (*Debug*). При этом окно консоли откроется и тут же закроется. Для того, чтобы выполнить задержку программы после вывода изменим ее следующим образом:

```
#include <stdio.h>
#include <conio.h>
int main()
{
    printf("Здравствуй, мир\n");
    getch();
}
```

В данном случае мы добавили в программу вызов функции *getch*, прототип которой определен в файле *conio.h*, где описаны функции ввода-вывода для консольного терминала. Данная функция ждет нажатия любой клавиши и возвращает ее код. После повторного запуска проекта мы увидим результаты работы функции вывода (рис. 1.12).

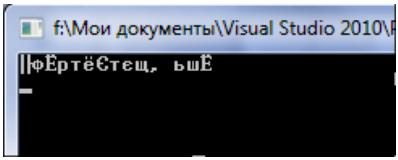


Рис. 1.12. Результат работы программы

Однако в окне консоли мы не увидим слов *Здравствуй, мир*. Это происходит потому, что текст программы был набран в кодировке win-1251 (кодовая страница CP1251), а по умолчанию в окне консоли используется кодовая страница CP866. Для ее изменения используем функцию *setlocale*, которая изменяет так называемую схему

локализации или *локаль*. Локаль определяет кроме кодировки символы валюты, систему мер и другие параметры. Прототип функции определен в файле *locale.h*. У нее два параметра. Первый определяет локализуемую категорию. В данном случае это кодировка символов, поэтому укажем константу LC_CTYPE (можно также LC_ALL – все категории). В качестве второго параметра должна быть указана строка с названием локали. В нашем случае это *Russian* или *rus*. Однако поскольку программа будет выполняться под управлением русифицированной операционной системы, то можно записать в качестве названия пустую строку. Это означает, что будет применена локаль, используемая по умолчанию в данной системе. Таким образом, окончательный вариант программы выглядит так:

```
#include <stdio.h>
#include <conio.h>
#include <locale.h>
int main()
{
    setlocale(LC_CTYPE,"");
    printf("Здравствуй, мир\n");
    getch();
}
```

Теперь рассмотрим такое средство отладки программ, как пошаговое выполнение. В Visual Studio имеется два варианта пошагового выполнения программы: с заходом внутрь функций (**F11**) и с обходом (**F10**). Чтобы не заходить внутрь библиотечных функций, нажмем **F10**. Далее при необходимости производится построение и слева от первого оператора появляется желтая стрелка, показывающая текущий оператор. Далее нажатием клавиши **F10** выполняем строки программы. При наличии в программе переменных их значения можно отслеживать во вкладках информационного окна *Видимые, Локальные, Контрольные значения 1*.

После выполнения последней строки в редакторе в отдельной вкладке открывается файл *crtexe.c*, в который передается управление. Данный файл содержит функцию *mainCRTStartup*, которая запускается перед функцией *main* и предназначена для инициализации библиотеки времени выполнения языка С – *C Run-Time Library* (*CRT*). После выхода из функции *main* данная функция также должна завершить свою работу. Для того чтобы не выполнять ее операторы по отдельности, можно нажать комбинацию клавиш **Shift+F11** (шаг с выходом). Другим вариантом является прекращение отладки (**Shift+F5**).

Операторы цикла языка С/С++

В языке С/С++ имеются три оператора цикла:

1. Оператор цикла *for*.

Данный оператор используется в основном в тех случаях, когда в программе необходимо организовать выполнение цикла с параметром, т.е. цикла в котором некоторая переменная изменяется от начального до конечного значения с заданным шагом. Синтаксис этого оператора следующий:

for (<*выражение_1*>; <*выражение_2*>; <*выражение_3*>)
    <*оператор*>

Тело цикла составляет либо один оператор, либо несколько операторов, заключенных в фигурные скобки { ... } (составной оператор).

Выражение_1 описывает инициализацию цикла (обычно присваивает начальное значение управляющей переменной).

Выражение_2 – проверка условия завершения цикла. Если оно истинно, то выполняется оператор тела цикла.

Выражение_3 вычисляется после каждой итерации (обычно изменяет на каждом шаге значение управляющей переменной).

Пример*:*

```
for (int i=1;i<=10;i++)
  printf("i=%d\n",i);
```

Этот фрагмент программы осуществляет вывод на экран десяти значений переменной *i* от 1 до 10.

В отличие от некоторых других языков программирования, здесь в цикле *for* параметр может иметь любой тип и изменяться с произвольным шагом.

Любое из трех выражений может отсутствовать, но точки с запятой их разделяющие опускать нельзя

```
for ( ; ;)// Бесконечный цикл
```

Выражения 1 и 3 могут состоять из нескольких выражений, объединенных операцией запятая.

Пример:

Написать программу, вычисляющую значение функции $z = \ln(x)/\sin(y)$ при $x \in [1;1.5]$ изменяющимся с шагом $h_1 = 0{,}1$ и $y \in [1;2]$ и изменяющимся с шагом $h_2 = 0{,}2$.

```
/*Включение в текст программы заголовочного файла стандартной
библиотеки ввода-вывода:*/
#include <stdio.h>
```

```
/*Включение в текст программы заголовочного файла с прототипами математических функций:*/
#include <math.h>
int main()
{
   double x,y,z;
   for (x=1,y=1;y<=2;x=x+0.1, y=y+0.2)
   {
      z=log(x)/sin(y);
      printf("x=%5.2f y=%5.2f z=%5.2f\n",x,y,z);
   }
}
```

2. Оператор цикла *while*.

Данный оператор реализует цикл с предусловием. Его синтаксис:

while (<*выражение*>)

<*оператор*>

Если выражение в скобках истинно, то выполняется оператор тела цикла, который может быть и составным.

Пример:

Реализовать предыдущий пример, с применением оператора *while*.

```
#include <stdio.h>
#include <math.h>
int main()
{
   double x=1,y=1,z;
   while (y<=2)
   {
      z=log(x)/sin(y);
      printf("x=%5.2f y=%5.2f z=%5.2f\n",x,y,z);
      x+=0.1;
      y+=0.2;
   }
}
```

3. Оператор цикла *do-while*.

Данный оператор реализует цикл с постусловием. Его синтаксис:

do

<*оператор*>

while (<*выражение*>);

Если выражение в скобках истинно, то выполняется оператор тела цикла.

Пример:

Реализовать предыдущий пример, с применением оператора *do-while*.

```
#include <stdio.h>
#include <math.h>
int main()
{
   double x=1,y=1,z;
   do
   {
      z=log(x)/sin(y);
      printf("x=%5.2f y=%5.2f z=%5.2f\n",x,y,z);
      x+=0.1;
      y+=0.2;
   }
   while (y<=2);
}
```

СОДЕРЖАНИЕ РАБОТЫ

1. Ознакомиться с теоретическим материалом.
2. В среде Visual Studio 2010 создать решение (консольное приложение). Настроить его свойства по аналогии с примером, рассмотренным в теоретических сведениях. В составе решения составить программу, которая выводит на экран ФИО студента, выполняющего работу и номер группы. Также программа должна содержать описание двух целочисленных переменных, которые вводятся с клавиатуры, а затем их сумма выводится на экран. Использовать сначала средства ввода-вывода языка С, затем С++.
3. При наборе программы отработать использование основных возможностей редактора кода.
4. Произвести отладку программы в обычном и пошаговом режимах. В отчет внести текст программы, а также скриншоты информационного окна после построения и при пошаговом выполнении программы (со значениями локальных переменных) и окна консоли с результатом работы программы.
5. Выбрать алгоритм, составить его блок-схему и программу с использованием оператора цикла *for* для вычисления и вывода на экран в точках $x_i=a+i\cdot h$, $i=0,1,2\ldots,n$, $h=(b-a)/n$ промежутка $[a,b]$ значений функции $y=f(x)$, указанной в варианте задания (см. ниже). Также программа должна определять наибольшее и

среднее значение функции. Предусмотреть проверку вычисляемых значений аргумента на принадлежность области допустимых значений. Ввод исходных данных (a, b, n) осуществлять с клавиатуры.

6. Составить аналогичные блок-схему и программу, но с использованием оператора цикла *while* или *do-while* на выбор.
7. Создать новое решение, в которое в виде отдельных проектов включить программы, созданные при выполнении пунктов 5 и 6. В отчет внести обе блок-схемы и программы, а также результаты их тестирования.

ВАРИАНТЫ ЗАДАНИЯ

1. $y = \dfrac{x}{x^2 - 1} + \log_3(x + 2)$,

   $x \in [2; 3]$, $n = 10$

2. $y = \dfrac{x^3}{(x+1)(x+2)} + \dfrac{\arcsin(1-x)}{\sqrt[3]{1 - \ln x}}$,

   $x \in [1; 2]$, $n = 10$

3. $y = \dfrac{\sin x}{1 - \cos x} \cdot \dfrac{\operatorname{tg}^3(\ln(1-x))}{\left|1 + x \cdot e^{-x}\right|}$,

   $x \in [-1; -0{,}5]$, $n = 5$

4. $y = \dfrac{-\arccos(1-x)}{\sqrt[4]{x^3 - 1}} + (2 - x)\cos^2|x|$,

   $x \in [1{,}5; 2]$, $n = 5$

5. $y = \dfrac{\cos^2 x}{1 + \sin x} - \ln^2\left(\dfrac{x}{\sqrt[3]{x - 1}}\right)$,

   $x \in [2; 3]$, $n = 10$

6. $y = \dfrac{x^3 e^{x-1}}{x^3 - |x|} - \log_2(\sqrt{x} - x)$,

   $x \in [0{,}2; 0{,}8]$, $n = 6$

7. $y = \dfrac{\sqrt[3]{x + \sin x}}{x^2 - x^4} \cdot \arcsin^2 \sqrt[4]{3 - x}$,

   $x \in [2; 3]$, $n = 10$

8. $y = \dfrac{x^5 + e^{-2|x|}}{\sqrt[4]{9 - x^2}} \cdot \operatorname{tg}^3\left|\cos^2 x\right|$,

   $x \in [1; 2]$, $n = 10$

9. $y = \dfrac{\sin x + \dfrac{1}{x}}{\sqrt[3]{\operatorname{tg}^2\left(-\dfrac{x^3}{x^2 - 4}\right)}} + 2^{|x-1|}$,

   $x \in [1; 2]$, $n = 5$

10. $y = \dfrac{\sin^2 \dfrac{|x|}{2} + 3^{\frac{1}{x-1}}}{\sqrt[6]{x^4 - 16}} \cdot \sqrt{1 - \ln x}$,

    $x \in [2{,}2; 2{,}6]$, $n = 4$

11. $y=\dfrac{\sqrt[4]{8x^2-6x+1}}{\operatorname{arctg}\sqrt{2x+1}}+2^{\sin x/|x|}$,

$x\in[1;\,2]$, $n=10$

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

$x\in[3{,}5;\,4]$, $n=5$

13. $y=\sqrt[3]{\log_2(1-x)}+\dfrac{\operatorname{tg}(1+1/x)}{\sqrt{|x|}-2}$,

$x\in[-2;\,-1]$, $n=10$

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

$x\in[0{,}5;\,1{,}5]$, $n=10$

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

$x\in[0;\,1]$, $n=10$

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

$x\in[1;\,2]$, $n=10$

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

$x\in[3;\,4]$, $n=10$

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

$x\in[1;\,2]$, $n=10$

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

$x\in[0;\,1]$, $n=10$

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

$x\in[2;\,3]$, $n=10$

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

$x\in[1;\,2]$, $n=10$

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

$x\in[2;\,3]$, $n=10$

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

$x\in[1;\,2]$, $n=10$

24. $y=\dfrac{x}{\cos(x-\pi/2)\sin^2(x-\pi/2)}+e^{\sqrt{x}-|x|}$,

$x\in[2;\,3]$, $n=10$

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

$x \in [3; 4]$, $n = 10$

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

$x \in [1; 1{,}5]$, $n = 5$

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что в среде Visual Studio называется решением? Чем решение отличается от проекта?
2. Как создать решение?
3. Как в готовое решение добавляется еще один проект?
4. Как в состав проекта добавляются новые файлы?
5. Какие типы файлов входят в состав решение, проекта?
6. Каким образом в среде Visual Studio настраиваются свойства проекта?
7. Перечислите основные возможности редактора кода в среде Visual Studio?
8. Что такое конфигурация проекта? Как ее можно изменить?
9. Как можно произвести локализацию создаваемого консольного приложения?
10. Каким образом в среде Visual Studio производится отладка создаваемого приложения? Какие средства отладки вы знаете?
11. Формат записи оператора цикла *for*.
12. Формат записи оператора цикла *while*.
13. Формат записи оператора цикла *do-while*.
14. Каким образом можно включить несколько операторов в тело цикла?
15. Может ли управляющая переменная в цикле *for* быть вещественной?
16. Допустима ли форма записи цикла *for*, в которой отсутствует условие выхода? Если да, то сколько раз выполнится такой оператор?
17. Отличия оператора цикла *while* от *do-while*.

ЛАБОРАТОРНАЯ РАБОТА № 7

ОБРАБОТКА ОДНОМЕРНЫХ МАССИВОВ

Цель работы: приобрести практический опыт использования одномерных массивов.

КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Массивы являются примером структурных (составных) типов данных. Массив – это индексированный набор переменных определенного типа, имеющий общее для всех своих элементов имя. В зависимости от количества индексов, определяющих положение элемента, массивы подразделяются на одномерные, двумерные и т.д. Пример использования одномерного массива:

```
#include<stdio.h>
#define N 3
int main()
{
   // Описание переменных:
   int list[N], i;
   // Присваивание элементам массива значений:
   list[0] = 421;
   list[1] = 53;
   list[2] = 1806;
   // Вывод элементов массива на экран:
   printf("Элементы массива: \n ");
   for(i=0; i<N; i++)
      printf("%d-й элемент: %d \n ",i+1,list[i]);
}
```

Результаты работы данной программы будут иметь вид:

```
Элементы массива:
 1-й элемент: 421
 2-й элемент: 53
 3-й элемент: 1806
```

Выражение *int list[N]* объявляет *list* как массив переменных типа *int*, с объемом памяти, выделяемым для трех целых переменных (так как *N* равно 3). Индексы массива всегда целые и начинаются с нуля. К первой переменной массива можно обращаться как *list[0]*, ко второй – как

list[1], к третьей – как *list[2]*. В общем случае описание любого массива имеет следующий вид:

<тип> <имя>[размер]

Наряду с непосредственным присваиванием, существуют и другие способы ввода элементов массива. Например, элементы массива можно ввести с клавиатуры:

```
int a[10],i;
printf("Введите 10 элементов массива:");
for(i=0; i<10; i++)
{
   printf("%d-й элемент --> ",i+1);
   scanf("%d", &a[i]);
}
```

Другим способом является непосредственная инициализация массива:

```
int a[10]={1,2,3,4,5,6,7,8,9,10};
```

СОДЕРЖАНИЕ РАБОТЫ

Выбрать алгоритм, составить его блок-схему и программу для решения выбранного варианта задания. Исходный массив может быть введен с клавиатуры или инициализирован при описании.

Исходные и результирующие массивы вывести на экран в виде:

x_0 x_1 x_2 x_3 x_4
x_5 x_6 x_7 x_8 x_9
x_{10} x_{11} x_{12} x_{13} x_{14}
x_{15} x_{16} x_{17} x_{18} x_{19}

ВАРИАНТЫ ЗАДАНИЯ

1. В заданном массиве *X*, состоящем из 20 элементов, определить и вывести на экран первый отрицательный элемент и его порядковый номер, а затем заменить его произведением предшествующих значений. Если все элементы положительны, выдать соответствующее сообщение.
2. Задан целочисленный массив *X* из 20 элементов. Вывести на экран все группы идущих подряд одинаковых элементов. Выдать соответствующее сообщение, если таких групп элементов в массиве нет.

3. Задан массив *X* из 20 элементов и число *N* (*N*<20). Не прибегая к сортировке, определить и вывести на экран *N* наибольших элементов массива.
4. В заданном целочисленном массиве *X*, состоящем из 20 элементов и упорядоченном по неубыванию, определить и вывести на экран те элементы, которые можно представить суммой двух других элементов.
5. Задан целочисленный массив *X* из 20 элементов. Определить и вывести на экран те элементы, делителем которых является хотя бы один из других элементов.
6. В заданном массиве *X*, состоящем из 20 элементов, определить и вывести на экран количество положительных, отрицательных и равных нулю элементов. Если положительных элементов больше (меньше), чем отрицательных, то заменить нулями нужное число положительных (отрицательных) элементов, чтобы их количество совпадало.
7. Задан целочисленный массив *X* из 20 элементов. Из этого массива переписать в массив *Y* ту последовательность, которая образует арифметическую прогрессию как минимум из пяти членов. Выдать соответствующее сообщение, если таких последовательностей нет.
8. Задан целочисленный массив *X* из 20 элементов. Переписать в массив *Y* те элементы исходного массива, которые строго больше двух своих соседей. Элементы массива *Y* не должны повторяться.
9. Задан целочисленный массив *X* из 20 элементов. Из этого массива переписать в массив *Y* той же размерности подряд все отрицательные элементы, а оставшиеся места заполнить единицами. Расположить элементы образованного массива в порядке убывания.
10. Задан целочисленный массив *X* из 20 элементов. Определить, можно ли из его положительных элементов составить строго возрастающую последовательность.
11. Задан целочисленный массив *X* из 20 элементов, содержащий как четные, так и нечетные числа. Из этого массива переписать в массив *Y* подряд первые пять различных четных элементов. Если таких элементов менее пяти, заполнить оставшиеся позиции в массиве суммой нечетных элементов массива *X*.
12. Задан целочисленный массив *X* из 20 элементов, содержащий группы подряд идущих одинаковых элементов. Поменять местами первую и последнюю группы массива.

13. Задан целочисленный массив *X* из 20 элементов. Определить максимальное количество идущих подряд и упорядоченных по возрастанию положительных чисел.
14. Задан целочисленный массив *X* из 20 элементов. Определить сумму элементов, имеющих четные индексы и являющихся нечетными числами. Если таковых нет, увеличить на единицу все элементы с четными индексами и вывести на экран результирующий массив.
15. Задан массив *X* натуральных чисел из 20 элементов. Удалить из него элементы, являющиеся удвоенными нечетными числами.
16. Задан массив *X* натуральных чисел из 20 элементов. Переписать в массив *Y* элементы, дающие при делении на 7 остаток 1, 2 или 5. Элементы массива *Y* упорядочить по неубыванию. В случае отсутствия таких элементов выдать соответствующее сообщение.
17. Задан целочисленный массив *X* из 20 элементов. Переписать в массив *Y* те элементы, которые равны сумме предшествующих им значений.
18. Задан действительный массив *X* из 20 элементов, содержащий 10 положительных и 10 отрицательных чисел. Переставить элементы массива так, чтобы положительные и отрицательные числа чередовались.
19. Задан целочисленный массив *X* из 20 элементов, среди которых есть повторяющиеся. Переписать в массив *Y* только неповторяющиеся элементы исходного массива.
20. Задан целочисленный массив *X* из 20 элементов, среди которых есть повторяющиеся. Записать в массив *Y* по одному элементу из каждой группы одинаковых значений исходного массива.
21. Задан целочисленный массив *X* из 20 элементов. Получить массив *Y*, в который переписать те положительные элементы массива *X*, которые расположены между двумя отрицательными. Если таких элементов нет, вывести соответствующее сообщение. Элементы массива *Y* не должны повторяться
22. Задан целочисленный массив *X* из 20 элементов. Переписать в массив *Y* наибольшую по длине возрастающую последовательность исходного массива.
23. Задан целочисленный массив *X* из 20 элементов, среди которых есть повторяющиеся. Определить наименьший и наибольший элементы массива. Если они встречаются несколько раз, то оставить их по одному экземпляру, заменив остальные вхождения средним арифметическим наибольшего и наименьшего элементов.

24. Задан целочисленный массив X из 20 элементов, содержащий как положительные, так и отрицательные значения. Переставить элементы в массиве так, чтобы в начале располагались все положительные элементы, а затем все отрицательные. Порядок следования элементов в этих группах должен остаться прежним.
25. Задан целочисленный массив X из 20 элементов. Получить массив Y, в который записать те из элементов исходного массива в порядке следования, которые образуют наиболее длинную возрастающую последовательность.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Дайте определение массива.
2. Как производится доступ к отдельным элементам массива?
3. Что такое указатель? Как он описывается?
4. Что общего у понятий массива и указателя?
5. Как с помощью указателя обратиться к элементу массива?
6. Какие способы заполнения массива значениями вы знаете?
7. Как определяется символьный массив?
8. Что представляет собой строка символов в языке С?
9. Каково внутреннее представление строковых констант?
10. Чем ограничен размер строки символов в языке С?
11. Какие ограничения накладываются на индексы массивов?
12. Может ли быть индекс массива равен значению его размерности?

ЛАБОРАТОРНАЯ РАБОТА № 8

ОБРАБОТКА ДВУМЕРНЫХ МАССИВОВ. ФАЙЛОВЫЙ ВВОД-ВЫВОД. ПРИМЕНЕНИЕ ИТЕРАТИВНЫХ И РЕКУРСИВНЫХ ФУНКЦИЙ

Цель работы: ознакомиться с организацией двумерных массивов в языке C/C++; приобрести практические навыки в файловом вводе-выводе данных; ознакомиться с организацией передачи параметров в функции по ссылке; получить навыки описания рекурсивных функций.

КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Организация многомерных массивов

В языке C/C++ многомерные (в частности двумерные) массивы – это массивы, элементами которых в свою очередь являются другие массивы. Например, конструкция:

```
int  mass[3][5];
```

описывает массив из трех элементов, каждым из которых является массив из пяти элементов целого типа. Фактически это матрица (3×5). Доступ к элементам осуществляется указанием двух индексов:

```
mass[0][1]=25;//  присвоить 25 элементу, находящемуся в
                  // 1-й строке и 2-м столбце
```

Для работы с двумерными массивами, как правило, требуется применение вложенных циклов. В качестве примера рассмотрим ввод элементов массива с клавиатуры:

```
int a[5][5],i,j;
for(i=0;i<5;i++)
  for(j=0;j<5;j++)
    scanf("%d",&a[i][j]);
```

Двумерные массивы можно инициализировать при описании так же как и одномерные. Например:

```
int a[2][3]={
     {1,2,3},
     {4,5,6}
    }
```

Доступ к файлам

Ввод-вывод на верхнем уровне в языке Си осуществляется через ***потоки.*** Поток является файлом или физическим устройством (например, принтером или монитором), которое управляется с помощью указателей на объект типа FILE (определенный в *stdio.h*). Структура FILE содержит различную информацию о потоке, включая текущую позицию потока, указатель на соответствующие буферы и индикаторы ошибки или конца файла. Открывается поток с помощью функции *fopen*:

```
FILE *fopen(char *pathname, char *type);
```

Строка *pathname* – это путь к файлу и его имя. Строка *type* – разрешенный тип доступа (например `"w"` – для записи, `"r"` – для чтения). Функция возвращает указатель на структуру типа FILE, который в дальнейшем передается другим функциям ввода-вывода. Нулевое значение указателя (*NULL*) говорит об ошибке открытия файла.

Для закрытия потока используется функция *fclose*:

```
int fclose(FILE *fp);
```

В дополнение к потокам, создаваемым вызовом *fopen*, три предопределенных (стандартных) потока автоматически открываются всякий раз, когда начинается выполнение программы:

Имя предопределенного потока	Тип	Режим	Описание
stdin	ввод	Текстовый	Стандартный ввод
stdout	вывод	Текстовый	Стандартный вывод
stderr	вывод	Текстовый	Стандартная ошибка

Имя предопределенного потока допустимо указывать в любых функциях ввода-вывода вместо имени пользовательского потока. Закрыть предопределенный поток нельзя. Его можно только переопределить с помощью функции *freopen*, имеющей следующий прототип:

```
FILE *freopen(char *pathname, char *type,
              FILE *fp);
```

Для чтения данных из файла используется функция *fscanf*, являющаяся аналогом функции *scanf*, за исключением того, что первым параметром является указатель на файл. Например,

```
fscanf(fr, "%d", &a);
```

Для записи данных в файл используется функция *fprintf*, являющаяся аналогом функции *printf*, за исключением того, что первым параметром является указатель на файл. Например,

```
fprintf(fw, "y = %d", y);
```

Пример: написать программу, считывающую из файла *in.txt* элементы матрицы *A*(5×5) и записывающую их в файл *out.txt* в виде транспонированной матрицы.

```
#include <stdio.h>
#define N 5 // размерность матрицы
int main(void)
{
int i, j, a[N][N];
FILE *fp;//указатель на файловую структуру
fp=fopen("in.txt", "r"); //открываем файл in.txt
                         //для чтения
/*файл in.txt должен существовать в текущем каталоге и
содержать 25 элементов матрицы a. Далее проверяется
корректность открытия файла*/
if(fp)
{
  for(i=0;i<N;i++)
    for(j=0;j<N;j++)
      fscanf(fp, "%d", &a[i][j]);//считываем
                                 //элементы матрицы
  fclose(fp); //закрываем файл
  fp=fopen("out.txt", "w"); //открываем файл out.txt
                            //для записи
  //записываем элементы массива a в виде
  //транспонированной матрицы в файл out.txt:
  for(i=0;i<N;i++)
   {
     for(j=0;j<N;j++)
       fprintf(fp, "%5d", a[j][i]);
     fprintf(fp, "\n");
   }
  fclose(fp); //закрываем файл
} else printf("Входной файл отсутствует\n");
}
```

Описание функций

Определение функции состоит из ее заголовка и составного оператора тела функции. В заголовке указывается тип функции, ее имя и в скобках список формальных параметров (если они есть). Параметры любых типов (кроме массивов) в обычном случае передаются в функции по значению. Подразумевается, что функция может возвращать результат своей работы, объявленного при ее описании типа, через свое имя (если она не типа *void*). Функция может возвращать значение любого типа, кроме массивов и функций. Но функция также может возвращать указатель на объект любого типа, в том числе и на массив или функцию.

Однако в некоторых случаях, например при необходимости возврата нескольких значений, применяется передача параметров по ссылке. Для этого имеются две возможности:

1. Ссылочные параметры.

Пример: функция инкремента целого числа:

```
void inc(int &x)
{
 x++;
}
……….
// Вызов функции:
int x=1;
inc(x);
```

2. Указатели.

Пример: функция инкремента целого числа:

```
void inc(int *x)
{
 (*x)++;
}
……….
// Вызов функции:
int x=1;
inc(&x);
```

Поскольку имя массива является константным указателем, то он *всегда* передается в функцию по ссылке. В случае одномерного массива в функцию фактически передается адрес первого элемента. Если даже указать при описании такого параметра размерность массива, то она будет игнорироваться, так как нет никакого контроля за тем, выходит ли

при обращении к элементу его вычисленный адрес за границы массива. Поэтому при описании в качестве параметра функции одномерного массива возможны разные варианты записи: с указанием или без указания размерности массива или непосредственно в виде указателя на базовый тип элементов (см. пример ниже).

Двумерный массив представляет собой массив, элементами которого являются одномерные массивы (строки). При вызове функции, имеющей в качестве параметра двумерный массив, будет передаваться указатель на первую строку матрицы. В связи с этим также возможны разные варианты записи параметров: с указанием обеих размерностей массива или только одной (второй, т.е. количества элементов в строке), а также в виде указателя на одномерный массив, размер которого совпадает со второй размерностью матрицы (см. пример ниже).

Если среди параметров, передаваемых в функцию по ссылке, имеются такие, которые не должны изменяться в процессе ее работы, перед ними при описании можно указывать модификатор *const*.

Как было сказано выше, функция в качестве результата может возвращать указатель на функцию. Но также верным является то, что такой указатель может быть параметром функции. Общий синтаксис описания указателя на функцию следующий:

<*тип ф-ии*> (*<*имя указателя*>)(<*список типов парам-в ф-ии*>)

Например, если имеется функция:

```
int sum(int a, int b)
{
  return a+b;
}
```

то указатель, ссылающийся на нее можно описать следующим образом:

```
int (*pf)(int, int)=&sum;
```

Пример: дана целочисленная матрица $A(5\times5)$. Определить массив X из пяти элементов, равных элементам побочной диагонали матрицы A. Элементы матрицы по выбору пользователя либо вводятся с клавиатуры, либо генерируются случайным образом.

```
#include <stdio.h>
#include <stdlib.h> //Содержит прототипы функций для
                    //работы с ГСЧ
#include <conio.h>
#include <time.h> //Содержит прототипы функций для
                  //работы с временем
```

```
#include <locale.h>
#define N 5
// Функция ввода элементов матрицы с клавиатуры:
void input_matr(int a[N][N])
{
   int i,j;
   printf("Введите элементы матрицы A: \n");
   for (i=0;i<N;i++)
     for (j=0;j<N;j++)
       scanf("%d",&a[i][j]);
}
// Функция заполнения матрицы с помощью ГСЧ :
void input_matr_rand(int a[][N])
{
   int i,j;
   srand((unsigned)time(NULL)); //Инициализация ГСЧ
//Функция time возвращает текущее время в секундах
   for (i=0;i<N;i++)
     for (j=0;j<N;j++)
       a[i][j]=rand()%50; // Значения элементов от 0 до 49
}
// Функция вычисления элементов массива X:
int *mas_x(int (*a)[N], int *x,
           void (*pfunc)(int [N][N]))
{
  int i,j;
  // Вызов через указатель одной из двух функций ввода элементов a:
  pfunc(a);
  for (i=0, j=N-1;i<N;i++, j--)
    x[i]=a[i][j];
  return x;
}
// Функция вывода на экран элементов матрицы a и массива x:
void output(const int x[], const int a[][N] )
{
  int i,j;
  printf("Матрица A:\n");
  for (i=0;i<N;i++)
  {
    for (j=0;j<N;j++)
      printf("%5d",a[i][j]);
```

```
    printf("\n");
  }
  printf("Массив X:\n");
  for (i=0;i<N;i++)
    printf("%5d",x[i]);
  printf("\n");
}
int main()
{
  setlocale(LC_CTYPE,"");
  int a[N][N], x[N], c;
  void (*pfunc)(int [N][N]);
  do
  {
    printf("Ввод матрицы:\n1 - с клавиатуры\n2- ГСЧ\n");
    scanf("%d",&c);
  }
  while (c!=1 && c!=2);
  // Присваивание указателю адреса одной из функций:
  switch (c)
  {
    case 1: pfunc=&input_matr;
          break;
    case 2: pfunc=&input_matr_rand;
  }
  output(mas_x(a,x,pfunc),a);
  _getch();
}
```

Рекурсивные функции

Функция называется рекурсивной, если она вызывает саму себя (прямая рекурсия), или вызывает другую функцию, которая в свою очередь вызывает первую (косвенная рекурсия).

Глубина рекурсии должна быть конечной. При выполнении очередного рекурсивного вызова система создает в стеке новые экземпляры всех автоматических переменных функции и ее параметров. Поэтому при большой глубине рекурсии возможно переполнение стека и аварийное завершение работы программы.

Также следует аккуратно обращаться в рекурсивных функциях с глобальными переменными, так как их изменение отразится во всех последующих вызовах.

Пример: составить рекурсивную функцию вычисления факториала:

```
int factorial_recurs(int n)
{
   if (n==1 || n==0)   return 1;
    else return n*factorial_recurs(n-1);
}
```

СОДЕРЖАНИЕ РАБОТЫ

Выбрать алгоритм, составить его блок-схему и программу для решения выбранного варианта задания. Программа должна по выбору пользователя осуществлять ввод исходной матрицы с клавиатуры или из файла. Для этого программа должна содержать две соответствующие функции, указатель на одну из которых необходимо передавать в функцию для вычисления элементов массива *X*. Данная функция должна вызывать через указатель одну из функций ввода элементов матрицы, производить вычисление элементов массива *X* в соответствии с заданием и возвращать указатель на этот массив. Кроме того, программа должна содержать функцию для вывода на экран и в файл исходной матрицы и результирующего массива, а также рекурсивную функцию определения в соответствии с заданием величины *Y*.

В программе не должно быть глобальных переменных.

ВАРИАНТЫ ЗАДАНИЯ

1. Дана матрица A(5×5). Определить массив *X* из 5 элементов, каждый из которых равен сумме элементов соответствующей строки верхней треугольной матрицы. Определить величину *Y*, как наибольший из элементов массива *X*.
2. Дана матрица A(5×5). Определить массив *X* из 5 элементов, каждый из которых равен произведению элементов соответствующего столбца нижней треугольной матрицы. Определить величину *Y*, как сумму положительных элементов массива *X*.
3. Дана матрица A(5×5). Определить массив *X* из 5 элементов, каждый из которых равен наибольшему элементу соответствующей строки матрицы. Определить величину *Y*, как наименьший из положительных элементов массива *X*.

4. Дана матрица A(5×5). Определить массив *X* из 5 элементов, каждый из которых равен 1, если произведение элементов соответствующего столбца больше нуля и –1 в противном случае. Определить величину *Y*, как количество повторений 1 среди элементов массива *X*.
5. Дана матрица A(5×5). Определить массив *X* из 5 элементов, каждый из которых равен наименьшему из элементов соответствующего столбца матрицы. Определить величину *Y*, как количество нечетных элементов, расположенных перед наибольшим из элементов массива *X*.
6. Дана матрица A(5×5). Определить массив *X* из 5 элементов, каждый из которых равен 1, если количество положительных элементов в соответствующей строке больше количества отрицательных и –1 в противном случае. Определить величину *Y*, как количество четных элементов в первой строке матрицы A.
7. Дана матрица A(5×5). Определить массив *X* из 5 элементов, каждый из которых равен среднему арифметическому наибольшего и наименьшего из элементов соответствующего столбца матрицы. Определить величину *Y*, как сумму элементов, расположенных перед наименьшим элементом массива *X*.
8. Дана матрица A(5×5). Определить массив *X* из 5 элементов, каждый из которых равен 1, если в соответствующей строке элемент главной диагонали больше элемента побочной и –1 в противном случае. Определить величину *Y*, как количество нечетных элементов в первом столбце матрицы A.
9. Дана матрица A(5×5). Определить массив *X* из 5 элементов, каждый из которых равен 1, если элементы упорядочены по возрастанию или по убыванию и –1 в противном случае. Определить величину *Y*, как среднее арифметическое наибольшего и наименьшего элемента главной диагонали матрицы A.
10. Дана матрица A(5×5). Определить массив *X* из 5 элементов, каждый из которых равен первому отрицательному элементу соответствующей строки матрицы или нулю, если все элементы строки положительны. Определить величину *Y*, как индекс первого отрицательного элемента массива *X*.
11. Дана матрица A(5×5). Определить массив *X* из 5 элементов, каждый из которых равен 1, если все элементы соответствующей строки положительны и –1 в противном случае. Определить величину *Y*, как 1, если элементы первой строки матрицы образуют арифметическую прогрессию и 0 в противном случае.

12. Дана матрица A(5×5). Определить массив X из 5 элементов, каждый из которых равен соответствующему элементу столбца с наибольшей среди других столбцов суммой положительных элементов. Определить величину Y, как 1, если элементы массива X образуют последовательность Фибоначчи ($f_1 = f_2 = 1$, $f_i = f_{i-1} + f_{i-2}$ для $i > 2$) и 0 в противном случае.
13. Дана матрица A(5×5). Определить массив X из 5 элементов, каждый из которых равен 1, если наименьший элемент соответствующей строки положителен и –1 в противном случае. Определить величину Y, как наибольший из индексов элементов массива X, равных 1.
14. Дана матрица A(5×5). Определить массив X из 5 элементов, каждый из которых равен сумме элементов соответствующей строки, если они все либо положительны либо отрицательны, и нулю в противном случае. Определить величину Y, как сумму элементов массива X, расположенных после первого нулевого элемента.
15. Дана матрица A(5×5). Определить массив X из 5 элементов, каждый из которых равен среднему арифметическому элементов строки и столбца, на пересечении которых находится соответствующий элемент побочной диагонали. Определить величину Y, как произведение четных элементов, расположенных после наименьшего элемента массива X.
16. Дана матрица A(5×5). Определить массив X из 5 элементов, каждый из которых равен количеству вхождений в соответствующую строку наибольшего из элементов матрицы. Определить величину Y, как количество нулевых элементов массива X.
17. Дана матрица A(5×5). Определить массив X из 5 элементов, каждый из которых равен сумме элементов того столбца, в котором находится первый положительный элемент соответствующей строки, и нулю, если все элементы строки неположительны. Определить величину Y, как количество отрицательных элементов, расположенных перед наибольшим элементом массива X.
18. Дана матрица A(5×5). Определить массив X из 5 элементов, каждый из которых равен 1, если в соответствующем столбце есть возрастающая подпоследовательность из трех элементов и нулю в противном случае. Определить величину Y, как произведение нечетных элементов, расположенных перед первым встретившимся четным элементом массива X.

19. Дана матрица A(5×5). Определить массив *X* из 5 элементов, каждый из которых равен 1, если наименьший из элементов соответствующей строки совпадает с наименьшим элементом матрицы и –1 в противном случае. Определить величину *Y*, как сумму четных элементов первой строки матрицы, расположенных после первого нечетного элемента
20. Дана матрица A(5×5). Определить массив *X* из 5 элементов, каждый из которых равен элементам того столбца, в котором находятся наибольший и наибольший по модулю элементы матрицы, или элементам побочной диагонали, если они находятся в разных столбцах. Определить величину *Y*, как произведение отрицательных элементов массива *X*, расположенных после первого положительного элемента.
21. Дана матрица A(5×5). Определить массив *X* из 5 элементов, каждый из которых равен сумме элементов соответствующей строки, предшествующих первому в ней отрицательному элементу. Определить величину *Y*, как количество повторений наименьшего элемента в первой строке матрицы.
22. Дана матрица A(5×5). Определить массив *X* из 5 элементов, каждый из которых равен 1, если сумма модулей элементов соответствующего столбца больше наибольшего по модулю элемента матрицы и –1 в противном случае. Определить величину *Y*, как сумму положительных элементов первой строки матрицы, расположенных после первого нулевого элемента.
23. Дана матрица A(5×5). Определить массив *X* из 5 элементов, каждый из которых равен произведению элементов соответствующего столбца, расположенных за первым в нем отрицательным элементом. Определить величину *Y*, как количество отрицательных элементов первой строки матрицы, имеющих нечетные номера столбцов.
24. Дана матрица A(5×5). Определить массив *X* из 5 элементов, каждый из которых равен 1, если количество четных элементов в соответствующей строке больше количества нечетных и –1 в противном случае. Определить величину *Y*, как произведение положительных элементов, расположенных после набольшего элемента первой строки матрицы.
25. Дана матрица A(5×5). Определить массив *X* из 5 элементов, каждый из которых равен 1, если наибольший по модулю элемент соответствующей строки совпадает с наименьшим по модулю элементом побочной диагонали и –1 в противном случае. Определить величину *Y*, как сумму элементов первой строки

матрицы, расположенных между наибольшим и наименьшим элементом.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Каковы в языке C/C++ принципы размещения в памяти многомерных массивов? Как производится их описание?
2. Как производится обращение к элементам многомерного массива?
3. Какими способами можно произвести заполнение многомерного массива элементами?
4. Как осуществляется файловый ввод-вывод в языке C?
5. В каком файле определены прототипы функций ввода-вывода верхнего уровня?
6. Какие функции осуществляют открытие и закрытие файла?
7. Какие функции предназначены для форматированного ввода-вывода данных?
8. В чем заключается различие в принципах передачи в функцию параметров по значению и по ссылке?
9. Какие вы знаете способы передачи параметров по ссылке в языке C/C++?
10. Каким образом передаются в функции массивы?
11. Возможен ли возврат функцией таких типов данных, как структуры и объединения?
12. Возможен ли возврат функцией массива?
13. Назовите преимущества и недостатки рекурсивных функций по сравнению с итеративными.
14. В каком случае задача может иметь рекурсивное решение?
15. Каков механизм вызова рекурсивной функции?
16. Какие условия должны выполняться при описании рекурсивных функций?
17. Как описываются функции с косвенной рекурсией?

ЛАБОРАТОРНАЯ РАБОТА № 9

ПОБИТОВЫЕ ОПЕРАЦИИ ЯЗЫКА С/С++

Цель работы: получение навыков использования побитовых операций при работе с целочисленными объектами.

КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Побитовые операции производятся над всеми разрядами (битами) двоичного представления операндов. Побитовые операции применимы только к тем переменным и выражениям, которые имеют целочисленное внутреннее представление (т.е. к целым и символьным). Если операции производятся над операндами знакового типа, то необходимо учитывать, что отрицательные значения представляются в дополнительном коде и старший (знаковый) бит также участвует во всех операциях. В языке С/С++ имеются следующие побитовые операции (в порядке убывания приоритета):

1. **Побитовое отрицание** (~) – производит побитовую инверсию двоичного представления операнда.
   Пример: побитовое отрицание числа 250 (~250):
   Исходное представление числа 250 (беззнаковое, однобайтное):

1	1	1	1	1	0	1	0

   Результат:

0	0	0	0	0	1	0	1

   Или 5_{10}.

2. **Сдвиг** на определенное число разрядов: влево (<<) и вправо (>>).
   Пример:
   а) сдвиг числа 48 на два разряда влево (48<<2):
   Исходное представление числа 48 (беззнаковое, однобайтное):

0	0	1	1	0	0	0	0

   Результат:

1	1	0	0	0	0	0	0

   Или 192_{10}. Как видно сдвиг числа влево на *n* разрядов приводит к его умножению на 2^n;

   б) сдвиг числа 48 на два разряда вправо (48>>2):

Исходное представление числа 48 (беззнаковое, однобайтное):

0	0	1	1	0	0	0	0

Результат:

0	0	0	0	1	1	0	0

Или 12_{10}. Как видно сдвиг числа вправо на *n* разрядов приводит к его делению на 2^n.

Сдвиг не является циклической операцией и если происходит выход какого-либо бита за пределы разрядной сетки, то его значение теряется. Освобождающиеся при сдвиге разряды заполняются нулями. Если производится сдвиг вправо операнда знакового типа, то дополнение слева производится содержимым старшего бита.

3. **Побитовое И** (&) – производится над двумя операндами и если оба соответствующих бита равны единице, то результат – единица, в противном случае – ноль.
   Пример: 202 & 83:
   202:

1	1	0	0	1	0	1	0

   &
   83:

0	1	0	1	0	0	1	1

   Результат:

0	1	0	0	0	0	1	0

   Или 66_{10}.

4. **Побитовое исключающее ИЛИ** (^) – производится над двумя операндами и если оба соответствующих бита равны единице или нулю, то результат – ноль, в противном случае – единица.
   Пример: 202 ^ 83:
   202:

1	1	0	0	1	0	1	0

   ^
   83:

0	1	0	1	0	0	1	1

   Результат:

1	0	0	1	1	0	0	1

   Или 153_{10}.

5. **Побитовое ИЛИ** (|) – производится над двумя операндами и если оба соответствующих бита равны нулю, то результат – нуль, в противном случае – единица.
   Пример: 202 | 83:
   202:

1	1	0	0	1	0	1	0

   |
   83:

0	1	0	1	0	0	1	1

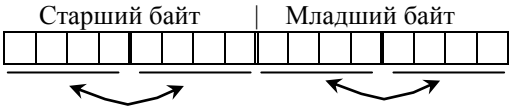
   Результат:

1	1	0	1	1	0	1	1

   Или 219_{10}.

Обычно побитовые операции применяются в тех случаях, когда необходимо выделить из двоичного представления числа содержимое некоторых его разрядов. Для этого применяют операцию *побитового И* между числом и специально подготовленным операндом, называемым ***маской***. Маска содержит единицы в тех разрядах, содержимое которых необходимо выделить и нули в остальных. В дальнейшем результат операции может обрабатываться другими побитовыми операциями в соответствии с задачей. В программе удобно записывать значение маски восьмеричными или шестнадцатеричными константами.

Пример: написать программу, обменивающую в коротком целом неотрицательном числе полубайты отдельно в младшем и старшем байтах.




```
#include <stdio.h>
#include <conio.h>
#include <locale.h>
int main (void)
{
  setlocale(LC_ALL,"");
  unsigned short  n,mask1,mask2,mask3,ml,st;
  printf("Введите исходное число:\n");
  scanf("%hx",&n);
  //Маска для правого полубайта младшего байта:
  mask1=0xf;
```

```
    //Маска для левого полубайта младшего байта:
    mask2=0xf0;
    //Маска для старшего байта:
    mask3=0xff00;
   //Следующий цикл выполнится два раза (по размеру переменной):
    for(int i=0;i<sizeof(n);i++)
    {
      //Выделяем правый полубайт преобразуемого байта:
      ml=n&mask1;
      //Выделяем левый полубайт преобразуемого байта:
      st=n&mask2;
      //Объединяем противоположный байт и сдвинутые
      //выделенные полубайты:
      n=(n&mask3)|(ml<<4)|(st>>4);
      //Получаем маску для правого полубайта
      // следующего (старшего)байта:
      mask1<<=8;
      //Получаем маску для левого полубайта
      // следующего (старшего)байта:
      mask2<<=8;
      //Получаем маску для преобразованного (младшего) байта:
      mask3>>=8;
    }
    printf("Преобразованное число=%hx\n",n);
    _getch();
  }
```

Фактически перечисленные выше действия по преобразованию числа можно заменить одним оператором:

```
n=((n&0xf)<<4)       //Выделяем и сдвигаем правый полубайт
                     //младшего байта
|((n&0xf0)>>4)       //Выделяем и сдвигаем левый полубайт
                     //младшего байта
|((n&0xf00)<<4)      //Выделяем и сдвигаем правый полубайт
                     //старшего байта
|((n&0xf000)>>4);    //Выделяем и сдвигаем левый полубайт
                     //младшего байта
```

СОДЕРЖАНИЕ РАБОТЫ

Выбрать алгоритм, составить его блок-схему и программу для решения выбранного варианта задания. Во всех вариантах предполагается, что размер короткого целого числа составляет 2 байта, а длинного – 4 байта. При необходимости ввод исходного значения и вывод результата может производиться в восьмеричном или шестнадцатеричном представлении.

ВАРИАНТЫ ЗАДАНИЯ

1. Дано короткое целое неотрицательное число. Произвести сортировку его двоичного представления так, чтобы все единицы располагались в старших разрядах, а нули в младших.
2. Дано длинное целое неотрицательное число. Получить два коротких целых неотрицательных числа, одно из которых заполнено содержимым битов исходного числа с четными номерами, а другое – с нечетными.
3. Дано длинное целое неотрицательное число. Получить короткое целое неотрицательное число, биты которого начиная с младших содержат результат побитового И битов исходного числа соответственно с номерами 31 и 0, 30 и 1 и т.д.
4. Дано короткое целое неотрицательное число. Выполнить циклический сдвиг его двоичного представления на k битов влево.
5. Дано длинное целое неотрицательное число. Выполнить циклический сдвиг его шестнадцатеричного представления на k цифр вправо.
6. Дано короткое целое неотрицательное число. Определить, является ли его двоичное представление палиндромом.
7. Дано короткое целое неотрицательное число. Преобразовать старший байт числа таким образом, чтобы его двоичное представление стало палиндромом.
8. Дано короткое целое неотрицательное число. Определить в его двоичном представлении количество соседств двух нулей и двух единиц.
9. Дано длинное целое неотрицательное число. Определить количество различных цифр в его шестнадцатеричном представлении.
10. Дано короткое целое неотрицательное число. Сформировать другое число, которое будет состоять только из четных восьмеричных цифр исходного числа.

11. Дано длинное целое неотрицательное число. Заменить в его шестнадцатеричном представлении все нечетные цифры на нули.
12. Дано длинное целое неотрицательное число. Определить количество вхождений в него минимальной из цифр его шестнадцатеричного представления.
13. Дано короткое целое неотрицательное число. Произвести в его двоичном представлении обмен битов с номерами 0 и 1, 2 и 3, 4 и 5 и т.д.
14. Дано короткое целое неотрицательное число. Выполнить инверсию двоичного представления входящих в его состав четных восьмеричных цифр.
15. Дано длинное целое неотрицательное число. Удалить из его шестнадцатеричной записи цифры, меньшие 5.
16. Дано короткое целое неотрицательное число. Определить в его двоичном представлении максимальное количество расположенных рядом единиц.
17. Даны два коротких целых неотрицательных числа. Определить совпадают ли множества восьмеричных цифр, из которых они состоят.
18. Дано короткое целое неотрицательное число. Просматривая его двоичное представление парами бит, начиная с младших, произвести замену каждой комбинации «01» на «11», а «10» на «00».
19. Дано длинное целое неотрицательное число. Заменить в его шестнадцатеричном представлении старшую цифру максимальной из цифр числа, а младшую – минимальной.
20. Дано короткое целое неотрицательное число. Заменить каждую входящую в его состав шестнадцатеричную цифру на количество единиц, имеющихся в ее двоичном представлении (например, цифру F надо заменить на 4).
21. Дано короткое целое неотрицательное число. Определить в его двоичном представлении длину максимальной последовательности чередующихся нулей и единиц.
22. Дано короткое целое неотрицательное число. Заменить его старший бит результатом побитового исключающего ИЛИ всех предшествующих ему бит.
23. Дано длинное целое неотрицательное число, шестнадцатеричное представление которого не содержит ни одного нуля. Просматривая число, начиная с младших разрядов, оставить в нем только первые вхождения каждой цифры. Остальные вхождения заменить нулями.

24. Дано короткое целое неотрицательное число. Определить количество составляющих его шестнадцатеричных цифр, содержащих в двоичном представлении равное количество нулей и единиц (например, 3, 5 и др.).
25. Дано короткое целое неотрицательное число. Записать в две старшие цифры его шестнадцатеричного представления количество битов исходного числа, содержащих единицы.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Перечислите побитовые операции, реализованные в языке C/C++ в порядке убывания приоритета.
2. В чем отличие операции побитового ИЛИ от операции арифметического сложения?
3. В чем отличие операций логического и побитового И?
4. Какие типы операндов допустимы в побитовых операциях?
5. Есть ли разница при выполнении операций сдвига целых знаковых и беззнаковых типов?
6. Существуют ли в языке C/C++ операции циклического сдвига?
7. Для чего, как правило, применяются побитовые операции?

ЛАБОРАТОРНАЯ РАБОТА № 10

ОБРАБОТКА ДИНАМИЧЕСКИХ МАССИВОВ И СВЯЗНЫХ СПИСКОВ ДАННЫХ

Цель работы: ознакомиться с организацией многомерных динамических массивов в языке C/C++; приобрести практические навыки в применении односвязных линейных списков при обработке данных из внешних файлов.

КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Динамическое распределение памяти дает возможность программисту более эффективно использовать ресурсы компьютера. В частности, оно позволяет создавать динамические переменные прямо во время работы программы и уничтожать их, когда в них отпала надобность. Такие переменные располагаются в так называемой динамической области памяти (*heap*) и доступ к ним производится с помощью указателей. Для создания динамических переменных можно использовать как средства стандартной библиотеки языка C, так и средства языка C++. При этом в динамической памяти могут располагаться как переменные простых типов, так и структурированных. В свою очередь структуры данных могут быть статическими (со смежным расположением элементов в памяти) и динамическими (с произвольным расположением элементов, связанных непосредственно через указатели). Примером структур первого типа могут служить динамические массивы, второго – односвязные линейные списки. Рассмотрим более подробно процессы их создания и обработки.

Динамические массивы

Для распределения динамической памяти в языке C++ имеется операция *new*. Например, для создания целочисленного динамического одномерного массива в общем случае из *n* элементов, можно использовать следующий фрагмент:

```
int *p, n=20;
p= new int[n];
```

Созданный динамический массив ничем неявно не инициализируется. Обращение к элементу такого массива производится вполне привычным образом, например:

```
p[15]=25;
```

Создание динамического двумерного массива с заранее неизвестными числом строк и столбцов проводится в два этапа. Сначала создается одномерный массив указателей, каждый из элементов которого ссылается на соответствующую, отдельно создаваемую строку матрицы.

```
int **p, i, n=5, m=10;
p= new int *[n];
for(i=0;i<n;i++)
   p[i]=new int[m];
```

Освобождение памяти, занимаемой массивом вне зависимости от числа измерений, производится операцией *delete []*. Например:

```
delete []p;
```

Односвязные линейные списки

Приведенные выше примеры по созданию динамических массивов показали большое удобство их применения при обработке данных заранее неизвестного объема. Однако и в таком подходе могут возникать некоторые неудобства. Например, если был создан динамический массив, и при дальнейшем выполнении программы оказалось, что необходимо увеличить его размер. Фактически нужно создать другой массив большего размера и переписать в него все элементы первого массива, а затем удалить его.

Очевидно, что в таких случаях более целесообразно применять динамические структуры данных, в которых элементы можно добавлять или удалять в любой момент по мере надобности. Поскольку элементы таких структур располагаются в памяти не смежно, то они должны содержать в себе указатели на другие элементы. Таким образом, содержимое каждого элемента разделяется на две части:

1. информационное поле, в котором содержатся те данные, ради которых и создается структура;
2. поле связок, в котором содержатся один или несколько указателей, связывающий данный элемент с другими элементами структуры;

Поскольку каждый элемент структуры может содержать в себе произвольное количество полей данных и указателей, то для ее реализации в языке C/C++ наиболее подходит такой тип данных, как самоадресуемая структура.

Классификация динамических структур данных производится по количеству указателей, которые содержит каждый элемент и по характеру связей между элементами. Наиболее простым видом является односвязный линейный список, каждый элемент которого содержит только один указатель на следующий элемент. Перемещение по списку производится последовательно, начиная с первого элемента, называемого головой списка. У последнего указатель на следующий элемент равен нулю. Для большей наглядности список удобно представлять в графической форме (рис. 5.1).

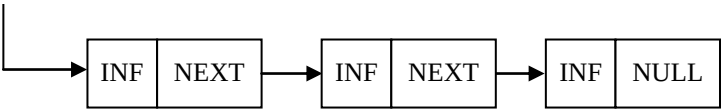


Рис. 5.1. Графическое представление односвязного линейного списка

Для примера рассмотрим создание списка, каждый элемент которого содержит целое число, введенное с клавиатуры. Заполнение списка происходит до тех пор, пока пользователь не введет ноль.

```
struct LIST
{  int n; //Информационное поле
     LIST *next; //Указатель на след. элемент списка
};
...................
LIST *lst=NULL; //Указатель для хранения головы списка
LIST *p; //Вспомогательный указатель
int temp; //Вспомогательная переменная для хранения
            //вводимых чисел
cin>>temp;
if (temp) //Проверка того, что первое значение не ноль
{
   lst=new LIST; //Выделение памяти под голову списка
   lst->n=temp; //Заполнение информационного поля
   p=lst; //Заносим адрес головы списка во вспом. указатель
```

```
    cin>>temp;
    while (temp)
    {
       //Если очередное значение не ноль создаем новый элемент:
       p->next=new LIST;
       p=p->next; //p теперь ссылается на созданный элемент
       p->n=temp; //Заполнение информационного поля
       cin>>temp;
    }
    p->next=NULL; //Признак окончания списка
}
```

Результатом работы такого фрагмента программы будет созданный список из произвольного числа элементов. Причем, как видно, указатель *lst* используется только при формировании первого элемента. В дальнейшем вся работа со списком производится через вспомогательную переменную *p*. Это связано с тем, что нельзя потерять адрес первого элемента, так как это единственная «точка входа» в список. В дальнейшем любая обработка списка начинается с обращения к его голове. Любая функция, выполняющая такую обработку, будет получать в качестве параметра адрес головы списка, и возвращать его в случае изменения.

Основными операциями со списками являются:

1. Вставка элемента в список. Новый элемент может вставляться в начало списка, в середину или в конец. Последний вариант фактически реализуется при создании списка.
2. Удаление элемента из списка. Здесь тоже возможны разные варианты. Общим является то, что новая связь устанавливается в обход удаляемого элемента, после чего происходит освобождение занимаемой им памяти.
3. Перестановка элементов списка. Такая операция, например, может понадобиться при выполнении сортировки списка. Переставляться могут только соседние элементы. При этом сами данные остаются на месте. Происходит только коррекция указателей.
4. Слияние двух списков в один. В этом случае в последнем элементе указатель на следующий элемент устанавливается равным адресу головы второго списка.
5. Копирование части списка в другой список. Во втором списке просто создаются новые элементы, информационные поля которых заполняются данными из первого списка.

Для примера рассмотрим выполнение наиболее общих случаев операций вставки и удаления элемента в список, а также перестановки двух соседних элементов. Во всех трех приведенных ниже фрагментах будем полагать, что указатели *prev*, *p1*, *p2* содержат адреса трех первых элементов списка.

Вставку нового элемента между элементами с адресами *p1* и *p2* можно выполнить с помощью следующего фрагмента:

```
LIST *prev=lst, *p1=prev->next, *p2=p1->next;
p1->next=new LIST; //Выделение памяти под новый элемент
p1->next->next=p2; //Установление связи со след. элементом
p1->next->n=...    //Заполнение информационного поля
                   //каким-либо значением
```

Для удаления элемента, адресуемого указателем *p1* можно применить следующий фрагмент:

```
LIST *prev=lst, *p1=prev->next, *p2=p1->next;
prev->next=p2; //Установление связи в обход удаляемого элем.
delete p1;     //Освобождение памяти
```

Перестановку элементов, адресуемых указателями *p1* и *p2* можно произвести с помощью следующего фрагмента:

```
LIST *prev=lst, *p1=prev->next, *p2=p1->next;
prev->next=p2; //Теперь вторым будет элемент с адресом p2
p1->next=p2->next; //Теперь элемент с адресом p1 будет
                   //связан с четвертым элементом
LIST *temp=p1; //Сохранение адреса третьего элемента
p1=p2; //В указатель p1 заносится адрес второго элемента
p1->next=temp; //Связывание второго и третьего элементов
p2=temp; //В  p2 заносится адрес третьего элемента
```

После выполнения всех процедур обработки списка, его можно удалить, освободив занимаемую им память. Для этого нужно продвигаться вперед по списку, удаляя его элементы, до тех пор, пока не встретится нулевой указатель:

```
p1=lst;
while (p1) //Пока p1 не станет равным нулю
{
    p2=p1->next; //Запоминаем адрес следующего элемента
     delete p1;
    p1=p2; //Переходим к следующему элементу
}
```

Достаточно часто, списки используются для обработки файлов, так как они могут иметь самые разные и заранее неизвестные размеры. Проиллюстрируем такой вариант использования односвязных линейных списков следующим примером.

Пример: дан текстовый файл, компонентами которого являются целые числа. Удалить из него все числа, меньшие заданного *n*. Оставшиеся числа записать в обратном порядке. Для обработки файла использовать односвязный линейный список, каждый элемент которого должен содержать одно число из файла.

```
#include <fstream>
#include <iostream>
#include <locale.h>
#include <stdlib.h>
using namespace std;
struct LIST {
int number;
LIST *next;
};
//Функция чтения данных и занесения их в создаваемый список:
LIST *read_list(LIST *lst)
{
  LIST *p;
  int t;
  ifstream f("1.txt");
  if (f) //Если файл существует
  {
      f>>t;
      if (!f.eof())//Если файл не пуст
      {
         lst=new LIST;
         p=lst;
         p->number=t;
         f>>t;
         while (!f.eof())//Пока не конец файла
          {
               p->next=new LIST;
               p=p->next;
               p->number=t;
               f>>t;
         }
         p->next=NULL;
```

```
     }
     else
           cout<<"Файл пустой"<<'\n';
  }
  else
  {
       cout<<"Файл отсутствует"<<'\n';
       exit(1);
  }
  return lst; //Возврат указателя на голову созданного списка
}
//Функция удаления элементов с полем number, меньшим n:
LIST *del_element(LIST *lst, int n)
{
 LIST *p1=lst,*p2=p1->next, *prev=lst;
 while (p1)
 {
   if(p1->number<n) //если элемент нужно удалять
   {
      delete p1;
      if(p1==lst) //удаленный элемент был головой списка?
       {
            //если да, то головой становится второй элемент:
             lst=p2;
            prev=lst;
            p1=lst;
      }
      else
      {
             //если нет, то устанавливаем связь в обход
             //удаленного элемента:
            prev->next=p2;
            p1=prev;
      }
   }
   else prev=p1; //если текущий элемент не удалялся, то
                 //он становится предыдущим
   //если удалялась не голова списка, то p1 перемещается дальше:
   if (p1!=p2) p1=p1->next;
   //если не достигнут конец списка, то p2 перемещается дальше:
   if (p2) p2=p2->next;
```

```
 }
 return lst;
}
//Функция записи элементов в файл в обратном порядке
// и освобождения памяти, занимаемой списком:
void write_inverse_list(LIST *lst)
{
  ofstream f;
  LIST *p1=lst,*p2, *prev;
  f.open("2.txt");
  while (p1) // внешний цикл будет работать, пока из списка
  {           // не удалятся все элементы
    prev=lst;
    p2=p1->next;
    while (p2) // передвигаемся по списку до конца
    {
      prev=p1;
      p1=p1->next;
      p2=p2->next;
    }
    f<<p1->number<<' '; // вывод в файл информ. поля
                        // последнего элемента
    delete p1;
    if(p1==lst) lst=NULL; // если p1 голова списка, то
                          // обнуляем lst
    else prev->next=NULL; // иначе у предпоследнего элем.
                          // обнуляем указатель на след.
    p1=lst; // в p1 снова заносим адрес головы списка или ноль
  }
  f.close();
}
int main()
{
  LIST *lst=NULL;
  int n;
  setlocale(LC_CTYPE,"");
  cout<<"Введите число n"<<'\n';
  cin>>n;
  lst=read_list(lst);
  lst=del_element(lst,n);
  write_inverse_list(lst);}
```

СОДЕРЖАНИЕ РАБОТЫ

Задание А

Выбрать алгоритм, составить его блок-схему и программу, выполняющую создание и обработку двумерного динамического массива, в соответствии со своим вариантом задания. Во всех вариантах предполагается, что размерность массива задается на этапе выполнения пользователем. Элементы матрицы вводятся с клавиатуры. На экран выводится исходная матрица и результаты работы программы.

ВАРИАНТЫ ЗАДАНИЯ

1. Дана матрица A($n\times n$). Найти сумму элементов верхней треугольной матрицы (относительно побочной диагонали). Заменить ее значением нижнюю треугольную матрицу.
2. Дана матрица A($n\times n$). Найти произведение элементов нижней треугольной матрицы (относительно побочной диагонали). Заменить его значением верхнюю треугольную матрицу.
3. Дана матрица A($n\times n$). Найти наибольшие элементы каждой строки матрицы. В каждой строке заменить нулями элементы, расположенные левее наибольшего.
4. Дана матрица A($n\times n$). Найти наибольшую из сумм элементов каждой строки. Заменить строкой с наибольшей суммой элементов остальные строки.
5. Дана целочисленная матрица A($n\times n$). Определить для каждой строки матрицы элементы, встречающиеся по одному разу.
6. Дана матрица A($n\times n$). Зеркально отразить элементы матрицы относительно главной диагонали.
7. Дана матрица A($n\times n$), все элементы которой различны. Найти наибольший из элементов матрицы. Произвести циклический сдвиг сначала строки, а затем столбца, в которых он находится так, чтобы переместить элемент в центр матрицы.
8. Дана матрица A($n\times n$). Найти количество положительных элементов в каждом из столбцов. Поменять местами столбцы с наибольшим и наименьшим количеством положительных элементов.
9. Дана матрица A($n\times n$), состоящая из ненулевых элементов. Упорядочить по возрастанию элементы каждого из столбцов. Если имеются несколько столбцов, содержащих одинаковый

набор элементов, то оставить без изменений только столбец с младшим индексом, заполнив остальные нулями.

10. Дана матрица A($n\times n$). Определить такие элементы матрицы, которые строго меньше всех своих соседей, в том числе и в разных строках.
11. Дана матрица A($n\times n$). Расположить строки матрицы по возрастанию сумм их элементов.
12. Дана матрица A($n\times n$). Переставляя ее строки и столбцы добиться того, чтобы наименьший элемент оказался в левом верхнем углу.
13. Дана матрица A($n\times n$). Найти в каждой строке такие элементы, слева от которых находятся только меньшие, а справа только большие значения.
14. Дана матрица A($n\times n$). Расположить столбцы матрицы по убыванию их наибольших элементов.
15. Дана матрица A($n\times n$). Определить пару строк с наибольшим скалярным произведением.
16. Дана матрица A($n\times n$). Определить является ли она симметричной относительно побочной диагонали.
17. Дана матрица A($n\times n$). Найти среднее арифметическое элементов каждой из строк и номера тех из них, у которых эти значения совпадают.
18. Дана матрица A($n\times n$). В каждой строке переместить нулевые элементы в ее начало, не изменяя порядок следования остальных.
19. Дана матрица A($n\times n$). Поменять в каждой строке местами наибольший и наименьший элементы.
20. Дана матрица A($n\times n$). Определить те элементы, расположенные выше главной диагонали, которые больше каждого из элементов, расположенных ниже нее. Если таких элементов нет, то выдать соответствующее сообщение.
21. Дана матрица A($n\times n$). Удалить из матрицы строку и столбец, содержащие наибольший элемент.
22. Дана матрица A($n\times n$). Переставить строки матрицы таким образом, чтобы элементы первого столбца были упорядочены по неубыванию.
23. Дана матрица A($n\times n$). Найти номера строк и столбцов, элементы которых образуют возрастающую последовательность.
24. Дана матрица A($n\times n$). Найти значения трех наибольших элементов матрицы и их индексы.
25. Дана матрица A($n\times n$), состоящая из ненулевых элементов. Переставить ее столбцы таким образом, чтобы левее

располагались те из них, в которых наименьший элемент отрицательный, а правее – в которых положительный. Порядок следования столбцов в каждой группе оставить прежним.

Задание Б

Выбрать алгоритм, составить его блок-схему и программу для решения своего варианта задания. Во всех вариантах предполагается для размещения в памяти содержимого файлов использовать односвязные линейные списки.

ВАРИАНТЫ ЗАДАНИЯ

1. Дан текстовый файл, компонентами которого являются целые числа. Вычислить среднее арифметическое всех компонент и удалить элементы, большие вычисленного значения. Перед каждым отрицательным элементом вставить положительное значение, равное ему по модулю.
2. Дан текстовый файл, компонентами которого являются положительные целые числа, меньшие 1000. Изменить содержимое файла таким образом, чтобы каждая строка имела структуру: *число – значение прописью*. В конец файла добавить числовое значение суммы всех компонент.
3. Дан текстовый файл, компонентами которого являются целые числа, не равные нулю. Файл содержит одинаковое количество положительных и отрицательных чисел. Преобразовать файл таким образом, чтобы не было соседств двух чисел с одним знаком.
4. Дан текстовый файл, компонентами которого являются целые числа, не равные нулю. Числа в файле идут в следующем порядке: десять положительных, десять отрицательных, десять положительных, десять отрицательных и т.д. Переписать компоненты таким образом, чтобы они располагались в следующем порядке: пять отрицательных, пять положительных и т.д.
5. Дан текстовый файл, который содержит группы символов, разделенные пробелами. Если группа представляет собой целое число, заменить ее группой символов, соответствующих удвоенному значению числа. Если число вещественное, то заменить его на округленное целое значение.

6. Дан текстовый файл, содержащий одинаковое количество четных и нечетных чисел. Разместить в начале файла все нечетные числа, затем четные. Каждая группа должна быть отсортирована по возрастанию.
7. Даны два текстовых файла, содержащие целые числа, отсортированные в порядке возрастания. Получить третий, результирующий файл, содержащий элементы первых двух файлов, расположенные в порядке убывания.
8. Даны два текстовых файла. Проверить, совпадают ли их множества символов. В случае расхождения дополнить первый файл отсутствующими в нем символами.
9. Дан текстовый файл. Группы символов, разделенные пробелами, будем называть словами. Удалить из файла все однобуквенные слова и лишние пробелы.
10. Дан текстовый файл, компонентами которого являются целые числа. Не упорядочивая содержимое файла, удалить из него повторяющиеся элементы, оставив только первые вхождения.
11. Дан текстовый файл. Группы символов, разделенные пробелами, будем называть словами. Удалить из файла все слова, не являющиеся палиндромами.
12. Даны текстовый файл и натуральное число *n*. Удалить из файла все строки, содержащие более 50 символов. Оставшиеся строки циклически сдвинуть на *n* позиций вниз.
13. Дан текстовый файл. Строки этого файла расположить в порядке убывания их длины и удалить пять самых коротких из них.
14. Даны текстовый файл и строка *s*. Группы символов, разделенные пробелами, будем называть словами. Продублировать каждое слово, содержащее в качестве фрагмента строку *s*.
15. Дан текстовый файл. Группы символов, разделенные пробелами, будем называть словами. В файле оставить только по одному экземпляру каждого слова и добавить перед ними количество их вхождений в первоначальный текст.
16. Даны текстовые файлы *f1*, *f2*, *f3*. Если в файле *f1* содержатся вхождения файла *f2*, то заменить их на содержимое файла *f3*.
17. Дан текстовый файл, компонентами которого являются целые числа. Каждый элемент, начиная с третьего, заменить суммой двух предыдущих элементов. Первые два элемента заменить соответственно наибольшим и наименьшим значениями преобразованного файла.

18. Дан текстовый файл, компонентами которого являются целые числа. Удалить из файла группы подряд идущих одинаковых чисел.
19. Даны текстовые файлы *f1, f2, f3*. Группы символов, разделенные пробелами, будем называть словами. В файле *f1* оставить только те слова, которые содержатся также в файлах *f2* и *f3*.
20. Дан текстовый файл. Заменить его содержимое перечнем символов, которые в нем встречаются, упорядоченным в порядке убывания частоты их вхождений.
21. Дан текстовый файл. Группы символов, разделенные пробелами, будем называть словами. Если какое-то слово состоит из двух, разделенных дефисом, то разбить его на два отдельных слова, отделенных друг от друга пробелом.
22. Дан текстовый файл, компонентами которого являются целые числа. Если какое-то число встречается один раз, то продублировать его. Дополнить файл количеством добавленных таким образом элементов.
23. Дан текстовый файл, каждая строка которого представляет собой дату в формате *дд.мм.гг*. Преобразовать файл таким образом, чтобы каждая строка содержала наименование месяца и год, после которых, отделенный двоеточием располагался список дней данного месяца, присутствующих в первоначальном перечне. Например: Январь 2007: 5,12,24,30
24. Дан текстовый файл. Преобразовать его таким образом, чтобы все строки имели длину, не превышающую длину его наименьшей строки. Лишние символы при необходимости переносятся в строки, расположенные ниже.
25. Дан текстовый файл, содержащий программу на языке Си. Произвести в тексте замену символических констант, описанных директивой *#define*, на замещающие строки в соответствии с правилами препроцессора. Считать, что в тексте отсутствуют комментарии и проход файла осуществляется один раз.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое динамическая переменная?
2. Какие средства для распределения динамической памяти вы знаете?
3. Каким образом в языке C/C++ создаются динамические массивы?

4. Каким образом в языке C/C++ освобождается память, занимаемая динамическим массивом?
5. Что такое динамическая структура данных?
6. В чем состоят основные отличия динамических и статических структур данных?
7. Что такое односвязный линейный список?
8. Как создать односвязный линейный список?
9. Как удалить односвязный линейный список?
10. Какие операции с линейными списками данных вам известны?

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Информатика: базовый курс / под ред. С.В. Симоновича. – СПб.: Питер, 2008. – 640 с.

2. Информатика: учеб. / под ред. Н.В. Макаровой. – 3-е изд. – М.: Финансы и статистика, 2007. – 768 с.

3. *Могилев, А.В.* Информатика: учеб. пособие/ А.В. Могилев, Н.И. Пак, Е.К. Хеннер. – М.: Академия, 2004. – 848 с.

4. *Острейковский, В.А.* Информатика: учеб. / В.А. Острейковский. – М.: Высшая школа, 2007.– 511 с.

5. *Демидович, Е.М.* Основы алгоритмизации и программирования. Язык СИ: учеб. пособие/ Е.М. Демидович. – СПб.: БХВ-Петербург, 2006. – 439 с.

6. *Костюкова, Н.И.* Язык Си и особенности работы с ним: учеб. пособие/ Н.И. Костюкова, Н.А. Калинина. – М.: БИНОМ. Лаборатория знаний, 2006. – 205 с.

7. *Павловская, Т.А.* С/С++. Программирование на языке высокого уровня: учеб./ Т.А. Павловская. – СПб.: Питер, 2009.– 432 с.

8. *Подбельский, В.В.* Курс программирования на языке Си: учеб./ В.В. Подбельский, С.С. Фомин. – М.: ДМК Пресс, 2012.– 384 с.

9. *Скляров, В.А.* Программирование на языках СИ и СИ++: учеб. пособие/ В.А. Скляров. – М.: Высшая школа, 1999. – 288 с.

Учебное издание

ИНФОРМАТИКА

Методические указания к выполнению лабораторных работ
для студентов очной формы обучения направления
бакалавриата 230400 – Информационные
системы и технологии

Составители: Рога Сергей Николаевич
Смышляев Артем Геннадьевич
Солопов Юрий Иванович

Подписано в печать 15.07.13. Формат 60х84 /16. Усл. печ. л. 7,3. Уч.-изд.л. 7,8.
Тираж 77 экз. Заказ Цена
Отпечатано в Белгородском государственном технологическом университете
им. В. Г. Шухова
308012, г. Белгород, ул. Костюкова, 46