Информационные процессы

Информационными называются процессы, связанные с *получением, хранением, обработкой и передачей* информации.

Примеры информационных процессов в деятельности человека

Получение информации

Получение информации основано на отражении различных свойств процессов, объектов и явлений окружающей среды. Этот процесс выражается в восприятии с помощью органов чувств.

Для улучшения восприятия информации человек придумал различные индивидуальные приспособления и приборы – очки, бинокль, микроскоп, стетоскоп, различные датчики и т. д.

Хранение информации

Хранение информации имеет большое значение для **многократного использования информации и передачи информации во времени**. Для долговременного хранения используются книги, в настоящее время – компьютерные носители, устройства внешней памяти и др. Информация чаще всего хранится для неоднократной дальнейшей работы с ней. В этом случае для ускорения поиска информация должна быть как-то упорядочена. В библиотеках это – картотеки, при хранении с использованием компьютера – размещение информации в определенных папках, в более сложных случаях – это базы данных, информационно-поисковые системы и т. д.

Обработка информации

Обработка информации подразумевает **преобразование ее к виду, отличному от исходной формы или содержания информации**. Процесс изменения информации может включать в себя, например, такие действия как численные расчеты, редактирование, упорядочивание, обобщение, систематизация и т. д.

Результаты обработки информации **в дальнейшем используются в тех или иных целях**, например: **получение новой информации** из уже известной путем логических рассуждений или математических вычислений (например, решение геометрической задачи); **изменение формы представления информации без изменения ее содержания** (например, перевод текста с одного языка на другой); **упорядочение** (сортировка) информации (например, упорядочение расписания движения поездов по времени их отправления).

Передача информации

Передача информации необходима для ее распространения. Основными устройствами для быстрой передачи информации на большие расстояния в настоящее время являются телеграф, радио, телефон, телевизионный передатчик, телекоммуникационные сети на базе вычислительных систем. Такие средства связи принято называть **каналами передачи информации**. Следует отметить, что в процессе передачи информации, она может искажаться или теряться. Это происходит в тех случаях, когда информационные каналы плохого качества или на линии связи присутствуют шумы (помехи).

Передача информации – это всегда двусторонний процесс, в котором есть **источник** и есть **приемник информации**. Источник передает информацию, а приемник ее получает.

Информация. Свойства информации

С точки зрения информатики, информация – это связанные между собой сведения, изменяющие наши представления о явлении или объекте окружающего мира.

Основные свойства информации

Понятность	Этим свойством обладает только та информация, которая выражена в форме, понятной тем, кому она предназначена, в противном случае информация становится бесполезной.
Полезность (ценность)	Это комплексный показатель качества информации. Зависит от того, какие задачи можно решить, используя эту информацию. Однако ценность информации – это понятие субъективное, т. к. информация, полезная для одного человека, может быть совершенно бесполезной для другого.
Достоверность	Информация достоверна, если она содержит сведения, отражающие истинное положение дел. Часто из-за искажений информации это свойство утрачивается. Кроме того, достоверная информация со временем может стать недостоверной, так как она обладает свойством устаревать, то есть перестает отражать истинное положение дел.
Актуальность	Она определяется степенью сохранения ценности информации в момент ее использования. Актуальную информацию очень важно иметь при работе в изменяющихся условиях, т. к. в таком случае только вовремя полученная (или обновленная) информация может принести пользу (примером может служить прогноз погоды).
Полнота и точность	Полнота информации означает, что она содержит минимальный, но достаточный для принятия правильного решения состав. Нарушение полноты информации сдерживает принятие решений и может повлечь ошибки.

Основные формы представления информации с точки зрения информатики: **символьная, текстовая, графическая и звуковая**.

За ***единицу*** принимается такое количество информации, которое содержит сообщение, уменьшающее неопределенность знаний в два раза. Такая единица называется **БИТ.**

Следующая по величине единица – **байт**

1 байт = 2^3 бит = 8 бит

Далее следуют:

1 Кбайт (килобайт) = 1024 байта

1 Мбайт (мегабайт) = 1024 Кбайта

1 Гбайт (гигабайт) = 1024 Мбайта

Основные тенденции развития информационного общества

Изменение структуры экономики и структуры труда	**В информационном обществе деятельность человека будет во многом зависеть от умения эффективно использовать имеющуюся информацию.** Использование компьютеров во всех сферах человеческой деятельности должно обеспечить доступ к достоверным источникам информации, избавить человека от рутинной работы, позволить ускорить принятие оптимальных решений, автоматизировать обработку информации не только в производственной, но и в социальной сферах. В результате движущей силой развития общества станет производство информационного, а не материального продукта. Этот процесс должен привести к созданию информационного общества, в котором главную роль будут играть знания и интеллект.
Рост информационной культуры	**Современное понимание информационной культуры заключается в умении и потребности человека работать с информацией средствами новых информационных технологий**. Она включает в себя гораздо больше, чем простой набор навыков технической обработки информации с помощью компьютера и телекоммуникационных средств. Культурный (в широком смысле) человек должен уметь оценивать получаемую информацию качественно, понимать ее полезность, достоверность и т. д. Существенный элемент информационной культуры – владение методикой коллективного принятия решений. Умение взаимодействовать в информационном поле с другими людьми – важный признак человека информационного общества.

Изменения в сфере образования	Большие изменения произойдут в информационном обществе в сфере образования. **Одна из принципиальных проблем, стоящих перед современным образованием, – сделать его более доступным для каждого человека.** Эта доступность имеет и экономические, и социальные, и технологические аспекты. В силу своего динамизма информационное общество потребует от своих членов непрерывного, на протяжении десятков лет, обучения. Это позволит человеку не отставать от времени, быть способным сменить профессию, занять достойное место в социальной структуре общества.
Свобода доступа к информации и свобода ее распространения	Современные информационные технологии чисто технически открыли безграничный простор для информационных обменов. **Свобода доступа к информации и свобода ее распространения – обязательное условие демократического развития, способствующее экономическому росту, добросовестной конкуренции на рынке.** Лишь опираясь на полную и достоверную информацию, можно принимать правильные и взвешенные решения в политике, экономике, науке, практической деятельности. Огромное значение имеет свобода распространения информации культурно-просветительного характера. Она способствует росту культурного и образовательного уровня общества.

Развитие и массовое использование информационных и коммуникационных технологий	**Создание телекоммуникационной инфраструктуры, включающей в себя сети передачи данных. Появление огромных баз данных, доступ к которым через сети получили миллионы людей. Выработка единых правил поведения в сетях и поиск в них информации.** Огромную роль в обсуждаемом процессе сыграло создание международной компьютерной сети Интернет. Сегодня она представляет собой колоссальную и быстро растущую систему, число пользователей которой приближается к 200 миллионам человек. Информационные и коммуникационные технологии постоянно развиваются.
Изменения уклада жизни людей	**Формирование информационного общества существенно отразится на повседневной жизни людей.** О том, насколько глубокими будут эти изменения, можно только догадываться. Так, массовое внедрение телевидения в 60–70-х годах XX века существенно изменило быт людей, причем не только в лучшую сторону. С одной стороны, у миллионов людей появилась возможность доступа к сокровищам национальной и мировой культуры, с другой – сократилось живое общение, стало больше стереотипов, насаждаемых телевидением, сузился круг чтения. Недавнее достижение Интернет-технологий – поход за покупками реальных товаров в виртуальный Интернет-магазин – может развиться в информационном обществе вплоть до ликвидации современной системы торговли.

Основные этапы развития вычислительной техники

<table>
<tr><th colspan="2">Ручной – с 50-го тысячелетия до н. э.</th></tr>
<tr><td colspan="2">Этот период автоматизации вычислений начался на заре человеческой цивилизации и базировался на использовании пальцев рук, камешков, палочек и т. п. Постепенно формировалась потребность в изобретении устройств, помогающих счету.
Одно из таких устройств известно под названием абак, вычисления здесь выполнялись перемещением костей или камешков. Подобные счетные устройства использовались в Греции, Японии и Китае. Аналогом абака в древней Руси являлись дошедшие до наших дней счеты.</td></tr>
<tr><td>Дж. Непер,
начало XVII века</td><td>Изобрел логарифмическую линейку, которая успешно использовалась в нашей стране еще 15–20 лет назад для проведения несложных инженерных расчетов. Она, несомненно, является венцом вычислительных инструментов ручного периода автоматизации.</td></tr>
</table>

Механический* – *с середины XVII века	
Развитие механики в XVII веке стало предпосылкой создания вычислительных устройств и приборов, использующих механический способ вычислений.	
Блез Паскаль, 1642 г.	Создал первую действующую модель счетной суммирующей машины, которая могла выполнять операции сложения и вычитания.
Готфрид фон Лейбниц, 1670–1680 гг.	Сконструировал счетную машину, позволяющую выполнять все четыре арифметических операции. Счетная машина Лейбница послужила прообразом для создания арифмометра – механического устройства для практических вычислений. Позднее арифмометр многократно совершенствовался, в том числе русскими учеными-изобретателями *П. Л. Чебышевым и В. Т. Однером*. Арифмометр использовался вплоть до середины XX века и явился предшественником современного калькулятора.
Чарльз Бэббидж	Выдвинул идею создания программно-управляемой счетной машины, имеющей *арифметическое устройство, устройство управления, ввода и печати*. Первая спроектированная Бэббиджем машина была создана в 1822 году и работала на *паровом* двигателе. Второй проект Бэббиджа – аналитическая машина, использующая принцип программного управления и предназначавшаяся для вычисления любого алгоритма. Проект не был реализован, но получил широкую известность и высокую оценку ученых.
леди Ада Лавлейс	Одновременно с Чарльзом Бэббиджем работала леди Ада Лавлейс. Она разработала первые программы для его машины, заложила многие идеи и ввела ряд понятий и терминов программирования, сохранившихся до настоящего времени.

Электромеханический – с девяностых годов XIX века	
Г. Холлерит	Создает в США первый счетно-аналитический комплекс, предназначенный для обработки результатов переписи населения в нескольких странах, в том числе и в России. Для проведения вычислений Холлерит наряду с механическими устройствами впервые применяет ***электричество***. Машина Холлерита содержала клавишный перфоратор, позволяющий перфорировать около 100 отверстий в минуту одновременно на нескольких картах (повторяющуюся информацию: штат, округ и прочее), машину для сортировки и табулятор. Машина для сортировки представляла собой набор ящиков с крышками, где карты продвигались между «считывающими» штырями на пружинах и резервуаром со ртутью. Когда штырь попадал в отверстие на перфокарте, то касался ртути и замыкал электрическую цепь, открывая крышку соответствующего ящика. Туда и попадала перфокарта. Табулятор работал аналогичным образом, только замыкание цепи приводило к увеличению содержимого соответствующего счетчика на единицу. В дальнейшем фирма Г. Холлерита стала одной из четырех фирм, положивших начало известной корпорации IBM.
А. Тьюринг и Э. Пост	Огромное влияние на дальнейшее развитие вычислительной техники оказали работы математиков, которые доказали *принципиальную возможность решения автоматами любой проблемы при условии, что ее можно представить в виде алгоритма* с учетом выполняемых машиной операций.

Электронный – с сороковых годов XX века	
США (ЭНИАК), 1946 г. ***СССР (МЭСМ), 1950 г.***	**Первые электронно-вычислительные машины (ЭВМ)**, способные *автоматически по заданной программе обрабатывать большие объемы информации*, были построены в 1946 г. в США (**ЭНИАК**) и в 1950 г. в СССР (**МЭСМ**). Первые ЭВМ были *ламповыми* (включали в себя десятки тысяч ламп), очень дорогими и очень большими (занимали громадные залы), поэтому их количество измерялось единицами, в лучшем случае десятками штук. Они использовались для проведения громоздких и точных вычислений в научных исследованиях, при проектировании ядерных реакторов, расчетов траекторий баллистических ракет и т. д. Программы для первых ЭВМ, написанные на машинном языке, представляли собой очень длинные последовательности нулей и единиц, так что составление и отладка таких программ было чрезвычайно трудоемким делом.

Дальнейшее совершенствование ЭВМ определялось *прогрессом* в области электроники (т. е. развитием **элементной базы**). Было положено начало новому поколению ЭВМ.
Под ***поколением ЭВМ*** принято понимать все типы и модели ЭВМ, построенные на одних и тех же научных и технологических принципах. Каждое следующее поколение отличается от предыдущего принципиально другой технологией изготовления новых электронных элементов.

Характерные черты ЭВМ каждого поколения

Поколение ЭВМ	Характеристики			
	I поколение	II поколение	III поколение	IV поколение
Годы применения	1946–1958	1959–1963	1964–1976	1977–...
Элементная база	электронно-вакуумные лампы, резисторы, конденсаторы, реле	полупроводниковые элементы, транзисторы	интегральные схемы (ИС)	большие интегральные схемы (БИС)
Количество ЭВМ в мире (шт.)	десятки	тысячи	десятки тысяч	миллионы
Габариты	в виде громоздких шкафов, занимает специальный зал	в виде стоек чуть выше человеческого роста	близки к габаритам II поколения	напольный и настольный варианты
Быстро-действие	10–20 тыс. оп./сек.	до 1 млн. оп./сек.	от сотен тысяч до миллионов оп./сек.	более десятков миллионов
Носители информации	перфокарты, перфоленты	магнитные ленты	магнитные ленты и магнитные диски	диски – магнитные, лазерные, магнитооптические

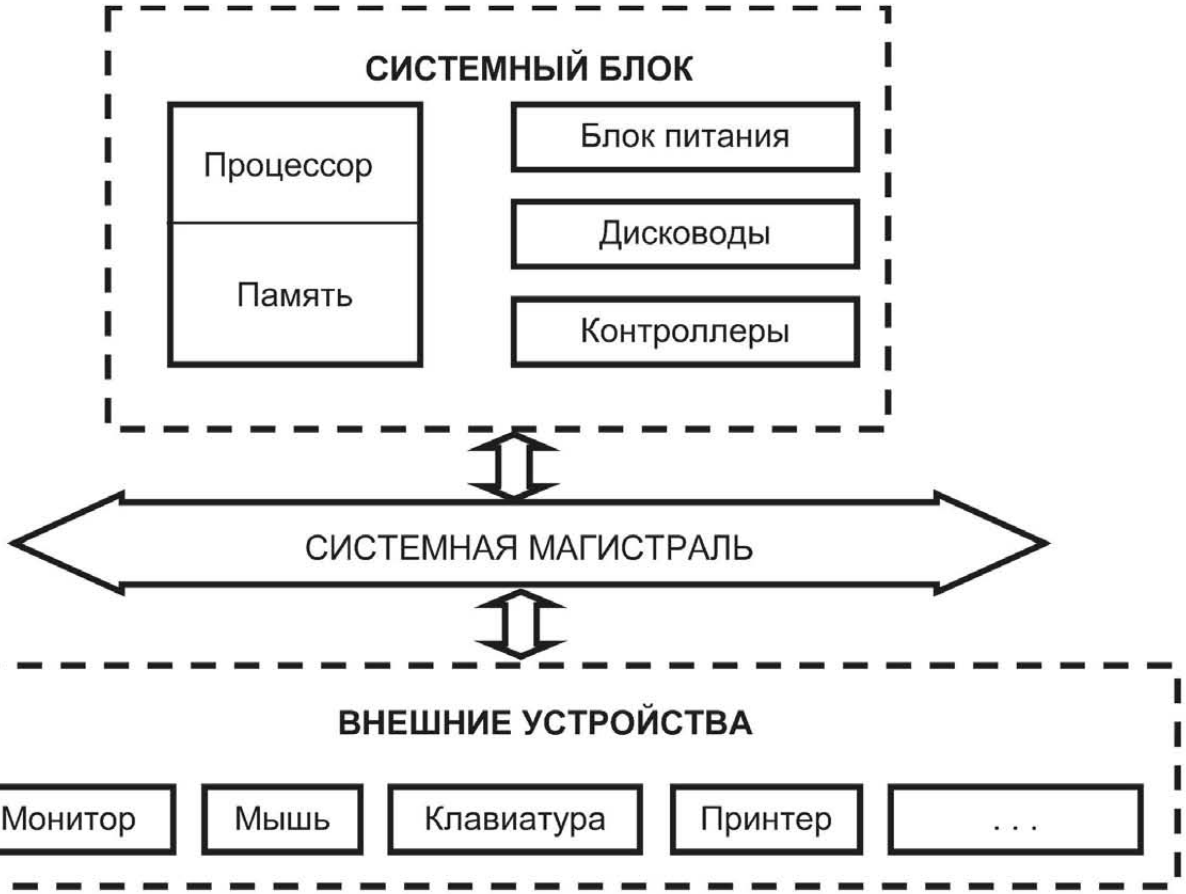
Особенности	сложные эксплуатация и обслуживание, перегрев, необходимость в специальной системе охлаждения	упрощение обслуживания – при неисправности происходит замена всей платы целиком, а не каждого элемента в отдельности	появление дисплеев, графопостроителей, упрощение эксплуатации благодаря применению принципа модульности	массовое производство персональных компьютеров. Появление средств мультимедиа. Реализация принципа открытой архитектуры. Создание микропроцессорных вычислительных систем, компьютерных сетей
Характер программного обеспечения	программирование в кодах, автокодах	алгоритмические языки программирования, появление АСУ и СУТП	режим разделения времени. Появление ППП, СУБД, САПР	совместимость программного обеспечения

Магистрально-модульный принцип построения компьютера

Магистраль или системная шина – это набор электронных линий, связывающих воедино процессор, память и периферийные устройства по адресации памяти, передачи данных и служебных сигналов.

Шина состоит из трех многоразрядных шин, соединяющих все модули: **шины данных, шины адресов** и **шины управления**. На шине адреса устанавливается адрес требуемой ячейки памяти или устройства, с которым будет происходить обмен информацией. По шине данных собственно и будет передана необходимая информация. Регулирует этот процесс шина управления.

Функциональная схема компьютера



Процессор, память, внешние устройства

Процессор является центральным устройством компьютера и отвечает за обработку всех видов информации, а также обеспечивает согласованные действия всех узлов, входящих в состав компьютера.

Соответственно наиболее важными частями процессора являются **арифметико-логическое устройство (АЛУ)** и **устройство управления (УУ)**.

Внутри процессора имеются специальные ячейки (**регистры**) для оперативного хранения обрабатываемых данных и некоторой служебной информации.

Для хранения как данных, так и программ их обработки в компьютере служит устройство – **память**. Согласно **фундаментальному принципу фон Неймана**, для обоих типов информации используется единое устройство.

Память компьютера принято делить на внешнюю и внутреннюю

Внешняя память

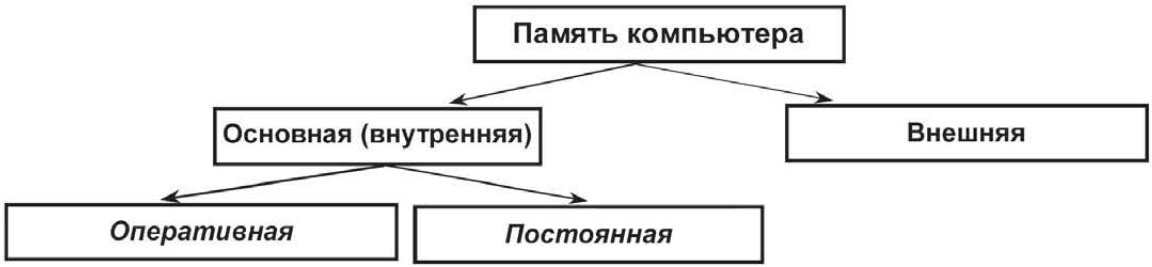
Используется для долговременного хранения информации. В качестве устройств внешней памяти обычно выступают накопители на гибких магнитных дисках (НГМД), накопители на жестких магнитных дисках (винчестерах) и оптические накопители (CD-ROM и DVD-ROM). В конструкции устройств внешней памяти имеются механически движущиеся части, поэтому скорость их работы существенно ниже, чем у полностью электронной внутренней памяти. Тем не менее, внешняя память позволяет сохранить огромные объемы информации с целью последующего использования.

Внутренняя память	
Это быстродействующая электронная память компьютера, расположенная на его системной плате. В составе внутренней памяти имеется оперативное запоминающее устройство (ОЗУ) и постоянное запоминающее устройство (ПЗУ).	
ОЗУ	**ПЗУ**
Его главное назначение состоит в том, чтобы хранить данные и программы для решаемых в текущий момент задач. Это могут быть данные, программы их обработки, промежуточные результаты и т.п. При выключении компьютера содержимое оперативной памяти стирается. Оперативная память обычно характеризуется сравнительно небольшим объемом (до 512 Мб) и очень высокой скоростью обращения к информации.	Эта память используется компьютером только для чтения и служит для хранения программ и данных начальной загрузки компьютера и тестирования его узлов. Как видно из названия, информация в ПЗУ не зависит от состояния компьютера.

В составе любого компьютера имеются специальные устройства ввода и вывода информации, так называемые внешние или периферийные устройства, например:

- клавиатура (для ввода числовой и текстовой информации);
- монитор, на экране которого высвечивается числовая, текстовая, графическая и видеоинформация;
- манипулятор мышь (для ввода графической информации или работы с графическим интерфейсом программ);
- принтер (для сохранения информации на бумаге);
- сканер (для ввода в компьютер фотографий или рисунков, позволяет с помощью специального программного обеспечения распознать в прочитанном изображении текст и сохранить его в виде, пригодном для редактирования в обычном текстовом редакторе);
- микрофон (служит для ввода звуковой информации).

Устройства памяти компьютера. Носители информации



Оперативная память	Предназначена для хранения информации, изменяющейся в ходе выполнения процессором заданных операций (данных, программ, промежуточных результатов и т. п.) При выключении компьютера содержимое оперативной памяти стирается. Оперативная память обычно характеризуется сравнительно небольшим объемом (до 512 Мб) и очень высокой скоростью обращения к информации.
Постоянная память	Используется компьютером только для чтения и служит для хранения программ и данных начальной загрузки компьютера и тестирования его узлов. Содержимое этой памяти устанавливается непосредственно при изготовлении и в дальнейшем не изменяется. После выключения компьютера содержимое ПЗУ сохраняется.
Внешняя память	Для долговременного хранения информации используется внешняя память. В качестве устройств внешней памяти обычно выступают накопители на гибких магнитных дисках (НГМД), накопители на жестких магнитных дисках (винчестерах) и оптические накопители (CD-ROM и DVD-ROM). Самыми медленными из них по скорости обмена данными являются гибкие диски (0,05 Мбайт/с), а самыми быстрыми – жесткие диски (до 100 Мбайт/с).

Для работы с **внешней** памятью необходимо наличие накопителя (устройства, обеспечивающего запись и (или) считывание информации) и устройства хранения – носителя.

В настоящее время используются следующие виды накопителей	***Им соответствуют основные виды носителей***
• накопители на гибких магнитных дисках (НГМД); • накопители на жестких магнитных дисках (НЖМД); • накопители на магнитной ленте (НМЛ); • накопители CD-ROM, CD-RW, DVD.	• гибкие магнитные диски (Floppy Disk), диски для сменных носителей; • жесткие магнитные диски (Hard Disk); • кассеты для стримеров и других НМЛ; • диски CD-ROM, CD-R, CD-RW, DVD.

Важными характеристиками внешней памяти являются *объем, время доступа и скорость обмена информацией*

Тип накопителя	***Емкость носителя***	***Скорость обмена данными (Мб/с)***
Дискеты 3.5"	1,44 Мб	0,05
Винчестеры	до 50 Гб	до 100
CD-ROM	650 Мб	до 7,8
DVD-ROM	до 17 Гб	до 21

Перевод целых чисел в двоичную систему счисления и системы, родственные двоичной

Для перевода целых чисел в двоичную систему счисления, согласно описанному выше алгоритму, используется их последовательное деление на основание системы счисления 2, остатки от деления выписываются в порядке, обратном их получению.

Пример перевода десятичного числа **654** в двоичную систему счисления:

Частное	654	327	163	81	40	20	10	5	2	1
Остаток	0	1	1	1	0	0	0	1	0	1

Таким образом, ответ: $654_{10} = 1010001110_2$.

Для перевода целого числа из двоичной системы счисления в систему счисления, основанием которой является степень двойки, надо объединить цифры двоичного числа в группы по столько цифр, каков показатель вышеуказанной степени (например, если перевод осуществляется в восьмеричную систему, то группы будут содержать по три цифры, т. к. $8 = 2^3$). Группировка производится *справа налево*. Если в последней группе недостает цифр, слева дописываются нули. Каждая группа заменяется соответствующей цифрой новой системы в соответствии с приведенными таблицами.

Для восьмеричной системы:

P=2	000	001	010	011	100	101	110	111
P=8	0	1	2	3	4	5	6	7

Для шестнадцатеричной системы:

P=2	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
P=16	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

Представление целых чисел в компьютере

Чтобы получить внутреннее представление целого положительного числа N, хранящегося в k-разрядной ячейке памяти, необходимо:

1. перевести число N в двоичную систему счисления;
2. полученный результат дополнить слева незначащими нулями до k разрядов.

Пример внутреннего представления числа $1607_{10} = 11001000111_2$ в 2-байтовой ячейке:

0	0	0	0	0	1	1	0	0	1	0	0	0	1	1	1

Для хранения целых чисел со знаком отводится две ячейки памяти (16 битов), при этом старший (крайний левый) разряд отводится под знак числа.

Если число положительное, то в этот разряд записывается 0, если отрицательное, то 1.

Пример представления десятичного числа $2356_{10} = 100100110100_2$ в 16-разрядной сетке:

0	0	0	0	1	0	0	1	0	0	1	1	0	1	0	0

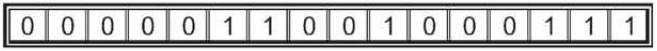
Представление положительных чисел с учетом знака называется **прямым кодом** числа. Для представления отрицательных чисел используется **дополнительный код**.

Для получения **дополнительного кода** отрицательного числа можно использовать следующий алгоритм:

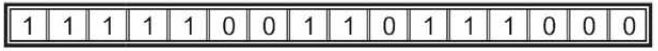
1. Модуль числа (число без знака) записывают в прямом коде в **n** двоичных разрядах.
2. Значения всех битов прямого кода инвертируют, т. е. все единицы заменяют на нули, а все нули – на единицы. Таким образом, получают обратный код числа.
3. К полученному обратному коду прибавляют единицу.

Пример внутреннего представления целого отрицательного числа -1607

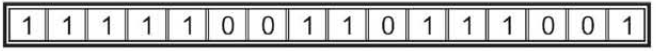
1. Внутреннее представление положительного числа 1607:



2. Инвертированием получим обратный код:



3. Добавим единицу:



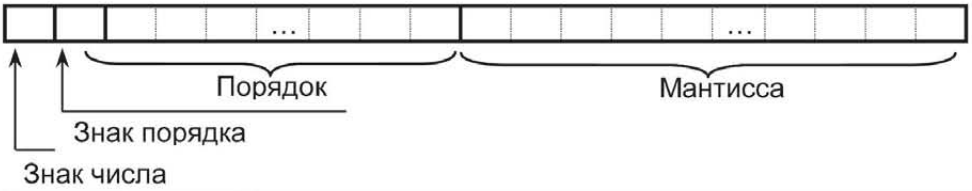
Это и есть внутреннее двоичное представление числа **-1607.**

Для положительных чисел прямой, обратный и дополнительный коды совпадают, а для отрицательных – нет.

Представление вещественных чисел

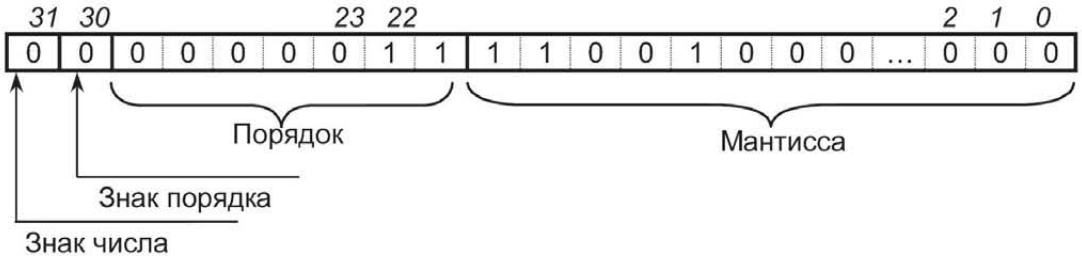
Вещественные числа хранятся в памяти компьютера и обрабатываются процессором в формате с плавающей запятой. В этом случае число A представляется в виде $A = m \cdot q^n$, где: **m** – мантисса числа; **q** – основание системы счисления; **n** – порядок числа.

Число в формате с плавающей запятой занимает в памяти компьютера **4 байта** (если это число так называемой двойной точности, то – 8 байтов). При этом отдельно выделяются разряды для хранения знака числа, знака порядка, порядка и самой мантиссы..



Чем больше разрядов отводится под запись мантиссы, тем выше точность представления числа.

Пример записи числа $6{,}25_{10} = 110{,}01_2 = 0{,}11001 \cdot 2^{11}$, представленного в нормализованном виде, в четырехбайтовом формате с семью разрядами для записи порядка.



Двоичное кодирование текстовой информации

При вводе в компьютер текстовой информации **каждая буква кодируется определенным числом**, а при выводе на внешние устройства (экран или печать) для восприятия человеком по этим числам строятся изображения букв. Соответствие между набором букв и числами называется **кодировкой** символов.

- Традиционно **для кодирования одного символа** используется количество информации, равное **1 байту (8 битам).**
- Кодирование заключается в том, что каждому символу ставится в соответствие уникальный десятичный код (или соответствующий ему двоичный код).
- Код символа хранится в памяти компьютера, где занимает **1 байт**.
При таком способе можно закодировать **256** различных символов **($256 = 2^8$)**.

Такое количество символов достаточно для представления текстовой информации, включая прописные и заглавные буквы русского алфавита, цифры, знаки, графические символы и т. д.

Каждому символу такого алфавита ставится в соответствие уникальный десятичный код от 0 до 255, а каждому десятичному коду соответствует 8-разрядный двоичный код от 00000000 до 11111111. Таким образом, **компьютер различает символы по их коду**.

● Присвоение символу конкретного кода является вопросом соглашения, которое фиксируется в конкретной кодовой таблице.

● **В качестве международного стандарта принята кодовая таблица ASCII.**
В этой кодовой таблице латинские буквы (прописные и строчные) располагаются в алфавитном порядке. Расположение цифр также упорядочено по возрастанию значений. Это правило соблюдается и в других таблицах кодировки и называется **принципом последовательного кодирования алфавитов.**

● **Стандартными** в этой таблице кодов ASCII являются только **первые 128 символов**, т. е. символы с номерами от нуля (двоичный код 00000000) до 127 (01111111). Сюда входят буквы латинского алфавита, цифры, знаки препинания, скобки и некоторые другие символы.

● Остальные 128 кодов, начиная со 128 (двоичный код 10000000) и кончая 255 (11111111), используются для кодировки букв национальных алфавитов, символов псевдографики и научных символов.

Наиболее распространенной в настоящее время является кодировка **MS Windows**, которая обозначается как **CP1251** или **Windows 1251**. В настоящее время все большее число программ начинает поддерживать стандарт **Unicode**, который позволяет кодировать практически все языки и диалекты жителей Земли.
Этот стандарт отводит на каждый символ не ***один*** байт, а ***два***, поэтому с его помощью можно закодировать **65536 различных символов (65536 = 2^{16})**.

Кодирование цветовой информации

- **Одним байтом** можно закодировать **256 различных цветов** (2^8 цветов).
- Если на кодирование цвета выделить **два байта**, то можно закодировать **65536 различных цветов** (2^{16} цветов).
- Если для кодирования использовать **3 байта**, то – **16,5 млн цветов** (2^{24} цветов). Такой режим позволяет создать на экране изображение, не уступающие по качеству цвета живой природе.

Цветовые модели

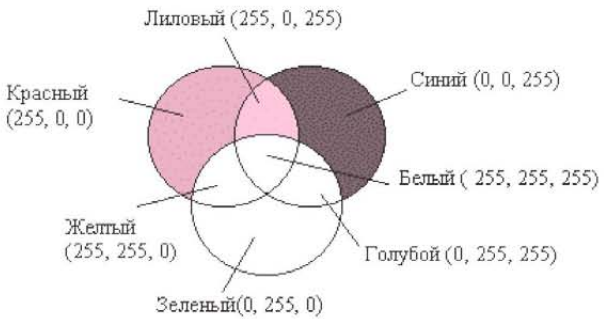
Способ разделения цветового оттенка на составляющие компоненты называется ***цветовой моделью***. В компьютерной графике применяются три цветовые модели: **RGB, CMYK, HSB**.

Цветовая модель RGB

Используется для **излучаемого** цвета, т. е. при подготовке экранных документов. Любой цвет можно представить в виде комбинации трех основных цветов: ***красного*** (**R**ed), ***зеленого*** (**G**reen) и ***синего*** (**B**lue).

Эти цвета называются **цветовыми составляющими.** При кодировке цвета точки изображения с помощью трех байтов, первый байт кодирует красную составляющую, второй – зеленую, третий – синюю.

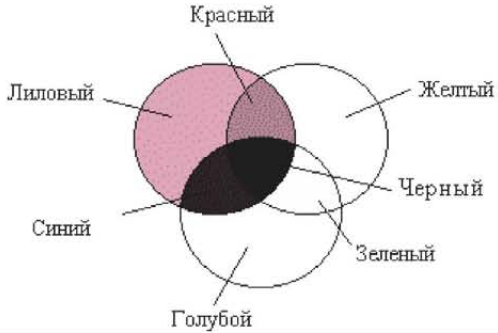
Чем больше значение байта цветовой составляющей, тем ярче этот цвет. При наложении одной составляющей на другую яркость суммарного цвета также увеличивается. Поэтому цветовая модель **RGB**, использующаяся для излучаемого цвета, называются **аддитивной**.



Цветовая модель CMYK

Используется при работе с **отраженным** цветом, т. е. для подготовки печатных документов. Цветовыми составляющими этой модели являются цвета:
голубой (**C**yan), ***лиловый*** (**M**agenta), ***желтый*** (**Y**ellow) и ***черный*** (**B**lack).

Эти цвета получаются в результате вычитания основных цветов модели RGB из белого цвета. Черный цвет задается отдельно.
Увеличение количества краски приводит к уменьшению яркости цвета. Поэтому цветовая модель **CMYK**, использующаяся для отраженного цвета, называются **субтрактивной.**



Системы цветов **RGB** и **CMYK** связаны с **ограничениями**, накладываемыми аппаратным обеспечением (монитор компьютера в случае RGB и типографские краски в случае CMYK).

Цветовая модель HSB

Наиболее удобна для человека, т. к. она хорошо согласуется с моделью восприятия цвета человеком. Компонентами модели HSB являются: ***тон*** (**H**ue), ***насыщенность*** (**S**aturation), ***яркост***ь цвета (**B**rightness).

Тон – это конкретный оттенок цвета.
Насыщенность характеризует его интенсивность, или чистоту.
Яркость же зависит от примеси черной краски, добавленной к данному цвету.

Модель **HSB** удобно применять при создании собственно изображения, а по окончании работы изображение можно преобразовать в модель RGB или CMYK.

Программы, предназначенные для создания и обработки графических изображений, называются ***графическими редакторами***.

Таблицы истинности логических функций

Таблицей истинности логической функции принято называть ***табличное представление логической операции***, в котором присутствуют все возможные сочетания значений входных переменных и получаемые при этом значения выходных переменных (результатов логической операции).

Таблица истинности функции логического

отрицания (инверсии)

X	$\overline{X}$
0	1
1	0

сложения (дизъюнкции)

X	Y	$Z = X \vee Y$
0	0	0
0	1	1
1	0	1
1	1	1

умножения (конъюнкции)

X	Y	$Z = X \,\&\, Y$
0	0	0
0	1	0
1	0	0
1	1	1

Свойства

дизъюнкции

$X \vee Y \equiv Y \vee X;$

$(X \vee Y) \vee Z \equiv X \vee (Y \vee Z);$

$X \vee 0 \equiv X,$

где 0 – тождественно ложное высказывание;

$X \vee \overline{X} \equiv 1;$

$X \vee 1 \equiv 1,$

где 1 – тождественно истинное высказывание;

$X \vee X \equiv X$

конъюнкции

$X \,\&\, X \equiv X;$

$X \,\&\, Y \equiv Y \,\&\, X;$

$(X \,\&\, Y) \,\&\, Z \equiv X \,\&\, (Y \,\&\, Z);$

$X \,\&\, \overline{X} \equiv 0;$

$X \,\&\, 0 \equiv 0,$

где 0 – тождественно ложное высказывание;

$X \,\&\, 1 \equiv X,$

где 1 – тождественно истинное высказывание

Основные законы алгебры логики

Закон непротиворечия
$A \& \overline{A} = 0$

Закон исключенного третьего
$A \vee \overline{A} = 1$

Законы де Моргана
$\overline{A \vee B} = \overline{A} \& \overline{B}, \quad \overline{A \& B} = \overline{A} \vee \overline{B}$

Закон двойного отрицания
$A = \overline{\overline{A}}$

Закон коммутативности. При операциях логического умножения и логического сложения логические переменные можно менять местами.
$A \& B = B \& A, \quad A \vee B = B \vee A$

Закон ассоциативности. Если в логическом выражении используется только операция логического умножения или только операция логического сложения, то можно пренебречь скобками или расставить их произвольно.
$(A \& B) \& C = A \& (B \& C), \quad (A \vee B) \vee C = A \vee (B \vee C)$

Закон дистрибутивности. За скобки можно выносить как общие множители, так и общие слагаемые.
$(A \& B) \vee (A \& C) = A \& (B \vee C),$ $(A \vee B) \& (A \vee C) = A \vee (B \& C)$

Закон поглощения
$A \vee (A \& B) = A, \; A \& (A \vee B) = A$

Закон исключения (склеивания)
$(A \& B) \vee (\overline{A} \& B) = B, \; (A \vee B) \& (\overline{A} \vee B) = B$

Кроме этого справедливы следующие законы
$A \vee A = A, \quad A \& A = A, \quad A \vee 1 = 1,$ $A \vee 0 = 0, \quad A \& 1 = A, \quad A \& 0 = 0.$

Правила составления таблиц истинности для сложных логических функций

Для любой логической функции можно построить *таблицу истинности*, которая определяет ее истинность или ложность при всех возможных комбинациях значений аргументов (логических переменных).

При построении таблиц истинности целесообразно придерживаться следующего *алгоритма действий*:

1. Сначала определяют количество строк в таблице истинности. Количество строк будет равно 2^n, (где **n** – количество логических переменных) плюс строка заголовка.

2. Далее определяют количество столбцов в таблице истинности, оно равно количеству логических переменных плюс количество логических операций.

3. Затем строится таблица истинности с указанным количеством строк и столбцов, столбцы подписываются, в таблицу вносятся всевозможные наборы значений исходных логических переменных.

4. И, наконец, выполняются необходимые логические операции, таблица истинности заполняется по столбцам.

Пример составления таблицы истинности для логической функции, содержащей операции отрицание, дизъюнкцию и конъюнкцию

«Составление таблицы истинности для логической функции $Z = (A \vee E) \& (\bar{A} \vee \bar{E})$»

1. Определим количество строк в нашей таблице истинности.

Мы имеем две логические переменные A и E, следовательно, количество строк будет равно $2^2 + 1 = 5$.

2. Выясним, какое количество столбцов необходимо.

У нас – две логические переменные, над которыми будет выполнено 5 логических операций – это $A \vee E$, $\bar{A}$, $\bar{E}$, $\bar{A} \vee \bar{E}$ и, наконец, $(A \vee E) \& (\bar{A} \vee \bar{E})$. Значит, нам понадобится 7 столбцов.

3. Построим таблицу, подпишем столбцы и заполним ее исходными значениями логических переменных. В результате должна получиться следующая таблица:

A	E	$A \vee E$	$\bar{A}$	$\bar{E}$	$\bar{A} \vee \bar{E}$	$(A \vee E) \& (\bar{A} \vee \bar{E})$
0	0					
0	1					
1	0					
1	1					

4. **Теперь выполним необходимые операции и последовательно (слева направо) заполним все столбцы.** Результатом будет готовая таблица истинности:

A	E	$A \vee E$	$\bar{A}$	$\bar{E}$	$\bar{A} \vee \bar{E}$	$(A \vee E) \& (\bar{A} \vee \bar{E})$
0	0	0	1	1	1	0
0	1	1	1	0	1	1
1	0	1	0	1	1	1
1	1	1	0	0	0	0

Логическая схема полусумматора

Таблица сложения одноразрядных двоичных чисел с учетом переноса в старший разряд:

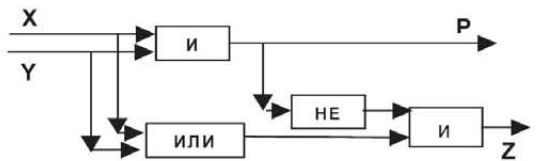
Слагаемые		*Сумма*	*Перенос*
X	***Y***	***Z***	***P***
0	0	0	0
0	1	0	0
1	0	1	0
1	1	0	1

Здесь **X** и **Y** – слагаемые,
Z – сумма (полученная в данном разряде двоичная цифра),
P – перенос в старший разряд.

Переносу в следующий разряд можно поставить в соответствие обычную операцию логического умножения (конъюнкцию), т. е. P = X & Y.
Логическое выражение, определяющее сумму двоичных цифр в данном разряде:

$$Z = (X \vee Y) \,\&\, \overline{(X \,\&\, Y)}.$$

Схема **полусумматора**, выполняющего суммирование одноразрядных двоичных чисел без учета переноса из младшего в старший разряд.



Данная схема называется **полусумматором**, т. к. выполняет суммирование одноразрядных двоичных чисел без учета переноса из младшего в старший разряд.

Полный одноразрядный сумматор должен иметь три входа: X, Y (слагаемые), P_0 (перенос из младшего разряда) и два выхода: сумму Z и перенос P.

В этом случае **таблица сложения** будет иметь следующий вид:

Слагаемые		*Перенос из младшего разряда*	*Сумма*	*Перенос*
X	*Y*	P_0	*Z*	*P*
0	0	0	0	0
0	1	0	1	0
1	0	0	1	0
1	1	0	0	1
0	0	1	1	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	1

Переносу можно поставить в соответствие логическое сложение результатов попарного логического умножения входных переменных (X, Y, P_0).

Формула для переноса:

$$P = (X \,\&\, Y) \vee (X \,\&\, P_0) \vee (Y \,\&\, P_0).$$

Значение суммы:

$$Z = (X \vee Y \vee P_0) \,\&\, P \vee (X \,\&\, Y \,\&\, P_0).$$

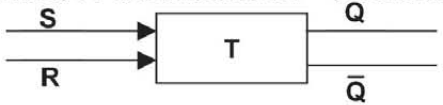
Для сложения многоразрядных двоичных чисел применяют многоразрядный двоичный сумматор, который представляет собой комбинацию одноразрядных сумматоров. На каждый разряд двоичного числа ставится одноразрядный сумматор, при этом выход переноса младшего разряда сумматора является входом для сумматора старшего разряда.

Триггер

Триггер – это электронная схема, применяемая в регистрах компьютера для запоминания одного разряда двоичного кода. Триггер имеет два устойчивых состояния, одно из которых соответствует двоичной единице, а другое – двоичному нулю.

Условное обозначение RS-триггера

(S и R, соответственно, от английских set – установка и reset – сброс):



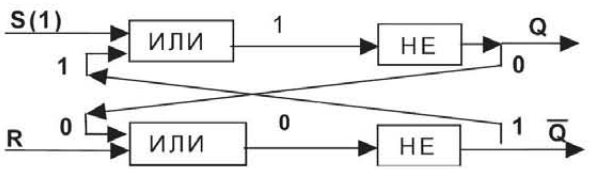
Он имеет два симметричных входа S и R и два симметричных выхода Q и $\overline{Q}$. На каждый из двух входов S и R могут подаваться входные сигналы в виде кратковременных импульсов.

Наличие импульса на входе считается единицей, а его отсутствие – нулем. *В обычном состоянии на оба входа триггера подается сигнал «0» и триггер хранит «0».*

Для записи в триггер «1» на вход S подают сигнал «1». При этом видно, что триггер перейдет в состояние «1» и будет устойчиво находиться в этом состоянии даже после того, как сигнал на входе S исчезнет (станет равным «0»).

Для того, чтобы **сбросить информацию** и подготовить триггер к приему новой, на вход R подается сигнал «1», при этом триггер возвращается к исходному «нулевому» состоянию.

Триггер можно построить из двух логических элементов ИЛИ и двух элементов НЕ:



Один триггер может запомнить только один разряд двоичного кода. Для запоминания, например, одного байта информации, потребуется 8 триггеров. Современные микросхемы памяти содержат миллионы триггеров.

Файловая система компьютера

Файл – **это программа или данные, имеющее имя и хранящееся в долговременной (внешней) памяти.**

На каждом носителе информации может храниться большое количество файлов. Порядок хранения файлов на диске определяется установленной ***файловой системой***.

Файловая система – это система хранения файлов и организации каталогов.

Для дисков с небольшим количеством файлов (до нескольких десятков) удобно применять ***одноуровневую файловую систему***, когда каталог (оглавление диска) представляет собой линейную последовательность имен файлов.

Если на диске хранится много файлов, то для облегчения поиска нужного файла рекомендуется ©объединять» их в папки по определенной тематике. Каждой папке тоже присваивается имя, но в отличие от имени файла, его принято писать без расширения. Одни папки могут находиться внутри других, тогда их называют ***вложенными папками***.

Таким образом, на диске кроме самих файлов хранятся **папки** (или **каталоги**).

Каждый диск можно рассматривать как ***главную папку***. Совокупность папок, вложенных друг в друга, образует так называемую иерархическую файловую систему, которая имеет ***«древовидную»*** структуру.

Любой носитель изначально имеет один каталог, который создается операционной системой без участия пользователя. Этот каталог называется **корневым**.

Корневой каталог на каждом носителе внешней памяти существует **в единственном экземпляре**. Все другие каталоги создаются либо пользователем, либо могут быть автоматически созданы программами.

Файлы и каталоги, зарегистрированные в ***одном каталоге***, должны иметь ***уникальные*** имена. Файлы (или каталоги), зарегистрированные на одном и том же носителе информации, но в разных каталогах, могут иметь совпадающие имена.

Программное обеспечение компьютера

Системное ПО

Операционные системы и оболочки, утилиты, драйверы, программы-упаковщики (архиваторы), антивирусные программы, программы оптимизации и контроля качества дискового пространства, программы восстановления и защиты информации, коммуникационные программы, организующие обмен данными между компьютерами, программы для записи CD-ROM, CD-R и т. д.

управляют ресурсами компьютера – центральным процессором, памятью, вводом-выводом; создают копии используемой информации, проверяют работоспособность устройств компьютера, выдают справочную информацию о компьютере; проверяют правильность функционирования устройств компьютера для обнаружения неисправностей в процессе эксплуатации, указывают причину и место неисправности.

Прикладное ПО

Обеспечивают выполнение необходимых пользователю работ, в том числе облегчают процесс создания новых программ для компьютера.

Системы программирования – инструментальное средство для разработки программ на различных языках программирования.

Приложения – предоставляют пользователю возможность обрабатывать текстовую, графическую, числовую, звуковую и видеоинформацию, работать в компьютерных сетях (текстовые и графические редакторы, электронные таблицы, системы управления базами данных, приложения для создания мультимедиа-презентаций, коммуникационные программы, системы компьютерной графики, системы автоматического проектирования, бухгалтерские программы, компьютерные словари, переводчики и пр.).

Понятие алгоритма. Свойства алгоритма

Алгоритм – это понятное и точное указание исполнителю совершить последовательность действий, направленных на решение поставленной задачи. Исполнителем может являться как техническое устройство, так и человек.

С помощью алгоритма может быть описана работа любого технического устройства, в частности компьютера.

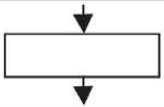
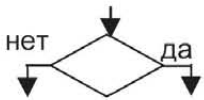
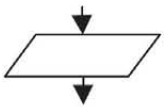
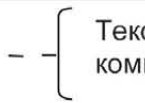
Все алгоритмы обладают некоторыми общими свойствами:

Дискретность	Процесс решения задачи должен быть разбит на последовательность отдельных шагов, каждый из которых называется командой. Наиболее существенным здесь является тот факт, что алгоритм есть последовательность четко выделенных пунктов, – такие объекты принято называть ***дискретными***. Таким образом, разделение информационного процесса в алгоритме на отдельные команды является важным свойством алгоритма и называется ***дискретностью***.
Понятность	Чтобы исполнитель мог выполнить преобразование объекта согласно алгоритму, он должен быть в состоянии понять и выполнить каждую команду. Поэтому каждый алгоритм должен составляться в расчете на конкретного исполнителя с учетом его возможностей. Это свойство алгоритмов называется ***понятностью***. У каждого исполнителя имеется перечень команд, которые он может исполнить. Такой перечень (список) называется ***системой команд исполнителя*** (СКИ). Понятными исполнителю будут являться только те команды, которые попадают в этот список.

Определенность	Команды, образующие алгоритм (или, можно сказать, входящие в систему команд исполнителя), должны быть предельно четкими и однозначными. Их результат не может зависеть от какой-либо дополнительной информации извне алгоритма. Сколько бы раз вы не запускали программу, для одних и тех же исходных данных всегда будет получаться один и тот же результат. Такое свойство называется **определенностью (детерминированностью)**. При наличии ошибок в алгоритме это свойство может иногда нарушаться.
Результативность	Результат выполнения алгоритма должен быть обязательно получен, т.е. правильный алгоритм не может обрываться безрезультатно из-за какого-либо непреодолимого препятствия в ходе выполнения. Кроме того, любой алгоритм должен завершиться за конечное число шагов. Такое свойство алгоритма называется **результативностью (конечностью)**. Большинство алгоритмов данным требованиям удовлетворяют, но при наличии ошибок возможны нарушения результативности.
Корректность	Любой алгоритм создан для решения той или иной задачи, поэтому необходима уверенность, что это решение будет правильным для любых допустимых исходных данных. Указанное свойство алгоритма принято называть его **корректностью**. В связи с этим большое значение имеет тщательное тестирование алгоритма перед его использованием. Грамотная и всесторонняя отладка для сложных алгоритмов часто требует значительно больших усилий, чем собственно разработка этих алгоритмов. Именно результатом недостаточной тщательности тестирования чаще всего объясняются многочисленные сюрпризы, преподносимые современным программным обеспечением в процессе эксплуатации.
Массовость	Алгоритм имеет смысл разрабатывать только в том случае, когда он будет применяться многократно для различных наборов исходных данных. **Массовость** алгоритма в отдельных случаях может нарушаться: к числу подобных исключений можно отнести алгоритмы пользования некоторыми простыми автоматами (для них входными данными служит единственный тип монет).

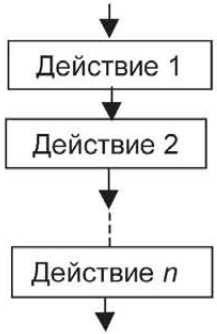
Условные обозначения в блок-схемах алгоритмов

Схема алгоритма **представляет собой систему связанных геометрических фигур, каждая из которых обозначает один этап процесса решения задачи и называется *блоком*.**
Порядок выполнения блоков указывается стрелками, соединяющими блоки.
В схеме блоки размещаются сверху вниз в порядке их выполнения.

Наименование	Обозначение	Комментарии
Пуск-останов		Начало или конец алгоритма, вход/выход в подпрограмму
Процесс		Действия, вычисления или группа операций
Принятие решения		Разветвление в алгоритме, проверка условия
Ввод/вывод		Ввод/вывод данных в общем виде
Комментарии		Комментарии

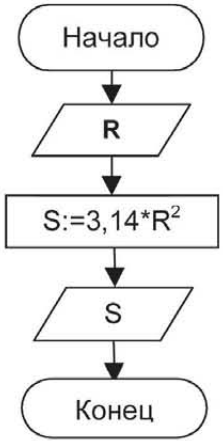
Линейная алгоритмическая конструкция

Линейным называется алгоритм, в котором все этапы решения задачи выполняются строго последовательно, без пропусков и повторений.



Пример линейного алгоритма – задача вычисления площади круга S при заданном значении радиуса R.

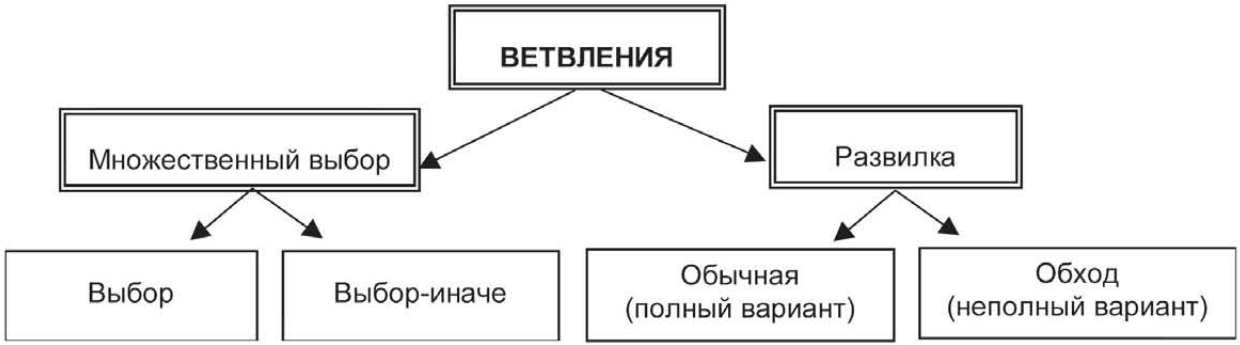
Блок-схема данного алгоритма:



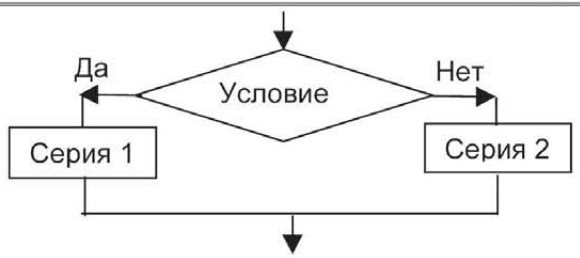
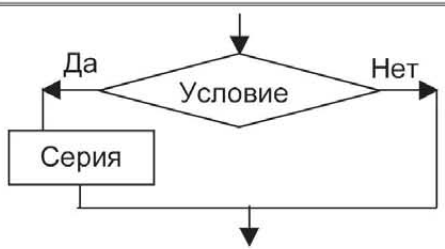
Словесная запись данного алгоритма:

1. Прочесть значение R;
2. Умножить значение R на 3,14;
3. Умножить результат на значение R;
4. Сохранить полученный на предыдущем шаге результат как значение S.

Разветвляющиеся алгоритмические структуры



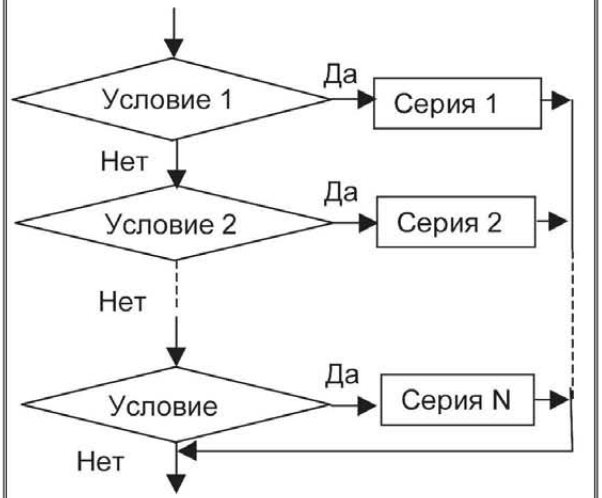
Алгоритмическая структура «Развилка»

Полный вариант	Неполный вариант («обход»)
Блок-схема	
	
Описние на алгоритмическом языке	
если *логическое выражение (условие)* **то** *серия команд 1* **иначе** *серия команд* **конец** ветвления	**если** *логическое выражение (условие)* **то** *серия команд* **конец** ветвления

Алгоритмическая структура «Множественный выбор»

«Выбор»

Блок-схема

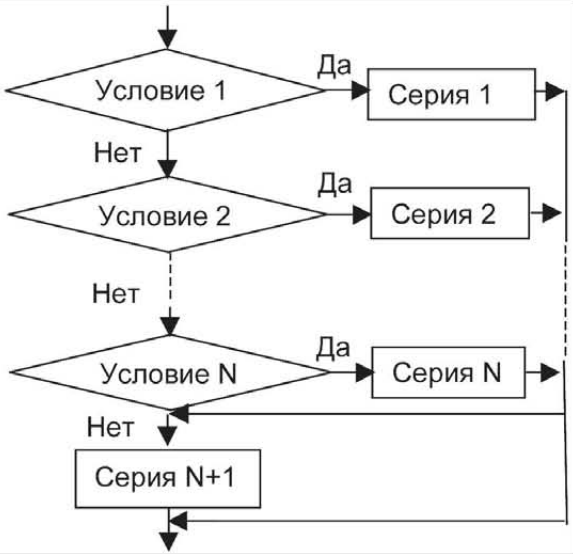


Описание на алгоритмическом языке

выбор
  при условие 1: серия команд 1
  при условие 2: серия команд 2
  . . .
  при условие N: серия команд N
все

«Выбор-иначе»

Блок-схема



Описание на алгоритмическом языке

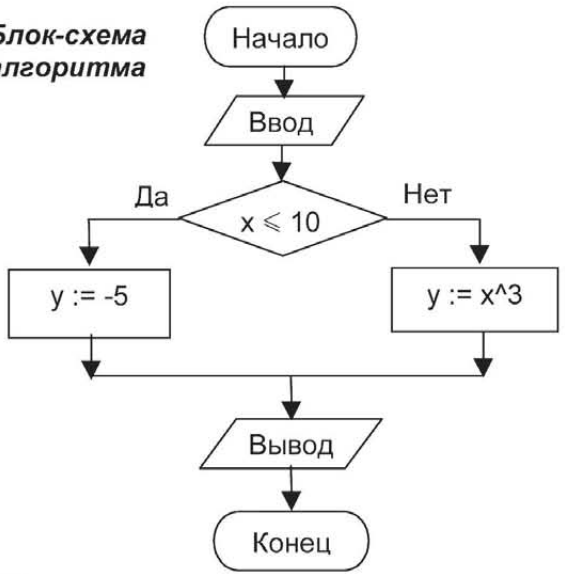
выбор
  при условие 1: серия команд 1
  при условие 2: серия команд 2
  . . .
  при условие N: серия команд N
  иначе серия команд N+1
все

Пример алгоритма, содержащего алгоритмическую структуру «полное» ветвление

Задача. Составить алгоритм, вычисляющий значение функции **y(x)** для заданного **x**:

$$y(x) = \begin{cases} -5, & \text{при } x \leqslant 10; \\ x^3, & \text{при } x > 10. \end{cases}$$

Блок-схема алгоритма

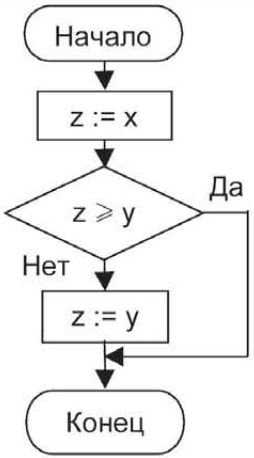


Комментарий. В зависимости от результата сравнения значения **x** с числом **10** переменной **y** присваивается одно из двух возможных значений (**-5** или $\mathbf{x^3}$).

Пример алгоритма, содержащего алгоритмическую структуру «неполное» ветвление (обход)

Задача. Составить алгоритм выбирающий максимальное из двух чисел **x** и **y**. Присвоить его значение переменной **z**.

Блок-схема алгоритма

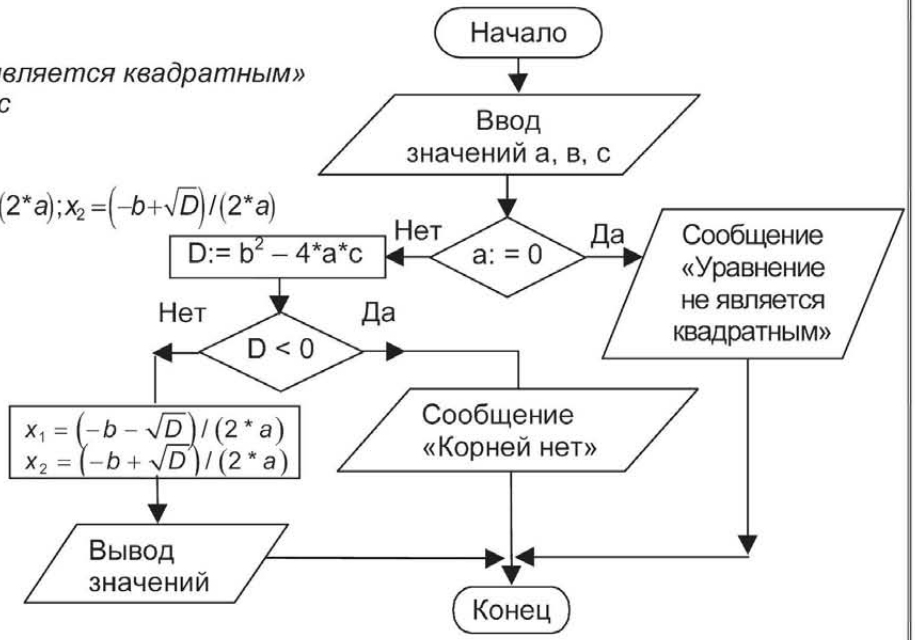


Комментарий. Просматривая числа по очереди, мы «увидим» сначала только первое число **x**. Будем считать его максимальным и присвоим **z** значение **x**. Затем сравним полученное значение **z** со вторым числом **y**. Если окажется, что наше значение **z** меньше **y**, то переменной **z** присвоим новое значение – **y**, в противном случае – оставим **z** без изменения.

Пример разветвляющегося алгоритма
«Решение квадратного уравнения»

На алгоритмическом языке	*В виде блок-схемы:*

начало
  ввод *a, в, c*
    если *a* =
      то *«Уравнение не является квадратным»*
      иначе *D:= $b^2 - 4*a*c$*
    если *D < 0*
      то *«Корней нет»*
      иначе $x_1 = \left(-b - \sqrt{D}\right) / (2*a); x_2 = \left(-b + \sqrt{D}\right) / (2*a)$
конец



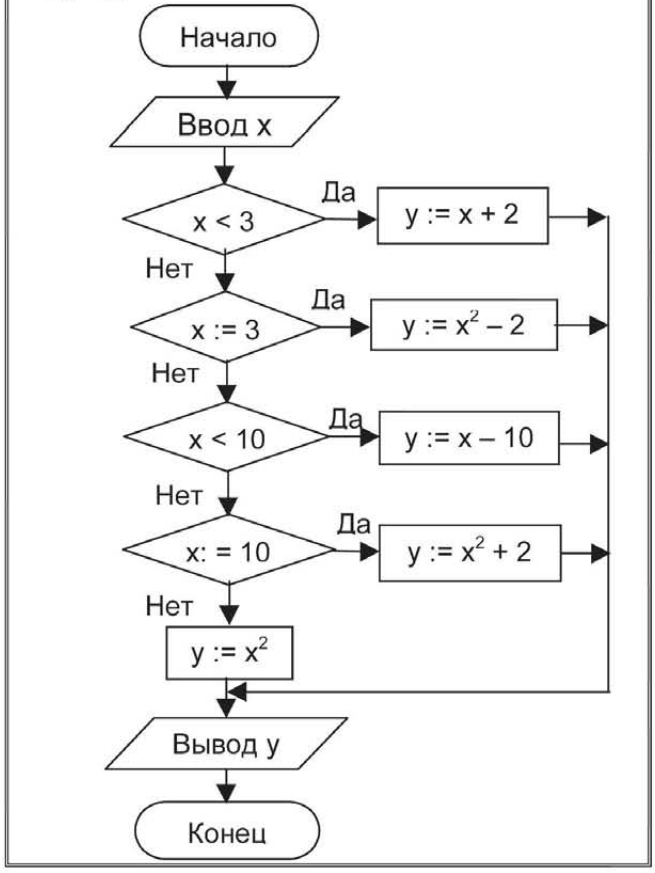
Пояснения к алгоритму: После ввода значений коэффициентов производится проверка значения *a*. При ***a = 0*** уравнение становится линейным, – выполнение алгоритма прекращается. В противном случае (когда ***a*** отлично от **0**) производится вычисление дискриминанта, а затем – его проверка. Если ***D < 0***, то уравнение не имеет корней, выполнение алгоритма прекращается. В противном случае (***D > 0***) производится вычисление корней $\mathbf{x_1}$ и $\mathbf{x_2}$. Пользователю предъявляется полученный результат и выполнение алгоритма на этом заканчивается.

Пример алгоритма с использованием ветвления типа «Выбор»

Вычисление значения функции:

$$y = \begin{cases} x + 2, & \text{если } x < 3; \\ x^2 - 2, & \text{если } x = 3; \\ x - 10, & \text{если } 3 < x < 10; \\ x^2 + 2, & \text{если } x = 10; \\ x^2, & \text{если } x > 10. \end{cases}$$

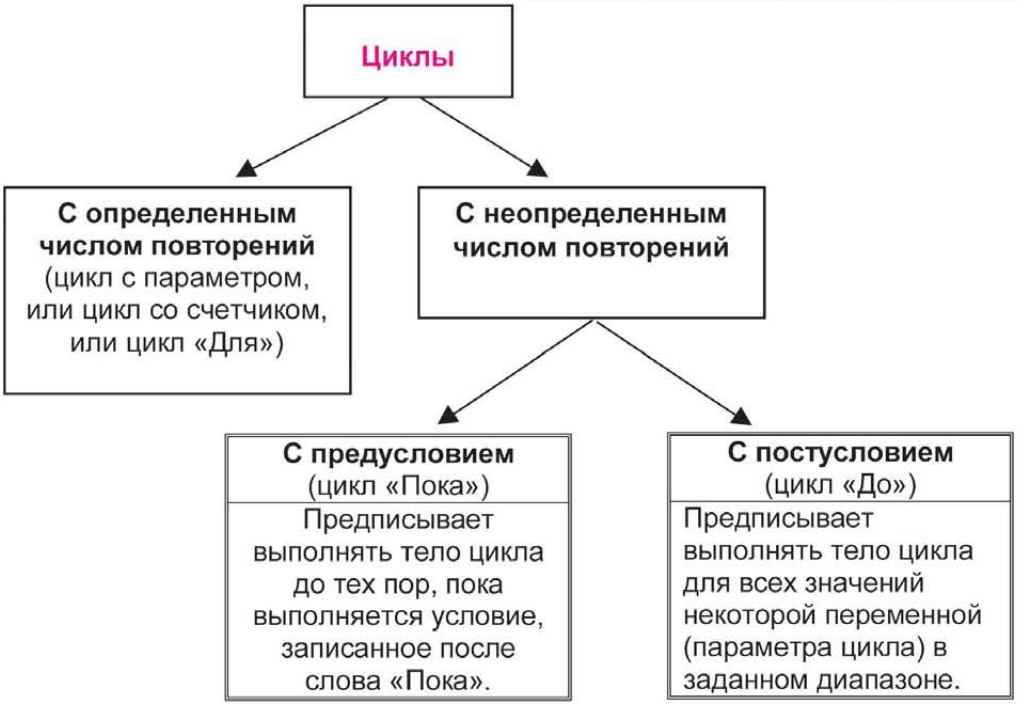
Блок-схема алгоритма с алгоритмической структурой «Выбор – иначе»:



Циклические алгоритмические структуры

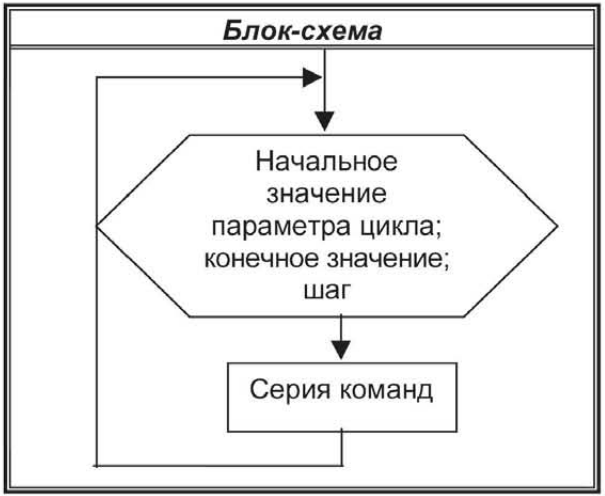
Алгоритмическая структура «Цикл» ***обеспечивает многократное выполнение некоторой последовательности действий,*** **которая называется телом цикла.**

Иногда внутри тела цикла бывает необходимо организовать внутренний цикл.
Такая структура называется **вложенные циклы**.



Цикл с параметром

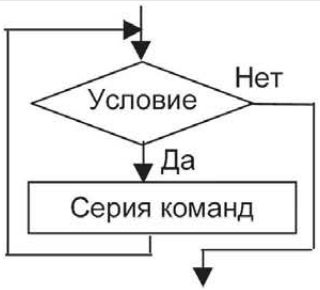
Блок-схема



Словесное описание

1. Вычисляются значения выражений, определяющие начальное и конечное значения параметра цикла;
2. параметру цикла присваивается начальное значение;
3. параметр цикла сравнивается с конечным значением;
4. если параметр цикла превосходит (при положительном шаге) конечное значение параметра цикла (или, наоборот, меньше конечного значения параметра цикла при отрицательном шаге), переход к п. 8, иначе к следующему пункту;
5. выполняется тело цикла;
6. параметр цикла автоматически изменяется на значение шага;
7. переход к п. 3;
8. конец цикла.

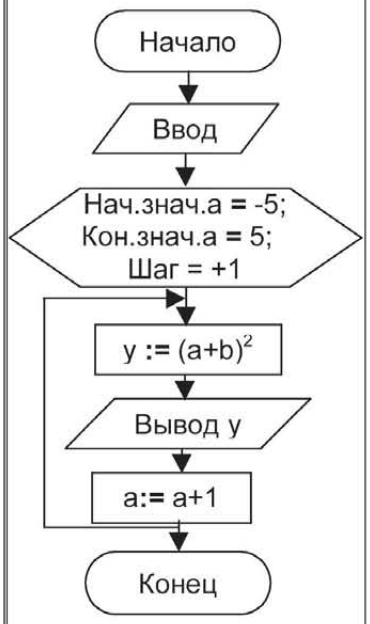
Циклы с условием

Цикл с предусловием (цикл «Пока»)	Цикл с постусловием (цикл «До»)
Блок-схема	
	

Словесное описание:	
1. Вычисляется значение логического выражения (проверяется истинность заданного условия); 2. если значение логического выражения *истинно*, переход к следующему пункту, иначе – переход к п. 5; 3. выполняется серия команд (тело цикла); 4. переход к п. 1; 5. конец цикла.	1. Выполняется серия команд (тело цикла); 2. вычисляется значение логического выражения; 3. если значение логического выражения *истинно*, переход к п. 1), иначе – к следующему пункту; 4. конец цикла.

Пример использования алгортмической структуры «Цикл»

в задаче расчета значений функции по формуле $y = (a + b)^2$
при значениях ***a*** из интервала [-5, 5] с шагом +1.



Для решения данной задачи можно использовать **цикл с параметром** (см. блок-схему слева).

Параметром цикла является значение переменной **a**, для котрой заданы начальное (-5) и конечное значение (5),
а также шаг изменения (+1).

Тело цикла состоит из серии команд:

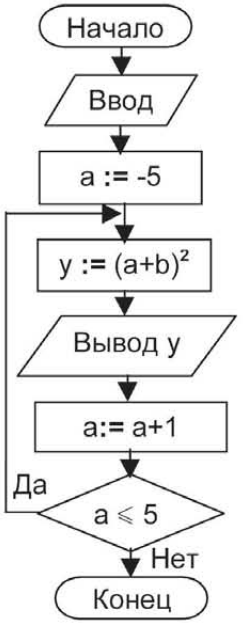
y := (a + b)²;

Вывод y;

a:= a + 1.

Продолжение цикла или же выход из него контролируется с помощью проверки условия a ⩽ 5.

Данный цикл с параметром можно рассматривать и как **цикл с постусловием**, что хорошо видно на блок-схеме алгоритма справа.



Определение результата выполнения алгоритма, записанного на алгоритмическом языке или представленного в виде блок-схемы

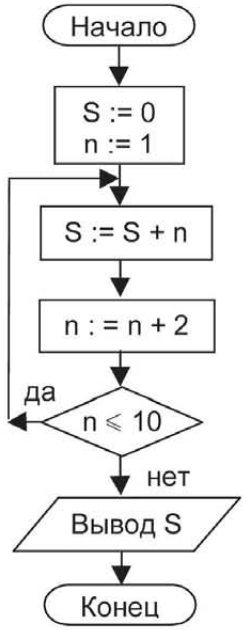
Алгоритм на алгоритмическом языке:

```
алг   сумма чисел (цел S)
      рез   S
нач   нат   n
      S := 0
      для n от 1 до 10 шаг 2
      нц
      S := S + n
      кц
кон
```

Изменение значений переменных n и S в цикле:

Сколько раз выполнен цикл	Значение **n**	Оператор присваивания, который будет выполняться в теле цикла	Значение **S**
1	1	S := 0 + 1	1
2	3	S:= 1 + 3	4
3	5	S := 4 + 5	9
4	7	S := 9 + 7	16
5	9	S := 18 + 9	27

Алгоритм в виде блок-схемы



Представленный алгоритм является алгоритмом вычисления **суммы натуральных** (целых положительных) чисел из интервала от 1 до 10. Результат выполнения алгоритма **S=27.**

Переменная в программировании (тип, имя, значения)

В алгоритмических языках программирования **переменные** предназначены для хранения и обработки данных в программах.

Переменной называется величина, значение которой может *меняться* в процессе выполнения программы.

С понятием переменной связаны следующие характеристики (атрибуты):

Имя	Тип	Значение
Определяет обозначение переменной и ее место в памяти. Имя любой переменной (идентификатор) уникально и *не может меняться* в процессе выполнения программы. Имя переменной должно обязательно начинаться с буквы.	**Определяет множество допустимых значений переменной и множество применимых к ней операций, объем занимаемой памяти и способ представления в памяти.** Простейший способ задания типа переменной – использование в идентификаторе переменной определенного суффикса (специального значка), который приписывается к имени переменной.	**Это динамическая характеристика, которая может меняться многократно в ходе исполнения алгоритма.** Во время выполнения алгоритма в каждый конкретный момент величина имеет какое-то значение или не определена. Значениями переменных могут быть данные различных типов (целые или вещественные числа, последовательности символов, логические значения и т. д.).

Операция присваивания

Любая переменная может получить или изменить свое значение с помощью **оператора присваивания**.
В общем виде оператор присваивания можно записать так:
<имя переменной> : = <выражение>

Оператор выполняется следующим образом:	**Свойства операции присваивания:**
Вычисляется выражение в правой части оператора. После этого переменная, указанная в левой части, получает вычисленное значение. При этом тип выражения должен быть совместим по присваиванию с типом переменной, а значения всех переменных, входящих в выражение, были определены.	1. Пока переменной не присвоено значение, она остается неопределенной; 2. значение, присвоенное переменной, сохраняется в ней вплоть до выполнения следующего присваивания этой переменной нового значения; 3. новое значение, присвоенное переменной, заменяет ее предыдущее значение.

Основные понятия алгоритмического программирования (данные, операторы, функции, процедуры и т. д.)

<table>
<tr><td rowspan="3">Данные</td><td colspan="2">Данные – величины, обрабатываемые программой.</td></tr>
<tr><td>Переменные</td><td>Константы</td></tr>
<tr><td>Данные, значение которых может меняться в процессе выполнения программы</td><td>Данные, значение которых не изменяется в процессе выполнения программы.</td></tr>
<tr><td>Имена (идентификаторы)</td><td colspan="2">Используются для обозначения объектов в программе (переменных, массивов, функций и др.). Каждая переменная имеет свое уникальное имя.</td></tr>
<tr><td>Операции</td><td colspan="2">– арифметические операции +, – , * , / и др. ;
– логические операции и, или, не;
– операции отношения <, >, ⩽, ⩾, =, < >;
– операция сцепки («присоединения», «конкатенации») символьных значений друг с другом с образованием одной длинной строки; изображается знаком «+».</td></tr>
<tr><td>Выражения</td><td colspan="2">Предназначаются для выполнения необходимых вычислений, состоят из констант, переменных, указателей функций (например, exp(x)), объединенных знаками операций.
Выражения записываются в виде линейных последовательностей символов (без подстрочных и надстрочных символов, «многоэтажных» дробей и т. д.), что позволяет вводить их в компьютер, последовательно нажимая на соответствующие клавиши клавиатуры.</td></tr>
</table>

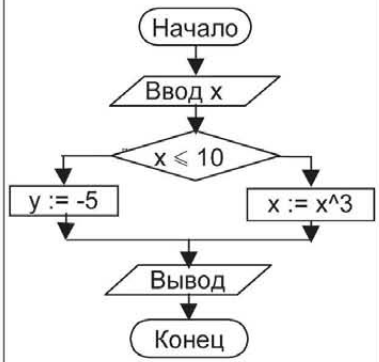
Операторы (команды)	Текст любой программы состоит из отдельных предложений. Обычно они называются **операторами**. Как правило, оператор содержит имя и данные и указывает, какую операцию и над какими величинами надо выполнить. Одна строка программы может содержать один или несколько операторов. **В состав операторов входят:** **– ключевые слова;** **– данные;** **– выражения и т. д.**
Функции	Для наиболее употребительных функций программы их вычисления записаны в память компьютера, в библиотеки программ, а сами функции включены в состав языков программирования. Такие функции называются **встроенными** (или стандартными). Для вычисления таких функций в программе достаточно указать имя функции и значение ее аргумента. **Каждый язык программирования имеет свой набор стандартных функций.**
Процедуры	**Процедура – это самостоятельная программная единица, которая выполняется по команде из другой программы или процедуры.** Процедура оформляется определенным образом, к ней можно обращаться из разных точек программы любое число раз. При этом такая процедура может решать каждый раз одну и ту же задачу с разными значениями исходных данных.

Пример разработки алгоритма и программы для решения задачи, содержащей команды ветвления (операторы ветвления).

Вычисление значения функции y(x) для заданного x:

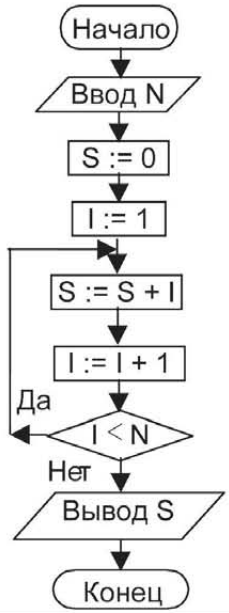
$$y(x) = \begin{cases} -5, \text{ при } x \leqslant 10 \\ x^3, \text{ при } x > 10 \end{cases}$$

Блок-схема алгоритма	*Бейсик*	*Паскаль*

	REM вычисление значения функции y(x)
	INPUT «Введите значение x»; x
	IF X <= 10 THEN y = -5
	ELSE y = x^3
	PRINT «y =»; y
	END
		program zadacha (input, output);
		{вычисление значения функции y(x)}
		var
		x, y: real;
		begin
		writeln (‘Введите значение x’);
		readln (x);
		if x <= 10 then y:= -5
		else y := x*x*x;
		writeln (‘y =’ , y)
		end.

Пример разработки алгоритма и программы для решения задачи, содержащей команду повторения (оператор цикла) «Вычисление суммы натуральных чисел от 1 до N».

Блок-схема алгоритма	*Бейсик*	*Паскаль*

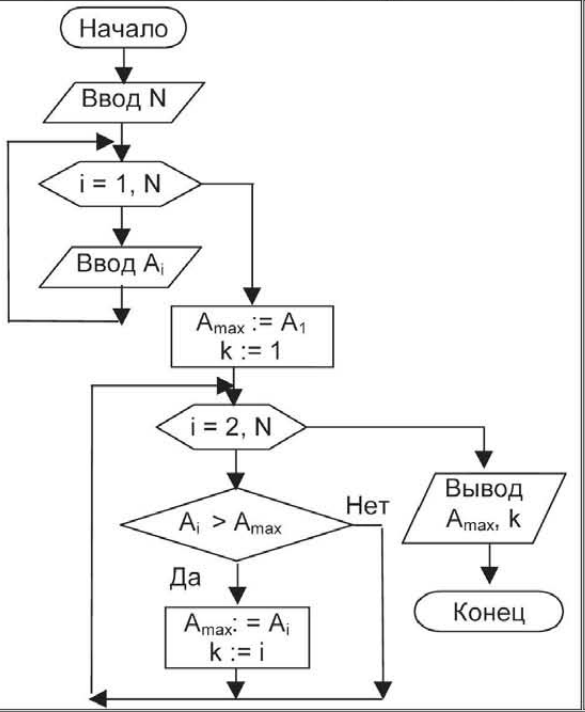
	INPUT «Введите натуральное число N»; N
	S = 0
	FOR I = 1 TO N
	S = S + I
	NEXT I
	PRINT «S = » ; S
	END
		program zadacha (input, output)
		var
		n, i, s : integer
		begin
		writeln ('Введите натуральное число');
		readln (n);
		s := 0;
		for i := 1 to n do s := s + i;
		writenln ('Сумма натуральных чисел =', s)
		end.

	Комментарии Натуральное число N вводится с клавиатуры в процессе выполнения программы. Искомая сумма обозначается идентификатором S (s). Параметром цикла является переменная I (i), которая изменяется от 1 до N (n) с шагом 1, за счет чего и происходит перебор всех значений натуральных чисел от 1 до N (n). По окончании вычислений результат печатается на экране компьютера.	

Задача определения максимального элемента в одномерном массиве целых чисел

***Исходные данные*: N (n)** – количество элементов в массиве, **A** – сам массив.

Результаты: **K** – номер (индекс) того элемента в массиве, который оказался максимальным, **AMAX (amax)** – значение максимального элемента.

Блок-схема алгоритма	*Комментарии*
	***Промежуточные величины:*** **I (i)** – текущий индекс элемента (переменная цикла). Поиск наибольшего значения осуществляется последовательным сравнением значений. В качестве начального значения наибольшего элемента массива выбирается значение первого элемента, а сам цикл, выполняющий сравнение, начинается со второго элемента.

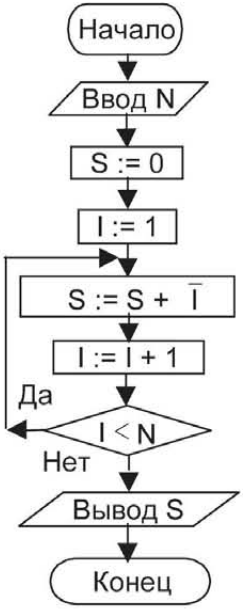
Бейсик:	*Паскаль:*

CLS
INPUT «N = » ; N
DIM A(N)
FOR I = 1 TO N ‘Ввод массива A
PRINT «A(“ ; I ; ”) = »;
INPUT A(I)
NEXT I
AMAX = A(1) : K = 1 ‘Поиск максимального элемента
FOR I = 2 TO N
IF A(I) > AMAX THEN AMAX = A(I) : K = I
NEXT I
PRINT «Наибольший элемент номер» ; K
PRINT «его значение»; AMAX
END
	program max
	uses crt;
	type mas = array [1..20] of real;
	var a : mas;
	i, n : integer;
	k : integer;
	amax : real;
	begin
	ClrScr;
	write (‘Введите N = ‘);
	readln(n);
	for i := 1 to n do {Ввод элементов массива A}
	begin
	write (‘a[‘ , i, ‘]=’);
	readln (a[i])
	end;
	amax := a[1]; k := 1; {Поиск максимального элемента}
	for i := 2 to n do
	if a[i] > amax then
	begin
	amax := a[i];
	k := i
	end;
	writeln;
	writeln (‘Наибольший элемент номер’, k);
	writeln (‘его значение’ , amax : 5 : 1);
	readln
	end.

Пример решения расчетной задачи с использованием математических функций

«Вычисление значения выражения $s = 1 + \sqrt{2} + \sqrt{3} + \sqrt{4} + \ldots + \sqrt{n}$ для заданного натурального числа n».

Блок-схема алгоритма	*Бейсик*	*Паскаль*

	INPUT «Введите натуральное число n»; N
	S = 0
	FOR I = 1 TO N
	S = S + SQR (I)
	NEXT I
	PRINT «S = »; S
	END
		program zadacha (input, output);
		{вычисление суммы квадратных корней натуральных чисел}
		var
		n, i : integer;
		s : real
		begin
		writeln (‘Введите натуральное число’);
		readln (n);
		s := 0;
		for i = 1 to n do s := s + sqrt (i);
		writeln (‘Сумма чисел =’ , s)
		end.

Комментарии

Данная задача реализует пример классического циклического алгоритма. Интерес здесь представляет использование стандартной математической функции «корень квадратный».

В языках программирования существуют определенные правила записи таких функций. В частности, языки Бейсик и Паскаль требуют, чтобы рядом с именем стандартной функции (SQR или sqrt – соответственно) в скобках был указан аргумент функции, в нашем случае – это подкоренное выражение.

Пример разработки алгоритма (программы)
на обработку данных строкового типа

Программа, определяющая, является ли данное слово «перевертышем», т. е. читается ли оно одинаково слева направо и справа налево.

Школьный алгоритмический язык

```
алг Перевертыш (арг лит Slovo, рез лит Otvet)
        надо  Otvet = «Перевертыш», если Slovo совпадает с собой после переворачивания
нач цел            Dlina, i,   лог Flag
        Dlina := длин(Slovo)
        i := 1; Flag := да
        нц пока (i <= Dlina / 2) и Flag
        Flag := (Slovo[i] = Slovo[Dlina – i + 1])
        i := i + 1
        кц
        если Flag
                то Otvet:= «Перевертыш»
                иначе Otvet:= «Не перевертыш»
        все
кон
```

Комментарии

При составлении алгоритма использованы функция **длин(Slovo),** возвращающая количество символов в строке Slovo и операция **Slovo[i]**, которая «вырезает» из строки Slovo символ с порядковым номером **i**.

QBasic

```
CLS
INPUT «Введите слово : » , SLOVO$
Dlina = LEN(SLOVO$)
   ' Сравниваются пары букв: первая буква с последней,
    ' вторая буква с предпоследней и т.д.
i = 1
Flag = 0
WHILE (i <= Dlina / 2) AND (Flag = 0)                    'цикл до первой несовпавшей пары
букв
  Letter1$ = MID$ (SLOVO$, i, 1)      'первая буква пары
  Letter2$ = MID$ (SLOVO$, Dlina – i + 1, 1)   'вторая буква пары
  IF   Letter1$ = Letter2$   THEN  i = i + 1
  ELSE  Flag = 1
WEND
PRINT
PRINT   «Ответ : слово » ;  SLOVO$;
IF   Flag = 0    THEN PRINT «– перевертыш.»  ELSE
          PRINT « – не перевертыш. »
END
```

Комментарии

При составлении программы использованы:
функция **LEN(SLOVO$),** которая возвращает количество символов в строке SLOVO$,
и функция **MID$ (SLOVO$, i, 1)**, которая возвращает один символ строки SLOVO$
с номером **i.**

Turbo Pascal

```
Program TurnOver;
        Uses Crt;
        Var  Slovo   : String;
             Dlina, i : Integer;
             Flag    : Boolean;
BEGIN
  ClrScr;
  Write ('Введите слово : ');  ReadLn (Slovo);
  Dlina := Length (Slovo);
  {Сравниваются пары букв: первая буква с последней,}
  {вторая буква с предпоследней и т. д.}
   i := 1;
   Flag := TRUE;
   While  (i <= Dlina / 2) and Flag   do                    {цикл до первой несовпавшей}
     Begin    {пары букв (если такая есть)}
       Flag := (Slovo[i] = Slovo[Dlina – i + 1]);
       i := i+1
     end;
  WriteLn; Write ('О т в е т : слово ', Slovo);
    If  Flag  then  WriteLn (' – перевертыш. ')
       else WriteLn(' – не перевертыш.');
    ReadLn
END.
```

Комментарии

При составлении программы использована функция **Length (Slovo),** которая возвращает количество символов в строке Slovo.

Технология решения задач с помощью компьютера

Работа по решению прикладной задачи на компьютере проходит через следующие этапы:

– постановка задачи;
– математическая формализация;
– построение алгоритма;
– составление программы на языке программирования;
– отладка и тестирование программы;
– проведение расчетов и анализ полученных результатов.

Эту последовательность называют **технологической цепочкой** решения задачи на компьютере.

Постановка задачи	Если задача конкретная, то под постановкой задачи понимают ответ на два вопроса: какие исходные данные известны и что требуется определить. Если задача обобщенная, то при постановке задачи понадобится еще ответ на третий вопрос: какие данные допустимы. **Таким образом, постановка задачи включает в себя:** сбор информации о задаче; формулировку условия задачи; определение конечных целей решения задачи; определение формы выдачи результатов; описание данных (их типов, диапазонов величин, структуры и т. п.)
Математическая формализация	На этом этапе строится **математическая модель – система математических соотношений – формул, уравнений, неравенств и т. д., отражающих существенные свойства объекта или явления.** Однако далеко не всегда удается найти формулы, явно выражающие искомые величины через данные. В таких случаях используются математические методы, позволяющие дать ответы той или иной степени точности. В случае большого числа параметров, ограничений, возможных вариантов исходных данных модель явления может иметь очень сложное математическое описание, поэтому часто построение математической модели требует упрощения требований задачи. Необходимо выявить самые существенные свойства объекта, явления или процесса, закономерности, внутренние связи, роль отдельных характеристик. Выделив наиболее важные факторы, можно пренебречь менее существенными.

Построение алгоритма	Наиболее эффективно математическую модель можно реализовать на компьютере в виде алгоритмической модели. Для этого может быть использован язык блок-схем или какой-нибудь псевдокод, например, учебный алгоритмический язык. Разработка алгоритма включает в себя **выбор метода проектирования алгоритма**; **выбор формы записи алгоритма** (блок-схемы, псевдокод и др.); **выбор тестов и метода тестирования; проектирование самого алгоритма**.
Составление алгоритма на языке программирования. Отладка и тестирование программы	Первые три этапа – это работа без компьютера. Дальше следует собственно программирование на определенном языке в определенной системе программирования. **Программирование включает в себя следующие виды работ:** выбор языка программирования; уточнение способов организации данных; запись алгоритма на выбранном языке программирования. **Под отладкой программы понимается процесс испытания работы программы и исправления обнаруженных при этом ошибок.** Обнаружить ошибки, связанные с нарушением правил записи программы на языке программирования (синтаксические и семантические ошибки), помогает используемая система программирования. Пользователь получает сообщение об ошибке, исправляет ее и снова повторяет попытку исполнить программу. Проверка на компьютере правильности алгоритма производится с помощью тестов. **Тест – это конкретный вариант значений исходных данных, для которого известен ожидаемый результат.** На тестах проверяется правильность реализации программой запланированного сценария. Однако при решении некоторых задач можно не составлять программу на языке программирования, а использовать современные приложения (электронные таблицы, системы управления базами данных и пр.). В этом случае не понадобятся отладка и тестирование программы.
Проведение расчетов и анализ получаемых результатов	На этом этапе производится **анализ результатов решения задачи** и в случае необходимости – **уточнение математической модели** (с последующей корректировкой алгоритма и программы). Программы, имеющие большое практическое или научное значение, используются длительное время. Иногда даже в процессе эксплуатации программы могут исправляться, дорабатываться.

Понятие модели. Материальные и информационные модели

Модели позволяют представить в наглядной форме объекты и процессы, недоступные для непосредственного восприятия.

Разные науки исследуют объекты и процессы под разными углами зрения и строят различные типы моделей. Один и тот же объект иногда имеет множество моделей, а разные объекты могут описываться одной моделью.

Модель никогда не характеризует объект полностью. В процессе построения модели выделяют главные, наиболее существенные для данной модели свойства, которые зависят от цели моделирования.

М о д е л ь – это некий новый объект, который отражает существенные особенности изучаемого объекта, явления или процесса.

Все модели можно в принципе разбить на два класса:
материальные (или предметные) и **информационные** (знаковые).

Материальные модели

Воспроизводят геометрические, физические и другие свойства объектов в материальной форме.

Самый простой пример материальной модели – это детские игрушки, играя которыми маленький человек получает первое представление об окружающем мире. Кроме этого, примерами моделей могут служить глобус, различные анатомические муляжи, которые используются на уроках биологии, модели молекул и кристаллических решеток (используются при изучении строения вещества), макет застройки жилого микрорайона, макет самолета и т. п.

Информационные модели

Представляют собой объекты и процессы в форме рисунков, схем, чертежей, таблиц, формул, текстов и т. д.

Широко известны такие школьные информационные модели, как рисунок цветка в учебнике ботаники, географическая карта, физическая формула (физика), блок-схема алгоритма (информатика), периодическая система элементов Менделеева (химия), уравнение (математика).

Все информационные модели представляют объекты и процессы в *образной* или *знаковой* форме.

Образные модели представляют собой зрительные образы объектов, зафиксированные на каком-либо носителе информации (бумаге, фото-, кинопленке и т. д.).

Знаковые информационные модели строятся с использованием различных языков (знаковых систем). Знаковая модель может быть представлена в виде текста (например, программа на каком-то языке программирования), формулы (например, описывающей процесс расширения газа при постоянной температуре), таблицы (например, периодической таблицы элементов Менделеева).

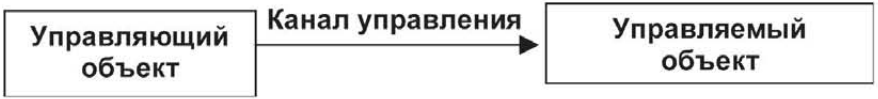
Среди основных целей информационного моделирования можно выделить: создание объектов с заданными свойствами, определение последствий воздействия на объект и принятие правильного решения, прогнозирование поведения объекта моделирования, а также достижение эффективного управления объектом или процессом (принятие управляющих решений).

Замкнутые и разомкнутые системы управления

Управление – это *целенаправленное* взаимодействие объектов, одни из которых являются управляющими, а другие – управляемыми.

В том случае, когда **управляющий объект посылает свои команды исполнительному объекту, без учета его состояния, воздействия передаются только в одном направлении.** Такая система называется **разомкнутой**.

Информационную модель
разомкнутой системы управления
можно наглядно представить с помощью следующей схемы:



Более совершенные системы управления отслеживают результаты деятельности управляемой системы.

В таких системах дополнительно появляется еще один информационный поток – от объекта управления к системе управления; его принято называть ***обратной связью***.

По каналу обратной связи передаются сведения о состоянии объекта и степени достижения (или, наоборот, не достижения) цели управления.

В этом случае система управления называется **замкнутой**.

Информационная модель **замкнутой системы управления** наглядно представлена на схеме:

Главный принцип управления в замкнутой системе – выдача управляющих команд в зависимости от получаемых сигналов обратной связи.

В такой системе управляющий объект стремится скомпенсировать любое отклонение управляемого объекта от состояния, предусмотренного целями управления.

Обратную связь, при которой управляющий сигнал стремится уменьшить (скомпенсировать) отклонение от некоторой поддерживаемой величины, называется **отрицательной.**

Изучением общих закономерностей процессов управления занимается специальная наука, которая называется кибернетикой.

Растровая и векторная графика

Графический редактор – это программа, предназначенная для создания, редактирования и просмотра графических изображений.

В современных компьютерах существует два принципиально различных способа хранения изображений: **растровый** и **векторный**. Соответственно и графические редакторы можно разделить на две категории: **растровые** и **векторные**.

	Векторная графика	*Растровая графика*
Назначение	Векторные графические изображения являются оптимальным средством хранения высокоточных графических объектов, таких как чертежи, схемы и пр., для которых имеет значение сохранение четких и точных контуров.	Растровые графические редакторы подходят для обработки фотографий и рисунков. Растровые изображения формируются в процессе преобразования графической информации из аналоговой формы в цифровую (например, в процессе сканирования рисунков и фотографий, при использовании цифровых и фотокамер и т. д.). Растровые изображения можно получить и непосредственно в программах растровой или векторной графики путем преобразовании векторных изображений.
Принцип формирования изображения	**Векторные изображения формируются из таких объектов как точка, линия, окружность, прямоугольник и пр.** Эти объекты хранятся в памяти компьютера в виде графических примитивов и описывающих их математических формул.	**В растровом формате изображение задается по точкам, как мозаика. Эти точки называют пиксель (pixel).** Цвет каждого пикселя задается независимым образом. Растр – это дискретная структура, то есть всегда можно выделить определённые элементы.

	Векторная графика	Растровая графика
Основные достоинства	◆ Изменение масштаба без потери качества и практически без увеличения размеров исходного файла; ◆ огромная точность (до сотой доли микрона); ◆ небольшой размер файла по сравнению с растровыми изображениями; ◆ прекрасное качество печати; ◆ возможность редактирования каждого элемента изображения в отдельности.	Растровое изображение имеет большие преимущества при работе с фотореалистичными объектами, например, сценами природы или фотографиями людей. Наш мир по идее растровый. И его объекты трудно представить в векторном, то есть математическом, представлении, как это происходит в случае работы с векторными изображениями.
Недостатки	Не могут обеспечить высокую точность передачи градаций цветов и полутонов.	Масштабирование растрового изображения, по причине его дискретности, приводит к потере части информации, вызывает необратимые потери качества изображения. При попытке изменить размеры рисунка его контуры и цветопередача заметно искажаются. Кроме того, растровые изображения занимают гораздо больше места в памяти компьютера в сравнении с векторными.
Примеры соответствующих графических редакторов	Для обработки векторной графики используются векторные графические редакторы, например, такие как, CorelDraw, Adobe Illustrator, MS Draw.	Графические редакторы, работающие с растровым изображением: PhotoPaint, PhotoShop, PhotoFinish, Picture Man, Paint и др.

Текстовые редакторы, текстовые процессоры и их основные возможности

Текстовые редакторы (процессоры) предназначены для создания, редактирования, форматирования, сохранения во внешней памяти и печати текстовых документов. Обычно **текстовыми редакторами** принято называть программы, выполняющие простейшие операции по редактированию текста, а **процессорами** – программы, обладающие расширенными по сравнению с редакторами возможностями для компьютерной обработки текста.

Основные функции текстовых процессоров:

- создание документов;
- редактирование документов: перемещение по тексту, вставка и замена символов, удаление, перемещение, копирование, поиск и замена фрагментов текста, отмена команд; вставка фрагментов других документов или целых документов и т. д.;
- сохранение документов во внешней памяти (на дисках) и чтение из внешней памяти в оперативную;
- форматирование документов, т. е. выполнение преобразований, изменяющих форму (внешний вид) документа: оформление отдельных символов и абзацев, страниц, документа в целом – изменение длины строки, межстрочного расстояния, выравнивания текста, изменение шрифта, его размера, применение различного начертания шрифтов и т. д.;
- печать документов (или их некоторой части);
- автоматическое составление оглавлений и указателей в документе;
- создание и форматирование таблиц;
- внедрение в документ рисунков, формул и др.;
- проверка пунктуации и орфографии.

Обычно текстовые процессоры предусматривают две основные операции изменения формата документа:

- форматирование произвольной последовательности символов (от одного до любого количества, чаще всего эта последовательность предварительно выделяется);
- форматирование абзацев.

Основные элементы текстового документа:

◆ символ – минимальная единица текстовой информации;
◆ слово – произвольная последовательность букв и цифр, ограниченная с двух сторон служебными символами;
◆ строка – произвольная последовательность символов между левой и правой границами абзаца;
◆ предложение – произвольная последовательность слов, завершающаяся точкой;
◆ абзац – часть текста, которая завершается специальным символом конца абзаца, при этом допускаются пустые абзацы;
◆ страницу составляют строки и абзацы, таблицы и внедренные в документ объекты;
◆ наиболее крупной единицей является собственно документ, где все составляющие его абзацы определенным образом структурированы, снабжены при необходимости заголовками, выстроена иерархия структурных разделов.

При форматировании символов можно изменить:	**При форматировании абзацев можно изменить:**
◆ шрифт; ◆ начертание шрифта (полужирный, курсив, подчеркнутый); ◆ размер шрифта; ◆ межсимвольный интервал; ◆ применить к символам эффекты (нижний, верхний индекс, малые строчные буквы и т. д.)	◆ способ выравнивания строк абзаца (влево, вправо, по центру, по ширине); ◆ отступ в красной строке абзаца; ◆ ширину и положение абзаца на странице межстрочное расстояние и расстояние между соседними абзацами; ◆ создать специальные абзацы (маркированные или нумерованные списки и т. д.).

В современных редакторах реализован механизм встраивания и внедрения объектов **OLE (Object Linking Embedding),** что позволяет копировать и вставлять объекты из одного приложения в другое. Например, в текстовый документ можно встроить *изображения, анимацию, звук и даже видеофрагменты* и таким образом из обычного документа получить мультимедийный.

Обычно в состав текстовых процессоров включаются специальные программные модули, которые служат для проверки *орфографии и синтаксиса*. Такие системы содержат **словари и грамматические правила** для нескольких языков, что позволяет исправлять ошибки в многоязычных документах. Кроме того, в их составе, как правило, есть функция **Автозамена**, которая автоматически исправляет наиболее часто встречающиеся опечатки.

Электронные таблицы

Электронные таблицы (табличные процессоры) – это программа обработки числовых данных, представленных в виде таблиц.

Электронная таблица позволяет хранить в табличной форме большое количество исходных данных, результатов, а также связей между ними (математических или логических соотношений). При изменении исходных данных все результаты автоматически пересчитываются.

Рабочее поле электронной таблицы **состоит из столбцов и строк**.

Заголовки столбцов обычно обозначаются буквами латинского алфавита (сочетаниями букв): **A, G, AB, …**, а заголовки строк – числами: **1, 16, 278, …**

На месте пересечения столбца и строки таблицы образуется **ячейка**. Это *минимальный* элемент электронной таблицы, над которым можно выполнять те или иные операции.

A	B	C	D	E	F	…
1						
2						
3						
…						

Каждая ячейка таблицы имеет свой собственный **адрес**. **Адрес** ячейки электронной таблицы состоит из **заголовка столбца и номера строки**, например: Al, F123, D7 (например, в представленной таблице адрес выделенной ячейки – D 2).

Ячейка, над содержимым которой производятся какие-то действия, выделяется рамкой и называется **активной**.

Электронные таблицы позволяют работать с тремя основными **типами данных**: **числами, текстами и формулами**.

В формулах могут использоваться имена ячеек (ссылки на адреса ячеек). **Существуют два основных типа ссылок**: **относительные** и **абсолютные**. Различия между ними проявляются при копировании формулы из активной ячейки в другую ячейку.

Относительная ссылка	Абсолютная ссылка
Используется для указания адреса ячейки, вычисляемого относительно ячейки, в которой находится формула. При перемещении или копировании, вставке или удалении строки (столбца) относительные ссылки автоматически обновляются в зависимости от нового положения формулы. В силу этого сохраняется правильность расчетов при любых указанных выше действиях над ячейками, содержащими формулы. Относительные ссылки имеют следующий вид: D1, B7.	Используется в тех случаях, когда необходимо, чтобы при изменении местоположения формулы адрес ячеек, используемых в формуле, не изменялся. В абсолютных ссылках перед неизменяемым значением адреса ячейки ставится знак доллара (например, F1).

При изменении исходных значений, входящих в формулу, результат пересчитывается **автоматически**.

Электронные таблицы позволяют:

- выполнять сортировку данных по возрастанию или убыванию; а также сортировать данные по нескольким столбцам, при этом может быть задана последовательность сортировки столбцов;
- осуществлять поиск данных в соответствии с указанными условиями, которые принято называть **фильтрами**; фильтры задаются с помощью условий поиска (больше, меньше, равно и т. д.) и значений; в результате поиска будут найдены те ячейки, в которых содержатся данные, удовлетворяющие заданному фильтру;
- представлять числовые данные в наглядном виде с помощью графиков и диаграмм.

Базы данных и системы управления базами данных

База данных – это совокупность определенным образом организованной информации, позволяющая упорядоченно хранить данные о группе объектов, обладающих одинаковым набором свойств.

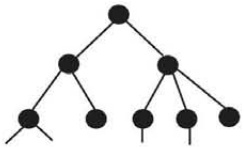
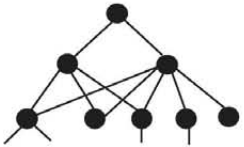
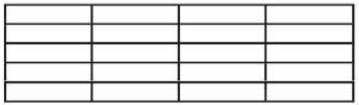
Базами данных являются, например, различные справочники, энциклопедии, каталоги библиотек, картотеки кадрового состава учреждений и т. п.

Для создания баз данных, а также выполнения операции поиска и сортировки данных предназначены специальные программы – **системы управления базами данных (СУБД)**. Система управления базой данных позволяет обрабатывать обращения к базе данных, поступающие от прикладных программ конечных пользователей.

Современная система управления базами данных выполняет следующие функции:

- ✓ Ввод информации в базу данных и обеспечение ее логического контроля.
- ✓ Возможность исправления информации.
- ✓ Удаление устаревшей информации.
- ✓ Контроль непротиворечивости данных.
- ✓ Защита данных от разрушения.
- ✓ Поиск информации с заданными свойствами.
- ✓ Автоматическое упорядочивание информации в соответствии с определенными требованиями.
- ✓ Обеспечение коллективного доступа к данным нескольких пользователей одновременно.
- ✓ Защита от несанкционированного доступа к данным.

Информация в базах данных хранится в упорядоченном виде.
Существует несколько различных типов баз данных: иерархические, сетевые и табличные.

Иерархическая		Иерархические базы данных графически могут быть представлены как дерево, состоящее из объектов различных уровней. Верхний уровень занимает один объект, второй – объекты второго уровня и т. д. Между объектами существуют связи, каждый объект может включать в себя несколько объектов более низкого уровня. Иерархической базой данных является Каталог папок Windows.
Сетевая		Сетевая база данных отличается от иерархической тем, что в ней каждый элемент вышестоящего уровня может быть связан одновременно с любыми элементами следующего уровня. Вообще, на связи между объектами в сетевых моделях не накладывается никаких ограничений. Сетевой базой данных фактически является Всемирная паутина глобальной компьютерной сети Интернет. Гиперссылки связывают между собой сотни миллионов документов в единую распределенную сетевую базу данных.
Табличная (реляционная)		Табличная (или реляционная) база данных содержит перечень объектов одного типа, т. е. объектов с одинаковым набором свойств. Такую базу данных удобно представлять в виде двумерной таблицы (а чаще – нескольких связанных между собой таблиц).

Табличная (реляционная база данных)

Табличная (или реляционная) база данных содержит перечень объектов одного типа, т. е. объектов с одинаковым набором свойств. Такую базу данных удобно представлять в виде двумерной таблицы (а чаще – нескольких связанных между собой таблиц).

Столбцы такой таблицы называют полями; каждое поле характеризуется своим **именем** (названием соответствующего свойства) и **типом данных**, отражающих значения данного свойства. При этом каждое поле обладает определенным набором свойств (размер, формат и др.).

Строки таблицы являются записями об объекте; эти записи разбиты на поля столбцами таблицы. Запись базы данных содержит набор значений различных свойств объекта.

В каждой таблице должно быть по крайней мере одно **ключевое поле**, содержимое которого уникально для любой записи в этой таблице. Значения ключевого поля однозначно определяют каждую запись в таблице. Часто в качестве ключевого поля используется поле, содержащее **тип данных Счетчик.**

В реляционных базах данных используется следующие основные типы полей:

Тип поля	Описание
Счетчик	целые числа, которые задаются автоматически при вводе записей и не могут быть изменены пользователем
Числовой	этот тип имеют поля, значения которых могут быть только числами
Символьный (текстовый)	такой тип имеют поля, в которых хранятся символьные последовательности (слова, тексты, коды и пр.), содержащие до 255 символов
Дата/время	дата и время
Логический	значения *Истина* или *Ложь* (или «Да»/«Нет»)

От типа величины зависят те действия, которые можно с ней производить.
Например, с числовыми величинами можно выполнять арифметические операции, а с символьными и логическими – нельзя.

Пример табличной базы данных

База данных «Учащиеся» представляет собой перечень объектов (учеников), каждый из которых имеет фамилию, имя, отчество. В качестве характеристик (свойств) выступает номер личного дела, класс, дата рождения.

№ личного дела	*Класс*	*Фамилия*	*Имя*	*Отчество*	*Дата рождения*
К-25	8«б»	Кузнецов	Александр	Владимирович	13.10.84
П-20	9«б»	Прохоров	Роман	Алексеевич	23.05.85
Л-12	8«а»	Леонов	Виктор	Александрович	07.12.84
У-17	8«б»	Уфимцева	Наталья	Андреевна	24.08.84
М-33	10«а»	Морозова	Татьяна	Сергеевна	24.06.84
Ф-13	9«б»	Федорова	Елена	Станиславовна	01.11.85

Локальные и глобальные компьютерные сети

Под компьютерной сетью понимают систему *распределенных* на территории аппаратных, программных и информационных ресурсов (средств ввода/вывода, хранения и обработки информации), связанных между собой каналами передачи данных.

При этом обеспечивается совместный доступ пользователей к информации (базам данных, документам и т. д.) и ресурсам (жесткие диски, принтеры, накопители CD-ROM, модемы, выход в глобальную сеть и т. д.).

По территориальному признаку сети делят на **локальные, региональные и глобальные**.

Локальные	Региональные	Глобальные
Локальные сети охватывают ресурсы, расположенные недалеко друг от друга (чаще всего это соседние здания и прилегающая к ним территория – например, локальная сеть школы, вуза, компьютерного клуба и т. д.). Локальные сети предоставляют пользователям возможность не только быстрого обмена информацией, но и совместной работы на принтерах и других периферийных устройствах и даже одновременной обработки документов.	Региональные сети охватывают город, район, область, небольшую республику. Иногда выделяют корпоративные сети, где важно защитить информацию от несанкционированного доступа (например, сеть Министерства обороны, банковские сети и т. п.). Корпоративная сеть может объединять тысячи и десятки тысяч компьютеров, размещенных в различных странах и городах (в качестве примера можно привести сеть корпорации Microsoft).	Глобальные сети охватывают всю страну, несколько стран и целые континенты. На сегодняшний день в мире насчитывается более 200 глобальных сетей. Из них наиболее известной и самой популярной является сеть Интернет. В настоящее время на десятках миллионов компьютеров, подключенных к Интернету, хранится громадный объем информации (сотни миллионов файлов, документов и т. д.) и сотни миллионов людей пользуются информационными услугами глобальной сети.

При конструировании локальных сетей используют следующие виды кабелей:

– **витая пара**; скорость передачи информации до 100 Мбит/с на расстояние до 90 м;
– **коаксиальный кабель**; позволяет передавать информацию на расстояние до 2000 м со скоростью 2–44 Мбит/с;
– **волоконно-оптический кабель**; позволяет передавать информацию на расстояние до 10 000 м со скоростью до 100 Гбит/с.

Для работы в глобальной сети пользователю необходимо иметь соответствующее аппаратное и программное обеспечение.

Программное обеспечение можно разделить на два класса:

✓ **программы-серверы**, которые размещаются на узле сети, обслуживающем компьютер пользователя;

✓ **программы-клиенты**, размещенные на компьютере пользователя и пользующиеся услугами сервера.

Информационные сервисы сети Интернет

Электронная почта	Самый популярный сервис Интернета, является исторически первой информационной услугой компьютерных сетей и не требует обязательного наличия высокоскоростных и качественных линий связи. **Любой пользователь Интернета может получить свой «почтовый ящик» на одном из почтовых серверов Интернета (обычно на почтовом сервере провайдера), в котором будут храниться передаваемые и получаемые электронные письма.** Электронное письмо представляет собой обычный текст, дополненный некоторой служебной информацией. К письму может прилагаться сопутствующая информация в виде графических, звуковых или иных файлов. Чтобы отправить электронное письмо, отправитель должен подключиться к Интернету и передать на свой почтовый сервер сообщение. Почтовый сервер сразу же отправит это письмо через систему почтовых серверов Интернет на почтовый сервер получателя, и оно попадет в его почтовый ящик. Однако получатель получит письмо только после того, как соединится с Интернетом и «скачает» почту из своего почтового ящика на собственный локальный компьютер.
Электронные доски объявлений (BBS)	Первоначально этот вид сетевого общения возник как средство обмена программами и технической информацией между программистами-любителями. Постепенно отдельные BBS начали объединяться. Наиболее удачным вариантом такого объединения стала сеть **FidoNet**, обмен данными в которой производился по телефонной сети. Пользование этой сетью абсолютно бесплатно, т. к. не требует обращения к посредникам – поставщикам услуг (провайдерам).
Телеконференции	**Телеконференция – это обмен электронными сообщениями по определенной тематике.** Любое сообщение, отправленное в ту или иную тему конференции, попадает всем, кто участвует в работе этой темы. Участник готовит свои сообщения по тем или иным темам, а затем соединяется с сервером конференции. Компьютер передает написанные сообщения (если они, конечно, есть) и принимает все новое, что появилось по интересующим абонента темам. В Интернете существуют десятки тысяч конференций или групп новостей (news), каждая из которых посвящена обсуждению какой-либо проблемы: образованию, искусству, программированию, бизнесу и т. д. **Любой конференции выделяется свой почтовый ящик на серверах Интернета, поддерживающих работу этой телеконференции.** Принцип работы в телеконференциях мало чем отличается от принципа работы с электронной почтой. Пользователь имеет возможность посылать свои сообщения в любую телеконференцию и читать сообщения, посланные другими участниками.

Файловые архивы	Программное обеспечение, размещаемое на таких серверах, можно разделить на две большие группы: **свободно распространяемое программное обеспечение freeware и условно бесплатное программное обеспечение shareware**. Многие производители программного обеспечения и компьютерного оборудования заинтересованы в широком бесплатном распространении программного обеспечения. К таким программным средствам можно отнести новые недоработанные (бета) версии программных продуктов, драйверы к новым устройствам или улучшенные драйверы к уже существующим и т. д. В рекламных целях на файловых серверах фирмы часто размещают также условно бесплатное программное обеспечение (программы с ограниченным сроком действия или программы с ограниченными функциональными возможностями). Для работы с серверами файловых архивов можно использовать браузеры, однако удобнее пользоваться специальными программами, которые называются **менеджерами загрузки файлов**.
WWW	Это наиболее распространенная служба Интернета в настоящее время. Она получила настолько широкое распространение, что начинает вмещать в себя все остальные перечисленные выше службы. Так, используя программное обеспечение для WWW, можно получать доступ к файловым архивам, отправлять почту, участвовать в конференциях и т. д. **Основой Всемирной паутины является принцип гиперссылок.** В любом месте Web-страницы может быть поставлена ссылка на другую страницу, связанную по смыслу с данной. Причем страница, на которую делается ссылка, может находиться не только на данном компьютере, но и на любом другом, в том числе, расположенном на другом конце земного шара. Благодаря такой организации взаимных ссылок все материалы фактически объединяются в единое целое, образуя, образно говоря, всемирную информационную паутину. Для путешествия по ней требуется специальное программное обеспечение, которое называют **браузером**.
Поисковые сервера	Одной из важнейших задач работы с информацией в сети Интернет является ее поиск. Для решения этой задачи создаются специальные поисковые сервера, которые просматривают огромные объемы информации и составляют базы ссылок на размещенные в Интернете материалы. **Существует две разновидности поисковых серверов: поисковые каталоги и поисковые указатели.** Среди известных поисковых каталогов можно назвать российский **List.ru** и зарубежный **Yahoo!**. Крупнейшими отечественными поисковыми системами являются **Яндекс, Рамблер** и **Апорт**. В зарубежной части Сети наиболее известны **Alta Vista, Lycos**.

Компьютерные вирусы

Компьютерные вирусы – это программы, которые могут «размножаться» (создавать свои копии) и скрытно внедрять свои копии в файлы, загрузочные сектора дисков и документы. При этом копии могут сохранять способность дальнейшего распространения. Вирус может дописывать себя везде, где он имеет шанс выполниться.

По среде обитания вирусы можно разделить на **файловые, загрузочные, макровирусы и сетевые**.

Файловые	**Файловые вирусы внедряются в исполняемые файлы (программы) и активизируются при их запуске.** **После запуска зараженной программы вирус находится в оперативной памяти компьютера и может заражать другие файлы вплоть до момента выключения компьютера или перезагрузки операционной системы.** При этом могут быть заражены даже файлы данных (например, звуковые или графические). Поэтому не рекомендуется запускать на выполнение файлы, полученные из сомнительного источника и не проверенные предварительно антивирусными программами.
Загрузочные	**Загрузочные вирусы записывают себя в загрузочный сектор диска.** **При загрузке операционной системы с зараженного диска вирусы внедряются в оперативную память компьютера. В дальнейшем загрузочный вирус ведет себя как файловый.** Чтобы обезопасить себя от подобных вирусов, не загружайте операционную систему с гибких дисков и установите на BIOS вашего компьютера защиту от изменений загрузочного сектора.

Макровирусы	**Макровирусы заражают файлы документов Word, электронных таблиц Excel.** Макровирусы фактически являются **макрокомандами (макросами)**, которые встраиваются в документ. **После загрузки зараженного документа в соответствующее приложение макровирусы постоянно присутствуют в памяти компьютера и могут заражать другие документы.** Угроза заражения прекращается только после закрытия приложения. Профилактика заражения такими вирусами состоит в отказе от загрузки макросов, однако таким образом вы отключите и полезные макросы, содержащиеся в документе.
Сетевые	**Сетевые вирусы – это вирусы, распространяющиеся и заражающие компьютеры по компьютерной сети.** Заражение может произойти и, например, **при получении зараженных файлов с серверов файловых архивов.** Существуют и специфические вирусы, которые распространяются через электронную почту и WWW. К ним относятся, например, так называемые **Интернет-черви (Worm)**. **Эти вирусы распространяются во вложенных в почтовое сообщение файлах.** Такие вирусы, как правило, активизируются по определенным датам и уничтожают файлы на дисках зараженного компьютера.

Профилактика заражения компьютерным вирусом

Основные признаки появления в системе вируса
– замедление работы некоторых программ; – увеличение размеров файлов (особенно выполняемых); – появление не существовавших ранее «странных» файлов, особенно в каталоге Windows или корневом; – уменьшение объема доступной оперативной памяти; – внезапно возникающие разнообразные видео и звуковые эффекты; – заметное снижение скорости работы в Интернете (вирус или троянец могут передавать информацию по сети); – жалобы от друзей (или провайдера) о том, что к ним приходят непонятные письма; вирусы любят рассылать себя по почте; – исчезновение файлов и каталогов или искажение их содержимого; – невозможность загрузки операционной системы; – изменение размеров, даты и времени модификации файлов; – частые зависания и сбои в работе компьютера.

Общие рекомендации по профилактике заражения вирусом

– Проверяйте на наличие вирусов все поступающие извне данные, в том числе через гибкие и компакт-диски, а также по любым сетям.
– Периодически проверяйте все жесткие диски вашего компьютера на наличие вирусов.
– Старайтесь использовать лицензионные программные продукты.
– Не пускайте за свой компьютер друзей с неизвестно откуда взявшимися «игрушками».
– Всегда защищайте свои гибкие диски от записи при работе на других компьютерах, если на них не будет производиться запись информации.
– Не оставляйте в кармане дисковода для гибких магнитных дисков дискету при включении или перезагрузке компьютера, чтобы исключить заражение компьютера загрузочными вирусами.
– Регулярно обновляйте вирусную базу своих антивирусных программ.

Помните!

При борьбе с вирусами не стоит стирать все файлы вашего компьютера подряд. При этом можно удалить важные системные файлы, что приведет к невозможности работы на компьютере.
На этом построено действие «психологических» вирусов, рассчитанных именно на то, что пользователь своими руками разрушит систему.

Для защиты компьютеров от вирусов создаются специальные антивирусные программы. Они способны либо обнаружить вирус, либо обнаружить и обезвредить его.
К наиболее популярным антивирусным программам относятся российские программы **Dr Web, ADinf, AVP** и зарубежные **Norton Antivirus, Dr/Solomon.**

Основные способы защиты информации на локальном компьютере и в компьютерных сетях

Под **информационной безопасностью** понимается защищенность информации от случайных или преднамеренных воздействий естественного или искусственного характера, чреватых нанесением ущерба владельцам или пользователям информации.

Аспекты информационной безопасности

Доступность	Целостность	Конфиденциальность
возможность за разумное время получить требуемую информационную услугу	защищенность информации от разрушения и несанкционированного изменения	защита от несанкционированного доступа (прочтения)

Можно выделить следующие уровни защиты информации:

Предотвращение	доступ к информации и технологии предоставляется только для пользователей, получивших доступ от собственника информации
Обнаружение	обеспечивается раннее обнаружение преступлений и злоупотреблений, даже если механизмы защиты при этом были обойдены
Ограничение	уменьшается размер потерь, если преступление все-таки произошло, несмотря на меры по предотвращению и обнаружению
Восстановление	обеспечивается эффективное восстановление информации при наличии документированных и проверенных планов по восстановлению

Для обеспечения безопасности при работе в Интернете можно специальным образом настроить браузер. Например, Internet Explorer позволяет распределять по «зонам безопасности» любые файлы, которые вы можете открыть или получить (от файлов на вашем компьютере до файлов в Интернете).

Имеется 4 категории зон:

Местная зона безопасности	Зона надежных узлов	Зона ограниченных узлов	Зона Интернета
Содержит, как правило, любые адреса узлов, расположенных в данной организации. По умолчанию для такой зоны назначен средний уровень защиты.	К ней относятся узлы, которым вы доверяете и с которых можно загружать информацию и программы, не беспокоясь о возможном повреждении ваших собственных данных или компьютера. По умолчанию для этой зоны защита отсутствует.	К ней относятся узлы, которым вы не доверяете, считая небезопасным загружать с них информацию или запускать программы. По умолчанию для этой зоны назначен высокий уровень защиты.	Как правило, к этой зоне относится все, что не имеет отношения к вашему компьютеру, внутренней сети или иной зоне. По умолчанию для этой зоны назначен высокий уровень защиты.

Когда вы рискуете своей безопасностью, Internet Explorer может предупредить вас об этом. Например, если вы собираетесь послать какие-то свои данные на небезопасный узел, Internet Explorer известит вас, что этот узел не является безопасным.
Если же узел сообщает о своей безопасности, но предоставляет сомнительные гарантии, Internet Explorer предупредит вас о том, что данный узел может быть опасным.

Назначение служебных клавиш

Backspace	Удаляет символ слева от курсора.
Caps Lock	Включает/выключает режим прописных букв.
Ctrl, Alt	Самостоятельного действия не имеют, действуют только при совместном нажатии с буквенной или управляющей клавишей.
Ctrl + End	Перемещают курсор в конец документа.
Ctrl + Home	Перемещают курсор в начало документа.
Delete	Удаляет символ справа от курсора. Удаляет все, что выделено.
End	Перемещает курсор к концу строки.
Enter	Вводит набранную команду или текст. В текстовом редакторе переводит курсор на следующую строку.
Esc	Отменяет текущее действие. Обеспечивает выход из программы.
Home	Перемещает курсор в начало строки.
End	Перемещает курсор в конец строки.
Insert	Включает режим вставки или замены символа.
Num Lock	Переключает режимы работы дополнительной клавиатуры (режим ввода чисел, режим управления).
Page Down	Перемещает курсор вниз на виртуальную страницу.
Page Up	Перемещает курсор вверх на виртуальную страницу.
Schift	При одновременном нажатии с буквенной клавишей позволяет получать прописные буквы. При одновременном нажатии с цифровой клавишей – символы.
Tab	Устанавливает курсора на определенную позицию в строке.
Курсорные клавиши	Перемещают курсор на позицию влево, вправо, на строку вверх, вниз.
Ctrl + курсорные клавиши	Перемещают курсор на целое слово (стрелка влево или вправо), на целый абзац (стрелка вверх или вниз).