Исмаил Садыгов
Рамин Махмудзаде
Наида Исаева

ИНФОРМАТИКА

Учебник для 8-го класса общеобразовательной школы

Утверждено приказом Министерства образования Азербайджанской Республики №311 от 05.03.2008

8

BAKI NƏŞR
Bakı — 2010

Утверждено
Министерством образования
Азербайджанской Республики

Научный редактор: Расим Алигулиев, член-корреспондент НАНА, д.т.н., профессор
Рецензенты: Алекпер Алиев, д.т.н., профессор
Хаят Ахундова, педагог школы №164 г.Баку
Фатьма Гаджиева, педагог школы №83 г.Баку
Перевод: Наиля Баннаева

Информатика – учебник для 8-го класса общеобразовательной школы.
И.Дж. Садыгов, Р.А.Махмудзаде, Н.Р.Исаева. Баку, “Bakınəşr”, 2010, 128 стр.
ISBN-978-9952-430-06-8

© Министерство образования Азербайджанской Республики, 2010
© “Bakınəşr”. “TM group”. 2010

1

ИНФОРМАЦИЯ

1.1. ИНФОРМАЦИЯ. ДАННЫЕ. ЗНАНИЕ

Информация. *Информатика* изучает свойства информации. *Информация* – это сведения, помогающие нам ориентироваться в мире, в котором мы живем, поэтому она также необходима человеку, как воздух или вода. *Информация* – это знания, получаемые нами посредством телевидения, радио, газет, книг. Карты и произведения изобразительного искусства – это *информация*. Тысячи научных произведений, отражающих основы существующей цивилизации – это *информация*. И то, что написано в ваших учебниках – тоже *информация*.

Информатика

Информатика изучает технологии сбора, хранения, обработки, представления и передачи информации.

Данные. Человек извлекает информацию из данных. Данные можно создавать и уничтожать, тиражировать и перекомпоновывать. Сопоставляя данные, можно выделять среди них схожие и различные. Данные можно хранить на различных материальных носителях – бумаге, компакт-дисках, магнитных дисках, а значит, и обрабатывать при помощи различных технических устройств.

Человек преобразует данные в интересующую его информацию. Другими словами, данные можно сравнить с набором букв, написанным каким-либо алфавитом, а информацию – с рассказом либо сообщением, которые имеют определенный смысл. Человек извлекает информацию из фактов, оценивает ее, анализирует и на основе сделанных им аналитических выводов принимает решение.

Знание. Путем выполнения определенного действия или комплекса действий информация может преобразовываться в знание. Знание – это информация из общественного опыта, полученная экспериментальным путем и используемая для решения различных задач.

Данные. Информация. Знание

Значение этих терминов необходимо знать в точности, так как нюансы различия между ними очень тонки.

1. **Данные.** *Данные* – это некое сообщение, причем совершенно несущественно, имеет ли это сообщение смысл. Например, набор символов «19091985» или «VD51FGD» - это *данные*.
2. **Информация.** Информация – это данные, имеющие определенный смысл. Так, строка символов становится *информацией*, когда она что-то выражает. Скажем, строка «19091989» может быть как кодом какого-то товара, так и чьей-либо датой рождения (19 сентября 1989 года). Она становится информацией тогда, когда вы определяете, чего она касается.
3. **Знание.** Совершая над информацией определенные операции по соответствующим правилам, мы преобразуем ее в знание. Например, зная, что строка «19091989» означает чью-либо дату рождения (19 сентября 1989 года), можно сделать вывод, что данный человек достиг 18 лет и может участвовать в выборах. В данном случае, применив правило: «Граждане, достигшие 18 лет, имеют право участвовать в выборах», мы преобразуем информацию в *знание*.

1. Что изучает информатика?
2. В чем различие между информацией и данными?
3. Что такое знание?

1.2. ПРИЕМ ИНФОРМАЦИИ. ОСНОВНЫЕ СВОЙСТВА ИНФОРМАЦИИ.

Окружающий нас мир полон образов, звуков, запахов, и каждый из них – это источник информации. Всю эту информацию доносят до человека его органы чувств.

Человек обладает пятью органами чувств – *органами зрения, слуха, обоняния, вкуса и осязания.*

Получая разнообразную информацию, человек определяет для себя степень ее полезности.

Информация – это часть окружающего нас мира, другими словами, она – его объект. И поэтому, как любой объект, информация обладает определенными свойствами, отличающими ее от других объектов.

Объективность информации – это ее соответствие объективной действительности. Объективность крайне редко бывает абсолютной. Это объясняется тем, что данные по природе своей всегда объективны, однако методы анализа и обработки информации, как правило, субъективны. Например, “стоит жаркая погода” – это субъективная информация, а “температура воздуха равна 25°С” – объективная информация.

Полнота информации определяется количеством собранной информации, касающейся того или иного предмета или явления.

Достоверность информации – это ее неискаженность. Информация может быть искажена как преднамеренно, так и неумышленно. Она может быть ложной, или не полностью достоверной.

Адекватность информации отражает степень соответствия образа, создаваемого с помощью этой информации, реальному объекту, процессу, явлению и т.п. Создавая модель любого объекта, необходимо, чтобы эта модель была адекватна объекту, то есть как можно больше соответствовала объекту.

Актуальность информации отражает степень ее соответствия текущему моменту времени. Это особенно важно в сфере управления. Информация, актуальная вчера, сегодня может оказаться уже не актуальной.

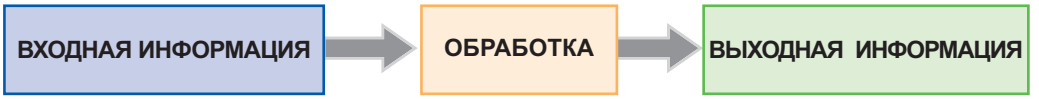
Понятность информации предполагает изложение информации языком, понятным для ее получателя.

1. Перечислите свойства информации.
2. Приведите пример, раскрывающий суть каждого из признаков информации.
3. Какими свойствами обладает информация, получаемая посредством телевидения и радио?
4. Приведите примеры неактуальной информации.

1.3. СБОР, ОБРАБОТКА, ПЕРЕДАЧА, ПОИСК И ХРАНЕНИЕ ИНФОРМАЦИИ.

С глубокой древности люди задумывались над способами упрощения и ускорения работы с информацией. Изобретение книгопечатания дало возможность быстрого распространения информации и упростило ее хранение. XIX столетие ознаменовалось грандиозными переменами в сфере передачи информации: сначала почтовые перевозки стали осуществляться при помощи новейших транспортных средств, а затем были изобретены телеграф и телефон. В наши же дни информацию можно мгновенно передать в любую точку мира. Сейчас информацию хранят не только на бумаге, но и на магнитной ленте, компакт-дисках, в памяти компьютера.

По мере приобретения человеком жизненного опыта, то есть с увеличением количества имеющейся у него информации, повышаются его способности к анализу и принятию решений.

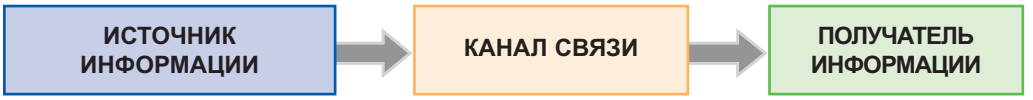


Входная информация. Выходная информация

Входная информация – это информация, получаемая человеком или устройством.
Выходная информация – это информация, полученная в результате обработки ее человеком или компьютером.

Человек давно научился обрабатывать собранную информацию при помощи технических устройств. Среди таких устройств особое место занимает компьютер, который способен обрабатывать огромное количество информации за малое время.

Без обмена информацией развитие человечества оказалось бы в принципе невозможным. Одним из главнейших факторов развития человечества является накопление информации, распространение полученных знаний между людьми и передача этих знаний из поколения в поколение.



Информация из информационного источника, проходя по каналу связи, доставляется до приемника информации. Каждая из трех частей вышеуказанной схемы характеризуется собственными свойствами (параметрами), от которых зависит качество передачи.

Но это упрощенная схема передачи информации. В действительности же данный процесс выглядит так:

Источник информации может представлять собой как живое существо, так и техническое устройство. Из источника информация попадает в кодирующее устройство, которое преобразует ее в удобную для передачи форму. С кодирующими устройствами мы сталкиваемся ежедневно: это микрофон телефона, лист бумаги и т.д. Пройдя по каналу связи, информация попадает в декодирующее устройство, которое преобразует кодированную информацию в форму, понятную для получателя. Человеческий глаз и человеческое ухо – сложнейшие декодирующие устройства.

Информацию необходимо хранить в таком виде, чтобы ее можно было при необходимости быстро и легко найти. До появления компьютеров этой цели служили всевозможные картотеки и каталоги.

По мере развития информационных технологий для хранения и поиска информации преимущественно стали использовать компьютеры.

При хранении информации применяются следующие способы ее защиты:

– безопасность здания, в котором хранится секретная информация;

– идентификация личности при доступе к секретной информации (пароль, пропуск, отпечаток пальца, голос, подпись);

– ограничение доступа людей к информации в зависимости от их полномочий;

– использование проверенных каналов и устройств связи;

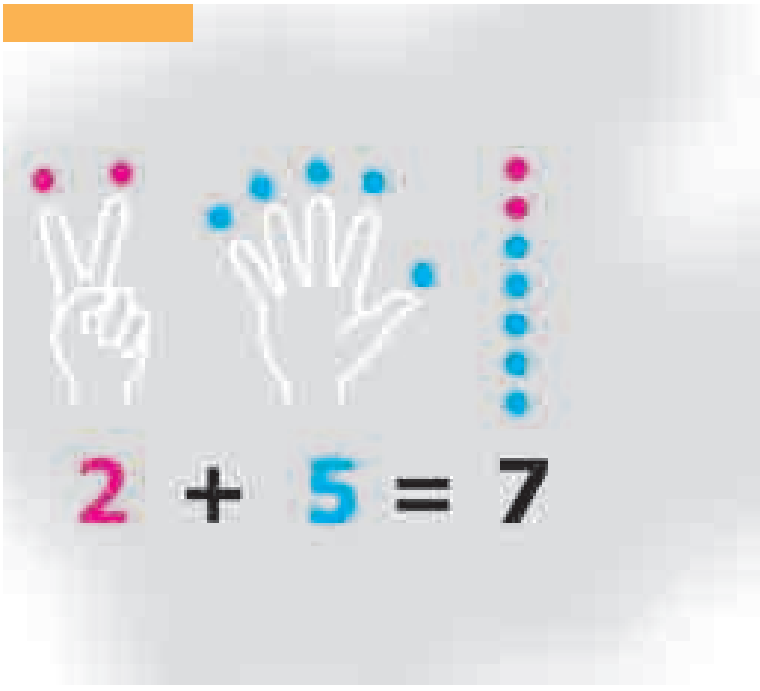
– криптографические преобразования информации при помощи шифров.

Криптография – это наука, занимающаяся методами кодирования и декодирования информации.

1. Что понимается под обработкой информации?
2. Как называется информация, полученная в результате обработки?
3. Из каких этапов состоит процесс передачи информации?
4. Какие методы применяются для защиты информации?

2

ИСТОРИЯ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ



2.1. ИСТОРИЯ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

Загибая пальцы при подсчете чего-либо, или же складывая числа при помощи пальцев, мы используем пальцы в качестве простейшего компьютера.

История развития компьютера берет свое начало именно со счета на пальцах и продолжается в наши дни. На протяжении многих веков средства для счета оставались довольно простыми и не имели существенных преимуществ перед счетом на пальцах.

Несмотря на то, что пальцы удобны для простых вычислений, они неприменимы для проведения вычислений с большими числами. Кроме пальцев, первобытный человек использовал для счета мелкие камеш-

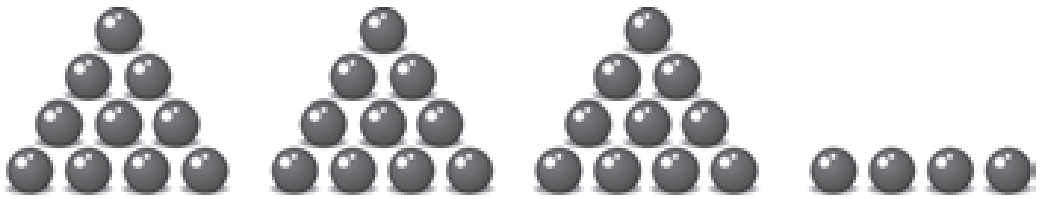
Латинское слово «digitus», означающее «палец», вовсе не случайно приобрело в английском языке новое значение: «digit» означает «цифра».

ки. К примеру, пересчитывая принадлежащее ему поголовье скота, он заменял каждое животное камешком, и в конце подсчетов мог по общему количеству камешков определить поголовье скота. Но если скота было уж очень много, это затрудняло подсчеты. В этом случае человек начинал складывать из камешков небольшие пирамиды.

Но сколько камешков должно было быть в каждой пирамиде, чтобы было удобно вести вычисления? Первобытный человек пришел к выводу, что поскольку у него на обеих руках 10 пальцев, то и в каждой пирамиде для удобства подсчета должно быть по 10 камешков.

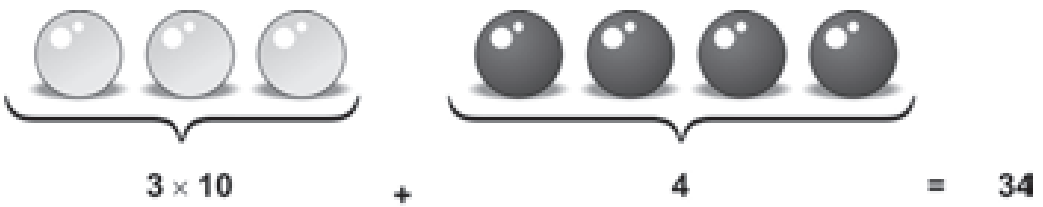
Предположим, что при таком способе вычислений у первобытного человека получилось три пирамиды, по 10 камешков в каждой, и еще 4 отдельных камешка. В этом случае он говорил:

“У меня есть три по десять да еще четыре животных”.



При таком способе подсчета даже для относительно небольших чисел (например, “34”) приходилось использовать много камешков, для работы же с большими числами подобный принцип применения камешков и вовсе не годился.

Следующим шагом являлось изображение каждого десятка голов скота одним камешком другого цвета. Первобытный человек вполне мог обозначать единицы черными камешками, а десятки – белыми. В этом случае для изображения числа “34” потребовалось бы всего 7 камешков: 3 белых и 4 черных.



Этот способ годился для работы и с гораздо большими числами – если обозначить сотни, скажем, красными камешками.

В этом случае для того, чтобы изобразить, например, число “154”, потребовалось бы всего 10 камешков.

Первые инструменты счета. У простых способов счета были свои недостатки – с теми же камешками, например, работать было неудобно. Очередной идеей стало использование разноцветных бусинок, нанизанных на проволоку. Этот инструмент счета стали применять в Китае примерно с 1200 года до нашей эры. Бусинки перемещались по проволокам, закрепленным на деревянной раме. Бусинки на первой проволоке означали единицы, на второй – десятки, на третьей – сотни, и т.д. Перемещая бусинки по проволоке, можно было складывать и вычитать числа. Это приспособление называлось *абак*.

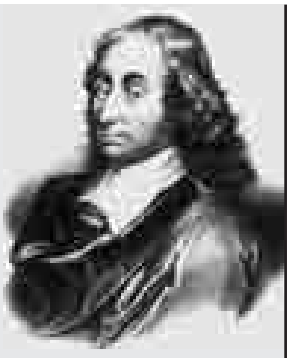
а) б)

Первые вычислительные средства

а) узелки на веревках служили для запоминания; б) абак.

Долгое время люди производили вычисления при помощи счетов. Счеты, использовавшиеся вплоть до середины XX века, ведут свое происхождение от абака. Однако сложение и вычитание больших чисел на счетах требует достаточно много времени.

Механические машины. В начале XVII века математика заняла лидирующее положение среди прочих наук. Физики и астрономы этого времени столкнулись с необходимостью проведения долгих и сложных математических расчетов. Возникла



Блез Паскаль
(1623-1663)

Изобрел первое счетное устройство. Видя, как отец – сборщик налогов – день-деньской ведет утомительные подсчеты, он решил изобрести счетное приспособление.
В 1962 году, в возрасте всего 19 лет, Паскаль создал и представил общественности первую механическую вычислительную машину – «Паскалину».

потребность в более совершенных инструментах, способных проводить подобные расчеты за короткое время.

Основным недостатком абака (счетов) было то, что перевод единиц из одного разряда в другой осуществлялся вручную. Механизировать этот процесс на уровне камешков (косточек) было невозможно, следовательно, необходимо было создать механизм, обладающий возможностью работы с 10-ю положениями (состояниями), соответствующими 10 цифрам. Эту идею впервые воплотил в жизнь гениальный французский математик **Блез Паскаль**, когда ему было всего 19 лет. После долгих поисков он изобрел довольно сложное для той эпохи устройство из 10 зубчатых колес. Зубцы на каждом были пронумерованы от “0” до “9”. Каждый поворот колеса на один зубец обозначал увеличение обозначаемого им разряда на единицу. При совершении первым из колес полного оборота (до цифры “9” включительно) специальный механизм автоматически передвигал второе колесо на один зубец. При полном повороте второго колеса автоматически передвигалось на один зубец третье колесо, и т.д. Говоря современным языком, Паскаль изобрел многоразрядный механический счетчик (принцип его и поныне используется в автомобильных спидометрах, электрических счетчиках и т.д.).

Паскалина

Паскаль более десяти лет совершенствовал свою машину. Он изготовил 50 ее экземпляров – из латуни, слоновой кости и других материалов. Из них до наших дней дошли только восемь. Однако “Паскалина” (так была названо эта машина) не получила широкого применения. Во-первых, она была довольно дорогой, а во-вторых, что гораздо важнее, она предусматривала сложение и вычитание только 6-10 разрядных чисел. Что же касается таких сложных операций, как умножение и деление, то они тогда вообще не были механизированы – этого события пришлось ждать еще полвека.

Арифмометр. Умножение многозначных чисел представляет собой, по сути, многократное сложение. Производя на бумаге умножение “в столбик”, мы именно так и поступаем. При вычислении по такому алгоритму на машине Паскаля один и тот же множи-

тель пришлось бы вводить несколько раз (к примеру, 6 раз). Периодически этот множитель надо было бы сдвигать на разряд влево, что, конечно же, довольно утомительно.

В 1673 году видный немецкий математик, философ, дипломат **Готфрид Лейбниц** предложил очень оригинальное решение этой проблемы. Он внес в устройство вычислительной машины два принципиальных усовершенствования.

Во-первых, для повторного введения одних и тех же чисел он применил ступенчатый валик (по ступеньке для каждого уровня) и поворотный рычаг. При повороте рычага на один оборот, в зависимости от положения промежуточной шестерни, вал сдвигался на заданную долю оборота.

Во-вторых, он разместил механизм ввода чисел на подвижной каретке, так что при каждом очередном этапе умножения каретка сдвигалась на разряд влево от счетного механизма. (В последующих конструкциях арифмометров оказалось удобнее установить механизм ввода на неподвижной части, а счетный механизм – наоборот, на каретке, поэтому она стала двигаться вправо).

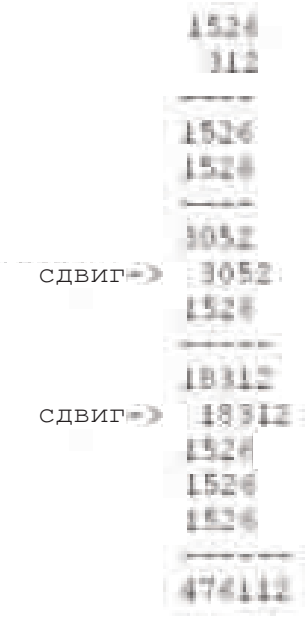
Таким образом, стало возможным, введя множитель единожды, многократно вводить его, путем сдвига, в счетное устройство и в итоге осуществлять операцию умножения. Аналогичным способом, вновь и вновь извлекая (вычитая) делитель из делимого, можно было осуществлять операцию деления. Для этого рычаг устройства нужно было вращать в противоположную сторону.

Аналитическая машина Бэббиджа. С точки зрения развития вычислительной техники XVIII век не был насыщен событиями, зато начало XIX века ознаменовалось многими крупными техническими изобретениями. В 1804 году французский механик **Жозеф-Мари Жаккар** (1752 - 1834), изобрел станок, который автоматически, без участия человека, ткал узорчатые ткани. Станок управлялся программным механизмом на перфокартах. Станок Жаккара произвел революцию в технике, ибо воплощал новый технологический принцип – принцип программного управления. Вскоре

Готфрид Лейбниц
(1646-1716)

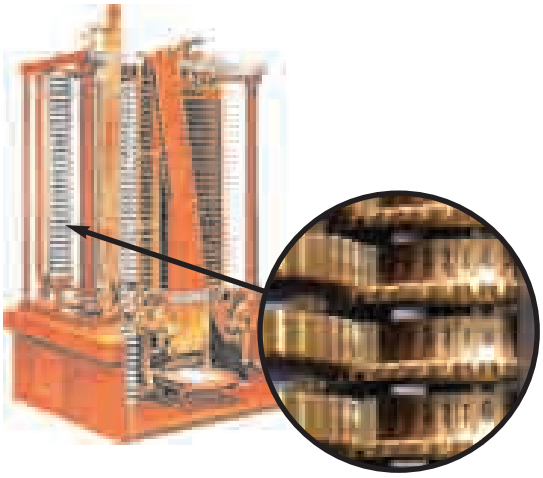
В 1672 году Лейбниц, будучи в Париже, познакомился с голландским математиком и астрономом Христианом Гюйгенсом. Наблюдая, сколько вычислений приходится делать астроному, Лейбниц пришел к выводу о необходимости изобретения механического устройства для облегчения этой работы.

Лейбниц писал: *«Это недостойно таких замечательных людей – подобно рабам, терять время на вычислительную работу, которую можно было бы доверить кому угодно при использовании машины».*



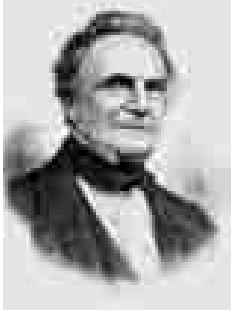
появились и другие программно-управляемые автоматы. Известный английский ученый и изобретатель **Чарльз Бэббидж** (1791-1871) впервые выдвинул идею применить данный принцип к вычислительному устройству.

Круг научных интересов Бэббиджа был очень широк: он опубликовал множество научных статей по математике, физике, астрономии, геологии, экономике, языкознанию. Он был автором не только научных работ, но и ряда изобретений. Более 50 лет своей жизни Бэббидж посвятил конструированию вычислительных машин.



Внешний вид разностной машины

В 1822 году он приступил к созданию устройства, именуемого *Разностной машиной (Difference engine)* и предназначенного для автоматического составления сложных математических таблиц. После 12 лет работы над этим проектом Бэббидж отказался от него и начал работу над *Аналитической машиной (Analytical engine)*. Новая машина должна была быть более сложной и решать разнообразные задачи. Введение команд и данных предполагалось осуществлять с помощью перфокарт. Воплощению в жизнь этой своей мечты Бэббидж посвятил всю свою оставшуюся жизнь, однако так и не достиг успеха, потому что его идеи опередили свое время примерно на сто лет. Технологии его эпохи не позволяли создать подобную машину.



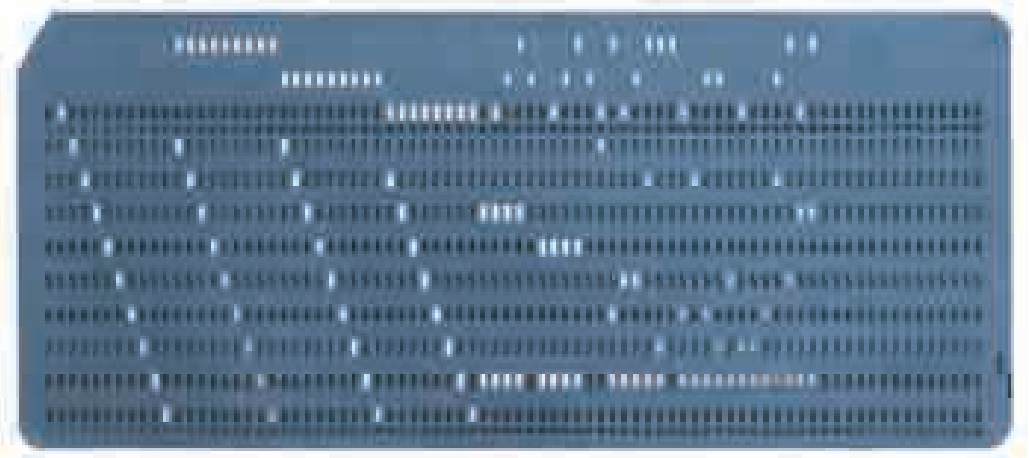
Чарльз Бэббидж
(1791-1871)

Ада Лавлейс
(1815–1852)

Современница Бэббиджа Ада Лавлейс была одним из тех людей, кто оценил значение аналитической машины. Ее зачастую называют *первым программистом* в мире. Некоторые приемы ее теоретических разработок по управлению последовательностью вычислений и поныне используются в программировании. Например, она дала описание команд, обеспечивающих повторяемость последовательностей определенных действий. В современном программировании подобная конструкция называется циклом.

В честь Ады Лавлейс один из языков программирования был назван *«Ада»*.

Перфокарта. Долгое время перфорированные (пробитые) карты, или перфокарты, были основным средством ввода данных в компьютер. Внешний вид перфокарт практически не изменился с того момента, когда в 1890 году Герман Холлерит (1860 - 1929) применил их в табуляторе для анализа результатов переписи населения США. Последние перфокарты весьма напоминали те, что выпустила в оборот компания IBM в 1928 году. Верхний левый угол их был слегка срезан, чтобы не возникало путаницы при складывании в колоду.



Перфокарта с 80 колонками

Цифры, буквы и прочие символы представляются в соответствии с кодом, предложенным Г.Холлеритом, каждая в одной из 80 колонок. Цифры кодируются одной пробивкой в нижней строчке, буквы – двумя пробивками (одна – на так называемой « цифровой строчке», вторая – на так называемой «зонной строчке»). Прочие символы кодируются двумя и более пробивками в других строчках.

1. Что представляли из себя первые вычислительные средства?
2. Кто изобрел первую механическую вычислительную машину?
3. Какие математические операции можно было осуществлять при помощи вычислительной машины Лейбница?
4. Почему Бэббидж так и не смог закончить работу над созданием своей Аналитической машины?
5. В честь кого из людей, перечисленных в данном параграфе, был назван один из языков программирования?

Первый компьютер – ENIAC

2.2. ПОКОЛЕНИЯ КОМПЬЮТЕРОВ

Появление электронно-вычислительных машин стало возможным в результате двух важных событий:

1. Были придуманы способы представления информации – чисел и слов – в двоичной форме. Каждое число или слово изображалось в виде последовательности единиц и нулей.
2. Были изобретены электронные устройства, позволяющие запоминать числа и слова в двоичной форме.

Последующее развитие компьютеров было связано, в первую очередь, именно с прогрессом в электронике. В результате этого развития появился термин “поколение компьютеров”. Каждое новое поколение отличается от предыдущего изменением элементной базы и расширением (в той или иной степени) класса решаемых задач. На практике же ощутимым различием в элементных базах

(радиолампы, транзисторы, интегральные схемы) отличаются лишь первые три поколения. В дальнейшем развитие элементных баз свелось к уменьшению размеров интегральных схем и повышению степени интеграции.

Первым компьютером стал созданный в 1946 году в США **ENIAC** (**E**lectronical **N**umerical **I**ntegrator **a**nd **C**alculator – Электронно-цифровой интегратор и вычислитель). Схемы этого компьютера были созданы на основе электронных ламп. ENIAC был создан по заказу армии США в исследовательской лаборатории баллистики для расчета артиллерийских таблиц, но при перепрограммировании был способен решать самые разные задачи и поэтому стал первым широкомасштабным электронно-вычислительным компьютером. Его архитектура была разработана в 1943 году учеными Пенсильванского университета **Джоном Преспером Эккертом** и **Джоном Уильямом Мочли**.

Основой компонентной базы ENIAC были вакуумные лампы. В общей сложности в нем было 17468 ламп, 7200 кремниевых диодов, 1500 реле, 70000 резисторов и 10000 конденсаторов. ENIAC потреблял 150 кВт электроэнергии. Расчеты производились в десятичной системе: за секунду он мог производить 300 операций умножения или 5000 операций сложения. Вес ENIAC составлял 30 тонн. Занимала эта машина около 167 кв.м.

Первое поколение компьютеров. Компьютеры первого поколения называют *электронно-вычислительными машинами (ЭВМ)*. К ним относятся ламповые ЭВМ, созданные в бывшем СССР после появления первого представителя этого поколения – ENIAC. В 1950 году была сдана в эксплуатацию **МЭСМ** *(Малая электронно-вычислительная машина)*.

ЭВМ этого поколения отличались крупными габаритами, высоким потреблением электроэнергии, сложностью в управлении и ненадежностью. Они использовались до начала 60-х гг. XX века.

Второе поколение компьютеров. В 1947 году группа американских физиков под руководством Уильяма Шокли разработала полупроводниковый элемент, названный транзистором. Транзистор многократно превосходил ламповый триод по надежности, а электроэнергии потреблял во много раз меньше него.

Транзисторы легли в основу элементной базы компьютеров второго поколения. Один транзистор заменял собой 40 электронных ламп, он работал быстрее, был более дешевым и надежным по сравнению с лампами. Производство компьютеров второго поколения продолжалось с 1957-го года до конца 60-х гг.

Третье поколение компьютеров. В 1959 году **Джек Килби** (1923–2005), сотрудник компании “Texas Instruments” (США), предложил объединить несколько транзисторов, конденсаторов и резисторов на полупроводниковой микросхеме. Так была создана *интегральная микросхема*.

Начиная с 1964 года интегральные микросхемы стали основой элементной базы компьютеров третьего поколения. Это помогло снизить цену на компьютеры, и их, наконец, смогли приобретать высшие и специальные учебные заведения.

Четвертое поколение компьютеров. С развитием технологии производства микросхем количество элементов, размещаемых на них, неуклонно росло. В 1970 году на одной микросхеме уже размещалось более 100 транзисторов. С того времени интегральные схемы стали подразделяться на малые, средние, большие и т.д. В наши дни на микросхемах содержатся несколько миллионов элементов.

С появлением в 70-х годах XX столетия компьютеров на больших интегральных схемах, стало возможным говорить об их четвертом поколении. В результате высокого уровня интеграции размеры компьютеров уменьшились настолько, что машины четвертого поколения часто именуют *микрокомпьютерами*. Цены на них к тому времени снизились настолько, что компьютеры смогли приобретать некоторые отделы и лаборатории учреждений.

Переход на сверхбольшие интегральные схемы позволил настолько уменьшить габариты компьютеров, что ими стало возможным оснащать отдельные рабочие места. Именно тогда возник новый термин – *“персональный компьютер”*.

В 1974 году был создан компьютер “Альтаир 8800”, а в 1977 году появился компьютер “Apple II” компании “Apple Computer” – прообраз современного муль-

тимедийного компьютера. С этого и началась эра персональных компьютеров.

С тех пор персональные компьютеры сами изменились настолько, что можно выделить несколько поколений их собственных моделей. С этой точки зрения мы сейчас находимся на этапе перехода от седьмого к восьмому поколению персональных компьютеров.

Пятое поколение компьютеров. В 1980-е годы в Японии была принята правительственная программа по развитию компьютерной отрасли и искусственного интеллекта. Было объявлено, что компьютеры пятого поколения – это компьютеры будущего. Они будут обладать искусственным интеллектом. Эти компьютеры будут понимать естественный язык людей, воспринимать и обрабатывать информацию из рукописных и печатных текстов, определять пользователя по голосу.

В XIX веке английский педагог **Уильям Шенкс** потратил 28 лет на то, чтобы вычислить 707 цифр после запятой, используемого в геометрии числа π («пи»). Современная программа для компьютера способна вычислить это значение за несколько секунд. При первом же рассчете по программе уже на 528-й десятичной цифре была обнаружена ошибка в вычислениях Шенкса.

Компьютер пятого поколения – PIM/m-1

Компьютеры пятого поколения будут обрабатывать не данные, а знания. За 10 лет на разработки было истрачено огромное количество денег, но программа завершилась, не достигнув цели. Появление Интернета сделало многие идеи проекта устаревшими.

Несмотря на провал заявленного проекта компьютера пятого поколения, в различных странах мира (Японии, США и др.) ведутся исследования по отдельным направлениям этого проекта. На данный момент уже созданы интеллектуальные системы, идентифицирующие человеческий голос.

В наши дни созданы суперкомпьютеры. Вычисления в них ведутся параллельно, поэтому скорость их работы намного превышает скорость обычных компьютеров. Например, в суперкомпьютере “Columbia” (США) параллельно работают 10240 процессоров. Если для проведения вычислений одного эксперимента этому компьютеру потребуется 80 часов, то обычному компьютеру – 18 лет.

Суперкомпьютеры применяются во многих сферах:

– для раскрытия генетического кода человеческого организма;
– для изучения атмосферных явлений и изменений климата;
– для прогнозирования стихийных бедствий (землетрясений, наводнений);
– в военных целях;
– в атомной и молекулярной физике;
– в космических исследованиях и т.д.

1. Чем различались первое и второе поколение компьютеров?
2. В каком поколении компьютеров стали впервые применяться интегральные схемы?
3. Благодаря чему уменьшились габариты персональных компьютеров?
4. Чем отличается суперкомпьютер от обычного компьютера?

3

АРИФМЕТИЧЕСКИЕ ОСНОВЫ КОМПЬЮТЕРА

3.1. СИСТЕМЫ СЧИСЛЕНИЯ

“Всё сущее есть число”. Таков был главнейший лозунг пифагорейцев, отражающий необычную роль чисел в повседневной жизни человека.

Уже в глубокой древности люди научились считать и записывать числа. Только записывали они их совершенно по-другому – не так, как мы. Несмотря на это, число обозначалось одним или несколькими символами, которые именовали цифрами.

Цифра

Цифрами называются символы, участвующие в записи чисел и соответствующие определенному алфавиту.

А что же такое число? Первоначально число было связано с исчисляемыми предметами, но с появлением письменности оно отделилось от предметов, и возникло понятие натуральных чисел. Измерения, при которых измеряемая величина не укладывалась в целое число единиц измерения (эталона), породили необходимость возникновения дробей. В дальнейшем понятие числа в математике получило еще большее развитие, и теперь оно считается одним из основных понятий не только математики, но и информатики.

Число

Число – это показатель той или иной величины.

Число записывается цифрами в соответствии с определенными правилами. На разных этапах развития человечества эти правила были разными у разных народов. В наши дни эти правила принято называть *системами счисления*.

Системы счисления

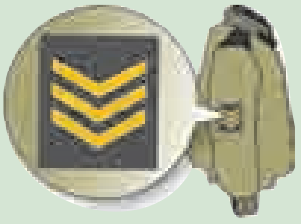
Системы счисления – это способы (правила) записи чисел при помощи цифр.

Системы счисления бывают позиционные и непозиционные. Непозиционные системы счисления зародились задолго до позиционных и только на определенном этапе исторического развития появились позиционные системы счисления.

Непозиционные системы счисления. Необходимость в счете возникла естественным образом еще в каменном веке – для подсчета собранных плодов, пойманной рыбы и количества добытой дичи. Археологические находки, относящиеся к эпохе палеолита (камни, кости животных), показывают, что люди того времени старались группировать точки, черточки, зарубки *по три*, *по четыре*, *по пять* или *по семь*. Такая система записи чисел именуется *единичной системой счисления*, так как в ней запись необходимого числа ведется при помощи повторения одного и того же знака, символизирующего единицу. *Группировка* и *вспомогательные знаки* использовались лишь для облегчения восприятия больших чисел.

Единичная система счисления, которую использовали еще первобытные люди, ведя счет на стенах пещер при помощи зарубок, сделанных костями животных и ветками, не забыта и поныне.

На рукавах мундиров курсантов военных школ имеются нашивки в виде линий – шевроны. Их количество обозначает курс, на котором учится данный курсант.



На фюзеляжах асов военно-воздушных сил количеством звездочек обозначалось количество сбитых вражеских самолетов.

Подсчет предметов по одному удобно вести, только если их не слишком много. Вести подсчет больших количеств предметов по таким правилам, то есть методом повторения, довольно утомительно, поэтому возникла идея *объединения единиц в группы*. Появился счет *пятерками*, *десятками*, *двадцатками* (по числу пальцев на руках и ногах). Затем стали приобретать свои, отдельные, названия такие числа, как десять раз по десять (сотня), десять раз по сто (тысяча). Для удобства записи основные числа стали обозначаться специальными символами. Так, если во время счета получалось две сотни, семь десятков и четыре предмета, то символ сотни повторно записывали два раза, символ десятка – семь, а символ единицы – четыре раза. Для обозначения единиц, десятков и сотен использовались отличающиеся друг от друга символы.

При такой записи чисел символы можно было записывать в любой последовательности – значение числа от этого не менялось. Оттого, что при такой записи позиция символа не имеет значения, подобные

системы именуются непозиционными. Системы счисления древних египтян, греков и римлян были непозиционными.

В непозиционных системах счисления количественный эквивалент ("вес") цифры не зависит от ее расположения в записи числа. Рассмотрим записанное в римской системе счисления число III. В десятичной системе счисления это 3. В записи числа III используются три одинаковых "цифры" (I), равных по достоинству. Другими словами, при записи данного числа, вне зависимости от положения, "вес" цифр одинаков и равен единице.

Римская система счисления. В римской системе счисления в качестве цифр используются латинские буквы (их называют *римскими цифрами*).

I	V	X	L	C	D	M
1	5	10	50	100	500	1000

В римской системе число CCXXXII состоит из двух сотен, трех десятков и двух единиц и равно 232.

Состав числа CCXXXII						
100	100	10	10	10	1	1
232						

Если при написании числа римскими цифрами цифры выстраиваются слева направо в порядке убывания, то их значения складываются.

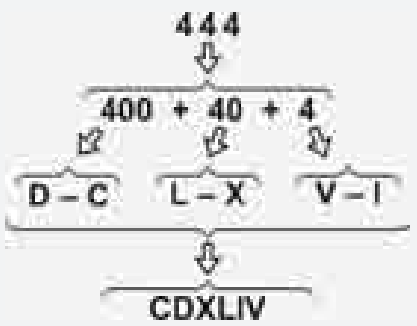
Если же меньшая по значению цифра стоит слева от большей, то их значения вычитаются.

Если слева стоит большая цифра, а справа – меньшая, то их значения складываются.	VI = 5 + 1 = 6
Если слева стоит меньшая цифра, а справа – большая, то их значения вычитаются.	IV = 5 – 1 = 4

MCMXCVIII = 1000+(-100+1000)+(-10+100)+5+1+1+1=1998

MMVIII= 1000 + 1000 + 5 + 1 + 1 + 1 = 2008

Теперь рассмотрим способ записи римскими цифрами числа 444.



Алфавитные системы. Наиболее совершенными из непозиционных систем являются *алфавитные системы*. Такие системы счисления были у финикийцев, греков, арабов, славян и других народов. В алфавитных системах числа от 1 до 9, целые десятки (от 10 до 90) и целые сотни (от 100 до 900) обозначались определенными буквами алфавита.

Система счета абджад

В арабском алфавите каждой букве присвоено числовое значение. То есть, в этом алфавите каждая буква может быть выражена числом, и наоборот, каждое число – буквой. Выходит, что и у каждого слова имеется, помимо буквенного, еще и числовое значение. Используя это, можно кодировать слова и производить различные расчеты. Такой метод кодирования и вычисления называется системой счета абджад.

ا	ب	ج	د	ه	و	ز	ح	ط	ى
1	2	3	4	5	6	7	8	9	10
ك	ل	م	ن	س	ع	ف	ص	ق	
20	30	40	50	60	70	80	90	100	
ر	ش	ت	ث	خ	ذ	ض	ظ	غ	
200	300	400	500	600	700	800	900	1000	

Древние арабы, тюрки, фарсы записывали даты всех событий при помощи букв, которые, как уже упоминалось, имели и числовое значение. Таким образом, слова не только отображали то или иное событие, но и несли в себе информацию о его дате. Например, Мухаммед Физули в поэме “Лейли и Меджнун” зашифровал дату завершения труда над этой поэмой в выражении: “В единое слившиеся двое влюбленных” — “**Bir** olmaq ilə **iki aşiq**”.

Сначала подсчитаем числовое значение слова “aşiq” (“влюбленный”) по правилу абджад:

عاشق = (70) + (1) + (300) + (100) = 471.

Полученное число надо умножить на два, так как в тексте речь идет о двух влюбленных. К результату прибавим единицу («В единое…») и получим **943** год по хиджре:

$$(471 \cdot 2 + 1 = 943).$$

В переводе на нынешнее летоисчисление это **1537** год.

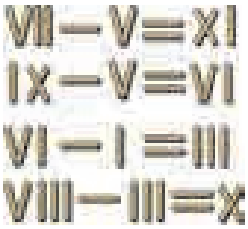
1. Что такое система счисления?
2. На какие группы подразделяются системы счисления?
3. Почему римская система счисления является непозиционной?
4. Что такое алфавитные системы?
5. Какие числа записаны здесь римскими цифрами?

   CXXVII

   MCMXVIII

   MCMXCI

6. Восстановите равенство, передвинув всего одну палочку.



3.2. ПОЗИЦИОННЫЕ СИСТЕМЫ СЧИСЛЕНИЯ

Системы счисления, основанные на принципе позиционности, возникли, независимо друг от друга, в древнем Вавилоне (Междуречье), у народности майя и, наконец, в Индии. Это доказывает, что возникновение принципа *позиционности* не было случайностью.

Ученые Вавилона использовали шестидесятеричную систему счисления, однако на практике чаще применялся ее сложный гибрид с десятичной системой. Индийские математики, многое позаимствовавшие у вавилонских ученых, разработали десятичную систему счисления. В VI веке они создали способ записи чисел при помощи всего 9 цифр. Вместо нуля сначала использовали пробел, затем – точку или маленький кружок. В IX веке появился специальный знак для нуля. Долгое время понятие нуля выглядело неясным и абстрактным (зачем, мол, нужен знак для обозначения чего-то несуществующего?), однако со временем новый метод записи чисел стал всем понятен и привычен.

В позиционной системе счисления количественный эквивалент (“вес”) цифры зависит от ее положения в записи числа. Рассмотрим число 222. В его записи трижды повторяется число “2”. Однако “вклад” в величину числа у каждой из этих цифр различный. Первая двойка обозначает сотни, вторая – десятки, третья – единицы. Если сравнить “вес” перечисленных цифр, то окажется, что первая из них в 10 раз “больше” второй и в 100 раз “больше” третьей. Подобный принцип отсутствует в непозиционных системах.

Первоначально в Древнем Вавилоне использовалась непозиционная система счисления, однако затем ее сменила позиционная. Но в этой системе при перемене места (позиции) цифры ее значение возрастало не в 10 раз, как в системе, ныне используемой нами, а в 60 раз. То есть, вавилоняне использовали шестидесятеричную систему счисления. Долгое время в вавилонской счетной системе отсутствовал нуль, то есть не было символа пропущенного разряда. Но с началом составления обширных математических и астрономических таблиц возникла необходимость в подобном символе. Некоторые следы вавилонской системы счисления можно наблюдать и в нашей современной жизни.
Например:

1 час = 60 минут

1 минута = 60 секунд

Позиционные системы счисления

В *позиционных системах счисления* величина, обозначаемая цифрой, зависит от позиции, занимаемой данной цифрой в записи числа.

У позиционных систем счисления есть два огромных преимущества:

1. Простота выполнения вычислений.
2. Ограничение количества символов, необходимых для записи чисел.

Позиционных систем счисления много, но самой известной из них является система счисления с основанием десять. В этой системе счисления для записи чисел применяется десять различных знаков:

0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

Основание системы счисления

Количество цифр, используемых позиционной системой счисления, определяет основание системы счисления.

В системах счисления, основания которых меньше 10, используются имеющиеся цифры, меньшие основания этих систем. Например, в восьмеричной системе счисления применяются только цифры 0, 1, 2, 3, 4, 5, 6, 7.

В системах счисления, основания которых больше 10, недостающие цифры обозначаются прописными буквами латинского алфавита. Например, в шестнадцатеричной системе используются следующие цифры:

Название системы счисления определяется числом, которое составляет его основание. Так, система счисления с основанием 10 называется десятичной, система счисления с основанием 8 – восьмеричной.

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, Б, Ъ, Д, E, Ф.

Здесь буквы A, B, C, D, E, F (цифры шестнадцатеричной системы) соответствуют числам 10, 11, 12, 13, 14, 15, 16 десятичной системы счисления. В таблице 3.1. указано, как следует считать числа в восьмеричной системе:

Таблица 3.1.

Десятичная	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Восьмеричная	1	2	3	4	5	6	7	10	11	12	13	14	15	16	17	20

При счете в восьмеричной системе (так же, как и в других системах счисления, отличных от десятичной) рекомендуется говорить не “десять”, “одиннадцать” и т.д., а “один ноль”, “один один” и т.д. Это указывает на то, что речь идет не о десятичной системе. В таблице 3.1. отражена, например, правильность таких равенств:

8 в десятичной = **10** в восьмеричной
13 в десятичной = **15** в восьмеричной
16 в десятичной = **20** в восьмеричной

Для проведения расчетов в восьмеричной системе можно воспользоваться таблицей 3.2.

Таблица 3.2.

Десятичная	8	16	24	32	40	48	56	64	72	128	512
Восьмеричная	10	20	30	40	50	60	70	100	110	200	1000

При работе с различными системами счисления основание системы, к которой принадлежит то или иное данное число, обозначают нижним индексом, например, 26_{10}, 32_{8}.

Позиционная система счисления

В компьютерной литературе и в некоторых языках программирования встречаются числа, записанные в четырех различных системах счисления – двоичной, восьмеричной, десятичной и шестнадцатеричной. Приняты определенные соглашения по поводу их письменной формы.

1. Десятичные числа записываются обычным способом, например:
   12345; 54321; 222.22
2. Перед восьмеричными числами указывается ведущий нуль, например:
   012345; 054321; 0222.22
3. Перед шестнадцатеричными числами указывается ведущий префикс “0x” или “0X”, например:
   0x12345; 0x54321; 0x222.22
4. Если по контексту и так ясно, что речь идет о двоичном числе, то в его записи не применяется никаких специальных знаков, например:
   0111 0010; 0011 0001

   В противном случае после числа пишется “BIN”:
   111 BIN; 1111 1111 BIN

Развернутая форма записи числа. Число $A=\overline{a_n a_{n-1}\ldots a_1 a_0}$ ($a_n, a_{n-1}, \ldots a_1, a_0$ – цифры числа), записанное в системе счисления с основанием p, можно представить в виде многочлена:

$$A = a_н п^н + a_{н-1} п^{н-1} + a_{н-2} п^{н-2} + \ldots + a_1 п^1 + a_0 п^0$$

Представление числа в виде многочлена называется развернутой формой записи числа. Например, десятичное число 12345 можно записать так:

$$12345 = 1\cdot 10000 + 2\cdot 1000 + 3\cdot 100 + 4\cdot 10 + 5 = 1\cdot 10^4 + 2\cdot 10^3 + 3\cdot 10^2 + 4\cdot 10^1 + 5\cdot 10^0.$$

Если число не является целым, то первый разряд после запятой соответствует числителю дроби, знаменатель которой равен основанию данной системы счисления. Второй разряд после запятой соответствует числителю дроби, знаменатель которой равен квадрату основания данной системы счисления, третий разряд – числителю дроби, знаменатель которой равен кубу основания данной системы счисления и т.д. В таком случае число A, записанное в виде

$$\overline{a_n a_{n-1} a_{n-2}\ldots a_1 a_0, a_{-1} a_{-2}\ldots a_{-m}}$$

можно представить в виде многочлена:

$$A = a_n p^n + a_{n-1} p^{n-1} + a_{n-2} p^{n-2} + \ldots + a_1 p^1 + a_0 p^0 + a_{-1} p^{-1} + a_{-2} p^{-2} + \ldots + a_{-m} p^{-m}$$

Например:

$$123{,}45 = 1\cdot 10^2 + 2\cdot 10^1 + 3\cdot 10^0 + 4\cdot 10^{-1} + 5\cdot 10^{-2}$$

1. Какие системы счисления называются позиционными?
2. К какой системе счисления можно отнести число 12350?
3. Используя таблицы 3.1 и 3.2, напишите числа в восьмеричной системе, соответствующие приведенным числам в десятичной системе:

$$17_{10},\ 20_{10},\ 25_{10},\ 30_{10},\ 41_{10},\ 58_{10},\ 65_{10}.$$

4. Сравните числа:
   а) 5_{10} и 5_8
   б) 1111_2 и 1111_8
5. Запишите числа 241, 242, 243, 250, 251 в системе с основанием 240.

3.3. ПЕРЕВОД ЧИСЕЛ ИЗ ОДНОЙ СИСТЕМЫ СЧИСЛЕНИЯ В ДРУГУЮ

Как перевести число, записанное в одной системе счисления (например, десятичной), в другую систему счисления (например, семеричную)?

Вам уже, наверное, известно, что записать некоторое число А в семеричной системе, значит, представить его в виде следующей суммы:

$$A = a_n \cdot 7^n + a_{n-1} \cdot 7^{n-1} + \ldots + a_1 \cdot 7^1 + a_0$$

Значит, чтобы записать некое число А в семеричной системе, нужно найти коэффициенты a_0, a_1, .., a_n, имеющие одно из значений от 0 до 6. Разделим число А на семь. Остаток составит, естественно, a_0–а потому что, как видно из вышеприведенной развернутой формы записи числа А, все его слагаемые, кроме последнего, делятся нацело на семь. Результат деления числа А на семь снова разделим на семь. На сей раз остаток составит a_1. Если продолжить этот процесс и далее, то мы найдем все цифры числа А в семеричной системе.

Перевод целых чисел из десятичной системы счисления в другую систему счисления. Переведем в семеричную систему счисления число 3287_{10}. Разделив его на семь, получим 469, и 4 в остатке. Значит, в записи числа 3287 в семеричной системе последней цифрой будет 4. Чтобы получить следующую цифру, разделим полученное в результате предыдущего деления число 469 на семь. Частное составит 67, остаток – 0. Значит, вторая с конца цифра записи числа 3287 в семеричной системе – 0. Затем делим 67 на семь и получаем 9, и 4 в остатке. Этот остаток (4) и будет третьей с конца цифрой записи 3287 в семеричной системе. Наконец, разделив 9 на 7 (последний раз), получаем 1, а остаток составит 2. Эта

цифра – 2 будет четвертой с конца в записи, а частное, то есть 1, будет пятой с конца, или первой с начала (так как дальше это число нацело на семь не делится). Получаем:

$$3287_{10} = 12404_{7}$$

Правая часть этого равенства – сокращенная форма выражения:

$$1 \cdot 7^4 + 2 \cdot 7^3 + 4 \cdot 7^2 + 0 \cdot 7 + 4$$

Точно так же, как $3 \cdot 10^3 + 2 \cdot 10^2 + 8 \cdot 10 + 7$ – запись числа 3287_{10}.

Вышеприведенные вычисления для перевода числа 3287_{10} в семеричную систему удобнее представить в следующем виде:

```
3287 | 7
 4   469 | 7
 0    67 | 7
 4     9 | 7
       2   1
```

или

```
3287 : 7 =  469  4
 469 : 7 =   67  0
  67 : 7 =    9  4
   9 : 7 =    1  2  ⇒ 12404₇
```

Разумеется, подобный способ можно применять для перевода любого числа не только в семеричную систему, но и в любую другую систему счисления.

Правила перевода любого числа A из десятичной системы счисления в систему счисления с основанием p таковы:

– разделить число A на p: полученный остаток будет цифрой первого разряда в искомой записи (последней цифрой числа);
– полученное частное вновь разделить на p: полученный остаток будет цифрой второго разряда, и т.д.;
– процесс этот нужно продолжать до тех пор, пока частное не окажется меньше основания системы счисления (то есть меньше p) – последнее полученное частное будет цифрой высшего разряда.

Вот еще один пример: перевести число 100 в двоичную систему. Применив указанные правила, получим:

```
100|2
 0 50|2
   0 25|2
     1 12|2
       0 6 |2
         0  3 |2
            1  1
```

Итак, $100_{10} = 1100100_2$

При работе на компьютере иногда приходится переводить числа из десятеричной системы в двоичную. О двоичной системе счисления вы более подробно узнаете на следующем уроке.

В рассмотренных выше примерах исходной системой являлась десятичная. Но по тем же правилам можно переводить числа из какой угодно системы в какую угодно – для этого нужно выполнить ту же последовательность действий, что и в приведенных примерах. Проводить эти операции нужно не в десятичной системе счисления, а в той системе, в которой записано данное число.

Заметим, что производить деление в системе счисления, отличной от десятичной, довольно затруднительно.

Перевод дробных чисел из десятичной системы счисления в другую. При переводе дробных чисел из одной системы счисления в другую сначала переводится целая часть числа, затем – дробная. Целая часть переводится по указанным выше правилам.

А вот правила перевода дробной части:

- дробная часть записанного в десятичной системе числа умножается на основание той системы, в какую переводится число;
- целая часть полученного произведения (она может равняться и нулю) будет очередной цифрой в записи дробной части числа в новой системе;
- дальнейшие операции совершаются над остатком произведения;
- последняя десятичная дробь в новой системе может оказаться бесконечной (периодичной). В этом случае операция считается законченной, если достигнута необходимая точность (определен период дроби).

Переведем, к примеру, дробное число 0,6875 в восьмеричную систему счисления.

$$0{,}6875 \cdot 8 = 5{,}5$$
$$0{,}5 \cdot 8 = 4$$

Таким образом, $0{,}6875_{10} = 0{,}54_8$.

А теперь переведем то же самое дробное число в пятеричную систему счисления. Это, обычно, записывается следующим способом:

0	6875
	× 5
3	4375
	× 5
2	1875
	× 5
0	9375
	× 5
4	6875

Дробная часть числа повторяется, то есть будут возникать одни и те же цифры в одинаковой последовательности. Значит, $0{,}6875_{10}=0{,}(3204)_5$. В скобках указан период дроби.

1. В ящике 31_8 шариков. Из них 12_8 шариков красные, 17_8 – желтые. Докажите, что здесь нет ошибки.
2. В классе 1111_2 девочек и 1010_2 мальчиков. Сколько учащихся в классе?
3. Назовите свой возраст в шестнадцатеричной системе счисления.
4. Запишите число 2008 в двоичной, восьмеричной и шестнадцатеричной системах счисления.
5. Переведите число 321_{10} в семеричную систему.

3.4. ДВОИЧНАЯ СИСТЕМА СЧИСЛЕНИЯ

Как вы знаете, все устройства компьютера работают по сигналам. Эти сигналы удобнее всего представлять в двоичной форме. Элементы компьютерных систем всегда находятся в одном из двух состояний: “есть ток – нет тока”, “открыто – закрыто”, “поднято – опущено”, “намагничено – не намагничено”, и т.д. Эти парные состояния можно показать, используя две цифры двоичной системы счисления: 0 и 1.

Для человеческого восприятия числа, записанные в двоичной системе, слишком длинны и неудобны, например:

$$9999_{10} = 10011100001111_2.$$

Длинные последовательности нулей и единиц плохо воспринимаются и тяжело запоминаются. Для более краткой и удобной записи чисел используются шестнадцатеричная либо восьмеричная системы счисления. Их основания (8 и 16)

тесно связаны с 2: $16=2^4$, $8=2^3$. Между двоичной и восьмеричной, а также двоичной и шестнадцатеричной системами счисления имеются простые связи, которые представлены в следующих таблицах.

8-ная	**2**-ная
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

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

В связи с этим существует очень легкий способ перевода чисел из двоичной системы в шестнадцатеричную или в восьмеричную.

Для перевода целых (и дробных) двоичных чисел в систему счисления с основанием $p=2^n$ можно использовать следующее правило:

– Заданное число разбивается на группы из n символов, начиная с самой первой цифры справа для целых чисел и самой первой цифры слева (после запятой) для дробных чисел. При необходимости для получения полных *n* групп крайняя слева (у дробных чисел - крайняя справа) группа дополняется нулями;

– Каждая группа рассматривается как *n*-значное двоичное число и заменяется соответствующей (эквивалентной) цифрой в системе счисления с основанием *p*;

– Полученная запись будет записью данного числа в системе счисления с основанием *p*.

Например, требуется перевести число 10011100001111_2 в восьмеричную систему счисления. Для этого запишем его следующим образом:

$$\underbrace{010}_{2}\ \underbrace{011}_{3}\ \underbrace{100}_{4}\ \underbrace{001}_{1}\ \underbrace{111}_{7}$$

Заменив каждую группу, состоящую из двоичных цифр, восьмеричной цифрой, получим:

$$10011100001111_2 = 23417_8$$

Вот интересный пример, связанный с двоичной системой записи чисел. Допустим, ваш товарищ задумал произвольное число от 1 до 1000. Вы, задавая ему вопросы, должны отгадать это число. Отвечать можно только “да” или “нет”. Для отгадывания числа используйте не более 10 вопросов.

Первый вопрос может быть примерно такой:

“Раздели задуманное тобой число на 2. Оно делится без остатка?” Если ответ “да”, запишите себе “0”, если “нет” – “1” (другими словами, мы отмечаем остаток от деления этого числа на 2).

Второй вопрос: *“Раздели на 2 частное, полученное при первом делении задуманного числа. Оно делится без остатка?” И вновь, если ответ “да”, запишите себе “0”, если “нет” – “1”.*

Последующие вопросы будут точно такими же.

И всякий раз, если ответ будет “да”, записывайте себе “0”, если “нет” - “1”. Проведя эту процедуру 10 раз, вы получите 10 цифр (единиц и нулей). Нетрудно представить себе их как запись искомого числа в двоичной системе. Ведь ряд заданных вами вопросов – ни что иное, как метод перевода заданного числа в двоичную систему счисления. Десяти вопросов в данном случае вполне хватает, потому что любое число от 1 до 1000 в двоичной системе будет записано не более чем 10-значным числом. Если предположить, что загаданное число уже записано в двоичной системе, то для его определения потребуется система вопросов, при которой о каждой его цифре будет последовательно спрошено, равняется ли она нулю.

Операции над двоичными числами крайне просты.

Следующие равенства демонстрируют правила сложения:

$0 + 0 = 0,$
$0 + 1 = 1$
$1 + 1 = 10_2$

```
  1001
+
  1101
-----------
 10110
```

Таблица умножения
в двоичной системе:

$0 \cdot 0 = 0$
$0 \cdot 1 = 0$
$1 \cdot 0 = 0$
$1 \cdot 1 = 1$

```
    1001
  ×
    1101
--------------
    1001
  1001
1001
-------------
1110101
```

Единственным недостатком двоичной системы счисления является, как уже говорилось ранее, ограниченное количество цифр, в связи с чем запись числа полу-

чается очень длинной. Но несмотря на этот недостаток, двоичная система счисления широко используется в различных областях техники, в частности, в компьютерной сфере.

Как известно, компьютер работает с различной информацией (текстами, изображениями, звуковыми записями). Но сколь бы различной ни была информация, в компьютере она хранится в единой форме, то есть в закодированном виде. Эта единая форма, как уже отмечалось – двоичный код, состоящий из последовательности нулей и единиц. Один разряд двоичного кода (0 или 1) называется бит (от англ. binary digit – "двоичная цифра").

Количество различных символов, которые можно закодировать двоичным способом, зависит от разряда двоичного кода. Один бит может выражать два значения. Двумя битами можно выразить уже четыре значения:

00, 01, 10, 11.

При увеличении количества разрядов кода на единицу количество значений, представленных данным кодом, увеличивается в два раза.

Для кодирования реального текста требуется около 100 различных символов (прописные и строчные буквы, цифры, знаки препинания, часто встречающиеся математические символы и т.д.), поэтому необходим 7-битный либо 8-битный код. Второй вариант более удобен, так как 8 – это степень числа 2:

$$8 = 2^3.$$

В восьмиразрядном коде каждая группа из восьми битов отображает один элемент (символ). Последовательность из восьми битов называется *байтом*.

Один байт способен представлять 2^8=256 различных значений. Но иногда целесообразнее группировать данные не в байты, а в более крупные группы – так, первый из семейства компьютеров PC компании IBM использовал 16-разрядные значения. Группу из 16 взаимосвязанных битов называют "словом". Слово состоит из двух байтов.

Впоследствии возникли такие единицы измерения, как двойное слово (32 бита, или 4 байта) и четверное слово (64 бита, или 8 байт). Обычно в одном байте зашифровывается один символ текстовой информации, поэтому объем текстового документа в байтах, как правило, равен количеству символов в этом документе.

Наиболее часто используемая единица измерения – килобайт (Кбайт). А поскольку в вычислительной технике числа представляются в двоичной системе счисления, то:

1 Кбайт = 1024 байт = 2^{10}байт.

Слово *"byte"* возникло примерно в 1965 году, когда в компании IBM решили обозначить новое понятие словом "bite" (в переводе с англ. "кусок"), но писать его как "byte", во избежание путаницы. Некоторое время словом "байт" обозначали просто количество битов в потоке данных. Но в середине 60-х гг минувшего века, с созданием компанией IBM вычислительной машины "System/360", этим словом стали обозначать группу из *8 битов*.

Но так как человеку удобнее использовать десятичную систему счисления, то в сферах, где это не столь важно, один килобайт рассматривают как тысячу байтов (1Кбайт = 1000байт). Объем текста в один килобайт – примерно полстраницы.

Названия более крупных единиц измерения образуются прибавлением приставок мега-, гига-, тера-.

1 Мбайт (один мегабайт) = 2^{10} Кбайт = 2^{20}байт;

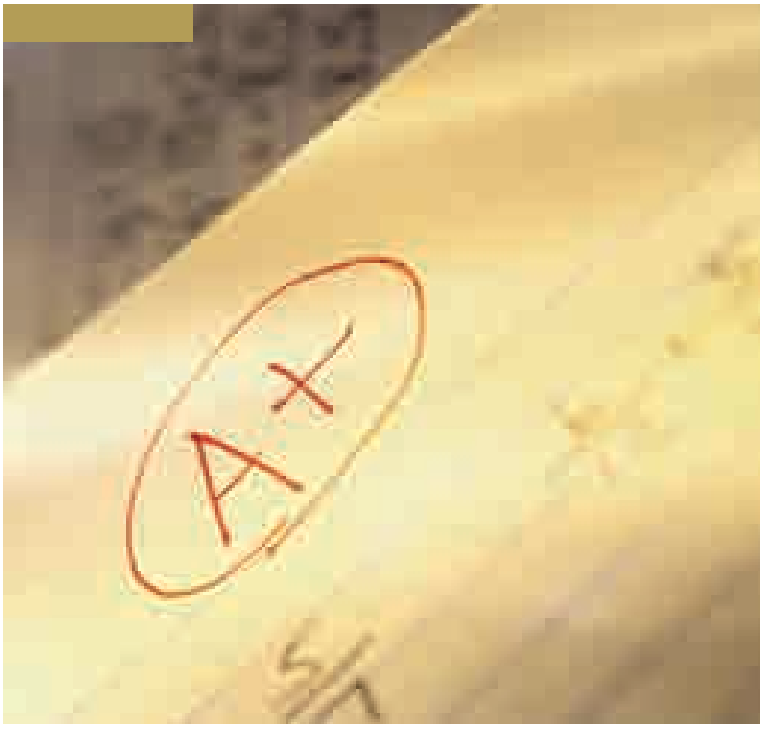
1 Гбайт (один гигабайт) = 2^{10} Мбайт = 2^{30}байт.

1 Тбайт (один терабайт) = 2^{10} Гбайт = 2^{40}байт

1. Почему в компьютерах используется двоичная система счисления?
2. Опишите правила перевода чисел из двоичной системы счисления в восьмеричную и десятичную системы.
3. Переведите двоичные числа 101011011; 1111110011; 100000001110 в восьмеричную систему.
4. Переведите числа A54; 21E; 34AD из шестнадцатеричной системы счисления сначала в двоичную, а потом в восьмеричную.

4

ЛОГИЧЕСКИЕ ОСНОВЫ КОМПЬЮТЕРА



4.1. ФОРМАЛЬНАЯ ЛОГИКА

Во всех современных компьютерах применяется логическая система, созданная **Джорджем Булем**. При обработке данных в компьютере процессор выполняет математические и логические операции над ними. Поэтому для того, чтобы получить полное представление о строении компьютера, важно ознакомиться с логическими элементами, лежащими в его основе.

Логика – наука о законах и формах мышления. Как и в других науках, в логике имеются собственные понятия.

Термин “логика» происходит от древнегреческого слова “logos», которое можно перевести как “слово, мысль, рассуждение».

Джордж Буль
(1815-1864)

В 1847 году Буль написал масштабную статью на тему “Математический анализ логики”, а в 1854 году развил свои идеи в работе “Исследование законов мышления”.
Буль создал оригинальную алгебру – систему правил и обозначений, которые могут быть применены к любым возможным объектам, единицам, цифрам и буквам. В его честь эта алгебра получила название булевой алгебры.

Основные понятия формальной логики. Одним из основных разделов науки логики является формальная логика.

Основные понятия формальной логики таковы:

– высказывание
– утверждение
– рассуждение
– умозаключение.

Высказывание – это предложение, которое может быть либо истинным, либо ложным. Высказывание представляет собой повествовательное предложение, в котором что-либо утверждается либо отрицается. Например:

высказывание *“А” – гласная буква”*

истинно, а высказывание

“Река Араз впадает в Черное море”

ложно.

Существует только две возможные логические оценки:

“истина” и “ложь”.

По-английски эти оценки называются – “true” и “false” и кратко обозначают буквами **T** и **F**.

Логические оценки можно обозначить и двоичными цифрами – нулем (0) и единицей (1). Каким из этих символов обозначать “истину”, а каким – “ложь”, не столь важно. Но обычно за единицу **(1)** принимается “истина”, а за нуль **(0)** – “ложь”.

Следует отметить, что истинность или ложность высказывания определяется не формальной логикой. Этим занимается конкретная наука, к которой имеет

отношение то или иное высказывание. Так, логическая оценка суждения:

“Ананасы растут на деревьях” (“ложь”)

определяется не формальной логикой, а ботаникой.

Утверждение – это высказывание, требующее либо доказательства, либо опровержения. Например:

“Сумма внутренних углов треугольника равна 180°”.

Рассуждение – это цепочка высказываний или утверждений, связанных определенным образом между собой.

Умозаключение – это логическая операция, в результате которой из данного высказывания (или нескольких высказываний) получается новое высказывание.

Доказательство – это цепочка логических умозаключений, показывающая истинность некоторого утверждения при выполнении определенных условий.

Наука, изучающая доказательства математическими методами, называется *математической логикой*. Высказывания делятся на две категории:

– логические утверждения;
– предикаты.

Логические утверждения – это особые утверждения, которые либо истинны, либо ложны (другими словами, логические константы). Нижеследующие высказывания являются логическими константами:

– “2 × 2 = 4” (Истина)
– “Компьютер был изобретен в середине XIX века” (Ложь)

Предикаты – это логические высказывания, меняющие свое значение в зависимости от входящих в их состав величин, другими словами, логических переменных.

Примеры предикатов с логическими переменными:

- **“a+b>c”**
  (в зависимости от значений a, b и c высказывание будет либо истинным, либо ложным);

- *“***N – целое число”** (в зависимости от значения N высказывание будет либо истинным, либо ложным).

1. Что такое высказывание?
2. Приведите по одному примеру истинного и ложного высказывания.
3. Какие из этих высказываний истинны?

а) “Низами Гянджеви – автор произведения “Фархад и Ширин”
б) “Третье тысячелетие началось 1 января 2000 года”
в) “Крокодил сильнее льва”
г) “Самая высокая гора Азербайджана – Савалан”
д) “Два в кубе равняется восьми”

4.2. ЛОГИЧЕСКИЕ ОПЕРАЦИИ

Итак, объекты булевой алгебры – высказывания. При помощи логических операций высказывания объединяются, образуя логические выражения.

Над высказываниями можно произвести три основные операции:

1) инверсия (логическое отрицание),
2) дизъюнкция (логическое сложение),
3) конъюнкция (логическое умножение).

Логическое отрицание. *Логическое отрицание* – унарная (одноместная) операция, так как в ней участвует только одно высказывание. Другими словами, у нее имеется только один аргумент. Высказывания обозначаются, как и в алгебре, прописными латинскими буквами:

A, B, C…

Операцию логического отрицания обозначают частицей “не” (не A), символом ¬ (¬A) или чертой над суждением ($\bar{A}$).

В результате операции отрицания логическое значение высказывания меняется на противоположное, то есть истинное высказывание становится ложным, а ложное – истинным. Это можно показать при помощи следующей таблицы:

Таблица истинности для логического отрицания	
A	¬A
Ложь	Истина
Истина	Ложь

Таблица, включающая все возможные значения исходного высказывания и значения, получаемые в результате операции, называется *таблицей истинности*.

Логическое сложение (дизъюнкция). *Логическое сложение* – бинарная (двухместная) операция, в которой участвуют два высказывания. Другими словами, у нее имеется два аргумента. Эта операция обозначается союзом “или”, знаком V, а иногда символом “+”.

Результат операции логического сложения будет истинным, если или А, или В, или оба одновременно будут истинны. Результат будет ложным, если А и В ложны одновременно.

Рассеянный Анар потерял ключ от дома. Но как только родители сменили замок на двери, он нашел потерянный ключ. Возможно ли создать такой “суперзамок”, чтобы Анар мог открыть его и прежним ключом, но никто посторонний вообще не смог бы его открыть? Как вы думаете, на какой логической операции был бы основан принцип действия такого замка?

Таблица истинности для логического сложения

A	B	A V B
Истина	Истина	Истина
Истина	Ложь	Истина
Ложь	Истина	Истина
Ложь	Ложь	Ложь

Примеры для операции логического сложения:

- “Для того чтобы успешно сдать экзамены, нужны либо знания, либо везение”
- “Высказывание **A**: “p – четное число”, то есть p = {2, 4, 6, 8, 10, …}.

  Высказывание **B**: “p делится на 3”, то есть p = {3, 6, 9, 12, 15, …}.

  Тогда результат логической операции A v B –
  “p – четное число” ИЛИ “p делится на 3” будет
  p = {2, 3, 4, 6, 8, 9, 10, 12, 15, …} – объединение двух множеств.

Логическое умножение (конъюнкция). Логическое умножение – бинарная (двухместная) операция, так как в ней участвуют два высказывания. Другими словами, у нее имеется два аргумента. Эта операция обозначается союзом “и”, знаком $\wedge$ или &, а иногда символом $*$.

Некий богач, опасаясь воров, установил на двери сразу два замка. С какой логической операцией можно сравнить процесс отпирания такой двери?

Результат операции логического умножения будет истинным только в том случае, если A и B истинны одновременно.

Таблица истинности для логического умножения

A	B	A ∧ B
Истина	Истина	Истина
Истина	Ложь	Ложь
Ложь	Истина	Ложь
Ложь	Ложь	Ложь

Чтобы не путать операции логического сложения и логического умножения, запомните такой «факт»: логическое сложение более «покладистое» («Хоть что-нибудь!»), тогда как логическое умножение более «суровое» («Всё или ничего!»)

Примеры операции логического умножения:

- “Педагог должен быть и грамотным, и терпеливым”
- “Для того чтобы достичь цели, нужны умение и упорство”
- Высказывание **A**: “*p* делится на 5”, то есть p = {5, 10, 15, 20, 25, …}.

Высказывание **B**: “p меньше 20”, то есть p = {1, 2, 3, 4, 5, …, 19}.

Тогда результатом логической операции A ∧ B будет: “p делится на 5” И “p меньше 20” будет
p = {5, 10, 15} - пересечение двух множеств.

- Формула успеха Эйнштейна:

$$У = С \wedge О \wedge Т$$

Здесь, У – успех, С – способность, О – обстановка, Т – труд.

Прочие логические операции. Изученные выше операции являются основными логическими операциями. Существуют и другие логические операции, но они не считаются основными, так как их можно выразить посредством трех основных операций. Ознакомимся с двумя из них:

импликацией и эквивалентностью.

Логическое следование (импликация). *Логическое следование* – бинарная операция. Эту операцию можно выразить словами “если A, то B”, для ее обозначения используется символ “$\Rightarrow$”.

Результат импликации будет ложным только в том случае, если исходное условие A истинно, а B – ложно. В остальных случаях он будет истинным.

Таблица истинности для логического следования (импликации)		
A	B	$A \Rightarrow B$
Истина	Истина	Истина
Истина	Ложь	Ложь
Ложь	Истина	Истина
Ложь	Ложь	Истина

Пример операции логического следования:

- Высказывание **A**: “x делится на 9”;

  Высказывание **B**: “x делится на 3”.

  Тогда результатом логической операции $\mathbf{A \Rightarrow B}$ –

  “Если число делится на 9”, то оно делится на 3 – является истина.

Эквивалентность. *Эквивалентность,* как логическая операция, определяет итог сравнения двух простых логических выражений и обозначается знаком "⇔".

Результат такой операции будет истинным только в том случае, если **А** и **В** ложны или истинны одновременно, в противном случае он будет ложным.

Таблица истинности для операции эквивалентности		
А	Б	А ⇔ Б
Истина	Истина	Истина
Истина	Ложь	Ложь
Ложь	Истина	Ложь
Ложь	Ложь	Истина

Пример операции эквивалентности:

- Высказывание **А**: "Сумма цифр числа х делится на 3";

  Высказывание **В**: "х делится на 3".

  Тогда результат логической операции **А** ⇔ **В** –

  "Число делится на 3" только в том случае, когда сумма его цифр делится на 3 – является истина.

Логическое выражение, состоящее из нескольких простых (или сложных) логических выражений, связанных между собой логическими операциями, называется *сложным логическим выражением*.

Пусть, например, A, B и C – три простых логических выражения. Тогда составленное из них сложное логическое выражение может выглядеть так:

$$D = \neg (A \vee B \wedge C)$$

Подобно математическим операциям в математических выражениях, логические операции в сложных логических выражениях выполняются в определенном порядке, в соответствии со степенью значимости. Вот этот порядок:

1. Инверсия $(\neg)$;
2. Конъюнкция $(\wedge)$;
3. Дизъюнкция $(\vee)$;
4. Импликация $(\Rightarrow)$;
5. Эквивалентность $(\Leftrightarrow)$.

Для изменения указанного порядка выполнения операций, в логических выражениях используют скобки.

I. Какие из приведенных предложений являются высказываниями? Опередите их истинность.

1. Перечислите основные устройства компьютера.
2. Кто отсутствует в классе?
3. Стамбул – столица Турции.
4. 1+1=10.
5. Что посеешь, то и пожнешь.
6. Чему равно расстояние от Баку до Тебриза?
7. Пингвины обитают в Арктике.
8. Открой окно!

II. Запишите высказывания в логической форме.

1. Число 19 – нечетное, двухзначное.
2. Корова – не является хищником.
3. Орхан старше Севиль. Тогрул старше Севиль.
4. Если число делится на 2, то оно четное.
5. Если Гюнель – сестра Турала, то Турал – брат Гюнель.
6. Алпай поедет летом в деревню и, если погода окажется хорошая, будет купаться там в речке.

III. Измените высказывания, используя отрицание.

1. Все люди – высокие.
2. Числа 5, 35 и 67 – дробные.
3. Число 18927 не делится на 3.
4. Ночью бывает светло.

4.3. ИСТОРИЯ НАУКИ ЛОГИКА

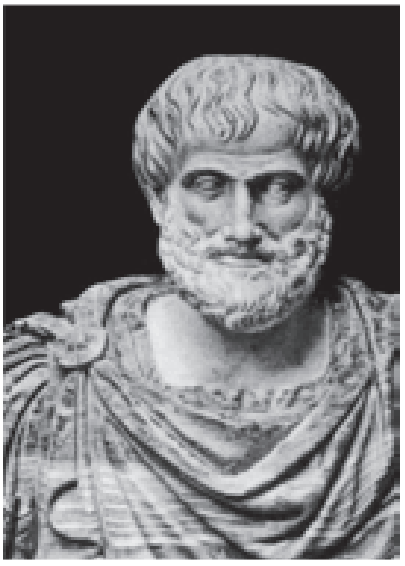
Логика, с которой вы ознакомились, изучает истинные и ложные высказывания. Основы этой логики, именуемой классической, заложил древнегреческий философ **Аристотель**, поэтому ее часто называют *аристотелевой логикой*.

Культура, базирующаяся на этой логике, опирающейся на категоричные условия типа

да / нет, старый / новый, верх / низ, черный / белый

и исключающей третье условие, не менялась тысячелетие.

Исламская культура, приняв за базовую основу аристотелевую логику, выявила и заполнила пробелы в этой логике. **Абу Наср аль-Фараби**, а позднее – **Ибн Сина (Авиценна)** стали использовать в ней не слова, а символы.



Аристотель
(384 – 322 до н.э.)

Абу Наср аль-Фараби
(↔ 870 - 950)

Опираясь на эту логику, исламская культура совершила невиданный прорыв. Результаты его были усвоены Западом, и западная культура вновь, опираясь на эту логику, совершила очередной скачок в развитии. Такая логика, называемая символической, была систематизирована в начале XX века **Б.Расселом**.

Но эта логика не успевала за быстрым развитием науки и техники, ей противоречили научные эксперименты, в особенности – физические явления и законы

физики. Все это послужило толчком к новым исканиям, и наконец, в 1965 году азербайджанец **Лютфи Алескерзаде** предложил миру *нечеткую логику* (*fuzzy logic*), отражающую не идеальную четкость, принятую в классической и символической логике, а неопределенные и неточные реалии, господствующие в природе.

Слово “fuzzy” в переводе с английского означает “мутный”, “туманный”. “Fuzzy logic” дословно переводится как “неясная логика”.

Лютфи Заде, выражая сущность своей теории, отказался от использования классической научной терминологии.

В русском языке эта логика получила название нечеткой, в азербайджанском для ее обозначения принят термин “qeyri-səlis məntiq”, в турецком – “bulanık mantık”.

А в Японии, где наиболее широко применяется эта теория, за этим понятием закрепилось слово “fuzzy”.

В нечеткой логике всё, чего ни коснись, уже не является исключительно *белым* или *черным*, а может обладать любым из оттенков *серого*. Там все уже не является исключительно *хорошим* или *плохим*, а может быть какой-то вариацией хорошего или плохого, став темой дискуссии (например, *“очень плохо”, “неплохо”, “сносно”, “посредственно”, “нехорошо”, “очень хорошо”*).

В классической логике понятия “большой / маленький”, “длинный / короткий” и т.п. имеют точные (четкие) границы. Скажем, высокими принято считать людей, чей рост превышает 170 см. Тогда ответом на вопрос:

“Чингиз – высокий?”

в классической логике, если рост Чингиза, свыше 170 см, будет: “Да, высокий”, если же он равен 169 или меньше, то ответом будет: “Нет, невысокий”.

А вот нечеткая логика спросит: “Насколько высок ростом Чингиз?” В отличие от классической логики, нечеткая логика не дает *конкретных* значений, не обозначает “высокий” как *1*, а “невысокий” как *0*. Она предлагает *более смягченные* значение – *0.1, 0.2, 0.3*... Таким образом, рост 169 см она не назовет невысоким (0), а обозначит, например, как рост 0.2. Разумеется, у нечеткой логики тоже имеются определенные границы применимости, зависящие от конкретных условий. И гибкость этих границ – основное отличие нечеткой логики от классической. Именно эта гибкость (подвижность) приносит гораздо большую пользу и более точные результаты в областях применения нечеткой логики.

В наши дни модели управления, основанные на нечеткой логике, широко применяются в различных сферах: в бытовой технике (пылесосы и стиральные машины с функцией нечеткой логики), в сложных технологических процессах (доменный процесс, процессы в атомном энергоблоке), в управлении динамическими объектами (поезда метро, автомобили, вертолеты, роботы и т.п.).

Лютфи Алескерзаде

Всемирно прославленный ученый, “отец нечеткой логики” **Лютфи Рагим оглу Алескерзаде** родился в Баку 4 февраля 1921 года. Его отец происходил из видного ардебильского рода. В начале 30-х гг, когда ситуация в Советском Союзе, тяжелая и опасная для представителей всех слоев общества, а в особенности для бизнесменов, инженеров, журналистов, достигла своего пика, семья Лютфи приняла решение эмигрировать из страны. В 1931 году, когда Лютфи было 10 лет, они переехали в Тегеран. Мальчик очень увлекался чтением и к 14 годам уже перечитал всех классиков – к этому времени в его личной библиотеке было около 3000 книг. Юный Лютфи закончил Американский колледж Альборц в Тегеране, а затем – Тегеранский университет, получив степень бакалавра по специальности “Электротехника”.

В 1944 году Лютфи переехал в США с целью продолжить обучение. Он поступил в Массачусетский технологический институт (MTU) и в 1946 году получил там степень магистра по специальности “Электротехника”, а затем, в 1949 году в Колумбийском университете, – докторскую степень.

С 1959 года, уже в Калифорнийском университете в Беркли, Лютфи Заде занимается в основном исследованиями в области теории линейных систем и теории

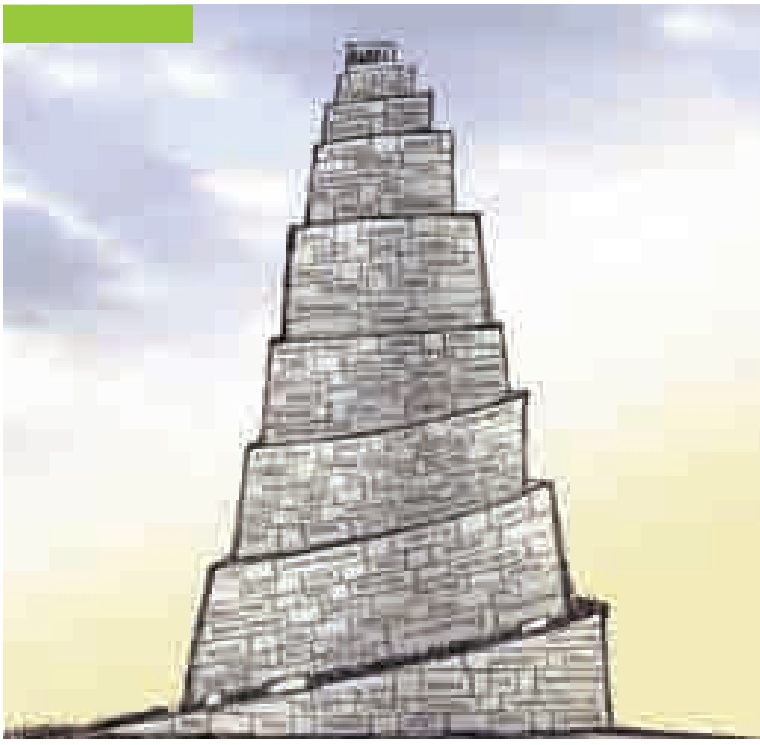
автоматического управления. В 1965 году, в возрасте 44 лет, он публикует первую работу по нечетким множествам, впоследствии принесшую ему мировую известность. Эта работа, имеющая историческое значение, открыла новое направление в науке – нечеткие множества (fuzzy sets). Другими словами, она заложила основы теории нечеткой логики (fuzzy logic). Как и всякая великая идея, эта теория встретила на своем пути немало препятствий (особенно в США) и стала предметом бурных дискуссий. Дело дошло даже до того, что в Департаменте образования США всерьез поднимался вопрос о запрете на ее преподавание. (Кстати, существует мнение, что Лютфи Заде сам спровоцировал трудности на пути выдвижения своей теории, назвав ее “fuzzy” – у него, мол, оказалось бы гораздо меньше противников, если бы он дал ей приемлемое название, ибо слово “нечеткая” никак не сочетается со словом “логика”).

Пока в Штатах шла полемика по этому поводу, японцы представили миру регулятор с функцией нечеткой логики, который по скорости в 100 раз превосходил лучшие PID-регуляторы. Это событие, а также неустанные усилия самого Лютфи Заде по пропаганде его теории, дали наконец свои результаты.

В настоящее время нечеткая логика включена в программы многих университетов мира. А Лютфи Заде и поныне продолжает свои исследования, выдвигает оригинальные идеи и создает новые научные направления. И каждая его статья становится, без преувеличения, событием в научном мире.

1. Кто заложил основы формальной логики?
2. Кто является основоположником нечеткой логики?
3. В чем отличие нечеткой логики от классической?
4. В каких сферах применяется нечеткая логика?

5 ПРОГРАММИРОВАНИЕ



5.1. АЛГОРИТМИЧЕСКИЙ ЯЗЫК

Для того чтобы компьютер смог выполнить ту или иную нужную нам работу, необходимо задать ему соответствующую программу. Чтобы быть понятными для компьютера, программы должны быть написаны по четким правилам. Набор таких правил называется языком программирования. Компьютер, воспринимая программу, написанную на языке программирования, преобразует ее в ту или иную форму, сохраняет в своей памяти. После этого, по желанию пользователя, он может начать выполнять команды данной программы.

Существует множество языков программирования. В числе наиболее популярных из них – FORTRAN, ALGOL, PASCAL, BASIC, ADA, C, LOGO, PROLOG, JAVA и др.

Алгоритм. Одним из основных понятий информатики и математики является понятие алгоритма. *Алгоритм – это точный, конечный набор инструкций, описывающих порядок действий возможного исполнителя для решения поставленной задачи.*

Исполнитель алгоритма – это абстрактная или реальная система, выполняющая инструкции, приведенные в алгоритме.

Школьный алгоритмический язык. Для написания алгоритмов мы будем использовать особый, *школьный алгоритмический язык* (в дальнейшем будем называть его просто алгоритмическим языком).

Правила алгоритмического языка базируются на правилах языков программирования. Поэтому изучение алгоритмического языка в дальнейшем поможет вам в изучении любого из языков программирования. Вот как, например, выглядит алгоритм вычисления суммы квадратов первых ста натуральных чисел на школьном алгоритмическом языке и на языках программирования Basic и Pascal:

a) школьный алгоритмический язык

```
алг сумма квадратов
нач цел s, i
| s: = 0
| нц для i от 1 до 100
| | s: = s + i* i
| кц
| вывод "сумма квадратов =", s
кон
```

b) Basic

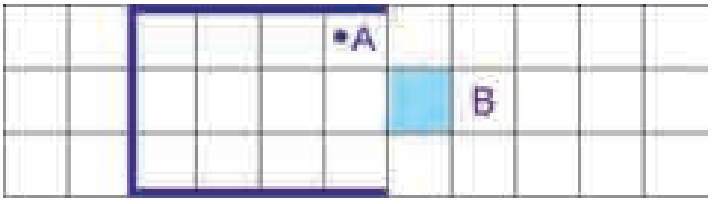
```
10 S = 0
20 FOR I = 1 TO 100
30 S = S + I*I
40 NEXT I
50 PRINT "Сумма квадратов ="; S
60 STOP
```

c) Pascal

```
program sum;
var s, i : integer;
begin
    s := 0;
    for i := 1 to 100 do
        s := s + i*i;
    writeln ('Сумма квадратов =', s)
end.
```

Изучим составление и написание алгоритмов на алгоритмическом языке при помощи алгоритмов, управляющих исполнителями “Робот” и “Чертежник”.

Исполнитель “Робот”. Робот работает на клеточном поле и занимает ровно одну клетку (квадратик) поля. Клетки могут быть разделены стенами. На бумаге, классной доске или на экране компьютера рабочую среду Робота изобразим следующим образом:



Ознакомимся с пятью командами, которые могут быть даны Роботу:

“вверх”, “вниз”, “вправо”, “влево”, “закрасить”.

При выполнении команд “вверх”, “вниз”, “вправо”, “влево” Робот перемещается на соседнюю клетку. Иногда эти команды оказываются невыполнимыми. Например, на вышеприведенном рисунке видно, что над Роботом имеется стена, а значит, в этом случае выполнить команду “вверх” из точки A невозможно.

При выполнении команды “закрасить” Робот закрашивает клетку, на которой в данный момент находится. Если клетка уже окрашена, при выполнении этой команды она окрашивается заново.

Допустим, нужно переместить Робота из клетки A в клетку B. Вот возможный алгоритм выполнения этой задачи, составленный на алгоритмическом языке:

```
алг ход конем
  дано | Робот в клетке A
  надо | переместить Робота в клетку B
нач
  вправо
  вправо
  вниз
кон
```

Общие правила алгоритмического языка. Подобно любому языку, алгоритмический язык имеет собственный *словарь*. Его основу составляют слова, используемые для записи команд, подаваемых исполнителю. Определенную часть их составляют слова, смысл и правила использования которых определены раз и навсегда. Это *служебные* (*ключевые*) слова. В записи алгоритма служебные слова обычно бывают представлены в сокращенном виде, а также каким-либо образом выделены (подчеркнуты, даны жирным шрифтом или другим цветом).

Основные служебные слова

алг (алгоритм)
арг (аргумент)
рез (результат)
нач (начало)
кон (конец)
цел (целый)
вещ (вещественное)
сим (символьный)
лит (литерный)
лог (логический)
таб (таблица)
нц (начало цикла)
кц (конец цикла)
длин (длина)
дано
надо
если
то
иначе
все
пока
для
от
до
знач
и
или
не
да
нет
при
выбор
ввод
вывод

Общий вид алгоритма в алгоритмическом языке:

```
алг название алгоритма
    дано | условия применения
           алгоритма
    надо | цель выполнения
           алгоритма
нач
    тело алгоритма (последова-
    тельность команд)
кон
```

Каждый алгоритм делится на две части: заголовок и тело. Часть алгоритма до служебного слова **нач** называется *заголовком алгоритма*, а часть между служебными словами **нач** и **кон** – телом алгоритма.

У каждого алгоритма, написанного на алгоритмическом языке, должно быть название. Оно подбирается так, чтобы четко отражать назначение алгоритма, то есть показывать, для решения какой задачи он предназначен. Для выделения названия алгоритма перед ним записывается служебное слово **алг** (алгоритм).

Служебные слова **алг**, **нач** и **кон** записываются друг под другом, причем слова **нач** и **кон**, а также отдельные блоки алгоритма во избежание путаницы соединяются вертикальной чертой |. Справа от этой черты записывается *последовательность команд* – тело алгоритма.

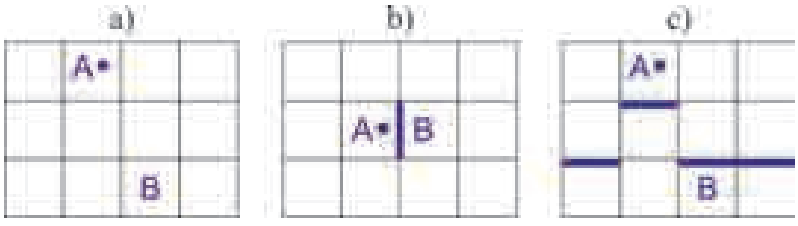
Команды алгоритма записываются последовательно, причем каждая – с новой строчки. Если же несколько команд записываются на одной строчке, то они отделяются друг от друга *точкой с запятой* - “ ; ” .

Комментарии. В алгоритме “Ход конем” в строчках **дано** и **надо** после знака “ | ” записаны комментарии. Комментарии записывают в конце нужной строчки

после знака “ | ” , а если комментарий многострочный, то знак “|” ставится перед его продолжением на каждой строке. При выполнении алгоритма компьютер пренебрегает комментариями, словно их и нет вовсе, потому что комментарии не подлежат выполнению – они предназначены лишь для облегчения понимания смысла алгоритма.

Вызов команд. При выполнении алгоритма компьютер последовательно отдает Роботу приказы выполнить команды, записанные в алгоритме. Каждый такой приказ исполнителю выполнить команду называется вызовом команды.

1. Что такое алгоритмический язык и для чего он предназначен?
2. Что такое служебные слова? Раскройте значение служебных слов **алг**, **нач**, **кон**.
3. Измените алгоритм “Ход конем”, подобрав альтернативный маршрут.
4. Составьте алгоритм для перемещения Робота в заданной рабочей среде из клетки A в клетку B.



5.2. ВСПОМОГАТЕЛЬНЫЕ АЛГОРИТМЫ

При составлении алгоритма решения очередной задачи можно воспользоваться ранее составленными алгоритмами. Алгоритмы, целиком использованные в составе других алгоритмов, называют *вспомогательными*. Алгоритм, в котором использованы вспомогательные алгоритмы, при определенных условиях и сам может стать

вспомогательным для другого алгоритма. Использование ранее составленных алгоритмов для составления новых широко применяется в практике алгоритмизации. Прежде чем рассмотреть правила использования вспомогательных алгоритмов, ознакомимся еще с одним исполнителем.

Исполнитель "Чертежник". Чертежник может составлять рисунки, графики, схемы на бесконечном поле. У Чертежника есть перо, которое бывает поднятым или опущенным. Оно может перемещаться по плоскости. При перемещении пера в опущенном состоянии остается след.

Чертежник может выполнять 4 команды:

подними перо
опусти перо
переведи в точку (**арг вещ** x, y)
сдвинь на вектор (**арг вещ** a, b)

арг (**арг**умент) и **вещ** (**вещ**ественное) показывают, что у команд

"переведи в точку"

и

"сдвинь на вектор"

имеются аргументы, которые могут быть любыми действительными числами. Аргументы указываются при вызове команды, например:

"переведи в точку (2.5, 0.5)".

Команда "сдвинь на вектор". Если перо Чертежника находится в точке (x, y), то по команде

"сдвинь на вектор (a, b)"

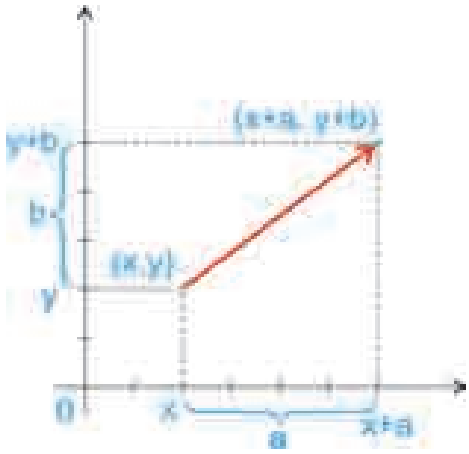
оно перейдет в точку

(x+a, y+b).

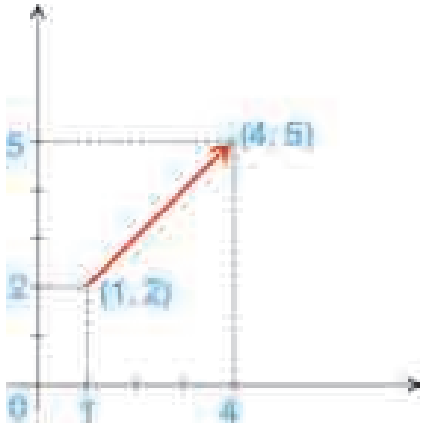
Если a – положительное число, то движение будет направлено вправо, если отрицательное – то влево. Точно так же, если b – положительное число, движение будет направление вверх, а если отрицательное – вниз. Таким образом, если a и b – положительные целые числа, то по команде

"сдвинь на вектор (a, b)"

перо Чертежника передвинется на a клеток вправо и на b клеток вверх

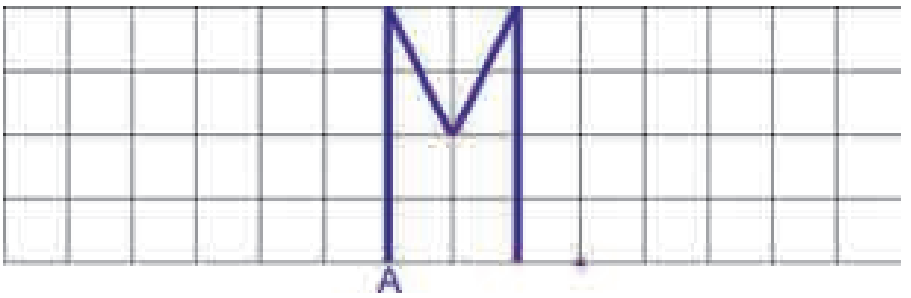


а)



б)

Когда перо Чертежника опущено, при его передвижении из одной точки в другую, от точки (x, y) до точки (x + a, y + b) чертится отрезок. Этот отрезок - вектор с координатами (a, b), перенесенный в точку (x, y) – отсюда и название соответствующей команды. Например, если перо Чертежника опущено и находится в точке (1, 2), то при выполнении команды “сдвинь на вектор (3, 3)” оно перейдет в точку (1 + 3, 2 + 3), то есть прочертит вектор из точки (1, 2) в точку (4, 5).



При помощи Чертежника можно начертить любую фигуру, состоящую из отрезков. Давайте составим алгоритм, согласно которому Чертежник может изобразить на клеточном поле букву **“М”**. Если перо не установлено в исходное положение, надо будет использовать команду “сдвинь на вектор”.

```
алг буква М
  дано | перо поднято и находится в точке А
  надо | чтобы Чертежник начертил букву М и
       | поднял перо

нач
| опусти перо
| сдвинь на вектор (0, 4)
| сдвинь на вектор (1, -2)
| сдвинь на вектор (1, 2)
| сдвинь на вектор (0, -4)
| подними перо
кон
```

Можно написать схожие алгоритмы и для вычерчивания других букв алфавита. Допустим, что мы написали алгоритмы для вычерчивания букв **А**, **Л**, **П**. Тогда алгоритм для вычерчивания слова **АЛП** можно записать так::

```
алг слово АЛП
дано  | перо поднято
надо  | чтобы Чертежник начертил слово АЛП и
      | поднял перо

нач
|  буква А
|  буква Л
|  буква П
кон
```

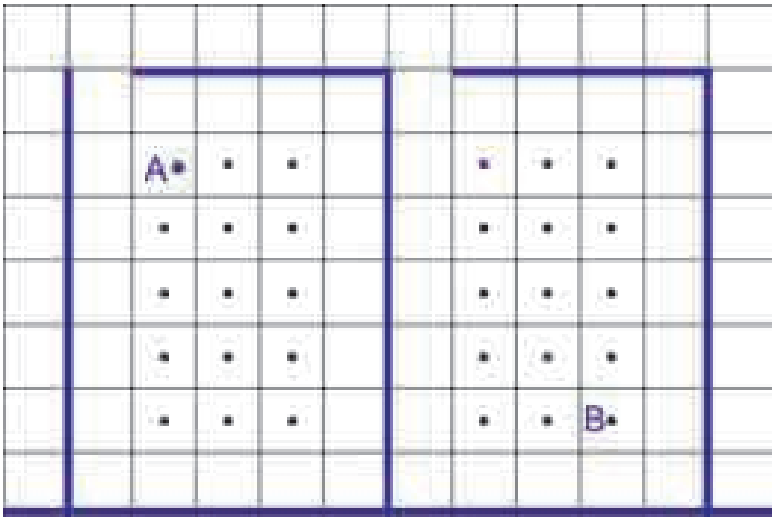
В данном случае команды “буква А”, “буква Л”, “буква П” служат вспомогательными алгоритмами. Алгоритм же “слово АЛП” – основной.

Теперь рассмотрим такой алгоритм:

```
алг имя АЛПАЙ
 дано | Перо поднято
 надо | Написать слово АЛПАЙ
      | и поднять перо
нач
   слово АЛП
   буква А
   буква Й
кон
```

В этом примере алгоритм “слово АЛП” относительно алгоритма “имя АЛПАЙ” является вспомогательным алгоритмом, а относительно алгоритма “буква А” – уже основным.

1. Что такое вспомогательный алгоритм?
2. Напишите для Робота алгоритм прохождения из точки А в точку В с закрашиванием отмеченных на рисунке клеток.



3. Изобразите результаты выполнения следующих алгоритмов:

а)

```
алг тоннель
нач
  квадрат(10); сдвинь на вектор (1, 1)
  квадрат(7); сдвинь на вектор (1, 1)
  квадрат(4); сдвинь на вектор (1, 1)
  квадрат(1);
кон

алг квадрат (арг вещ a)
нач
  опусти перо
  сдвинь на вектор (a, 0)
  сдвинь на вектор (0, a)
  сдвинь на вектор (-a, 0)
  сдвинь на вектор (0, -a)
  подними перо
кон
```

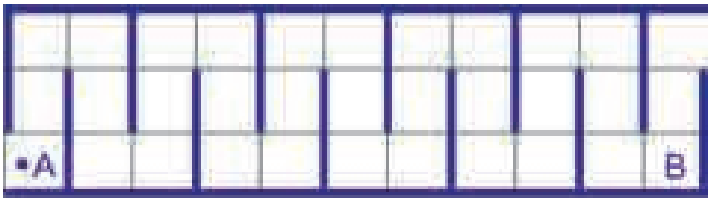
б)

```
алг спираль
нач
  опусти перо
  виток (1); виток (3); виток (5);
  виток (7); виток (9)
  подними перо
кон

алг виток (арг вещ a)
нач
  сдвинь на вектор (a, 0)
  сдвинь на вектор (0, -a)
  сдвинь на вектор (-a-1, 0)
  сдвинь на вектор (0, a+1)
кон
```

4. Напишите алгоритмы вычерчивания букв **А**, **Л**, **П**.

5.3. ЦИКЛЫ

Рассмотрим следующую задачу. Пусть требуется переместить Робота из клетки A в клетку B.



Путь, который должен пройти Робот, можно разделить на пять одинаковых частей. Если прохождение каждой такой части записать в виде команд на одной строке, то алгоритм решения данной задачи будет таким:

```
алг переход из клетки А в клетку В
  дано | Робот находится в клетке А
  надо | чтобы Робот оказался в клетке В

нач
| вверх; вверх; вправо; вниз; вниз; вправо
| вверх; вверх; вправо; вниз; вниз; вправо
| вверх; вверх; вправо; вниз; вниз; вправо
| вверх; вверх; вправо; вниз; вниз; вправо
| вверх; вверх; вправо; вниз; вниз; вправо
кон
```

При составлении алгоритмов часто возникает необходимость многократного повторения последовательности определенных команд. В этом случае для упрощения записи алгоритма используют специальную команду алгоритмического языка - цикл типа n раз.

Цикл типа n раз.

Общий вид цикла типа **n раз** таков:

```
нц количество раз
| тело цикла (последовательность команд)
кц
```

Служебные слова **нц** (начало цикла) и **кц** (конец цикла) записываются одно под другим и соединяются вертикальной чертой. Справа от этой черты записываются повторяющиеся команды (тело цикла). При выполнении алгоритма последо-

вательность этих команд повторяется указанное количество раз. Согласно правилам алгоритмического языка, количество повторений может выражаться любым целым числом. Другими словами, оно может равняться нулю или быть отрицательным. Это не считается ошибкой – в этом случае тело цикла просто не будет выполняться, и управление перейдет к команде, следующей сразу за **кц**.

Теперь запишем приведенный выше алгоритм с использованием команды цикла:

```
алг переход из клетки А в клетку В
  дано | Робот находится в клетке А.
  надо | чтобы Робот оказался в клетке В.

нач
| нц 5 раз
| | вверх; вверх; вправо; вниз; вниз; вправо
| кц
кон
```

При использовании цикла **n раз** можно переместить Робота на сколько угодно клеток в каком угодно направлении:

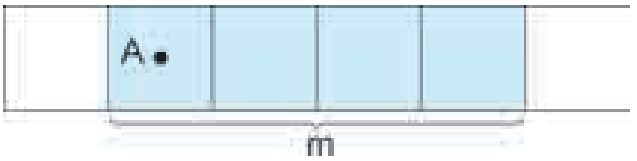
```
алг вверх (арг вещ n)
  дано | на поле Робота нет стен.
  надо | чтобы Робот переместился
       | на n клеток вверх.

нач
| нц n раз
| | вверх
| кц
кон
```

По этому же принципу можно составить алгоритмы “вниз (**арг вещ** n)”, “вправо (**арг вещ** n)” и “влево (**арг вещ** n)”.

Рассмотрим следующий алгоритм:

```
алг закрась ряд (арг вещ m)
  дано | на поле Робота нет стен.
       | Робот находится в клетке А.
  надо | чтобы Робот  последовательно закрасил
       | m клеток справа от точки А
       | и вернулся в точку А.
```

```
нач
| нц повторить m раз
|   закрась; вправо
| кц
| влево (m)
кон
```



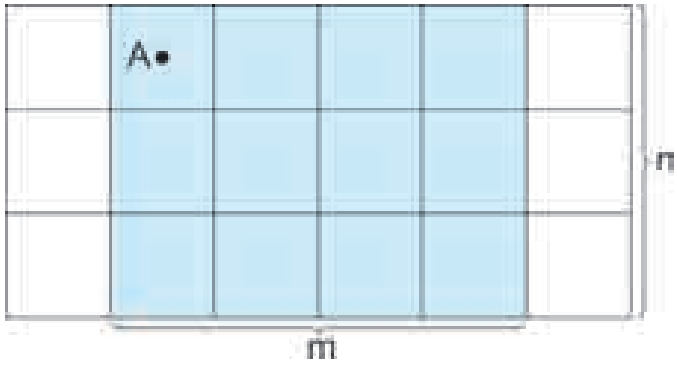
При m = 4 Робот, исполнитель алгоритма “закрась ряд (4)”, выполняет 12 команд:

```
закрась; вправо
закрась; вправо
закрась; вправо
закрась; вправо
влево; влево; влево; влево
```

Вызов вспомогательных алгоритмов внутри цикла. К вспомогательным алгоритмам можно обращаться в любом месте основного алгоритма, в том числе и внутри цикла. В приведенном ниже алгоритме внутри цикла n раз повторяется вспомогательный алгоритм

“закрась ряд”

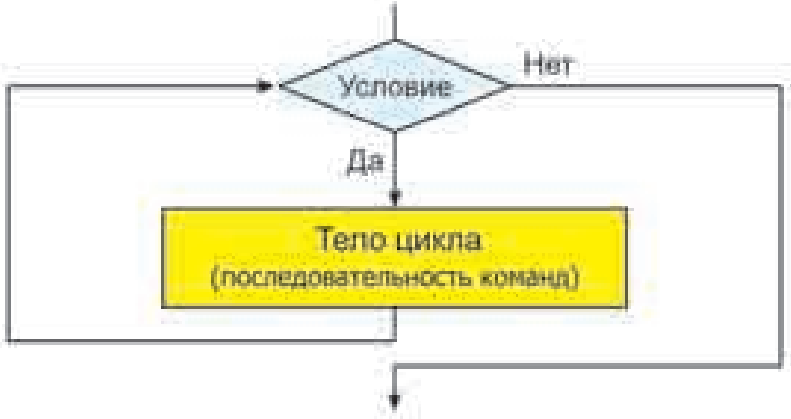
```
алг закрасить прямоугольник (арг вещ m, n)
  дано | на поле Робота нет стен.
       | Робот находится в клетке A.
  надо | чтобы Робот закрасил прямоугольник
       | размером m*n и вернулся в точку A.

нач
| нц повторить n раз
|   закрась ряд (m)
|   вниз
| кц
| вверх (n)
кон
```



Цикл типа пока. В рассмотренных выше алгоритмах количество повторений для команд, отдаваемых Роботу, было известно заранее. Однако зачастую заранее не известно необходимое количество повторений в цикле. В этом случае используется цикл типа **пока**.

Общий вид цикла **типа** пока таков:

```
нц пока условие
| тело цикла (последовательность команд)
кц
```

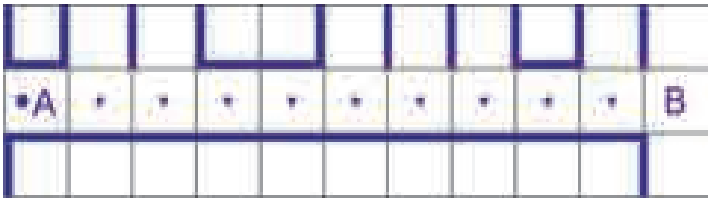


Составить такой цикл не так-то легко. Выглядящий кратким цикл типа **пока** может включать довольно длинную последовательность операций. Во избежание ошибок циклы типа **пока** нужно составлять пошагово:

1. Определить, когда цикл должен завершиться, то есть условие его окончания, и записать после служебного слова **пока** противоположное условие, то есть условие продолжения цикла.

2. Выяснить, что и как должно меняться в цикле и определить промежуточное состояние после нескольких повторений.
3. Представить, что должно произойти после выполнения цикла один раз (его еще называют “шагом цикла”), то есть написать тело цикла.
4. Проверить, не является ли цикл бесконечным, то есть проконтролировать его завершение.

Рассмотрим следующую задачу. Допустим, Робот находится в клетке A - крайней левой точке горизонтального коридора. Стена внизу коридора сплошная, в стене вверху коридора имеется несколько проходов. Требуется составить алгоритм, согласно которому Робот смог бы переместиться из клетки A в клетку B таким образом, чтобы закрасить все клетки коридора.



Если количество клеток коридора заранее неизвестно, то при написании алгоритма не обойтись без цикла типа **пока**.

Когда должен завершиться цикл?

Когда Робот переместится в клетку B.

Но чем отличается клетка B от остальных клеток коридора?

Как видно из рисунка, под всеми клетками коридора проходит стена, и только под клеткой B ее нет. Значит, после служебного слова **пока** следует указать условие “внизу стена”. При выполнении каждого шага алгоритма Робот должен закрашивать клетку, в которой находится, и перемещаться на следующую.

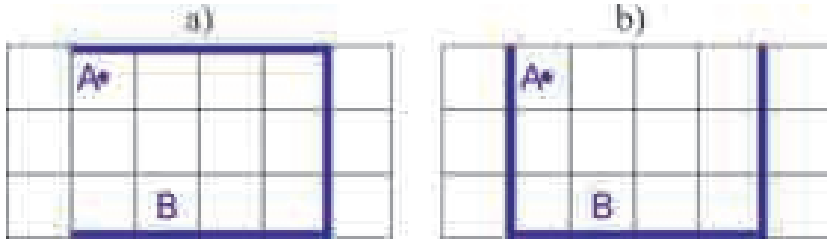
```
алг закрасить коридор
 дано | Робот находится в левом конце коридора.
 надо | Робот должен достигнуть правого края коридора
      | закрасив его клетки.
нач
| нц пока внизу стена
| | закрась
| | вправо
| кц
кон
```

1. Какие существуют команды цикла? Сравните их.
2. Что нужно уяснить для составления цикла типа **пока**?
3. Сколько клеток будет закрашено и сколько команд компьютер даст Роботу после каждого из этих вызовов?

   а) закрась прямоугольник (1, 1)
   б) закрась прямоугольник (0, 11)
   в) закрась прямоугольник (9, 0)
   г) закрась прямоугольник (9, 11)

4. Робот находится в клетке A. Каким будет результат выполнения цикла для двух случаев?

```
нц пока вверху нет стены
| вправо
кц
```



5. Робот находится в прямоугольнике, огражденном стенами. Внутри прямоугольника других стен нет. Составьте алгоритм, согласно которому Робот закрасит все клетки вдоль стен.

5.4. УСЛОВИЯ В АЛГОРИТМИЧЕСКОМ ЯЗЫКЕ. ВЕТВЛЕНИЕ

Простые алгоритмы линейны – команды в них выполняются последовательно, в порядке их записи в алгоритме. Но, обычно, в реальной жизни линейные алгоритмы, отображающие решение задач, встречаются крайне редко (в противном случае жизнь стала бы невероятно скучной и убогой).

Жизненные задачи гораздо более сложны, и для их решения требуются гибкие алгоритмы, меняющие порядок выполнения команд в процессе реализации алгоритма.

Выбор нужного порядка выполнения команд из нескольких возможных вариантов при переходе к очередному шагу алгоритма достигается посредством *ветвления*.

Ветвление – одна из основных алгоритмических структур. Оно основано на проверке одного или нескольких условий и выполнении определенных операций в зависимости от того, какое из условий оказалось верным.

В алгоритмическом языке структура ветвление реализуется с помощью команд **если** и **выбор**.

Команда если

Общий вид команды **если** имеет вид:

```
если условие
| то действия 1
| иначе действия 2
всё
```

или:

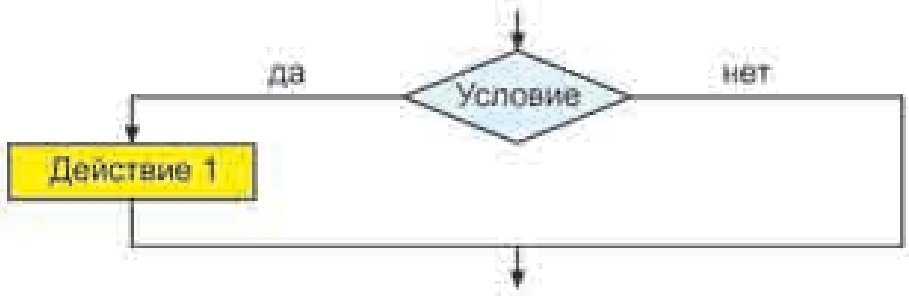
```
если условие
| то действия 1
всё
```

Служебные слова **если, то, иначе** имеют тот же смысл, что и в обычном языке, а слово **все** означает конец команды. Оно записывается прямо под словом **если** и соединяется с ним вертикальной чертой. Между словами **то** и **иначе** записывается в одну или в несколько строчек последовательность команд алгоритмического языка (действия 1), между словами **иначе** и **все** – другая последовательность команд (действия 2). Последовательности команд между словами **иначе** и **все** (действия 2) может и не быть.

При выполнении команды **если** компьютер проверяет условие, указанное между словами **если** и **то**. Если условие удовлетворяется, то есть ответ “да”, то выполняются действия 1. Если же условие не удовлетворяется, то есть ответ “нет”, то выполняется последовательность команд, идущих после слова **иначе** – действия 2 (если она есть).

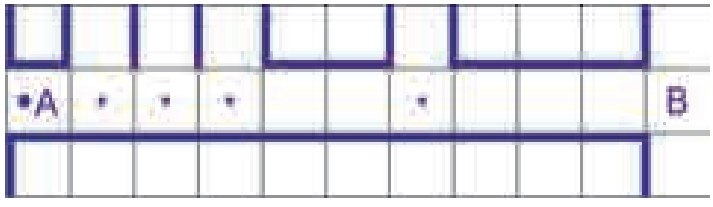
Если же условие не удовлетворяется и нет слова **иначе** и идущих после него действий, то компьютер переходит к выполнению команд, следующих после слова **все**.

а) блок-схема команды с двуальтернативным условием



б) блок-схема команды с одноальтернативным условием

Рассмотрим следующую задачу. Робот находится в крайней левой точке горизонтального коридора. Стена внизу коридора сплошная, в стене вверху коридора имеется несколько проходов. Требуется составить алгоритм, согласно которому Робот смог бы переместиться из клетки А в клетку В таким образом, чтобы закрасить клетки коридора, над которыми имеются проходы. Длина коридора, количество проходов и порядок их расположения неизвестны.



Вспомним похожую задачу из пройденной темы. В ней требовалось закрасить все клетки коридора. Эта задача отличается от предыдущей тем, что нужно закрашивать только те клетки, над которыми имеются проходы. Другими словами,

если имеется над клеткой проход, то закрасить ее, в противном случае – нет. Для записи этого на алгоритмическом языке воспользуемся командой условия:

```
если вверху пусто
| то закрась
все
```

При выполнении этой команды компьютер спрашивает у Робота: “вверху пусто?”. Если Робот отвечает “да”, то компьютер дает ему команду “закрась”. Если же Робот отвечает “нет”, то перехода к команде “закрась” не происходит.

```
алг отметить проходы коридора
 дано | Робот находится в крайней левой клетке коридора
 надо | чтобы Робот достиг крайней правой клетки
      | коридора и закрасил те его клетки,
      | над которыми имеются проходы
нач
| нц пока внизу стена
| | если вверху пусто
| | | то закрась
| | всё
| | вправо
| кц
кон
```

Условия в рассмотренных нами командах **если** и **пока** можно выразить одним вопросом, например “Вверху пусто?” или “Внизу стена?” Условия, выраженные в такой форме, называются простыми.

Условия, в которых совместно применяются простые условия, служебные слова **и**, или и **не**, а также скобки для указания последовательности проверки условий, называются составными.

Например:

- `вверху пусто` **`или`** `слева пусто`
- `клетка закрашена` **`и`** `справа пусто`
- `x > 0` **`и`** `y > 0`
- **`не`** `(x=0` **`и`** `y=0)`

Команда выбор. Как вы заметили, максимальное количество выбора вариантов в команде **если** всего два. Однако если количество возможных вариантов довольно велико, то использовать данную команду не целесообразно.

Рассмотрим, к примеру, алгоритм оценки знаний учащихся:

```
если ответ отличный
  то оценка := 5
  иначе
  если ответ хороший
    то оценка := 4
    иначе
    если ответ удовлетворительный
      то оценка := 3
      иначе
      если ответ неудовлетворительный
        то оценка := 2
      все
    все
  все
все
```

Как можно заметить, в этом алгоритме несколько команд условного перехода размещены одна внутри другой. При составлении и прочтении такого алгоритма возникают определенные сложности. Если же вариантов очень много, то работать с таким алгоритмом становится весьма тяжело. На этот часто встречающийся случай во всех языках программирования, в том числе и в школьном алгоритмическом языке, предусмотрена более удобная команда. Это команда **выбор**.

Общий вид команды **выбор** таков:

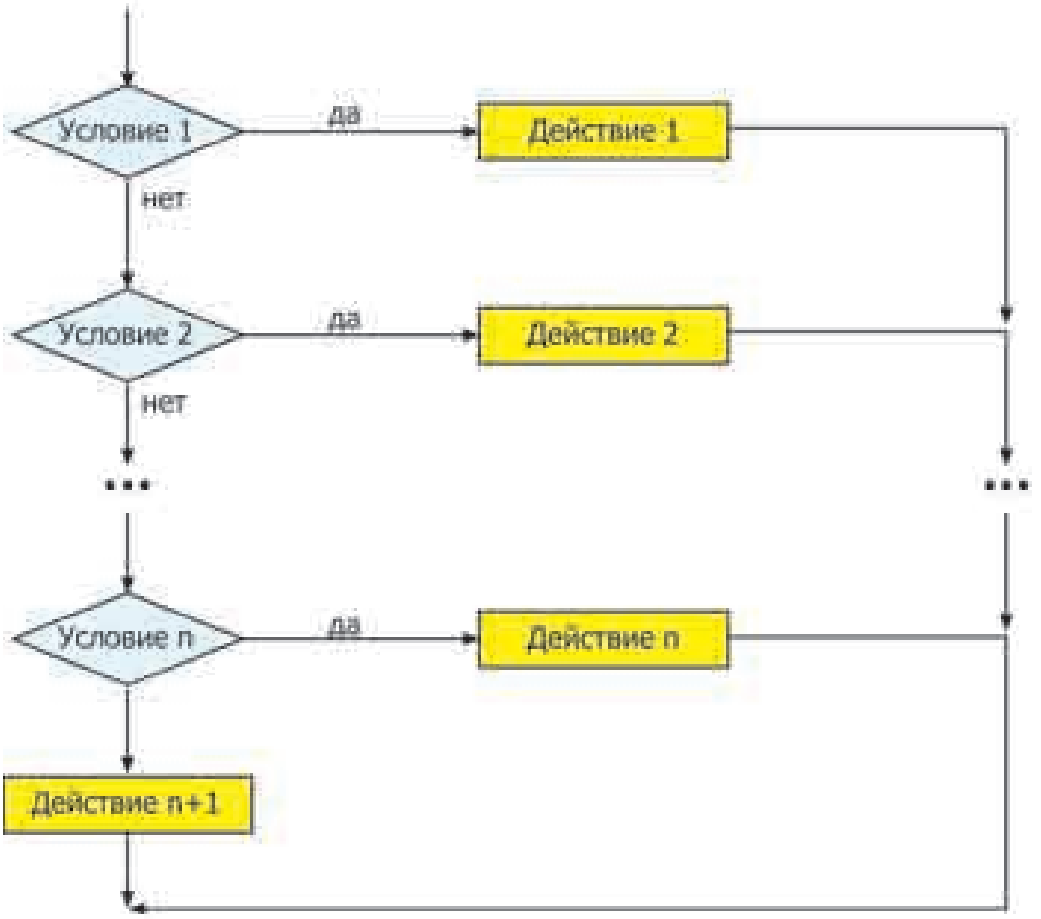
```
выбор
  при усл 1: действия 1
  при усл 2: действия 2
  ...
  при усл n: действия n
  иначе действия n+1
все
```

Как и в случае с командой **если**, здесь также может отсутствовать вариант

иначе. Используя команду **выбор**, можно записать вышеприведенный алгоритм об оценке следующим образом:

```
выбор
  при ответ отличный : оценка := 5
  при ответ хороший : оценка := 4
  при ответ удовлетворительный : оценка := 3
  при ответ неудовлетворительный : оценка := 2
все
```

Можно заметить, что эта запись алгоритма короче и понятнее записи этого алгоритма с командой **если**.



Блок-схема исполнения команды **выбор**

Рассмотрим еще одну задачу. Допустим, что Робот находится в клетке, у которой нет хотя бы одной стены и требуется вывести Робота из этой клетки. Если воспользоваться командой **выбор**, то алгоритм решения этой задачи будет таким:

```
алг выход из клетки
| дано | Робот находится в клетке, у которой нет
       | хотя бы одной стены
| надо | вывести Робота из клетки
нач
выбор
    при вверху пусто : вверх
    при справа пусто : вправо
    при внизу пусто : вниз
    при слева пусто : влево
все
```

Здесь указаны четыре условия для команды **выбор** и четыре действия, соответствующие им. Но бывает, что при совпадении выполнимости нескольких условий выполняется только одно действие. Например, если при решении данной задачи клетка ни с одной стороны не ограничена стеною, то есть открыта со всех сторон, то при выполнении вышеприведенного алгоритма будет выполнена только первая по счету команда, то есть “вверх”.

1. Что такое ветвление? С помощью каких команд алгоритмического языка оно выполняется?
2. В каких случаях удобнее использовать команду **выбор**?
3. Какую задачу решает данный фрагмент алгоритма?

```
если a >= b
| то c := a
| иначе c := b
все
```

4. Допустим, переменная **день** отражает порядковый номер дня в неделе. Используя команду **выбор**, составьте алгоритм определения названий дней недели соответственно номеру, указываемому переменной **день**.

5.5. ПЕРЕМЕННЫЕ ВЕЛИЧИНЫ В АЛГОРИТМИЧЕСКОМ ЯЗЫКЕ

Рассмотрим следующую задачу. Допустим, что Робот, как показано на рисунке, находится в точке A над стеной, длина которой неизвестна. Требуется переместить Робота в точку B, находящуюся прямо под точкой A снизу от этой стены.

Не имея возможности "протаранить" стену, Робот вынужден двигаться вдоль нее. Поэтому эту задачу можно решить так: если под Роботом имеется стена, то он перемещается вправо. Одновременно подсчитывается количество пройденных клеток. Когда стена закончится, надо спустить Робота вниз и начать перемещать влево. Вот здесь нам и понадобится информация о количестве пройденных им клеток. Если задать Роботу движение влево именно на такое число клеток, он попадет как раз в клетку B.

При составлении этого алгоритма для компьютера требуется запоминать, изменять и использовать соответствующую информацию (в нашем случае - информацию о количестве шагов вправо). Для этого в алгоритмическом языке используются *переменные величины* или просто *переменные*.

Понятие переменной было заимствовано из математики и физики. При составлении алгоритмов, предназначенных для решения математических и физических задач, переменные алгоритмического языка обычно соответствуют математическим и физическим переменным величинам.

Каждая переменная характеризуется *именем*, *значением* и *типом*. Имя переменной (например, n, m, d) – это обозначение величины в алгоритме. При выполнении алгоритма каждая переменная либо может иметь *значение*, либо ее значение не определено. Если значение переменной может быть только целым числом, то это *целая переменная* (**цел**), если же любым числом, как целым, так и дробным – то это *действительная* или *вещественная переменная* (**вещ**). Эта характеристика (в данном случае – быть целой или вещественной переменной) называется *типом переменной*. Существуют и другие типы переменных величин - *логический* (**лог**), *символьный* (**сим**) и др.

Описание переменной. Для того чтобы компьютер мог работать с переменной, надо заранее указать ее имя и тип, например, **цел** n. Такое указание называется *описанием переменной*. Описания переменных, используемых в промежуточных вычислениях, располагаются в строке непосредственно за словом **нач**.

```
алг обогнуть стену
  дано | Робот сверху от стены
  надо | чтобы Робот оказался по другую сторону стены
нач цел n
```

Команда присваивания. Для сохранения или изменения переменной в алгоритмическом языке имеется специальная команда - команда присваивания. Общий ее вид таков:

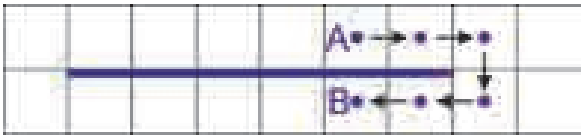
имя переменной := выражение

Знак ":=" называется знаком присваивания и читается как "присвоить". Например, команда "n:=a" читается как "значение a присвоить переменной n". При выполнении команды присваивания компьютер сначала вычисляет значение выражения в правой части, а затем переносит полученное значение в то место памяти, которое отведенно для переменной, стоящей в левой части команды присваивания.

Составим алгоритм "обогнуть стену", используя для количества пройденных вправо шагов целую переменную n.

```
алг обогнуть стену
  дано | Робот сверху от стены.
  надо | чтобы Робот оказался по другую сторону стены.
нач цел n
  | n := 0
  | нц пока внизу стена
  |   вправо; n := n + 1
  | кц
  | вниз
  | влево (n)
кон
```

Проследим, как выполняется этот алгоритм. Сначала компьютер присваивает переменной n значение 0 (n := 0). Затем он дает Роботу команду "вправо" и увеличивает значение n на единицу (n := n+1). При каждом выполнении тела алгоритма значение переменной n будет равно количеству сделанных Роботом шагов вправо. Поэтому при выполнении вспомогательного алгоритма "влево (n)" компьютер n раз даст Роботу команду "влево", в результате чего Робот переместится точно в нужную клетку.



Табличные величины. Секрет мощи компьютера – в высокой скорости его работы и в огромной памяти. При выполнении всех приведенных здесь алгоритмов в память компьютера было заложено, в общем-то, ничтожное количество информации – одно или несколько действительных (целых) чисел. Но как поступить, если надо иметь дело с сотнями, тысячами, сотнями тысяч и более чисел? Для написания алгоритмов, работающих с большими объемами информации, в алгоритмическом языке предусмотрены специальные *табличные величины* (*табличные переменные*), или просто *таблицы*.

Табличные переменные формируются из других переменных (обычно – целых или действительных), которые называют *элементами таблицы*. Наиболее часто используется такой тип таблиц, как линейные. Как и прочие переменные величины, линейные таблицы тоже занимают место в памяти компьютера и характеризуются именем и типом.

Например:

цел таб k

значения элементов	**9**	**4**	**0**	**-17**	**123**
индексы элементов	1	2	3	4	5

Запись **“цел таб k”** означает, что величина **k** – табличная (**таб**) и состоит из целых чисел (**цел**). Значение таблицы **k** составляют пять целых чисел: 9, 4, 0, -17 и 123.

Элементы таблицы не имеют отдельных имен. Для указания первого элемента используется запись типа k [1]. Например, при выполнении команды

к [3] := к [2] + к [4]

значения второго и четвертого элементов таблицы k складываются (k [2] + k [4]) и полученное значение присваивается третьему элементу этой таблицы.

```
m := 0 | количество положительных элементов
нц для i от 1 до n
| если a[i] > 0
| | то m := m + 1
| все
кц
```

Вышеприведенный алгоритм предназначен для определения количества положительных элементов в таблице a, состоящей из n элементов, а нижеследующий – для нахождения наибольшего значения в таблице.

```
max := a[1]
нц для i от 2 до n
| если a[i] > max
| | то max := a[i]
| все
кц
```

1. Значение переменной **x** равно 3. Чему оно будет равняться после выполнения нижеследующей команды?

   а) `x := 5`
   б) `x := x + 5`
   в) `y := x`

2. Каковы будут результаты следующих команд?

   а) `a[2] := 0`
   б) `b[2] := b[1] + 1`
   в) `c[3] := c[1] + c[2]`

3. Определите значение переменной **s** после выполнения следующих команд.

а)

```
i := 0
s := 0
нц пока i < 3
| i := i + 1
| s := s + i*i
кц
```

б)

```
s := 0
нц для i от 1 до 3
| s := s + i
| нц для j от 2 до 3
| | s := s + j
| кц
кц
```

4. Составьте алгоритм для подсчета суммы всех значений элементов в таблице.

5. Дана таблица **a** из **n** элементов. Напишите алгоритм нахождения номера элемента таблицы, значение которого равно **x**.

5.6. ВВОД И ВЫВОД ИНФОРМАЦИИ

До сих пор при составлении алгоритмов мы предполагали, что компьютер выполняет их автоматически, без вмешательства со стороны человека. Зачастую в процессе выполнения алгоритма возникает необходимость в информационном обмене между компьютером и человеком. Для этой цели в алгоритмическом языке имеются команды ввода информации с клавиатуры в память компьютера и команды вывода информации на экран (из памяти компьютера). Это команды **ввод** и **вывод**.

Так как в алгоритмическом языке для сохранения информации в компьютере пользуются переменными, то в командах **ввод/вывод** указывают имена переменных.

Рассмотрим простой пример.

```
алг произведение
нач цел m, n
    вывод "Введите два числа:"
    ввод m, n
    вывод нс, "Произведение чисел равно", m*n
кон
```

Этот алгоритм будет выполняться в следующей последовательности:

1. Сначала на экране появится надпись "Введите два числа:"
2. По команде **ввод** m, n компьютер начнет ждать ввода с клавиатуры двух чисел.
3. По команде (**вывод нс** "Произведение чисел равно", m*n) компьютер вычислит произведение чисел и выведет его на экран с новой строчки (**нс** – служебное слово).

Команды ввод и вывод. В команде **ввод** перечисляются имена переменных через запятую. При выполнении этой команды компьютер ожидает ввода пользователем (то есть набора на клавиатуре) соответствующих значений. После завершения ввода компьютер присваивает эти значения перечисленным в команде **ввод** переменным. Чтобы ввести в компьютер набранную на клавиатуре информацию, следует нажать клавишу <Enter>.

В команде **вывод** тексты, переменные, выражения отделяются друг от друга запятой. При выполнении команды компьютер подсчитывает значения выражений и отображает результаты на экране. Если информация не умещается на строке, она переносится на следующую строку. Но можно перейти на следующую строку до окончания текущей – для этого используется служебное слово **нс**.

Циклическая команда для. Допустим, требуется составить алгоритм, предполагающий ввод целого числа n, а затем – вывод на экран квадратов первых n натуральных чисел (от 1 до n). Для решения этой задачи можно использовать цикл пока:

```
алг квадраты
нач цел n, i
|  вывод "Введите количество чисел:"
|  ввод n
|  нц пока i <= n
|  |  вывод нс, i**2
|  |  i := i + 1
|  кц
кон
```

При выполнении этого алгоритма команда **вывод нс,** i**2 выполняется сначала для i = 1, затем для i = 2, и т.д. После каждого вывода на экран значение i увеличивается на единицу. В последний раз команда **вывод** выполняется для i = n.

В программировании широко распространено выполнение команд или серии команд для значений в определенном диапазоне (в данном случае от 1 до n). Для этого в алгоритмическом языке предусмотрена циклическая команда **для**. Используя ее, можно записать вышеприведенный алгоритм следующим образом:

```
алг квадраты
нач цел n, i
|  вывод "Введите количество чисел:"
|  ввод n
|  нц для i от 1 до n
|  |  вывод нс, i**2
|  кц
кон
```

Общий вид цикла для таков:

```
нц для i от i1 до i2
|  тело цикла (последовательность команд)
кц
```

Здесь i1 и i2 - целые числа, переменные либо выражения, имеющие целое значение. При выполнении цикла его тело выполняется при значениях i = i1, i = i1+1, i = i1+2, … i = i2. Согласно правилам алгоритмического языка, i1 и i2 могут быть целыми числами, причем i2 может быть и меньше i1 – это не считается

ошибкой, просто в этом случае тело цикла выполняться не будет и выполнение алгоритма продолжится непосредственно с команды, следующей за служебным словом кц.

Алгоритм для подсчета факториала натурального числа n можно записать следующим образом (напомним, что факториалом натурального числа n называется произведение 1 · 2 · 3 · ... · n; факториал обозначается как n! Отметим, что по определению 1!= 1):

```
алг факториал (арг цел n)
  дано | n >= 1
  надо | f = 1 · 2 · 3 · ... · n
нач
  f := 1
  нц для k от 1 до n
  | f := f * k
  кц
  вывод n, '! = ', f
кон
```

1. Какая команда алгоритмического языка используется для ввода данных с клавиатуры в память компьютера?
2. В каком случае в алгоритмах применяется команда **вывод**?
3. Что появится на экране после выполнения команды **вывод нс** “8-й”, “класс”?
4. Напишите алгоритм вывода на экран среднего арифметического двух чисел, введенных с клавиатуры.
5. Что появится на экране после выполнения следующей последовательности команд?

```
переведи в точку (0, 0).
опусти перо
нц для k от 1 до 3
| сдвинь на вектор (k, 0)
| сдвинь на вектор (0, - k)
| сдвинь на вектор (-k-1, 0)
| сдвинь на вектор (0, k+1)
кц
```

6. Напишите алгоритм вычисления суммы $n^2 + (n+1)^2 + \ldots + (2n)^2$

5.7. РЕЗУЛЬТАТЫ АЛГОРИТМОВ И АЛГОРИТМЫ-ФУНКЦИИ

Алгебраические выражения в алгоритмическом языке. В алгоритмическом языке можно использовать не только числа, но и алгебраические выражения (формулы). Правила алгоритмического языка позволяют везде, где можно указать отдельные числа, указывать при необходимости и арифметические выражения.

Правила записи алгебраических выражений в алгоритмическом языке. Алгебраические выражения в алгоритмическом языке записываются в строку, то есть имеют линейную форму. Правила этой записи таковы:

- символы выражения записываются цепочкой - например, вместо "x_1 и v_0" нужно писать "x1, v0"
- для записи операции умножения используется звездочка (*), для записи операции деления - наклонная черта (/), для записи операции возведения в степень – две "звездочки" (**)
- знаки обозначения операций пропускать нельзя – например, произведение чисел 4 и a нельзя записать как "4a" – только как "4*a"
- аргументы функций, как и аргументы вспомогательных алгоритмов, записываются в круглых скобках – например, sqrt (x), max (xy).

Таблица знаков операций и стандартных функций в алгоритмическом языке

Действия, имена функций	Форма записи
Сложение	x + y
Вычитание	x - y
Умножение	x * y
Деление	x / y
Возведение в степень	x ** y
Квадратный корень	sqrt (x)
Абсолютное значение	abs (x)
Знак числа	sign (x)
Минимум чисел x и y	min (x, y)
Максимум чисел x и y	max (x, y)
Остаток от деления целых чисел x на y	mod (x, y)
Частное от деления целых чисел x на y	div (x, y)
Целая часть вещественного числа x	int (x)

Виды переменных в алгоритмическом языке. В алгоритмах используется несколько видов величин. В алгоритме "факториал" переменная n, именуемая

аргументом, служит для передачи информации от основного алгоритма к вспомогательному. Но иногда возникает необходимость возвращения (передачи) информации от вспомогательного алгоритма к основному, и для этого в алгоритмическом языке предусмотрены *результатные величины* (результаты).

Рассмотрим простой алгоритм.

```
алг гипотенуза (арг вещ a, b, рез вещ c)
  дано | длина катетов прямоугольного треугольника
       | a ≥ 0 и b ≥ 0
  надо | найти c - длину гипотенузы этого треугольника
нач
| c := sqrt(a**2 + b**2)
кон
```

Здесь запись “**рез вещ** c” означает, что результатом (**рез**) выполнения данного алгоритма является действительная (вещественная) (**вещ**) переменная величина c.

Выполнение алгоритма с результатами. При вызове вспомогательного алгоритма в качестве его результатов указываются имена величин, значения которых вычисляются в основном алгоритме. Например, для того чтобы найти при помощи основного алгоритма под названием “вычисление” гипотенузу x прямоугольного треугольника с катетами p – 1 и p + 1, достаточно вызвать вспомогательный алгоритм “гипотенуза”.

```
алг вычисление
нач вещ p, x
|   ...
|   гипотенуза (p-1, p+1, x)
|   ...
кон
```

Рассмотрим, как выполняется данный алгоритм. Допустим, что на момент вызова вспомогательного алгоритма значение переменной p равно 7.

При вызове

“гипотенуза (p-1, p+1, x)”

компьютер присваивает значения, соответствующие

p - 1 = 6 и p + 1 = 8

аргументам a и b вспомогательного алгоритма “гипотенуза”. Затем выполняется сам алгоритм “гипотенуза” и подсчитывается значение переменной c:

c = sqrt (6**2 + 8**2) = 10

По достижении строчки **кон** компьютер присваивает значение результата с переменной x основного алгоритма “вычисление”:

x = 10

После этого компьютер заканчивает выполнение вспомогательного алгоритма “гипотенуза” и продолжает выполнение основного алгоритма “вычисление”.

Общие правила выполнения команды вызова вспомогательного алгоритма

1. Перед началом выполнения вспомогательного алгоритма в памяти компьютера отводится место для него и значения аргументов, указанные в команде вызова основного алгоритма, присваиваются аргументам вспомогательного алгоритма.
2. На время выполнения вспомогательного алгоритма выполнение основного алгоритма приостанавливается.
3. После выполнения вспомогательного алгоритма значения его результатов присваиваются величинам, указанным в команде вызова основного алгоритма. После выполнения вспомогательный алгоритм полностью стирается из памяти компьютера (если он будет вызван снова, все повторится заново: компьютер отведет для него место в памяти, присвоит значения аргументам, и т.д.).

Решение квадратного уравнения.

```
алг квадра (арг вещ a, b, c, рез цел n, рез вещ x1, x2)
  дано | a ≠ 0; a, b, c - коэффициенты квадратного уравнения
  надо | найти n - количество различных корней, корни x1,
       | x2 (если есть)
нач вещ d
|   d := b**2 - 4*a*c | дискриминант
|   если d < 0
|     то n := 0
|     иначе
|     если d = 0
|       то n := 1; x1 := -b/(2*a); x2 := x1
|       иначе
|       n := 2
|       x1 := (-b-sqrt(d))/(2*a)
|       x2 := (-b+sqrt(d))/(2*a)
|     все
|   все
кон
```

Алгоритмы-функции. Если в записи выражения необходимо указать какую-либо функцию (см. таблицу функций выше), то необходимо записать название функции, а затем в скобках – ее аргумент. Например, квадратный корень из 13 записывается как sqrt (13).

В алгоритмическом языке можно создавать новые функции и использовать их в дальнейшем, как стандартные. Для этого нужно задать *алгоритмы вычисления этих функций*, или *алгоритмы-функции*.

Алгоритм вычисления функции пишется по обычным правилам и отличается от обычного алгоритма только формой записи заголовка алгоритма и особым служебным словом **знач**.

Допустим, в основном алгоритме требуется использовать функцию $s(t)=\frac{t^2}{2}$

Алгоритм для вычисления этой функции записывается следующим образом:

```
алг вещ s (арг вещ t)
нач
    знач := t**2 / 2
кон
```

Служебное слово **вещ** перед "s" указывает на то, что значение функции s - действительное (вещественное) число. Служебное слово **знач** используется в алгоритме для указания значения функции. После выполнения алгоритма значение этой величины становится значением функции.

Выполнение алгоритма-функции. Проследим выполнение алгоритма-функции на примере команды

y := s(0.3)*1000

Запись в правой части этой команды s(0.3) указывает на то, что нужно вызвать алгоритм-функцию s и вычислить его значение для аргумента t = 0.3. При появлении в алгоритме алгоритма-функции компьютер отводит в памяти место для него и запоминает значение аргумента.

Тело алгоритма s состоит из одной команды:

знач := t**2 / 2

В соответствии с правилами алгоритмического языка компьютер заменяет переменную t в правой части на ее значение 0.3, вычисляет выражение

0.3**2/2 = 0.09/2 = 0.045

и присваивает полученное значение величине **знач**.

На этом выполнение алгоритма-функции s заканчивается. Компьютер подставляет на место вызова s(0.3) в формуле s(0.3)*1000 значение 0.045, полученное

для величины **знач**, находящейся в последней строчке, и уничтожает в памяти место, отведенное для алгоритма-функции s.

Затем вычисляется 0.045*1000 = 45 и это значение присваивается y. На этом вычисление “y := s (0.3)*1000” заканчивается.

1. Каковы правила записи алгебраических выражений в алгоритмическом языке?
2. Изложите правила выполнения команды вызова вспомогательного алгоритма.
3. Чем отличается алгоритм-функция от обычного алгоритма?
4. Составьте следующие алгоритмы-функции:

   а) найти наибольшее из трех чисел – a,b и c;
   б) вычислить количество различных размещений трех чисел – a, b и c;
   в) найти последнюю цифру целого числа n > 0;
   г) найти первую цифру целого числа n > 0.

6

КОММУНИКАЦИОННЫЕ ТЕХНОЛОГИИ

6.1. КОМПЬЮТЕРНЫЕ СЕТИ

С созданием электронно-вычислительных машин (ЭВМ) возникла необходимость создания взаимосвязей между ними с целью увеличения скорости обработки информации и экономии программных и аппаратных ресурсов. Созданные для этих целей первые немногочисленные сети соединяли крупные ЭВМ с большими вычислительными центрами. Настоящий “бум” на такие сети возник только после появления персональных компьютеров, так как те очень быстро получили широкое распространение. Компьютеры стали соединять локальными сетями, которые, в свою очередь, соединялись друг с другом, а также с глобальной сетью. За последние 15-20 лет к сети присоединились сотни миллионов компьютеров, в результате чего более миллиарда пользователей получили возможность связываться друг с другом.

Сеть. Ресурсы

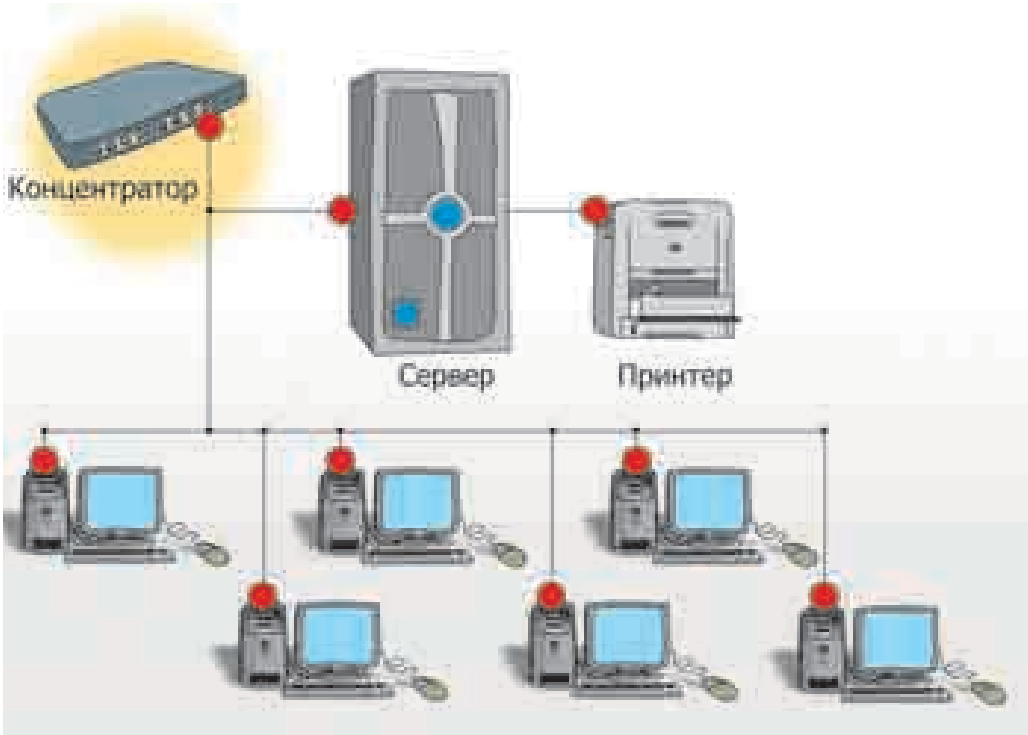
Сеть (Network) – это группа компьютеров или других устройств, объединенных тем или иным способом с целью обмена информацией и совместного использования ресурсов.

Ресурсы – это совместно используемые в сети программы, файлы, а также принтеры и другие периферийные устройства.

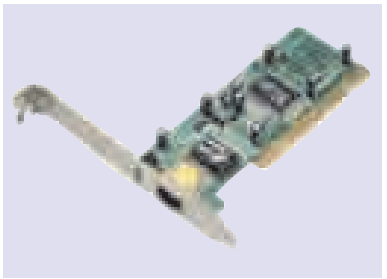
Существует два основных вида компьютерных сетей:

1) локальная вычислительная сеть (local area network – LAN);
2) глобальная вычислительная сеть (wide area network – WAN).

LAN объединяет компьютеры в пределах небольшой территории (например, одного здания). Такие сети удобны тем, что они делают доступными для каждого пользователя ресурсы других компьютеров данной сети во время совместной работы. Если бы не было локальных сетей, люди вынуждены были бы постоянно обмениваться дискетами или часто использовать принтер.



Общий вид компьютерной сети



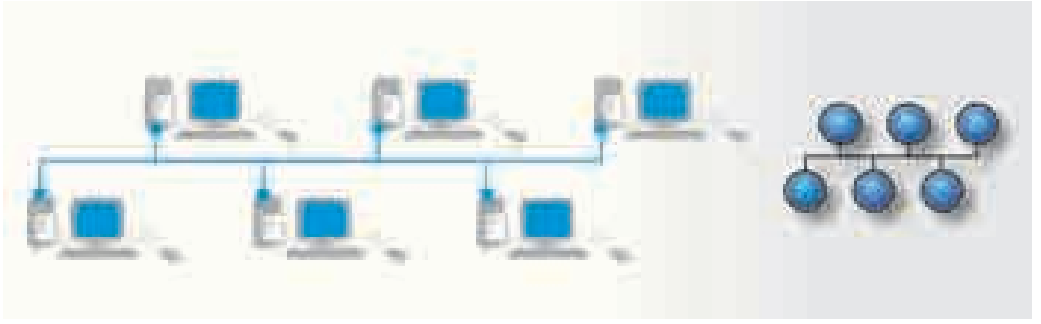
Разумеется, для объединения компьютеров в сеть требуется дополнительное оборудование. Важнейшая составляющая аппаратного обеспечения компьютерной сети – *сетевой адаптер*, или NIC (Network Information Card). Его иногда называют *интернет-адаптером* или *сетевой картой*. Он может быть и отдельной картой, и элементом материнской платы компьютера.

Компьютеры можно соединять друг с другом для обмена информацией между ними и для совместного использования устройств, подключенных к какому-либо из них. Соединенные таким образом два (и более) компьютера образуют компьютерную сеть. Их может объединять как обычная телефонная линия, так и сложная спутниковая связь.

Для подсоединения оборудования сети (компьютеров и других устройств) используется *сетевой кабель*, посредством которого компьютеры и центральное устройство обмениваются сигналами.

При создании компьютерной сети огромное значение имеет топология, то есть схема расположения сетевых устройств и прокладываемого кабеля. Выбирать топологию следует таким образом, чтобы сеть была надежной и рациональной, чтобы было удобно управлять потоком данных в ней. Существует три основных типа топологии сетей. Рассмотрим их подробнее.

“Шина” (Bus). Такой тип топологии предполагает, что все компьютеры сети соединены одним кабелем.



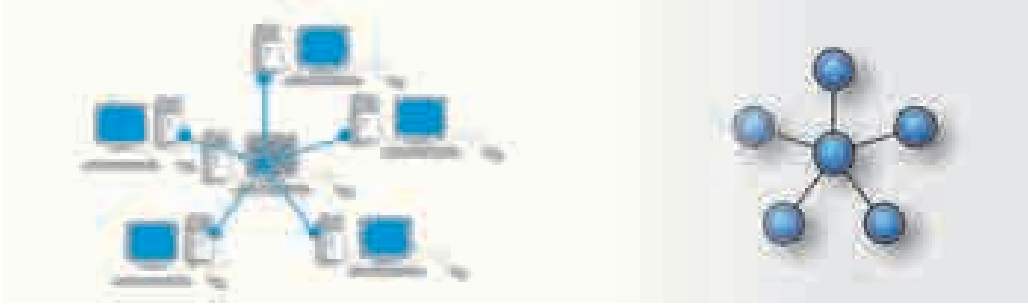
В подобной сети данные рассылаются одновременно всем компьютерам. Каждый компьютер проверяет поступающую информации, и если она адресована ему, то принимает ее и обрабатывает. Топологию “шина” иногда называют также линейной (line).

Это очень простой и наиболее экономичный тип топологии, так как он требует наименьшего расхода кабеля. Вместе с тем у него имеется ряд недостатков. Если какая-то часть такой сети выходит из строя, то перестает работать вся сеть. По этой причине топология “шина”, которая была очень популярна около десяти лет назад, ныне при установке сетей практически не используется.

“Кольцо” (Ring). При такой топологии каждый компьютер так соединяется с двумя соседними, чтобы иметь возможность получать информацию от одного из них и передавать ее другому. Последний компьютер соединяется с первым – и кольцо замыкается.

У этого типа топологии, как и у предыдущего, недостатки также перевешивают достоинства. Например, в подобных сетях сложно обнаруживать неисправности. Кроме того, выход из строя одного компьютера оказывает влияние на соединенные с ним компьютеры и далее – на всю сеть. Поэтому бывшие одно время очень популярными сети типа “кольцо” сейчас встречаются очень редко.

“Звезда” (Star). Этот тип топологии, который появился на заре развития вычислительной техники, предполагает, что компьютер каждого пользователя сети подключен к наиболее мощному центральному компьютеру.



При такой конфигурации весь поток данных идет через центральный компьютер, который отвечает за управление обменом информацией между всеми элементами сети. У сетей с таким типом топологии тоже есть недостатки. Так, центральный

компьютер несет такую огромную нагрузку, что уже не может выполнять почти никакую другую работу, кроме обслуживания сети, а в случае выхода его из строя парализуется вся сеть. Сети такого типа сейчас также встречаются крайне редко.

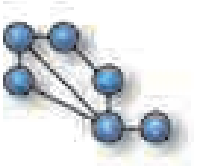
Теперь ознакомимся подробнее с принципами связи между компьютерами в сети.

Для общения друг с другом люди часто используют устную речь. Но такая форма общения удобна только в том случае, если собеседники находятся рядом друг с другом. Представьте теперь, что вам нужно послать письмо другу, живущему в другом городе, а то и в другой стране. Тут уже не обойтись без целого ряда действий: придется изложить информацию на бумаге, подписаться, вложить лист в конверт, надписать на конверте адреса отправителя и получателя, наклеить марку и, наконец, отдать письмо почтальону (или бросить в почтовый ящик). Дальнейшая судьба письма зависит уже не от вас, а от почтовой службы. Поездом, кораблем, самолетом или другим видом транспорта оно будет доставлено в ту страну, в тот город, где живет ваш друг. Там оно поступит в почтовое отделение, а оттуда – в почтовый ящик адресата. И лишь тогда получатель, то есть ваш друг, сможет открыть конверт и прочитать вашу информацию. Причем при возникновении проблем на том или ином этапе доставки (например, при отсутствии почтальона или в связи с различием правил написания адреса в разных странах) письмо может и вовсе не добраться до адресата.

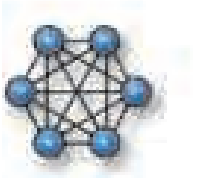
- В наши дни наибольшее распространение получила гибридная топология **«Звезда - шина» (Star Bus)**.
- Существуют и другие типологии, основанные на сочетании основных типологий:

«дерево» (Tree)

«петля» (Mesh)



«полное соединение» (Fully Connected)

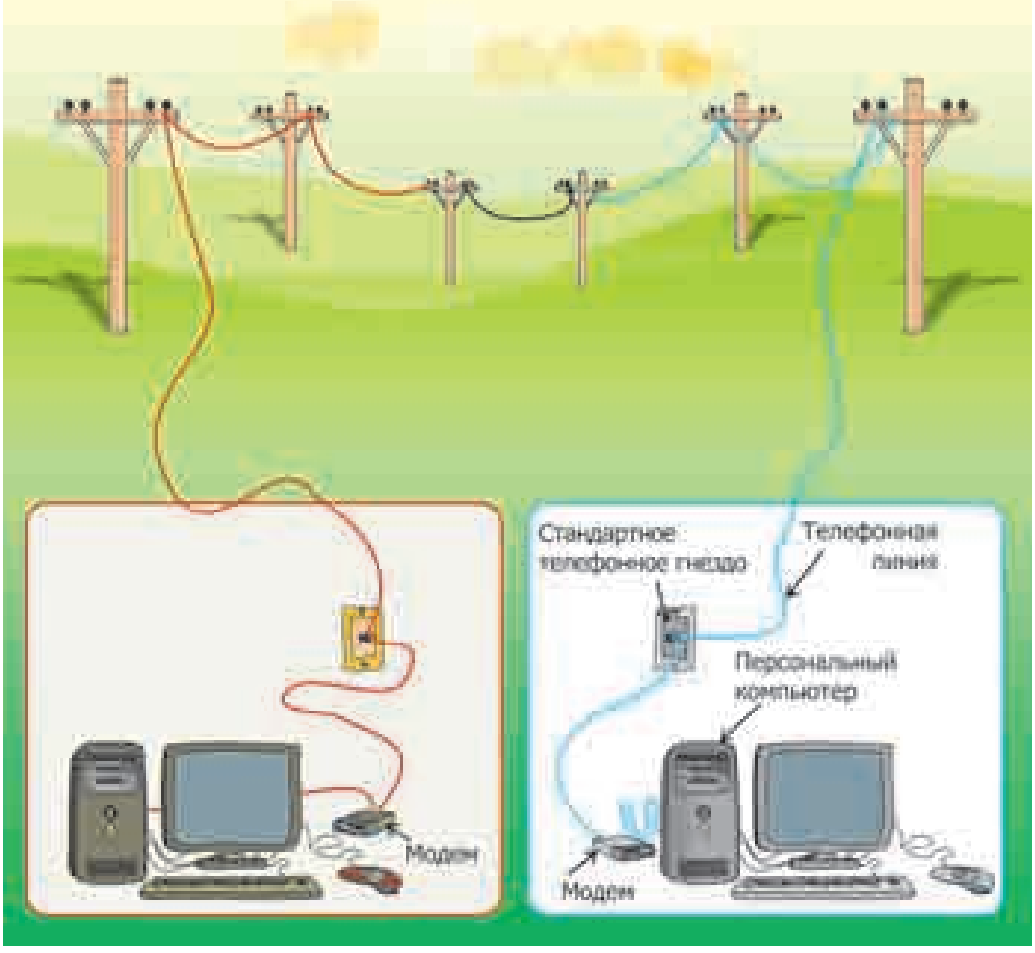


Компьютеры в сети не могут непосредственно общаться между собой - они пока еще не научились “говорить” напрямую друг с другом, поэтому для общения им необходимо выполнение целого ряда последовательных процедур, которые именуются сетевым протоколом.

Протокол

Протокол – это набор правил и процедур, упорядочивающих взаимные связи компьютеров в сети.

При соединении компьютеров друг с другом при помощи телефонной линии используется устройство, которое называется модем. В телефонной системе, в отличие от компьютера, данные имеют не цифровую, а аналоговую форму. Модем передающего компьютера преобразует пересылаемые с помощью телефонной линии данные в аналоговую форму, а модем принимающего компьютера, приняв аналоговый сигнал, преобразует его обратно в цифровую форму и передает в компьютер.



1. Что такое компьютерная сеть?
2. Какой аппарат требуется для создания компьютерной сети?
3. Назовите виды компьютерных сетей.
4. Что такое сетевой протокол?
5. Что такое топология сетей и каковы ее основные типы?

6.2. ИНТЕРНЕТ

Сейчас, наверняка, не найдется человека, который не слышал бы об Интернете. Вы, возможно, пользуетесь им для какой-либо работы или в целях развлечения. В последнее время Интернет развивается огромными темпами. В наши дни он, можно сказать, целиком изменил компьютерный мир. Возможно, в обозримом будущем он коренным образом изменит и весь наш привычный жизненный уклад.

Что такое Интернет и как он функционирует?

Интернет – самая крупная компьютерная сеть в мире.

В разговорной речи Интернет часто именуют просто Net (Сеть) и пишут это слово с прописной буквы. Однако Интернет – это не просто сеть. Это сеть сетей, позволяющая людям свободно обмениваться информацией друг с другом.

Согласно последним исследованиям Консорциума систем Интернет (Internet Systems Consortium) число компьютеров, подключенных к Интернету, на январь 2008 года составило 541 677 360.

В наши дни люди, говоря об Интернете, как правило, имеют в виду выполнение с его помощью какой-либо работы, поиск в нем чего-либо, возможность встретиться благодаря ему с кем-либо … Миллионы подключенных к Интернету компьютеров ведут информационный обмен друг с другом. Возможности Интернета настолько обширны, что для получения полного представления о всех его службах не хватило бы не только нескольких уроков, но и нескольких книг. Так что ограничимся их кратким перечислением.

Электронная почта (e-mail). На данный момент это наиболее распространенная служба связи. С ее помощью можно общаться с человеком вне зависимости от того, в какой точке земного шара он находится. Электронная почта часто заменяет людям традиционную почту, а также телефон или факс.

World Wide Web (WWW). Сегодня, говоря об Интернете, имеют в виду в первую очередь именно web, хотя он – лишь одна из служб Интернета. В отличие от первых появившихся служб Интернета, **web** объединил в себе такие элементы, как тексты, изображения, видеоклипы, анимацию, а также возможность распространения в эфире новостей и музыки. Число новых веб-сайтов стремительно растет.

Групповые дискуссии. С целью общения друг с другом на самые различные темы люди, находящиеся в разных точках земного шара, используют чаты **[chat room]**. Беседа в чате может идти как между небольшими (или

большими) коллективами, так и между двумя людьми. Широко используются соответствующие службы (чаты) Yahoo! и Microsoft, созданные на основе технологии Internet Relay Chat, а также www.

Обмен мгновенными сообщениями. Если на вашем компьютере и на компьютерах ваших друзей установлены соответствующие программы, вы можете обмениваться мгновенными информационными сообщениями. Такие программы, как Windows Messenger, Yahoo Messenger и ICQ, позволяют мгновенно выводить отправленное сообщение на экран получателя (адресата).

Чтобы получить доступ в Интернет, необходимо следующее:

1) компьютер с подсоединенным к нему модемом;
2) соответствующее программное обеспечение;
3) финансовые возможности для оплаты;
4) определение провайдера – поставщика, обеспечивающего услуги Интернета.

Лучше всего подключиться к Интернету, оплатив соответствующие услуги провайдера (от англ. provide – "обеспечивать"), так как он обеспечивает непосредственный выход в сеть, круглосуточную техническую поддержку и (при необходимости) обучение основам пользования Интернетом.

Провайдера сокращенно называют также ISP (Internet Service Provider – *"обеспечивающий услуги Интернета"*) – произносится это как "ай-эс-пи". Перечислив деньги провайдеру, пользователь может приобрести следующие услуги:

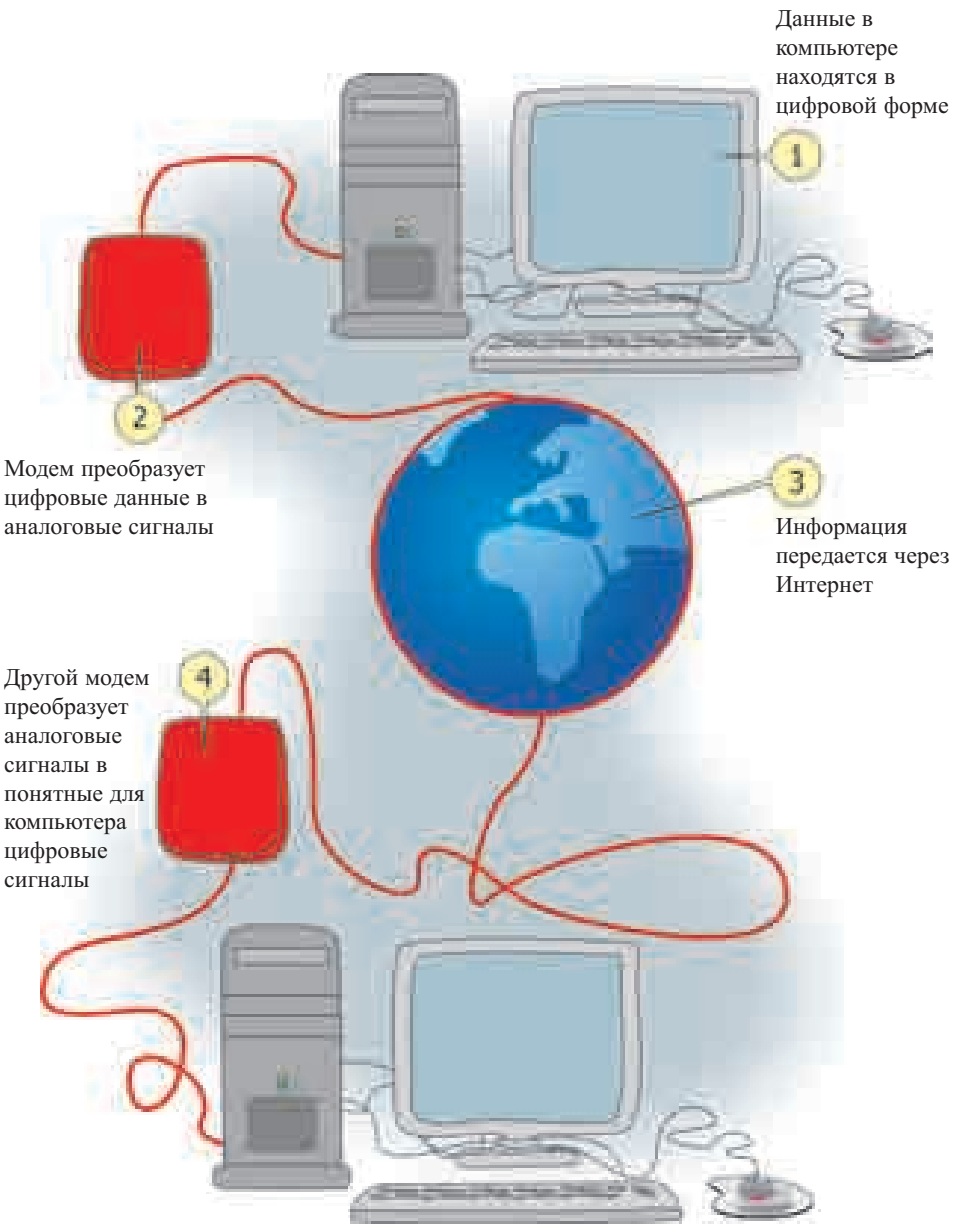
– выход в Интернет;
– электронную почту;
– поддержку.

Поскольку оборудование и программное обеспечение у всех провайдеров сходное, то их возможности и предоставляемые услуги мало чем различаются – отличие может наблюдаться только в расценках, ассортименте услуг и их качестве.

Простейший способ подключения к Интернету – при помощи телефонной линии.

Посредством телефонной линии информацию можно пересылать двумя способами, в зависимости от типа сигнала – аналогового (в виде звуковых волн) или цифрового (в виде электрических частот).

Поскольку данные в компьютере находятся в цифровой форме, то по аналоговой телефонной линии они пересылаются при помощи устройства, именуемого **модем** (**мо**дулятор – **дем**одулятор). Модем преобразует цифровые данные в аналоговые сигналы, и наоборот, аналоговые сигналы – в цифровые данные. На рисунке показано, каким образом два компьютера могут обмениваться информацией через Интернет.



В стационарных компьютерах используются модемы двух типов: внутренние и внешние. Внутренний модем располагается внутри системного блока компьютера, а внешний подключается к одному из портов системного блока при помощи кабеля.

Как работает модем?

Ваш модем подает сигнал, который «заявляет» о вашем компьютере тому компьютеру, с которым намеревается создать соединение, и тот компьютер подтверждает ваши полномочия для использования сети. Внешне этот процесс сопровождается гудением и писком модема.

Модемы различаются также по скорости приема-передачи данных. Скорость модема измеряется в битах в секунду (bps).

Для подключения к Интернету при помощи обычного модема (соединение dial-up) следует подключить его к телефонной линии. При таком способе выхода в Интернет на протяжении всего сеанса соединения телефонная линия будет занята.

Высокоскоростной Интернет. Однако пользователей Интернет уже не может устраивать медленная и не очень надежная связь, описанная выше. В школы, в ВУЗы, в учереждения и к нам домой уже приходит высокоскоростной и надежный Интернет. Его нам предоставляют новые технологии. Упомянем некоторые из них:

ADSL (англ. *Asymmetric Digital Subscriber Line* – ассиметричная цифровая абонентская линия) – модемная технология, превращающая стандартные телефонные линии в линии высокоскоростного доступа. По одной и той же телефонной линии вы можете выходить в Интернет и одновременно разговаривать по телефону.

GPRS (англ. *General Packet Radio Service* – пакетная радиосвязь общего пользования) – надстройка над технологией мобильной связи GSM, осуществляющая пакетную передачу данных. GPRS позволяет пользователю сети сотовой связи производить обмен данными с другими устройствами в сети GSM и с внешними сетями, в том числе Интернет.

Оптоволоконная связь – средство связи на больших расстояниях, построенное на основе волоконно-оптических линий связи. Скорость передачи данных может измеряться сотнями гигабитов в секунду.

Беспроводной широкополосный Интернет. Не надо никаких линий. Установил дома специальный модем, договорился с провайдером – и выходи в Интернет.

Час пик

Когда одновременно с вами в Сети находится огромное количество людей, скорость загрузки веб-страниц снижается. Интернет в этом смысле похож на дорожную трассу – когда на ней слишком много автомобилей, движение замедляется. Так и в Интернете: чем больше в нем на данный момент отправителей и получателей информации, тем ниже скорость ее передачи.

Для приема или передачи большого объема данных лучше выходить в Сеть поздней ночью или ранним утром, когда там малолюдно.

1. Что такое Интернет?
2. Какие службы Интернета вы знаете?
3. Что необходимо для подключения к Интернету?
4. Какие функции выполняет модем?

6.3. WWW

Именно *World Wide Web* (сокращенно – Web или Veb) сделал Интернет популярным. Изначально в Сети имелись только тексты, но с появление Web этот “скучный” этап ее развития завершился: в Интернете появилась графика и другие мультимедийные элементы.

Интернет превратился во Всемирную “паутину” миллионов Web-серверов, рассеянных по всему миру.

Web-сервер – это программа, которая использует протокол HTTP (HyperText Transfer Protocol) для передачи данных с подключенного к Сети компьютера.

Web-браузер. Гипертекст

Для просмотра на пользовательском компьютере полученной с web-сервера информации используется специальная программа – web-браузер. Основная функция web-браузера отображать гипертекст, потому что большинство информации в Web представляет собой именно гипертекст. Гипертекст – это текст, с которого можно перейти на другой текст или объект.

Для облегчения создания, хранения и размещения во Всемирной паутине гипертекста традиционно применяется язык

HTML [HyperText Markup Language].

Гипертекст, подготовленный на языке HTML, записывается в файл. HTML-файлы – наиболее распространенные Интернет-ресурсы. HTML-файл, размещенный на web-сервере, называется web-страницей. Совокупность взаимосвязанных web-страниц называется web-сайтом.

Web-сайт

Web-сайт (web-site) – это совокупность web-страниц, тесно связанных между собой по тематике и (или) принадлежащих какой-либо компании, группе или отдельному человеку.

Каждая web-страница имеет свой *URL-адрес*. Он определяет официальное название того или иного Интернет-ресурса и состоит из двух частей. Первая – это идентификатор ресурса, вторая указывает *местонахождение ресурса*. Типичный URL-адрес выглядит так:

http://www.azerbaijan.az/_Sosiety/_Education/_education_a.html.

Первая часть адреса –

http://

– идентификатор ресурса, или просто идентификатор. Он информирует браузер о том, какой протокол или язык использовался при создании данного ресурса.

Вторая часть URL-адреса отражает местонахождение ресурса (в данном случае – web-страницы). Для того чтобы лучше понять ее смысл, проще разделить ее, в свою очередь, еще на две части:

– первая часть (www.azerbaijan.az) – это доменное имя компьютера, то есть имя главного компьютера (все компьютеры в Сети имеют собственные имена, позволяющие отличать один от другого);

– вторая часть (/_Sosiety/_Education/_education_a.html) – это название отдельно взятого web-ресурса.

Для того чтобы путешествовать по Всемирной паутине, необходимо особое программное обеспечение, а точнее – web-браузер, или просто браузер (от англ. to browse – "просматривать"). Наиболее известный из браузеров – **Microsoft Internet Explorer** (IE). Помимо него, имеются такие браузеры, как Netscape Navigator, Mozilla, Opera, Safari.

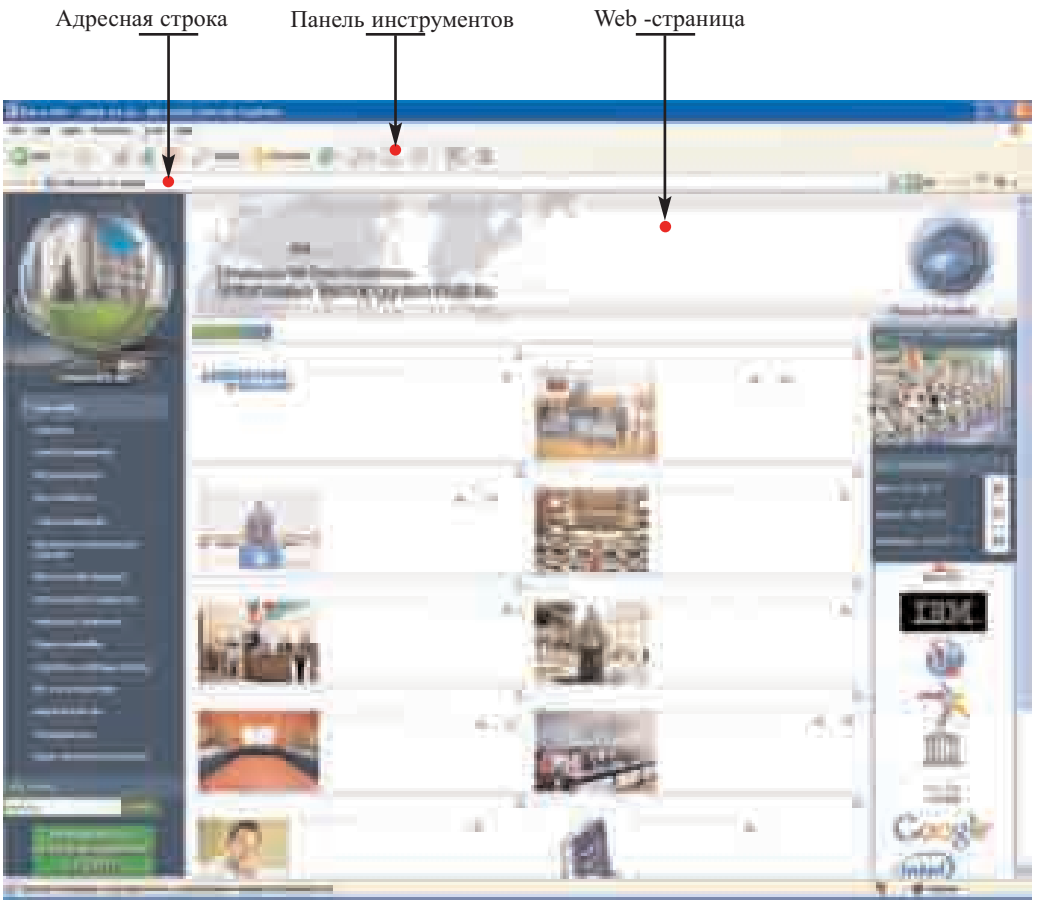
Чтобы начать работать с Internet Explorer, дважды щелкните на его иконке на Рабочем столе. На экране откроется окно

Internet Explorer.

Адреса просмотренных страниц сохраняются в

Адресной строке (Adress bar).

Туда же вводится и адрес страницы, которую требуется открыть.



Окно браузера Internet Explorer

Тексты и иллюстрации на **web**-странице располагаются примерно так же, как на обычной странице – например, журнальной. На некоторых web-страницах имеются “живые” изображения, а некоторые страницы открываются под музыкальное сопровождение (оно порой раздражает).

Первый web -сайт

Первый в мире web-сайт был представлен общественности 6 августа 1991 года. Его создатель **Тим Бернерс-Ли** опубликовал на нем новую технологию **WWW**, основанную на протоколе передаче данных **HTTP**, системе **URL**-адресов и языке разметки гипертекста **HTML**. На сайте также были приведены принципы действия и правила установки web - серверов и web-браузеров. Впоследствии Тим Бернерс-Ли разместил на этом сайте список ссылок на другие сайты, так что этот сайт стал к тому же первым в мире Интернет-каталогом.

Как найти информацию в Сети?

Для этой цели служат *поисковые системы* (*Search Service*). Поисковые системы – это особые web-страницы, состоящие из гигантских каталогов web-сайтов, в которых можно найти что угодно. Одна из них (лучшая, но далеко не единственная) расположена по адресу www.google.com (или www.google.az).

Поисковая система	Адрес
Ask Jeeves	www.ask.com
Rambler	www.rambler.ru
Yahoo!	www.yahoo.com
Aport	www.aport.ru

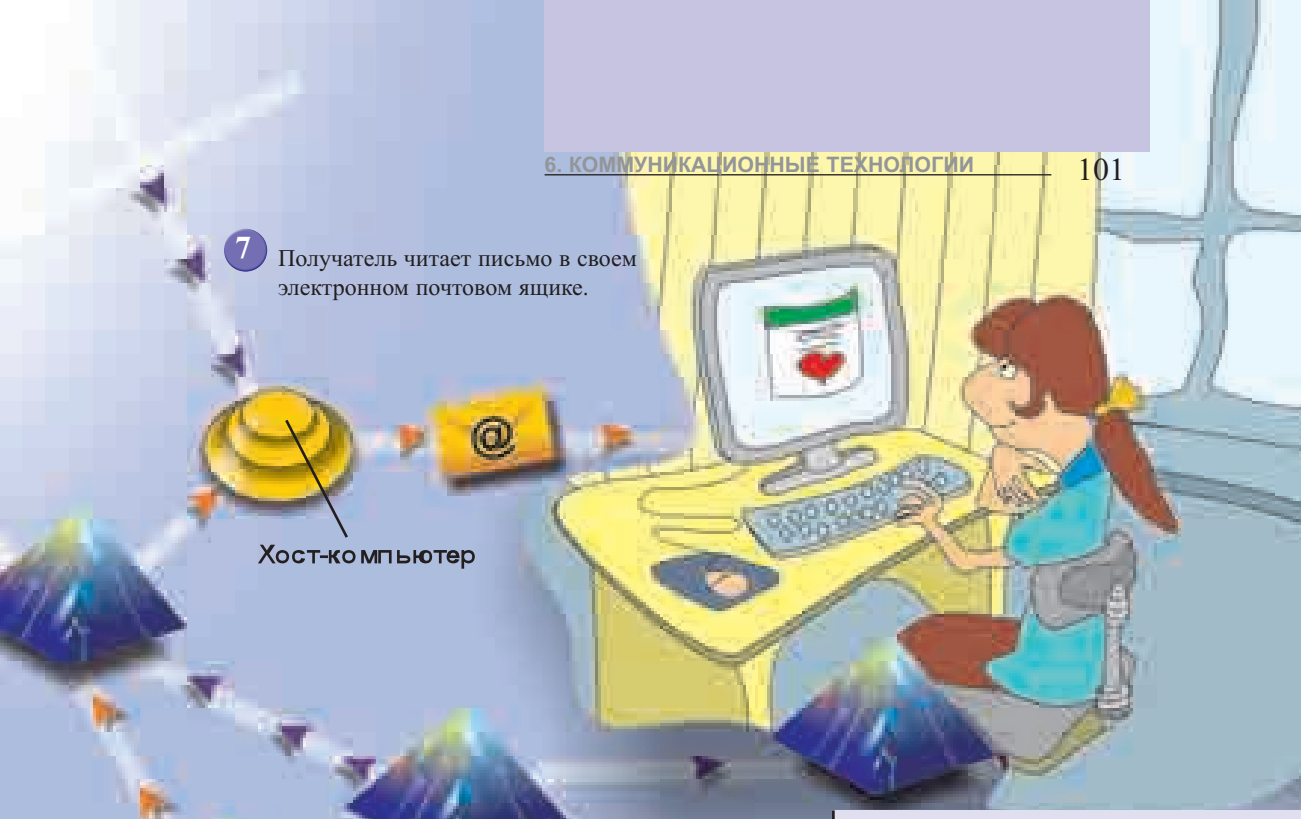
1. Что такое web-сервер и web-браузер?
2. Для чего используется web-браузер?
3. Что такое web-сайт?
4. Что такое URL-адрес и из каких элементов он состоит?

6.4. ЭЛЕКТРОННАЯ ПОЧТА

Электронная почта — широко распространенная и самая популярная служба Интернета. Принцип работы электронной почты интуитивно понятен тем, кто пользовался обычной почтой. В сети имеется *хост-компьютер*, который сортирует письма, рассылает их по почтовым отделениям и принимает корреспонденцию из этих отделений. Его можно сравнить с центральным почтамтом.

Каждый абонент почтового отделения имеет собственный почтовый ящик, куда отправляется вся адресованная ему корреспонденция. Роль ключа от почтового ящика выполняет сетевой пароль.

Основные преимущества электронной почты – ее оперативность и высокое качество связи. Скорость доставки, круглосуточный режим работы, возможность массовой рассылки одной и той же информации, сохранение полученной корреспонденции на хост-компьютере на протяжении какого угодно периода времени по желанию пользователя, и многое другое отличает электронную почту от обычной.



7 Получатель читает письмо в своем электронном почтовом ящике.

6 Нужный маршрутизатор перенаправляет почту на хост-компьютер, к которому подсоединяется компьютер получателя, и помещает письмо в электронный почтовый ящик.

5 Информация передается от одного маршрутизатора к другому, пока не достигнет цели.

4 Хост-компьютер отправляет электронную почту одному из маршрутизаторов (router), а тот – на следующий.

- Маршрутизаторы связаны между собой телефонными линиями и кабелями. Прежде чем дойти до адресата, сообщение электронной почты может обойти полсвета.
- Если маршрутизатор перегружен или неисправен, почта передается посредством другого, нормально работающего маршрутизатора.

Для отправки и получения писем посредством этой службы нужен электронный почтовый адрес. Этот адрес, присваиваемый каждому пользователю, уникален (то есть аналогичного не может быть ни у кого другого) и выглядит примерно так:

comebody@comewhere.com

Первая его часть (“somebody”) — это имя пользователя. Оно может быть выбрано вами лично, а может быть предложено вам провайдером. Символ “@“ (а с хвостиком) отделяет имя пользователя от доменного имени.

Доменное имя (“somewhere.com”) – это имя центрального компьютера вашего провайдера. Сочетание имени пользователя и имени почтового сервера провайдера представляет собой точный адрес для отправки корреспонденции.

Люди веками искали, а затем использовали различные пути обмена информацией, но ни один из этих способов не сравнится по скорости и надежности с электронной почтой.

В древности одним из способов передачи наиболее важной информации был дым сигнальных костров. В 1588 году весть о приближении испанского флота к берегам Англии была передана по “огненной цепочке” с одного конца страны на другой всего за 18 минут! Однако в пасмурную, сырую погоду костер как средство связи ненадежен.

Письма доставляли пешие или конные гонцы. Широко известен случай, когда в 490 г. до н.э. после Марафонской битвы воин по имени Фидиппид пробежал, не останавливаясь, дистанцию в 40 км, чтобы донести до Афин весть о победе греков. Произнеся слова “Ликуйте! Мы победили”, он замертво рухнул на землю.

В Древней Греции сообщения отправляли с почтовыми глубями, привязывая записку к лапке птицы. Именно так, например, рассылались из Афин известия об итогах Олимпийских игр. Сообщение доходило за два-три часа (если голубь не погибал и не сбивался с курса).

В I веке н. э. в Китае, Иране и Римской империи появились почтовые службы. Сообщения, записанные на свитках, переправляли конным путем или водным транспортом. Такая доставка на большие расстояния занимала порой целые недели. Почтовая служба работала только для государственных должностных лиц. Частные лица не имели права ею пользоваться, но большинству из них это было и не нужно по причине их неграмотности.

Первая общественная почтовая служба была создана в 1464 году французским королем Людовиком XI. Письма доставляли конные гонцы. Позднее для защиты от разбойников стали использовать охраняемые почтовые кареты. К XVII веку в Англии похищения корреспонденции достигли такого размаха, что люди вынуждены были для подстраховки разрезать письма надвое и каждую половинку посылать отдельно.

В XIX веке доставка почты стала вестись по железной дороге, и стоимость почтовых услуг сильно снизилась. В 1918 году открылся первый почтовый авиарейс Вашингтон – Нью-Йорк. Письма и посылки доставлялись из одного города в другой за два дня.

В 1837 году был изобретен телеграф, позволяющий быстро передавать информацию на огромные расстояния при помощи азбуки Морзе.

В 1866 году по дну Атлантического океана был проложен кабель, соединивший Европу с Америкой. Письма, отправленные этим же путем пароходами, доходили до адресатов за 11 дней, телеграфные же сообщения - за считанные минуты.

Первое сообщение электронной почты было отправлено в 1971 году, с 1980 года начался обмен электронной информацией между учреждениями и университетами, а с 1990 года электронная почта распространилась по всему миру.

1. Чем отличается обычная почта от электронной?
2. Что должен сделать пользователь, чтобы отправить письмо по Интернету?
3. Создайте свой электронный почтовый ящик (если имеется такая возможность) и отправьте краткое письмо другу.

6.5. ПРОГРАММА OUTLOOK EXPRESS

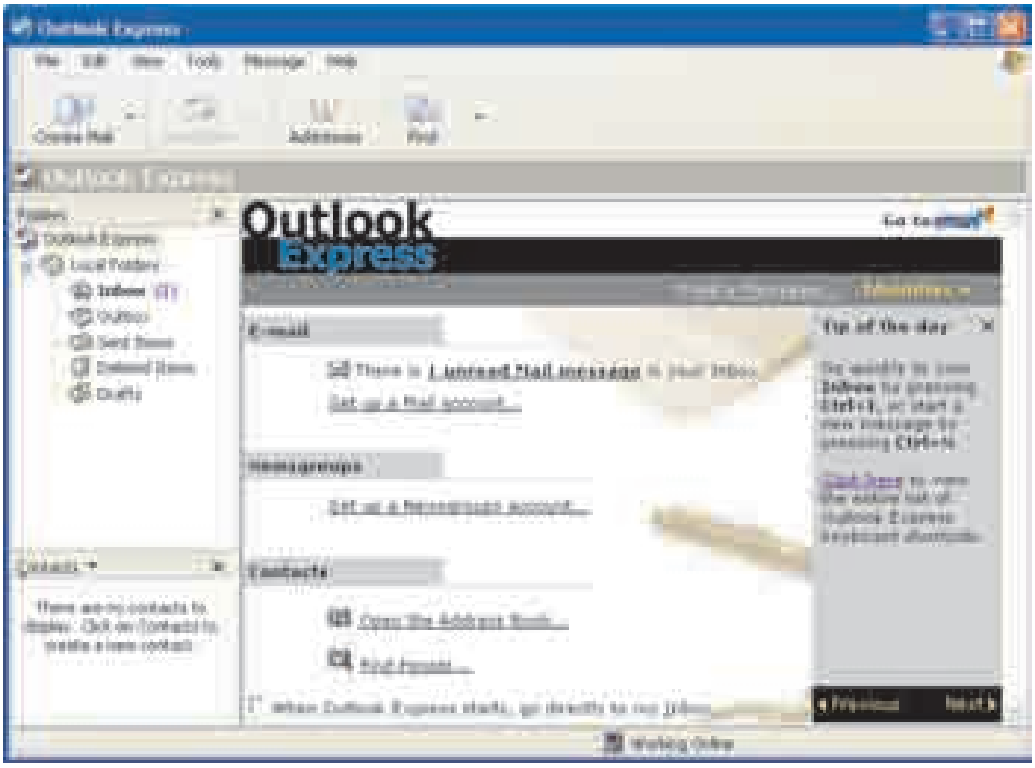
Помимо web-браузера в стандартном пакете Windows XP есть почтовая программа – Outlook Express.

Чтобы запустить программу Outlook Express, надо произвести двойной щелчок мышью по ее значку на Рабочем столе.

Если соединение с Интернетом на данный момент отсутствует, программа Outlook Express после запуска сама наладит подключение к нему.

При запуске программа Outlook Express в первую очередь проверяет наличие новых сообщений. Ее окно состоит из трех панелей:

1) **Список папок;** 2) **Контакты;** 3) **Задания**.



Окно Outlook Express

Папки (Folders). В левом верхнем углу окна программы располагается список папок отправленных, принятых, удаленных и других сообщений.

Контакты (Contacts). В левом нижнем углу программы располагается список контактов – адресов людей, с которыми вы переписываетесь.

Сведения о сообщениях. Справа располагается "домашняя страница", отражающая сообщения, группы новостей и пр. Outlook можно настроить таким образом, чтобы при запуске сразу открывать папку Входящие (Inbox). Для этого нужно выбрать команды

Сервис (Tools) ⇨ Свойства (Options)

и в открывшемся диалоговом окне выбрать команду:

При запуске сразу открывать папку Входящие
(When starting go directly to my 'Inbox-folder)

Главное окно Outlook Express само по себе малоинформативно. Чтобы просмотреть недавно полученные, а также свежие сообщения, нужно щелкнуть на строчке Входящие (Inbox), и тогда правая часть окна разделится на две области.

Правая и левая, а также верхняя и нижняя части окна заключены в рамки. Наведя курсор на рамку, ее можно передвигать, увеличивая или уменьшая таким образом эту часть окна.

Для упорядочивания (сортировки) электронной почты служит панель Папки (Folders) программы Outlook Express. В ней имеется пять стандартных папок:

– Входящие (Inbox). Здесь хранятся письма, которые вы еще не прочитали, а также те, которые вы уже прочитали, но еще не удалили (или еще не поместили в другие папки).

– Исходящие (Outbox). В этой папке содержатся пока не отправленные вами письма.

– Отправленные (Send items). Здесь хранятся уже отправленные вами письма.

– Удаленные (Deleted items). В эту папке сохраняются удаленные вами письма.

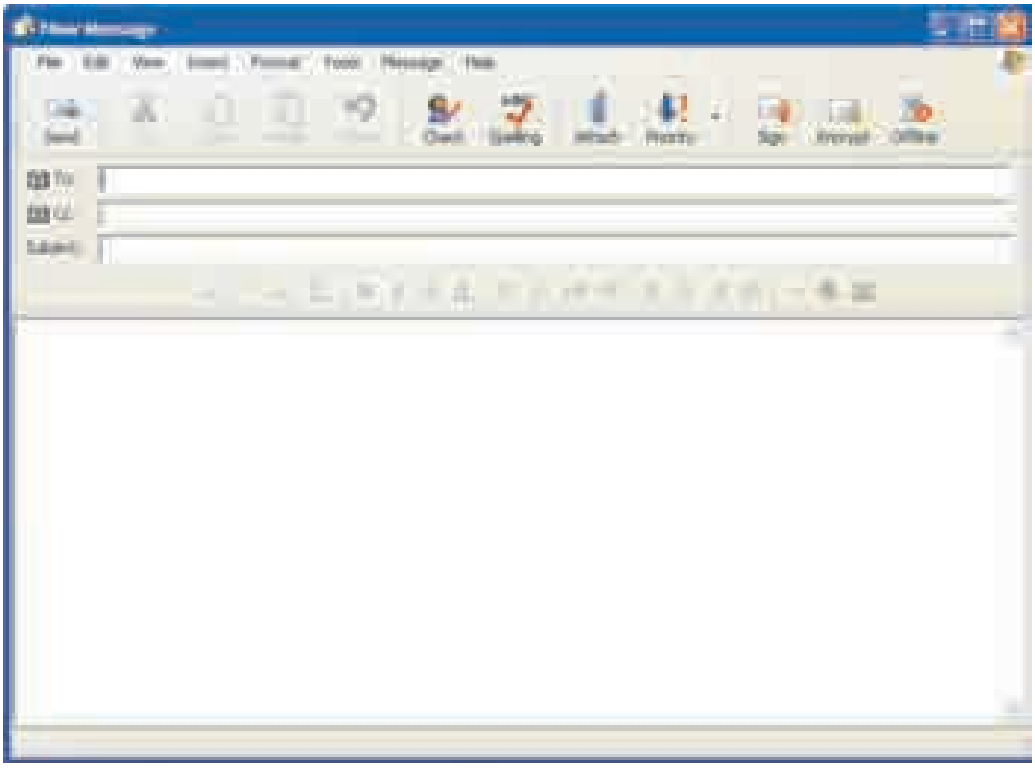
– Черновики (Drafts). В этой папке содержатся письма, которые вы пока не собираетесь отправлять.

Чтобы отправить письмо, сначала нужно щелкнуть на кнопке Создать сообщение (Creat mail) . Откроется окно Новое сообщение (New Message), в котором следует ввести необходимый текст.

В поле Кому (To) указывается электронный адрес получателя письма.

В поле Копия (Cc) указывается адрес человека (или адреса сразу нескольких людей), которому вы хотели бы отправить копию этого письма (если вам необходимо такое сделать). Однако стоит помнить, что каждый получатель такого письма-копии будет автоматически извещен о том, что данное письмо было направлено не только ему.

В поле Тема (Subject) дается краткое описание содержания письма. Оно должно как можно более точно достоверно отображать содержание, так как именно тему письма получатель увидит, прежде всего, в папке Входящие, причем точно в таком виде, в каком ее задал отправитель.



Создание нового сообщения в программе Outlook Express

Не забудьте написать текст письма! Для проверки орфографии текста нужно щелкнуть на кнопке Правописание (ABC spelling) . Система проверки орфографии выделяет “сомнительные” слова в тексте, после чего пользователю остается только выбрать нужный вариант данного слова в открывшемся диалоговом окне (точно так же, как это делается в текстовом редакторе).

И вот сообщение готово, остается только отправить его. Щелкните на кнопке Отправить (Send) – и письмо уйдет адресату.

Если же вы по какой-то причине передумали отправлять уже написанное письмо, просто закройте окно Создать сообщение (New Message). При этом программа спросит у вас, нужно ли сохранить данное сообщение. Если вы ответите Да (Yes), то письмо будет сохранено в папке Черновики (Drafts), если Нет (No), то письмо будет удалено.

Создание нового сообщения. Проверка орфографии. Папка Черновики (Drafts)

- Для создания нового сообщения выберите команду

  File ⇨ New ⇨ Mail Message

  или комбинацию клавиш

  <Ctrl+N>

- Программа проверки орфографии работает только в том случае, если на компьютере установлена программа Word либо пакет Microsoft Office.
- Если вы решили отправить одно из писем, находящихся в папке Черновики (Drafts), дважды щелкните сначала на самой этой папке, а затем – на нужном письме в ней. После чего это письмо можно будет отредактировать и послать адресату, щелкнув на кнопке Отправить (Send).

Для прочтения полученного письма выберите его в папке Входящие (Inbox). Текст письма откроется в правой части окна.

Для того чтобы распечатать текст, нужно либо выбрать команду Файл (File) ⇨ Печать (Print), либо щелкнуть на кнопке Печать (Print) на панели инструментов.

Чтобы ответить на полученное письмо, нужно щелкнуть на кнопке Ответить (Replay) , и тогда программа Outlook Express автоматически выполнит ряд действий:

1. Впишет адрес получателя в поле Кому (To).
2. Сделает перед исходной темой оригинала (присланного письма) в поле Тема (Subject) пометку Ответ (Re:).
3. *Процитирует* текст оригинала (это удобно для напоминания о предмете переписки).

Написав текст ответа в данном окне, нужно щелкнуть на кнопке Отправить (Send).

Если требуется отправить полученное вами письмо третьему лицу, нужно щелкнуть на кнопке Переслать (Forward) . В открывшемся окне появится копия оригинала (присланного письма), которую вы при желании можете дополнить собственными комментариями. Для того чтобы послать это письмо, нужно щелкнуть на кнопке Отправить (Send).

Для того чтобы удалить письмо, нужно щелкнуть на кнопке Удалить (Delete) . При этом письмо перемещается в папку Удаленные (Deleted items).

Для того чтобы одновременно отправить ответ на письмо не только его автору, но и целому ряду других адресатов, нужно щелкнуть на кнопке Ответить всем (Replay All) .

Смайлики

Для указания в письме некоторых эмоций в электронной переписке пользуются смайликами (от англ. smile – «улыбка») – стилизованными изображениями человеческого лица. Для этого обычные печатные символы рассматриваются под углом в 90°:

:-)	–	радость;
;-)	–	подмигивание;
:-(	–	грусть;
:-P	–	высунутый язык;
:-D	–	смех;
:-O	–	удивление.

1. Какие почтовые программы используются в Windows XP?
2. Что нужно сделать, чтобы просмотреть полученные электронные письма?
3. Проверьте свой электронный почтовый ящик при помощи программы Outlook Express и пошлите кому-нибудь письмо.

7

ПРАКТИКУМ

Практическая работа №1. ПЕРЕВОД ЧИСЕЛ ИЗ ОДНОЙ СИСТЕМЫ СЧИСЛЕНИЯ В ДРУГУЮ ПРИ ПОМОЩИ ПРОГРАММЫ КАЛЬКУЛЯТОР

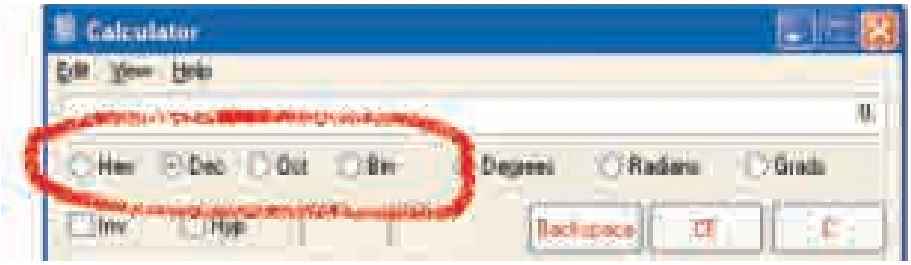
Цель работы: научиться переводить целые числа из десятичной системы счисления в двоичную и наоборот при помощи программы Калькулятор.

Вы уже научились переводить числа из десятичной системы счисления в двоичную, восьмеричную и шестнадцатеричную системы счисления. Но подобные вычисления подчас отнимают много времени. Каким образом облегчить эту работу? Для этого можно использовать программу Калькулятор операционной системы Windows.

Чтобы запустить программу Калькулятор, надо открыть меню **Пуск (Start)** и выбрать в нем последовательно команды **Программы (All Programs)** ⇒ **Стандартные (Accsessories)** ⇒ **Калькулятор (Calculator)**. После запуска программы появится ее окно.

Для того чтобы перевести окно программы из вида **Стандартный (Standart)** в вид **Инженерный (Scientific)**, нужно из основного меню выбрать команду **Вид-Инженерный (View-Scientific)**. Калькулятор “расширится”, и на нем возникнут дополнительные кнопки.

Обратите внимание на эту группу кнопок:



Bin – означает двоичную систему счисления; при выборе этой кнопки на числовой панели Калькулятора будут активны только две кнопки – “1” и “0”.

Oct – означает восьмеричную систему счисления; в этом режиме можно использовать цифры от 0 до 7.

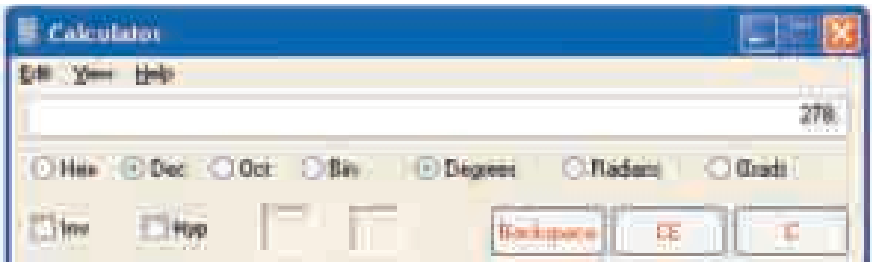
Dec – означает десятичную систему счисления; в этом режиме можно использовать цифры от 0 до 9.

Hex – означает шестнадцатеричную систему счисления; в этом режиме можно использовать цифры от 0 до 9 и буквы от **“A”** до **“F”**.

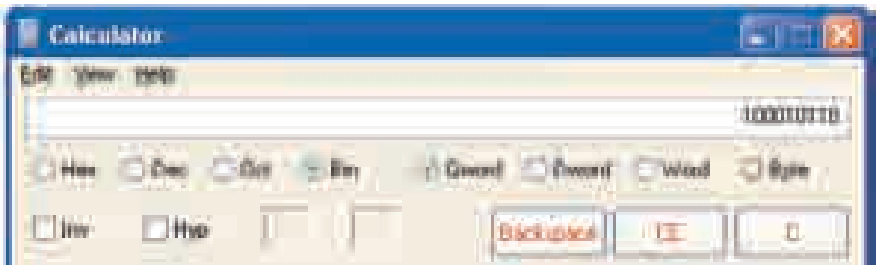
По умолчанию исходной задается десятичная система счисления **(Dec)**.

Для перевода числа из десятичной системы счисления в двоичную нужно:

1. Щелкая на кнопках Калькулятора (или же посредством клавиш клавиатуры) ввести в пустой строке нужное число – например, “278”.



2. Щелкнуть на кнопке **Bin**. Появится изображение заданного числа в двоичной системе счисления.



Значит, $278_{10}=100010110_2$.

ЗАДАНИЕ

Используя программу **Калькулятор,** переведите числа 567, 346, 11, 89, 333 в двоичную и восьмеричную системы счисления.

Практическая работа №2. СОЗДАНИЕ БЛОК-СХЕМ В ПРОГРАММЕ PAINT

Цель работы: научиться чертить простейшие блок-схемы в программе Paint.

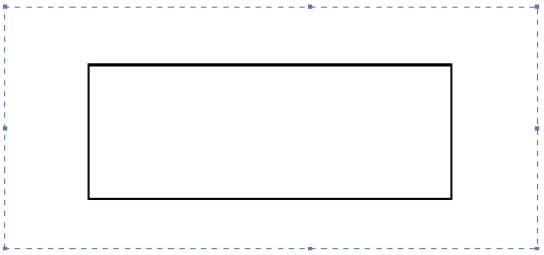
Как вы знаете, для изображения в программе Paint простых фигур, вроде прямоугольника или овала, используются соответствующие инструменты и .

Помимо этого, в блок-схемах используются параллелограммы (блок ввода и вывода переменных) и ромбы (блок проверки условий).

Для того чтобы начертить параллелограмм, нужно:

1. Взяв инструмент **Линия** , выбрать в открывшемся меню настройки толщину линии.
2. Нарисовать прямоугольник.

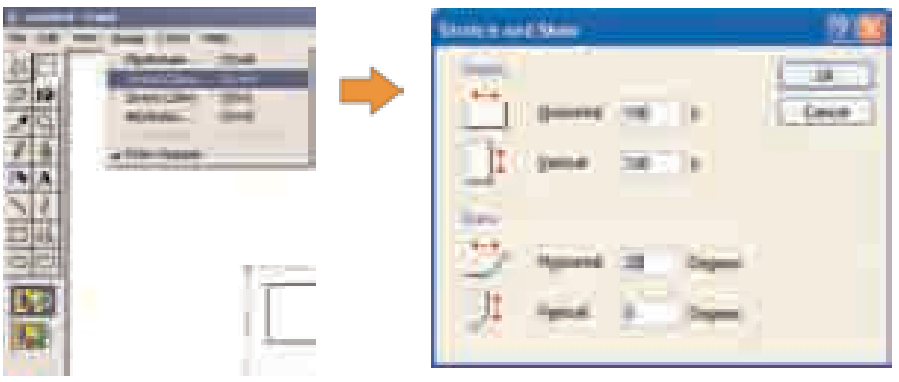
3. Выделить прямоугольник при помощи инструмента Выделение (Select) .



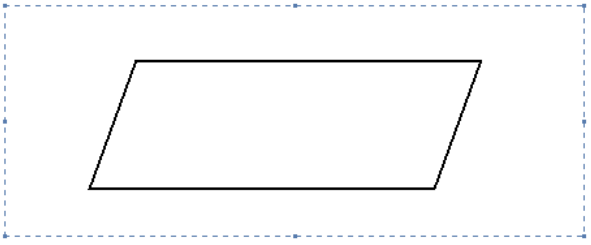
4. Выбрать команду

Рисунок (Image) ⇒ Растянуть/Наклонить… (Stretch/Skew…)

5. В открывшемся окне, в разделе **Наклонить (Skew)**, в поле **По горизонтали (Horizontal)** ввести **20** и щелкнуть на кнопке **ОК**.



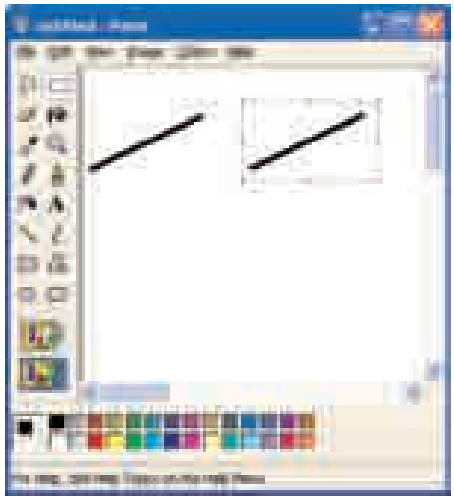
Прямоугольник превратится в параллелограмм.



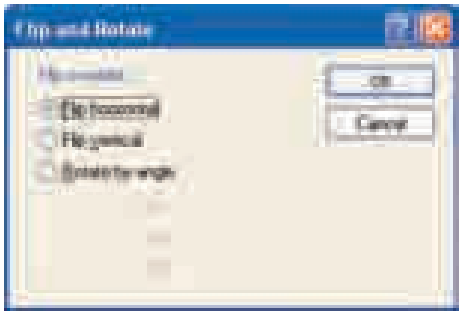
Для того чтобы начертить ромб, нужно:

1. Начертить отрезок, как показано на рисунке.

2. Взяв инструмент **Выделение (Select)** и удерживая нажатой клавишу **Ctrl**, скопировать его.



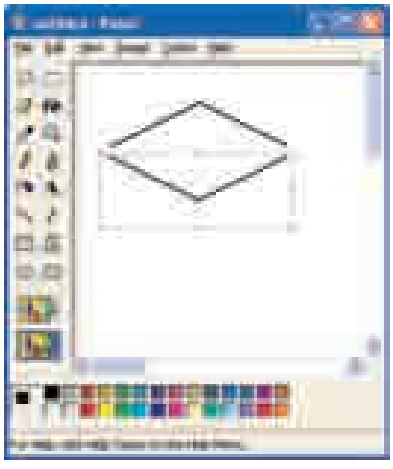
3. Выбрать команду **Рисунок (Image)** ⇒ **Отразить/Повернуть… (Flip /Rotate…)**.
4. В открывшемся окне выбрать **Отразить слева направо (Flip horizontal)** и щелкнуть на кнопке **ОК**.



5. Соединить концы двух отрезков – первоначального и вновь полученного.

6. Чтобы убрать фон, на котором располагается фигура, нужно, взяв инструмент Выделение (Select), выбрать в меню настройки нижний из двух рисунков.

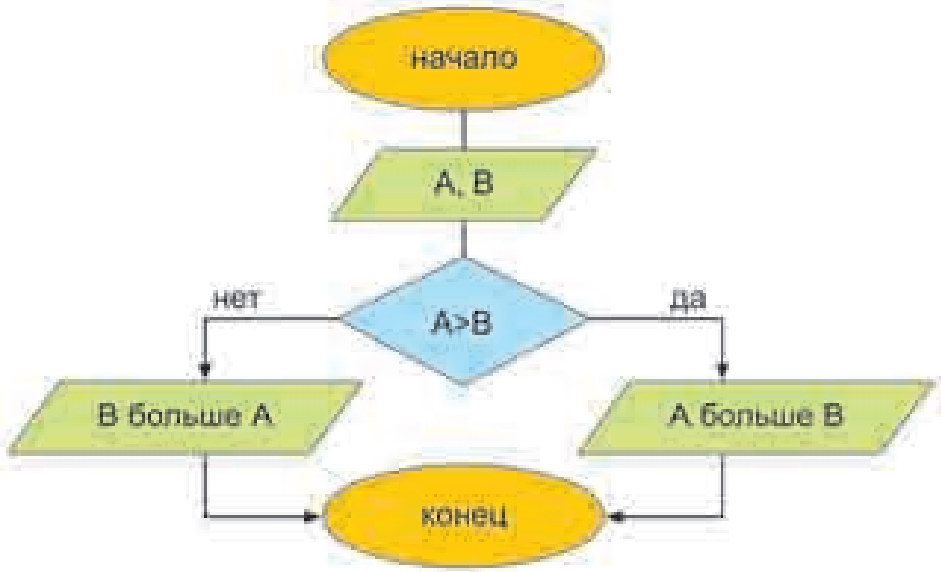
7. Скопировать полученную фигуру. Затем в окне **Отразить/Повернуть… (Flip /Rotate…)** выбрать **Отразить сверху вниз (Flip vertical)**.



8. Соединить половинки ромба в целое.

ЗАДАНИЕ

Начертите в программе Paint блок-схему алгоритма для вычисления большего из двух чисел



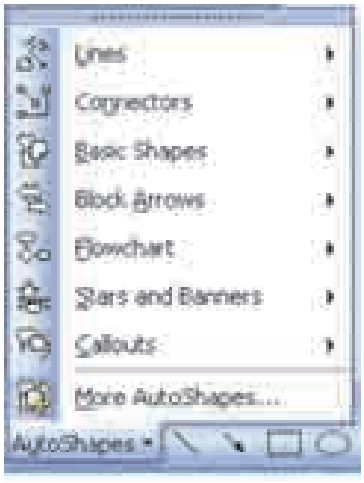
При наборе текста используйте шрифт **Times New Roman, 12 pt**.

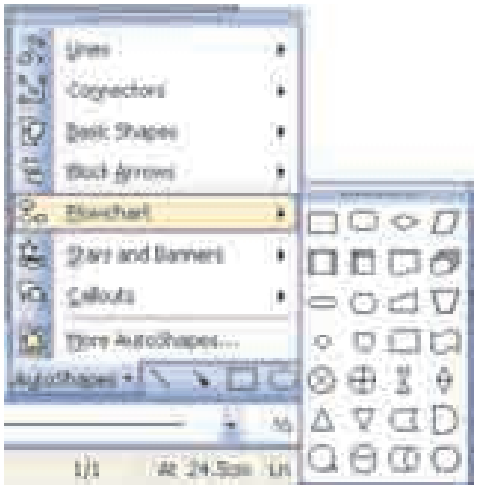
Практическая работа №3. ИЗОБРАЖЕНИЕ СТАНДАРТНЫХ ГРАФИЧЕСКИХ ОБЪЕКТОВ БЛОК-СХЕМ В ПРОГРАММЕ WORD

Цель работы: построить простейшие блок-схемы с применением элементов текстового редактора Word.

Как вам известно, в программе Word инструменты для создания изображений располагаются на панели **Рисование (Draw)**.

Для создания графических элементов зачастую пользуются готовыми Автофигурами **(AutoShapes)**, находящимися на панели инструментов **Рисование (Draw)**. Стандартные элементы блок-схем приведены в соответствующем пункте ее меню – **Блок-схема (Flowchart)**.





Чтобы найти требуемую фигуру для блок-схемы, нужно:

1. Выбрать команду **Автофигуры (AutoShapes)** ⇒ **Блок-схема (Flowchart)** на панели инструментов **Рисование (Draw)**.
2. В открывшемся меню щелкнуть левой кнопкой мыши на нужной фигуре.
3. Удерживая левую кнопку мыши, увеличить фигуру на рабочем поле до необходимого размера, а затем отпустить кнопку.
4. Чтобы переместить выделенную (то есть маркированную по краю) фигуру, надо, щелкнув на ней левой кнопкой мыши и удерживая ее, перетащить фигуру на нужное место.

5. Чтобы поместить текст внутрь фигуры, надо щелкнуть на ней правой кнопкой мыши и, выбрав в открывшемся меню строку **Добавить текст (Add text)**, ввести нужный текст.

Чтобы связать воедино фигуры и соединяющие их отрезки, надо, используя инструмент **Выбор объектов (Select Objects)** на панели инструментов **Рисование (Draw)**, выделить нужные объекты и, щелкнув на рисунке правой кнопкой мыши, выбрать в открывшемся контекстном меню команду Группировка

Группировка (Groping) ⇒ Группировать (Group)

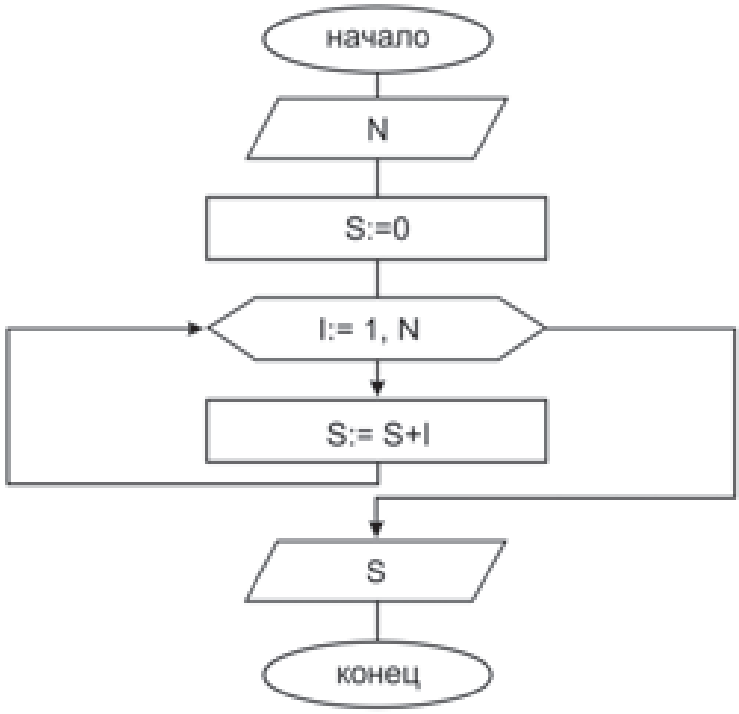
В результате объект, состоящий из отдельных блоков, превратится в единый графический объект, и тогда над ним можно будет совершать любые операции, применимые к графическим объектам.

ЗАДАНИЕ

Используя графические объекты панели **Рисование (Draw)**, составьте блок-схему алгоритма для нахождения суммы

1 + 2 + …+ N

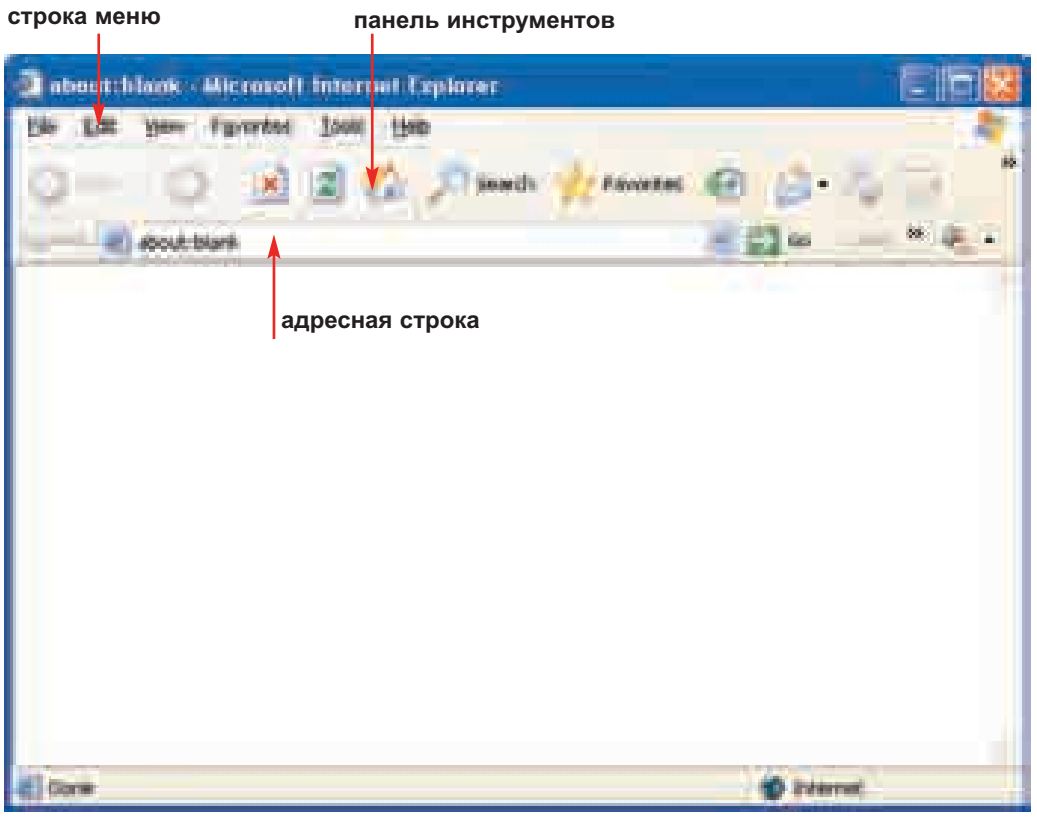
где **N** - натуральное число.



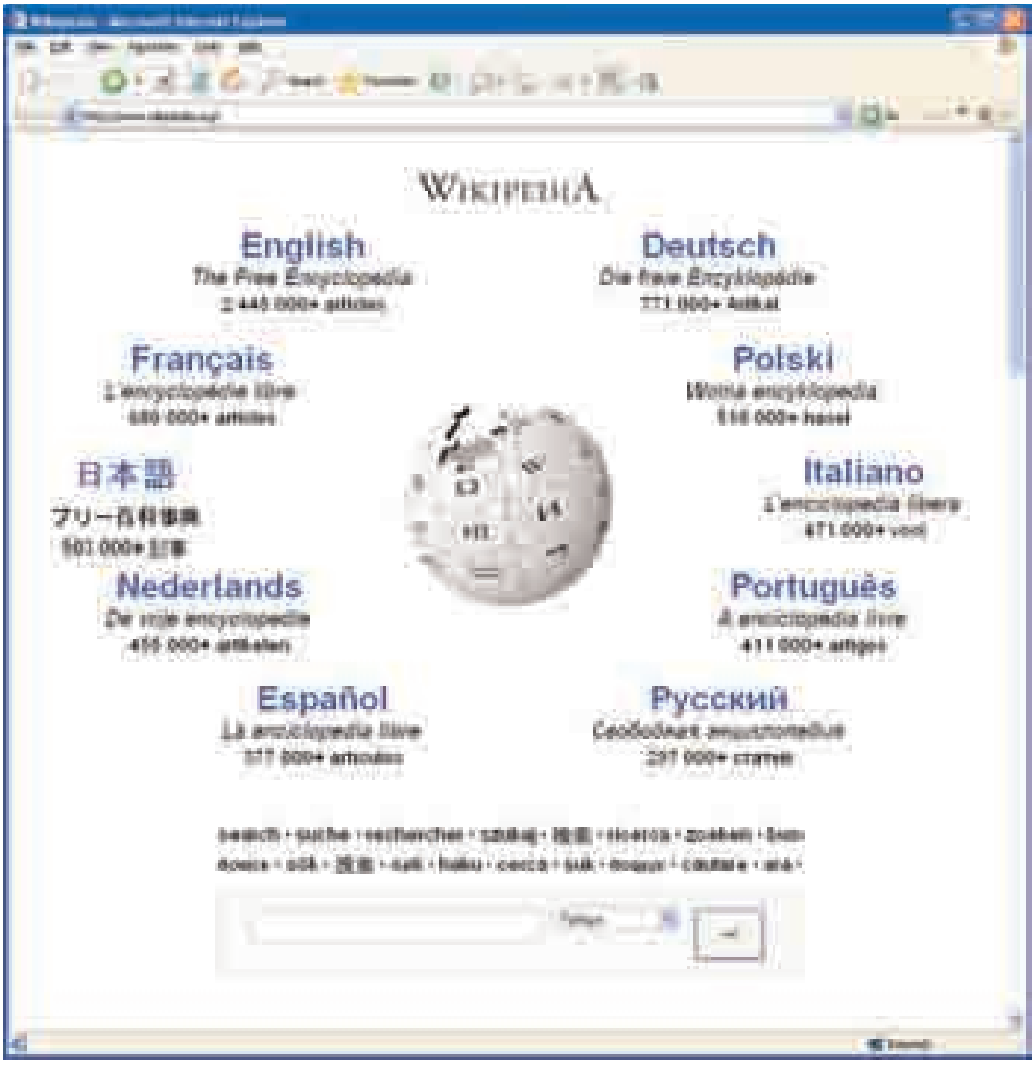
Практическая работа №4. ОЗНАКОМЛЕНИЕ С ПРОГРАММОЙ INTERNET EXPLORER

Цель работы: научиться настраивать браузер Internet Explorer и просматривать с его помощью веб-страницы.

Чтобы запустить программу Internet Explorer, сделайте двойной щелчок по ее иконке на Рабочем столе. На экране откроется окно **Internet Explorer**:



Адрес веб-страницы вводится в **адресную строку**. Наберите в адресной строке **www.wikipedia.org** и нажмите кнопку . В рабочем поле окна откроется главная страница сайта.



Если при наведении курсора мыши на слово или предложение он принимает форму руки , это означает, что щелкнув на этом слове или предложении, можно перейти на другую страницу.

Для того чтобы сохранить открытую веб-страницу в памяти компьютера, нужно выбрать в строке меню команду

Файл (File) ⇒ Сохранить как (Save as)

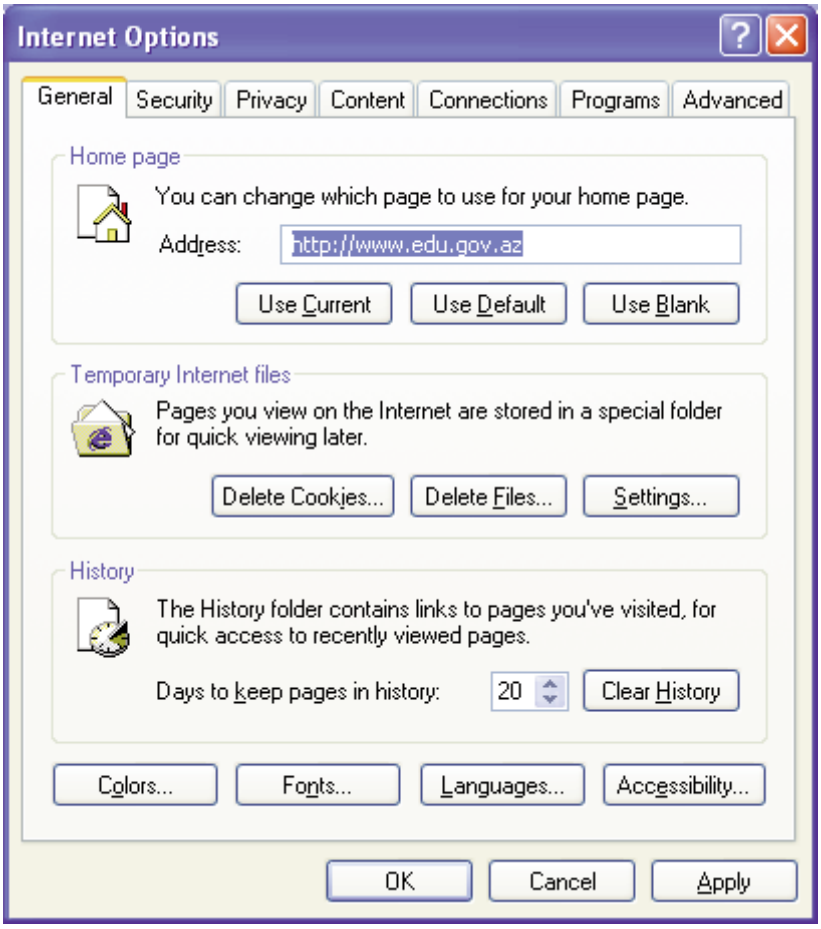
В качестве домашней страницы **Internet Explorer** вы может выбрать любую страницу (в том числе пустую).

Это делается так:

1 Введите в адресную строку адрес нужной страницы. Например,

www.wikipedia.org

2 Выберите команду
Сервис (Tools) ⇒ Свойства обозревателя (Internet Options).
В открывшемся диалоговом окне перейдите на вкладку
Общие (General) и щелкните на кнопке **С текущей (Use Current)**.



3 Щелкните на кнопке **ОК** и закройте окно

Если вы хотите установить в качестве домашней пустую страницу, то выбрав команду

Сервис (Tools) ⇒ Свойства обозревателя (Internet (Options)

и перейдя на вкладку **Общие (General)**, щелкните на кнопке **С пустой (Use blank)**, а затем на кнопке **ОК**.

ЗАДАНИЕ

При помощи браузера **Internet Explorer** посетите следующие сайты:

www.president.az

www.edu.gov.az

www.tgdk.gov.az

www.ikt.az

www.baku.ru

Переходите по гиперссылкам с уже открытых страниц на другие.

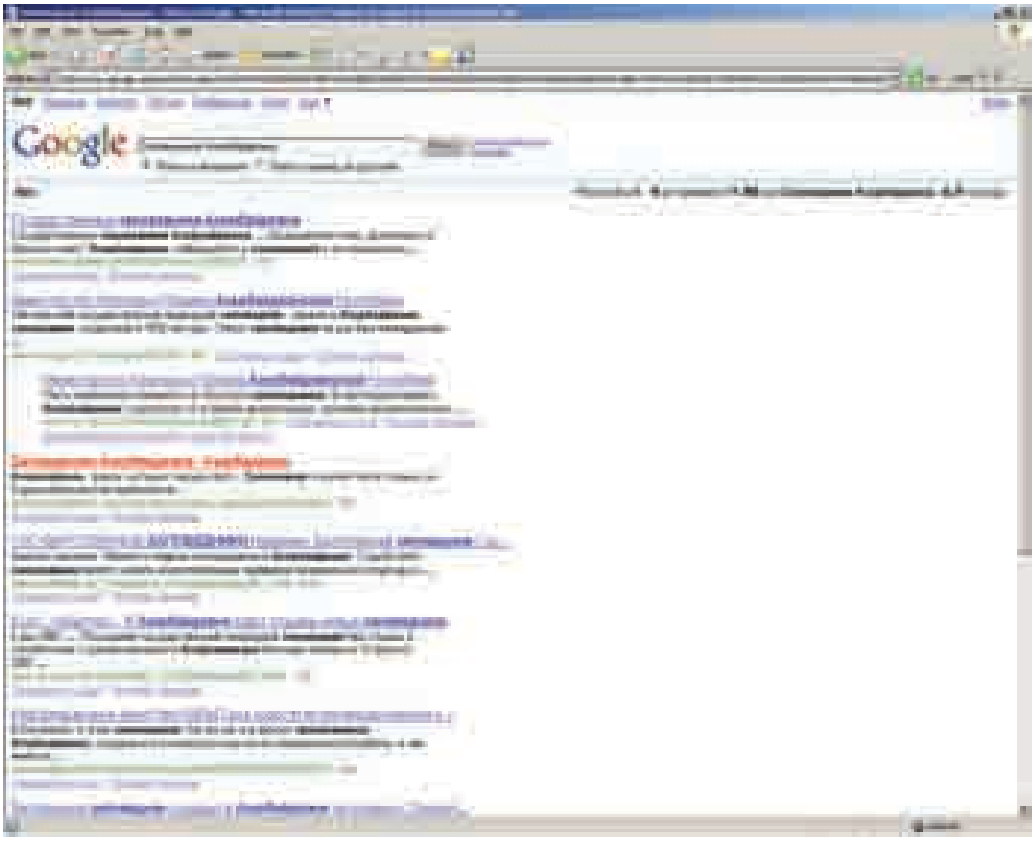
Практическая работа №5. Поиск информации в Интернете

Цель работы: поиск в Интернете нужных документов и файлов с помощью различных поисковых систем

Запустите программу Internet Explorer. Введите в адресную строку адрес поисковой системы, например

www.google.az.

В открывшемся окне поисковой системы **Google** введите в пустую строчку (поле поиска) запрос – например, **“заповедники Азербайджана”** – и щелкните на кнопке **Поиск (Search)**. Во вновь открывшемся окне будут показаны результаты поиска.



Как видите, по данному запросу было обнаружено около 54 800 документов. Первые 10 из них отражены в открывшемся окне. Чтобы открыть какую-либо из этих страниц, надо щелкнуть правой кнопкой мыши на соответствующей гиперссылке (они отмечены синим цветом) и в открывшемся списке выбрать **Открыть в новом окне (Open in New Window)**.

ЗАДАНИЕ

1. При помощи поисковой системы **Google** соберите информацию о том или ином городе (например, Баку, Москва, Стамбул, Вашингтон, Лондон и др.).
2. Сохраните в памяти компьютера наиболее интересные из найденных материалов.

Программа
XVIII класс
(1 час в неделю, итого 32 часа)

I. Информация (3 часа)
Информация. Данные. Знание. Основные свойства информации. Сбор, обработка, передача, поиск и защита информации.

II. История вычислительной техники (2 часа)
История вычислительной техники. Поколения компьютеров.

III. Арифметические основы компьютера (5 часов)
Системы счисления. Позиционные системы счисления. Взаимосвязи между системами счисления. Двоичная система счисления.
Практическая работа №1. Перевод чисел из одной системы счисления в другую при помощи программы Калькулятор

IV. Логические основы компьютера (3 часа)
Формальная логика. Логические операции. История науки логики.

V. Программирование (11 часов)
Алгоритмический язык. Вспомогательные алгоритмы. Циклы. Условия в алгоритмическом языке. Ветвление. Переменные величины в алгоритмическом языке. Ввод и вывод информации. Результаты алгоритмов и алгоритмы-функции.
Практическая работа №2. Изображение блок-схем в программе Paint.
Практическая работа №3. Изображение стандартных графических объектов блок-схем в программе Word.

VI. Коммуникационные технологии (8 часов)
Компьютерные сети. Интернет. “Всемирная паутина”. Электронная почта. Программа Outlook Express.
Практическая работа №4. Ознакомление с программой Internet Explorer.
Практическая работа №5. Поиск информации в Интернете.

ОГЛАВЛЕНИЕ

1 | ИНФОРМАЦИЯ
1.1. Информация. Данные. Знание 3
1.2. Прием информации. Основные свойства информации 5
1.3. Сбор, обработка, передача, поиск и хранение информации 6

2 | ИСТОРИЯ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ
2.1. История вычислительной техники 9
2.2. Поколения компьютеров 16

3 | МАТЕМАТИЧЕСКИЕ ОСНОВЫ КОМПЬЮТЕРА
3.1. Системы счисления 21
3.2. Позиционные системы счисления 25
3.3. Перевод чисел из одной системы счисления в другую 29
3.4. Двоичная система счисления 33

4 | ЛОГИЧЕСКИЕ ОСНОВЫ КОМПЬЮТЕРА
4.1. Формальная логика 37
4.2. Логические операции 40
4.3. История науки логики 47

5 | ПРОГРАММИРОВАНИЕ
5.1. Алгоритмический язык 51
5.2. Вспомогательные алгоритмы 56
5.3. Циклы 61
5.4. Условия в алгоритмическом языке. Ветвление 67
5.5. Переменные величины в алгоритмическом языке 73
5.6. Ввод и вывод информации 77
5.7. Результаты алгоритмов и алгоритмы-функции 80

6 | КОММУНИКАЦИОННЫЕ ТЕХНОЛОГИИ
6.1. Компьютерные сети 85
6.2. Интернет 91
6.3. WWW 95
6.4. Электронная почта 100
6.5. Программа Outlook Express 104

7 | ПРАКТИКУМ

Практическая работа №1. Перевод чисел из одной системы счисления в другую при помощи программы **Калькулятор** 109
Практическая работа №2. Изображение блок-схем в программе **Paint** 112
Практическая работа №3. Изображение стандартных графических объектов блок-схем в программе **Word** 117
Практическая работа №4. Ознакомление с программой **Internet Explorer** 119
Практическая работа №5. Поиск информации в Интернете . . . 123

Программа 125

Исмаил Джалал оглу Садыгов
Рамин Алиназим оглу Махмудзаде
Наида Ризван гызы Исаева

Информатика – **8.** Учебник для 8-го класса средней школы.

“Bakınəşr”, “TM group, 2010.

Формат 70×100 1/16. Бумага офсетная №1. Физ.печ.л. 8,0.
Подписано в печать 25.05.2010. Гарнитура “Tims New Roman”. Тираж 9300.
Бесплатно.