Информатика
7 класс
Теория

Содержание

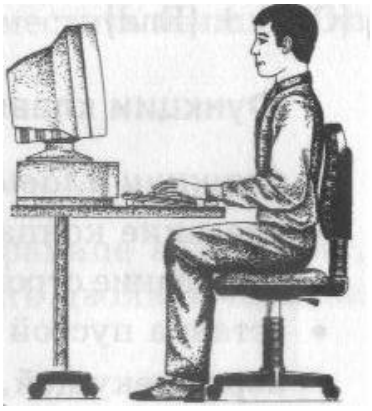
© **Минцис Дмитрий Александрович**
учитель информатики, заместитель директора по ИКТ школы № 53
Эйхгорн Елена Владимировна
учитель информатики школы № 53
2010 год

Техника безопасности и организация рабочего места в кабинете информатики

Чтобы избежать несчастного случая, поражения электрическим током и поломки оборудования необходимо выполнять следующие правила:

- Входите в компьютерный класс спокойно, не торопясь и не толкаясь, не задевая мебель и оборудование и только с разрешения преподавателя.
- Не включайте и не выключайте компьютеры без разрешения преподавателя.
- Не трогайте провода и разъемы соединительных кабелей.
- Не прикасайтесь к экрану и тыльной стороне монитора.
- Не размещайте на рабочем месте посторонние предметы.
- Не вставайте со своих мест, когда в кабинет входят посетители.
- Не пытайтесь самостоятельно устранять неисправности в работе аппаратуры; при неполадках и сбоях в работе компьютера немедленно прекратите работу и сообщите об этом преподавателю.
- Работайте на клавиатуре чистыми сухими руками; легко нажимайте на клавиши, не допуская резких ударов и не задерживая клавиши в нажатом положении.

Чтобы не навредить своему здоровью, необходимо соблюдать ряд простых рекомендаций:



- Неправильная посадка за компьютером может стать причиной боли в плечах и пояснице. Поэтому сидите свободно, без напряжения, не сутулясь, не наклоняясь вперед и не наваливаясь на спинку стула. Ноги ставьте прямо на пол, одна возле другой, не вытягивайте их и не подгибайте.
- Туловище должно находиться от стола на расстоянии 15-20 см. линия взора должна быть направлена в центр экрана. Если вы имеете очки для постоянного ношения, работайте в очках.
- Плечи при работе должны быть расслаблены. Желательно, чтобы предплечья находились на одном уровне с клавиатурой.
- При напряженной длительной работе глаза переутомляются, поэтому каждые 5 минут отрывайте взгляд от экрана и смотрите на что-нибудь находящееся вдали.

1. Роль информации в деятельности человека.

1.1. Информация. Информатика

Информация играет в жизни людей огромную роль. Информация для человека – это знания, которые он получает из различных источников. Человек изучает окружающий мир с момента появления на Земле. Сведения об окружающем мире мы воспринимаем с помощью органов чувств и представляем в удобной для дальнейшей работы форме.

Совершенно очевидна роль информации и для животных: только способность вовремя получить информацию об окружающем мире позволяет им выжить в условиях естественного отбора. Применимо понятие «информация» и в растительном мире: например, растения распускаются весной только при определенных погодных условиях.

Итак, будем называть **информацией** данные об объектах окружающего мира, которыми обмениваются между собой люди и живая природа с целью увеличения знания о нем.

Все знания, которыми обладает каждый человек, условно можно разделить на две группы: **факты** и **правила**. К фактам относятся знания об определенных явлениях (солнечные и лунные затмения происходят тогда, когда Солнце, Луна и Земля как бы выстраиваются друг за другом в линию), событиях (первый персональный компьютер фирмы IBM появился 1981 году), свойствах объектов (вес первой электронной вычислительной машины равнялся 30 тоннам) и зависимостях между объектами (у квадрата все углы прямые и все стороны равны). К правилам относятся знания о последовательностях действий, направленных на решение конкретной задачи (выполнить фонетический разбор слова, вылечиться от простуды, решить уравнение). Существуют две формы получения знаний (информации): **чувственное** и **логическое.**

С помощью органов чувств мы познаем отдельные предметы и явления окружающего мира. В этом случае информация на следующие **виды**:

- Визуальная - информация, поступающая через органы зрения (глаза): при чтении книги, сообщений с экрана монитора, просмотре телепередачи и т.д. Через зрение мы получаем до 90% информации об окружающем мире.
- Аудиальная - информация, поступающая через органы слуха (уши): восприятие речи, музыки и т.д. Через слух мы получаем до 9% информации.
- Тактильная - информация, поступающая через органы осязания (кожа), например восприятие «на ощупь» состояния поверхности предмета (гладкий или шершавый) или его температуры (холодный или теплый).

- Вкусовая - информация, поступающая через органы вкуса (язык), например восприятие вкуса еды.
- Обонятельная - информация, поступающая через органы обоняния (нос): ощущение запахов, которые распространяют объекты.

Законы мира, сущность предметов и явлений, общее в них мы познаем с помощью мышления. Именно мышление позволяет нам грамотно сформулировать определение понятия информатика. **Информатика** - наука о поиске, получении, хранении, обработке и передаче информации при помощи информационных систем, важнейшим элементом которых является компьютер.

1.2. Свойства информации

Люди должны обмениваться между собой качественной информацией. Это позволит лучше понять друг друга. В повседневной жизни от свойств информации часто зависят жизнь и здоровье людей. Выделяют пять свойств информации, определяющие ее качество: **полезность** (учебник по физике 7 класса содержит для вас полезную информацию, но для ученика 10 класса в нем нет ничего нового), **достоверность** (ошибка в справочнике по математике не позволит вам правильно решить задачу); **актуальность** (своевременный прогноз погоды может уберечь вас от природных катаклизмов); **полнота (**договариваясь о встрече с другом важно точно оговорить время и место встречи); **понятность** (поймете ли вы информацию, которая изложена в учебнике алгебры 10 класса).

1.3. Информационная деятельность человека

Информация необходима человеку в его жизни. Анализ и выводы человек делает сначала бессознательно, а потом осознанно. Все обучение в школе и в других учебных заведениях направлено на развитие умения быстро и качественно анализировать информацию. Вся деятельность человека связана с различными действиями с информацией: сбором, обработкой, передачей, хранением, поиском, защитой. С развитием компьютеров удалось значительно ускорить выполнение и повысить качество этих видов работы с информацией.

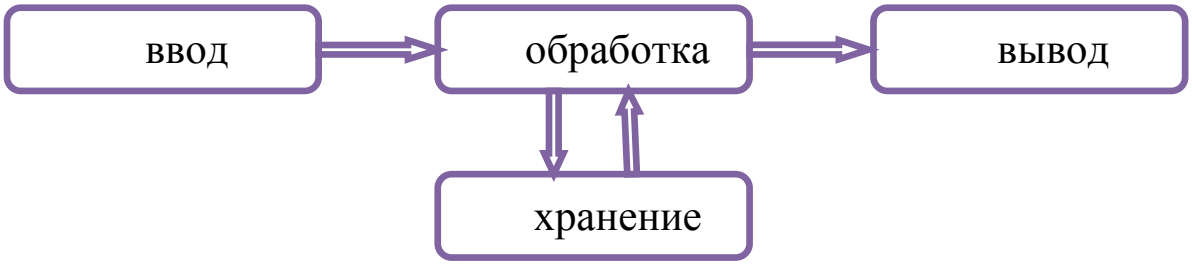
Любую информацию можно разделить на **входную**, которую получает человек или устройство, и **выходную**, которая получается после обработки входной информации человеком или устройством. Например, на уроках геометрии школьник изучает определения и теоремы. Когда учитель предлагает очередную задачу (входная информация), ученик обдумывает последовательность решения, вспоминая, что из изученного ему необходимо применить (обработка информации). Наконец, он находит ответ (выходная информация).

2. Техническое обеспечение информационных процессов

2.1. Компьютер как средство обработки информации

Структурная схема работы компьютера

Компьютер используется в различных областях человеческой деятельности. Например, для проектирования обуви и самолетов, для восстановления прежнего облика мумии, для контроля за уровнем масла в двигателе автомобиля, для управления полетом космического корабля и т.д. В каждом случае можно выделить основные этапы работы с информацией: ввод, хранение, обработка, преобразование. Условно, работу персонального компьютера (ПК) можно представить в виде схемы:



Для выполнения этих функций необходимы технические устройства и программы. Систему взаимосвязанных технических устройств, выполняющих ввод, хранение, обработку и вывод информации, называют **аппаратной частью** ПК. Совокупность всех программ, обеспечивающих процесс обработки информации, называют **программным обеспечением** ПК.

2.2. Основные устройства компьютера, их назначение.

Аппаратную часть ПК можно разделить на две основные составляющие: **системный блок**, осуществляющий обработку информации, и **периферийные устройства**, с помощью которых вводится или выводится информация из системного блока.

Системный блок

В системном блоке выделяют следующие устройства:

1. **материнская плата** - плата, к которой подключаются все устройства системного блока компьютера, а через них - вся периферия;
2. **центральный процессор** (ЦП) - микросхему, предназначенную для обработки информации и управления устройствами системного блока;
3. **оперативная память** (ОП) - микросхема, которая предназначена для хранения работающих программ и обрабатываемых данных. При выключении электропитания информация из ОП стирается.;
4. **внешняя память** - энергонезависимая память, предназначенная для долговременного хранения больших объемов информации.
5. **адаптеры внешних устройств (карты)** -

устройства, согласующие работу системного блока и внешних устройств. Например, видеоадаптер согласует работу системного блока и монитора, который подсоединяется к системному блоку именно через разъем видеоадаптера.

Периферийные устройства

Устройства ввода информации:

1. **Клавиатура**: устройство ввода символьной информации.

2. **Мышь**: устройство ввода графической информации, а также манипулятор для работы с графическим интерфейсом любой программы.

3. **Сканер** - устройство для преобразования информации с печатного оригинала в компьютерный формат. Сканирование осуществляется с двумя целями: сканирование графических изображений (фотографий, схем); сканирование текста для его дальнейшего распознавания и перевода в текстовый компьютерный формат.

4. **Дигитайзер** - устройство для высокоточного ввода графической информации, например рисунков или схем. Состоит из графического планшета и светового пера - небольшой ручки для рисования по данному планшету.

Устройства вывода информации:

1. **Монитор** - устройство для визуального вывода информации.

Активно внедряются во все сферы деятельности и интерактивные (сенсорные) мониторы, которые, кроме вывода информации, могут обеспечивать и ее ввод: по нажатию пальца или стилуса.

2. **Акустическая система**. К ней относятся не только устройства вывода звука (наушники, колонки), но и устройства ввода (микрофон).

3. **Принтер** - устройство для вывода информации в печатном виде. Различают матричные, струйные и лазерные принтеры. Основные характеристики: качество и скорость печати. У хороших лазерных принтеров скорость печати достигает десятков страниц в минуту, а качество близко к типографскому.

2.3. Представление информации в компьютере

В процессах восприятия, передачи и хранения информации живыми организмами, человеком и техническими устройствами происходит ее **кодирование**, т.е. процесс представления информации в виде кода. **Код** - система условных знаков. Соответственно **декодирование** – процесс, обратный кодированию. Основой компьютерного кодирования является двоичный код. Это обусловлено тем, что, если человек может воспринимать нюансы (плохо, очень плохо, мало хорошего, ничего плохого, очень хорошо и т.п.), то техническим системам это несвойственно. Технике больше подходят два

противоположных состояния: включено-выключено, замкнуто-разомкнуто, намагничено - не намагничено, есть сигнал – нет сигнала.

Любая информация в компьютере представляется при помощи двоичной системы счисления, алфавит которой состоит из двух знаков (цифр): 0 и 1. **Бит (алфавитный подход)** - одна цифра при двоичном кодировании: 0 или 1 (binary digit - двоичная цифра).

Итак, числовая, текстовая, графическая и звуковая информация может обрабатываться компьютером, если она представлена в двоичном коде, т.е. в виде последовательности нулей и единиц, которые мы называем данными. Для обработки в компьютере данные представляются в форме последовательностей электрических импульсов.

В таблице приведены примеры представления человеком и компьютером различных типов данных: числа 5, буквы «А», точки черного цвета и звука максимальной громкости.

Таблица **Представление информации человеком и компьютером**

Тип информации	Человек	Компьютер	
		Двоичный код	Последовательность электрических импульсов
Числовая	5	00000101	0 0 0 0 0 1 0 1
Текстовая	А	11000000	1 1 0 0 0 0 0 0
Графическая	●	00000000	0 0 0 0 0 0 0 0
Звуковая	Звук максимальной громкости	11111111	1 1 1 1 1 1 1 1

2.4. Системы счисления. Двоичная система счисления

Система счисления (с. сч.) - это знаковая система, в которой числа записываются по определенным правилам при помощи цифр. Цифра - это символ, имеющий определенное количественное значение.

Системы счисления делятся на позиционные и непозиционные.

1. **Непозиционная с. сч.** - с. сч., в которой количественное значение цифры не зависит от ее расположения в числе. Пример - римская с.сч.: вне зависимости от расположения в числе цифра I имеет значение 1, V - 5, X - 10, L - 50, C - 100, D - 500, M - 1000.
2. **Позиционная с.сч.** - с. сч., в которой количественное значение цифры зависит от ее расположения (позиции) в числе. Пример – широко используемая уже много веков арабская (десятичная) с.сч. Так в числе 125 цифра 5 имеет значение 5, а в числе 521 та же цифра 5 имеет значение 500.

Ниже рассмотрены основные понятия позиционных систем счисления на примере сравнения 10-ной и 2-ной систем счисления.

3. **Основание** с.сч. (P) - количество цифр, используемых для записи числа. Значения цифр лежат в пределах от 0 до P-1.

10-ная с. сч.: P = 10; алфавит цифр: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

2-ная с. сч.: P = 2; алфавит цифр: 0, 1.

Название с.сч. производится по ее основанию: 2-ная, 3-ная, …, 10-ная, …

4. **Разряд** цифры характеризует ее позицию в числе, причем отсчет ведется справа налево, разряд единиц = 0.

Пример: 10-ная с. сч.: $4\ 3\ 8\ 9 = 4\cdot10^3 + 3\cdot10^2 + 8\cdot10^1 + 9\cdot10^0$

3 2 1 0 – разряды

5. **Запись чисел**: при накоплении младшего разряда до максимальной цифры он обнуляется, а старший увеличивается на 1.

10-ная	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
2-ная	**0**	**1**	**10**	**11**	**100**	**101**	**110**	**111**	**1000**	**1001**	**1010**	**1011**	**1100**	**1101**	**1110**	**1111**

2.5. Перевод целых чисел из десятичной в другие системы счисления

Перевод из 10-ной в 2-ную систему счисления

10-ное число делится на 2, при этом остатки от деления, записываемые справа налево, образуют искомое 2-ное число. Деление производится до получения в целой части единицы, которая записывается первой цифрой искомого 2-го числа

Примеры.

$36_{10} = ?_2$

36	2
18	100100
9	
4	
2	
1	

$36_{10} = 100100_2$

Проверка: $1\ 0\ 0\ 1\ 0\ 0_2 = 2^5 + 2^2 = 32 + 4 = 36_{10}$

5 4 3 2 1 0

$25_{10} = ?_2$

25	2
12	11001
6	
3	
1	

$25_{10} = 11001_2$

Проверка: $1\ 1\ 0\ 0\ 1_2 = 2^4 + 2^3 + 2^0 = 16 + 8 + 1 = 25_{10}$

4 3 2 1 0

Общее правило перевода следующее:

- делим данное число на основание новой с.сч.; полученный при этом остаток даст цифру, стоящую в первом (крайнем правом) разряде новой записи числа;
- затем полученное частное снова делим на основание новой с.сч. и снова получаем остаток, являющийся следующей цифрой новой записи числа;
- такое последовательное деление продолжаем до тех пор, пока частное не окажется меньше, чем основание новой с.сч., это последнее частное будет цифрой старшего разряда новой записи числа.

$417_{10} = 631_8$

417	8
52	631
6	

2.6. Перевод чисел из 2, 8 и 16-ричной и других систем счисления в десятичную

Перевод целых чисел из 2-ной в 10-ную систему счисления

- Проставить номера разрядов цифр 2-го числа: справа налево, начиная с 0-го разряда.
- Просуммировать произведения цифр числа на 2 в степени номера разряда.

Пример. $\underset{4\ 3\ 2\ 1\ 0}{1\ 0\ 1\ 0\ 1_2} = 1\cdot2^4 + 0\cdot2^3 + 1\cdot2^2 + 0\cdot2^1 + 1\cdot2^0 = 16 + 4 + 1 = 21_{10}$

Перевод чисел из любой позиционной системы счисления в десятичную выполняется по тому же правилу, только значение каждого разряда умножаем на соответствующую степень основания новой с.сч.

Пример. $\underset{2\ 1\ 0}{A\ 0\ 7_{16}} = 10\cdot16^2 + 0\cdot16^1 + 7\cdot16^0 = 256 + 0 + 7 = 263_{10}$

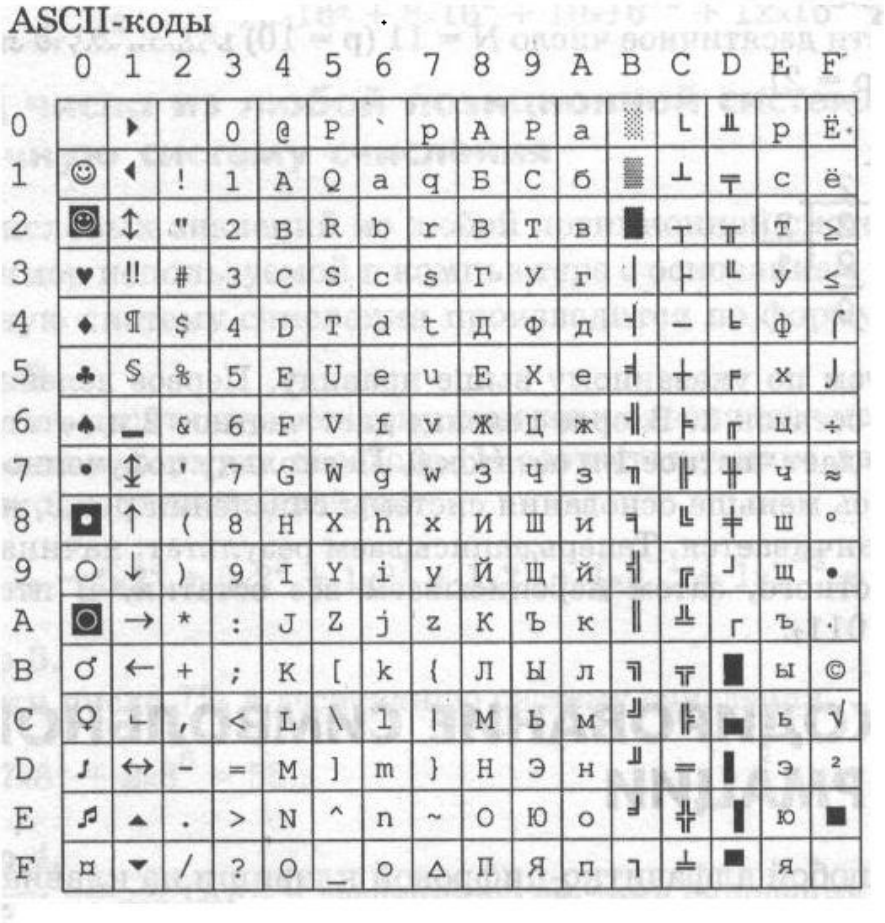
2.7. Кодирование текстовой информации

Каждый символ текста, как правило, кодируется при помощи 8 или 16 бит в зависимости от используемой кодировки.

Например, при наборе текста в Блокноте каждый символ текста кодируется при помощи 8 бит (1 байт). При нажатии на клавиатуре символьной клавиши вырабатывается соответствующий ей электрический сигнал, который преобразуется контроллером клавиатуры в двоичный код: от 00000000 (0) до 11111111 (255). Существует $2^8 = 256$ неповторяющихся комбинаций из 8 двоичных цифр, что позволяет кодировать 256 символов. Каждый символ имеет свой код в соответствии с кодовой таблицей, причем половина таблицы (коды 0 - 127) является интернациональной; вторая половина (коды 128 - 255) содержит национальный алфавит.

Интернациональная часть кодовой таблицы (соответствует ASCII - American Standard Code for Information Interchange) - от 0 до 127:

- 0 - 32: коды операций, например 8 - Backspace, 13 - Enter, 32 - пробел и т.п.
- 48 - 57: цифры (0 .. 9);
- 65 - 90: латинские прописные буквы (A .. Z);
- 97 - 122: латинские строчные буквы (a .. z).



Национальная часть кодовой таблицы (от 128 до 255) кодирует символы национального алфавита. Существует несколько кодовых таблиц для русского алфавита (кириллицы): КОИ8, CP1251, CP866, Mac, ISO; причем тексты, созданные в одной кодировке, не могут быть правильно отображены в другой без специальных программ-конверторов, которые встраиваются в приложения для работы с текстовыми документами. Например, для кодировки CP1251: 192 - 223 - коды русских заглавных букв (А .. Я, кроме Ё - 168); 224 – 255 - коды русских строчных букв (а .. я, кроме ё - 184).

Обратите внимание, что:

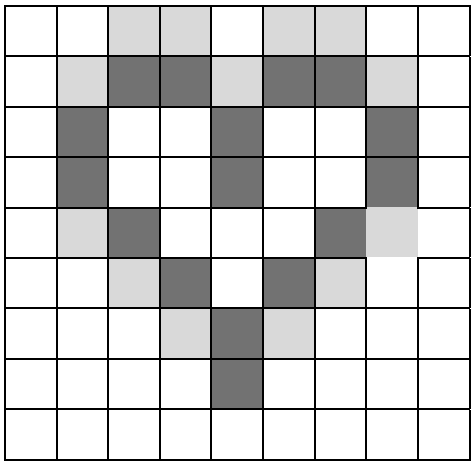
- Несмотря на одинаковое отображение на мониторе, русские и английские буквы представляются в компьютере по-разному. Например английское заглавное А имеет код 65 (01000001), а русское А – код 192 (11000000).
- Числа могут кодироваться непосредственно как числа (по правилам перевода в 2-ную систему), и как набор символов-цифр (по кодовой таблице). Например, 25 как число может быть представлено как 00011001, а как текст, состоящий из двух цифр 2 и 5: 00110010 00110101.

В связи с ограниченностью 8-битной кодировки (всего 256 символов) в настоящее время широкое распространение получил новый международ-

ный стандарт Юникод (Unicode), который отводит на хранение каждого символа 16 бит (2 байта) - с его помощью можно закодировать $2^{16} = 65536$ различных символов. Юникод включает в себя практически все современные письменности, а также математические, музыкальные и прочие символы. При этом диапазон интернациональной части кодовой таблицы (ASCII) полностью сохранен.

2.8. Кодирование графических изображений

Создавать и хранить графические объекты в компьютере можно как растровые или как векторные изображения: для каждого из них используется свой способ кодирования.



Растровое графическое изображение - изображение, сформированное из пикселей, каждый из которых может принимать некоторый цветовой оттенок. **Пиксел** (picture element) - минимальный элемент, из которых состоит растровое изображение. Каждый пиксел характеризуется цветом и координатами. Пиксели, которые чаще всего имеет форму квадрата, выстраиваются в табличном виде и благодаря различному цвету и яркости образуют изображение.

Цвет	Яркость	Красный	Зеленый	Синий
Черный	0	0	0	0
Синий	0	0	0	1
Зеленый	0	0	1	0
Голубой	0	0	1	1
Красный	0	1	0	0
Фиолетовый	0	1	0	1
Коричневый	0	1	1	0
Белый	0	1	1	1
Серый	1	0	0	0
Светло-синий	1	0	0	1
Светло-зеленый	1	0	1	0
Светло-голубой	1	0	1	1
Светло-красный	1	1	0	0
Светло-фиолетовый	1	1	0	1
Желтый	1	1	1	0
Ярко-белый	1	1	1	1

Рассмотрим кодирование цветовой палитры из 16 цветов. Разные цвета и их оттенки получаются за счет наличия или отсутствия трех основных цветов (красного, синего, зеленого) и степени их яркости. Тогда цвет каждого пикселя на экране кодируется 4 битами.

Векторное изображение представляет собой совокупность графических примитивов. Каждый примитив состоит из элементарных участков кривых, параметры которых (координаты узловых точек, радиус кривизны и др.) описываются математическими формулами. Для каждой линии указывается ее тип, толщина и цвет. Кодирование векторных изображений выполняется различными способами в зависимости от прикладной среды. В частности, формулы, описывающие участки кривых, могут кодироваться как обычная буквенно-числовая информация для дальнейшей обработки специальными программами.

2.9. Единицы измерения информации

Вам известно, что длина измеряется в миллиметрах, сантиметрах, метрах, километрах. Граммы, килограммы, центнеры, тонны – единицы измерения массы. Углы измеряются в градусах. Время – в секундах, минутах, часах. Представленная в цифровом виде информация тоже может быть измерена. Единицами измерения информации являются:

- Бит (0 или 1)
- Байт = 8 бит.
- Килобайт 1 Кбайт = 2^{10} байт = 1 024 байт ≈ 10^3 (тысяча) байт.
- Мегабайт 1 Мбайт = 2^{20} байт = 1 024 Кбайт ≈ 10^6 (миллион) байт.
- Гигабайт 1 Гбайт = 2^{30} байт = 1 024 Мбайт ≈ 10^9 (миллиард) байт.
- Терабайт 1 Тбайт = 2^{40}байт = 1 024 Гбайт ≈ 10^{12} (триллион) байт.
- Петабайт 1 Пбайт = 2^{50} байт = 1 024 Тбайт ≈ 10^{15} (квадриллион) байт.

Например, сообщение «ИНФОРМАТИКА» состоит из 11 символов, каждый из которых кодируется цепочкой из 8 нулей и единиц. Следовательно, это слово имеет информационный объем 88 битов, или 11 байтов.

Предположим, на каждой странице вашего учебника помещается 40 строк, в каждой строке – 60 символов. Следовательно, страница учебника имеет информационный объем 2400 байтов, а весь учебник, в котором, предположим, 176 страниц, - 422400 байтов. Чтобы перейти от байтов к килобайтам, разделим это число на 1024, получим примерно 413 Кбайт, что соответствует примерно 0,4 Мбайтам.

Значительно больший информационный объем имеют графические файлы. Так, изображение, состоящее из 400x800 пикселей, каждый из которых кодируется 3 байтами, имеет информационный объем 400*800*3 = 1 440 000 байт ≈ 1406 Кбайт ≈ 1,37 Мбайт.

3. Системная среда Windows

3.1. Графический интерфейс. Роль и структура окон

Базовой и необходимой составляющей программного обеспечения ПК без которой он не сможет работать в принципе и будет являться лишь набором отдельных устройств, является **операционная система**. Операционная система (ОС) обеспечивает совместное функционирование всех устройств компьютера и предоставляет пользователю доступ к его ресурсам.

Операционная система Windows характеризуется **дружественным** (удобным) интерфейсом пользователя. **Интерфейс** – совокупность средств и правил, которые обеспечивают взаимодействие устройств, программ и человека. Интерфейс ОС Windows является **графическим**, т.е. для управления работой ПК пользователю предоставлена совокупность графических объектов. После загрузки ОС становится доступным **Рабочий стол Windows** - экран монитора, на котором могут находиться:

1. ярлыки установленных программ, корзина (временное хранение удаленных файлов и папок, которые при необходимости можно восстановить) и другие объекты по выбору пользователя;
2. панель задач (нижняя строка экрана), на которой размещены:

- кнопка Пуск: запуск главного системного меню, с помощью которого можно получить удобный доступ к аппаратным и программным возможностям ПК. Например, к папке Компьютер (доступ ко всем файлам, находящимся на данном ПК) и папке Сетевое окружение (доступ к компьютерам, которые связаны с данным ПК по сети);
- кнопки приложений и папок, с которыми работает пользователь;
- кнопки часов, переключателя алфавита, регулятора громкости звука и т.п.

Основа интерфейса Windows - **окно** - часть экрана, ограниченная прямоугольной рамкой. Запуск любого объекта приводит к открытию его окна.
В ОС Windows используются различные виды окон: системное окно, окно приложения, окно документа, диалоговое окно, окно сообщения. Эти окна имеют множество управляющих элементов.

Структура окна

1. Заголовок окна: указывает, какое окно открыто.
2. Кнопки управления размером окна:

☐ - полноэкранное окно: окно занимает полностью весь экран.

🗗 - окно занимает часть экрана; в данном режиме можно менять размер окна и перемещать его по экрану.

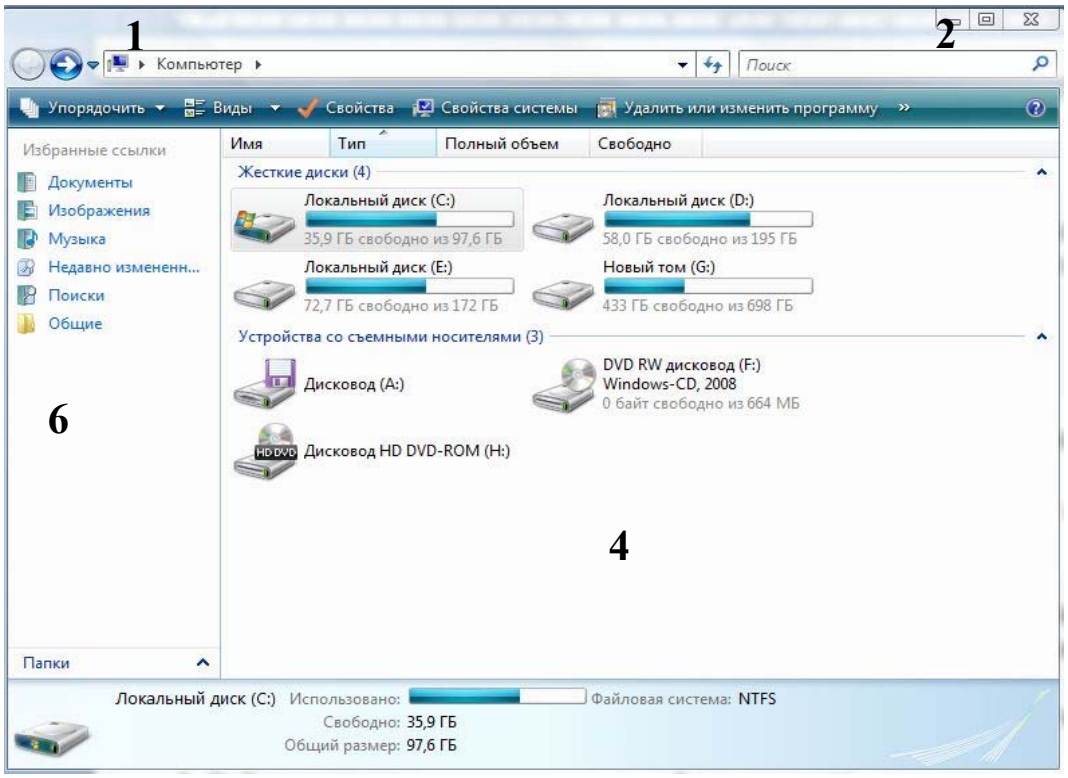
🗕 - свернутое окно: окно убрано с экрана и отображено в виде кнопки на панели задач, щелчок по которой развернет данное окно.

× - закрытие окна (завершение работы с ним).

3. Строка меню, содержащая все операции, предусмотренные для данного

окна (меню - это список команд, из которых можно сделать выбор).
4. Рабочее поле окна, на котором отображается его содержимое.
5. Строка состояния, в которой выводится дополнительная информация о выбранном на рабочем поле объекте.

Кроме этих основных элементов данное окно, например, содержит:
6. Область навигации, которая позволяет быстро переместиться из данной папки (Компьютер) в любую другую.
7. Полосы прокрутки (скроллинг) - прямоугольные полосы, предназначенные для перемещения по окну при помощи мыши, если объекты в рабочем поле не умещаются в видимой части экрана.



3.2. Файл и папка. Их параметры. Действия над файлами и папками

Все программы и данные в памяти компьютера хранятся в виде файлов. Файлы для удобства объединяют в папки. Таким образом, файловая система - это система хранения файлов в папках на логических дисках.

Файл - логически связанная порция информации, имеющая имя и хранящаяся во внешней памяти. Имя файла состоит из двух частей, разделенных точкой: собственно имени файла (до 255 символов) и расширения (как правило, 3-4 символа). Например: explorer.exe; Информатика.doc. Собственно имя файла задается пользователем, а расширение определяется автоматически программой, в которой создается файл.

Расширение является необязательной частью имени файла, однако его

использование очень удобно - оно указывает на тип файла, характеризующий хранящуюся в файле информацию: исполняемый файл - .exe, .com; текстовый файл - .txt, .doc, .docx; графика - .bmp .jpg .gif; аудио - .mp3 .wav .mid; видео - .avi .mpg и т.д. По умолчанию ОС не отображает расширение, но сопоставляет ему определенный графический значок, по которому можно определить тип файла, например:

 - текстовый документ с расширением .doc или .docx.

 - электронная таблица – документ с расширением .xls или .xlsx.

 - база данных – документ с расширением .mdb или .accdb.

Кроме имени файл имеет следующие параметры:
- объем памяти, занимаемый во внешней памяти;
- атрибуты, например «скрытый» (по умолчанию не отображается в ОС) или «системный» (очень важный для работы ОС);
- даты создания и последнего изменения и т.д.

Над файлом можно выполнять набор стандартных действий. Такие действия, как «создать», «сохранить», «закрыть», можно выполнить только в соответствующем приложении. Действия «открыть», «переименовать», «переместить», «копировать», «удалить» можно выполнить в системной среде и в некоторых прикладных программах.

В памяти любого ПК могут храниться сотни тысяч файлов. Поэтому для удобства хранения их группируют в папки. **Папка** (каталог) - это справочник группы файлов, в котором содержатся все данные о них, например местоположение на диске, дата и время создания и др. Имена папок записываются аналогично именам файлов, но без расширения. Часто применяется упрощенная трактовка: папка - это группа файлов. ОС традиционно отображает папки значками желтого цвета: 

Набор действий над папками во многом похож на тот, что применялся к файлам: создать, удалить, переименовать, скопировать, переместить, открыть, закрыть.

3.3. Приложение и документ

В Windows реализован **объектно-ориентированный подход**: при запуске документа автоматически открывается соответствующее ему приложение. **Приложение** - прикладная программа, работающая под Windows. Примеры: графический редактор Paint, текстовый редактор Word. **Документ** - файл, который создан и может быть отредактирован с помощью того

или иного приложения. Примеры документов: рисунок, созданный в графическом редакторе; текст, напечатанный в текстовом редакторе.

3.4. Совместная работа с несколькими программами Способы обмена данными

Windows позволяет запускать одновременно несколько приложений и открывать несколько документов, поэтому она названа **многозадачной средой.** Сколько бы программ не было запущено, работать пользователь может только в одной. Представьте, что вы сидите за обычным письменным столом, и ваша работа включает создание текста, рисунка и выполнение некоторых расчетов. На столе располагаются необходимые инструменты: бумага, ручка, калькулятор, карандаш, линейка и т.д. Однако вы не сможете одновременно рисовать, считать на калькуляторе и писать текст, а будете последовательно выполнять эти действия. Так и при работе на ПК запущено может быть несколько программ, все они могут располагаться на экране, но окно только одной программы будет активным. Выбор задачи из числа всех запущенных в качестве активной называется переключением между задачами.

Есть возможность создать документ, в котором объединены данные, созданные в разных приложениях, - **составной документ**.

Обмен данными в среде Windows осуществляется тремя способами: перетаскиванием объекта мышью, через буфер обмена и по технологии OLE. Ясно, что первым способом удобно пользоваться, когда обмен данными надо произвести между близко расположенными объектами. При втором способе используется **буфер обмена** – область памяти, которая служит для временного хранения данных, предназначенных для обмена. Этот способ годится в том случае, если нет необходимости редактировать вставленный объект. При третьем способе применяется **технология OLE** (связывание и внедрение объектов). Она позволяет вставить объект без связи его с приложением, в котором он был создан (внедрение) или связать его с источником (связывание). Внедренный объект можно редактировать средствами источника, но внесенные изменения не отражаются на исходном файле. При связывании объект OLE находится в документе-источнике, а приложение-приемник содержит информацию о том, где следует искать объект. Поэтому любые изменения исходного файла отражаются и в документе-приемнике.

4. Графические редакторы

4.1. Общая характеристика прикладной среды Особенности растровой и векторной графики

Графический редактор – прикладная программа, предназначенная для создания и редактирования изображений.

Растровое графическое изображение – изображение, сформированное из пикселей, каждый из которых может принимать некоторый цветовой оттенок. Подробнее – в разделе 2.8.

Бесспорным достоинством растрового изображения является его высокое качество, реалистичность, многообразие и плавность переходов цветовых оттенков. Поэтому все фото и сканированные изображения являются растровыми.

К недостаткам можно отнести большой объем занимаемой памяти (изображение состоит из огромного количества пикселей, для каждого из которых надо хранить информацию о его цвете и координатах) и чувствительность к масштабируемости (при уменьшении размера растровой картинки несколько соседних точек преобразуются в одну, поэтому теряется различимость мелких деталей изображения, а при увеличении изображения каждая точка преобразовывается в несколько, и цвет новообразованных пикселей подбирается из расчета цветов соседних пикселей - это приводит к искажению изображения).

Векторное графическое изображение - изображение, сформированное из графических примитивов. Графический примитив - это геометрический объект (отрезок, прямоугольник, эллипс и др.), который хранится в памяти в виде математических формул и числовых параметров, содержащих информацию о следующих его свойствах:

- Координаты фигуры. Например, отрезок задается координатами первой и последней точек, а окружность - координатами центра и радиусом.
- Цвет фигуры и закраски для области внутри нее.
- Параметры, определяющие форму кривизны и толщину линий фигуры.
- Параметры группы фигур, сгруппированных в единый объект.

Достоинства:

- Масштабирование без искажений. Поэтому векторные изображения являются оптимальными для хранения схем, чертежей, а также трехмерного моделирования: программы AutoCAD, Компас, Corel Draw, 3D Studio.
- Легкость анимации (Flash.
- Небольшой объем занимаемой памяти: запись команды фигуры вместо кодирования всех ее точек существенно сокращает размер

записываемого файла.

Недостатки:

- Все аппаратные устройства ввода-вывода графической информации требуют представления изображения в растровом виде. Поэтому для ввода-вывода векторного изображения требуется его преобразование в растровое.
- Векторные изображения не используются для хранения цифровых фотографий или сканированных изображений, т.к. векторная картинка человеческим глазом всегда воспринимается как искусственно созданный рисунок.

4.2. Объекты растровой графики и действия над ними

Графический редактор Paint относится к простейшим растровым графическим редакторам и является стандартным приложением Windows. Файлы имеют расширение bmp.

Процесс работы с рисунком условно разбивают на этапы: создание, редактирование, сохранение, печать. Основными объектами среды растрового редактора являются: пиксель (можно изменть его цвет или стереть), графический примитив (характеризуется цветом, контуром, заливкой, формой, местоположением, пропорциями), фрагмент (имеет форму и размер, его можно перемещать, масштабировать, копировать, вставлять, поворачивать, отражать) и сам рисунок (характеризуется цветностью и размером).

4.3. Назначение и настройки инструментов растровой графики

Основные инструменты для создания рисунка: карандаш, кисть, линия, кривая, прямоугольник, скругленный прямоугольник, многоугольник, эллипс, распылитель, заливка, надпись. Для большинства инструментов возможен выбор их параметров. Например, для линии настраивается ее толщина, для кисти – форма. Замкнутые объекты могут иметь разный тип заливки. Инструмент «надпись» для настройки имеет специальную панель атрибутов текста. Выбор цвета фигур или заливки осуществляется с помощью палитры или инструментом «выбор цвета».

4.4. Объекты векторной графики, их свойства

Объект	Свойства	Действия
Элементарный объект (примитив)	Форма, размер, контур, заливка, пропорции, расположение	Прорисовать, отредактировать, вырезать, скопировать, вставить, сгруппировать с другими, отменить, удалить, переместить, отразить, повернуть, сменить план

Сгруппированный рисунок	Сумма свойств входящих объектов, размер, расположение	Те же, что с элементарными объектами, кроме редактирования. Возможно частичное редактирование, при этом новые свойства присваиваются всем входящим объектам. Разгруппировать.

Для выполнения действий над векторными объектами их необходимо выделить.

4.5. Специфические действия над векторными объектами

Выделение нескольких объектов одновременно

Действие	Технология выполнения
Выборочное выделение объектов	Удерживая клавишу Shift, щелкать мышью на нужных объектах.
Выделение объектов, расположенных рядом	Активизировать инструмент Выбор объектов и обвести мышью нужные объекты

Редактирование векторного объекта – изменение его свойств

Действие	Технология выполнения
Изменение цвета контура объекта	Выбрать инструмент Цвет линий и цвет
Изменение толщины контура объекта	Выбрать инструмент Тип линий и толщину линии
Изменение цвета заливки объекта	Выбрать инструмент Цвет заливки и выбрать цвет
Изменение рисунка заливки объекта	Выбрать инструмент Цвет заливки и выбрать способ заливки (узор, текстура, градиент)
Нанесение надписи внутри объекта	В контекстном меню выбрать команду Добавить текст и напечатать текст на клавиатуре
Смена плана объекта	Выбрать команду Действия – Порядок, выбрать соответствующий план (на передний план, на задний план, переместить вперед, переместить назад, поместить перед текстом, поместить за текстом)
Группировка нескольких объектов в один	Выделить несколько объектов и выполнить команду Действия - Группировать
Разгруппировка сгруппированных ранее объектов	Выделить объект и выполнить команду Действия - Разгуппировать

5. Основы алгоритмизации и программирования.

5.1 Алгоритм. Исполнитель

Каждый человек в повседневной жизни решает огромное количество задач самой разной сложности. Некоторые задачи столь просты и привычны, что мы решаем их автоматически. Например, «купить хлеб», «закрыть дверь на ключ». Другие же, напротив, требуют усилий для поиска решения и достижения цели. Но решение даже самой простой задачи осуществляется за несколько последовательных шагов. Например, процесс покупки хлеба можно представить так: 1) взять деньги, 2) пойти в магазин, 3) выбрать хлеб, 4) оплатить покупку, 5) принести хлеб домой.

Алгоритм – это конечная последовательность точно определенных действий, приводящих к решению поставленной задачи.

При этом для алгоритма важен не только набор действий, но и то. В каком порядке они выполняются. Например, попробуем переставить в известном вам алгоритме нахождения НОК нескольких натуральных чисел четвертое действие на второе место. Вот, что получим: 1) разложить числа на простые множители, 2) найти произведение получившихся множителей, 3) выписать множители, входящие в разложение одного из чисел, 4) дописать к ним недостающие множители из разложений остальных чисел. Эту последовательность действий тоже можно исполнить, но к решению задачи она не приведет.

Первенство в разработке алгоритмов принадлежит человеку. Исполняют алгоритмы люди и всевозможные устройства – компьютеры, роботы, станки, спутники, бытовая техника и т.д.

Исполнитель - устройство, способное понимать и исполнять команды алгоритма. Исполнитель обладает следующими свойствами:

Система команд исполнителя (СКИ) - набор команд, который может выполнять данный исполнитель.

Среда исполнителя - совокупность объектов, над которыми исполнитель может совершать действия.

5.2 Свойства алгоритмов

Любой алгоритм должен отвечать следующим требованиям:

1. **Понятность** - в алгоритме должны содержаться только те команды, которые входят в его СКИ. Чем меньшей СКИ владеет исполнитель, тем сложнее становится описание алгоритма для него. Например, для исполнителя-десятиклассника алгоритм решения уравнения из курса алгебры 8 класса должен выглядеть так: решить уравнение, потом сообщить результат, - так как в его СКИ должно входить умение решать подобные уравнения. А для исполнителя-семиклассника потребуется подробно расписать все этапы решения данного уравнения, так как его СКИ еще не включает подобные знания.

2. **Дискретность** - дискретное (пошаговое) выполнение команд алгоритма с точной фиксацией выполнения одной команды и начала выполнения следующей. Например, для объяснения человеку алгоритма нахождения какого-либо здания в незнакомом городе важно пошагово расписать последовательность его действий:

1. Проехать две остановки на автобусе.
2. Перейти через подземный переход.
3. Пройти 150 метров прямо.

Такой дискретный алгоритм гораздо надежнее, нежели сложносочиненный путанный рассказ о дороге с упоминанием красот любимого города, что вызовет лишь путаницу в голове собеседника.

3. **Результативность** - алгоритм должен обязательно привести к необходимому конечному результату. Например, алгоритм перехода улицы (1. Подойти к пешеходному перекрестку. 2. Дождаться зеленого света светофора. 3. Перейти улицу.) может не сработать, так как не предусматривает возможности отсутствия или поломки светофора. Усовершенствуем его:

1. Подойти к пешеходному перекрестку.
2. Если светофор работает, то дождаться зеленого света и перейти улицу.
3. Если светофор не работает, то перейти улицу по правилам для нерегулируемых пешеходных переходов.

4. **Детерминированность** – каждый шаг алгоритма должен быть четким и однозначным: тогда выполнение алгоритма для исполнителя не потребует никаких дополнительных указаний. Например, шаг 3 предыдущего алгоритма содержит инструкцию для школьника «перейти улицу по правилам для нерегулируемых пешеходных переходов», которые он может не знать, следовательно, эти инструкции также должны быть включены в алгоритм.

5. **Массовость** - возможность применения данного алгоритма для решения многих задач одного типа. Какой смысл был бы в изучении алгоритма решения некоторого уравнения, если с его помощью решалось бы только данное уравнение?

5.3 Формы записи алгоритмов. Представление алгоритма в виде блок-схемы

Существует достаточно большое количество форм представления алгоритмов, основные из которых перечислены ниже.

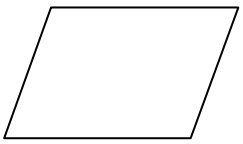
Словесная форма. Объясняя другому человеку решение некоторой проблемы, мы описываем ее алгоритм на естественном языке. Но каждый человек делает это по-своему, поэтому один и тот же алгоритм может быть описан разными людьми по-разному, а отсутствие четких стандартов в словесном выражении алгоритма часто приводит к недопониманию.

Программа. Если алгоритм предназначен для исполнения техническим устройством, например роботом-манипулятором или компьютером, то он представляется в виде программы на одном из языков программирования.

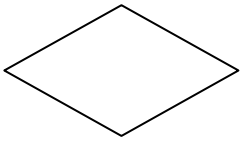
Графическая форма является наиболее наглядной для человека. Например, на упаковках продуктов быстрого приготовления и в инструкциях для бытовой техники алгоритм их использования часто приводится в рисунках.

Но особое место в информатике занимает такая графическая форма представления алгоритма как **блок-схема** – способ записи алгоритма с помощью стандартного набора (более 20-и) графических блоков, перечень и условные обозначения которых жестко зафиксированы ГОСТом (государственным стандартом). Основными графическими блоками, которые мы будем использовать, являются:

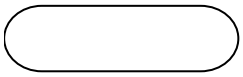
- операционный блок



- блок ввода и вывода информации



- блок проверки условия



- блок начала и конца алгоритма

5.4. Понятие программы

Программа – упорядоченная последовательность команд (инструкций), необходимых ПК для решения поставленной задачи.

Программирование – процесс составления программы для компьютера.

Программы так же, как и алгоритмы, обладают свойствами дискретности и детерминированности. Верно составленные программы должны быть конечны и правильны. Хорошие программы обладают свойством массовости.

Программа хранится в памяти компьютера. При запуске программы ПК выполняет команды в том порядке, в котором они записаны.

Вспомним основные правила оформления программы в среде ЛогоМиры:

- программа имеет заголовок, в котором указано ее имя, отражающее смысл программы;

- после заголовка с новой строки идет тело программы, в котором на языке программирования записан алгоритм;

- программа должна иметь признак завершения программы также записанный на отдельной строке;

- для пояснения алгоритма в программе удобно использовать комментарий, который должен начинаться с символа «;».

Вот пример программы рисования цифры «0» в Логомирах, в которой применены комментарии:

```
;Заголовок – имя программы, записанное непрерывной последовательностью символов: цифра_0
это цифра_0
;Начало тела программы
по вп 80 пр 90 вп 40 пр 90 вп 80 пр 90 вп 40 пр 90 пп
;Конец тела программы
конец
```

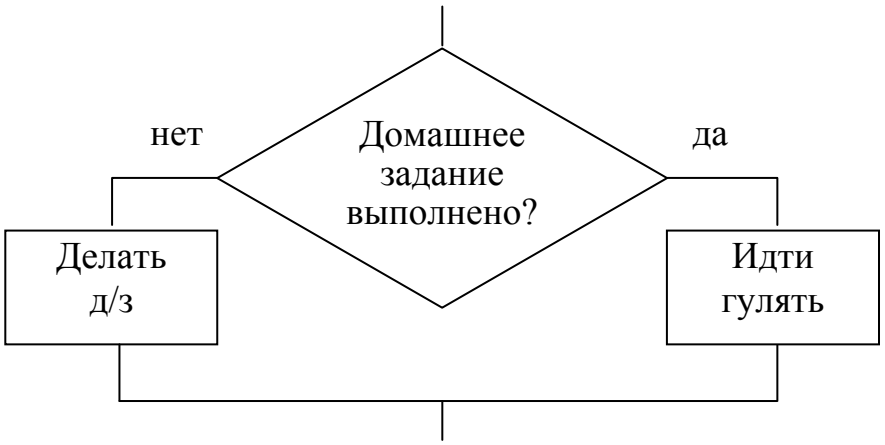
5.5. Основные алгоритмические структуры

Внутри алгоритмов можно выделить группы шагов, отличающиеся внутренней структурой – алгоритмические конструкции. К основным алгоритмическим конструкциям относятся следование, ветвление и цикл.

1. **Линейная** последовательность (следование) - группа шагов алгоритма, выполняемых последовательно друг за другом без каких-либо условий. Если весь алгоритм представляет собой линейную последовательность шагов, то его называют линейным.

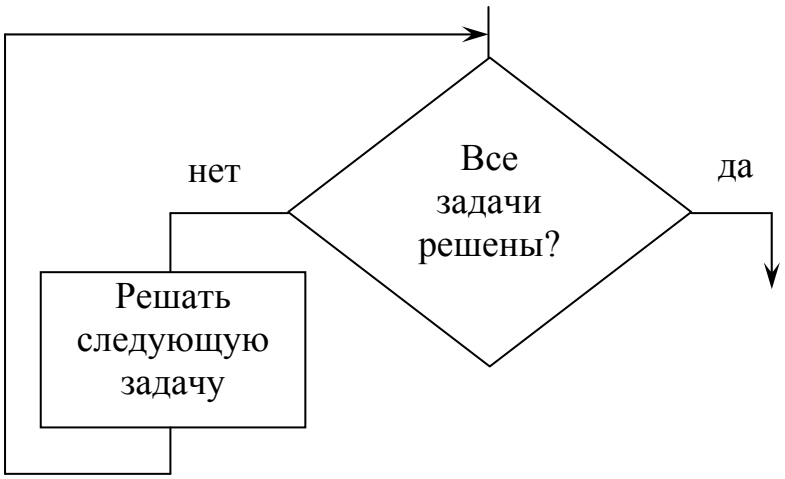
Пример: упрощенный алгоритм перехода улицы.

2. **Ветвление** – алгоритмическая конструкция, в которой выполнение того или иного действия зависит от выполнения некоторого условия. Пример: фрагмент алгоритма действий учащегося в послеурочное время.



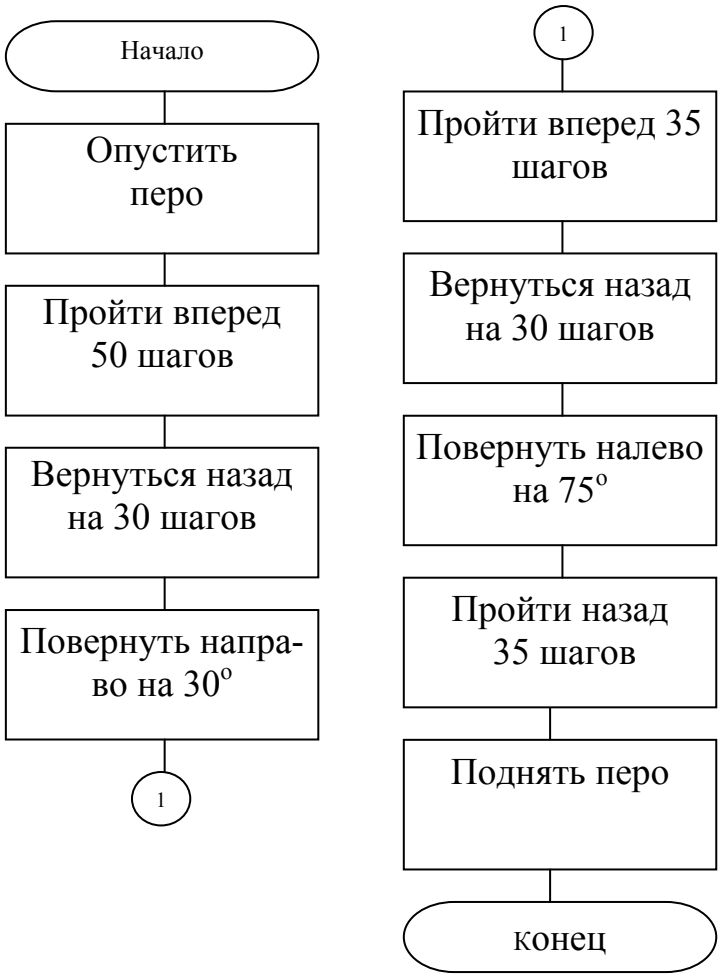
3. **Цикл** - алгоритмическая конструкция, в которой многократно выполняется одна и та же последовательность действий, называемая телом цикла.

Пример: фрагмент алгоритма действий учащегося на контрольной работе.



5.6. Пример линейного алгоритма и программы

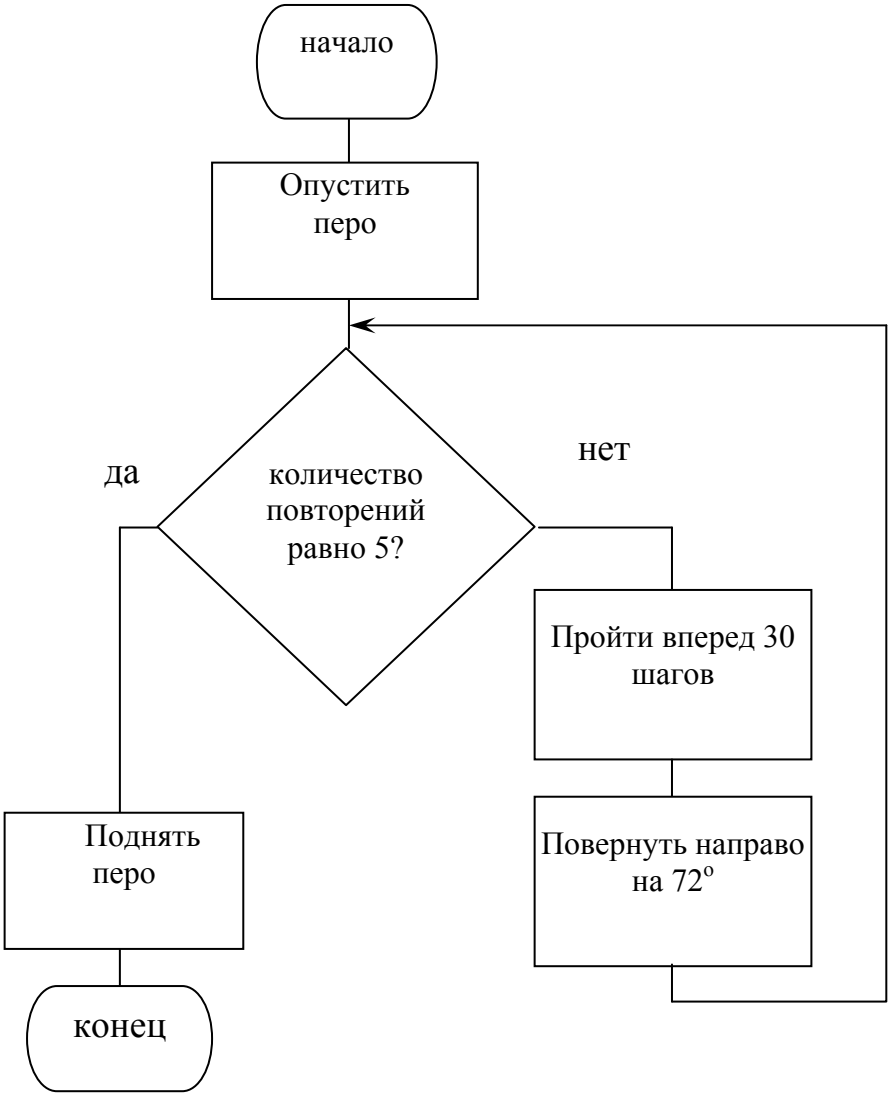
Блок-схема и программа рисования буквы «К» в ЛогоМирах (высота символа равна 50 шагам Черепашки).




```
это буква_К
по вп 50 нд 30 пр 30 вп 35 нд 30
лв 75 нд 35 пп
конец
```

5.7. Пример циклического алгоритма и программы

Построение правильных многоугольников – пример реализации циклического алгоритма. В языке Лого для записи циклического алгоритма применяется команда ***повтори.*** В этой команде два параметра: первый задает количество повторений (**условие цикла**), второй – список команд. Которые должны повторяться (**тело цикла**).

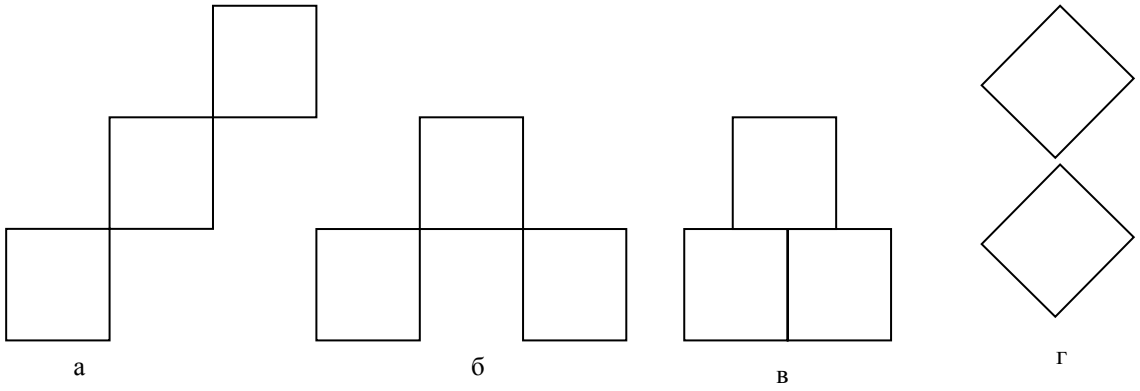


В данной задаче строится правильный пятиугольник со стороной 30 шагов Черепашки.

```
это прав5уг-к
по
повтори 5 [вп 30 пр 72]
пп
конец
```

5.8. Представление о процедуре и программном модуле

Рассмотрите объекты, изображенные на рисунке. Все они составлены из одинаковых квадратов. Разумно в алгоритмах построения этих изображений использовать предварительно созданную программу **квадрат,** которая будет в дальнейшем вызываться одноименной командой. В средах программирования вспомогательные программы обычно называют **процедурами.**

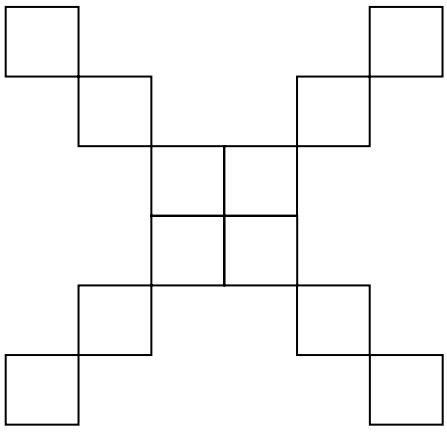


*это **квадрат***
по повтори 4 [вп 30 пр 90] пп
конец

Назовем рисунок под буквой а «лесенка». Составим программу рисования лесенки.

*это **лесенка***
повтори 3 [квадрат вп 30 пр 90 вп 30 лв 90]
конец

Различают **встроенные и пользовательские процедуры**. Все команды ЛОГО обращаются к встроенным процедурам. Вспомогательные программы, написанные вами и помещенные на лист программ (процедур), являются пользовательскими процедурами. Программа **квадрат** – пользовательская процедура, применяя которую вы должны точно представлять, в каком состоянии находится Черепашка перед ее выполнением и в каком состоянии она окажется после ее выполнения. От этого зависит выполнение всей программы. Поэтому при создании процедуры важно соблюдать **правило сохранения начального состояния исполнителя: процедура перед своим завершением должна восстановить исходные значения параметров исполнителя.**



А теперь, предположим, нам надо изобразить объект, являющийся композицией из нескольких лесенок. Назовем его «крест».

Объект, который составлен из более простых объектов и может использоваться для создания более сложных объектов, называется **модулем.** Программу, которая вызывает пользовательские процедуры, называют **программным модулем**. Программный модуль применяется для создания более сложных объектов, в отличие от процедуры, которая используется для создания элементарных объектов. В нашем случае программным модулем можно считать программу рисования лесенки, созданную с помощью

процедуры квадрат, и использовать ее для конструирования всевозможных композиций.

это ***крест***
повтори 4 [лесенка нд 90 лв 90 вп 90]
конец

Одни и те же процедуры могут применяться в разных модулях. В свою очередь, модули могут применяться как для создания отдельных объектов, так и для создания более сложных модулей. Такой подход к программированию называется **процедурным**.

5.9. Пример разветвляющегося алгоритма и программы

В среде ЛОГО есть команды, с помощью которых можно выбрать нужные действия, проанализировав условие:

- команда **если** позволяет реализовать неполную форму разветвляющегося алгоритма;
- команда **если_иначе** позволяет реализовать полную форму.

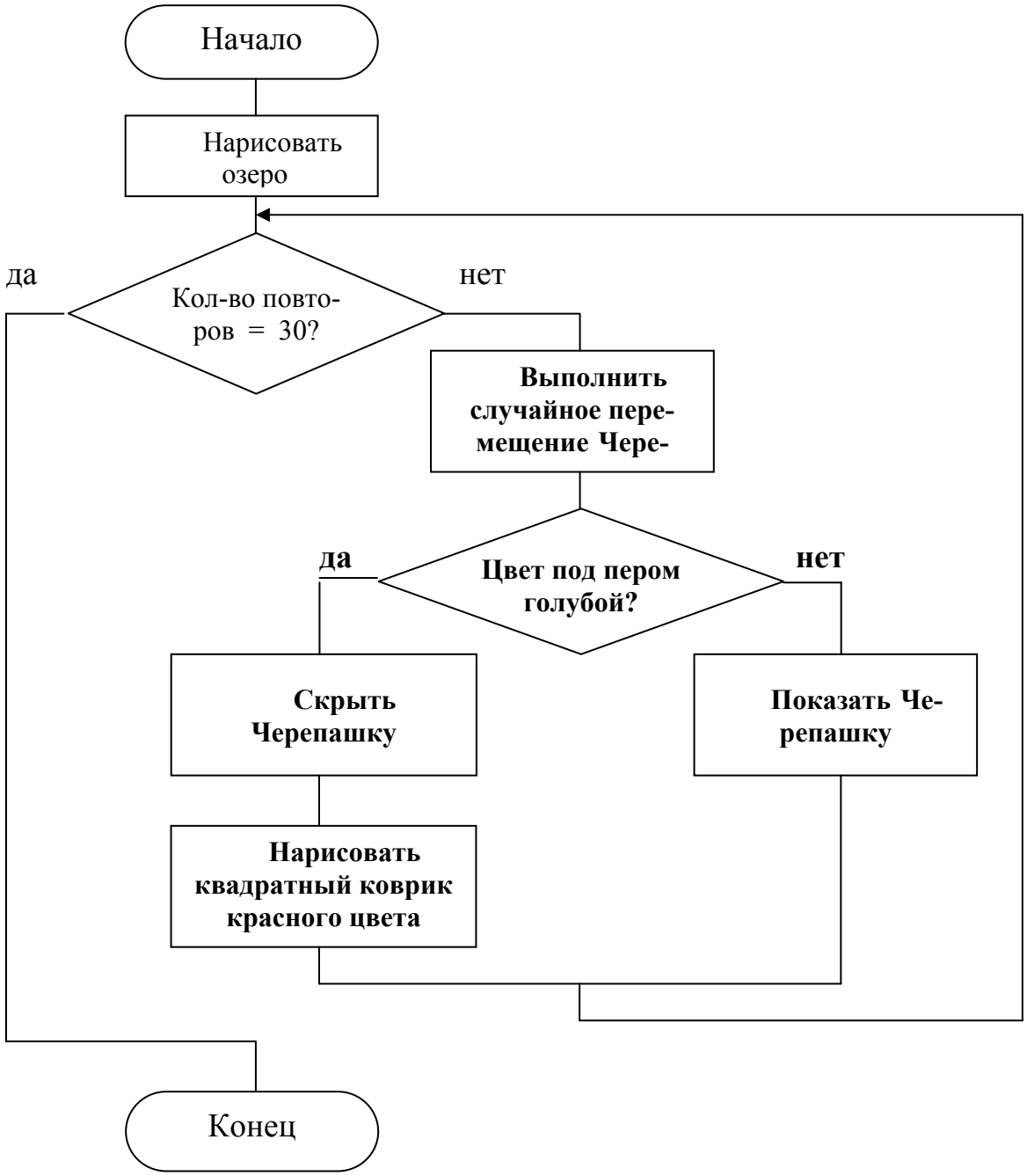
Варианты возможных действий задаются **списком команд,** в котором команды отделяются друг от друга пробелом.

если **условие [список команд] -** формат команды неполной формы; состоит из одного ключевого слова «если» и двух входных параметров. Условие представляет собой одну операцию сравнения или логическое выражение. Список команд содержит те команды, которые надо выполнить, если условие истинно; если условие ложно, то команды не выполняются.

если_иначе **условие [список команд 1] [список команд 2] –** формат команды полной формы; состоит из ключевого слова «если_иначе» и трех входных параметров. Первый список содержит те команды, которые надо выполнить, если условие истинно, а второй – когда условие ложно.

Предположим, наша задача состоит в том, чтобы организовать следующее поведение Черепашки: двигаясь по экрану случайным образом, Черепашка может оказаться в круге голубого цвета, называемом «озером», в этом случае она должна стать невидимой и нарисовать квадрат красного цвета; если Черепашка оказывается вне озера, то она должна быть видимой, но не оставлять никаких следов.

В алгоритме, представленном блок-схемой, виден (выделен полужирным начертанием) фрагмент алгоритма разветвляющегося типа полного формата.



это озеро
нц 85
по повтори 360 [вп 3 пр 1]
пр 45 вп 5 крась
пп
конец

это игра
озеро
повтори 30 [пр случайный 360 нд случайный 200
если_иначе цп = 85 [сч нц 15 по повтори 4 [вп 30 пр 90] пп жди 10 нц 85][пч жди 10]]
конец

5.10. Параметры

До сих пор в программах для создания графических объектов длина отрезка, угол поворота, цвет и т.д. задавались определенными числами. Однако, часто возникает необходимость построить такой же объект, но с другими значениями параметров.

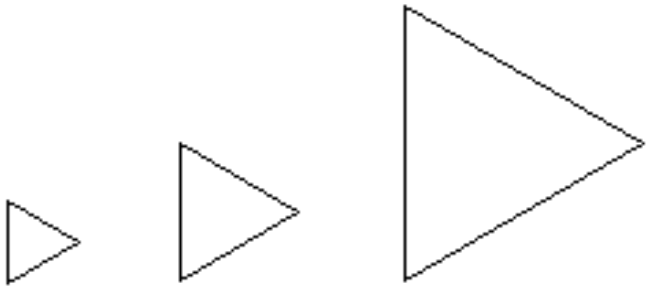
Рассмотрим, как это сделать, на примере построения правильных многоугольников. Сначала ответим на вопрос: какими параметрами отличаются друг от друга правильные многоугольники? Таких параметров два: длина стороны и количество сторон правильного многоугольника (второй параметр однозначно привязан к величине угла правильного многоугольника).

Попробуем сначала построить равносторонние треугольники с разными длинами сторон. Длину стороны, как меняющуюся величину, обозначим буквой **а.**

```
это прав_треуг а
по
повтори 3 [вп :а пр 120]
пп
конец
```

В заголовке программы появилось **имя параметра** (**а**). Это имя используется в команде **вп :а** и показывает, что конкретные данные для этой команды будут указаны в команде вызова этой процедуры. Здесь вместо указания конкретного числа используется так называемый формальный параметр.

Формальный параметр – это параметр, значение которого в процедуре может быть любым (но, конечно, допустимым). В языке ЛОГО двоеточие перед именем параметра показывает, что в данной команде используется значение параметра, которое будет задано пользователем при вызове этой процедуры. Вызывая процедуру с параметром, необходимо после ее имени обязательно задать **фактическое значение этого параметра**. Например, у нас на рисунке фактические значения параметра были равны 30, 50 и 100 шагам Черепашки, а команды вызова процедур, соответственно, выглядели так: *прав_треуг 30*, *прав_треуг 50* и *прав_треуг 100.*

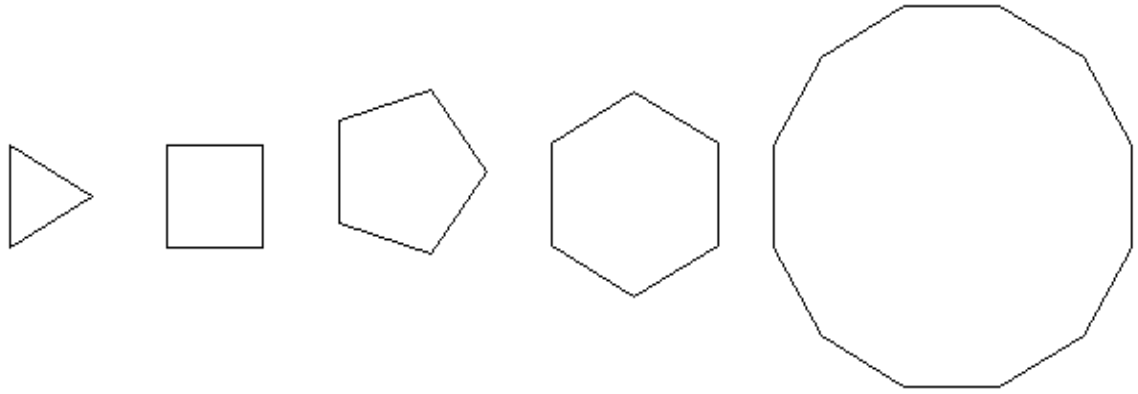


Аналогично рассуждая, можно построить правильные многоугольники с одинаковыми длинами сторон (например, 50), но с различным их количеством. Тогда меняющимся параметром будет количество сторон, обозначенное у нас буквой **n.** Значит, в заголовке программы появляется имя параметра **n,** которое используется в команде **пр 360 / :n.**

Программа рисования таких многоугольников будет выглядеть так:

```
это прав_мн_угол  n
по
повтори :n [вп 50 пр 360 / :n]
пп
конец
```

Указывая фактические значения параметра **n** 3,4,5,6 и 12, можно изобразить следующие фигуры:

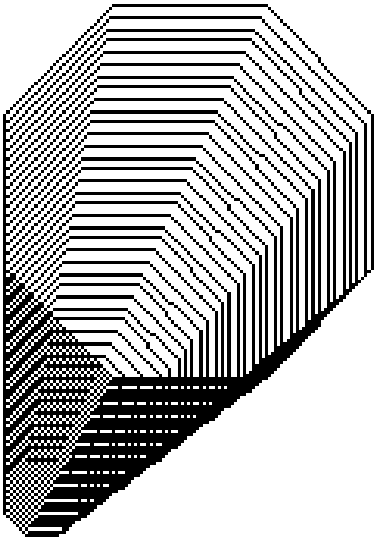


Для того, чтобы можно было использовать программу для построения любого правильного многоугольника, надо указать оба параметра:

```
это прав_мн_угол a n
по
повтори :n [вп :a пр 360 / :n]
пп
конец
```

5.11. Переменные. Переменная в алгоритме

На рисунке изображен объект, который условно можно назвать «Тоннель». Он состоит из 40 восьмиугольников, причем сторона каждого следующего на 1 пиксел длиннее предыдущего. Обозначим длину стороны каждого из восьмиугольников буквой **s**. В данной задаче значение **s** надо увеличивать на каждом шаге прохождения цикла. В программировании для таких ситуаций используют объект **переменная.**



Переменная – это объект в программе, имеющий имя и изменяемое значение. Для каждой переменной выделено определенное место в памяти компьютера.

В любой среде программирования различаются имя и значение переменной. **Имя переменной** показывает, в каком месте памяти компьютера хранится значение переменной. **Значение переменной** считывается из указанного ее именем места памяти. Значение переменной применяется как фактический параметр в команде.

Переменную можно создать, т.е. указать ей имя и присвоить ей значение. В среде ЛОГО для создания переменно применяется команда **пусть.** Эта же команда изменяет значение уже существующей переменной. В языке ЛОГО явно показывается различие между именем и значением переменной: имя предваряется кавычкой (*"*), а значение – двоеточием (:).

```
это тоннель
по пусть "s 10
повтори 40 [прав_мн_угол :s 8  пусть "s :s + 1 вп 2]
пп
конец
```

В программе построения тоннеля команда ***пусть "s 10*** задает длину стороны самого маленького восьмиугольника (10 шагов Черепашки), а команда ***пусть "s :s + 1*** изменяет значение переменной, увеличивая его на 1. В этом случае из памяти считывается старое (текущее) значение переменной ***s,*** к нему прибавляется 1, а затем новое значение записывается в то же место памяти.