«Число – есть сущность всех вещей, а организация Вселенной в ее определениях представляет собой вообще гармоническую систему чисел и их отношений»

Пифагор

Лекция 1: Введение в ИТ

Разработал: Павлов А.Н.

Сокращения

АВМ – аналоговая вычислительная машина

ВТ – вычислительная техника

ИТ – информационные технологии

ОС – операционная система

САПР – система автоматизированного проектирования

ЭВМ – электронная вычислительная машина

ЯП – язык программирования

Содержание курса

Основы программирования

- Введение в ИТ
- Программирование на языке Pascal
- Знакомство с VBScript и VBA
- Знакомство с математическим пакетом MathCAD
- Знакомство с табличным процессором Microsoft Excel

Введение в современные САПР

- Основы проектирования
- Виды САПР
- CAD/CAM/CAE/CAPP/PDM системы
- Специальное оборудование
- Выбор САПР

Выписка из учебного плана

№ п/п	Наименование	Учебная нагрузка, з.е.т./ч	Лк	Лб	СРС	Экз.	1 год	
							II	III
1	ИТ в самолетостроении 1	4/144	32	32	80	1	*	
2	ИТ в самолетостроении 2	4/144		64	80			*

Пояснения:
Лк – лекционные занятия
Лб – лабораторные занятия
СРС – самостоятельная работа студентов
З.е.т. – зачетная единица трудоемкости

Запланировано **12** лабораторных работ
по разделу «Основы программирования».

Учебники

1. Грошев А.С. Информатика: Учебник для вузов. – Архангельск, Арханг. гос. техн. ун-т, 2010. – 470 с.
2. Леонтьев В.П. Новейший самоучитель. Компьютер + Интернет 2013. – М.: ОЛМА Медиа Групп, 2013. – 640 с.
3. Малюх В.Н. Введение в современные САПР: Курс лекций. – М.: ДМК Пресс, 2010. – 192 с.
4. Малинина Л.А. Основы информатики: Учебник для вузов. – Ростов на Дону: Феникс, 2006. – 352 с.
5. Ли К. Основы САПР (CAD/CAM/CAE). – СПб: Питер, 2004. – 560 с.

Методички

1. Еремин О.Ф. Методическое пособие по программированию на языке Pascal ABC. – Моздок, 2009. – 49 с.

2. Лабораторный практикум по программированию на языке Паскаль: Учебное пособие. / Под ред. Найхановой Л.В. и Бильгаевой Н.Ц. – Улан-Удэ, 2004. – 176 с.

Программное обеспечение

1. Pascal ABC Net, версия 3.0
2. MathCAD 15

Содержание лекции

1. Понятие информации, ее виды и свойства
2. Единицы измерения информации
3. Системы счисления
4. Кодирование информации (текстовой, числовой, изображений, звука, видео)
5. Файлы, файловая структура и файловые системы
6. Алгоритмы
7. Этапы разработки программы
8. Языки программирования
9. Принципы программирования

Понятие информации. Ее виды и свойства

Определение

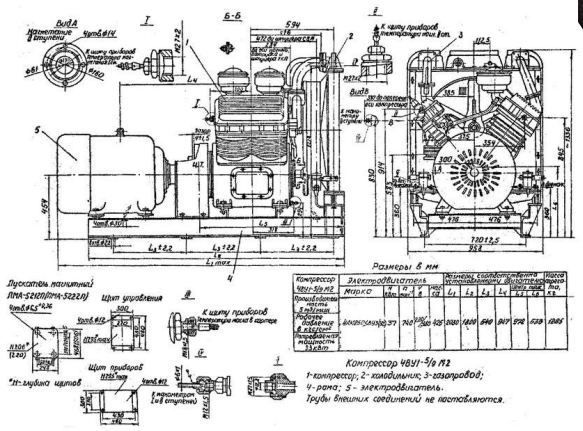
Информация - сведения, воспринимаемые человеком или специальными устройствами как отражение фактов материального мира в процессе коммуникации.

Определений информации существует множество, в силу широты этого понятия нет и не может быть строгого и универсального определения.

С точки зрения информатики под информацией понимают **данные**, т.е. информация в *формализованном* виде, пригодном для *передачи*, *хранения* и *обработки*.

Виды информации

Графическая – исторически первый вид информации, для которого был реализован способ хранения, сначала в виде наскальных рисунков, а позднее в виде картин, фотографий, схем, чертежей на бумаге, холсте и т.д.



Виды информации

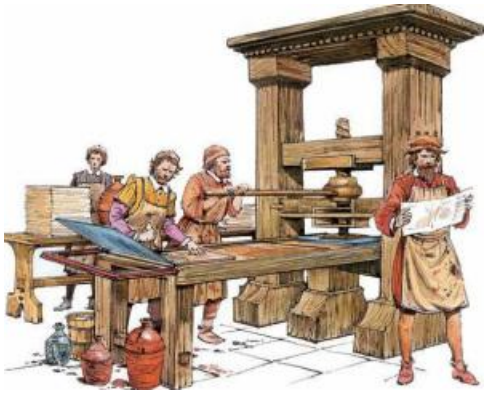
Звуковая – мир вокруг нас полон звуков. Задача их хранения и тиражирования была решена с изобретением Томасом Эдисоном фонографа в 1877 г. .

Разновидностью звуковой является **музыкальная** информация. Для нее был изобретен способ кодирования с использованием специальных символов, что делает возможным ее хранение и передачу.

Виды информации

Текстовая – способ кодирования речи человека буквами, причем разные народы имеют различные наборы букв для отображения речи.

Особенно большое значение этот способ приобрел после изобретения книгопечатания.



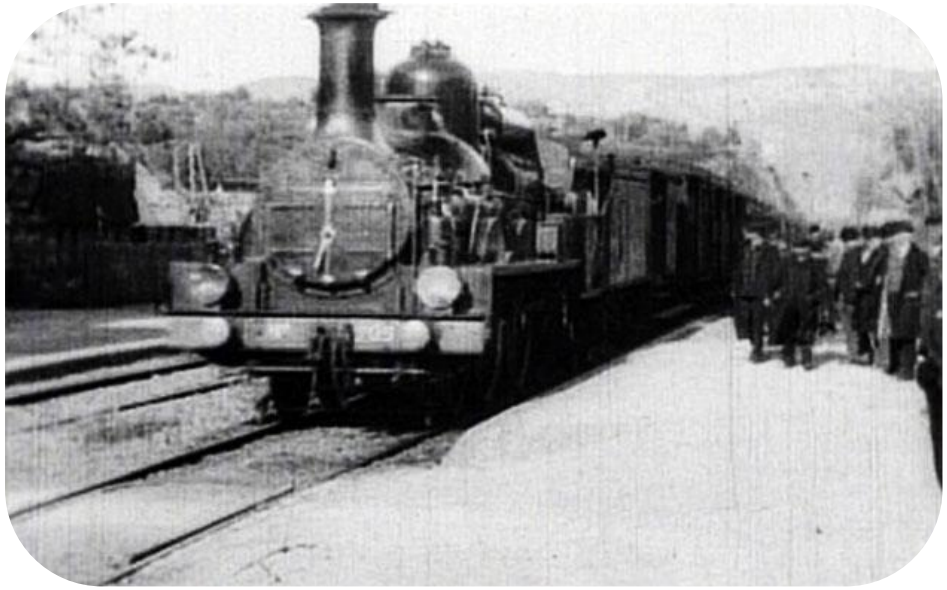
Числовая – количественная мера объектов и их свойств в окружающем мире.

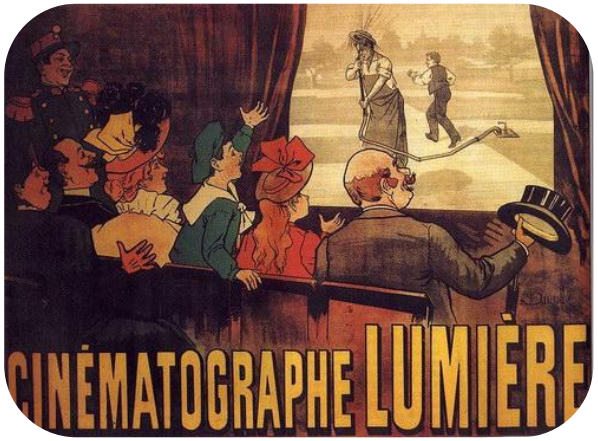
Особенно большое значение приобрела с развитием торговли, экономики и денежного обмена.

Для ее отображения также используется метод кодирования специальными символами (цифрами).

Виды информации

Видеоинформация – способ сохранения движущихся картин окружающего мира, появившийся с изобретением кино братьями Люмьер в 1894 г.



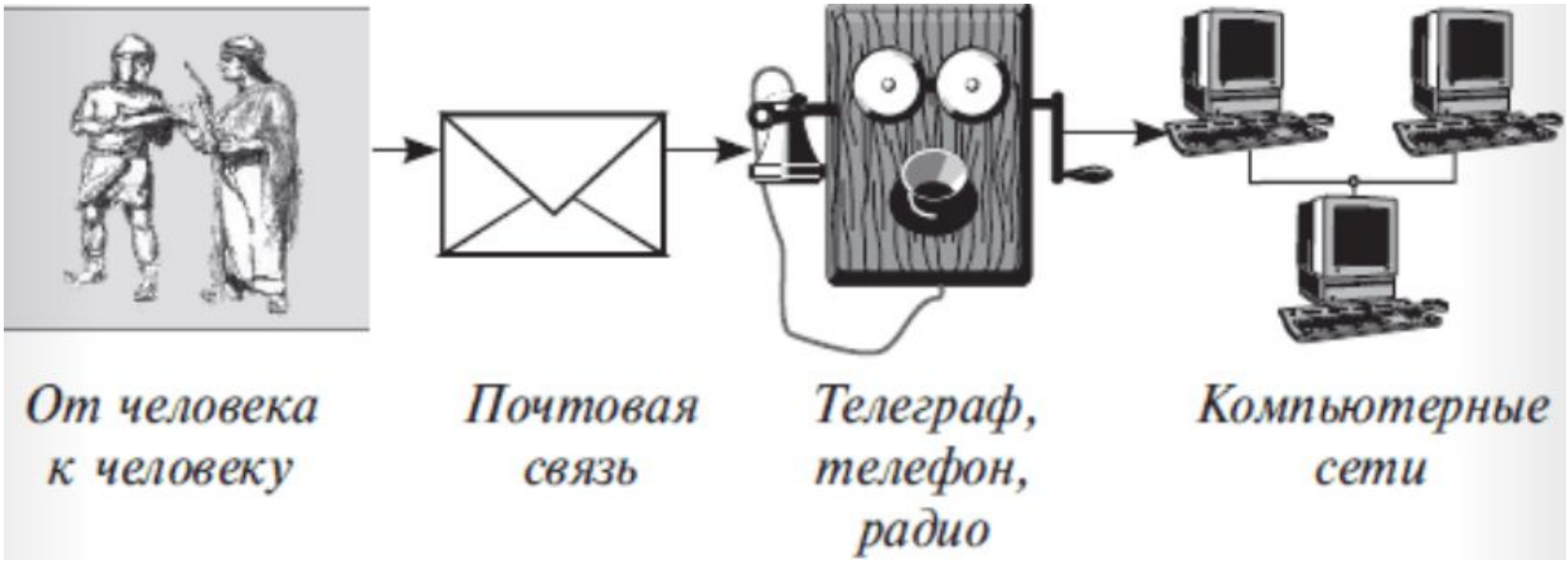


Виды информации

Существуют виды информации, для которых до сих пор не изобретено способов их кодирования и хранения – это **тактильная** информация, передаваемая ощущениями от прикосновений, и **органолептическая**, передаваемая запахами и вкусами.

Развитие способов обработки информации

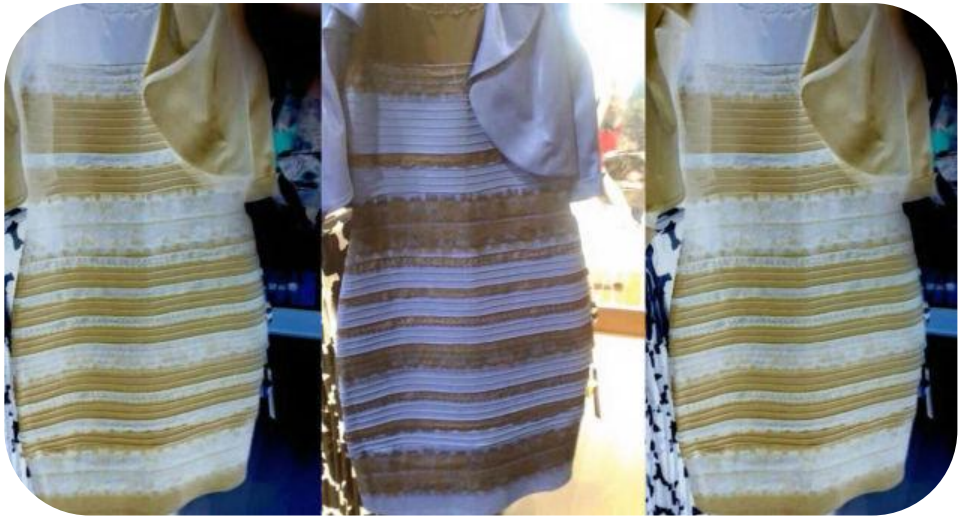
Развитие способов передачи информации



Свойства информации

- **Объективность.**

Информация объективна, если она не зависит от методов ее фиксации, чьего-либо мнения, суждения.



Пример: Сообщение «На улице тепло» несет субъективную информацию, а сообщение «На улице 22°С» - объективную, но зависящую от погрешности средства измерения.

Свойства информации

- **Достоверность.**

Информация достоверна, если она отражает истинное положение дел.

Объективная информация всегда достоверна, но достоверная информация может быть как объективной, так и субъективной.

Достоверная информация помогает принять правильное решение.

Недостоверной информация может быть по следующим причинам:

- преднамеренное искажение (дезинформация) или непреднамеренное искажение субъективного свойства;
- искажение в результате воздействия помех и недостаточно точных средств измерений.

Свойства информации

- **Полнота.**

  Информацию можно назвать полной, если ее достаточно для понимания и принятия решений.

  Неполная информация может привести к ошибочному мнению или решению.

- **Точность** определяется степенью ее близости к реальному состоянию объекта, процесса, явления (погрешностью средства измерения).

Свойства информации

- **Актуальность** – важность для настоящего времени, актуальность, насущность. Иногда только вовремя полученная информация может быть полезна.

- **Полезность** (ценность) может быть оценена применительно к нуждам конкретных ее потребителей и оценивается по тем задачам, которые можно решить с ее помощью.

Самая ценная информация – объективная, достоверная,. полная и актуальная.

Но иногда и необъективная, недостоверная информация имеет большую ценность для человека (художественная литература, кино).

Единицы измерения информации

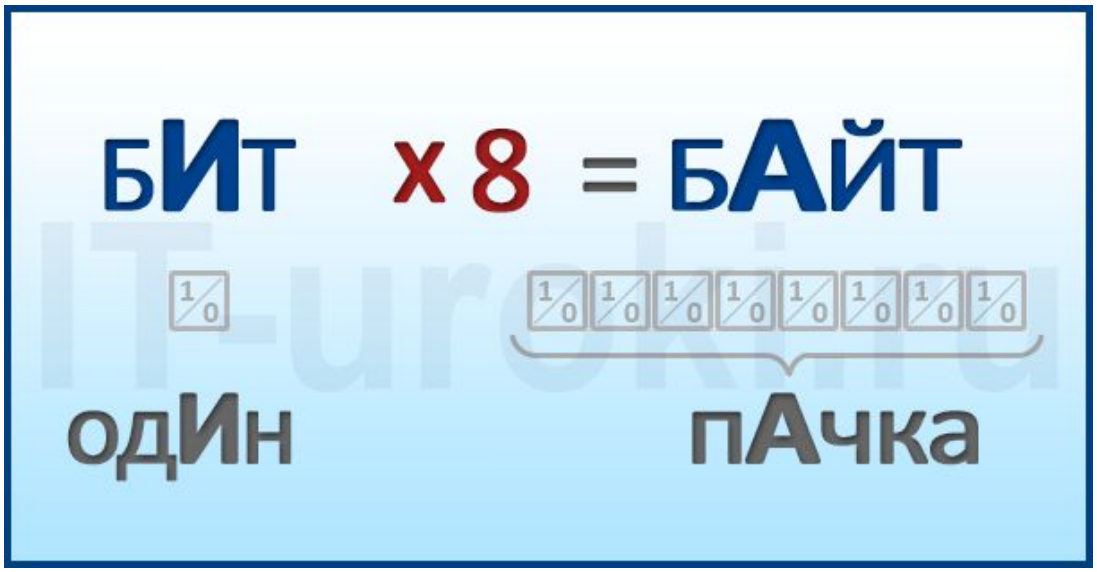
Единицы измерения информации

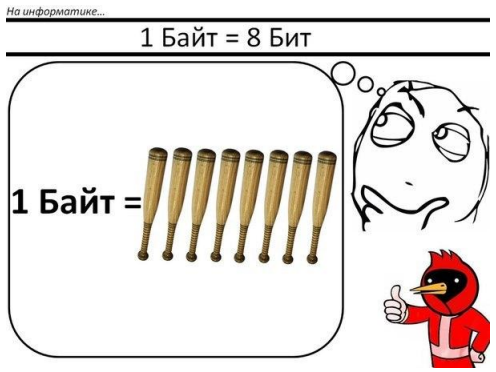
Бит (англ. *binary digit* – двоичное число) – минимальная единица измерения информации, принимает только одно из двух значений (в двоичной системе счисления 0 или 1).

В вычислительной технике (ВТ) значения «0» и «1» передаются различными напряжениями. Значение «0» представляется напряжением 0..0.8 В, а значение «1» – напряжением 2.4..5.0 В.

Единицы измерения информации

Байт – единица хранения и обработки цифровой информации; совокупность из 8 бит, обрабатываемых компьютером одномоментно.





Единицы измерения информации

Информационный объем сообщения измеряется в битах, байтах или других производных единицах:

1 килобайт (Кбайт) = 2^{10} байт = 1024 байт,
1 мегабайт (Мбайт) = 2^{10} Кбайт = 1024 Кбайт = 1048576 байт,
1 гигабайт (Гбайт) = 2^{10} Мбайт = 1024 Мбайт = 1073741824 байт,
1 терабайт (Тбайт) = 2^{10} Гбайт = 1024 Гбайт = 1099511627776 байт
и т. д.

В 1 Гб памяти можно сохранить:

- 500 000 страниц текста;
- св. 1500 фотографий высокого качества;
- аудиозапись 1000 ч. речи «телефонного» качества;
- 200 минутный фильм среднего качества;
- 150 секундный фильм высокого качества.

Системы счисления

Системы счисления

Система счисления – это способ записи чисел с помощью заданного набора специальных знаков (цифр).

Существуют системы позиционные и непозиционные.

В **непозиционных** системах счисления *вес цифры не зависит от позиции, которую она занимает в числе*.

Так, например, в римской системе счисления в числе XXXII (тридцать два) вес цифры X в любой позиции равен просто десяти.

В **позиционных** системах счисления *вес каждой цифры изменяется в зависимости от ее позиции в числе*.

Позиционные системы счисления

Были изобретены шумерам и вавилонянам, развиты индусами.

В любой позиционной системе счисления есть **основание**.

Основание – это количество различных символов, используемых для изображения цифр в данной системе.

Основанием может быть любое натуральное число – 2, 5, 8, 10, 16, …

Поэтому возможно бесконечное множество позиционных систем.

Позиционные системы счисления

Целое число без знака ***x*** в ***b***-ичной системе счисления представляется в виде суммы степеней числа ***b***:

$$x = \sum_{k=0}^{n-1} a_k b^k$$

где a_k – цифры $0 \le a_k \le (b-1)$

n – это разрядность числа;

k – номер разряда;

b^k – вес разряда.

Десятичная система счисления

В этой системе 10 цифр: 0, 1, 2,…9, информацию несет не только цифра, но и ее позиция.

Особую роль играют число 10 и его степени: 10, 100 и т. д.

Самая правая цифра числа показывает число единиц, вторая справа – число десятков, следующая – число сотен и т. д.

Например, число *сто три* представляется в десятичной системе счисления в виде:

$$103 = 1 \cdot 10^2 + 0 \cdot 10^1 + 3 \cdot 10^0.$$

Т.е. число 103 состоит из 1-й сотни, 0 десятков и 3-х единиц.

Двоичная система счисления

В этой системе всего две цифры: 0 и 1.

Особую роль играет число 2 и его степени: 2, 4, 8 и т. д.

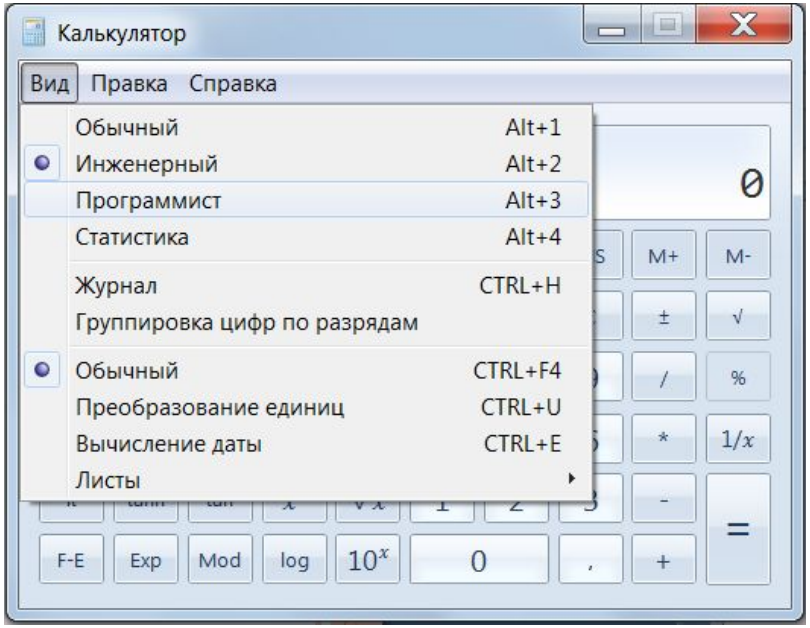
Самая правая цифра числа показывает число единиц, следующая цифра – число двоек, следующая – число четверок и т. д.

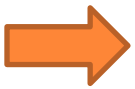
Эта система позволяет представить любое натуральное число в виде последовательности нулей и единиц.

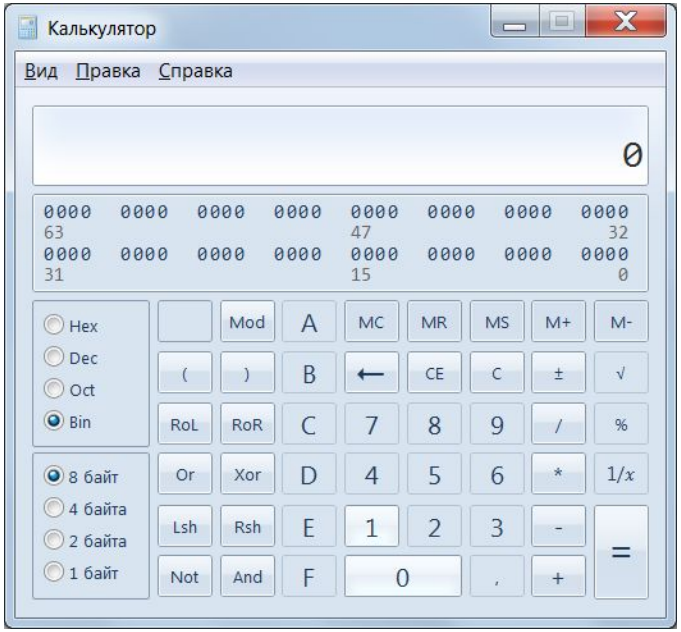
Например, двоичное число 11010_2 расшифровывается так:

$$0 \cdot 1 + 1 \cdot 2 + 0 \cdot 4 + 1 \cdot 8 + 1 \cdot 16 = 26_{10}$$

Двоичная система счисления







В двоичном виде можно представлять не только числа, но и любую другую информацию: тексты, картинки, фильмы и аудиозаписи.

Инженеров двоичное кодирование привлекает тем, что легко реализуется технически.

Программистов – тем, что позволяет представить информацию любой природы в двоичном коде, легко поддающемся обработке на ВТ.

Восьмеричная система счисления

В этой системе счисления 8 цифр: 0, 1, 2, 3, 4, 5, 6, 7. Цифра 1, указанная в самом младшем разряде, означает, как и в десятичном числе, просто единицу. Та же цифра 1 в следующем разряде означает 8, в следующем – 64 и т. д. Число $100_8 = 64_{10}$.

Чтобы перевести в двоичную систему, например, число 611 (восьмеричное), надо заменить каждую цифру эквивалентной ей двоичной триадой (тройкой цифр).

Для перевода двоичного числа в восьмеричную систему нужно разбить его на триады справа налево и заменить каждую триаду соответствующей восьмеричной цифрой.

Шестнадцатеричная система счисления

Запись числа в восьмеричной системе счисления достаточно компактна, но еще компактнее она получается в шестнадцатеричной системе.

В качестве первых десяти цифр взяты цифры 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, а в качестве остальных шести цифр используют первые буквы латинского алфавита: A, B, C, D, E, F.

Цифра 1, записанная в самом младшем разряде, означает просто единицу. Та же цифра 1 в следующем – 16 (десятичное), в следующем – 256 (десятичное) и т. д. Цифра F, указанная в самом младшем разряде, означает 15 (десятичное). Перевод из шестнадцатеричной системы в двоичную и обратно производится аналогично тому, как это делается для восьмеричной системы.

Перевод из одной системы счисления в другую

Десятичная система	Двоичная система	Шестнадцатеричная система
0	0	0
1	1	1
2	10	2
3	11	3
4	100	4
5	101	5
6	110	6
7	111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F
16	10000	10

Кодирование информации

(текстовой, числовой, изображений, звука, видео)

Представление информации в ВТ

В ВТ информация представляется с помощью электрических сигналов.

При этом возможны две формы ее представления:

- в виде непрерывного аналогового сигнала
- в виде нескольких дискретных сигналов

Непрерывная форма представления используется в аналоговых вычислительных машинах (АВМ). Они обладают высоким быстродействием, выполняют любое функциональное преобразование сигнала. Однако из-за сложности длительного хранения таких сигналов и их точного измерения АВМ не могут эффективно решать задачи, связанные с большими объемами информации.

Дискретная (цифровая) форма представления используется в цифровых электронно-вычислительных машинах (ЭВМ), которые легко решают задачи хранения, обработки и передачи больших объемов информации. В отличие от непрерывной величины, количество значений дискретной величины всегда будет конечным.

Кодирование информации

Кодирование информации – это ее представление в двоичной форме, удобной для восприятия ВТ.

Все данные кодируются последовательностью всего двух знаков: 0 и 1.

Количество значений, которое может быть закодировано в двоичной системе, зависит от количества разрядов:

$$N = 2^m$$

где ***n*** – количество разрядов (разрядность) данной системы.

Пример: Какое количество значений N можно закодировать 10-ю разрядами?

$N = 2^{10} = 1024$, т. е. в двоичной системе кодирования 10-ю разрядами можно закодировать 1024 независимых значения.

Текстовая информация

Используются таблицы кодировки ASCII (American Standard Code for Information Interchange), где каждый символ сопоставляется с определенным целым числом.

Восемь двоичных разрядов кодируют 256 различных символов.

Стандартная часть таблицы ASCII											
№ п/п	символ	двоичный код	№ п/п	символ	двоичный код	№ п/п	символ	двоичный код	№ п/п	символ	двоичный код
32	пробел	00100000	56	8	00111000	80	P	01010000	104	h	01101000
33	!	00100001	57	9	00111001	81	Q	01010001	105	i	01101001
34	"	00100010	58	:	00111010	82	R	01010010	106	j	01101010
35	#	00100011	59	;	00111011	83	S	01010011	107	k	01101011
36	$	00100100	60	<	00111100	84	T	01010100	108	l	01101100
37	%	00100101	61		00111101	85	U	01010101	109	m	01101101
38	&	00100110	62	>	00111110	86	V	01010110	110	n	01101110
39	'	00100111	63	?	00111111	87	W	01010111	111	o	01101111
40	(	00101000	64	@	01000000	88	X	01011000	112	p	01110000
41	)	00101001	65	A	01000001	89	Y	01011001	113	q	01110001
42	*	00101010	66	B	01000010	90	Z	01011010	114	r	01110010
43	+	00101011	67	C	01000011	91	[	01011011	115	s	01110011
44	,	00101100	68	D	01000100	92	\	01011100	116	t	01110100
45	-	00101101	69	E	01000101	93	]	01011101	117	u	01110101
46	.	00101110	70	F	01000110	94	^	01011110	118	v	01110110
47	/	00101111	71	G	01000111	95	_	01011111	119	w	01110111
48	0	00110000	72	H	01001000	96	'	01100000	120	x	01111000
49	1	00110001	73	I	01001001	97	a	01100001	121	y	01111001
50	2	00110010	74	J	01001010	98	b	01100010	122	z	01111010
51	3	00110011	75	K	01001011	99	c	01100011	123	{	01111011
52	4	00110100	76	L	01001100	100	d	01100100	124	|	01111100
53	5	00110101	77	M	01001101	101	e	01100101	125	}	01111101
54	6	00110110	78	N	01001110	102	f	01100110	126	~	01111110
55	7	00110111	79	O	01001111	103	g	01100111	127		01111111

Текстовая информация

Этого хватит, чтобы выразить различными комбинациями восьми битов все строчные и прописные символы английского и русского языков, а также знаки препинания, символы основных арифметических действий и некоторые общепринятые специальные символы.

Но не все так просто, и существуют определенные сложности.

Основная таблица ASCII

	0	1	2	3	4	5	6	7
0		►		0	@	P	`	p
1	☺	◄	!	1	A	Q	a	q
2	☻	↕	"	2	B	R	b	r
3	♥	‼	#	3	C	S	c	s
4	♦	¶	$	4	D	T	d	t
5	♣	§	%	5	E	U	e	u
6	♠	▬	&	6	F	V	f	v
7	•	↨	'	7	G	W	g	w
8	◘	↑	(	8	H	X	h	x
9	○	↓	)	9	I	Y	i	y
A	◙	→	*	:	J	Z	j	z
B	♂	←	+	;	K	[	k	{
C	♀	∟	,	<	L	\	l	¦
D	♪	↔	-	=	M	]	m	}
E	♫	▲	.	>	N	^	n	~
F	☼	▼	/	?	O	_	o	⌂

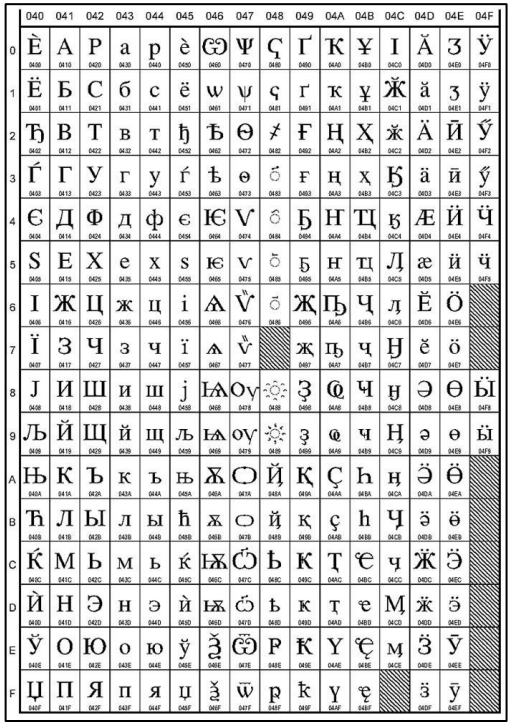
Расширенная таблица ASCII

	8	9	A	B	C	D	E	F
0	А	Р	а	░	└	╨	р	≡
1	Б	С	б	▙	┴	╤	с	±
2	В	Т	в	▛	┬	╥	т	≥
3	Г	У	г	▌	├	╙	у	≤
4	Д	Ф	д	▐	─	╘	ф	⌠
5	Е	Х	е	▟	┼	╒	х	⌡
6	Ж	Ц	ж	▞	╞	╓	ц	÷
7	З	Ч	з	▚	╟	╫	ч	≈
8	И	Ш	и	╗	╚	╪	ш	°
9	Й	Щ	й	╣	╔	┘	щ	∙
A	К	Ъ	к	║	╩	┌	ъ	·
B	Л	Ы	л	╦	╠	▀	ы	√
C	М	Ь	м	╝	═	▄	ь	ⁿ
D	Н	Э	н	╜	╬	▌	э	²
E	О	Ю	о	╛	╧	▐	ю	■
F	П	Я	п	┐	╨	▄	я	

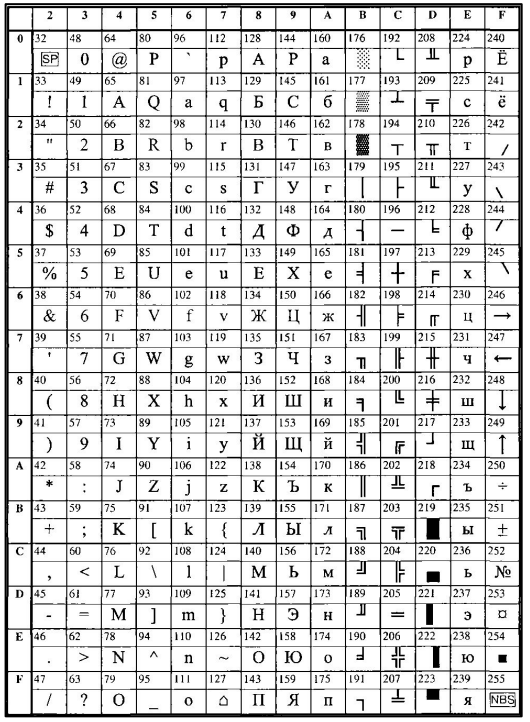
Текстовая информация

Раньше они были связаны с отсутствием необходимых стандартов, а в настоящее время, наоборот, вызваны изобилием одновременно действующих стандартов.

Практически для всех языков в мире созданы свои кодовые таблицы. Для того чтобы весь мир одинаково кодировал текстовые данные, нужны единые таблицы кодирования, что до сих пор пока не сделано.



Кодировка Unicode



Альтернативная кодировка ГОСТ 19768-87

Числовая информация

В ВТ существует два способа представления чисел:

1. Форма с фиксированной точкой – для целых чисел
2. Форма с плавающей точкой – для действительных* чисел

ВТ оперирует с числами, содержащими конечное число двоичных цифр (разрядов).

От количества разрядов зависит точность ВТ и диапазон представляемых чисел.

* Действительные (вещественные) числа – это рациональные числа и иррациональные числа. В качестве разделителя целой и дробной части может быть точка и запятая.

Числовая информация

Кодирование целых чисел:

Целые числа без знака (положительные) – для их хранения отводится последовательность из 8, 16 или 32 бит.

Пример: Максимальное 8-битное число $11111111_2 = 255_{10}$ будет храниться следующим образом (прямой код):

1	1	1	1	1	1	1	1

Максимальное целое неотрицательное число будет, если во всех ячейках единицы. Оно равно $2^N - 1$, где N – разрядность числа.

Для 16-разрядных чисел: $2^{16} - 1 = 65\ 535$

Для 32-разрядных: $2^{32} - 1 = 4\ 294\ 967\ 295$

Числовая информация

Кодирование целых чисел:

Целые числа со знаком (положительные и отрицательные) – для хранения также отводится 8, 16 или 32 бит, причем старший бит (слева) обозначает знак числа: 0 – положительное число; 1 – отрицательное.

При записи таких чисел используют не прямой, а дополнительный код, равный $2^N - A$, где A – прямой код числа.

Дополнительным называют код, в котором для положительного числа в знаковом разряде ставят 0, в цифровых – модуль числа, для отрицательного числа в знаковом разряде ставят 1, а в цифровых – дополнение числа до единицы (инвертирование цифр).

Числовая информация

Кодирование целых чисел:

Целые числа со знаком (положительные и отрицательные) – для хранения также отводится 8, 16 или 32 бит, причем старший бит (слева) обозначает знак числа: 0 – положительное число; 1 – отрицательное.

Пример: Число -1 в 8-разрядном двоичном коде выглядит, как 11111111, а -2 как 11111110 и т.д.

Дополнительный код позволяет заменить операцию вычитания операцией сложения, что исключает операцию вычитания из набора команд процессора.

8-разрядное число может изменяться в интервале от -128 до 127

16-разрядное – от -32 768 до 3 2767

32-разрядное – от -2 147 483 648 до 2 147 483 647

Числовая информация

Кодирование действительных чисел:

Число X приводится к нормализованной форме:

$$X = \pm M \cdot N^P$$

где M – мантисса (дробная часть);

N – основание системы счисления;

P – порядок числа.

Для десятичной системы счисления $X = \pm M \cdot 10^P$,

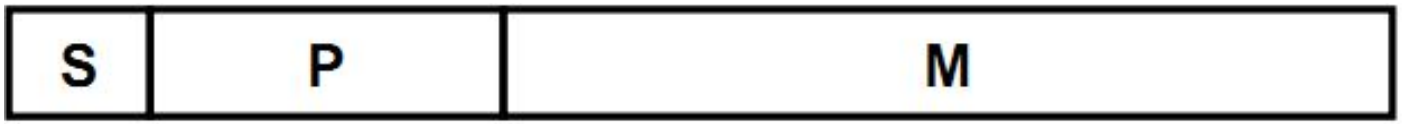
для двоичной $X = \pm M \cdot 2^P$

Пример: число 22.22_{10} в нормализованном виде выглядит, как $+0{,}2222 \cdot 10^2$.

Числовая информация

Кодирование действительных чисел:

Таким образом, действительное число в памяти компьютера хранится в двоичной системе в виде:



где **S** – признак знака числа.

Поскольку размер памяти под мантиссу и порядок ограничен, то действительные числа представляются с погрешностью, определяемой количеством разрядов в мантиссе, и имеют диапазон изменения, зависящий от количества разрядов в порядке числа.

Числовая информация

Кодирование действительных чисел:

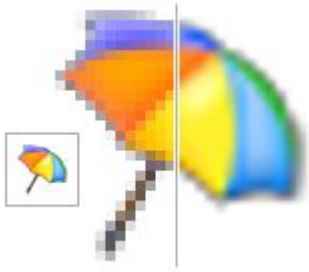
Особенности арифметики чисел с плавающей точкой могут существенно влиять на результаты расчетов, вплоть до того, что погрешность может сделать невозможным получение какого-либо результата вообще.

Данные с плавающей точкой по стандарту IEEE-754-1985

Тип	Размер, бит	Диапазон изменения чисел		Точность, количество цифр в числе	Машинное ε
		максимум	минимум		
single	32	$3.4 \cdot 10^{-38}$	$3.4 \cdot 10^{38}$	6	$1{,}192 \cdot 10^{-7}$
double	64	$1.7 \cdot 10^{-308}$	$1.7 \cdot 10^{308}$	15	$2{,}221 \cdot 10^{-16}$
long double	80	$3.4 \cdot 10^{-4932}$	$3.4 \cdot 10^{4932}$	19	$1{,}084 \cdot 10^{-19}$

Изображения

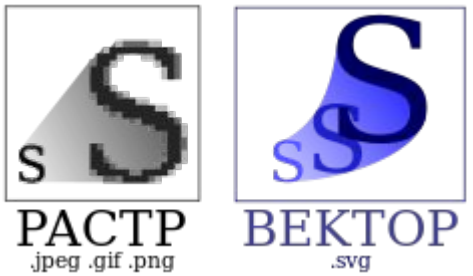
Изображение состоит из крошечных элементов – пикселей (англ. *picture element*).



Для каждого пикселя задаются свойства, такие как координаты, цвет, которые можно выразить целыми числами и использовать двоичный код для представления графической информации.

Изображение на экране монитора всегда состоит из отдельных пикселей и называется растровым.

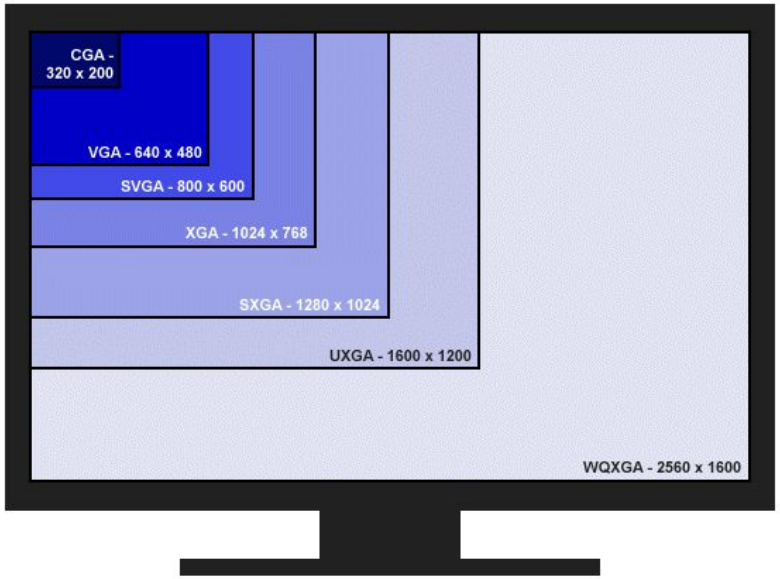
Однако для хранения его может использоваться и векторное представление, когда изображение представляется в виде набора графических объектов с их свойствами.



Изображения

Число пикселей по горизонтали и вертикали называют **разрешением**.

Используются разрешения 1024x768, 1280x800, 1280x1024, 1920x1080 и др.



Каждый пиксель нумеруется, начиная с нуля, *слева направо* и *сверху вниз*.

Изображения

Для кодирования цвета пикселя используются различные цветовые схемы:

Для **черно-белого** изображения пиксель может находится в одном из двух состояний: светится – не светится.

Для его кодирования достаточно 1 бита памяти:

- 1 – белый,
- 0 – черный

Черно-белое изображение буквы К

Задача 1

Сколько бит памяти требуется для хранения черно-белого изображения буквы К?

Дано:

Растровая сетка размером 10 х 10 с изображением буквы

Сколько бит памяти требуется для хранения информации?

Решение:

1. Количество пикселов, требующихся для хранения: 10×10=100.

2. Т.к. для хранения 1 пикселя требуется 1 бит памяти, объем памяти = 100 бит.

Ответ: Для хранения информации требуется 100 бит памяти.

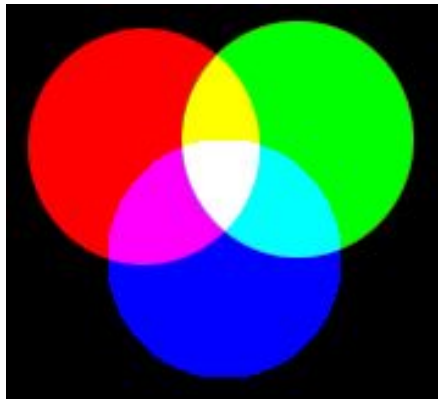
Изображения

Для **цветного** изображения чаще всего используется **RGB-схема**.

Любой цвет, видимый человеческим глазом, можно получить *смешиванием* основных цветов.

В качестве таких цветов используют:

- красный Red
- зеленый Green
- синий Blue.



Каждый пиксель на экране - это совокупность красной, зеленой и синей точки.

Эти точки расположены близко друг к другу и кажутся слившимися в одну.

Для хранения цвета из трех составляющих требуется *трехбитовый* двоичный код.

Цвет	R (красный)	G (зеленый)	B (синий)
	0	0	0
	0	0	1
	0	1	0
	0	1	1
	1	0	0
	1	0	1
	1	1	0
	1	1	1

8 цветов

Изображения

Т.е. с помощью трех цветов нельзя получить палитру из более, чем 8 цветов.

Однако на экранах современных компьютеров получают цветные изображения, составленные из сотен, тысяч и даже миллионов различных красок и оттенков.

Как это достигается?

Количество цветов увеличивают, изменяя яркости базовых цветов.

Если к трем битам базовых цветов добавить один бит интенсивности, управляющий яркостью всех трех цветов *одновременно*, то получится 16-цветная палитра.

16 цветов

Цвет	R (красный)	G (зеленый)	B (синий)	И (интенсивность)
	0	0	0	0
	0	0	1	0
	0	1	0	0
	0	1	1	0
	1	0	0	0
	1	0	1	0
	1	1	0	0
	1	1	1	0
	0	0	0	1
	0	0	1	1
	0	1	0	1
	0	1	1	1
	1	0	0	1
	1	0	1	1
	1	1	0	1
	1	1	1	1

Изображения

Бóльшее количество цветов получают при *раздельном* управлении интенсивностью базовых цветов.

Количество бит K для записи цвета одного пикселя называется глубиной цвета b.

$$K = 2^b$$

Пример: для получения 256 цветов требуется 8 бит = 1 байт на каждый пиксель, т.к. $256 = 2^8$

Минимальный объем видеопамяти должен быть таким, чтобы в нее помещался один кадр изображения

Изображения

Задача

Рисунок построен с использованием палитры 256 цветов на экране монитора с графическим разрешением 1024 x 768. Рассчитать объем памяти, необходимый для хранения этого рисунка.

Дано:

Растровая сетка размером 1024 x 768.

Количество цветов K = 256

Рассчитать объем видеопамяти.

Решение:

1. По формуле $K = 2^b$ определяем глубину цвета. $256 = 2^8$. Значит, глубина цвета b = 8 бит.

2. Количество пикселей на экране: 1024 x 768 = 786432.

3. 786432 * 8 бит = 786432 байт = 768 Кбайт

Ответ: Объем видеопамяти - 768 Кбайт.

Изображения

В RGB-схеме на кодирование цвета одной 3-хцветной точки необходимо 24 бита (3 байта), по 8 бит (1 байту) на каждый цвет.

Так можно обеспечить $2^{24} \approx 16{,}7$ млн. цветов!

Этот режим называется полноцветным (True Color)

RGB-схему еще называют *аддитивной*, она была стандартизована в 1931 г. и впервые использована в цветном телевидении.

Существуют и другие цветовые схемы:

1. Субтрактивная схема CMYK (Cyan-Magenta-Yellow-blacK), в которой цвета получают, вычитая базовые цвета из белого. Применяется в цветных принтерах.
2. В цветовой схеме HSV (Hue-Saturation-Value) цвета представляют через цвет, насыщенность и значение.
3. В схеме HLS (Hue-Lightness-Saturation) через оттенок, яркость и насыщенность.

Современные графические редакторы могут работать с несколькими цветовыми схемами.

Звук

Методы кодирования звуковой информации пришли в ВТ позднее всего, поэтому до сих пор не стандартизованы.

Выделяют **два** основных направления:

1. Частотная модуляция (FM, Frequency Modulation)
2. Таблично-волновой синтез (Wave-Table)

Звук

Частотная модуляция

Сложный звук раскладывают на последовательность простейших синусоидальных сигналов, частота которых определяет высоту звука, а амплитуда – громкость.

Естественный звук имеет непрерывный спектр, является аналоговыми. Его разложение в гармонические ряды и представление в виде дискретных цифровых сигналов выполняют аналогово-цифровые преобразователи.

Обратное преобразование выполняют цифро-аналоговые преобразователи.

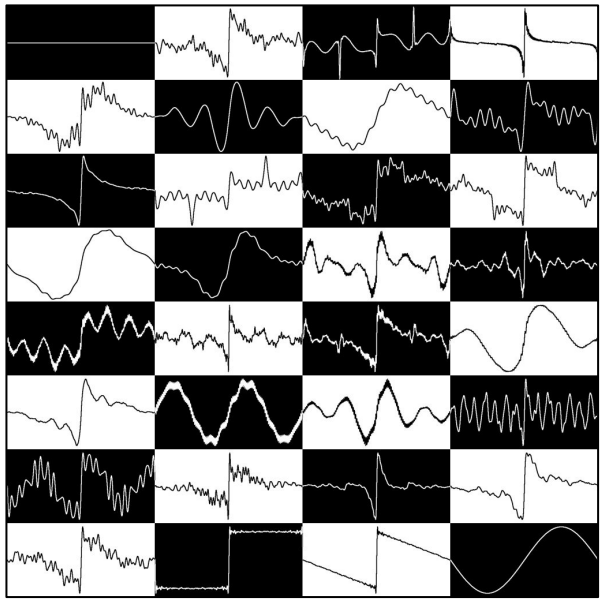
При таких преобразованиях часть информации теряется, поэтому качество звукозаписи обычно получается не вполне удовлетворительным.

Звук

Таблично-волновой синтез

Имеются заранее подготовленные таблицы, в которых хранятся образцы звуков для множества различных музыкальных инструментов.

Такие образцы называются сэмплами.



Числовые коды выражают тип инструмента, номер его модели, высоту тона, продолжительность и интенсивность звука, динамику его изменения.

Поскольку в качестве образцов используются «реальные» звуки, то качество звука получается очень высоким и приближается к качеству звучания реальных музыкальных инструментов.

Видео

Видеоинформация – это набор выводимых друг за другом изображений (кадров).

Если частота смены кадров превышает 24 кадра в секунду, то у зрителя создается впечатление непрерывного движения.

1 секунда несжатого видео с разрешением 640x480 (256 цветов) весит 9 Мб!

Если увеличить разрешение до 1280x1024 (16 млн. цветов), то объем станет 114 Мб(!!!).

В связи с большим объемом большую актуальность приобретают вопросы сжатия видеоинформации без снижения качества изображения.

Методов сжатия видеоинформации существует очень много, они постоянно совершенствуются. В основе всех методов лежит избыточность видеоинформации – между двумя соседними кадрами обычно изменяется только малая часть сцены, например, смещается небольшой объект на фоне неподвижного заднего плана.

Поэтому можно сохранять полную информацию только для так называемых **опорных кадров**, а для остальных достаточно информации о смещении объекта, об изменении фона и т.д.

Файлы, файловая структура и файловые системы

Файлы

При хранении данных решают проблемы: как сохранить данные в наиболее компактном виде и как обеспечить к ним удобный и быстрый доступ.

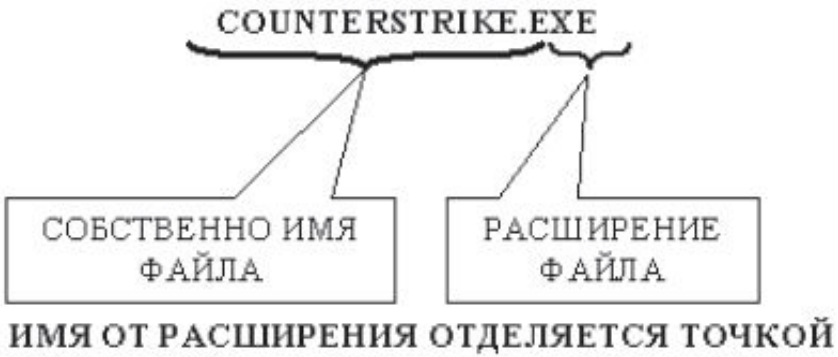
Файл – это единица хранения данных произвольного объема, обладающая собственным уникальным именем.

Каждый файл имеет **адрес** для обеспечения доступа.

Обычно в файле хранят данные, относящиеся к одному типу.

В этом случае тип данных определяет тип файла.

Имя файла



Собственно имя файла может состоять из букв русского и английского алфавитов, цифр и спецсимволов. Его длина не должна превышать 256 символов.

В зависимости от **расширения** файлы делятся на два типа:

- **исполняемые**
- **неисполняемые**

Типы файлов

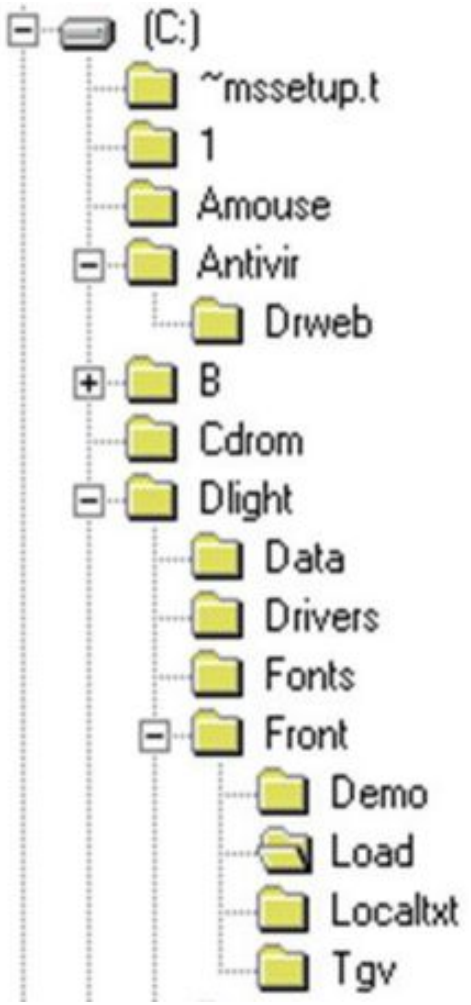
Файлы	
исполняемые	**неисполняемые**
Выполняются самостоятельно, т. е. не требуют специальных программ для их запуска. Имеют следующие расширения: • exe – готовый к исполнению файл (*tetris.exe; winword.exe*) • com – файл операционной системы (*command.com*) • sys – файл операционной системы (*Io.sys*) • bat – командный файл операционной системы MS-DOS (*autoexec.bat*)	Неисполняемые файлы для запуска требуют установки специальных программ. Могут иметь следующие расширения: • .txt, .doc, .rtf – текстовый файл • .bmp, .jpg, .gif – файл изображения • .arj, .rar, .zip – файл архива • .xls, .xlsx – электронная таблица • .mp3, .wav – аудиофайл • .avi, .mpeg – видеофайл

Файловая структура

В Windows папки и файлы образуют на дисках *иерархическую* файловую структуру.

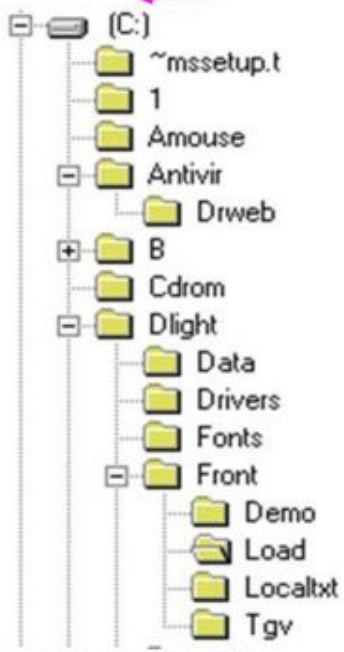
Файлы находятся в папках.

Папки вложены в другие папки, более высокого уровня.



Файловая структура

Папка самого высокого уровня называется **корневой** - она одна на каждом диске.



Назначение файловой структуры -

- обеспечить однозначное отыскание любого файла, если известно его имя и путь поиска.

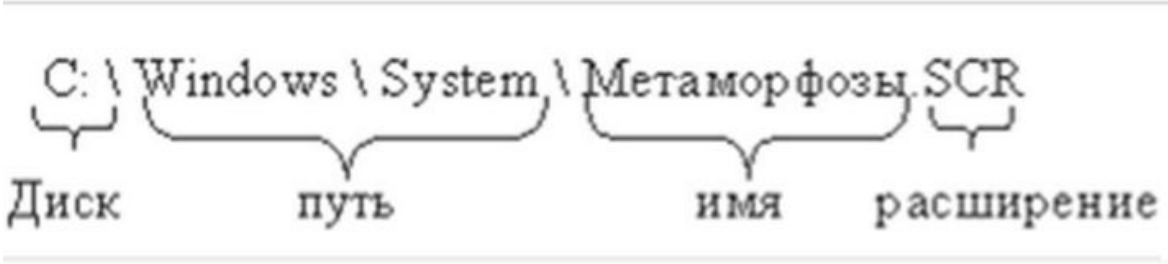
Путь поиска начинается с корневой папки (её имя совпадает с обозначением диска) и далее ведёт через все вложенные папки к той папке, где находится разыскиваемый файл.

`C:\Dlight\Front\Demo`

Создание и обслуживание файловой структуры - это одна из **сновных** функций **операционной системы**.

Файловая структура

Полное имя файла состоит из пути к файлу (начиная с имени диска) и имени самого файла.



Файловые системы

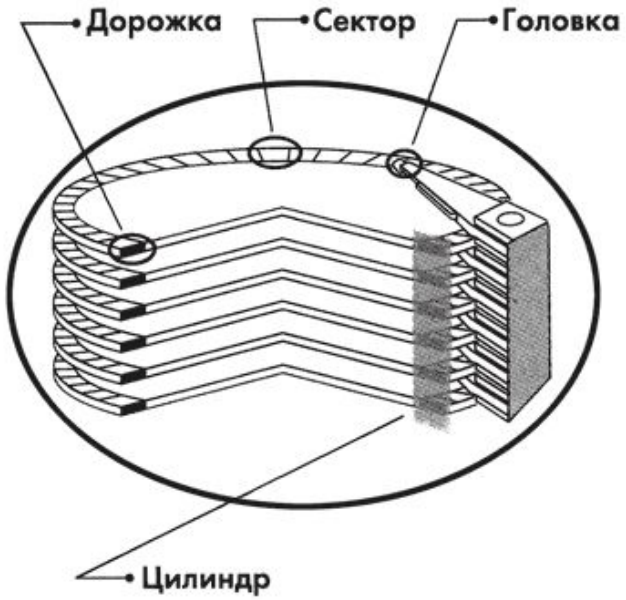
Файловая система – это набор соглашений, определяющих организацию данных на носителях информации.

Функции файловой системы:

- именование файлов
- программный интерфейс работы с файлами для приложений
- отображения логической модели файловой системы на физическую организацию хранилища данных
- организация устойчивости файловой системы к сбоям питания, ошибкам аппаратных и программных средств
- содержание параметров файла, необходимых для правильного его взаимодействия с другими объектами системы

Файловые системы

Жесткий диск рассматривается как трехмерная матрица, измерениями которой являются номера поверхности, цилиндра и сектора.



Данные о расположении файла хранятся в системной области диска в специальных FAT-таблицах (File Allocation Table – таблица размещения файлов).

Если файловая система диска не поддерживается ОС, то вся информация на этом диске окажется недоступной

Файловые системы

	NTFS	FAT 32	FAT
Поддерживаемые ОС	Windows XP, 2000, NT	MS-DOS Windows 95, 98, ME, XP, 2000, NT	MS-DOS Windows 95, 98, ME, XP, 2000, NT
Возможные размеры логических дисков	От 10 Мб до 2 Тб	От 512 Мб до 2 Тб	До 4 Гб
Возможные размеры хранимых файлов	Нет ограничений	Максимальный размер – до 4 Гб	Максимальный размер 2 Гб

Алгоритмы

Алгоритм

Алгоритм – набор инструкций, описывающих порядок действий исполнителя для достижения результата решения задачи за конечное число действий, при любом наборе исходных данных.

Понятие алгоритма – одно из основных в информатике.

Слово «алгоритм» происходит от имени великого узбекского математика IX в. аль-Хорезми.

Свойства алгоритма

1. Универсальность (массовость)

Алгоритм разрабатывается в общем виде, т.е. он должен быть применим для класса задач, отличающихся исходными данными.

2. Дискретность (пошаговость)

Алгоритм должен представлять решение задачи, как последовательное выполнение простых операций.

3. Однозначность

Алгоритм не допускает неоднозначного толкования.

4. Результативность (конечность)

Алгоритм за конечное число шагов должен либо решить задачу либо сообщить о невозможности ее решения.

Исполнитель алгоритма

Исполнителями служат многие технические устройства и, прежде всего, компьютер. Составление алгоритмов для него – дело особо ответственное и тщательное. Ведь компьютер не сможет домыслить и исправить ошибки, он делает только то, что ему указано.

Компьютерная программа – алгоритм, предназначенный для выполнения его на компьютере.

Чтобы компьютер смог выполнить программу, она должна быть записана в специальной форме, понятной компьютеру.

Алгоритмический язык – это набор правил записи компьютерной программы.

Способы представления алгоритма

- **словесное описание**
- **табличный**
- **блок-схема**
- **программа**

Способы представления алгоритма

Словесное описание:

Это, по существу, обычный язык, но с тщательным отбором слов и фраз, не допускающих лишних слов, двусмысленностей и повторений.

Алгоритм описывается в виде последовательности шагов. На каждом шаге определяется состав выполняемых действий и направление дальнейших вычислений. При этом, если на текущем шаге не указывается какой шаг должен выполняться следующим, то осуществляется переход к следующему шагу.

Алгоритм «Открывание замка»

1. Достать ключ
2. Вставить ключ в замочную скважину
3. Повернуть ключ 2 раза в нужном направлении
4. Вынуть ключ

Способы представления алгоритма

Табличный:

Используется в бухгалтерских документах, инженерных расчетах и многих других сферах деятельности.

Параметры и обозначения	Расчетные формулы и указания	Числовые значения
Угол профиля α	Нормальный исходный контур	20°
Делительный диаметр d	$d_1 = z_1 m$ $d_2 = z_2 m$	60 90
Межосевое расстояние a	$a = \frac{(z_1 + z_2)m}{2}$	75
Диаметр вершин зубьев d_a	$d_{a1} = d_1 + 2m$ $d_{a2} = d_2 + 2m$	66 96
Диаметр впадин d_f (справочный размер)	$d_{f1} = d_1 - 2(c + m)$ $d_{f2} = d_2 - 2(c + m)$	52,8 82,8
Постоянная хорда $\bar{s}_c$ Высота до постоянной хорды $\bar{h}_c$	$\bar{s}_{c1} = \bar{s}_{c2} = 1{,}387m$ $\bar{h}_{c1} = \bar{h}_{c2} = 0{,}748m$	4,16 2,24

Способы представления алгоритма

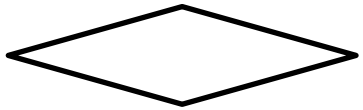
Блок-схема:

Является наиболее наглядным и достаточно простым способом представления алгоритма. Алгоритм составляется из графических блоков, внутри которых записывают описание команд или условий. Для указания последовательности выполнения блоков их соединяют линиями связи (потока информации).

- начало или конец описания алгоритма

- ввод исходных данных или вывод результатов

- арифметические и другие действия



- блок выбора (ветвления)

- блок входа в цикл с указанием параметров

Способы представления алгоритма

Блок-схема:

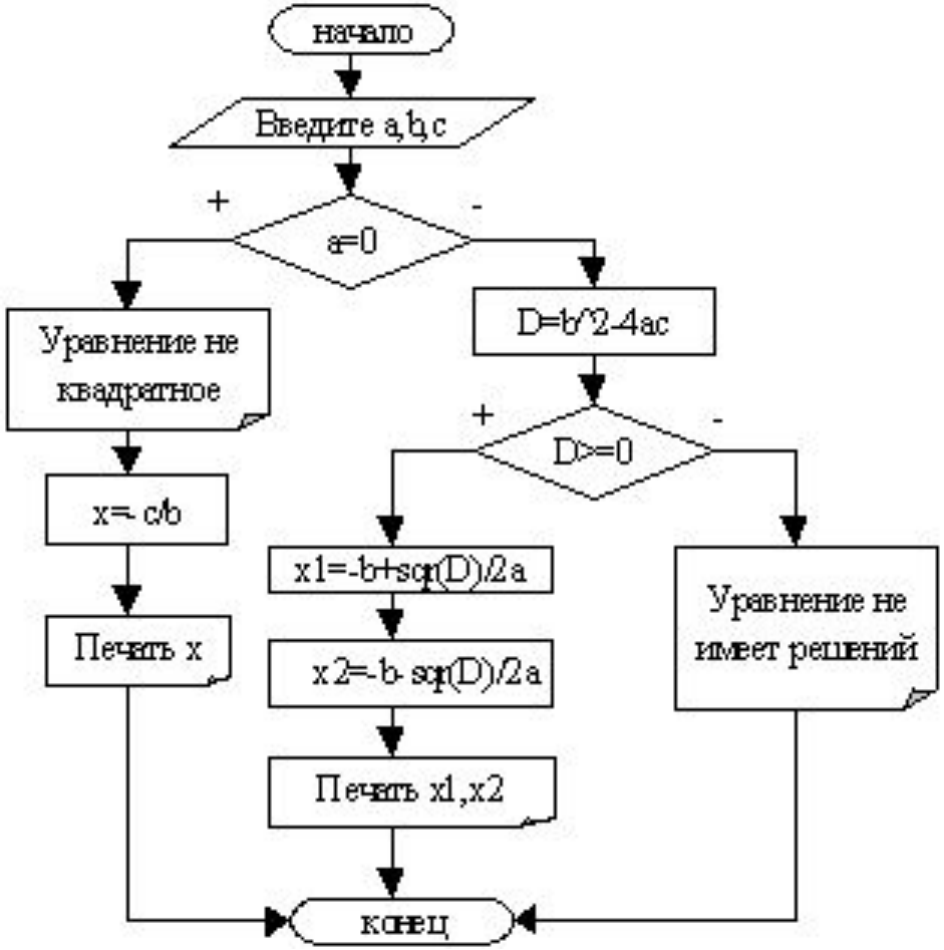
Правила изображения блок-схем:

1. *В блок-схеме можно использовать строго определённые типы блоков.*
2. *Стрелки на линиях связи можно не ставить при направлении сверху вниз и слева направо; противоположные направления обязательно указывают стрелкой на линии.*
3. *Для удобства блоки могут помечаться метками (буквами или цифрами).*
4. *Внутри блока ввода/вывода пишется ВВОД или ВЫВОД и перечисляются имена данных, подлежащих вводу/выводу.*
5. *Внутри блока действия для присваивания переменных значений используется знак присваивания.*

Способы представления алгоритма

Блок-схема:

Решение квадратного уравнения:

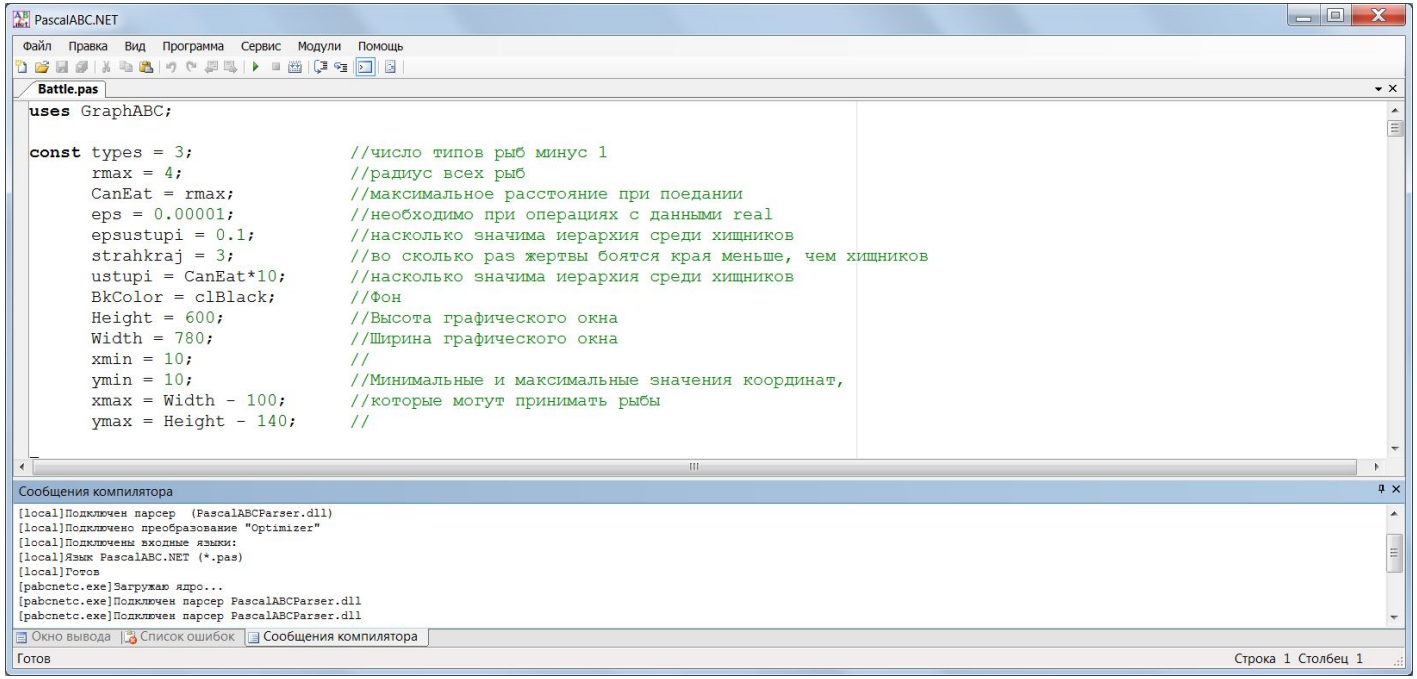


Способы представления алгоритма

Программа:

Первые три способа представления алгоритма предназначены для человека и только программа – это алгоритм для исполнителя-компьютера.

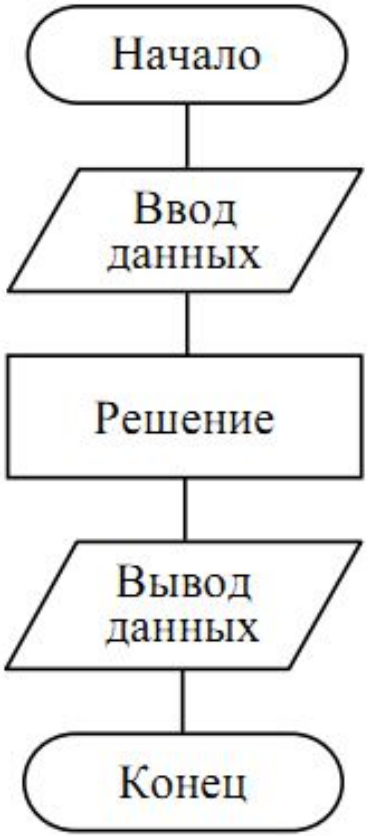
Программа записывается на языке программирования, понятном ЭВМ. Сейчас известно несколько сот языков программирования. Наиболее популярные Basic, C++, Pascal, Ассемблер и т.д.



Типы алгоритмов

1. Линейный:

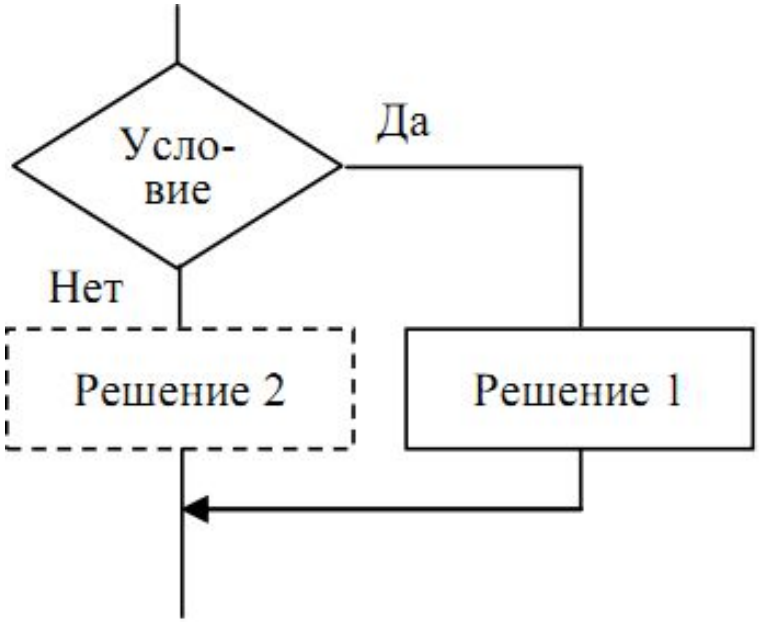
Неизменная последовательность операций от начала до конца алгоритма без повторов действий.



Типы алгоритмов

2. Разветвляющийся:

Последовательность операций может изменяться в зависимости от каких-либо условий.

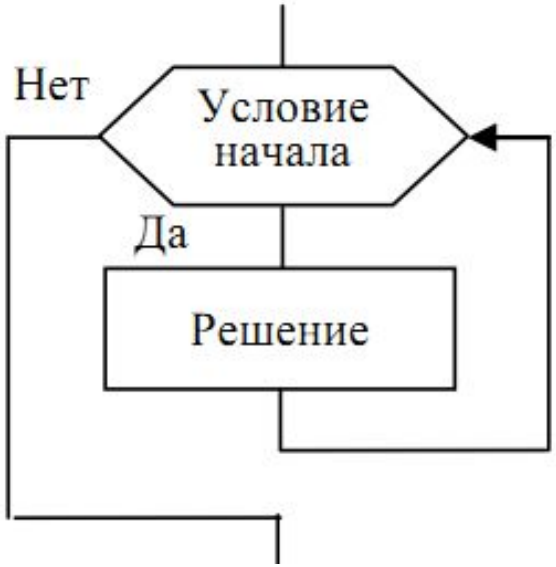


Типы алгоритмов

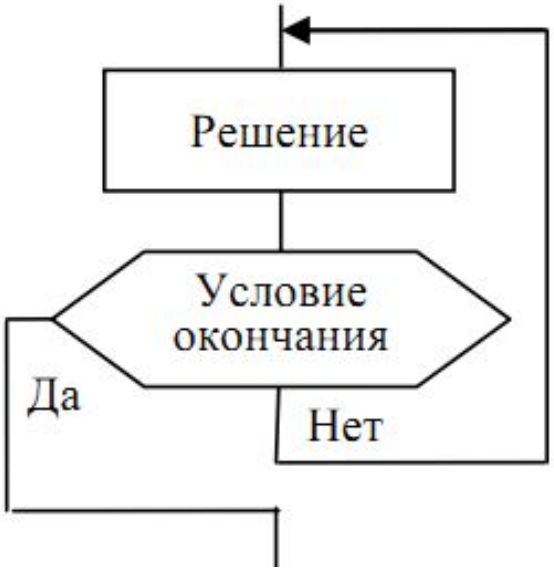
3. Циклический:

Операции могут повторяться многократно, число повторений зависит от выполнения некоторого условия.

Цикл с предусловием



Цикл с постусловием



Этапы разработки программы

Этапы разработки программы

1. Словесная постановка задачи

Точная формулировка задачи и цели, которые необходимо достигнуть при ее решении. Определение целей сводится к нахождению исходных и промежуточных величин.

2. Математическая постановка задачи

Запись условия задачи в виде математических соотношений. На этом же этапе выполняется выбор математического метода решения задачи. Он должен обеспечить решение задачи выполнением 4-х арифметических действий и элементарных функций. Для простых задач метод решения очевиден, поэтому этот этап опускается.

3. Разработка алгоритма и его блок-схемы

Устанавливается необходимая последовательность арифметических и логических действий для решения задачи. Эта последовательность представляется в виде блок-схемы.

Этапы разработки программы

4. Запись алгоритма на языке программирования (кодирование)

Алгоритм записывается в виде последовательности операторов выбранного языка.

5. Контрольное тестирование и отладка программы

Отладка необходима для выявления и устранения ошибок, допущенных на предыдущих этапах. Правильность выполнения программы осуществляется сравнением результатов, полученных при расчете нескольких вариантов задачи вручную и на ЭВМ. Полученный вручную расчет называется контрольным тестом.

6. Анализ результатов

Выполняется решение задачи для всего множества исходных данных. Анализ результатов выполняется, как правило, лицом, для которого решалась задача.

Языки программирования

Языки программирования

Язык программирования (ЯП) – формальная знаковая система для записи компьютерных программ.

ЯП бывают:

1. **Низкого уровня** – исполняются непосредственно
2. **Высокого уровня** – исполняются после преобразования (трансляции) в исполняемую программу

ЯП низкого уровня

Это машинно-зависимые ЯП, т.к. ориентированы на конкретный тип процессора и учитывают его особенности.

К ним относятся:

- **программирование в машинных кодах**
- **ассемблер**
- **макроассемблер**

Программы на ЯП низкого уровня получаются очень эффективные и компактные, т.к. программист использует все возможности процессора. Подобные ЯП применяют для написания небольших системных приложений, драйверов устройств, библиотек. Когда объем памяти небольшой (несколько Кб), альтернативы таким ЯП нет.

ЯП низкого уровня

Программы в машинных кодах:

На них можно посмотреть, если открыть любой исполняемый файл в Блокноте

```
0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68 ;
69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F ;
74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20 ;
6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00 ;
A7 DC F4 A6 E3 BD 9A F5 E3 BD 9A F5 E3 BD 9A F5 ;
```

Фрагмент программы в машинных кодах
(шестнадцатеричный код)

Сейчас сложные программы не пишут в машинных кодах.

ЯП низкого уровня

Ассемблер:

Представляет из себя мнемоническую запись машинных команд.

Используется также достаточно редко.

Фрагмент того же кода в машинных кодах и на Ассемблере:

Машинный код	Ассемблер
0E	PUSH CS
1F	POP DS
BA0E00	MOV DX,000E
B409	MOV AH,09
CD21	INT 21
B8014C	MOV AX,4C01
CD21	INT 21
54	PUSH SP
68	DB 68
69	DB 69
7320	JNB 0033
7072	JO 0087
6F	DB 6F
67	DB 67
7261	JB 007A
6D	DB 6D
206361	AND [BP+DI+61],AH
6E	DB 6E
6E	DB 6E
6F	DB 6F
и т.д.	

Мнемокод использует буквенные обозначения машинных операций (PUSH, POP, MOV и т.д.) и регистров памяти (CS, DS, DX, AH и т.д.).

Процесс трансляции ассемблерной программы в машинный код называется **ассемблированием**, а обратный процесс – **дизассемблированием**.

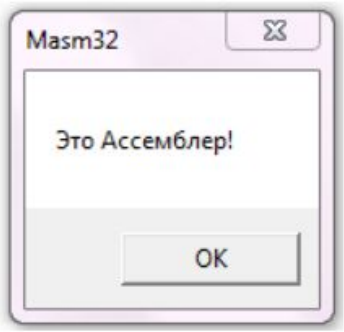
ЯП низкого уровня

Макроассемблер:

Современный макроассемблер для Windows (например, Masm32) занимает промежуточное положение между Ассемблером и языками высокого уровня: он использует команды обычного Ассемблера и множество системных процедур из библиотек ОС.

Программа на Masm32, создающая диалоговое окно:

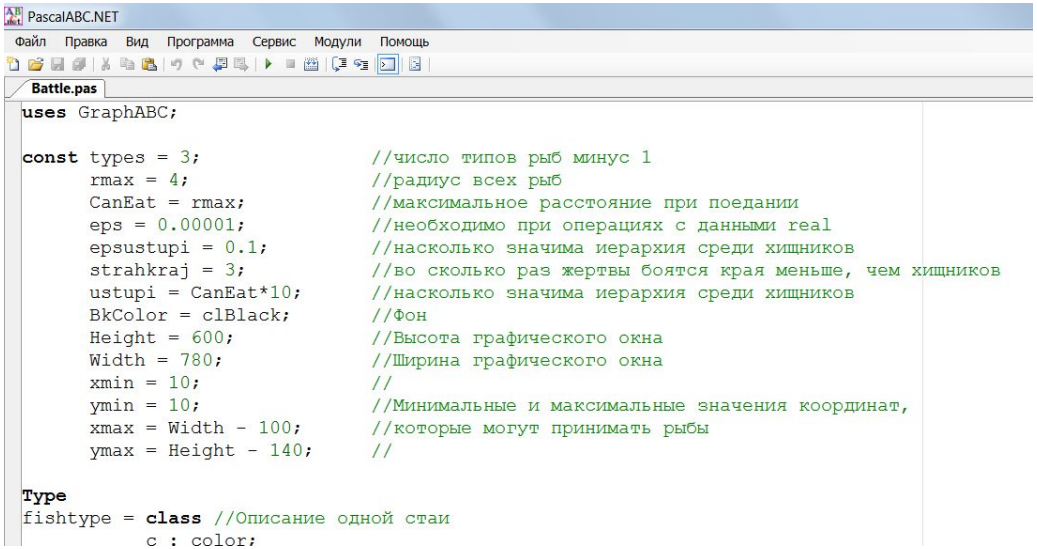
```
.386
.model flat, stdcall
option casemap :none
include \masm32\include\windows.inc
include \masm32\include\user32.inc
include \masm32\include\kernel32.inc
includelib \masm32\lib\user32.lib
includelib \masm32\lib\kernel32.lib
.code
start:
 jmp @F
    Titl  db "Masm32",0
    Msg  db  "Это Ассемблер!",0
  @@:
  invoke MessageBox,0,ADDR Msg,ADDR Titl,MB_OK
  invoke ExitProcess,0
end start
```



ЯП высокого уровня

Повышают производительность программистов за счет использования команд из английского слов, заменяющих много машинных инструкций.

Фрагмент текста программы на языке Pascal:




```
uses GraphABC;

const types = 3;                //число типов рыб минус 1
      rmax = 4;                 //радиус всех рыб
      CanEat = rmax;            //максимальное расстояние при поедании
      eps = 0.00001;            //необходимо при операциях с данными real
      epsustupi = 0.1;          //насколько значима иерархия среди хищников
      strahkraj = 3;            //во сколько раз жертвы боятся края меньше, чем хищников
      ustupi = CanEat*10;       //насколько значима иерархия среди хищников
      BkColor = clBlack;        //Фон
      Height = 600;             //Высота графического окна
      Width = 780;              //Ширина графического окна
      xmin = 10;                //
      ymin = 10;                //Минимальные и максимальные значения координат,
      xmax = Width - 100;       //которые могут принимать рыбы
      ymax = Height - 140;      //

Type
fishtype = class //Описание одной стаи
           c : color;
```

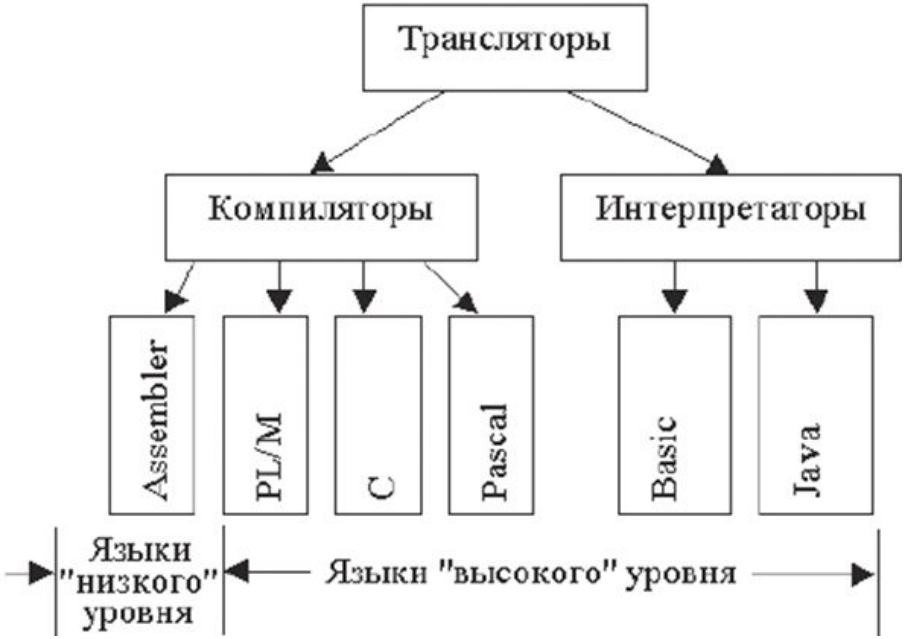
ЯП высокого уровня машинно-независимые, значительно ближе и понятнее человеку, не требуют знания архитектуры процессора. Можно использовать программу на различных ЭВМ.

ЯП высокого уровня

С помощью ЯП создается не готовая программа, а только ее текст, описывающий ранее разработанный алгоритм. Чтобы получить работающую программу, используются программы-трансляторы.

Существует два вида трансляторов:

- **интерпретаторы** – сканируют и проверяют исходный код сразу
- **компиляторы** – сканируют исходный код для генерации программы на машинном языке, которая затем выполняется отдельно.



ЯП высокого уровня

Компиляторы и интерпретаторы:

При использовании компиляторов весь исходный текст программы преобразуется в машинные коды, и именно эти коды записываются в память микропроцессора.

При использовании интерпретатора в память микропроцессора записывается исходный текст программы, а трансляция производится при считывании очередного оператора. Естественно, что быстродействие интерпретаторов намного ниже по сравнению с компиляторами, т. к. при использовании оператора в цикле он транслируется многократно.

Однако при программировании на ЯП высокого уровня объем кода, который нужно хранить во внутренней памяти, может быть значительно меньше по сравнению с исполняемым кодом. Еще одним преимуществом применения интерпретаторов является легкая переносимость программ с одного процессора на другой.

Преимущество интерпретаторов еще в том, что они допускает «непосредственный режим», который позволяет задавать компьютеру задачу и возвращать ответ, сразу после нажатия клавиши ENTER. Кроме того, интерпретаторы упрощают отладку. Можно, например, прервать обработку программы, отобразить содержимое определенных переменных, бегло просмотреть программу, а затем продолжить исполнение.

ЯП высокого уровня

Компиляторы и интерпретаторы:

Однако интерпретаторные языки имеют недостатки. Необходимо иметь копию интерпретатора в памяти все время, тогда как многие возможности интерпретатора могут не быть необходимыми для исполнения конкретной программы. При исполнении программных операторов интерпретатор должен сначала сканировать каждый оператор, а затем выполнить операцию. Операторы в циклах сканируются излишне много.

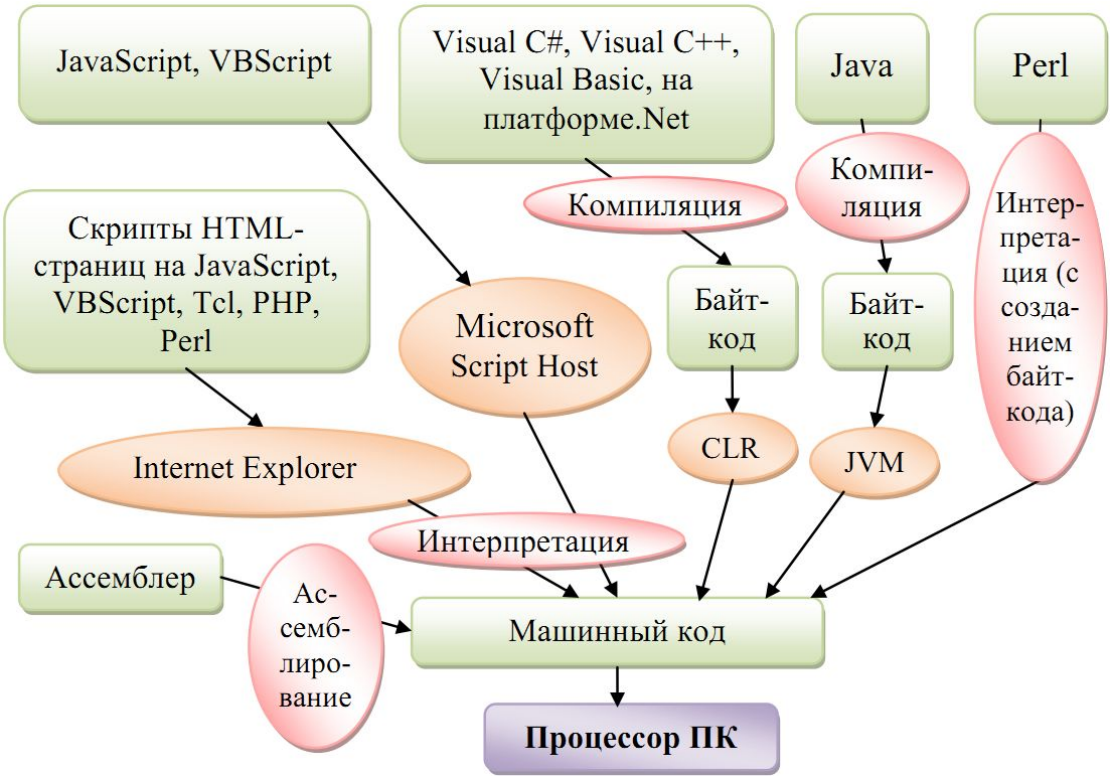
Компилятор – это транслятор текста на машинный язык, который считывает исходный текст. Он оценивает его в соответствии с синтаксической конструкцией языка и переводит на машинный язык. Другими словами, компилятор не исполняет программы, он их строит. Интерпретаторы невозможно отделить от программ, которые ими прогоняются, компиляторы делают свое дело и уходят со сцены. При работе с компилирующим языком, вам придется мыслить о ваших программах в признаках двух главных фаз их жизни: периода компилирования и периода прогона. Большинство программ будут прогоняться в 4..10 (а иногда в 100!) раз быстрее их интерпретаторных эквивалентов.

Оборотная сторона монеты состоит в том, что программы, расходующие большую часть времени на работу с файлами на дисках или ожидание ввода, не смогут продемонстрировать какое-то впечатляющее увеличение скорости.

ЯП высокого уровня

Для выполнения программ на ЯП высокого уровня необходимо выполнить их трансляцию (компиляцию или интерпретацию) в машинный код или компиляцию в промежуточный двоичный байт-код, который далее интерпретируется в различных ОС.

Методы исполнения программ в ОС Windows



ЯП высокого уровня

В настоящее время наибольшее распространение получили **универсальные языки** C#, C++, Basic, Pascal для разработки Windows-приложений.

На начальных этапах развития ИТ большую роль сыграли также языки Fortran, PL/1, Cobol, Algol, C и многие другие.

В связи с повсеместным распространением Интернета большое значение приобрели ЯП **для создания Интернет-приложений**: C#, PHP, Perl, JavaScript, VBScript.

Для решения задач **искусственного интеллекта** используются такие ЯП, как Lisp, Prolog и т.п.

Классификация ЯП высокого уровня

В зависимости от *метода написания программы* различают ЯП:

- **Процедурные**
- **Объектно-ориентированные**
- **Функциональные**
- **Логические**
- **Скрипты** (языки сценариев)
- **Языки, ориентированные на данные**

Классификация ЯП высокого уровня

Процедурные ЯП:

В этих ЯП программа разделяется на отдельные связанные друг с другом подпрограммы: процедуры и функции.

Головная программа и подпрограммы состоят из последовательности операторов, использующих библиотечные процедуры и функции.

Основные представители:

1. **FORTRAN** (1954) – первый процедурный язык
2. **Cobol** (1960) – создан, как язык для коммерчских приложений, эффективно работает с большими объемами данных
3. **Pascal** (1970) – последние версии имеют объектно-ориентированные возможности организации программ
4. **C**(1972) – первоначально создавался для ОС UNIX, как и Ассемблер может эффективно работать с данными
5. **Ada** (1983) – создан для Департамента Защиты США

Классификация ЯП высокого уровня

Объектно-ориентированные ЯП:

Дальнейшее развитие процедурных ЯП. Программа компонуется из объектов, предварительно созданных пользователем или библиотечных, с дальнейшим использованием их методов.

Основные представители:

1. **Simula** и **SmallTalk** (1967) – первые объектно-ориентированные ЯП
2. **C++** (1983) – широко используется во многих областях
3. **Java** (1995) – потомок C++, используется для Интернет-программирования, работает на любом компьютере вне зависимости от его архитектуры

Классификация ЯП высокого уровня

Функциональные ЯП:

Языки искусственного интеллекта.

Программа представляет собой последовательность функций и выражений, которые нужно вычислить. Основная структура данных – связной список. Программирование принципиально отличается от процедурного.

Основные представители:

1. **LISP** (1958) – «язык обработки списков», второй после Фортрана ЯП высокого уровня
2. **ML (MetaLanguage)** (1969) – создавался для автоматического доказательства теорем, оказался пригоден как ЯП общего назначения
3. **Miranda** (1985) – первый коммерческий чисто функциональный язык. Используется «ленивая» семантика.
4. **Haskell** (1990) – один из наиболее мощных функциональных языков. Дальнейшее развитие языков ML и Miranda.

Классификация ЯП высокого уровня

Логические ЯП:

Ориентированы на решение задач без описания алгоритма, языки для решения задач искусственного интеллекта.

Наиболее характерный представитель – язык PROLOG (1972), на котором написаны многие экспертные системы.

Программы на PROLOG отличается от программы на процедурном языке. Пролог-программа является собранием правил и фактов. Решение задачи достигается интерпретацией этих правил и фактов. При этом пользователю не требуется обеспечивать детальную последовательность инструкций, чтобы указать, каким образом осуществляется управление ходом вычислений на пути к результату. Вместо этого он только определяет возможные решения задачи и обеспечивает программу фактами и правилами, которые позволяют ей отыскать требуемое решение.

Классификация ЯП высокого уровня

Скрипты (языки сценариев):

Объектно-ориентированные языки для создания программ, исполняемых в определенной среде.

Например, программы на языках VBScript, JScript могут исполняться как сервером сценариев Windows Script Host, так и Интернет-браузером, когда в html-документа есть ссылка на файл скрипта.

Также программы на этих языках могут включаться в тело html-документа.

Языки Perl (1986) и Python (1990) появились как скриптовые языки для Unix, позже появились версии для Windows и Macintosh.

В программах на скриптовых языках часто используются вызовы процедур и функций системных библиотек, знание которых необходимо для использования этих языков.

Классификация ЯП высокого уровня

Языки, ориентированные на данные:

Созданы специально для работы с одним определенным типом данных.

Например, APL (1961) настроен на работу с матрицами и векторами без циклов. Snobol (1962) и его преемник Icon (1974) предназначены для обработки строковых данных, SETL (1969) – для работы с множествами.

Особое положение занимают ЯП для работы с базами данных – 4GL, 3GL, PL/SQL, FoxPro, а также библиотеки работы с базами данных – ActiveX Data Objects, Microsoft Data Access Objects, доступ к объектам которым можно получить из многих универсальных языков.

Большие возможности по работе с документами Word, электронными таблицами Excel и базами данных Access имеет встроенный язык пакета MS Office – Visual Basic for Applications (VBA).

Принципы программирования

Принципы программирования

- **алгоритмическое**
- **структурное**
- **объектно-ориентированное**

Алгоритмическое программирование

Программа представляет собой линейную последовательность операторов присваивания, цикла и условия.

Так решают не очень сложные задачи, объем программы – несколько сотен строк кода.

При бóльшем объеме понятность кода резко падает из-за того, что общая структура алгоритма теряется за конкретными операторами языка, выполняющими элементарные действия. Логика становится совсем запутанной, при попытке исправить один оператор вносится несколько новых ошибок

Структурное программирование

Для средних по объему приложений (несколько тысяч строк кода).

Структура программы должна отражать структуру решаемой задачи, чтобы алгоритм решения был ясно виден из исходного текста.

Для этого используют **подпрограммы** – наборы операторов, выполняющих нужное действие и не зависящих от других частей кода. Программа разбивается на множество мелких (до 50 операторов) подпрограмм, каждая из которых выполняет одно из действий. Комбинируя подпрограммы, формируют итоговый алгоритм уже не из простых операторов, а из блоков, имеющих определенную смысловую нагрузку, причем обращаться к таким блокам можно по названиям. Получается, что подпрограммы – это новые операторы, определяемые программистом.

Структурное программирование

Подпрограммы позволяют проектировать приложения «сверху вниз» – т.н. нисходящее проектирование. Сначала выделяется несколько подпрограмм, решающих самые глобальные задачи, потом каждый из этих модулей детализируется на более низком уровне, разбиваясь, в свою очередь, на другие подпрограммы, и так происходит до тех пор, пока не будет решена вся задача.

Это позволяет человеку мыслить на предметном уровне, не опускаясь до конкретных операторов и переменных. Кроме того, появляется возможность не реализовывать сразу некоторые подпрограммы, а временно откладывать, пока не будут закончены другие части. Например, если нужно вычислить сложную математическую функцию, то выделяется отдельная подпрограмма такого вычисления, но реализуется она временно одним оператором, который просто присваивает какое-то значение. Когда все приложение будет написано и отлажено, тогда можно приступить к реализации этой функции.

Структурное программирование

Немаловажно, что небольшие подпрограммы значительно проще отлаживать, что повышает общую надежность всей программы.

Подпрограммы можно использовать повторно. С системами программирования обычно поставляют большие библиотеки стандартных подпрограмм, которые позволяют значительно повысить производительность труда.

Подпрограммы бывают двух видов:

- **Процедуры** – просто выполняют группу операторов;
- **Функции** – вдобавок вычисляют некоторое значение и передают его обратно в главную программу.

Структурное программирование

Достоинства :

1. Программы, даже довольно крупные, становятся легко читаемыми, их без труда понимает не только ее автор, но и другие программисты. Это позволяет разрабатывать большие программы силами коллектива программистов и сопровождать их в течение многих лет.

2. В хорошо структурированных программах алгоритм представлен достаточно наглядно, что позволяет обойтись без блок-схем. По сути сама программа является блок-схемой.

3. Упрощается тестирование и отладка программы

Структурное программирование

Принципы:

1. Программа разбивается на отдельные модули, образующие иерархическую структуру.
2. Разработка сложной программы ведется «сверху вниз» – сначала разрабатывается головная программа, из которой вызываются подпрограммы более низкого уровня иерархии, которые на первом этапе не выполняют никаких действий («заглушки»). Далее разрабатывают подпрограммы, начиная с верхних уровней.
3. Названия в программах должны быть значимыми, говорить об их назначении.
4. Текст программы пишется наглядно с использованием отступов.
5. Запрещается использование безусловных переходов и меток. Вместо этого используется вызов подпрограмм

Объектно-ориентированное программирование

В 80-х годах в программировании возникло новое направление, основанное на понятии объекта.

Реальные объекты окружающего мира:

- имеют набор свойств
- способны разными методами изменять эти свойства
- способны реагировать на события, возникающие как в окружающем мире, так и внутри самого объекта.

Именно в таком виде в ЯП и реализовано понятие объекта:

Объект – абстрактная именованная сущность, обладающая набором свойств, которая может выполнять определенные операции и реагировать на внешние события своими методами.

Объектно-ориентированное программирование

Появление объектов качественно повлияло на производительность труда программистов. Максимальный объем приложений, которые может создать группа из 10 программистов, за несколько лет увеличился до миллионов строк кода (!), при этом удалось добиться высокой надежности программ и повторно использовать ранее созданные объекты в других задачах.

Объекты могут иметь идентичную структуру и отличаться только значениями свойств. В таких случаях в программе создается новый тип, основанный на единой структуре объекта. Он называется **классом**, а каждый конкретный объект, имеющий структуру этого класса, называется **экземпляром** класса.

Объектно-ориентированное программирование

Принципы:

1. **Инкапсуляция** – объединение данных с процедурами и функциями в единый блок кода (свойства и методы рассматриваются как поля объекта)

2. **Наследование** – возможность создания на основе имеющегося класса новые классы с наследованием всех его свойств и методов и добавлением собственных;

3. **Полиморфизм** – присвоение действию одного имени, которое затем совместно используется вниз и вверх по иерархии объектов, причем каждый объект иерархии выполняет это действие способом, подходящим именно ему.

Спасибо за внимание!!!