Информатика

Для курса 1 групп БАСО-01-17, БАСО-02-17, БАСО-03-17, БАСО-04-17,

По специальности
10.05.03 «Информационная безопасность автоматизированных систем»

1 семестр
2017-2018 учебный год

Москва, 2017

Содержание:

Кашкин Евгений Владимирович

кандидат технических наук,

доцент кафедры «Информатика»,

зам. зав. кафедры «Информатика»

e.kashkin@gmail.com

Тема №1
«Базовые понятия и определения »

Цели занятия

- **Задача. Решение задачи;**
- **Алгоритм. Свойства алгоритмов;**
- **Программа. Программное обеспечение;**
- **Информатика. Информация. Информационная технология;**
- **Данные. Числа в арифметике;**
- **Выражения. Операнды. Знаки операций. Идентификаторы. Константы;**
- **Законы арифметики (коммутативность, ассоциативность, дистрибутивность);**

Информа́тика (от информация и автоматика) — наука о методах и процессах сбора, хранения, обработки, передачи, анализа и оценки информации с применением компьютерных технологий, обеспечивающих возможность её использования для принятия решений.

Информатика включает дисциплины, относящиеся к обработке информации в вычислительных машинах и вычислительных сетях: как абстрактные, вроде анализа алгоритмов, так и конкретные, например разработка языков программирования и протоколов передачи данных.

Программа

- **Программа — термин, в переводе означающий «предписание», то есть предварительное описание предстоящих событий или действий, регламентированное алгоритмом.**

Информация.

Информация **для человека – это** ***знания*****, которые он получает из различных источников.**

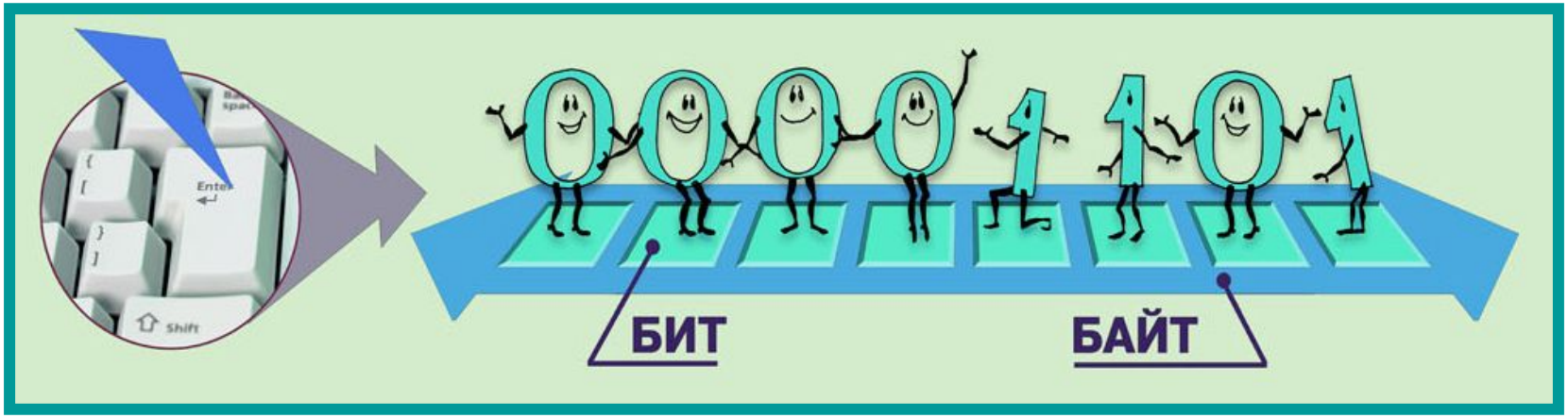
Свойства информации

- **Понятность**
- **Полезность**
- **Достоверность**
- **Актуальность**
- **Полнота**
- **Точность**

Единицы измерения информации

Единицами измерения информации являются биты (0 и 1) и байты.

1 байт – это 8 битов.



Количество информации

$$2^{i} = N$$

i – число бит информации

N – количество возможных событий

Размерности и их соотношения

- 8 бит = 1 байт
- 1 Кб = 1024 байта
- 1 Мб = 1024 Кб
- 1 Гб = 1024 Мб
- 1 Тб = 1024 Гб

Алгоритм

Алгоритм — набор инструкций, описывающих порядок действий исполнителя для достижения результата решения задачи за конечное число действий.

Алгоритмизация – процесс разработки алгоритма (плана действий) для решения задачи. Алгоритмы реализованные на компьютере решают сложные задачи:

- **в медицине;**
- **в производстве;**
- **в сфере безопасности**

Способы записи алгоритма

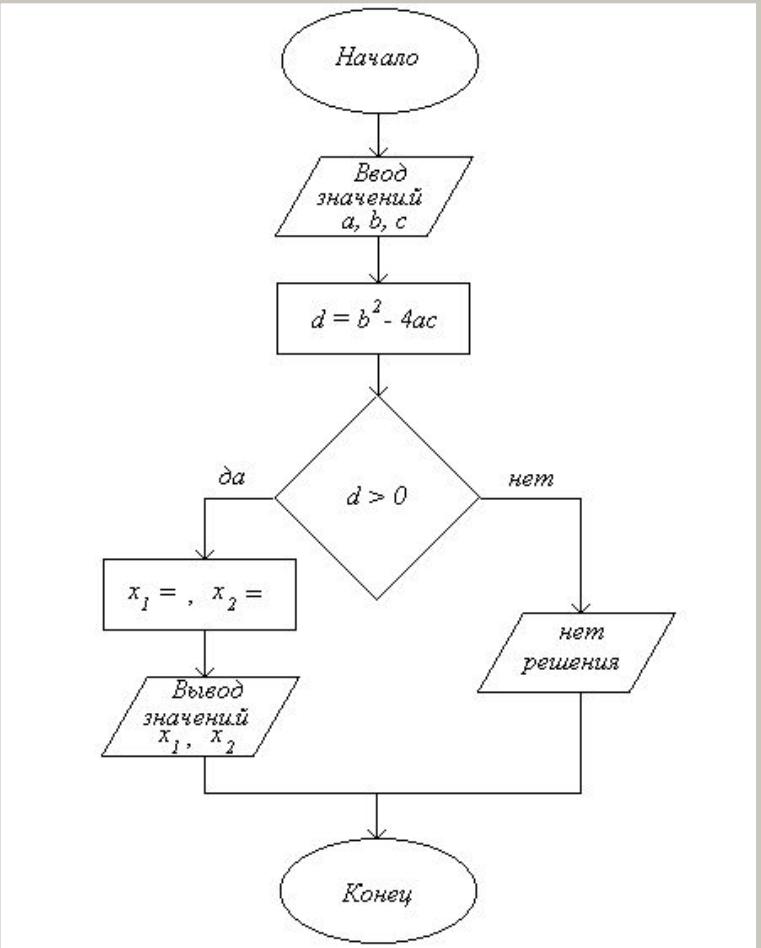
- Словесно-формульный (на естественном языке с использованием математических формул)
- Графический (блок-схема)
- На языке программирования (программа)

Словесно-формульный способ

$$A x^2 + B x + C = 0$$

1. Начать.
2. Ввод A, B, C.
3. $D = B^2 - 4 A C$.
4. Если $D < 0$, то идти к п. 6.
5. Если $D > 0$, то идти к п. 8.
6. Действительных корней нет.
7. Идти к п. 10.
8. $X_1 = (- B -$) / 2 A ; $X_2 = (- B +$) / 2 A .
9. Вывести значения X_1 и X_2.
10. Закончить.

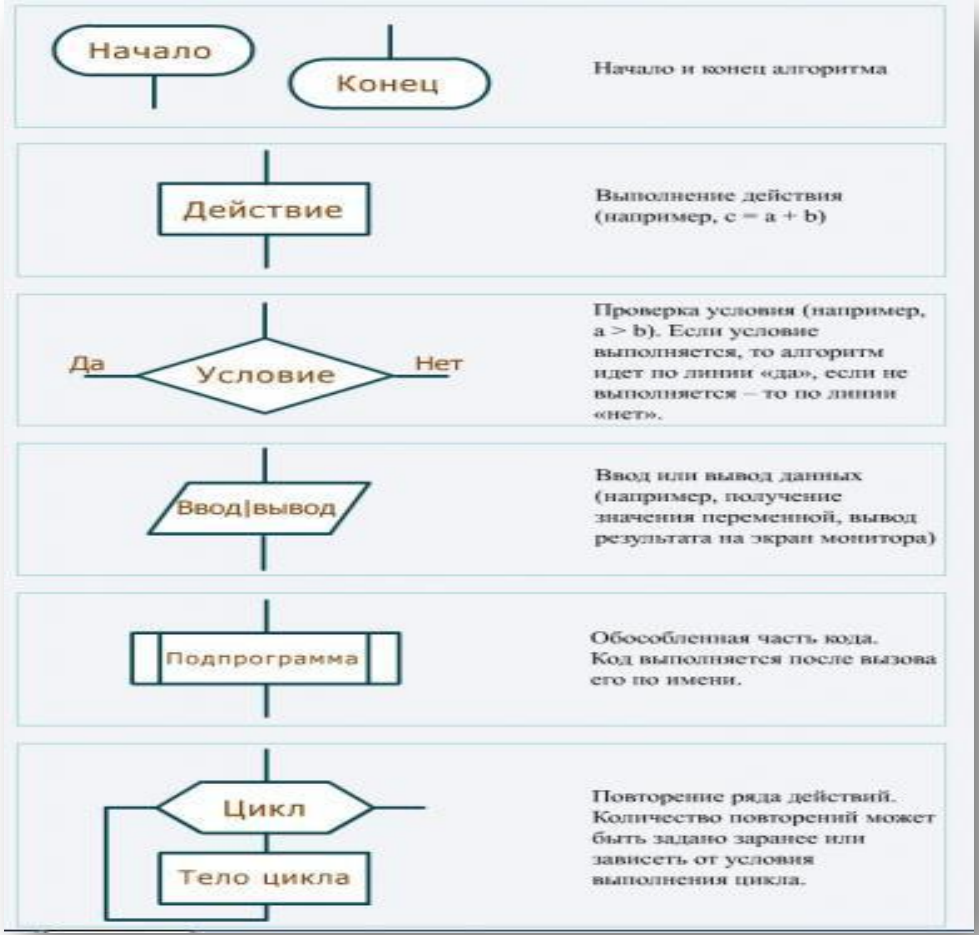
Графический способ



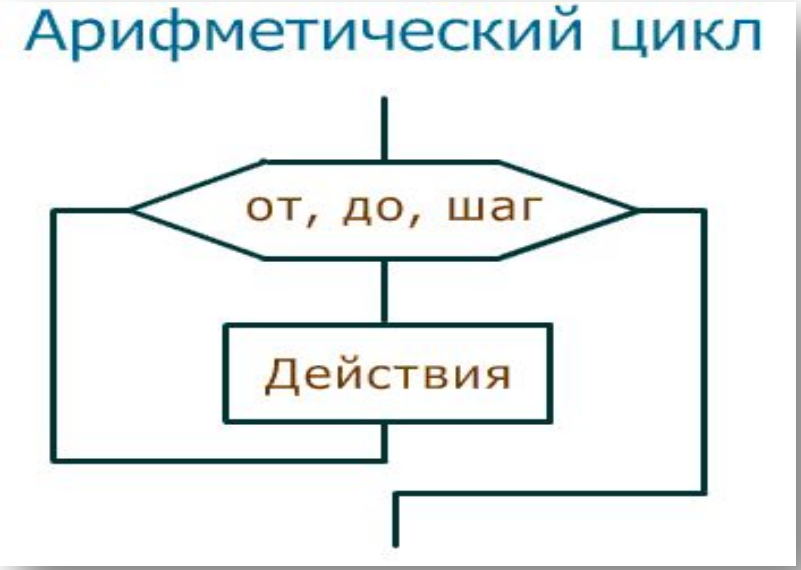
Запись на языке программирования

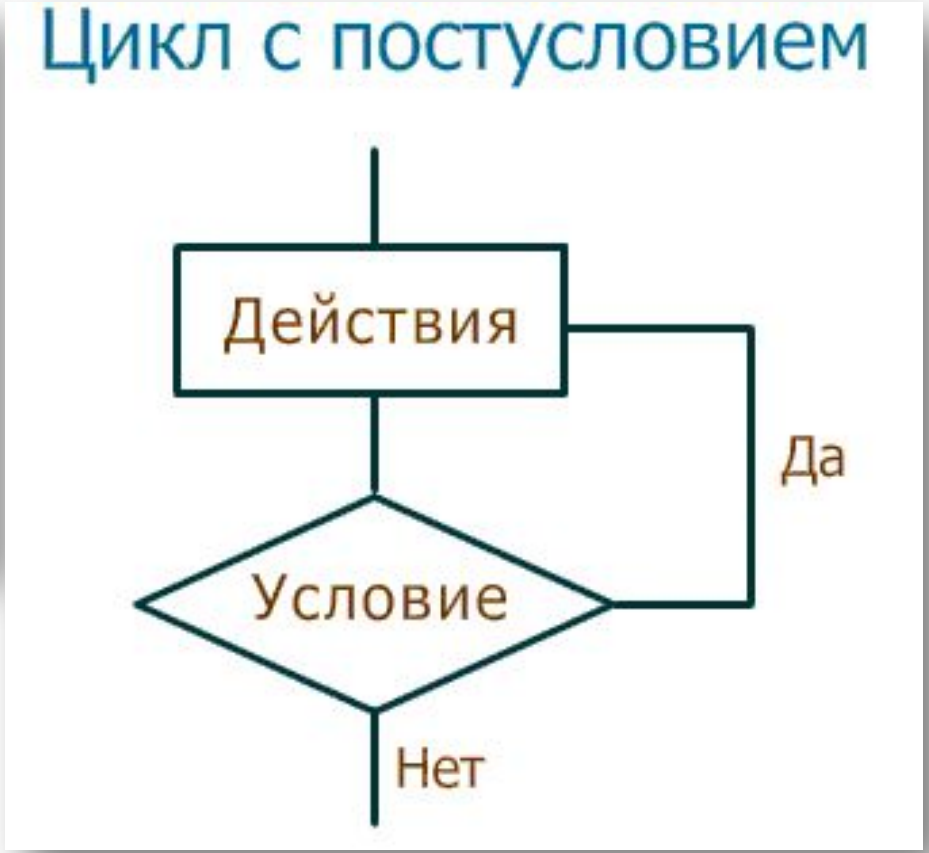
```
program example;
var a,b,c: integer;d,x1,x2:real;
begin
    writeln ('a,b,c');
    readln (a,b,c);
    d:=sqr(b)-4*a*c;
    if d<0 then
          begin
            writeln ('no korny');
          end
     else
           begin
              x1:=(-b-sqrt(d))/2*a;
              x2:=(-b+sqrt(d))/2*a;
              writeln ('x1=',x1,' x2=',x2);
           end;
    readln;
end.
```

Условные графические обозначения в блок-схемах

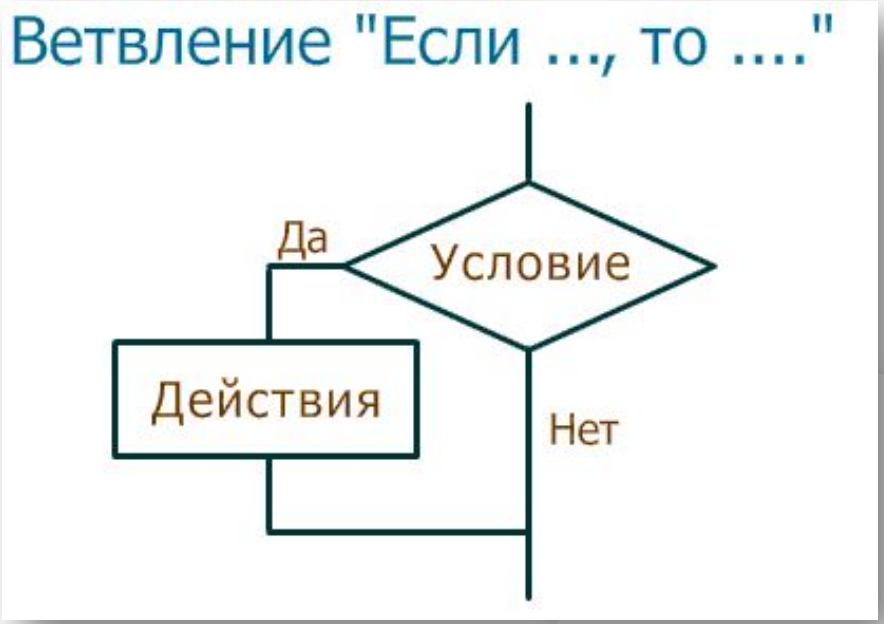


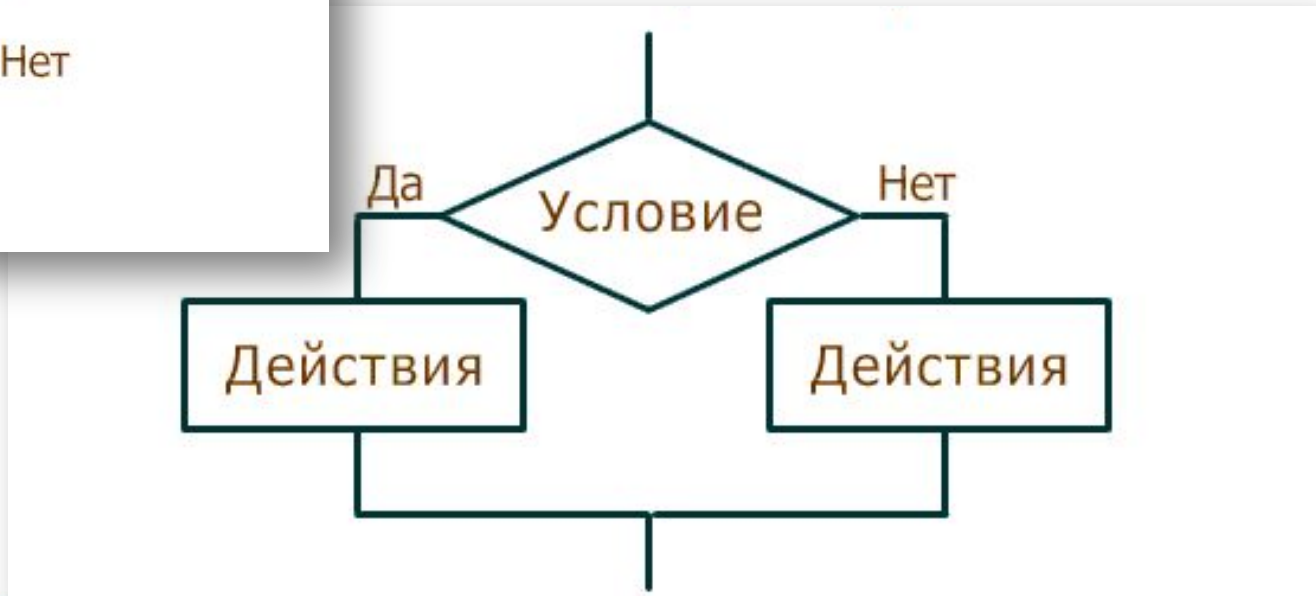
Цикл


Арифметический цикл
от, до, шаг
Действия


Цикл с постусловием
Действия
Да
Условие
Нет

Ветвление


Ветвление "Если ..., то ...."
Да
Условие
Действия
Нет


Да
Условие
Нет
Действия
Действия

Множественное ветвление

Информационные технологии

Информацио́нные техноло́гии (ИТ, от англ. *information technology*, IT) — широкий класс дисциплин и областей деятельности, относящихся к технологиям создания, сохранения, управления и обработки данных, в том числе с применением вычислительной техники. В последнее время под информационными технологиями чаще всего понимают компьютерные технологии. В частности, ИТ имеют дело с использованием компьютеров и программного обеспечения для создания, хранения, обработки, ограничения к передаче и получению информации. Специалистов по компьютерной технике и программированию часто называют ИТ-специалистами.

Данные.

Это значения, необходимые для выполнения программы, хранящиеся в ячейках памяти.

Типы данных **– числовые и текстовые**

***Операции с данными*:**

- **ввод (сбор) данных — накопление данных с целью обеспечения достаточной полноты для принятия решений;**
- **формализация данных — приведение данных, поступающих из разных источников, к одинаковой форме, для повышения их доступности;**
- **фильтрация данных — это отсеивание «лишних» данных, в которых нет необходимости для повышения достоверности и адекватности;**
- **сортировка данных — это упорядочивание данных по заданному признаку с целью удобства их использования;**
- **архивация — это организация хранения данных в удобной и легкодоступной форме;**
- **защита данных — включает меры, направленные на предотвращение утраты, воспроизведения и модификации данных;**
- **транспортировка данных — приём и передача данных между участниками информационного процесса;**
- **преобразование данных — это перевод данных из одной формы в другую или из одной структуры в другую.**

Знаки операций

Знак	Операция	Типы операндов	Тип результата
+	Сложение	Целые Хотя бы один вещественный	Целый Вещественный
-	Вычитание	Целые Хотя бы один вещественный	Целый Вещественный
*	Умножение	Целые Хотя бы один вещественный	Целый Вещественный
/	Деление	Целые или вещественные	Вещественный

Идентификаторы

Идентификаторы **- имена объектов и конструкций программы (меток, констант, типов, переменных, типов, процедур, функций, объектов, модулей, программ, полей в записях и т.д.).**

Имя состоит из латинской буквы, за которой могут следовать латинские буквы, цифры или символ подчеркивания.

Примеры:

first – правильно

a1 – правильно

1q – НЕ правильно

Константы

Это особый вид переменных, значение которых не меняется на протяжении работы всей программы.

Примеры:

Pi=3.14

M12=‘Декабрь’

Вопросы для самоподготовки:

- Что такое информация?
- Что такое алгоритм и каковы его свойства?
- В чем измеряется информация?
- Какие виды программного обеспечения существуют?

Выводы по теме 1

В рамках данной темы были получены знания в информатике, включающие в себя разбор понятий «задача», «информация», «операнд» и др.

Вопросы для самостоятельного изучения:

1. Программное обеспечение. Виды программного обеспечения
2. Числа в арифметике. Операнды. Выражения
3. Законы арифметики (ассоциативность, коммутативность, дистрибутивность
4. Свойства алгоритмов

Тема №2
«Представление данных. Принцип программного управления»

Цели занятия

- Основы алгебры логики;
- Системы счисления; Связи между системами счисления;
- Основы арифметики двоичных чисел;
- Принцип программного управления. Базовая архитектура и структура ЭВМ. Принцип фон Неймана;
- Единицы измерения ёмкости запоминающих устройств;
- Представление целых и вещественных чисел в памяти ЭВМ;
- Диапазоны представления чисел в двоичной системе счисления;
- Представление символьной информации. Кодовые таблицы;
- Понятие типа данных.

Основы алгебры логики.

Алгебра – это раздел математики, предназначенный для описания действий над переменными величинами, которые принято обозначать строчными буквами латинского алфавита – a, b, x, y и т.д. Действия над переменными величинами записываются в виде математических выражений.

Алгеброй логики называется аппарат, который позволяет выполнять действия над высказываниями.

Алгебру логику называют также алгеброй Буля, или булевой алгеброй, по имени английского математика Джорджа Буля, разработавшего в XIX веке ее основные положения.

Основы алгебры логики.

В булевой алгебре высказывания принято обозначать прописными латинскими буквами: A, B, X, Y. В алгебре Буля введены три основные логические операции с высказываниями: сложение, умножение, отрицание. Определены аксиомы (законы) алгебры логики для выполнения этих операций. Действия, которые производятся над высказываниями, записываются в виде логических выражений.

Простое логическое выражение

Простое логическое выражение состоит из одного высказывания и не содержит логические операции. В простом логическом выражении возможно только два результата — либо «истина», либо «ложь».

Примеры:

- ***a>b***
- ***существует a***

Сложное логическое выражение

Сложное логическое выражение содержит высказывания, объединенные логическими операциями. По аналогии с понятием функции в алгебре сложное логическое выражение содержит аргументы, которыми являются высказывания.

В качестве основных логических операций в сложных логических выражениях используются следующие:

- **НЕ (логическое отрицание, инверсия);**
- **ИЛИ (логическое сложение, дизъюнкция);**
- **И (логическое умножение, конъюнкция).**

Пример:

login='Vasya' и pass='123'

Таблицы истинности

Все операции алгебры логики определяются таблицами истинности значений. Таблица истинности определяет результат выполнения операции для всех возможных логических значений исходных высказываний. Количество вариантов, отражающих результат применения операций, будет зависеть от количества высказываний в логическом выражении, например:

- **таблица истинности одноместной логической операции состоит из двух строк: два различных значения аргумента — «истина» (1) и «ложь» (0) и два соответствующих им значения функции;**
- **в таблице истинности двуместной логической операции — четыре строки: 4 различных сочетания значений аргументов — 00, 01, 10 и 11 и 4 соответствующих им значения функции;**
- **если число высказываний в логическом выражении N, то таблица истинности будет содержать 2^N строк, так как существует 2^N различных комбинаций возможных значений аргументов.**

Операция НЕ — логическое отрицание (инверсия)

Логическая операция НЕ применяется к одному аргументу, в качестве которого может быть и простое, и сложное логическое выражение. Результатом операции НЕ является следующее:

- если исходное выражение истинно, то результат его отрицания будет ложным;
- если исходное выражение ложно, то результат его отрицания будет истинным.

Для операции отрицания НЕ приняты следующие условные обозначения:
не A, $\bar{A}$, not A, ¬A.
Результат операции отрицания НЕ определяется следующей таблицей истинности:
Результат операции отрицания истинен, когда исходное высказывание ложно, и наоборот.

A	не A
0	1
1	0

Операция ИЛИ — логическое сложение (дизъюнкция, объединение)

Логическая операция ИЛИ выполняет функцию объединения двух высказываний, в качестве которых может быть и простое, и сложное логическое выражение. Высказывания, являющиеся исходными для логической операции, называют аргументами. Результатом операции ИЛИ является выражение, которое будет истинным тогда и только тогда, когда истинно будет хотя бы одно из исходных выражений.

- **Применяемые обозначения: A или B, A V B, A or B.**
- **Результат операции ИЛИ определяется следующей таблицей истинности:**

A	B	A или B
0	0	0
0	1	1
1	0	1
1	1	1

Операция И — логическое умножение (конъюнкция)

Логическая операция И выполняет функцию пересечения двух высказываний (аргументов), в качестве которых может быть и простое, и сложное логическое выражение. Результатом операции И является выражение, которое будет истинным тогда и только тогда, когда истинны оба исходных выражения.

- **Применяемые обозначения: A и B, A ∧ B, A & B, A and B.**
- **Результат операции И определяется следующей таблицей истинности:**

A	B	A и B
0	0	0
0	1	0
1	0	0
1	1	1

Операция «ЕСЛИ-ТО» — логическое следование (импликация)

Эта операция связывает два простых логических выражения, из которых первое является условием, а второе — следствием из этого условия.

Применяемые обозначения:

- **если A, то B; A влечет B; if A then B; A→ B.**

Таблица истинности:

A	B	A → B
0	0	1
0	1	1
1	0	0
1	1	1

Операция «A тогда и только тогда, когда B» (эквивалентность, равнозначность)

Применяемое обозначение: A ↔ B, A ~ B.

Таблица истинности:

A	B	A↔B
0	0	1
0	1	0
1	0	0
1	1	1

Примеры:

1. ***День сменяет ночь тогда и только тогда, когда солнце скрывается за горизонтом;***
2. ***Добиться результата в спорте можно тогда и только тогда, когда приложено максимум усилий.***

Приоритет логических операций

1. Действия в скобках
2. Инверсия
3. Конъюнкция (&)
4. Дизъюнкция (V)
5. Импликация (→)
6. Эквивалентность (↔)

$$(A \,\&\, B) \vee (\overline{A} \,\&\, B)$$

Закон противоречия

$$A \,\&\, \overline{A} = 0;\; B \,\&\, \overline{B} = 0.$$

Закон исключенного третьего

$$A \vee \overline{A} = 1;\ B \vee \overline{B} = 1.$$

Закон двойного отрицания

$$\overline{\overline{A}} = A; \quad \overline{\overline{B}} = B.$$

Законы де Моргана

$$\overline{A \vee B} = \overline{A} \,\&\, \overline{B};\ \overline{A \,\&\, B} = \overline{A} \vee \overline{B}.$$

Законы повторения

$$A \& A = A; A \vee A = A;$$

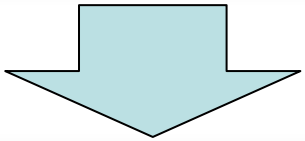
$$B \& B = B; B \vee B = B.$$

Законы склеивания

$$(A \,\&\, B) \vee (\overline{A} \,\&\, B) = B;\ (A \vee B) \,\&\, (\overline{A} \vee B) = B.$$

Пример использования законов алгебры логики

$$F = A \vee \overline{A \,\&\, B} \vee \overline{\overline{A} \vee B},$$



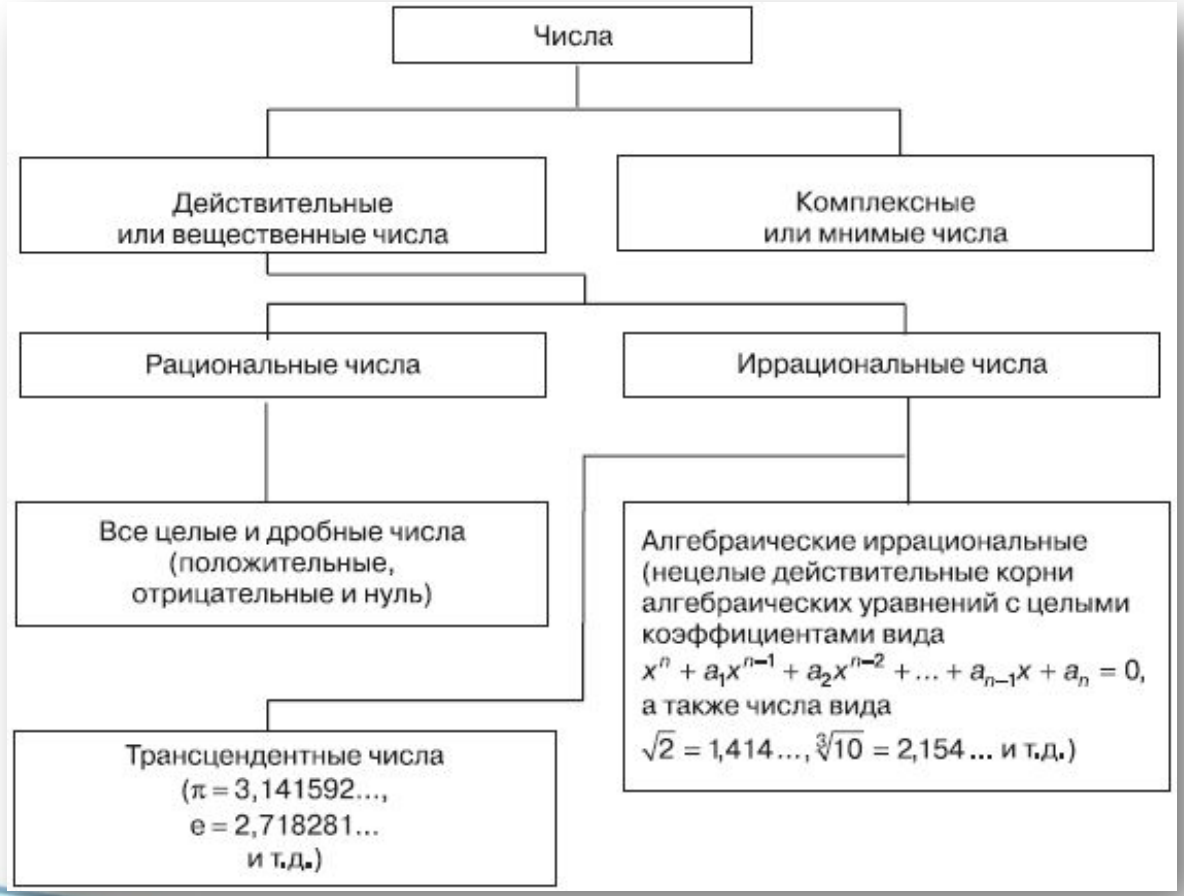
$$F = A \vee \overline{A \,\&\, B} \vee \overline{\overline{A} \vee B} = A \vee \overline{A} \vee \overline{B} \vee A \,\&\, \overline{B} =$$

$$= (A \vee \overline{A}) \vee \overline{B}(1 \vee A) = 1 \vee \overline{B} = 1.$$

Применение таблиц истинности к логическим функциям

A	B	C	$F = A \vee B \vee C$
0	0	0	0
1	0	0	1
0	1	0	1
1	1	0	1
0	0	1	1
1	0	1	1
0	1	1	1
1	1	1	1

Системы счисления



Числа записываются с использованием особых знаковых систем, которые называются системами счисления, в них числа записываются по определенным правилам с помощью символов некоторого алфавита, называемых цифрами.

Двоичная арифметика. Сложение.

Пример 1. **Сложить двоичные числа:**

$$100111_2 + 11101_2.$$

РЕШЕНИЕ:

$$\begin{array}{r} 100111 \\ + \quad 11101 \\ \hline 1000100 \end{array}$$

В итоге получаем:

$$100111_2 + 11101_2 = 1000100_2$$

Двоичная арифметика. Вычитание.

Если нам необходимо найти разность двух двоичных чисел, то нужно:

1. **Сравнять количество разрядов обоих чисел;**
2. **Инвертировать вычитаемое путем замены нулей единицами, а единицы – нулями;**
3. **Добавить дополнительную единицу;**
4. **Сложить оба числа;**
5. **Удалить единицу самого старшего разряда.**

Двоичная арифметика. Вычитание.

$$110011_2 - 001001_2$$

=

$$\begin{array}{r} + \quad 110011 \\ \hline 110100 \end{array}$$

Отбрасываем единицу старшего разряда, получаем:

101010.

В итоге получаем:

$$110011_2 - 1001_2 = 101010_2$$

Двоичная арифметика. Умножение.

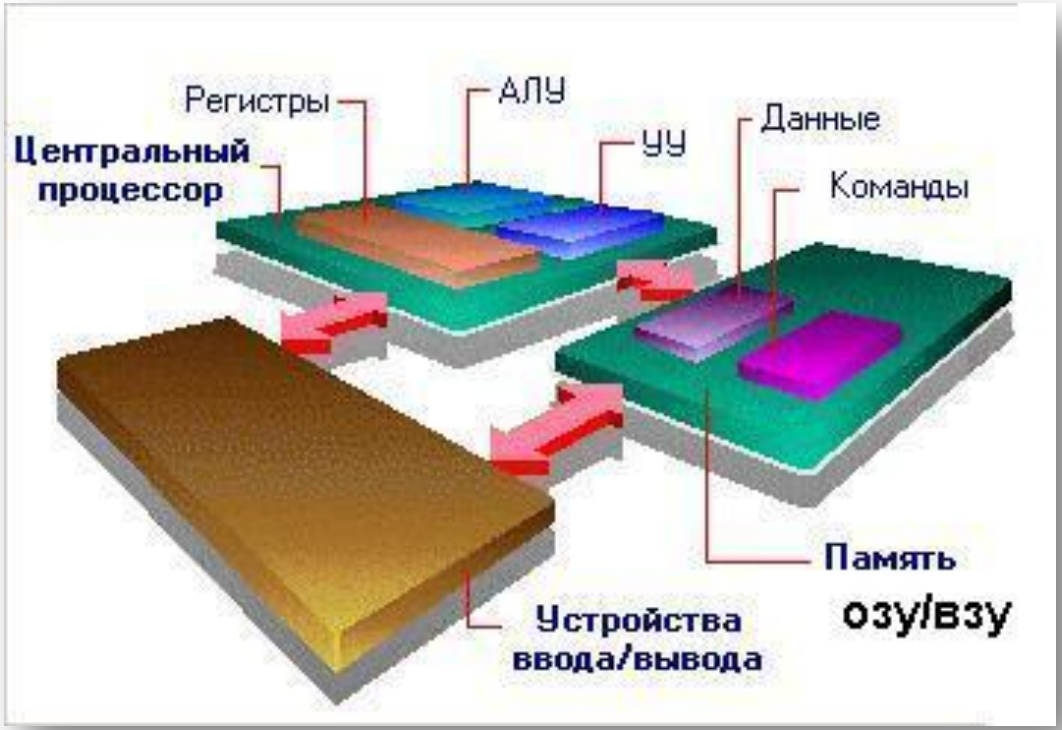
$1101101_2 * 101_2 =$

$$
\begin{array}{r}
1101101 \\
* \quad 101 \\
\hline
1101101 \\
+\ 1101101 \\
\hline
100010000
\end{array}
$$

Получаем:

$$1101101_2 * 101_2 = 1000100001_2$$

Базовая структура ЭВМ. Принцип фон Неймана.

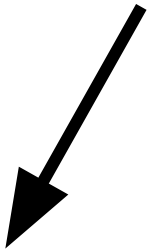


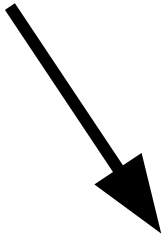
ЭВМ образца 1967г

Системный блок

Представление целых и вещественных чисел в памяти ЭВМ

Целые числа





Целые числа без знака

(только положительные)

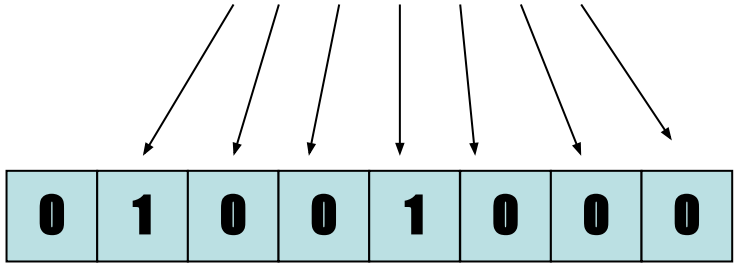
Целые числа со знаком (положительные и отрицательные)

Целые числа без знака

Обычно занимают в памяти один или два байта.

В однобайтовом формате значения от 00000000_2 до 11111111_2 (0…255)

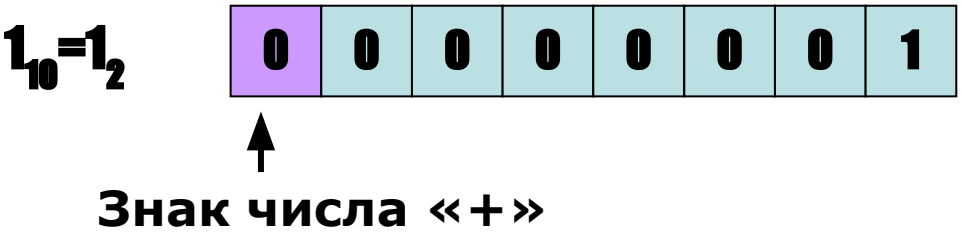
Пример $72_{10}=1001000_2$



Целые числа со знаком

Обычно занимают в памяти компьютера 1, 2 или 4 байта, при этом самый левый (старший) разряд содержит информацию о знаке числа.

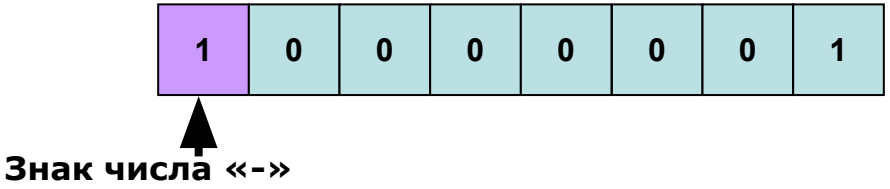
Знак «+» кодируется 0, а «-» - 1



Способы записи целых чисел в памяти компьютера

В компьютерной технике применяются три формы записи (кодирования) целых отрицательных чисел: прямой код, обратный код, дополнительный код.

Прямой код



Обратный код

Получается инвертированием всех цифр двоичного кода абсолютной величины числа, включая разряд знака: нули заменяются единицами, а единицы – нулями.

Пример

Число: -1.

Код модуля числа: 0 0000001.

Обратный код числа: 1 1111110.

1	1	1	1	1	1	1	0

Дополнительный код

Получается образованием обратного кода с последующем прибавлением единицы к его младшему разряду.

Пример

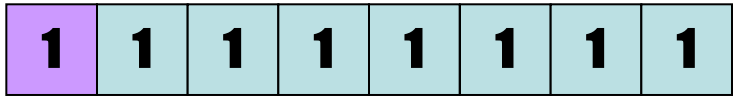
Число: -1.

Код модуля числа: 0 0000001.

Обратный код числа: 1 1111110

+1

1 1111111



Представление символьной информации

Для представления текстовой информации достаточно 256 различных символов.

$N = 2^I$, $256 = 2^I$, $2^8 = 2^I$,I = 8 битов

Для кодирования каждого знака требуется количество информации, равное 8 битам.

Для представления текста в памяти компьютера необходимо представить его в двоичной знаковой системе.

Каждому знаку необходимо поставить в соответствие уникальный двоичный код в интервале от 00000000 до 11111111 (в десятичном коде от 0 до 255)

Таблицы символов

Для представления символов и соответствующих им кодов используется *кодовая таблица*.

В качестве стандарта во всем мире принята таблица ASCII (American Standard Code for Information Interchange – Американский стандартный код для обмена информацией). Условно таблица разделена на части:

- от 0 до 32 коды соответствуют операциям;
- с 33 по 127 соответствуют символам латинского алфавита, цифрам, знакам арифметических операций и знакам препинания;
- со 128 по 255 являются национальными.

Представление данных. Принцип программного управления

32		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	□	11111111

Типы данных

1. Целые числа
2. Вещественные числа (дробные)
3. Символы
4. Строки
5. Логический операнд (да/нет, true/false)

Вопросы для самоподготовки:

- Что изучает алгебра логики?
- Какие основные операции присутсвуют в алгебре логики?
- Каких типов бывают системы счисления и в чем их отличие?
- Какие основные элементы компьютера образуют модель фон Неймана?
- Какие способы представления целых чисел существуют и в чем их отличие?
- Как кодируется символьная информация в ЭВМ?

Выводы по теме 2

В рамках данной темы были получены знания в области алгебры логики. Изучены основные логические операции. Рассмотрены системы счисления и способы перевода чисел из одной системы в другую. Обозначены элементы структурной схемы ЭВМ. Даны базовые техники хранения целыхи символьных данных в памяти ЭВМ.

Вопросы для самостоятельного изучения:

1. Системы счисления. Позиционные и непозиционные системы счисления
2. Системы счисления со связанными основаниями
3. Умножение в двоичной арифметике
4. Кодирование вещественных чисел в памяти ЭВМ

Тема №3
«Методологии и языки программирования»

Цели занятия

- **Стадии и этапы разработки программ. Проектирование. Реализация.**
- **Проблемы программирования;**
- **Методологии программирования. Классификация методологий программирования (структурное, объектно-ориентированное, логическое, функциональное, программирование в ограничениях).**
- **Структурное программирование. Базовые принципы (пошаговая детализация, модульное структурное программирование);**
- **Объектно-ориентированное программирование. Базовые принципы (абстрагирование; инкапсуляция; наследование, полиморфизм);**
- **Языки программирования. Классификация.**

Стадии и этапы разработки программ.

Определяются стандартами:

- **ГОСТ 34.601-90**
- **ISO/IEC 12207:2008 «System and software engineering — Software life cycle processes» (российский аналог — ГОСТ Р ИСО/МЭК 12207-2010 Информационная технология. Системная и программная инженерия. Процессы жизненного цикла программных средств)**

Модели жизненного цикла

Модель жизненного цикла ПО — структура, определяющая последовательность выполнения и взаимосвязи процессов, действий и задач на протяжении жизненного цикла. Модель жизненного цикла зависит от специфики, масштаба и сложности проекта и специфики условий, в которых система создается и функционирует.

Модель ЖЦ ПО включает в себя:

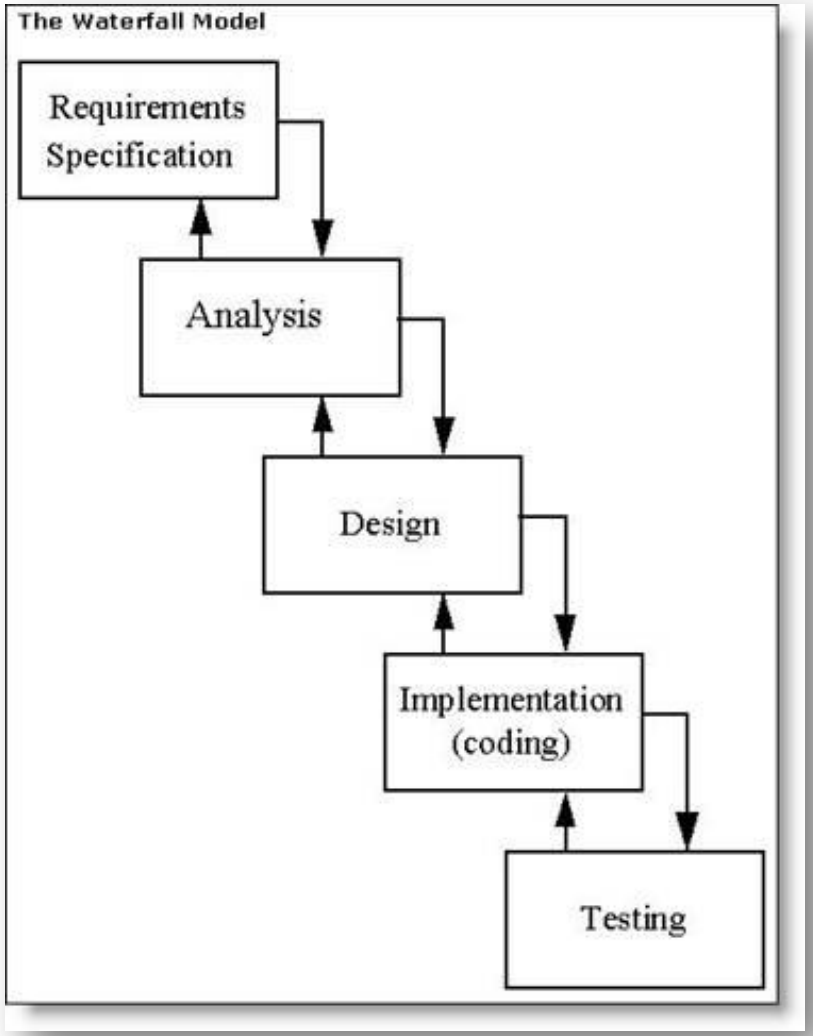
- **Стадии;**
- **Результаты выполнения работ на каждой стадии;**
- **Ключевые события — точки завершения работ и принятия решений.**

Водопадная (каскадная, последовательная) модель

Водопадная модель жизненного цикла была предложена в 1970 г. Уинстоном Ройсом. Она предусматривает последовательное выполнение всех этапов проекта в строго фиксированном порядке. Переход на следующий этап означает полное завершение работ на предыдущем этапе. Требования, определенные на стадии формирования требований, строго документируются в виде технического задания и фиксируются на все время разработки проекта. Каждая стадия завершается выпуском полного комплекта документации, достаточной для того, чтобы разработка могла быть продолжена другой командой разработчиков.

Этапы проекта в соответствии с каскадной моделью:

- **Формирование требований;**
- **Проектирование;**
- **Реализация;**
- **Тестирование;**
- **Внедрение;**
- **Эксплуатация и сопровождение.**

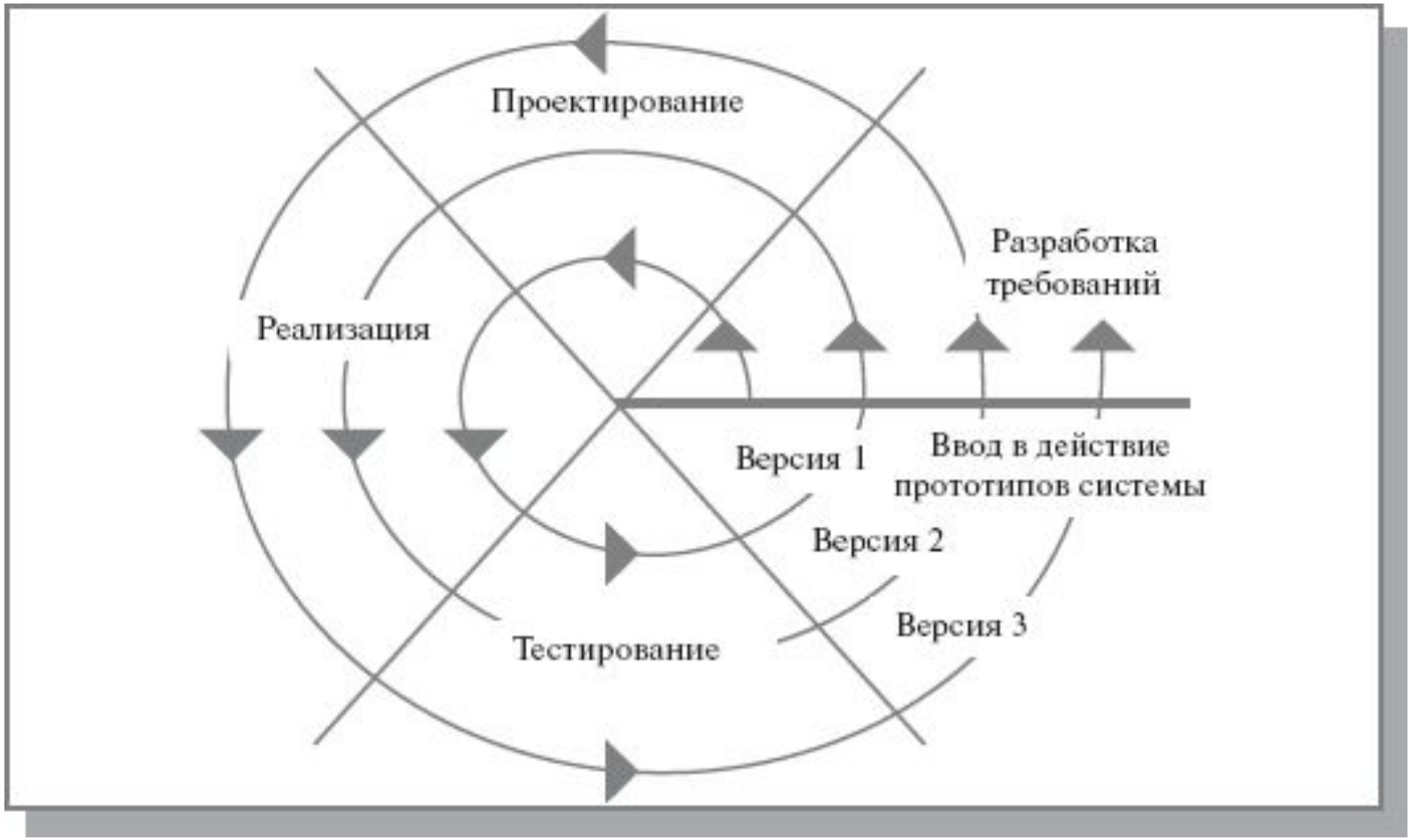

The Waterfall Model
Requirements Specification
Analysis
Design
Implementation (coding)
Testing

Спиральная модель

Спиральная модель была разработана в середине 1980-х годов Барри Боэмом. При использовании этой модели ПО создается в несколько итераций (витков спирали).

Каждая итерация соответствует созданию фрагмента или версии ПО, на ней уточняются цели и характеристики проекта, оценивается качество полученных результатов и планируются работы следующей итерации.

- На каждой итерации оцениваются:
- риск превышения сроков и стоимости проекта;
- необходимость выполнения ещё одной итерации;
- степень полноты и точности понимания требований к системе;
- целесообразность прекращения проекта.


Проектирование
Разработка требований
Реализация
Версия 1
Ввод в действие прототипов системы
Версия 2
Версия 3
Тестирование

Методологии программирования

Методология программирования — совокупность методов, применяемых на различных стадиях жизненного цикла программного обеспечения и имеющих общий философский подход.

Классификации:

- **Классификация по ядрам**
- **Классификация по топологической специфике**
- **Классификация по специфике реализации**

Классификация по ядрам

При подходе к методологии, как имеющей *ядро*, соответствующее способу описания алгоритма, и *дополнительные особенности*, можно выделить следующие пять основных ядер методологий:

- **Методология императивного программирования**
- **Методология структурного программирования**
- **Методология ООП**
- **Методология функционального программирования**
- **Методология логическое программирование**
- **Методология программирования в ограничениях**

Методология структурного программирования

Структу́рное программи́рование — методология разработки программного обеспечения, в основе которой лежит представление программы в виде иерархической структуры блоков. Предложена в 70-х годах XX века Э. Дейкстрой, разработана и дополнена Н. Виртом.

В соответствии с данной методологией любая программа представляет собой структуру, построенную из трёх типов базовых конструкций:

- **последовательное исполнение — однократное выполнение операций в том порядке, в котором они записаны в тексте программы;**
- **ветвление — однократное выполнение одной из двух или более операций, в зависимости от выполнения некоторого заданного условия;**
- **цикл — многократное исполнение одной и той же операции до тех пор, пока выполняется некоторое заданное условие (условие продолжения цикла).**

В программе базовые конструкции могут быть вложены друг в друга произвольным образом, но никаких других средств управления последовательностью выполнения операций не предусматривается.

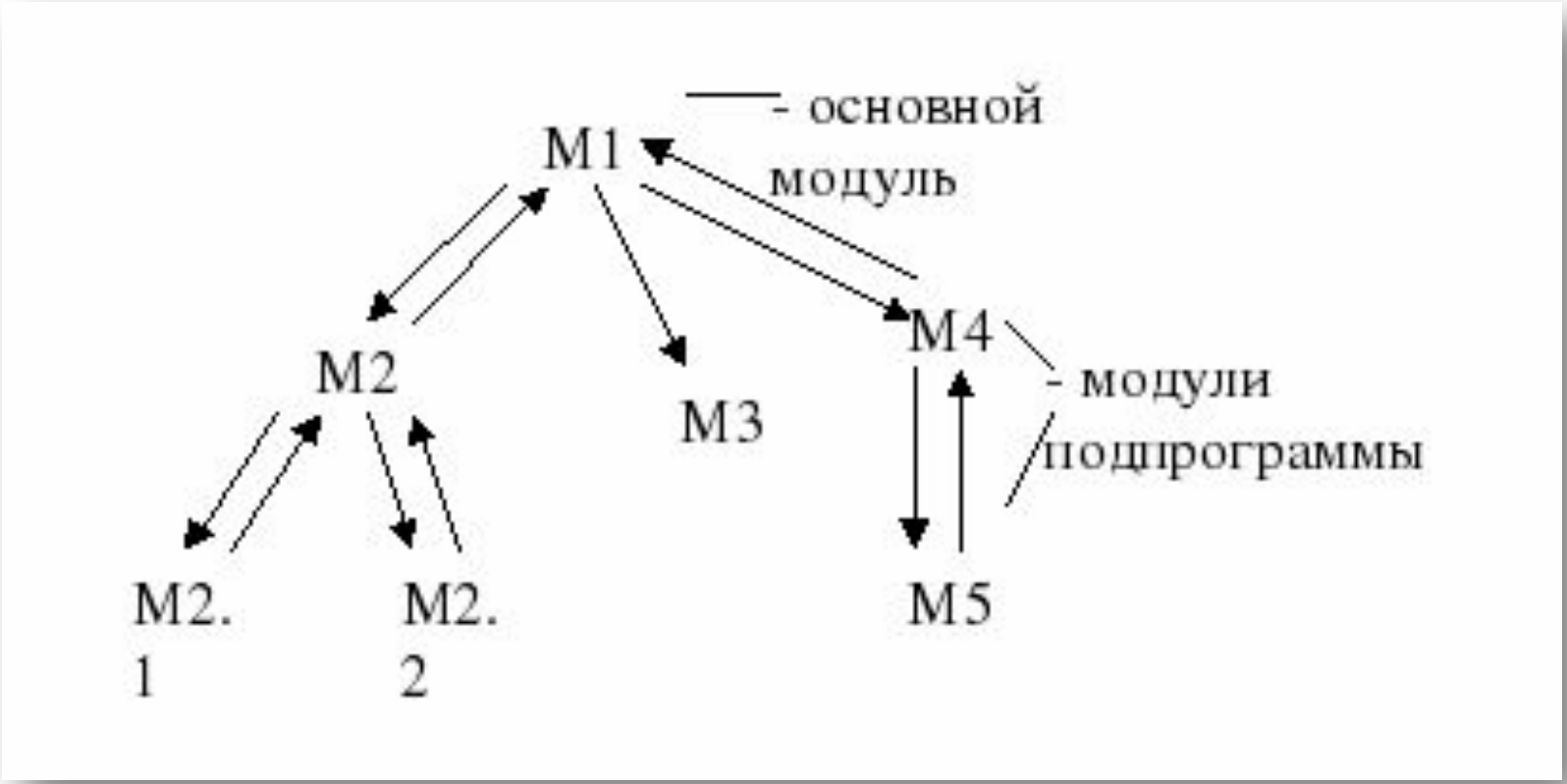
Методология объектно-ориентированного программирования

В центре ООП находится понятие объекта. Объект — это сущность, которой можно посылать сообщения, и которая может на них реагировать, используя свои данные. Объект — это экземпляр класса. Данные объекта скрыты от остальной программы.

Методология логического программирования

Логическое программирование — парадигма программирования, основанная на автоматическом доказательстве теорем, а также раздел дискретной математики, изучающий принципы логического вывода информации на основе заданных фактов и правил вывода. Логическое программирование основано на теории и аппарате математической логики с использованием математических принципов резолюций.

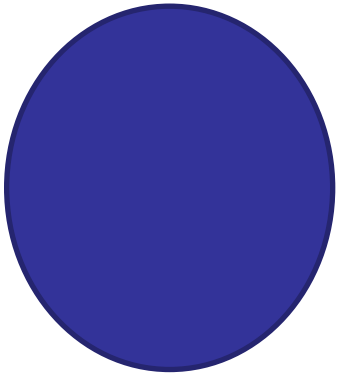
Структурное программирование.



R1 – радиус круга;

X1, Y1 – координаты центра круга;

Color1 – цвет круга.



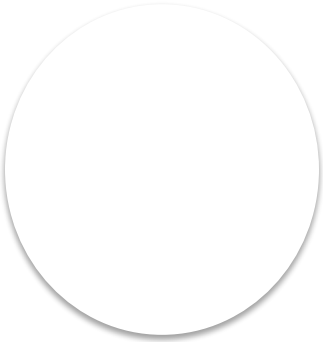
R2 – радиус круга;

X2, Y2 – координаты центра круга;

Color2 – цвет круга.

Draw1

Draw2



R_n – радиус круга;

X_n, Y_n – координаты центра круга;

Color_n – цвет круга.

Draw_n

Абстракция

Абстрагирование — это способ выделить набор значимых характеристик объекта, исключая из рассмотрения незначимые. Соответственно, абстракция — это набор всех таких характеристик.

Инкапсуляция

Инкапсуляция — это свойство системы, позволяющее объединить данные и методы, работающие с ними в классе, и скрыть детали реализации от пользователя.

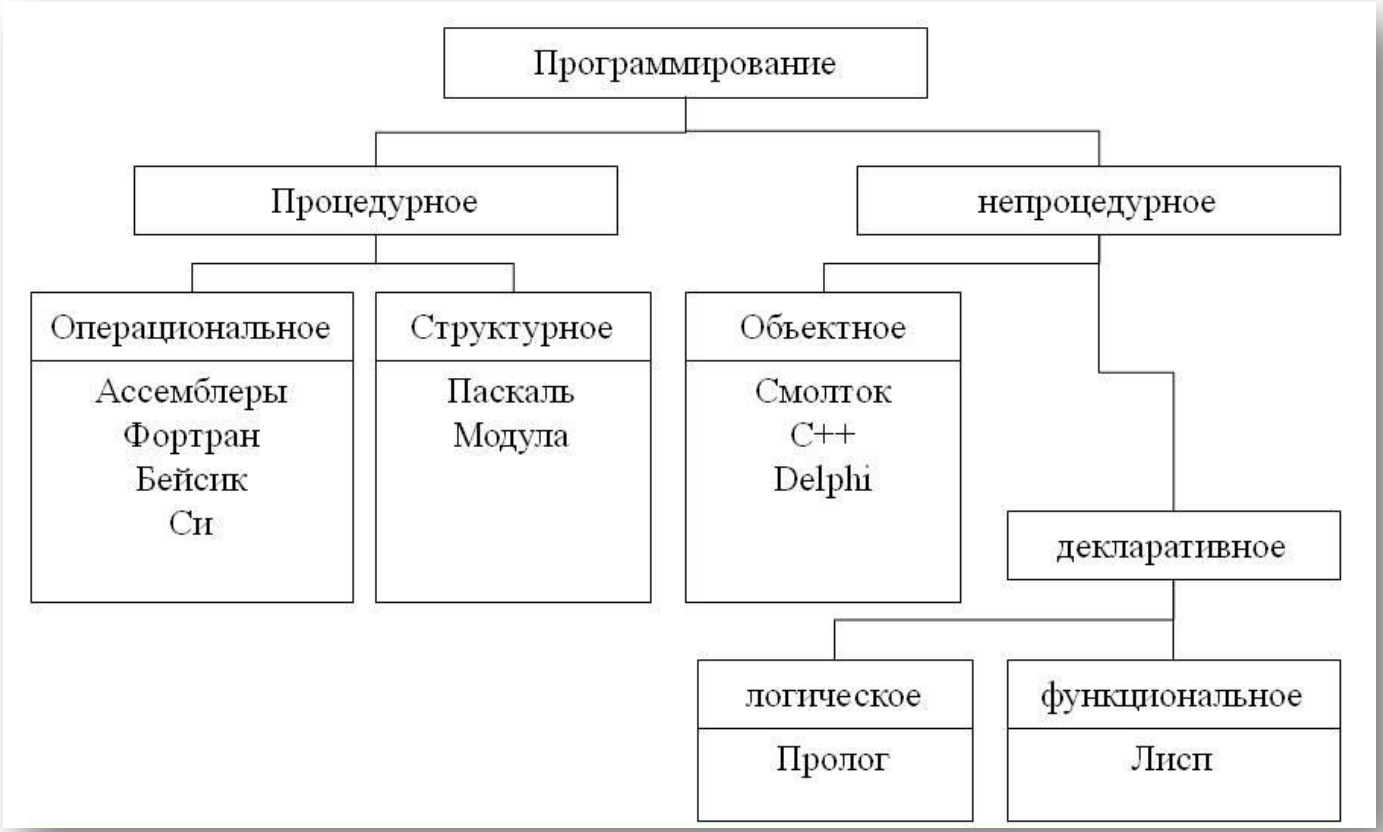
Наследование

Наследование — это свойство системы, позволяющее описать новый класс на основе уже существующего с частично или полностью заимствующейся функциональностью. Класс, от которого производится наследование, называется базовым, родительским или суперклассом. Новый класс — потомком, наследником или производным классом.

Полиморфизм

Полиморфизм — это свойство системы использовать объекты с одинаковым интерфейсом без информации о типе и внутренней структуре объекта.

Классификация языков программирования



Вопросы для самоподготовки:

- Какие модели жизненного цикла существуют? В чем их отличие?
- Какие методологии разработки программного обеспечения существуют?
- На какие группы разделяются языки программирования?
- В чем отличительные особенности объектно-ориентированного программирования?

Выводы по теме 3

В рамках данной темы были получены знания в области методологий разработки программного обеспечения. Обозначены основные модели жизненного цикла программного обеспечения. Выявлены методологии разработки программного обеспечения, их достоинства и недостатки. Классифицированы современные языки программирования.

Вопросы для самостоятельного изучения:

1. ГОСТ 34.601-90
2. Иттерационная модель жизненного цикла
3. Проблемы программирования: локальное и глобальное программирование
4. Функциональное программирование и программирование в ограничениях
5. Пошаговая детализация и нисходящее проектирование

Тема №4
«Структуры данных. Основы проектирования баз данных»

Цели занятия

- Базовые структуры данных – массивы и записи;
- Основные операции над структурами данных;
- Динамические структуры данных. Списки. Стеки. Деревья;
- Информационная система. Понятие базы данных.
- Требования пользователей к базам данных;
- Проектирование баз данных; Цели и этапы проектирования.
- Инфологический аспект. Модель «сущность-связь».
- Даталогический аспект. Модели данных (иерархическая, сетевая, реляционная) их достоинства и недостатки.
- Реляционные базы данных. Понятие отношения. Нормализация.
- Системы управления базами данных;
- Базы данных и компьютерные сети. Сетевые и распределённые базы данных.

Массивы

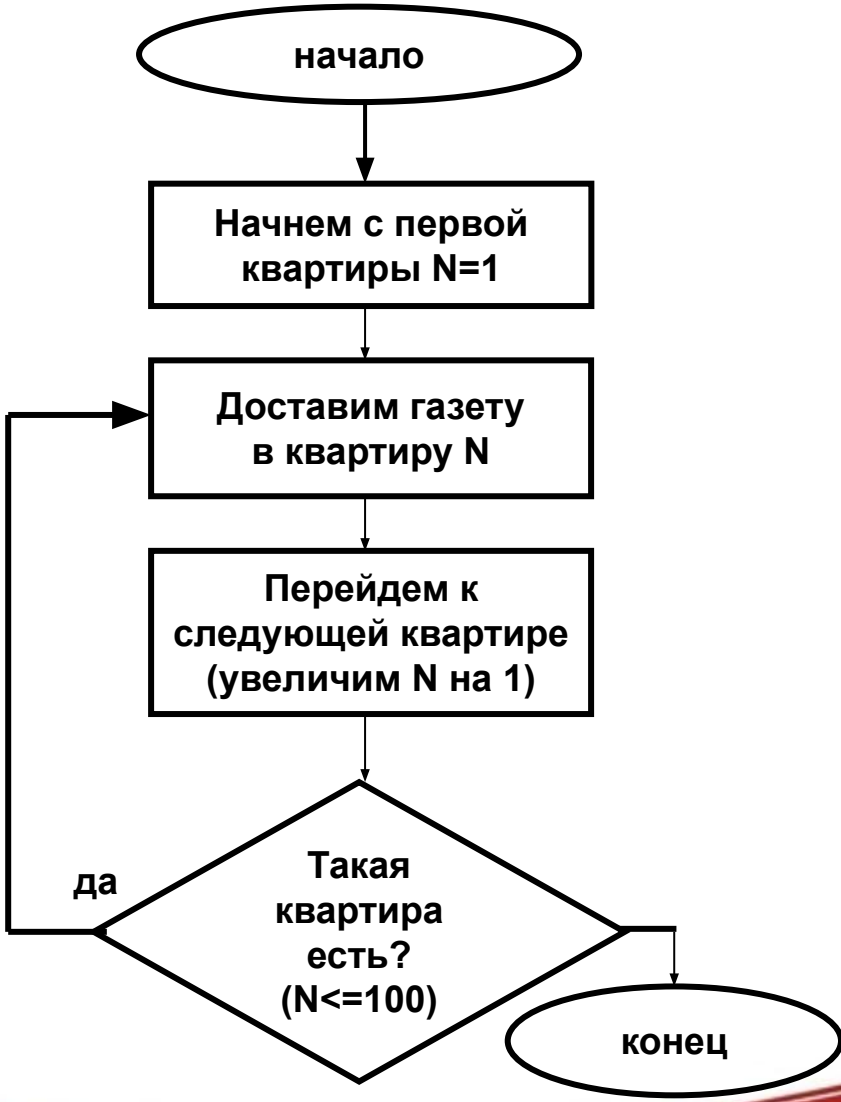
Массив – группа элементов одного типа, объединенных под общим именем.

Индекс – что-то (чаще всего номер), что позволяет отличать элементы массива один от другого и обращаться к ним.

Книга состоит из множества однотипных элементов – страниц, у каждой страницы есть номер (индекс), все страницы объединены под одним названием (название книги)

Пример

Описать алгоритм доставки свежего номера газеты во все квартиры дома, если квартиры нумеруются от 1 до 100.



Записи.

Запись – единица хранения информации в базе данных

Основные операции над структурами данных

Над всеми структурами данных могут выполняться четыре операции:

1. **создание,**
2. **уничтожение,**
3. **выбор (доступ),**
4. **обновление.**

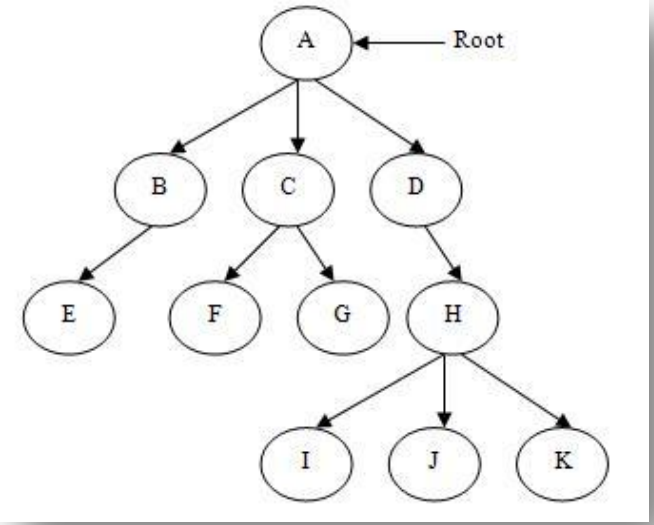
Динамические структуры данных. Списки.

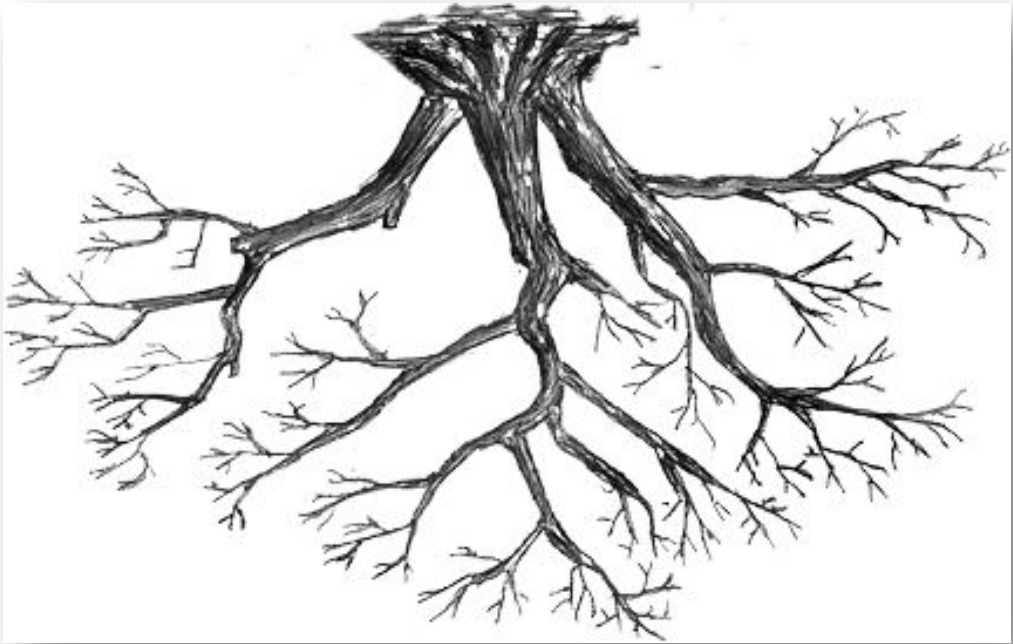
Характеристики

- **Длина списка. Количество элементов в списке.**
- **Списки могут быть типизированными или нетипизированными. Если список типизирован, то тип его элементов задан, и все его элементы должны иметь типы, совместимые с заданным типом элементов списка. Обычно списки, реализованные при помощи массивов, являются типизированными.**
- **Список может быть сортированным или несортированным**
- **В зависимости от реализации может быть возможен произвольный доступ к элементам списка.**

Динамические структуры данных. Деревья.

- **Дерево — одна из наиболее широко распространённых структур данных в информатике, эмулирующая древовидную структуру в виде набора связанных узлов. Является связанным графом, не содержащим циклы.**





Дерево – это структура данных, представляющая собой совокупность элементов и отношений, образующих иерархическую структуру этих элементов .

Каждый элемент дерева называется вершиной (узлом) дерева.

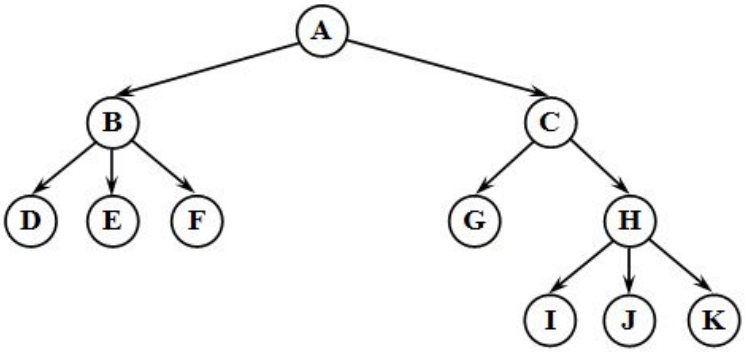
Вершины дерева соединены направленными дугами, которые называют ветвями дерева.
Начальный узел дерева называют корнем дерева, ему соответствует нулевой уровень.

Листьями дерева называют вершины, в которые входит одна ветвь и не выходит ни одной ветви.

Каждое дерево обладает следующими свойствами:

- **существует узел, в который не входит ни одной дуги (корень);**
- **в каждую вершину, кроме корня, входит одна дуга.**

Деревья особенно часто используют на практике при изображении различных иерархий. Например, популярны генеалогические деревья.



Все вершины, в которые входят ветви, исходящие из одной общей вершины, называются потомками, а сама вершина – предком. Для каждого предка может быть выделено несколько потомков.
Уровень потомка на единицу превосходит уровень его предка.
Корень дерева не имеет предка, а листья дерева не имеют потомков.

Высота (глубина) дерева определяется количеством уровней, на которых располагаются его вершины.
Высота пустого дерева рана нулю, высота дерева из одного корня – единице.

На первом уровне дерева может быть только одна вершина – корень дерева, на втором – потомки корня дерева, на третьем – потомки потомков корня дерева и т.д.

Поддерево – часть древообразной структуры данных, которая может быть представлена в виде отдельного дерева.

Степенью вершины в дереве называется количество дуг, которое из нее выходит.

Степень дерева равна максимальной степени вершины, входящей в дерево. При этом листьями в дереве являются вершины, имеющие степень нуль. По величине степени дерева различают два типа деревьев:

- **двоичные – степень дерева не более двух;**
- **сильноветвящиеся – степень дерева произвольная.**

Упорядоченное дерево – это дерево, у которого ветви, исходящие из каждой вершины, упорядочены по определенному критерию.

Деревья являются рекурсивными структурами, так как каждое поддерево также является деревом. Таким образом, дерево можно определить как рекурсивную структуру, в которой каждый элемент является:

- **либо пустой структурой;**
- **либо элементом, с которым связано конечное число поддеревьев.**

Действия с рекурсивными структурами удобнее всего описываются с помощью рекурсивных алгоритмов.

Списочное представление деревьев основано на элементах, соответствующих вершинам дерева.

Каждый элемент имеет поле данных и два поля указателей:

- **указатель на начало списка потомков вершины и**
- **указатель на следующий элемент в списке потомков текущего уровня.**

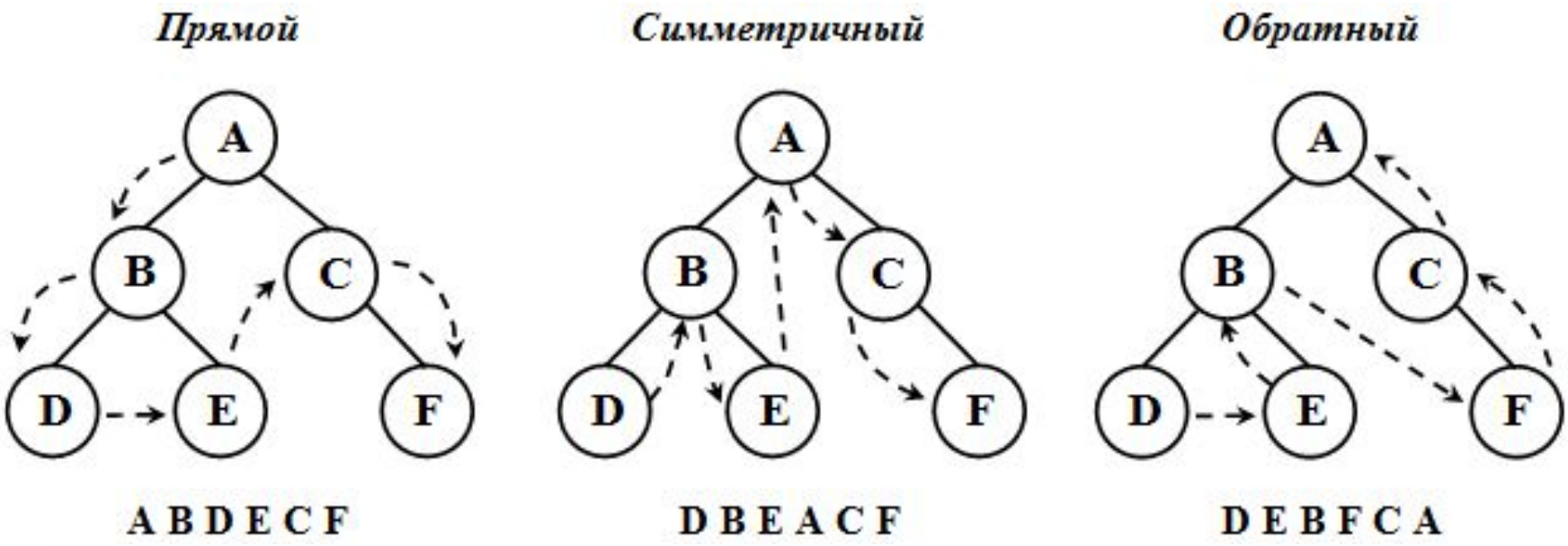
При таком способе представления дерева обязательно следует сохранять указатель на вершину, являющуюся корнем дерева.

Для того, чтобы выполнить определенную операцию над всеми вершинами дерева необходимо все его вершины просмотреть. Такая задача называется обходом дерева.

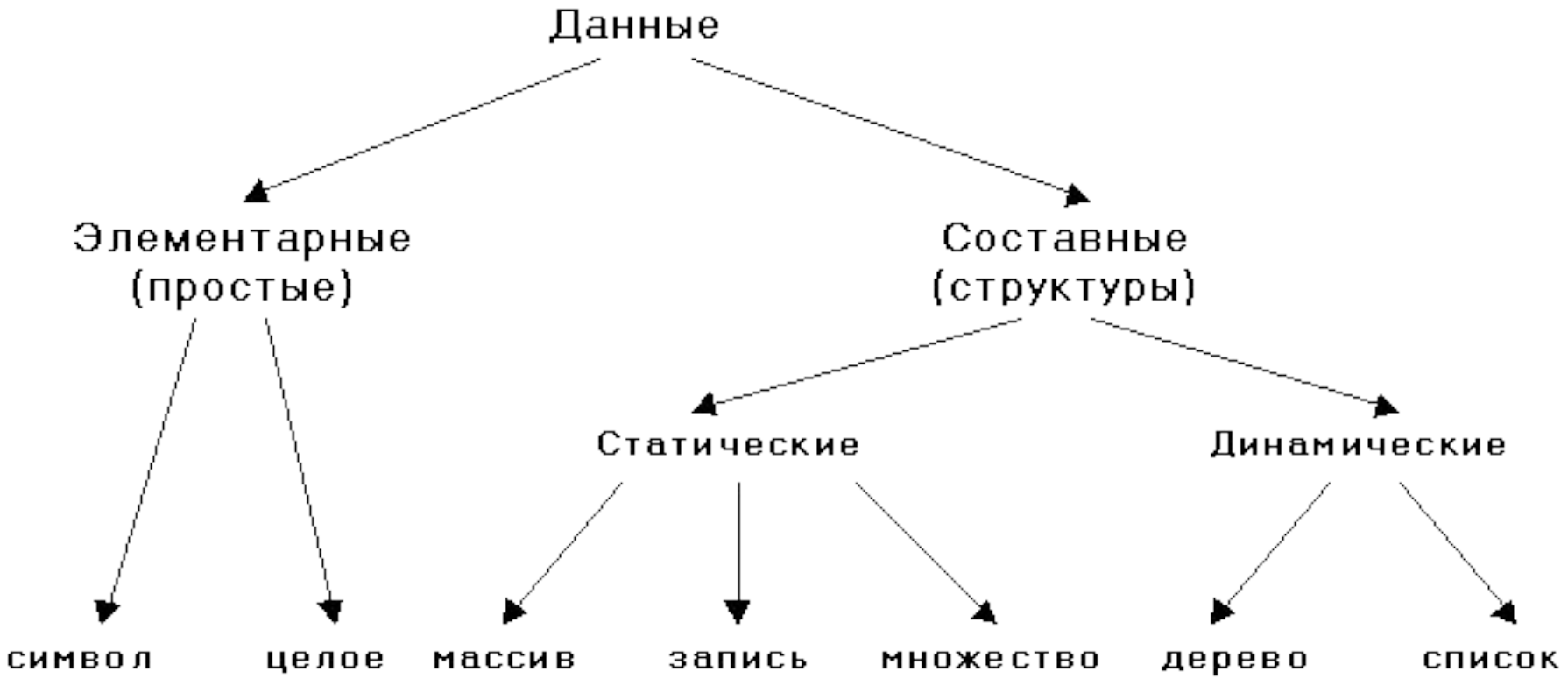
Обход дерева – это упорядоченная последовательность вершин дерева, в которой каждая вершина встречается только один раз.

При обходе все вершины дерева должны посещаться в определенном порядке. Существует несколько способов обхода всех вершин дерева. Выделим три наиболее ча используемых способа обхода дерева:

- прямой;
- симметричный;
- обратный.



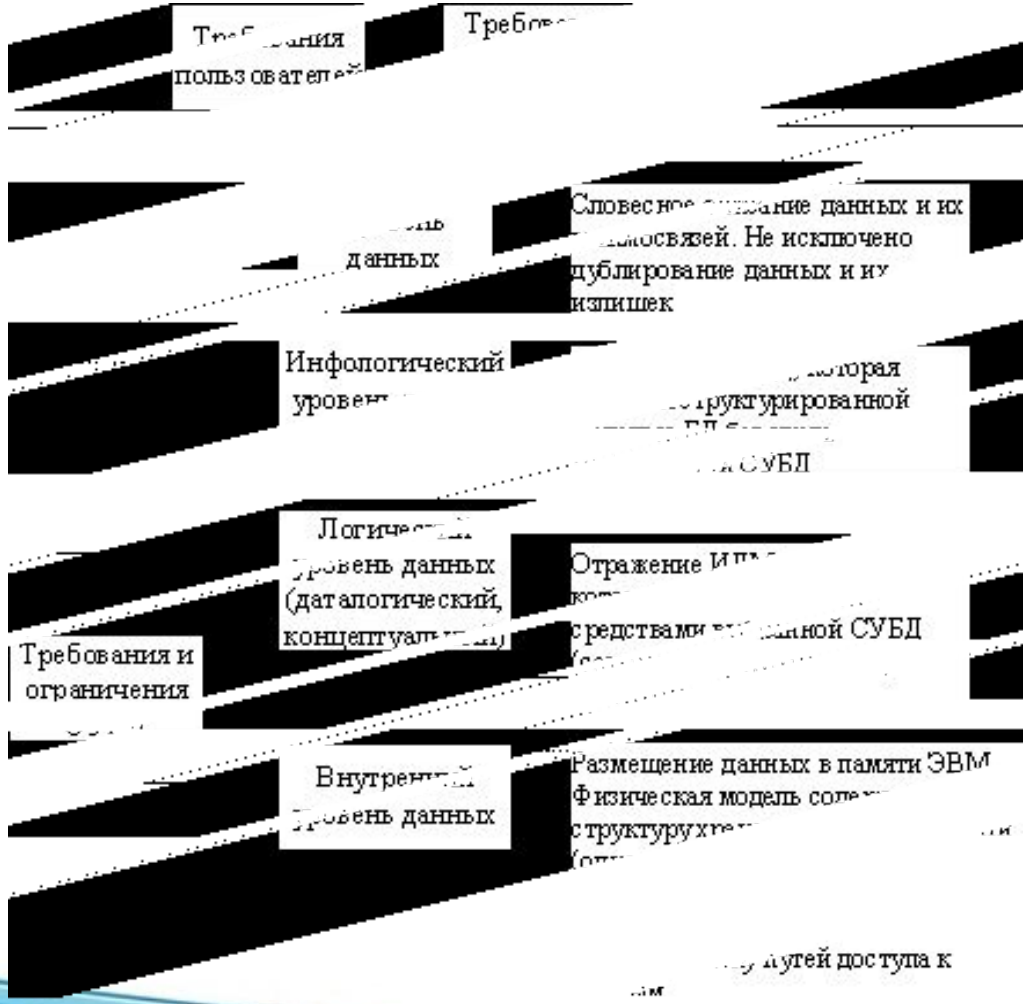
Структурная схема типов данных

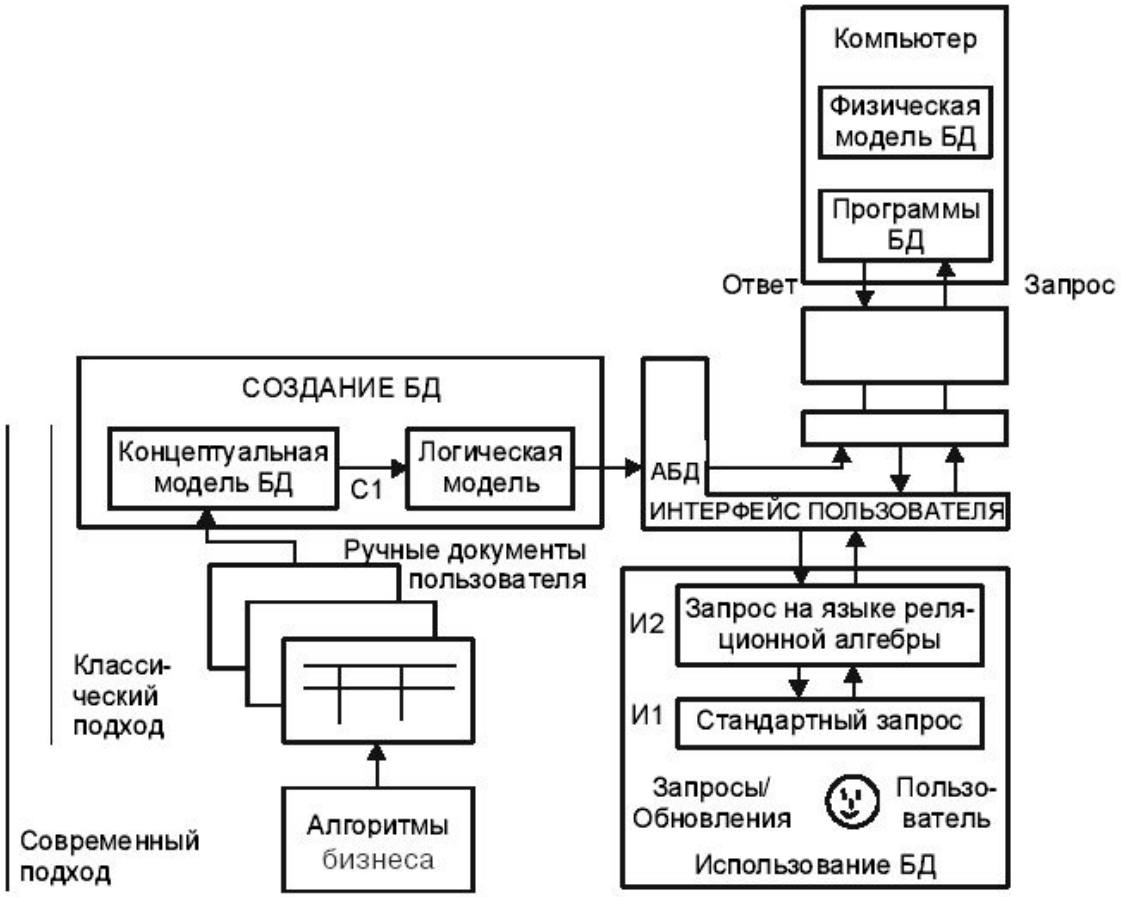


База данных.

База данных – совокупность данных, организованных по определенным правилам, предусматривающим общие принципы описания, хранения и манипулирования данными, независимо от прикладных программ

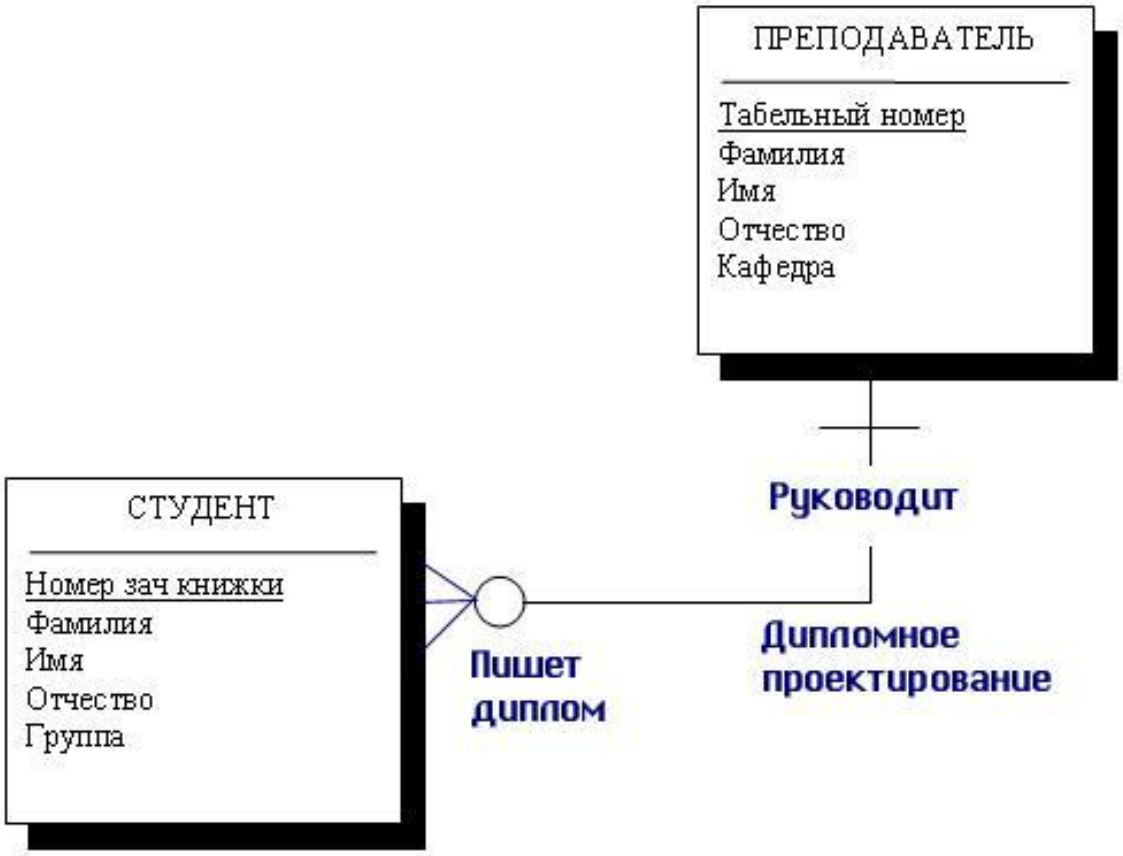
Проектирование баз данных




Компьютер
Физическая модель БД
Программы БД
Ответ
Запрос
СОЗДАНИЕ БД
Концептуальная модель БД
С1
Логическая модель
АБД
ИНТЕРФЕЙС ПОЛЬЗОВАТЕЛЯ
Ручные документы пользователя
Классический подход
Современный подход
Алгоритмы бизнеса
И2
Запрос на языке реляционной алгебры
И1
Стандартный запрос
Запросы/ Обновления
Пользователь
Использование БД


Анализ требований
Постановка задачи (ограничения):
быстродействие; объем памяти; защита данных; надежность и восстановление;
одно- многопользовательский режим
1
Анализ документов
Классический
Подход
Современный
Анализ алгоритмов
2
Выбор МД; Выбор СУБД;
Обеспечение целостности и защиты
Концептуальная БД
Однопользовательский
Режим
Многопользовательский
3
Схема
Подсхема
4
Логическая БД
Централизованная
Разновидность
Распределенная
Фрагментация
Локализация
Фрагментация
Целостность и восстановление
Физическая БД
Реализация
Реализация

Модель «сущность-связь»



Типы отношений в модели «сущность-связь»

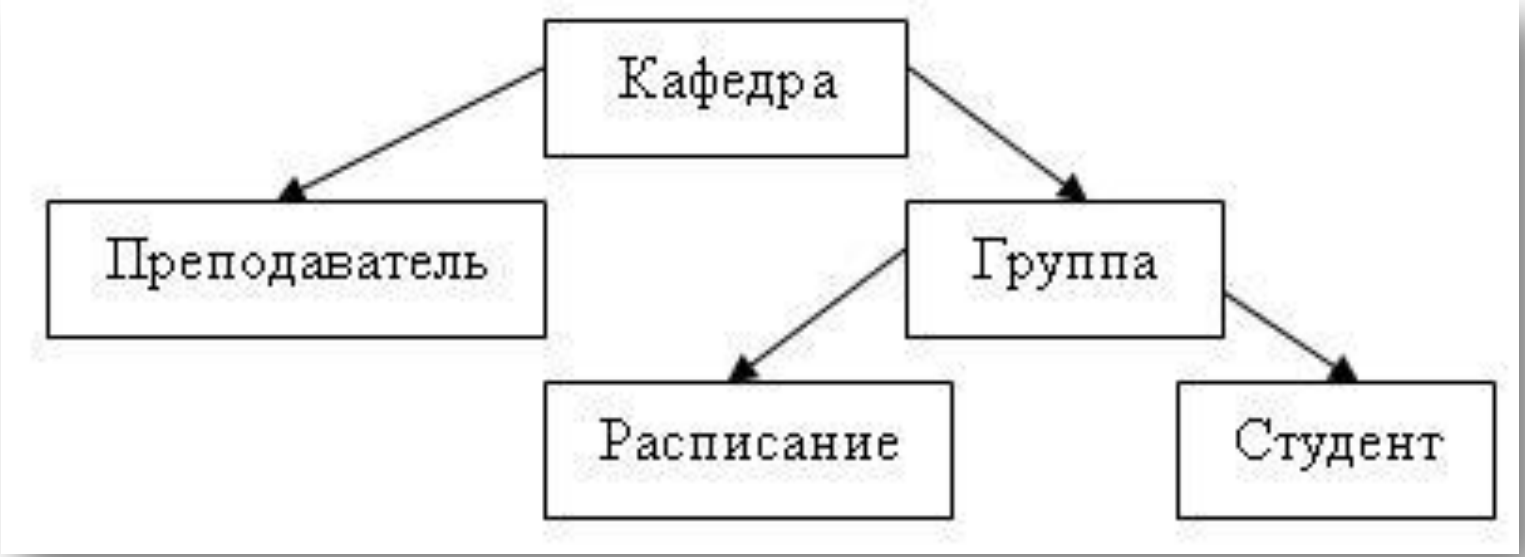
Модели данных.

Модель данных - совокупность структур данных и операций их обработки.

По способу установления связей между данными СУБД (системы управления базами данных) основывается на использовании трёх основных видов модели: иерархической, сетевой или реляционной; на комбинации этих моделей или на некотором их подмножестве.
Каждая из указанных моделей обладает характеристиками, делающими ее наиболее удобной для конкретных приложений.

Иерархическая модель

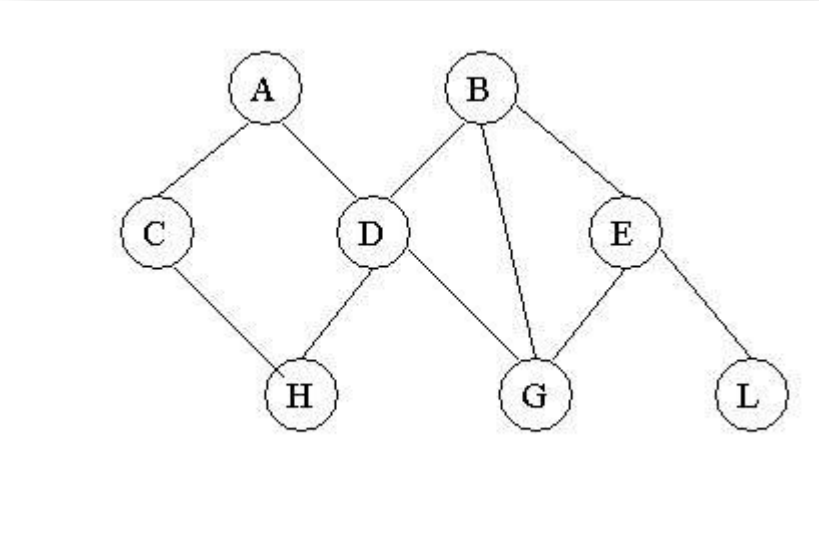
Иерархическая модель данных — представление базы данных в виде древовидной (иерархической) структуры, состоящей из объектов (данных) различных уровней.

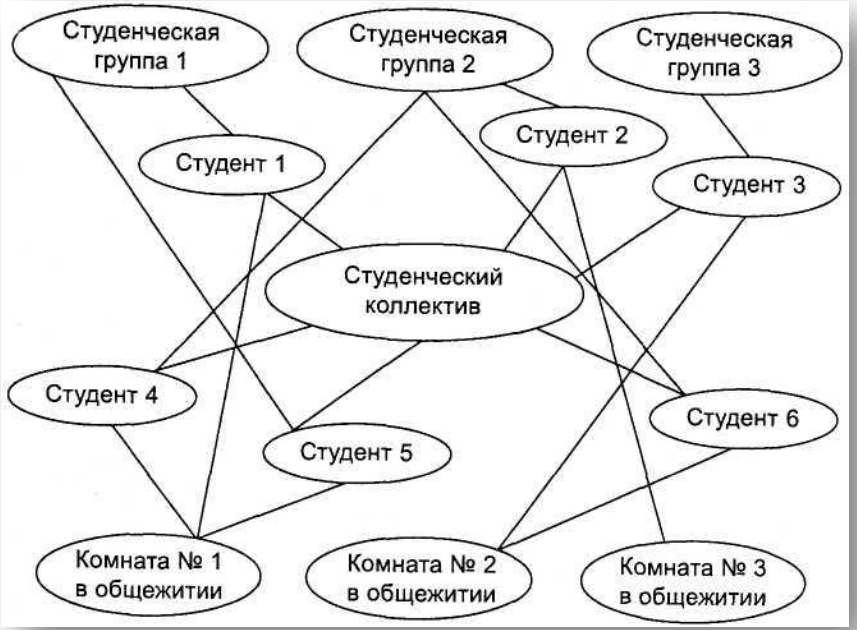


Сетевая модель.

Разница между иерархической моделью данных и сетевой состоит в том, что в иерархических структурах запись-потомок должна иметь в точности одного предка, а в сетевой структуре данных у потомка может иметься любое число предков.

Сетевая БД состоит из набора экземпляров определенного типа записи и набора экземпляров определенного типа связей между этими записями.





Реляционная модель

Реляционная модель данных – логическая модель данных, основанная на отношении одних записей к другим. Представляет собой таблицы и связи между ними. Впервые была предложена британским учёным сотрудником компании IBM Эдгаром Франком Коддом (E. F. Codd) в 1970 году. В настоящее время эта модель является фактическим стандартом, на который ориентируются практически все современные коммерческие СУБД.

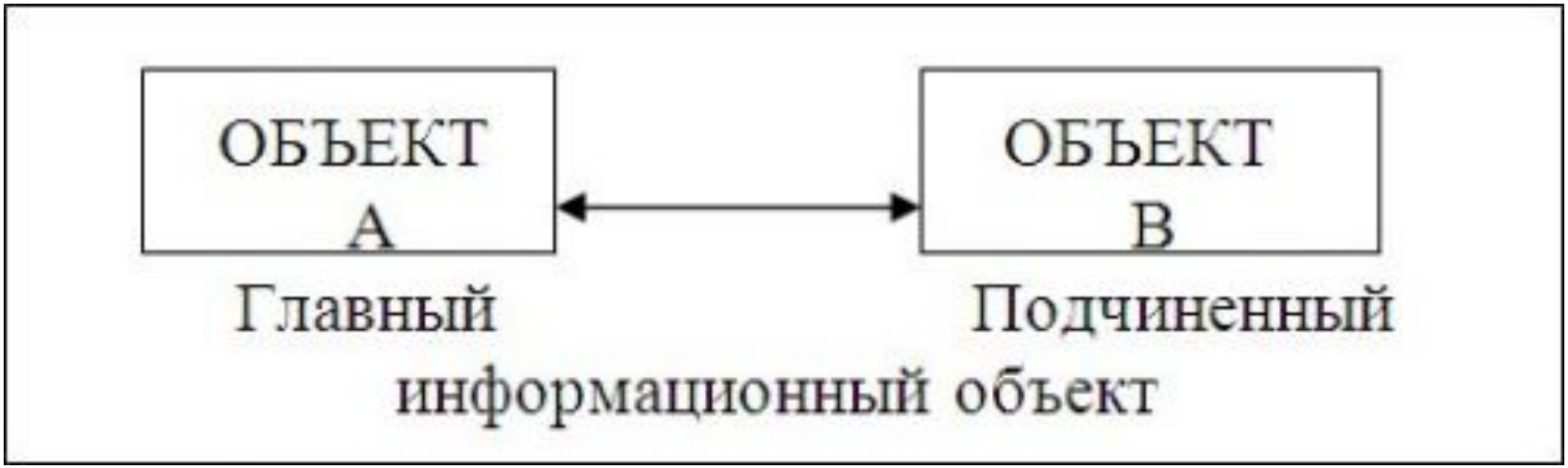
Реляционная модель

Отношения между таблицами. Связь 1:1

Одной записи Таблицы А соответствует одна запись Таблицы В

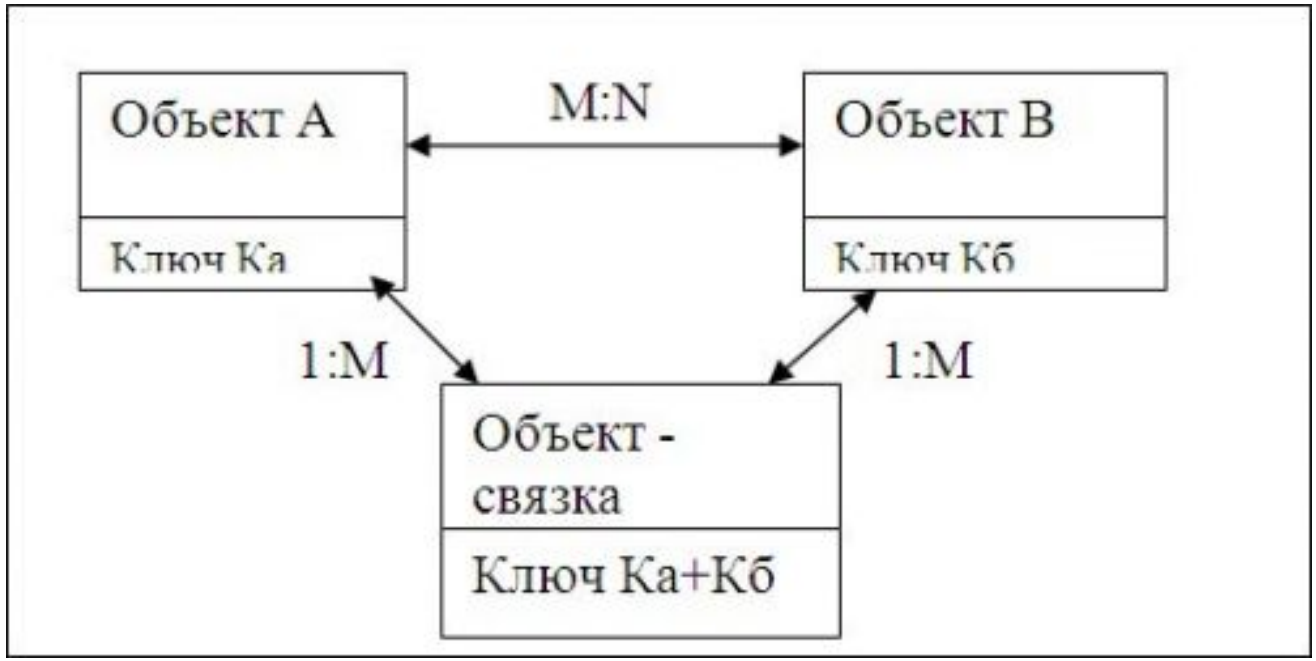
Отношения между таблицами. Связь 1:M и M:1

Одной записи Таблицы A соответствует несколько записей Таблицы B



Отношения между таблицами. Связь M:M

Многим записям Таблицы А соответствует много записей Таблицы В. В явном виде не существует. Реализуется через третью таблицу и две связи 1:M



Нормализация таблиц базы данных. Первая нормальная форма.

Таблица «Товары»

Наименование	Стоимость (руб.)	Склад
Холодильник «Лютый» Микроволвка «Вулкан»	5000	Основной
Фен «Торнадо»	1500	Основной Производственный
Телевизор ч/б «Рассвет»	1 100	Основной

Таблица «Товары»

Клиент	Стоимость (руб.)	Склад
Холодильник «Лютый»	5000	Основной
Микроволвка «Вулкан»	5000	Основной
Фен «Торнадо»	1500	Основной
Фен «Торнадо»	1500	Производственный
Телевизор ч/б «Рассвет»	1100	Основной

Нормализация таблиц базы данных. Вторая нормальная форма.

Таблица «Товары»

ID	Наименование	Стоимость (руб.)	Склад
1	Холодильник «Лютый»	5000	2
2	Микроволвка «Вулкан»	5000	2
3	Фен «Торнадо»	1500	2
4	Фен «Торнадо»	1500	1
5	Телевизор ч/б «Рассвет»	1100	2

Таблица «Склады»

ID	Наименование	Адрес
1	Производственный	Ул. Ленина 15
2	Основной	Ул. Кирова 28

Нормализация таблиц базы данных. Вторая нормальная форма.

Таблица «Товары»

ID	Наименование	Стоимость (руб.)	Склад
1	Холодильник «Лютый»	5000	2
2	Микроволвка «Вулкан»	5000	2
3	Фен «Торнадо»	1500	2
4	Фен «Торнадо»	1500	1
5	Телевизор ч/б «Рассвет»	1100	2

Таблица «Склады»

ID	Наименование	Город	Индекс
1	Производственный	Новосибирск	630423
2	Основной	Томск	520543

Вопросы для самоподготовки:

- Какие базовые структуры хранения данных существуют? В чем отличие между массивами и структурами? Где они применяются?
- Назовите основные операции над структурами данных.
- Какие бывают динамические структуры данных?
- Что такое база данных? Какие требования применяются к БД?
- Какие основные этапы проектирования БД существуют?
- Какие модели данных существуют?
- Что такое реляционные базы данных? Какие основные элементы реляционной базы данных существуют?
- В чем состоит суть нормализации БД?
- В каких случаях используются сетевые и распределенные БД?

Выводы по теме 4

В рамках данной темы были получены знания в области представления данных в ЭВМ: массивы, структуры, стеки, деревья. Обозначено понятие информационной системы и базы данных. Сформулированы основные этапы проектирования БД. Рассмотрен процесс нормализации БД.

Вопросы для самостоятельного изучения:

1. Динамические структуры данных – стеки
2. Понятие информационной системы. Классификация информационных систем
3. Требования пользователей к базам данных
4. Инфологическая модель
5. СУБД: виды, особенности
6. Сетевые и распределенные базы данных

СПАСИБО ЗА ВНИМАНИЕ!