Лекция № 14

Алгоритмическое обеспечение информатики

ПОНЯТИЕ АЛГОРИТМА

Слово «алгоритм» происходит от имени великого среднеазиатского ученого 8–9 вв. Аль-Хорезми.

Из математических работ Аль-Хорезми до нас дошли только две – алгебраическая и арифметическая. Вторая книга долгое время считалась потерянной, но в 1857 в библиотеке Кембриджского университета был найден ее перевод на латинский язык. В ней описаны четыре правила арифметических действий, практически те же, что используются и сейчас. Первые строки этой книги были переведены так: «Сказал Алгоритми. Воздадим должную хвалу Богу, нашему вождю и защитнику». Так имя Аль-Хорезми перешло в «Алгоритми», откуда и появилось слово «алгоритм».

Алгоритм – это система однозначных инструкций (указаний), которая определяет последовательность действий над выбранными объектами с целью получения результата за конечное число шагов.

Алгоритм – это заданное на некотором языке конечное предписание, задающее конечную последовательность выполнимых и точно определенных элементарных операций для решения задачи.

Алгоритм (по Колмогорову) – это система вычислений, выполняемых по строго определенным правилам, которая после какого-либо числа шагов заведомо приводит к решению поставленной задачи.

Алгоритм (по Маркову) – это точное предписание, определяющее вычислительный процесс, идущий от варьируемых исходных данных к искомому результату.

Алгоритмы:

-численные;

-логические.

Численные алгоритмы – это алгоритмы, в соответствии с которыми решение задач сводится к арифметическим действиям.

Логические алгоритмы – это алгоритмы, в соответствии с которыми решение задач сводится к логическим действиям.

Требования к алгоритмам:

-Содержать конечное количество элементарно выполнимых предписаний, т.е. удовлетворять требованию конечности записи;

-Выполнять конечное количество шагов при решении задачи, т.е. удовлетворять требованию конечности действий;

-Быть единым для всех допустимых исходных данных, т.е. удовлетворять требованию универсальности;

-Приводить к правильному по отношению к поставленной задаче решению, т.е. удовлетворять требованию правильности.

СВОЙСТВА АЛГОРИТМА

- дискретность;
- определенность;
- результативность;
- массовость.

Дискретность – последовательное выполнение простых или ранее определённых (подпрограммы) шагов. Преобразование исходных данных в результат осуществляется дискретно во времени.

Определенность состоит в совпадении получаемых результатов независимо от пользователя и применяемых технических средств (однозначность толкования инструкций).

Результативность означает возможность получения результата после выполнения конечного количества операций.

Массовость заключается в возможности применения алгоритма к целому классу однотипных задач, различающихся конкретными значениями исходных данных (разработка в общем виде).

Для задания алгоритма необходимо описать следующие его элементы:

- набор объектов, составляющих совокупность возможных исходных данных, промежуточных и конечных результатов;
- правило начала;
- правило непосредственной переработки информации (описание последовательности действий);
- правило окончания;
- правило извлечения результатов.

СПОСОБЫ ОПИСАНИЯ АЛГОРИТМОВ

К основным способам описания алгоритмов можно отнести следующие:

- словесно-формульный (на естественном языке);
- с помощью граф-схем (граф - совокупность точек и линий, в которой каждая линия соединяет две точки. Точки называются вершинами, линии - рёбрами);
- псевдокод;
- с помощью диаграмм Нэсси-Шнейдермана;
- программный.

Словесно-формульный способ

При ***словесно-формульном*** способе алгоритм записывается в виде текста с формулами по пунктам, определяющим последовательность действий.

Пусть, например, необходимо найти значение следующего выражения: **y=2a-(x+6).**

Словесно-формульным способом алгоритм решения этой задачи может быть записан в следующем виде:

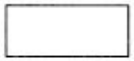
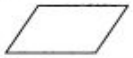
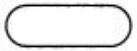
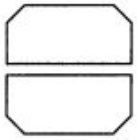
1.Ввести значения *a* и *x.*
2.Сложить x и 6.
3.Умножить *a* на 2.
4.Вычесть из 2a сумму (x+6).
5.Вывести y как результат вычисления выражения.

ГРАФИЧЕСКИЙ СПОСОБ

При графическом представлении алгоритм изображается в виде последовательности связанных между собой функциональных блоков, каждый из которых соответствует выполнению одного или нескольких действий.

Такое графическое представление называется схемой алгоритма или **блок-схемой**. В блок-схеме каждому типу действий (вводу исходных данных, вычислению значений выражений, проверке условий, управлению повторением действий, окончанию обработки и т.п.) соответствует геометрическая фигура, представленная в виде **блочного символа**. Блочные символы соединяются **линиями переходов**, определяющими очередность выполнения действий.

Основные элементы схемы алгоритма (по ГОСТ 19.701-90)

№	Символическое обозначение	Наименование	Описание
1		Процесс	Блок функции обработки данных любого вида: выполнение определенной операции или группы операций, приводящее к изменению значения, формы, размещения информации или к определению направления дальнейшего движения
2		Решение	Блок решения или функции переключательного типа. Внутри блока записывается условие. Блок имеет один вход и два альтернативных выхода: «да» — условие выполнено, «нет» — условие не выполнено
3		Данные	Блок отображает данные, носитель данных не определен
4		Терминатор	Блок отображает выход во внешнюю среду и вход из внешней среды (начало или конец схемы)
5		Соединитель	Символ отображает выход в часть схемы и вход из другой части этой схемы и используется для обрыва линии и продолжения ее в другом месте. Соответствующие символы-соединители должны быть одинаковыми
6		Предопределенный процесс	Блок для отображения подпрограммы или модуля
7		Подготовка	Блок отражает модификацию команды или группы команд с целью воздействия на некоторую последующую функцию (установка переключателя, модификация индексного регистра или инициализация программы)
8		Комментарий	Символическое обозначение используется для добавления комментариев. Пунктирные линии в символе комментария связаны с соответствующим блоком. Текст комментариев должен быть помещен около скобки
9		Линия	Символ отображает поток данных или управление
10		Граница цикла	Символ, состоящий из двух частей, отображает начало и конец цикла. Обе части символа имеют один и тот же идентификатор. Условия для инициализации, приращения, завершения и т. д. помещаются внутри символа в начале или в конце в зависимости от расположения операции, проверяющей условие

ПСЕВДОКОД

Псевдокод представляет собой систему обозначений и правил, предназначенную для единообразной записи алгоритмов.

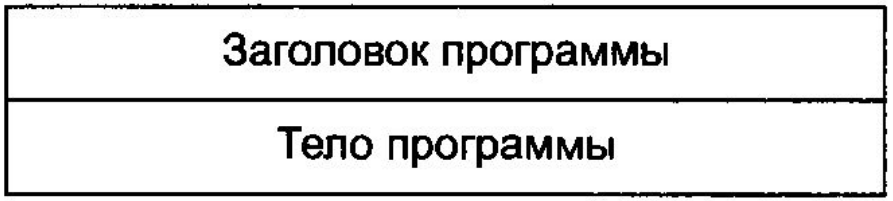
Для записи предложений используются: русский язык, формальные языки предметных областей, в которых решается исходная задача; ключевые слова псевдокодов.

Для реализации псевдокодов, в них резервируются следующие ключевые слова:

АЛГОРИТМ,
НАЧАЛО_алгоритма,
КОНЕЦ_алгоритма,
ПОДАЛГОРИТМ,
НАЧАЛО_вспомогательного алгоритма,
КОНЕЦ_вспомогательного алгоритма,
НАЧАЛО_описания переменных,
КОНЕЦ_описания переменных,
НАЧАЛО_если <условие>,
ТО,
ИНАЧЕ,
КОНЕЦ_если,
НАЧАЛО_цикла с предусловием <условие входа в цикл>,
КОНЕЦ_цикла с предусловием,
НАЧАЛО_цикла с постусловием,
КОНЕЦ_цикла с постусловием <условие выхода из цикла>,
НАЧАЛО_цикла с параметром <параметр, его диапазон и шаг>,
КОНЕЦ_цикла с параметром <параметр цикла>.

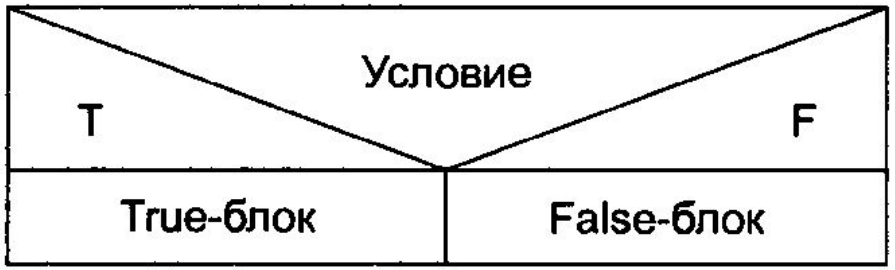
ДИАГРАММА НЭССИ-ШНЕЙДЕРМАНА

Программа

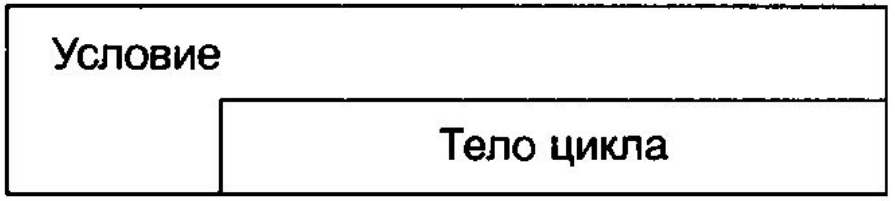


Простая конструкция

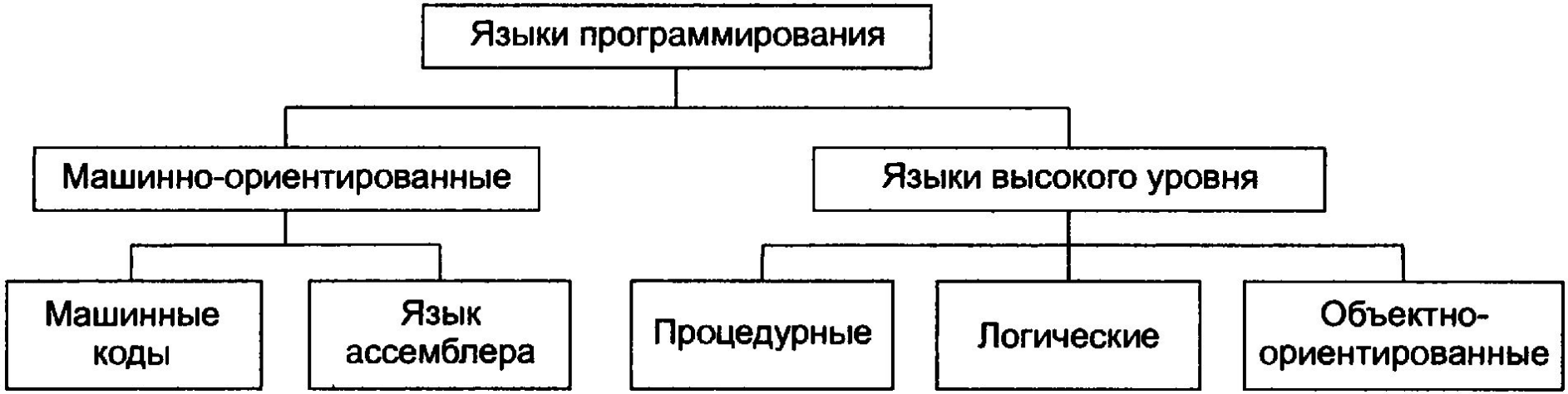
Условный оператор



Циклическая конструкция



ПРОГРАММНЫЙ СПОСОБ



- *машинно-ориентированные языки* — машинные языки и языки ассемблера;
- *машинно-независимые языки* — языки высокого уровня.

Машинно-ориентированные языки — это языки низкого уровня, требующие указания мелких деталей процесса обработки данных. Языки же высокого уровня имитируют естественные языки, используя некоторые слова естественного языка и общепринятые математические символы. Эти языки более удобны для человека.

Языки высокого уровня делятся на процедурные, логические и объектно-ориентированные.

- *Процедурные*, или *алгоритмические*, языки (Basic, Pascal, C и др.) предназначены для однозначного описания алгоритмов в виде некоторой последовательности операторов языка.
- *Логические* языки (Prolog, Lisp и др.) ориентированы не на разработку алгоритма решения задачи, а на систематическое и формализованное описание задачи, из которого должно следовать решение.
- *Объектно-ориентированные* языки (Object Pascal, C++, Java, C# и др.), в основе которых лежит понятие объекта, сочетают в себе данные и действия над ними. Программа на объектно-ориентированном языке, решая некоторую задачу, по сути, описывает часть мира, относящуюся к этой задаче. Описание действительности в форме системы взаимодействующих объектов естественнее, чем в форме взаимодействующих процедур.

БАЗОВЫЕ АЛГОРИТМИЧЕСКИЕ СТРУКТУРЫ

Логическая структура любого алгоритма может быть представлена комбинацией трех базовых структур: **следование, ветвление, цикл.**

1. Базовая структура "следование". Образуется последовательностью действий, следующих одно за другим:

Естественный алгоритмический язык

действие 1
действие 2
. . . . . .
действие n

Язык блок-схем

2. Базовая структура "ветвление".

Обеспечивает в зависимости от результата проверки условия (**да** или **нет**) выбор одного из альтернативных путей работы алгоритма. Каждый из путей ведет к **общему выходу**, так что работа алгоритма будет продолжаться независимо от того, какой путь будет выбран.

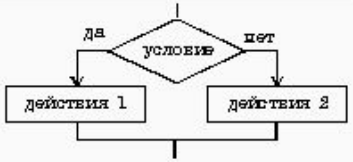
Структура **ветвление** существует в четырех основных вариантах:
если—то;
если—то—иначе;
выбор;
выбор—иначе.

1. если—то

```
условие
действия
```

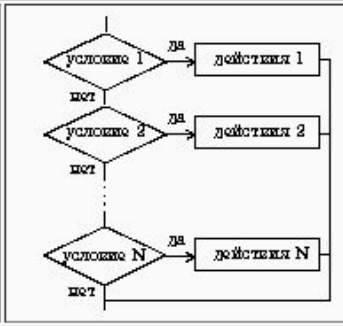
2. если—то—иначе

```
условие
действия 1
   действия 2
```



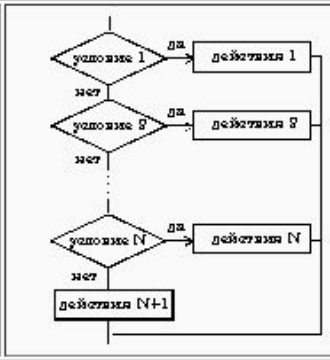
3. выбор

```
условие 1: действия 1
условие 2: действия 2
. . . . . .
условие N: действия N
```

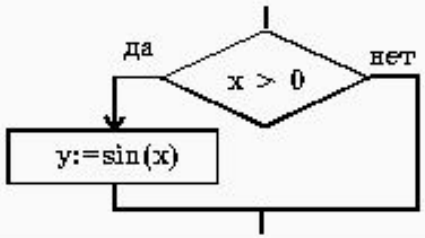


4. выбор—иначе

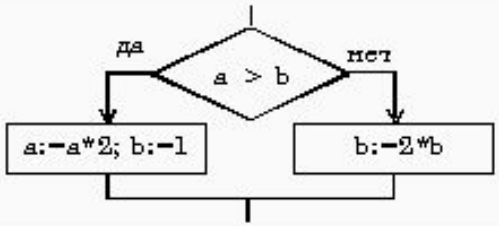
```
условие 1: действия 1
условие 2: действия 2
. . . . . .
условие N: действия N
  действия N+1
```



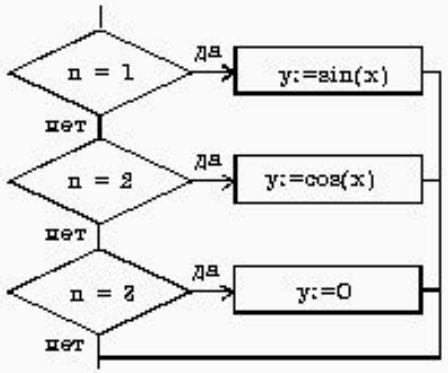
```
x > 0
y := sin(x)
```



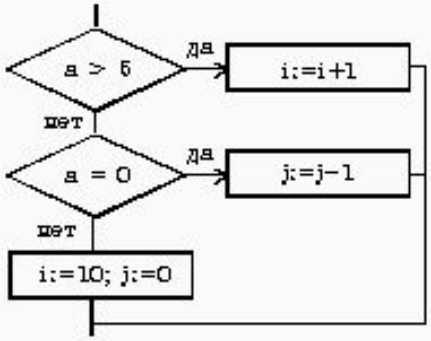

```
a > b
a := 2*a; b := 1
   b := 2*b
```




```
n = 1: y := sin(x)
n = 2: y := cos(x)
n = 3: y := 0
```




```
a > 5: i := i+1
a = 0: j := j+1
  i := 10; j:=0
```



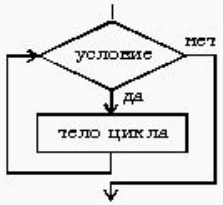
3. Базовая структура "цикл".

Обеспечивает **многократное выполнение некоторой совокупности действий,** которая называется **телом цикла**. Основные разновидности циклов представлены в таблице:

Цикл типа **пока**.

Предписывает выполнять тело цикла до тех пор, пока выполняется условие, записанное после слова пока.

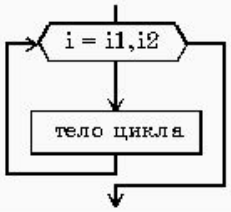
```
      условие
тело цикла
(последовательность действий)
```



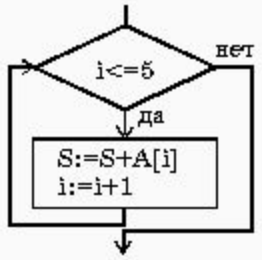
Цикл типа **для**.

Предписывает выполнять тело цикла для всех значений некоторой переменной (параметра цикла) в заданном диапазоне.

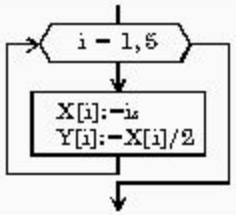
```
     i     i1     i2
тело цикла
(последовательность действий)
```




```
       i <= 5
S := S+A[i]
i := i+1
```




```
      i     1     5
X[i] := i*i*i
Y[i] := X[i]/2
```



ПРЕДСТАВЛЕНИЕ И ОБРАБОТКА ДАННЫХ

Алгоритм, реализующий решений некоторой конкретной задачи, всегда работает с данными.

Данные: входные (исходные), выходные (результирующие), промежуточные.

Переменные – это данные, значения которых могут изменяться в процессе выполнения алгоритма.

Константы – это данные, значения которых не меняются в процессе выполнения алгоритма.

$$S = \pi \cdot R^2$$

Каждая переменная или константа имеет свое уникальное имя – идентификатор, представляющий собой последовательность букв и цифр, начинающуюся с буквы.

Тип данных – это характеристика данных, определяющая множество значений и операций, которые могут быть применены к этим данным, а также правили их выполнения.

Простой (базовый) тип данных – это тип используемой в алгоритме конкретной переменной или константы.

Структурированный тип данных – это набор однотипных или разнотипных данных, с которыми алгоритм может работать как с одной именованной переменной.

ПРОСТЫЕ ТИПЫ ДАННЫХ

Целочисленные типы - обозначают множества целых чисел в различных диапазонах. Имеется пять целочисленных типов, различающихся диапазоном допустимых значений и размером занимаемой оперативной памяти. Целочисленные типы обозначаются идентификаторами: **Byte**, **ShortInt**, **Word**, **Integer**, **LongInt**; их характеристики приведены в следующей таблице.

Тип	Диапазон	Размер в байтах
Byte	0 ... 255	1
ShortInt	-128 ... 127	1
Word	0 ... 65535	2
Integer	-32768 ... 32767	2
LongInt	-2147483648 ... 2147483647	4

Значения целых типов записываются в программе привычным способом:

123 4 -3 +345 -699

Наличие десятичной точки в записи целого числа недопустимо. Будет ошибкой записать целое число следующим образом:

123.0

Кроме привычной десятичной формы записи допускается запись целых чисел в шестнадцатеричном формате, используя префикс $, например:

$01AF $FF $1A $F0A1B

Регистр букв A,B, ..., F значения не имеет.

Допустимые операции:

- присваивание;
- все арифметические: +, - ,*, /, div, mod (при обычном делении [/] результат вещественный!);
- сравнение <, >, >=, <=, <>, =.

Логический тип (Boolean) - состоит всего из двух значений: **False** (ложно) и **True** (истинно). Слова **False** и **True** определены в языке и являются, по сути, логическими константами. Регистр букв в их написании несущественен: **FALSE = false**. Значения этого типа являются результатом вычислений условных и логических выражений и участвуют во всевозможных условных операторах языка.

Допустимые операции:
- присваивание;
- сравнение: <, >, >=, <=, <>, =;
- логические операции: NOT, OR, AND, XOR

Символьный тип (Char) - это тип данных, состоящих из одного символа (знака, буквы, кода). Значением типа Char может быть любой символ из набора ASCII. Если символ имеет графическое представление, то в программе он записывается заключенным в одиночные кавычки (апострофы), например:

'ж' 's' '.' '*' ' '-(пробел)

Для представления самого апострофа его изображение удваивается: **''''**.
Если же символ не имеет графического представления, например, символ табуляции или символ возрата каретки, то можно воспользоваться эквивалентной формой записи символьного значения, состоящего из префикса # и ASCII-кода символа:

#9 #32 #13

Допустимые операции:
- присваивание;
- сравнение: <, >, >=, <=, <>, =. Большим считается тот символ, который имеет больший ASCII-номер.

Строковый тип (String, String[n]) - этот тип данных определяет последовательности символов - строки. Параметр n определяет максимальное количество символов в строке. Если он не задан, подразумевается n=255. Значение типа "строка" в программе запиывается как последовательность символов, заключенных в одиночные кавычки (апострофы), например

'Это текстовая строка' 'This is a string'
'1234' - это тоже строка, не число
'' - пустая строка

Допустимые операции:
- присваивание;
- сложение (конкатенация, слияние); например, **S := 'Зима'+' '+'пришла!';**
- сравнение: <, >, >=, <=, <>, =.

Строки считаются равными, если имеют одинаковую длину и посимвольно эквивалентны.

Вещественные типы - обозначают множества вещественных чисел в различных диапазонах. Имеется пять вещественных типов, различающихся диапазоном допустимых значений и размером занимаемой оперативной памяти. Вещественные типы обозначаются идентификаторами: **Real, Single, Double, Extended, Comp**; их характеристики приведены в следующей таблице.

Тип	Диапазон	Размер в байтах
Real	$2.9 \cdot 10^{-39}$ ... $1.7 \cdot 10^{38}$	6
Single	$1.5 \cdot 10^{-45}$ ... $3.4 \cdot 10^{38}$	4
Double	$5.0 \cdot 10^{-324}$ ... $1.7 \cdot 10^{308}$	8
Extended	$3.4 \cdot 10^{-4932}$ ... $1.1 \cdot 10^{-4932}$	10
Comp	$-2 \cdot 10^{63}$ ... $+2 \cdot 10^{63}-1$	8

Допустимые операции:
- присваивание;
- все арифметические: +, - ,*, / ;
- сравнение: <, >, >=, <=, <>, =.

СТРУКТУРИРОВАННЫЕ ТИПЫ ДАННЫХ

***Массив*(array).** Он представляет собой заранее известное количество однотипных элементов, снабженных индексами. Массив может быть одномерным или многомерным.

***Запись*(record).** Она включает в себя несколько полей, тип которых может отличаться друг от друга. Например, товар на складе описывается следующими величинами: наименование, количество, цена, наличие сертификата качества и т.д. В этом примере наименование – величина типа string, количество – integer, цена – real, наличие сертификата – boolean. Запись представляет собой наиболее общий и гибкий структурированный тип данных, так как она может быть образована из неоднотипных компонентов и в ней явным образом выражена связь между элементами данных, характеризующими реальный объект.

***Строка*(string)** – последовательность символов кодовой таблицы персонального компьютера. Количество символов в строке может изменяться от 0 до 255.

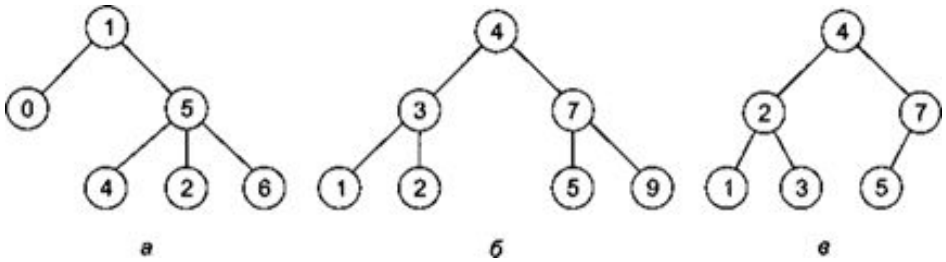
***Множество* (set)** – это набор взаимосвязанных по какому-либо признаку или группе признаков элементов. Каждый элемент во множестве называется *элементом множества*. Множество должно состоять из порядковых элементов, и их число не должно превышать 255.

***Файл*(file)** – последовательность однотипных компонентов, записанных на внешнем носителе под определенным именем. Тип этих компонентов может быть любой, за исключением типа – файла. Размер файла не объявляется.

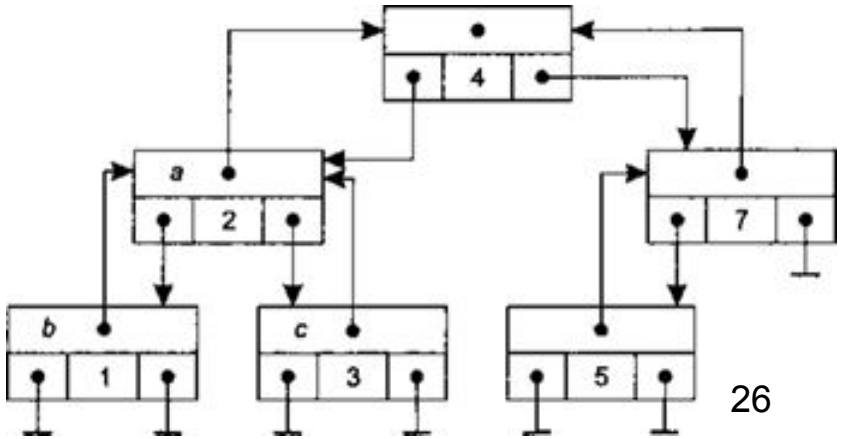
Представление и обработка данных в виде деревьев

Элементы данных могут образовывать и более сложные структуры, чем линейный список. Часто данные, подлежащие обработке, образуют иерархическую структуру, которую необходимо отобразить в памяти компьютера и, соответственно, описать в структурах данных. Такая структура получила название ***дерева.*** Каждый элемент такой структуры, называемый узлом, может содержать ссылки на элементы более низкого уровня иерархии, а может быть, и на объект, находящийся на более высоком уровне иерархии. Узел, находящийся на самом верхнем уроне иерархии, называется ***корневым.***

Корень дерева — это единственный узел, не имеющий непосредственного предка.



а *б* *в*



Представление и обработка данных в виде графов

Одной из форм визуализации информации, видом информационных моделей, которая позволяет наглядно увидеть не только объекты, но и отношения между ними, называется **графом.**

Графом является совокупность точек, соединенных между собой линиями. Точки называют **вершинами** графа. Они могут изображаться точками, кружочками, прямоугольниками и пр. Линии, соединяющие вершины, называются **дугами** (если задано направление от одной вершины к другой) или **ребрами** (если направленность двусторонняя, то есть направления равноправны).