КЕМЕРОВСКИЙ ИНСТИТУТ (филиал)
РОССИЙСКИЙ ГОСУДАРСТВЕННЫЙ ТОРГОВО-ЭКОНОМИЧЕСКИЙ УНИВЕРСИТЕТ
КАФЕДРА ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

# Информатика и программирование

Лебедева Т.Ф.

*ТЕМЫ ДОКЛАДОВ*:

1. **История развития вычислительной техники. Поколения компьютеров.**
2. **Классификация компьютеров**
3. **Основные устройства компьютера: процессоры**
4. **Основные устройства компьютера: устройства памяти**
5. **Основные устройства компьютера: устройства ввода-вывода**

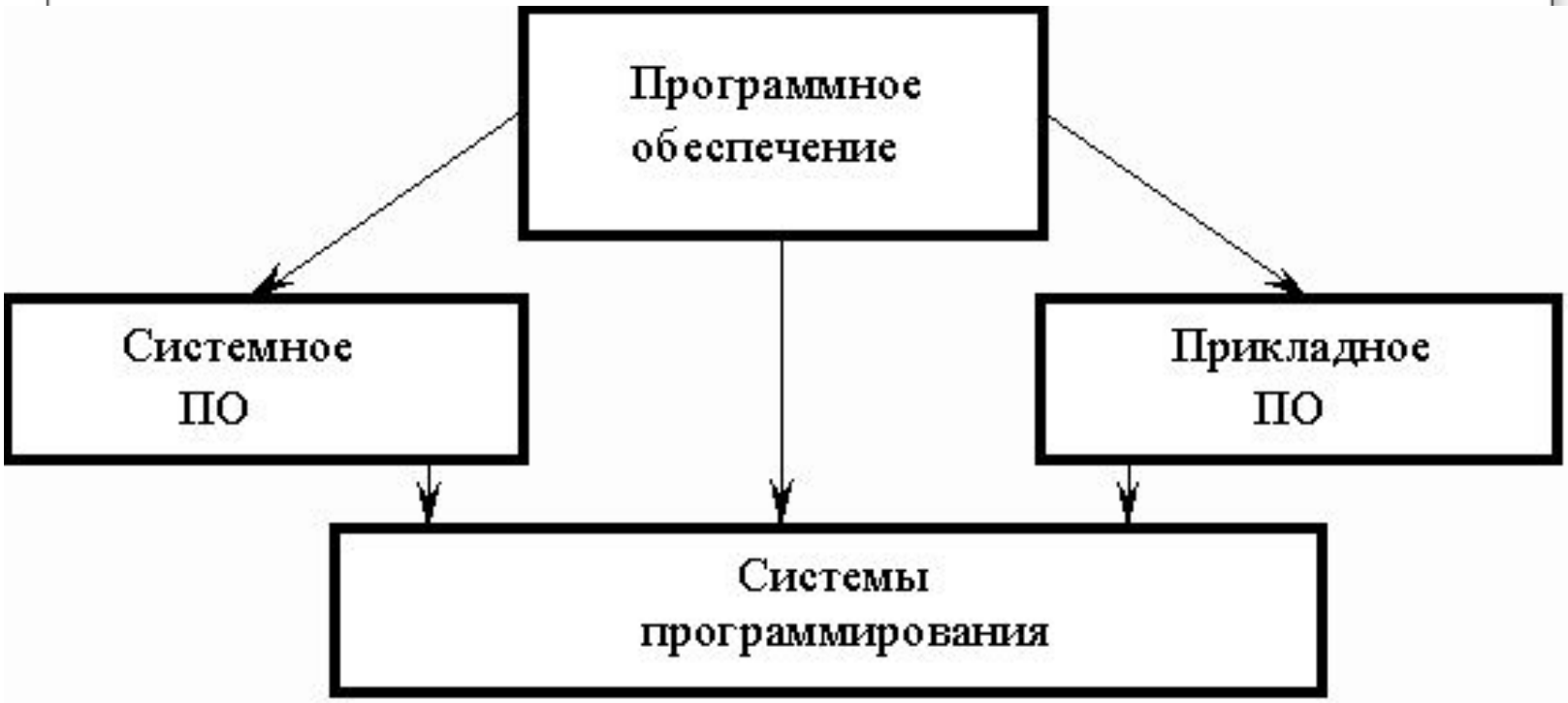
## 2.1 Классификация программного обеспечения

ЭВМ является мощным средством для решения задач, причем самых различных. Определим классы задач, решаемых с помощью компьютера:

- *Вычислительные* – классические, традиционные для ЭВМ, и характеризуются они большим объемом вычислений. К ним относятся задачи, в которых применяются численные методы, например: решение уравнений, обработка экспериментальных данных, а также инженерные расчетные задачи, проведение различных экономических вычислений.
- *Информационные* – задачи, в основе которых лежат функции хранения и обработки больших объемов информации. Под обработкой здесь понимается поиск, выборка, простейшие вычисления.
- *Другие* – задачи, охватывающие практически все сферы человеческой деятельности: работа с текстами, передача данных на большие расстояния, управление производственными агрегатами, бытовыми приборами, обучение, диагностика заболеваний, подготовка фильмов, музыки и т. д.
- **Программное обеспечение (ПО)** - это совокупность всех программ и соответствующей документации, обеспечивающая использование ЭВМ для решения задач разного класса.

Различают системное и прикладное ПО. Схематически программное обеспечение можно представить так:

## 2.1 Классификация программного обеспечения



**Системное ПО** – это совокупность программ для обеспечения работы компьютера. Системное ПО подразделяется на базовое и сервисное. Системные программы предназначены для управления работой вычислительной системы, выполняют различные вспомогательные функции (копирования, выдачи справок, тестирования, форматирования и т. д).

**Базовое ПО** включает в себя:

- операционные системы;
- оболочки;
- сетевые операционные системы.

**Сервисное ПО** включает в себя программы (утилиты):

- диагностики;
- антивирусные;
- обслуживания носителей;
- архивирования;
- обслуживания сети.

**Прикладное ПО** – это комплекс программ для решения задач определённого класса конкретной предметной области. Прикладное ПО работает только при наличии системного ПО.

Прикладные программы называют приложениями. Они включает в себя:

- текстовые процессоры;
- табличные процессоры;
- базы данных;
- интегрированные пакеты;
- системы иллюстративной и деловой графики (графические процессоры);
- экспертные системы;
- обучающие программы;
- программы математических расчетов, моделирования и анализа;
- игры;
- коммуникационные программы и др.

**Системы программирования** – это совокупность программ для разработки, отладки и внедрения новых программных продуктов.

Системы программирования обычно содержат:

- трансляторы (компиляторы или интерпретаторы);
- среду разработки программ;
- библиотеки справочных программ (функций, процедур);
- отладчики;
- редакторы связей и др.


Реальная задача
Построение математической модели
Определение исходных данных и результатов
Высказывание предложений
Выбор способа решения (постановка задачи)
Алгоритм
Программа
Ввод в ЭВМ и отладка
Тестирование
Получение результата
Анализ результата

**Построение математической модели включает в себя:**

- формализацию задачи (математическая постановка).
- выбор метода решения.

**Модель -** формализованное представление реального объекта, процесса или явления, выраженное различными средствами: математическим соотношением, числами, текстами, графиками, рисунками, словесным описанием, материальным объектом.

**Математическая модель** – система математических соотношений, описывающих процесс или явление.

В дальнейшем будем называть построение математической модели **математической постановкой задачи**.

## *2.2 Этапы решения задачи на ЭВМ*

I. *Формализация задачи (математическая постановка). Выбор метода решения.*

II. *Разработка алгоритма решения задачи.*

III. *Представление алгоритма на алгоритмическом языке.*

IV. *Ввод программы в ЭВМ.*

V. *Отладка и тестирование программы.*

VI. *Использование программы для решения задач и анализ полученных результатов.*

**Математическая постановка** представляет описание задачи с помощью математических выражений и/или словесное описание частей задачи и логических связей между ними.

Под **алгоритмом решения задачи** будем понимать конечную последовательность действий, определяющую процесс переработки исходных данных в результат.

**Выбор метода решения** необходим при наличии нескольких методов решения математически формализованной задачи.

**Алгоритмический язык** представляет собой набор символов, систем правил написания (синтаксиса) и истолкования (семантики) конструкций из этих символов, предназначенных для описания алгоритмов.

**Ввод программы в ЭВМ** осуществляется с помощью клавиатуры или из внешней памяти или из компьютерной сети. При необходимости программа "переводится" на машинный язык с помощью специальных программ - трансляторов или интерпретаторов.

**Отладка программы** представляет собой процесс обнаружения и исправления ошибок.

**Тестирование** - установление факта достоверности получаемых результатов. Правильность работы программы устанавливается с помощью специальных контрольных просчетов – **тестов**. Для тестов подбираются наборы исходных данных, для которых заранее известен результат. Анализ результатов тестирования (сравнение полученных на ЭВМ с ожидаемыми) позволяет выявить ошибки в алгоритме и программе.

## 2.2 Математическая постановка задачи

Математическая постановка или представление задачи в математическом виде и включает следующие действия:

- а) обозначение переменных;
- б) классификация переменных по типам;
- в) классификация переменных по группам: исходные данные, результаты, промежуточные результаты;
- г) запись расчетных формул и логических связей в той последовательности, в которой они должны вычисляться на ЭВМ

- ***Переменными*** называются поименованные объекты программы, значения которых могут изменяться в процессе вычисления.

Для того, чтобы полностью описать переменную, необходимо указать четыре характеристики: имя, тип переменной, ее значение и адрес.

Имена переменных, программ, функций, констант и т. д. называются ***идентификаторами.***

Под ***типом данных*** понимается множество допустимых значений этих данных. Среди типов, используемых в алгоритмических языках, есть стандартные (предопределенные) и определяемые программистом.

Обычно имеются следующие группы простых типов: **целые**, **вещественные**, **логические**, **символьные**, **строковые**.

Тип данных связан со смысловым содержанием задачи и определяет представление данных в памяти ЭВМ, т.е. количество байт памяти, занимаемой каждой переменной.

*Целый тип* выбирается для переменных, значение которых не может содержать дробной части, например, количество студентов, номер работника в списке, табельный номер. Все физические величины, коэффициенты имеют вещественный тип: масса, сила, размеры, объемы и т. д.

Для представления *вещественных чисел* используются форматы: с фиксированной точкой (десятичная запятая заменяется точкой) – 12.5678, -67854.906; с плавающей точкой или с порядком – 1.45e06, что соответствует записи 1.45*106, или –5.7e-12 - -5.7*10-12.

*Символьный тип* используется для представления данных, выражаемых одним символом (любой символ из множества, имеющихся на клавиатуре).

*Строковый тип* определяет строку любых символов, например: с=’Иванов А.И.’

*Логический тип* используется для данных, которые могут иметь только два значения: истина (true или ‘да’ или 1) и ложь (false или ’нет’ или 0).

- Можно выделить следующие группы типов: ***простые и составные***. *Простой* тип определяет упорядоченное множество значений параметра. *Составной* тип состоит из элементов других типов: массивы, записи, файлы

Рассмотрим выполнение математической постановки на примерах

**Пример 1.** Вычислить площадь треугольника, если заданы длины трех сторон треугольника.

- а) Введем обозначения переменных, т.е. дадим имена всем объектам задачи: a,b,c –длины трех сторон; s – площадь треугольника; p – полупериметр.
- б) Определим тип переменных: a,b,c,s,p – простые переменные вещественного типа.
- в) Исходные данные: a,b,c; результат: s; промежуточный результат: p.
- г) Запишем расчетные формулы в нужной последовательности:
  p=(a+b+c)/2
- s=(p(p-a)(p-b)(p-c))^0.5

При записи расчетных формул необходимо соблюдать следующее:

а) все переменные, входящие в правую часть формулы должны быть определены, т.е. вычислены ранее или введены как исходные данные.

б) рекомендуется ввести новые переменные для повторяющихся выражений, встречающихся в формулах.

Пример 2: Написать программу, вычисляющую S = y + z,

$$y = \frac{(a+b)^3 - \sin cx}{a - 3{,}75} \qquad z = \ln cx - \sqrt[3]{\frac{a+b+3{,}75}{a}}$$

Если a, b, c, x заданы.

Математическая постановка:

- а) введем новые переменные для повторяющихся выражений: t=a+b, p=cx ;
- б) a, b, c, x, t, p, y, z, s - простые переменные вещественного типа;
- в) исходные данные: a, b, c, x;
- результат: S;
- промежуточные переменные: t, p, y, z.;

- г) расчетные формулы:

$$\begin{cases} t = a + b \\ p = cx \\ y = \dfrac{t^3 - \sin p}{a - 3{,}75} \\ z = \ln p - \sqrt[3]{\dfrac{t + 3{,}75}{a}} \\ s = y + z \end{cases}$$

## 2.3 Алгоритм и его свойства

Разработка алгоритма предполагает установление последовательности вычислительных действий, управляющих связей и проверок логических условий, а также определение действий ЭВМ по вводу данных и выводу результатов. Термин "алгоритм" произошел от имени средневекового узбекского математика Аль Хорезми, который еще в 9-м веке предложил правила выполнения 4-х арифметических действий в десятичной системе счисления.

***Алгоритм*** – конечная последовательность действий, ведущая от исходных данных к результату.

Алгоритмизация не является только прерогативой математики. В обычной жизни всякой целенаправленной деятельности сопутствует заранее созданный алгоритм. таким образом, алгоритм можно трактовать как технологическую инструкцию из отдельных предписаний, выполнение которых в заданной последовательности приводит к заранее предвидимому результату.

Основные свойства алгоритмов:

• ***Массовость*** - возможность использовать один алгоритм для решения серии однотипных задач с различными вариантами исходных данных.

• ***Однозначность*** или *детерминированность* - алгоритм должен содержать конечное число предписаний, не допускающих произвола исполнителя, не оставляющих исполнителю свободы выбора. Многократное повторение алгоритма с одинаковыми исходными данными должно приводить к одному и тому же результату.

•***Дискретность*** - разделение выполнения решения задачи на отдельные операции. Поочередное выполнение команд алгоритма за конечное число шагов приводит к решению задачи, к достижению цели.

• ***Конечность*** - возможность получения результата через конечное число шагов, выполненных за конечное время. Несмотря на кажущуюся очевидность последнего свойства, оно является чрезвычайно важным, так как очень часто создаются бесконечные алгоритмы. Такая ситуация в программировании носит название «зацикливание».

**Способы записи алгоритмов**

Для записи алгоритмов используют самые разнообразные средства. Выбор средства определяется типом исполняемого алгоритма. Выделяют следующие основные способы записи алгоритмов:

- **- вербальный**, когда алгоритм описывается на естественном языке;
- **- символьный**, когда алгоритм описывается с помощью набора символов;
- **- графический**, когда алгоритм описывается с помощью набора графических изображений.

Текстовая запись алгоритмов имеет следующие недостатки: неоднозначность слов и понятий; отсутствие наглядности.

Первого недостатка удается избежать, если для описания алгоритмов использовать *псевдокоды.* В этом случае используются не любые слова естественного языка, а вполне определенные: начало, конец, ввод, вывод, если … и т.д.

Общепринятыми способами записи являются графическая запись с помощью блок-схем и символьная запись с помощью какого-либо алгоритмического языка.

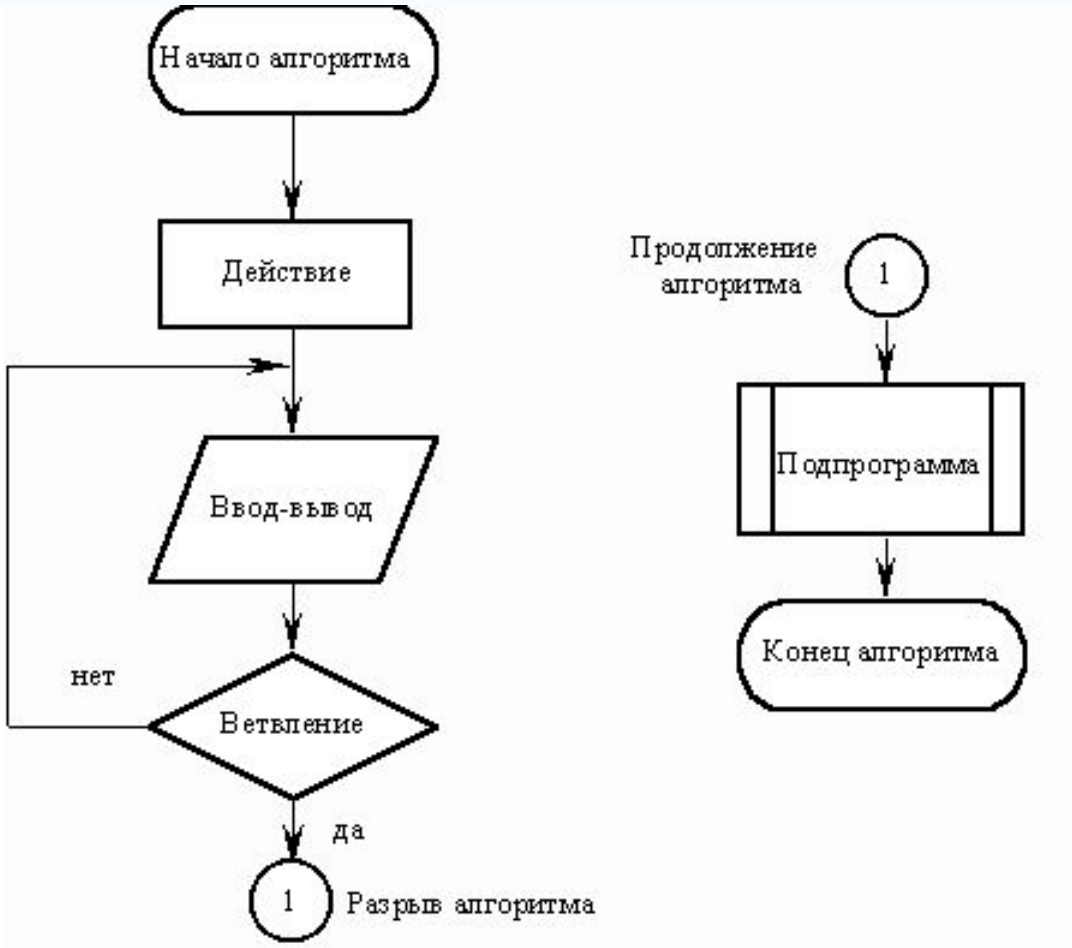
**Блок-схема отличается следующим:**

- каждому действию соответствует определенный вид фигуры (овал, прямоугольник, параллелограмм, ромб, шестиугольник);
- внутри фигур записываются формулы или краткая инструкция;
- фигуры соединяются линиями со стрелками, которые называются *линиями потока* и указывают направления перехода от одной операции к другой. Причем, если выбирается направление вниз или вправо, то стрелки можно не ставить;
- фигуры или блоки в блок-схемах могут иметь номера, проставляемые слева в разрыве верхней линии;
- линии потока не должны пересекаться, поэтому при необходимости используются *соединители* – элементы с буквой или цифрой внутри.

Конфигурация элементов схем определена

ГОСТ 19.701-90 «Схемы алгоритмов, программ, данных и систем»

# пример блок-схемы



В 1977 году математики Бем и Якопини доказали, что алгоритмы сколь угодно сложной структуры могут быть реализованы с использованием всего 3-х управляющих структур:

- Последовательное выполнение операций - ***следование***;
- Ветвление алгоритма на группы операций в зависимости от выполнения некоторых условий **-** ***ветвление***;
- Циклическое многократное выполнение группы операций до выполнения некоторого условия, формируемого в процессе вычислений **-** ***цикл***.

Соответствующие операторы в записи алгоритмов называются *условными операторами* и *операторами циклов* (следование не имеет специального оператора и выражается просто последовательной записью инструкций вычисления, ввода, вывода).

**Условные операторы в наших примерах звучат как:**

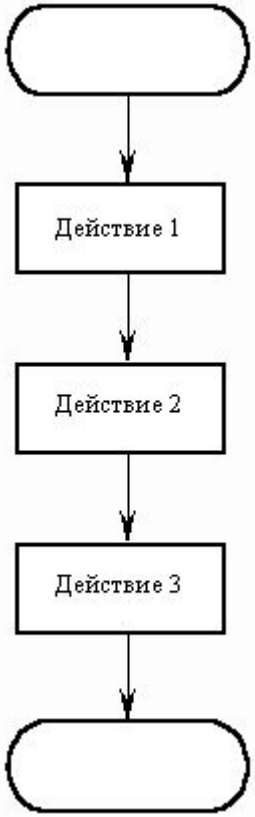
- *если* <условие выполнено> *то* последовательность операций *иначе* другая последовательность операций.

**Операторы циклов в описаниях на естественном языке мы формулируем словами:**

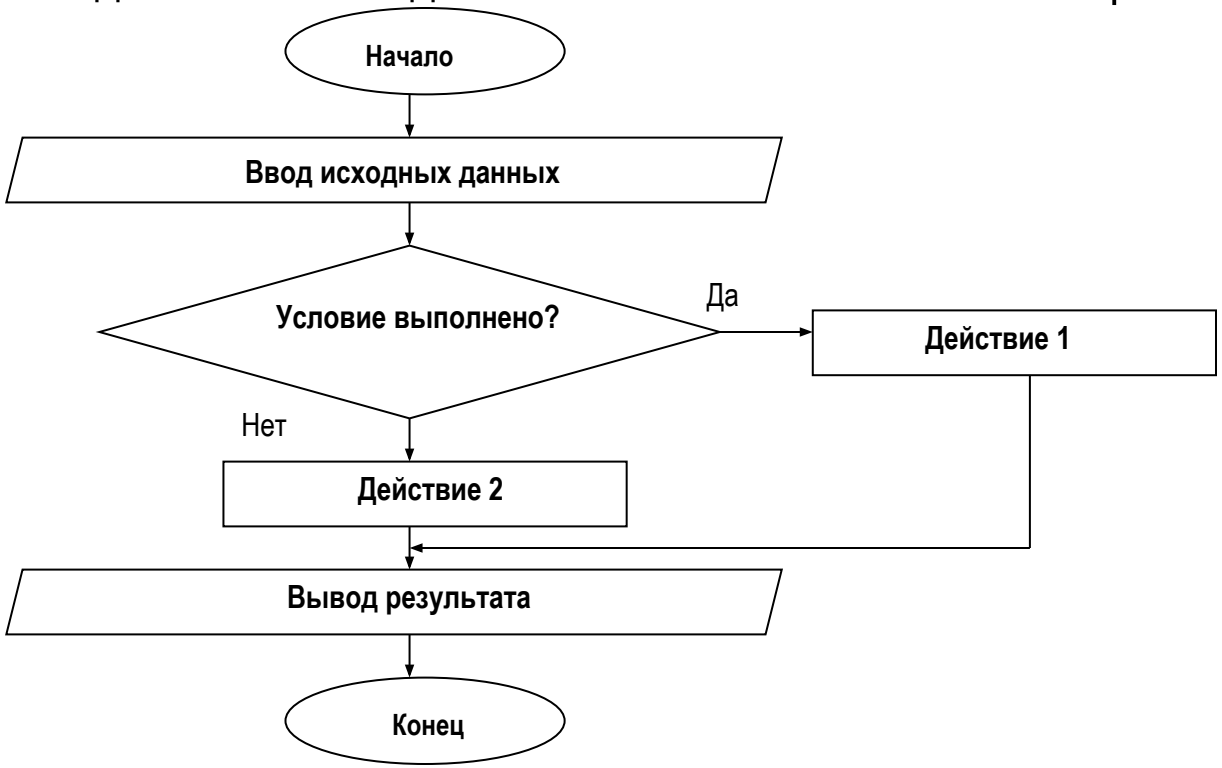
- 1. "Пока истинно некоторое условие - повторять заданные действия" (цикл с предусловием);
- 2. "Повторять заданные действия пока ложно некоторое условие" (цикл с постусловием);
- 3. "Повторять заданные действия N раз" (цикл со счетчиком).

В зависимости от последовательности выполнения действий в алгоритме выделяют алгоритмы линейной, разветвленной и циклической структуры.

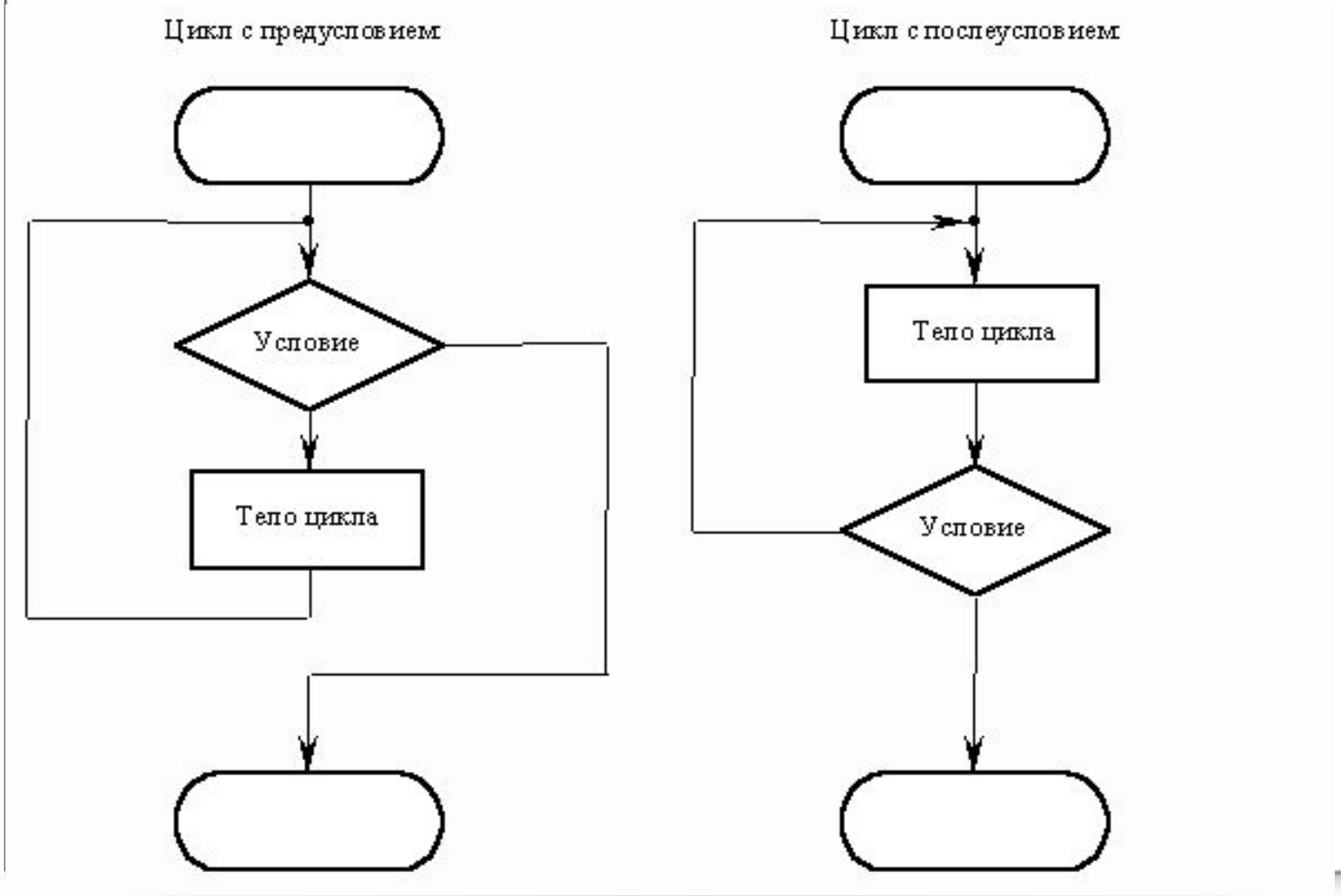
В алгоритмах **линейной структуры** действия выполняются последовательно одно за другим:



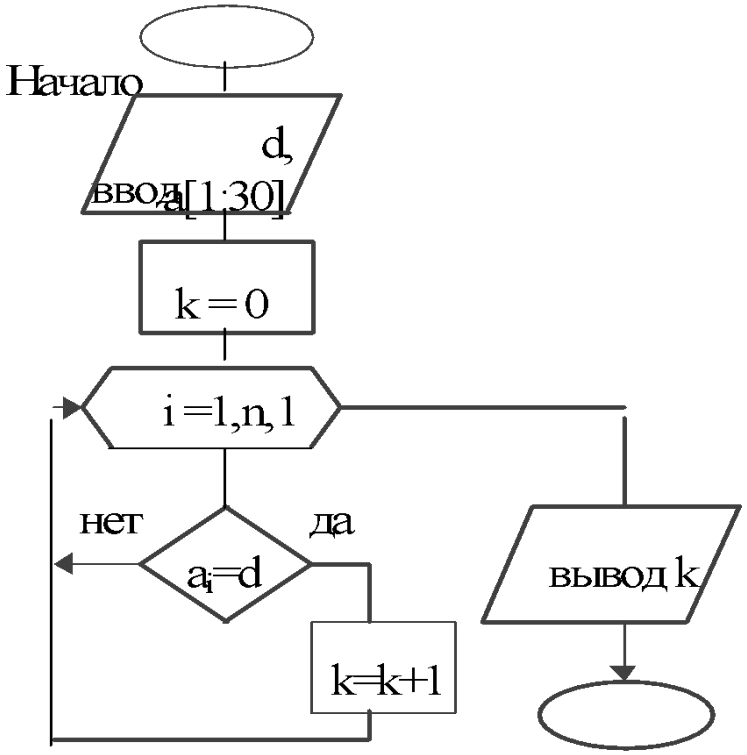
В алгоритмах **разветвленной структуры** в зависимости от выполнения или невыполнения какого-либо условия производятся различные последовательности действий. Каждая такая последовательность действий называется ветвью алгоритма.



В алгоритмах **циклической структуры** в зависимости от выполнения или невыполнения какого-либо условия выполняется повторяющаяся последовательность действий, называющаяся телом цикла.



## Цикл со счетчиком



## 2.4 Понятие алгоритмического языка

**Языки программирования** (алгоритмические языки) – специально разработанные искусственные языки, предназначенные исключительно для записи алгоритмов, исполнение которых поручается ЭВМ.

Наиболее общей классификацией языков программирования является по степени зависимости от машинного языка:

- **Машинно-зависимые –** машинные, машинно-ориентированные – низкий уровень
- **Машинно-независимые** – процедурные, проблемные – высокий уровень

**Машинно-зависимые языки** – это языки, наборы операторов и изобразительные средства которых существенно зависят от особенностей ЭВМ (внутреннего языка, адресности, структуры памяти и т.д.).

**Машинный язык –** это система команд компьютера. Программы, написанные на машинном языке не требуют компиляции**.** Пример фрагмента программы для трехадресной ЭВМ:

**001 0 0 1 1 0 1 1 0 0 1 0 1**

**код операции адрес 1-го операнда адрес 2-го операнда адрес результата**

**Машинно-ориентированные языки** - языки символического кодирования: (Автокод, Ассемблер). Операторы этого языка – это те же машинные команды, но записанные мнемоническими кодами, а в качестве операндов используются не конкретные адреса, а символические имена, например:

**ADD 1, B**

Языки программирования **высокого уровня** машинно-независимы, т.к. они ориентированы не на систему команд той или иной ЭВМ, а на систему операндов, характерных для записи определенного класса алгоритмов. Однако программы, написанные на языках высокого уровня, занимают больше памяти и медленнее выполняются, чем программы на машинных языках.

В **процедурных** языках программа явно описывает действия, которые необходимо выполнить, а результат задается только способом получения его при помощи некоторой процедуры, которая представляет собой определенную последовательность действий. Среди процедурных языков выделяют в свою очередь **структурные** и **операционные языки**. В структурных языках одним оператором записываются целые алгоритмические структуры: ветвления, циклы и т.д. В операционных языках для этого используются несколько операций. Широко распространены следующие структурные языки: Паскаль, Си, Ада, ПЛ/1. Среди операционных известны Фортран, Бейсик, Фокал.

**Проблемно-ориентированные** языки предназначены для пользователей-непрограммистов и направлены для решения узкого круга задач. На них определяется *что делать*, а не *как это делать*. К непроцедурному программированию относятся **функциональные** и **логические** языки

Еще один вид классификации языков программирования определяется существующей методологией программирования:

1) **методология императивного программирования**

языки программирования: FORTRAN (1954), ALGOL (1960) , PASCAL (1972) , C (1974)

2) **методология объектно-ориентированного программирования**

языки программирования: Simula (1962), Smalltalk (1972), C++ (1983), Object Pascal (1984), Java (1995), C# (2000)

3) **методология функционального программирования**

языки программирования: LISP (1958), Refal (1968), Miranda (1985)

4) **методология логического программирования**

языки программирования: PROLOG (1971)

5) **методология программирования в ограничениях**

языки программирования: УТОПИСТ (1980), IDEAL (1981), OPS5 (1987)

Ядра методологий определяются способом описания алгоритма.

## 2.5 Основные понятия императивного языка программирования

Обычный разговорный язык состоит из четырех основных элементов: символов, слов, словосочетаний и предложений. Алгоритмический язык содержит подобные элементы, только слова называют элементарными конструкциями, словосочетания - выражениями, предложения - операторами. Алгоритмический язык (как и любой другой язык), образуют три его составляющие: алфавит, синтаксис и семантика.

**Алфавит** – фиксированный для данного языка набор символов (букв, цифр, специальных знаков и т.д.), которые могут быть использованы при написании программы.

**Синтаксис** - правила построения из символов алфавита специальных конструкций, с помощью которых составляется алгоритм.

**Семантика** - система правил толкования конструкций языка.

Таким образом, программа составляется с помощью соединения символов алфавита в соответствии с синтаксическими правилами и с учетом правил семантики.

Методология императивного программирования – это исторически первая поддерживаемая аппаратно методология программирования. Она ориентирована на модель фон Неймана. Императивное программирование пригодно для решения задач, в которых последовательное исполнение каких-либо команд является естественным.

Основным синтаксическим понятием является оператор. Первая группа – атомарные или простые операторы, у которых никакая их часть не является самостоятельным оператором, например оператор присваивания, оператор перехода, оператор вызова процедуры.

Вторая группа – структурные (составные) операторы, объединяющие другие операторы в новый, более крупный оператор, например, составной оператор, оператор цикла, операторы выбора.

Для описания синтаксиса языка программирования будет использована расширенная **система обозначений Бэкуса-Наура**. В ней одни понятия определяются через другие последовательно.

Основные понятия заключаются в угловые скобки.

Символ **::=** означает «определяется как»

Символ **|** означает «или»

Символ ***** означает «произвольное количество повторений (в том числе ноль раз) того символа, за которым он указан»

Символы, указанные в квадратных скобках, являются необязательными.

<оператор>**::=** <простой оператор> | <структурный оператор>

<простой оператор> **::=** <оператор присваивания>|

<оператор вызова >

<структурный оператор> **::=** <оператор последовательного исполнения>| <оператор ветвления>|<оператор цикла>

<оператор присваивания> **::=** <переменная>=<выражение>

<оператор вызова > ::= <имя подпрограммы>[(<параметр> * )]

<оператор последовательного исполнения> **::= begin** <оператор> * **end**

<оператор ветвления> **::=**

**if** <логическое выражение> **then** <оператор>* **[ else** <оператор> *]

<оператор цикла> **::= while** <логическое выражение> **do** <оператор>

- **Язык программирования Паскаль. Знакомство со средой программирования Турбо Паскаль.**

Паскаль – язык профессионального программирования, который назван в честь французского математика и философа Блеза Паскаля (1623–1662) и разработан в 1968–1971 гг. Никлаусом Виртом. Первоначально был разработан для обучения, но вскоре стал использоваться для разработки программных средств в профессиональном программировании.

Турбо Паскаль – это система программирования, созданная для повышения качества и скорости разработки программ (80-е гг.). Слово Турбо в названии системы программирования – это отражение торговой марки фирмы-разработчика Borland International (США).

Систему программирования Турбо Паскаль называют *интегрированной* (integration – объединение отдельных элементов в единое целое) средой программирования, т.к. она включает в себя редактор, компилятор, отладчик, имеет сервисные возможности.

Основные файлы Турбо Паскаля:

Turbo.exe – исполняемый файл интегрированной среды программирования;

Turbo.hlp – файл, содержащий данные для помощи;

Turbo.tp – файл конфигурации системы;

Turbo.tpl – библиотека стандартных модулей, в которых содержатся встроенные процедуры и функции (SYSTEM, CRT, DOS, PRINTER, GRAPH, TURBO3, GRAPH3).

## Структура программы

Программа на Borland Pascal состоит из трех частей:

- заголовка,
- раздела описаний,
- раздела операторов.

*Заголовок программы* не является обязательным, он состоит из служебного слова program и идентификатора - имени программы.

*Р а з д е л  о п и с а н и й* содержит описания всех используемых *программой ресурсов* (полей данных, подпрограмм и т.д.).

*Р а з д е л  о п е р а т о р о в* заключается в так называемые *операторные скобки* **begin ...end** и заканчивается точкой.

Между операторными скобками записывают управляющие операторы программы, которые разделяют специальным знаком - точкой с запятой «;». Если точка с запятой стоит перед end, то считается, что после точки с запятой стоит «пустой» оператор.

В тексте программы возможны комментарии, которые помещают в фигурные скобки.

Посмотрим, как выглядит Pascal программа, которая реализует алгоритм Евклида для определения наибольшего общего делителя двух чисел a и в:

```
Program example; {заголовок программы}
{раздел описаний}
Var a,b:integer; {объявление переменных}
{раздел операторов}
Begin
    Write ('Введите два натуральных числа:') ;
 {запрашиваем ввод данных}
    Readln(a,b); {вводим значения}
    while a<>b do   {цикл-пока a ≠b}
      if a>b then a:=a - b {если a>b, тогда a:=a-b} else b:= b-a;
{иначе b:=b-a}
    Writeln('Наибольший общий делитель равен ',a);
 {выводим результат}
End.    {конец программы}
```

- **Программы на языке Паскаль имеют блочную структуру:**

1. Блок типа PROGRAM – имеет имя, состоящее только из латинских букв и цифр. Его присутствие не обязательно, но рекомендуется записывать для быстрого распознавания нужной программы среди других листингов.
2. Блок описаний состоит в общем случае из 7 разделов:
   - раздел описания модулей (**uses**);
   - раздел описания меток (**label**);
   - раздел описания констант (**const**);
   - раздел описания типов данных (t**ype**);
   - раздел описания переменных (**var**);
   - раздел описания процедур и функций;
3. Операторный блок, который содержит раздел описания операторов.

Общая структура программы на языке Паскаль следующая:

```
Program ИМЯ..;  {заголовок программы}
Uses ...;               {раздел описания модулей}
Var ..;          {раздел объявления переменных}
  ...
Begin            {начало исполнительной части программы}
  ...          {последовательность
  ...            операторов}
End.           {конец программы}
```

# Вопросы?