# Файловая структура. Хранение и доступ к информации

- Диалог с машиной строго регламентирован.
- Байт = 8 бит (ранее был машинно-зависимым, 7-9 бит);
- Слово = 2 байта;
- Двойное слово = 4 байта;
- Учетверенное слово = 8 байт.

Большие объемы информации измеряются в следующих единицах

- Килобайтах – $2^{10}$ (1024),
- Мегабайтах - $2^{20}$
- Гигабайтах – $2^{30}$
- Терабайтах - $2^{40}$

- В качестве единицы хранения данных принят объект переменной длины называемый **файлом.**
- **Файл – это последовательность произвольного числа байтов, обладающая собственным уникальным именем.**
- Доступ к файлу осуществляется по имени, а не по адресу (как в записях внутри файла или блоках на диске).

- Современные операционные системы позволяют разместить на одном физическом диске несколько файловых систем, выделив для каждой из них фиксированную часть диска – логические диски.

# Система управления файлами

Специальное программное обеспечение реализует работу с файлами по принятым спецификациям файловой системы. Это обеспечение называется – **системой управления файлами.**

- Работа с недисковыми периферийными устройствами как с файлами.
- Обмен данными между файлами, устройствами, файлами и устройствами.
- Защита файлов от несанкционированного доступа.
- Создание, удаление, переименование файлов (наборов данных) посредством специальных управляющих программ реализующих пользовательский интерфейс.

- Через файловую систему осуществляется связь системных обрабатывающих программ.
- Централизованное распределение дискового пространства.
- Управление данными.

- Для решения этих задач необходимо, чтобы данные имели строго упорядоченную структуру.
- Они должны включать в себя «дополнительную нагрузку» в виде адресных данных, без которых нельзя получить доступ к интересующей нас информации.
- Система хранит таблицу преобразования имен файлов в адреса – **директорию или каталог**. Адресные данные тоже имеют определенный размер и тоже подлежат хранению.

Система FAT поддерживается абсолютным большинством операционных систем – MS DOS, OS/2, Windows 95/98/ME, Windows NT/2000/XP, Linux.

**FAT – File Allocation Table** – это таблица, в которой указывается:

- Непосредственно адресуемые участки логического диска, отведенные для размещения на них файлов или их фрагментов.
- Свободные области дискового пространства.
- Дефектные области диска.

- Хранимые данные могут быть разных типов, их тип и определяет **тип файла.**
- Типы файлов являются дополнительными сведениями для использования адекватного метода их извлечения из файла. Этот метод определяется автоматически по указанному типу файла.

# Категории файлов

По форме хранимой информации файлы делятся на две категории: текстовые и двоичные.

- Текстовый файл – предназначен для чтения пользователем,
- Файлы не являющиеся текстовыми – двоичные.

# Имена файлов

- Имя файла (набор символов - букв, цифр и спецзнаков) состоит из двух частей – собственно имени и расширения, которое не является обязательным.
- В Pascal имя может содержать до 8 символов, расширение – до 3 (система FAT).
- Системы FAT32,  NTFS работают с длинными именами.

*Примеры: command.com, config.sys, autoexec.bat.*

- *.com, .exe – исполняемые файлы*
- *.bat – командный файл*
- *.pas - паскаль*
- *.for - фортран*
- *.c - си*
- *.asm – ассемблер*

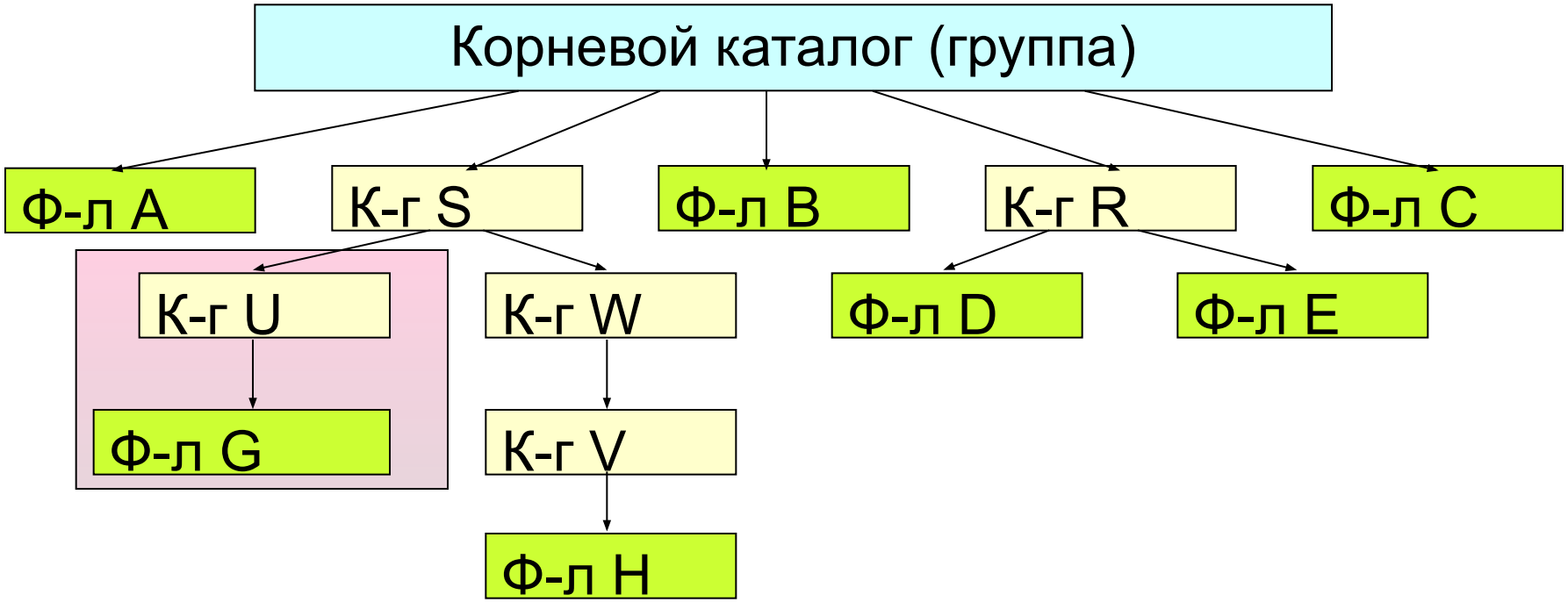
# КАТАЛОГИ

- Имена файлов регистрируются в каталогах (директориях).
- Каталоги (просто списки имен файлов) – специальное место на диске, в котором хранятся сведения о записанных файлах (имена, размеры, начальный адрес, атрибуты, даты создания и т.д.).
- Любая информация, хранящаяся на диске, записана в файл, то и Каталог является Файлом специфического вида.

- Детальная информация о расположении каждого блока данных, назначенного каталогу или файлу, хранится в специальной управляющей области диска, называемой «таблицей размещения файлов» (FAT).
- На диске может быть размещено множество каталогов,  в каждом из них записано множество файлов. Но каждый файл регистрируется только в одном каталоге.

- Каждый диск, в принципе, может содержать каталоги двух типов: корневой каталог и все остальные каталоги.
- Корневой каталог всегда присутствует на диске и максимально допустимое число записей на нем определяется в процессе формирования диска и не может быть изменено.
- Подкаталоги корневого каталога могут иметь любой уровень вложенности и любой размер.

# Файловая система




- Каталог U – текущий.
- Путь доступа из каталога U к файлу H:
  .. \ W \ V \ H
- или  Корневой \ S \ W \ V \H

- Если в команде MS-DOS указывается имя файла, то этот файл будет искаться в текущем каталоге. Если возникает необходимость указать в команде файл не из текущего каталога, то вместе с именем файла нужно указать и путь доступа к нему.
- Путь – последовательность имен каталогов, разделенных символом « \ », которая описывает маршрут от текущего (или корневого) каталога к каталогу, содержащему искомый файл. В записи пути может использоваться символ « .. », что означает подъем на уровень выше по иерархической структуре.

***Пример:***

***текущий каталог « U », путь доступа к файлу « E» может быть следующим:***

- ***..\..\R\E- из текущего;***
- ***R\E - из корня.***
- Строка, состоящая из обозначения логического дисковода, полного пути от корневого каталога и имени файла с расширением, образует **полностью квалифицированное имя файла**, однозначно описывающее этот файл.

  ***Дисковод:\путь\имя_файра.расширение.***

  Напрмер*:* ***D:\R0\R\F***

# ОСНОВНЫЕ КОМАНДЫ MS-DOS

Остановка выполнения команды.

- Ctrl-C
- Ctrl-Break

Приостановка вывода информации на экран.

- Ctrl-S

Действия с каталогами и файлами.

Смена текущего дисковода.

- > A: переход на диск A

Изменение текущего каталога.

- >cd  [дисковод:]путь

Вывод содержимого файла на экран.

- >type  [дисковод:][путь]\имя_файла

Удаление файлов.

- >del [дисковод:][путь]\имя_файла

Просмотр каталога.

- >dir [дисковод:]путь

Создание каталога.

- >md  [дисковод:]путь

Уничтожение каталога.

- >rd  [дисковод:]путь

Переименование файла.

- >ren [дисковод:][путь]имя_файла_ст [дисковод:][путь]имя_файла_нов

Копирование файла.

- >copy [дисковод:][путь]имя_файла_0 [дисковод:][путь]имя_файла_1

Примечание: Можно использовать символы “ * “ и “? “ в написании имен.

- *.pas – любой набор символов в имени файла
- proba.* - любое расширение имя файла
- ?roba.pas – любой символ в имени файла на первой позиции
- pr?ba.pas - любой символ в имени файла на третьей позиции

- >DIR содержимое текущего каталога
- >DIR/P постраничный вывод
- >DIR путь к каталогу содержимое этого каталога
- >DIR \ содержимое корневого каталога
- >DIR имя_файла информация о файле
- >DIR \>CATALOG.LIST перенаправление потока в файл

Изменение текущего каталога.

- > CD \ переход к корневому каталогу
- > CD .. переход на уровень выше

# Основные принципы алгоритмизации

**Понятие алгоритма.**

- Алгоритм – конечная последовательность строго и четко сформулированных правил, которые позволяют решить те или иные задачи.

**Свойства алгоритмов.**

- Каждое указание алгоритма предписывает выполнение только одного конкретного действия.
- Нельзя перейти к другому указанию, не выполнив предшествующее указание.
- Разделение задачи на отдельные конкретные операции называется дискретностью алгоритма.
- Отдельные указания алгоритма называются командой.

При решении задачи (локальной или общей) необходимо производить либо логические, либо арифметические действия. Исходя из этого, алгоритмы можно классифицировать как:

- Поиска и выборки,
- Сортировки,
- Численные,
- Сравнения с образцом,
- На графах,
- Параллельные,
- Недетерминированные.

# Алгоритмы поиска и выборки, сортировки

- Алгоритмы поиска и выборки, сортировки связаны с обработкой больших массивов данных – нахождением нужных элементов массива и его упорядочения.

# Численные алгоритмы

- решение алгебраических и трансцендентных уравнений,
- решение систем линейных алгебраических уравнений,
- решение систем обыкновенных дифференциальных уравнений,
- решение уравнений в частных производных,
- оптимизация,
- обработка числовых данных.

# Алгоритмы

- Сравнения с образцом – сравнение строк.
- На графах – построение топологических матриц (например, карта дорог).
- Параллельные – расчет на нескольких процессорах.
- Недетерминированные алгоритмы связаны с большим временем счета. Решение нужно «угадывать».

# Анализ алгоритмов

- Одну и ту же задачу можно решать различными алгоритмами.
- Прежде чем определять эффективность алгоритма нужно доказать правильность решения:
- Сравнить с аналитикой;
- Проверить на предельные случаи;
- Сравнить с ранними версиями;
- Проверить экспериментом.

- Эффективность определяется по числу операций и зависимости числа операций от размера входных данных.

- Классы входных данных.
- Пример: Определить максимальное число из 10 чисел. Комбинаций м.б. 3 628 800.

Классов – 10:

максимальное число на первом месте

максимальное число на втором месте

и т.д.

# Нисходящее проектирование.

- Наиболее эффективное – это проектирование и разработка программ сверху вниз. Сначала выделяются глобальные модули. Каждый из этих модулей затем детализируется. Процесс продолжается до самого низшего уровня вложенности модулей.
- В этом методе хорошо просматривается связь алгоритма программы со смысловым содержанием самой задачи. В идеале основная программа состоит из обращений к логическим блокам (модулям), которые в свою очередь также могут состоять из обращений к модулям, содержащим более подробную проработку их содержания. И так до модулей самого глубокого уровня, в которых и производятся конкретные вычислительные действия.

Пример: Графическое отображение сортировки массива чисел.

- Ввод начальных данных
- Математическая обработка массива.
- Подготовка данных для графики.
- Графическое отображение результатов.
- Завершение программы.

Пример: решить уравнение $a*x^3+b*x^2+c*x+d=0$

- Нужно предусмотреть возможность изменения коэффициентов
- В более широком понимании (исходя из пользовательских потребностей) предусмотреть возможность записи уравнений более высокой (но ограниченной) степени.
- Здесь требуется именно пользовательское понимание сути задачи, чтобы не усложнять алгоритм введением каких-либо иных формул записи уравнений.

При разработке алгоритма можно выделить несколько этапов:

- Построить блочную структуру программы, отражающую логику работы алгоритма.
- Представить, какие численные алгоритмы нужно использовать в расчетных блоках.
- Оценить быстродействие разработанной схемы.

**Формы записи алгоритмов.**

- На обычном языке.
- На алгоритмическом языке.
- В виде блок-схемы.

# Алгоритм: описание по шагам.

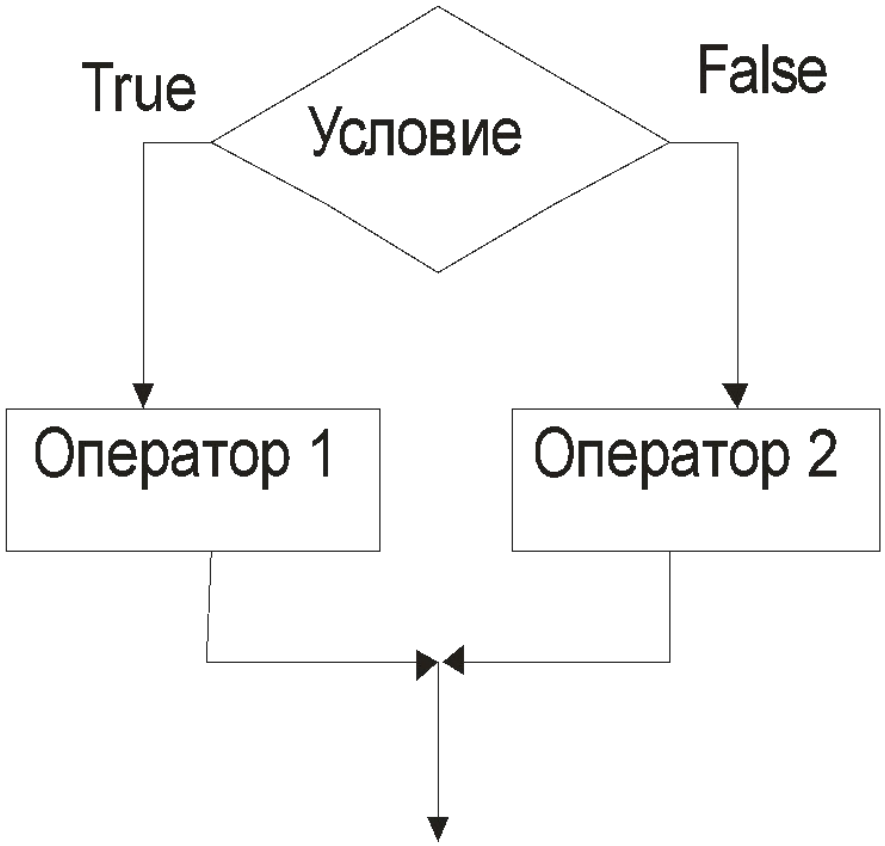
- шаг1: задать n, a(1)
- шаг2: задать m=a(1), i=1
- шаг3: если n>i, то перейти к шагу 4; иначе перейти к шагу 7.
- шаг4: положить i=i+1, Ввести a(i)
- шаг5: если a(i)<m, то перейти к шагу 6 ; иначе перейти к шагу 3.
- шаг6: положить m=a(i), перейти к шагу 3
- шаг7: Вывод m

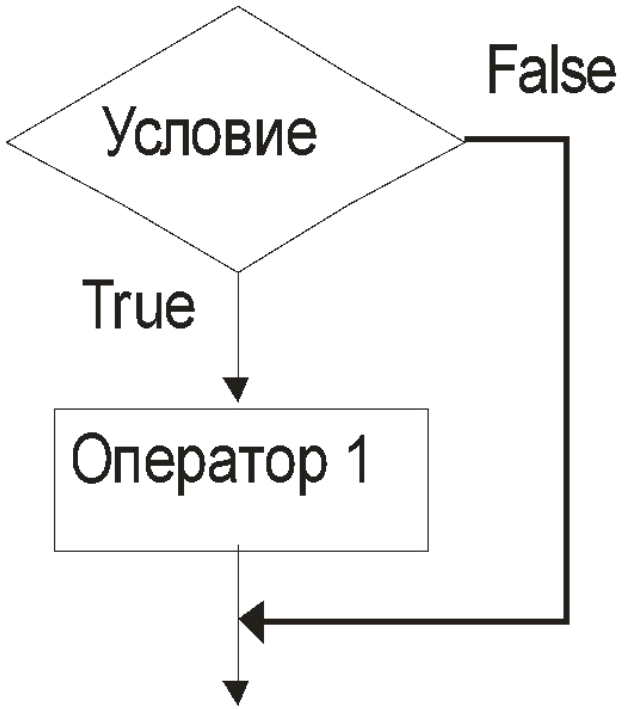
# Алгоритм: псевдоязык.

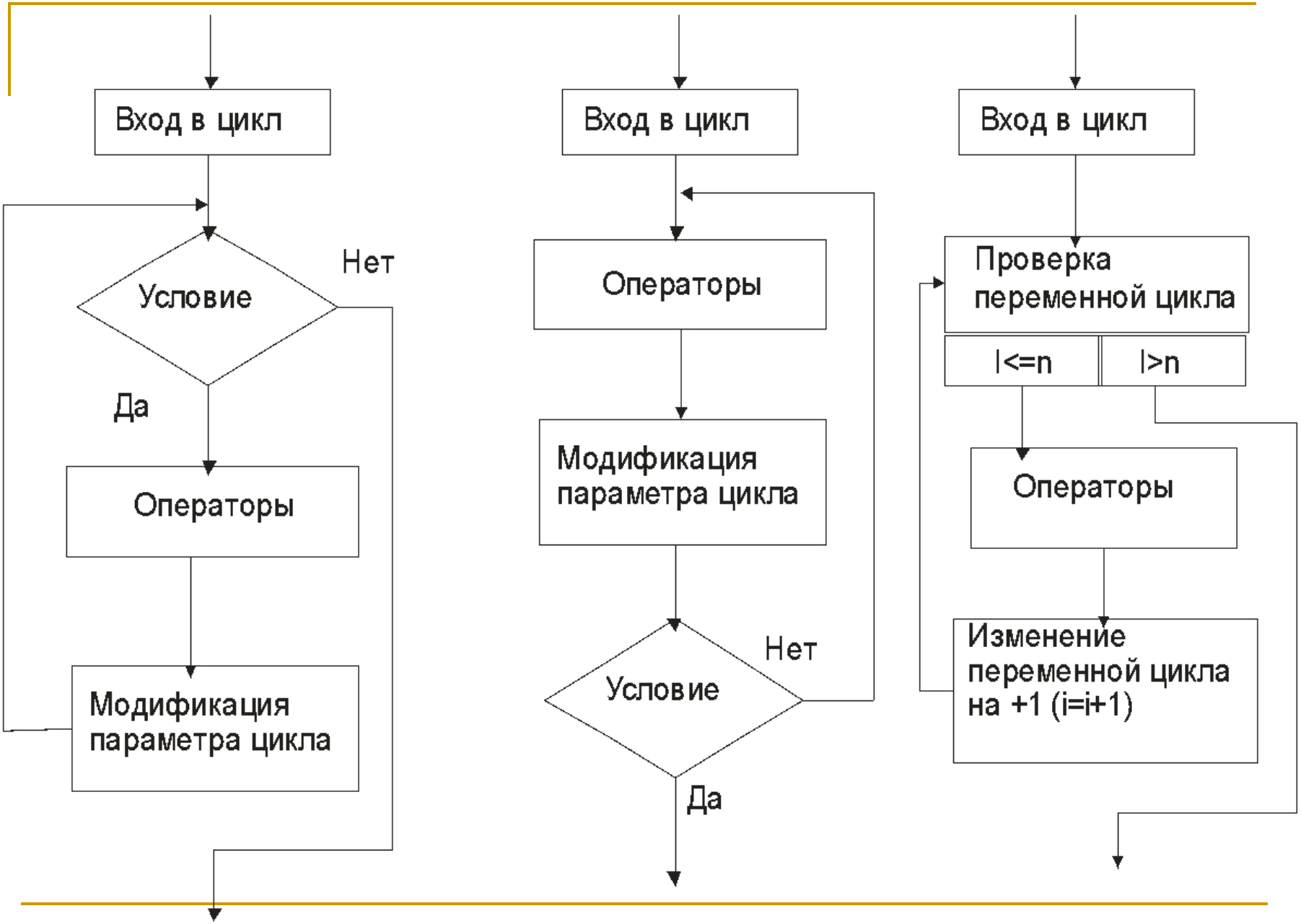
```
      Read  n, a(1)
1:     If n>i then
       i=i+1, read  a(i)
       else
         goto 2
     If a(i)<m then
       m=a(i), goto 1
2:   write  m
```

По форме своей организации алгоритмы подразделяются на следующие виды:

- Линейные,
- Разветвляющиеся,
- Циклические


True
Условие
False
Оператор 1
Оператор 2


Условие
False
True
Оператор 1


Вход в цикл
Условие
Нет
Да
Операторы
Модификация параметра цикла
Вход в цикл
Операторы
Модификация параметра цикла
Условие
Нет
Да
Вход в цикл
Проверка переменной цикла
I<=n
I>n
Операторы
Изменение переменной цикла на +1 (i=i+1)

- Одной из основных проблем программирования является быстродействие программы, которое определяется количеством операций сравнения, количеством арифметических операций (аддитивные и мультипликативные).
- Как правило, для различных наборов начальных данных существуют три варианта реализации алгоритма – наихудший, наилучший и средний.

Пример: Вычисление многочлена – схема Горнера.

$$P(x) = a_0 + a_1 * x + a_2 * x^2 + a_3 * x^3 + a_4 * x^4 + a_5 * x^5 + .....a_N * x^n$$

$$P(x) = a_0 + (a_1 + (a_2 + (a_3 + (a_4 + (a_5 + .....(a_{n-1} + a_N * x\ )...) * x) * x) * x) * x) * x)$$

- При n=5 в первой схеме производим 20 умножений и 5 сложений, а в схеме Горнера – 5 умножений и 5 сложений.