Алгоритмизация и программирование

План

1. Понятие алгоритма и его свойства
2. Способы описания алгоритмов
3. Основные алгоритмические конструкции
4. Базовые алгоритмы
5. Простые и структурированные типы данных
6. Создание программ
7. Основные понятия языка программирования Basic
8. Классификация и обзор языков программирования
9. Методы проектирования программ
10. Жизненный цикл программного обеспечения

1. Понятие алгоритма и его свойства

Алгоритм (от algoritmi)- предписание, однозначно задающее процесс преобразования исходной информации в виде последовательности элементарных дискретных шагов, приводящих за конечное число их применений к результату.

Мухаммед ибн Муса аль-Хорезми (783-850)

Разновидности алгоритмов:

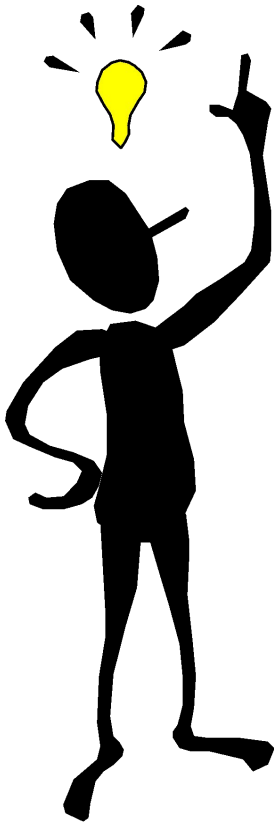
вычислительные – работают с простыми видами данных (числа, векторы, матрицы), но процесс вычисления может быть длинным и сложным;

информационные – реализуют небольшие процедуры обработки (например, поиск элементов, удовлетворяющих определенному признаку), но для больших объемов информации;

управляющие – непрерывно анализируют информацию, поступающую от тех или иных источников, и выдают результирующие сигналы, управляющие работой тех или иных устройств.

Свойства алгоритма

- *Дискретность* – последовательное выполнение простых или ранее определённых (подпрограммы) шагов. Преобразование исходных данных в результат осуществляется дискретно во времени.
- *Понятность* – каждая команда алгоритма должна быть понятна тому, кто исполняет алгоритм; в противном случае, эта команда и, следовательно, весь алгоритм в целом не могут быть выполнены.
- *Определенность* - каждое правило алгоритма должно быть четким, однозначным и не оставлять места для произвольного толкования.
- *Результативность* означает возможность получения результата после выполнения конечного количества операций.
- *Корректность* - решение должно быть правильным для любых допустимых исходных данных.
- *Массовость* заключается в возможности применения алгоритма к целому классу однотипных задач, различающихся конкретными значениями исходных данных (разработка в общем виде).

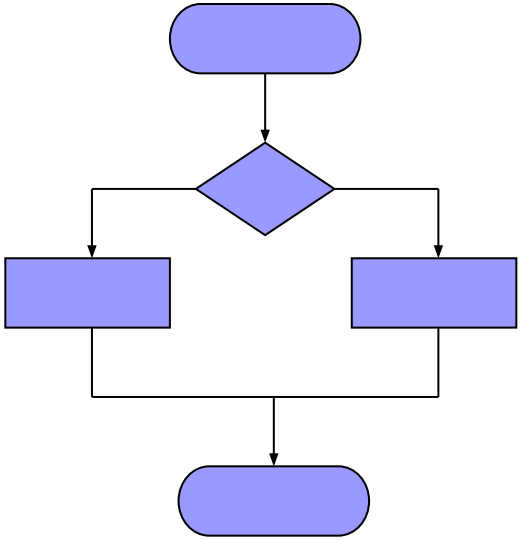


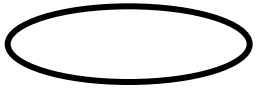
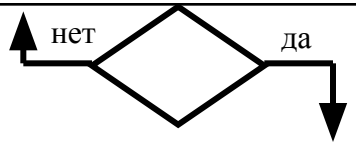
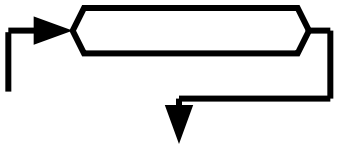
Составление алгоритма является обязательным этапом автоматизации любого процесса.

2. Способы описания алгоритмов

- словесный (на естественном языке);
- формульно-словесный;
- табличный (обычно носит вспомогательный характер);
- графический (использует элементы блок-схем).

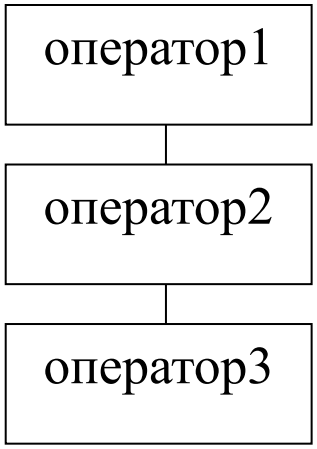
Блок-схема - графическое изображение структуры алгоритма, в котором каждый этап процесса переработки данных представляется в виде геометрических фигур (блоков), имеющих определенную конфигурацию в зависимости от характера выполняемых при этом операций.

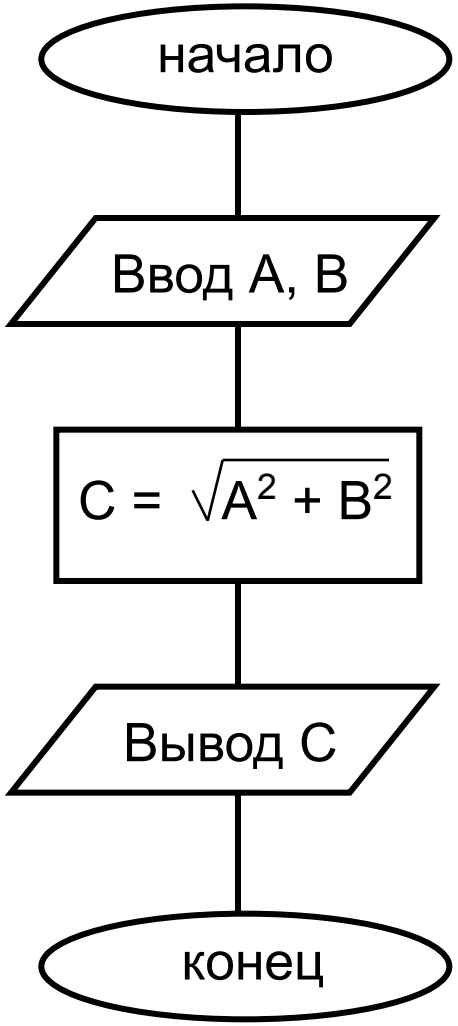


	- начало или конец алгоритма
	- ввод/вывод данных или результата на экран монитора
	- процесс – арифм.выражение или операция присваивания
	- проверка условия
	- подпрограмма
	- вывод на принтер
	- циклический процесс.

3. Основные алгоритмические конструкции

Линейным принято называть вычислительный процесс, в котором этапы вычислений выполняются в линейной последовательности и каждый этап выполняется только один раз.





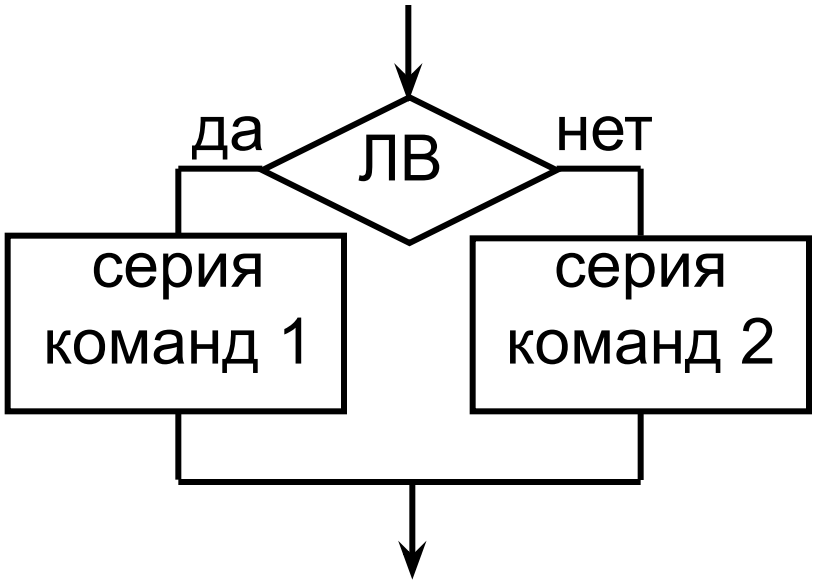
Блок-схема вычисления гипотенузы по теореме Пифагора

Разветвляющийся вычислительный процесс реализуется по одному из нескольких заранее предусмотренных направлений (ветвей) в зависимости от выполнения некоторого условия (логического выражения - ЛВ).

Ветвящийся процесс, включающий в себя две ветви, называется *простым*, более двух ветвей - *сложным*.

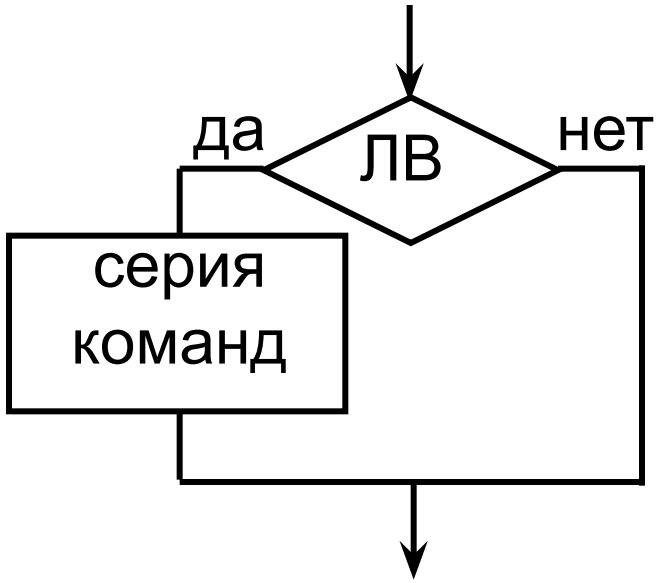
полное ветвление

если-то-иначе



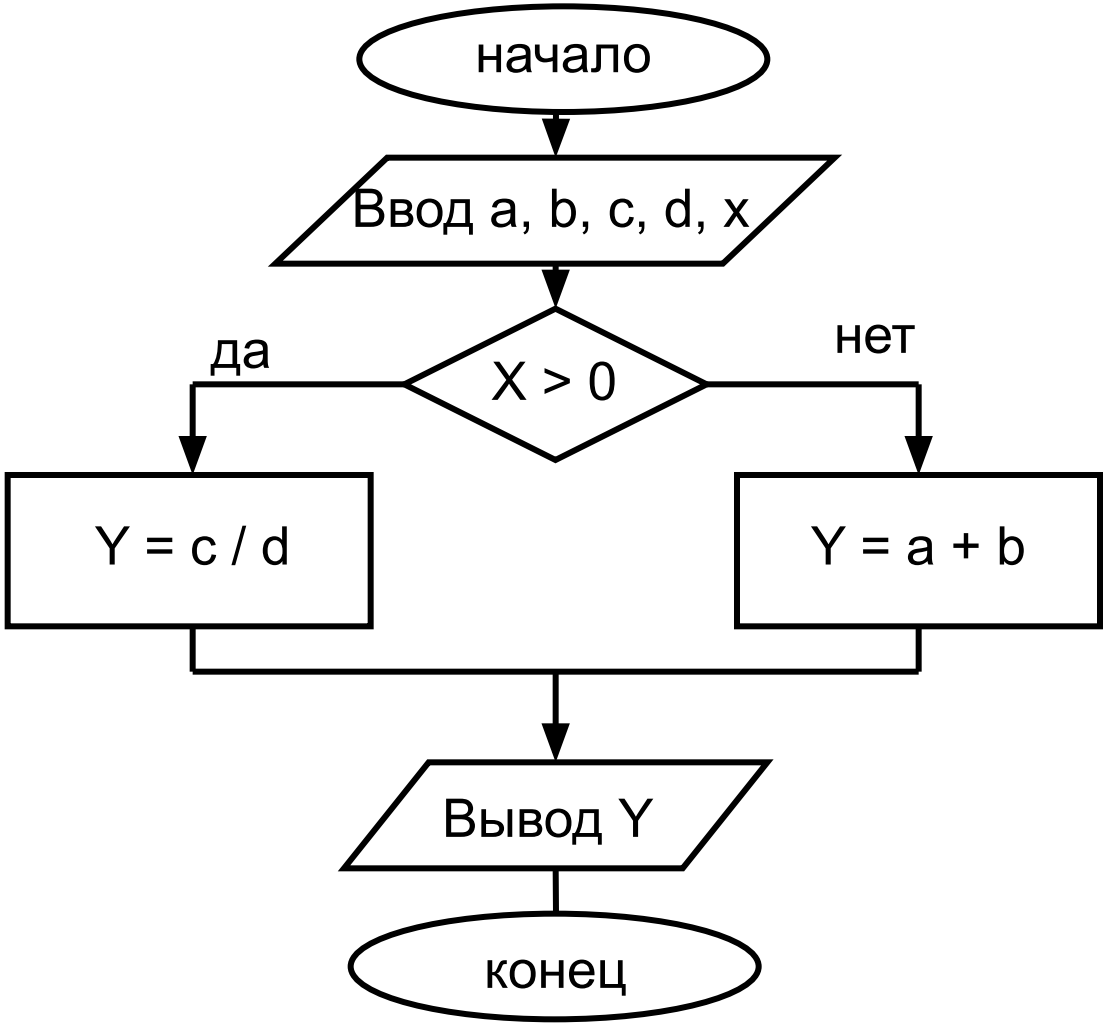
неполный вариант ветвления

если-то



Алгоритм вычисления функции:

$$y = \begin{cases} a + b, \text{если } X \le 0 \\ c/d, \text{если } X > 0 \end{cases}$$

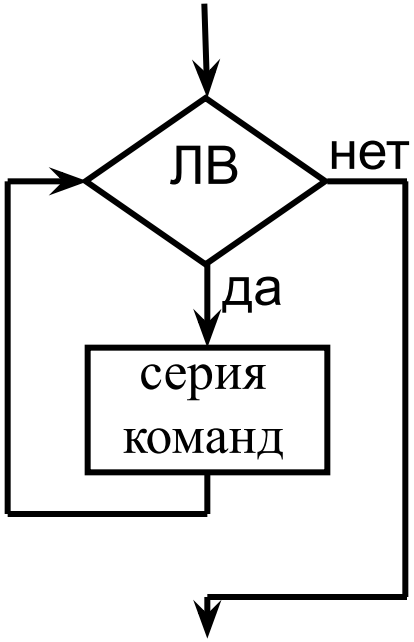


Циклический вычислительный процесс (цикл) включает участки, на которых вычисления выполняются многократно по одним и тем же математическим формулам, но при разных значениях исходных данных.

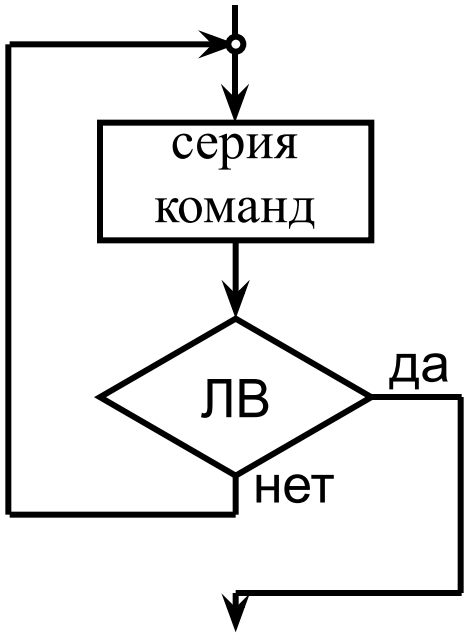
Цикл называется ***детерминированным (цикл с параметром),*** если число повторений тела цикла заранее известно или определено.

Цикл называется ***итерационным (с пред- и постусловием),*** если число повторений тела цикла заранее неизвестно, а зависит от значений переменных, участвующих в вычислениях.

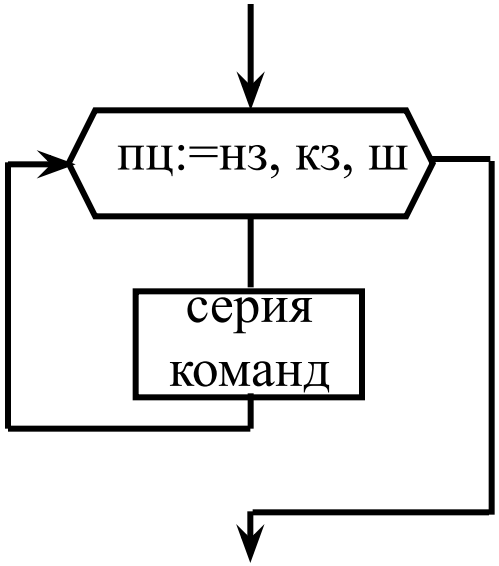
цикл с предусловием



цикл с постусловием



цикл с параметром

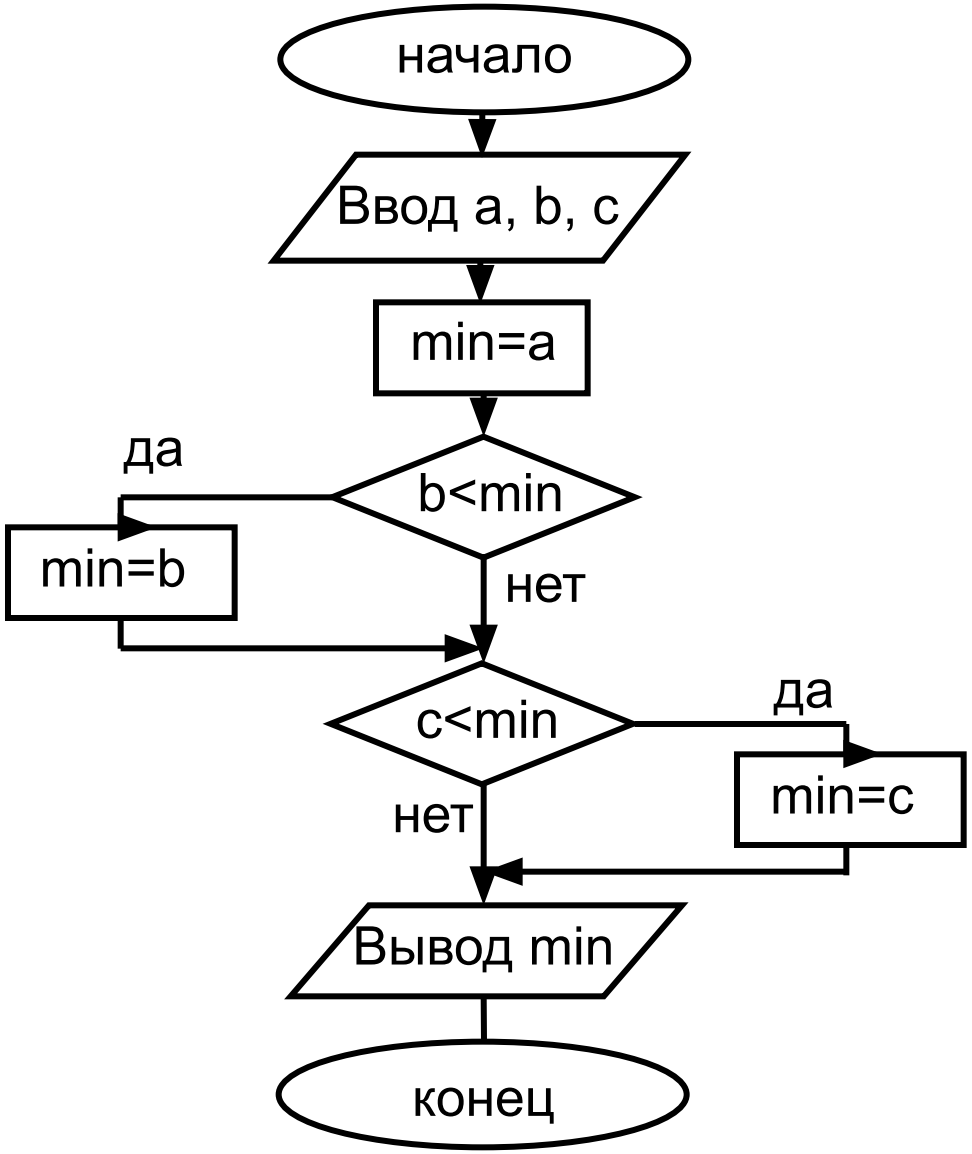


4. Базовые алгоритмы

Алгоритм поиска наибольшего (наименьшего) значения:

за max (min) принимаем значение любого из данных и поочередно их сравниваем. Если окажется, что очередное значение входного данного больше (меньше) max (min) , то max (min) присваиваем это значение. Алгоритм использует неполное ветвление.

Пример. Заданы три числа a, b, c. Найти значение наименьшего из них.



a=9 b=3 c=5

min=9

3<9

min=3

5<3

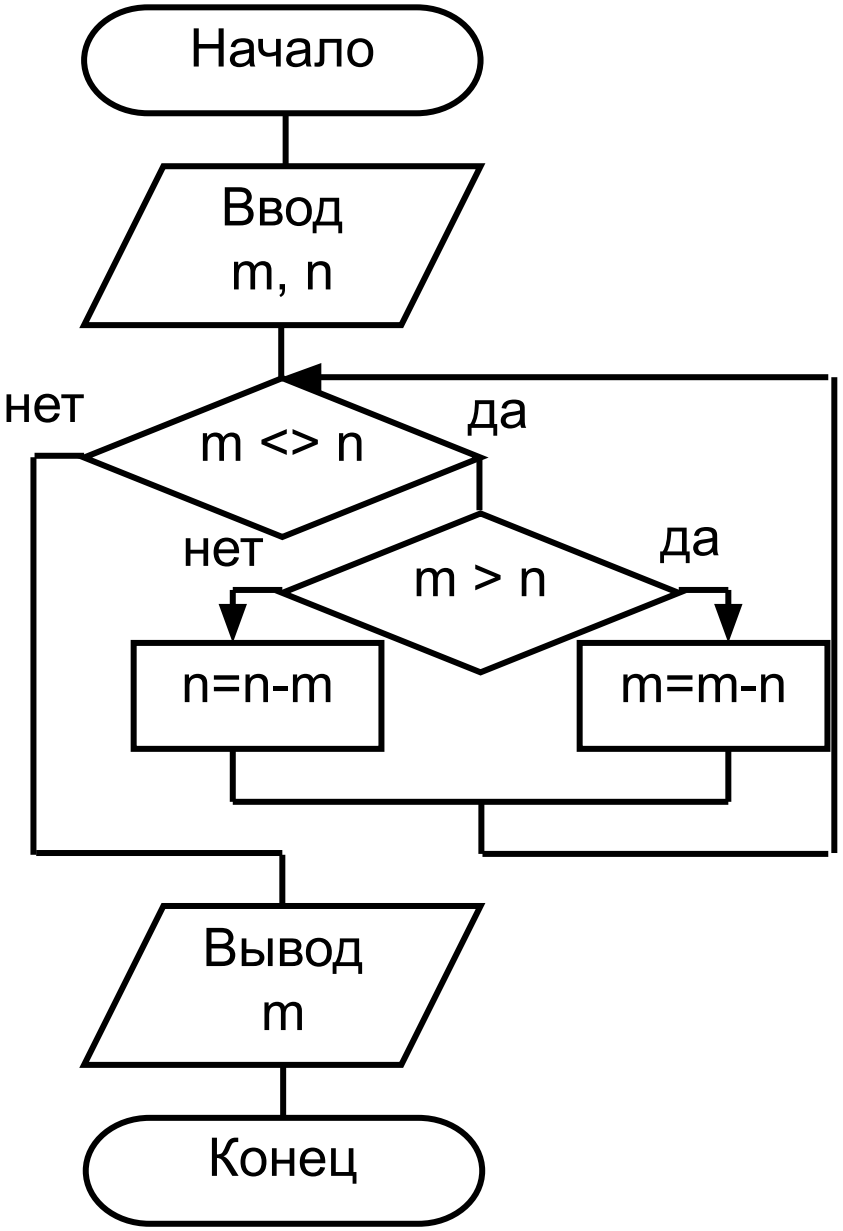
Алгоритм Евклида – алгоритм нахождения НОД (наибольшего общего делителя) двух натуральных чисел *m* и *n* (m≠n). Используется цикл с предусловием, в который вложена операция ветвления

m=18 n=12

m=6

n=6

НОД=6

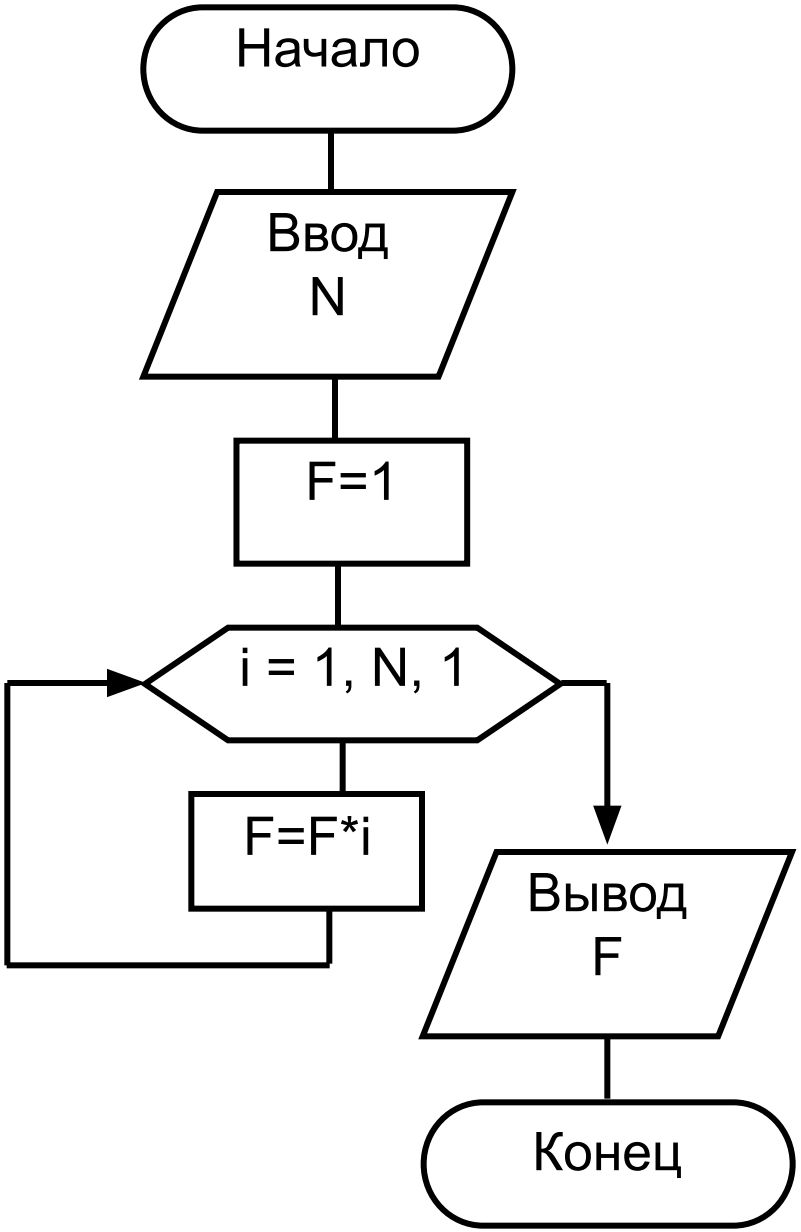


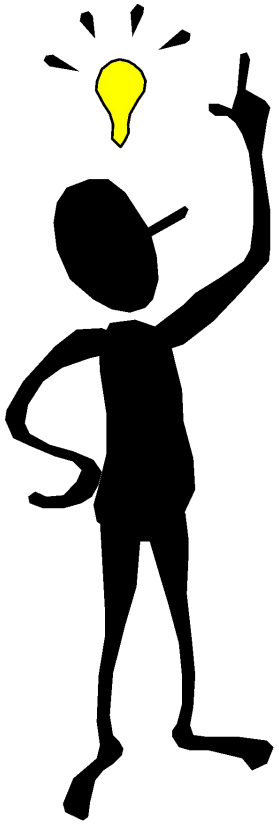
Пример. Вычислить факториал F натурального числа N ($N!=1 \cdot 2 \cdot 3 \ldots \cdot N$). Используется цикл со счетчиком i.

N=4

F=1

i=1 F=1*1=1
i=2 F=1*2=2
i=3 F=2*3=6
i=4 F=6*4=24





Правило произведения:

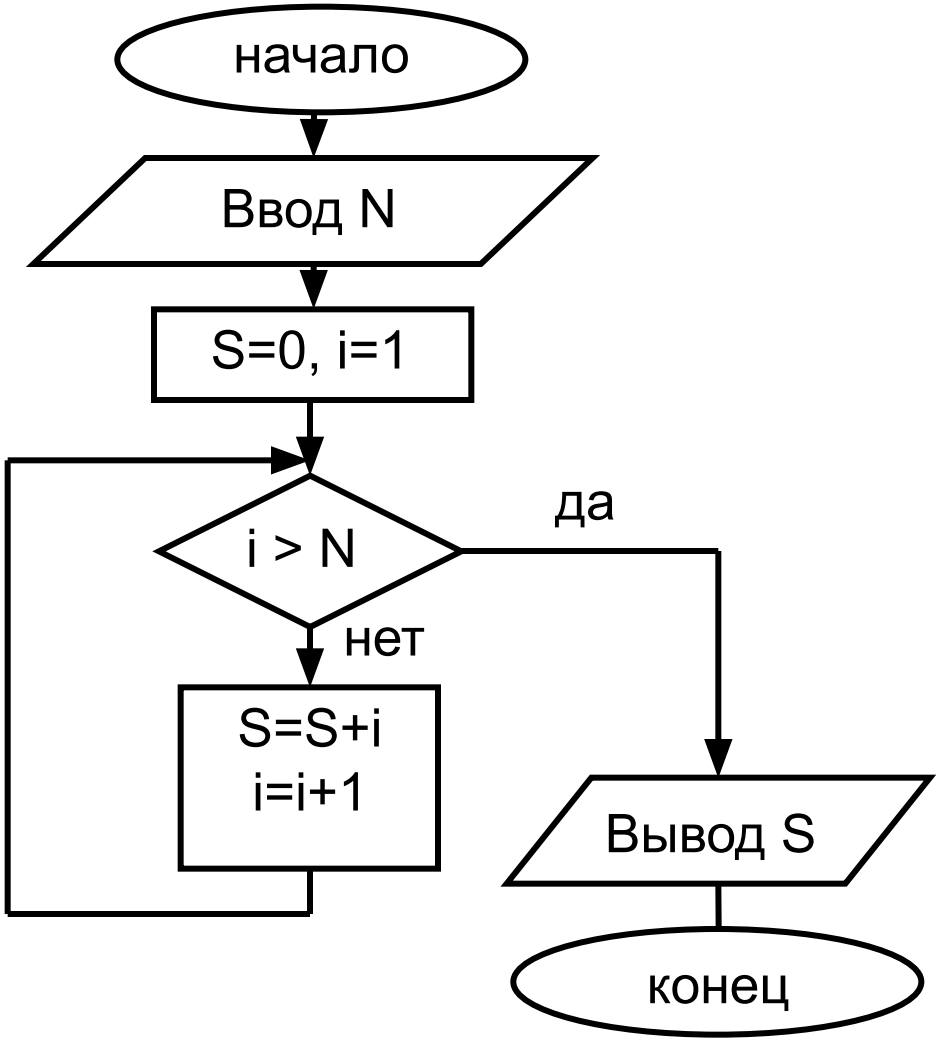
- начальное значение произведения P=1;
- в теле некоторой циклической конструкции выполнить команду:
  P = P * <множитель>

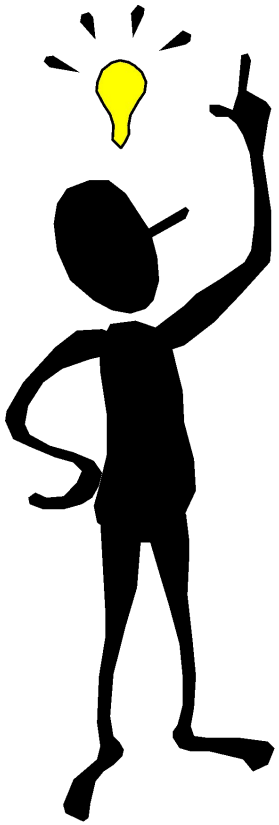
Пример. Составим алгоритм вычисления суммы N первых натуральных чисел. Используется цикл с предусловием.

N=5

S=0	i=1
S=0+1=1	i=2
S=1+2=3	i=3
S=3+3=6	i=4
S=6+4=10	i=5
S=10+5=15	i=6

S=15

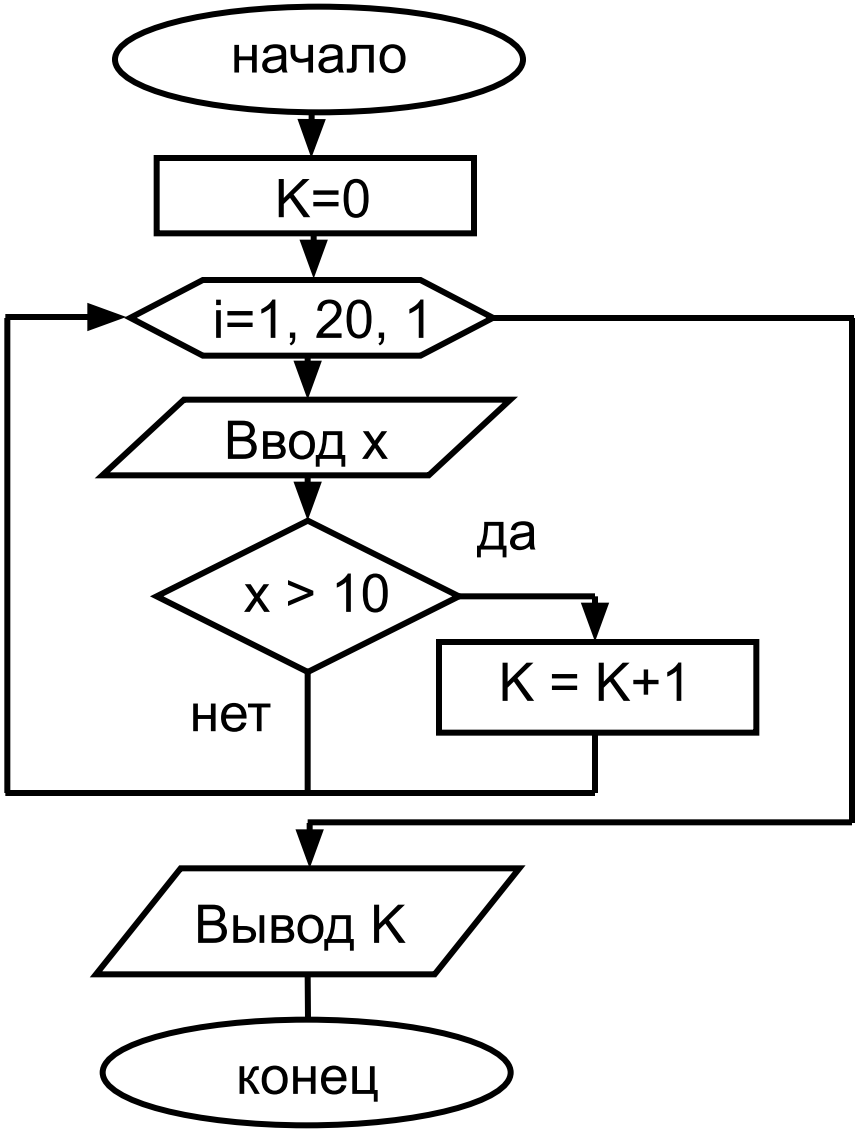




Правило суммирования:

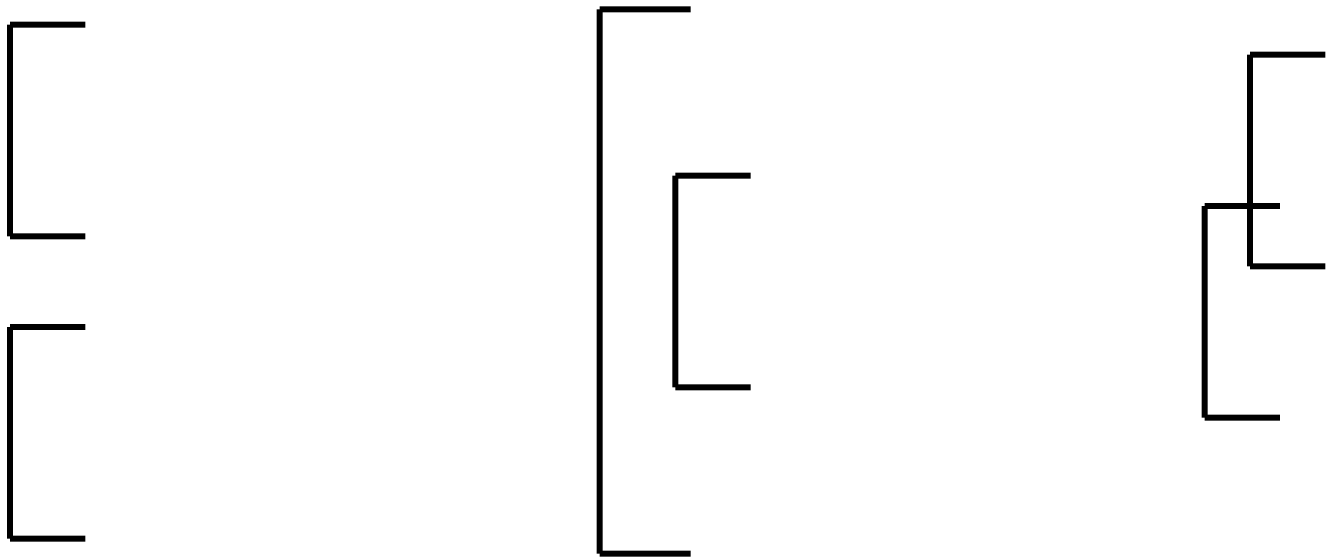
- начальное значение суммы S=0;
- в теле некоторой циклической конструкции выполнить команду:
  S = S + <слагаемое>

Пример. Задано 20 чисел. Сколько среди них чисел, больших 10?



Правило счетчика:

- начальное значение счетчика K=0;
- в теле некоторой циклической конструкции выполнить команду:
  K = K + 1



а
последовательные

б
вложенные

в
запрещенные

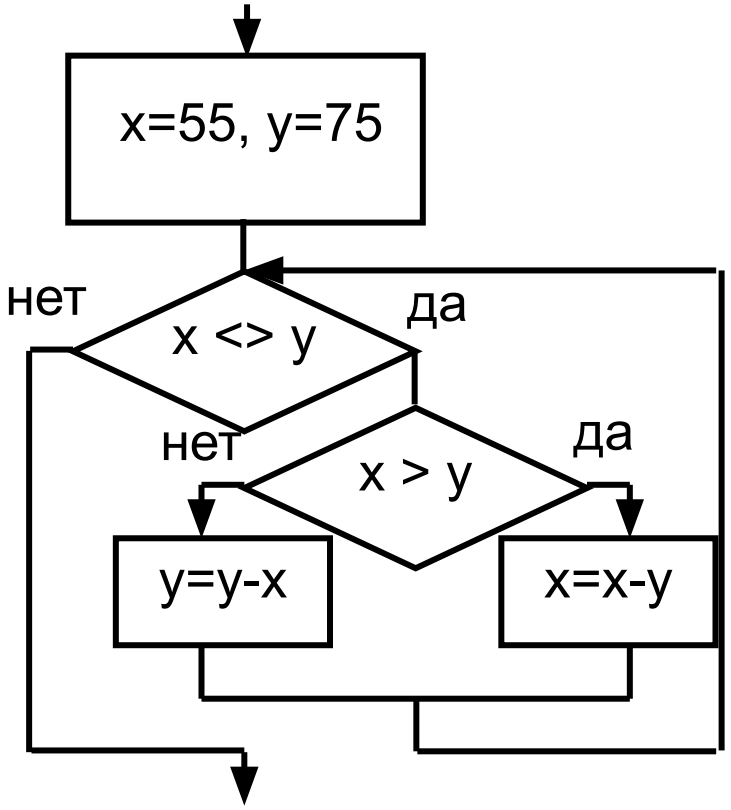
Рис. Расположение циклов

Алгоритм любой задачи может быть представлен как комбинация представленных выше элементарных алгоритмических структур, поэтому данные конструкции: линейную, ветвящуюся и циклическую, называют ***базовыми***.

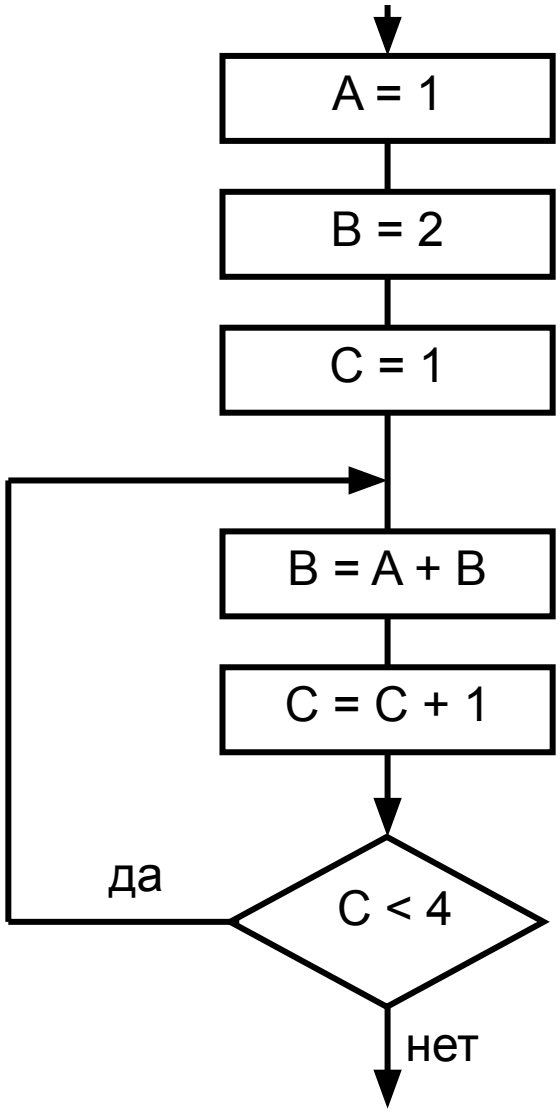
Рекурсивным называется алгоритм, организованный таким образом, что в процессе выполнения команд на каком-либо шаге он прямо или косвенно *обращается сам к себе*.

Задания для самостоятельной работы

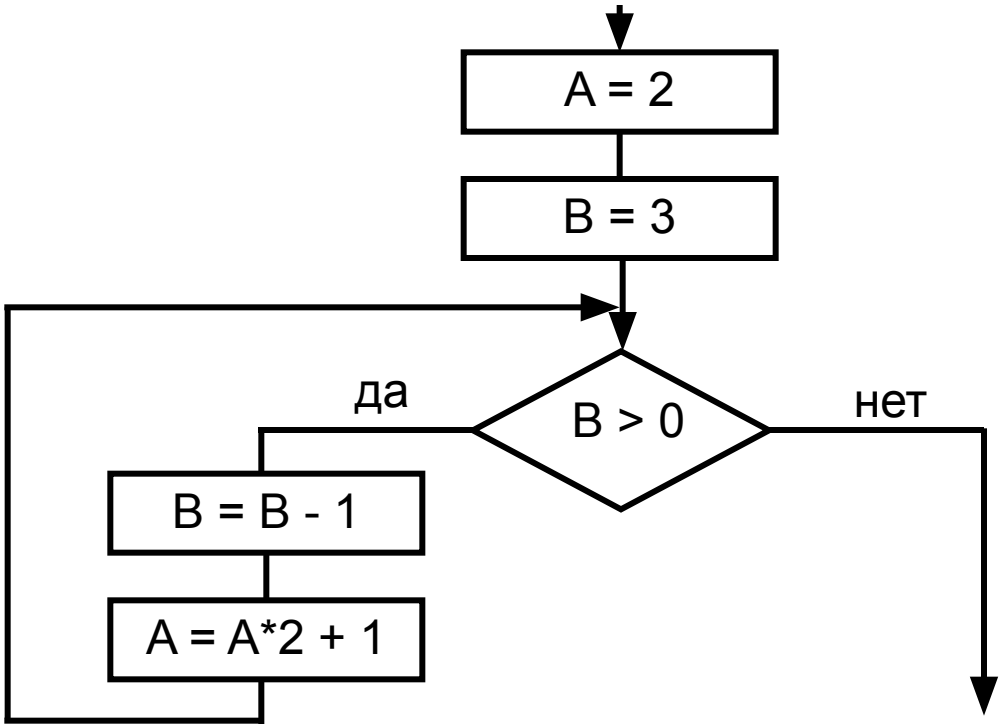
1) Определите значение целочисленной переменной x после выполнения следующего фрагмента алгоритма:



2) Определите значение переменной B :



3) Определите значение переменной А :



5. Простые и структурированные типы данных

Простые типы данных: целые и вещественные числа, символы и логические величины.

Переменная - это именованный объект (ячейка памяти), который может изменять свое значение.

Тип переменной задает:

- используемый способ записи информации в ячейке памяти;
- необходимый объем памяти для ее хранения.

Если переменные присутствуют в программе, на протяжении всего времени ее работы — их называют *статическими.*

Переменные, создающиеся и уничтожающиеся на разных этапах выполнения программы, называют *динамическими.*

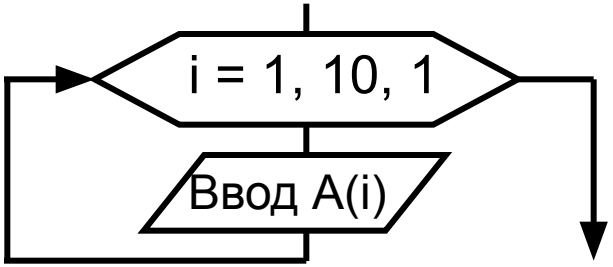
Данные, значения которых не изменяются на протяжении работы программы, называют *константами*.

Тип данных, позволяющий хранить вместе под одним именем несколько переменных, называется **структурированным**.

Массив - упорядоченная совокупность однотипных величин, имеющих общее имя, элементы которой адресуются (различаются) порядковыми номерами (индексами).

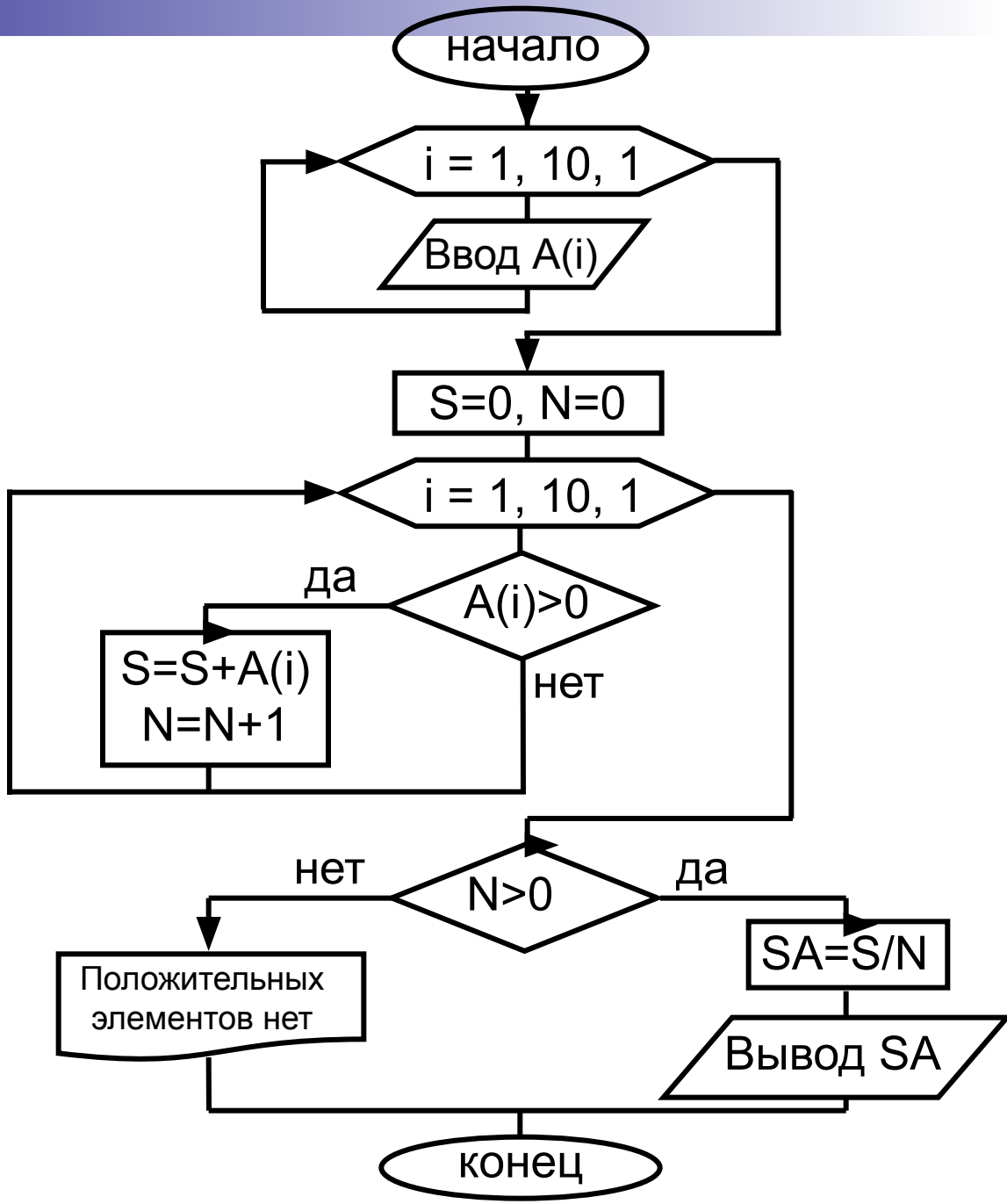
Количество элементов массива называют *размерностью.*

Блок-схема алгоритма ввода элементов массива A(10)

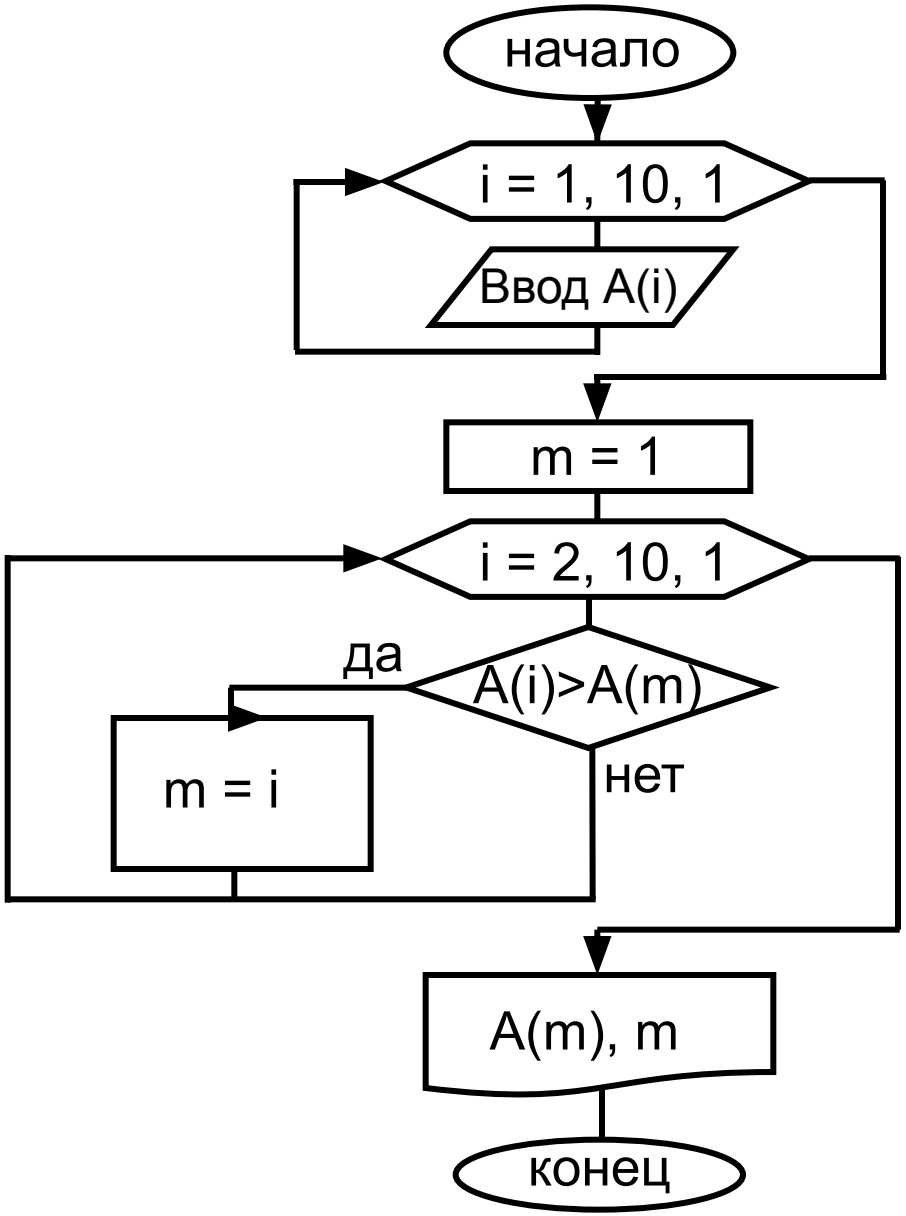


Пример.

Вычислить среднее арифметическое положительных элементов массива A(10).



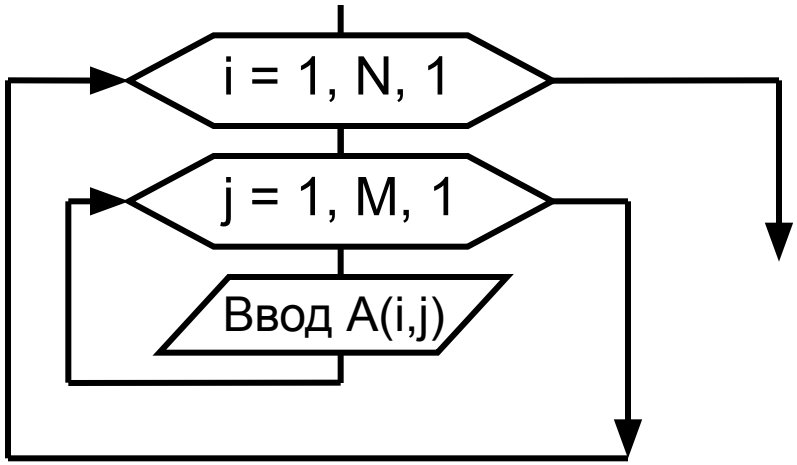
Пример. В массиве A(10) найти наибольший элемент и его индекс.



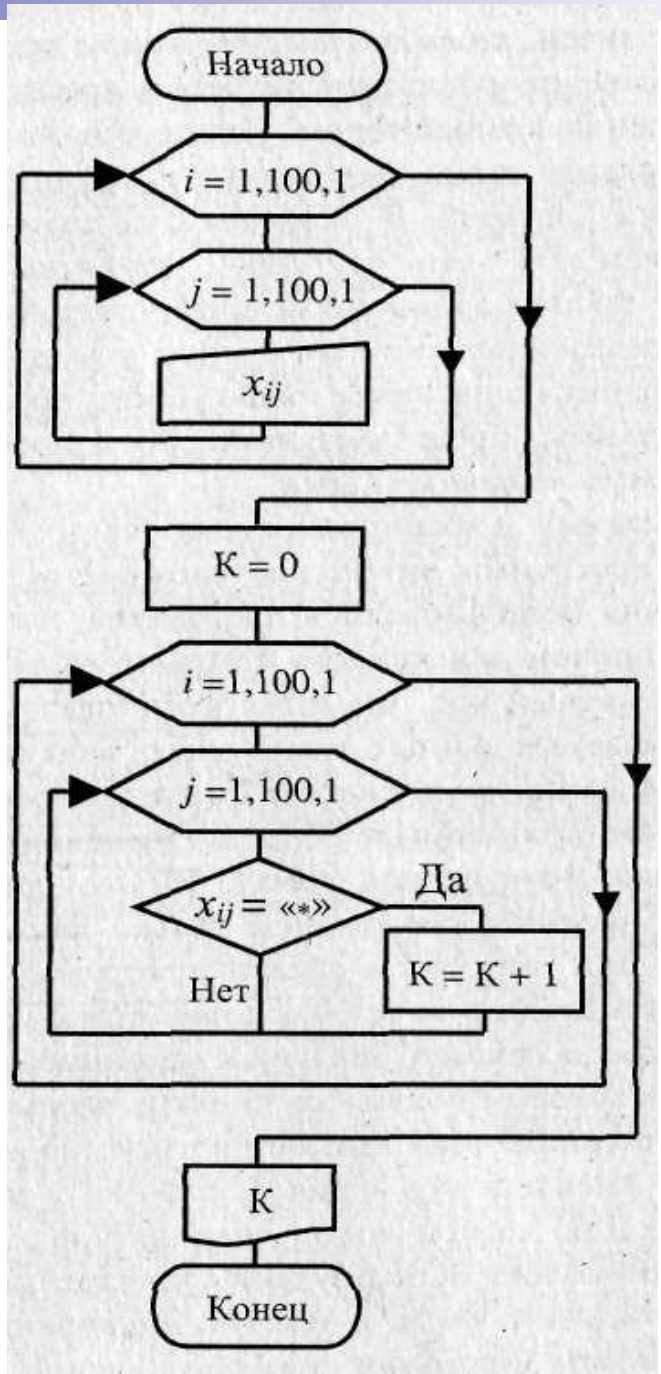
Двумерный массив характеризуется двумя размерностями N и M, определяющими число строк и столбцов соответственно.

$$
\begin{array}{r}
\text{столбец } j \\
\text{строка } i
\end{array}
\begin{pmatrix}
a_{11} & a_{12} & a_{13} & \cdots & a_{1M} \\
a_{21} & a_{22} & a_{23} & \cdots & a_{2M} \\
\cdots & \cdots & \cdots & \cdots & \cdots \\
 & & & a_{ij} & \\
\cdots & \cdots & \cdots & \cdots & \cdots \\
a_{N1} & a_{N3} & a_{N3} & \cdots & a_{NM}
\end{pmatrix}
$$

Алгоритм ввода матрицы *A(N×M).*



Пример. Задана матрица символов X (100x100), представляющая собой карту ночного неба; звездам на карте соответствуют символы «*». Определить: сколько звезд на карте?



6. Создание программ

Программа - это описание алгоритма и данных на некотором языке программирования, предназначенное для последующего автоматического выполнения.

Программирование - это

1) раздел информатики, изучающий методы и приемы составления программ для компьютеров;

2) теоретическая и практическая деятельность, связанная с созданием программ.

Свойства программ:

- *Выполнимость* - возможность выполнения программы на данном типе компьютеров.
- *Мобильность* - возможность переноса программы на другой тип компьютеров.
- *Правильность* программы - правильность результатов, получаемых с помощью данной программы.
- *Эффективность* - минимум времени выполнения, минимум машинной памяти и других ресурсов компьютера.

Язык программирования — это система обозначений, служащая для точного описания программ или алгоритмов для ЭВМ.

Основные требования, предъявляемые к языкам программирования:

- *наглядность*;
- *единство*;
- *гибкость*;
- *модульность*;
- *однозначность*.

Алфавит - это фиксированный для данного языка набор основных символов («букв алфавита»), из которых должен состоять любой текст, написанный на нем (никакие другие символы в тексте не допускаются).

Синтаксис - система правил, определяющих допустимые конструкции языка программирования из букв алфавита.

Семантика определяет смысловое значение предложений языка.

В зависимости от детализации предписаний определяют **уровень** языка программирования (чем меньше детализация, тем выше уровень). По данному критерию различают следующие языки программирования:

- машинные (самого низкого уровня);
- машинно-ориентированные (ассемблеры);
- машинно-независимые (высокого уровня).

Машинные и машинно-ориентированные языки требуют подробного описания самых мелких деталей процесса обработки данных.

Язык ассемблера — это машинно-зависимый язык низкого уровня, в котором отдельным машинным командам соответствуют мнемонические (легко запоминаемые) имена, записываемые в текстовом виде.

Языки *высокого уровня* более разнообразны и интуитивно понятны для человека. Преимущества:

- универсальность - программа, написанная на таком языке, может выполняться на разных машинах;
- независимость от аппаратуры, т.к. языки высокого уровня в значительной мере являются *машинно-независимыми*.

Перевод программы с алгоритмического языка на машинный осуществляется с помощью специальной программы — **транслятора**.

Существуют два типа трансляторов: компиляторы и интерпретаторы.

В реально функционирующих системах программирования используются обе технологии — компиляции и интерпретации.

- *Интерпретатор* берет очередной оператор языка из текста программы, анализирует его структуру и затем сразу исполняет.
- *Компиляторы*, напротив, полностью обрабатывают весь текст программы, прежде чем запускать ее на исполнение.

Рис. Процесс создания программы, готовой к исполнению

Состав **системы программирования**:

- Текстовый редактор (необходимый для создания и редактирования исходного кода программы на языке программирования)
- Компилятор
- Редактор связей
- Отладчик (позволяет анализировать работу программы во время ее выполнения, выполнять программу по шагам)
- Библиотеки функций (готовые подпрограммы, реализующие стандартные функции - математические, логические и т.п.)
- Справочная система.

Среды быстрого проектирования (Rapid Application Development, RAD-среды) используют визуальный подход.

Наибольшую популярность приобрели: для языка Basic - Microsoft Visual Basic; Pascal - Borland Delphi; C++ - Borland C++Builder; Java - Symantec Cafe.

К системам проектирования, использующим визуальные средства разработки, можно отнести также AutoCAD, системы лабораторных исследований Lab View, MATLAB, математический пакет Maple.

CASE-технологии (Computer Aided Software Engineering – автоматизированное проектирование и создание программ) - это метод проектирования информационных систем, позволяющий в наглядной форме моделировать предметную область, анализировать эту модель на всех этапах разработки и сопровождения ИС и разрабатывать приложения в соответствии с информационными потребностями пользователей.

CASE-технологии предоставляют специальные графические средства (диаграммы) для изображения различного рода моделей.

7. Основные понятия языка программирования Basic

Метка – это произвольное обозначение, которое начинается с латинской буквы и заканчивается двоеточием, например: a1:, BC400:, vivod:

Команды, записываемые в одной строке, отделяются двоеточием, например:

A = 5 : B$ = «Символ T» : rm = 1+exp(5.2)

Любую команду в программе можно снабдить поясняющим текстом – **комментарием**, который можно записать двумя способами:

1) С помощью оператора REM:

REM произвольный_текст

2) С помощью апострофа’:

’ произвольный_текст

Алфавит языка BASIC включает:

- все латинские прописные и строчные буквы;
- арабские цифры (0-9);
- служебные знаки.

Имя переменной – это произвольный набор символов (от 1 до 40), причем первый символ должен быть латинской буквой, а остальные – латинскими буквами или цифрами.

BASIC различает пять **типов данных** и определят их по *суффиксу*, т.е. по последнему символу в имени переменной.

% - целое число (от –32768 до +32767) – занимает в памяти 2 байта;

& - длинное целое число (от –2147483648 до +2147483647) – 4 байта;

! – вещественное число обычной точности (по модулю может достигать 3.402823E+38) – 4 байта;

- вещественное число двойной точности (по модулю могут достигать 1.7977…D+308) – 8 байта;

$ - строка символов (длина строки символов: от 0 до 32767).

Например:

RAS% = 5 : t2$ = “Windows”

Если суффикс опущен, по умолчанию считается, что тип этой переменной – вещественное число обычной точности.

DEFINT I-L - все переменные, имена которых начинаются с букв, лежащих в указанном диапазоне (т.е. с I, J, K, L), будут считаться целыми (INTEGER).

Общий формат команды описания типа:

DEFINT X-Y (целые числа, INTeger)

DEFLNG X-Y (длинные целые числа, LoNG)

DEFSNG X-Y (вещественные числа обычной точности, SiNGle)

DEFDBL X-Y (вещественные числа двойной точности, DouBLe)

DEFSTR X-Y (строки символов, STRing)

Значения констант записываются явно – числом или строкой символов, или с помощью специального оператора:

CONST имя_ константы = значение

Например:

a% = 1.2 : z$ = “Москва – Париж” или CONST P3 = 60, n% = 12

Арифметические выражения – это выражения, которые содержат числа в явном виде, переменные, константы, функции, а также знаки арифметических действий (+, -, *, /, ^). Значением арифметического выражения является число.

В QBASIC употребляются еще два знака арифметических операций:

\ - целочисленное деление (дробная часть отбрасывается);

MOD – вычисление остатка от деления.

Математические функции

Название	*Математический вид*	*Basic*
Синус	Sin x	SIN (x)
Косинус	Cos x	COS (x)
Тангенс	tg x	TAN (x)
Арктангенс	arctg x	ATN (x)
Логарифм натуральный	ln x	LOG (x)
Абсолютное значение	|x|	ABS (x)
Корень квадратный	$\sqrt{x}$	SQR (x)
Экспонента	e^x	EXP (x)

- FIX (арифм_выражение) – возвращает целую часть арифм_выражения.
- INT (арифм_выражение) – возвращает наибольшее целое, которое меньше или равно значению арифм_выражения.
- CINT (арифм_выражение) – округляет значение арифм_выражения по правилам арифметики.

Например:

PRINT FIX(24.8); INT(24.8); CINT(24.8)

24 24 25

PRINT FIX(-24.3); INT(-24.3); CINT(-24.3)

-24 -25 -24

Условные выражения – выражения, содержащие числа, переменные, функции, строки символов, а также знаки: = ; < > ; >; <; >=; <=.

Условное выражение принимает логическое значение: TRUE (истина) или FALSE (ложь).

Логические выражения – состоят из условных выражений, которые соединяются между собой знаками логических операций

- AND («и», конъюнкция)
- OR («или», дизъюнкция)
- NOT (отрицание)

Например, NOT (a > b)

Логическое выражение принимает логическое значение: TRUE (истина) или FALSE (ложь).

Символьные выражения содержат строки символов.

Конкатенация - соединение строк. Знак этой операции – «+» (плюс), а результат операции – новая строка символов.

Например, значением выражения «Оболочка» + «Windows» является строка символов «Оболочка Windows».

Сравнение строк ведется по кодам символов, входящих в сравниваемые строки.

Например: "DOG" > "CAT", "M16 ">"M16“

Операция присваивания:

Имя_переменной = выражение

Например:

X = 100

text$ = “Basic”

Ввод данных с клавиатуры:

INPUT [“строка_подсказка” ;] список переменных

Строка_подсказка – произвольный текст, который выдается на экран, начиная с текущей позиции курсора. В списке переменных через запятую указываются имена переменных, которые принимают вводимые данные.

Например:

INPUT “Введите коэффициент b и код режима”; b!, kr$

На экран выводится:

Введите коэффициент b и код режима? _

Объявление *блока данных:*

DATA список_констант

В списке_констант через запятую указываются значения констант из вашего набора, например:

10 DATA 5, 25, 12.3, 56, “ABC”

20 DATA 7, “BASIC”, “WINDOWS”, 78.3

Читать из блока данных:

READ список_переменных

В списке_переменных через запятую указываются имена переменных, которым присваиваются значения констант из блока данных.

Например:

READ a%, p%, c!

a%=5, p%=25, c!=12.3

Восстановление указателя блока данных:

RESTORE [номер_строки или метка]

где номер_строки (метка) должен указывать на оператор DATA.

Например:

```
RESTORE 20,
READ a%, zt$, kl$
```

a% = 7, zt$ = “BASIC” , kl$ = “WINDOWS”.

Вывод данных на дисплей:

PRINT список_выражений

В списке_выражений перечисляются выражения, значения которых надо вывести на дисплей.

Например, команда

PRINT “мне”; k%; “лет”

выведет на экран: мне 16 лет (если значение k% равно 16)

Если в качестве разделителя в списке_выражений указана запятая, QBASIC выводит данные по зонам, каждая зона – 14 позиций.

Безусловный переход:

GOTO номер_строки или метка

Например:

10 GOTO m1

. . .

m1: PRINT “Решение получено”

Условный переход (ветвление)

IF… THEN… ELSE… (ЕСЛИ… ТО… ИНАЧЕ…)

Блочный формат

IF условие_1 THEN

блок команд_1

[ELSEIF условие_2 THEN

блок команд_2

. . .

[ELSE

блок команд_n]…]

END IF

Линейный формат

IF условие THEN блок_1 [ELSE блок_2]

Цикл со счетчиком FOR…NEXT (для…следующий)

FOR счетчик = начало TO конец [STEP шаг]
блок_команд
NEXT счетчик

<u>Пример</u>

Подсчитаем сумму четных чисел в промежутке от 1 до 50.

```
S = 0 ‘                начальное значение суммы
FOR i = 2 TO 50 STEP 2
   S = S + i
NEXT i
PRINT  “Сумма четных чисел S =”; S
END
```

С помощью FOR…NEXT можно организовать **вложенные** циклы – каждый со своим FOR, NEXT и счетчиком:

```
FOR i = …
    FOR j = …
        …
    NEXT j
NEXT i
```

Из цикла FOR…NEXT с помощью оператора **EXIT FOR** можно выйти “досрочно”. Управление передается команде, следующей за NEXT.

Цикл WHILE…WEND (пока…конец).

WHILE условие

блок _команд

WEND

<u>Пример</u>

Подсчитаем сумму четных чисел в промежутке от 1 до 50.

```
S = 0 ’                 начальное значение суммы
i = 2 ’                 первое четное число
WHILE  i < = 50
   S = S + i
   i = i + 2
WEND
PRINT  “Сумма четных чисел S =”; S
END
```

Универсальный цикл DO…LOOP (делать…цикл).

с предусловием:

DO WHILE условие
блок_команд
LOOP

DO UNTIL условие
блок_команд
LOOP

с постусловием:

DO
блок_команд
LOOP WHILE условие

DO
блок_команд
LOOP UNTIL условие

Пример. Подсчитаем сумму четных чисел в промежутке от 1 до 50.

1)
```
S = 0 : i = 2
DO WHILE  i < = 50
    S = S + i
    i = i + 2
LOOP
PRINT  “S =”; S
```

2)
```
S = 0 : i = 2
DO UNTIL  i > 50
    S = S + i
    i = i + 2
LOOP
PRINT  “S =”; S
```

3)
```
S = 0 : i = 2
DO
   S = S + i
   i = i + 2
LOOP WHILE  i < = 50
PRINT  “S =”; S
```

4)
```
S = 0 : i = 2
DO
   S = S + i
   i = i + 2
LOOP UNTIL  i > 50
PRINT  “S =”; S
```

Описание массива:

DIM имя (размер) [AS тип]

Например:

DIM A(10) AS Integer, B(5) AS String

Пусть дан числовой массив из N элементов. Надо отсортировать его по возрастанию.

Сортировка методом "пузырька". Сравниваем элементы массива попарно и, в случае, если они расположены не по порядку, меняем их местами. В результате максимальное число после каждого шага сортировки как бы всплывает в конец массива, на свое заслуженное место.

Рассмотрим массив A = {5, 4, 3, 2, 1}
Отсортируем его по возрастанию

5 4 3 2 1
4 5 3 2 1
4 3 5 2 1
4 3 2 5 1
4 3 2 1 **5**

4 3 2 1 5
3 4 2 1 5
3 2 4 1 5
3 2 1 **4** **5**

3 2 1 4 5
2 3 1 4 5
2 1 **3** **4** **5**

2 1 3 4 5
1 **2** **3** **4** **5**

```
INPUT n
DIM A(n)
FOR i=1 to n        ‘ввод элементов массива
   A(i)=INT(20*RND)-10      ' A(i)=INT((max-min)*RND)+min
   PRINT A(i)
NEXT i
FOR i=1 to n-1      ‘сортировка
  FOR j=i+1 to n
       IF a(i)>a(j) THEN SWAP a(i), a(j)
  NEXT j
NEXT i
FOR i=1 to n    ‘вывод элементов массива
   PRINT A(i)
NEXT I
END
```

$$\begin{matrix} \mathbf{a_{11}} & a_{12} & a_{13} & a_{14} & \mathbf{a_{15}} \\ a_{21} & \mathbf{a_{22}} & a_{23} & \mathbf{a_{24}} & a_{25} \\ a_{31} & a_{32} & \mathbf{a_{33}} & a_{34} & a_{35} \\ a_{41} & \mathbf{a_{42}} & a_{43} & \mathbf{a_{44}} & a_{45} \\ \mathbf{a_{51}} & a_{52} & a_{53} & a_{54} & \mathbf{a_{55}} \end{matrix}$$

$i+j=n+1$

правило элементов побочной диагонали

$i=j$

правило элементов главной диагонали

<u>Пример.</u>

Вывести на печать часть двумерного массива:

$$\begin{matrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} \\ & a_{22} & a_{23} & a_{24} & \\ & & a_{33} & & \end{matrix}$$

```
INPUT n
DIM A(n,n)
FOR i=1 to n
  FOR j=1 to n
      A(i,j)=INT(20*RND)-10
     PRINT A(i,j);
  NEXT j
  PRINT
NEXT i
FOR i=1 to n
  FOR j=1 to n
     IF i<=j and i+j<=n+1 THEN PRINT A(i, j); ELSE PRINT “   “;
   NEXT j
   PRINT
NEXT i
```

Функции для работы с *символьными данными:*

ASC (симв_выражение) - возвращает целое число – десятичный код первого символа значения симв_выражения (в соответствии с кодовой таблицей символов).
Например: PRINT ASC (“Q”) ‘На экране: 81

CHR$ (код_символа) - возвращает символ, соответствующий заданному коду.
Например: PRINT CHR$ (65) ‘На экране: A

VAL (симв_выражение) - рассматривает значение симв_выражения как цепочку цифр и возвращает *число*, представленное этими цифрами.
Например: PRINT VAL (“+6.53”) ‘На экране: +6.53

STR$ (арифм_выражение) - возвращает строку символов, представляющую в цифрах значение арифм_выражения.
Напр.:n% = 24: PRINT “Лот №” + STR$ (n%) ‘На экране: Лот№24

LEN (симв_выражение) - возвращает длину значения симв_выражения (длину строки в символах).
Например: a% = LEN (“Текст”): PRINT a% ‘На экране: 5

Максимальная длина строки составляет 255 символов.

LEFT$ (симв_выражение, n) - возвращает строку символов из n левых символов значения симв_выражения.
Например:a$ = LEFT$ (“Париж-” + “Москва”, 5): PRINT a$
На экране: Париж

RIGHT$ (симв_выражение, n) - возвращает строку символов из n правых символов значения симв_выражения. Например:
b$ = RIGHT$ (“Токио, Япония”, 6): PRINT b$ ‘На экране: Япония

MID$ (симв_выражение, k, n) - возвращает фрагмент симв_выражения из n символов, начиная с k – го (k = 1, 2, …).
Напр.: c$ = MID$ (“Галактика”, 3, 4) PRINT c$ ‘На экране: лакт

LTRIM$ (симв_выражение) - возвращает копию строки с удаленными пробелами слева.

RTRIM$ (симв_выражение) - возвращает копию строки с удаленными пробелами справа.

SPACE$ (n) - возвращает строку пробелов длиной n (0-32767).

STRING$ (k, n) - создает и возвращает строку одинаковых символов. k - длина строки, n - десятичный код символа.

Пример. Вывод строки символов в обратном порядке.

```
INPUT “Введите любую строку символов”; c$
n = LEN (c$)
DIM a$ (n)
FOR i = 1 TO n
   a$ (i) = MID$ (c$, i, 1)
NEXT i
FOR i = n TO 1 STEP –1
   PRINT  a$ (i)
NEXT i
END
```

Графический режим работы адаптера устанавливается оператором

SCREEN N

где N – номер видеорежима (0-13).

Рекомендуемые режимы: 9 (для видеоадаптера EGA) и 12 (для VGA).

Видеорежим	Разрешение	Текст
7	320×200	40×25
9	640×350	80×25
12	640×480	80×30

Установка цветов:

COLOR C1, C2

где C1 – номер цвета символов,

C2 – номер фона.

Отдельный пиксель можно “зажечь” заданным цветом любой из двух универсальных команд:

PRESET (x, y) [, C]

PSET (x, y) [, C]

Номер цвета пикселя с возвращает оператор

POINT (x, y)

где x и y – координаты пикселя.

Координаты выводимого текста можно указать с помощью команды

LOCATE x, y

где x, y – координаты курсора.

Например:

A = 48

LOCATE 12, 44

PRINT “A=”; a; ‘ На экране: A = 48

Графические примитивы

LINE (x1, y1) – (x2, y2), C [, x]

Если параметр x опущен, оператор LINE вычерчивает **линию** цветом C с координатами x1, y1 (начало линии) и x2, y2 (конец линии).

Если указан параметр x, вычерчивается **прямоугольник**, причем x1, y1 – координаты левого верхнего угла прямоугольника, x2, y2 – координаты правого нижнего угла. При x = B вычерчивается контур прямоугольника, при x = BF – закрашенный (цветом C) прямоугольник. Например:

LINE (60, 110) – (260, 60), 14 ‘ Линия

LINE (140, 120) – (300, 220), 6, B ‘ Прямоугольник

LINE (380,60) – (580, 180), 4, BF ‘ Закрашенный прямоугольник

CIRCLE (x, y), r, C, [, f1, f2, e]

x, y - координаты центра окружности (или эллипса)

r - радиус окружности или эллипса (в точках).

Если последние три параметра опущены, цветом C рисуется **окружность**.

Если вы хотите нарисовать **дугу**, укажите **f1** и **f2** – значения углов в радианах, определяющих начало и конец дуги. Углы отсчитываются против часовой стрелки.

Если вы хотите нарисовать **эллипс**, укажите **e** – отношение вертикальной оси эллипса к горизонтальной. Например:

CIRCLE (110, 340), 60, 14 ‘окружность

CIRCLE (220, 340), 50, 6, 0, 1.57 ‘дуга

CIRCLE (390, 340), 60, 2, , , .6 ‘эллипс

PAINT (x, y), C1, C2

- закрашивает замкнутую область цветом C1.

x, y - координаты любой точки, которая находится внутри области;

C2 – цвет границы замкнутой области.

VIEW (x1,y1) – (x2,y2) [, c1, c2]

- выделяет на экране прямоугольную область, “окно”;

(x1,y1), (x2,y2) – координаты верхнего левого и правого нижнего углов.

c1 – цвет заливки; c2 – цвет контура.

WINDOW (xa, ya) – (xb, yb)

- задает в окне привычную декартову систему координат;
- (xa,ya) и (xb,yb) – предельные значения x и y в окне.

8. Классификация и обзор языков программирования

ЯЗЫКИ ПРОГРАММИРОВАНИЯ

Процедурные (императивные)	Объектно-ориентированные	Декларативные
- проблемно-ориентированные; - универсальные	- объектные; - визуальные	- функциональные; - логические

Процедурные языки при решении задачи требуют в той или иной форме явно записать порядок (процедуру) этого решения.

Проблемно-ориентированные языки:

- *Fortran (FORmula TRANslation)* - предназначен для решения научно-технических задач;
- *Cobol (COmmon Business Oriented Language - общепринятый деловой язык)* - компилируемый язык для решения экономических задач.
- *Basic (Beginners All-purpose Symbolic Instruction Code - универсальный символический код для начинающих)* - для решения небольших вычислительных задач в диалоговом режиме.

Универсальные языки:

- *PL/1 (Programming Language One)* - первый многоцелевой универсальный язык.
- *Pascal* - учебный язык программирования, создан в 1968-1971 гг. Никлаусом Виртом.
- *Ada* - предназначен для создания и длительного сопровождения больших программных систем, управления процессами в реальном масштабе времени.
- C - был разработан в лаборатории Bell для реализации операционной системы UNIX в начале 70-х гг.
- C+,
- Modula.

Декларативные языки построены на описании данных и искомого результата (определяют что надо вычислить, а не как это сделать, в отличие от процедурных).

В ***функциональном (аппликативном)*** программировании программа представляет собой последовательность описаний функций и выражения, которые необходимо вычислить.

- Лисп (LISP, LISt Processing — обработка списков) - создан в 1959 г. Джоном Маккарти, ориентирован на структуру данных в форме списка и позволяет организовать эффективную обработку больших объемов текстовой информации.
- РЕФАЛ,
- Haskell.

Логическое (реляционное) программирование основано на символической логике.

- Пролог (PROLOG, PROgramming in LOGic — программирование в терминах логики) - язык искусственного интеллекта, создан в 1973 г. французским ученым Аланом Кольмероэ
- Mercury,
- KLO.

Объектно-ориентированные языки. Основная идея:

«объект = данные + процедуры (методы)»

- *Объект* - это совокупность свойств (параметров) определенных сущностей и методов их обработки (программных средств).
- *Свойство* характеризует объект и его параметры. Все объекты наделены определенными свойствами, которые в совокупности и выделяют (определяют) его.
- *Методом* называют набор допустимых действий над объектом или его свойствами.
- *Событие* - это характеристика изменений состояния объекта.

- *Классом* называют совокупность объектов, характеризующихся общностью свойств и применяемых к ним методов обработки. Любой созданный объект становится *экземпляром класса*.
- *Полиморфизм* - возможность использования методов с одинаковыми именами для обработки данных разных типов.

Важнейшие принципы объектного подхода: инкапсуляция и наследование.

- *Инкапсуляция* – объединение данных и свойственных им процедур (методов) обработки в одном объекте.
- *Наследование* предусматривает создание новых классов на базе существующих, что дает возможность классу-потомку иметь (наследовать) все свойства класса-родителя.

- C++ - компактная компилирующая система, была разработана в начале 80-х гг. в лаборатории Bell корпорации AT&T.
- Java – язык интерпретирующего типа, был создан компанией Sun в начале 90-х годов на основе Си++.

Системы визуального программирования имеют интерфейс, позволяющий при составлении текста программы видеть те графические объекты, для которых она пишется.

- Visual Basic, Delphi, C++ Builder, Visual C++.
- VBA (Visual Basic for Application) является общей языковой платформой для приложений Microsoft Office, служит для написания *макросов* — программ, предназначенных для автоматизации выполнения многих операций.

Языки программирования баз данных

Имеют функциональное назначение.

Для обработки больших массивов информации и выборки записей по определенным признакам в начале 70-х гг. был создан структурированный язык запросов SQL (Structured Query Language).

Практически в каждой СУБД имеется свой универсальный язык, ориентированный на ее особенности. В Oracle имеется встроенный язык PL/SQL, в Informix — INFORMIX 4GL, в Adabas - Natural и т.д.

Языки программирования для компьютерных сетей (*скрипт-языки)* являются интерпретируемыми:

- *HTML* (Hyper Text Markup Language) — универсальный язык разметки гипертекста, используемый для подготовки Web-документов для сети Internet; разработан британским учёным Тимом Бернерсом-Ли в 1991—1992 г. в Швейцарии.
- *Perl -* предназначен для эффективной обработки больших текстовых файлов, разработал *в* 80-х гг. Ларри Уолл.

- *Tcl/Tk* - ориентирован на автоматизацию рутинных операций и состоит из мощных команд, выполняющих обработку нетипизированных объектов, разработал в конце 80-х гг. Джон Аустираут
- *VRML.* Язык моделирования виртуальной реальности, создан в 1994 г. Сейчас VRML вытесняется форматом X3D.

Языки моделирования

При моделировании систем применяются формальные способы их описания — *формальные нотации,* с помощью которых можно представить объекты и взаимосвязи между ними в системе. Такие системы называют CASE-системами.

9. Методы проектирования программ

1) алгоритмическое программирование;
2) структурное проектирование (проектирование на основе потоков данных);
3) объектно-ориентированное проектирование.

Алгоритмическое программирование

Текст программы представляет собой линейную последовательность операторов присваивания, цикла и условных операторов.

В основе **структурного проектирования** лежит целенаправленное структурирование задачи на отдельные составляющие.

Типичными методами структурного проектирования являются:

- структурное программирование;
- нисходящее проектирование;
- модульное программирование;
- событийно-ориентированное программирование.

Структурное программирование основано на модульной структуре программного продукта и типовых управляющих структурах алгоритмов обработки данных (линейная, ветвление, цикл). Структура программы должна отражать структуру решаемой задачи. Для этого используют *подпрограммы* — набор операторов, выполняющих нужное действие и независящих от других частей исходного кода.

Наличие подпрограмм позволяет вести разработку программ «*сверху вниз*»; такой подход называется ***нисходящим проектированием*.**

Модульное программирование базируется на использовании логически взаимосвязанной совокупности функциональных элементов (модулей).

Принципы модульного программирования во многом сходны с принципами нисходящего проектирования.

Событийное программирование является развитием идей нисходящего проектирования; оно основано на определении и постепенной детализации реакции программы на различные события.

Объектно-ориентированное программирование

Эта технология разработки программных продуктов объединяет данные и процессы их обработки в новые логические сущности - объекты, каждый из которых может наследовать характеристики (методы и данные) других объектов, обеспечивая тем самым повторное использование программного кода.

Такой принцип конструирования программ называют *восходящим программированием.*

10. Жизненный цикл программного обеспечения

Жизненный цикл программного обеспечения – это непрерывный процесс, который начинается с момента принятия решения о необходимости его создания и заканчивается в момент его полного изъятия из эксплуатации.

Этапы, характерные для решения большинства задач с помощью компьютера:

1) **Постановка задачи:** сбор информации;
формулировка условий;
определение конечных целей решения задачи;
определение формы выдачи результатов;
описание данных.

2) Этап формализации: анализ и исследование задачи (модели); анализ существующих аналогов; анализ технических и программных средств; разработка математической модели; разработка структур данных.

Моделирование – создание и исследование модели, замещающей оригинал.

Модель - это новый объект, который отражает некоторые существенные стороны изучаемого объекта.

Различают модели:

- *детерминированные* (в системах отсутствуют случайные воздействия) и *стохастические* (в системах присутствуют вероятностные воздействия);
- *предметные* (материальные) и *знаковые* (информационные);
- *статические* (описывающие систему в определенный момент времени) и *динамические* (рассматривающие поведение системы во времени).

В свою очередь, динамические модели подразделяют на *дискретные*, (в которых все события происходят по интервалам времени), и *непрерывные* (все события происходят непрерывно во времени).

Информационная модель (в общем случае) – это описание объекта моделирования.

Формализация – это замена реального объекта или процесса его формальным описанием, т.е. информационной моделью.

Формы информационных моделей: вербальные (словесные, описательные), графические, табличные, математические, формально-логические.

Уровни информационных моделей:

- *концептуальная модель*– обеспечивает интегрированное представление о предметной области (технологические карты, план производства); имеет слабо-формализованный характер
- *логическая* – формализуется из предыдущей (к.м.) путем выделения конкретной части, ее детализации и формализации
- *математическая модель* – л.м., формализующая на языке математики взаимосвязи в выделенной предметной области.

3) **Разработка алгоритма**: выбор метода решения задачи; выбор формы записи алгоритма; проектирование алгоритма.

4) **Программирование:** выбор языка программирования; уточнение способов организации данных; запись алгоритма на выбранном языке.

5) **Тестирование и отладка программы:** синтаксическая отладка; отладка семантики и логической структуры; тестовые расчеты и анализ результатов тестирования; совершенствование программы.

Когда программа закончена (готова работоспособная *альфа-версия*), она поступает на тестирование

Тестирование - проверка правильности работы программы в целом, либо ее составных частей.

Отладка - это процесс поиска и устранения ошибок (синтаксических и логических) в программе, производимый по результатам ее выполнения на компьютере.

6) **Анализ результатов решения задачи** и уточнение в случае необходимости математической модели с повторным выполнением этапов 2-5.

7) **Сопровождение программы:** ее доработка для решения конкретных задач, а также составление технической документации к решенной задаче, к математической модели, к алгоритму, к программе, к набору тестов, к использованию.

Сопровождение программы - документация и инструкция по эксплуатации программы.