

---

# **ОСНОВЫ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЯ**

---

- 1. История алгоритма**
- 2. Понятие алгоритма и его свойства**
- 3. Способы записи алгоритма**
- 4. Виды алгоритмов**
- 5. Понятие языка программирования и классификация языков.**

Многие, не сведущие в математике люди думают, что поскольку назначение аналитической машины Бэббиджа – выдавать результаты в численном виде, то природа происходящих в ней процессов должна быть арифметической и численной, а не алгебраической и аналитической. Но они ошибаются. Машина может упорядочивать и комбинировать числовые значения так же, как и буквы или любые другие символы общего характера. В сущности, при выполнении соответствующих условий она могла бы выдавать результаты и в алгебраическом виде.

- АВГУСТА АДА, графиня Лавлейс (1844)

# 1. История алгоритма

- Понятие *алгоритм (algorithm)* является основным для всей области компьютерного программирования.
- В средние века математики считали на абаках, а алгоритмики использовали «algorism» - старинное слово, означающее «выполнение арифметических действий с помощью арабских цифр».
- Вообще-то оно происходит от «algorithmi» - латинского написания имени Муххамеда аль-Хорезми (IX век), средне-векового математика, разработавшего правила арифметики многозначных чисел.
- В дальнейшем произошло слияние с корнем греческого происхождения *arithmetic*. В 18 веке латинское выражение *algorithmus infinitesimalis* использовалось для определения «способов выполнения действий с бесконечно малыми величинами, открытых Лейбницем».
- К 1950 году это слово ассоциировалось с алгоритмом Евклида, который представлял собой процесс нахождения наибольшего общего делителя.

---

## Алгоритм 1. (Алгоритм Евклида).

Даны два целых положительных числа  $a$  и  $b$ .

Требуется найти *наибольший общий делитель*, т.е. наибольшее целое положительное число, которое нацело делит оба числа  $a$  и  $b$ .

1.[Нахождение остатка.] Разделим  $a$  на  $b$ , и пусть остаток от деления будет равен  $p$  ( $0 \leq p < b$ )

2.[Сравнение с нулем.] Если  $p=0$ , то выполнение алгоритма прекращается;  $b$  – искомое значение.

3.[Замещение.] Присвоить  $a=b$ ,  $b=p$  и вернуться к шагу 1.

---

# Притча об индийском мудреце

В давние времена, один индийский мудрец оказал большую услугу своему правителю. Правитель решил отблагодарить его и предложил ему самому выбрать награду. На что мудрец ответил, что пожелал бы видеть шахматную доску, на каждой клетке которой были бы разложены зернышки пшена в следующем порядке: на первой – 2, на второй –  $2^2=4$ , на третьей –  $2^3=8$ , на четвертой  $2^4=16$ , и так далее на всех клетках.

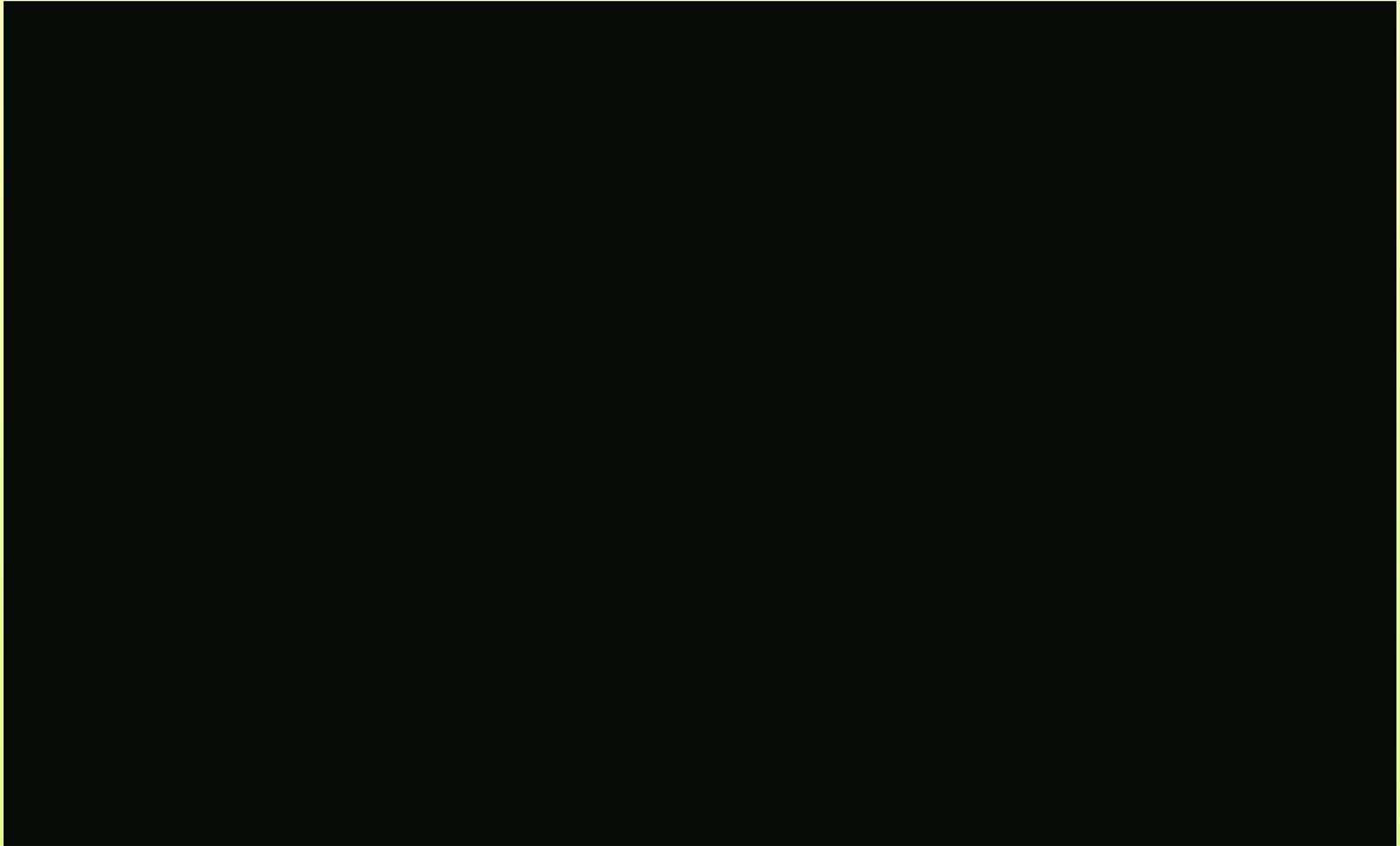
---

# Притча об индийском мудреце

Сначала правитель обрадовался легкости расплаты. Но вот выполнить обещание не смог, так как он и его слуги вряд ли когда-нибудь смогли бы отсчитать 264 зерен на последнюю клетку, что соответствует примерно 18,4 миллиардам миллиардов (!).



О том как нельзя пробить стену головой,  
нужно еще и думать!?



---

# Притча об индийском мудреце

Суть проблемы в том, что достаточно незначительно изменить входные данные, чтобы перейти от решаемой задачи к нерешаемой. Каждый человек в зависимости от своих счетных способностей может определить, начиная с какой клетки (пятнадцатой или допустим, восемнадцатой) продолжать отсчитывать зерна для него не имеет смысла.

То же самое можно определить и для компьютера.

---

## 2. Основные математические понятия

### 2.1. Числа, степени, логарифмы

- *Целые числа* -  $\dots, -3, -2, -1, 0, 1, 2, 3, \dots$
- *Рациональное число* – отношение (частное двух целых чисел),  $a/b$ , где  $b$  – положительное число.
- *Действительное число* – величина  $x$ , которая имеет десятичное представление:

$$x = n + 0.d_1d_2d_3$$

где  $n$  – целое число, а каждое  $d_i$  – любая из цифр от 0 до 9, причем в конце не должно быть бесконечной последовательности из идущих подряд девяток.

---

*Комплексное число* – это величина  $z$ , которую можно представить в виде  $z=x+iy$ , где  $x$  и  $y$  – действительные числа, а  $i$  – особая величина, удовлетворяющая условию  $i^2=-1$  ( $i$  - называется мнимой единицей). Будем называть  $x$  и  $y$  действительной и мнимой частями  $z$ , а модуль комплексного числа  $z$  определим как

$$|z| = \sqrt{x^2 + y^2}$$

Если  $u$  и  $v$  – действительные числа, для которых  $u < v$ , то замкнутым интервалом  $[u..v]$  будем называть множество действительных чисел  $x$ , таких, что  $u < x < v$ .

---

*Открытый интервал*  $(u..v)$  – это множество действительных  $x$ , таких, что  $u < x < v$ . Полуоткрытые интервалы  $[u..v)$  и  $(u..v]$  определяются аналогично. Будем допускать, что  $u$  может принимать значение  $-\square$ , а  $v$  –  $+\square$ , в этом случае соответствующая сторона интервала остается открытой и будем считать, что нижней и верхней границы у него нет. Таким образом, например, запись  $(-\square..+\square)$  обозначает множество всех действительных чисел, а запись  $[0..+\square)$  – множество неотрицательных действительных чисел.

---

■ Пусть  $b$  – положительное действительное число. Если  $n$  – целое число, то *степень*  $b^n$  определяется известными правилами

$$b^0=1; b^n=b^{n-1}b \text{ если } n>0; b^n=b^{n+1}/b \text{ если } n<0$$

$$b^{x+y}=b^x b^y; (b^x)^y=b^{xy}, \text{ где } x \text{ и } y - \text{ целые числа}$$

■ Дано положительное действительное число  $y$ . Действительное число  $x$ , при котором  $y=b^x$  (при условии  $b \neq 1$ ) называется *логарифмом  $y$  по основанию  $b$*  и записывается как  $x=\log_b y$ . Согласно этому определению имеем  $\log_b x$

$$X=b^{\log_b(b^x)}$$

■ Логарифмы с основанием  $e$  принято называть *натуральными логарифмами*, при этом используется следующая запись:  $\ln x = \log_e x$

## 2.2 Суммы и произведения

Пусть  $a_1, a_2, \dots, a_n$  – произвольная последовательность чисел. Часто возникает потребность в изучении сумм вида  $a_1 + a_2 + \dots + a_n$ . Данную сумму записывают обычно в виде:

$$\sum_{j=1}^n a_j \quad \text{или} \quad \sum_{1 \leq j \leq n} a_j$$

где  $j$  – целое число, называемое *индексной переменной*. Если  $n=0$ , то значение суммы тоже равно нулю.

Для произведения вида  $a_1 * a_2 * \dots * a_n$  существует следующее обозначение в виде формулы:

$$\text{или} \quad \prod_{j=1}^n a_j \quad \prod_{1 \leq j \leq n} a_j$$

### 3. Понятие алгоритма и его свойства

Давайте теперь дадим несколько современных трактовок определения термина «алгоритм»:

- *Алгоритм* – это система формальных правил, однозначно приводящая к решению поставленной задачи.
- **Алгоритм**- это последовательность арифметических и логических действий над данными, приводящая к получению решения поставленной задачи.
- Алгоритм – это система точно сформулированных правил, определяющая процесс преобразования допустимых исходных данных (входной информации) в желаемый результат (выходную информацию) за конечное число шагов.

## Свойства алгоритма:

- *дискретность* – разбиение процесса обработки информации на более простые этапы (шаги выполнения), т.е. алгоритм состоит из отдельных пунктов или шагов
  - *определённость (детерминированность)* - каждый шаг алгоритма должен быть строго сформулирован (иметь точный смысл), т.е. однозначность выполнения каждого отдельного шага преобразования информации;
  - *связанность* - на каждом следующем шаге используются результаты предыдущего.
-

- 
- *конечность* – алгоритм должен завершаться после конечного числа шагов
  - *результативность* – алгоритм должен приводить к получению конечных результатов
  - *массовость* – алгоритм должен быть пригоден для любых допустимых исходных данных
  - *эффективность* – применение должно давать какой-то положительный временной результат.
-

---

Каждый алгоритм имеет вход и выход. Вход алгоритма – это совокупность его *исходных данных*. Множество допустимых значений переменных на входе алгоритма называются *областью определения алгоритма*. Выход алгоритма – это совокупность результатов его работы.

---

---

## 4. Способы записи алгоритма

- а) словесно-формульный
  - б) структурная схема и алгоритм (ССА)
  - в) специальные языки (алгоритмические и псевдокоды)  
*(псевдокод - искусственный неформальный язык, обычно состоит из элементов обычного языка с элементами программирования)*
  - г) графический способ
-

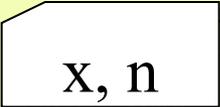
---

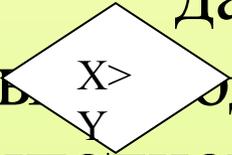
Словесная форма обычно используется для алгоритмов, ориентированных на исполнителя — человека. По словесному описанию не всегда возможна формализация процесса решения задачи. Наиболее универсальное средство представления алгоритма — это схемы алгоритмов и программ. Схема алгоритма (*блок-схема*) — это графическое представление его структуры. Оно представляет собой направленный граф, в котором этапы процесса обработки данных изображены в виде определенных геометрических фигур, соединенных линиями со стрелками.

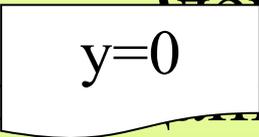
---

# Основные фигуры алгоритмов и программ

 элемент схемы, определяющий начало работы

 элемент схемы, определяющий ввод данных

 да Элемент схемы, определяющий выбор одной из альтернатив выполнения алгоритма в зависимости от условия разветвления

 элемент схемы, определяющий выходные данные

---

$X=Y+C-D$  Элемент схемы алгоритма, определяющий процесс формирования новых значений (*вычислительный блок*)

конец Элемент схемы, определяющий конец

X

Элемент схемы, определяющий процессы ввода и вывода информации

---

---

## 5. Виды алгоритмов

Различают алгоритмы *линейной, разветвляющейся и циклической структуры*, а также алгоритмы со структурой вложенных циклов. Алгоритмы решения сложных задач могут включать все перечисленные структуры, которые используются для реализации отдельных участков общего алгоритма.

---

---

■ *Алгоритм линейной структуры* – алгоритм, в котором блоки выполняются последовательно друг за другом, в порядке, заданном схемой. Такой порядок выполнения называется естественным.

■ Пример1. Вычислить высоты треугольника со сторонами  $a$ ,  $b$ ,  $c$ , используя формулы:

$$h_a = 2/a \sqrt{p(p-a)(p-b)(p-c)};$$

$$h_b = 2/b \sqrt{p(p-a)(p-b)(p-c)};$$

$$h_c = 2/c \sqrt{p(p-a)(p-b)(p-c)};$$

$$\text{где } p = (a+b+c)/2.$$

---

■ При решении данной задачи для исключения повторений следует вычислять высоты не по приведенным выше формулам непосредственно, а используя промежуточную переменную

$$t=2 \sqrt{p(p-a)(p-b)(p-c)};$$

тогда  $h_a=t/a$ ,  $h_b=t/b$ ,  $h_c=t/c$ .

■ При этом схема алгоритма решения имеет вид, представленный на рис.1:

начало

a, b, c

$p = (a+b+c)/2$

$t = 2\sqrt{p(p-a)(p-b)(p-c)}$

$h_a = t/a$

$h_b = t/b$

$h_c = t/c$

$h_a, h_b, h_c$

конец

Рис.1 Блок-схема  
линейного  
алгоритма

---

## 5.2. Алгоритмы разветвляющейся структуры.

Часто в зависимости от каких-либо промежуточных результатов вычисление осуществляется либо по одним, либо по другим формулам, т.е. в зависимости от выполнения некоторого логического условия вычислительный процесс осуществляется по одной или другой ветви. Алгоритм такого вычислительного процесса называется *алгоритмом разветвляющейся структуры (ветвлением)*.

В общем случае число ветвей в алгоритме разветвляющейся структуры **не обязательно равно двум!**

---

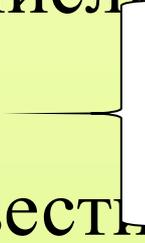
---

■ Пример 2. Вычислить значение функции  $z = x^3/y$ , где  $y = \sin(n*x) + 0,5$

■ Для удовлетворения свойства *массовости* и *результативности* алгоритма необходимо, чтобы при любых исходных данных был получен результат или сообщение о том, что задача не может быть решена при заданных данных. Действительно, если  $y=0$ , то задача не может быть решена, т.к. деление на нуль невозможно. Поэтому в алгоритме необходимо предусмотреть такое условие и выдать в качестве результата информацию о том, что  $y=0$ .

---

Таким образом, вычислительный процесс имеет две ветви. В одной ветви при  $y=0$  необходимо вычислить и отпечатать значение переменной  $u$ , в другой – вывести на печать информацию, что  $y=0$ . такой вычислительный процесс можно описать следующей условной формулой:

$z =$   вычислить  $z=x^3/y$ , если  $y \neq 0$   
вывести 'y=0', если  $y=0$ .

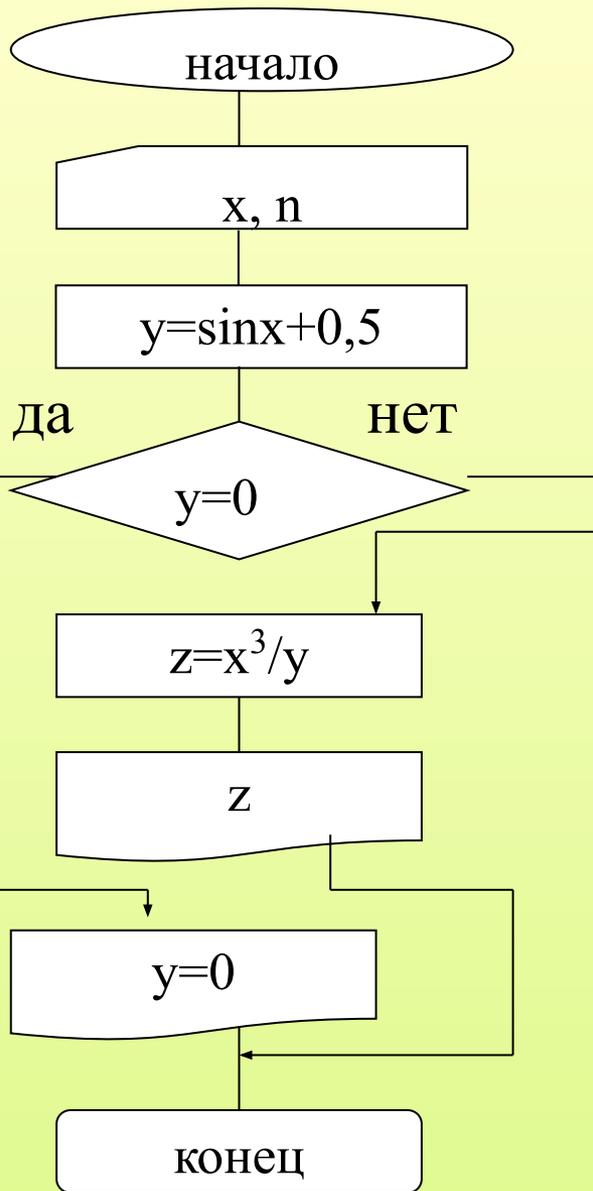


Рис.2 Блок-схема  
разветвляющегося  
алгоритма

---

## 5.3 Алгоритмы циклической структуры

- Часто при решении задач приходится многократно вычислять значения по одним и тем же математическим зависимостям для различных значений входящих в них величин. Такие многократно повторяемые участки вычислительного процесса называются *циклами*.
  - Использование циклов позволяет существенно сократить объем схемы алгоритма и длину соответствующей ей программы.
  - Различают *циклы с заданным и неизвестным числом повторений*.
-

---

Для организации цикла необходимо выполнить следующие действия:

- 1) задать перед циклом начальное значение переменной, изменяющейся в цикле;
  - 2) изменять переменную перед каждым новым повторением цикла;
  - 3) проверять условие окончания или повторения цикла;
  - 4) управлять циклом, т.е. переходить к его началу, если он не закончен, или выходить из него по окончании.
- Последние 3 функции выполняются многократно.
-

- 
- Переменная, изменяющаяся в цикле, называется *параметром цикла*. В одном цикле может быть несколько параметров.
  - Переменную, значения которой вычисляются машиной и хранятся в одной и той же ячейке памяти, называют *простой переменной*, а переменную, являющуюся элементом массива, - *переменной с индексом*. Следует иметь в виду, что параметром цикла является при использовании простой переменной сама переменная, а при использовании переменной с индексом – ее индекс.
-

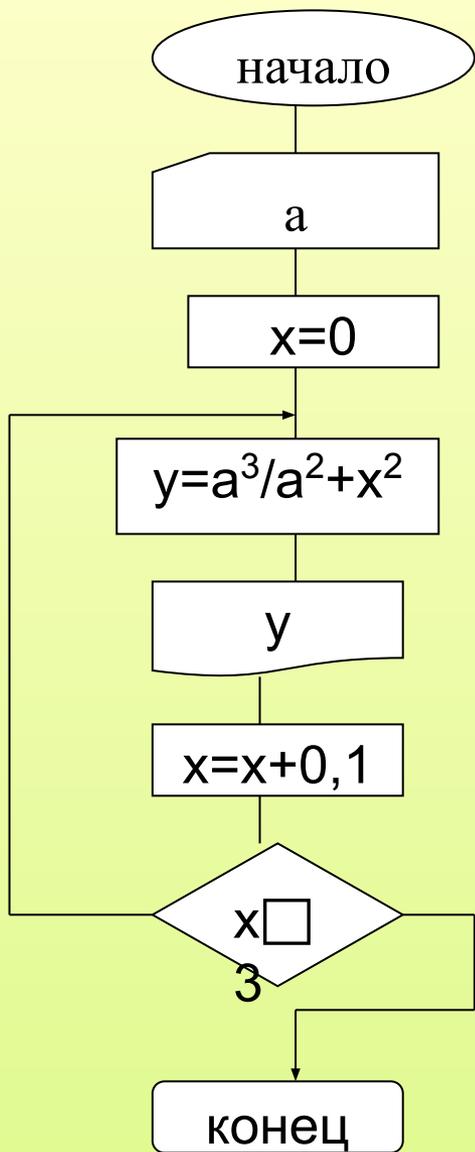
---

■ Пример 3. Вычислить и вывести на печать значения функции  $y = a^3/a^2 + x^2$  при значении  $x$ , изменяющемся от 0 до 3 с шагом 0,1.

■ Это цикл с заданным числом повторений  $n$ , вычисляемым по выражению  $n = \lfloor (x_k - x_n)/h \rfloor + 1$ , где  $x_k$  и  $x_n$  – конечное и начальное значения аргумента;  $h$  – шаг изменения аргумента; Скобки  $\lfloor \rfloor$  означают, что берется целая часть от деления.

---

3а



3б

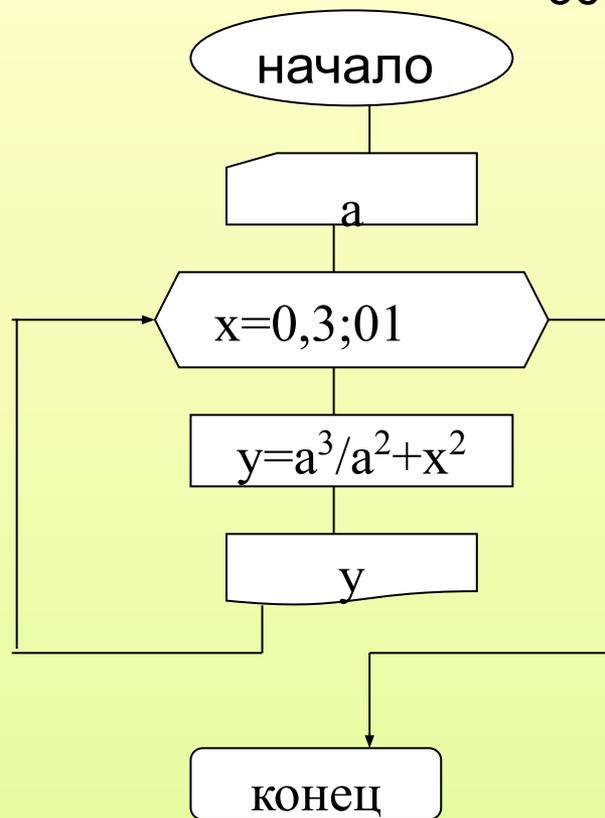


Рис.3. Блок-схема циклической структуры

## 6. Понятие языка программирования и классификация языков.

**Язык программирования** – формализованный язык для описания алгоритма решения задачи на компьютере.

Языки программирования, если в качестве признака классификации взять синтаксис образования конструкции, можно условно разделить на классы:

- ▣ Машинные языки (computer language) – языки программирования, воспринимаемые аппаратной частью компьютера (машинные коды);
- ▣ Машинно-ориентированные языки (computer-oriented language) – языки программирования, которые отражают структуру конкретного типа компьютера (ассемблеры);

- 
- ▣ **Алгоритмические языки** (algorithmic language) – не зависящие от архитектуры компьютера языки программирования для отражения структуры алгоритма (Паскаль, Фортран, Бейсик и др.);
  - ▣ **Процедурно-ориентированные языки** (procedure-oriented language) – языки программирования, где имеется возможность описания программы как совокупности процедур (подпрограмм);
  - ▣ **Проблемно-ориентированные языки** (universal programming language) – языки программирования, предназначенные для решения задач определенного класса (Лисп, РПГ, Симула и др.);
  - ▣ **Интегрированные системы программирования.**
-

---

Управлять компьютером нужно по определенному алгоритму. Описание способа решения задачи в виде конечной (по времени) последовательности действий ещё называется *формальным*. Для представления алгоритма в виде, понятном компьютеру, служат *языки программирования*. Алгоритм действий, записывается на одном из таких языков, в итоге получается *текст программы* – полное, законченное и детальное описание алгоритма на языке программирования.

---

---

Затем этот текст программы специальными служебными приложениями, которые называются *трансляторами* либо переводится в машинный код, либо исполняется.

---

---

## Компиляторы и интерпретаторы.

С помощью языка программирования создается не готовая программа, а только ее текст, описывающий ранее разработанный алгоритм. Чтобы получить работающую программу, надо этот текст либо автоматически перевести в машинный код (для этого служат программы-*компиляторы*) и затем использовать отдельно от исходного текста либо сразу выполнять команды языка, указанные в тексте программы (этим занимаются программы-*интерпретаторы*).

---

---

**Интерпретатор** берет очередной оператор языка из текста программы, анализирует его структуру и затем сразу его исполняет (обычно после анализа оператор транслируется в некоторое промежуточное представление или даже машинный код для более эффективного дальнейшего использования). Только после того как текущий оператор успешно выполнит, интерпретатор перейдет к следующему. При этом, если один и тот же оператор должен выполняться в программе многократно, интерпретатор всякий раз будет выполнять его так, как будто встретил впервые.

---

---

**Компиляторы** полностью обрабатывают весь текст программы (он иногда называется *исходный код*). Они просматривают его в поисках синтаксических ошибок, выполняют определенный смысловой анализ и затем автоматически переводят (*транслируют*) на машинный язык – генерируют машинный код. В результате законченная программа получается компактной и эффективной, работает быстрее программы, выполняемой с помощью интерпретатора.

---

## Уровни языков программирования

Разные типы процессоров имеют разные наборы команд. Если язык программирования ориентирован на конкретный тип процессора и учитывает его особенности, то он называется *языком программирования низкого уровня*. Имеется в виду, что операторы языка близки к машинному коду и ориентированы на конкретные команды процессора. С помощью языков низкого уровня создаются очень эффективные и компактные программы, т.к. разработчик получает доступ ко всем возможностям процессора. С другой стороны, при этом требуется очень хорошо понимать устройство компьютера, затрудняется отладка больших приложений, а результирующая программа не может быть перенесена на компьютер с другим типом процессора.

---

*Языки программирования высокого уровня* значительно ближе и понятнее человеку, нежели компьютеру. Особенности конкретных компьютерных архитектур в них не учитываются, поэтому создаваемые программы на уровне исходных текстов легко переносимы на другие платформы, для которых создан транслятор этого языка. Разрабатывать программы на языках высокого уровня с помощью понятных и мощных команд значительно проще, а ошибок при создании программ допускается гораздо меньше.

---

# Виды языков программирования высокого уровня

**Fortran (Фортран).** Это первый компилируемый язык, созданный Джимом Бэкусом в 50-е годы. Основным критерием при разработке компиляторов Фортрана являлась эффективность исполняемого кода. Хотя в Фортране впервые был реализован ряд важнейших понятий программирования, удобство создания программ было принесено в жертву возможности получения эффективного машинного кода. Однако для этого языка было создано огромное количество библиотек, начиная от статистических комплексов и кончая пакетами управления спутниками.

---

**Cobol (Кобол).** Это компилируемый язык для применения в экономической области и решения бизнес-задач, разработанный в начале 60-х годов. Он отличается большой «многословностью» - его операторы иногда выглядят как обычные английские фразы. Были реализованы очень мощные средства работы с большими объёмами данных, хранящимися на различных внешних носителях. На этом языке создано очень много приложений, которые активно эксплуатируются и сегодня. Достаточно сказать, что наибольшую зарплату в США получают программисты на Коболе.

---

---

**Algol (Алгол).** Компилируемый язык, созданный в 1960 году. Он был призван заменить Фортран, но из-за более сложной структуры не получил широкого распространения. В 1968 году была создана версия Алгол 68, по своим возможностям и сегодня опережающая многие языки программирования, однако из-за отсутствия достаточно эффективных компьютеров для нее не удалось своевременно создать хорошие компиляторы.

---

---

**Pascal (Паскаль).** Язык Паскаль, созданный в конце 70-х годов основоположником множества идей современного программирования Никлаусом Виртом, во многом напоминает Алгол, но в нем ужесточен ряд требований к структуре программы и имеются возможности, позволяющие успешно применять его при создании крупных проектов.

---

---

**Basic (Бейсик).** Для этого языка имеются и компиляторы, и интерпретаторы, а по популярности он занимает первое место в мире. Он создавался в 60-х годах в качестве учебного языка и очень прост в изучении.

---

---

**C (Си)**. данный язык был создан в лаборатории Bell и первоначально не рассматривался как массовый. Он планировался для замены ассемблера, чтобы иметь возможность создавать столь же эффективные и компактные программы, и в то же время не зависеть от конкретного типа процессора.

Си во многом похож на Паскаль и имеет дополнительные средства для прямой работы с памятью (*указатели*). На этом языке в 70-е годы написано множество прикладных и системных программ и ряд известных операционных систем (Unix).

---

---

**C++ (Си++).** Си++ - это объектно-ориентированное расширение языка Си, созданное Бьярном Страуструпом в 1980 году. Множество новых мощных возможностей, позволивших резко повысить производительность программистов, наложилось на унаследованную от языка Си определенную низкоуровневость, в результате чего создание сложных и надежных программ потребовало от разработчиков высокого уровня профессиональной подготовки.

---

---

**Java (Джава, Ява).** Этот язык был создан компанией Sun в начале 90-х годов на основе Си++. Он призван упростить разработку приложений за счет исключения из него всех низкоуровневых возможностей. Компиляция происходит не в машинный код, а в платформно-независимый байт-код (каждая команда занимает один байт). Этот байт-код может выполняться с помощью интерпретатора – виртуальной Java-машины JVM (Java Virtual Machine). Благодаря наличию множества Java-машин программы на Java можно переносить не только на уровне исходных текстов, но и на уровне двоичного байт-кода, поэтому по популярности язык Ява сегодня занимает второе место в мире после Бейсика.

---

---

Особое внимание в развитии языка **Java** уделяется двум направлениям: поддержке всевозможных мобильных устройств и микрокомпьютеров, встраиваемых в бытовую технику (технология Jini) и созданию платформно - независимых программных модулей, способных работать на серверах в глобальных и локальных сетях с различными операционными системами (технология Java Beans). Пока основной недостаток этого языка – невысокое быстродействие, так как язык Ява интерпретируемый.

---

---

## Языки программирования баз данных.

Эта группа языков отличается от алгоритмических языков прежде всего решаемыми задачами. *База данных* – это файл (или группа файлов), представляющий собой упорядоченный набор *записей*, имеющих единообразную структуру и организованных по единому шаблону (как правило, в табличном виде).

---

---

При работе с базами данных чаще всего требуется выполнять следующие операции:

- ❑ создание / модификация свойств / удаление таблиц в базе данных;
  - ❑ поиск, отбор, сортировка информации по запросам пользователей;
  - ❑ добавление новых записей;
  - ❑ модификация существующих записей;
  - ❑ удаление существующих записей.
-

---

Как только появилась потребность в обработке больших массивов информации и выборки групп записей по определенным признакам, для этого был создан структурированный язык запросов **SQL** (Structured Query Language). Он основан на мощной математической теории и позволяет выполнять эффективную обработку баз данных, манипулируя не отдельными записями, а *группами* записей.

---

---

Для управления большими базами данных и их эффективной обработки разработаны СУБД (Системы Управления Базами Данных).

Практически в каждой СУБД помимо поддержки языка SQL имеется также свой уникальный язык, ориентированный на особенности этой СУБД и не переносимый на другие системы.

Сегодня в мире насчитывается пять ведущих производителей СУБД: Microsoft (SQL Server), IBM (DB2), Oracle, Software AG (Adabas), Informix и Sybase.

---

---

## Языки программирования для Интернета

С активным развитием глобальной сети было создано немало реализаций популярных языков программирования, адаптированных специально для Интернета. Все они отличаются характерными особенностями: языки являются интерпретируемыми, интерпретаторы для них распространяются бесплатно, а сами программы – в исходных текстах. Такие языки называют *скрипт-языками*.

---

---

**HTML.** Общеизвестный язык для оформления документов. Он очень прост и содержит элементарные команды форматирования текста, добавления рисунков, задания шрифтов и цветов, организации ссылок и таблиц. Все Web-страницы написаны на языке HTML или используют его расширения.

**Perl.** В 80-х годах Ларри Уолл разработал язык Perl. Он задумывался как средство эффективной обработки больших текстовых файлов, генерации текстовых отчетов и управления задачами. По мощности Perl значительно превосходит языки типа Си. В него введено много часто используемых функций работы со строками, массивами, управления процессами, и др.

---

**Tcl/Tk.** Придуман в 80-х годах Джоном Аустираутом. Ориентирован на автоматизацию рутинных процессов и состоит из команд, предназначенных для работы с абстрактными нетипизированными объектами. Он независим от типа системы и при этом позволяет создавать программы с графическим интерфейсом.

**VRML.** Был создан в 1994 году для организации виртуальных трехмерных интерфейсов в Интернете. Он позволяет описывать в текстовом виде различные трехмерные сцены, освещения и тени, создавать свои миры, путешествовать по ним, «облетать» со всех сторон, вращать, масштабировать, и т.д.

# Основные аспекты языка программирования

1. Алфавит – это конечный набор неделимых символов, из которых строятся все конструкции языка. Такие неделимые символы называются литерами или буквами. Обычно алфавит языка программирования включает арабские цифры, малые и большие латинские буквы и ряд специальных символов таких как (. ; \ \* - = < >). Обычно такой набор соответствует символам типовой компьютерной клавиатуры. Поскольку алфавит – конечный набор символов, его можно определить простым перечислением.

---

---

2. **Лексика**. лексику языка программирования составляют простейшие элементы языка, имеющие самостоятельный смысл. Такие элементы называют *лексемами* языка программирования. Например, лексемами являются изображения констант, имена, переменные, т.н. *служебные слова*, имеющие фиксированный в языке смысл, например, `begin`.

3. **Синтаксис** — это набор правил, задающих структурно верные конструкции языка программирования. синтаксис обычно задается с помощью формализованных средств.

---

---

4. **Семантика** – это набор правил, сопоставляющий синтаксически корректным концепциям языка их смысл. Описывая семантику той или иной концепции, как правило, мы будем говорить о том, какие действия выполняет исполнитель алгоритма в соответствии с данной концепцией. Определяя семантику языковой концепцией, мы будем использовать естественный язык. Существуют и другие способы определения семантики, а также формальные способы определения семантики.

5. **Прагматика** – это совокупность методов, приемов решения практических задач с помощью языковых концепций.

---

## 7. Системы программирования

В самом общем случае для создания программы на выбранном языке программирования нужно иметь следующие компоненты:

1) **Текстовый редактор.** Лучше использовать специализированные редакторы, которые ориентированы на конкретный язык программирования и позволяют в процессе ввода текста выделять ключевые слова и идентификаторы разными цветами и шрифтами. Подобные редакторы созданы для всех популярных языков и дополнительно могут автоматически проверять правильность синтаксиса программы непосредственно во время ее ввода.

---

2) Исходный текст с помощью *программы-компилятора* переводится в машинный код. Если обнаружены синтаксические ошибки, то результирующий код создан не будет.

На этом этапе уже возможно получение готовой программы, но чаще всего в ней не хватает некоторых компонентов, поэтому компилятор обычно выдает промежуточный *объектный код* (двоичный файл, стандартное расширение .OBJ).

---

---

3) Исходный текст большой программы состоит, как правило, из нескольких **модулей** (файлов с исходными текстами), т.к. хранить все тексты в одном файле неудобно. Каждый модуль компилируется в отдельный файл с объектным кодом, который затем надо объединить в одно целое.

Сгенерированный код модулей и подключенные к нему стандартные функции надо не просто объединить в одно целое, а выполнить такое объединение с учетом требований операционной системы, то есть получить на выходе программу, отвечающую определенному формату.

---

---

Объектный код обрабатывается специальной программой – *редактором связей* или *сборщиком*, который выполняет связывание объектных модулей и машинного кода стандартных функций, находя их в библиотеках, и формирует на выходе работоспособное приложение – *исполнимый код* для конкретной платформы.

Если по каким-то причинам один из объектных модулей или нужная библиотека не обнаружены (например, неправильно указан каталог с библиотекой), то сборщик сообщает об ошибке и готовой программы не получается.

---

---

4) *исполнимый код* – это законченная программы, которую можно запустить на любом компьютере, где установлена операционная система, для которой эта программа создавалась. Как правило, итоговый файл имеет расширение .EXE или .COM.

---