

АЛГОРИТМ

И его
формальное
исполнение

КИБЕРНЕТИК

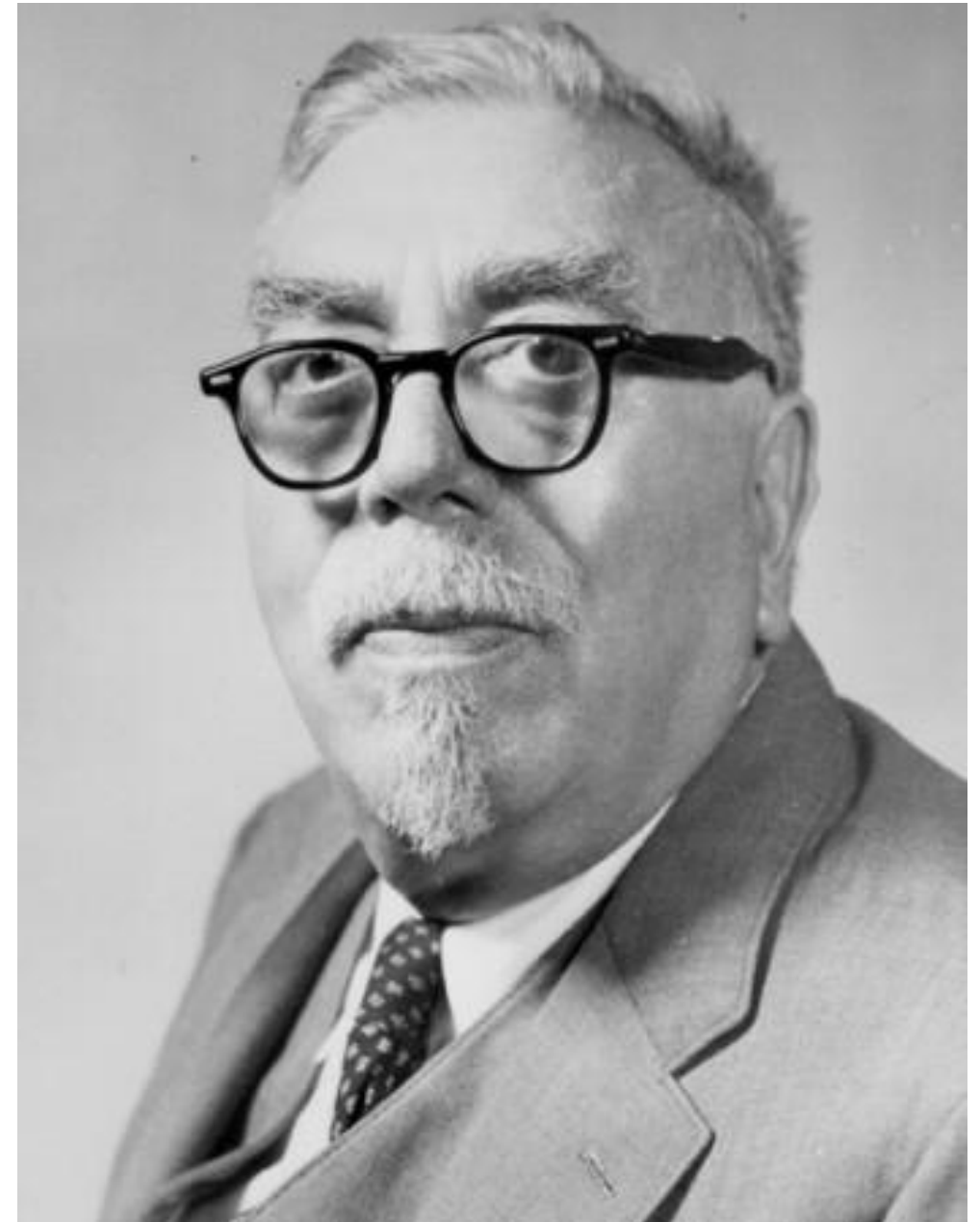
А

В 1948 г. В США и Европе вышла книга Норберта Винера «Кибернетика, или Управление и связь в животном и машине».

С этого момента и стали говорить о новой науке – кибернетике.

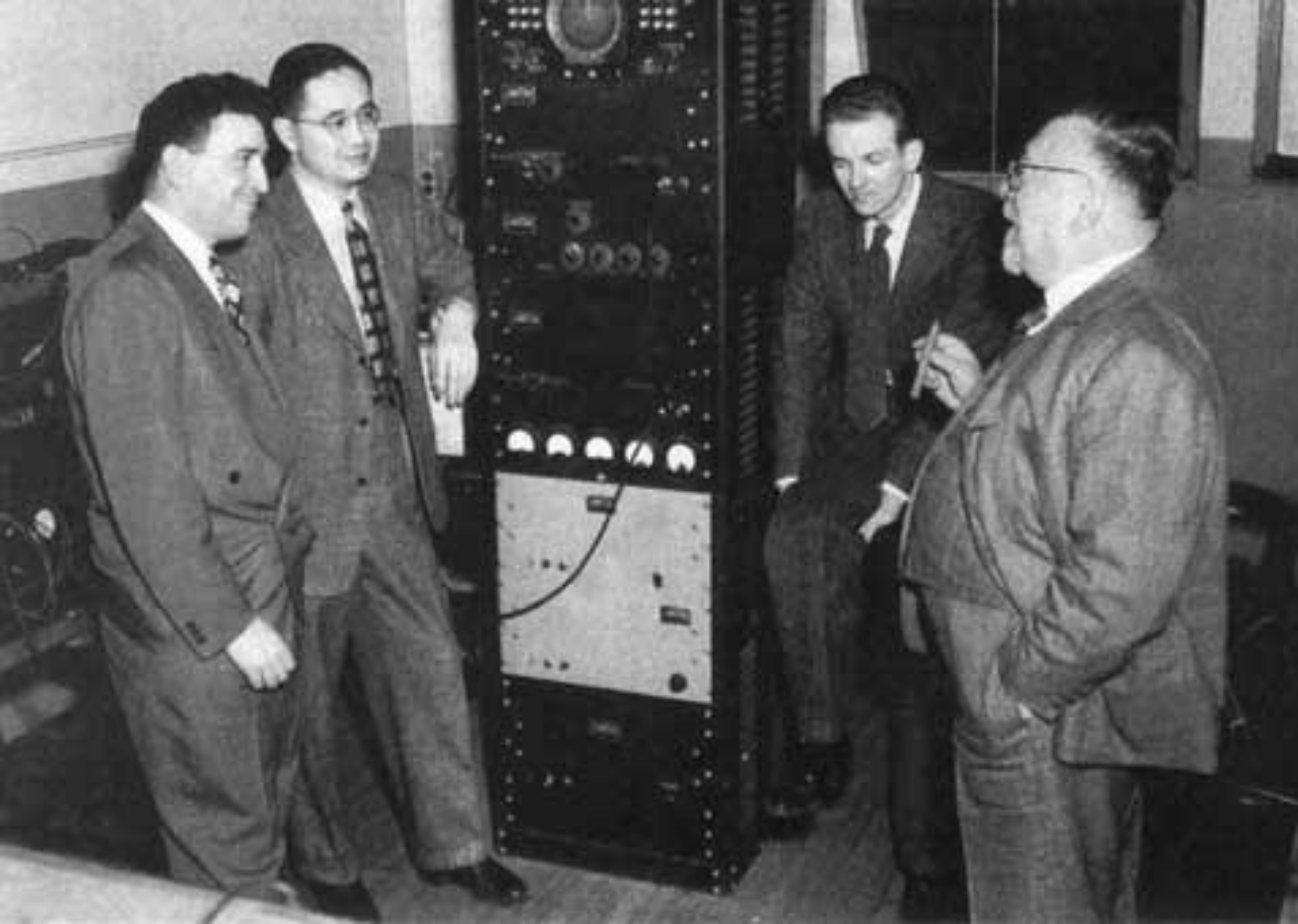
Кибернетика – наука об общих свойствах процессов управления в живых и неживых системах.

Управление – это целенаправленное воздействие одних объектов (управляющих) на другие объекты – управляемые.



Норнберт Винер (1894 – 1964

гг.)



**Норнберт
Винер
(1894 - 1964 гг.)
(справа),
Массачусетский
технологически
й институт.**

АЛГОРИТМ

Все управляющие воздействия производятся в форме команд.

Команды отдаются с определенной целью.

Последовательность команд по управлению объектом, выполнение которых приводит к достижению поставленной ранее цели называется **алгоритмом управления.**

ПРОИСХОЖДЕНИЕ СЛОВА «АЛГОРИТМ»

Слово «**алгоритм**» происходит от имени арабского учёного Мухаммед ибн Муса ал-Хорезми.

В латинском переводе книги Ал-Хорезми правила начинались словами «**Алгоризми сказал**». С течением времени люди забыли, что «Алгоризми» - это автор правил, и стали просто называть правила алгоритмами.



ПОНЯТИЕ АЛГОРИТМА

Алгоритм – это конечная система правил, сформулированная на языке исполнителя, которая определяет последовательность перехода от допустимых исходных данных к конечному результату и которая обладает определенными свойствами.

Алгоритм – это строго определенная последовательность действий при решении задачи.

Алгоритм содержит несколько шагов.

Шаг алгоритма – это каждое отдельное действие алгоритма.

Исполнитель – это объект, умеющий выполнять определенный набор действий. Исполнителем может быть человек, робот, животное, компьютер.

Система команд исполнителя (СКИ) – это все команды, которые исполнитель умеет выполнять.

Среда исполнителя – обстановка, в которой функционирует исполнитель.

КТО ИГРАЕТ РОЛЬ ИСПОЛНИТЕЛЯ И УПРАВЛЯЮЩЕГО В СЛЕДУЮЩИХ СИСТЕМАХ: ШКОЛА, САМОЛЕТ, СТАЯ ВОЛКОВ?

Систе

ма

Школ

а

Самол

ет

Стая

ВОЛКОВ

Управляющ

ий

Администрац

ия

Пило

Стюбардес

сы

Вожа

к

Исполните

ль

Коллектив,

учащиеся

Самол

Пассажир

ы

Остальные

ВОЛКИ

ЗАДАНИЕ: НАЗОВИ ИСПОЛНИТЕЛЕЙ СЛЕДУЮЩИХ ВИДОВ РАБОТЫ:

- Уборка мусора во дворе
- Обучение детей в школе
- Вождение автомобиля
- Ответ у доски
- Приготовление пицци
- Печатание документа на принтере

СВОЙСТВА АЛГОРИТМА

Необходимая задача:

Звонок по телефону... Как позвонить?

Алгоритм действий:

- 1) поднять телефонную трубку;
- 2) если услышал длинный гудок, то набрать номер, иначе выполнить п. 6 (телефон не исправен);
- 3) определить тип гудков: «вызов» или «занято». Если «вызов», перейти на п. 4, если «занято», перейти на п. 6;
- 4) дожждаться 5 вызывающих гудков;
- 5) если за это время абонент не поднял трубку, то выполнить п. 6.
- 6) Положить трубку

А если мы не закончим действие 4, и сразу будем выполнять действие 5, нам удастся дозвониться?

А если мы будем делать все действия сразу?

СВОЙСТВА АЛГОРИТМОВ

Дискретность – разбиение выполнения алгоритма на последовательность законченных действий-шагов, и каждое действие должно быть закончено исполнителем прежде, чем он приступит к исполнению следующего действия.

2 апреля 1973 года был сделан первый звонок с помощью мобильного телефона.

Мартин Купер (Martin Cooper) держит в руках беспроводной телефон Motorola DynaTAC.

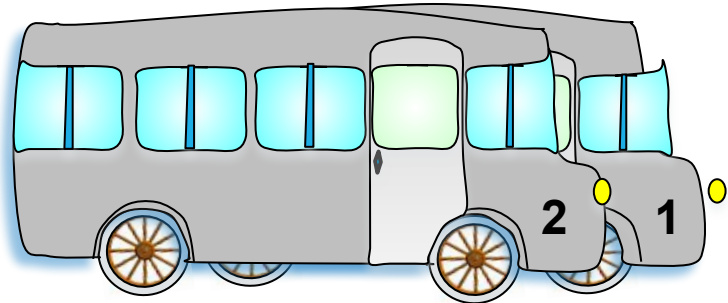


СВОЙСТВА АЛГОРИТМОВ

Необходимая задача:

Поездка на автобусе номер 2

- 1) Прийти на автобусную остановку;
- 2) Если нет автобуса, то дождаться его приезда;
- 3) Иначе, посмотреть номер маршрута;
- 4) Если номер маршрута – 2, то сесть в него;
- 5) Иначе п. 2.



СВОЙСТВА АЛГОРИТМОВ

Детерминированность – на каждом шаге однозначно определено преобразование объектов среды исполнителя, полученной на предыдущих шагах алгоритма.

Последовательность строго соблюдается.

СВОЙСТВА АЛГОРИТМОВ

Нахождение большего из двух чисел

- 1) Из числа A вычесть число B .
- 2) Если получилось отрицательное значение, то сообщить, что число B больше.
- 3) Если получилось положительное значение, то сообщить, что число A больше.

Если $(A-B) < 0$, тогда число B -
больше

Если $(A-B) > 0$, тогда число A -
больше

Результативность – исполнение алгоритма должно приводить к конкретному результату.

Это свойство требует, чтобы в алгоритме *не было ошибок*.

СВОЙСТВА АЛГОРИТМОВ

Конечность – завершение работы алгоритма за конечное число шагов.

Математика и информатика работает только с конечными объектами и процессами. Бесконечные алгоритмы (*защелкивание*) считаются ошибкой, либо не рассматриваются.

Массовость – алгоритм правильно работает на некотором множестве исходных данных (*область применимости алгоритма*), т.е. алгоритм пригоден для решения любой задачи из некоторого класса задач.

Т.е. один и тот же алгоритм можно применять к большому числу данных.

Это свойство не следует понимать как возможность решить много задач.

СВОЙСТВА АЛГОРИТМОВ

Понятность. Алгоритм должен быть понятен не только автору, но и исполнителю.

Выполнимость. Алгоритм должен содержать команды, записанные на понятном языке и выполнимые исполнителем.

СВОЙСТВА АЛГОРИТМА

Дискретность

- Процесс решения задачи разбит на последовательно выполняемые шаги.

Понятность и выполнимость

- Алгоритм должен состоять из команд, понятных исполнителю, написанных на его языке, входящие в его СКИ.

Результативность

- Исполнение алгоритма должно приводить к конкретному результату.

Массовость

- Один и тот же алгоритм можно применять к большому количеству данных.

Детерминированность

- Последовательность команд алгоритма должна выполняться

ФОРМЫ ЗАПИСИ АЛГОРИТМОВ

1. Словесно-формульный

Например, Составить алгоритм решения арифметического выражения $(23+34)*57/3$

1 шаг $23+34=57$

2 шаг $57*57=3249$

3 шаг $3249/3=1083$

2. С помощью алгоритмического языка

Например, Составить алгоритм решения алгебраического выражения $x=2y+z$

алг Выражение

арг y, z :цел

рез x :цел

нач

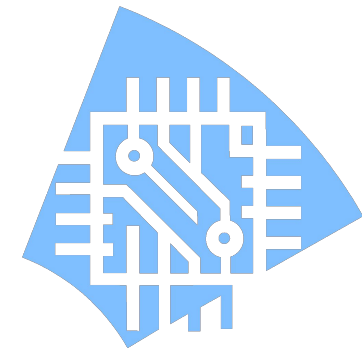
$x:=2*y$

$x:=x+z$



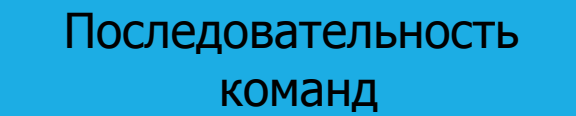

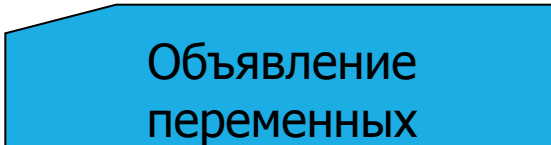
кон

3. Таблицы

4. Блок-схемы



ЭЛЕМЕНТЫ БЛОК-СХЕМЫ

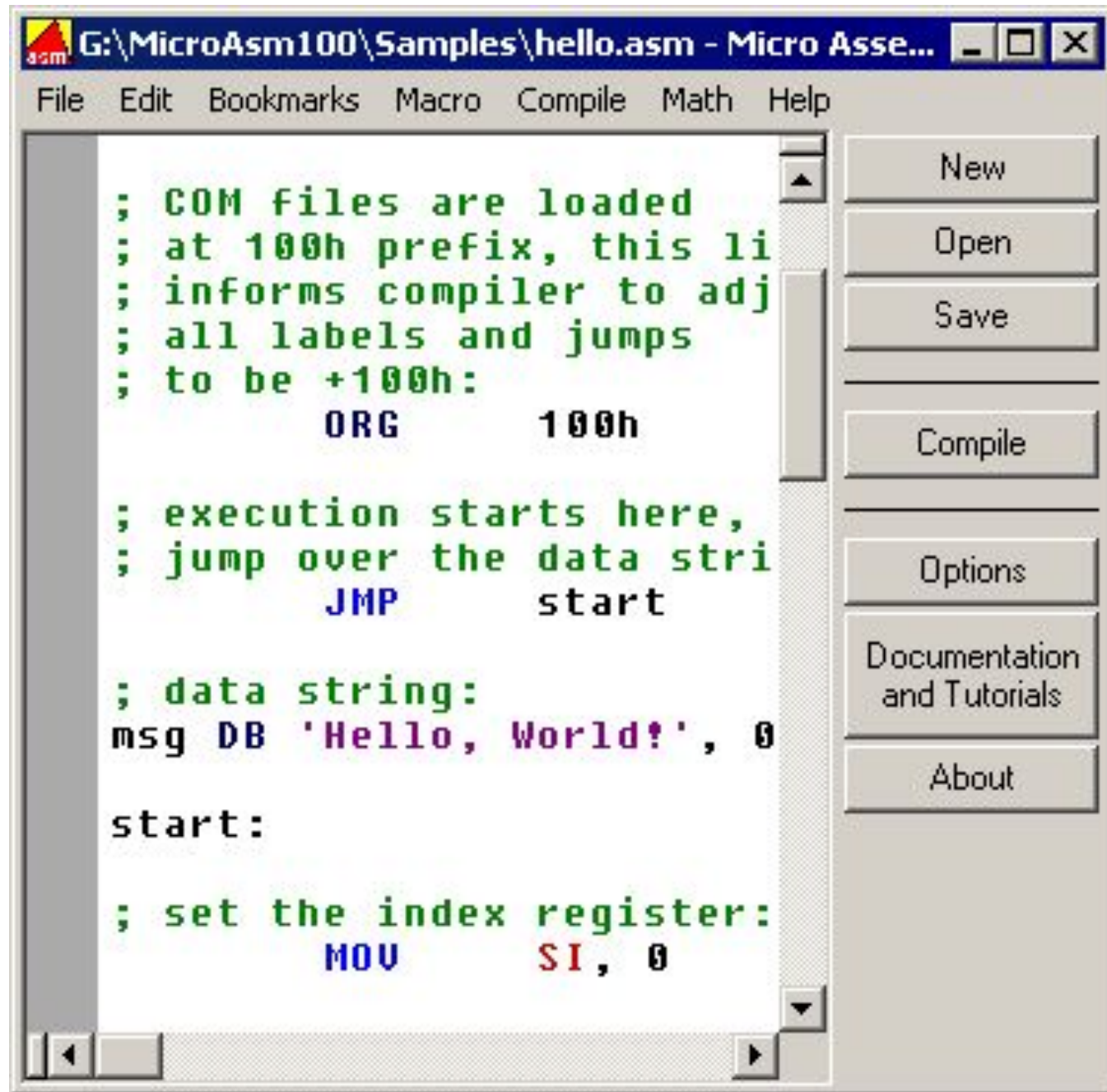
 <p>Начало</p>	Прямоугольник с закругленными углами, применяется для обозначения начала или конца алгоритма
 <p>Данные</p>	Параллелограмм, предназначен для описания ввода или вывода данных, имеет один вход сверху и один выход внизу
 <p>Последовательность команд</p>	Прямоугольник, применяется для описания линейной последовательности команд, имеет один вход сверху и один выход внизу
 <p>Условие</p>	Ромб, служит для обозначения условий в алгоритмических структурах «ветвление» и «выбор», имеет один вход сверху и два выхода (налево, если условие выполняется, и направо, если условие не выполняется)
 <p>Объявление переменных</p>	Прямоугольник со срезанным углом, применяется для объявления переменных или ввода комментариев.

МАШИННЫЙ ЯЗЫК

The image shows a debugger window with the title "view notepad.exe - Far". The main area displays assembly code for "notepad.exe" at address 01008240. The code includes instructions like "mov", "xor", "push", "call", "cmp", and "je". A yellow box highlights the assembly code from address 01008140 to 01008190. A red arrow points to the instruction at 01008090: "mov ecx, 0C0000000". To the right of the assembly code is a memory dump showing hexadecimal values and their corresponding ASCII characters. At the bottom, a menu bar includes "Global", "File", "Edit", "Reload", "Direct", "Kiat", and "10".

```
view notepad.exe - Far
01008010: 78 00 00 00-01 00 00 00-4E 00 6F 00-74 00 65 00  x 0 Note
01008020: 70 00 61 00-64 00 00 00-FF FF FF FF-01 00 00 00  p a d
01008030: 03 00 00 00-05 00 00 00-0A 00 00 00-0B 00 00 00  v
01008040: 10 00 00 00-11 00 00 00-0C 00 00 00-12 00 00 00  >
01008050: 13 00 00 00-18 00 00 00-19 00 00 00-1A 00 00 00  ||
01008060: 1E 00 00 00-1F 00 00 00-20 00 00 00-22 00 00 00  ^
01008070: 23 00 00 00-2B 00 00 00-2C 00 00 00-2D 00 00 00  #
01008080: 2E 00 00 00-2F 00 00 00-30 00 00 00-32 00 00 00  .
01008090: 34 0  mov ecx,0C0000000 4 3 > ?
010080A0: 17 0  mov edx,000401000  I F G P
010080B0: 51 0  xor eax,edx  Q R S
010080C0: 2C 8  push eax  ,H 00H 04H 08H 0
010080D0: 3C 8  push 000000080 ;" H" <H 0DH 0EH 0FH 0
010080E0: 4C 8  push 003  LH 0PH 0aH 0tH 0
010080F0: 58 8  push eax  xH 0xH 0'H 0dH 0
01008100: 68 8  push eax  hH 0iH 0pH 0tH 0
01008110: 78 8  push ecx  xH 0jH 0qH 0vH 0
01008120: 88 8  push edx  vH 0yH 0pH 0vH 0
01008130: 9C 8  call 001008547 ----- (1) vH 0yH 0aH 0vH 0
01008140: AC 8  cmp eax,-001 ;"0"  mH 0zH 0wH 0vH 0
01008150: 98 1  je 001008478 ----- (2) v! 0  v 0w 0
01008160: 02 0  mov [000401000],eax 0 aH 0  tH 0
01008170: 05 0  push 000  z 0w 0z 0w 0
01008180: 04 1  push d,[000401000]  z> vH 0z> 0w 0
01008190: 0B 1  call 001008541 ----- (3) z> P 0y
010081A0: 00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
010081B0: B9 00 00 00-C0 BA 00 10-40 00 33 C0-50 68 80 00  v v >@ 3 vPhH
010081C0: 00 00 6A 03-50 50 51 52-E8 7A 03 00-00 83 F8 FF  j vPPQR v z v
010081D0: 0F 84 A5 02-00 00 A3 0A-10 40 00 6A-00 FF 35 0A  <D v 0  v >@ j v50
010081E0: 10 40 00 E8-59 03 00 00-A3 0E 10 40-00 8B 00 0E  >@ v v v >@ n v
010081F0: 10 40 00 83-C1 52 E8 95-02 00 00 A3-7D 10 40 00  >@ v v R v X0 v ] >@
01008200: C7 05 12 10-40 00 00 00-01 00 81 3D-0E 10 40 00  v z i >@ 0 v = >@
01008210: 00 00 01 00-77 0A A1 0E-10 40 00 A3-12 10 40 00  0 w v >@ v i >@
Global 2 File 3 Reload 5 Direct 8 Kiat 9 10
```

АССЕМБЛЕР



The screenshot shows the Micro Assembler (asm) software interface. The title bar reads "G:\MicroAsm100\Samples\hello.asm - Micro Asse...". The menu bar includes "File", "Edit", "Bookmarks", "Macro", "Compile", "Math", and "Help". The main text area contains the following assembly code:

```
; COM files are loaded
; at 100h prefix, this li
; informs compiler to adj
; all labels and jumps
; to be +100h:
        ORG     100h

; execution starts here,
; jump over the data stri
        JMP     start

; data string:
msg DB 'Hello, World!', 0

start:

; set the index register:
        MOV     SI, 0
```

On the right side of the window, there is a vertical toolbar with buttons for "New", "Open", "Save", "Compile", "Options", "Documentation and Tutorials", and "About".

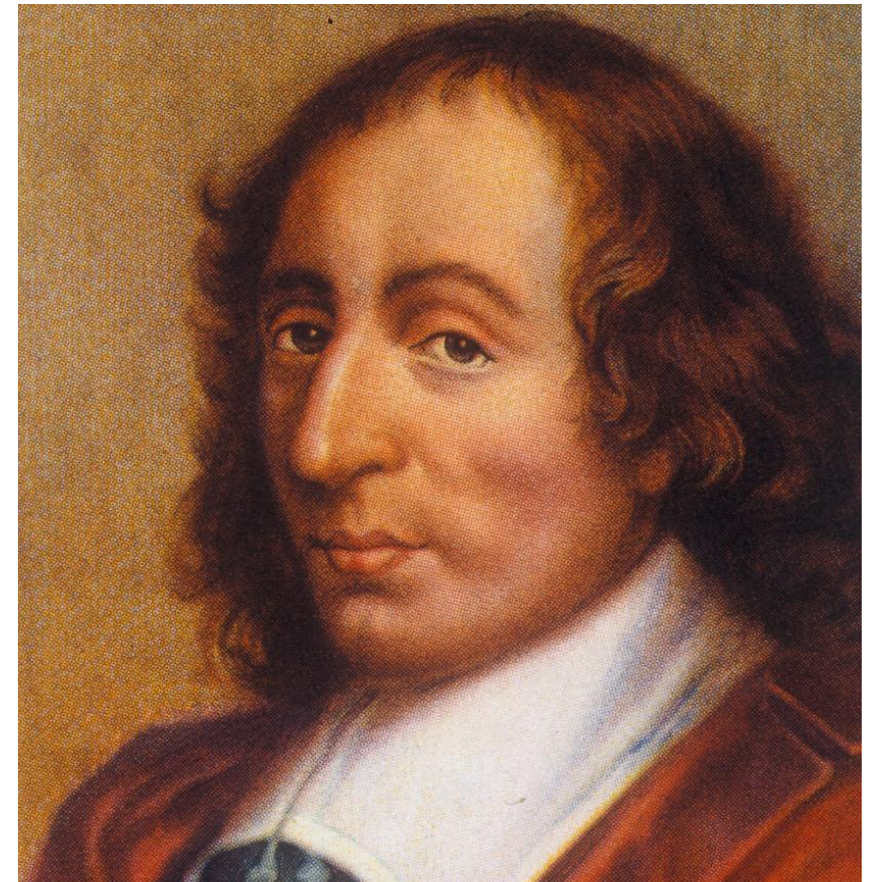
ЯЗЫКИ
ПРОГРАММИРОВАНИЯ
ВЫСОКОГО УРОВНЯ

QBASIC



PASCAL

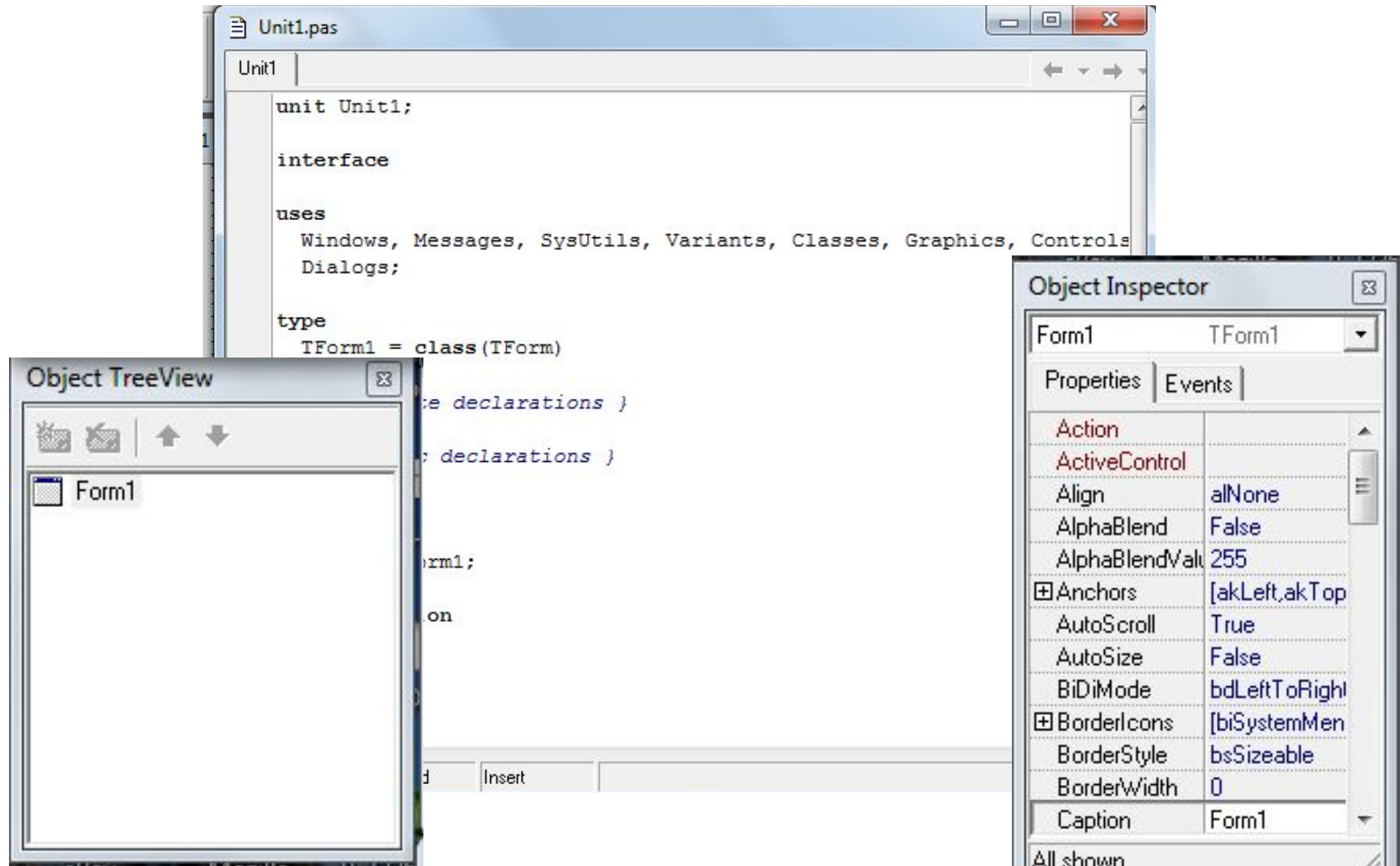
```
Turbo Pascal
File Edit Search Run Compile Debug Tools Options Window Help
PAS\EXAMPLOO.PAS
< 11 областная олимпиада школьников. Киров. 1999 год.
Ребус UOLVO + FIAT = MOTOR >
Program Rebus;
Uses Crt;
Const S:array[1..5] of longint = (10,100,1000,10000,100000);
Var A1, A2, A3:longint;
    U,O,L,F,I,A,T:byte;
    Ss:set of 0..9;
Procedure Prov; < A1 + A2 = A3 ? >
var B:longint;
    Pr:byte;
begin
  b:=a3; pr:=b div s[4]; if pr in Ss then exit;
  b:=b mod s[4]; pr:=b div s[3]; if pr <> 0 then exit;
  b:=b mod s[3]; pr:=b div s[2]; if pr <> 1 then exit;
  b:=b mod s[2]; pr:=b div s[1]; if pr <> 0 then exit;
  pr:=b mod s[1]; if pr in Ss then exit;
  Writeln(A1,' + ',A2,' = ',A3);
end;
18:26
F1 Help F2 Save F3 Open Alt+F9 Compile F9 Make Alt+F10 Local menu
```



Французский физик-
математик
Блез Паскаль

Программа Pascal, названная в
честь
Блеза Паскаля

DELPHI



ТИПЫ АЛГОРИТМОВ

1. Линейный
2. Разветвлённый(алгоритмические структуры «ветвление» и «выбор»)
3. Циклический (алгоритмическая структура «цикл»)
4. вспомогательный

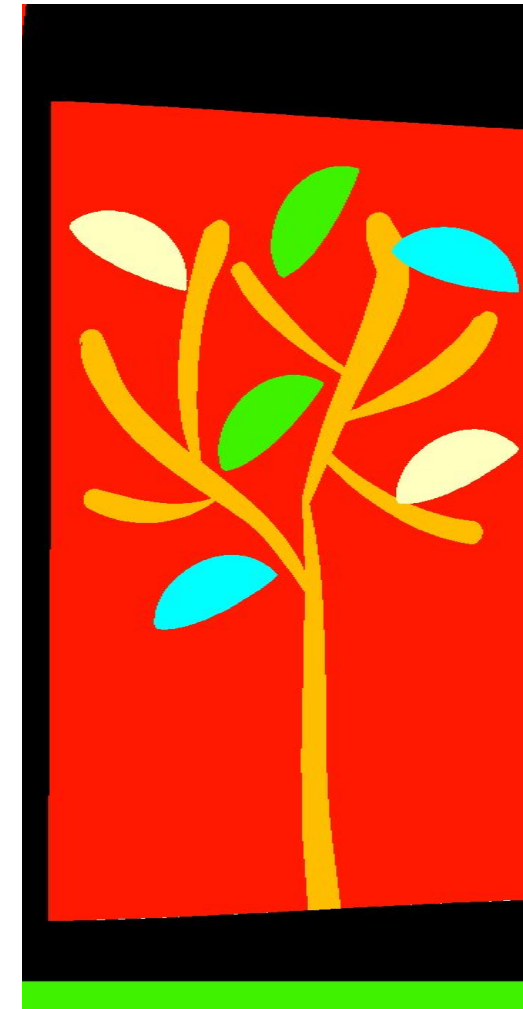
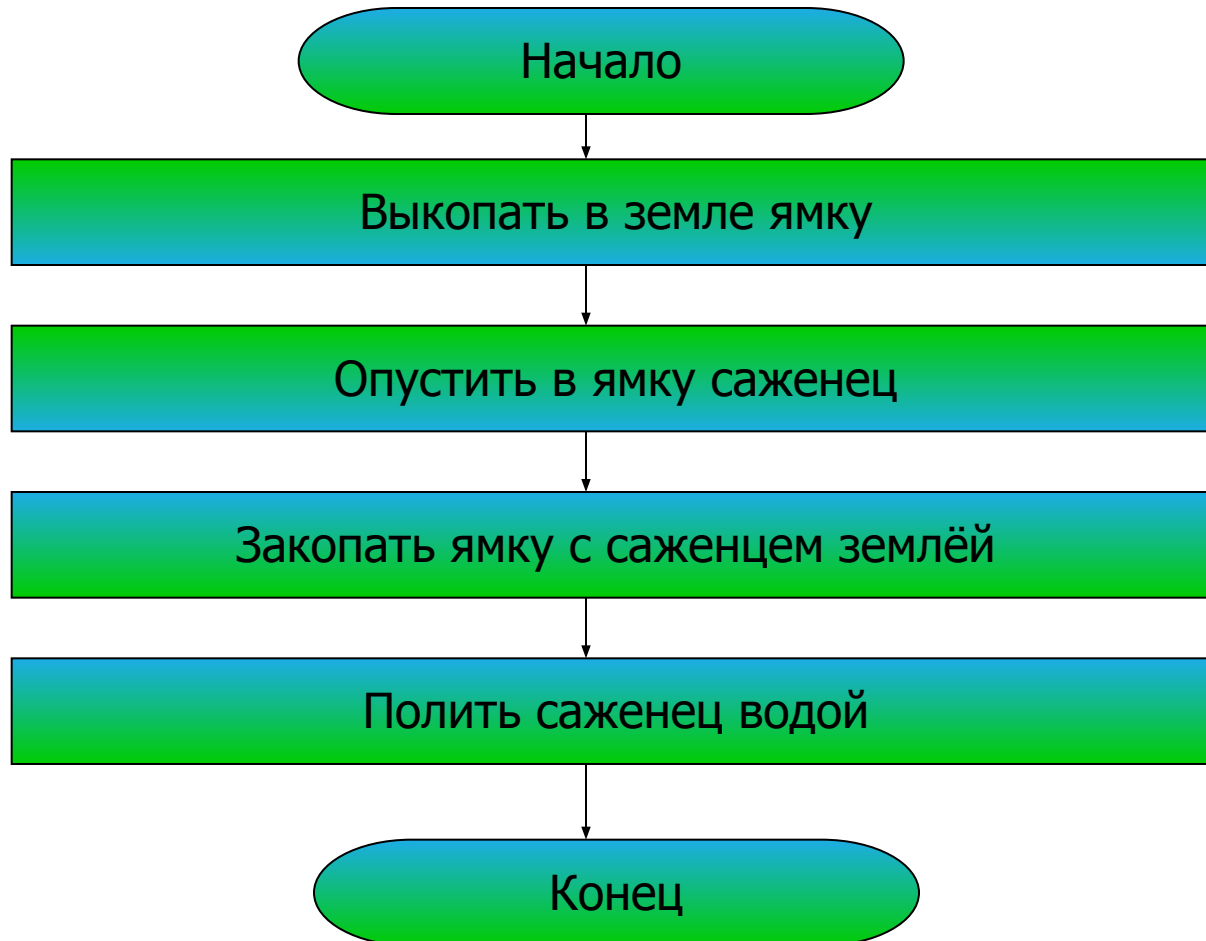


ТИПЫ АЛГОРИТМОВ

1. **Линейный алгоритм** – это алгоритм, в котором команды выполняются **последовательно одна за другой**.
2. **Разветвлённый алгоритм** – алгоритм, в котором в зависимости от истинности или ложности **условия** выполняются одна или другая серия команд.
3. **Циклический алгоритм** – это алгоритм, в котором одна и та же последовательность действий совершается **множественно** (или ни разу) до тех пор, пока выполняется условие.
4. **Вспомогательный алгоритм** – **самостоятельный алгоритм**, снабжённый таким **заголовком**, который позволяет вызывать этот алгоритм из других алгоритмов.

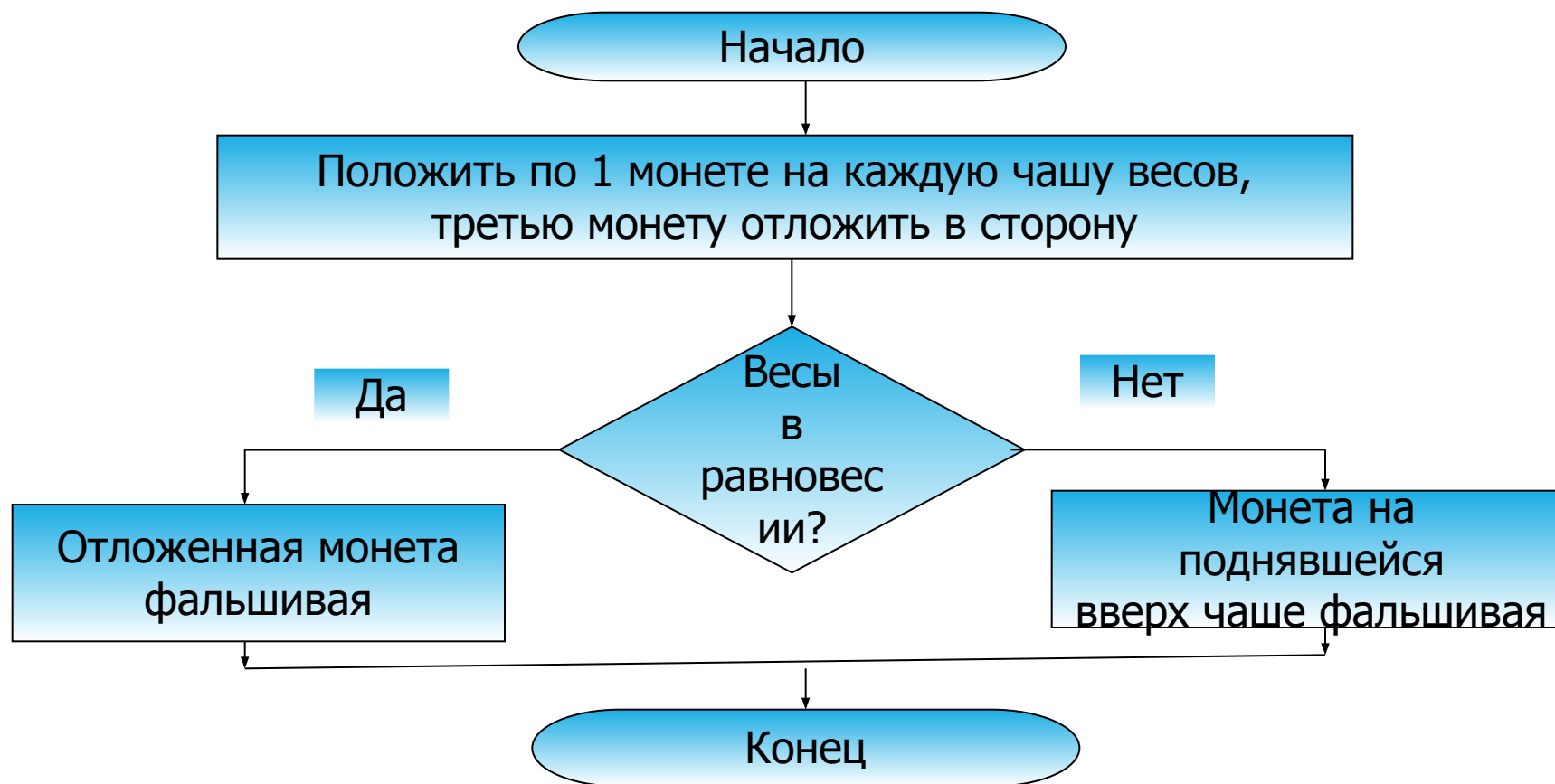
ЛИНЕЙНЫЙ АЛГОРИТМ

Пример. Алгоритм посадки дерева.



РАЗВЕТВЛЁННЫЙ АЛГОРИТМ

Из трёх монет одинакового достоинства одна фальшивая (лёгкая). Как её найти с помощью одного взвешивания на чашечных весах без гирь?



ЦИКЛИЧЕСКИЙ АЛГОРИТМ

Домашнее задание по математике

