

Управление исполнителями

§ 29. Алгоритмы и исполнители

Что такое алгоритм?

Алгоритм – это порядок выполнения действий.

Исполнитель – это устройство или одушевлённое существо (человек), способное понять и выполнить команды, составляющие алгоритм.

Формальные исполнители: не понимают (и не могут понять) смысл команд.

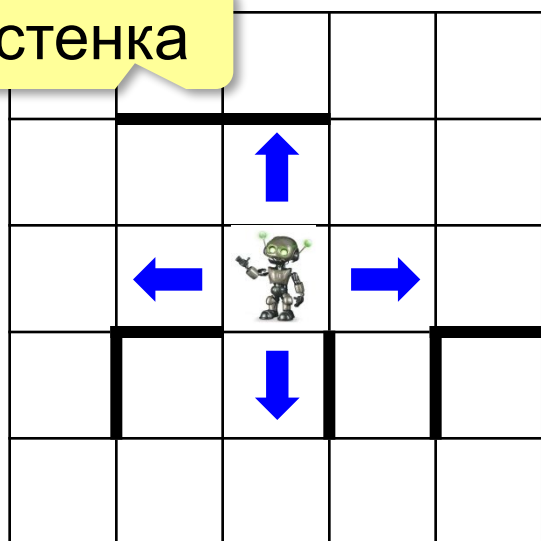


Мухаммед ал-Хорезми
(ок. 783–ок. 850 гг.)

Алгоритм — это точное описание порядка действий некоторого исполнителя.

Исполнитель Робот

стенка



Среда — это обстановка, в которой работает исполнитель.

Система команд исполнителя (**СКИ**):

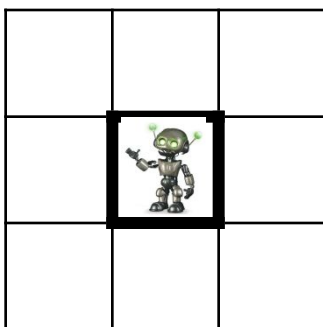
вверх

вправо

вниз

влево

Состояние исполнителя:



Какие команды может выполнить Робот?

Свойства алгоритма

Дискретность — алгоритм состоит из отдельных команд, каждая из которых выполняется ограниченное (не бесконечное) время.

Понятность — алгоритм содержит только команды, входящие в **систему команд исполнителя**.

Определённость — при каждом выполнении алгоритма с одними и теми же исходными данными должен быть получен один и тот же результат.



Если какое-то свойство нарушено, это не алгоритм!

Необязательные свойства алгоритма

- ? **Конечность** (результативность) — для корректного набора данных алгоритм должен заканчиваться с некоторым результатом (не **зацикливаться**).
- ? **Корректность** — для допустимых исходных данных алгоритм должен приводить к правильному результату.
- ? **Массовость** — алгоритм можно использовать для решения множества однотипных задач с различными исходными данными (решение «в буквах»).

Одна задача – много алгоритмов

Задача. Вычислите

$$S = 1 + 2 + 3 + 4 + 5 + \dots + 99 + 100$$



Как можно вычислять?



Решение К.Ф. Гаусса:

$$\begin{aligned} 1 + 100 &= 2 + 99 = 3 + 98 = \dots \\ &= 50 + 51 = 101 \end{aligned}$$

$$S = 50 \cdot 101 = 5050$$



Какой алгоритм лучше? Почему?

Управление исполнителями

Ручное (непосредственное, «с пульта»):



Можно и без плана!

Программное (по готовой программе):



бортовой
компьютер

Программа — это алгоритм,
записанный на языке, понятном
компьютеру.

Управление исполнителями

§ 30. Способы записи алгоритмов

Алгоритм «О»

Словесная форма:

Даны два натуральных числа. Пока первое число не меньше второго, заменять его на разность первого и второго. Результат работы алгоритма — полученное первое число.

	<i>Исходные данные</i>	<i>Шаг 1</i>	<i>Шаг 2</i>
<i>a</i>	5	3	1
<i>b</i>	2	2	2



Меняется ли b ?



▪ неоднозначность естественных языков

Алгоритм «О»

По шагам:

Вход: два натуральных числа, a и b .

Шаг 1. Если $a < b$, перейти к шагу 4 (**Стоп**).

Шаг 2. Заменить a на $a - b$.

Шаг 3. Перейти к шагу 1.

Шаг 4. Стоп.

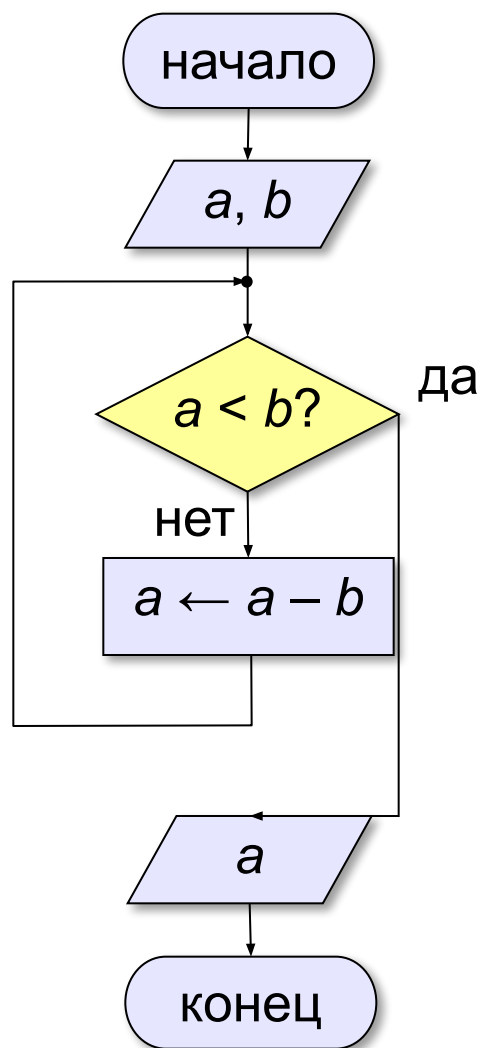
Результат: значение a .



▪ не все знают русский язык

Алгоритм «О»

Блок-схема:



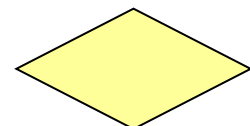
Условные обозначения



начало и конец алгоритма



ввод и вывод данных



условие (выбор)



операции с данными

присвоить a
значение $a - b$

Ручная прокрутка (трассировка)

Вход: два натуральных числа, a и b .

Шаг 1. Если $a < b$, перейти к шагу 4.

Шаг 2. Заменить a на $a - b$.

Шаг 3. Перейти к шагу 1.

Шаг 4. Стоп.

Результат: значение a .

	Действие	Условие верно?	a	b
1	Вход		19	5
2				
3				
4				
5				
6				
7				
8				
9				

исходные данные



Где ответ?

Переменные

Переменная — это величина, значение которой можно изменять во время работы алгоритма.

Вход: два натуральных числа, a и b .

Шаг 1. Если $a < b$, перейти к шагу 4.

Шаг 2. Заменить a на $a - b$.

Шаг 3. Перейти к шагу 1.

Шаг 4. Стоп.

Результат: значение a .

$a \leftarrow a - b$

или

$a := a - b$

присваивание
значения

Языки программирования

Программа — это алгоритм, записанный на языке, понятном компьютеру.

 Какой язык понимает компьютер?

Алгоритм «О»:

```
101110000000111100000000
101110110000010000000000
0011101111000011
0111110000000100
0010101111000011
1110101111111000
1100110100100000
```

 Что плохо?

 ■ сложно писать и понимать программы

Язык ассемблера

Машинные коды:

```
101110000000111100000000
101110110000010000000000
0011101111000011
0111110000000100
0010101111000011
1110101111111000
1100110100100000
```

Язык ассемблера:

```
mov ax, 15
mov bx, 4
m:   cmp ax, bx
     j1 end
     sub ax, bx
     jmp m
end: int 20h
```

Ассемблер — это программа, которая переводит символьную запись команд в машинные коды.



Машинные коды и язык ассемблера – это языки низкого уровня (машинно-ориентированные)!



■ **непереносимость программ**

зависят от
процессора!

Языки высокого уровня

- 1) легко понимаются человеком
- 2) не «привязаны» к командам конкретного процессора

Школьный алгоритмический язык:

```
цел а, б
```

```
а := 15
```

```
б := 4
```

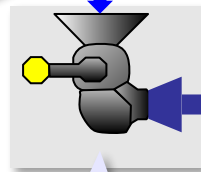
```
нц пока а >= б
```

```
а := а - б
```

```
кц
```



Как процессор поймёт?



Транслятор (переводчик) — это программа, которая переводит программу на языке высокого уровня в машинные коды.

Языки высокого уровня

1957: FORTRAN = FORmula TRANslator
для решения научных задач

1972: C (Д. Ритчи, К. Томпсон)

↳ **C++, C#, Java, JavaScript, ...**

1991: Python (Г. ван Россум)

Для программирования сайтов:
PHP, JavaScript

Логическое программирование:
Prolog

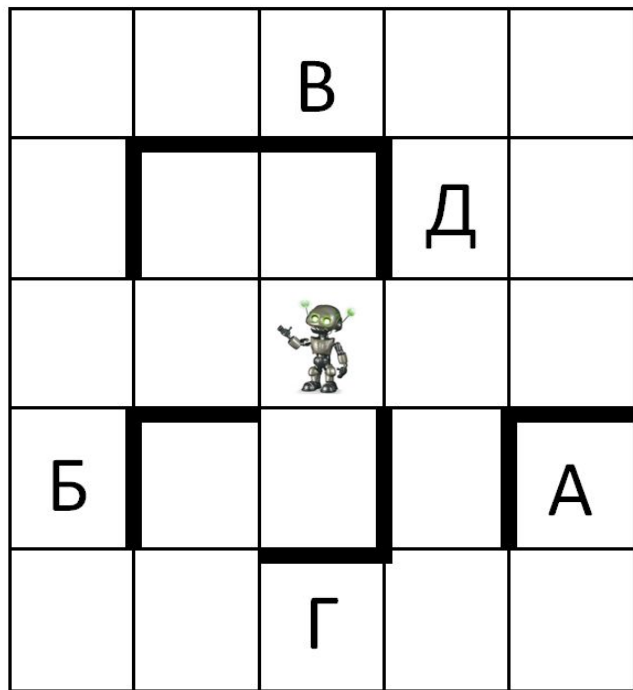
Учебные языки:
BASIC, Паскаль, Школьный алгоритмический язык

Управление исполнителями

§ 31. Примеры исполнителей

Формальный исполнитель

Формальный исполнитель — это исполнитель, который одну и ту же команду всегда понимает однозначно и выполняет одинаково.



СКИ Робота

1. вверх
2. вправо
3. вниз
4. влево

443 → Б

2114 → В

23321 → А

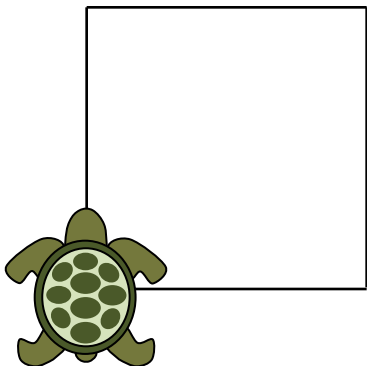
2334 → Г

21 → Д

? Куда?

? Как иначе?

Исполнитель Черепаха



вперед 30

вправо 90

вперед 30

вправо 90

вперед 30

вправо 90

вперед 30

вправо 90

шагов

градусов



Как нарисовать окружность?

$$= \frac{360^\circ}{4}$$

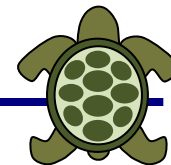
число
сторон

повтори 4 [вперед 30 вправо 90]

повтори 12 [вперед 50 вправо 45]

повтори 10 [вперед 50 вправо 60]

Исполнитель Черепаха



повтори 4 [вперед 30 вправо 45]

незамкнутая ломаная

повтори 45 [вперед 30 вправо 45
вправо 45]

повтори 12 [вправо 15 вперед 30
вправо 45]

повтори 5 [вправо 15 вперед 30
вправо 15]

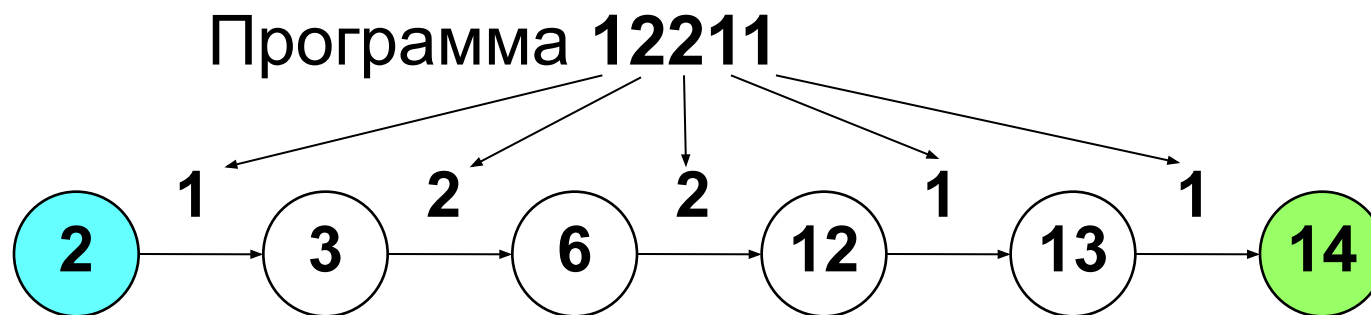
повтори 15 [вправо 80 вперед 30
влево 35]

Исполнитель Удвоитель

Работает с одним числом и умеет выполнять с ним две операции (команды):

1. прибавь 1
2. умножь на 2

Программа – это последовательность номеров команд, которые нужно выполнить.



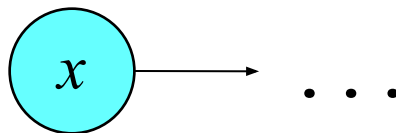
начальное
число

результат

Исполнитель Удвоитель

1. прибавь 1

2. умножь на 2



Какие числа можно получить?

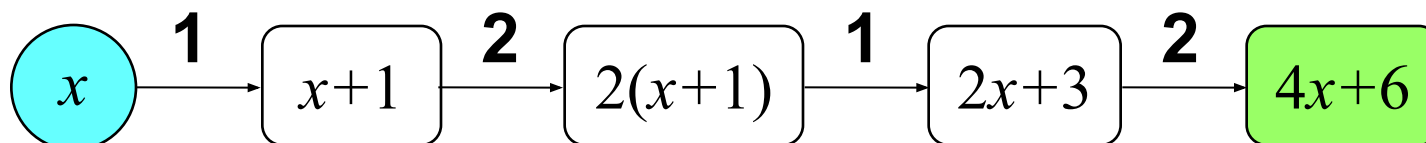
• при целом $x \geq 0$

$x, x+1, x+2, \dots$

• при целом $x < 0$

любые целые

Программа **1212**



Могли ли получить 36? а 34?

Исполнитель Шифровальщик

Если цепочка символов начинается с гласной буквы, Шифровальщик переставляет последнюю букву в начало слова, а если с согласной, то меняет местами первую и вторую буквы.

согласная

Этот алгоритм применили к слову **КОТИК**. Какое слово получилось?

КОТИК → **ОКТИК**

Этот алгоритм **дважды** применили к слову **КОТИК**. Какое слово получилось?

КОТИК → **ОКТИК** → **КОКТИ**

Исполнитель Шифровальщик

Если в цепочке символов чётное количество букв, Шифровальщик добавляет в середину слова букву Я, а если нечётное – удваивает среднюю букву.

Этот алгоритм применили к слову **КОТИК**. Какое слово получилось?

КОТИК → **КОТТИК**

Этот алгоритм **дважды** применили к слову **КОТИК**. Какое слово получилось?

КОТИК → **КОТТИК** → **КОТЯТИК**

Исполнитель Шифровальщик

АБВГДЕЁЖЗИКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ

А → Б Б → В



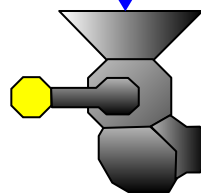
Что делать с Я?

Я → А

ПРИВЕТ ВАСЯ

П → Р

Р → С



РСКГЁУ ГБТА

Расшифруйте:

АВМПЛП ← ЯБЛОКО

НПСЛПГЭ ← МОРКОВЬ

ЛМАЛТБ ← КЛЯКСА

Шифр Цезаря

Управление исполнителями

§ 32. Оптимальные программы

Что такое оптимальная программа?

Оптимальная программа — это самая лучшая программа по какому-то показателю.



Как сравнить две программы?

Напишите две программы для Удвоителя:

$3 \rightarrow \dots \rightarrow 7$



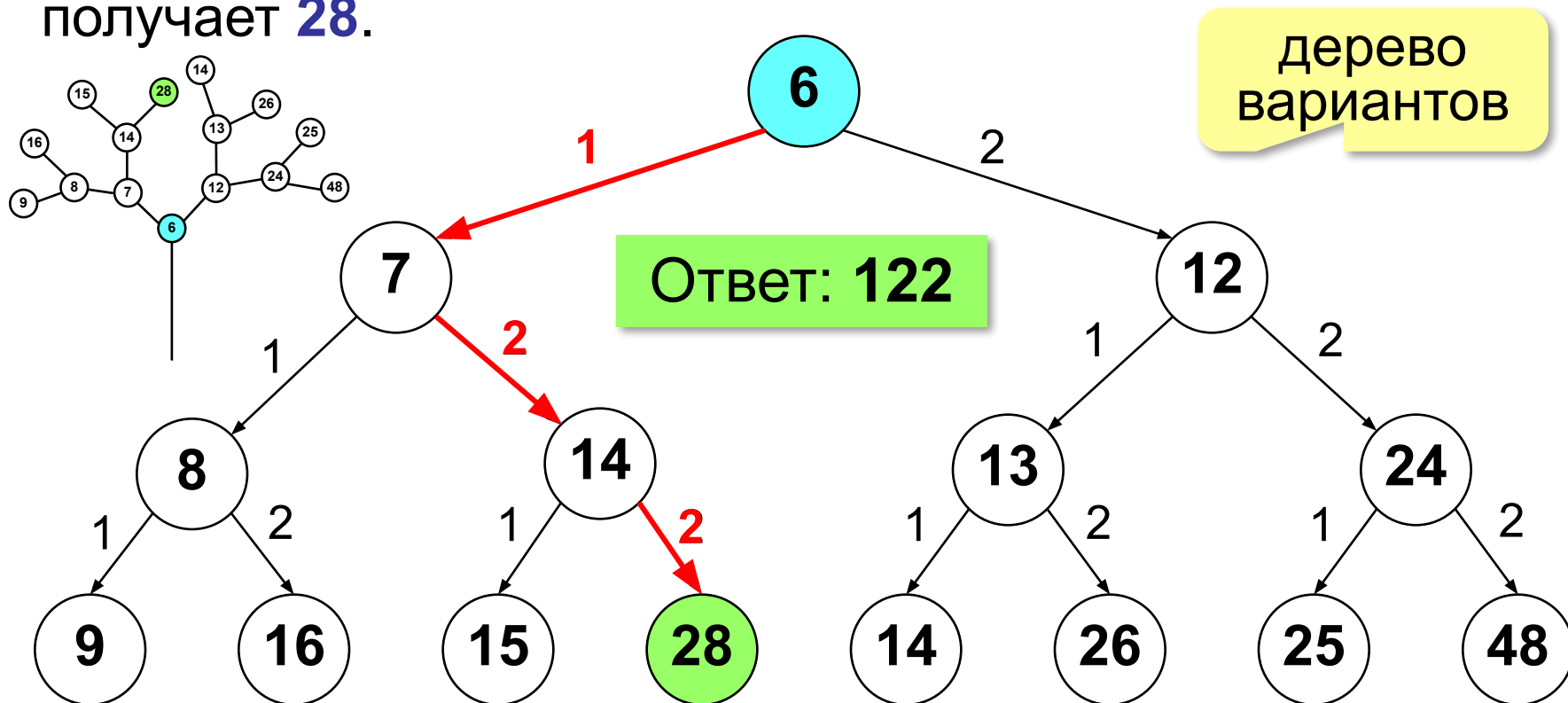
Всегда ли оптимальная программа лучше других по всем критериям?

Составление программы

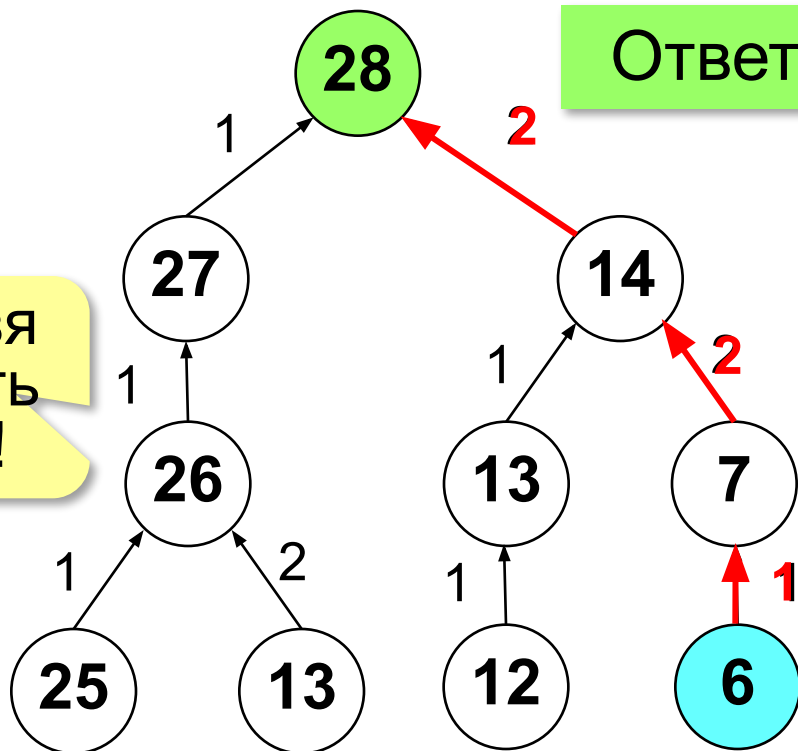
Используя команды:

1. прибавь 1
2. умножь на 2

написать самую короткую программу, которая из **6** получает **28**.



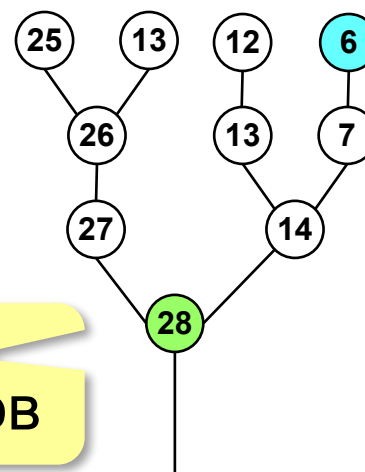
Составление программы (с конца)



Ответ: 122

нельзя
делить
на 2!

дерево
вариантов



Почему решение
«с конца» короче?



Решение «с конца» короче, если в списке команд есть **необратимая операция** (каждое целое число можно умножить на 2, но не каждое делится на 2)!