Управление исполнителями

Управление исполнителями

§ 29. Алгоритмы и исполнители

Что такое алгоритм?

Алгоритм – это порядок выполнения действий.

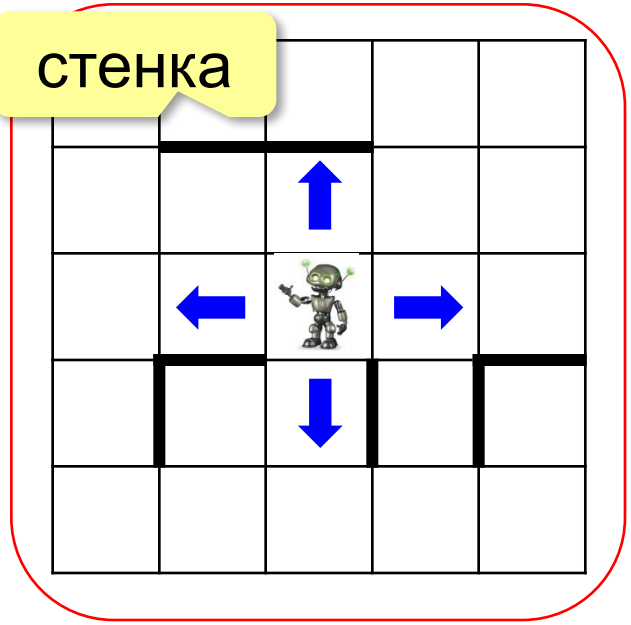
Исполнитель – это устройство или одушевлённое существо (человек), способное понять и выполнить команды, составляющие алгоритм.

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

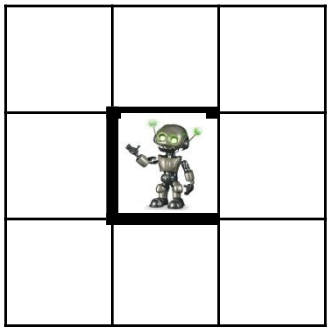


Среда — это обстановка, в которой работает исполнитель.

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

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

? Как можно вычислять?

Решение К.Ф. Гаусса:

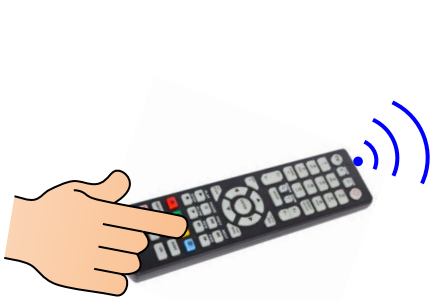
$$1 + 100 = 2 + 99 = 3 + 98 = \ldots$$
$$= 50 + 51 = 101$$

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

? Какой алгоритм лучше? Почему?

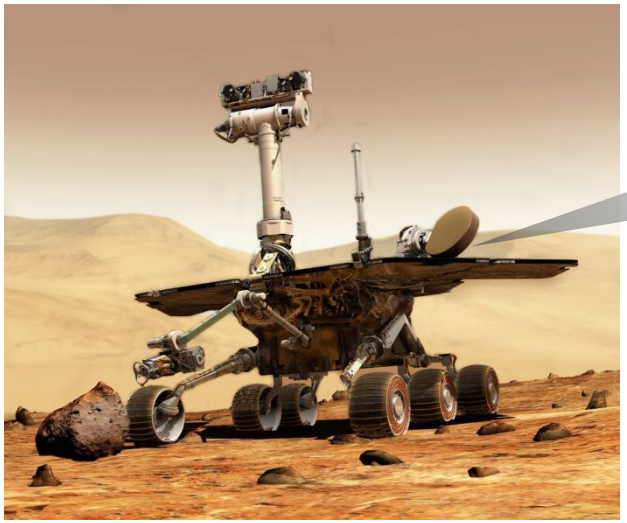
Управление исполнителями

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.

	Исходные данные	*Шаг 1*	*Шаг 2*
a	**5**	**3**	**1**
b	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.

Шаг 2. Заменить *a* на *a* – *b*.

Шаг 3. Перейти к шагу 1.

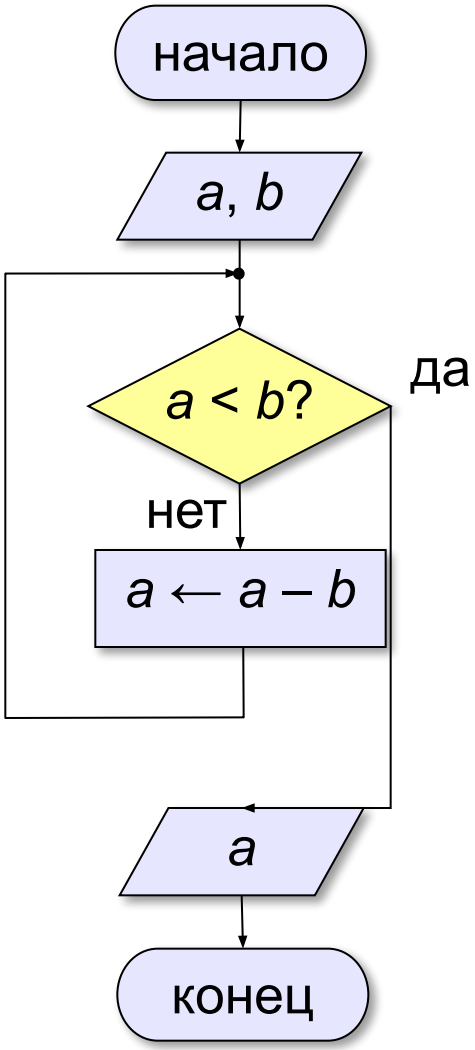
Шаг 4. Стоп.

Результат: значение *a*.

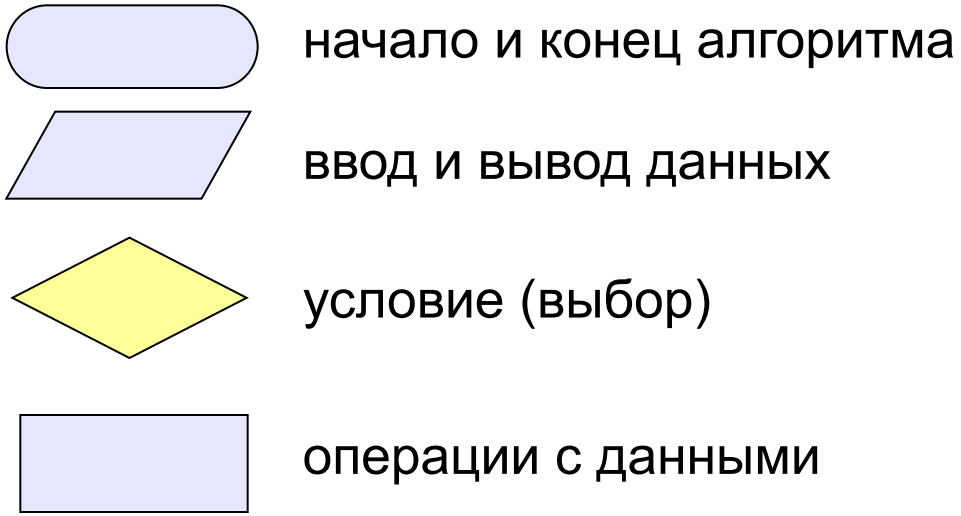
- не все знают русский язык

Алгоритм «О»

Блок-схема:



Условные обозначения



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 ← a – b
```
или
```
a := a – b
```
присваивание значения

Языки программирования

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

Какой язык понимает компьютер?

Алгоритм «О»:

```
10111000000111100000000
10111011000001000000000
00111011111000011
01111110000000100
00101011111000011
11101011111111000
11001101001000000
```

Что плохо?

- сложно писать и понимать программы

Язык ассемблера

Машинные коды:	Язык ассемблера:	
10111000000111100000000		mov ax, 15
10111011000001000000000		mov bx, 4
001110111000011	m:	cmp ax, bx
011111000000100		jl end
001010111000011		sub ax, bx
1110101111111000		jmp m
1100110100100000	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

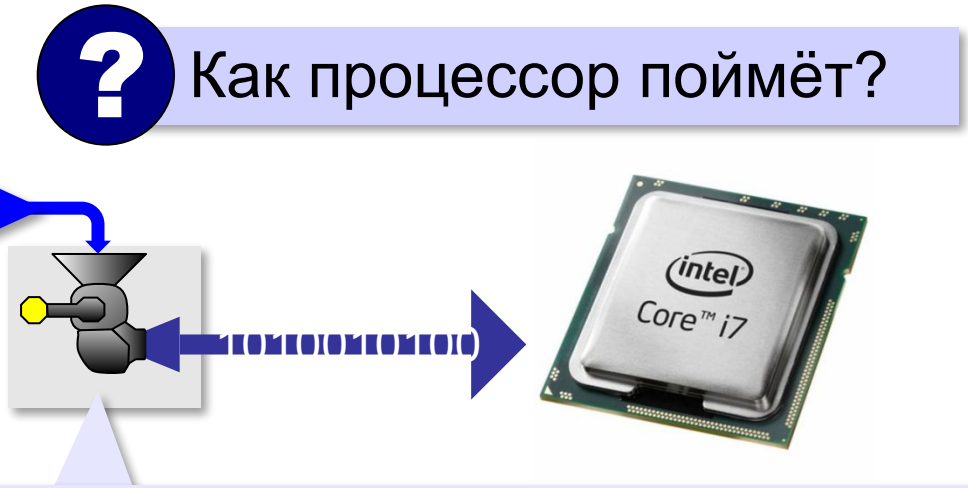
зависят от процессора!

Языки высокого уровня

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

Школьный алгоритмический язык:

```
цел a, b
a:=15
b:=4
нц пока a>=b
  a:=a-b
кц
```



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:

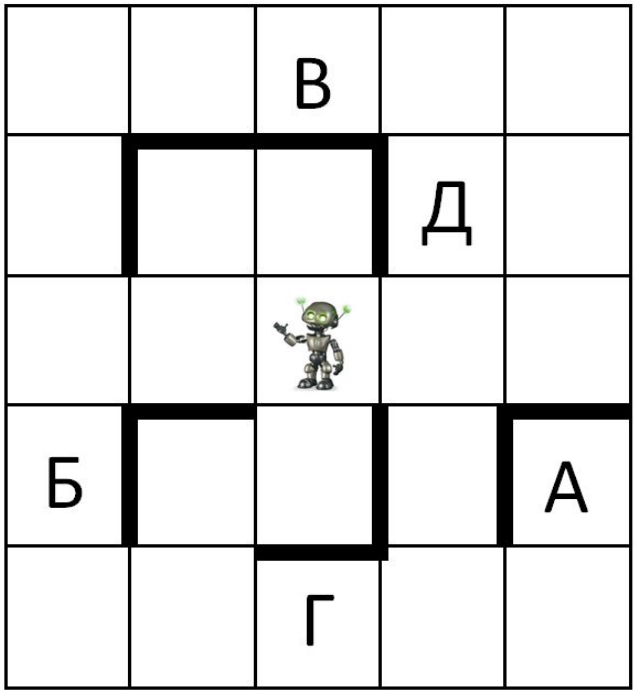
BASIC, Паскаль, Школьный алгоритмический язык

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**

? Куда?

23321 → **А**

? Как иначе?

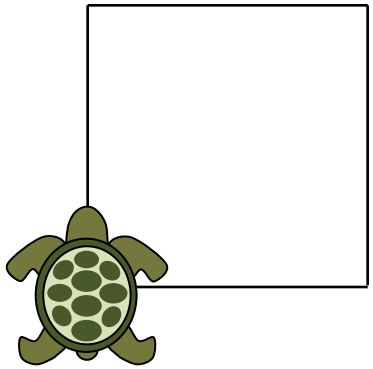
443 → **Б**

2334 → **Г**

2114 → **В**

21 → **Д**

Исполнитель Черепаха



```
вперед 30
вправо 90
вперед 30
вправо 90
вперед 30
вправо 90
вперед 30
вправо 90
```

шагов

градусов

? Как нарисовать окружность?

$= \frac{360°}{4}$

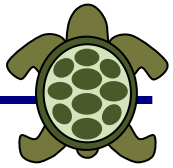
число сторон

```
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 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

АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ

А→Б Б→В

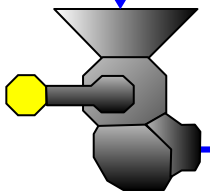
Что делать с Я?

Я→А

ПРИВЕТ ВАСЯ

П→Р

Р→С



РСКГЁУ ГБТА

Расшифруйте:

АВМПЛП ← ЯБЛОКО

НПСЛПГЭ ← МОРКОВЬ

ЛМАЛТБ ← КЛЯКСА

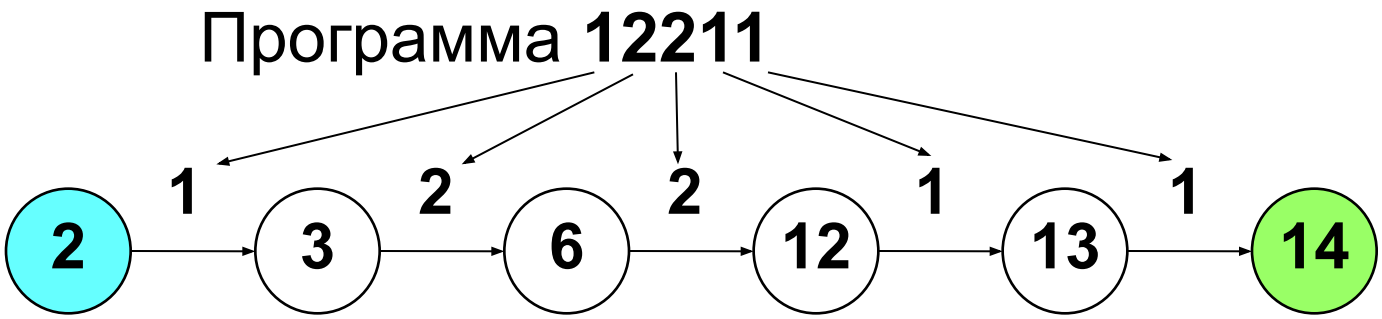
Шифр Цезаря

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.

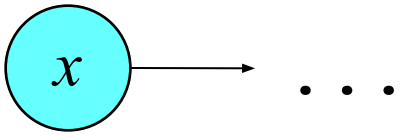


начальное число

результат

Исполнитель Удвоитель

1. **прибавь 1**
2. **умножь на 2**



Какие числа можно получить?

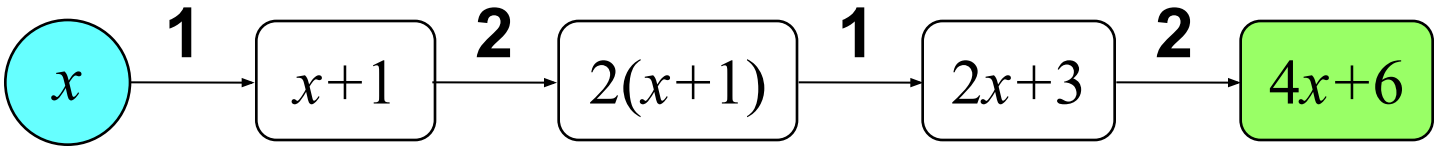
- при целом $x \geq 0$

  $x, x+1, x+2, \ldots$

- при целом $x < 0$

  любые целые

Программа **1212**



Могли ли получить 36? а 34?

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 → … → 7

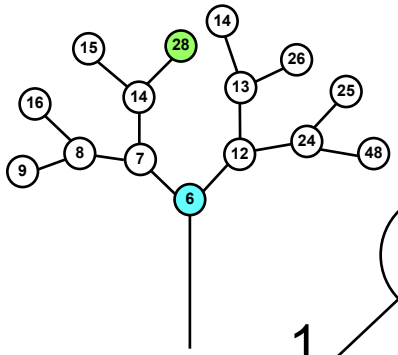
Всегда ли оптимальная программа лучше других по всем критериям?

Составление программы

Используя команды:

1. **прибавь 1**
2. **умножь на 2**

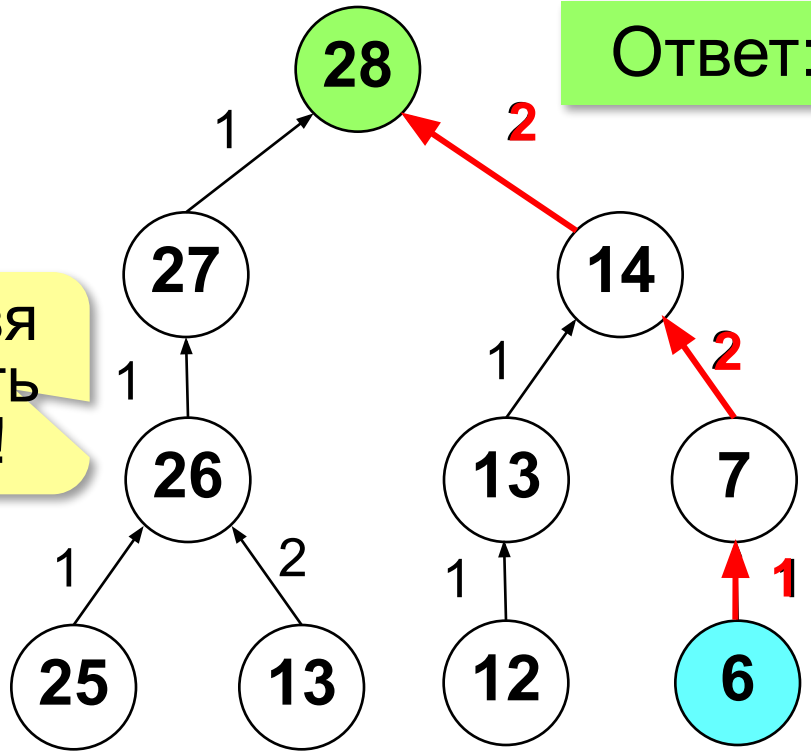
написать самую короткую программу, которая из **6** получает **28**.



дерево вариантов

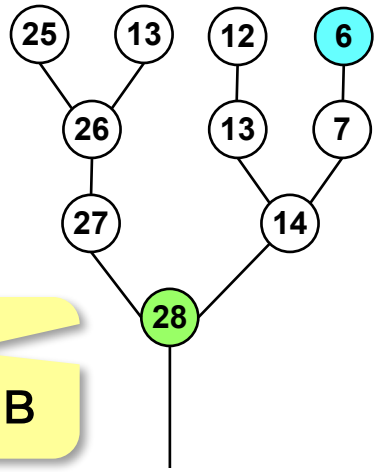
Ответ: **122**

Составление программы (с конца)



Ответ: **122**

нельзя делить на 2!



дерево вариантов

? Почему решение «с конца» короче?

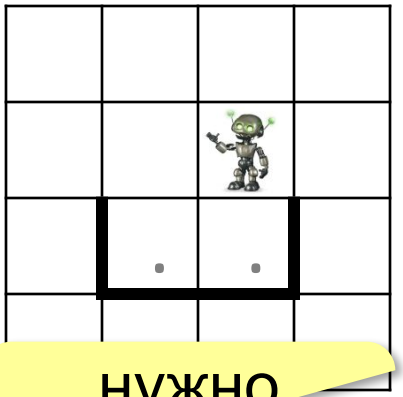
! Решение «с конца» короче, если в списке команд есть **необратимая операция** (каждое целое число можно умножить на 2, но не каждое делится на 2)!

Управление исполнителями

§ 33. Линейные алгоритмы

Что такое линейный алгоритм?

В **линейном алгоритме** команды выполняются в том порядке, в котором они записаны.



нужно закрасить

СКИ Робота:

закрасить

```
использовать Робот
алг Переход
нач
   вниз
   закрасить
   влево
   закрасить
кон
```

подключить исполнителя

служебные (зарезервированные) слова языка

Ошибки в программах

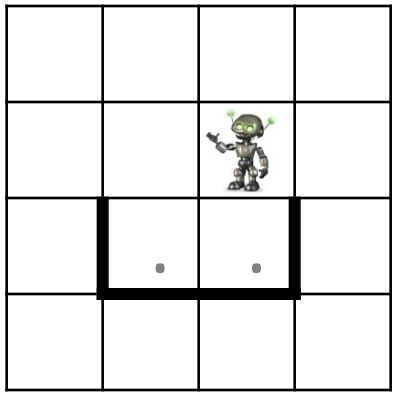
Синтаксические: исполнитель не понимает команду, так как она неверно записана.

~~закрась~~
~~налево~~

→

закрасить
влево

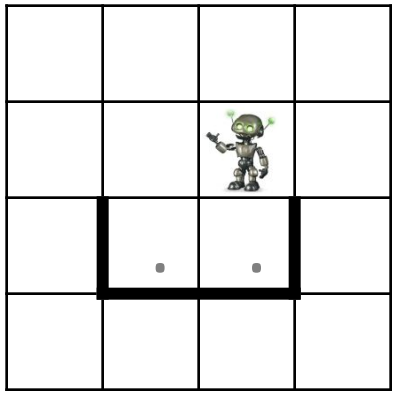
Логические: исполнитель понимает и выполняет команды, но делает не то, что нужно.



```
закрасить
влево
закрасить
```

Ошибки в программах

Логические ошибки могут привести к **отказу**!



```
вниз
закрасить
вправо
закрасить
```

столкновение со стенкой

При вычислениях: **деление на 0**.

Отладка – это поиск и исправление ошибок в программе.

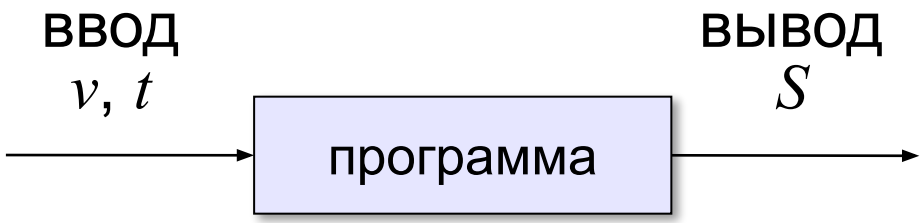
F8 – выполнение по шагам.

Вычислительные задачи

Задача. Сколько километров проехал автомобиль за **2 часа**, если его средняя скорость равна **60 км/ч**?

Массовость: решаем « в буквах».

время – t, скорость – v, расстояние – S



Вход: v, t.
Шаг 1. $S \leftarrow v \cdot t$.
Результат: значение S.

? Программа линейная?

Вычислительные задачи

```
алг Путь
нач
   вещ v, t, S
   вывод "Введите скорость: "
   ввод v
   вывод "Введите время: "
   ввод t
   S:=v*t
   вывод "
кон
```

вещественные – могут быть с дробной частью!

переменные

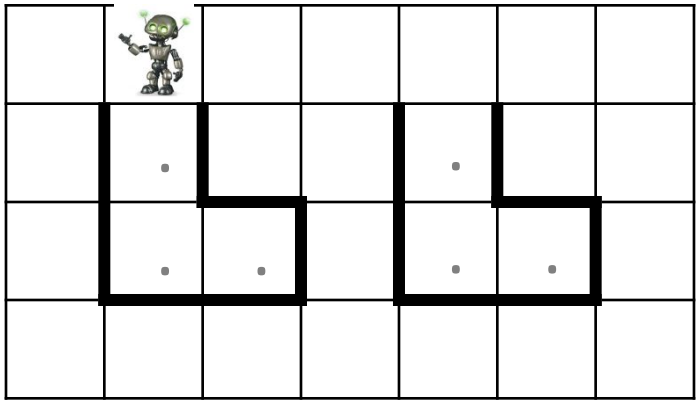
```
>> 13:21:47 - Новая программа - Начало выполнения
Введите скорость: 60
Введите время: 2
Расстояние: 120.0
>> 13:21:52 - Новая программа - Выполнение завершено
```

! Это решение «в буквах»!

Управление исполнителями

§ 34. Вспомогательные алгоритмы

Зачем это нужно?



? Какая ошибка?

```
алг Два сапога
нач
   Сапог
   вправо; вправо
   вправо
   Сапог
кон
```

ВЫЗОВ

ВЫЗОВ

вспомогательный алгоритм (**процедура**)

```
алг Сапог
нач
   вниз; закрасить
   вниз; закрасить
   вправо; закрасить
   влево; вверх; вверх
кон
```

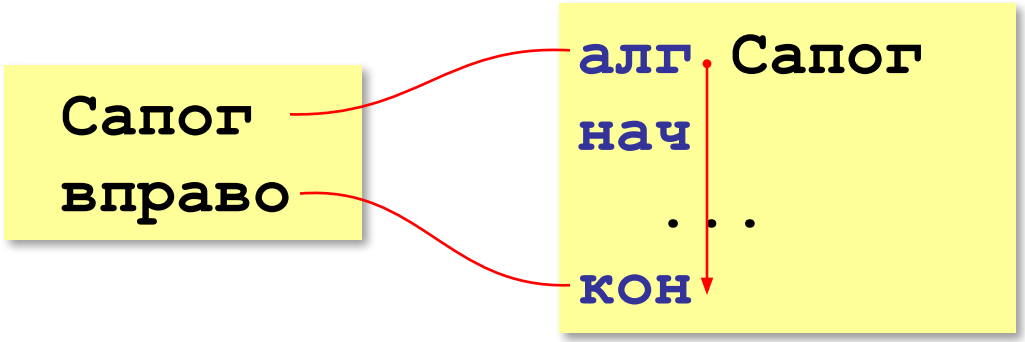
Вспомогательные алгоритмы

Вспомогательный алгоритм решает отдельную задачу и может быть использован при решении более сложных задач.

- чтобы он выполнился, его нужно вызвать:

```
Сапог
```

- **возврат:** после завершения его работы управление передаётся следующей команде вызывающего алгоритма



F7 – по шагам с входом в процедуры.

Два метода составления программ

1. Последовательное уточнение («сверху вниз»)

сначала:

```
алг Два сапога
нач
   Сапог
   вправо; вправо
   вправо
   Сапог
кон
```

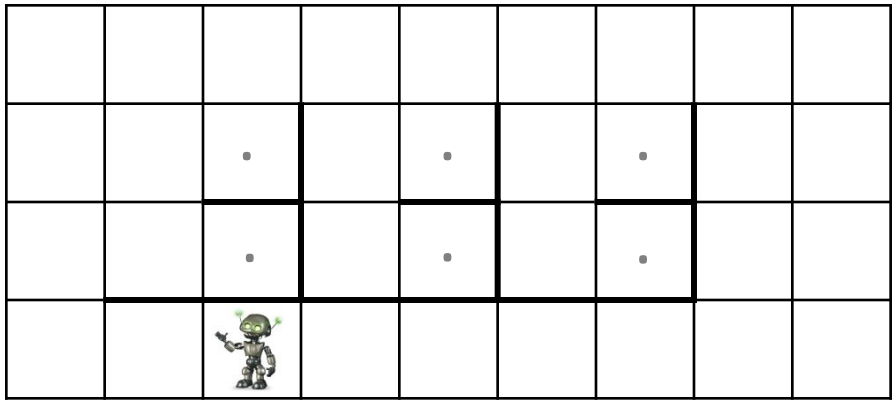
потом:

```
алг Сапог
нач
   ...
кон
```

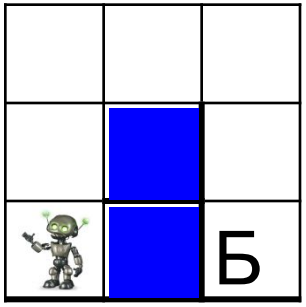
выделили части задачи, для которых будем писать процедуру

Два метода составления программ

2. «Снизу вверх» – сначала составить процедуры, потом собрать основную программу.



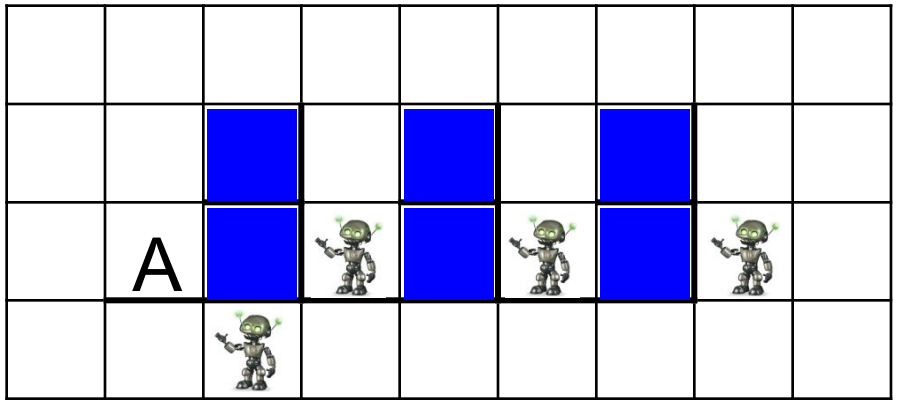
процедура:




```
алг Пара
нач
   вправо
   закрасить
   влево; вверх
   вправо
   закрасить
   вверх; вправо
   вниз; вниз
кон
```

Проектирование «снизу вверх»

Сборка основной программы:



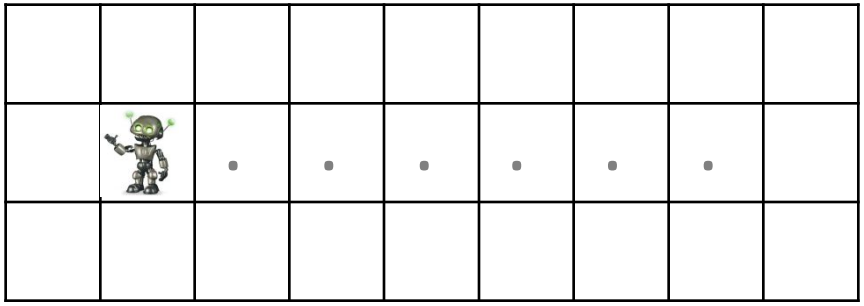

```
алг ТриПары
нач
   влево; влево
   вверх; вправо
   Пара
   Пара
   Пара
кон
```

привести в удобную начальную точку

Управление исполнителями

§ 35. Циклические алгоритмы

Что такое циклический алгоритм?



Цикл – это многократное выполнение некоторой последовательности действий.

начало цикла

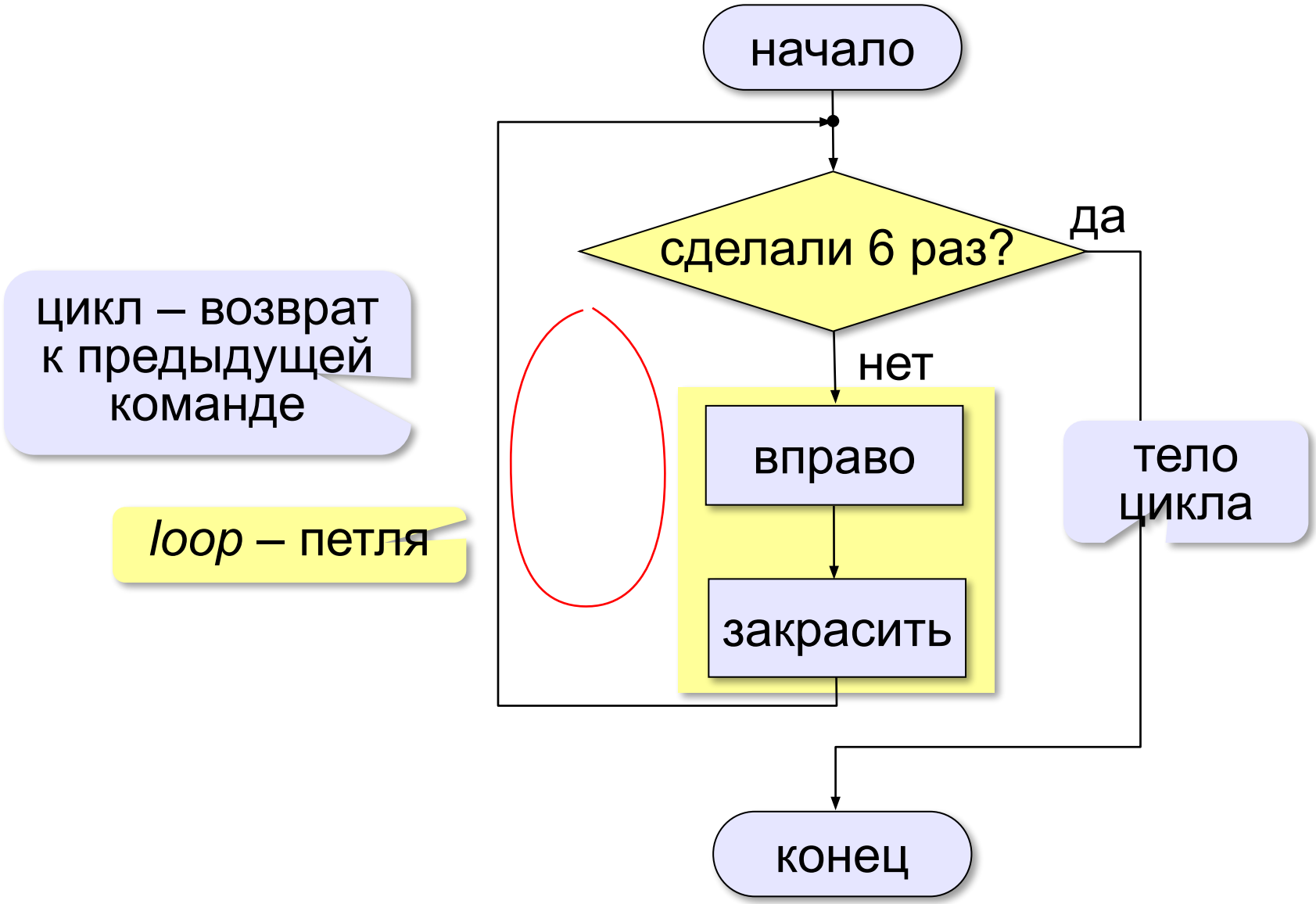
```
нц 6 раз
    вправо
    закрасить
кц
```

тело цикла

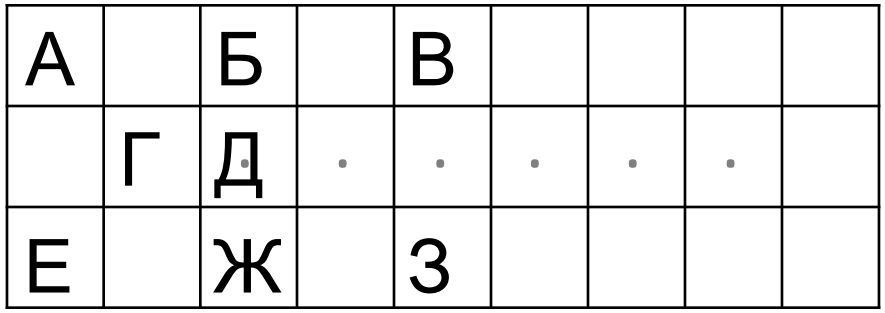
конец цикла

? А если ряд из 6000 клеток?

Блок-схема циклического алгоритма



Выбор начального положения



? Куда привести Робота перед началом цикла?

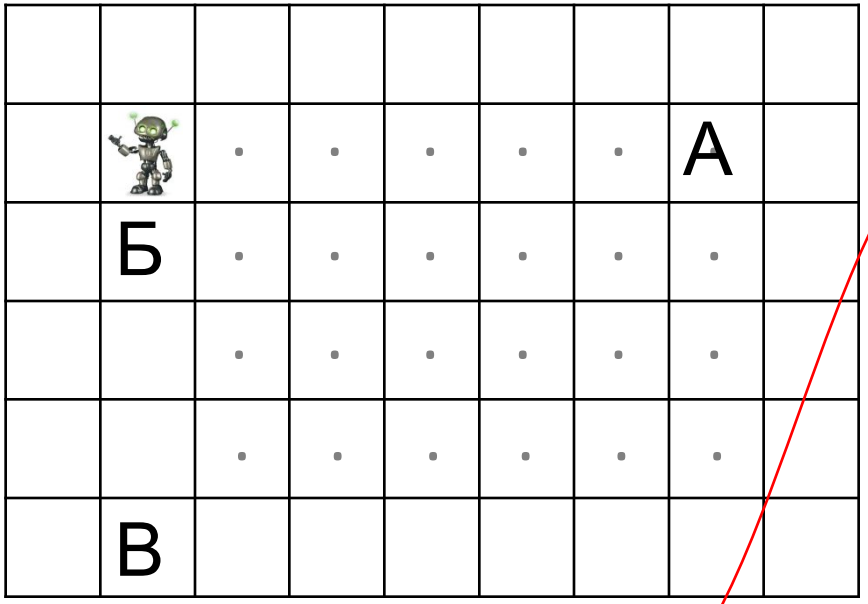
```
нц 6 раз
   вправо
   закрасить
кц
```

в клетку Г

```
нц 6 раз
   закрасить
   вправо
кц
```

в клетку Д

Вложенные циклы




```
нц 4 раз
  | закрасить ряд
  | к следующему ряду
кц
```

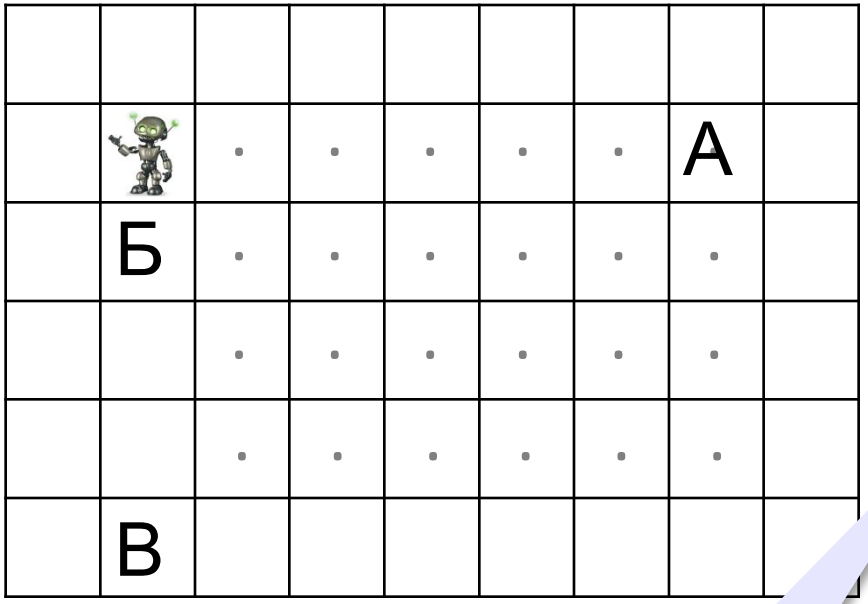
комментарии – пояснения для человека

```
| закрасить ряд
нц 6 раз
  вправо
  закрасить
кц
```

```
| к следующему ряду
вниз
нц 6 раз
  влево
кц
```

! Это циклы!

Вложенные циклы




```
нц 4 раз
  | закрасить ряд
    нц 6 раз
      вправо
      закрасить
    кц
  | к следующему ряду
  вниз
    нц 6 раз
      влево
    кц
кц
```

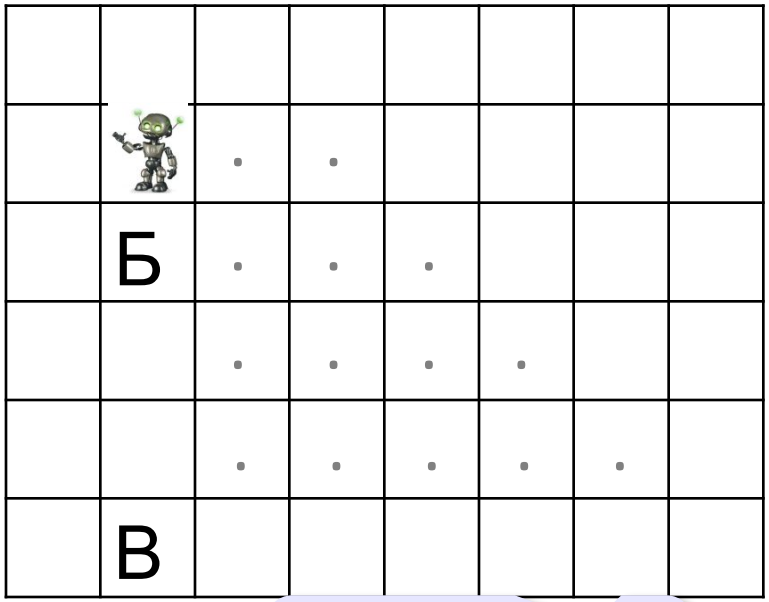
Вложенный цикл – это цикл внутри другого цикла.

? Где остановится Робот?

Управление исполнителями

§ 36. Переменные

Зачем нужны переменные?



длина ряда – величина **переменная**

N

```
N:=2
```

начальное значение

```
нц N раз
   вправо
   закрасить
кц
вниз
нц N раз
   влево
кц
```

? Как меняется N?

изменение с каждым шагом

```
N:=N+1
```

Использование переменных

```
цел N
N:=2
нц 4 раз
   нц N раз
      вправо
      закрасить
   кц
   вниз
   нц N раз
      влево
   кц
   N:=N+1
кц
```

объявление **переменной**

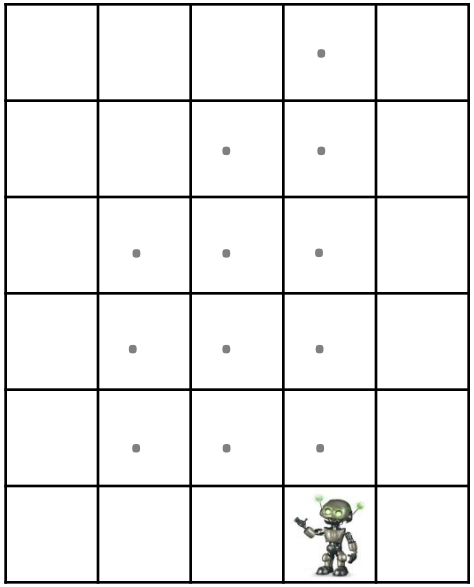
- тип переменной:
  - **цел** – целая
  - **вещ** – вещественная
  - **лог** – логическая
  - **лит** – строка символов
- допустимые операции
- сколько места выделить в памяти

следующий ряд на 1 клетку длиннее

? Что плохо?

Процедуры с параметрами

параметр



Если все ряды одинаковые (4 клетки):

```
алг Ряд
нач
  нц N раз
    вверх
    закрасить
  кц
кон
```

меняется!

! Это переменная!

? Что плохо?

добавить переход к началу следующего ряда!

Использование:

```
алг Трапеция
нач
  Ряд(5)  | при N = 5
  Ряд(4)  | при N = 4
  Ряд(3)  | при N = 3
кон
```

Управление исполнителями

§ 37. Циклы с условием

Что такое цикл с условием?

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*.

? Это цикл? ✓

? Число повторений известно? ✗

? Когда завершится? ***a < b***

? При каком условии продолжается? ***a ≥ b***

Логические команды

Подойти к стене:

? Как управлять с пульта?
Что нужно уметь определять?

Логическая команда — это запрос, на который исполнитель отвечает «да» или «нет».

`сверху стена`	`сверху свободно`
`справа стена`	`справа свободно`
`снизу стена`	`снизу свободно`
`слева стена`	`слева свободно`

логическое значение

Обратная связь — это данные, которые передаются от датчиков к управляющему устройству.

Цикл с условием

Подойти к стене:

```
алг До стены
нач
   нц пока слева свободно
      влево
   кц
кон
```

? А если нет стенки?

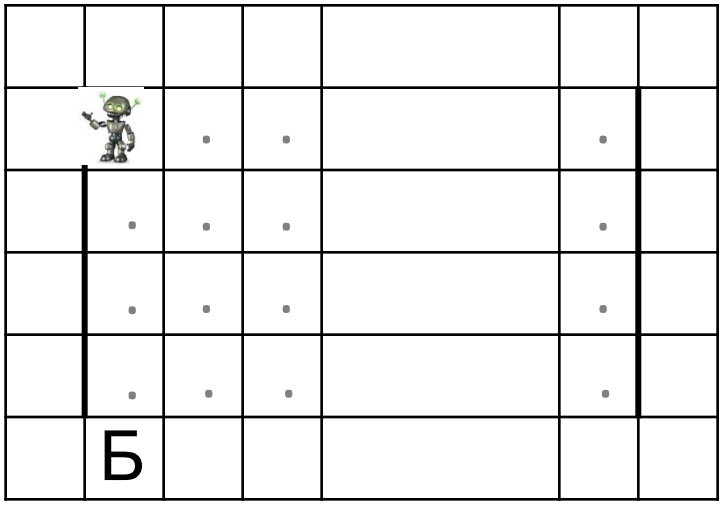
цикл выполняется, пока условие истинно

Зацикливание — это ситуация, когда цикл выполняется бесконечно.

? А если Робот рядом со стеной?

Вложенные циклы

4 ряда неизвестной длины:




```
нц 4 раз
   | подзадача 1
   | подзадача 2
кц
```

? Что это за подзадачи?

Закрасить ряд:

```
нц пока справа свободно
   вправо
   закрасить
кц
```

Перейти к следующему:

```
нц пока слева свободно
   влево
кц
вниз
```

? Что плохо?

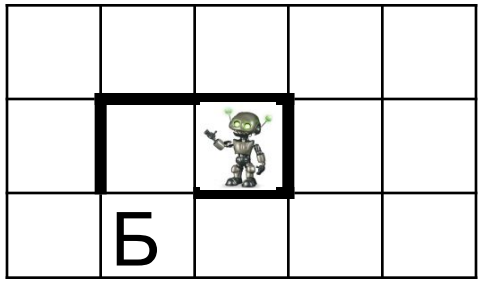
Управление исполнителями

§ 38. Разветвляющиеся алгоритмы

Что такое разветвляющийся алгоритм?

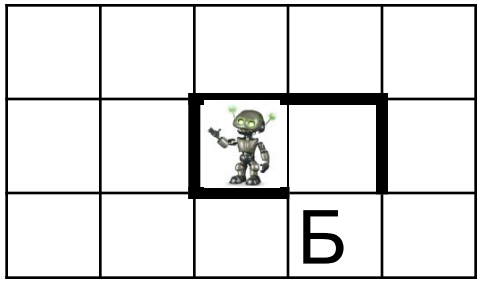
Привести Робота в клетку Б

а)




```
влево
вниз
```

б)




```
вправо
вниз
```

условие

? Как различить два случая?

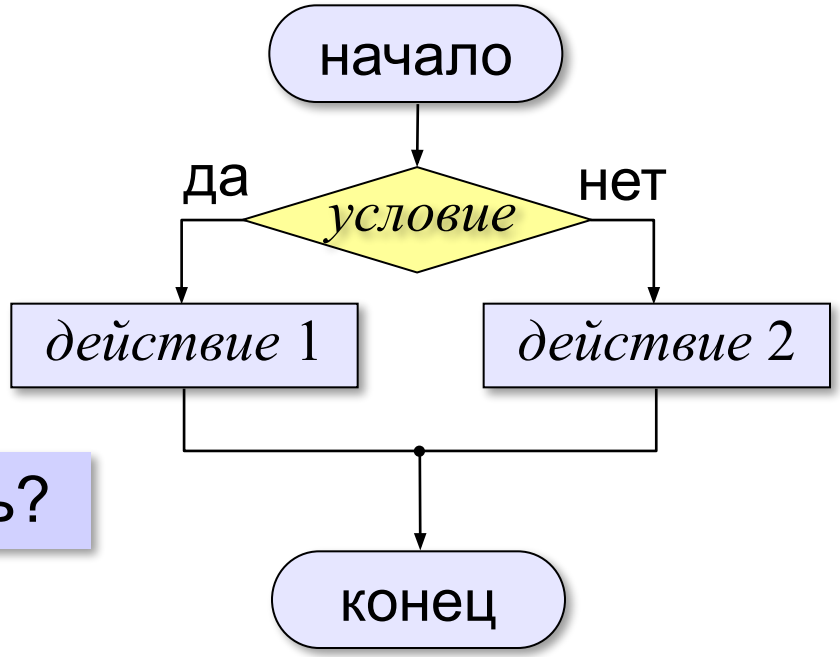
выполняется, если условие **ложно**

```
если слева свободно то
      влево
      вниз
   иначе
      вправо
      вниз
все
```

выполняется, если условие **истинно**

Разветвляющийся алгоритм

```
если слева свободно то
     влево
     вниз
  иначе
     вправо
     вниз
все
```

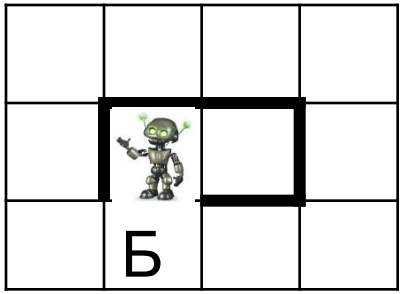


? Как улучшить?

```
если слева свободно то
   влево
иначе
   вправо
все
вниз
```

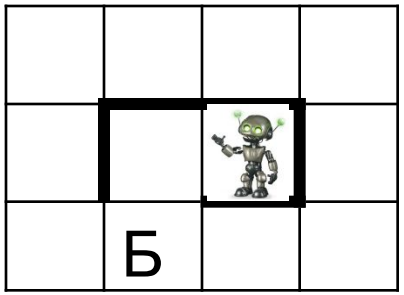
Ветвление в неполной форме

а)




```
вниз
```

б)

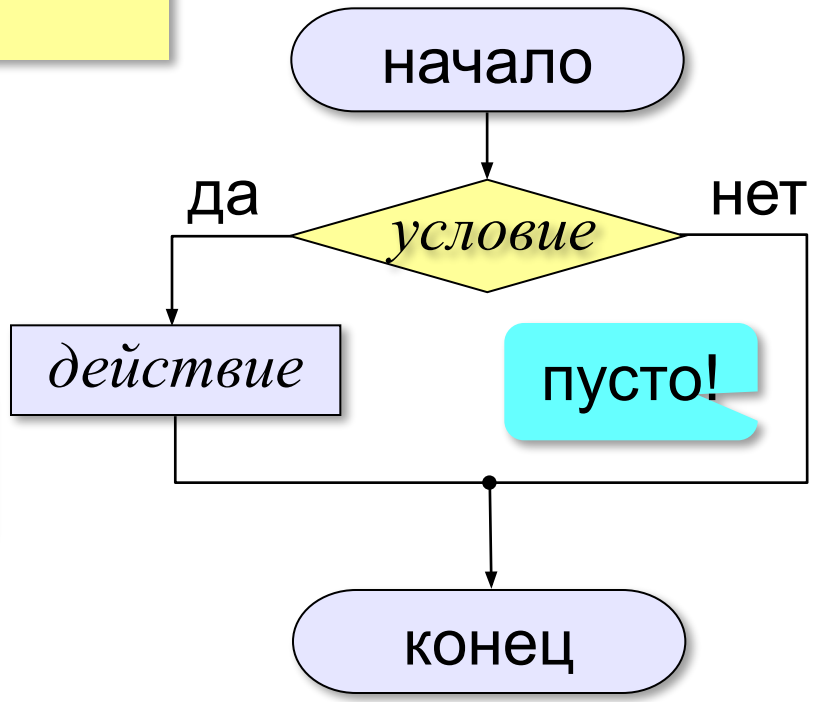



```
влево
вниз
```

? Как различить два случая?

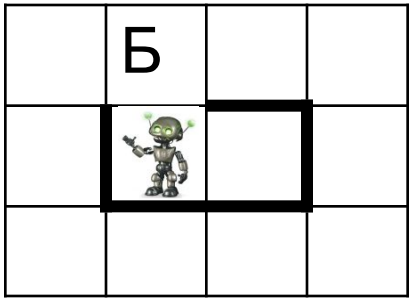
```
если слева свободно то
   влево
все
вниз
```

иначе ничего делать не нужно!



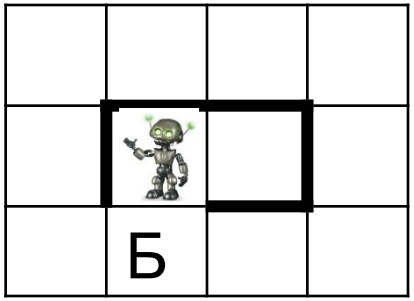
Вложенное ветвление

а)



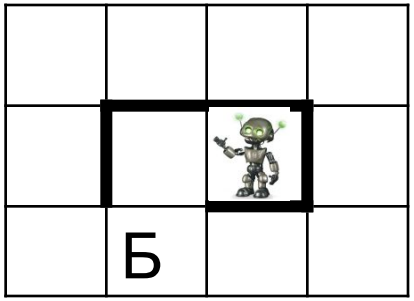

```
вверх
```

б)




```
вниз
```

в)




```
влево
вниз
```

? Как отличить *а* от *б* и *в*?

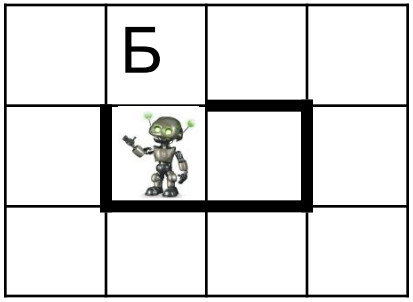
```
если сверху свободно то а)
```

? Как отличить *б* от *в*?

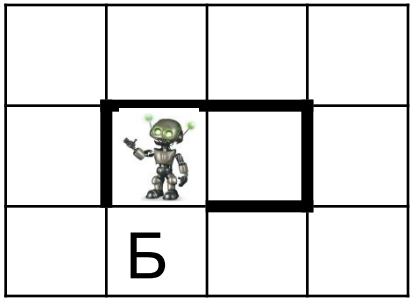
```
если снизу свободно то б)
```

Вложенное ветвление

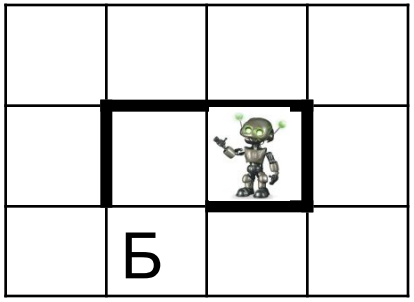
а)



б)



в)



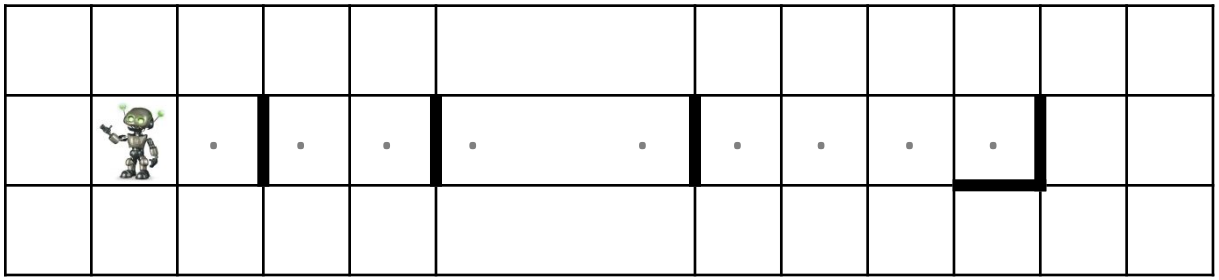

```
если сверху свободно то
    | работаем с задачей а
  иначе
    если снизу свободно то
      | работаем с задачей б
    иначе
      | работаем с задачей в
    все
все
```

вложенное ветвление!

Управление исполнителями

§ 39. Ветвления и циклы

Пример задачи



? Когда остановиться?

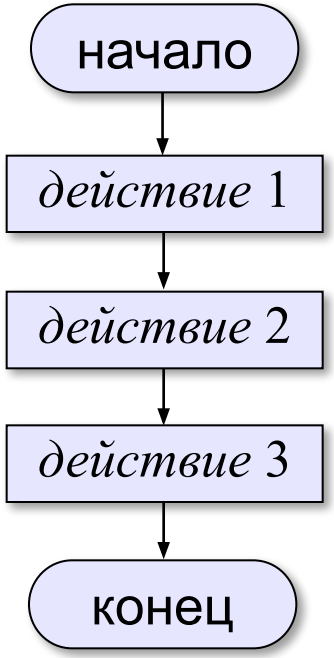
? Как различить два случая?

```
нц пока снизу свободно
  если справа свободно то
    вправо
  иначе
    вверх; вправо; вниз
  все
  закрасить
кц
```

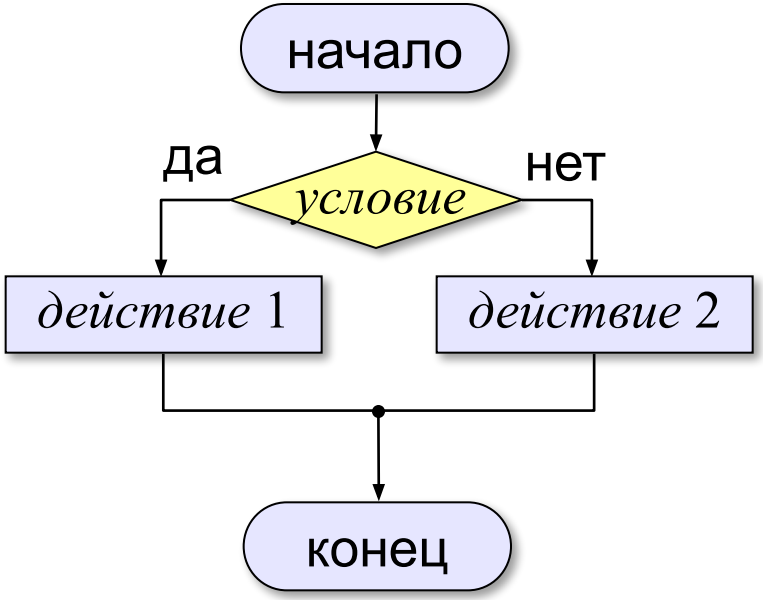
Базовые алгоритмические конструкции

Алгоритм решения любой задачи можно составить с помощью трёх базовых конструкций — **следования, ветвления и цикла.**

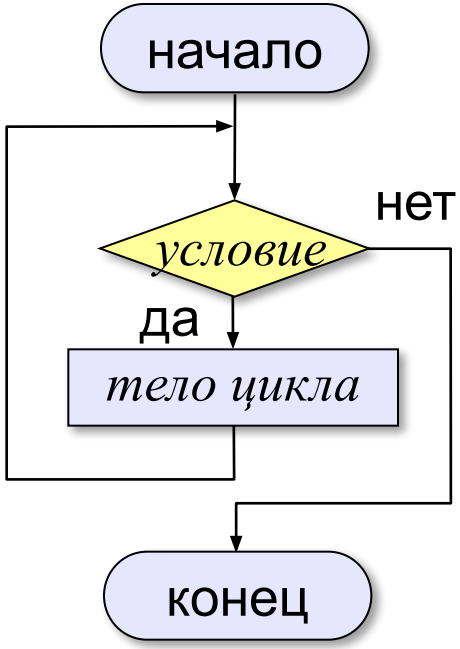
следование:



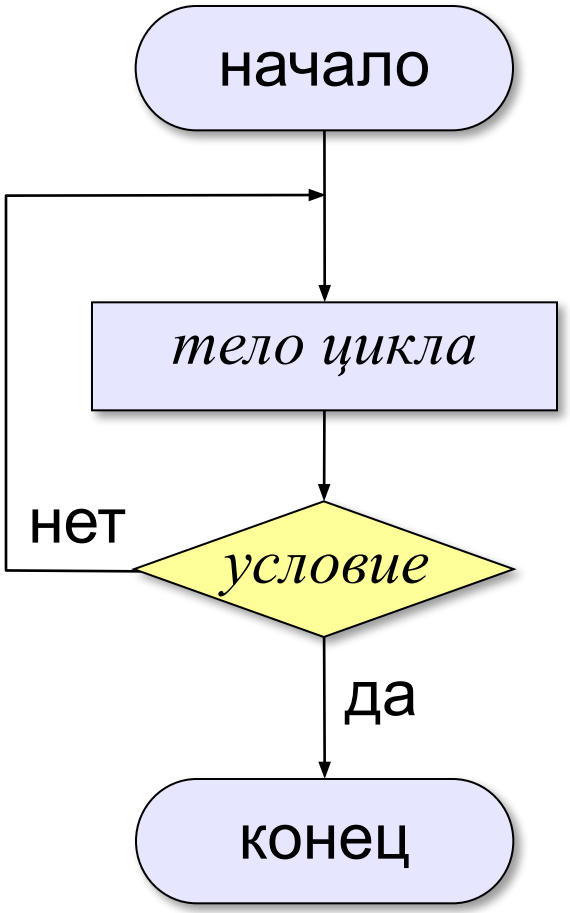
ветвление:



цикл:



Цикл с постусловием



? Может ли не выполниться ни разу?

? Что происходит, если условие истинно?

Пример: ввести число, которое обязательно должно быть положительным.

Анализ алгоритмов для Раздвоителя

1. **вычти 1**
2. **раздели на 2**

только для чётных!

Алгоритм 1:

```
нц пока N не ноль
  вычти 1
кц
```

? Что делают?

N → 0

Алгоритм 2:

```
нц пока N не ноль
  если N - чётное то
    раздели на 2
  иначе
    вычти 1
  все
кц
```

? Какой лучше?

- по длине?
- по скорости?

Конец фильма

ПОЛЯКОВ Константин Юрьевич
д.т.н., учитель информатики
ГБОУ СОШ № 163, г. Санкт-Петербург
kpolyakov@mail.ru

ЕРЕМИН Евгений Александрович
к.ф.-м.н., доцент кафедры мультимедийной дидактики и ИТО ПГГПУ, г. Пермь
eremin@pspu.ac.ru

Источники иллюстраций

1. nasa.gov
2. intel.com
3. иллюстрации художников издательства «Бином»
4. авторские материалы