

Исполнители и алгоритмы

Мы называли **исполнителем** любое устройство, которое способно выполнять определённый набор действий. Все команды, которые понимает исполнитель, образуют **систему команд** этого исполнителя. В каждый момент исполнитель находится в одном из возможных для него **состояний**. **Команды-приказы** изменяют состояние исполнителя. При выполнении **команды-запроса** исполнитель сообщает информацию о своём состоянии, состояние исполнителя не изменяется. **Начальным состоянием** исполнителя мы называем его состояние перед выполнением программы.

Система команд исполнителя *Водолей*

наполни А
наполни В
наполни С
перелей из А в В
перелей из В в С
перелей из А в С
перелей из В в А
перелей из С в В
перелей из С в А
вылей А
вылей В
вылей С

Система команд исполнителя *Удвоитель*

умножь на 2
прибавь 1

Ты знаком с исполнителями *Водолей*, *Удвоитель*, *Кузнечик* и *Робот*.

В системе команд *Водолея* команд-запросов нет. Состояние *Водолея* (при заданной вместимости сосудов) описывается тем, сколько мер воды в данный момент налито в каждый сосуд.

В системе команд *Удвоителя* команд-запросов нет. Состояние исполнителя *Удвоитель* описывается только одним числом — это число отображается на экране. В начальном состоянии *Удвоителя* на экране обычно отображается число 0, но в задачах бывают

и другие начальные состояния.

Исполнитель *Кузнечик* работает на числовой прямой:



Эта числовая прямая получилась объединением двух числовых лучей: один числовой луч построен от точки «0» вправо,

а другой числовой луч построен от точки «0» влево. На левом числовом луче числа помечены знаком «-». Скоро ты узнаешь, что такие числа называются отрицательными.

Кузнечик — это не один исполнитель, а много похожих исполнителей. *Кузнечики* различаются системами команд. У каждого *Кузнечика* в системе ровно две команды (вперёд... и назад...), но числа в командах (количество шагов, на которые прыгает *Кузнечик*) могут быть разными. Пример системы команд исполнителя *Кузнечик* приведён справа.

**Пример
системы команд
исполнителя
*Кузнечик***

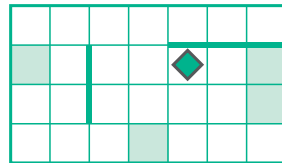
вперёд 3
назад 2

Кузнечик в начальном состоянии обычно стоит в точке «0», но в некоторых задачах может быть задано и другое начальное состояние.

Исполнитель *Робот* работает на прямоугольном поле, разбитом на клетки. Между некоторыми клетками поля могут стоять стены (на рисунке стены отмечены жирными линиями). Некоторые клетки поля могут быть закрашены. Сам *Робот* всегда занимает ровно одну клетку поля.

Исполнитель *Робот* умеет выполнять 17 команд: 5 команд-приказов и 12 команд-запросов. Мы изучили пока только команды-приказы: вверх, вниз, вправо, влево, закрасить.

По командам вверх, вниз, вправо, влево *Робот* перемещается в соседнюю клетку поля в указанном направлении. Если на пути *Робота* оказывается стена, команда не может быть выполнена: происходит отказ. В этом случае *Робот* команду не выполняет и выдаёт сообщение об ошибке. Например, из состояния, показанного на рисунке, *Робот* не может выполнить команду вверх.



По команде закрасить *Робот* закрашивает ту клетку, в которой стоит. Если эта клетка уже была закрашена, она останется закрашенной, т. е. команда будет выполнена, но никаких видимых изменений при этом не произойдёт.

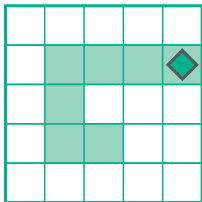
В быту человек чаще всего управляет устройствами-исполнителями **непосредственно**: даёт исполнителю команду, исполнитель выполняет её, затем человек смотрит на результат и даёт следующую команду и т. д. При этом человек выбирает следующую команду в зависимости от результата выполнения

предыдущей команды и состояния исполнителя после её выполнения.

Однако часто возникают ситуации, когда непосредственное управление неудобно или даже невозможно. В таких случаях можно составить план действий (иногда говорят «программу действий»), в котором будет указано, какие команды нужно давать исполнителю. Такой план называется **алгоритмом**. Алгоритм должен быть описан так, чтобы его мог выполнить компьютер. Для записи алгоритмов есть специальные языки, они называются **языками программирования**. Алгоритм, записанный на языке программирования, называют **программой**.

Выполняя программу, компьютер читает её и даёт команды исполнителю. Какие выдавать команды и когда, компьютер узнаёт из программы. В программе могут быть **команды-запросы**. В этом случае команда, которую компьютер даст исполнителю, будет зависеть от ответа исполнителя на запрос.

В учебнике для записи алгоритмов мы используем **школьный алгоритмический язык** и для краткости называем его **алгоритмический язык**. С этим языком ты познакомился в 5 классе. Например, вот алгоритм крючок и поле *Робота* после выполнения этого алгоритма:



алг крючок

| | | |
|-------------|--|-----------------|
| дано | | слева от Робота |
| | | не меньше 3 |
| | | клеток поля, |
| | | снизу не меньше |
| | | 2 клеток |
| надо | | Робот нарисовал |
| | | крючок и |
| | | вернулся назад |

нач

закрасить
влево
закрасить
влево
закрасить
влево
закрасить
вниз
закрасить
вниз
закрасить
вправо
закрасить
вправо
вверх
вверх

кон