Глава 3
ОСНОВНЫЕ АЛГОРИТМИЧЕСКИЕ КОНСТРУКЦИИ
§ 11. Алгоритмы и исполнители

Большинству исполнителей для выполнения алгоритма нужны команды, записанные на формальном языке. Для того чтобы взаимодействие человека и технического устройства было плодотворным, человек обучается формальным языкам, например языкам программирования. Другой путь — совершенствовать исполнителей так, чтобы они понимали естественный язык человека. Человечество пока ещё в начале такого пути, однако уже сегодня существуют исполнители с голосовым управлением.

Первыми бытовыми устройствами с голосовым управлением были стиральные машины и сотовые телефоны. В настоящее время голосовое управление имеют бытовые компьютеры, автомобили, музыкальные центры, кондиционеры, лифты и др.

11.1. Понятие алгоритма

Вспомним некоторые понятия, с которыми вы познакомились в 6-м классе.

Алгоритм — понятная и конечная последовательность точных действий (команд), формальное выполнение которых позволяет получить решение поставленной задачи.

Исполнитель алгоритма — человек, группа людей или техническое устройство, которые понимают команды алгоритма и умеют правильно их выполнять.

Система команд исполнителя — команды, которые понимает и может выполнить исполнитель.

Любой исполнитель имеет ограниченную систему команд. Все они разделяются на группы:

1) команды, которые непосредственно выполняет исполнитель;

2) команды, меняющие порядок выполнения других команд исполнителя.

Компьютер является универсальным исполнителем.

Запись алгоритма в виде последовательности команд, которую может выполнить компьютер, называют **программой.**

Выделяют следующие способы представления алгоритмов (пример 11.1):

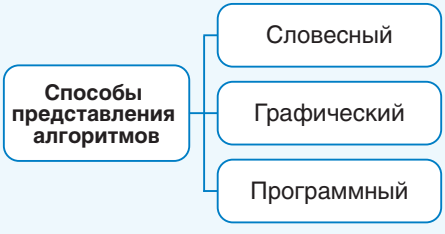
- **словесный** — описание алгоритма средствами естественного языка с точной и конкретной формулировкой фраз;
- **графический (блок-схема)** — графическое изображение команд алгоритма с использованием блоков в виде геометрических фигур, отрезков или стрелок, соединяющих эти блоки и указывающих на порядок выполнения команд;
- **программный** — запись алгоритма в виде программы.

На практике могут применяться комбинации вышеперечисленных способов записи алгоритмов, а также другие способы представления: **табличный**, описание алгоритма с помощью **математических формул**, **псевдокод** (описание структуры алгоритма на частично формализованном языке).

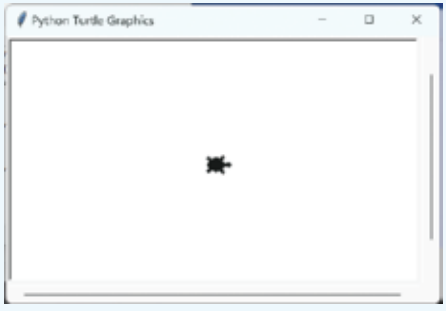
Голосовые помощники Google Assistant и Алиса, созданная компанией Яндекс, могут осуществлять поиск, поддерживать беседу, играть в игры, помогают управлять системой «Умный дом» и многое другое.

Пример 11.1. Способы представления алгоритмов.

На разных этапах решения задачи могут применяться разные способы описания алгоритма. На этапе обсуждения используются словесные и формульные способы. На этапе проектирования рекомендуется использовать графическое представление. Для проверки результатов возможно табличное представление. Программный способ используют на этапе непосредственного применения для решения прикладных задач.



Пример 11.2. Среда обитания исполнителя *Черепаха*.



Пример 11.3. Основные команды исполнителя *Черепаха*.

Команда	Действие
`penup()`	Не оставлять след при движении
`pendown()`	Оставлять след при движении
`forward(X)`	Пройти вперёд X пикселей
`backward(X)`	Пройти назад X пикселей
`left(X)`	Повернуться налево на X градусов
`right(X)`	Повернуться направо на X градусов

11.2. Исполнитель *Черепаха*

В прошлом году вы познакомились с исполнителем *Черепаха*, который умеет строить рисунки и чертежи на координатной плоскости.

Среда обитания исполнителя *Черепаха* — координатная плоскость (пример 11.2). Исходное положение *Черепахи* — точка с координатами (0, 0), перо опущено.

Основные команды исполнителя *Черепаха* представлены в примере 11.3.

Пример 11.4. Прямоугольный участок, длина которого в 2 раза больше ширины, огородили забором длиной 120 метров. Определите длину и ширину участка. Составьте алгоритм и напишите программу, выполнив которую исполнитель *Черепаха* построит чертёж забора этого участка. Масштаб: 1 м соответствует 10 пикселям.

Словесное описание алгоритма вычисления длины и ширины участка:

1. Длина участка в два раза больше ширины, поэтому в сумме длина и ширина составят три одинаковых части. Забор огораживает участок по периметру. Периметр прямоугольника равен

удвоенной сумме длины и ширины, следовательно, он равен шести одинаковым частям.

2. Значение ширины участка получается делением длины забора на шесть частей.

3. Для нахождения значения длины участка удваиваем значение ширины.

Математическая запись действий:

1) $(1 + 2) \cdot 2 = 6$ (частей).

2) $120 : 6 = 20$ (м).

3) $20 \cdot 2 = 40$ (м).

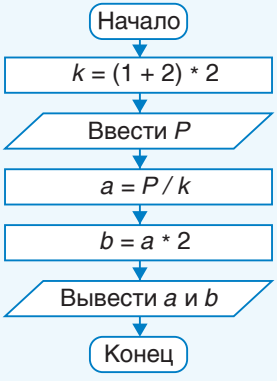
Словесное описание алгоритма рисования участка в масштабе 1 : 10:

```
Установить размеры поля 500 × 300
Поднять перо
В точку (-200, 100)
Опустить перо
Вперёд (400)
Направо (90)
Вперёд (200)
Направо (90)
Вперёд (400)
Направо (90)
Вперёд (200)
```

11.3. Алгоритмическая конструкция *следование*

Существует большое количество алгоритмов, в которых все команды выполняются последовательно одна за другой в том

Пример 11.4. Блок-схема алгоритма.



Программа для исполнителя *Черепаха*:

```
import turtle
turtle.shape('turtle')
turtle.setup(500, 300)
turtle.penup()
turtle.setpos(-200, 100)
turtle.pendown()

turtle.forward(400)
turtle.right(90)
turtle.forward(200)
turtle.right(90)
turtle.forward(400)
turtle.right(90)
turtle.forward(200)
```

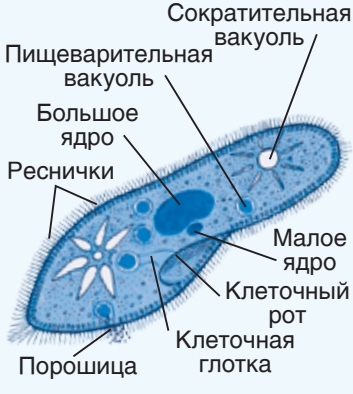
Результат работы программы:

Пример 11.5. Алгоритм приготовления бутерброда.

1. Отрезать ломтик батона.
2. Положить на батон лист салата.
3. Отрезать кусочек копчёного мяса.
4. Положить мясо на лист салата.
5. Отрезать кусочек помидора.
6. Положить помидор на мясо.

Пример 11.6. Алгоритм выполнения лабораторной работы по биологии «Строение инфузории туфельки».

1. Рассмотреть под микроскопом внешний вид и внутреннее строение инфузории туфельки.
2. Зарисовать инфузорию туфельку и обозначить названия её органоидов.
3. Подвести итог работы.



порядке, в котором они записаны. В таких алгоритмах отсутствуют команды, меняющие порядок выполнения других команд. Программы такого вида вы составляли в прошлом году для исполнителя *Черепаха*.

Алгоритмическая конструкция *следование* — последовательность команд алгоритма, которые выполняются в том порядке, в котором они записаны.

Алгоритмическая конструкция *следование* отображает естественный, последовательный порядок выполнения действий в алгоритме.

Следование использовалось в примере 11.4 на с. 73, в котором описывался алгоритм вычисления длины и ширины участка и построения прямоугольника исполнителем *Черепаха*.

Алгоритмическая конструкция *следование* представлена в примерах 11.5 и 11.6.

11.4. Вспомогательные алгоритмы

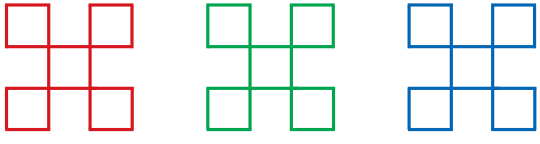
Часто в одной программе нужно рисовать одно и то же изображение несколько раз. Получе-

ние такого изображения удобно оформлять в виде вспомогательного алгоритма, который можно использовать нужное число раз.

Вспомогательный алгоритм — это алгоритм, целиком используемый в составе другого алгоритма.

Вспомогательный алгоритм решает некоторую подзадачу основной задачи. Вызов вспомогательного алгоритма в программе заменяет несколько команд одной. В языке Python *вспомогательные алгоритмы* записываются в виде функций. Формат описания функции приведён в примере 11.7.

Пример 11.8. Написать программу, выполнив которую исполнитель *Черепаха* нарисует следующее изображение:



Данный рисунок состоит из трёх одинаковых фигур. Для рисования одной из них можно оформить вспомогательный алгоритм:

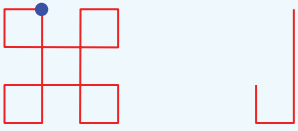
Пример 11.7. Формат описания функции.

Пример 11.8. Программа для исполнителя *Черепаха*.

```
import turtle

turtle.shape('turtle')
turtle.setup(670, 240)

def chast():
    turtle.forward(150)
    turtle.right(90)
    turtle.forward(50)
    turtle.right(90)
    turtle.forward(50)
    turtle.right(90)

def p(x, y):
    turtle.penup()
    turtle.setpos(x, y)
    turtle.pendown()
    turtle.setheading(270)

def figura(x, y, c):
    p(x, y)
    turtle.color(c)
    chast()
    chast()
    chast()
    chast()

figura(-200, 100, 'red')
figura(0, 100, 'green')
figura(200, 100, 'blue')
```

Пример 11.8. *Продолжение.*

Начальная точка и часть фигуры:



Результат выполнения программы:

Над проектом по созданию программы может работать несколько человек (или десятков человек). Каждый из членов коллектива делает часть своей работы и оформляет её как отдельный вспомогательный алгоритм.

функцию `figura(x, y, c)`. Параметры `x`, `y` и `c` определяют координаты точки начала рисования и цвет фигуры. Каждая фигура в свою очередь состоит из четырёх одинаковых частей. Рисование одной части выполняет функция `chast()`. Для перемещения *Черепахи* в позицию с координатами (x, y) и поворота на 270° используется вспомогательный алгоритм `p(x, y)`.

Описание основного алгоритма:

```
Рисование фигуры
(-200, 100, 'red')
Рисование фигуры
(0, 100, 'green')
Рисование фигуры
(200, 100, 'blue')
```

1. Что такое алгоритм?
2. Какие способы записи алгоритмов вам известны?
3. На каком этапе решения задачи используется словесный способ описания алгоритма?
4. Что называют алгоритмической конструкцией *следование*?
5. Какие алгоритмы называются вспомогательными?
6. Для чего нужны вспомогательные алгоритмы?
7. Как описывается вспомогательный алгоритм на языке Python?

Упражнения

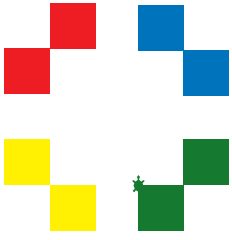
1 Определите, какие рисунки получатся после выполнения *Черепахой* следующих программ. Изобразите рисунки в тетради и проверьте правильность своих действий, выполнив программы на компьютере.

а)
```
import turtle
turtle.shape('turtle')
turtle.setup(500,500)
turtle.penup()
turtle.setpos(-100,-100)
turtle.pendown()
turtle.setheading(108)
turtle.forward(200)
turtle.right(72)
turtle.forward(200)
turtle.right(72)
turtle.forward(200)
turtle.right(72)
turtle.forward(200)
turtle.right(72)
turtle.forward(200)
turtle.right(72)
```

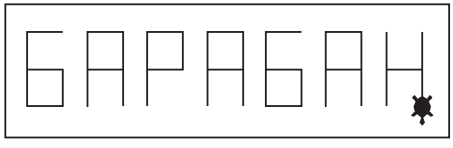
б)
```
import turtle
turtle.shape('turtle')
turtle.setup(500,500)
def chast():
    turtle.forward(200)
    turtle.right(90)
    turtle.forward(50)
    turtle.left(90)
    turtle.forward(50)
    turtle.right(90)
turtle.penup()
turtle.setpos(-100,150)
turtle.pendown()
chast()
chast()
chast()
chast()
```

2 Напишите для исполнителя *Черепаха* программы получения следующих изображений.

а)



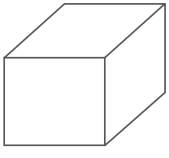
б)



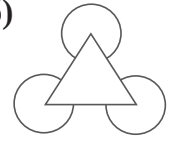
3 Придумайте свои рисунки и составьте программы для их рисования с помощью исполнителя *Черепаха*.

4* Исполнитель *Чертёжник* может рисовать только отрезки. Проанализируйте рисунки. Какие из них сможет выполнить исполнитель *Чертёжник*? Почему? Все ли из этих рисунков сможет нарисовать исполнитель *Черепаха*?

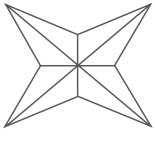
а)



б)



в)



г)