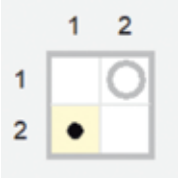
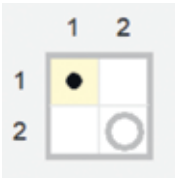
2. *Робот* находится в какой-то клетке поля размером 2 × 2. Он должен закрасить клетку в углу, противоположном начальному.





§ 16. Использование основных алгоритмических конструкций для исполнителя *Робот*

Последовательное выполнение команд в программе определяется структурой *следование*. Для организации повторяющихся действий в алгоритме используется команда *цикл*. Команда *ветвление* позволяет выполнять одну или другую последовательность команд в зависимости от истинности условия.

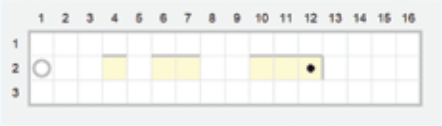
Следование, *цикл* и *ветвление* — **базовые алгоритмические конструкции.** Используя эти конструкции как элементы некоего «конструктора», можно составлять и разрабатывать любые алгоритмы.

Команды *цикл* и *ветвление* управляют порядком выполнения других команд в программе и относятся к командам управления. Использование алгоритмической

Перед человеком постоянно возникают разнообразные задачи, для которых существуют различные алгоритмы решения. При всём многообразии алгоритмов для их записи достаточно трёх алгоритмических конструкций (структур): *следование*, *цикл*, *ветвление*.

Это положение было выдвинуто в середине 1970-х гг. учёным Эдсгером Вибе Дейкстрой (1930–2002 гг.). Его труды оказали влияние на развитие информатики и информационных технологий. Дейкстра являлся одним из разработчиков концепции структурного программирования, участвовал в разработке языка программирования Алгол. Известен своими разработками в области математической логики и теории графов.

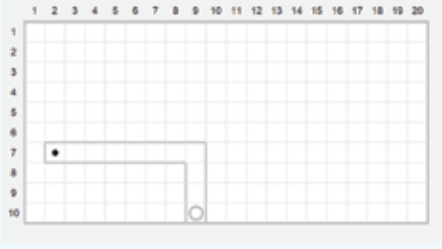
Пример 16.1. Одна из возможных начальных обстановок.



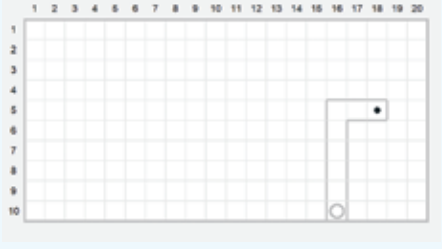
Программа для исполнителя *Робот*:

```
from pyrob.api import *
@task
def prim_16_1():
    while not wall_right():
        if wall_up():
            fill_cell()
        move_right()
    if wall_up():
        fill_cell()
run_tasks()
```

Пример 16.2. Одна из возможных начальных обстановок.



Другая обстановка:



конструкции *следование* предполагает отсутствие управляющих конструкций.

Рассмотрим примеры алгоритмов, содержащих несколько алгоритмических конструкций.

Пример 16.1. Написать программу для закрашивания некоторых клеток на поле. Окончание движения *Робота* определяет стена справа. По пути нужно закрасить те клетки, над которыми есть стена.

Для решения задачи *Робот* при движении вправо должен проверять каждую клетку на своём пути. Если условие «сверху стена» выполняется, то *Робот* закрашивает эту клетку. После проверки клетки *Робот* сдвигается вправо. Эти действия выполняются в цикле, пока справа нет стены.

После цикла необходима команда *ветвление*, так как для крайней клетки поля условие «нет стены справа» является ложным и клетка в цикле не закрашивается.

В этой задаче внутри конструкции цикла используется конструкция *ветвление*.

Пример 16.2. *Робот* должен дойти до конца «коридора» переменного размера. В «коридоре» может быть поворот влево или вправо.

Для решения задачи *Робот* сначала перемещается вверх, до тех пор, пока истинно условие «нет стены сверху». Стена, появившаяся сверху, означает, что начался поворот «коридора». Если слева нет стены, то в «коридоре» надо поворачивать влево, иначе в «коридоре» *Робот* поворачивает вправо. Дальше он двигается в выбранном направлении, пока не дойдёт до стены.

При решении данной задачи используется сначала конструкция *цикл*, а затем конструкция *ветвление*. В команде *ветвление* каждая последовательность команд представляет собой цикл.

Пример 16.3*. *Роботу* необходимо переместиться из верхнего левого угла поля в нижний левый угол. При этом на поле присутствуют стены. *Робот* начинает движение вправо до правой границы поля и спускается на одну клетку вниз. Затем движение продолжается влево до левой границы поля, после — спуск на одну клетку. Эти действия *Робот* должен повторить 4 раза.

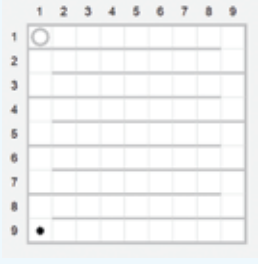
В данной задаче внутри цикла с параметром используются два других цикла с предусловием.

Пример 16.2. *Продолжение.*

Программа для исполнителя *Робот*:

```
from pyrob.api import *
@task
def prim_16_ 2():
    while not wall_up():
        move_up()
    if wall_left():
        while not wall_right():
        move_right()
    else:
        while not wall_left():
        move_left()
run_tasks()
```

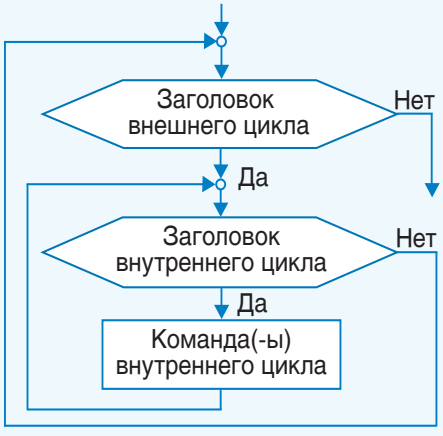
Пример 16.3*. Начальная обстановка.



Программа для исполнителя *Робот*:

```
from pyrob.api import *
@task
def prim_16_ 3():
    for i in range (4):
    while wall_down():
        move_right()
        move_down()
    while wall_down():
        move_left()
        move_down()
run_tasks()
```

Пример 16.4. Блок-схема вложенных циклов.



Структуру, когда внутри одного цикла выполняется другой цикл, называют **вложенным циклом.**

Один из возможных вариантов вложенных циклов представлен на блок-схеме в примере 16.4.

Как видно из примеров, базовые алгоритмические структуры можно комбинировать друг с другом так, как этого требует алгоритм решения задачи.

1. Назовите базовые алгоритмические конструкции.
2. Какие базовые алгоритмические конструкции относят к элементам управления?
3. Приведите примеры использования базовых алгоритмических конструкций.
4*. Что такое вложенный цикл?

Упражнения

1 Объясните, какие алгоритмические конструкции используются в приведённых ниже программах. Нарисуйте блок-схемы данных алгоритмов. Предложите пример начальной обстановки, в которой алгоритм выполнится корректно.

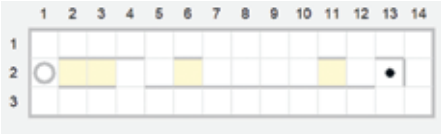
а)

```
from pyrob.api import *
@task
def upr_16_1_a():
    while wall_left():
        fill_cell()
        move_down()
run_tasks()
```

б)

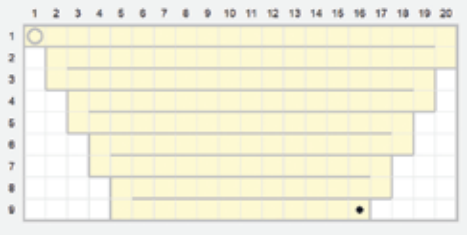
```
from pyrob.api import *
@task
def upr_16_1_b():
    while cell_is_filled():
        if not wall_left():
            move_left()
run_tasks()
```

2 Для решения задачи upr_16_2 Миша написал программу, но она работает неправильно. Какие ошибки допустил Миша?



```
from pyrob.api import *
@task
def upr_16_2():
    while  wall_right():
        if wall_up() or wall_down():
            fill_cell()
        move_right()
    if wall_up() and wall_down():
        fill_cell()
run_tasks()
```

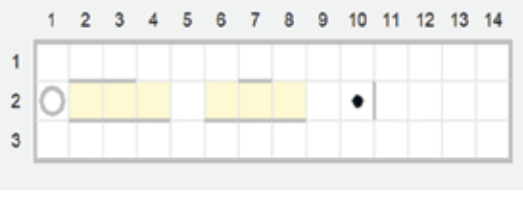
3 Заполните пропуски (многоточие на жёлтом фоне) в программе решения задачи upr_16_3 так, чтобы она работала верно.

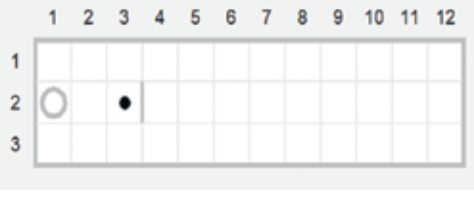


В строке поля *Робота* может быть произвольное количество клеток.

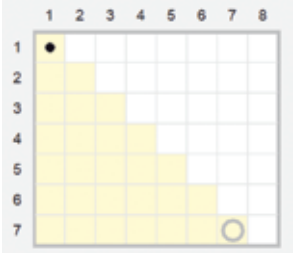
```
from pyrob.api import *
@task
def  upr_16_3():
     for  i  in  range  (4):
          fill_cell()
          while  ...:
               move_right()
               fill_cell()
          move_down()
          fill_cell()
          while  ...:
               move_left()
               fill_cell()
     ...
     while  not  wall_right():
          move_right()
          fill_cell()
run _tasks()
```

4 Решите задачу upr_16_4, используя внутри цикла команду *ветвление*.





5* Решите задачу `upr_16_5`, используя вложенные циклы. Размер поля заранее не известен.



6* Придумайте задачу для исполнителя *Робот*, в которой будут использоваться различные алгоритмические конструкции.

§ 17. Язык программирования Python

Компьютер (англ. computer — *вычислитель*) — устройство (или система), способное выполнять заданную чётко определённую изменяемую последовательность операций. Чаще всего это операции численных расчётов.

Гвидо ван Россум приступил к созданию языка Python в декабре 1989 г. в центре математики и информатики в Нидерландах. Ван Россум является основным автором Python. Он продолжал выполнять центральную роль в принятии решений относительно развития языка вплоть до 12 июля 2018 г.

Желание упростить расчёты присуще человеку с древних времён. Создавая различные приспособления для счёта, человек прошёл долгий путь. Сегодняшний компьютер способен выполнять сотни миллионов операций в секунду. Для решения вычислительных задач требуется сначала составить алгоритм, а затем записать его в виде программы, используя какой-либо язык программирования.

Язык программирования устанавливает набор правил, определяющих внешний вид программы и действия, которые выполнит исполнитель под её управлением.