**§ 20. Программирование циклических алгоритмов**

**Как организовать цикл?**

Допустим, мы хотим вывести 5 раз на экран слово «привет». Можно, конечно, записать 5 одинаковых команд:

**print( "привет" )**

**print( "привет" )**

**print( "привет" )**

**print( "привет" )**

**print( "привет" )**

Но что если нужно будет сделать какие-то действия 1000 или 1000000 раз? В этом случае можно организовать цикл. Про-стейший цикл, нужный нам в этой задаче, мы хотели бы запи-сать так:

**# сделай 5 раз:**

**print( "привет" )**

К сожалению, в языке Python (как и во многих других язы-ках программирования) такого цикла нет3. Однако можно лег-ко запрограммировать те же действия немного по-другому. Да-вайте разберёмся, как можно организовать цикл в языке Python.

Вы знаете, что программа выполняется автоматически. И при этом в любой момент нужно знать, сколько раз уже выпол-нен цикл и сколько ещё осталось выполнить. Для этого необхо-димо использовать ячейку памяти (переменную). В ней можно, например, запоминать, сколько раз цикл уже был выполнен. Такую переменную целого типа часто называется *счётчиком*.

Сначала в переменную-счётчик записывают ноль (цикл ещё не выполнен ни разу), а после каждого повторения цикла уве-личивают значение счётчика на единицу:

**count = 0**

**while count < 5: # заголовок цикла**

**print( "привет" )**

**count += 1 # увеличение счётчика**

В этой программе используется новое служебное слово *while*, по-сле которого записано условие.

*Найдите и запишите в тетрадь перевод английских слов while, count.*

Поскольку цикл связан с повторением, циклические алго-ритмы называют *итерационными* (от лат. *iteratio* – повторе-ние). Каждое выполнение тела цикла называют *итерацией*.

Все операторы, которые выполняются в цикле (они называ-ются *телом цикла*) сдвигаются вправо на одинаковое число по-зиций, так же как и в условном операторе. Этот приём позволя-ет обойтись без операторных скобок, ограничивающих тело цик-ла в других языках программирования.

Нам нужно выполнять цикл 5 раз, то есть пока счётчик не станет равен 5. Об этом говорит *заголовок цикла*

**while count < 5:**

Его можно прочитать как «делай, пока *count* < 5».

После каждой итерации цикла переменная *count* увеличи-вается на 1 – цикл выполнен ещё один раз. Если программист забудет написать этот оператор, произойдёт *зацикливание*: про-грамма никогда не остановится, потому что условие *count* < 5 никогда не станет ложным.

Цикл можно построить и по-другому: сразу записать в счёт-чик нужное количество итераций, и после каждой итерации цикла *уменьшать* счётчик на 1. Тогда цикл должен закончить-ся при нулевом значении счётчика.

*Запишите в тетрадь цикл со счётчиком, который уменьшается от нужного значения до нуля.*

Этот вариант несколько лучше, чем предыдущий, посколь-ку счётчик сравнивается с нулём, а такое сравнение выполняет-ся в процессоре автоматически.

**Циклы с условием**

Цикл, в котором проверка условия выполняется при входе (перед выполнением очередного шага) называется *циклом с предусловием*, то есть циклом с предварительной проверкой ус-ловия. Перед тем, как начать выполнение цикла, мы проверя-ем, нужно ли это делать вообще. Это можно сравнить с такой ситуацией: перед тем, как прыгнуть в бассейн, нужно прове-рить, есть ли в нём вода.

Все циклы, записанные в начале параграфа – это циклы с предусловием. У них есть два важных свойства:

цикл не выполнится ни разу, если условие в самом начале ложно;

как только нарушается условие в заголовке цикла, его рабо-та заканчивается.

Рассмотрим ещё одну задачу, которая решается с помощью цикла с условием. Требуется ввести с клавиатуры натуральное число и найти сумму цифр его десятичной записи. Например, если ввели число 123, программа должна вывести сумму 1+2+3 = 6.

Сначала составим алгоритм решения это задачи. Предпо-ложим, что число записано в переменной *N*. Нам нужно как-то разбить число на отдельные цифры.

*Запишите команды, с помощью которых можно:*

*записать в переменную d последнюю цифру числа, находящегося в переменной N;*

*отбросить последнюю цифру числа, находящегося в переменной N*;

*добавить значение переменной d к значению, находя-щемуся в переменной s.*

Вспомним, что остаток от деления числа на 10 равен по-следней цифре его десятичной записи. Запишем эту цифру в переменную **digit**:

**digit = N % 10**

Сумму цифр будем хранить в целой переменной **summa**. В самом начале, пока ни одну цифру ещё не обработали, значение этой переменной равно нулю:

**summa = 0**

Для того чтобы добавить к предыдущей сумме новую цифру, нужно заменить значение переменной *summa* на *summa*+*digit*, то есть выполнить присваивание

**summa += digit**

Для того чтобы затем отсечь последнюю цифру числа *N*, разде-лим *N* на 10 (основание системы счисления):

**N = N // 10**

Эти три операции – выделение последней цифры числа, увеличение суммы и отсечение последней цифры – нужно вы-полнять несколько раз, пока все цифры не будут обработаны (и отсечены!) и в переменной *N* не останется ноль:

**N = int( input("Введите число: ") )**

**summa = 0**

**while N != 0:**

**digit = N % 10**

**summa += digit**

**N = N // 10**

**print( "Сумма цифр", summa )**

*Выполните ручную прокрутку программы при N* = *123. Определите итоговое значение переменной summa.*

Для введённого числа 123 программа должна выдать ответ 6 (последнее значение переменной *sum*). Это правильный ответ.

В отличие от предыдущего примера, здесь количество ша-гов цикла заранее неизвестно, оно определяется количеством цифр введённого числа.

*Сколько раз выполнится цикл, если ввести однозначное число? двузначное? K-значное? число* 0*?*

*Какова может быть сумма цифр двузначного числа? трёхзначного? K-значного?*

Докажем, что эта программа не зациклится, то есть не бу-дет работать бесконечно. Цикл завершается, когда переменная *N* становится равна нулю, поэтому нужно доказать, что это обя-зательно случится. По условию заданное число – натуральное, на каждом шаге цикла оно делится на 10 (остаток отбрасывает-ся), поэтому значение переменной *N* каждый раз уменьшается. В результате после очередного уменьшения оно обязательно станет равно нулю.

**Задачи**

1. Значения переменных *a* и *b* равны *a* = 4 и *b* = 6. Определите, сколько раз выполнится цикл, и чему будут равны значения этих переменных после его завершения:

|  |  |
| --- | --- |
| а) | **while a < b:**  **a += 1** |
| б) | **while a < b:**  **a += b** |
| в) | **while a > b:**  a = a + 1; |
| г) | **while a < b:**  b **= a - b;** |
| д) | **while a < b:**  **a = a - 1;** |

2. Найдите ошибку в программе:

**k = 0**

**while k < 10:**

**print( "привет" )**

Как её можно исправить?

3. Что будет выведено на экран в результате работы следующе-го цикла?

|  |  |  |  |
| --- | --- | --- | --- |
| а) | | **k = 1**  **while k < 5:**  print( k, end=" " )  k += 1 | |
| б) | | **k = 4**  **while k < 10:**  **print( k\*k, end=" " )**  k += 1 | |
| в) | | **k = 12**  **while k > 3:**  print( 2\*k-1, end=" " )  **k -= 1** | |
| г) | | **k = 5**  **while k < 10:**  **print( k\*k, end=" " )**  **k += 1** | |
|  | | |
| д) | **k = 15**  **while k > 6:**  print( k-1, end=" " )  **k -= 1** | |

Домашнее задание:

1. Напишите программу, которая получает с клавиатуры коли-чество повторений и выводит столько же раз какое-нибудь сообщение.

2. Напишите программу, которая получает с клавиатуры нату-ральное число и определяет, сколько раз в его десятичной записи встречается цифра 1.

3. Напишите программу, которая получает с клавиатуры нату-ральное число и находит наибольшую цифру в его десятич-ной записи.