# Лабораторная работа № 3. Алгоритм циклической структуры

В предыдущих работах мы рассматривали задачи, использующие алгоритмы линейной структуры. В такой структуре действия производятся последовательно, друг за другом. А что, если нам необходимо запрограммировать повторяющуюся последовательность действий? Более того, если нам нужно повторять данные действия в зависимости от каких-либо условий? Тут нам на помощь приходят циклические структуры.

В алгоритме циклической структуры (*цикле*) серия команд (*тело цикла*) повторяется многократно. При этом нужно указать, либо сколько раз исполнитель должен выполнить тело цикла, либо при каком *условии* исполнитель будет повторять тело цикла еще раз.

|  |  |
| --- | --- |
| Вербальный алгоритм | Блок-схема |
| Пока условие цикла соответствует истине, выполняй тело цикла. Иначе, выполняй оператор после цикла. | Нет  Да  Тело цикла  Условие |

В качестве примера рассмотрим следующую задачу.

***Задача***. *Подсчитать факториал указанного числа.*

**Математическая постановка задачи.** Введем математические обозначения величин: n – вводимое число, чей факториал необходимо вычислить; f - факториал.

Для решения задачи необходимо вспомнить математическое определение факториала:

n! = 1 \* 2 \* 3 \* … \* n = f

Проще говоря, факториал – произведение всех чисел от 1 до указанного. Для решения задачи необходимо использовать цикл с условием, так как заранее неизвестно вводимое число, следовательно, и количество повторений цикла.

***Решение:***

Начало

i = 1

f = 1

Ввод n

Нет

i <= n

Да

f = f \* i

i += 1

Вывод f

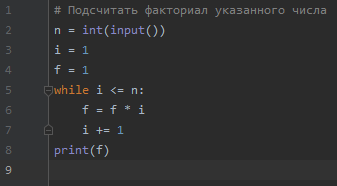
Конец

Заметим, что для выполнения нам понадобилась специальная переменная *i*, которая является счетчиком, содержащим текущее число ряда. Если текущее число ряда стало больше, чем число, факториал которого необходимо найти, прерываем цикл и завершаем выполнение программы.

Для отладки алгоритма достаточно сравнить результаты его исполнения с результатами ручного решения задачи.

### *Программная реализация алгоритма*

1. ***Запрос данных пользователя***Перед началом реализации главной части программы, необходимо запросить у пользователя число, факториал которого необходимо вычислить. Вспоминаем, что для успешной работы с числом в дальнейшем, его необходимо перевести в тип данных “integer” с помощью функции *int()*
2. ***Инициализация переменных***Для работы программе понадобится две предварительно инициализированные переменные: *i* и *f*. Переменная *i* содержит текущее число ряда, тогда как переменная *f* накапливает результат последнего произведения чисел ряда. После завершения работы цикла данная переменная будет содержать результат вычисления факториала.
3. ***Объявление цикла while***Цикл *while* имеет синтаксис, схожий с другими управляющими конструкциями языка:   
   *while условие:**тело цикла*Соответственно, объявление цикла в нашем случае выглядит таким образом:  
   *while i <= n:*
4. ***Тело цикла***В теле цикла — умножение текущего числа из ряда на результат предыдущего произведения. Затем, увеличение содержимого переменной *i* на единицу с помощью оператора одновременного вычисления суммы и присвоения (*i += 1*), который аналогичен данной записи: *i = i + 1*.
5. ***Вывод результата***В конце программы выводим результат работы, содержащийся в переменной *f*.

Таким образом, финальный код программы имеет данный вид:

**ЗАДАЧИ НА САМОСТОЯТЕЛЬНУЮ РАБОТУ**

Решить линейным алгоритмом следующие задачи:

1. Считать сумму введенных с клавиатуры чисел до тех пор, пока не будет введен -1.
2. Напишите программу используя цикл с предусловием while для вывода каждого четного положительного числа от 1 до 30 в одну строку через пробел.
3. Выведите на экран, все нечетные числа, делящиеся на 5 нацело, лежащие в диапазоне от 10 до 115.
4. Пользователь вводит целые числа, в ответ выводите символ $. Если введенное пользователем число равно 16, то завершите цикл не выводя ничего в ответ.
5. С клавиатуры вводится натуральное число. Найти его наибольшую цифру. Например, введено число 49821. Наибольшая цифра в нем 9.
6. В программе генерируется случайное целое число от 1 до 10. Пользователь должен его отгадать не более чем за 3 попытки. После каждой неудачной попытки должно сообщаться больше или меньше введенное пользователем число, чем то, что загадано. Если за 3 попытки число не отгадано, то вывести загаданное число.
7. Найти сумму и произведение цифр, введенного натурального числа.
8. Сформировать из введенного числа обратное по порядку входящих в него цифр и вывести на экран.