УВАЖАЕМЫЕ СТУДЕНТЫ!

ВАМ НЕОБХОДИМО ВЫПОЛНИТЬ СЛЕДУЮЩЕЕ:

1. Ознакомиться с теорией и законспектировать не менее трех страниц.
2. Составить и ответить на вопросы, записать примеры.
3. Предоставит фото отчет в течении трех дней .

Отправить преподавателю на почту v.vika2014@mail.ru и указать свою Ф.И.О, группу, и название дисциплины тел 0721744922

Тема: Составление циклических алгоритмов. Алгоритмы циклической структуры.

Цель: *Приобретение навыков построения алгоритмов циклической структуры*.

Базовая структура цикл. Обеспечивает многократное выполнение некоторой совокупности действий, которая называется телом цикла. Используется при решении задач табулирования и построения графиков функций, вычисления сумм и произведений (конечных и бесконечных), решения уравнений и т.п. Алгоритмы, реализующие такие вычисления, называют циклическими, а повторяющиеся участки вычислений - циклами. Использование циклов позволяет выполнять большие объемы вычислений при помощи компактных программ. Структура цикл существует в трех вариантах: цикл с предусловием (цикл ПОКА), цикл с постусловием (цикл ДО) и цикл с параметром. На рисунке 10 изображен цикл с предусловием, а на рисунке 11 – цикл с постусловием, которые называют условными циклическими алгоритмами

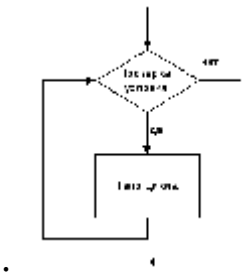


Рисунок 1-Цикл--ПОКА

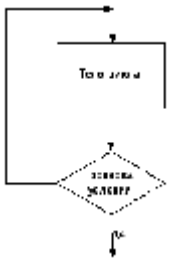


Рисунок 2- Цикл-ДО

Эти циклы взаимозаменяемы и обладают некоторыми отличиями.

- в цикле с предусловием условие проверяется до тела цикла, в цикле с постусловием – после тела цикла;
- в цикле с постусловием тело цикла выполняется хотя бы один раз, в цикле с предусловием тело цикла может не выполниться ни разу;
- в цикле с предусловием проверяется условие продолжения цикла, в цикле с постусловием – условие выхода из цикла.

При написании условных циклических алгоритмов следует помнить следующее. Во-первых, чтобы цикл имел шанс когда-нибудь закончиться, содержимое его тела должно обязательно влиять на условие цикла. Во-вторых, условие должно состоять из корректных выражений и значений, определенных еще до первого выполнения тела цикла.

Кроме того, существует безусловный циклический алгоритм (рисунок 3), который удобно использовать, если известно, сколько раз необходимо выполнить тело цикла.

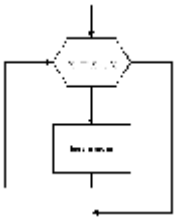
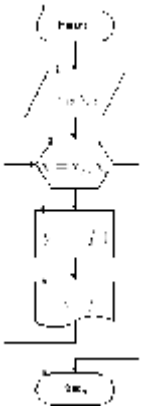


Рисунок 3- Цикл с параметром

Типичным примером такого алгоритма является алгоритм решения задачи табулирования функции одной переменной.

Пример. Вычислить значения функции $y = f(x)$ некоторой переменной x, изменяющейся от начального значения x_o до конечного x_k с постоянным шагом h. Эта задача реализуется с помощью цикла с заданным количеством повторений, которое определяется как $n = mod((x_k - x_o)/h) + 1$. Процедура вычислений сводится к следующему: вычисляется значение y для начального значения x, равного x_0. Затем x увеличивается на значение шага h, и для него вычисляется значение y. Последний этап повторяется до тех пор, пока x не превысит конечного значения x_k. (Схема алгоритма решения задачи приведена на рисунке 4). Переменная x называется параметром цикла. Назначение блоков 1, 2 и 8 ясно из схемы алгоритма. В блоке 3 осуществляется подготовка цикла -присвоение начального значения x_o параметру цикла x. Блоки 4 и 5 представляют собой тело цикла, т. е. участок вычислений, повторяющихся при различных значениях параметра цикла. В блоке 6 текущее значение параметра цикла x увеличивается на значение шага h. Результат операции $x+h$ присваивается переменной x, предыдущее значение x при этом стирается. В блоке 7 проверяется условие продолжения цикла. При выполнении условия $x<=x_k$ цикл повторяется, при невыполнении — повторение цикла прекращается, т. е. осуществляется выход из цикла.

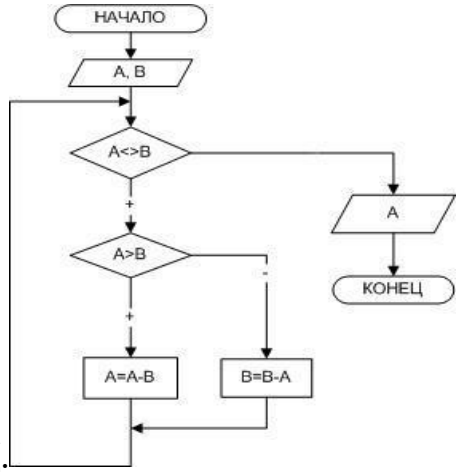
Рисунок 4- Цикл с заданным числом повторений с использованием блока условия	 Рисунок 5- Цикл с заданным числом повторений с использованием блока модификации

С использованием блока модификации схема алгоритма имеет вид, приведенный на рисунке 5. Блок модификации 3 помещается в начале цикла и выполняет те же функции, что и блоки 3, 6, 7 в схеме на рисунке 4. Запись $x = x_o, x_k, h,$ помещенная в блоке модификации, представляет собой заголовок цикла и предписывает следующие действия: «Для x, изменяющегося от x_0 до x_k с шагом h, выполнить операции, указанные в блоках, следующих за блоком модификации, и ограниченные линией, входящей слева в блок модификации». Выход из цикла обозначается линией связи, выходящей справа из блока модификации. Если величины x_0, x_k, h представлены в блоке модификации константами, например $x = 0,50,1,$ то при значении шага, равном единице, его запись можно опустить: $x=0, 50$

Оформление схем циклических алгоритмов с использованием блока модификации является предпочтительным.

Рассмотрим использование алгоритмов циклической структуры на конкретных примерах.

Пример 1. Найти наибольший общий делитель (НОД) двух натуральных чисел А и В. Входные данные: А и В. Выходные данные: А – НОД.



Для решения поставленной задачи воспользуемся алгоритмом Евклида: будем уменьшать каждый раз большее из чисел на величину меньшего до тех пор, пока оба значения не станут равными.

В блок–схеме решения задачи, представленной на рис. 6, для решения поставленной задачи используется цикл с предусловием, то есть тело цикла повторяется до тех пор, пока А не равно В.

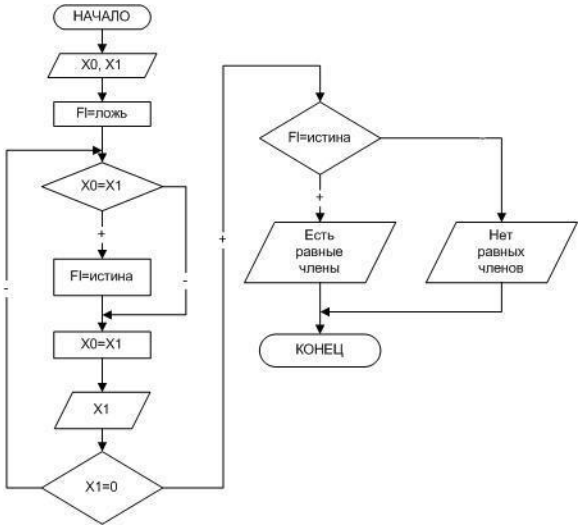
Рисунок 6-Поиск наибольшего общего делителя двух чисел

Рисунок 6 - Поиск наибольшего общего делителя двух чисел

Пример Вводится последовательность чисел, 0 – конец последовательности. Определить, содержит ли последовательность хотя бы два равных соседних числа.

Рисунок 7- Поиск равных соседних элементов последовательности

Входные данные: X_0 – текущий член последовательности, X_1 – следующий член последовательности.



Выходные данные: сообщение о наличии в последовательности двух равных соседних элементов.

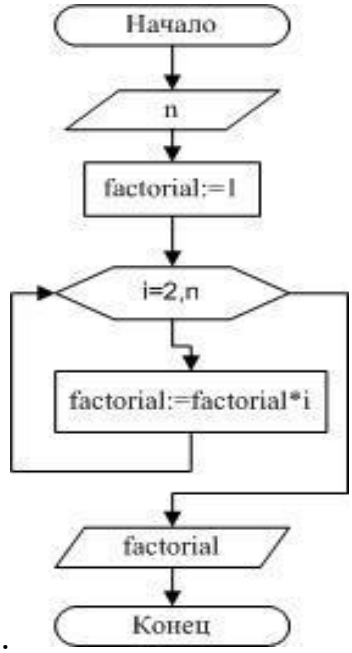
Вспомогательные переменные: F_l – логическая переменная, сохраняет значение «истина», если в последовательности есть равные рядом стоящие члены и «ложь» - иначе.

Блок–схема решения задачи приведена на рисунке. 7. Применение здесь цикла с постусловием обосновано тем, что необходимо вначале сравнить два элемента последовательности, а затем принять решение об окончании цикла.

*Пример 3.*Вычислить факториал числа N (N!=1*2*3 …*N).

Входные данные: N– целое число, факториал которого необходимо вычислить.

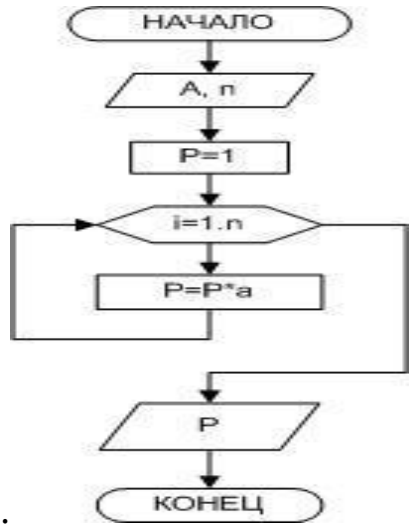
Выходные данные: factorial– значение факториала числа N, произведение чисел от 1 до N, целое число.



Промежуточные данные: i– целочисленная переменная, принимающая значения от 2 до N с шагом 1, параметр цикла. Блок-схема приведена на рисунке 8.

Рисунок 8 - Вычисление факториала

Итак, вводится число N. Переменной factorial, предназначенной для хранения значения произведения последовательности чисел, присваивается начальное значение, равное единице. Затем организуется цикл, параметром которого выступает переменная i. Если значение параметра цикла меньше или равно N, то выполняется оператор тела цикла, в котором из участка памяти с именем factorial считывается предыдущее значение произведения, умножается на текущее значение параметра цикла, а результат снова помещается участок памяти с именем factorial. Когда параметр i становится больше N, цикл заканчивается, и на печать выводится значение переменой factorial, которая была вычислена в теле цикла.



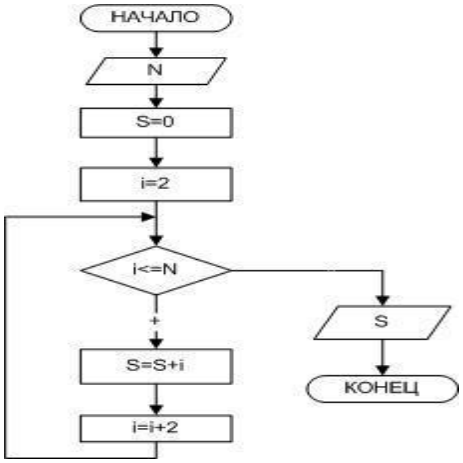
Пример 4. Вычислить $a^n(n>0)$.

Входные данные: *a* – вещественное число, которое необходимо возвести в целую положительную степень *n*.

Выходные данные*: p* (вещественное число) – результат возведения вещественного числа а в целую положительную степень *n*.

Рисунок 9 - Возведение вещественного числа в целую степень

Промежуточные данные: *i*– целочисленная переменная, принимающая значения от 1 до *n* с шагом 1, параметр цикла. Известно, что для того, чтобы получить целую степень *n* числа *a*, нужно умножить его само на себя *n* раз. Результат этого умножения будет храниться в участке памяти с именем *p*. При выполнении очередного цикла из этого участка предыдущее значение будет считываться, умножаться на основание степени *a* и снова записываться в участок памяти *p*. Цикл выполняется *n* раз.



Пример 5.Вычислить сумму натуральных четных чисел, не превышающих N.

Входные данные: N – целое число.

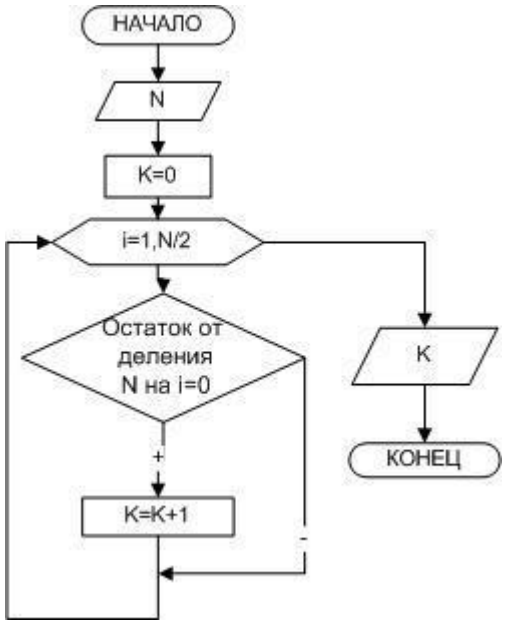
Выходные данные: S – сумма четных чисел.

Промежуточные данные: i – переменная, принимающая значения от 2 до N с шагом 2, следовательно, также имеет целочисленное значение.

Рисунок 10 - Вычисление суммы четных, натуральных чисел

При сложении нескольких чисел необходимо накапливать результат в определенном участке памяти, каждый раз считывая из этого участка предыдущее значение суммы и прибавляя к нему следующее слагаемое. Для выполнения первого оператора накапливания суммы из участка памяти необходимо взять такое число, которое не влияло бы на результат сложения. Т.е. перед началом цикла переменной, предназначенной для накапливания суммы, необходимо присвоить значение нуль.

Пример 6. Дано натуральное число N. Определить K – количество



делителей

этого числа, не превышающих его (N=12, его делители 1, 2, 3, 4, 6, K=5).

Входные данные: N – целое число.

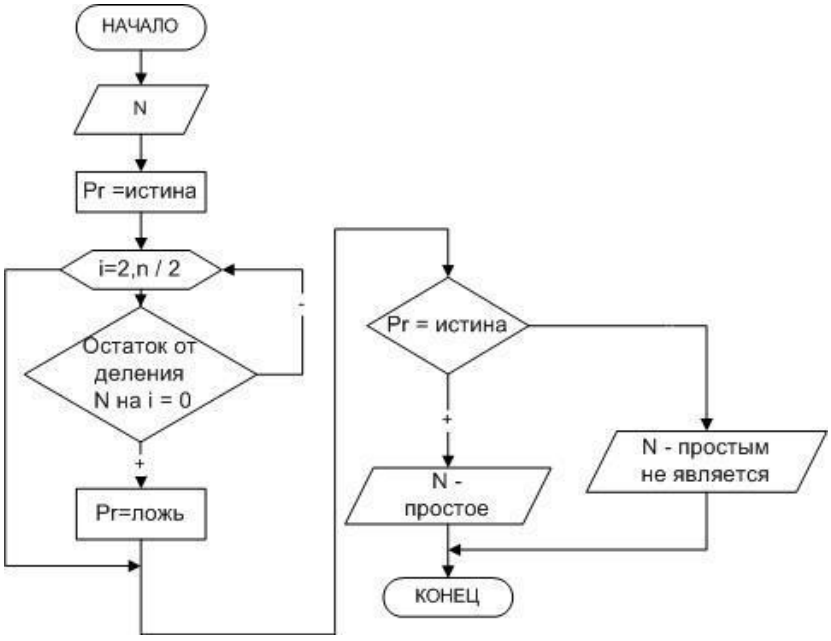
Выходные данные: целое число K – количество делителей N.

Промежуточные данные: i – параметр цикла, возможные делители числа N.

Рисунок 11 - Определение количества делителей натурального числа

В блок-схеме, изображенной на рисунке 11, реализован следующий алгоритм: в переменную K, предназначенную для подсчета количества делителей заданного числа, помещается значение, которое не влияло бы на результат, т.е. нуль. Далее организовывается цикл, в котором изменяющийся параметр i выполняет роль возможных делителей числа N. Если заданное число делится нацело на параметр цикла, это означает, что i является делителем N, и значение переменной K следует увеличить на единицу. Цикл необходимо повторить N/2 раз.

*Пример 7.*Дано натуральное число N. Определить, является ли оно простым. Натуральное число N называется простым, если оно делится нацело без остатка только на единицу и N. Число 13 – простое, так как делится только на 1 и 13, N=12 не является простым, так как делится на 1, 2, 3, 4, 6 и



1

Входные данные: N – целое число. Выходные данные: сообщение.

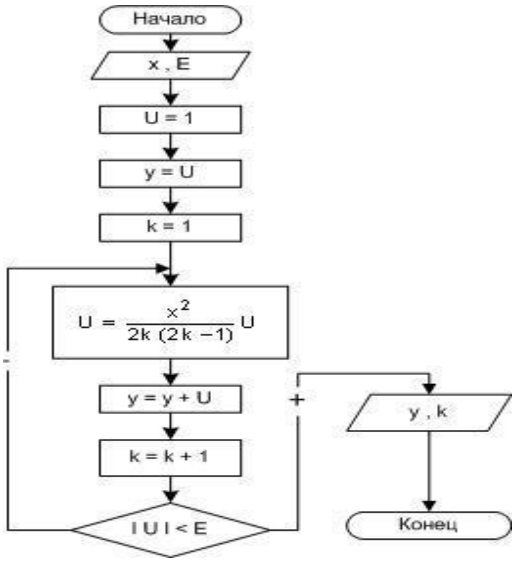
Промежуточные данные: i – параметр цикла, возможные делители числа N.

Рисунок 12 - Определение простого числа

Алгоритм решения этой задачи (рисунок 12) заключается в том, что число N делится на параметр цикла i, изменяющийся в диапазоне от 2 до N/ Если среди значений параметра не найдется ни одного числа, делящего заданное число нацело, то N – простое число, иначе оно таковым не является.

Обратите внимание на то, что в алгоритме предусмотрено два выхода из цикла. Первый естественный, при исчерпании всех значений параметра, а второй - досрочный. Нет смысла продолжать цикл, если будет найден хотя бы один делитель из указанной области изменения параметра.

Пример 8. Вычислить значения функции $y = chx = \sum_{k=1}^{\infty} U_k$ используя рекуррентную $U_{k+1} = \frac{x^2}{(2k-1)*2k} U_k$ (k=1,2,3,…) формулу. Вычисления прекратить при выполнении условия $|U_{k+1}| < \varepsilon$.



Определить количество итераций. Начальные значения параметров равны U_1=1; x=5.5; точность ε принять равной 10^{-4}.

Входные данные: U – начальное приближение, x, E – точность (ε).

Выходные данные: y — значение вычисленной функции, k — количество итераций.

Рисунок 13 - Вычисление суммы значений функции с заданной точностью

Пример 9. Дано натуральное число N. Определить самую большую цифру и ее позицию в числе (N=573863, наибольшей является цифра 8, ее позиция – четвертая слева).

Входные данные: N – целое число.

Выходные данные: max – значение наибольшей цифры в числе, pos – позиция этой цифры в числе.

Рисунок 14 - Определение количества цифр в числе

Промежуточные данные: i – параметр цикла, kol – количество цифр в числе, M – переменная для временного хранения значения N.

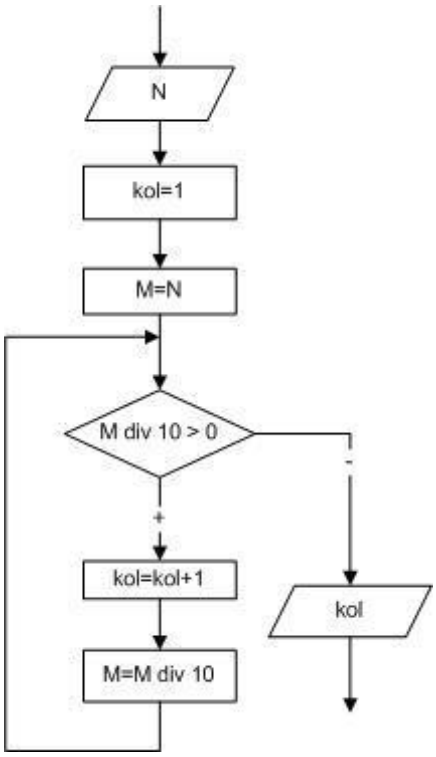
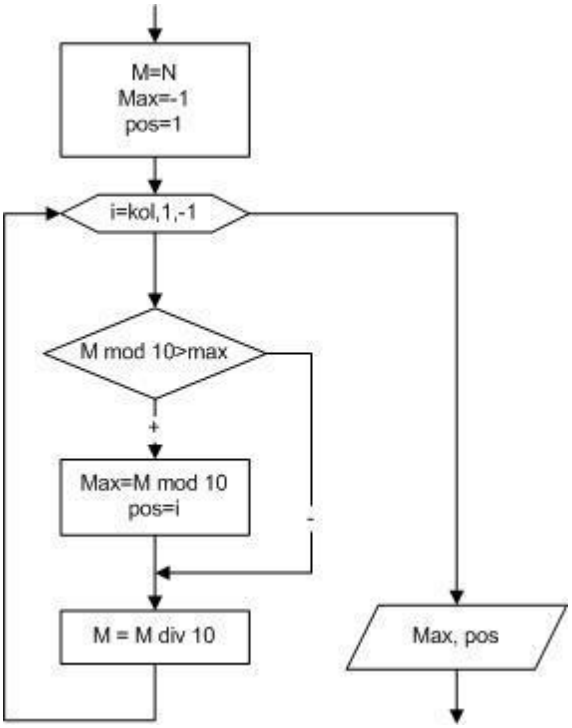


Рисунок 15 - Определение максимальной цифры в числе и ее позиции

Разобьем решение этой задачи на два этапа. Вначале найдем количество цифр в заданном числе (рисунок 14), а затем определим наибольшую цифру и ее позицию (рисунок 15). Для того, чтобы подсчитать количество цифр в числе, необходимо определить, сколько раз заданное число можно разделить на десять нацело. Если цифры числа известны, определить наибольшую из них не составит труда. Алгоритм поиска максимального значения в некоторой последовательности цифр заключается в следующем. В ячейку, в которой будет храниться максимальный элемент (max), записывают значение, меньшее любого из элементов последовательности (в нашем случае max=-1, так как цифры числа находятся в диапазоне от 0 до 9). Затем сравнивают элементы последовательности со значением ячейки max, если найдется элемент, превышающий значение предполагаемого максимума, то ячейке max необходимо присвоить значение этого элемента и, соответственно, запомнить его номер в последовательности

(в нашем случае переменной pos присваивается значение параметра цикла



i).

В алгоритме поиска минимума вначале в переменную min записываем значение, заранее большее любого элемента последовательности. Затем все элементы последовательности сравниваем с min, если встретится значение меньшее, чем min, переписываем его в переменную min