Тема урока: Программирование циклов с заданным условием продолжения/окончания работы.
Дата урока: 29.04.2020

Указания к работе
1. Изучите теоретический материал.
2. Письменно в тетради выполните самостоятельную работу (фотографии прислать 29.04.2020 до 18:00).
3. Выполните домашнее задание.

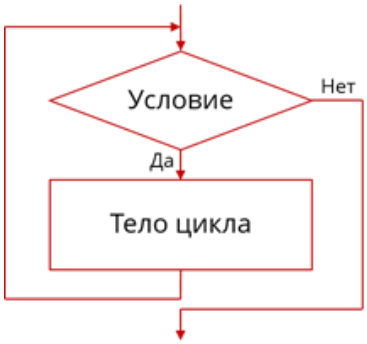
Теоретический материал

В разветвляющихся алгоритмах в зависимости от условия выполняется одна из двух последовательностей действий. Но часто возникает ситуация, когда от условия зависит количество исполнений некоторой последовательности действий. Для описания таких ситуаций используются циклические алгоритмы.

Циклическими называются алгоритмы, которые содержат помимо прочих конструкцию повторения.
Повторение (цикл) – это алгоритмическая конструкция, представляющая собой последовательность действий, которая выполняется многократно. Последовательность действий, исполняемых в цикле, называется **телом цикла**.

Циклы с заданным условием продолжения работы (с предусловием)

Рассмотрим, как записывается цикл с предусловием на языке Паскаль. Для этого используется оператор цикла, который записывается служебным словом **while**. После этого слова через пробел записывается условие цикла, а после условия тоже через пробел записывается служебное слово **do**. На русский язык эту конструкцию можно перевести «Пока <условие> делай». На следующей строке, на расстоянии одного пробела от оператора цикла, будет следовать тело цикла. Оно состоит из серии операторов, записанных в порядке своего исполнения. Как правило, они записываются в логических скобках, то есть между служебными словами **begin** и **end**. Если тело цикла состоит из одного оператора, то логические скобки записывать необязательно.




```
while <условие> do
begin
 <оператор 1>;
 <оператор 2>;
 …
end;
```

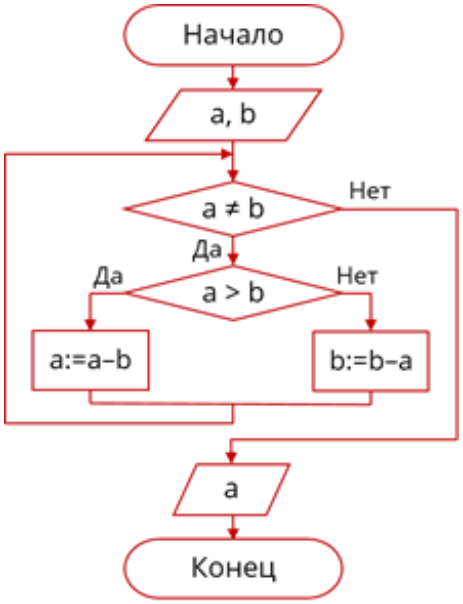
Задача 1. Написать программу поиска наибольшего общего делителя двух целых положительных чисел по алгоритму Эвклида (наибольшим общим делителем двух чисел называется набольшее число, на которое без остатка делятся оба числа).

Словесное описание алгоритма Эвклида

Пока числа не равны между собой – наибольшее из них заменяется разностью его самого и наименьшего числа. После чего выводится любое из них.

Сначала пользователь вводит два целых числа, назовём их **a** и **b**. После этого следует условный блок **a** ≠ **b**. Если это условие выполняется – будет следовать ещё один условный блок, который будет определять из двух не равных чисел наибольшее. Его условием будет **a** > **b**. Если это условие выполняется, то переменной **a** мы должны присвоить значение разности **a** и **b**. В противном случае **b** будет больше **a**, и переменной **b** мы присвоим значение **b - a**. После выполнения этого ветвления мы должны вернуться к условному блоку **a** ≠ **b**. Если условие этого блока не будет выполняться, нам достаточно вывести на экран значение любой из переменных, например **a**.

Напишем программу nod. Для решения задачи понадобятся две переменных, числа **a** и **b**. По условию задачи, это целые числа, поэтому переменные будут принадлежать к целочисленному типу **integer**. Запишем логические скобки. Сначала будет следовать оператор **writeln**, который будет выводить поясняющее сообщение, что это программа расчёта НОД двух чисел и запрос на ввод двух чисел. Теперь запишем оператор **readln (a, b)**. Дальше запишем оператор цикла с предусловием **while**. Его условие будет **a<>b**. Дальше будет следовать служебное слово **do**. Так как цикл будет содержать всего один условный оператор, логические скобки указывать не требуется. Запишем условный оператор **if**. Его условием будет **a > b**.



После служебного слова **then** будет следовать оператор присваивания **a:=a-b**. Далее будет следовать служебное слово **else**. После него будет следовать оператор присваивания **b:=b-a**. После цикла достаточно написать оператор вывода значения переменной **a**.

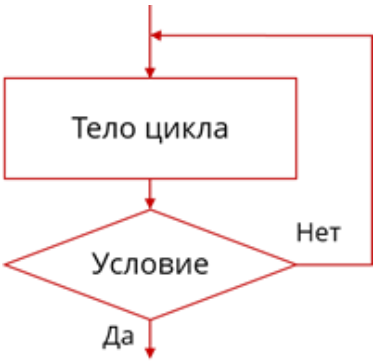
```
program nod;
var
 a, b: integer;
begin
 writeln ('Программа расчёта НОД двух чисел. Введите два числа.');
 readln (a, b);
 while a<>b do
  if a>b
  then a:=a-b
  else b:=b-a;
 write ('НОД равен ', a);
end.
```

Возможны случаи, когда алгоритм Эвклида будет работать медленно. Например, когда одно число во много раз больше другого. При числах 1 000 000 и 1, цикл в нашей программе повторится 999 999 раз. Поэтому появился усовершенствованный алгоритм Эвклида.

Словесное описание усовершенствованного алгоритма Эвклида: пока оба числа не равно нулю, наибольшее из двух чисел заменяется остатком от деления себя на наименьшее число. После выполнения этих действий ненулевое число и будет равно значению НОД.

Циклы с заданным условием окончания работы (с постусловием)

Рассмотрим, как записывается цикл с постусловием на языке Паскаль. Для цикла с постусловием есть оператор, обозначающийся служебным словом **repeat**. После него со следующей строки, на расстоянии одного пробела следует тело цикла, которое представляет собой последовательность операторов. После тела цикла на одном уровне со словом **repeat** следует служебное слово **until**. После него следует условие цикла, которое представляет собой логическое высказывание. После условия следует точка с запятой. На русский язык эту конструкцию можно перевести как «Выполнять тело цикла, пока не условие». Обратим внимание, что служебные слова **repeat** и **until** сами играют роль логических скобок, то есть выделять тело цикла служебными словами **begin** и **end** не требуется.



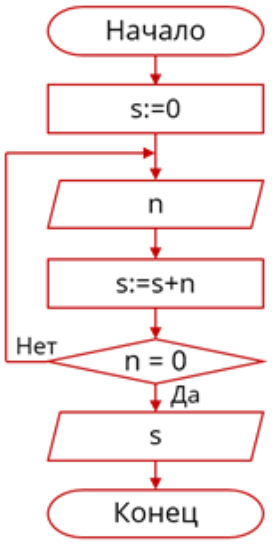

```
repeat
 <оператор 1>;
 <оператор 2>;
 …
until <условие>;
```

Задача 2. Написать программу для расчёта суммы чисел, введённых пользователем. При этом пользователь вводит произвольное количество чисел, а для выхода из программы вводит ноль.

Словесное описание алгоритма

Обозначим **s** – сумму чисел. Вначале она должна быть равна 0. Присвоим ей это значение. Далее пользователь введёт с клавиатуры значение слагаемого, обозначим его **n**. После этого программа увеличит значение суммы на текущее слагаемое. Далее с помощью условного блока проверим, равно ли введённое слагаемое нулю. Если условие не выполняется, значит снова вернёмся к считыванию

очередного слагаемого. Если же очередное слагаемое равно нулю, выведем на экран итоговое значение суммы. На этом программа завершит свою работу.



Напишем программу summa. Объявим две переменные: **s** и **n**. Так как в условии задачи не сказано, что пользователь вводит целые числа, переменные будут принадлежать к вещественному типу **real**. Запишем логические скобки. В начале программы будет следовать оператор **writeln**, который будет выводить на экран поясняющее сообщение о том, что это программа расчёта суммы слагаемых, вводимых пользователем. Дальше присвоим переменной **s** значение 0. После этого запишем оператор цикла с постусловием **repeat**. На следующей строке сразу запишем служебное слово **until** и условие цикла: **n=0**. То есть цикл будет повторяться до тех пор, пока очередное слагаемое не окажется равным 0. Теперь запишем тело цикла. В начале цикла будет следовать оператор **write**, который будет выводить запрос на ввод очередного слагаемого или 0 для выхода. Далее будет следовать оператор **readln (n)**. Теперь запишем оператор присваивания **s:=s+n**. На этом написание цикла будет завершено. После цикла будет следовать оператор **write**, который будет выводить на экран сообщение о том, что работа программы завершена, и что итоговая сумма равна **s**.

```
program summa;
var
 s, n: real;
begin
 writeln ('Программа расчёта суммы чисел, введённых пользователем.');
 s:=0;
 repeat
  write ('Текущая сумма: ', s, '. Введите очередное слагаемое или 0 для
выхода>>>');
  readln (n);
  s:=s+n;
 until n = 0;
 write ('Итоговая сумма: ', s);
end.
```

Самостоятельная работа: решить задачи
Учебник, задание 2, 4, 6 на стр. 141-142

Дополнительное задание: выполнить задание на сайте Российская Электронная Школа (скриншот с результатом прислать 29.04.2020 до 18:00): https://resh.edu.ru/subject/lesson/3062/train/#188433

Домашнее задание: выучить §3.5.1-3.5.2