Тема. Применение рекурсивных функций.

Задание А. В задании дается 2 задачи и одновременно поясняется действие рекурсии. Для сравнения приводятся программы, выполняющие действия с помощью рекурсии и итерационным способом. Рекомендуется сравнить время выполнения и того и другого способа. Примеры учебные, поясняющие оформления рекурсивной подпрограммы.

Теоретические сведения

Рекурсия – это определение объекта через самого себя.

С помощью рекурсии в математике определяются многие бесконечные множества, например множество натуральных чисел.

Процедуры и функции в современных языках программирования так же могут вызывать сами себя.

Рекурсивными называются процедуры и функции, которые вызывают сами себя.

При использовании рекурсивных подпрограмм велика опасность, что вложенные вызовы будут продолжаться бесконечно (это похоже на зацикливание цикла). Поэтому в таких подпрограммах необходимо предусмотреть условие, которое проверяется на каждом шаге и заканчивает вложенные вложенные вызовы, когда перестает выполняться.

При новом рекурсивном вызове компьютер делает так:

1. Запоминается состояние вычислений на данном этапе.
2. В стеке (особой области памяти) создается новый набор локальных переменных (чтобы не испортить переменные текущего вызова).

Поскольку при каждом вызове затрачивается новая память и расходуется время на вызов подпрограммы и возврат из нее, при использовании рекурсии необходимо помнить, что т глубина рекурсии должна быть достаточно мала.

Глубина рекурсии – количество вложенных вызовов.

Программа, использующая рекурсию с большой глубиной, будет выполняться долго и может вызвать переполнение стека (нехватку стековой памяти).

Поэтому, если задача может быть легко решена без использования рекурсии, рекурсию использовать не желательно. Доказано, что любая рекурсивная программа может быть написана без использовании рекурсии, хотя такая реализация может оказаться очень сложной

Практическая часть

Пример 1. Вычислить факториал натурального числа **n**!

Решение. Типичной функцией, на которой объясняют рекурсию, является **факториал** **n**!=1*2*3*…*n. Он может быть описан следующим образом

$$n!=\begin{cases}1, & n=1(n=0)\\ (n-1)!*n, & n>1\end{cases}$$

То есть для вычисления факториала, если n>1, нужно сначала вычислить (n-1)!. А для вычисления последнего надо сначала вычислить (n-2)!. Соответственно, для вычисления (n-2)! надо знать (n-3)! и т.д. Значение 1! (0!) задается явно. В программе реализованы два метода вычисления факториала: с помощью рекурсии {**} и прямой {*} – вычислением по формуле с использованием цикла.

```
var
   s:longint;
   n: word;
{*подпрограмма без использованием рекурсии}
function f_iter(n:word): longint;
var     i:word;
   fi :longint;
begin
 fi:=1;
 for i:=1 to n do   fi:=fi*i;
 f_iter:=fi;
end;
```

```
{**подпрограмма с использованием рекурсии}
function f(n:word): longint;
 begin
      if (n=1) or (n=0) then f:=1  else f:=n*f(n-1);
end;
begin  {основная программа }
  write('Введите  n=');   read(n);
  s:=f(n);
  writeln('Рекурсия: n!=',s);
  writeln('Цикл: n!=',f_iter(n));
  write('click <enter>');   readln;
end.
```

Проверьте работу программы для n=2, 4, 9. Запишите ответы.

Пример 2. Вычислить n-ое число Фибоначчи.
Решение. Для закрепления понятия рекурсии запрограммируем вычисление **n-го числа Фибоначчи** также двумя методами: с использованием рекурсии и итерационным.

```
var    s:longint;
       n: word;
{*подпрограмма без использованием рекурсии}
function fib_iter(n:word): longint;
  var  i:word;
       a,b,c :longint;
 begin
   a:=1;      b:=1;
    if (n=1) or (n=0) then fib_iter:=1 else
       begin
        for i:=2 to n do begin
                         c:=a+b;
                         a:=b;
                         b:=c;
                         end;
        fib_iter:=c;
       end;end;
```

```
 {*подпрограмма с использованием рекурсии}
function fib(n:word): longint;
 var f:longint;
 begin
    if (n=1) or (n=0) then fib:=1
          else fib:=fib(n-1) + fib(n-2);
 end;
begin    { основная программа}
  write('Введите  n=');
  read(n);
  writeln('Рекурсия: ',fib(n));
  writeln('Цикл: ',fib_iter(n));
  write('click <enter>');
   readln;
end.
```

Проверьте работу программы для n=2, 6, 10. Запишите ответы.
Сравните время выполнения и того и другого способа, указав n достаточно большим.

Пример 3. Вычислить наибольший общий делитель двух натуральных чисел.
Решение. Рекурсивный **алгоритм Евклида вычисления НОД**:

$$НОД(a,b)=\begin{cases} НОД(a-b,b), & если \quad a>b \\ a \quad , & если \quad a=b \\ НОД(a,b-a), & если \quad b>a \end{cases}$$

Словами этот рекурсивный алгоритм можно выразить следующим образом. Для того, чтобы найти НОД двух чисел a и b, необходимо их сравнить. Если a=b, то НОД равен a. Если же одно из чисел больше другого, то нужно из большего вычесть меньшее, и повторно применить вышеизложенный алгоритм к паре чисел: разности большего и меньшего и меньшему из чисел. Процесс завершится, когда оба числа становятся равными.

```
Var     a, b : longint;
   function NOD (a,b:longint):longint;
      begin
         if a>b      then NOD := NOD(a-b,b)
           else  if a<b        then NOD := NOD (a,b-a)        else NOD := a;
      end;
begin      {основная программа}
   readln (a,b);
   writeln (NOD (a,b));
end.
```

Замечания:
а) Просмотреть действие рекурсивного алгоритма можно с помощью режима Вид /Окно отладки/ и выбрать в качестве просматриваемых переменных переменную ‘n’ (можно косвенно увидеть прямой и обратный ход рекурсии) при исполнении программы пошагово (нажимая F7). Например, для n=4 n будет последовательно меняться n=4,3,2,1,1,2,3,4
б) можно в рекурсивной подпрограмме ввести дополнительные переменные, присвоить им значения этих подпрограмм и просматривать уже значения дополнительных переменных.

Задание Б. Написать рекурсивную программу вычисления целой неотрицательной степени числа A.
По определению $A^n=A*A*A*…..*A=A^{n-1}*A$ (n>0); $A^0=1$
Из определения следует, что для того, чтобы вычислить n-ю степень числа, необходимо знать значение (n-1)-й степени этого числа. Если же n=0, то значение степени равно 1.

Контрольные вопросы

1. Структура подпрограмм функция и процедура. Вызов подпрограмм.
2. Локальные и глобальные переменные
3. Параметры подпрограмм
4. Рекурсивные подпрограммы