Итерация и рекурсия

Итерация – от человека,
Рекурсия – от Бога.

-

Л.Питер Дойч

Определение процедуры:

Процедура <имя>(<список входных параметров; список выходных параметров>)

<действие 1>

<действие 2>

……

[<имя>:=<выражение>]

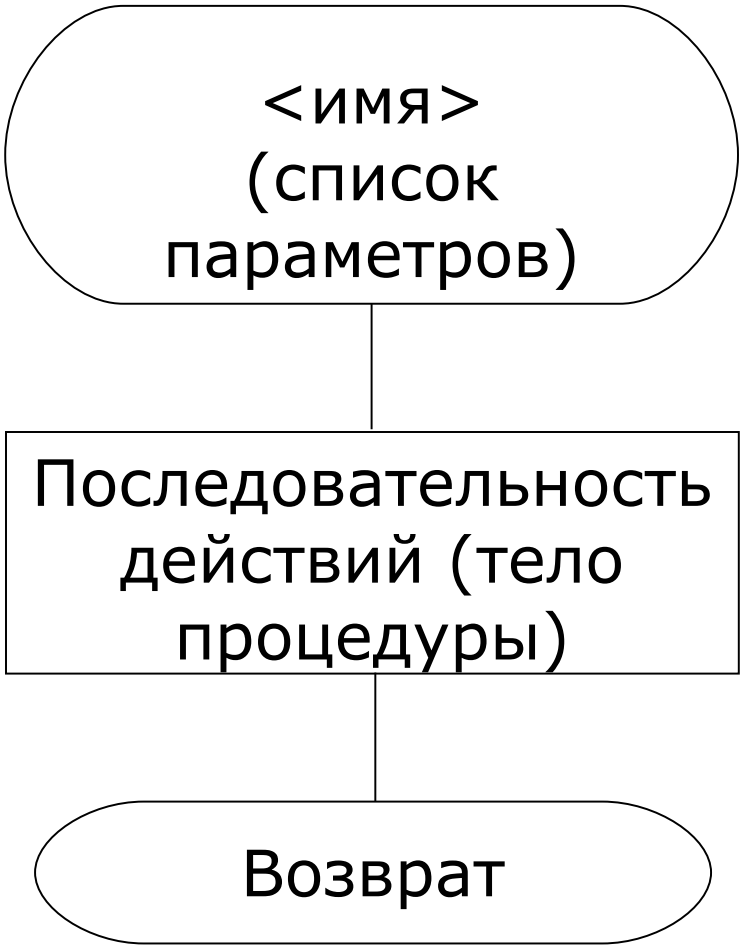
Возврат

Конец

Вызов процедуры:

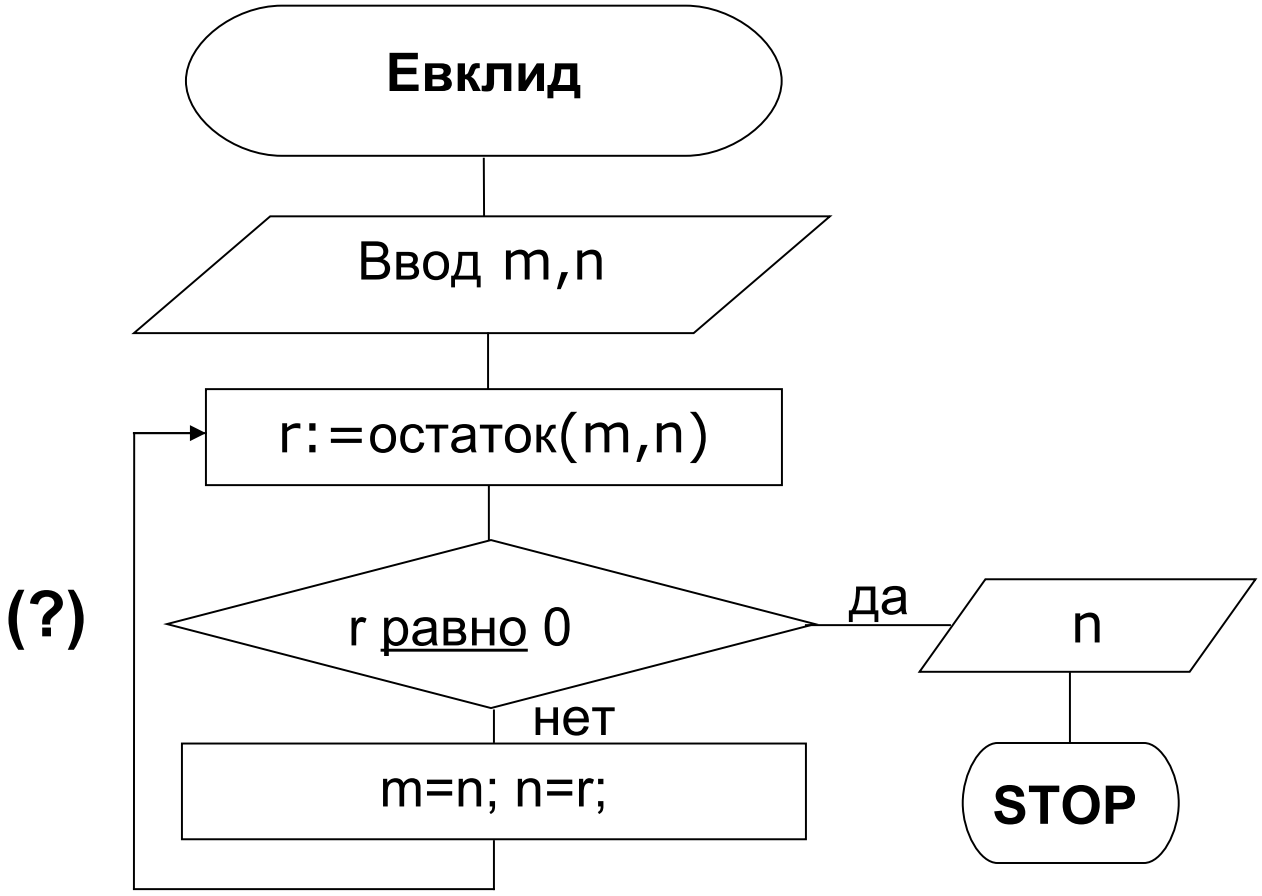
<имя>(<список значений входных параметров>;
<список выходных параметров>)

Определение процедуры:

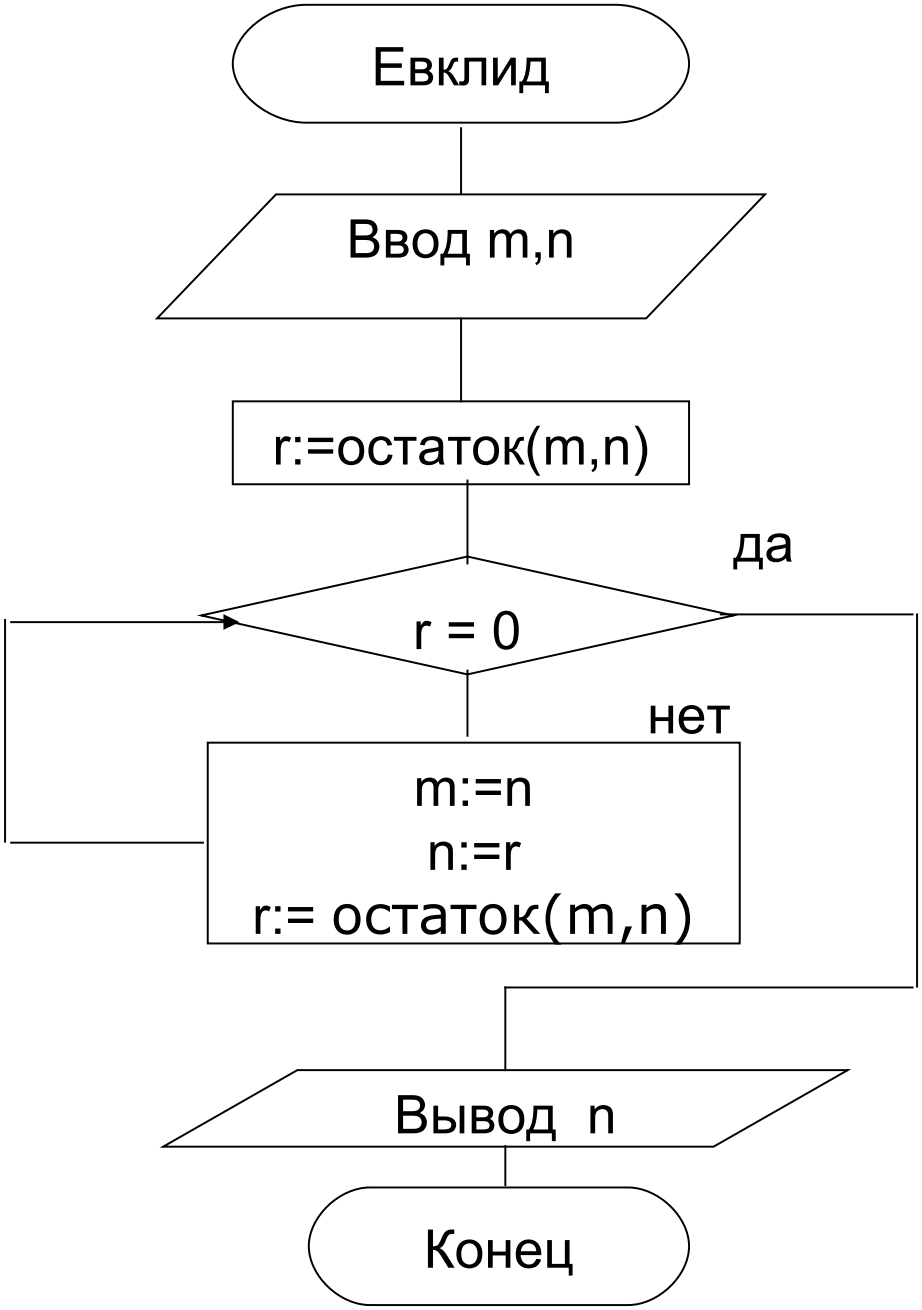


Вызов процедуры:

Введение процедуры позволяет создавать собственные стандартные блоки, которые, наряду с основными алгоритмическими конструкциями можно использовать при разработке алгоритмов.


Евклид
Ввод m,n
r:=остаток(m,n)
(?)
r равно 0
да
n
нет
m=n; n=r;
STOP

```
Процедура  остаток(m, n)
      r:=m
      Цикл-пока  r >= n
          r:=r-n
      Конец цикла
      остаток:=r
      Возврат
Конец
```


Евклид
Ввод m,n
r:=остаток(m,n)
да
r = 0
нет
m:=n
n:=r
r:= остаток(m,n)
Вывод n
Конец

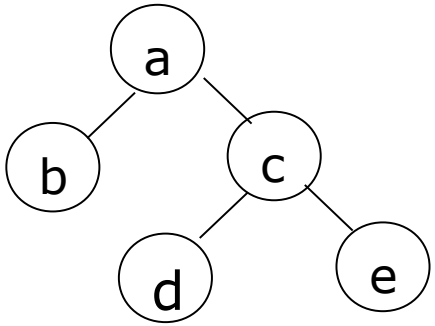
Введение понятия процедуры необходимо и достаточно для разработки ***рекурсивных алгоритмов****.*

Примеры рекурсивного определения:

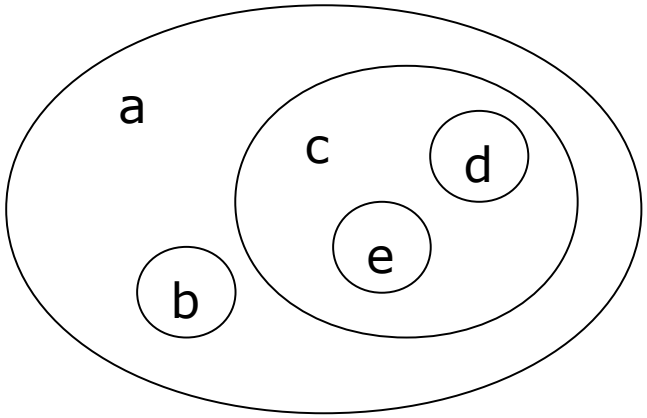
1. Матрёшка – это кукла, внутри которой находится матрёшка.
2. Если внутри куклы может находится больше одной матрешки, то такой объект структурно эквивалентен ***дереву***.

Древовидные структуры могут быть представлены различными способами –

- в виде диаграмм



или



- в виде вложенных скобок ***a(b,c(d,e))***
- в виде отступов

```
a
  b
  c
    d
    e
```

и т.д.

Пример рекурсивного определения функции (факториала):

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

Алгоритм называется *рекурсивным*, если в его определении содержится прямое или косвенное использование этого же алгоритма.

Такие алгоритмы содержат процедуры с вызовом самой себя:

Прямая рекурсия:

Процедура
рекурсия
……..
рекурсия
……..
Возврат
Конец

Косвенная рекурсия:

Процедура
рек1
……
рек2
……
Возврат
Конец

Процедура
рек2
……
рек1
……
Возврат
Конец

Если алгоритм содержит повторяющиеся действия, то их можно представить в *итеративной форме*, используя циклы, или в *рекурсивной форме*, используя рекурсивный вызов процедуры.

Рекурсивные процедуры, как и циклы, могут приводить к бесконечным вычислениям. Поэтому обращение к таким процедурам должно управляться некоторым условием, которое когда-то становится ложным.

```
Процедура  fact(n)
  Если n=0
  то
     fact:=1
  иначе
     fact:=n*fact(n-1)
  Конец-если
  Возврат
Конец
```

```
Процедура  fact(n)
  fact:=1
  Если n=0
  то  Возврат
  Конец-если
  Для    i:=1, n
     fact:=i*fact;
  Конец-цикл
  Возврат
Конец
```

Пусть имеется некий **Список** элементов. Надо построить алгоритм поиска хотя бы одного элемента с данным значением.

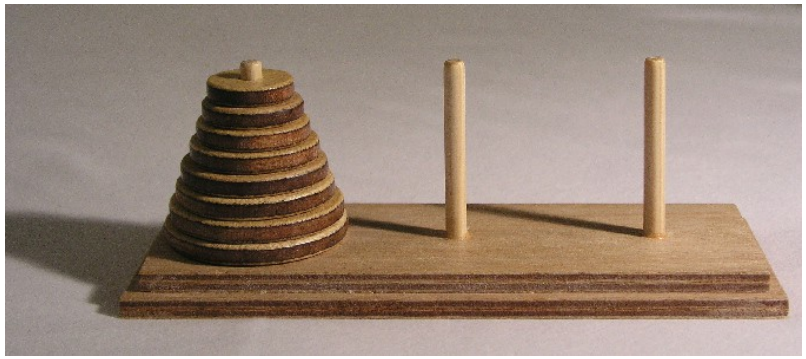
Последовательный алгоритм поиска:

```
Процедура Поиск(список, искомое)
 Если список=пустой
 то    Вывод «Поиск неудачен»
 иначе
  проверяемое:= Первый_элемент_списка(список)
  Цикл-пока проверяемое не равно искомое и список не равно пустой
  проверяемое:=Следующее_значение_в_списке(список, проверяемое)
  Конец-цикл
  Если  проверяемое=искомое
  то   Вывод «Элемент имеется в списке»
  иначе  Вывод «Элемент отсутствует в списке»
  Конец-если
 Конец-если
 Возврат
Конец
```

Если **Список** упорядоченный по какому-то правилу, то для поиска элемента с данным значением можно построить более эффективный алгоритм.

Алгоритм двоичного поиска:

```
Процедура Поиск(список, искомое)
  Если список=пустой
  то    Вывод «Поиск неудачен»
  иначе
   проверяемое:= Средний_элемент_списка(список)
   Если  проверяемое= искомое
   то Вывод ««Элемент имеется в списке»
   иначе
       Если   проверяемое > искомое
       то
         Поиск(верхняя_половина_списка, искомое)
       иначе
         Если проверяемое < искомое
            Поиск(нижняя_половина_списка, искомое)
          Конец-если
        Конец-если
      Конец-если
  Конец-если
  Возврат
Конец
```

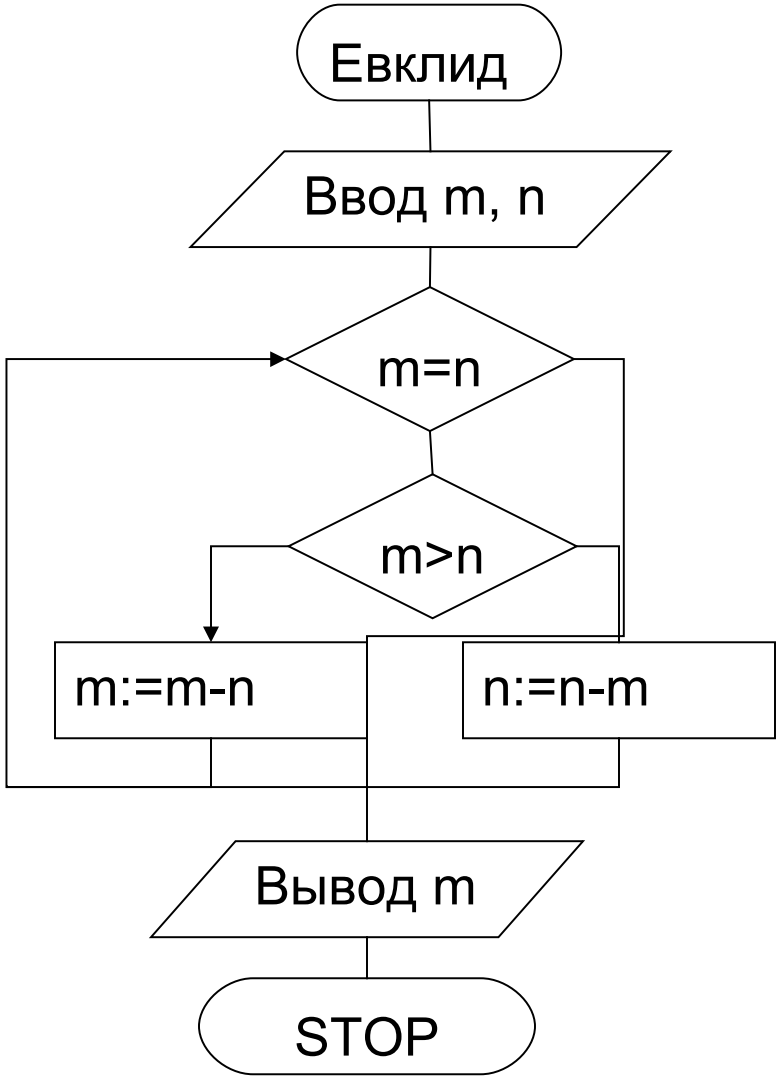


Задача о ханойских башнях

```
Процедура Ханой(n, a,b,c)
Если n=1
то
  a переместить на b
иначе
  Ханой(n-1, a,c,b)
  a переместить на b
  Ханой(n-1, c,b,a)
Возврат
Конец
```

```
Процедура Ханой(n, a,b,c;list)
 Если n=1
 то
   Добавить( (a,b), list)
 иначе
   Ханой(n-1, a,c,b; list1)
   Добавить(list1, list)
   Добавить( (a,b), list)
   Ханой(n-1, c,b,a; list2)
   Добавить(list2, list)
 Возврат
Конец
```

//list – список пар, задающих перемещения



Упражнение: замените итеративный алгоритм нахождения НОД рекурсивным.

Литература

1. Соболь Б.В. и др. Информатика. Ростов-на-Дону: Феникс, 2007, 447с.
2. Брукшир Дж. Информатика и вычислительная техника. М: ПИТЕР, 2004, 619с.
3. Вирт Н. Алгоритмы и структуры данных. М: Мир, 1989, 360с.
4. Кнут Д. Искусство программирования. Т1. Основные алгоритмы. М: Вильямс, 712с.