УВАЖАЕМЫЕ СТУДЕНТЫ!

ВАМ НЕОБХОДИМО ВЫПОЛНИТЬ СЛЕДУЮЩЕЕ:

1. Ознакомиться с теорией и законспектировать в конспект.
2. Выполните задания и ответить навопросы.
3. Предоставит отчет в файле .doc, и базу данных 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: Вспомогательные алгоритмы (процедуры)

Вспомогательные алгоритмы и процедуры

В теории алгоритмов известно понятие вспомогательного алгоритма. Вспомогательным называется алгоритм решения некоторой подзадачи из основной решаемой задачи. В таком случае алгоритм решения исходной задачи называется основным алгоритмом.

В качестве примера рассмотрим следующую задачу: требуется составить алгоритм вычисления степенной функции с целым показателем y = xk, где k — целое число, x ≠ 0. В алгебре такая функция определена следующим образом:

$$x^n = \begin{cases} 1 & n = 0, \\ \dfrac{1}{x^{-n}} & n < 0, \\ x^n & n > 0. \end{cases}$$

Для данной задачи в качестве подзадачи можно рассматривать возведение числа в целую положительную степень.

Учитывая, что 1/x-n = (1/x) -n, запишем основной алгоритм решения этой задачи.

```
алг Степенная функция
цел n; вещ x,y
нач ввод x,n
  если n=0
  то  y:=1
  иначе  если   n>0
         то  СТЕПЕНЬ(x,n,y)
         иначе  СТЕПЕНЬ(1/x,-n,y)
         кв
кв
вывод y
кон
```

Здесь дважды присутствует команда обращения к вспомогательному алгоритму с именем СТЕПЕНЬ. Это алгоритм возведения вещественного основания в целую положительную степень путем его многократного перемножения. Величины, стоящие в скобках в команде обращения к вспомогательному алгоритму, называются фактическими параметрами.

В учебном алгоритмическом языке вспомогательные алгоритмы оформляются в виде процедур. Запишем на алгоритмическом языке процедуру СТЕПЕНЬ.

```
процедура СТЕПЕНЬ(вещ a,цел k,вещ z)
цел i
нач z:=1; i:=1
    пока i≤k, повторять
    нц z:=z*a
       i:=i+1
    кц
кон
```

Заголовок вспомогательного алгоритма начинается со слова «процедура», после которого следует имя процедуры и в скобках — список формальных параметров. В этом списке перечисляются переменные-аргументы и переменные-результаты с указанием их типов. Здесь a и k — формальные параметры-аргументы, z — параметр-результат. Следовательно, процедура степень производит вычисления по формуле z = ak. В основном алгоритме «Степенная функция» обращение к процедуре производится путем указания ее имени с последующим в скобках списком фактических параметров. Между формальными и фактическими параметрами процедуры должны выполняться следующие правила соответствия:

- по количеству (сколько формальных, столько и фактических параметров);

- по последовательности (первому формальному соответствует первый фактический параметр, второму — второй и т.д.);
- по типам (типы соответствующих формальных и фактических параметров должны совпадать).

Фактические параметры-аргументы могут быть выражениями соответствующего типа.

Обращение к процедуре инициирует следующие действия:

1. Значения параметров-аргументов присваиваются соответствующим формальным параметрам.
2. Выполняется тело процедуры (команды внутри процедуры).
3. Значение результата передается соответствующему фактическому параметру, и происходит переход к выполнению следующей команды основного алгоритма.

В процедуре степень нет команд ввода исходных данных и вывода результатов. Здесь присваивание начальных значений аргументам (a, n) производится через передачу параметров-аргументов. А присваивание результата переменной (y) происходит через передачу параметра-результата (z). Таким образом, передача значений параметров процедур — это третий способ присваивания (наряду с командой присваивания и командой ввода).

Использование процедур позволяет строить сложные алгоритмы методом последовательной детализации.

Процедура — подпрограмма, имеющая произвольное количество входных и выходных данных.

Описание процедуры имеет вид:

```
procedure <имя_процедуры> (<описание параметров-значений>;
          var: <описание параметров-переменных>);

 begin
   <операторы>
 end;
```

В заголовке процедуры после её имени приводится перечень формальных параметров и их типов. Входные параметры, значения которых не изменяются в программе, должны быть параметрами-значениями.

Выходные (результирующие) параметры должны быть параметрами-переменными.

Для вызова процедуры достаточно указать её имя со списком фактических параметров. В качестве параметров-значений можно указывать имена переменных, константы и выражения.

Например, заголовок процедуры вычисления наибольшего общего делителя может быть описан так:

procedure nod (a, b: integer; **var** c: integer);

Возможны следующие варианты вызова этой процедуры:

nod (36, 15, z) — в качестве параметров-значений использованы константы;

nod (x, y, z) — в качестве параметров-значений использованы имена переменных;

nod (x+y, 15, z) — в качестве параметров-значений использованы выражение и константа.

В любом случае между фактическими и формальными параметрами должно быть полное соответствие по количеству, порядку следования и типу.

Пример 1. Напишем процедуру для нахождения наибольшего общего делителя двух чисел с помощью алгоритма Евклида. Используем её для нахождения наибольшего общего делителя следующих шести чисел: 16, 32, 40, 64, 80 и 128.

program n_6;	Заголовок главной программы
const m: **array** [1..6] **of** integer = (16, 32, 40, 64, 80, 128);	Раздел описания констант
var i, x, y, z: integer;	Раздел описания переменных

procedure nod (a, b: integer; **var** c: integer);
begin
while a<>b **do**
if a>b **then** a:=a-b **else** b:=b-a;
c:=a
end;
	Раздел описания подпрограмм

begin
x:=m[1];
for i:=2 **to** 6 **do**
begin
y:=m[i];
nod (x, y, z);
x:=z
end;
writeln ('НОД=', x)
end.
	Раздел операторов главной программы

Измените программу так, чтобы с её помощью можно было найти: а) наибольший общий делитель следующих пяти чисел: 12, 24, 30, 48 и 51; б) наибольший общий делитель произвольных десяти целых двузначных чисел.

Функции

Описание функции имеет вид:

```
function <имя_функции> (<описание входных данных>):
         <тип_функции>;
begin
  <операторы>;
  <имя_функции>:=<результат>
end;
```

В заголовке функции после её имени приводится описание входных данных — указывается перечень формальных параметров и их типов. Там же указывается тип самой функции, т. е. тип результата.

Функция — подпрограмма, имеющая единственный результат, записываемый в ячейку памяти, имя которой совпадает с именем функции. Поэтому в блоке функции обязательно должен присутствовать оператор <имя_функции>: =<результат>.

Для вызова функции достаточно указать её имя со списком фактических параметров в любом выражении, в условиях (после слов **if, while, until**) или в операторе **write** главной программы.

Пример 2. Напишем программу нахождения максимального из четырёх целых чисел, использующую функцию поиска максимального из двух чисел:

`program n_7;`	Заголовок главной программы
`var a, b, c, d, f: integer;`	Раздел описания переменных

`function max (x, y: integer): integer;`
`begin`
`  if x>y then max:=x else max:=y;`
`end;`
	Раздел описания подпрограмм

`begin`
`  readln (a, b, c, d);`
`  f:=max(max(a, b), max(c, d));`
`  writeln ('f=', f);`
`end.`
	Раздел операторов главной программы

Измените программу так, чтобы с её помощью можно было найти:

а) максимальное из чисел a, b, c;

б) максимальное из чисел b, c, d;

в) минимальное из четырёх чисел;

г) разность максимального и минимального из четырёх чисел.

Пример 3. В январе Саше подарили пару новорождённых кроликов. Через два месяца они дали первый приплод — новую пару кроликов, а затем давали приплод по паре кроликов каждый месяц. Каждая новая пара также даёт первый приплод (пару кроликов) через два месяца, а затем — по паре кроликов каждый месяц. Сколько пар кроликов будет у Саши в декабре?

Составим математическую модель этой задачи. Обозначим через f(n) количество пар кроликов в месяце с номером n. По условию задачи, f(1) = 1, f(2) = 1, f(3) = 2. Из двух пар, имеющихся в марте, дать приплод в апреле сможет только одна: f(4) = 3. Из пар, имеющихся в апреле, дать приплод в мае смогут только пары, родившиеся в марте и ранее: f(5) = f(4) + f(3) = 3 + 2 = 5. В общем случае: $f(n) = f(n - 1) + f(n - 2)$, $n \geq 3$.

Числа 1, 1, 2, 3, 5, 8,... образуют так называемую **последовательность Фибоначчи**, названную в честь итальянского математика, впервые решившего соответствующую задачу ещё в начале XIII века.

Оформим в виде функции вычисление члена последовательности Фибоначчи.

```
function f (n: integer): integer;
begin
  if (n=1) or (n=2) then f:=1
  else f:=f(n-1)+f(n-2)
end;
```

Полученная функция — **рекурсивная**; в ней реализован способ вычисления очередного значения функции через вычисление её предшествующих значений.

Вопросы и задания

1. Ознакомьтесь с материалами презентации к параграфу, содержащейся в электронном приложении к учебнику. Дополняет ли презентация информацию, содержащуюся в тексте параграфа?

2. Для чего используются подпрограммы?

3. В чём основное различие процедур и функций?

4. Напишите программу вычисления наименьшего общего кратного следующих четырёх чисел: 36, 54, 18 и 15. Используйте процедуру вычисления наибольшего общего делителя двух чисел.

5. Напишите программу перестановки значений переменных a, b, c в порядке возрастания, т. е. так, чтобы a < b < c. Используйте процедуру **swap**.

```
procedure swap (var x, y: integer);
  var m: integer;
begin
  m:=x;
  x:=y;
  y:=m
end;
```

Исходные данные вводятся с клавиатуры.

Пример входных данных	Пример выходных данных
1 2 3	1 2 3
2 1 3	1 2 3
3 1 2	1 2 3
2 3 1	1 2 3

6. Видоизмените программу сортировки массива выбором так, чтобы в ней использовалась процедура выбора наибольшего элемента массива.

7. Напишите программу вычисления выражения: s = 1! + 2! + 3! +...+ n!

Здесь n! — факториал числа n. n! = 1*2*...*(n—1)*n. Используйте функцию вычисления факториала