Процедуры и функции

1. Понятие подпрограммы;
2. Стандартные функции и процедуры;
3. Пользовательские функции и процедуры;
4. Область видимости;
5. Параметр-значение, параметр-переменная;
6. Реентерабельность;
7. Рекурсия.

Процедуры и функции

Подпрограмма – это именованная часть программы, представляющая собой некоторое собрание операторов, структурированных аналогично основной программе.

Подпрограммы не необходимы, но их наличие заметно облегчает работу программиста и увеличивает «ценность» кода.

Описываются подпрограммы между разделом описания переменных и началом тела основной программы

Процедуры и функции

Так выглядит структура всей программы в Pascal:

Program *<имя программы>; { Заголовок программы }*
Uses *<имя 1, имя 2, … >; { Раздел описания модулей }*
Label *<метка 1, метка 2,…>; { Раздел описания меток }*
Const *<имя 1, имя 2,…>; { Раздел описания констант }*
Type *{ Раздел описания типов }*
<имя типа 1> = <определение типа 1>;
<имя типа 2> = <определение типа 2>;
…
Var *<имя 1, имя 2,…>: <тип>; { Раздел описания переменных }*
Procedure*; {Раздел описания процедур}*
Function*; {Раздел описания функций}*
Begin *{ Тело программы }*
<операторы>
End*.*

Процедуры и функции

1. Понятие подпрограммы;
2. **Стандартные функции и процедуры;**
3. Пользовательские функции и процедуры;
4. Область видимости;
5. Параметр-значение, параметр-переменная;
6. Реентерабельность.

Процедуры и функции

Стандартные функции и процедуры PASCAL:

- процедуры управления программой;
- функции преобразования;
- арифметические функции;
- порядковые процедуры и функции;
- строковые процедуры и функции;
- процедуры и функции динамического распределения памяти;
- прочие процедуры и функции.

Процедуры и функции

Процедуры управления программой:

Procedure Break;
Процедура осуществляет досрочный выход из циклов For, While или Repeat. Процедура должна находиться внутри этих циклов, иначе транслятор сообщит об ошибке.

Procedure Continue;
Процедура досрочно начинает следующую итерацию циклов For, While или Repeat.
Процедура должна находиться внутри этих циклов, иначе транслятор сообщит об ошибке.

Procedure Exit;
Осуществляет немедленный выход из текущей подпрограммы. Если текущей подпрограммой является главная программа, она завершает работу.

Procedure Halt[(Exitcode: Integer)];
Процедура выполняет аварийное завершение программы. Чтобы нормально завершить.

Процедуры и функции

Функции преобразования :

function Chr(a: byte): char;

Преобразует код в символ в кодировке Windows

function Ord(a: char): byte;

Преобразует символ в код в кодировке Windows

function Ord(a: integer): integer;

Возвращает порядковый номер значения a

function Trunc(x: real): integer;

Возвращает целую часть числа x

function Round(x: real): integer;

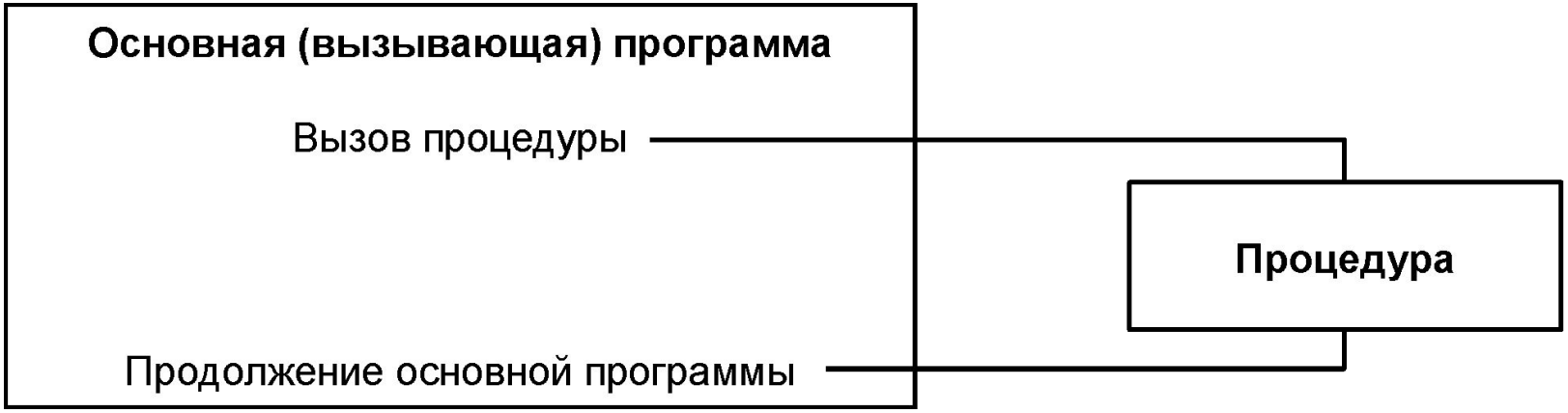
Возвращает x, округленное до ближайшего целого

Процедуры и функции

1. Понятие подпрограммы;
2. Стандартные функции и процедуры;
3. **Пользовательские функции и процедуры;**
4. Область видимости;
5. Параметр-значение, параметр-переменная;
6. Реентерабельность;
7. Рекурсия.

Процедуры и функции

Процедурой называется особым образом оформленный фрагмент программы, имеющий собственное имя. Упоминание этого имени в тексте программы приводит к активизации процедуры и называется её вызовом.



Процедуры и функции

Функция отличается от процедуры тем, что результат её работы возвращается в виде значения этой функции, поэтому вызов функции может использоваться наряду с другими операндами в выражениях.

Заголовок процедуры имеет вид:

PROCEDURE <имя> [(<сп. ф. п.>)];

заголовок функции:

FUNCTION <имя> [(<сп. ф. п.>)]: <тип>;

Здесь <имя> – имя подпрограммы;

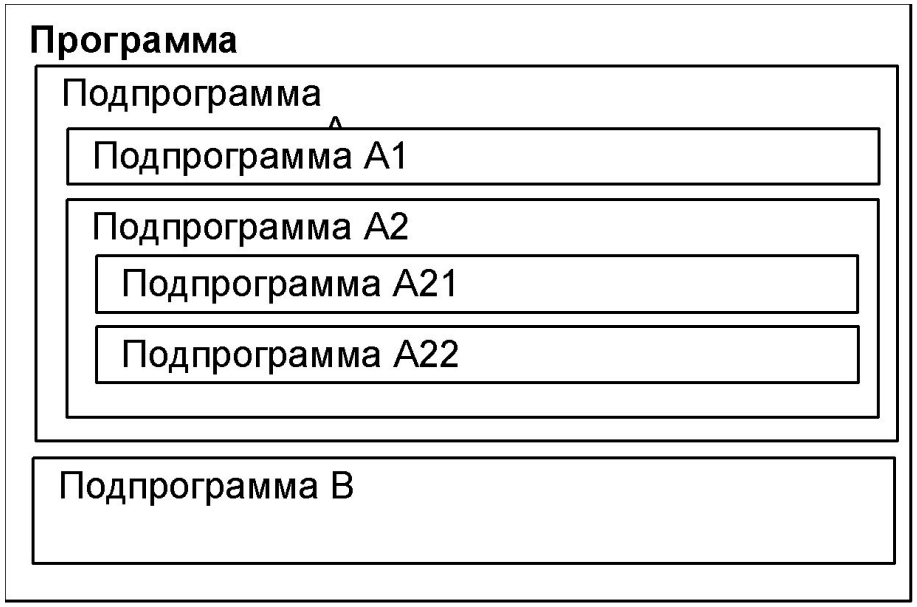
<сп. ф. п.> – список формальных параметров;

<тип> – тип возвращаемого функцией результата..

```
procedure SB (a: real; b: integer; c: char);
function F (a: real; b: real): real;
function F (a, b: real): real;
```

Процедуры и функции

При описании подпрограммы указывается её заголовок и тело. Тело подпрограммы***, подобно программе***, состоит из раздела описаний и раздела исполняемых операторов. В разделе описаний подпрограммы могут встретиться описания подпрограмм низшего уровня, т.е. вложенных подпрограмм.



Процедуры и функции

Пример. Разработать функцию возведение числа в степень. $a^x = e^{x \ln a}$

```
var
  x,y: real;
 function power( a, b: real): real;
  begin
        if a>0 then power := exp(b*ln(a))
         else power := 0;
  end;  {function power}

begin
repeat
  readln(x,y);
  writeln( power(x,y) );
until false;
end.
```

Процедуры и функции

Пример. Разработать процедуру, которая выводит на экран прямоугольник из заданных символов определенного размера.

```
procedure box (s: char; w,h: integer);
   var i,j:integer;
   begin
      for i := 1 to h do begin
         for j := 1 to w do
            write (s);
         writeln
      end;
      writeln
   end;
begin
   box ('+', 10, 5);
   box ('r', 20, 3);
   box ('|', 50, 10);
   box ('$', 12, 4);
end.
```

Процедуры и функции

1. Понятие подпрограммы;
2. Стандартные функции и процедуры;
3. Пользовательские функции и процедуры;
4. **Область видимости;**
5. Параметр-значение, параметр-переменная;
6. Реентерабельность;
7. Рекурсия.

Процедуры и функции

Подпрограмме доступны только те объекты верхнего уровня, которые описаны до описания данной подпрограммы. Эти объекты называются ***глобальными*** по отношению к подпрограмме.

Имена, ***локализованные*** в подпрограмме, могут совпадать с ранее объявленными глобальными именами. В этом случае считается, что локальное имя «закрывает» глобальное и делает его недоступным. Одноименные глобальные и локальные переменные – это разные переменные

Процедуры и функции

```
var k: integer;
procedure a;
   var x, z: real;
   begin
      { через x, z обозначены две величины –
      локальные переменные для a;
      k – глобальная переменная для a }
    end;
procedure b;
   var x, y: integer;
   begin
      { через x, y обозначены две другие величины –
      локальные переменные для b;
      k – глобальная переменная для b }
end;
begin
{ k – единственная переменная, которую
можно использовать в основной ветке программы }
……
end.
```

Процедуры и функции

```
var
   i: integer;
   a: real;

function p(d: real):real;
   var i: integer;
   begin
      i := 3;
      p := i + 10 * d;
   end;

begin
   a := 2.0;
   i := 15;
   p(a);
   writeln(' i = ', i, ' a = ', a);

   readln
end.
```

Две совершенно разные переменные:
первая i – глобальная для всей программы,
вторая i – локальная для функции p

Результат:
i = 15 a = 2

Процедуры и функции

1. Понятие подпрограммы;
2. Стандартные функции и процедуры;
3. Пользовательские функции и процедуры;
4. Область видимости;
5. **Параметр-значение, параметр-переменная;**
6. Реентерабельность;
7. Рекурсия.

Процедуры и функции

Любой из формальных параметров *подпрограммы* может быть либо **параметром-значением**, либо **параметром-переменной**. *В предыдущем примере параметры a и b определены как параметры-значения.* Чтобы определить параметры как параметры-переменные, перед ними необходимо поставить зарезервированное слово var, например

function power(var a: real, b: real): real;

При вызове подпрограммы, в качестве параметра-переменной должна указываться переменная соответствующего типа, в качестве параметра-значения может быть указано любое выражение соответствующего типа.

power(x, 3*sin(x)); **или** **power(x, 3.17);**

Процедуры и функции

```
var
  a, b: integer;

procedure inc2( var c: integer; b: integer );
  begin
     c := c + c;   b := b+b;
     writeln('удвоенные :', c:5, b:5);
  end;

begin
 a  := 5;   b := 7;
 writeln('исходные  :', a:5, b:5);
 inc2(a,b);
 writeln('результат : ', a:5, b:5);
end.
```

Результат выполнения программы:

исходные : 5 7

удвоенные : 10 14

результат : 10 7

Процедуры и функции

1. Понятие подпрограммы;
2. Стандартные функции и процедуры;
3. Пользовательские функции и процедуры;
4. Область видимости;
5. Параметр-значение, параметр-переменная;
6. **Реентерабельность;**
7. Рекурсия.

Процедуры и функции

Реентерабельная, или **повторно входимая функция** - это функция, которая может быть использована более чем одной задачей без риска потери данных.

Реентерабельная функция может быть в любое время прервана и продолжена позже без потерь данных. Реентерабельные функции либо используют локальные переменные, либо защищают свои данные, размещённые в глобальных переменных.

Функция fn не является реентерабельной:

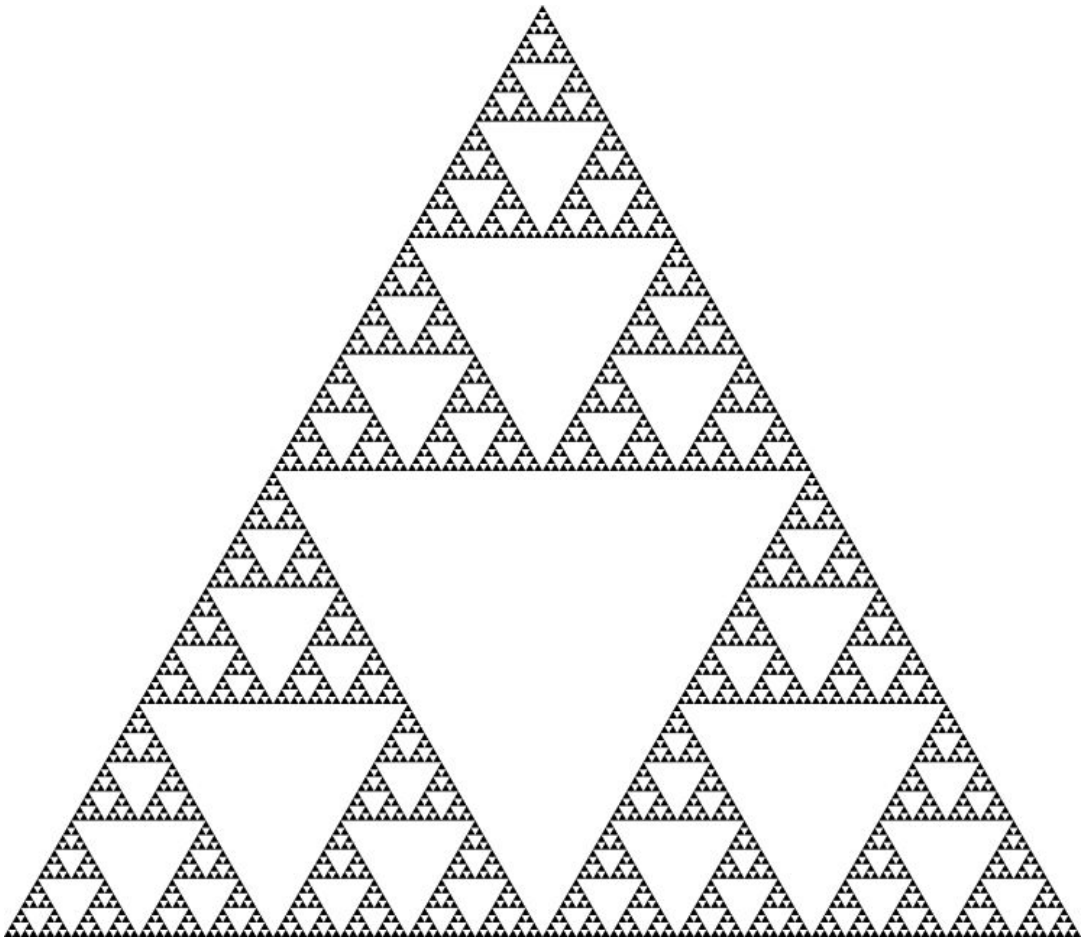
```
var
temp : integer;
function fn (x,y : integer):integer;
  begin
    temp  := x;
    x := y;
    y := temp;
    fn := x + y;
  end;
```

Процедуры и функции

1. Понятие подпрограммы;
2. Стандартные функции и процедуры;
3. Пользовательские функции и процедуры;
4. Область видимости;
5. Параметр-значение, параметр-переменная;
6. Реентерабельность;
7. **Рекурсия**.

Процедуры и функции

Реку́рсия — в определении, описании, изображении какого-либо объекта или процесса внутри самого этого объекта или процесса, то есть ситуация, когда объект является частью самого себя



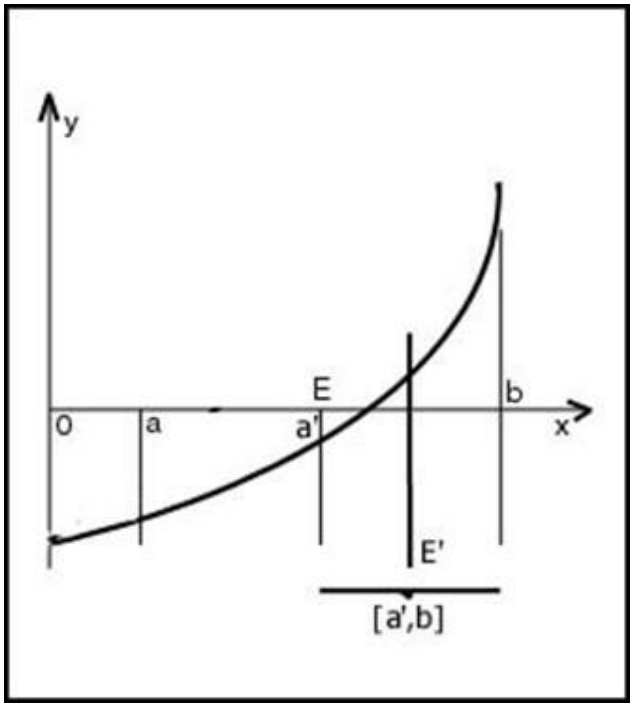
Процедуры и функции

Рекурсия – это такой способ организации вычислительного процесса, при котором подпрограмма в ходе выполнения составляющих её операторов обращается сама к себе

Пример. Рекурсия. Вычисление факториала

```
var n: integer;
 function fact(n: integer): integer;
  begin
  if n=0 then fact := 1
  else fact := n * fact(n-1);
  end;
 begin
   readln(n);
   writeln('n!=', fact(n));
end.
```

Пример: Реализовать метод половинного деления на основе рекурсии




```
const Eps = 0.01;            //точность вычисления корня
 var a,b: real;            // границы отрезка
 function y(x: real):real;  // задание функции, корни которой отделяются
  begin
     y := x + 5;
  end;
```

Процедуры и функции

```
//продолжение
function FindRoot(a,b: real): real;
var
  m: real;
  Result: real;
begin
m := (a+b) / 2;
if abs(m-a)>Eps then
    if y(a)*y(m)>0 then  Result:= FindRoot(m,b)
      else Result:= FindRoot(a,m)
else  Result := m;
FindRoot := Result;
end; {function FindRoot}

begin
  writeln('Введите левую и правую границы области поиска, a и b');
  readln(a,b);
  if y(a)*y(b)>0 then writeln('Нет корней или более одного корня')
    else writeln(Корень уравнения: ', FindRoot(a,b));
end.
```