Подпрограммы. Функции и процедуры.

Кулебякин В.В.

Создание пользовательских подпрограмм

Очень часто процесс решения какой – либо задачи может быть мысленно представлен как последовательность решения более простых подзадач.

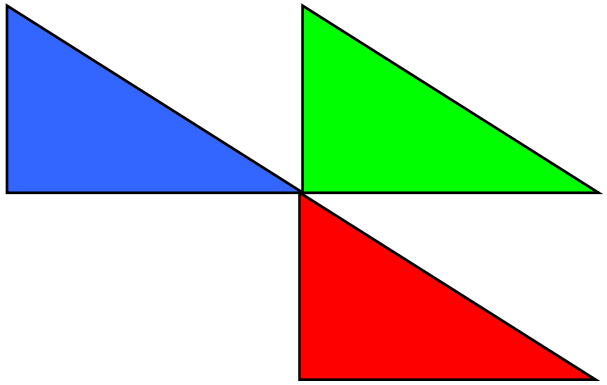
Процедуры и функции по своей структуре подобны обычным программам и имеют общее название – **подпрограммы**.

Применение подпрограмм дает возможность избежать повторения одной и той же последовательности операторов, а так же конструировать программу, разбивая ее на отдельные подзадачи – выполняя декомпозицию. Для сложных задач это существенно упрощает процесс программирования.

ЕСЛИ В ПРОГРАММЕ ЧТО-ТО ПОВТОРЯЕТСЯ – ОНА НАПИСАНА НЕПРАВИЛЬНО!

Процедуры

Задача: Построить фигуру:



Можно ли решить известными методами?

Особенность: Три похожие фигуры.

общее: размеры, угол поворота

отличия: координаты, цвет

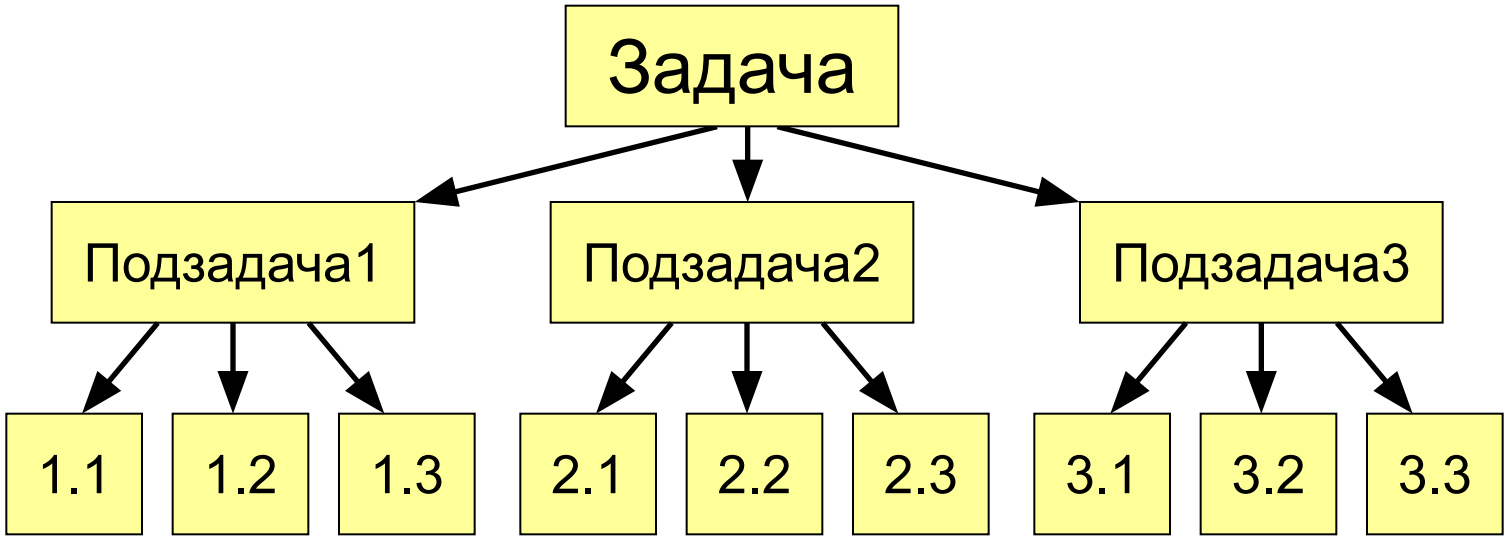
Сколько координат надо задать?

Процедуры

Процедура – это вспомогательный алгоритм, который предназначен для выполнения некоторых действий.

Применение:

- выполнение одинаковых действий в разных местах программы
- разбивка программы (или другой процедуры) на подзадачи для лучшего восприятия



Процедуры

- Процедура состоит из заголовка и тела. По структуре ее можно рассматривать как программу в миниатюре. Когда процедура описана, ее можно вызвать по имени из любой точки программы, где она видна (в том числе из нее самой!). Когда процедура выполнит свою задачу, программа продолжится с оператора, следующего непосредственно за оператором вызова процедуры. Использование имени процедуры в программе называется оператором вызова процедуры.
- Все процедуры и функции подразделяются на две группы: встроенные и созданные программистом.

Процедура=маленькая программа

- *procedure <имя процедуры> (<список формальных параметров>) ;*
- *const ...;*
- *type ...;*
- *var ...;*
- *begin*
- *<операторы>*
- *end;*

Все эти элементы ЛОКАЛЬНЫЕ и видны только внутри данной процедуры

Процедуры

Особенности:

- все процедуры расположены **выше** основной программы
- в заголовке процедуры перечисляются **формальные** параметры, они обозначаются именами, поскольку могут меняться

```
procedure Tr( x, y, r, g, b: integer);
```

- при вызове процедуры в скобках указывают **фактические** параметры (числа или арифметические выражения) **в том же порядке**

```
Tr (200, 100, 0, 255, 0);
```

Процедуры

Особенности:

- для каждого формального параметра после двоеточия указывают его тип

```
procedure A (x: real; y: integer; z: real);
```

- если однотипные параметры стоят рядом, их перечисляют через запятую

```
procedure A (x, z: real; y, k, l: integer);
```

- внутри процедуры параметры используются так же, как и переменные

Процедуры

Особенности:

- в процедуре можно объявлять дополнительные **локальные** переменные, остальные процедуры не имеют к ним доступа

```
program qq;
  procedure A(x, y:
integer);
    var a, b:
      real;
    begin
      a := (x + y)/6;
      ...
    end;
  end;
begin
  ...
end.
```

локальные переменные

Параметры-переменные

Задача: составить процедуру, которая меняет местами значения двух переменных.

Особенности:

надо, чтобы изменения, сделанные в процедуре, стали известны вызывающей программе

```
program qq;
var x, y: integer;
  procedure Exchange ( a, b: integer );
  var c: integer;
  begin
    c := a; a := b; b := c;
  end;
begin
  x := 1; y := 2;
  Exchange ( x, y );
  writeln ( 'x = ', x, ' y = ', y );
end.
```

эта процедура работает с **копиями** параметров

```
x = 1 y = 2
```

Вызов процедуры

- Вызов процедуры для выполнения осуществляется по ее имени, за которым в круглых скобках следует список фактических параметров, т.е. передаваемых в процедуру данных:
- *<имя процедуры> (<список фактических параметров>);*
- Если у процедуры нет параметров, то их список (в том числе круглые скобки) не указывается.

```
Power(a,b);
Main;
```

Параметры-переменные

параметры могут изменяться

```
procedure Exchange ( var   a, b: integer );
var c: integer;
begin
  c := a; a := b; b := c;
end;
```

Применение:

таким образом процедура (и функция) может возвращать несколько значений,

Запрещенные варианты вызова

```
Exchange ( 2, 3 );        { числа }
Exchange ( x+z, y+2 );    { выражения }
```

Вложенные процедуры и функции

Как и любые другие элементы программы, процедуры и функции могут быть локальными, если они описаны внутри другой процедуры или функции.

```
Procedure Main;
Procedure Main1(c:Real);
 Begin
…
End;
Begin …
 Main1
End;
```

Процедура Main1 является встроенной по отношению к процедуре Main и видна только внутри Main