Подпрограммы

В практике программирования часто складываются ситуации, когда одну и ту же группу операторов, реализующих определенную цель, требуется повторить без изменений в нескольких местах программы. Для избавления от столь нерациональной траты времени

была предложена концепция подпрограммы.

Подпрограмма – именованная, логически законченная группа операторов языка, которую можно вызвать для выполнения любое количество раз из различных мест программы. В языке Free Pascal существуют два вида подпрограмм: процедуры и функции. Главное отличие процедуры от функции заключается в том, что результатом исполнения операторов, составляющих тело функции, всегда является некоторое значение, поэтому функцию можно использовать непосредственно в выражениях, наряду с переменными и константами.

Общие сведения о подпрограммах. Локальные и глобальные переменные

Итак, подпрограмма – это поименованный набор описаний и операторов, выполняющих определенную задачу. Информация, передаваемая в подпрограмму для обработки, называется параметрами, а результат вычислений – значениями. Обращение к подпрограмме назы-

вают вызовом. Перед вызовом подпрограмма должны быть обязательно описана в разделе описаний. Описание подпрограммы состоит из заголовка и тела. В заголовке объявляется имя подпрограммы и в круглых скобках ее параметры, если они есть (для функции необходимо сообщить тип возвращаемого ею результата). Тело подпрограммы следует за заголовком и состоит из описаний и исполняемых операторов.

Любая подпрограмма может содержать описание других подпрограмм. Константы, переменные, типы данных могут быть объявлены как в основной программе, так и в подпрограммах различной степени вложенности. Переменные, константы и типы, объявленные в основной программе до определения подпрограмм, называются глобальными, они доступны всем функциям и процедурам. Переменные, константы и типы, описанные в какой-либо подпрограмме, доступны только в ней и называются локальными.

Для правильного определения области действия идентификаторов (переменных) необходимо придерживаться следующих правил:

•каждая переменная, константа или тип должны быть описаны перед использованием;

•областью действия переменной, константы или типа является та подпрограмма, в которой они описаны;

•все имена в пределах подпрограммы, в которой они объявлены, должны быть уникальными и не должны совпадать с именем самой подпрограммы;

•одноименные локальные и глобальные переменные – это разные переменные, обращение к таким переменным в подпрограмме трактуется как обращение к локальным переменным (глобальные переменные недоступны);

•при обращении к подпрограмме доступны объекты, которые объявлены в ней и до ее описания.

Формальные и фактические параметры. Передача параметров в подпрограмму

Обмен информацией между вызываемой и вызывающей функциями осуществляется с помощью механизма передачи параметров. Переменные, указанные в заголовке подпрограммы называются формальными параметрами или просто параметрами подпрограммы.

Эти переменные могут использоваться внутри подпрограммы. Список переменных в операторе вызова подпрограммы – это фактические параметры, или аргументы.

Механизм передачи параметров обеспечивает обмен данных между формальными и фактическими параметрами, что позволяет выполнять подпрограмму с различными данными. Между фактическими параметрами в операторе вызова и формальными параметрами в заголовке подпрограммы устанавливается взаимно однозначное соответствие. Количество, типы и порядок следования формальных и фактических параметров должны совпадать.

Передача параметров выполняется следующим образом. Вычисляются выражения, стоящие на месте фактических параметров. В памяти выделяется место под формальные параметры в соответствии с их типами. Выполняется проверка типов и при их несоответствии выдается диагностическое сообщение. Если количество и типы формальных и фактических параметров совпадают, то начинает работать механизм передачи данных между фактическими и формальными параметрами.

Формальные параметры процедуры можно разделить на два класса: параметры-значения и параметры-переменные.

При передаче данных через параметры-значения в подпрограмму передаются значения фактических параметров, и доступа к самим фактическим параметрам из подпрограммы нет.

При передаче данных параметры-переменные заменяют формальные параметры, и, следовательно, в подпрограмме есть доступ к значениям фактических параметров. Любое изменение параметров-переменных в подпрограмме приводит к изменению соответствующих им формальных параметров. Следовательно, входные данные следует передавать через параметры-значения, для передачи изменяемых в результате работы подпрограммы данных следует использовать параметры-переменные.

От общетеоретических положений перейдем к практическому использованию подпрограмм при решении задач. Изучение подпрограмм начнем с процедур.

Процедуры

Описание процедуры имеет вид:

procedure имя_процедуры(формальные_параметры);
label
    описание_меток;
const
    описание_констант;
type
    описание_типов;
var
    описание_переменных;
begin
    //Тело процедуры.
end;

Начинается описание с заголовка процедуры, где procedure – ключевое слово языка, имя_процедуры – любой допустимый в языке Free Pasacal идентификатор, формальные_параметры – имена формальных параметров и их типы, разделенные точкой с запятой.

Рассмотрим примеры заголовков процедур с параметрами-значениями:

procedure name_1(r:real; i:integer; c:char);

Однотипные параметры могут быть перечислены через запятую:

procedure name_2(a,b:real; i,j,k:integer);

Список формальных параметров не обязателен и может отсутствовать:

procedure name_3;

Если в заголовке процедуры будут применяться параметры-переменные, то перед

ними необходимо указывать служебное слово **var,** перед параметрами-значениями слово **var** отсутствует:

procedure name_4(x,y:real; var z:real);

//x, y – параметры-значения,

//z – параметр-переменная.

После заголовка идет тело процедуры, которое состоит из раздела описаний (константы, типы, переменные, процедуры и функции, используемые в процедуре) и операторов языка, реализующих алгоритм процедуры.

Для обращения к процедуре необходимо использовать оператор вызова:

имя_процедуры(список_фактических_параметров);

Фактические параметры в списке оператора вызова отделяются друг от друга запятой:

a:=5.3; k:=2;

s:='a';

name_1(a, k, s);

Если в описании процедуры формальные параметры отсутствовали, то и при вызове их быть не должно:

name_3;

ЗАДАЧА Найти действительные корни квадратного уравнения ax2+bx+c=0.

Решение

```
//Процедура для вычисления действительных корней квадратного уравнения.
procedure korni(a,b,c:real;var x1,x2:real; var pr:boolean);
//Входные параметры процедуры (параметры-значения):
//a,b,c – коэффициенты квадратного уравнения;
//Выходные параметры процедуры (параметры-переменные):
//x1,x2 – корни квадратного уравнения;
//pr – логическая переменная,
//принимает значение ложь, если в уравнении нет корней
//и значение истина в противном случае.
var
        d:real;
begin
        d:=b*b-4*a*c;
        if d<0    then pr:=false
                  else
                  begin
                          pr:=true;
                          x1:=(-b+sqrt(d))/2/a;
                          x2:=(-b-sqrt(d))/(2*a);
                  end
end;
 //Конец подпрограммы
//Основная программа
var
        a_,b_,c_,x1_,x2_,x_:real; pr_:boolean;
begin
        write('a_:='); readln(a_);
        write('b_:='); readln(b_);
        write('c_:='); readln(c_);
        if a_=0 then  //Если a_=0, то уравнение квадратным не является.
        Begin  //Решение  линейного уравнения bx+c=0.
                  if b_<>0 then begin        x_:=-c_/b_;  writeln('x=',x_); end
                           else       writeln('Нет корней');
        end
        else
                  //Решение уравнения ax^2+bx+c=0.
        Begin
                  korni(a_,b_,c_,x1_,x2_,pr_);  //Вызов процедуры.
                  if pr_=false then  writeln('Нет корней')
                  else  writeln('x1=',x1_,' x2=',x2_);
        end;
end.
```

Функции

Описание функции также состоит из заголовка и тела:

```
function имя_функции(формальные_параметры):тип;
label
        описание_меток;
const
        описание_констант;
type
        описание_типов;
var
        описание_переменных;
begin
        //Тело функции.
End;
```

Заголовок функции содержит: служебное слово function, любой допустимый в языке Free Pascal идентификатор — имя_функции; имена формальных параметров и их типы, разделенные точкой с запятой - формальные_параметры, тип возвращаемого функцией значения - тип (функции могут возвращать скалярные значения целого, вещественного, логического, символьного или ссылочного типа).

Примеры описания функций:

```
function fun_1 (x:real):real;
function fun_2(a, b:integer):real;
```

Тело функции состоит из раздела описаний53 (константы, типы, переменные, процедуры и функции, используемые в процедуре) и операторов языка, реализующих ее алгоритм. В теле функции всегда должен быть хотя бы один оператор, присваивающий значение имени функции.

Например:

```
function fun_2(a, b:integer):real;
begin
        fun_2:=(a+b)/2;
end;
```

Обращение к функции осуществляется по имени с указанием списка фактических параметров, разделенных запятой:

```
имя_функции (список_фактических_параметров);
```

Например:

```
y:=fun_1(1.28);
z:=fun_1(1.28)/2+fun_2(3,8);
```

ЗАДАЧА. Вводится последовательность из N целых чисел, найти среднее арифметическое совершенных чисел и среднее геометрическое простых чисел последовательности.

Для решения поставленной задачи понадобятся две функции:

prostoe – определяет, является ли число простым, аргумент функции целое число N, функция возвращает true (истина), если число простое и false (ложь) – в противном случае;

soversh – определяет, является ли число совершенным; входной параметр целое число N, функция возвращает true (истина), если число простое и false (ложь) – в противном случае.

```
//Функция, которая определяет простое число.
function prostoe(N:word):boolean;
        var i:word;
begin
        prostoe:=true;
        for i:=2 to N div 2 do
                if N mod i = 0 then
                        begin   prostoe:=false;   break;   end;
end;
```

```
//Функция, которая определяет совершенное число.
function soversh(N:word):boolean;
	var i:word; S:word;
begin
	soversh:=false;
	S:=0;
	for i:=1 to N div 2 do
		if N mod i =0 then S:=S+i;
		if S=N then soversh:=true;
end;
var
	X:word;
	K,kol_p,kol_s,i:byte;
	Sum,Pro:real;
begin
 //Начало основной программы.
//Ввод количества элементов.
	write('K='); readln(K);
	Sum:=0; //Переменная для накапливания суммы.
	Pro:=1; //Переменная для вычисления произвед.
	kol_p:=0; //Счетчик простых чисел.
	kol_s:=0; //Счетчик совершенных чисел.
	for i:=1 to K do
	begin
		//Ввод элемента последовательности.
		Writeln('X='); readln(X);
		//Если число простое,
		if prostoe(X) then
		begin
			Pro:=Pro*X;//выполнить операцию умножения,
			kol_p:=kol_p+1;//увеличить счетчик простых чисел.
		end;
		//Если число совершенное,
		if soversh(X) then
		begin
			Sum:=Sum+X;//выполнить операцию умножения,
			kol_s:=kol_s+1;//увеличить счетчик совершенных чисел.
		end;
	end;
//Если были найдены совершенные числа,
	if kol_s<> 0 then
	begin
		Sum:=Sum/kol_s;//вычислить среднее арифметическое.
		writeln('Среднее арифметическое совершенных чисел ', Sum:5:2);
	end
	else
	 //иначе вывести сообщение:
	writeln('Совершенных чисел в последовательности нет.');
	//Если были найдены простые числа,
	if kol_p<>0 then
	begin
		Pro:= exp(1/kol_p*ln(Pro));//вычислить среднее геометрическое.
		writeln('Среднее геометрическое простых чисел ',Pro:5:2);
	end
	else//иначе вывести сообщение:
	writeln('Простых чисел в последовательности нет');
end.
```

Индивидуальные задания

Выполнить любые 5 задач из приведенного списка

1. Написать программу, которая автоматизирует процесс перевода градусной меры угла в радианную и наоборот, в зависимости от выбора пользователя. То есть пользователь должен выбрать, как он будет вводить угол, в радианах или в градусах. Введет в радианах, ответ получит в градусах и, соответственно, введет в градусах, ответ получит в радианах.
2. Написать программу для решения уравнений:
   •линейное $ax+b=0$;
   •квадратное $ax^2+bx+c=0$;
   •кубическое $ax^3+bx^2+cx+d=0$.
3. Создать программу, которая выводит на экран таблицу значений функций f(x) и g(x).
4. Вводится последовательность целых чисел, 0 – конец последовательности. Определить, содержит ли последовательность хотя бы одно число, сумма цифр в котором равна их количеству. Создать процедуру, которая возвращает сумму и количество цифр в числе.
5. Вводится последовательность целых чисел, 0 – конец последовательности. Определить, содержит ли последовательность хотя бы одно совершенное число. Для определения совершенного числа создать функцию.
6. Вводится последовательность из N целых положительных элементов. Определить, содержит ли последовательность хотя бы одно простое число. Для определения простого числа создать функцию.
7. Вводится последовательность из N целых положительных элементов. Посчитать количество чисел палиндромов. Для определения палиндрома создать функцию.
8. Вводится последовательность из N целых положительных элементов. Подсчитать количество совершенных чисел в последовательности. Для определения совершенного числа создать функцию.
9. Поступает последовательность целых положительных чисел, 0 – конец последовательности. Определить, в каком из чисел больше всего делителей. Для подсчета делителей числа использовать функцию.
10. Поступает последовательность целых положительных чисел, 0 – конец последовательности. Определить, в каком из чисел больше всего цифр. Для подсчета количества цифр числа использовать функцию.
11. Вводится последовательность из N целых положительных элементов. Найти число с минимальным количеством цифр. Для определения количества цифр в числе использовать функцию.
12. Вводится последовательность из N целых элементов. Для всех положительных элементов последовательности вычислить значение факториала и вывести его на печать. Вычисление факториала оформить в виде функции.
13. Поступает последовательность целых положительных чисел, 0 – конец последовательности. Вывести на экран все числа последовательности, не являющиеся простыми, и их делители. Определение простого числа оформить в виде функций.
14. Вводится последовательность из N целых элементов. Вывести на экран все числа последовательности, являющиеся совершенными, и их делители. Определение совершенного числа оформить в виде функций.
15. Поступает последовательность целых положительных чисел, 0 – конец последовательности. Найти среднее арифметическое простых чисел в этой последовательности. Определение простого числа оформить в виде функций.
16. В последовательности из N целых положительных элементов найти число с наибольшим количеством нулей в своем представлении. Составить функцию для подсчета нулей в числе.
17. В последовательности из N целых положительных элементов найти сумму всех палиндромов. Для определения палиндрома создать функцию.
18. Поступает последовательность целых положительных чисел, 0 – конец последовательности. Посчитать количество элементов последовательности, имеющих в своем представлении цифру 0. Создать процедуру, возвращающую значение истина, если в числе есть нули, и ложь в противном случае.
19. Поступает последовательность целых положительных чисел, 0 – конец последовательности. Для каждого числа найти количество нулей и единиц. Создать процедуру, которая возвращает количество нулей и единиц в заданном числе.
20. Вводится последовательность из N целых элементов. Для каждого элемента последовательности найти среднее значение его цифр. Создать функцию для расчета среднего значения цифр в числе.
21. Поступает последовательность целых положительных чисел, 0 – конец последовательности. Определить количество цифр и наименьшую цифру для каждого числа последовательности. Написать

процедуру, которая для заданного числа возвращает два параметра: количество цифр в нем и наименьшую цифру.