электронный научно-технический журнал
ИНЖЕНЕРНЫЙ ВЕСТНИК
Издатель ФГБОУ ВПО "МГТУ им. Н.Э. Баумана". Эл No. ФС77-51036. ISSN 2307-0595

Особенности рационального использования пользовательских подпрограмм

 # 09, сентябрь 2015

Ваулин А. С.[1], к.т.н., доцент, Мартынюк Н. Н.[1,*]

УДК: 004.424

[1]Россия, МГТУ им. Н.Э. Баумана
[*] martynyuk1950@mail.ru

Введение

При разработке программ перед разработчиком стоят задачи создания программы, обладающей рядом свойств (массовость, результативность и др.) и, отвечающей требованиям (объем занимаемой памяти, быстродействие, наглядность и др.).

Одним из путей сокращения объема памяти программы и улучшению ее наглядности является рациональное применение пользовательских подпрограмм. Авторы не только уделили внимание принципам и методике использования пользовательских подпрограмм, но и предложили методику приближенной оценки объема программ.

Учитывая важность назначения параметров подпрограмм, авторы компактно и наглядно изложили классификацию параметров, оформив ее в виде таблицы.

В статье подчеркивается, что применение пользовательских подпрограмм оказывает влияние на стиль, качество, надежность и значительно сокращает объем программы.

Структура программы с использованием подпрограмм

В языке Паскаль используются подпрограммы в виде процедур и функций [1-9]. Процедура позволяет получить несколько результатов (или, в частном случае, ни одного результата) и возвращает результаты в программу через фактические параметры. Функция предназначена для получения одного результата, который связан с именем функции или предопределенной переменной Result.

```
Program Project1; // Заголовок программы;
//Разделы описаний глобальных параметров
Const  …;
Type  …;
Var  …;
    Procedure   PR(KX: integer; var Z: real); { Заголовок процедуры со  списком
  формальных параметров}
// Разделы описаний локальных параметров
```

```
Const …;
Type …;
Var …;

Procedure  // описание локальных процедур
Function  // описание локальных функций;
Begin
//тело процедуры;
End;

Function  FC(X: real; Y: real): real; // Заголовок  функции со списком формальных параметров
                                      // и типом результата функции
// Разделы описаний локальных параметров
Const …;
Type …;
Var …;
Procedure  // описание локальных процедур
Function  // описание локальных функций;

Begin
….;
FC:=<возвращаемый результат>;
End;

Begin //раздел операторов
  ….
  PR(K,X);   //обращение к процедуре
  …;
  PR(R,Y);   //обращение к процедуре
  …
   R:=FC(A,C);  …//обращение к функции
  …
End.
```

Обмен данными между программой и подпрограммой осуществляется посредством параметров.

Классификация параметров подпрограмм

Рациональное использование параметров подпрограмм позволяет сократить объем используемой памяти и время обработки программы. Установка соответствия между параметрами программ и подпрограмм возлагается на пользователя, поэтому необходимо

понимать взаимодействие параметров с входными и выходными данными и их влияние на работу программы при разных способах передачи [9].

В таблице 1 приведена классификация параметров подпрограмм.

Таблица 1. Классификация параметров подпрограмм

N пп	Признаки классификации параметров	Наименование параметров	Оформление параметров
1	По месту записи	Параметры: - формальные, - фактические.	Запись параметров: - в заголовке подпрограммы, - при вызове подпрограммы.
2	По области действия	Параметры: -локальные, - глобальные.	Область действия: - в подпрограмме, - в программе и подпрограмме.
3	По связи с данными	Параметры: - входные, - выходные, - входные/выходные.	Запись параметров: - без ключевого слова, - с ключевым словом - out, - с ключевым словом - var.
4	По способу передачи	Параметры: - значения, - константы, - переменные	Запись параметров: - без ключевого слова, - с ключевым словом- const, - с ключевым словом - var.
5	По использованию подпрограмм в качестве параметров	Параметр: - функция, - процедура	Запись: - function(список параметров), - procedure(список параметров)

Программирование с использованием процедур

Структура процедуры повторяет структуру программы и включает в себя: заголовок, разделы описаний, тело процедуры. В общем виде структура процедуры имеет вид:

```
Procedure <имя процедуры>(<список формальных параметров>);
  Const      //  описание локальных констант
  Type       //  описание локальных типов данных
  Var        //  описание локальных переменных
  Procedure // описание локальных (внутренних) процедур
  Function// описание локальных (внутренних) функций
Begin
        //  тело процедуры
End;
```

В список формальных параметров включаются входные и выходные параметры с указанием их типов. Параметры одного типа можно объединить в список (например: n1, n2, n3: integer). Если имена выходных формальных параметров имеют разный тип, то в списке каждому имени формального параметра предшествует слово Var или Out**,** при этом параметры разных типов разделяет точка с запятой (;).

При описании формальных параметров могут использоваться только простые типы или ранее определенные пользователем в разделе определения типов.

Например:

```
Const n=150;
Type Tmas=array[1..n] of real;
Procedure PR( Var  a:Tmas; n:integer; Var c1,c2:real; Out d:real);
```

Вызов процедуры осуществляется оператором процедуры в виде:

<имя_процедуры >(< список фактических параметров>);

Между фактическими параметрами при ее вызове и формальными параметрами в заголовке процедуры должно быть соответствие: количество, порядок следования и типы формальных и фактических параметров должны совпадать.

При обращении к процедуре выполняется замена формальных параметров на фактические параметры, которые передаются из вызывающей программы. В списке фактические параметры пишутся через запятую, без указания типов.

При выполнении оператора вызова процедуры работа основной, вызывающей, программы приостанавливается, начинается выполнение процедуры. После завершения работы процедуры выполнение основной программы возобновляется с оператора, который следует за оператором вызова процедуры.

Пример программы с использованием процедур

Определить наименьшую и наибольшую температуры каждого месяца одного квартала года и найти наименьшую и наибольшую из них. Температуры месяцев tm1, tm2, tm3 заданы одномерными массивами, а наименьшие и наибольшие температуры месяцев квартала формируются в массиве tminmax 123.

Рационально выделить типовые фрагменты обработки в процедуры: ввод массивов и поиск наименьшей и наибольшей температуры месяца.

```
program Project14;
{$APPTYPE CONSOLE}
uses
  SysUtils;
Const   nmax=31;
Type
Tmas=array[1.. nmax] of integer; // тип описания одномерного массива
Procedure VvodMas(var tx:Tmas; nx:integer); // Ввод массива температур месяца
Var i: integer;
begin
writeln(' Введите элементы массива: tx');
for i:=1 to nx do  read(tx[i]);
readln;
end;
```

```
Procedure PoiskMinMax(Const tx:Tmas; nx: integer; out txmin, txmax: integer);
//поиск минимальной и максимальной температуры месяца
Var i: integer;
begin
 txmin:=tx[1];
 txmax:=tx[1];
for i:=2 to nx do
begin
if tx[i]<txmin then txmin:=tx[i];
if tx[i]>txmax then txmax:=tx[i];
end;
end;

var   // объявление глобальных параметров
tm1, tm2, tm3,tminmax123:Tmas;
n1, n2, n3, i: integer;
tmin, tmax: integer;
begin // основная программа
 writeln(' Введите количество дней месяцев квартала n1,n2,n3');
readln(n1,n2,n3);
 //writeln('введите элементы массива tm1');
VvodMas(tm1,n1); //вызов процедуры ввода  массива температур 1-го месяца
//writeln('введите элементы массива tm2');
VvodMas(tm2,n2);//вызов процедуры ввода  массива температур 2-го месяца
//writeln('введите элементы массива tm3');
VvodMas(tm3,n3);      //вызов процедуры ввода  массива температур 3-го месяца
PoiskMinMax(tm1,n1,tmin, tmax); // вызов процедуры
 tminmax123[1]:=tmin; tminmax123[2]:=tmax;
 writeln('Минимальная и максимальная температура 1-го месяца');
 writeln('tmin=', tmin:3,' ':4, 'tmax=', tmax:3);
PoiskMinMax(tm2,n2,tmin, tmax); // вызов процедуры
 tminmax123[3]:=tmin; tminmax123[4]:=tmax;
 writeln('Минимальная и максимальная температура 2-го месяца');
 writeln('tmin=', tmin:3,' ':4, 'tmax=', tmax:3);
PoiskMinMax(tm3,n3,tmin, tmax); // вызов процедуры
 tminmax123[5]:=tmin; tminmax123[6]:=tmax;
  writeln('Минимальная и максимальная температура 3-го месяца');
 writeln('tmin=',tmin:3,' ':4,'tmax=', tmax:3);
PoiskMinMax(tminmax123,6,tmin,tmax);
writeln('Min и Max температуры квартала');
writeln('tmin=',tmin:3,' ':4,'tmax=', tmax:3);
```

```
readln;
end.
```

Программирование с использованием функций

Структура функции имеет вид:

```
Function <имя функции>(<список формальных параметров>):<тип результата>;
  Const      //  описание локальных констант
  Type       //  описание локальных типов данных
  Var        //  описание  локальных переменных
  Procedure // описание локальных процедур
  Function// описание локальных функций
 Begin
  //  тело функции
  <имя функции>:=<результат>;   или    Result:=<результат>;
 End;
```

Список формальных параметров функции обычно содержит только входные параметры или может не содержать параметров.

Тип результата может быть любой, кроме файлового типа. Результат обработки должен быть связан с именем функции или с переменной Result.

При использовании имени функции в теле функции должен быть хотя бы один оператор присваивания, в котором имени функции присваивается окончательный результат.

Переменная Result по умолчанию есть своя в каждой функции. Тип ее тот же, что тип результата. Она может использоваться так же, как любая переменная этого типа. Но при выходе из функции ей так же, как при использовании имени функции, должен быть присвоен окончательный результат.

Вызвать функцию можно в выражении, где допустим операнд такого же типа, как у функции. Вызов функции имеет следующий вид:

```
<имя_функции >(< список фактических параметров> )
```

Количество, порядок следования и тип формальных и фактических параметров должны совпадать.

Пример программы с использованием функции

Определить среднюю температуру каждого месяца одного квартала и найти среднюю из них. Температуры месяцев tm1, tm2, tm3 заданы одномерными массивами.

Рационально выделить типовые фрагменты обработки в процедуру ввода массива температур месяца и функцию вычисления средней температуры месяца.

```
Program SredTemp;
 $APPTYPE CONSOLE}
Uses SysUtils;
```

```
Const nmax=31;
Type    Tmas=array[1..nmax] of integer;
Procedure VvodMas(Var tx:Tmatr;  nx:integer);
// Ввод массива температур месяца
Var  i:integer;
begin
   writeln(’Введите массив ’);
   for j:=1 to nx do read(tx[i]);
   readln;
end;

Function Sred(Const tx:Tmas; nx:integer):real;
// Вычисление средней температуры с использованием переменной Result
Var  i,s:integer;
Begin
  s:=0;
  for i:=1 to nx do
  s:=s+tx[[i]
  Result:=s/nx;
end;

var   // объявление глобальных параметров
  tm1,tm2,tm3,tm123:Tmas;
  n1, n2, n3: integer;
  tsr1, tsr2, tsr3, tsr123: real;
begin // основная программа
  writeln(' Введите количество дней месяцев квартала n1,n2,n3');
   readln(n1,n2,n3);
  writeln('введите массив tm1:');
  VvodMatr(tm1,n1);
  writeln('введите массив tm2');
  VvodMatr(tm2,n2);
  writeln('введите массив tm3');
  VvodMatr(tm3,n3);
  tsr1:=Sred(tm1, n1);
  tm123[1]:= tsr1;
  writeln('Средняя температура 1-го месяца квартала ');
   writeln(' tsr1=', tsr1:5:2);
  tsr2:=Sred(tm2, n2);
  tm123[2]:= tsr1;
  writeln('Средняя температура 2-го месяца квартала ');
   writeln(' tsr2=', tsr2:5:2);
```

```
  tsr3:=Sredt(m3, n3);
  tm123[3]:= tsr3;
  writeln('Средняя температура 3-го месяца квартала ');
   writeln(' tsr3=', tsr3:5:2);
  tsr123:=Sred(tm123, 3);
  writeln('Средняя температура квартала');
   writeln(' tsr123=', tsr123:5:2);
  readln
 End.
```

Сравнительная оценка объема программ

Предлагаемая авторами приближенная оценка объема программ без подпрограмм и с использованием подпрограмм позволяет провести сравнительный анализ по объему занимаемой памяти программой.

Ниже приводятся выражения, позволяющие оценить объемы программ в строках.

Объем программы в строках без использования подпрограмм, считая, что в одной строке записывается один оператор, определяется выражением:

$$V_{пр} = V_{общ} + \sum_{i=1}^{n} (V_{фpi} * K_{фpi})$$

где:

$V_{пр}$ - количество строк программы без подпрограмм,

$V_{общ}$ - количество строк программы без повторяющихся фрагментов,

$V_{фpi}$ - количество строк i-го повторяющегося фрагмента,

$K_{фpi}$ - число повторений i-го фрагмента,

n - количество повторяющихся фрагментов.

Объем программы в строках с использованием подпрограмм определяется выражением:

$$V_{прп} = V_{общ} + \sum_{i=1}^{n} V_{пni} + \sum_{i=1}^{n} V_{обpi}$$

где:

$V_{прп}$ - количество строк программы с подпрограммами,

$V_{пni}$ - количество строк i-ой подпрограммы,

$V_{обpi}$ - количество строк операторов вызова i-ой подпрограммы.

Количество строк i-ой подпрограммы определяется выражением:

$$V_{пni} = V_{фpi} + V_{оформп}$$

где:

V_{nni} - число строк i-ой подпрограммы,

$V_{оформп}$ - число строк, необходимых для оформления подпрограммы (заголовок, описание локальных параметров, операторные скобки тела подпрограммы).

Используя данные выражения, можно оценить объем программы без использования процедур на приведенном выше примере:

$$V_{пр} = V_{общ} + V_{ппввод} * 3 + V_{ппmin\,max} * 4 = V_{общ} + 9 + 28 = V_{общ} + 37$$

С использованием процедур объем программы будет следующим:

$$V_{прп} = V_{общ} + V_{ппввод} + V_{ппmin\,max} + V_{обрввод} + V_{обрmin\,max} =$$
$$= V_{общ} + 7 + 11 + 3 + 4 = V_{общ} + 25$$

Пример программы с процедурами демонстрирует принципы организации программ с подпрограммами. Авторы акцентировали внимание на принципах организации, поэтому объем программы и подпрограмм сознательно сокращены. На практике объем подпрограмм исчисляется десятками строк, а сами подпрограммы многократно повторяются.

На основе проведенного анализа можно сделать выводы, что использование подпрограмм сокращает объем программы и улучшает ее наглядность. Сокращение объема программы при использовании подпрограмм будет тем значительнее, чем больше объем повторяющегося фрагмента и чем чаще он выполняется.

При использовании подпрограмм необходимо рационально выбирать размер подпрограммы, их количество в программе и использовать минимально связи между ними. Уменьшение размера подпрограммы снижает среднее количество ошибок в программе. С другой стороны, передача управления между многочисленными подпрограммами занимает немало времени. Поэтому не следует создавать программы с большим числом подпрограмм. Размер подпрограммы рекомендуется ограничивать 50-60 строками, при записи в каждой строке программы одного оператора языка программирования.

По возможности следует ослаблять связи между подпрограммами, что приводит к уменьшению распространения ошибок на другие подпрограммы. В связи с этим предпочтительнее использовать только локальные переменные, определяемые внутри подпрограммы. В этих случаях каждая подпрограмма не может влиять на данные других подпрограмм.

При разбиении задачи на подзадачи с выделением соответствующих подпрограмм большое значение приобретает с точки зрения предотвращения ошибок точное и краткое описание подзадачи в такой форме, чтобы это описание можно было бы рассматривать как задание на программирование.

Заключение

Рациональное использование подпрограмм позволяет значительно сократить объем программы. Эффект от сокращения программы будет напрямую зависеть от объема типовой обработки данных и количества необходимых обращений к подпрограмме. Улучшается наглядность программы, что в свою очередь приводит к ускорению ее разработки и отладки.

Список литературы

[1]. Алексеев В.Е., Ваулин А.С., Куров А.В. Практикум по программированию. Обработка числовых данных. Учеб. пособие. / под. ред. Б.Г. Трусова. М.: Изд-во МГТУ им. Баумана. 2008. 288 с.

[2]. Алексеев В.Е., Ваулин А.С., Петрова Г.Б. Вычислительная техника и программирование. Практикум по программированию. / Под ред. А.В. Петрова. М.: Высшая школа. 1991. 400 с.

[3]. Петров А.В., Алексеев В.Е., Ваулин А.С., Петрова Г.Б., Титов М.А., Шкатов П.Н. Вычислительная техника и программирование: учебник для технических вузов / под ред. А.В. Петрова. М.: Высшая школа. 1990. 479 с.

[4]. Агабеков Л.Е., Борисов С.В., Ваулин А.С., Козлов А.Д., Куров А.В., Серебрякова И.Л. Инструментальные средства персональных ЭВМ. В 10 кн. Кн.4. Программирование в среде Турбо-Паскаль: Учебное пособие для втузов / под ред. Б.Г. Трусова. М.: Высшая школа. 1993. 142 с.

[5]. Алексеев В.Е., Ваулин А.С. Электронные вычислительные машины: В 8 кн. Кн.7: Практикум по программированию. 2-е изд., перераб. и доп. / под ред. А.Я. Савельева. М.: Высшая школа. 1993. 207 с.

[6]. Ваулин А.С. Электронные вычислительные машины: В 8 кн. Кн. 5. Языки программирования: (Паскаль, ПЛ/М). 2-е изд., перераб. и доп. / под ред. А.Я. Савельева. М.: Высшая школа. 1993. 157 с.

[7]. Фаронов В.В. Программирование на персональных ЭВМ в среде Турбо-Паскаль. 2-е изд. М.: Изд-во МГТУ им. Баумана. 1992. 442 с.

[8]. Ваулин А.С., Криницына Л.Ф., Мартынюк Н.Н. Практикум по программированию в среде Delphi. Методические указания к выполнению заданий по курсу «Информатика». М.: Изд-во МГТУ им. Баумана. 2007. 37 с.

[9]. Ваулин А.С., Криницына Л.Ф., Мартынюк Н.Н. Программирование с использованием подпрограмм в среде DELPHI. / Электронное учебное издание, методические указания к лабораторным работам по дисциплине «Информатика». // ФГУП «Информрегистр». Текстовое (символьное) электронное издание. М.: МГТУ им. Н.Э. Баумана. 2011. Режим доступа: http://www.db.inforeg.ru/Inet/GetEzineByID/287698 (дата обращения: 2.09.15)