5. ПОДПРОГРАМИ

5.1. Прости и съставни програми

Програмите, които разглеждахме до тук ще наричаме прости програми. Програмите, които се разработват във връзка със задачи, които поставя практиката, обикновено са сложни и големи по обем. Всяка по-голяма задача обаче може да бъде формулирана като съвкупност от подзадачи. Често и самите подзадачи от своя страна, могат да бъдат разбити на подзадачи от по-ниско ниво и т.н.. Програмните езици от високо ниво, какъвто е и езикът Паскал, позволяват да се създадат *подпрограми* за решаването на отделните подзадачи и тези подпрограми да се обединят в единна програма. Програма, изградена по този начин, е *съставна програма*. Тя се състои от множество програмни единици, като една от тях, тази която ги обединява, е главна програмна единица и обикновено за краткост се нарича главна програма. Всички останали програмни единици са подпрограми и могат да се изпълняват само като компоненти на някаква съставна програма, т.е. подпрограмите не могат да се изпълняват самостоятелно.

Съставната програма се структурира, като някои подпрограми се влагат пряко в главната програма, други се влагат във вложените и т. н. Дълбочината на вложеност не е ограничена.

Този подход на изграждане на програмите има редица предимства:

- създаването и настройката на отделните подпрограми могат да се възложат на различни специалисти и по този начин силно да се съкрати времето, необходимото за създаването на цялата програма;
- рязко се повишава прегледността на програмите и с това се намалява броят на допусканите грешки;
- повтарящи се участъци от програмата се записват еднократно като подпрограма и след това всеки такъв участък се замества с едно обръщение към подпрограмата;
- създава се възможност да се изграждат библиотеки от готови подпрограми, които силно съкращават работата на програмистите при писане на нови програми.

В програмите на Паскал могат да се създават два вида подпрограми: подпрограма-функция и подпрограма-процедура. За краткост е прието те да се наричат съответно *функция и процедура*.

Подпрограми са например множеството вградени функции и процедури, които ние вече познаваме и сме ползвали. Такива са: abs(x), sqr(x), sqrt(x), sin(x), cos(x), arctan(x), exp(x), ln(x), trunc(x), round(x), odd(x), chr(x), ord(x), pred(x), succ(x), read, readln, write, writeln.

Освен вградените подпрограми, Турбо Паскал предлага още много готови (стандартни) подпрограми, обединени съобразно предназначението си в отделни библиотеки, наречени стандартни библиотечни модули. На някои от тях специално ще се спрем по-късно. Независимо от наличието на вградени и стандартни подпрограми, на програмистите често се налага да създават собствени подпрограми и собствени библиотечни модули. Настоящата глава е посветена на създаването на собствени подпрограми.

5.2. Структура на подпрограмите

Както вече отбелязахме, в програмите на Паскал могат да се създават два вида подпрограми: *функция и процедура*.

Подпрограмите на Паскал (функции и процедури) имат следната структура:

- заглавие;
- част I - дефиниране на обекти;
- част II - описание на операциите (действията).

Тази структура се вижда и от следните две примерни подпрограми:

1. Функция, която връща по-голямата от две зададени стойности, има вида:

```
Function Max2 (X, Y:real):real;       {Заглавие}
{Част I - описание на локалните обекти}
 Var
   Z:real;
{ Част II - Описание на операциите в функцията Max2 }
 Begin
   If X > Y
     then  Z:=X
     else  Z:=Y;
   Max2:=Z
End;
```

2. Процедура, която разменя стойностите на две зададени променливи, има вида:

```
Procedure Change (Var X, Y:real);          {Заглавие}
{Част I - описание на локалните обекти}
 Var Z:real;
{ Част II - Описание на операциите в процедурата Change }
 Begin
   Z:=X; X:=Y; Y:=Z
End;
```

Нека да си припомним, че и простата програма има същите компоненти (заглавие, описание на обекти и описание на операции), но съществуват две съществени разлики:

- заглавието на подпрограмата е задължително, няма подпрограма без заглавие;
- подпрограмата завършва със символа точка и запетая, а не с точка.

5.2.1. Заглавие на подпрограма

Заглавието на подпрограмата, когато тя е подпрограма-функция, има вида:

Function *Име на функцията (Описание на фиктивните параметри): Тип на функцията;*

а когато тя е подпрограма-процедура, вида му е:

Procedure *Име на процедурата(Описание на фиктивните параметри);*

Описанието на фиктивните параметри представлява списък от компоненти, разделени с точка и запетая (;), а компонентите могат да имат вида:

Списък от имена на фиктивни параметри: Име на типа на фиктивните параметри;

или

Var *Списък от имена на фиктивни параметри : Име на типа на фиктивните параметри;*

като имената на фиктивните параметри в списъка се разделят със запетая, а името на типа може да бъде име на стандартен тип или име на вече дефиниран

нестандартен тип, например, както е показано в следния пример

i, j : integer; **Var** k,m : integer; Ch: char; **Var** A,B,C : real; P,Q : TipMas,

където TipMas е някакъв нестандартен тип, дефиниран в програмна единица, в която е вложена пряко или непряко подпрограмата с горните фиктивни параметри.

Както ще видим по-късно, при обръщение към подпрограмата те се заместват с действителни параметри и по този начин се обменят данни между програмните единици.

Фиктивните параметри, предшествани от **Var**, се наричат **параметри-променливи**. При обръщение към подпрограмата те се заместват с действителни параметри от същия тип, които могат да бъдат променливи или компоненти на съставни променливи. На фиктивните параметри-променливи може да се гледа като на псевдоними на действителните параметри, защото при изпълнението на подпрограмата компютърът работи с действителните параметри, но чрез техните псевдоними. Стойност, присвоена на псевдонима, е стойност и на съответната действителна променлива, включително и след връщане от подпрограмата. Поради това параметрите-променливи могат да се използват както за предаване на данни (стойности) на подпрограмата, така и за връщане на резултати (стойности) от подпрограмата, т.е. параметрите-променливи могат да се използват като входни, изходни и входно-изходни параметри.

Фиктивните параметри, непредшествани от **Var**, се наричат **параметри-стойности**. При обръщение към подпрограмата те се заместват с действителни параметри от същия тип, които могат да бъдат изрази (константата, променливата и обръщението към функция са частен случай на израз). При изпълнение на подпрограмата, компютърът разглежда стойностите на действителните параметри като начални стойности на параметрите-стойности. След връщане от подпрограмата, стойностите, присвоени на параметрите-стойности, не се запазват и не са достъпни за програмната единица, която се е обърнала към подпрограмата. Поради това параметрите-стойности могат да се използват само за предаване на данни (стойности) на подпрограмата, т.е. те могат да се използват само като входни параметри.

Заглавието на подпрограмата е важно, защото чрез него се указват:

- видът на подпрограмата (процедура или функция);
- името на подпрограмата - то се избира от програмиста по правилата за избиране имена на обекти;
- фиктивните параметри;
- типът на функцията - той може да е някой от шестте прости типове (реален, целочислен, логически, символен, изброен и ограничен) и типа символен низ.

5.2.2. Дефиниране на обектите

Частта описание на обектите на една подпрограма може да включва същите раздели като простата програма:

- дефиниране на етикети;
- дефиниране на константи;
- дефиниране на типове;
- дефиниране на променливи;
- вложени подпрограми.

При дефинирането на константи, типове и променливи могат да се използват константи и типове, описани в програмните единици, в които е вложена подпрограмата.

Всички константи, типове и променливи, описани в дадена подпрограма, са

нейни локални (собствени) обекти. Те са използваеми както в нея, така и във всички подпрограми, вложени в нея, но за вложените те са нелокални (несобствени).

5.2.3. Описание на операциите (действията)

Частта описание на операциите съдържа операторите, които трябва да се изпълнят, за да се получи резултатът, очакван от подпрограмата, те описват алгоритъма със средствата на езика Паскал. Съществено ново тук е, че докато в операторите на простата програма участват само нейните обекти (дефинирани в самата нея), в операторите на подпрограмата участват:

- **собствените фиктивни параметри** - такива са променливите, описани (декларирани) в заглавието на подпрограмата, те могат да бъдат разглеждани като предадени на подпрограмата;
- **собствените обекти** - такива са обектите (етикети, константи и променливи), дефинирани в подпрограмата;
- **несобствени фиктивни параметри** - това са фиктивните параметри на подпрограмите, в които е вложена пряко или косвено подпрограмата;
- **несобствените обекти** - такива са константите и променливите, описани в главната програма или в подпрограмите, в които подпрограмата е вложена пряко или непряко. Несобствен обект става недостъпен както за дадена подпрограма, така и за всички вложени в нея подпрограми, когато в подпрограмата се дефинира собствен обект с името на несобствения. За тях е достъпен ново дефинираният обект със същото име.

5.3. Структура на съставна програма

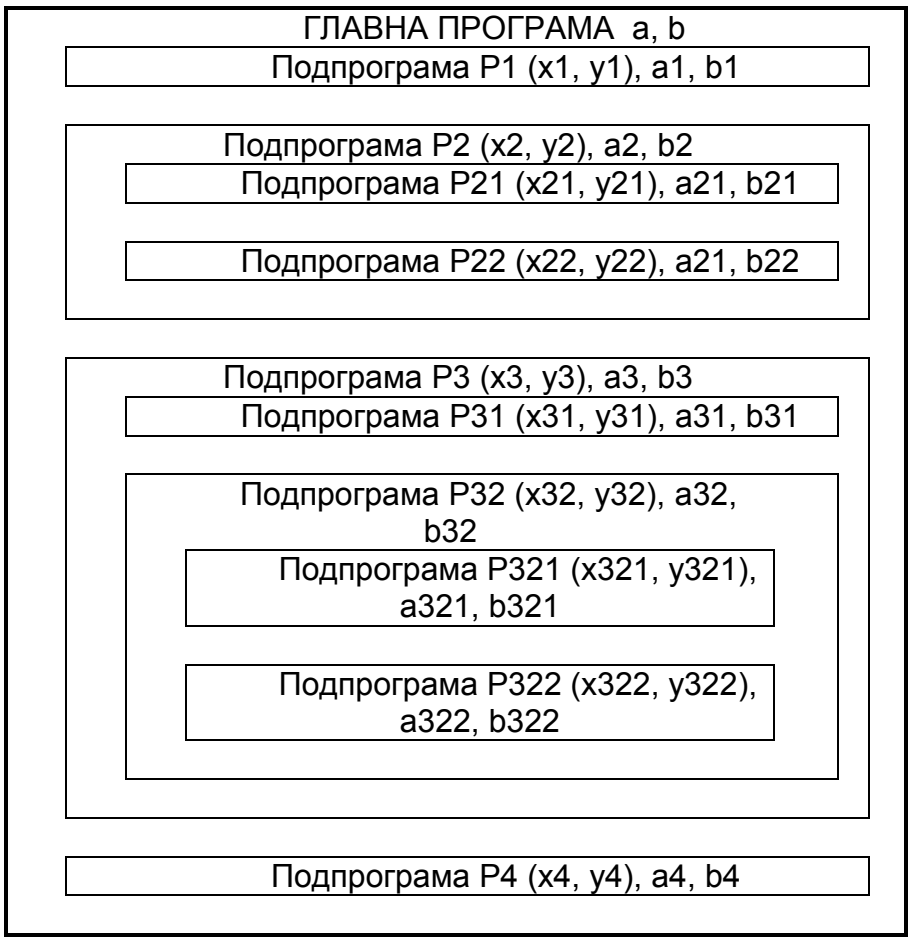
Да предположим, че искаме да създадем съставна програма със следната организация:

- в главната програма са вложени подпрограмите P1, P2, P3 и P4;
- в подпрограмата P2 са вложени подпрограмите P21 и P22;
- в подпрограмата P3 са вложени подпрограмите P31 и P32;
- в подпрограмата P32 са вложени подпрограмите P321 и P322.

Текстовете на подпрограмите на тази програма трябва да разположим както следва:

- текстовете на подпрограмите P1, P2, P3 и P4 - в описателната част на главната програма**;**
- текстовете на подпрограмите P21 и P22 - в описателната част на подпрограмата P2;
- текстовете на подпрограмите P31 и P32 - в описателната част на подпрограмата P3;
- текстовете на P321 и P322 - в описателната част на P32.

Организацията на тази програма е показано на Фиг.5.1, където в скоби са посочени фиктивните параметри, а след скобите - собствените обекти на съответната подпрограма.



Фиг.5.1

5.4. Обръщение към подпрограма. Действителни параметри

Обръщението към подпрограма или извикването на подпрограма има вида

Име на подпрограмата(Списък от действителни параметри)

Действителните параметри са разделени със запетаи и трябва да съответстват на фиктивните по брой, по място, по тип и по вид.

Обръщението към функция може да представлява елемент от израз или израз в оператор, когато функцията има стойност, т.е. когато във функцията на името на функцията е присвоена стойност. Обръщението към функция, на чието име не се присвоява стойност, както и обръщението към процедура, представлява оператор и се нарича оператор-обръщение.

Програма 5.1. Програма, която въвежда стойностите на четири променливи A, B, P и Q и извършва следните операции:

- извежда по-голямата от стойностите на A и B;
- извежда по-голямата от сумите A + B и P + Q;
- намира най-голямата от стойностите на A, B, P и Q, присвоява я на R и извежда R;
- разменя стойностите на A и B.

```
{ Главна програма - дефиниране на нейните променливи }
Var
  A, B, P, Q, R: real;
```

```
{Подпрограма-функция, която връща по-голямата от две зададени стойности}
Function Max2 (X, Y:real):real;
 Var Z:real;          {Дефиниране на локална променлива Z }
 Begin       {Описание на операциите в подпрограмата-функция Max2 }
   If X > Y
     then Z:=X
     else Z:=Y;
   Max2:=Z
End;
```

{Подпрограма-процедура, която разменя стойностите на две зададени променливи}

```
Procedure Change (Var X, Y:real);
 Var Z:real;          {Дефиниране на локална променлива Z }
 Begin       {Описание на операциите в подпрограмата-процедура Change }
   Z:=X; X:=Y; Y:=Z;
End;
{ Описание на операциите в главната програма}
Begin
   Write('A='); Readln (A);
   Write('B='); Readln (B);
   Writeln ( 'По-голямата от  стойности на A и B е ', Max2(A,B):5:2);
   Write('P='); Readln (P);
   Write('Q='); Readln (Q);
   Writeln ( 'По-голямата от двете суми A+B и P+Q е ', Max2(A+B,P+Q):5:2);
   R:= Max2(Max2(A,B), Max2(P,Q);
   Writeln ( 'Най-голямата от стойности на A, B, P и Q е ', R:5:2);
   Writeln ( 'Стойностите на A и B преди размяната са: ','A=', A:5:2, '  B=',B:5:2);
   Change(A,B);
   Writeln ( 'Стойностите на A и B след размяната са: ','A=', A:5:2, '  B=',B:5:2);
   Readln
End.
```

Програмата е реализирана като съставна и се състои от главна програма и две вложени в нея подпрограми - функцията Max2, която връща по-голямата от две зададени стойности, и процедурата Change, която разменя стойностите на две зададени променливи.

Първата подпрограма е функция, защото връща една стойност (по-голямата от двете зададени) и върнатата стойност е от тип real, който е допустим като тип на функция. Втората подпрограма е процедура, защото връща повече от една стойности - тя връща разменените стойности на две зададени променливи.

Фиктивните параметри на функцията Max2 са от вида параметри-стойности, защото се ползват само като входни параметри. Това обаче дава възможност те да бъдат заместени с действителни параметри изрази, което, както е показано в главната програма, създава известни улеснения за програмиста.

Фиктивните параметри на процедура Change са от вида параметри-променливи (псевдоними), защото се използват като входно-изходни параметри. Те обаче не могат да бъдат заместени с действителни параметри изрази, а само с действителни параметри променливи.

В главната програма са използвани различни възможности за обръщение към функцията и с това е показано, че програмистът винаги, когато е възможно, трябва да прави подпрограмата си подпрограма-функция.

5.5. Достъпност на обектите

За операторите от дадена подпрограма са достъпни:

- собствените обекти - типовете, константите и променливите, декларирани в самата нея;
- собствените фиктивните параметри на подпрограмата;
- несобствени обекти - типовете, константите и променливите, декларирани както в главната програма, така и във всички подпрограми, в които е вложена пряко или непряко нашата подпрограма;
- несобствените фиктивни параметри - фиктивните параметри на всички подпрограми, в които е вложена пряко или непряко нашата подпрограма.

В следващата по-долу таблица са дадени несобствените обекти и несобствените фиктивни параметри за всяка подпрограма от примерната програма, дадена на Фиг.5.1.

Нека да не забравяме, че несобствен обект или фиктивен параметър става недостъпен както за дадена подпрограма, така и за всички вложени в нея подпрограми, когато в подпрограмата се дефинира собствен обект с името на несобствения. За тях е достъпен ново дефинираният обект със същото име. Например, за подпрограма P31 няма да са несобствен a3 и b, ако в нея са дефинирани променливи и/или фиктивни параметри с имена a3 и b.

Програмна единица	Несобствени обекти и фиктивни параметри
P1	a, b
P2	a, b
P21	a, b, a2, b2, x2, y2
P22	a, b, a2, b2, x2, y2
P3	a, b
P31	a, b, a3, b3, x3, y3
P32	a, b, a3, b3, x3, y3
P321	a, b, a3, b3, x3, y3, a32, b32, x32, y32
P322	a, b, a3, b3, x3, y3, a32, b32, x32, y32
P4	a, b

5.6. Достъпност на подпрограмите

Подпрограмата P от дадена програма може да се обръща към следните подпрограми:

- подпрограми, които са пряко вложени в P;
- към себе си;
- подпрограми, които са вложени заедно с P в някаква подпрограма, но са разположени пред P;
- подпрограми, в които е вложена P пряко или непряко;
- подпрограми, вложени заедно с подпрограма Q, в която е вложена пряко или косвено P, но при влагането са разположени пред Q;

За примерната програма, дадена на Фиг.5.1, допустими са обръщенията посочени в следната таблица:

Програмна единица	Допустими обръщения за програмната единица
Главна програма	P1, P2, P3, P4
P1	P1
P2	P1, P2, P21, P22
P21	P1, P2, P21

P22	P1, P2, P21, P22
P3	P1, P2, P3, P31, P32
P31	P1, P2, P3, P31
P32	P1, P2, P3, P31, P32, P321, P322
P321	P1, P2, P3, P31, P32, P321
P322	P1, P2, P3, P31, P32, P321, P322
P4	P1, P2, P3, P4

5.7. Изпълнение на съставна програма

При изпълнение на съставни програми оперативната памет най-общо се разделя на следните четири основни области:

- област на програмния код;
- област на статичните обекти;
- област на собствени обекти и фиктивните параметри на подпрограмите (системен стек);
- област на динамичните обекти.

Статични са обектите, декларирани в главната програма. Те остават в областта на статичните обекти през цялото време на изпълнение на програмата и затова са достъпни за операторите във всички програмни единици.

Собствени са всички обекти, декларирани в описателните части на подпрограмите. Един обект е собствен само за подпрограмата, в която е дефиниран, за всички вложени в нея подпрограми той вече е несобствен, но мястото му е в системния стек.

Динамичните обекти се създават и унищожават по време на изпълнението на програмата. В този учебник с тях няма да се занимаваме.

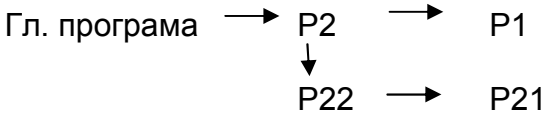
Стекът често се сравнява с куп чинии, защото когато към купа добавяме нова чиния, я разполагаме най-отгоре, когато от купа вземаме чиния, вземаме най-горната.

Системният стек е област от оперативната памет, която започва от определен адрес, с който се фиксира неговото “дъно”. Запълването на системния стек започва от дъното. При всяко обръщение към подпрограма в системния стек се отделя място за:

- собствените обекти на подпрограмата;
- фиктивните параметри на програмата. В място за фиктивен параметър-стойност се записва стойността на съответния действителен параметър, а в място за фиктивен параметър-променлива се записва адресът на променливата действителен параметър. Затова фиктивните параметри-стойности се наричат още *параметри, предавани по стойност*, фиктивни параметри-променливи - *параметри, предавани по адрес.*
- адресът, на който трябва да се предаде управлението след връщане от подпрограмата, за да продължи изпълнението на програмата.

При излизане от подпрограмата мястото в стека, което е било заето от нейни обекти се освобождава.

Да допуснем, че в процеса на изпълнението на програмата от Фиг.5.1, е реализирана следната последователност на обръщения:



При обръщението на главната програма към P2 в системния стек се отделя място за нейните нужди и започва изпълнението на операторите й, които

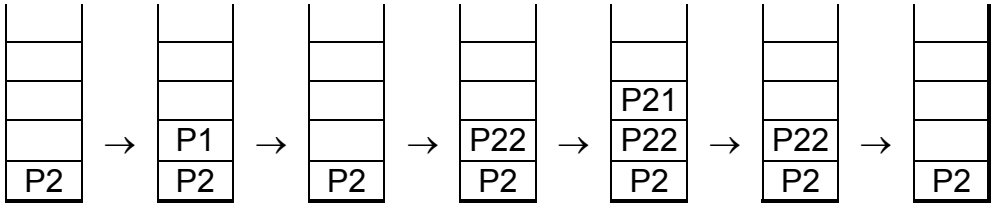
включват обръщение към P1. При реализацията на това обръщение в системния стек се отделя място за нуждите на P1. Следва изпълнение на операторите на P1 следва излизане от P1 и освобождаване на мястото в стека, заемано от нея. Следва изпълнение на следващите оператори на P2 и те трябва да съдържат обръщение към P22. При реализацията на това обръщение в системния стек се обособява място за нуждите на P22 и започва изпълнението на операторите на P22. Те следва да съдържат обръщение към P21. При реализацията на това обръщение в системния стек се обособява място за нуждите на P21 и започва изпълнението на операторите на P21. След приключване на изпълнението им следва излизане от P21 и освобождаване на мястото в стека, заемано от нея. Следва завършване изпълнението на P22, излизане от нея и освобождаване на заеманото от нея място в стека. По-нататък следва завършване изпълнението на P2, излизане от нея и освобождаване на мястото в стека, заемано от нея. Сега вече стекът е празен и се изпълняват операторите от главната програма, които могат да съдържат и нови обръщения към подпрограми. Всичко това е показано на Фиг.5.2.

В някои версии на езика (например в Турбо Паскал) изпълнението на подпрограмата може да бъде прекратено и да се излезе от нея преди тя да е изпълнена до края. Това може да стане посредством процедурата **Exit**.

Например операторът

If X < 0 **then** Exit

предизвиква излизане от изпълняваната подпрограма, ако X е по-малко от 0 и операторите до края на подпрограмата остават неизпълнени.



Фиг.5.2

5.8. Процедура или функция?

Всеки път, когато на програмистът му се налага да направи подпрограма, той трябва да вземе решение каква да е тя - функция или процедура. Отговорът на този въпрос следва да се търси в разликата между процедурата и функцията. А съществената разлика е следната:

- всяко обръщение към функция се замества със стойността на функцията - онази стойност, която е присвоена в изпълнимата част на функцията на името й. Това позволява обръщението към функцията да присъства в изрази или самото то да се явява в ролята на израз в някакъв оператор, а това опростява описанието на операциите в програмните единици. Типът на присвоената стойност обаче трябва да съвпада с типа на функцията, а той може да бъде някой от простите типове или тип символен низ. Ако на името на функцията не се присвоява стойност, за тип на функцията може да се посочи кой да е от позволените, но тогава обръщението не се замества със случайна стойност и следователно присъствието му в изрази е недопустимо, т.е. в този случай функцията е равностойна на процедура.

- обръщението към процедурата не се замества със стойност (понятие стойност на процедурата изобщо не съществува). Поради това обръщението към

процедурата не може да присъства в никакви оператори. То винаги е отделен оператор.

От това сравнение става ясно, че винаги, когато е възможно, трябва да даваме предпочитание на функцията. Но нека да си припомним, че подпрограмата може да е функция само, когато тя връща една единствена стойност и тази стойност е от скаларен тип или тип символен тип.

5.9. Фиктивни параметри или несобствени променливи и константи?

Може да се напише съставна програма, подпрограмите на която са без фиктивни параметри, защото всички обекти на главната програма и подпрограмите, в които тя е вложена са достъпни за нея. Следната програма е такъв вариант на програма 5.1, разгледана по-горе.

Програма 5.2.

```
Var
  A,B,P,Q,R,S X,Y,C:real;
Function Max2:real;
 Var Z:real;
 Begin
   If X > Y
     then Z:=X
     else Z:=Y;
   Max2:=Z
 End;
Procedure Change;
 Var Z:real;
 Begin
   Z:=X; X:=Y; Y:=Z;
 End;
Begin
   Write('A='); Readln (A);
   Write('B='); Readln (B);
   X:=A; Y:=B;
   Writeln ( 'По-голямата от  стойностите на A и B е ', Max2:5:2);
   Write('P='); Readln (P);
   Write('Q='); Readln (Q);
   X:=A+B; Y:=P+Q;
   Writeln ( 'По-голямата от двете суми A+B и P+Q е ', Max2:5:2);
   X:=A; Y:=B; R:=Max2;
   X:=P; Y:=Q; S:=Max2;
   X:=R; Y:=S; R:= Max2;
   Writeln ('Най-голямата от стойности на A, B, P и Q е ', R:5:2);
   Writeln ('Преди размяната: ','A=', A:5:2, '  B=',B:5:2);
   X:=A; Y:=B;
   Change;
   A:=X; B:=Y;
   Writeln ('След размяната: ','A=', A:5:2, '  B=',B:5:2);
   Readln
End.
```

Тази възможност на пръв поглед изглежда по-проста и поради това е по-примамлива, но подпрограмите, използващи несобствени променливи, са по-неудобни за включване в нови програми, защото изискват съгласуване на имената на несобствените променливи с програмната единица, която се обръща към тях.

Както се вижда от горната програма, подпрограмите са малко по-прости, но за сметка на това главната програма е значително по-сложна (Защо?). Това показва, че програмистите трябва да предпочитат подпрограмите с параметри.

Програма 5.3. Програма за създаване на таблица със стойностите на k-тата степен на числата от An до Ak със стъпка s, където An и Ak са зададени и An < Ak, а к може да има цяла, не цяла, положителна или отрицателна стойност . Редовете в таблицата да са разделени с хоризонтална линия.

Програмата е изградена като съставна, т.е. като главна програма с две вложени в нея подпрограми. Те са:

- функция Stepen за изчисляване стойностите на функцията $y=a^k$, като се използва това, че $a^k = e^{k.\ln(a)}$, и стандартните функции ln(x) и exp(x);
- процедура HorLin, извеждаща хоризонтална отсечка на екрана.

```
Var
  a, An, Ak, s, k: real;
{Функция за повдигане на реално число на реална степен }
Function Stepen(Osnova, StepPok : real) : real;
Const Eps = 1E-7;
Begin
  If abs(Osnova) < Eps
    then Stepen := 0
    else if abs(StepPok) < Eps
           then Stepen := 1
           else Stepen := Exp(StepPok * Ln(Osnova))
End;
{Процедура за изобразяване на хоризонтална отсечка от позиция M до N}
Procedure HorLin(M, N : integer);
Var i : integer;
Begin
  Write(' ':M-1);
  For i := M to N do Write('_');
  WriteLn;
End;
Begin
  Write('Задайте начална стойност: '); ReadLn( An );
  Write('Задайте крайна стойност: '); ReadLn(Ak);
  Write('Задайте стъпка: '); ReadLn( s );
  Write(' Задайте степенния показател: '); ReadLn(k);
  Writeln;
  WriteLn('Число':7,' ','Степен ':11);
  HorLin(2,20);
  a:=An;
  Repeat
    WriteLn(a:6:2,' ', Stepen(a,k):12:5);
    HorLin(2,20);
    a:=a+s
  until a > Ak + 0.00001;
  Readln
End.
```

Програма 5.4. Програма за намиране броя на дните между зададена начална дата (Dn,Mn,Gn) и зададена крайна дата (Dk,Mk,Gk).

Такава програмата беше създадена в гл.4 (Програма 4.5). Тук обаче програмата е изградена като съставна, т.е. тя се състои от:

- главна програма;

- функция BroDni, която е вложена в главна програма;
- функция BrDniMes, която от своя страна е вложена във функцията BroDni.

И двете подпрограми са функции защото връщат по едно цяло число. Те не използват нелокални обекти, за да е възможно включването и в други програми без съгласуване на имена с имената на нелокалните обекти.

```
Var Dn,Mn,Gn,Dk,Mk,Gk,BroiDni:integer;
{ Процедура BroDni за определяне броя на дните между двете дати}
Function BroDni(D1,M1,G1,D2,M2,G2:integer) : integer;
 Var M, G, BrDni:integer;
{ Функция BrDniMes за намиране броя на дните на зададен месец, вложена в процедурата BroDni }
 Function BrDniMes(Mes, God : integer) : integer;
   Var B:integer;
   Begin
     Case Mes of
       1,3,5,7,8,10,12:        B:=31;
       4,6,9,11:               B:=30;
       2:       If (God mod 400=0) or (God mod 4=0) and (God mod 100<>0)
                       then B:=29
                       else B:=28
     end;
     BrDniMes:=B;
     End;
{Край на функцията BrDniMes}
Begin {Начало на описанието на операциите в процедурата BroDni}
   G:=G1; M:=M1; BrDni:=0;
   Repeat
     BrDni:=BrDni+BrDniMes(M, G);
     M:=M+1;
     If M=13
       then begin G:=G+1;M:=1 end
   until (G=G2) and (M=M2);
   BroDni:=BrDni+D2-D1
End;
{Край на процедурата BroDni}
{Начало на описанието на действията в главната програма}
Begin
   Writeln('Въведете началната дата: ');
   Write('  - ден: '); Readln(Dn);
   Write('  - месец: '); Readln(Mn);
   Write('  - година: '); Readln(Gn);
   Writeln('Въведете крайната дата: ');
   Write('  - ден: '); Readln(Dk);
   Write('  - месец: '); Readln(Mk);
   Write('  - година: '); Readln(Gk);
   BroiDni:=BroDni(Dn,Mn,Gn,Dk,Mk,Gk);
   Writeln('От ',Dn,'.',Mn,'.',Gn,' до ',Dk,'.',Mk,'.',Gk,' има ',BroiDni,' дни.');
   Readln
End.
```

В тази програма функцията BrDniMes е вложена във функцията BroDni, но програмата ще работи и ако BrDniMes е разположена в главната програма, но пред функцията BroDni.

Програма 5.5. Програма, която извежда всички числа в интервала M..N, чиито цифри образуват строго нарастваща последователност.

Програмата е с подпрограма-функция от логически тип, която връща стойност True, когато цифрите на числото образуват строго нарастваща последователност и стойност False, когато цифрите на числото не образуват строго нарастваща последователност.

```
Var
  i,M,N:integer;
Function Narastvane(K:integer):boolean;
 Var
  c1,c2:integer;
 Begin
    c1:=K mod 10;
    Repeat
      c2:=c1;
      K:=K div 10;
      c1:=K mod 10;
    until (K=0) or (c1>=c2);
    If K=0
      then Narastvane:=true
      else Narastvane:=false
  End;
Begin
  Write('Задайте началното число: '); Readln(M);
  Write('Задайте крайното число: '); Readln(N);
  For i:=M to N do
    If Narastvane(i) then Writeln(i);
  Readln
End.
```

Препоръчваме на читателя да се опита да реализира горните програми, като навсякъде замени функцията с процедура и процедурата с функция.