Подпрограми

1.Тяло на подпрограмата.

Тялото е блок и може да включва всички раздели на един блок. Включително може да съдържа и раздел на процедури и функции. Те са вложени в дефинираната подпрограма. Всички имена които се създават в блока са локални в него и вън от блока не се виждат (използват).

Правила за видимост съвпадат с правилата за видимост в pascal, а именно : всяко име действа от мястото на описанието напред в блока до края на блока, който съдържа това описание. Името се вижда и във вложените блокове освен ако в някой вложен блок това име се опише отново. От това локално описание с предимство и временно засенчва предимството на по-глобалното описание. След изхода от вложения блок действието на глобалния блок се възстановява.

В тялото на подпрограмата може да се работи и с имена, които са дефинирани в обхващащия блок или постъпват в блока от интерфейсната част на някакъв unit. Ако подпрограмата има формални параметри, счита се, че те имат статус на локални данни, поради което локални имена не могат да съвпадат с имената на локални параметри.

Разделът на операторите реализира алгоритъма на подпрограмата. Нормално се счита, че физическият край на този раздел съвпада с логическия файл на този алгоритъм. Винаги може да се използва exit, който причинява незабавен край на изпълнението й и връщане на управлението във викащата среда. Използването на exit се счита по принцип за неправилно.

Тялото на процедурите и функциите се различава само по едно нещо – функцията трябва да върне стойност на мястото на извикване (тук може да се използва и функция като процедура (стойността се игнорира). Това се осъществява като на името на функцията някъде в тялото на функцията трябва да се присвои стойност. Последно присвоената стойност е онази, която се връща. Освен това предварително дефинирана

тук е стандартна променлива result. Тя изпълнява аналогично ф-я като името на result е известно на компилатора, не трябва да се дефинира. Result се държи винаги като променлива, ако е от дясно на присвояваната стойност, ако е отляво на присвояваната стойност получава стойност. Дори при турбо паскал ф-иите са скаларни. Типа на връщаната стойност е без ограничения – не може да връща само:

- инстанции на старите тип Object тип и техни производни;

- файлови типове;

Всичко друго може да се връща.

```
Type

Complex=record

X,y : real;

End;

Function sum(c1,c2:complex):complex;

Begin

Result.x=c1.x+c2.x;
```

Result.y=c1.y+c2.y

End;

Формални параметри.

Има 4 вида формални параметри :

- параметри стойности. Те напълно се покриват с понятието от паскал. За такива формални параметри може да бъде заместен съвместим израз в качество на аргумент. Този израз се изчислява и от получената стойност се получава екземпляр на форм. параметър. В подпрограмата се губи абсолютно всякаква връзка по отношение на аргумента с викащата среда, ако такъв формален параметър се отделя върху тялото. Обработката се извършва в/у екземпляра следователно на самия аргумент не се влияе по никакъв начин. Получава се разкъсване м/у аргумента и формалния параметър – ако се промени параметъра, аргумента си стои без промяна. Формален параметър предходен от служебната дума var е променлив. Аргумент за такъв формален параметър може да бъде само данна с разпределена памет. Подпрограмата получава адреса на тази памет (извършва се т.нар. заместване по адрес), които адрес се използва за реализиране на достъп – прави се косвено адресиране. Връзката м/у викащата среда и формалния параметър съществува. Var параметрите могат да бъдат използвани за внасяне на данни в подпрограмите и за изнасяне на резултат от нея. Параметри, които са файлови променливи или структурирани типове, които са файлови променливи задължително се заместват като формални параметри.

- Формални параметри предхождани от служебна дума const – има свойства на параметри променливи, но не може да бъде променен. Счита се че е само за четене. Освен това е забранено да бъде предавана (извикана) в подпрограмата друга подпрограма, освен като параметри const или параметри стойности;

- Формалният параметър out е също параметър, който се замества с адрес. Чрез него не могат да бъдат внесени данни в подпрограмата. Out параметъра е само за

извеждане на данни от подпрограмата. Параметри, които са стойности задължително трябва да имат тип. Параметри var, const и out могат да бъдат безтипови.

Пример : procedure abc(var x,y);

Използва се факта, че безтиповите параметри се заместват по адрес т.е. подпрограмата получава памет. Върху тази памет може да се наложи типов шаблон, който се дефинира в подпрограмата и паметта да бъде обработена в съответствие с изискванията на този тип. Когато се извиква подпрограма в която участват безтипови параметри, за тях е без значение какво ще се замести.

```
Function sum(var x; N : integer):integer;

Type

XType=Array[0,,1000] of Byte;

Var

I : integer;

Begin

Result:=0;

For I=0 to N do
```

Result=result++xtype(x)[I]

- Параметри отворени масиви. Той се дължи в съответствие с правилата:

1) счита се че е нулево базиран т.е. базира се от нула горна граница на индекса.

2) Такъв параметър може да бъде обработен само по елементи;

3) За такъв формален параметър може да се замести аргумент, който е проста скаларна променлива от типа на компонентите.

Подразбрани стойности на параметрите

Формални параметри, които представляват стойности от прост тип могат да бъдат инициализирани като след параметъра се запише = стойност.

=

Допустима е инициализацията само за параметри стойности и константи. Инициализиращата стойност влиза в действие ако при текущото активиране или извикване на подпрограмата за този случай не се подаде аргумент. В противен случай се работи с подадения аргумент.

Правила за изпълнение на подразбрани параметри

- ако за някакъв форм. параметър е зададена подразбрана стойност всички параметри след него трябва да има подразбрана стойност;

- ако по някаква причина се прави изпреварващо описание на заглавието на ф-ята, подразбраните стойности се задават в заглавието. В по следващата реализация на ф-ята подразбрани параметри не се записват.

- Използването на подразбрани стойности трябва да се съобрази с т.нар. претоварване (предефиниране) на ф-ии (няколко реализации на ф-я с еднои също име). Не трябва да възниква противоречие м/у шаблоните за извикване породени от предефинирането на стойности подразбрани по параметри.

- При извикване на ф-я с подразбрани параметри, могат да се изпускат аргументи от дясно на ляво.