روالها و تابع ها:

در مورد زیر برنامه ها و نقش آنها در تسهیل برنامه نویسی و همچنین فواید این ایده در نحـوه نوشـتن برنامه صحبت شد. اینک به پیاده سازی این ایده ها در محیط *Delphi* می پردازیم.

قبل پرداخت به ساختار تعریف زیر برنامه ها که در اکثر زبانهای برنامه نویـسی بـه دو دسـته توابـع و روالها تقسیم می شوند – چند نکته تکنیکی را با هم بررسی کنیم.

در تعبیر و تبیین راهبرد پیمانه ای گفتیم که اصولاً در چه مواردی زیر برنامه ها مفیدند، یکی در هنگام انجام یکسری کارهای تکراری بود و دیگری کارهایی که قالب یکسان دارند ولی بـر روی داده هـای متفـاوتی عمل می کنند، از دسته اول می توان به یک برنامه اشاره کرد که در جایگاه های متفـاوت بایـد یـک وسـیله الکترونیکی را خاموش کند، حال اگر تعداد زیادی دستور جهت خاموش کردن دستگاه لازم باشد. این کـار را یک بار به عنوان زیر برنامه می نویسیم و سپس در جایگاههای مورد نیاز آنرا اجرا می کنـیم. مثـال از دسـته دوم همان تابع محاسبه عددی انتگرال معین است که با ورودیهای متفاوت حدود بالا و پایین و تابع مورد نظـر نتیجه را محاسبه کرده و بر می گرداند. در مکالمات روزمره هم از کارهای روتین زیاد نام می بریم کـه همـین زیر برنامه ها هستند و روتین نام دیگر همین زیر برنامه است.

بنابراین یک مساله جابجا شدن سرنخ اجرای برنامه به یک زیربرنامه و سپس بازگشت از آن و مـساله دیگر انتقال داده ها از برنامه به زیر برنامه و بالعکس است.

در فصل اول اشاره کردیم که هم برنامه و هم داده ها در حافظه ذخیره می شوند و از طرفی پـر واضـح است که پارامترها و سرنخ فراخوانی زیر برنامه ها هم در حافظه ذخیره می شوند به این علـت نیـاز بـه یـک

شبکه رشد – شبکه ملی مدارس ایران
Olympiad.roshd.ir

راهکار برای ساماندهی به این موارد ذخیره شده در حافظه است. از تاریخچه تحول راهکارها کـه بگـذریم در حال حاضر این ساماندهی به عهده پردازشگر مرکزی است، نتیجه انجام راهکار این است که سیـستم عامـل بخشی از حافظه را به نحوی به پارامترها و سرنخ فراخوانیها اختصاص می دهد، به این بخش پشته می گویند، این بخش حافظه خود یک ساختار دارد که در فصل پنجم به آن خواهیم پرداخت این نشان می دهد که اگر به دفعات بسیار زیاد زیر برنامه تو و تو حساب کنید سرنخ از دست *CPU* **در می رود!**

بـدین معنـا کـه حافظـه پـشته کفـاف ذخیـره اطلاعـات لازم را نمـی دهـد بـه ایـن حـال *Stack Overflow* **گویند، هر چند این خطا در زمان برنامه نویسی در سیستم عامل** *Dos* **اهمیت داشت و بـا استراتژی های جدید کار با حافظه بسیار بعید است که بتواند برنامه ای را بصورت مذکور دچار خطا کنید!**

برای نوشتن یک زیر برنامه یکی از دو ساختار زیر استفاده می شود اولی برای رولها و دومی برای توابع:

ساختار 1:

```
procedure < Name > [( param1 : var_type ; ( param 2 : var_type...) ] ;
begin
  // Pr ocucedure   body
end ;
```

ساختار 2:

```
function < Name > [( param1 : var_type ; ( param 2 : var_type...) ] ; return _type ;
begin
  // function   body
end ;
```

شبکه رشد – شبکه ملی مدارس ایران
Olympiad.roshd.ir

$< Name >$ها اسامی اختصاص یافته به هر کدام از زیر برنامه ها، $Param_i$ نام پارامترهای انتقال داده شده به زیر برنامه $\text{var} - type$ ها انواع داده هستند همان انواع متغیرها مثل $Interger$,... پارامترها در برنامه عیناً مانند متغیر قابل دسترسی اند. در نهایت $return - type$ هم عیناً نوع متغیر است و نوع متغیری است که تابع به برنامه اصلی باز می گرداند. بخش $body$ بدنه زیر برنامه است و کل متن اصلی زیر برنامه در این بخش نوشته می شود.

در مورد انتقال پارامترها به زیربرنامه نکاتی وجود دارد، هر چند که گفتیم پارامترها عیناً مشابه متغیرها هستند ولی در عمل سه نوع متفاوت از پارامتر وجود دارد که در هر مورد نحوه برخورد متفاوت است، این موارد بدون آوردن مثال قابل توضیح نیست لذا باید تا کامل شدن توضیح بدویات تعریف زیر برنامه صبر کنید.

در توابع برای آنکه مقداری که می خواهید به برنامه بازگردانید مشخص کنید دو راه وجود دارد یکی استفاده از متغیری هم نام با اسم تابع البته نه به این معنا که آن را تعریف کنید بلکه فقط استفاده کنید مثلاً

```
function  f (x : real) : real;
begin
M
f  := x * x_i ;
end;
```

ولی در $Delphi$ علاوه بر امکان فوق که مشترک با $Pascal$ است، یک متغیر تعریف شده دیگر وجود دارد که استفاده از آن ترجیح دارد. در تمامی توابع تعریف شده متغیر $result$ با نوع متغیر مشابه تابع وجود دارد که می توان استفاده کرد:

شبکه رشد – شبکه ملی مدارس ایران
Olympiad.roshd.ir

```
function  f (x : real) : real;
begin
M
  result := x * x;
end;
```

مزیت روش دوم این است که می توان از متغیر *result* در هر دو سوی عبارت انتساب استفاده کرد

مثلاً

R

```
result := result * 2;
```

ولی از *f* نمی توان چون در سمت راست = معادل فراخوانی خود تابع است:

Q

$$f_1 = f * 2$$

اما نکته دیگر اینکه همانند اینکه برنامه را بر چند زیر برنامه تقسیم کردید می توانید زیر برنامه ها را به چند تکه کوچکتر تقسیم کنید که البته برای این سطح زیر برنامه اسمی نمی گذاریم فقط توضیح زیر را که در مورد کار با توابع و روالها ضروری است در نظر بگیرید. در فاصله بین تعریف صورت تابع و *begin* می توان عیناً مانند فاصله بین *begin, program* زیر برنامه تعریف کرد *type* تعریف کرد و متغیر تعریف کرد. متغیرها و *type* ها و زیر برنامه های تعریف شده در این بخش فقط در *body* ی زیر برنامه قابل استفاده اند. به این محدودیت *Scope* گفته می شود. عبارت قبلی را به این نحو می توان بیان کرد که *Scope* متغیرها و روالها و ... تعریف شده در فاصله بین نام زیر برنامه و *begin* فقط خود زیر برنامه است. لذا عبارتی مانند زیر مجازند.

شبکه رشد – شبکه ملی مدارس ایران
Olympiad.roshd.ir

```
procedure  average(a,b : iterger) : real;
var
  c : real;
begin
  c := (a + b) / 2;
  result := c;
end;
```

حال به این بپردازیم که در برنامه چگونه می توان از این زیر برنامه های نوشته شده استفاده کرد.

یـک روش عبـارت مقابـل اسـت:

$$< subroutne - name > \left[(p^7 \left[, p^2 ... \right]) \right];$$

که برای هر دوی تابع و رول مشترک است مثلاً:

```
var
  k ,e : interger;
begin
M
  average(k ,e)
M
end;
```

و حالت دوم $< var \quad name >:=< Subroutine - name > \left[(p1[, p2...]) \right];$

که منظور از عبارت بالا فقط انتساب نیست بلکه در هر جایگاهی که مشابه نـسبت دهـی اسـت قابـل

استفاده است مثلاً:

شبکه رشد – شبکه ملی مدارس ایران

```
var
    k ,e : Integer ;
  d : real ;
begin
   M
    d := average(k ,e);
end ;
```

و یا

```
  M
if   average( k ,e ) > 10  then
begin
M
end ;
M
```

که جایگاه استفاده از توابع را بخوبی نشان می دهد.

اگر دقت کرده باشید سؤالی پیش می آید. وقتی که روال ها یا تابع ها با پارامترها فراخوانی مـی شـود تاثیر روالها و توابع بر متغیرها ی استفاده شده به عنوان پارامتر چیست؟این سؤال وقتی مطرح است که عـدد ثابت به عنوان پارامتر داده نشده باشد.

برای روشنتر شدن بحث روال و فراخوانی نمونه زیر را در نظر بگیرید.

```
procedure   mul (a,b ,c : real );
begin
  c := a* b ;
end ;
var
  k ,l ,m : real
begin
  mul (k ,l ,m );
end.
```

در نحوه تعریف پارامتر دقت کنیـد، ایـن نـوع تعریـف پـارامتر هـم مجـاز اسـت اشـتباهی نـشده!

شبکه رشد – شبکه ملی مدارس ایران
Olympiad.roshd.ir

$mul(k,l,m);$

فراخوانی mul چه تاثیری بر m,l,k دارد؟

حال مساله همان سه نوع پارامتر پیش می آید دریک حالت که حالت فوق است برنامه اجرا مـی شـود ولی mul هیچ تاثیری بر k,m,l ندارد. در این حالت یک کپی از m,l,k ایجاد می شود و زیر برنامه بـا ایـن کپی ها کار می کند و لذا هیچ تاثیری بر برنامه اصلی ندارد.

حالـــــــــــــــــــــــــــــت دوم $const$ اســـــــــــــــــــــــــــــت.

$procedure\ \ mul\left(const\ \ a,b,c : real\right);$

⋮

در این حالت قبل اجرای برنامه پیغام خطا دریافت خواهید کرد $(compile\ error)$ لـذا در ایـن حالـت پارامترها به عنوان ثوابت درون برنامه هستند و از دست نخوردن آنها در طول برنامه مطمئن هستیم.

حالـــــــــــــــــــــــــــت ســـــــــــــــــــــــوم حالـــــــــــــــــــــت var :

$procedure\ \ mul(\text{var}\ \ a,b,c : real);$

در این حالت خود پارامترها به برنامه منتقل مـی شـوند و هـر تغییـری در پارامترهـا عینـاً تغییـر در متغیرهایی از برنامه است که به عنوان پارامتر منتقل شده اند. در این نوع تعریف پارامترها نمی توانیـد عـدد ثابت به عنوان پارامتر به برنامه انتقال دهید یعنی مثلاً $mul(k,l,2)$ قابل قبول نیست.

این بحث مفصل را در همین جا به پایان می بریم ولی جهت بهتر تفهیم شدن بحث بـه مثالهـای آتـی توجه کنید.

شبکه ملی مدارس ایران – شبکه رشد
Olympiad.roshd.ir

مثالهای زیر را بررسی و تحلیل کنید:

برای اجرای آنها دقت کنید که به روش زیر عمل کنید که ابتدا پروژه حاضر در دلفی را ببندید و سپس یک پروژه جدید از نوع *Console Application* **ایجاد کنید و کد نوشته شده را وارد کنید برای آشنایی با نحوه انجام این کارها فصل هفتم را مطالعه کنید.**

در نهایت برای فهم برنامه این را در نظر بگیرید که دستور *Write ln* **هر چه به آن داده شود روی صفحه چاپ می کند و دستور** *read ln;* **منتظر ورود یک کد** *Enter* **(فشردن کلید** *Enter* **) می شود.**

```
program Sample 1;
function max( a,b : Interger ) : int erger ;
begin
  if  a > b  then
    result := a
else
    result := b ;
end ;
begin
Write ln(max( 10,8));
Write ln(max( 5,7));
read ln;
end .
```

شبکه رشد – شبکه ملی مدارس ایران
Olympiad.roshd.ir

```
Program Sample 2;
procedure DivMod(var a,b : Interger );
var
  c : Interger ;
begin
  c : a;
  a := c div b;
  b := c mod b;
end ;
var
  x1,x2 : integer ;
begin
x1 := 124;
x2 := 3;
Div Mod( x1,x2 );
Writeln( x1,x2 );
readln ;
end .
```

شبکه رشد – شبکه ملی مدارس ایران
Olympiad.roshd.ir