## بهینه سازی :

مباحث بهینه سازی[1] مباحث پیچیده ای بوده و ختم کلام در این مباحث اصولاً ممکن نیست. در این تعادل تنها چند نکته که از لحاظ دستور زبانی به کمک بهبود سرعت اجرای برنامه می آید بیان می کنیم. مابقی بهینه سازی شامل تحلیلهای مرتبه زمان اجرا $(Order)$ به بخش طراحی الگوریتم واگذار می شود.

اولین اصل حذر از هر گونه عبارت شرطی است. به این معنا که حتی المقدور از شرط ها استفاده نکنید مثلاً در مورد محاسبه ماکزیمم و مینیمم دو عدد. فرض کنید که تابع قدر مطلق به روشی پیاده سازی شده:

$$\max(a,b)=\frac{|a+b|+|a-b|}{2} \qquad \min(a,b)=\frac{|a+b|-|a-b|}{2}$$

برای اینکه ایده ای از پیاده سازی قدر مطلق بدون استفاده از عبارات شرطی داشته باشید، روش زیر را در مورد اعداد صحیح و نمایش " مکمل دو" برای اعداد منفی در نظر بگیرید.

```
sb := n and $80000000
sb := sb shr 31;
hn := sb *$FFFFFFFF;
n := (n -sb)×or hn;
```

فرض کنید که $n,hn,sb$ همه اعداد **32** بیتی صحیح هستند. کل عملیات بالا باعث می شود محتویات $n$ در انتهای کار قدر مطلق $n$ در ابتدای کار باشند.

یک نکته در این گونه صرفه جوئیها در پیچیدگی نتیجه شده از این روش است نکته دوم این است که ممکن است چیزی که جای شرط جایگزین می کنید آنقدر طولانی شود که خود شرط سریعتر اجرا شود.

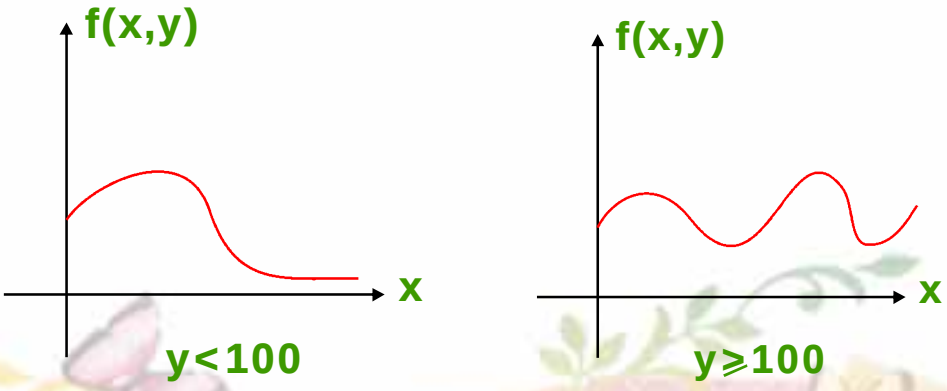
به هر حال در مورد بالا می توان بدون بکار بردن شرط $\min,\max$ را محاسبه کرد.

نکته دوم عبارات منطقی است. نتیجه عبارت منطقی $a \quad and \quad b$ اگر $a$ ناصحیح باشد کلا ناصحیح است مستقل از اینکه $b$ چه باشد. $Delphi$ در محاسبه مقادیر منطقی از این نکته استفاده کرده و اگر اولین عنصر در $and$ صفر و یا اولین عنصر $or$ یک باشد باقی عبارت بررسی نمی شود یعنی در محاسبه

$m : a \quad and (b \quad xor (C \quad and \quad not \quad d ));$

اگر $a$ غلط باشد دیگر $Delphi$ بقیه عبارت را محاسبه نمی کند. بنابراین در هنگام نوشتن اینگونه شرط ها دقت کنید ساده ترین بخش محاسبه در ابتدای عمل $and$ قرار گیرد. یک مثال خوب عبارت گفته شده است محاسبه درستی یا غلطی بسیار ساده تر از عبارت سمت راست است لذا $a$ در سمت چپ نوشته شده.

در نهایت استفاده از متغیرهای تابعی در جای خود بسیار مفید است. مثلاً در نظر بگیرید که یک تابع دو متغیره دارید:




که بصورت بالا تعریف می شود. حال اگر تابع فوق در زیر برنامه ای بصورت زیر بکار رفته باشد.

```
procedure Dummy(y):Real);
var X ,W :Real;
begine
  X :=O;W :=O;
  While(X <250) do
  begin
    W :=f (X ,Y )+X ;
    X :=X +0.5;
 end;
end;
```

**می توان زیر برنامه بالا را بصورت زیر نوشت**

**( البته با فرض آنکه** $f(x, y \geq 100) = f_2(x), f(x, y < 100) = f_1(x)$ **باشند. )**

```
procedure   Dummy(y):Real);
 var X ,W :Real;
 f :function(P1,P2 :Real):Real;
begin
 if   y ≥100  then  f :f1;
 else  f :=f2;
 X :=O;W :=O;
 While(X <250)do
 begin
  W :=f (X );
  X :=X +0.5;
 end;
end;
```

**در حالت اول به ازای هر بار اجرا شدن** $f(x, y)$ **یکبار شرط** $y \geq 100$ **اجرا می شد یعنی 250 بار اجرا شدن یک شرط در حالیکه در حالت دوم یکبار شرط** $y \geq 100$ **اجرا شده و از نتیجه در کل ادامه برنامه استفاده می شود. این کار با کم کردن تعداد اجراهای شرطی بصورت سرعت برنامه را بالا می برد.**

*Optimization* [1]