اولین تابع که مشخص هست :

```
function Random_Serial(PLen: Integer): string;
var char: string;
begin
  Randomize;
  char    := '1234657890';
  Result  := '';
  repeat
    Result := Result + Char[Random(Length(Char)) + 1];
  until (Length(Result) = PLen)
end;
```

از اسمش پیداست با توجه به وردیPLen یک رقم تصادفی ایجاد می کنه .
یعنی اگر اینجوری باشه :
Random_Serial(7);
خروجی 7 رقم به صورت تصادفی میده !
میرویم سراغ تابع اصلی .
تا اینجا همان طور که می دانید شکل سریال به صورت زیر است :
(1234-5678-9000-1000)
حالا ما نیاز داریم 4 تا عدد تصادفی ایجاد کنیم .

```
r1 := Random_Serial(4);
//r2 := Random_Serial(4);//we dont need it !
r3 := Random_Serial(4);
r4 := Random_Serial(4);
```

اما نکته اینجاست که r2 رو حذف کردیم.
چرا؟ چون اگه یادتون باشه کد قسمت دوم از سریال نامبر
فقط جنبه برابری داشت پس اینجا نیازی بهش نداریم چون در جلوتر خودمان آن را تولید می کنیم .

N1 هم Hash شده کلمه (Easy Gang) هست که برای کد ازآن استفاده شده .

```
n1:='ec7235e71546232da9f7cc02f32c9d0e';
```

اینجا اولین عدد سریال رو برمی داریم می ریزیم تو r_g خود سریال هم (r1) تصادفی بود .

```
r_g := copy(r1,0,1);//get the firs char of random serial1 (r1)
```

اینجا به اندازه ی همان عدد قبلی (r_g) تا کل کلمه Hash شده (n1) می شمارد جلو می رود و همه را در n2 ذخیره می کند.

```
for i := strtoint(r_g)+1 to length(n1) do n2 := n2 + (n1[i]);
```

اینجا 10رقم از خودش (n2) جدا می کند و در n2 (خودش) ذخیره می کند :

```
n2 := copy(n2,0,10);
```

```
1234-5678-9000-1000
MD5 : ec7235e71546232da9f7cc02f32c9d0e
```

اگه یادتون باشه بعد از این کار ها با یک عدد دیگر جمع می شد .
یعنی عدد سوم از قسمت اول. اینجا آن عدد سوم رو می گیرد ذخیره می کند در r_g2 :

```
r_g2 := copy(r1,3,1);
```

حالا همه را به هم بست میدهد . یادتون هست که گفتم :
"قسمت های سوم و چهارم سریال نامبر با کد 10 رقمی جداشده بالا مخلوط می شود
عدد مخلوط شده با یک عدد دیگر جمع می شود (3 c7235e7154 90001000)"

```
1234-5678-9000-1000
MD5 : ec7235e71546232da9f7cc02f32c9d0e
c7235e7154 + 3
```

اینجا همان اتفاق می افتد کد ها مخلوط شده و با عدد سوم از قسمت اول جمع شده در R_Total ذخیره می شود :

```
R_Total := (r3+r4) + (n2) + (r_g2);
```

مرحله بعدی این هست که باید کد به دست آمده تبدیل به (MD5) بشود که در این خط این اتفاق می افتد :

```
MD5_H := StrMD5(R_Total);
```

حالا نوبت می رسد به آن 5 رقم،
خاطرتون هست که گفتم جدا شدن این 5 رقم بستگی دارد به رقم دوم از قسمت اول یعنی :

```
1234-5678-9000-1000
MD5 : a5c514ce566dcfc8856f757c45998c28
```

خط زیر آن عدد رو ذخیره می کند در r_g3 :

```
r_g3 := copy(r1,2,1);
```

این خط هم مثل بالا کارش این است که به اندازه عدد دوم از قسمت اول تا کل کد Hash شده را
می شمارد و در متغیر n3 ذخیره می کند :

```
for j := strtoint(r_g3)+1 to length(MD5_H) do n3 := n3 + (MD5_H[j]);
```

حالا چون ما فقط 5 رقم رو میخواستیم در این خط 5 رقم را از کل n3 جدا می کنیم :

```
n3 := copy(n3,0,5);
```

حالا اینجا دقت کنید ! کد ها باید 4 رقم 4 رقم باشد .
در Part1 ما 3 رقم r1 رو می گیرم با 1 دانه اول n3 جمع می کنیم !

می شود قسمت اول سریال نامبر که 4 حرف دارد .

```
Part1 := copy(r1,0,3) + copy(n3,0,1);
```

Part2 هم از n3 دو تا می شمارد و 4 تا جدا می کند :

```
Part2 := copy(n3,2,4);
```

این هم شد قسمت دوم از سریال نامبر !

حالا همه را با هم جمع می کنیم و بین آنها "-" میزاریم :

```
Final_Serial := Part1 + '-' + Part2 + '-' + r3 + '-' + r4;
```

دست آخر هم خروجی رو به تابع می دهیم :

```
Result := Final_Serial;
```

procedure Copy(s : ShortString; index, count : Integer) : ShortString;

موفق باشید
مهدی هزاوه ای
(c) 2007

Feedback : impostor_76171@yahoo.com - www.impostor.blogfa.com