제 5 강. Dynamic Array(상자에 찍찍이를 붙여보자)

Dynamic Array

우리는 일상 생활 속에서 Dynamic이라는 말을 많이 쓴다. 뭔가 움직임을 나타내는 말인데, 우리가 앞 시간에 배웠던 상자가 움직인다니 무슨 말을 하는지 영 감이 잡히지 않을 수 도 있겠다. 배열이 Dynamic 하다라는 말은 이제까지의 상자와는 다르게 상자에 '찍찍이'가 붙어있다는 말이다. 왜 옷이나 신발에 쓰이는 붙여다가 떼어낼 수 있는 찍찍이 말이다. 상자에 찍찍이가 붙어있어서 우리가 원하면 더 몇 개를 더 추가로 붙일 수 있고, 필요 없을 때에는 다시금 때어버릴 수 도 있는 편리한 상자라는 말이다. 조금 더 특별한 상자이기에 앞 강의에서 배운 Array의 선언 방식과 살짝 다르다.

Dynamic Array Syntax

Dim [Array Name]() as [Data Type]
Redim [Array Name][Array Size]

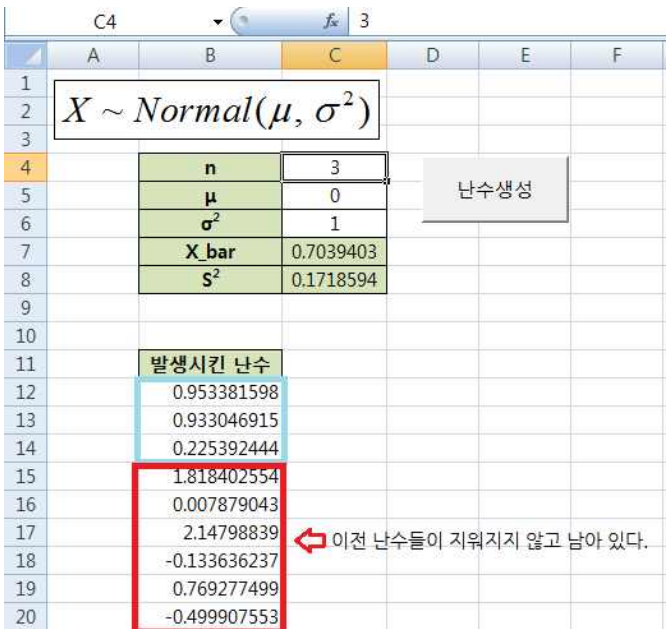
첫 번째 줄은 앞에서 배운 Array선언과 비슷하다. 단, 하나 차이가 나는 것은 바로 처음 상자를 만들 때에는 상자 크기를 말해주지 않는다. 그냥 Visual Basic에게 '찍찍이 붙은 상자 만들꺼야.' 하고 알려주는 거다. 비로소 두 번째 줄에 가서야 '이제 몇 개 붙여서 상자 만들어줘~!" 하고 말하는 셈이다. 앞 강의에서 배운 난수 생성 프로그램의 봉인을 해제 시켜 보자!

[Example 5.1]

```
Sub Random_Sample()

    Dim n As Long, mu As Double, i As Long
    Dim sigma As Double, X_Sum As Double

    Dim X_Sample() As Double      ' <- 찍찍이가 붙은 상자를 만들 준비!

    n = Sheets(1).Range("C4")(1, 1)
    mu = Sheets(1).Range("C4")(2, 1)
    sigma = Sqr(Sheets(1).Range("C4")(3, 1))

    ReDim X_Sample(1 To n)        ' <- 사용자가 원하는 만큼 상자 만들기!

    For i = 1 To n
        X_Sample(i) = mu + sigma * WorksheetFunction.NormSInv(Rnd())
        X_Sum = X_Sum + X_Sample(i)
```

```
    Next i

    Dim X_bar As Double
    X_bar = X_Sum / n

    Dim S_Variance As Double

    For i = 1 To n
        S_Variance = S_Variance + (X_Sample(i) - X_bar) ^ 2
    Next i

    S_Variance = S_Variance / (n - 1)

    Sheets(1).Range("C7")(1, 1) = X_bar
    Sheets(1).Range("C7")(2, 1) = S_Variance

    For i = 1 To n
        Sheets(1).Range("B12")(i, 1) = X_Sample(i)
    Next i

End Sub
```

짜잔~! 위의 코드를 사용하면 사용자가 원하는 만큼의 상자를 그때그때 만들어 줄 수 있기 때문에 4강에서 배웠던 난수 생성 프로그램의 발생 개수의 봉인을 풀 수가 있는 것이다. 하지만 아직 할 일이 남아 있다. 우리가 만든 엑셀 프로그램에는 중대한 버그가 있기 때문이다.



바로 새로 발생시키는 난수의 개수가 이전에 뽑은 난수 개수보다 작아지면, 이전 난수들이 지워지지 않고 시트에 그대로 남아있게 되는 것이다. 그림에서 빨간 색깔 부분의 난수를 이전 시행에서 뽑힌 난수이고, 위에서 3개 칸의 난수만 바뀌게 된다. 따라서 매번 난수를 뽑을 때 마다 "B12" 셀 아래를 한번 싹~ 지워주는 작업을 추가 해야 한다. 이 작업은 꽤나 많이 쓰이는 작업이므로 잘 알아두었다가 계속해서 써먹도록 하자. Sub문의 바로 아래에 다음과 같은 청소작업을 나타내는 코드를 추가 하자.

```
Dim output As Range
Set output = Sheets(1).Range("B11").CurrentRegion
output.Offset(1, 0).Resize(output.Rows.Count - 1, output.Columns.Count).Clear
```

일단 이 코드는 output이라는 범위를 담을 수 있는 변수를 선언 한 후, 특정 범위를 다시 재조정하여 내용을 지우는 코드라는 것만 알고 넘어가도록 하자!

Option base

이제 배열에 대해서 더 자세하게 배워보는 시간을 갖자. 우선 Option Base라는 것은 배열의 시작점을 설정하는 구문이다. 다음 두 개의 코드를 비교해보자.

[Example 5.2]

<Case 1>

```
Option Base 0
Sub Option_Base_test()

    Dim A(5) As Double

End Sub
```

<Case 2>

```
Option Base 1
Sub Option_Base_test()

    Dim A(5) As Double

End Sub
```

먼저 앞에서 배운 배열의 선언 방식인 Array Name(Start To End)방식을 생략하여 예제와 같이 나타낼 수 있다는 사실을 알 수 있고, Case 1의 경우 배열의 크기는 A(0 To 5)인 6칸짜리 상자가 만들어지고, Case 2의 경우 배열의 크기는 A(1 To 5)인 5칸짜리 상자가 만들어지게 된다.

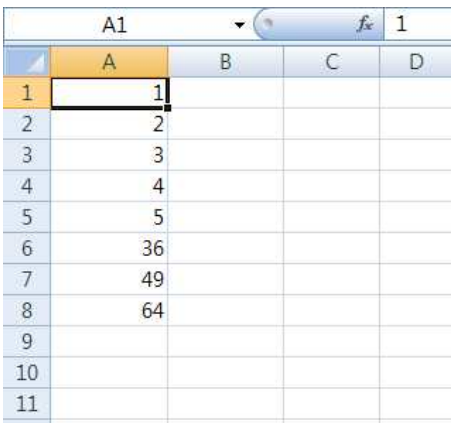
Preserve(찍찍이 사용법)

앞에서 Dynamic Array를 설명할 때, 찍찍이가 붙어있는 상자라고만 했지 상자를 떼고 붙이는 방법을 설명하지는 않았다. 아래의 예제를 통하여 5개가 있는 상자에 3칸을 더 붙여서 숫자를 입력해보자.

[Example 5.3]

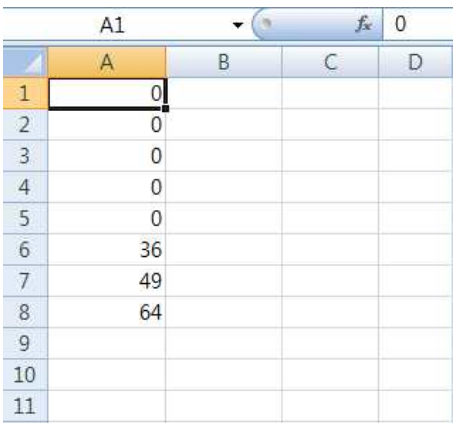
```
Option Base 1
Sub DynArray()

    Dim i As Integer
    Dim myArray() As Integer      '<- 찍찍이 상자 준비
    ReDim myArray(5)              ' <- 상자 5개를 이어 붙이기

    For i = 1 To 5
        myArray(i) = i
    Next i

    ReDim Preserve myArray(8)     '<- 5개 있던 상자에 3개 더 추가! & 기존 데이터 보존!

    For i = 6 To 8
        myArray(i) = i * i
    Next i

    For i = 1 To 8
        Sheets(1).Range("A1")(i, 1) = myArray(i)
    Next i
End Sub
```

[Example 5.3]에 쓰인 Preserve 구문을 통하여 기존의 상자에 3칸의 빈 상자를 추가 할 수 있게 되었다. 상자 추가만 했는데 웬 "보존?" 이라고 생각하는 독자들에게 박수를 보낸다! 사실 상자만 3칸 더 붙이기 위해서는 Redim myArray(8) 이라고 써도 8칸짜리 상자가 만들어진다. 하지만 여기서 중요한 것은 기존 5개짜리에 있던 데이터는 다 날아가고, 8칸짜리의 빈 상자가 생성되게 된다. 위 예제에서 Preserve를 뺀 코드를 통하여 결과를 확인해 보자.



	A
1	1
2	2
3	3
4	4
5	5
6	36
7	49
8	64
9	
10	
11	

<Preserve 사용시>



	A
1	0
2	0
3	0
4	0
5	0
6	36
7	49
8	64
9	
10	
11	

<Preserve 미사용시>

Lbound()와 Ubound() (상자의 처음과 끝)

이제까지 배운 Dynamic Array는 상자의 크기를 사용자의 마음대로 변하게 할 수 있다는 것이 장점이었다. 하지만 이러한 장점은 프로그래밍을 할 때에는 오히려 불편한 것이다. 처음과 끝이 마구마구 바뀌는 것이기 때문에 우리가 저번에 배운 Loop문을 사용하기가 어렵다는 것이다. 만약 사이즈가 얼마인지 모르는 배열에 처음부터 끝까지 내용을 채우고 싶을 때에는 어떻게 해야 할까? 다음의 코드를 살펴보자.

[Example 5.4]

```
Option Base 1
Sub Bound_Function()

    Dim i As Integer, n As Integer
    Dim myArray() As Integer

    n = Sheets(1).Range("A1")(1, 1)

    ReDim myArray(n)

    For i = LBound(myArray) To UBound(myArray)
        myArray(i) = i
    Next i

End Sub
```

[Example 5.4]의 코드는 사용자가 어떤 수를 입력하든지 상자의 처음부터 끝까지 숫자를 입력할 수 있다.