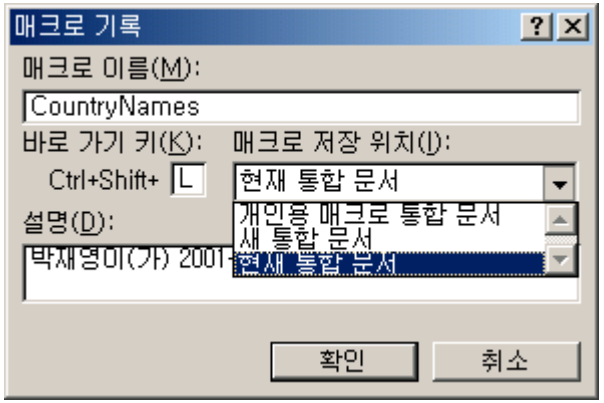
엑셀VBA

백영일

목 차

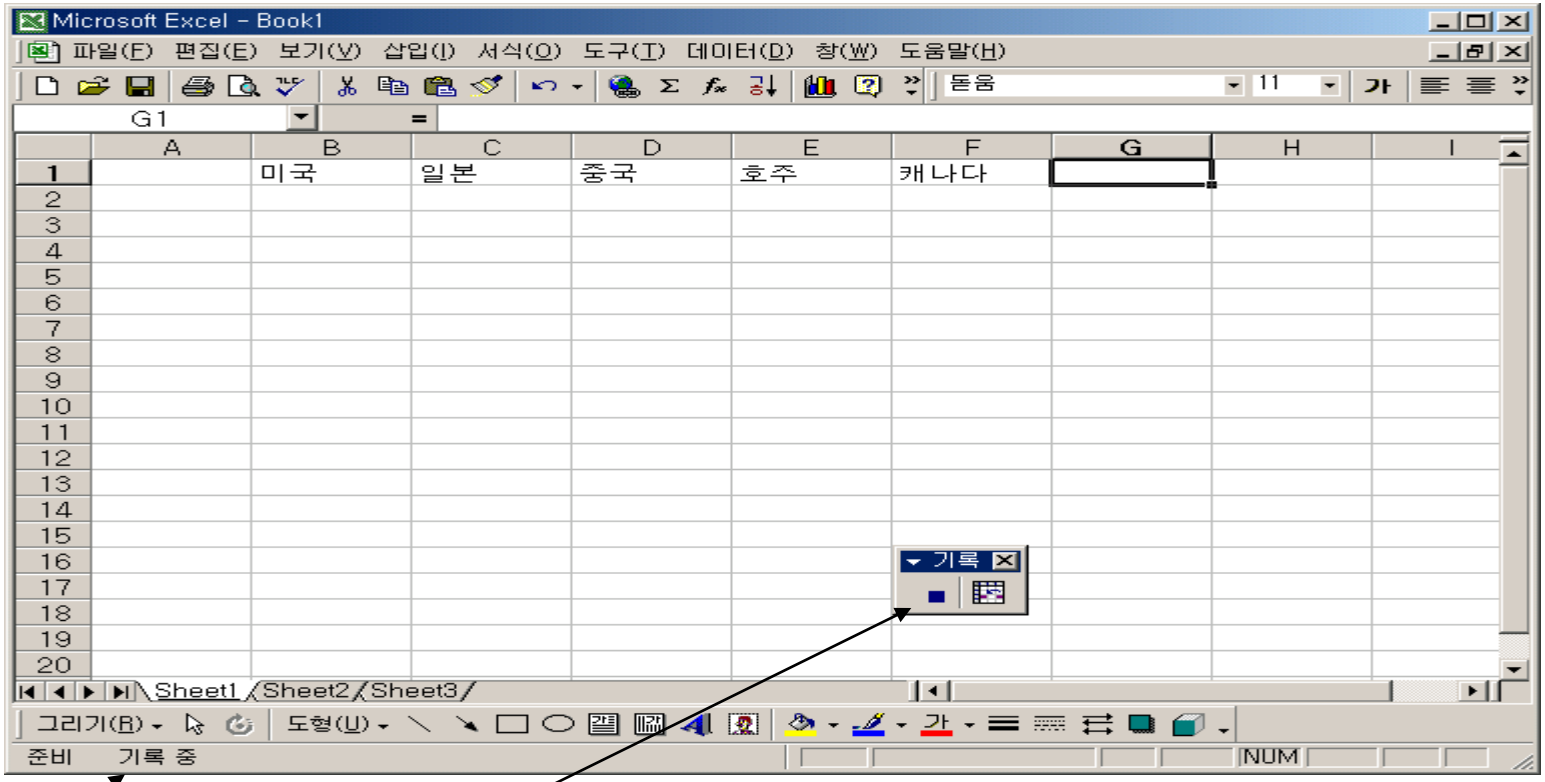
매크로의 기록

도구 > 매크로 > 새 매크로 기록을 누르면 아래 대화상자가 나온다.



매크로 이름에 CountryNames 를 입력해 보자
매크로 이름은 반드시 문자로 시작해야 하고, 문자/숫자/밑줄(_)의 조합만 가능하며,
전체 문자열 수는 255개 이상 입력이 불가능하다. (단어와 단어 사이에 blank 같은 것이 들어가면 안됨~)
매크로 이름은 나중에 누가 보더라도 어느 매크로인가 하는 것을 알 수 있도록 구체적인 이름으로
입력하는 것이 좋고, 단어와 단어는 밑줄로 하든지 아니면 대소문자로 구별해 주는 것이 좋다.
바로가기 키는 나중에(매크로 기록 후에)매크로를 실행할 수 있는 단축키를 의미한다.
지정해도 되고, 안해도 되는데, 여기서는 대문자 L을 입력했다.
소문자로 입력할 경우 Ctrl+"알파벳"의 형태가 되고, 대문자로 입력하면 Ctrl+Shift+"알파벳"의
형태가 된다.
매크로 저장 위치는 개인용 통합문서로 하면 어느 파일에서나 엑셀 파일이면 이 매크로를 실행
할 수 있고, 새 통합문서로 하면 새 문서가 생기면서 여기에 매크로가 기록된다. 여기서는 일반적으로
많이 사용하게 되는 현재 통합 문서로 지정해 주도록 합니다.
확인 버튼을 누르고 아래와 같이 매크로를 기록해 보자.

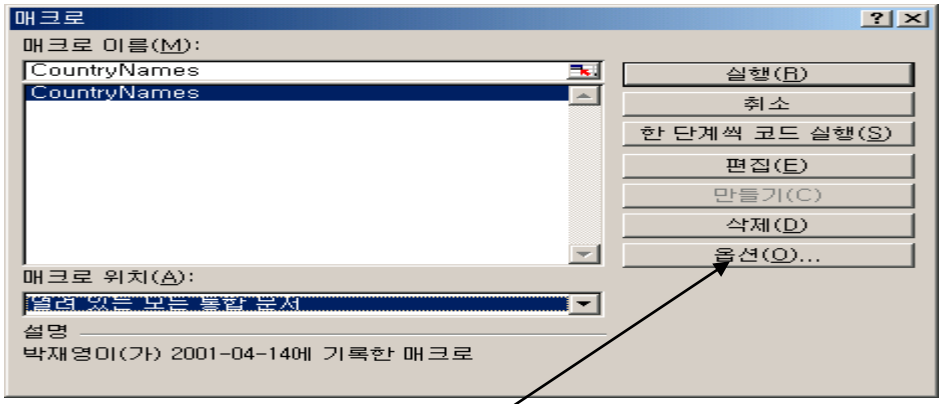
매크로의 기록



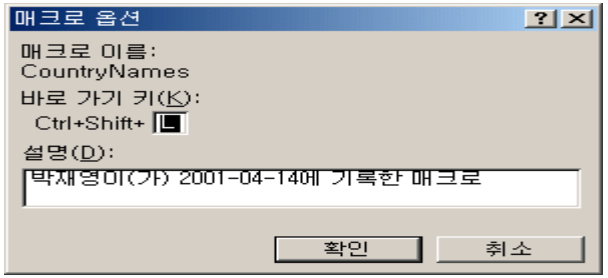
요기에 기록중이라는 메시지가 뜨면서 기록되기 시작한다.
현재 A1셀에 커서가 있는데, B1으로 옮기고 미국 F1 캐나다 까지 차례대로 입력한다.
다 입력한 후 **기록중지 버튼**을 누르면 매크로기록은 일단 끝난다.

매크로의 실행

Macro 기록이 끝나면 이렇게 한 것은 현재 열려있는 모든 sheet 상에서 매크로를 실행시킬 수 있다
매크로를 실행할 수 있는 방법은 여러가지가 있다.
도구 > 매크로 > 매크로 버튼을 누르면 아래와 같은 대화상자가 나온다.



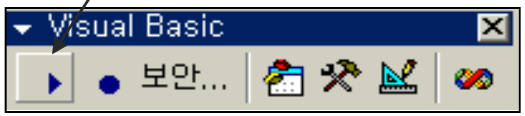
여기서 방금 기록한 CountryNames 라는 이름의 매크로를 실행시키면 된다.
실행시키면 현재 B2부터 F1까지 국가명이 순식간에 입력될 것이다.
이 매크로는 열려있는 통합문서의 어느 시트에 가서 실행해도 된다.
위 대화상자에서 **옵션버튼**을 누르면 아래와 같은 대화상자가 나온다.



앞에서 지정한 대로 바로가기 키가 나오는데,
바로가기 키가 Ctrl+Shift+L 인 이유는 앞에 매크로 기록시에 "대문자 L"로 지정해 주었기 때문이다.
매크로 실행 대화상자를 닫고, 현재 열려있는 시트에 가서 "Ctrl+Shift+L"키를 동시에 누르더라도 위 매크로는 실행된다.

매크로의 실행

매크로를 실행하는 또다른 방법으로 단축아이콘을 사용할 수도 있다.
엑셀 메뉴바 (파일, 편집, 보기 --- 등등이 적혀있는 부분) 에다 커서를 갖다대고
오른쪽 버튼을 누른후 Visual Basic 에 관련된 아이콘들이 나오는데 여기서
바로 **요 버튼 (실행버튼)** 을 누르더라도 위 매크로 대화상자가 나온다.



상대참조와 절대참조

앞에서와 같이 매크로를 한번 기록만 해 놓으면 열려있는 어느 시트에서나 단축키 하나로 반복되는 작업을 간단히 할 수 있다.
그런데 앞의 방식대로 기록했을 때, 단점은 항상 그 시트의 B1부터 F1 셀에 다섯개의 국가명을 입력한다는 점이다. 커서를 어디에 두더라도 항상 B1부터 국가명이 입력된다.
현재 활성화된 셀 (ActiveCell)을 기준으로 국가명을 입력되게 할 수는 없을까?
실제로 작업하다 보면 후자의 방법, 즉 현재셀을 기준으로 어떤 반복된 작업을 할 필요성을 더 많이 느낄 것이다.
전자 (앞의 시트상에서 설명한 방법) 의 경우를 **절대참조 (Absolute Recording)** 라고 하고,
후자(현재 활성화된 셀을 기준으로 기록하는 방법)의 경우를 **상대참조 (Relative Recording)** 라고 한다.
상대참조로 기록하는 방법은 간단하다.
앞서 절대참조 기록하는 방법과 동일한데, 다만 요 단추가 활성화되도록 한 다음 기록하면 된다.

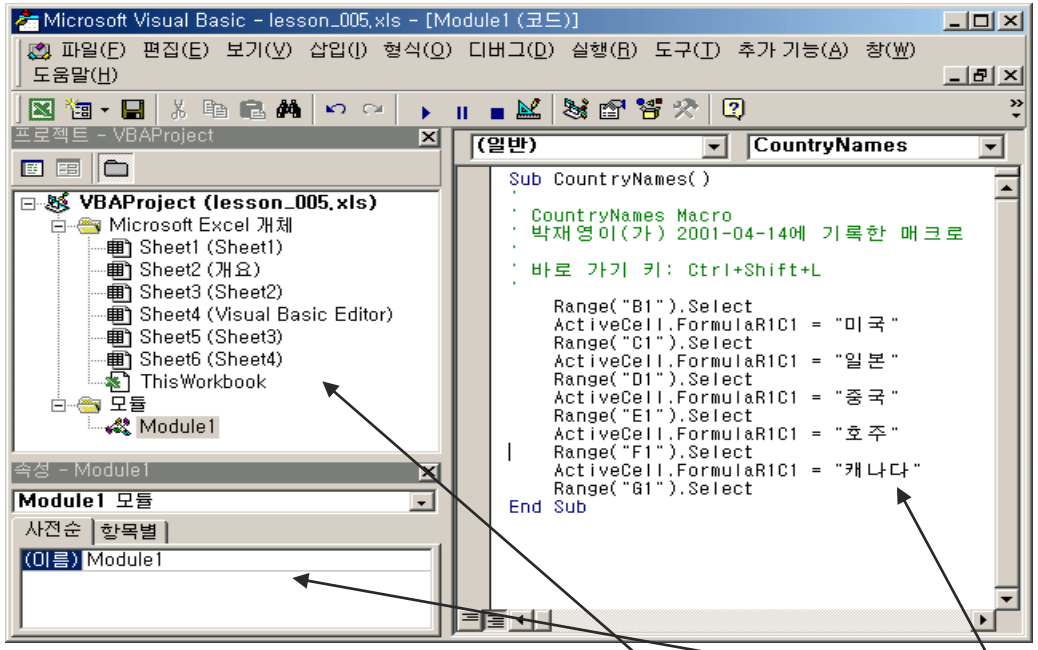
앞에서 절대참조 기록시에는 이 단추가 활성화 되어 있지 않은 상태였다.

(절대참조로 기록시)

(상대참조로 기록시)

Visual Basic Editor에 관하여

도구 > 매크로 > 매크로 실행으로 가서 CountryNames란 매크로를 선택한후 편집 버튼을 누르든가,
아니면 도구 > 매크로 > Visual Basic Editor 를 실행시키면 (단축키 : Alt + F11),
아래와 같은 이상한 화면이 나오는데, 이것을 Visual Basic 편집기라고 부른다.



우리가 CountryNames란 이름으로 기록한 매크로는 Module1이란 Module 내에 코드 창에 하나하나 기록되어 있다.
프로젝트 탐색기에서 Modlue1을 double click하면 Module1 내에 있는 코드를 볼 수 있다.
만일 Module1을 지우고 싶다면 Module1 선택후 마우스 오른쪽 버튼 클릭 > module1 삭제를 누르면 된다.
이 경우 CountryNames란 매크로도 삭제된다.
삭제버튼을 누르기 전에 module1을 내보내겠습니까? 란 메시지가 뜨는데 만일 내보낼 경우 *.bas 형태의 bas란 확장자를 가진 basic 파일을 만들어 내는데 이는 Visual Basic 코드 만을 저장하고 가져올 수 있음을 의미한다.
Module 을 하나 더 추가하고 싶으면 Module1 선택후 마우스 오른쪽버튼 클릭 > 삽입 > 모듈을 누르면 모듈이 하나씩 추가된다.
이렇게 해서 모듈 내에서 매크로를 직접 만들수 있다.
이런 매크로를 VBA(Visual Basic for Application)에서는 procedure 라고 한다.
Procedure는 앞서 기록한 바와 같은 형태의 Sub Procedure와 Function Procedure의 두가지 형태가 있다.
주로 사용되는 것은 Sub Procedure이며, Function Procedure는 Excel 에 내장되어 있지 않은 함수 (이를 사용자 정의 함수: User Defined Function 이라고 함)를 만들때 사용합니다.

VB 편집기에서 주로 사용하는 항목들은 프로젝트탐색기, 속성창, 코드창 등이다.
만일 위와 같은 창들이 안보일 경우 VB 편집기 상단 메뉴에서 보기 > 프로젝트탐색기, 보기 > 속성창, 보기 > 코드 등을 선택하면 보일 것이다.

VBA코드 입력하기(9s.xls)

아주 간단한 형태의 Sub Procedure를 코딩해 보자
VB 편집기의 프로젝트 탐색기에서 모듈을 하나 추가한 후 아래 코드를 그대로 입력해 보자.

```
Sub Temp_1()
   Activesheet.Range("b1").Select
End Sub
```

앞의 강좌에서 설명한 바와 같이 Visual Basic 에서 sub Procedure의 이름은 문자로 시작해서 문자/숫자/밑줄의 조합으로 가능하며, 공백은 없어야 하고, 총 255자 이내라야 한다.위에서는 이 규칙에 따라 Temp_1을 사용했다. 이렇게 코딩한 후 Visual Basic 편집기 내의 메뉴에서 실행 > Sub/사용자 정의 폼 실행을 택하면 (또는 단축키 F5) 엑셀 활성화된 시트 (Activesheet) 의 B1이 선택된다. 또는 앞의 강좌에서 배운 대로 워크시트 상에서 도구 > 매크로 > 매크로 실행으로 위 Procedure 를 실행시킬 수도 있고, 또한 매크로 편집으로 가서 단축키를 입력한 후
단축키로 실행할 수도 있다.

•VBA를 사용하는 이유

첫째, 모든 작업을 매크로 기록 만으로 자동화(Automation) 시킬 수는 없다.
예를 들어 특정 디렉토리 내에 txt 형태의 파일이 200여개 정도 있다고 할때, Visual Basic Recorder를 사용해서 작업을 하려면 200여개의 파일을 일일이 열고나서 매크로 기록 후 다시 반복하도록 매크로를 실행해야 할 것이다. 이런 식으로 작업한다는 것은 너무나 비효율적이다. 몇 줄 안돼는 Visual Basic 코드만 쓰면 이런 작업은 쉽게 할 수 있다. 또 한가지 예를 들어 논리 판단의 문제가 들어갈 경우 매크로 기록에는 한계가 있다. 어떤 값을 보고 Yes/No 에 따라, 또는 값이 크거나 작음에 따라 각기 다른 자동화 작업이 이루어 지도록 하고 싶다면 논리 판단을 할 수 있도록 coding을 해 주어야 하는데, 이 경우 매크로 기록만으로는 한계가 있다.

둘째, 직접 Coding 하는 것이 매크로 기록에 의한 방법보다 훨씬 flexible 하다.
매크로 기록은 똑같은 작업만 그대로 하는 것이지만, 한번 Coding을 제대로 해 놓으면, 몇몇 Procedure들의 결합으로 작업을 아주 쉽게 할 수 있다. 대부분 회사일이 비슷한 작업의 연속인데, 매크로 기록은 아주 똑같은 작업만 할 뿐 약간만 다른 작업이라도 코드를 수정해 주지 않는 이상 다른 작업은 못한다. 하지만 직접 Code를 짜서 입력해 놓으면 수정하기도 간단할 뿐만 아니라, 비슷비슷한 회사 업무를 아주 효율적으로 할 수가 있다. 또한 꼭 필요한 몇몇 Code들은 별도 파일 형태로 저장해 놓으면 자기가 한번 짜 놓은 Code 들의 결합으로 아주 긴 Process의 업무도 쉽게 자동화시킬 수가 있는 것이다.

버튼을 만들자(10.xls)

지금까지는 도구 > 매크로 실행으로 또는 단축키로 매크로를 실행하는 방법을 배웠다.
그리고 Visual Basic Editor에서 직접 매크로를 실행할 수도 있다는 것을 익혔다.

이 외에도 매크로는 버튼을 만들어 실행 시킬 수도 있다.

아래와 같은 버튼을 만들고

코드를 입력한다.,

MsgBox Application.UserName & "님 안녕하세요.!", , “Hyundai-Steel"

Application.UserName 이라는 것은 컴퓨터 이름을 뜻한다.
만일 컴퓨터 이름을 "주윤발"로 해 놓은 사람은 주윤발님 안녕하세요 란 메시지가 생길 것이다.

MsgBox에 대하여

메시지 상자란 말 그대로 간단한 메시지를 보여주는 상자이다.

MsgBox 연습1

코드를 살펴보면,

```
Sub MakeMsgBox_1()
MsgBox "프린터가 켜져 있습니까?",vbOkOnly + vbOKCancel, "krazy"
End Sub
```

구 문 :	MsgBox(prompt[, buttons] [, title] [, helpfile, context])
prompt	필수. 대화 상자 내의 메시지로 나타나는 문자열 식입니다. prompt의 최대 길이는 약 1024 문자이며, 사용된 문자의 너비에 따라 다릅니다. prompt 구성이 1줄 이상이면, 캐리지 리턴 문자 (Chr(13)), 라인 피드 문자 (Chr(10)), 캐리지 리턴 및 라인 피드 문자 (Chr(13) & Chr(10))를 이용하여 줄 구분을 합니다.
Buttons	선택. 단추의 수와 형태와 사용할 아이콘의 형태, 기초 단추의 정체 및 메시지 상자의 양식을 지정하는 값의 합을 나타내는 숫자 식입니다. 생략 되었을 때 buttons의 기본값은 0 입니다.
Helpfile	선택. 도움말 파일을 이용하여 상세한 도움말을 대화 상자에 제공합니다. helpfile이 부여되면, context도 반드시 부여되어야 합니다.
Context	선택. 도움말 작성자가 적절히 작성한 도움말 항목에 부여된 도움말 문 번호를 나타내는 숫자 식입니다. context가 부여되면, helpfile도 반드시 부여되어야 합니다.

위의 도움말 처럼 메시지 내용인 prompt만 필수이고 나머지는 선택 요소이므로 [ ]로 묶여 있다.
즉, 생략 가능하다는 말이다.

Msgbox의 button에 관하여

일반적인 형태의 MsgBox는 아래와 같은 형식임을 알았다.

Sub MakeMsgBox()

MsgBox prompt, buttons, title

End Sub

*Button 인수

상수	값	설 명
vbOKOnly	0	[확인] 단추만 나타냅니다.
VbOKCancel	1	[확인] 와 [취소] 단추를 나타냅니다.
VbAbortRetryIgnore	2	[중단], [재시도], 및 [무시] 단추를 나타냅니다.
VbYesNoCancel	3	[예], [아니오], 및 [취소] 단추를 나타냅니다.
VbYesNo	4	[예] 및 [아니오] 단추를 나타냅니다.
VbRetryCancel	5	[재시도] 및 [취소] 단추를 나타냅니다.
VbCritical	16	[중대 메시지] 아이콘을 나타냅니다.
VbQuestion	32	[질의 경고] 아이콘을 나타냅니다.
VbExclamation	48	[메시지 경고] 아이콘을 나타냅니다.
VbInformation	64	[메시지 정보] 아이콘을 나타냅니다.

Msgbox기록법

앞에서는 아래와 같은 형태의 순차적인 기록법을 택해 왔다.
MsgBox "프린터가 켜져 있습니까?", vbYesNoCancel + vbDefaultButton3,"krazy“
이는 어디까지나 Prompt, Buttons, Title 순으로 해야한다는 의미이다.

하지만 위 코드를 이렇게 바꿔도 된다.
MsgBox Title:="krazy", Prompt:="프린터가 켜져 있습니까?", Buttons:= vbYesNoCancel +
vbDefaultButton3,
이 경우는 순서를 마음대로 해도 된다.

다만 Title 이라는 인수 뒤에 반드시 := 를 붙여야 한다.

이 경우도 물론 앞에서 처럼 상수 대신 숫자값을 적어도 무방하다.

MsgBox의 반환값(10.xls)

메시지 상자는 단순 경고 뿐만 아니라, 간단한 형태의 질문이다. 따라서 메시지 상자가 의미를 지니려면, 예/아니오 등을 택할 때 다른 행동을 취하도록 지정할 수 있어야 하겠다.

예를 택하면 빨강으로 바뀌고, 아니오를 택하면 다시 흰색으로 바뀐다.

```
Sub ReturnValue()
        Answer = MsgBox("현재 셀을 빨간색으로 칠할까요 ?", vbYesNo + vbQuestion, "이것은 Return Value에 대한 연습입니다.")
        If Answer = vbYes Then
                ActiveCell.Interior.ColorIndex = 3
        Else
                ActiveCell.Interior.ColorIndex = 0
        End if
End Sub
```

위에서 처럼 "예"는 vbYes로 받는다. 답에 대해서 무엇으로 받는가에 대한 반환값은 아래와 같다.

상수	값	설명
vbOK	1	[확인]
vbCancel	2	[취소]
vbAbort	3	[중단]
vbRetry	4	[재시도]
vbIgnore	5	[무시]
vbYes	6	[예]
vbNo	7	[아니오]

앞의 경우처럼 vbOK 대신 아라비아 숫자 1을 적어도 무방하다.

InputBox(15.xls)

MsgBox와 비슷한 형태로 InputBox가 있다.

MsgBox가 객관식 질문인데 반해 InputBox는 주관식으로 생각하면 된다.

만일 섭씨 온도를 화씨 온도로 바꾸어 표현하려고 할때,

섭씨8도를 화씨로 바꾸면 화씨46.4도 입니다.

코드를 살펴보면,

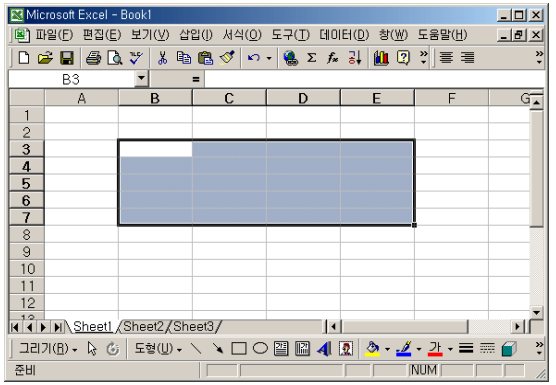
```
Sub C_to_F()
   C = InputBox("섭씨 온도를 입력해 주세요")
   F = C * 9 / 5 + 32
   ActiveSheet.Range("a12").Value = "섭씨" & C & "도를 화씨로 바꾸면 화씨" & F & "도 입니다."
End Sub
```

문자열 사이에 변수 (여기서는 C 와 F) 를 적어 넣을 때는 " & 변수 & " 형태로 해야 한다.

Object(16.xls)

VBA 는 객체지향 프로그래밍 (OOP : Object Oriented Programming) 언어이다.

객체지향언어에서 모든 가리키는 대상은 object에 해당한다.



우주, 지구, 대한민국, 서울, 종로구 등 모든 것이 object 이다.

VBA에서는 workbook, worksheet, range 등이 object에 해당한다.

현재 시트의 B11:C14 를 언급할 때

```
Activesheet.Range("b11:c14")
```

로 표시하는데, activesheet 도 object 이고 Range("b11:c14") 도 하나의 object이다.

만일 어떤 범위에 Sales 라는 이름을 지정했을 경우 (엑셀 메뉴에서 삽입>이름>정의 메뉴를 선택해서 특정 범위에 대한 이름을 지정할 수 있음), 범위를 다음과 같이 지정할 수 있다.

```
Range("Sales")
```

여러가지 object 가 있을 수 있는데, 여기서는 자주사용하는 것 중 하나인 Activecell과 Selection의 차이점을 짚고 넘어가자.

위 그림과 같이 범위선택이 되어 있을 경우 activecell 이라는 object는 B3을 가리키고, Selection이라는 object는 B3:E7을 가리킨다.

Collections

Object에는 Range와 같이 단수 개념의 Object가 있는 반면, Worksheets, Workbooks와 같은 복수 형태로 쓰는 Object가 있다. 이러한 복수 형태의 Object들을 별도의 개념으로 떼어 내어 Collections 라는 개념으로 사용한다. 집들이 모여 빌딩이 되고, 빌딩들이 모여 도시가 되듯이, 엑셀에 사용되는 Object에도 계층구조가 있다.

우리가 보통 주소를 표기할 때 "대한민국. 서울. 관악구. 신림동“ 과 같은 형식으로 표기하듯, Excle VBA에서는
Application.Workbooks("VBA_005.xls").Worksheets("Collections").Range("a12")
와 같이 Application이라는 Object, Workbooks라는 object, Worksheets라는 Object, Range 라는 Object를 사용한다.

Application은 Excel VBA에서 최상위 Object 인데, 엑셀, 파워포인트 등도 Application이라는 하나의 Object이다.
위의 경우 앞의 것을 다 생략하고 Range("a12")라고 해도 무방하지만, 가급적이면 Application을 생략하더라도, Workbooks object 개념까지는 명시적으로 표시해 주는 것이 낫다.

아주 긴 여러 개의 코드를 합쳐서 하나의 자동화 작업을 할 때, 구체적으로 적어 줘야 수정할 때 햇갈리지 않고, 실제 작동하는데 있어서도 혼란이 없다.

Application과 같은 Object 는 대부분 생략이 가능하다.

우리가 보통 편지를 쓸 때 대한민국 서울시 라고 하지 않고 서울시 라고 하더라도 대한민국 모든 우체부가 알아 듣듯이, 엑셀 내에서도 Application 이라고 적지 않더라도, 알아서 인식한다.

Properties(18.xls)

Object 만으로 하나의 VBA 구문이 될 수는 없다.
영어에도 주어 동사가 있듯이, VBA 구문에도 Object + Property 또는 Object + Method 의 형태를 띤다.
그리고 Object와 Object, Object와 Property, Object와 Method 사이는 점(.) 으로 연결된다.

코드를 보면,

```
Workbooks("VBA_005.xls").Worksheets("Properties").Range("a12").Value=15
```

Object

Properties

결국 구문은 object.object.object.property 형태이다. 하나의 예를 더 들면,

```
Workbooks("VBA_005.xls").Worksheets("Properties").Range("a12").ColumnWidth=12
```

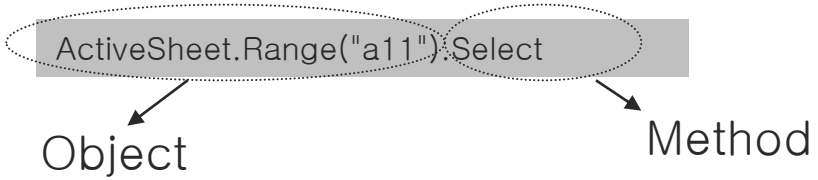
Object

Properties

열 너비를 12로 하라는 의미인데 이 구문도 비슷한 구조이다.
value, columnwidth 외에도 NumberFormat, colorindex 등 여러 Property가 있습니다.

Methods(19.xls)

앞에서 배운 value, columnwidth 등의 property(속성) 외에도 선택하라(Select), 복사하라(Copy), 붙여넣어라(Paste) 등의 명령구문을 작성할 수 있는데, 이는 Method(방법)에 해당한다.



이 외에도 copy, paste 등이 Method 개념이다.

ActiveSheet.Range("a11:a13").Copy
두 구문 모두 object + Method 형태의 구문이다.

위위 예에서 주의할 것은 Destination과 같이 매개변수로 사용될 경우= 이 아니라 반드시 := 형태로 입려해 줘아 한다는 것이다.

위 두 줄은 아래 한줄로도 표시 가능하다.
ActiveSheet.Range("a11:b13").Copy Destination:=ActiveSheet.Range("b20")

Event(20.xls)

VBA에서 알아두어야 할 또하나의 개념으로 event가 있다.
특정 셀을 더블클릭했을때, 특정 버튼을 눌렀을 때 어떤 작업을
하도록 할 수 있다는 것이다.
대표적인 예가 auto_open 을 들 수 있는데,
아래 구문은 파일이 열리는 Event가 발생하자 마자
안녕하세요 ---님 ! 이라는 메시지 상자를 띄우는 구문이다.

```
Sub Auto_Open()

        MsgBox "안녕하세요" & Application.UserName & "님 !"

End Sub
```

Range Object

VBA에서 가장 많이 사용하게되는 object가 Range Object 라고 생각됩니다.

간단한 예를 들어보면,

```
Application.Activesheet.Range("d5").Interior.ColorIndex = 3
```

지난번 강좌에 설명드린 것처럼 application은 object 중에 가장 상위 개념입니다.
이 경우 application을 생략하고 Activesheet 부터 적어도 됩니다.
Application을 꼭 적어 줘야 하는 구문도 있지만, 이는 특수한 경우이고, 일반적으로 Application을 생략하고 적습니다.
대부분의 코드는 모듈에 적게 되는데, 모듈에 적을 경우에는 반드시 Activesheet 라는 것은 적어야 하고, Visual Basic 편집기의 특정 sheet에 코드를 적을 경우에는 activesheet 라는 것까지 생략해도 됩니다.

위의 예에서 acvtivesheet는 Object, Range("d5")와 interior도 Object, Colorindex는 Property에 해당됩니다.
엑셀의 현재 활성화된 시트의 셀 d5 셀의 내부 색상을 빨간색으로 하라 의미죠---.

Range 하나의 셀도 될 수 있지만, 일정 영역을 가리킬 수도 있습니다.
ActiveSheet.Range("E22:G23").Select
또한 아래와 같이 서로 떨어져 있는 영역의 지정도 가능합니다.
ActiveSheet.Range("C32:C35,E32:E35,G32:G35").Select
물론 비연속적인 셀들도 선택이 가능하구요.
ActiveSheet.Range("C46,E46,G46").Select
만일 특정 범위에 이름을 정의해 놓은 상태라면, 아래처럼 이름에 대한 범위도 지정할 수 있죠.
ActiveSheet.Range("Table").Select
이름은 엑셀 메뉴의 삽입 > 이름 > 정의 에서 이름을 정하고 범위를 선택하면 되는데, 위의 경우 "D56:G60" 에 해당되는 범위를 table 이라는 이름으로 미리 정의해 놓은 경우입니다.
특정 셀에 이름을 정의해 놓고, 특정 셀에서 이름이 정의된셀까지의 범위도 선택가능합니다.

Range Object(22.xls)

ActiveSheet.Range(ActiveSheet.Range("E63"), ActiveSheet.Range("TargetCell")).Select
위의 경우는 Activesheet 가 여러 번 나오므로 다음과 같이 With --- End With 구문으로 간결하게 할 수도 있습니다.

```
With ActiveSheet
    .Range(.Range("E63"), .Range("TargetCell")).Select
  End With
```

현재 시트 외의 범위에 대한 계산도 할 수 있는데,
앞의 개요 시트의 Q3에서 Q12까지의 셀에는 1부터 10까지의
숫자가 들어 있는데, 이들 범위의 합을 메시지 상자로 나타내어 보도록 하겠습니다.

```
With Sheets("개요")
    MsgBox WorksheetFunction.Sum(.Range(.Range(“a1"), .Range(“b3")))
  End With
```

주의!!!
일반 엑셀 함수를 VBA로 표현할 때에는
"WorksheetFunction.엑셀함수" 와 같은 형식으로 해야 합니다.

Cells Property(23.xls)

Cells Property는 다음과 같은 형태로 사용됩니다.
Cells(row, Column)

Cells(8,4)는 8행 4열의 의미인데, 앞 쉬트에서 설명드린 Range를 이용하면 Range("d8") 과 같은 의미죠.
Range는 직접 어떤 셀 또는 범위를 입력하기 편리한 반면, Cells의 경우 변수와 함께 사용할 경우, 특히 순환문에서 편리하게 사용됩니다.

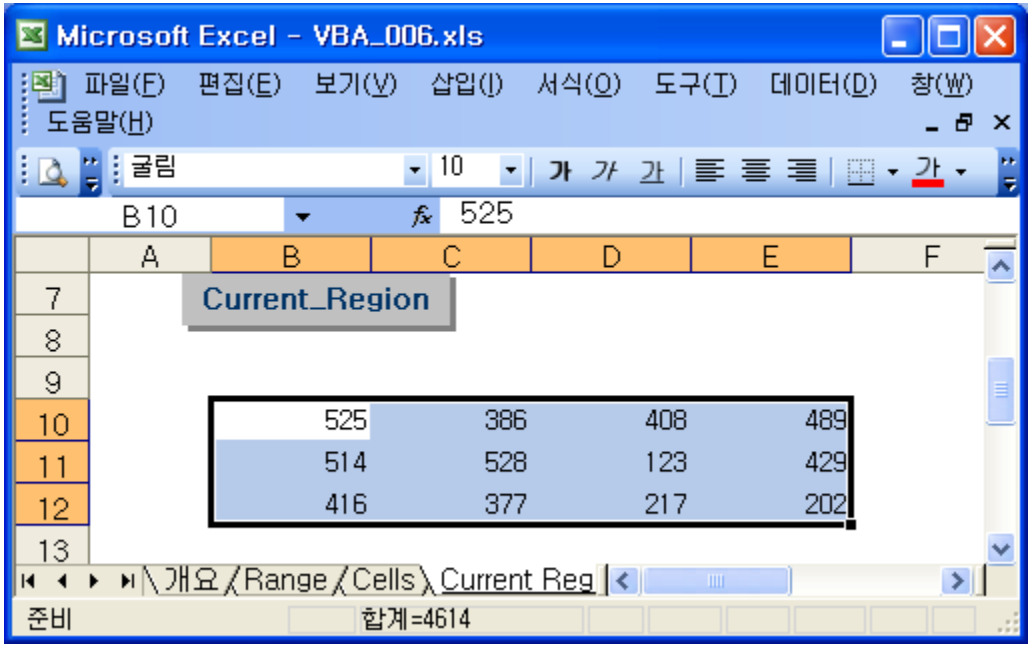
```
Sub Cells_2()

    Dim i As Integer
    Dim J As Integer

    For i = 1 To 5
        For J = 1 To 7
            ActiveSheet.Cells(5 + i, 1 + J).Value = i + J
        Next J
    Next i

End Sub
```

Cells는 또한 sheet 전체를 선택할 때 (단축키 Ctrl + a) 에도 사용됩니다.
이 경우 Cells(row, column)의 형태가 아니라, 그냥 Cells 만 적어주면 됩니다.

```
ActiveSheet.Cells.select
```

CurrentRegion Property(24.xls)

데이터가 들어 있는 특정 영역을 모두 선택할 때
보통 단축키 Ctrl + * 를 쓰거나 또는 Ctrl + Shift + 오른쪽화살표 + 아래쪽화살표 를 선택하는데 Visual Basic 코드로는 Current Region을 사용하면 됩니다.




```
ActiveSheet.Range("B5").CurrentRegion.Select
```

Offset Property(25.xls)

Offset은 특정 셀을 중심으로 아래로 몇칸, 오른쪽으로 몇칸 떨어져 있는 셀을 선택할 때 사용됩니다.
각 셀의 숫자들 중에서 20보다 큰 값들만 빨간색으로 칠하고 싶을 때, 각 셀들을 하나씩 택해서 그 값이 20보다 작은 값인지를 체크해 보아야 하는데, 이 경우 Offset을 사용하면 편리합니다.

```
Sub Offset_2()
ActiveSheet.Range("B3").Select
    Do
        If ActiveCell.Value < 20 Then
            ActiveCell.Interior.ColorIndex = 3
        Else
            ActiveCell.Interior.ColorIndex = 0
        End If
        ActiveCell.Offset(1, 0).Activate
    Loop Until IsEmpty(ActiveCell)
End Sub
```

여기서 조건문 IF 는
IF   -------Then

Else

End IF
의 형태로 사용되고, 아래의 Do --- Loop Until구문은
Do

Loop Until
Until 이하의 조건을 만족할 때까지 Do 와 Loop Until 사이의 코드를 반복실행하라는 의미입니다.
앞의 경우 IsEmpty(ActiveCell) 이므로, 현재 활성화된 셀값이 공란일때까지 계속 구문을 반복 실행하라는 의미입니다.

End Property

```
ActiveSheet.Range("A1").End(xlDown).Select
```

이는 A1셀에서 Ctrl + 아래쪽 화살표를 누른 것과 같은 효과입니다.
즉 A1에서 시작해서 아래쪽 방향으로 바로 다음에 공란이 나오는 마지막 셀을
선택해 줍니다.
비슷한 식으로 End(xlToRight), End(xlUp), End(xlToLeft) 도 있습니다.

```
ActiveSheet.Range(ActiveSheet.Range("B3"), ActiveSheet.Range("B3").End(xlToRight)).Select
```

위 코드는

```
With ActiveSheet
    .Range(.Range("B3"), .Range("B3").End(xlToRight)).Select
End With
```

와 동일합니다.

```
ActiveSheet.Range(ActiveSheet.Range("C3"),
        ActiveSheet.Range("C3").End(xlDown)).Select
```

위 코드는

```
With ActiveSheet
.Range(.Range("C3"), .Range("C3").End(xlDown)).Select
End WIth
```

와 동일합니다.

```
ActiveSheet.Range(ActiveSheet.Range("G3"),
        ActiveSheet.Range("G3").End(xlDown)).Offset(0,2).Select
```

위 코드는

```
With ActiveSheet
    .Range(.Range("G3"), .Range("G3").End(xlDown)).Offset(0, 2).Select
End WIth
```

와 동일합니다.

IF 문

앞의 예에서 처럼 조건이 하나일 경우에는
IF (조건 1) Then

Else

End If

와 같은 식으로 하면되고,
조건이 둘일 경우에는 아래와 같은 식으로 하면 됩니다.

IF (조건 1) Then

Elseif (조건 2) Then

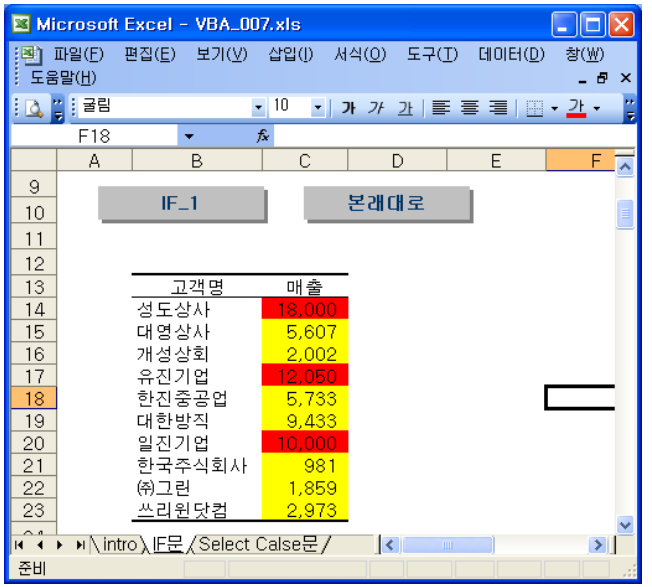
Else

End If

그리고 조건이 둘 이상일 때에는 Elseif를 계속 사용해도 되지만,
다음 시트에서 설명드릴 Select Case 구문을 이용하는 편이
쉬울 것입니다.

IF문 예제(28.xls)

```
Sub IF_1()

  Dim i As Integer      ' 변수 선언 : i 를 정수로 선언하고,
  Dim k As Range      ' 변수 선언 : k 를 범위로 선언

  For i = 1 To 10       ' i는 1부터 10까지

    Set k = ActiveSheet.Range("C10").Offset(i, 0)        '현재시트의 C13을 기준으로 밑으로 i번째 셀을 k 라 한다.

    If k >= 10000 Then      ' 만일 k 가 10000 보다 크거나 같을 경우,
      k.Interior.ColorIndex = 3   ' 셀 내부의 색을 빨간색으로 하고,
    Else                          ' 아닐 경우 (k가 10000보다 작을 경우)
      k.Interior.ColorIndex = 27   ' 셀 내부의 색을 노란색으로 하라
    End If

  Next i

End Sub
```



Select Case문

앞의 IF문과 비슷한 구조인데, Select Case 문의 경우

```
Select Case (변수값)
  case Is A     ' 변수값 =  A 인 경우
    -----
  case Is B     ' 변수값 =  B 인 경우
    -----
  case Is C     ' 변수값 =  C 인 경우
    -----
  case Else   ' 그 외의 값인 경우
    -----
End Select
```

IF 문과는 달리 여러 개의 조건 비교가 가능하다는 장점이 있지만, Select Case는 특정 변수값에 대한 단순 비교라는 점에서 IF 문과는 쓰임에 약간의 차이가 있습니다. 그리고 위의 경우 Case is 에서 is 는 생략 가능하고,

앞의 예에서는

```
Case Is < 1000
    k.Offset(0, 1) = "  - 1천원"
```

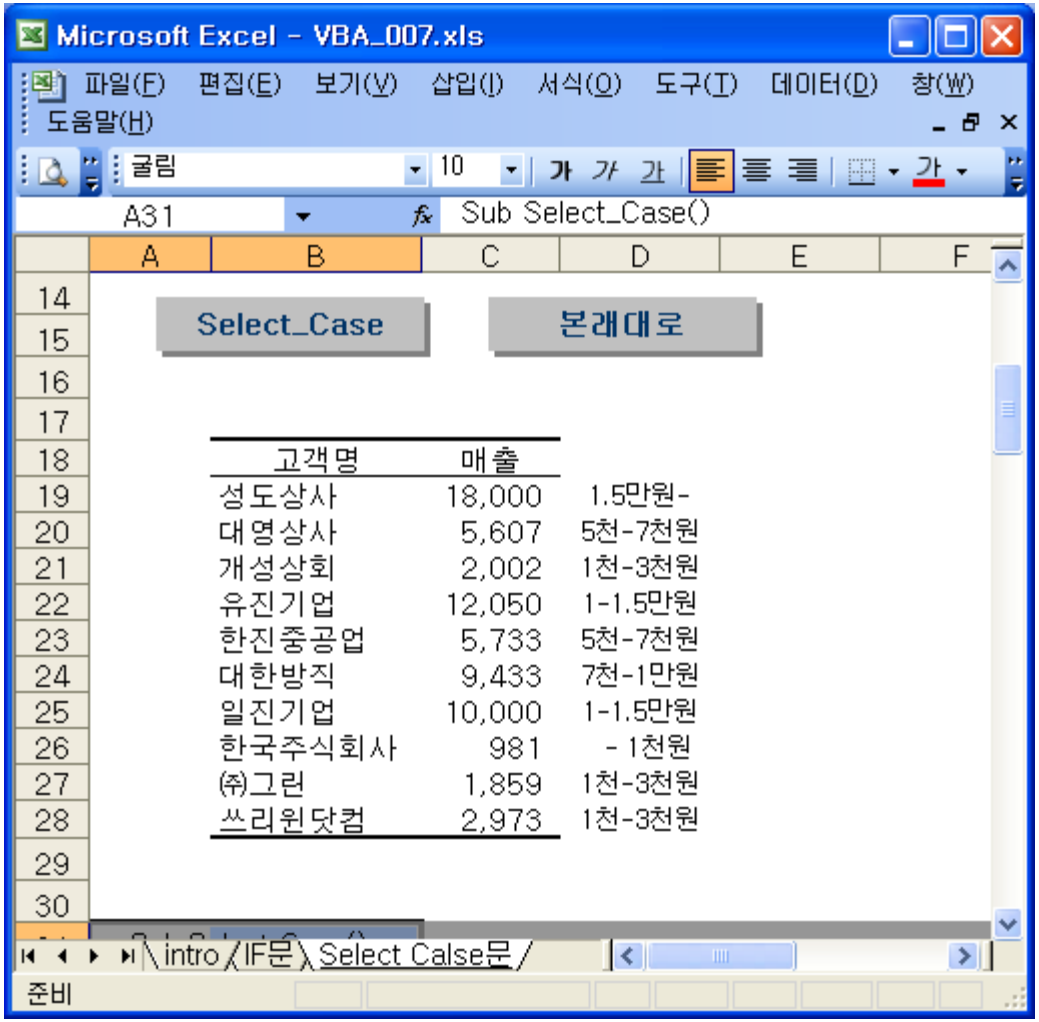
라는 표현 대신에

```
Case Is < 1000 :    k.Offset(0, 1) = "  - 1천원“
```

와 같은 식으로 한줄로 사용해도 됩니다.

Select Case문 예제(30.xls)

```
Sub Select_Case()

  Dim i As Integer
  Dim k As Range

  For i = 1 To 10
    Set k = ActiveSheet.Range("C6").Offset(i, 0)
    Select Case k
      Case Is < 1000
          k.Offset(0, 1) = "  - 1천원"
      Case Is < 3000
          k.Offset(0, 1) = "1천-3천원"
      Case Is < 5000
          k.Offset(0, 1) = "3천-5천원"
      Case Is < 7000
          k.Offset(0, 1) = "5천-7천원"
      Case Is < 10000
          k.Offset(0, 1) = "7천-1만원"
      Case Is < 15000
          k.Offset(0, 1) = "1-1.5만원"
      Case Else
          k.Offset(0, 1) = "1.5만원-"
    End Select
  Next i

End Sub
```



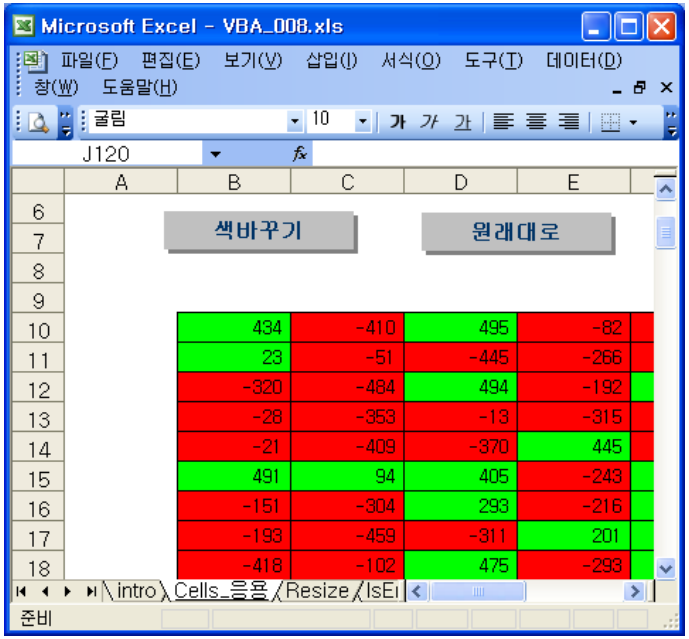
Cells Property응용(31.xls)

특정 범위 내 각 셀에 대해 하나씩 조건에 맞는지를 알아보려 할 때,
For Each 특정셀 in 셀범위

Next 특정셀
과 같은 구문을 사용하게 됩니다.

```
Sub ChangeColors_1()

    Dim MyRng
    Dim i
    Dim j

    Set MyRng = ActiveSheet.Range("Table")

    For i = 1 To MyRng.Rows.Count
        For j = 1 To MyRng.Columns.Count
            If MyRng.Cells(i, j).Value < 0 Then
                MyRng.Cells(i, j).Interior.ColorIndex = 3
            Else
                MyRng.Cells(i, j).Interior.ColorIndex = 4
            End If
        Next j
    Next i
End Sub
```

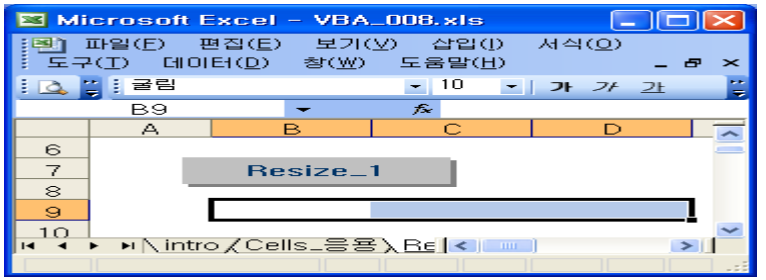
```
Sub ChangeColors_4()
  Dim MyRng
   For Each MyRng in Activesheet.Range("Table")
        If MyRng.Value < 0 Then
          MyRng.Interior.ColorIndex = 3
        Else
          MyRng.Interior.ColorIndex = 4
        End If
  Next MyRng
End Sub
```



Resize Property(32.xls)

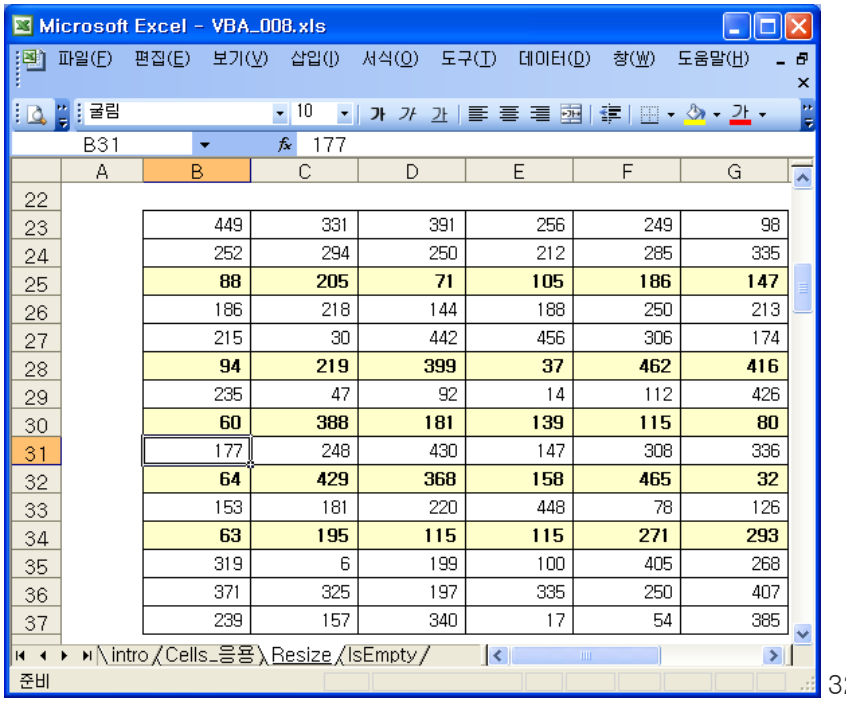
Resize Property는 본래 Range의 좌상단은 동일하고, 행열만 다를 경우의 범위를 다시 지정할 때 사용됩니다.

```
ActiveSheet.Range("B9").Resize(1, 3).Select
```



테이블에서 첫번째 행에 있는 값이 100보다 작은 열만 찾아서 그열의 숫자가 들어 있는 6번째 행까지 옅은 노란색을 칠하고, 글자도 굵게 표시하고 싶다면, 아래와 같이 하면 됩니다.

```
Sub Resize_2()
    Dim MyRng
    For Each MyRng In ActiveSheet.Range("b23:b37")
        If MyRng.Value < 100 Then
            MyRng.Resize(1, 6).Font.Bold = True
            MyRng.Resize(1, 6).Interior.ColorIndex = 19
        Else
            MyRng.Resize(1, 6).Font.Bold = False
        End If
    Next MyRng
End Sub
```



IsEmpty Function(33.xls)

미국	CPU	450M	256
		1G	212
	Ram	128M	105
		64M	188
일본	CPU	450M	37
		1G	14
	Ram	128M	139
		64M	147
		256M	158
중국	CPU	1G	448
		450M	115
	Ram	64M	100
		128M	335
		128M	17

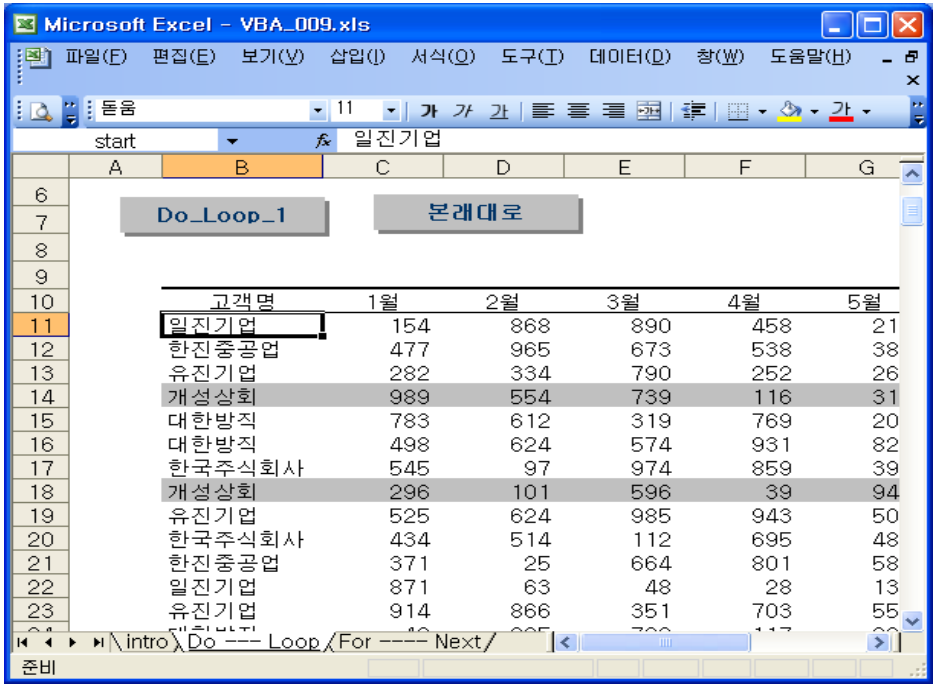
→

미국	CPU	450M	256
미국	CPU	1G	212
미국	Ram	128M	105
미국	Ram	64M	188
일본	CPU	450M	37
일본	CPU	1G	14
일본	Ram	128M	139
일본	Ram	64M	147
일본	Ram	256M	158
중국	CPU	1G	448
중국	CPU	450M	115
중국	Ram	64M	100
중국	Ram	128M	335
중국	Ram	128M	17

```
(프로시져 1)
Sub IsEmpty_1()
    ActiveSheet.Range("b27").Activate
    Call Pro
    ActiveSheet.Range("C27").Activate
    Call Pro
End Sub

(프로시져 2)
Sub Pro()
    Do
    If IsEmpty(ActiveCell.Value) Then
       ActiveCell.Value = ActiveCell.Offset(-1,
         0).Value
       ActiveCell.Offset(1, 0).Activate
    Else
       ActiveCell.Offset(1, 0).Activate
    End If

    Loop Until IsEmpty(ActiveCell.Offset(0,
        2).Value)

End Sub
```

Do-Loop(34.xls)

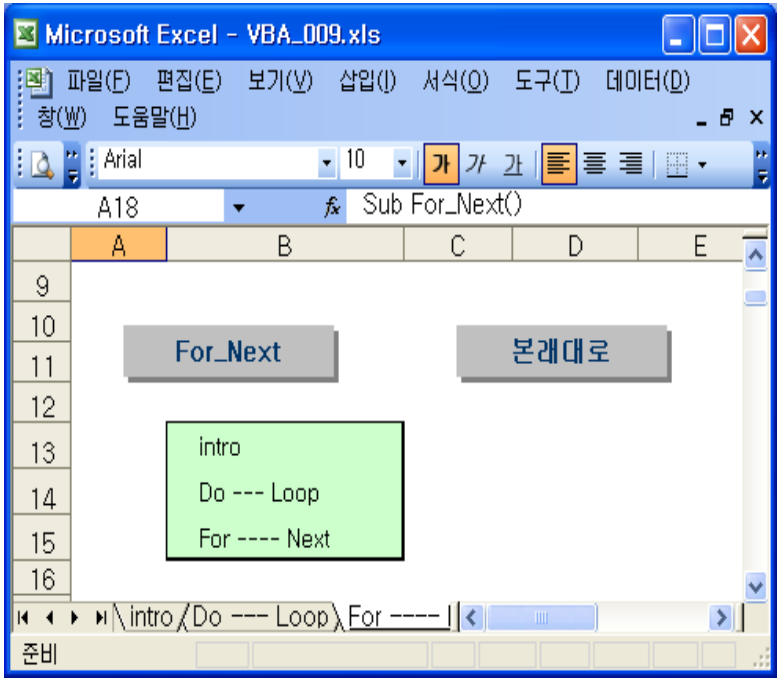
```
Sub Do_Loop_1()

  Application.ScreenUpdating = False

  Dim CompanyName As String

  CompanyName = InputBox("칠하고 싶은 기업이름을 입력하세요", "krazy", "개성상회")
  Range("start").Activate

  Do While ActiveCell.Value <> ""
     If ActiveCell.Value = CompanyName Then
        ActiveCell.Resize(1, 6).Interior.ColorIndex = 15
        ActiveCell.Offset(1, 0).Activate
     Else
        ActiveCell.Offset(1, 0).Activate
     End If
  Loop

  Range("start").Activate
  Application.ScreenUpdating = True
End Sub
```



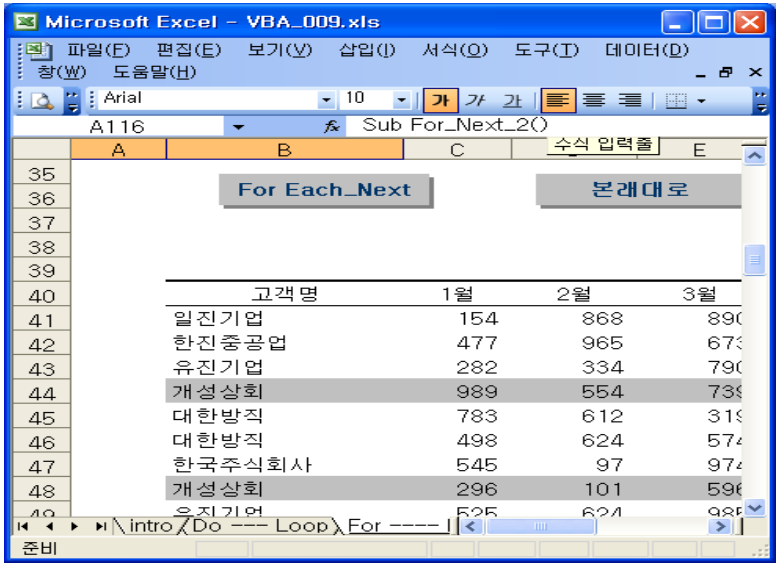
For – Next(35.xls)

현재 화일의 시트명을 나열

```
Sub For_Next()

  For i = 1 To Worksheets.Count
     Cells(12 + i, 2).Value = Worksheets(i).Name
  Next i

End Sub
```

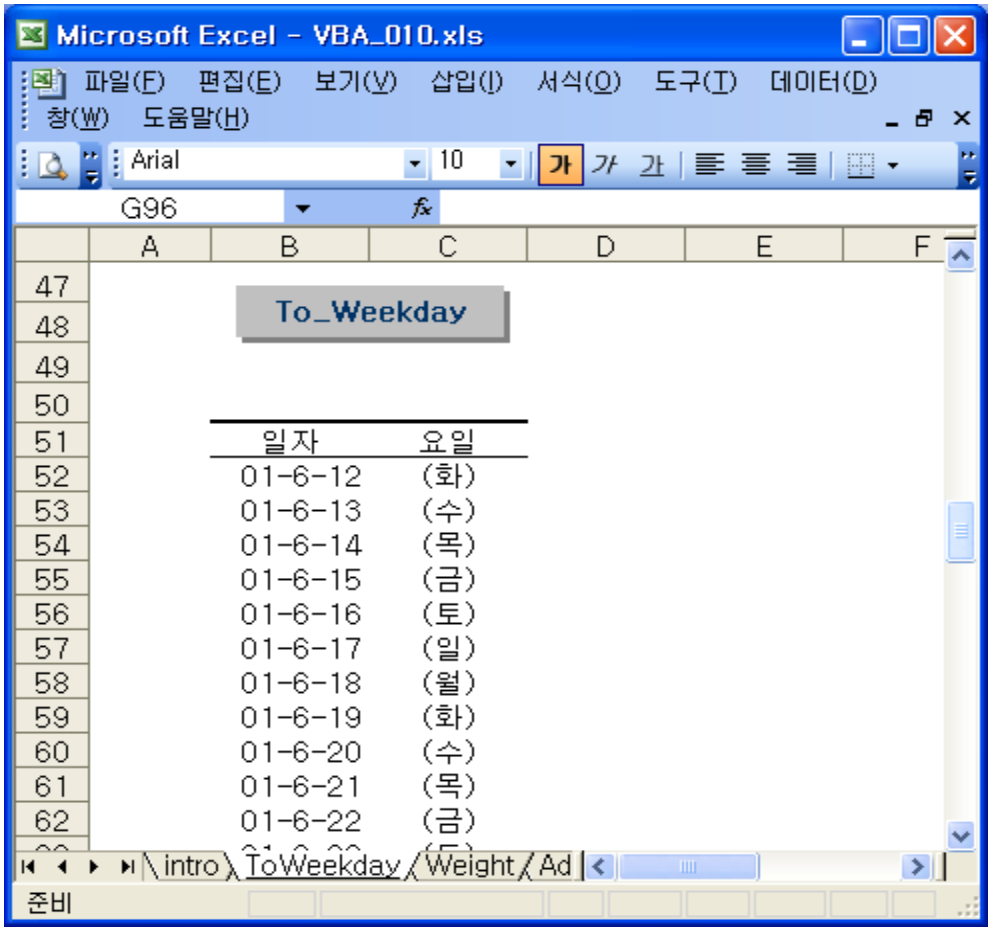



```
Sub For_Next_2()

  Dim CompanyName As String
  Dim rng As Range

  CompanyName = InputBox("칠하고 싶은 기업이름을 입력하세요",  "krazy",
        "개성상회")

  For Each rng In ActiveSheet.Range("b41:b112")
     If rng = CompanyName Then
        rng.Resize(1, 6).Interior.ColorIndex = 15
     End If
  Next rng

End Sub
```



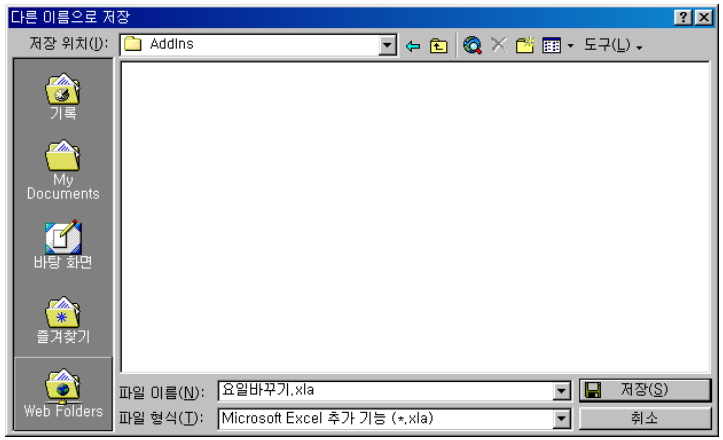
ToWeekDay(37.xls)

```
Function Tow(MyDate As Date)
    i = WorksheetFunction.Weekday(MyDate)
   Select Case i
     Case 1
        Tow = "(일)"
     Case 2
        Tow = "(월)"
      Case 3
        Tow = "(화)"
      Case 4
        Tow = "(수)"
      Case 5
        Tow = "(목)"
      Case 6
        Tow = "(금)"
      Case 7
        Tow = "(토)"
   End Select
End Function
```

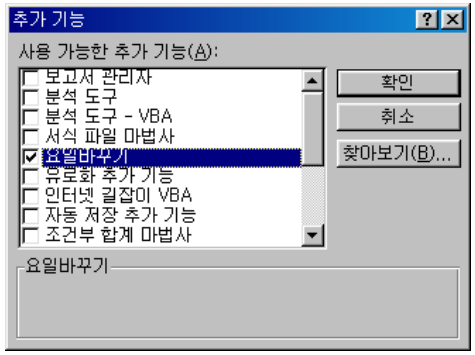


Add-In

필요할 때마다 자주쓰는 사용자정의 함수가 있는 파일을 열어서 값을 붙여넣고, 수식을 계산하는 것은 좀 번거롭다는 생각이 듭니다. 이럴 때 Add-in 이라는 기능을 활용하면 편리합니다. 본 파일을 다른이름으로 저장하되 "요일바꾸기.xla" 등으로 저장해 보세요.



이 경우 아무 디렉토리에 저장해도 상관없지만 보통은 Office 프로그램이 깔려있는 "Program Files₩Microsoft Office₩Office₩Addins" 에다가 저장해 둡니다.
그 다음에 도구> 추가기능에서 그 파일을 다시 지정해 줍니다.



이렇게 해 놓으면 어느 파일에서나 Toweekday, 비만도 등의 함수가 작동하게 됩니다.

배열(38.xls)

배열이란 하나 이상의 데이터를 담을 수 있는 VBA 변수들을 말합니다.
배열은 단일 변수를 사용하는 것보다 훨씬 유연한(Flexible) 작업을 할 수 있습니다.

```
Sub Array_1( )

  Dim MyWeek As Variant
  Dim i As Integer

  MyWeek = Array("월", "화", "수", "목", "금", "토", "일")

  For i = 0 To 6
    Range("start").Offset(0, i) = MyWeek(i)
  Next i

End Sub
```

Option Base(38.xls)

```
Option Base 1

Sub Array_2( )
  Dim MyWeek As Variant
  Dim i As Integer

  MyWeek = Array("월", "화", "수", "목", "금", "토", "일")

  For i = 1 To 7
    Range("start2").Offset(0, i - 1) = MyWeek(i)
  Next i

End Sub
```

앞에서 배열의 시작값은 0 으로 한다고 했는데, 0 으로 하기 싫고
1로 하고 싶을 때 맨 윗줄에 Option Base 1 이라고 적어주면 됩니다.

Ubound 와 Lbound(38.xls)

```
Sub Array_3( )

  Dim MyWeek As Variant
  Dim i As Integer

  MyWeek = Array("월", "화", "수", "목", "금", "토", "일")

  For i = LBound(MyWeek) To UBound(MyWeek)
    Range("start3").Offset(0, i) = MyWeek(i)
  Next i

End Sub
```

위의 예에서는 배열의 개수가 얼마되지 않아서 일일이 숫자를 지정해 줘도 되지만, 배열의 개수가 많거나 아니면 유동적일 경우 Lbound와 Ubound 함수를 사용하면 편리합니다.
여기서 Lbound(MyWeek)는 제일 낮은배열순서인 0을 Ubound(MyWeek)는 제일 높은 배열순서인 6을 의미합니다.
참고로 Lbound, Ubound 에서 L은 Lower, U는 Upper의 약자입니다.

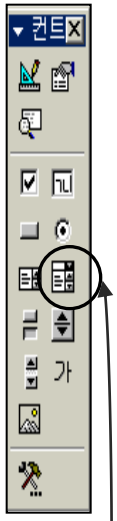
다중 배열(41.xls)

```
Sub ConditionalSum()

  Dim CSum As Variant
  Dim Start As Variant
  Dim Finish As Variant
  Dim TotalTime As Variant
  Dim i As Integer

  Start = Timer
  With Range("b3:e10243")

    For i = 1 To .Rows.Count
      If .Cells(i, 1) = "김현수" And .Cells(i, 2) = "수박" Then
        CSum = CSum + .Cells(i, 4)
      End If
    Next i

  End With
    Finish = Timer
    TotalTime = Finish - Start
MsgBox "김현수가 팔았던 수박의 매출합계는" _
        & Format(CSum, "##,##0") & "원 입니다. 걸린 시간은 " _
        & Format(TotalTime, "##,##0.00") & "초 입니다."
End Sub
```

```
Sub Array_4()

  Dim Rev As Variant
  Dim CSum As Variant
  Dim Start As Variant
  Dim Finish As Variant
  Dim TotalTime As Variant
  Dim i As Integer

  Start = Timer
  Rev = Range("b3:e10243")

    For i = 1 To UBound(Rev, 1)
      If Rev(i, 1) = "김현수" And Rev(i, 2) = "수박" Then
        CSum = CSum + Rev(i, 4)
      End If
    Next I

  Finish = Timer
  TotalTime = Finish - Start
  MsgBox "김현수가 팔았던 수박의 매출합계는" _
        & Format(CSum, "##,##0") & "원 입니다. 걸린 시간은 " _
        & Format(TotalTime, "##,##0.00") & "초 입니다."
End Sub
```

Ubound는 Ubound(배열변수, 차원) 의 형태로 사용되며 여기서는 rev 라는 배열 변수가 rev(열, 행) 의 형태로 사용되었으므로 1차원은 열의 개수, 2차원은 행의 개수를 의미합니다. 참고로 예에서 Ubound(rev,2) 라고 하면 rev 범위내 행의 개수인 4를 반환합니다.

차트그리기 설명



이것이 콤보박스 입니다.

이 콤보박스를 더블클릭하면 아래와 같이 매크로 Visual Basic Editor로 바로 이동합니다.

```
Private Sub ComboBox1_Change()
End Sub
```

그런데 지금까지와 다른 점은 1) 모듈이 아니라 "설명" 이라는 현재 시트에 VBA 코드를 입력할 수 있도록 된다는 점, 2) Sub 대신에 Private Sub 라는 프로시저가 호출된다는 점, 입니다.
VBA 코드는 VBE의 현재시트에 기록할 수도 있고, ThisWorkbook에 기록할 수도 있고, 모듈에 기록할 수도 있습니다. 다만 VBA가 작용되는 범위(Scope)에 차이가 있을 뿐입니다.

```
Sub Auto_Open()
Dim Region
Dim i As Integer
Region = Array("서울", "부산", "광주")
For i = LBound(Region) To UBound(Region)
  Worksheets("차트그리기").ComboBox1.AddItem Region(i)
Next i
End Sub
```

차트그리기 설명(42.xls)

```
Private Sub ComboBox1_Change()

  ActiveSheet.ChartObjects.Delete
                ' 이미 그려진 차트 오브젝트를 지우고
   With Charts.Add.Location(where:=xlLocationAsObject, Name:="차트그리기")
                 ' 차트그리기 시트에다 새로운 차트를 삽입
      .ChartType = xlColumnClustered
                 ' 차트종류는 세로막대형 (보기->개체찾아보기 )
      .SetSourceData Source:=Range("" & ComboBox1.Value & ""), PlotBy:=xlRows
                ' 차트의 원본데이터 범위는 콤보박스1의 값이 있는 범위로 하고, 열기준으로 그리되,
      .HasTitle = True
                ' 제목은 있게하고

      .ChartTitle.Text = ComboBox1.Value & "지역 월별 매출"
                ' 제목이름은 "콤보박스1의 값 & 지역 월별 매출"로 하고
     With .Parent
        .Top = Range("B8").Top
        .Left = Range("B8").Left
        .Height = 270
        .Width = 400
      End With
   End With
End Sub
```

Auto_Open()

Auto_Open 프로시져는 파일이 열리자마자 특정 코드가 실행되도록 할때 사용합니다. 본 파일을 열었을 때 인사말이 나타나도록 했는데 이것은 코드 모듈에 다음과 같은 구문을 넣으면 됩니다.

```
Sub Auto_Open()
MsgBox Application.UserName & "님! 안녕하세요?"
End Sub
```

Worksheet_Calculate(45.xls)

이 프로시져는 시트에서 계산할 경우 특정 코드가 실행되도록 할 때 사용하는 프로시져로서 반드시 해당 시트에 코드를 작성해야 합니다.

```
Sub Worksheet_Calculate()
    If Range("sales").Value < 600 Then
       MsgBox "매출 목표는 600 이상이어야 합니다"
    End If
End Sub
```

특정 행이나 열을 삭제하더라도 메시지가 나타나게 되는데,
이는 행, 열 삭제시 엑셀에서 재계산을 하도록 되어 있기 때문에,
위의 코드가 실행되는 것입니다.

Workbook_BeforePrint

이 프로시져는 인쇄를 하기 직전에 특정 코드가 실행되도록 할 때 사용하는 프로시져로서 주로 머리말이나 꼬리말 작성시에 사용됩니다.
예를들어 인쇄시 회사명과 파일 경로를 꼬리말에 항상 들어가도록 할 경우 아래 코드를 모듈에 추가하면 됩니다.

```
Sub Workbook_BeforePrint(Cancel As Boolean)
    For Each Worksheet In ThisWorkbook.Worksheets   ' 현재 파일 각 시트마다
       With Worksheet.PageSetup                      ' 프린터 설정
          .LeftFooter = "Hyundai Steel"          ' 좌측 꼬리말을 ' Hyundai Steel '로 설정
          .CenterFooter = ""                     ' 가운데 꼬리말은 공백("")
          .RightFooter = ThisWorkbook.FullName   ' 우측 꼬리말은 현재 파일경로를 입력하세요.
       End With
    Next Worksheet
End Sub
```

미리보기를 해 보시면, 꼬리말이 추가된 것을 확인하실 수 있을 것입니다.
위의 코드는 각 시트마다 실행되는 것으로써 Thisworkbook에 추가해야 합니다.
꼬리말 대신 머리말을 설정하고자 할 경우에는 Footer 대신 Header 라는 Object 를 사용하면 됩니다.

단순 피벗 매크로 기록(47.xls)

단순피벗은 가장 단순한 형태의 피벗 테이블입니다.
일단 피벗테이블을 매크로로 기록했을 경우의 코드부터 살펴보겠습니다.

```
Sub Macro1()
    ActiveWorkbook.PivotCaches.Add(SourceType:=xlDatabase, SourceData:= _
        "sheet1!R1C1:R5000C5").CreatePivotTable TableDestination:="", _
    TableName:=  "피벗 테이블1", DefaultVersion:=xlPivotTableVersion10
    ActiveSheet.PivotTableWizard TableDestination:=ActiveSheet.Cells(3, 1)
    ActiveSheet.Cells(3, 1).Select
    With ActiveSheet.PivotTables("피벗 테이블1").PivotFields("일자")
        .Orientation = xlPageField
        .Position = 1
    End With
    With ActiveSheet.PivotTables("피벗 테이블1").PivotFields("지점")
        .Orientation = xlColumnField
        .Position = 1
    End With
    With ActiveSheet.PivotTables("피벗 테이블1").PivotFields("품목")
        .Orientation = xlRowField
        .Position = 1
    End With
    ActiveSheet.PivotTables("피벗 테이블1").AddDataField
          ActiveSheet.PivotTables( _
        "피벗 테이블1").PivotFields("매출"), "합계:매출", xlSum
End Sub
```

매크로를 어떻게 기록하는가에 따라 코드는 약간씩 다르게 나타날 수 있습니다. 위의 경우 빈 피벗테이블을 먼저 만들어 놓고, 행과 열을 채우는 식으로 작성했는데, 피벗테이블 마법사로 만들 경우 약간 다르게 나타날 수도 있습니다. VBA로 피벗테이블을 만들면 약간 복잡해 보일 수도 있는데, 주요 개체(object)들만 살펴보면 다음과 같습니다. PivotCahes : 통합문서의 모든 피벗 테이블 캐시를 나타내는 컬렉션 개체를 반환
PivotTables : 워크시트에서 피벗테이블 보고서 컬렉션 개체를 반환
PivotFields : PivotTable 개체에서 필드 컬렉션 개체를 반환
PivotItems : 필드 범주 내에서 각각의 데이터 컬렉션 개체를 반환
CreatePivotTable : pivot cache의 데이터를 이용해서 피벗테이블을 만드는 PivotCache 개체의 method
PivotTableWizard : 피벗테이블 마법사 개체로 VBA에서는 거의 사용되지 않음

단순 피벗 코드 수정

```
Sub Pivot_1()
    Dim PCache As PivotCache
    Dim PTable As PivotTable
    Set PCache = ActiveWorkbook.PivotCaches.Add(SourceType:=xlDatabase, SourceData:= _
        Range("a1").CurrentRegion.Address)
    Set PTable = PCache.CreatePivotTable(TableDestination:="", TableName:="피벗테이블")
    With PTable
        .PivotFields("일자").Orientation = xlPageField
        .PivotFields("지점").Orientation = xlColumnField
        .PivotFields("품목").Orientation = xlRowField
        .PivotFields("매출").Orientation = xlDataField
    End With
End Sub
```

먼저 PCache와 PTable이라는 PivotCache 변수와 PivotTable이라는 변수를 선언합니다.
여기서 주의할 점은 둘다 Object 변수이므로 반드시 Set 구문 형태로 변수를 정의해야 한다는 것입니다. Add 메소드를 이용해서 PivotCache를 만듭니다. 이 경우 앞의 매크로 기록에서 절대참조로 기록된 것("Sheet1!R1C1:R5000C5")과는 달리 'Range("a1").CurrentRegion.Address'과 같은 식으로 A1 셀을 중심으로 입력된 셀범위 전체를 피벗테이블의 소스 데이터로 지정해 주면 됩니다. 그리고 나서 CreatePivotTable 메소드를 이용해서 '피벗테이블'이란 이름의 피벗테이블을 만듭니다. TableDestination은 지정해 줘도 되고 별도로 지정하지 않을 경우(TableDestination:="")에는 Default 값인 A1셀을 기준셀로 피벗테이블을 만듭니다. 그리고 피벗테이블에, 일자, 지점, 품목, 매출이라는 페이지 필드, 열 필드, 행 필드, 데이터 필드를 넣어 줍니다.

응용피벗 코드

이번에는 앞의 작업을 조금 더 발전시켜 수량과 매출을 피벗테이블에 나타내고,
매출에서 수량을 나눈 단가라는 필드를 하나 더 추가시키는 작업을 해 보도록 하겠습니다.
우선 아래의 Pivot_2라는 Sub 프로시져의 코드를 보시죠.

```
Sub Pivot_2()
    Dim PCache As PivotCache
    Dim PTable As PivotTable

    Application.DisplayAlerts = False
    On Error Resume Next
    Worksheets("피벗결과").Delete
    On Error GoTo 0

    Application.DisplayAlerts = True
    Set PCache = ActiveWorkbook.PivotCaches.Add(SourceType:=xlDatabase, SourceData:= _
       Range("a1").CurrentRegion.Address)
    Worksheets.Add.Name = "피벗결과"
    Set PTable = PCache.CreatePivotTable(TableDestination:=Worksheets("피벗결과") _
            .Range("a1"), TableName:="피벗테이블")
    With PTable
        .PivotFields("일자").Orientation = xlPageField
        .PivotFields("지점").Orientation = xlColumnField
        .PivotFields("품목").Orientation = xlRowField
        .PivotFields("매출").Orientation = xlDataField
        .PivotFields("개수").Orientation = xlDataField
        .CalculatedFields.Add "단가", "매출/개수"
        .PivotFields("단가").Orientation = xlDataField
        .PivotFields("합계 : 매출").Caption = "매출(원)"
        .PivotFields("합계 : 개수").Caption = "개수(개)"
        .PivotFields("합계 : 단가").Caption = "단가(원)"
    End With
End Sub
```

단가=매출/개수라는 수식은 'CalculateFields.Add "단가","매출/개수"' 라고 표현합니다.

변수 선언 후 아래 구문은 '피벗결과'라는 시트가 이미 존재할 경우 '다음 시트를 삭제합니다' 라는 경고메시지 없이(Application.DisplayAlerts = False) 삭제하도록 하는 구문인데, 혹시 시트가 없을 경우 에러를 일으키지 않도록 하기 위해 'On Error Resume Next' 구문과 'On Error GoTo 0'라는 환원 구문을 넣어준 것입니다.

Multi Pivot(50.xls)

매우불만, 2: 불만, 3: 보통, 4: 만족, 5: 매우만족'으로 가정했을 경우, 피벗 테이블을 작성하는 VBA 구문은 다음과 같습니다.

```
Sub Multi_Pivot()

    Dim PCache As PivotCache
    Dim PTable As PivotTable
    Dim Question As String
    Dim i As Integer
    Dim j As Integer

    Application.DisplayAlerts = False
    On Error Resume Next
    Worksheets("피벗결과").Delete
    On Error GoTo 0
    Application.DisplayAlerts = True
    Set PCache =
          ActiveWorkbook.PivotCaches.Add(SourceType:=xlDatabase, _
          SourceData:=
          Worksheets(“sheet1").Range("a1").CurrentRegion.Address)
    Worksheets.Add.Name = "피벗결과"
    i = 1
    For j = 1 To 4
        Question = Worksheets(" sheet1").Cells(1, j + 2)
        Set PTable = PCache.CreatePivotTable _
            (TableDestination:=Worksheets("피벗결과").Cells(i, 1), _
             TableName:=Question)
    i = i + 10
    With PTable.PivotFields(Question)
          .Orientation = xlDataField
          .Name = "빈도"
    End With
    With PTable.PivotFields(Question)
          .Orientation = xlDataField
          .Name = "비율"
          .Calculation = xlPercentOfTotal
    End With
    With PTable
        .AddFields RowFields:=Array(Question, "data")
        .PivotFields("성별").Orientation = xlColumnField
        .PivotFields("data").Orientation = xlColumnField
    End With
    Next j
    With Worksheets("피벗결과").Range("A:A")
       .Replace "1", "매우 불만"
       .Replace "2", "불만"
       .Replace "3", "보통"
       .Replace "4", "만족"
       .Replace "5", "매우 만족"
       .Parent.Range("A:G").EntireColumn.AutoFit
    End With
End Sub
```

VBA 를 이용해서 파일 목록 출력

VBA 에서 파일에 접근하는 방법은 크게 세가지가 있습니다.

1. 전통적인 (Traditional) VBA 함수를 이용하는 방법으로 이 방법은 Excel의 모든 버전에서 작동합니다.
2. FileSearch Object 를 이용한 방법으로 Excel 97 이상의 버전에서 작동합니다.
3. FileSystemObject 라는 Microsoft Scripting Library를 이용한 방법으로 엑셀 2000 이상의 버전에서 작동합니다.

이번 강좌에서는 위의 각각의 방법들을 이용해서 파일 또는 드라이브에 접근하는 방법들을 간단히 소개할까 합니다.

전통적인 방법으로 파일목록 출력 (52.xls)

```
Sub ExtractFiles_1()

    Dim MyDir As String, MyFile As String
    Dim i As Integer
    MyDir = "C:\Windows\ "

    MyFile = Dir(MyDir & "*.bmp")
    i = 1
    Do While MyFile <> ""
       With ActiveSheet.Range("G11")
          .Offset(i, 0) = MyFile
          .Offset(i, 1) = FileLen(MyDir & MyFile)
          .Offset(i, 2) = FileDateTime(MyDir & MyFile)
       End With
       MyFile = Dir
       i = i + 1
    Loop
End Sub
```

FileSearch Object를 이용

```
Sub ExtractFiles_2()

    Dim MyDir As String, MyFile As String
    Dim i As Integer
    Dim fs As Object

    MyDir = "C:\Windows\"
    MyFile = "*.bmp"
    Set fs = Application.FileSearch

    With fs
      .NewSearch
      .LookIn = MyDir
      .Filename = MyFile
      .SearchSubFolders = False
      .Execute

      For i = 1 To .FoundFiles.Count
        With ActiveSheet.Range("G11")
          .Offset(i, 0) = fs.FoundFiles(i)
          .Offset(i, 1) = FileLen(fs.FoundFiles(i))
          .Offset(i, 2) = FileDateTime(fs.FoundFiles(i))
        End With
      Next i
    End With

End Sub
```

FileSystemObject Object

이번에는 PC의 드라이브 정보를 추출하는 방법을 설명드리겠습니다.
Excel 2000 부터는 FileSystemObject 라는 Object가 도입되었는데,
이를 이용해서 PC의 드라이브에 관한 정보를 추출할 수 있습니다.
앞서 설명한 것처럼 이 프로시져는 Excel 2000 이상의 버전에서만 작동합니다.

```
Sub ExtractDriveInfo()

    Dim FS, D
    Dim i As Integer

    Set FS = CreateObject("Scripting.FileSystemObject")
    i = 13

    On Error Resume Next

    For Each D In FS.Drives
        i = i + 1

        Cells(i, 1) = D.DriveLetter

        If D.IsReady Then
            Cells(i, 2) = "들어있음"
        Else
            Cells(i, 2) = "없음"
        End If

        Select Case D.DriveType
            Case 0: Cells(i, 3) = "알수없음"
            Case 1: Cells(i, 3) = "이동식"
            Case 2: Cells(i, 3) = "하드 드라이브"
            Case 3: Cells(i, 3) = "네트워크"
            Case 4: Cells(i, 3) = "CD-ROM"
            Case 5: Cells(i, 3) = "RAM 디스크"
        End Select

        Cells(i, 4) = D.VolumeName
        Cells(i, 5) = D.TotalSize
        Cells(i, 6) = D.AvailableSpace

    Next D

    On Error GoTo 0

End Sub
```

FileSystemObject Object

위 코드에서 가장 중요한 구문은

```
Set FS = CreateObject("Scripting.FileSystemObject")
```

입니다. Scripting.FileSystemObject 구문을 이용해서 PC의 파일시스템에
접근할 수가 있는 것입니다.
전체 구문은 For Each - Next 구문으로 되어있고, 파일 시스템 중 드라이브에
다음 구문들로 접근할 수 있도록 만들었습니다.

```
For Each D In FS.Drives
 D.DriveLetter      ' 드라이브의 문자, a 드라이브, c 드라이브 등
 D.Isready          ' a 드라이브, CD-Rom 드라이브 등에 디스켓이나 CD 가 있으면 True, 없으면 False
 D.DriveType        ' 각 타입에 따라 0-5까지의 숫자를 반환하는데 위 구문에서는 Select Case 구문으로 처리
 D.VlumeName    ' 드라이브 이름
 D.TotalSize        ' 드라이브 용량
 D.AvailableSpace  ' 드라이브의 사용가능 용량
Next D
```

Application Object에 바로 오는 Property

아래 VBA 구문들의 차이는 무엇일까요?

Range("a1").Select
Activesheet.Range("a1").Select
Application.Activesheet.Range("a1").Select

모두 A1 셀을 선택하라는 의미입니다. 두번째 구문은 현재시트의 A1 셀을 선택하라는 의미이고, 세번째 구문은 엑셀이라는 애플리케이션에서 현재시트의 A1 셀을 선택하라는 의미입니다.
여기서 Application이나 Activesheet, Range("a1")을 Object 라고 합니다.
그리고 Select 라는 것은 Method 라고 합니다. 즉, VBA 구문은
[Object.][Object.]Object.Method
와 같은 구조로 성립될 수 있습니다. 영어에서 주어와 동사가 있듯이 VBA 에서는 Object와 Method 가 있고, Object는 여러개 나올 수 있다는 것을 알 수 있습니다. Object 라는 것을 말 그대로 어떤 대상 또는 객체를 가리킵니다. 그리고 이런 Object 는 계층구조가 있습니다. Range의 상위 Object는 Activesheet 이고, VBA 에서 최상위 Object 는 Application 입니다.
한가지 예를 더 들면,

Application.Activesheet.Range("a1").ColumnWidth = 16

이라고 했을 경우 ColumnWidth는 Property(속성) 입니다.
이처름 Object 뒤에 Method 또는 Property 가 붙어서 하나의 VBA 구문을 형성합니다.
그런데 일부 Property는 Application Object 뒤에 바로 오는 경우가 있습니다.
예들들어

Application.ScreenUpdating = False

와 같은 구문이 있는데 이 구문을

Application.Activesheet.ScreenUpdating = False
ScreenUpdating = False

등과 같은 식으로 사용하면 잘못된 구문으로 인식되거나 이상한 결과를 얻을 수 있습니다.
대표적인 것으로 ScreenUpdating, DisplayAlerts, StatusBar 등이 있는데, 이런 Property들은 반드시 Application Object를 앞에 붙여서 사용해야만 제대로 작동합니다.

ScreenUpdating Property

한칸한칸 화면을 갱신하는 것이 보임

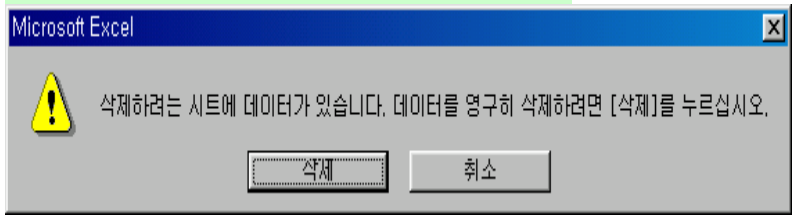
```
Sub Sample_1()

    For i = 1 To 10
      For j = 1 To 10

       k = (i - 1) * 10 + j

       Range("a9").Offset(i, j).Activate
       ActiveCell.Value = k

       If k Mod 3 = 0 Then
       ActiveCell.Interior.ColorIndex = 36
       End If

      Next j
   Next i

       Range("b10").Select

End Sub
```

화면 전체를 한번에 갱신함

```
Sub Sample_2()

   Application.ScreenUpdating = False

   For i = 1 To 10
    For j = 1 To 10
     k = (i - 1) * 10 + j
     Range("a28").Offset(i, j).Activate
     ActiveCell.Value = k
     If k Mod 3 = 0 Then
     ActiveCell.Interior.ColorIndex = 36
     End If
     Next j
   Next i
   Range("b10").Select

   Application.ScreenUpdating = True
End Sub
```

Displayalerts Property

```
Sub Sample_3()

    Worksheets("예제시트1").Delete

End Sub
```



그런데 문제가 있지요?
아마도 다음과 같은 경고메시지가 뜰 것입니다.
여기서 삭제 버튼을 눌러줘야 시트가 삭제됩니다.
매크로 기록으로 코드를 작성한 것을 그대로 매크로로 실행시키더라도 위의 경고 메시지는 없어지지 않습니다.

이런 경고메시지를 무시하고 항상 Default 값을 선택해 주도록하는 속성이 바로 DisplayAlerts 입니다.
위의 코드를 약간 수정해서 예제시트2를 삭제해 보도록 하겠습니다.
앞의 코드와는 달리 아래와같이 바꾸어주면 예제시트2가 순식간에 없어집니다.
코드는 다음과 같습니다.

```
Sub Sample_4()

    Application.DisplayAlerts = False
    Sheets("예제시트2").Delete
    Application.DisplayAlerts = True

 End Sub
```

Application.DisplayAlerts = False 를 코드 앞에 선언해 주면 됩니다.
앞의 ScreenUpdating 에서 처럼 코드이 마지막 부분은 True 값으로 바꾸어 줍니다.
Default 값은 True 입니다.

StatusBar Property

```
Sub Sample_5()

Dim i As Integer
Dim j As Integer

' 1단계 작업 부분입니다.
For i = 1 To 5000
Range("B18").Value = i
Application.StatusBar = "현재 1단계 작업이" & Format(i / 5000, "0.0%") & " 진행중입니다."
Next i

' 2단계 작업 부분입니다.
For j = 1 To 3000
ActiveSheet.Range("C18").Value = j
Application.StatusBar = "현재 2단계 작업이" & Format(j / 3000, "0.0%") & " 진행중입니다."
Next j

' 작업 완료라고 표시되는 부분입니다.
Application.StatusBar = "작업완료 !"

' 작업 완료라고 표시되는 상태에서 3초간 멈춰있다가, 원상태로 복귀시켜 줍니다.
If Application.Wait(Now + TimeValue("0:00:03")) Then
Application.StatusBar = False
End If

End Sub
```

SendKeys

엑셀에서 값을 복사해서 메모장에 붙여넣기를 하고, Test.txt 란 이름으로 저장하는 작업을 VBA로 할 수 있을까요?
일반적으로 워드, 액세스, 엑셀, 파워포인트, 아웃룩 등은 모두 VBA 기능을 가지고 있어서 서로 호환이 가능합니다만, 다른 프로그램인 메모장, 계산기 등을 엑셀 VBA로 실행시켜 특정 작업을 하도록 할 수 있을까요?

```
Sub Sample_6()
' shell 함수를 사용해서 메모장을 실행시킵니다.
Dim x
x = Shell("C:\windows\notepad.exe", vbNormalFocus)
AppActivate x


'Temp 영역 (intro 시트의 B3:B6)을 복사합니다.
Range("Temp").Copy


'SendKeys 속성을 이용해서 붙여넣기(Ctrl + V)를 합니다.
Application.SendKeys "^V", True

'SendKeys 속성을 이용해서 다른이름으로 저장 (Alt + F, A)을 택한후 Test 라고 입력하고 엔터키(~)를 칩니다.
Application.SendKeys "%FATest~", True
End Sub
```

키	코드
백스페이스	{BACKSPACE} 또는 {BS}
아래쪽 화살표	{DOWN}
오른쪽 화살표	{RIGHT}
왼쪽 화살표	{LEFT}
위쪽 화살표	{UP}
Break	{BREAK}
Caps Lock	{CAPSLOCK}
Clear	{CLEAR}
Delete 또는 Del	{DELETE} 또는 {DEL}
End	{END}
Enter	~ (물결표)
Enter(숫자 키패드)	{ENTER}
Esc	{ESCAPE} 또는 {ESC}
F1부터 F15까지	{F1}부터 {F15}까지
Help	{HELP}
Home	{HOME}
Insert	{INSERT}
Num Lock	{NUMLOCK}
Page Down	{PGDN}
Page Up	{PGUP}
Return	{RETURN}
Scroll Lock	{SCROLLLOCK}
Tab	{TAB}
Shift	+ (더하기 기호)
Ctrl	^ (캐럿)
Alt	% (백분율 기호)

OnKey Method

```
Sub Sample_7()
Application.OnKey "{PGDN}", "MyMsg"
End Sub

Sub MyMsg()
MsgBox "지금은 " & Now & "입니다."
End Sub
```

OnKey Method에 대한 한가지 예만 더 들어 보기로 하겠습니다.
엑셀에서 아주 많이 사용하는 단축키 중 하나가 Ctrl + C 일 것입니다.
"복사하라"는 의미죠.
그런데, OnKey Method를 사용해서 Ctrl + C 를 눌렀을 때 다른 프로시져를 수행하도록 할 수 있습니다. 이 경우 다음과 같이 코드를 작성하면 됩니다.

Application.OnKey "^c", "특정프로시져"

이렇게 할 경우 Ctrl + C 를 눌렀을 때, '복사하기'라는 본래 기능을 없어지고 특정 프로시져가 실행됩니다. Ctrl 키를 ^ 로 표시하는 것은 앞의 SenKeys 속성 설명에서의 도움말과 같습니다.
그리고, 본래 기능을 되돌리고 싶다면 다음과 같이 코드를 작성해서 실행시키면 됩니다.

Application.OnKey "^c", "특정프로시져"

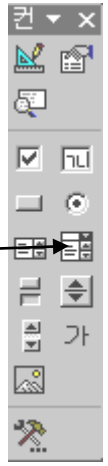
이 경우 주의할 것은 c를 반드시 소문자로 해야 합니다. 대문자 C 로 할 경우 Ctrl + Shift + C로 인식됩니다.
OnKey Method 와 관련해서 한가지만 더 설명드린다면, 아래 코드를 사용해서 엑셀 본래의 단축키 기능을 없앨 수도 있습니다.
즉, Ctrl + C를 눌렀을 경우 '복사하기'가 되는데, 복사하기 실행을 하지 못하도록 막을 수도 있습니다. 앞의 예에서는 특정 프로시져를 수행하도록 했는데, 그 자리에 아무 것도 입력하지 않고 따옴표만 입력해 놓으면,
엑셀의 Ctrl + C 기능을 마비시킬 수 있습니다.
다음과 같이 말이죠~.

Application.OnKey "^c", “”

콤보박스 만들기

도구메뉴에서 사용자지정을 택하고 Visual Basic을 택하면 아래와 같은 Visual Basic 도구상자가 표시됩니다.

여기서 이 부분을 누르면 아래와 같이 컨트롤 도구상자가 표시됩니다.



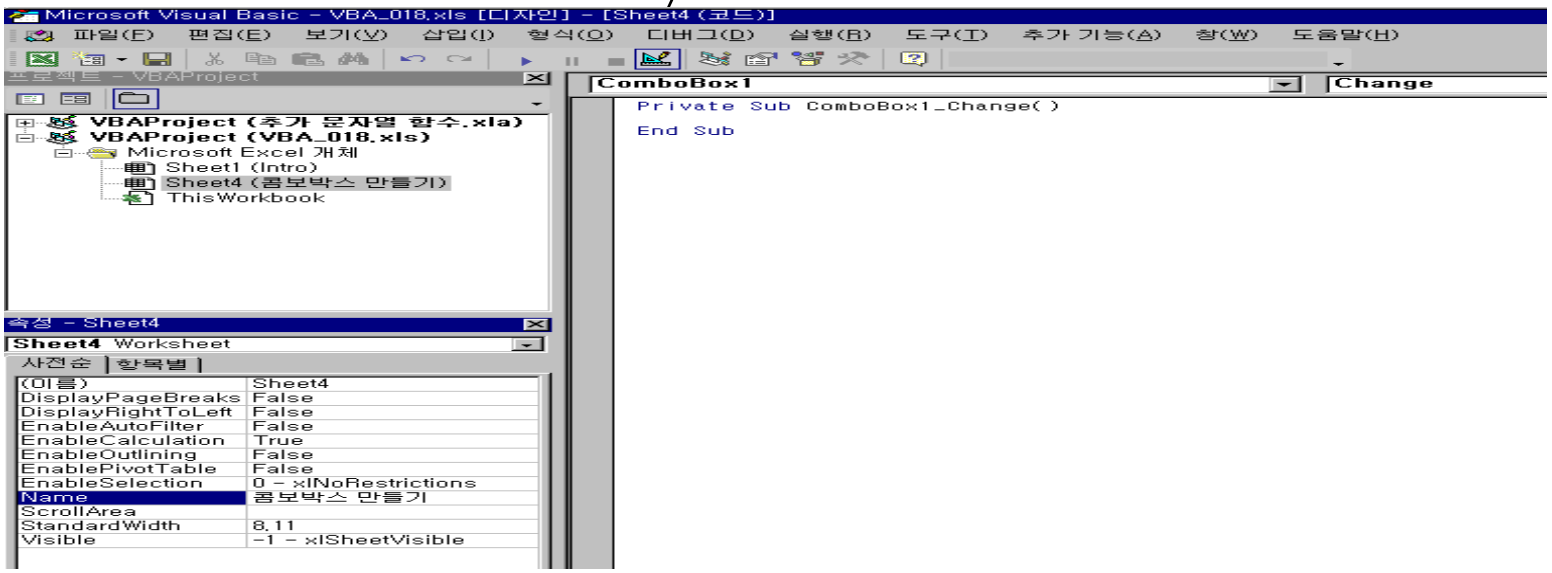
컨트롤 대화상자에서 이 부분을 누르고 적당한 크기로 박스를 그리면
아래와 같이 콤보박스가 그려지면서 Visual Basic 도구상자의 디자인 모드가
활성화 되면서 Visual Basic 디자인 모드가 됩니다.

이 부분이 활성화되면서 디자인모드가 됩니다.

이 상태에서 콤보상자를 두번 누르면 Visual Basic Editor가 열리면서 다음과 같이 코드가 자동생성됩니다.

콤보박스 만들기

그리고 Visual Basic Editor의 도구모음에도 디자인모드가 활성화되는데, 여기서 코드를 다 작성한 후에 버튼을 눌러 디자인모드를 끝내고 VBE를 닫고 엑셀파일로 돌아오면 Visual Basic 컨트롤 도구상자의 디자인 모드도 끝난 상태로 연동이 되면서 다음 시트에서 설명드릴 콤보상자에 값을 넣은 후, 값을 선택하면 바로 ComboBox1_Change 라는 매크로가 실행되게 됩니다.



이처럼 컨트롤 도구상자에서 콤보박스를 만들어 더블클릭하면 자동으로 기본 VB 코드가 작성되는데, 다른 코드와 다른 점은 모듈에 생성되는 것이 아니라 시트에 코드가 생성된다는 점입니다.
그리고 콤보박스를 하나 더 만들면 이름이 ComboBox2, ComboBox3와 같은 식으로 일련번호가 메겨집니다.
이 콤보 박스에 목록을 입력하는 방법은 다음 시트에서 설명드리도록 하겠습니다.

콤보박스에 목록 채우기(64.xls)

```
Private Sub Workbook_Open()

Dim Region
Dim i As Integer

Region = Array("강동", "강서", "강남", "강북")

For i = LBound(Region) To UBound(Region)
  Sheet5.ComboBox1.AddItem Region(i)
Next i

End Sub
```

Lbound 함수와 Ubound 함수는 앞의 VBA에서 배열 이용하기 강좌에서 설명드린 VBA 내장함수입니다.
위의 코드 For i = LBound(Region) To UBound(Region) 대신 For i = 0 to 3으로 바꿔도 됩니다.

참고로 위의 코드에서 빨간 박스에 들어있는 부분은 Additem Method 대신 List Property로 처리해도 됩니다.
이 경우 배열은 횡적으로 숫자가 나열되어 있고, 콤보상자는 세로 형태의 배열이므로 Transpose 워크시트 배열함수를 이용해서 뿌려주면 됩니다.

```
Sheet5.ComboBox1.List = WorksheetFunction.Transpose(Region)
```

이렇게 처리하는 것이 코딩상으로는 훨씬 간결합니다.

콤보박스에 매크로 연결하기 (64_1.xls)

이번에는 이들 목록중 하나는 택했을 때 어떤 작업을 하도록 하는 매크로를 짜 보도록 하겠습니다.
방법은 ListIndex 속성을 이용하는 방법과 Value 속성을 이용하는 방법 두가지가 있습니다. 아래 콤보상자를 눌러 보시죠.

먼저 Value 속성을 사용할 경우 코드를 보시면 다음과 같습니다.
참고로 코드는 모듈이 아니라 현재시트의 코드에 작성해야 합니다.

```
Private Sub ComboBox1_Change()

Range("D13").Value = ComboBox1.Value

End Sub
```

그리고 ListIndex 속성을 이용할 수도 있는데, 이 경우의 코드는 다음과 같습니다.
ListIndex는 0부터 시작하므로 ListIndex + 1로 설정해줘야 강동, 강서, 강남, 강북이
각각 1, 2. 3, 4, 로 표시됩니다.

```
Private Sub ComboBox1_Change()

Range("D15").Value = ComboBox1.ListIndex + 1

End Sub
```

이런식으로 해서 콤보박스에 VBA를 이용해서 값을 담고,
또 워크시트에 값 또는 인덱스를 표시할 수가 있습니다.

콤보박스 작업예(64_1.xls)

```
Private Sub Workbook_Open()

Dim Year
Dim Month
Dim Unit

Year = Array("1999", "2000", "2001")
Month = Array("1월", "2월", "3월", "4월", "5월", "6월", "7월",
        "8월", "9월", "10월", "11월", "12월")
Unit = Array("개수", "매출")

  With Sheet2
    .ComboBox1.List = WorksheetFunction.Transpose(Year)
    .ComboBox2.List = WorksheetFunction.Transpose(Month)
    .ComboBox3.List = WorksheetFunction.Transpose(Unit)
  End With

End Sub
```

```
Sub MySub()

Dim MyYear
Dim MyMonth
Dim MyUnit
Dim k As Range

With Sheet2
  MyYear = .ComboBox1.Value
  MyMonth = .ComboBox2.ListIndex + 1
  MyUnit = .ComboBox3.Value
End With

For Each k In Sheet2.Range("C11:F13")
   k.FormulaArray = "=SUM((년도=" & MyYear &
        ")*(월=" & MyMonth & ")*(상품=RC2)*(지
        역=R10C)*" & MyUnit & " )"
Next k

End Sub
```

```
Private Sub ComboBox1_Change()
Call MySub
End Sub
```

```
Private Sub ComboBox2_Change()
Call MySub
End Sub
```

```
Private Sub ComboBox3_Change()
Call MySub
End Sub
```

설정및 코드설명

셀이름정의

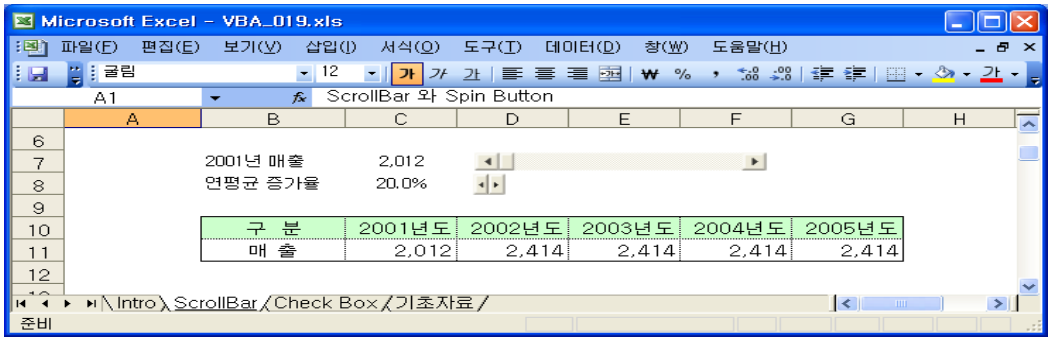
년도 : = sheet3!$A$2:$A$433
월 : = sheet3!$B$2:$B$433
지역 : = sheet3!$C$2:$C$433
상품 : = sheet3!$D$2:$D$433
개수 : = sheet3!$E$2:$E$433
매출 : = sheet3!$F$2:$F$433

위 코드에서 어려운 부분은 배열함수로 입력하는 부분입니다.
코드가 어려울 경우 일단 매코로 기록으로 기록을 한 후 매크로기록기에 자동으로 기록된 코드를 약간만 수정해 주면 됩니다.

우선 첫번째 셀인 C19 셀에 2001년, 3월, 개수를 배열식으로 입력하는 작업을 매크로로 기록한 후 수식을 보면 다음과 같이 나타나는데,
FormulaArray = "=SUM((년도=2001)*(월=3)*(상품=RC2)*(지역=R18C)*개수)“

여기서 빨간색 부분이 모두 ComboBox1,2,3의 값, 즉 MyYear, MyMonth, MyUnit로 대치시켜야 합니다. 이 경우 문자열기호 " 와 & 기호를 적절히 사용하여 다음과 같은 식으로 입력해 줘야 합니다. (빨간색으로 되어 있는 부분이 바뀐 부분입니다.)
FormulaArray = "=SUM((년도=" & MyYear & ")*(월=" & MyMonth & ")*(상품=RC2)*(지역=R18C)*" & MyUnit & ")"

ScrollBar 와 Spin Button(68.xls)



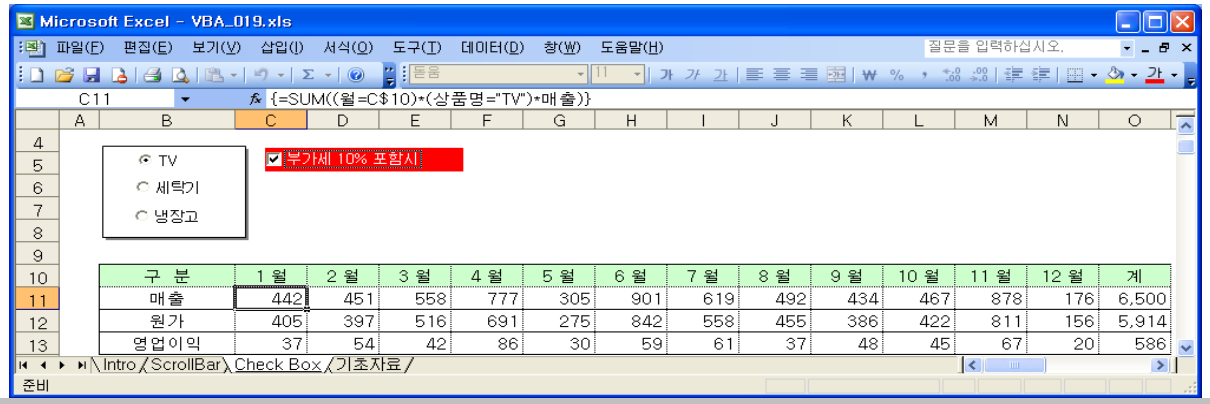

```
Private Sub ScrollBar1_Change()

ScrollBar1.Min = 2000
ScrollBar1.Max = 4000
Range("c7").Value = ScrollBar1.Value

End Sub
```

Scroll 이벤트를 사용해도 되는데, 이 경우 차이점은 스크롤하고 있을때 값이 변한다는 점입니다. 즉, Change 이벤트의 경우는 스크롤 바에서 네모난 부분을 마우스로 끌어다가 놓는 순간에 값이 "짠" 하고 바뀌지만, Change 대신 Scroll 이벤트를 사용하면, 네모를 마우스로 끌고있는 순간에 계속 값이 바뀌게 됩니다.

```
Private Sub SpinButton1_SpinDown()

With Range("c8")
    .Value = WorksheetFunction.Max(0, .Value - 0.05)
End With

End Sub

Private Sub SpinButton1_SpinUp()

With Range("c8")
    .Value = WorksheetFunction.Min(1, .Value + 0.05)
End With

End Sub
```

SpinDown은 말 그대로 왼쪽 화살표를 눌렀을 경우 감소하는 값으로 최대값(Max)과 음의 변화한도값(-0.05)를 지정해 줍니다.
SpinUp은 반대로 오른쪽 증가버튼을 눌렀을 경우이므로 최소값과 양의 변화한도값을 지정해 주면 되겠죠.

Check Box 와 Option Button(69.xls)




```
Private Sub CheckBox1_Click()

If CheckBox1.Value Then
   Range("c15").Value = 0.1
Else
   Range("c15").Value = 0
End If

End Sub
```

```
Private Sub Options()

  Select Case True
       Case OptionButton1.Value
            ProductName = "TV"
       Case OptionButton2.Value
            ProductName = "세탁기"
       Case OptionButton3.Value
            ProductName = "냉장고"
  End Select

  Application.ScreenUpdating = False
     Range("C11").FormulaArray = "=SUM((월=R10C)*(상품명=""" & ProductName & """)*매출)"
     Range("C12").FormulaArray = "=SUM((월=R10C)*(상품명=""" & ProductName & """)*원가)*(1+R15C3)"
     Range("C11:C12").Copy                    '복사해서 붙여 넣음 (속도를 위해)
     Range("D11:N12").PasteSpecial Paste:=xlPasteFormulas
     Range("C11").Select
  Application.ScreenUpdating = True

End Sub
```

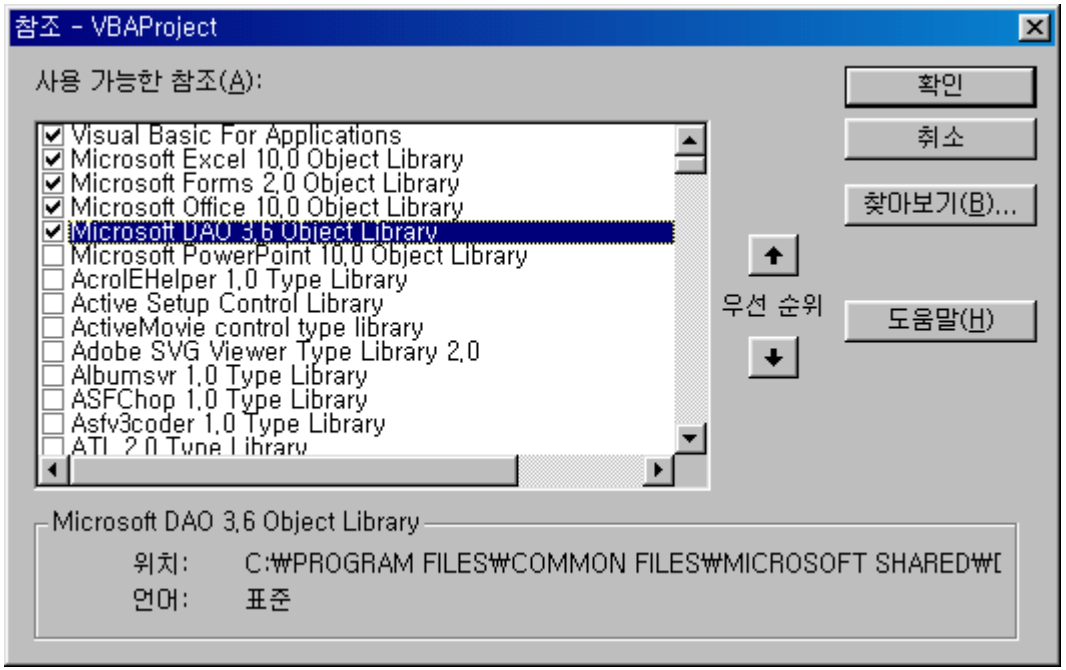
```
Private Sub OptionButton1_Click()
  Call Options
End Sub
Private Sub OptionButton2_Click()
  Call Options
End Sub
Private Sub OptionButton3_Click()
  Call Options
End Sub
```

설정및 코드설명

배열식을
Range("C11").FormulaArray = "=SUM((월=R10C)*(상품명=""" & ProductName & """)*매출)"
와 같은 식으로 한 이유는 기초자료 시트에 월, 상품명, 매출, 원가 등의 이름으로 미리 이름을
정의해 놓았기 때문에 가능합니다.
즉, 기초자료 시트에
월 =기초자료!A2:A37
상품명 =기초자료!B2:B37
매출 =기초자료!C2:C37
원가 =기초자료!D2:D37

아래 배열식에서 주의할 점은
Range("C11").FormulaArray = "=SUM((월=R10C)*(상품명=""" & ProductName & """)*매출)"
빨간 색 박스안에 있는 부분입니다.
수식에서 문자열은 따옴표(" ") 로 처리되는데 이를 VBA에서 표현해 주려고 할때 위와 같이 표현해 주면 됩니다.

텍스트 파일을 Database 형태로 읽어들여 Access 파일에 입력



DAO(Data Access Object)를 이용한 기법이므로 VBE(Visual Basic Editor)의 도구 > 참조 메뉴에서 아래 그림처럼 microsoft DAO 3.6 Object Library에 참조를 걸어주어야 합니다.

아무튼 이번 강좌는 텍스트 파일을 database 형태로 가공해서 엑셀에서 읽어들이고, 이를 엑세스 파일에 Export 시키는 작업을 VBA로 구현하는 것입니다.

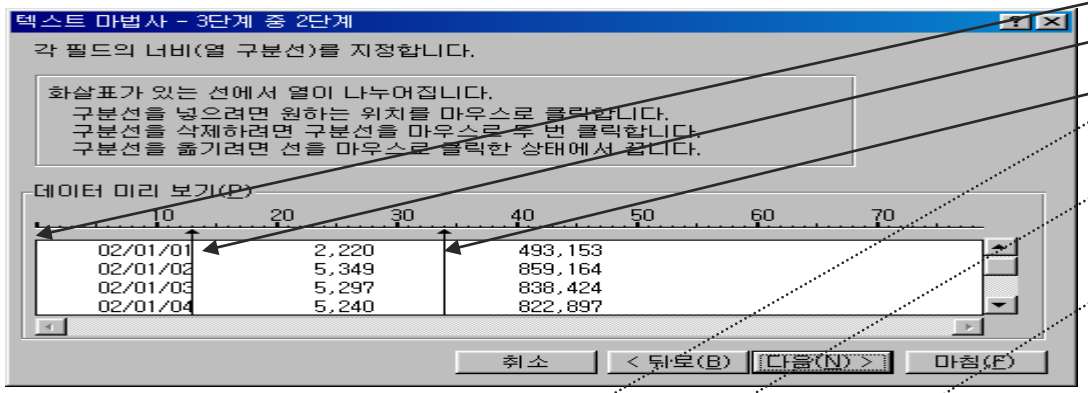
ExtractData(72.xls)

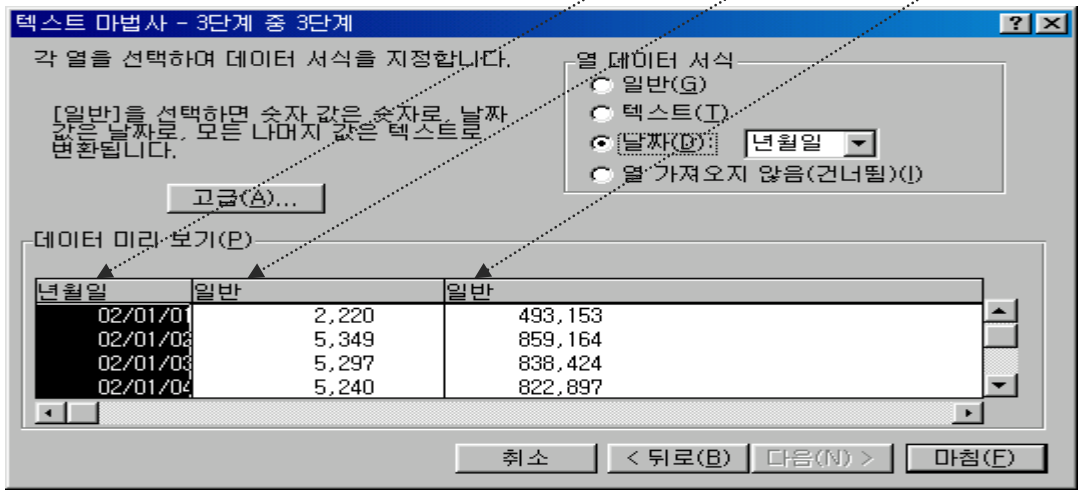
Data 시트를 하나 만들고 Text.txt 파일을 읽어들이는 ImportData 에 대한 코드입니다.

```
Sub ImportData()
' 시트를 하나 삽입하고, 시트이름은 data 로 하고,  첫번째 열에 일자, 수량, 매출 등의 필드명을 입력합니다.
 With Worksheets.Add
      .Name = "data"
      .Cells(1, 1).Value = "일자"
      .Cells(1, 2).Value = "수량"
      .Cells(1, 3).Value = "매출"
  End With
' 현재 폴더와 같은 경로에 있는 text.txt 파일을 '너비가 일정함' 옵션으로 열되, 각각 13, 34 번째 칸을 기준으로 텍스트나누기를 하는데, 첫번째 텍스트는 숫자 서식으로
      합니다.
  Workbooks.OpenText Filename:=ThisWorkbook.Path & "\72.TXT", DataType:=xlFixedWidth, FieldInfo:=Array(Array(0, 5), Array(13, 1), Array(34, 1))
' Do ---- Loop Until 구문 (현재셀의 첫번째 문자열 값이 "총" 이라는 문자가 나올때까지 반복)
  Do
' 현재셀의 맨 왼쪽 문자가 숫자일 경우만 남기고 나머지 경우에는 전체 행을 삭제합니다.
  If IsNumeric(Left(ActiveCell, 1)) = True Then
    ActiveCell.Offset(1, 0).Activate
  Else
    ActiveCell.EntireRow.Delete
  End If

  Loop Until Left(ActiveCell, 1) = "총"
' 총 이라는 문자열이 있는 전체 행을 지우고난 후 a1 셀을 기준으로 인접 블록을 모두 복사합니다.
  ActiveCell.EntireRow.ClearContents
  ActiveSheet.Range("a1").CurrentRegion.Copy

' 다음 윈도우 (즉 텍스트 파일이 아닌 본 강좌파일의 data 시트) 의 A2 셀에 값만 붙이고 다시 텍스트 파일로 돌아갑니다.
  ActiveWindow.ActivateNext
  Range("A2").PasteSpecial Paste:=xlValues
  ActiveWindow.ActivateNext
' 텍스트 파일을 닫습니다. 이 경우 변경사항을 저장하시겠습니까? 라고 묻는 경고메시지는 무시하고 (즉 변경내용을 저장하지 않고) 닫습니다.
  Application.DisplayAlerts = False
  ActiveWorkbook.Close
  Application.DisplayAlerts = True
' data 시트의 첫번째 행을 yy-MM-DD 의 숫자형식으로 하고, DataInput 시트를 활성화시킵니다.
  Columns("A:A").NumberFormatLocal = "yy""-""m""-""d"
   Worksheets("sheet1").Activate

End Sub
```

ExtractData

```
Workbooks.OpenText Filename:=ThisWorkbook.Path & "\Text.TXT", DataType:=xlFixedWidth, FieldInfo:=Array(Array(0, 5), Array(13, 1), Array(34, 1))
```





이 구문에서 각 배열은 Array(a,b) 와 같은 형태로 되어 있는데, a 값은 문자열을 나누는 폭을 뜻하고,
b 값은 5일경우 날짜형식, 1일 경우 일반 형식, 9일 경우 생략을 뜻합니다.

DeleteData(72.xls)

DeleteData 프로시져는 database.mdb라는 엑세스 파일에서 data 테이블에 있는 모든 레코드 값을 삭제하는] 작업입니다.

```
Sub DeleteData()
' Database와 Recordset 에 대한 변수를 선언합니다.
Dim DB As Database
Dim RS As Recordset
' database.mdb 라는 엑세스 파일에서 Delete * From data 라는 SQL 구문을 실행한 후 Database를 닫습니다.
Set DB = OpenDatabase(ThisWorkbook.Path & "\72.mdb")
DB.Execute "Delete * From data"
DB.Close

End Sub
```

ToAccess(72.xls)

ToAccess 프로시져는 엑셀 data 시트의 모든 내용을 같은 경로에 있는 Access 파일의 data 테이블에 입력하는 프로시져입니다.

```
Sub ToAccess()
' Database와 SQL 구문으로 사용될 strSQL 이라는 변수를 선언합니다.
Dim DB As Database
Dim strSQL As String
' OpenDatabase 구문을 이용해서 현재 폴더와 같은 경로에 에 있는 Excel 8.0 형식의 Database를 열고,
  Set DB = OpenDatabase(ThisWorkbook.FullName, False, False, "Excel 8.0;")
' 엑셀의 data 시트를 몽땅 택해서 같은 경로에 있는 database.mdb 파일의 data 테이블에 입력하는 SQL 구문을 실행합니다.
  strSQL = "Insert Into data In '" & ThisWorkbook.Path & "\72.mdb' Select * From [data$]"
  DB.Execute strSQL
' 현재 DB를 닫고,
  DB.Close
' DB를 비웁니다.
  Set DB = Nothing
End Sub
```

DeleteSheet(72.xls)

DeleteSheet 프로시져는 첫번째 프로시져인 ImportData 프로시져로 생성된 data 라는 이름을 가진 시트를 삭제하는 작업입니다.

```
Sub DeleteSheet()
Application.DisplayAlerts = False
Worksheets("data").Delete
Application.DisplayAlerts = True
End Sub
```

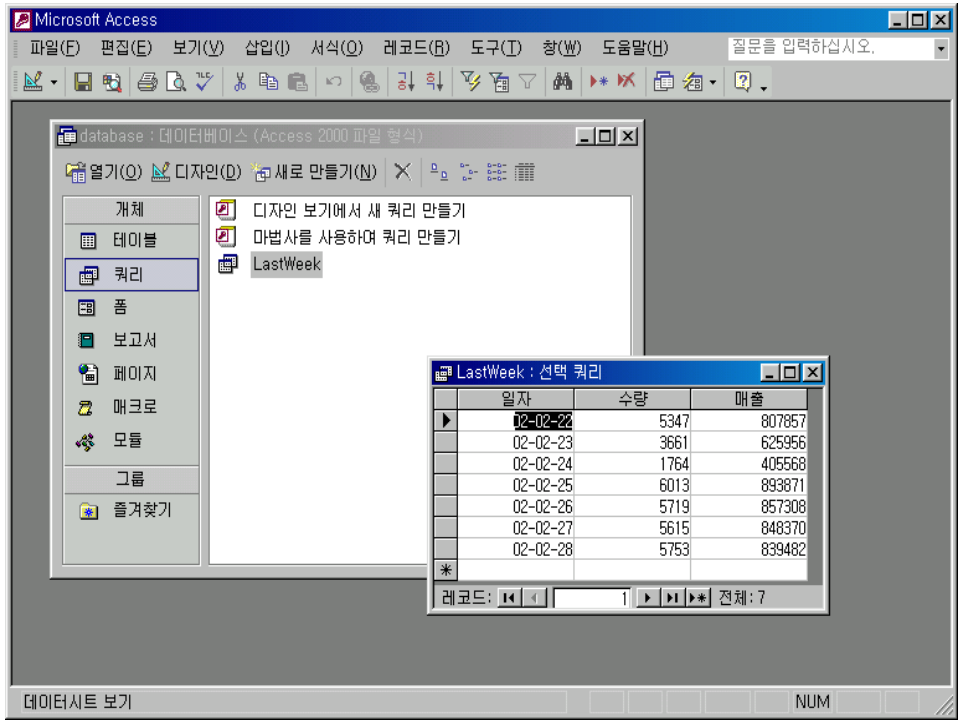
OneClick(72.xls)

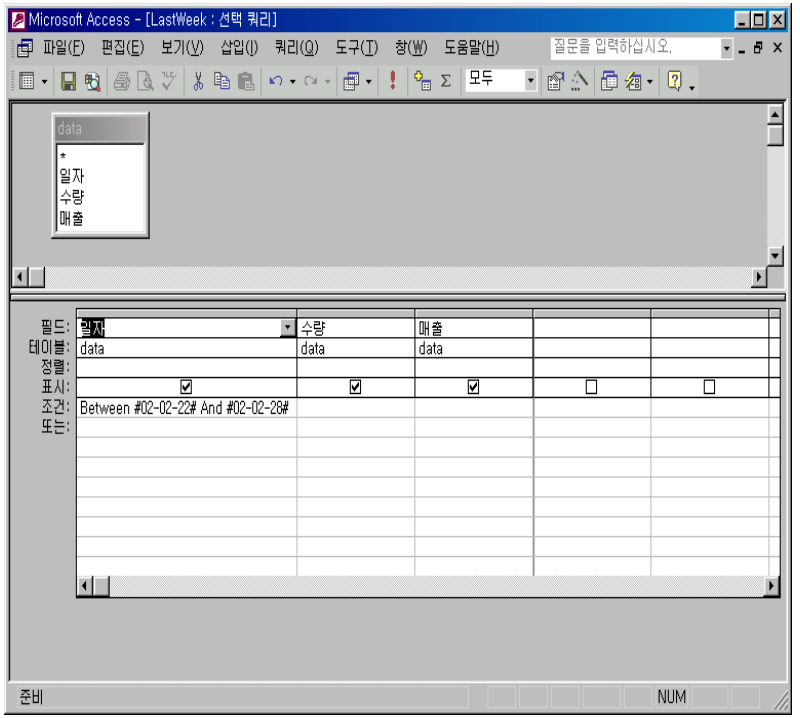
Oneclick 프로시져는 앞의 4단계로 이루어진 각 단계별 프로시져를 한번에 연결하는 프로시져 입니다.
이 경우 각 코드를 Private 가 아닌 Public Sub (아무 말없이 단순히 Sub 로 선언하면 Public 임) 로 선언했을 경우, 다른 모듈에서 도 호출이 가능하고, 이 경우 Call 구문을 사용해서 각 프로시져를 호출합니다.

```
Sub OneClick()

Call ImportData
Call DeleteData
Call ToAccess
Call DeleteSheet

End Sub
```

복잡한 프로시져일수록 위와 같이 각 프로시져를 세분화해서 모듈화한 후 Call 구문으로 연결시켜 주는 것이 모르는 사람이 코드를 처음접하거나 다음에 필요에 의해서 코드를 수정할 경우 가독성(Readability)을 높일 수 있습니다.

Access의 Query 결과를 엑셀에 불러들이기





데이터의 분량이 너무 많아 엑셀에 들어가지 않을 경우라도, Access 에 데이터를 몽땅 넣어놓고 원하는 조건의 값만 엑셀에서 불러들여 분석할 수가 있답니다.

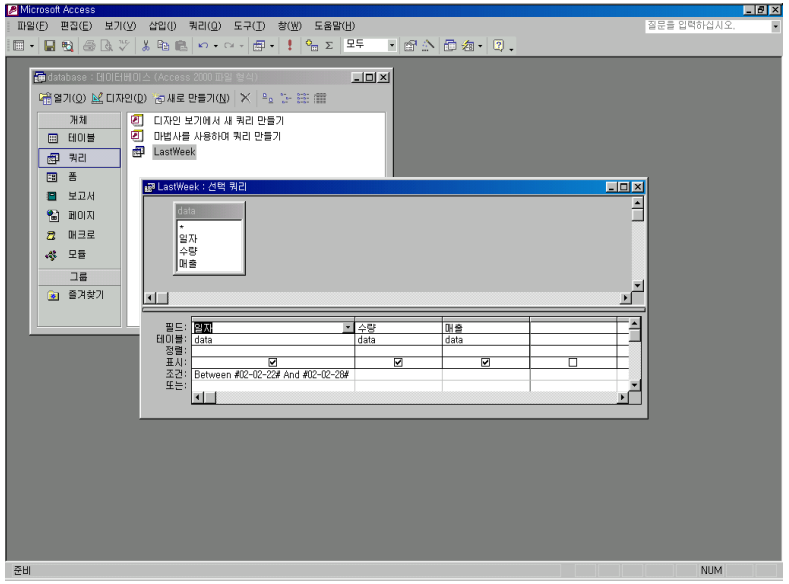
Access의 Query 결과를 엑셀에 불러들이기(78.xls)

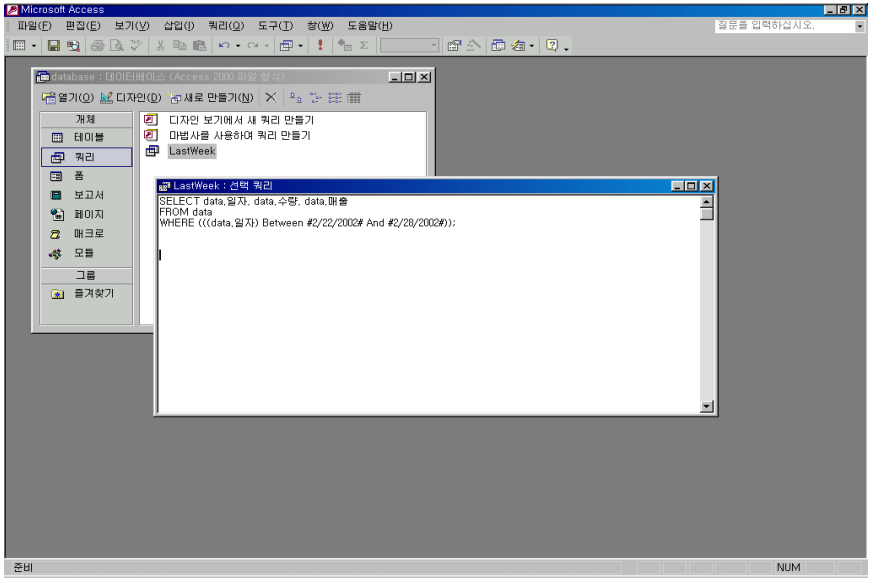
```
Sub WithQuery()
' 엑세스에 접근하기 위해서 DataBase, Recordset 형태의 변수를 DB, RS로 선언합니다.
   Dim DB As Database
   Dim RS As Recordset
   Dim i As Variant
' 현재 폴더의 database.mdb를 DB, LastWeek란 쿼리를 RS란 변수에 담습니다.
   Set DB = OpenDatabase(ThisWorkbook.Path & "\database.mdb")
   Set RS = DB.OpenRecordset("LastWeek")
' B1 셀을 기준으로 인접 셀에 내용이 있을경우 모두 지웁니다.
   Range("B10").CurrentRegion.Clear
' B10 셀부터 C10, D10 셀 등 옆으로 엑세스의 필드명을 그대로 불러옵니다.
   For i = 0 To RS.Fields.Count - 1
     Cells(10, i + 2).Value = RS.Fields(i).Name
   Next
' B11 셀에다 RS를 그대로 복사합니다.
   With ActiveSheet.Range("B11")
      .CopyFromRecordset RS
      .CurrentRegion.AutoFormat Format:=xlRangeAutoFormatClassic2
   End With
' DB를 닫습니다.
   DB.Close

End Sub
```

SQL 구문을 이용해서 Access의 데이터를 엑셀에 불러들이기(78.xls)

```
Sub WithSQL()
' 변수선언 (쿼리를 불러올때와는 달리 Startdate, Enddate를 문자형 변수로 추가로 선언)
Dim StartDate As String
Dim EndDate As String
Dim DB As Database
Dim RS As Recordset
Dim i As Variant
' 시작일과 종료일을 InputBox로 묻고 각각 StartDate, EndDate에 담습니다.
Set DB = OpenDatabase(ThisWorkbook.Path & "\database.mdb")
StartDate = InputBox("시작일을 입력하세요.", "krazy", "2002-02-22")
EndDate = InputBox("종료일을 입력하세요.", "krazy", "2002-02-28")
'OpenRecordset Method를 이용해서 SQL 구문을 직접 입력합니다.
Set RS = DB.OpenRecordset("SELECT data.일자, data.수량, data.매출 FROM data WHERE (((data.일자) Between #" & StartDate & "#
         And #" & EndDate & "#));")

' 새 시트를 하나 추가하고,
Sheets.Add
' 필드명을 하나씩 불러들인 후
  For i = 0 To RS.Fields.Count - 1
    Cells(1, i + 1).Value = RS.Fields(i).Name
  Next
' CopyFromRecordset Method를 이용해서 엑셀에 갖다 붙이고, 서식을 적당히 지정해 줍니다.
  With ActiveSheet
    .Range("A2").CopyFromRecordset RS
    .Columns(1).NumberFormatLocal = "yy""-""mm""-""dd"
    .Columns("B:C").NumberFormatLocal = "##,##0"
    .Columns("A:C").AutoFit
  End With
' Database를 닫아줍니다.
  DB.Close

End Sub
```

Access 파일의 LastWeek 쿼리 디자인





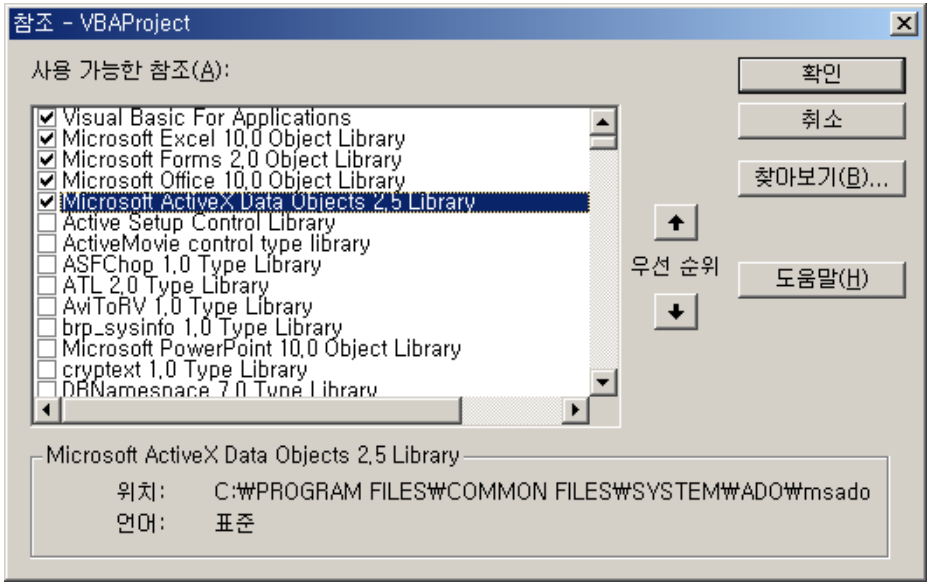
왼쪽화면에서 Access 보기 메뉴에서 SQL 보기를 택하면 오른쪽의 화면이 나온다.

SQL 구문만 다시 적어 보면,
SELECT data.일자, data.수량, data.매출 FROM data
WHERE (((data.일자) Between #2/22/2002# And #2/28/2002#));
이 부분을 한줄로 고쳐 위의 VBA 구문에 그대로 대입하면 됩니다.
단, 주의할 점은 위의 SQL 구문은 2002년 2월 22일부터 2002년 2월 28일까지 일자의 거래내역을 추출하라는 의미인데, 이를 각각 StartDate, EndDate라는 변수로만 바꿔주면 됩니다. 즉,
#2/22/2002# And #2/28/2002# 라는 표현 대신
#" & StartDate & "# And #" & EndDate & "# 로 고쳐줍니다.

ADO를 이용한 Data Exporting

DAO는 Access 와의 연동을 위한 Object 인 반면, ADO는 SQL을 사용하는 일반적인 DataBase 와 연동이 가능하고, ASP 등 Web 기반의 솔루션 제작에도 응용할 수 있는 보편화된 기법입니다. 엑셀 97 이하의 버전에서는 처리 속도가 빠르다는 이유로 DAO를 주로 이용했으나, 최근 PC 사양이 좋아짐에 따라 개인 업무에도 ADO를 사용는 추세에 따라 ADO 기법을 소개합니다.
앞의 강좌에서 DAO에 참조를 걸어준 것과 마찬가지로 VBE의 도구 > 참조 메뉴에서 아래 그림처럼 Microsoft ActiveX Data Objects 2.5 Library에 참조를 걸어주어야 합니다.

그리고 XP 이하 버전일 경우2.1, 2.0 등 다른 Library를 참조시켜줘야 하는 경우도 있습니다.



ADO를 이용해서 Access로 Export 하기(83.xls)

```
Sub ExportToAccessWithADO()
' 변수선언
Dim CNN As New ADODB.Connection
Dim RS As New ADODB.Recordset
Dim r As Integer
' ODBC에서 Microsoft Jet 엔진 이용, 본 파일과 같은경로에 있는 database.mdb 파일 Open
CNN.Open "Provider=Microsoft.Jet.OLEDB.4.0; Data Source=" & ThisWorkbook.Path & "\database.mdb;"
' data 테이블 Open
RS.Open "data", CNN, adOpenKeyset, adLockOptimistic, adCmdTable
' 8열부터 H행에 값이 있을 동안 순환하되 각 H,i,j 행의 값들을 Year, month, Revenues 필드에 입력
   r = 8
   Do While Len(Range("H" & r).Value) > 0
      With RS
         .AddNew
         .Fields("year") = Range("H" & r).Value
         .Fields("month") = Range("I" & r).Value
         .Fields("revenues") = Range("J" & r).Value
         .Update
      End With
      r = r + 1
   Loop
' Recordset, Connection 해제
   RS.Close
   Set RS = Nothing
   CNN.Close
   Set CNN = Nothing
End Sub
```

ADO를 이용해서 Access로 Export 하기

ADO를 이용할 경우 ADODB.Connection, ADODB.Recordset 을 이용하고, ODBC 에서 Microsoft Jet 엔진을 이용하기 위해 다음과 같이 선언하는 것을 제외하고는 DAO와 비슷합니다.

CNN.Open "Provider=Microsoft.Jet.OLEDB.4.0; Data Source=" & ThisWorkbook.Path & "\database.mdb;“

위 구문에서 빨간 색으로 된 부분이 핵심입니다.
그리고 DAO의 경우 SQL 의 Insert 문을 사용해서
Insert Into data In ' & ThisWorkbook.Path & "₩database.mdb' Select * From [현재시트명$]

과 같은 식으로 시트 전체를 입력하는 방법을 이용했는데, ADO 에서는 각각의 필드에 한 셀의 값들을 순환시켜 입력하도록 했습니다.

ADO 에서 커서를 열때 사용되는 Open Method의 구문은 다음과 같은데,
recordset.Open *Source*, *ActiveConnection*, *CursorType*, *LockType*, *Options*
위 구문은 각 인수에 대한 옵션을 다음과 같이 지정해 주었습니다.

RS.Open "data", CNN, adOpenKeyset, adLockOptimistic, adCmdTable

"data" 라는 것은 데이터 Source로 Access 의 경우 테이블이 여기에 해당합니다. ActiveConnection은 말 그대로 ADODB.Connection 에 해당합니다. 여기서는 CNN이라는 변수를 적어주면 됩니다.

ADO를 이용해서 Access에서 엑셀로 Data Import 하기(85.xls)

```
Sub ImportWithADO()
' 변수선언
Dim CNN As New ADODB.Connection
Dim RS As New ADODB.Recordset
Dim i As Integer
' ODBC에서 Microsoft Jet 엔진 이용, 본 파일과 같은경로에 있는 database.mdb 파일 Open
CNN.Open "Provider=Microsoft.Jet.OLEDB.4.0; Data Source=" & ThisWorkbook.Path & "\85.mdb;"
' 연도별 매출 쿼리 Open
RS.Open "연도별매출", CNN, adOpenStatic, adLockOptimistic, adCmdTable
' 쿼리의 필드명 불러오기
    For i = 0 To RS.Fields.Count - 1
       Range("G8").Offset(0, i).Value = RS.Fields(i).Name
    Next
' Recordset 복사해서 붙여넣기
       Range("G9").CopyFromRecordset RS
' Recordset, Connection 해제
   RS.Close
   Set RS = Nothing
   CNN.Close
   Set cn = Nothing
End Sub
```

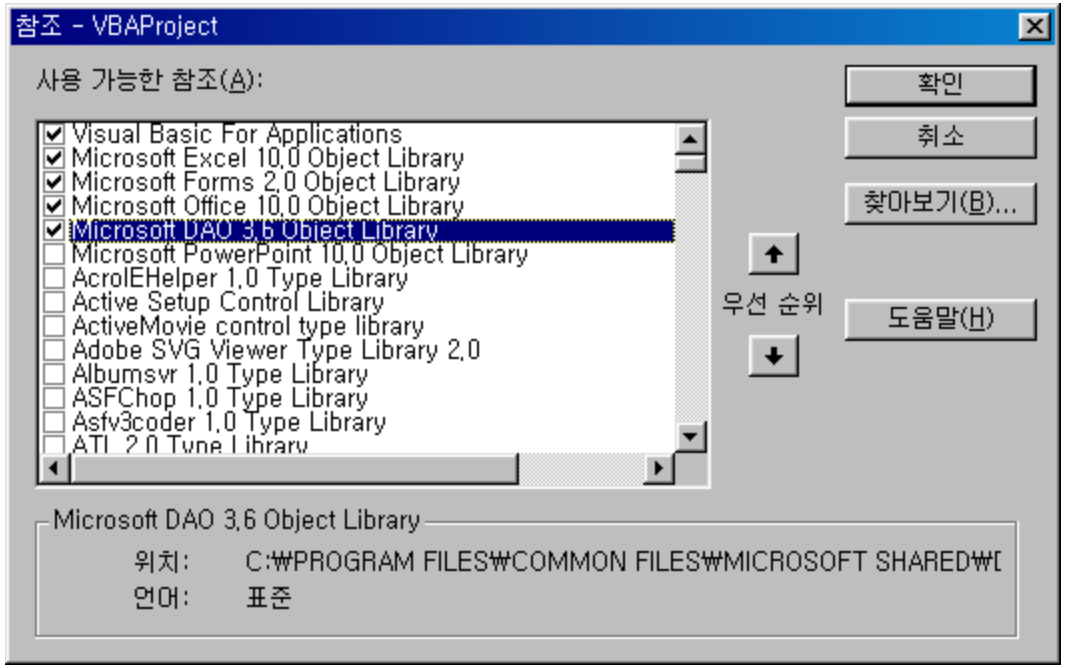
ODBC 에서 Microsoft Jet 엔진을 이용하기 위해 다음과 같이 선언하는 것은 앞의 강좌와 동일합니다.
CNN.Open "Provider=Microsoft.Jet.OLEDB.4.0; Data Source=" & ThisWorkbook.Path & "\database.mdb;"
ADO 에서 커서를 열때 사용되는 Open Method의 구문도 앞 강좌에서 설명드린 것과 같습니다.

RS.Open "연도별매출", CNN, adOpenStatic, adLockOptimistic, adCmdTable
database.mdb 파일 내에 '연도별매출'이라는 쿼리를 만들어 놓았기 때문에 위의 작업이 가능합니다만,
Access 가 아닌 다른 DB일 경우 SQL 구문을 직접 입력해도 됩니다.
즉,
RS.Open "연도별매출", CNN, adOpenStatic, adLockOptimistic, adCmdTable
라는 구문 대신 이에 해당하는 SQL 구문을 다음과 같이 직접 입력해줘도 됩니다.

RS.Open "SELECT data.year, Sum(data.revenues) AS 매출 FROM data GROUP BY data.year;", CNN, , , adCmdText
이 구문은 SQL 구문을 모르더라도 Access 에서 '연도별매출'이라는 쿼리를 실행시킨 상태에서 보기>SQL 보기 메뉴를 선택하면 나오는데, 이 부분을 그대로 복사해서 코드에 붙여넣기를 하시면 됩니다.

Excel 파일을 DataBase로 활용한 DAO

Access 같은 database 프로그램 없이 Excel 파일이 database 형식으로 되어 있다고 가정하고, Excel 파일에서 특정 조건에 해당하는 것들만 읽어 들이는 기법에 대해 소개합니다.
본 강좌 파일의 압축을 풀면 database.xls 라는 엑셀 파일이 하나 더 들어 있을 것입니다.
이 파일은 반드시 본 강좌파일과 같은 폴더에 넣고 실행해야 제대로 작동합니다.
database.xls 파일을 열어 보면, sample 이라는 시트가 있고, 그 시트 내에 date, Product, revenue 라는 세개의 field 명이 있습니다. 그리고 각 레코드는 2002년 1월 1일부터 2002년 12월 31일까지 TV, 세탁기, 냉장고의 일별 매출 실적이 나와 있습니다.
이번 강좌 역시 VBE(Visual Basic Editor)의 도구>참조 메뉴에서 Microsoft DAO 3.6 Object Library (XP 버전일 경우, 다른 버전은 3.5, 3.0 등) 에 참조를 걸어 줘야 합니다.



DataBase 전체를 불러오기 (87.xls)

```
Sub SQL_1()

Dim db As dao.Database
Dim rs As dao.Recordset
Dim i As Variant

Set db = OpenDatabase(ThisWorkbook.Path & "\87db.xls", False, True, "Excel 8.0;")
Set rs = db.OpenRecordset("SELECT * FROM [sample$] WHERE [product] LIKE 'TV'")

Sheets.Add
   For i = 0 To rs.Fields.Count - 1
     Cells(1, i + 1).Value = rs.Fields(i).Name
   Next
   With ActiveSheet
      .Range("A2").CopyFromRecordset rs
      .Columns(1).NumberFormatLocal = "yy""-""mm""-""dd"
      .Columns("B:C").NumberFormatLocal = "##,##0"
      .Columns("A:C").AutoFit
   End With
rs.Close
Set rs = Nothing
db.Close
Set db = Nothing

End Sub
```

DataBase 전체를 불러오기

OpenDatabase Method를 사용해서 db를 다음과 같이 정의했는데,
Set db = OpenDatabase(ThisWorkbook.Path & "₩database.xls", False, True, "Excel 8.0;")

같은 경로에 있는 database.xls 파일을 열어라는 의미입니다.
False는 공유모드로 열기(True면 단독모드)라는 의미이고, 그 다음의 True 는 읽기모드(False일 경우 쓰기까지 가능)로 열어라는 의미입니다. 만일 Excel에 xxx 라는 password가 걸려있을 경우 Excel 8.0; 대신
Excel 8.0; pwd=xxx;
와 같이 바꿔주면 됩니다. 이런 기법은 엑셀의 경우 뿐만 아니라, Access 에도 적용됩니다.
Web 폴더 내의 Access 파일에 PassWord를 걸어놓고 InputBox로 PassWord를 아는 사람만 접근 가능하도록 databasae를 설계할 경우 유용합니다.
그리고 Recordset 정의 부분은 OpenRecordset Method를 이용한다는 점은 앞의 강좌들과 동일한데,다만 SQL 구문에서 Table (엑셀의 경우 시트가 하나의 테이블이 됨) 을 정의할 때 [시트명$]와 같은 식으로 사용한다는 점이 다릅니다.

Set rs = db.OpenRecordset("SELECT * FROM [sample$] WHERE [product] LIKE 'TV'")
이런 기법들을 이용하면 닫혀있는 엑셀 파일 자체를 database로 활용해서
DAO를 사용할 수 있습니다.

Excel DAO를 이용한 SQL 구문 연습(89.xls)

```
public stSQL As String

Sub ExcelDAO()

Dim db As dao.Database
Dim rs As dao.Recordset
Dim i As Variant

Set db = OpenDatabase(ThisWorkbook.Path & "₩sql 예제.xls", False, True, "Excel 8.0;")
Set rs = db.OpenRecordset(stSQL)

Sheets.Add

    For i = 0 To rs.Fields.Count - 1
      Cells(1, i + 1).Value = rs.Fields(i).Name
    Next
    With ActiveSheet
       .Range("A2").CopyFromRecordset rs
       .Columns(1).NumberFormatLocal = "yy""-""mm""-""dd"
       .Columns("B:F").NumberFormatLocal = "##,##0"
       .Columns("A:F").AutoFit
    End With
rs.Close
Set rs = Nothing
db.Close
Set db = Nothing

End Sub
```

Excel DAO를 이용한 SQL 구문 연습 (89.xls)

```
Sub Select_1()
  stSQL = "SELECT 일자, 품목 FROM [거래내역$]"
  Call ExcelDAO
End Sub
Sub Select_2()
  stSQL = "SELECT * FROM [거래내역$]"
  Call ExcelDAO
End Sub
Sub Operators()
  stSQL = "SELECT 품목, 제조회사, 가격, 원가, 가격-원가 as 공헌이익 FROM [단가표$]"
  Call ExcelDAO
End Sub
Sub Concatenation()  '제조회사와 품목을 하나로 엮어서 3개의 필드로 표시
  stSQL = "SELECT 제조회사 & "" "" & 품목 As 제품, 가격, 원가 FROM [단가표$]"
  Call ExcelDAO
End Sub
Sub Where_1()
  stSQL = "SELECT * FROM [거래내역$] Where 품목 = ""마우스"""
  Call ExcelDAO
End Sub
Sub Where_2()
  stSQL = "SELECT * FROM [거래내역$] Where 품목 = ""마우스"" or 품목 = ""키보드"""
  Call ExcelDAO
End Sub
```

Excel DAO를 이용한 SQL 구문 연습 (89.xls)

```
Sub Where_3()
  stSQL = "SELECT * FROM [거래내역$] Where 개수 >= 800"
  Call ExcelDAO
End Sub
Sub Where_4()
  stSQL = "SELECT * FROM [거래내역$] Where 일자 Between #2002-05-01# and #2002-05-31#"
  Call ExcelDAO
End Sub
Sub Where_5()
  stSQL = "SELECT * FROM [거래내역$] Where 품목 like ""모*"""
  Call ExcelDAO
End Sub
Sub OredrBy_1()
  stSQL = "SELECT * FROM [단가표$] Order By 제조회사"
  Call ExcelDAO
End Sub
Sub OredrBy_2()
  stSQL = "SELECT * FROM [단가표$] Order By 제조회사 Desc"
  Call ExcelDAO
End Sub
Sub OredrBy_3()  '첫번째는 제조회사, 두번째는 품목
  stSQL = "SELECT * FROM [단가표$] Order By 제조회사, 품목"
  Call ExcelDAO
End Sub
```

Excel DAO를 이용한 SQL 구문 연습

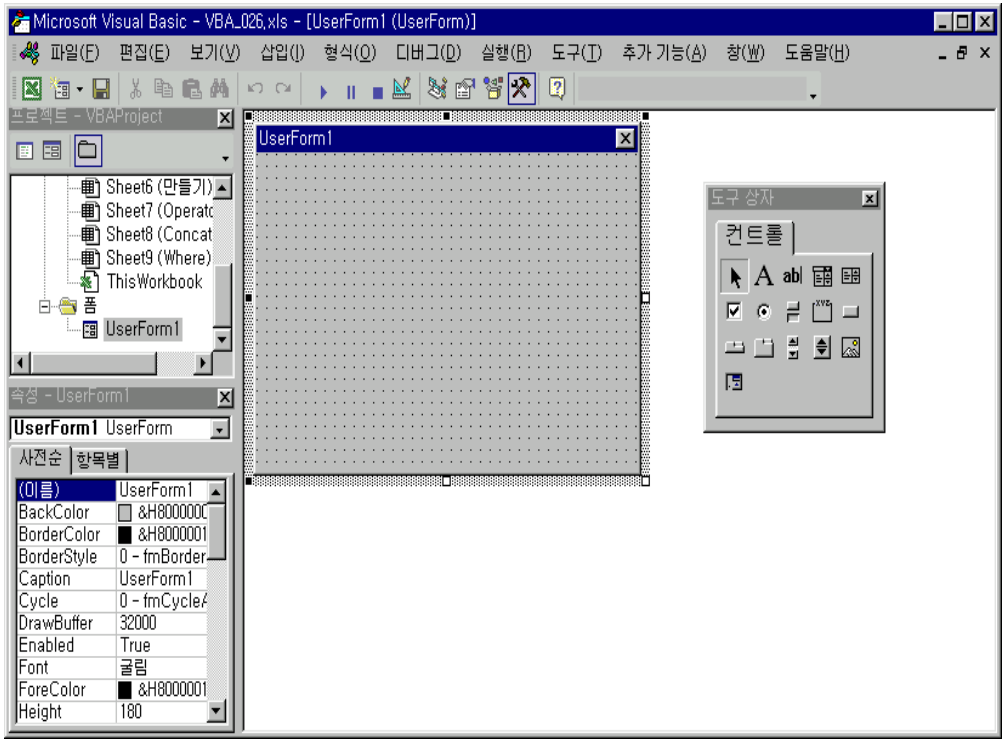
```
Sub GroupBy_1()  '거래내역 시트에서 각 품목별 거래 수량의 합을 합계수량
  stSQL = "SELECT 품목, sum(개수) as 합계수량 FROM [거래내역$] Group By 품목"
  Call ExcelDAO
End Sub
Sub GroupBy_2()  'Where 구문을 이용해서 5월 한달동안의 품목별 수량을 합계수량
  stSQL = "SELECT 품목, sum(개수) as 합계수량 FROM [거래내역$] Where (일자 Between #2002-05-01# and
#2002-05-31#) Group By 품목"
  Call ExcelDAO
End Sub
```

지금까지는 한 테이블(엑셀에서는 시트)에 대한 데이터만 추출하는 경우였습니다.
두개 이상의 테이블을 연결시켜 추출할 경우 Join 을 사용하고 가장 일반적인 것이 Inner Join 입니다.
거래내역과 단가표란 두개의 테이블이 있고, 품목이란 필드명이 둘다 있으므로 다음과 같이 Join을 걸어주면 거래내역 테이블에 제조회사, 단가 등을 불러들일 수 있습니다.

```
Sub InnerJoin_1()
  stSQL = "SELECT [거래내역$].일자, [거래내역$].품목, [단가표$].제조회사,"
  stSQL = stSQL & "[거래내역$].개수, [단가표$].가격 FROM [거래내역$]"
  stSQL = stSQL & "INNER JOIN [단가표$] ON [거래내역$].품목 = [단가표$].품목"
  Call ExcelDAO
End Sub
```

UserForm 만들기

가장 원시적인 형태의 UserForm을 만들오 보도록 하겠습니다.
먼저 VBE(Visual Basic Editor)를 실행시켜 삽입>사용자 정의 폼을 선택하면, VBE 모양이 다음과 같이 바뀔 것입니다. 다음과 같은 모양이 아닐 경우 보기 > 프로젝트탐색기 또는 보기 > 속성창 등을 선택하면 아마도 왼쪽의 화면들이 나타날 것입니다.



이렇게 하면 가장 원시적인 형태의 사용자 정의 폼이 만들어 지는 것입니다.
사용자 정의 폼의 이름은 기본값이 UserForm1 이 되고 Caption 값도 UserForm1 이 됩니다.
여기서 속성 창에 Caption 값을 “Hyundai Steel"로 바꿔주면 Caption만 바뀌고 실제 이름은 UserForm1로 남게 됩니다.

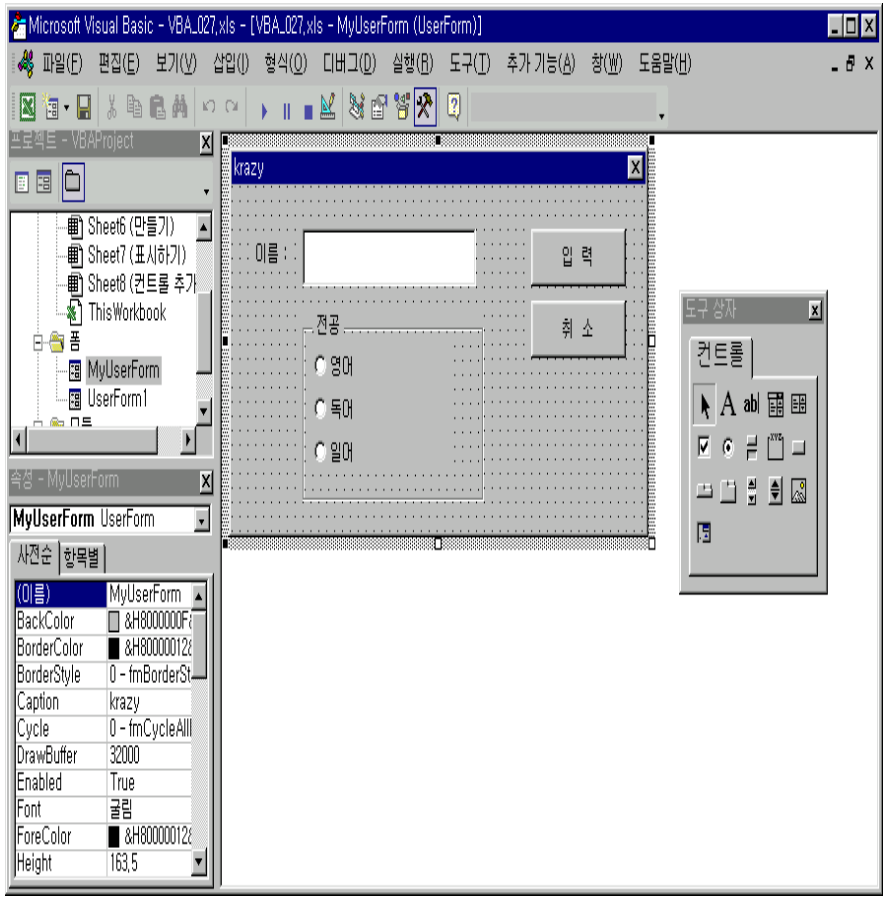
UserForm 표시하기

이번에는 사용자 정의 폼을 워크시트의 화면상에 나타나도록 해 보겠습니다.
VBE 의 삽입메뉴에서 모듈을 선택해서 module1을 추가하고 다음 코드를 입력해 보세요.

```
Sub ShowUserForm()

    UserForm1.Show

End Sub
```

그 다음 아래 버튼에 연결시켜 위 코드를 실행시켜 보세요.
그러면 앞 시트에서 만든 아무런 입력값도 없는 멍텅구리 UserForm이 나타날 것입니다.

UserForm에 컨트롤 추가하기 (95.xls)



설명

1. 이름이란 글자는 도구상자에서 라벨을 선택하고 직접 입력합니다.
2. 이름 옆의 이름 입력 상자는 텍스트 상자 컨트롤을 적당한 크기로 그려 줍니다.
   이름은 TextBox1 로 되어 있는데 txName 이라고 속성창에서 바꿔 줍니다.
3. 프레임 컨트롤을 도구상자에서 선택하고 적당한 크기로 그려준 후, 속성의 Caption에 "전공"이라고 입력합니다.
4. 프레임 안에 옵션버튼을 3개를 그려 넣고 Caption을 각각 영어, 독어, 일어로 고쳐준 후, 이름은 OptionButton1, OptionButton2, OptionButton3 을 각각 opEnglish, opGerman, opJapanese 으로 고쳐줍니다. 이렇게 이름을 바꿔주는 이유는 나중에 코딩할 때 가독성을 높이기 위해서 입니다.
   옵션 버튼간 간격을 맞추기 힘들면 옵션버튼 3개를 몽땅 선택(Shift 키를 누른 상태에서 마우스로 선택하면 됨)한 상태에사 VBE 메뉴에서 형식 > 맞춤 > 왼쪽을 선택해 주면 됩니다.
5. 명령단추를 두개 만들고 Caption을 입력, 취소로 고쳐준 후 이름을 각각 CmdOK, CmdCancel로 고쳐줍니다.

이렇게 MyUserForm을 만든 후 다음 코드를 모듈에 적어넣고, 다시 버튼에 연결시킵니다.

```
Sub ShowMyUserForm()
   MyUserForm.Show
End Sub
```

앞에서 만든 것보다 훨씬 형태를 많이 갖추고 있지만 역시 확인, 취소 버튼도 작동하지 않는
멍텅구리 UserForm 입니다.

코드 입력하기(95.xls)

앞에서 만든 MyUserForm에 주요 코드를 입력해 보도록 하겠습니다.
VBE 의 취소 버튼을 더블클릭하면 다음과 같은 코드가 모듈이 아닌 사용자 정의 폼 의 코드부분에 자동 생성됩니다.
Private Sub CmdCancel_Click()
End Sub
이는 취소 버튼의 이름을 CmdCancel 이라고 정해줬기 때문입니다.
여기에다 다음과 같이 코드를 입력해 줍니다.

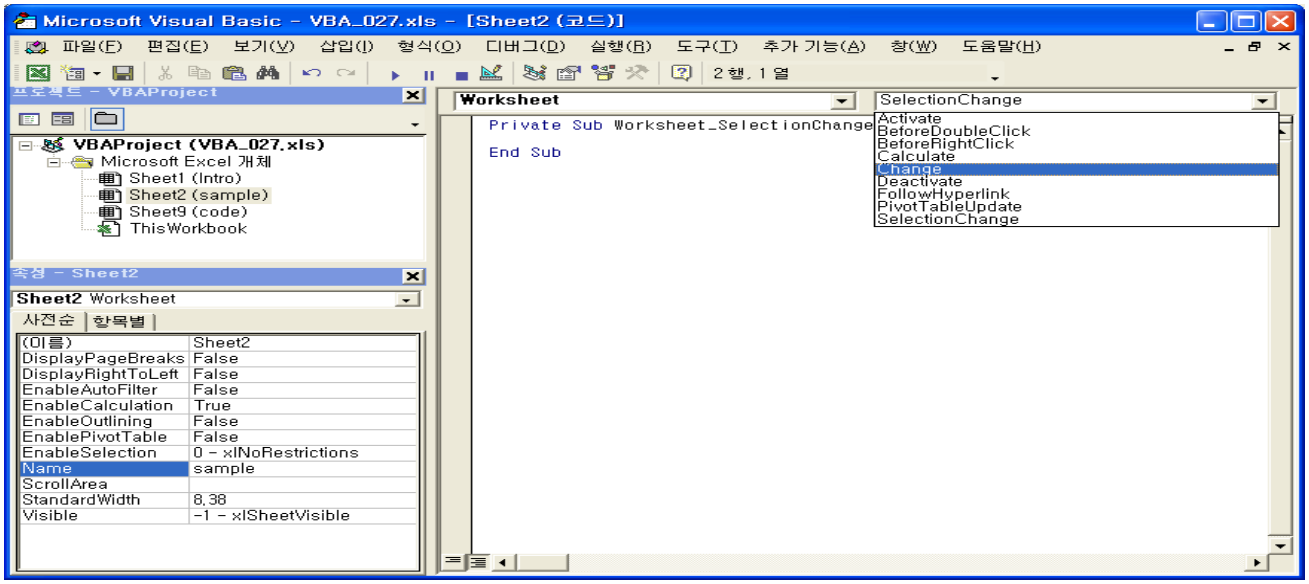
```
Private Sub CmdCancel_Click()
  Unload MyUserForm
End Sub
```

그리고 입력 버튼을 더블클릭후 다음 코드를 입력합니다.

```
Private Sub CmdOK_Click()
  r = Application.WorksheetFunction.CountA(Range("a:a")) + 1

  Cells(r, 1) = txName

  If opEnglish Then Cells(r, 2) = "영어"
  If opGerman Then Cells(r, 2) = "독일어"
  If opJapanese Then Cells(r, 2) = "일본어"

End Sub
```

이렇게 해서 옆의 완성된 보습 시트에보 보시는 것처럼 간단한 형태의 UserForm을 만들 수 있습니다.
다른 컨트롤들도 도움말을 참조해서 여러가지 만들어 보시기 바랍니다.

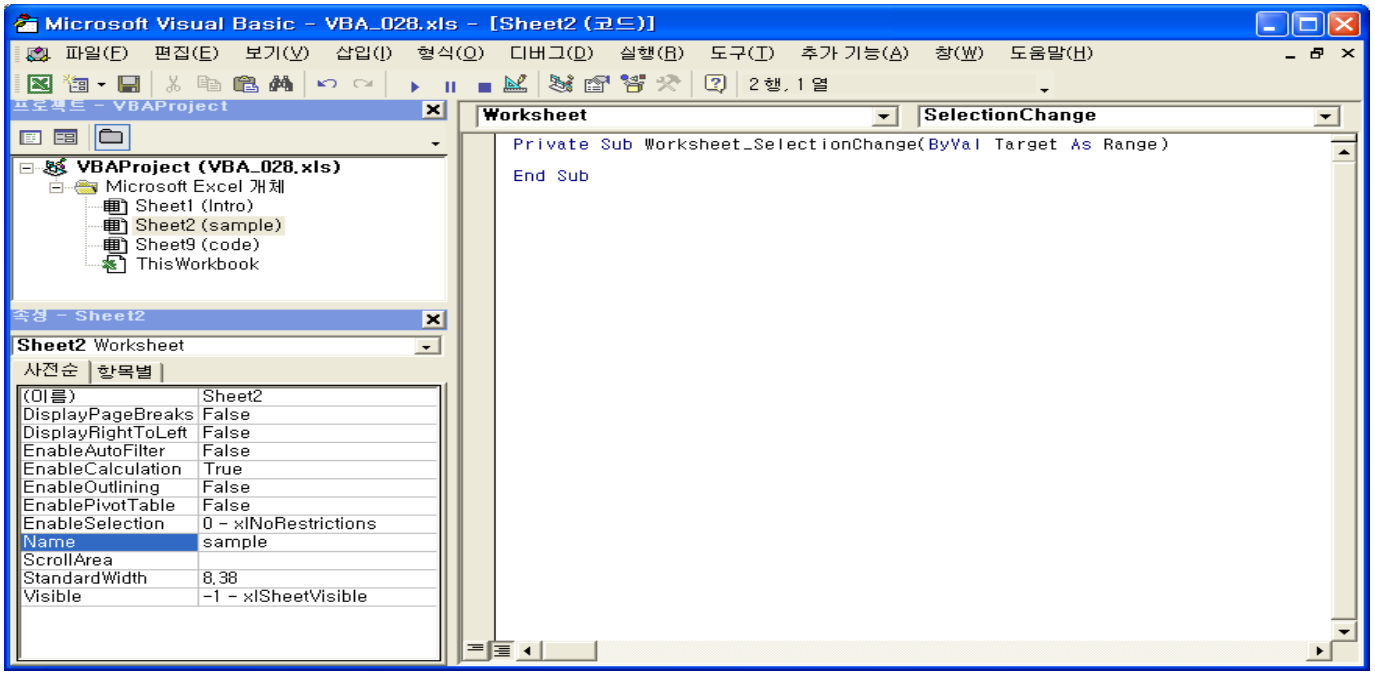
숫자 Index를 입력하면 셀값이 자동으로 해당 값으로 바뀌도록 하기(97.xls)



이 프로시져는 시트가 바뀔 때마다 코드가 실행되는 프로시져 입니다.
이 상태에서 다음과 같이 코드를 직접 입력합니다.

```
Private Sub Worksheet_Change(ByVal Target As Range)

  On Error Resume Next
  MyIndex = Array("답", "전", "대", "임", "도", "구", "천", "묘", "잡")
  If Not Intersect(Target, Range("E3:E49")) Is Nothing Then
   Target.Value = MyIndex(Target.Value - 1)
  End If

End Sub
```

선택한 셀이 있는 행열만 다른 색으로 강조하기(98.xls)



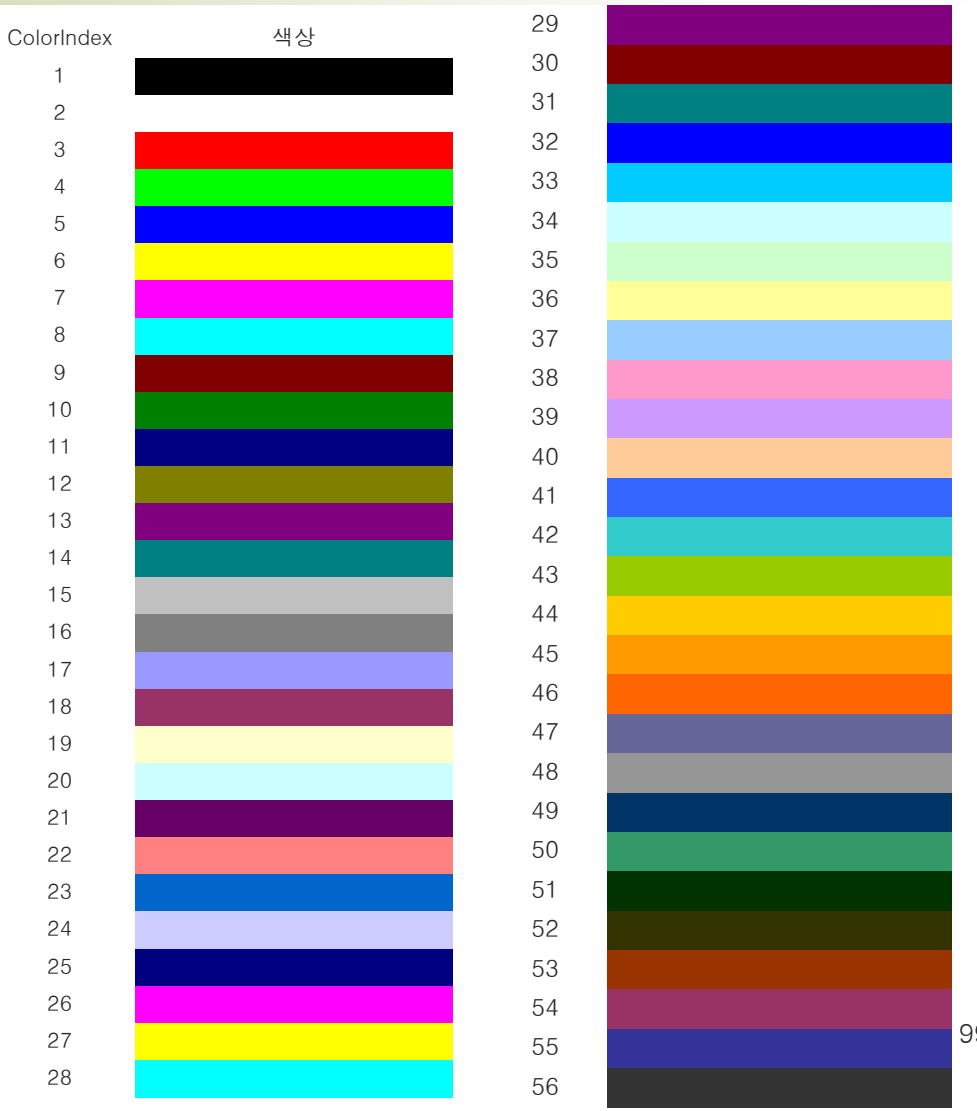

```
Private Sub Worksheet_SelectionChange(ByVal Target As Range)
        Cells.Interior.ColorIndex = xlNone
        With ActiveCell
          .EntireRow.Interior.ColorIndex = 36
          .EntireColumn.Interior.ColorIndex = 4
        End With
End Sub
```

ColorIndex 속성

```
Sub ExFont()

If Range("A14:A21").Font.ColorIndex = 1 Then
   Range("A14:A21").Font.ColorIndex = 5
Else: Range("A14:A21").Font.ColorIndex = 1
End If

End Sub
```

```
Sub Exinterior()

If Rows("54:61").Interior.ColorIndex = 15 Then
   Rows("54:61").Interior.ColorIndex = 3
Else: Rows("54:61").Interior.ColorIndex = 15
End If

End Sub
```

```
Sub Exborders()

If Range("F35:K42").Borders.LineStyle = xlNone Then
   Range("F35:K42").Borders.ColorIndex = 5
Else: Range("F35:K42").Borders.LineStyle = xlNone
End If

End Sub
```



인수가 없는 사용자 정의 함수 만들기

각 셀의 수식을 보시기 바랍니다. 수식은 {}로 묶여있어 배열식으로 입력된 것을 알 수 있습니다. 즉, BrenchNames 란 함수를 입력할 때에는 7개 셀범위(여기서는 B15:H15)를 미리 선택한 상태에서 =BrenchNames() 라고 입력후 Ctrl + shift + Enter 키를 동시에 눌러 입력합니다.

```
Function BrenchNames()

BrenchNames = Array("강남지사", "강북지사", "경인지사", "부산지사", "대구지사", "대전지사", "광주지사")

End Function
```

아마도 "만일 지사명을 세로로 입력해야 할 경우에는?" 하고 의문을 가지시는 분들도 계실 것입니다만, 이 경우 아래와 같이 transpose 함수를 이용하시면 됩니다.
=TRANSPOSE(brenchnames())

사용자 정의 메뉴 만들기(101.xls)

엑셀에는 파일-편집-보기- --- - 도움말의 9개 메뉴가 있습니다. VBA 를 이용해서 원하는 메뉴를 추가할 수가 있습니다.
메뉴 생성 및 삭제 코드 작성하기

```
' Special 이라는 메뉴 생성

Sub NewMenu()
    Dim MyMenu As CommandBarControl
    Set MyMenu = CommandBars(1).Controls.Add _
        (Type:=msoControlPopup, _
         Before:=CommandBars(1).FindControl(ID:=30010).Index, _
         temporary:=True)

    With MyMenu
        .Caption = "&Special"

        With .Controls.Add(Type:=msoControlButton)
            .Caption = "데이터 생성(&D)"
            .FaceId = 150
            .OnAction = "ProduceData"
        End With

        With .Controls.Add(Type:=msoControlButton)
            .Caption = "&양수만 더하기(&P)"
            .OnAction = "SumPlus"
        End With

        With .Controls.Add(Type:=msoControlButton)
            .Caption = "&음수만 더하기(&M)"
            .OnAction = "SumMinus"
        End With

        With .Controls.Add(Type:=msoControlPopup)
            .Caption = "조건부 서식..."
            .BeginGroup = True

            With .Controls.Add(Type:=msoControlButton)
                .Caption = "최소값 표시(&N)"
                .OnAction = "MinValue"
            End With

            With .Controls.Add(Type:=msoControlButton)
                .Caption = "최대값 표시(&X)"
                .OnAction = "MaxValue"
            End With

        End With

    End With

End Sub
```

사용자 정의 메뉴 만들기(101.xls)

```
' Special 메뉴 지우기
Sub DeleteMenu()
    On Error Resume Next
    CommandBars(1).Controls("Special").Delete
End Sub

'컨트롤 구조를 알아내는 함수
Sub ListUpControls()

Dim CmdControl As CommandBarControl
Dim CmdBar As CommandBar
Dim i As Integer

i = 2

For Each CmdBar In CommandBars
  Cells(i, 1).Value = CmdBar.Name
  i = i + 1
  For Each CmdControl In CmdBar.Controls
      Cells(i, 2).Value = CmdControl.Caption
      Cells(i, 3).Value = CmdControl.ID
      i = i + 1
   Next CmdControl

Next CmdBar

End Sub
```

Special 이라는 메뉴도 이들 첫번째 부류의 메뉴가 되는데, Controls.Add 라는 식으로 선언합니다. 이 경우 위치는 ID 번호가 30010이라는 컨트롤, 즉 도움말 메뉴 앞에 지정합니다.
참고로 각 메뉴별 ID 번호는 다음과 같습니다.

메뉴명	ID
파일	30002
편집	30003
보기	30004
삽입	30005
서식	30006
도구	30007
데이터	30011
창	30009
도움말	30010

Sub 메뉴에 대한 코드 작성하기

```
' -1000 부터 1000까지 무작위 숫자 생성
Sub ProduceData()
Dim rng As Range

  For Each rng In Range("A1:E20")
     rng.Value = Int(Rnd() * 1000)
      If rng Mod 2 = 0 Then
      rng.ClearContents
      rng.Value = Int(Rnd() * -1000)
      End If
  Next rng

End Sub

' 양수만 더하기
Sub SumPlus()
Dim rng As Range
Dim MyVal As Double

  MyVal = 0
  For Each rng In Selection
     If rng.Value > 0 Then
     MyVal = MyVal + rng.Value
     End If
  Next rng

  MsgBox "선택한 범위에서 양수의 합은" & Format(MyVal,
         "##,##0") & "입니다."

End Sub
```

```
' 음수만 더하기
Sub SumMinus()
Dim rng As Range
Dim MyVal As Double

  MyVal = 0
  For Each rng In Selection
     If rng.Value < 0 Then
     MyVal = MyVal + rng.Value
     End If
  Next rng

  MsgBox "선택한 범위에서 음수의 합은" & Format(MyVal, "##,##0") & "
입니다."
End Sub

' 최대값에 빨간색으로 표시하기
Sub MaxValue()

  With Selection
     .FormatConditions.Delete
     .FormatConditions.Add Type:=xlExpression, Formula1:= _
       "=" & Selection.Resize(1, 1).Address(False, False) & _
        "=MAX(" & Selection.Address & ")"
     .FormatConditions(1).Interior.ColorIndex = 3
  End With


End Sub
```

Sub 메뉴에 대한 코드 작성하기

```
' 최소값에 초록색으로 표시하기
Sub MinValue()

   With Selection
      .FormatConditions.Delete
      .FormatConditions.Add Type:=xlExpression, Formula1:= _
        "=" & Selection.Resize(1, 1).Address(False, False) & _
         "=Min(" & Selection.Address & ")"
      .FormatConditions(1).Interior.ColorIndex = 4
   End With


End Sub
```

이벤트 프로시져로 연결하기

본 파일을 열자마자 Special 이라는 사용자 정의 메뉴가 생성되었습니다.
또한 본 파일을 닫자마자 Special 이라는 메뉴는 사라집니다.
파일이 열리자마자, 또는 파일을 닫기 직전에 이런 장치를 해야 하는데, 이는 각각 Workbook_Open, Workbook_BeforeClose 라는 이벤트 프로시져를 이용합니다.
Code2 시트에서 작성한 NewMenu, DeleteMenu 프로시져를 파일이 열리는 순간, 그리고 파일이 닫히기 직적에 실행되도록 하기 위해 ThisWorkbook 모듈에 다음 코드를 작성합니다.

```
' 파일이 열리자 마자 NewMenu 프로시져 호출
Private Sub Workbook_Open()
Call NewMenu
End Sub

' 파일이 닫히기 직전에 DeletMenu 프로시져 호출
Private Sub Workbook_BeforeClose(Cancel As Boolean)
Call DeleteMenu
End Sub
```

이런 식으로 해서 사용자 정의 메뉴를 이용한 솔루션이 완성됩니다.
자신이 만든 여러 솔루션을 이런식으로 파일 형태로 배포할 수도 있고, XLA 파일 형태로 저장해서 추가기능으로 만들어 다른 사람에게 배포할 수도 있답니다.

FaceID

```
Sub ListUpFaces()

Dim CmdControl As CommandBarControl
Dim CmdBar As CommandBar
Dim a As Integer, b As Integer, c As Integer

On Error Resume Next

Set CmdBar = CommandBars.Add(Position:=msoBarFloating, _
          MenuBar:=False, temporary:=True)
Set CmdControl = CmdBar.Controls.Add(Type:=msoControlButton, _
          temporary:=True)
c = 1
Do While Err.Number = 0
   For b = 1 To 10
    a = a + 1
    CmdControl.FaceId = a
    CmdControl.CopyFace
    If Err.Number <> 0 Then Exit For
    ActiveSheet.Paste Cells(c, b + 1)
    Cells(c, b).Value = a
   Next b
   c = c + 1
Loop
CmdBar.Delete
End Sub
```

엑셀과 워드의 연동

엑셀과 오피스 각 프로그램(워드, 파워포인트, 아웃록, 액세스 등) 각 프로그램 간에는 VBA란 프로그래밍 언어를 통해서 서로 연동됩니다.
본 강좌에서는 엑셀과 워드의 연동에 대해 설명드리겠습니다.
먼저 Early Binding과 Late Binding의 개념, 그리고 엑셀 데이터를 워드로 옮기는 기법 반대로 워드의 자료를 엑셀로 읽어들이는 기법, 끝으로 현재차트를 워드 문서로 옮기는 기법을 소개합니다.

Early Binding과 Late Binding
ToWordText
FromWordText
ToWordChart

각 시트로 가서 버튼들을 눌러보고, 각 프로시져에 대한 해당 설명을 보시기 바랍니다.

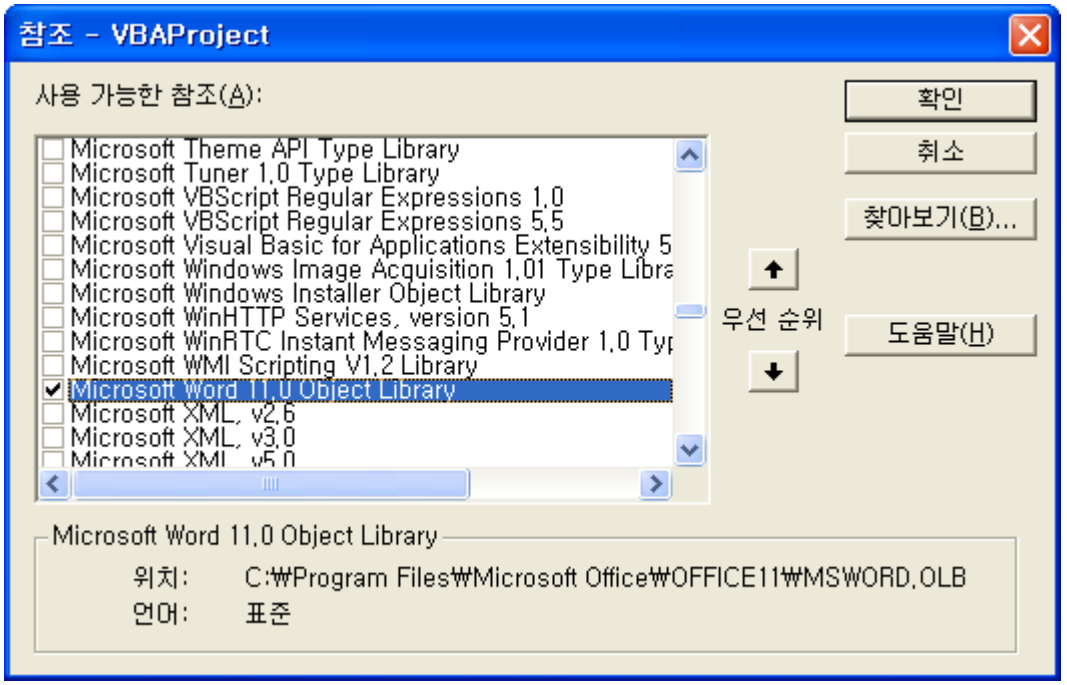
Late Binding(108.xls)

엑셀과 워드를 연결해 주는(Binding) 기법으로 Early Binding과 Late Binding 기법이 있습니다.
Early Binding 기법은 '내 니하고 미리 연결해 줄께' 라고 엑셀이 워드에게 알려주고 프로시져를 진행시키는 기법이고,
Late Binding은 프로시져가 일단 실행되고 나서 연결할 필요가 있을때 연결에 관한 해당 코드를 적어주는 기법입니다.
일반적으로 Early Binding 기법을 많이 사용합니다만, 원리를 알기 위해 Late Binding 기법부터 먼저 설명드리죠. 일단 다음 버튼을 눌러 보세요.

```
Sub LateBinding()

    'Word Application을 WordApp 라는 Object 변수로 선언.
    Dim WrdApp As Object
    Set WrdApp = CreateObject("Word.Application")

    'Word 버전을 메시지박스 형태로 반환.
    MsgBox "현재 당신이 사용하고 있는 워드 버전은" & vbLf _
        & "워드" & WrdApp.Version & "입니다."

    'Word Application을 끊고 변수를 비움.
    WrdApp.Quit
    Set WrdApp = Nothing

End Sub
```

라이브러리 추가

만일 에러가 나면, VBE(Visual Basic Editor)를 실행시키고, VBE '도구-참조' 메뉴에서 다음 그림처럼 Microsoft Word 11.0 Object Library에 체크를 해 줘야 합니다.
버전은 엑셀 버전에 따라 다른데, 오피스 2003은 11.0 XP(2002)는 10.0, 2000은 9.0입니다.



Early Binding

```
Sub EarlyBinding()

    Dim WrdApp As New Word.Application

    MsgBox "현재 당신이 사용하고 있는 워드 버전은" & vbLf _
        & "워드" & WrdApp.Version & "입니다."

    WrdApp.Quit
    Set WrdApp = Nothing

End Sub
```

여기서는 CreateObject 함수를 사용해서 WordApp라는 Object를 별도로 선언하지 않고,
엑셀 VBA 상에서 Word 라는 Application 개체를 바로 사용할 수 있는 장점이 있습니다.
코드 구성이 앞의 방법보다 훨씬 간단하죠.
다른 Word Object들도 바로 엑셀 VBA로 언급할 수가 있는데, 이 모든것이 미리 Word Library를
참조해 놓고 사용하기 때문에(Early Binding) 가능한 것입니다.
미리 참조시켜 놓아야 하는 단점은 있지만, 코드 구성이 가능하고, Word 와의 호환이 용이하기 때문에
많이 사용하는 기법입니다.

To WordText(111.xls)

다음과 같은 거래내역에 대해 각 건별로 팩스를 보낸다고 가정하죠.
팩스 메시지는 아래 색칠한 부분으로 동일하고, 거래내역만 아래 표처럼 다를 경우, 일일이 팩스 내용을 워드로 만들기란 쉽지가 않습니다.
이 경우 VBA를 이용해서 워드에 각 거래내역을 1페이지짜리 팩스 내용을 만들 수 있습니다. C 드라이브에 Temp라는 폴더를 만들고, 다음 버튼을 눌러 보시죠.
(Early Binding 기법을 사용했기 때문에 미리 Word Library를 참조시켜 놓아야 제대로 작동합니다.)

수신자	발주일	품목	개수	매출
박재영	2008-08-28	TV	5	4,000,000
김경수	2008-08-29	세탁기	4	2,800,000
진현일	2008-08-31	냉장고	8	9,600,000

귀하께서 주문하신 물품은 다음과 같습니다.
확인후 저희 부서로 연락 주시면 고맙겠습니다.
안녕히 계십시오.
Fax) 02-6220-XXXX

```
Sub ToDocText()

' 워드 프로그램을 wrdApp 라는 변수로 선언
   Dim wrdApp As Word.Application
   Set wrdApp = CreateObject("Word.Application")

' 웨드 프로그램 실행시 눈에 보이도록 Visible 속성을 True로 설정 (False로 설정 또는 미설정시 실행과정이 눈에 보이지 않게 됨.)
   wrdApp.Visible = True

' 혹시 C:/Temp 폴더에 Word1.doc 란 이름의 파일이 존재하면 파일 삭제
   If Dir(ThisWorkbook.Path & "₩doc₩Word1.doc") <> "" Then
         Kill ThisWorkbook.Path & "₩doc₩Word1.doc"
   End If
```

To WordText

```
' Word Application의 Document(문서파일) 하나 추가.
   With wrdApp
        .Documents.Add

' 엑셀 6행부터 8행까지 거래내역 표를 작성용 순환문
    For i = 6 To 8

    With .Selection

' 6행이 아닐경우에는 페이지 나눔(wdPageBreak)을 하라.
        If i <> 6 Then .InsertBreak Type:=wdPageBreak

' 워드에서 제목을 " 주 문 확 인 서"라고 입력, Aril Bold체, 크기 14로 지정
        .Font.Name = "Aril"
        .Font.Size = 14
        .Font.Bold = True
        .ParagraphFormat.Alignment = wdAlignParagraphCenter
        .TypeText Text:="주 문 확 인 서"
        .TypeParagraph
        .TypeParagraph

' 이하 굴림체, 13 폰트로 지정, 왼쪽정렬
        .Font.Name = "굴림체"
        .Font.Size = 12
        .Font.Bold = False
        .ParagraphFormat.Alignment = wdAlignParagraphLeft

' 수신자, 발주일 정보 입력
        .TypeText Text:="수 신 자:" & vbTab & Format(Cells(i, 2).Value, "yy-mm-dd")
        .TypeParagraph
        .TypeText Text:="발 주 일:" & vbTab & Cells(i, 3).Value
        .TypeParagraph
        .TypeParagraph

' 팩스메시지 입력
        .TypeText Text:=Range("H5").Value
        .TypeParagraph
        .TypeParagraph

' 품목, 개수, 매출정보 입력
        .TypeText Text:="품    목:" & vbTab & Cells(i, 4).Value
        .TypeParagraph
        .TypeText Text:="개    수:" & vbTab & Cells(i, 5).Value & "개"
        .TypeParagraph
        .TypeText Text:="매    출:" & vbTab & Cells(i, 6).Value * 0.00001 & "백만원"
        .TypeParagraph

    End With
    Next i

' 현재문서를 C:₩Temp 폴더 내 Word1.doc 란 이름으로 저장
      .ActiveDocument.SaveAs (ThisWorkbook.Path & "₩doc₩Word1.doc")
   End With

' 워드 프로그램을 끝내고, wrdApp란 변수값을 비움
   wrdApp.Quit
   Set wrdApp = Nothing

End Sub
```

From WordText(111.xls)

이번에는 워드에 있는 문서를 읽어와 엑셀의 각 셀에 뿌려주는 기법입니다.
앞에서 만든 C:₩Temp 폴더의 word1.doc 파일을 읽어들여 보도록 하죠.
워드파일이 생성된 상태에서 Word Library를 참조시킨후 아래 버튼을 눌러 보세요.

```
Sub ReadWordDoc1()

Dim wrdDoc As Word.Document
Dim i As Integer
Dim wrdText As String, wrdRange As Word.Range


Set wrdDoc = GetObject(ThisWorkbook.Path & "₩doc₩Word1.doc")

With Range("B11")
    .Value = "Word1.doc 파일에 있는 내용"
    .Font.Bold = True
End With

With wrdDoc

    '한 문단씩 끊어서 읽기
    For i = 1 To .Paragraphs.Count
       Set wrdRange = .Range(Start:=.Paragraphs(i).Range.Start, _
            End:=.Paragraphs(i).Range.End)
'Enter 키를 눌렀을 때 생기는 표시 제외
        wrdText = Left(wrdRange.Text, Len(wrdRange.Text) - 1)

        'B 열의 끝에다 워드 텍스트 불러오되 Clean 함수로 Tab 키 표시 제외
        Range("B65536").End(xlUp).Offset(1, 0).Value = _
            Application.WorksheetFunction.Clean(wrdText)

    Next i

    '워드 문서 닫기
    .Close

End With

'워드 문서를 메모리에서 비우기
Set wrdDoc = Nothing

'현재 엑셀문서 저장
ActiveWorkbook.Saved = True

End Sub
```

From WordText(111.xls)

코드를 약간 변형시켜, 수신자, 발주일, 품목, 개수, 매출 등 특정 문자열로 시작하는 행만 순서대로 읽어들일 수도 있습니다.

```
Sub ReadWordDoc2()

Dim wrdDoc As Word.Document
Dim i As Integer
Dim j As Variant
Dim wrdText As String, wrdRange As Word.Range

Set wrdDoc = GetObject(ThisWorkbook.Path & "₩doc₩Word1.doc")

'워드의 찾을 문자열을 배열 변수로 선언
k = Array("수 신 자", "발 주 일", "품  목", "개  수", "매  출")

With Range("B5")
      .Value = "Word1.doc 파일에 있는 내용"
      .Font.Bold = True
End With

With wrdDoc
    For i = LBound(k) To UBound(k)
       For j = 1 To .Paragraphs.Count
          Set wrdRange = .Range(Start:=.Paragraphs(j).Range.Start, _
              End:=.Paragraphs(j).Range.End)
          wrdText = Left(wrdRange.Text, Len(wrdRange.Text) - 1)

           '워드의 첫 문자열이 배열변수에 해당할 경우만 값 반환
          If InStr(1, wrdText, k(i), "1") > 0 Then
             Range("B65536").End(xlUp).Offset(1, 0).Value = _
               Application.WorksheetFunction.Clean(wrdText)
          End If

       Next j
    Next i

    .Close

End With

Set wrdDoc = Nothing

ActiveWorkbook.Saved = True

End Sub
```

From WordText(115.xls)

이번에는 차트 오브젝트를 복사해서 워드의 원하는 위치에 개체삽입 형식으로 붙여넣는 작업을 VBA로 구현해 보도록 하겠습니다.

```
Sub ToDocChart()

  ActiveSheet.ChartObjects(1).Activate
  ActiveChart.ChartArea.Copy

 Set wrdApp = CreateObject("Word.Application")
 wrdApp.Visible = True

If Dir("C:\Temp\Word2.doc") <> "" Then
   Kill "C:\Temp\Word2.doc"
End If

With wrdApp
     .Documents.Add

   With .Selection
        .Font.Size = 14
        .Font.Bold = True
        .ParagraphFormat.Alignment = wdAlignParagraphLeft
        .TypeText Text:=Range("a11").Value
        .TypeParagraph
        .PasteSpecial Link:=False, DataType:=wdPasteOLEObject
    End With

     .ActiveDocument.SaveAs ("C:\Temp\Word2.doc")

 End With

 wrdApp.Quit
 Set wrdApp = Nothing

End Sub
```

데이터 라벨에 플래쉬 효과 넣기 (116.xls)

```
Sub FlashChart1()

' 차트오브젝트, Flag 등을 변수로 선언합니다.
 Dim Cht As Chart
 Dim i As Integer
 Dim Flag As Boolean

' 시트 첫번째 차트 오브젝트(玄齋시트에는 차트 하나뿐임)를 Cht 라는 변수에 담습니다.
'일반 변수가 아닌 오브젝트를 변수에 담는 경우이므로 Set 문을 사용합니다.
 Set Cht = ActiveSheet.ChartObjects(1).Chart

' i 값이 1부터 6까지 순환하면서,
 For i = 1 To 6

'Flag 값이 False 이면, 데이터 라벨값이 보이도록 하고 Flag 값을 True로 설정합니다.
   If Flag = False Then
      Cht.ApplyDataLabels (xlShowValue)
      Flag = True
'Flag 값이 True이면, 데이터 라벨값이 보이지 않도록 한후 Falg 값을 False로 설정합니다.
   Else
      Cht.ApplyDataLabels (xlNone)
      Flag = False
   End If

'1초간 기다린 후
   Application.Wait Now + TimeSerial(0, 0, 1)

' For --- Next 사이 구문을 반복합니다.
 Next i

End Sub
```

막대차트의 막대에 플래쉬 효과 넣기(116.xls)

```
Sub FlashChart2()

' 차트오브젝트를 Cht 라는 변수로 선언합니다.
 Dim Cht As Chart
 Dim i As Integer

' 시트 첫번째 차트 오브젝트(玄齋시트에는 차트 하나뿐임)를 Cht 라는 변수에 담습니다.
 Set Cht = ActiveSheet.ChartObjects(1).Chart

' 첫번째 계열 테두리와 막대 색상을 흰색으로 지정합니다. (여기서는 계열이 1개 뿐인 차트임)
 With Cht.SeriesCollection(1)
     .Border.ColorIndex = 2
     .Interior.ColorIndex = 2
 End With

' 첫번째 계열의 각 포인트를 순환하면서 0.7초 간격으로 색상값을 차례로 빨간 색으로 바꿔줍니다.
 For i = 1 To Cht.SeriesCollection(1).Points.Count
    With Cht.SeriesCollection(1).Points(i)
        .Border.ColorIndex = 3
        .Interior.ColorIndex = 3
    End With
    Application.Wait Now + TimeSerial(0, 0, 0.7)
 Next i

End Sub
```