1. Macro & VBA

1) Macro

매크로란 일련 명령어의 조합으로 자주 사용하는 일련의 명령어를 기록해 두었다가 단축키나 버튼 클릭만으로 반복 재생할 수 있게 한다.
매크로를 사용함으로 반복적인 작업을 간단하게 실행시킬 수 있고, 복잡한 계산 과정 및 작업과정을 단순화할 수 있다.

2) VBA(Visual Basic Application)

매크로는 내부적으로 어플리케이션을 위한 비주얼 베이직 언어로 작성된다. 매크로를 기록하는 언어가 VBA이므로 매크로와 VBA는 결국 같은 것이라고 할 수 있다.

3) 매크로가 필요한 경우

- 반복적이거나 복잡한 일련의 엑셀 작업을 하나로 묶어서 재사용할 때
- 엑셀 기본 함수로 지원되지 않는 계산이나 복잡한 계산식을 쉽게 사용 가능하도록 새로운 함수를 정의할 때
- 엑셀의 기본 기능으로 처리할 수 없는 새로운 기능을 만들어 사용할 때
- 엑셀을 기본으로 하는 전문적인 프로그램을 개발할 때

2. VBA 프로그램의 구성

1) 애플리케이션(Application)

애플리케이션은 액셀, 워드, 액세스, 파워포인트와 같은 응용 프로그램을 말하며, 액셀에서는 현재 실행 중인 프로그램인 액셀 자체를 의미한다. VBA 코드의 최상위 계층을 말하며, 액셀에서는 액셀 자체를 의미하기 때문에 액셀의 기능이나 정보를 확인할 때 사용한다.

2) 프로젝트(Project)

프로그램을 구성하는 모듈과 폼, 클래스 모듈의 집합을 의미하는 것으로 액셀 VBA에서는 하나의 통합 문서에 작성되는 모든 VBA 코드 내용을 하나의 프로젝트라 보면 된다.

3) 모듈(Module)

프로시저의 집합으로, 프로젝트를 구성하는 기본단위이다. 모듈은 표준 모듈과 폼 모듈, 클래스 모듈 등으로 구분된다.

표준 모듈	워크시트 모듈(Sheet로 표시되는 모듈)과 ThisWorkbook 모듈, 공용 모듈(일반적으로 사용하는 모듈)이 있다. 워크시트 모듈은 각각 하나씩 만들어지며, ThisWorkbook 모듈은 통합문서 즉, 액셀 파일 하나에 하나씩 만들어 진다.
폼 모듈	사용자 정의 폼을 정의하고 사용자 정의 폼의 컨트롤에 이벤트 프로시저를 작성하는 모듈
클래스 모듈	개체를 새롭게 정의해서 사용할 수 있도록 작성하는 모듈로 개체의 속성, 매서드, 이벤트를 정의하는 모듈

4) 프로시저(Procedure)

특정 기능을 실행하기 위해 모여진 명령문 집합으로, 실행 방법에 따라 Sub, Function, Property로 구분된다. 매크로 기록기를 이용해 작성한 VBA 코드는 Sub 프로시저를 사용하는 것이고, 액셀의 함수라고 불리는 기능은 Function 프로시저를, 개체의 속성을 새로 정의할 때는 Property 프로시저를 사용한다.

5) 사용자 정의 폼(User Form)

자료의 입출력을 효과적으로 하기 위한 대화상자로, 액셀 VBA를 이용해 사용자가 직접 설계한 대화상자를 만들 때 사용한다. 폼은 하나의 정보를 입출력할 수 있는 단위인 컨트롤 개체로 구성된다.

3. 프로시저의 구성

1) 개체(Object)

개체란 분리될 수 있는 하나하나의 작업단위이다. 액셀에서는 통합문서도 하나의 개체이고, 통합문서를 구성하는 시트, 셀, 도형들도 개체이다. 한 마디로 개체란 어떤 작업의 대상이 될 수 있는 것을 말한다.

2) 컬렉션(Collection)

관련있는 한 개 이상의 개체 집합을 말하는 것으로 한꺼번에 특정 개체들에 대한 작업을 처리할 때 사용하는 개념으로 일반적으로 개체 이름에 "S"를 붙여 사용한다.

Sheet	Sheets
worksheet	worksheets
Activecell	Selection

3) 속성(Property)

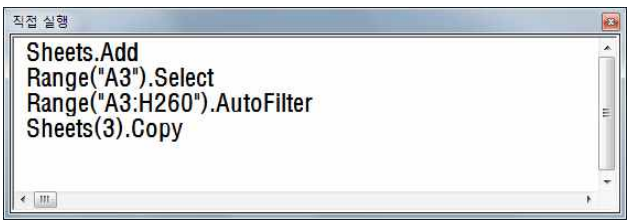
개체의 크기, 색, 모양 등과 같은 개체의 특성이나 상태를 말하는 것으로 VBA 코드에서는 개체와 속성 사이에 마침표를 찍어 구분한다.

지정된 속성을 표시할 때	개체.속성 MsgBox Sheet1.Name
새로운 속성을 지정할 때	개체.속성=새로운 속성값 Sheet1.Name="일사분기"

4) 매서드(Method)

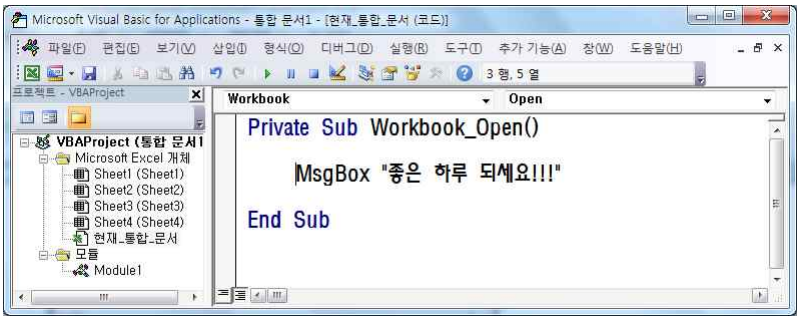
개체가 실행할 수 있는 동작, 행동을 의미한다.

'개체.매서드' 로 사용



5) 이벤트(Event)

개체가 어떤 행동을 할 때 발생하는 사건을 말한다. 즉, 액셀에서는 통합문서를 열 때나 닫을 때, 키보드를 누를 때 등 통합문서 개체의 이벤트이다.



6) 클래스(Class)

개체의 형식적인 정의를 의미한다. 개체의 속성과 동작 등을 제어하는 방법을 정의함으로써 개체의 모습을 완성시킨다.

4. VBA 연산자

1) 연산자의 종류와 우선순위

우선순위	산술	비교	논리
높음	지수(^)	같다(=)	Not
	부정(-)	같지 않다(<>)	And
	곱셈(*), 나눗셈(/)	작다(<)	Or
	정수 나눗셈(\ )	크다(>)	Xor
	나머지 연산(Mod)	작거나 같다(<=)	Eqv
	덧셈, 뺄셈(+, -)	크거나 같다(>=)	
	문자열 연결(&)	Like	
		Is	
낮음	높음 ◂		▸ 낮음

2) VBA 추가 연산자 사용법

연산자	기능	사용예
\ 정수나눗셈	나눗셈의 몫을 구함	? 11 \ 4
Mod 나머지 연산	나눗셈의 나머지를 구함	? 11 Mod 4
Like	문자열의 일부 포함된 값 비교 • ? : 한 문자 • * : 0개 이상의 문자 • # : 한 숫자(0~9) • [charlist] : charlist 안의 한 문자 • [!charlist] : charlist 안에 없는 문자	? "horizons" like "hor*" ? "신림1동" like "신?동" ? "F" like "[A~Z]" ? "F" like "[!A~Z]"
Is	두 개체가 같은지를 비교할 때 사용하며, 일반적인 숫자, 문자열 등의 비교에는 사용하지 않는다.	? Sheets(1) is ActiveSheet

5. 변수 & 상수

1) 변수

프로그램 처리과정에서 중간 계산값이나 결과값들을 잠시 보관해야 할 필요가 있는데, 이런 경우 변수를 이용하여 자료를 보관한다. 변수는 컴퓨터의 메모리 중 일부에 이름을 정의하여 사용하는 것을 말하며, 이렇게 정의된 메모리, 즉 변수 이름에 값을 기억시킬 때는 "변수=값" 형태로 사용된다.

2) 변수 선언

Dim 변수이름 [As 데이터유형]

변수 선언문	설 명
Dim int나이 As Integer	정수만 기억하는 변수 선언
Dim str성명 As String	텍스트를 기억하는 변수 선언
Dim rng시작 As Range	셀영역을 기억하는 변수 선언
Dim sht기초 As Worksheet	워크시트를 기억하는 변수 선언

3) VBA 데이터 형식 종류

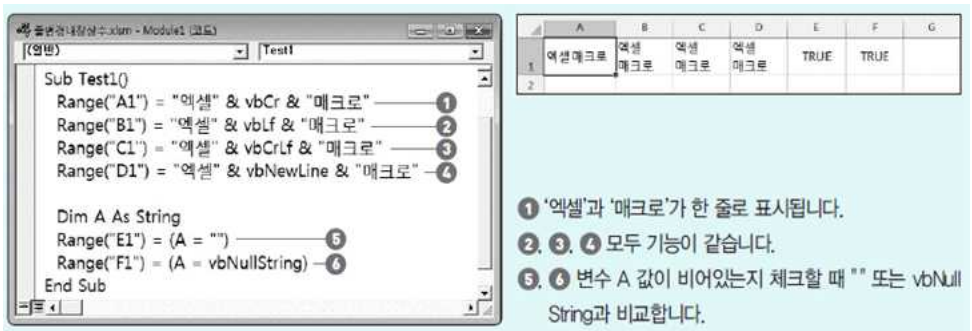
데이터 형식	저장 용량
Byte	1 바이트
Integer	2 바이트 정수
Long	4 바이트 정수
Single	4 바이트 실수
Double	8 바이트 실수
Date	8 바이트

데이터 형식	저장 용량
Currency	8바이트 고정 십진소수
String	문자열
Variant	16 바이트
Boolean	2 바이트
Object	4 바이트
사용자 정의 형식	type문으로 선언된 요소들이 가진 크기

4) 줄변경에 사용되는 내장 상수

vbCr, vbCRLF 등은 MsgBox나 셀에 내용을 입력할 때 줄을 변경하기 위해 사용하는 상수이다.

내장상수	설 명
vbCR	Carrage Return(줄변경) 기능으로, MsgBox에서는 줄이 변경되지만, 워크시트 셀에서는 줄변경 되지 않는다.
vbLF	LineFeed(줄이동)으로 일반적인 경우 vbCR과 별 차이가 없으나, 셀에서의 줄변경시에 이용된다.
vbCRLF	줄변경과 줄이동을 동시에 사용



6. MsgBox 함수

1) 형식

간단한 메시지 내용을 출력할 때 주로 사용. 사용방법에 따라 단순한 출력 용도로만 사용할 수도 있고, 몇 개의 단추를 표시한 후 선택한 단추 종류에 따라 다른 작업을 처리할 수도 있다.

메시지만 출력	MsgBox "메시지 내용" [, 버튼 종류] [,제목]
선택한 단추값 반환	변수=MsgBox("메시지 내용", 버튼 종류 [+아이콘 종류, "제목"])

2) MsgBox 버튼 종류

상수	값	설명
vbOKOnly	0	[확인] 단추만
vbOKCancel	1	[확인] [취소]
vbAbortRetryIgnore	2	[중단] [재시도] [무시]
vbYesNoCancel	3	[예] [아니오] [취소]
vbYesNo	4	[예] [아니오]
vbRetryCancel	5	[재시도] [취소]

3) MsgBox 아이콘 종류

상수	값	아이콘	설명
vbCritical	0		중대 메시지
vbQuestion	1		질의 경고
vbExclamation	2		메시지 경고
vbInformationl	3		메시지 정보

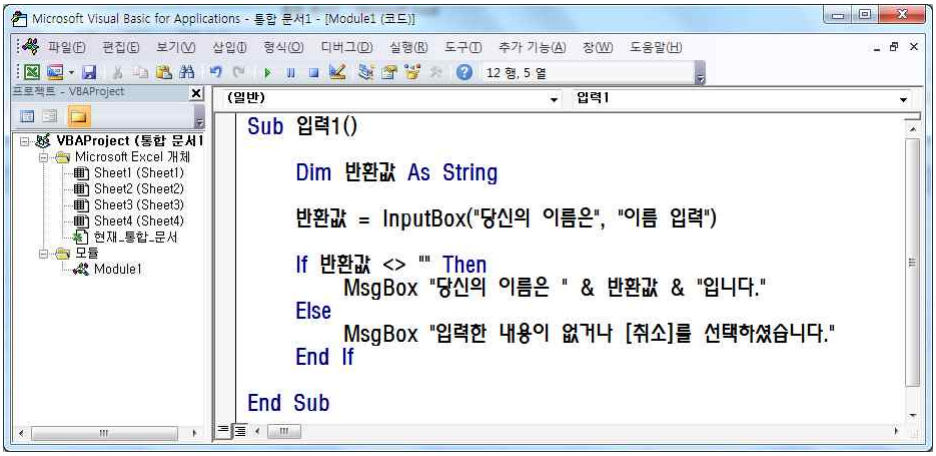
7. InputBox 함수

1) 형식

MsgBox와 반대로 데이터 하나를 입력받을 때 사용한다. 반환되는 값은 항상 문자열 형태이므로 범위를 반환받거나 날짜와 같은 특별한 형태의 자료를 반환할 때는 사용할 수 없다.

변수 = InputBox("메시지 내용" [, 제목] [, 기본값] [, 가로위치] [, 세로위치])

2) 실습 예제



8. InputBox 매서드

1) 형식

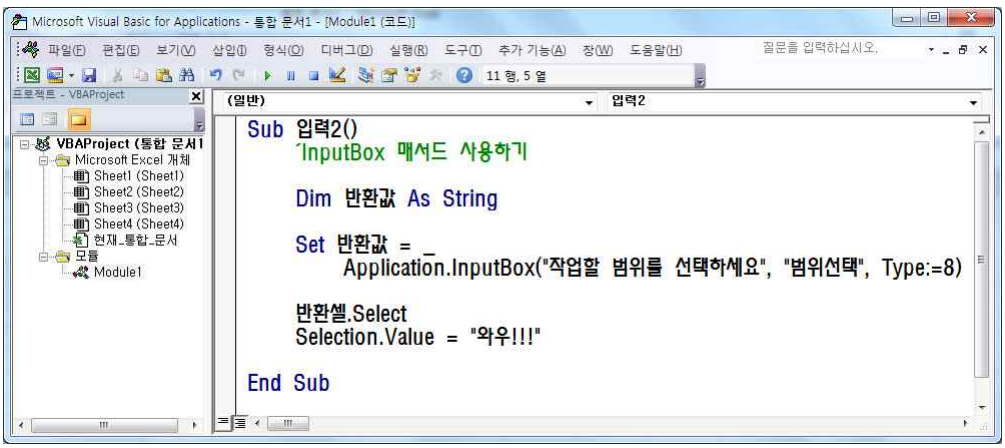
InputBox 함수는 문자열로 모든 자료를 입력받기 때문에 셀영역 등의 값은 입력 받을 수 없다. 셀 영역을 개체 형태로 입력받을 때는 반드시 Application.InputBox 매서드를 사용해야 한다. 형식과 사용법은 InputBox함수와 유사하지만 반환데이터 형식을 type이란 인수를 이용하여 지정할 수 있다.

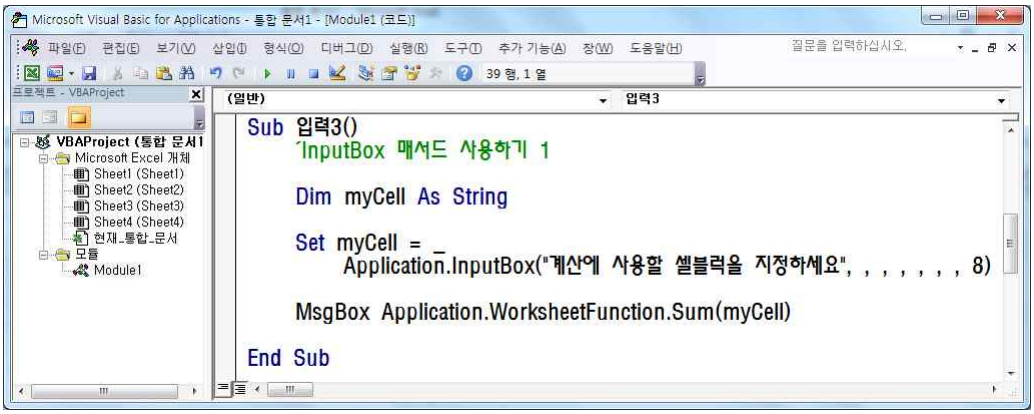
```
Application.InputBox("메시지 내용" , "제목", Type:=8 )
```

2) 사용방법

```
Dim 변수명 As Range
Set 변수 = Application.InputBox("메시지 내용" , "제목", Type:=8 )
```

3) 실습 예제





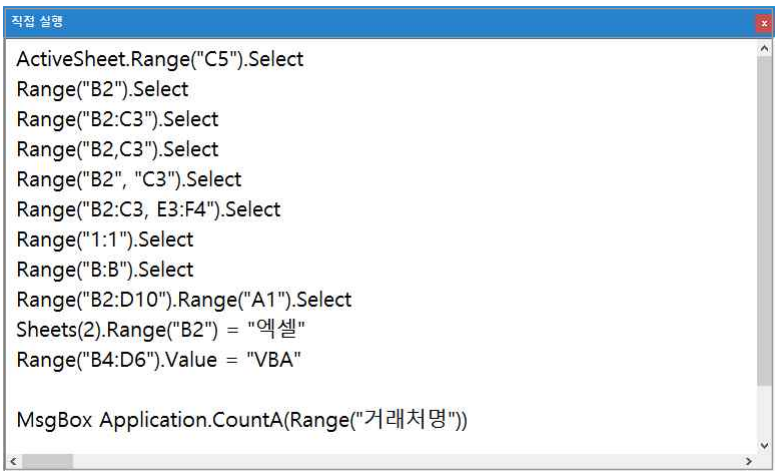
9. Range개체

1) Range 속성 – 작업 영역 지정

개체.Range(시작셀 [,종료셀])

Range 속성 셀참조예제	실행 결과
Range("A1").Select	A1 셀을 선택
Range("A1:A10").Select	A1:A10 셀 범위를 선택
Range("A1", "A10").Select	A1
Range("A1, A10").Select	A1셀과 A10셀을 선택
Range("Myrange").Select	'Myrange' 로 정의된 이름 범위를 선택

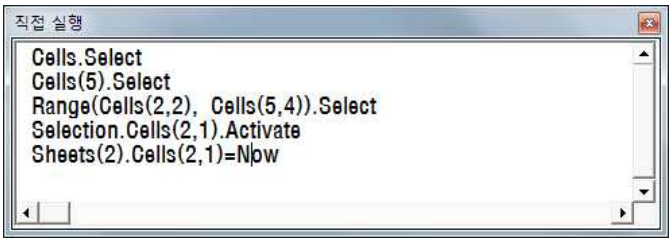
Ex) Range("A1").Value = 100
Range("A1","A10") = "Excel"



2) Cells 속성 – 행열 좌표에 의한 셀 지정

개체.Cells(행번호 [,열번호])

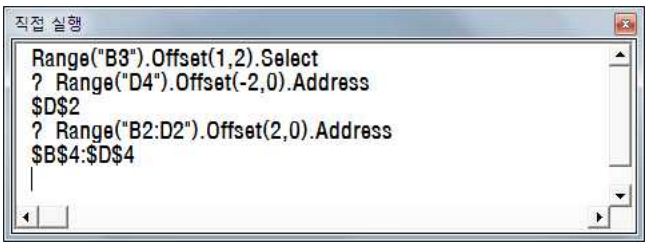
개체 영역에서 행과 열 번호에 해당하는 위치에 있는 한 셀을 반환한다. 이 속성은 숫자를 이용해 한 셀을 지정하기 때문에 For문 등과 같은 반복문에서 동적으로 셀 위치를 지정할 때 자주 사용한다.



3) Offset 속성 – 상대적 위치의 작업범위 지정

Range개체.Offset(이동행수, 이동열수)

기준 셀 영역으로부터 행과 열에 지정된 숫자만큼 상대적으로 이동한 셀 영역을 반환한다. 행과 열의 이동에 사용하는 숫자는 음수, 0, 양수 모두 사용할 수 있는데, 양수이면 행/열 방향으로, 음수이면 반대방향으로 이동한다. Cells 속성은 하나의 셀을 반환하지만, Offset속성은 기준 셀 영역을 어떻게 지정하는지에 따라 하나 이상의 영역을 반환한다.
Offset속성도 Cells 속성과 마찬가지로 숫자를 이용해 상대적인 위치를 지정하므로 For문과 같은 반복문에서 자주 사용한다.

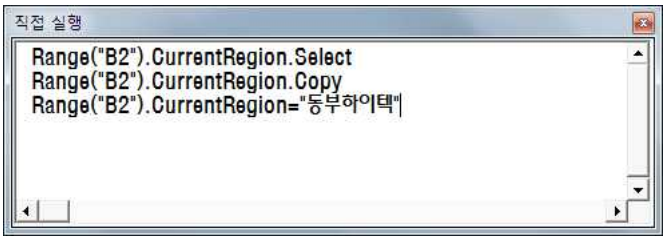


4) CurrentRegion / End 속성 – 연속영역 지정

Range개체.CurrentRegion	상하좌우 연속된 데이터 영역
Range개체.End(방향)	특정 방향으로 연속된 데이터 영역
Sheet개체.UsedRange	특정 시트에서 사용된 셀 영역

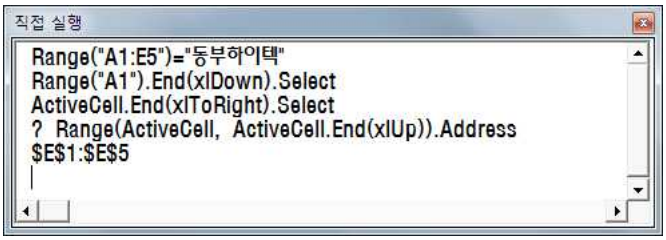
정확한 행과 열의 크기를 알지 못하는 상태에서, 현재 셀부터 데이터가 입력된 연속 셀 영역을 블록으로 설정해야하는 경우처럼 연속 셀 설정에 사용한다.

Range개체.CurrentRegion



Range개체.End(방향)

이동방향	기능
End(xlUp)	연속 데이터의 위쪽 끝 셀
End(xlDown)	연속 데이터의 아래쪽 끝 셀
End(xlToLeft)	연속 데이터의 왼쪽 끝 셀
End(xlToRight)	연속 데이터의 오른쪽 끝 셀



Sheet개체.UsedRange

5) Columns / Rows / EntireColumns / EntireRows – 행/열 단위의 작업 범위 지정

개체.Columns(열범위)	지정한 개체에서 해당 열 전체 영역 반환
개체.Rows(열범위)	지정한 개체에서 해당 행 전체 영역 반환
Range개체.EntireColumn	지정된 셀 영역을 포함하는 열 전체 반환
Range개체.EntireRow	지정된 셀 영역을 포함하는 행 전체 반환

Columns, Rows 속성

```
ActiveSheet.Columns.Select
Rows(2).Select

Columns("b:d").Select
Range("b2:e4").Rows("1:2").Select

Range("b2:e4").Rows("1").Style = "강조색1"

Range("b2:e4").Select
MsgBox Selection.Rows.Count

Sheets(2).Columns(1).ColumnWidth = 1
```

EntireColumn, EntireRow 속성

```
Range("b2:e5").EntireColumn.Select
Range("b2:e5").EntireRow.Select
Range("b2:e5").EntireColumn.Hidden = True
Range("b2:e5").EntireRow.Hidden = False
```

6) SpecialCells – 조건에 맞는 작업 영역 선택

형식 : Range개체.SpecialCells(종류, [값종류])

특정 셀 영역에서 빈 셀, 또는 수식을 입력한 셀만을 대상으로 작업해야 하는 경우 [홈] 탭 - [편집] 그룹 - [찾기 및 선택] - [이동 옵션]을 선택, [이동 옵션] 대화상자를 열고 해당기능을 처리하는데 VBA에서는 SpecialCells 매서드를 이용한다.

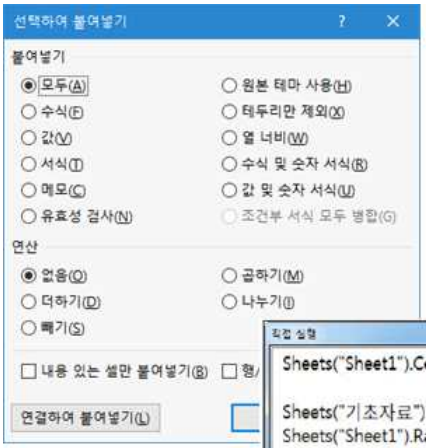
[이동 옵션] 대화상자의 '종류'		예약어(내장 상수)
메모		xlCellTypeComments
상수		xlCellTypeConstants
수식		**xlCellTypeFormulas**
빈 셀		**xlCellTypeBlanks**
마지막 데이터 셀		xlCellTypeLastCell
화면에 보이는 셀만		**xlCellTypeVisible**
조건부 서식	모두	xlCellTypeAllFormatConditions
	조건 일치	xlCellTypeSameFormatConditions
데이터 유효성	모두	xlCellTypeAllValidation
	조건 일치	xlCellTypeSameValidation

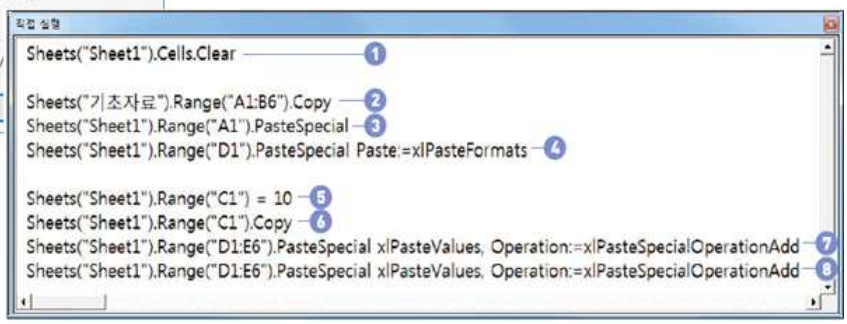
7) Value, Text, Formula 속성

Value	셀 내용을 지정하거나 표시
Text	셀 내용을 셀 서식이 적용된 형태의 텍스트로 표시
Formula	셀 내용을 셀 서식이 적용된 형태의 텍스트로 표시
FormulaR1C1	Formula 속성과 기능이 같고 수식 지정시 Row 번호와 Column 번호로 셀 주소 지정

8) PasteSpecial 메서드 – 선택하여 붙여넣기

Range개체.PasteSpecial (Paste, Operation, SkipBlanks, Transpose)





Paste 상수	값	설명
xlPasteAll	–4104	모두(기본값)
xlPasteAllExceptBorders	7	테두리만 제외
xlPasteAllMergingConditionalFormats	14	조건부 서식 모두 병합
xlPasteAllUsingSourceTheme	13	원본 테마 사용
xlPasteColumnWidths	8	열 너비
xlPasteComments	–4144	메모
xlPasteFormats	–4122	서식
xlPasteFormulas	–4123	수식
xlPasteFormulasAndNumberFormats	11	수식 및 숫자 서식
xlPasteValidation	6	유효성 검사
xlPasteValues	–4163	값
xlPasteValuesAndNumberFormats	12	값 및 숫자 서식

9) 기타 속성

ActiveCell	ActiveCell.Font.Bold = True
Address / AddressLocal	셀 주소를 반환
Areas	비연속적으로 선택된 전체 범위를 반환
Borders	네 개의 셀 테두리를 반환
ColumnWidth	열의 너비를 반환 및 설정
Dependents	셀이 참조되고 있는 셀 범위를 반환
DirectDependents	셀이 직접 참조되고 있는 셀 범위를 반환
DirectPrecedents	셀이 직접 참조하고 있는 모든 셀 범위를 반환
EditDirectlyInCell	셀을 더블클릭하여 편집 못하도록 설정
FormulaArray	배열 수식을 반환 및 설정
FormulaHidden	수식 입력줄에 수식이 나타나지 않도록 설정
FormulaLabel	수식 레이블을 반환 및 설정
HasFormula	셀에 수식이 있는지 조사
Height / Width	셀의 높이와 너비를 반환
Hidden	행과 열의 숨기기를 설정
HorizontalAlignment VerticalAlignment	문자열의 맞춤을 설정
Interior	셀의 내부를 참조(셀 음영색 등)
MergeArea	병합된 셀 범위를 참조
MergeCells	셀 범위의 병합 및 병합 취소 설정
Name	셀의 이름을 반환 및 설정
NumberFormat	셀의 표시형식을 반환 및 설정
Precedents	셀이 참조하고 있는 모든 셀 범위를 반환
Previous / Next	앞쪽과 뒤쪽 셀을 참조
ReferenceStyle	A1 참조형식과 R1C1 참조형식을 반환 및 설정
Row / Column	셀 범위의 최초 행 번호, 열 번호를 반환
RowHeight	행의 높이를 반환 및 설정
ShrinkToFit	문자 크기를 열 너비로 축소
Text	셀의 문자열을 반환
UsedRange	사용하는 셀 범위를 반환
Value / Value2	셀 값을 반환 및 설정 Value2에는 통화 및 날짜형식을 사용할 수 없다.
WrapText	문자 크기를 열 너비에 맞춤

10) Range개체 메서드

Activate	셀을 활성화
AddCommnet	범위에 메모를 추가
AdvancedFilter	고급 필터하기
AutoFill	자동 채우기 설정
AutoFilter	자동필터 하기
AutoFit	지정 범위내의 열너비, 행높이를 가장 알맞게 조정
Clear	셀 범위의 모든 것 지우기
ClearComments	셀 범위의 메모 지우기
ClearContents	셀 범위 내의 내용 및 수식 지우기
ClearFormats	셀 범위 내의 서식지우기
ClearNotes	셀 범위에서 메모와 소리 지우기
ClearOutline	셀 범위에서 테두리 지우기
Delete	셀 삭제
Insert	빈 셀을 삽입
Paste	셀을 붙여넣기
PasteSpecial	선택하여 붙여넣기
Select	셀을 선택

11) 색 지정하기

속성(매서드)		형 식	기능
Color 속성	**내장상수**	개체.Color = 색번호	**내장상수, RGB 함수, QBColor 함수**
	RGB 함수		
	QBColor 함수		
ColorIndex 속성		개체.ColorIndex = 색번호	1~56까지의 번호 이용
ThemeColor 속성		개체.ThemeColor = 색번호	테마에 따라 테마색 이용

방법 1 개체.Color = ColorConstants.색

▲ 내장 상수 목록

방법 2 개체.Color = RGB(red, green, blue)

▲ RGB 함수 색 번호

방법 3 개체.Color = QBColor(color index)

▲ QBColor 함수 색 번호

ColorIndex 속성을 이용한 색

번호	색상	실제색	번호	색상	실제색	번호	색상	실제색
1	검정		21	진한자주		41	연한파랑	
2	흰색		22	산호색		42	바다색	
3	빨강		23	바다색		43	라임	
4	온녹색		24	담청색		44	황금색	
5	파랑		25	진한파랑		45	연한주황	
6	노랑		26	분홍		46	주황	
7	분홍		27	노랑		47	청회색	
8	옥색		28	밝은옥색		48	회색40%	
9	진한빨강		29	보라		49	진한청록	
10	녹색		30	진한빨강		50	해록	
11	진한파랑		31	진한청록		51	진한녹색	
12	진한노랑		32	파랑		52	황록색	
13	보라		33	하늘색		53	밤색	
14	청록		34	연한옥색		54	진한보라	
15	회색25%		35	연한녹색		55	남색	
16	회색50%		36	연한노랑		56	회색80%	
17	빙카색		37	흐린파랑				
18	자주색		38	다홍				
19	살아색		39	연한보라				
20	연한옥색		40	황갈색				

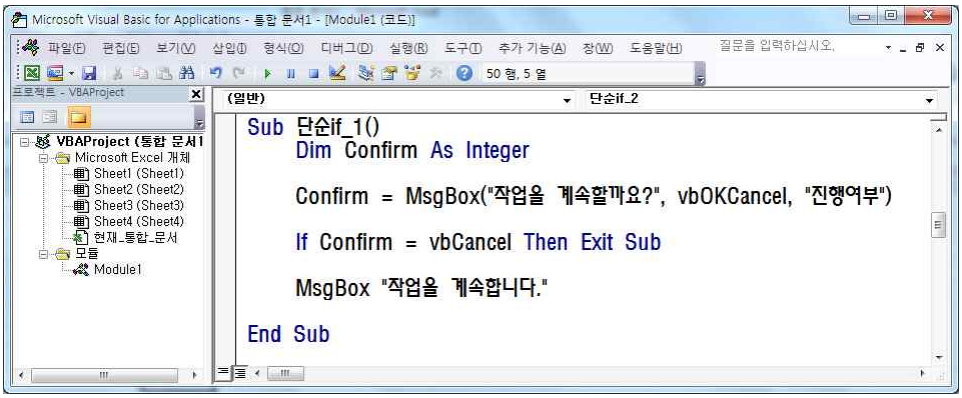
▲ ColorIndex 속성에 사용하는 Index(번호)별 색 목록

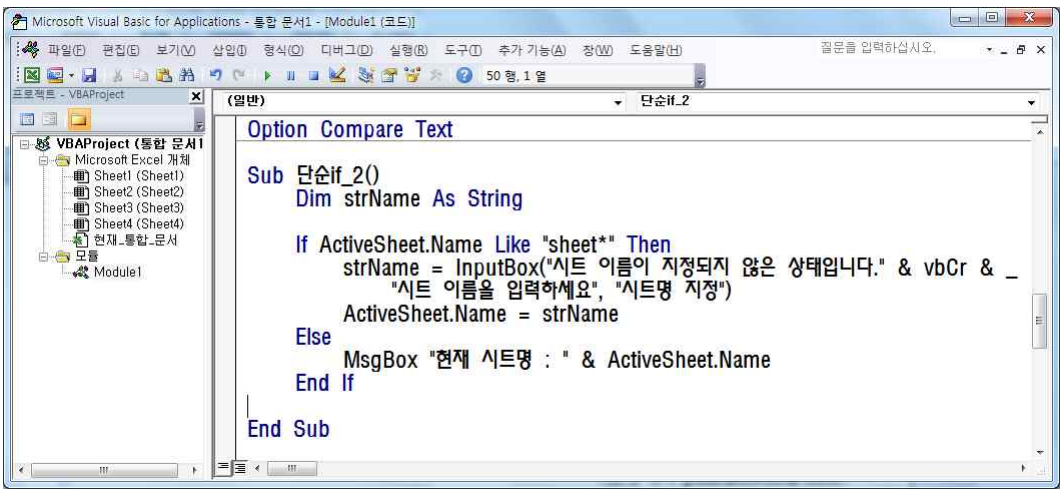
10. VBA 제어문

1) 단순if

조건을 만족할 때만 작업실행	조건을 만족할 때와 그렇지 않은 때 나눠서 처리

If 조건식 Then
실행문
End If
	If 조건식 Then
	조건을 만족할 때 실행문
	Else
	조건을 만족하지 않을 때 실행문
	End If

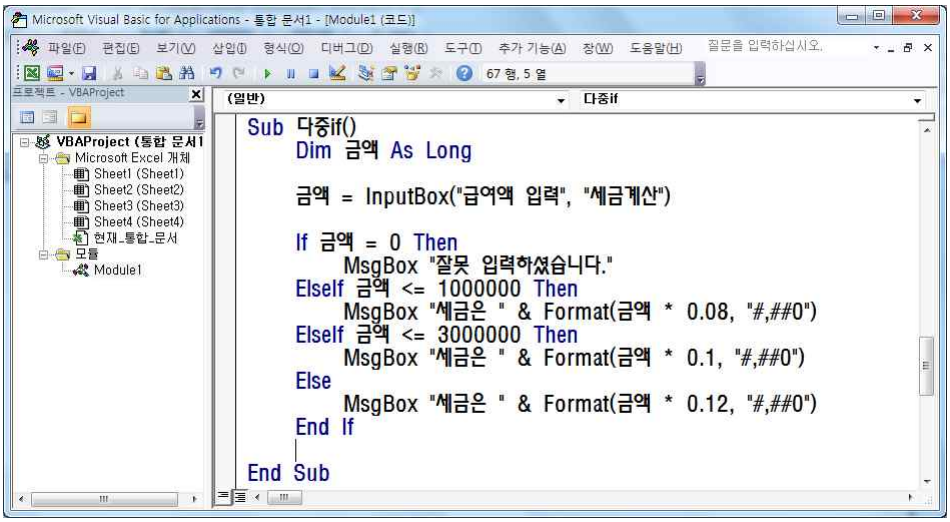




2) 다중 If

조건에 따라 세가지 이상의 처리방법으로 처리

```
If 조건식1 Then
    조건1을 만족할 때 실행문
ElseIf
    조건2를 만족하지 않을 때 실행문
[ElseIf 문 반복]
Else
    조건1,2를 모두 만족하지 않을 때 실행문
End If
```



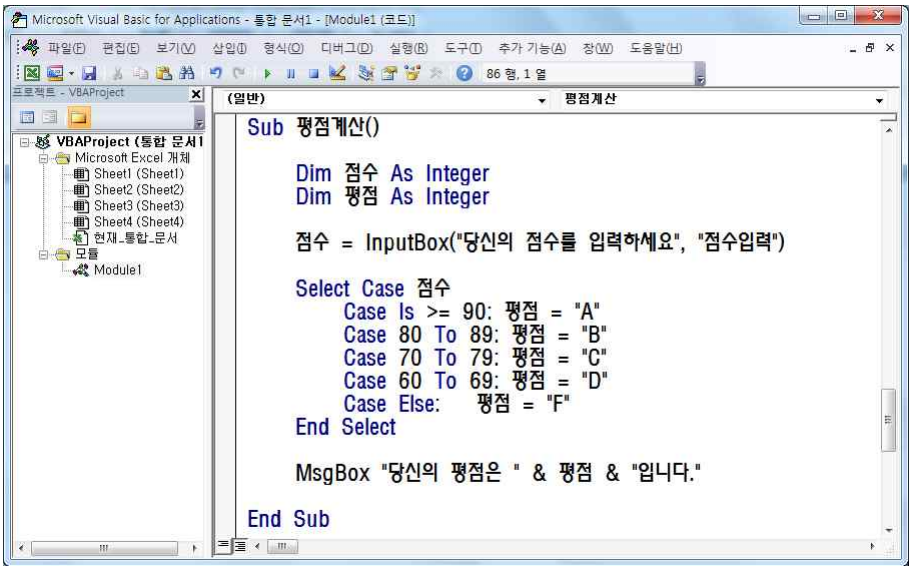
3) And/Or 사용하여 여러 조건 체크하기

```
조건1 And 조건2 ....
조건1 Or 조건2 ....
```

If문에서 조건을 지정할 때 여러 가지 조건을 함께 확인해야 할 경우에 사용

4) Select Case문으로 다중조건 처리하기

```
Select Case 식(변수)
    Case 값1
        실행문1
    [Case 값2
        실행문2
     CaseElse
        실행문n]
End Select
```

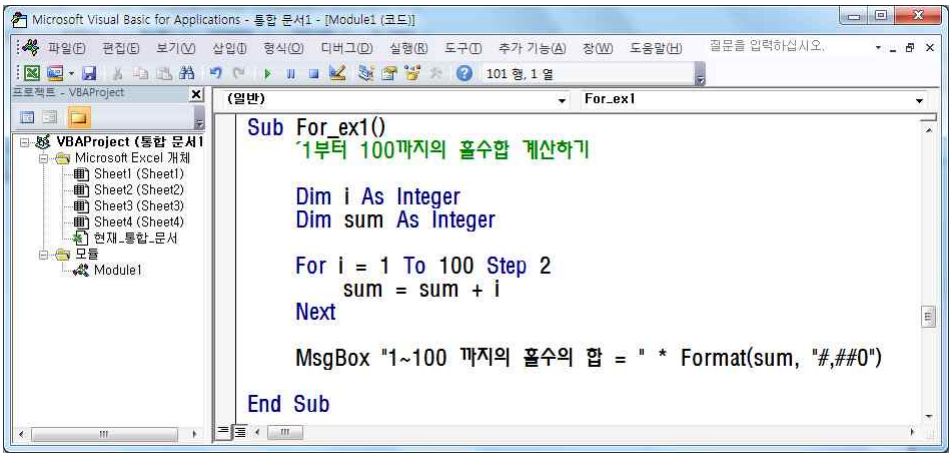


11. VBA 반복문

1) For문

```
For 카운트변수 = 초기치 To 최종치 [Step 증감치]
    실행문
Next [카운트변수]
```

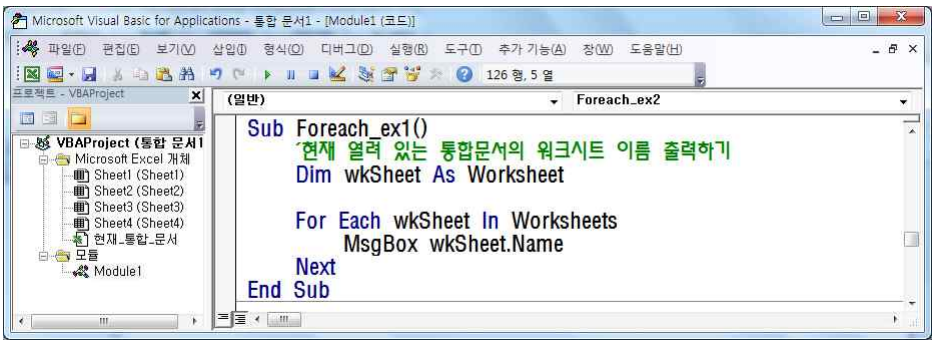
정확한 반복 횟수를 알 경우 사용

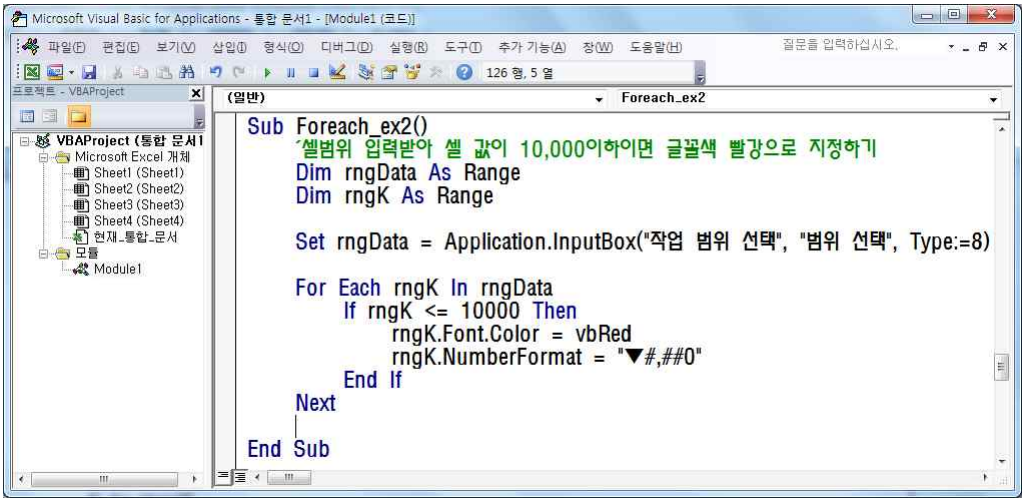


2) For Each문

셀영역이나 여러 워크시트, 여러 통합 문서처럼 개체들의 집합(컬렉션)을 대상으로 한 개씩 개별 개체에 대한 방복 작업을 처리할 때 사용한다. For Each문은 개체를 대상으로 반복하기 때문에 개체 변수는 반드시 컬렉션 개체와 동일한 데이터 형식으로 지정해야 한다.

```
For Each 개체변수 In 컬렉션개체
    실행문
Next
```





3) Do While문

여러 번 반복 처리할 때 반복할 횟수를 알고 있다면 For문을 사용하지만, 반복 횟수를 알 수 없고 특정 조건을 만족허가나 반대로 만족하지 않을 때까지 반복해야 하는 경우에는 Do문을 사용 Do While문은 조건식의 결과가 True 값을 가지는 동안만 반복한다.

```
Do While 조건문
    실행문
Loop
```

4) Do Until문

Do While문이 조건식의 결과가 True값을 가지는 동안만 반복하는 경우에 반해 Do Until은 조건을 만족할 때까지 반복한다. 즉, 조건식의 결과가 False 값을 가지는 동안만 반복한다.

```
Do Until 조건문
    실행문
Loop
```

12. 시트 데이터 통합하기

1) 머리글 항목을 복사하는 프로시저

```
Option Explicit

Dim SumSheet As Worksheet

'머리글 항목 복사
Private Sub 머리글복사()

    Set SumSheet = Worksheets(1)

    Worksheets(2).Range("a2", Worksheets(2).Range("a2").End(xlToRight)).Copy
    SumSheet.Range("b2").PasteSpecial
    SumSheet.Range("a2") = "월구분"
    Application.CutCopyMode = False

End Sub
```

2) 각 시트별 내용 복사 프로시저

```
'내용을 복사하는 프로시저
Private Sub 내용복사()

    Set SumSheet = Worksheets(1)
    Dim iRow As Integer
    Dim iSheet As Integer

    For iSheet = 2 To Worksheets.Count
        With Worksheets(iSheet)
            iRow = SumSheet.Range("A2").CurrentRegion.Rows.Count + 1
            SumSheet.Cells(iRow, 1) = .Name
            .Range(.Range("a3").End(xlDown), .Range("a3").End(xlToRight)).Copy
            SumSheet.Cells(iRow, 2).PasteSpecial
        End With
    Next

    Application.CutCopyMode = False

End Sub
```

3) 월 채우기 프로시저

```
'A열에 월 채우기
Private Sub 월채우기()

    Set SumSheet = Worksheets(1)
    Dim strMon As String

    SumSheet.Range("a3").Select

    Do While ActiveCell.Offset(0, 1) <> ""
        If ActiveCell <> "" Then
            strMon = ActiveCell
        Else
            ActiveCell = strMon
        End If
        ActiveCell.Offset(1, 0).Select
    Loop

End Sub
```

4) 세 개의 프로시저를 순서대로 실행하는 메인 프로시저

```
Sub 시트데이터통합()
    Application.ScreenUpdating = False

    Call 머리글복사
    Call 내용복사
    Call 월채우기

    Application.ScreenUpdating = True

End Sub
```

13. 여러 파일 데이터 통합하기

1) 프로그램에서 파일 열기

```
Option Explicit

Dim SumSheet As Worksheet

Sub 파일데이터모으기()

    Dim fileNo As Variant
    Dim i As Integer
    Dim wb As Workbook
    Dim iRow As Integer

    Set SumSheet = ThisWorkbook.Worksheets(1)

    On Error GoTo 오류처리

    '오픈할 파일이름 얻어오기
    fileNo = Application.GetOpenFilename(filefilter:="엑셀파일(*.xls*), *.xls*", _
        MultiSelect:=True)

    Application.ScreenUpdating = False
    Application.DisplayAlerts = False

    '파일을 열고 내용 복사
    For i = 1 To UBound(fileNo)
        Set wb = Workbooks.Open(Filename:=fileNo(i), ReadOnly:=True)
        iRow = sumsheet.Range("a3").CurrentRegion.Rows.Count + 3
        SumSheet.Cells(iRow, 1) = wb.Sheets(1).Name

        With wb.Sheets(1)
            .Range(.Range("a3").End(xlDown), .Range("a3").End(xlToRight)).Copy
        End With

        SumSheet.Cells(iRow, 2).PasteSpecial

        wb.Close
    Next

    Call 월채우기
    Application.ScreenUpdating = True
    Application.DisplayAlerts = True

    Exit Sub
    오류처리:
    MsgBox "파일을 선택하지 않았습니다."

End Sub
```

```
Private Sub 월채우기()

    Set SumSheet = Worksheets(1)
    Dim strMon As String

    SumSheet.Range("a3").Select

    Do While ActiveCell.Offset(0, 1) <> ""
        If ActiveCell <> "" Then
            strMon = ActiveCell
        Else
            ActiveCell = strMon
        End If
        ActiveCell.Offset(1, 0).Select
    Loop

End Sub
```