2. LPI 개요

2.1. LPI란

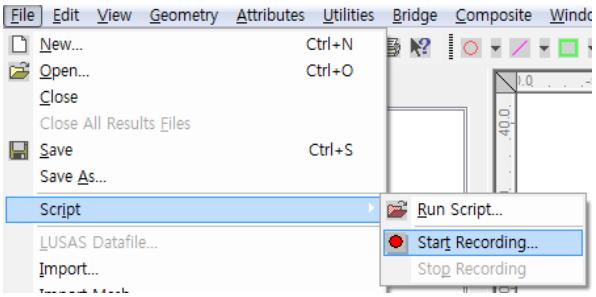
LUSAS는 모델러에서 사용자가 작업하는 모든 내용을 Visual Basic Script를 사용하여 표현할 수 있도록 하였으며, 또한 사용자가 작성한 Visual Basic Script를 모델러에서 실행시킬 수 있도록 한 것입니다. 따라서 사용자가 모델러에서 사용하는 명령문을 이해하여 프로그램을 작성하면, 모델링과 해석, 결과출력의 모든 과정을 처리할 수 있게 됩니다.

이렇게 모델러에서 수행하는 모든 작업을 Visual Basic Script 명령문으로 내보내거나, 사용자가 작성한 VBScript 프로그램을 실행하는 기능을 통틀어 LPI (LUSAS Programmable Interface)라고 합니다.

2.2. LPI 실행 테스트

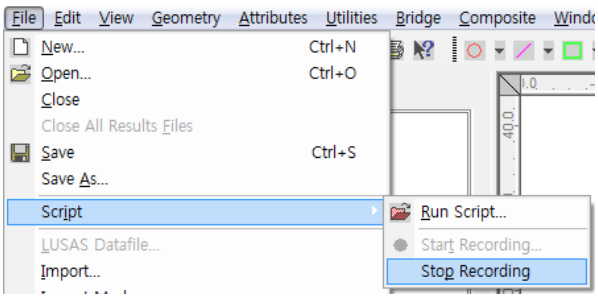
다음을 이용하여 LUSAS 모델러에서 작업한 내용을 스크립트 형태로 기록할 수 있습니다.

1) 다음의 메뉴를 실행하여 스크립트 기록모드를 시작합니다.
(File / Script / Start recording..)



2) 작업을 수행합니다.

3) 스크립트 기록모드를 종료합니다.
(File / Script / Stop recording..)



4) 작업폴더에서 저장된 스크립트 내용을 확인하고,

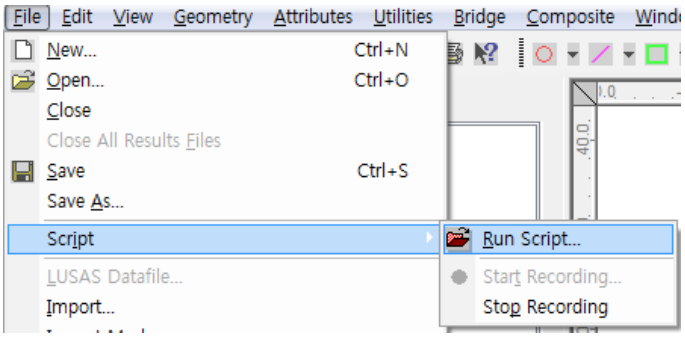
```
$ENGINE=VBScript

'*** Create Line
Call geometryData.setAllDefaults()
call geometryData.setCreateMethod("straight")
call geometryData.addCoords(0.0, 0.0, 0.0)
call geometryData.addCoords(10.0, 0.0, 0.0)
call geometryData.setLowerOrderGeometryType("coordinates")
call database.createLine(geometryData)

'*** Modify zoom/rotation
Call view.setScaledToFit(true)
```

일부 내용을 변수로 지정하여 저장합니다.

```
$ENGINE=VBScript

For i=1 to 10

        Ycoord = i*10
        '*** Create Line
        Call geometryData.setAllDefaults()
        call geometryData.setCreateMethod("straight")
        call geometryData.addCoords(0.0, Ycoord, 0.0)
        call geometryData.addCoords(10.0, Ycoord, 0.0)
        call geometryData.setLowerOrderGeometryType("coordinates")
        call database.createLine(geometryData)

Next

'*** Modify zoom/rotation
Call view.setScaledToFit(true)
```

5) 모델러를 초기화 하고, 기록된 스크립트를 실행합니다.

(File / Script / Run Script...., 또는  아이콘 클릭하여 스크립트 실행)



이후, 수행하는 스크립트 예제 역시 모델러에서 위와 동일한 방법으로 스크립트를 실행하여 테스트 합니다.

2.3. LPI 활용 방법

2.3.1. [매크로] 로 활용

가장 보편적으로는 모델링 또는 결과처리 과정에서 반복되는 작업을 단축시키는데 사용합니다.

- 매번 사용하는 기본적인 기하특성/ 재료특성 등을 *.vbs 파일로 저장해 두었다가 모델링 전에 실행
- 하중케이스를 바꿔가며 수십 · 수백 장의 그림파일을 만들어야 할 때 매크로로 지정
- 필요한 결과값들 만을 추출하여 Excel 파일로 일괄 저장시키는 경우

2.3.2. [자동화] 툴 제작

유사한 형태의 다양한 모델링을 자주 반복할 때에는 몇 가지 제원 입력으로 자동모델링이 되도록 할 수 있습니다.

- 돔 구조 최적화 (삼성물산건설, 2009)
  - 4가지 타입의 돔 구조 자동 모델링
  - 단면 최적화 (Excel에 입력한 단면제원 내에서 반복 수행)
  - 주요 검토 결과 보고서 출력

2.4. LPI 특징

2.4.1. 편의성

사용자가 작업하는 내용을 VBScript 명령문으로 확인한 후 필요한 부분에 대해 변수 처리하여 수정하여 프로그래밍을 작성하면 되므로, 초보자도 간단한 목적의 매크로에 쉽게 사용할 수 있습니다.

2.4.2. 대화창 구성

조금 복잡한 프로그램이나 매크로의 경우, 대화창도 구성할 수 있습니다.

2.4.3. 모델러 메뉴 등록

자주 사용하는 매크로나 프로그램인 경우에는 모델러 메인메뉴에 등록하여 나만의 모델러 환경을 구성하여 사용할 수 있습니다.

2.5. LPI 기본 구조

```
$ENGINE=VBSCRIPT                          : Script engine선언
...                                       : 프로그래밍
call subprogram(a,b,c)                    : 부프로그램 호출
a=subFunction(d,e,f)                      : 사용자 정의 함수 사용
...                                       : 프로그래밍
---------------------------
sub subprogram(x,y,z)                     : 부프로그램 내용
                    ...
End sub
---------------------------
Function subFunction(x,y,z)               : 사용자 함수 정의
                    ...
End Function
---------------------------
```

Scripts 는 프로그래밍 언어에서 제공되는 것으로 LUSAS는 아래의 Script Engine을 수용합니다.

- VBSCRIPT : Microsoft Visual Basic Scripting Edition (v3.1)
- JVSCRIPT : Microsoft Java Scripting Edition (v3.1)

☞ 따라서, Lusas Manual에 표기되지 않은 Visual Basic Scripts들도 프로그래밍에 사용할 수 있습니다.

2.6. 적용 규칙

▶ 변수는 사전 정의될 필요는 없으며, 변수가 사용되는 경우에 따라 문자변수가 되거나 실수변수가 되기도 합니다. 변수의 사용은 편리하지만 연산작업을 수행할 경우 문자연산으로 처리될 수 있는 등 주의가 필요합니다.

▶ 배열은 사용 전에 정의되어야 합니다.
예) dim a(10,10), redim array(10,20) 등

▶ 대소문자는 구별되지 않습니다.

▶ 명령어 외 개발자가 참조하기 위한 참조문은 문장 앞에 (')를 기록하여 구분합니다.

▶ VBscript에서 사용하는 모든 함수를 사용할 수 있습니다. 대표적인 연산자로 +, -, *, /, ^, sqr(실수) 등을 사용합니다.

▶ 삼각함수에는 Radian이 사용됩니다.

3. 변수

3.1. 변수의 종류

정수/ 실수/ 문자 변수의 구분 없이 사용방법에 따라 가변적으로 활용할 수 있습니다.
즉 지역변수는 부프로그램이나 사용자 함수 내에서 정의되며, 주프로그램에서 정의된 광역변수와는 완전히 별개로 인식되고 해당 부프로그램이나 함수정의문 내에서만 유효하게 됩니다.

3.2. 변수 정의

```
dim anyname
dim anyname1, anyname2, ......
```

☞ 알파벳 문자로 시작해야 하며, 포인트(.)는 사용할 수 없고, 변수 이름의 길이는 255자 이내여야 합니다.
대소문자는 구분하지 않습니다.

☞ 변수 종류의 뚜렷한 구분을 두고 있지 않기 때문에 주의하지 않으면 수 연산을 해야 하는데 문자 연산을 해버리는 경우가 발생합니다. 예를 들어 1+1의 결과로 2를 기대하지만 11이 되어버리는 경우입니다. 이를 방지하기 위해서 CInt, CStr, CdbI를 연산 작업이 되기 전에 설정하여 오류를 방지할 수 있을 것입니다.

☞ CInt(10.555) - 소수점 첫째자리에서 반올림. 출력값 : 11
Int(10.555) - 소수자리 버림. 출력값 : 10
Round(10.555,2) - 표시할 소수점 자리수 입력(여기서는 둘째자리까지 표시하도록 2 입력).
소수점 셋째 자리에서 반올림. 출력값 : 10.56
CStr(1)+CStr(2)= 12 – 숫자를 문자로 출력할 때 사용
CdbI(10.333) - 실수를 정의할 때 사용. 출력값 : 10.333

3.3. 배열 변수 정의

배열 변수는 한 가지 이름에 Index를 두어 여러 가지의 값을 저장하게 하는 변수입니다.

■ 정적 배열의 정의

```
dim variable(4)
```

위와 같이 변수 뒤에 Index를 추가하여 정의하며, 이 경우 variable(0), variable(1), variable(2), variable(3), variable(4)의 5개 변수를 한 번에 정의한 것과 같습니다.

주로 정의하고자 하는 배열의 개수가 확실한 경우에 사용하며, 정의한 변수에 따른 저장 공간을 할당하기 때문에 배열의 개수가 정확하지 않은 상태에서 초기에 너무 많은 변수를 정의하지 않는 것이 좋습니다.

■ 동적 배열의 정의

dim variable()

처음에 몇 개의 배열을 만들어야 하는지를 모르는 경우에 사용하며, 나중에 배열의 개수가 정해져서 실제로 사용하게 되었을 경우에 아래와 같이 재 지정하여 사용합니다.

redim variable(4)

☞ 동적 배열 사용 예: 부프로그램이나 함수를 정의할 때 전달할 변수가 필요한데, 그 변수는 부프로그램 안에서 개수가 정해질 경우가 있습니다. 이럴 경우 우선 동적 배열을 정의해서 부프로그램으로 변수명을 넘기고 실제 사용할 변수의 정의는 부프로그램에서 실시하게 합니다.

☞ 단, redim 으로 동적 배열을 재정의하게 되면, 이전 배열에 저장된 값들은 삭제가 됩니다. 이전 값을 사용해야 할 필요가 있을 경우에는 아래와 같이 preserve 키워드를 포함시키면 됩니다.

redim preserve variable(4)

```
$ENGINE=VBSCRIPT
redim a(2)
a(0) = 1
a(1) = 2
a(2) = 3
redim preserve a(3)
a(3) = 4

msgbox a(0)  -> 출력값 1
msgbox a(3)  -> 출력값 4
```

4. 제어문

4.1. FOR 문

■ 형식

```
for 변수=시작수 to 끝수 step 변화간격
  변수를 활용한 명령문
   [exit for]
  명령문들
next
```

4.2. DO ... LOOP 문

■ 형식 1

```
do while (판단문)
  명령문들
  [exit do]
  명령문들
loop
```

■ 형식 2

```
do
 명령문들
  [exit do]
 명령문들
loop while (판단문)
```

4.3. IF 문

■ 형식 1

```
if (판단문1) then (실행문) [ else (실행문) ]
```

■ 형식 2

```
if (판단문1)
   명령문들
[ elseif (판단문2)]
   명령문들
[ else ]
   명령문들
end if
```

☞ [ ]안 내용은 필요시에만 사용

■ 판단문에 사용되는 비교 연산자

=, 〈〉, 〉, 〈, 〉=, 〈=

☞ 〈〉 : ≠ 같지 않음을 의미

■ 판단문의 연결에 사용되는 논리 연산자

and, or, not, xor, eqv

(논리합, 논리곱, 부정, 제외, 같음)

☞ xor : 배타적 논리값, 참과 거짓이 섞여 있어야만 참으로 인정

■ 산술 연산자

+, -, *, /, ·, mod, &

☞ mod : 어떤 수를 다른 수로 나눈 나머지 값 출력

7 mod 3 : 출력값 1

4.4. SELECT CASE 문

■ 형식

```
select case (변수)
  case 1
    명령문들
  case 2
    명령문들
      ....
  case else
    명령문들
end select
```

5. Object 와 Method 의 이해

5.1. Object 와 Method의 개념

■ Object

각종 속성을 가지고 있는 대상을 Objects라고 하며, 아래와 같이 Objects들을 설정하여 활용할 수 있습니다.

set test1=getMainMenu() : 메인 메뉴의 속성을 가진 test1라는 Object를 정의

■ Method

위에서 test1 과 같은 것이 'Object' 가 되며, 이 Object를 생성하기 위해 사용되는 getMainMenu()와 같은 것은 'Method' 라고 하며, 속성을 확인하거나 변경할 때 사용합니다.

☞ Object와 Method 의 이해를 위해

```
$ENGINE=VBSCRIPT
set database=getDatabase()
set Point1=database.getPoint(20)
Xcoord = point1.getX()
Ycoord = point1.getY()
```

위는 20번 Point을 Point1이라는 Object로 칭한다는 의미입니다.
즉, Point1은 20번이라는 번호를 가지고 있으며, 어떤 좌표를 가지고 있으며, 어느 Line의 일부로서 정의되어 있는가 하는 등의 모든 정보(속성)을 포함하는 개체에 부여되는 이름이 되는 것입니다. 이 과정에서 사용되는 getDatabase(), getPoint(20), getX()는 모두 'Method' 라고 칭합니다.

☞ 또한 20번 Point의 좌표값들을 얻어내기 위해 순차적으로 Object를 정의해야 하는 것을 알 수 있습니다.
즉, 모델링 하고 있는 모델 전체를 의미하는 'database' Object를 정의했고, 이 Object의 일부분인 'Point1' 이라는 Object를 다시 정의하였습니다.

☞ Object를 정의할 때는 'Set' 을 사용합니다.

5.2. 상위 Object 정의 없이 사용될 수 있는 Method (최상위 Object)

Method 서식	산출물 / 결과	비 고
getMainMenu()	메뉴전반 Object	
getDatabase()	데이터베이스 Object	작업 중인 모델링
newDatabase()	새 데이터베이스 Object	초기화 작업 실시
getSelection()	선택되어 있는 모든 Object	Script 실행 전 선택된 것
processCommand(Command$)	-	Command 실행
createObject(Object$)	자동화 Object	엑셀 등과 데이터 교환
createSimpleDialog('생략')	True / False	대화창. OK 입력시 True
getTextWindow()	-	텍스트 출력창 자체 의미

☞ $는 "" 를 포함한 문자열 또는 문자변수를 입력해야 함을 의미

5.2.1. getMainMenu()

- 예: set MainMenu=getMainMenu()
- 내용 : Modeller의 메뉴 구성에 대한 모든 내용을 포함하는 MainMenu라는 이름의 Object를 정의합니다.

5.2.2. getDatabase()

- 예: set database=getDatabase()
- 내용 : Modeller에서 작업되고 있는 전체 내용을 포함하는 database라는 이름의 Object를 정의합니다.

5.2.3. newDatabase()

- 예: set database=newDatabase()
- 내용 : Modeller를 초기화시키고, Modeller에서 작업되고 있는 전체 내용을 포함하는 database라는 이름의 Object를 정의합니다.

5.2.4. getSelection()

- 예: set selected=getSelection()
- 내용 : 현재 명령이 실행되기전에 Modeller에서 마우스 혹은 기타의 방법으로 선택된 Geometry나 Mesh등이 있다면, 이들의 속성을 모두 가지는 selected 라는 이름의 Object를 정의합니다.

5.2.5. CreateObject(Object$)

- 예: set fileSys = CreateObject("Scripting.FileSystemObject")
- 내용 : Windows에서 사용되는 디렉토리 구조의 내용을 포함하는 fileSys라는 이름의 Object를 정의합니다.

- 예: set Set excelsheet=CreateObject("excel.sheet")

■ 내용 : Excel의 Sheet 1개를 생성하고, 이 Sheet의 모든 속성을 포함하는 excelsheet라는 이름의 Object를 정의합니다. 이와 관련한 내용도 파일 입출력 관련하여 나중에 다시 다루겠습니다.

5.2.6. createSimpleDialog (....)

대화창을 구성하는 Object로 별도로 다루기로 하고 여기서는 자세한 설명을 생략합니다.

5.2.7. getTextWindow()

■ 예: set textwindow=getTextWindow()

■ 내용 : Modeller의 텍스트 출력창을 가리키는 textwindow라는 이름의 Object를 정의합니다. Modeller내에 text window에 메시지를 출력하거나 출력된 메시지를 읽고자 할 때 사용합니다.

5.3. TextWindow (Object)

추후 예제를 다루어가면서 프로그램 오류 여부를 확인하는 방법으로 활용될 몇 가지 Object와 Method를 먼저 다루겠습니다.

5.3.1. Method의 종류

텍스트 화면, 즉 Modeller의 Text Window에 문자열을 출력함으로써 필요한 결과나 에러메시지를 올릴 수 있으며, 그 내용을 읽어 프로그램에서 참조케 할 수 있습니다.

Method 서식	산출물 / 결과
writeLine(문자열$)	문자열 출력
readLine(행번호)	지정한 행 번호의 텍스트 Object
countLines()	문서내 행 수

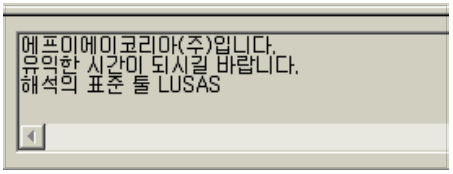
5.3.2. 예제

■ 스크립트 예

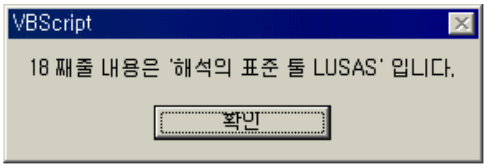
행번	스크립트
1	$ENGINE=VBSCRIPT
2	Set textwindow = gettextwindow()
3	textwindow.writeline("안녕하세요")
4	textwindow.writeline("에프이에이코리아(주)입니다.")
5	textwindow.writeline("유익한 시간이 되시길 바랍니다.")
6	textwindow.writeline("해석의 표준 툴 LUSAS")
7	
8	m=textwindow.countLines()
9	For i=m-1 to m-4 step -1
10	text=i &" 째줄 내용은 '" & textwindow.readLine(i)&" ' 입니다."
11	msgbox text
12	Next

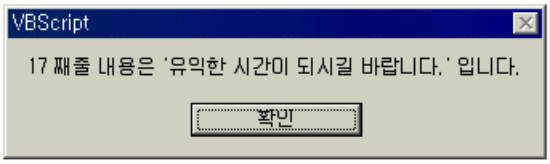
■ 결과

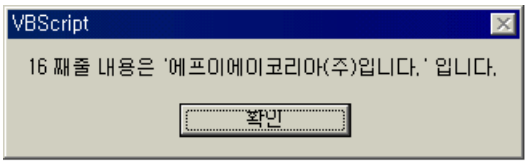
1. Modeller 하단의 Text Window에 아래 그림과 같이 메시지가 출력됩니다.

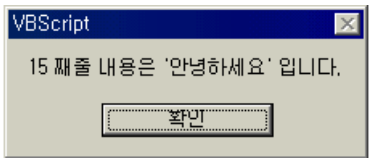


2. 아래와 같은 메시지 창이 순차적으로 나타나게 됩니다. 단, 여기에 표시되는 줄 수는 사용자의 Modeller의 Text Window에 기록된 줄 수에 따라 다르게 나타날 것입니다.









■ 내용 분석

1	
2	Modeller의 Text Window의 속성을 가지는 textwindow라는 이름의 Object를 정의
3-7	각 문장을 textwindow라는 Object, 즉 Modeller Text Window에 지정 문자열 출력
8	현재 작업중인 Modeller의 Text Window에 기록된 문자열 행수를 변수 m에 저장.
9	
10	
11	Text Window에 마지막으로 기록된 행의 내용부터 한 줄씩 위로 5개 행에 기록된 내용을 역으로 보여주게 됩니다. 행번호는 0부터 시작하므로 10번째 행의 내용을 읽으려면 9번 행을 읽으면 됩니다. msgbox 에 대한 내용은 뒤에 다시 다룹니다.

5.4. Msgbox (Method)

5.4.1. 서식

MsgBox(전달내용 [,버튼][, 제목][, 도움말, 문맥])

인수	설명
전달내용	대화 상자에서 메시지로 나타나는 문자식. 프롬프트의 최대 길이는 사용되는 문자의 너비에 따라 다르지만 약 1,024자. 두 줄 이상의 프롬프트이면 캐리지 리턴 문자(Chr(13)), 라인 피드 문자(Chr(10)) 또는 캐리지 리턴-라인 피드 문자의 조합인(Chr(13) & Chr(10))을 사용하여 줄을 구분할 수 있다. **문장내 " 를 사용하고 싶을 때는 Chr(34)를 사용**
버튼	표시할 버튼의 종류와 번호, 사용할 아이콘 유형, 생략 시 기본값은 0.
제목	대화 상자의 제목 표시줄에 표시할 문자열. 생략 시 기본값은 사용중인 프로그램명.
도움말	대화 상자의 문맥 인식 도움말을 제공하기 위해 사용할 도움말 파일을 식별하는 문자식으로 문맥과 같이 사용.
문맥	도움말 작성자가 해당 도움말 항목으로 지정한 도움말 문맥 번호를 식별하는 수식.

■ 메시지창에 표시될 버튼의 종류를 지정하는 문자열 상수

버튼 종류	값	설명
vbOKOnly	0	**확인** 단추만 표시합니다. (기본값)
vbOKCancel	1	**확인** 및 **취소** 단추를 표시합니다.
vbAbortRetryIgnore	2	**중단, 다시 하기** 및 **무시** 단추를 표시합니다.
vbYesNoCancel	3	**예, 아니오** 및 **취소** 단추를 표시합니다.
vbYesNo	4	**예** 및 **아니오** 단추를 표시합니다.
vbRetryCancel	5	**다시 하기** 및 **취소** 단추를 표시합니다.
vbCritical	16	**중대 오류 메시지** 아이콘을 표시합니다.
vbQuestion	32	**경고 쿼리** 아이콘을 표시합니다.
vbExclamation	48	**경고 메시지** 아이콘을 표시합니다.
vbInformation	64	**정보 메시지** 아이콘을 표시합니다.

■ 실행 후 사용자가 선택한 버튼 별 출력값

상수	값	단추
vbOK	1	확인
vbCancel	2	취소
vbAbort	3	중단
vbRetry	4	다시 하기
vbIgnore	5	무시
vbYes	6	예
vbNo	7	아니오

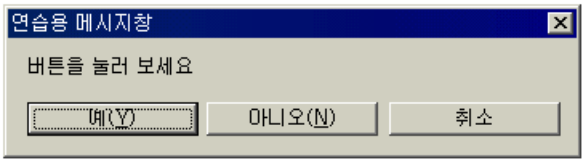
5.4.2. 예제

■ 스크립트 예

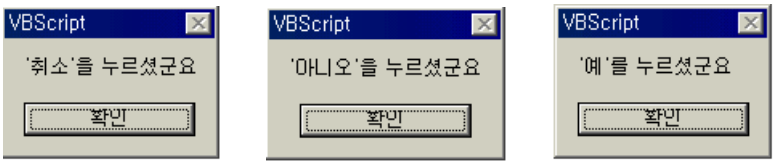
행번	스크립트
1	$ENGINE=VBScript
2	buttonReturn=msgbox ("버튼을 눌러 보세요", vbYesNoCancel, "연습용 메시지창")
3	if buttonReturn=vbYes then msgbox " '예' 를 누르셨군요"
4	if buttonReturn=vbNo then msgbox " '아니오' 를 누르셨군요"
5	if buttonReturn=vbCancel then msgbox " '취소' 를 누르셨군요"
1	$ENGINE=VBScript
2	buttonReturn=msgbox ("버튼을 눌러 보세요", 3, "연습용 메시지창")
3	if buttonReturn=6 then msgbox " '예' 를 누르셨군요"
4	if buttonReturn=7 then msgbox " '아니오' 를 누르셨군요"
5	if buttonReturn=2 then msgbox " '취소' 를 누르셨군요"

■ 결과

위 두 스크립트는 같은 작업을 하며, 아래와 같은 메시지창을 보여줍니다.



버튼을 누름에 따라서 아래와 같이 다른 종류의 메시지창을 다시 띄울 것입니다.



☞ Tip

1. 선택한 버튼을 확인할 필요가 없다면

msgbox "버튼을 눌러 보세요", vbYesNoCancel, "연습용 메시지창 "

과 같이 괄호를 생략할 수 있습니다.

2. 전달 내용을 제외한 나머지 인수는 생략할 수 있습니다.

5.5. 연습문제

1에서 100까지 합산하여 아래의 내용을 Modeller 하단에 출력하게 하는 스크립트 작성

출력결과 :

1+2+3+4+5+...+100 = 5050

행번	스크립트
1	$ENGINE=VBScript
2	
3	Set textwindow = gettextwindow()
4	For i = 1 to 100 step 1
5	Sum=Sum+i
6	If (i<100) Then
7	textsum=textsum&i&"+"
8	Else
9	textsum=textsum&i&"="
10	textwindow.writeline(textsum&Sum)
11	End If
12	Next

☞ Tip

& : 문자와 문자 또는 문자와 숫자를 연결할 때 사용

"" : 문자열로 지정할 때 사용

따옴표는 문자열을 정의할 때 사용되므로 화면에 출력되지 않습니다. 따옴표를 출력하고자 한다면 chr(34)를 사용하여 정의할 수 있습니다.