I. LPI의 이해

1. 개요

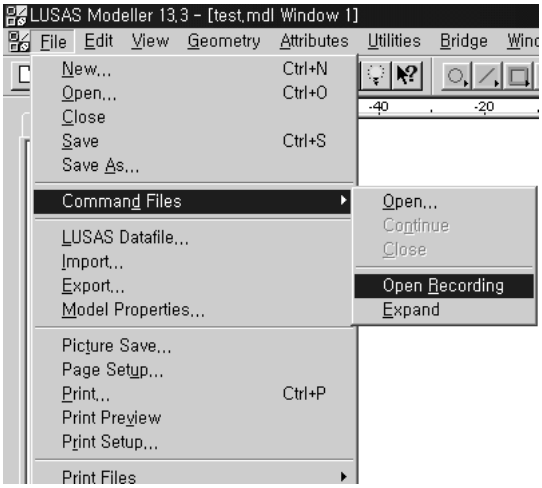
LUSAS Modeller에서 작업하는 모든 내용은 명령어 형태로 표현될 수 있습니다.
이를 Parametric Language라고 하며, 이 명령어 내 수치들을 변수화 시키고 제어문 등을 추가하면 내부 프로그램이 되어서 대화창을 사용해가며 작업을 통제할 수 있게 되는데, 이러한 것을 통칭하여 LPI 라고 합니다.
Parametric Language와 더불어 변수 정의 및 제어문 등으로 사용될 수 있는 프로그래밍 언어로는 V12 이전에는 C++가 사용되었고, V13 이후에 VBScript, JVScript를 추가적으로 사용할 수 있게 되었습니다.
여기서는 C++와 VBScript를 사용하는 방법을 주로 다루기로 하겠습니다.

2. Parametric Language의 이해

2.1. Sub-Session File

Modeller에서 작업하는 내용을 명령어의 형태로 기록하는 파일입니다.
내부프로그래밍의 기초가 되는 것으로 생성된 명령어를 이해할 수 있는 정도면 되는데, 아래의 작업을 통해 테스트 해 볼 수 있겠습니다.

1. 작업 전 기록 모드로 설정하고 파일명을 부여
   File/Command Files/Open...



2. 프로그래밍 하고자 하는 내용과 관련된 작업을 수행
3. 기록 모드를 해제하고 저장
   File/Command Files/Close
   이 때 생성되는 파일의 확장자는 cmd.

☞ Sub-Session File을 지정하지 않을 경우에도 LUSAS는 항상 백업을 위해서 작업 내용을 명령어 형태로 저장을 해 두는데 이를 Session File이라고 하며, 파일명 LUSASM_0.SES로 작업디렉토리에 보관됩니다.

2.2. Command File

Modeller를 구동시키는 명령어들의 나열로 구성된 파일로 확장자가 cmd 인 파일을 말합니다.
앞서의 Sub-Session 파일도 결국 Command File의 일종입니다.

Command File은 다음과 같은 경우에 활용됩니다.
1. 모델 파일 (*.mdl)의 크기가 커서 저장이 곤란할 때 File/SaveAs...에서 Cmd 파일(*.cmd)로 저장하면 모델링 과정을 표현하는 텍스트 명령문만을 가진 파일로 저장이 되므로, 파일 사이즈를 줄여 저장할 수 있게 됩니다. 이 파일을 Modeller에서 다시 불러오면 모델링의 전 작업을 처음부터 다시 수행하여 모델을 복원시키는 역할을 합니다.
2. Sub-Session File을 사용자의 목적에 맞게 수정하거나 Modeller 상에서 대화 형식으로 모델링을 할 수 있도록 프로그래밍 언어를 추가하고자 할 때 유용합니다.

LPI를 활용하는 측면에서는 2.번 항에 해당하는 활용법이 될 것입니다.

2.3. Parametric Language의 활용예

□ 문 : 아래와 같이 10 x 40 평면을 구성하는 Sub-Session File을 생성하고 이를 활용하여 20x50, 30x60 평면을 구성하는 Command File을 작성해 보세요.

■ 답 :
1. Modeller를 열고 Sub-Session File (s01-1040.cmd) 이름을 부여하여 기록 모드 설정.
2. 좌측선을 표현하는 10m 수직선 정의
3. 이 선을 X방향으로 30m Sweeping 하여 면을 생성
4. Sub-Session File을 닫기
5. 아래와 같은 내용을 가진 'S01-1040.cmd'라는 Sub-Session이 생성됨.

```
SET VIEW AUTO_RESIZE OFF
DEFINE LINE BY_COORDINATES LN=* X=0 Y=0 Z=0 X=0 Y=10 Z=0
SET VIEW AUTO_RESIZE
SET VIEW SELECTION AT X=0.0283822 Y=2.81457 SELTYPE=SET FILTER=ALL
DEFINE TRANSFORMATION TRANSLATION ITSET=1 X=40 Y=0 Z=0
DEFINE SURFACE BY_SWEEPING SN=* LN=1 ITSET=1
DELETE TRANSFORMATION ITSET=1
```

6. 위 Command File을 수정하여 20x50 평면을 구성하는 Command File을 작성

```
SET VIEW AUTO_RESIZE OFF
DEFINE LINE BY_COORDINATES LN=* X=0 Y=0 Z=0 X=0 Y=20 Z=0
SET VIEW AUTO_RESIZE
SET VIEW SELECTION AT X=0.0283822 Y=2.81457 SELTYPE=SET FILTER=ALL
DEFINE TRANSFORMATION TRANSLATION ITSET=1 X=50 Y=0 Z=0
DEFINE SURFACE BY_SWEEPING SN=* LN=1 ITSET=1
DELETE TRANSFORMATION ITSET=1
```

7. 아래와 같이 30x60 평면을 구성하는 Command File을 간략화

```
DEFINE LINE BY_COORDINATES LN=1 X=0 Y=0 Z=0 X=0 Y=30 Z=0
DEFINE TRANSFORMATION TRANSLATION ITSET=1 X=60 Y=0 Z=0
DEFINE SURFACE BY_SWEEPING SN=* LN=1 ITSET=1
DELETE TRANSFORMATION ITSET=1
```

위에 사용된 Parametric Language는 직관적으로 그 뜻을 이해할 수 있을 것으로 여겨지나, 정확한 내용을 알고자 하시면 'LUSAS Modeller Reference Manual'을 참조하면 되겠습니다.

7.번 과정에 사용된 명령문(Parametric Language)을 분석해보면,

▷ DEFINE LINE BY_COORDINATES LN=1 X=0 Y=0 Z=0 X=0 Y=30 Z=0
: (0,0,0)과 (0,30,0) 지점을 잇는 Line 1번을 정의

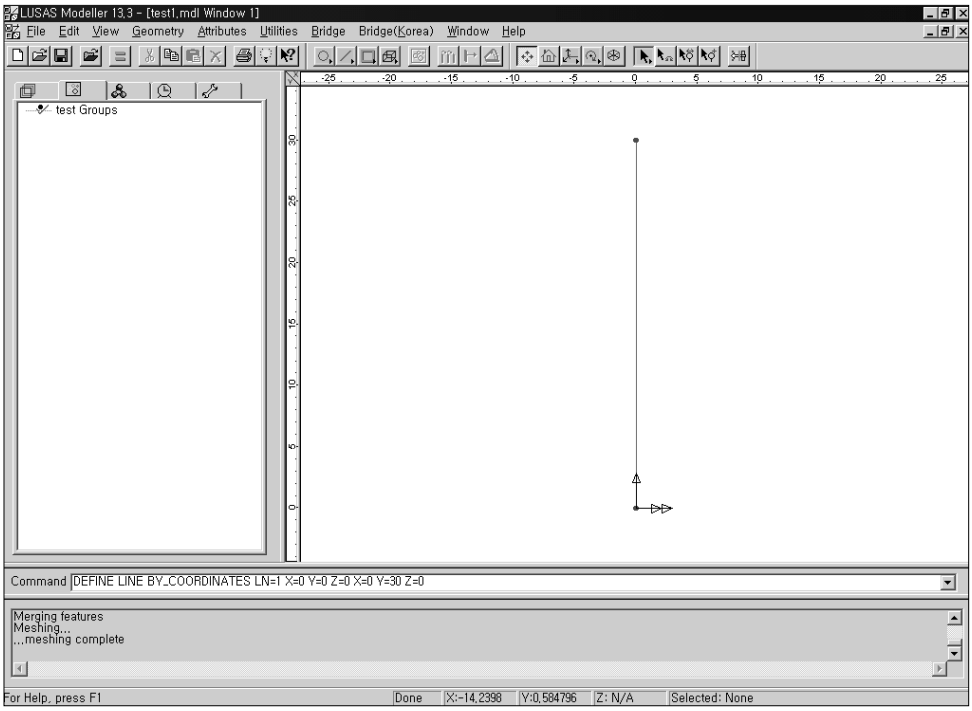
▷ DEFINE TRANSFORMATION TRANSLATION ITSET=1 X=60 Y=0 Z=0
: X방향으로 60만큼 전진하는 Transformation 데이터셋 1번을 정의

▷ DEFINE SURFACE BY_SWEEPING SN=* LN=1 ITSET=1
: Transformation 1번에서 정의된 바에 따라 Line 1번을 Sweeping 시켜서 Surface *번을 정의

▷ DELETE TRANSFORMATION ITSET=1
: Transformation 1번 데이터셋을 삭제

☞ Tip : 데이터셋 번호가 필요한 자리에 '*'는 '다음번호'를 의미한다. 아래의 경우에는 새로이 정의되는 Surface 번호는 현재 정의되어 있는 마지막 Surface 번호의 다음 번호를 사용하겠다는 의미가 됩니다.
DEFINE SURFACE BY_SWEEPING SN=* LN=1 ITSET=1

☞ Tip : 위와 같은 Parametric Command들은 Modeller의 Command Box에 입력하면 그 내용대로 수행됩니다. 예로, "DEFINE LINE BY_COORDINATES LN=1 X=0 Y=0 Z=0 X=0 Y=30 Z=0"를 아래와 같이 Modeller의 Command Box에 입력하시면 지정한 대로 Line을 생성시키는 것을 확인할 수 있습니다.


LUSAS Modeller 13.3 - [test1.mdl Window 1]
File Edit View Geometry Attributes Utilities Bridge Bridge(Korea) Window Help
test Groups
Command DEFINE LINE BY_COORDINATES LN=1 X=0 Y=0 Z=0 X=0 Y=30 Z=0
Merging features
Meshing...
...meshing complete
For Help, press F1
Done
X:-14.2398
Y:0.584796
Z: N/A
Selected: None

II. C언어와의 접목

1. 개요

Parametric Command에 각종 변수를 사용하고, 제어문을 추가하는 등의 프로그래밍 작업을 할 수 있다. 이러한 프로그래밍에는 일정한 규칙이 필요한데, LUSAS가 채택하고 있는 프로그래밍 규칙은 C++과 VBScript, 그리고 JVScript 입니다.

여기서는 C++에 기초한 프로그래밍 규칙에 대해 알아보기로 하겠습니다.

C++을 사용하여 프로그래밍으로 접근할 경우 프로그래밍이 접목된 파일은 확장자 cmd로 저장됩니다. 즉, 프로그래밍 작업이 가해지지 않은 순수 Parametric Command 만이 들어 있는 Command File과 같은 확장자를 사용합니다.

2. 기본 구조

```
Global 변수의 정의
{
    Local 변수의 정의
    Parametric Commands
}
```

* Global 변수 : LUSAS Modeller 및 모델에도 저장되어 항시 사용할 수 있게 되는 변수
  다른 Command file에서도 사용할 수 있습니다.
* Local 변수 : 해당 {} 작업 중에만 유효한 임시 변수, Modeller에 기록되지 않습니다.
* 중괄호{ } : 프로그래밍의 단위. if 문 등 기타 명령어의 영역을 표시하는데도 사용됩니다.

3. 적용 규칙

▶ 모든 변수는 사용 전에 먼저 정의되어야 합니다.

▶ 중괄호 안에서 정의된 변수는 중괄호 내에서 사용될 때에만 유효합니다.

▶ 대소문자는 구별되지 않습니다.

▶ 모든 내부 명령어는 한 줄에 하나만 사용합니다. 단, 한 명령어가 여러 줄에 걸쳐 기록될 경우는 '...'을 끝에 표시함으로써 문장이 다음 줄로 연결되고 있음을 표시합니다.

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

4. 변수의 종류

정수 변수 (Integer) : -2,147,483,646 ~ + 2,147,483,646

실수 변수 (Real) : (+/-)1.0e-38 ~ (+/-) 1.0e38 (PC의 경우)

문자 변수 (Char) : 57개까지의 문자열을 기록하는 변수
문자로 시작되어야 하며, 숫자와 '_', '$' 혼용 가능.

5. 변수의 정의

int integer__A
real real__B=100, real__C=200
char string$="Character variable test"

☞ $는 문자변수명에 사용하면 숫자 변수와의 구별이 용이해집니다.

6. Printf 문의 이해

printf("문자테스트")
: Modeller 하단의 Text Window에 '문자열'을 출력
sprintf(string$,"문자테스트")
: 문자변수 string$에 '문자테스트'이라는 문자열을 대입

7. 변수의 활용

문자열 내에 변수값을 포함시키는 방법으로 활용합니다.
%d: 정수 변수의 값이 들어갈 자리에 표시.
%f: 실수 변수의 값이 들어갈 자리에 표시.
%s: 문자 변수의 값이 들어갈 자리에 표시.

■ 적용 예

```
int integer=5
real real=10.4
char string$
!string$에 정수 변수 integer의 값을 포함한 문자열을 갖게 하기 위해 sprintf문을 사용
sprintf(string$,"정수값 integer=%d", integer)
!Text Window에 각 변수의 내용을 출력하게 함.
printf("정수변수 integer=%d", integer)
printf("실수변수 real=%f", real)
printf("문자열 string$의 내용은 '%s'이다.", string$)
```

■ 적용 결과

위 내용을 test001.cmd 파일로 저장하여, Modeller에서 실행시키면, 모델러 하단의 Text Window에서 아래와 같은 결과를 얻을 수 있습니다.

7.1. 배열 변수의 사용

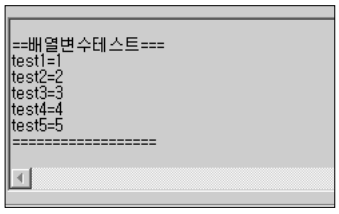
C++형식으로 프로그램을 통제할 뿐, C++ 전체를 다 사용할 수는 없는 제약이 있어 불행히도 배열은 지원하지 않으나 다음 항의 예와 같은 형태로 배열과 유사하게 활용할 수 있게 됩니다.

■ 적용 예

```
int i
char command$
for (i=1; i<=5; i=i+1)
{
   sprintf(command$,“int test%d=%d”, i,i)
   command$
}
!배열변수의 정의와 변수역할 수행이 잘 되었는지 확인하기 위한 출력문 작성
printf("==배열변수테스트===")
printf(“test1=%d”, test1)
printf(“test2=%d”, test2)
printf(“test3=%d”, test3)
printf(“test4=%d”, test4)
printf(“test5=%d”, test5)
printf("==================")
```

■ 적용 결과

위 내용을 test002.cmd 파일로 저장하여, Modeller에서 실행시키면, Modeller 하단의 Text Window에서 아래와 같은 결과를 얻을 수 있습니다.



8. 변수값 입력을 위한 대화창

Command file이 실행되고 있는 동안 모델러에 대화창을 생성시켜서 필요한 변수의 값을 지정할 수 있습니다.

■ 형식

inquire "입력창 제목" 변수명1 "변수입력 요구문1" 변수명2 "변수입력 요구문2" [반복]

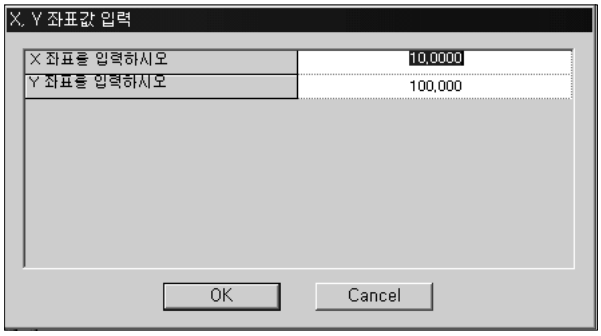
■ 적용 예

```
    real xi, yi
!초기값의 부여
    xi=10
    yi=100
!대화창구성
    inquire "X, Y 좌표값 입력" ...
     xi "X 좌표를 입력하시오" ...
     yi "Y 좌표를 입력하시오"
!입력내용의 확인
    printf("X좌표로 %f를 입력하셨습니다.",xi)
    printf("Y 로 %f를 입력하셨습니다.",yi)
```

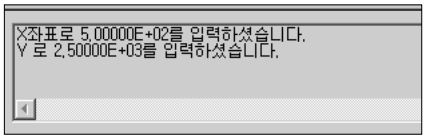
■ 적용 결과

위 내용을 test003inquire.cmd 파일로 저장하여, Modeller에서 실행시키면, 아래 그림과 같은 대화창을 열게 되며, 입력한 값에 따라 모델러 하단의 Text Window에서 그 입력 결과를 확인할 수 있을 것입니다.

<대화창>



각각 500,2500을 입력함.



9. 출력

출력을 위한 별개의 명령어를 사용할 필요 없이, 그림 파일이나 텍스트 파일 등을 출력하는 LUSAS의 기능을 그대로 사용합니다.

참고로, 기록될 텍스트 파일을 지정해 둔 상태에서 printf문을 사용하면 Modeller의 Text Window 대신 지정 파일에 내용을 인자하게 됩니다.

■ 적용 예

```
print open textoutput
    printf("종합결과를 출력합니다")
    print result summary nodes
print close
```

■ 적용 결과

위 내용을 test004print.cmd 파일로 저장하고, 임의의 결과 파일 (TrSample.mys)을 Modeller에 부른 상태에서 실행시키면, 작업디렉토리 내에 textoutput.prn 파일이 생성된 것을 확인할 수 있게 됩니다. textoutput.prn 파일의 내용은 아래와 같이 결과 Summary를 포함하고 있을 것입니다.

```
종합결과를 출력합니다
LINEAR/DYNAMIC ANALYSIS
Current Results File=D:\내 문서\Business\07프로젝트\17스크립트연재\truss.mys ID    0
Current Selected Load ID     =            1
"Loadcase 1"
Averaged Forces And Moments In Element Local Axes
     Node       Fx
Maximum   0.8333E+00
   Node            10
Minimum -0.8536E+01
   Node             7
Max Strain Energy Density Value  1.79425837E-07 at Node          7
Min Strain Energy Density Value  7.30994152E-08 at Node         13
Total Strain Energy For Active Set =  2.29090288E-05
Max Plastic Work Density Value  0.00000000E+00 at Node          1
Min Plastic Work Density Value  0.00000000E+00 at Node          1
Total Plastic Work For Active Set =  0.00000000E+00
```

10. 제어문

10.1. GOTO 문

명령문의 수행 순서를 바꿉니다.

■ 적용 예

```
printf("시작합니다.")
goto a
printf("이 문장이 보이면 goto문이 실행되지 않은 것입니다.")
a:
printf("이 문장이 보이면 goto문이 실행된 것입니다.")
```

10.2. ON CANCEL GOTO 문

INQUIRE문에서 사용자가 변수값 입력 대신 Cancel 버튼을 선택했을 때 이동 위치 지정

10.3. IF 문

■ 형식

```
if ( 판단문 1 )
{
 명령문들
}
elseif ( 판단문2 )
{
 명령문들
}
else
{
 명령문들
}
```

■ 판단문에 사용되는 비교 연산자

== , !=, >, <, >=, <=

■ 판단문의 연결에 사용되는 논리 연산자

&&, ||

■ 산술 연산자

+, -, *, /, ^, ;

10.4. FOR loop문

■ 형식

```
for (시작조건; loop 종료 조건; 증분표시)
{
 명령문들
}
```

☞ 주의 : 명령문의 범위를 지정하는 {, } 는 한 줄에 하나만 위치할 수 있으며, 다른 명령문과 한 줄에 중복되어 사용될 수 없습니다.

10.5. 일반함수

+,-,*,/,**	: 더하기, 빼기, 곱하기, 나누기, 거듭제곱
sin(),cos(),tan()	: Radian입력에 의한 삼각함수, 예: sin(3.141598)=0
sind(),cosd(),tand()	: Degree 입력에 의한 삼각함수, 예: sind(30)=0.5
asin(),acos(),atan()	: Radian 값을 얻어내는 역삼각함수, 예: asin(0)=3.141598
asind(),acosd(),atand()	: Degree 값을 얻어내는 역삼각함수, 예: asind(0.5)=30
atan2(x1,x2)	: tan(x1/x2)의 Radian 값을 갖는 역삼각함수
atan2d(x1,x2)	: tan(x1/x2)의 Degree 값을 갖는 역삼각함수
sinh(deg),cosh(deg),tanh(deg)	: Degree 입력에 의한 Hyperbolic function
rtod(rad), dtor(deg)	: Radian을 Degree로, Degree를 Radian으로, 예:rtod(3.14)=180
abs()	: 절대값으로 변환, 예: abs(-3.1415)=3.1415
sqrt()	: 제곱근, 예: sqrt(4)=2
sqr()	: 제곱, 예: sqr(4)=16
int()	: 정수값으로 변환, 예: int(3.1415)=3
nint()	: 반올림 정수값으로 변환, 예: nint(3.5)=4
max(a,b)	: a,b중 큰 값, 예: max(3,4)=4
min(a,b)	: a,b중 작은 값, 예: min(3,4)=3
pi	: 원주율

11. m$largest(Type)의 이해

모델링 상태 중 지정한 Type의 최종 데이터셋 번호를 의미합니다. 프로그래밍 도중 데이터셋을 정의하거나 참조할 때 유용하게 활용됩니다.

구 분	적용할 수 있는 Type 명
Geometry	point, line, combined_line, surface, volume
Attribute	mesh, geometry, material, support, loading, equivalence, search_area, retained_freedom, local_coordinate, slide_property, slide_table, composite, constraint_equation, activate, deactivate, damping_properties, damping_table
Utilities	variation, transformation, background_grid, control, component, load_curve, influence_line
Load Case	loadcase_all, loadcase_loading, loadcase_support, loadcase_activate, loadcase-deactivate, loadcase_load_curve, loadcase_slide_table, loadcase_control
Thermal	thermal_gap_property, thermal_contact_property, thermal_environment_property, thermal_radiation_property, thermal_surface_definition, thermal_gap_definition, thermal_radiation_definition
Post Processing	dataset, frequency_psd, spectrum, user, sn_curve, combination, envelope, graph_number

■ **사용 예**

int k=m$largest(point)

k는 현재 작업 중인 모델 중 가장 마지막 Point의 번호를 값으로 가지는 변수가 됨.

12. Command File 프로그래밍 주의사항

▶ 좌표값이 필요한 곳에는 변수명를 그대로 사용할 수 있습니다.

DEFINE LINE BY_COORDINATES LN=1 X=0 Y=0 Z=0 X=0 Y=ycoord Z=0

▶ 데이터셋 번호를 기입하는 항목에는 변수를 사용할 수 없습니다.

변수를 사용하고자 할 경우에는 sprintf문을 이용하여 LUSAS명령문을 완성하여 사용하여야 합니다.

```
sprintf(command$,"int test%d=%d", i,i)
command$
```

▶ 새로이 정의되는 각종 데이터셋의 번호를 입력하는 항목에 표시된 '*'는 자동으로 다음 번호를 부여한다는 의미가 됩니다.

▶ LUSAS 명령어중 Assign을 사용할 때 적용 대상을 지정할 때에 '[시작번호]t[끝번호]i[간격]' 과 같은 형태로 입력할 수 있습니다.

▶ 데이터셋의 제목을 부여할 때 사용되는 문자열에는 공란이 들어가지 않도록 해야 합니다.

III. VBScript의 접목

1. 개요

Parametric Command에 각종 변수를 사용하고, 제어문을 추가하는 등의 프로그래밍 작업을 할 수 있습니다. 이러한 프로그래밍에는 일정한 규칙이 필요한데, LUSAS가 채택하고 있는 프로그래밍 규칙은 C++과 VBScript, 그리고 JVScript 입니다.
여기서는 VBScript에 기초한 프로그래밍 규칙에 대해 알아보기로 하겠습니다.
VBScript을 사용하여 프로그래밍으로 접근할 경우 프로그래밍이 접목된 파일은 확장자 vbs로 저장합니다.

2. 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.

3. 적용 규칙

▶ 변수는 사전 정의될 필요는 없으며, 변수가 사용되는 경우에 따라 문자변수가 되거나 실수변수가 되기도 합니다. 편리하나 연산 작업을 수행할 경우 문자연산으로 처리될 수 있는 등 주의가 필요하게 됩니다.
▶ Command File과 달리 배열을 사용할 수 있으며, 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.

4. 변수의 사용

4.1. 변수의 종류

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

4.2. 변수의 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, Cdouble를 연산 작업이 되기 전에 설정하여 오류를 방지할 수 있을 것입니다.

4.3. 배열 변수의 정의

배열 변수는 한가지 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.

■ 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 재 정의하게 되면, 이전 배열에 저장된 값들은 삭제가 됩니다. 이전 값을 사용해야 할 필요가 있을 경우에는 아래와 같이 preserve 키워드를 포함시키면 됩니다.

```
redim preserve variable(4)
```

5. 제어문

5.1. FOR 문

■ 형식

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

5.2. DO ... LOOP 문

■ 형식 1

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

■ 형식 2

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

5.3. 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, 생략가

■ 판단문에 사용되는 비교 연산자

=, <>, >, <, >=, <=

■ 판단문의 연결에 사용되는 논리 연산자

and, or, not, xor, eqv

(논리곱, 논리합, 부정, 제외, 같음)

■ 산술 연산자

+, -, *, /, ^, mod, ₩, &

5.4. SELECT CASE 문

■ 형식

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

6. Object 와 Method 의 이해

■ 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 합니다.

☞ 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.

7. 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

7.1. getMainMenu()

■ 예: set MainMenu=getMainMenu()

■ 내용 : Modeller의 메뉴 구성에 대한 모든 내용을 포함하는 MainMenu라는 이름의 Object를 정의합니다.

7.2. getDatabase()

■ 예: set database=getDatabase()

■ 내용 : Modeller에서 작업되고 있는 전체 내용을 포함하는 database라는 이름의 Object를 정의합니다.

7.3. newDatabase()

■ 예: set database=newDatabase()

■ 내용 : Modeller를 초기화시키고, Modeller에서 작업되고 있는 전체 내용을 포함하는 database라는 이름의 Object를 정의합니다.

7.4. getSelection()

■ 예: set selected=getSelection()

■ 내용 : 현재 명령이 실행 되기 전에 Modeller에서 마우스 혹은 기타의 방법으로 선택된 Geometry나 Mesh등이 있다면, 이들의 속성을 모두 가지는 selected 라는 이름의 Object를 정의합니다.

7.5. processCommand(command$)

■ 예: processCommand("DEFINE LINE BY_COORDINATES LN=1 X=0 Y=0 Z=0 X=0 Y=30 Z=0")

■ 내용 : LUSAS의 Parametric Command를 문자열로 넣어주면, 해당 내용을 Modeller에서 실행시키는 역할을 합니다. Modeller의 Command Box에 명령문을 입력하는 것과 같은 효과입니다.

7.6. 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.

7.7. createSimpleDialog (....)

대화창을 구성하는 Object로 별도로 다루기로 하고 여기서는 자세한 설명을 생략합니다.

7.8. 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.

8. ProcessCommand 문의 이해

8.1. 형식

processCommand(Parametric Command 문자열)

8.2. 사용예

■ 예: processCommand("DEFINE LINE BY_COORDINATES LN=1 X=0 Y=0 Z=0 X=0 Y=30 Z=0")
■ 내용 : LUSAS의 Parametric Command를 문자열로 넣어주면, 해당 내용을 Modeller에서 실행시키는 역할을 합니다. Modeller의 Command Box에 명령문을 입력하는 것과 같은 효과입니다.

8.3. Command File과의 비교 A

앞에서 다룬 30x60 Surface를 구성하는 Command File을 Script File로 단순하게 변화시킨다면 아래와 같이 할 수 있을 것입니다.

▶ Command File (test005surdef.cmd)

```
DEFINE LINE BY_COORDINATES LN=1 X=0 Y=0 Z=0 X=0 Y=30 Z=0
DEFINE TRANSFORMATION TRANSLATION ITSET=1 X=60 Y=0 Z=0
DEFINE SURFACE BY_SWEEPING SN=* LN=1 ITSET=1
DELETE TRANSFORMATION ITSET=1
```

▶ Script File (test005surdef.vbs)

```
$ENGINE=VBSCRIPT
ProcessCommand("DEFINE LINE BY_COORDINATES LN=1 X=0 Y=0 Z=0 X=0 Y=30 Z=0")
ProcessCommand("DEFINE TRANSFORMATION TRANSLATION ITSET=1 X=60 Y=0 Z=0")
ProcesCommand("DEFINE SURFACE BY_SWEEPING SN=* LN=1 ITSET=1")
ProcessCommand("DELETE TRANSFORMATION ITSET=1")
```

☞ ProcessCommand() 의 () 안에는 완성된 문자열 또는 문자변수를 입력해야 합니다.

8.4. Command File과의 비교 B - 변수를 가지는 경우

변수 xcoord, ycoord 값의 크기에 따른 Surface를 생성하는 스크립트의 작성

▶ Command File (test006surdef.cmd)

```
real xcoord = 10
real ycoord = 20

DEFINE LINE BY_COORDINATES LN=1 X=0 Y=0 Z=0 X=0 Y=ycoord Z=0
DEFINE TRANSFORMATION TRANSLATION ITSET=1 X=xcoord Y=0 Z=0
DEFINE SURFACE BY_SWEEPING SN=* LN=1 ITSET=1
DELETE TRANSFORMATION ITSET=1
```

▶ Script File (test006surdef.vbs)

```
$ENGINE=VBSCRIPT

xcoord = 10
ycoord = 20
command1="DEFINE LINE BY_COORDINATES LN=1 X=0 Y=0 Z=0 X=0 Y="&ycoord&" Z=0"
command2="DEFINE TRANSFORMATION TRANSLATION ITSET=1 X="&xcoord&" Y=0 Z=0"
ProcessCommand(command1)
ProcessCommand(command2)

ProcessCommand("DEFINE SURFACE BY_SWEEPING SN=* LN=1 ITSET=1")
ProcessCommand("DELETE TRANSFORMATION ITSET=1")
```

☞ Tip : 문자열을 연결할 때에는 & 사용합니다.

9. 메뉴의 구성과 관련된 Method

9.1. Menu 구성과 관련된 Method의 종류

아래 Method 들은 전체 메뉴를 속성으로 갖는 Object 하에서 사용될 수 있습니다.
즉, Set MainMenu=getMainMenu() 와 같이 상위 Object가 정의되어 있어야 합니다.

Method 서식	산출물 / 결과
getSubMenu(메뉴명$)	지정한 메뉴 Object
insertMenu(메뉴명1$,새메뉴명2$)	메뉴1 앞에 추가한 새 메뉴 Object
appendMenu(메뉴명$)	주메뉴에 추가한 새 메뉴 Object
removeMenu(메뉴명$)	지정한 메뉴 삭제
insertItem(항목명$, 새항목명$, 명령문$)	상위 Object (메뉴)내 기존 항목명 앞에 새항목을 추가하고 이 항목이 선택되면 명령문이 실행되게 한다.
appendItem(항목명$,명령문$)	상위 Object(메뉴)내 마지막에 새항목 추가
removeItem(항목명$)	항목 삭제
renameItem(이전항목명$, 새항목명$)	
insertSeparator(항목명$)	항목명 앞에 구분선 표시
isMenu(메뉴명$)	지정 메뉴명이 있으면 True, 아니면 False
isSeparator(항목명$)	지정 항목명 앞에 구분선이 있으면 True
countItems()	등록된 메뉴의 개수
existsMenu(항목명$)	지정 메뉴명이 있으면 True, 아니면 False

☞ Tip
1. AppendItem은 실행항목을 추가하는 개념이며, AppendMenu는 부메뉴를 추가하는 개념입니다.
2. 명령문$ 에는 LUSAS 명령어를 넣거나, 또는 매크로/자동화 Command file이나 Script file을 호출하는 내용을 넣어 활용할 수 있습니다.

9.2. 예제 실습

■ 스크립트 예

test007Menu.vbs

```
$ENGINE=VBSCRIPT
set MainMenu = getMainMenu()
if ( MainMenu.existsMenu("&Test") ) then
  MainMenu.removeMenu("&Test")
end if

set testMenu=MainMenu.insertMenu("&Window","&Test")
set backMenu=testMenu.AppendMenu("배경변화")

backMenu.AppendItem "흑색배경", "set view background 0 0 0"
backMenu.AppendItem "백색배경", "set view background 100 100 100"
```

☞ Tip : &을 메뉴명 또는 항목명 중간에 넣으면 & 뒤의 문자는 화면상에 밑줄이 추가되어 나타나며, 이것이 단축키가 됩니다.

■ 결과

위 내용을 test007Menu.vbs로 저장하고, 이것을 Modeller에서 실행을 시키면 아래 그림과 같이 Modeller의 주메뉴 중 Window메뉴 앞에 Test라는 메뉴가 추가되게 됩니다.

흑색배경 메뉴를 선택하면 Modeller의 배경화면이 흑색으로 바뀌게 될 것입니다.

■ 내용 분석

$ENGINE=VBSCRIPT
set MainMenu = getMainMenu()
: 주메뉴의 모든 속성을 가지는 MainMenu 라는 이름의 Object 정의

if (MainMenu.existsMenu("&Test")) then
MainMenu.removeMenu("&Test")
end if
: 주메뉴에 Test 라는 이름의 메뉴가 있으면 이것을 삭제

set testMenu=MainMenu.insertMenu("&Window","&Test")
: 주메뉴 중 Window 앞에 Test 라는 이름의 메뉴를 추가하고, 추가된 메뉴의 속성들은 testMenu 라는 Object가 갖도록 지정

set backMenu=testMenu.AppendMenu("배경변화")
: Test메뉴에 부메뉴"배경변화"를 추가하고, 추가된 부메뉴의 속성을 가지는 Object는 backMenu라고 정의

backMenu.AppendItem "흑색배경", "set view background 0 0 0"
: 부메뉴에 "흑색배경"이라는 이름을 가지는 Item을 추가하고, 이것이 선택되면 LUSAS Parametric Command "set view background 0 0 0"가 수행되도록 지정
backMenu.AppendItem "백색배경", "set view background 100 100 100"
: 부메뉴에 "백색배경"이라는 이름을 가지는 Item을 추가하고, 이것이 선택되면 LUSAS

Parametric Command "set view background 100 100 100"가 수행되도록 지정

☞ Tip
배경화면 색을 바꾸는 작업을 Modeller에서 하려고 하면
1) Modeller 작업창에 마우스를 올려두고
2) 오른쪽 버튼을 클릭하면 나타나는 Pop-up 메뉴에서 작업창의 Properties를 선택할 수 있고
3) 여기에서 바탕화면 색깔을 지정할 수 있게 됩니다.

☞ Tip
위 작업들을 Sub-session File을 열어두고 작업을 수행하면 바탕화면색을 흑색, 백색으로 지정할 때에 해당하는
LUSAS Parametric Command 는 각각
"set view background 0 0 0"
"set view background 100 100 100"
이 됨을 알 수 있습니다.

10. 대화창을 구성하는 Method

10.1. 서식

createSimpleDialog ("입력창 제목", 세로제목 배열명, 가로제목 배열명, 입력치 배열명)

▶ createSimpleDialog은 대화창에서 사용자가 '확인'을 선택하면 "true", 아니면 "false"를 돌려줍니다. 이 값을 변수로 받아서 사용자의 작업 과정을 체크하여 프로그래밍에 활용합니다. (예: 데이터 입력 '취소'를 선택하면 프로그램의 진행을 중단하도록 프로그래밍)

▶ 배열명(배열 변수)는 사전에 정의되어야 하고, redim 선언문을 사용하여 변수를 정의하는 것이 편리한 경우가 많습니다. 단, 변수가 가지는 값은 항상 문자열이어야 합니다.

▶ 사용예

okpressed=createSimpleDialog(title, labels, headers, data)

10.2. 예제 실습

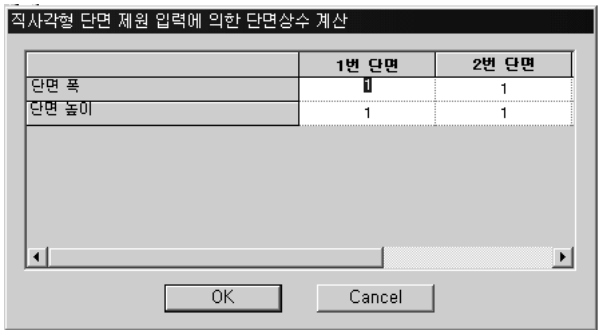
■ 스크립트 예

test008dial.vbs

```
$ENGINE=VBSCRIPT
title="직사각형 단면 제원 입력에 의한 단면상수 계산"
xinputs=2
yinputs=1
ReDim headers(xinputs)
ReDim labels(yinputs)
ReDim data(xinputs,yinputs)
labels(0)="단면 폭"
labels(1)="단면 높이"
for i=0 to 2 step 1
   headers(i)=i+1&"번 단면"
next
for i=0 to 2
  for j=0 to 1
    data(i,j)="1"
  next
next

okpressed=createSimpleDialog(title, labels, headers, data)
```

■ 결과

위 내용을 test008dial.vbs로 저장하고, 이것을 Modeller에서 실행을 시키면 아래 그림과 같이 입력을 요구하는 대화창이 생성됩니다.



각 항목에 입력되는 값들이 소속될 변수명은 아래의 표와 같습니다.

직사각형 단면 제원 입력에 의한 단면상수 계산			
	1번 단면	2번 단면	3번 단면
단면 폭	*data(0, 0)*	*data(1, 0)*	*data(2, 0)*
단면 높이	*data(0, 1)*	*data(1, 1)*	*data(2, 1)*

■ 내용 분석

```
$ENGINE=VBSCRIPT
title="직사각형 단면 제원 입력에 의한 단면상수 계산"
```
: title 이라는 문자변수에 문자열 대입

```
xinputs=2
yinputs=1
```

```
ReDim headers(xinputs)
```
: xinput=2 이므로 header(0),header(1),header(2) 3개의 배열변수 정의. 대화창 상단 제목들을 가지는 변수로 활용하기 위해 정의

```
ReDim labels(yinputs)
```
: yinput=1 이므로 labels(0), labels(1) 2개의 배열변수 정의. 대화창 좌측 제목으로 사용될 제목들을 가지는 변수로 활용하기 위해 정의.

```
ReDim data(xinputs,yinputs)
```
: data(0,0) ~ data(2,1)까지 총 6개의 배열변수 정의. 대화창에서 사용자가 입력하는 값들을 받아들여 저장하기 위해 정의.

```
labels(0)="단면 폭"
labels(1)="단면 높이"
```
: 배열변수 labels에 문자열 부여. 대화창 좌측의 제목으로 사용

```
for i=0 to 2 step 1
  headers(i)=i+1&"번 단면"
next
```

: 배열변수 headers에 문자열 부여, 즉 headers(0)="1번 단면", headers(1)="2번 단면", headers(2)="3번 단면" 으로 지정. 대화창 상단의 제목으로 사용

```
for i=0 to 2
 for j=0 to 1
  data(i,j)="1"
 next
next
```

: 배열변수 data(0,0) ~ data(2,1)의 총 6개에 각각 "1"이라는 초기값을 부여. 대화창에서 사용자입력란에 초기값으로 표시됨.

okpressed=createSimpleDialog(title, labels, headers, data)

: headers(0)~headers(2)까지 3개의 상단제목과, labels(0)~labels(1)까지 2개의 좌측 제목을 가진 data(0,0)~data(2,1)까지의 6개의 입력값을 받을 수 있는 대화창 생성. 사용자가 입력을 정상적으로 수행하고 "OK"를 선택하면 "True"를, "Cancel"을 선택하면 "False"를 수행 후 결과보고를 하게되므로, okpressed 라는 변수는 "True" 혹은 "False"라는 논리값이 됨.

11. TextWindow 관련 Method

11.1. TextWindow와 관련된 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

11.2. 예제 실습

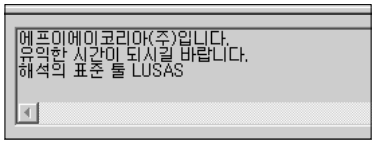
■ 스크립트 예

test009textw.vbs

```
$ENGINE=VBSCRIPT
Set textwindow = gettextwindow()
textwindow.writeline("안녕하세요")
textwindow.writeline("에프이에이코리아(주)입니다.")
textwindow.writeline("유익한 시간이 되시길 바랍니다.")
textwindow.writeline("해석의 표준 툴 LUSAS")

m=textwindow.countLines()
For i=m-1 to m-4 step -1
        text=i &" 째줄 내용은 '" & textwindow.readLine(i)&"' 입니다."
        MsgBox text
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

```
$ENGINE=VBSCRIPT
Set textwindow = gettextwindow()
```
: Modeller의 Text Window의 속성을 가지는 textwindow라는 이름의 Object를 정의

```
textwindow.writeline("안녕하세요")
textwindow.writeline("에프이에이코리아(주)입니다.")
textwindow.writeline("VBS 교육 과정에 오신 것을 환영합니다.")
textwindow.writeline("유익한 시간이 되시길 바랍니다.")
textwindow.writeline("해석의 표준 툴 LUSAS")
```
: 각 문장을 textwindow라는 Object, 즉 Modeller Text Window에 지정 문자열 출력

```
m=textwindow.countLines()
```
: 현재 작업중인 Modeller의 Text Window에 기록된 문자열 행수를 변수 m에 저장.

```
For i=m-1 to m-5 step -1
    text=i &" 째줄 내용은 '" & textwindow.readLine(i)&"' 입니다."
    MsgBox text
Next
```
: 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.

12. 텍스트 파일 입출력 관련 Method

파일 입출력과 관련해서는 사용하는 Script 언어의 기능을 전적으로 사용하며, LUSAS 자체에서 제공하는 Method는 없는데, 일반적으로 다음과 같이 내부 함수를 정의하여 사용하면 큰 불편 없이 사용할 수 있습니다.

12.1. 새로운 파일을 생성하는 함수(Function)

■ 함수 예

```
FUNCTION newTextFile (fileSpec, overwriteExisting)
  set fileSys = CreateObject("Scripting.FileSystemObject")
  set newTextFile = fileSys.CreateTextFile(fileSpec, overwriteExisting)
END FUNCTION
```

■ 내용 분석

```
FUNCTION newTextFile (fileSpec, overwriteExisting)
        : 부프로그램 선언문

  set fileSys = CreateObject("Scripting.FileSystemObject")
        : 디렉토리구조를 속성으로 하는 fileSys라는 Object 정의

  set newTextFile = fileSys.CreateTextFile(fileSpec, overwriteExisting)
        : fileSpec - 파일명을 포함하는 디렉토리 구조 (예: "c:\lusas13\test.txt")
          overwriteExisting - 덮어 쓰기에 대한 논리값 (True 혹은 False)
        즉, fileSpec 에 해당하는 파일을 새로 생성하고, 그 속성을 가지는 Object
        이름은 newTextFile로 한다는 것입니다. 생성하고자 하는 파일이 이미 존재
        하는 경우는 overwriteExisting의 값이 True이면 덮어 씌우고, False이면 파
        일을 생성하지 않게 됩니다.

END FUNCTION
        : 부프로그램 종료 문
```

12.2. 기존 파일을 읽기 위해서 열기 위한 함수 (Function)

■ 함수 예

```
FUNCTION getTextFileForReading (fileSpec)
  const ForReading = 1
  set fileSys = CreateObject("Scripting.FileSystemObject")
  set getTextFileForReading = fileSys.OpenTextFile(fileSpec, ForReading)
END FUNCTION
```

■ 내용 분석

생략

12.3. 파일 입출력 관련 함수의 활용

■ 활용 예

test009textf.vbs

```
$ENGINE=VBScript
fileName = "d:\lusas13\projects\test.txt"
set outFile = newTextFile (fileName, TRUE)
outFile.write "Date:" & Date & " Time: " & Time & vbNewLine
outFile.close()
set inFile = getTextFileForReading (fileName)
text = inFile.readall
inFile.close()
msgbox text

FUNCTION newTextFile (fileSpec, overwriteExisting)
set fileSys = CreateObject("Scripting.FileSystemObject")
set newTextFile = fileSys.CreateTextFile(fileSpec, overwriteExisting)
END FUNCTION

FUNCTION getTextFileForReading (fileSpec)
const ForReading = 1
set fileSys = CreateObject("Scripting.FileSystemObject")
set getTextFileForReading = fileSys.OpenTextFile(fileSpec, ForReading)
END FUNCTION
```

■ 결과

1. Modeller 에 현재 날짜와 시간을 알리는 메시지창이 나타납니다.
2. 위 fileName에 지정한 디렉토리에 test.txt 파일이 생성되며,

Date:2001-12-5 Time: 오후 6:54:00

와 같은 내용의 문자열을 가지고 있게 됩니다.

■ 내용 분석

$ENGINE=VBScript
fileName = "d:\lusas13\projects\test.txt"
set outFile = newTextFile (fileName, TRUE)
: "d:\lusas13\projects\test.txt" 라는 문자열과 True라는 논리값을 매개변수값으로 하여 newTextFile이라는 제목의 함수를 호출합니다.

newTextFile(fileSpec,overwriteExisting) 함수는 fileSpec = "d:\lusas13\projects\ test.txt" 이 되고, overwriteExisting = TRUE 로 인식하여 작업을 수행합니다.

newTextFile 함수는 산출물은 새로이 생성된 파일의 속성을 갖는 Object이므로, outFile은 이렇게 하여 생성된 새 파일의 속성을 갖는 Object가 됩니다.

outFile.write "Date:" & Date & " Time: " & Time & vbNewLine
: 앞과정에서 생성된 새로운 파일에 날짜와 시간을 기록합니다.
Date는 시스템의 현재 날짜를 나타내는 내부함수. (예: 2001-12-5)
Time은 시스템의 현재 시간을 나타내는 내부함수. (예: 오후 6:54:00)
vbNewLine은 줄바꿈을 의미하는 VBScript의 문자열 상수(String Constant).

outFile.close()
: 작업중인 파일 (outFile Object)을 저장하고 닫기
set inFile = getTextFileForReading (fileName)

: "d:\lusas13\projects\test.txt"를 매개변수값으로 하여 getTextFileForReading 함수를 호출
getTextFileForReading (fileSpec) 함수는 fileSpec = "d:\lusas13\projects\ test.txt" 으로 인식하여 작업을 수행
getTextFileForReading (fileSpec) 함수는 산출물은 읽기 위해 열어 놓은 파일의 속성을 갖는 Object이므로, inFile은 이 Object가 됩니다.

```
text = inFile.readall
```
: inFile에 해당하는 파일의 모든 내용을 읽어서 text라는 변수에 저장

```
inFile.close()
```
: 작업중인 파일 (inFile Object)를 닫기

```
msgbox text
```
: text에 저장중인 문자열을 메시지창으로 표시

```
FUNCTION newTextFile (fileSpec, overwriteExisting)
set fileSys = CreateObject("Scripting.FileSystemObject")
set newTextFile = fileSys.CreateTextFile(fileSpec, overwriteExisting)
END FUNCTION
```
: 생략 (앞에서 설명)

```
FUNCTION getTextFileForReading (fileSpec)
const ForReading = 1
set fileSys = CreateObject("Scripting.FileSystemObject")
set getTextFileForReading = fileSys.OpenTextFile(fileSpec, ForReading)
END FUNCTION
```
: 생략 (앞에서 설명)

☞ Tip
: VBScript에서 사용하는 문자열 상수

상수	값	설명
vbCr	Chr(13)	캐리지 리턴
vbCrLf	Chr(13) & Chr(10)	캐리지 리턴-라인 피드 조합
vbFormFeed	Chr(12)	폼 피드. Microsoft Windows에서는 사용되지 않습니다.
vbLf	Chr(10)	라인 피드
vbNewLine	Chr(13) & Chr(10) 또는 Chr(10)	각 플랫폼에 따라 적절한 플랫폼 특정의 줄 바꿈 문자
vbNullChar	Chr(0)	값이 0인 문자
vbNullString	값이 0인 문자열	길이가 0인 문자열("")과 다르며, 외부 프로시저를 호출하는 데 사용됩니다.
vbTab	Chr(9)	수평 탭
vbVerticalTab	Chr(11)	수직 탭. Microsoft Windows에서는 사용되지 않습니다.

13. 메시지창을 띄우는 Method

13.1. 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

13.2. 사용 예

■ 스크립트 예

test010msg1.vbs

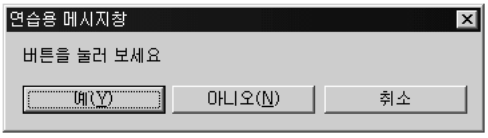
```
$ENGINE=VBScript
buttonReturn=msgbox ("버튼을 눌러 보세요", vbYesNoCancel, "연습용 메시지창")
if buttonReturn=vbYes then msgbox "Yes를 누르셨군요"
if buttonReturn=vbNo then msgbox "No을 누르셨군요"
if buttonReturn=vbCancel then msgbox "Cancel을 누르셨군요"
```

test010msg2.vbs

```
$ENGINE=VBScript
buttonReturn=msgbox ("버튼을 눌러 보세요", 3, "연습용 메시지창")
if buttonReturn=6 then msgbox "Yes를 누르셨군요"
if buttonReturn=7 then msgbox "No을 누르셨군요"
if buttonReturn=2 then msgbox "Cancel을 누르셨군요"
```

■ 내용

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"
와 같이 괄호를 생략할 수 있습니다.
2. 전달 내용을 제외한 나머지 인수는 생략할 수 있습니다.

14. Excel과 데이터 주고 받기

엑셀과의 데이터 교환으로 계산서 출력까지를 과정을 쉽게 처리할 수 있을 것입니다.
데이터 교환 관련 내용은 프로그래밍의 주 부분은 아니며, 아래의 표준안을 적절히 수정하여 그대로 적용하면 되므로 세부적인 설명은 생략하겠습니다.
보다 자세한 설명이나 필요한 기능들은 VBScript 매뉴얼을 참조시기 바랍니다.

14.1. 새 파일을 작성하고자 할 경우

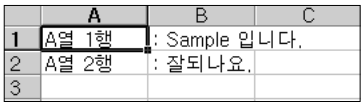
■ 스크립트 예

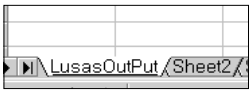
test011toExcel.vbs

```
$ENGINE=VBSCRIPT
' Open Excel application
    set ExcelApp = CreateAutomationObject("Excel.Application")

' 생성시킬 파일 이름과 시트 이름을 정의
    FilePath="c:\lusas13\projects\"
    FileName="LusasExcel"
    SheetName="LusasOutPut"

' Excel 파일 생성
    ExcelApp.DefaultFilePath = FilePath
    Set WorkBook=ExcelApp.WorkBooks.Add
    WorkBook.Title=FileName

' 생성된 파일 내 작업 시트 정의
    set WorkSheet= WorkBook.WorkSheets(1)
    WorkSheet.Name=SheetName

' 내용 편집
    WorkSheet.Cells(1,1).Value = "A열 1행"
    WorkSheet.Cells(1,2).Value = ": Sample 입니다."
    WorkSheet.Cells(2,1).Value = "A열 2행"
    WorkSheet.Cells(2,2).Value = ": 잘되나요."

' 편집한 파일을 저장
    Fname=FileName + ".xls"
    WorkBook.SaveAs(FName)

' Excel 사용 종료
    ExcelApp.Quit
    Set ExcelApp = Nothing
```

■ 결과

위에서 지정한 디렉토리에 LusasExcel.xls 파일을 생성하고 아래 그림과 같은 내용이 기록됩니다.
작업한 시트의 이름은 LusasOutPut이 되어 있습니다.





■ 내용 분석

```
$ENGINE=VBSCRIPT
' Open Excel application
    set ExcelApp = CreateAutomationObject("Excel.Application")

' 생성시킬 파일 이름과 시트 이름을 정의
    FilePath="c:\lusas13\projects\"
    FileName="LusasExcel"
    SheetName="LusasOutPut"

' Excel 파일 생성
    ExcelApp.DefaultFilePath = FilePath                    : 엑셀의 작업디렉토리 지정
    Set WorkBook=ExcelApp.WorkBooks.Add                    : 새로운 엑셀파일 생성
    WorkBook.Title=FileName                                : 생성된 파일의 이름을 부여

' 생성된 파일 내 작업 시트 정의
    set WorkSheet= WorkBook.WorkSheets(1)
            : 생성된 파일의 첫 번째 시트를 WorkSheet라는 Object로 정의
    WorkSheet.Name=SheetName                               : 시트에 제목 지정

' 내용 편집
    WorkSheet.Cells(1,1).Value = "A열 1행"                 : A열1행에 값을 기록
    WorkSheet.Cells(1,2).Value = ": Sample입니다." : B열1행에 값을 기록
    WorkSheet.Cells(2,1).Value = "A열 2행"
    WorkSheet.Cells(2,2).Value = ": 잘되나요."

' 편집한 파일을 저장
    Fname=FileName + ".xls"
    WorkBook.SaveAs(FName)

' Excel 사용 종료
    ExcelApp.Quit
    Set ExcelApp = Nothing
```

14.2. 기존 파일을 열고 읽거나 편집하고자 할 경우

■ 스크립트 예

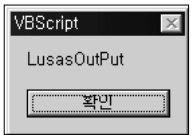
test011fromExcel.vbs

```
$ENGINE=VBSCRIPT

' Open Excel application
   Set ExcelApp=CreateAutomationObject("Excel.Application")

' 불러올 파일명 정의
   FilePath="c:\lusas13\projects\"
   FileName="LusasExcel"

' Excel 파일 불러오기
   ExcelApp.DefaultFilePath=FilePath
   Set WorkBook=ExcelApp.WorkBooks.Open(FileName)

' 사용할 Excel 시트 지정
   Set WorkSheet=WorkBook.WorkSheets(1)
   SheetNameOpened=WorkSheet.Name
   MsgBox SheetNameOpened

' 작업 중인 시트의 값 읽기
   sampletext=WorkSheet.Cells(1,1).text
   MsgBox sampletext

' 작업 중인 시트에 값을 수정/추가기록 하기
   WorkSheet.Cells(1,2).Value = "B열 1행입니다."
   WorkSheet.Cells(4,1).Value = "수정이 되어 저장된 것이 확인되었습니까"

' 편집한 파일을 저장
    NewFileName="LusasExcelModi"
    Fname=NewFileName + ".xls"
    WorkBook.SaveAs(FName)
    Workbook.close

' Excel 사용 종료
    ExcelApp.Quit
    Set ExcelApp = Nothing
```

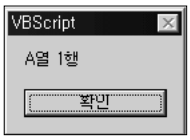
■ 결과

위에서 지정한 디렉토리에 있는 LusasExcel.xls 파일을 열고

첫 번째 시트의 제목을 화면에 표시하고,



A열 1행의 값을 읽어 들여 그 내용을 화면에 표시한 후



파일의 내용을 일부 수정하여 아래와 같이 되며

	A	B	C	D
1	A열 1행	B열 1행입니다.		
2	A열 2행	: 잘되나요.		
3				
4	수정이 되어 저장된 것이 확인되었습니까			
5				

LusasExcelModi.xls로 이름을 변경하여 저장.

■ 내용 분석

```
$ENGINE=VBSCRIPT

' Open Excel application
   Set ExcelApp=CreateAutomationObject("Excel.Application")

' 불러올 파일명 정의
   FilePath="c:\lusas13\projects\"
   FileName="LusasExcel"

' Excel 파일 불러오기
   ExcelApp.DefaultFilePath=FilePath
   Set WorkBook=ExcelApp.WorkBooks.Open(FileName)
            : 디폴트 디렉토리내에서 지정한 파일을 읽어들이기


' 사용할 Excel 시트 지정
   Set WorkSheet=WorkBook.WorkSheets(1)
            : 첫 번째 시트를 WorkSheet라는 Object로 정의
   SheetNameOpened=WorkSheet.Name          : 작업을 하고자 하는 시트의 이름 확인
   MsgBox SheetNameOpened                  : 시트 이름을 메시지창으로 보여주기


' 작업 중인 시트의 값 읽기
   sampletext=WorkSheet.Cells(1,1).text    : 지정한 셀의 값을 읽어내기
   MsgBox sampletext

' 작업 중인 시트에 값을 수정/추가기록 하기
   WorkSheet.Cells(1,2).Value = "B열 1행입니다."
   WorkSheet.Cells(4,1).Value = "수정이 되어 저장된 것이 확인되었습니까"


' 편집한 파일을 저장
    NewFileName="LusasExcelModi"
    Fname=NewFileName + ".xls"
    WorkBook.SaveAs(FName)
    Workbook.close
            : 이 작업을 하지 않으면 이름 변경되기 전의 파일이 작업이 완료되지 못하고 메모
리에 계속 상주하고 있어 오류의 원인이 됩니다.
```

```
' Excel 사용 종료
    ExcelApp.Quit
    Set ExcelApp = Nothing
```

14.3. 여러 개의 시트를 대상으로 작업하고자 할 경우

여러개의 시트를 대상으로 작업을 하는 경우는 아래와 같이 해당 시트를 별개의 Object로 정의하여 활용하면 됩니다. WorkSheets(i)라는 Method는 지정된 Excel파일에서 I번째에 해당하는 시트를 가리킵니다.

set WorkSheet1 = WorkBook.Worksheets(1)

set WorkSheet2 = WorkBook.Worksheets(2)

아래에서 그 사용예를 기술하되, 정의를 보다 일반적이고 간편하게 하기 위하여 개별적으로 시트의 Object명을 구분하는 대신 배열을 사용하였습니다.

■ 스크립트 예

test011toExcelSheets.vbs

```
$ENGINE=VBSCRIPT
' Open Excel application
    set ExcelApp = CreateAutomationObject("Excel.Application")

' 생성시킬 파일 이름과 시트 이름을 정의
    FilePath="d:\lusas13\projects\"
    FileName="LusasExcelsheets"
    SheetCount=2
    ReDim SheetName(sheetCount)
    ReDim WorkSheet(SheetCount)
    SheetName(1)="LusasOutPut1"
    SheetName(2)="LusasOutput2"

' Excel 파일 생성
    ExcelApp.DefaultFilePath = FilePath
    Set WorkBook=ExcelApp.WorkBooks.Add
    WorkBook.Title=FileName

' 생성된 파일 내 작업 시트 정의
    ReDim WorkSheet(SheetCount)
    For i=1 to SheetCount
            set WorkSheet(i) = WorkBook.Worksheets(i)
      WorkSheet(i).Name = sheetName(i)
    Next
' 내용 편집
    WorkSheet(1).Cells(1,1).Value = "첫번째 시트의 A열 1행"
    WorkSheet(1).Cells(1,2).Value = ": Sample 입니다."
    WorkSheet(1).Cells(2,1).Value = "첫번째 시트의 A열 2행"
    WorkSheet(1).Cells(2,2).Value = ": 잘되나요."

    WorkSheet(2).Cells(1,1).Value = "두번째 시트의 A열 1행"
    WorkSheet(2).Cells(1,2).Value = ": Sample 입니다."
    WorkSheet(2).Cells(2,1).Value = "두번째 시트의 A열 2행"
    WorkSheet(2).Cells(2,2).Value = ": 잘되나요."
' 계속 ...
```

```
' 편집한 파일을 저장
    Fname=FileName + ".xls"
    WorkBook.SaveAs(FName)

' Excel 사용 종료
    ExcelApp.Quit
    Set ExcelApp = Nothing
```

■ 결과

생략

■ 내용분석

생략

15. Geometry 정의 관련 Method

15.1. 종류 및 서식

Method 서식	산출물 / 결과
createPoint(x, y, z 좌표)	새로 생성한 Point Object
countPoints()	모델링 된 Point 총 개수
getPointByNumber(포인트번호)	지정한 Point Object
getPoint(인덱스(0th, 1st, 2nd,...))	지정한 Point Object
createLine(Point 또는 Line Object, 직선형태("Straight"/"Spline")	새로 생성한 Line Object
createLineByCoordinates (x1,y1,z1,x2,y2,z2 좌표)	새로 생성한 Line Object
createLineByPoints(Point Object, Point Object)	새로 생성한 Line Object
createArc(3개의 Point Object(시작, 중간, 끝)를 표현하는 배열)	새로 생성한 Line Object
countLines()	모델링 된 Line 총 개수
getLineByNumber(선번호)	지정한 Line Object
getLine(인덱스(0th, 1st, 2nd,...))	지정한 Line Object
createSurface(필요한 수의 Point Objects나 Line Objects 배열)	새로 생성한 Surface Object
countSurfaces()	모델링 된 Surface 총 개수
getSurfaceByNumber(면번호)	지정한 Surface Object
getSurface(인덱스(0th, 1st, 2nd,...))	지정한 Surface Object
createVolume(필요한 수의 Point, Line 또는 Surface Objects 배열)	새로 생성한 Volume Object
countVolumes()	모델링 된 Volume 총 개수
getVolumeByNumber(체적번호)	지정한 Volume Object
getVolume(인덱스(0th, 1st, 2nd,...))	지정한 Volume Object
getLargestPointNumber()	모델링 내 가장 큰 Point 번호
getLargestLineNumber()	모델링 내 가장 큰 Line 번호
getLargestSurfaceNumber()	모델링 내 가장 큰 Surface 번호
getLargestVolumeNumber()	모델링 내 가장 큰 Volume 번호
GetTypeCode()	현재 Object의 종류에 해당하는 식별 번호 1-point 2-line 4-surface 5-volume 6-node 9-element 13-group

15.2. 예제 실습

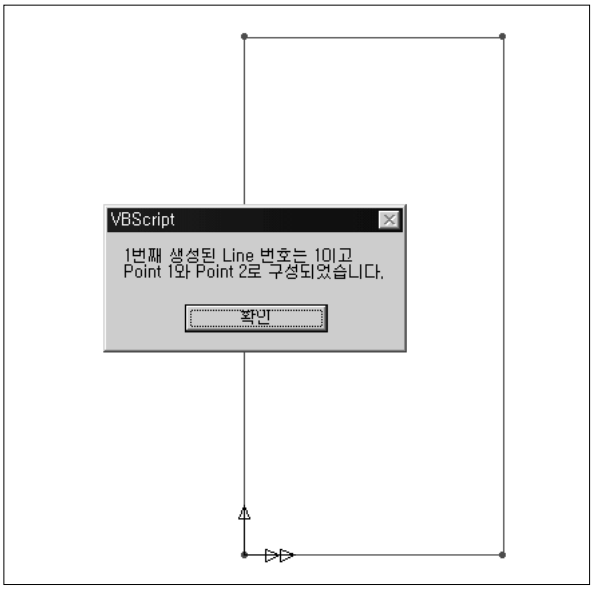
■ 스크립트 예

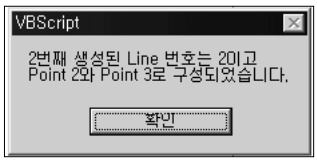
test012geo1.vbs

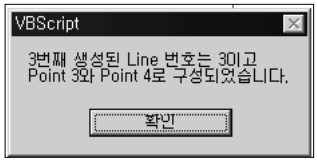
```
$ENGINE=VBSCRIPT
Set database=getDatabase()
Dim points(3)
Dim lines(3)
Dim surface1

width=10
height=20
Set points(0)=database.createPoint(0,0,0)
Set points(1)=database.createPoint(width,0,0)
Set points(2)=database.createPoint(width,height,0)
Set points(3)=database.createPoint(0,height,0)

Set surface1=database.createSurface(points)


For I=0 to 3
 Set lines(i)=database.getLine(i)
 linenumber=lines(i).getNumber()
 Set Point1=lines(i).getLOF(0)
 Set Point2=lines(i).getLOF(1)
 string1=(i+1)&"번째 생성된 Line 번호는 "&lineNumber&"이고"
 string2="Point "&point1.getNumber()&"와 Point "&point2.getNumber()&"로 구성되
었습니다."
 MsgBox string1&vbNewLine&string2
Next

NumberOfLines=database.countLines()
MsgBox "현재 모델 파일에 있는 총 Line의 갯수는 "& NumberOfLines &"개입니
다."
```

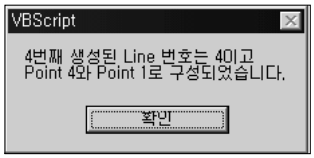
■ 결과

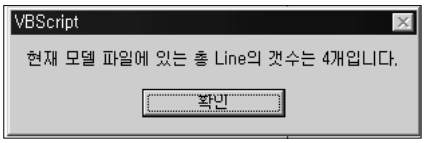
위 내용을 test012geo1.vbs로 저장하고, 이것을 Modeller에서 실행을 시키면 아래 그림과 같이 Modeller에 10X20 크기의 Surface가 정의되고, 4개의 메시지창을 통해 생성된 Line의 번호와 Line을 구성하는 Pont들의 번호를 알려주며, 마지막으로 현재 Modeller에 정의되어 있는 Line의 총개수를 표시합니다.











■ 내용 분석

```
$ENGINE=VBSCRIPT
Set database=getDatabase()
```
: 현재 Modeller 작업 데이터베이스를 database라는 Object로 정의

```
Dim points(3)
Dim lines(3)
Dim surface1
width=10
height=20

Set points(0)=database.createPoint(0,0,0)
Set points(1)=database.createPoint(width,0,0)
Set points(2)=database.createPoint(width,height,0)
Set points(3)=database.createPoint(0,height,0)
```
: Modeller(database라는 Object)에 4개의 Point 정의
정의된 Point는 points()라는 배열변수형태의 Object로 보관.

```
Set surface1=database.createSurface(points)
```
: points라는 배열변수가 가진 4개의 Point Objects들, 즉 points(0), points(1), points(2), points(3)로 구성된 Surface정의하고, 정의된 Surface의 속성은 Surface1이라는 이름의 Object로 보관.

```
For I=0 to 3
 Set lines(i)=database.getLine(i)
```
: database.getLine(0)은 Modeller에서 처음 정의된 Line을 의미합니다. 즉 0~3까지 처음 정의된 4개의 Line을 lines()라는 Object로 명명해 두겠다는 것입니다.

```
 linenumber=lines(i).getNumber()
```
: getNumber()는 해당 Object가 Modeller에서 인식되는 번호를 얻겠다는 것입니다.
예를 들어 lines(0)는 앞에서 Modeller에서 최초에 생성된 Line을 의미하는 것이므로 lines(0).getNumber()는 그 Line의 번호가 됩니다.

```
 Set Point1=lines(i).getLOF(0)
```
: lines(0).getLOF(0)은 lines(0)이 가리키는 Line을 구성하는 하위구조, 즉 Point 들 중에서 Line정의에 처음 사용된 Point Object를 의미합니다.

```
 Set Point2=lines(i).getLOF(1)
```
: lines(0).getLOF(1)은 lines(0)이 가리키는 Line을 구성하는 하위구조, 즉 Point 들 중에서 Line정의에 두 번재 사용된 Point Object를 의미합니다.

```
 string1=(i+1)&"번째 생성된 Line 번호는 "&lineNumber&"이고"
 string2="Point "&point1.getNumber()&"와 Point "&point2.getNumber()&"로 구성되었습니다."
 MsgBox string1 & vbNewLine & string2
```
: 생성되는 Line의 순서대로, 그 Line의 번호와, 그 Line을 구성하는 Point들의 번호를 메시지창으로 보여주게 합니다. vbNewLine은 줄바꿈을 의미하는 문자열상수로 VBScript가 이미 내부적으로 가지고 있으므로 별도의 정의 없이 사용할 수 있습니다.

```
Next

NumberOfLines=database.countLines()
MsgBox "현재 모델 파일에 있는 총 Line의 갯수는 "& NumberOfLines &"개입니다."
```

16. Geometry 통제 및 활용을 위한 Method

16.1. Geometry (Point/Line/Surface/Volume 공통) 통제용

Method 서식	산출물 / 결과
countLOF()	현 Geometry를 구성하는 하위 Geometry의 개수
getLOFindex (0th,1st, 2nd, 3rd etc.)	지정한 하위 Geometry Object
countHOF()	현 Geometry가 구성하고 있는 상위 Geometry개수
getHOF(index (0th,1st, 2nd, 3rd) etc.)	지정한 상위 Geometry Object
countAttributes(type$)	현 Geometry에 적용된 Attribute type$의 개수
getAttribute(index(0th,1st, 2nd, 3rd etc.),type$)	현 Geometry에 적용된 Attribute type$ 중 지정한 순번의 Attribute Object
setPen(pen번호)	현 Geometry의 색깔 변경
getNumber()	현 Geometry Object의 모델러 내 표기 번호
isExcludedFromAdaptivity()	현 Geometry가 최적화 대상이 아니면 True

16.2. Geometry / Point 통제용

Method 서식	산출물 / 결과
getX()	현재 Point의 X 좌표값
getY()	현재 Point의 Y 좌표값
getZ()	현재 Point의 Z 좌표값
getXYZ(변수명1,변수명2,변수명3)	현재 Point의 XYZ 좌표값을 인수에 있는 변수에 할당
moveTo(x,y,z)	지정 위치 이동 작업
moveBy(Dx, Dy, Dz)	지정 간격 이동 작업
copy(Dx, Dy, Dz)	지정 간격 복사 작업
countNodes()	Point내 Node의 개수
getNode(index (0th,1st, 2nd, 3rd etc.))	Point내 지정한 순번의 Node Object

16.3. Geometry / Line 통제용

Method 서식	산출물 / 결과
copy(Dx, Dy, Dz)	지정 간격 복사 작업 후 생성된 Line Object
getLineType()	현재 Line의 형태를 표시하는 수 1-straight 2-arc 3-spline 4-combined line
countElements()	현재 Line내 Element의 개수
getElement(index (0th,1st, 2nd, 3rd etc.))	현재 Line내 해당 순번에 해당하는 Element Object
countNodes()	현재 Line내 Node의 개수
getNode(index (0th,1st, 2nd, 3rd etc.))	현재 Line내 해당 순번에 해당하는 Node Object
getArcAngle()	라인형태가 2번 Arc일 경우 그 중심각, 아니면 0
getArcRadius()	라인형태가 2번 Arc일 경우 그 반경, 아니면 0
getArcCentreX()	라인형태가 2번 Arc일 경우 중심좌표, 아니면 0
getArcCentreY()	라인형태가 2번 Arc일 경우 중심좌표, 아니면 0
getArcCentreZ()	라인형태가 2번 Arc일 경우 중심좌표, 아니면 0

16.4. Geometry / Surface 통제용

Method 서식	산출물 / 결과
copy(Dx, Dy, Dz)	지정 간격 복사 작업 후 생성된 Surface Object
countElements()	현재 Surface내 Element의 개수
getElement(index (0th,1st, 2nd, 3rd etc.))	현재 Surface내 해당 순번에 해당하는 Element Object
countNodes()	현재 Surface내 Node의 개수
getNode(index (0th,1st, 2nd, 3rd etc.))	현재 Surface내 해당 순번에 해당하는 Node Object

16.5. Geometry / Volume 통제용

Method 서식	산출물 / 결과
copy(Dx, Dy, Dz)	지정 간격 복사 작업 후 생성된 Volume Object
countElements()	현재 Volume내 Element의 개수
getElement(index (0th,1st, 2nd, 3rd etc.))	현재 Volume내 해당 순번에 해당하는 Element Object
countNodes()	현재 Volume내 Node의 개수
getNode(index (0th,1st, 2nd, 3rd etc.))	현재 Volume내 해당 순번에 해당하는 Node Object

17. Attribute 및 결과처리 관련 Method

17.1. Attribute 정의 및 인식 관련

Method 서식	산출물 / 결과
countNodes()	모델링 된 Node의 총 개수
getNodeByNumber(Node번호)	지정한 Node Object
getNode(인덱스(0th, 1st, 2nd,...))	지정한 Node Object
countElements()	모델링 된 Element의 총 개수
getElementByNumber(Element번호)	지정한 Element Object
getElement(인덱스(0th, 1st, 2nd,...))	지정한 Element Object
getLargestAttributeNumber(type$)	모델링 내 가장 큰 type$ 데이터셋 번호
getFirstAttributeByNumber(type$)	모델링 내 가장 작은 type$ 데이터셋 번호
getNextAttributeByNumber(type$,번호)	지정 번호 이후 가장 작은 type$ 데이터셋 번호
existsAttribteByNumber(번호,type$)	지정한 번호의 type$이 있으면 True
existsAttributeByName(이름$,type$)	지정한 이름의 type$이 있으면 True
getCurrentLoadcaseNumber()	현재 작업중인 하중케이스 번호
getLargestLoadCaseNumber()	모델링 내 가장 큰 하중케이스 번호
getMaterialDatasets(배열)	모델링 내에 적용된 각 물성데이타셋의 개수를 표시한 정수 배열

▶ 위 Method의 인수로써 표시한 type$ 의 종류

MESH	LOADING	LOADING TABLE	SUPPORT	TRANSFORMATION
LOCAL COORDINATE	DAMPING TABLE	VARIATION	LOAD CURVE	CONSTRAINTS
SEARCH AREA	RETAINED FREEDOMS	SLIDELINE PROPERTIES	SLIDELINE TABLE	EQUIVALENCE
THERMAL SURFACE	THERMAL GAP	RADIATION SURFACE	ACTIVATE	DEACTIVATE
DAMPING PROPERTIES	INFLUENCE LINE	THERMAL GAP PROPERTIES	THERMAL ENVIRONMENT PROPERTIES	THERMAL RADIATION PROPERTIES
THERMAL CONTACT	GEOMETRY	MATERIAL	COMPOSITE	BACKGROUND GRID

☞ 사용예

```
LastMeshDatasetNumber = database.getLargestAttributeNumber("MESH")
```

17.2. Mesh (Element/Node 공통) 통제용

Method 서식	산출물 / 결과
getNumber()	Mesh Dataset 번호
isVisible()	현재 Mesh가 화면상에 보이는 설정이면 True

17.3. Node 통제용

Method 서식	산출물 / 결과
getX()	현재 Node의 X 좌표값
getY()	현재 Node의 Y 좌표값
getZ()	현재 Node의 Z 좌표값

17.4. Element 통제용

Method 서식	산출물 / 결과
countNodes()	현재 Element내 Node의 개수
countGaussPoints()	현재 Element내 Gauss Points의 개수
getNode(index (0th,1st, 2nd, 3rd etc.))	현재 Element내 지정 순번의 Node Object
GetComputationalWeight()	정수
getFeature()	현재 Element를 생성한 Feature Object
getSlidelineNumber()	현재 Element에 적용되어 있는 Slideline 번호

17.5. Node 결과의 활용

Method 서식	산출물 / 결과
getResults(wha$t,column)	현재 Node의 지정한 결과값
getLayerResults(what$layername$, column)	현재 Node의 지정한 결과값
getVectorResults(what$)	현재 Node의 지정한 벡터 결과값
getVectorLayerResults(what$layer, name$)	현재 Node의 지정한 벡터 결과값
getMatrixResults(what$)	현재 Node의 지정한 Matrix 결과값
getMatrixLayerResults(what$layer, name$)	현재 Node의 지정한 Matrix 결과값
setUserResults(value)	
getActivePosition()	

■ 결과값 활용을 위한 Method에서 What$ / What$Layer의 의미

Modeller의 결과치 선택 시 Entity에 해당하는 내용으로 아래와 같습니다.

STRESS	STRESS (TOP)	STRESS (MIDDLE)	STRESS (BOTTOM)
FLUX	FLUX (TOP)	FLUX (MIDDLE)	FLUX (BOTTOM)
STRAIN	STRAIN (TOP	STRAIN (MIDDLE)	STRAIN (BOTTOM)
GRADIENT	GRADIENT (TOP)	GRADIENT (MIDDLE)	GRADIENT (BOTTOM)
PLASTIC STRAIN	PLASTIC (TOP)	PLASTIC (MIDDLE)	PLASTIC (BOTTOM)
CREEP STRAIN	CREEP (TOP)	CREEP (MIDDLE)	CREEP (BOTTOM)
STRETCH	STRETCH (TOP)	STRETCH (MIDDLE)	STRETCH (BOTTOM)
DISPLACEMENT	POTENTIAL	LOADING	REACTION
RESIDUAL	VELOCITY	ACCELERATION	SLIDE RESULTS
THERMAL RESULTS	REACTION STRESS		

■ 결과값 활용을 위한 Method에서 name$의 의미

Modeller에서 결과치 선택 시 Component에 해당하며, 해석에 사용한 Element의 종류에 따라 다르나 Thick Plate를 사용하였을 경우를 예로 들면 아래와 같습니다.

MX	MY	MXY	Sx	Sy	Mmax	MMin	MI
BETA	ME	Mx(T)	My(T)	Mx(B)	My(B)	StEngD	Nabs

■ 벡터 출력 양식

What=stress 또는 Strain 또는 Stretch일 경우에만 가능한 출력양식이며, What=stress이고 사용된 Element 가 Thick Plate일 때 벡터 결과는 아래의 서식과 같습니다.

(MX, MY, MXY, Sx, Sy)T

■ Matrix 출력 양식

What=stress 또는 Strain 또는 Stretch일 경우에만 가능한 출력양식이며, What=stress이고 사용된 Element 가 Thick Plate일 때 벡터 결과는 아래의 서식과 같습니다.

MX	MXY	0	0
MXY	MY	0	0
0	0	Sx	0
0	0	0	Sy

17.6. Element 결과의 활용

Method 서식	산출물 / 결과
getGaussResult(gaussPoint index(0th,1st..), what$, column)	현재 Element의 지정 순번에 해당하는 Gauss의 지정 결과값
getGaussLayerResults(gaussPoint index(0th,1st..), what$layername$, column)	현재 Element의 지정 순번에 해당하는 Gauss의 지정 결과값
getGaussVectorResults(gaussPoint index(0th,1st..), what$)	현재 Element의 지정 순번에 해당하는 Gauss의 지정 결과 벡터
getGaussVectorLayerResults(gaussPoint index(0th,1st..), what$layer, name$)	현재 Element의 지정 순번에 해당하는 Gauss의 지정 결과 벡터
getGaussMatrixResults(gaussPoint index(0th,1st..), what$)	현재 Element의 지정 순번에 해당하는 Gauss의 지정 결과 Matrix
getGaussMatrixLayerResults(gaussPoint index(0th,1st..), what$layer, name$)	현재 Element의 지정 순번에 해당하는 Gauss의 지정 결과 Matrix

17.7. Vector 결과의 활용

Method 서식	산출물 / 결과
getCountValues()	Vector 결과 항목의 개수
getIValue(열번호)	지정한 열에 해당하는 결과값

17.8. Matrix 결과의 활용

Method 서식	산출물 / 결과
getCountRows()	Matrix 결과의 행 수
getCountColumns()	Matrix 결과의 열 수
getIJValue(i,j), I는 열 번호, j는 행번호	Matrix결과 내 지정 위치의 결과값