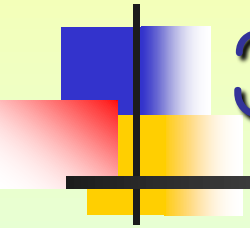


3장 비주얼베이직 6.0 문법

3-2 비주얼베이직 6.0 기본 문법

- TimeValue() 함수
  - DateValue() 함수와 마찬가지로 일정 규칙에 정해진 문자열을 인자로 하여 Time 함수형을 표시

```
Print TimeValue("13:2")
Print TimeValue("1:2 PM")
Print TimeValue("13:02:00")
결과값 : 1:02:00  오후
```

- **MagBox()와 InputBox() 함수**
  - MsgBox() 함수 : 대화상자에 메시지를 표시하는 함수로 사용자가 선택한 버튼의 정수값을 리턴
    - MsgBox() 함수의 리턴값

상 수	값	설 명
vbOK	1	[확인] 버튼을 눌렀을 경우
vbCancel	2	[취소] 버튼을 눌렀을 경우
vbAbout	3	[중단] 버튼을 눌렀을 경우
vbRetry	4	[재시도] 버튼을 눌렀을 경우
vbIgnore	5	[무시] 버튼을 눌렀을 경우
vbYes	6	[예] 버튼을 눌렀을 경우
vbNo	7	[아니오] 버튼을 눌렀을 경우

3-2 비주얼베이직 6.0 기본 문법

- MsgBox 명령
  - 메시지 박스 함수와 동일한 기능을 하지만 리턴되는 값은 없다.

```
MsgBox(메시지 [, 버튼상수] [, 타이틀] [, 도움말 항목 , 도움말 항목의 번호])
```

  - 메시지 : 사용자에게 전달할 메시지
  - 타이틀 : 메시지 박스의 타이틀에 표시된 문자열
  - 버튼상수 : 박스에 표시될 버튼의 종류. 버튼과 아이콘 또는 기타 상수를 함께 사용하려면 각 해당 상수를 더해서 사용

종 류	상 수	값	설 명
버튼	vbOKOnly	0	[확인] 버튼
	vbOkCancel	1	[확인], [취소] 버튼
	vbAbortRetryIgnore	2	[중단], [재시도], [무시] 버튼
	vbYesNoCancel	3	[예], [아니오], [취소] 버튼
	vbYesNo	4	[예], [아니오] 버튼
	vbRetryCancel	5	[재시도], [취소] 버트
아이콘	vbCritical	16	치명적 메시지 아이콘
	vbQuestion	32	질의 경고 메시지 아이콘
	vbExclamation	48	경고 메시지 아이콘
	vbinformation	64	정보 메시지 아이콘

3-2 비주얼베이직 6.0 기본 문법

- MsgBox 명령

종 류	상 수	값	설 명
기타	vbDefaultButton1	0	첫 번째 버튼을 기본값으로
	vbDefaultButton2	256	두 번째 버튼을 기본값으로
	vbDefaultButton3	512	세 번째 버튼을 기본값으로
	vbDefaultButton4	768	네 번째 버튼을 기본값으로
	vbApplicationModal	0	응용 프로그램에서의 모달 속성을 가짐
	vbSystemModal	4096	시스템에서의 모달 속성을 가짐

- InputBox() 함수
  - InputBox() 함수는 메시지 박스와는 다르게 사용자의 입력을 필요로 할 때 사용하는 대화상자이다. 리턴값은 사용자가 입력한 문자열

```
MsgBox(메시지 [, 타이틀] [, 기본값] [, xpos] [, ypos] [, helpfile, context])
```

  - 메시지 : 사용자에게 전달할 메시지
  - 타이틀 : 대화상자에 타이틀에 표시된 문자열
  - 기본값 : 대화상자안의 입력 텍스트 박스에 기본적으로 표시될 문자열

3-2 비주얼베이직 6.0 기본 문법

- **연산자의 종류와 우선 순위**

산술 연산자		관계 연산자	
-	음수를 표현 밸셈	<	보다 작다
+	덧셈	<=	보다 작거나 같다
/	부동 소수 나눗셈	>	보다 크다
\	정수 나눗셈	>=	보다 크거나 같다
*	곱셈	<>	같지 않다
^	누승	=	같다 오른쪽 항을 왼쪽 항에 대입
Mod	나머지 연산	Ls	두 개의 객체 변수가 동일한 객체를 지칭하는지를 비교
&	문자열 연결	Like	패턴일치 비교

논리연산자	Not	논리적 부정 : 참 -> 거짓
	And	논리곱 : 둘 다 참인 경우만 참
	Or	논리합 : 둘 다 거짓인 경우만 거짓
	Xor	배타적 논리합 : 서로의 논리값이 서로 다른 경우만 참
	Equ	양 쪽 두 식의 논리값이 서로 같으면 참
	Imp	첫 번째 항이 참 이고 두 번째 항이 거짓일 대 거짓

3-2 비주얼베이직 6.0 기본 문법

- **연산자의 종류와 우선 순위**

연산자 우선순위		
	^	누승
	-	음수
	*/	곱셈, 나눗셈
	\	정수 나눗셈
	Mod	나머지 연산
	+-	덧셈, 뺄셈

예제)

```
Private Sub Foum_Paint()
    X = 123
    Y = x ^ 3 * 2 / 3 + 10
Print y
End Sub
```

결과값 : 1240588

```
Private Sub Foum_Paint()
    X = 123
    Y = (((x ^ 3) * 2) / 3) + 10
Print y
End Sub
```

결과값 : 1240588

3-2 비주얼베이직 6.0 기본 문법

- **제어문**
  - 조건 비교문 if : 조건을 비교하여 다른 명령을 수행하고자 할 때 if문을 사용
    - If ~ then ~ End If

```
If 조건 then 처리 내용
```

If 문에서의 조건은 관계 연산자가 주로 사용된다.

관계 연산자	사용예	의 미
=	x = y	x는 y와 같다
>	x > y	x는 y보다 크다
<	x < y	x는 y보다 작다
>=	x >= y	x는 y보다 크거나 같다
<=	x <= y	x는 y보다 작거나 같다
<>	x <> y	x는 y는 같지 않다

If 문에서 조건이 참일 때, 두 가지 이상의 명령을 수행하고자 할 때

```
If 조건 then
   처리 내용1
   처리내용2
   ....
End If
```

3-2 비주얼베이직 6.0 기본 문법

- If ~ then ~ Else ~ End If

  If 문에서의 조건이 맞지 않을 경우의 처리는 Else문을 사용하여 처리

  If 문은 단지 조건을 가지고 그것이 참이냐 거짓이냐에 따라 다른 내용을 수행하도록 해주는 제어문

```
If 조건 then
   처리 내용1
   처리내용2
Else
   처리내용3
   처리내용4
End If
```

조건이 둘 이상인 제어구조를 원한다면 Elseif를 사용

```
If 조건 then
   처리 내용1
   처리내용2
Elseif 조건2 then
   처리내용3
   처리내용4
Else ...
   처리내용5
End If
```

3-2 비주얼베이직 6.0 기본 문법

- For ~ Next 문
  - For ~ Next 는 지정한 횟수만큼 구문을 반복실행 하게 하는 명령어
  - 카운터(count)의 값은 step이 양수인지 음수인지에 따라 늘어나기도 하고 줄어들기도 한다.
  - For ~ Next 문이 아직 끝나지 않았지만 루프를 종료해야 할 경우도 있는데 이때는 루프안에서 Exit For 를 사용

```
For 카운트변수 = 초기식 To 정지식 [Stop 증감식]
        [반복될 명령]
Next [카운트변수]
```

  - 프로그램이 실행 중 For 문을 만나면

변수 값이 시작값으로 세팅된 후, For 문과 Next 문 사이에 있는 명령을 실행하고 Next 문을 만나면 변수값을 1만큼 증가시킨 후, 변수값이 끝값과 같거나 크면 순환을 멈추고, 그렇지 않으면 다시 For 문으로 돌아가 명령을 반복한다.

  - For ~ Next는 단순히 동일한 명령을 반복하는 데 그치지 않고, 변경하는 변수 값을 프로그램에 반영할 수 있다는 장점

3-2 비주얼베이직 6.0 기본 문법

- Do ~ Loop 문
  - Do ... Loop Until ... 문

```
Do
   ... 반복할 내용 ...
Loop Until 조건
```

  - For Loop는 어떤 횟수만큼 실행하고 Do Loop는 어떤 조건이 참일 경우만 실행
  - For ... Next문은 카운터 변수가 있어서 정해진 횟수만큼 반복하면 되지만, Do .. Loop문에는 카운터 변수가 없기 때문에 언제 순환을 끝낼 것인가를 결정
    이 순환을 종료하는 상황을 Until 뒤에 따라오는 조건에 부여

```
Do
   ...
   ...
  Loop Until sum > 1000
```

  - Do Unitl ... Loop 문
  - Do Until ... Loop 문은 Do ... Loop Until문과 동일한 기능을 수행하지만, 조건을 비교하는 위치가 뒤냐 앞이냐의 차이가 있다.
  - 차이점은 Do ... Loop Until 문은 순환문이 적어도 한 번은 수행. 이는 조건을 비교하는 Until 구문이 뒷부분에 위치하고 있기 때문

3-2 비주얼베이직 6.0 기본 문법

- Do ... Loop While ... 문
  - Do ... Loop While ... 문은 Until 조건문을 사용한 Do 순환문과 비슷하지만, 조건을 비교하는 방식이 다르다.

```
Do
   ... 반복할 내용 ...
Loop While 조건
```

- Do While ... Loop 문
  - Do ... Loop Until 문과 Do Until ... Loop 문의 차이처럼, Do While ... Loop 문은 조건을 먼저 비교한 후, 순환 여부를 결정

```
X = 0
   Do While X < 10
         Print X
         X = X + 1
   Loop
결과값 : 없음
```

3-2 비주얼베이직 6.0 기본 문법

- While ~ Wend 문
  - While ~ Wend 문은 Do ~ Loop문 보다 덜 강력한 명령이다.
  - Do ~ Loop문은 Exit Do로 종료할 수 있지만 While ~ Whend문은 조건이 참 일 동안 루프를 계속 수행

```
While 조건식
    [ 반복할 명령 ]
Wend
```

- Select Case 문
  - 여러 개의 If ~ Then 구조를 사용할 때 혼란을 야기할 수도 있게 된다. 이때 Select Case문을 쓰는 것이 훨씬 용이하다. Select Case문은 어떤 대상을 일치 가능한 모든 목록과 비교

```
Select Case 변수
  Case 값1
     처리1
  Case 값2
     처리2
  Case 값3
     처리3
     ....
End Select
```

3-2 비주얼베이직 6.0 기본 문법

- Select Case 문
  - Select Case 문에서 여러 개의 값에 동일한 처리를 하고 싶으면 콤마(,)로 묶어서 나타낸다.
  - 범위를 비교하기 위해서는 비교 연산자를 사용할 수 있다. (=. <>, >, <. >=, <=)
  - Is 키워드와 To 키워드를 사용

키워드	사용예	용 도
Is	Case Is <10	변수의 값을 Is 다음의 식과 비교한다. 왼쪽의 사용 예는 변수 값이 10 미만일 때를 의미
To	Case 10 To 20	변수 값의 범위를 정한다. 왼쪽의 사용 예는 변수의 값이 10에서 10 사이일 때를 의미

- GoTo 문
  - GoTo 문은 주어진 프로시저, 함수, 이벤트 프로시저 안에서만 원하는 다른 위치로 프로그램 흐름을 이동
  - 참고로 하나의 프로시저에서 다른 프로시저나 서브 함수 등으로는 이동할 수 없다.

```
GoTo line
....
Line:
```

3-2 비주얼베이직 6.0 기본 문법

- With ~ End With 문
  - 객체의 속성을 참조하는데 있어서 그 객체의 속성을 한꺼번에 참조할 수 있도록 해주는 편리한 기능을 제공
  - 객체의 속성을 지정하거나 메소드를 사용할 때, 때로는 여러 개의 속성을 동시에 변경하거나 여러 번 메소드를 반복해서 사용하는 경우가 있다.
  - 일일히 객체의 이름을 사용하기 보다는 With ... End With 문을 사용하면 코드가 깔끔하고 보기 편리

```
With 객체
      [ 실행명령 ]
End With
```

3-2 비주얼베이직 6.0 기본 문법

- **함수와 프로시저**
  - Return 값의 여부에 따른 분류
    - Sub 프로시저
      - 프로시저는 일정한 기능을 수행하는 코드 명령어. 이전에는 “절차문”이라고도 번역
      - 프로시저는 일을 수행하기 위해서 매개변수를 입력받을 수 있으나 결과 값은 되돌려 받을 수 없다. 결과값을 받고자 한다면 함수를 사용

```
[ Private | Public |Friend | Static] Sub 프로시저명 [ (매개변수) ]
        [ 실행명령 ]
        [ Exit Sub ]
        [ 실행명령 ]
End Sub
```

      - [Private | Public |Friend | Static]는 제한자.
        - 제한자를 사용하지 않을 경우는 내부적으로 public으로 선언

3-2 비주얼베이직 6.0 기본 문법

- 함수
  - 함수는 프로시저와 비슷한 기능을 가지고 있으나 프로시저와는 달리 연산결과의 리턴값을 받을수 있다.

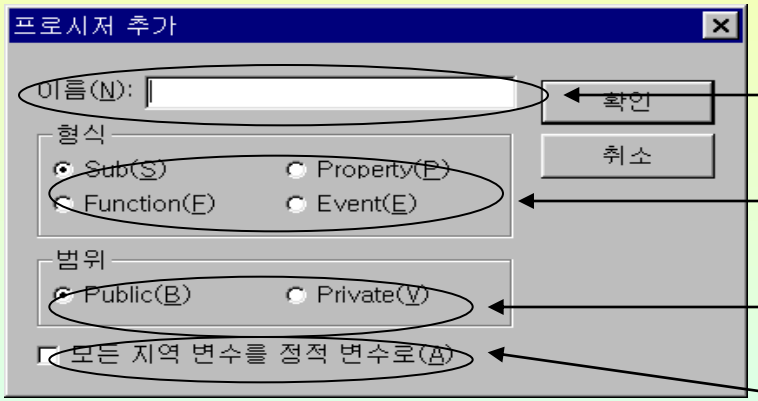
```
[ Private | Public |Friend | Static] Function 함수명 [ (매개변수) ] [ As 데이터 형 ]
        [ 실행명령 ]
        [ Exit Function ]
        [ 실행명령 ]
End Sub
```

  - 함수에서는 연산의 결과값을 돌려주는 방법으로 자신의 이름을 사용. 이것은 함수도 일종의 포인터이기 때문에 가능. 함수를 사용할 경우 함수명 뒤에 리턴되는 데이터의 크기를 명시
  - 프로그램 내부에서는 Public처럼 사용되고 외부에서는 Private처럼 사용되는 것이 Friend란 선언문이다. 단 변수에는 사용할 수 없다.

- Scope(범위)에 따른 분류
  - Public 프로시저
  - Private 프로시저

3-2 비주얼베이직 6.0 기본 문법

- 비주얼 베이직에서 Sub 프로시저와 함수의 선언방법
  - 비주얼베익직의 메뉴항목에서 “도구” 메뉴를 선택한 후 “프로시저 추가” 항목을 선택하게 되면 함수나 Sub 프로시저를 설정할 수 있는 대화상자를 볼 수 있다.



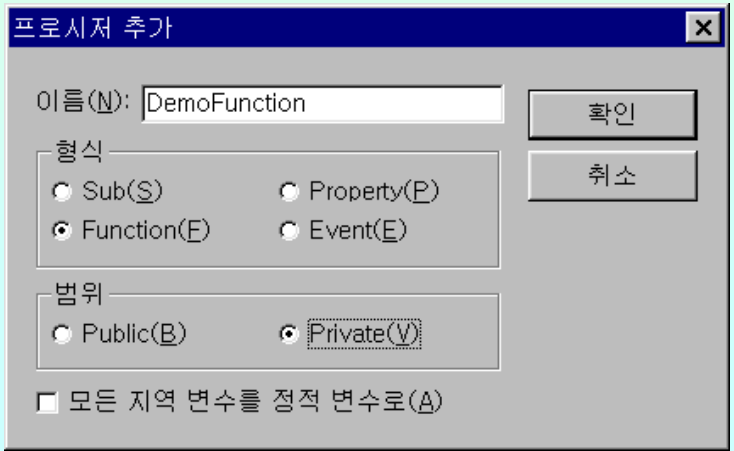
추가할 프로시저의 이름을 설정

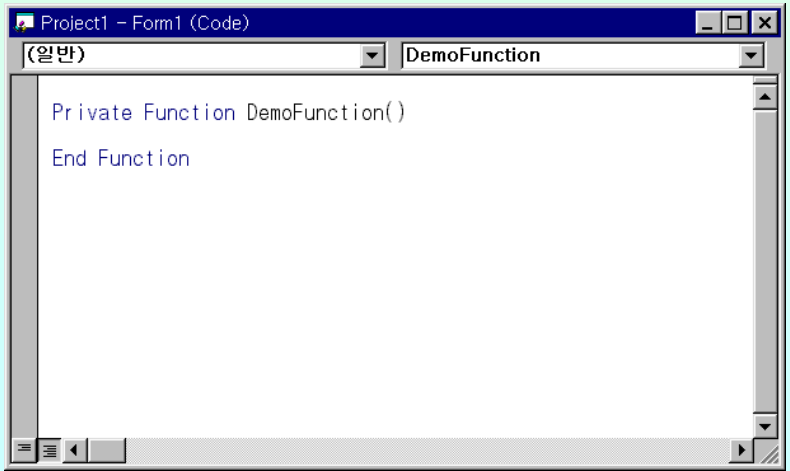
추가할 프로시저의 형식을 설정

추가할 프로시저의 범위를 설정

추가할 프로시저내의 변수를 Static으로 설정

  - “DemoFunction”이란 함수를 생성





3-2 비주얼베이직 6.0 기본 문법

- 매개변수 (파라미터) 전달방법
  - Call By Reference
    - 디폴트 방식
    - 키워드 ByRef사용

```
Private Sub changeProc(X as Integer, Y as Integer)
    Temp = X
    X = Y
    Y = temp
End Sub
```

  - Call By Value
    - 키워드 ByVal 사용
  - 선택적인 인수
    - 키워드 Optional사용

3-2 비주얼베이직 6.0 기본 문법

- **배열**
  - 배열이란 같은 형식의 데이터들로 이루어진 집합
  - 고정길이 배열
    - 처음 배열을 선언할 때 그 크기를 정해 주어야 한다. 선언 방법은 변수 선언과 크게 다르지 않으나 저장할 데이터의 수를 지정
    - 배열을 선언할 때 사용되는 배열의 크기(상한값)를 “첨자”라고 한다.
    - 상한값은 Long 데이터형으로 -2,147,483,648~2,147,483,647의 범위를 벗어날 수 없다.

```
Dim 배열이름(상한값) As 데이터형식
```

    - 배열의 하한값 설정

```
Dim 배열이름(하한값 To 상한값) As 데이터형식
```

    - 다차원 배열(Mulitdimensional Array)

```
Dim KorScore(9) As Integer
Dim EngScore(9) As Integer
```

3-2 비주얼베이직 6.0 기본 문법

- 고정 크기 배열 사용을 위한 제언
  - 저장할 데이터의 용도를 고려
  - 저장할 데이터의 수를 고려

- 동적 배열(Dynamic Array)
  - 고정 크기 배열을 사용할 때, 얼마나 큰 배열을 만들어야 할지 정확하게 알지 못한다면 쓸데없이 큰 배열을 생성 또는 요구사항보다 작은 배열을 만들어 에러를 유발할 지도 모를 경우에 사용

```
Dim 배열이름() As 데이터형식
```

  - 동적 배열의 크기를 설정하고자 한다면 ReDim 문을 사용

```
ReDim 배열이름(배열의 크기)
```

3-2 비주얼베이직 6.0 기본 문법

- **객체 변수**
  - 비주얼베이직에서는 폼이나 컨트롤같은 객체를 하나의 변수로 사용 할 수 있는데 이 변수를 객체 변수
  - 객체를 변수로 사용하려면 우선 객체의 주소를 저장할 수 있는 객체 변수를 선언

```
'- 객체 변수 ObjVar를 폼 Form1형으로 선언
  Dim ObjVar As Form1
'- 객체 변수 ObjVar를 텍스트 박스형으로 선언
  Dim ObjVar As TextBox
```

  - 객체 변수를 선언하고 그 객체 변수를 특정한 컨트롤과 연결시켜 주어야 한다. Set 명령을 이용해서 연결

```
'- 객체 변수 ObjVar를 픽쳐 박스형으로 선언하고 사용하려는 픽쳐 박스
  컨트 롤과 연결
        Dim ObjVar As PictureBox
        Set ObjVar = picColor
```

3-2 비주얼베이직 6.0 기본 문법

- **주석 및 들여쓰기**
  - 비주얼베이직에서의 주석문은 어퍼스트로피(‘)를 사용하는데 Rem에 의한 주석문도 가능. 설정된 주석문은 코드 Windows에서 녹색으로 나타나고 로직과는 무관한 문장으로 변화.

```
Dim A as Integer ‘ 변수 A는 정수형이다.
Dim B as String  ‘ Rem 변수 B는 문자열이다.
```

  - 코드를 한줄에 작성을 다 못하고 다음 줄로 넘어가는 경우에는 언더스코어(_)를 다음줄로 넘어가기 전에 한칸을 띄운 상태에서 사용
  - 한줄에 여러 명령을 기술하고자 할 때는 명령 사이에 콜론(:)을 사용하여 나열

```
Private Sub Form_Paing()
        x = 123
        Print x : Print Str(x) : Print TypeName(x) : Print _
        TypeName(Str(x))
End Sub
결과값 : 123(정수) 123(문자열) Integer String
```