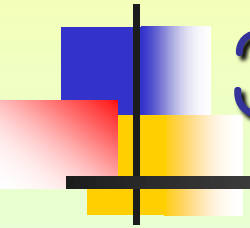


3장 비주얼베이직 6.0 문법

3-1 객체지향 프로그래밍(OOP)의 개념

정의	“Windows 환경에서 절차적인 과정을 따르지 않고 시각적인 객체를 중심으로 프로그램을 제어하는 기법”을 이용한 프로그래밍
장점	➢ 새로운 기능의 추가로 인해 기존코드의 수정이 거의 없어 유지보수가 용이 ➢ 한번 작성한 객체는 다시 재사용이 가능

- **객체**
  - 속성과 메소드를 포함하는 하나의 실체로 표현. 그 실체는 어떤 사물일 수도 있고 추상적인 존재
  - 예로 명령버튼의 (Name)속성과 Caption 속성을 설정한 후 명령 버튼에 Click 이라는 이벤트가 발생했을 때의 동작되는 내용들을 작성했을때 한 객체(명령버튼)는 설정된 속성과 동작방법(메소드)를 통하여 프로그램안에서 하나의 객체
  - 클래스란 객체를 생성하는 설계서이고, OOP의 전부

3-1 객체지향 프로그래밍(OOP)의 개념

- **추상화(Abstraction)**
  - 추상화는 OOP에 있어서 가장 핵심적인 개념이며 더 정확한 의미로 축약화
  - 비주얼베이직으로 프로그램을 작성할 때 사용하려는 컨트롤이 어떻게 작동하고 무슨 작업을 할 수 있는지를 축약해서 우리들에게 표현되는 것이 추상화 개념
- **캡슐화(Encapsulation)**
  - 데이터와 그것을 조작하는 코드를 같이 묶는 구조를 말하는 것
  - 객체를 추상화하는데 있어서 그 객체의 정보 즉 데이터를 캡슐화하는 작업은 매우 중요

3-1 객체지향 프로그래밍(OOP)의 개념

- **상속성(Inheritance)**
  - OOP는 객체를 재사용 할 수 있다는 장점을 갖게 하여 기존의 프로그래밍 기법과 차별되도록 하는 것이 상속성의 개념
  - 비주얼베이직의 텍스트 박스를 예로 텍스트 박스를 사용하기 위해 컨트롤 박스에서 그 컨트롤을 마우스로 선택한 후 폼위에 배치
  - 이는 텍스트 박스의 성질을 그대로 상속받는 객체를 그 특성에 맞게 다시 설계
- **다형성(Polymorphism)**
  - OOP에서는 하나의 객체가 같은 동작방법 즉, 같은 메소드에 대해서 서로 다른 반응을 하는 것
  - OOP용어로 오버로딩(Overloadeing)

3-2 비주얼베이직 6.0 기본 문법

- **변수의 종류 및 선언**
  - 변수란? 프로그램에서 사용하는 자료를 임시로 저장해 두는 기억장소
  - 이러한 기억장소는 사용자의 편의를 돕기 위하여 이름을 붙여 구분

Dim
ReDim
Static
Public
Private

변수명 AS 데이터형, 변수명 AS 데이터형

  - Dim(ReDim)문 : 변수를 선언하기 위한 가장 기본적인 방법
    - 데이터형을 지정

Dim Var As String : 변수 Varr가 문자열 변수임을 선언

    - 고정배열을 만듬

Dim X (1 to 100, 1 to 200) : 고정 2차원 배열 x를 선언

3-2 비주얼베이직 6.0 기본 문법

- 동적배열을 만듬

Dim Temp() : 동적배열을 선언. 나중에 배열이 프로시저안에서 재설정(Redim)되어 사용될 때 크기 재조정이 가능

```
Sub Form_Clik()
   ReDim Temp(50)
End Sub
```

- 고정 길이 문자열을 만듬

Dim MyStr As String * 20 : MyStr 변수는 문자를 20개 즉, 20byte의 크기의 갖는다.

- Dim을 처음으로 사용할 때, 숫자 배열의 모든 요소는 0으로 설정되고, 문자 배열의 모든 요소는 NULL로 설정

3-2 비주얼베이직 6.0 기본 문법

- Static문 : 지역변수의 특수한 유형으로 다른 프로시저에 가더라도 그 값이 영속적이므로 어떤 것을 기억하거나 축적을 시킬 때 Static으로 선언

  Static Counter As Integer : 정적변수 Counter를 정수형으로 선언

  - Static 변수는 프로시저안에서만 선언

- Public문

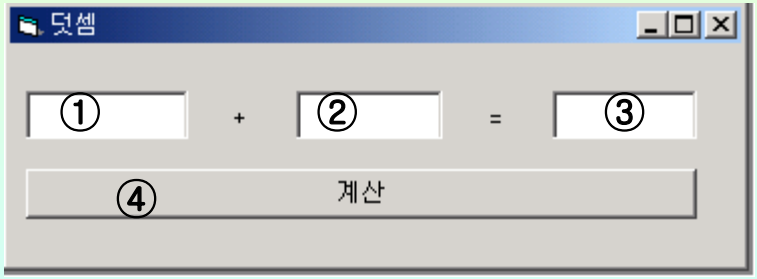
  - 프로그램 모든 곳에서 사용이 가능. 즉 어느 폼이나 프로시저 등에서 그 변수의 값이 영속적이어서 읽고 쓰기가 가능
  - 모듈이나 폼의 일반 선언부, 프로시저에서 선언이 된다. 이때 그 사용범위는 전체 폼이나 모듈에 있어서 유효

```
모듈에 선언된 경우
        Public Pub As Integer (모듈에서 선언)
        Pub = 1 (다른 폼에서 사용)
        X = Pub
폼의 일반선언부, 프로시저에서 선언된 경우
        Public Pub As Integer (Form1에서 선언)
        Form1.Pub = 1 (다른 폼에서 사용)
        X = Form1.Pu
```

  - 모듈이란 하나의 프로젝트에서 전역적으로 쓰이는 변수나 프로시저 또는 상수등을 모아놓은 프로그램

3-2 비주얼베이직 6.0 기본 문법

- Private문
  - Public의 반대 개념이라고 정의
  - 모든 프로시저 앞에는 Private가 자동으로 나타난다. 이는 프로시저가 Public인데 비주얼베이직이 Private로 만들려 하는 이유
  - Public으로 한 번 사용된 변수나 프로시저 이름은 그 프로그램에서 다른 용도로 다시 사용하지 못하고 Private로 선언된 프로시저안의 변수들은 기본적으로 Private의 성질을 갖는다.
- 변수선언을 이용한 프로그램 작성 순서 -1



< 컨트롤의 구성 >

종류	속성	값
Form(폼)	Name	frmCalc
	Caption	덧셈
① Text	Name	txtNum1
	Text	
② Text	Name	txtNum2
	Text	
③ Text	Name	txtNumTotal
	Text	
④ CommandButton	Name	cmdCalc
	Caption	계산

3-2 비주얼베이직 6.0 기본 문법

- 변수의 작성
  - '일반'항목에 변수를 선언한 예로 Dim 명령 형식으로 사용

```
Dim 변수명
```

  - 두 개 이상의 변수를 동시에 만들고 싶으면 컴마(,)로 구분

```
Dim 변수명1, 변수명2, 변수명3...
```

- 변수 이름 작성
  - 변수이름을 작성하는 규칙은 컨트롤 이름 작성 방법과 동일하게 적용

- 255자 이내 영문자, 숫자, 밑줄 문자(_)로 구성
- 첫 번째 문자는 영문자로 구성
- Sub, For, End 등 예약어
  (비주얼베이직이 재부적으로 사용하는 단어)는 사용 불가
- 동일 폼 내에 중복된 이름 사용 불가

3-2 비주얼베이직 6.0 기본 문법

- 변수에 값 할당하기
  - '계산'버튼(cmdCalc)을 누르면(cmdCalc_Click) 다음과 같은 상황이 발생

```
- 변수 Num1에 txtNum1의 Text속성에 입력되어 있는 값을 할당
- 변수 Num2에 TxtNum2의 Text 속성에 입력되어 있는 값을 할당
- 변수 NumTotal에 Num1과 num2를 합한 값을 할당
- txtNumTotal의 Text 속성에 변수 NumTotal의 값을 입력
```

  - 변수에 값을 입력하는 방법

```
변수명 = 값
```

  - 변수에 값을 할당할 때 사용하는 '='은 같다는 의미가 아니라 오른쪽 값을 왼쪽의 변수에 집어넣으라는 의미

- **데이터 형**
- **데이터 형의 종류**
  - Boolean
    - True(-1)와 False(0) 두가지만을 표현. Boolean의 기본값은 False.

```
Dim blnRunning As Boolean '테이프가 실행중인지 점검한다.
If Recorder.Direction = 1 Then blnRunning = True
```

3-2 비주얼베이직 6.0 기본 문법

- Integer
  - -32768~32767사이의 정수만을 표현
  - X% = 1/50 X%의 결과값은 0이 된다.
    X = 1/50 X의 결과값은 0.5가 된다.
    X% = 15/4 X%의 결과값은 4가 된다(반올림).
- Long
  - -2147483648~2147483647사이의 정수만을 표현
  - 숫자 범위를 넘게 지정하면 런타임시 오류가 발생
- Decimal
  - 소수점이 있을 때와 없을 때의 범위가 바뀐다.
  - 소수점이 없을 때엔 Long 정수의 범위와 정밀도를 갖고, 소수점이 있을 때엔 배정도(Double) 부동 소수점 유형의 범위와 정밀도를 갖는다.
  - 그러나 Decimal 유형으로 변수를 선언할 수는 없다.

Dim Deci As Decimal -> 잘못된 선언
Deci = Cdec(1) ->변수 Deci를 Desimal형으로 선언하며 1로 초기화

3-2 비주얼베이직 6.0 기본 문법

- Enum 형
  - 서로 관련있는 상수들을 열거하고자 할 때 사용.
  - 모듈에 정의 해주어야 하며 '이뉴머레이션'으로 발음

```
[Private | Public] Enum이름
구성요소이름 [ = 상수표현 ]
                :
End Enum
```

```
Public Enum colorName
                colorBlack = 0
                colorRed = 1
                colorBlue = 2
                colorGreen = 3
                colorWhite = 4
        End Enum
```

  - 모듈에 선언하게 되면 프로그램 작성을 할 때 숫자를 사용하는 것보다 훨씬 더 수월하게 프로그래밍을 할 수 있으며, 프로그램의 유지보수 작업에도 많은 도움을 주게 된다.

3-2 비주얼베이직 6.0 기본 문법

- 사용자 정의형(Type형)
  - 어떤 언어에서나 기본적인 데이터형만으로 프로그램을 작성하는 것은 불가능
  - 사용자 정의형은 모듈이나 일반 선언부에 정의하여야 하며 사용할 때는 사용자 정의형으로 선언된 변수형으로 변수를 선언해서 사용

```
[Private | Public] Type 사용자 정의 변수명
        구성요소 As 데이터형
            :
End Type
```

```
Type EmployeeRecord
   ID As Integer
   Name As String * 20
   Address As String * 30
End Type
```

```
모듈에 사용자 정의형으로 EmployeeRecord 변수를 선언
Sub CreateRecord()
        Dim MyRecord As EmployeeRecord
        MyRecord.ID = 1234
        myRecord.Name = “Kim ju pyo”
        MyRecord.Address = “Seoul Korea”
End Sub
```

  - 변수 MyRecord를 EmployeeRecord형으로 선언하고 MyRecord로 참조되는 각 구성 요소에 데이터를 설정

3-2 비주얼베이직 6.0 기본 문법

- 문자열 자료형(String형)
  - 숫자 값이 아닌 문자열만을 포함하는 변수는 String 유형으로 선언.

```
Private S As String
```

  - + : 여러 개의 문자열을 하나로 연결할 때 사용하는 기호
  - & : 문자열이 아닌 다른 형과 문자열을 연결시킬때 사용
  - 문자열 변수 또는 인수의 기본값 : 새 데이터가 지정되었을 때 길이가 변하는 가변 길이 문자열이나 다음 구문처럼 고정 길이 문자열로 지정

```
String * size
```

행 기 능	함 수
2개의 문자열 비교	StrComp()
소문자나 대문자로 변환	LCase(), Ucase()
반복문자의 문자열 작성	Space(10),String(5, "$")
문자열 길이 구하기	Len()
문자열 형식 지정	Format()
문자열 정렬	Lset(), Rest()
문자열 조작	Instr(), Left(), Ltrim(), Med(), Right(), Rtrim(), Trim()
문자열 변환	StrConv()

3-2 비주얼베이직 6.0 기본 문법

- 데이터 형의 변환

문자 -> 숫자	intValue = Val("1234") intValue의 결과값은 숫자 1234
숫자 -> 문자	strValue = Str(1234) ascValue의 결과값은 문자열 "1234"
문자 -> ASCII	ascValue = Asc("A") ascValue의 결과값은 숫자 65
ASCII -> 문자	chrValue = Chr(65) chrValue의 결과값은 문자 "A"
Double -> Byte	dblValue = 123.45 byteValue = Cbyte(dblValue) byteValue의 결과값은 숫자 123
소문자 -> 대문자	lowValue = "Welcome To Windows" upValue = UCase(lowValue) upValue의 결과값은 문자열 "WELCOME TO WINDOWS"
인수의 데이터 타입 -> 문자열	TypeName함수 : 변수안에 저장된 데이터의 종류를 알아냄. 특히Variant변수에 저장된 데이터의 종류를 알아내는데 유용 - whatType = TypeName(strVar) : whatType의 결과값은 문자열 "String" - whatType = TypeName(intVar) : whatType의 결과값은 문자열 "integer" - whatType = TypeName(CurVar) : whatType의 결과값은 문자열 "Currency" - whatType = TypeName(NullVar) : whatType의 결과값은 문자열 "Null" 배열의 데이터 유형 Dim N() As Integer Print TypeName(N) 결과값 : Integer() Variant 변수의 데이터 유형 X = 1 Print TypeName(×) 결과값 : Integer

3-2 비주얼베이직 6.0 기본 문법

- **변수 선언의 응용과 상수 선언**
  - 변수선언의 응용
    - 변수의 선언을 Dim이라는 명령을 사용하여 처리
    - 비주얼베이직에서는 변수를 선언하지 않고도 사용할 수 있는데 그 변수의 데이터 형은 기본적으로 Variant가 된다.
    - 변수를 사용하지 않고 사용하는 것은 프로그램의 논리적 버그를 초래하므로 프로그램 앞 부분에 지시자를 사용하여 선언되지 않은 변수는 사용하지 못하게 하는 것이 바람직

```
Option Explicit
```

    - 비주얼베이직 5.0 이후부터는 변수, 함수, 컨트롤 이름, 그리고 컨트롤 제작할 때 이벤트까지 한글로 가능

형식	Dim 변수명 as 데이터형식

< 한글 변수로 선언된 소스 >

```
Option Explicit
   Dim 홈페이지주소 as String
   Dim 서버포트번호 as integer
   ……
```

3-2 비주얼베이직 6.0 기본 문법

- **상수 선언**

상수를 선언할 경우는 Const 지시자를 사용
상수는 값에 대한 매크로라고 할 수 있다. 즉 특정값(수나 문자, 문자열)을 다른 이름으로 치환하여 사용

  - 상수의 필요성
    - 소스의 가독성.
    - 한 번의 정의로 여러 번 코딩하는 번거러움을 피할 수 있다.
    - 비주얼베이직에서 상수를 정의하는 방법

```
[public|private] Const 상수명 [As 데이터형] = 값
```

[public|private]은 제한자로써 비주얼베이직에만 있고 선택적인 사항으로써 선언해도 좋고 선언안해도 좋다. 다만 안 쓸 경우는 기본적으로 public이된다. 그리고 상수는 [As 데이터형]를 사용하여 데이터 형을 가질 수 있다. 이것 역시 선택사항

  - 사용자 상수 작성
    - Const문을 사용하면 수학 기호나 날짜/시간을 나타낼수 있을 뿐 아니라 문자열 상수를 정의하는데 사용

```
Const conPi = 3.14159265358979
Public Const conMaxPlanets As Integer = 9
Const conReleaseDate = #1/1/95#
Public Const conVersion = “07.10.A”
Const conCodeName = “Enigma”
```

3-2 비주얼베이직 6.0 기본 문법

- 등호의 오른쪽 표현은 숫자 또는 문자열로 나타나는 식일 수도 있고, 이전에 정의된 상수를 사용하여 정의

```
Const conPi2 = conPi * 2
```

- 쉼표를 사용하여 상수를 분리하면 같은 줄에 여러 개의 상수를 정의

```
Public Vonst conPi = 3.14, conMaxPlanets = 9, conWorldPop = 6E+09
```

- Const 문은 변수 선언의 경우와 같은 참조 범위를 가지며 적용되는 규칙도 같다.
- 프로시저 내에서만 존재하는 상수는 프로시저 내에서 선언하다. 모듈내의 모든 프로시저가 사용할 수 있으나 모듈 외부에서는 사용하지 못하는 상수는 모듈의 선언부에 선언
- 응용 프로그램 전체에서 사용 가능한 전역 상수는 표준 모듈 선언부의 Const 앞에 Public 키워드를 붙여 선언

비주얼베이직 함수

내장 함수

- End : 프로그램을 종료할 경우 사용

```
End
```

- Str : 숫자형 변수를 문자(String) 값으로 반환한다. 숫자들이 문자열로 변환되면 number의 부호를 위해 문자열 앞쪽에 한 개의 공간이 예약

```
Str(number)
```

- Format 함수는 숫자 값을 날짜, 시간, 화폐 단위나 다른 사용자 정의 형식으로 형식화할 때 이용하며, Str 함수와는 달리 number의 부호를 위한 앞쪽 공간을 포함하지 않는다.

3-2 비주얼베이직 6.0 기본 문법

- Val : 문자열 변수(String)를 숫자 값(Double)으로 반환한다. 숫자값으로 인식할 수 없는 문자열을 만나는 순간 Val 함수는 해당되는 첫 문자에서부터 읽기를 중단한다. 그러나 &O(8진수), &H(16진수)와 같은 문자는 인식할 수 있으며, 공백, 탭, 라인 피드와 같은 문자는 인수에서 제외한 후 처리한다.
  Val 함수는 마침표(.)만을 유효한 십진수 구분자로 인식

```
Val(string)
```

- Beep : 윈도우의 기본음을 발생시키거나 스피커에서 "~삑"소리를 나타낸다.

```
Beep
```

- 문자열 함수
  - Left : Left는 문자열 변수의 왼쪽으로부터 지정한 양만큼의 문자열을 가져올 수 있게 해 준다.

```
X = "Welcome to Windows"
Y = Left(X,7)
Print Y
결과값 : Welcome
```

  - Ltrim : 문자열 변수의 왼쪽에 있는 공란을 제거

```
X = "Korea"
Print Ltrim(X)
결과값 : Korea (왼쪽의 빈공간이 사라짐)
```

3-2 비주얼베이직 6.0 기본 문법

- Mid : 문자열 변수 왼쪽에서부터 시작해서 지정한 만큼을 가져온다. Mid에서는 두 개의 숫자를 지정해야 하는데 텍스트를 가져오기 시작할 위치와 가져올 문자의 수를 지정

```
X = “abcdefghijklmnopqrstuvwxyz”
Y = Mid(X, 5, 3)
      Print Y
결과값 : efg
```

```
X = “This is Mid function”
      Mid(X, 1, 4) = “That”
      Print X
결과값 : That is Mid function
```

- Right : 문자열 변수의 오른쪽에서 지정된 길이의 문자열을 가져온다.

```
Print Right(“ABCDE”, 2)
결과값 : DE
```

- Rtrim : 문자열 변수의 오른쪽에 있는 공란들을 제거

```
X = “ABC”
Y = Rtrim(X)
Print Len(X), Len(Y)
결과값 : 4 3
```

3-2 비주얼베이직 6.0 기본 문법

- Format() 함수
  - Format() 함수는 필요에 따라 텍스트 문자(숫자)로서 숫자들을 서식화하여 화면이나 프린터에 표시

```
X = 1000000
Print X
Print Format(X, “#,###,###”)
결과값 : 1000000    1,000,000
```

```
X = 123
Print Format(X, “# /d/a/y”)
결과값 : 123 day
(매타문자의 기능을 없애기 위해 “/” 사용)
```

- **날짜, 시간 관련 함수**
  - 날짜, 시간 함수

```
Print Date                  결과값 : 98-08-11
Print Time                  결과값 : 1:57:18 오전
Print Now                   결과값 : 2004-08-11 1:57:18 오전
Print Second(Time)          결과값 : 18
Print Minute(Time)          결과값 : 57
Print Hour(Time)            결과값 : 1
Print Day(Date)             결과값 : 11
Print Month(Date)           결과값 : 8
Print Year(Date)            결과값 : 2004
```

3-2 비주얼베이직 6.0 기본 문법

- Weekday() 함수
  - 특정 날짜의 요일을 1~7까지의 상수로 리턴하며 Format() 함수의 날짜 서식에서 “dddd” 옵션과 같은 기능을 함.

상 수	값	설 명
vbSunday	1	일요일
vbMonday	2	월요일
vbTuesday	3	화요일
vbWednesday	4	수요일
vbThursday	5	목요일
vbFriday	6	금요일
vbSaturday	7	토요일

```
Print WeekDay(Now)
        결과값 : 3 (화요일)
```

- DateSerial(year, month, day) 함수
  - DateSerial() 함수의 각각의 인자들은 변수나 계산식 또는 숫자가 올 수 있으며 리턴되는 값은 Date 함수형으로 변환.

```
Print DateSerial(2004, 8, 11)
        결과값 : 2004-08-11
```

```
Dim year, month, day As Integer
        Year= 2004
        Month=8
        Day=10
        Print DateSerial(year, month, day + 1)
    결과값 : 2004-08-11
```

3-2 비주얼베이직 6.0 기본 문법

- DateValue() 함수
  - DateValue() 함수는 일정 규칙으로 정해진 문자열을 인자로 하여 Date 함수형을 표시

```
Print DateValue("jan 23 2004")
Print DateValue("jan 23, 2004")
Print DateValue("23-Jan-2004")
Print DateValue("23 January 2003")
Print DateValue("1-23-2004")
결과값 : 2004-01-23
```

- TimeSerial(hour, minute, second) 함수
  - Dateserial() 함수와 마찬가지로 각각의 인자들은 변수나 계산식 또는 숫자가 올 수 있으며 리턴되는 값은 Time 함수형으로 변환

```
Print TimeSerial(12, 8, 11)
        결과값 : 12:08:11 오후
```

```
Dim hour, minute, second As Integer
        hour=0
        minute=8
        second=10
        Print TimeSerial(hour, minute, second + 1)
    결과값 : 12:08:11    오전
```

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