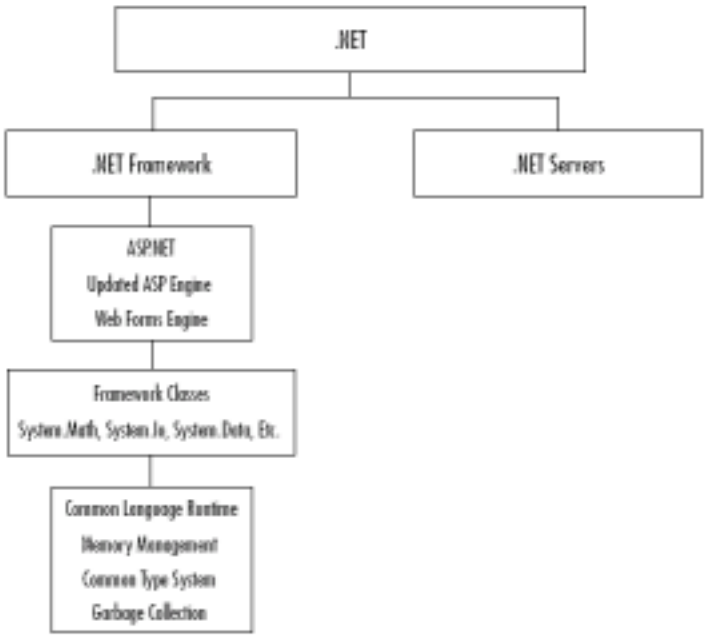
1. Visual Basic .NET 프로그래밍의 기초

1.1. Visual Basic .NET의 구조

Visual Basic .NET = Visual Design + Basic + .NET Framework

<그림1> .NET의 구조



1.2. VB.NET 프로그램의 구성

① 폼(form)
② 클래스 모듈(class module)
③ 모듈(module)

1.3. 기본 개념들

① 폼(Form)
응용프로그램의 인터페이스를 구성하는 부분으로 여러분들이 설계해야 할 윈도우로 생각하면 된다. 사용자 인터페이스와 이에 대한 **이벤트** 처리가 주된 목적이다.

② 이벤트(event)
폼 또는 콘트롤이 인식할 수 있는 어떤 행동(예: Click, Load, Change . . .)

③ ActiveX 콘트롤(Control)
폼 위에 배치하여 폼을 디자인 할 수 있는 그래픽 오브젝트
특정한 일을 수행하도록 미리 만들어져 있는 것으로 프로그램을 개발하는 도중에 손쉽게 사용할 수 있도록 하는 객체(object)를 말한다.
예) 버튼, 라벨, 콤보박스, 텍스트박스, . .

④ 객체(object)

속성(property)과 메소드(method)가 있다.
예) 폼(form), 각종 콘트롤(control)

⑤ 메소드(method)
객체에 작용하는 프로시저나 함수

⑥ 속성(property)
각각의 오브젝트가 가지고 있는 윈도우와 관련된 속성

⑦ 모듈(Module)
모든 폼에서 공통적으로 사용할 수 있는 코드를 작성하여 두는 부분이다.

⑧ 프로시저(procedure)
논리적인 단위로 묶여 호출되는 비주얼 베이직의 명령문의 모임으로 서브프로시저(Sub procedure)와 함수프로시저(Function procedure)로 나뉜다.

※ 이벤트 프로시저(event procedure) : 이벤트가 발생할 때 처리되는 프로시저

1.4. 프로그래밍 방법

비베 프로그래밍 방법은 2가지 작업으로 분리될 수 있다. 폼을 디자인하는 과정과 코드를 작성하는 과정이다. 폼을 디자인 하는 과정을 사용자 인터페이스를 설계하는 과정이며, 코드 작성은 이벤트에 대한 처리 루틴을 작성하는 것과 프로그램에서 사용할 프로시저와 함수를 제작하는 것이다.

(1) Application 작성 과정

1. 윈도우를 원하는 대로 디자인한다.
2. 윈도우에 사용한 오브젝트들의 속성값들을 설정한다.
3. 여러 가지 컨트롤에 관련된 이벤트에 대해 코드를 작성한다.

(2) Visual Basic 프로그램의 구조

Visual Basic 프로그램은 표준 빌드 블록을 가집니다.
Visual Basic 코드는 프로젝트 모듈에 저장됩니다. 프로젝트는 응용 프로그램으로 컴파일되는 여러 개의 파일로 구성됩니다. 프로젝트를 시작하고 코드 편집기를 열면 일부 코드가 이미 적절한 위치에 올바른 순서로 표시되어 있습니다. 다음 순서를 따라 코드를 작성해야 합니다.

Option 문
Imports 문
Main 프로시저
Class, Module 및 Namespace 문(해당되는 경우)

또한 프로그램에 조건부 컴파일 문이 포함될 수도 있습니다. 이러한 문은 모듈의 어떤 위치에도 놓일 수 있습니다. 일반적으로 프로그래머는 이 문을 모듈의 끝에 포함

시킵니다.
위와 다른 순서로 문을 입력하면 오류가 발생할 수 있습니다.

① Option 문
Option 문을 사용하면 그 다음 코드에 대해 기본 규칙을 구성하여 구문 오류와 논리 오류가 발생하지 않도록 합니다. Option Explicit 문은 모든 변수의 선언과 맞춤법이 올바르도록 하므로 나중에 디버깅할 때 소요되는 시간을 상당히 줄여 줍니다. Option Strict 문은 서로 다른 형식의 변수 사이에서 작업할 때 발생할 수 있는 논리 오류와 데이터 손실을 방지하는 데 유용합니다. Option Compare 문에서는 각 문자열의 상호 비교 방식을 Binary 또는 Text 중 하나로 지정합니다.

② Imports 문
Imports 문을 사용하면 가져오는 네임스페이스 내에서 정의되는 클래스 및 기타 형식을 한정하지 않고도 이름을 지정할 수 있습니다.

③ Main 프로시저
Main 프로시저는 응용 프로그램의 "시작 지점"이 되는 프로시저로, 코드를 실행할 때 첫 번째로 액세스되는 프로시저입니다. 즉, Main에는 먼저 액세스해야 할 코드가 포함됩니다. Main을 사용하면 프로그램을 시작할 때 가장 먼저 로드되는 폼을 결정하거나, 응용 프로그램의 복사본이 시스템에서 이미 실행 중인지를 확인하거나, 응용 프로그램의 변수 집합을 구성하거나, 응용 프로그램에 필요한 데이터베이스를 열 수 있습니다. Main은 네 가지 형태로 사용됩니다.

- Sub Main()
- Sub Main(ByVal CmdArgs() As String)
- Function Main() As Integer
- Function Main(ByVal CmdArgs() As String) As Integer

이러한 프로시저 중에서 가장 일반적인 것은 Sub Main()입니다. 자세한 내용은 Hello World!의 Visual Basic 버전을 참조하십시오.

④ 클래스, 모듈 및 네임스페이스 문
클래스와 모듈은 소스 파일에서 대부분의 코드를 구성합니다. 여기에는 사용자가 작성하는 코드의 대부분(주로 Sub, Function, Method 및 Event 문)과 응용 프로그램이 실행되는 데 필요한 변수 선언 및 기타 코드가 함께 포함됩니다.

⑤ 조건부 컴파일 문
조건부 컴파일 문은 모듈 내의 모든 위치에 올 수 있습니다. 이 문은 런타임 시 특정 조건에 맞으면 실행되도록 설정됩니다. 조건부 코드는 디버깅 모드에서만 실행되므로 응용 프로그램을 디버깅할 때에도 이 문을 사용할 수 있습니다.

(3) Module의 구조

module에는 폼모듈(form module), 코드모듈(code module)이 있다. 크게 선언부, 이벤트 프로시저, 일반프로시저로 구성되어있다.

```
Imports System.Math '⓪
Public Class 폼이름
 Inherits System.Windows.Forms.Form

 '주석문                    '①
 변수 선언                      '②
 DLL함수 선언                   '③

 Sub 이벤트프로시저         '④
    지역 변수 선언              '⑤
    초기화
    Call 프로시저()             '⑥
 End Sub

 Sub 프로시저()
    지역 변수 선언
    초기화
    문
 End Sub
 Function 함수이름()            '⑦
    지역 변수 선언
    초기화
    문
    함수= 리턴값            '⑧
 End Function

End Class '
```

① 주석문(comments)

'(apostrophe)로 시작한다. 코드의 가독성을 높이기 위해 프로그래머가 코드에 대한 설명을 다는데에 주로 사용한다.

② 변수선언 (declarations)

· 선언부에 Option Explicit On이라고 지정하면, 코드에 있는 모든 변수명은 반드시 선언되어야 사용할 수 있게 된다(선언되지 않은 변수는 컴파일 시 에러를 냄).

· *지역 변수*는 프로시저 내에서 선언되는 변수이지만 *모듈 변수*는 모듈 수준, 즉 모듈 내부에 있지만 모듈의 프로시저 외부에서 선언되는 변수입니다.

· 예) Dim int As Integer ’지역변수 선언

③ DLL함수 선언(고급, 구조화)

외부에서 만들어진 라이브러리(DLL) 함수를 이용할 때 미리 선언해야 한다.

Visual Basic 6.0에서는 Visual Basic에서 제공하지 않는 기능에 액세스하기 위해

Declare 문을 사용하여 Windows API를 호출합니다. 그러나 Visual Basic .NET에서는 대부분의 Windows API 기능이 .NET Framework에 포함되어 있으므로 Windows API에 액세스해야 할 경우가 많지 않습니다.

④ 이벤트 프로시저

콘트롤의 사용 중에 발생한 이벤트를 처리하는 서브 프로시저이다.
폼 모듈에만 있는 프로시저로, 코드 모듈에서는 사용하면 안된다.

예)

```
Sub MyForm_DragDrop (Source As Control, X As Single, Y As Single)
 .
 .
 .
End Sub
```

⑤ 지역변수 선언

필요한 지역변수를 선언하고 코드를 작성한다. 이 변수는 프로시저가 끝나면 자동으로 소멸된다.

예)

```
Sub SubDemo (RLen, RWid)
    Dim Area  As Single                    '지역변수 선언
    Area = RLen * RWid
    MsgBox "Total area is " & Area
End Sub
```

⑥ Sub 프로시저의 호출

프로그램의 제어를 서브 프로시저나 DLL 프로시저에게 넘긴다.(하청을 준다)

예) MainCalc라는 **이름**의 Sub Procedure를 호출할 경우(NumA, NumB, NumC라는 세 개의 변수를 넘기는 경우이다.)

a) Call 명령어를 사용

```
Call MainCalc(NumA, NumB, NumC)
```

b) Call 명령어 없이 사용

```
MainCalc NumA, NumB, NumC
```

⑦ 함수 프로시저(User-Defined Function)

반환 값(return value)이 있는 프로시저

즉, 매개변수(parameter) ⇝ 함 수 ⇝ 반환 값(return value)

예)

'함수 정의

```
Function CalcDiscrimnant( a as single, b as single, c as  single)
    '
    CalcDiscrimnant = b*b - 4*a*c          'return value
    '
```

```
    End Function
'함수 사용
    Dim d as single
    d = CalcDiscrimnant(1, 2, 3)
```

*** 모르는 것이 있을 때 ☞ Help(F1)**

1.5. 오브젝트명 부여 규칙

코드의 가독성을 높이기 위해 보통 다음과 같은 규칙으로 오브젝트명을 부여하며, 이렇게 하여 오브젝트명, 변수명만 보아도 그 뜻을 알 수 있게 프로그래밍 하는 것이 상당히 중요하다.

오브젝트	축약어	명명예
form	frm	frmMain
check box	chk	chkReadOnly
combo box	cbo	cboEnglish
command button	cmd	cmdOK
data	dat	datBiblio
frame	fra	fraLanguage
grid	grd	grdPrices
image	img	imgIcon
label	lbl	lblName
list box	lst	lstCodes
menu	mnu	mnuFileOpen
option button	opt	optKorean
picture box	pic	picDiskSpace
text box	txt	txtOutput
timer	tmr	tmrAlarm
dialogue box	dlg	dlgFileOpen

· 같은 방식으로 변수명에도 접두어를 두어 변수형을 알아볼 수 있게 하는 것이 좋다. 다음은 구별하는 것을 권장하는 변수형과 그 축약어의 예이다.

변수형	축약어	사용예
global(전역변수)	g	gKValue
string(문자)	str	strOutput
integer(정수)	int	intZoneCounter

<과제> Label, Command button, Textbox를 포함하여 6개 이상의 control을 사용하여 각자 자유롭게 프로그램을 작성할 것.

조건:
· 반드시 하나 이상의 모듈을 작성하고 하나 이상의 함수 및 서브루틴을 모듈 안에 작성할 것.
· 오브젝트 명명시 위의 오브젝트명 부여 규칙을 따를 것
· Command button을 눌렀을 때 Textbox 또는 Label의 내용이 변경되는 이벤트 핸들링을 포함할 것

2. 변수(variable)와 데이터 형식(data type)

♣ 변수란?

☞ 프로그램의 실행 도중 필요한 값을 저장하는 곳을 말한다.

☞ MSDN Library의 '변수, 기본개념'을 보세요.

♣ 데이터 형식(data type)이란?

☞ 컴퓨터 메모리에 저장할 변수의 형태를 지칭하는 것이다.

예) Integer, Long, Single, Double, String, Object, Structure ,.....

☞ MSDN Library의 '데이터 형식'을 보세요.

2.1. 변수

(1) 변수 이름 정하는 규칙

▶문자로 시작해야 한다

▶키워드나 "."은 사용할 수 없다

▶중복 선언은 허용되지 않는다

▶255자 이내여야 한다

(2) 변수 선언 명령어

▶Dim: 지역변수, 모듈변수 선언.

▶Private: 모듈변수로만 선언 가능. 다른 모듈에서는 사용할 수 없도록 할 때.

▶Public: 모듈변수로만 선언 가능. 어떤 모듈이라도 자유롭게 사용 가능.

▶Static: 정적변수의 선언시 사용. 정적변수는 변수가 선언된 곳의 실행이 끝나더라도 변수의 값을 잃어버리지 않고 계속 갖고 있는 변수이다. 주의할 것은, 이 값은 해당 프로시저 내에서만 사용할 수 있다는 것이다.

구문;

Dim(또는 Global, Public, Private. .) [변수이름] As [데이타형]

예) Dim int As Integer

Dim [변수이름][형식선언문자]

예) int%

(3) 통용범위

변수의 선언 위치와 명령어에 따라 세 종류의 통용범위가 결정된다.

통용범위	선언위치	명령어
지역	프로시저 안	Dim, Static, ReDim
모듈내, 폼내	폼 또는 모듈의 선언부	Dim, Private, Public
전역	모듈의 선언부	Global

2.2. 데이터 형식(data type)

(1) 일반

♣ 메모리의 효율적인 관리를 위해서는 적절한 변수를 사용해야 한다. 적절한 변수를 사용한다는 것은 메모리의 사용이 가장 적으면서도 원하는 값을 표현할 수 있는 데이터형을 선택한다는 뜻이다. VB.NET가 지원하는 데이터 형식 중 대표적인 것들은 다음 표와 같다.

☞ 도움말에서 '데이터형식'을 참고해서 VB에서 지원되는 모든 데이터 형식을 체크하세요.

♣ 만약 변수를 선언할 때 데이터 형식을 지정하지 않으면, VB.NET은 Object로 인식한다!

다음 표에서는 Visual Basic .NET 데이터 형식과 각 형식이 지원하는 공용 언어 런타임 형식, 일반 저장소 할당 및 값 범위를 보여 줍니다.

Visual Basic 형식	공용 언어 런타임 형식 구조체	일반 저장소 할당	값 범위
Boolean	System.Boolean	2바이트	True 또는 False
Byte	System.Byte	1바이트	0 ~ 255(부호 없음)
Char	System.Char	2바이트	0 ~ 65535(부호 없음)
Date	System.DateTime	8바이트	0:00:00(0001년 1월 1일) ~ 11:59:59 PM(9999년 12월 31일)
Decimal	System.Decimal	16바이트	소수 자릿수가 없는 경우: +/-79,228,162,514,264,337,593,543,950,335 사이 소수 자릿수가 28개인 경우: +/-7.9228162514264337593543950335 사이 0이 아닌 숫자 중에서 최소 숫자는 +/-0.0000000000000000000000000001(+/-E-28)입니다.
Double (배정밀도 부동 소수점)	System.Double	8바이트	-1.79769313486231570E+308 ~ -4.94065645841246544E-324(음수), 4.94065645841246544E-324 ~ 1.79769313486231570E+308(양수)
Integer	System.Int32	4바이트	-2,147,483,648 ~ 2,147,483,647
Long(long integer)	System.Int64	8바이트	-9,223,372,036,854,775,808 ~ 9,223,372,036,854,775,807
Object	System.Object(클래스)	4바이트	Object 형식의 변수에는 모든 형식을 저장할 수 있습니다.
Short	System.Int16	2바이트	-32,768 ~ 32,767
Single (단정밀도 부동 소수점)	System.Single	4바이트	-3.4028235E+38 ~ -1.401298E-45(음수), 1.401298E-45 ~ 3.4028235E+38(양수)
String (가변 길이)	System.String(클래스)	구현하는 플랫폼에 따라 다름	0 ~ 약 20억 개의 유니코드 문자
사용자 정의 데이터 형식 (구조체)	(System.ValueType에서 상속)	구현하는 플랫폼에 따라 다름	구조체의 각 멤버는 데이터 형식에 의해 결정되고 다른 멤버의 범위와는 무관한 범위를 갖습니다.

참고

텍스트를 포함하는 문자열에서는 StrConv 함수를 사용하여 텍스트 형식을 다른 형식으로 변환하십시오.

메모리 사용

기본 데이터 형식을 선언할 때 메모리 사용량이 일반 저장소 할당량과 같다고 가정하는 것은 위험합니다. 공용 언어 런타임에서 응용 프로그램이 실행되고 있는 플랫폼의 특성에 따라 저장소를 할당합니다. 간혹 공용 언어 런타임은 선언된 요소를 가능한 한 조밀하게 압축합니다. 반면에 메

모리 주소를 원래의 하드웨어 경계에 맞추는 경우도 있습니다. 또한 64비트 플랫폼과 32비트 플랫폼에서 저장소를 할당하는 방식은 서로 다릅니다.
구조체나 배열과 같은 복합 데이터 형식의 각 멤버에는 같은 고려 사항이 적용됩니다. 또한 일부 복합 형식에는 메모리가 추가로 필요합니다. 예를 들어, 배열은 배열 자체뿐 아니라 각 차원에 대해서도 메모리를 사용합니다. 32비트 플랫폼의 경우 현재 각 차원에 대해 기본적인 12바이트 외에 8바이트가 추가로 필요합니다. 64비트 플랫폼에서는 요구 사항이 두 배로 증가합니다. 단순히 구성 요소의 일반 저장소 할당량을 모두 더하는 것만으로 필요한 메모리를 계산할 수는 없습니다.
기본 또는 복합 데이터 형식을 참조하는 Object는 해당 데이터 형식에 포함된 데이터 외에 4바이트를 추가로 사용합니다.

다음 표에서는 Visual Basic 6.0과 Visual Basic .NET의 각 정수 형식을 보여 줍니다.

정수크기	Visual Basic 6.0 형식 및 형식 문자	Visual Basic .NET 형식 및형식문자	공용 언어 런타임 형식
부호 있는 8비트	(없음)	(없음)	System.SByte
부호 있는 16비트	Integer(%)	Short(없음)	System.Int16
부호 있는 32비트	Long(&)	Integer(%)	System.Int32
부호 있는 64비트	(없음)	Long(&)	System.Int64

① String 데이타형
항상 문자열만 저장한다. 문자열의 길이는 기본적으로 가변적이다.
예)
```
Dim Str As String        ' 가변길이
```

② 상수 (constant)
프로그램에서 의미가 있거나 값이 결정되어 그 값이 변하지 않는 것은 상수를 사용한다.

구문;
```
Const [상수명] = 값
```

예) ' 상수는 기본값으로 Private을 갖습니다.
```
Const MyVar = 459
```

③ 사용자 정의 데이터 형식 ★

프로그래머가 기존에 VB에서 제공하는 여러 가지 데이터 형식을 조합하여 자신이 필요한 데이터 형식을 스스로 정의하여 사용할 수 있도록 한 것이다. 사용자 정의 데이터 형식을 선언할 경우에는 Structure이라는 명령어를 사용하면 된다.
Single, String처럼 변수 선언시 하나의 데이터 형식으로 쓸 수 있다.

· 구문;
```
Structure [구조체이름]
        Dim [변수이름] As [데이타형]
        Dim [변수이름] As [데이타형]
            ⋮
        Dim [변수이름] As [데이타형]
End Structure
```

예)

```
Structure Person     'Person 이라는 Data Type을 정의
    Dim Name As String * 8        'Person 이라는 type 내의 요소들 정의
    Dim Number As Long
    Dim Salary As Currency
End Structure

Dim Young As Person   'Young이라는 변수를 위에서 사용자가 정의한 데이터 형식으로
                      '쓰겠다고 선언
Young.Salary = 500
```

(2) 형변환(casting)

① 형변환 = 변수의 데이터형을 다른 데이터형으로 변환하는 것

형변환 함수	기능
CLng	Long형으로 변환
CStr	String형으로 변환
CInt	Integer형으로 변환
CDbl	Double형으로 변환
CBool	Boolean형으로 변환

② 형변환 옵션

- Option Strict On : 형변환을 명시적으로 해야 한다.
- Option Strict Off : 형변환을 암시적으로 해도 된다.

(3) VB와 VB.NET의 달라진 점

Visual Basic .NET에서는 Deftype 문이 지원되지 않습니다. Currency 데이터 형식도 지원되지 않습니다. 대신 모든 money 변수 및 계산에서 소수점 양쪽의 자릿수를 더 많이 처리할 수 있는 새로운 Decimal 데이터 형식이 사용됩니다. 또한 Decimal은 공용 언어 런타임에서 직접 지원됩니다.

Visual Basic .NET에서 Date는 8바이트 정수 값인 공용 언어 런타임 DateTime 데이터 형식을 사용합니다. 이러한 데이터 표현 방식의 차이로 인해 Visual Basic .NET에서는 Date 및 Double 데이터 형식 간에 암시적인 변환이 수행되지 않습니다. Double과 Visual Basic 6.0의 Date 형식 간에 변환을 하려면 System 네임스페이스에 있는 DateTime 구조체의 ToOADate 및 FromOADate 메서드를 사용합니다.

Visual Basic .NET에서는 정수를 포함하는 두 개의 Object 변수를 곱하여 오버플로가 발생하면 결과가 64비트 데이터 형식 Long 데이터 형식으로 변경됩니다.

하나의 데이터 형식을 다른 데이터 형식에 지정하는 경우 LSet 및 RSet을 사용할 수 없습니다. 이렇게 하면 특히 구조적으로 형식이 불안정한 작업이 발생하여 확인할 수 없는 코드가 생성될 수 있습니다.

<과제> 사용자 정의 데이터형을 하나 만들고, 그것을 이용하는 간단한 프로그램을 만들어보세요.

<과제2> MSDN에서 '인수, 정의' 부분을 읽어보고, ByVal과 ByRef의 차이를 체크하세요.

2.3. 배열(array)

편.리.한. 변수의 형태이다.
하나의 이름으로 참조되는, **같은 형**의 여러 개의 데이터 뭉치.

배열에는 정적배열과 동적배열 2가지가 있다.

(1) 정적 배열

정적배열은 선언하면서 그 크기를 미리 지정하는 배열이다.
배열의 하한치는 0부터 시작된다.

구문 : Dim [배열이름] ([배열의 원소 개수-1]) As [데이타형]

예)
Dim a(29) As Integer ' 0부터 29까지의 배열요소를 갖는다
Dim b(9,9) As Single ' 2차원 배열, 배열요소갯수=배열의 크기=10*10=100

(2) 동적 배열

동적배열은 선언하면서 크기를 지정하지 않는다.
즉, 실행시에 크기가 변경될 수 있다.

▶모듈이나 해당 프로시저에서 Dim 또는 Private 등으로 선언한다(이 때, 배열의 크기는 정하지 않고 단지 "()"만 붙여준다)
▶ Redim 키워드를 사용하여 배열의 크기를 정한다.

예)
Dim summation() As Integer
Redim summation(10, 10) As Integer

※동적배열을 ()로 선언할 때는 컴퓨터는 데이터 형식만 인식하므로, 메모리의 영역을 할당하고 있지 않다가...Redim을 만나게 되면 그때서야 메모리를 할당한다.

동적배열은 프로그램의 실행 시, 배열의 크기를 알 수 없을 때 유용하게 사용할 수 있고, Redim을 행한 프로시저가 종료되면 메모리에서 차지하고 있던 영역이 메모리에 반환된다는 이점이 있다.

3. 기본적인 명령어들

3.1. 연산자

산술 연산자	
+	덧셈
−	뺄셈 또는 음의 부호
*	곱셈
/	나눗셈
^	거듭제곱
\	몫 구하기 예) 10\3 = 3
Mod	나머지 구하기 예) C = A Mod B · A를 B(실수를 정수로 반올림 한)로 나누고, 나머지만을 C에 반환한다
&	문자열의 덧셈
대입 연산자	
=	왼쪽 값에 오른쪽 값을 대입
기타 연산자	
Is	두 객체 변수를 비교하여 True 또는 False 를 리턴하는 연산자. 예) C = A Is B · A가 참조하는 변수와 B가 참조하는 변수가 같으면 C는 True, 다르면 False를 반환받는다
Like	두 문자열을 비교... 예)C="015B" Like "###?" · 문자열 015B가 ###? 와 같은 패턴이면 C는 True, 다른 패턴이면 False를 반환받는다

논리 연산자	
Not	부정
And	논리곱
Or	논리합
Xor	양쪽이 서로 다르면 True, 같으면 False
Eqv	양쪽이 동일하면 True, 다르면 False
Imp	T Imp F 이면 False, 그 외는 모두 True
관계연산자	
=	같다
<>	같지 않다
<	작다
>	크다
<=	작거나 같다
>=	크거나 같다
? : 한 문자 * : 0개 이상의 문자 # : 한 숫자 [ ] : 대괄호 내에 있는 한 문자 [!] : 대괄호 내에 없는 문자	

※중요한 VB 내장함수들

Asc(문자열)	문자열의 첫 번째 문자열에 대응하는 문자코드의 Integer 반환 · 문자코드의 정수값을 알고싶다면, MSDN의 "ASCII 문자 집합"을 참조.
Chr(문자코드)	문자코드에 대응하는 문자열(String)을 반환
Str(숫자)	숫자형 변수의 데이터 형식을 문자형으로 변환하여 반환 예) MyString = Str(512) '*문자열 " 512"를 반환* *· 반환될 때 맨 앞의 공백은 부호용으로 할당된다*
Val(문자열)	문자형 변수의 데이터 형식을 숫자형으로 변환하여 반환
Hex(숫자)	숫자에 대응하는 16진수의 값을 반환
Format(문자열, 형식)	출력함수 문자열이나 숫자를 특정형식으로 출력할 때(자리값을 맞출 때)
MsgBox	화면에 윈도우의 기본적인 메시지 박스를 출력
Open, Close	파일을 읽고 쓰고, 제어하고자 할 때 (4장에서 다룸)

3.2. 선택문(분기문)

(1) If Then 문

조건이 참이면(곧, 조건식의 값이 0이 아니면) 명령문을 수행하고, 조건식이 거짓이면(곧, 조건식의 값이 0이면) 그냥 If문을 빠져나온다.

▶ 구문 1

```
If   조건   Then   명령문
```

▶ 구문 2

```
If   조건   Then
        명령문
        ⋮
End If
```

▶ 구문 3

```
If   조건   Then
        명령문 블럭1
Else
        명령문 블럭2
End If
```

▶ 구문 4

```
If   조건 1   Then
        명령문 블럭1
ElseIf 조건 2 Then
        명령문 블럭2
Else
        명령문 블럭3
End If
```

▶ 구문 5

```
If  조건 1  Then
        명령문 블럭1
ElseIf  조건 2 Then
        명령문 블럭2
ElseIf  조건 3 Then
        명령문 블럭3
ElseIf  조건 4 Then
        명령문 블록4
Else
        명령문 블록5
End If
```

예)

```
If D>∅ Then            ' 이차방정식의 판별식
    x1=(-b+ Sqrt(D))/(2*a)
    x2=(-b- Sqrt(D))/(2*a)
ElseIf D=∅ Then
    x1=x2=-(b/(2*a))
Else                   '위 상황 외의 나머지 경우
    x1=(-b+ Sqrt(-D))/(2*a)
    x2=(-b- Sqrt(-D))/(2*a)
End If
```

(2) Select Case 문

▶ 구문

```
Select Case 조건
        Case 조건 1
                명령문 1
        Case 조건 2
                명령문 2
            ⋮
        Case Else
                명령문 n
End Select
```

; 조건이 조건 *1*인 경우에는 명령문 *1*을 실행하고, 이도저도 아니면 명령문 *n*을 실행한다

예)

```
Select Case grade
    Case 91 To 100
            string = "A학점"
    Case 81 To 90
            string = "B학점"
    Case 71 To 80
            string = "C학점"
    Case Else
            string = "D학점"
End Select
```

♣ InputBox

사용자로부터 문자 혹은 숫자를 입력받을 때 사용하는 함수이다.
숫자를 입력받을때는; InputBox
문자를 입력받을때는; InputBox$

3.3. 순환문(반복문)

(1) For ... Next 문

▶ 구문 1

```
For  카운터 = 초기값  To  끝값  (Step 증가량)
       명령문a
       (Exit For)
       명령문b
Next 카운터
```

증가량을 별도로 주지 않으면, 1씩 증가한다. 강제로 순환문을 빠져 나오려면 Exit For 라고 하면 된다.

▶ 구문 2

```
For  카운터 1 = 초기값 1 To  끝값 1  (Step 증가량)
      For 카운터 2 = 초기값 2 To  끝값 2
             For 카운터 n = 초기값 n  To  끝값 n

                    명령문

             Next 카운터 n
      Next 카운터 2
Next 카운터 1
```

(2) Do ... Loop 문

```
Do
        문장블럭
        (Exit Do)
        문장블럭
Loop
```

```
Do While 조건            '또는 "Until 조건"
        문장블럭
        (Exit Do)
        문장블럭
Loop
```

```
Do
        문장블럭
        (Exit Do)
        문장블럭
Loop Until 조건          '또는 "While 조건"
```

* Do~Loop 문의 여러 가지 형식(구별!!)

반복하기 전에 조건식 검사		일단 한 번 실행문을 수행하고 조건식 검사	
조건식이 참일 경우 반복함	조건식이 거짓일 경우 반복함	조건식이 참일 경우 반복함	조건식이 거짓일 경우 반복함
Do While 조건식 실행문 [Exit Do] 실행문 Loop	Do Until 조건식 실행문 [Exit Do] 실행문 Loop	Do 실행문 [Exit Do] 실행문 Loop While 조건식	Do 실행문 [Exit Do] 실행문 Loop Until 조건식

3.4. GoTo문

```
GoTo 이름

이름:
        문장블럭
```

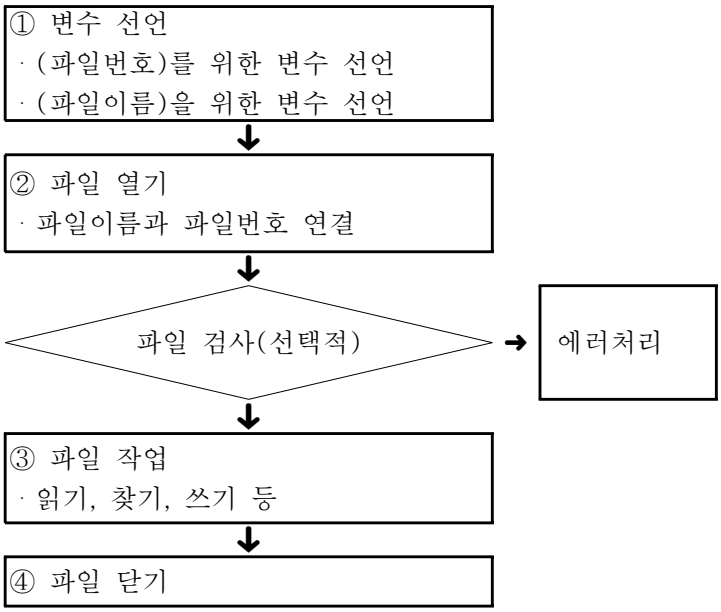
'이름: '이 있는 곳으로 무조건 건너가라.
요즘에 GoTo 없는 프로그램을 선호하므로 잘 사용하지 않는다. 그러나, 실행에러를 처리하는 데 유용하다.

<과제 1> 1℃, 2℃, 3℃, . . 10℃까지 10개의 섭씨온도를 화씨온도로 변환하여 출력하는 프로그램을 작성하세요. 순환문(For~Next, Do~Loop 등)을 사용하여 10개의 값이 한번의 이벤트에 모두 출력되도록 하세요.

4. 파일 입출력

4.1. 파일 읽기와 쓰기의 개요

(1) 파일 읽기와 쓰기의 순서



[그림 1] 파일 읽기와 쓰기의 순서

(2) 파일 액세스 형식, 함수, 문

파일 자체는 디스크에 있는 일련의 관련 데이터 바이트에 불과합니다. 응용 프로그램에서는 파일에 액세스할 때 해당 바이트가 문자, 데이터 레코드, 정수, 문자열 등의 데이터 형식 중에서 어떤 형식을 나타내는지를 가정해야 합니다. 또한 사용자가 파일의 액세스 형식을 지정하여 응용 프로그램에 데이터 형식을 알려 줄 수 있습니다.
사용하는 파일 액세스 형식은 파일에 포함된 데이터 종류에 따라 다릅니다. Visual Basic에서는 다음의 세 가지 파일 액세스 형식이 제공됩니다.

- 순차 형식. 연속적인 블록의 텍스트 파일을 읽고 쓰기 위한 것입니다.
- 임의 형식. 고정 길이 레코드로 구조화된 텍스트 또는 이진 파일을 읽고 쓰기 위한 것입니다.
- 이진 형식. 임의로 구조화된 파일을 읽고 쓰기 위한 것입니다.

*순차 액세스*는 일반 텍스트 파일에서 사용하도록 디자인되었습니다. 파일의 각 문자는 줄 바꿈 문자와 같은 텍스트 서식 시퀀스 또는 텍스트 문자를 나타내도록 가정됩니다. 데이터는 ANSI 문자로 저장됩니다.
*임의 액세스*용으로 연 파일은 같은 길이의 레코드 집합 또는 정보를 포함하는 필드 집합으로 구성된다고 가정됩니다. 사용자 정의 형식을 사용하여 각각 다른 데이터 형식을 가질 수 있는 많은 필드로 구성되는 레코드를 만들 수 있습니다. 데이터는 이진 정보로 저장됩니다.
*이진 액세스*를 사용하면 사용자의 요구에 맞게 숫자, 문자열 또는 두 가지 형식을 모두 갖는 데이터를 저장하는 데 파일을 사용할 수 있습니다. 이진 액세스는 데이터 형식이나 레코드 길이가 미리 가정되지 않는다는 점을 제외하고는 무작위 액세스와 비슷합니다. 그러나 데이터를 제대로 검색하기 위해서는 데이터가 파일에 쓰여진 방식을 정확하게 알아야 합니다. 예를 들어, 일련의 이름 및 전화 번호를 저장하는 경우 첫째 필드(이름)가 텍스트이고 둘째 필드(전화 번호)가 숫자라는 사실을 기억해야 합니다.

(3) 파일 액세스 함수 및 문

다음 함수가 순차, 임의 및 이진 파일 액세스에 사용됩니다.

Dir 함수	EOF 함수	FileCopy 함수
FileDateTime 함수	FileLen 함수	FreeFile 함수
GetAttr 함수	Loc 함수	LOF 함수
Seek 함수	SetAttr 함수	

다음 표에서는 세 가지의 직접 파일 액세스 형식에 각각 사용할 수 있는 파일 액세스 문, 함수 및 메서드를 모두 보여 줍니다.

문, 함수 및 메서드	순차	임의	이진
FileClose 함수	X	X	X
FileOpen 함수	X	X	X
Input 함수	X		X
InputString 함수	X		
LineInput 함수	X		
Print, PrintLine 함수	X		
Write, WriteLine 함수	X		

4.2. 순차(sequential) 접근

(1) 파일 열기

▶ 명령어: FileOpen

▶ 파일열기에는 3가지 모드가 있다. 어떤 모드로 파일을 열 것인가?

① OpenMode.Input : 읽기전용 모드. 기존의 파일이 있어야만 사용가능하다.

② OpenMode.Output : 쓰기전용 모드, 파일이 없으면 만든 후 데이터를 쓴다.

③ OpenMode.Append : 추가모드, 파일의 끝으로 이동하여 데이터를 추가할 준비를 한다.

▶ FileOpen 문

FileOpen (파일번호, 파일이름, OpenMode.Input)

예) FileOpen(*FileNumber*, *FileName*, OpenMode.Input)

```
'1)파일입출력을 위한 변수 선언
Dim 파일이름 As String
Dim 파일번호 As Integer
Dim 한줄내용 As String
Dim 모든내용 As String
'
'2) 변수 할당 및 파일 열기
파일번호 = FreeFile()
'
dlgOpen.Filter = "Midas Output files (*.anl)|*.anl"
dlgOpen.ShowDialog()

파일이름 = dlgOpen.FileName
If 파일이름 = "" Then '파일을 선택하지 않을 경우
    Exit Sub
End If
'
FileOpen(파일번호, 파일이름, OpenMode.Input) '파일열기
'
'3) 파일내용 읽기
모든내용 = ""
Do While Not (EOF(파일번호))
    한줄내용 = LineInput(파일번호)
    모든내용 = 모든내용 & 한줄내용 & vbCrLf '
Loop
'
txtResult.Text = 모든내용
'
'4) 파일 닫기
FileClose(파일번호)
'
```

▶ 파일을 여는 온전한 구문

예) d:\test.txt 파일을 읽기전용모드로 열었다가 닫는다.

♣ FreeFile 함수

: (Open문에서) 사용할 수 있는 파일 번호(Integer)를 반환하는 함수

※ Open한 파일은 **반드시** Close해준다 !

과거의 App.Path는 System.Reflection.Assembly.GetExecutingAssembly.Location

(2) 파일 읽기 (입력)

▶ 명령어

① Input 함수

: 쉼표, 라인끝 으로 데이터를 구분하여 읽어와서, 변수에 데이터를 입력한다 열려 있는 순차 파일에서 데이터를 읽은 다음 변수에 할당합니다.

Public Sub Input(*FileNumber* As Integer, ByRef *Value* As Object)

FileNumber 필수적 요소. 임의의 유효한 파일 번호입니다. *Value* 필수적 요소. 파일에서 읽은 값이 할당되는 변수로, 배열이나 개체 변수는 여기에 사용될 수 없습니다.

② LineInput 함수

: 파일로부터 한 **라인 단위**로 데이터를 읽어와서, 변수에 데이터를 입력한다

열려 있는 순차 파일에서 한 줄을 읽은 다음 문자열 변수에 할당합니다.

Public Function LineInput(ByVal *FileNumber* As Integer) As String

FileNumber

필수적 요소. 임의의 유효한 파일 번호입니다.

설명

LineInput 함수를 사용하여 읽은 데이터는 대개 Print 함수를 사용하여 파일에 쓰여집니다. LineInput 함수는 캐리지 리턴(Chr(13)) 또는 캐리지 리턴-줄 바꿈(Chr(13) + Chr(10)) 시퀀스가 나타날 때까지 파일에서 한 번에 한 문자씩 읽습니다. 캐리지 리턴-줄 바꿈 시퀀스는 문자열 뒤에 추가되지 않고 건너뜁니다.

예제

다음 예제에서는 LineInput 함수를 사용하여 순차 파일에서 한 줄을 읽은 다음 변수에 할당합니다. 이 예제에서 TESTFILE은 여러 줄의 샘플 데이터로 이루어진 텍스트 파일로 가정합니다.

```
Dim TextLine As String
FileOpen(1, "TESTFILE", OpenMode.Input)   ' Open file.
While Not EOF(1)   ' Loop until end of file.
   TextLine = LineInput(1)
   ' Read line into variable.
   Debug.WriteLine(TextLine)   ' Print to the console.
End While
FileClose(1)   ' Close file.
```

▶ 파일 읽기에 자주 쓰는 함수

♣ LOF 함수 ; LOF(*파일번호*)

: Length Of File

: Open 문으로 연 파일의 크기를 바이트 단위로 나타내는 Long을 반환하는 함수

♣ EOF 함수 ; EOF(*파일번호*)

: End Of File

: 파일의 끝이면 TRUE를 반환하고, 아니면 FALSE를 반환하는 함수

(3) 파일 쓰기 (출력)

① Print, PrintLine 함수

: 출력할 내용을 그대로 파일에 쓴다. 화면 표시용으로 서식이 지정된 데이터를 순차 파일에 씁니다.

Public Sub Print(ByVal *FileNumber* As Integer, ByVal ParamArray *Output*() As Object)

Public Sub PrintLine(ByVal *FileNumber* As Integer, _ ByVal ParamArray *Output*() As Object)

FileNumber : 필수적 요소. 임의의 유효한 파일 번호입니다.

Output : 선택적 요소. 파일에 쓰는 쉼표로 구분된 하나 이상의 식입니다.

Output 인수 설정

다음은 *Output* 인수 설정에 대한 설명입니다.

설정	설명
SPC(*n*)	출력에 공백 문자를 삽입하는 데 사용됩니다. 여기에서 *n*은 삽입할 공백 문자의 개수입니다.
TAB(*n*)	절대 열 번호에 삽입점을 배치하는 데 사용됩니다. 여기에서 *n*은 열 번호입니다. 다음 출력 영역의 시작 부분에 삽입점을 배치하려면 인수 없이 TAB키를 사용합니다.
expression	출력할 숫자 식 또는 문자열 식입니다.

설명

Print는 줄 끝의 줄 바꿈을 포함하지 않지만, PrintLine은 줄 바꿈을 포함합니다.
Print 함수를 사용하여 쓴 데이터는 대개 LineInput 함수나 Input 함수를 사용하여 파일에서 읽습니다.
PrintLine의 경우 *Output*을 생략하면 빈 줄이 파일에 출력되고, Print의 경우에는 아무 것도 출력되지 않습니다. 여러 개의 식이 쉼표로 구분되어 탭 경계에 정렬됩니다. 그러나 쉼표와 TAB 키를 함께 사용하면 일관성 없는 결과가 생성될 수 있습니다.
Boolean 데이터의 경우 True 또는 False가 출력됩니다. True 및 False 키워드는 로케일에 관계 없이 번역되지 않습니다.
날짜 데이터는 시스템에서 인식하는 표준 약식 날짜 형식을 사용하여 파일에 쓰여집니다. 날짜 또는 시간 구성 요소가 0이거나 없는 경우 지정된 부분만 파일에 쓰여집니다.
Output 데이터가 Empty이면 파일에 아무 것도 쓰여지지 않습니다. 그러나 *Output* 목록 데이터가 DBNul이면 파일에 Null이 쓰여집니다.
Error 데이터의 경우 Error errorcode가 출력됩니다. Error 키워드는 로케일에 관계 없이 번역되지 않습니다.
Print 함수를 사용하여 파일에 쓴 모든 데이터는 국가별로 인식됩니다. 즉, 적절한 소수 구분 기호를 사용하여 데이터의 서식이 지정됩니다. 데이터를 출력하여 여러 로케일에서 사용하려면 Write를 사용해야 합니다.

예제

다음 예제에서는 Print 및 PrintLine함수를 사용하여 파일에 데이터를 씁니다.

```
FileOpen(1, "c:₩trash.txt", OpenMode.Output)   ' Open file for output.
Print(1, "This is a test.")   ' Print text to file.
PrintLine(1)   ' Print blank line to file.
PrintLine(1, "Zone 1", TAB(), "Zone 2")   ' Print in two print zones.
PrintLine(1, "Hello", " ", "World")   ' Separate strings with space.
PrintLine(1, SPC(5), "5 leading spaces ")   ' Print five leading spaces.
PrintLine(1, TAB(10), "Hello")   ' Print word at column 10.

' Assign Boolean, Date, and Error values.
Dim aBool As Boolean
Dim aDate As DateTime
aBool = False
aDate = DateTime.Parse("February 12, 1969")

' Dates and booleans are translated using locale settings of your system.
PrintLine(1, aBool, " is a Boolean value")
PrintLine(1, aDate, " is a date")

FileClose(1)   ' Close file.
```

② Write, WriteLine 함수

: 쉼표, 따옴표를 삽입한다

데이터를 순차 파일에 씁니다. Write 함수를 사용하여 쓴 데이터는 대개 Input 함수를 사용하여 파일에서 읽습니다.

Public Sub Write(ByVal *FileNumber* As Integer, ByVal ParamArray *Output* As Object)
- 또는 -

Public Sub WriteLine(ByVal *FileNumber* As Integer, ByVal ParamArray *Output*() As Object)

FileNumber : 필수적 요소. 임의의 유효한 파일 번호가 포함된 **정수**입니다.
Output : 선택적 요소. 파일에 쓰는 쉼표로 구분된 하나 이상의 식입니다.

설명

*Output*을 생략하면 파일에 빈 줄이 출력됩니다. 식이 여러 개 있으면 쉼표로 구분할 수 있습니다. Write 함수는 Print 함수와 달리 파일에 쓰여지는 문자열을 둘러 싸는 인용 부호와 항목 사이에 쉼표를 삽입합니다. 목록에 구분 기호를 삽입할 필요는 없습니다. Write 함수를 사용하여 파일에 데이터를 쓰는 경우 다음과 같은 데이터 형식만 지원됩니다. 또한 로케일에 관계 없이 Input 함수를 사용하여 데이터를 읽고 올바르게 해석할 수 있도록 다음과 같이 일반적인 사항을 가정합니다.

- 숫자 데이터를 쓸 때 항상 소수 구분 기호로 마침표를 사용합니다.

Boolean 데이터의 경우 #TRUE# 또는 #FALSE#가 출력됩니다. True 및 False 키워드는 로케일에 관계 없이 번역되지 않습니다.

- Date 데이터는 일반적인 날짜 형식을 사용하여 파일에 쓰여집니다. 날짜 또는 시간 구성 요소가 0이거나 없는 경우 지정된 부분만 파일에 쓰여집니다.

Output 데이터가 Empty이면 파일에 아무 것도 쓰여지지 않습니다. 그러나 null 데이터의 경우에는 #NULL#이 쓰여집니다.
Error 데이터의 경우 #ERROR errorcode#가 출력됩니다. Error 키워드는 로케일에 관계 없이 번역되지 않습니다.

WriteLine 함수는 *Output*의 마지막 문자를 파일에 쓴 다음 줄 바꿈 문자(캐리지 리턴-줄 바꿈 또는 Chr(13) + Chr(10))를 삽입합니다.

참고

Input 함수를 사용하는 경우 포함된 인용 부호가 들어 있는 문자열(예: "1,2""X")을 쓰지 않도록 해야 합니다. Input 함수는 이 문자열을 두 개의 별도 문자열로 구문 분석합니다.

예제

다음 예제에서는 Write 함수를 사용하여 순차 파일에 원시 데이터를 씁니다.

```
FileOpen(1, "TESTFILE", OpenMode.Output) ' Open file for output.
Write(1, "This is a test.")  ' Print text to file.
WriteLine(1)  ' Print blank line to file.
WriteLine(1, "Zone 1", TAB(), "Zone 2")   ' Print in two print zones.
WriteLine(1, "Hello", " ", "World")      ' Separate strings with space.
WriteLine(1, SPC(5), "5 leading spaces ")    ' Print five leading spaces.
WriteLine(1, TAB(10), "Hello")   ' Print word at column 10.

' Assign Boolean, Date, and Error values.
Dim aBool As Boolean
Dim aDate As DateTime
aBool = False
aDate = DateTime.Parse("February 12, 1969")

' Dates and Booleans are translated using locale settings of
' your system.
WriteLine(1, aBool, " is a Boolean value")
WriteLine(1, aDate, " is a date")

FileClose(1)   ' Close file.
```

4.3. 임의(random) 파일 입·출력

♣ 바이너리 파일을 열고, 읽고, 쓴다. 레코드라고 하는 블록을 단위로 하여 접근하므로 주소록, 기상데이타 등과 같이 일정한 크기를 갖는 파일을 접근하는 경우에 유용하게 쓰인다.

♣ 순차접근에서는 열기모드(Input,Output,Append 중 하나)를 지정하여 그에 따라 읽기 및 쓰기를 하였지만, 임의접근에서는 열기모드는 Random 이다.

(1) 임의 파일 액세스를 위한 변수 선언

응용 프로그램에서는 임의 액세스용으로 파일을 열기 전에 파일의 데이터를 처리하는 데 필요한 모든 변수를 선언해야 합니다. 여기에는 파일의 레코드에 해당하는 사용자 정의 형식과 임의 액세스 파일의 처리와 관련된 데이터를 보유하는 기타 변수의 표준 형식이 포함됩니다.
임의 액세스용으로 파일을 열기 전에 먼저 파일에 포함되어 있거나 포함될 레코드에 해당하는 구조를 정의하십시오. 예를 들어, 다음 예제의 가상 Employee Records 데이터베이스에는 사용자 정의 데이터 형식(Person)이 들어 있습니다.

```
Structure Person
    Public ID As Integer
    Public MonthlySalary As Decimal
    Public LastReviewDate As Long
    <VBFixedString(15)> Public FirstName As String
    <VBFixedString(15)> Public LastName As String
    <VBFixedString(15)> Public Title As String
    <VBFixedString(150)> Public ReviewComments As String
End Structure
```

구조 정의에서의 필드 변수 선언

임의 액세스 파일의 모든 레코드는 길이가 같아야 하므로 사용자 정의 형식의 문자열 요소도 고정 길이를 가지도록 하는 것이 좋습니다. 예를 들어, 위의 Person 형식 선언에서 FirstName 및 LastName은 15자의 고정 길이를 갖습니다. 고정 길이 문자열을 선언하려면 VBFixedString 특성으로 해당 길이를 설정하십시오.
실제 문자열에 문자열이 쓰여지는 문자열 요소의 고정 길이보다 더 적은 수의 문자가 포함된 경우 Visual Basic에서는 레코드의 후행 공백을 공백(문자 코드32)으로 채웁니다. 해당 문자열이 필드 크기보다 더 길면 자동으로 잘립니다. 가변 길이 문자열을 사용하는 경우, FilePut 함수로 저장했거나 FileGet 함수로 검색한 레코드의 전체 크기가 FileOpen 함수에 지정된 레코드 길이보다 크지 않아야 합니다.

(2) 파일 열기

임의 액세스용으로 파일을 열려면 다음 구문으로 FileOpen 함수를 사용합니다.

◦ FileOpen(*FileNumber*, *FileName*, OpenMode.Random, , , *RecordLength*)

FileNumber 및 *FileName*은 각각 열려는 파일의 번호와 파일의 이름을 지정합니다. *RecordLength*는 각 레코드의 크기를 바이트 단위로 지정합니다. *RecordLength*가 파일에 쓰여진 레코드의 실제 길이보다 더 작으면 오류가 발생합니다. *RecordLength*가 레코드의 실제 길이보다 더 크면 일부 디스크 공간이 불필요하게 사용되더라도 레코드는 기록됩니다. Visual Basic에서 모든 String 변수는 ASCII 문자열을 저장하며 사용자는 해당 ASCII 문자열의 바이트 길이를 지정해야 합니다.
임의 파일 액세스를 위한 변수 선언에서 가상 Employee Records 데이터베이스는 먼저 Person 데이터 형식을 정의한 다음 레코드 처리에 사용할 다른 두 변수와 함께 해당 형식에 대한 레코드 변수를 선언했습니다. 다음은 Employee Records 예제에 이어지는 코드로, 사용자 정의 형식 Person의 Employee 데이터를 허용하도록 임의 액세스 파일을 여는 방법을 보여 줍니다.

```
Sub Main()
    Dim FileNum As Integer, RecLength As Long, Employee As Person
    ' Calculate the record length.
```

```
    RecLength = Len(Employee)
    ' Get the next available file number.
    FileNum = FreeFile()
    ' Open the new file with the FileOpen statement.
    FileOpen(FileNum, "MYFILE.DAT", OpenMode.Random, , , RecLength)
End Sub
```

(3) 파일 읽기

임의 액세스 파일을 편집하려면 먼저 파일에서 프로그램 변수로 레코드를 읽어 온 다음, 변수의 값을 변경합니다. 임의 파일 액세스를 위한 변수 선언 및 임의 액세스용으로 파일 열기에서 가상 Employee Records 데이터베이스는 직원 레코드를 읽고 쓰는 데 사용할 변수를 정의했습니다. 다음 절차에서는 이러한 변수를 사용하여 직원 레코드를 읽고 복사하는 방법을 단계적으로 설명합니다.

레코드를 변수로 읽어 온 다음 편집하려면 Employee Records 파일의 레코드를 Employee 변수로 복사하는FileGetObject 메서드를 사용합니다.

- FileGet(FileNum, Employee, Position)
- 이 코드 줄의 FileNum에는 FileOpen 함수가 파일을 여는 데 사용한 번호가 포함되고, Position에는 복사할 레코드의 번호가 포함되며, 사용자 정의 형식 Person으로 선언된 Employee에는 해당 레코드의 내용이 수신됩니다.

(4) 파일 쓰기

임의 액세스 파일에서 프로그램 변수로의 레코드 읽기를 편집한 후에는 FilePut 함수를 사용하여 레코드를 바꾸거나 추가합니다. 임의 파일 액세스를 위한 변수 선언 및 임의 액세스용으로 파일 열기에서 가상 Employee Records 데이터베이스는 직원 레코드를 읽고 쓰는 데 사용할 변수를 정의했습니다. 다음 절차에서는 이러한 변수를 사용하여 직원 레코드를 바꾸고 추가하는 방법을 단계적으로 설명합니다.

레코드를 바꾸려면

FilePut을 사용하여 바꾸려는 레코드의 위치를 지정합니다. 임의 파일 액세스 예제는 다음 코드를 통해 이 작업을 보여 줍니다.

FilePut(FileNum, Employee, Position)

이 코드는 Position에서 지정하는 레코드 번호를 Employee 변수의 데이터로 바꿉니다.

레코드를 추가하려면

FilePut을 사용하여 Position 변수의 값을 파일의 레코드 수에 1을 더한 값으로 설정합니다. 예를 들어, 다섯 개의 레코드를 포함하는 파일에 레코드를 하나 추가하려면 Position을 6으로 설정합니다.

Employee Records 예제의 경우 위의 FilePut 문 대신 다음 문을 사용하면 Position에서 지정한 레코드를 덮어쓰지 않고 파일 끝에 레코드가 추가됩니다.

```
LastRecord = LastRecord + 1
FilePut(FileNum, Employee, LastRecord)
```

4.4. System.IO를 이용한 파일 입출력

```
Imports System.IO

Module TestMod
  Sub Main()
    Dim oFile As FileStream = New FileStream _
      ("C:\Temp\Sample.txt", FileMode.Open, FileAccess.Read)
    Dim oStream As StreamReader = New StreamReader(oFile)
    MsgBox(oStream.ReadLine)
    oFile.Close()
    oStream.Close()
  End Sub
End Module
```

4.5. 문자열을 다루는 함수

.NET Framework의 **String** 클래스는 문자열을 쉽게 비교하고 조작할 수 있도록 다양한 기본 제공 메서드를 제공합니다. 이제 문자열에 대한 데이터를 가져오거나 현재 문자열을 조작하여 새 문자열을 만드는 것은 그리 어려운 일이 아닙니다. 또한 Visual Basic .NET 언어는 이러한 여러 가지 기능을 복제하는 메서드를 기본적으로 가지고 있습니다.

(1) 문자열 조작 메서드 유형

이 단원에서는 문자열을 분석하고 조작할 수 있는 여러 가지 다양한 방법에 대해 설명합니다. 이 중 몇몇 메서드는 Visual Basic 언어의 일부이고 나머지는 **String** 클래스 고유의 것입니다.
Visual Basic .NET 메서드는 해당 언어의 고유 함수로 사용되며 코드에서 자격 증명 없이 사용될 수 있습니다. 다음 예제에서는 Visual Basic .NET 문자열 조작 명령의 일반적인 사용 예를 보여 줍니다.

```
Dim aString As String = "SomeString"
Dim bString As String
bString = Mid(aString, 3, 3)
```

위의 예제에서 **Mid** 함수는 aString에 대해 직접 연산을 수행하여 값을 bString에 대입합니다.
또한 **String** 클래스의 메서드를 사용하여 문자열을 조작할 수도 있습니다. **String** 클래스에는 *공유* 메서드와 *인스턴스* 메서드라는 두 가지 유형의 메서드가 있습니다.
공유 메서드는 **String** 클래스 자체에서 생기고 클래스의 인스턴스로 작업할 필요가 없는 메서드입니다. 이 메서드는 **String** 클래스의 인스턴스 대신 클래스(**String**)의 이름을 사용하여 한정할 수 있습니다. 예를 들면 다음과 같습니다.

```
Dim aString As String
bString = String.Copy("A literal string")
```

위의 예제에서 **String.Copy** 메서드는 정적 메서드로서, 할당된 식에 사용되고 결과 값을 bString에 대입합니다.
이와는 대조적으로 인스턴스 메서드는 **String** 클래스의 특정 인스턴스에서 생기고 인스턴스 이름을 사용하여 한정해야 합니다. 예를 들면 다음과 같습니다.

```
Dim aString As String = "A String"
Dim bString As String
bString = aString.SubString(2,6) ' bString = "String"
```

위의 예제에서 **SubString** 메서드는 **String** 인스턴스(aString)의 메서드이며 aString에 대해 연산을 수행하여 해당 값을 bString에 대입합니다.

(2) Nothing과 문자열

Visual Basic 런타임과 .NET Framework는 **Nothing**이 문자열로 사용되는 경우 약간 다르게 계산됩니다. 다음 예제를 참조하십시오.

```
Dim MyString As String = "This is my string"
Dim stringLength As Integer
' Explicitly set the string to Nothing.
MyString = Nothing
' stringLength = 0
stringLength = Len(MyString)
' This line, however, causes an exception to be thrown.
stringLength = MyString.Length
```

Visual Basic .NET 런타임은 **Nothing**을 빈 문자열("")로 계산합니다. 그러나 .NET Framework는 **Nothing**을 빈 문자열로 계산하지 않으므로 **Nothing**에 대해 문자열 연산을 수행할 때마다 예외를 throw합니다.

(3) 문자열 비교

String.Compare 메서드를 사용하여 두 문자열을 비교할 수 있습니다. 이것은 기본 문자열 클래스의 오버로드된 정적 메서드입니다. 이 메서드를 사용하여 두 문자열을 사전순에 따라 직접 비교할 수 있는데, 이 방법은 가장 일반적이며 Visual Basic StrComp 함수와 비슷합니다. 다음 예제에서는 이 메서드가 사용되는 방법을 보여 줍니다.

```
Dim myString As String = "Alphabetical"
```

```
Dim secondString As String = "Order"
Dim result As Integer
result = String.Compare (myString, secondString)
```

이 메서드는 정렬 순서를 기반으로 하는 두 비교 문자열 간의 관계를 나타내는 정수를 반환합니다. 양수 결과 값은 첫째 문자열이 둘째 문자열보다 크다는 것을 나타냅니다. 음수 결과 값은 첫째 문자열이 작다는 것을 나타내고 0은 두 문자열이 같음을 나타냅니다. 빈 문자열을 포함하는 모든 문자열은 null 참조보다 큰 것으로 계산됩니다.
String.Compare 메서드의 추가 오버로드를 사용하면 대/소문자 또는 culture 서식을 고려할지 여부와 제공된 문자열 내의 부분 문자열을 비교할지 여부를 나타낼 수 있습니다. 문자열을 비교하는 방법에 대한 자세한 내용은 String.Compare 메서드를 참조하십시오. 관련 메서드에는 String.CompareOrdinal 메서드와 String.CompareTo 메서드가 있습니다.

(4) 문자열 내의 문자열 검색

문자열 내의 문자 관련 데이터 및 문자열 내에서 해당 문자의 위치를 알고 있으면 유용할 때가 있습니다. 문자열은 문자의 배열(Char 인스턴스)로 간주할 수 있으며 Chars 속성을 사용하여 해당 문자의 인덱스를 참조하면 특정 문자를 검색할 수 있습니다. 예를 들면 다음과 같습니다.

```
Dim myString As String = "ABCDE"
Dim myChar As Char
myChar = myString.Chars(3) ' myChar = "D"
```

다음 예제와 같이 String.IndexOf 메서드를 사용하여 특정 문자가 있는 인덱스를 반환할 수 있습니다.

```
Dim myString As String = "ABCDE"
Dim myInteger As Integer
myInteger = myString.IndexOf("D")  ' myInteger = 3
```

위의 예제에서는 문자열 내에서 "C" 문자의 첫째 인스턴스에 해당하는 인덱스를 반환하기 위해 myString의 IndexOf 메서드를 사용했습니다. IndexOf는 오버로드된 메서드이고 다른 오버로드는 문자 집합을 검색하거나 문자열 내의 문자열을 검색하기 위한 메서드를 제공합니다. 또한 Visual Basic .NET 명령 InStr를 사용하여 이와 비슷한 기능을 수행할 수도 있습니다. 이러한 메서드에 대한 자세한 내용은 String.IndexOf 메서드와 InStr 함수를 참조하십시오. String.LastIndexOf 메서드를 사용하여 문자열 내에서 문자의 마지막 위치를 검색할 수도 있습니다.

(5) 이전 문자열에서 새 문자열 만들기

문자열을 사용하는 경우 현재 문자열을 수정하여 새 문자열을 만들 수 있습니다. 전체 문자열을 대문자로 변환하거나 후행 공백을 트리밍하는 것처럼 간단한 작업을 수행하거나 문자열에서 부분 문자열을 추출하는 등의 좀 더 복잡한 작업을 수행할 수도 있습니다. System.String 클래스는 문자열을 수정, 조작 및 이전 문자열에서 새 문자열을 만들기 위한 다양한 옵션을 제공합니다.
여러 문자열을 결합하려면 연결 연산자(& 또는 +)를 사용합니다. 또한 String.Concat 메서드를 사용하여 일련의 문자열 또는 개체에 포함된 문자열을 연결할 수도 있습니다. 다음은 String.Concat 메서드 예제입니다.

```
Dim aString As String = "A"
Dim bString As String = "B"
Dim cString As String = "C"
Dim dString As String = "D"
Dim myString As String
' myString = "ABCD"
myString = String.Concat(aString, bString, cString, dString)
```

Visual Basic .NET 함수(UCase 함수와 LCase 함수) 또는 메서드(String.ToUpper 메서드와 String.ToLower 메서드)를 사용하여 문자열을 모두 대문자 또는 모두 소문자로 변환할 수 있습니다. 예제는 아래와 같습니다.

```
Dim myString As String = "UpPeR oR LoWeR cAsE"
Dim newString As String
' newString = "UPPER OR LOWER CASE"
newString = UCase(myString)
' newString = "upper or lower case"
newString = LCase(myString)
' newString = "UPPER OR LOWER CASE"
newString = myString.ToUpper
```

```
' newString = "upper or lower case"
newString = myString.ToLower
```

String.Format 메서드와 Visual Basic .NET Format 명령은 지정된 문자열에 서식을 적용하여 새 문자열을 생성할 수 있습니다. 이 명령에 대한 내용은 Format 함수 또는 String.Format 메서드를 참조하십시오.

때때로 문자열에서 선행 공백이나 후행 공백을 제거해야 할 수도 있습니다. 예를 들어 맞춤 목적으로 공백이 삽입되어 있는 문자열을 구문 분석해야 할 수 있습니다. String.Trim 메서드 함수 또는 Visual Basic .NET의 Trim 함수를 사용하여 이러한 공백을 제거할 수 있습니다. 다음 예제를 참조하십시오.

```
Dim spaceString As String = _
"        This string will have the spaces removed        "
Dim oneString As String
Dim twoString As String
' This removes all trailing and leading spaces.
oneString = spaceString.Trim
' This also removes all trailing and leading spaces.
twoString = Trim(spaceString)
```

후행 공백만 제거하려면 String.TrimEnd 메서드또는 RTrim 함수를 사용하고 선행 공백만 제거하려면 String.TrimStart 메서드 또는 LTrim 함수를 사용합니다. 자세한 내용은 LTrim, RTrim 및 Trim 함수를 참조하십시오.

String.Trim 함수 및 관련 함수를 사용하면 문자열의 끝에서 특정 문자의 인스턴스를 제거할 수도 있습니다. 다음 예제에서는 "#" 문자의 모든 선행 및 후행 인스턴스를 트리밍합니다.

```
Dim myString As String = "#####Remove those!######"
Dim oneString As String
OneString = myString.Trim("#")
```

또한 String.PadLeft 또는 String.PadRight를 사용하여 선행 문자나 후행 문자를 추가할 수도 있습니다.

문자열 본문 내에서 문자가 초과하는 경우 String.Remove 메서드를 사용하여 문자를 삭제하거나 String.Replace 메서드를 사용하여 다른 문자로 바꿀 수 있습니다. 예를 들면 다음과 같습니다.

```
Dim aString As String = "This is My Str@o@o@ing"
Dim myString As String
Dim anotherString As String
' myString = "This is My String"
myString = aString.Remove(14, 5)
' anotherString = "This is Another String"
anotherString = myString.Replace("My", "Another")
```

String.Replace 메서드를 사용하여 개별 문자 또는 문자열을 바꿀 수 있으며 Visual Basic .NET Mid 문을 사용하여 내부 문자열을 다른 문자열로 바꿀 수도 있습니다.

또한 다음 예제와 같이 String.Insert 메서드를 사용하여 다른 문자열 내에 문자열을 삽입할 수도 있습니다.

```
Dim aString As String = "This is My Stng"
Dim myString As String
' Results in a value of "This is My String".
myString = aString.Insert(13, "ri")
```

String.Insert 메서드에서 사용하는 첫째 매개 변수는 뒤에 문자열이 삽입될 문자의 인덱스이고 둘째 매개 변수는 삽입될 문자열입니다.

String.Join 메서드를 사용하여 문자열 배열과 구분 기호 문자열을 함께 연결할 수 있습니다. 다음 예제를 참조하십시오.

```
Dim shoppingItem(2) As String
Dim shoppingList As String
shoppingItem(0) = "Milk"
shoppingItem(1) = "Eggs"
shoppingItem(2) = "Bread"
shoppingList = String.Join(",", shoppingItem)
```

이 코드를 실행한 후 shoppingList의 값은 "Milk,Eggs,Bread"가 됩니다. 배열에 빈 멤버가 있는 경우에도 이 메서드는 배열의 모든 빈 인스턴스 사이에 구분 기호 문자열을 추가합니다.

또한 String.Split 메서드를 사용하여 단일 문자열에서 문자열 배열을 만들 수도 있습니다. 다음 예제에서는 앞의 예제와 반대의 경우를 보여 주는데 쇼핑 목록을 사용하여 쇼핑 항목 배열을 만

듭니다. 이 경우 구분 기호는 Char 데이터 형식의 인스턴스이므로 리터럴 형식 문자 c와 함께 추가됩니다.

```
Dim shoppingList As String = "Milk,Eggs,Bread"
Dim shoppingItem(2) As String
shoppingItem = shoppingList.Split(","c)
```

Visual Basic .NET Mid 함수를 사용하여 문자열에 대한 부분 문자열을 생성할 수 있습니다. 다음 예제에서 이 함수의 사용을 보여 줍니다.

```
Dim aString As String = "Left Center Right"
Dim rString, lString, mString As String
' rString = "Right"
rString = Mid(aString, 13)
' lString = "Left"
lString = Mid(aString, 1, 4)
' mString = "Center"
mString = Mid(aString, 6,6)
```

String.Substring 메서드를 사용하여 문자열에 대한 부분 문자열을 생성할 수도 있습니다. 이 메서드는 두 개의 인수를 받는데 하나는 부분 문자열을 시작할 문자 인덱스이고 또 하나는 부분 문자열의 길이입니다. String.Substring 메서드는 Mid 함수와 매우 유사하게 동작합니다. 예제는 아래와 같습니다.

```
Dim aString As String = "Left Center Right"
Dim subString As String
' subString = "Center"
subString = aString.SubString(5,6)
```

String.SubString 메서드와 Mid 함수 사이에는 매우 중요한 한 가지 차이점이 있습니다. Mid 함수는 시작할 부분 문자열의 문자 위치를 나타내는 인수를 받고(1부터 시작) String.SubString 메서드는 문자열 내에서 부분 문자열을 시작할 문자의 인덱스를 받습니다(0부터 시작). 따라서 "ABCDE"라는 문자열이 있는 경우 각 문자는 Mid 함수와 사용되는 경우 1,2,3,4,5로 번호가 매겨지고 System.String 함수와 사용되는 경우에는 0,1,2,3,4로 번호가 매겨집니다.

4.6. 디렉토리, 파일을 다루는 주요 함수

(1) 일반 함수 이용

함수명	설명
Dir(Path명, 속성)	구체적인 패턴, 파일 속성 혹은 드라이브의 볼륨 레이블과 연관되는 파일, 폴더의 이름에 관한 문자열을 돌려준다. Path명과 속성은 생략 가능하고, 파일명은 드라이브와 폴더를 포함한 파일명이다.
CurDir	현재 폴더에 대한 경로 명을 돌려준다.
ChDir	현재 폴더를 변경한다.
MkDir 폴더명	폴더를 생성한다. 폴더 명에 완전경로가 없으면 현재 폴더에 지정된 서브 폴더를 만든다.
RmDir 폴더명	지정된 폴더를 삭제한다. 폴더 명에 완전경로가 없으면 현재 폴더에 지정된 서브 폴더를 삭제한다.
ChDrive 드라이브명	현재 드라이브를 변경한다.
Kill 파일명	지정된 파일을 삭제한다.
LOF(파일번호)	파일의 길이(bytes)를 출력
EOF(파일번호)	파일의 끝이면 True를 출력

예)

```
If Dir("C:\*.Zip")="" Then
    MsgBox "Zip 파일이 존재하지 않습니다."
End If
'현재 폴더에서 확장자가 Zip인 파일을 찾는다. 찾으면 첫 번째 찾은 파일명을 Dir의 값으로
'리턴한다. 못 찾으면 ""의 문자열(이것은 Null 문자열이 아니고 아무것도 없는 문자열임)을
'돌려준다.
'
Kill "C:\*.bak"  '루트폴더에서 확장자가 bak인 파일을 모두 지운다.
```

(2) Directory 클래스 이용

Visual Basic .NET을 사용하면 Directory 클래스를 통해 폴더 및 드라이브와 관련된 작업을 수행할 수 있습니다. 이 클래스를 사용하면 폴더를 만들거나, 편집하거나, 삭제할 수 있으며 시스템의 드라이브를 유지 관리할 수 있습니다. 다음 표에서는 자주 사용되는 Directory 클래스의 메서드를 보여 줍니다.

방법	설명
CreateDirectory	*경로*로 지정된 모든 디렉터리를 만듭니다.
Delete	오버로딩된 경우를 나타내며, 디렉터리 및 해당 내용을 삭제합니다.
Exists	지정된 경로가 디스크에 있는 기존의 DirectoryInfo 개체를 나타내는지 여부를 확인합니다.
GetDirectories	오버로딩된 경우를 나타내며, 현재 디렉터리에 있는 디렉터

	리 배열을 가져옵니다.
GetDirectoryRoot	지정된 경로의 루트 부분을 반환합니다.
GetLogicalDrives	현재 시스템에 있는 논리 드라이브의 이름을 검색합니다.
GetParent	지정된 경로의 부모 디렉터리를 검색합니다.
Move	DirectoryInfo 개체 및 해당 내용을 새 경로로 이동합니다.
SetCurrentDirectory	현재 디렉터리를 설정합니다.

5. 기타

5.1. 메뉴작성

; 폼에서 메뉴편집기를 이용한다.

5.2. 다이얼로그 박스(대화상자)

; VB에서도 윈도우에서 제공하는 다이얼로그(Common 다이얼로그)박스를 사용한다
; 도구모음-구성요소 中 Microsoft Common Dialog Control을 추가하면 된다

▶ Dialog 컨트롤
*열기(Open) 다이얼로그 박스 **속성:**
다이얼로그박스이름.Filter = "텍스트 문서(*.TXT)|*.TXT|모든파일(*.*)|*.*"

각 필터링 옵션에서 필터 문자열은 필터에 대한 설명 다음에 세로 막대(|)와 필터 패턴을 포함합니다. 서로 다른 필터링 옵션의 문자열은 세로 막대로 구분됩니다.
필터 문자열은 "텍스트 파일 (*.txt)|*.txt|모든 파일(*.*)|*.*"처럼 구성됩니다. *.*"
파일 형식을 세미콜론으로 구분하면 여러 필터 패턴을 하나의 필터에 추가할 수 있습니다. 예: "Image Files(*.BMP;*.JPG;*.GIF)| *.BMP;*.JPG;*.GIF| *.*"

다이얼로그박스이름.InitDir = App. Path *'상자가 열릴때 접근할 디렉토리=현디렉토리*
다이얼로그박스이름.CancelError = False *'취소버튼을 에러로 처리할 것인지의 여부*
다이얼로그박스이름.DefaultExt ="*.TXT" *'열기 대화상자에서 디폴트로 사용할 확장자*

♧ *다이얼로그박스이름*.FileName *'선택한 파일 이름은 FileName속성에 저장된다*
다이얼로그박스이름.FontSize *'선택한 폰트사이즈는 FontSize속성에 저장된다*
다이얼로그박스이름.FromPage *'선택한 시작페이지는 FromPage속성에 저장된다*
다이얼로그박스이름.Color *'선택한 칼라는 Color속성에 저장된다*

등등....

※ Font 다이얼로그에서는 반드시 Flags 속성을 설정해 주어야 에러가 발생하지 않는다
다이얼로그박스이름.Flags = cdlCFBoth Or cdlCFEffects

5.3. 디버깅과 에러처리

(1) 개요

1. '도구-옵션' 의 편집기 탭에서의 설정
2. 디버그 모드
   - ▶ F5 : 실행
   - ▶ F11 : 한 단계씩 코드 실행
   - ▶ F10 : 프로시저 단위 실행
   - ▶ Ctrl+B : 중단점
3. Debug 오브젝트

   실행중에 디버그 윈도우에 결과를 임시로 출력한다

   그러나, 실행파일이 만들어지고나면 Debug와 관련된 문장은 무시된다

Visual Basic 참고 : Visual Basic .NET에서는 Visual Basic의 이전 버전에서 사용되었던 Debug.Print 메서드 대신 Debug.Write, Debug.WriteIf, Debug.WriteLine 및 Debug.WriteLineIf 메서드를 사용합니다.

(2) 응용 프로그램 코드에 추적 문 추가

추적에 가장 흔히 사용되는 메서드는 Write, WriteIf, WriteLine, WriteLineIf, Assert, Fail과 같이 수신기에 출력을 쓰기 위한 메서드입니다. 이러한 메서드는 두 분류로 나눌 수 있습니다. Write, WriteLine 및 Fail 메서드는 모두 무조건적으로 출력을 내보내는 반면 WriteIf, WriteLineIf 및 Assert는 Boolean 조건을 테스트한 다음 조건의 값에 따라 쓰거나 쓰지 않습니다. WriteIf와 WriteLineIf는 조건이 true일 경우 출력을 내보내고 Assert는 조건이 false일 경우 출력을 내보냅니다.

추적 및 디버깅 전략을 구상할 때는 출력 모양을 고려해야 합니다. 관련 없는 정보로 채워진 다수의 Write 문을 사용하면 읽기 어려운 로그가 만들어집니다. 반면에 WriteLine을 사용하여 상호 관련 있는 문을 분리된 줄에 배치하면 어떤 정보가 서로 연결되어 있는지 구분하기가 어렵습니다. 일반적으로 여러 소스의 정보를 결합하여 단일 정보 메시지를 만들 경우에는 다수의 Write 문을 사용하고 완전한 단일 메시지를 만들려는 경우에는 WriteLine 문을 사용합니다.

완전한 줄을 쓰려면

- WriteLine 또는 WriteLineIf 메서드를 호출합니다.
- 이 메서드가 반환하는 메시지 끝에는 캐리지 리턴이 추가되므로 Write, WriteIf, WriteLine 또는 WriteLineIf가 반환하는 다음 메시지는 그 다음 줄에서 시작됩니다.
- ' Visual Basic
- Dim errorFlag as Boolean = False
- Trace.WriteLine("Error in AppendData procedure.")
- Trace.WriteLineIf(errorFlag, "Error in AppendData procedure.")

부분적인 줄을 쓰려면

- Write 또는 WriteIf 메서드를 호출합니다.
- Write, WriteIf, WriteLine 또는 WriteLineIf가 출력하는 다음 메시지가 Write 또는 WriteIf 문이 출력하는 메시지와 같은 줄에서 시작됩니다.
- ' Visual Basic
- Dim errorFlag as Boolean = False
- Trace.WriteIf(errorFlag, "Error in AppendData procedure.")
- Debug.WriteIf(errorFlag, "Transaction abandoned.")
- Trace.Write("Invalid value for data request")

메서드 실행 전후에 특정 조건이 있는지 확인하려면

```
▫ Assert 메서드를 호출합니다.
▫ ' Visual Basic
▫ Dim I as Integer = 4
▫ Trace.Assert(I = 5, "I is not equal to 5.")
▫
```

참고

추적과 디버깅 모두에서 Assert를 사용할 수 있습니다. 이 예제에서는 호출 스택을 Listeners 컬렉션의 모든 수신기로 출력합니다. 자세한 내용은 관리되는 코드에 어설션 사용 및 Debug.Assert 메서드를 참조하십시오.

(3) 스위치를 사용하여 조건부 쓰기 제어

스위치 상태에 따라 추적 출력을 쓸 것인지를 결정할 수도 있습니다. BooleanSwitch는 단순한 설정/해제 스위치이며 TraceSwitch는 여러 수준의 설정이 있는 스위치입니다. 자세한 내용은 추적 스위치를 참조하십시오.

WriteIf 및 WriteLineIf 메서드를 사용하여 특정 스위치를 테스트하고 적합한한 경우 메시지를 쓸 수 있습니다. BooleanSwitch를 사용하여 추적 정보를 쓰려면 If 문 또는 WriteIf 문에 Enabled 필드를 사용하십시오. 몇 가지 예제에 대해서는 .NET Framework 참조의 BooleanSwitch.Enabled 속성을 참조하십시오. TraceSwitch를 사용하여 추적 정보를 쓰려면 TraceSwitch의 Level 속성을 사용하십시오. 예제는 .NET Framework 참조의 TraceSwitch.Level 속성을 참조하십시오.

다음 예제에서는 dataSwitch라는 BooleanSwitch의 Enabled 속성을 사용하여 줄을 쓸 것인지의 여부를 결정합니다.

```
' Visual Basic
Trace.WriteLineIf(dataSwitch.Enabled, "Starting connection procedure")
```

TraceSwitch는 여러 설정 수준을 제공하고 이러한 수준에 해당하는 속성 집합을 노출합니다. 그러므로 Boolean 속성 TraceError, TraceWarning, TraceInfo 및 TraceVerbose는 WriteIf 또는 WriteLineIf 문의 일부분으로 처리될 수 있습니다. 이 예제의 코드는 TraceSwitch가 Error 이상의 추적 수준으로 설정된 경우에만 지정된 정보를 씁니다.

```
' Visual Basic
Trace.WriteLineIf(myTraceSwitch.TraceError, "Error 42 occurred")
```

참고

TraceSwitch를 테스트할 때 TraceSwitch가 테스트할 수준 이상이면 true로 간주됩니다.앞의 예제에서는 추적 기능이 사용되는 경우 항상 WriteLineIf 메서드를 호출합니다. 그러므로 이 예제에서는 WriteLineIf의 둘째 인수를 확인하는 데 필요한 코드를 항상 실행해야 합니다. 그러나 다음 코드를 사용하여 BooleanSwitch를 먼저 테스트한 다음 테스트가 성공한 경우에만 일반 Trace.Write 메서드를 호출하면 성능이 일반적으로 더욱 향상됩니다.

```
' Visual Basic
If MyBooleanSwitch.Enabled Then
    Trace.WriteLine("Error 42 occured")
End If
```

추적 메서드를 호출하기 전에 Boolean 값을 테스트하면 추적 기능이 항상 WriteLineIf의 모든 매개 변수를 확인하므로 불필요한 코드가 실행되지 않습니다. 이 방법은 응용 프로그램이

정상 동작 모드일 때 TraceSwitch가 해제된 경우에만 성능이 향상됩니다. TraceSwitch가 설정되어 있으면 응용 프로그램이 WriteLineIf의 모든 매개 변수를 확인해야 하므로 응용 프로그램의 전체적인 실행 시간이 늘어납니다.

<과제> 텍스트 파일에 다음의 형식으로 각 층의 재료 및 물성치를 기입하여 프로그램에서 로딩하면 조합된 구조체의 **열관류율**(K-value)을 구하는 프로그램을 작성하라.

입력파일 형식: 한 줄 단위로 다음의 변수들을 입력
구조체 이름, 구조체 위치순서(왼쪽부터 1번), 이 층의 두께, 열전도율

조건:
· 파일의 로딩은 실시간으로 다이얼로그 박스를 통해서 할 것
· 파일을 읽을 때 순환문을 사용하여 EOF를 만날 때까지 읽도록 할 것
· 위의 입력데이터를 배열의 형태로 다루도록 한다.
· 구조체의 개수는 최대 5개로 제한하며, 계절은 여름으로 한정하고 중공층은 없는 것으로 한다.
· 접촉저항은 무시한다.

<과제> 위의 과제에서 각층의 재료 데이터를 사용자 정의 데이터형(Type)으로 설정하여 프로그램을 변경하라.
· Type내의 변수는 Name, Number, Thickness, HTC(열전도율)를 반드시 포함할 것

6. Visual Basic 언어 키워드

다음 표에는 Visual Basic .NET 언어 키워드가 나열되어 있습니다. 모든 언어 키워드는 예약되어 있습니다.

AddHandler	AddressOf	Alias	And
AndAlso	Ansi	As	Assembly
Auto	Boolean	ByRef	Byte
ByVal	Call	Case	Catch
CBool	CByte	CChar	CDate
CDec	CDbl	Char	CInt
Class	CLng	CObj	Const
CShort	CSng	CStr	CType
Date	Decimal	Declare	Default
Delegate	Dim	DirectCast	Do
Double	Each	Else	ElseIf
End	Enum	Erase	Error
Event	Exit		False
		#ExternalSource	
Finally	For	Friend	Function
Get	GetType	GoTo	Handles
If...Then...Else	Implements	Imports	In
Inherits	Integer	Interface	Is
Let	Lib	Like	Long
Loop	Me	Mod	Module
MustInherit	MustOverride	MyBase	MyClass
Namespace	New	Next	Not
Nothing	NotInheritable	NotOverridable	Object
On	Option	Optional	Or
OrElse	Overloads	Overridable	Overrides
ParamArray	Preserve	Private	Property
Protected	Public	RaiseEvent	ReadOnly
ReDim		REM	RemoveHandler
	#Region		
Resume	Return	Select	Set
Shadows	Shared	Short	Single
Static	Step	Stop	String
Structure	Sub	SyncLock	Then
Throw	To	True	Try
TypeOf	Unicode	Until	Variant
When	While	With	WithEvents
WriteOnly	Xor	#Const	#ExternalSource
#If...Then...#Else	#Region	-	&
&=	*	*=	/
/=	₩	₩=	^
^=	+	+=	=
-=			

참고 Variant와 Let은 Visual Basic .NET에서는 더 이상 사용되지 않지만 키워드로서 보유됩니다.

7. 프로그래밍 요소 지원 변경 사항 요약

Visual Basic .NET에서는 주로 공용 언어 런타임과의 상호 운용성을 위해 다양한 프로그래밍 요소를 지원하는 방식이 변경되었습니다. 대부분의 Visual Basic 6.0 요소의 이름이 바뀌고 다시 분류되거나 Visual Basic .NET의 다른 프로그래밍 요소와 결합되었습니다. 공용 언어 런타임에 포함된 기능으로 인해 필요하지 않게 된 몇 가지 요소는 더 이상 지원되지 않습니다.
IDE(통합 개발 환경), 웹 기능, 프로젝트, 폼, 상수, Circle, Line 및 Pset 메서드를 비롯하여 Visual Basic에 발생한 변경 내용에 대한 자세한 내용은 숙련된 Visual Basic 개발자를 위한 Visual Basic .NET 소개를 참조하십시오.
다음 표에서는 변경된 프로그래밍 요소와 이 프로그래밍 요소를 대체하는 요소를 보여 줍니다.

Visual Basic 6.0 프로그래밍 요소	Visual Basic .NET의 해당 프로그래밍 요소	네임스페이스, 클래스, 런타임 라이브러리 위치
Abs 함수	Abs 메서드	System 네임스페이스, Math 클래스
AscB 함수	Asc 함수	Visual Basic 런타임 라이브러리 멤버, Strings 모듈
As Any 키워드 구	Visual Basic .NET에서는 지원되지 않습니다. Visual Basic의 선언 구문 변경 사항 참조	적용되지 않음
Atn 함수	Atan 메서드	System 네임스페이스, Math 클래스
Calendar 속성	CurrentCulture 속성	System.Globalization 네임스페이스, CultureInfo 클래스
ChDir 문	ChDir 함수	Visual Basic 런타임 라이브러리 멤버, FileSystem 모듈
ChDrive 문	ChDrive 함수	Visual Basic 런타임 라이브러리 멤버, FileSystem 모듈
Chr$, ChrB, 함수	Chr 함수	Visual Basic 런타임 라이브러리 멤버, Strings 모듈
Close 문	FileClose 함수	Visual Basic 런타임 라이브러리 멤버, FileSystem 모듈
Cos 함수	Cos 메서드	System 네임스페이스, Math 클래스
Currency 데이터 형식	Decimal 데이터 형식. Visual Basic의 데이터 형식 변경 사항 참조	Visual Basic 런타임 라이브러리 멤버, VariantType 열거형
CVDate 함수	DateValue 함수	Visual Basic 런타임 라이브러리 멤버, DateAndTime 모듈
CVError 함수	Error 문	적용되지 않음
Date 함수, Date 문	Now 속성, Today 속성	Visual Basic 런타임 라이브러리 멤버, DateAndTime 모듈
Date$ 함수	DateString 속성	Visual Basic 런타임 라이브러리 멤버, DateAndTime 모듈
Debug.Assert 메서드	Assert, Fail 메서드	System.Diagnostics 네임스페이스, Debug 클래스
Debug.Print 메서드	Write, WriteIf, WriteLine 및 WriteLineIf 메서드	System.Diagnostics 네임스페이스, Debug 클래스
Deftype 문	Visual Basic .NET에서는 지원되지 않습니다. Visual Basic의 데이터 형식 변경 사항 참조	적용되지 않음
DeleteSetting 문	DeleteSetting 함수	Visual Basic 런타임 라이브러리 멤버, Interaction 모듈
DoEvents 함수	DoEvents 메서드	System.Windows.Forms 네임스페이스, Application 클래스
Empty 키워드	Nothing	적용되지 않음
Eqv 연산자	= 연산자. Visual Basic의 부울 연산자 변경 사항 참조	적용되지 않음
Exp 함수	Exp 메서드	System 네임스페이스, Math 클래스
FileCopy 문	FileCopy 함수	Visual Basic 런타임 라이브러리 멤버, FileSystem 모듈
Get 문	FileGet 함수	Visual Basic 런타임 라이브러리 멤버, FileSystem 모듈
GoSub 문	Visual Basic .NET에서는 지원되지 않습니다. Return 문을 사용하십시오. Visual Basic의 제어문 변경 사항 참조	적용되지 않음
Initialize 이벤트	Visual Basic .NET에서는 지원되지 않습니다. Sub New를 사용하십시오. Visual Basic의 클래스 및 인터페이스 변경 사항과 생성자 및 소멸자 사용 참조	적용되지 않음

Imp 연산자	Visual Basic .NET에서는 지원되지 않습니다. Not 및 Or 연산자, Visual Basic의 부울 연산자 변경 사항 참조	적용되지 않음
Input #, Input$ 문, Input$, InputB, InputB$ 함수	Input 함수	Visual Basic 런타임 라이브러리 멤버, FileSystem 모듈
Instancing 속성	Visual Basic .NET에서는 지원되지 않습니다. Visual Basic의 클래스 및 인터페이스 변경 사항 참조	적용되지 않음
InStrB 함수	InStr 함수	Visual Basic 런타임 라이브러리 멤버, Strings 모듈
IsEmpty 함수	IsNothing 함수	Visual Basic 런타임 라이브러리 멤버, Information 모듈
IsMissing 함수	Visual Basic .NET에서는 지원되지 않습니다. Visual Basic의 프로시저 선언 변경 사항 참조	적용되지 않음
IsNull 함수	IsDbNull 함수	Visual Basic 런타임 라이브러리 멤버, Information 모듈
IsObject 함수	IsReference 함수	Visual Basic 런타임 라이브러리 멤버, Information 모듈
Kill 문	Kill 함수	Visual Basic 런타임 라이브러리 멤버, FileSystem 모듈
LCase$ 함수	LCase 함수	Visual Basic 런타임 라이브러리 멤버, Strings 모듈
Left$, LeftB, LeftB$ 함수	Left 함수	Visual Basic 런타임 라이브러리 멤버, Strings 모듈
LenB 함수	Len 함수	Visual Basic 런타임 라이브러리 멤버, Strings 모듈
Let, Set 대입문	Visual Basic .NET에서는 지원되지 않습니다. 새 Set 문은 이전의 Set 문과 관련이 없습니다. Visual Basic의 기본 속성 변경 사항 참조	적용되지 않음
Line Input # 문	LineInput 함수	Visual Basic 런타임 라이브러리 멤버, FileSystem 모듈
Lock 문	Lock, Unlock 함수	Visual Basic 런타임 라이브러리 멤버, FileSystem 모듈
Log 함수	Log 메서드	System 네임스페이스, Math 클래스
LSet, RSet 문	LSet 함수, PadRight, PadLeft. Visual Basic의 데이터 형식 변경 사항 참조	Visual Basic 런타임 라이브러리 멤버, Strings 모듈 및 System 네임스페이스, String 클래스
LTrim$ 함수	LTrim 함수	Visual Basic 런타임 라이브러리 멤버, Strings 모듈
MidB 함수	Mid 함수	Visual Basic 런타임 라이브러리 멤버, Strings 모듈
MidB 문	Mid 문	Visual Basic 런타임 라이브러리 멤버, Strings 모듈
MkDir 문	MkDir 함수	Visual Basic 런타임 라이브러리 멤버, FileSystem 모듈
Name 문	Rename 함수	Visual Basic 런타임 라이브러리 멤버, FileSystem 모듈
Now 함수	Now 속성	Visual Basic 런타임 라이브러리 멤버, DateAndTime 모듈
Null 키워드	Nothing	적용되지 않음
Oct$ 함수	Oct 함수	Visual Basic 런타임 라이브러리 멤버, Conversion 모듈
On ... GoSub 구문	Visual Basic .NET에서는 지원되지 않습니다. Select Case 문을 사용하십시오. Visual Basic의 제	적용되지 않음

	어문 변경 사항 참조	
On ... GoTo 구문	Visual Basic .NET에서는 지원되지 않습니다. Select Case 문을 사용하십시오. Visual Basic의 제어문 변경 사항 참조	적용되지 않음
Open 문	FileOpen 함수	Visual Basic 런타임 라이브러리 멤버, FileSystem 모듈
Option Base 문	Visual Basic .NET에서는 지원되지 않습니다. Visual Basic의 배열 범위 변경 사항 참조	적용되지 않음
Option Private Module 문	Visual Basic .NET에서는 지원되지 않습니다. Module 문을 사용하십시오. Visual Basic의 클래스 및 인터페이스 변경 사항 참조	적용되지 않음
Print # 문	Print, PrintLine 함수	Visual Basic 런타임 라이브러리 멤버, FileSystem 모듈
Property Get, Property Let, Property Set 문	Visual Basic .NET에서는 지원되지 않습니다. Visual Basic의 속성 변경 사항 참조	적용되지 않음
Put 문	FilePut 함수	Visual Basic 런타임 라이브러리 멤버, FileSystem 모듈
Reset 문	Reset 함수	Visual Basic 런타임 라이브러리 멤버, FileSystem 모듈
Right$, RightB 함수	Right 함수	Visual Basic 런타임 라이브러리 멤버, Strings 모듈
RmDir 문	RmDir 함수	Visual Basic 런타임 라이브러리 멤버, FileSystem 모듈
Round 함수	Round 메서드	System 네임스페이스, Math 클래스
RSet, LSet 문	RSet 함수. Visual Basic의 데이터 형식 변경 사항 참조	Visual Basic 런타임 라이브러리 멤버, Strings 모듈
RTrim$ 함수	RTrim 함수	Visual Basic 런타임 라이브러리 멤버, Strings 모듈
SaveSetting 문	SaveSetting 함수	Visual Basic 런타임 라이브러리 멤버, Interaction 모듈
Scale 메서드	Visual Basic .NET에서는 지원되지 않습니다.	적용되지 않음
Set, Let 대입문	Visual Basic .NET에서는 지원되지 않습니다. 새 Set 문은 이전의 Set 문과 관련이 없습니다. Visual Basic의 기본 속성 변경 사항 참조	적용되지 않음
SetAttr 문	SetAttr 함수	Visual Basic 런타임 라이브러리 멤버, FileSystem 모듈
Sgn 함수	Sign 메서드	System 네임스페이스, Math 클래스
Sin 함수	Sin 메서드	System 네임스페이스, Math 클래스
Sqr 함수	Sqrt 메서드	System 네임스페이스, Math 클래스
String 함수	String 생성자. Visual Basic의 String ($) 함수 변경 사항 참조	System 네임스페이스, String 클래스
String ($) 함수	Visual Basic .NET에서는 지원되지 않습니다. Visual Basic의 String ($) 함수 변경 사항 참조	적용되지 않음
Terminate 이벤트	Visual Basic .NET에서는 지원되지 않습니다. Sub Dispose 및 Sub Finalize를 사용하십시오. Visual Basic의 클래스 및 인터페이스 변경 사항 및 생성자 및	적용되지 않음

	소멸자 사용 참조	
Time 함수, Time 문	TimeOfDay 속성. DateTime 구조체, Date 데이터 형식 참조	Visual Basic 런타임 라이브러리 멤버, DateAndTime 모듈
Time$ 함수	TimeString 속성	Visual Basic 런타임 라이브러리 멤버, DateAndTime 모듈
Timer 함수	Timer 속성	Visual Basic 런타임 라이브러리 멤버, DateAndTime 모듈
Trim$ 함수	LTrim, RTrim 및 Trim 함수	Visual Basic 런타임 라이브러리 멤버, Strings 모듈
Type 문	Visual Basic .NET에서는 지원되지 않습니다. Structure 문을 사용하십시오. Visual Basic의 구조체 선언 변경 사항 참조	적용되지 않음
UCase$ 함수	UCase 함수	Visual Basic 런타임 라이브러리 멤버, Strings 모듈
Unlock 문	Lock, Unlock 함수	Visual Basic 런타임 라이브러리 멤버, FileSystem 모듈
Variant 데이터 형식	Object 데이터 형식. Visual Basic의 유니버설 데이터 형식 변경 사항 참조	적용되지 않음
Wend 키워드	While...End While 문 및 End 문. Visual Basic의 제어문 변경 사항 참조	적용되지 않음
Width # 문	FileWidth 함수	Visual Basic 런타임 라이브러리 멤버, FileSystem 모듈
Write # 문	Write, WriteLine 함수	Visual Basic 런타임 라이브러리 멤버, FileSystem 모듈