모듈 3

변수Variable들, 변수의 타입Type들, 타입 변환하기Conversion

변수(데이터를 메모리에 담아두는 그릇)을 사용할 수 있게 됩니다.
연산자를 사용하여 변수에 값을 담거나 숫자 계산을 할 수 있게 됩니다.
숫자, 문자 등 여러가지 유형의 데이터를 담기 위한 변수의 타입을 이해합니다.
타입 변환 (예: 숫자 타입에서 문자 타입)을 할 수 있게 됩니다.

입력input 을 받아서 한가지 표현식expression을 통해 값value을 계산해내기 위해, 그것도 단 한번만 하기 위해, 완전한 기능을 갖춘 소프트웨어 프로그램을 작성할 필요는 없다. 하지만, 이런 계산을 자주 해야 하고 게다가 입력이 매번 다르다면, 계산 절차process 를 자동화하는 것이 마땅하다.

할당Assignment 연산자Operator

여러분이 각도degree, DEGR를 라디안radian, RAD으로 변환하고 싶을 때, 변환 공식을 델파이로 적어보면 다음과 같다.

```
RAD:= (DEGR * 3.14) / 180;
```

할당assignment 연산자operator는 모든 프로그래밍 언어에서 가장 기본이고 중요한 명령이다. 할당 기호 := 오른쪽에 있는 표현식에서 계산된 값은 할당 기호 왼쪽에 있는 변수variable 안으로 들어간다. 할당 기호는 기호 두 개(: 와 =)로 만든다. 하지만 델파이는 이것을 하나의 기호로 해석한다.

내부 작동 방식은 다음과 같다. 할당 연산자 오른쪽에 있는 수식을 먼저 계산하여 값을 구한다. 그리고 나서 이 값을 할당 연산자 왼쪽에 있는 변수에 할당한다.

할당 연산자 형식

```
<변수 이름>:= <표현식>;
```

이제, 변수variable에 대해 말해보자. 변수란 메모리 안에 있는 어느 위치이며 이름을 가진다. 변수는 안에 들어가는 값에 따라 그 변수를 위해 확보하는 메모리의 크기가 다르다. 모든 변수를 받을 수 있도록 커다란 메모리 공간 하나를 확보할 수도 있겠지만, 그렇게 하면 사용할 수 있는 메모리가 금새 바닥나게 된다. 변수에 할당된 메모리 크기를 정확히 알려면 그 변수의 데이터data 타입type을 알아야 한다.

델파이에서 변수가 필요한 작업이나 이유들은 다양하다. 변수를 사용하려면, 먼저 변수를 정의define해야 한다. 변수 정의 구역은 *var* 키워드로 시작한다. 이 구역은 프로시저 정의 뒤에 이어지는 *begin* 키워드의 앞에 위치한다.

변수 정의 형식

```
var
   <변수 이름>: <데이터 타입>;
```

변수 이름은 다음의 기준에 따라 문자를 나열하여 만든다.

- 영문자, 숫자(0-9), 밑줄
- 반드시 영문자로 시작

변수 이름의 길이는 얼마든지 길어도 되며(문자 수 제한이 없다), 대소문자를 가리지 않는다.

한 글자로 변수 이름을 만들지는 말자. 반복 중인 횟수를 가지고 있기 위한 변수나 잠시만 필요한 변수 등 이름이 될 만한 의미가 없는 경우에는 한 글자로 된 이름도 괜찮다.

반복 중인 횟수를 가지고 있기 위한 변수로는 I와 J를 쓴다. 변수 이름이 한 글자인 다른 예로는 SString, [스트링, 문자열] 또는 RRadius, 반지름이 있다. 이름을 정할 때는 그 변수가 가진 의미를 이

름 만으로 바로 알아 차릴 수 있도록 하자. l(소문자 L)은 변수 이름으로 쓰지 말자. 숫자 1과 혼동될 수 있다.

예문

```
procedure TfrmMy.btnMyClick(Sender: TObject);
var
   k: Integer;
begin
end;
```

위 예에서 변수 k 를 *Integer* 타입으로 선언하였다.

델파이에는 데이터 타입이 많다. 데이터 타입의 일부를 살펴보자.

정수는 아래에 있는 데이터 타입들 중 하나로 표현할 수 있다.

정수형 데이터 타입(data tape)들

데이터 타입	범위	표현
Integer	−2,147,483,648 ~ 2,147,483,647	32-bit, signed
Cardinal	0 ~ 4,294,967,295	32-bit, 양수
ShortInt	−128 ~ 127	8-bit, signed
SmallInt	−32,768 ~ 32,767	16-bit, signed
LongInt	−2,147,483,648 ~ 2,147,483,647	32-bit, signed
Int64	-2^{63} ~ $2^{63}-1$	64-bit, signed
Byte	0 ~ 255	8-bit, unsigned
Word	0 ~ 65535	16-bit, unsigned
LongWord	0 ~ 4,294,967,295	32-bit, unsigned

이 책에서는 주로 *Integer*[인티저] 정수를 사용한다.

실수를 위한 데이터 타입들도 다양하다. 우리는 *Real*[리얼] 데이터 타입을 쓰겠다.

실수형 데이터 타입(data tape)들

데이터 타입	범위	유효숫자 자릿수 (정밀도)	바이트 (Bytes)
Real48	$\pm 2.9 \times 10^{-39} \sim \pm 1.7 \times 10^{38}$	11 – 12	6
Real	$\pm 5.0 \times 10^{-324} \sim \pm 1.7 \times 10^{308}$	15–16	8
Single	$\pm 1.5 \times 10^{-45} \sim \pm 3.4 \times 10^{38}$	7–8	4
Double	$\pm 5.0 \times 10^{-324} \sim \pm 1.7 \times 10^{308}$	15–16	8
Extended	$\pm 3.6 \times 10^{-4951} \sim \pm 1.1 \times 10^{4932}$	19–20	10
Comp	$-2^{63} \sim 2^{63} - 1$	19–20	8
Currency	–922337203685477.5808 ~ +922337203685477.5807	19–20	8

실수는 정규화된 과학 표기법(지수 표기법)으로 표현한다. 예를 들어,

$3{,}28 \cdot 10^{17}$ $1{,}4 \cdot 10^{-9}$ $-5{,}101 \cdot 10^{4}$

델파이에서, 이 숫자들은 다음과 같이 적는다.

3.28e+17	1.4e–09	–5.101e+4
328e15	0.14e–8	–5101e+1
0.328e+18	140e–11	–510100e–1

산술Arithmetic 표현식Expression

할당 연산자를 다시 살펴보자. 왼쪽에 있는 값의 데이터 타입은 오른쪽에 있는 산술 표현식에서 계산된 결과값이 가지게 되는 데이터 타입과 짝이 맞아야 한다.

산술Arithmetic 표현식Expression은 규칙에 맞는 산수 계산 표현식이다. 이것은 상수constant, 변수variable, 함수function, 산술 연산자(+,*,- , / , 지수 등), 괄호 (와) 를 가지고 만든다.

산술 표현식에 사용되는 연산자(operators)들

연산자	연산자 이름	피연산수의 데이터 타입	결과 데이터 타입
+	더하기	Integer	Integer
		Real이 1개 이상 있으면	Real
–	빼기	Integer	Integer
		Real이 1개 이상 있으면	Real
*	곱하기	Integer	Integer
		Real이 1개 이상 있으면	Real
/	나누기	Real, Integer	Real
Div	인티저 나누기	Integer	Integer
Mod	인티저 나누기의 나머지	Integer	Integer

예문

13 div 4 = 3　　13 mod 4 = 1

–13 div 4 = –3　　–13 mod 4 = –1

13 div –4 = –3　　13 mod –4 = 1

–13 div –4 = 3　　–13 mod –4 = –1

0 div 2 = 0　　1 mod 2 = 1

2 div 5 = 0　　2 mod 5 = 2

계산된 값이 정수라면, 실수 데이터 타입 안에 넣을 수 있다. 하지만, 반대로 실수 값을 정수 타입에 넣을[assign] 수는 없다.

숫자 두개를 더하기

`frmCalc`라는 폼을 만들고 그 안에 `edtA`과 `edtB`라는 텍스트박스 2개, `btnAdd`라는 버튼 하나, `lblResult`라는 레이블 하나를 올려둔다.

`btnAdd` 버튼의 *Caption* 프로퍼티를 셋팅한다. 텍스트박스의 *Text* 프로퍼티는 비워둔다. `btnAdd` 버튼을 더블클릭한다. 코드 창이 나타난다. 우리는 여기에 버튼 클릭 이벤트핸들러

의 코드를 작성한다.

먼저 변수를 선언하자.

```
var
a, b, c: Single;
```

텍스트박스 edtA에서 a 값을, edtB에서 b 값을 가져온다. 텍스트박스에는 *String*[문자열] 값이 들어간다는 점을 명심하자. 문자열을 숫자 데이터 타입으로 변환해야 덧셈을 할 수 있다. 함수 *StrToFloat*(문자열)를 사용한다.

```
begin
    a:= StrToFloat(edtA.Text); //숫자 A에 값을 넣는다.
    b:= StrToFloat(edtB.Text); //숫자 B에 값을 넣는다
    c:= a + b; //A와 B를 더한다
    lblResult.Visible:= True; //레이블이 보이도록 한다
    lblResult.Caption:= FloatToStr(c); //결과값을 레이블에 넣는다 (넣기 전에 우선
String으로 변환해야 한다)
end;
```

변수 두 개를 더한 결과는 세 번째 변수 c에 저장하고, 그 결과를 다시 *String*으로 변환하기 위해 *FloatToStr*(숫자)을 사용한다. 그리고 나서 lblResult 레이블을 표시한다. 레이블을 표시하는 코드를 다음과 같이 바꾸면 결과가 더 좋게 표현된다.

```
lblAnswer.Caption:= edtA.Text +'+' +edtB. Text +'=' +FloatToStr(c);
```

델파이에서 *StrToFloat*과 *FloatToStr*을 사용할 때에는 소수점 형식을 잘 보아야 한다. 프로그램 유닛[Unit] 안에서 소수점 표시는 반드시 마침표만 사용할 수 있다. 하지만, 입력 창이나 출력 창에 표시되는 소수점은 운영체제의 지역 설정에 따라 달라진다. 예를 들어 유럽 국가들 중에는 소수점으로 쉼표를 사용하는 곳도 많다.

함수 *IntToStr*(정수)은 *Integer*를 *String*으로, 함수 *StrToInt*(문자열)는 *String*을 *Integer*로 변환한다.

실습

Exercise 1.

텍스트박스 2개, 레이블 1개, 버튼 4개를 사용하여, 사칙연산을 하는 계산기를 만들자.

Exercise 2.

화씨Fahrenheit와 섭씨Celsius를 서로 변환해주는 프로그램을 만들자.

(Tf=9/5*Tc+32).

Exercise 3.

시속 0 km와 초속 0 m 를 서로 변환해주는 프로그램을 만들자.