

[언어 트랙 - 기초] 델파이 언어 기초2

- 루틴이 왜 필요한지 ?
- 함수와 프로시저의 차이점
- 함수와 프로시저 선언 방법
- 매개변수 전달 방법
 - Call by Value
 - Call by Reference
- 매개변수 디폴트 값 설정
- 오버로드 함수 정의
- 프로시저/함수 호출
- [실습] 프로시저와 함수 작성해 보기

[언어 트랙 - 기초] 델파이 언어 기초2

•프로시저와 함수가 왜 필요한가?

프로그램을 작성할 때 어떤 특정 업무(비즈니스 로직)를 코드로 구현한다. 이러한 코드가 한 부분에서만 필요한 것이 아니고 여러 부분에서 필요하다면 아래와 같이 코드를 구현 할 것이다.

소스 A

```
Statement1;  
Statement2;  
StateMent3;
```

소스 B

```
Statement1;  
Statement2;  
StateMent3;
```

이렇게 작성하면 어떠한 단점이 생길까? (좋은 프로그램 작성 방법에서 어느 부분이 위반될까요?)네, 먼저 소스 코드 부분이 길어진다. 또 하나는 수정이 생기는 경우 각각 수정을 하기 때문에 유지 보수 면에서 공수가 많이 든다. 그래서 자주 사용하는 공통의 코드부분을 모듈화 하는데 이를 루틴 또는 서브루틴 이라고 부른다.

[언어 트랙 - 기초] 델파이 언어 기초2

•프로시저와 함수의 차이점

루틴은 반환 값(리턴)이 있는 루틴과 반환 값이 없는 루틴으로 분리되는데 델파이에서는 전자를 Function(함수), 후자를 Procedure(프로시저)라고 부른다. 선언 시에 이 규칙이 지켜지지 않을 경우 컴파일 오류가 발생한다. 참고로 C, C++, 자바의 경우는 Function, Void Function 이라고 부른다.

• 프로시저와 함수 선언 및 코드구현방법

선언(Definetion)은 컴파일러에게 루틴이름 매개변수의 정보 등을 알려주는 작업이다.

• 선언방법

Procedure 프로시저 이름(매개변수이름:매개변수타입, 매개변수이름:매개변수타입..);
Function 함수이름(매개변수이름:매개변수타입, 매개변수이름:매개변수타입....):리턴타입

[언어 트랙 - 기초] 델파이 언어 기초2

- ❖ 함수와 프로시저는 0개 이상의 매개변수 목록을 가진다.
- ❖ 여러 개의 매개변수 사용 시 세미콜론(;) 으로 구분하고 같은 자료형 사용 시 쉼표로 구분한다.
- ❖ Interface 부분은 자기 유닛의 선언 아래 부분은 물론 외부 유닛에서도 컴파일 하기 전에 바인딩 하여(uses하여) 사용 할 수 있는 영역이다. 주로 전역변수, 서브루틴들을 선언하는 곳이다.
- ❖ Implementation은 원래 인터페이스 선언된 루틴들을 구현하는 곳인데 이 영역에도 함수나 프로시저를 선언 할 수 있다. 단 여기에 선언된 루틴들은 자기 유닛 선언 이후 부분에서만 사용할 수 있고 외부에서도 사용할 수 없다.

• 구현방법

```
procedure 프로시저이름(매개변수이름: 매개변수타입, 매개변수이름: 매개변수타입, ...);  
{ 선언부 : 구현부에서 사용할 변수, 상수, 타입 지정 }  
begin  
    { 구현부 }  
end;
```

```
function 함수이름(매개변수이름:매개변수타입, 매개변수이름:매개변수타입...): 반환값타입;  
begin  
    { 구현부 }  
  
    Result := 반환값;  
end;
```

[언어 트랙 - 기초] 델파이 언어 기초2

• 매개변수란(Parameter) 무엇이고 왜 필요한가?

퇴직금을 계산하는 루틴을 작성한다고 가정해 보겠다. 퇴직금은 해당 사원의 급여에 근속연수를 곱한다고 정하겠다. 그럼 루틴을 수행하기 위해 어떠한 정보가 최소한 필요합니까? 네 해당직원의 입사일자, 급여정보가 필요하다. 또는 그 직원의 해당 정보를 읽을 수 있는 사원코드가 필요하다. 이를 해당 루틴의 매개변수라고 부른다. 즉 매개변수는 변수의 특별한 한 종류로서, 프로시저, 함수 등과 같은 서브루틴의 인풋으로 제공되는 여러 데이터 중 하나를 가리키기 위해 사용된다. 여기서 서브루틴의 인풋으로 제공되는 여러 데이터들을 전달인자(argument) 라고 부른다. 보통 매개변수의 목록은 서브루틴의 정의 부분에 포함되며, 매번 서브루틴이 호출될 때 마다 해당 호출에서 사용된 전달인자들을 각각에 해당하는 매개변수에 대입시켜 준다.

• 매개변수의 전달방식

함수 호출 시 인자 전달 방법에는 Call-by-value(값에 의한 호출)와 Call-by-reference(참조에 의한 호출)이 있다 먼저 이 둘을 비교하며 설명한 후, 델파이의 인자 전달 방식에 대해서도 설명하겠습니다.

- Call-By-Value (값에 의한 호출)
- Call-By-Reference (참조에 의한 호출)

[언어 트랙 - 기초] 델파이 언어 기초2

- 사용 예

- **Call By Value**

Function Add(x,y:integer):integer

- **Call By Const**

Procedure ShowMessage(const Msg:string)

- **Call By Reference**

procedure TForm28.FormCloseQuery(Sender: TObject; var CanClose: Boolean);

- 매개변수 디폴트 값 지정방법

매개변수 선언 시 기본값 지정이 가능하다. 맨 뒤의 매개변수부터 연속적으로 기본값 지정 가능

[언어 트랙 - 기초] 델파이 언어 기초2

같은 같은 이름의 변수나 루틴들은 같은 유니트내에 선언 할 수 가 없다. 프로시저, 함수 종류가 다른 경우에도 같은 이름을 사용하지 못한다.

이러한 문제를 오버로드 함수를 통해 해결해 보도록 하겠다.

오버로드란 같은 이름의 함수를 여러 개 정의하고, 매개변수의 유형과 개수를 다르게 하여 다양한 유형의 호출에 응답하게 하는 방법이다.

- 오버로드 함수 선언

```
function Divide(x,y:integer):integer; overload;  
function Divide(x,y:real):extended; overload;
```

- 오버로드 함수 호출

```
procedure TForm1.Button7Click(Sender: TObject);  
begin  
  Button7.Caption := IntToStr(Divide(12,4));  
end;
```

```
procedure TForm1.Button7Click(Sender: TObject);  
begin  
  Button7.Caption := FloatToStr(Divide(12.0,4.0));  
end;
```

[언어 트랙 - 기초] 델파이 언어 기초2

[실습]

- 두 개의 정수를 더하는 함수 Add를 선언하고 구현한다.
- 두개의 정수를 더하는 함수 Sub를 선언하고 구현한다.
- 폼에 데이터 박스를 2개 놓고 입력한 수를 이용하여 위에서 작성한 함수 Add,Sub를 호출하여 그 결과값을 아래와 같이 TLabel 컴포넌트에 표시한다.



- Add 함수의 매개변수 값을 디폴트로 2와 3으로 지정하여 호출하여 보자
- 이번에는 실수형 2개를 더하는 Add함수를 overload를 이용하여 작성해 보자