과제 #4. declaration\_specifiers & declarator

소프트웨어학부 20180325 임성현

Declaration\_specifiers와 declarator은 program – translation\_unit – external\_declaration – declaration. 즉, declaration을 구성한다.

Declaration은 declaration\_specifiers와 init\_declarator\_list로 구성된다. Declaration\_specifiers는 type\_specifier 또는 storage\_class\_specifier 로 구성되며 뒤에 연달아 다른 decalartion\_specifiers가 올 수 있다.

Type\_specifier는 크게 struct\_specifier 또는 enum\_specifier 또는 TYPE\_IDENTIFIER로 구성된다.

Struct\_specifier는 struct\_or\_union IDENTIFIER {struct\_declaration\_list}로 구성된다.

i.e. **struct s{ int x, y; struct a; int a: 16;};**

1. struct안에 struct을 선언이 가능하다.

2. 단, 선언은 가능하나 **int x = 0;** 같은 초기화가 불가능하다.

3. struct\_declaration\_list가 반복되는 구성이다. 즉, declaration\_specifiers가 반복되지 않기에 **static int x;** 처럼 storage\_class\_specifier가 올 수 없다.

Enum\_specifier는 enum IDENTIFIER {enumerator\_list}로 구성된다.

i.e. **enum e{one, two, three, four = 4};**

1. struct와 달리 값의 **four = 4;** 같은 값의 초기화가 가능하다. 단, **int x; enum {one = x;}** 처럼 semantic 관점에서 constant 값이 아니면 컴파일 에러가 발생한다.

2. struct와 달리 선언을 한 뒤에 참조가 가능하다.

TYPE\_IDENTIFIER는 integer\_type\_specifier 또는 floating\_point\_type\_specifier 또는 void\_type\_specifer 또는 typedef\_name 으로 구성된다.

i.e. **int x; float y; void func(); typedef struct s{…} my\_s;**

storage\_class\_specifier는 auto 또는 static 또는 typedef 로 구성 된다.

init\_declarator\_list는 init\_declarator로 구성된 init\_declarator\_list가 연달아 오는 형태이다. Init\_declarator는 declarator로 구성되며 initializer를 통해 값을 초기화 할 수 있다. Declarator는 direct\_declarator로 구성되며 pointer가 존재할 수 있다. Direct\_declarator는 IDENTIFIER 또는 (declarator) 또는 direct\_declarator뒤에 (parameter\_type\_list\_opt) 또는 [constant\_expression\_opt] 가 올 수 있다.

i.e. **int \*p; int a[10]; float \*f(); float (\*fun) ();**

1. 변수의 경우 return type, size에 유의해야 한다.

2. 함수의 경우 return type, body, parameter에 유의해야 한다.

다음 예시를 통해 내용을 정리해보자.

i.e. **int \* (\* x) (int, …);**

첫번째 parameter가 int고, 그 이후 부터 생략가능하며 return type이 int를 가리키는 포인터를 가진 함수를 가리키는 포인터 x이며 각 구성별로 분해하면 다음과 같다.

int : declaration\_specifers – type\_spcifier – TYPE\_IDENTIFIER - int

\* : init\_declarator\_list – init\_declarator – pointer - \*

( \* x ) : init\_declarator\_list -init\_declarator – direct\_declarator – pointer와 direct\_declartor로 구성된 (declarator)

(int , …); init\_declarator\_list – init\_declarator – declarator – direct\_declarator – (parameter\_type\_list\_opt)

Parameter\_type\_list는 parameter\_declaration이 연달아 올 수 있으며, 이는 declaration\_specifier 뒤에 declarator 또는 abstract\_declarator 아니면 그 자체로만 구성된다. 정리하면 declaration\_specifers init\_declarator\_list 형태가 parameter에서도 반복된다. Abstract\_declarator의 경우 이름을 생략할 수 있다.

i.e. **int fun(int a[], float b(int, …), double (\*\* )[10])**

또한, 이와 같은 선언들을 볼 때, 프로그래머는 선언들이 의미하는 바가 무엇인지 정확히 파악해야 한다.