Compiler Assignment #3

(Declaration Specifiers

and

Declarator)

소프트웨어학부

컴파일러

유재우 교수님

20201758

이창준

2025.03.23

**1. Declaration Specifiers**

declaration specifiers는 1. type\_specifiers 2. storage\_class\_specifiers 3. type\_qualifier로 각각 유도될 수 있고, 추가로 위 요소들이 혼자만 나타는게 아니라 같이 4. type\_specifiers declaration\_specifiers 5. storage\_class\_specifier declaration\_specifiers 6. type\_qualifier declaration\_specifiers로 추가적인 declaration\_specifiers가 나타날 수 있다. 그리고 이제 storage\_class\_specifier는 1. auto 2. static 3. typedef 4. register 5. extern과 같은 예약어들로 유도 될 수 있다. 이 문법대로라면, auto static와 같은 구성도 문법적으로는 옳은 것이다. 하지만 의미적으로는 틀린 것이므로 이는 semantic analysis 과정에서 걸러진다. static은 프로그램 시작 시 생성되고, 프로그램이 종료되면 사라지도록 하는 예약어이며, 이와 반대로 auto는 scope 내에서만 유효하고 scope를 벗어나면 사라지도록 한다. type\_qualifier같은 경우에는 1. const 2. volatile 둘 중 하나로 유도 된다. const는 해당 변수를 상수화하여 다른 값으로 변경하지 못하도록 한다. type\_specifier는 1. struct\_specifier 2. enum\_specifier 3. TYPE\_IDENTIFIER로 유도되며 TYPE\_IDENTIFIER는 int, float과 같은 예약어로 설정된 data type이나, 사용자가 typedef를 통해 정의한 data type을 뜻한다. struct\_specifier는 1. struct\_or\_union IDENTIFIER { struct\_declaration\_list } 2. struct\_or\_union { struct\_declaration\_list } 3. struct\_or\_union IDENTIFIER로 유도될 수 있다. struct\_or\_union은 1. struct 2. union 으로 유도할 수 있다. struct\_declaration\_list는 1. struct\_declaration 2. struct\_declaration\_list struct\_declaration으로 유도될 수 있다. struct\_declaration\_list는 간단히 말해서 struct나 union의 구성 변수들의 모임이라고 말할 수 있다. struct\_declaration은 1. specifier\_qualifier\_list struct\_declarator\_list로 유도될 수 있다. specifier\_qualifier\_list는 1. type\_specifier 2.type\_qualifier 3. type\_specifier specifier\_qualifier\_list 4. type\_qualifier specifier\_qualifier\_list로 유도 될 수 있다. 유도 가능한 구성이 declaration specifiers와 비슷한데, declaration specifiers와 달리 storage\_class\_specifiers가 제외 된 것이 특징이다. struct\_declarator\_list는 1. struct\_declarator 2. struct\_declarator\_list, struct\_declarator로 유도될 수 있다. 2번 규칙 덕분에 int a, b, c 와 같이 한 번에 여러 변수를 선언할 수 있다. struct\_declarator는 1. declarator 2. constant\_expression 3. declarator : constatne\_expression 으로 유도될 수 있다. constant\_expression은 %10 과 같이 사용하며 해당 변수의 bit 수를 제한하여 사용하기 위해서 사용한다. 그리고 = initializer가 없는 모습을 볼 수 있는데, 이는 구조체 정의부에서 초기화가 불가능한 것을 의미한다. enum\_speicifier는 1. enum IDENTIFIER { enumerator\_list } 2. enum { enumerator\_list } 3. enum IDENTIFIER로 유도될 수 있다. enumerator\_list는 1. enumerator 2. enumerator\_list, enumerator로 유도될 수 있으며 enumerator는 1. IDENTIFER 2. IDENTIFER = constant\_expression으로 유도될 수 있다. constant\_expression은 상수로 이루어진 expression을 뜻한다.

2. Declarator

declarator는 1. pointer direct\_declarator 2. direct\_declarator로 유도될 수 있다. pointer는 1. \*type\_qualifier 2. \* type\_qualifier pointer로 유도될 수 있다. direct\_declarator는 1. IDENTIFIER 2. ( declarator ) 3. direct\_declaratr [ constant\_expression\_opt ] 4. direct\_declarator ( parameter\_type\_list\_opt )로 유도될 수 있다. constant\_expression\_opt는1. empty 2. constant\_expression으로, parameter\_type\_list\_opt는 1. empty 2. parameter\_type\_list로 유도될 수 있다.