선언, 블록, 영역

- 선언
  - 변수에 이름과 데이터 타입 등의 속성을 부여하는 문장
  - 선언의 종류

선언	설명
명시적 선언	선언문을 사용하여 변수 이름을 나열하고 이들에 속성을 부여하는 방법
묵시적 선언	실행 시간 중에 일어나거나 프로그램 실행 과정에서 변경되는 바인딩

    - 명시적 선언
      - 변수 x의 데이터 타입을 예약어 int를 사용하여 명시적으로 지정
      - 예) int x;
    - 묵시적 선언
      - FORTRAN은 선언문 없이 변수 이름을 그냥 사용하면 그 이름이 선언된 것으로 간주
      - 변수 이름이 'I', 'J', 'K', 'L', 'M', 'N'으로 시작되면 정수 타입으로, 그렇지 않으면 실수 타입으로 선언

선언, 블록, 영역

- C/C++에서는 선언과 정의를 구분
  - 선언 : 부분적인 속성을 바인딩

```
int max (int num1, int num2) ;
```

  - 정의 : 모든 잠재적인 속성(코드, 제어문 등)을 바인딩

```
int max(int num1, int num2)
{
  if (num1 > num2)
        return num1;
  retrun num2
}
```

선언, 블록, 영역

- 블록
  - 일련의 문장 집합으로 자체적인 선언을 가질 수 있는 프로그램 단편
  - ALGOL 60의 블록
    - 임의의 블록 내의 선언: 지역적(local)
    - 블록 밖의 선언: 비지역적(nonlocal)

```
a: begin integer i, j;
     b: begin real x, y;
          ⋮
        end b;
     ⋮
   end a;
```

선언, 블록, 영역

- Ada에서의 블록 표현
  - declare로 시작하는 begin~ end

```
declare
  선언문
begin
  문장들
end;
```

  - 선언문이 없는 경우, declare 생략하여 표현

```
begin
  문장들;
end;
```

선언, 블록, 영역

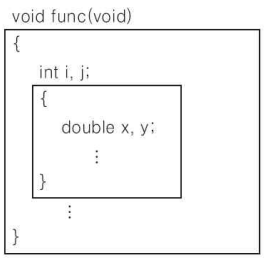
- 블록을 사용하는 Ada 예제

```
01  with TEXT_IO;
02  use TEXT_IO;
03  procedure block is
04      package INT_IO is new TEXT_IO.INTEGER_IO (integer);
05      use INT_IO;
06  begin
07      put("before block");
08      declare
09          X: integer := 1;
10      begin
11          put("block");
12          put(X);
13      end;
14      put("behind block");
15 end block;
```

선언, 블록, 영역

- C 기반 언어의 블록
  - 복합문이라 함
  - 중괄호로 묶어 표현

  - 예) 두 개의 블록 구조
    - func 본체를 나타내는 블록
    - func 본체 안에 내포된 블록



선언, 블록, 영역

- 영역
  - 이름의 사용이 허락되고 있는 범위
    - 예) 변수 x의 영역은 선언된 지점부터 func 함수의 끝까지

```
void func(void)
{
    int x;
     :        x의 영역
}
```

  - 정적 영역
    - 이름에 해당하는 변수를 찾을 때 자신을 포함하고 있는 블록에서 선언되었는지 살펴본다.
    - 아니면 그 바깥쪽 블록에서 선언되었는지를 살펴본다.
    - 임의의 블록에서 변수 x에 대한 참조가 이루어졌을 때 변수 x의 선언문을 찾는 과정
      1. 임의의 블록에 속한 선언문에 x에 대한 선언문 찾기
      2. 없으면, 임의의 블록을 포함하는 바깥쪽 블록의 선언문에서 x에 대한 선언문 찾기
      3. x에 대한 선언문을 찾을 때까지 계속됨
      4. 가장 큰 블록의 선언문에서도 x를 발견하지 못하면 선언되지 않은 변수 오류 발생

선언, 블록, 영역

- 정적 영역을 위한 ALGOL 예제

```
01  a: begin integer i, j; real x, y;
02      b: begin integer x, y;
                  ...
03                i := x;
04                j := y;
05      end b;
06      c: begin real i, j;
                  ...
07                x := i;
08                y := j;
09      end c;
                  ...
10  end a;
```

블록 a : 블록 b의 i, j에 대해 선언
블록 c의 x, y에 대해 선언

블록 b : 변수 x, y는 2행에 선언
변수 i,j는 1행에 선언

블록 c : 변수 i, j는 6행에 선언
변수 x, y는 1행에 선언

선언, 블록, 영역

- 정적 영역을 위한 ALGOL 예제

```
01 a: begin integer i, j; real x, y;
02     b: begin integer x, y;
                ...
03             i := x;
04             j := y;
05     end b;
06     c: begin real i, j;
                ...
07             x := i;
08             y := j;
09     end c;
                ...
10 end a;
```

블록 a : 블록 b의 i, j에 대해 선언
블록 c의 x, y에 대해 선언

블록 b : 변수 x, y는 2행에 선언
변수 i,j는 1행에 선언

블록 c : 변수 i, j는 6행에 선언
변수 x, y는 1행에 선언

선언, 블록, 영역

- 정적 영역을 위한 C++ 예제

```
01  #include <iostream>
02  using std::cout;
03  using std::endl;
04
05  int x;
06
07  void func(void)
08  {
09      double x;
10      x = 3.14;
11      cout << x << endl;
12  }
13
14  int main(void)
15  {
16      x = 20;
17      cout << x << endl;
18      func();
19
20      return 0;
21  }
```

전역변수

지역변수

선언, 블록, 영역

- 영역 지정 연산자를 사용하는 C++ 예제

```
01  #include <iostream>
02  using std::cout;
03  using std::endl;
04
05  int x;
06
07  void func(void)
08  {
09      double x;
10      x = 3.14;
11      ::x = 30;
12      cout << x << endl;
13  }
14
15  int main(void)
16  {
17      x = 20;
18      cout << x << endl;
19      func();
20      cout << x << endl;
21
22      return 0;
23  }
```

영역 지정 연산자를 사용하여 전역 변수인 5행의 x를 나타냄

선언, 블록, 영역

- 오류가 발생하는 C++ 예제

```
01  #include <iostream>
02  using std::cout;
03   using std::endl;
04
05  int main(void)
06  {
07      {
08          int x;
09      }
10      x = 30;
11      cout << x << endl;
12
13      return 0;
14  }
```

접근 불가능

선언, 블록, 영역

- 동적 영역
  - 이름에 해당하는 변수를 찾을 때 외향적인 구조에 기반하지 않고, 부프로그램들의 호출 순서에 기반
    - 먼저 자신을 포함하고 있는 블록에서 선언되었는지를 보고
    - 아니면 자신을 포함한 블록(부프로그램)을 호출한 문장을 포함하고 있는 블록에서 선언된 것인지를 조사한다.
    - 해당 변수를 찾을 때까지 반복
  - LISP 예

```
01  -> (defvar s 10)
02  -> (defun f (x) (+ x s))
03  -> (defun g (s) (f (+ s 11)))
04  -> (g 5)
```