9 장 보충 자료 – Scope and Storage Class

■ **scope: 변수, 함수, goto label 등의 명칭이 사용될 수 있는 프로그램 텍스트의 영역; 공간적 개념**

- **global** (전역): 전체 프로그램; nonstatic external variable, nonstatic(global) function 전역 변수, 전역 함수 /(static이 없는)
- **file** (파일): 해당 소스 파일; external static variable, static function 전역 static 변수, static 함수
- **function** (함수): 해당 함수; goto label
- **block** (블록): 해당 블록; function parameter, local variable(auto/register variable, local static variable)
  (parameter: register 선언 가능, auto 선언은 허용되지 않음)

■ **storage class: 변수, 동적 메모리 등이 기억 공간을 차지하고 있는 생존시간(lifetime) ; 시간적 개념**

storage class	생존시간	초기화 시점	default 초깃값	초기화 수식	저장 위치
static (정적)	프로그램 시작 → 끝	프로그램 시작 시 단 한 번만 초기화	*zero*	상수 수식만 가능	static area
automatic (자동) **register** (레지스터)	해당 블록 시작 → 끝	해당 블록 진입 시마다 매번 다시 초기화	undefined	일반 수식 가능 (함수 호출 포함)	runtime stack *or* register
dynamic (동적)	malloc/calloc → free	calloc 호출 시	malloc: undefined calloc: *zero*	-	heap

■ **storage class & scope**

- **static**
  - nonstatic external variable → global
  - external static variable → file
  - local static variable → block
- **automatic, register**
  - function parameter, auto/register variable → block

```
int e;                  // e: nonstatic external variable → static, global
static int es;          // es: external static variable → static, file
static int fs() { … }           // fs: static function → file
void f (int p, register int pr) {   // f: nonstatic(global) function → global
        // p: parameter → automatic, block
        // pr: parameter → register, block
  auto double a;        // a: local auto variable → automatic, block
  register int r;       // r: local register variable → register, block
  static int s;         // s: local static variable → static, block
  …
  L:    // L: goto label → function
  …
}

// static s;      →  static int s;
// auto a;        →  auto int a;
// register r;    →  register int r;
```