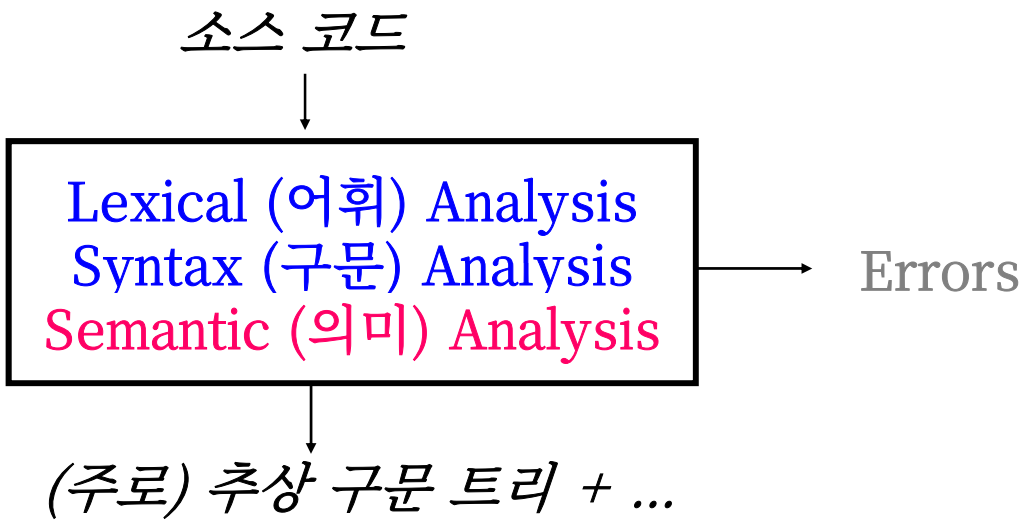
Compiler (컴파일러)
Semantic Analysis Ⅱ
-Scopes and Symbol Tables

2007년 2학기
충남대학교 전기정보통신공학부
조은선

Semantic Analysis



- 프로그램 constructs (변수, 객체, 식, 문장...) 의 올바른 사용 여부를 분석, 확인
- Scope 관련 : 변수가 선언되기 전에 쓰였나? 두 번 정의 됐나?
- Type 관련 : 변수와 assign되는 값과 타입이 맞는가?

Scope

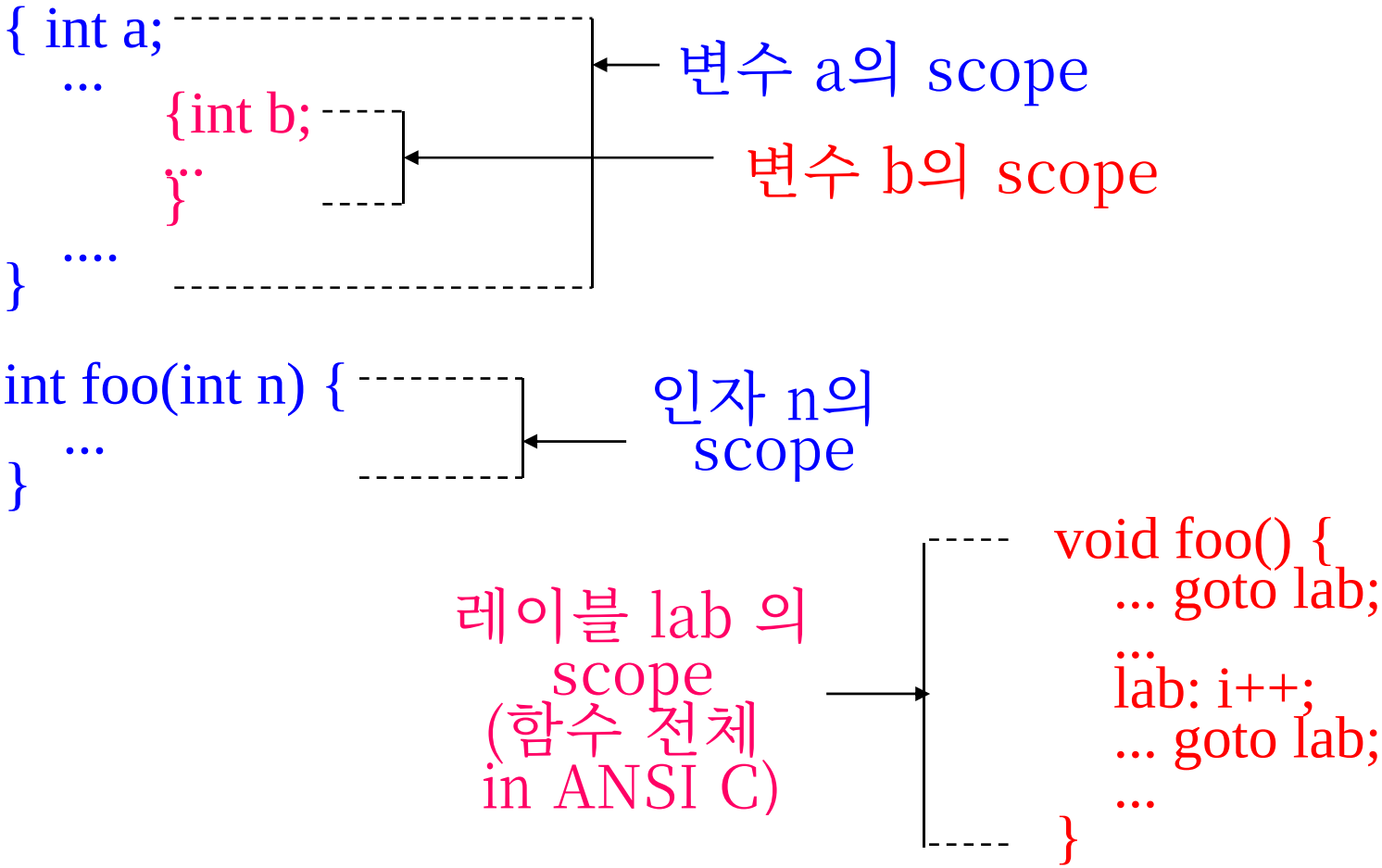
- 식별자 (Identifiers)
  - 변수, 상수, 함수이름, lables ...
- Lexical Scope
  - 프로그램을 문자적으로(textual) 볼 때 특정 범위
    - 문장 block, 형식인자 리스트, 함수 정의문, 소스 파일, 프로그램 전체...
- 식별자의 scope
  - 그 선언이 참조되는 lexical scope

  예) 변수의 scope:

  문장 block 내 (local 변수), 함수 내 (형식 인자), 소스 파일 (global 변수), 전체 프로그램 (extern 변수)

  cf. How about fields? methods?

변수 Scope : PL 복습



Semantic Rules for Scopes

- Main rules:
  - Rule 1: 각 식별자는 자신의 scope 에서 한번만 선언되어야함
  - Rule 2: 같은 종류의 식별자를 한 lexical scope 내에서 두 번이상 선언하면 안됨

```
class X {
    int X;
    void X(int X) {
        X: ...
        goto X;
    }
}
```

```
int X(int X) {
    int X;
    goto X;
    {
        int X;
        X: X = 1;
    }
}
```

문제가 있을까?
있다면,
어디서 발생하나?

Symbol Tables

- Symbol tables
  - 심벌들에 관한 정보를 관리하는데 사용되는 자료구조
  - 의미 분석단계에서 식별자의 성질 (scope, type)을 참조하거나 코드 생성할 때 주로 사용 됨
  - 선언될 때 정보 수집해서 테이블에 삽입(insert)
  - 사용되는 부분에서 테이블 참조 (get)
- 테이블 entry : 식별자 이름 + info
  - 예)

NAME	KIND	TYPE	ATTRIBUTES
foo	func	int,int → int	extern
m	arg	int	
n	arg	int	const
tmp	var	char	const

Scope Information in Symbol Tables

```
int x;
void f(int m) {
    float x, y;
    ...
    {int i, j; ....; }
    {int x; l: ...; }
}

int g(int n) {
    char t;
    ... ;
}
```

- Block structured 언어 특징
  - 각 block (lexical scope) 내에는 local 변수 선언들 존재

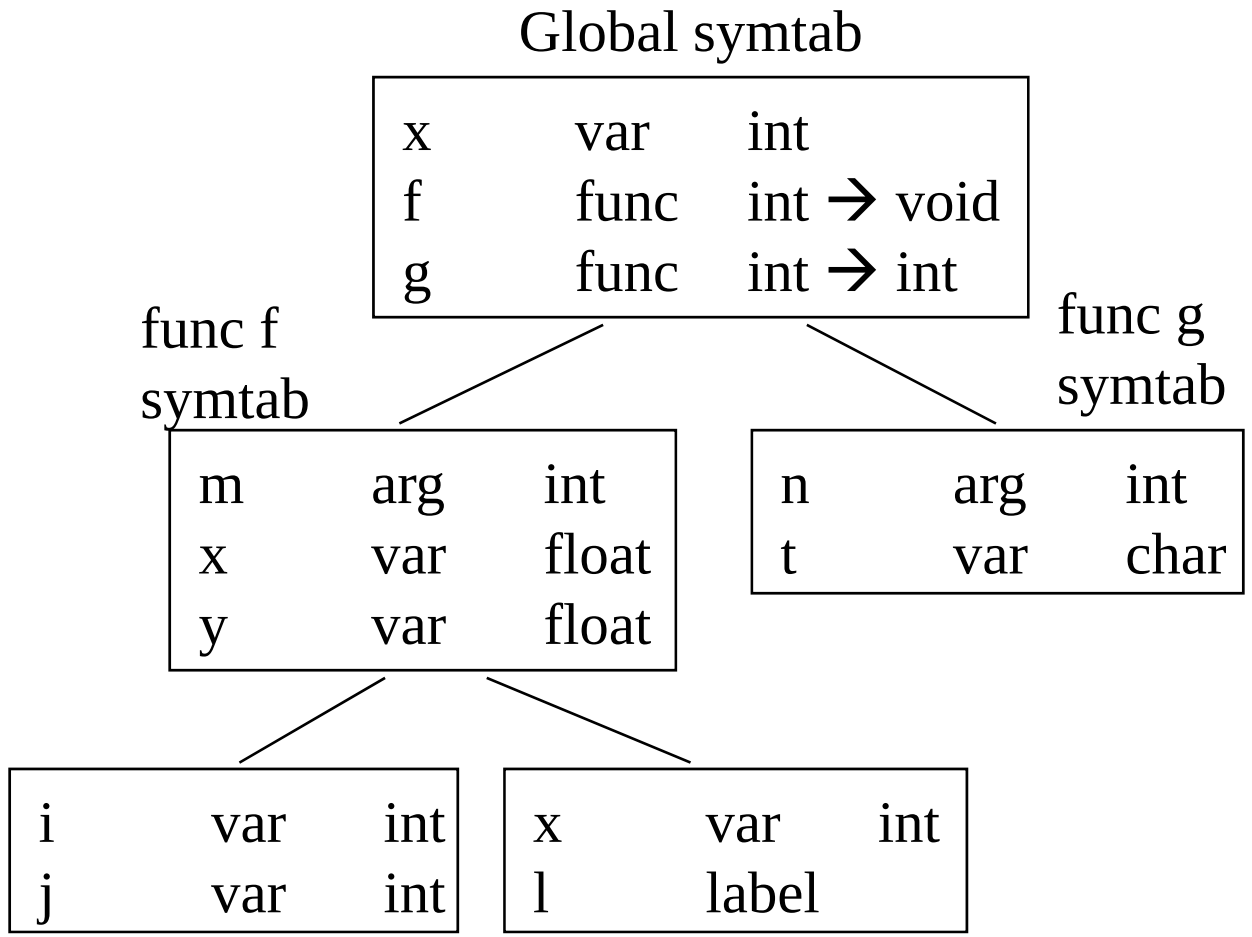
➔ 한 lexical 마다 한 symbol table

  - Hierarchy of scopes :
    - 각 block (lexical scope)은 다른 subblock을 포함가능하고,
    - 자기 block 에서 선언되지 않고, 자신을 포함하는 block 에서 선언된 변수 사용 가능

➔ Hierarchy of symbol tables

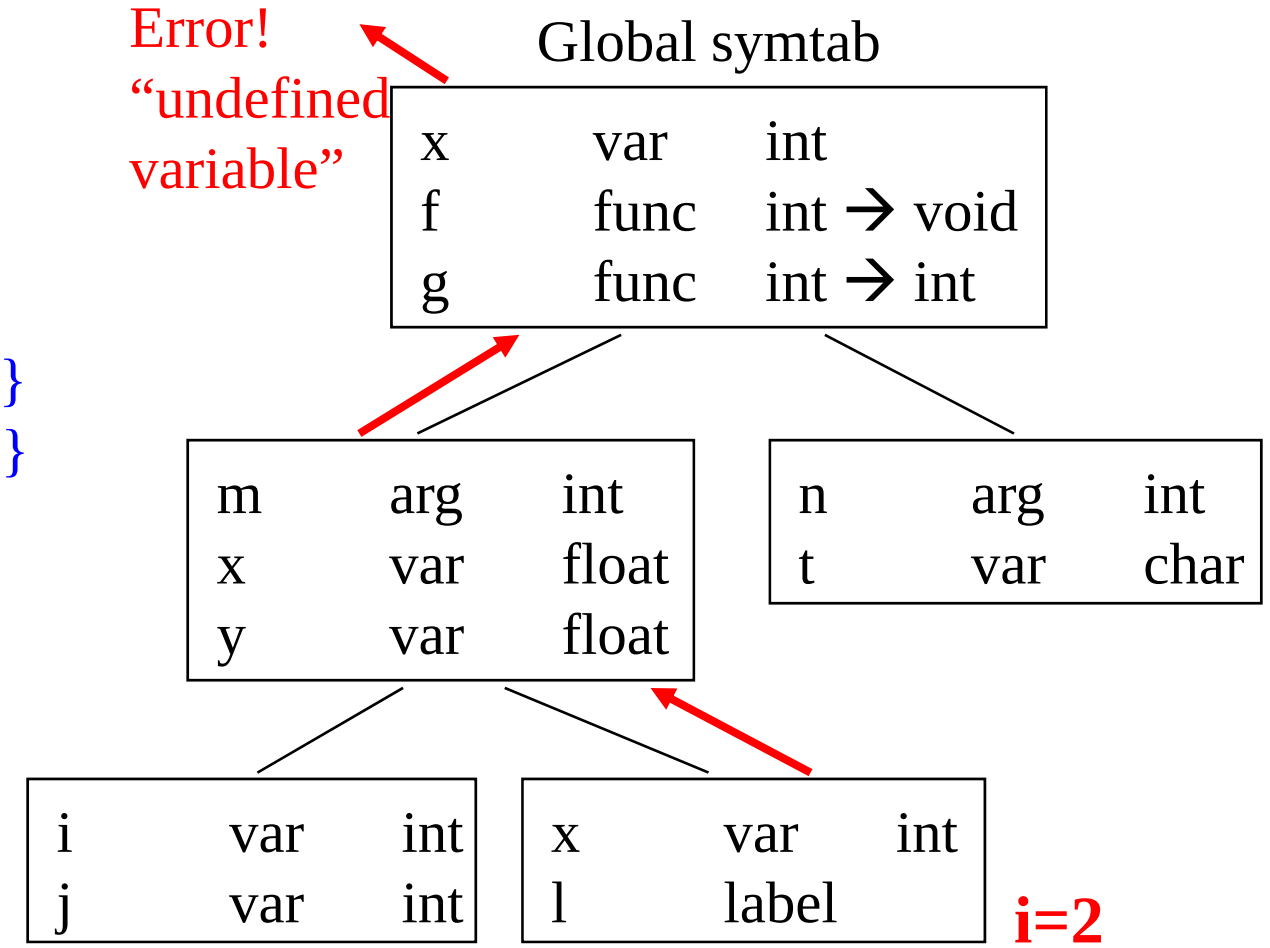
Examples

```
int x;

void f(int m) {
    float x, y;
    ...
    {int i, j; ....; }
    {int x; l: ...; }
}

int g(int n) {
    char t;
    ... ;
}
```



오류 Checking

```
int x;
void f(int m) {
    float x, y;
    ...
    {int i, j; x=1; }
    {int x; l: i=2; }
}

int g(int n) {
    char t;
    x=3;
}
```

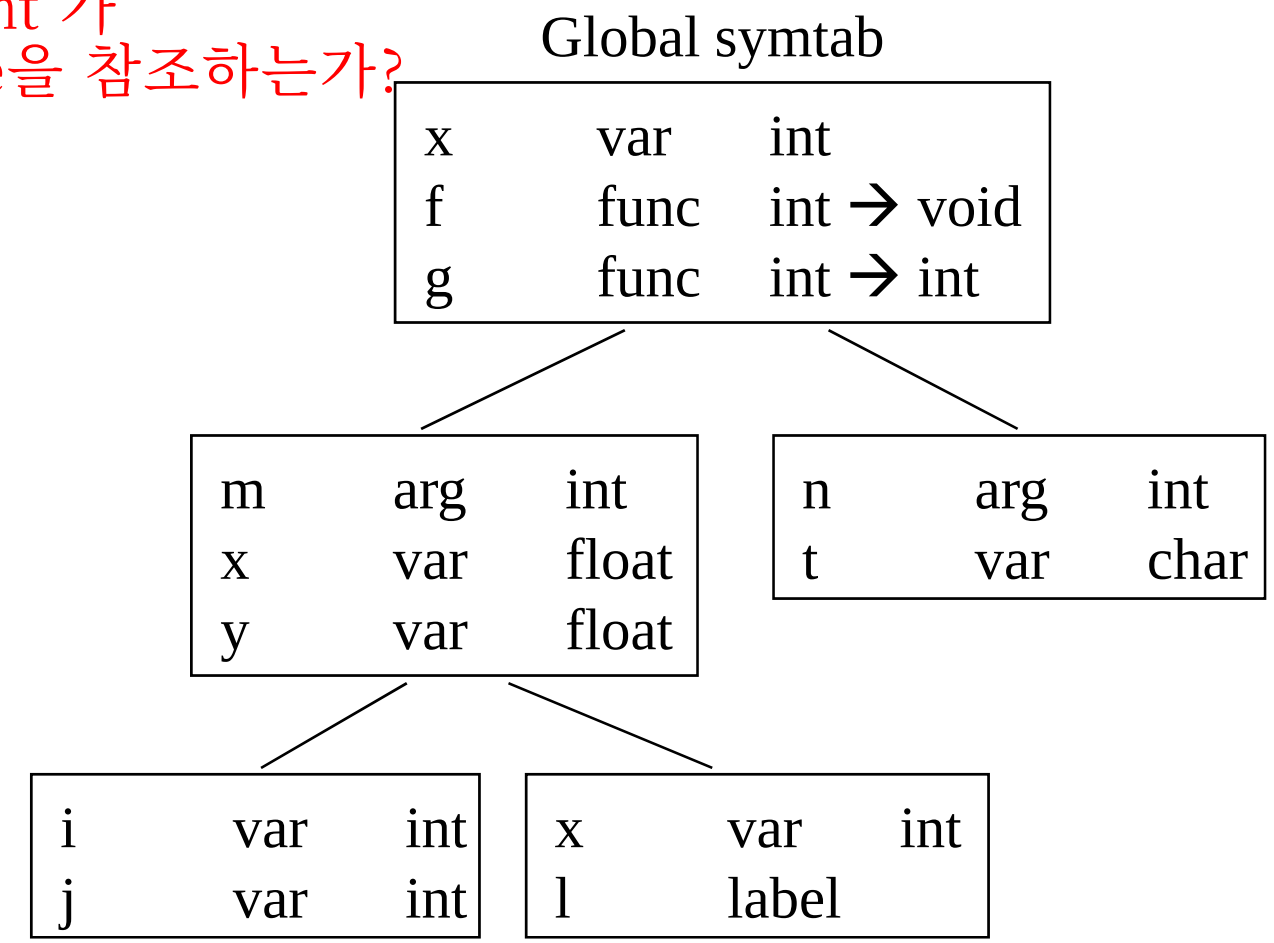


- 현재 scope 에서 시작해서 hierarchy 위쪽으로 올라가면서 찾음
- 끝까지 없으면 에러

Class Problem

각 x 의 assignment 가
어느 symbol table을 참조하는가?

```
int x;
void f(int m) {
    float x, y;
    ...
    {int i, j; x=1; }
    {int x; l: x=2; }
}

int g(int n) {
    char t;
    x=3;
}
```



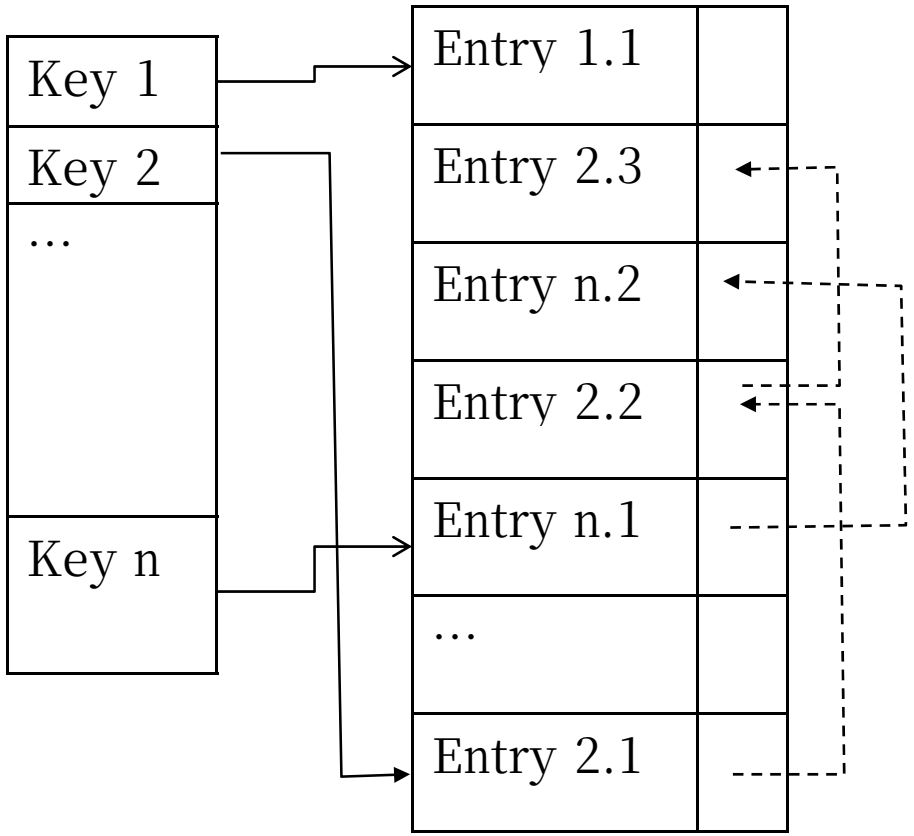
Symbol Table Implementation

- 필요한 operation
  - 테이블 구성 : AST 까지 만들어져야 가능
    삽입 (insertion) : 해당 식별자가 선언될 때
  - 검색 (lookup) : 해당 식별자가 사용될 때 (checking)

    cf. forward reference 는?
- Efficiency를 위해
  - 테이블 entry에서 식별자 이름
    - 이름들의 스트링 값을 따로 보관하는 스트링 풀로의 포인터
  - Local 테이블 구조 : hash 테이블 사용
  - Global하게는 N-ary tree 구조 :
    - 트리를 쓰지는 않는다. : 포인터 유지등의 비용 때문
    - 사용의 locality : scope을 빠져 나가면 해당 local table은 필요가 없어짐

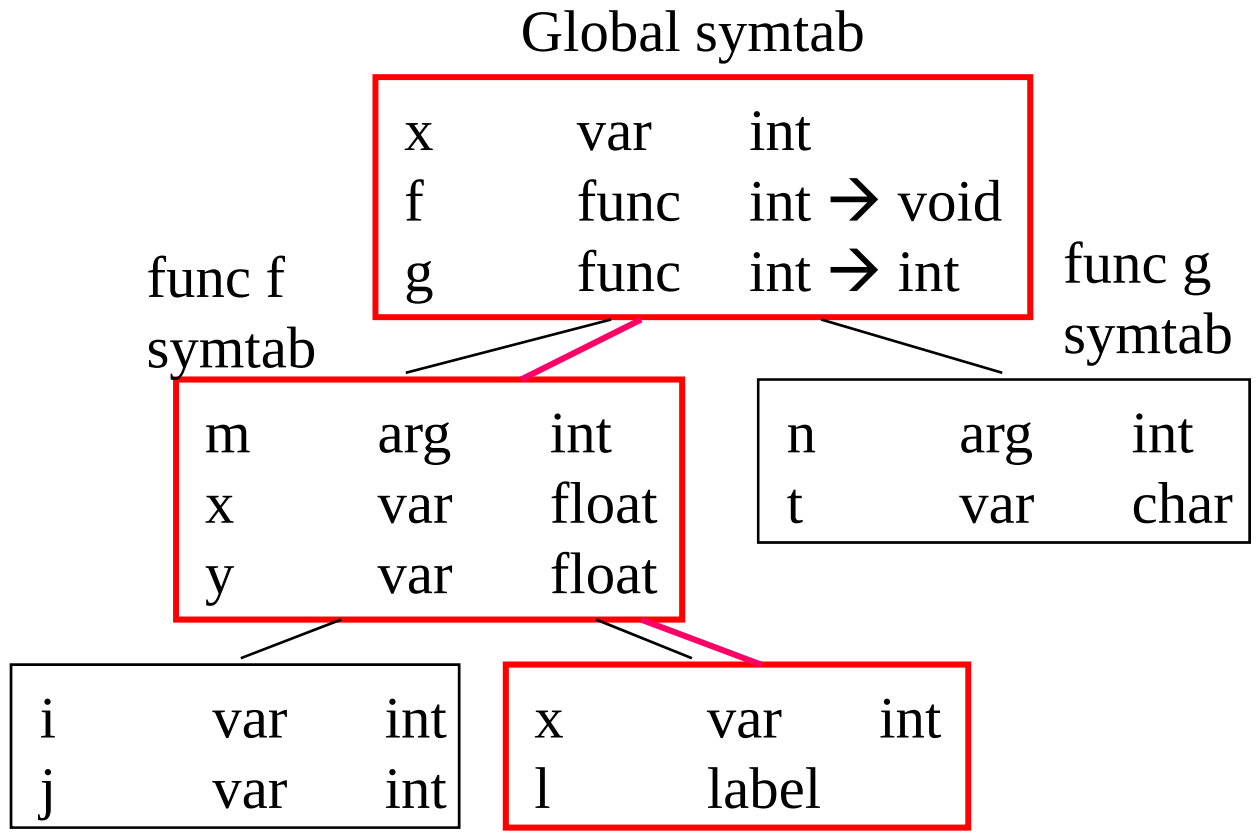
Local Table

- Local Hash Table 만 생각하면...

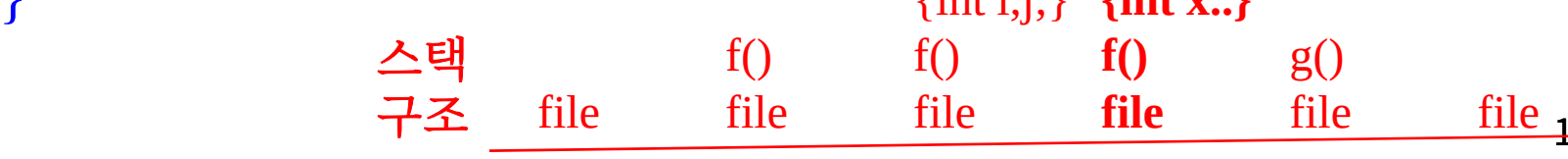


Global Table Hierarchy-스택을 사용

```
int x;

void f(int m) {
    float x, y;
    ...
    {int i, j; ....; }
    {int x; l: ...; }
}

int g(int n) {
    char t;
    ... ;
}
```



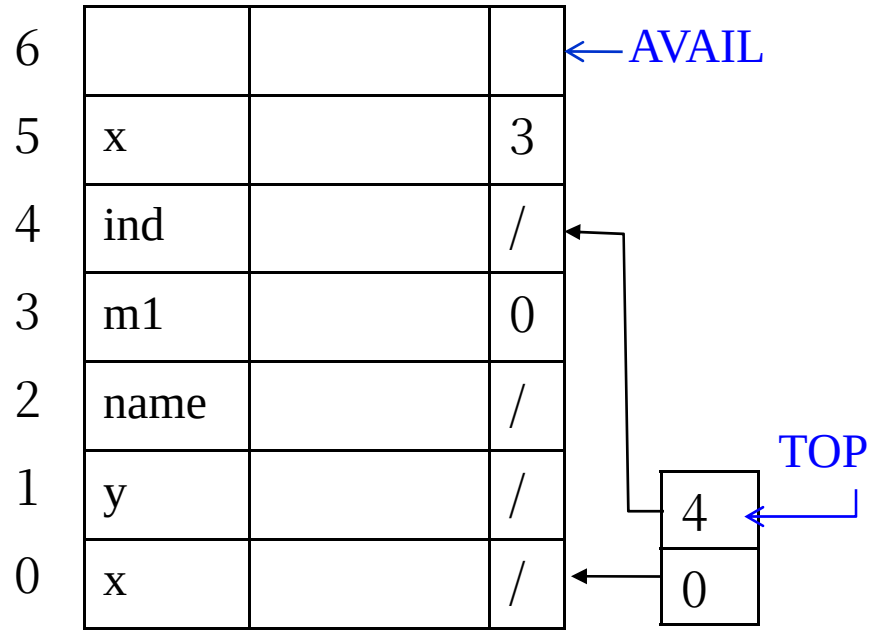
스택 구조



Hierarchies of Local Tables

```
int x,y;
char name;
void m1(int ind) {
    int x;
}
void m2 (int j) {
  {
     int f[j];
     char test;
   }
}
```

0	/
1	5
2	2
3	/
4	4
5	/
6	/
7	1
8	/
9	/
10	/



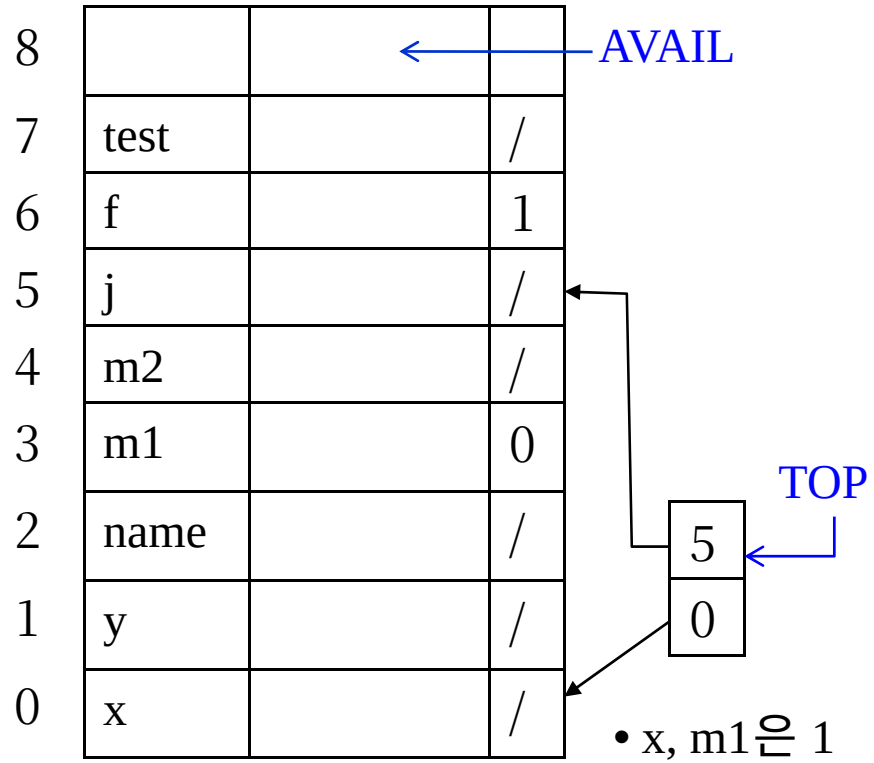
- x, m1은 1
- name은 2
- ind, j는 4
- test는 6
- f, y는 7
- m2는 10

"스택과 해쉬의 공존"

Hierarchies of Local Tables (cont ’)

```
int x,y;
char name;
void m1(int ind) {
    int x;
}
void m2 (int j) {
  {
     int f[j];
     char test;
   }
}
```

0	/
1	3
2	2
3	/
4	5
5	/
6	7
7	6
8	/
9	/
10	4



- x, m1은 1
- name은 2
- ind, j는 4
- test는 6
- f, y는 7
- m2는 10