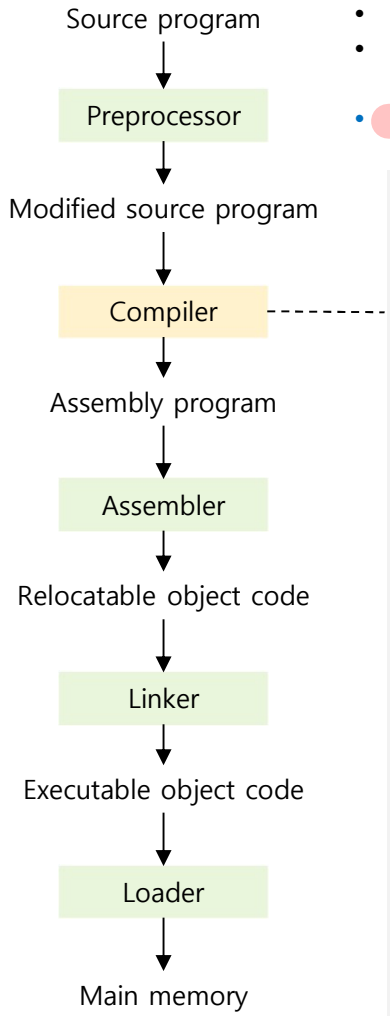
개요

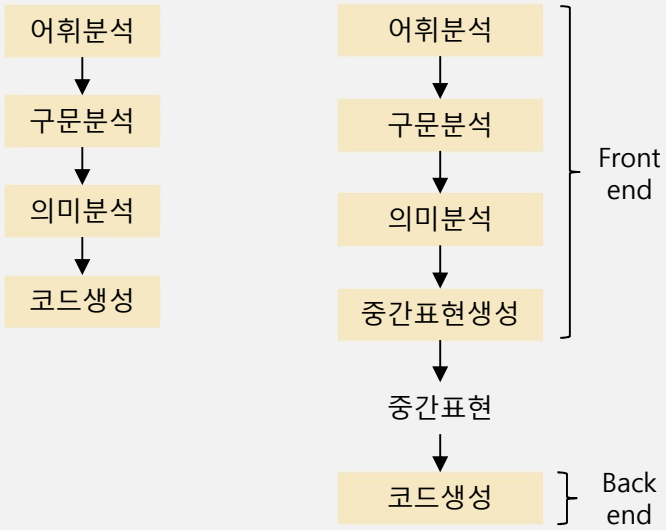
목차

- Scope, symbol table
- Semantic analysis

Compiler 개요



- 어휘분석 → 소스프로그램을 문자 나열로 읽어 들여 의미 있는 어휘 단위인 토큰들로 분할
- 구문분석 → 소스프로그램이 프로그래밍언어의 문법에 맞게 작성되었는지 검사(소스프로그램이 언어의 문법에서 유도되는지 검사하고, 소스프로그램의 유도 트리를 추상구문트리로 변환)
- **의미분석 → 소스프로그램이 의미적으로 오류가 없는지 검사(type checking, name binding 등)**

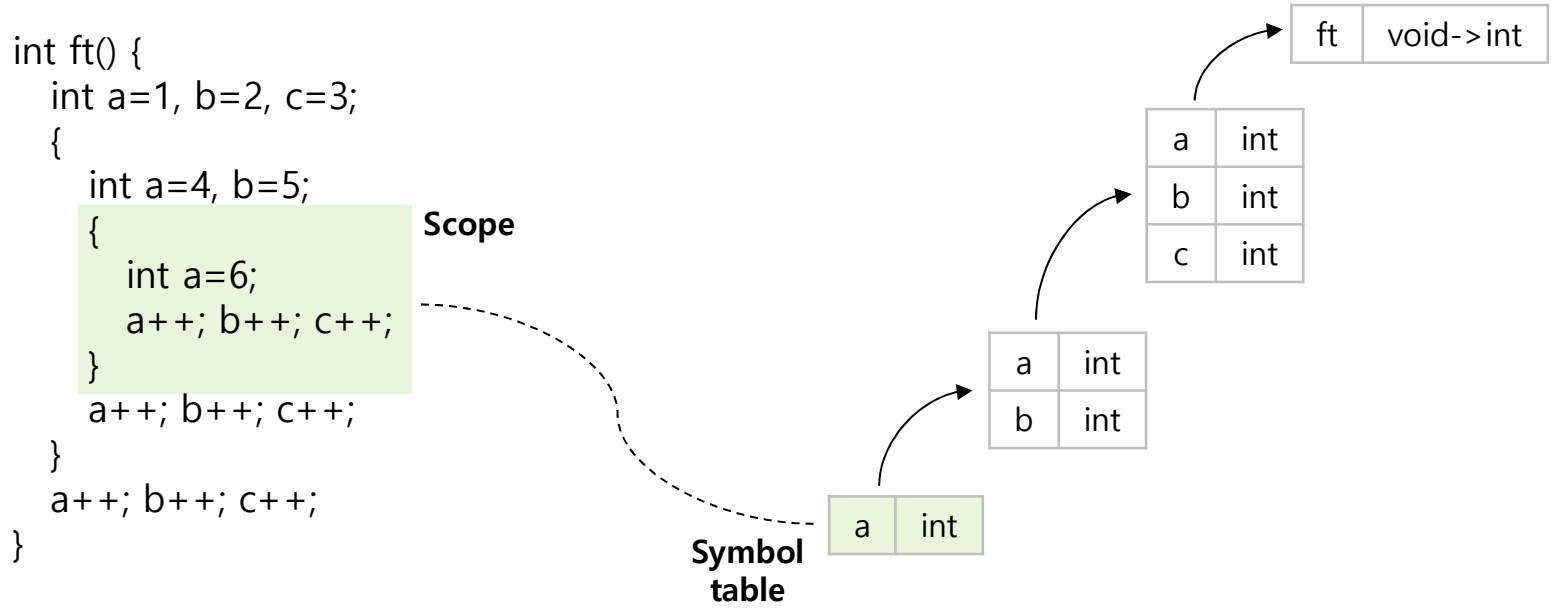


- n개 서로 다른 CPU(예: x86, ARM, SPARK)에서 동작하는 m개 서로 다른 프로그래밍언어의 컴파일러들을 모두 제작한다고 가정할 때,
- 일체형 컴파일러로 제작한다면 $m \times n$개 컴파일러 제작이 필요하지만,
- 중간표현을 도입하여 front-end와 back-end를 분리하면 총 m개 front-end와 n개 back-end 제작으로 충분

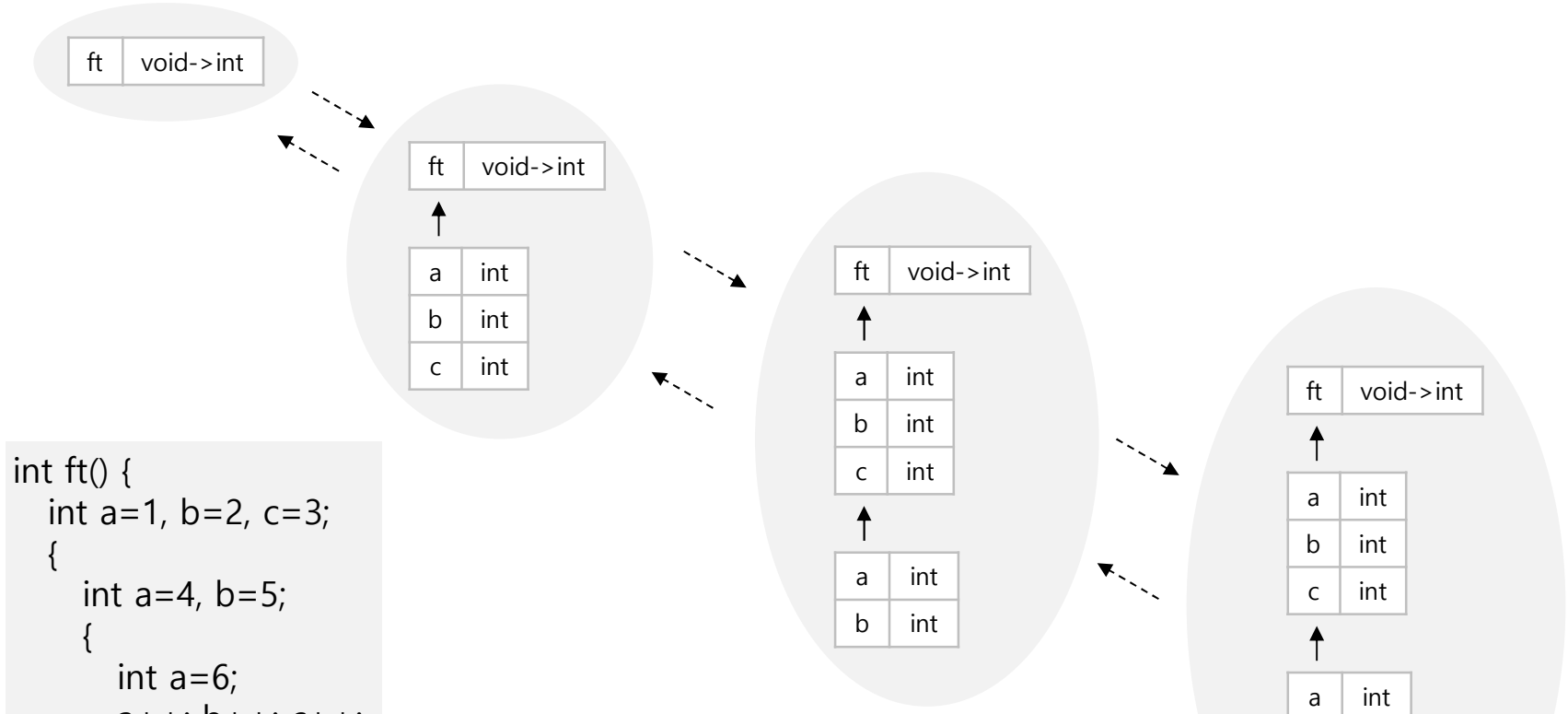
Scope, symbol table (1/2)

Scope, symbol table

- Scope은 하나 이상의 선언(declaration)이 적용되는 코드 영역
- 식별자의 scope은 해당 식별자의 특정 선언의 scope
- Scope이 다르다면, 코드 내 서로 다른 여러 개체(예: 변수) 참조 위해 동일 식별자 사용 가능
- Scope은 심볼테이블을 통해 구현 가능 → 서로 다른 scope에 대해 별도의 심볼테이블을 구성

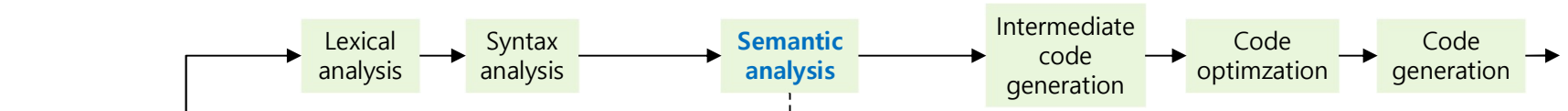


Scope, symbol table (2/2)




```
int ft() {
   int a=1, b=2, c=3;
   {
      int a=4, b=5;
      {
         int a=6;
         a++; b++; c++;
      }
      a++; b++; c++;
   }
   a++; b++; c++;
}
```

Semantic analysis (1/3)




```
...
int v=1;
int ft(){
    v++;
    return v;
}
int main(){
    printf("%d₩n", v);
    int v=10;
    printf("%d₩n", v+ft());
    for(int i=1;i<=5;i++){
        int v=100;
        printf("%d₩n",v+ft());
    }
    printf("%d₩n", v);
    return 0;
}
```

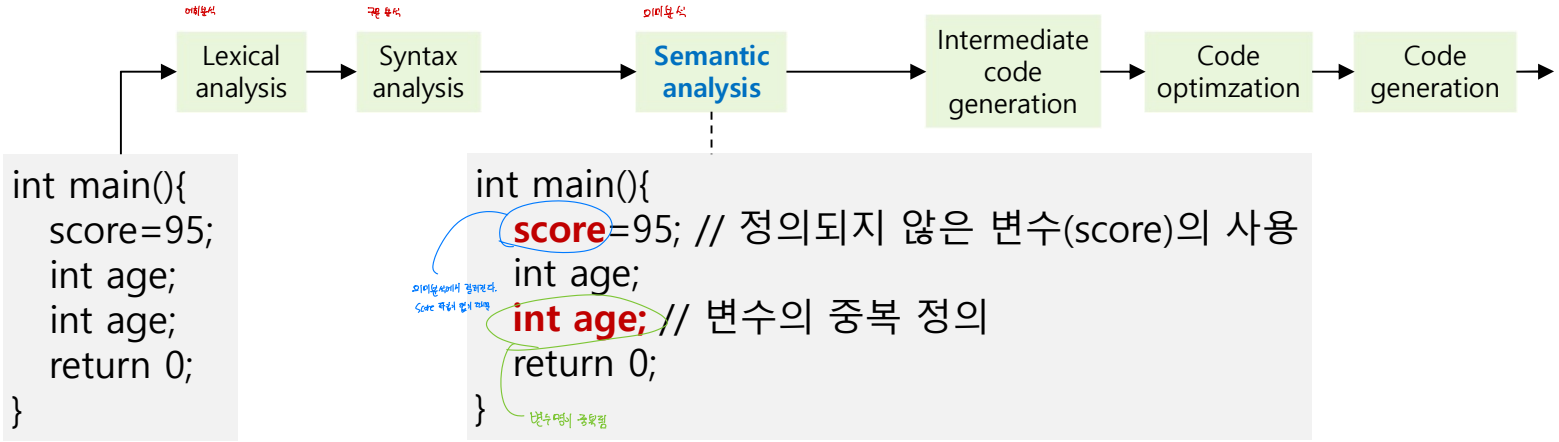
```
...
int v=1;
int ft(){
    v++;
    return v;
}
int main(){
    printf("%d₩n", v);
    int v=10;
    printf("%d₩n", v+ft());
    for(int i=1;i<=5;i++){
        int v=100;
        printf("%d₩n",v+ft());
    }
    printf("%d₩n", v);
    return 0;
}
```

```
1
12
103
104
105
106
107
10
```

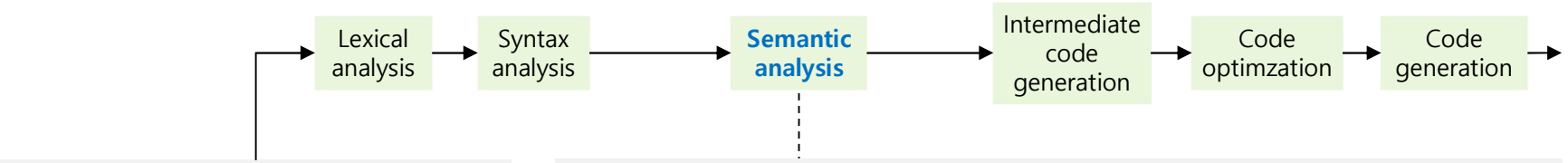
Name resolution

- 식별자(예: 변수명)의 사용과 선언(정의)을 연결하는 것
- 대부분 PL에서는, 변수의 scope이 구별된다면, 동명 변수를 허용

Semantic analysis (2/3)



Semantic analysis (3/3)




```
public class Test {
    public static void main(String[] args) {
        int x;
        int y;
        y=x;
        int y;
        x=z;
        boolean b=true;
        x=y+b;
        if(x) y=100;
        b=!y;
        b=x && y;
        b=x>="1";
        int i1=1, i2=2, i3;
        double d1=1.0, d2=2.0, d3;
        i3=i1+i2;
        d3=d1+d2;
        String s="Korea"+i1;
        d3=d1+i2;
    }
}
```

```
public class Test {
    public static void main(String[] args) {
        int x;
        int y;
        y=x;                    // 초기화되지 않은 변수(x)의 사용
        int y;                  // 변수의 중복 정의
        x=z;                    // 정의되지 않은 변수(z)의 사용
        boolean b=true;
        x=y+b;                  // int와 boolean에 덧셈 연산 적용 미정의
        if(x) y=100;            // 조건식이 boolean이 아님
        b=!y;                   // int에 연산자 !의 적용 미정의
        b=x && y;               // int, int에 연산자 &&의 적용 미정의
        b=x>="1";               // int와 String에 관계연산자 >=의 적용 미정의
        int i1=1, i2=2, i3;
        double d1=1.0, d2=2.0, d3;
        i3=i1+i2;               // 연산자 +는 int, int의 덧셈(x86: add)
        d3=d1+d2;               // 연산자 +는 double, double의 덧셈(x86: fadd, ...)
        String s="Korea"+i1;    // 연산자 +는 문자열 간 연결(concatenation)
        d3=d1+i2;               // i2는 int에서 double로 암묵적 형변환(type coercion)
    }
}
```

- **위 자바 코드의 많은 의미적 오류는 type checking(타입검사)을 통해 발견됨**

References

- 박두순. (2016). (내공 있는 프로그래머로 길러주는)컴파일러의 이해. 한빛아카데미.
- 창병모. (2021). 프로그래밍 언어론 : 원리와 실제. 인피니티북스.
- Aho, A., Lam, M., Sethi, R., Ullman, J. (2006). Compilers: Principles, Techniques, and Tools (2nd Ed.). Addison Wesley.
- Nystrom, R. Crafting interpreter. https://craftinginterpreters.com/
- Sebesta, R. (2012). Concepts of Programming Languages (10th. ed.). Pearson.
- Thain, D. (2023). Introduction to Compilers and Language Design.
- Paxson, V. (1995). Flex, version 2.5.
- Donnelly, C., Stallman, R. (2008). Bison.
- LexAndYacc.pdf (epaperpress.com)
- Tom Niemann, LEX & YACC. http://epaperpress.com/lexandyacc

References

- 박두순. (2016). (내공 있는 프로그래머로 길러주는)컴파일러의 이해. 한빛아카데미.
- 창병모. (2021). 프로그래밍 언어론 : 원리와 실제. 인피니티북스.
- Aho, A., Lam, M., Sethi, R., Ullman, J. (2006). Compilers: Principles, Techniques, and Tools (2nd Ed.). Addison Wesley.
- Nystrom, R. Crafting interpreter. https://craftinginterpreters.com/
- Sebesta, R. (2012). Concepts of Programming Languages (10th. ed.). Pearson.
- Thain, D. (2023). Introduction to Compilers and Language Design.
- Paxson, V. (1995). Flex, version 2.5.
- Donnelly, C., Stallman, R. (2008). Bison.
- LexAndYacc.pdf (epaperpress.com)
- Tom Niemann, LEX & YACC. http://epaperpress.com/lexandyacc