컴파일러 과제 3.2

Source Program 출력하기

제출마감일자 : 2016.11.14

담당교수 : 유재우 교수님
소속학부 : 컴퓨터학부
학번 : 20142577
이름 : 홍상원

1. 과제 개요

1) 목표

- lex 와 yacc 프로그램을 이해한다.
- syntax tree 를 이해한다.

2) 구현사항

- yacc 명세서에서 호출되는 함수들의 일부분을 구현한다. [syntax.c]
- lex 에 의해 생성된 lexical analyzer 와 yacc 에 의해 생성된 syntax analyzer 를 거쳐 만들어진 syntax tree 를 고려하여 역으로 원시 프로그램을 출력하는 프로그램을 작성한다. [print.c]

2. 구현결과

1) syntax.c

```
// set declarator type (or element type)
A_ID *setDeclaratorElementType(A_ID *id, A_TYPE *t)
{
	//구현한 부분 시작
	 A_TYPE *tt;
	 if(id->type == NIL) {//타입 정보가 없으면
		id->type = t;//바로 연결
	 }
	 else {//타입 정보가 있으면
		tt = id->type;
		while(tt->element_type)//마지막까지 가서
			tt = tt->element_type;
		tt->element_type = t;//마지막 엘리먼트 타입으로 연결
	 }
	//구현한 부분 끝

	 return (id);
}

// check function declarator and return type
A_ID *setFunctionDeclaratorSpecifier(A_ID *id, A_SPECIFIER *p)
{
	 A_ID *a;
	 // check storage class
	 if (p->stor)
		syntax_error(25);
	 setDefaultSpecifier(p);
	 // check check if there is a function identifier immediately before '('
	 //구현한 부분 시작
```

```
        if(id->type->kind != T_FUNC) {//T_FUNC 타입이 아니면 오류
                syntax_error(21);
                return (id);
        }
        else {{//T_FUNC 타입이면
                id = setDeclaratorElementType(id, p->type);//엘리먼트 타입 연결
                id->kind = ID_FUNC;//id 를 ID_FUNC 로 설정
        }
        //구현한 부분 끝
        // check redeclaration
        a=searchIdentifierAtCurrentLevel(id->name,id->prev);
        if (a) {
                if (a->kind!=ID_FUNC || a->type->expr)
                        syntax_error(12,id->name);
                else { // check prototype: parameters and return type
                //구현한 부분 시작
                        if(isNotSameFormalParameters(a->type->field, id->type-
>field))// 프로토타입에 선언된 파라미터와 다르면
                                syntax_error(22, id->name);//오류
                        if(isNotSameType(a->type->element_type, id->type-
>element_type))//프로토타입에 선언된 타입과 다르면
                                syntax_error(26, a->name);//오류
                //구현한 부분 끝
                }
        }
        // change parameter scope and check empty name
        a=id->type->field;
        while (a) {
                if (strlen(a->name))
                        current_id=a;
                else if (a->type)
                        syntax_error(23);
                a=a->link; }
        return(id);
}

// set declarator_list type and kind based on storage class
A_ID *setDeclaratorListSpecifier(A_ID *id, A_SPECIFIER *p)
{
        A_ID *a;
        setDefaultSpecifier(p);
        a=id;
```

```
	while (a) {
		if (strlen(a->name) && searchIdentifierAtCurrentLevel(a->name,a->prev))
			syntax_error(12,a->name);
		//구현한 부분 시작
		a = setDeclaratorElementType(a, p->type);// 엘리먼트 타입 연결
		if(p->stor == S_TYPEDEF)//typedef 인 경우
			a->kind = ID_TYPE;
		else if(a->type->kind == T_FUNC)//함수인 경우
			a->kind = ID_FUNC;
		else// 그 이외
			a->kind = ID_VAR;
		a->specifier=p->stor;

		if(a->specifier == S_NULL)//static, typedef 이외인 경우
			a->specifier = S_AUTO;
		a = a->link;
		//구현한 부분 끝
	}
	return(id);
}

// set declarator_list type and kind
A_ID *setParameterDeclaratorSpecifier(A_ID *id, A_SPECIFIER *p)
{
	// check redeclaration
	if (searchIdentifierAtCurrentLevel(id->name,id->prev))
		syntax_error(12,id->name);
	// check papameter storage class && void type
	if (p->stor || p->type==void_type)
		syntax_error(14);
	//구현한 부분 시작
	setDefaultSpecifier(p);//specifier 초기설정
	id = setDeclaratorElementType(id, p->type);//엘리먼트 타입 연결
	id->kind = ID_PARM;//파라미터 타입으로 설정
	//구현한 부분 끝
	return(id);
}



//구조체 필드 설정
A_ID *setStructDeclaratorListSpecifier(A_ID *id, A_TYPE *t)
```

```
{
	A_ID *a;
	a=id;
	while (a) {
		//구현한 부분 시작
		//같은 레벨인지 확인하여 중복선언여부 검사
		if(searchIdentifierAtCurrentLevel(a->name, a->prev))
			syntax_error(12, a->name);
		a = setDeclaratorElementType(a, t);//엘리먼트 타입 연결
		a->kind = ID_FIELD;//구조체 필드 타입으로 설정
		a = a->link;//구조체 필드 연결
		//구현한 부분 끝
	}
	return(id);
}
```

2) print.c

```
#include <stdio.h>
#include <string.h>
#include "type.h"
char * node_name[] = {
	"N_NULL",
	"N_PROGRAM",
	"N_EXP_IDENT",
	"N_EXP_INT_CONST",
	"N_EXP_FLOAT_CONST",
	"N_EXP_CHAR_CONST",
	"N_EXP_STRING_LITERAL",
	"N_EXP_ARRAY",
	"N_EXP_FUNCTION_CALL",
	"N_EXP_STRUCT",
	"N_EXP_ARROW",
	"N_EXP_POST_INC",
	"N_EXP_POST_DEC",
	"N_EXP_PRE_INC",
	"N_EXP_PRE_DEC",
	"N_EXP_AMP",
	"N_EXP_STAR",
	"N_EXP_NOT",
	"N_EXP_PLUS",
	"N_EXP_MINUS",
```

```
        "N_EXP_SIZE_EXP",
        "N_EXP_SIZE_TYPE",
        "N_EXP_CAST",
        "N_EXP_MUL",
        "N_EXP_DIV",
        "N_EXP_MOD",
        "N_EXP_ADD",
        "N_EXP_SUB",
        "N_EXP_LSS",
        "N_EXP_GTR",
        "N_EXP_LEQ",
        "N_EXP_GEQ",
        "N_EXP_NEQ",
        "N_EXP_EQL",
        "N_EXP_AND",
        "N_EXP_OR",
        "N_EXP_ASSIGN",
        "N_ARG_LIST",
        "N_ARG_LIST_NIL",
        "N_STMT_LABEL_CASE",
        "N_STMT_LABEL_DEFAULT",
        "N_STMT_COMPOUND",
        "N_STMT_EMPTY",
        "N_STMT_EXPRESSION",
        "N_STMT_IF",
        "N_STMT_IF_ELSE",
        "N_STMT_SWITCH",
        "N_STMT_WHILE",
        "N_STMT_DO",
        "N_STMT_FOR",
        "N_STMT_RETURN",
        "N_STMT_CONTINUE",
        "N_STMT_BREAK",
        "N_FOR_EXP",
        "N_STMT_LIST",
        "N_STMT_LIST_NIL",
        "N_INIT_LIST",
        "N_INIT_LIST_ONE",
        "N_INIT_LIST_NIL"};
void print_ast(A_NODE *);
void prt_program(A_NODE *, int);
void prt_initializer(A_NODE *, int);
```

```
void prt_arg_expr_list(A_NODE *, int);
void prt_statement(A_NODE *, int);
void prt_statement_list(A_NODE *, int);
void prt_for_expression(A_NODE *, int);
void prt_expression(A_NODE *, int);
void prt_A_TYPE(A_TYPE *, int);
void prt_A_ID_LIST(A_ID *, int);
void prt_A_ID(A_ID *, int);
void prt_A_ID_NAME(A_ID *, int);
void prt_STRING(char *, int);
void prt_integer(int, int);
extern A_TYPE *int_type, *float_type, *char_type, *void_type, *string_type;

//원시 프로그램 출력을 시작할 때 호출하는 함수
void print_ast(A_NODE *node)
{
	prt_program(node,0);
}

//프로그램 함수
void prt_program(A_NODE *node, int s)
{
	switch(node->name) {
		case N_PROGRAM:
			prt_A_ID_LIST(node->clink, s+1);
			break;
	}
}
//초기화 함수
void prt_initializer(A_NODE *node, int s)
{
	switch(node->name) {
		case N_INIT_LIST:
			if(node->llink->name == N_INIT_LIST) {// 배열, 구조체 등 초기화 시 중괄호
삽입
				printf("{");
			}

			prt_initializer(node->llink, s+1);

			if(node->rlink->name != N_INIT_LIST_NIL)//여러 개의 초기화 값들이
존재하면 콤마로 구분
```

```
                        printf(",");
                prt_initializer(node->rlink, s+1);

                if(node->rlink->name == N_INIT_LIST_NIL)//배열, 구조체 등의 초기화 종료
                        printf("}");
                break;
        case N_INIT_LIST_ONE:
                prt_expression(node->clink, s+1);
                break;
        case N_INIT_LIST_NIL:
                break;
    }
}

//수식함수
void prt_expression(A_NODE *node, int s)
{
    switch(node->name) {
        case N_EXP_IDENT :
                prt_A_ID_NAME(node->clink, s+1);
                break;
        case N_EXP_INT_CONST :
                prt_integer(node->clink, s+1);
                break;
        case N_EXP_FLOAT_CONST :
                prt_STRING(node->clink, s+1);
                break;
        case N_EXP_CHAR_CONST :
                prt_integer(node->clink, s+1);
                break;
        case N_EXP_STRING_LITERAL :
                prt_STRING(node->clink, s+1);
                break;
        case N_EXP_ARRAY :
                prt_expression(node->llink, s+1);
                printf("[");
                prt_expression(node->rlink, s+1);
                printf("]");
                break;
        case N_EXP_FUNCTION_CALL :
                prt_expression(node->llink, s+1);
                        printf("(");
```

```
            prt_arg_expr_list(node->rlink, s+1);
                    printf(")");
            break;
    case N_EXP_STRUCT :
            prt_expression(node->llink, s+1);
            printf(".");
            prt_STRING(node->rlink, s+1);
            break;
    case N_EXP_ARROW :
            prt_expression(node->llink, s+1);
            printf("->");
            prt_STRING(node->rlink, s+1);
            break;
    case N_EXP_POST_INC :
            prt_expression(node->clink, s+1);
            printf("++");
            break;
    case N_EXP_POST_DEC :
            prt_expression(node->clink, s+1);
            printf("--");
            break;
    case N_EXP_PRE_INC :
            printf("++");
            prt_expression(node->clink, s+1);
            break;
    case N_EXP_PRE_DEC :
            printf("--");
            prt_expression(node->clink, s+1);
            break;
    case N_EXP_AMP :
            printf("&");
            prt_expression(node->clink, s+1);
            break;
    case N_EXP_STAR :
            printf("*");
            prt_expression(node->clink, s+1);
            break;
    case N_EXP_NOT :
            printf("!");
            prt_expression(node->clink, s+1);
            break;
    case N_EXP_PLUS :
```

```
            printf("+");
            prt_expression(node->clink, s+1);
            break;
    case N_EXP_MINUS :
            printf("-");
            prt_expression(node->clink, s+1);
            break;
    case N_EXP_SIZE_EXP :
            printf("sizeof(");
            prt_expression(node->clink, s+1);
            printf(")");
            break;
    case N_EXP_SIZE_TYPE :
            printf("sizeof(");
            prt_A_TYPE(node->clink, s+1);
            printf(")");
            break;
    case N_EXP_CAST :
            printf("(");
            prt_A_TYPE(node->llink, s+1);
            printf(")(");
            prt_expression(node->rlink, s+1);
            printf(")");
            break;
    case N_EXP_MUL :
            prt_expression(node->llink, s+1);
            printf("*");
            prt_expression(node->rlink, s+1);
            break;
    case N_EXP_DIV :
            prt_expression(node->llink, s+1);
            printf("/");
            prt_expression(node->rlink, s+1);
            break;
    case N_EXP_MOD :
            prt_expression(node->llink, s+1);
            printf("%");
            prt_expression(node->rlink, s+1);
            break;
    case N_EXP_ADD :
            prt_expression(node->llink, s+1);
            printf("+");
```

```
            prt_expression(node->rlink, s+1);
            break;
    case N_EXP_SUB :
            prt_expression(node->llink, s+1);
            printf("-");
            prt_expression(node->rlink, s+1);
            break;
    case N_EXP_LSS :
            prt_expression(node->llink, s+1);
            printf("<");
            prt_expression(node->rlink, s+1);
            break;
    case N_EXP_GTR :
            prt_expression(node->llink, s+1);
            printf(">");
            prt_expression(node->rlink, s+1);
            break;
    case N_EXP_LEQ :
            prt_expression(node->llink, s+1);
            printf("<=");
            prt_expression(node->rlink, s+1);
            break;
    case N_EXP_GEQ :
            prt_expression(node->llink, s+1);
            printf(">=");
            prt_expression(node->rlink, s+1);
            break;
    case N_EXP_NEQ :
            prt_expression(node->llink, s+1);
            printf("!=");
            prt_expression(node->rlink, s+1);
            break;
    case N_EXP_EQL :
            prt_expression(node->llink, s+1);
            printf("==");
            prt_expression(node->rlink, s+1);
            break;
    case N_EXP_AND :
            prt_expression(node->llink, s+1);
            printf("&&");
            prt_expression(node->rlink, s+1);
            break;
```

```
		case N_EXP_OR :
			prt_expression(node->llink, s+1);
			printf("||");
			prt_expression(node->rlink, s+1);
			break;
		case N_EXP_ASSIGN :
			prt_expression(node->llink, s+1);
			printf("=");
			prt_expression(node->rlink, s+1);
			break;
	}
}
//파라미터 수식 함수
void prt_arg_expr_list(A_NODE *node, int s)
{
	switch(node->name) {
		case N_ARG_LIST :
			prt_expression(node->llink, s+1);
			if(node->rlink->name != N_ARG_LIST_NIL)
				printf(", ");
			prt_arg_expr_list(node->rlink, s+1);
			break;
		case N_ARG_LIST_NIL :
			break;
	}
}
//실행문 함수
void prt_statement(A_NODE *node, int s)
{
	switch(node->name) {
		case N_STMT_LABEL_CASE :
			printf("case ");
			prt_expression(node->llink, s+1);
			printf(":₩n");
			prt_statement(node->rlink, s+1);
			printf("₩n");
			break;
		case N_STMT_LABEL_DEFAULT :
			printf("default:₩n");
			prt_statement(node->clink, s+1);
			printf("₩n");
			break;
```

```
case N_STMT_COMPOUND:
        printf("{₩n");
        if(node->llink)
                prt_A_ID_LIST(node->llink, s+1);
                prt_statement_list(node->rlink, s+1);
                printf("}₩n");
        break;
case N_STMT_EMPTY:
        printf(";₩n");
        break;
case N_STMT_EXPRESSION:
        prt_expression(node->clink, s+1);
        printf(";₩n");
        break;
case N_STMT_IF_ELSE:
        printf("if(");
        prt_expression(node->llink, s+1);
        printf(")");
        prt_statement(node->clink, s+1);
        printf("else ");
        prt_statement(node->rlink, s+1);
        break;
case N_STMT_IF:
        printf("if(");
        prt_expression(node->llink, s+1);
        printf(")");
        prt_statement(node->rlink, s+1);
        break;
case N_STMT_SWITCH:
        printf("switch(");
        prt_expression(node->llink, s+1);
        printf(")");
        prt_statement(node->rlink, s+1);
        break;
case N_STMT_WHILE:
        printf("while(");
        prt_expression(node->llink, s+1);
        printf(")");
        prt_statement(node->rlink, s+1);
        break;
case N_STMT_DO:
        printf("do ");
```

```
                prt_statement(node->llink, s+1);
                printf("while(");
                prt_expression(node->rlink, s+1);
                printf(")");
                break;
        case N_STMT_FOR:
                printf("for(");
                prt_for_expression(node->llink, s+1);
                printf(")");
                prt_statement(node->rlink, s+1);
                break;
        case N_STMT_CONTINUE:
                printf("continue;₩n");
                break;
        case N_STMT_BREAK:
                printf("break;₩n");
                break;
        case N_STMT_RETURN:
                printf("return ");
                if(node->clink)
                        prt_expression(node->clink, s+1);
                printf(";₩n");
                break;
    }
}
//실행문 리스트 함수 : 여러 개 실행문 출력
void prt_statement_list(A_NODE *node, int s)
{
    switch(node->name) {
        case N_STMT_LIST:
                prt_statement(node->llink, s+1);
                prt_statement_list(node->rlink, s+1);
                break;
        case N_STMT_LIST_NIL:
                break;
    }
}
//for 문을 위한 수식문
void prt_for_expression(A_NODE *node, int s)
{
    switch(node->name) {
        case N_FOR_EXP :
```

```
                        if(node->llink)
                                prt_expression(node->llink, s+1);
                        printf(";");
                        if(node->clink)
                                prt_expression(node->clink, s+1);
                        printf(";");
                        if(node->rlink)
                                prt_expression(node->rlink, s+1);
                        break;
        }
}
//정수형 출력
void prt_integer(int a, int s)
{
        printf("%d", a);
}
//문자열 출력
void prt_STRING(char *str, int s) {
        printf("%s", str);
}

char
*type_kind_name[]={"NULL","ENUM","ARRAY","STRUCT","UNION","FUNC","POINTER","VOID"
};
//int declared = 0;//자기참조 구조체 출력시 무한 루프에 빠지는 것을 막기 위해 사용
//자료형 출력 함수
void prt_A_TYPE(A_TYPE *t, int s)
{
        if (t==int_type)
                printf("int");
        else if (t==float_type)
                printf("float");
        else if (t==char_type)
                        printf("char");
        else if (t==void_type)
                printf("void");
        else if (t->kind==T_NULL)
                printf("");
        else
                switch (t->kind) {
                        case T_ENUM:
                                t->prt=TRUE;
```

```
		printf("enum ");
		printf("{₩n");
		prt_A_ID_LIST(t->field,s+2);
		printf("}");
		break;
	case T_POINTER:
		t->prt=TRUE;
		prt_A_TYPE(t->element_type,s+2);
		printf("*");
		break;
	case T_ARRAY:
		t->prt=TRUE;
		prt_A_TYPE(t->element_type,s+2);
		printf("[");
		if (t->expr)
			prt_expression(t->expr,s+2);
		printf("]");
		break;
	case T_STRUCT:
		t->prt=TRUE;
		printf("struct ");
		/*
		// 자기참조 구조체 출력 시 무한루프를 방지하기 위해 사용
		if(declared != 0)
			break;
		*/
		printf("{₩n");
		//declared++;
		prt_A_ID_LIST(t->field,s+2);
		printf("}");
		break;
	case T_UNION:
		t->prt=TRUE;
		printf("union ");
		printf("{₩n");
		prt_A_ID_LIST(t->field,s+2);
		printf("₩n}");
		break;
	case T_FUNC:
		t->prt=TRUE;
		printf("(");
		prt_A_ID_LIST(t->field,s+2);
```

```
                                printf(") ");

                                if (t->expr) {
                                        prt_statement(t->expr,s+2);
                                }
                                prt_A_TYPE(t->element_type,s+2);

                        }
}

//A_ID 리스트 함수
void prt_A_ID_LIST(A_ID *id, int s)
{
        while (id) {
                prt_A_ID(id,s);
                if(id->link != NULL && id->kind == ID_PARM)//함수 파라미터가 여러 개라면
                        printf(", ");//콤마 출력
                if(id->kind == ID_PARM && id->type == 0)
                        printf("...");
                if(id->link != NULL && id->kind == ID_ENUM_LITERAL)//열거형 타입 명칭 상수가
여러 개라면
                        printf(", ");//콤마 출력
                if(id->kind == ID_FIELD)//구조체 필드멤버라면
                        printf(";₩n");//세미콜론으로 멤버 구분
                id=id->link;//다음 id 로 감
        }
}
char
*id_kind_name[]={"NULL","VAR","FUNC","PARM","FIELD","TYPE","ENUM","STRUCT","ENUM_
LITERAL"};
char *spec_name[]={"NULL","AUTO","TYPEDEF","STATIC"};
//A_ID 이름 출력함수
void prt_A_ID_NAME(A_ID *id, int s)
{
        printf("%s", id->name);
}
//A_ID 출력함수
void prt_A_ID(A_ID *id, int s)
{
        if(id->kind == ID_ENUM_LITERAL)//열거형 명칭상수 출력
                printf("%s", id->name);
        if (id->type) {
```

```
		printf("%s ", id->name);
		if(id->specifier == S_NULL) {

		}
		else if(id->specifier == S_AUTO) {

		}
		else if(id->specifier == S_STATIC) {
			printf("static ");
		}
		else if(id->specifier == S_TYPEDEF) {
			printf("typedef ");
		}
		prt_A_TYPE(id->type,s+2);
		if(id->specifier == S_TYPEDEF)
			printf(";₩n");
	}

	if (id->init) {//초기화
		printf("=");
		if (id->kind==ID_ENUM_LITERAL) {
			prt_expression(id->init,s+2);
		}
		else {
			if(id->kind == ID_ENUM ||
				id->kind == ID_STRUCT ||
				(id->kind == ID_VAR && id->type->kind == T_ARRAY))

				if(id->init->clink == NULL || id->init->clink->name !=
N_EXP_STRING_LITERAL)//배열, 구조체 초기화
					printf("{");
			prt_initializer(id->init,s+2);
		}
	}

		if(id->kind == ID_FUNC) {
			if(id->type->expr == NULL)
				printf(";");
			printf("₩n");
		}
		else if(id->kind == ID_PARM) {
```

```
                }
                else if(id->kind == ID_VAR) {
                        printf(";₩n");
                }
}
```

3. 실행결과

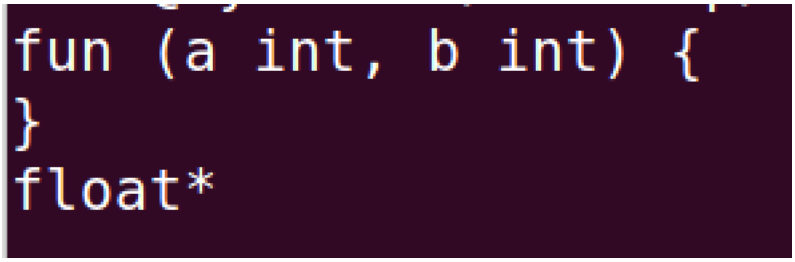
1) 함수 선언문 출력

- 입력 소스코드

```
float *fun(int a, int b) {

}
```

- 실행결과



2) 일반선언문 출력

- 입력 소스코드

```
static int *kim[10], j = 20+30;
int fun();
typedef struct {
        int val;
} STR;
```

- 실행결과

```
kim static int*[10];
j static int=20+30;
fun () int;
STR typedef struct {
val int;
};
```

3) 초기화 선언자 출력

- 입력 소스코드

```
int kim=10+20, lee[2][3]={{1,2,3},{4,5,6}};
int hong[]={10,11,12};
int park[3][2][4] =
{{{1,2,3,4},{5,6,7,8}},{{9,10,11,12},{13,14,15,16}},{{17,18,19,20},{21,22,23,24}}};
```

- 실행결과

```
kim int=10+20;
lee int[3][2]={{1,2,3},{4,5,6}};
hong int[]={10,11,12};
park int[4][2][3]={{{1,2,3,4},{5,6,7,8}},{{9,10,11,12},{13,14,15,16}},{{17,18,19,20},{21,22,23,24}}};
```

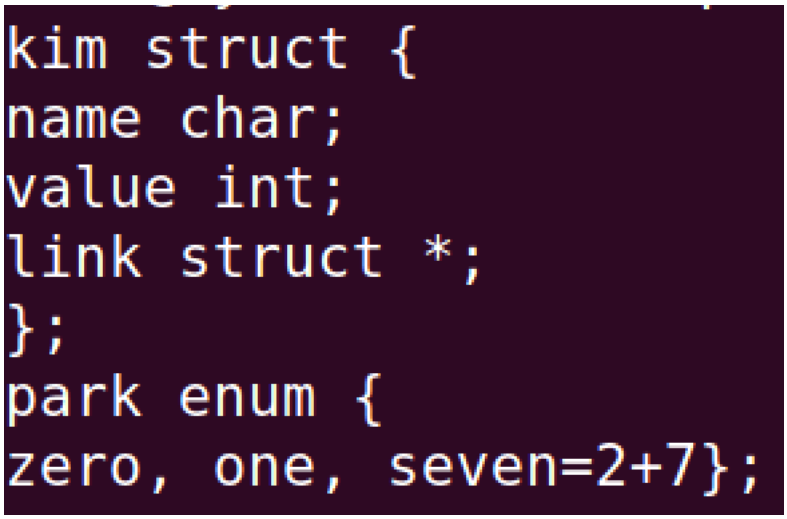
3) 타입 명시자 / 나열형 타입 명시자 출력

- 입력 소스코드

```
 struct node {
        char name;
        int value;
        struct node *link;
} kim;

enum ee {
        zero,
        one,
        seven=2+7
} park;
```

-실행결과



4) 일반선언자 출력

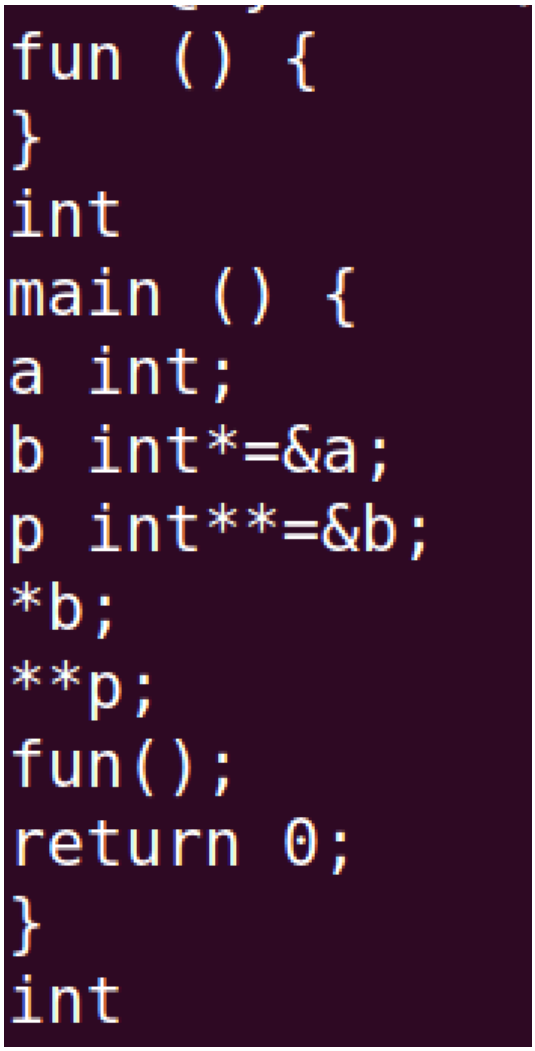
- 입력 소스코드

```
int fun() {

}

int main() {
	int a;
	int *b =&a;
	int **p = &b;
	*b;
	**p;
	fun();
	return 0;
}
```

- 실행결과

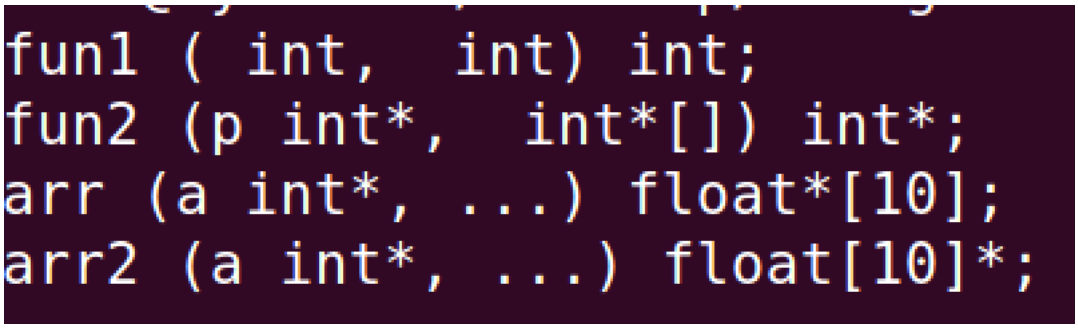


5) 추상선언자 출력

- 입력 소스코드

```
int fun1 (int,int);
int *fun2 (int*p,int*[]);
float (*arr[10]) (int*a, ...);
float (*arr2)[10](int*a, ...);
```

- 실행결과



6) 수식 출력

- 입력 소스코드

```
struct s {
	struct s *a;
	int b;
} lee[10];
void fun(int, int);
int main() {
	int kim[5] = {0};
	int i = 0;
	int a = 10, b = 5, c = 1, d = 2;
	"Korea";
	(kim + 3);
	kim[i+3];
	lee[1].a->b++;
	&kim[2];
	sizeof(float*[10]);
	(int)(x + 3.14);
	fun(2, i + 10);
	a+b*c||d;
	return 0;
}
```

- 실행결과

```
lee struct {
a struct *;
b int;
}[10];
fun ( int,  int) void;
main () {
kim int[5]={0};
i int=0;
a int=10;
b int=5;
c int=1;
d int=2;
"Korea";
kim+3;
kim[i+3];
lee[1].a->b++;
&kim[2];
sizeof(float*[10]);
(int)(a+3.14);
fun(2, i+10);
a+b*c||d;
return 0;
}
```

```
int
```

7) 선택문 출력

- 입력 소스코드

```
int main() {
	int x = 0;
	switch(x) {
		case 0:
			break;
		case 1:
			break;
		case 2:
			break;
		default:
			break;
	}
	return 0;
}
```

- 실행결과

```
main () {
x int=0;
switch(x){
case 0:
break;

case 1:
break;

case 2:
break;

default:
break;

}
return 0;
}
int
```

8) 반복문 출력

- 입력 소스코드

```
int main() {
        int x = 0;
        int i;
        while(1) {
                x = x+1;
                if(x > 100)
                break;
        }
```

```
        for(i = 0; i < 10; i++) {
                x--;
                ++x;
                x = x + i;
        }
        return 0;
}
```

- 실행결과