컴파일러 프로젝트 3 보고서 2012-11598 민두기

1. Undeclared Variable & Functions

Unary -> ID에서 variable이나 function이 선언 된 적 있는지 체크한다.

Finddecl(id \*)함수는 lexer에 넘겨 받은 id\* 를 사용해서 현재 ste stack에 존재하는지 같은 id\*를 가지고 있는 ste가 있는지 찾고 있으면 그 ste의 decl\* 을 넘겨주고 없으면 error문구를 띄우고 NULL을 넘긴다.

1. Redeclaration of Same Variables at Same Scope

def -> type\_specifier pointers ID ‘;’같은 선언하는 production에서 현재 scope에서 또 선언하려는 지 체크한다.

Check\_is\_declared\_for\_else(id \*)함수는 같은 id \* 를 가지는 ste가 현재 scope에 존재하는지 check하여 없으면 NULL을 return 해준다.

1. No Type Conversion

Expr -> unary ‘=’ expr 에서 check한다.



우선 LHS가 variable인지 체크하고 그 후 LHS의 type과 RHS의 type이 assign가능 한지 체크한다.

1. Pointer Operation

Pointer의 operation은 주어진 연산만 가능하게 하였다. 즉 주어진 연산에서 type을 check 할 때 pointer는 오류로 받아들이지 않는다.

다만 &나 \*에서는 먼저 variable인지 체크 하게 하였다. 프로젝트의 spec에서는 variable이 아닌것에 &나 \*를 붙일 필요가 없기 때문이다.

Ex)

1. Operation on Structures

Structure 의 경우 주어진 연산만 가능하게 하였다. 또한 ‘.’ 이나 ‘->’은 structure 만 사용 가능하게 하였다.



1. Structure (Pointer) Declaration

Structrue의 경우 scope상관없이 선언 되는 것이므로 따로 stack 을 만들어서 따로 관리하도록 하였따. Pointer declaration의 경우 딱히 신경쓰지 않았는데 저절로 잘 구현된거 같다.

1. Function

Return type은 assign체크와 동일하게 하였다. Actual argument와 formal arguments또한 assign체크와 동일하게 하였다.

다만 formal들을 엮을 때 같은 variable을 부를 수 있으니까 새로 만들어 엮었다

1. Operand Check와 LValue(Assignment) Check

LValue는 위에 설명한 type casting과 동일하게 구현하였다. Operand Check는 operand마다 각각의 함수를 만들어서 ppt에 적힌 type만 받아들이게 하였다.

다만 inttype만 가능한것 같은 경우에는 따로 그 type과 inttype을 비교해서 아니면 에러가 나게 하였다.

사실 여러 제한사항을 만족시키는것은 그리 어렵지 않았지만 전체적인 룰없이 하나 하나 다 체크를 하는게 고욕이었다. 또한 어려웠던 점들은 free시킬 때 어디까지 free시키고 function을 정의 할 때 formals를 집어넣을 때 어디까지 카피 시키냐였다. Pop\_scope()와 free\_scope()를 따로 만들어서 scope가 끝나면 free하게 하였으며, 프로그램이 종료되면 모든 scope들을 free하게 하였다.