**컴파일러 프로젝트 보고서 3 Semantic Analysis**

신소재공학부 2012006276 진현수

1) 컴파일 방법 및 환경

* Linux – Fedora 19
* gcc 컴파일 , yacc(bison)

2) 시멘틱 분석 구현 과정 및 주요 소스코드 설명

* main.c 수정

Semantic Analysis 과정을 확인 하기 위해서 다음과 같이 수정

#define NO\_PARSE FALSE

#define NO\_ANALYZE FALSE

#define NO\_CODE TRUE

int EchoSource = TRUE;

int TraceScan = FALSE;

int TraceParse = FALSE;

int TraceAnalyze = TRUE;

int TraceCode = FALSE;

* symtab.h , symtab.c 수정

BucketList를 감싸는 Scope 구조체를 생성한다.

typedef struct ScopeRec

{ char \* funcName;

int nestedLevel;

struct ScopeRec \* parent;

BucketList hashTable[SIZE]; /\* the hash table \*/

} \* Scope;

Static Scope를 구현하기 위해 Scope를 Stack 형태로 관리 하는 Function들을 추가

Scope sc\_top( void );

void sc\_pop( void );

void sc\_push( Scope scope );

* analyze.h , analyze.c 수정

Compound State를 추가 할 때 마다 새로운 Scope를 만들어 Stack에 Push. Scopmpound State를 빠져나갈때 Stack을 Pop. 이 때, Function의 경우 argument들과 Declaration의 Scope가 같도록 주의

새로운 선언이 있을 때는 *현재의 Scope*와 검사하여 중복이 있는지를 확인, 중복이 있으면 에러를 발생.

변수를 사용할 경우 Scope의 Stack을 가까운 순서대로 검색하여 변수가 있는지 확인하고, 없으면 에러를 발생.

Input, Output Function을 추가합니다. line number는 -1로 설정.

* 예시 및 결과화면

