
Project #3. Semantic Analysis Symbol Table & Type Checker

Yongjun Park
Hanyang University



Symbol Table & Type Checker

- Implement symbol table and type checker
- Traverse syntax tree created by parser
- Files to modify
 - globals.h
 - main.c
 - ytil.h, util.c
 - scan.h scan.c
 - parse.h, parse.c
 - symtab.h, symtab.c
 - analyze.h, analyze.c



main.c

- **Modify NO_ANALYZE, TraceParse, and TraceAnalyze to suit your assignment**



symtab.h, symtab.c

- Add scope and type to symbol table
- Implement hash table

```
-void st_insert( char * name, int lineno, int loc );  
+void st_insert( char * scope, char * name, ExpType type, int lineno, int loc );  
  
/* Function st_lookup returns the memory  
 * location of a variable or -1 if not found  
 */  
-int st_lookup ( char * name );  
+BucketList st_lookup ( char * scope, char * name );  
+BucketList st_lookup_excluding_parent ( char * scope, char * name );
```

```
typedef struct BucketListRec  
{ char * name;  
  ExpType type;  
  LineList lines;  
  int memloc ; /* memory location for variable  
  struct BucketListRec * next;  
} * BucketList;  
  
/* The record for each scope,  
 * including name, its bucket,  
 * and parent scope.  
 */  
typedef struct ScopeListRec  
{ char * name;  
  BucketList bucket[SIZE];  
  struct ScopeListRec * parent;
```

analyze.c

- **Modify symbol table generation**
 - buildSymtab(), insertNode(), traverse(), ... , scope and type concept
- **Modify the checkNode() function to check the semantics of C-Minus**
- **Insert built-in function**
 - Input(), output()



Symbol Table in Tiny

```
1: { Sample program
2:   in TINY language -
3:   computes factorial
4: }
5: read x; { input an integer }
6: if 0 < x then { don't compute if x <= 0 }
7:   fact := 1;
8:   repeat
9:     fact := fact * x;
10:    x := x - 1
11:  until x = 0;
12:  write fact { output factorial of x }
13: end
```

Variable Name	Location	Line Numbers					
x	0	5	6	9	10	10	11
fact	1	7	9	9	12		

Location

- Counter for variable memory locations.
- Never overlapped in a scope.



Symbol Table in C-Minus

```
1: /* A program to perform Euclid's
2:  Algorithm to computer gcd */
3:
4: int gcd (int u, int v)
5: {
6:     if (v == 0) return u;
7:     else return gcd(v,u-u/v*v);
8:     /* u-u/v*v == u mod v */
9: }
10:
11: void main(void)
12: {
13:     int x; int y;
14:     x = input(); y = input();
15:     output(gcd(x,y));
16: }
```

Name	Type	Location	Scope	Line Numbers
output	Void	0	global	0 15
Input	Integer	1	global	0 14 14
gcd	Integer	2	global	4 7 15
main	Void	3	global	11
u	Integer	0	gcd	4 6 7 7
v	Integer	1	gcd	4 6 7 7 7
x	Integer	0	main:11~16	13 14 15
y	Integer	1	main:11~16	13 14 15

Implementation Notes

- Variables follow scope of each compound statement.
- Throws an error when an undeclared variable is used.
- Built-in functions should be always accessible.
- As long as the scope concept is implemented properly, you can use any implementation or output form.

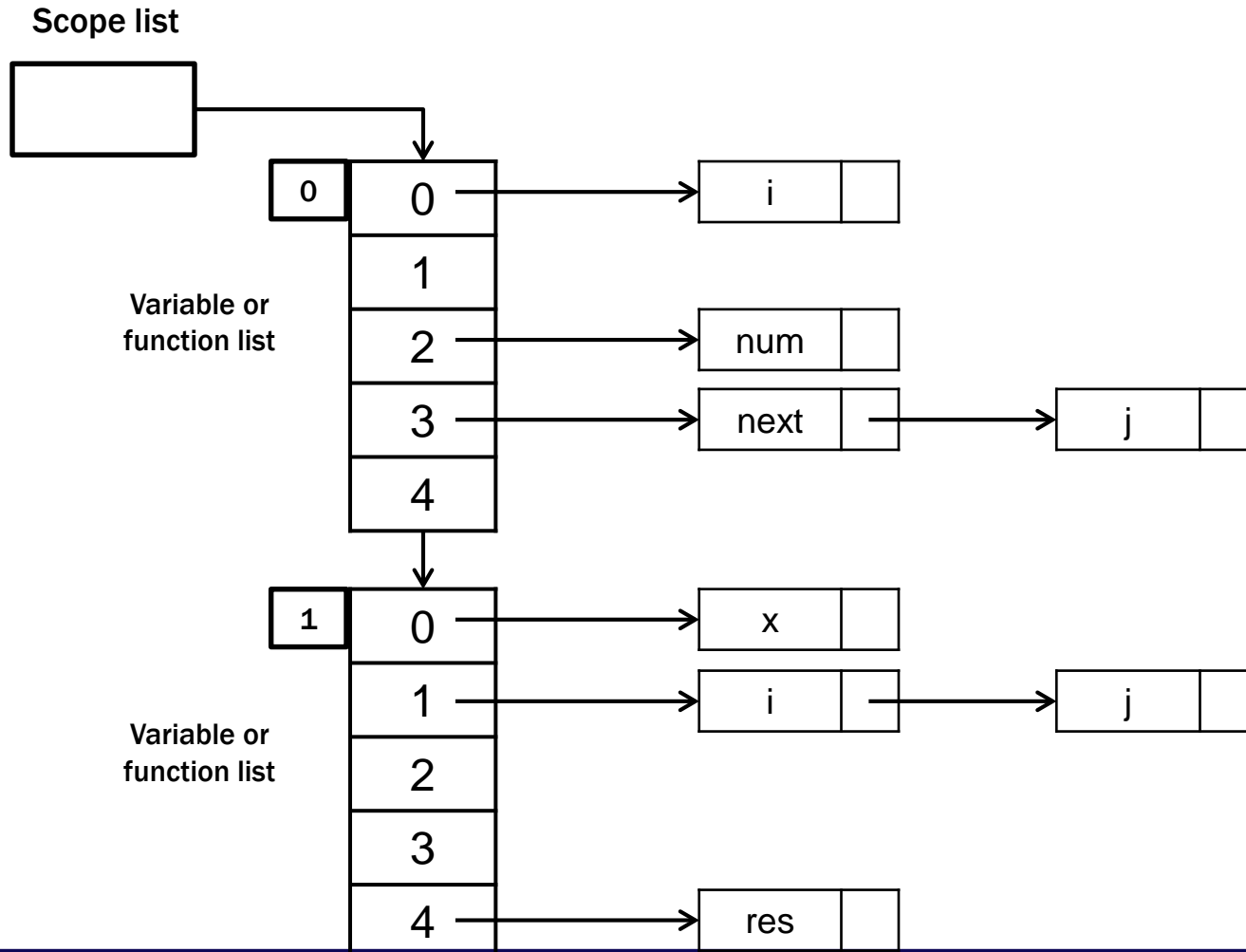


Built-in function

- **int input()**
 - One integer value is input from the user.
- **void output()**
 - Prints the value of arg.
- **These two functions are considered to be global functions defined by default.**



Symbol Table in C-Minus (Sample)



Type Checker

- **Type checking for functions and variables.**
 - The type “void” is only available for functions.
 - Check return type.
 - Verify the type match of two operands when assigning.
 - Check the argument number when calling function.
 - Check if conditional of “If” or “While” has a value.
 - Check other things by referring to C-Minus syntax.



Report & Full Source Tree

- **Contents**

- Build environment(OS, compiler, ...).
- Semantic analysis implementation process and source code description of principal parts.

- **Format**

- hwp, doc, pdf, ...

- **Submission deadline**

- Push until Wednesday, December 6, 2017, 23:59:59.
- Master branch will be cloned at 0:00 on Thursday, December 7, 2017.

