
Project #1. Scanner 2019

Scanner

- **Implementation of C-scanner (both 2 methods)**
 - **Implementation of C-Scanner using C-code (modify Tiny compiler code)**
 - **globals.h main.c util.h util.c scan.h scan.c**
 - Implementation of C-Scanner using lex(flex) by Tiny.l modification



Lexical Convention of C-Minus

- **Keyword**

else if int return void while (lower case)

- **Symbol**

+ - * / < <= > >= == != = ; ,
() [] { } /* */

- **Token**



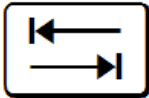
*ID = letter letter**

*NUM = digit digit**

letter = a | ... | z | A | ... | Z

digit = 0 | 1 | ... | 9

Lexical Convention of C-Minus

- **White space:** , , 
 - Ignore other cases except WS between *ID*, *NUM*, and keywords (ex: beginning and end of line)
- **Comments (/ * ... * /)** follow normal C notation.
 - Cannot be nested.
- **Please see “Kenneth C. Louden book p. 491-492”**

Hint: where to see?

- **globals.h**
 - TokenType should be modified for C-Minus

```
#endif

/* MAXRESERVED = the number of reserved words */
#define MAXRESERVED 8
#define MAXRESERVED 12

typedef enum
    /* book-keeping tokens */
    {ENDFILE,ERROR,
    /* reserved words */
-   IF,THEN,ELSE,END,REPEAT,UNTIL,READ,WRITE,
+   IF,ELSE,WHILE,RETURN,INT,VOID, /* discarded */ THEN,END,REPEAT,UNTIL,READ,WRITE,
    /* multicharacter tokens */
    ID,NUM,
    /* special symbols */
-   ASSIGN,EQ,LT,PLUS,MINUS,TIMES,OVER,LPAREN,RPAREN,SEMI
+   ASSIGN,EQ,NE,LT,LE,GT,GE,PLUS,MINUS,TIMES,OVER,LPAREN,RPAREN,LBRACE,RBRACE,LCURLY,RCURLY,SEMI,COMMA
    } TokenType;

extern FILE* source; /* source code text file */
```

Hint: where to see?

- **main.c**
 - To meet scanner project goal
 - NO_PARSE, EchoSource, TraceScan

```
#include "globals.h"

/* set NO_PARSE to TRUE to get a scanner-only compiler */
#define NO_PARSE FALSE
#define NO_PARSE TRUE
/* set NO_ANALYZE to TRUE to get a parser-only compiler */
#define NO_ANALYZE FALSE

=====

FILE * code;

/* allocate and set tracing flags */
-int EchoSource = FALSE;
-int TraceScan = FALSE;
+int EchoSource = TRUE;
+int TraceScan = TRUE;
int TraceParse = FALSE;
int TraceAnalyze = FALSE;
int TraceCode = FALSE;
```

Hint: where to see?

- **scan.c**
 - Need to add states for C-Minus DFA

```
/* states in scanner DFA */
typedef enum
- { START, INASSIGN, INCOMMENT, INNUM, INID, DONE }
+ { START, INEQ, INCOMMENT, INNUM, INID, DONE, INLT, INGT, INNE, INOVER, INCOMMENT_ }
    StateType;
```

Hint: where to see?

- **scan.c**
 - Reserved word should be added for C-Minus

```
/* lookup table of reserved words */
static struct
{
    char* str;
    TokenType tok;
} reservedWords[MAXRESERVED]
- = {{ "if", IF }, { "then", THEN }, { "else", ELSE }, { "end", END },
-   { "repeat", REPEAT }, { "until", UNTIL }, { "read", READ },
-   { "write", WRITE } };
+ = {{ "if", IF }, { "else", ELSE }, { "while", WHILE }, { "return", RETURN }, { "int", INT }, { "void", VOID },
+   /* discarded */ { "then", THEN }, { "end", END }, { "repeat", REPEAT }, { "until", UNTIL }, { "read", READ }, { "write", WRITE }
+   };
```


Hint: where to see?

- **scan.c**
 - Need to modify getToken for C-Minus

```
-      case '/':  
-          currentToken = OVER;  
-          break;  
      case '(':  
          currentToken = LPAREN;  
          break;  
      case ')':  
          currentToken = RPAREN;  
          break;  
+      case '{':  
+          currentToken = LCURLY;  
+          break;  
+      case '}':  
+          currentToken = RCURLY;  
+          break;  
+      case '[':  
+          currentToken = LBRACE;  
+          break;  
+      case ']':  
+          currentToken = RBRACE;  
+          break;
```

Hint: where to see?

- **util.c**
 - Need to modify printToken() for C-Minus

```
fprintf(listing,
        "reserved word: %s\n", tokenString);
break;
- case ASSIGN: fprintf(listing, ":=\n"); break;
+ case ASSIGN: fprintf(listing, "=\n"); break;
+ case EQ: fprintf(listing, "=\n"); break;
+ case NE: fprintf(listing, "!=\n"); break;
case LT: fprintf(listing, "<\n"); break;
- case EQ: fprintf(listing, "=\n"); break;
+ case LE: fprintf(listing, "<=\n"); break;
+ case GT: fprintf(listing, ">\n"); break;
+ case GE: fprintf(listing, ">=\n"); break;
case LPAREN: fprintf(listing, "(\n"); break;
case RPAREN: fprintf(listing, ")\n"); break;
+ case LBRACE: fprintf(listing, "[\n"); break;
+ case RBRACE: fprintf(listing, "]\n"); break;
+ case LCURLY: fprintf(listing, "{\n"); break;
+ case RCURLY: fprintf(listing, "}\n"); break;
case SEMI: fprintf(listing, ";\n"); break;
+ case COMMA: fprintf(listing, ",\n"); break;
case PLUS: fprintf(listing, "+\n"); break;
case MINUS: fprintf(listing, "-\n"); break;
case TIMES: fprintf(listing, "*\n"); break;
```

Example– Tiny compiler modification

test.cm

```
/* A program to perform Euclid's  
Algorithm to computer gcd */
```

```
int gcd (int u, int v)
```

```
{
```

```
    if (v == 0) return u;
```

```
    else return gcd(v,u-u/v*v);
```

```
    /* u-u/v*v == u mod v */
```

```
}
```

```
void main(void)
```

```
{
```

```
    int x; int y;
```

```
    x = input(); y = input();
```

```
    output(gcd(x,y));
```

```
}
```

Comment

Execution:

`./cminus_cimpl test.cm`

```

C-MINUS COMPILATION: test.cm
1: /* A program to perform Euclid's
2:  Algorithm to computer gcd */
3:
4: int gcd (int u, int v)
    4: reserved word: int
    4: ID, name= gcd
    4: (
    4: reserved word: int
    4: ID, name= u
    4: ,
    4: reserved word: int
    4: ID, name= v
    4: )
5: {
6:     5: {
        6: if (v == 0) return u;
        6: reserved word: if
        6: (
        6: ID, name= v
        6: ==
        6: NUM, val= 0
        6: )
        6: reserved word: return
        6: ID, name= u
        6: ;
    7:     else return gcd(v,u-u/v*v);
    7: reserved word: else
    7: reserved word: return
    7: ID, name= gcd
    7: (
    7: ID, name= v
    7: ,
    7: ID, name= u
    7: -
    7: ID, name= u
    7: /
    7: ID, name= v
    7: *
    7: ID, name= v
    7: )
    7: ;
8:     /* u-u/v*v == u mod v */
9: }
9: }

```

```

10:
11: void main(void)
    11: reserved word: void
    11: ID, name= main
    11: (
    11: reserved word: void
    11: )
12: {
13:     12: {
        13: int x; int y;
        13: reserved word: int
        13: ID, name= x
        13: ;
        13: reserved word: int
        13: ID, name= y
        13: ;
    14:     x = input(); y = input();
        14: ID, name= x
        14: =
        14: ID, name= input
        14: (
        14: )
        14: ;
        14: ID, name= y
        14: =
        14: ID, name= input
        14: (
        14: )
        14: ;
    15:     output(gcd(x,y));
        15: ID, name= output
        15: (
        15: ID, name= gcd
        15: (
        15: ID, name= x
        15: ,
        15: ID, name= y
        15: )
        15: )
        15: ;
    16: }
17: EOF

```

Scanner

- **Implementation of C-scanner (both 2 methods)**
 - Implementation of C-Scanner using C-code (modify Tiny compiler code)
 - `globals.h main.c util.h util.c scan.h scan.c`
 - Implementation of C-Scanner using lex(flex) by Tiny.l modification



lex / flex

- Lexeme analysis
- Automatically generate a target scanner based on input Res
- Work with yacc (bison)
- <http://flex.sourceforge.net/>
 - Manual: <http://flex.sourceforge.net/manual/>

lex environment

- **Ubuntu 14.04 기준:**
 - apt-get install flex
- **Usage**
 - tiny.l (in Tiny source) should be modified
 - flex <Lex Filename>
 - ex) flex cminus.l
 - lex.yy.c will be created
- **Output can be different**
 - It's okay if the output is somewhat different from the previous work.

Hint: where to see?

- **globals.h, main.c, util.c**
 - Same as manual implementation
- **scan.c**
 - This file is not used because getToken() is automatically generated using flex



Hint: where to see?

- **cminus.l**
 - Should be created for C-Minus using tiny.l

Example– Flex

test.cm

```
/* A program to perform Euclid's  
Algorithm to computer gcd */
```

```
int gcd (int u, int v)
```

```
{
```

```
    if (v == 0) return u;
```

```
    else return gcd(v,u-u/v*v);
```

```
    /* u-u/v*v == u mod v */
```

```
}
```

```
void main(void)
```

```
{
```

```
    int x; int y;
```

```
    x = input(); y = input();
```

```
    output(gcd(x,y));
```

```
}
```

Comment

Execution:

`./cminus_flex test.cm`

C-MINUS COMPILATION: test.cm

```
4: reserved word: int
4: ID, name= gcd
4: (
4: reserved word: int
4: ID, name= u
4: ,
4: reserved word: int
4: ID, name= v
4: )
5: {
6: reserved word: if
6: (
6: ID, name= v
6: ==
6: NUM, val= 0
6: )
6: reserved word: return
6: ID, name= u
6: ;
7: reserved word: else
7: reserved word: return
7: ID, name= gcd
7: (
7: ID, name= v
7: ,
7: ID, name= u
7: -
7: ID, name= u
7: /
7: ID, name= v
7: *
7: ID, name= v
7: )
7: ;
```

```
9: }
11: reserved word: void
11: ID, name= main
11: (
11: reserved word: void
11: )
12: {
13: reserved word: int
13: ID, name= x
13: ;
13: reserved word: int
13: ID, name= y
13: ;
14: ID, name= x
14: =
14: ID, name= input
14: (
14: )
14: ;
14: ID, name= y
14: =
14: ID, name= input
14: (
14: )
14: ;
15: ID, name= output
15: (
15: ID, name= gcd
15: (
15: ID, name= x
15: ,
15: ID, name= y
15: )
15: )
15: ;
16: }
17: EOF
```



Compilation

- **Use the Makefile!**
 - Use Tiny compiler Makefile
 - The Makefile should be modified



Compilation using flex

- **Compilation using flex**

- Following code should be added to Makefile

```
#by flex
cminus_flex: $(OBJF_FLEX)
    $(CC) $(CFLAGS) main.o util.o lex.yy.o -o cminus_flex -lfl

lex.yy.o: cminus.l scan.h util.h globals.h
    flex cminus.l
    $(CC) $(CFLAGS) -c lex.yy.c -lfl
```

- *'-lfl': must be added*
 - *'cminus.l' should exist in the same folder with other header files.*
 - OS X: -ll



Report

- **Guideline (~5pages)**

- Compilation method and environment
- Explanation about how to implement and how to operate
- Example and Result Screenshot

- **File format**

- MS Word, HWP, PDF, ...
- GitLab Wiki Not Allowed

(If you want, write report in markdown and **take screenshot** and submit in other formats(PDF, JPEG, ...))

Submission

- **Submission directory in repository: 1_Scanner**
(Please submit all your codes and reports into the submission directory)
- **Questions**
compiler.teachingassistant@gmail.com
- **Scanner submission deadline**
– **10/20(Sun) 23:59:59**

Contact (Prof. Yongjun Park)

- **Submission**

- Where: Using GitLab

- <https://hconnect.hanyang.ac.kr>

- Git Project:

- `https://hconnect.hanyang.ac.kr/2019_ELE4029_12214/2019_ELE4029_Student#.git`

- Example URL: `https://hconnect.hanyang.ac.kr/2019_ELE4029_12214/2019_ELE4029_2019000000.git`

- The Submission Directory is in Repo: 1_Scanner, 2_Parser, 3_Semantic, ...

- Teaching Assistant

- `compiler.teachingassistant@gmail.com`

- If you don't have the GITLAB account, **please let him know** the account information after creation.

- **What to submit**

- All the source codes and the report

Q&A

