C-Minus Compiler Project

Compilation Requirement

- GCC-11
- FLEX 2.6.4

Language Element

Keyword

- if
- else
- while
- return
- int
- void

Symbol

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

Identifier and Number rule

- letter = [a-zA-Z]
- digit = [0-9]
- ID = letter (letter | digit) *
- NUM = digit digit *

Project 1 - Scanner

How to use

Custom c-minus code scanner

```
$ make cminus_cimpl
$ ./cminus_cimpl <filename>
```

Flex c-minus code scanner

```
$ make cminus_lex
$ ./cminus_lex <filename>
```

How to implement

Code Scanner with custom c code

`==`와 `!=`, `<=`와 `<`처럼 분기가 나뉘는 문자에 대해서는 다른 `state`로 전이되도록 설계했다. 이후 전이된 `state`에서 어떤 token으로 scan될지 판단한다.

```
case START:
    if (isdigit(c))
        state = INNUM;
    else if (isalpha(c))
        state = INID;
    else if (c == '=')
        state = INEQ;
    else if (c == '<')
        state = INLT;
    else if (c == '>')
        state = INGT;
    else if (c == '!')
        state = INNE;
    else if (c == '/') {
        save = FALSE;
        state = INOVER;
    }
    else if ((c == ' ') || (c == '\t') || (c == '\n'))
        save = FALSE;
    else
    {   state = DONE;
        switch (c)
        { case EOF:
            save = FALSE;
            currentToken = ENDFILE;
            break;
        case ',':
            currentToken = COMMA;
            break;
        case '+':
            currentToken = PLUS;
            break;
        case '-':
            currentToken = MINUS;
            break;
        case '*':
            currentToken = TIMES;
            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;
        case ';':
            currentToken = SEMI;
            break;
        default:
            currentToken = ERROR;
            break;
        }
    }
    break;
```

`state`들을 표로 표현하면 아래와 같다.

`state`	첫 글자	설명
`START`		시작 `state`
`INNUM`	`digit`	숫자 토큰을 만드는 중
`INID`	`letter`	ID 토큰을 만드는 중
`INEQ`	`=`	`==`의 가능성이 있는 상태
`INLT`	`<`	`<=`의 가능성이 있는 상태

state	첫 글자	설명
INGT	>	>=의 가능성이 있는 상태
INNE	!	!=의 가능성이 있는 상태
INOVER	/	주석 /* */의 가능성이 있는 상태
INCOMMENT		INOVER일 때 *을 입력받은 상태
INCOMMENT_		INCOMMENT일 때 *을 입력받은 상태 여기서 /을 입력받으면 주석이 끝난것이므로, START로 전이된다
DONE		토큰화가 끝난 상태

INEQ를 예로 들면, =를 입력받은 후, =를 다시 입력받으면 해당 토큰은 ==로 결정된다. 반면 =가 아닌 문자를 받으면 =로 결정되고, 따라서 입력받았던 문자를 되돌리기 위해 ungetNextChar()이 필요하다.

```
case INEQ:
    if (c == '=')
    {   state = DONE;
        currentToken = EQ;
    }
    else
    {   ungetNextChar();
        save = FALSE;
        state = DONE;
        currentToken = ASSIGN;
    }
    break;
```

주석 처리를 예로 들면, START에서 /를 받아 INOVER로 전이되면, 주석인지 아니면 /연산인지 판단이 필요하다. 따라서 INOVER에서 *가 아닌 문자가 들어왔을 경우에는 OVER로 판단하고, 토큰화를 끝낸다. 반면 *가 들어왔을 때는 주석으로 판단하고 INCOMMENT로 전이한다.

INCOMMENT에서 *를 입력받으면 주석을 끝내는 심볼일 것으로 예상할 수 있기 때문에 INCOMMENT_로 전이한다.

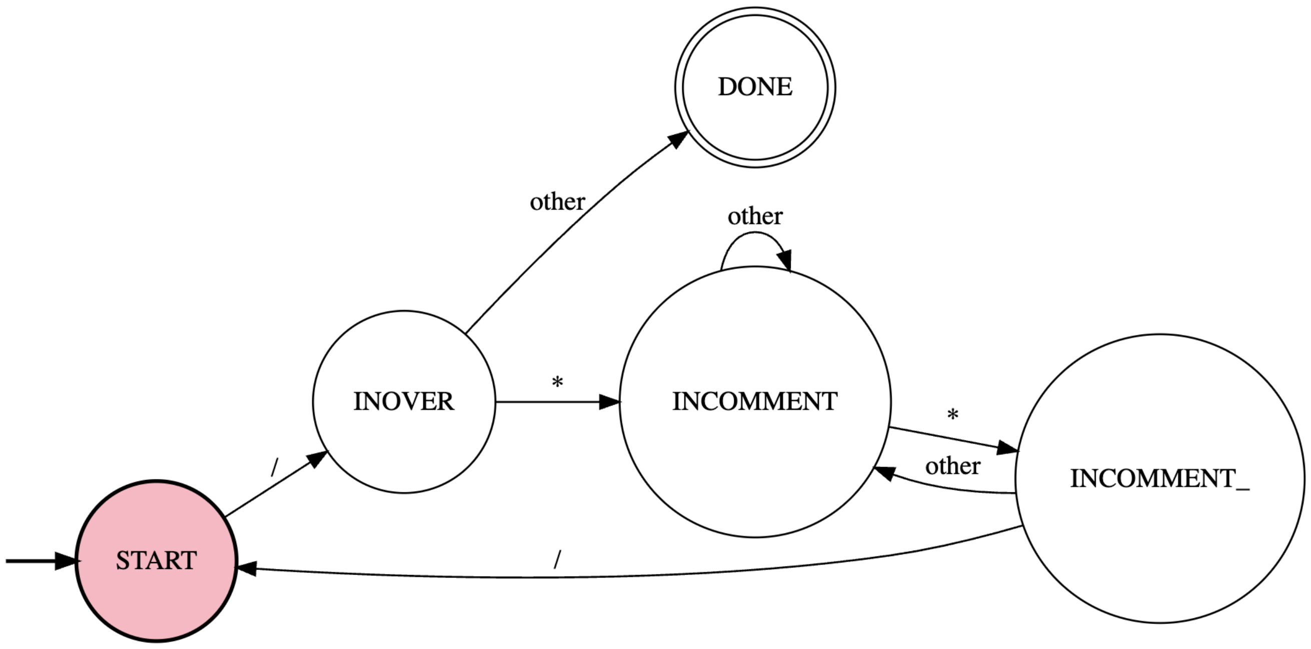
INCOMMENT_에서 /를 입력받으면 주석을 끝내는 심볼임을 확정지을 수 있기 때문에 START로 전이하고, 토큰화를 재개한다. 만약 다른 문자를 받는다면 주석이 끝난다고 예상할 수 없는 상황이므로 INCOMMENT로 되돌아간다.

만약 주석이 닫히지 않는다면, EOF를 출력하도록 설계했다.

```
case INOVER:
    if (c == '*')
    {   save = FALSE;
        state = INCOMMENT;
    }
    else
    {   ungetNextChar();
        state = DONE;
```

```
            currentToken = OVER;
        }
        break;
    case INCOMMENT:
        if (c == '*')
        {   save = FALSE;
            state = INCOMMENT_;
        } else if (c == EOF)
        {   save = FALSE;
            state = DONE;
            currentToken = ENDFILE;
        } else save = FALSE;
        break;
    case INCOMMENT_:
        if (c == '/')
        {   save = FALSE;
            state = START;
        }
        else if (c == EOF)
        {   state = DONE;
            currentToken = ENDFILE;
        }
        else
        {   save = FALSE;
            state = INCOMMENT;
        }
        break;
```

DFA로 표현하면 아래와 같다.



Code Scanner made by FLEX

`Keyword`와 `Symbol`에 따라 `globals.h`에 정의된 `TokenType`을 반환한다.

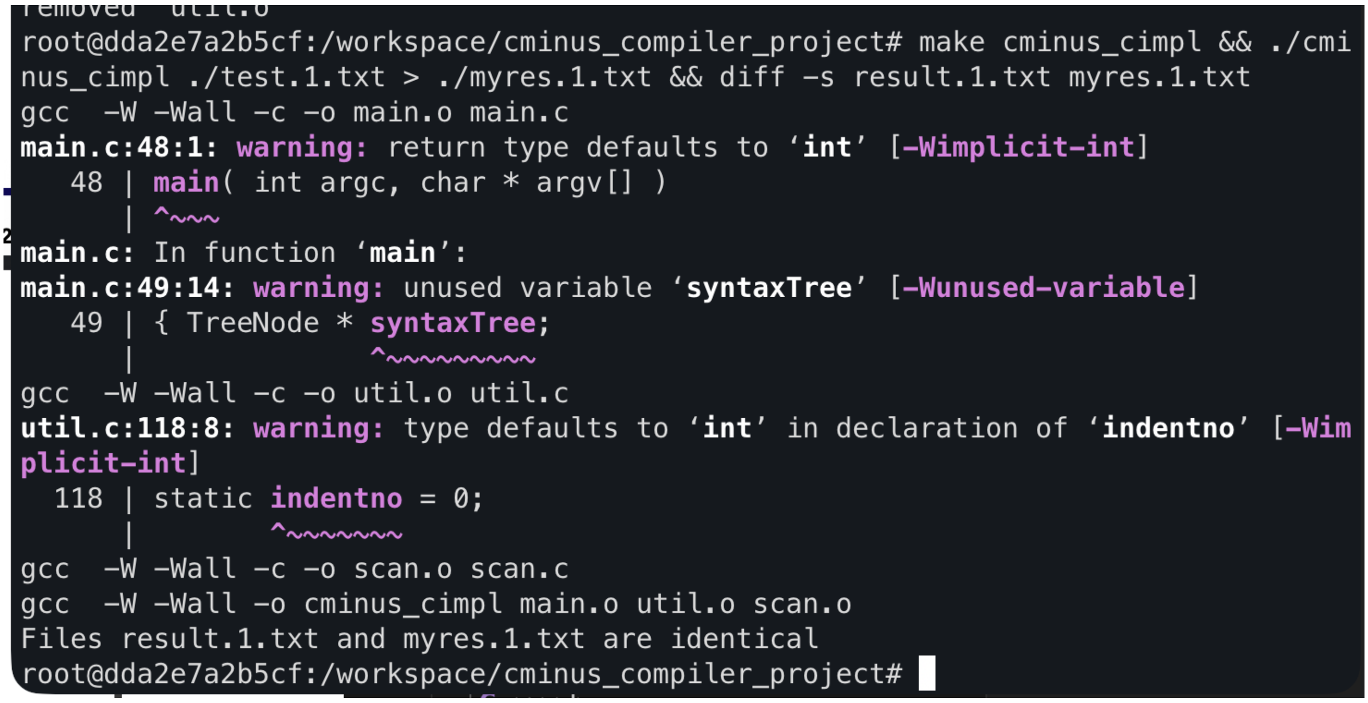
/* */의 경우에는 /*을 입력받았을 때 COMMENT로 전이하고, COMMENT에서 */을 입력받았을 때 초기 state인 INITIAL로 전이한다.

```
%x COMMENT

%%

"if"            {return IF;}
"else"          {return ELSE;}
"while"         {return WHILE;}
"return"        {return RETURN;}
"int"           {return INT;}
"void"          {return VOID;}
"="             {return ASSIGN;}
"=="            {return EQ;}
"!="            {return NE;}
"<"             {return LT;}
"<="            {return LE;}
">"             {return GT;}
">="            {return GE;}
"+"             {return PLUS;}
"-"             {return MINUS;}
"*"             {return TIMES;}
"/"             {return OVER;}
"("             {return LPAREN;}
")"             {return RPAREN;}
"{"             {return LCURLY;}
"}"             {return RCURLY;}
"["             {return LBRACE;}
"]"             {return RBRACE;}
";"             {return SEMI;}
","             {return COMMA;}
{number}        {return NUM;}
{identifier}    {return ID;}
{newline}       {lineno++;}
{whitespace}    {/* skip whitespace */}
"/*"            {BEGIN(COMMENT);}
<COMMENT>"*/"   {BEGIN(INITIAL);}
<COMMENT>\n     {lineno++;}
<COMMENT>.      {}
.               {return ERROR;}
```

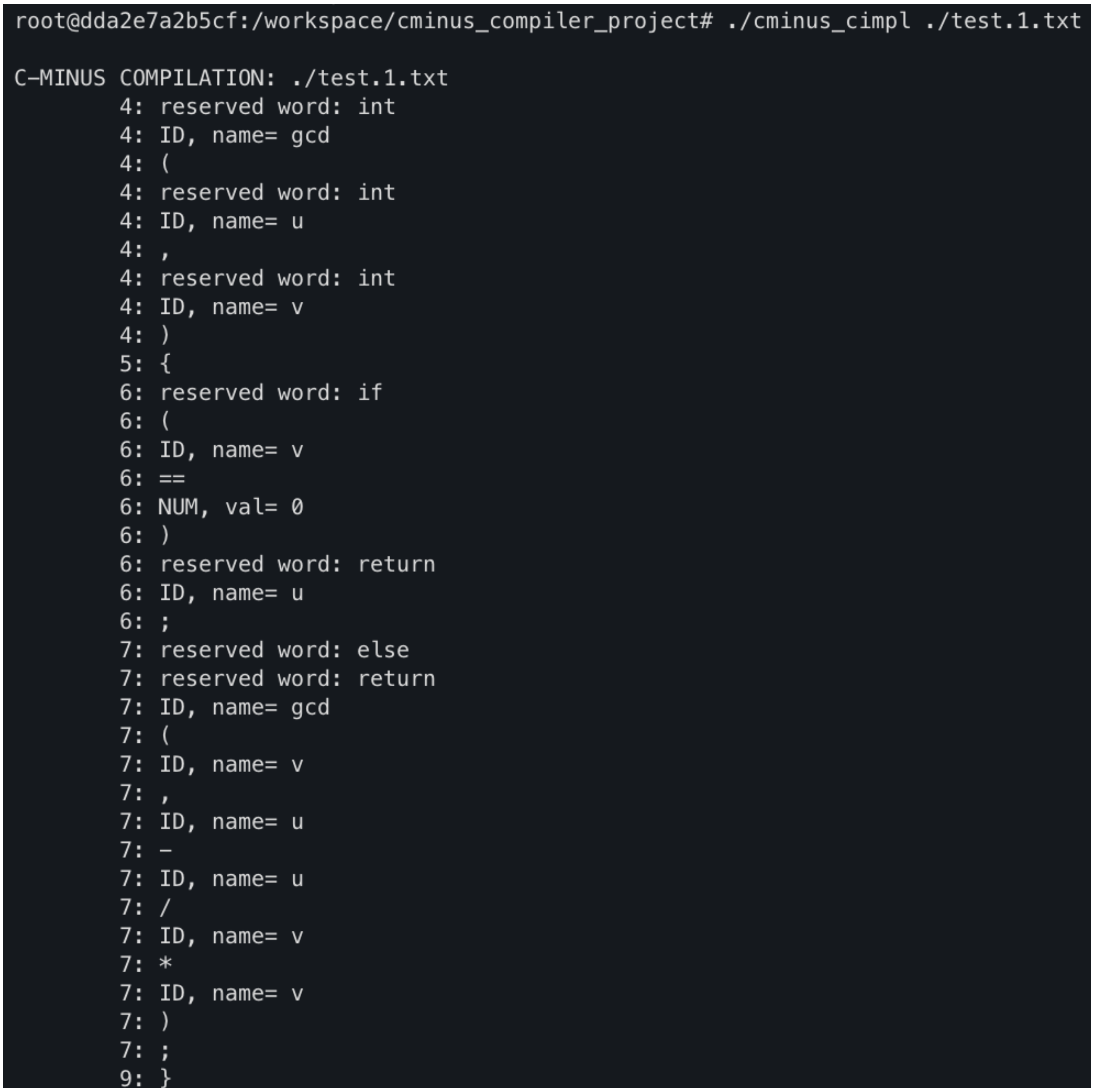
Test Case

Code Scanner with custom c code

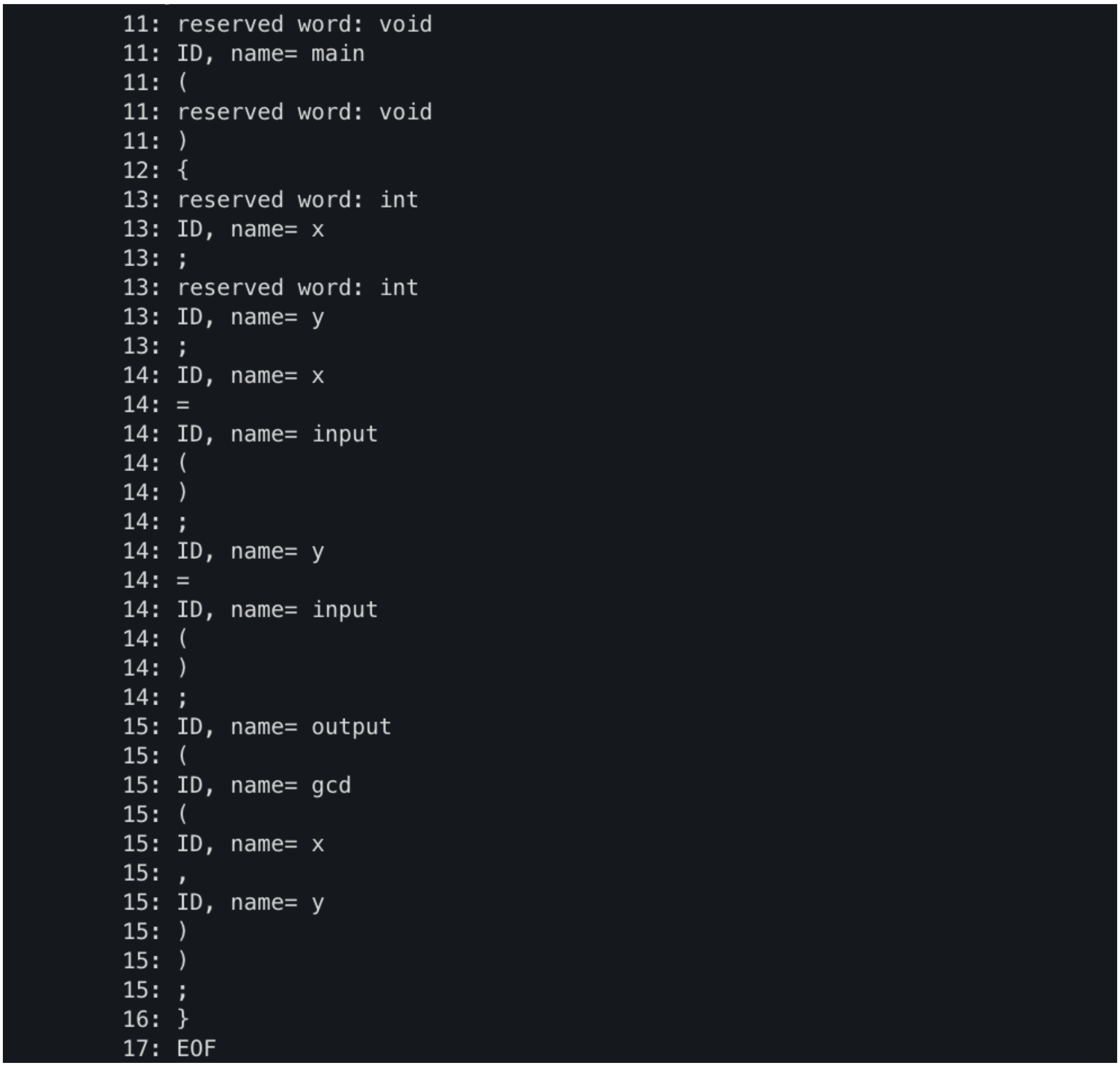
Test Case #1



```
removed  util.o
root@dda2e7a2b5cf:/workspace/cminus_compiler_project# make cminus_cimpl && ./cmi
nus_cimpl ./test.1.txt > ./myres.1.txt && diff -s result.1.txt myres.1.txt
gcc  -W -Wall -c -o main.o main.c
main.c:48:1: warning: return type defaults to ‘int’ [-Wimplicit-int]
   48 | main( int argc, char * argv[] )
      | ^~~~
main.c: In function ‘main’:
main.c:49:14: warning: unused variable ‘syntaxTree’ [-Wunused-variable]
   49 | { TreeNode * syntaxTree;
      |              ^~~~~~~~~~
gcc  -W -Wall -c -o util.o util.c
util.c:118:8: warning: type defaults to ‘int’ in declaration of ‘indentno’ [-Wim
plicit-int]
  118 | static indentno = 0;
      |        ^~~~~~~~
gcc  -W -Wall -c -o scan.o scan.c
gcc  -W -Wall -o cminus_cimpl main.o util.o scan.o
Files result.1.txt and myres.1.txt are identical
root@dda2e7a2b5cf:/workspace/cminus_compiler_project# 
```

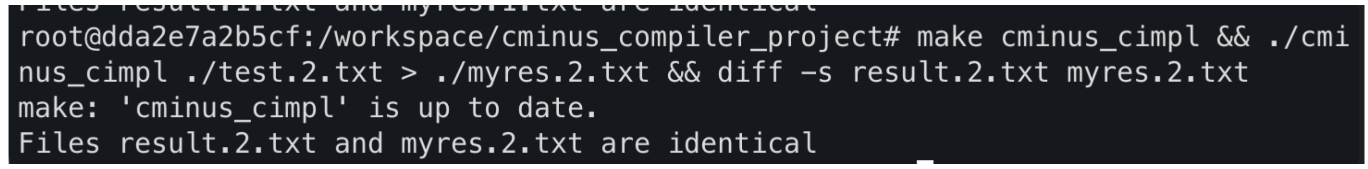


```
root@dda2e7a2b5cf:/workspace/cminus_compiler_project# ./cminus_cimpl ./test.1.txt

C-MINUS COMPILATION: ./test.1.txt
        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: }
```

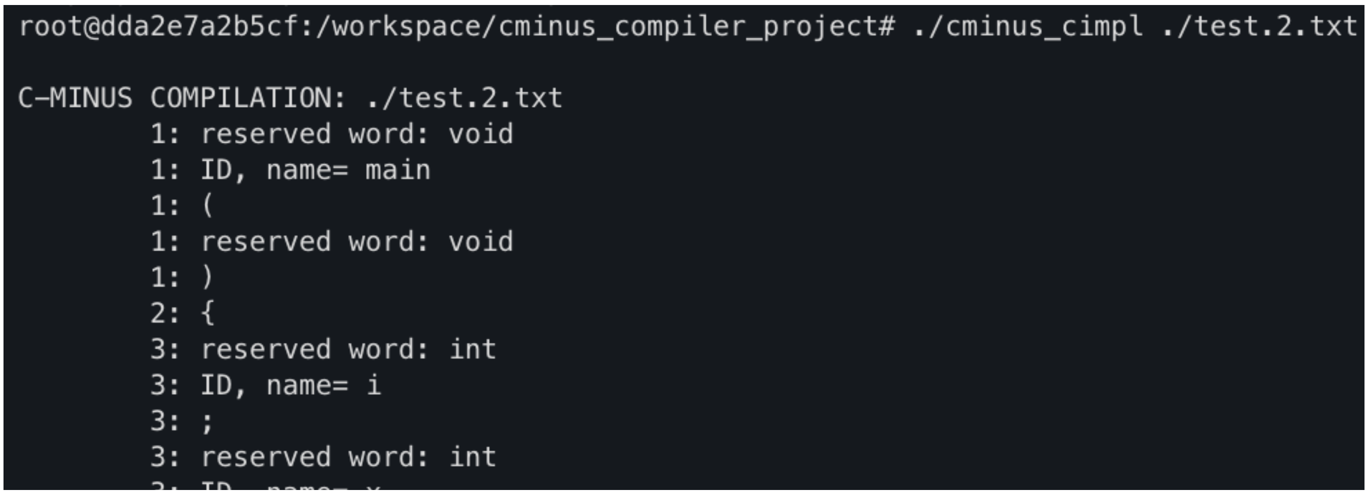


```
        11: reserved word: void
        11: ID, name= main
        11: (
        11: reserved word: void
        11: )
        12: {
        13: reserved word: int
        13: ID, name= x
        13: ;
        13: reserved word: int
        13: ID, name= y
        13: ;
        14: ID, name= x
        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
```

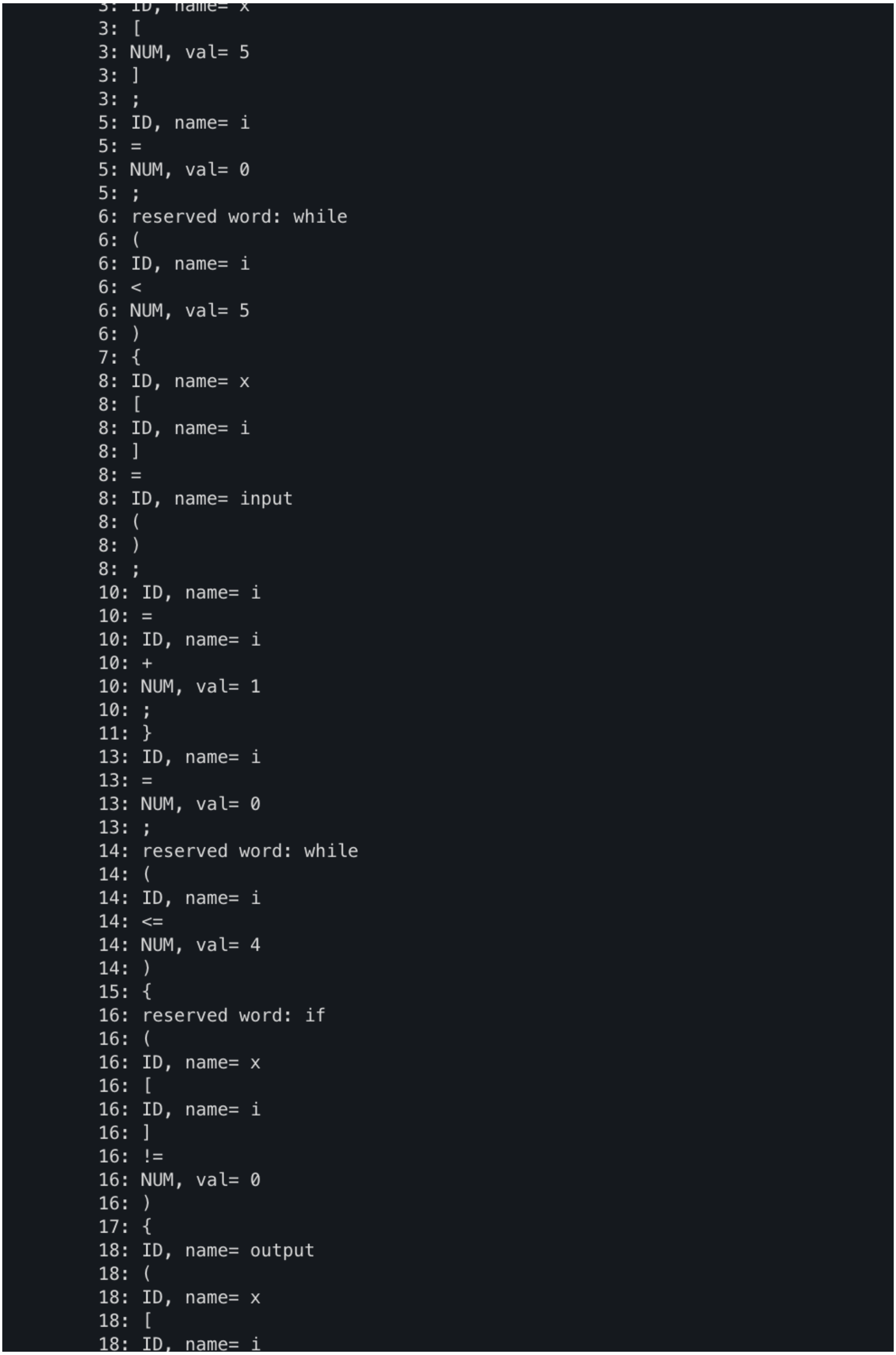
Test Case #2



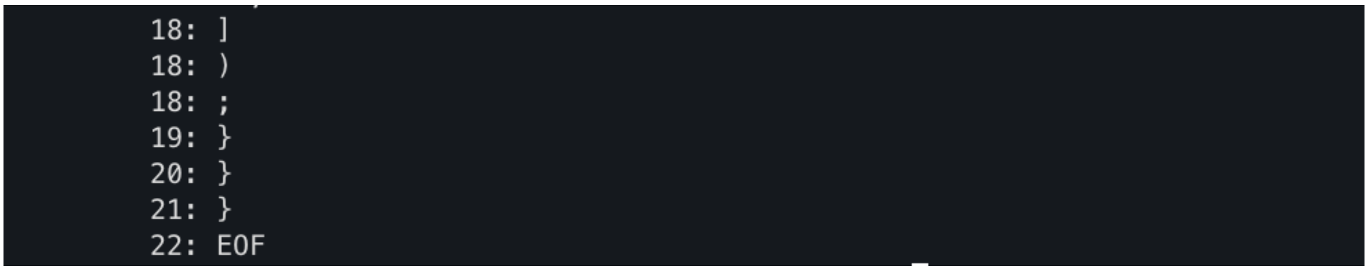
```
Files result.1.txt and myres.1.txt are identical
root@dda2e7a2b5cf:/workspace/cminus_compiler_project# make cminus_cimpl && ./cmi
nus_cimpl ./test.2.txt > ./myres.2.txt && diff -s result.2.txt myres.2.txt
make: 'cminus_cimpl' is up to date.
Files result.2.txt and myres.2.txt are identical
```



```
root@dda2e7a2b5cf:/workspace/cminus_compiler_project# ./cminus_cimpl ./test.2.txt

C-MINUS COMPILATION: ./test.2.txt
        1: reserved word: void
        1: ID, name= main
        1: (
        1: reserved word: void
        1: )
        2: {
        3: reserved word: int
        3: ID, name= i
        3: ;
        3: reserved word: int
        3: ID, name= x
```



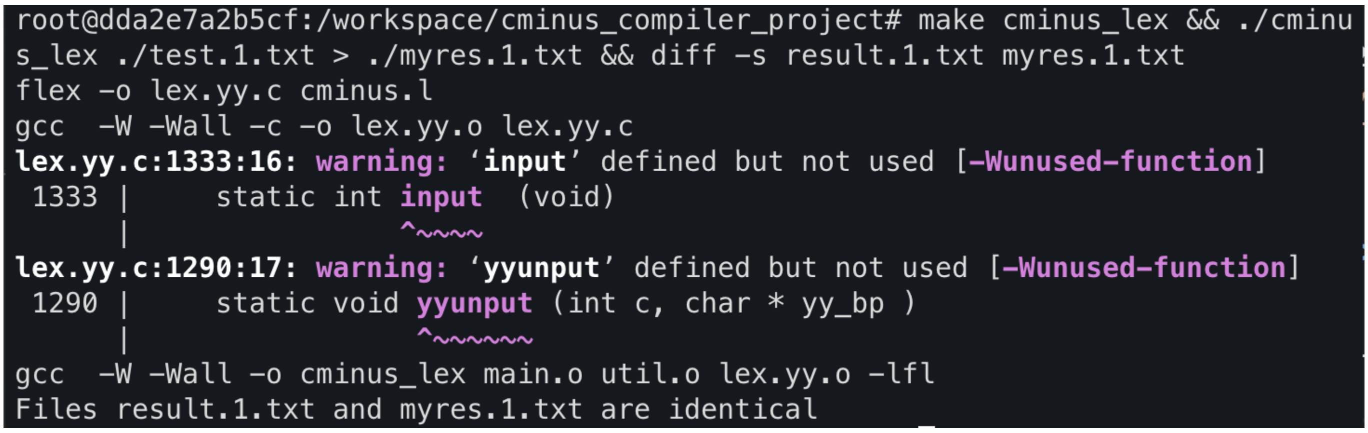
```
3: ID, name= x
3: [
3: NUM, val= 5
3: ]
3: ;
5: ID, name= i
5: =
5: NUM, val= 0
5: ;
6: reserved word: while
6: (
6: ID, name= i
6: <
6: NUM, val= 5
6: )
7: {
8: ID, name= x
8: [
8: ID, name= i
8: ]
8: =
8: ID, name= input
8: (
8: )
8: ;
10: ID, name= i
10: =
10: ID, name= i
10: +
10: NUM, val= 1
10: ;
11: }
13: ID, name= i
13: =
13: NUM, val= 0
13: ;
14: reserved word: while
14: (
14: ID, name= i
14: <=
14: NUM, val= 4
14: )
15: {
16: reserved word: if
16: (
16: ID, name= x
16: [
16: ID, name= i
16: ]
16: !=
16: NUM, val= 0
16: )
17: {
18: ID, name= output
18: (
18: ID, name= x
18: [
18: ID, name= i
```



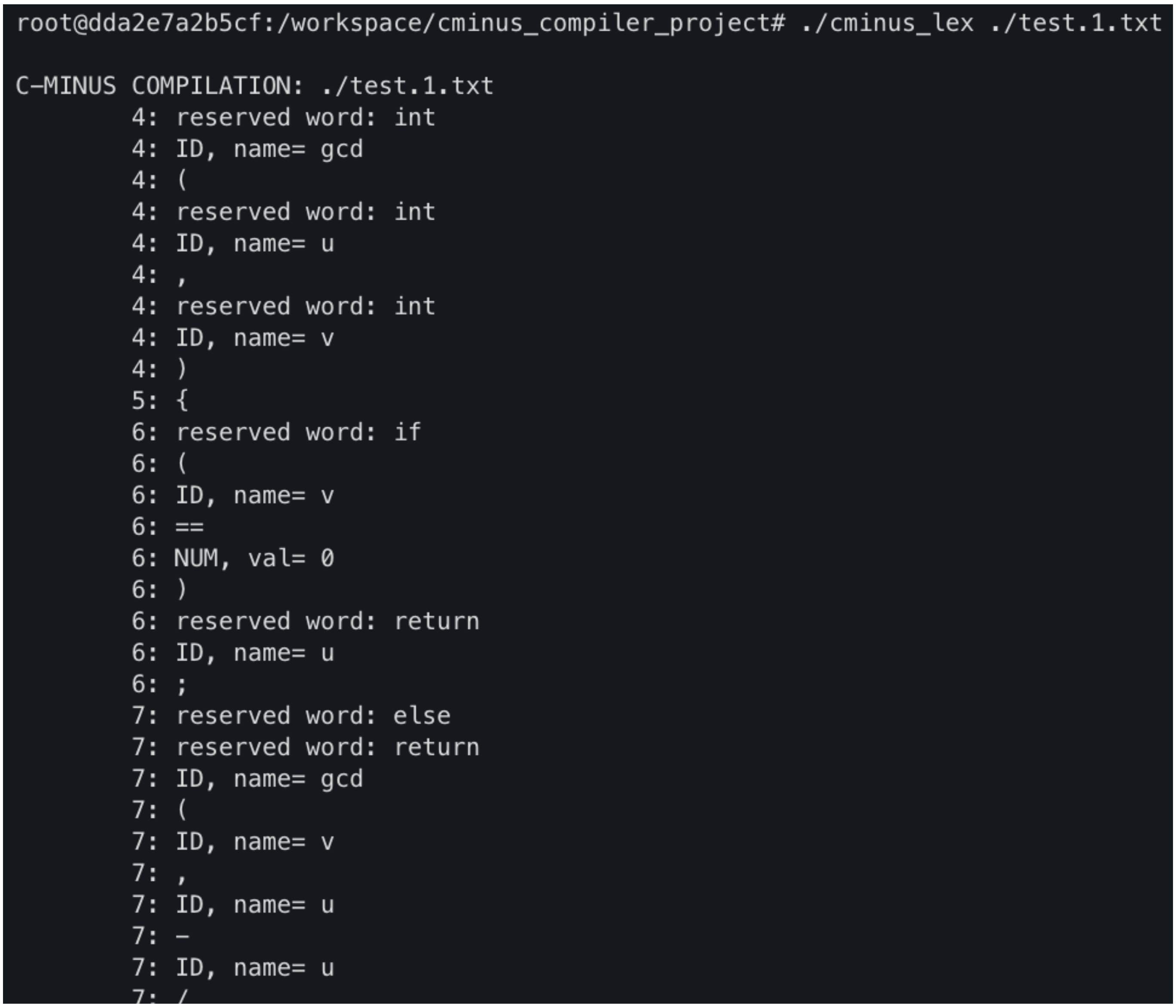
```
        18: ]
        18: )
        18: ;
        19: }
        20: }
        21: }
        22: EOF
```

Code Scanner made by FLEX

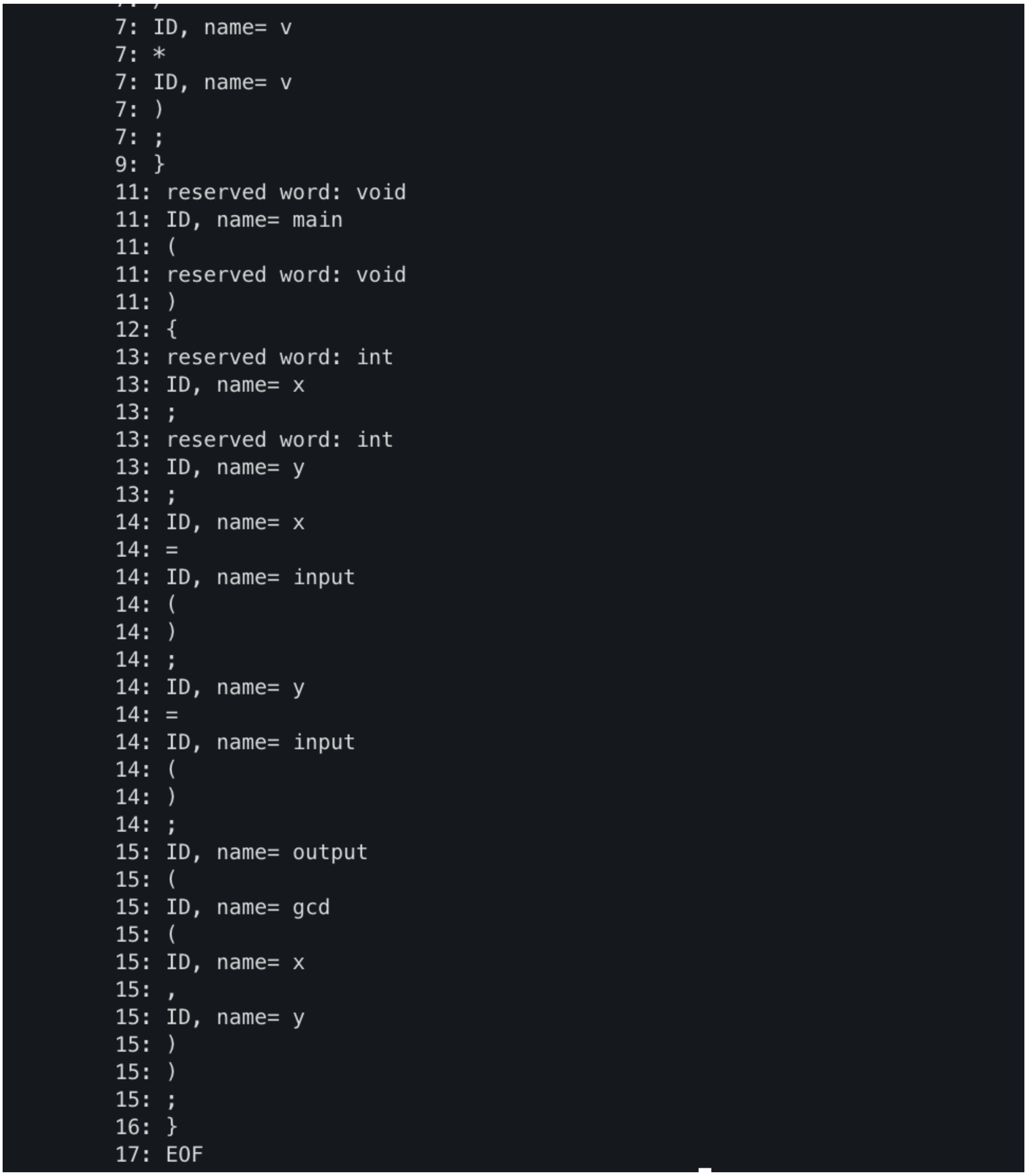
Test Case #1



```
root@dda2e7a2b5cf:/workspace/cminus_compiler_project# make cminus_lex && ./cminu
s_lex ./test.1.txt > ./myres.1.txt && diff -s result.1.txt myres.1.txt
flex -o lex.yy.c cminus.l
gcc  -W -Wall -c -o lex.yy.o lex.yy.c
lex.yy.c:1333:16: warning: ‘input’ defined but not used [-Wunused-function]
 1333 |     static int input  (void)
      |                ^~~~~
lex.yy.c:1290:17: warning: ‘yyunput’ defined but not used [-Wunused-function]
 1290 |     static void yyunput (int c, char * yy_bp )
      |                 ^~~~~~~
gcc  -W -Wall -o cminus_lex main.o util.o lex.yy.o -lfl
Files result.1.txt and myres.1.txt are identical
```

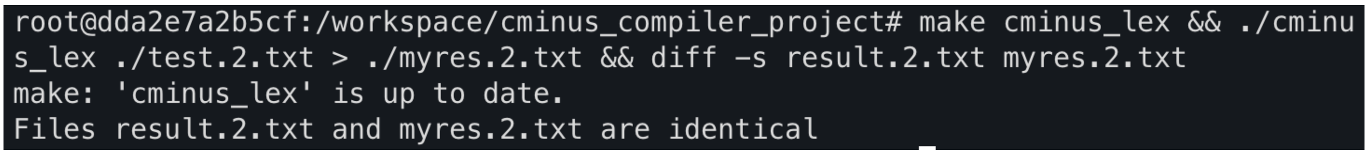


```
root@dda2e7a2b5cf:/workspace/cminus_compiler_project# ./cminus_lex ./test.1.txt

C-MINUS COMPILATION: ./test.1.txt
        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: /
```

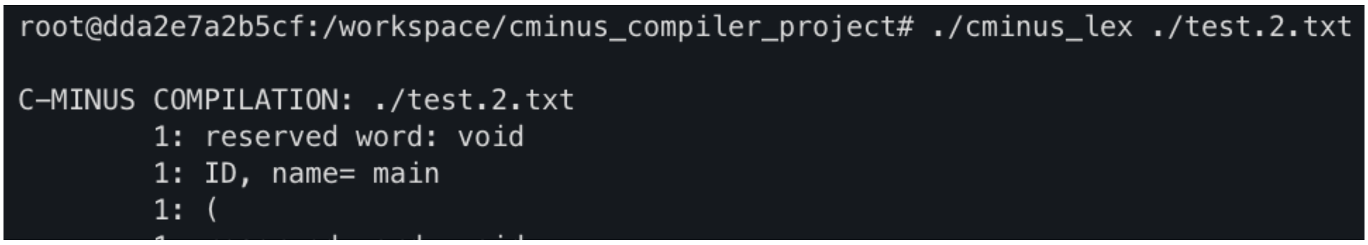


```
        7: ID, name= v
        7: *
        7: ID, name= v
        7: )
        7: ;
        9: }
        11: reserved word: void
        11: ID, name= main
        11: (
        11: reserved word: void
        11: )
        12: {
        13: reserved word: int
        13: ID, name= x
        13: ;
        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
```

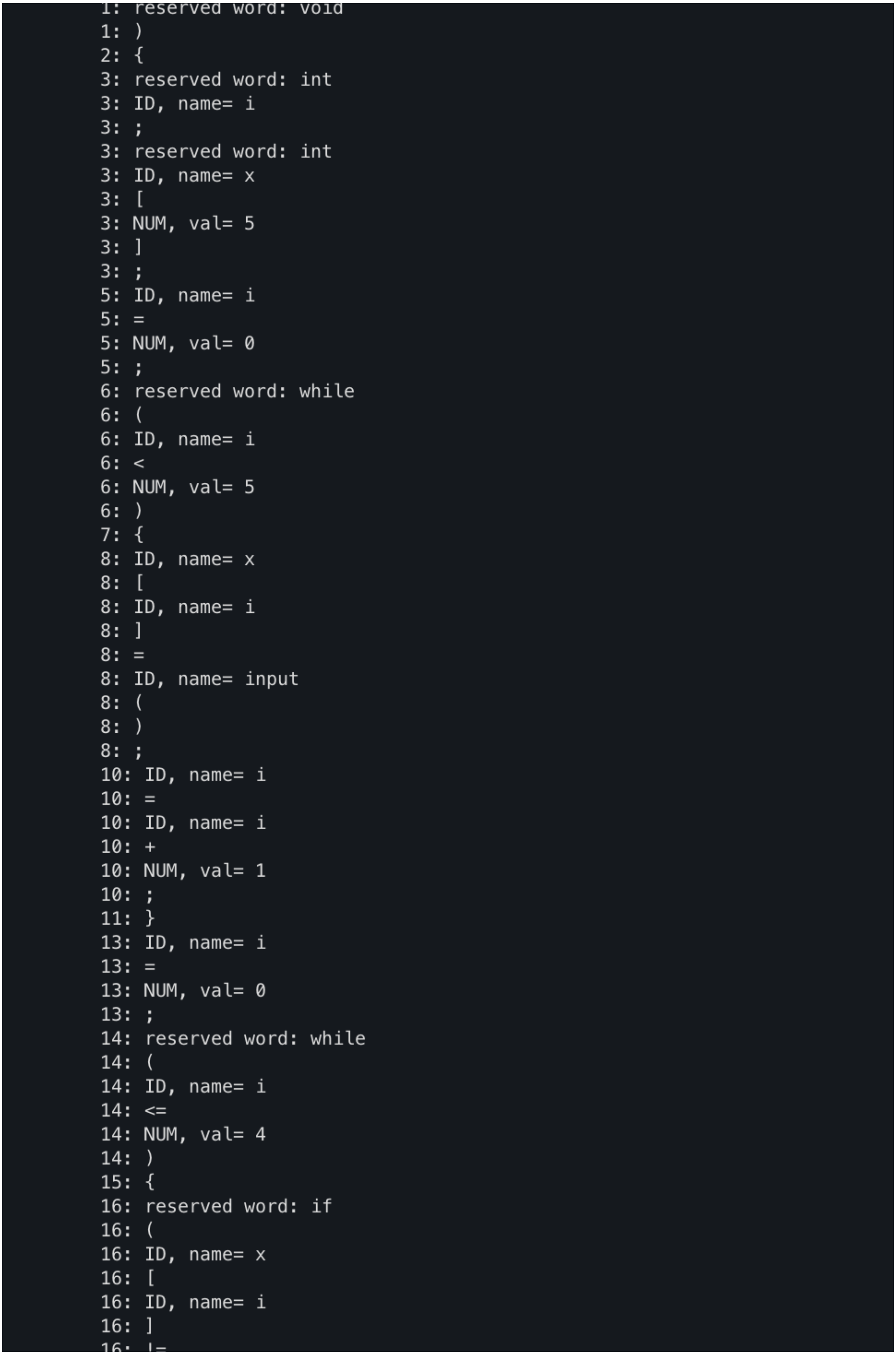
Test Case #2



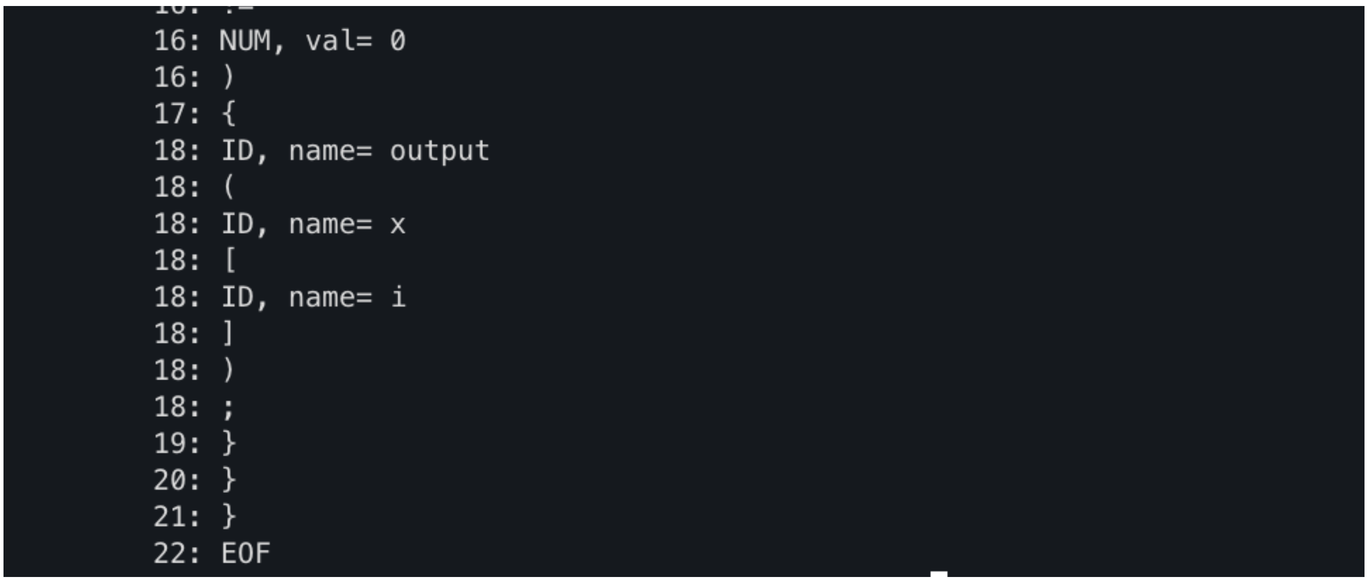
```
root@dda2e7a2b5cf:/workspace/cminus_compiler_project# make cminus_lex && ./cminu
s_lex ./test.2.txt > ./myres.2.txt && diff -s result.2.txt myres.2.txt
make: 'cminus_lex' is up to date.
Files result.2.txt and myres.2.txt are identical
```



```
root@dda2e7a2b5cf:/workspace/cminus_compiler_project# ./cminus_lex ./test.2.txt

C-MINUS COMPILATION: ./test.2.txt
        1: reserved word: void
        1: ID, name= main
        1: (
```



```
1: reserved word: void
1: )
2: {
3: reserved word: int
3: ID, name= i
3: ;
3: reserved word: int
3: ID, name= x
3: [
3: NUM, val= 5
3: ]
3: ;
5: ID, name= i
5: =
5: NUM, val= 0
5: ;
6: reserved word: while
6: (
6: ID, name= i
6: <
6: NUM, val= 5
6: )
7: {
8: ID, name= x
8: [
8: ID, name= i
8: ]
8: =
8: ID, name= input
8: (
8: )
8: ;
10: ID, name= i
10: =
10: ID, name= i
10: +
10: NUM, val= 1
10: ;
11: }
13: ID, name= i
13: =
13: NUM, val= 0
13: ;
14: reserved word: while
14: (
14: ID, name= i
14: <=
14: NUM, val= 4
14: )
15: {
16: reserved word: if
16: (
16: ID, name= x
16: [
16: ID, name= i
16: ]
16: !=
```



```
16: NUM, val= 0
16: )
17: {
18: ID, name= output
18: (
18: ID, name= x
18: [
18: ID, name= i
18: ]
18: )
18: ;
19: }
20: }
21: }
22: EOF
```