

Group ID – 44

PROJECT NO. - 31

GROUP MEMBERS –

I) LAXIT LALL (20114049)

II) MOHAMMED FAZIL AHMED
(20114056)

There is a verbose mode which activates when “-v” is included in command line. It shows tokens and grammar used in the given line.

Lexer function tokenizes the line and stores token in “store” array. The raw values related to tokens are stored in “var_store” array.

Segregate function in lexer segregates number, variables and others. Others are eliminated in parser function.

Parser function takes the “store” array and checks for the grammar. The grammar used is :

$S \rightarrow A; \mid \text{cout sep var } X;$

$X \rightarrow \text{sep var } X \mid E$

$A \rightarrow \text{int var eq } B$

$B \rightarrow C B'$

$B' \rightarrow /CB' \mid ,A \mid E$

$C \rightarrow (B) \mid \text{num} \mid \text{var}$

Here E is epsilon

To check the grammar, LL(1) parsing is used

Allote function allots value to the variables. If variable is equal to a number or a variable then its value is equal to the number or the value associated with other variable. If variable is equal to an expression then solve function is called.

Solve function takes var_store array and reduces it to single element array which contains the solution of the expression then the solution is stores in the variable.

Check_number function checks whether given string is number or not.

Assign_number function assign number to variable.

Check_var function checks whether given string is variable or not.

Find_terminator function find position of comma or eol.

Test cases:-

1. `int a=36 ,int b=6, int c= 3;`

`int x= a/(b/c), int y =(a/b)/c;`

`cout << x << y;`

output: 12 3

2. `int a= 10, int b=5,int c=5;`

`int x = a/(b/c), int y = (a/b)/c;`

`cout << a << b << c << x << y;`

output: 10 5 5 10 0

3. `int x = a+b+c;`


Output : invalid syntax

4. `int a = 120, int b=60, int c=30,int d=10, int e=2;`

`int x = a/b/c/d/e, int y= a/(b/(c/(d/e))), int z= ((a)/(b));`

`cout << x << y << z;`

output: 0 12 2

 C:\Users\admin\Documents\c++\journey\compiler_design_new.exe

```
int a =36,int b=6,int c=2;
int x= a/(b/c), int y= (a/b)/c;
cout << x << y;
12 3
```

```
int a =10,int b=5, int c=5;
int x= a/(b/c), int y= (a/b)/c;
cout << a << b << c << x << y;
10 5 5 10 0
```

```
int y = a;
cout << y;
10
```

```
int z = d;
value of d is unsigned
```

```
int x = a+b+c;
invalid syntax
```

Process returned 0 (0x0) execution time : 236.141 s
Press any key to continue.

```
laxit@laxit:~/Desktop$ ./a.out -v
int a =10,int b=5,int c=5;
tokens are: int var eq num comma int var eq num comma int var eq num eol
S->A;
A->int var eq B
pop
pop
pop
B->CB1
C->num
pop
B1-> ,A
pop
A->int var eq B
pop
pop
pop
B->CB1
C->num
pop
B1-> ,A
pop
A->int var eq B
pop
pop
pop
B->CB1
C->num
pop
B1->e
pop

int x = a/(b/c);
tokens are: int var eq var division lparen var division var rparen eol
S->A;
A->int var eq B
pop
pop
pop
B->CB1
C->var
pop
B1->/CB1
pop
C->(B)
pop
B->CB1
C->var
```

```
int x = a/(b/c);
tokens are: int var eq var division lparen var division var rparen eol
S->A;
A->int var eq B
pop
pop
pop
B->CB1
C->var
pop
B1->/CB1
pop
C->(B)
pop
B->CB1
C->var
pop
B1->/CB1
pop
C->var
pop
B1->e
pop
B1->e
pop

cout << x;
tokens are: cout seperator var eol
S-> cout sep var X;
pop
pop
pop
X->E
pop
10
```

```
laxit@laxit:~/Desktop$ g++ comp_design.cpp
laxit@laxit:~/Desktop$ ./a.out
int a=120,int b=60,int c=30,int d=10,int e=2;
int x = a/b/c/d/e,int y= a/(b/(c/(d/e))),int z =((a)/(b));
cout << x << y << z;
0 12 2
```