

Practical 03 - Lex & YACC Implementation - IT24103581

Simple Example: Hello World Command

Lex File (hello.l)

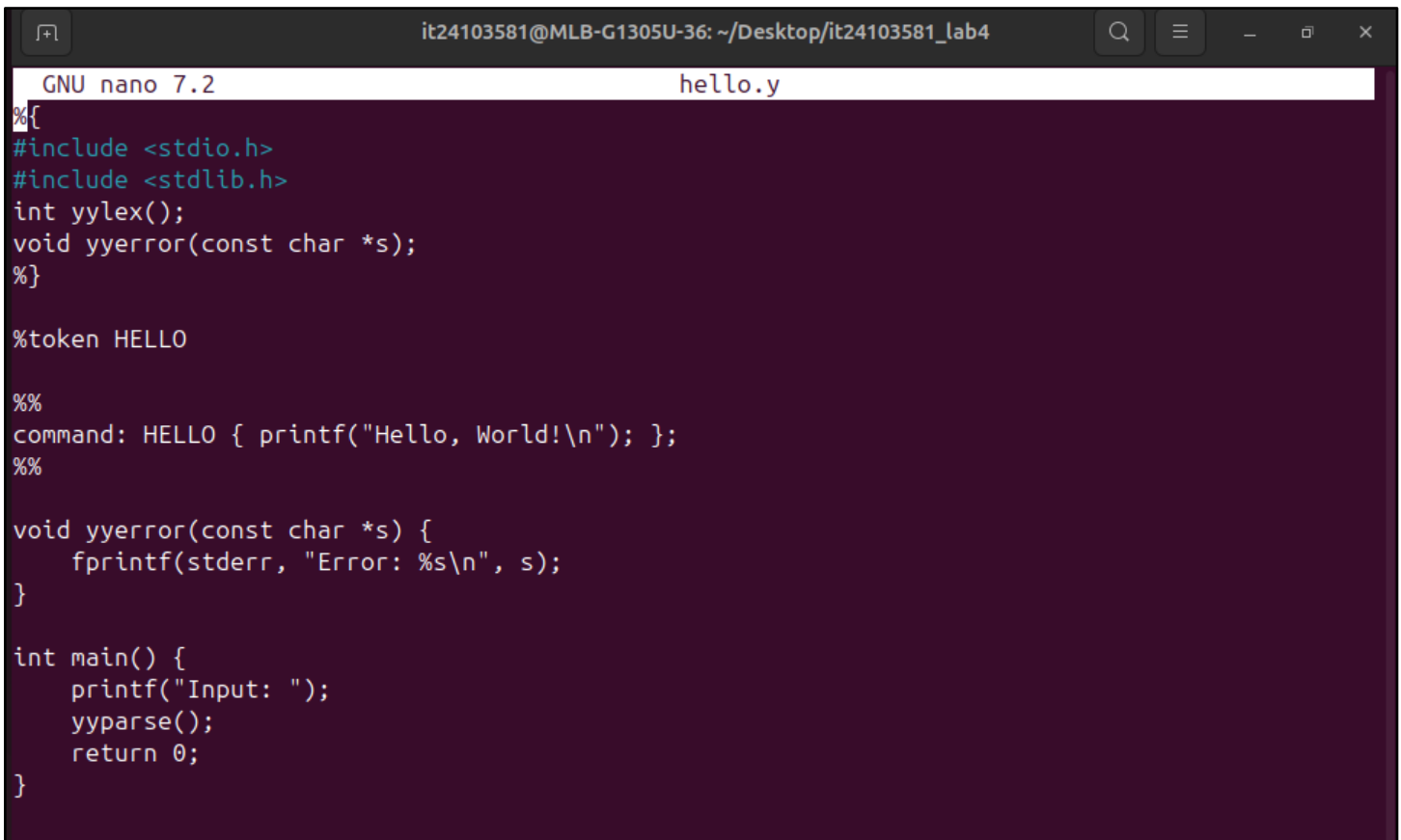


```
GNU nano 7.2 hello.l
%{
#include "y.tab.h"
%}

HELLO "hello"

%%
{HELLO} { return HELLO; }
[ \t\n] ; /* Ignore whitespace */
.      { printf("Invalid token: %s\n", yytext); }
%%
```

YACC File (hello.y)



```
GNU nano 7.2 hello.y
%{
#include <stdio.h>
#include <stdlib.h>
int yylex();
void yyerror(const char *s);
%}

%token HELLO

%%
command: HELLO { printf("Hello, World!\n"); };
%%

void yyerror(const char *s) {
    fprintf(stderr, "Error: %s\n", s);
}

int main() {
    printf("Input: ");
    yyparse();
    return 0;
}
```

Compilation and Execution:

```
it24103581@MLB-G1305U-36:~/Desktop/it24103581_lab4$ nano hello.l
it24103581@MLB-G1305U-36:~/Desktop/it24103581_lab4$ nano hello.y
it24103581@MLB-G1305U-36:~/Desktop/it24103581_lab4$ yacc -d hello.y
it24103581@MLB-G1305U-36:~/Desktop/it24103581_lab4$ lex hello.l
it24103581@MLB-G1305U-36:~/Desktop/it24103581_lab4$ gcc -o hello y.tab.c lex.yy.c -L/mingw64/lib -lfl
it24103581@MLB-G1305U-36:~/Desktop/it24103581_lab4$ ./hello
Input: hello
Hello, World!
```

Implementing a Simple Calculator

BNF Grammar

```
<expr> ::= <expr> + <term>
         | <expr> - <term>
         | <term>
```

```
<term> ::= <term> * <factor>
         | <term> / <factor>
         | <term> % <factor>
         | <factor>
```

```
<factor> ::= NUMBER
          | ( <expr> )
```

Lex File – calc.l

```
%{
#include "calc.tab.h"
#include <stdlib.h>
%}

%option noyywrap

%%
[0-9]+ { yylval.num = atoi(yytext); return NUMBER; }
[ \t] ;
\n { return '\n'; }

"+" { return '+'; }
"-" { return '-'; }
"*" { return '*'; }
"/" { return '/'; }
%" " { return '%'; }
"(" { return '('; }
")" { return ')'; }

. { printf("Invalid character\n"); }
```

```
%%
```

YACC File – calc.y

```
%{
#include <stdio.h>
#include <stdlib.h>

int yylex();
void yyerror(const char *s);
%}

%union {
    int num;
}

%token <num> NUMBER
%type <num> expr

%left '+' '-'
%left '*' '/' '%'

%%

input:
    | input line
    ;

line:
    expr '\n' { printf("Result = %d\n", $1); }
    | '\n'
    ;

expr:
    expr '+' expr { $$ = $1 + $3; }
    | expr '-' expr { $$ = $1 - $3; }
    | expr '*' expr { $$ = $1 * $3; }
    | expr '/' expr
    {
        if ($3 == 0) {
            printf("Error: Division by zero\n");
            $$ = 0;
        } else {
            $$ = $1 / $3;
        }
    }
    | expr '%' expr
    {
```

```

    if ($3 == 0) {
        printf("Error: Modulus by zero\n");
        $$ = 0;
    } else {
        $$ = $1 % $3;
    }
}
| '(' expr ')' { $$ = $2; }
| NUMBER      { $$ = $1; }
;

%%

```

```

int main() {
    printf("Simple Calculator Ready\n");
    yyparse();
    return 0;
}

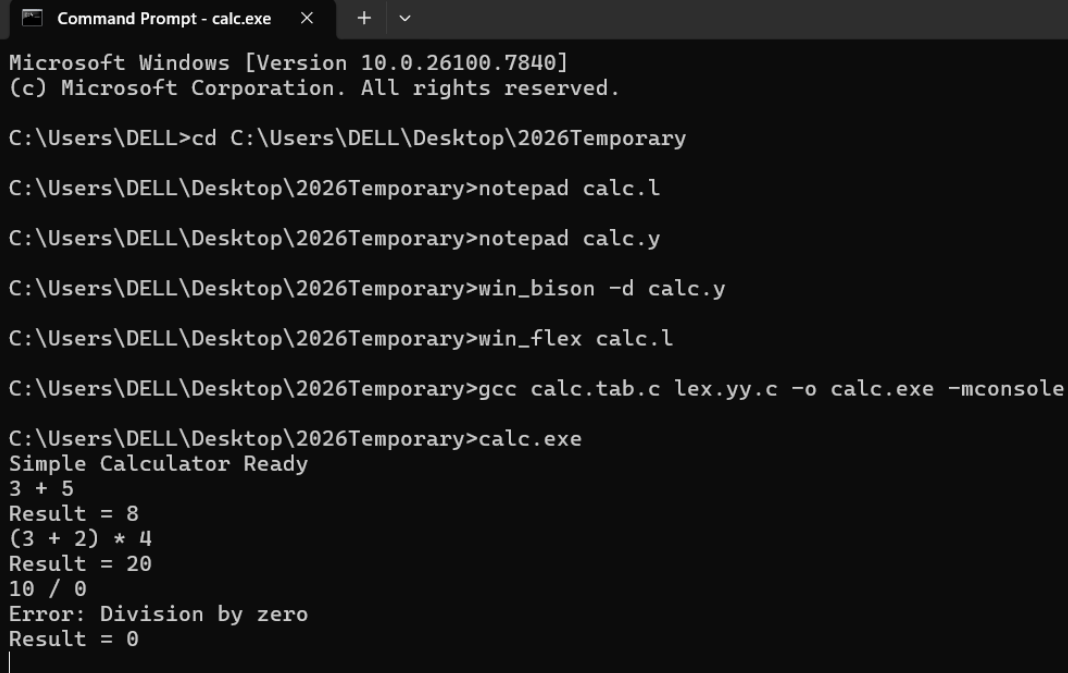
```

```

void yyerror(const char *s) {
    printf("Syntax Error\n");
}

```

Compilation and Execution:



```

Command Prompt - calc.exe  ×  +  ▾

Microsoft Windows [Version 10.0.26100.7840]
(c) Microsoft Corporation. All rights reserved.

C:\Users\DELL>cd C:\Users\DELL\Desktop\2026Temporary

C:\Users\DELL\Desktop\2026Temporary>notepad calc.l

C:\Users\DELL\Desktop\2026Temporary>notepad calc.y

C:\Users\DELL\Desktop\2026Temporary>win_bison -d calc.y

C:\Users\DELL\Desktop\2026Temporary>win_flex calc.l

C:\Users\DELL\Desktop\2026Temporary>gcc calc.tab.c lex.yy.c -o calc.exe -mconsole

C:\Users\DELL\Desktop\2026Temporary>calc.exe
Simple Calculator Ready
3 + 5
Result = 8
(3 + 2) * 4
Result = 20
10 / 0
Error: Division by zero
Result = 0
|

```

Section 3: Vehicle Control System

BNF Grammar

<command> ::= <component> <action>
 | <component> <action> <value>

<component> ::= engine
 | doors
 | lights
 | speed

<action> ::= start
 | stop
 | lock
 | unlock
 | on
 | off
 | set

<value> ::= NUMBER

Lex File - vehicle.l

```
%{
#include "vehicle.tab.h"
%}

%option noyywrap

%%
"engine" { return ENGINE; }
"doors"  { return DOORS; }
"lights" { return LIGHTS; }
"speed"  { return SPEED; }

"start" { return START; }
"stop"  { return STOP; }
"lock"  { return LOCK; }
"unlock" { return UNLOCK; }
"on"    { return ON; }
"off"   { return OFF; }
"set"   { return SET; }

[0-9]+ { yylval.num = atoi(yytext); return NUMBER; }

\n { return '\n'; }
[ \t] ;
. { printf("Invalid input\n"); }

%%
```

Lex File - vehicle.y

```
%{
#include <stdio.h>
#include <stdlib.h>

int yylex();
void yyerror(const char *s);
%}

%union {
    int num;
    char* str;
}

%token ENGINE DOORS LIGHTS SPEED
%token START STOP LOCK UNLOCK ON OFF SET
%token <num> NUMBER

%type <str> component action

%%

input:
    | input line
    ;

line:
    command '\n'
    ;

command:
    component action
    {
        printf("Command: %s %s\n", $1, $2);
    }
    | SPEED SET NUMBER
    {
        printf("Command: Speed Set %d\n", $3);
    }
    ;

component:
    ENGINE { $$ = "Engine"; }
    | DOORS { $$ = "Doors"; }
    | LIGHTS { $$ = "Lights"; }
    ;

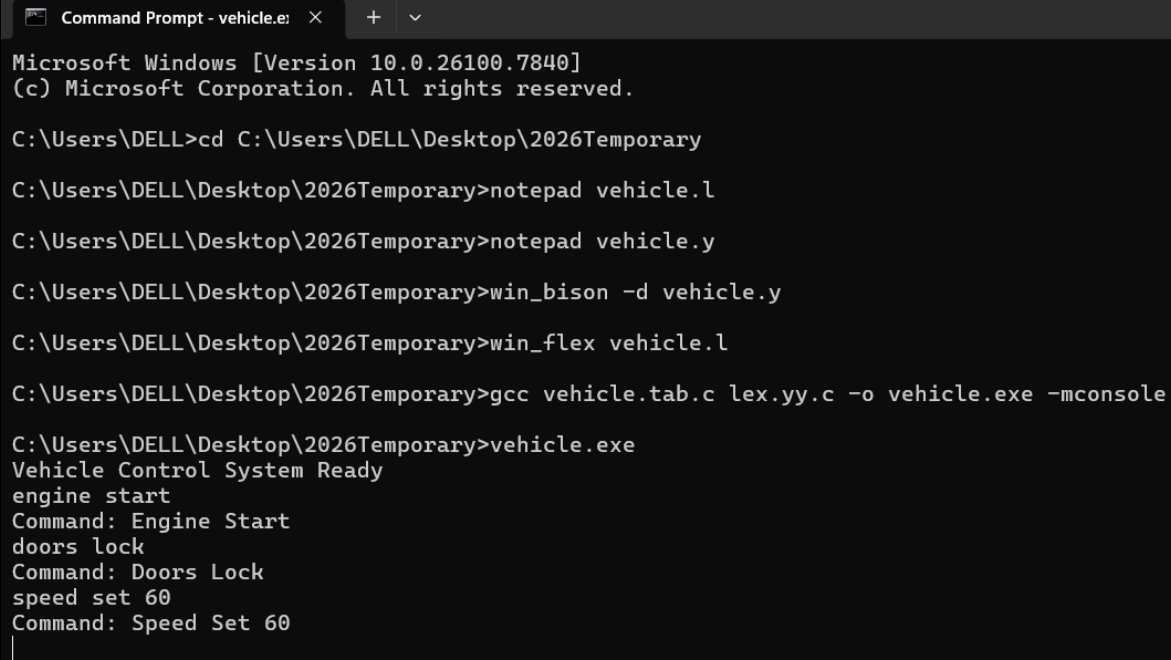
action:
    START { $$ = "Start"; }
    | STOP { $$ = "Stop"; }
    | LOCK { $$ = "Lock"; }
    | UNLOCK { $$ = "Unlock"; }
    | ON { $$ = "On"; }
    | OFF { $$ = "Off"; }
    ;

%%

int main() {
    printf("Vehicle Control System Ready\n");
    yyparse();
    return 0;
}
```

```
void yyerror(const char *s){
    printf("Syntax Error\n");
}
```

Compilation and Execution:



The screenshot shows a Windows Command Prompt window titled "Command Prompt - vehicle.e". The window displays the following commands and their outputs:

```
Microsoft Windows [Version 10.0.26100.7840]
(c) Microsoft Corporation. All rights reserved.

C:\Users\DELL>cd C:\Users\DELL\Desktop\2026Temporary
C:\Users\DELL\Desktop\2026Temporary>notepad vehicle.l
C:\Users\DELL\Desktop\2026Temporary>notepad vehicle.y
C:\Users\DELL\Desktop\2026Temporary>win_bison -d vehicle.y
C:\Users\DELL\Desktop\2026Temporary>win_flex vehicle.l
C:\Users\DELL\Desktop\2026Temporary>gcc vehicle.tab.c lex.yy.c -o vehicle.exe -mconsole
C:\Users\DELL\Desktop\2026Temporary>vehicle.exe
Vehicle Control System Ready
engine start
Command: Engine Start
doors lock
Command: Doors Lock
speed set 60
Command: Speed Set 60
|
```