CompilerProject Structure

Header

- Using import,
- Ex:

```
import stdio.h;
```

Comment

- Single line comment
  - Using `//`,
  - Ex:

```
int a = 25;
// this is single line comment

int b = 123; // also valid here
```

- Multi-line comment
  - Using `/* */`,
  - Support nested also,
  - Ex:

```
    int y = x sub 999;
    /*multi line comment
        possible to have
        /* nested also */
    */
```

Starting point of execution

- Defined as:
- Ex:

```
static void entryPoint(){
    // starts from here
}
```


Variable

- Supporeted Data type
  - `int`, `float`, `double`
- Declaration:
  - Using:

```
data_type var1, var2;
```

  - Ex:

```
int a;
int b;

int c,d,e;
```

- Initialization
  - Using:

```
data_type var1 = const_value, var2 = calc_value;
```

  - Ex:

```
int m = 102, n = m add 10;
// m = 102, n = 112
```

- Assignment
  - Using

```
var = const_value or calc_value;
```

  - Ex:

```
    int z;
    z = 100; // constant
    z = 10 add 20; // calculated value
```

- Extra:
    - Initialized to `0` by default,
    - Error if `duplicate variable` is declared,
    - Error if assign value to `undeclared variable`,
    - All variables are become global once declared,
    - Variable remains gloabl until `discarded` explicitly,
- Ex:

```
static void main(){
    // [] <--------------------------------

    int c = 1055;
    int d = 3443;

    // [c,d] <-----------------------------
    c = 1220;

    if( 100 lt d){
        c = 433;
        int dd = 343;
        // [c,d,dd] <----------------------
    }

    c = 4333;

    // [c,d,dd] <--------------- see dd here

    discard d;

    // [c,dd] <----------------------------
}
```

- Arithmatic operators
    - Addition`(add)`,
    - Subtraction `(sub)`,
    - Multiplication `(mul)`,
    - Division `(div)`,
    - Difference `(dif)`,
    - Remainder `(rem)`
    - Ex:

```
float f1 = 243, f2=11;
int sum = f1 add f2;
```



```
double b = 102.44;
b = b sub 2.44;
```

- Conditional operators
    - less than `(lt)`,
    - greater than `(gt)`,
    - equal `(eq)`,
    - not equals `(neq)`
    - less equal `(le)`,
    - greater equal `(ge)`
    - Ex:

```
if( 100 lt d){
    c = 433;
    int dd = 343;
}
```

- `justInCase` structure:
    - Equivalent to `if`,
    - Structure:

```
justInCase(vc @ vc){
    // body
}
```

`vc` = variable or constant, `@` = conditional operator

    - Ex:

```
if( 100 lt d){
    c = 433;
    int dd = 343;
}
```

    - vc = variable or constant
    - no else statment
- Looping
    - Structure:

```
till(vc @ vc){
    //body
}
```

- Ex:

```
int i=10;
till(i lt 100){
    //body
}
```

- Output to console
  - Using `println()`,
  - Structure:

```
println(vc1, vc2, ... vcn);
```

  - Comma(``,``) and space(`' '`) both are valid as separator,
  - Ex:

```
float f1 = 243, f2=11;

println(f1,f2);

double height = 25;
println("My height is: ",height "km");
```

  - Output is like

```
243.000 11.000  < - - - - - - - - - -

My height is: 25.000 km < - - - - - - - - - - -
```

- Overall example:

```
import stdio.h;
import test.h;

//starting point of program
static void entryPoint(){
```



```
double height = 25;
println("My height is: ",height "km");

//variable declaration
int a = 34;
float as=5;

println(a);

a = 255;

println(a);

discard a;

double b = 102.44, c= 123;

println(b, c);

b = b sub 2.44;
println(b);

justInCase( b gt 99){
   println(b," is larger than 100");
}

justInCase( b lt 500){
   println(b " is lees than 100");
}

justInCase(b eq 100){
   println(b, " is exactly 100");
}

//single line comment

int x = 1000; // initializing

int y = x sub 999;
/*
multi line comment
possible to have
/*
nested also
*/
*/

println(y);

float f1 = 243, /* also valid here */ f2=11;

println(f1,f2);
```

```
    int sum = f1 add f2;

    discard b,x,y,c,f1,f2;
    println(sum);

    int i=10;
    till(i lt 100){

    }


    int m = 102, n = m add 10;
    println(m,n);

    int z;
    z = 100;
    z = 10 add 20;
    println(z);

    // header(0), single(1), multi(2), var(3),
    // print(4), dis(5), calc_val(6), if(7),
    // till(8), cond(9)
    // int tempCounter[10];

}
```

Output


```
D:\Documents\COURSES\3.2\Labs\Compiler\CompilerProject>app
Imported stdio.h;
Imported test.h;
execution started

My height is: 25.000 km < - - - - - - - - - -

34.000  < - - - - - - - - - -

255.000  < - - - - - - - - - -

102.440 123.000  < - - - - - - - - - -

100.000  < - - - - - - - - - -

100.000  is larger than 100 < - - - - - - - - - -

100.000  is lees than 100 < - - - - - - - - - -

100.000  is exactly 100 < - - - - - - - - - -

1.000  < - - - - - - - - - -

243.000 11.000  < - - - - - - - - - -

add 243.000000 11.000000

254.000  < - - - - - - - - - -

loop start till

loop cond: (i lt 100)

loop start {: {

loop others:


loop end:    }

add 102.000000 10.000000

102.000 112.000  < - - - - - - - - - -

add 10.000000 20.000000

30.000  < - - - - - - - - - -

----------Printing all variables----------
height(double) 25.000000 -> as(float) 5.000000 -> sum(int) 254 -> i(int) 10 -> m(int) 102 -> n(int) 112 -> z(int) 30 ->

Total comment: 10
Program compiled successfully

---------- Summary ----------

Header:             2
```

```
Single line comment: 8
Multi line comment:  2
Variable:            7
println:             13
discard:             2
val_calculation:     5
justInCase:          3
till:                1
condition:           4
```

Run using

```
flex code.l
gcc lex.yy.c constant.c var_list.c -o app
app
```