**FLCD Lab. 8**

**FLEX file**

s

%{

#include <math.h>

%}

%option noyywrap

ID [A-Za-z][A-Za-z0-9]\*

NOT\_ID [^ ][A-Za-z][A-Za-z0-9]\*

INTEGER [+-]?(0|[1-9][0-9]\*)

STRING \"[A-Za-z0-9 \_\.,:;!\?\'#]\*\"

%%

{INTEGER} printf( "An integer: %s (%d)\n", yytext, atoi( yytext ) );

{STRING} printf( "A string: %s\n", yytext );

main|return|read|write|true|false|equals|lessEqual|lessThan|greaterEqual|greaterThan|if|else|for|int|string printf( "A keyword: %s\n", yytext );

{ID} printf( "An identifier: %s\n", yytext );

{NOT\_ID} printf( "Unrecognized identifier: %s\n", yytext );

"+"|"-"|"\*"|"/"|"%"|"=" printf( "An operator: %s\n", yytext );

";"|"{"|"}"|"("|")"|"["|"]" printf( "A separator: %s\n", yytext );

[ \t\n]+ /\* eat up whitespace \*/

. printf( "Unrecognized character: %s\n", yytext );

%%

int main(int argc, char\* argv[])

{

++argv, --argc; /\* skip over program name \*/

if ( argc > 0 )

yyin = fopen( argv[0], "r" );

else

yyin = stdin;

yylex();

}

**DEMO’s**

* **p1**

A keyword: int

A keyword: main

A separator: (

A separator: )

A separator: {

A keyword: int

An identifier: no

An operator: =

An integer: 0 (0)

A separator: ;

A keyword: int

An identifier: no2

An operator: =

An integer: 0 (0)

A separator: ;

A keyword: int

An identifier: no3

An operator: =

An integer: 0 (0)

A separator: ;

A keyword: int

An identifier: minimum

An operator: =

An integer: 0 (0)

A separator: ;

A keyword: read

An identifier: no

A separator: ;

A keyword: read

An identifier: no2

A separator: ;

A keyword: read

An identifier: no3

A separator: ;

An identifier: minimum

An operator: =

An identifier: no

A separator: ;

A keyword: if

A separator: (

An identifier: minimum

A keyword: lessThan

An identifier: no2

A separator: )

A separator: {

An identifier: minimum

An operator: =

An identifier: no2

A separator: ;

A separator: }

A keyword: if

A separator: (

An identifier: minimum

A keyword: lessThan

An identifier: no3

A separator: )

A separator: {

An identifier: minimum

An operator: =

An identifier: no3

A separator: ;

A separator: }

A keyword: write

An identifier: minimum

A separator: ;

A keyword: return

An integer: 0 (0)

A separator: ;

A separator: }

* **p1err**

A keyword: int

A keyword: main

A separator: (

A separator: )

A separator: {

A keyword: int

An identifier: no

An operator: =

Unrecognized character: ~

A separator: ;

A keyword: int

An identifier: no2

An operator: =

An integer: 0 (0)

A separator: ;

A keyword: int

An identifier: no3

An operator: =

An integer: 0 (0)

A separator: ;

A keyword: int

Unrecognized identifier: 0minimum

An operator: =

An integer: 0 (0)

A separator: ;

A keyword: read

An identifier: no

A separator: ;

A keyword: read

An identifier: no2

A separator: ;

A keyword: read

An identifier: no3

A separator: ;

Unrecognized identifier: 0minimum

An operator: =

An identifier: no

A separator: ;

A keyword: if

A separator: (

Unrecognized identifier: 0minimum

A keyword: lessThan

An identifier: no2

A separator: )

A separator: {

Unrecognized identifier: 0minimum

An operator: =

An identifier: no2

A separator: ;

A separator: }

A keyword: if

A separator: (

Unrecognized identifier: 0minimum

A keyword: lessThan

An identifier: no3

A separator: )

A separator: {

Unrecognized identifier: 0minimum

An operator: =

An identifier: no3

A separator: ;

A separator: }

A keyword: write

Unrecognized identifier: 0minimum

A separator: ;

A keyword: return

An integer: 0 (0)

A separator: ;

A separator: }

* **p2**

A keyword: int

A keyword: main

A separator: (

A separator: )

A separator: {

A keyword: int

An identifier: no

An operator: =

An integer: 0 (0)

A separator: ;

A keyword: int

An identifier: divisor

An operator: =

An integer: 0 (0)

A separator: ;

A keyword: read

An identifier: no

A separator: ;

A keyword: if

A separator: (

An identifier: no

A keyword: lessThan

An integer: 2 (2)

A separator: )

A separator: {

A keyword: write

An integer: 0 (0)

A separator: ;

A keyword: return

An integer: 0 (0)

A separator: ;

A separator: }

A keyword: if

A separator: (

An identifier: no

A keyword: equals

An integer: 2 (2)

A separator: )

A separator: {

A keyword: write

An integer: 1 (1)

A separator: ;

A keyword: return

An integer: 0 (0)

A separator: ;

A separator: }

A keyword: for

A separator: (

An identifier: divisor

An operator: =

An integer: 3 (3)

A separator: ;

An identifier: divisor

An operator: \*

An identifier: divisor

A keyword: lessEqual

An identifier: no

A separator: ;

An identifier: divisor

An operator: =

An identifier: divisor

An operator: +

An integer: 2 (2)

A separator: )

A separator: {

A keyword: if

A separator: (

An identifier: no

An operator: %

An identifier: divisor

A keyword: equals

An integer: 0 (0)

A separator: )

A separator: {

A keyword: write

An integer: 0 (0)

A separator: ;

A keyword: return

An integer: 0 (0)

A separator: ;

A separator: }

A separator: }

A keyword: write

An integer: 1 (1)

A separator: ;

A keyword: return

An integer: 0 (0)

A separator: ;

A separator: }

* **p3**

A keyword: int

A keyword: main

A separator: (

A separator: )

A separator: {

A keyword: int

An identifier: count

An operator: =

An integer: 0 (0)

A separator: ;

A keyword: int

An identifier: numbers

A separator: [

An integer: 100 (100)

A separator: ]

An operator: =

An integer: 0 (0)

A separator: ;

A keyword: int

An identifier: sum

An operator: =

An integer: 0 (0)

A separator: ;

A keyword: int

An identifier: index

An operator: =

An integer: 0 (0)

A separator: ;

A keyword: read

An identifier: count

A separator: ;

A keyword: for

A separator: (

An identifier: index

An operator: =

An integer: 0 (0)

A separator: ;

An identifier: index

A keyword: lessEqual

An identifier: count

A separator: ;

An identifier: index

An operator: =

An identifier: index

An operator: +

An integer: 1 (1)

A separator: )

A separator: {

A keyword: read

An identifier: numbers

A separator: [

An identifier: index

A separator: ]

A separator: ;

An identifier: sum

An operator: =

An identifier: sum

An operator: +

An identifier: numbers

A separator: [

An identifier: index

A separator: ]

A separator: ;

A separator: }

A keyword: write

An identifier: sum

A separator: ;

A keyword: return

An integer: 0 (0)

A separator: ;

A separator: }