CANDET ANDREI-GABRIEL GROUP 932 16 DEC 2020

Lex-Yacc lab 8

Source: https://github.com/cinnamonbreakfast/flcd/tree/main/lab8

Supporting emojis in code 😊

How to:

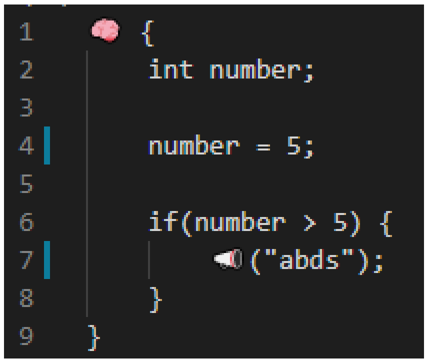
1. lex specif.lxi
2. gcc lex.yy.c -o exe -ll
3. ./exe < p1.pizza

specif.lxi

```
%{
#include <stdio.h>
#include <string.h>
int lines = 0;
%}

%option noyywrap
%option caseless

DIGIT     [0-9]
WORD      \"[a-zA-Z0-9]*\"
NUMBER    [+-]?[1-9][0-9]*|0$
CHARACTER  \'[a-zA-Z0-9]\'
CONST     {WORD}|{NUMBER}|{CHARACTER}
ID        [a-zA-Z][a-zA-Z0-9_]*

%%

array            {printf("Reserved word: %s\n", yytext);}
map              {printf("Reserved word: %s\n", yytext);}
const            {printf("Reserved word: %s\n", yytext);}
do               {printf("Reserved word: %s\n", yytext);}
else             {printf("Reserved word: %s\n", yytext);}
if               {printf("Reserved word: %s\n", yytext);}
int              {printf("Reserved word: %s\n", yytext);}
elif             {printf("Reserved word: %s\n", yytext);}
while            {printf("Reserved word: %s\n", yytext);}
for              {printf("Reserved word: %s\n", yytext);}
range            {printf("Reserved word: %s\n", yytext);}
class            {printf("Reserved word: %s\n", yytext);}
struct           {printf("Reserved word: %s\n", yytext);}
string           {printf("Reserved word: %s\n", yytext);}
float            {printf("Reserved word: %s\n", yytext);}
char             {printf("Reserved word: %s\n", yytext);}
boolean          {printf("Reserved word: %s\n", yytext);}
READ             {printf("Reserved word: %s\n", yytext);}
WRITE            {printf("Reserved word: %s\n", yytext);}
📢                {printf("Reserved word: %s\n", yytext);}
return           {printf("Reserved word: %s\n", yytext);}
fun              {printf("Reserved word: %s\n", yytext);}
```

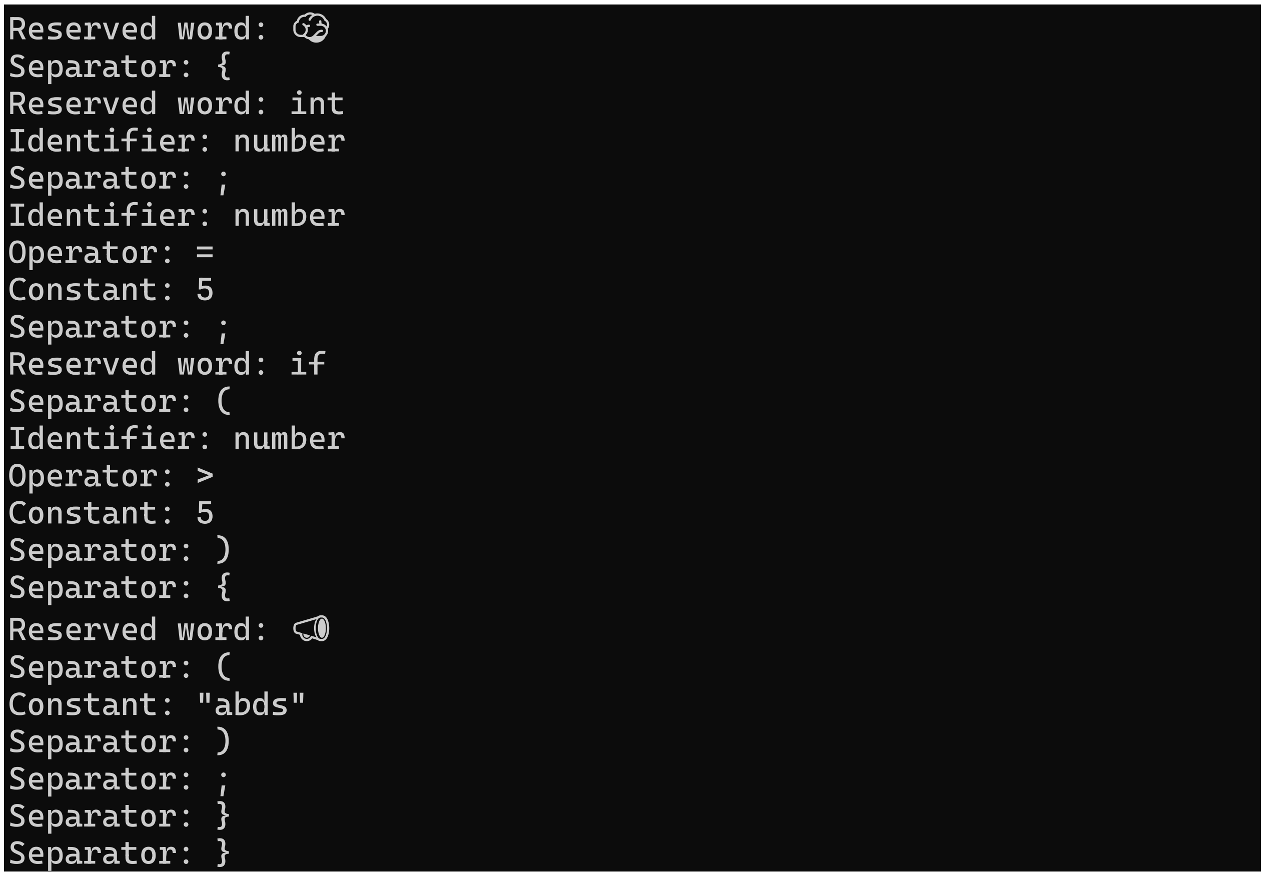
```
key                 {printf("Reserved word: %s\n", yytext);}
value               {printf("Reserved word: %s\n", yytext);}
main                {printf("Reserved word: %s\n", yytext);}
entry               {printf("Reserved word: %s\n", yytext);}
🧠        {printf("Reserved word: %s\n", yytext);}

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

{CONST}     {printf( "Constant: %s\n", yytext );}

":"         {printf( "Separator: %s\n", yytext );}
";"         {printf( "Separator: %s\n", yytext );}
","         {printf( "Separator: %s\n", yytext );}
"."         {printf( "Separator: %s\n", yytext );}
"{"         {printf( "Separator: %s\n", yytext );}
"}"         {printf( "Separator: %s\n", yytext );}
"("         {printf( "Separator: %s\n", yytext );}
")"         {printf( "Separator: %s\n", yytext );}
"["         {printf( "Separator: %s\n", yytext );}
"]"         {printf( "Separator: %s\n", yytext );}
"+"         {printf( "Operator: %s\n", yytext );}
"-"         {printf( "Operator: %s\n", yytext );}
"*"         {printf( "Operator: %s\n", yytext );}
"/"         {printf( "Operator: %s\n", yytext );}
"<"         {printf( "Operator: %s\n", yytext );}
">"         {printf( "Operator: %s\n", yytext );}
"<="    {printf( "Operator: %s\n", yytext );}
">="    {printf( "Operator: %s\n", yytext );}
"!="    {printf( "Operator: %s\n", yytext );}
"=="    {printf( "Operator: %s\n", yytext );}
"="      {printf( "Operator: %s\n", yytext );}
"!"         {printf( "Operator: %s\n", yytext );}
"?"         {printf( "Operator: %s\n", yytext );}
"==="   {printf( "Operator: %s\n", yytext );}


[ \t]+      {}
[\n]+ {lines++;}

[+-]?0[0-9]* {printf("Illegal constant at line %d 🙄\n", lines);}

[0-9~@#$%^][a-zA-Z0-9] {printf("Illegal identifier at line %d 😠\n", lines);}



\"[a-zA-Z0-9] {printf("Aoleu 😯 expected end of string on line %d\n", lines);
}

%%
```

Example of inputs:



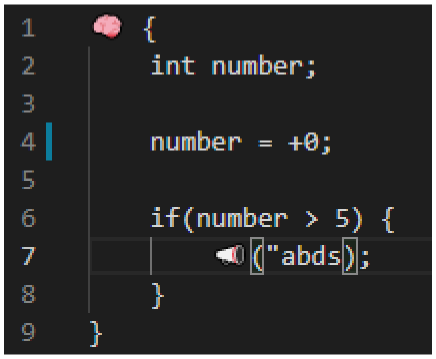
```
🧠 {
    int number;

    number = 5;

    if(number > 5) {
        📢("abds");
    }
}
```

This is a correct program, it has no errors and it should be parsed fine. Output:



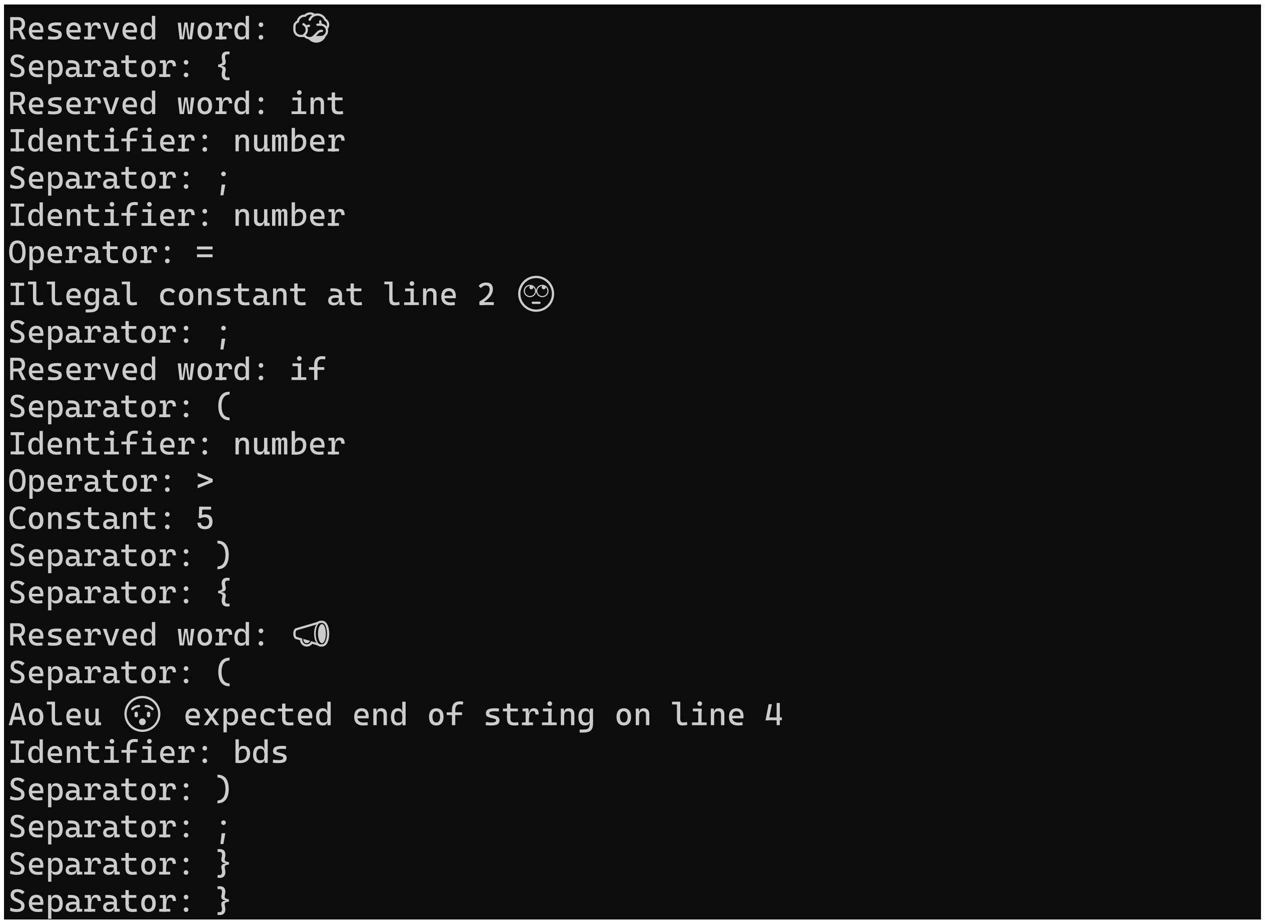
```
Reserved word: 🧠
Separator: {
Reserved word: int
Identifier: number
Separator: ;
Identifier: number
Operator: =
Constant: 5
Separator: ;
Reserved word: if
Separator: (
Identifier: number
Operator: >
Constant: 5
Separator: )
Separator: {
Reserved word: 📢
Separator: (
Constant: "abds"
Separator: )
Separator: ;
Separator: }
Separator: }
```

Program number 2, on the other hand



```
1  🧠 {
2      int number;
3
4      number = +0;
5
6      if(number > 5) {
7          📣("abds);
8      }
9  }
```

Contains an error on line 4 (invalid constant err) and another one on line 7 (end of string err). Output:



```
Reserved word: 🧠
Separator: {
Reserved word: int
Identifier: number
Separator: ;
Identifier: number
Operator: =
Illegal constant at line 2 🙄
Separator: ;
Reserved word: if
Separator: (
Identifier: number
Operator: >
Constant: 5
Separator: )
Separator: {
Reserved word: 📣
Separator: (
Aoleu 😯 expected end of string on line 4
Identifier: bds
Separator: )
Separator: ;
Separator: }
Separator: }
```