LEX lab

Github url:

https://github.com/Akitektuo/University/tree/master/3rd%20year/FLCT/lab/Lab9

compile.sh

```
flex specif.lxi
gcc lex.yy.c -o exe -ll
./exe < inputFiles/p1.txt > outputFiles/output.txt
cat outputFiles/output.txt
```

specif.lxi

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

int lines = 0;

struct Element {
    char key[100];
    int position;
} element;

char symbolTable[100][100];
int syntaxTableSize = 0;
int programInternalFormSize = 0;
struct Element programInternalForm[100];

void addPredefinedToPif(char* text) {
    strcpy(element.key, text);
    element.position = -1;
    programInternalForm[programInternalFormSize++] = element;
}

int addToSymbolTable(char* text) {
    for (int i = 0; i < syntaxTableSize; ++i) {
        if (strcmp(symbolTable[i], text) == 0)
            return i;
    }
    strcpy(symbolTable[syntaxTableSize], text);
```

```
    return syntaxTableSize++;
}

void addUserDefinedToPif(char* text) {
    strcpy(element.key, text);
    element.position = addToSt(text);
    programInternalForm[programInternalFormSize++] = element;
}
%}

%option noyywrap
%option caseless
RESERVED
\b(int|bool|string|input|print|when|otherwise|in|while|each)\b
BOOL        \b(false|true)\b
STRING      \"(\\\"|[^\"])*\"
NUMBER      \b(0|([+\-]?[1-9]\d*))\b
CONST       {NUMBER}|{BOOL}|{STRING}
ID          [a-zA-Z][0-9a-zA-Z_]*
OPERATOR
"=="|"="|"\+\+"|"\+="|"\+"|"--"|"-="|"-"|"\*="|"\*"|"/="|"/"|"%="|"%"
|"!="|"<="|"<"|">="|">"|"!"|"&="|"\|="|"&"|"\|"
SEPARATOR   [()}{\]\[ "]

%%

{RESERVED}  { printf("Reserved: %s\n", yytext);
addPredefinedToPif(yytext); }

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

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

{OPERATOR}  { printf("Operator: %s\n", yytext);
addPredefinedToPif(yytext); }

{SEPARATOR} { printf("Separator: %s\n", yytext);
addPredefinedToPif(yytext); }

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

. printf("Error on line %d\n", lines + 1);
```

```
%%

void printSymbolTable() {
    printf("Symbol table: \n");
    for (int i = 0; i < syntaxTableSize; ++i) {
        printf("\t%d. %s\n", i, symbolTable[i]);
    }
    printf("\n");
}

void printProgramInternalForm() {
    printf("Program Internal Form: \n");
    for (int i = 0; i < programInternalFormSize; ++i) {
        printf("\t%d -> %s\n", pif[i].position, pif[i].key);
    }
    printf("\n");
}

int main(int argc, char** argv) {
    yyin = stdin;
    yylex();
    printf("\nParsed %d lines\n\n", lines);
    printProgramInternalForm();
    printSymbolTable();
}
```