Gabriel Santos da Silva. Compiladores.

FLEX:

- Inicialmente, instale o flex na sua maquina:

  sudo apt-get install flex

- Será necessario a criação de um arquivo FLEX.l

  O arquivo contem as seguintes seções: Três partes, separadas por linhas que começam com %%:

[Definições]
%%
[Regras]
%%
[Código do usuário] //aqui contem somente a função principal chama yylex(), que é a função gerada pelo Flex para realizar a análise léxica.

- Com o arquivo montado devemos usar:

  1) Gerar o arquivo C:

  flex FLEX.l

  2) Compilar código C:

  gcc lex.yy.c -o lex_cod -lfl

  3) Executar:

  ./lex_cod < codigo.txt

  OBS: codigo.txt contem um exemplo real de um codigo para esse trabalho, contendo declarações, atribuições, IF, else e etc.

  NÃO é necessario criar um arquivo codigo.txt, podemos simplesmente usar ./lex_cod e entrar com sequencia de tokens para avaliação.

- Saída:

A saida deve ser os tokens analisados pelo analisador LEXICO Flex.

BISON:

- Inicialmente, instale o Bison na sua maquina:

sudo apt-get install bison

- Será necessario a criação de um arquivo Bison.y

  O arquivo contem as seguintes seções:

%{
Prólogo
%}
Declarações do Bison
%%
Regras gramaticais
%%
Epílogo

ATENÇÃO:
O Bison trabalha em conjunto com o Flex, com base no arquivo flex.l anterior, devemos criar o flex_mod.l com as seguintes alterações:

```
%{
#include "y.tab.h" //arquivo y.tab.h
extern int yylineno; //para contar as linhas
%}

// {...}resto do codigo {...}

"int"       { return TIPO_INT; } //agora são returns.

[ \t]+ ; //nao ignora mais as quebras de linhas.

\n { yylineno++; } //contar as linhas.

// {...}resto do codigo {...}
```

RETIRAR A SEÇÃO "%%[Código do usuário]" COM O MAIN (UNICO MAIN PRESENTE NESSA ETAPA ESTARÁ PRESENTE NO BISON.

- Com o arquivo flex_mod.l montado devemos usar:

1) Gerar o arquivo C:

flex flex_mod.l

yacc -d Bison.y //-d server para gerar o arquivo y.tab.h com as definições dos tokens. obs: warnings podem ocorrer...

2) Compilar código C:

gcc lex.yy.c y.tab.c -o compilador //obs: warnings podem ocorrer...

3) Executar:

./compilador cod_exemplo

OBS: cod_exemplo contem um exemplo real de um codigo para esse trabalho, contendo declarações, atribuições, IF, else e etc.

NÃO é necessario criar um arquivo cod_exemplo, podemos simplesmente usar ./compilador e entrar com sequencia de codigos para avaliação.

- Saída:

Caso o codigo sejá valido, a saida é "Análise sintática finalizada com sucesso.", porem ao inserir um erro ao codigo como "while K(x < 20)" deve retornar uma saida:

Erro na linha 'N': 'erro'
Análise sintática concluída com erros.