UNIVERSIDADE FEDERAL DE VIÇOSA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

PEDRO CARDOSO DE CARVALHO MUNDIM - 3877

TRABALHO PRÁTICO 0 - COMPILADORES (CCF 441)
EXPLORANDO O FLEX

Trabalho Prático da disciplina Compiladores - CCF 441, do curso de Ciência da Computação da Universidade Federal de Viçosa - Campus Florestal

Professor: Daniel Mendes Barbosa

FLORESTAL
Junho 2022

SUMÁRIO

1. Introdução

Este trabalho consiste em explorar a ferramenta flex (um lex mais recente), a fim de aplicar alguns conceitos estudados durante a disciplina até o momento. Para isso, foi pedido a criação de dois arquivos.l, em que o primeiro deles consiste em um tutorial descrito pelo professor, no qual o mesmo definiu os tokens, e o segundo em um arquivo criado pelo aluno, contendo também os tokens criados pelo mesmo.

Para o primeiro arquivo, denominado lex.l, tem-se o teste dado pelo professor e um teste criado pelo aluno (arquivos de texto). Para o segundo arquivo, denominado lex2.l, criado pelo aluno, tem-se apenas o teste também criado pelo aluno. Todos esses arquivos (.l e testes), estarão em um .zip na entrega deste trabalho, juntamente com o relatório. Além disso, também estarão presentes os arquivos gerados após a execução dos testes.

2. Desenvolvimento

2.1 Testes de Execução e Resultados - lex.l

Agora serão mostrados os testes de execução e resultados para o arquivo lex.l, que contém os tokens definidos pelo professor, que foram os seguintes (as explicações de como eles são reconhecidos encontra-se junto de cada um):

- **Inteiro** - qualquer número inteiro, com um ou mais algarismos.
- **Positivo** - qualquer número positivo, então é necessário que [+] seja passado antes do algarismo.
- **Negativo** - qualquer número negativo, então é necessário que [-] seja passado antes do(s) algarismo(s).
- **Decimal** - qualquer número (positivo ou negativo), com um ou mais algarismos, seguido por um ponto final, seguido por um ou mais algarismos.
- **Placa** - seguindo o padrão das placas antigas, tem-se que são formadas por três letras, seguido por um hífen, seguido por quatro algarismos.
- **Palavra** - qualquer conjunto que contenha letras.
- **Telefone** - seguindo o padrão antigo dos telefone, tem-se que são formados por quatro algarismos, seguido por um hífen, seguido por mais quatro algarismos.
- **Nome** - os nomes podem ser formados por uma ou mais palavras, de acordo com a definição da formação de palavras dada anteriormente.

Será executada a entrada (testeDaniel.txt) definida pelo docente e mostrada sua saída. A entrada dada é mostrada na Figura 1.

875878 -3355456 abc5464 abc-5464 ABC-5464 453-2345 9486-0847
Daniel Mendes Barbosa 32.345 Palavra Qualquer 3567-3224
Daniel Mendes Barbosa Daniel Mendes Barbosa Menezes200

Figura 1: Entrada definida pelo professor para o lex.l.

Para executar o arquivo lex.l, basta entrar na pasta nomeada como Lex 1, e executar os seguintes comandos no terminal (Figura 2):

```
pedro@pedro-ubuntu:~/Documentos/Compiladores/Lex 1$ flex lex.l
pedro@pedro-ubuntu:~/Documentos/Compiladores/Lex 1$ gcc lex.yy.c
pedro@pedro-ubuntu:~/Documentos/Compiladores/Lex 1$ ./a.out
Digite o nome do arquivo:
testeDaniel.txt
```

Figura 2: Comandos para a execução do arquivo lex.l.

Antes de mostrar a saída, observe o código principal do arquivo lex.l, em que foi utilizado expressões regulares para facilitar a compreensão do mesmo, de acordo com as definições discutidas anteriormente, conforme a Figura 3.

```
delim       [ \t\n]
ws        {delim}+

inteiro [0-9]+
positivo [+][0-9]+
negativo [-][0-9]+
decimal [+|-]*[0-9]+[.][0-9]+
placa [A-Z][A-Z][A-Z][-][0-9][0-9][0-9][0-9]+
palavra [a-zA-Z]+
telefone [0-9][0-9][0-9][0-9][-][0-9][0-9][0-9][0-9]+
nome {palavra}[" "]{palavra}[" "]{palavra}[" "]*{palavra}*

%%


{ws}        {/*nenhuma acao e nenhum retorno*/}


{inteiro} {printf("Foi encontrado um numero inteiro positivo. LEXEMA: %s\n", yytext);}
{positivo} {printf("Foi encontrado um numero inteiro positivo. LEXEMA: %s\n", yytext);}
{negativo} {printf("Foi encontrado um numero inteiro negativo. LEXEMA: %s\n", yytext);}
{decimal} {printf("Foi encontrado um numero com parte decimal. LEXEMA: %s\n", yytext);}
{placa} {printf("Foi encontrado uma placa. LEXEMA: %s\n", yytext);}
{palavra} {printf("Foi encontrado uma palavra. LEXEMA: %s\n", yytext);}
{telefone} {printf("Foi encontrado um telefone. LEXEMA: %s\n", yytext);}
{nome} {printf("Foi encontrado um nome proprio. LEXEMA: %s\n", yytext);}
```

Figura 3: Código principal - expressões regulares do lex.l.

O resultado da execução do código anterior, de acordo com essa primeira entrada, é mostrado na Figura 4.

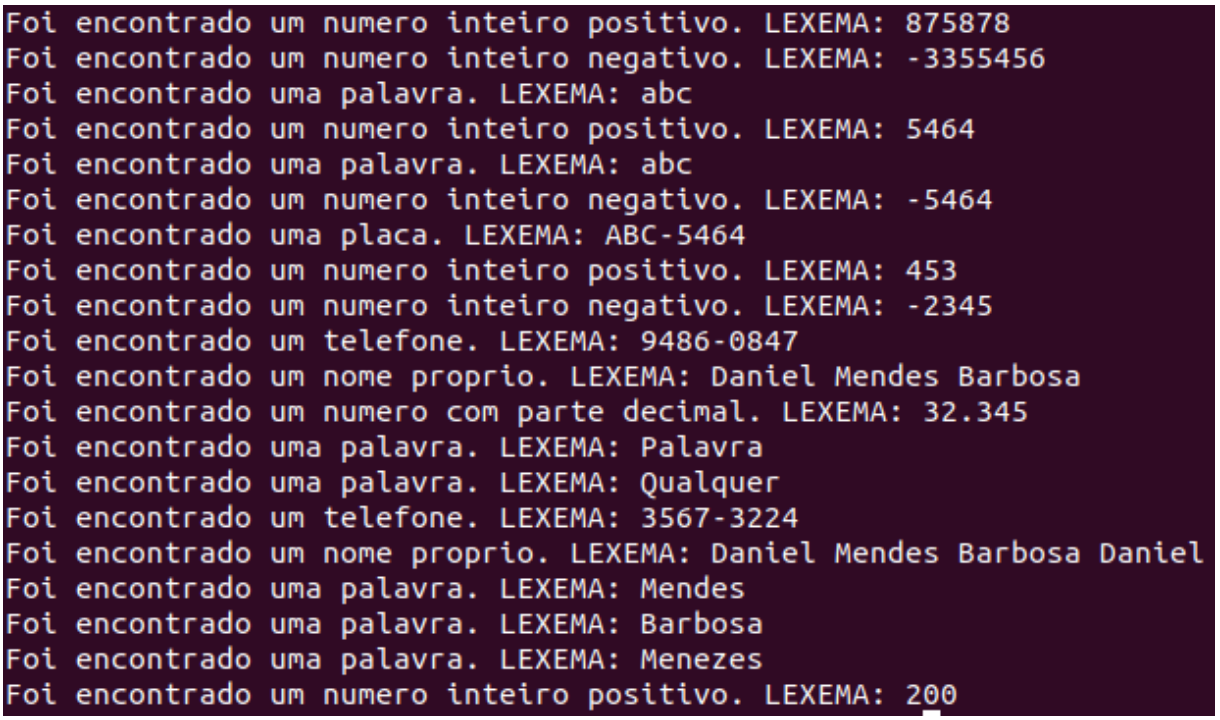

Figura 4: Resultado para a entrada definida pelo professor para o lex.l.

Agora, será exposto a execução do mesmo arquivo lex com a entrada criada pelo aluno (entradaMinha.txt), a qual é mostrada na Figura 5.

876478 -3355456 abc5464 -44465456 5478932465 4843 pedro carvalho
PeDRo Carvalho abc-5498 RPG-8512 453-3241 94486-045847
games +6465465-4564 32.321'5 1226-0336 ZElda Pedro 3877

Figura 5: Entrada definida pelo aluno para o lex.l.

O resultado desta execução é mostrado na Figura 6. Da mesma forma que no exemplo do professor, foram identificados números, palavras, telefones, dentre outros.

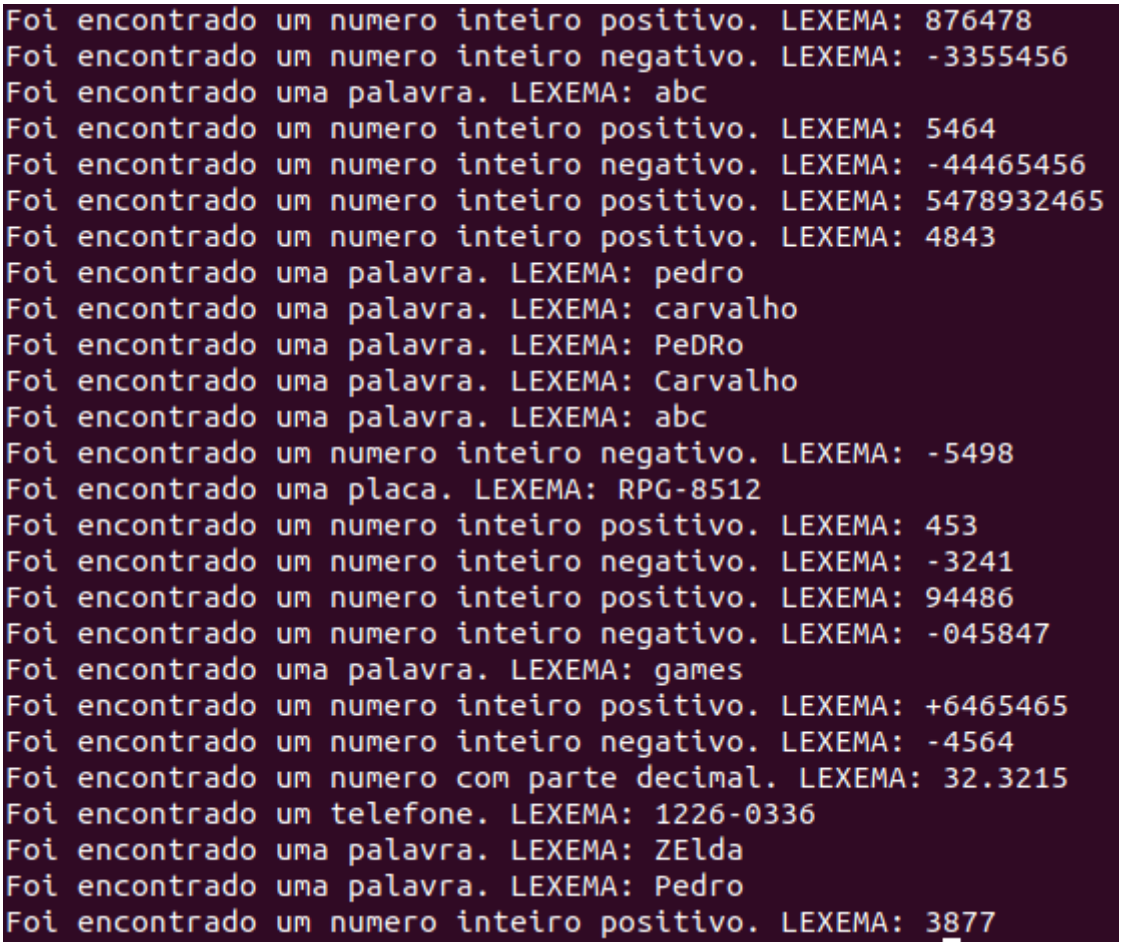

Figura 6: Resultado para a entrada definida pelo aluno para o lex.l.

2.2 Testes de Execução e Resultados - lex2.l

Agora serão mostrados os testes de execução e resultados para o arquivo lex2.l, que contém os tokens definidos pelo aluno. Os tokens pensados foram os seguintes (as explicações de como eles são reconhecidos encontra-se junto de cada um):

- **CPF** - segue o padrão de três algarismos, um ponto, três algarismos, um ponto, três algarismos, um hífen e dois algarismos.
- **Matrícula** - seguindo o padrão da UFV, tem-se as letras EF seguidas por cinco algarismos, sendo o primeiro deles um 0.
- **Email** - é necessário que reconheça qualquer algarismo ou letra, além da possibilidade de ponto ou subtração. Em seguida, é necessário um @, seguido do nome do domínio. Por fim, pensando em e-mails brasileiros, tem-se opções do final com ".com.br", ou mesmo ".br". Dessa forma, é utilizado o fecho de Kleene para aceitar essas regras.
- **Coeficiente** - seguindo o padrão da UFV, tem-se dois algarismos, seguidos por um ponto final, seguido por mais um algarismo.

- **Código de Disciplina** - seguindo o padrão da UFV, tem-se três letras, seguido por um hífen, seguido por três algarismos.
- **Nome de Disciplina** - definem-se as regras para geração de palavras (a mesma regra utilizada no primeiro arquivo lex). Os nomes das disciplinas serão formados por uma ou até quatro palavras nesse caso.

Será executada a entrada (entrada.txt) definida pelo aluno e mostrado sua saída. A entrada dada é mostrada na Figura 7.

CCF-441 CCF-355 EF05568 98.1 76.5
pcardosomundim@gmail.com danielmendes@ufv.br 876.345.234-97
EF03877 23.4 654.987.231-09 67.9 LEF-100
MAF-243 Calculo Diferencial e Integral

Figura 7: Entrada definida pelo aluno para o lex2.l.

Para executar o arquivo lex2.l, basta entrar na pasta nomeada como Lex 2, e executar os seguintes comandos no terminal (Figura 8):

```
pedro@pedro-ubuntu:~/Documentos/Compiladores/Lex 2$ flex lex2.l
pedro@pedro-ubuntu:~/Documentos/Compiladores/Lex 2$ gcc lex.yy.c
pedro@pedro-ubuntu:~/Documentos/Compiladores/Lex 2$ ./a.out
Digite o nome do arquivo:
entrada.txt
```

Figura 8: Comandos para a execução do arquivo lex2.l.

Antes de mostrar a saída, observe o código principal do arquivo lex2.l, em que foi utilizado expressões regulares para facilitar a compreensão do mesmo, de acordo com as definições discutidas anteriormente, conforme a Figura 9.

```
delim       [ \t\n]
ws        {delim}+


palavra [a-zA-Z]+
cpf [0-9][0-9][0-9][.][0-9][0-9][0-9][.][0-9][0-9][0-9][-][0-9][0-9]
matricula [E][F][0][0-9][0-9][0-9][0-9]
email [a-zA-Z|0-9|.|_]+[@][a-zA-Z]+[.][a-zA-Z][a-zA-Z][a-zA-Z]*[.]*[a-zA-Z]*[a-zA-Z]*
coeficiente [0-9][0-9][.][0-9]
codigo_disciplina [A-Z][A-Z][A-Z][-][0-9][0-9][0-9]
nome_disciplina {palavra}[" "]{palavra}[" "]{palavra}[" "]*{palavra}


%%


{ws}        {/*nenhuma acao e nenhum retorno*/}


{cpf} {printf("Foi encontrado um cpf. LEXEMA: %s\n", yytext);}
{matricula} {printf("Foi encontrado uma matrícula. LEXEMA: %s\n", yytext);}
{email} {printf("Foi encontrado um email. LEXEMA: %s\n", yytext);}
{coeficiente} {printf("Foi encontrado um coeficiente. LEXEMA: %s\n", yytext);}
{codigo_disciplina} {printf("Foi encontrado um código de disciplina. LEXEMA: %s\n", yytext);}
{nome_disciplina} {printf("Foi encontrado um nome de disciplina. LEXEMA: %s\n", yytext);}
```

Figura 9: Código principal - expressões regulares do lex2.l.

O resultado da execução do código anterior é mostrado na Figura 10.

```
Foi encontrado um código de disciplina. LEXEMA: CCF-441
Foi encontrado um código de disciplina. LEXEMA: CCF-355
Foi encontrado uma matrícula. LEXEMA: EF05568
Foi encontrado um coeficiente. LEXEMA: 98.1
Foi encontrado um coeficiente. LEXEMA: 76.5
Foi encontrado um email. LEXEMA: pcardosomundim@gmail.com
Foi encontrado um email. LEXEMA: danielmendes@ufv.br
Foi encontrado um cpf. LEXEMA: 876.345.234-97
Foi encontrado uma matrícula. LEXEMA: EF03877
Foi encontrado um coeficiente. LEXEMA: 23.4
Foi encontrado um cpf. LEXEMA: 654.987.231-09
Foi encontrado um coeficiente. LEXEMA: 67.9
Foi encontrado um código de disciplina. LEXEMA: LEF-100
Foi encontrado um código de disciplina. LEXEMA: MAF-243
Foi encontrado um nome de disciplina. LEXEMA: Calculo Diferencial e Integral
```

Figura 10: Resultado para a entrada definida pelo aluno para o lex2.l.

No caso deste exemplo, definido pelo aluno, tem-se o reconhecimento de códigos de disciplina, matrículas, coeficientes, e-mails, dentre outros, de acordo com as definições dos tokens mostradas anteriormente, seguindo as expressões regulares do código.

3. Arquivos lex

Nesta seção serão mostrados os códigos completos de ambos os arquivos lex (lex.l e lex2.l).

3.1 lex.l

```
%{

/*codigo colocado aqui aparece no arquivo gerado pelo flex*/

%}

/* This tells flex to read only one input file */
%option noyywrap

/* definicoes regulares */

delim        [ \t\n]
ws        {delim}+

inteiro [0-9]+
positivo [+][0-9]+
negativo [-][0-9]+
decimal [+|-]*[0-9]+[.][0-9]+
placa [A-Z][A-Z][A-Z][-][0-9][0-9][0-9][0-9]+
palavra [a-zA-Z]+
telefone [0-9][0-9][0-9][0-9][-][0-9][0-9][0-9][0-9]+
nome {palavra}[" "]{palavra}[" "]{palavra}[" "]*{palavra}*

%%


{ws}        {/*nenhuma acao e nenhum retorno*/}



{inteiro} {printf("Foi encontrado um numero inteiro positivo. LEXEMA: %s\n", yytext);}
{positivo} {printf("Foi encontrado um numero inteiro positivo. LEXEMA: %s\n", yytext);}
{negativo} {printf("Foi encontrado um numero inteiro negativo. LEXEMA: %s\n", yytext);}
{decimal} {printf("Foi encontrado um numero com parte decimal. LEXEMA: %s\n", yytext);}
{placa} {printf("Foi encontrado uma placa. LEXEMA: %s\n", yytext);}
{palavra} {printf("Foi encontrado uma palavra. LEXEMA: %s\n", yytext);}
{telefone} {printf("Foi encontrado um telefone. LEXEMA: %s\n", yytext);}
{nome} {printf("Foi encontrado um nome proprio. LEXEMA: %s\n", yytext);}

%%

/*codigo em C. Foi criado o main, mas podem ser criadas outras funcoes aqui.*/

int main(void) {
    FILE *fp;
    char filename[50];
    printf("Digite o nome do arquivo: \n");
    scanf("%s",filename);
    fp = fopen(filename,"r");
    if(fp != NULL)
        yyin = fp;

    /* Call the lexer, then quit. */
    yylex();
    return 0;
}
```

Figura 11: Arquivo lex.l.

3.2 lex2.l

```
%{

/*codigo colocado aqui aparece no arquivo gerado pelo flex*/

%}

/* This tells flex to read only one input file */
%option noyywrap

/* definicoes regulares */

delim        [ \t\n]
ws        {delim}+


palavra [a-zA-Z]+
cpf [0-9][0-9][0-9][.][0-9][0-9][0-9][.][0-9][0-9][0-9][-][0-9][0-9]
matricula [E][F][0][0-9][0-9][0-9][0-9]
email [a-zA-Z|0-9|._]+[@][a-zA-Z]+[.][a-zA-Z][a-zA-Z][a-zA-Z]*[.]*[a-zA-Z]*[a-zA-Z]*
coeficiente [0-9][0-9][.][0-9]
codigo_disciplina [A-Z][A-Z][A-Z][-][0-9][0-9][0-9]
nome_disciplina {palavra}[" "]{palavra}[" "]{palavra}[" "]*{palavra}


%%


{ws}        {/*nenhuma acao e nenhum retorno*/}


{cpf} {printf("Foi encontrado um cpf. LEXEMA: %s\n", yytext);}
{matricula} {printf("Foi encontrado uma matrícula. LEXEMA: %s\n", yytext);}
{email} {printf("Foi encontrado um email. LEXEMA: %s\n", yytext);}
{coeficiente} {printf("Foi encontrado um coeficiente. LEXEMA: %s\n", yytext);}
{codigo_disciplina} {printf("Foi encontrado um código de disciplina. LEXEMA: %s\n", yytext);}
{nome_disciplina} {printf("Foi encontrado um nome de disciplina. LEXEMA: %s\n", yytext);}



%%

/*codigo em C. Foi criado o main, mas podem ser criadas outras funcoes aqui.*/

int main(void)
{
    FILE *fp;
    char filename[50];
    printf("Digite o nome do arquivo: \n");
    scanf("%s",filename);
    fp = fopen(filename,"r");
    if(fp != NULL)
        yyin = fp;

    /* Call the lexer, then quit. */
    yylex();
    return 0;
}
```

Figura 12: Arquivo lex2.l.

4. Considerações Finais

O trabalho foi de suma importância para revisar os conceitos aprendidos em sala de aula, bem como ter uma visão prática de como tais conceitos são aplicados.

5. Referências

[1] AHO, A.V.; LAM, M.S.; SETHI, R.; ULLMAN, J.D. Compiladores: Princípios, técnicas e ferramentas. Segunda Edição. Pearson Addison-Wesley, 2008.

[2] Niemann Thomas. A Compact Guide to Lex & Yacc. ePaper Press.