

Compiladores

2ºAno de Engenharia Informática

Trabalho Prático

Relatório de Desenvolvimento

No âmbito de Compiladores;

Trabalho realizado por:

Francisco Carvalho 78557

Henrique Dias 78804

Carlos Pinto 78174

Índice

Introdução e objetivo de trabalho.....	3
Proposta de resolução para Exercício 1	4
1. MANUTENÇÃO(V)	4
2. CARREGA-BATERIA(V).....	6
3. ENTREGA(L,M,Q)	8
4. RECOLHE(LISTA)	10
5. ESTADO(I).....	12
6. Init-Estado.....	14
Proposta de resolução para o Exercício 2.....	16
Proposta de resolução para o Exercício 3.....	18
Proposta de resolução para o Exercício 4.....	20
Proposta de resolução para o Exercício 4 alínea B.....	22
Conclusão	24

Introdução e objetivo de trabalho

Para a disciplina de Compiladores do 2ºAno 1ºSemestre de Engenharia Informática, fomos desafiados com o tema da fábrica Compilando&Construindo.

A fábrica Compilando&Construindo utiliza um veículo elétrico autónomo para transportar os materiais necessários entre o armazém, onde os materiais são armazenados, e as linhas de montagem, onde os produtos são assembled. Existe também um posto de manutenção, onde o veículo é reparado em caso de avaria; e um posto de carregamento de energia elétrica, onde o veículo pode recarregar a sua bateria.

Na 1ªFase do trabalho foi proposta que, para cada tarefa, ou seja, para a manutenção, carregamento da bateria, entregas, recolha de materiais e estado atual do veículo, a criação de expressões regulares e os seus devidos autómatos finitos.

Na 2ªFase do trabalho foi proposta várias tarefas, tais como, para cada instrução fazer a leitura com o analisador léxico LEX/FLEX, atribuir um estado inicial do veículo e atualizando o mesmo depois das várias instruções do veículo e o seu estado.

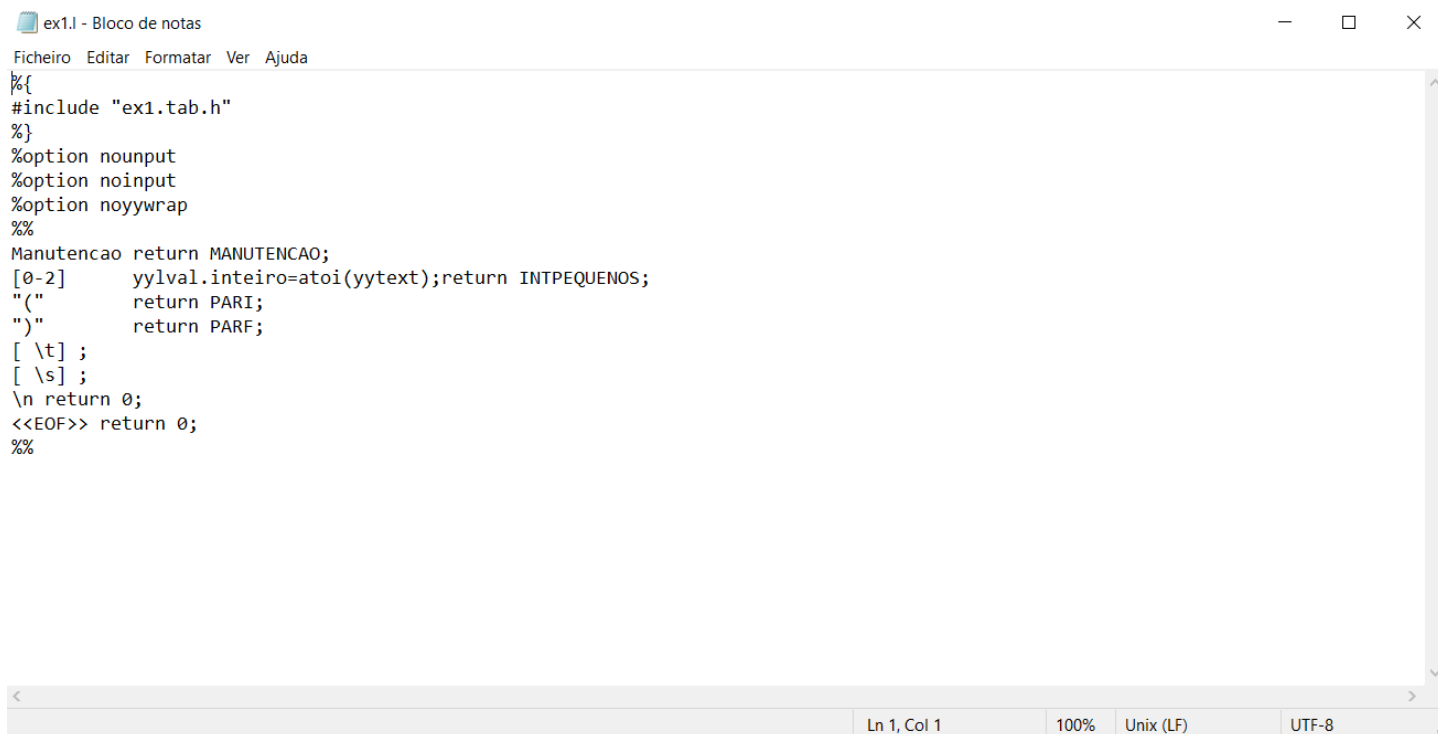
Na 3ªFase do trabalho foi proposto, com o analisador sintático YACC/BISON e respetiva interação com LEX/FLEX, atribuir um estado inicial do veículo e atualizando o mesmo depois das várias instruções do veículo e o seu estado.

Proposta de resolução para Exercício 1

1. MANUTENÇÃO(V)

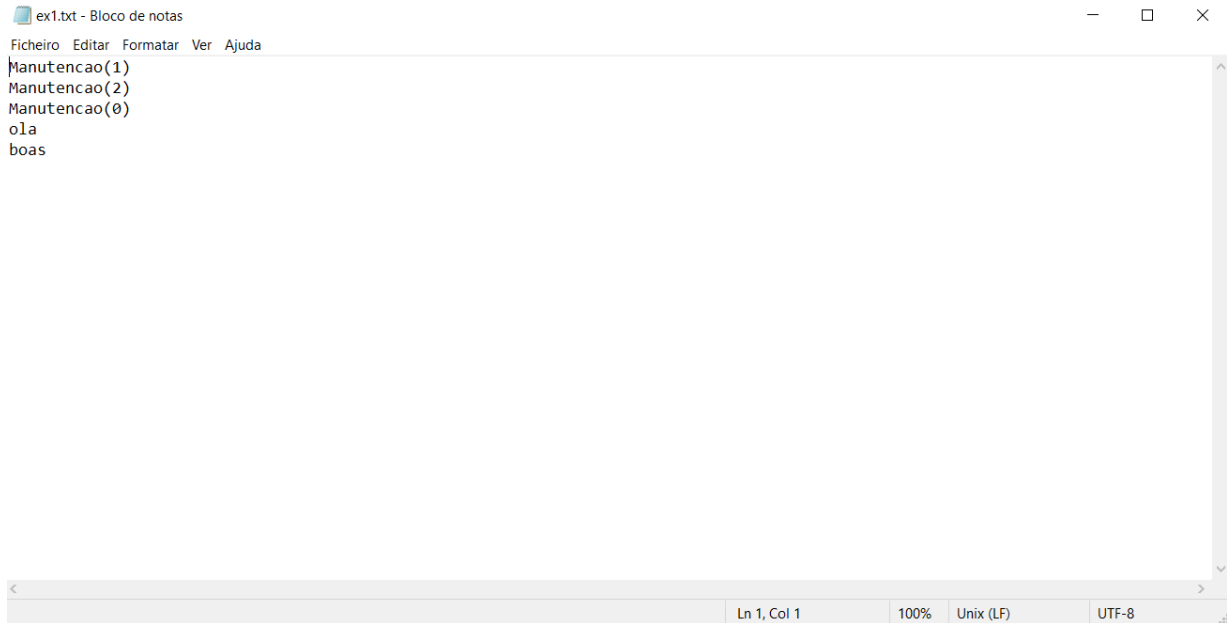
Indica ao veículo que se deve dirigir para o posto de manutenção, onde V pode assumir valores de 0, 1 ou 2. 0 significa que se deve deslocar imediatamente, 1 significa que antes de se deslocar para o posto de manutenção deve primeiro terminar alguma tarefa que esteja a desempenhar no momento, e 2 significa que antes de se deslocar para o posto de manutenção deve terminar todas as tarefas que tenha pendentes.

Para este primeiro exercício, foi colocada as diferentes partes da expressão regular no ficheiro LEX, e de seguida arranjadas de maneira o BISON fazer a leitura, ou seja, a análise sintática.




```
ex1.l - Bloco de notas
Ficheiro  Editar  Formatar  Ver  Ajuda
%{
#include "ex1.tab.h"
%}
%option nounput
%option noinput
%option noyywrap
%%
Manutencao return MANUTENCAO;
[0-2]      yylval.inteiro=atoi(yytext);return INTPEQUENOS;
"("        return PARI;
")"        return PARF;
[ \t] ;
[ \s] ;
\n return 0;
<<EOF>> return 0;
%%
```

Ficheiro Lex para a procura das expressões regulares referentes a “Manutencao”



```
ex1.txt - Bloco de notas
Ficheiro Editar Formatar Ver Ajuda
Manutencao(1)
Manutencao(2)
Manutencao(0)
ola
boas
Ln 1, Col 1 100% Unix (LF) UTF-8
```

Ficheiro de entrada (input)



```
ex1.y - Bloco de notas
Ficheiro Editar Formatar Ver Ajuda
%{
#include <stdio.h>
int nerros=0;
int yyerror(char *s);
int yylex();
}%
%union{
char *id;
int inteiro;
float real;}
%token <id> MANUTENCAO PARI PARF
%token <inteiro> INTPEQUENOS
%start inicio
%%
inicio:
| MANUTENCAO PARI INTPEQUENOS PARF {printf ("Frase da Manutencao");}
;
%%
int main(){
yyparse();
if(nerros==0){ printf("FRASE VÁLIDA!\n"); }
else{ printf ("FRASE NAO ACEITE %d \n",nerros); }
}
int yyerror(char *s){
nerros++;
}
```

Ficheiro YACC/BISON

2. CARREGA-BATERIA(V)

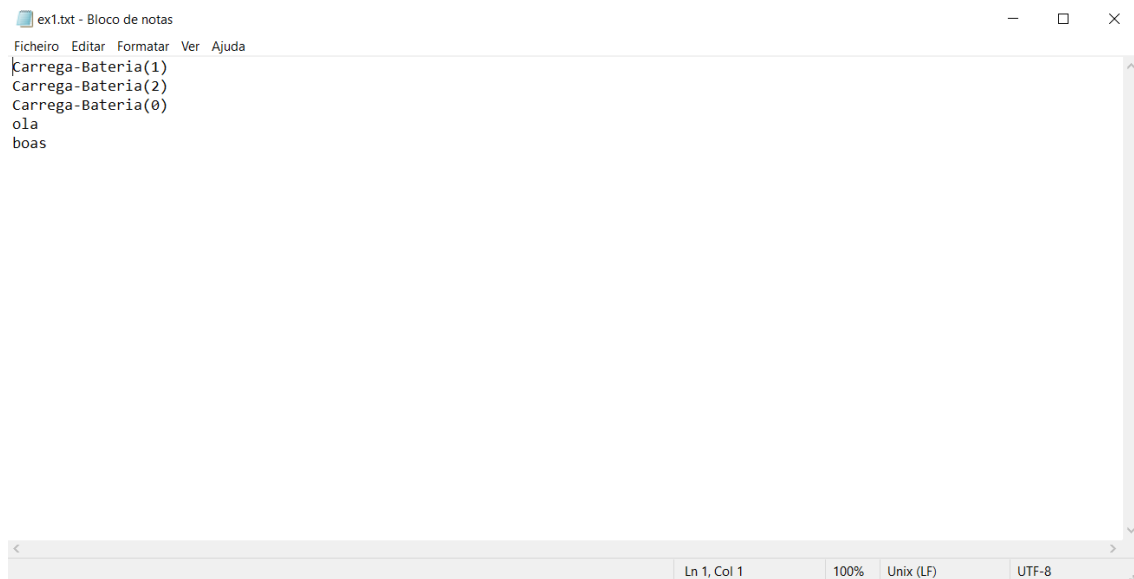
Indica ao veículo que se deve dirigir para o posto de carregamento, onde V pode assumir valores de 0, 1 ou 2. 0 significa que se deve deslocar imediatamente, 1 significa que antes de se deslocar para o posto de carregamento deve primeiro terminar alguma tarefa que esteja a desempenhar no momento, e 2 significa que antes de se deslocar para o posto de carregamento deve terminar todas as tarefas que tenha pendentes.

ex1.l - Bloco de notas

Ficheiro Editar Formatar Ver Ajuda

```
%{
#include "ex1.tab.h"
%}
%option nounput
%option noinput
%option noyywrap
%%
Carrega-Bateria return CARREGA;
[0-2]      yyval.inteiro=atoi(yytext);return INTPEQUENOS;
"("        return PARI;
")"        return PARF;
[ \t] ;
[ \s] ;
\n return 0;
<<EOF>> return 0;
%%
```

Ficheiro Lex para a procura das expressões regulares referentes a “Carrega-Bateria”



```
ex1.txt - Bloco de notas
Ficheiro Editar Formatar Ver Ajuda
Carrega-Bateria(1)
Carrega-Bateria(2)
Carrega-Bateria(0)
ola
boas
Ln 1, Col 1 100% Unix (LF) UTF-8
```

Ficheiro de entrada (input)

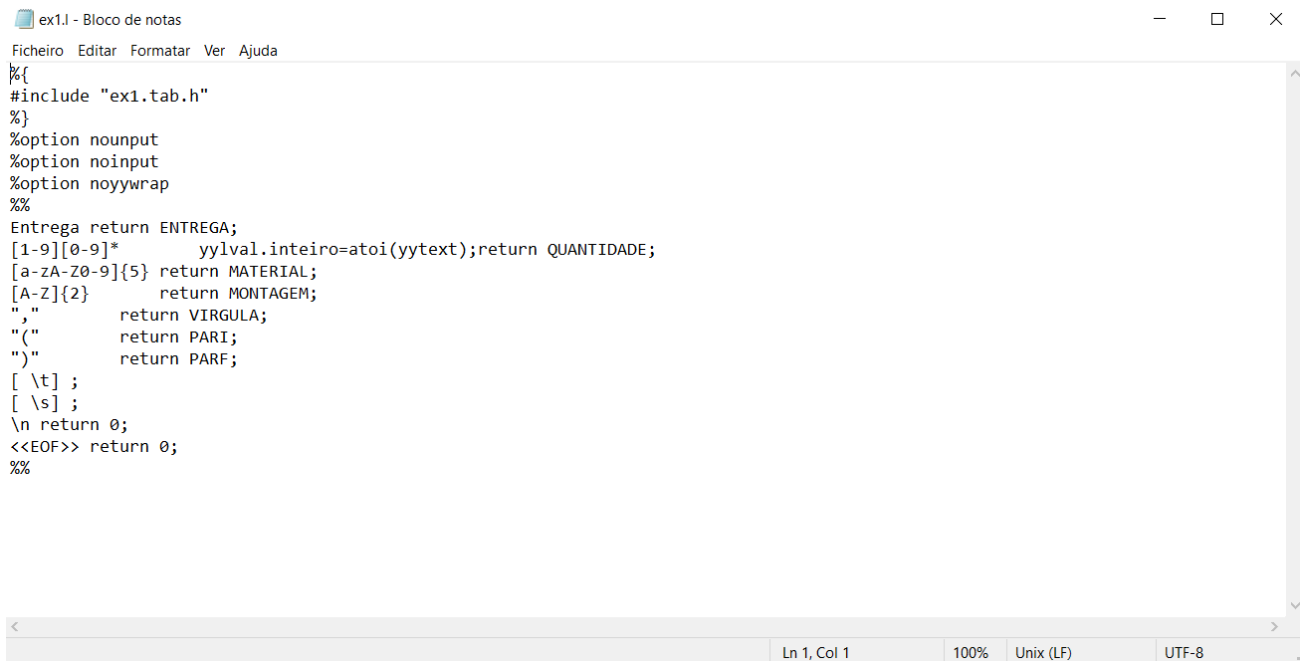


```
ex1.y - Bloco de notas
Ficheiro Editar Formatar Ver Ajuda
%{
#include <stdio.h>
int nerros=0;
int yyerror(char *s);
int yylex();
}%
%union{
char *id;
int inteiro;
float real;}
%token <id> CARREGA PARI PARF
%token <inteiro> INTPEQUENOS
%start inicio
%%
inicio:
| CARREGA PARI INTPEQUENOS PARF {printf ("Frase de Carrega-Bateria");}
;
%%
int main(){
yyparse();
if(nerros==0){ printf("FRASE VÁLIDA!\n"); }
else{ printf ("FRASE NAO ACEITE %d \n",nerros); }
}
int yyerror(char *s){
nerros++;
}
```

Ficheiro YACC/BISON

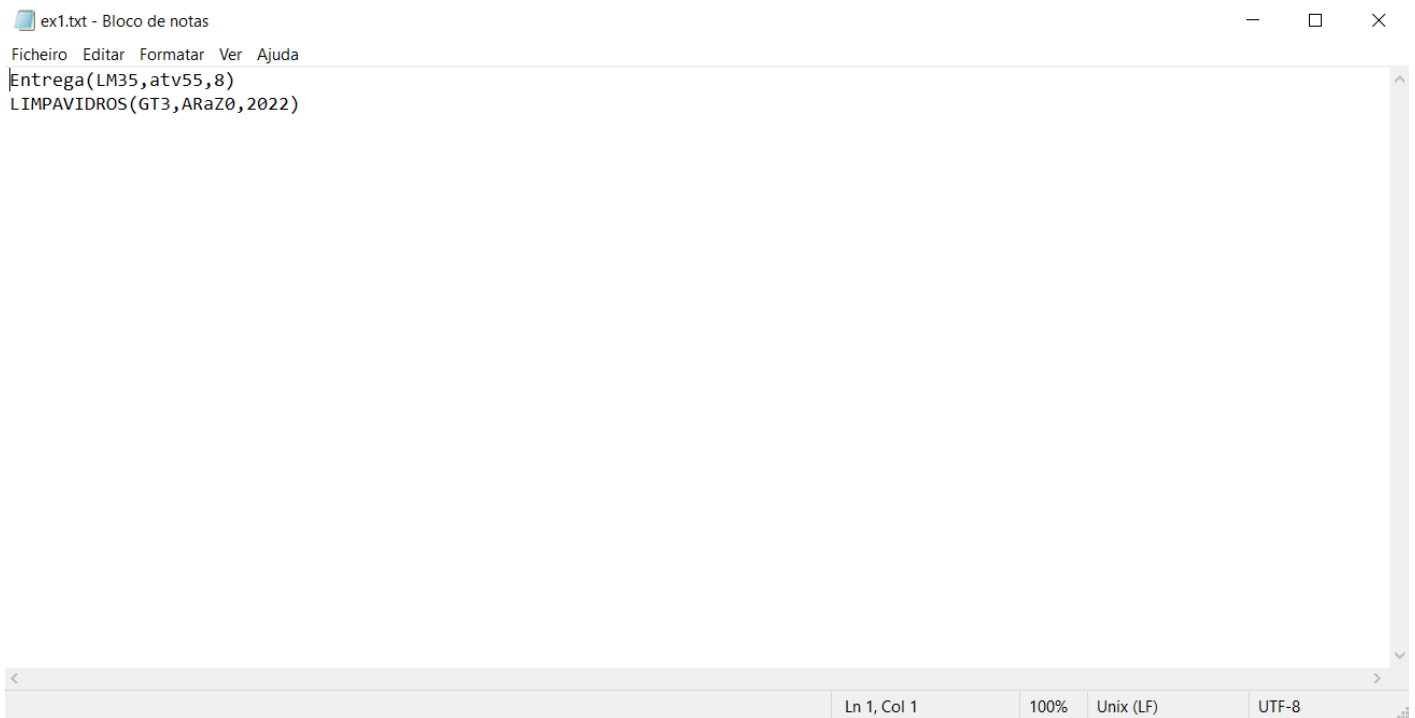
3. ENTREGA(L,M,Q)

Indica ao veículo que deve efetuar uma entrega numa linha de montagem. L identifica a linha de montagem através de um valor numérico entre 1 e 100 antecedido por duas letras maiúsculas. M identifica o material através de um código de 5 caracteres composto por letras e/ou números. Q representa a quantidade através de um valor numérico que pode assumir valores maiores que zero.



```
ex1.l - Bloco de notas
Ficheiro Editar Formatar Ver Ajuda
%{
#include "ex1.tab.h"
}%
%option nounput
%option noinput
%option noyywrap
%%
Entrega return ENTREGA;
[1-9][0-9]*      yylval.inteiro=atoi(yytext);return QUANTIDADE;
[a-zA-Z0-9]{5}   return MATERIAL;
[A-Z]{2}        return MONTAGEM;
", "            return VIRGULA;
"("             return PARI;
")"             return PARF;
[ \t] ;
[ \s] ;
\n return 0;
<<EOF>> return 0;
%%
```

Ficheiro Lex para a procura das expressões regulares referentes a “Entrega”



Ficheiro de entrada (input)

```
ex1.y - Bloco de notas
Ficheiro Editar Formatar Ver Ajuda
%{
#include <stdio.h>
int nerros=0;
int yyerror(char *s);
int yylex();
}%
%union{
char *id;
int inteiro;
float real;}
%token <id> ENTREGA PARI PARF MATERIAL MONTAGEM VIRGULA
%token <inteiro> QUANTIDADE
%start inicio
%%
inicio:
| ENTREGA PARI MONTAGEM QUANTIDADE VIRGULA MATERIAL VIRGULA QUANTIDADE PARF {printf ("Frase da Entrega");}
;
%%
int main(){
yyparse();
if(nerros==0){ printf("FRASE VÁLIDA!\n"); }
else{ printf ("FRASE NAO ACEITE %d \n",nerros); }
}
int yyerror(char *s){
nerros++;
}
```

Ficheiro YACC/BISON


4. RECOLHE(LISTA)

Indica ao veículo que se deve deslocar ao armazém para efetuar a recolha de uma lista de materiais. LISTA representa a lista de materiais, iniciada e terminada por [e] respetivamente, e onde cada elemento da lista é representado por uma tupla no formato (M,Q) onde M identifica o material através de um código de 5 caracteres composto por letras e/ou números e Q representa a quantidade através de um valor numérico que pode assumir valores inteiros maiores que zero.

Exemplo: RECOLHE([(A4gt6,300), (cbv45,3), (12345,21)])

```
ex1.l - Bloco de notas
Ficheiro Editar Formatar Ver Ajuda
%{
#include "ex1.tab.h"
}%
%option nounput
%option noinput
%option noyywrap
%%
Recolhe return RECOLHE;
"("[a-zA-Z0-9]{5}", "[1-9][0-9]*")"      yylval.inteiro=atoi(yytext);return METODO;
(", "("[a-zA-Z0-9]{5}", "[1-9][0-9]*")")*  return METODOREP;
"'"                                     return VIRGULA;
"("                                     return PARI;
")"                                     return PARF;
"["                                     return RECTI;
"]"                                     return RECTF;
[ \t] ;
[ \s] ;
\n return 0;
<<EOF>> return 0;
%%
```

Ficheiro Lex para a procura das expressões regulares referentes a “Recolhe”

 ex1.txt - Bloco de notas

Ficheiro Editar Formatar Ver Ajuda

Recolhe([(GT3RS,300), (FRE45,56), (TRS35,69)])

Ficheiro de entrada (input)

 ex1.y - Bloco de notas

Ficheiro Editar Formatar Ver Ajuda

```
%{
#include <stdio.h>
int nerros=0;
int yyerror(char *s);
int yylex();
}%
%union{
char *id;
int inteiro;
float real;}
%token <id> RECOLHE PARI PARF VIRGULA METODO METODOREP RECTI RECTF
%start inicio
%%
inicio:
| RECOLHE PARI RECTI METODO METODOREP RECTF PARF {printf ("Frase do Recolhe");}
;
%%
int main(){
yyparse();
if(nerros==0){ printf("FRASE VÁLIDA!\n"); }
else{ printf ("FRASE NAO ACEITE %d \n",nerros); }
}
int yyerror(char *s){
nerros++;
}
```

Ficheiro de saída (output)

5. ESTADO(I)

Indica ao veículo que deve comunicar o seu estado atual. I identifica a informação que deve ser comunicada, podendo assumir o valor de B (representa o estado da bateria); M (representa os materiais e quantidades que está a carregar); T (representa as tarefas que tem pendentes); ou qualquer combinação entre estas 3 letras, sendo que se existir mais que uma letra, estas devem ser separadas por uma vírgula.

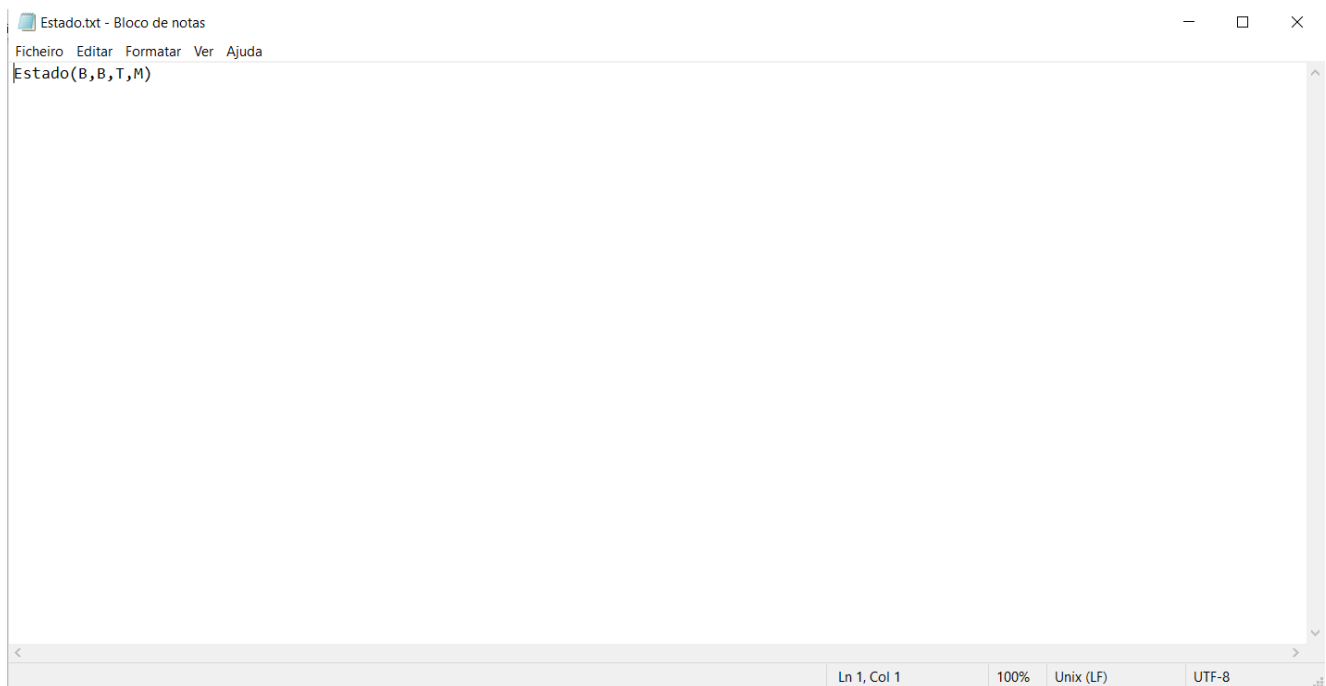


ex1.I - Bloco de notas

Ficheiro Editar Formatar Ver Ajuda

```
%{
#include "ex1.tab.h"
%}
%option nounput
%option noinput
%option noyywrap
%%
Estado return ESTADO;
(B|T|M|','(B|T|M))* return LETRAS;
[ \t] ;
[ \s] ;
\n return 0;
<<EOF>> return 0;
%%
```

Ficheiro Lex para a procura das expressões regulares referentes a “Estado”



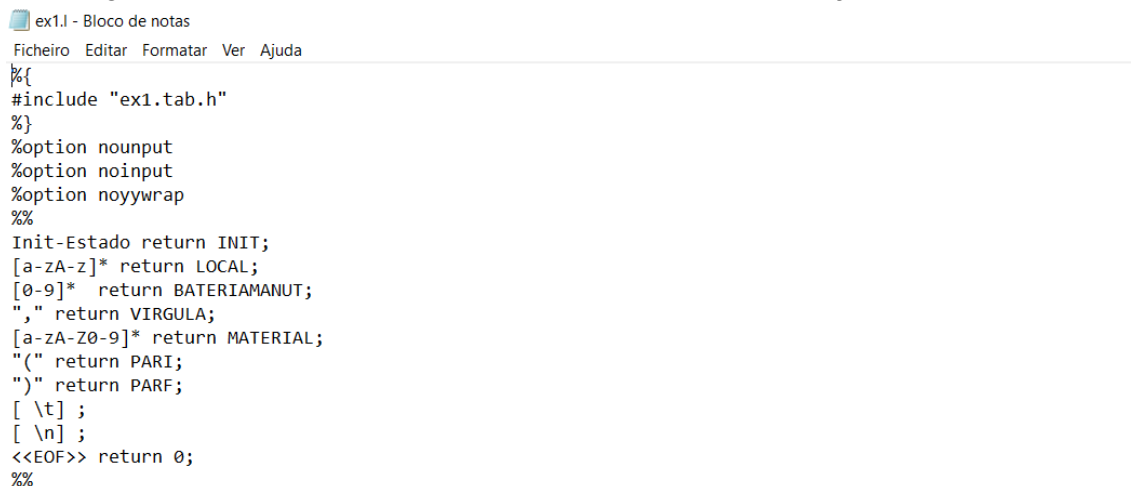
Ficheiro de entrada (input)

```
ex1.y - Bloco de notas
Ficheiro  Editar  Formatar  Ver  Ajuda
%{
#include <stdio.h>
int nerros=0;
int yyerror(char *s);
int yylex();
}%
%union{
char *id;
int inteiro;
float real;}
%token <id> ESTADO LETRAS
%start inicio
%%
inicio:
| ESTADO LETRAS  {printf ("Frase do Estado");}
;
%%
int main(){
yyparse();
if(nerros==0){ printf("FRASE VÁLIDA!\n"); }
else{ printf ("FRASE NAO ACEITE %d \n",nerros); }
}
int yyerror(char *s){
nerros++;
}
```

Ficheiro YACC/BISON

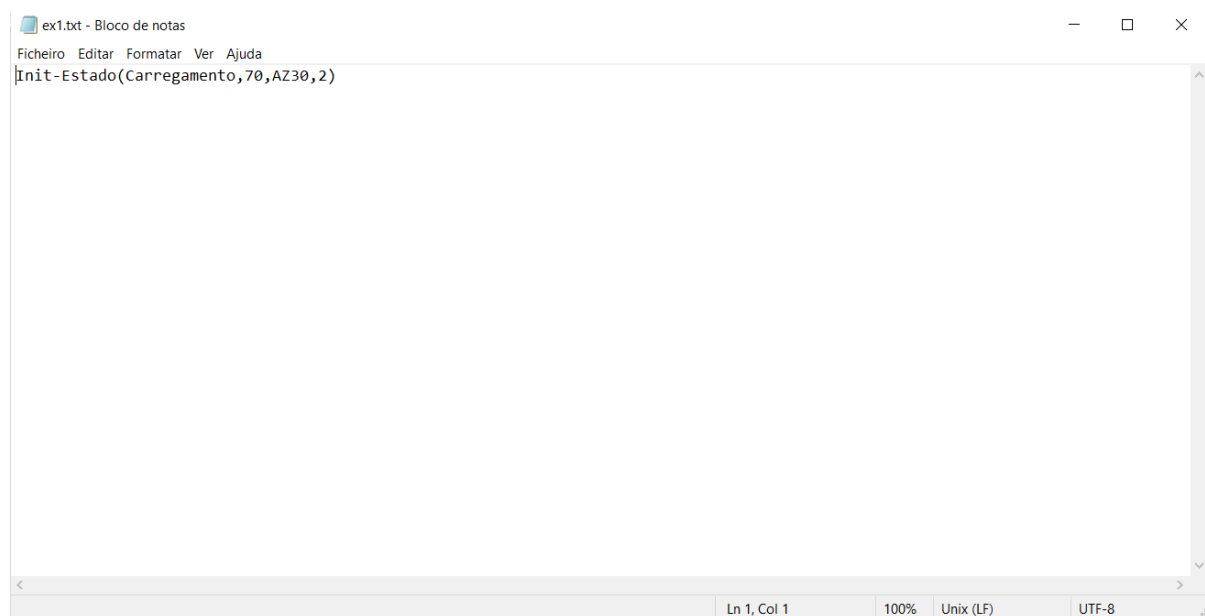
6. Init-Estado

INIT-ESTADO(L,B,M,N) – onde L indica a localização inicial do veículo, B o estado (%) inicial da bateria, M representa os materiais e respetivas quantidades que está a carregar, e N o número de vezes que o veículo foi à manutenção.



```
ex1.l - Bloco de notas
Ficheiro Editar Formatar Ver Ajuda
%{
#include "ex1.tab.h"
}%
%option nounput
%option noinput
%option noyywrap
%%
Init-Estado return INIT;
[a-zA-Z]* return LOCAL;
[0-9]* return BATERIAMANUT;
"," return VIRGULA;
[a-zA-Z0-9]* return MATERIAL;
 "(" return PARI;
 ")" return PARF;
[ \t] ;
[ \n] ;
<<EOF>> return 0;
%%
```

Ficheiro LEX/FLEX



```
ex1.txt - Bloco de notas
Ficheiro Editar Formatar Ver Ajuda
Init-Estado(Carregamento,70,AZ30,2)
```

Ficheiro de entrada(input)

```
%{
#include <stdio.h>
int nerros=0;
int yyerror(char *s);
int yylex();
}%
%union{
char *id;
int inteiro;
float real;}
%token <id> INIT PARI LOCAL VIRGULA BATERIAMANUT MATERIAL PARF
%start inicio
%%
inicio:
| INIT PARI LOCAL VIRGULA BATERIAMANUT VIRGULA MATERIAL VIRGULA BATERIAMANUT PARF {printf ("Frase do INIT ESTADO");}
;
%%
int main(){
yyparse();
if(nerros==0){ printf("FRASE VÁLIDA!\n"); }
else{ printf ("FRASE NAO ACEITE %d \n",nerros); }
}
int yyerror(char *s){
nerros++;
}
```

Ficheiro YACC/BISON

Proposta de resolução para o Exercício 2

Despoletar as ações correspondentes a cada uma das instruções, guardando e imprimindo, após cada instrução correta recebida, o estado atual do veículo (estado da bateria, localização atual, lista e quantidade de peças de cada tipo que está a transportar, número de vezes que foi à manutenção).

Neste exercício, a cada leitura da expressão regular foram feitas as diferentes regras, dentro do ficheiro BISON. Depois a impressão da informação está dentro da função main().

```
ex1.l - Bloco de notas
Ficheiro Editar Formatar Ver Ajuda
%{
#include "ex1.tab.h"
%}
%option nounput
%option noinput
%option noyywrap
%%
Manutencao return MANUTENCAO;
[0-2]      yylval.inteiro=atoi(yytext);return INTPEQUENOS;
"("[a-zA-Z0-9]{5}", "[1-9][0-9]*")"      yylval.inteiro=atoi(yytext);return METODO;
"(", "("[a-zA-Z0-9]{5}", "[1-9][0-9]*")")*"      return METODOREP;
"("      return PARI;
")"      return PARF;
[1-9][0-9]*      yylval.inteiro=atoi(yytext);return QUANTIDADE;
[a-zA-Z0-9]{5} return MATERIAL;
[A-Z]{2}      return MONTAGEM;
Carrega-Bateria return C;
Estado return E;
Recolhe return R;
Entrega return ENTREGA;
", "      return VIRGULA;
"["      return RECTI;
"]"      return RECTF;
[ \t] ;
[ \s] ;
[ \n] ;
<<EOF>> return 0;
%%
```

Ficheiro LEX/FLEX

ex1.txt - Bloco de notas

Ficheiro Editar Formatar Ver Ajuda

```
Manutencao(0)
Manutencao(1)
Carrega-Bateria(2)
Entrega(GT3,ARaZ0,30002)
Entrega(gt3,23AL,10)
ola
boas
```

Ficheiro de entrada (input)

ex1.y - Bloco de notas

Ficheiro Editar Formatar Ver Ajuda

```
%{
#include <stdio.h>
int nerros=0;
int bateria=100;
int nvezes=0;
int yyerror(char *s);
int yylex();
char localizacao[50] = "Armazem";
}%
%union{
char *id;
int inteiro;
float real;}
%token <id> MANUTENCAO PARI PARF C R METODO METODOREP RECTI RECTF L E MATERIAL MONTAGEM QUANTIDADE VIRGULA OUTRO ENTREGA
%token <inteiro> INTPEQUENOS
%start inicio
%%
inicio:
| MANUTENCAO PARI INTPEQUENOS PARF inicio {nvezes++; printf("\n %d vezes que foi a manutenção", nvezes);}
| C PARI INTPEQUENOS PARF inicio {printf ("\n Frase de Carrega-Bateria");}
| ENTREGA PARI MONTAGEM QUANTIDADE VIRGULA MATERIAL VIRGULA QUANTIDADE PARF
| OUTRO inicio
;
%%
int main(){
printf("\n Bateria: %d", bateria);
printf("\n Localizacao: %s", localizacao);
yyparse();
if(nerros==0){ printf("\n FRASE VÁLIDA!\n"); }
else{ printf ("\n"); }
}
int yyerror(char *s){
nerros++;
}
```

Ficheiro YACC/BISON

Proposta de resolução para o Exercício 3

Validar situações irregulares, e lançar um alerta quando estas ocorrerem, nomeadamente:

Validar se o veículo recebe uma instrução que o leve a recolher uma quantidade que o faça exceder a sua capacidade.

Validar se o veículo recebe uma instrução que o leve a entregar um tipo de material ou uma quantidade que não está a transportar.

Validar se recebe uma instrução para a qual o estado da bateria não é suficiente.

Validar se é pedido um carregamento quando a bateria está a 100%.

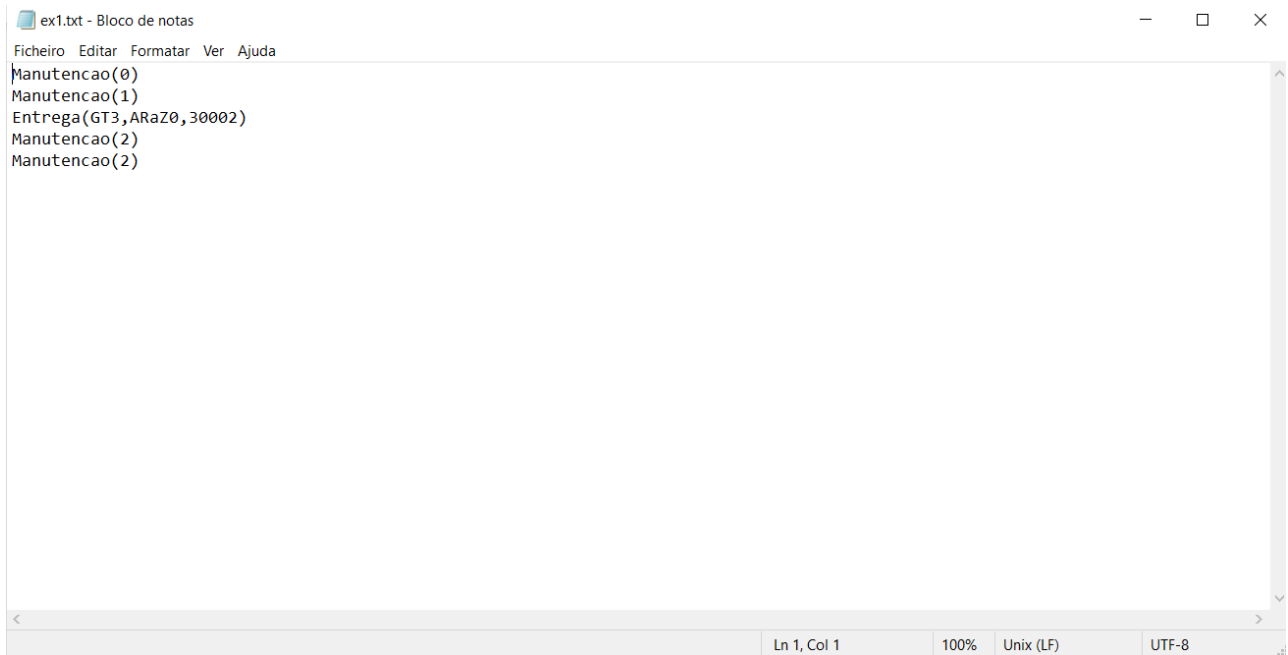
Validar e gerar um alerta se o veículo for chamado à manutenção mais de 3 vezes. Se isso acontecer deverá colocar o contador a zero ou então lançar aviso com múltiplos de 3 idas.

Para este exercício, foram construídas as diferentes regras, ou seja, mudanças de bateria, mudanças de localização, número de vezes que foi a manutenção. Todas estas foram feitas dentro da função main().

ex1.l - Bloco de notas

Ficheiro Editar Formatar Ver Ajuda

```
%{
#include "ex1.tab.h"
%}
%option nounput
%option noinput
%option noyywrap
%%
Manutencao return MANUTENCAO;
[0-2]      yylval.inteiro=atoi(yytext);return INTPEQUENOS;
"("[a-zA-Z0-9]{5}";"[1-9][0-9]*")"      yylval.inteiro=atoi(yytext);return METODO;
("","("[a-zA-Z0-9]{5}";"[1-9][0-9]*")")*      return METODOREP;
"("      return PARI;
")"      return PARF;
[1-9][0-9]*      yylval.inteiro=atoi(yytext);return QUANTIDADE;
[a-zA-Z0-9]{5} return MATERIAL;
[A-Z]{2}      return MONTAGEM;
Carrega-Bateria return C;
Estado return E;
Recolhe return R;
Entrega return ENTREGA;
", "      return VIRGULA;
"["      return RECTI;
"]"      return RECTF;
[ \t] ;
[ \s] ;
[ \n] ;
<<EOF>> return 0;
%%
```



Ficheiro de entrada (input)



```
%{
#include <stdio.h>
#include <string.h>
int nerros=0;
int bateria=100;
int nvezes=0;
int yyerror(char *s);
int yylex();
char localizacao[50] = "Armazem";
%}
%union{
char *id;
int inteiro;
float real;}
%token <id> MANUTENCAO PARI INTPEQUENOS PARF inicio {bateria-=10; nvezes++; strcpy(localizacao,"Armazem");}
%token <inteiro> INTPEQUENOS
%start inicio
%%
inicio:
| MANUTENCAO PARI INTPEQUENOS PARF inicio {bateria-=10; nvezes++; strcpy(localizacao,"Armazem");}
| C PARI INTPEQUENOS PARF inicio {bateria=100; printf ("\n Frase de Carrega-Bateria");}{strcpy(localizacao,"Carregamento");}
| ENTREGA PARI MONTAGEM QUANTIDADE VIRGULA MATERIAL VIRGULA QUANTIDADE PARF inicio {bateria-=10;}{strcpy(localizacao,"Entrega");}{printf("\n Frase do Entrega");}
| OUTRO inicio
;
%%
int main(){
yyparse();
if(nvezes=3){nvezes=0; printf("\n Reset a Manutencao"); } else {nvezes++;}
printf("\n Vezes que foi a Manutencao: %d", nvezes);
if(bateria<20){printf("Precisa de carregar... "); strcpy(localizacao,"Carregamento"); bateria=100; printf("\n Bateria a 100...Carro pronto a andar");} else {bateria=bateria;}
printf("\n Bateria: %d", bateria);
printf("\n Localizacao: %s",localizacao);
if(nerros==0){ printf("\n FRASE VÁLIDA!\n"); }
else{ printf ("ERRO\n"); }
}
int yyerror(char *s){
nerros++;
}
}
```

Ficheiro YACC/BISON

Proposta de resolução para o Exercício 4

Imprimir o estado final do veículo (estado da bateria, localização final, lista e quantidade de peças de cada tipo que está a transportar, número de vezes que foi à manutenção) após terminada a análise do input.

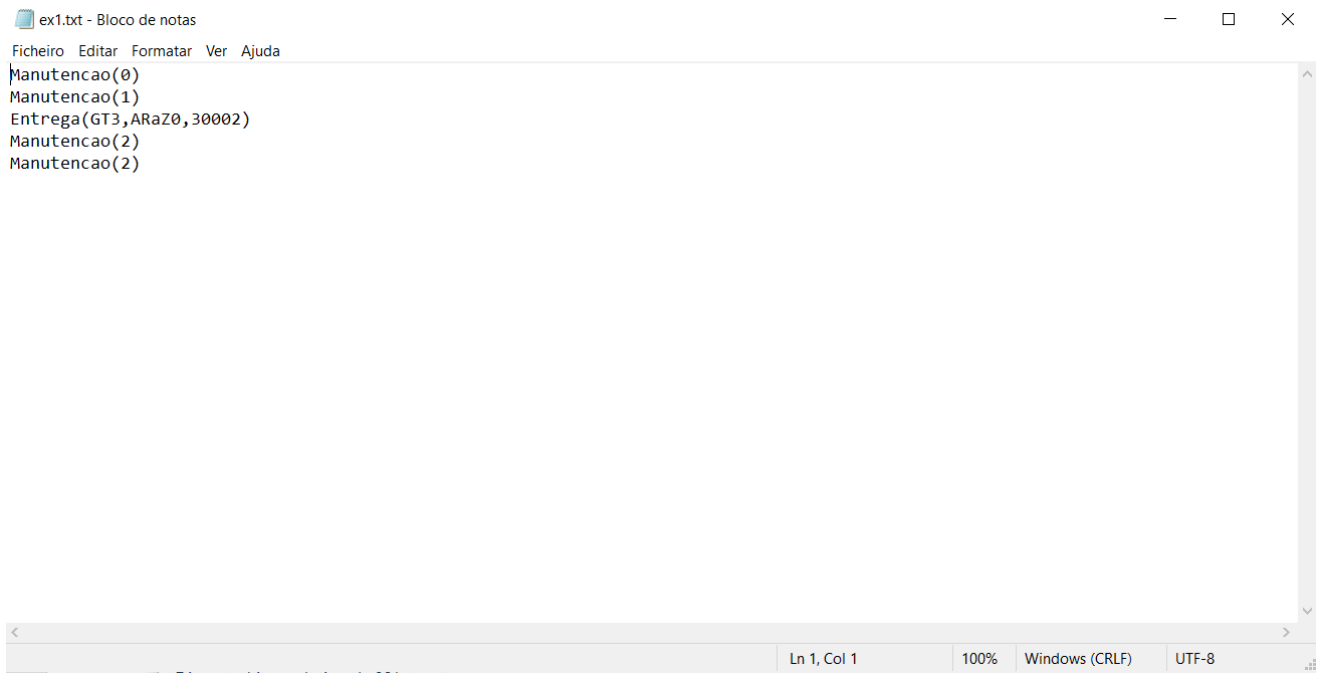
Basicamente, através das diferentes regras do ex3, imprimi-mos as informações finais do veículo aqui no ex4.

ex1.l - Bloco de notas

Ficheiro Editar Formatar Ver Ajuda

```
%{
#include "ex1.tab.h"
%}
%option nounput
%option noinput
%option noyywrap
%%
Manutencao return MANUTENCAO;
[0-2]      yylval.inteiro=atoi(yytext);return INTPEQUENOS;
("[a-zA-Z0-9]{5}", "[1-9][0-9]*")      yylval.inteiro=atoi(yytext);return METODO;
("","[a-zA-Z0-9]{5}", "[1-9][0-9]*")*      return METODOREP;
("("      return PARI;
")"      return PARF;
[1-9][0-9]*      yylval.inteiro=atoi(yytext);return QUANTIDADE;
[a-zA-Z0-9]{5} return MATERIAL;
[A-Z]{2}      return MONTAGEM;
Carrega-Bateria return C;
Estado return E;
Recolhe return R;
Entrega return ENTREGA;
", "      return VIRGULA;
"["      return RECTI;
"]"      return RECTF;
[ \t] ;
[ \s] ;
[ \n] ;
<<EOF>> return 0;
%%
```

Ficheiro Lex para a procura das expressões regulares referentes a “Ex4”



```
ex1.txt - Bloco de notas
Ficheiro  Editar  Formatar  Ver  Ajuda
Manutencao(0)
Manutencao(1)
Entrega(GT3,ARaZ0,30002)
Manutencao(2)
Manutencao(2)
```

Ficheiro de entrada (input)



```
ex1.y - Bloco de notas
Ficheiro  Editar  Formatar  Ver  Ajuda
%{
#include <stdio.h>
#include <string.h>
int nerros=0;
int bateria=100;
int nvezes=0;
int yyerror(char *s);
int yylex();
char localizacao[50] = "Armazem";
}%
%union{
char *id;
int inteiro;
float real;}
%token <id> MANUTENCAO PARI PARF C R METODO METODOREP RECTI RECTF L E MATERIAL MONTAGEM QUANTIDADE VIRGULA OUTRO ENTREGA
%token <inteiro> INTPEQUENOS
%start inicio
%%
inicio:
| MANUTENCAO PARI INTPEQUENOS PARF inicio {bateria-=10; nvezes++; strcpy(localizacao,"Armazem");}
| C PARI INTPEQUENOS PARF inicio {bateria=100; printf ("\n Frase de Carrega-Bateria");}{strcpy(localizacao,"Carregamento");}
| ENTREGA PARI MONTAGEM QUANTIDADE VIRGULA MATERIAL VIRGULA QUANTIDADE PARF inicio {bateria-=10;}{strcpy(localizacao,"Entrega");}{printf("\n Frase do Entrega");}
| OUTRO inicio
;
%%
int main(){
yyparse();
if(nvezes=3){nvezes=0; printf("\n Reset a Manutencao"); } else {nvezes++;}
printf("\n Vezes que foi a Manutencao: %d", nvezes);
if(bateria<20){printf("Precisa de carregar... "); strcpy(localizacao,"Carregamento"); bateria=100; printf("\n Bateria a 100...Carro pronto a andar");} else {bateria=bateria;}
printf("\n Bateria: %d", bateria);
printf("\n Localizacao: %s",localizacao);
if(nerros==0){ printf("\n FRASE VÁLIDA!\n"); }
else{ printf ("ERRO\n"); }
}
int yyerror(char *s){
nerros++;
}
}
```

Ficheiro YACC/BISON

Proposta de resolução para o Exercício 4 alínea B

Crie um ficheiro de texto para teste, que contenha a sequência de instruções necessárias para que o veículo consiga, partindo do estado inicial, recolher do armazém e entregar nas seguintes linhas de montagem, as quantidades de materiais indicadas de seguida:

- Linha: LM035; Material: A4gt6; Quantidade: 20
- Linha: RV002; Material: A4gt6; Quantidade: 20
- Linha: IU100; Material: 12dF3; Quantidade: 10

```
ex1.l - Bloco de notas
Ficheiro Editar Formatar Ver Ajuda
%{
#include "ex1.tab.h"
%}
%option nounput
%option noinput
%option noyywrap
%%
Manutencao return MANUTENCAO;
[0-2]      yylval.inteiro=atoi(yytext);return INTPEQUENOS;
"("[a-zA-Z0-9]{5}", "[1-9][0-9]*")"      yylval.inteiro=atoi(yytext);return METODO;
(", "("[a-zA-Z0-9]{5}", "[1-9][0-9]*")")*      return METODOREP;
"("      return PARI;
")"      return PARF;
[1-9][0-9]*      yylval.inteiro=atoi(yytext);return QUANTIDADE;
[a-zA-Z0-9]{5} return MATERIAL;
[A-Z]{2}      return MONTAGEM;
Carrega-Bateria return C;
Estado return E;
Recolhe return R;
Entrega return ENTREGA;
" ,"      return VIRGULA;
"["      return RECTI;
"]"      return RECTF;
[ \t] ;
[ \s] ;
[ \n] ;
<<EOF>> return 0;
%%
```

Ficheiro Lex para a procura das expressões regulares referentes a “Ex4”

```
ex1.txt - Bloco de notas
Ficheiro Editar Formatar Ver Ajuda
RECOLHE([(A4gt6,010),(atv55,009),(12H45,004)])
ENTREGA(LM35,atv55,8)
```

Ficheiro de entrada (input)

```
ex1.y - Bloco de notas
Ficheiro Editar Formatar Ver Ajuda
%{
#include <stdio.h>
#include <string.h>
int nerros=0;
int bateria=100;
int nvezes=0;
int yyerror(char *s);
int yylex();
char localizacao[50] = "Armazem";
}%
%union{
char *id;
int inteiro;
float real;}
%token <id> MANUTENCAO PARI PARF C R METODO METODOREP RECTI RECTF L E MATERIAL MONTAGEM QUANTIDADE VIRGULA OUTRO ENTREGA
%token <inteiro> INTPEQUENOS
%start inicio
%%
inicio:
| MANUTENCAO PARI INTPEQUENOS PARF inicio {bateria-=10; nvezes++; strcpy(localizacao,"Armazem");}
| C PARI INTPEQUENOS PARF inicio {bateria=100; printf("\n Frase de Carrega-Bateria");}{strcpy(localizacao,"Carregamento");}
| ENTREGA PARI MONTAGEM QUANTIDADE VIRGULA MATERIAL VIRGULA QUANTIDADE PARF inicio {bateria-=10;}{strcpy(localizacao,"Entrega");}{printf("\n Frase do Entrega");}
| OUTRO inicio
;
%%
int main(){
yyparse();
if(nvezes=3){nvezes=0; printf("\n Reset a Manutencao"); } else {nvezes++;}
printf("\n Vezes que foi a Manutencao: %d", nvezes);
if(bateria<20){printf("Precisa de carregar... "); strcpy(localizacao,"Carregamento"); bateria=100; printf("\n Bateria a 100...Carro pronto a andar");} else {bateria=bateria;}
printf("\n Bateria: %d", bateria);
printf("\n Localizacao: %s",localizacao);
if(nerros==0){ printf("\n FRASE VÁLIDA!\n"); }
else{ printf("ERRO\n"); }
}
int yyerror(char *s){
nerros++;
}
```

Ficheiro YACC/BISON

Conclusão

No nosso projeto falamos sobre o problema a que fomos propostos, refletimos sobre o que tínhamos e quais eram os nossos objetivos e de que modo podíamos chegar a eles, tendo em conta o que dispúnhamos.

A partir deste trabalho, conseguimos ter uma melhor compreensão sobre a análise sintática, como ela funciona e a implementação da mesma em ficheiros de entrada(input).

Todos os resultados finais foram alcançados para esta 3ª Fase.