Departamento de Engenharia Informática

Licenciatura em Engenharia Informática

Linguagens e Programação

Parsers Ascendentes

Considerando a atual conjuntura de confinamento social, o ginásio **BoraLá a Mexer** solicitou o desenvolvimento de um analisador sintático que validasse o formato dos registos das atividades dos seus clientes, para analisar o esforço diário envolvido em atividades físicas.

Para tal, desenvolveu um processo automático que permite receber a listagem com a descrição dos registos que os clientes realizaram, num ficheiro de texto cujas linhas têm a seguinte estrutura em formato EBNF (em que os campos opcionais são apresentados entre [] e os que se repetem 0 ou mais vezes entre { }):

{ **<tipo_de_atividade> [data] <duracao> <distancia> {marco}'\n'}**

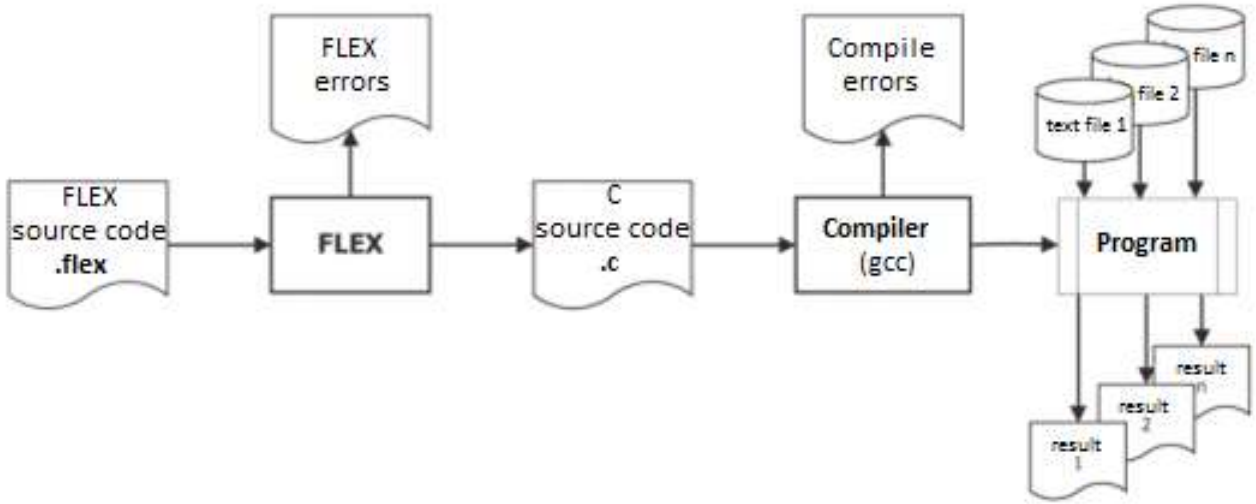
Em que:

- <**tipo_de_atividade**> pode ser **CORRIDA** ou **CICLISMO**;
- <**data**> é uma *String* no formato **ANO/MÊS/DIA**, indicativa da data da atividade, onde o ANO deve estar entre 2000 e 2099, o MÊS deve representar um número de mês válido com dois dígitos e o DIA um valor numérico, também com dois dígitos, entre 01 e 31;
- <**duracao**> representa o tempo da atividade, no formato **HORAS:MINUTOS**;
- <**distancia**> representa um valor **real** seguido do símbolo **KM** indicando a distância da atividade;
- <**marco**> é uma *String*, com o máximo de 32 caracteres e sem espaçamentos, com a descrição de um local de interesse por onde a atividade passou;

Defina a gramática para o ficheiro anteriormente descrito, e crie utilizando o ***Flex*** e o ***Bison*** um programa que:

- Reconheça a validade do ficheiro e implemente uma estratégia simples de recuperação de erros;
- Valide o input e, para cada linha, apresente o tempo total da atividade em minutos. No final da execução deverá apresentar a distância total percorrida global e por tipo de atividade, conforme exemplo seguinte:

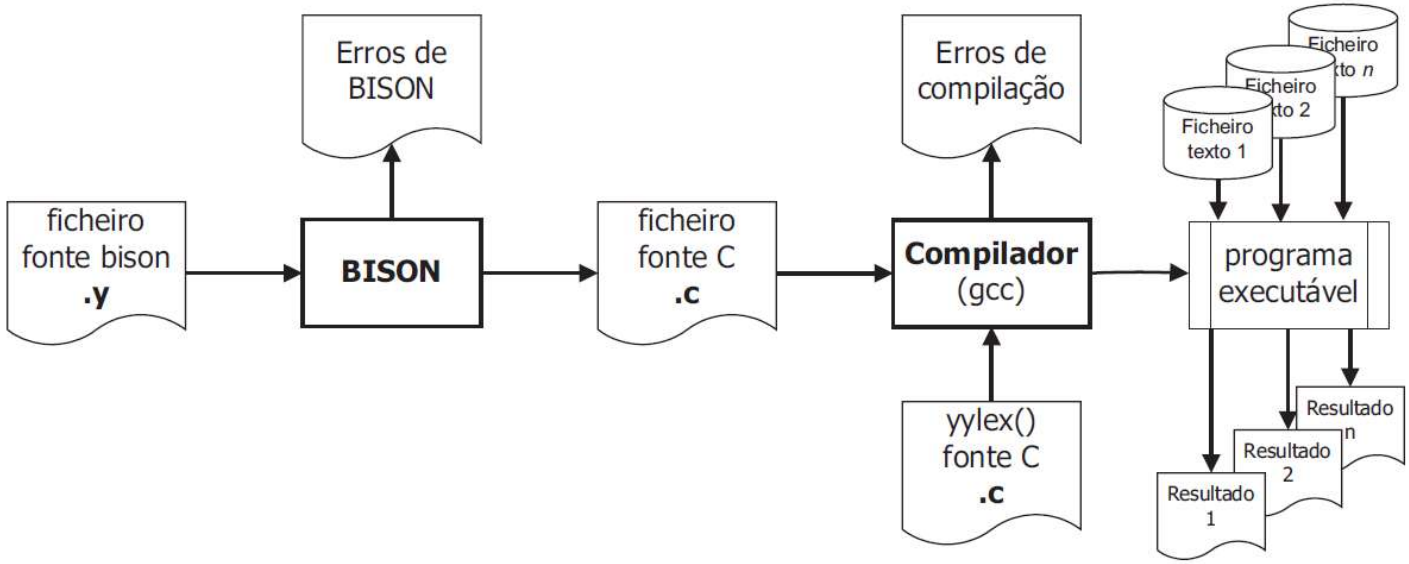
Input	Output
CORRIDA 00:18 5.0KM	Duracao: 18 minutos, 18 total
CICLISMO 2019/12/20 01:30 50.0KM GONDOMAR GAIA	Duracao: 90 minutos, 108 total
CORRIDA 01:25 21.1KM	Duracao: 85 minutos, 193 total
CORRIDA 03:30 42.2KM PORTO GAIA	Duracao: 210 minutos, 403 total
	Distancia corrida: 68.3KM
	Distancia ciclismo: 50.0KM
	Distancia total: 118.3KM
	Ficheiro válido




```
flex file.flex
gcc lex.yy.c -lfl
./a.out

Or


flex -oExample.c Example.flex
gcc Example.c –o Program -lfl
./Program < Data.txt
```




```
bison -d file.y
flex file.flex
gcc file.tab.c lex.yy.c -lfl
./a.out
Ou

bison -d example.y
flex -o example.c example.flex
gcc example.tab.c example.c –o Program -lfl
./Program < Data.txt
```

Flex:

```
%{
   #include "expr.tab.h"
   extern int nerros;
%}
%option noinput
%option nounput
%option header-file="lex.yy.h"
%%
[0-9]+          {yylval.inteiro=atoi(yytext); return INTEGER;}
[0-9]+\.[0-9]+ {yylval.real=atof(yytext); return REAL;}
CORRIDA         return CORRIDA;
CICLISMO        return CICLISMO;
KM                   return KM;
20[0-9][0-9]\/((0[1-9])|1[0-2])\/(0[1-9]|[1-2][0-9]|3[0-1])  {yylval.s = strdup(yytext); return DATA;}
[a-zA-z]{1,32}                              {yylval.s = strdup(yytext); return MARCO;}
:               return yytext[0];
\n              return yytext[0];
[ \t]       ;
.               {nerros++;printf("Erro lexico\n");}
%%
```

input() can be used in an action to return the next character from the input stream.
unput(c) can be used to insert a character into the input stream, so that the character will be scanned when the action finishes.

Bison

```
%{  /* demo08/expr.y  */
     #include "lex.yy.h"
     void yyerror( char * s );
    int nerros=0;
    float totalCorrida = 0;
    float totalCiclismo = 0;
    int totalMinutos = 0;
%}
%union {
   double real;
   intinteiro;
   char* s;
}

%token <inteiro> INTEGER
%token <real> REAL
%token <s> DATA MARCO
%token CORRIDA CICLISMO KM END_OF_FILE

%type <real> S TOTAL LINHA
```

```
%%
S : TOTAL  { printf("Distancia Corrida: %.1fKM\n",totalCorrida);
             printf("Distancia Ciclismo: %.1fKM\n",totalCiclismo);
             printf("Distancia total: %.1fKM\n",$1);
           } ;

TOTAL : TOTAL LINHA { $$=$1+$2;}
       |   {$$=0;} ;

LINHA : CORRIDA DATA_OPC INTEGER ':' INTEGER REAL KM MARCOS_OPC '\n' { totalMinutos= totalMinutos + ($3 * 60) + $5;
                                  printf("Duracao: %d minutos, %d total\n", (($3 * 60) + $5), totalMinutos);
                                  totalCorrida = totalCorrida + $6;
                                  $$=$5;}
      | CICLISMO DATA_OPC INTEGER ':' INTEGER REAL KM MARCOS_OPC '\n' { totalMinutos= totalMinutos + ($3 * 60) + $5;
                                  printf("Duracao: %d minutos, %d total\n", (($3 * 60) + $5), totalMinutos);
                                  totalCiclismo = totalCiclismo + $6;
                                  $$=$5;}
DATA_OPC : DATA ;
         | ;

MARCOS_OPC : MARCO MARCOS_OPC ;
           | ;

   ;

/* NÃO É OBRIGATÓRIA A UTILIZAÇÃO DE TODAS AS LINHAS DISPONÍVEIS*/

%%
int main() {
    yyparse();
    if (nerros == 0) {
        printf("Ficheiro valido\n");
    } else {
        printf("Ficheiro invalido, n erros=%d \n", nerros);
    }
    return 0;
}
```