**MUSIC TRANSLATOR**

**MOHAMED BASITH.H-2017103024**

**VELMURUGAN.G-2017103047**

**PROBLEM STATEMENT :**

The translator translates music notations from simple music notation to ABC notation.

**TOOLS USED :**

1.LEX AND YACC IS USED.

2.PATTERN MATCHING AND RETURN OF TOKEN IS DONE IN LEX FILE.

**HOW TO RUN :**

bison -y -d trans.y

flex trans.l

gcc -c y.tab.c lex.yy.c

gcc y.tab.o lex.yy.o -o trans.exe

./trans.exe <notation.txt

**INPUT & OUTPUT FILES :**

**trans.l** :

/\* simple music notation lex file \*/

%{

#include "y.tab.h"

#include <stdlib.h>

void yyerror(char \*);

int transform(int org);

%}

%%

[drmfslt] {

yylval.num = transform(\*yytext - 'a');

return SYMBOLS;

}

[DRMFSLT] {

yylval.num = transform(\*yytext - 'A') + 10;

return SYMBOLS;

}

[1-9] {

yylval.num = atoi(yytext);

return ID;

}

[0] {

yylval.num = 8;

return SYMBOLS;

}

[(:)/\!@>] {

return \*yytext;

}

[-]+ {

yylval.num = strlen(yytext) + 1;

return EN;

}

[ \t\n]+ ;

. yyerror("unknown symbols");

%%

int yywrap(void) {

return 1;

}

int transform(int org) {

if (org == 3) {

// Dd

return 0;

} else if (org == 17) {

// Rr

return 1;

} else if (org == 12) {

// Mm

return 2;

} else if (org == 5) {

// Ff

return 3;

} else if (org == 18) {

// Ss

return 4;

} else if (org == 11) {

// Ll

return 5;

} else if (org == 19) {

// Tt

return 6;

} else {

return -1;

}

}

**trans.y :**

//yacc file for music notation

%{

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define MAX\_SYMBOLS 3

void yyerror(char \*);

int yylex(void);

char \*trans(int sym);

char \*int2str(int sym);

char \*aPrint(char \*first, char \*second);

char \*bPrint(int id, char \*content);

char \*cPrint(int type, char \*sym);

char \*dPrint(int type, char \*first, char \*second);

char \*tags[10];

const char low\_s[7] = {'d', 'r', 'm', 'f', 's', 'l', 't'};

const char upp\_s[7] = {'D', 'R', 'M', 'F', 'S', 'L', 'T'};

const char\* low\_abc[7] = {"G,", "A,", "B,", "C", "D", "E", "F"};

const char\* upp\_abc[7] = {"G", "A", "B", "c", "d", "e", "f"};

%}

%token <iVal> SYMBOLS ID EN

%union {

char \*charPr;

int iVal;

int num;

}

%type <charPr> melody tagcall sentences onecombo twocombo syllable

%%

melody:

sentences {}

;

tagcall:

'(' ID ':' sentences ')' { $$ = bPrint($2, $4); }

| '@' ID { printf("%s", tags[$2]); $$ = tags[$2]; }

;

sentences:

sentences onecombo { $$ = aPrint($1, $2); }

| sentences twocombo { $$ = aPrint($1, $2); }

| sentences tagcall {}

| { $$ = ""; }

;

onecombo:

'\\' syllable { $$ = cPrint(0, $2); }

| '/' syllable { $$ = cPrint(1, $2); }

| syllable { $$ = $1; }

;

twocombo:

onecombo '>' onecombo { $$ = dPrint(0, $1, $3); }

| '!' onecombo onecombo { $$ = dPrint(1, $2, $3); }

| '!' onecombo '>' onecombo { $$ = dPrint(2, $2, $4); }

;

syllable:

SYMBOLS { $$ = trans($1); }

| EN { $$ = int2str($1); }

;

%%

char \*trans(int sym) {

if (sym >= 0 && sym < 7) {

int len = strlen(low\_abc[sym]);

char \*result = malloc(sizeof(char) \* len);

strcpy(result, low\_abc[sym]);

return result;

}

else if (sym >= 10 && sym <17) {

// printf("test: %d\n", sym);

int len = strlen(upp\_abc[sym - 10]);

char \*result = malloc(sizeof(char) \* len);

strcpy(result, upp\_abc[sym-10]);

return result;

}

else if (sym == 8) {

char \*result = "z";

return result;

}

else {

yyerror("Syntax error");

}

}

char \*int2str(int sym) {

char \*result = malloc(sizeof(char) \* 100);

sprintf(result, "%d", sym);

return result;

}

char \*aPrint(char \*first, char \*second) {

// printf("test\n");

int len = strlen(first) + strlen(second);

char \*result = malloc(sizeof(char) \* len);

strcpy(result, first);

strcat(result, second);

printf("%s", second);

return result;

}

char \*bPrint(int id, char \*content) {

// printf("%s\n", content);

int len = strlen(content);

char \*result = malloc(sizeof(char) \* len);

strcpy(result, content);

tags[id] = result;

return result;

}

char \*cPrint(int type, char \*sym) {

int len = strlen(sym) + 1;

char \*result = malloc(sizeof(char) \* len);

if (type == 0) {

strcpy(result, "\_");

strcat(result, sym);

return result;

}

else if (type == 1) {

strcpy(result, "^");

strcat(result, sym);

return result;

}

}

char \*dPrint(int type, char \*first, char \*second) {

if (type == 0) {

int len = strlen(first) + strlen(second) + 1;

char \*result = malloc(sizeof(char) \* len);

strcpy(result, first);

strcat(result, ">");

strcat(result, second);

return result;

}

else if (type == 1) {

int len = strlen(first) + strlen(second) + 2;

char \*result = malloc(sizeof(char) \* len);

strcpy(result, first);

strcat(result, "/");

strcat(result, second);

strcat(result, "/");

return result;

}

else if (type == 2) {

int len = strlen(first) + strlen(second) + 6;

char \*result = malloc(sizeof(char) \* len);

strcpy(result, first);

strcat(result, "3/4");

strcat(result, second);

strcat(result, "1/4");

return result;

}

}

void yyerror(char \*s) {

fprintf(stderr, "%s\n", s);

}

int main(void) {

printf("Sno: 1\n");

printf("Title: This is the title.\n");

printf("Composer: This is the composer.\n");

printf("Length: 1/4\n");

printf("K: G\n");

yyparse();

printf("\n");

}



