#include <stdio.h>

#include <stdlib.h>

#include <string.h>

//1、读取源代码到缓冲区。构造字符到整型的映射表；将char转换为int

struct CtoI{

char ch;

int in;

};

//初始化CtoITable

void init\_CtoITable(struct CtoI CtoITable[]);

//字符转换为整型

int getIfromC(char ch,struct CtoI \*CtoITable);

#define MAX 2000

struct sourceBuffer{

int source[MAX]; //存放源代码的缓冲区

int num; //source中有效数据的个数

}sour;

struct sourceBuffer readSource(FILE \*fp,struct CtoI \*CtoITable);

//2、构造规则表

#define R 5

#define C 6

//只有一个初态，[1][0]

void init\_RulesTable(int rules[R][C]);

/\*

3、对输入串作分析

当前指针p,初始值0;

当前状态s,初始值rules[1][0];

\*/

struct RC{

int r;

int c;

};

//根据s得到该状态在规则表对应的行号r，根据sour.source[p]得到该符号在规则表中对应的列号c，新状态在rules[r][c]；返回值为新状态的行号和列号，如果为-1表示出错

struct RC getRC(int s,int p,int rules[R][C]);

bool parseSource(struct RC rc,int rules[R][C],int \*s);

int main()

{

struct CtoI CtoITable[4];

struct RC rc;

FILE \*fp;

int rules[R][C],p;

int s = rules[1][0];

init\_CtoITable(CtoITable);

init\_RulesTable(rules);

sour = readSource(fp,CtoITable);

p = 0;

for(p;p < sour.num;p++){

rc = getRC(s,p,rules);

if(!parseSource(rc,rules,&s)){

printf("读取的字符不符合规则表");

return 0;

}

}

printf("读取的字符符合规则表");

return 0;

}

struct sourceBuffer readSource(FILE \*fp,struct CtoI \*CtoITable){

struct sourceBuffer sour;

char file[20];

sour.num=0;

printf("Enter the file name:");

scanf("%s",file);

if((fp=fopen(file,"r"))==NULL){

printf("Can't open %s\n",file);

}else{

while(!feof(fp)){

sour.source[sour.num]=getIfromC(fgetc(fp),CtoITable);

sour.num++;

}

}

return sour;

}

void init\_CtoITable(struct CtoI CtoITable[]){

CtoITable[0].ch = 'a';

CtoITable[1].ch = 'b';

CtoITable[2].ch = 'c';

CtoITable[3].ch = 'd';

CtoITable[0].in = 201;

CtoITable[1].in = 202;

CtoITable[2].in = 203;

CtoITable[3].in = 204;

}

int getIfromC(char ch,struct CtoI \*CtoITable){

int a;

if(ch == 'a')

a = CtoITable[0].in;

else if(ch == 'b')

a = CtoITable[1].in;

else if(ch == 'c')

a = CtoITable[2].in;

else if(ch == 'd')

a = CtoITable[3].in;

return a;

}

void init\_RulesTable(int rules[R][C]){

rules[0][0] = -1;

rules[0][1] = 201;

rules[0][2] = 202;

rules[0][3] = 203;

rules[0][4] = 204;

rules[0][5] = -1;

rules[1][0] = 0;

rules[1][1] = 1;

rules[1][2] = 1;

rules[1][3] = -1;

rules[1][4] = -1;

rules[1][5] = -2;

rules[2][0] = 1;

rules[2][1] = 2;

rules[2][2] = -1;

rules[2][3] = 1;

rules[2][4] = -1;

rules[2][5] = -1;

rules[3][0] = 2;

rules[3][1] = -1;

rules[3][2] = -1;

rules[3][3] = 3;

rules[3][4] = 3;

rules[3][5] = -1;

rules[4][0] = 3;

rules[4][1] = 3;

rules[4][2] = -1;

rules[4][3] = -1;

rules[4][4] = -1;

rules[4][5] = -1;

}

struct RC getRC(int s,int p,int rules[R][C]){

struct RC rc;

int i = 1 , j= 1;

for(i;i<5;i++){

if(s == rules[i][0])

break;

}

for(j;j<5;j++){

if(sour.source[p] == rules[0][j])

break;

}

rc.c = j;

rc.r = i;

return rc;

}

bool parseSource(struct RC rc,int rules[R][C],int \*s){

if(rules[rc.r][rc.c] == -1 || rules[rc.r][rc.c] == -2)

return false;

else{

\*s = rules[rc.r][rc.c];

return true;

}

}

