简易词法分析程序

1. 词法分析程序采用标准C语言实现
2. 程序识别的标准保留字有:program function procedure array const file lable packed

var record set type of case do downto else for goto if repeat then to until while with

forward and not or in div mod begin end nil 36个、

1. 分析类别:标识符、无符号整数、单字符分界符、双字符分界符、注释头符号、单纯冒号、单纯斜竖

|  |  |  |
| --- | --- | --- |
| 单词名称 | 类别编码 | 记忆符 |
| program | 1 | PROGRAM |
| function | 2 | FUNCTION |
| procedure | 3 | PROCEDURE |
| array | 4 | ARRAY |
| const | 5 | CONST |
| file | 6 | \_FILE |
| lable | 7 | LABLE |
| packed | 8 | PACKED |
| var | 9 | VAR |
| record | 10 | RECORD |
| set | 11 | SET |
| type | 12 | TYPE |
| of | 13 | OF |
| case | 14 | CASE |
| do | 15 | DO |
| downto | 16 | DOWNTO |
| else | 17 | ELSE |
| for | 18 | FOR |
| goto | 19 | GOTO |
| if | 20 | IF |
| repeat | 21 | REPEAT |
| then | 22 | THEN |
| to | 23 | TO |
| until | 24 | UNTIL |
| while | 25 | WHILE |
| with | 26 | WITH |
| forward | 27 | FORWARD |
| and | 28 | AND |
| not | 29 | NOT |
| or | 30 | OR |
| in | 31 | IN |
| div | 32 | DIV |
| mod | 33 | MOD |
| begin | 34 | BEGIN |
| end | 35 | END |
| nil | 36 | NIL |
| 标识符 | 37 | ID |
| 整常数 | 38 | INT |
| + | 39 | ADD |
| - | 40 | MINUS |
| \* | 41 | STAR |
| / | 42 | DIVI |
| ( | 43 | LPAR |
| ) | 44 | RPAR |
| , | 45 | COMMA |
| ; | 46 | SEMI |
| : | 47 | COLON |
| := | 48 | ASSIGN |
| = | 49 | EQUAL |

注:注释识别匹配模式为：/\*…\*/

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <ctype.h>

#define MAX 4096 //设置输入字符串长度最大为4096

#define MAX\_WORD 66 //设置最大单词长度

#define REMAIN 36 //设置保留字的个数

//枚举类型,设置所有的类别编码

enum Sym {

NOTHING,

PROGRAM, FUNCTION, PROCEDURE, ARRAY, CONST,

\_FILE, LABLE, PACKED, VAR, RECORD,

SET, TYPE, OF, CASE, DO,

DOWNTO, ELSE, FOR, GOTO, IF,

REPEAT, THEN, TO, UNNTIL, WHILE,

WITH, FORWARD, AND, NOT, OR,

IN, DIV, MOD, BEGIN, END,

NIL, ID, INT, ADD, MINUS,

STAR, DIVI, LPAR, RPAR, COMMA,

SEMI, COLON, ASSIGN, EQUAL

};

const char \* remainSym[REMAIN] = {

"program","function","procedure","array","const",

"file" ,"lable" ,"packed" ,"var" ,"record",

"set" ,"type" ,"of" ,"case" ,"do",

"downto" ,"else" ,"for" ,"goto" ,"if",

"repeat" ,"then" ,"to" ,"until","while",

"with" ,"forward" ,"and" ,"not" ,"or",

"in" ,"div" ,"mod" ,"begin","end","nil"

};

char g\_char; //字符型全局变量

char g\_token[MAX\_WORD];//字符数组

int g\_num;//读入的整型数值

enum Sym g\_symbol;//当前识别的单词类型

char coding[MAX];//输入的源代码

int index;//coding数组的游标

/\*functions\*/

void getSym(); //识别组合单词符号

void initial();//初始化全局变量,作为每一次调用getSym的先行函数

int reserve();//查找是否保留字

void error() {//程序分析出错处理-->代表源程序词法不合法

printf("抱歉,词法分析发现存在不合法内容,程序终止\n");

exit(EXIT\_SUCCESS);

}

char getNextChar() {//获取一个char

return coding[index++];

}

void retract() {//回退

index--;

}

/\*functions\*/

int main(void) {

printf("Hello,欢迎使用简易词法分析程序,请输入你所要分析的源代码(输入终止符表示结束ctrl-D/ctrl-Z)\n");

int ch;

while ((ch = getchar()) != EOF) {

coding[index++] = ch;

}

index = 0;

while (getNextChar() != '\0') {

retract();

getSym();

if (g\_symbol == NOTHING)

continue;

else if (g\_symbol > NOTHING && g\_symbol <= NIL)

printf("识别出保留字:%s\n", remainSym[g\_symbol - 1]);

else if (g\_symbol == ID)

printf("识别出标识符:%s\n", g\_token);

else if (g\_symbol == INT)

printf("识别出整常数:%d\n", g\_num);

else if (g\_symbol == ADD)

printf("识别出+\n");

else if (g\_symbol == MINUS)

printf("识别出-\n");

else if (g\_symbol == STAR)

printf("识别出\*\n");

else if (g\_symbol == DIVI)

printf("识别出/\n");

else if (g\_symbol == LPAR)

printf("识别出(\n");

else if (g\_symbol == RPAR)

printf("识别出)\n");

else if (g\_symbol == COMMA)

printf("识别出,\n");

else if (g\_symbol == SEMI)

printf("识别出;\n");

else if (g\_symbol == COLON)

printf("识别出:\n");

else if (g\_symbol == ASSIGN)

printf("识别出赋值符号:=\n");

else if (g\_symbol == EQUAL)

printf("识别出相等符号=\n");

}

printf("分析结束,谢谢使用\n");

return 0;

}

void getSym() {

initial(); //初始化

g\_char = getNextChar();

while (isspace(g\_char))

g\_char = getNextChar();

if (isalpha(g\_char)) {

while (isalnum(g\_char)) {

strcat(g\_token, &g\_char);

g\_char = getNextChar();

}

retract();

int resultValue = reserve();

if (resultValue == 0) g\_symbol = ID;

else g\_symbol = resultValue;

}

else if (isdigit(g\_char)) {

while (isdigit(g\_char)) {

strcat(g\_token, &g\_char);

g\_char = getNextChar();

}

retract();

g\_num = atoi(g\_token);

g\_symbol = INT;

}

else if (g\_char == ':') {

g\_char = getNextChar();

if (g\_char == '=') g\_symbol = ASSIGN;

else {

retract();

g\_symbol = COLON;

}

}

else if (g\_char == '=') g\_symbol = EQUAL;

else if (g\_char == '+') g\_symbol = ADD;

else if (g\_char == '-') g\_symbol = MINUS;

else if (g\_char == '\*') g\_symbol = STAR;

else if (g\_char == '(') g\_symbol = LPAR;

else if (g\_char == ')') g\_symbol = RPAR;

else if (g\_char == ',') g\_symbol = COMMA;

else if (g\_char == ';') g\_symbol = SEMI;

else if (g\_char == '/') {

g\_char = getNextChar();

if (g\_char == '\*') {

while (1) {

g\_char = getNextChar();

if (g\_char == '\*') {

g\_char = getNextChar();

if (g\_char == '/')

break;

else

continue;

}

}

}

else {

retract();

g\_symbol = DIVI;

}

}

else error();

}

void initial() {

g\_char = '\0';

memset(g\_token, 0, MAX\_WORD \* sizeof(char));

g\_num = -1;

g\_symbol = NOTHING; //设置为0,不在单词类别编码表内

}

int reserve() {

int i;

for (i = 0; i < REMAIN; i++) {

if (\_stricmp(remainSym[i], g\_token) == 0)

return (i + 1);

}

return 0;

}