实验编号：1**四川师大 编译原理 实验报告 2020**年**4**月**12**日

**计算机科学学院** 2017级 4 班 实验名称：词法分析程序

姓名：\_韩勇\_\_ 学号：\_2017110409\_\_ 指导老师：\_\_吴贞东\_\_ 实验成绩:\_\_\_\_\_

**实验 一 词法分析程序\_**

1. 实验目的及要求

通过本实验的编程实践，使学生了解词法分析的任务，掌握词法分析程序设 计的原理和构造方法，使学生对编译的基本概念、原理和方法有完整的和清楚的理 解，并能正确地、熟练地运用。输入的常量说明串，要求最后以分号作结束标志；

1、编程时注意编程风格：空行的使用、注释的使用、缩进的使用等。

2、将标识符填写的相应符号表须提供给编译程序的以后各阶段使用。

3、根据测试数据进行测试。测试实例应包括以下三个部分: 全部合法的输入。 各种组合的非法输入。 由记号组成的句子。

4、词法分析程序设计要求输出形式:

1. 实验内容  
   （1）从源程序文件中读入字符。

（2）统计行数和列数用于错误单词的定位。

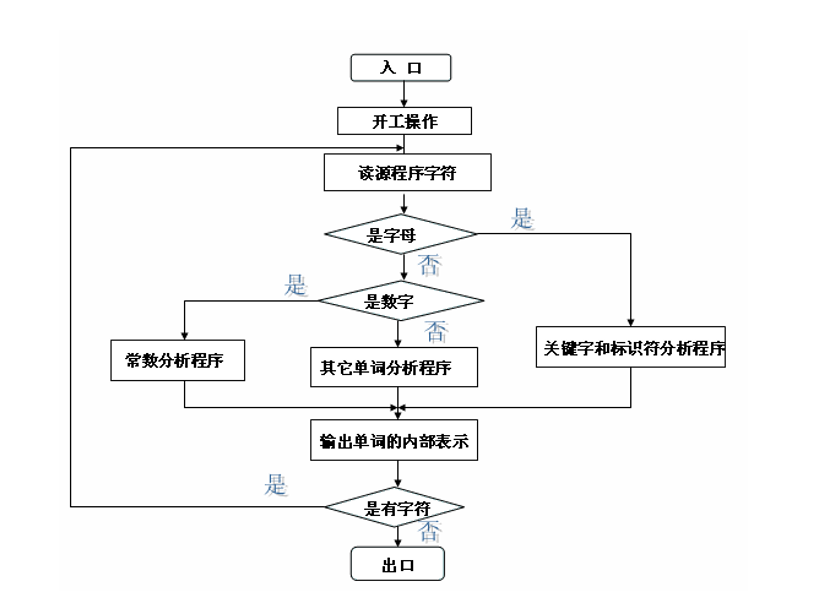
（3）删除空格类字符，包括回车、制表符空格。

（4）按拼写单词，并用（内码，属性）二元式表示。(属性值——token 的机内 表示)

（5）如果发现错误则报告出错 7

（6）根据需要是否填写标识符表供以后各阶段使用实验主要流程、基本操作或核心代码、算法片段（该部分如不够填写，请另加附页）

流程图：



代码：

//头文件

#include <iostream>

#include <map>

#include <algorithm>

#include <string>

using namespace std;

string instr;//输入符号串

int index;//当前输入符号读入字符的位置

char character;//全局变量字符，存放最新读入的字符

string token;//字符数组，存放已读入的字符序列

map<string, int> Symbol;//标识符集

map<string, int> Digit;//常数集

map<string, int>::iterator ite;

const int len = 100;

string Reserve[len];//保留字表

string Boundary[2\*len];//界符

string Operator[3 \* len];//运算符

struct Binary {

Binary(int c,int i,string v="-") {

category = c;

index = i;

value = v;

}

int category=0;

int index=0;

string value="-";

};

void init\_Reserve() {//构造保留字表的函数

Reserve[1] = "main";

Reserve[2] = "int";

Reserve[3] = "if";

Reserve[4] = "else";

Reserve[5] = "while";

Reserve[6] = "for";

Reserve[7] = "read";

Reserve[8] = "write";

Reserve[9] = "bool";

Reserve[10] = "break";

Reserve[11] = "case";

Reserve[12] = "catch";

Reserve[13] = "char";

Reserve[14] = "class";

Reserve[15] = "const";

Reserve[16] = "continue";

Reserve[17] = "default";

Reserve[18] = "delete";

Reserve[19] = "do";

Reserve[20] = "double";

Reserve[21] = "enum";

Reserve[22] = "false";

Reserve[23] = "true";

Reserve[24] = "float";

Reserve[25] = "friend";

Reserve[26] = "goto";

Reserve[27] = "inline";

Reserve[28] = "long";

Reserve[29] = "new";

Reserve[30] = "private";

Reserve[31] = "protected";

Reserve[32] = "public";

Reserve[33] = "return";

Reserve[34] = "short";

Reserve[35] = "signed";

Reserve[36] = "sizeof";

Reserve[37] = "static";

Reserve[38] = "struct";

Reserve[39] = "switch";

Reserve[40] = "this";

Reserve[41] = "try";

Reserve[42] = "typedef";

Reserve[43] = "unsigned";

Reserve[44] = "using";

Reserve[45] = "virtual";

Reserve[46] = "void";

Reserve[47] = "include";

Reserve[48] = "iostream";

Reserve[49] = "namespace";

Reserve[50] = "std";

}

void init\_Operator() {//初始化运算符表

Operator[210] = "+";

Operator[211] = "-";

Operator[212] = "\*";

Operator[213] = "/";

Operator[214] = "<";

Operator[215] = "<=";

Operator[216] = ">";

Operator[217] = ">=";

Operator[218] = "!=";

Operator[219] = "==";

Operator[220] = "=";

}

void init\_Boundary() {//界符表初始化

Boundary[121] = "(";

Boundary[122] = ")";

Boundary[123] = ",";

Boundary[124] = ";";

Boundary[125] = "{";

Boundary[126] = "}";

Boundary[127] = "#";

}

void getChar() {//读入一个字符

character = instr[index++];

}

void getnbc() {//读入非空白字符

while (character == ' ') {

getChar();

}

}

void concat() {//连接字符串

token = token + character;

}

bool letter() {//判断是否为字母

if ((character >= 'A'&&character <= 'Z') || (character >= 'a'&&character <= 'z'))

return true;

return false;

}

bool digit() {//判断是否为数字

if (character >= '0'&&character <= '9')

return true;

return false;

}

void retract(){//回退字符的函数

character = ' ';

index--;

}

int reserve() {//匹配保留字符

for (int i = 0; i < len; i++)

if (Reserve[i] == token)return i;

return -1;

}

string symbol() {

ite = Symbol.find(token);

if (ite != Symbol.end()) {

return ite->first;

}

else {

Symbol[token] = Symbol.size();

return token;

}

}

string constant() {

ite = Digit.find(token);

if (ite != Digit.end()) {

return ite->first;

}

else {

Digit[token] = Digit.size();

return token;

}

}

Binary error() {

cout << token << "\t-->\t该单词不存在" << endl;

return Binary(0, 0);

}

//词法分析函数，逐个识别单词

Binary LexAnalyze() {

token = "";

getChar();

getnbc();

string val;

int num = -1;

switch (character) {

case'a':

case'b':

case'c':

case'd':

case'e':

case'f':

case'g':

case'h':

case'i':

case'j':

case'k':

case'l':

case'm':

case'n':

case'o':

case'p':

case'q':

case'r':

case's':

case't':

case'u':

case'v':

case'w':

case'x':

case'y':

case'z':

case'A':

case'B':

case'C':

case'D':

case'E':

case'F':

case'G':

case'H':

case'I':

case'J':

case'K':

case'L':

case'M':

case'N':

case'O':

case'P':

case'Q':

case'R':

case'S':

case'T':

case'U':

case'V':

case'W':

case'X':

case'Y':

case'Z':

while (letter() || digit()) {//为字母或数字

concat();//追加到token末尾

getChar();//读取下一个字符

}

retract();//回退一个字符

num = reserve();//查看保留字表

if (num != -1) {

return Binary(num,0);

}

else {

val = symbol();//查看标识符表

return Binary(1, Symbol[val],val);

}

break;

case'0':

case'1':

case'2':

case'3':

case'4':

case'5':

case'6':

case'7':

case'8':

case'9':

while (digit()) {//为数字

concat();

getChar();

}

retract();

val = constant();//查看常数表

return Binary(2,Digit[val],val);

break;

case'<':

getChar();

if (character == '=') return Binary(215,0);//返回<=符号

else {

retract();

return Binary(214, 0);//返回<符号

}

break;

case'>':

getChar();

if (character == '=')return Binary(217, 0);//返回>=符号

else {

retract();

return Binary(216, 0);//返回>符号

}

break;

case'=':

getChar();

if (character == '=') return Binary(219, 0);//返回==符号

else {

retract();

return Binary(220, 0);//返回=符号

}

break;

case'!':

getChar();

if (character == '=')return Binary(218, 0);

else return error();

break;

case'+':

return Binary(210, 0);

break;

case'-':

return Binary(211, 0);

break;

case'\*':

return Binary(212, 0);

break;

case'/':

return Binary(213, 0);

break;

case'(':

return Binary(121, 0);

break;

case')':

return Binary(122, 0);

break;

case',':

return Binary(123, 0);

break;

case';':

return Binary(124, 0);

break;

case'{':

return Binary(125, 0);

break;

case'}':

return Binary(126, 0);

break;

case'#':

return Binary(127, 0);

break;

default:

return error();

}

}

void show\_table() {

cout << "=================="<<"保留字"<<"==================" << endl;

cout << "保留字符\t类别编码" << endl;

for (int i = 0; i < len; i++) {

if (Reserve[i] != "") {

if(Reserve[i].size()>=8)

cout << Reserve[i] << "\t" << i << endl;

else

cout << Reserve[i] << "\t\t" << i << endl;

}

}

cout << "\n==================" << "界符" << "==================" << endl;

cout << "界符\t\t类别编码" << endl;

for (int i = 0; i < 2 \* len; i++) {

if (Boundary[i] != "") {

cout << Boundary[i] << "\t\t" << i << endl;

}

}

cout << "\n==================" << "运算符" << "==================" << endl;

cout << "运算符\t\t类别编码" << endl;

for (int i = 0; i < 3 \* len; i++) {

if (Operator[i] != "") {

cout << Operator[i] << "\t\t" << i << endl;

}

}

cout << "\n==================" << "标识符" << "==================" << endl;

cout << "标识符\t\t类别编码\t表中位置" << endl;

for (ite=Symbol.begin(); ite != Symbol.end(); ite++) {

if(ite->first.size()>=8)

cout << ite->first<< "\t1\t\t" << ite->second << endl;

else

cout << ite->first << "\t\t1\t\t" << ite->second << endl;

}

cout << "\n==================" << "常数表" << "==================" << endl;

cout << "常量值\t\t类别编码\t表中位置" << endl;

for (ite = Digit.begin(); ite != Digit.end(); ite++) {

cout << ite->first << "\t\t2\t\t" << ite->second << endl;

}

}

int main() {

init\_Reserve();//保留字表初始化

init\_Boundary();//界符表初始化

init\_Operator();//运算符表初始化

Symbol.clear();//标识符集初始化

Digit.clear();//常数集初始化

index = 0;

character = ' ';

token = "";

cout << "请输入待词法分析的源程序代码：输入@代表结束输入\n" << endl;

//源程序代码输入处理

string in;

while (cin >> in&&in != "@") {

instr = instr + " " + in;

}

//识别二元组初始化

Binary word(0,0,"-");

//循环进行词法分析直到识别所有单词符号

cout << "\n------------------------识别结果------------------------" << endl;

while (index < instr.size()) {

word=LexAnalyze();

if (word.category != 0) {

cout << "识别单词:\t(" << word.category << "," << word.value << ")" << endl;

}

}

//展示构造的各种词汇表

cout << "\n------------------------词汇表展示------------------------\n" << endl;

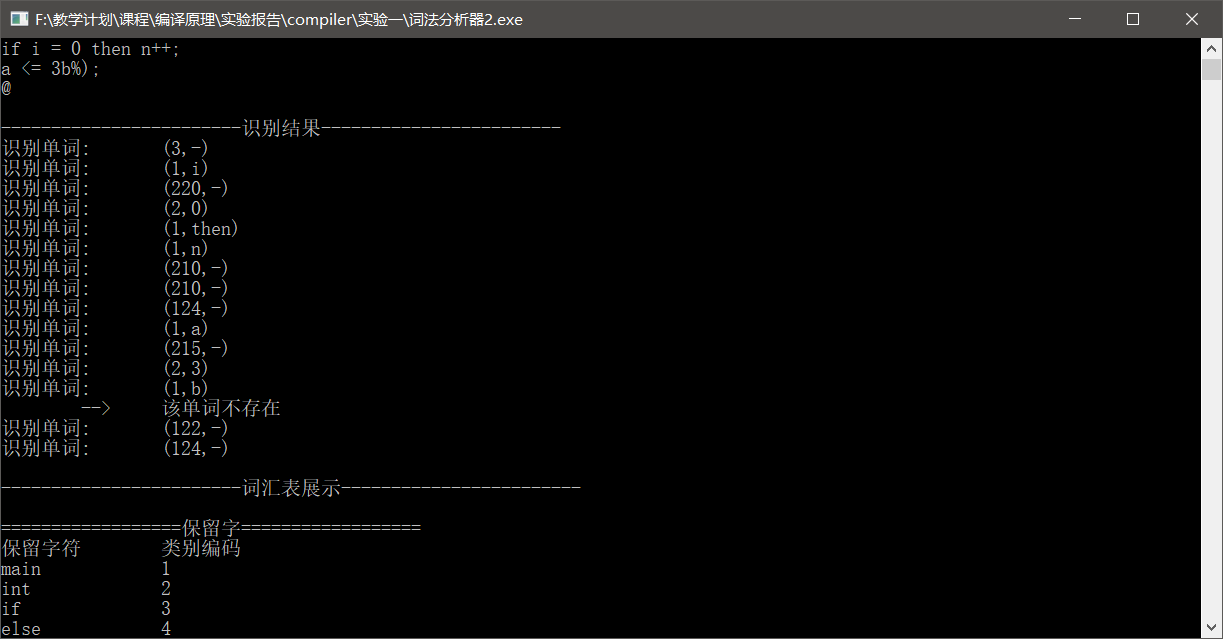
show\_table();

//system("pause");

return 0;

}

运行结果：





1. 实验结果的分析与评价（该部分如不够填写，请另加附页）

通过实验更好的了解了语言运行的根本，有利于我们以后更好的进行相关方面的学习。

注：实验成绩等级分为（90－100分）优，（80－89分）良，(70-79分)中，（60－69分）及格，（59分）不及格。