**软件学院实践报告（编译原理）**

**课程编号： 实践课程名称：编译原理 学年：2018-2019 学期：秋**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **学生姓名** | | 吴志镛，陈凌云，徐德全，徐梓耀，门越聪 | **学号** | 2017211869，2017211868，2017211858，201752204，2017211878 |
| **指导教师姓名** | | 修佳鹏 | 完成时间 | 2019年10月22日 |
| **实验名称** | | 词发分析器 | | |
| **实验目的** | 通过对词法的分析，以及状态机的构造知识的学习，编写一个词法编译器，能够简单的分析代码的标识符、数字和分隔符等，并且能够将存在的错误进行简单的提示。 | | | |
| **实验要求** | 1. 根据附录给定的文法，从输入的类 C 语言源程序中，识别出各个具有独立意义的单词，即标识符、运算符、分隔符等 2. 词法分析后可查看输出的符号表 3. 保存符号表和 TOKEN 串表（如：文本文件） 4. 使用LEX程序编写词发分析器   遇到错误时可显示提示信息 | | | |
| **实验环境** | 1. C语言编写 windows10系统 Visual Studio2017 2. C++语言编写 windows10系统 Visual Studio2017 3. LEX程序编写 Windows10系统 Parser Generator编写 flex编译 | | | |
| **人员分工** | 主要工程：使用C++编写编写 C法编译器  分工如下：（本小组为了确保每个人都能学会如何编译词法分析器，于是在组中又分成两个部分，吴志镛和陈凌云使用C语言实现词法分析器并研究LEX程序，徐德全，门越聪，徐梓耀负责使用c++实现词法分析器）  吴志镛负责整体的C语言的架构编写（包含主函数scan之间分流，识别单行注释，多行注释，预处理语句等）以及对LEX程序的尝试编写。  陈凌云负责C语言编译器中的识别整数和浮点数的功能实现，基础函数的是实现统计结果等。  徐德全负责实现C++程序里面的主题架构。  门越聪负责实现C++程序里面DFA对象的建立  徐梓耀负责实现DFA的一些基本方法。 | | | |
| **设计方案** | C/C++语言的设计方案如下：  首先我们先找出C语言词发分析器需要识别的各个部分明确我们的任务，我们的任务是要将我们识别的部分放到Token里面并还需要计算总共有多少个字符，多少行以及识别出来的各个种类的记号的个数，除此之外我们还需要识别出非法字符。  明确了我们任务之后，我们对任务中不同的方面实现如下，首先总的框架中，为了识别出不同记号，我们采用switch的架构，每一种情况都用一个case去分支，如果case中还需要区分的话,那么在从case里面套一个switch去来区分不同种类的记号。  之后我们使用Token这样一个数组粗放我们扫描的符号，每次识别出来一个记号就使用变量记录下来，这样就确保我们可以记录识别的符号总数以及不同种类记号的数目，至于如何算有多上行，这就很简单了，直接可以扫描文件的是得到。最后时候关于非法字符的判断，我们把我们在整体的switch中，除了可以识别的部分，那么不能识别的部分就单独放一个case，说明出现了违法字符。  LEX的程序设计：  LEX程序自已提供了一套词法分析的功能，所以我们编写LEX程序的时候只需要使用正则表达式把我们要识别的部分列出来，然后在主程序的进行识别匹配就好了。 | | | |
| **成源程序** | 详见附录 | | | |
| **实验结果** | C语言和C++都可以做到完整的识别出C语言的所有记号，Lex程序的注释部分识别需要完善。结果如下：  控制台打印出总的情况： 测试代码：    Txt文件中存放记号列表： | | | |
| **实验分析** | 实验要求的功能都已经全部实现，并且新增加了对非法字符的识别，预处理语句以及注释识别，而且尝试了c/c++/lex空三种的语言的编写，我们产生了三个词法分析器，实验结果很丰富，虽然lex程序并不是很完美但是有的功能还是拥有了，c语言编写程序比较完美，我个人认为是因为人数的原因，因为这作业个人觉得难度不大，以后可以考虑一个人写一个这样有助于大家都增长知识。 | | | |
| **心得体会** | 在这次实验中，我深切地体会到词法编译器的具体实现是如何进行的，将大量的不同的字符串进行分析是一件辛苦的事情，很难想象早期计算机科学家在对第一个代码程序进行分析时，经历过怎样的苦苦思索。除此之外，我对于lex程序有了一定的了解，因为lex程序是基于unix的系统，他的编写也很符合unix的系统的风格。 | | | |

程序附录：

#pragma once

#include <stdio.h>

#include <stdlib.h>

//判断输入的字符是否为数字

int isDigit(char ch);

//判断输入的字符是否为字母（大写+小写）

int isLetter(char ch);

//判断输入的字符是否为空白符

int isBlank(char ch);

#include "Conio.h"

//判断输入的字符是否为数字

int isDigit(char ch) {

if (ch >= '0' && ch <= '9') {

return 1;

}

else {

return 0;

}

}

//判断输入的字符是否为字母（大写+小写）

int isLetter(char ch) {

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

return 1;

}

else {

return 0;

}

}

//判断输入的字符是否为空白符

int isBlank(char ch) {

if (ch == ' ' || ch == '\t' || ch == '\b' || ch == '\n'){

return 1;

}

else {

return 0;

}

}

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <math.h>

#include "Conio.h"

/\*\*\*\*\*\*全局变量定义\*\*\*\*\*\*\*/

FILE\* readp, \* writep; //文件指针

char token[30] = { 0 }; //存放每次扫描的结果

char keylist[32][20] = { //C语言的32个关键字，按照字典序升序排列

"auto", "break", "case", "char", "const", "continue",

"default", "do", "double", "else", "enum", "extern",

"float", "for", "goto", "if", "int", "long",

"register", "return", "short", "signed", "sizeof", "static",

"struct", "switch", "typedef", "union", "unsigned", "void",

"volatile", "while"

};

char symbolList[100][5] = { //运算符和分界符的表

"+", "-", "\*", "/", "<", "<=", ">", ">=", "=", "==",

"!=", "<<", ">>", "++", "--", "+=", "-=", "\*=", "/=",

"&=", "|=", ";", "(", ")", "^", ",", "\"", "\'", "&",

"&&", "|", "||", "%", "~", "<<", ">>", "[", "]", "{",

"}", "\\", ".", "\?", ":", "!"

};

char identifierTable[500][30] = { 0 }; //标识符表，默认为空

int identifierNum = 0; //标识符总数

int lineCount = 1; //记录代码总行数

int charCount = -1; //记录字符总数

int wordCount = 0; //记录单词的个数

int stringCount = 0;

int numCount = 0; //记录常数的个数

int operatorCount = 0; //记录运算符的个数

int delimiterCount = 0; //记录界符的个数

char getChar(); //从文件中读入一个字符

void scan(); //主函数，进行记号的识别

char scanFloat(char ch); //子功能函数，识别整数和浮点数

char scanString(char ch); //子功能函数，识别字符串结构

int isCKey(); //子功能函数，判断一个字符串是否为C语言的关键字，如果是，返回该关键字在表中的位置，否则返回-1

void clearToken(); //清除程序中的缓冲区

int insertID(); //向标识符表中加入新的标识符，并返回标识符的地址（如果已经插入，则不进行插入操作）

int findSymbol(char str[]); //在运算符和分界符表中查找符号，并返回在表中的地址

void printResult(); //在控制台中打印统计结果

int main() {

char ch;

char str[10] = "asfa\e\n\5";

//从文件中读入源代码

if ((readp = fopen("SourceCode.txt", "r")) == NULL) {

printf("file cannot open \n");

exit(0);

}

if ((writep = fopen("TokenStream.txt", "w")) == NULL) {

printf("file cannot open \n");

exit(0);

}

scan();

printResult();

if (fclose(readp) != 0) {

printf("file cannot be closed \n");

}

if (fclose(writep) != 0) {

printf("file cannot be closed \n");

}

system("pause");

return 0;

}

//主函数，进行记号的识别

void scan() {

char ch;

ch = getChar();

while (ch != EOF) {

//ch是空白符：跳过所有的空白符

if (isBlank(ch)) {

while (isBlank(ch = getChar()));

}

//ch是字母：标识符 or 关键字

else if (isLetter(ch)) {

//扫描出完整的标识符

int p = 0;

do {

token[p++] = ch;

ch = getChar();

} while (isDigit(ch) || isLetter(ch) || ch == '\_');

wordCount++;

//判断标识符token是否为C语言关键字

int addr;

if ((addr = isCKey()) != -1) {

fprintf(writep, "%s --- 关键字%d\n", token, addr);

}

else {

//如果是标识符，将其插入符号表中

addr = insertID();

fprintf(writep, "%s --- 标识符%d\n", token, addr);

}

clearToken();

}

//ch是数字：整数 or 浮点数

else if (isDigit(ch)) {

ch = scanFloat(ch);

clearToken();

}

else if (ch == '\"') {

ch = scanString(ch);

}

//ch是其他字符：运算符 or 界符

else if(ch == '+' || ch == '-' || ch == '\*' || ch == '/' || ch == ';' || ch == '(' || ch == ')' || ch == '^'

|| ch == ',' || ch == '\'' || ch == '~' || ch == '#' || ch == '%' || ch == '['

|| ch == ']' || ch == '{' || ch == '}' || ch == '\\' || ch == '.' || ch == '\?' || ch == ':'

|| ch == '=' || ch == '<' || ch == '>' || ch == '!' || ch == '&' || ch == '|'){

//对ch进行分情况处理

switch (ch) {

case '#': {

//预处理部分的内容，因此本行跳过

while (1) {

ch = getChar();

if (ch == '\n' || ch == '\b') {

break;

}

}

ch = getChar();

break;

}

case '/': {

ch = getChar();

switch (ch) {

case '/': {

//单行注释结构，跳过该行内容

while (1) {

ch = getChar();

if (ch == '\n' || ch == '\b') {

break;

}

}

//ch = getChar();

break;

}

case '\*': {

//多行注释结构，跳过后面的所有代码，直到出现'\*/'

int flag = 0; //flag用来标识上一个字符是否为\*号

while (1) {

ch = getChar();

if (flag && ch == '/') {

//结束注释识别

break;

}

if (ch == '\*') {

flag = 1;

}

else {

flag = 0;

}

}

//ch = getChar();

break;

}

case '=': {

char str[] = "/=";

operatorCount++;

fprintf(writep, "/= --- 运算符%d\n", findSymbol(str));

//ch = getChar();

break;

}

default: {

char str[] = "/";

operatorCount++;

fprintf(writep, "/ --- 运算符%d\n", findSymbol(str));

break;

}

}

ch = getChar();

break;

}

case '=':

case '\*':

case '%':

case '!': {

//识别= \* % ! == \*= %= !=

token[0] = ch;

ch = getChar();

if (ch == '=') {

token[1] = ch;

ch = getChar();

}

operatorCount++;

fprintf(writep, "%s --- 运算符%d\n", token, findSymbol(token));

clearToken();

break;

}

case '<':

case '>':

case '+':

case '-': {

//识别++ -- += -= + -

token[0] = ch;

ch = getChar();

if (token[0] == ch || ch == '=') {

token[1] = ch;

ch = getChar();

}

operatorCount++;

fprintf(writep, "%s --- 运算符%d\n", token, findSymbol(token));

clearToken();

break;

}

case '&':

case '|': {

//识别&& || &= |= & | << >> <= >= < >

token[0] = ch;

ch = getChar();

if (token[0] == ch || ch == '=') {

token[1] = ch;

ch = getChar();

}

operatorCount++;

fprintf(writep, "%s --- 运算符%d\n", token, findSymbol(token));

clearToken();

break;

}

case '~':

case '^': {

//其他只由一个字符组成的运算符

char str[2] = { ch };

operatorCount++;

fprintf(writep, "%s --- 运算符%d\n", str, findSymbol(str));

ch = getChar();

break;

}

default: {

//剩余的均为界符，且均只由一个字符组成

char str[2] = { ch };

delimiterCount++;

fprintf(writep, "%s --- 界符%d\n", str, findSymbol(str));

ch = getChar();

break;

}

}

}

//出现C语言不允许的非法字符：报错

else {

printf("WARNING: 第 %d 行出现非法字符，忽略后继续分析...\n", lineCount);

while (!isBlank(ch = getChar()));

}

}

}

//子功能函数，识别整数和浮点数

char scanFloat(char ch){

//将整数部分扫描出来，放到缓冲区中

int p = 0;

do {

token[p++] = ch;

ch = getChar();

} while (isDigit(ch));

if (isLetter(ch) && ch !='e' && ch != 'E') {

printf("WARNING: 第 %d 行出现非法字符，忽略后继续分析...\n", lineCount);

while (!isBlank((ch = getChar())));

return ch;

}

//num1存放整数部分，num2存放小数部分，num3存放指数部分

int num1 = 0, num3 = 0;

double num2 = 0;

for (int i = 0; i < strlen(token); i++) {

char temp = token[i];

num1 = num1 \* 10 + (temp - '0');

}

clearToken();

if (ch == '.') { //识别到小数点，说明该数有小数部分

//将小数部分扫描出来，放到缓冲区中

int p = 0;

while (isDigit((ch = getChar()))) {

token[p++] = ch;

}

if (p == 0) {

//小数点后面没有数字，报错

printf("WARNING: 第 %d 行出现非法字符，忽略后继续分析...\n", lineCount);

while (!isBlank((ch = getChar())));

return ch;

}

else {

for (int i = strlen(token) - 1; i >= 0; i--) {

char temp = token[i];

num2 = (num2 + (temp - '0')) / 10.0;

}

clearToken();

}

}

if (ch == 'e' || ch == 'E') {

//指数部分

int signal = 1;

int p = 0;

ch = getChar();

if (ch == '+' || ch == '-') {

if (ch == '-') {

signal = -1;

}

ch = getChar();

if (!isDigit(ch)) {

printf("WARNING: 第 %d 行出现非法字符，忽略后继续分析...\n", lineCount);

while (!isBlank((ch = getChar())));

return ch;

}

else {

token[p++] = ch;

}

}

else if (isDigit(ch)) {

token[p++] = ch;

}

else {

printf("WARNING: 第 %d 行出现非法字符，忽略后继续分析...\n", lineCount);

while (!isBlank((ch = getChar())));

return ch;

}

while (isDigit((ch = getChar()))) {

token[p++] = ch;

}

for (int i = 0; i < strlen(token); i++) {

char temp = token[i];

num3 = num3 \* 10 + (temp - '0');

}

num3 = num3 \* signal;

}

clearToken();

numCount++;

if (num2 == 0) {

int num = num1 \* (int)(pow(10, num3));

fprintf(writep, "%d --- 整常数\n", num);

}

else {

double num = (num1 + num2) \* pow(10, num3);

fprintf(writep, "%.2f --- 浮点数\n", num);

}

return ch;

}

char scanString(char ch){

//将整数部分扫描出来，放到缓冲区中

int p = 0;

token[p++] = ch;

while (1) {

ch = getChar();

token[p++] = ch;

if (ch == '\"') {

break;

}

}

stringCount++;

fprintf(writep, "%s --- 字符串\n", token);

clearToken();

ch = getChar();

return ch;

}

int insertID() {

int isExist = 0;

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

if (strlen(token) == strlen(identifierTable[i])) {

int flag = 1;

for (int j = 0; j < strlen(token); j++) {

if (token[j] != identifierTable[i][j]) {

flag = 0;

break;

}

}

if (flag) {

isExist = 1;

return i;

}

}

}

//如果token未在表中，将其插入表中

if (!isExist) {

for (int i = 0; i < strlen(token); i++) {

identifierTable[identifierNum][i] = token[i];

}

identifierNum++;

return identifierNum - 1;

}

return -1;

}

int findSymbol(char str[]){

for (int i = 0; i < 50; i++) {

if (strlen(str) == strlen(symbolList[i])) {

int flag = 1;

for (int j = 0; j < strlen(str); j++) {

if (str[j] != symbolList[i][j]) {

flag = 0;

break;

}

}

if (flag) {

return i;

}

}

}

return -1;

return 0;

}

void printResult(){

printf("\n\n\n/---------统计结果----------/\n");

printf("代码总行数: %d\n", lineCount);

printf("代码中字符总数: %d\n\n", charCount);

printf("代码中每种记号的个数：\n");

printf("\t单词: %d\n", wordCount);

printf("\t字符串: %d\n", stringCount);

printf("\t无符号数: %d\n", numCount);

printf("\t运算符: %d\n", operatorCount);

printf("\t界符: %d\n", delimiterCount);

printf("/---------标识符表----------/\n");

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

printf("%2d ------- %s\n", i, identifierTable[i]);

}

}

int isCKey() {

for (int i = 0; i < 32; i++) {

if (strlen(token) == strlen(keylist[i])) {

int flag = 1;

for (int j = 0; j < strlen(token); j++) {

if (token[j] != keylist[i][j]) {

flag = 0;

break;

}

}

if (flag) {

return i;

}

}

}

return -1;

}

char getChar() {

char ch = fgetc(readp);

if (!isBlank(ch) && ch != EOF) {

charCount++;

}

if (ch == '\n' || ch == '\b') {

lineCount++;

}

return ch;

}

void clearToken() {

int l = strlen(token);

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

token[i] = 0;

}

}

LEX程序：

%{

#include <stdio.h>

#include <stdlib.h>

int count = 0;

%}

delim [" "\n\t]

whitespace {delim}+

operator \+|-|\\*|\/|:=|>=|<=|#|=

reservedWord [cC][oO][nN][sS][tT]|[vV][aA][rR]|[pP][rR][oO][cC][eE][dD][uU][rR][eE]|[bB][eE][gG][iI][nN]|[eE][nN][dD]|[iI][fF]|[tT][hH][eE][nN]|[wW][hH][iI][lL][eE]|[dD][oO]|[rR][eE][aA][dD]|[cC][aA][lL][lL]|[wW][rR][iI][tT][eE]|[wW][rR][iI][tT][eE][lL][nN]

delimiter [,\.;\(\)]

constant ([0-9])+

identfier [A-Za-z]([A-Za-z][0-9])\*

%%

{reservedWord} {count++;printf("%d\t(1,‘%s’)\n",count,yytext);}

{operator} { count++;printf("%d\t(2,‘%s’)\n",count,yytext); }

{delimiter} {count++;printf("%d\t(3,‘%s’)\n",count,yytext);}

{constant} {count++;printf("%d\t(4,‘%s’)\n",count,yytext);}

{identfier} {count++;printf("%d\t(5,‘%s’)\n",count,yytext);}

{whitespace} { /\* do nothing\*/ }

%%

void main()

{

printf("词法分析器输出类型说明:\n");

printf("1：保留字\n");

printf("2：运算符\n");

printf("3：分界符\n");

printf("4：常 数\n");

printf("5：标识符\n");

printf("\n");

yyin=fopen("example.txt","r");

yylex(); /\* start the analysis\*/

fclose(yyin);

system("PAUSE");/\*暂停停, 使DOS窗口停住\*/

}

int yywrap()

{

return 1;

}