**软件学院实践报告（编译原理）**

**课程编号： 3152100131 实践课程名称：编译原理 学年：2018-2019 学期：秋**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **学生姓名** | | 白烨淞 | **学号** | 2016522050 |
| **指导教师姓名** | | 修佳鹏 | 完成时间 | 2018.11.11 |
| **实验名称** | | **词法编译器** | | |
| **实验目的** | 为了让学生更加清楚了解词法分析的过程，通过动手实践，使学生对构造编译系统的基本理论、编译程序的基本结构有更为深入的理解和掌握；使学生掌握编译程序设计的基本方法和步骤；能够设计实现编译系统的重要环节。同时增强编写和调试程序的能力。 | | | |
| **实验要求** | 1. 对单词的构词规则有明确的定义；  2.编写的分析程序能够正确识别源程序中的单词符号； 识别出的单词以<单词符号，种别码>的形式保存在符号表中； | | | |
| **实验环境** | Windows10  VS2017 | | | |
| **设计方案** |  | | | |
| **成源程序** | (详见附录) | | | |
| **实验结果** | C:\Users\BYS\AppData\Roaming\Tencent\Users\1198804523\TIM\WinTemp\RichOle\}K8N20FLZD5TG`X2VDEO0SM.png   C:\Users\BYS\AppData\Roaming\Tencent\Users\1198804523\TIM\WinTemp\RichOle\16$0][85SP~SGBY]3EGX3@1.png  C:\Users\BYS\AppData\Roaming\Tencent\Users\1198804523\TIM\WinTemp\RichOle\SXGK]NE~9SQ2PB[U(VQEU10.png | | | |
| **实验分析** | 1. 词法分析程序打开源文件，读取文件内容，直至遇上’$’文件结束符，然后读取结束。  2. 对读取的文件进行预处理，从头到尾进行扫描，和/\*  \*/的内容，以及一些无用的、影响程序执行的符号如换行符、回车符、制表符等。  3. 对源文件从头到尾进行扫描了，从头开始扫描，这个时候扫描程序首先要询问当前的字符是不是空格，若是空格，则继续扫描下一个字符，直至不是空格，然后询问这个字符是不是字母，若是则进行标识符和保留字的识别；若这个字符为数字，则进行数字的判断并使用一个嵌套循环来检测命名错误。否则，依次对这个字符可能的情况进行判断，若是将所有可能都走了一遍还是没有知道它是谁，则认定为错误符号，输出该错误符号，然后结束。每次成功识别了一个单词后，单词都会存在token中。然后确定这个单词的种别码，最后进行下一个单词的识别。  4. 主控程序主要负责对每次识别的种别码syn进行判断，对于不同的单词种别做出不同的反应，直至遇到syn=0;程序结束。 | | | |
| **心得体会** | 通过此次实验，让我了解到如何设计、编制并调试词法分析程序，加深对词法分析原理的理解；熟悉了构造词法分析程序的手工方式的相关原理，使用某种高级语言（例如Python语言）直接编写此法分析程序。另外，也让我重新熟悉了C++语言的相关内容，加深了对C++语言的用途的理解。 | | | |

程序附录：#include "pch.h"

#include <iostream>

#include<stdio.h>

#include<string.h>

#include<string>

#include<stdlib.h>

#include <iostream>

#include <fstream>

using namespace std;

//保留字表

string reserveWord[100] = {

"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","exit"

};

//界符运算符表

string reservedOperator[100] = {

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

"!=", ";", "(", ")", "^", ",", "\"", "\'", "#", "&",

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

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

};

static string identifierTable[1000] = { "" };//标识符表

char Word[50];

char Error[1000];//存储错误的标识符

int pointOferror=0;//错误数组的指针

int A\_error = 0;//发生错误的标志位

int searchReserve(string reserveWord[100], char s[])

{

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

{

if (reserveWord[i].compare(s) == 0)

{//成功查找

return i + 1;//返回种别码

}

}

return -1;//否则返回-1，代表查找不成功，即为标识符

}

bool IsLetter(char letter)

{//注意C语言允许下划线也为标识符的一部分可以放在首部或其他地方

if (letter >= 'a'&&letter <= 'z' || letter >= 'A'&&letter <= 'Z' || letter == '\_')

{

return true;

}

else

{

return false;

}

}

bool IsDigit(char digit)

{

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

{

return true;

}

else

{

return false;

}

}

void filterResource(char r[], int p)

{

char tempString[10000];

int count = 0;

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

{

if (r[i] == '/'&&r[i + 1] == '/')

{//若为单行注释“//”,则去除注释后面的东西，直至遇到回车换行

while (r[i] != '\n')

{

i++;//向后扫描

}

}

if (r[i] == '/'&&r[i + 1] == '\*')

{//若为多行注释“/\* 。。。\*/”则去除该内容

i += 2;

while (r[i] != '\*' || r[i + 1] != '/')

{

i++;//继续扫描

if (r[i] == '$')

{

printf("注释出错，没有找到 \*/，程序结束！！！\n");

exit(0);

}

}

i += 2;//跨过“\*/”

}

if (r[i] != '\n'&&r[i] != '\t'&&r[i] != '\v'&&r[i] != '\r')

{//若出现无用字符，则过滤；否则加载

tempString[count++] = r[i];

}

}

tempString[count] = '\0';

strcpy\_s(r, strlen(tempString) + 1,tempString);//产生净化之后的源程序

}

void Scanner(int &note, char resourceCode[], char token[], int &p)

{

int i, count = 0;//count用来做token[]的指示器，收集有用字符

char ch;//作为判断使用

ch = resourceCode[p];//读入一个字符

while (ch == ' ')

{//过滤空格，防止程序因识别不了空格而结束

p++;

ch = resourceCode[p];

}

for (i = 0; i < 20; i++)

{//每次收集前先清零

token[i] = '\0';

}

if (IsLetter(resourceCode[p]))

{//开头为字母

token[count++] = resourceCode[p];//收集

p++;//下移

while (IsLetter(resourceCode[p]) || IsDigit(resourceCode[p]))

{//后跟字母或数字

token[count++] = resourceCode[p];//收集

p++;//下移

}//多读了一个字符既是下次将要开始的指针位置

token[count] = '\0';

note = searchReserve(reserveWord, token);//查表找到种别码

if (note == -1)

{//若不是保留字则是标识符

note = 100;//标识符种别码

}

return;

}

else if (IsDigit(resourceCode[p]))

{//首字符为数字

while (IsDigit(resourceCode[p]))

{//后跟数字

token[count++] = resourceCode[p];//收集

p++;

while (IsLetter(resourceCode[p]))

{

token[count++] = resourceCode[p];//收集

p++;

A\_error = 1;

}

}//多读了一个字符既是下次将要开始的指针位置

token[count] = '\0';

if (A\_error == 1)

{

while (token[pointOferror]!='\0')

{

Error[pointOferror] = token[pointOferror];

pointOferror++;

}

Error[pointOferror++] = ' ';

}

note = 99;//常数种别码

}

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

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

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

{//若为运算符或者界符，查表得到结果

token[0] = resourceCode[p];

token[1] = '\0';//形成单字符串

for (i = 0; i < 36; i++)

{//查运算符界符表

if (reservedOperator[i].compare(token)== 0)

{

note = 33 + i;//获得种别码，使用了一点技巧，使之呈线性映射

break;//查到即推出

}

}

p++;//指针下移，为下一扫描做准备

return;

}

else if (resourceCode[p] == '<')

{//<,<=,<<

p++;//后移，超前搜索

if (resourceCode[p] == '=')

{

note = 38;

}

else if (resourceCode[p] == '<')

{//左移

p--;

note = 58;

}

else

{

p--;

note = 37;

}

p++;//指针下移

return;

}

else if (resourceCode[p] == '>')

{//>,>=,>>

p++;

if (resourceCode[p] == '=')

{

note = 40;

}

else if (resourceCode[p] == '>')

{

note = 59;

}

else

{

p--;

note = 39;

}

p++;

return;

}

else if (resourceCode[p] == '=')

{//=.==

p++;

if (resourceCode[p] == '=')

{

note = 42;

}

else

{

p--;

note = 41;

}

p++;

return;

}

else if (resourceCode[p] == '!')

{//!,!=

p++;

if (resourceCode[p] == '=')

{

note = 43;

}

else

{

note = 68;

p--;

}

p++;

return;

}

else if (resourceCode[p] == '&')

{//&,&&

p++;

if (resourceCode[p] == '&')

{

note = 53;

}

else

{

p--;

note = 52;

}

p++;

return;

}

else if (resourceCode[p] == '|')

{//|,||

p++;

if (resourceCode[p] == '|')

{

note = 55;

}

else

{

p--;

note = 54;

}

p++;

return;

}

else if (resourceCode[p] == '$')

{//结束符

note = 0;//种别码为0

}

else

{//不能被以上词法分析识别，则出错。

printf("error：there is no exist %c \n", ch);

exit(0);

}

}

void showError()

{

cout << "错误：";

int i = 0;

while (i<pointOferror)

{

cout << Error[i];

i++;

}

}

int main()

{

//打开一个文件，读取其中的源程序

char resourceCode[10000];//存放代码

char token[20] = { 0 };

int note = -1, i;//初始化

int p = 0;//源程序全局唯一指针

errno\_t err;

FILE \*fp, \*fp1;//准备文件流

if ((fopen\_s(&fp,"D:\\program.txt", "r")) != 0)

{//打开源程序

cout << "can't open this file";

exit(0);

}

resourceCode[p] = fgetc(fp);//从文件流中读入一个字符存入数组

while (resourceCode[p] != '$')//将源程序读入resourceCode[]数组

{

p++;

resourceCode[p] = fgetc(fp);

}

resourceCode[++p] = '\0';//在数组中存结束符

fclose(fp);//关闭流

cout << endl << "源程序为:" << endl;

cout << resourceCode << endl;

//对源程序进行过滤

filterResource(resourceCode, p);

p = 0;//从头开始读

if ((fopen\_s(&fp1,"D:\\program.txt", "a+")) != 0)

{//打开源程序

cout << "can't open this file";

exit(0);

}

while (note != 0)

{

//启动扫描

Scanner(note, resourceCode, token, p);

if (note == 100)

{//标识符

for (i = 0; i < 1000; i++)

{//插入标识符表中

if (identifierTable[i].compare(token) == 0)

{//已在表中

break;

}

if (identifierTable[i] == "")

{//查找空间

identifierTable[i]=token;

break;

}

}

printf("(标识符 ,%s)\n", token);

fprintf(fp1, "(标识符 ,%s)\n", token);

}

else if (note >= 1 && note <= 32)

{//保留字

cout << "(" << reserveWord[note - 1] << " ,--)" << endl;

//fprintf(fp1, "(%s , --)\n", reserveWord[note - 1]);

}

else if (note == 99)

{//const 常数

if (A\_error!=1)

{

cout << "(常数 " << " , token)" << endl;

//fprintf(fp1, "(常数 , %s)\n", token);

}

A\_error = 0;

}

else if (note >= 33 && note <= 68)

{

cout << "(" << reservedOperator[note - 33] << " ,--)" << endl;

//fprintf(fp1, "(%s , --)\n", reservedOperator[note - 33]);

}

}

for (i = 0; identifierTable[i]!=""; i++)

{//插入标识符表中

cout << "第" << i+1 << "个标识符："<< identifierTable[i] << endl;

//fprintf(fp1, "第%d个标识符： %s\n", i + 1, identifierTable[i]);

}

showError();

fclose(fp1);

return 0;

}