# 编译原理程序设计实验报告

## ——实验题目

班级：计算机1606班 姓名：戚子强 学号：20164625

### 实验亮点部分

1. 第三次实验分别用4种方式（递归子程序下降法、LL(1)分析法、LR(0)分析法和算符优先分析法）实现四元式语法制导翻译；

2. 第二次实验分别用4种方式（递归子程序下降法、LL(1)分析法、LR(0)分析法和算符优先分析法）实现语义分析；

3. 第一次实验的词法分析器支持C99多数词法，可以分析该词法分析器源代码本身。

### **第一次实验报告**

### 本次实验亮点部分：

第一次实验的词法分析器支持C99多数词法，可以分析该词法分析器源代码本身。

### **实验目标：**

词法分析扫描器的设计实现。

### **实验内容：**

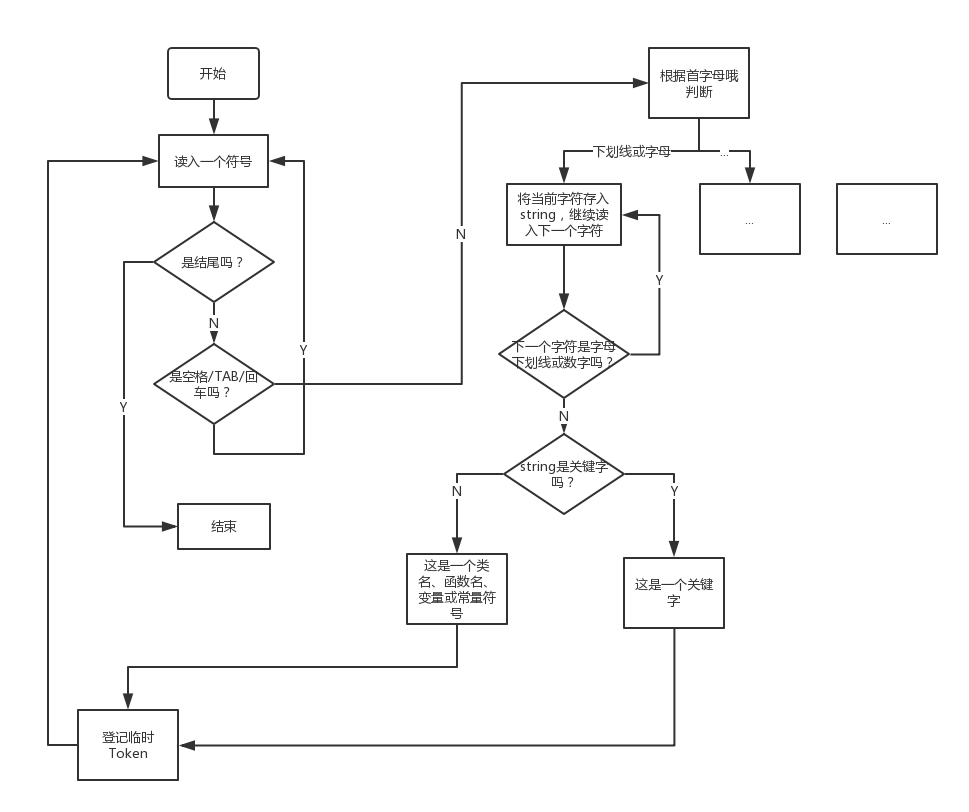
（包括：概要设计、数据结构、流程图、关键函数等有选择填写）

基本思想是首先从文件内读取源代码内容，以空格、Tab和回车区分每一个独立的单位，这个“单位”可能是一个关键字、变量或者其他类型；再利用开头符号进行第一次分拣，分拣后再继续读入字符到一个字符串变量中，直到一个界符出现；接下来登记这个字符串并进行合理分类，生成对应的Token串。

Token格式设计如下：



程序流程图如下：



源程序代码：（加入注释）

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

FileName:WordFilter.cpp

Function: Design and Implementation of Lexical Analysis Scanner

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <iostream>

#include <fstream>

#include <string>

#include <cassert>

#include "IsKeywords.h"

using namespace std;

/\*

public class tokens

{

int tokenId;

string tokenType, tokenValue;

tokens \*next = new tokens;

}

\*/

int main()

{

char ch; //当前读入的字符

string str; //单词单位

int token; //TOKEN格式详见文件token.xlsx

string fileName = "a.txt"; //从文件读入

ifstream infile;

infile.open(fileName.data());

assert(infile.is\_open());

infile >> noskipws;

infile >> ch;

//分别记录标识符、常量和界符，如果后续出现重复的词语则不再新增token，只与旧的token关联

string identifier[1024], constant[1024], boundary[1024];

int idNum = 0, conNum = 0, bounNum = 0;

while(!infile.eof() && ch!=EOF)

{

//过滤换行、Tab或空格

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

{

infile >> ch;

}

//若是字母或下划线开头，是变量、函数或类名或者关键字

else if((ch>='a' && ch<='z') ||

(ch>='A' && ch<='Z') ||

(ch == '\_'))

{

str = "";

token = 0;

while((

(ch>='a' && ch<='z') ||

(ch>='A' && ch<='Z') ||

(ch == '\_') ||

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

(!infile.eof()) )

{

str = str + ch;

infile >> ch;

}

cout<<"("<<str<<",";

//判断是否是关键字

token = isKeywords(str);

if(token != 0) cout<<token<<")"<<endl;

//以字母或下划线开头的，不是关键字就是标识符

/\*

else

{

identifier[idNum++] = str;

token = 200 + idNum; cout<<token<<endl;

}

\*/

else

{

int haveId = 0;

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

{

if(identifier[i] == str)

{

token = 200 + i;

cout<<token<<")"<<endl;

haveId = 1;

break;

}

}

if(haveId == 0)

{

identifier[idNum++] = str;

token = 200 + idNum;

cout<<token<<")"<<endl;

}

}

}

//若是数字开头，一定是数字常量

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

{

//token = 3;

str = "";

while(((ch>='0' && ch<='9')||(ch == '.')) && !(ch=='\n'||ch=='\t'||ch==' ') && (!infile.eof()) )

{

str = str + ch;

infile >> ch;

}

constant[conNum++] = str;

token = 300 + conNum;

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

}

//若是引号开头，一定是字符(串)常量

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

{

// token = 3;

str = "";

str = str + ch;

infile >> ch;

while(ch != '\'' && ch != '\"' && !infile.eof())

{

str = str + ch;

infile >> ch;

}

str = str + ch;

infile >> ch;

constant[conNum++] = str;

token = 300 + conNum;

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

}

//若是//或者/\*开头，是注释

else if(ch == '/')

{

infile >> ch;

//单行注释

if(ch == '/')

{

token = 501;

str = "//";

infile >> ch;

while(ch != '\n' && !infile.eof())

{

str = str + ch;

infile >> ch;

}

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

}

//多行注释

else if(ch == '\*')

{

token = 502;

str = "/\*";

infile >> ch;

while(!infile.eof())

{

str = str + ch;

infile >> ch;

if(ch == '\*')

{

infile >> ch;

if(ch == '/')

{

str = str + "\*/";

infile >> ch;

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

break;

}

else

{

str = str + "\*";

}

}

}

}

//判断注释有错误情况

else

{

token = -1;

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

}

}

//其他情况，是运算符(界符)

else

{

token = 4;

str = "";

/\*

while((!((ch>='a' && ch<='z')||(ch>='A' && ch<='Z')||(ch == '\_')||(ch>='0'&&ch<='9')||(ch=='\n'||ch=='\t'||ch==' '||ch=='/')))&& (!infile.eof()))

{

str = str + ch;

infile >> ch;

}

\*/

str = str + ch;

infile >> ch;

int haveBoun = 0;

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

{

if(boundary[i] == str)

{

token = 400 + i;

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

haveBoun = 1;

break;

}

}

if(haveBoun == 0)

{

boundary[bounNum++] = str;

token = 400 + bounNum;

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

}

}

}

infile.close();

return 0;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

FileName:IsKeywords.h

Function: Determine whether the given word is the keyword.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <string>

#include <iostream>

using namespace std;

string keywords[73] =

{

"asm",

"do",

"if",

"return",

"typedef",

"auto",

"double",

"inline",

"short",

"typeid",

"bool",

"dynamic\_cast",

"int",

"signed",

"typename",

"break",

"else",

"long",

"sizeof",

"union",

"case",

"enum",

"mutable",

"static",

"unsigned",

"catch",

"explicit",

"namespace",

"static\_cast",

"using",

"char",

"export",

"new",

"struct",

"virtual",

"class",

"extern",

"operator",

"switch",

"void",

"const",

"false",

"private",

"template",

"volatile",

"const\_cast",

"float",

"protected",

"this",

"wchar\_t",

"continue",

"for",

"public",

"throw",

"while",

"default",

"friend",

"register",

"true",

"delete",

"goto",

"reinterpret\_cast",

"try",

"alignas",

"alignof",

"char16\_t",

"char32\_t",

"constexpr",

"decltype",

"noexcept",

"nullptr",

"static\_assert",

"thread\_local"

};

int isKeywords(string str)

{

for(int i = 0; i < 72; i++)

{

if(str == keywords[i])

return (101 + i);

}

}

程序运行结果：（截屏）

输入：





输出：

