编译原理设计基于C语言的词法分析

实验报告

学院：计算机科学与工程

班级：200402

姓名：夏新宇

学号：20203644

实验一：设计基于C语言的词法分析器

1. 实验题目

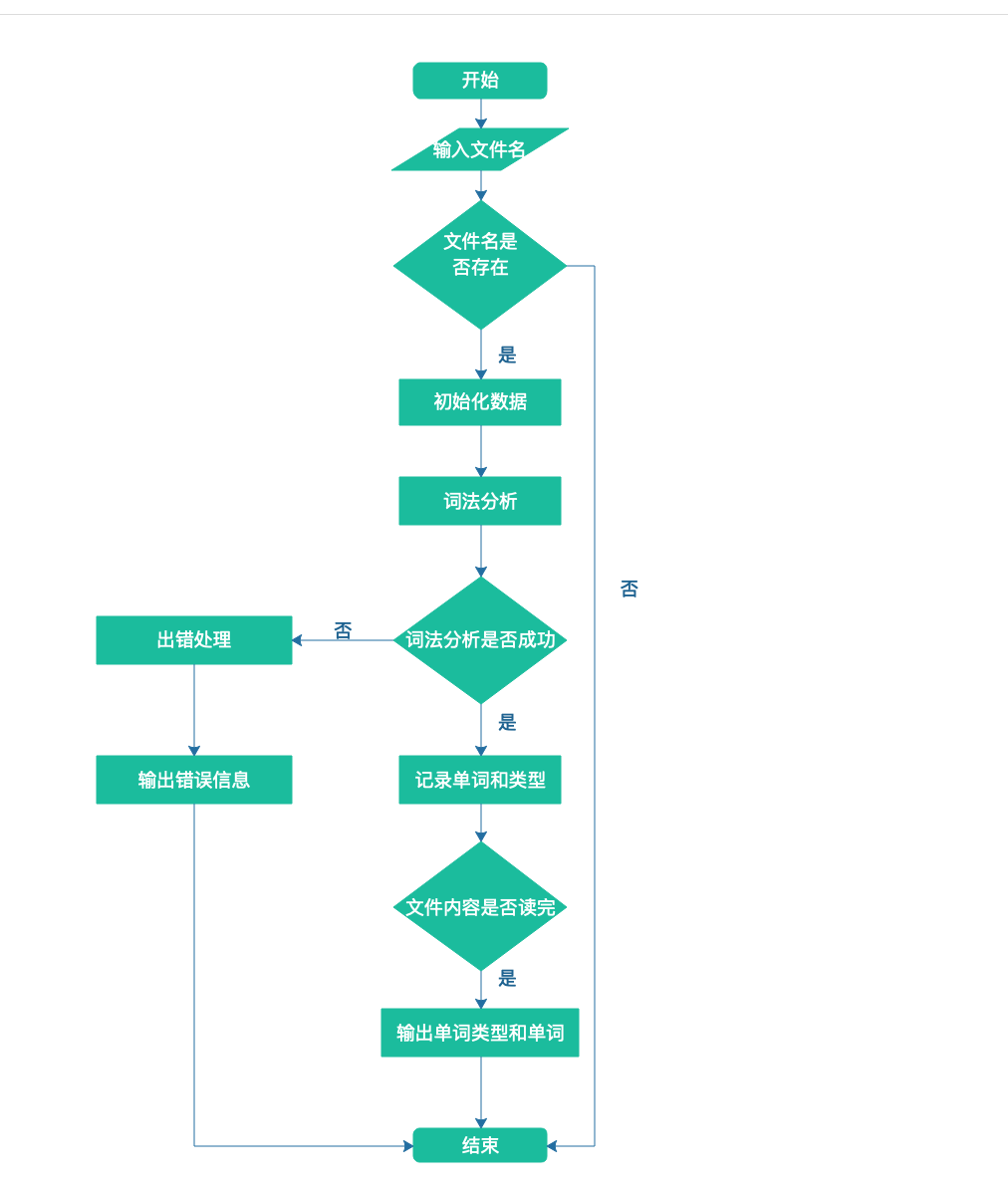
设计基于C语言的词法分析器

1. 实验目的

通过设计编制调试 C 语言的词法分析程序，加深对词法分析原理的理解。并掌握在 对程序设计语言源程序进行扫描过程中将其分解为各类单词的词法分析方法。

编制一个读单词过程，从输入的源程序中，识别出各个具有独立意义的单词，即基 本字、标识符、常数、运算符、分隔符五大类。并依次输出各个单词的内部编码及单词 符号自身值。

三． 词法分析的算法分析



四．程序代码

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#include<iostream>

using namespace std;

string keywords[20] = { "include","void","main","int","char","float","double","if",

"else","then","break","continue", "for","do","while",

"printf","scanf", "begin","end","return" };

char rz[99999] = " ";

string id[10000];

int pp = 0;

string nu[10000];

int qq = 0;

int isLetter(char a) { //判断是否是字母请完成此函数

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

return 1;

else

return 0;

}

int isDigit(char a)  { //判断是否是数字请完成此函数

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

return 1;

else

return 0;

}

int alpha(int st){ //识别保留字和标识符，给此函数的的语句加上注释

char wordbuf[20] = " "; //初始化长度20的字符数组

int n = 0; //作为wordbuf数组的下标指针，指向wordbuf的最后一个字符

for( ; ; ){ //方法体中执行break语句时循环终止

wordbuf[n] = rz[st]; //从数组rz从下标st位置开始，依次将字符赋值到word数组里，word数组下标从0开始

st++; //数组下标自增

n++; //数组下标自增

if ((isDigit(rz[st]) == 1) || (isLetter(rz[st]) == 1) || (rz[st] == '\_')) //如果rz数组中当前字符是数字或者英文字符或者是'\_'

wordbuf[n] = rz[st];  // 向wordbuf数组中添加字符rz[st]

else //否则break；结束for循环

break;

}

int flag = 0;  //记录是否找到合法关键字

for (int k = 0; k < 20; k++) { //遍历关键字字符串数组，每个keywords[k] 代表一个关键字

if (strcmp(keywords[k].c\_str(),wordbuf) == 0)  //如果关键字字符串数组中有关键字和wordbuf相等，flag = 1；

flag = 1;

}

if (flag == 0){ //flag = 0 说明在没个keywords没有找到关键字

int idx = -1; //寻找合法关键字或非法关键字起始下标

for (int t = 0; t < pp; t++){ // pp 代表当前标识符个数，根据pp对id数组进行遍历

if (strcmp(id[t].c\_str(), wordbuf) == 0){ //利用idx确定标识符起始位置

idx = t; //完成赋值

}

}

if (idx != -1) printf(" (id,%d) ", idx);  //说明找到指定标识符，输出标识符位置

else{

id[pp] = wordbuf; //没找到指定标识符，将指定标识符添加到id数组中

printf(" (id,%d) ", pp); // 输出标识符所在位置

pp++; //pp计数器自增

}

}else{

printf(" (");

for (int i = 0; i < n; i++){ //n是wordbuf数组长度根据n输出wordbuf字符

printf("%c", wordbuf[i]); //输出wordbuf字符数组内容

}

printf(",-) "); //输出标识符类型

}

return st; //返回当前st下标 // 返回st的值，代表遍历当前rz数组的rz[st]字符

}

int number(int st){ //识别整数

char numbuf[20] = " "; int n = 0; //初始化长度20的字符数组，用于存放数字

int k = 0;  //控制numbuf中'.'的数量，最多1个

int flag = 0; //flag = 1 说明不是合法数字

for ( ; ;){ //

numbuf[n] = rz[st];

st ++;

n ++;

if (isDigit(rz[st]) == 1){

numbuf[n] = rz[st];

}else if ((k == 0) && (rz[st] == '.')){

numbuf[n] = rz[st];

k++;

}else if (isLetter(rz[st]) == 1){

numbuf[n] = rz[st];

flag = 1; //遍历到的当前字符不是数字

continue;

}else break;

}

if (flag == 0){

int id = -1; //寻找合法整数或非法整数起始下标

for (int t = 0; t < qq; t++) //在nu数组中寻找等于numbuf的元素值下标

if (strcmp(nu[t].c\_str(), numbuf) == 0)

id = t;

if (id != -1)

printf(" (nu,%d) ", id);

else{

nu[qq] = numbuf;

printf(" (nu,%d) ", qq); //如果没找到，numbuf值添加到nu数组里

qq++;

}

}else {

printf(" (");

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

printf("%c", numbuf[i]); //输出非法整数值

printf(",error digital!) ");

}

return st; // 返回st的值，代表遍历当前rz数组的rz[st]字符

}

int anotation(int st){ //处理除号和注释

char tabuf[9999] = " "; // 该数组用与保存字符'/'后面的字符串

int n = 0; //作为wordbuf数组的下标指针，指向wordbuf的最后一个字符

st ++; //st自增，指向rz数组中'/'的下一个字符

if (rz[st] == '/'){ //当前字符是'/'说明这段语句是注释

printf(" (//,-)"); //输出字符串  (//,-)

st++; // st自增

while (rz[st] != 10){  //读到换行符终止

tabuf[n] = rz[st]; //将字符rz[st]赋值给tabuf[n]

st++; //st自增

n++; //n自增

}

printf(" \n 注释");

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

printf("%c", tabuf[i]); //输出注释中内容

}else if (rz[st] == '\*'){  // 当前字符是'\*'说明这段语句是多行注释

printf(" (/\*,-) ");  //输出字符串  (/\*,-)

st++; //st自增

int stt = st + 1;

while (1){

if (rz[st] == '\*' && rz[st + 1] == '/') //如果当前字符是'\*'并且下一个字符是'/'说明读取到多行注释结尾

break; //结束循环

tabuf[n] = rz[st]; //将字符rz[st]赋值给tabuf[n]

st ++; //st自增

n ++; //n自增

if (rz[st + 1] == '\0'){

printf("(/\* error!!\n)"); //没有读取到多行注释结尾的符号"\*/"则输出这条语句，代表非法注释

return st + 1; //函数结束调用，执行权返回给main主函数

}

}

printf(" \n 注释");

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

printf("%c", tabuf[i]); //输出注释内容

printf(" (\*/,-) ");  //输出字符串  (\*/,-)

st = st + 2; //偏移量2，代表指向"\*/"后边的字符

}else if (rz[st] == '='){ //代表 /= 操作

st++; //st自增

printf(" (/=,-) "); // 输出字符串 (/=,-)

}else printf(" (/,-) "); //代表除号

return st; // 返回st的值，代表遍历当前rz数组的rz[st]字符

}

int other(int st) {

switch (rz[st]) {

case'=':

st++;

if (rz[st] == '=') {

st++;

printf(" (rlop,==) ");

} else

printf(" (rlop,=) ");

break;

case'+':

st++;

if (rz[st] == '=') {

st++;

printf(" (+=,-) ");

} else if (rz[st] == '+') {

st++;

printf(" (++,-) ");

} else printf(" (+,-) ");

break;

case'-':

st++;

if (rz[st] == '=') {

st++;

printf(" (-=,-) ");

} else if (rz[st] == '-') {

st++;

printf(" (--,-) ");

} else

printf(" (-,-) ");

break;

case'\*':

st++;

if (rz[st] == '=') {

st++;

printf(" (\*=,-) ");

} else

printf(" (\*,-) ");

break;

case'>':

st++;

if (rz[st] == '=') {

st++;

printf(" (rlop,>=) ");

} else printf(" (rlop,>) ");

break;

case'<':

st++;

if (rz[st] == '=') {

st++;

printf(" (rlop,<=) ");

} else

printf(" (rlop,<) ");

break;

case'%':

st++;

if (rz[st] == '=') {

st++;

cout<< " (\\" << "%" << "=,-) " << endl;

// printf(" (\%=,-) ");

} else

cout<< " (\\" << "%" << ",-) " << endl;

// printf(" (\%,-) ");

break;

case'!':

st++;

if (rz[st] == '=') {

st++;

printf(" (!=,-) ");

} else

printf(" (!,wrong thing!) ");

break;

case'&':

st++;

if (rz[st] == '&') {

st++;

printf(" (&&,-) ");

} else printf(" (&,worng word!) ");

break;

case'|':

st++;

if (rz[st] == '|') {

st++;

printf(" (||,-) ");

} else

printf(" ( |,worng word ! ) ");

break;

case'{':

st++;

printf(" ({,-) ");

break;

case'}':

st++;

printf(" (},-) ");

break;

case'(':

st++;

printf(" ((,-) ");

break;

case')':

st++;

printf(" (),-) ");

break;

case'[':

st++;

printf(" ([,-) ");

break;

case']':

st++;

printf(" (],-) ");

break;

case':':

st++;

printf(" (:,-) ");

break;

case'#':

st++;

printf(" (#,-) ");

break;

case';':

st++;

printf(" (;,-) ");

break;

case'.':

st++;

printf(" (.,-) ");

break;

case',':

st++;

printf(" (,,-) ");

break;

case '&':

st++;

break;

case '^':

st++;

break;

case 10:

st++;

printf("\n");

break;

case 34:

st++;

printf(" (\",-) ");

break;

case 39:

st++;

printf(" (',-) ");

break;

default:

printf(" (%c,worngthing) ", rz[st]);

st++;

}

return st;

}

int choice(int st) { //根据读入的单词的第一个字符确定调用不同的单词识别函数

if (isLetter(rz[st]) == 1)

st = alpha(st);

else if (isDigit(rz[st]) == 1)

st = number(st);

else if (rz[st] == '/')

st = anotation(st);

else

st = other(st);

return st;

}

int main(){

int i = 0;

FILE\* fp;

char name[10]; printf("请输入文件名:\n"); //读取文件路径

scanf("%s", name);

if ((fp = fopen(name, "r")) == NULL) { //如果存在指定文件则打开

    printf("Open error!");

    exit(0);

}

char ch = fgetc(fp);

while (ch != EOF) //逐个获取文件中字符，直到文件末尾

{

    rz[i] = ch;

i++;

    ch = fgetc(fp);

}

    fclose(fp); //关闭文件

    int j = 0;

while (rz[j] != '\0')

j = choice(j); cout << endl << " 程序中标示符如下 " << endl;

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

cout << i << " " << id[i] << endl;

cout << " 程序中数字如下" << endl;

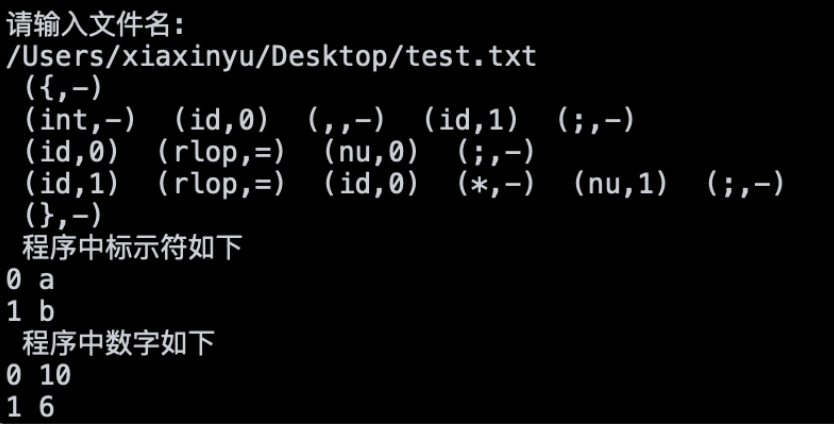
for (j = 0; j < qq; j++)

cout << j << " " << nu[j] << endl;

system("pause");

}

五．运行结果及分析



1. 首先在控制台打印提示信息
2. 输入文件所在路径并回车
3. 得到单词的内部编码以及单词符号自身值
4. 分别输出标识符和数字的编号以及值

六．实验心得

通过[编译原理](https://so.csdn.net/so/search?q=%E7%BC%96%E8%AF%91%E5%8E%9F%E7%90%86&spm=1001.2101.3001.7020" \t "/Users/xiaxinyu/Documents\\x/_blank)的这次程序实验，在自已动手体验的情况下，更加透彻地理解了词法分析的过程，以及该算法。对于以后由模型向程序代码的转化能力上，有了很大的锻炼。以后我会更加专心的研究计算机知识，不断将现实中遇到的实际问题，向程序方面转变，做到灵活运用所学知识。