苏 州 大 学 实 验 报 告

院、系	计算机科学与技术学院	年级专业	22 计科	姓名	姜涛	学号	2227405073
课程名称	编译原理实践					成绩	
指导教师	段湘煜	同组实验者	无		实验日期	2024/9/6	

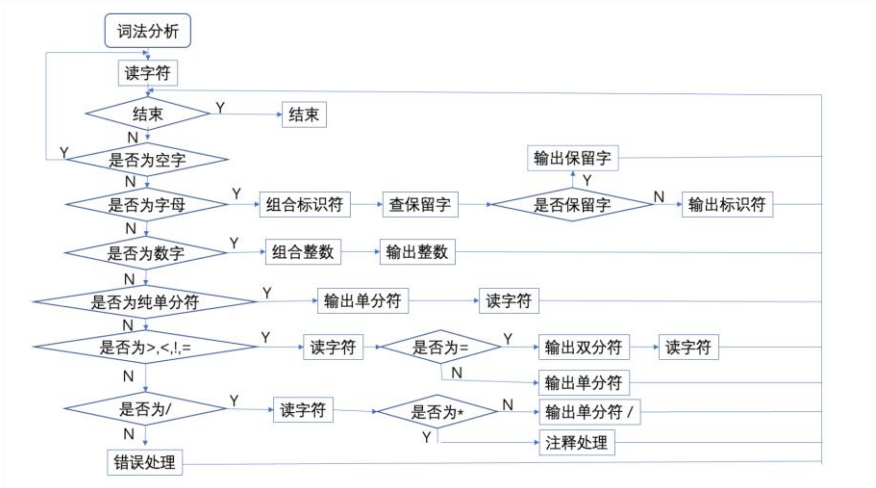
实 验 名 称　　实现词法分析器

一. 实验题目

我们将用一个抽象机的汇编语言作为 TEST 编译器的目标语言。TEST 机的指令仅能作为 TEST 机的目标。实际上，TEST 机具有精简指令集计算机的一些特性。TEST 机的模拟程序直接从一个文件中读取汇编代码并执行它，因此避免了由汇编语言翻译为机器代码的过程。此外，为了避免与外部的输入输出例程连接的复杂性，TEST 机有内部整型的 I/O 设备；在模拟时，它们都对标准设备读写。
用 C 语言实现一个 TEST 语言的词法分析器。

二. 实验原理及流程框图

TEST 语言的程序结构很简单，它在语法上相当于 C 的函数体，即由一对大括号括起来的语句序列，没有过程或函数。声明语句、表达式语句以及控制语句的写法都与 C 语言类似，但规定：一条声明语句只能声明一个整型简单变量，没有数组；控制语句只是 if、while 和 for 三个语句，这三个控制语句本身也可包含语句序列；表达式仅限于布尔表达式和整型算术表达式，布尔表达式由对两个算术表达式的比较组成，该比较实用<、<=、>、>=、==和!=比较算符；算术表达式可以包括整型常量、变量以及+、-、*、/这 4 个运算符。另外，还可以有复合语句。为了能实现输入输出，又添加了 read 语句和 write 语句。TEST 语言还可以有注释，注释用/*和*/括起来，但注释不能嵌套。



图表 1 词法分析程序流程图

三. 实验步骤

TEST 编译器包括以下的 C 文件：

教务处制

TESTmain.c: 主程序，先后调用词法分析、语法分析及语义分析和代码生成。
TESTscan.c: 词法分析，接受用 TEST 语言编写的程序，输出的单词符号程序将作为语法分析的输入。
TESTparse.c: 语法、语义分析及 TEST 机的汇编代码生成，如果分析中发现有错误，则报告错误

在程序开始时,首先读入一个字符,若为空字符,则继续读,直到读进一个非空字符。读进的字符有如下 6 种情况,要进行不同的处理。
(1)字母。继续读,直到遇见空格、分界符、非字母数字字符或到文件尾。组合标识符,查保留字表。若为保留字,输出相应单词记号;若不是,输出标识符的单词记号及单词值。
(2)数字。继续读,直到遇见空格或非数字字符出现或到文件尾。输出无符号整数的单词记号及数字串。
(3)非=、<、>、等与双分界符首字符不同的单分界字符。输出相应单词记号及单分界符。
(4)=、<、>、!。读下一个字符,判断是否为双字符分界符,若是,组成双字符分界符,输出相应单词记号及双分界符;若不是,输出单分界符记号。
(5)/。读下一个字符。若不是*,输出/的单词记号;若是*,进行注释处理。词法分析不输出“/*”,并要跳过整个注释内容直到遇到“*/”为止,然后返回开始状态,继续识别下一个单词符号。
(6)非法字符。如果读进的字符不属于上而任一情况,则说明词法分析程序从源程序读人了一个不合法的字符,即该字符不属于程序语言所定义的所有单词符号的首字符集合。词法分析程序在遇到不合法字符时要进行错误处理,报告错误信息,跳过这个字

四. 实验结果及分析

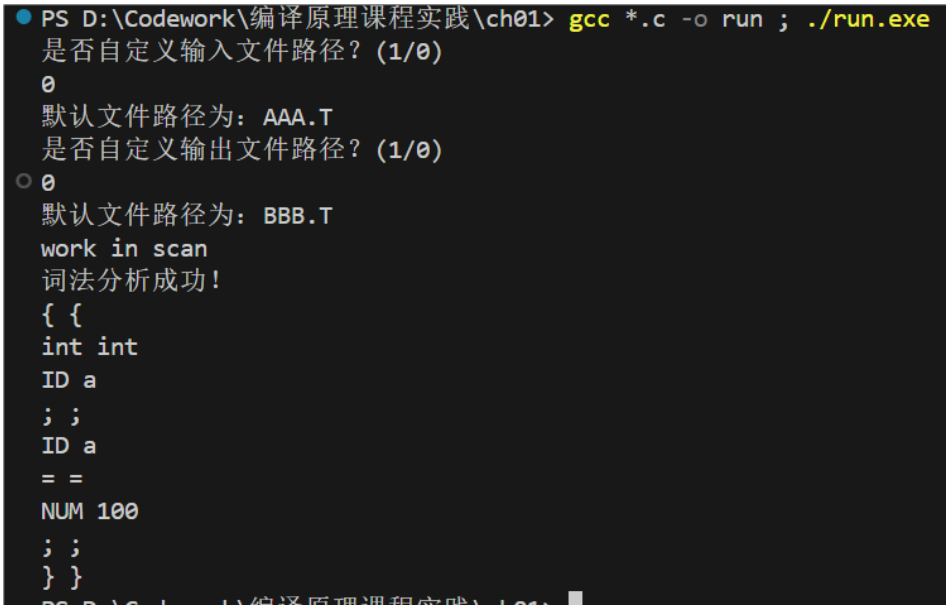
预期结果：
针对如下例程：

```
{
    int a;
    a=100;
}
```

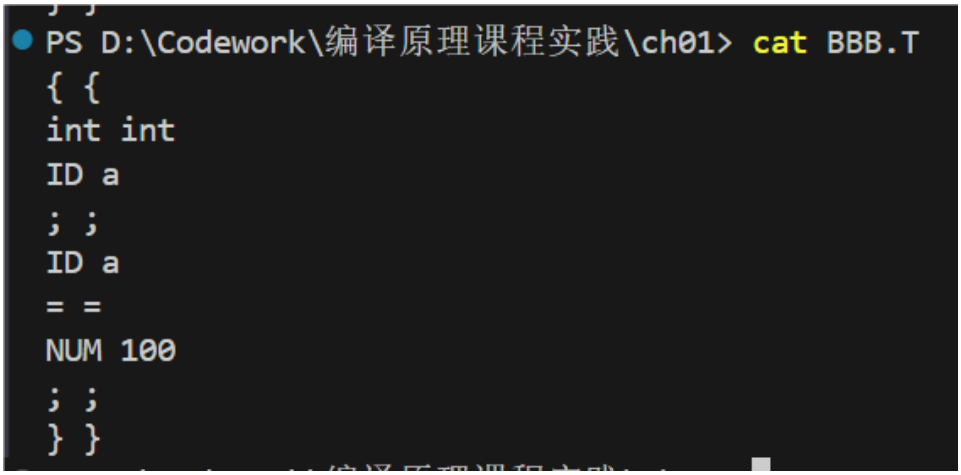
图表 2 例程

预期结果：词法分析识别出{后,输出“{ {”； 识别出 int,输出“int int”;而识别出标识符 a 后,应输出“ID a”;识别出无符号整数 100 后,应输出“NUM 100”。

实际运行结果：



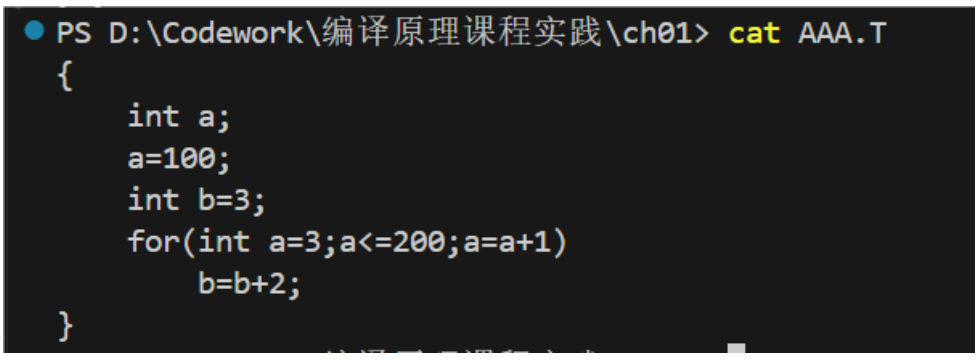
图表 3 运行测试样例图



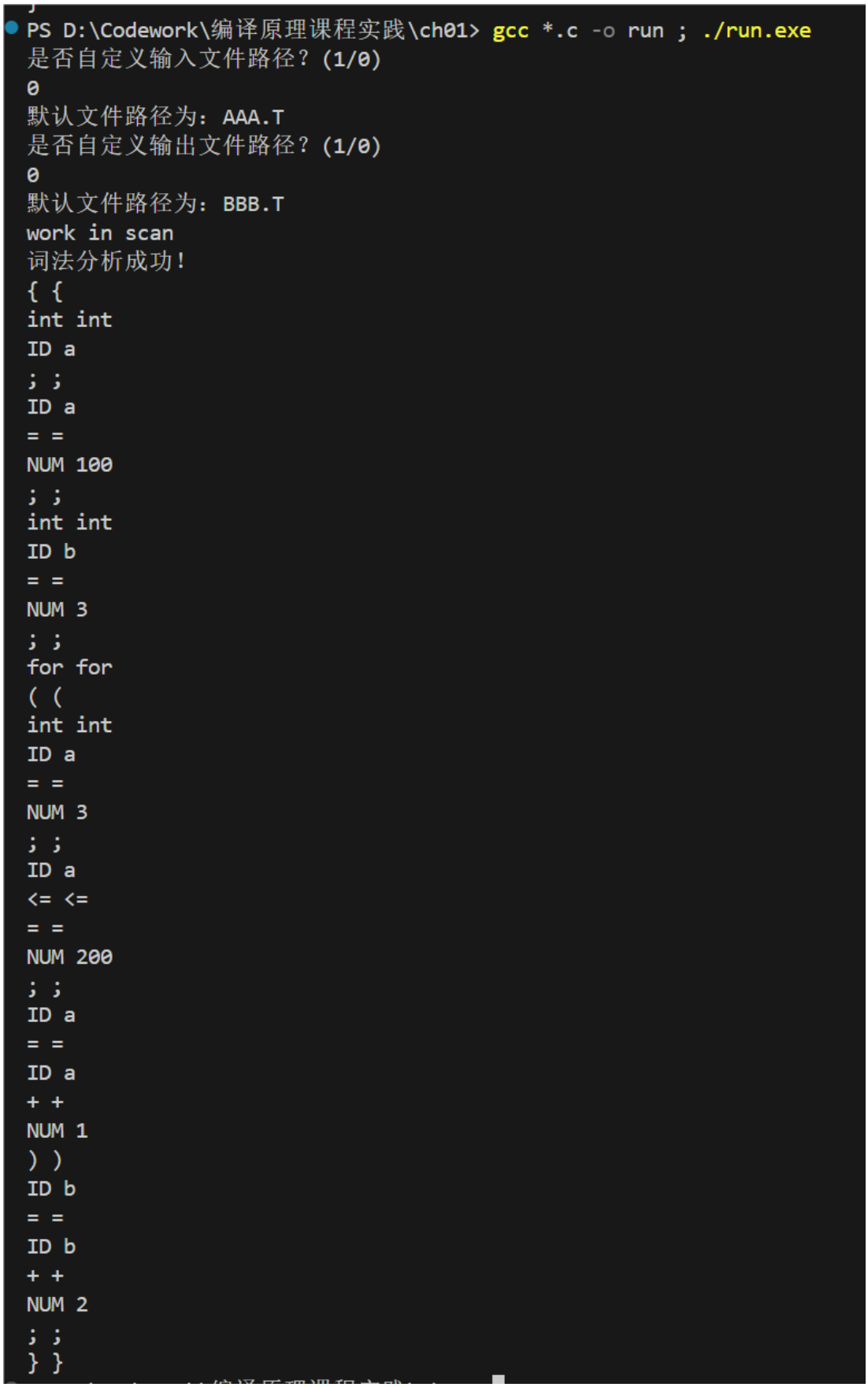
图表 4 例程运行保存文件结果

可以看到所写的词法分析器能够正常的对源文件进行词法分析并将结果打印在屏幕上并保存在指定文件中。

对例程进行修改后测试是否能够正常输出结果：



图表 5 修改后的例程



图表 6 修改后程序运行结果

可以看到该词法分析器能够正常的对输入文件进行此法分析。

对输入文件中添加非法字符来测试编译失败报错：

```
PS D:\Codework\编译原理课程实践\ch01> cat .\AAA.T.
{
    int a;
    a=100;
    @;
}
```

图表 7 添加非法字符后的 AAA.T

```
PS D:\Codework\编译原理课程实践\ch01> gcc *.c -o run ; ./run.exe
是否自定义输入文件路径？(1/0)
0
默认文件路径为：AAA.T
是否自定义输出文件路径？(1/0)
0
默认文件路径为：BBB.T
work in scan
error when scan char: @ with ascii: 64词法分析有误，编译停止！
```

图表 8 成功识别非法字符运行结果

五. 实验总结

在本次实验中，通过实现一个简单的词法分析器，进一步加深了对编译原理中词法分析阶段的理解和掌握。实验中主要通过 C 语言编写了一个用于 TEST 语言的词法分析器，能够识别语言中的关键字、标识符、常量、运算符和分界符等基本成分，并对其进行相应的处理。同时，程序还能够处理注释及非法字符，并提供了相应的错误报告机制。

六. 代码

/*TESTmain.c*/

```
#include <stdio.h>
#include <ctype.h>
#include <stdlib.h>

extern int TESTscan();

char Scanin[300],Scanout[300];

char Filein[300], Resout[300];

FILE * fin, *fout;

void main()
{

    char c;
    int p=0;
```

```
char filename[20]="AAA.T";
char target[20]="BBB.T";

printf("是否自定义输入文件路径？(1/0)\n");
int readfilename=0;
scanf("%d",&readfilename);

if(readfilename==0)
{
    printf("默认文件路径为：%s\n",filename);
}
else
{
    printf("请输入文件名：");
    scanf("%s",filename);
}
printf("是否自定义输出文件路径？(1/0)\n");

int writetarget=0;
scanf("%d",&writetarget);
if(writetarget==0)
{
    printf("默认文件路径为：%s\n",target);
}
else
{
    printf("请输入文件名：");
    scanf("%s",target);
}

fin = fopen(filename, "r");
fout = fopen(target, "w");

while(c!=EOF)
{
    c=fgetc(fin);
    Filein[p++]=c;
}
Filein[p]='\0';

//printf("123");
int es = 0;
es = TESTscan();
if(es > 0)
```

```
        printf("词法分析有误，编译停止！");
    else
        printf("词法分析成功！\n");

    if(es==0)
        printf("%s",Resout);

    //fout = fopen("file.txt", "w");
    if (fout == NULL)
    {
        perror("error opening file");
        exit(0);
    }
    fputs(Resout, fout);
    fclose(fin);
    fclose(fout);


}
```

/*TESTscan.c*/

```
#include <stdio.h>
#include <ctype.h>
#include <string.h>
#include <stdbool.h>

//下面定义保留字表，为简化程序，实用字符指针数组保存所有保留字
//如果想增加保留字，可继续添加，并修改保留字数目 keywordSum
#define keywordSum 10
char * keyword[keywordSum]={"if","else", "for", "while", "do", "int","read",
"write"};
//下面定义纯单分界符
char singleword[50] = "+-*(){};,:";
//下面定义双分界符的首字符
char doubleword[10]= "><=!";
extern char Filein[300], Resout[300];
//用于接收输入输出文件名，在 test_main.c 中定义

char word[50];
char tmp[50];
int p=0;
```

```
bool isKeyword(char * str)
{
    int i;
    for(i=0;i<keywordSum;i++)
    {
        if(keyword[i]==NULL)
            continue;
        //printf("fk %s\n",keyword[i]);
        if(strcmp(str,keyword[i])==0)
            return true;
    }
    return false;
}



int TESTscan()
{

    printf("work in scan\n");
    //return 0;
    int dex=0;
    int cnt=0;
    while(dex<300 && cnt++<300)
    {
        char c=Filein[dex];
        //printf("fk %c\n",c);
        if(c=='\0' || c==-1)
        {
            return 0;
        }

        if(c==' ' || c==10 || c=='\t' || c=='\n')//空
        {
            dex++;
            continue;
        }
        else if(isalpha(c))//字母
        {

            p=0;
            word[p++]=c;

            //存在下一个字符，下一个字符是字母或数字
            while(dex+1<300 && (isalpha(Filein[++dex]) || isdigit(Filein[dex])))
```

```
        {
            word[p++]=Filein[dex];
        }

        word[p]='\0';


        //printf("%s\n",word);

        if(isKeyword(word))
        {
            //printf("%s %s\n",word,word);
            sprintf(tmp,"%s %s\n",word,word);
            strcat(Resout,tmp);
        }
        else
        {
            //printf("ID %s\n",word);
            sprintf(tmp,"ID %s\n",word);
            strcat(Resout,tmp);
        }


    }//
    else if(isdigit(c))//数字
    {
        p=0;
        word[p++]=c;
        while(dex+1<300 && isdigit(Filein[++dex]))
        {
            word[p++]=Filein[dex];
        }
        word[p]='\0';
        //printf("NUM %s\n",word);
        sprintf(tmp,"NUM %s\n",word);
        strcat(Resout,tmp);
    }
    else if(strchr(singleword,c)!=NULL)//是纯单分界符
    {
        //printf("%c %c\n",c,c);
        sprintf(tmp,"%c %c\n",c,c);
        strcat(Resout,tmp);
        dex++;
    }
    else if(strchr(doubleword,c)!=NULL)//是双分界符
    {
```

```
            dex++;
            char next=Filein[dex];
            if(next=='=')
            {
                //printf("%c%c %c%c\n",c,next,c,next);
                sprintf(tmp,"%c%c %c%c\n",c,next,c,next);
                strcat(Resout,tmp);
            }
            else
            {
                //printf("%c %c\n",c,c);
                sprintf(tmp,"%c %c\n",c,c);
                strcat(Resout,tmp);
            }

        }
        else if(c=='/')
        {
            dex++;
            char next=Filein[dex];
            if(next=='*')
            {
                //注释
                while(dex+1<300 && Filein[++dex]!='*')
                {
                    //pass
                }
                if(dex+1<300 && Filein[++dex]=='/')
                {
                    continue;
                }
            }
        }
        else
        {
            printf("error when scan char: %c with ascii: %d",c,c);
            return 3;//error
        }

    }

}
```