

编译原理

递归下降语法分析设计原理与实现

学 院：

专 业：

学生姓名：

学 号：

北京交通大学

2023年10月

1. **程序功能描述**

程序的主要功能描述

1、将二元式序列文件作为程序的输入；

2、对输入的字符串做出正确的递归下降分析判断，判断其是否是给定文法的算术表达式；

3、能发现简单的错误，例如非法字符输入和合法输入但其中某处推导中出现的语法错误等；

1. **主要数据结构**

|  |  |
| --- | --- |
| 变量及类型 | 用途 |
| char token[MAX] | 用于存放输入的句子，并判断是否匹配 |
| char token1[MAX]; |
| char token2[MAX]; |
| char current | 存放当前的推导位置 |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

1. **程序结构描述**
2. **设计方法**

相关宏定义如图3.1。

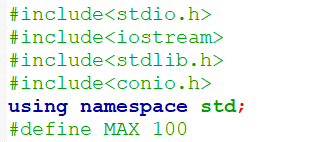


图3.1 宏定义

提前定义好数组保存相关的内容，如图3.2所示。

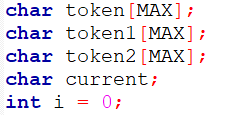


图3.2

对于每一个识别子程序，设计方法如下：

1. 遇到中介符号i时if (current==i）,读入下一个输入符号
2. 遇到非终结符号时，调用该非终结符的识别函数

# 3、遇到形如A->ε，若当前输入符号不是A的follow集中元素，则报错。

程序流程图如图3.4所示：

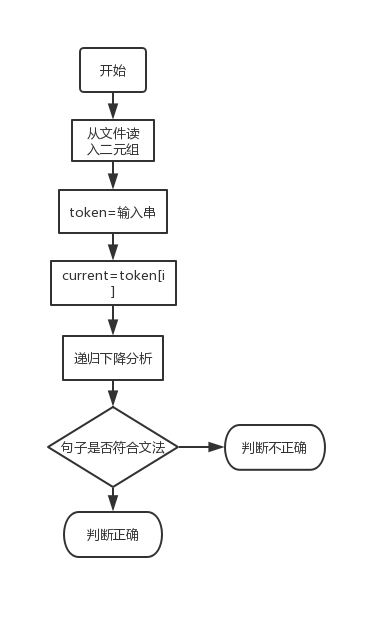
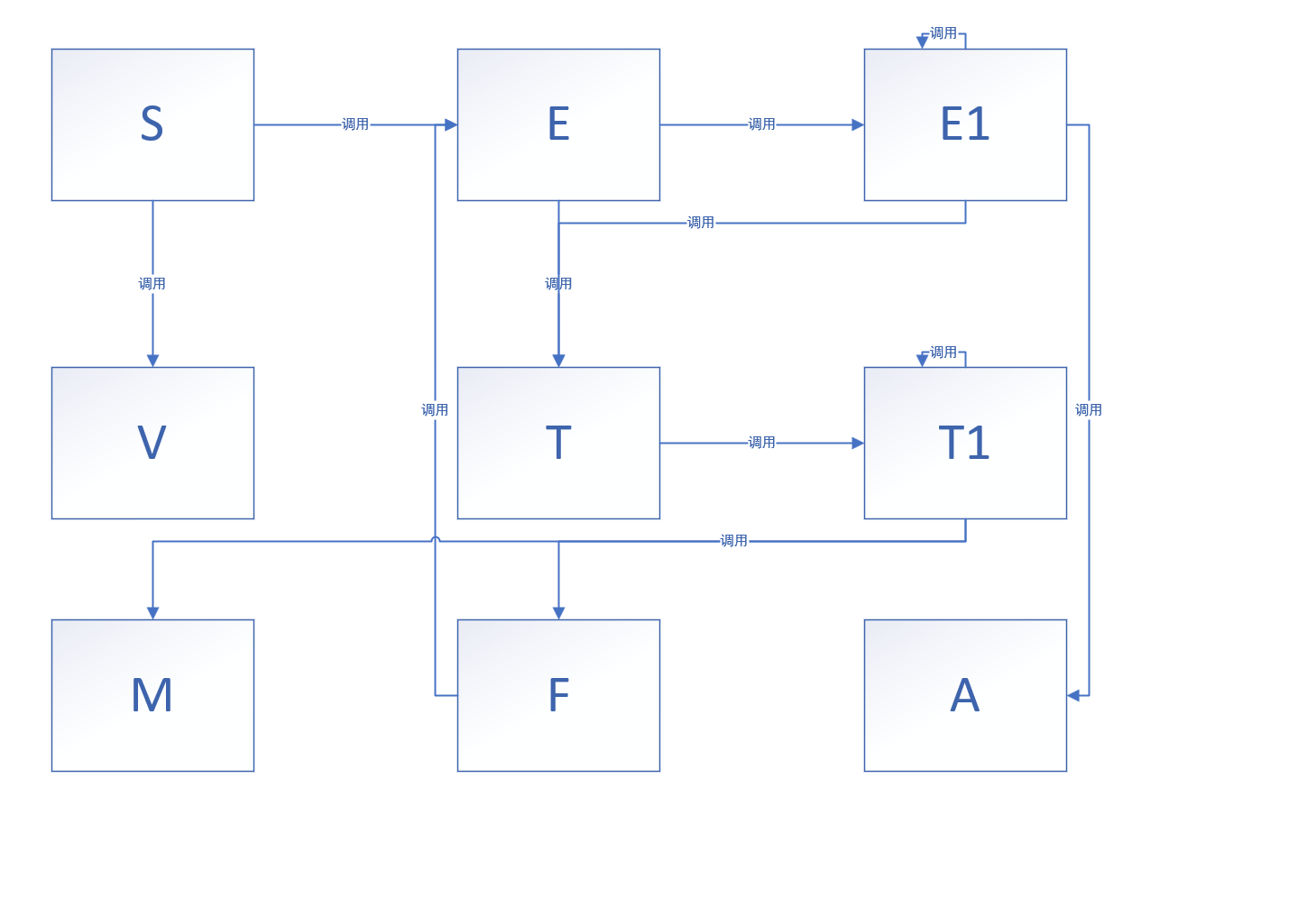


图3.4 程序流程图

1. **函数定义及函数间的调用关系**

|  |  |
| --- | --- |
| 函数名称 | 函数功能描述 |
| void init() | 初始化 |
| void judge(char ch, int i) | 判断文件读取是否合法，包括检查非法字符 |
| void advance() | 读取下一个字符 |
| void S() | 非终结符S的语句 |
| void E() | 非终结符E的语句 |
| void E() | 非终结符E1的语句 |
| void T() | 非终结符T的语句 |
| void T1() | 非终结符T1的语句 |
| void M() | 非终结符M的语句 |
| void A() | 非终结符A的语句 |
| void F() | 非终结符F的语句 |
| void V() | 非终结符V的语句 |
| void error(char \*c) | 输出含有语法错误的地方 |
| void error1() | 输出错误提示并终止程序 |
| void readFile() | 读取输入文件 |

函数间的调用关系如图3.5所示：



说明：调用读取字符函数忽略

图3.5 函数间调用关系

1. **程序测试**

测试用例1：test1.cpp

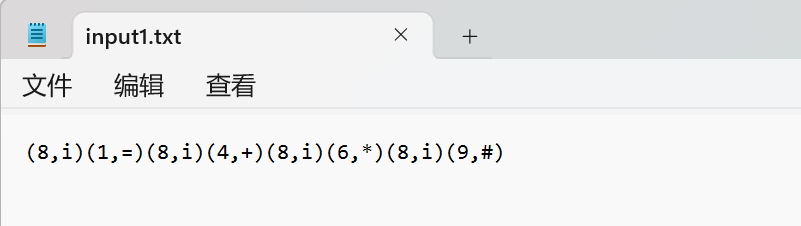
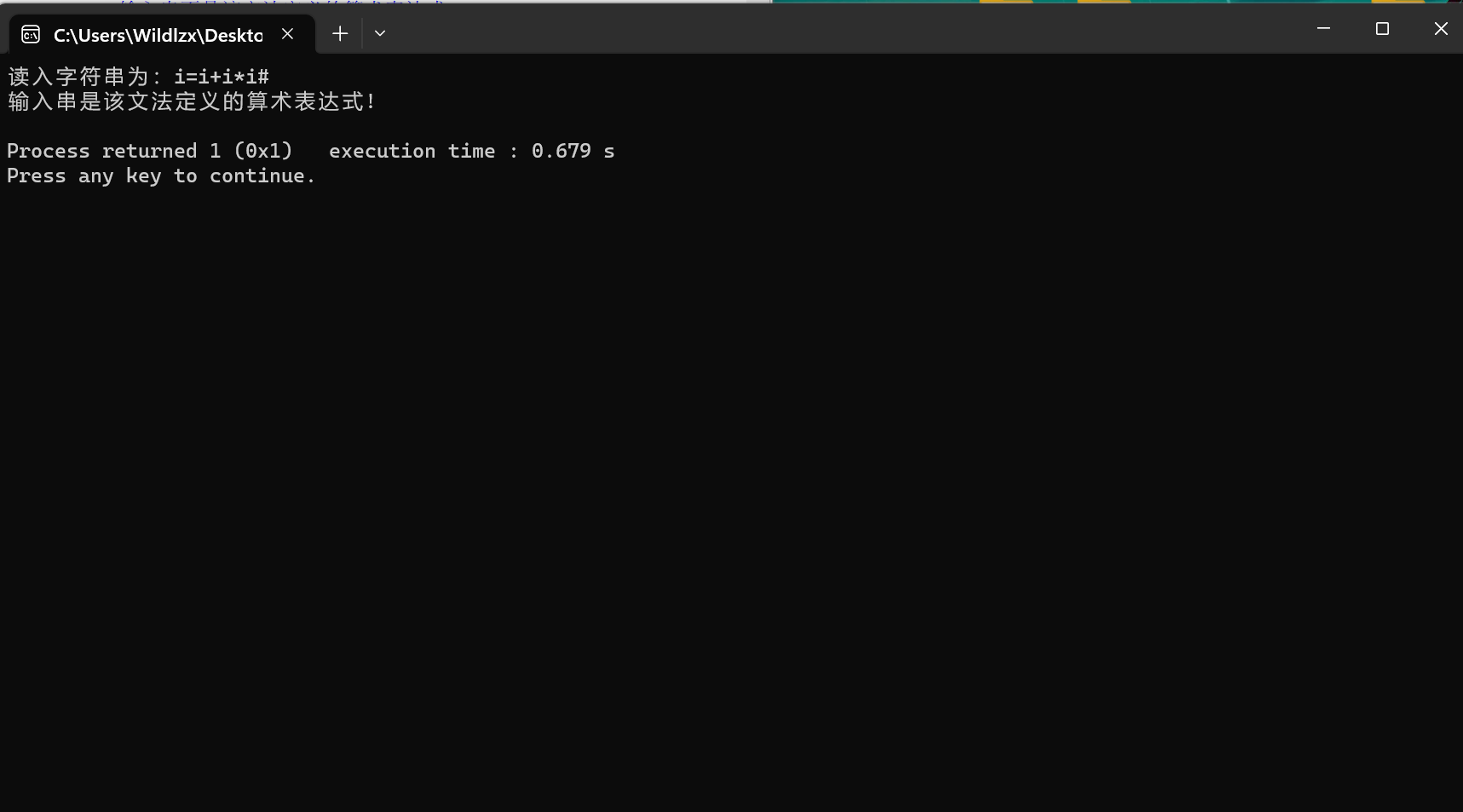


图4.1 测试用例1

程序执行结果：（XXXX输出至yyyy.txt）



测试用例2：test2.cpp

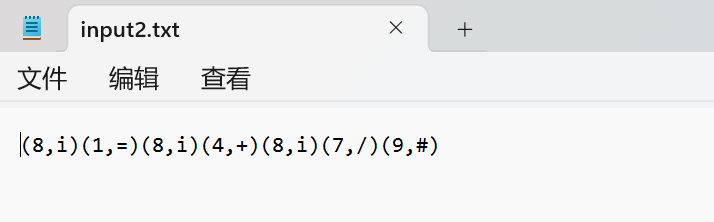


图4.2 测试用例2

程序执行结果：

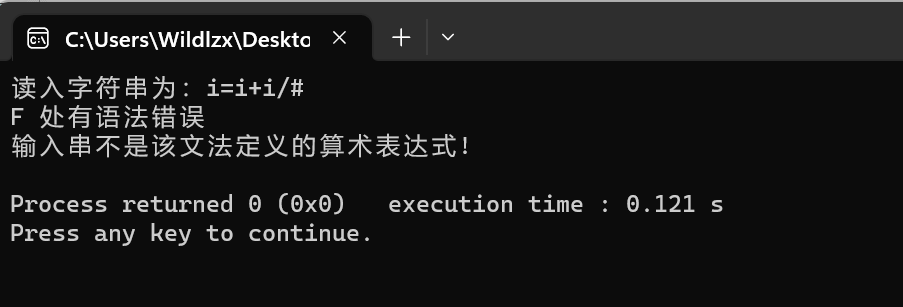


图4.3 测试结果

五．实验结果分析

由测试结果可见，在本次实验中，我编写的程序可以成功实现递归下降语法分析。并且，由于我的程序中加入了错误判断，可以有效实现发现并标识语法错误。最后，由于实现的功能是函数化的，本程序可为之后的大型编译器提供递归下降分析的接口。

六．研究性教学实验感受

刚开始写实验的时候有些无从下手，后面通过看书和查阅资料逐步了解需要完成任务和实现的方法，然后开始逐步实现应有的功能函数。通过这次实验让我对词法分析的理解更深了一步，这也是做实验和学习这门课程的目的所在。一步步完善实验的过程也是在增强自己的知识。

7. 附件（源代码列表）

#include<stdio.h>

#include<iostream>

#include<stdlib.h>

#include<conio.h>

using namespace std;

#define MAX 100

char token[MAX];

char token1[MAX]; //用于检测文件输入是否匹配

char token2[MAX];

char current;

int i = 0;

void S();

void E();

void E1();

void T();

void A();

void F();//函数推导过程

void T1();

void M();

void V();

void error(char \*c); //提出语法错误

void error1(); //提出非法输入错误

void init() //初始化

{

current = token[i];

}

void judge(char ch, int i) //判断文件读取内容是否合法，包括检查非法字符和不匹配现象

{

switch (ch)

{

case '1':

if (token[i] != '=')

error1();

break;

case '2':

if (token[i] != '(')

error1();

break;

case '3':

if (token[i] != ')')

error1();

break;

case '4':

if (token[i] != '+')

error1();

break;

case '5':

if (token[i] != '-')

error1();

break;

case '6':

if (token[i] != '\*')

error1();

break;

case '7':

if (token[i] != '/')

error1();

break;

case '8':

if (token[i] != 'i')

error1();

break;

case '9':

if (token[i] != '#')

error1();

break;

default:

error1();

break;

}

}

void advance() //读取下一个字符

{

if (i == MAX)

exit(0);

current = token[++i];

}

void S()

{

if (current == 'i')

{

V();

if (current == '=')

{

advance();

E();

}

}

else

error(const\_cast<char\*>("S"));

}

void E()

{

if (current == 'i' || current == '(')

{

T();

E1();

}

else

error(const\_cast<char\*>("E"));

}

void E1()

{

if (current == '+' || current == '-')

{

A();

T();

E1();

}

else

if (!(current == ')' || current == '#'))

error(const\_cast<char\*>("E1"));

}

void T()

{

if (current == 'i' || current == '(')

{

F();

T1();

}

else

error(const\_cast<char\*>("T"));

}

void T1()

{

if (current == '\*' || current == '/')

{

M();

F();

T1();

}

else

if (!(current == ')' || current == '#' || current == '+' || current == '-'))

error(const\_cast<char\*>("T1"));

}

void M()

{

if (current == '\*')

advance();

else

if (current == '/')

advance();

else

error(const\_cast<char\*>("M"));

}

void A()

{

if (current == '+')

advance();

else

if (current == '-')

advance();

else

error(const\_cast<char\*>("A"));

}

void F()

{

if (current == '(')

{

advance();

E();

if (current == ')')

advance();

}

else

if (current == 'i')

advance();

else

error(const\_cast<char\*>("F"));

}

void V()

{

if (current == 'i')

advance();

else

error(const\_cast<char\*>("V"));

}

void error(char \*c)

{

printf("%s 处有语法错误\n", c);

cout << "输入串不是该文法定义的算术表达式!" << endl;

exit(0);

}

void error1()

{

cout << "非法输入错误！" << endl;

cout << "输入串不是该文法定义的算术表达式!" << endl;

exit(0);

}

void readFile()

{

int j = 1;

int i = 0;

FILE \*fp;

//i=i+i\*i

if ((fp = fopen("input2.txt", "r")) == NULL)

{

cout << "文件打开失败！" << endl;

exit(0);

}

/\*i=i+i/

if ((fp = fopen("input2.txt", "r")) == NULL)

{

cout << "文件打开失败！" << endl;

exit(0);

}\*/

token1[0] = fgetc(fp);

while (!feof(fp))

{

token1[j++] = fgetc(fp);

if ((j % 5) == 0)

{

token2[i] = token1[j - 4];

token[i] = token1[j - 2];

judge(token2[i], i);

i++;

}

}

if ((j - 1) % 5 != 0) {

error1();

}

cout << "读入字符串为：" << token << endl;

}

int main()

{

readFile();

init();

S();

if (current == '#')

cout << "输入串是该文法定义的算术表达式!" << endl;

return 1;

}