華中科技大學

课程设计报告

题目：基于高级语言源程序格式处理工具

课程名称：程序设计综合课程设计

专业班级：cs2003

学　　号：U202015360

姓　　名：胡沁心

指导教师：卢萍

报告日期：2021/10/14

计算机科学与技术学院

任务书

设计内容

在计算机科学中，抽象语法树（abstract syntax tree 或者缩写为 AST），是将源代码的语法结构的用树的形式表示，树上的每个结点都表示源程序代码中的一种语法成分。之所以说是“抽象”，是因为在抽象语法树中，忽略了源程序中语法成分的一些细节，突出了其主要语法特征。

抽象语法树(Abstract Syntax Tree ,AST)作为程序的一种中间表示形式,在程序分析等诸多领域有广泛的应用.利用抽象语法树可以方便地实现多种源程序处理工具,比如源程序浏览器、智能编辑器、语言翻译器等。

在《高级语言源程序格式处理工具》这个题目中，首先需要采用形式化的方式，使用巴克斯（BNF）范式定义高级语言的词法规则（字符组成单词的规则）、语法规则（单词组成语句、程序等的规则）。再利用形式语言自动机的的原理，对源程序的文件进行词法分析，识别出所有单词；使用编译技术中的递归下降语法分析法，分析源程序的语法结构，并生成抽象语法树,最后可由抽象语法树生成格式化的源程序。

设计要求

1. 语言定义

选定 C 语言的一个子集，要求包含：

（1）基本数据类型的变量、常量，以及数组。不包含指针、结构，枚举等。

（2）双目算术运算符（+-*/%），关系运算符、逻辑与（&&）、逻辑或（||）、赋值运算符。不包含逗号运算符、位运算符、各种单目运算符等等。

（3）函数定义、声明与调用。

（4）表达式语句、复合语句、if 语句的 2 种形式、while 语句、for 语句，return 语句、break 语句、continue 语句、外部变量说明语句、局部变量说明语句。

（5）编译预处理（宏定义，文件包含）

（6）注释（块注释与行注释）

2. 单词识别

设计 DFA 的状态转换图（参见实验指导），实验时给出 DFA，并解释如何在状态迁移中完成单词识别（每个单词都有一个种类编号和单词的字符串这 2 个特征值），最终生成单词识别（词法分析）子程序。

注：含后缀常量，以类型不同作为划分标准种类编码值，例如 123 类型为 int，123L 类型为 long，单词识别时，种类编码应该不同；但 0x123 和 123 类型都是 int，种类编码应该相同。

3. 语法结构分析

（1）外部变量的声明；

（2）函数声明与定义；

（3）局部变量的声明；

（4）语句及表达式；

（5）生成(1)-(4)（包含编译预处理和注释）的抽象语法树并显示。

4. 按缩进编排生成源程序文件。

参考文献

参考文献

[1] 曹计昌，卢萍，李开. C 语言与程序设计. 电子工业出版社，2013

[2]严蔚敏等.数据结构（C 语言版）.清华大学出版社，

[3] Larry Nyhoff. ADTs, Data Structures, and Problem Solving with C++. Second Edition,Calvin College,2005

[4] 殷立峰. Qt C++跨平台图形界面程序设计基础. 清华大学出版社,2014:192～197

[5] 严蔚敏等.数据结构题集（C 语言版）.清华大学出版社

目录

1 引言

1.1 课题背景与意义

抽象语法树（Abstract Syntax Tree，AST），或简称语法树（Syntax tree），是源代码语法结构的一种抽象表示。

1.1.1 背景

抽象语法树以树状的形式表现编程语言的语法结构，树上的每个节点都表示源代码中的一种结构。之所以说语法是“抽象”的，是因为这里的语法并不会表示出真实语法中出现的每个细节。

1.1.2 意义

和抽象语法树相对的是具体语法树（通常称作分析树）。一般的，在源代码的翻译和编译过程中，语法分析器创建出分析树，然后从分析树生成 AST。一旦 AST 被创建出来，在后续的处理过程中，比如语义分析阶段，会添加一些信息。抽象语法树并不依赖于源语言的语法，也就是说语法分析阶段所采用的上下文无文文法，因为在写文法时，经常会对文法进行等价的转换（消除左递归，回溯，二义性等），这样会给文法分析引入一些多余的成分，对后续阶段造成不利影响，甚至会使合个阶段变得混乱。因些，很多编译器经常要独立地构造语法分析树，为前端，后端建立一个清晰的接口。

1.2 国内外研究现状

（1）AST 生成工具:

Esprima：解析器；Acorn：esprima 后的轮子,目前使用最多。webpack 也使用此工具；Astexplorer：在线生成工具；Espree：最初从 esprima fork 出来，来自 eslint，用于 eslint；babel-parser：原 babylon，最初从 acorn fork 出来；UglifyJS2：自带 parser；shift-parser-js：自己定义了一套 AST 规范 shift-spec。

（2）可以使用的插件工具:

esprima：code=>AST 代码转 AST；estraverse：traverse AST 转换树；escodegen：AST=>code；

（3）基于 AST 的工具

Eslint； Webpack；UglifyJS：代码压缩；Prettier：使用@babel/parser、angular-estree-parser；Typescript：自带 parser、transformer、codeGenerator；代码重构：1.Recast；2.jscodeshift；3.react-codemod；其他，例如：IDE 错误提示、格式化、高亮、自动补全等

（4）AST 转换工具

estraverse

（5）代码生成工具：

escodegen

1.3 课程设计的主要研究工作

1.了解 AST 的概念、作用和流程。

2.了解各种 AST 生成网站和软件。

3.阅读网络上不同作者写的详细 AST 解说，并使用生成 AST 的网站，了解 AST 呈现的形式。

4.了解编译器的运行原理，报错和警告的流程

2 系统需求分析与总体设计

2.1 系统需求分析

1.读取 c 代码,识别单词,包括要求中所有关键词及其语句，标识符，界定符，运算符，文件包括，宏定义以及注释的读取和识别。其中能够识别的数字类型有：十进制，带后缀的十进制，八进制，十六进制，小数。支持负数识别和括号运算。

2.对不合法字符、不合法字符串进行报错，并将所有单词及其编码写入文件中。

3.分析语法，进行语法报错，其中包括：

1)所有界定符和运算符的错误使用

2)重复定义或未定义的变量和函数

3)没有 if 的 else 和 else-if 语句

4)break、continue、return 语句的错误使用

5)函数调用时参数错误

6)变量赋值类型错误

7)不合法的宏定义命名和变量命名

8)对没有效果的语句输出警告

4.根据测试代码生成 AST

5.遍历生成的 AST，输出 AST 内容

6.根据 AST 编排缩进，写入新的 c 文件

2.2 系统总体设计

(1)功能模块(每个文件对应一个模块)

表 2-1 功能模块说明表

文件	功能模块	功能
main.cpp	主函数	文件的打开、关闭，调用各个模块等各项基本指令
read.h	词法分析	读取文件内容，获得字符串和符号的类型编号，对非法字符报错
judge.h	判别类型	判断读取的字符串的类型。包括字符类型，关键字类型，数字类型，其中数字类型包括十进制，带后缀的十进制，八进制，十六进制，小数。判断字符串中字符的合法性

compile.h	语法分析	子程序的选择和递归调用，所有语句的处理，编译预处理，变量的声明，函数的声明和定义，for，while，if-else，if-else-if，return，break，continue语句，行注释，块注释，条件语句，赋值语句的处理并建AST
tree.h	建AST	AST结点的新建和插入
error.h	报错	所有语法和字符的判错报错，以及部分语句的警告
traverse.h	输出AST	中序遍历AST，编排缩进，输出内容
save.h	源文件编排	遍历AST，编排缩进，写入源程序文件中

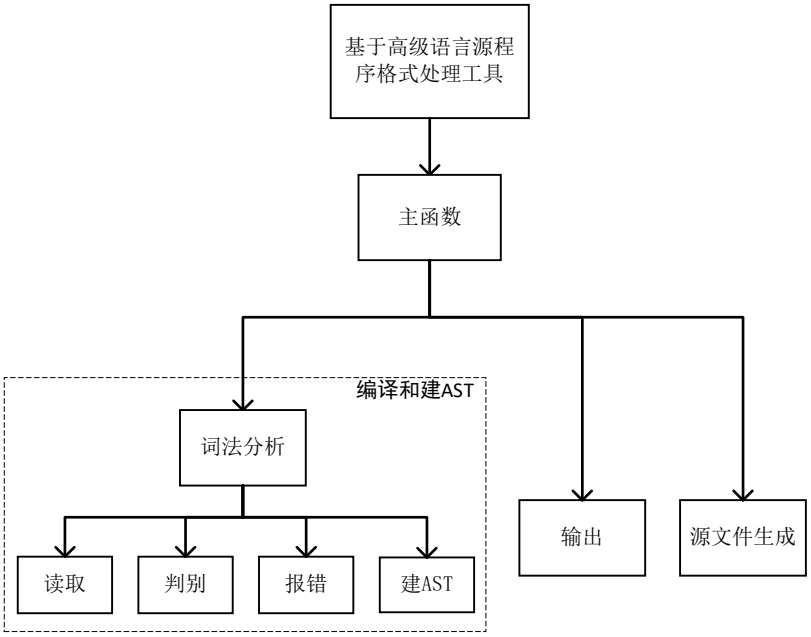

图 2-1 系统模块结构图

3 系统详细设计

3.1 数据结构的定义

（1）需要处理的数据

表 3-1 系统数据

数据	类型	名字	说明
宏定义，函数，变量总数	int	n	
当前所在的字符的位置	int	i	
测试用例内容	string	s	下标为 i
当前读取的字符串	string	t	
读取的字符串长度	int	j	
读取的前一字符串	string	pr	
读取的前一个符号的编号	int	pre	
标记是否已进入下一嵌套	bool	c	
语句嵌套层数	int	nest	
循环层数	int	loop	
当前行数	int	l	
圆括号层数	int	paren	
方括号层数	int	bracket	
当前所在函数序号	int	temp	
注释	string	note	
符号类型编号	enum	token_k	
字符串类型编号	enum	string_k	
报错类型编号	enum	error_k	
当前的父亲结点	Node*	T	
每层嵌套所在的父亲节点	Node*[10]	f	下标为 nest
不同优先级运算符的父亲节点	Node*[5]	r	优先级 0 1 2 3 4 符号 = &&，|| <,>,==,!= +,- *,/,%
声明过的宏定义、变量及函数的信息	variable	st	下标为 n
当前读取的字符的编号	dfa	op	

表 3-2 系统数据结构

数据	数据类型	数据项	数据项类型	名字
声明过的宏定义、变量及函数的信息	struct (variable)	变量数据类型或函数返回值类型	int	type
		变量或函数名	char	name
		函数参数类型	int	param_type
		函数参数名字	char	param_name
		数组变量大小	char	array_size
AST 结点数据结构	struct (Node)	结点类型	char	key1
		具体信息	char	key2
		孩子数量	int	sum
		孩子结点	Node*[10]	child
读取的字符的编号	struct (dfa)	字符串编号	int	s
		字符编号	int	t

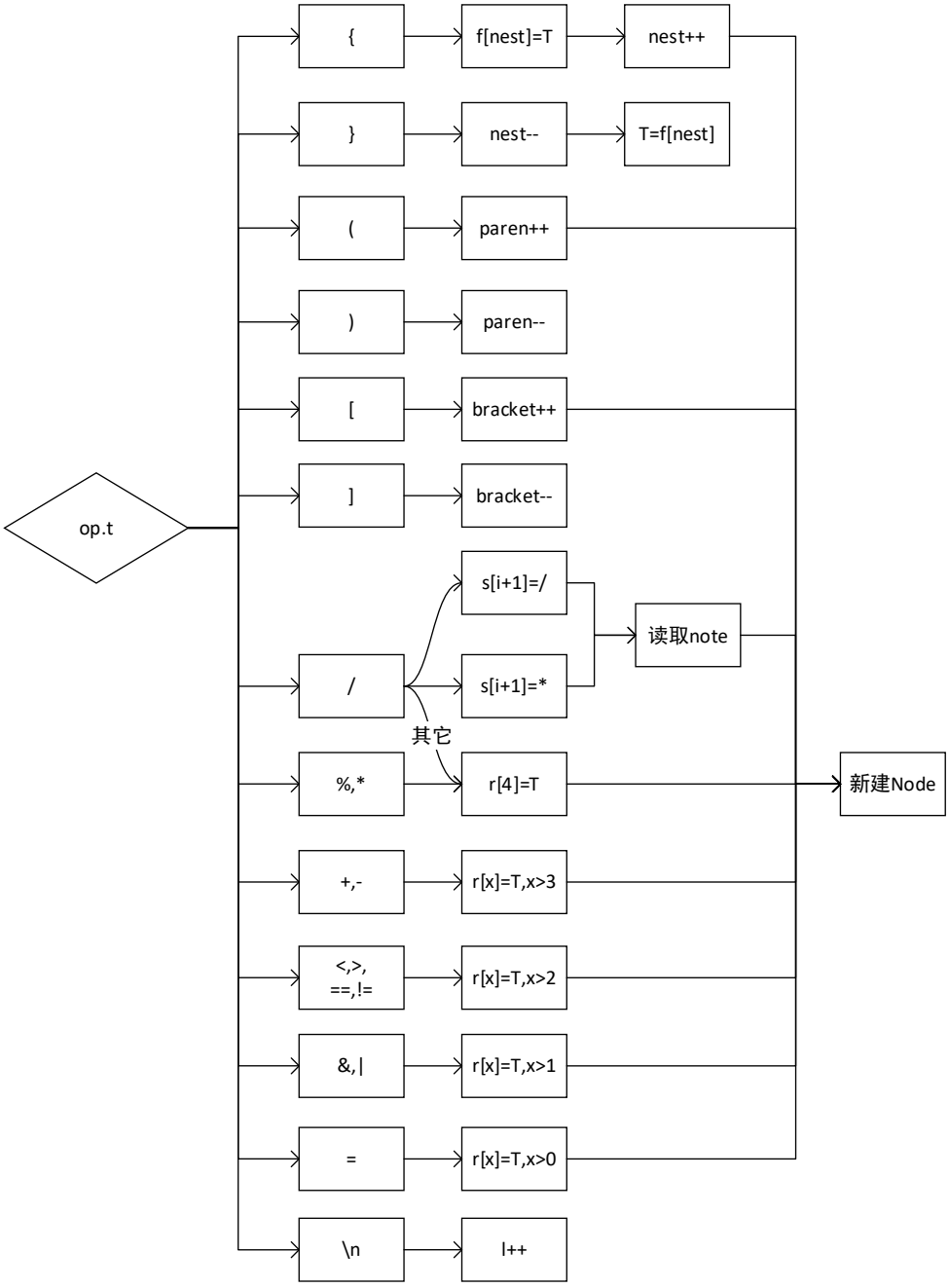

图 3-1 系统数据联系图 1

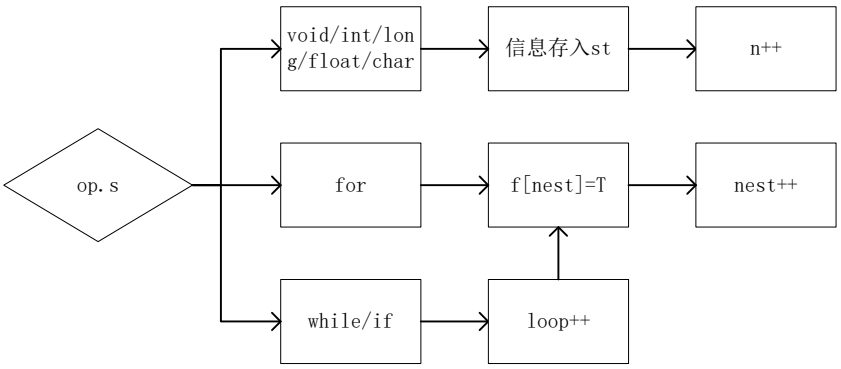

图 3-2 系统数据联系图 2

3.2 主要算法设计

1.主程序模块

流程：

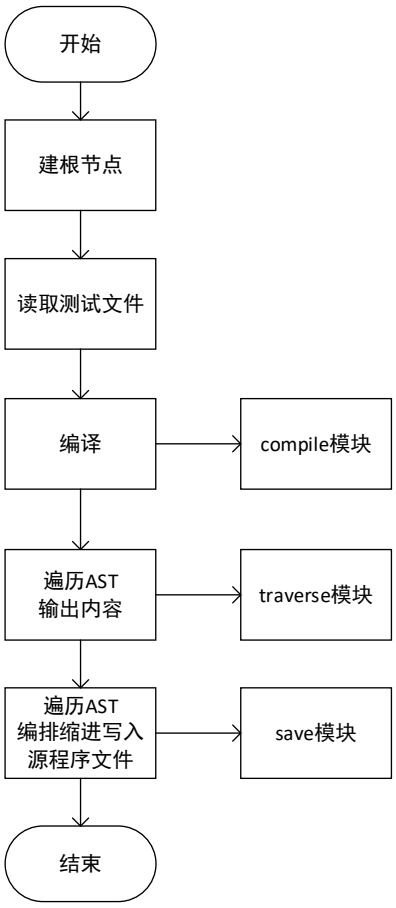

图 3-3 主程序流程图

2.读取模块

算法思想：递归

流程：先读取一个字符串和一个字符，分别获取它们的类型编号

1)DFA（识别读取的字符串和符号）：

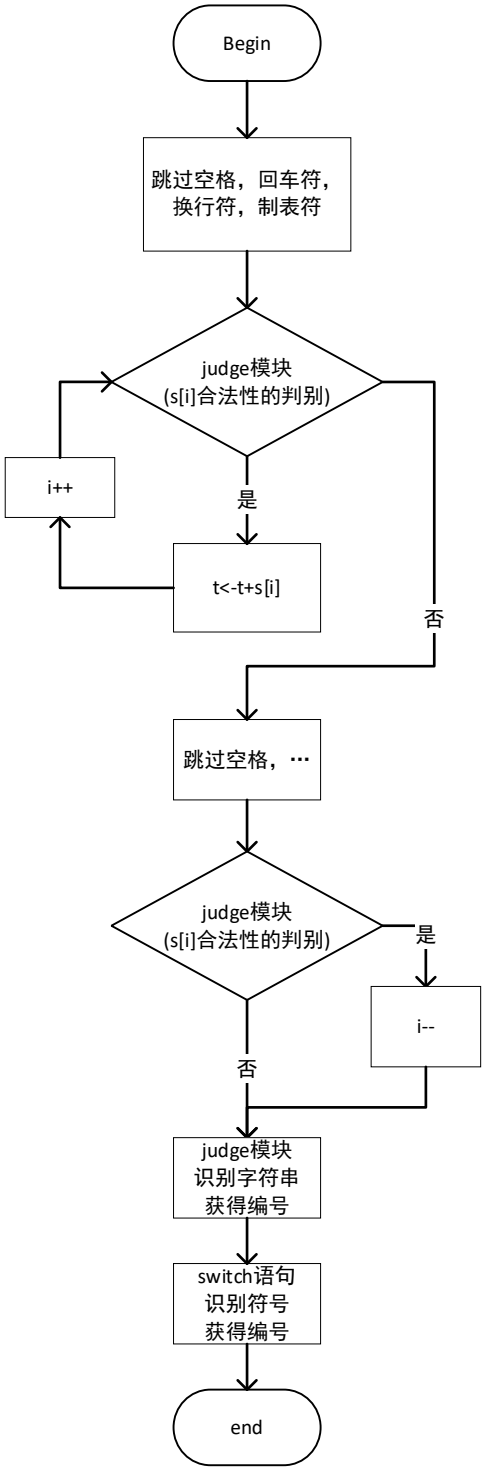

图 3-4 DFA 流程图

2)注释读取：当读到注释界定符时，进入编译模块，读取行注释和块注释，写入AST中，读到换行符或界定符时结束

3.判别模块

流程：四个函数分别实现四个功能。

1)数字类型判别

2)字符判别

3)关键字判别

4)字符串合法字符判别

前三个函数的返回值为当前读取的字符串的类型编号。第四个函数若返回值为0，结束文件的当前字符串读取，否则继续读取文件。

4.编译模块

算法思想：过程型递归（子程序下降递归）

流程：分为两类语句，含关键字的语句和含运算符的语句

1)关键字语句部分：根据获取的字符串编号，通过switch语句选择相应的子函数进行递归调用，直到文件结束。其中读到if语句、for语句、while语句的条件语句或赋值语句时，调用运算符处理程序。return语句若有返回值，读完关键字后调用运算符处理程序。

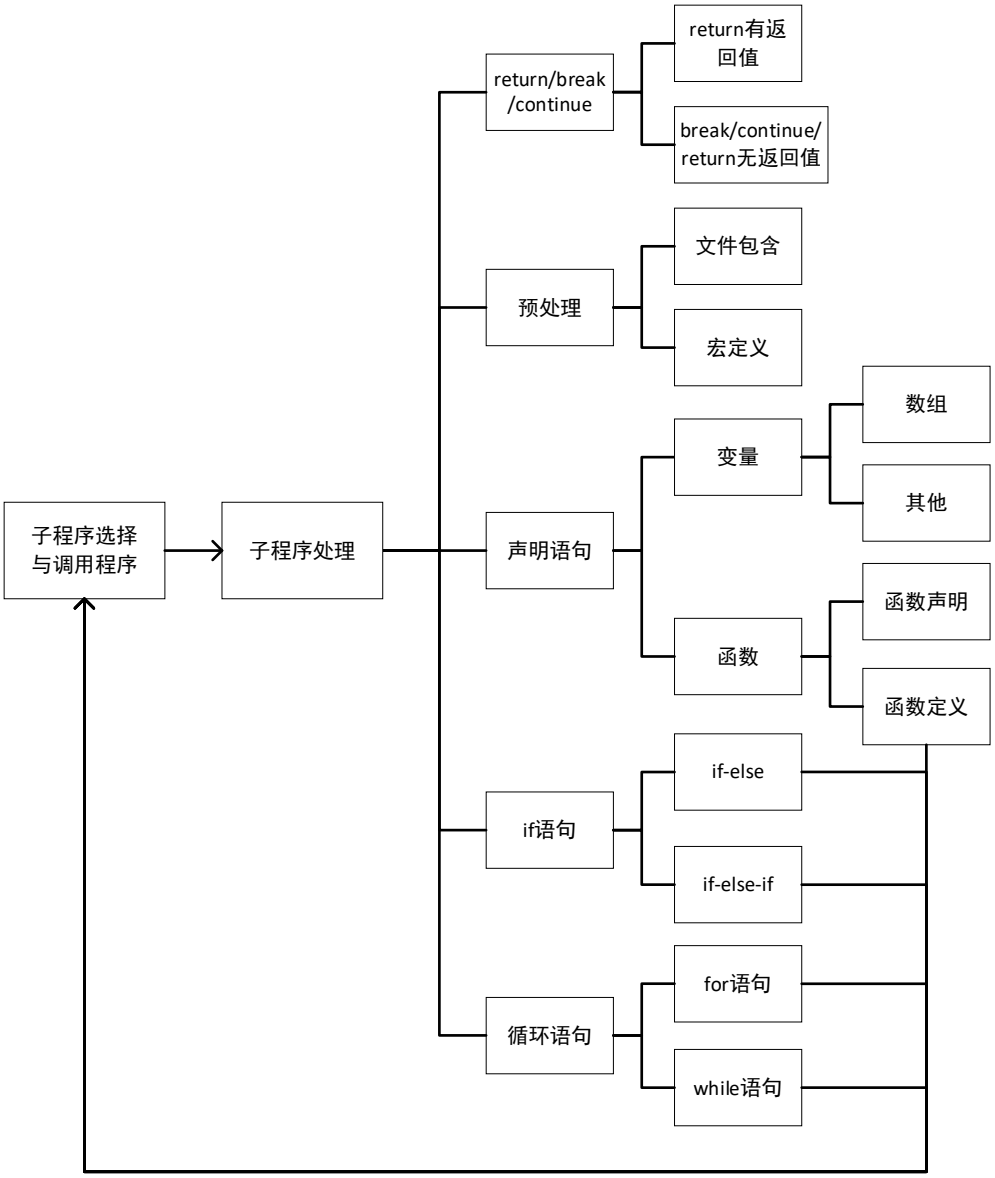

图 3-5 关键字语句部分

2)条件语句，赋值语句，函数调用部分：处理语句的运算符，根据运算符的优先级建 AST。

1.运算符处理程序的作用：判断当前读取的字符串是否是函数，如果是，调用函数调用程序，否则将当前字符串写入 ASR，并进行递归中转

2.递归中转程序的作用：插入运算符结点，更新所有运算符的父亲结点，读取下一字符串和运算符或分隔符。最后调用运算符处理程序，实现递归中转。

3.函数调用程序的作用：处理函数调用，将函数调用部分建入 AST 中，并对函数参数进行判错。

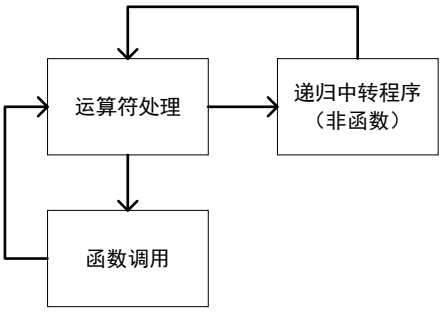

图 3-6 含运算符的语句部分

5.建AST模块

流程：分为两种情况，由编译模块的函数选择调用

1)在当前结点下创建新的孩子结点

2)在当前结点和该结点的最后一个孩子结点之间插入新结点

6.报错模块

流程：根据编译模块模块的指令运行，七个函数实现七个功能：

1) 变量或函数未定义判定
2) 变量或函数重复定义判定
3) 变量与函数命名合法性判定
4) 宏定义命名合法性判定
5) 符号判错
6) 合法字符串判断
7) 其他语法报错(switch语句选择报错类型，包括变量赋值类型报错，函数参数类型报错，无效果语句警告，break、continue、return语句报错等)

7.输出AST模块

算法思想：结构性递归

流程：先根遍历AST，根据结点的深度编排缩进，输出结点中的内容

8.源文件编排模块

算法思想：结构性递归

流程：从根节点开始先根遍历，遇到条件语句、赋值语句、函数调用、含返

回值的返回语句等可能含有运算符的语句，改用中序遍历，含运算符的语句结束后，继续先根遍历。同时根据语句嵌套的层数编排缩进，写入源文件程序中。

4 系统实现与测试

4.1 系统实现

硬件环境：CPU：i7-1165G7，内存：512MB，硬盘：512GB，显卡：2G

软件环境：Windows xp操作系统，Code::Blocks 10.05

根据 3.1 的设计，用 C 语言定义各种数据类型：

```
typedef struct variable{//变量信息储存
    int type;                   //数据类型或函数返回类型
    char name[100];             //变量或函数名
    int param_type;             //函数参数类型
    char param_name[100]; //函数参数名字
    char array_size[100];   //数组变量大小
}variable;
typedef struct Node{        //语法树结点结构
    char key1[100];         //类型
    char key2[100];         //具体信息
    int sum;                //孩子数量
    Node* child[10];        //孩子们
}Node,*Tree;
typedef struct dfa{         //字符出和符号的类型编号结构
    int s,t;
}dfa;
```

函数说明：

1.主程序部分

函数：int main()，执行基本指令，进入其他模块

流程：

1. 新建整个语法树的根节点，将测试文件读入字符串s中。
2. 依次进入编译模块，遍历模块和源程序编排模块。

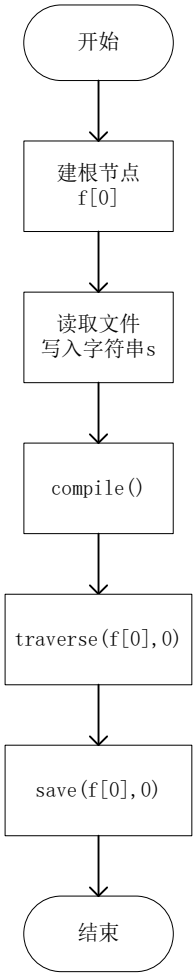

图 4-1 主程序流程图

2.读取模块

函数：

1)void skip()：跳过无意义的字符

流程：跳过连续的空格、\t、\r、\n。读到\n时行数l加一

2)void read()：读取文件，获取类型编号

流程：

1. 按DFA读取一个字符串和一个字符
2. 如果字符串的第一个字符是减号，继续读取
3. 若读到注释界定符，进入编译模块，调用delimit函数，将注释写进AST中。
4. 用switch语句选择，分别获取字符串和字符的类型编号。若有不合法的

符号，进行报错。

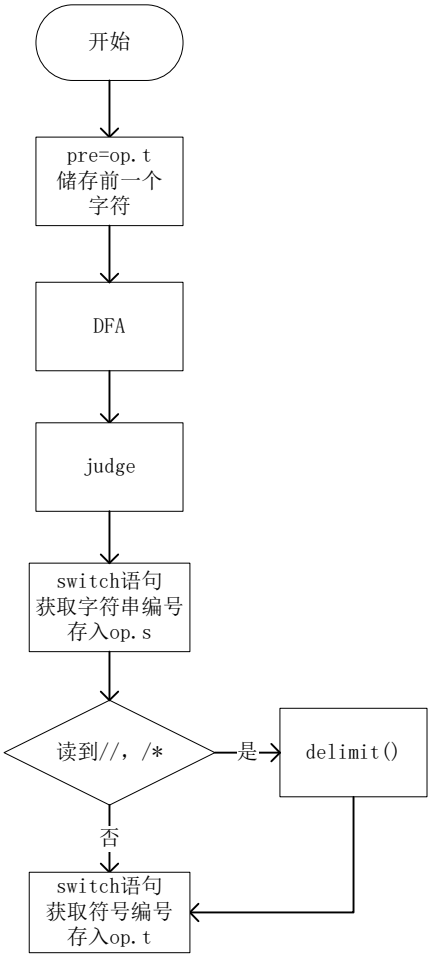

图 4-2 文件读取

3.judge(识别模块)

包括函数：

1)int num_judge(int jj)：数字类型识别，jj为字符串长度

流程：

1. 如果读取的字符串中，第一个字符为‘-’，即读到的是负数，去掉字符串中的符号，字符串长度减一。
2. 如果字符串长度为1且字符为0~9，那么就是int型
3. 否则继续识别，流程如下图

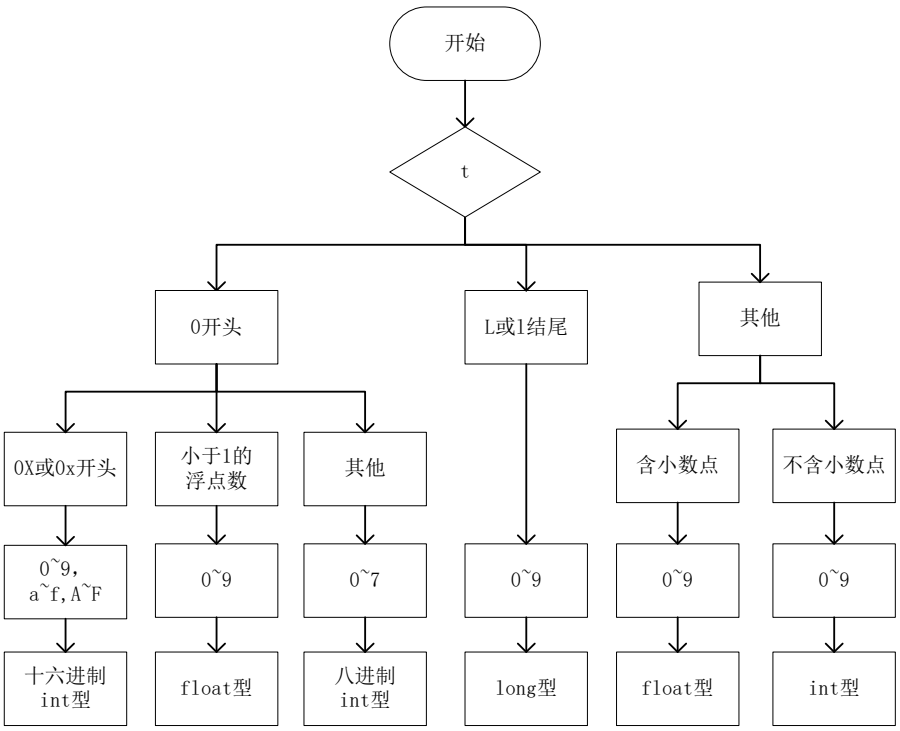

图4-3 数字类型识别

2)int char_judge()：字符识别

流程：若首个字符和末位字符为单引号，则返回char型，否则返回0

3)int key_judge()：关键字识别

流程：根据字符串的值，返回相应的关键字的编号，不是关键字就返回0

4) int judge()：字符串合法字符识别

流程：判断当前读取的字符是否为减号、数字、字母、下划线、小数点或单引号，不是则返回0，否则返回1

4.编译模块

包括函数：

1)int subroutine()：子函数选择和调用程序

流程：

1. 进入读取模块，读取文件内容，获得类型编号
2. 更新父亲结点
3. 若读到‘}’时返回0，表示复合语句块结束。

4. 若文件结束，返回1。

5. 否则根据获取的字符串编号，通过switch语句选择子函数进行递归调用。

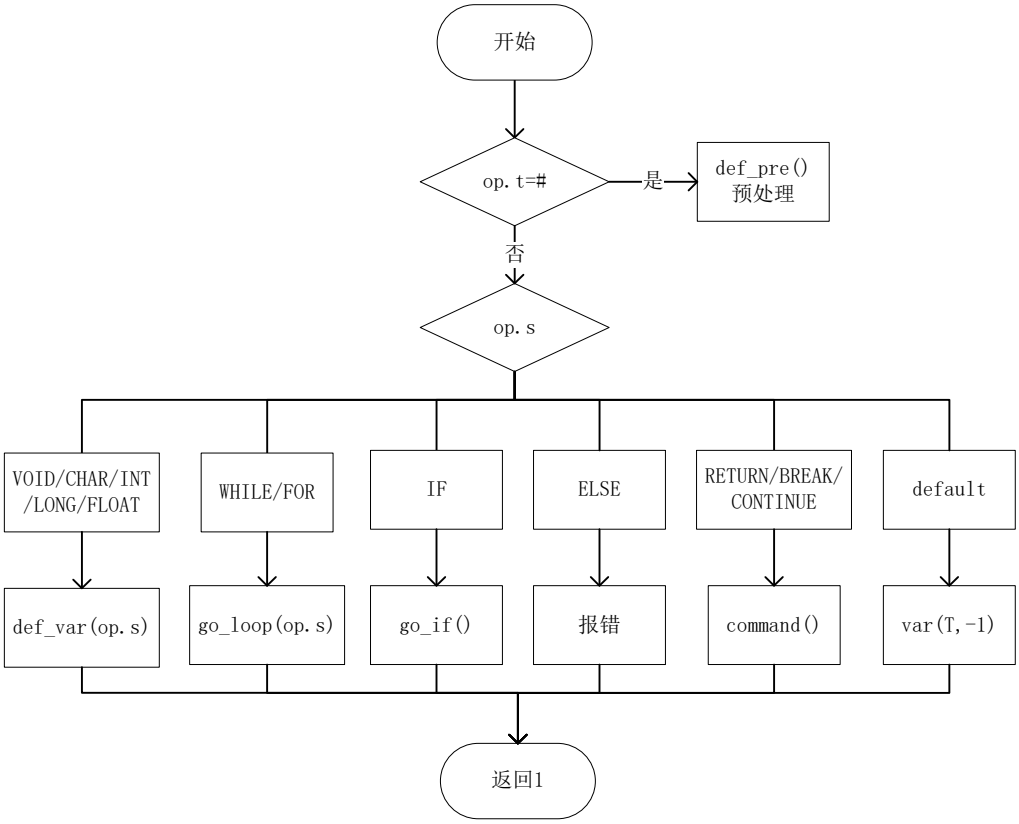

图4-4 子程序调用

2)void compile()：总编译指令

流程：while循环调用subroutine函数。当返回值为0时，代表复合语句块结束，返回原函数，否则继续进行while循环，直到文件结束。

3)void init(int type)：将新定义的宏定义或变量或函数存入st中。

流程：

1. 数据类型赋值为type
2. 数据名赋值为t
3. 参数类型初始化为0。

4)void delimit()：读取注释。

流程：

1. 若为行注释，读到\n结束。若为块注释，读到\n行数l加一，读到*/结束。

2. 读取的字符串存入note，写入AST中。

5)void def_pre()：预处理，文件包含和宏定义

流程：

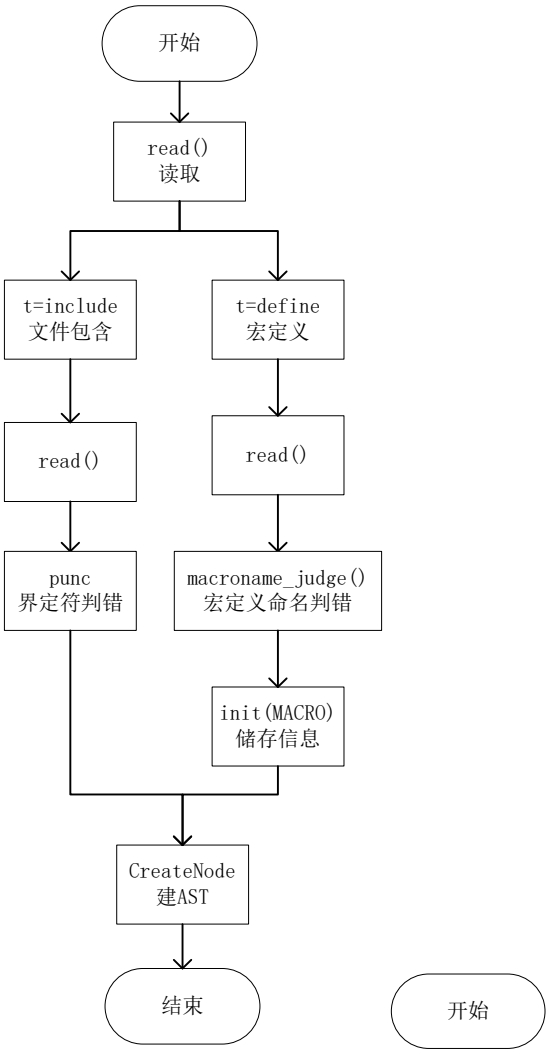

图4-5 预处理子程序

6)void def_var(int type)：变量，函数的声明和定义子程序。

type为关键字的数据类型编号

流程：

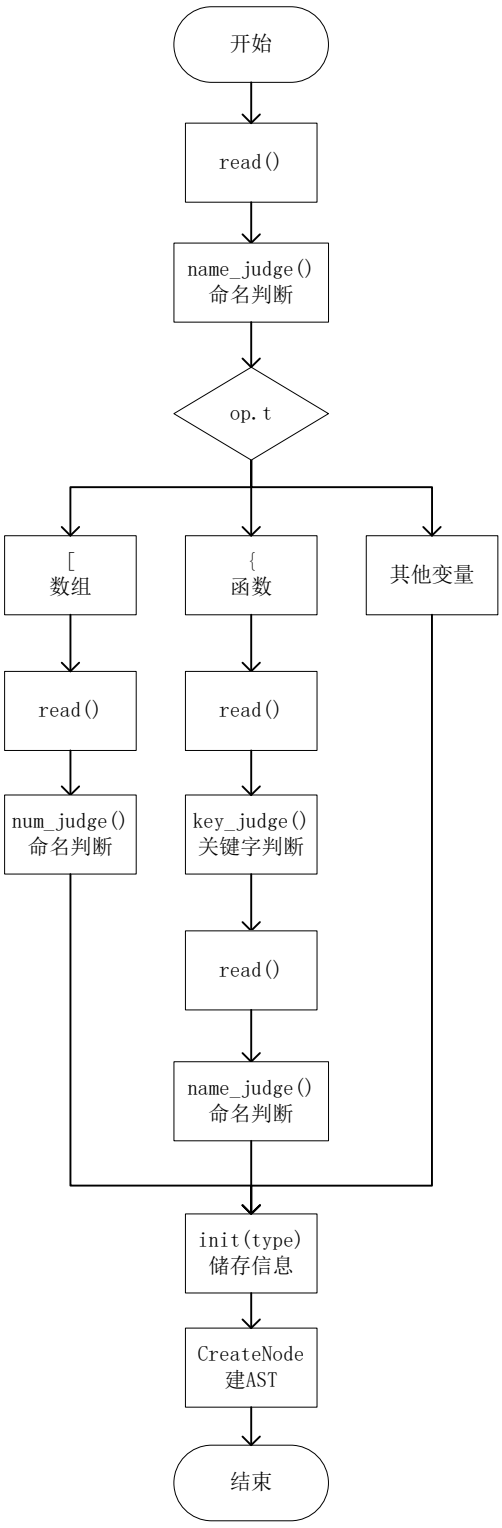

图4-6 声明语句子程序

7) void command()：return，break，continue语句处理

流程：

1. 识别关键字

2. 若是有返回值的return语句，调用运算符处理程序

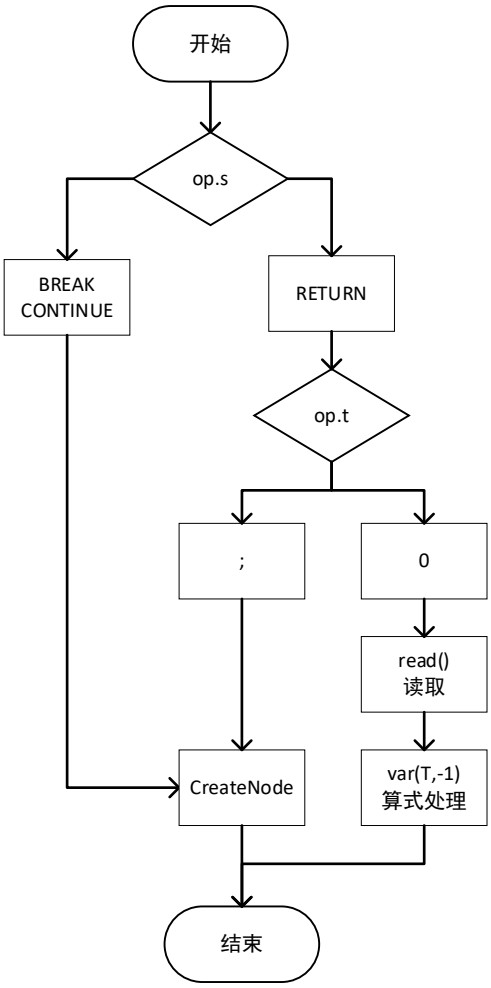

图4-7 命令语句子程序

8) void go_loop(int type)：循环语句处理

type为该语句的关键字的类型编号

流程：

1. 循环层数和嵌套层数分别加一
2. 识别关键字，调用相应函数
3. 复合语句判断，进入下一层嵌套
4. 语句结束后，循环层数和嵌套层数分别减一

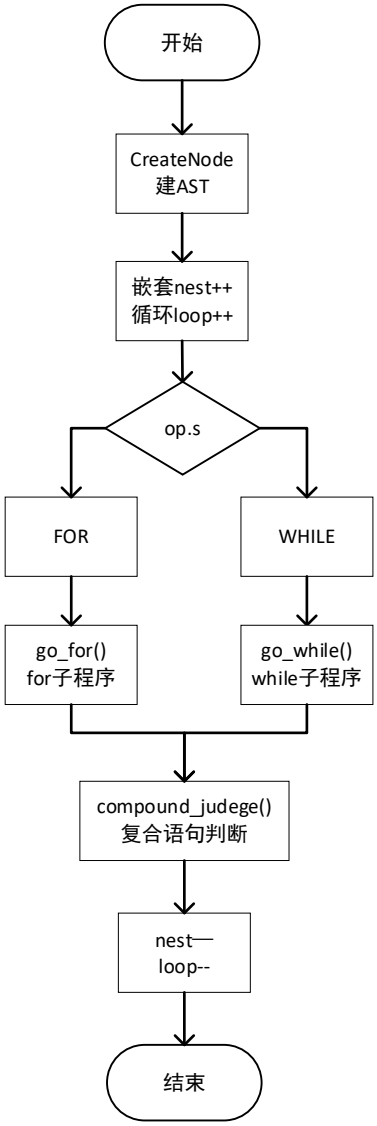

图4-8 循环语句子程序

9) void go_for()：for语句处理

流程：读取括号中的三个字句，依次为赋值语句，条件语句，赋值语句，若语句不为空，即j!=0，调用var(T,-1)进行处理。否则跳过，处理下一个语句。每读完一个子句，判别语句是否有效果和符号是否有错并报错。

10) void go_while()：while语句处理

流程：

1. 读取括号中的条件语句
2. 若语句为空，报错，否则调用var(T,-1)进行条件语句处理。

11) void compound_judge()：判断是否是复合语句并进行递归之前的部分操作

流程：

1. 判断是否是复合语句
2. 进入编译模块，如果为复合语句，调用总编译函数，否则调用子程序递归函数。
3. c赋值为0，表示离开嵌套语句。

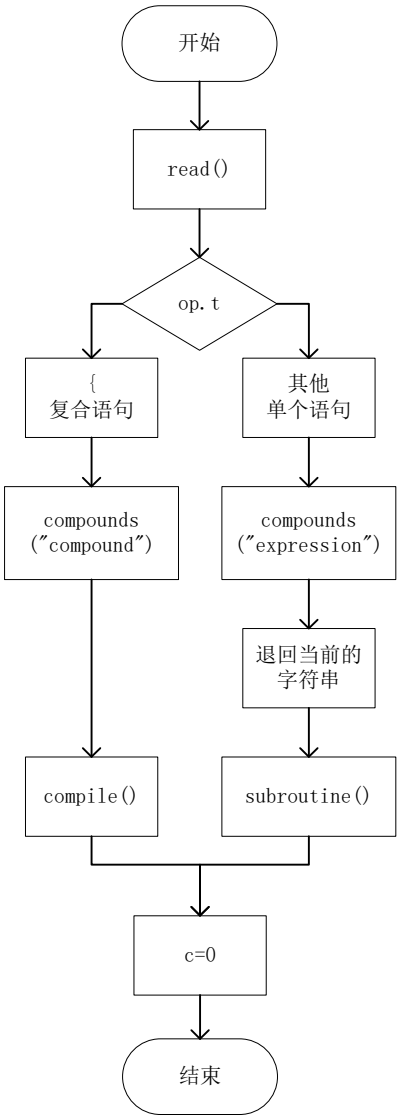

图4-9 复合语句判断

12) void compounds(const char* r)：处理for，while，if的语句嵌套

流程：

1. 字符串r为“expression”或“compound”，表示单个语句或复合语句，作为要

写入AST的信息。

2. 进入建AST模块，调用CreateNode函数新建结点，新结点作为新的父亲结点
3. 嵌套层数nest加一。
4. c赋值为1，表示进入嵌套语句。

13) void reassign(int x)：更新父亲结点

流程：for循环语句，将优先级比x低的运算符的父亲节点更新为当前的父亲结点T。

14) recursion1(int x,int y)：递归中转程序1，针对赋值运算符和括号运算符的递归中转。

x为需要更新的父亲结点的最高优先级，y为当前运算符的优先级

流程：

1. 先更新所有运算符的父亲结点
2. 然后读取下一个要处理的字符串和运算符
3. 进行下一次递归。

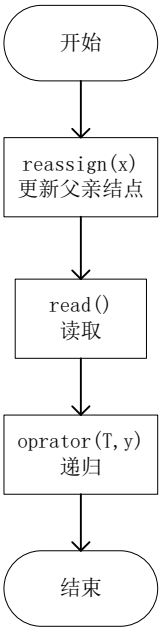

图4-10 递归1

15) recursion2(int x, const char* tt) ：递归中转程序2，针对算数运算符和关系运算符的递归中转。

x为当前运算符的优先级，tt为要写入AST的运算符。

流程：

1. 先在AST中插入当前的运算符
2. 更新所有运算符的父亲结点
3. 然后读取下一个要处理的字符串和运算符
4. 进行下一次递归。

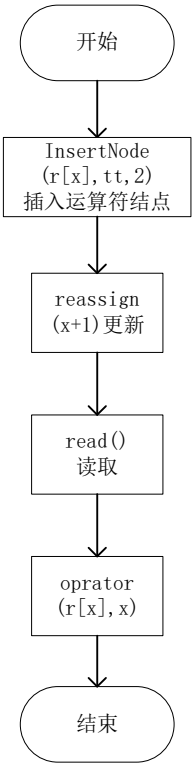

图4-11 递归2

16)void func(Tree root)：函数调用语句处理。

root为建该调用函数结点的根节点。

流程：

1. 先更新父亲结点
2. 判断该函数调用是否有参数，没有就报错返回
3. 然后读取函数参数，判断参数的个数是否错误。
4. 继续递归。

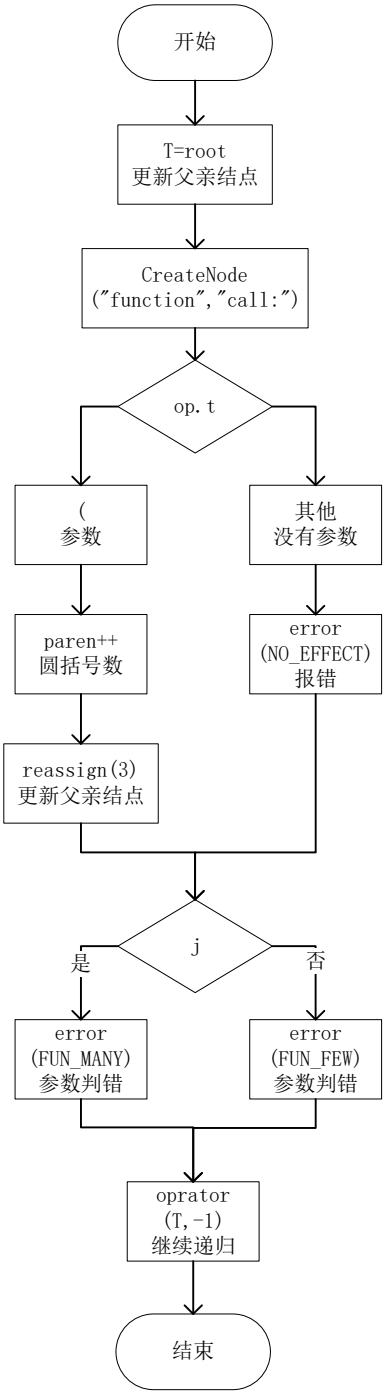

图4-13 函数调用处理

17) void assign()：对“+=”“*=”类型符号的处理

例如j+=i，将算式转化为j=j+i。上一个字符串已经保存在pr中。

流程：

1. 将新的数和赋值运算符加入AST
2. 将pr中的字符串加入AST中

3. 最后将赋值运算符的两个结点调换顺序。

示意图如下

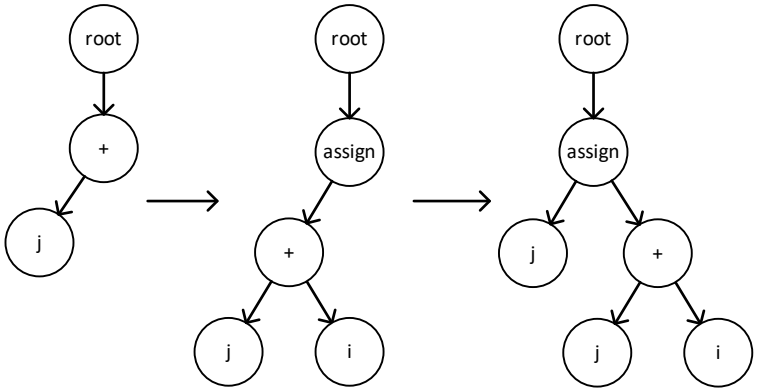

图4-12 算式处理示意图

18)void oprator(Tree root,int x)：运算符处理。

root为根节点指针。若x>=0，则为上一个运算符的优先级，否则作0处理。

流程：

1. 根据x的值选择建AST的根节点，若x>=0，T=root，否则T=root的最后一个孩子。
2. 判别字符串编号op.s。若不是标识符或数字或字符，报错返回。若是函数名，调用func(T)，否则建AST。
3. Str赋值为为运算符字符串。
4. 判别字符编号op.t。若读到'}'、圆括号数paren<0、方括号数bracket<0，报错并返回。
5. 读取其他合法符号的操作具体如下图：

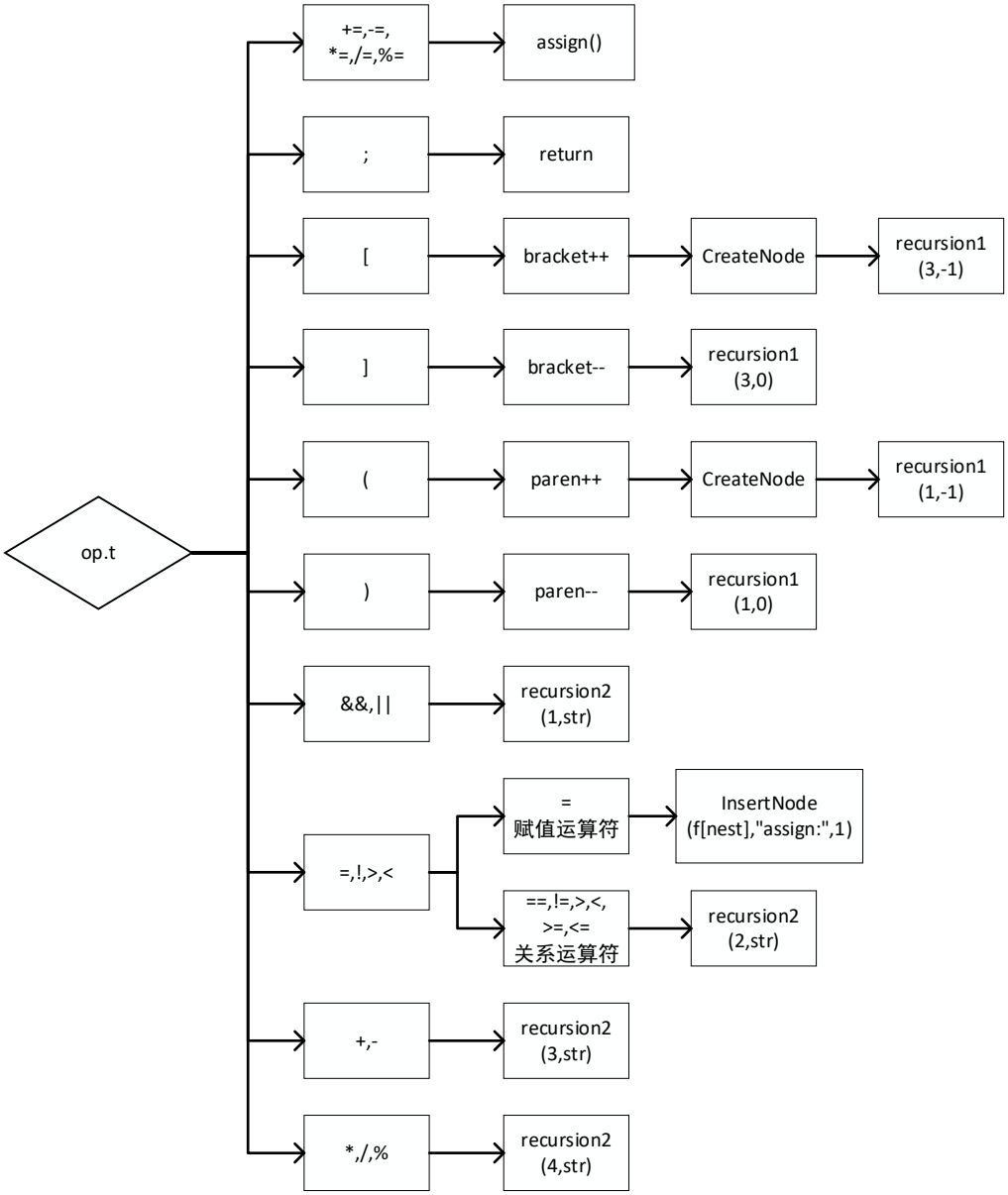

图4-14 运算符处理

5.tree(建树模块)

函数包括：

1)void CreateNode(const char *t1, const char *t2)：在当前结点下创建新的孩子结点

流程：

1. 新建结点p，作为T的第sum个结点
2. t1写入p->key1，t2写入p->key2
3. T的孩子总数T->sum加一

新结点结构如下：

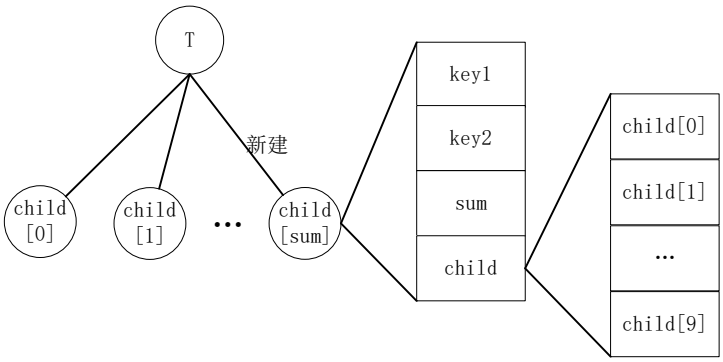

图4-15 结点数据结构

2)void InsertNode(Tree root,const char* str,int x)：在两个结点中插入结点x为标记数字。

流程：

1. 新建结点p
2. x为单数则将字符串写入key1，双数则写入key2。当x=3时，为“+=”类型的运算符，key2写入一个空格作为标记。

示意图如下

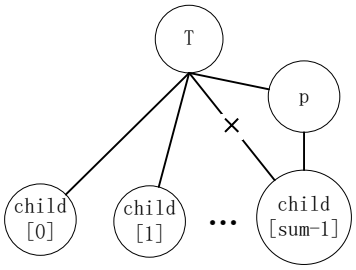

图4-16 插入结点示意图

9.error(报错模块)

包括函数函数：

1)int punc (int x)：标点报错

流程：若op.t!=x，返回0，若op.t不是’;’，退回一个字符。

2)int redeclare_judge()：重复定义判别

流程：遍历st，将每个变量或函数名与当前字符串进行比较，判断是否重复定义字符串t，不是返回1，否则返回0

3)int var_judge()：已定义变量判别

流程：

1. 遍历st，将每个变量或函数名与当前字符串进行比较，判断字符串t是否是已定义的变量或函数
2. 将当前所在的函数的参数与字符串相比较，判断是否相同
3. 如果都不相同返回0，否则返回1

4)void declare_judge()：字符串合法性判别

流程：

1. 调用判别模块中的函数，判断字符串t是否是变量，关键字，数字或字符，若都不是，报错。
2. 若判断为char类型，如果界定符之间的字符个数大于1，警告。

5)void name_judge()：判断变量或函数的命名是否合法

流程：

1. 判断首字母是否为数字
2. 判断是否含有除数字、大小写字母、下划线外的其他符号，若有，报错。

6)void macroname_judge()：判断宏定义的命名是否合法

流程：判断是否有除大写字母、下划线外的其他符号，若有，报错。

7)void error(int x)：其他语法报错，x为报错类型的编号

流程：根据x的值，通过switch语句输出不同类型的报错或警告信息。

10.输出AST模块

函数：void PreOrderTraverse(Tree T,int d)

流程：先根遍历AST，根据结点的深度编排缩进，输出结点中key1和key2字符串的内容

11.源文件编写模块

包括函数：

1) void indent(int d)：按照语句嵌套层数编排缩进，d为当前所在的层数

流程：若d>1，输出(d-1)*4个空格

2) void next(Tree T,int d)：遍历T的所有孩子节点

T为当前所在的结点，d为当前所在的层数

流程：for循环语句，按顺序调用save函数，遍历T的每一个孩子结点

3) void InOrderTraverse(Tree T)：输出条件语句和赋值语句

T为当前所在的结点

流程：

以T为根节点进行中序遍历，输出结点内容：

1. 如果内容为”assign”，输出’=’
2. 如果内容为”=”，输出’==’
3. 如果内容为”paren”，输出左右圆括号
4. 如果内容为”bracket”，输出左右方括号
5. 其他按原样输出。

4) void save(Tree T,int d)：先序遍历，输出语句。

流程：先后对key1和key2进行判别，进行相应操作。具体如下图

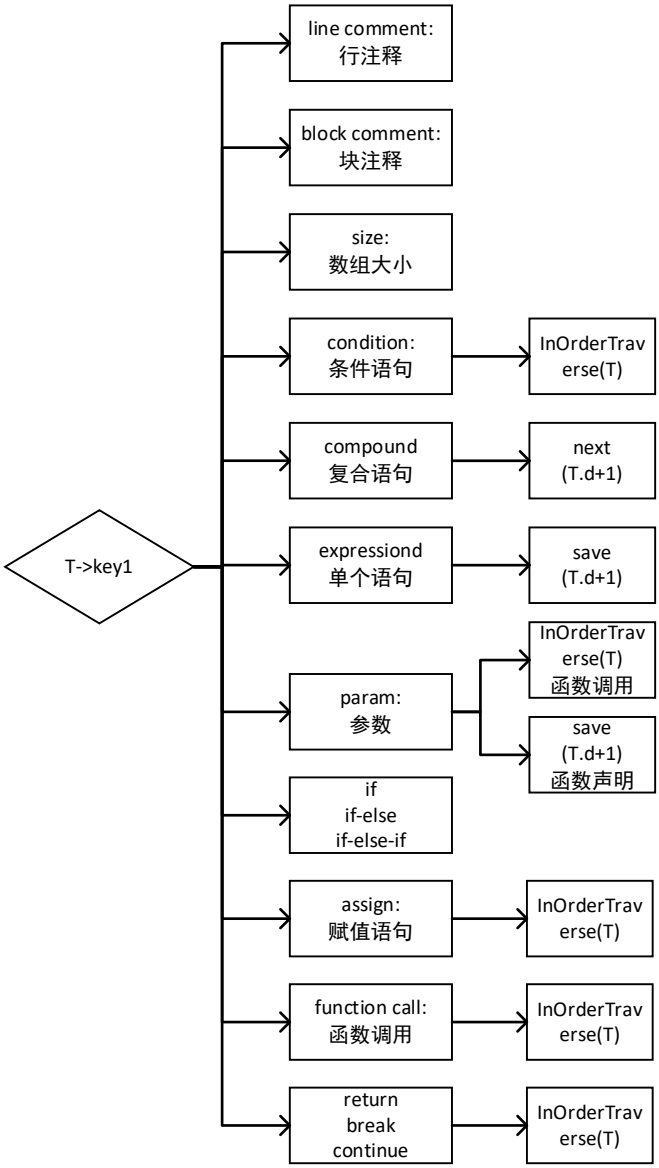

图4-17 生成源程序文件(key1)

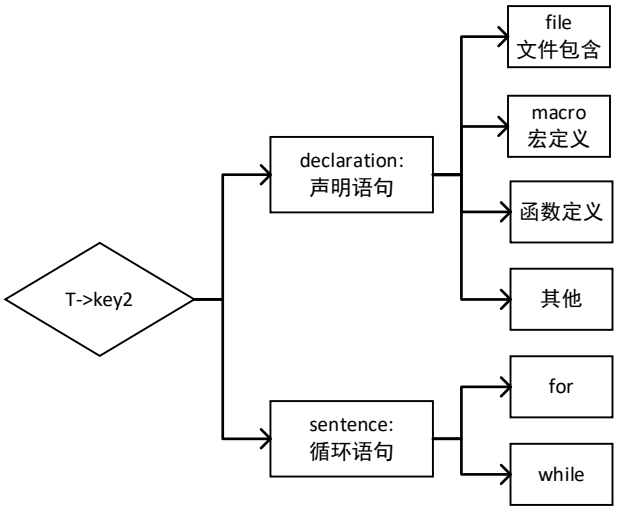

图4-18 生成源程序文件(key2)

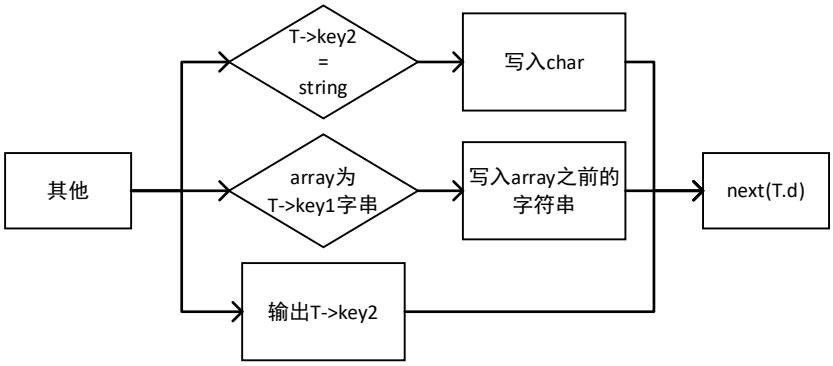

图4-19 生成源程序文件(其它)

由于编译模块递归算法运用多，函数调用关系复杂，且读取模块、报错模块和建树模块的函数调用次数很多，画入图中会过于混乱，所以部分关系不在函数调用图中显示。下图为去除部分读取模块调用关系、报错模块和建树模块的函数调用关系图，但可以展示程序的主要结构和主要运行过程。

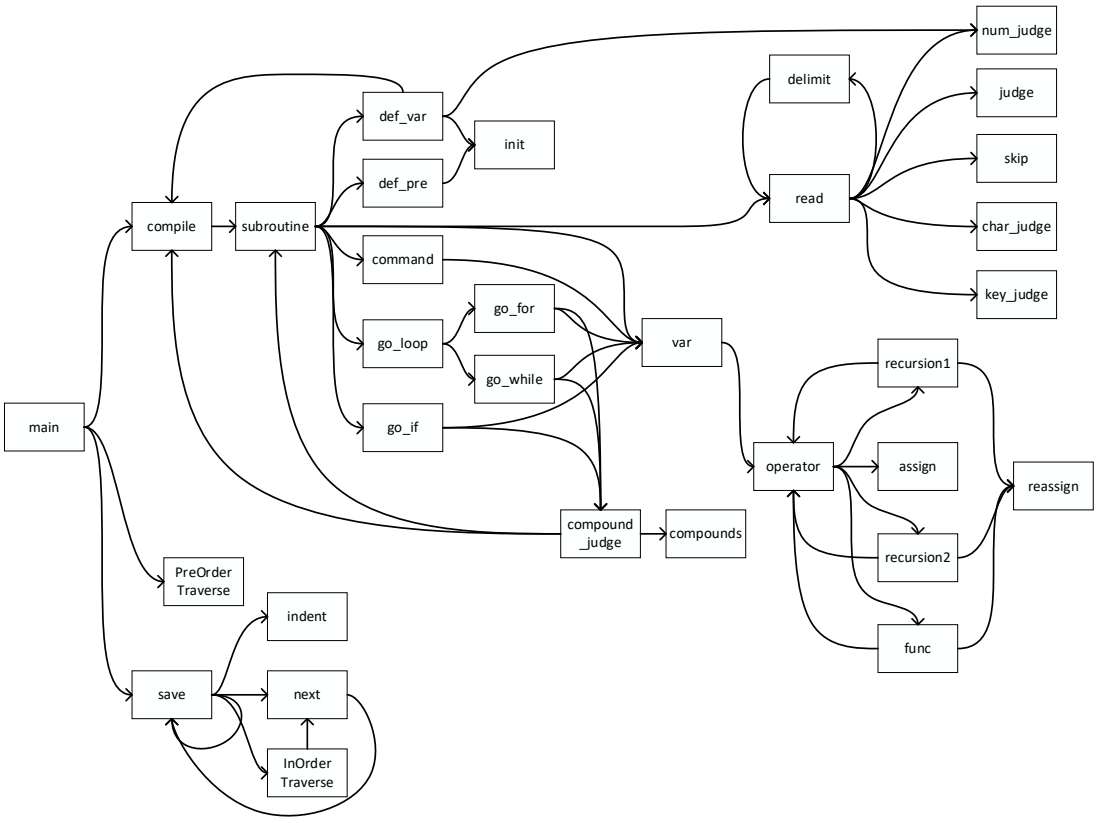

图4-20 函数调用关系图

4.2 系统测试

软件测试方法：

1)静态测试：分析或检查源程序的语句、结构、过程等来检查程序是否有错误

2)动态测试：构造测试实例，执行程序并分析结果

一共选取了三个模块进行测试：

1.编译模块

（1）功能与设计目标

语法结构分析与生成抽象语法树。

要求测试用例包含函数声明，定义、表达式（各种运算符均在某个表达式中出现）、所有的语句，以及 if 语句的嵌套，循环语句的嵌套。测试用例中没有出现的语句和嵌套结构，视作没有完成该种语法结构的分析。

常见的语句包括：

1.表达式语句； 2.if 语句； 3.if else 语句； 4.while 语句;

5.for 语句； 6. return 语句； 7.break 语句； 8.continue 语句；

9.复合语句； 10.函数定义； 11.函数声明； 12.函数调用；

13. if 语句嵌套； 14.循环语句嵌套； 15.外部变量说明语句;

16. 局部变量说明语句。

显示抽象语法树，要求能由抽象语法树说明源程序的语法结构

（2）测试大纲

1.测试目的：通过测试验证系统已经达到设计目标

2.测试环境：

硬件环境：CPU：i7-1165G7，内存：16GB，硬盘：512GB，显卡：2G

软件环境：Windows xp操作系统，Code::Blocks 10.05

3.测试方法

运行程序，输入要测试的文件名。本模块的测试文件名为eg1.c（即没有错误的完整c语言代码）。输入后运行程序。分析结果，检查是否能实现设计目标。

4.测试内容

测试文件：

```
#include<stdio.h>
#include<stdlib.h>
#define N 10
int a[100];
int b(int i){      //5
        int x;
        x=i+1;
        return x+056;           //Here is a line comment.
}
long n;         //10
/*
* Here is a block comment.
*/
void f();
int main(){
        int i;
        int j;
        float x;
        x=1.2;
        char c;
        char s[10];
        s[1]='y';
        n=12L;                  //20
        c='t';
        f();
        i=i+j;
        for(i=1;i<=n||n>0;){        //Here is a line comment.
                a[i+1]=b(i+1)+1;
                j*=i;
                for(j=0;j<N;j=j+1)
                        a[i]=a[i]*2+1;
        }
        i=-1;
        j=0x1a;
        while(j) {                  //30
                i=i<n && (n!=2 || i*2+j/3-n%4>4);
                if(a[i]%2)    //32
                        j=(i+2)*j/5+6;
                else if(a[i]==3) j=j/2-j%(3+i);
                else j*=i;
                if(j){                              //35
                        if(j<0) continue;
                        else j=j-2;
                }
                else break;
        }                                   //40
        return 0;
}

void f(){
        n=n+1;
        return;
}
```

以下是对测试文件的分块说明，以证明测试文件包含设计要求中所有功能的

测试。

1.基本数据类型的变量、常量，以及数组

变量	long n;　　//10 int i; float x; char c;
数组	int a[100]; char s[10];

常量	x=1.2;
	s[1]='y';
	n=12L;　　//20
	c='t';
	i=-1;
	j=0x1a;
	return x+056;　　//Here is a line comment.

2.双目算术运算符（+-*/%），关系运算符、逻辑与（&&）、逻辑或（||）、赋值运算符。不包含逗号运算符、位运算符、各种单目运算符等等

```
a[i+1]=b(i+1)+1;
j*=i;
i=i<n && (n!=2 || i*2+j/3-n%4>4);      //31
```

3.函数定义、声明与调用

函数定义	int b(int i){　　//5
	　　int x;
	　　x=i+1;
	　　return x+056;　　//Here is a line comment.
	}

函数声明	void f();
函数调用	f(); a[i+1]=b(i+1)+1;

4.所有语句

表达式语句	j=(i+2)*j/5+6;

复合语句	void f(){
	　　n=n+1;
	　　return;
	}

if-else语句	if(j<0) continue; else j=j-2;

if-else-if语句	if(a[i]%2)
	　　j=(i+2)*j/5+6;
	else if(a[i]==3) j=j/2-j%(3+i);
	else j*=i;

while语句	while(j) {　　//30
	i=i<n && (n!=2 || i*2+j/3-n%4>4);
	　　if(a[i]%2)　//32

	j=(i+2)*j/5+6;
	else if(a[i]==3) j=j/2-j%(3+i);
	else j*=i;
	if(j){ //35
	if(j<0) continue;
	else j=j-2;
	}
	else break;
	} //40

for语句	for(j=0;j<N;j=j+1) a[i]=a[i]*2+1;

return语句	return x+056; //Here is a line comment.
	return 0;

break语句	else break;
continue语句	if(j<0) continue;
外部变量说明	long n;
局部变量说明	int i;

if语句嵌套	if(j){ //35
	if(j<0) continue;
	else j=j-2;
	}
	else break;

for语句嵌套	for(i=1;i<=n||n>0;){ //Here is a line comment.
	a[i+1]=b(i+1)+1;
	j*=i;
	for(j=0;j<N;j=j+1)
	a[i]=a[i]*2+1;
	}

5.编译预处理

头文件包含	#include<stdio.h> #include<stdlib.h>
宏定义	#define N 10

6.注释

行注释	for(i=1;i<=n||n>0;){ //Here is a line comment.
块注释	/* * Here is a block comment. */

（3）运行结果

本模块的测试结果需要通过输出AST模块来可视化。

运行截图：

C:\Users\Superb\Desktop\课设\bin\Release\课设.exe

```
输入需要编译的文件名：eg1.c
line  message

file declaration:
    Name: stdio.h
file declaration:
    Name: stdlib.h
macro declaration:
    Name: N
    value: 10
extern declaration:
    type: int array
    name: a
    size: 100
function declaration:
    type: int
    name: b
    param:
        type: int
        name: i
    compound sentence:
        line comment: 5
        local declaration:
            type: int
            name: x
        assign:
            ID: x
            +
                ID: i
                ID: 1
        return
            +
                ID: x
                ID: 056
        line comment: Here is a line comment.
extern declaration:
    type: long
    name: n
line comment: 10
block comment:
* Here is a block comment.
function declaration:
    type: int
    name: main
    param:
        type: void
    compound sentence:
        local declaration:
            type: int
            name: i
        local declaration:
            type: int
            name: j
        local declaration:
            type: float
            name: x
```

图4-1 运行截图1

```
assign:
    ID: x
    ID: 1.2
local declaration:
    type: char
    name: c
local declaration:
    type: string
    name: s
    size: 10
assign:
    ID: s
        subscript:
            ID: 1
    ID: 'y'
assign:
    ID: n
    ID: 12L
line comment: 20
assign:
    ID: c
    ID: 't'
function call:
    name: f
    param:
        void
```

```
assign:
    ID: i
    +
        ID: i
        ID: j
for sentence:
    assign:
        ID: i
        ID: 1
    condition:
        ||
            <=
                ID: i
                ID: n
            >
                ID: n
                ID: 0
    null
    compound sentence:
        line comment: Here is a line comment.
        assign:
            ID: a
                subscript:
                    +
                        ID: i
                        ID: 1
            +
                function call:
                    name: b
                    param:
                        +
                            ID: i
                            ID: 1
                ID: 1
```

图4-2 运行截图2


assign:
ID: j
*
ID: j
ID: i
for sentence:
assign:
ID: j
ID: 0
condition:
<
ID: j
ID: N
assign:
ID: j
+
ID: j
ID: 1
expression sentence:
assign:
ID: a
subscript:
ID: i
+
*
ID: a
subscript:
ID: i
ID: 2
ID: 1
assign:
ID: i
ID: -1
assign:
ID: j
ID: 0x1a
while sentence:
condition:
ID: j
compound sentence:
line comment: 30
assign:
ID: i
&&
<
ID: i
ID: n
paren:
||
!=
ID: n
ID: 2
>
-
+
*
ID: i
ID: 2

图4-3 运行截图3

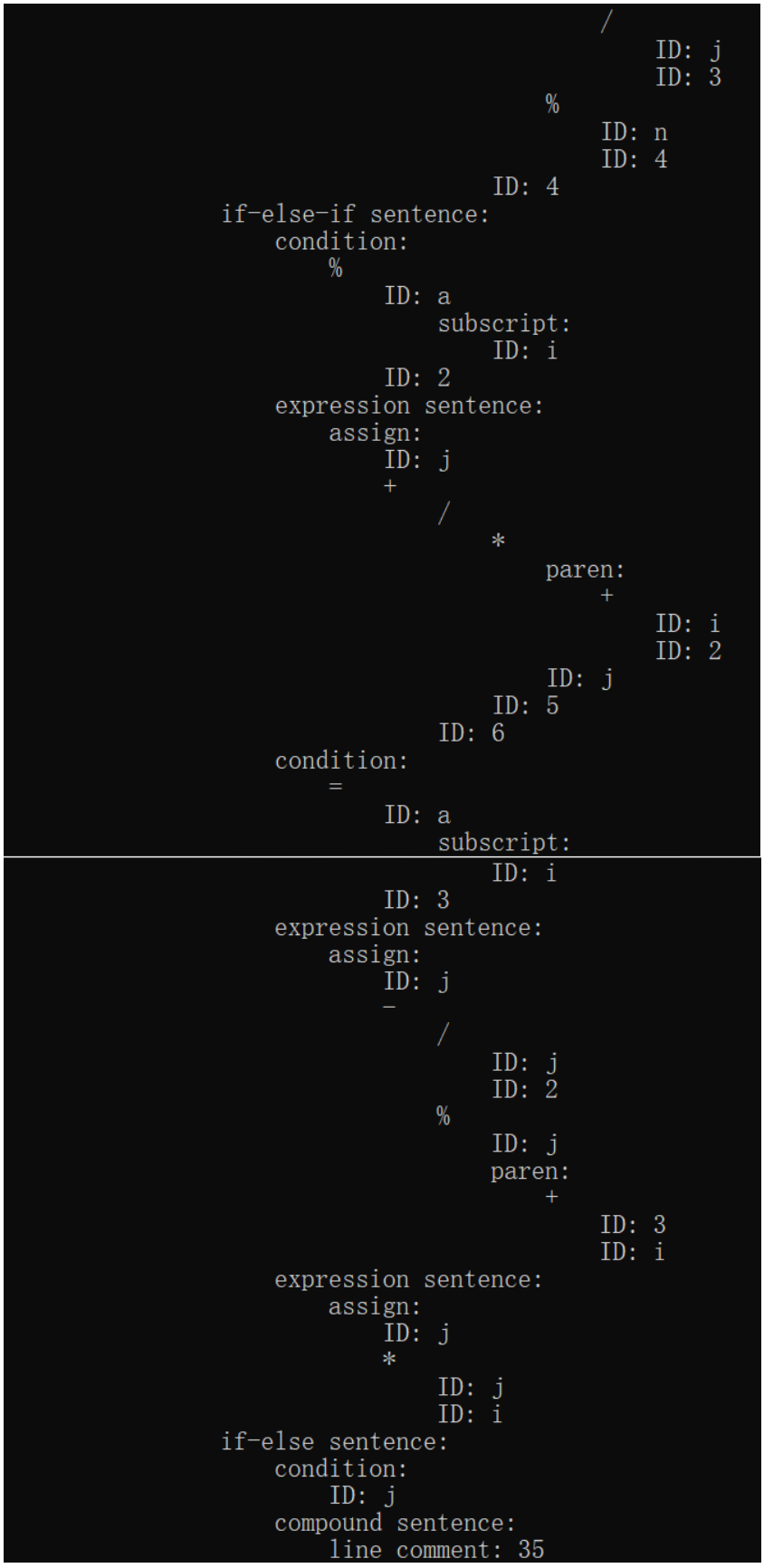
/
ID: j
ID: 3
%
ID: n
ID: 4
ID: 4
if-else-if sentence:
condition:
%
ID: a
subscript:
ID: i
ID: 2
expression sentence:
assign:
ID: j
+
/
*
paren:
+
ID: i
ID: 2
ID: j
ID: 5
ID: 6
condition:
=
ID: a
subscript:
ID: i
ID: 3
expression sentence:
assign:
ID: j
-
/
ID: j
ID: 2
%
ID: j
paren:
+
ID: 3
ID: i
expression sentence:
assign:
ID: j
*
ID: j
ID: i
if-else sentence:
condition:
ID: j
compound sentence:
line comment: 35

图4-4 运行截图4

```
                    if-else sentence:
                        condition:
                            <
                                ID: j
                                ID: 0
                        expression sentence:
                            continue
                        expression sentence:
                            assign:
                                ID: j
                                -
                                    ID: j
                                    ID: 2
                expression sentence:
                    break
        line comment: 40
        return
            ID: 0
function declaration:
    type: void
    name: f
    param:
        type: void
    compound sentence:
        assign:
            ID: n
            +
                ID: n
                ID: 1
        return

Process returned 0 (0x0)   execution time : 3.752 s
Press any key to continue.
```

图4-5 运行截图5

（4）分析

该模块能够处理没有错误的完整c语言代码并生成AST，以下是说明：

1.变量、常量、数组、外部及局部变量定义

```
extern declaration:
    type: int array
    name: a
    size: 100
```

图4-6 外部变量定义&数组

```
        local declaration:
            type: int
            name: x
```

图4-7 局部变量定义

```
        assign:
            ID: x
            ID: 1.2
```

图4-8 变量&常量

2.所有运算符

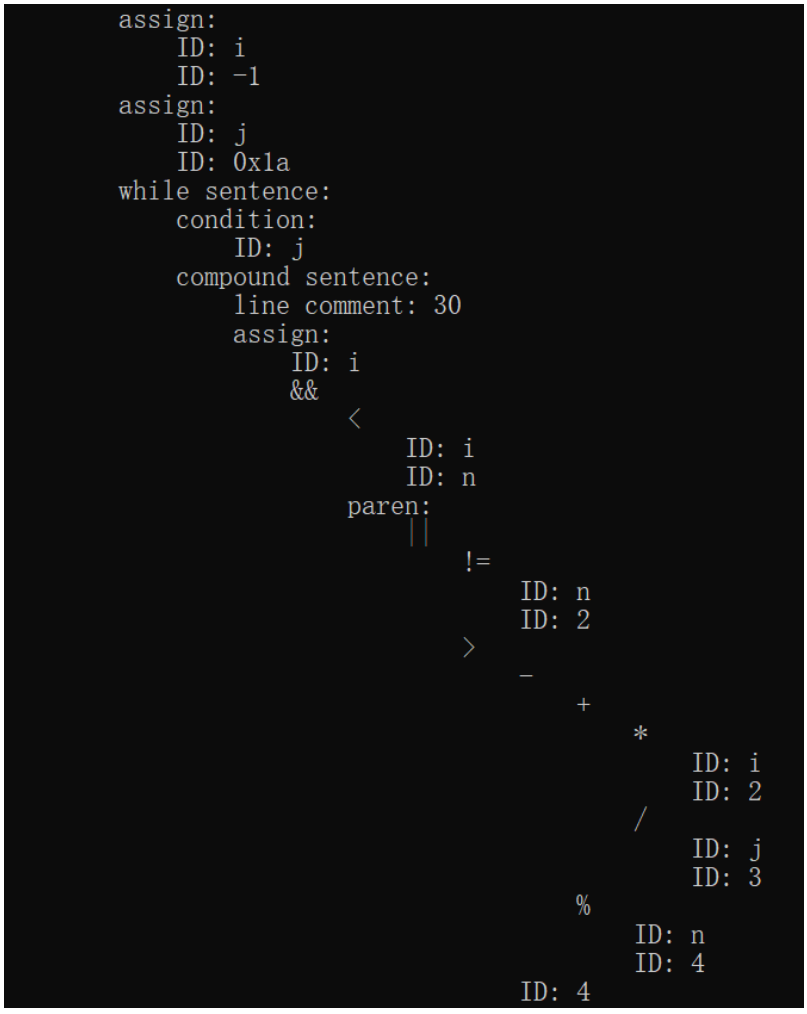

图4-9 运算符

3.函数定义、声明与调用

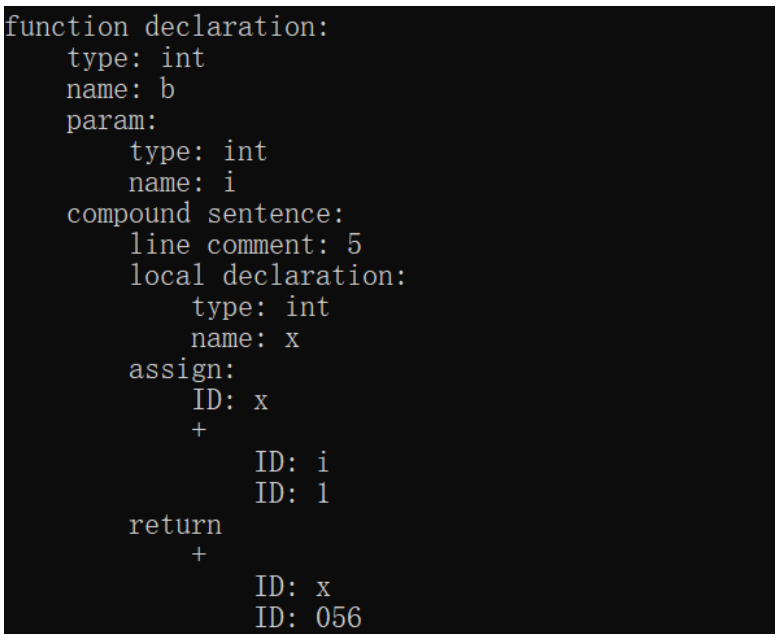

图4-10 函数定义

```
function declaration:
    type: void
    name: f
    param:
        type: void
```

图4-11 函数声明

```
        function call:
            name: f
            param:
                void
```

图4-12 函数调用

4.所有语句

1)for语句

```
                        function call:
                            name: b
                            param:
                                +
                                    ID: i
                                    ID: 1
                        ID: 1
                assign:
                    ID: j
                    *
                        ID: j
                        ID: i
                for sentence:
                    assign:
                        ID: j
                        ID: 0
                    condition:
                        <
                            ID: j
                            ID: N
                    assign:
                        ID: j
                        +
                            ID: j
                            ID: 1
                    expression sentence:
                        assign:
                            ID: a
                                subscript:
                                    ID: i
                            +
                                *
                                    ID: a
                                        subscript:
                                            ID: i
                                    ID: 2
                                ID: 1
```

图4-13 for语句

2)if语句

```
                    if-else sentence:
                        condition:
                            <
                                ID: j
                                ID: 0
                        expression sentence:
                            continue
                        expression sentence:
                            assign:
                                ID: j
                                -
                                    ID: j
                                    ID: 2
```

图4-14 if-else语句

```
            if-else-if sentence:
                condition:
                    %
                        ID: a
                            subscript:
                                ID: i
                        ID: 2
                expression sentence:
                    assign:
                        ID: j
                        +
                            /
                                *
                                    paren:
                                        +
                                            ID: i
                                            ID: 2
                                    ID: j
                                ID: 5
                            ID: 6
                condition:
                    =
                        ID: a
                            subscript:
                                ID: i
                        ID: 3
                expression sentence:
                    assign:
                        ID: j
                        -
                            /
                                ID: j
                                ID: 2
                            %
                                ID: j
                                paren:
                                    +
                                        ID: 3
                                        ID: i
                expression sentence:
                    assign:
                        ID: j
                        *
                            ID: j
                            ID: i
```

图4-15 if-else-if语句

3)while语句

```
while sentence:
    condition:
        ID: j
    compound sentence:
        line comment: 30
```

图4-16 while语句

4)break语句

```
expression sentence:
    break
```

图4-17 break语句

5)continue语句

```
expression sentence:
    continue
```

图4-18 continue语句

6)return语句

```
return
    +
        ID: x
        ID: 056
```

图4-19 return语句

7)复合语句

```
compound sentence:
    assign:
        ID: n
        +
            ID: n
            ID: 1
    return
```

图4-20 复合语句

8)循环语句嵌套

```
for sentence:
    assign:
        ID: i
        ID: 1
    condition:
        ||
            <=
                ID: i
                ID: n
            >
                ID: n
                ID: 0
    null
    compound sentence:
        line comment: Here is a line comment.
        assign:
            ID: a
                subscript:
                    +
                        ID: i
                        ID: 1
            +
                function call:
                    name: b
                    param:
                        +
                            ID: i
                            ID: 1
                ID: 1
        assign:
            ID: j
            *
                ID: j
                ID: i
        for sentence:
            assign:
                ID: j
                ID: 0
            condition:
                <
                    ID: j
                    ID: N
            assign:
                ID: j
                +
                    ID: j
                    ID: 1
            expression sentence:
                assign:
                    ID: a
                        subscript:
                            ID: i
                    +
                        *
                            ID: a
                                subscript:
                                    ID: i
                            ID: 2
                        ID: 1
```

图4-21 循环语句嵌套

9)if语句嵌套

```
if-else sentence:
    condition:
        ID: j
    compound sentence:
        line comment: 35
        if-else sentence:
            condition:
                <
                    ID: j
                    ID: 0
            expression sentence:
                continue
            expression sentence:
                assign:
                    ID: j
                    -
                        ID: j
                        ID: 2
    expression sentence:
        break
```

图4-22 if语句嵌套

5.编译预处理

```
file declaration:
    Name: stdio.h
file declaration:
    Name: stdlib.h
macro declaration:
    Name: N
    value: 10
```

图4-23 编译预处理

6.注释

```
line comment: 10
block comment:
* Here is a block comment.
```

图4-24 注释

2.报错模块

（1）功能与设计目标

报错功能，指出不符合语法规则的错误位置和不符合单词定义的符号位置。

（2）测试大纲

1.测试目的：通过测试验证系统已经达到设计目标

2.测试环境：

硬件环境：CPU：i7-1165G7，内存：16GB，硬盘：512GB，显卡：2G

软件环境：Windows xp操作系统，Code::Blocks 10.05

3.测试方法

运行程序，输入要测试的文件名。本模块的测试文件名为eg2.c（有错误语法和错误符号的完整c语言代码）。输入后运行程序。分析结果，检查是否能实现设计目标。

4.测试内容

测试文件：

```
#include stdio.h>
#include "stdlib.h"
#define N1 10
int a[100];
int b(int i){     //5
    int x;
    x=i+1
    return x+058;          //Here is a line comment.
}
long n;         //10
/*
* Here is a block comment.
*/
void f();   @
int main(){
    int i;
    int j;
    long i;
    float 0x;
    x=1.2.3;
    char c;
    char s[10];
    s[1]='yz';
    n=12Ll;                 //20
    c[0]='t';
    f(1);
    i=i+j;
    for(i=1;i<=n||n>0; {       //Here is a line comment.
        a[i+1]=b()+1;
        j*=i;
        for(j=0;j<n;j+1)
            a=a[i]*2+1;
    }
    i=-1;
    j=0x1a;
    break;
    while j) {              //30
        i=i<n && (n!=2 || i*2+j/3-n%4>4);     //31
        if(a[i%2)   //32
            j=(i+2)*j/5+6;
        else if(a[i]==3) j=j/2-j%(3+i); //33
        else j*=i;
        if(){                           //35
            if(j<0) continue;
```

```
                else j=j-2;
                else j=2;
            }
            else break;
        }                                   //40
        return;
}

void f(){
        n=n+1;
        return 0;
```

以下对文件中错误的语句和符号部分进行说明

错误符号	@
头文件界定符错误	#include stdio.h>
宏定义命名错误	#define N1 10
标识符错误	return x+058;　　//Here is a line comment.
变量名重复	int i; long i;
变量命名错误	float 0x;
数字错误	x=1.2.3;
变量未定义	x=1.2.3;
赋值字符过长	s[1]='yz';
数字错误	n=12Ll;　　//20
变量类型错误	c[0]='t'; a=a[i]*2+1;
函数参数错误	f(1); a[i+1]=b()+1;
分隔符号缺失	x=i+1

括号缺失	for(i=1;i<=n||n>0; {　　//Here is a line comment.
	while j)
	if(a[i%2)　//32
	void f(){
	n=n+1;
	return 0;

无效果语句	for(j=0;j<n;j+1)
break，continue语句不在循环语句中	break;
条件缺失	if(){　　//35
else缺少if	else j=2;
函数返回类型错误	return; return 0;

（3）运行结果

```
输入需要编译的文件名：eg2.c
line  message
 1   error: #include expects 34FILENAME34 or <FILENAME>
 3   error: wrong name defination
 8   error: expected ';'
 8   error: wrong number for '058'
 14   error: error token
 18   error: redeclaration of 'i' with no linkage
 19   error: wrong name definiton
 20   error: x undeclared
 20   error: wrong number for '1.2.3'
 23   warning: multi-character character constant
 24   error: wrong number for '12L1'
 25   error: 'c' is neither array nor pointer
 26   error: too many arguments to function
 28   error: expected ')'
 29   error: too few arguments to function
 32   error: incompatible types for 'a'
 36   error: statement not within loop or switch
 37   error: expected '('
 39   error: expected ']'
 43   error: expected expression before ')' token
 46   error: 'else' without a previous 'if'
 50   error: return-statement with a value in function returning 'void'
 50   warning: statement with no effect
 55   error: return statement with no value
 56   expected '}'

Process returned 0 (0x0)   execution time : 4.954 s
Press any key to continue.
```

图4-25 运行截图

（4）分析

该模块能够完成表格中的所有报错功能。

以下表格将错误内容和报错内容进行对应。

错误符号	@
14 error: error token	
头文件界定符错误	#include stdio.h>
1 error: #include expects 34FILENAME34 or <FILENAME>	
宏定义命名错误	#define N1 10
3 error: wrong name defination	
标识符错误	return x+058; //Here is a line comment.
8 error: wrong number for '058'	
变量名重复	int i; long i;
18 error: redeclaration of 'i' with no linkage	
变量命名错误	float 0x;
19 error: wrong name definiton	
数字错误	x=1.2.3;
20 error: wrong number for '1.2.3'	
变量未定义	x=1.2.3;
20 error: x undeclared	
赋值字符过长	s[1]='yz';
23 warning: multi-character character constant	

数字错误	n=12Ll; //20
24 error: wrong number for '12L1'	
变量类型错误	c[0]='t'; a=a[i]*2+1;
25 error: 'c' is neither array nor pointer 32 error: incompatible types for 'a'	
函数参数错误	f(1); a[i+1]=b()+1;
26 error: too many arguments to function 29 error: too few arguments to function	
分隔符号缺失	x=i+1
8 error: expected ';'	

括号缺失	for(i=1;i<=n||n>0; { //Here is a line comment.
	while j)
	if(a[i%2) //32
	void f(){
	n=n+1;
	return 0;

28 error: expected ')' 37 error: expected '(' 39 error: expected ']' 56 expected '}'	
无效果语句	for(j=0;j<n;j+1)
50 warning: statement with no effect	
break，continue语句不在循环语句中	break;
36 error: statement not within loop or switch	
条件缺失	if(){ //35
43 error: expected expression before ')' token	
else缺少if	else j=2;
46 error: 'else' without a previous 'if'	
函数返回类型错误	return; return 0;
50 error: return-statement with a value in function returning 'void' 55 error: return statement with no value	

3.源文件编排模块

（1）功能与设计目标

缩进编排重新生成源程序文件。对测试模块 1 生成的抽象语法树进行遍历，按缩进编排的方式写到.c 文件中，查看文件验证是否满足任务要求。

（2）测试数据

1.测试目的：通过测试验证系统已经达到设计目标

2.测试环境：

硬件环境：CPU：i7-1165G7，内存：16GB，硬盘：512GB，显卡：2G

软件环境：Windows xp操作系统，Code::Blocks 10.05

3.测试方法

运行程序，输入要测试的文件名。本模块的测试文件名为eg3.c（没有错误的缩进错误的代码）。输入后运行程序。分析结果，检查是否能实现设计目标。

4.测试内容

测试文件：

```
#include<stdio.h>
#include<stdlib.h>
#define N 10
int a[100];
int b(int i){      //5
        int x;
x=i+1;
        return x+056;           //Here is a line comment.
}
long n;         //10
/*
* Here is a block comment.
*/
void f();
int main(){
   int i;int j;
        float x;    x=1.2;
        char c;
        char s[10];
        s[1]='y';
        n=12L;                  //20
  c='t';f();
        i=i+j;
        for(i=1;i<=n||n>0;){        //Here is a line comment.
        a[i+1]=b(i+1)+1;
        j*=i;
        for(j=0;j<N;j=j+1)
        a[i]=a[i]*2+1;
        }i=-1;
        j=0x1a;
        while(j) {                  //30
        i=i<n && (n!=2 || i*2+j/3-n%4>4);
                if(a[i]%2)
                        j=(i+2)*j/5+6;
        else if(a[i]==3) j=j/2-j%(3+i);
else j*=i;
                if(j){                              //35
        if(j<0) continue;
                else j=j-2;
        }
                else break;
        }                                   //40
return 0;
}
void f(){n=n+1;return;}
```

（3）运行结果

```
2   #include<stdio.h>
3   #include<stdlib.h>
4   #define N 10
5   int a[100];
6   int b(int i){        //5
7       int x;
8       x=i+1;
9       return x+056;        //Here is a line comment.
10  }
11  long n;      //10
12  /*
13  * Here is a block comment.
14  */
15  void f();
16  int main(){
17      int i;
18      int j;
19      float x;
20      x=1.2;
21      char c;
22      char s[10];
23      s[1]='y';
24      n=12L;       //20
25      c='t';
26      f();
27      i=i+j;
28      for(i=1;i<=n||n>0;){         //Here is a line comment.
29          a[i+1]=b(i+1)+1;
30          j*=i;
31          for(j=0;j<N;j=j+1)
32              a[i]=a[i]*2+1;
33      }
34      i=-1;
```

图4-26 运行截图1

```
35      j=0x1a;
36      while(j){        //30
37          i=i<n&&(n!=2||i*2+j/3-n%4>4);
38          if(a[i]%2)
39              j=(i+2)*j/5+6;
40          else if(a[i]==3)
41              j=j/2-j%(3+i);
42          else
43              j*=i;
44          if(j){       //35
45              if(j<0)
46                  continue;
47              else
48                  j=j-2;
49          }
50          else
51              break;
52      }        //40
53      return 0;
54  }
55  void f(){
56      n=n+1;
57      return;
58  }
```

图4-27 运行截图2

（4）分析

该模块能够编排缩进，将AST中的内容用c语言代码表示出来。

5 总结与展望

5.1 全文总结

对自己的工作做个总结，主要工作如下：

（1）查阅资料，了解抽象语法树的功能、形式和生成方法

（2）根据设计要求设计编写符合要求的测试文件

（3）测试编写生成 AST 的程序

（4）对测试用例进行测试，根据测试结果修改程序，使其功能符合设计要求。

（5）写实验报告

5.2 工作展望

在今后的研究中，围绕着如下几个方面开展工作

（1）程序运行效率的提高

（2）对更多种类语句（switch、do-while 等）、运算符（逗号运算符、位运算符、单目运算符等）的处理、各种头文件包含的函数（scanf、printf、strcpy 等）的处理

（3）程序稳定性的提升，不会因为读到语法错误而中断运行。

6 体会

在开始编写测试文件和代码之前，我在查找参考资料时遇到了很大问题，因为网上的大部分关于 AST 的资料都基于 java script 语言，而我对这种语言没有任何了解。因此我只能对 AST 的概念和原理进行了解，并尝试使用了几个在线生成 AST 网站。但任务书要求的功能不多，我能够运用原来掌握的知识来实现。在写代码和调试过程中，我对函数的调用、递归以及树的各种功能的实现有了更熟练的掌握。

参考文献

[1]Wikipedia：https://en.wikipedia.org/wiki/Abstract_syntax_tree

[2]https://www.cnblogs.com/qinmengjiao123-123/p/8648488.html

[3]https://blog.csdn.net/weixin_39408343/article/details/95984062

[4]https://zhuanlan.zhihu.com/p/102385477

[5]https://juejin.cn/post/6844903725228621832

[6]https://zhuanlan.zhihu.com/p/81877656

[7]https://github.com/jamiebuilds/the-super-tiny-compiler

7 附录

main.cpp

```
#include <bits/stdc++.h>
using namespace std;
#include "def.h"              //全局变量定义
#include "judge.h"            //判断字符串类型
#include "tree.h"             //建立抽象语法树
#include "read.h"             //读取
#include "error.h"            //报错
#include "traverse.h"         //遍历语法树
#include "save.h"             //编排生成缩进源程序文件
#include "compile.h"          //编译
int main(){
    char filename[10];
    T=(Tree)malloc(sizeof(Node));     //总的根节点
    T->sum=0;
    strcpy(T->key1,"root");
    f[0]=T;         //f 用于储存每层语句的根节点
    printf("输入需要编译的文件名：");
    scanf("%s",filename);
    fp=fopen(filename,"rb");
    while(fscanf(fp,"%c",&s[i])!=EOF) i++;   //将语句读入字符串
    fclose(fp);
    i=-1;
    printf("line    message\n");
    compile();          //编译
    if(nest>0) printf(" %d      expected '}'\n",l);         //复合语句层数报错
    else if(nest<0) printf(" %d      expected '}' deleted\n",l);
    if(strcmp(filename,"eg2.c")){
        printf("\n");
        PreOrderTraverse(f[0],0);     //遍历语法树
        fp=fopen("main.c","wb");
        save(f[0],0);     //遍历语法树并重新编排进源程序文件
        fclose(fp);
    }
    return 0;
}
```

def.h

```
typedef struct variable{//变量信息储存
    int type;                 //数据类型或函数返回类型
    char name[100];           //变量或函数名
    int param_type;           //函数参数类型
```

```
    char param_name[100];//函数参数名字
    char array_size[100];       //数组变量大小
}variable;
typedef struct Node{       //语法树结点结构
    char key1[100];         //类型
    char key2[100];         //具体信息
    int sum;                 //孩子数量
    Node* child[10];        //孩子们
}Node,*Tree;
typedef struct dfa{         //语法树结点结构
    int s,t;
}dfa;
dfa op;
variable st[100];     //变量信息
enum
token_k{ERROR_TOKEN,PLUS,MINUS,PER,SLASH,ASTERISK,AND,OR,ASSI
GN,BANG,LC,RC,LP,RP,LB,RB,LA,RA,SEMI,WELL,QUOT,DELIMIT};   //符号
类型
enum
string_k{VOID=22,INT,LONG,FLOAT,CHAR,IF,ELSE,FOR,WHILE,BREAK,CO
NTINUE,RETURN,CHAR_CONST,IDENT,INT_CONST,LONG_CONST,FLOAT_
CONST,MACRO};
enum
error_k{COMPOUND,CURVES,HEAD,FUN_FEW,FUN_MANY,FUN_VOID,FUN
_NOTVOID,NO_EFFECT,PAREN,BRACKET,ARRAY,IDENTIFIER,LOOP,FLAG
,IF_ELSE};     //报错类型
Tree T,f[10],r[5];
FILE *fp;
char s[100000],t[100],pr[10];
int i,j=1,k,l=1,n,nest,loop,temp,bracket,paren,pre;
bool c;
void compile();         //总编译函数
int subroutine();     //子程序调用
void delimit();         //注释
void save(Tree T,int d);       //生成缩进源程序文件
void oprator(Tree root,int x);   //运算符处理
```

read.h

```
void skip(){       //跳过空格，回车符,制表符和换行符
    while(s[i]==' '||s[i]==9||s[i]==10||s[i]==13){
        if(s[i]==10) l++;     //读到换行符，行数加一
        i++;              //读下一个字符
    }
}

void read(){
```

```
pre=op.t;           //保存上一个标点或无意义字符
i++;                 //读下一个字符
skip();             //跳过无意义字符
for(j=0;judge();i++,j++)
    t[j]=s[i];              //字符串保存至 t 中
if(!j&&s[i]=='-'){          //读负数
    t[0]='-';
    i++;
    skip();
    for(j=1;judge();i++,j++)
        t[j]=s[i];
}
t[j]='\0';
skip();
if(judge()) i--;       //如果最后读到的不是无意义字符，返回前一个字符
//printf("%s %c\n",t,s[i]);
op.s=op.t=0;
if(j){
    if(num_judge(j)) op.s=num_judge(j); //数字
    else if(char_judge()) op.s=CHAR_CONST; //字符
    else if(key_judge()) op.s=key_judge();
    else op.s=IDENT;    //标识符
}
if(s[i]=='/'&&(s[i+1]=='/'||s[i+1]=='*')) delimit(); //读到注释界定符
else{
    switch(s[i]){
        case '(': op.t=LP;break;
        case ')': op.t=RP;break;
        case '[': op.t=LB;break;
        case ']': op.t=RB;break;
        case '{': op.t=LC;break;
        case '}': op.t=RC;break;
        case '&': op.t=AND;break;
        case '|': op.t=OR;break;
        case '<': op.t=LA;break;
        case '>': op.t=RA;break;
        case '+': op.t=PLUS;break;
        case '-': op.t=MINUS;break;
        case '*': op.t=ASTERISK;break;
        case '/': op.t=SLASH;break;
        case '%': op.t=PER;break;
        case ';': op.t=SEMI;break;
        case '=': op.t=ASSIGN;break;
        case '!': op.t=BANG;break;
        case '#': op.t=WELL;break;
        case 34:case 39:op.t=QUOT;break;
        case ' ':case 13:case 10:case 0: return; //无意义字符，返回
        default: printf(" %d      error: error token\n",l);read();break;    //非法字
```

```
符
            }
        }
}
```

judge.h

```
int num_judge(int jj){                          //判断数字的类型
    int ii,cnt,x=INT_CONST;                 //默认 int
    char tt[10];
    if(t[0]=='-'){                          //处理负数
        strcpy(tt,t+1);
        jj--;
    }
    else strcpy(tt,t);
    if(tt[0]<'0'||tt[0]>'9') return 0;   //第一个字符不合法,不是数字
    if(jj>1){     //判断多位数
        if(jj>2&&t[1]!='x'&&t[1]!='X'&&(tt[jj-1]<'0'||tt[jj-1]>'9')){      //有后缀的
数字
            if(tt[jj-1]!='L'&&tt[jj-1]!='l') return -1;
            jj--;
            for(ii=0;ii<jj;ii++)
                if(tt[ii]<'0'||tt[ii]>'9')
                    return -1;                  //字符串中含有不是数字的数，返回
0
            return LONG_CONST;                      //返回 long 类型
        }
        if(tt[0]=='0'){                             //首位为 0
            if(tt[1]=='0') return -1;           //第二位也为 0，返回 0
            else if(tt[1]=='.'){                //小数点为第二位的的 float 类型，如
1.1
                for(ii=2;ii<jj;ii++)
                    if(tt[ii]<'0'||tt[ii]>'9')
                        return  -1;                 //字符串中含有不是数字的数，
返回 0
                return FLOAT_CONST;             //返回 float 类型
            }
            else if(tt[1]=='x'||tt[1]=='X'){       //十六进制数
                for(ii=2;ii<jj;ii++)
              if(!(tt[ii]>='0'&&t[ii]<='9'||t[ii]>='a'&&t[ii]<='f'||t[ii]>='A'&&t[ii]<='F'))
//从第三位开始为 0~9 或 a~f 或 A~F
                        return -1;
                return INT_CONST;
            }
            else{                                           //八进制数
                for(ii=1;ii<jj;ii++)
                    if(tt[ii]<'0'||tt[ii]>'7')   //数字为 0~7
```

```
                        return -1;
                    return INT_CONST;
                }
            }
            else{
                for(cnt=ii=0;ii<jj;ii++){
                    if(tt[ii]=='.'){            //如果包含小数点
                        cnt++;
                        x=FLOAT_CONST;          //为 float 类型
                    }
                    else if(tt[ii]<'0'||tt[ii]>'9'||cnt>1) return -1;      //出现 0~9 之外的
数字或第二个小数点
                }
                return x;
            }
        }
        if(t[j-1]<'0'||t[j-1]>'9') return -1;       //判断一位数
        return INT_CONST;
}

char char_judge(){              //判断是否为字符
        if(t[0]!=39||t[j-1]!=39) return 0;
        if(j>3) return -1;   //字符过长，返回最后一个字符
        return 1;
}

int key_judge(){                //判断是否为关键字
        if(!strcmp(t,"char")) return CHAR;
        if(!strcmp(t,"int")) return INT;
        if(!strcmp(t,"long")) return LONG;
        if(!strcmp(t,"float")) return FLOAT;
        if(!strcmp(t,"for")) return FOR;
        if(!strcmp(t,"while")) return WHILE;
        if(!strcmp(t,"break")) return BREAK;
        if(!strcmp(t,"continue")) return CONTINUE;
        if(!strcmp(t,"else")) return ELSE;
        if(!strcmp(t,"if")) return IF;
        if(!strcmp(t,"return")) return RETURN;
        if(!strcmp(t,"void")) return VOID;
        return 0;
}

int judge(){                //判断是否是字符串中的合法字符
if(s[i]>='0'&&s[i]<='9'||s[i]>='a'&&s[i]<='z'||s[i]>='A'&&s[i]<='Z'||s[i]=='.'||s[i]=='_'||s
[i]==39) return 1;
        return 0;
}
```

compile.h

```
Tree rt,tr;
void reassign(int x){     //更新运算符的父亲节点
      for(int ii=x;ii<5;ii++)
            r[ii]=T;
}
void recursion1(int x,int y){     //递归 1 x 表示优先级
      reassign(x);       //更新
      read();
      oprator(T,y);
}
void recursion2(int x,const char* tt){    //递归 2 x 表示优先级
      InsertNode(r[x],tt,2);         //插入运算符
      reassign(x+1);           //更新
      read();
      oprator(r[x],x);
}

void assign(){
      read();
      CreateNode("ID:",t);
      InsertNode(r[0],"assign:",3);
      CreateNode("ID:",pr);
      swap(T->child[0],T->child[1]);
}

void func(Tree root){            //函数调用处理，root 为当前父亲结点
      T=root;
      CreateNode("function call:","\0");         //函数调用
      T=T->child[T->sum-1];
      CreateNode("name:",t);                      //函数名
      if(op.t!=LP){                        //没有参数
            error(NO_EFFECT);              //
            punc(';');
            return;
      }
      paren++;                                 //括号层数加一
      CreateNode("param:","\0");    //参数
      T=T->child[T->sum-1];
      reassign(3);                          //更新运算符的父亲节点
      read();
      if(!j){         //参数为空
            CreateNode("\0","void");
            if(st[k].param_type!=VOID)  error(FUN_FEW);       //参数类型与定义不
符，报错
      }
      else     if(st[k].param_type==VOID&&strcmp(t,"void"))     error(FUN_MANY);
```

```
//参数类型与定义不符，报错
    oprator(T,-1);
}

void oprator(Tree root,int x){   //x 为当前的运算符优先级
    if(x<0) T=root;        //可以直接在 root 上加新结点
    else T=root->child[root->sum-1];      //在 root 的最后一个孩子节点加新结点
    if(j){
        if(op.s<34&&op.s>0||pre==RP){ //读到
            i-=j;i--;
            return;
        }
        declare_judge(); //判断变量或函数是否已声明
        if(k!=n&&st[k].param_type){     //函数调用
            tr=root;      //记录当前根节点
            func(T);
        }
        else  if(k!=n&&st[k].array_size[0]&&op.t!=LB)  error(ARRAY);     //数组
赋值报错
        else CreateNode("ID:",t);
    }
    else if(op.t==RA){i--;return;}        //读到‘}’，返回
    else
if(op.t==ASSIGN&&(pre==PLUS||pre==MINUS||pre==ASTERISK||pre==SLASH||p
re==PER)){
        assign();
        return;
    }
    else
if(op.t!=LP&&pre!=RB&&pre!=RP&&!(op.t==RP&&pre==RB)&&!(op.t==SEMI&
&pre==LP)&&!(op.t==RP&&pre==LP)){
        strcpy(r[0]->key2,"null");
        r[0]->sum=0;
        printf(" %d     error: expected expression before '%c' token\n",l,s[i]);     //
运算符前没有变量，报错
    }
    char str[3];
    str[0]=s[i],str[1]='\0';      //符号存入字符串
    switch(op.t){
        case SEMI: return;     //语句结束
        case LB:
            if(!st[k].array_size[0]) error(IDENTIFIER);     //非数组类型报错
            rt=root;      //记录当前根节点
            bracket++;   //方括号层数+1
            T=T->child[T->sum-1];
            CreateNode("subscript:","\0");
            T=T->child[T->sum-1];
            recursion1(3,-1);     //读数组下标
```

```
            break;
        case RB:
            bracket--;   //方括号层数-1
            if(bracket<0) return; //方括号数错误，返回
            T=rt;    //返回记录的根节点
            recursion1(3,0);      //递归
            break;
        case LP:
            tr=T;    //记录当前根节点
            paren++;      //括号数+1
            CreateNode("paren:","\0");
            T=T->child[T->sum-1];
            recursion1(1,-1);    //读括号内的算式
            break;
        case RP:
            if(paren<1) return; //条件语句结束，返回
            paren--;      //括号数-1pre
            T=tr;    //返回记录的根节点
            recursion1(1,0);
            break;
        case AND:case OR:   //优先级为 1
            str[1]=s[i],str[2]='\0';
            if(s[i+1]==s[i]) i++;
            else printf(" %d     error: expected '%s'\n",l,str);     //只接受‘||’或
‘&&’
            recursion2(1,str);
            break;
        case ASSIGN:case LA:case RA:case BANG: //优先级为 2
            if(s[i+1]!='='&&op.t==ASSIGN){        //赋值
                InsertNode(f[nest],"assign:",1);
                recursion1(1,-1);
            }
            else{
                if(s[i+1]=='='){             //两个字符的运算符
                    if(op.t!=ASSIGN) str[1]=s[i+1],str[2]='\0';
                    i++;
                }
                recursion2(2,str);
            }
            break;
        case PLUS:case MINUS:   //优先级为 3
            strcpy(pr,t);    //pr 用于储存上一个字符串
            recursion2(3,str);
            break;
        case ASTERISK:case SLASH:case PER: //优先级为 4
            strcpy(pr,t);
            recursion2(4,str);
            break;
```

```
            default: if(strcmp(f[nest]->key1,"for")) punc(';'); //不合法字符
        }
}

void var(Tree root,int x){
        paren=bracket=0;        //括号层数初始化
        for(int  ii=0;ii<5;ii++)  r[ii]=root;  //将不同优先级的运算符的父亲节点都更新为当前根节电
        oprator(root,x);      //编译
        if(paren>0) error(PAREN);
        if(bracket>0) error(BRACKET);
}

void init(int type){                //将已定义的变量和函数保存在结构体数组中
        st[n].type=type;
        strcpy(st[n].name,t);
        st[n++].param_type=0;             //参数类型初始化为 0
}

void compounds(const char* r){
        CreateNode(r,"sentence:");
        f[++nest]=T->child[T->sum-1];     //进入 if，for，while 或函数声明之下的语句
}

void compound_judge(){
        read();
        if(op.t==LC){
                if(j) error(COMPOUND);
                compounds("compound");
                compile();
        }
        else{
                c=1;
                i-=j;i--;
                compounds("expression");
                subroutine();
                nest--;
                c=0;
        }
}

void delimit(){                 //注释处理
        int jj;
        char str[3];
        Tree tmp=T;
        T=f[nest];
        str[0]=s[i],str[1]=s[i+1],str[2]='\0';
        i+=2;
```

```
    if(s[i-1]=='/'){            //行注释
        char* note=strtok(s+i,"\r");      //读取换行符之前的所有字符
        i+=strlen(note);
        if(!c) CreateNode("line comment:",note);
    }
    else{           //块注释
        char note[100];
        for(jj=0;s[i]!='*'||s[i+1]!='/';i++,jj++){   //读取界定符之前的换行符
            note[jj]=s[i];
            if(s[i]==10) l++;
        }
        note[jj-1]='\0';i++;
        if(!c) CreateNode("block comment:",note);
        str[0]='*',str[1]='/';
    }
    T=tmp;
    if(!j) read();
}

void def_pre(){
    read();
    if(!strcmp(t,"include")){           //文件包含
        if(s[i]!=34&&s[i]!='<') error(HEAD);      //标点报错
        read();
        CreateNode("file","declaration:");
        T=T->child[T->sum-1];
        CreateNode("Name:",t);
        if(s[i]!=34&&s[i]!='>'||pre==34&&s[i]!=34||pre=='<'&&s[i]!='>')
            error(HEAD);            //标点报错
    }
    else if(!strcmp(t,"define")){     //宏定义
        read();             //读名字
        macroname_judge();          //命名合法性判断
        init(MACRO);            //储存宏定义
        CreateNode("macro","declaration:");
        T=T->child[T->sum-1];
        CreateNode("Name:",t);
        i++;
        char *tt=strtok(s+i,"\r");   //读定义内容
        i+=strlen(tt);
        CreateNode("value:",tt);
    }
}

void def_var(int type){         //变量和函数定义
    char str1[10],str2[10];
    strcpy(str1,t);
    read();
```

```
    name_judge();              //判断命名合法性
    init(type);
    if(op.t==LP&&!nest){     //函数声明
        temp=n-1;            //记录函数序号
        CreateNode("function","declaration:");
        f[++nest]=T->child[T->sum-1];
        T=f[nest];
        CreateNode("type:",str1);            //返回值类型
        CreateNode("name:",t);                 //名字
        CreateNode("param:","\0");   //参数
        T=T->child[T->sum-1];
        read();
        if(!j){                //无参数
            op.s=VOID;
            strcpy(str1,"void");
        }
        else{
            switch(op.s){
                case VOID: strcpy(str1,"void");break;
                case CHAR: strcpy(str1,"char");break;
                case INT: strcpy(str1,"int");break;
                case LONG: strcpy(str1,"long");break;
                case FLOAT: strcpy(str1,"float");break;
                default: printf(" %d      error: wrong parameter type\n",l);       //不
是数据类型关键字，报错
            }
        }
        st[n-1].param_type=op.s;        //储存参数类型
        CreateNode("type:",str1);
        if(op.s!=VOID){
            read();
            name_judge();       //判断命名合法性
            strcpy(st[n-1].param_name,t);        //储存参数名字
            CreateNode("name:",t);
        }
        punc(')');          //标点报错
        read();
        T=f[nest];
        if(op.t==LC){          //复合语句
            if(j) error(COMPOUND);          //若‘{’前有字符，报错
            compounds("compound");
            compile();
            nest--;                //定义结束后退出一层
            n=temp+1;             //函数内部变量删除
        }
        else{
            if(!punc(';')) i-=j,i--;       //标点报错
            nest--;
```

```
            }
            temp=0;
        }
        else if(op.t==LB){        //数组
            if(type!=CHAR) strcpy(str2,strcat(str1," array"));        //数组
            else strcpy(str2,"string");        //字符串
            if(!nest) CreateNode("extern","declaration:");          //外部变量
            else CreateNode("local","declaration:");        //局部变量
            T=f[nest]->child[f[nest]->sum-1];
            CreateNode("type:",str2);
            CreateNode("name:",t);
            read();
            punc(']');          //标点报错
            if(!num_judge(j)) printf(" %d     error: wrong number\n",l); //判断数字合
法性
            strcpy(st[n-1].array_size,t);       //保存数组大小
            CreateNode("size:",t);
            read();
            punc(';');        //标点报错
        }
        else{
            if(!nest) CreateNode("extern","declaration:");
            else CreateNode("local","declaration:");
            T=T->child[T->sum-1];
            CreateNode("type:",str1);
            CreateNode("name:",t);
            punc(';');
        }
}

void command(){       //return,break,continue 语句处理
        CreateNode(t,"\0");
        if(op.s==RETURN){
            if(nest<=0) error(FLAG);       //不在函数中，报错
            if(op.t!=SEMI){
                if(st[temp].type==VOID) error(FUN_VOID);       //函数返回值报错
                read();
                T=T->child[T->sum-1];
                var(T,-1);
            }
            else if(st[temp].type!=VOID) error(FUN_NOTVOID);       //函数返回值报
错
        }
        else if(loop<=0)    error(LOOP);     //不在循环中，报错
        punc(';');
}

void go_if(){       //if 语句处理
```

```
    CreateNode(t,"sentence:");
    punc('(');
    f[++nest]=T->child[T->sum-1];     //进入下一层
    T=f[nest];
    read();
    Tree root=T;              //保存根节点
    CreateNode("condition:","\0");   //条件
    T=T->child[T->sum-1];
    if(j) var(T,-1);       //处理条件语句
    else{
        error(CURVES);          //语句为空
        CreateNode("\0","null");
    }
    punc(')');          //标点报错
    T=f[nest];
    compound_judge();
    int fl=0;
    while(1){
        read();
        if(op.s==ELSE){          //读 else
            fl++;
            read();
            T=root;
            if(op.s==IF){       //else-if 语句处理
                fl++;
                CreateNode("condition:","\0");
                T=T->child[T->sum-1];
                punc('(');
                read();
                var(T,-1);          //处理条件语句
                punc(')');
                T=f[nest];
                compound_judge();
            }
            else{            //else 语句处理
                i-=j;i--;
                compound_judge();
                break;
            }
        }
        else{
            i-=j;i--;   //没有 else 或 else-if 语句，重读
            break;
        }
    }
    if(fl) strcat(f[nest]->key1,"-else");
    if(fl>1) strcat(f[nest]->key1,"-if");
    nest--;        //退出一层
}
```

```
void go_for(){     //for 循环
    if(j) var(T,-1);       //处理 for 的第一个赋值语句
    else CreateNode("\0","null");
    punc(';');
    if(strcmp(f[nest]->child[f[nest]->sum-1]->key1,"assign:"))
        error(NO_EFFECT);       //不是赋值语句，报错
    T=f[nest];
    CreateNode("condition:","\0");
    T=T->child[T->sum-1];
    read();
    if(j) var(T,-1);       //处理 for 的第二个条件语句
    else CreateNode("\0","null");
    punc(';');
    T=f[nest];
    read();
    if(j) var(T,-1);       //处理 for 的第三个赋值语句
    else CreateNode("\0","null");
}

void go_while(){         //while 循环
    CreateNode("condition:","\0");
    T=T->child[T->sum-1];
    if(j) var(T,-1);       //处理条件语句
    else{
        error(CURVES);
        CreateNode("\0","null");
    }
}

void go_loop(int type){         //循环语句
    loop++;         //嵌套数+1
    punc('(');
    CreateNode(t,"sentence:");
    f[++nest]=T->child[T->sum-1];     //进入下一层
    T=f[nest];
    read();
    if(type==WHILE) go_while();         //while 语句
    else go_for();             //for 语句
    punc(')');
    T=f[nest];
    compound_judge();
    nest--;
    loop--;
}

void compile(){       //编译
    while(s[i]||j)          //读至文件结束
```

```
        if(!subroutine())
            return;
}

int subroutine(){
    read();
    if(!s[i]&&!j) return 1;
    if(op.t==RC){   //读到'}'
        if(j) error(COMPOUND);
        nest--;        //退出一层
        return 0;
    }
    T=f[nest];
    if(!j&&op.t==WELL) def_pre();        //预处理
    else{
        switch(op.s){
            case VOID: case CHAR:case INT:case FLOAT:case LONG:
def_var(op.s);break;
            case WHILE:case FOR: go_loop(op.s);break;
            case IF: go_if();break;
            case ELSE: error(IF_ELSE);break;    //else 没有 if，报错
            case BREAK:case CONTINUE:case RETURN: command();break;
            default: var(T,-1);        //函数调用或赋值语句处理
        }
    }
    return 1;
}
```

tree.h

```
void CreateNode(const char* t1,const char* t2){        //新结点
    T->child[T->sum]=(Tree)malloc(sizeof(Node));      //新结点
    strcpy(T->child[T->sum]->key1,t1);                //字符串写入
    strcpy(T->child[T->sum]->key2,t2);
    T->child[T->sum++]->sum=0;                //孩子个数初始化为 0
}

void InsertNode(Tree root,const char* str,int x){         //插入结点
    Tree p=(Tree)malloc(sizeof(Node));                //新结点
    p->sum=0;                                          //孩子个数初始化
    p->key1[0]=p->key2[0]='\0';
    if(x%2){
        strcpy(p->key1,str);                      //字符串写入
        if(x==3) strcpy(p->key2," ");
    }
    else strcpy(p->key2,str);
    p->child[p->sum++]=root->child[root->sum-1];          //新结点作为父亲节点
```

```
的孩子
    root->child[root->sum-1]=p;              //原孩子作为新结点的孩子
    T=p;
}
```

error.h

```
int punc(char x){    //标点报错
    if(s[i]!=x){
        if(x!=';') i--;
        printf(" %d    error: expected '%c'\n",l,x);
        return 0;
    }
    return 1;
}

int redeclare_judge(){
    for(int ii=0;ii<n;ii++)
        if(!strcmp(st[ii].name,t)){              //判断是否重复命名
            if(!st[ii].param_type){
                printf(" %d    error: redeclaration of '%s' with no linkage\n",l,t);
                return 0;
            }
        }
    return 1;
}

int var_judge(){
    for(k=0;k<n;k++)          //判断变量是否已声明过
        if(!strcmp(t,st[k].name))
            return k+1;
    return 0;
}

void name_judge(){                  //判断命名是否合法
    if(!(t[0]>='a'&&t[0]<='z'||t[0]>='A'&&t[0]<='Z'||t[0]=='_')){          //第一个字
符不能为数字
        printf(" %d    error: wrong name definiton\n",l);
        return;
    }
    if(key_judge()){              //是否为关键字
        printf(" %d    error: two or more data types in declaration specifiers\n",l);
        return;
    }
    for(int ii=0;ii<j;ii++)

if(!(t[ii]>='0'&&t[ii]<='9'||t[ii]>='a'&&t[ii]<='z'||t[ii]>='A'&&t[ii]<='Z'||t[ii]=='_')){
    //字符不合法
```

```
                printf(" %d      error: wrong name definiton\n",l);
                return;
            }
        redeclare_judge();
}

void declare_judge(){

if(!var_judge()&&!num_judge(j)&&!char_judge()&&strcmp(t,st[temp].param_name)
&&!key_judge())
            printf(" %d      error: %s undeclared\n",l,t);
        else if(num_judge(j)<0) printf(" %d      error: wrong number for '%s'\n",l,t);
        else  if(char_judge()==-1)  printf("  %d       warning:  multi-character  character
constant\n",l);
}

void macroname_judge(){
        for(int ii=0;t[ii];ii++){
            if((t[ii]<'A'||t[ii]>'Z')&&t[ii]!='_'){                //判断宏定义命名中是否有
不合法字符
                printf(" %d      error: wrong name defination\n",l);
                return;
            }
        }
        redeclare_judge();
}

void error(int x){   //其他部分报错
        switch(x){
            case COMPOUND: declare_judge();printf(" %d      error: expected ';' after
'%s'\n",l,t);break;
            case  CURVES:  printf("  %d       error:  expected  expression  before  ')'
token\n",l);break;
            case HEAD: printf(" %d      error: #include expects %dFILENAME%d or
<FILENAME>\n",l,34,34);break;
            case  FUN_FEW:  printf("  %d        error:  too  few  arguments  to
function\n",l);break;
            case  FUN_MANY:  printf("  %d        error:  too  many  arguments  to
function\n",l);break;
            case  FUN_VOID:printf("  %d        error:  return  statement  with  no
value\n",l);break;
            case FUN_NOTVOID:printf(" %d      error: return-statement with a value in
function returning 'void'\n",l);
            case  NO_EFFECT:  printf("  %d        warning:  statement  with  no
effect\n",l);break;
            case ARRAY:printf(" %d      error: incompatible types for '%s'\n",l,t);break;
            case  IDENTIFIER:printf("  %d        error:  '%s'  is  neither  array  nor
pointer\n",l,t);break;
            case  LOOP:printf("  %d        error:  statement  not  within  loop  or
switch\n",l);break;
```

```
            case FLAG:printf(" %d      error: statement not within function\n",l);break;
            case IF_ELSE:printf(" %d      error: 'else' without a previous 'if'\n",l);break;
            case PAREN:printf(" %d      error: expected ')'\n",l);break;
            case BRACKET:printf(" %d      error: expected ']'\n",l);break;
        }
}
```

traverse.h

```
void PreOrderTraverse(Tree T,int d){
        if(d){
            for(int ii=1;ii<d;ii++) printf("        ");          //缩进
            if(T->key1[0]) printf("%s ",T->key1);
            printf("%s\n",T->key2);
        }
        for(int ii=0;ii<T->sum;ii++)
            PreOrderTraverse(T->child[ii],d+1);          //遍历所有孩子
}
```

save.h

```
void indent(int d){          //缩进
        fputs("\n",fp);
        for(int i=1;i<d;i++)
            fputc('\t',fp);
}

void next(Tree T,int d){       //先序遍历所有孩子结点
        for(int i=0;i<T->sum;i++)
            save(T->child[i],d);
}

void InOrderTraverse(Tree T){            //中序遍历输出赋值语句或条件语句
        //printf("%s %s\n",T->key1,T->key2);
        if(T->sum){
            if(!strcmp(T->key1,"function call:")) next(T,2);    //函数调用
            else if(!strcmp(T->key1,"paren:")){          //括号
                fputc('(',fp);
                InOrderTraverse(T->child[0]);
                fputc(')',fp);
            }
            else if(!strcmp(T->key1,"subscript:")){ //方括号
                fputc('[',fp);
                InOrderTraverse(T->child[0]);
                fputc(']',fp);
            }
            else if(!strcmp(T->key1,"ID:")){       //数组
```

```
                fputs(T->key2,fp);
                InOrderTraverse(T->child[0]);
            }
            else{
                InOrderTraverse(T->child[0]);
                if(!strcmp(T->key2,"="))       //==
                    fputs("==",fp);
                else fputs(T->key2,fp);
                InOrderTraverse(T->child[1]);
            }
        }
        else if(strcmp(T->key2,"void")) fputs(T->key2,fp);    //空
}

void save(Tree T,int d){
        //printf("%s %s\n",T->key1,T->key2);
        if(!d) next(T,d+1);
        else if(!strcmp(T->key1,"line comment:"))     //行注释
            fprintf(fp,"\t\t//%s",T->key2);
        else if(!strcmp(T->key1,"block comment:"))    //块注释
            fprintf(fp,"\n/*%s*/",T->key2);
        else if(!strcmp(T->key1,"size:")){           //数组大小
            fprintf(fp,"[%s]",T->key2);
            next(T,d);
        }
        else if(!strcmp(T->key1,"condition:"))    //条件语句
            InOrderTraverse(T->child[0]);
        else if(!strcmp(T->key1,"compound")){     //复合语句
            fputc('{',fp);
            next(T,d+1);
            indent(d);
            fputc('}',fp);
        }
        else if(!strcmp(T->key1,"expression"))          //单个语句
            next(T,d+1);
        else if(!strcmp(T->key1,"param:")){         //函数形参
            fputc('(',fp);
            if(T->sum==1) InOrderTraverse(T->child[0]);         //调用
            else next(T,d);         //定义
            fputc(')',fp);
        }
        else
if(!strcmp(T->key1,"if")||!strcmp(T->key1,"if-else")||!strcmp(T->key1,"if-else-if")){
        //if 语句
            int tmp=0,fl=0;                 //用于记录上一个 else-if 语句的序号
            for(int i=0;i<T->sum;i++){
                if(!strcmp(T->child[i]->key1,"condition:")){       //条件语句
                    indent(d);
```

```
                    if(i) fputs("else ",fp);       //else-if 语句
                    fputs("if(",fp);
                    save(T->child[i],d);
                    fputc(')',fp);
                }
                else if(!strcmp(T->child[i]->key2,"sentence:")){
                    if(tmp==i-1&&tmp&&fl){          //else 语句
                        indent(d);
                        fputs("else",fp);
                    }
                    tmp=i;
                    fl=1;
                    save(T->child[i],d);
                }
                else{           //有注释的情况
                    tmp++;
                    save(T->child[i],d);
                }
            }
        }
        else if(!strcmp(T->key1,"assign:")){       //赋值语句
            indent(d);
            if(!strcmp(T->key2," ")){
                T=T->child[1];
                fputs(T->child[0]->key2,fp);
                fprintf(fp,"%s=",T->key2);
                fputs(T->child[1]->key2,fp);
            }
            else{
                InOrderTraverse(T->child[0]);
                fputc('=',fp);
                InOrderTraverse(T->child[1]);
            }
            fputc(';',fp);
        }
        else if(!strcmp(T->key1,"function call:")){     //函数调用
            indent(d);
            next(T,d);
            fputc(';',fp);
        }
        else
if(!strcmp(T->key1,"return")||!strcmp(T->key1,"continue")||!strcmp(T->key1,"break")
){
            indent(d);
            fputs(T->key1,fp);
            if(T->sum){
                fputc(' ',fp);
                InOrderTraverse(T->child[0]);
            }
```

```
            fputc(';',fp);
        }
        else if(!strcmp(T->key2,"declaration:")){     //声明语句
            indent(d);
            if(!strcmp(T->key1,"file"))         //文件包含
                fprintf(fp,"#include<%s>",T->child[0]->key2);
            else if(!strcmp(T->key1,"macro")){       //宏定义
                fputs("#define ",fp);
                next(T,d);
            }
            else{
                next(T,d);
                if(strcmp(T->child[T->sum-1]->key1,"compound"))         //函数定义
除外
                    fputc(';',fp);
            }
        }
        else if(!strcmp(T->key2,"sentence:")){    //循环语句
            indent(d);
            fprintf(fp,"%s(",T->key1);
            if(!strcmp(T->key1,"for")){         //for
                for(int i=0;i<3;i++){
                    if(strcmp(T->child[i]->key2,"null")){
                        if(!strcmp(T->child[i]->key1,"assign:")){     //赋值
                            InOrderTraverse(T->child[i]->child[0]);
                            fputc('=',fp);
                            InOrderTraverse(T->child[i]->child[1]);
                        }
                        else if(!T->child[i]->key1[0])   //无意义语句
                            InOrderTraverse(T->child[i]);
                        else save(T->child[i],d);     //条件语句
                    }
                    if(i<2) fputc(';',fp);
                    else fputc(')',fp);
                }
                if(T->sum>3)       //有注释
                    for(int ii=3;ii<T->sum;ii++)
                        save(T->child[ii],d);
            }
            else if(!strcmp(T->key1,"while")){    //while
                for(int i=0;i<T->sum;i++)
                    if(!strcmp(T->child[i]->key1,"condition:")){
                        InOrderTraverse(T->child[i]->child[0]);         //条件语句
                        fputc(')',fp);
                    }
                    else save(T->child[i],d);
            }
        }
        else{
```

```
            if(!strcmp(T->key2,"string")) fputs("char",fp);   //字符串定义
            else if(strstr(T->key2," array"))              //数组定义
                  for(int ii=0;T->key2[ii]!=' ';ii++)
                        fputc(T->key2[ii],fp);
            else fputs(T->key2,fp);
            if(strcmp(T->key1,"name:")) fputc(' ',fp);   //名字后面不需要空格
            next(T,d);   //遍历孩子们
      }
}
```