编译原理实验报告

实验二 语法分析

学号：202011998088　　姓名：臧祝利　　日期：2022年10月15日

实验要求

基于实验一，编写一个程序对使用 C--语言书写的源代码进行语法分析，并打印语法树。

基本要求

- 对程序进行语法分析，输出语法分析结果；
- 能够识别多个位置的语法错误

附加要求

- 按照正确的缩进格式输出带有行号信息的语法树
- 能够处理简单的语法错误

错误类型

总共写出16个错误规约式；

大致分类为

①缺少分号

②缺少右括号

③语法错误

实验分工

小组成员：臧祝利、陈金利、孙泽林、赵建锟

分工如下：

> 臧祝利：语法树+部分错误+思考题5
>
> 陈金利：syntax.y文件框架+部分错误+思考题2,4
>
> 孙泽林：cf.l文件的修改+部分错误+思考题1
>
> 赵建锟：Debug+部分错误+思考题3

实验环境

- Linux：Ubuntu 20.04 LTS
- Flex： V2.6.4
- GCC： V9.3.0
- Bison：V3.5.1

实验设计

语法树

tree.h

```
#ifndef TREE_H
#define TREE_H

#include <string.h>
#include <stdio.h>
#include <stdarg.h>
#include <stdlib.h>

typedef int NODE_TYPE;  //结点类型
#define NON_TERMINATOR 0  //非终结符
#define INT_TYPE 1  //int型
#define FLOAT_TYPE 2  //浮点型
#define STRING_TYPE 3  //字符型

typedef struct node{
    struct node *child;  //孩子
    struct node *sibling;  //兄弟
    int linenum;  //行号
    char *name;  //规约式名称
    NODE_TYPE type;  //类型
    union{
        char *id;  //识别到的内容
        int intvalue;  //识别到的整数
        float floatvalue;  //识别到的浮点数
    };
}Node;

Node *root;  //根节点
Node *CreateNode(char *name,int linenum,NODE_TYPE type);  //创建一个新结点
Node *InsertNode(Node *child,char *name,int linenum,NODE_TYPE type);  //建树
void AddSibling(Node *first,int num, ... );  //连接兄弟
void PrintNode(Node* node,FILE *f);  //打印结点
void TravelTree(Node *root,int dep,FILE *f);  //遍历输出

#endif
```

tree.c

```
#include <malloc.h>
#include "tree.h"

Node *CreateNode(char *name,int linenum,NODE_TYPE type){
    Node *node = (Node *)malloc(sizeof(Node));  //申请空间
    node->child = NULL;  //赋值
    node->sibling=NULL;
    node->name = strdup(name);
    node->linenum = linenum;
    node->type = type;
    node->intvalue = 1;  //默认操作
    return node;
}
Node *InsertNode(Node *child,char *name,int linenum,NODE_TYPE type){
    Node *father = (Node *)malloc(sizeof(Node));  //父节点
    father->child = child;  //链接孩子
    father->sibling = NULL;  //赋值
```

```
    father->name = strdup(name);
    father->linenum = linenum;
    father->type = type;
    father->intvalue = 1;
    root = father;  //自下而上建树
    return father;
}
void AddSibling(Node *first,int num, ... ){  //添加兄弟
    va_list valist;
    va_start(valist,num);
    int i;
    Node *cur = first;  //链表，挨个添加兄弟
    for (i=0;i<num;i++){
        Node *curNode = va_arg(valist,Node *);
        cur->sibling = curNode;
        cur = cur->sibling;
    }
    va_end(valist);
}
void PrintNode(Node* node,FILE *f){  //输出结点内容
    if (node->type == STRING_TYPE){
        fprintf(f,"%s : %s\n",node->name,node->id);
    }
    else if (node->type == FLOAT_TYPE){
        fprintf(f,"FLOAT: %f\n",node->floatvalue);
    }
    else if (node->type == INT_TYPE){
        fprintf(f,"INT: %d\n",node->intvalue);
    }
    else{
        fprintf(f,"%s (%d)\n",node->name,node->linenum);
    }
}
void TravelTree(Node *root,int dep,FILE *f){  //遍历树
    if (root == NULL) return;
    int i;
    for (i=0;i<dep;i++){
        fprintf(f,"\t");
    }
    PrintNode(root,f);
    TravelTree(root->child,dep+1,f);  //递归输出
    TravelTree(root->sibling,dep,f);
}
```

错误

①解决全局变量最后缺少分号的问题；

```
ExtDef :
    ...
    | Specifier error{
            char msg[100];
        sprintf(msg, "Missing ';' ");
        printerror(msg);
    }
    ;
```

②解决结构体内部定义出现的问题

```
StructSpecifier :
    ...
    | STRUCT OptTag LC error RC {
        char msg[110];
        sprintf(msg,"Syntax error");
        printerror(msg);
    }
    ;
```

添加中括号内可以使用INT类型来进行索引；

③解决中括号内部类型不对的问题，比如使用float或其他无法被规约的内容；

④解决缺少右括号的问题；

```
VarDec :
    ...
    | VarDec LB ID RB {
        $$ = InsertNode($1, "VarDec",@1.first_line, NON_TERMINATOR);
        AddSibling($1,3,$2,$3,$4);
    }
    | VarDec LB error RB {
            char msg[100];
        sprintf(msg, "Syntax error.");
        printerror(msg);
    }
    | VarDec LB ID error{
        char msg[100];
        sprintf(msg, "Missing \"]\".");
        printerror(msg);
    }
    ;
```

实验结果

基本要求

`test.cmm` 内容如下：

```
int main()
{
    else i = 4;
    int K3AB_gH;
    i = ;
    return _k1a;
}
```

结果输出如下：

```
Error type B at line 3: Syntax error.
Error type B at line 5: Syntax error.
```

`test1.cmm`

```c
int main()
{
    int i = 4;
    int K3AB_gH;
    i = 6;
    return _k1a;
}
```

无输出

`test2.cmm`

内容:

```c
int main(){
    int i = 3;
    %666
    if(i - 2 == 1){   // note
        i = i + 8
    }
    float p[i = 2.52;
}
```

结果为

```
Error Type A at line 3, char 5: Mysterious character: '%'
Error type B at Line 3: error: Missing ";"
Error type B at Line 5: error: Missing ";"
Error type B at Line 7: Missing "]".
Error type B at Line 7: Syntax error.
```

`test3` 内容为

```c
float f, m[6];

struct ms{
    int kk;
} ms;

int main(){
    int q[10];
    i = 3;
    ms.kk = i && fun();
}

int fun(int a, float f[2]){
    if(i - 2 == 1){
    q[1] = i + ms.kk;
    }
}
```

输出的语法树内容与要求一致，内容太长，这里忽略，详细见output.txt；

附加要求

`test1.2.cmm`

```
int main()
{
  float a[10][2];
  int i;
  a[5,3] = 1.5;
  if (a[1][2] == 0) i = 1 else i = 0;
}
```

结果如下：

```
Error type B at Line 5: Syntax error.
Error type B at Line 6: error: Missing ";"
```

`test1.6.cmm`

```
int main()
{
  int i = 09;
  int j = 0x3G;
}
```

结果如下：

```
Error Type A at line 3, char 11: Illegal float number: '09'
Error Type A at line 4, char 11: Illegal float number: '0x3G'
```

`test1.8.cmm`

```
int main()
{
  float i = 1.05e;
}
```

结果如下：

```
Error Type A at line 3, char 13: Illegal float number: '1.05e'
```

自加要求

其他错误，文件内容如下：

`tests.cmm`

```
int a;
struct node{
   int i=;
};
int main(){
    int i[1.54] = 3;
    int a,b=+;
    int i=;
    while(i=){i=i+1;}
    a[5,2]=4;
    b[1.24]=4;
    fun(a,b=);
    return 0
}
```

输出如下：

```
Error type B at Line 3: Syntax error.
Error type B at Line 6: Syntax error.
Error type B at Line 7: Syntax error.
Error type B at Line 7: Syntax error.
Error type B at Line 8: Syntax error.
Error type B at Line 9: Syntax error.
Error type B at Line 10: Syntax error.
Error type B at Line 12: Syntax error.
Error type B at Line 13: Missing ';' .
```

实验反思

思考题5

> 如何设计打印语法树的数据结构?

使用 多叉树链表+孩子-兄弟表示法 ;

定义如下:

```
typedef struct node {
    struct node *child;
    struct node *sibling;
    int linenum;
    char *name;
    NODE_TYPE type;
    union {
        char *id;
        int intvalue;
        float floatvalue;
    };
}Node;
```

其中 `child` 是孩子结点， `sibling` 是父亲结点， `linenum` 是结点的所在行号， `name` 为其规约的名字， `type` 为结点类型，用来判断如何输出结果;

由于一个结点只有一个属性，因此使用一个联合体来点名其结点属性，即是字符型或整型或浮点型;