《编译原理》专题2设计

目标任务

实验项目

完成以下描述赋值语句的LL(1)文法的递归下降分析程序

```
G[S]:
S→V=E
E→TE’
E’→ATE’\|ε
T→FT’
T’→MFT’\|ε
F→ (E)\|i
A→+\|-
M→\*\|/
V→i
```

设计说明

终结符号i为用户定义的简单变量，即标识符的定义。

设计要求

1. 输入串应是词法分析的输出二元式序列，即某算术表达式“专题1”的输出结果，输出为输入串是否为该文法定义的算术表达式的判断结果；
2. 递归下降分析程序应能发现简单的语法错误；
3. 设计两个测试用例（尽可能完备，正确和出错），并给出测试结果；
4. 选做：如有可能，考虑如何用文法描述C语言的if语句，使整个文法仍然为LL1文法，并使得你的递归下降程序可以分析赋值语句和if语句。

程序功能描述

1. 解析 LL(1) 文法
2. 输入, 解析一个二元式数组文件
3. 根据LL(1)文法识别,分析二元式文件并输出结果

数据结构

```
extern int current; //当前要分析的字符
extern int istrue;  //判断输入的表达式是否为该文法定义的算数表达式，0代表是，1代表不是
int FE(FILE *fp);   //E的函数
int FEp(FILE *FP);  //E'的函数
int FT(FILE *fp);   //T的函数
int FTp(FILE *fp);  //T'的函数
int FM(FILE *fp);   //M的函数
int FA(FILE *fp);   //A的函数
int FF(FILE *fp);   //F的函数
```

程序结构描述

主函数

处理输入输出, 调用FE函数开始递归下降分析

```
FILE *fp;
    if((fp = fopen("src.txt","r")) == NULL)
    {
        printf("打开文件失败\n");
        exit(-1);
    } else{
        freopen("output.txt", "w", stdout);
        current = fgetc(fp);
        FE(fp);
        if(istrue == 0)
            printf("合法的表达式\n");
        else
            printf("不合法的表达式\n");
    }
    fclose(fp);
    fclose(stdout);
    return 0;
```

FE函数

根据当前分析栈的内容和下个输入字符转移状态. 其余函数同理. 故不再一一说明

```
int FE(FILE *fp)
{
    int t = 0, ep = 0;

    if (current == 'i' || current == '(')
    {
        //current = fgetc(fp);
```

```
        t = FT(fp);
        if( t == 0)
        {
            //current = fgetc(fp);
            ep = FEp(fp);
            if(ep == 0)
            {
                printf("E -> TE'\n");
                istrue = 0;
            } else
            {
                istrue = 1;
            }
        } else{
            istrue = 1;
        }
    } else{
        istrue = 1;
    }
    //current = fgetc(fp);
    if(istrue == 0)
        return 0;
    else
        return 1;
}
```

测试

测试用例输入

```
i*(i+i)#
i*(i+i#
```

测试用例输出

```
F -> i
M -> *
F -> i
T' -> ε
T -> FT'
A -> +
F -> i
T' -> ε
T -> FT'
E' ->ATE'
E' -> ε
```

```
E -> TE'
F -> (E)
T' -> MFT'
T' -> ε
T -> FT'
E' -> ε
E -> TE'
合法的表达式

F -> i
M -> *
F -> i
T' -> ε
T -> FT'
A -> +
F -> i
T' -> ε
T -> FT'
E' ->ATE'
E' -> ε
E -> TE'
不合法的表达式
```

源代码

```c
#include <stdio.h>
#include <stdlib.h>
#include "parser.h"

int istrue = 0; //0代表正确，1代表错误
int current;

int main() {
    FILE *fp;
    if((fp = fopen("src.txt","r")) == NULL)
    {
        printf("打开文件失败\n");
        exit(-1);
    } else{
        freopen("output.txt", "w", stdout);
        current = fgetc(fp);
        FE(fp);
        if(istrue == 0)
            printf("合法的表达式\n");
        else
            printf("不合法的表达式\n");
    }
    fclose(fp);
```

```
    fclose(stdout);
    return 0;
}

int FE(FILE *fp)
{
    int t = 0, ep = 0;

    if (current == 'i' || current == '(')
    {
        //current = fgetc(fp);
        t = FT(fp);
        if( t == 0)
        {
            //current = fgetc(fp);
            ep = FEp(fp);
            if(ep == 0)
            {
                printf("E -> TE'\n");
                istrue = 0;
            } else
            {
                istrue = 1;
            }
        } else{
            istrue = 1;
        }
    } else{
        istrue = 1;
    }
    //current = fgetc(fp);
    if(istrue == 0)
        return 0;
    else
        return 1;
}

int FEp(FILE *fp)
{
    int a = 0, t = 0;
    istrue = 0;
    while (1)
    {
        if(current == '+' || current == '-')
        {
            //current = fgetc(fp);
            a = FA(fp);
            if(a == 0)
            {
```

```
                t = FT(fp);
                if(t == 0)
                {
                    printf("E' ->ATE'\n");
                    //current = fgetc(fp);
                } else{
                    istrue = 1;
                    break;
                }
            } else{
                istrue = 1;
                break;
            }
        } else{
            if (current == ')' || current == '#')
            {
                printf("E' -> ε\n");
                istrue = 0;
                break;
            } else{
                istrue = 1;
                break;
            }
        }
    }
    //current = fgetc(fp);
    if(istrue == 0)
        return 0;
    else
        return 1;
}

int FT(FILE *fp)
{
    int f = 0,tp = 0;
    istrue = 0;
    if(current == 'i' || current == '(')
    {
        //current = fgetc(fp);
        f = FF(fp);
        if(f == 0)
        {
            //current = fgetc(fp);
            tp = FTp(fp);
            if(tp == 0)
            {
                printf("T -> FT'\n");
                istrue = 0;
            } else{
```

```
                istrue = 1;
            }
        } else{
            istrue = 1;
        }
    } else{
        istrue = 1;
    }
    //current = fgetc(fp);
    if(istrue == 0)
        return 0;
    else
        return 1;
}

int FTp(FILE *fp)
{
    int m = 0, f = 0;
    istrue = 0;
    while (1)
    {
        if(current == '*' || current == '/')
        {
            //current = fgetc(fp);
            m = FM(fp);
            if(m == 0)
            {
                //current = fgetc(fp);
                f = FF(fp);
                if(f == 0)
                {
                    printf("T' -> MFT'\n");
                    //current = fgetc(fp);
                } else{
                    istrue = 1;
                    break;
                }
            } else{
                istrue = 1;
                break;
            }
        } else{
            if (current == ')' || current == '#' || current == '+' ||
current == '-')
            {
                printf("T' -> ε\n");
                istrue = 0;
                break;
            } else{
```

```
                istrue = 1;
                break;
            }
        }
    }
    //current = fgetc(fp);
    if(istrue == 0)
        return 0;
    else
        return 1;
}

int FM(FILE *fp)
{
    istrue = 0;
    if(current == '*')
    {
        printf("M -> *\n");
        istrue = 0;
    } else if (current == '/'){
        printf("M -> /\n");
        istrue = 0;
    } else{
        istrue = 1;
    }
    current = fgetc(fp);
    if(istrue == 0)
        return 0;
    else
        return 1;
}

int FA(FILE *fp)
{
    istrue = 0;
    if(current == '+')
    {
        printf("A -> +\n");
        istrue = 0;
    } else if(current == '-'){
        printf("A -> -\n");
        istrue = 0;
    } else{
        istrue = 1;
    }
    current = fgetc(fp);
    if(istrue == 0)
        return 0;
    else
```

```
        return 1;
}

int FF(FILE *fp)
{
    int e = 0;
    istrue = 0;
    if(current == '(')
    {
        current = fgetc(fp);
        e = FE(fp);
        if(e == 0)
        {
            if(current == ')')
            {
                current = fgetc(fp);
                printf("F -> (E)\n");
                istrue = 0;
            } else{
                istrue = 1;
            }
        } else{
            istrue = 1;
        }
    } else{
        if(current == 'i')
        {
            printf("F -> i\n");
            current = fgetc(fp);
            istrue = 0;
        }
        else
            istrue = 1;
    }
    if(istrue == 0)
        return 0;
    else
        return 1;
}
```