

编译原理上机实验报告

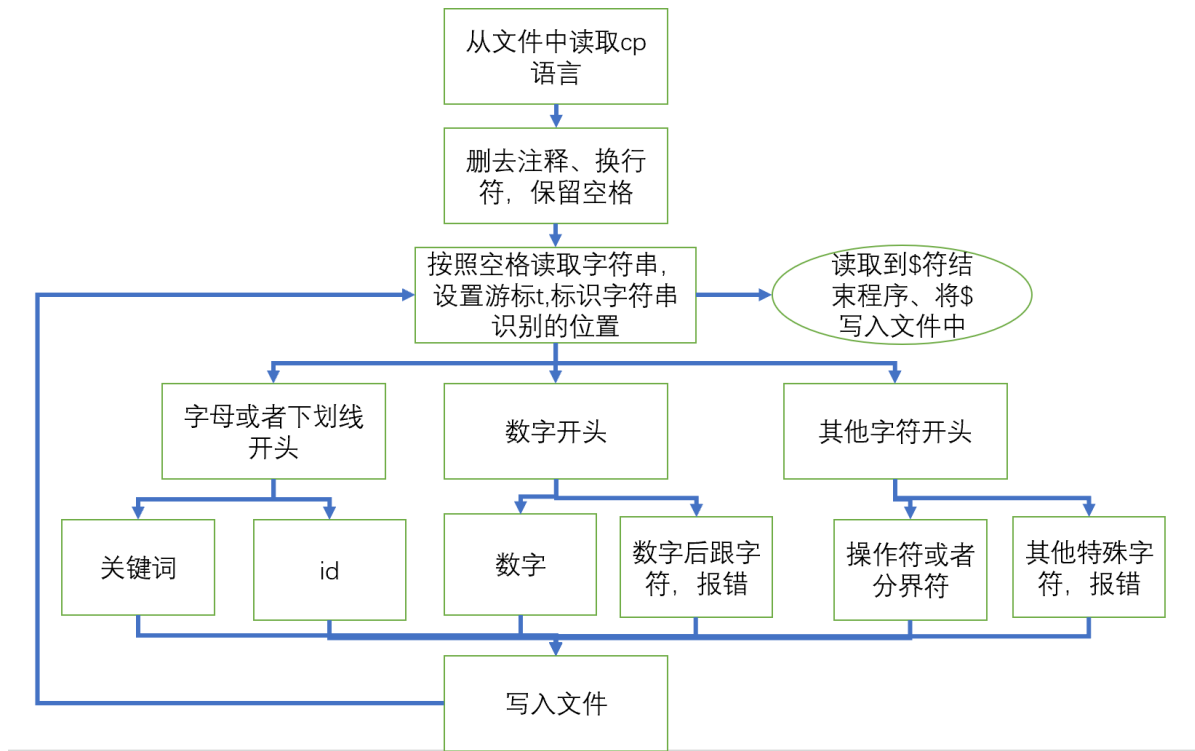
盛琼怡 17041802

目录

编译原理上机实验报告	0
专题一 词法分析程序设计	2
1、程序框图	2
2、主要数据结构和函数设计	2
3、测试用例及结果	4
(1) 简单的 cp 语言	4
(2) 算术式子	5
(3) 特殊符号	6
(4) 数字开头后跟字母	6
(5) 关键词后跟数字	7
专题二 递归下降分析子程序	7
1、程序框图	7
2、主要数据结构和函数设计	8
3、测试用例及结果	9
(1) 算术表达式	9
(2) 存在错误的表达式 1	10
(3) 存在错误的表达式 2	11
专题三、七 LL(1)语法分析	12
1、程序框图	12
2、主要数据结构及函数设计	12
3、测试用例及结果	14
(1) 文法处理测试	14
(2) 正确的 LL1 语言测试	15
(3) 错误的 LL1 语言测试	18
学习总结	19
1、程序结构设计	19
2、个人总结	19

专题一 词法分析程序设计

1、程序框图



2、主要数据结构和函数设计

```
char key[12][6] =
{
    " ", "begin", "end", "if", "then", "else", "for", "while", "do", "and", "or", "not"
};
char opOrde[18][3] =
{
    " ", "+", "-", "*", "/", ">", "<", "=", ":", ">=", "<=", "<>", "++", "--", "(", ")", ";", "#"
};

//读入数据,进行预处理
int readFile()
{
    //判断是否是关键词
    int iskey(char *c)
    {
        //判断是否是操作符或分界符
        int isop(char *c)
        {
            //判断是字母、数字、下划线、还是特殊字符
            int isC(char c)
            {
            }
```

```

//主程序
int main()
{
    printf("%d\n", readfile());
    FILE *fread = fopen("pretreat.txt", "rt");
    FILE *outfile = fopen("C:\\Users\\shengqiong\\Desktop\\test\\final.txt", "w");
    char txt[10000], op[3], c[100], num[32], id[100];
    fgets(txt, 10000, fread);
    int i=0;
    int j;
    int t, f;
    int u=0, OP; //记录数字或id长度
    while(txt[i]!='$') //读取数据
    {
        j=0;
        if(txt[i]==' ')
            i++;
        while(txt[i]!=' ')
        {
            c[j]='\0'; //按照空格获取字符串
            printf("%s\n", c);
            t=0; //游标
            f=isC(c[0]);
            //小写字母或者下划线开头
            if(f==0 || f==2)
            {
                //数字开头
                else if(f==1)
                {
                    //运算符开头
                }
                else
                {
                    i++;
                }
            }
            //写入结束符、关闭文件
            fprintf(outfile, "$");
            fclose(fread);
            fclose(outfile);
            return 0;
        }
    }
}

```

(详细代码及注释见源程序)

3、测试用例及结果

(1) 简单的 cp 语言

```
helloworld - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
#这里是注释
Dstates := 123;
p := 1; j := 1;
while j <= p do
#这里也是注释
(
    i := 0;
    if ei = 1 for i <= p
    then Dstates:=i
    else (p := p+1;
          Dstatesp := e;
          Dtrana := p;
    )
    j := j+1;
)
$


final - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
<30,Dstates>
<:=,-->
<29,123>
<,,-->
<30,p>
<:=,-->
<29,1>
<,,-->
<30,j>
<:=,-->
<29,1>
<,,-->
<while,-->
<30,j>
<<=,-->
<30,p>
<do,-->
<(,-->
<30,i>
<:=,-->
<29,0>
<,,-->
<if,-->
<30,ei>
<=,-->
<29,1>
```

```

<for,-->
<30,i>
<=<,-->
<30,p>
<then,-->
<30,Dstates>
<:=,-->
<30,i>
<else,-->
<(<,-->
<30,p>
<:=,-->
<30,p>
<+,-->
<29,1>
<;,-->
<30,Dstatesp>
<:=,-->
<30,e>
<;,-->
<30,Dtrana>
<:=,-->
<30,p>
.
<;,-->
<),-->
<30,j>
<:=,-->
<30,j>
<+,-->
<29,1>
<;,-->
<),-->
$

```

(2) 算数式子

 helloworld - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

(a + b) - c * d/e;

\$

final - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
<(,-->  
<30,a>  
<+,-->  
<30,b>  
<),-->  
<-,-->  
<30,c>  
<*,-->  
<30,d>  
</,-->  
<30,e>  
<;,-->  
$
```

(3) 特殊符号

helloworld - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
~@ "  
$
```

final - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
err:err op  
err: can not find this op  
$
```

(4) 数字开头后跟字母

helloworld - 记事本

文件(F) 编辑(E) 格式(O) 查看(V)

```
123tsads  
$
```

final - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

err: id can not follow with number

<29,123>

\$

(5) 关键词后跟数字

helloworld - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮

begin123

\$

final - 记事本

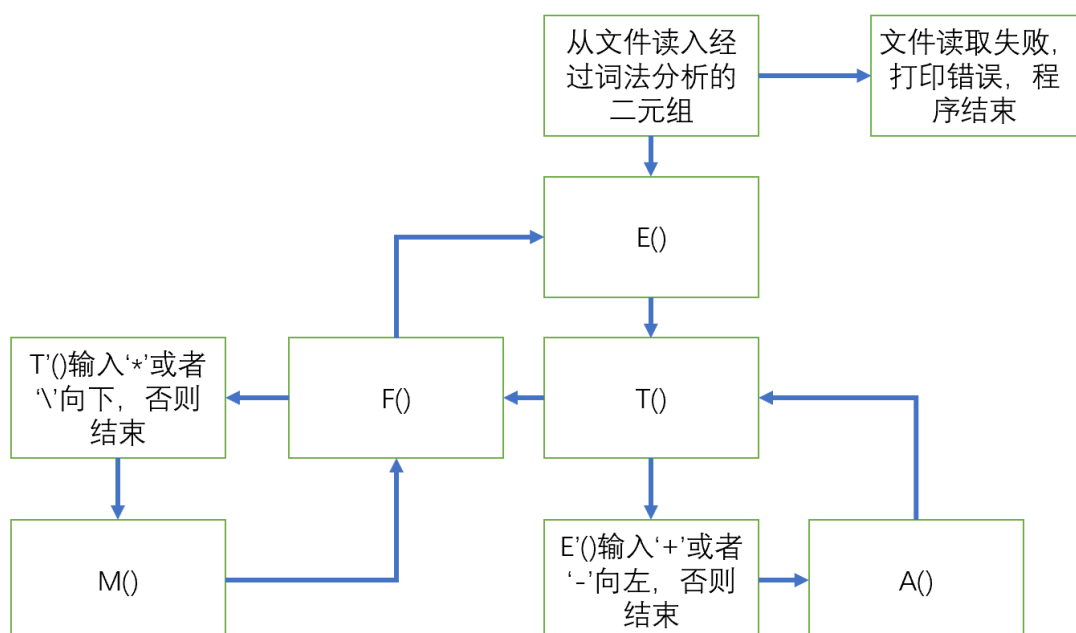
文件(F) 编辑(E) 格式(O) 查看(V)

<30,begin123>

\$

专题二 递归下降分析子程序

1、程序框图



2、主要数据结构和函数设计

```
#include<stdio>
char s[200][20];
int k=0,n=0;
void E();
//从文件读入二元组
int readFile()
{

//A->+||-
void A()
{

//M->*||/
void M()
{

//F->(E)||i
void F()
{

//Tp->MFTp
void Tp()
{

//T->FTp
void T()
{

//Ep->ATEp
void Ep()
{


//E->TE!
void E()
{

//主函数
int main()
{
```

(详细代码及注释见源程序)

3、测试用例及结果

(1) 算数表达式

 final - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
<(-->  
<30,a>  
<+,->  
<30,b>  
<),-->  
<-,->  
<30,c>  
<*,-->  
<30,d>  
</,->  
<30,e>  
<;,->  
$
```

```
F::i  
Tp::  
T::FTp  
A::+  
F::i  
Tp::  
T::FTp  
Ep::  
Ep::ATEp  
Ep::  
E::TEp  
F::(E)  
Tp::  
T::FTp  
A::-  
F::i  
M::*  
F::i  
M::/  
F::i  
Tp::  
Tp::MFTp  
Tp::  
Tp::MFTp  
Tp::  
T::FTp  
Ep::  
Ep::ATEp  
Ep::  
E::TEp
```

(2) 存在错误的表达式 1

缺少右括号

final - 记事本

文件(F) 编辑(E) 格式(O)

<(-->

<30,a>

<+-->

<30,b>

<--->

<30,c>

<*,-->

<30,d>

</-->

<30,e>

<;-->

\$

```
F::i
Tp::
T::FTp
A::+
F::i
Tp::
T::FTp
A::-
F::i
M::*
F::i
M::/
F::i
Tp::
Tp::MFTp
Tp::
Tp::MFTp
Tp::
T::FTp
Ep::
Ep::ATEp
Ep::
Ep::ATEp
Ep::
E::TEp
F error need )
Tp::
T::FTp
Ep::
E::TEp
```

(3) 存在错误的表达式 2

除号重复

final - 记事本

文件(F) 编辑(E) 格式(O) 查看(V)

<30,a>

<+ ,-->

<30,b>

<- ,-->

<30,c>

<* ,-->

<30,d>

</ ,-->

</ ,-->

<30,e>

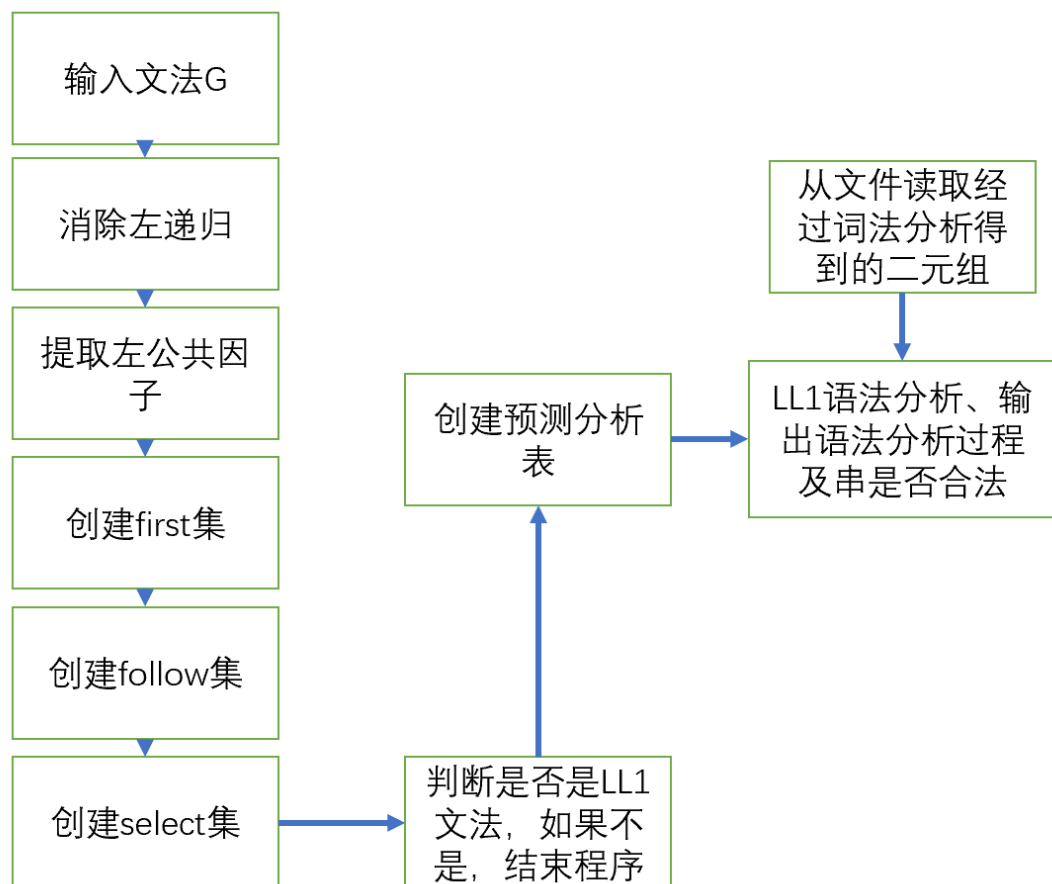
< ; ,-->

\$

```
F::i
Tp::
T::FTp
A::+
F::i
Tp::
T::FTp
A::-
F::i
M::*
F::i
M::/
F error
M::/
F::i
Tp::
Tp::MFTp
Tp::
Tp::MFTp
Tp::
Tp::MFTp
Tp::
T::FTp
Ep::
Ep::ATEp
Ep::
Ep::ATEp
Ep::
E::TEp
```

专题三、七 LL(1)语法分析

1、程序框图



2、主要数据结构及函数设计

```

const int maxcp = 100; //文法最大条数
const int maxf = 1000; //文法中包含的最大字符数量
int length=0; //文法长度
int lengths=0; //select集长度

//文法、First集、Follow集、select集存储结构
struct cp
{
//预测分析表存储结构
struct chart
{
//模拟堆栈
struct stacks
{

//函数提前声明
void print(cp *a);
bool isup(string s);
int replace_a();
int replace_b();
void print_select(cp *a);
void find_follow(int i,int tlen);
int find_select(int t3,string t2,int i,int j,int tlen);

//消除左递归
void remove_left_recursion()
{
//提取左公共因子
void extract_left_common_factor()
{
//创建First集
void create_first()
{
//替代First集中的非终结符
int replace_a()
{
//创建Follow集
void create_follow()
{
//替代Follow集中的非终结符
int replace_b()
{
//创建select集
void create_select()
{
//查找应该加入select集的元素,将其加入集合中
int find_select(int t3,string t2,int i,int j,int tlen)
{
//查找follow集
void find_follow(int i,int tlen)
{

```

```

//判断是否是LL1文法
bool judge()
{
//创建预测分析表
void create_chart()
{
//判断非终结符
bool isup(string s)
{
//打印出文法、First集、Follow集
void print(cp *a)
{
//打印出select集
void print_select(cp *a)
{
//打印出预测分析表
void print_chart(chart *a)
{
//主函数
int main()
{

```

本程序中使用‘~’代替‘ ’。

本程序存在的问题：输入的非终结符不能超出文法输入的范围，且 id 必须为单个字符，未作 id 解析。输入的文法不能包含空格。

(详细代码及注释见源程序)

3、测试用例及结果

(1) 文法处理测试

```

S->Aa|b
A->Ac|Sd|Acb
$

```

```

C:\Users\shengqiongyi\De
S->Aa b
A->Ac | Sd | Acb
$
消除左递归
S->Aa b
A->bdA~
A~->cA~ | adA~ | cbA~ | # |

提取左公共因子
S->Aa b
A->bdA~
A~->cA~~ | adA~ | # |
A~~->A~ | bA~ |

first集
S->b
A->b
A~->c | a | # |
A~~->c | b | a | # |

follow集
S->$
A->a
A~->a
A~~->a

select集
S->Aa b,
S->b b,
A->bdA~ b,
A~->cA~~ c,
A~->adA~ a,
A~-># a,
A~~->A~ b,
A~~->bA~ b,

false LL1

```

(2) 正确的 LL1 语言测试

输入文法

```

E->TE~
E~->ATE~|#
T->FT~
T~->MFT~|#
F->(E)|i
A->+|-
M->*/
$

```

输入句子 $i+i*i-i/i+(i-i)$ 经过词法分析的二元组

final - 记事本

文件(F) 编辑(E) 格式(O)

<30,i>

<+,->

<30,i>

<*,-->

<30,i>

<-,->

<30,i>

</,-->

<30,i>

<+,->

<(->

<30,i>

<-,->

<30,i>

<),-->

\$

E->TE[~]

E[~]->ATE[~]|#

T->FT[~]

T[~]->MFT[~]|#

F->(E)|i

A->+|-

M->*|/

\$

消除左递归

E->TE[~]

E[~]->ATE[~]|#

T->FT[~]

T[~]->MFT[~]|#

F->(E)|i

A->+|-

M->*|/

提取左公共因子

E->TE[~]

E[~]->ATE[~]|#

T->FT[~]

T[~]->MFT[~]|#

F->(E)|i

A->+|-

M->*|/

first集

E->(| i |

E[~]->+ | # | - |

T->(| i |

T[~]->* | # | / |

F->(| i |

A->+ | - |

M->* | / |

```

follow集
E->$ ) |
T->+ $ | - | ) |
E~->$ ) |
A->( | i |
F->* | + | / | $ | - | ) |
T~->+ $ | - | ) |
M->( | i |

select集
E->TE~ (, i,
E~->ATE~ +, -,
E~-># $, ),
T->FT~ (, i,
T~->MFT~ *, /,
T~-># +, $, -, ),
F->(E) (,
F->i i,
A->+ +,
A->- -,
M->* *,
M->/ /,

```

```

true LL1
预测分析表
( i
E E->TE~ E->TE~
+ - $ )
E~ E~->ATE~ E~->ATE~ E~-># E~->#
( i
T T->FT~ T->FT~
* / + $ - )
T~ T~->MFT~ T~->MFT~ T~-># T~-># T~-># T~->#
( i
F F->(E) F->i
+ -
A A->+ A->-
* /
M M->* M->/

```

```

语法分析
E->TE~
T->FT~
F->i
T~->#
E~->ATE~
A->+
T->FT~
F->i
T~->MFT~
M->*
F->i
T~->#
E~->ATE~
A->-
T->FT~
F->i
T~->MFT~
M->/
F->i
T~->#
E~->ATE~
A->+
T->FT~
F->(E)
E->TE~
T->FT~
F->i
T~->#
E~->ATE~
A->-
T->FT~
F->i
T~->#
E~->#
T~->#
E~->#
correct language

```

(3) 错误的 LL1 语言测试

输入文法

```

E->TE~
E~->ATE~|#
T->FT~
T~->MFT~|#
F->(E)|i
A->+|-
M->*/
$

```

输入句子 i**i- 经过词法分析的二元组

文件(F) 编辑(E)

<30,i>

<*,-->

<*,-->

<30,i>

<-,-->

\$

与上一个测试用例同使用算术表达式文法，结果一致。此处只记录语法分析部分的结果

```
语法分析
E->TE~
T->FT~
F->i
T~->MFT~
M->*
err language
```

学习总结

1、程序结构设计

所有程序公用 final.txt

专题一，对 CP 语言进行词法分析，将词法分析的结果存入 final.txt。

专题二，读取 final.txt 进行递归下降分析子程序。

专题三、七，在程序中输入文法 G，程序自动解析文法，如果是 LL1 文法，则从 final.txt 中读取 CP 语言分析后的二元组，进行语法分析。

2、个人总结

通过本次实验对词法分析、递归下降分析子程序、LL1 语法分析都有了更深刻的理解。在写专题一时，明白了代码结构的重要性，应该把每个部分独立处理，使

逻辑更加清醒。同时我们平时在写代码时也应该注意代码风格，合理利用空格。

在编写词法分析时我发现，如果写 id 和各种操作符时使用了空格，那么对后续的词法分析会非常的方便。专题三和七花费了我非常多的时间，让我意识到了数据结构的重要性，其次是 c++ 不太适合写这种需要频繁处理字符串的程序，要想通过一个字符找到它对应的产生式还是使用其他语言会方便许多，比如 JavaScript 或者 python。