湖南科技大学2017-2018第二学期

数据结构课程设计报告

**学 院** ： 计算机科学与工程

**专业班级：** 17级计算机科学与科学五班

**姓 名：** 刘雅琴

**学 号：** 1705010526

**指导教师：** 廖苗

**时 间**： 2018.6.25-2018.7.6

**地 点**： 逸夫楼414机房

|  |
| --- |
| 指导教师评语：  **成绩：**  **签名：**  **年 月 日** |

# 任务A 复杂度分析(Ⅰ)

## 一、任务描述

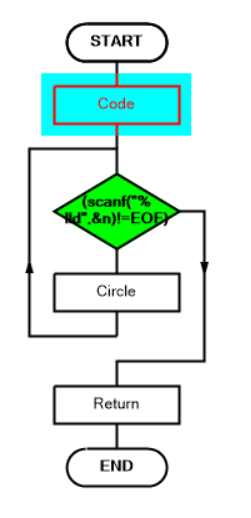
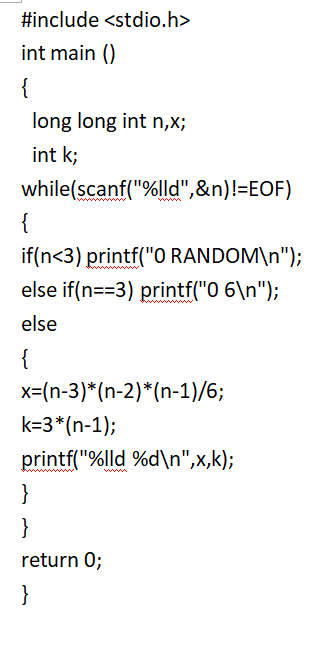
## 分析如下代码 for(i=1;i<n;i++) for(j=1;j<i;j++) for(k=1;k<j;k++) printf("\n"); 问printf语句共执行了几次？这段代码执行完以后i+j+k值为多少？

## 二、分析与设计

（1）通过计算一个较小的n值发现规律，得出公式共执行了(n-3)\*(n-2)\*(n-1)/6次，i+j+k最终的值为3\*(n-1)。

（2）数据对象为整型数，基本操作为数据之间的加减乘除。

（3）流程图

## 三、性能分析

内存：1092

耗时：0

语言：C

代码长度：296B

## 四、调试与测试数据

样例输入：

6

样例输出：

10 15

## 五、小结

注意题中取RANDOM的情况，i,j,k可能没有赋初值，相加后是随机值

# 任务B 复杂度分析(Ⅱ)

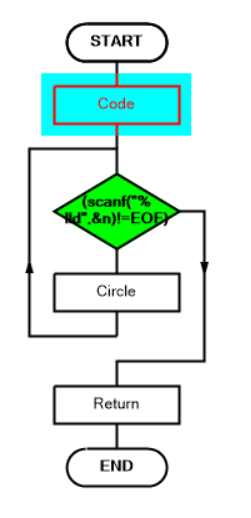
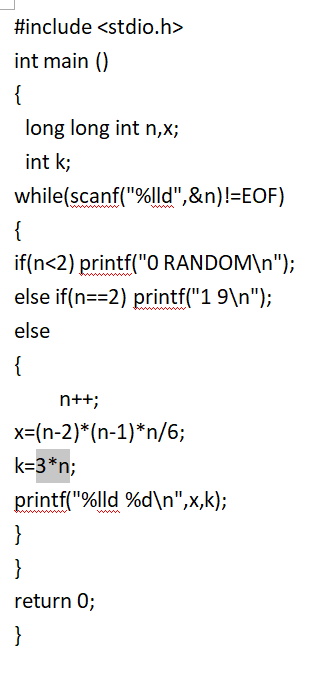
## 任务描述

有如下代码段(n为正整数)：  
i=1;  
while(i++<n){  
j=1;  
while(j++<i){  
k=1;  
while(k++<j)  
printf("\n");  
}  
}   
问printf语句共执行了几次？这段代码执行完以后i+j+k值为多少？

1. **分析与设计**
2. 此题与任务A相似，因为自加方式与上一题有些不同，只需要将公式稍微变动一下就行。得出公式共执行了(n-2)\*(n-1)\*n/6次，i+j+k最终的值为3\*n。

（2）数据对象为整型数，基本操作为数据之间的加减乘除。

（3）流程图

## **性能分析**

内存：1092

耗时：4

语言：C

代码长度：304B

1. **调试与测试数据**

样例输入

3

样例输出

4 12

## **五、小结**

注意题中取RANDOM的情况。

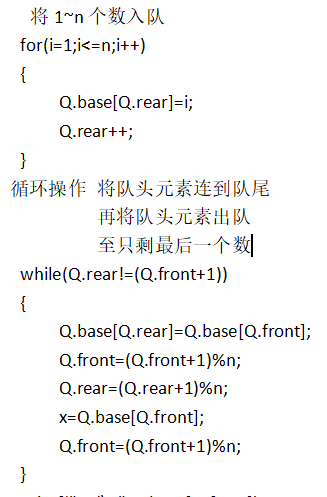
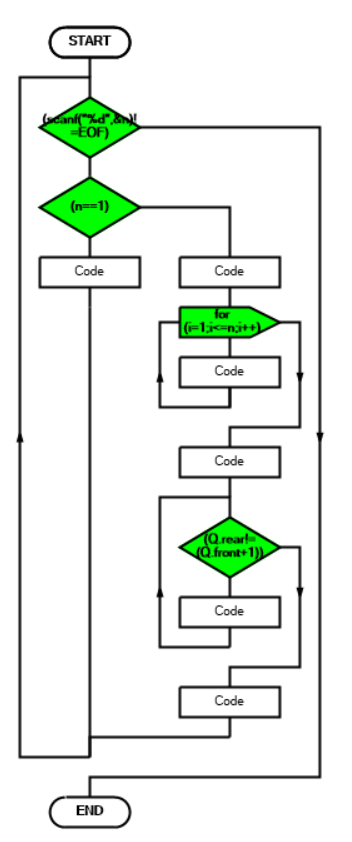
# 任务C Josephus问题(Ⅰ)

## **任务描述**

n个人排成一圈，按顺时针方向依次编号1，2，3…n。从编号为1的人开始顺时针"一二"报数，报到2的人退出圈子。这样不断循环下去，圈子里的人将不断减少。最终一定会剩下一个人。试问最后剩下的人的编号。

要求程序模拟题意来实现。

1. **分析与设计**
2. 思路：根据提示中的出队顺序这一信息，这题可以用队列的删除与插入来模拟报数人数的减少。即首先1~n个数入队，然后循环操作1、将队头元素连到队尾2、将队头元素出队。直至队列中剩下一个数。
3. 数据结构及算法：数据对象为整型，通过队列的结构了解，并进行删除、插入操作。
4. 流程图



## **性能分析**

内存：1092

耗时：24

语言：C

代码长度：785B

1. **调试与测试数据**

样例输入

7

2

样例输出

7

1

## **小结**

在进行队列插入元素操作时要进行Q.front=(Q.front+1)%n的操作防止溢出。

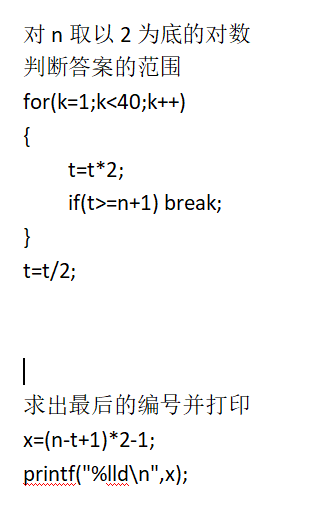
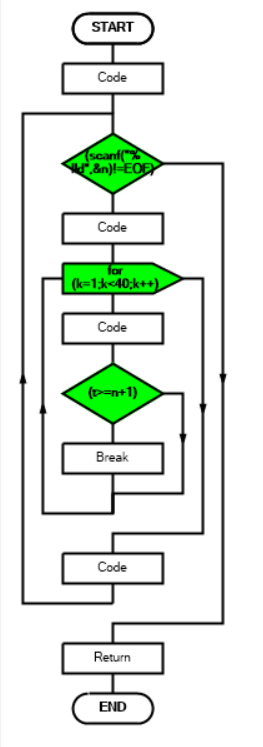
# 任务D Josephus问题(Ⅱ)

## 一、任务描述

n个人排成一圈，按顺时针方向依次编号1，2，3…n。从编号为1的人开始顺时针"一二"报数，报到2的人退出圈子。这样不断循环下去，圈子里的人将不断减少。最终一定会剩下一个人。试问最后剩下的人的编号。

n很大，直接模拟题意不行，请找出规律。

1. **分析与设计**
2. 思路：本题与上题的不同之处在于n的非常大，只能找规律求解。先写出n为1~16的答案，然后再找规律。
3. 数据元素为整型，可以通过对n取以2为底的对数先判断答案大致在哪个范围，再根据公式求解。
4. 流程图



## **性能分析**

内存：1092

耗时：0

语言：C

代码长度：341B

1. **调试与测试数据**

样例输入

7

2

样例输出

7

1

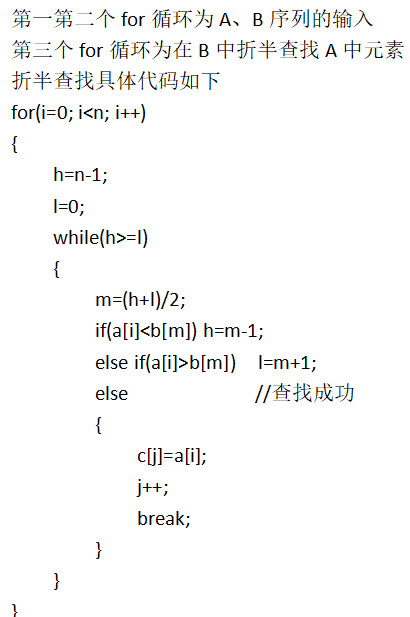
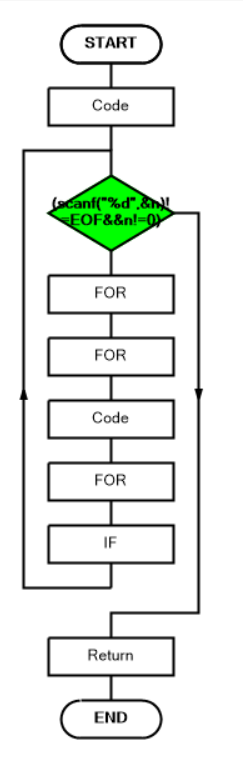
## **小结**

找规律时注意答案与2的关系，例如2的倍数、2的倍数+1等。

# 任务G 交集

## 任务描述

有两个相等长度的正整数序列A和B，都是有序的(递增排序)，同时一个序列中没有重复元素，现在需要求这两个序列的交――序列C，同时打印输出。

1. **分析与设计**
2. 思路：1、用两个数组存储待检查序列A和B，再另外开辟一个数组C存储交序列。
3. 遍历A中元素，并依次在B中折半查找，查找成功，遍历下一元素，查找失败删除改元素
4. 数据元素是整型，基本操作为对数据进行比较、对数组的折半查找及数组中删除元
5. 

## **性能分析**

内存：1556

耗时：132

语言：C

代码长度：1123B

1. **调试与测试数据**

样例输入

5

1

2

5

6

7

1

2

4

6

9

0

样例输出

3 1 2 6

## **小结**

一开始的想法是用链表存储，但是链表移动没有数组方便，之后就决定用数组存储。

**任务H 大爱线性表**

## 任务描述

不少参赛同学刚学数据结构，对线性表最是熟悉不过。这里我们给线性表增加两个特殊的操作，第一个是‘R’ 操作，表示逆转整个表，如果表长为L，原来的第i个元素变成第L-i+1个元素。第二个操作是‘D’，表示删除表的第一个元素，如果表为空，则返回一个“error”信息。我们可以给出一系列的‘R’ 和‘D’组合，例如“RDD”表示先逆转表，然后删除最前面的两个元素。

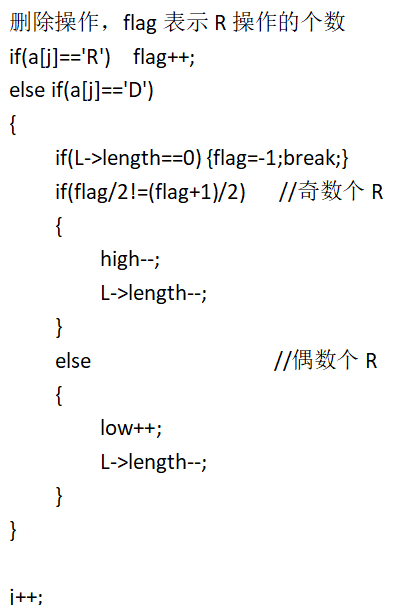
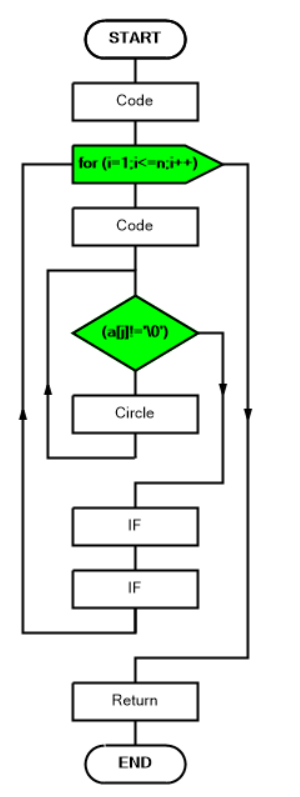
本题的任务是给定表和一个操作串S，求出执行S后的表，如果中途出现‘D’操作于空表，输出“error”。

1. **分析与设计**
2. 思路：1、考虑到逆转操作次数过多可能会引起时间超限，将逆转操作进行改进，即根据逆转的次数来判断最后的子序列是否逆转。

2、用low和high标记该序列最首和最末位置，在D操作前如果有奇数个R则删除最末元素（high--）；如果有偶数个R则删除最首元素（low++）。

（2）数据元素为整型，存储在线性表结构中，进行删除操作。

（3）流程图（主函数）



以上代码是流程图中Circle部分

之后的两个if部分为输出序列部分：最后如果flag是奇数，从high位置到low的顺序输出元素。如果flag是偶数，从low位置到high位置输出元素。

## **性能分析**

内存：1484

耗时：188

语言：C

代码长度：1874B

1. **调试与测试数据**

样例输入

4

RDD

4

[1,2,3,4]

DD

1

[42]

RRD

6

[1,1,2,3,5,8]

D

0

[]

样例输出

[2,1]

error

[1,2,3,5,8]

error

## **小结**

本来是用的线性表将逆转操作一次次进行，每次删除都删除最首元素，之后时间超限，才考虑到优化算法。

# 任务I 单词检查(Ⅰ)- 顺序表实现

## 任务描述

许多应用程序，如字处理软件，邮件客户端等，都包含了单词检查特性。单词检查是根据字典，找出输入文本中拼错的单词，我们认为凡是不出现在字典中的单词都是错误单词。不仅如此，一些检查程序还能给出类似拼错单词的修改建议单词。 例如字典由下面几个单词组成：

bake cake main rain vase

如果输入文件中有词vake ，检查程序就能发现其是一个错误的单词，并且给出 bake, cake或vase做为修改建议单词。

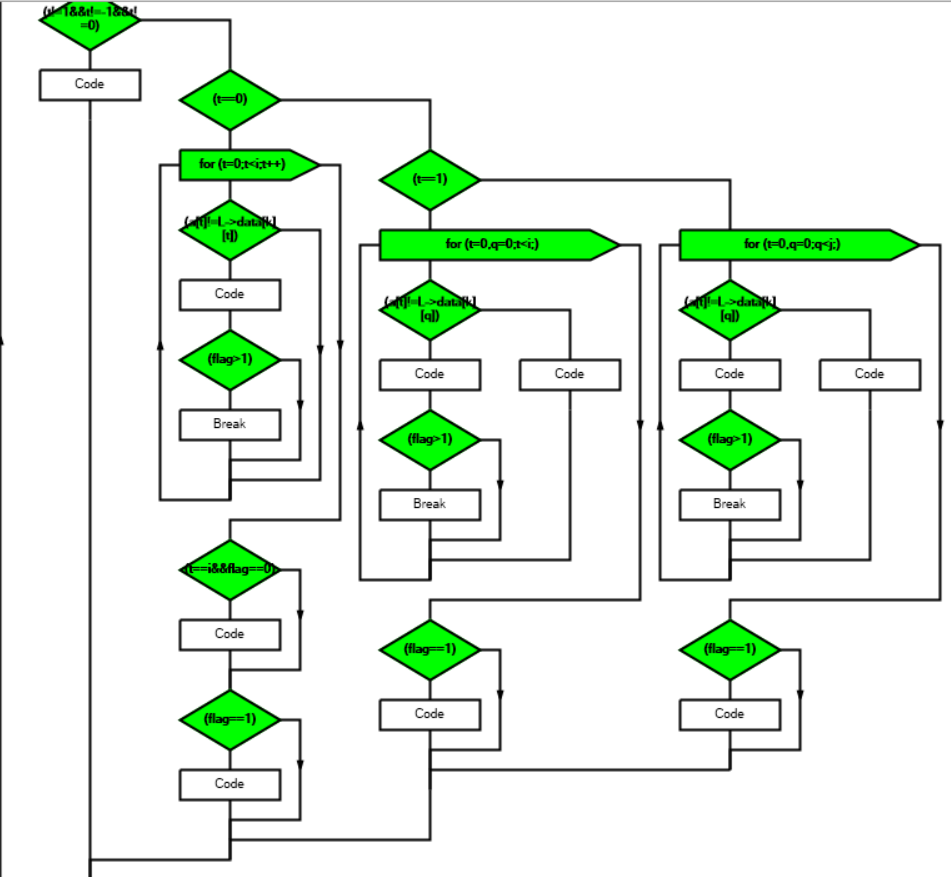
修改建议单词可以采用如下生成技术：

(1)在每一个可能位置插入‘a-'z'中的一者

(2)删除单词中的一个字符

(3)用‘a'-'z'中的一者取代单词中的任一字符

1. **分析与设计**
2. 思路：1、首先用顺序表将字典文件储存起来。
3. 遍历字典中每个单词。比较遍历单词与输入单词的长度，如果长度相差1或0，进行检查错误操作。
4. 检查错误时，根据两个互相比较单词的不同个数flag来判断是正确还是相似还是不匹配。flag为0是，查找成功，为1时，相似输出，为2时，不匹配。
5. 数据元素为字符串，对该顺序表进行遍历及与待测单词进行比较操作。
6. 流程图



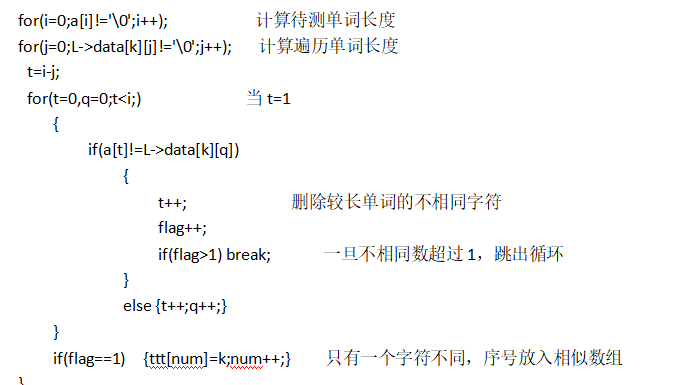
该流程图为判断遍历单词与待测单词是否相等、相似、不匹配的操作。

如流程图中t（即两单词长度之差）在比较范围内可能为1，-1，0

1和-1的主要区别在于当字符不同时，删除哪个单词中的字符（删除长度较长的单词的字符）。

此外将每个相似单词的地址用数组ttt【10000】存储，便于后续的输出。

以下为单词比较部分代码：



## **性能分析**

内存：1556

耗时：132

语言：C

代码长度：1123B

1. **调试与测试数据**

样例输入

i

is

has

have

be

my

more

contest

me

too

if

award

#

me

aware

m

contest

hav

oo

or

i

fi

mre

#

样例输出

me is correct

aware: award

m: i my me

contest is correct

hav: has have

oo: too

or:

i is correct

fi: i

mre: more me

1. **小结**

对于顺序表的存储结构不太熟悉，忘记开辟新节点。记住malloc函数的用法

# 任务J 复杂度分析(Ⅱ)

## 任务描述

题目同上任务I。

编写一个能读入字典文件，并对一个输入文件的单词进行检查，列出其中的错误单词并给出修改建议。

本题要求使用使用二叉排序树维护字典。

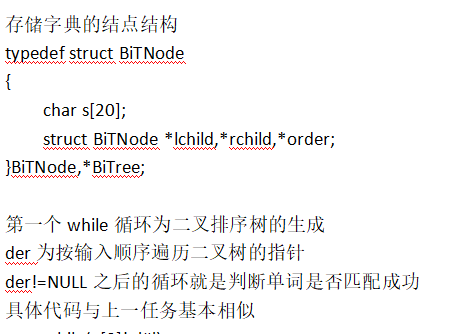
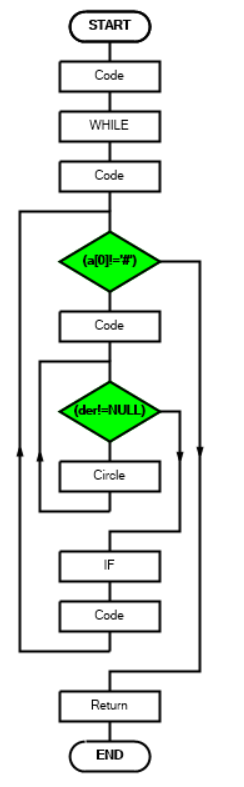
1. **分析与设计**
2. 思路：1、除去用二叉排序树存储字典外，一些单词比较的操作与上一题基本相似

2、这一题在生成二叉排序树时根据字典序来生成。

3、此外，在结点的结构中还另外加了一个\*order的指针域，可以按照输入字典的顺序来遍历二叉树，以便输出相似单词的顺序与样例一致。

（2）数据元素为字符串，基本操作为生成带有字典序的二叉排序树、后序遍历二叉排序树、遍历二叉排序树查找单词等。

（3）流程图



## **性能分析**

内存：1620

耗时：108

语言：C

代码长度：3456B

1. **调试与测试数据**

同上题任务I

## **小结**

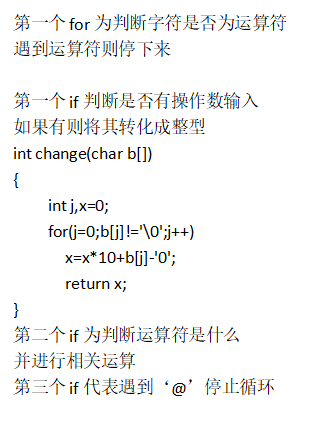
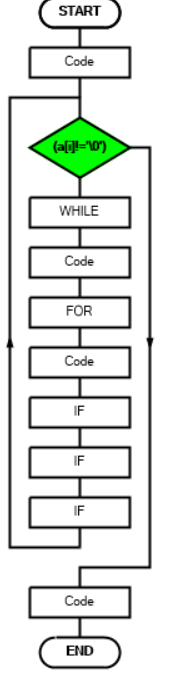
存储二叉树的时候要注意字典序。要考虑到相似单词的顺序。

# 任务L 后缀表达式求值

## 任务描述

为了便于处理表达式，常常将普通表达式（称为中缀表示）转换为后缀{运算符在后，如X/Y写为XY/表达式。在这样的表示中可以不用括号即可确定求值的顺序，如：(P+Q)\*(R－S) → PQ+RS－\*。后缀表达式的处理过程如下：扫描后缀表达式，凡遇操作数则将之压进堆栈，遇运算符则从堆栈中弹出两个操作数进行该运算，将运算结果压栈，然后继续扫描，直到后缀表达式被扫描完毕为止，此时栈底元素即为该后缀表达式的值。

1. **分析与设计**
2. 思路：该题可以利用栈存储，具体步骤题中有说明，只需要写好进栈、出栈等函数即可。
3. 数据元素为字符，具体操作为进栈、出栈及简单加减乘除运算
4. 流程图



## **性能分析**

内存：1092

耗时：0

语言：C

代码长度：1630B

**四、调试与测试数据**

样例输入

14 3 20 5 / \*8 - + @

样例输出

18

## **小结**

注意遇到运算符停下循环时要i++跳到下一个字符。

# 任务M 中缀表达式转后缀表达式

**一、任务描述**

输入一个中缀表达式，编程输出其后缀表达式，要求输出的后缀表达式的运算次序与输入的中缀表达式的运算次序相一致。为简单起见，假设输入的中缀表达式由＋（加）、－（减）、×（乘）、／（除）四个运算符号以及左右圆括号和英文字母组成，其中算术运算符遵守先乘除后加减的运算规则。假设输入的中缀表达式长度不超过300个字符，且都是正确的，即没有语法错误，并且凡出现括号其内部一定有表达式，即内部至少有一个运算符号。

中缀表达式转后缀表达式的方法：

1.遇到操作数：直接输出（添加到后缀表达式中）

2.栈为空时，遇到运算符，直接入栈

3.遇到左括号：将其入栈

4.遇到右括号：执行出栈操作，并将出栈的元素输出，直到弹出栈的是左括号，括号不输出。

5.遇到其他运算符：加减乘除：弹出所有优先级大于或者等于该运算符的栈顶元素，然后将该运算符入栈

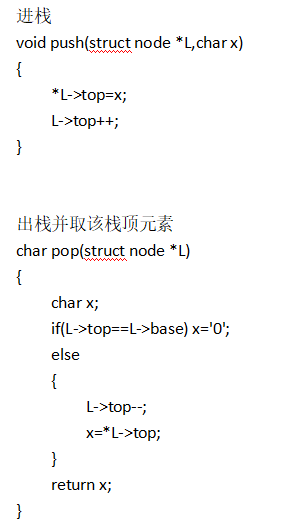
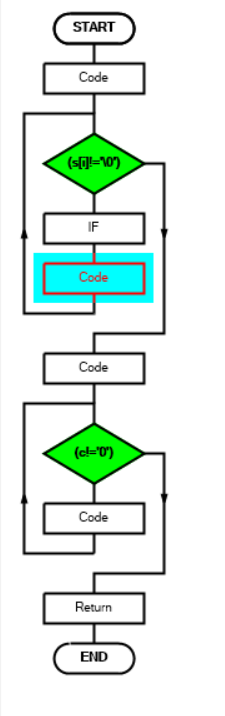
6.最终将栈中的元素依次出栈，输出。

**二、分析与设计**

（1）该题可以利用栈存储，具体步骤题中有说明，只需要根据题中步骤按条件写好进栈、出栈等函数即可。

（2）数据元素为字符。进行入栈出栈操作。

（3）流程图



## **三、性能分析**

内存：1092

耗时：0

语言：C

代码长度：1900B

**四、调试与测试数据**

样例输入

X+A\*(Y-B)-Z/F

样例输出

XAYB-\*+ZF/-

## **五、小结**

取栈顶元素时也将元素出栈了，如果这个元素不符合出栈要求，还得重新将该元素入栈。

**任务N 二叉树的创建和文本显示**

## 任务描述

编一个程序，读入先序遍历字符串，根据此字符串建立一棵二叉树（以指针方式存储）。

例如如下的先序遍历字符串：

A ST C # # D 10 # G # # F # # #

各结点数据（长度不超过3），用空格分开，其中“#”代表空树。

建立起此二叉树以后，再按要求输出二叉树。

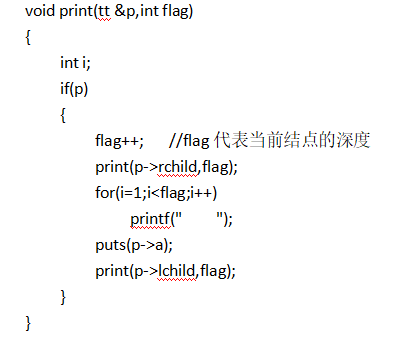
1. **分析与设计**
2. 思路：1、根据先序递归算法创建二叉树，再根据右子树、根节点、左子树的顺序打印二叉树

2、打印二叉树时，因为要根据结点的深度打印空格，所以设一个全局变量flag来存储深度。

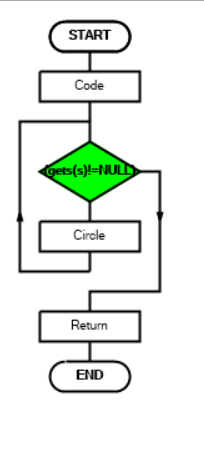
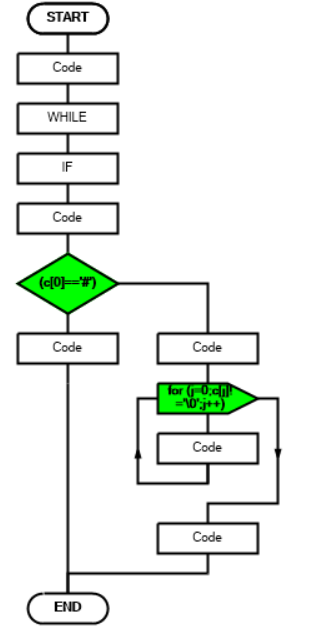
（2）数据元素为字符串，主要操作为先序递归算法二叉树的创建和二叉树的遍历打印。

1. 流程图

打印二叉树函数部分



创建二叉树递归函数 主函数



## **性能分析**

内存：1092

耗时：0

语言：C++

代码长度：1224B

1. **调试与测试数据**

样例输入

A ST C # # D 10 # G # # F # # #

4 2 1 # # 3 # # 5 # 6 # #

样例输出

A

F

D

G

10

ST

C

6

5

4

3

2

1

## **小结**

弄清楚全局变量和局部变量之间的区别，以及在函数中形参的传递等。

# 任务 O 表达式树的创建与输出

## 一、任务描述

编一个程序，读入先序遍历字符串，根据此字符串建立一棵二叉树（以指针方式存储），请注意的是，我们保证该树一定是表达式树（见教材5.2 5.8）。

例如下面的先序遍历字符串：

+ 13 # # \* 5 # # 9 # #

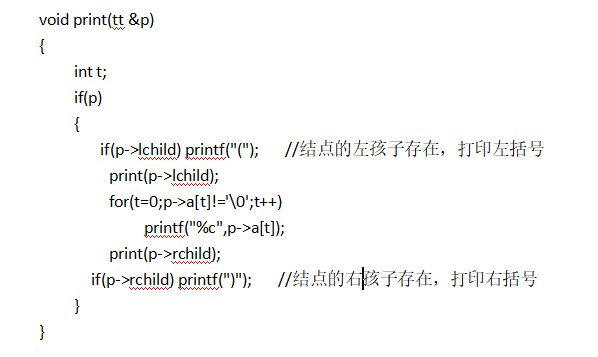
运算符只可能是加减乘除，数值为小于等于100，各结点用空格分开，其中“#”代表空树。

## 分析与设计

（1）思路：二叉树的创建与上一题基本相似，只需要判断在什么时候加括号就行了。

（2）数据元素为字符串，主要操作为先序递归算法二叉树的创建和二叉树的遍历打印。

（3）流程图与上一题相同，不同的在于打印函数



## 性能分析

内存：1092

耗时：0

语言：C++

代码长度：1251B

## 四、调试与测试数据

样例输入

+ 13 # # \* 5 # # 9 # #

\* + 13 # # 5 # # 9 # #

样例输出

(13+(5\*9))

((13+5)\*9)

## 五、小结

注意字符串的输入，将整个输入转换成单个字符串时i++不要忘记。

# 任务 P 表达式树的值

## 一、任务描述

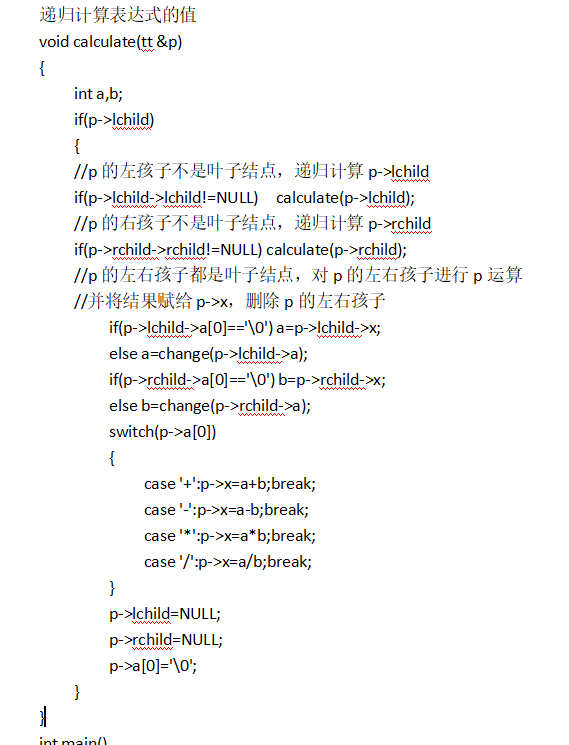
读入表达式树的先序遍历字符串，求其值。运算符只可能是加减乘除，保证输入的每个子表达式树的结果都是整数值且可以用C语言的int类型表达。

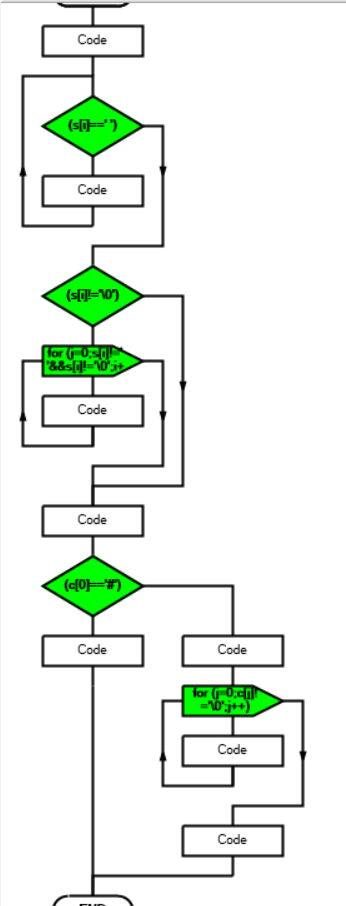
## 二、分析与设计

(1) 思路：此题要对二叉树存储的表达式进行计算，在上一题的结构中再加一个数据元素元素x，存储左右子树递归计算后的值。

(2) 数据元素为整型和字符串。

(3)流程图及代码





## 三、性能分析

内存：1092

耗时：0

语言：C++

代码长度：2212B

## 调试与测试数据

+ 13 # # \* 5 # # 9 # #

\* + 13 # # 5 # # 9 # #

样例输出

(13+(5\*9))=58

((13+5)\*9)=162小结

## 五、小结

进行p->lchild->lchild运算时要考虑到指针越界的情况，在之前加上while（p->lchild）防止指针越界。