数据结构实验报告

张文

2007-06-26

1 设计人员相关信息

1.1 设计者姓名、学号、班级：

张文
200532530040
2005级信息安全2班

1.2 设计日期：

2007年6月9日

1.3 上机环境：

WindowsXP SP2
Microsoft Visual C++ 6.0

2 程序设计相关信息

2.1 实验题目：

实验题目：7.11.7

编写一个程序，用二叉树来表示代数表达式，树的每个结点包含一个运算符，代数表达式由输入得到（其中，只包含=、-、*、/和用一个字母表示的数且没有错误，并且按照先加减后乘除的原则）。

2.2 实验项目组成：

7-11-7.cpp
7-11-7.exe

2.3 实验项目的程序结构（程序中的函数调用关系图）：

程序函数列表

1. struct BTree* Partition(char *ThisExp,int start,int end)
   用途：按照运算法则，在给定范围内，确定二叉树新节点的符号，并以此符号为界限，分别递归处理它的左右区间。递归的边界条件是在给定区间内无原算符号。函数返回当前建立的新节点的地址。

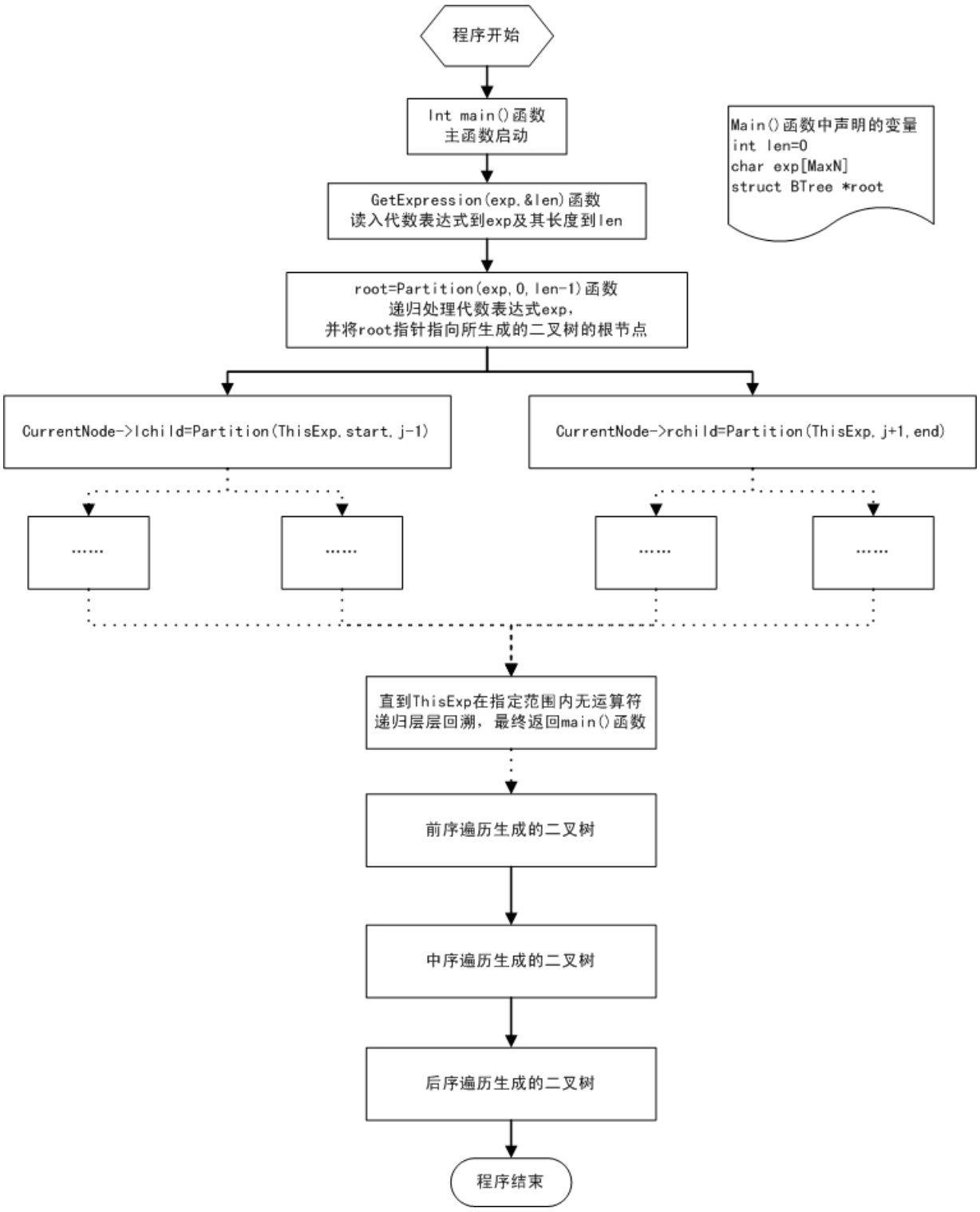
2. int GetExpression(char *ThisExp,int *n)
   用途：从屏幕缓冲区获取用户输入的代数表达式。

3. void InOrderTravel(struct BTree * Point)
   用途：对所给二叉树进行中序遍历

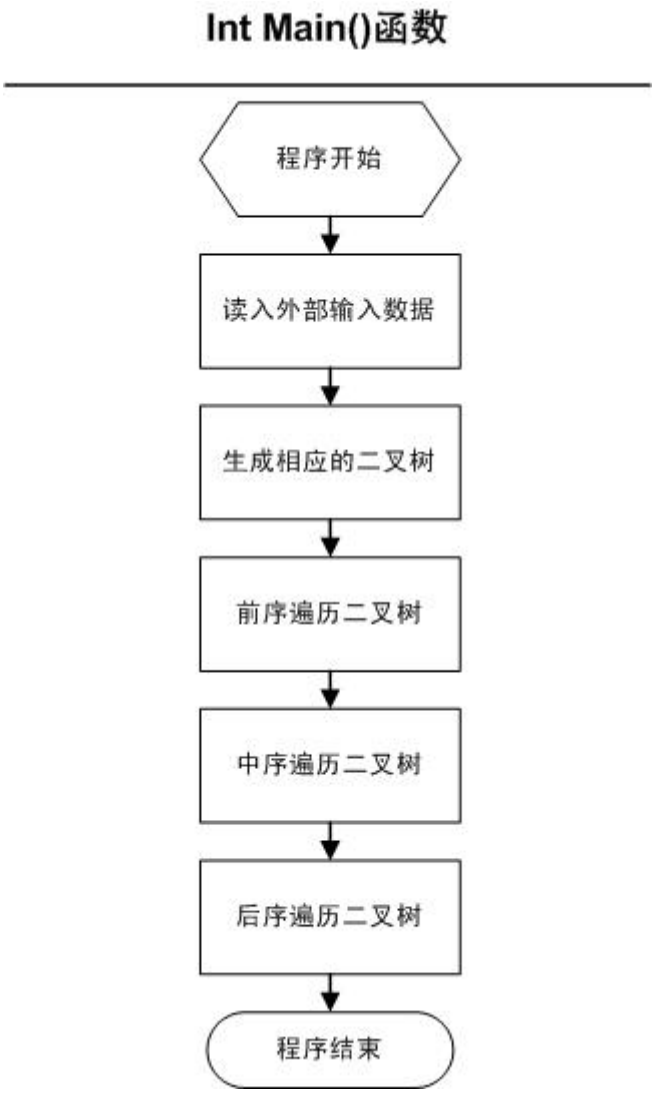
4. void PreOrderTravel(struct BTree * Point)
   用途：对所给二叉树进行前序遍历

5. void PostOrderTravel(struct BTree * Point)
   用途：对所给二叉树进行后序遍历

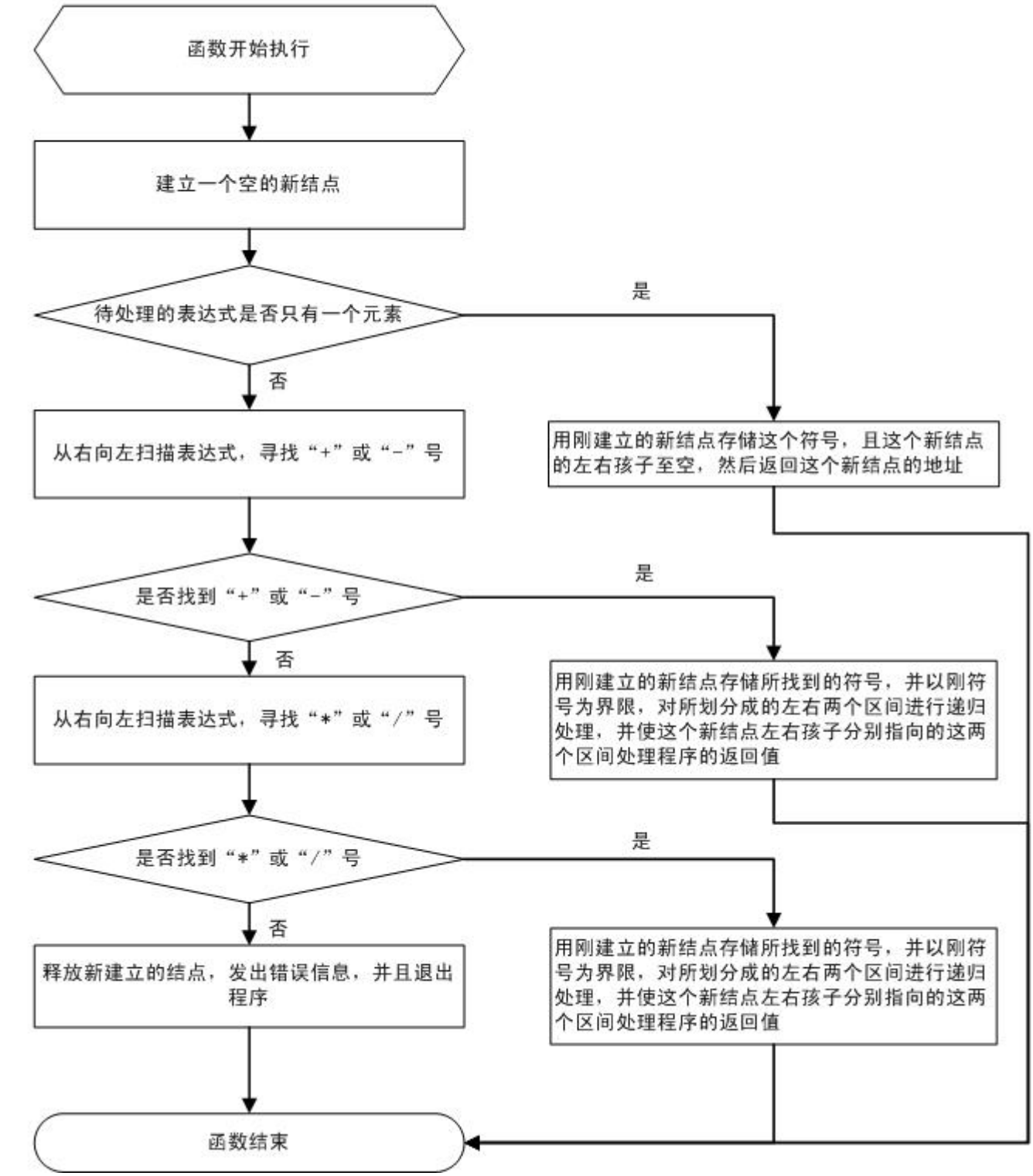
函数关系调用图


程序开始
Int main()函数
主函数启动
Main()函数中声明的变量
int len=0
char exp[MaxN]
struct BTree *root
GetExpression(exp,&len)函数
读入代数表达式到exp及其长度到len
root=Partition(exp,0,len-1)函数
递归处理代数表达式exp，
并将root指针指向所生成的二叉树的根节点
CurrentNode->lchild=Partition(ThisExp,start,j-1)
CurrentNode->rchild=Partition(ThisExp,j+1,end)
……
……
……
……
直到ThisExp在指定范围内无运算符
递归层层回溯，最终返回main()函数
前序遍历生成的二叉树
中序遍历生成的二叉树
后序遍历生成的二叉树
程序结束

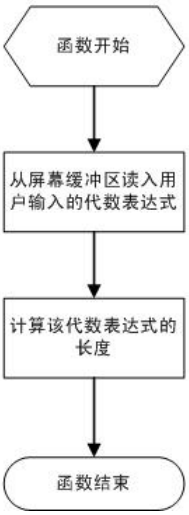
2.4 算法描述或流程图：


Int Main()函数
程序开始
读入外部输入数据
生成相应的二叉树
前序遍历二叉树
中序遍历二叉树
后序遍历二叉树
程序结束

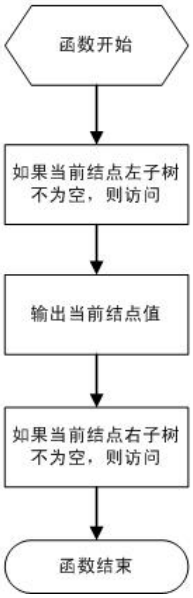
struct BTree* Partition(char *ThisExp,int start,int end)函数



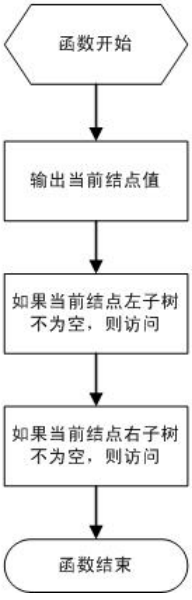
int GetExpression(char *ThisExp,int *n)函数



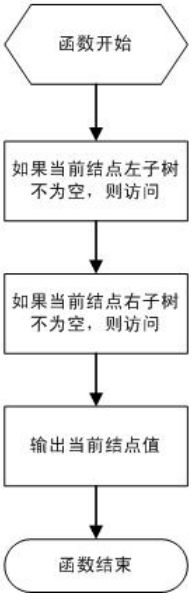
void InOrderTravel(struct BTree * Point)函数



void PreOrderTravel(struct BTree * Point)函数



void PostOrderTravel(struct BTree * Point)函数



2.5 实验数据和实验结果：

1. 输入数据：a+b
   输出数据：
   前缀表达式:+ab
   中缀表达式:a+b
   后缀表达式:ab+

2. 输入数据：a/b
   输出数据：
   前缀表达式:/ab
   中缀表达式:a/b
   后缀表达式:ab/

3. 输入数据：a+b*c-e/f
   输出数据：
   前缀表达式:-+a*bc/ef
   中缀表达式:a+b*c-e/f
   后缀表达式:abc*+ef/-

实验结果

这个实验基本完成，但尚有许多地方还可以进行改进，有对输入代数表达式合法性的检测；未知变量由多个字母组成的情形；加入各种括号的情形；还有二叉树的图形表示，而非使用遍历来显示二叉树结构；程序结束后对已分配空间的回收等等。

通过实验，再次加深了我对二叉树的认识，以及对二分、递归等概念的复习，总的来说，还是有一定获益的。

而通过实验报告的编写，也让我了解了TEX这款闻名遐迩的排版软件，以及Microsoft Office Visio等软件。通过着许许多多知识的学习、联系、与运用，我的能力得到了一定提高。

3 程序源代码

```
1  #include<stdio.h>
2  #include<string.h>
3  #include<stdlib.h>
4
5  #define MaxN 20
6
7  struct BTree
8  {
9      char value;
10     struct BTree *lchild,*rchild;
11 };
12
13 struct BTree* Partition(char *,int,int);
14 int GetExpression(char*,int*);
15 void InOrderTravel(struct BTree *);
16 void PreOrderTravel(struct BTree*);
17 void PostOrderTravel(struct BTree*);
18
19 int main()
20 {
21     int len=0;
22     char exp[MaxN];
23     struct BTree *root;
24
25     GetExpression(exp,&len);
```

```
26      root=Partition(exp,0,len-1);
27      printf("\nPreOrderExpression:");
28      PreOrderTravel(root);
29      printf("\nInOrderExpression:");
30      InOrderTravel(root);
31      printf("\nPostOrderExpression:");
32      PostOrderTravel(root);
33      printf("\n");
34      return 0;
35  }
36
37  int GetExpression(char *ThisExp,int *n)
38  {
39      scanf("%s",ThisExp);
40      *n=(int)strlen(ThisExp);
41      return 0;
42  }
43
44  struct BTree* Partition(char *ThisExp,int start,int end)
45  {
46      int i,j;
47      struct BTree* CurrentNode;
48
49      i=start;
50      j=end;
51      CurrentNode=(struct BTree*)malloc(sizeof(BTree));
52
```

```
53      if(i==j)//deal with the number node
54      {
55          CurrentNode->value=ThisExp[j];
56          CurrentNode->lchild=NULL;
57          CurrentNode->rchild=NULL;
58          return CurrentNode;
59      }
60
61      while(i<j && ThisExp[j]!='+' && ThisExp[j]!='-')j--;//deal with the + and - signs
62      if(ThisExp[j]=='+' || ThisExp[j]=='-')
63      {
64          CurrentNode->value=ThisExp[j];
65          CurrentNode->lchild=Partition(ThisExp,start,j-1);
66          CurrentNode->rchild=Partition(ThisExp,j+1,end);
67          return CurrentNode;
68      }
69
70      i=start;
71      j=end;
72      while(i<j && ThisExp[j]!='*' && ThisExp[j]!='/')j--;//deal with the * and / signs
73      if(ThisExp[j]=='*' || ThisExp[j]=='/')
74      {
75          CurrentNode->value=ThisExp[j];
76          CurrentNode->lchild=Partition(ThisExp,start,j-1);
77          CurrentNode->rchild=Partition(ThisExp,j+1,end);
78          return CurrentNode;
79      }
```

```
80      else
81      {
82          free(CurrentNode);
83          printf("/nError/n");
84          exit(0);
85      }
86  }
87
88  void InOrderTravel(struct BTree * Point)
89  {
90      if(Point->lchild!=NULL)InOrderTravel(Point->lchild);
91      printf("%c",Point->value);
92      if(Point->rchild!=NULL)InOrderTravel(Point->rchild);
93  }
94
95  void PreOrderTravel(struct BTree *Point)
96  {
97      printf("%c",Point->value);
98      if(Point->lchild!=NULL)PreOrderTravel(Point->lchild);
99      if(Point->rchild!=NULL)PreOrderTravel(Point->rchild);
100 }
101
102 void PostOrderTravel(struct BTree *Point)
103 {
104     if(Point->lchild!=NULL)PostOrderTravel(Point->lchild);
105     if(Point->rchild!=NULL)PostOrderTravel(Point->rchild);
106     printf("%c",Point->value);
```

```
107  }
```