

Project4 实验报告

21307035 邓栩瀛

1、 程序功能简要说明

- (1) 输入并识别前缀表达式，将其转换为中缀表达式
- (2) 对表达式中的变量进行赋值，并进行计算
- (3) 输入前缀表达式与运算符，与原表达式构成新的复合表达式

2、 程序运行截图，包括计算功能演示、部分实际运行结果展示、命令行或交互式界面效果等

(1) 输出界面

```
Enter 1:请输入前缀表达式
Enter 2:输出中缀表达式
Enter 3:对变量赋值
Enter 4:计算表达式的值
Enter 5:创建复合表达式
Enter 6:Exit

Please enter your choice
```

(2) 输入前缀表达式

```
Please enter your choice
1
Please enter the correct prefix expression:
+a*bc
Expression constructed successfully!
The midfix expression is
a+b*c
```

(3) 输出中缀表达式

```
Please enter your choice
2
The midfix expression is
a+b*c
```

(4) 对变量赋值

```
Please enter your choice:
3
Please enter the name of the variable to be assigned: a

The value to be assigned to is: 1
The expression after assignment is: 1+b*c
```

(5) 计算表达式

```
Please enter your choice:
4
1+2*3=7
```

(6) 创建复合表达式

```
Please enter your choice:
5
Please enter another expression: +*5^x2*8x
New expression constructed successfully!
The midfix of new expression is: 5*x^2+8*x

Please enter an operator: *

Expression E compounded successfully!Its expression is:
(1+2*3)*(5*x^2+8*x)The midfix representation of a composite prefix expression is: (1+2*3)*(5*x^2+8*x)
```

(7) 退出程序运行

```
Please enter your choice:
6

Press any key to exit!
```

3、 部分关键代码及其说明

- (1) ReadExpr(E) ——以字符序列的形式输入语法正确的前缀表达式并构成表达式 E

```
(2) int ReadExpr(BiTree &E, char *exp)
(3) {
```

```

(4)      int i,length;//length 为 exp 的长度
(5)      stack<SElemType> S;//操作符栈
(6)      BiTree p,q;
(7)      gets(exp);
(8)      length=strlen(exp);
(9)      E=(BiTree)malloc(sizeof(BiTNode));
(10)     E->lchild=NULL;
(11)     E->rchild=NULL;
(12)     if(length==1)
(13)     {
(14)         if(isdigit(exp[0]))
(15)         {
(16)             E->data.tag=INT;
(17)             E->data.num=exp[0]-'0';
(18)             return 1;
(19)         }
(20)         else if(isalpha(exp[0]))
(21)         {
(22)             E->data.tag=CHAR;
(23)             E->data.c=exp[0];
(24)             return 1;
(25)         }
(26)         else
(27)         {
(28)             cout<<"ERROR"<<endl;
(29)             return 0;
(30)         }
(31)     }
(32)     else
(33)     {
(34)         judge(E,exp[0]);
(35)         q=E;
(36)         S.push(q);
(37)         S.push(q);
(38)         for(i=1;i<length && !S.empty();i++)
(39)         {
(40)             p=(BiTree)malloc(sizeof(BiTNode));
(41)             judge(p,exp[i]);
(42)             p->lchild=NULL;
(43)             p->rchild=NULL;
(44)             if(isoperator(exp[i]))
(45)             {
(46)                 if(!q->lchild)//左非空, 往左走
(47)                 {

```

```

(48)                q->lchild=p;
(49)                S.push(p);
(50)                q=p;
(51)            }
(52)            else//右非空，往右走
(53)            {
(54)                q->rchild=p;
(55)                S.push(p);
(56)                q=p;
(57)            }
(58)        }
(59)        else//非运算符，出栈
(60)        {
(61)            if(!q->lchild)//左非空，往左走
(62)            {
(63)                q->lchild=p;
(64)                q=S.top();
(65)                S.pop();
(66)            }
(67)            else
(68)            {
(69)                q->rchild=p;
(70)                q=S.top();
(71)                S.pop();
(72)            }
(73)        }
(74)    }
(75)    if(S.empty() && i>=length)
(76)    {
(77)        return 1;
(78)    }
(79)    else
(80)    {
(81)        return 0;
(82)    }
(83) }
(84) }

```

(2) WriteExpr(E)——用带括弧的中缀表示式输出表达式 E

```

1. void WriteExpr(BiTree E)
2. {
3.     if(E)//树非空

```

```

4.      {
5.          //左子树
6.          if(E->lchild && E->lchild->data.tag==CHAR)//左子树非空且为字符
7.          {
8.              if(Compare(E->data.c,E->lchild->data.c))
9.                  //当前节点比左子树优先级要高
10.             {
11.                 cout<<"(";
12.                 WriteExpr(E->lchild);
13.                 cout<<")";
14.             }
15.             else WriteExpr(E->lchild);
16.         }
17.         else
18.             WriteExpr(E->lchild);
19.         //根节点
20.         if(E->data.tag==INT)
21.         {
22.             cout<<E->data.num;
23.         }
24.         else
25.         {
26.             cout<<E->data.c;
27.         }
28.         //右子树
29.         if(E->rchild && E->rchild->data.tag==CHAR)
30.         {
31.             if(Compare(E->data.c,E->rchild->data.c))
32.             {
33.                 cout<<"(";
34.                 WriteExpr(E->rchild);
35.                 cout<<")";
36.             }
37.             else WriteExpr(E->rchild);
38.         }
39.         else WriteExpr(E->rchild);
40.     }
41. }

```

(3) Assign(V, c)——实现对变量 V 的赋值($V = c$)

(4) `void Assign(BiTree &E, char V, int num, int &flag)//flag 判断赋值成功`

```

(5)  {
(6)      if(E)
(7)      {
(8)          if(E->data.tag==CHAR && E->data.c==V)
(9)          {
(10)              E->data.tag==INT;
(11)              E->data.num=num;
(12)              flag=1;
(13)          }
(14)          Assign((E->lchild),V,num,flag);
(15)          Assign((E->rchild),V,num,flag);
(16)      }
(17) }

```

(4) Value(E)——对算术表达式 E 求值

```

1. int Value(BiTree E)
2. {
3.     if(E)
4.     {
5.         //叶子节点
6.         if(!E->lchild && !E->rchild && E->data.tag==INT)
7.         {
8.             return E->data.num;
9.         }
10.        return operate(Value(E->lchild),E->data.c,Value(E->rchild));
11.    }
12.    else return 0;
13. }

```

(5) CompoundExpr(P, E1, E2)——构成一个新的复合表达式

(E1)P(E2)

```

1. void CompoundExpr(char P,BiTree &E1,BiTree E2)
2. {
3.     BiTree E;
4.     E=(BiTree)malloc(sizeof(BiTNode));
5.     E->data.tag=CHAR;
6.     E->data.c=P;
7.     E->lchild=E1;

```

```
8.      E->rchild=E2;
9.      E1=E;
10.     cout<<"Expression E compounded successfully!"<<endl;
11.     cout<<"Its expression is"<<endl;
12.     WriteExpr(E);
13. }
```