Project4 实验报告

21307035 邓栩瀛

1、 程序功能简要说明

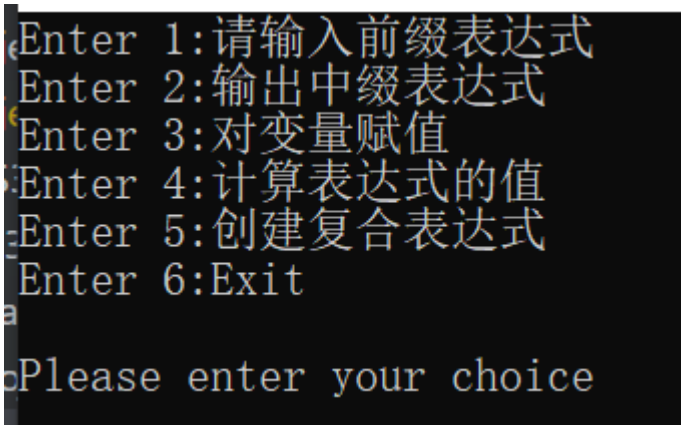
（1）输入并识别前缀表达式，将其转换为中缀表达式

（2）对表达式中的变量进行赋值，并进行计算

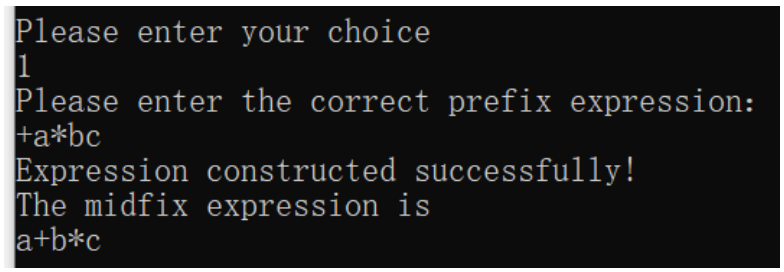
（3）输入前缀表达式与运算符,与原表达式构成新的复合表达式

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

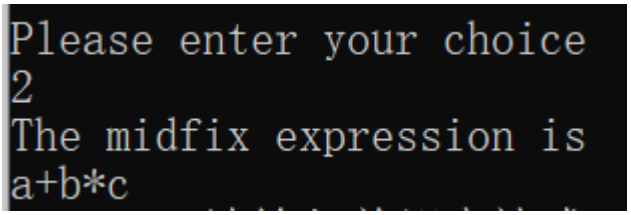
（1）输出界面



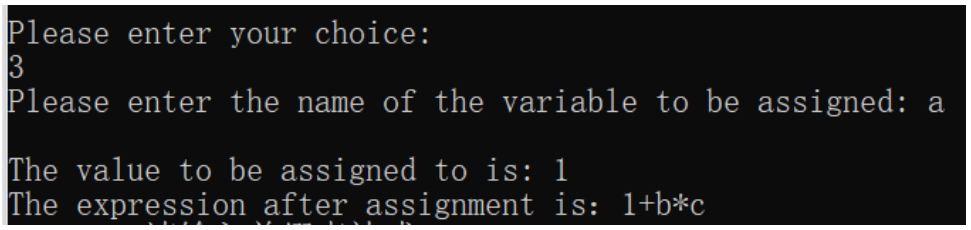
（2）输入前缀表达式



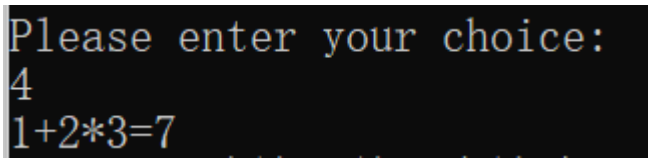
（3）输出中缀表达式



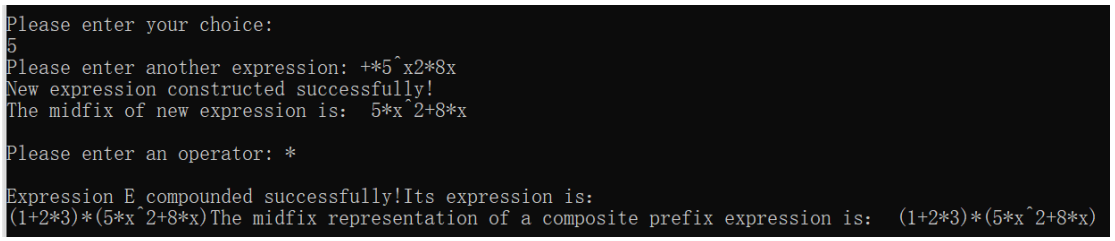
（4）对变量赋值



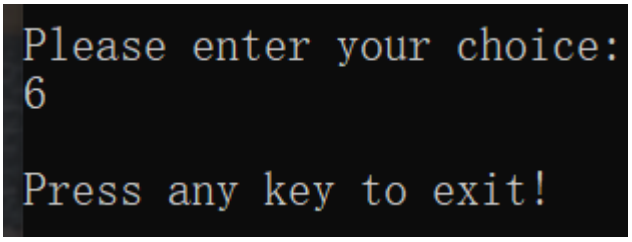
（5）计算表达式



（6）创建复合表达式



（7）退出程序运行



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)                {
```

```
                    q->lchild=p;
                    S.push(p);
                    q=p;
                }
                else//右非空，往右走
                {
                    q->rchild=p;
                    S.push(p);
                    q=p;
                }
            }
            else//非运算符，出栈
            {
                if(!q->lchild)//左非空，往左走
                {
                    q->lchild=p;
                    q=S.top();
                    S.pop();
                }
                else
                {
                    q->rchild=p;
                    q=S.top();
                    S.pop();
                }
            }
        }
        if(S.empty() && i>=length)
        {
            return 1;
        }
        else
        {
            return 0;
        }
    }
}
```

（2）WriteExpr(E)——用带括弧的中缀表示式输出表达式 E

```
void WriteExpr(BiTree E)
{
    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
```

```
{
    if(E)
    {
        if(E->data.tag==CHAR && E->data.c==V)
        {
            E->data.tag==INT;
            E->data.num=num;
            flag=1;
        }
        Assign((E->lchild),V,num,flag);
        Assign((E->rchild),V,num,flag);
    }
}
```

（4）Value(E)——对算术表达式 E 求值

```
int Value(BiTree E)
{
    if(E)
    {
        //叶子节点
        if(!E->lchild && !E->rchild && E->data.tag==INT)
        {
            return E->data.num;
        }
        return operate(Value(E->lchild),E->data.c,Value(E->rchild));

    }
    else return 0;
}
```

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

```
void CompoundExpr(char P,BiTree &E1,BiTree E2)
{
    BiTree E;
    E=(BiTree)malloc(sizeof(BiTNode));
    E->data.tag=CHAR;
    E->data.c=P;
    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. }
```