Project 3

数据科学与计算机 15352146 1507 简智勇

实现：在求导数方面指数为未知数的没有实现

主要函数说明

一、读入函数

string ReadExpr(string s)

观察得：算术表达式树的叶子节点必定为 0-9 或者 a-z；所以传入一个字符串，

在 0-9 或者 a-z 后面补上两个##表示叶子节点；

实质为改造字符串；

二、树的创建函数

bnode* createtree(string s , int &pos){ }

主要思想：

观察得：算术表达式树的叶子节点必定为 0-9 或者 a-z；所以传入一个字符串，

在 0-9 或者 a-z 后面补上两个##表示叶子节点；

所以树的创建函数运用递归算法：

```
bnode* root = NULL;
if(pos < s.length() && s[pos] != invalid) //当该位置不为空时
{
    root = new bnode;
    root->data = s[pos];
    root->lchild = createtree(s , ++pos);
```

root->rchild = createtree(s , ++pos);

}

return root;

三、 输出中缀表达式函数

void WriteExpr(string por, bool &flag){ }

por 为后缀表达式字符串，flag 返回是否存在格式错误；

利用后缀表达式的运算法则，利用栈来进行，每提取一个算式，加上括号（出最后的计算除外），输出的就为中缀表达式

四、 赋值函数

void assign(bnode* root , string V , string value){ }

运用广度优先遍历法则，没找到一个 V，将 V 换成 value；

五、 求值函数

void Value(string posexpr){ }

同样利用树求出其后缀遍历式，然后利用后缀式进行运算求值；

六、 两个表达式的复合

bnode* CompoundExpr(char op , bnode* root1 , bnode* root2) {}

新建一个节点作为新的根，两棵树分别为其左右子树；

七、 合并常数项函数

void MergeConst(bnode *root){ }

利用遍历法，从最低端开始计算，只要节点两端为数字，就进行计算，然后节点数据改为结果，左右子树指向空，接下来继续往上每个节点依次判断。

特殊情况：只要一边为 0 的时候，假若节点为*，那么直接节点数据变成 0，左右指数指向空，若左边为 0，假若节点为/，节点数据变为 0，左右指数指向空;

八、 求导函数

void Diff(bnode* root , string var){ }

利用递归算法；

1、 加减法求导一样，在一起讨论，先对左子树进行求导，假若左子树数据为 var，变成 1，假如为运算符，那么继续递归，假如为其他变量，则变成 0；然后对右子树进行求导，原理同左子树；

2、 当对乘法求导时，考虑(ab)' = a'b + ab' ； 所以加入节点为*，先进行新字树的创建，*号变为+号，左右子树都变为 a*b，然后左子树对左子树进行求导，右子树对右子树进行求导。同理，当数据为 var 时，变为 1，当为运算符时，继续递归，若为其他变量，变为 0；

3、 除法求导和乘法求导一样，因为 (a/b)' = (a'b – ab') / (b*b) ，所以除号的左子树改为(a*b-a*b) ， 右子树改为 b*b，实际进行操作的是对左子树的左子树的左子树和左子树的右子树的右子树，同理，当数据为 var 时，变为 1，当为运算符时，继续递归，若为其他变量，变为 0；