

## 写在前面

※：第一次用这个仓库，怕传漏传错了所以把文件也附在blackboard上了，如果有什么问题请于老师直接看blackboard上的代码，谢谢于老师：)

## 测试案例

# CPP ASSIGNMENT2 REPORT@Jiko

## 代码实现

- 指数运算 (exp)
  - 思路
  - 代码实现
- III其他小工具 (printTools)
- IV数据读入与处理 (main)
  - 读入方式
    - 变量行处理
      - 思路
      - 代码实现
    - 算式行处理
    - 计算
      - 小工具
      - 指数
        - sqrt()
        - ^
        - 代码实现
      - 括号
      - 加减乘除
        - + - \* /
        - 括号及加减乘除的代码实现
- 写在后面

## I 数字存储 (numberSaver)

### struct number

把数字转为可用于计算的struct **number**。

struct **number**由4个元素组成：

```
1 struct number{
2     int num[10000]; //存储数字
3     int len;        //有效数字的长度
4     int pLoc;       //小数点的位置
5     int zf;         //数字的正负
6     number(){
7         memset(num, 0, sizeof(num));
8         len = 0;
9         zf = 1;
10        pLoc = -1;
11    }
12};
```

例：

数字235.23,

zf=1, pLoc=1, len=5, num: 32532

数字114.514

zf=-1, pLoc=2, len=6, num: 415411

※：默认的len为0，数组num从左到右为从低位到高位。

## cutZero

用于去除头尾无意义的0:

```
1 struct number cutZero(number key){
2     while (key.num[key.len-1] == 0&&key.len-2>key.pLoc){//去掉头部的0
3         key.len--;
4     }
5     int zeroCounter = 0;//去掉尾部的0
6     int originLen = key.len;
7     int ploc_tem = key.pLoc;
8     for (int i = 0; i < key.pLoc+1; ++i) {
9         if (key.num[i] == 0){
10             key.len--;
11             ploc_tem--;
12             zeroCounter++;
13         }else{
14             break;
15         }
16     }
17     key.pLoc = ploc_tem;
18     for (int i = zeroCounter,j=0; i < originLen; ++i) {
19         key.num[j++] = key.num[i];
20     }
21     return key;
22 }
```

例:

-114.0处理后变为-114;

01234.34000处理后变为1234.34。

## 其他小工具

函数名	功能
charToNum	把字符串转为 <code>number</code> 类型, <code>num</code> 中仅保留数字, 将负号和小数点转移到 <code>z</code> 和 <code>pLoc</code> 中
integerBit	返回整数位的个数
decimalBit	返回小数位的个数
compareNoZf	比较两数绝对值的大小

## II基础运算实现 (Operator)

### 加 (plu)

长度: 两数中较大的小数位加两数中较大的整数位加一即为结果的位数 (若最后一位没有使用到会在最后使用cutZero除去无意义的0, 详见 [数字存储-cutZero](#)) ;

小数点位置: 与较大小数位的数相同;

正负：若同号则为这一符号，若异号则稍作改变调用减法（sub）。

## 思路

同位相加放到结果的同位上，最后从左至右依次进位。

## 代码实现

```
1 struct number plu(number n1,number n2){
2     if (n1.zf == n2.zf){
3         int pLoc = n1.pLoc>=n2.pLoc?n1.pLoc:n2.pLoc;
4         int cha = abs(n1.pLoc - n2.pLoc);
5         number longer = n1.pLoc>=n2.pLoc?n1:n2;
6         number shorter = n1.pLoc<n2.pLoc?n1:n2;
7         number result;
8         result.pLoc = pLoc;
9         result.len = pLoc+((n1.len-n1.pLoc)>=(n2.len-n2.pLoc)?(n1.len-
n1.pLoc):(n2.len-n2.pLoc));
10        result.zf = n1.zf;
11        int index_l = 0;
12        int index_s = 0;
13        for (int i = 0; i < result.len; ++i) {
14            if(i<cha){
15                result.num[i] = longer.num[index_l++];
16            } else{
17                result.num[i] =
result.num[i]+longer.num[index_l++]+shorter.num[index_s++];
18                if(result.num[i]>=10){
19                    if(i == result.len-1){
20                        result.len++;
21                    }
22                    result.num[i+1]+=1;
23                    result.num[i]-=10;
24                }
25            }
26        }
27        return result;
28    } else{ //异号转为减法
29        if (n1.zf == 1){
30            return sub(n1,n2);
31        } else{
32            return sub(n2,n1);
33        }
34    }
35 }
```

## 减 (sub)

长度：两数中较大的小数位加两数中较大的整数位加一即为结果的位数（若最后一位没有使用到会在最后使用cutZero除去无意义的0，详见 数字存储-cutZero）；

小数点位置：与较大小数位的数相同；

正负：若同号且被减数绝对值大于减数绝对值，正负与被减数相同；若同号且被减数绝对值小于减数绝对值，正负与被减数相反；若异号则稍加处理调用加法（plu）。

## 思路

同位作减法放到结果的同位上，最后从左至右依次借位。

## 代码实现

```
1  struct number sub(number n1,number n2){
2      if(n1.zf!=n2.zf){//异号转为加法
3          if(n1.zf == 1){//正的减负的
4              number new2;
5              new2 = n2;
6              new2.zf = 1;
7              return plu(n1,new2);
8          }else{//负的减正的
9              number new2;
10             new2 = n2;
11             new2.zf = -1;
12             return plu(n1,new2);
13         }
14     }else{
15         number big;
16         number small;
17         number result;
18         if(compareNoZf(n1,n2)==1){                //num n1>n2
19             big = n1;
20             small = n2;
21             result.zf = n1.zf;
22         }else if(compareNoZf(n1,n2)==-1){          //num n1<n2
23             big = n2;
24             small = n1;
25             result.zf = -n1.zf;
26         }else{                                     //num n1=n2
27             result.pLoc = -1;
28             result.zf = 1;
29             result.len = 1;
30             return result;
31         }
32         int cha = big.len - small.len;
33         result.len = max(integerBit(n1), integerBit(n2))+max(decimalBit(n1),
decimalBit(n2));
34         for (int i = result.len-1,j = big.len-1,k = small.len-1,c = 1; i >=0
; i--) {
35             if (c<cha){
36                 result.num[i] = big.num[j];
37                 c++;
38             }else{
39                 if (j>=0&&k>=0){
40                     result.num[i] = big.num[j--] - small.num[k--];
41                 }else if (j<0){
42                     result.num[i] = result.num[i] - small.num[k--];
43                 } else if (k<0){
44                     result.num[i] = result.num[i] + big.num[j--];
45                 }
46             }
47         }
48     }
```

```

48         for (int i = 0; i < result.len; ++i) {
49             if (result.num[i]<0){
50                 result.num[i]+=10;
51                 result.num[i+1]-=1;
52             }
53         }
54         result.pLoc = max(n1.pLoc,n2.pLoc);
55         return cutZero(result);
56     }
57 }

```

## 乘 (mul)

长度：最长为两数的位数相加

小数点位置：两数的小数点位置相加减一

正负：同号为正，异号为负

## 思路

数1的第*i*位与数2的第*j*位相乘放到结果的第*i+j*位上，最后从左至右进位。

## 代码实现

```

1  struct number mul(number n1,number n2){
2      number result;
3      result.len = n1.len+n2.len;
4      result.pLoc = n1.pLoc+n2.pLoc+1;
5      if (n1.zf == n2.zf){
6          result.zf = 1;
7      }else{
8          result.zf = -1;
9      }
10     for (int i = 0; i < n1.len; ++i) {
11         for (int j = 0; j < n2.len; ++j) {
12             result.num[i+j]+=n1.num[i]*n2.num[j];
13         }
14     }
15     int tem = 0;
16     for (int i = 0; i < result.len; ++i) {
17         result.num[i]+=tem;
18         tem = result.num[i]/10;
19         result.num[i] = result.num[i]%10;
20     }
21     return cutZero(result);
22 }

```

## 除 (div)

始终没有想到很好的方法，借用了[网络上的代码](#)，由于改代码仅适用两个整数相除的情况，且小数点后的数全部被舍弃，于是我稍加改动来适配自己的思路。网上代码的部分放到了另一个class “divide\_internet”中。其中bigDivide。其中 bigDivide 即为两个整数相除的结果。

## 代码实现

```
1 struct number div(number n1,number n2){
2     bn o1,o2;
3     int di = 8;
4     if (integerBit(n1)< integerBit(n2)){
5         di+= integerBit(n2)- integerBit(n1);
6     }
7     n1 = mul(n1, tenTimes(di));
8     printInfo(n1);
9     printInfo(n2);
10    o1.len = n1.len;
11    o2.len = n2.len;
12    for (int i = 0; i < o1.len; ++i) {
13        o1.data[i] = n1.num[i];
14    }
15    for (int i = 0; i < o2.len; ++i) {
16        o2.data[i] = n2.num[i];
17    }
18    bn oResult = bigDivide(o1,o2);
19    number result;
20    result.len = n1.len;
21    result.pLoc = n1.pLoc-n2.pLoc-1;
22    if (n1.zf==n2.zf){
23        result.zf = 1;
24    } else {
25        result.zf = -1;
26    }
27    for (int i = 0,j = 0; j<oResult.len; ++i,++j) {
28        result.num[i] = oResult.data[j];
29    }
30    result.pLoc+=di;
31    for (int i = 0; i < oResult.len; ++i) {
32        cout<<oResult.data[i];
33    }
34    return cutZero(result);
35 }
```

## 指数运算 (exp)

### 思路

将一个数自乘n次

## 代码实现

```
1 struct number exp(number n1,int times){
2     number result;
3     if (times%2==0){
4         result.zf = 1;
5     } else {
6         result.zf = n1.zf;
7     }
8     if (times == 0){
```

```

9         result.len = 1;
10        return result;
11    }else if (times>0){
12        result = n1;
13        while (times>1){
14            result = mul(result,n1);
15            times--;
16        }
17        return cutZero(result);
18    } else{
19        result = n1;
20        while (times<-1){
21            result= mul(result,n1);
22            times++;
23        }
24        number one;
25        one.num[0] = 1;
26        one.zf = 1;
27        one.len = 1;
28        return cutZero(div(one,result));
29    }
30 }

```

## III 其他小工具 (printTools)

```

1 void printInfo(number s);           //打印number类型的所有数据（仅用于debug）
2 void printNumber(number s);        //打印number存放的值的实际数据（最终答案用这个打印）

```

## IV 数据读入与处理 (main)

### 读入方式

一行一行读入，含有等号的行即为变量赋值行，最后不含有等号的行为算式行。

### 变量行处理

※：为防止与其他算法关键字重叠，目前只开放了x, y, z三个变量的赋值，可以根据需求增加。

### 思路

使用两个数组，一个是变量数组，一个是`number`（详见 数字存储-struct number）的数组，将等号左边的变量名存在变量数组的第i位，将等号右边的数字直接转为`number`形式存入`number`数组的第i位。

### 代码实现

```

1     int index = 0;
2     int index_equal = -1;
3     while (true){
4         cin.getline(input,10000);
5         for (int i = 0; i < sizeof input&&input[i]!='\0'; ++i) {
6             if (input[i] == '='){
7                 index_equal = i;
8                 break;
9             }

```



```

10     }
11     if (index_equal <= 0){
12         break;
13     } else{
14         char tem = input[0];
15         append(index_value++,tem, charToNum(input,index_equal+1,(sizeof
input)-1));
16     }
17     memset(input, '\0', sizeof (input));
18     index_equal = -1;
19 }

```

## 算式行处理

将数据和符号分开存放于两个数组中，一个数组为**char**类型的，一个为**number**（详见 数字存储 struct number）类型的。

**char**的空位用'0'填充，**number**的空位由**zero**填充（**zero**为一个**number**类型，其长度为0）。

例：为简便表示**number**数组中直接显示对应的值，**z**即**zero**。

算式为：2+sqrt(4)+5\*2

0 + s q r t ( 0 ) + 0 \* 0

2 z z z z z z 4 z z 5 z z

接下来根据优先级分步依次使用 `calculate0` `calculate1` `calculate2` 计算。如果要添加更高优先级的只需要再添加函数 `calculaten`（**n**可为任意后缀）并置于其他之前或根据优先级插入合适位置即可。嵌套规则也可直接在新函数中引用其他函数。

## 计算

当前有三个不同的优先级

分别为 指数 对应 `calculate0`， 括号 对应 `calculate1`， 加减乘除 对应 `calculate2`

## 小工具

```

1 int findLast(char* deal,number* deal_num,int head,int tail,int key);//找到该符号后一位相邻数
2 int findPrev(char* deal,number* deal_num,int head,int tail,int key);//找到该符号前一位相邻数

```

## 指数

### sqrt()

平方该数并存在s的位置。

^

调用**exp函数**（详见 基础运算实现-指数运算）。

用小工具查找前后相邻数实施运算

## 代码实现

```
1 void calculate0(char* deal,number* deal_num,int head,int tail){
2     number zero;
3     for (int i = head; i < tail; ++i) {
4         if (deal[i] == 's'&&deal[i+1] ==
5         'q'&&deal[i+2]=='r'&&deal[i+3]=='t'){
6             for (int j = i; j <= i+4; ++j) {
7                 deal[j] = '0';
8             }
9             deal[i+6] = '0';
10            deal_num[i] =
11            exp(deal_num[findLast(deal,deal_num,head,tail,i)],2);
12            deal_num[findLast(deal,deal_num,head,tail,i)] = zero;
13        }
14    }
15    for (int i = head; i < tail; ++i) {
16        if (deal[i] == '^'){
17            deal[i] = '0';
18            int last = findLast(deal,deal_num,head,tail,i);
19            int prev = findPrev(deal,deal_num,head,tail,i);
20            deal_num[prev] = exp(deal_num[prev],deal_num[last].getValue());
21        }
22    }
23 }
```

## 括号

找到括号位置，将括号内的算式看作一个新的算式，先在其中执行加减乘除运算。

## 加减乘除

+ - \* /

用小工具查找前后相邻数调用基础运算中的加减乘除，将结果存在靠前的`number`位置上。

## 括号及加减乘除的代码实现

```
1 void calculate1(char* deal,number* deal_num,int head,int tail){
2     int counterK = 0;
3     int indexL = -1;
4     int indexR = -1;
5     number zero;
6     for (int i = head; i <= tail; ++i) {
7         if (indexL == -1&&deal[i] == '('){
8             indexL = i;
9             counterK ++;
10            deal[i] = '0';
11        } else if (deal[i] == ')'){
12            counterK --;
13            if (counterK == 0){
14                indexR = i;
15                deal[i] = '0';
16            }
17        }
18        if (indexL != -1&&indexR != -1){
```

```

19         calculate2(deal,deal_num,indexL,indexR);
20     }
21 }
22 }
23 void calculate2(char* deal,number* deal_num,int head,int tail){
24     number zero;
25     for (int i = head; i <= tail; ++i) {
26         if (deal[i] == '*' || deal[i] == '/'){
27             if (deal[i] == '*'){
28                 int prev = findPrev(deal,deal_num,head,tail,i);
29                 int last = findLast(deal,deal_num,head,tail,i);
30                 deal[i] = '0';
31                 deal_num[prev] = mul(deal_num[prev],deal_num[last]);
32                 deal_num[last] = zero;
33             } else if(deal[i] == '/'){
34                 int prev = findPrev(deal,deal_num,head,tail,i);
35                 int last = findLast(deal,deal_num,head,tail,i);
36                 deal[i] = '0';
37                 deal_num[prev] = div(deal_num[prev],deal_num[last]);
38                 deal_num[last] = zero;
39             }
40         }
41     }

```

例：

## 写在后面

在cpp的学习中感觉自己已经被java惯坏了，本以为同为流行的编程语言应该差不了多少，却发现cpp需要注意和学习的地方比java多太多。另外在project1的乘法中我使用的是直接在数组上做文章，有点让自己的大脑过载了，这次使用了新学习的结构体一下子方便了好多，之前搞了好几天的乘法现在几个小时就能成功实现了。虽然每次做project都腰酸背痛，眼睛像要冒火，但做完之后看着各个代码各司其职成就感真的爆棚，希望能继续在于老师的课和project上学习更多有趣的新知识~