CPP_Project2_Calculator@Jiko

- **Name：纪可鸣**
- **SID：12112813**

仓库网址： https://github.com/JikoSchnee/SUSTECH_CS205_C-C-Project/tree/main/Project2/code

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 :)

本次Project分为四个部分，分别为 **数字存储 基础运算实现 其他小工具** 和 **数据读入与处理**，分别放在四个对应的.cpp文件中

测试案例

```
x=3
y=6
x+2*y

2^2+5

2*(1+5)

x=99999999999999999999999999.9999
y=0.001
x+y

x=99999999999999999999999999.9999
x+x

114514.114514*114514.2346

2.45/114514
```

目录

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：

```
struct number{
    int num[10000]; //存储数字
    int len;        //有效数字的长度
    int pLoc;       //小数点的位置
    int zf;         //数字的正负
    number(){
        memset(num, 0, sizeof(num));
        len = 0;
        zf = 1;
        pLoc = -1;
    }
};
```

> *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：

```
struct number cutZero(number key){
    while (key.num[key.len-1] == 0&&key.len-2>key.pLoc){//去掉头部的0
        key.len--;
    }
    int zeroCounter = 0;//去掉尾部的0
    int originLen = key.len;
    int ploc_tem = key.pLoc;
    for (int i = 0; i < key.pLoc+1; ++i) {
        if (key.num[i] == 0){
            key.len--;
            ploc_tem--;
            zeroCounter++;
        }else{
            break;
        }
    }
    key.pLoc = ploc_tem;
    for (int i = zeroCounter,j=0; i < originLen; ++i) {
        key.num[j++] = key.num[i];
    }
    return key;
}
```

> *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 *number* 类型， *num* 中仅保留数字，将负号和小数点转移到 *zf* 和 *pLoc* 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){
```

```
                    if(i == result.len-1){
                        result.len++;
                    }
                    result.num[i+1]+=1;
                    result.num[i]-=10;
                }
            }
        }
        return result;
    } else{ //异号转为减法
        if (n1.zf == 1){
            return sub(n1,n2);
        } else{
            return sub(n2,n1);
        }
    }
}
```

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

```
struct number sub(number n1,number n2){
    if(n1.zf!=n2.zf){//异号转为加法
        if(n1.zf == 1){//正的减负的
            number new2;
            new2 = n2;
            new2.zf = 1;
            return plu(n1,new2);
        }else{//负的减正的
            number new2;
            new2 = n2;
            new2.zf = -1;
            return plu(n1,new2);
        }
    }else{
        number big;
        number small;
        number result;
        if(compareNoZf(n1,n2)==1){          //num n1>n2
            big = n1;
            small = n2;
            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          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

```
struct number mul(number n1,number n2){
    number result;
    result.len = n1.len+n2.len;
    result.pLoc = n1.pLoc+n2.pLoc+1;
    if (n1.zf == n2.zf){
        result.zf = 1;
    }else{
        result.zf = -1;
    }
    for (int i = 0; i < n1.len; ++i) {
        for (int j = 0; j < n2.len; ++j) {
            result.num[i+j]+=n1.num[i]*n2.num[j];
        }
    }
    int tem = 0;
    for (int i = 0; i < result.len; ++i) {
        result.num[i]+=tem;
        tem = result.num[i]/10;
        result.num[i] = result.num[i]%10;
    }
    return cutZero(result);
}
```

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

```
struct number div(number n1,number n2){
    bn o1,o2;
    int di = 8;
    if (integerBit(n1)< integerBit(n2)){
        di+= integerBit(n2)- integerBit(n1);
    }
    n1 = mul(n1, tenTimes(di));
    printInfo(n1);
    printInfo(n2);
    o1.len = n1.len;
    o2.len = n2.len;
    for (int i = 0; i < o1.len; ++i) {
        o1.data[i] = n1.num[i];
    }
    for (int i = 0; i < o2.len; ++i) {
        o2.data[i] = n2.num[i];
    }
    bn oResult = bigDivide(o1,o2);
    number result;
    result.len = n1.len;
    result.pLoc = n1.pLoc-n2.pLoc-1;
    if (n1.zf==n2.zf){
```

```
            result.zf = 1;
        } else {
            result.zf = -1;
        }
        for (int i = 0,j = 0; j<oResult.len; ++i,++j) {
            result.num[i] = oResult.data[j];
        }
        result.pLoc+=di;
        for (int i = 0; i < oResult.len; ++i) {
            cout<<oResult.data[i];
        }
        return cutZero(result);
    }
```

指数运算（exp）

思路

将一个数自乘n次

代码实现

```
    struct number exp(number n1,int times){
        number result;
        if (times%2==0){
            result.zf = 1;
        } else {
            result.zf = n1.zf;
        }
        if (times == 0){
            result.len = 1;
            return result;
        }else if (times>0){
            result = n1;
            while (times>1){
                result = mul(result,n1);
                times--;
            }
            return cutZero(result);
        } else{
            result = n1;
            while (times<-1){
                result= mul(result,n1);
                times++;
            }
            number one;
            one.num[0] = 1;
            one.zf = 1;
            one.len = 1;
            return cutZero(div(one,result));
        }
    }
```

Ⅲ其他小工具（printTools）

```
void printInfo(number s);       //打印number类型的所有数据（仅用于debug）
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

```
    int index = 0;
    int index_equal = -1;
    while (true){
        cin.getline(input,10000);
        for (int i = 0; i < sizeof input&&input[i]!='\0'; ++i) {
            if (input[i] == '='){
                index_equal = i;
                break;
            }
        }
        if (index_equal <= 0){
            break;
        } else{
            char tem = input[0];
            append(index_value++,tem, charToNum(input,index_equal+1,(sizeof
input)-1));
        }
        memset(input,'\0',sizeof (input));
        index_equal = -1;
    }
```

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+sqrt(0)+0*0
>
> 2zzzzzz4zz5z2

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

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

```
void calculate0(char* deal,number* deal_num,int head,int tail){
    number zero;
    for (int i = head; i < tail; ++i) {
        if (deal[i] == 's'&&deal[i+1] == 'q'&&deal[i+2]=='r'&&deal[i+3]=='t'){
            for (int j = i; j <= i+4; ++j) {
                deal[j] = '0';
            }
            deal[i+6] = '0';
            deal_num[i] = exp(deal_num[findLast(deal,deal_num,head,tail,i)],2);
            deal_num[findLast(deal,deal_num,head,tail,i)] = zero;
        }
    }
    for (int i = head; i < tail; ++i) {
        if (deal[i] == '^'){
            deal[i] = '0';
            int last = findLast(deal,deal_num,head,tail,i);
            int prev = findPrev(deal,deal_num,head,tail,i);
            deal_num[prev] = exp(deal_num[prev],deal_num[last].getValue());
        }
    }
}
```

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

```
void calculate1(char* deal,number* deal_num,int head,int tail){
    int counterK = 0;
    int indexL = -1;
    int indexR = -1;
    number zero;
    for (int i = head; i <= tail; ++i) {
        if (indexL == -1&&deal[i] == '('){
            indexL = i;
            counterK ++;
            deal[i] = '0';
        } else if (deal[i] == ')'){
            counterK --;
            if (counterK == 0){
                indexR = i;
                deal[i] = '0';
            }
        }
        if (indexL!=-1&&indexR!=-1){
            calculate2(deal,deal_num,indexL,indexR);
        }
    }
}
void calculate2(char* deal,number* deal_num,int head,int tail){
    number zero;
    for (int i = head; i <= tail; ++i) {
        if (deal[i] == '*'||deal[i] == '/'){
            if (deal[i] == '*'){
                int prev = findPrev(deal,deal_num,head,tail,i);
                int last = findLast(deal,deal_num,head,tail,i);
                deal[i] = '0';
                deal_num[prev] = mul(deal_num[prev],deal_num[last]);
                deal_num[last] = zero;
            } else if(deal[i] == '/'){
                int prev = findPrev(deal,deal_num,head,tail,i);
                int last = findLast(deal,deal_num,head,tail,i);
                deal[i] = '0';
                deal_num[prev] = div(deal_num[prev],deal_num[last]);
                deal_num[last] = zero;
            }
        }
    }
```

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~