report

1201508华羽霄

2022年9月24日 星期六 22:22

一、判断合法性

在正式开始计算之前，我先对程序进行了测试，判断输入是否合法，并且返回输入的类型（type：0为非法，1为整数，2为小数，3为科学计数），以及是否为负数（if_neg：0为正数，1为负数），以下是测试样例：

1. 合法的输入：

测试样例：1

```
(base) hyx13701490089@huayuxiaodeMacBook-Pro pro1 % ./a.out
1 correct type:1 if_neg:0%
```

测试样例：-1

```
(base) hyx13701490089@huayuxiaodeMacBook-Pro pro1 % ./a.out
1 correct type:1 if_neg:1%
```

测试样例：1.1

```
(base) hyx13701490089@huayuxiaodeMacBook-Pro pro1 % ./a.out
1 correct . correct 1 correct type:1 if_neg:0%
```

测试样例：1e1

```
(base) hyx13701490089@huayuxiaodeMacBook-Pro pro1 % ./a.out
1 correct e correct 1 correct type:3 if_neg:0%
```

测试样例：1e-1

```
(base) hyx13701490089@huayuxiaodeMacBook-Pro pro1 % ./a.out
1 correct e correct - correct 1 correct type:3 if_neg:0%
```

2. 非法的输入：

测试样例：a

```
(base) hyx13701490089@huayuxiaodeMacBook-Pro pro1 % ./a.out
a error type:0 if_neg:0%
```

测试样例：1.1.1

```
(base) hyx13701490089@huayuxiaodeMacBook-Pro pro1 % ./a.out
1 correct . correct 1 correct . error 1 correct type:0 if_neg:0%
```

测试样例：-1-1

```
(base) hyx13701490089@huayuxiaodeMacBook-Pro pro1 % ./a.out
1 correct - error 1 correct type:0 if_neg:1%
```

测试样例：1e1.1

```
(base) hyx13701490089@huayuxiaodeMacBook-Pro pro1 % ./a.out
1 correct e correct 1 correct . error 1 correct type:0 if_neg:0%
```

测试样例：1e1e

```
(base) hyx13701490089@huayuxiaodeMacBook-Pro pro1 % ./a.out
1 correct e correct 1 correct e error type:0 if_neg:0%
```

测试样例：1e1-1

```
(base) hyx13701490089@huayuxiaodeMacBook-Pro pro1 % ./a.out
1 correct e correct 1 correct - error 1 correct type:0 if_neg:0%
```

代码如下：

1. if_neg方法，用来判断正负（已弃用）：

```
//判断是否为负数：true为负数，false为正数
bool if_neg(string test_str){
    //分割出每个字符
    int len_str=test_str.length();
    char test_char [test_str.length()];
    strcpy(test_char, test_str.c_str());
    bool if_neg=false;
    int start=0;
    if(test_char[0]=='-'){
        if_neg=true;
    }
    return if_neg;
}
```

2. type方法，用来判断数字类型（已弃用）：

```
//判断输入类型：0为不合法，1为整数，2为小数，3为科学计数
int type(string test_str){
    //引入总的判断,true代表没问题
    bool if_correct=true;
    //分割出每个字符
    int len_str=test_str.length();
    char test_char [test_str.length()];
    strcpy(test_char, test_str.c_str());
    //char test_char[len_str]=test_str.split("");
    //test:
    // for(int i=0;i<len_str;i++){
    //     cout << test_char[i] << " ";
    // }
    //判断是否为负数，如果是，检查从第二位开始
    int start=0;
    if(test_char[0]=='-'){
        start=1;
    }else{
        start=0;
    }
    //test:
    //cout << start << endl;
    int len_char=sizeof(test_char);
    //引入判断是否为小数点
    bool expect_dot=true;
    //引入判断是否为e
    bool expect_e=true;
    //引入判断是否为负号，默认false
    bool expect_neg=false;
    //引入判断数字类型：0为错误，1为整数，2为小数点，3为科学计数
    int type=1;


    //判断每一位是否为数字
    for(int i=start;i<len_char;i++){
        //test_char[i]=getchar();

        //test:
        //cout << test_char[i] << " ";
        if(isdigit(test_char[i])){
            //判断数字

            //test:
            //cout << "correct" << " ";
        }else if((test_char[i]=='.')&&(expect_dot)){
```

```
            //判断小数点
            expect_dot=false;
            type=2;
            //test:
            //cout << "correct" << " ";

        }else if((test_char[i]=='e')&&(expect_e)){
            //判断科学计数法
            expect_dot=false;
            expect_e=false;
            type=3;
            if((test_char[i+1]!='-')){
                expect_neg=false;
            }else{
                expect_neg=true;
            }
            //test:
            //cout << "correct" << " ";
        }else if((test_char[i]=='-')&&(expect_neg)){
            //判断科学计数负数
            expect_neg=false;
            //test
            //cout << "correct" << " ";
        }else{
            type=0;
            //test:
            //cout << "error" << " ";
        }
    }
    //test:
    //cout << "type:" << type << " ";
    //cout << "if_neg:" << if_neg;
    return type;
}
```

---

3. judge方法，是type方法的改进，代替了type方法：

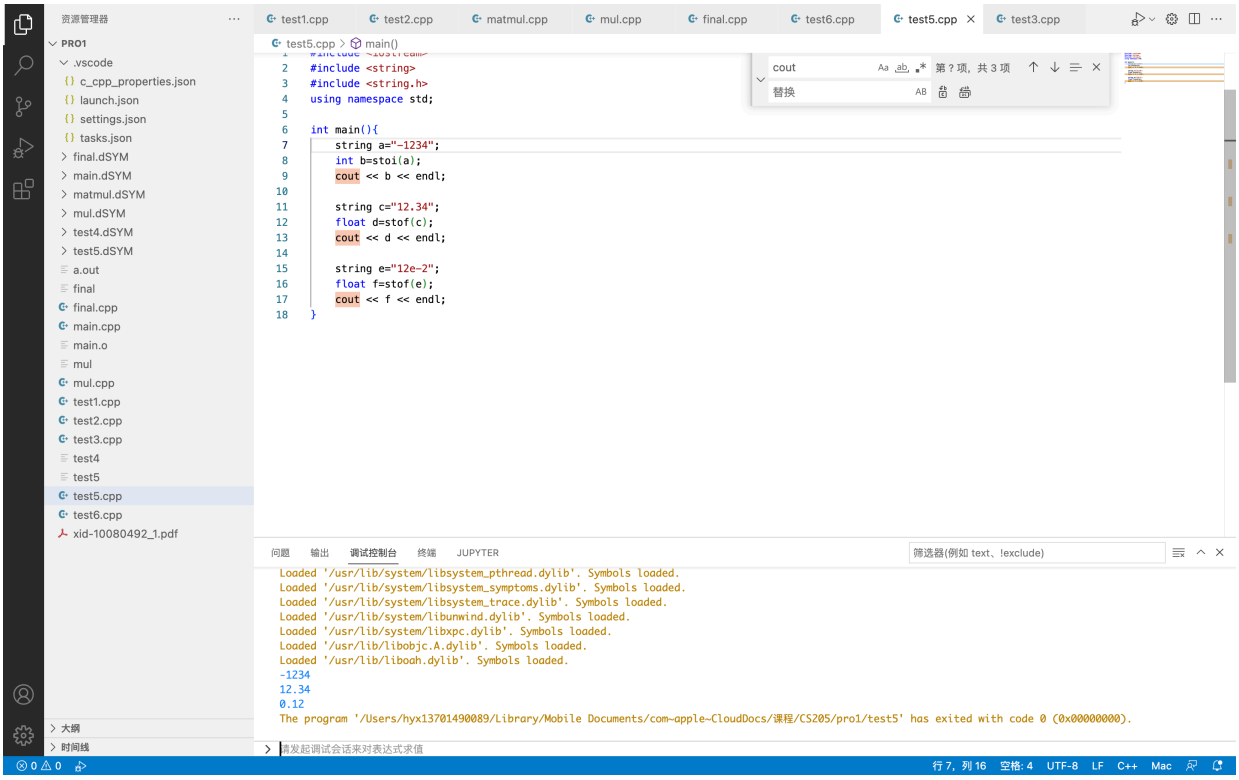
```
//判断输入类型：false为非法，true为合法
bool judge(string test_str){
    //引入合法判断,true代表非法
    bool judge=false;
    //分割出每个字符
    int len_str=test_str.length();
    char test_char [test_str.length()];
    strcpy(test_char, test_str.c_str());
    //char test_char[len_str]=test_str.split("");
```

```
//test:
// for(int i=0;i<len_str;i++){
//     cout << test_char[i] << " ";
// }
//判断是否为负数，如果是，检查从第二位开始
int start=0;
if(test_char[0]=='-'){
    start=1;
}else{
    start=0;
}
//test:
//cout << start << endl;
int len_char=sizeof(test_char);
//引入判断是否为小数点
bool expect_dot=true;
//引入判断是否为e
bool expect_e=true;
//引入判断是否为负号，默认false
bool expect_neg=false;
//判断每一位是否为数字、小数点、e
for(int i=start;i<len_char;i++){
    //test_char[i]=getchar();

    //test:
    //cout << test_char[i] << " ";
    if(isdigit(test_char[i])){
        //判断数字

        //test:
        //cout << "correct" << " ";
    }else if((test_char[i]=='.')&&(expect_dot)){
        //判断小数点
        expect_dot=false;
        //test:
        //cout << "correct" << " ";

    }else if((test_char[i]=='e')&&(expect_e)){
        //判断科学计数法
        expect_dot=false;
        expect_e=false;
        //判断e之后是否为负号
        if((test_char[i+1]!='-')){
            expect_neg=false;
        }else{
            expect_neg=true;
        }
        //test:
        //cout << "correct" << " ";
```

```
            //cout << "correct" << " ";
        }else if((test_char[i]=='-')&&(expect_neg)){
            //判断科学计数负数
            expect_neg=false;
            //test
            //cout << "correct" << " ";
        }else{
            return true;
            //test:
            //cout << "error" << " ";
        }
    }
    //test:
    //cout << "judge:" << judge << " ";
    //cout << "if_neg:" << if_neg;
    return false;
}
```

二、string输入转换

我在网上查到了一种能将string类型的数字，转换成int或者float类型的方法，名叫sto，后缀i或者f分别对应int或者float。于是我修改了之前type方法，把数据类型从4种精简成了2种：0为合法，1为float。测试样例如下：



代码如下：

1. multiply方法，未考虑太大或太小的数（已弃用）：

```
float multiply(float a, float b){
     float result = a * b;
    return result;
}
```

2. main方法，程序的主入口：

```
int main(){
    string a_input="";
    string b_input="";
    cout << "input a:" <<endl;
    cin >> a_input;
    cout << "input b:" <<endl;
    cin >> b_input;
    cout << "result:" << endl;
    bool judge_a=judge(a_input);
    bool judge_b=judge(b_input);
    //test:
    //cout << "judge:" << judge_a << endl;
    //test:
    //cout << "judge_a: " << judge_a << endl;
    //cout << "judge_b: " << judge_b << endl;
    //if_error为false，输入不合法
    bool if_error = judge(a_input) || judge(b_input);
    //test:
    //cout << "if_error: " << if_error << endl;
    if(if_error){
        cout << "error" << endl;
        return 0;
    }
    //初始化数字变量
    float a_float=stof(a_input);
    float b_float=stof(b_input);
    cout << multiply(a_float,b_float) << endl;
    return 0;
}
```

三、最终测试

测试样例：2 * 3

```
(base) hyx13701490089@huayuxiaodeMacBook-Pro pro1 % ./a.out
input a:
2
input b:
3
result:
6
```

测试样例：3.1416 * 2

```
(base) hyx13701490089@huayuxiaodeMacBook-Pro pro1 % ./a.out
input a:
3.1416
input b:
2
result:
6.2832
```

测试样例：3.1415 * 2.0e-2

```
(base) hyx13701490089@huayuxiaodeMacBook-Pro pro1 % ./a.out
input a:
3.1415
input b:
2.0e-2
result:
0.06283
```

测试样例：a * 2

```
(base) hyx13701490089@huayuxiaodeMacBook-Pro pro1 % ./a.out
input a:
a
input b:
2
result:
error
```

测试样例：1234567890 * 1234567890

```
(base) hyx13701490089@huayuxiaodeMacBook-Pro pro1 % ./a.out
input a:
1234567890
input b:
1234567890
```

```
result:
1.52416e+18
```

测试样例：1.0e200 * 1.0e200

```
(base) hyx13701490089@huayuxiaodeMacBook-Pro pro1 % ./a.out
input a:
1.0e200
input b:
1.0e200
result:
libc++abi: terminating with uncaught exception of type std::out_of_range: stof: out of range
zsh: abort      ./a.out
```

```
(base) hyx13701490089@huayuxiaodeMacBook-Pro pro1 % ./a.out
input a:
1e200
input b:
1e200
result:
inf
```

我们观察到，当数据过大时，float已经无法满足最大最小值，于是我改用了double。当运算结果超过数据类型边界时，程序不会报错，而是输出inf。

代码如下：

```
double multiply(double a, double b){
    double result = a * b;
    return result;
}


//初始化数字变量
    double a_double=stod(a_input);
    double b_double=stod(b_input);
    cout << multiply(a_double,b_double) << endl;
```

四、心得体会

这次作业是我第一次入门C++写的作业，写了很多很多版本，最终版本已经是7.0了，之前也试过很多种办法，不过到最后都失败了。算上已弃用的版本，大概有1000多行代码了，每一个单词，每一个字母都是辛辛苦苦敲出来的。这次作业涉及到很多课堂上讲过或没讲过的知识点，我对数据类型、循环结构、方法函数等等等等，都有了更加深入的了解。“纸上得来终觉浅，绝知此事要躬行。”上课讲的内容听听感觉很简单，但一定要等到实操才有体会。另外我感觉我写的代码还是比较优雅的，没有什么废话，结构也很简明清楚，老师您觉得呢？