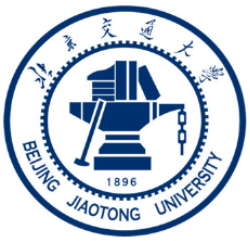

北京交通大學

BEIJING JIAOTONG UNIVERSITY

《软件综合训练》

实验报告

实验名称:	多功能计算器
学号:	
姓名:	
学院:	计算机与信息技术学院
日期:	2022 年 7 月 10 日

目录

一、 需求分析

1. 问题描述

许多时候，我们在学习研究中会遇到很大的数据，这样的数，早已超过了一般编程语的内置数字类的数值范围。对这样的数，我们在本文中称作大数，对于大数之间的运算，我们称作大数运算。这种运算直接调用一般的运算符和运算函数，会有溢出的错误，因此我们需要采用全新的算法，进行大数运算。

2. 功能需求

按照实验要求，我们要实现大数之间的基本四则运算，拓展运算要完成大数的 sin、cos 三角函数运算、平方和立方、不同进制数之间的转换等功能。基本要求是对于大数运算，不能调用有关大数运算的库函数进行编程。还可以完成诸如 x 的 y 次方和 log10（x）这样的复杂运算。

本实验核心要求就是编写大数的运算函数。

实验要求基本的演示界面。

3. 开发环境

系统：windows10
语言：Cpp
开发环境：Clion 2022、MinGW 10.0

4. 开发过程

本次实验课，一波三折。

周一到周三，在第一题 SA 上死磕，成功完成了 SA 软件在 QT5.9 的调试和升级工作。程序可以正常运行，相关函数调用正常，测试样例符合预期。

但是这样的结果对于整体实验要求相差甚远，几经思度之下，无奈在周四上午改换成了第三题计算器进行编码。所有的函数编写和测试，包括基础要求、拓展运算、加分项目，都在周四一天完成了，周五上午主要是编写了一个测试用的菜单函数，周五下午进行了答辩。

周六日完成了实验报告的编写。

二、 概要设计

1. 总体设计

设计思路如下:

大数运算的基础在于加减乘除运算，加减乘除运算又在于加减运算。

分为符号计算和数值计算两部分，符号计算在菜单函数中完成，其他由各自运算对应的函数完成。

加减运算，采用了 string 类进行保存数据，所有的数采用字符串的形式存储。运算时，一位一位的进行运算，转为数字进行加减和进位借位之间的操作，结果保存为 string 每次运算一位保存一位。在此基础上，我们采用分段计算的方法，分为小数部分、较小的数、较大的数三段进行运算。注意，在编写加法函数时，采用了整数运算和小数运算分开计算的做法，但是在后边的运算中，统一把整数变成小数进行运算，不再分类讨论。

乘法运算。采用较小的数作为乘数，较大的数作为被乘数，按位进行乘法运算，即为按位按权值调用加法函数进行反复加，结果保存在 vector<string> 中。最后把向量组中的数全部加起来，得到最后结果。

除法运算。整体结果分为两次计算，先计算出商的整数部分和余数，再对余数进行运算得到商的小数部分。商、余数、小数，打包成一个结构体放回。重大向小运算，主要是调用减法不断试商，发现差比被除数小，即是试商成功，结果保存为 string。通过控制迭代次数防止死循环，也可以进而控制小数精度。

平方、立方运算，直接调用乘法函数运算得到结果。

Sin、Cos 函数采用周期性质，调用除法，以高精度 2pi 作为除数，得到余数，再得到结果。大数变小数进行运算，大大减少了运算量。

进制转换核心是二进制与十进制的转换。二进制转十进制，重新编写了配套的二进制减法二进制除法，进行除进制数“1010”运算。得到的每次运算的余数，通过映射函数对应为 10 进制，最后的结果逆序就是在十进制下的结果。十进制转二进制，调用十进制除法，其他同上述。二进制转四、八、十六进制运算，主要利用性质，采取两位变一位、三位变一位、四位变一位的映射函数即可完成，适当的前补 0 可以正则化。逆运算为一位变多位，只需要编写逆映射函数即可。其他进制之间的转换主要是以二进制为桥梁，再转换。

幂运算。通过把指数分为，整数部分和小数部分，整数部分调试乘法迭代运算，小数部分通过正则化，截取、近似等等，直接运算，做到减小误差。还加入数值大小判断机制，偏小可以直接计算，分为两套计算思路。总体来说这样的算法效率不高。

自然对数运算。通过对数函数性质，对真数进行处理，转换为科学计数法形式，转乘法变加法，对于剩下的对于运算采用截取的方法直接计算出一个很小的结果在和另一部分加和得到结果。这里我们可以在科学计数法转变时，对于有效数字可以转换为 10.xxxx 的形式而非 1.xxxxx 形式，考虑到对数函数的斜率问题，随着自变量增大快速减小，这样得到的一个小结果误差会更小。

数据预处理部分在菜单函数中完成，包括符号的处理，防止除数为 0，对数函数定义域限定问题。对于前导 0 和后缀 0 的数字表示正则化，不再详细叙述。

2. 类的设计

主要使用 cpp STL 中自带的 String 和 vector 类，这样简单快捷，安全性高。

在除法和进制转化为设计了简单的结构体最后结果的放回，他们只重写了构造函数，其他的基本函数全部默认，形式非常简单。

说明	结构体定义

除法返回结构体	struct divide_result{
	divide_result(string &q, string &r, string &q_p);
	string quotient, reminder, quotient_pot;
	};

二进制转换为其他进制结构体	struct two_shift{
	string four, eight, ten, sixteen;
	two_shift(string &four_, string &eight_, string &ten_, string &sixteen_);
	};

四进制转换为其他进制结构体	struct four_shift{
	string two, eight, ten, sixteen;
	four_shift(string &two_, string &eight_, string &ten_, string &sixteen_);
	};

八进制转换为其他进制结构体	struct eight_shift{
	string two, four, ten, sixteen;
	eight_shift(string &two_, string &four_, string &ten_, string &sixteen_);
	};

十进制转换为其他进制结构体	struct ten_shift{
	string two, four, eight, sixteen;
	ten_shift(string &two_, string &four_, string &eight_, string &sixteen_);
	};

十六进制转换为其他进制结构体	struct sixteen_shift{
	string two, four, eight, ten;
	sixteen_shift(string &two_, string &four_, string &eight_, string &ten_);
	};

表 1 类的设计

3. 运行界面的设计

这一点也是本次实验最让人感到无奈的一点。由于系统问题，本人的笔记本无法安装 Qt6，而一开始完成 SA 的调试是在可以安装 Qt5 的老旧台式机上完成的，在时间紧急的情况上，本人无法在编译运行调试上花费太多时间，于是没有办法在短时间内完成 GUI 的设计。最终呈现的结果是一个 Menu 函数列出 16 个功能的命令行菜单形式。

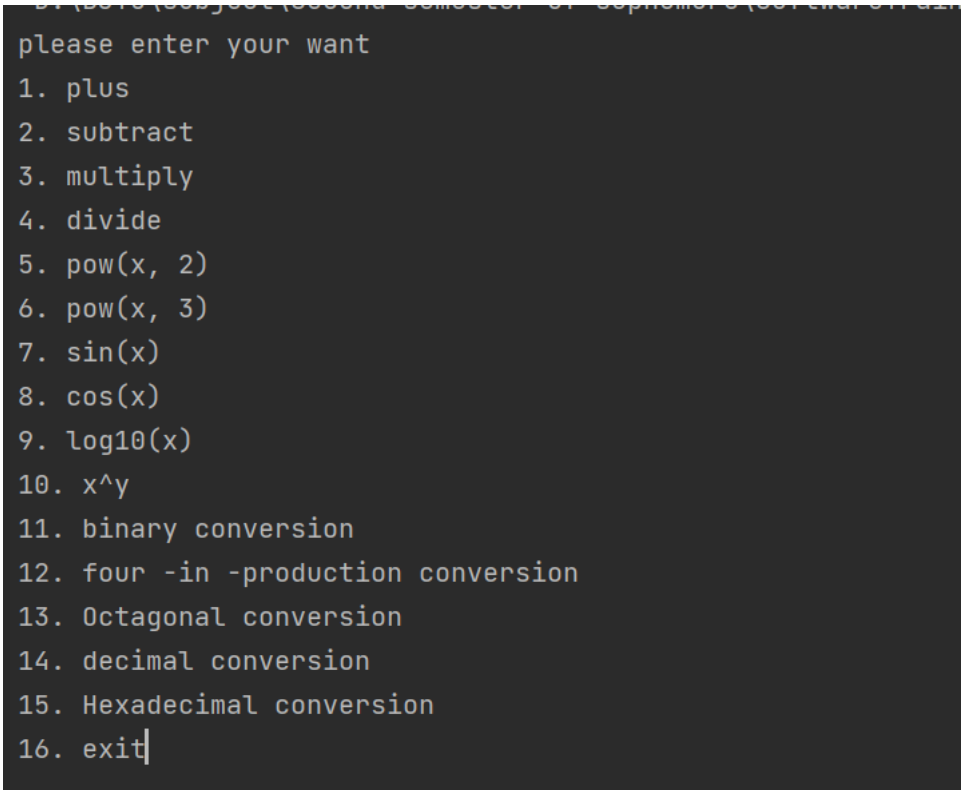

图 1 菜单函数运行界面

三、 详细设计

1. 重要函数原型

设计思路见概要设计。

1. 加法
   string add(string a, string b)
   a b 为两个加数，return string 结果
2. 减法
   string subtraction(string a, string b)
   a 被减数，b 减数，return string 结果
3. 乘法
   string multiply(string a, string b)
   a b 乘数，return string 结果
4. 除法
   divide_result divide(string a, string b, int control_num)
   a 被除数，b 除数，divide_result 包含商、余数、带小数的商的结构体，
   control_num 精度控制
5. Sin 函数
   double sin_my(string a, int control_num)
   a 弧度，control_num 精度控制

6. Cos 函数
   double cos_my(string a, int control_num)
   a 弧度，control_num 精度控制
7. 平方
   string square_my(string a)
   a 待求变量
8. 立方
   string third_square_my(string a)
   a 待求变量
9. 二进制转换
   two_shift binary(string a)
   a 二进制数，two_shift 包含所有进制转换结果的结构体
10. 四进制转换
    four_shift four(string a)
    a 四进制数，four_shift 包含所有进制转换结果的结构体
11. 八进制转换
    eight_shift eight(string a)
    a 八进制数，eight_shift 包含所有进制转换结果的结构体
12. 十六进制转换
    sixteen_shift sixteen(string a)
    a 十六进制数，sixteen_shift 包含所有进制转换结果的结构体
13. 十进制转换
    ten_shift ten(string a)
    a 十进制数，ten_shift 包含所有进制转换结果的结构体
14. X 的 y 次幂
    string x_power_y(string x, string y)
    x 底数，y 指数
15. 自然对数函数
    string logarithm10(string x)
    x 真数

2. 重要函数流程图

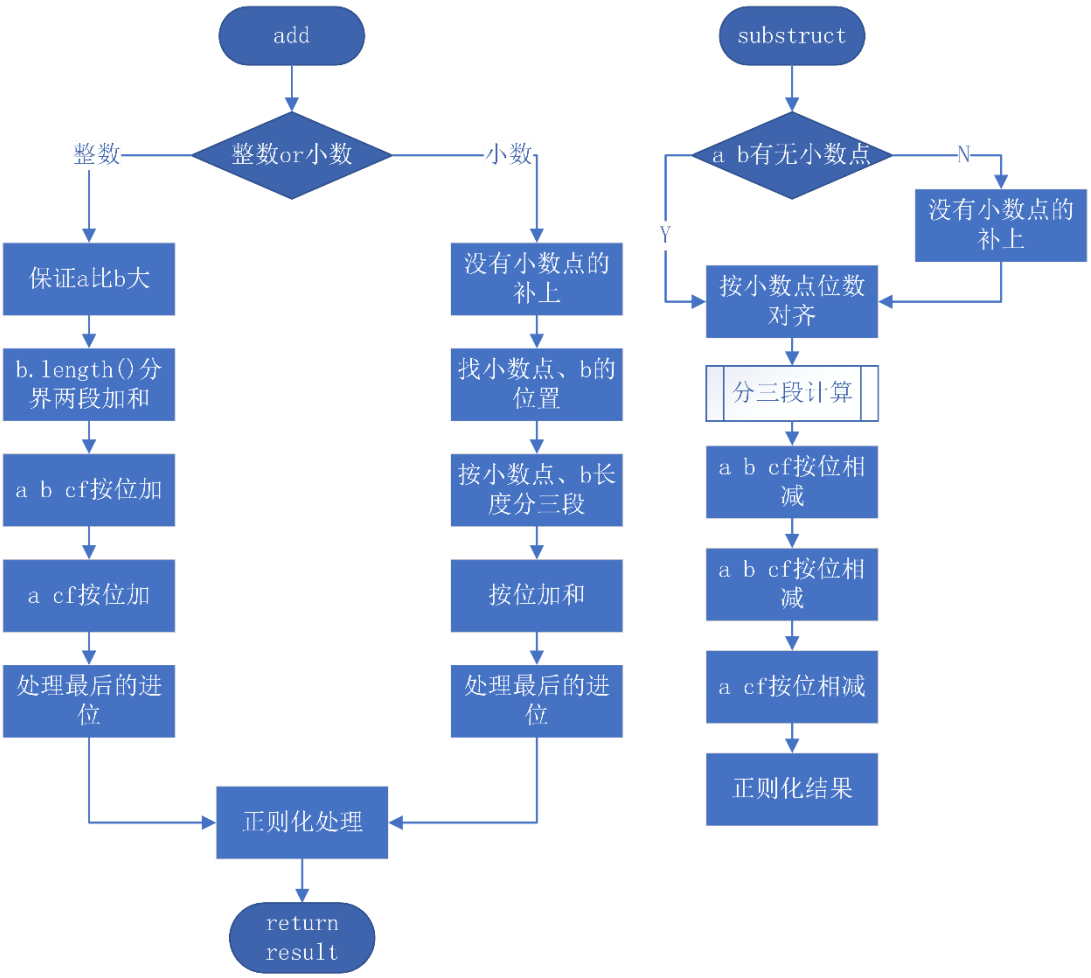

图 2 add（）substruct（）

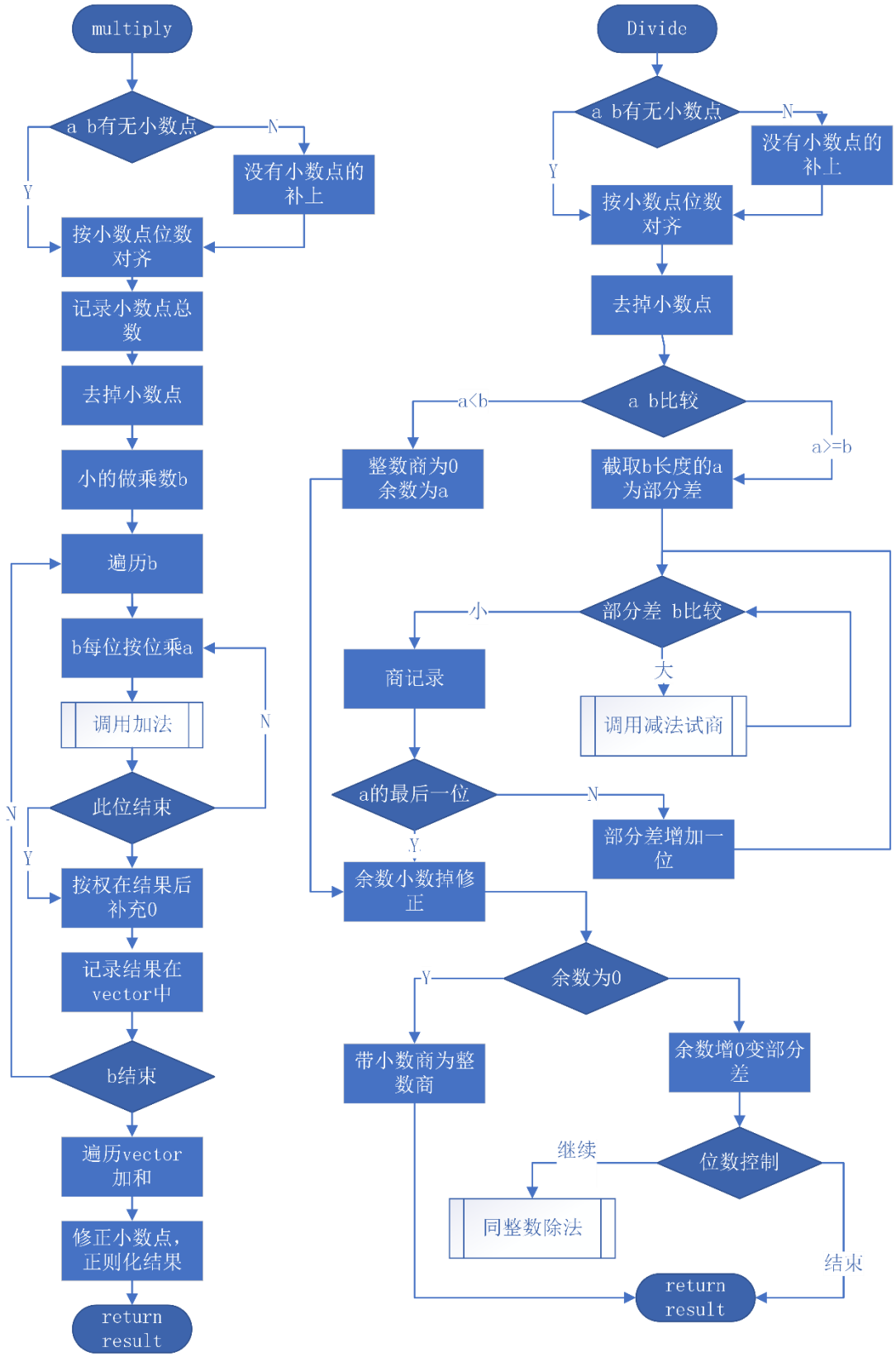

图 3 multiply（） divide（）

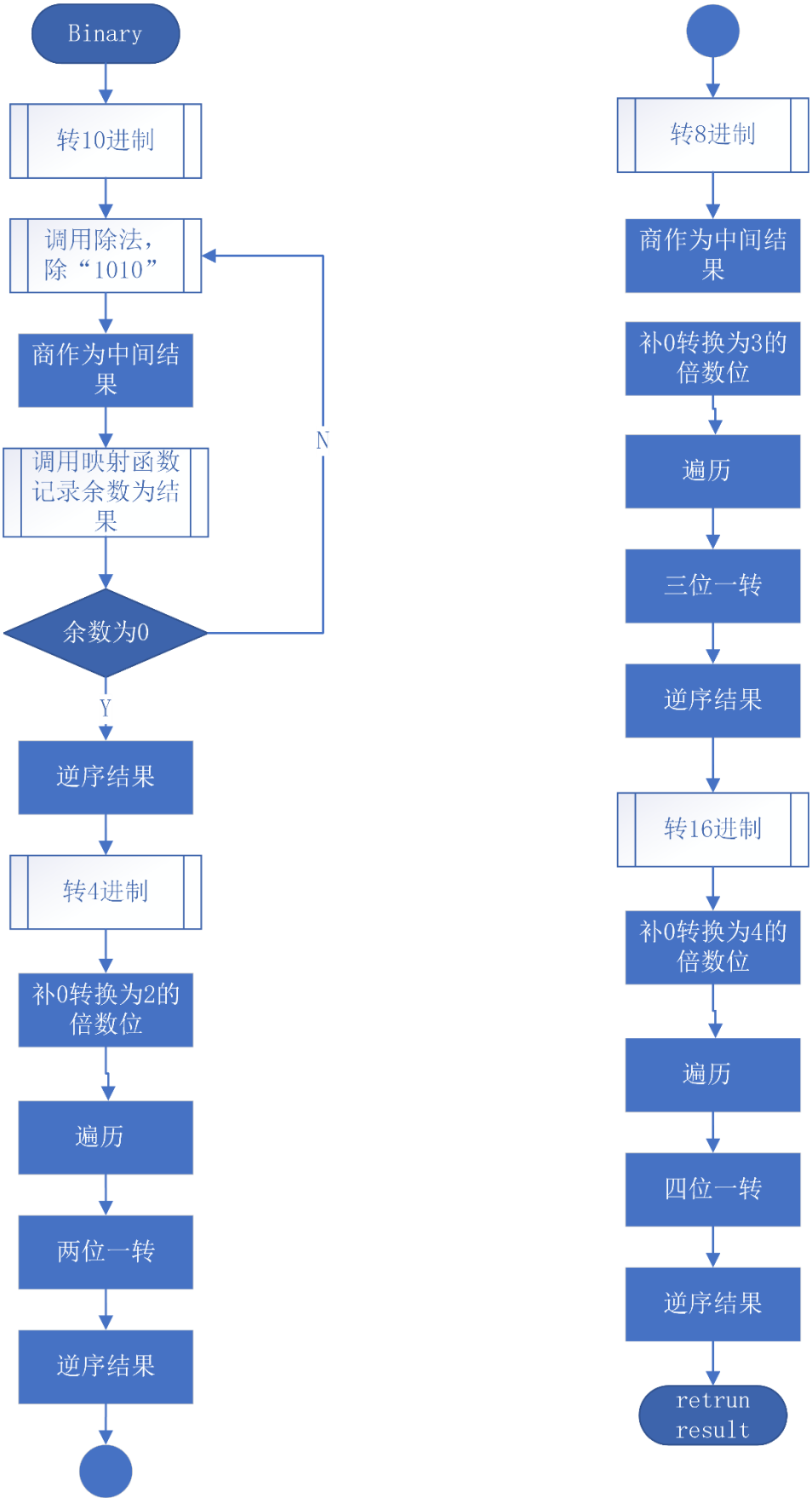
Binary
转10进制
调用除法，除“1010”
商作为中间结果
调用映射函数记录余数为结果
余数为0
N
Y
逆序结果
转4进制
补0转换为2的倍数位
遍历
两位一转
逆序结果
转8进制
商作为中间结果
补0转换为3的倍数位
遍历
三位一转
逆序结果
转16进制
补0转换为4的倍数位
遍历
四位一转
逆序结果
retrun result

图 4 binary（）

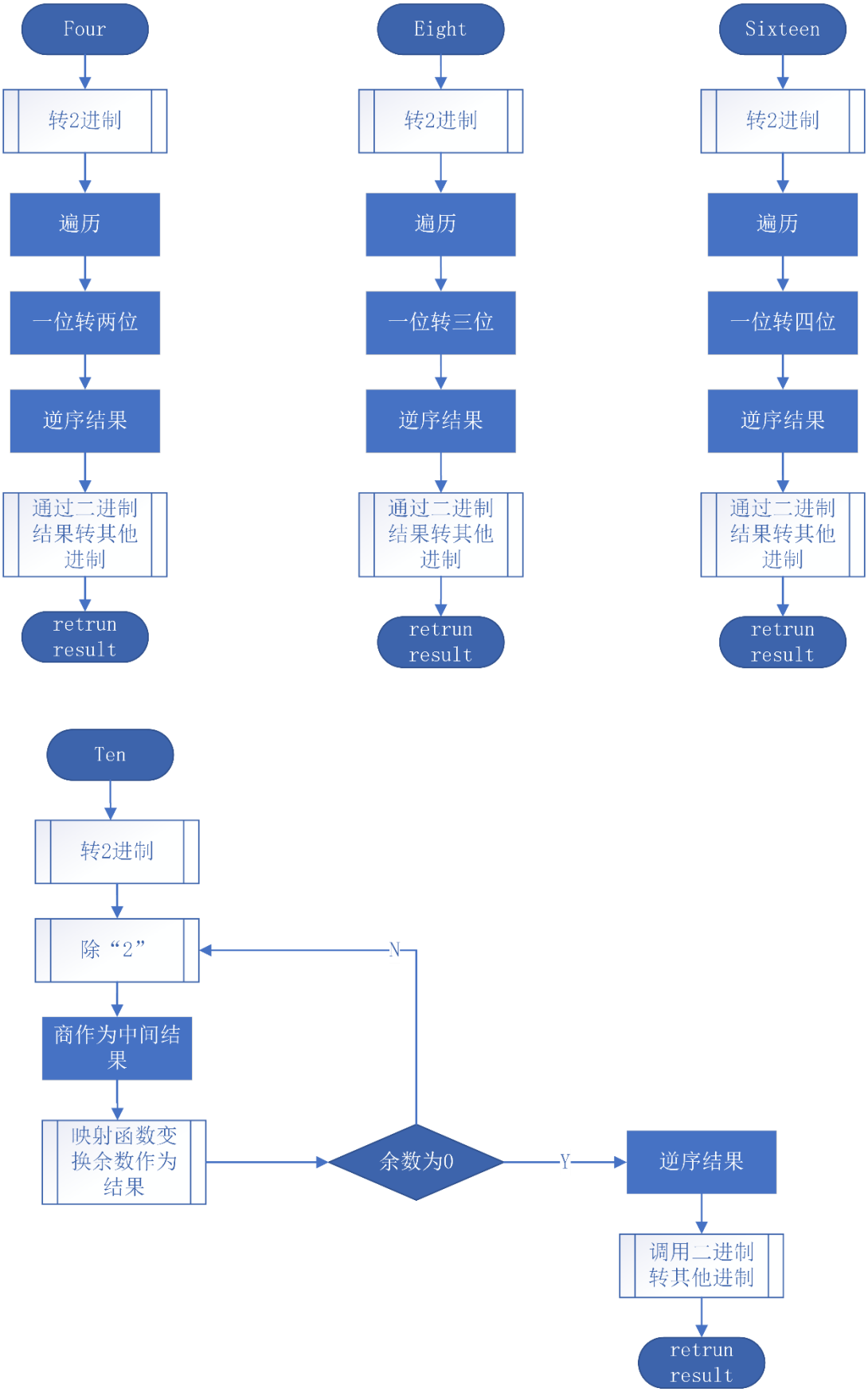

图 5 four（） eight（） ten（）

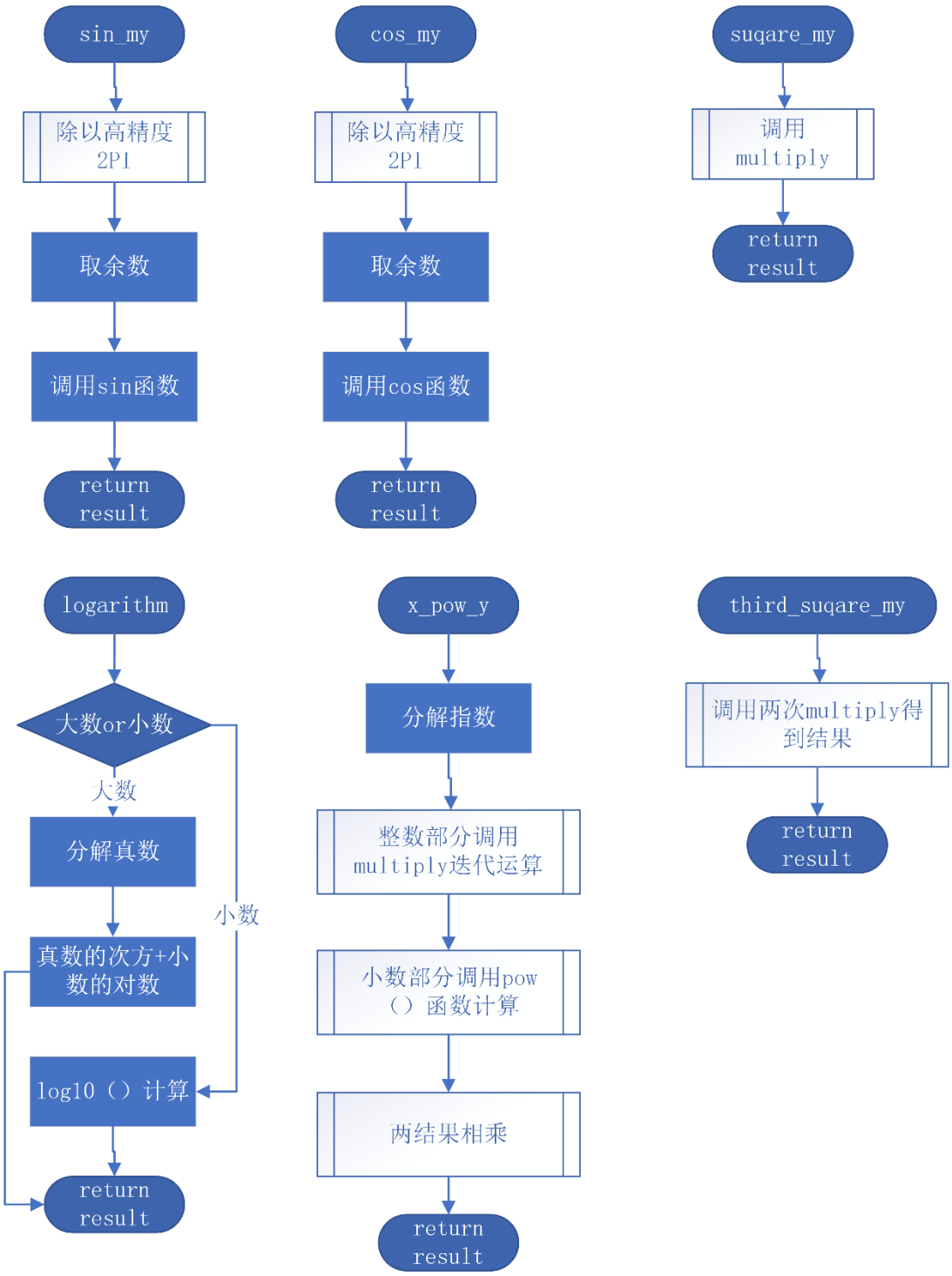

图 6 sin_my（ ） cos_my（） logarithm（） square_my（） third_of_my（） x_pow_y（）

四、 测试分析

测试主要对计数结果正确性和精度以及运算速度进行测试。对于 sin_my、cos_my、logarithm、x_pow_y 要求有一定精度即可，其他函数要求结果完全正确。

测试用例随机操作键盘输入，标准结果由 python3.7 和网页进制转换工具得出，最后

人工比对。

```
1
please enter two number
4657876578978681 269876567987677
4927753146966358
please enter your want
```

```
2
please enter two number
5456754567565778.236786786776 76785677866.67867
5456677781887911.558116786776
please enter your want
```

```
3
please enter two number
768798679778767678978 6.57866865767665 7686877968767.09867
5909661634008640830487590479803862 0.8555307216435050555
please enter your want
```

```
4
please enter two number
65786879876778.9089192083068949 27032071023721.2321307
please enter the number of decimal your want
10
```

```
5
please enter the num
32849032509327
10790589367988221023419929 29
please enter your want
```

```
6
please enter the num
2817329074023727
22362107518078414713586708195756639938039741583
please enter your want
```

```
7
please enter the num
374234093284029.340293
-0.999434
please enter your want
```

```
8
please enter the num
890289302.2323
0.112201
please enter your want
```

```
9
please enter the num
217301701242122.2313
14.337063
please enter your want
```

```
10
please enter the num x y
23820.2323
2327
1423392088808973886318360347594909189619674951375
```

```
11
please enter the num
101101011010101111111
2 to 8: integer 5532577
2 to 16: integer 16B57F
 to 4 11223111333
 to 8 5532577
 to 10 1488255
 to 16 16B57F
```

```
12
please enter the num
11223111333
4 to 2: integer 10110101101010111111
2 to 8: integer 5532577
2 to 16: integer 16B57F
 to 2 10110101101010111111
 to 8 5532577
 to 10 1488255
 to 16 16B57F
```

```
13
please enter the num
5532577
8 to 2: integer 10110101101010111111
2 to 8: integer 5532577
2 to 16: integer 16B57F
 to 2 10110101101010111111
 to 4 11223111333
 to 10 1488255
 to 16 16B57F
```

```
14
please enter the num
1488255
2 to 8: integer 5532577
2 to 16: integer 16B57F
 to 2 10110101101010111111
 to 4 11223111333
 to 8 5532577
 to 16 16B57F
```

```
15
please enter the num
16B57F
16 to 2: integer 10110101101010111111
2 to 8: integer 5532577
2 to 16: integer 16B57F
 to 2 10110101101010111111
 to 4 11223111333
 to 8 5532577
 to 10 1488255
```

图 7 计数器原始测试样例 共 15 张 组图

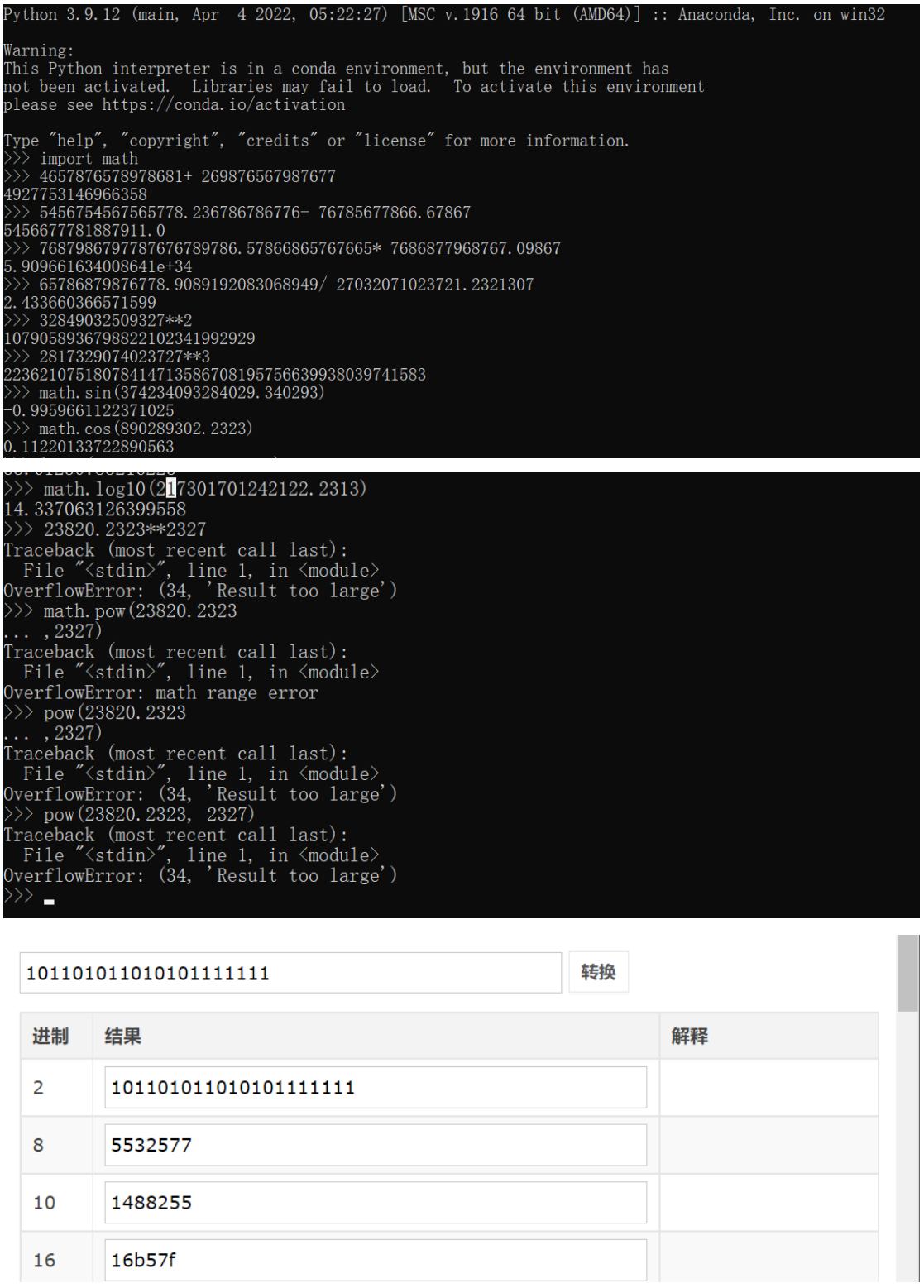

图 8 检验结果 共 3 张 组图

可测试结果完全正确。

需要说明的是幂函数测试中，结果过于庞大，无法截图，最后所有的测试结果以文本形式提交，且此数据 python 的内置函数无法运算，本计算器计算时间约为 20s，可以接受。

Sin_my cos_my 函数精度当 22 位数运算时可以保证 2 位有效数字，14 位数运算时可以

保证 6 位有效数字（迭代结果方位内完全正确）。

对数函数在 20 位数运算时，迭代结果范围内完全正确。

五、 总结体会与创新点

1. 总结体会

总的来说，本次实验收获还是非常大的。

虽然说一波三折，经历了很多挫折，但是有问题，有困难解决了才有进步。特别是一开始在完全没有接触过软件的跨平台移植的情况下，对于 sa 的编译工作完全无从入手。于是从头开始学习，掌握基本的编译原则，一步一步的调试，最后把 sa 在自己的平台上运行起来，还是特别兴奋的。

但是对于接下来的加入函数，还有内部接口的调用工作就无能为力了。在这个时候我做出了大胆而果决的决定，最终转向了计算器的设计研究。最后虽然只花了一天时间，但是代码算法完成度很高、运算效率还有算法有效性都令人满意。

特别是在这次的编程中，把之前学习到许多软件算法方面的设计思路，测试方法使用上了。以前写算法总是一边写一边改一边调，总体效率不高。现在先画出各种流程图，完成相应的变量设计，在纸上修改一定的代码，再在计算机上完成编码，编写的代码简洁美观 Bug 少。特别是运用了黑盒测试技术和白盒测试技术，快速定位了自己的 bug，以前有的 bug 甚至一天都调不出来，现在分分钟就可以知道位置。还有大量的采用了标准编码原则，加入了大量的注释。这样对于后来层层函数调用出现 bug 时的函数复查非常有效。

在这些方面，在这次实验中都收获了很多，相当于对整体的软件编码能力有了一个总体的训练。当然也有很多不足的地方，比如说因为我系统的问题，无法安装 qt6。（现在初步认定是联想的出场基带系统有一些问题，稍后会进行系统重装，进行验证）从而没有设计出比较美观的界面，只有一个命令行界面。但是考虑到时间紧，任务重，仅仅花了一天的时间就完成了所有的编码，并在星期五进行了答辩，答辩效果十分令人满意。总体来说，这次实验课收获是非常丰富的。

2. 创新点

实验的创新点。

幂函数方面，运算速度其实还是总体上比较令人满意的。但是在精度上其实没有办法保证。这方面其实考虑到软件算法优化的局限性问题，目前的主流思路还是在硬件上有所优化，这一点是超出了能力的范围。

本来只是想着这次可以在比较成熟的大数运算路线上完成相关的编码，以减少 bug 为主，快速完成整个算法设计，但是没想到编写过程中灵光乍现，探索出了更多的算法创新，在最后的完成度上非常的高，算法的复杂度被大幅下降，有效性大幅提高。

三角函数方面利用了数学三角函数的数学性质进行周期转换，最后在 0~2π或-π到π上再进行运算，比传统直接套用泰勒展开公式进行复杂的乘方和阶乘运算相比，可以运算的范围大幅增加，运算速度有数量级的进步。真正完成了三角函数大数运算。

对数函数中采用了转换成科学计数法的方法进行真数分离，总体思路是大数变小数，

乘法变加法。特别是我们在分离时不采用第一位为 1 到 9 的科学计数法表示而采用第一位为 10 到 99 的科学技术法表示，利用了对数函数的数学性质，其导数在 y=1 之后快速变小。这样可以非常有效的减小误差，上面的测试样例已经说明了在我们的迭代周期内数值计算是完全正确的。