题目要求

第一个题目要求利用位运算的知识

并通过位运算的方式完成十进制加减乘法；

第二个题目要求计算1000！，结果超过两千位，远超int或者long long所能储存的值，应使用高精度乘法算法进行计算。

主要流程及解决思路

第一题：位运算共包括六种：与（&），或（|），非（~），异或（^），左移（<<），右移（>>）。左移和右移运算可进行快速乘2，除2操作。

a	b	a&b	a|b	~a	a^b
0	0	0	0	1	0
0	1	0	1	1	1
1	0	0	1	0	1
1	1	1	1	0	0

根据该真值表可以进行位运算等操作。

计算机通过补码进行运算，即无需考虑加负数的问题。根据上学期的数字逻辑知识，可知完成十进制加法器可以通过异或运算计算本位（异或运算当且仅当a！=b时值为1），可计算出所有本位。再通过与运算计算出进位情况，当且仅当a == 1 && b == 1时值为1，符合运算进位的需求，此时进位保留在本位，应使用左移运算，完成进位操作，再循环进行进位与异或计算答案的异或，直到进位为0；

减法运算，通过补码运算法则，对数据进行取补码运算先取反再加1，然后进行加法运算。乘法运算需要先判断a和b的正负，并转为正数，a*b，b个a相加，通过循环，完成b个a的相加运算，最后进行正负的判断。

第二题：高精度计算，通过int数组，每个int存一位数，通过-1指示该位是否使用。

高精度模拟了竖式计算法，完成按位相加后，需要维护数组的值在0到9之间。

为操作方便，数据逆序存储，累乘操作需保存结果且乘数每次循环加1，完成后逆序输出。

程序难点以及我遇到的问题

第一题：之前未曾学习有关位运算实现的相关知识，只有去年学习的数字逻辑电路的知识进行辅助，未了解C++通过补码计算的规则，在进行加法运算时错误认为需要提前判断正负，导致减法运算时没有头绪。仔细完成相关知识学习之后，该问题迎刃而解。

第二题：高精度加法和乘法的运用。

程序的优缺点

第一题：设置了较为合理的操作选项，完成了六种位运算封装并测试和三种十进制运算，使用迭代计算，减少了递归调用的时间损耗，减少堆栈的占用。

第二题：使用int存位，使函数书写变得简洁，使用-1进行标识，方便了数位的判断，但使用memset()造成了时间的浪费。采用定长数组，在计算较小数的阶乘时，会造成空间的浪费。使用int存位相较于char存位，也造成了更大的浪费，若使用vector或者是string将进一步减少相关浪费。第一版将两个乘数全部使用高精度进行运算，浪费了空间和时间。在第二版中考虑到只有计算阶乘过程中第二个数不会出现爆int的情况，把第二个数直接以int形式乘到高精度数组中，然后进行数组维护，使程序更为简洁，提高了程序的效率。

我的收获

进一步学习了位运算，学习C/C++中接近底层的操作。巩固了高精度算法的运用，在编程过程中可以更多的考虑空间和时间的占用。在debug过程中有效利用IDE自带调试功能，以及在关键位置输出结果的方式，进行了函数的测试和使用。在C语言课程设计过程中学习的相关开发知识，进一步得到夯实，由过程化编程，进一步学习了面向对象的思想，巩固了类封装的相关知识。希望今后可以进一步提升自己编程能力，进一步优化程序。

代码实现

https://github.com/LeoMeng86/Cpp_experiment.git