高精度/高精度（扩展）

string a,b;

a/b

从a的高位开始一个一个去处理

t字符串

a="12345" b="15" c="";

遍历a从a中取数放入t

t="1" 如果t>b t=t-b 减几次商几 商进c="0"

t="12" 如果t>b t=t-b 减几次商几 商进c="0"

t="123" 如果t>b t=t-b 减几次商几 商进c="8"; t="3"

t="34" .....

1000以内n的阶乘

```
#include <iostream>
using namespace std;
int s[100000]={1};
int len=1;//当前有效位数
int main(){
    int n;
    cin>>n;
    for(int i=1;i<=n;i++){
        //高精度数字的每一位
        for(int j=0;j<len;j++){
            s[j]*=i;
        }
        for(int j=0;j<len;j++){
            s[j+1]+=s[j]/10;
            s[j]%=10;
        }
        if(s[len]>0){//梳理有没有超出现有位数的情况
            while(s[len]>=10){//高位进位  进位到无位数可以进位的时候停
                s[len+1]+=s[len]/10;
                s[len]%=10;
                len++;
            }
            len++;
        }

        //乘法的答案是多少位
        //
    }
    for(int j=len-1;j>=0;j--){
        cout<<s[j];
    }
```

```
    cout<<endl;


    return 0;
}
```

STL模板库- queue(队列) , set(集合) ， map(映射)

queue

```
// 线性数据结构
// 队列：先进先出
// 栈：后进先出  先进后出

#include <bits/stdc++.h>
#include <queue>//队列头文件
using namespace std;

//容器名<类型> 名字;
//命名规则和其他一致
queue<int> Q;
int t;
int main(){
    Q.push(5);//存入5
    Q.push(1);//存入1
    Q.push(4);//存入4
    // Q.empty()  判断数组是否为空   空为真  非空为假
    // Q.size();  数组元素个数
    while(!Q.empty()){//队列只要有元素 Q.size()>0
        t=Q.front();//取出队首元素（复制取出的，队首没有删除）
        cout<<t<<endl;
        Q.pop();//删除
    }
}
```

指针

```
#include <bits/stdc++.h>
#include <queue>//队列头文件
using namespace std;
//内存中的每一个字节都有自己的编号
//我们可以通过这个编号去找到当前的变量
//这个编号我们称其为地址

//&：引用(取地址符) 取得变量的地址
//计算会考虑存储一个变量的地址：指针类型
//*：指针表示：声明阶段代表这是一个指针变量(*概念上跟随变量名)
//所有指针的空间占用是固定的：4B
//*：使用阶段：解引用：  根据地址取得变量
int main(){
    int a=3;
    int *pa=&a;//pa存a的地址
```

```
    float b=3.14;
    float *pb=&b;
    char c='c';
    char* pc=&c;
    double *x,*y;
    cout<<a<<" "<<*pa<<endl;
    *pa=10;
    cout<<a<<" "<<*pa<<endl;

//  int s[1000]={1,2,3};
//  for(int i=0;i<1000;i++){
//      cout<<*(s+i)<<endl;
//      //s[i] == *(s+i)
//  }
}
```

priority_queue优先队列

```
#include <bits/stdc++.h>
#include <queue>
using namespace std;
//优先队列
priority_queue<int>Q;//大的先出  每一次插入操作 logn
//priority_queue<int,vector<int>,greater<int> > T;//小的先出
//一般用小的在前我们采用大的先出的负数来写
int main(){
    Q.push(-3);
    Q.push(-1);
    Q.push(-5);

    while(Q.size()>0){
        cout<<-Q.top()<<endl;//取出
        Q.pop();
    }
//  T.push(3);
//  T.push(1);
//  T.push(5);
//  while(T.size()>0){
//      cout<<T.top()<<endl;
//      T.pop();
//  }

    return 0;
}
```

set(集合) multiset;

```
#include <bits/stdc++.h>
#include <set> //集合
using namespace std;
multiset<int>t;//可插入重复元素
set<int>s;//整数类型的s集合  可以理解为一个从小到达数组
//每种只能出现一次，不会有重复
int main(){
```

```
    s.insert(5);
    s.insert(3);
    s.insert(3);
    s.insert(1);
    s.insert(4);
    //s.begin() 开始位置
    //s.end() 结束位置
//  set<int>::iterator i;  //标准写法
//  for(i=s.begin();i!=s.end();i++){
//      cout<<*i<<" ";
//  }
    //auto 根据变量赋值的类型自动调整 (声明一定要赋值)
//  for(auto i=s.begin();i!=s.end();i++){
//      cout<<*i<<" ";
//  }
    // 1 3 4 5
    set<int>::iterator pt=s.find(3);//得到3的位置
    //迭代器可以使用 ++,--

//  s.erase(3);//删除3
    for(auto i=s.begin();i!=s.end();i++){
        cout<<*i<<" ";
    }
    cout<<endl;
    //s.clear();//清空
    auto lt=s.lower_bound(3);//找到第一个大于等于3的位置
    auto rt=s.upper_bound(3);//找到第一个大于3的值
    lt++;
    cout<<*lt<<" "<<*rt<<endl;

    s.size();
    s.empty();
}
```

map映射

```
#include <bits/stdc++.h>
#include <map>
using namespace std;

map<string,int>T;//桶T
//map<桶名字类型,桶里存的数据类型> 名字
map<int,int>S;//大小是浮动的
int main(){

}
```