文件IO

当面对输入样例比较多的题目时，在调试过程中，反复手动输入很浪费时间，这时推荐文件IO。在`main`函数前加入如下代码：

```
#ifndef ONLINE_JUDGE
freopen("a.in","r",stdin);
freopen("a.out","w",stdout);
#endif
```

这份代码在本地运行时，会从`a.in`中读取输入，输出到`a.out`中，当代码被复制粘贴到OJ上时，不用任何修改（**这四句不用删**），就会走标准输入输出。

原理在于使用了`ifndef`，OJ的编译命令加了`-DONLINE_JUDGE`选项，在OJ评测时，会检测到已经`define`了`ONLINE_JUDGE`，故不会执行两条`freopen`语句，在本地执行时则一定会执行。

这个方法可能不适用于其他OJ。更灵活也更黑科技的写法如下：

```
#ifdef __Ando
freopen("ando.in","r",stdin);
freopen("ando.out","w",stdout);
#endif
```

这里使用的是`ifdef`，同样只需要在`main`前加入这四句，同样在复制到OJ提交时不需要删掉这部分。实现原理是，我**修改了本地的**`stdio.h`**的内容**，在里面加了一句

```
#define __Ando
```

当我调用库文件时，就会检测到已经`define`了`__Ando`，自动走文件IO，在OJ评测时则检测不到，故会走标准IO。

这种方法做到了**任意一个C/C++代码在本地都走文件IO，复制到任意一个OJ上都走标准IO**。如果想要节省调试时间，且希望可以无脑直接把调试好的代码提交到任何一个OJ上，推荐使用这个方法，`define`的东西不要太常见，最好用自己的ID或者名字，防止对库文件造成干扰。

注意，在修改库文件时，如果使用*VSCode*打开，可能提示权限不够，不允许修改，换成*Dev C++*打开就不会有这个问题。

最后，**改库需谨慎，注意不要误删什么东西。**

关于IO优化

这里给出几份常用的C++快读板子，大家自行食用。**如果要迁移到C语言中，请去掉最开头的inline!**

常规版（int类型，支持负数）：

```
inline int read(){
    int p=0,f=1;    char    c=getchar();
    while (c<48||c>57)    {if (c=='-')    f=-1;    c=getchar();}
    while (c>=48&&c<=57)    p=(p<<1)+(p<<3)+c-48,c=getchar();
    return p*f;
}

// int x = read();
```

高级加速版（int类型，支持负数）：

```
char buf[1<<15],*fs,*ft;
inline char gc(){
    return (fs==ft&&(ft=(fs=buf)+fread(buf,1,1<<15,stdin),fs==ft))?0:*fs++;}

inline int gint(){
    int x=0,f=1,ch=gc();
    while(ch<'0'||ch>'9'){if (ch=='-')    f=-1;    ch=gc();}
    while(ch>='0'&&ch<='9') x=(x<<1)+(x<<3)+ch-'0',ch=gc();
    return x*f;
}

// 使用gint读取数据
// int x = gint();
// 注意，由于实现机制特殊，这个板子只适用于输入只有数字的情况，如果输入中含有字符或字符串需要处理，请勿使用
```

快速输出板子：

```
inline void Write(int x) {
    static int sta[15];
    if (x<0)    {putchar('-');    x=-x;}
      register int top=0;
      do{
        sta[top++]=x%10,x/=10;
      }while (x);
      while (top) putchar(sta[--top]+48);
}
// Write(998244353);
```

关闭C++的cin，cout与stdio的绑定：

```
std::ios::sync_with_stdio(false);
std::cin.tie(0);
// 如果编译开启了 C++11 或更高版本，建议使用 std::cin.tie(nullptr);
```