基础知识

1. 通用头文件 `#include<bits/stdc++.h>`
   这个头文件基本包含了常用的库函数

---

2. 换行用 `'\n'` ，这要比 `endl` 更快

---

3. 取消同步流
   `ios::sync_with_stdio(0),cin.tie(0),cout.tie(0);`
   + 在输入比较多的情况下可以使用
   + 具体效果是在输入全部完成后统一进行输出，不会在每一步结束后都进行输出

---

4. 科学计数法 2*10^5 : `2e5`

---

5. 在全局变量里 `a[N]` 会自动赋初值为0
   在局部变量里 `a[N]` 不会自动赋初值为0
   数组可以定义在全局变量里:
   - 不用进行初始化，会自动赋值为0
   - 如果将数组定义在局部变量里，占用的是栈的空间，如果开的数组过大（比如超过1mb），程序将不能正常运行;
     而如果将数组定义在全局变量里，则占用的是全局区，可以容纳较大的数组。
   - 注意：如果应用 `<bits/stdc++.h>` 头文件，数组在全局区不能命名为 `data` ，会产生冲突。

---

6. 正则表达式 `%[^\n]` 表示不是回车，可以用于连续的输入

```
比如 ：  scanf("%[^\n]",s);
```

---

7. 读取一整行字符串:

```
string s;
getline(cin,s);
```

---

8. 题目有时会有连续输入的问题
   比如：下面有一些数据要进行处理（并不告诉有多少组数据）

```
处理方法：
        int n;
        while(cin >> n){ ... }
        即将输入放在 while 循环中
```

---

9. 小数格式化输出
   `cout<<fixed<<setprecision(2)`
   - 要是浮点型数据才有效果，整形数据输出还会是整形
   - 会四舍五入,并不是直接舍去末尾的数
   - 要包含 `<iomanip>` 头文件

---

10. 常量字符串也可以用 []

```
比如 ：  cout << "Hello"[2];
        输出的是 e
```

- 在输出一段需要换行的数据时可以使用这个技巧

```
比如:    for(int i = 1;i<=n;++i)
        {
            for(int j = 1;j<=n;++j)
            {
                cout << data[i][j] << " \n"[j == n];
            }
        }
        在 j!=n 时 j==n 结果为0 ，则 " \n"[j == n] 实际为 ' ';
        在 j==n 时 j==n 结果为1 ，则 " \n"[j == n] 实际为 '\n';
        从而实现了换行效果
```

---

11. 重命名
    `using ll = long long;`

---

12. 定义常量

- 定义常量使用 `const int N = 1e5 + 9;`
  而不要用 `#define N 1e5 + 9` ，后者可能导致错误
- 根据实际需求确定取值，一般要比需求取大一点，避免溢出

---

13. 数组的赋值
    赋值不要从 `data[0]` 开始 ,从 `data[1]` 开始，一方面符合实际信息，另一方面便于数据的处理。

```
int a[N];a[0] = 0;
for(int i = 1;i<=n;++i)
{
    cin >> a[i];
}
```