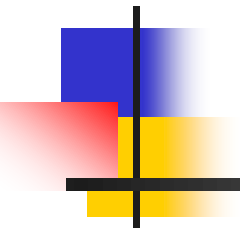


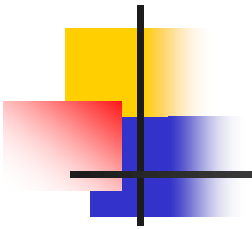
常用技巧

主要内容

- **C++ 中的 C**
  - **C++ 的头文件**
  - 常量： **const vs define**
  - 代码模板
- 常见输入输出模式
  - 文件输入
  - **EOF 是什么**
  - 多 **Case** 输入
- 数组大小的问题
- 数字的大小问题

C++ 中的 C

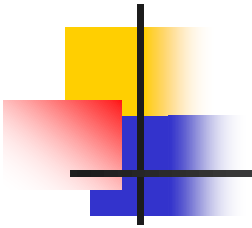
- C 是面向过程的
  是把整个大程序分为一个个的子函数；
- C++ 是面向对象的
  是把整个程序划分为一个个的类。
- C++ 是完全兼容 C 的， C 是 C++ 的子集，C++ 是 C 的超集。C++ 又对 C 做了很多补充和提升，因此使用 C++ 会比使用纯 C 更方便。
- 混用C和C++：用 C++ 写函数（可能会使用几个现成类），但并没有自己定义类，因此还是在面向过程的范畴内编程。



- **C++ 的头文件**
  - **C 的头文件都是以.h结尾的，如 stdio.h、 string.h。**
  - **C++ 的头文件分为两类（都没有后缀.h）**
    - **继承自 C 的头文件：去掉.h在前面加c，**

      如 cstdio、cstring。
    - **C++ 特有的头文件：如 iostream、 string。**
  - **使用 C++ 特有的头文件需在代码开头声明名称空间**

    using namespace std;



- 常量： **const vs. define**
  - 严格意义上讲 C 中是没有常量的概念的，所谓的`#define N 100`只是一种不加任何检查的文本替换。
  - C++ 中用关键字`const`定义常量，如`const int N = 1010;`。
  - 用`const`的定义的常量是有明确的数据类型的，在程序中可以进行类型检查和转换，更严谨，更规范。
  - 最常用的两个常量：
    - 定义最大数据范围：`const int N = 1e3 + 10;`
    - 定义无穷大：`const int INF = 0x3f3f3f3f;`

代码模板

Code

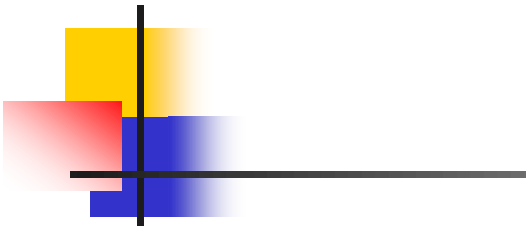
```
#include <cstdio>
#include <cstring>
#include <algorithm>
using namespace std;
typedef long long ll;
const int INF = 0x3f3f3f3f;
const int N = 1e3 + 10;

/*定义全局变量和函数*/
int  main()
{
    /*主函数内容*/
    return 0;
}
```

常见输入输出模式

- 文件输入
  - 当数据量大且需要多次调试时，每次手工输入数据或者复制粘贴效率太低。
  - 可以把输入数据放在同目录下一个文本文件**in.txt**中，每次程序执行时自动从文件中读取数据。

Code

```
int main()
{
    freopen(''in.txt'', ''r'', stdin);
    /*Your Code...*/
    return 0;
}
```



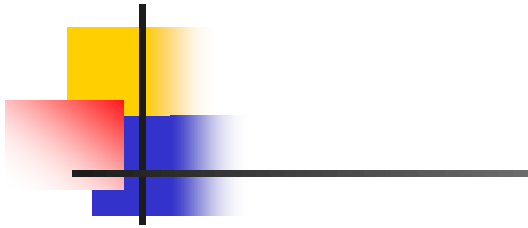
- **EOF是什么（End of File）**
  - EOF 就是-1：`#define EOF -1`
  - 在输入文件最后的一个不可见字符，占一个字节，ASCII 码为-1，二进制为1111 1111
    原码1000 0001 -> 反码11111 1110 -> 补码1111 1111

Code

```
while (scanf(''%d'', &n) != EOF) {...}
```

Code

```
while (~scanf(''%d'', &n)) {...}
```



- **多case输入（多个测试用例一起输入）**
  - 指定用例组数，需要输出用例序号
  - 指定用例组数，不需要输出用例序号
  - 不指定用例组数，有结束标记
  - 不指定用例组数，无结束标记

指定用例组数，需要输出用例序号

Code

```
int main()
{
    int T;
    scanf(''%d'', &T);
    for (int cas = 1; cas <= T;
                                cas++)
    {
        /*Your Code...*/
        printf(''Case #%d'', cas);
    }
    return 0;
}
```

Input

```
3
1 1
2 2
3 3
```

指定用例组数，不需要输出用例序号

Code

```
int main()
{
    int T;
    scanf(''%d'', &T);
    while (T--)
    {
        /*Your Code...*/
    }
    return 0;
}
```

Input

```
3
1 1
2 2
3 3
```

不指定用例组数，有结束标记

Code

```
int main()
{
    int n, m;
    while (scanf(''%d%d'',&n,&m),n)
    {
        /*Your Code...*/
    }
    return 0;
}
```

Input

```
1 1
2 2
3 3
0 0
```

```
31  int main()
32  {
33      int m, n;
34
35      scanf("%d%d", &m, &n);
36      while (~(m==0 && n==0))
37      {
38          printf("%d,%d\n", m, n);
39          scanf("%d%d", &m, &n);
40      }
41
42      return 0;
43  }
```

不指定用例组数，无结束标记

Code

```
int main()
{
    int n, m;
    while (~scanf(''%d%d'', &n,&m))
    {
        /*Your Code...*/
    }
    return 0;
}
```

Input

```
1 1
2 2
3 3
```

数组大小的问题

- “全局变量在静态存储区分配内存，局部变量在栈上分配内存。程序运行时会动态创建一个堆栈段，里面存放着调用栈，保存着函数的调用关系和局部变量。如果函数内的数组太大，可能会造成栈溢出。”
- **大数组最好开成全局变量。**
- 或者动态分配内容

```
 5
 6   int a[N+2];
 7
 8   int main()
 9   {
10       int b[N];
11       printf("Hello");
12   }
```

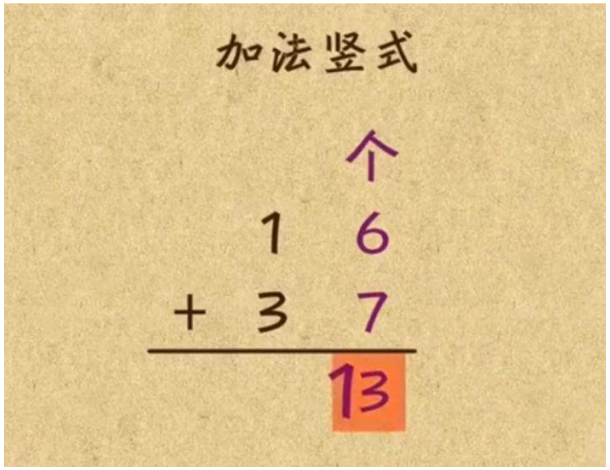
数字的大小问题（32bits）

- **int**
  2147483647 ≈ 2e9
- **long long**
  9223372036854775807 ≈ 1e19
- **unsigned long long**
  2^{64}-1 ≈ 2e19
- **20! ≈ 2e18**

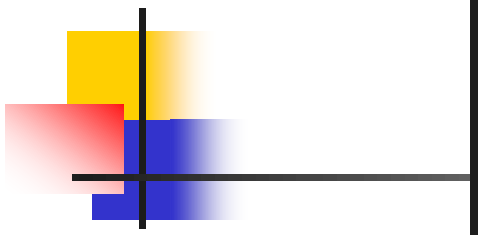
- **高精度加法/减法：**

加法核心代码：

```
c[i] += a[i] + b[i];
if (c[i] >= 10) {
    c[i + 1]++;
    c[i] -= 10;
}
```



减法同理，注意a和b之间的大小关系



```c
#include<stdio.h>
#include<string.h>
int a[100010], b[100010], c[100010];
char s[100010];
void string_to_num(int a[]) {
    scanf("%s", s);
    a[0] = strlen(s);
    for (int i = 1; i <= a[0]; i++)
        a[i] = s[a[0] - i] - '0';
}
int main() {
    memset(a, 0, sizeof(a));
    memset(b, 0, sizeof(b));
    memset(c, 0, sizeof(c));
    string_to_num(a); string_to_num(b);
    int len = a[0] > b[0] ? a[0] : b[0];
    for (int i = 1; i <= len; i++) {
        c[i] += a[i] + b[i];
        if (c[i] >= 10) {
            c[i + 1]++;
            c[i] -= 10;
        }
    }
    if (c[len + 1] != 0)    len++;
    for (int i = len; i >= 1; i--)
        printf("%d", c[i]);
    return 0;
}
```

友情提醒

从提交代码发现的问题

代码风格

```
#include <stdio.h>

int main()
{
    int i, n = 0;

    for (i = 0; i < 10; ++i) {
        if (n % 2 == 0) {
            n = n + i;
        } else {
            n = n - i;
        }
    }

    printf("%d\n", n);

    return 0;
}
```

- 注意花括号的位置
- 缩进应该是四个空格
- for、while等语句和括号间要有空格
- ,;等后要有空格或换行
- 运算符的前后应有空格
- 函数名后的括号无空格
- if、for、while后必须使用花括号
- 要在适当的位置添加空行
- 函数要加 return

输入的有效性检查

```
5  void string_to_num(int a[]) {
6      scanf("%s", s);
7      a[0] = strlen(s);
8      for (int i = 1; i <= a[0]; i++)
9          a[i] = s[a[0] - i] - '0';
10 }
```