# 程序开头结尾定式

C程序开头会有这些东西：

#include <isotream>

using namespace std;

作用：std 是一个命名空间。不同的命名空间可以有相同的类名被定义，using namespace std;就是指明下面的程序使用std,如果不用这句指明的话就要用std::string(string是std空间中定义的也可以在全局空间中定义,只要名字空间不一样即可)。否则可以默认名字空间中有std.便不用std::来修饰。这是C++新标准中有的，解决多人合作编程时名字冲突问题。如果用（旧标准）#include<iostream.h>就不需写这句话，但是如果用（新标准）#include<iostream>就必须要写。

开头结束之后，程序一般都包含这些东西：

int main() {

...//中间代码内容略

return 0;

}

这是因为定义了一个有整型返回值的主函数int main，因此要返回一个整数型。这个值一般是0，表示没有出现异常情况正常返回。比如在bat或shell脚本中，调用一个命令如何知道它执行有没有成功呢？就看它的退出码。当然也可以 return -1 或者其他值，返回非零值表示主函数执行过程中的异常情况。具体什么数字代表什么异常，可以由编程者自己定义。C中也有不返回值得主函数void main：

#include <iostream> //头文件

using namespace std; //名字空间

void main() //无返回值主函数

{

cout<<"hello world!\n"; //输出语句

}

# VS 2015操作

## 快捷键整理

批量自动对齐：（选中需要批量对其的行）先按Ctrl + K，再按Ctrl + F。

## debug模式

进入debug模式是F5，或者点击窗口上的本地Windows调试器。debug如果要逐行运行，首先要在暂停的位置加上断点；F10是每次前进一整步（比如某个步骤调用了其它函数，按F10是直接输出这一步的结果）；F11是每次前进一行（如果某个步骤调用了其它函数，按F11是跳转到调用的那个函数，继续逐行执行）。执行过程中，局部变量，全局变量的值都会清晰的在窗口中显示出来。

CTRL+F5是编译运行，F5运行结束后窗口不会一闪而过。

# I/O

程序输入输出是解题的基本。输入输出有两种主要的形式，即输入流、输出流形式：cin, cout；以及格式化输入、输出：scanf, printf。

## 输入、输出流

输入输出流需要用到头文件#include <iostream>。

以下命令用于设定输出格式，需要用到头文件#include <iomanip>：

1. 使输出宽度固定（如果输出值宽度小于这个宽度，会在左边自动填充空格；如果输出值宽度大于这个宽度，按照输出值宽度输出）：

cout << setw(3) << no; //将no按照宽度为3输出

1. 使输出宽度固定，并且左边填充自己想要的字符，而不是空格，可以用setfill命令配合setw使用：

cout << setw(3) << setfill ('0') << no; //将no按照宽度为3输出，左边空白填充0

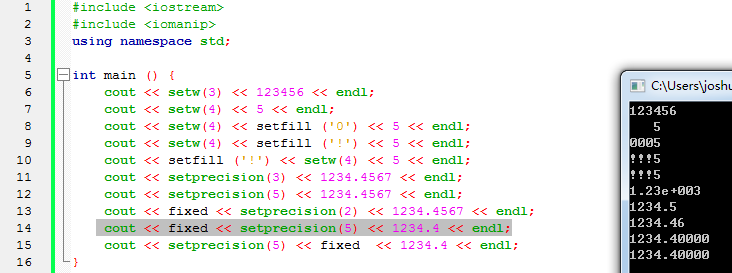
3、固定输出的有效数字数量，根据数字的实际情况，有可能有小数，也有可能编程科学计数法：

cout << setprecision(5) << no; //将no保留5位有效数字输出

1. 固定小数位数输出，在setprecision前面或者后面加上fixed：

cout << fixed << setprecision(5) << no; //将no保留五位小数（整数位数不受影响）

下面是这些命令的效果展示：



## 格式化输入、输出

格式化输入、输出需要用到头文件#include <stdio.h>。

下面是scanf和printf的例子：

scanf("%d,%d:%d",&a,&b,&c); //格式符号和变量地址一一对应

printf("a=%d,b=%d,c=%d",a,b,c); //格式符号和变量名一一对应

scanf需要注意：

1. 后面跟的必须是已经定义好的变量的地址（指针）；

2. %后面跟的是数据类型，具体见后表；

3. 每个格式类型之间的分隔符号就是输入时候用来分隔的符号，比如上面前两个%d之间用逗号，后两个之间用冒号，则从exe界面输入数据是需要这样输入:100,200:500（假设abc分别为100，200，500），如果%d之间无符号，则输入数据按照空格分隔。

printf需要注意：

1. 可以通过%5.3f这样的形式设定最小输出宽度和保留小数位数。其中前面的5是最小输出宽度，比如%5d，如果输出数字小于5位，则前面补空格；如果大于五位，按照实际位数输出。后面的.3是输出小数位数，比如%.2f就是这个浮点数输出时保留两位小数。%5.3f就是输出最小宽度为5，保留三位小数。浮点数保留整数输出则是%.0f。

2. 后面输出的形式可以自己定，只要数据类型符号和变量一一对应就行，比如printf("a的值是%d,b的值是%d,c的值是%d",a,b,c);是可以的，假设abc分别为100，200，500，则最终输出的效果是：a的值是100,b的值是200,c的值是500。

|  |  |
| --- | --- |
| d | 十进制整数 |
| o | 八进制整数 |
| x | 十六进制整数 |
| u | 无符号十进制整数 |
| f | 浮点数 |
| c | 单个字符 |
| s | 字符串 |

## 数字后面的空格

很多题目要求输出一系列数，用空格隔开，但是最后一个数字后面不加空格。通常有两种做法：第一种，如果容易判断一个数字是不是最后一个数字（比如循环数量确定，且每次循环输出一个值，那么循环的最后一次就是最后一个数字），那么就设置一个逻辑语句判断最后一个数字，将最后一个数字只输出数字，不输出空格。第二种，如果一个数字是否是最后一个数字很难判断，比如浙大“程序设计入门——C语言”里面的“完数”这个题目，很难判断一个完数是不是这个范围里的最后一个完数，此时可以先输出第一个数（一般第一个数较最后一个数更容易判断），然后在以后每一个输出数字的前面加空格。

# 函数

C语言中可以传递数组的地址给一个函数，call函数时将数组名直接写上而不加角标就是将数组地址传递给函数。但是需要注意的是，要接受数组的函数在定义时必须写清楚接受函数的类型、维度，比如下面的定义就包括了接受的数组mat的类型（char）和维度（2维），并且要注意，传递多维数组时必须指明除了第一维外的所有维度长度。

int update(char mat[][100], int mat\_len){ //向公式传递多维数组，必须指明所有维度的长度，只有第一维可以省略。

而call这个函数时就只需要将数组名写上即可（如果将角标也写上，是将数组的值复制给函数，而不是将地址传给函数）：

update(mat, n);