oiClass

信息学讲义

（入门篇）

陈旭龙　梁　健　卢剑锋　著

吉林教育出版社

oiClass

信息学讲义

（入门篇）

陈旭龙　梁　健　卢剑锋　著

吉林教育出版社

图书在版编目 (CIP) 数据

oiClass信息学讲义. 入门篇 / 陈旭龙, 梁健, 卢剑锋著. — 长春 : 吉林教育出版社, 2019.3
ISBN 978-7-5553-6962-2

Ⅰ. ①o… Ⅱ. ①陈… ②梁… ③卢… Ⅲ. ①C++语言—程序设计—高等职业教育—教学参考资料 Ⅳ. ①TP312.8

中国版本图书馆CIP数据核字(2019)第051916号

oiClass 信息学讲义　入门篇　　陈旭龙　梁　健　卢剑锋　著

责任编辑　钱丽丽　　**封面设计**　优盛文化

出　　版　吉林教育出版社（长春市同志街 1991 号　邮编　130021）
发　　行　吉林教育出版社
印　　刷　定州启航印刷有限公司

开　　本　787 毫米 ×1092 毫米　1/16
印　　张　18.25
字　　数　400 千字
版　　次　2019 年 3 月第 1 版
印　　次　2019 年 3 月第 1 次印刷
书　　号　ISBN 978-7-5553-6962-2
定　　价　55.00 元

前　言

我们想编写一本关于 OI（信息学奥林匹克）的教材很久了。虽然我们三人还算是“青年教师”，但是踏入 OI 教学都有一段不短的时间，看着信息学联赛（NOIP）和信息学竞赛（NOI）由冷到热，近年来逐渐受各级各类学校重视，越来越多的青少年学生踏入编程的行列，相关的教材也随即涌现，但是在教学中能让学生看懂的教材并不多，有些教材还是停留在语法为主的阶段，有些教材对算法的含义轻轻带过，让人看得云里雾里。其实，编程说到底还是为人类服务的，算法也是来自我们的生活，只是隐匿在事物的背后，不容易被发觉，我们可以通过多种方法，启发学生揭开这层神秘的面纱，通过学习编程和算法的规律，解决一个又一个实际的问题，这才是信奥的目标，也是信息学课程存在的价值。

本书结合 oiClass 题库网站，以“C++ 语言基础—算法—数据结构”为主线，循序渐进，轻语法，重实践；以培养通过编程解决问题的能力为目标，通过通俗易懂的语言，激发学生的学习兴趣；以“计算思维”为核心，巧妙地通过例子，让学生尝试以计算机科学家的角度思考问题；精挑细选例题及习题，避免题海战术，让学生花最少的时间训练，培养学生扎实的代码能力；oiClass 题库网站中还设有“比赛”项目，便于教师展开阶段性测试，评估教学的效果。此外，书本和网站还配有完备的习题解答、参考代码和测试数据等学习资源。全书分为 20 个单元，由浅入深，基本覆盖了信息学竞赛的所有入门知识。

与大部分信奥教练不同，我们都是在成为信息技术教师后才开始的“OI”之路，都是先获得了校内教学的成果再涉足信奥培训，其中陈旭龙老师、梁健老师分别获得过广州市信息技术优质课评比初中、高中组一等奖，卢剑锋老师长期担任学校的学科带头人，因此我们编写的角度都是从学生出发，想尽一切办法让学生理解语言的基础和算法的含义，非常适合作为 NOIP 青少年培训入门教程。

感谢广州科技贸易职业学院信息工程学院邬厚民院长为我们提供的帮助，没有他的帮助和鼓励，本书无法面世。

因水平和时间问题，难免出错，欢迎广大读者不吝指正。

编者

2018 年 12 月于广州

目 录

C++ 程序结构

C++ 是众多编程语言中的一种，也是最受欢迎的语言之一，使用极其广泛，关于其语言背景，在此先略过不谈，有兴趣者请自行搜索吧。以下直接就 C++ 程序结构开始讲起。

对于入门者而言，以下是 C++ 最基本的程序结构。

```
1.#include<iostream>    //引用头文件
2.
3.using namespace std;    //声明命名空间
4.
5.int main(){    //主函数
6.    int a,b,c;
7.    cin>>a>>b;
8.    c=a+b;
9.    cout<<c;
10.}
```

接下来，我们简要介绍这三部分。

`#include<iostream>` *//引用头文件*

`#include<>` 是 C++ 中引用头文件的格式，方括号中加入头文件的名字。为什么要引用头文件呢？头文件就像一个抽屉，里面放着各种工具，这些工具在程序中叫作函数（如果你不理解什么是函数，没关系，反正我们现在也用不到），如果我们要使用这些工具，就得先打开抽屉，而引用头文件就是告诉程序打开这个抽屉，我要从这个抽屉中拿工具。在以上代码的示例，#include<iostream> 是告诉程序打开一个叫 iostream 的抽屉，然后我们从这个抽屉中拿出了 cin 和 cout 这两个工具，这两个工具在主函数中使用。如果不先引用 iostream 头文件，主函数中 cin 和 cout 工具将不能使用。当然，C++ 中有很多抽屉（头文件），每个抽屉中都放着各种工具，这在我们以后的学习中会慢慢接触到的。

`using namespace std;` *//声明命名空间*

声明命名空间，其实这句话不是必需的，但是使用它，可以使我们的程序变得简洁一点。为什么要声明命名空间呢？打个比方，假设在学校中，三年级（1）班有一个同学叫小

明，四年级（2）班也有一个同学叫小明，如果我们在广播中叫小明，大家就会不知道我们具体叫的是哪一班的小明，除非我们加上班级，如三年级（1）班的小明。但是，我们可以先做个声明：以下我所读的名字都是三年（1）班的同学，接着我读到小明，尽管这里没有加限定，但是大家都知道我所指的是三年（1）班的小明。同理，C++ 中有很多工具（函数），工具名难免相同，所以在使用工具的时候就得声明是哪个抽屉中的工具，比如用 cin，就得写成 std::cin，用 cout 就得写成 std::cout，这就是给 cin 和 cout 加限定条件，而使用命名空间，告诉程序，我使用的工具都是 std 这个命名空间内的，例如：我说的 cin，就是 std::cin；我说的 cout，就是 std::cout，这样，我们就可以把工具前的限定条件 std:: 省略掉了。当然，如果这个比喻不能让你理解，也没关系，毕竟这个语句也不影响我们的编程学习。

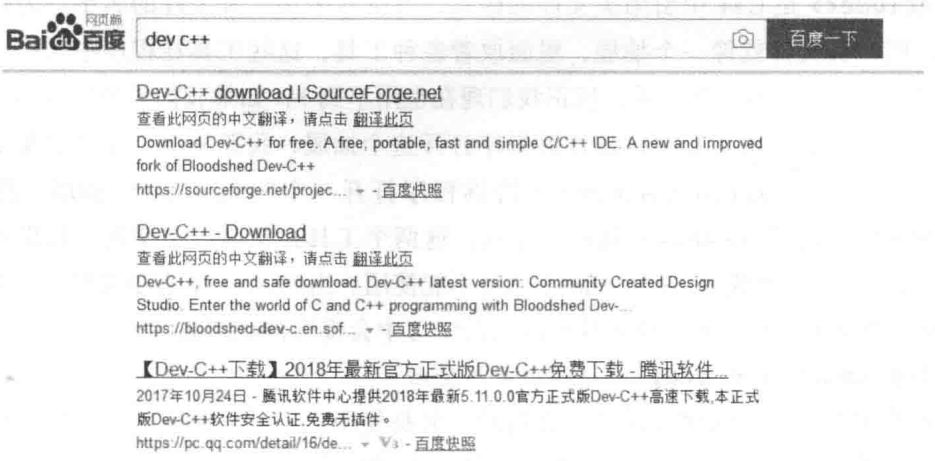
接下来是 main 函数：

```
1.int main(){
2.    int a,b,c;
3.    cin>>a>>b;
4.    c=a+b;
5.    cout<<c;
6.}
```

以上是主函数的格式，每个 C++ 程序必须有个 main() 函数才能运行。在详细解释 main() 函数这段代码之前，我们先运行下这段代码。

输入代码

理论上你可以用任何文本工具编写代码，在这里推荐大家使用基于 Windows 平台的 Dev-C++。Dev-C++ 是免费软件，下载安装都很方便，随便百度搜索下都下载得到。如下图。

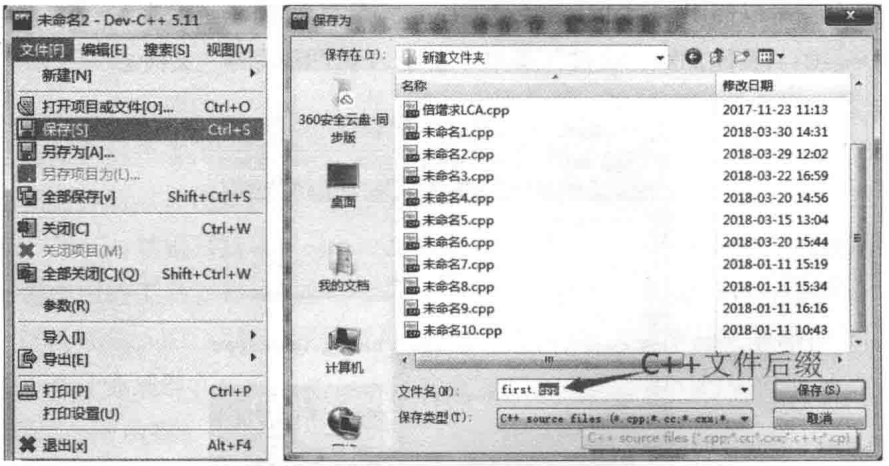


MAC 平台也可以使用类似 CLion 或者 Xcode 等软件编写 C++ 程序。

打开 Dev-C++ 软件，从“文件”菜单中新建一个“源代码”文件。

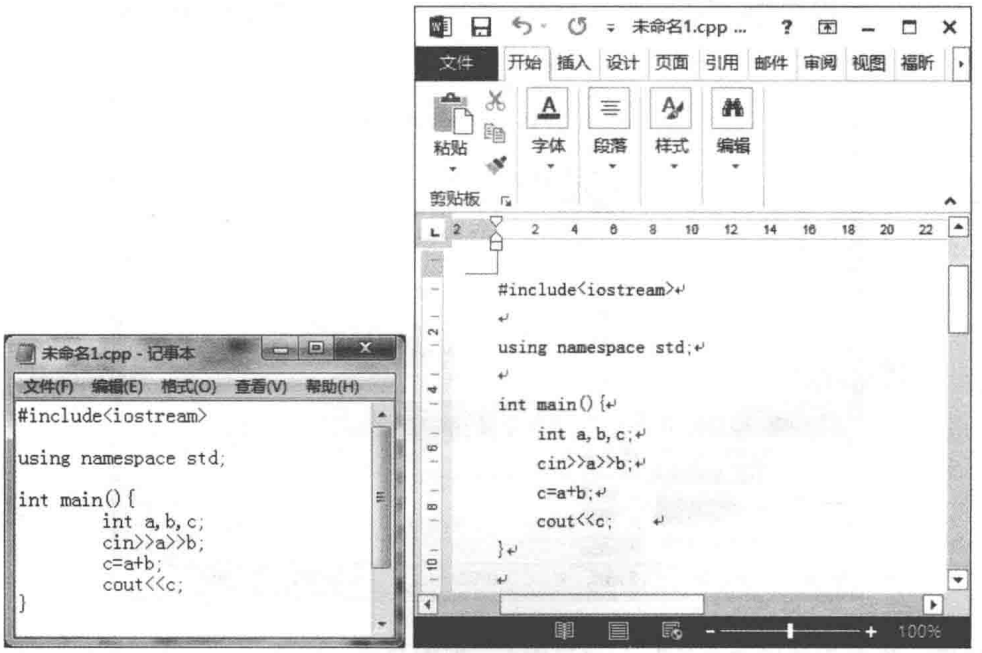
在新建的源代码文件中输入以下代码，特别要注意的是标点符号不能错。

然后，单击菜单栏“文件”→“保存”，保存代码。注意：C++ 程序文件的默认后缀为 .cpp。



编译程序

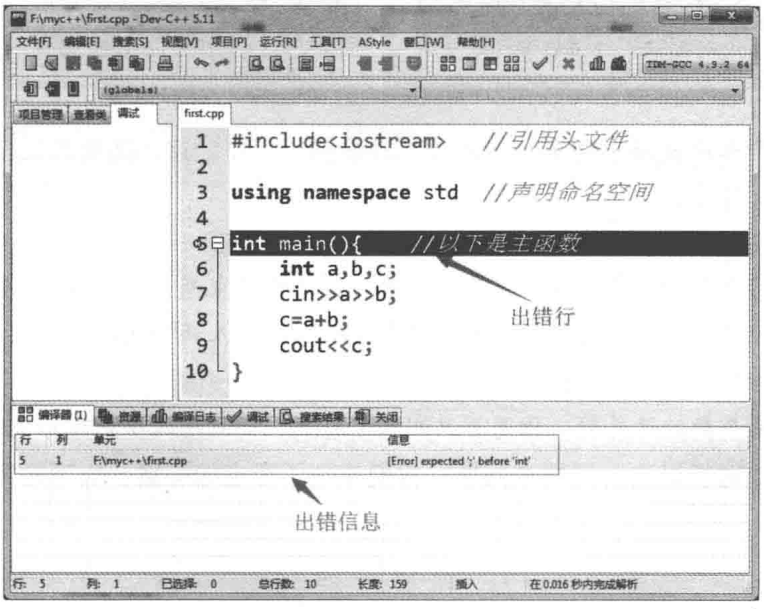
如果你写好了上述代码，恭喜你完成首次 C++ 编程，但是，这仅仅是代码，它还不是程序。程序是指能够被电脑直接执行的文件，比如后缀为 .exe 的文件。保存代码后你再瞧瞧，这个程序的后缀是 .cpp，这个文件可以被任何文本处理软件打开，可以用记事本打开，也可以用 Word 打开（如下图），打开之后，也就只有上述代码。



那么，如何将代码变成程序呢？打个比方，要将一堆面粉变成香喷喷热乎乎的面包，其中还需要一个烘焙的过程，在这里，代码就好比面粉，程序就好比面包，要将代码变成程序，还需要一个烘焙的过程，这个过程，我们叫作编译。编译就是将代码转换成程序的过程。

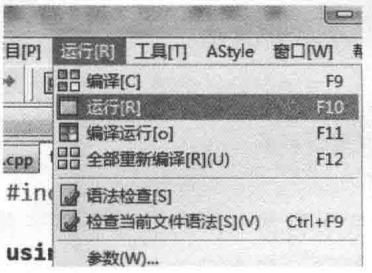
在 Dev-C++ 中编译程序，在菜单栏中选择“运行”→“编译”（如下图），进行编译程序。如果代码没有错误，将能正确生成一个 .exe 文件，可以在 .cpp 文件所在同级目录中查找。

如果编译出错，则可以结合出错信息在出错行附近查找错漏，留心每行语句是否用了分号（;）结尾，头文件名字是否打错等。

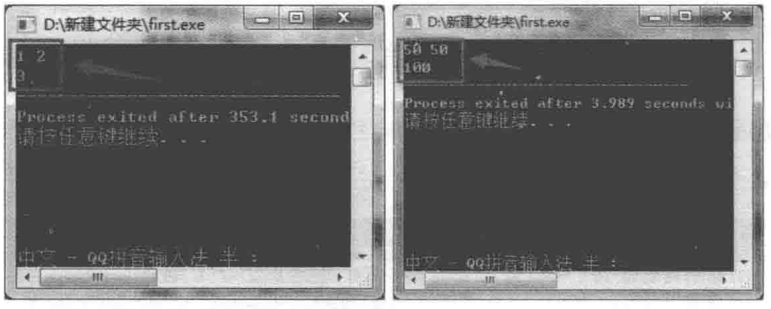


运行程序

代码编译成功后，我们就可以运行程序了，这就像面包烘烤完成，我们当然是开吃啦。单击菜单栏“运行”→“运行”（如下左图），运行程序，此时你会看到如下右图的一个黑乎乎的窗口，里面的光标在闪动，似乎在等待着什么。是的，它在等待你输入数据。



这时，你可以尝试输入两个数字，数字与数字之间用空格隔开，然后按下回车键，看看发生什么。多次运行程序，输入不同的两个数字，看看程序输出的数据有什么变化。



没错，这个程序实现了两个数的加法。看看你的程序，这么简短的几行代码就实现了两个数的加法，兴奋吧！

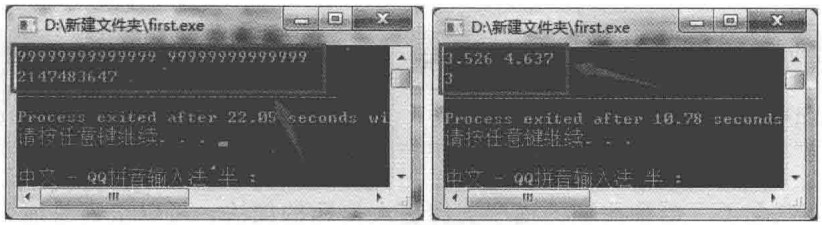
先别急着高兴，这里有几个问题需要你做以下实验验证一下：

1. 你成功写出了两个数的加法，你能否改造你的程序求两个数的减法、乘法、除法？

2. 在运行程序时，你输入的最大的数是多大？如果输入两个比较大的数，比如 10 的 18 次方级别的两个数，程序还能返回正常结果吗？

3. 在进行两个数的运算时，你是否只测试了整数，如果输入小数程序能正常工作吗？

仔细观察，对于以上的后两个问题，我们找到了否定的答案。以下是实验截图，有图有真相。



太诡异了，为什么会出现这种现象呢？

数据类型

上述问题，跟数据类型有关系。数据类型是程序的基础，它告诉我们数据的意义以及我们能在数据上执行的操作。C++ 语言支持广泛的数据类型。它定义了几种基本内置类型（如字符、整型、浮点数等），同时也为程序员提供了自定义数据类型的机制。在这里，我们先认识两种基本的类型：整型和浮点型。

整型

整型，其实也是整数，C++ 中用 int 来标识。与数学上整数的概念不同，计算机中的整型是一个有限的概念，数学上的整数是一个无穷的概念，不存在一个最大的整数，但是在 C++ 中，int 所能表示的整数是有一个范围的，即在 -2147483648 至 2147483647 之间，超出这个范围，就不是 int 所能表示的了。当然，在 C++ 中，关于整型的标识符可不止 int 一个，还有 short、long long 等。如果我们把整型比喻成一个物品类，比如衬衣，那么 int 就相当于衬衣中的中码，short 就是小码，long long 就是大码，当然还有加大码，不过现在这三个整型标识符就足够我们用了。记住，每个整型标识符（short、int、long long）的表示范围都是有限的。以下是关于整型更详细的范围介绍。

数据类型	定义标识符	占字节数	数值范围
短整型	short [int]	2(16 位)	-32768 ~ 32767
整型	[long] int	4(32 位)	-2147483648 ~ 2147483647
长整型	long [int]	4(32 位)	-2147483648 ~ 2147483647
超长整型	long long [int]	8(64 位)	-9223372036854775808 ~ 9223372036854775807
无符号整型	unsigned [int]	2(16 位)	0 ~ 65535
无符号短整型	unsigned short [int]	2(16 位)	0 ~ 65535
无符号长整型	unsigned long [int]	4(32 位)	0 ~ 4294967295

（续 表）

数据类型	定义标识符	占字节数	数值范围
无符号超长整型	unsigned long long	8(64 位)	0 ~ 18446744073709551615

认识整型概念了，要让我们的程序支持更大整数的加法，可以很容易地实现，如下图所示。但是，尽管改成了大码的整型 long long，它也是有一个范围限制的。

```
1.#include<iostream>
2.using namespace std;
3.int main(){
4.    long long a,b,c;    //这里 int 改成了 long long
5.    cin>>a>>b;
6.    c=a+b;
7.    cout<<c;
8.}
```

浮点型

浮点型是用来表示小数的数据类型，在 C++ 中浮点型的标识符有 float、double 和 long double，这三个类型也好比衣服的小码、中码、大码，表示的数据范围不同。具体范围如下：

数据类型	定义标识符	数值范围	占字节数	有效位数
单精度实型	float	-3.4E-38 ~ 3.4E+38	4(32 位)	6 ~ 7 位
双精度实型	double	-1.7E+308 ~ 1.7E+308	8(64 位)	15 ~ 16 位
长双精度实型	long double	-3.4E+4932 ~ 1.1E+4932	16(128 位)	18 ~ 19 位

表中的数值范围用的是科学记数法，如 3.4E+38 是指 3.4*1038

同理，认识浮点型后，我们也很容易将我们的程序改造成支持小数运算，方法如下：

```
1.#include<iostream>
2.using namespace std;
3.int main(){
4.    double a,b,c;    //这里将 int 类型改成了 double 类型
5.    cin>>a>>b;
6.    c=a+b;
7.    cout<<c;
8.}
```

变量

其实数据类型，指的是一种规范、一种标准，好比我们说大码的衣服，指的是这件衣服领口的标准、袖长的标准，并没有具体指哪一件衣服，我们说整型 int，指的是一个能够表示整数，且数据范围在 -2147483648 至 2147483647 之间的一种规范，并没有具体告诉程序怎样来存储这样的数据。

正如我们可以按照大码的尺寸来生产衣服一样，我们也可以按照 int 的标准来制定存储空间，这样所制定的存储空间，我们称为变量。变量提供一个具名的、可供程序操作的存储空间，C++ 中的每个变量都有其数据类型，数据类型决定着变量所占的内存空间大小、存储值的范围以及变量能参与的运算。声明变量的格式如下：

类型说明符 变量名列表;

这里，我们举例来说明：

```
int a;
```

这句话的意思是告诉程序，我需要分配一个能装 int 类型数据的内存空间，并且给这个内存空间取名为 a。这里的内存空间就像一个容器一样，它能装 int 类型的数据，就意味着它只能装整数，而且整数的范围在 -2147483648 至 2147483647 之间。简单的理解就是，变量是一种能装特定数据类型的容器，具体能装什么数据类型，在声明的时候就确定了。以下是更丰富的变量声明方式：

```
1.int sum;    //分配一个能装 int 数据类型的内存空间，给它取名为 sum
2.float price,cost,pay;  //声明三个 float 类型的变量，分别取名为 price,cost,pay
3.int a=5,b=10,c=1;  //声明三个 int 类型的变量 a、b、c,并且给它们赋初值
```

这里还需要特别强调的是，声明变量时，变量名是由我们自己取名的，取名的时候需要遵守以下规范：

● C++ 的标识符（identifier）由字母、数字和下划线组成，其中必须以字母或下划线开头。

●标识符的长度没有限制，但是对大小写字母敏感。

● C++ 语言保留了一些名字供语言本身使用，这些名字不能被用作标识符。

练习：请指出下面的名字中哪些是非法的？

```
1.int double=3.14;
2.int _;
3.int catch-22;
4.int 1_or_2=1;
5.double Double=3.14;
```

答案：

```
1.int double=3.14;  //非法，double 是数据类型关键字
2.int _;  //合法，下划线开头，但不建议这样取变量名
3.int catch-22;  //非法，包含非法字符“-”，变量名只能是数字、字母和下划线
4.int 1_or_2=1;  //非法，必须以字母和下划线开头
5.double Double=3.14;  //合法，Double 和 double 是不一样的，区分大小写
```

赋值语句

刚刚，我们说了变量是一个容器，可以往里面装数据，但是怎么装呢？这里就需要用到我们的赋值语句。赋值语句的作用就是将一个数据存储到变量中，赋值语句的一般形式：

```
变量 = 表达式；  //不要漏了末尾的分号哦
```

在 C++ 语言中，“=”作为赋值运算符，而不表示“等于”判断。赋值语句是由赋值表达式再加上分号构成的表达式语句。样例分析：

```
1.a=3;    //将数值 3 存储在左边的变量 a 中
2.b=4;    //将数值 4 存储在左边的变量 b 中
3.c=a+b; //计算 a+b 的结果，将结果存储在左边的变量 c 中
```

注意：赋值运算符的左侧运算对象必须是一个变量。以下赋值语句是非法的。

```
1.1024=k;  //非法赋值，1024 不能是变量名
2.i+j=k;  //非法赋值,i+j 不能是变量名
```