JISUANJI CHENGXU SHEJI JICHU II C/C++

计算机程序设计基础 II

C/C++

景 红 编著

西南交通大学出版社

四川省“十二五”普通高等教育本科规划教材

计算机程序设计基础 II
(C/C++)

景 红 编著

本书数字资源汇总

西南交通大学出版社
·成 都·

内容提要

本规划教材以实际应用为主线，由浅入深地介绍了计算机程序设计中的基本概念、基础知识和基本技能，可帮助学生（读者）掌握编程解决问题的一般思路与基本方法。其内容主要包括：软件开发、算法描述和C/C++语言基本语法等基础知识，常用数据类型和经典问题通用算法，编程基本方法和基本调试技能。

本教材由《计算机程序设计基础I (C/C++)》和《计算机程序设计基础II (C/C++)》两分册构成。分册I为基础篇，主要内容包括结构化程序设计，可用于3学分课程教学。分册II为提高篇，主要内容包括面向对象程序设计和STL（标准模板库），可与分册I一起用于4~5学分课程教学。

本教材的内容全面系统，叙述简明易懂，案例丰富实用，适合作为高等院校学生学习计算机程序设计的教材，同时也可作为自学C/C++语言的指导书和参考书。

图书在版编目（CIP）数据

计算机程序设计基础. Ⅱ，C/C++ / 景红编著. —
成都：西南交通大学出版社，2018.4
ISBN 978-7-5643-6158-7

Ⅰ. ①计… Ⅱ. ①景… Ⅲ. ①C语言－程序设计－高等学校－教材 Ⅳ. ①TP311.1②TP312.8

中国版本图书馆CIP数据核字（2018）第075119号

计算机程序设计基础Ⅱ（C/C++）

景 红 编著

责任编辑	姜锡伟
封面设计	何东琳设计工作室
出版发行	西南交通大学出版社 （四川省成都市二环路北一段111号 西南交通大学创新大厦21楼）
发行部电话	028-87600564　028-87600533
邮政编码	610031
网址	http://www.xnjdcbs.com
印刷	四川煤田地质制图印刷厂
成品尺寸	185 mm × 260 mm
印张	14.75
字数	387千
版次	2018年4月第1版
印次	2018年4月第1次
书号	ISBN 978-7-5643-6158-7
定价	38.00元

课件咨询电话：028-87600533
图书如有印装质量问题　本社负责退换

前　言

在高校本科教育中，“计算机程序设计”是理工科类专业的一门重要公共基础课程。其教学目标是，在课程结束时学生能够：养成利用计算机解决问题的思维方式，具有计算思维素养、创新意识和团结合作的工程职业素质；掌握一门高级程序设计语言的基础知识，具有使用计算机编程解决实际问题的基本能力；为未来在本学科领域使用计算机进行应用研究、技术开发等工作奠定基础。本规划教材是针对高校计算机程序设计课程编写的，并根据使用了多年的教材《计算机程序设计基础（C++）》（景红主编，西南交通大学出版社 2009 年版）的教学效果以及课程教改成果，构建起基于丰富实用的案例和以问题为主线的知识体系。全书针对编程求解过程和学生学习特点，从思路分析、数据结构规划、算法设计、程序实现等方面做了比较细致和全面的讨论，并力求语言简明、概念准确、目的明晰。其目标是使学习者通过学习达到：熟悉 Visual C++程序的开发和调试环境；掌握 C/C++语言的基础知识；掌握面向过程、面向对象和 STL 程序设计的基本方法和基本调试技能；掌握常用数据类型（包括基本数据类型、自定义数据类型）的用法，以及一些经典问题的通用算法；能够使用 C++语言编程解决一般性问题。

本规划教材的内容组织很好地利用了西南交通大学出版社的数字化教学平台，为学习者提供了多维度的学习支持，包括微视频讲解、编程训练、知识点测试及结果分析等。

参加本规划教材撰写工作的人员都是在高校长期从事计算机教学与科研工作的一线教师，并有着丰富的教学经验。其中，参加《计算机程序设计基础 I (C/C++)》分册撰写工作的教师是：3.4 节　吴燕、景红，4.4.1～4.4.3 节　张旭丽、景红，4.4.4～4.4.8 节　冯晓红、景红，5.5 节　刘霓、景红，6.4 节　戴克俭、景红，其余各章节和数字化资源　景红；参加《计算机程序设计基础Ⅱ（C/C++）》分册中撰写工作的教师是：8.4.3 节　李茜，其余各章节和数字化资源　景红。在本教材完稿之际，刘金艳、陈小平、刘倩、钟灿、胡桂珍、王绍清和唐加胜等老师对该书内容提出了非常宝贵的修改意见，在此我们要表示深深的感谢。同时，我们要感谢西南交通大学从事计算机基础教学工作的全体教师。

身处教学一线的我们，力图撰写一本更适合培养学生计算机程序设计能力的好教材。但我们也深知，不同的学习者有着不同的个性特征、认知结构、学习动机和学习风格等，因此本教材不一定能满足所有学习者的需要。同时，在撰写过程中失误在所难免。所以，我们衷心希望广大读者能够不吝赐教。

编　者

2018 年于西南交通大学

多媒体资源目录

序号	章	节	资源名称	资源类型	页码
1	扉页		本书数字资源汇总	专题	扉页
2	目录		《计算机程序设计基础Ⅰ》目录清单	图文	目录-2
3	第6章（续）函数的使用	6.5	6.15-1 填空练习	图文	3
4		6.5	6.15-2 填空练习	图文	4
5		6.6	6.16 填空练习	图文	6
6		6.7	6.17 填空练习	图文	7
7		6.8	6.18 填空练习	图文	8
8		6.8	程序欣赏 1	可执行文件	9
9		本章练习		图文	12
10	第8章 自定义数据类型与链表	8.1.1	8.1 填空练习	图文	15
11		8.1.1	8.1 问题拓展（2）	图文	16
12		8.1.2	8.2 填空练习	图文	17
13		8.2.1	8.3 填空练习	图文	20
14		8.2.1	8.3 问题拓展（1）	图文	20
15		8.2.2	8.4 填空练习	图文	25
16		8.3.1	8.5 填空练习	图文	30
17		8.3.1	8.5 问题拓展（1）	图文	34
18		8.4.1	8.6 源程序	图文	37
19		8.4.1	8.6 头文件	图文	37
20		8.4.1	8.6 问题拓展	图文	39
21		8.4.1	8.7 参考程序	图文	39
22		8.4.1	8.7 问题拓展	图文	41
23		8.4.2	8.8 参考程序	图文	41
24		8.4.3	8.9 参考程序	图文	46
25		本章练习		题库	56
26	第9章 面向程序设计	9.1.1	9.1-1 填空练习	图文	62
27		9.2.2	9.1-2 填空练习	图文	76
28		9.2.2	9.1 问题拓展（1）	图文	79
29		9.2.2	9.1 问题拓展（2）	图文	79
30		9.3.1	9.2 填空练习	图文	87
31		9.3.1	9.2 思考与示例	图文	91

续表

序号	章	节	资源名称	资源类型	页码
32	第 9 章　面向程序设计	9.3.1	9.2 问题拓展（2）	图文	93
33		9.3.2	9.3-1 填空练习	图文	99
34		9.3.2	9.3-1 程序欣赏	可执行文件	101
35		9.3.3	9.3-2 填空练习	图文	106
36		9.3.3	9.3-3 填空练习	图文	109
37		9.3.3	9.3-2 问题拓展（1）	图文	111
38		9.3.3	9.3-2 问题拓展（2）	图文	111
39		9.3.4	9.4 填空练习	图文	114
40		9.3.5	9.5 填空练习	图文	117
41		9.3.6	9.6 填空练习	图文	123
42		9.3.7	9.7 填空练习	图文	126
43		9.4	9.8 填空练习	图文	130
44		9.4	9.8 问题拓展	图文	132
45		9.5.1	9.9 参考程序	图文	135
46		9.5.2	9.10 参考程序	图文	138
47		9.5.3	9.11 参考程序	图文	141
48		本章练习		题库	144
49	第 10 章　STL 程序设计	10.2.1	10.1-1 填空练习	图文	153
50		10.2.1	10.1-2 填空练习	图文	153
51		10.2.1	附录 7	图文	156
52		10.2.2	10.2 填空练习	图文	157
53		10.2.3	10.3-1 填空练习	图文	161
54		10.2.3	10.3-2 填空练习	图文	162
55		10.2.3	10.3-2 问题拓展	图文	163
56		10.2.3	10.3-3 填空练习	图文	169
57		10.2.3	10.3-4 填空练习	图文	171
58		10.2.4	10.4 填空练习	图文	174
59		10.2.5	10.5-1 填空练习	图文	178
60		10.2.5	10.5-2 填空练习	图文	181
61		10.2.6	10.6 填空练习	图文	187
62		10.2.6	10.7 填空练习	图文	189
63		10.2.7	10.8 填空练习	图文	197
64		10.2.8	10.9 填空练习	图文	199

续表

序号	章	节	资源名称	资源类型	页码
65	第 10 章 STL 程序设计	10.2.9	10.10 填空练习	图文	201
66		10.3.1	10.11 填空练习	图文	203
67		10.3.2	10.12 填空练习	图文	207
68		10.3.3	10.13 填空练习	图文	212
69		10.4.1	10.14 参考程序	图文	216
70		10.4.2	10.15 参考程序	图文	217
71		10.4.3	10.16 参考程序	图文	219
72		10.4.3	10.17 参考程序	图文	220
73		10.4.3	10.18 参考程序	图文	221
74		本章练习		题库	222
75	附录	附录 1 ASCII 代码表		图文	224
76		附录 2 C++的词法单位		图文	
77		附录 3 C++的基本数据类型与转义字符		图文	
78		附录 4 C++运算符的优先级和结合性		图文	
79		附录 5 几类常用系统函数简介		图文	
80		附录 6 常见出错信息含义表		图文	
81		附录 7 STL 的常用运算符和成员函数		图文	
82		附录 8 STL 相关泛型算法		图文	
83	微课程	微课程 1　第 1 章引论		视频	225
84		微课程 2　软件开发和程序编制		视频	
85		微课程 3　计算机算法		视频	
86		微课程 4　编制一个简单的程序		视频	
87		微课程 5　调试程序的基本方法		视频	
88		微课程 6　基本数据类型		视频	
89		微课程 7　基本预算		视频	
90		微课程 8　顺序与选择结构程序设计		视频	
91		微课程 9　嵌套与多路分支选择结构		视频	
92		微课程 10 循环结构的实现		视频	
93		微课程 11 嵌套循环结构的实现		视频	
94		微课程 12 系统函数的使用		视频	
95		微课程 13 用户自定义函数的使用		视频	

多媒体资源使用帮助：

1. 请按照本书封底的操作提示，使用微信扫描封底二维码，关注“交大 e 出版”微信公众号并成为本书数字会员。

2. 多媒体资源目录中的所有资源在书中相应位置都设有二维码，请使用手机微信扫描该二维码，直接点击即可免费阅读/获取相应资源。

目　录

[基础篇——SP]

[提高篇——OOP]

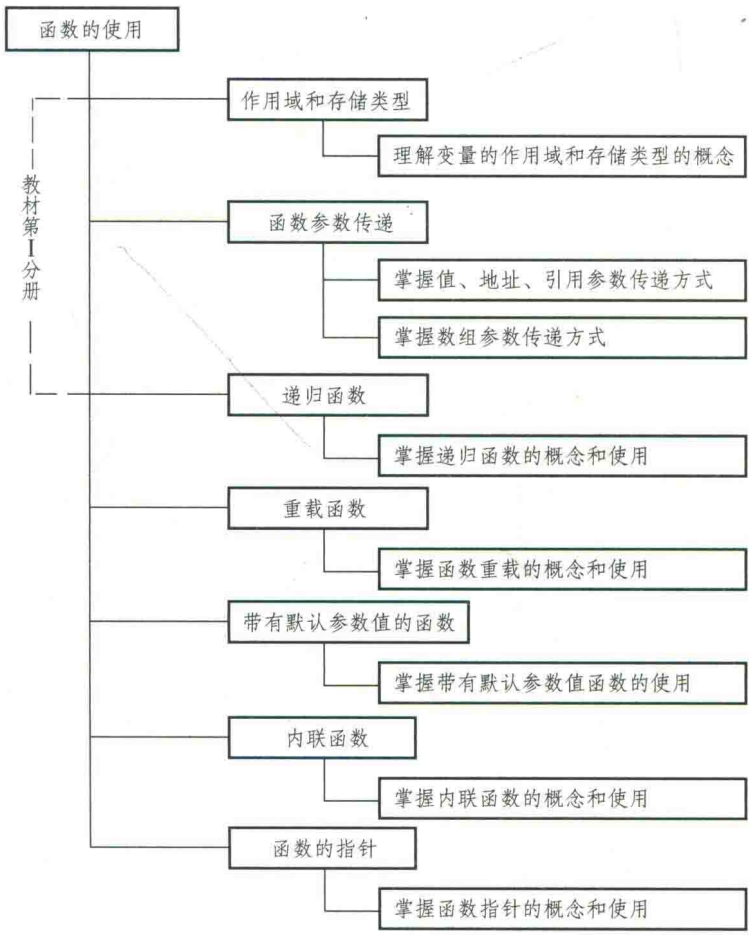
[提高篇——GP]

《计算机程序设计基础Ⅰ》目录清单

第 6 章　函数的使用（续）

通过前面的学习可知，C++系统提供了丰富的标准函数可以助力编程解决很多问题。但是在实际应用中，总会存在一些问题的解决没有适宜的标准函数可用的情况，这时就不可避免地需要根据问题特性来自定义用户函数。为此，这里学习 C++语言中函数的深入使用。

本章预期学习成果：



本章案例索引表：

语法知识点	应用案例	页码
函数重载	6.5 节案例 6.15：两数比较	3
带有默认参数值的函数	6.6 节案例 6.16：求和运算（1）	5
内联函数	6.7 节案例 6.17：求和运算（2）	7
函数指针	6.8 节案例 6.18：两数比较	8
	6.8 节程序欣赏 1：绘制彩色的小狮子	9
	6.8 节“语法知识与编程技巧”例 1	11

6.5 函数的重载

在同一个作用域内，一个函数名可以对应多个函数实现并通过不同的参数类型或参数个数加以区分。也就是说，对于同名函数，编译系统能够通过识别实参的个数和类型来确定具体调用其中的哪个函数。这种方法称为函数的重载。

【案例 6.15】 两数比较。

◇ 问题背景

从键盘上输入两个整数和两个实数，分别输出它们的较大者。

◇ 编程实现 1

```
//6.15-1 两数比较
#include<iostream>
using namespace std;
//============================
int maxInt(int xInt, int yInt)
{    int zInt;
     zInt=(xInt>yInt?xInt:yInt);
     return zInt;
}
//============================
double maxDouble(double xDouble, double yDouble)
{    double zDouble;
     zDouble=(xDouble>yDouble?xDouble:yDouble);
     return zDouble;
}

void main(void)
{    int aInt,bInt;
     double cDouble,dDouble;
     cout<<"请输入两个整型数："<<endl;
     cin>>aInt>>bInt;
     cout<<"请输入两个实型数："<<endl;
     cin>>cDouble>>dDouble;
     //============================
     cout <<"两个整数中较大者为："<<maxInt (aInt,bInt)<<endl;
     //============================
     cout<<"两个浮点数中较大者为："<<maxDouble (cDouble,dDouble)<<endl;
}
```

填空练习

◇ 运行结果

```
D:\test\123\Debug\123.exe
请输入两个整型数：
123 234
请输入两个实型数：
13.045 13.046
两个整数中较大者为：234
两个浮点数中较大者为：13.046
请按任意键继续. . .
```

◇ 程序分析

分析程序 1，不难发现：maxInt 函数和 maxDouble 函数只是参数类型有所不同，而且在编制程序时需要特别注意数据与函数名的对应，既不方便也容易出错。

一个好的解决方法是：函数重载。

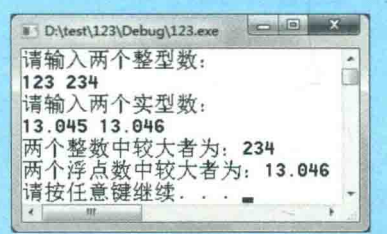
◇ 编程实现 2

```
//6.15.2 两数比较——函数重载
#include<iostream>
using namespace std;
//==========================
int maxUser(int xInt, int yInt)
{    int zInt;
     zInt=(xInt>yInt?xInt:yInt);
     return zInt;
}
//==========================
double maxUser(double xDouble, double yDouble)
{    double zDouble;
     zDouble=(xDouble>yDouble?xDouble:yDouble);
     return zDouble;
}

void main(void)
{    int aInt,bInt;
     double cDouble,dDouble;
     cout<<"请输入两个整型数："<<endl;
     cin>>aInt>>bInt;
     cout<<"请输入两个实型数："<<endl;
     cin>>cDouble>>dDouble;
     //============================
     cout <<"两个整数中较大者为："<<maxUser(aInt,bInt)<<endl;
     //============================
     cout<<"两个浮点数中较大者为："<<maxUser(cDouble,dDouble)<<endl;
}
```

填空练习

◇ 运行结果



★语法知识与编程技巧★

函数的重载

一般来说，在同一个作用域中定义的函数名不能一样。但是有一种情况例外，那就是函数的

重载。函数的重载是指允许有两个及以上的函数使用同一个函数名，但是形参的个数或者形参的类型必须有所不同。而重载函数的返回值类型可以相同也可以不同。

当发生对重载函数的调用时，匹配按以下顺序进行：

✓ 参数类型是否严格匹配，如果找到了，就用那个函数。

✓ 经过内部类型转换寻求一个匹配，只要找到了，就用那个函数。

✓ 经过强制类型转换寻求一个匹配，只要找到了，就用那个函数。

说明：

◆ 判断同名函数是否为重载函数，是以函数参数而不是函数的返回值为依据的。所以，在C++中，如果在同一个作用域中有两个同名函数的参数表完全相同而返回值不同，并不会被判定为重载，而会被判定为不合法。

例 1：以下两个函数是重载函数：

```
int maxUser(int,int);
double maxUser(double,double);
```

当调用名为 maxUser 的函数时，编译系统将根据实参和形参的类型及个数的最佳匹配，自动确定调用哪一个函数。

例 2：以下两个函数不是重载函数：

```
int maxUser(int,int);
float maxUser(float,float);  //编译系统将给出错误提示。
```

错误原因：编译系统区分函数参数类型时，实际上是以参数占据的存储空间来区分的。由于 int 型和 float 型均为 4 个字节，所以编译系统无法区别 int 型和 float 型参数。

◆ 一般不应将不同功能的函数定义为重载函数。对函数进行重载设计是为了让具有相似功能的操作具有相同的名字，从而提高程序的可读性和适应性。而如果对完全没有相似功能的函数进行重载，既失去了本来的目的，又容易造成调用结果的误解和混淆。

6.6 带有默认形参值的函数

在函数的应用中，常常也会遇到这种情况，即在多次调用某个子函数时，传递的实参值相同。对此，可以将该值指定为该函数对应形参的默认值，使调用更加灵活。也就是说，C++允许在函数原型声明或函数定义中，为部分或全部形参指定默认值，并且，通常将这样的函数称为带有缺省参数值的函数。它为函数调用带来**方便性**和**灵活性**。

【案例 6.16】 求和运算。

◇ 问题背景

通过阅读和上机调试该程序，学习带有缺省参数值的函数的使用方法。

◇ 编程实现

```
//6.16 求和运算(1)
#include <iostream>
using namespace std;
int add(int x=10,int y=20)  //给出了全部默认值
{    int z;
     z=x+y;
     return(z);
}
double add(double x,double y=5.0)  // 任何一个带默认值的参数的右边不得有不带默认值的参数
{    double z;
     z=x+y;
     return(z);
}

void main(void)
{    //根据 2 个实参都是整数，调用第 1 个函数，且实参值起作用
     cout<<"add(3,5)="<<add(3,5)<<endl;
     //根据 2 个实参都是 double 型数据，调用第 2 个函数，且实参值起作用
     cout<<"add(1.2,4.6)="<<add(1.2,4.6)<<endl;
     //根据实参是整型数据，调用第 1 个函数，且第 2 个默认的形参值会起作用
     cout<<"add(3)="<<add(3)<<endl;
     //根据实参是 double 型，调用第 2 个函数，且第 2 个默认的形参值会起作用
     cout<<"add(1.2)="<<add(1.2)<<endl;
     //调用第 1 个函数（因为它给出了所有默认形参值），且所有默认形参值起作用
     cout<<"add()="<<add()<<endl;
}
```

填空练习

◇ 运行结果

```
C:\Users\jd\documen...
add(3,5)=8
add(1.2,4.6)=5.8
add(3)=23
add(1.2)=6.2
add()=30
请按任意键继续. . .
```

★语法知识与编程技巧★

函数形参默认值的使用方法

在调用带有默认值参数（也称为缺省参数）的函数时，其参数传递规则是：从左至右，如果给出了实参，则以实参传递；如果没有给出实参，则使用形参默认值。

在使用带有默认值参数的函数时，应注意以下几点：

① 函数的形参可以全部带有默认值，也可以只有部分带有默认值。而在后一种情况下，带有默认值的参数必须位于形参列表中的最右边，即任何一个带默认值的参数的右边不能出现不带默认值的参数。也就是说，必须按形参列表从右向左的顺序来指定缺省形参值。

② 如果在同一个源文件中既有该函数原型，又有该函数定义，则只能在该函数原型中指定参数的默认值。

③ 调用带有默认值参数的函数时，编译系统将会根据实际给出的实参进行调用和计算。简单地讲就是用实参替代原来的默认参数（这种替代是按照从左到右的顺序进行的），若没有实参则取默认参数值。

6.7 内联函数

在发生函数调用时，编译系统要做许多工作，主要包括断点现场保护、数据进栈、执行函数体、数据出栈、恢复现场和断点等，资源开销很大。而当函数体很小而又需要反复调用时，由于函数体的运行时间相对较少，而函数调用所需的栈操作等却要花费比较多的时间，运行效率与代码重用的矛盾就变得很突出。

为了解决上述矛盾，C++提供了一种被称作**内联函数**的机制。该机制通过将函数体的代码直接插入函数调用处（不像函数调用那样需要现场保护等）来节省函数的时间开销，这一过程叫作**内联函数的扩展**。由于在扩展时对函数的每次调用均要进行扩展，所以，内联函数实际上是一种以空间换时间的方案。

【案例 6.17】 求和运算。

◇ 问题背景

通过阅读和上机调试该程序，学习内联函数的使用方法。

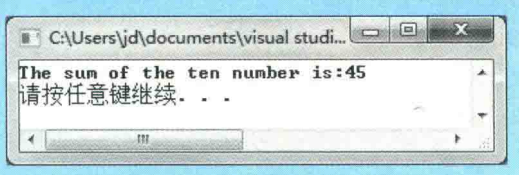
◇ 编程实现

```
//6.17 求和运算(2)
#include <iostream>
using namespace std;
//内联函数原型声明
inline int add(int,int);

int main(void)
{    int i,sum=0;
     for(i=0;i<10;i++)
          sum=add(sum,i);
     cout<<"The sum of the ten number is:"<<sum<<endl;
     return 0;
}
//内联函数的定义
inline int add(int a,int b)
{
     return a+b;
}
```

填空练习

◇ 运行结果



内联函数的使用方法

内联函数也被称为内置函数或在线函数。当定义一个函数时，在函数名前加上关键字 inline，该函数即被定义为内联函数。

内联函数既具有函数的优点，又可以提高程序运行效率。但是，由于重复编码会产生较长的代码，所以内联函数通常都非常小。而且，并不是所有的函数都能设计为内联函数。

C++对内联函数有很多使用限制，具体内容如下：

① 在内联函数中不能定义任何静态变量；

② 内联函数中不能有复杂的流程控制语句，如循环语句、switch 和 goto 语句等；

③ 内联函数不能是递归的；

④ 内联函数中不能使用数组；

⑤ 加有关键字 inline 的函数定义必须出现在该函数的调用之前，或者在函数的原型声明语句中必须加有关键字 inline。

说明： 如果不满足上述限制条件，程序可以通过编译，但该函数没有实现“内联”。也就是说，编译系统将把它当作普通函数来处理。

使用内联函数虽然节省了程序运行的时间开销，但增大了代码占用内存的空间开销。因此，在具体编程时应仔细权衡时间开销和空间开销之间的矛盾，以确定是否采用内联函数。

6.8 函数的指针

函数在编译时会被分配一个入口地址，即函数在内存中的首地址，称为函数的指针。因此，可以使用一个指针变量获取函数的指针，然后调用该函数。

【案例 6.18】 两数比较。

◇ **问题背景**

从键盘上输入两个整数，输出它们的较大者。

◇ **编程实现**

```
//6.18 两数比较——函数的指针
#include <iostream>
using namespace std;
//=======================================================
int maxUser(int x,int y)
{
      return x>y ? x : y;
}
```

填空练习