

# C++高级语言程序设计

徐洁

北京邮电大学网络空间安全学院

## 课程介绍:

**参考教材:** [1] 《C++ Primer Plus 第6版 中文版》, [美] 史蒂芬·普拉达 (Stephen Prata) 著, 张海龙, 袁国忠 译, 人民邮电出版社, 2020-07-01。

**成绩:** 课堂出勤5%、平时作业15%、兴趣小组30%、期末50%

**代码要求:** 写程序要符合google代码规范

**课程作业要求:** 作业两次课收一次

# 课程介绍：教学内容详述



## 基础+核心

### ➤ 1 C++语言概述（3课时）

- ❑ 内容：计算机编程语言的发展历史，C++语言的特点，面向过程与面向对象程序设计及示例
- ❑ 重点：C++面向对象的思维方式
- ❑ 难点：面向过程与面向对象编程的设计异同

### ➤ 2 C++语言概述实验（3课时）

- ❑ 内容：介绍C++代码编写的规范，编写面向过程和面向对象的程序
- ❑ 重点：VS面向对象集成开发环境的使用，简单的C++程序的编写
- ❑ 难点：C++语言和C语言的编写差异

### ➤ 3 函数（3课时）

- ❑ 内容：函数的功能和意义，函数的参数传递方式，函数的调用过程，函数的递归调用
- ❑ 重点：函数的递归调用，变量的作用域，变量的存储类型，值传递
- ❑ 难点：函数的调用过程和递归调用，变量的作用域和存储类型

### ➤ 4 编译预处理（3课时）

- ❑ 内容：有参数宏，无参宏，条件编译
- ❑ 重点：无参宏、有参宏的定义和使用
- ❑ 难点：有参宏与函数的异同，多文件程序的组织

# 课程介绍：教学内容详述

## ➤ 5 C++基础实验（3课时）

- ❑ 内容：指导学生完成递归函数、数组、结构体、共用体和枚举类型相关编程
- ❑ 重点：使用vs进行程序调试，有参宏的定义和使用
- ❑ 难点：函数的递归调用

## ➤ 6 指针和引用（3课时）

- ❑ 内容：指针的定义和计算，指针与数组的关系，数组指针，指向指针的指针
- ❑ 重点：指针的定义和运算，指针与二维数组，指向指针的指针，new和delete的使用
- ❑ 难点：指针与二维数组，数组指针

## ➤ 7 指针和引用实验（3课时）

- ❑ 内容：指导学生完成指针、指针数组、数组指针、指向指针的指针相关编程
- ❑ 重点：指针与二维数组的关系，new和delete的实现
- ❑ 难点：带参程序的调试

## ➤ 8 类和对象（6课时）

- ❑ 内容：类和对象的关系，访问控制权限，构造函数，析构函数
- ❑ 重点：访问控制属性，构造函数与析构函数及其特殊性，友元函数与友元类
- ❑ 难点：This指针及其应用，静态成员



基础+核心

# 课程介绍：教学内容详述



## 基础+核心

### ➤ 9 类和对象实验（3课时）

- ❑ 内容：指导学生完成类和对象、构造函数和析构函数相关编程
- ❑ 重点：构造函数的定义与使用，析构函数的定义与使用
- ❑ 难点：友元函数和友元类的使用

### ➤ 10 运算符重载（3课时）

- ❑ 内容：友元函数重载，成员函数重载，几个特殊操作符的重载
- ❑ 重点：运算符重载为成员函数，运算符重载为友元函数
- ❑ 难点：赋值、++、--和下标运算符重载

### ➤ 11 运算符重载实验（3课时）

- ❑ 内容：指导学生完成成员函数重载、友元函数重载和特殊成员函数重载
- ❑ 重点：成员函数重载，友元函数重载
- ❑ 难点：特殊符合的重载

### ➤ 12 继承和派生（3课时）

- ❑ 内容：单一派生，多重派生，派生过程中的主要问题和支配关系，虚函数，纯虚函数
- ❑ 重点：单一派生，多重派生，基类成员vs对象成员，虚函数
- ❑ 难点：赋值兼容，虚基类，虚析构函数

# 课程介绍：教学内容详述

## ➤ 13 继承和派生实验（3课时）

- ❑ 内容：指导学生完成单一继承、多重继承和运行时多态相关编程
- ❑ 重点：继承与派生的实现方式，虚函数的实现方式
- ❑ 难点：纯虚函数的实现方式

## ➤ 14 输入/输出流类库（3课时）

- ❑ 内容：输入流，输出流，提取运算符与插入运算符的重载
- ❑ 重点：标准设备的输入/输出，文件的输入/输出
- ❑ 难点：重载提取和插入运算符



基础+核心

# 课程目标：形成规范的面向对象的编程习惯

基础+核心

Object Based  
(基于对象)

➤ 以良好的方式编写C++ 类

- ❑ 不带指针的类
  - Complex
- ❑ 带指针的类

Object Oriented  
(面向对象)

➤ 学习类之间的关系

- ❑ 继承
- ❑ 派生

◆ Google开源项目风格指南

- 英文原版: <https://github.com/google/styleguide>
- 中文版: <https://github.com/zh-google-styleguide/zh-google-styleguide>

# C++语言概述

- C/C++简介
- 简单的C++程序
- 标准输入输出
- C++程序的开发步骤和上机调试流程
- 调试C++程序



# C/C++ 简介

程序：

用于解决特定问题而为计算机编写的指令序列。

程序设计语言：

用于书写计算机程序的语言。

```
110010100100000011100
001110100011011110010
110011010110010000001
110010100100000011100
001110100011011110010
110011010110010000001
110010100100000011100
```

机器语言

可读性差

```
MOV AH, 9
INT 21H
MOV AH, 4CH
INT 21H
```

汇编语言

可移植性差

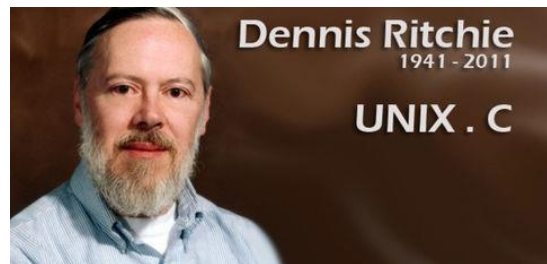
```
if (a>b)
    max=a;
else
    max=b;
```

高级语言

自然直观通用

## C语言：

- ✚ 二十世纪七十年代Bell实验室的Dennis ; Ritchie为开发UNIX系统而编写设计了C语言；
- ✚ 1978年与Brian W. Kernighan合著了《The C Programming Language》；
- ✚ 1983年ANSI推出第一个C语言标准ANSI C；
- ✚ 此后ISO又陆续采纳了C99、C11等标准



## C语言的主要特点

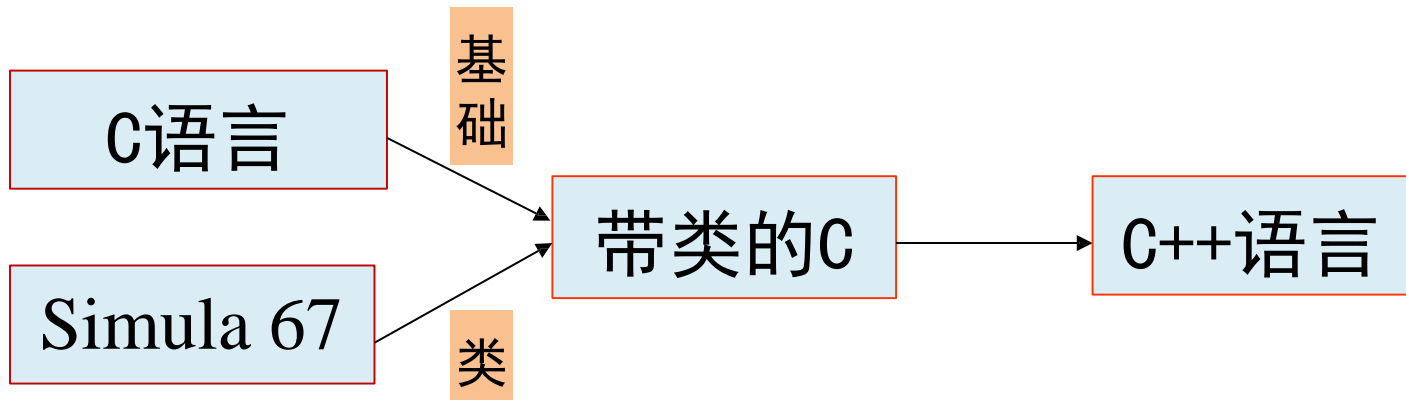
- 语言结构化，简洁，规模小，数据类型丰富，使用灵活方便。非常适用于设计和编写大型系统软件、大型应用软件，又适用于编写小程序。
- 兼有高级语言和汇编语言的特点。程序表述灵活方便，结构性好，目标程序质量高，程序执行效率高。
- 程序的可移植性好。源于①已将与硬件有关的语言成分尽可能剥离，由库函数实现；②有丰富的预编译命令支持；③标准化程度高，有ANSI/ISO国际标准。

## C语言的主要不足

- 随着C语言的广泛应用，它的一些不足受到人们的关注，如对数据类型检查较弱，没有对面向对象技术的支持，随着软件工程规模的扩大，难以适应开发特大型的程序等。

## C++语言：

- 1980年贝尔实验室的Bjarne Stroustrup博士及其同事对C语言进行了改进和扩充，并把Simula 67中类的概念引入到C中，1983年正式命名为C++。为了满足面向对象程序设计的要求。



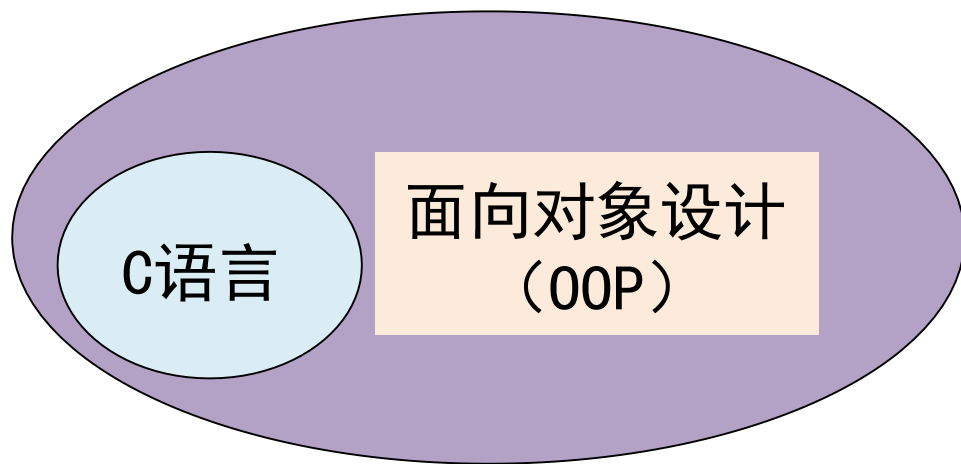
# C++语言:

年份	C++标准	名称
1998	ISO/IEC 14882:1998	C++98
2003	ISO/IEC 14882:2003	C++03
2011	ISO/IEC 14882:2011	C++11
2014	ISO/IEC 14882:2014	C++14
2017	ISO/IEC 14882:2017	C++17
2020	ISO/IEC 14882:2020	C++20
2023	Yet to be determined	C++23



## C++与C的联系：

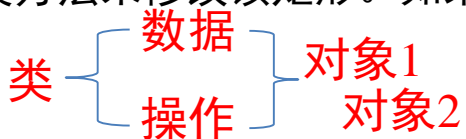
C++语言是C语言的超集，它完全涵盖了C语言的内容。在C语言基础上添加了类代表的面向对象语言、C++模板支持的泛型编程。



C++语言构成



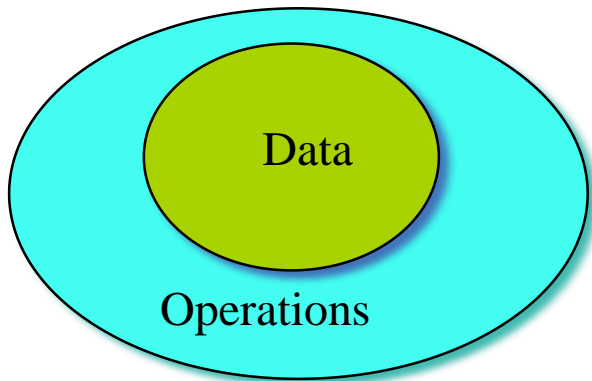
## C与C++的编程理念：

- 一般来说，计算机语言要处理两个概念-**数据**和**算法**（函数）。数据是程序使用和处理的对象，而算法是程序使用的方法。C语言侧重如何设计一个过程，它强调的是编程的算法方面。过程化编程首先要确定计算机应采取的操作，然后使用编程语言来实现这些操作。
- C++语言是一种面向对象的语言，它强调的是数据。在C++中，类是一种规范，它描述了这种新型数据结构，对象是根据这种规范构造的特定数据结构。例如，假设正在开发一个能够绘制矩形的计算机绘图程序，则可以定义一个描述矩形的类。定义的数据部分包括顶点的位置、长和宽、4条边的颜色和样式、矩形内部的填充颜色和图案等；定义的操作部分可以包括移动、改变大小、旋转、改变颜色和图案、将矩形复制到另一个位置上等操作。当使用该程序来绘制矩形时，它将根据类定义创建一个对象。该对象保存了描述矩形的所有数据值，因此可以使用类方法来修改该矩形。如果绘制两个矩形，程序将创建两个对象，每个矩形对应一个。

类 { 数据 } 对象1  
          { 操作 } 对象2

## C与C++的编程理念：

- 面向对象编程OOP不仅是将数据和方法合并为类。还可以**保护数据**，使其免遭不适当的访问。
- 编写代码时，设计有用、可靠的类是一项艰巨的任务。而在大的厂商支持下，在不同编译器下，有了大量有用的类库。可以方便程序员重用和修改现有的、经过仔细测试的代码。这正是C++的优点之一。

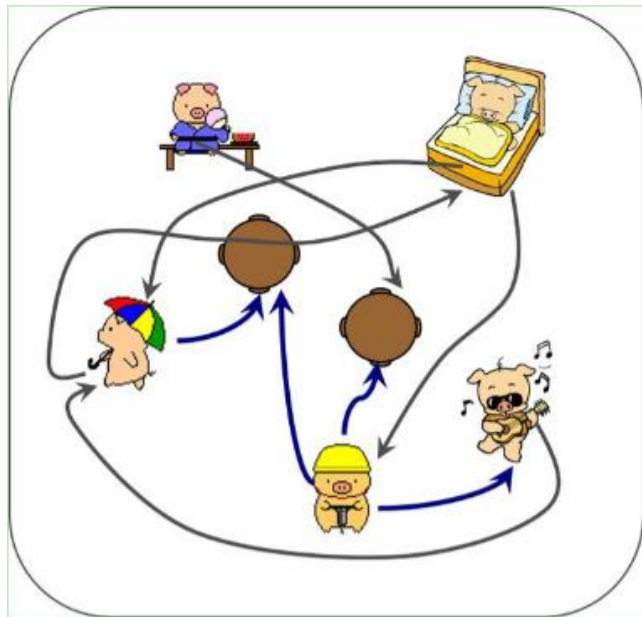


## 泛型编程理念：

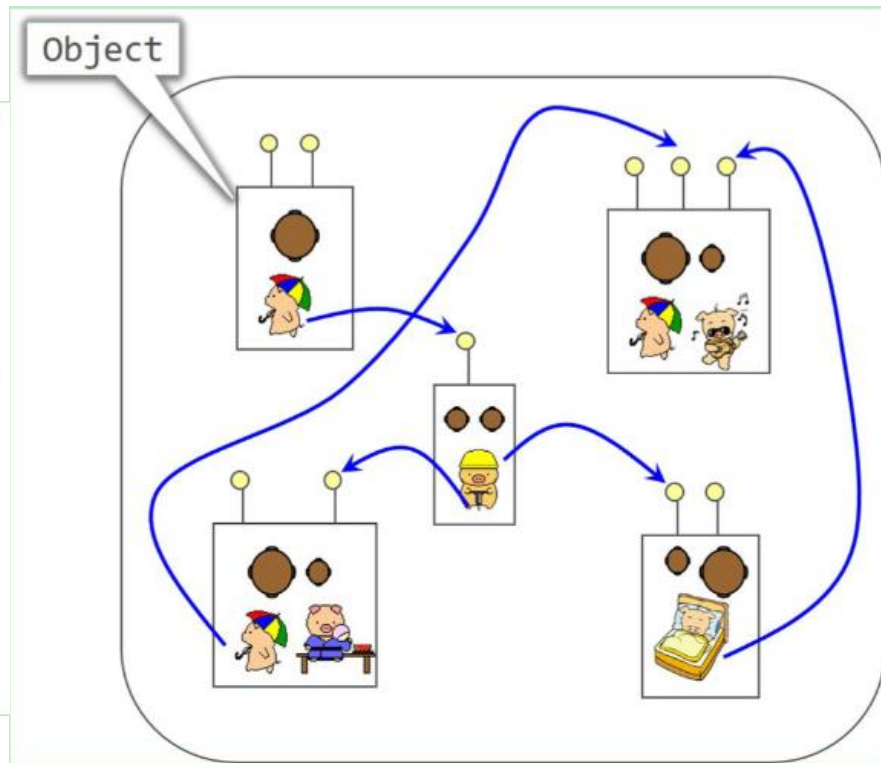
- 泛型编程与OOP的目标相同，即使重用代码和抽象通用概念的技术更简单。不过OOP强调的是编程的数据方面，而泛型编程强调的是**独立于特定数据类型**。它提供了执行常见任务（如对数据排序或合并链表）的工具。
- 术语泛型指的是创建独立于类型的代码。C++的数据表示有多种类型-整数、小数、字符、字符串、用户定义的、由多种类型组成的复合结构。例如，要对不同类型的数据进行排序，通常必须为每种类型创建一个排序函数。而泛型编程中，只需编写一个泛型（不是特定类型的）函数，并将其用于各种实际类型。

# 编程理念的图例：

- 结构化编程（SP）

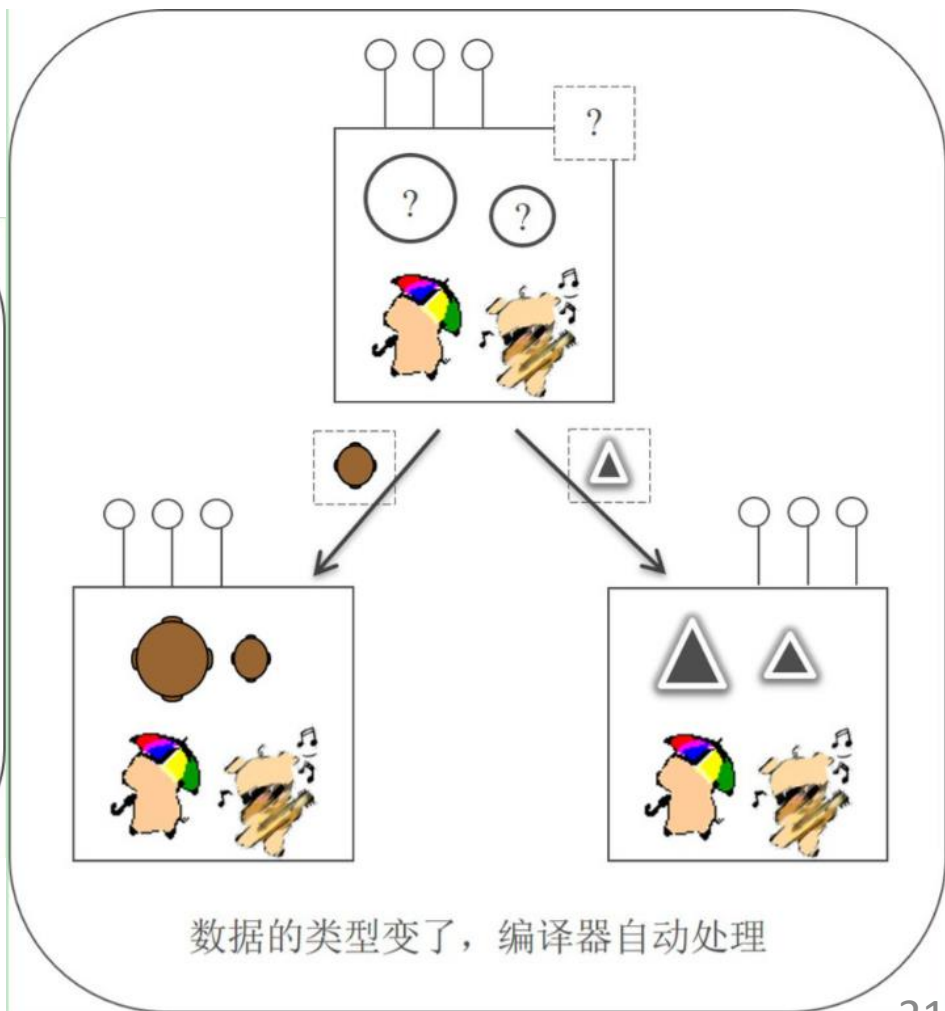
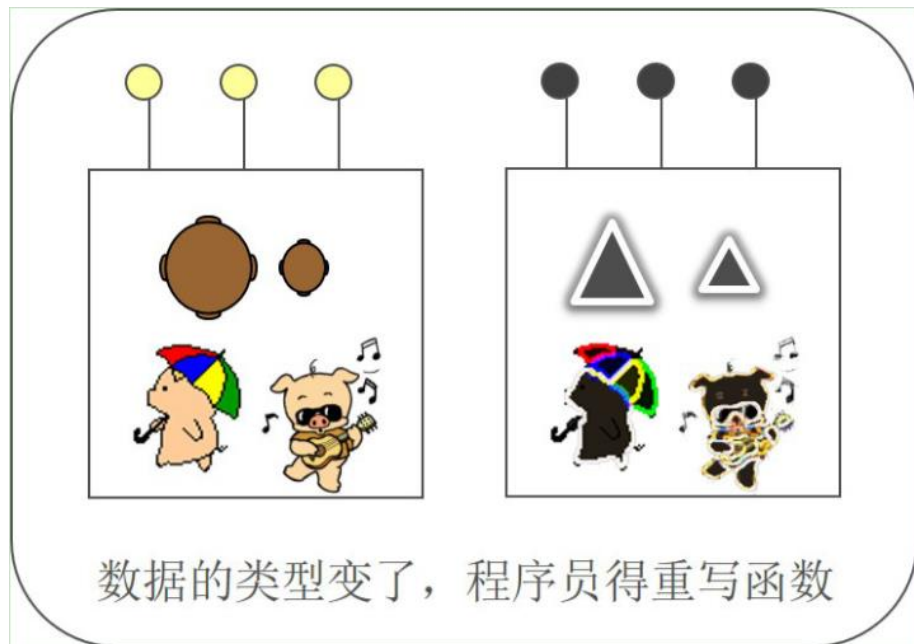


- 面向对象编程（OOP）



## 编程理念的图例：

- 泛型编程（GP）



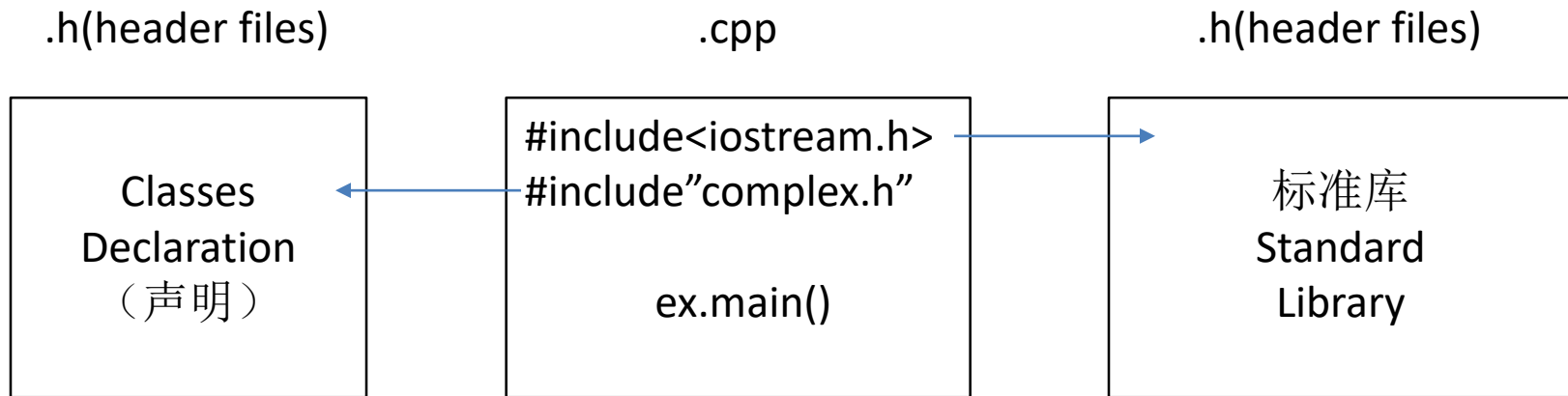
## C与C++的区别：

- ✚ C语言是一种结构化程序设计语言，侧重如何设计一个过程，对输入进行运算处理得到输出。
- ✚ C++语言是一种面向对象的语言，侧重如何构造一个与描述问题相对应的对象模型。
- ✚ C与C++解决问题的思想方法不一样。前者强调算法和数据结构，而后者融入了抽象设计的概念。

# 简单的C++程序

# 简单的C++程序

## C++代码基本形式



- 延申文件名不一定是.h或.cpp，也可能是.hpp或其他，甚至无延申名。



# 简单的C++程序

## C++注释

```
/*  
Our first C++ code  
Data:20220303  
*/
```

**多行注释：**对程序做详细说明

**注释的作用：**提高程序的可读性。

**注释的处理：**编译器对它不做处理，不生成目标代码。

**注释的种类：**

- ①用“/\*”和“\*/”把注解括起来，可出现在程序的任何位置，可做**多行注释**、**单行注释**和**嵌入注释**，常用于多行注释。
- ②用“//”表示从此开始到本行结束为注释，常用于单行注释。

## 示例1.1：面向过程程序设计。输入圆的半径，求该圆的面积

### 编译预处理命令

```
#include <iostream>
using namespace std;
int main()
{
    double r, s;
    cin >> r;
    s = 3.14 * r * r;
    cout << s << endl;
    return 0;
}
```

```
//文件包含命令
//使用std命名空间
//函数头

//定义r和s为double类型变量
//从键盘输入数据给变量r
//计算圆的面积并赋值给s
//屏幕输出s的值且光标移到下行行首
//程序暂停执行，按任意键继续
//函数返回0
```

函数

函数头

函数体

注释语句

## 示例1.1：输入圆的半径，求该圆的面积

由于C++语言没有专门的输入/输出(简称I/O)语句，此处借用了头文件`iostream.h`中预定义的标准输入对象`cin`(默认为键盘)和标准输出对象`cout`(默认为显示器)实现数据I/O，因此需要包含头文件`iostream.h`。

```
#include <iostream>
using namespace std;
```

```
int main()
{
    double r, s;
    cin >> r;
    s = 3.14 * r * r;
    cout << s << endl;
    return 0;
}
```

```
//文件包含命令
//使用std命名空间
//函数头
```

```
//定义r和s为double类型变量
//从键盘输入数据给变量r
//计算圆的面积并赋值给s
//屏幕输出s的值且光标移到下行行首
//程序暂停执行，按任意键继续
//函数返回0
```

为空或 (void)  
表示函数无参数

表示函数值  
为int类型

# 主函数框架

```
int main()
```

```
{
```

定义变量

```
double r , s;
```

输入已知

```
cin>>r;
```

求解处理

```
s=3.14*r*r;
```

输出结果

```
cout<<s<<endl;
```

```
return 0;
```

```
}
```

## 总结：

- (1) C/C++程序由一个或多个函数构成，main有且只有一个；
- (2) 无论一个程序中有多少个函数，执行总是从main函数开始；
- (3) 注释语句可增加程序可读性，以“//”开始，或以“/\*”开始，以“\*/”结束；
- (4) 分号是语句结束的标志；
- (5) 书写形式自由：  
一行内可以写多条语句，一条语句也可以分写在不同行上；
- (6) C和C++语言中区分大小写字母；

## 基本符号：

(1) 52个字母：

A~Z    a~z

(2) 10个数字字符：

0 ~9

(3) 下划线

—

(4) 特殊符号：

单、双引号及运算符，如+、-、\*、/、&

## 关键字：

系统已定义过的、有特定含义、不能它用的专用单词。

如：int、char、break、for、define

## 标识符：

这里专指用户自定义标识符，用于为程序中的变量、常量、函数取名。

- (1) 只能由字母、数字和下划线组成，且必须以字母或下划线开头；
- (2) 关键字不能作标识符用；
- (3) 定义标识符最好能简洁且“见名知意”，以提高程序的可读性；

如用average表示平均值，day表示日期。

示例：判断下面哪些可以用作合法的用户自定义标识符

2A   area\_of\_circle   A-B   M.D   Double   double

合法：

area\_of\_circle   Double

不合法：

2A   A-B   M.D   double



# 标准输入输出

# 标准输入输出

- ✚ 程序的输入输出是程序与用户沟通的桥梁，通过输入输出操作实现用户与程序的交互；
- ✚ 已知数据的获取表达最通用的方法是通过输入实现；
- ✚ 将求解结果呈现给用户则是通过输出实现；
- ✚ **标准输入输出**指系统指定的标准设备的输入输出，即从键盘输入，在显示器屏幕上输出；

## C和C++的输入输出

- ✚ C语言通过调用系统提供的标准库函数scanf和printf实现输入输出，而C++则是通过对系统提供的两个标准流类对象cin和cout的操作来实现输入输出的；
- ✚ C++语言为与C语言兼容，保留了用printf和scanf进行输出和输入的方法；
- ✚ C++类型安全性检查更严格、支持用户自定义类型的输入输出；
- ✚ C++实现简单的标准输入输出较C语言更方便、更适合初学者；本课程采用C++的标准输入输出表达。

# 数据输出

格式:

```
cout<<表达式1<<表达式2<<……<<表达式n;
```

说明:

- ✚ cout代表显示器，“<<”是插入运算符。上述语句意为将各表达式的值插入到显示器屏幕上，即输出各表达式的值；
- ✚ 各表达式可以是任意类型，数据的输出格式由系统自动决定；
- ✚ 注意表达式之间必须是“<<”分隔，而不能用逗号分隔；
- ✚ cout的定义信息存放在C++的输入输出流库中，故使用cout必须使用如下的文件包含预处理命令及名字空间说明语句：

```
#include <iostream>
```

```
using namespace std;
```

## 数据输出示例：

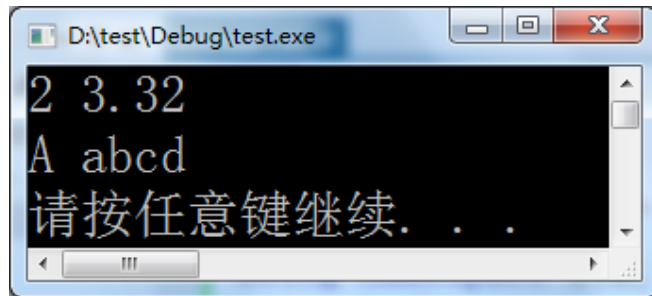
```
#include <iostream>
using namespace std;
int main()
{
```

```
    int a=2;
    double b=3.32;
    char c='A';
```

```
    cout<<a<<' '<<b<<endl;    //a和b的内容之间输出一个空格符
    cout<<c<<' '<<"abcd"<<endl; //""引起的字符序列是字符串
    return 0;
```

```
}
```

定义变量：  
a为整型，b为实  
型，c为字符型



格式控制符：换行

# 数据输入

格式:

```
cin>>变量1>>变量2>>……>>变量n;
```

说明:

- ✚ cin代表键盘，“>>”是提取运算符。上述语句意为从键盘提取数据分别给各变量，即输入各变量的值；
- ✚ 变量可为任意类型，表达式间必须“>>”分隔，而不能用逗号分隔；
- ✚ 输入数据时，各数据间用空格、换行、制表符分隔；
- ✚ 与cout不同，“>>”后必须是变量，故不能加endl
- ✚ cin的定义信息也存放在C++的输入输出流库中，故使用cin程序前需加：  
#include <iostream>  
using namespace std;

## 数据输入示例:

```
#include <iostream>
using namespace std;
int main()
```

```
{
```

```
    int a;
```

```
    double b;
```

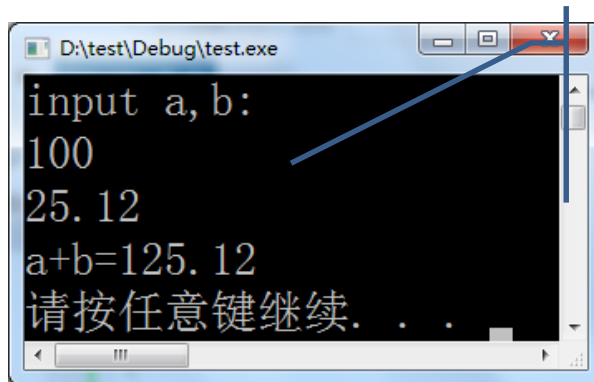
```
    cout<<"input a,b:";
```

```
    cin>>a>>b;
```

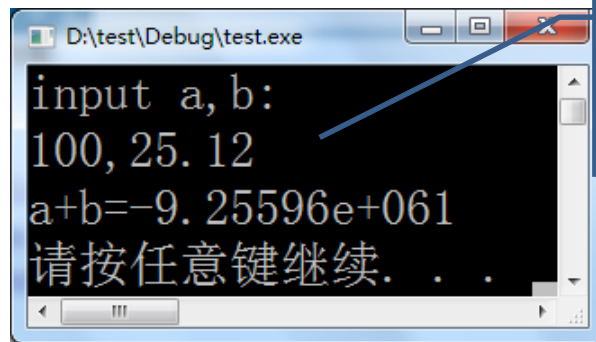
```
    cout<<"a+b="<<a+b<<endl;
```

```
    return 0;
```

```
}
```



回车符分隔



逗号分隔  
结果有误

## 2. cin

cin>>变量 1 >>变量2>>.....>>变量n;

说明：数据之间用空格、TAB键或回车分隔

```
#include <iostream>
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
    int a;
```

```
    float b;
```

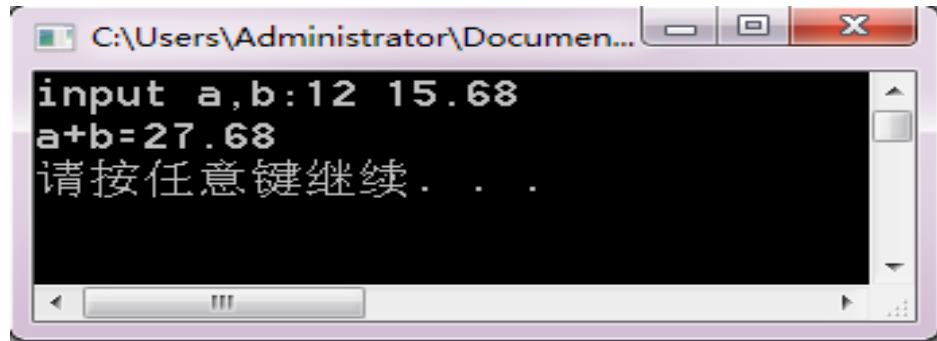
```
    cout<<"input a,b:";
```

```
    cin>>a>>b;
```

```
    cout<<"a+b="<<a+b<<endl;
```

```
    return 0;
```

```
}
```



```
input a,b:12 15.68
a+b=27.68
请按任意键继续...
```



## 面向对象程序设计的主要思路

- 对象是现实世界中客观存在的事物，复杂的对象可以由简单对象组成，如火车站对象由售票处、行李房、信号灯、站台等对象组成。
- 面向对象程序设计的主要思路是，把一个复杂问题看成一个复杂对象，将一个复杂对象按对象分解成若干个简单对象，每个简单对象通过定义一个类来解决，如果简单对象还不够简单，再继续分解下去，直到所有简单对象都能解决为止。这样解决一个复杂问题的对象就可以通过使用一系列解决简单对象的类来实现。这也是“自顶向下，逐步求精”的程序设计方法，只不过模块的基本单位是对象而已。

- 通过定义一个类来解决一个对象时，需要定义该对象的数据属性和函数属性，其中数据属性反映对象的状态，函数属性反映对象的行为。
- 数据属性通常不直接对外，以最大限度地保证对象行为的正常，这对于一个由许多对象组成的大型复杂系统来说是至关重要的。
- 函数属性是对象为外界提供服务的接口，它是按功能分解的函数，通常包括建立和初始化对象的构造函数、清理和撤消对象的析构函数、设置和获取数据属性的成员函数、以及解决实际问题的成员函数。

## 示例1.2 面向对象程序设计。输入圆的半径，计算并输出该圆的面积。

```
#include<iostream>
using namespace std;

class Circle{    //定义一个类，计算圆的面积
private:
    double r;    //定义成员数据变量，存放圆的半径
public:
    Circle(double a)//定义构造函数，创建和初始化对象
    {   r=a;   }
    ~Circle()    //定义析构函数，清理和撤消对象
    {   }
    void SetRadius(double a)//定义成员函数，设置圆的半径
    {   r=a;   }
```

class head

class body

```

double GetRadius( ) const//定义成员函数，获取圆的半径
{ return r; }
double Area( ) const    //定义成员函数，计算圆的面积
{ return 3.14*r*r; }
};

int main(void)
{ double r;    //定义浮点型变量r，用于存放圆的半径
  cout<<"输入圆的半径:";
  cin>>r;
  Circle c(r);    //定义Circle类的对象c
  cout<<"半径为"<<c.GetRadius( )<<"的圆的面积="
    <<c.Area( )<<'\\n';
  return 0;
}

```

## 示例1.2程序说明

- 主函数main将要解决的问题分解成三个简单对象，即输入对象cin输入圆的半径，自定义对象c计算圆的面积，输出对象cout输出圆的面积，最终解决整个问题。
- 定义一个类Circle，计算圆的面积。包括一个成员数据变量和五个成员函数。为了最大限度保证成员数据r的安全性和正确性，首先将数据成员的对外访问权限设定为私有的，即不允许外界直接访问，如程序第4行的“private:”所示；其次，设立了成员函数SetRadius和GetRadius为外界间接访问圆的半径r提供服务。而对于为外界提供特定服务的成员函数来说，其对外访问权限设定为公有的，如程序第6行的“public:”所示。

- 类是数据类型，与前面看到的int、double、float类似，但这是C++语言预定义的，用户可直接使用，而对于用户自定义的数据类型来说，必须先定义后使用。例如，程序的第3~17行定义了Circle类，程序的第23行定义了一个Circle类的对象c。
- 用面向对象的程序设计方法设计程序时，对于类的定义既要考虑到成员数据的安全性，又要考虑其通用性(即能解决一类问题)，还要考虑未来的代码可重用性，因此，类的功能通常是自我完善的，即自治的，尽管类的定义看起来有些"臃肿"。
- 有关类和对象的详细介绍，详见第8章节。

# C++程序的开发步骤和上机调试流程

## C++程序的开发步骤和上机调试流程

- C++集成开发环境(IDE)：集多种编程工具于一体，如源程序的编辑、编译、链接、运行、调试等，使用非常方便。例如 Visual Studio Community (C++)、Eclipse CDT+Gcc、Xcode (Mac)、Visual C++ 6.0等
- C++的编程环境支持C和C++程序的编译和调试。通常约定，当源程序文件的扩展名为".c"时，则为C程序；而当文件的扩展名为".cpp"时，则为C++程序。
- 本课程约定：
  - ①主要介绍标准C++语言及其程序设计。
  - ②本课程中所有程序都在VS2010或以上环境调试运行。
  - ③本课程中所有例题的源程序文件扩展名均为".cpp"。



## C++程序的开发步骤和上机调试流程

- 一个C++程序的具体开发步骤为：
- (1)分析问题，产生解题步骤，即解题算法。
- (2)根据解题算法编写C++源程序。
- (3)利用编辑器编辑源程序并保存。所存文件的扩展名为".cpp"。
- (4)编译源程序，并产生目标程序。在MS Windows操作系统中，目标程序文件的扩展名为".obj"。

- (5)链接。将一个或多个目标程序与本程序所引用的库函数进行链接后，产生一个可执行文件。在MS Windows操作系统中，可执行文件的扩展名为". exe"。
- (6)调试程序。运行可执行文件，分析运行结果。若结果不正确，则要修改源程序，并重复以上过程，直到得到正确的结果为止。
- (7)优化。进一步提高程序的运行效率，主要通过改进所用算法，缩短程序运行时间；通过合理分配使用内存，减少所用存储空间。
- 其中，第(1)、(2)两步在上机前完成；第(3)~(6)步在上机时完成，如图1-1所示；第(7)步在上机后思考。第(1)~(6)步是对所有学习编程者的基本要求，第(7)步适用于有兴趣、学有余力且有志成为编程高手者。

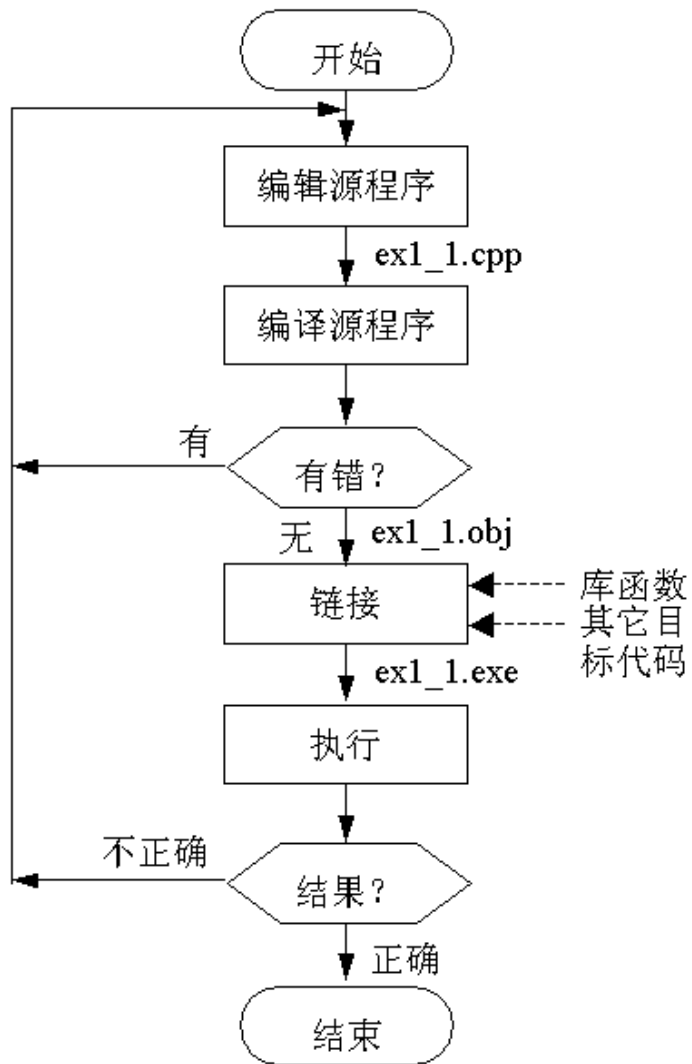
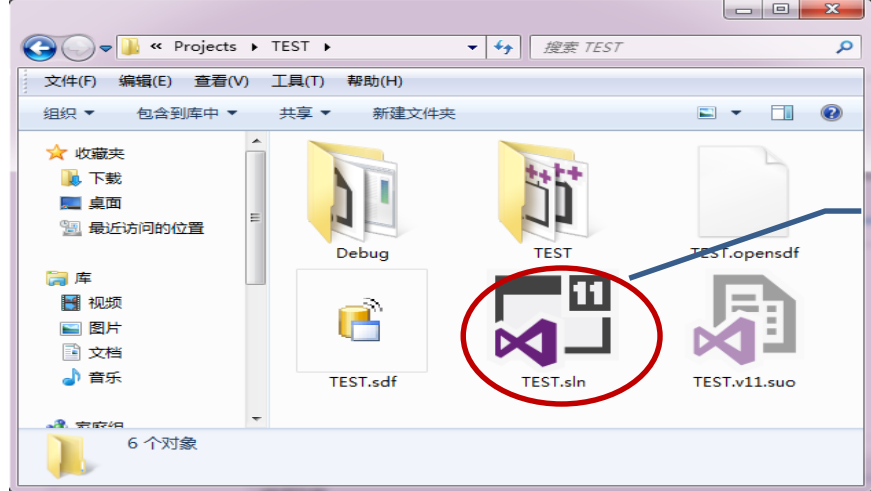
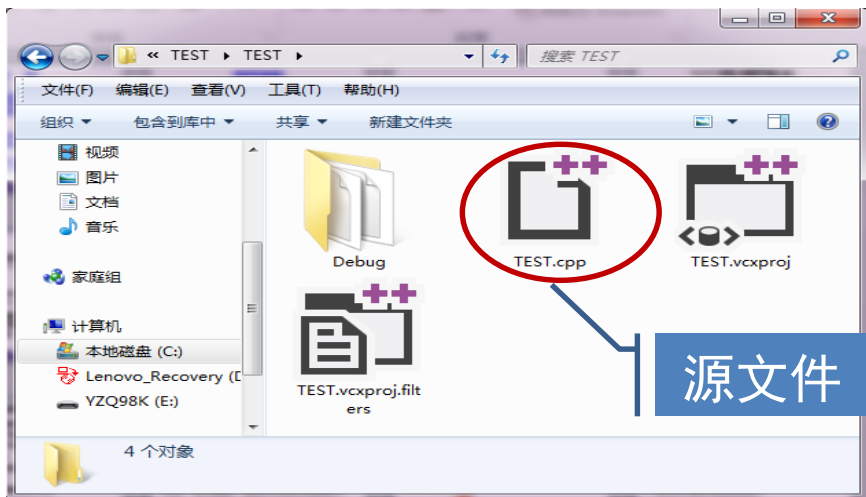


图1-1 C++程序上机调试流程

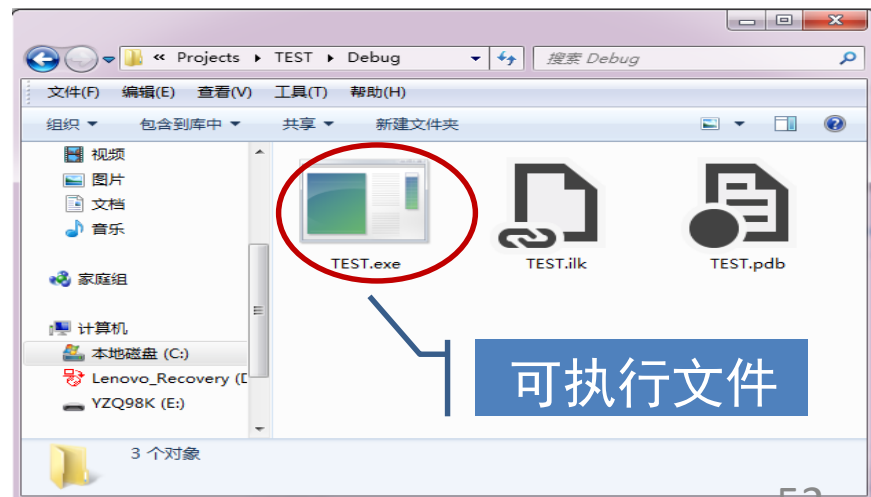
# 几个重要文件



解决方案文件



源文件



可执行文件

# 调试C++程序

# 程序调试



## 语法错误

- 错误(error): 无法成功编译, 需修改正确才能生成目标文件;
- 警告(warning): 可运行, 但结果可能不正确;

 逻辑错误: 程序可运行, 但出现结果与预期不一致的错误;

 运行错误: 在运行时产生的异常错误;

```
#include <iostream>
using namespace std;
int main( )
{
    int x,y;
    t=x;
    x=y;
    t=y;
    cin>>x>>y>>endl;
    cout<<"x="<<x<<"y="<<y<<endl;
    return 0;
}
```

# VS 编写调试C++程序步骤

## 1. 创建控制台应用或空项目

- ① 文件
  - 新建
  - 项目





## ② 选定：空项目

配置新项目

空项目 C++ Windows 控制台

项目名称(J)

Test

位置(L)

C:\Users\1\source\repos

解决方案名称(M) ⓘ

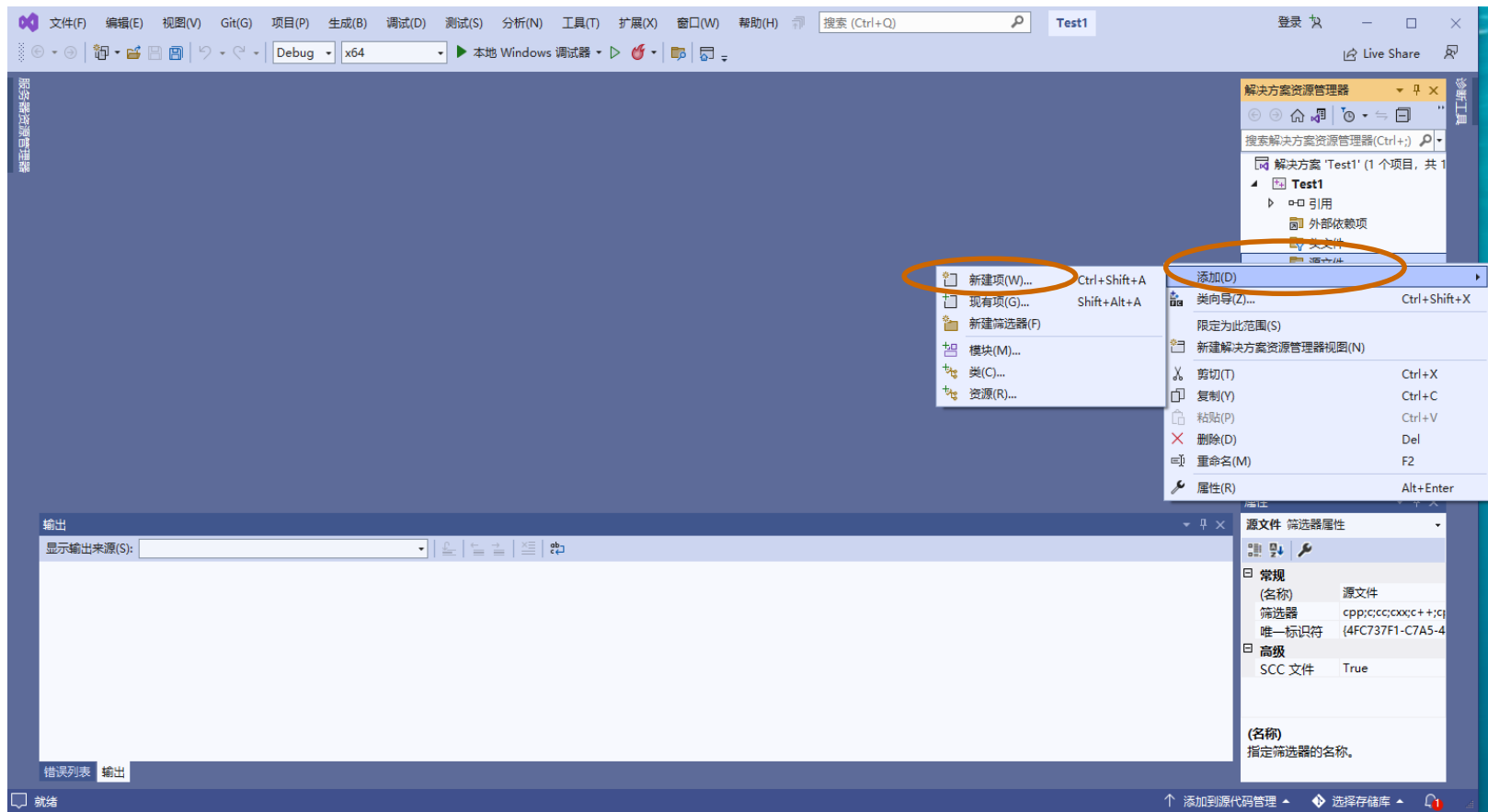
Test

☒ 将解决方案和项目放在同一目录中(D)

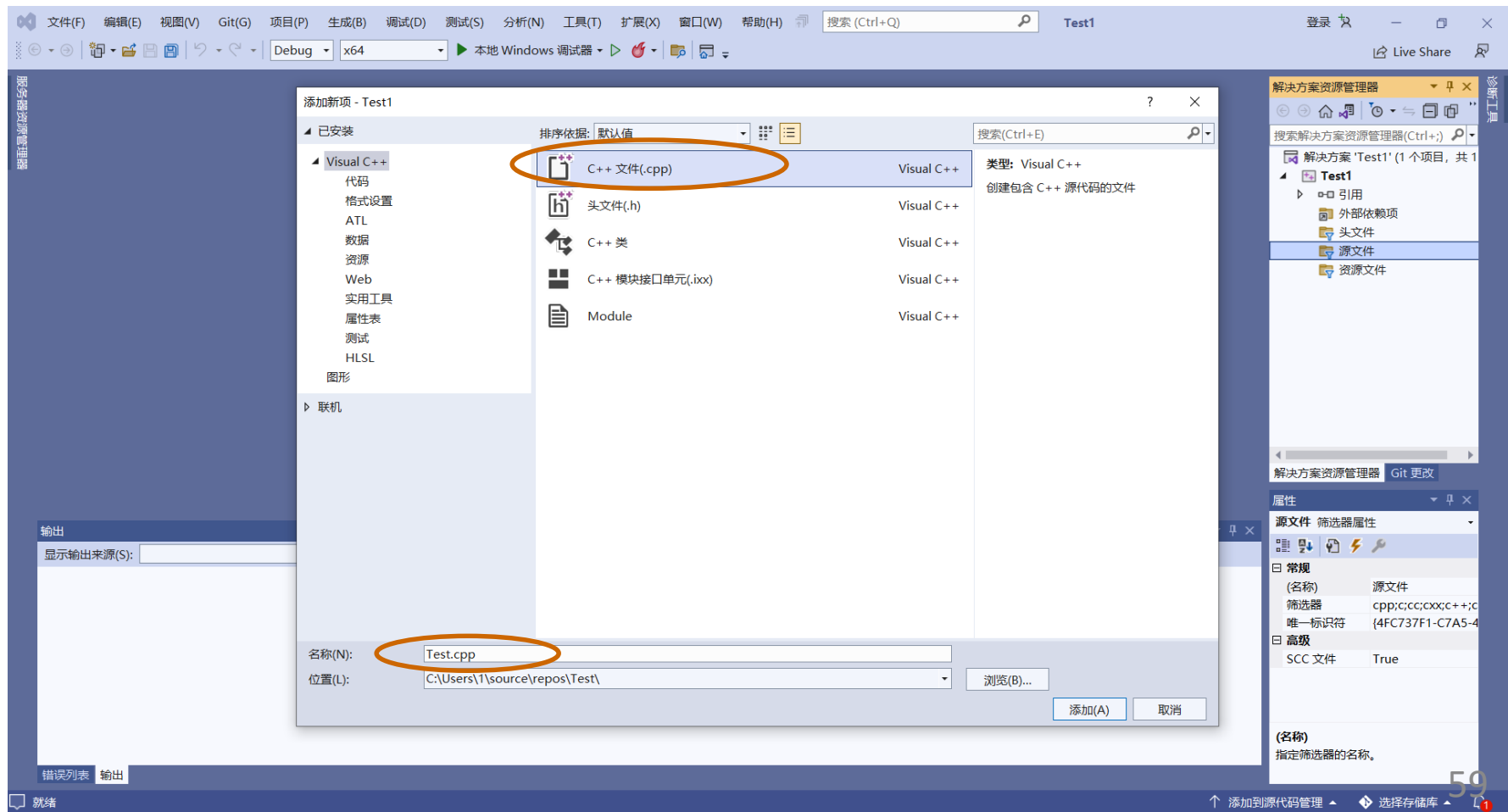
上一步(B) 创建(C)

## 2. 添加C++文件(.cpp)

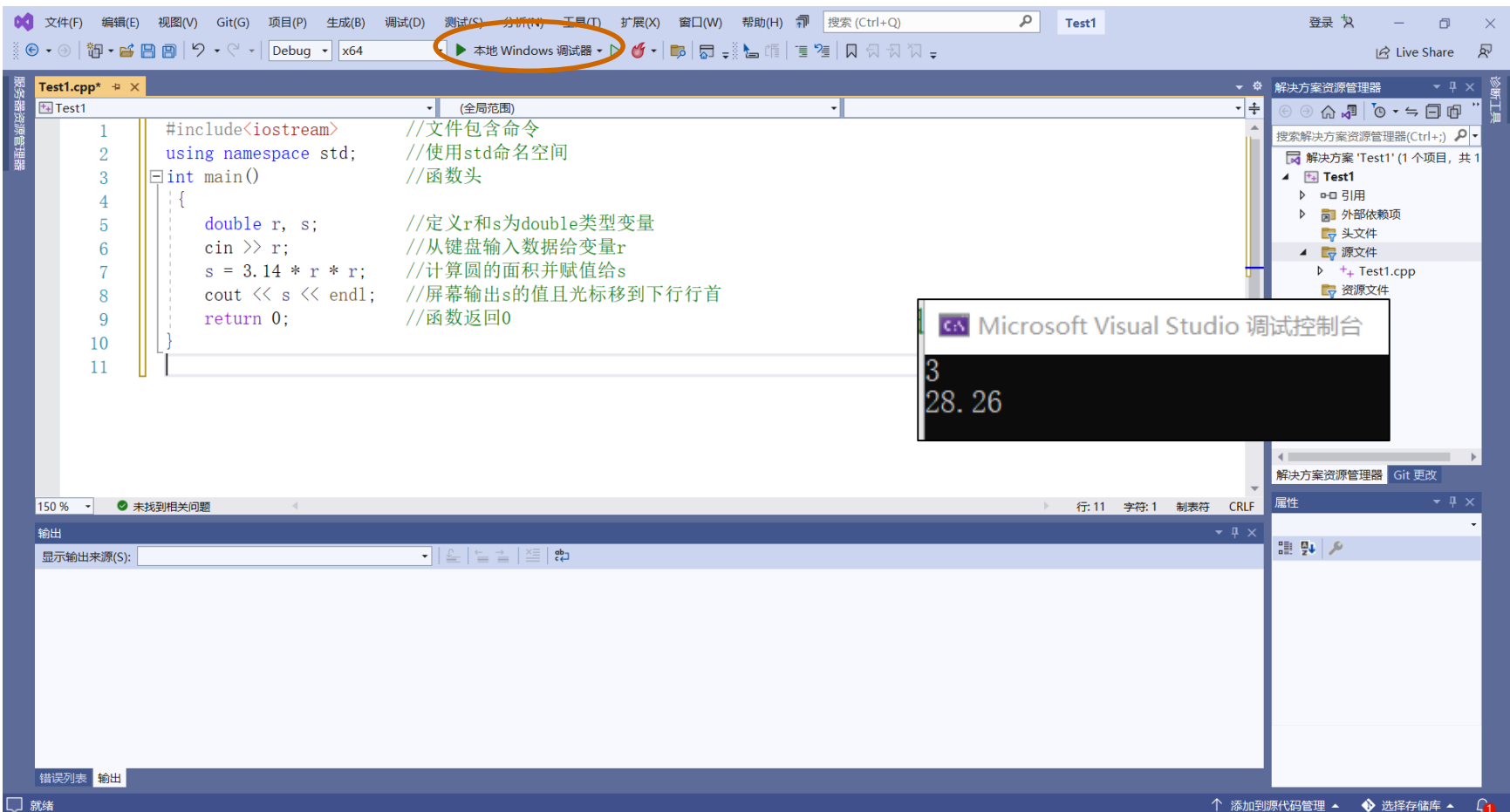
### ① 源文件的快捷菜单 | 添加 | 新建项



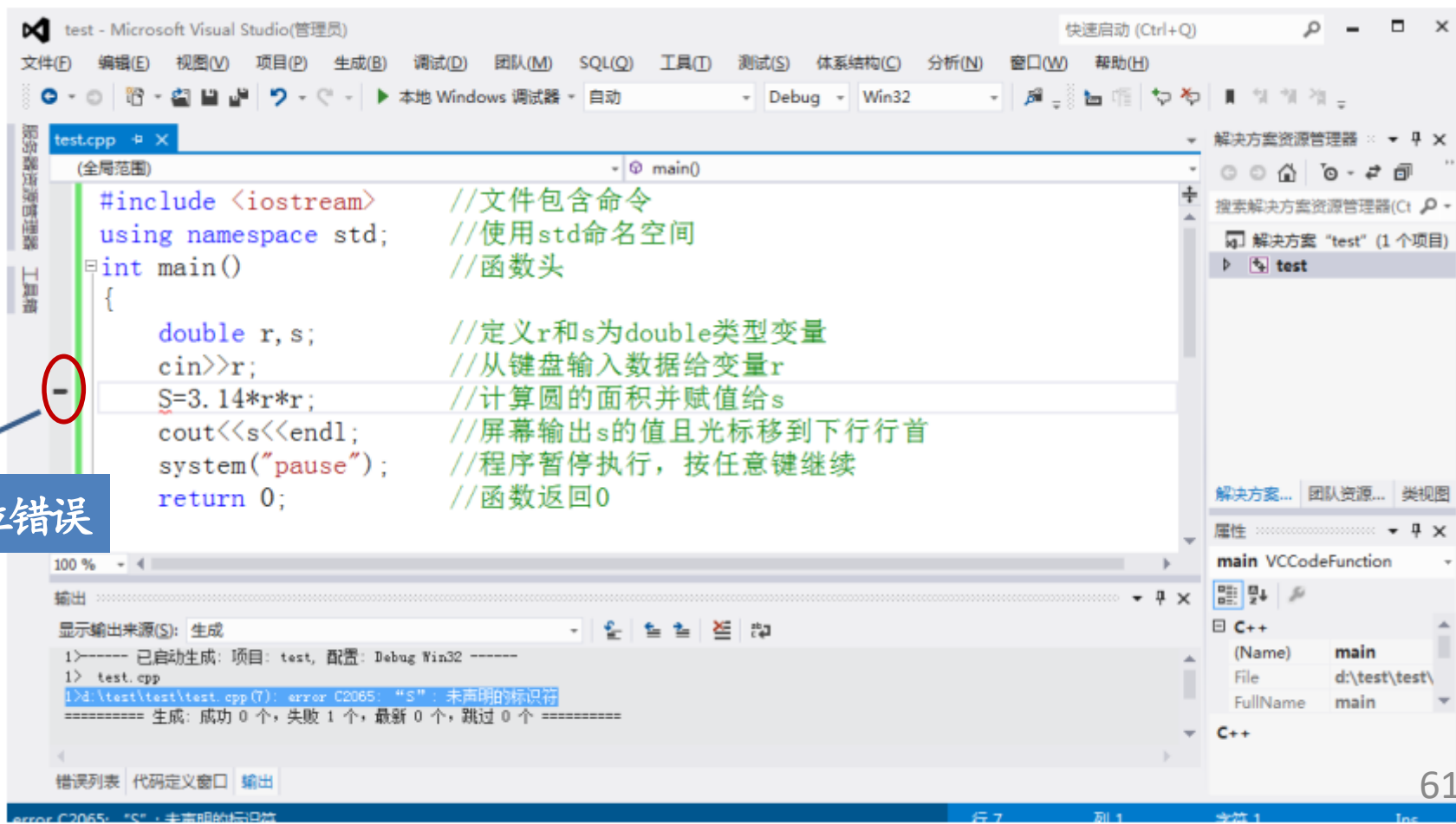
## ② 选定：C++文件(.cpp)



## 4. 运行程序



# 例，语法错误：未定义的标识符



# 输入源程序

test - Microsoft Visual Studio(管理员)

快速启动 (Ctrl+Q)

文件(F) 编辑(E) 视图(V) 项目(P) 生成(B) 调试(D) 团队(M) SQL(Q) 工具(T) 测试(S) 体系结构(C) 分析(N) 窗口(W) 帮助(H)

本地 Windows 调试器 自动 Debug Win32

test.cpp\* x

(全局范围)

```
#include <iostream> //文件包含命令
using namespace std; //使用std命名空间
int main() //函数头
{
    double r,s; //定义r和s为double类型变量
    cin>>r; //从键盘输入数据给变量r
    s=3.14*r*r; //计算圆的面积并赋值给s
    cout<<s<<endl; //屏幕输出s的值且光标移到下行行首
    system("pause"); //程序暂停执行，按任意键继续
    return 0; //函数返回0
}
```

源程序

解决方案资源管理器

搜索解决方案资源管理器(Ctrl+Shift+F)

解决方案 "test" (1 个项目)

test

解决方案... 团队资源... 类视图

属性

错误列表

0 个错误 | 0 个警告 | 0 个消息

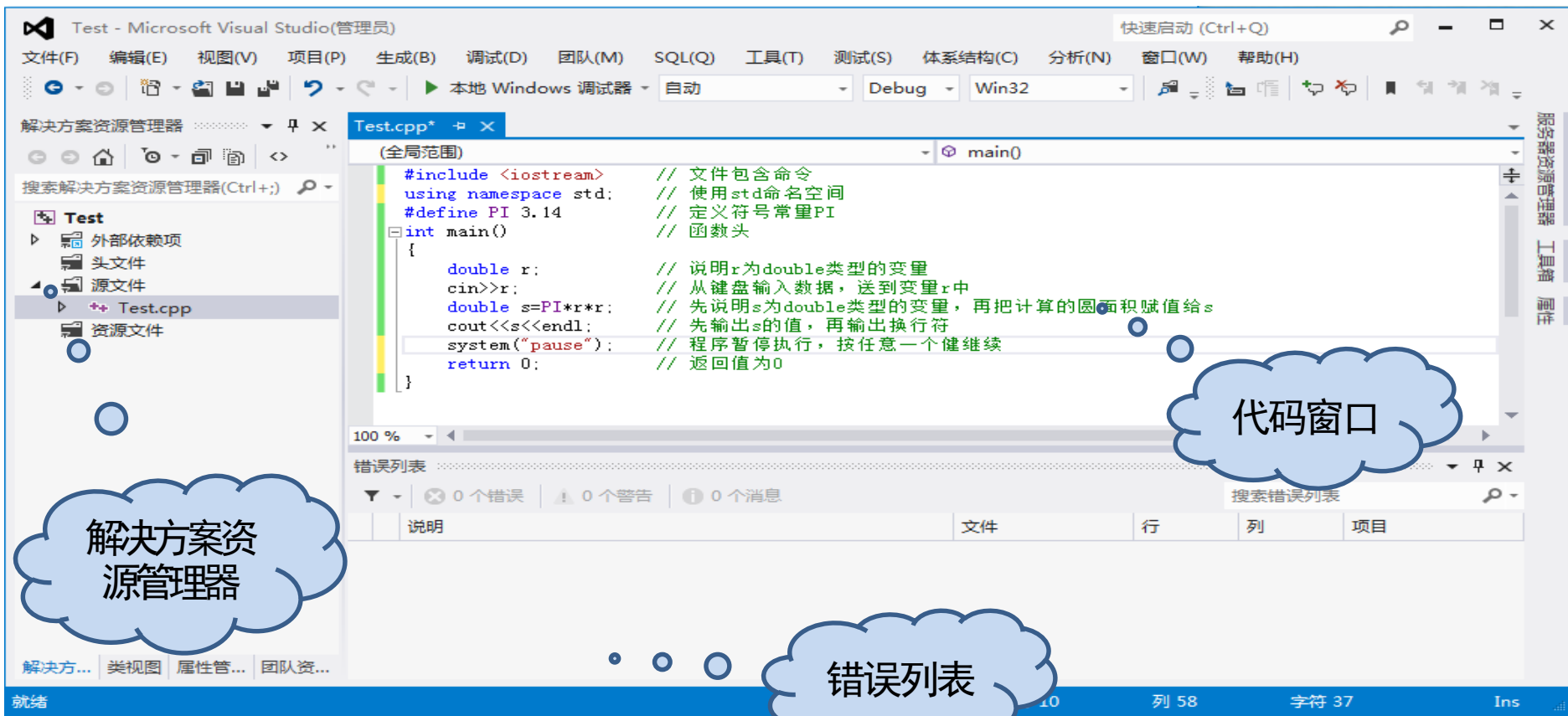
搜索错误列表

说明	文件	行
----	----	---

错误列表 | 代码定义窗口 | 输出

就绪 行 1 列 1 字符 1 Ins

# Visual Studio C++



作业:

1. 编写一个计算梯形面积的程序。要求梯形的上底、下底和高在定义变量时直接赋值。
2. 编写计算一个学生三门课平均成绩的程序,要求学生成绩从键盘输入。