高等学校电子信息类“十二五”规划教材

C++语言简明教程

主编　吴延海　黄 健

西安电子科技大学出版社
http://www.xduph.com

高等学校电子信息类专业“十二五”规划教材

C++语言简明教程

主　编　吴延海　黄　健

副主编　唐善成　刘晓佩

李新民　吴　楠

西安电子科技大学出版社

内 容 简 介

全书共 8 章，内容包括 C 语言基本语法、C++ 基本语法特征、类和对象、类的继承性、类的多态性、友元和运算符重载、贪吃蛇游戏设计及学籍管理系统。

附录介绍了 Visual C++ 集成开发环境和 Visual C++ 下的程序调试。

本书内容由浅入深，通俗易懂，突出重点，偏重应用。书中每章均辅以参考范例和课后习题，便于读者自学和参考。

本书可作为高等学校非计算机类专业的本科生教材，还可作为 C++ 语言的自学教材和参考书。

图书在版编目(CIP)数据

C++语言简明教程/吴延海，黄健主编. —西安：西安电子科技大学出版社，2012.8
高等学校电子信息类专业“十二五”规划教材
ISBN 978-7-5606-2870-7

Ⅰ. ① C… Ⅱ. ① 吴… ② 黄… Ⅲ. ① C 语言—程序设计—高等学校—教材 Ⅳ. ① TP312

中国版本图书馆 CIP 数据核字(2012)第 163206 号

策　　划　毛红兵
责任编辑　马武装　毛红兵
出版发行　西安电子科技大学出版社(西安市太白南路 2 号)
电　　话　(029)88242885　88201467　　邮　　编　710071
网　　址　www.xduph.com　　电子邮箱　xdupfxb001@163.com
经　　销　新华书店
印刷单位　陕西天意印务有限责任公司
版　　次　2012 年 8 月第 1 版　2012 年 8 月第 1 次印刷
开　　本　787 毫米×1092 毫米　1/16　印　张　15.5
字　　数　360 千字
印　　数　1～3000 册
定　　价　26.00 元

ISBN 978-7-5606-2870-7/TP • 1357

XDUP 3162001-1

前　言

本书是作者在多年从事 C++ 语言教学的基础上编写的。针对学生在学习过程中遇到的困惑和问题，书中精简了 C++ 语言的概念和定义，知识点以够用为度。本书内容由浅入深、语言简洁，概念明确，通俗易懂；在结构上，删减掉了一些过于复杂和不常用的知识点，知识点分布连贯，突出重点，注重应用。

本书共分 8 章，系统地讲述了 C++ 语言的基本概念、基本语法，编程的基本方法和应用实例的开发。

第 1 章 C 语言基本语法。介绍了 C 语言的数据类型；C 语言的 5 种基本语句：赋值、自增自减、分支、循环和流程控制语句；函数输入、输出型参数；系统库函数和用户自定义函数。此外，举例介绍了指针、动态内存分配和链表。

第 2 章 C++ 基本语法特征。介绍了 C++ 语言数据类型中的枚举类型和结构体类型，以及输入输出系统、变量作用域、引用类型、const 类型等。讨论了 C++ 语言相比 C 语言的改进之处。

第 3 章类和对象。介绍了 C++ 语言中类的声明与定义、成员函数及类成员访问控制；对象的引用和指针、对象的初始化与构造函数、对象的复制与拷贝构造函数及对象的销毁与析构函数。在静态成员中，介绍了静态成员变量和静态成员函数。

第 4 章类的继承性。介绍了 C++ 语言中继承的本质、方式及实现；子类对象的初始化与构造函数、子类对象的销毁与析构函数及子类和父类对象的指针；类的多重继承、父类成员名冲突、父类分解和虚继承。

第 5 章类的多态性。介绍了 C++ 语言多态性概念、多态性种类；C++ 语言运行时的多态性。重点介绍了滞后联编、虚函数、虚函数和覆盖继承的区别以及纯虚函数和抽象类。

第 6 章友元和运算符重载。介绍了 C++ 语言友元概念，定义友元的方法：友元函数、友元类和友元成员函数。重点介绍了运算符重载为类的成员函数方法、运算符重载为类的友元函数方法以及两种运算符重载方法的比较。

第 7 章贪吃蛇游戏设计。介绍了游戏设计思路及各模块的定义、实现和测试，内容包括：控制台屏幕输出控制类模块、游戏地图数据封装类模块、贪吃蛇类对象模块、游戏引擎封装类模块等。

第 8 章学籍管理系统。介绍了系统的设计思路及各模块的定义与设计，内容包括链表存储基类、学生派生类节点、班级派生类节点、翻页显示工具栏、菜单与输入控制，最后介绍了系统功能集成与测试。

本书由吴延海主编，黄健编写第 8 章，唐善成编写第 3、4 章，刘晓佩编写第 7 章、附录 A、附录 B，李新民编写第 5、6 章，吴楠编写第 1、2 章，全书由吴延海、黄健统稿。本书在编写过程中得到了作者单位的支持和其他同事及研究生的帮助，同时也得到了西安

电子科技大学出版社的大力支持，特别是参与本书编辑的马武装、毛红兵等为此书的出版付出了辛苦的劳动，在此一并表示感谢。

鉴于作者水平有限，书中难免存在错误和不妥之处，恳求读者批评指正。

编 者

2012 年 7 月

目　录

第 1 章　C 语言基本语法

本书主要讨论 C++ 语言，而 C++ 语言是在 C 语言基础上发展起来的一种混合了面向过程语言要素和面向对象语言要素的程序设计语言。本章首先对 C 语言程序设计的基础知识进行简单的介绍。

1.1　数 据 类 型

1.1.1　基本数据类型

数据类型规定了一类数据的数据位长度(或字节个数)、取值范围以及对该类数据所能进行的操作。

基本数据类型是系统已定义的数据类型。C 语言共定义了 7 种基本数据类型，其中 4 种为整型数，2 种为浮点型数，1 种为字符型数。数据类型不同，所定义的变量占用的内存空间、取值范围以及对该类数据所能进行的操作也不同。

C 语言定义的 7 种基本数据类型及相应的关键字如下：

- 整型：byte、short、int、long；
- 浮点型：float、double；
- 字符型：char。

C 语言的基本数据类型、字节数和取值范围如表 1.1 所示。

表 1.1　基本数据类型、字节数和取值范围

数据类型关键字	字节数	取值范围
byte	1	−128～127，即-2^7～2^7-1
short	2	−32 768～32 767，即-2^{15}～$2^{15}-1$
int	4	−2 147 483 648～2 147 483 647，即-2^{31}～$2^{31}-1$
long	8	-2^{63}～$2^{63}-1$
float	4	3.4e−038～3.4e+038
double	8	1.7e−038～1.7e+038

C 语言的字符串用字符数组表示，如语句：

```
char s[20]="Hello World!";
```

该语句定义了长度为 20 的字符数组变量 s，且该变量的初始值为“Hello World!”。另外，C 语言还有空类型，其关键字为 void。空类型主要用来定义函数返回值的类型。

1.1.2 构造数据类型

构造数据类型是根据已定义的一个或多个数据类型用构造的方法来定义的。也就是说，一个构造类型的值可以分解成若干个“成员”或“元素”。每个“成员”都是一个基本数据类型或又是一个构造类型。在 C 语言中，构造类型有数组、结构体和共用体。下面分别对其进行简单的介绍。

1. 数组

数组是一个由若干相同类型变量组成的集合。在 C 语言中，数组元素用数组名后面跟带方括号的下标表示，例如：a[10]，b[3][3]，c[2][3][4] 均为 C 语言数组元素。根据数组元素的下标的个数，分为一维数组、二维数组和多维数组。

数组的声明如下：

```
类型标识符 数组名[常量表达式 1][常量表达式 2]…[常量表达式 n];
```

例如：

```
int a[5];
float b[3][5];
```

2. 结构体

前面已经介绍了基本数据类型，但是只有这些数据类型是不够的。有时需要将不同类型的数据组合成一个有机的整体，以便于引用。比如每位学生都具有学号、姓名、年龄、成绩等属性，而一个班级的学生在学号上又具有一定的联系。C 语言中，解决这样的问题是通过结构体数据类型。结构体的一般定义语句为

```
struct 结构体名
{
    成员列表;
};
```

其中，struct 是 C 语言的关键字，是结构体类型的标志；结构体名是结构体标识符；成员列表中的每一项都由已定义数据类型名和成员名两部分组成。由于结构体中所有成员的数据类型都是已定义的，因此可以把结构体看做一个新的、用户自定义的数据类型。换句话说，一旦定义了一个结构体，就可以用该结构体定义变量。

例如，要处理学生信息时，加入要处理的学生信息，包括学生的学号、姓名、性别、年龄，就可以把学生的这些信息定义成一个结构体。结构体定义的语句如下：

```
struct student
{
   long number;              //学号
   char name[20];            //姓名
   char sex[3];              //性别
    int age;                 //年龄
};
```

对结构体类型的变量，既可以整体处理，也可以按成员分量处理。整体处理的例子

如下：

```
struct student x={100001，"张三"，"男"，26}，y，*p;
y=x;                                    //结构体赋值
p = &x;                                 //结构体地址赋值
```

按成员分量处理的例子如下：

```
struct student x= {100001，"张三"，"男"，26}，y，z，*p;
y.number = x.number;                    //变量的成员分量赋值
p=&x;                                   //结构体地址赋值
z.number = p->number;                   //指针类型变量的成员分量赋值
```

注意：这里指针类型变量的成员表示方法和非指针类型变量的成员表示方法不同。

3. 共用体

共用体也是一种构造类型，它的主要特点是，共用体变量中的所有成员占用同一段存储空间，这段空间的大小就是所有成员中所需存储数的最大者。而结构体变量中的成员各自占用自己的存储空间，这是两者的本质区别。另外，共用体类型说明及变量定义都与结构体类型说明及变量定义的方式类似。

共用体类型说明形式：

```
union 共用体类型名
 {
   成员列表;
}
```

其中，union 为 C 语言关键字，共用体类型名只要符合 C 语言标识符命名规则即可。共用体变量的定义与结构体变量的定义方式类似，例如：

```
union un1
 {
    int a;
    char b;
    float c;
} x;
```

由此可以看出，共用体“union un1”共有 3 个成员，而成员 c 所需的存储空间最大，是 4 个字节。所以共用体变量 x 共占用存储空间 4 个字节，它的 3 个成员共享此段空间。

需要说明的是：

(1) 在某一时刻，这段空间中只能存储一个成员的数据，这个数据就是最后一次赋予的值。

(2) 不能对共用体变量进行初始化，也不能进行整体赋值运算。

例如：

```
x.a =10;
x.b = 'e';
x.c =80.2;
```

则这段存储空间保存的值是最后一次赋予的数据 80.2。

1.2 基本语句

和其他的高级语言一样，C 语言的语句用来向计算机发出操作指令，一条语句编译后产生若干条机器指令，一个实际的程序由若干条语句组成。C 语言的基本语句包含赋值语句、自增自减语句、分支语句、循环语句和流程控制语句。

1.2.1 赋值语句

赋值语句是由赋值表达式加上一个分号构成的，其基本的语句语法为

<变量> <赋值运算符> <表达式>;

其中，最常用的赋值运算符是“=”。另外，还有复合运算符 +=、-=、/=、*= 等。表达式分为算术表达式、关系表达式和逻辑表达式。赋值表达式中的表达式主要是算术表达式。算术表达式是由算术运算符组成的表达式，例如：

```
int b=-1, a=10;                    //变量定义并赋初值
b = a * 5 + 3 ;                    //把算术表达式 a*5+3 的值赋给变量 b
b -= a;                            //把算术表达式 b-a 的值赋给变量 b
```

语句 b -= a; 也可写成：

```
b = b-a;
```

1.2.2 自增自减语句

自增自减语句是特殊情况下赋值语句的简略形式，其作用是使变量的值加 1 或减 1。例如：

```
int i=3, j=0;          //定义变量并赋值
i--;                   //使用 i 后，使 i 的值减 1
++j;                   //使用 j 前，使 j 的值加 1
```

其中，i--和++j 的作用相当于 i = i-1 和 j = j+1。自增自减语句常用于循环语句中，使循环变量自动加 1；也用于指针变量，使指针指向下一个地址。

1.2.3 分支语句

分支语句用来根据不同的条件构造不同的语句执行流程。分支语句包括 if 语句和 switch 语句。if 语句的基本语法形式为

```
if(<条件>)
    <语句>
或
if(<条件>)
    <语句 1>
else
    <语句 2>
```

第一种分支语句的含义是：若条件成立，则执行语句，否则跳过执行后续语句；第二种分支语句的含义是：若条件成立，则执行语句 1，否则执行语句 2。

条件是由关系表达式或逻辑表达式组成的一个其值为真(非 0)或为假(0)的表达式。

当条件后面的语句多于一条时，要用一对花括号“{ }”把这些语句括起来。

【例 1.1】　编写一个程序，输入一个实数，输出它的绝对值。

```
//-------------------------
//c0101.cpp
//-------------------------
#include <stdio.h>
main()
{
   float x,y;
   printf("Input x: ");
   scanf(" %f",&x);
   if(x>=0)
       y= x;
   else
       y= -x;
   printf("|%f|=%f\n", x,y);
}
```

程序运行结果：

(1)

```
Input x:  3.2↙
|3.200000| = 3.200000
```

(2)

```
Input x: -3.2↙
|-3.200000| = 3.200000
```

if 语句也可以嵌套使用，即可以在一个 if 语句中又包含另一个 if 语句，从而构成程序执行的多个分支。但是在大多数情况下，当程序存在多个分支时，一般使用 switch 语句。switch 语句的基本语法形式为

```
switch(<表达式>)
{
   case <常量 1>: <语句 1>
   case <常量 2>: <语句 2>
      ⋮
   case <常量 n>: <语句 n>
   default      : <语句 n +1>
}
```

例如，要求按照考试成绩的等级输出百分制分数段，可以用 switch 语句实现：

```
switch(grade)
{
    case 'A': printf ("90～100\n");
    case 'B': printf ("80～89\n");
    case 'C': printf ("70～79\n");
    case 'D': printf ("60～69\n");
    case 'E': printf ("<60\n");
    default : printf ("error\n");
}
```

switch 语句中，当表达式的值与某一个 case 后面的常量表达式的值相等时，就执行此 case 后面的所有语句；如果表达式的值与任何 case 后面的常量都不匹配，则执行 default 语句后面的语句。

1.2.4 循环语句

循环语句用来构造满足一定条件时，对同一个程序段重复执行若干次的程序结构。

循环语句主要有 while 语句和 for 语句两种。while 语句主要用来构造循环次数不固定的循环，for 语句主要用来构造循环次数固定的循环。实际上，这两种循环语句的功能完全一样，也就是说，既可以把 while 语句构造的循环结构改造成 for 语句构造的循环结构，也可以把 for 语句构造的循环结构改造成 while 语句构造的循环结构。

while 语句的语法形式为

```
while(<表达式>)
    <循环体>
```

当表达式为非 0(代表逻辑“真”)时，反复执行 while 语句中的内嵌语句，直到表达式为 0 为止。

for 语句的语法形式为

```
for(<初始表达式>; <终止表达式>; <增量表达式>)
    <循环体>
```

其循环执行的过程为

(1) 先求解初始表达式。

(2) 求解终止表达式，若其值为“真”(值为非 0)，则执行 for 语句中指定的内嵌语句，然后执行下面(3)步。若为“假”(值为 0)，则结束循环，转到(5)步。

(3) 求解增量表达式。

(4) 转回上面第(2)步继续执行。

(5) 循环结束，执行 for 语句下面的一个语句。

【例 1.2】　编写程序求 $\sum_{i=1}^{100} i$，即 1+2+3+ … + 100。

```
//-------------------------
//c0102.cpp
```

```
//-------------------------
#include <stdio.h>
main()
{  int i;
   long sum=0;
   for(i=1;i<101;i++)
      sum += i;
   printf("sum=%ld\n", sum);
}
```

由于本例的循环次数事先已知，所以用 for 语句实现较好。为了说明 while 语句和 for 语句的功能完全相同，再用 while 语句设计如下：

```
#include <stdio.h>
main()
{  int i=1;
   long sum =0;
   while(i<=100)
      {   sum+=i;
          i++;}
   printf("sum=%ld\n", sum);
}
```

两个程序的运行结果均为

```
sum = 5050
```

循环语句还有 do-while 语句，其实现功能和 while 语句完全相同，只是语法略有不同。

1.2.5 流程控制语句

流程控制语句主要有 break 语句和 continue 语句。

break 语句的语法形式为

```
break;
```

其语句含义是跳出当前的 switch 语句或循环语句。

continue 语句的语法形式为

```
continue;
```

其语句含义是结束本次循环，即跳过循环语句中尚未执行的语句，接着进行循环条件的判定。continue 语句只用在 for、while 和 do-while 等循环语句中，一般与 if 语句一起使用，可以加速循环。

【例 1.3】 设计一个程序完成以下功能：若输入英文字母 Y 或 y，则继续执行；若输入英文字母 N 或 n，则结束；若输入其他字符，则不做任何处理。

```
//-------------------------
//c0103.cpp
//-------------------------
```