第一章　C++语言简介

考点一：初识 C++语言的特点

一般称现实世界中客观存在的事物为对象。C++语言是一种编译式的、通用的、大小写敏感的编程语言，完全支持面向对象的开发模式。

1、基本的输入/输出

C++将数据从一个对象流向另一个对象的流动的抽象称为“流”。从流中获取数据的操作称作为提取操作，向流中添加数据的操作称为插入操作。其中“>>”为流提取符，从标准输入设备取得数据；“<<”为流插入符，向输出设备屏幕输出信息。

C++类库中提供了输入流类 istream 和 ostream。cin 和 cout 分别是 istream 和 ostream 类的对象，用来实现基本的键盘输入和屏幕输出。

cin 的一般格式如下：

cin >> 变量 1 >> 变量 2 >>…… >>变量 n;

cout 的一般格式如下：

cout << 表达式 1 << 表达式 2 <<……<<表达式 n;

2、头文件

C++与 C 语言类似，也使用头文件保存程序中用到的声明，例如函数声明、常量定义等。每个头文件都用 include 指令包含，每条#include 指令仅包含一个头文件，如果需要包含多个头文件，则需要使用多条 #include 指令。

除了可以使用系统提供的头文件外，程序员还可以定义自己的头文件，并在程序中使用#include 指令将其包含起来。通常，使用尖括号括住系统提供的头文件，使用双引号括住程序员自己定义的头文件。

3、命名空间

所谓命名空间（namespace）也称为名字空间，是一种将程序库名称封装起来的方法，它提高了程序的性能和可靠性。

C++提供了 using 语句，可以简化前面的写法。using 语句有两种形式：

using 命名空间 :: 标识符;

using　namespace　命名空间;

4、函数参数的默认值

在 C++语言中，可以在声明函数时为形参指定默认值。当调用默认参数值的函数时，调用语句中可以不给出对应的实参，这就相当于调用该函数时以默认值作为参数。

C++语言规定，提供默认值时必须从右到左的顺序提供，即默认值的形参必须在形参列表的最后。调用函数时，主调函数的实参与被调函数的形参按从左至右的顺序进行匹配对应。如果实参的个数与形参的个数相等，则它们一一对应。如果实参的个数 m 少于形参的个数 n，则函数原型形参表中最前面的 m 个形参与 m 个实参相对应，后面的 n-m 个形参则使用默认值进行初始化。

5、引用

引用相当于给变量起了一个别名。变量对应于某个内存地址，如果给某个变量起了别名，相当于变量和这个引用都对应到同一地址。因此，使用引用时没有分配新的存储区域，它本身并不是新的数据类型。

“引用”的定义格式如下：

类型名 &引用名 = 同类型的变量名;

引用通常用于函数的参数表中或作为函数的返回值。对引用实质性的理解如下：

（1）在 C++中，函数调用时参数的传递有两种方式：传值和传引用。传值，实际上是传递对象的值；传引用是传递对象的首地址值。在传值调用函数的情况下，形参是实参的副本，形参的改变不会影响到实参；在传引用调用函数的情况下，形参的改变就意味着实参的改变。

（2）引用作为函数的返回值。返回引用的函数原型格式如下：

数据类型 &函数名(参数列表);

6、const 与指针共同使用

当 const 与指针共同使用时，其书写的位置不同，语句含义也不同。const 修饰指针变量时，基本含义如下：

（1）如果唯一的 const 位于符号*的左侧，表示指针所指数据是常量，数据不能通过本指针改变，但可以通过其他方式进行修改；指针本身是变量，可以指向其他的内存单元。

（2）如果唯一的 const 位于符号*的右侧，表示指针本身是常量，不能让该指针指向其他内存地址；指针所指的数据可以通过本指针进行修改。

（3）在符号*的左右各有一个 const 时，表示指针和指针所指数据都是常量，既不能让指针指向其他地址，也不能通过指针修改所指向的内容。

7、内联函数

使用内联函数，编译器在编译时并不生成函数调用，而是将程序中出现的每一个内联函数的调用表达式直接用该内联函数的函数体进行替换。它的定义在前，调用在后，定义时只需在函数头返回值类型的前面加上关键字 inline。定义格式如下：

inline 返回值类型 函数名（形参表）

{

函数体

}

内联函数主要应用于代码量少的函数，编译时编译程序将整个函数体的代码复制到调用该函数的位置，而不会编译成函数调用的指令。如果函数体中有循环语句和 switch 语句则通常不定义为内联函数。

8、函数的重载

所谓函数的重载，是指在程序的同一范围内声明几个功能类似的同名函数。函数名加上参数表称为函数的签名。编译器会根据重名的各个函数的签名，来确定调用的具体函数版本。

实现函数的重载必须满足下列条件之一：

- 参数表中对应的参数类型不同。
- 参数表中参数个数不同。

9、指针和动态内存分配

指针变量中保存的是一个地址，也可称为指针指向一个地址。C++语言提供了一种“动态内存分配”机制，在程序运行期间，根据实际需要，临时分配内存空间用于存储数据。使用 new 运算符实现动态内存分配，格式如下：

new 类型名[size];

C++语言还提供了 delete 运算符，用来释放动态分配的内存空间。格式如下：

delete 指针;

10、用 string 对象处理字符串

C++标准模板库中提供了 string 数据类型，专门用于处理字符串。string 是一个类，这个类型的变量称为“string 对象”。要在程序中使用 string 对象，必须在程序中包含头文件 string，即在程序的最前面，加上如下语句：

#include <string>

声明格式：

string 变量名;

考点二：C++语言的程序结构

C++程序以.cpp 作为文件扩展名，文件中包含若干类和若干函数。程序中必须有且有一个名为 main（不是 C++的关键字）的主函数，这是程序的总入口。主函数也称为主程序。程序从主函数的开始处执行，按照其控制结构，一直执行到结束。

程序的结束通常是遇到了以下两种情形之一：

（1）在主函数中遇到 return 语句。

（2）执行到主函数最后面的括号 } 。

主函数中可以调用程序中定义的其他函数，但其他函数不能调用主函数。主函数仅是系统为执行程序时所调用的。

C++程序中，仍沿用 C 语言的注释风格，如下：

（1）从/* 开始，到 */结束，这之间的所有内容都视作注释。

（2）从// 直到行尾，都是注释。

第二章　面向对象的基本概念

考点一：结构化程序设计

在结构化程序设计中，采用自顶向下、逐步求精及模块化的思想，将复杂的大问题层层分解为许多简单的小问题。

考点二：面向对象程序设计的概念和特点

所谓函数，是模块的基本单位，是对处理问题的一种抽象。

结构化程序设计使用的是功能抽象，面向对象程序设计不仅能进行功能抽象，而且能进行数据抽象。“对象”实际上是功能抽象和数据抽象的统一。

面向对象的程序设计有“抽象”、“封装”、“继承”、“多态”4 个基本特点。

（1）抽象：是一种从一般的观点看待事物的方法，即集中于事物的本质特征，而不是具体细节或具体实现。

（2）封装：把对象的属性和操作结合成一个独立的系统单位，并尽可能隐蔽对象的内部细节。

（3）继承：一个类可以获得另一个类的特性的机制，继承支持层次概念。

（4）多态：不同的对象可以调用相同名称的函数，但可导致完全不同的行为的现象。

考点三：类

1、类的定义

在 C++中，类是用户自定义的数据类型。使用时，要先定义这个类型，然后像声明基本数据类型的变量一样声明类类型的变量，即对象，之后才可以使用。类定义的一般格式如下：

```
class 类名
{
        访问范围说明符:
            成员变量 1
            成员变量 2
            ……
            成员函数声明 1
            成员函数声明 2
            ……
        访问范围说明符:
            更多成员变量
            更多成员函数声明
            ……
};
```

类是具有唯一标识符的实体，即类名不能重复。类定义以“;”结束，大括号中的部分称为类体。

“访问范围说明符”一共有 3 中，分别是 public（公有）、private（私有）和 protected（保护），在类定义中可以以任意的次序出现任意多次。

类的成员按功能划分，包括成员变量和成员函数；按访问权限划分，包括公有成员、私有成员和保护成员。

2、定义成员函数

类中声明的成员函数用来对数据成员进行操作，还必须在程序中实现这些成员函数。成员函数既可以在类体内定义，也可以在类体外定义。在类体外的定义格式如下：

```
返回值类型  类名::成员函数名（参数列表）
{
    成员函数的函数体
}
```

其中“类名::成员函数名”是作用域运算符，表明其后的成员函数时属于这个特定类的；“类名”是成员函数所属类的名字，“返回类型”是这个成员函数返回值的类型。

如果在声明类的同时，在类体内给出成员函数的定义，则默认为内联函数。也可以使用关键字 inline 将成员函数定义为内联函数。

考点四：使用类对象

1、定义对象，即类变量的基本方法如下：

```
方法一：  类名  对象名 1,对象名 2,……;
或        类名  对象名 1（参数）,对象名 2（参数）,……;
或        类名  对象名 = 类名（参数）;
```

创建对象后，C++会为它分配相应的空间，用来存储对象所有的成员变量，而类中定义的成员函数则被分配到存储空间中的一个公用区域，由该类的所有对象共享。

```
方法二：  类名 *对象指针名 = new 类名;
    或    类名 *对象指针名 = new 类名（）;
    或    类名 *对象指针名 = new 类名（参数）;
```

用 new 创建对象时返回的是一个对象指针，这个指针指向本类刚创建的这个对象。

2、访问对象的成员

对象和引用都使用运算符“.”访问对象的成员，指针则使用“->”运算符。通过对象访问成员变量的一般格式如下：

```
对象名.成员变量名    或    对象指针名->对象成员名
```

调用成员函数的一般格式如下：

```
    对象名.成员函数名(参数表)
```

考点五：类成员的可访问范围

1、public 的含义是公有的，使用它修饰的类的成员可以在程序的任何地方被访问；

2、private 的含义是私有的，使用它修饰的类的成员仅能在本类内被访问；

3、protected 的含义是保护的，它的作用介于 public 与 private 之间，使用它修饰的类的成员能在本类内及子类中被访问。

三种关键字出现的次数和先后次序都没有限制。成员的可访问范围由它之前离它最近的那个访问范围说明符决定。在默认情况下，类的所有成员都是私有的。

考点六：标识符的作用域与可见性

C++中标识符的作用域有函数原型作用域、局部作用域（块作用域）、类作用域和命名空间作用域。

1、函数原型作用域

在声明函数原型时形参的作用范围就是函数原型作用域。

2、局部作用域

程序中使用相匹配的一对大括号括起来的一段程序称为块。作用域局限在块内的称为局部作用域。

3、类作用域

类可以被看成一组有名字的成员的集合，类 X 的成员 m 具有类作用域，对 m 的访问方式有如下 3 种方式：

（1）如果在类 X 的成员函数中没有声明同名的局部作用域标识符，那么在该函数内可以直接访问成员 m。

（2）在类外，可以通过表达式 x.m 或者 X：：m 来访问，其中 x 是类 X 的对象。

（3）在类外，可以通过 ptr->m 访问，其中 ptr 是指向类 X 的一个对象的指针。

4、命名空间作用域

一个大型的程序通常由不同模块构成，不同模块中的类和函数之间有可能发生重名，容易引发错误。命名空间是为了消除同名引起的歧义。

定义命名空间的一般形式如下：

```
namespace  命名空间名{
命名空间内的各种声明（函数声明、类声明、……）
}
```

一个命名空间确定了一个命名空间作用域，在其内部可以直接引用当前命名空间中声明的标识符，如果需要引用其他命名空间的标识符，需要使用下面的方式：

命名空间名：：标识符名

对于在不同的作用域声明的标识符，可见性的一般原则如下：

（1）标识符要声明在前，引用在后。

（2）在同一个作用域中，不能声明同名的标识符。在没有互相包含关系的不同作用域中声明的同名标识符，互不影响。

（3）如果存在两个或多个具有包含关系的作用域，外层声明了一个标识符，而内层没有再次声明同名标识符，那么外层标识符在内层仍然可见。