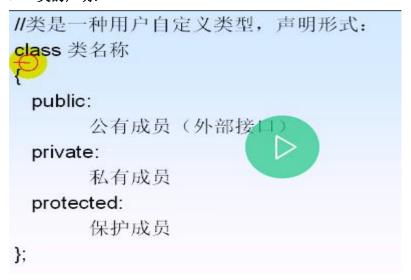
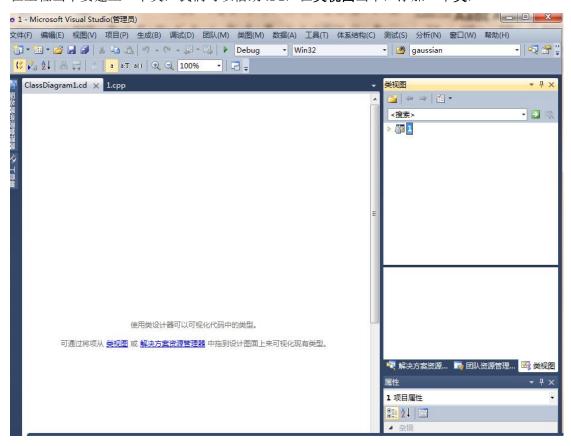
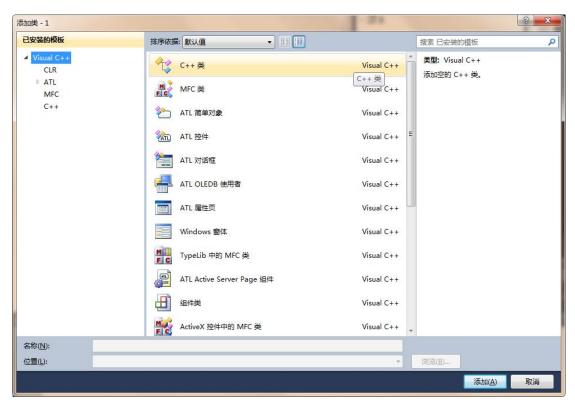
类和对象

【类声明】 【公有、私有、保护成员】 【数据抽象和封装】 ▶ 类的声明:

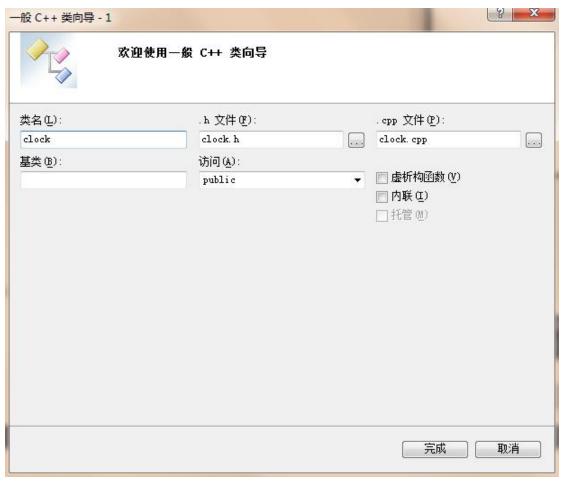


最后这个分号是不能省略的,接下来我们编写一个类: 在工程当中要建立一个类,我们可以借助 IDE,在类视图当中,添加一个类,





这里我们添加一个时钟类(clock):



它会产生两个文件, 一个是头文件(.h)一个是实现文件(.cpp)。并且 IDE 还会产生一些额

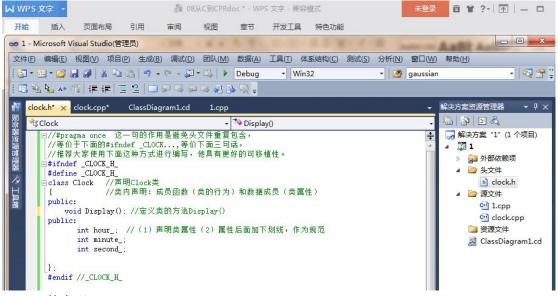
外的代码。

```
(全局范围)
                                     #pragma once
                     ∃class clock
                                     public:
                                                                          clock (void) ;
                                                                                   clock (void);
                              1:
clock.h* clock.cpp* × ClassDiagram1.cd
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        解决方案资源管理器
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       - 7 ×
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             (全局范围)
                     ⊟#include "clock.h"
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             从解决方案"1"(1个项目)
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             ⊿ 3 1
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                1 外部依赖项
                  -clock::clock(void)
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                ▲ 🧁 头文件
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               i clock.h
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          / 源文件

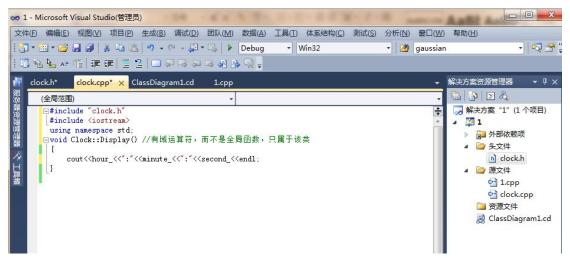
    1.cpp
    1.cpp

                  ⊟clock::~clock(void)
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               cock.cpp
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                一 咨源文件
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                ClassDiagram1.cd
```

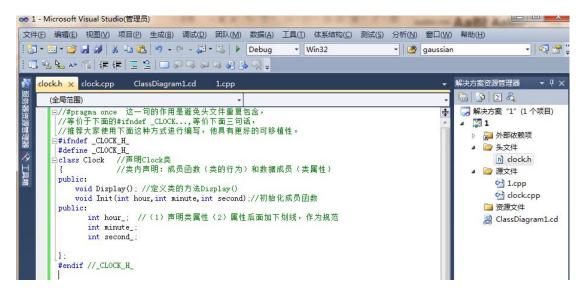
对于初学者来说,这些代码我们都手工进行别写(把自动生成的都删掉),也就是说我们使用**类向导**生成类,仅仅是为了产生这两个文件。当然大家也可以在这两个目录底下创建这两个文件,然后添加到工程当中。编写这个类:

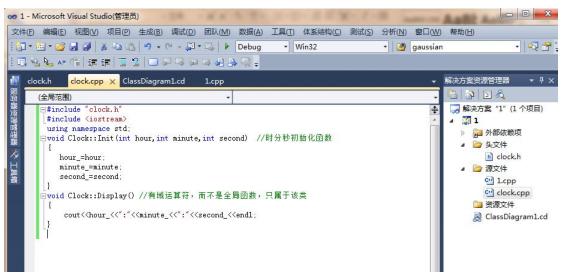


Display 的实现:



接下来我们编写程序的测试程序,并添加初始化函数 Init():







接下来我们运行这个程序:

```
_ _ _ <u>_</u> _ _ _ <u>X</u>
                                . . . . . . . .
∞ 1 - Microsoft Visual Studio(管理员)
文件(P) 编辑(E) 视图(V) 项目(P) 生成(B) 调试(D) 团队(M) 数据(A) 工具(T) 体系结构(C) 测试(S) 分析(N) 窗口(W) 帮助(H)
🛅 🕶 🗃 🚰 🛃 🥻 🛅 🧸 🖹 🖒 🗢 🖂 🖟 🖟 Debug
                                               → Win32
                                                                gaussian
                                                                                       - | 🖏 🕾
    14 No. Ant 1 [ # # ] 그 일 | 🗆 🔊 🗣 📦 🖓 💂
  clock.h clock.cpp ClassDiagram1.cd 1.cpp X
                                                                           解决方案资源管理器 ▼ P ×
服务器资源管理器
                                                                           🖺 🗿 🖪 🗞
     (全局范围)
                                     → wmain(void)
      #include
                                                                            🔜 解决方案 "1" (1 个项目)
     int main(void)
                                                                            4 📑 1
         ▷ 🔓 外部依赖项
                                                                              ▲ 🧁 头文件
X III
                                                                                  n clock.h
         return 0:
                                                                              ▲ 🧁 源文件

    1.cpp

                                                                                  cod clock.cpp
                                                                               一 资源文件
                                                                               ClassDiagram1.cd
                                                            - - X
   C:\Windows\system32\cmd.exe
   10:20:15
请按任意键继续. . .
```

假设我们过了 1s:

```
clock.h
            clock.cpp ClassDiagram1.cd
                                       1.cpp X
服务器资源管理器
                                              main(void)
     (全局范围)
       #include "clock.h"
      ∃int main(void)
       {
                     //定义一个对象:前面我们已经创建了类,我们可以把类看成一种类型,类是抽象的,
          Clock c;
          c. Init (10, 20, 15)://初始化
          c.Display()://将它打印出来
工具箱
          //假设现在过了1s
          c.second_+=1;
          //然后再把它显示出来
          c.Display();
          return 0;
```



由于我们的 second_是声明在 public,所以说类外部是可以访问的,接下来我们看一下类的三种访问权限。

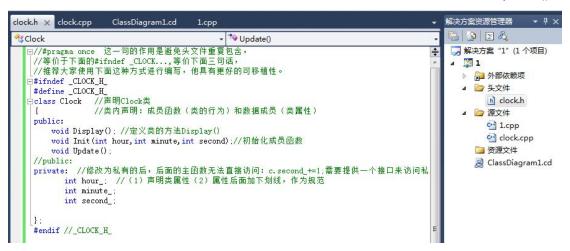
▶ 公有、私有、保护成员

- ◆ 在关键词 public 后面声明,它们是**类和外部的接口**,任何外部函数都可以访问公有类型数据和函数。
- ◆ 在关键词 private 后面声明,只允许本类中的函数访问,而类外部的任何函数都不能访问。
- ❖ 在关键词 protected 后面声明,与 private 类似,其主要的差别表现在继承和派生时对派生类的影响不同。

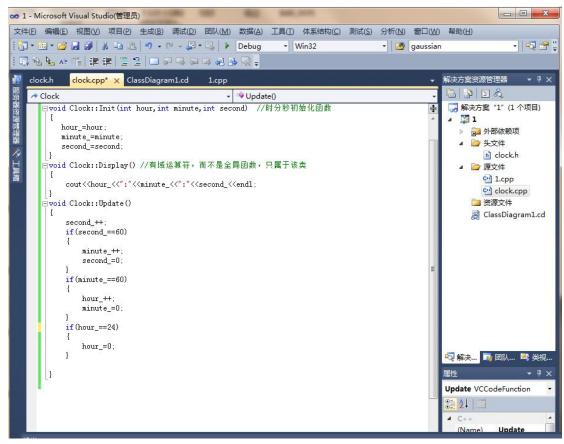
如果将前面的程序中的 public 修改为 private,如下:就会出错,无法访问 private 成员。

```
(全局范围)
      □//#pragma once 这一句的作用是避免头文件重复包含,
        //等价于下面的#ifndef _CLOCK...,等价下面三句话,
        //推荐大家使用下面这种方式进行编写,他具有更好的可移植性。
       #ifndef _CLOCK_H_
        #define _CLOCK_H_
プロ具備
       class Clock //声明Clock类
                     //类内声明:成员函数(类的行为)和数据成员(类属性)
        public:
           void Display(); //定义类的方法Display()
           void Init(int hour, int minute, int second)://初始化成员函数
        //public:
              int hour_; //(1) 声明类属性(2) 属性后面加下划线,作为规范
              int minute_;
              int second_;
        #endif //_CLOCK_H_
   显示輸出来源(S): 生成
                                                       · 👂 🖨 🛝 🔻 🖃
   17生成居如时间方 2016/9/30 重期五 14:27:58。
   1>InitializeBuildStatus:
   1> 正在创建"Debug\1.unsuccessfulbuild",因为已指定"AlwaysCreate"。
   1>ClCompile:
   1> 1. cpp
   1>e:\cppnote\08\1\1\clock: f(lock::second_": 无法访问 private 成员(在 "Clock" 类中声明)
1> e:\cppnote\08\1\1\clock.h(15): 参见 "Clock::second_" 的声明
1> e:\cppnote\08\1\1\clock.h(7): 参见 "Clock" 的声明
   1> 正在生成代码...
```

但是可以通过定义接口(函数)来修改这些成员变量:添加接口函数 Update():



接口函数的具体实现:

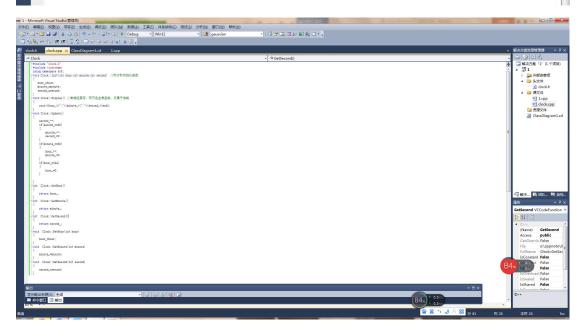


运行结果:



如果想直接访问私有成员,肯定是不可以的,这个时候就需要提供一些借口出来,直接访问。

```
□//#pragma once 这一句的作用是避免头文件重复包含,
|//等价于下面的#ifndef _CLOCK...,等价下面三句话,
//推荐大家使用下面这种方式进行编写,他具有更好的可移植性。
□#ifndef _CLOCK_H_
#define _CLOCK_H_
□class Clock //声明Clock类
              //类内声明:成员函数(类的行为)和数据成员(类属性)
 {
 public:
    void Display(); //定义类的方法Display()
    void Init(int hour, int minute, int second)://初始化成员函数
    void Update();
    int GetHour();
    int GetMinute();
    int GetSecond();
    void SetHour(int hour);
    void SetMinute(int minute);
    void SetSecond(int second);
 //public:
 private: //修改为私有的后,后面的主函数无法直接访问: c.second_+=1:需要提供一个接口来访问私 int hour_; //(1)声明类属性(2)属性后面加下划线,作为规范
       int minute_;
       int second_;
 };
 #endif //_CLOCK_H_
```



```
(全局范围)
 #include
 int main(void)
              //定义一个对象: 前面我们已经创建了类,我们可以把类看成一种类型,类是抽象的,对象是具体的
    c. Init (10, 20, 15)://初始化
    c. Display()://将它打印出来
    //假设现在过了1s
    c. Update()
    //然后再把它显示出来
    c.Display();
    //如果我们想直接更改私有成员的话,肯定是不可以的
    c. SetHour (15);
    c. Display();
    return 0;
                                                              _ D X
C:\Windows\system32\cmd.exe
 10:20:16
 15:20:16
 请按任意键继续. . . ■
```

如果我们不考虑继承的话,实际上我们只需要考虑 public 和 private 类型,公有的表示外界的函数可以访问,私有表示只有类内部的函数才能访问,也就是说只有类内部的代码才能改变类内部的状态,换句话说就是外因必须通过内因起作用,这就比较哲学了。对于 protected 类型来说,仅仅只是和 private 在继承和派生的时候不一样。它表示保护成员在派生类的成员函数是能访问的。关于这点,我们在讲到继承的时候再详细的说明。

数据抽象和封装

数据抽象是一种依赖于接口和实现分离的编程(和设计)技术,类设计者(也叫设计程序员)必须关心类是如何实现的,但使用该类的程序员(服务程序员)不必了解这些细节,使用者只要抽象的考虑该类型是做什么的,而不必具体的考虑该类如何工作。

假设我们实现一个栈类:

这就是数据抽象的编程思想。

▶ 封装是一项将低层次的元素组合起来形成新的、高层次的实体技术,函数是 封装的一种形式:函数所执行的细节行为被封装在函数这个更大的实体中, 被封装的元素隐藏了它们的实现细节——可以调用函数,但是不能直接访问 函数所执行的语句。同样的,类是一个封装的实体:它代表若干成员的聚集, 设计良好的类隐藏了类实现的细节。 并非给客户越多的东西就是越好的服务,相反,仅仅给他需要关注的东西才是最好的服务。(哲学观点),这样可以防止粗心的程序员破坏类内部的数据结构。只能通过类所暴露出来的接口来访问,类内部的数据成员。这就是数据抽象与封装的意义。