

以钟表为例。定义钟表数据和行为。

头文件 `clock.h`

```
1  #pragma once
2  class tagClock {
3  private:
4      //数据
5      int nHour;
6      int nMinute;
7      int nSecond;
8
9  public:
10     //行为, 函数声明
11     int __stdcall SetTime(int nH, int nM, int nS);
12 };
```

实现头文件中定义的成员函数, 源文件 `clock.cpp`

```
1  #include "stdafx.h"
2  #include "clock.h"
3
4  //函数定义, 需要加上结构体名作用域
5  int tagClock::SetTime(int nH, int nM, int nS) {
6      //this指针表示当前调用该函数的对象的地址
7      this->nHour = nH;
8      nMinute = nM;
9      nSecond = nS;
10
11     return 0;
12 }
```

实现所有函数后, 主函数中调用 `main.cpp`

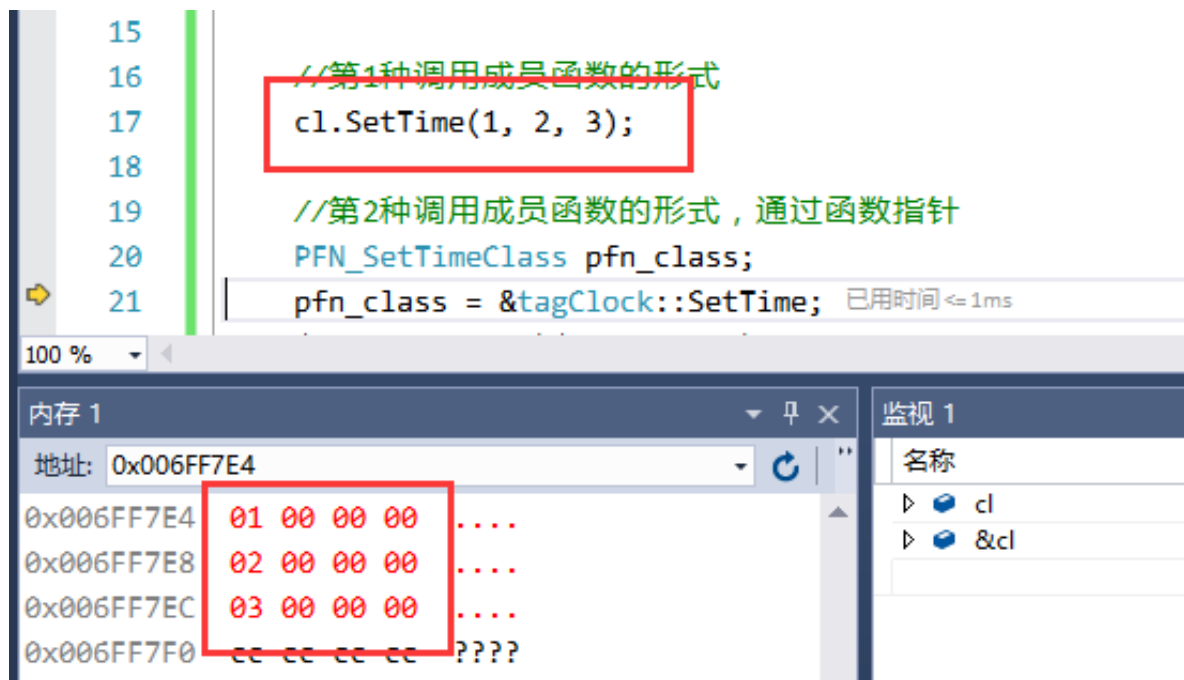
```
1  #include "stdafx.h"
```

```

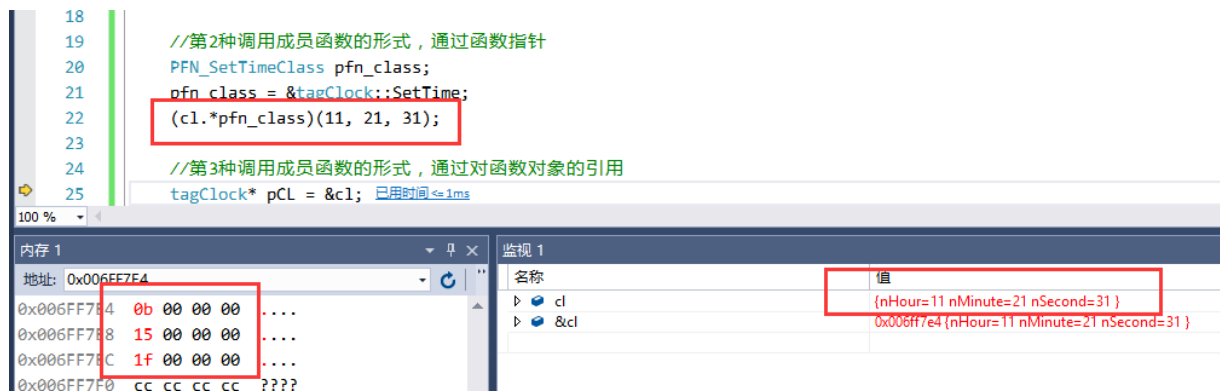
2  #include "clock.h"
3
4  //定义类中的函数指针
5  typedef int(__stdcall tagClock::*PFN_SetTimeClass)(int nH, int
    nM, int nS);
6  //参数传递: 寄存器 栈
7  // thiscall --> 用于类的成员函数的调用约定, 编译器内部使用
8  // fastcall
9  // cdecl
10 // stdcall
11
12 int main()
13 {
14     tagClock cl;
15
16     //cl.nHour = 1; //不可访问
17
18     //第1种调用成员函数的形式
19     cl.SetTime(1, 2, 3);
20
21     //第2种调用成员函数的形式, 通过函数指针
22     PFN_SetTimeClass pfn_class;
23     pfn_class = &tagClock::SetTime;
24     (cl.*pfn_class)(11, 21, 31);
25
26     //第3种调用成员函数的形式, 通过对类对象的引用
27     tagClock* pCL = &cl;
28     //PFN_SetTimeClass pfn_class;
29     //pfn_class = &tagClock::SetTime;
30     (pCL->*pfn_class)(10, 20, 30);
31
32     return 0;
33 }

```

- 第1种调用成员函数的形式



- 第2种调用成员函数的形式，通过函数指针



- 第3种调用成员函数的形式，通过对类对象的引用

