C++面向对象程序设计

第三讲：封装

计算机科学学院@中南民族大学

抽象：获取实体的属性和行为

对具体实体（对象）进行概括，抽取一类对象的公共属性和行为。

- 注意本质，围绕重点，抓住共性 ——》**属性和行为**
- 属性：数据抽象，某类对象的属性或状态（**对象相互区别的依据**）
  - 标识属性
  - 确定属性的值域
- 行为：方法抽象，某类对象的行为特征或具有的功能
  - 标识行为
  - 行为的处理对象（**有哪些？来源是什么？**）
  - 行为的处理结果（**有什么？谁接受/收处理结果？**）

抽象的实例：时间

- 时间的构成要素（应用范围）
  - 时、分、秒、百分秒、毫秒、微秒、纳秒
- 时间的使用
  - 啥时候了？几点了？
    - 6点（上午还是下午？）
  - 30分钟后出发、2小时后去哪儿等等
    - 到底是啥时候呢?
  - ……

抽象的实例：时间属性

- 属性
  - 小时
    - 值域：0-12 / 0-23？
  - 分钟
    - 值域：0-59
  - 秒
    - 值域：0-59

抽象的实例：时间功能

- 设置当前时间
  - 设定时间对象的值(时间），谁如何给定?
- 显示
  - 将时间对象的值打印在屏幕上
- 时/分/秒
  - 获取时间对象的分量值（时、分、秒）
- 计算
  - 加上时/分/秒后的时间值，是改变对象本身的值，还是生成新的对象
- ……
- 还能想到什么?

封装时间对象——类

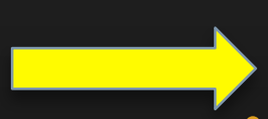
整合属性与行为并进行程序语言表示—〉对象泛化为类
对象泛称—〉类名；属性—〉成员变量；行为—〉成员函数

- 时间
  - 属性
    - 时（0-23）
    - 分（0-59）
    - 秒（0-59）
  - 行为
    - 设定时间（时、分、秒）
    - 显示时间（）
    - 增加分量（时/分/秒）

- Time
  - 成员变量
    - char hour; 0-23
    - char minute; 0-59
    - char second; 0-59
  - 成员函数
    - void SetTime(char, char ,char)
    - void ShowTime（）
    - void Add(char, char）

类定义（封装）（续）

- Time
  - 成员变量
    - char hour; 0-23
    - char minute; 0-59
    - char second; 0-59
  - 成员函数
    - void SetTime(char, char ,char)
    - void ShowTime （）
    - void Add(char, char)



- CTime
  - 成员变量
    - int hour; 0-23
    - int minute; 0-59
    - int second; 0-59
  - 成员函数
    - void SetTime(int, int, int)
    - void ShowTime （）
    - void add(int, char)
    - void addHour(int)
    - void addSec(int)
    - void addMin(int)

封装：定义时间类(头文件)

文件名：mytime

```
#ifndef TIME_H_
#define TIME_H_
namespace nsname {
class CTime {
public:
    int hour;
    int minute;
    int second;
    void SetTime(int h, int m, int s);
    void ShowTime();
    void addHour(int);
    void addMin(int);
    void addSec(int);
};
} //namespace nsname
#endif /* TIME_H_ */
```

类的实现——时间类

文件名： mytime.cpp

```
#include "mytime”
#include <iostream>
#include <iomanip>
using namespace std;
namespace nsname {

void CTime::SetTime(int h, int m, int s)
{
      hour = h;
      minute = m;
      second = s;
}
void CTime::ShowTime()
{
      cout << hour << “:” << minute << second;
}
```

```
void CTime::addHour(int h)
{
      hour += h;
}
void CTime::addMin(int m)
{
      minute += m;
}
void CTime::addSec(int s)
{
      second += s;
}

}
```

类的使用——时间对象

```
#include "mytime"

using namespace nsname;
int main() {
        CTime c1;
        c1.SetTime(14,20,30);
        c1.ShowTime();
        c1.SetTime(8,2,30);
        c1.ShowTime();
        c1.SetTime(8,2,3);
        c1.ShowTime();
    c1.addHour(5);
    c1.ShowTime();
        return 0;
}
```

类定义（封装）
类的属性和行为的访问限制

- 限制数据成员和函数成员的访问权限
  - 公有 public
    - 完全公开的属性和行为
  - 私有 private
    - 个体专属的属性和行为
  - 保护 protected
    - 家族私有的属性和行为

类的定义

- 类是一种用户自定义的数据类型
- 声明形式
  - Class 类名称
  - {
    - public:
      - 公有成员
    - private:
      - 私有成员
    - protected:
      - 保护成员
  - } ;

注意给定访问限制的方式：
从限制方式开始的后续成员，
直到变更访问限制

默认的成员访问限制是“私有成员”

类的定义（续）

- 成员访问限制符号对其后续成员均有效，直到遇到下一个成员访问限制符号；
- 紧跟在类名称的后面的成员，没有访问限制符号注明的话，均为私有成员；
- 从程序的角度而言
  - 随时随地可以访问的是类的公有成员；
  - 类的私有成员则只能在类的成员函数中被访问；
  - 类的保护成员可以被类及其派生类的成员函数中被访问。

成员私有化

- 限制非类成员函数对成员变量的访问
- 限制对象的属性和行为被其他对象访问和使用

```
class CTime
{
private:
    int hour, minute, second;
public:
    int getHour(); //取值函数
    void setHour(int h); //设值函数
    int getMinute(); //取值函数
    void setMinute(int m); //设值函数
    int getSecond(); //取值函数
    void setSecond(int s); //设值函数
    void SetTime(int h, int m, int s);
    void ShowTime();
    void addHour(int);
    void addMin(int);
    void addSec(int);
};
```

```
int CTime::getHour()
{
    return hour;
}

void CTime::SetHour(int h)
{
    if((h >= 0) && (h < 24)) hour = h;
}

void CTime::addSec(int s)
{
    second += s;
    addMin(second / 60);
    second %= 60;
}
void CTime::addMin(int m)
{
    minute += m;
    addHour(minute / 60);
    minute %= 60;
}
void CTime::addHour(int h)
{
    hour += h;
    hour %= 24;
}
```

示例

```
int main()
{
    CTime c1;
    c1.SetTime(24,20,32); //              设值函数：if((h >= 0) && (h < 24)) hour = h;
    c1.ShowTime();
    cout << endl;

    c1.SetTime(23,20,32); //              设值函数：if((h >= 0) && (h < 24)) hour = h;
    c1.addSec(180);  //                   180秒后的时间值会是多少?
    c1.ShowTime();
    cout << endl;

    c1.addMin(180);  //                   180分后的时间值会是多少?
    c1.ShowTime();
    cout << endl;
    return 0;
}
```

课堂作业：

- 定义一个日期类
  - 能想到什么，就做什么！！

构造对象

- 思考问题
  - 变量在使用前必须有值
  - 定义类时，并不能其成员变量赋值
  - 如何才能在定义对象时，就使对象的属性有效呢?
- 构造函数
  - 作用：确定对象的初始形态
  - 如何定义构造函数
  - 何时如何调用构造函数
    - 定义对象时自动调用的类的成员函数
      - 隐式调用
    - 动态构造对象时调用的成员函数
      - 显式调用

构造函数的作用

- 确定对象的初始形态
  - 属性值有效
    - 对成员变量进行赋值，使对象的属性有效/有意义
  - 分配有效的空间
    - 申请内存空间
  - 确定对象的初始状态
    - 空闲、繁忙
  - ……

声明和定义构造函数

- 定义时间类的构造函数
- class CTime {
  - ……
  - CTime(); //类的构造函数（默认构造函数，没有任何形参！）
    - 函数名与类名相同，但没有返回值的类型
    - 事实上，不声明和定义构造函数（即构造函数缺失时），编译器也会自动生成一个函数体为空的默认构造函数
- }

- CTime::CTime()
- {
  - hour = 0;
  - minute = 0;
  - second = 0;
- }

使用构造函数

- 定义对象时，隐式调用默认构造函数
  - CTime c1;

- 动态构造对象时，显式调用构造函数
  - CTime * pc1 = new CTime();

问题

- 如何让程序员在使用时间对象时，能灵活地赋以合理的时间值
  - 想一下
    - int x = 5;
    - int x = 20;
    - double x = 0.2;
    - double x = 10;
    - int x[] = {31, 29, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
    - struct person zhangsan={“zhangsan”, ‘F’, “1995-10-1”};//假设在结构体定义

程序员在定义变量时可以按需赋值

丰富构造函数

- 让程序员自主赋值给对象
  - 存在问题：该调用哪个函数来构造对象，如何确定函数的参数
  - 解决方法：共用函数名称（标识符）
  - 程序员：使用不同的构造函数来构造对象
- 函数重载
  - 函数名相同
  - 形参列表不同
    - 数量不同
    - 顺序不同
    - 类型不同
    - 数量、顺序、类型都不同

重载构造函数

```
CTime(int h, int m, int s)
{
    hour = h; minute = m; second = s;
}

CTime(const string &atime)
{
    //请完成拆分字符串，分离出时分秒并赋给hour, minute和second
}

#include <ctime> //包含C的时间类型函数库time.h
CTime(time_t atime)    //time_t 时间类型
{
    //请实现该函数体
}
```

示例

- CTime c1(“10:23:32”);
- CTime c2(10, 23, 32);
- CTime c3(“11:45:00”);

- CTime *p1 = new CTime(“10:23:32”);
- CTime *p2 = new CTime(10, 23, 32);
- CTime *p3 = new CTime(“11:45:00”);