Objects and Classes

【目标】

- 理解抽象数据类型，类的概念
- 定义和实现类
  - 访问控制机制
  - 数据成员
  - 类方法（类函数成员）
- 创建和使用类对象
- 特别的成员函数
  - 构造函数【对象创建时自动调用，可重载以实现用不同参数进行创建】
  - 析构函数【对象撤销时自动调用】
  - const 成员函数
- this 指针
  - 在类内部标记类对象【实体】的指针
- 其中，控制机制
  - 访问控制【实现private数据成员的封装，只能通过public类方法调用】
  - const成员函数【专用于const对象调用的函数】

```
// stock20.h -- augmented version
#ifndef STOCK20_H_
#define STOCK20_H_
#include <string>

class Stock
{
private:
    std::string company;
    int shares;
    double share_val;
    double total_val;
```

```
    void set_tot() { total_val = shares * share_val;
}
public:
    Stock();        // default constructor
    Stock(const std::string & co, long n = 0, double
pr = 0.0);
    ~Stock();       // do-nothing destructor
    void buy(long num, double price);
    void sell(long num, double price);
    void update(double price);
    void show()const;
    const Stock & topval(const Stock & s) const;
};

#endif

// stock20.cpp -- augmented version
#include <iostream>
#include "07-stock20.h"
using namespace std;
// constructors
Stock::Stock()        // default constructor
{
    shares = 0;
    share_val = 0.0;
    total_val = 0.0;
}

Stock::Stock(const std::string & co, long n, double
pr)
{
    company = co;
    if (n < 0)
    {
        std::cout << "Number of shares can't be
negative; "
                  << company << " shares set to
0.\n";
        shares = 0;
    }
    else
```

```
        shares = n;
    share_val = pr;
    set_tot();
}

// class destructor
Stock::~Stock()        // quiet class destructor
{
}

// other methods
void Stock::buy(long num, double price)
{
     if (num < 0)
    {
        std::cout << "Number of shares purchased
can't be negative. "
             << "Transaction is aborted.\n";
    }
    else
    {
        shares += num;
        share_val = price;
        set_tot();
    }
}

void Stock::sell(long num, double price)
{
    using std::cout;
    if (num < 0)
    {
        cout << "Number of shares sold can't be
negative. "
             << "Transaction is aborted.\n";
    }
    else if (num > shares)
    {
        cout << "You can't sell more than you have! "
             << "Transaction is aborted.\n";
    }
```

```
    else
    {
        shares -= num;
        share_val = price;
        set_tot();
    }
}

void Stock::update(double price)
{
    share_val = price;
    set_tot();
}

void Stock::show() const
{
    using std::cout;
    using std::ios_base;
    // set format to #.###
    ios_base::fmtflags orig =
        cout.setf(ios_base::fixed,
ios_base::floatfield);
    std::streamsize prec = cout.precision(3);

    cout << "Company: " << company
        << "  Shares: " << shares << '\n';
    cout << "  Share Price: $" << share_val;
    // set format to #.##
    cout.precision(2);
    cout << "  Total Worth: $" << total_val << '\n';

    // restore original format
    cout.setf(orig, ios_base::floatfield);
    cout.precision(prec);
}

const Stock & Stock::topval(const Stock & s) const
{
    if (s.total_val > total_val)
        return s;
    else
```

```
128         return *this;
129 }
```