Занятие 10

Определение и реализация класса. Элементы (члены) класса – данные (переменные) и методы (функции). Открытый и закрытый доступ к элементам класса. Создание и использование объектов класса.

Пример 1. В примере реализуется работа с пакетами акций компаний. Акция – ценная бумага, которая определяет долю владения компанией и дает различные права ее владельцу (акционеру), например, право на получение части прибыли от деятельности компании в виде дивидентов. Для работы с пакетами акций удобно ввести новый тип данных, класс `Stock`, который и будет здесь изучаться.

Класс – это тип данных, который вводится для представления каких-либо понятий (предметов). Объект – это переменная класса, то есть определенного типа. Возможность вводить абстрактные типы данных – одно из самых полезных расширений C++ по сравнению с C. Конечно, новые типы данных можно вводить и в C, но C++ более приспособлен для этого. Определение класса включает в себе перечень элементов-данных, которые описывают состояние объектов, и элементов-функций (называемых методами), которые описывают поведение объектов. Элементы бывают открытого (public) типа и закрытого (private). К элементам открытого типа доступ свободный, как и к элементам структур. Открытые элементы образуют так называемый интерфейс, так как позволяют управлять объектами и организовать их взаимодействие с другими объектами. К элементам закрытого типа можно получить доступ только находясь уже внутри открытых методов, то есть только через интерфейс.

Определение класса `Stock` дается в заголовочном файле `stock.h`. Во втором файле `stock.cpp` дается определение каждого метода, то есть что конкретно происходит при его вызове. В третьем файле `run_stock.cpp` показано как можно работать с объектами нового класса.

Файл `stock.h`:

```
class Stock { // даётся определение класса
 private:     // элементы (данные и функции) с закрытым доступом
  std::string company; // название компании
  long shares;          // количество акций
  double share_val;    // цена одной акции
  double total_val;    // полная цена пакета акций
  void set_tot() { total_val = shares * share_val; } // метод, расчет
   цены пакета
 public:  // элементы (данные и функции) с открытым доступом
  void acquire(const std::string & co, long n, double pr); // начальное
   приобретение
  void buy(long num, double price);  // метод реализует покупку акций
  void sell(long num, double price); // продажа акций
  void update(double price);         // изменяет цену
  void show() const;                 // выводит информацию на экран
};
```

При описании перед именем каждой функции, которая является элементом класса `Stock`, ставится `Stock::`, что задает её принадлежность этому классу.

Файл `stock.cpp`:

```cpp
#include <iostream>
#include "stock.h"

/*** Методы ***************************************************/
void Stock::acquire(const std::string & co, long n, double pr){
  company = co; // внутри методов есть доступ ко всем (private
                // и public) элементам объекта, к которому
                // применяется метод.
                // Здесь company - элемент объекта, к которому
                // применяется метод acquire()
  if (n < 0){// простая проверка
    std::cout << "Number of shares cannot be negative; "
        << company << " shares set to 0.\n";
    shares = 0;
  }
  else
    shares = n;

  share_val = pr;
  set_tot();   // есть прямой доступ к этому методу
}
/*************************************************************/
void Stock::buy(long num, double price) {
  // buy shares and update current price per share
  if (num < 0){ // проверка
    std::cout << "Number of shares purchased cannot be negative. "
        << "Transaction is aborted.\n";
  } else {
    shares += num;
    share_val = price;
    set_tot();
  }
}
/*************************************************************/
void Stock::sell (long num, double price) {
  // sell shares and update current price per share

  // NEEDS TO BE WRITTEN

}
/*************************************************************/
void Stock::update(double price){ // change the price
  share_val = price;
  set_tot();
}

/*************************************************************/
void Stock::show() const {
  /*
   * Выводится информация о пакете акций.
   * Применяется форматирование вывода для удобства чтения.
   *
   * const после аргументов метода означает, что метод не изменяет
   * состояние объекта, к которому он применяется
   */

```

```
57   using std::cout;
58   using std::ios_base;
59
60   // Сохранение и изменение исходного формата вывода
61   ios_base::fmtflags orig = cout.setf(ios_base::fixed, ios_base::
      floatfield);
62   std::streamsize prec = cout.precision(2);
63
64   cout << "Company: " << company  << ", " << "Shares: " << shares << ",
      ";
65   cout << "Share Price: $" << share_val << ", ";
66   cout << "Total Value: $" << total_val << '\n' ;
67
68   // Восстановление исходного формата
69   cout.setf(orig, ios_base::floatfield);
70   cout.precision(prec);
71 }
```

Файл run_stock.cpp:

```
1  #include <iostream>
2  #include <string>
3
4  #include "stock.h" // заголовочный файл с определениями
5
6  using namespace std;
7  /*************************************************************/
8  void run_stock(void){
9
10   Stock s1; // создается объект s1 класса Stock
11
12   /*** начальная покупка пакета акций ***/
13   s1.acquire("IBM", 10, 125.0); // создали пакет акций через вызов
14                                 // метода acquire
15
16   cout << "\nInitial acquiring\n";
17   s1.show();          // вывод информации на экран
18
19   // Теперь купим акций и проверим, чти изменилось
20   s1.buy(1, 150.0);  // купить акций, применив метод buy
21   cout << "\nAfter buying\n";
22   s1.show();          // проверяем, что изменилось
23
24   // Обратите внимание, что прямое изменение количества акций, например,
25   // s1.shares += 1;
26   // работать не будет, так как у нас нет прямого доступа
27   // к элементу shares.
28
29   // Цена акций изменилась
30   s1.update(100);    // изменяем цену
31   cout << "\nAfter price change\n";
32   s1.show();          // проверяем, что изменилось
33
34   // Теперь продадим часть акций и проверим, чти изменилось
35   //
36   //
37   //
```

```
38 }
39 /*************************************************************************/
40 int main() {
41     run_stock();
42     return 0;
43 }
```

Обратите внимание, что выполнение программы начинается всегда с `main`. В данном случая вызывается одна функция `run_stock`, которая и иллюстрирует работу с классом. Откомпилировав все файлы, можно создать исполняемый файл.
Результат выполнения:

```
Initial acquiring
Company: IBM, Shares: 10, Share Price: $125.00, Total Value: $1250.00

After buying
Company: IBM, Shares: 11, Share Price: $150.00, Total Value: $1650.00

After price change
Company: IBM, Shares: 11, Share Price: $100.00, Total Value: $1100.00
```

Домашнее задание

Задание 10.1. Напишите метод `Stock::sell()`, который осуществляет продажу акций (см. строчку 35 в файле `stock.cpp`), и продемонстрируйте его работу, вызывая его в функции `run_stock()`.

Задание 10.2. Представьте, что Вам надо разработать программу учета состояния автомобилей для службы такси. Вам надо хранить следующую информацию: марка машины, год выпуска, цена, пробег, запас бензина. Создайте класс `AUTO`, в котором хранятся эти данные в скрытом от пользователя виде. Разработайте методы, с помощью которых можно записывать эту информацию в объекты этого класса, изменять состояния показателей пробега и запаса бензина, а также выводить информацию на экран. Напишите пример использования этого класса и разработанных методов.