Занятие 11

Конструкторы и деструкторы класса. Функции – члены const. Указатель this. Создание массивов объектов и работа с ними.

Пример 1. В данном примере рассматриваются усовершенствования работы с классом `Stock` из предыдущего занятия. В частности, показывается создание двух конструкторов и деструктора. Мы заменим метод `acquire()`, который отвечал за начальную покупку акций в предыдущем примере, на конструктор.

Конструктор (constructor) – это метод (функция) класса, который вызывается при создании объекта и используется для инициализации элементов объекта. Если пользователь не создает собственного конструктора, то вызывается встроенный конструктор (или конструктор по умолчанию, default constructor), который ничего не делает. Можно разработать несколько конструкторов, перегрузив их как обычные функции. Конструктор имеет то же имя, что и название класса.

Деструктор (destructor) – метод (функция) класса, который вызывается при уничтожении объекта, то есть при выходе из блока, где он вводился. Деструкторы вызываются неявным образом. Если пользователь не создает собственного деструктора, то вызывается встроенный деструктор, который ничего не делает. Деструкторы обычно нужны для освобождения динамической памяти. В используемом примере деструктор просто выводит сообщение на экран, что позволяет понять, когда он вызывается. Имя деструктора – название класса со знаком тильда (~) впереди.

Существует встроенный указатель на объекты, которых имеет имя `this`. Если какой-либо метод (включая конструкторы) применяется к объекту, то внутри этого метода значение указателя `this` – это адрес этого объекта. Обратите внимание на его использование в конструкторе `Stock()` и в функции `topval()`, а также на адреса объектов `s1` в `s2` при их создании внутри `run_stock()`.

Обратите внимание, что после методов `show()` и `get_status()` класса `Stock` стоит ключевое слово `const`. Это означает, что данный метод не изменяет никакие элементы объекта, к которому он применяется.

Обратите внимание на использование слов `const` в методе `topval()`. Этот метод не изменяет ни аргумент, ни объект, вызывающий этот метод. Также он возвращает ссылку на объект, который не должен изменяться.

Файл `stock.h`:

```
class Stock { // дается определение класса
 private:  // элементы с закрытым доступом
  std::string company; // название компании
  long shares;          // количество акций
  double share_val;    // цена одной акции
  double total_val;    // стоимость всех акций
  void set_tot() { total_val = shares * share_val; }
 public:  // элементы с открытым доступом
  // void acquire(const std::string & co, long  n, double pr);

  Stock();  // конструктор без аргументов

  // конструктор с аргументами, есть значения по умолчанию
  Stock(const std::string &co, long n=0, double pr=0.0);
```

```
15
16    ~Stock(); // деструктор
17
18    void buy(long num, double price);  // метод реализует покупку акций
19    void sell(long num, double price); // продажа акций
20    void update(double price);         // изменяет цену
21    void show() const;                 // выводит информацию на экран
22
23    void get_status(std::string &s, int *shrs, double *sval,
24        double *tval) const; // получает скрытые данные
25
26    const Stock &topval(const Stock &s) const;
27    // сравнивает цену двух пакетов акций
28 };
29 void show_all_stocks(const Stock *stocks, int N); // не элемент класса
```

Файл stock.cpp:

```
1  #include <iostream>
2  #include "stock.h"
3
4  /*** конструкторы и деструкторы ************************************/
5  Stock::Stock(){ // конструктор без аргументов
6    company = "no name";
7    shares = 0;
8    share_val = 0.0;
9    total_val = 0.0;
10
11   // Можно также использовать имена с указателем this
12   this->total_val = 0.0; // эквивалентно total_val = 0.0;
13
14   // распечатаем адрес объекта, который инициализируется
15   std::cout << "Constructor is applied to object at " << this << std::
      endl;
16 }
17
18 Stock::Stock(const std::string &co, long n, double pr){
19   //конструктор с аргументами
20   company=co;
21   if (n < 0) {
22     std::cout << "Number of shares can't be negative; "
23         << company << " shares set to 0.\n";
24     shares = 0;
25   }
26   else
27     shares = n;
28
29   share_val = pr;
30   set_tot() ;
31 }
32
33 Stock::~Stock(){ // деструктор, выводящий сообщение
34   std::cout << "Bye, " << company << "!\n";
35 }
36
37 /*** Другие методы *************************************************/
38 void Stock::buy(long num, double price) {
```

```
39    // buy shares and update current price per share
40    if (num < 0){
41      std::cout << "Number of shares purchased can't be negative. "
42          <<"Transaction is aborted.\n";
43    } else {
44      shares += num;
45      share_val = price;
46      set_tot();
47    }
48  }
49  /**************************************************************************/
50  void Stock::sell(long num, double price) {
51    // cell shares and update current price per share
52
53    // NOT USED HERE
54  }
55  /**************************************************************************/
56  void Stock::update(double price){ // change the price
57    share_val = price;
58    set_tot();
59  }
60  /**************************************************************************/
61  void Stock::get_status(std::string &com, int *s, double *sv,
62             double *tv) const{
63    //get information about the stock
64   com=company;  *s=shares; *sv=share_val;  *tv=total_val;
65  }
66  /**************************************************************************/
67  const Stock & Stock::topval(const Stock & s) const {
68    /*
69     * Сравнивает цену пакета акций (объект класса Stock), к которому
70     * применяется данный метод, и пакета акций, который передается
71     * в аргументе.  Возвращает ссылку на пакет акций с большей ценой.
72     */
73
74    if (s.total_val > total_val)
75      return s;       // в аргументе цена больше, возвращаем ссылку s
76    else
77      return *this;   // this указывает на объект, к которому применяется
78                      // метод. Возвращаем ссылку на объект, пременяя *
79    // Можно и короче
80    // return (s.total_val > total_val) ? s : *this;
81  }
82  /**************************************************************************/
83  void Stock::show() const {
84    /*
85     * Выводится информация о пакете акций.
86     * Применяется форматирование вывода для удобства чтения.
87     *
88     * const после аргументов означает, что метод не изменяет
89     * состояние объекта, к которому он применяется
90     */
91
92    using std::cout;
93    using std::ios_base;
94
```

```
95    // Сохранение и изменение исходного формата вывода
96    ios_base::fmtflags orig = cout.setf(ios_base::fixed, ios_base::
       floatfield);
97    std::streamsize prec = cout.precision(2);
98
99    cout << "Company: " << company  << ", " << "Shares: " << shares << ",
       ";
100   cout << "Share Price: $" << share_val << ", ";
101   cout << "Total Value: $" << total_val << '\n' ;
102
103   // Восстановление исходного формата
104   cout.setf(orig, ios_base::floatfield);
105   cout.precision(prec);
106 }
107
108 /*************************************************************************/
109 void show_all_stocks(const Stock *stocks, int N){ // не элемент класса
110   /*
111    * Использует цикл для печати на экран информации о всех пакетах
112    * акций из массива. Информация форматируется для удобства чтения.
113    */
114
115   using namespace std;
116   cout <<"Stock holdings:\n";
117   string s;
118   int shares;
119   double share_val, total_val;
120
121   printf("    %15s %11s %12s %11s\n", "Company", "Shares ", "$/share", "
       Total $$");
122   for (int n = 0; n < N; n++){
123     stocks[n].get_status(s, &shares, &share_val, &total_val);
124     printf("%3d %15s %10d %13.2f %11.2f\n", n+1, s.c_str(), shares,
       share_val, total_val);
125   }
126 }
```

Файл run_stock.cpp:

```
1  #include<iostream>
2  #include<string>
3  #include<cstdio>
4
5  #include"stock.h"
6
7  const int STKS = 4; // количество объектов в массиве
8
9  /*************************************************************************/
10 void run_stock(void){ // пример работы с объектами
11
12   using namespace std;
13
14   /*** Создание объекта класса  ***/
15   Stock s1;          // неявный вызов конструктора без аргументов
16   Stock s2=Stock(); //   явный вызов конструктора без аргументов
17
18   // распечатаем адреса объектов s1 и s2
```

```
19    cout << "Object s1 is at " << &s1 << endl;
20    cout << "Object s2 is at " << &s2 << endl;
21
22    s1.show();         // вывести значения на экран, используя метод show
23    s2.show();
24
25    Stock s3("IBM", 4, 160.00);             // конструктор с аргументами
26    //Stock s3=Stock("IBM", 4, 160.00);     // так тоже можно
27    s3.show();
28    s3.buy(1, 150.0);  // купить акций, применив метод buy
29    s3.show();
30
31    Stock s4("Boeing"); // конструктор с аргументами, используются
       значения по умолчанию
32    s4.show();
33
34    /*** Создание массива объектов  ***/
35    Stock mystocks[STKS]={Stock("Apple",   10, 135.72),
36        Stock("Sysco",    10,  52.35),
37        Stock("Intel",   100,  36.48),
38        Stock("Alphabet", 5, 846.55)};
39    // Можно также
40    //stocks[0]=Stock("Apple",   10, 135.72);// 10 shares for 135.72 each
41    //stocks[1]=Stock("Sysco",   10,  52.35);
42    //stocks[2]=Stock("Intel",  100,  36.48);
43    //stocks[3]=Stock("Alphabet", 5, 846.55);
44
45    /*** Вывести на экран информация о массив объектов ***/
46    show_all_stocks(mystocks, STKS); // начальное состояние
47
48    /*** Операции с пакетами акций из массива ***/
49    mystocks[0].buy(11, 150.0);// применяем метод buy к 0-му элементу
50    mystocks[2].update(35.45); // применяем метод update ко 2-му элементу
51    show_all_stocks(mystocks, STKS); // новое состояние
52
53    /*
54     * Часть кода для нахождения пакета с наибольшей ценой.
55     * Вводим *top, который буден показывать на него, и в цикле
56     * сравниваем цены пакетов попарно.
57     */
58    const Stock *top = &mystocks[0]; //установка указателя на 0-ой элем.
59    for (int st = 1; st < STKS; st++)// сравниваем цены пакетов в цикле
60      top = &top->topval(mystocks[st]);
61
62    // Теперь top указывает на самый ценный пакет акций
63    std::cout << "\nMost valuable holding:\n";
64    top->show();
65  }
66  /*************************************************************************/
67  int main() {
68      run_stock();
69      return 0;
70  }
```

При выполнении программы обратите внимание, когда и в каком порядке вызываются деструкторы.

Результат выполнения:

```
Constructor is applied to object at 0x7ffdba256390
Constructor is applied to object at 0x7ffdba256350
Object s1 is at 0x7ffdba256390
Object s2 is at 0x7ffdba256350
Company: no name, Shares: 0, Share Price: $0.00, Total Value: $0.00
Company: no name, Shares: 0, Share Price: $0.00, Total Value: $0.00
Company: IBM, Shares: 4, Share Price: $160.00, Total Value: $640.00
Company: IBM, Shares: 5, Share Price: $150.00, Total Value: $750.00
Company: Boeing, Shares: 0, Share Price: $0.00, Total Value: $0.00
Stock holdings:
            Company     Shares       $/share    Total $$
  1           Apple         10        135.72     1357.20
  2           Sysco         10         52.35      523.50
  3           Intel        100         36.48     3648.00
  4        Alphabet          5        846.55     4232.75
Stock holdings:
            Company     Shares       $/share    Total $$
  1           Apple         21        150.00     3150.00
  2           Sysco         10         52.35      523.50
  3           Intel        100         35.45     3545.00
  4        Alphabet          5        846.55     4232.75

Most valuable holding:
Company: Alphabet, Shares: 5, Share Price: $846.55, Total Value: $4232.75
Bye, Alphabet!
Bye, Intel!
Bye, Sysco!
Bye, Apple!
Bye, Boeing!
Bye, IBM!
Bye, no name!
Bye, no name!
```

Домашнее задание

Задание 11.1. Пусть имеется следующее объявление класса, описывающего движение точки в пространстве:

```
1 class Point{
2  private:
3   std::string label; // имя (метка) точки
4   double x;          // координата
5   double y;          // координата
6  public:
7   Point (std::string s, double a = 0, double b = 0) ;
8   // создает объект с меткой s и значениями координат a, b
9
```

```
10    void showpoint() const;
11    // отображает метку и текущие значения координат
12
13    void movepoint(double x1, double y1);
14    // добавляет к координатам вызывающего объекта величины x1, y1
15
16    double dist(const Point & p1) const;
17    // Возвращает расстояние от вызывающего объекта до p1
18
19    Point add(std::string s, const Point & p1) const;
20    // Складывает x из p1 и x вызывающего объекта,
21    // складывает y из p1 и y вызывающего объекта,
22    // создает новый объект класса Point с полученными
23    // значениями x, y и меткой s и возвращает его
24
25    void reset (double a = 0, double b = 0) ;
26    // устанавливает x, y равными a, b
27
28    Point middle(std::string s, const Point & p1) const;
29    // Возвращает новую точку с именем s и координатами
30    // посередине между вызываемой точкой и p1.
31 };
```

Создайте определения методов (элементов-функций) и напишите программу, которая использует этот класс. Программа должна состоять из нескольких файлов: заголовочный файл, файл с определениями методов класса и файл, в котором непосредственно используется этот класс.