Last updated: 2019-03-31
vadimov@i.ua

Практическое занятие 8 (семестр2).

Совет: Попытайтесь выполнить все примеры Unit8. Поэкспериментируйте со спецификаторами доступа и изучите результаты.

Задача 8.1. Дана следующая, почти законченная программа, добавьте недостающие оператор-функции:

```
#include <iostream >
using namespace std;
class array {
    int nums[10];
public:
    array();
    void set(int n[10]);
    void show();
    array operator +(array obj2);
    array operator -(array obj2);
    array operator ==(array obj2);
};
array::array() {
    for (int i=0; i <10; i++) nums[i] = 0;
}
void array::set(int *n) {
    for (int i=0; i<10; i++) nums[i] = n[i];
}
void array::show() {
    for (int i=0; i<10; i++)
        cout << nums[i] << ' ';
    cout << "\n";
}
// Fill in operator functions.
int main() {
    array obj1, obj2, obj3;
    int i[10] = {1, 2, 3, 4, 5, 6, 7, 8 ,9 ,10};
    obj1.set(i);
    obj2.set(i);
    obj3 = obj1 + obj2;
    obj3.show();
    obj3 = obj1 - obj3;
    obj3. show();
    if (obj1 == obj2)
        cout << "obj1 equals obj2\n";
    else
        cout << "obj1 does not equal obj2\n";
    if (obj1 == obj3)
        cout << "obj1 equals obj3\n";
    else
        cout << "obj1 does not equal obj3\n";
    return 0;
}
```

Перегруженный оператор + должен поэлементно складывать оба операнда. Перегруженный оператор — должен вычитать все элементы правого операнда из элементов левого. Перегруженный оператор == должен возвращать значение **true**, если все элементы обоих операндов равны, в противном случае он должен возвращать значение **false**.

Задача 8.2. Переработайте программу Задача 8.1 так, чтобы перегрузить операторы с использованием дружественных функций.

Задача 8.3. Используя класс и функции Задача 8.1, перегрузите оператор ++ с помощью функции-члена класса, а оператор — с помощью дружест венной функции. (Перегрузите только префиксные формы операторов ++ и —).

Задача 8.4. Исследуйте следующую конструкцию:

```
#include <iostream >
using namespace std;
class mybase {
    int a, b;
public:
int c;
    void setab(int i, int j) { a = i; b = j; }
    void getab(int &i, int &j) { i = a; j = b; }
};
class derived1 : public mybase {
// ...
};
class derived2 : private mybase {
// ...
};
int main() {
    derived1 obj1;
    derived2 obj2;
    int i, j;
    // ...
}
```

Какая из следующих инструкций правильна внутри функции **main()** и почему?

```
A) obj1.getab(i, j);
B) obj2.getab(i, j);
C) obj1.c = 10;
D) obj2.c = 10;
```

Задача 8.5. Если бы переменные **a** и **b** внутри класса **myclass** Задача 8.4 стали не закрытыми (по умолчанию), а защищенными членами, изменился бы какой-нибудь из ваших ответов на вопросы этого упражнения? Если да, то почему?

Задача 8.6. В приведенном ниже фрагменте добавьте конструктор для класса **myderived**. Он должен передать указатель на инициализируемую строку конструктору класса **mybase**. Кроме того, конструктор **myderived()** должен инициализировать переменную **len** длиной строки.

```
#include <iostream>
#include <cstring>
using namespace std;
```

```
class mybase {
    char str[80];
public:
    mybase(char *s); { strcpy (str, s); }
    char *get() { return str; }
};
class myderived : public mybase {
    int len;
public:
    // add myderived() here...
    int getlen() { return len; }
    void show() { cout << get () << '\n'; }
};
int main() {
    myderived obo("hello");
    obj.show();
    cout << obj.getlen() << '\n';
    return 0;
}
```

Задача 8.7. Используя следующий фрагмент, создайте соответствующие конструкторы **car()** и **truck()**. Они должны передавать необходимые аргументы объектам класса **vehicle**. Кроме этого конструктор **car()** должен при создании объекта инициализировать переменную **passengers**, а конструктор **truck()** - переменную **loadlimit**.

```
#include <iostream >
using namespace std;
// A base class for various types of vehicle:
class vehicle {
    int num_wheels;
    int range;
public:
    vehicle(int w, int r) {
        num_wheels = w;
        range = r;
    }
    void showv() {
        cout << "Wheels: " << num_wheels << '\n';
        cout << "Range: " << range << '\n';
    }
};
class car : public vehicle {
    int passengers;
public:
    void show() {
        showv();
        cout << "Passengers: " << passengers << '\n';
    }
};
class truck : public vehicle {
    int loadlimit;
public:
     void show() {
        showv();
```

```
        cout << "Loadlimit: " << loadlimit << '\n';
    }
};
int main() {
    car objc(5, 4, 500);
    truck objt(3000, 12, 1200);
    cout << "Car:\n";
    objc.show();
    cout << "Truck:\n";
    objt.show();
    return 0;
}
```

Для конструкторов **car()** и **truck()** объекты должны объявляться следующим образом:

```
car obj(passengers, wheels, range);
truck obj(loadlimit, wheels, range);
```

Задача 8.8. Используя следующую иерархию классов, создайте конструктор класса C так, чтобы он инициализировал переменную k и передавал аргументы конструкторам A() и B().

```
#include <iostream >
using namespace std;
class A {
    int i;
public:
    A(int a) { i = a; }
};
classB {
    int j;
public:
    B(int a) { j = a; }
};
class C : public A, public B {
    int k;
public:
    /* Create C() so that it initializes k and passes arguments to both A() and B() */
};
```

Задача 8.9. Создайте исходный базовый класс **building** для хранения числа этажей и комнат в здании, а также общую площадь комнат. Создайте производный класс **house**, который наследует класс **building** и хранит число ванных комнат и число спален. Кроме этого создайте производный класс **office**, который наследует класс **building** и хранит число огнетушителей и телефонов.

Задача 8.10. Дан следующий фрагмент программы, впишите детали, как указано в комментариях:

```
#include <iostream>
using namespace std;
class planet {
protected:
    double distance ; // miles from the sun
    int revolve ; // in days
public:
    planet(double d, int r) { distance = d; revolve = r; }
};
class earth : public planet {
```

```
    double circumference; // circumference(окружность) of orbit
public:
   /* Create earth(double d, int r). Have it pass the distance and days of revolution back
to planet.
      Have it compute the circumference of the orbit. (Hint: circumference = 2r *3.1416.)
*/
   /* Create a function called show() that displays the information.*/
};
int main() {
    earth obj(93000000, 365);
    obj.show();
    return 0;
}
```

Задача 8.11. Иерархия классов из Задача 8.7 с классом vehicle. В программе имеется ошибка и код не форматирован. Найдите ошибку.

Подсказка: попытайтесь провести компиляцию и изучите сообщения об ошибках.

```
# include <iostream >
using namespace std;
// A base class for various types of vehicles .
class vehicle
{
int num_wheels ;
int range ;
public :
vehicle (int w, int r)
{
num_wheels = w;
range = r;
}
void showv ()
{
cout << " Wheels : " << num_wheels << '\n'; cout << " Range : " << range << '\n';
}
};
enum motor {gas , electric , diesel };
class motorized : public vehicle
{
enum motor mtr ;
public :
motorized ( enum motor m, int w, int r) : vehicle (w, r)
{
mtr = m;
}
void showm ()
{
cout << " Motor : ";
switch (mtr )
{
case gas : cout << "Gas \n";
break ;
case electric : cout << " Electric \n";
break ;
case diesel : cout << " Diesel \n";
```

```
break ;
}
}
};
class road_use : public vehicle
{
int passengers ;
public :
road_use (int p, int w, int r) : vehicle (w, r)
{
passengers = p;
}
void showr ()
{
cout << " Passengers : " << passengers << '\n';
}
};
enum steering { power , rack_pinion , manual };
class car : public motorized , public road_use
{
enum steering strng ;
public :
car ( enum steering s, enum motor m, int w, int r, int p) :
road_use (p, w, r), motorized (m, w, r), vehicle (w, r)
{
strng = s; }
void show () { showv (); showr (); showm (); cout << " Steering : "; switch ( strng ) {
case power : cout << " Power \n"; break ; case rack_pinion : cout << " Rack and Pinion
\n";
break ; case manual : cout << " Manual \n";
break ; } } }; int main () { car c(power , gas , 4, 500 , 5);
c. show ();
return 0;
}
```

При компиляции программы вы могли увидеть предупреждающее сообщение (или, возможно, сообщение об ошибке), связанное с использованием инструкции **switch** внутри классов **car** и **motorised**. Почему?

Задача 8.12. Здесь представлена переработанная версия класса **coord** из предыдущего Практического занятия. Форматирование кода нарушено. Теперь этот класс используется в качестве базового для класса **quad**, в котором помимо координат хранится номер квадранта, к которому принадлежит точка с этими координатами. Запустите программу и попытайтесь понять и объяснить полученный результат.

```
#include <
iostream>using namespace std;
class coord {
public:
int x, y; // coordinate values
coord () { x=0; y=0; }
coord (int i, int j) { x=i; y=j; }
void get_xy (int &i, int &j) { i=x; j=y; }
```

```
coord operator +( coord ob2);
coord operator -( coord ob2);
coord operator =( coord ob2);
};
// Overload + relative to coord class .
coord coord :: operator +( coord ob2)
{
coord temp ; cout << " Using coord operator +() \n"; temp .x = x + ob2 .x; temp .y = y +
ob2 .y; return temp ; } // Overload - relative to coord class . coord coord :: operator -(
coord ob2) {
coord temp ;
cout << " Using coord operator -() \n";
temp .x = x - ob2 .x;
temp .y = y - ob2 .y;
return temp ;
}
// Overload = relative to coord .
coord coord :: operator =( coord ob2)
{
cout << " Using coord operator =() \n";
x = ob2.x;
y = ob2.y;
return * this ; // return the object that is assigned to
}
class quad : public coord
{
int quadrant ;
public :
quad ()
{
x = 0;
y = 0;
quadrant = 0;
}
quad (int x, int y) : coord (x, y)
{
if(x >=0 && y >= 0)
quadrant = 1;
else if(x <0 && y >=0) quadrant = 2; else if(x <0 && y <0) quadrant = 3;
else quadrant = 4;
}
void showq ()
{
cout << " Point in Quadrant : " << quadrant << '\n';
}
quad operator =( coord ob2 );
};
quad quad :: operator =( coord ob2 )
{
cout << " Using quad operator =() \n";
x = ob2.x;
y = ob2.y;
if(x >=0 && y >= 0)
quadrant = 1;
else if(x <0 && y >=0)
```

```
quadrant = 2;
else if(x <0 && y <0)
quadrant = 3; else
quadrant = 4; return * this ; }
int main ()
{
quad o1 (10 , 10) , o2 (15 , 3) , o3;
int x, y;
o3 = o1 + o2; // add two objects - this calls operator +()
o3. get_xy (x, y);
o3. showq ();
cout << "(o1+o2) X: " << x << ", Y: " << y << "\n";
o3 = o1 - o2; // subtract two objects o3. get_xy (x, y);
o3. showq (); cout << "(o1 -o2) X: " << x << ", Y: " << y << "\n";
o3 = o1; // assign an object o3. get_xy (x, y); o3. showq (); cout << "(o3=o1) X: " << x
<< ", Y: " << y << "\n"; return 0;
}
```

Переработайте эту программу, чтобы в ней использовались дружественные оператор-функции.