Αντικειμενοστρεφής Προγραμματισμός

Εργαστήριο 3

Υπερφόρτωση Συναρτήσεων

Η C++ ως αντικειμενοστρεφής γλώσσα προγραμματισμού μας δίνει δυνατότητα να έχουμε επικάλυψη των συναρτήσεων. Μπορούμε δηλαδή να χρησιμοποιήσουμε το ίδιο όνομα για να περιγράψουμε δυο ή περισσότερες συναρτήσεις.
Στο παράδειγμά μας ορίζουμε τρεις διαφορετικές συναρτήσεις abs(int x), abs(float x), abs(double x) με το ίδιο όνομα, κάθε μία από τις οποίες επιστρέφει την απόλυτη τιμή ενός int, ενός float και ενός double αντίστοιχα. Ο μεταγλωττιστής θα αντιλαμβάνεται ποια από τις τρεις πρέπει να χρησιμοποιηθεί κάθε φορά, ανάλογα με το πλήθος και *των τύπο* των ορισμάτων που δίνονται.

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

int abs(int a);
double abs(double a);

int main()
{
	int i = -100;
	double j = 2.1345;

	cout << abs(i) << endl;
	cout << abs(j) << endl;
	system("pause");
	return 0;
}

int abs(int a)
{
	if(a < 0)
		return -a;
	else
		return a;
}

double abs(double a)
{
	if(a < 0)
		return -a;
	else
		return a;
}
```

Άσκηση 1:
Να ορίσετε μία κλάση `rectangle`, η οποία να περιέχει τα παρακάτω μέλη:

α) το μήκος και το ύψος του ορθογωνίου (να είναι private)
β) μία συνάρτηση που να επιστρέφει το εμβαδό του ορθογωνίου.
γ) δύο υπερφορτωμένες εκδόσεις της συνάρτησης `set_dimensions()`. Η πρώτη έκδοση να δέχεται δύο παραμέτρους (πλάτος και ύψος), ενώ η δεύτερη να δέχεται μόνο μια παράμετρο (το πλάτος και το ύψος θα είναι ίδια → τετράγωνο).

Να γραφεί ένα πρόγραμμα, το οποίο να δημιουργεί ένα αντικείμενο της κλάσης `rect`, να καλεί τις δύο εκδόσεις της `set_dimensions()` και να εμφανίζει κάθε φορά το εμβαδό του ορθογωνίου.

Υπερφόρτωση τελεστών

Η υπερφόρτωση τελεστή αποτελεί μια ειδική περίπτωση υπερφόρτωσης συνάρτησης. Η C++ επιτρέπει στον προγραμματιστή να εκχωρεί πρόσθετες λειτουργίες στους βασικούς τελεστές της C++, όπως είναι αυτοί της πρόσθεσης, της αφαίρεσης κλπ. Με αυτό τον τρόπο, οι βασικοί τελεστές της C++ μπορούν, συσχετιζόμενοι με μια κλάση (και όχι μόνο), να έχουν ιδιαίτερο αποτέλεσμα όταν εφαρμόζονται σε αντικείμενα της κλάσης αυτής.

Για παράδειγμα, σε μια κλάση Matrix η οποία περιέχει ως member variable ένα πίνακα m[], μπορούμε να ορίσουμε ότι ο τελεστής +, όταν εφαρμόζεται σε αντικείμενα τύπου Matrix να εκτελεί την πρόσθεση πινάκων. Προφανώς αν ο τελεστής + εφαρμοστεί σε απλές μεταβλητές (όπως int, float κλπ.) θα εκτελεί τη γνωστή πράξη της πρόσθεσης.

Για να επιτευχθεί η υπερφόρτωση τελεστών πρέπει να οριστεί μια συνάρτηση μέλος ή μια συνάρτηση φιλική προς την κλάση η οποία θα υλοποιεί όλες τις επιμέρους λειτουργίες που θα γίνονται όταν χρησιμοποιείται ο υπερφορτωμένος τελεστής. Στην περίπτωση των πινάκων που προαναφέρθηκαν, η συνάρτηση του τελεστή θα αποκτά πρόσβαση στους πίνακες m[] των αντικειμένων Matrix που θα προστεθούν, και προσθέτει ένα προς ένα τα αντίστοιχα μέλη των πινάκων αυτών.

Η γενική μορφή μιας τέτοιας συνάρτησης είναι :

- **τύπος όνομα_κλάσης::operator#(λίστα_ορισμάτων)** όπου :
- **τύπος** : Ο τύπος επιστροφής του στοιχείου το οποίο θα είναι το αποτέλεσμα της
- πράξης.
- **όνομα_ κλάσης** : Δηλώνει το όνομα της κλάσης με την οποία συσχετίζεται ο τελεστής
- # : Αποτελεί το τελεστή που μπορεί να υπερφορτωθεί. Μπορεί να είναι κάποιος

από τους :

!	~	+	-	*	&	/	%
<<	>>	<	<=	>	>=	==	!=
^	|	&&	>	+=	-=	*=	/=
^=	|=	&=	%=	<<=	>>=	,	→*
→	()	{}	=	++	--	new	delete

Σημειώνεται ότι εκτός από τον τελεστή = , η υπερφόρτωση τελεστών μπορεί να μεταβιβαστεί (inherited) και στις παραγόμενες κλάσεις της τάξης για την οποία ορίστηκε.

Προσοχή: Κάποιοι τελεστές δεν μπορούν να υπερφορτωθούν:

.	? :	::	.*	sizeof
Direct member selection	conditional	Scope resolution	Direct member selection	

Είναι γεγονός ότι μπορούμε να ορίσουμε ένα τελεστή να εκτελεί μια πράξη πολύ διαφορετική από αυτή που εκτελείται όταν τον εφαρμόζουμε σε απλές μεταβλητές. Για παράδειγμα, μπορούμε να ορίσουμε τον τελεστή '+' να εκτελεί αφαίρεση πινάκων. Όμως η υπερφόρτωση τελεστών για πράξεις άσχετες με τη βασική χρήση του τελεστή δεν αποτελεί καλή προγραμματιστική τεχνική γιατί μπορεί να μπερδέψει αυτούς που ενδεχομένως να χρησιμοποιήσουν την κλάση αυτή μελλοντικά.

Υπερφόρτωση μοναδιαίων τελεστών

Με τον όρο μοναδιαίοι (unary) τελεστές εννοούμε αυτούς που εφαρμόζονται σε μία μόνο μεταβλητή, π.χ. οι i++, ++i, κ.α. Αντίστοιχα, δυαδικοί (binary) είναι οι τελεστές που εφαρμόζονται σε δύο μεταβλητές, π.χ. i+j, x*y, κλπ.

```
#include <iostream>
using namespace std;
class Counter
   {
   private:
      int count;                  //count
   public:
      Counter() : count(0)        //constructor  no args
      {  }

      Counter(int c) : count(c)   //constructor, one arg
      {  }

      int get_count()             //return count
      { return count; }

      Counter operator ++ ()      //increment count (prefix)
      {                           //increment count, then return
         Counter temp(++count);   //create a temporary object
         return temp;             //and return it
      }

      Counter operator ++ (int)   //increment count (postfix)
      {                           //Alternative: return an unnamed temporary
         return Counter(count++); //object initialized to this
      }                           //count, then increment count
   };

int main()
{
   Counter c1, c2;                        //c1=0, c2=0

   cout << "\nc1=" << c1.get_count();    //display
   cout << "\nc2=" << c2.get_count();

   ++c1;                                  //c1=1
   c2 = ++c1;                             //c1=2, c2=2 (prefix)

   cout << "\nc1=" << c1.get_count();    //display
   cout << "\nc2=" << c2.get_count();

   c2 = c1++;                             //c1=3, c2=2 (postfix)

   cout << "\nc1=" << c1.get_count();    //display again
   cout << "\nc2=" << c2.get_count() << endl;
   return 0;
```

```
}
```

Άσκηση 2
Να υλοποιήσετε την κλάση Fraction η οποία αναπαριστά ένα κλάσμα της μορφής a/b (αριθμητής/παρονομαστής). Μια τέτοια κλάση μπορεί να χρησιμοποιηθεί σε περιπτώσεις που δεν θέλουμε να χρησιμοποιήσουμε δεκαδικούς αριθμούς, αλλά κλάσματα.

Να υλοποιήσετε επίσης τους τελεστές ++ (prefix και postfix) ώστε να αυξάνουν το κλάσμα κατά ένα.

Να υλοποιήσετε και τους απαραίτητους κατασκευαστές, καθώς και μια συνάρτηση που εκτυπώνει το κλάσμα με την μορφή: a/b

Υπερφόρτωση δυαδικών τελεστών

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

class Matrix
{
private:
	int arr[3];
public:
	Matrix(int dim1, int dim2, int dim3)
	{
		arr[0] = dim1;
		arr[1] = dim2;
		arr[2] = dim3;
	}

	Matrix()
	{
		arr[0] = 0;
		arr[1] = 0;
		arr[2] = 0;
	}

	void print()
	{
		cout << "Matrix=("<<arr[0]<<", "<<arr[1]<<", "<<arr[2] <<")"<< endl;
	}

	Matrix operator+(Matrix m)
	{
		Matrix temp;
		temp.arr[0] = arr[0] + m.arr[0];
		temp.arr[1] = arr[1] + m.arr[1];
		temp.arr[2] = arr[2] + m.arr[2];
		return temp;
	}

	Matrix operator-(Matrix m)
	{
		Matrix temp;
		temp.arr[0] = arr[0] - m.arr[0];
		temp.arr[1] = arr[1] - m.arr[1];
		temp.arr[2] = arr[2] - m.arr[2];
		return temp;
	}
};

int main()
{
	Matrix m1(1,3,5);
```

```
        Matrix m2(6,1,4);

        Matrix m3;
        m3 = m1 + m2;

        m3.print();
        system("pause");
        return 0;
}
```

Άσκηση 3

Στην παραπάνω κλάση Fraction να υλοποιήσετε και τους τελεστές + και – για την πρόσθεση και αφαίρεση μεταξύ κλασμάτων.
Παράδειγμα:
2/3 + 4/5 → 22/15

Σημείωση: Μην επιχειρήσετε να κάνετε απλοποίηση του κλάσματος. Χρησιμοποιείστε το ελάχιστο κοινό πολλαπλάσιο.