Εργαστήριο 8ο

Friend functions / operator overloading

Άσκηση 1 (friend functions, χρήση του const)

Να δημιουργήσετε το παρακάτω πρόγραμμα Lab8E1 που αναπαριστά σημεία στον δισδιάστατο χώρο. Το πρόγραμμα αποτελείται από την δήλωση της κλάσης Point (αρχείο Point.h)

```
class Point {
public:
	Point();
	Point(double x1, double y1);
	double get_x();
	double get_y();
private:
	double x;
	double y;
};
```

Καθώς και από τον ορισμό των μεθόδων της κλάσης (αρχείο Point.cpp)

```
#include "pch.h"
#include "Point.h"

Point::Point() : x(0), y(0)
{
}

Point::Point(double x1, double y1) : x{ x1 }, y{ y1 }
{
}

double Point::get_x()
{
	return x;
}

double Point::get_y()
{
	return y;
}
```

Και τέλος από το αρχείο που περιέχει το κυρίως πρόγραμμα (συνάρτηση main)

```
#include <iostream>
#include "Point.h"
using namespace std;

bool equal(Point a, Point b);

int main()
{
	Point s1, s2(10, 0);
	if (equal(s1, s2)) {
		cout << "Points s1 and s2 are the same\n";
	}
	else {
		cout << "Points s1 and s2 are different\n";
	}
	return 0;
```

```
}

bool equal(Point a, Point b)
{
	return (a.get_x() == b.get_x()) && (a.get_y() == b.get_y());
}
```

Να απαντήσετε στις παρακάτω ερωτήσεις:

1. Μπορεί η equal να λειτουργήσει χωρίς την χρήση των accessor methods (get_x() & get_y()); Αν όχι γιατί;
2. Θα μπορούσε να υλοποιηθεί η equal σαν method; Ποια θα ήταν η διαφοροποίηση της σε σχέση με την υπάρχουσα function;

Να πραγματοποιήσετε τις παρακάτω αλλαγές

1. Βάλτε σε σχόλια την δήλωση και τον ορισμό της function equal. Στην θέση της βάλτε την παρακάτω δήλωση και ορισμό.

```
class Point {
public:
	friend bool equal(Point& a, Point& b);

(υπόλοιπος κώδικας class, main)

bool equal(Point& a, Point& b) {
	return ((a.x == b.x) && (a.y == b.y));
}
```

Απαντήστε στις παρακάτω ερωτήσεις

   a. Είναι method η friend function;
   b. Ποια είναι η διαφορά της από τις κανονικές functions;
   c. Ποιο είναι το πλεονέκτημα στο να περνάνε object σαν ορίσματα με την χρήση call by reference;
   d. Χρειάζεται η equal να κάνει αλλαγές στις τιμές των x,y των παραμέτρων; Μπορεί να κάνει; Πως θα το διορθώσετε για να μην κάνει αλλαγές κατά λάθος;
2. Να δημιουργήσετε μια friend function με το όνομα add που θα δέχεται σαν ορίσματα δύο Point object. Η add θα επιστρέφει ένα Point object που έχει σαν συντεταγμένες το άθροισμα των συντεταγμένων των ορισμάτων της. Δοκιμάστε την στην main.
3. Δημιουργήστε μια μέθοδο με το όνομα print_point που τυπώνει τις συντεταγμένες ενός σημείου. Δοκιμάστε την στην main.

Απαντήστε στις παρακάτω ερωτήσεις:

1. Μπορεί η μέθοδος print_point να μεταβάλει τα private fields του object το οποίο την καλεί; Χρειάζεται να το κάνει; Πως μπορούμε να προστατέψουμε τα private fields;

ΒΑΣΙΚΕΣ ΕΝΝΟΙΕΣ ΤΗΣ ΑΣΚΗΣΗΣ
Ποια είναι η διαφορά μιας friend function από μία κανονική;
Γιατί όταν τα ορίσματα είναι objects καλό είναι να περνάν by reference;

Ποια η λειτουργία του const σε όρισμα object by reference;
Ποια η λειτουργία του const στο τέλος την δήλωσης μιας method;

Άσκηση 2 (overloading operators – constructor for conversion)

Μετατρέψτε τον κώδικα της add σε αυτόν που φαίνεται παρακάτω:

```
class Point {
public:
	friend Point operator +(const Point& a, const Point& b);
(υπόλοιπος κώδικας)
…
Point operator +(const Point& a, const Point& b) {
	Point sum;
	sum.x = a.x + b.x;
	sum.y = a.y + b.y;
	return sum;
}
```

Δοκιμάστε μέσα στην main τον παρακάτω κώδικα

```
s1 = s1 + s2;
	cout << "to s1 einai " << s1.print_point() << endl;
```

Να απαντήσετε στις παρακάτω ερωτήσεις;

1. Μπορείτε να θεωρήσετε ότι ένας operator (όπως π.χ. ο +) είναι στην ουσία μια function;
2. Αν απαντήσατε ναι στην προηγούμενη ερώτηση τότε ποιο θεωρείτε ότι είναι το όνομα αυτής της function;

Μετατρέψτε και την print_point που έχετε δημιουργήσει σε έναν overloaded operator με τον παρακάτω κώδικα

```
class Point {
public:
	friend ostream& operator <<(ostream &out, const Point &a);
(υπόλοιπος κώδικας)
…
ostream & operator<<(ostream & out, const Point & a)
{
	out << "x=" << a.x << " y=" << a.y << endl;
	return out;
}
```

Να απαντήσετε στις παρακάτω ερωτήσεις:

1. Ποιο είναι το πρώτο όρισμα του operator <<;
2. Ποιος είναι το τύπος επιστροφής του operator <<;

Να πραγματοποιήσετε τις παρακάτω αλλαγές
1. Να υπερφορτώσετε τον τελεστή (==). Η συνάρτηση equal μπορεί να σας βοηθήσει
2. Να υπερφορτώσετε τον τελεστή (>>) ώστε να διαβάζει τις δύο συντεταγμένες ενός σημείου.

Προσθέστε τον παρακάτω constructor που δημιουργεί ένα Point από έναν μόνο δεκαδικό αριθμό (τον τοποθετεί στην συντεταγμένη x και τοποθετεί 0 στην y).

```
Point(double a); // μέσα στην δήλωση της κλάσης

Point::Point(double a) : x(a), y(0)
{
}
```

Μέσα στην main εκτελέστε τον κώδικα s1 = s1 + 40; Θα εκτελεστεί; Αν ναι γιατί;

Άσκηση 3

Να δημιουργήσετε μία κλάση που να περιγράφει ένα Matlab matrix με στοιχεία ακέραιους αριθμούς. Συγκεκριμένα η κλάση θα περιέχει τις ακόλουθες μεταβλητές μέλη:

- Δύο που θα περιγράφουν τον αριθμό γραμμών και στηλών του matrix
- Μία τύπου vector που θα περιέχει τα στοιχεία του matrix. Τα στοιχεία του matrix θα αποθηκεύονται στο vector κατά στήλες.

Στην παραπάνω κλάση θα υπερφορτώσετε τους παρακάτω τελεστές
1. Ανάθεσης. Θα αναθέτει τιμές στα στοιχεία που βρίσκονται στις αντίστοιχες θέσεις
2. Τελεστή <<. Θα πρέπει να τον τυπώνει με την μορφή matrix (δηλαδή κάθε γραμμή σε μία γραμμή)
3. Τελεστή >>. Θα πρέπει να διαβάζει τα στοιχεία (κατά στήλη).
4. Ισότητας. Θα πρέπει να είναι τα στοιχεία στις αντίστοιχες θέσης ίσα
5. Ανισότητας. Θα πρέπει να μην είναι ίσα
6. Πρόσθεσης. Θα προσθέτει τα στοιχεία που βρίσκονται στην ίδια θέση.
7. Μικρότερο. Θα ελέγχει τα στοιχεία με την σειρά. Θα πρέπει το πρώτο στοιχείο που είναι διαφορετικό να είναι μικρότερο
8. Μεγαλύτερο. Θα ελέγχει τα στοιχεία με την σειρά (όπως είναι στο vector). Θα πρέπει το πρώτο στοιχείο που είναι διαφορετικό να είναι μεγαλύτερο
9. Αύξησης, μείωσης κατά ένα
10. Index. Θα έχει την ίδια λειτουργικότητα με το linear indexing του Matlab

Οι τελεστές θα πρέπει να εφαρμόζονται μόνο σε matrix που έχουν τις ίδιες διαστάσεις. Θα φροντίσετε για όλες τις καλές πρακτικές που αναφέρθηκαν στην θεωρία του μαθήματος. Να δημιουργήσετε μία main function που να δοκιμάζει όλους τους τελεστές που δημιουργήσατε.

Άσκηση 3

Να δημιουργήσετε μία κλάση που να περιγράφει την ώρα μιας ημέρας. Η κλάση θα αποτελείται από τρεις ακεραίους που θα περιγράφουν την ώρα τα λεπτά και τα δευτερόλεπτα αντίστοιχα (σε μορφή 0-24).
Στην παραπάνω κλάση θα υπερφορτώσετε τους παρακάτω τελεστές.
1. Ανάθεσης. Θα αναθέτει τιμές στην ώρα, λεπτά, δευτερόλεπτα.

2. Τελεστή <<. Θα πρέπει να τον τυπώνει με την μορφή HH:MM:SS
3. Τελεστή >>. Θα πρέπει να διαβάζει τις ώρες, λεπτά, δευτερόλεπτα (αν είναι μεγαλύτερα να επιστρέφει 0,0,0).
4. Ισότητας. Θα πρέπει να είναι η ίδια ώρα
5. Ανισότητας. Θα πρέπει να μην είναι η ίδια ώρα
6. Πρόσθεσης. Θα προσθέτει τις δύο ώρες. (προσοχή αν γίνεται υπέρβαση στα όρια λεπτών, δευτερολέπτων). Αν γίνεται υπέρβαση στις ώρες θα ξεκινάει από το 0
7. Μικρότερο. Θα ελέγχει αν η μία ώρα είναι μικρότερη από την άλλη
8. Μεγαλύτερο. Θα ελέγχει αν η μία ώρα είναι μεγαλύτερη από την άλλη
9. Index. Θα παίρνει index 1-3 και θα επιστρέφει δευτερόλεπτα, λεπτά, ώρες αντίστοιχα. Σε διαφορετική τιμή θα τυπώνει μήνυμα λάθους και θα τερματίζει το πρόγραμμα.

Θα φροντίσετε για όλες τις καλές πρακτικές που αναφέρθηκαν στην θεωρία του μαθήματος. Να δημιουργήσετε μία main function που να δοκιμάζει όλους τους τελεστές που δημιουργήσατε.