Πανεπιστήμιο Δυτικής Μακεδονίας
Τμήμα Μηχανικών Πληροφορικής & Τηλεπικοινωνιών
Αντικειμενοστραφής Προγραμματισμός I
Εργαστήριο 4

Στο 4^{ο} εργαστήριο θα ασχοληθούμε την υπερφόρτωση

Στόχοι εργαστηρίου:

- Συναρτήσεις Friends
- Συναρτήσεις global

Έχουμε την κλάση Money η οποία έχει 2 δεδομένα μέλη:

- int dollars;
- int cents;

Η κλάση Money έχει τις ακόλουθες συναρτήσεις:

- Money();
- Money(int d, int c);
- void display_money( );
- int getDollars();
- int getCents();

καθώς και 1 φιλική συνάρτηση η οποία συγκρίνει τα 2 αντικειμένα Money που παίρνει σαν ορίσματα και τυπώνει μήνυμα σχετικά με το ποιο έχει περισσότερα χρήματα:

- friend void compare(Money m1, Money m2);

επίσης θέλουμε να αθροίζουμε τα συνολικά χρήματα 2 αντικειμένων Money και να τα αποθηκεύουμε σε ένα τρίτο αντικείμενο τύπου Money. Αυτό μπορεί να γίνει με 3 τρόπους:

- με συνάρτηση μέλος **void add(Money m1, Money m2);** Η οποία αθροίζει τα χρήματα των αντικειμένων m1, m2 και το αποτέλεσμα το αποθηκεύει στο αντικείμενο που καλεί τη συνάρτηση

- με φιλική συνάρτηση **friend Money addMoney(Money m1, Money m2)** η οποία αθροίζει τα χρήματα των αντικειμένων m1, m2 και το αποτέλεσμα το αποθηκεύει στο αντικείμενο που επιστρέφει.

- με global συνάρτηση **Money MoneyAdd(Money m1, Money m2)** η οποία αθροίζει τα χρήματα των αντικειμένων m1, m2 και το αποτέλεσμα το αποθηκεύει στο αντικείμενο που επιστρέφει.

Money.h

```
class Money
{
   public:
      Money();
      Money(int d, int c);
      void add(Money m1, Money m2);
      void display_money( );
      int getDollars();
      int getCents();
      friend void compare(Money m1, Money m2);
      friend Money addMoney(Money m1, Money m2);
   private:
      int dollars;
      int cents;
};
```

Money.cpp

```
#include<iostream>
#include<cstdlib>
#include "money.h"
using namespace std;
Money::Money()
{
}

Money::Money(int d, int c)
{
      dollars = d;
      cents = c;
}

void Money::display_money()
{
    cout << "$" << dollars << "." << cents << endl;
}
int Money::getDollars()
{
return dollars;
}
int Money::getCents()
```

```
{
        return cents;
}
void Money::add(Money m1, Money m2)
{
   int extra = 0;
   cents = m1.cents + m2.cents;
   if(cents >=100){
      cents = cents - 100;
      extra = 1;
   }

   dollars = m1.dollars + m2.dollars + extra;
}

void compare(Money m1, Money m2)
{
 if ((m1.dollars*100+m1.cents)>(m2.dollars*100+m2.cents))
    cout<<"First object has more money";
    else cout<< "Second object has more money";

}
Money addMoney(Money m1, Money m2)
{
        int d=m1.dollars+m2.dollars;
        int c=m1.cents+m2.cents;
        if (c>100)
        {
                c=c-100;
                d=d+1;
        }
        Money m(d,c);
        return m;
}
```

Main.cpp

```
#include <iostream>
#include "money.h"
using namespace std;
/* run this program using the console pauser or add your own getch, system("pause") or input
loop */

Money MoneyAdd(Money m1, Money m2);

int main(int argc, char** argv) {

   int d, c;
   Money m1, m2, sum;
   m1 = Money(1,23);

  cout << "The first money is:";
   m1.display_money();
```

```
   m2 = Money(2,98);
   cout << "The second money is:";
   m2.display_money();

    compare(m1,m2);
    cout<<endl;

   sum.add(m1,m2);
   cout <<endl<< "The sum is with member function usage:";
   sum.display_money();

   Money m4=addMoney(m1,m2);
   cout <<endl<< "The sum is with friend function usage:";
   m4.display_money();

          Money m5=MoneyAdd(m1,m2);
          cout <<endl<< "The sum is with global function usage:";
           m4.display_money();

          return 0;
}

Money MoneyAdd(Money m1, Money m2)
{
          int d=m1.getDollars()+m2.getDollars();
          int c=m1.getCents()+m2.getCents();
          if (c>100)
          {
                    c=c-100;
                    d=d+1;
          }
          Money m(d,c);
          return m;
}
```

<u>Άσκηση εργαστηρίου</u>

Να ορισθεί μία κλάση με το όνομα Student με τρια δεδομένα μέλη:
int am για τον αριθμό μητρώου
char[] για το ονοματεπώνυμο
int grades[8] για τις βαθμολογίες σε 8 μαθήματα (πίνακας).

Επίσης, να γραφούν οι εξής συναρτήσεις μέλη:

- Μία συνάρτηση για την ανάθεση τιμών στα δεδομένα της κλάσης (SetData())
- Μία συνάρτηση για την εμφάνιση των τιμών (PrintData())
- Μία συνάρτηση για την επιστροφή της μέγιστης βαθμολογίας
- Μία συνάρτηση για την επιστροφή της ελάχιστης βαθμολογίας
- Μία συνάρτηση για την επιστροφή του μέσου όρου βαθμολογίας
- Μία συνάρτηση get για την επιστροφή του private δεδομένου

- Να γίνει συνάρτηση μέλος compareStudents η οποία θα συγκρίνει το μέσο όρο 2 φοιτητών και θα επιστρέφει το όνομα του φοιτητή με το μεγαλύτερο ΜΟ.
- Να γίνει φιλική συνάρτηση findOlder προς την κλάση Student η οποία θα συγκρίνει τους ΑΜ 2 φοιτητών και θα επιστρέφει το φοιτητή ο οποίος είναι αρχαιότερος (έχει μικρότερο αριθμό μητρώου)
- Να γίνει global findStudentsAvg συνάρτηση η οποία θα υπολογίζει το μέσο όρο των βαθμών 2 φοιτητών.

Ακολούθως, να γραφεί πρόγραμμα όπου θα δηλώνονται 2 αντικείμενα της κλάσης Student και θα αρχικοποιούνται με τη συνάρτηση SetData.

- Να τυπώσετε τη μικρότερη και τη μεγαλύτερη νβαθμολογία των 2 φοιτητών.
- Να τυπώσετε το ΜΟ των 2 φοιτητών.
- Να συγκρίνετε τους ΜΟ όρους των 2 φοιτητών με την compareStudents και να τυπώσετε μήνυμα με το όνομα του φοιτητή με το μεγαλύτερο ΜΟ.
- Να χρησιμοποιήσετε τη φιλική συνάρτηση findOlder για την εύρεση του αρχαιότερου φοιτητή και την εκτύπωση των δεδομένων του.
- Να χρησιμοποιήσετε τη global συνάρτηση findStudentsAvg για την εύρεση του ΜΟ των βαθμών των 2 φοιτητών και την εκτύπωση του.