ΤΕΙ ΣΕΡΡΩΝ

ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΣΕΡΡΩΝ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΕΠΙΚΟΙΝΩΝΙΩΝ

ΑΝΤΙΚΕΙΜΕΝΟΣΤΡΑΦΗΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ

(Σημειώσεις Εργαστηρίου)

Ευάγγελος Γ. Ούτσιος

Σέρρες 2004

ΠΕΡΙΕΧΟΜΕΝΑ

ΤΕΙ ΣΕΡΡΩΝ
ΣΤΕΦ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΕΠΙΚΟΙΝΩΝΙΩΝ
Αντικειμενοστραφής προγραμματισμός

Εργαστήριο 1

(Κλάσεις και αντικείμενα)

Να αναπτυχθεί πρόγραμμα όπου θα δηλώνεται μία κλάση με το όνομα **vector** με δύο μέλη-δεδομένα **int x**, **int y** και τις συναρτήσεις μέλη:

- **void readData()**
- **void setData(int xi, int yi)**
- **void printData(void)**

Στη συνέχεια:

- Να δηλώνονται δύο αντικείμενα **v1** και **v2** τύπου **vector.** Το **v1** να τίθεται (1,2) και να τυπώνεται στην οθόνη.
- Το **v2** να δίνεται από τον χρήστη και μετά να τυπώνεται στην οθόνη.
- Να γίνει δοκιμή προσπέλασης αλλαγής των μελών-δεδομένων των δύο αντικειμένων.
- Να γίνει συνάρτηση εγκατάστασης (**constructor**) χωρίς ορίσματα.
- Να γίνει δεύτερη συνάρτηση εγκατάστασης (υπερφόρτωση) με ορίσματα. (**int xi, int yi**) και το **v1** να αρχικοποιείται στη δήλωσή του.
- Να γίνει συνάρτηση αποσύνδεσης (**destructor**) χωρίς ορίσματα.
- Να γίνει συνάρτηση μέλος **metro** χωρίς παραμέτρους και με επιστρεφόμενη τιμή τύπου **float**, η οποία να επιστρέφει το μέτρο (μήκος) του διανύσματος (ως γνωστόν αν $\vec{A} = (x, y)$ τότε $|\vec{A}| = \sqrt{x^2 + y^2}$. Στη συνέχεια να υπολογίζεται και να εμφανίζεται στην οθόνη το μέτρο των διανυσμάτων **v1** και **v2**.

Μία απλή κλάση

Το παρακάτω πρόγραμμα περιέχει μία κλάση και δύο αντικείμενα αυτής της κλάσης. Αν και απλό, το πρόγραμμα δείχνει τη βασική δομή και τα γενικά χαρακτηριστικά των κλάσεων της C++.

```
#include <iostream.h>
#include <stdio.h>
#include <conio.h>
#include <string.h>

class Person
{
   private:
         char name[30];
         int age;
   public:
         Person()
         {
            strcpy(name," ");
            age = 0;
         }
         Person(char name1[], int age1)
         {
            strcpy(name, name1);
            age = age1;
         }
         void readData()
         {
            cout << "    Enter name:";
            cin >> name;
            cout << "    Enter age:";
            cin >> age;
         }
         void printData()
         {
            cout << "    The name of the person is " << name << endl;
            cout << "    The age of the person is " << age << endl;
         }
};          // τέλος κλάσης

void main()
{
   Person  p1;        // δήλωση  αντικειμένου,  χρήση constructor χωρίς ορίσματα
   Person  p2("GEORGIOU", 35);  // δήλωση  αντικειμένου, χρήση constructor με
                                // ορίσματα
   cout << "Input data for object p1:" << endl;
   p1.readData();     // κλήση συνάρτησης-μέλους για ορισμό δεδομένων
```

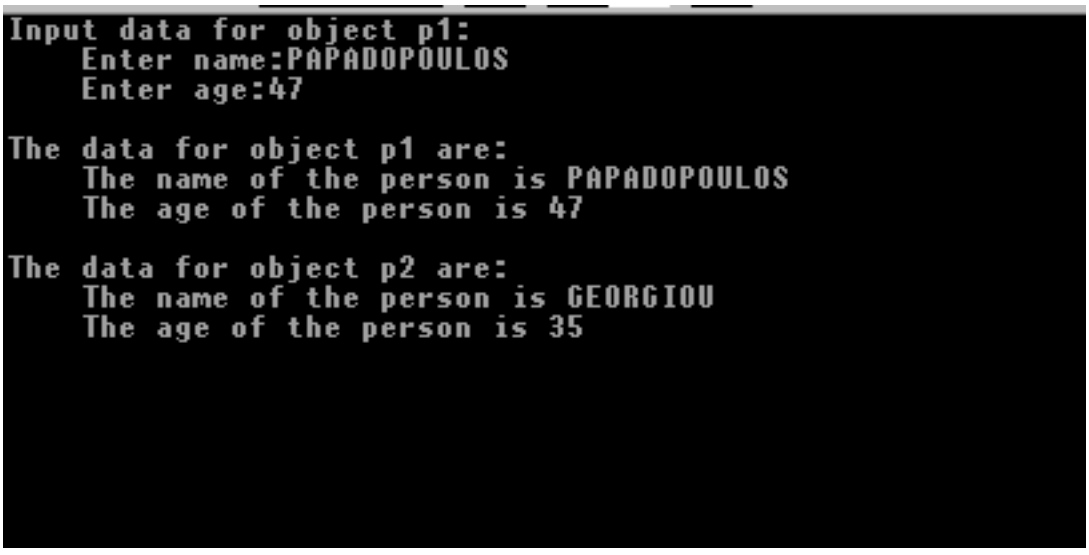
```
   cout << "The data for object p1 are:" << endl;
   p1.printData();    // κλήση συνάρτησης μέλους για εμφάνιση δεδομένων

   cout << "The data for object p2 are:" << endl;
   p2.printData();   // κλήση συνάρτησης μέλους για εμφάνιση δεδομένων
   getch();
}
```

Η κλάση Person περιέχει δύο στοιχεία δεδομένων και δύο βασικές συναρτήσεις-μέλη. Αυτές οι δύο συναρτήσεις παρέχουν μοναδική πρόσβαση στα στοιχεία δεδομένων έξω από την κλάση. Η πρώτη συνάρτηση-μέλος – *readData()* - μας δίνει τη δυνατότητα να πληκτρολογήσουμε δύο τιμές και να τις αποδώσουμε άμεσα στα στοιχεία δεδομένων και η δεύτερη συνάρτηση-μέλος – *printData()* - εμφανίζει αυτές τις τιμές.

Η τοποθέτηση των δεδομένων και των συναρτήσεων μαζί, σε μία ενότητα, είναι η κεντρική ιδέα του αντικειμενοστραφούς προγραμματισμού.

Επιπρόσθετα, η κλάση περιέχει δύο συναρτήσεις-μέλη, που έχουν το ίδιο όνομα με την κλάση. Οι συναρτήσεις αυτές ονομάζονται συναρτήσεις εγκατάστασης (*constructors*) και έχουν σαν σκοπό να αρχικοποιούν τα δεδομένα ενός αντικειμένου όταν αυτό δηλώνεται. Όταν κατά τη δήλωση ενός αντικειμένου δε μεταβιβάζονται κάποιες τιμές, τότε εκτελείται ο constructor χωρίς ορίσματα, που μπορεί να μη κάνει τίποτα ή π.χ. να μηδενίσει τα δεδομένα. Όταν όμως μεταβιβάζονται τιμές στη δήλωση κάποιου αντικειμένου, τότε οι τιμές αυτές αποδίδονται αυτόματα στα δεδομένα-μέλη του αντικειμένου.

Έξοδος προγράμματος



Συμπληρωματικές ασκήσεις

Άσκηση 1

Να ορισθεί μία κλάση με το όνομα Akeraios με ένα δεδομένο μέλος τύπου int Επίσης, να γραφούν οι εξής συναρτήσεις μέλη:

- Μία συνάρτηση για την ανάθεση τιμής στο δεδομένο της κλάσης
- Μία συνάρτηση για την εμφάνιση της τιμής
- Δύο συναρτήσεις εγκατάστασης, μία χωρίς όρισμα και μία με όρισμα για την αρχικοποίηση της τιμής του δεδομένου μέλους
- Μία συνάρτηση που θα δέχεται μία ακέραια τιμή (ως εκθέτη) και θα υπολογίζει τη δύναμη του δεδομένου

Ακολούθως, να γραφεί πρόγραμμα όπου θα δηλώνεται ένα αντικείμενο της κλάσης Akeraios και θα γίνεται προσπέλαση των συναρτήσεων μελών.

Άσκηση 2

Να ορισθεί μία κλάση με το όνομα Time με τρία δεδομένα μέλη για τις ώρες, τα λεπτά και τα δευτερόλεπτα. Επίσης, να γραφούν οι εξής συναρτήσεις μέλη:

- Μία συνάρτηση για την ανάθεση τιμών στα δεδομένα της κλάσης
- Μία συνάρτηση για την εμφάνιση των τιμών στη μορφή ΩΩ:ΛΛ:ΔΔ
- Δύο συναρτήσεις εγκατάστασης, μία χωρίς όρισμα και μία με όρισμα για την αρχικοποίηση των τιμών των δεδομένων

Ακολούθως, να γραφεί πρόγραμμα όπου θα δηλώνεται ένα αντικείμενο της κλάσης Time και θα γίνεται προσπέλαση των συναρτήσεων μελών.

Άσκηση 3

Να ορισθεί μία κλάση με το όνομα Student με τρία δεδομένα μέλη για τον αριθμό μητρώου, το ονοματεπώνυμο και τις βαθμολογίες σε 8 μαθήματα (πίνακας).

Επίσης, να γραφούν οι εξής συναρτήσεις μέλη:

- Μία συνάρτηση για την ανάθεση τιμών στα δεδομένα της κλάσης
- Μία συνάρτηση για την εμφάνιση των τιμών
- 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

Ακολούθως, να γραφεί πρόγραμμα όπου θα δηλώνεται ένα αντικείμενο της κλάσης Student και θα γίνεται προσπέλαση των συναρτήσεων μελών.

ΤΕΙ ΣΕΡΡΩΝ
ΣΤΕΦ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΕΠΙΚΟΙΝΩΝΙΩΝ
Αντικειμενοστραφής προγραμματισμός

Εργαστήριο 2

(Αντικείμενα ως ορίσματα συναρτήσεων, επιστροφή αντικειμένων από συναρτήσεις)

Στο πρόγραμμα του προηγούμενου εργαστηρίου, όπου δηλώνεται μία κλάση με όνομα **vector,** να γραφούν οι παρακάτω συναρτήσεις:

- **void add(vector v, vector u)**
  Δέχεται ως παραμέτρους δύο αντικείμενα τύπου vector και υπολογίζει το άθροισμα των συντεταγμένων τους, το οποίο εκχωρεί στο αντικείμενο που η συνάρτηση είναι μέλος. Ως γνωστό,
  $v_3.x = v_1.x + v_2.x$
  $v_3.y = v_1.y + v_2.y$

- **vector plus(vector v)**
  Δέχεται ως παράμετρο ένα αντικείμενο τύπου vector και υπολογίζει το άθροισμα των συντεταγμένων του με τις συντεταγμένες του αντικειμένου που η συνάρτηση είναι μέλος και επιστρέφει το άθροισμα αυτό σε ένα τρίτο αντικείμενο.

- **vector kprod(int k)**
  Δέχεται ως παράμετρο έναν ακέραιο και υπολογίζει το γινόμενο του ακεραίου με τις συντεταγμένες του αντικειμένου που η συνάρτηση είναι μέλος και επιστρέφει το γινόμενο αυτό σε ένα νέο αντικείμενο.
  Ως γνωστό,
  $v_2.x = v_1.x * k$
  $v_2.y = v_1.y * k$

- **void diff(vector v, vector u)**
  *Να γραφεί κατ' αντιστοιχία της **add**.*

- **vector minus(vector v)**
  *Να γραφεί κατ' αντιστοιχία της **plus**.*

- **int innerprod(vector v)**
  Δέχεται ως παράμετρο ένα αντικείμενο τύπου vector και αφού υπολογίσει το εσωτερικό γινόμενο αυτού του διανύσματος με το αντικείμενο που η συνάρτηση είναι μέλος, επιστρέφει το γινόμενο αυτό. Ως γνωστόν, το εσωτερικό γινόμενο δύο διανυσμάτων v_1 και v_2 είναι
  $v_1.x * v_2.x + v_1.y * v_2.y$

Αντικείμενα σαν ορίσματα συναρτήσεων, επιστροφή αντικειμένων από συναρτήσεις

Στο πρόγραμμα που ακολουθεί, ορίζεται μία κλάση Account με ένα στοιχείο δεδομένων balance, που αναφέρεται στο τρέχον ποσό ενός λογαριασμού. Ορίζονται, επίσης, δύο συναρτήσεις-μέλη για την ανάληψη, *withdraw(),* και την κατάθεση, *deposit(),* χρημάτων στο λογαριασμό και μία συνάρτηση-μέλος, *getBalance(),* που επιστρέφει, κατά την κλήση της, το τρέχον ποσό λογαριασμού.

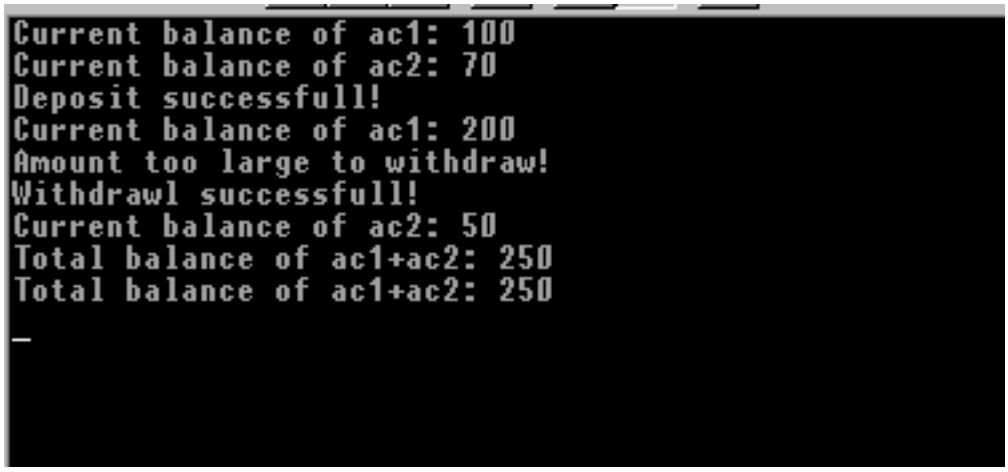
Επίσης, ορίζεται μία συνάρτηση-μέλος, *addBalance(),* η οποία έχει σα στόχο να προσθέσει τα ποσά δύο διαφορετικών λογαριασμών. Τα ποσά αυτά, αφού προστεθούν, αποδίδονται στο αντικείμενο στο οποίο η συνάρτηση είναι μέλος – δηλ. στο αντικείμενο που κάλεσε τη συνάρτηση.

Τέλος, ορίζεται μία συνάρτηση-μέλος, *sumBalance(),* η οποία έχει σα στόχο, επίσης, να προσθέσει τα ποσά δύο διαφορετικών λογαριασμών, χρησιμοποιώντας όμως μία διαφορετική προσέγγιση. Μεταβιβάζεται μόνο το ένα αντικείμενο σαν όρισμα, και η τιμή του δεδομένου αυτού του αντικειμένου προστίθεται στην τιμή του δεδομένου του αντικειμένου στο οποίο η συνάρτηση είναι μέλος – δηλαδή του αντικειμένου με το οποίο καλείται η συνάρτηση. Το αποτέλεσμα αποδίδεται σε ένα προσωρινό αντικείμενο *temp*, που ορίζεται μέσα στη συνάρτηση. Τέλος, με την εντολή *return,* το προσωρινό αντικείμενο επιστρέφεται στο σημείο κλήσης και αποδίδεται ανάλογα.

```
#include <iostream.h>
#include <stdio.h>
#include <conio.h>

class Account
{
   private:
         float  balance;
   public:
         Account()
         {
            balance = 0;
         }
        Account(float balance1)
         {
            balance = balance1;
         }
```

```
        void  withdraw(float  money)
        {
           if (money <= balance)
          {
              balance = balance - money;
           cout << "Withdrawl successfull!" << endl;
          }
           else
              cout << "Amount too large to withdraw!" << endl;
        }
        void deposit(float  money)
        {
           balance += money;
           cout << "Deposit successfull!" << endl;
        }
        float getBalance()
        {
           return  balance;
        }
        void addBalance(Account x, Account y)
        {
           balance = x.balance + y.balance;
        }
        Account sumBalance(Account ac)
        {
           Account temp;
           temp.balance = balance + ac.balance;
           return temp;
        }
};

void main()
{
   Account  ac1(100.0), ac2(70.0), ac3, ac4;

   cout << "Current balance of ac1: " << ac1.getBalance() << endl;
   cout << "Current balance of ac2: " << ac2.getBalance() << endl;
   ac1.deposit(100.0);
   cout << "Current balance of ac1: " << ac1.getBalance() << endl;
   ac2.withdraw(80.0);
   ac2.withdraw(20.0);
   cout << "Current balance of ac2: " << ac2.getBalance() << endl;
   ac3.addBalance(ac1, ac2);
   cout << "Total balance of ac1+ac2: " << ac3.getBalance() << endl;
   ac4 = ac1.sumBalance(ac2);
   cout << "Total balance of ac1+ac2: " << ac4.getBalance() << endl;
   getch() ;
}
```

Έξοδος προγράμματος



Συμπληρωματικές ασκήσεις

Άσκηση 1

Στην προηγούμενη κλάση Akeraios, να γραφεί μία συνάρτηση με το όνομα addValues() η οποία θα δέχεται δύο αντικείμενα ως ορίσματα, θα προσθέτει τα δεδομένα τους και θα εκχωρεί το αποτέλεσμα στο αντικείμενο που η συνάρτηση είναι μέλος.

Να ξαναγραφεί η συνάρτηση ώστε να δέχεται ένα αντικείμενο ως όρισμα, να προσθέτει το δεδομένο του αντικειμένου αυτού με το δεδομένο του αντικειμένου που η συνάρτηση είναι μέλος και να επιστρέφει το αποτέλεσμα σε ένα τρίτο αντικείμενο.

Άσκηση 2

Στην προηγούμενη κλάση Time, να γραφεί να γραφεί μία συνάρτηση με το όνομα addTime() η οποία θα δέχεται δύο αντικείμενα ως ορίσματα, θα προσθέτει τους δύο χρόνους και θα εκχωρεί το αποτέλεσμα στο αντικείμενο που η συνάρτηση είναι μέλος.

Να ξαναγραφεί η συνάρτηση ώστε να δέχεται ένα αντικείμενο ως όρισμα, να προσθέτει το χρόνο του αντικειμένου αυτού με το χρόνο του αντικειμένου που η συνάρτηση είναι μέλος και να επιστρέφει το αποτέλεσμα σε ένα τρίτο αντικείμενο.

ΤΕΙ ΣΕΡΡΩΝ
ΣΤΕΦ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΕΠΙΚΟΙΝΩΝΙΩΝ
Αντικειμενοστραφής προγραμματισμός

Εργαστήριο 3

(Υπερφόρτωση αριθμητικών τελεστών)

1. Στο πρόγραμμα του προηγούμενου εργαστηρίου, όπου δηλώνεται μία κλάση με όνομα **vector,** να γραφούν οι παρακάτω συναρτήσεις:

- Μία συνάρτηση όπου θα γίνεται υπερφόρτωση του τελεστή '+'.
  Η συνάρτηση να είναι κατ' αντιστοιχία της ***plus****.*
- Μία συνάρτηση όπου θα γίνεται υπερφόρτωση του τελεστή '**-**'.
  Η συνάρτηση να είναι κατ' αντιστοιχία της ***minus****.*
- Μία συνάρτηση όπου θα γίνεται υπερφόρτωση του τελεστή '*****'.
  Η συνάρτηση να είναι κατ' αντιστοιχία της ***innerprod****.*

2. Να ορισθεί η κλάση **circle** με δεδομένα μέλη **x, y, r** και να γραφούν συναρτήσεις:

- constructors χωρίς ορίσματα και με ορίσματα
- Για τον υπολογισμό της *περιμέτρου* του κύκλου
- Για τον υπολογισμό του *εμβαδού* του κύκλου
- Για τον υπολογισμό του *εμβαδού της κυκλικής περιοχής* που προκύπτει από τη διαφορά των εμβαδών δύο ομόκεντρων κύκλων. Η συνάρτηση να ξαναγραφεί με υπερφόρτωση του τελεστή '**-**'.

Υπερφόρτωση αριθμητικών τελεστών

Είδαμε σε προηγούμενο παράδειγμα πώς δύο αντικείμενα Account μπορούν να προστεθούν, χρησιμοποιώντας μία συνάρτηση-μέλος:

ac3 = ac1.sumBalance(ac2);

Αν χρησιμοποιηθεί ο τελεστής + με υπερφόρτωση, τότε μπορούμε να έχουμε την εξής πρόταση:

ac3 = ac1 + ac2;

Το πρόγραμμα που ακολουθεί, υλοποιεί αυτή την περίπτωση:

```
#include <iostream.h>
#include <stdio.h>
#include <conio.h>

class Account
{
   private:
        float  balance;
   public:
        Account()
        {
           balance = 0;
        }
        Account(float balance1)
        {
           balance = balance1;
        }
        void  withdraw(float  money)
        {
           if (money <= balance)
             {
               balance = balance - money;
               cout << "Withdrawl successfull!" << endl;
             }
           else
              cout << "Amount too large to withdraw!" << endl;
        }
        void deposit(float  money)
        {
           balance += money;
          cout << "Deposit successfull!" << endl;
        }
        float getBalance()
        {
           return  balance;
        }
```

```
    Account operator + (Account ac)
    {
       Account temp;
       temp.balance = balance + ac.balance;
       return temp;
    }
};

void main()
{
   Account  ac1(100.0), ac2(70.0), ac3;

   cout << "Current balance of ac1: " << ac1.getBalance() << endl;
   cout << "Current balance of ac2: " << ac2.getBalance() << endl;

   ac3 = ac1 + ac2;
   cout << "Total balance of ac1+ac2: " << ac3.getBalance() << endl;
   getch() ;
}
```

Όταν στη main() εκτελείται η πρόταση

```
ac3 = ac1 + ac2;
```

τότε γίνεται υπερφόρτωση του τελεστή + (επειδή τα ac1 και ac2 έχουν ορισθεί ως αντικείμενα), προστίθενται τα δεδομένα των αντικειμένων ac1 και ac2 και το αποτέλεσμα αποδίδεται στο αντικείμενο ac3.

Να διευκρινισθεί, ότι η συνάρτηση χρησιμοποιεί ως όρισμα το αντικείμενο που βρίσκεται δεξιά του τελεστή (π.χ. το ac2). Ακόμα, η συνάρτηση είναι μέλος στο αντικείμενο που βρίσκεται αριστερά του τελεστή (π.χ. στο ac1) και έτσι η αναφορά στα δεδομένα αυτού του αντικειμένου είναι άμεση.

Έξοδος προγράμματος

Συμπληρωματικές ασκήσεις

Άσκηση 1

Στην προηγούμενη κλάση Akeraios, να γίνει τροποποίηση ώστε αντί να χρησιμοποιείται η συνάρτηση addValues() για να προσθέτει δύο ακεραίους, να χρησιμοποιείται ο τελεστής + με υπερφόρτωση.

Επίσης να γίνει χρήση του τελεστή – με υπερφόρτωση.

Άσκηση 2

Στην προηγούμενη κλάση Time, να γίνει τροποποίηση ώστε αντί να χρησιμοποιείται η συνάρτηση addTime() για να προσθέτει δύο χρόνους, να χρησιμοποιείται ο τελεστής + με υπερφόρτωση.

Επίσης να γίνει χρήση του τελεστή – με υπερφόρτωση.

ΤΕΙ ΣΕΡΡΩΝ
ΣΤΕΦ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΕΠΙΚΟΙΝΩΝΙΩΝ
Αντικειμενοστραφής προγραμματισμός

Εργαστήριο 4

(Υπερφόρτωση τελεστών σύγκρισης)

1. Στο πρόγραμμα του προηγούμενου εργαστηρίου, όπου δηλώνεται μία κλάση με όνομα **vector,** να γραφούν οι παρακάτω συναρτήσεις:

- Μία συνάρτηση όπου θα γίνεται υπερφόρτωση του τελεστή '=='.
  Η συνάρτηση θα δέχεται ένα αντικείμενο τύπου vector ως παράμετρο και θα ελέγχει τις συντεταγμένες αυτού του διανύσματος με τις συντεταγμένες του αντικειμένου τύπου vector που η συνάρτηση είναι μέλος. Αν οι αντίστοιχες συντεταγμένες των δύο διανυσμάτων είναι ίσες,, τότε η συνάρτηση θα επιστρέφει τη λογική τιμή true, ενώ σε άλλη περίπτωση θα επιστρέφει θα επιστρέφει false.

- Μία συνάρτηση όπου θα γίνεται υπερφόρτωση του τελεστή '**!**='.
  Η συνάρτηση θα είναι παρόμοια με την προηγούμενη, μόνο που θα επιστρέφει true όταν τα δύο διανύσματα δεν έχουν και τις δύο συντεταγμένες τους ίσες.

2. Στο πρόγραμμα όπου έχει ορισθεί η κλάση **circle** με δεδομένα μέλη **x, y, r** να γραφούν οι συναρτήσεις:

- Μία συνάρτηση όπου θα γίνεται υπερφόρτωση του τελεστή '==='.
  Η συνάρτηση θα επιστρέφει true όταν δύο κύκλοι έχουν τις ακτίνες τους ίσες, αλλιώς θα επιστρέφει false.

- Μία συνάρτηση όπου θα γίνεται υπερφόρτωση του τελεστή '>'.
  Η συνάρτηση θα επιστρέφει true όταν η ακτίνα ενός κύκλου είναι μεγαλύτερη από την ακτίνα ενός δεύτερου κύκλου.

- **bool omok(circle c)**
  Η συνάρτηση θα δέχεται ένα αντικείμενο τύπου circle ως παράμετρο και θα ελέγχει αν ο κύκλος αυτός είναι ομόκεντρος με το αντικείμενο τύπου circle που η συνάρτηση είναι μέλος. Αν είναι, τότε θα επιστρέφει true, ενώ σε άλλη περίπτωση θα επιστρέφει false.

Υπερφόρτωση τελεστών σύγκρισης

Όπως έχουμε την υπερφόρτωση αριθμητικών τελεστών, με παρόμοιο τρόπο μπορούμε να υλοποιήσουμε την υπερφόρτωση τελεστών σύγκρισης. Για παράδειγμα, στο επόμενο πρόγραμμα θα χρησιμοποιήσουμε τον τελεστή > (μεγαλύτερο από) με υπερφόρτωση, στην κλάση Account, για να μπορούμε να συγκρίνουμε δύο λογαριασμούς:

```
#include <iostream.h>
#include <stdio.h>
#include <conio.h>

class Account
{
   private:
        float  balance;
   public:
        Account()
        {
           balance = 0;
        }
       Account(float balance1)
        {
           balance = balance1;
        }
        void  withdraw(float  money)
        {
           if (money <= balance)
             {
                balance = balance - money;
                cout << "Withdrawl successfull!" << endl;
             }
           else
              cout << "Amount too large to withdraw!" << endl;
        }
        void deposit(float  money)
        {
           balance += money;
           cout << "Deposit successfull!" << endl;
        }
        float getBalance()
        {
           return  balance;
        }
```

```
        bool operator > (Account ac)
        {
           if (balance > ac.balance)
              return true;
           else
              return false;
        }
};

void main()
{
   Account  ac1(100.0), ac2(70.0);

   cout << "Current balance of ac1: " << ac1.getBalance() << endl;
   cout << "Current balance of ac2: " << ac2.getBalance() << endl;

   if (ac1 > ac2)
      cout << "Balance of ac1 > balance of ac2." << endl;
   else
      if (ac2 > ac1)
         cout << "Balance of ac2 > balance of ac1." << endl;
      else
         cout << "Balance of ac1 = balance of ac2." << endl;
   getch() ;
}
```

Στο προηγούμενο πρόγραμμα, όταν εκτελείται η εντολή

```
if (ac1 > ac2)
```

προκαλείται η κλήση της συνάρτησης με υπερφόρτωση. Αυτή με τη σειρά της συγκρίνει τα ποσά των δύο λογαριασμών και ανάλογα επιστρέφει μία λογική τιμή true ή false. Η τιμή αυτή ελέγχεται άμεσα από την εντολή ελέγχου

```
if (ac1 > ac2)
```

και τυπώνεται το ανάλογο μήνυμα.

Ο παραπάνω έλεγχος θα μπορούσε να γραφεί και ως εξής:

```
bool  result;

result = (ac1 > ac2);
if (result)      // δηλ. Αν το result == true
```

Έξοδος προγράμματος

Συμπληρωματικές ασκήσεις

Άσκηση 1

Στην προηγούμενη κλάση Akeraios, να γίνει υπερφόρτωση του τελεστή > για τη σύγκριση δύο αντικειμένων. Κατά τον έλεγχο θα επιστρέφεται μία boolean τιμή true/false ανάλογα αν ο πρώτος ακέραιος είναι μεγαλύτερος από το δεύτερο.

Επίσης να γίνει υπερφόρτωση του τελεστή ==.

Άσκηση 2

Στην προηγούμενη κλάση Time, να γίνει υπερφόρτωση του τελεστή > για τη σύγκριση δύο χρόνων. Κατά τον έλεγχο θα επιστρέφεται μία boolean τιμή true/false ανάλογα αν ο πρώτος χρόνος είναι μεγαλύτερος από το δεύτερο.

Επίσης να γίνει υπερφόρτωση του τελεστή ==.

Άσκηση 3

Στην προηγούμενη κλάση Student, να γίνει υπερφόρτωση του τελεστή > για τη σύγκριση δύο μέσων όρων βαθμολογίας για δύο διαφορετικούς σπουδαστές. Κατά τον έλεγχο θα επιστρέφεται μία boolean τιμή true/false ανάλογα αν ο πρώτος μέσος όρος βαθμολογίας είναι μεγαλύτερος από το δεύτερο.

Επίσης να γίνει υπερφόρτωση του τελεστή ==.

ΤΕΙ ΣΕΡΡΩΝ
ΣΤΕΦ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΕΠΙΚΟΙΝΩΝΙΩΝ
Αντικειμενοστραφής προγραμματισμός

Εργαστήριο 5

(Δείκτες και αντικείμενα)

1. Στην κλάση **vector** των προηγούμενων εργαστηρίων**,** να γραφεί η παρακάτω συνάρτηση:

```
void add(vector *p1, vector *p2)
{
  x = p1->x + p2->x;
  y = p1->y + p2->y;
}
```

Είναι η γνωστή συνάρτηση που προσθέτει τις συντεταγμένες δύο αντικειμένων τύπου vector και τις αποδίδει ως συντεταγμένες του αντικειμένου που η συνάρτηση είναι μέλος. Η αναφορά στα αντικείμενα γίνεται μέσω δεικτών.

Η κλήση της add γίνεται ως εξής:

```
c.add(&a,&b);    // όπου τα a, b, c είναι αντικείμενα τύπου vector
```
ή
```
p->add(&a,&b);  // όπου το p είναι δείκτης σε αντικείμενο τύπου vector
```

Επίσης να εκτελεσθεί το παρακάτω:

```
q = new vector(-8,-10);
q->print();
p = &c;
cout << "Address of object a is " << &a << endl;
cout << "Address of object b is " << &b << endl;
cout << "Address of object c is " << &c << endl;
cout << "Value  of pointer p is " << p << endl;
cout << "Address of pointer p is " << &p << endl;
cout << "Address of pointer q is " << &q << endl;
cout << "Value of pointer q -at heap where object is created by new- is "
     << endl;
delete q;
```

2. Για την κλάση **circle** προηγούμενων προγραμμάτων να γραφούν οι συναρτήσεις:

- **float perimeter()**
- **float area()**
- **float diffarea(circle *c)**

όπου θα καλούνται με τη χρήση δεικτών. Η δημιουργία αντικειμένων να γίνεται με την new.

Δείκτες και αντικείμενα

Οι δείκτες, πέρα από τους συνήθεις τύπους δεδομένων, μπορούν να δείχνουν και σε αντικείμενα. Αν κατά τη στιγμή που γράφουμε ένα πρόγραμμα, δε γνωρίζουμε πόσα αντικείμενα θα δημιουργήσουμε, τότε μπορούμε να χρησιμοποιήσουμε τον τελεστή new. Ο new επιστρέφει ένα δείκτη σε ένα ανώνυμο αντικείμενο και μπορούμε έτσι να δημιουργήσουμε αντικείμενα κατά τη διάρκεια εκτέλεσης του προγράμματος.

```
#include <iostream.h>
#include <stdio.h>
#include <conio.h>

class Person
{
   private:
         char name[30];
         int age;
   public:
         Person()
         {
            strcpy(name," ");
            age = 0;
         }
         Person(char name1[], int age1)
         {
            strcpy(name, name1);
            age = age1;
         }
         void readData()
         {
            cout << "    Enter name:";
            cin >> name;
            cout << "    Enter age:";
            cin >> age;
         }
         void printData()
         {
            cout << "    The name of the person is " << name << endl;
            cout << "    The age of the person is " << age << endl;
         }
};

void main()
{
   Person  *p1, *p2, *p3;   // δήλωση  δεικτών σε αντικείμενα
   Person  a("GEORGIOU", 35); // δήλωση  αντικειμένου, χρήση constructor
```

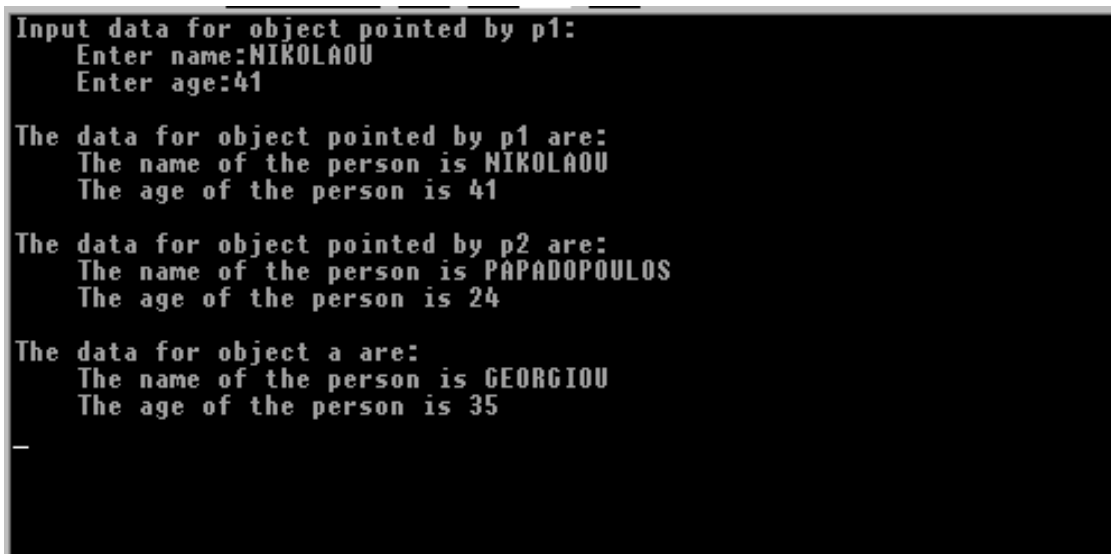
```
   cout << "Input data for object pointed by p1:" << endl;
   p1 = new Person; // Δημιουργία αντικειμένου
   p1->readData();
   cout << endl << "The data for object pointed by p1 are:" << endl;
   p1->printData();

   p2 = new Person("PAPADOPOULOS", 24); // Δημιουργία αντικειμένου,
                                        // απόδοση τιμών με χρήση
                                        // constructor με ορίσματα
   cout << endl << "The data for object pointed by p2 are:" << endl;
   p2->printData();

   p3 = &a;
   cout << endl << "The data for object a are:" << endl;
   p3->printData();

   delete p1;
   delete p2;
   delete p3;
   getch();
}
```

Στο προηγούμενο παράδειγμα, δηλώνονται τρεις δείκτες p1, p2 και p3 σε αντικείμενα τύπου Person. Με τον τελεστή new δεσμεύεται μνήμη για κάποιο αντικείμενο και ορίζεται ο δείκτης p1 να δείχνει στο συγκεκριμένο αντικείμενο. Για να αναφερθούμε στις συναρτήσεις-μέλη του αντικειμένου χρησιμοποιούμε τον τελεστή προσπέλασης μέλους ->.

Επίσης, με τον τελεστή new δεσμεύεται μνήμη για κάποιο άλλο αντικείμενο και ορίζεται ο δείκτης p2 να δείχνει σ' αυτό το αντικείμενο. Ταυτόχρονα με τη δημιουργία του αντικειμένου, του αποδίδονται και συγκεκριμένες τιμές, με χρήση της συνάρτησης εγκατάστασης με ορίσματα.

Τέλος, η διεύθυνση ενός τρίτου αντικειμένου a, το οποίο δημιουργείται κατά τη δήλωσή του, αποδίδεται στο δείκτη p3. Έτσι, η προσπέλαση της συνάρτησης για την εμφάνιση των δεδομένων του αντικειμένου γίνεται, εναλλακτικά, με τη χρήση του δείκτη.

Έξοδος προγράμματος



Συμπληρωματικές ασκήσεις

Άσκηση 1

Να γραφεί πρόγραμμα όπου θα ορίζεται η κλάση Triangle με τα εξής δεδομένα-μέλη:

float base;
float height;

που αντιπροσωπεύουν τις δύο κάθετες πλευρές ενός ορθογωνίου τριγώνου.

Επίσης, να γραφούν οι συναρτήσεις-μέλη:

- **float emvado()**
  Η συνάρτηση θα υπολογίζει και θα επιστρέφει το εμβαδόν του τριγώνου.
- **float ypotinousa()**
  Η συνάρτηση θα υπολογίζει και θα επιστρέφει την υποτείνουσα του τριγώνου.

Επίσης, να γραφούν κατάλληλες συναρτήσεις εγκατάστασης χωρίς ορίσματα και με ορίσματα. Να δηλωθούν δείκτες προς αντικείμενα της κλάσης Triangle, να δημιουργηθούν αντικείμενα με την new και να γίνει προσπέλαση των συναρτήσεων μελών μέσω των δεικτών.

Άσκηση 2

Να γραφεί πρόγραμμα όπου θα ορίζεται η κλάση Spoudastis με τα εξής δεδομένα-μέλη:

int am;
float grade;

που αντιπροσωπεύουν τον αριθμό μητρώου και την αντίστοιχη βαθμολογία για κάποιο σπουδαστή.

Επίσης, να γραφεί η εξής συνάρτηση-μέλος:

- **void compareGrade(Spoudastis *s)**
  Η συνάρτηση θα δέχεται έναν δείκτη σε ένα αντικείμενο της κλάσης Spoudastis και θα συγκρίνει τη βαθμολογία αυτού του σπουδαστή με τη βαθμολογία του σπουδαστή που αντιστοιχεί στο αντικείμενο που η συνάρτηση είναι μέλος. τυπώνοντας το αριθμό μητρώου του σπουδαστή που έχει τη μεγαλύτερη βαθμολογία.

Επίσης, να γραφούν κατάλληλες συναρτήσεις εγκατάστασης χωρίς ορίσματα και με ορίσματα. Να δηλωθούν δείκτες προς αντικείμενα της κλάσης Spoudastis, να δημιουργηθούν αντικείμενα με την new και να γίνει προσπέλαση των συναρτήσεων μελών μέσω των δεικτών.

ΤΕΙ ΣΕΡΡΩΝ
ΣΤΕΦ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΕΠΙΚΟΙΝΩΝΙΩΝ
Αντικειμενοστραφής προγραμματισμός

Εργαστήριο 6

(***Μετατροπή αντικειμένων διαφορετικών κλάσεων***)

- Να μελετηθούν τα προγράμματα που μετατρέπουν μία απόσταση από πόδια και ίντσες σε μία αντίστοιχη απόσταση σε μέτρα και εκατοστά.
- Να αναπτυχθούν δύο προγράμματα που θα μετατρέπουν ένα αντικείμενο της κλάσης vectRec, ορισμένο με **ορθογώνιες** συντεταγμένες (*rectangular coordinates*), σε ένα αντίστοιχο αντικείμενο, της κλάσης vectPol, με **πολικές** συντεταγμένες (*polar coordinates*).

Τα δεδομένα-μέλη των δύο κλάσεων ορίζονται ως εξής:

```
class vectRec
{
  private:
      float x, y;
};
```

```
class vectPol
{
  private:
      float len, ang;
};
```

Να γραφούν – και για τις δύο κλάσεις – συναρτήσεις εγκατάστασης χωρίς ορίσματα και με ορίσματα, καθώς επίσης και συναρτήσεις εμφάνισης των δεδομένων.

1ο πρόγραμμα: ***Ρουτίνα στο αντικείμενο προέλευσης***

Θα χρησιμοποιηθεί μία συνάρτηση μετατροπής - **operator vectPol() -** που θα είναι τοποθετημένη στην κλάση προέλευσης (δηλ. στην vectRec).

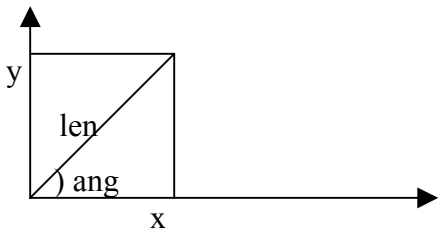
2ο πρόγραμμα: ***Ρουτίνα στο αντικείμενο προορισμού***

Θα χρησιμοποιηθεί μία συνάρτηση εγκατάστασης ενός ορίσματος - **vectPol(vectRec r) -** που θα είναι τοποθετημένη στην κλάση προορισμού (δηλ. στην vectPol). Επίσης θα πρέπει να γραφεί μία συνάρτηση (get) για την προσπέλαση των ιδιωτικών δεδομένων (x, y) της κλάσης vectRec.

Σημ.:

Για την μετατροπή από ορθογώνιες σε πολικές συντεταγμένες, να χρησιμοποιηθούν τα παρακάτω:

```
len = sqrt(x*x+y*y);
ang = atan(y/x) ;
```



Μετατροπές μεταξύ αντικειμένων διαφορετικών κλάσεων

Για να μετατρέψουμε αντικείμενα μεταξύ τους, που είναι ορισμένα σε διαφορετικές κλάσεις, ακολουθούμε την εξής διαδικασία.:

- Χρησιμοποιούμε μία συνάρτηση εγκατάστασης (constructor) με ένα όρισμα, αν θέλουμε η ρουτίνα μετατροπής να βρίσκεται στην κλάση του αντικειμένου προορισμού (δηλ. αριστερά από το =)
- Χρησιμοποιούμε μία συνάρτηση μετατροπής (operator), αν θέλουμε η ρουτίνα μετατροπής να βρίσκεται στην κλάση του αντικειμένου προέλευσης (δηλ. δεξιά από το =).

Τα δύο επόμενα παραδείγματα υλοποιούν και τις δύο αυτές περιπτώσεις, για τη μετατροπή μίας «Αγγλικής» απόστασης (πόδια και ίντσες) σε μία «Ελληνική» απόσταση (μέτρα και εκατοστά).

1) *Ρουτίνα στο αντικείμενο προέλευσης*

```
#include <iostream.h>
const float MTF = 3.28033;

class GrDist
{
   private:
         int m;
         float  cm;
   public:
         GrDist()
         {
            m = 0;
            cm = 0;
         }
         GrDist(int m1, float  cm1)
         {
            m = m1;
            cm = cm1;
         }
         void printDist()
         {
            cout << "    " << m << " meters, " << cm << " centimetres." << endl;
         }
};

class EngDist
{
   private:
         int feet;
         float  inches;
```

```
   public:
        EngDist()
        {
           feet = 0;
           inches = 0;
        }
        EngDist(int feet1, float  inches1)
        {
           feet = feet1;
           inches = inches1;
        }
        void printDist()
        {
           cout << "    " << feet << " feet, " << inches << " inches." << endl;
        }
        operator GrDist()
        {
           int met;
           float ekat, mf;
           mf = (feet+inches/12)/MTF;
           met = int(mf);
           ekat = (mf-met)*100;
           return  GrDist(met, ekat);
        }
};
main()
{
   GrDist  gr;
   EngDist  eng(5, 10.0);

   gr = eng;
   cout << endl << "English distance:" << endl;
   eng.printDist();
   cout << endl << "Greek Distance:" << endl;
   gr.printDist();
}
```

Έξοδος προγράμματος

```
English distance:
    5 feet, 10 inches.

Greek distance:
    1 meters, 77.8276 centimetres.
```

2) *Ρουτίνα στο αντικείμενο προορισμού*

```
#include <iostream.h>
const float MTF = 3.28033;

class EngDist
{
   private:
         int feet;
         float  inches;
   public:
         EngDist()
         {
            feet = 0;
            inches = 0;
         }
         EngDist(int feet1, float  inches1)
         {
            feet = feet1;
            inches = inches1;
         }
         void printDist()
         {
            cout << "    " << feet << " feet, " << inches << " inches." << endl;
         }
         int  getFeet()
         {
            return  feet;
         }
         float  getInches()
         {
            return  inches;
         }
};

class GrDist
{
   private:
         int m;
         float  cm;
   public:
         GrDist()
         {
            m = 0;
            cm = 0;
         }
         GrDist(int m1, float  cm1)
         {
            m = m1;
            cm = cm1;
         }
```

```
        GrDist(EngDist e)
        {
           int  ft;
           float  in, mf;
           ft = e.getFeet();
           in = e.getInches();
           mf = (ft + in/12) / MTF;
           m = int(mf);
           cm = (mf – m) * 100;
        }
        void printDist()
        {
           cout << “    “ << m << “ meters, “ << cm << “ centimetres.” << endl;
        }
};

main()
{
   GrDist  gr;
   EngDist  eng(5, 10.0);

   gr = eng;
   cout << endl << “English distance:“ << endl;
   eng.printDist();
   cout << endl << “Greek Distance:“ << endl;
   gr.printDist();
}
```

Έξοδος προγράμματος

```
English distance:
    5 feet, 10 inches.

Greek distance:
    1 meters, 77.8276 centimetres.
```

Συμπληρωματικές ασκήσεις

Άσκηση 1

Να αναπτυχθούν δύο προγράμματα που θα μετατρέπουν ένα αντικείμενο της κλάσης vectPol, ορισμένο με **πολικές** συντεταγμένες (*polar coordinates*), σε ένα αντίστοιχο αντικείμενο της κλάσης vectRec με **ορθογώνιες** συντεταγμένες (*rectangular coordinates*).

Τα δεδομένα-μέλη των δύο κλάσεων ορίζονται ως εξής:

```
class vectPol
{
   private:
        float len, ang;
};
```

```
class vectRec
{
   private:
        float x, y;
};
```

Να γραφούν – και για τις δύο κλάσεις – συναρτήσεις εγκατάστασης χωρίς ορίσματα και με ορίσματα, καθώς επίσης και συναρτήσεις εμφάνισης των δεδομένων.

Το πρώτο πρόγραμμα να χρησιμοποιεί μία ρουτίνα μετατροπής στο αντικείμενο προέλευσης, ενώ το δεύτερο πρόγραμμα να χρησιμοποιεί μία συνάρτηση εγκατάστασης στο αντικείμενο προορισμού.

Άσκηση 2

Να γραφούν δύο προγράμματα που με παρόμοιο τρόπο θα μετατρέπουν μία "ελληνική" απόσταση (m, cm) σε μία "αγγλική" απόσταση (feet, inches).

ΤΕΙ ΣΕΡΡΩΝ

ΣΤΕΦ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΕΠΙΚΟΙΝΩΝΙΩΝ

Αντικειμενοστραφής προγραμματισμός

Εργαστήριο 7

(***Υπερφόρτωση συνάρτησης/κλήση με αναφορά/απόδοση αρχικών τιμών***)

- Να μελετηθεί το φυλλάδιο που αναφέρεται στην υπερφόρτωση συνάρτησης, κλήση με αναφορά και απόδοση αρχικών τιμών.
- Να αναπτυχθεί ένα πρόγραμμα όπου θα ορίζεται μία κλάση Polygon με ένα δεδομένο-μέλος

```
char  typos;
```

και τις εξής συναρτήσεις μέλη:

void getType()

Η συνάρτηση θα ζητά από το χρήστη να κάνει μία επιλογή από το παρακάτω μενού και θα εκχωρεί την επιλογή στο δεδομένο-μέλος.

```
S. Square
T. Triangle
   Choice?
```

void processType()

Η συνάρτηση θα ελέγχει την επιλογή που έγινε και θα καλεί τις ανάλογες συναρτήσεις για την εισαγωγή τιμών και τον υπολογισμό του εμβαδού.
Η κλήση των συναρτήσεων θα γίνεται με το δείκτη ***this****.*
Η κλήση των συναρτήσεων για την εισαγωγή τιμών θα γίνεται με αναφορά

void getValues(float &s)

Η συνάρτηση θα επιστρέφει την πλευρά του τετραγώνου που θα δίνει ο χρήστης κατόπιν προτροπής.

void getValues(float &b, float &h)

Η συνάρτηση θα επιστρέφει τη βάση και το ύψος του τριγώνου που θα δίνει ο χρήστης κατόπιν προτροπής.
Η συνάρτηση έχει το ίδιο όνομα με την προηγούμενη (υπερφόρτωση).

void emvado(float x)

Η συνάρτηση θα υπολογίζει και θα εμφανίζει το εμβαδόν του τετραγώνου.

void emvado(float x, float y)

Η συνάρτηση θα υπολογίζει και θα εμφανίζει το εμβαδόν του τριγώνου.
Η συνάρτηση έχει το ίδιο όνομα με την προηγούμενη (υπερφόρτωση).

- ***Απόδοση αρχικών τιμών***
- ***Υπερφόρτωση συνάρτησης***
- ***Κλήση με αναφορά***

```
//--- Απόδοση αρχικών τιμών ----------
#include <iostream.h>
#include <stdio.h>
#include <conio.h>

void line(int num = 10,  char shape = '-');

void main()
{
   line( 5,'.');
   line(4);
   line();
   getch();
}

void line(int num, char shape)
{
   for(;num;num--)
        cout << shape;
   cout << endl;
}
```

Έξοδος προγράμματος

Στο παραπάνω παράδειγμα δηλώνεται μία συνάρτηση line() η οποία αναπαράγει κάποιο χαρακτήρα ένα καθορισμένο πλήθος φορών. Στα ορίσματα της συνάρτησης έχουν αποδοθεί κάποιες αρχικές τιμές, οι οποίες θα χρησιμοποιηθούν, όταν εκτελεσθεί η συνάρτηση, εφόσον δεν έχουμε δώσει κάποιες αντίστοιχες τιμές κατά την κλήση της συνάρτησης.. Θα πρέπει να τονισθεί ότι μπορούμε να παραλείψουμε κάποια τιμή κατά την κλήση μόνο από το τέλος. Για παράδειγμα η παρακάτω κλήση δεν θα ήταν αποδεκτή:

line(‘*”);

προσδοκώντας ότι η συνάρτηση θα τύπωνε 10 αστερίσκους.

```
// ---- Υπερφόρτωση συνάρτησης ----------------
#include <iostream.h>
#include <stdio.h>
#include <conio.h>

void print(float x);
void print(int x);
void print(char x);
void print(float x, float y);
main()
{
   float a = 2.4;
   int b=1;

   print(a);
   print(float(4.3));
   print('+');
   print(b);
   print((int)5);
   print(5.1,2.4);
   getch();
}
void print(float x)
{
   cout << "f = " << x << endl;
}
void print(int x)
{
   cout << "i = " << x << endl;
}
void print(char x)
{
   cout << "c = " << x << endl;
}
void print(float x, float y)
{
   cout << "s = " << x+y << endl;
}
```

Έξοδος προγράμματος

```
// ----- Ψευδώνυμα – κλήση με αναφορά ----
void  callByPointer(int *p);
void  callByReference(int &p);

main()
{
   int i=500, &p=i;  // Ψευδώνυμο

   cout << &i << ":" << i << " = " << &p << ":" << p << endl;

   callByPointer(&i);
   cout << i << endl;
   callByReference(i);
   cout << i << endl;
}
void  callByPointer(int *p)
{
  (*p)++;
}
void  callByReference(int &p)
{
   p++;
}
```

Έξοδος προγράμματος

```
Address of i: 0063FE00
Value of i: 500
Address of p: 0063FE00
Value of p: 500
Value of i after callByPointer: 501
Value of i after callByReference: 502
_
```

Συμπληρωματικές ασκήσεις

Άσκηση 1

Να γραφεί πρόγραμμα για την επίλυση εξισώσεων $1^{ου}$ και $2^{ου}$ βαθμού. Στο πρόγραμμα θα χρησιμοποιούνται με –υπερφόρτωση- οι εξής συναρτήσεις μέλη:

- **void readCoefs(float &a, float &b)**
  Η συνάρτηση θα διαβάζει και θα επιστρέφει τους συντελεστές a και b μίας πρωτοβάθμιας εξίσωσης. Η κλήση της συνάρτησης θα γίνεται με αναφορά.
- **void readCoefs(float &a, float &b, float &c)**
  Η συνάρτηση θα διαβάζει και θα επιστρέφει τους συντελεστές a και b μίας δευτεροβάθμιας εξίσωσης. Η κλήση της συνάρτησης θα γίνεται με αναφορά.
- **void exisosi(float a, float b)**
  Η συνάρτηση θα δέχεται τις τιμές των συντελεστών a και b μίας πρωτοβάθμιας εξίσωσης και αφού βρίσκει τη λύση θα εμφανίζει τα αποτελέσματα..
- **void exisosi(float a, float b, float c)**
  Η συνάρτηση θα δέχεται τις τιμές των συντελεστών a, b και c μίας δευτεροβάθμιας εξίσωσης και αφού βρίσκει τη λύση θα εμφανίζει τα αποτελέσματα..

Άσκηση 2

Να γραφεί πρόγραμμα για τον υπολογισμό του εμβαδού διαφόρων γεωμετρικών σχημάτων. Στο πρόγραμμα θα χρησιμοποιείται με –υπερφόρτωση- η εξής συνάρτηση μέλος:

- **float emvado(float r)**
  Η συνάρτηση θα δέχεται την ακτίνα ενός κύκλου και αφού υπολογίζει, θα επιστρέφει το εμβαδόν του κύκλου.
- **float emvado(float a, float b)**
  Η συνάρτηση θα δέχεται τις πλευρές ενός ορθογωνίου και αφού υπολογίζει, θα επιστρέφει το εμβαδόν του ορθογωνίου.
- **float emvado(float b1, float b2, float y)**
  Η συνάρτηση θα δέχεται τις δύο βάσεις και το ύψος ενός τραπεζίου και αφού υπολογίζει, θα επιστρέφει το εμβαδόν του τραπεζίου.

ΤΕΙ ΣΕΡΡΩΝ
ΣΤΕΦ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΕΠΙΚΟΙΝΩΝΙΩΝ
Αντικειμενοστραφής προγραμματισμός

Εργαστήριο 8

(***Δημόσια κληρονομικότητα***)

1. Να μελετηθεί το παράδειγμα του φυλλαδίου που αναφέρεται στην κληρονομικότητα.
2. Να αναπτυχθεί ένα πρόγραμμα όπου θα ορίζεται μία βασική κλάση Student με τα εξής δεδομένα-μέλη:

```
int am;
char name[20];
char dept[30];
```

και τις εξής συναρτήσεις-μέλη:

- **void readData()**
  Η συνάρτηση θα ζητά - από το χρήστη- να εισαχθούν τα δεδομένα ενός σπουδαστή.
- **void printData()**
  Η συνάρτηση θα εμφανίζει τα δεδομένα κάποιου σπουδαστή.
- **int nameLength()**
  Η συνάρτηση θα βρίσκει και θα επιστρέφει το μήκος – σε χαρακτήρες – του ονόματος ενός σπουδαστή.

Επίσης, θα ορίζεται μία παράγωγη κλάση GradeReport, που θα κληρονομεί από τη Student, με τα εξής δεδομένα-μέλη:

```
char course[20];
float grade;
```

και τις εξής συναρτήσεις-μέλη:

- **void readData()**
  Η συνάρτηση θα καλεί πρώτα τη readData() της βασικής κλάσης και κατόπιν θα ζητά - από το χρήστη- να εισαχθούν τα δεδομένα για το μάθημα και το βαθμό.
- **void printData()**
  Η συνάρτηση θα καλεί πρώτα την printData() της βασικής κλάσης και κατόπιν θα εμφανίζει του μαθήματος και του βαθμού.

Να τονισθεί ότι θα πρέπει να γραφούν κατάλληλες συναρτήσεις εγκατάστασης (*constructors*) χωρίς ορίσματα και με ορίσματα και για τις δύο κλάσεις. Επιπλέον, οι συναρτήσεις εγκατάστασης της παράγωγης κλάσης θα καλούν τις αντίστοιχες συναρτήσεις εγκατάστασης της βασικής κλάσης.

Τέλος, να δηλωθούν στη main() αντικείμενα και για τις δύο κλάσεις και να γίνει προσπέλαση των αντίστοιχων συναρτήσεων-μελών τους. Επίσης, να γίνει προσπέλαση – με ένα αντικείμενο της παράγωγης κλάσης – της συνάρτησης nameLength(), η οποία είναι συνάρτηση-μέλος της βασικής κλάσης.

Δημόσια κληρονομικότητα

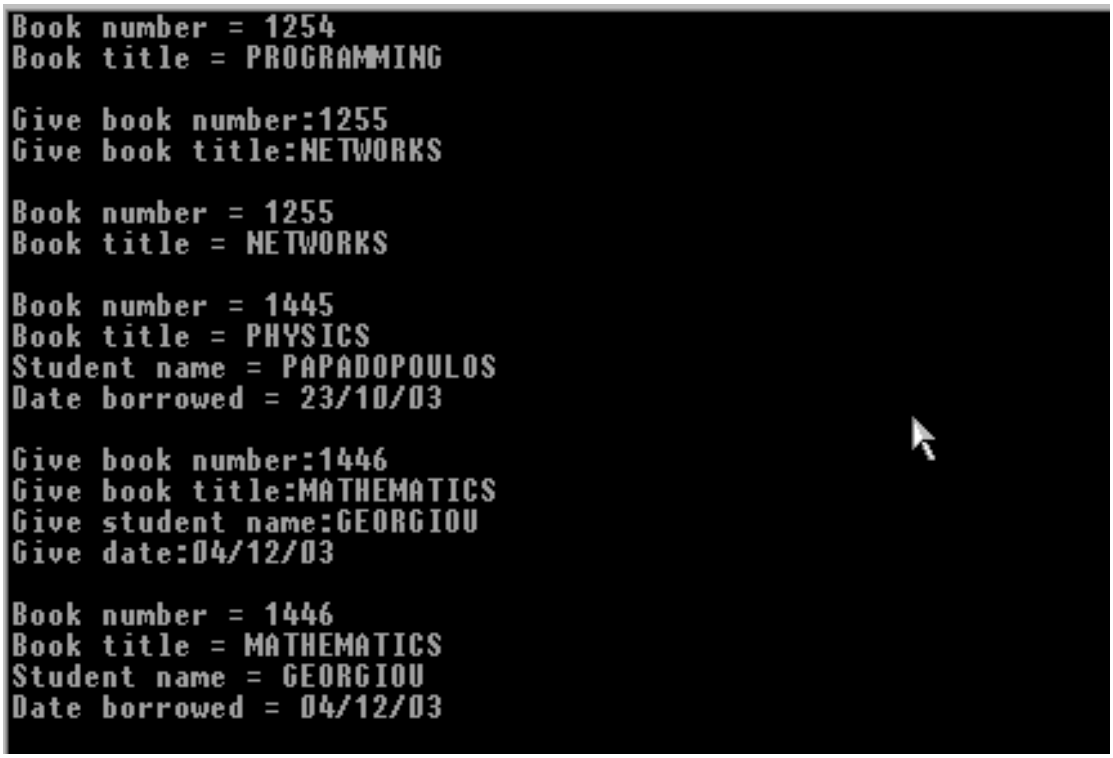
Η κληρονομικότητα είναι ένα από τα πιο ισχυρά χαρακτηριστικά του αντικειμενοστραφούς προγραμματισμού. Είναι ο μηχανισμός που επιτρέπει σε μία κλάση να κληρονομεί όλη τη συμπεριφορά και τις ιδιότητες μιας άλλης κλάσης. Η κλάση που κληρονομεί ονομάζεται **παράγωγη κλάση** (*derived class*), ενώ η κλάση που παρέχει την κληρονομικότητα ονομάζεται **βασική κλάση** (*base class*).

Στο παρακάτω πρόγραμμα ορίζεται μία βασική κλάση Book και μία παράγωγη κλάση Borrow που κληρονομεί την Book. Ακολούθως, δηλώνονται αντικείμενα και για τις δύο κλάσεις και γίνεται προσπέλαση των συναρτήσεών τους.

```
#include <iostream.h>
#include <stdio.h>
#include <conio.h>
#include <string.h>

class Book
{
   private:
      int number;
      char title[30];
   public:
      Book()
      {
         number = 0;
         strcpy(title, " ");
      }
      Book(int number0, char title0[])
      {
         number = number0;
         strcpy(title, title0);
      }
      void readData()
      {
         cout << "Give book number:";
         cin >> number;
         cout << "Give book title:";
         cin >> title;
      }
      void printData()
      {
         cout << "Book number = " << number << endl;
         cout << "Book title = " << title << endl;
      }
};
```

```
class Borrow : public Book    // Δημόσια κληρονομικότητα
{
   private:
      char studname[20];
      char bdate[10];
   public:
      Borrow():Book()
      {
         strcpy(studname, " ");
         strcpy(bdate, " ");
      }
      Borrow(int number1, char title1[], char studname1[], char bdate1[])
           :Book(number1, title1)
      {
         strcpy(studname, studname1);
         strcpy(bdate, bdate1);
      }
      void readData()
      {
         Book::readData();
         cout << "Give student name:";
         cin >> studname;
         cout << "Give date:";
         cin >> bdate;
      }
      void printData()
      {
         Book::printData();
         cout << "Student name = " << studname << endl;
         cout << "Date borrowed = " << bdate << endl;
      }
};
void main()
{
   Book bk1(1254, "PROGRAMMING"), bk2;
   Borrow br1(1445, "PHYSICS", "PAPADOPOULOS", "23/10/03"), br2;

   bk1.printData();
   cout << endl;
   bk2.readData();
   cout << endl;
   bk2.printData();
   cout << endl;
   br1.printData();
   cout << endl;
   br2.readData();
   cout << endl;
   br2.printData();
   getch();
}
```

Έξοδος προγράμματος



Συμπληρωματικές ασκήσεις

Άσκηση 1

Να γραφεί πρόγραμμα όπου θα ορίζεται η παρακάτω κλάση:

```
class  Employee
{
   int  am;
   char  name[20];
   int  categ;
}
```

Το δεδομένο categ δέχεται μόνον τις τιμές 1 και 2. Αν έχει την τιμή 1, τότε ο εργαζόμενος φορολογείται με 15%, ενώ αν έχει την τιμή 2 φορολογείται με 30%.

Να γραφούν κατάλληλες συναρτήσεις εγκατάστασης χωρίς ορίσματα και με ορίσματα, καθώς επίσης και συναρτήσεις για το διάβασμα και την εμφάνιση των τιμών των δεδομένων.

Επίσης, να ορίζεται μία κλάση EmpWork, που θα κληρονομεί δημόσια την προηγούμενη και θα έχει τα εξής δεδομένα μέλη:

```
class  EmpWork
{
   int  hours;
   float  payrate;
}
```

Ακολούθως, να γραφούν κατάλληλες συναρτήσεις εγκατάστασης, συναρτήσεις για το διάβασμα και την εμφάνιση των δεδομένων, καθώς και συναρτήσεις για τον υπολογισμό των μικτών αποδοχών, του φόρου και των καθαρών αποδοχών ενός εργαζομένου.

Άσκηση 2

Να γραφεί πρόγραμμα όπου θα ορίζεται η παρακάτω κλάση:

```
class  Car
{
   char  number[7];
   char  type[10];
   float  daycharge;
}
```

Να γραφούν κατάλληλες συναρτήσεις εγκατάστασης χωρίς ορίσματα και με ορίσματα, καθώς επίσης και συναρτήσεις για το διάβασμα και την εμφάνιση των τιμών των δεδομένων.

Επίσης, να ορίζεται μία κλάση Hire, που θα κληρονομεί δημόσια την προηγούμενη και θα έχει τα εξής δεδομένα μέλη:

```
class  Hire
{
   int  id;
   char  name[20];
   int  days;
}
```

Ακολούθως, να γραφούν κατάλληλες συναρτήσεις εγκατάστασης, συναρτήσεις για το διάβασμα και την εμφάνιση των δεδομένων, καθώς και μία συνάρτηση για τον υπολογισμό της χρέωσης κάποιου πελάτη, ο οποίος ενοικίασε ένα συγκεκριμένο αυτοκίνητο.

Εργαστήριο 9

(***Περιεκτικότητα***)

Να μελετηθεί το παράδειγμα που αναφέρεται στην περιεκτικότητα.

Να τροποποιηθεί το προηγούμενο πρόγραμμα έτσι ώστε κάποιο αντικείμενο της κλάσης GradeReport να περιέχεται στην κλάση Student. Ποιο συγκεκριμένα, τα δεδομένα-μέλη της κλάσης Student να ορίζονται ως εξής:

```
int am;
char name[20];
char dept[30];
GradeReport gr[4];   //περιεκτικότητα
```

Δηλαδή, για κάθε σπουδαστή, δηλώνονται ο αριθμός μητρώου, το όνομά του, το τμήμα που ανήκει και ένας πίνακας 4 αντικειμένων της κλάσης GradeReport, που αντιστοιχεί σε τέσσερα μαθήματα που παρακολουθεί ο σπουδαστής.

Να γραφούν κατάλληλες συναρτήσεις εγκατάστασης (*constructors*) χωρίς ορίσματα και με ορίσματα και για τις δύο κλάσεις.

Να γραφεί μία επιπλέον συνάρτηση για την κλάση Student:

- **float calcAvg()**
  Η συνάρτηση θα υπολογίζει και θα επιστρέφει το μέσο όρο βαθμολογίας του σπουδαστή.

Επίσης να γραφεί για την κλάση GradeReport η συνάρτηση:

- **float getGrade()**
  Η συνάρτηση θα επιστρέφει το βαθμό για κάποιο μάθημα.

Τέλος, να δηλωθούν στη main() δύο αντικείμενα της κλάσης Student, όπου το ένα αντικείμενο θα δέχεται τιμές στη δήλωση, ενώ το άλλο με κλήση της συνάρτησης readData(). Επίσης, να εμφανισθούν οι πληροφορίες και ο μέσος όρος βαθμολογίας και για τα δύο αντικείμενα.

Περιεκτικότητα

Η κληρονομικότητα μας δίνει τη δυνατότητα να ορίσουμε μία σχέση ανάμεσα σε δύο κλάσεις A και B. Μία άλλη σχέση που μπορεί να ορισθεί λέγεται **περιεκτικότητα**. Στο παρακάτω παράδειγμα έχουμε την περίπτωση όπου ένα αντικείμενο της κλάσης B περιέχεται μέσα στην κλάση A.

```
class B
{};
class A
{
  B  b;
};
```

Στο παρακάτω πρόγραμμα ορίζεται η προηγούμενη κλάση Borrow ακολουθούμενη από την κλάση Book. Μέσα στην κλάση Book ορίζεται ως δεδομένο-μέλος ένα αντικείμενο της κλάσης Borrow.

```
#include <iostream.h>
#include <stdio.h>
#include <conio.h>
#include <string.h>
class Borrow
{
  private:
    char studname[20];
    char bdate[10];
  public:
    Borrow()
    {
      strcpy(studname, " ");
      strcpy(bdate, " ");
    }
    Borrow(char studname1[], char bdate1[])
    {
      strcpy(studname, studname1);
      strcpy(bdate, bdate1);
    }
    void readData()
    {
      cout << "Give student name:";
      cin >> studname;
      cout << "Give date:";
      cin >> bdate;
    }
    void printData()
    {
```

```
        cout << "Student name: " << studname << endl;
        cout << "Date borrowed: " << bdate << endl;
    }
};
class Book
{
  private:
    int number;
    char title[30];
    Borrow b;       // Περιεκτικότητα
  public:
    Book()
    {
      number = 0;
      strcpy(title, " ");
    }
    Book(int number0, char title0[], Borrow b0)
    {
      number = number0;
      strcpy(title, title0);
      b = b0;
    }
    void readData()
    {
      cout << "Give book number:";
      cin >> number;
      cout << "Give book title:";
      cin >> title;
      b.readData();
    }
    void printData()
    {
      cout << "Book number: " << number << endl;
      cout << "Book title: " << title << endl;
      b.printData();
    }
};
void main()
{
  Book bk1(1445, "PHYSICS", Borrow("PAPADOPOULOS", "23/10/03")),
       bk2;

  bk1.printData();
  cout << endl;
  bk2.readData();
  cout << endl;
  bk2.printData();

  getch();
}
```

Έξοδος προγράμματος

Συμπληρωματικές ασκήσεις

Άσκηση 1

Να γραφεί πρόγραμμα όπου θα ορίζεται η σχέση της περιεκτικότητας ανάμεσα στις κλάσεις Employee και EmpWork, προβλήματος του προηγούμενου κεφαλαίου, ως εξής:

```
class  EmpWork
{
   int  hours;
   float  payrate;
};

class  Employee
{
   int  am;
   char  name[20];
   int  categ;
   EmpWork  emp;
};
```

Ακολούθως, να γραφούν κατάλληλες συναρτήσεις εγκατάστασης, συναρτήσεις για το διάβασμα και την εμφάνιση των δεδομένων, καθώς και συναρτήσεις για τον υπολογισμό των μικτών αποδοχών, του φόρου και των καθαρών αποδοχών ενός εργαζομένου.

Άσκηση 2

Να γραφεί πρόγραμμα όπου θα ορίζεται η σχέση της περιεκτικότητας ανάμεσα στις κλάσεις Car και Hire, προβλήματος του προηγούμενου κεφαλαίου, ως εξής:

```
class  Hire
{
   int  id;
   char  name[20];
   int  days;
};

class  Car
{
   char  number[7];
   char  type[10];
   float  daycharge;
   Hire   c;
};
```

Ακολούθως, να γραφούν κατάλληλες συναρτήσεις εγκατάστασης, συναρτήσεις για το διάβασμα και την εμφάνιση των δεδομένων, καθώς και μία συνάρτηση για τον υπολογισμό της χρέωσης κάποιου πελάτη, ο οποίος ενοικίασε ένα συγκεκριμένο αυτοκίνητο.

Εργαστήριο 10

(***Αντικείμενα σε αρχεία***)

Να γραφεί πρόγραμμα όπου θα ορίζεται μία κλάση Student με τα εξής δεδομένα-μέλη:

```
int am;
char name[20];
float grade;
```

Επίσης, να γραφούν οι εξής συναρτήσεις-μέλη της κλάσης Student:

- **void setData(int am1, char name1[], float grade1)**
  Η συνάρτηση θα δέχεται τις πληροφορίες κάποιου σπουδαστή και θα τις καταχωρεί στα δεδομένα του αντικειμένου που η συνάρτηση είναι μέλος.
- **void writeStudents(FILE *sf)**
  Η συνάρτηση θα διαβάζει από το πληκτρολόγιο τις πληροφορίες διαφόρων σπουδαστών – μέχρι να δοθεί 0 στον am – και αφού τις καταχωρεί σε αντικείμενα, καλώντας την προηγούμενη συνάρτηση setData(), θα αποθηκεύει τα αντικείμενα σε ένα αρχείο δίσκου.

Επίσης, να γραφεί ένα δεύτερο πρόγραμμα, όπου θα ορίζεται η ίδια κλάση Student και οι εξής συναρτήσεις-μέλη:

- **void printData()**
  Η συνάρτηση θα εμφανίζει τα δεδομένα του αντικειμένου, που είναι μέλος,, στην οθόνη.
- **void readStudents(FILE *sf)**
  Η συνάρτηση θα διαβάζει από το αρχείο τις πληροφορίες διαφόρων σπουδαστών, και καλώντας την προηγούμενη συνάρτηση printData(), θα τις εμφανίζει στην οθόνη.

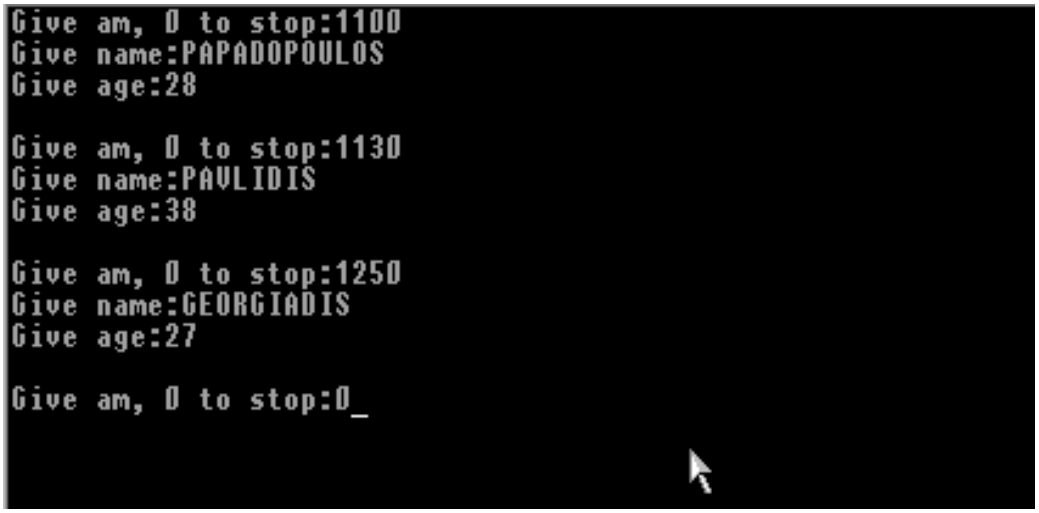
Αντικείμενα σε αρχεία

Τα δύο παρακάτω προγράμματα χρησιμοποιούνται για τη δημιουργία και το διάβασμα ενός αρχείου αντικειμένων στο δίσκο.

1) *Γράψιμο αντικειμένων σε αρχείο*

```
#include <iostream.h>
#include <fstream.h>
#include <stdio.h>
#include <conio.h>
class Person
{
  protected:
      int am;
      char name[20];
      int  age;
  public:
      void setData(int am1, char name1[], int age1)
      {
         am = am1;
         strcpy(name, name1);
         age = age1;
      }
};
void main()
{
   Person p;
   FILE *outfile;
   int armit;
   char onoma[20];
   int  ilikia;

   outfile = fopen("person.dat", “wb”);
   cout << "Give am, 0 to stop:";
   cin >> armit;
   while (armit != 0)
   {
      cout << "Give name:";
      cin >> onoma;
      cout << "Give age:";
      cin >> ilikia;
      p.setData(armit, onoma, ilikia);
      fwrite(&p, sizeof(p), 1, outfile);
      cout << endl << "Give am, 0 to stop:";
      cin >> armit;
   }
   fclose(outfile);
}
```

Έξοδος προγράμματος



Στο πρόγραμμα αυτό πληκτρολογούνται δεδομένα –αριθμός μητρώου, όνομα και ηλικία- για διάφορα άτομα, μέχρι να δοθεί αριθμός μητρώου 0 και αφού τοποθετηθούν σε αντικείμενα με τη συνάρτηση setData(), ακολούθως καταχωρούνται σε ένα αρχείο δίσκου με τη συνάρτηση fwrite().

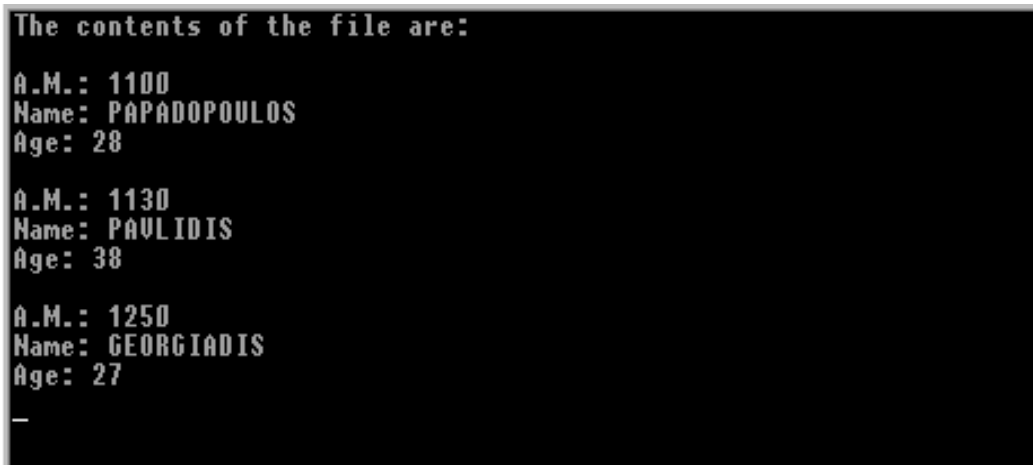
2) *Διάβασμα αντικειμένων από αρχείο*

```
#include <iostream.h>
#include <fstream.h>
#include <stdio.h>
#include <conio.h>
class Person
{
  private:
      int am;
      char name[20];
      int age;
  public:
      void printData()
      {
         cout << "A.M.: " << am << endl;
         cout << "Name: " << name << endl;
         cout << "Age: " << age << endl;
      }
};
void main()
{
   Person p;
   FILE *infile;

   infile = fopen("person.dat", “rb”);
   cout << "The contents of the file are:" << endl;
```

```
   fread(&p, sizeof(p), 1, infile);
   while (!feof(infile))
   {
      cout << endl;
      p.printData();
      fread (&p, sizeof(p), 1, infile)
   }
   getch();
   fclose(infile);
}
```

Στο πρόγραμμα αυτό διαβάζονται οι πληροφορίες των ατόμων από το προηγούμενο αρχείο, με χρήση της συνάρτησης fread(), ένα αντικείμενο τη φορά, και εμφανίζονται στην οθόνη με τη συνάρτηση printData().

Έξοδος προγράμματος



Συμπληρωματικές ασκήσεις

Άσκηση 1

Να γραφεί πρόγραμμα όπου θα ορίζεται μία κλάση Ergazomenos με τα εξής δεδομένα-μέλη:

```
int am;
char name[20];
float payrate;
```

και οι εξής συναρτήσεις-μέλη:

- **void readEmployees(FILE *f)**
  *Η συνάρτηση θα διαβάζει από το πληκτρολόγιο τις πληροφορίες διαφόρων εργαζομένων – μέχρι να δοθεί 0 στον am – και αφού τις καταχωρεί σε αντικείμενα, θα αποθηκεύει τα αντικείμενα σε ένα αρχείο δίσκου, με το όνομα **employee.dat**.*

- **void displayEmployees(FILE *f)**
  Η συνάρτηση θα διαβάζει από το αρχείο τις πληροφορίες διαφόρων εργαζομένων, θα τις μεταφέρει προσωρινά σε αντικείμενα και κατόπιν θα τις εμφανίζει στην οθόνη.

Άσκηση 2

Να γραφεί πρόγραμμα όπου θα ορίζεται μία κλάση Proion με τα εξής δεδομένα-μέλη:

```
int code;
char descriptio[30];
float unitprice;
```

και οι εξής συναρτήσεις-μέλη:

- **void readItems(FILE *f)**
  Η συνάρτηση θα διαβάζει από το πληκτρολόγιο τις πληροφορίες διαφόρων προϊόντων – μέχρι να δοθεί 0 στον κωδικό προϊόντος – και αφού τις καταχωρεί σε αντικείμενα, θα αποθηκεύει τα αντικείμενα σε ένα αρχείο δίσκου, με το όνομα ***items.dat****.*
- **void displayItems(FILE *f)**
  Η συνάρτηση θα διαβάζει από το αρχείο τις πληροφορίες διαφόρων προϊόντων, θα τις μεταφέρει προσωρινά σε αντικείμενα και κατόπιν θα τις εμφανίζει στην οθόνη.

ΤΕΙ ΣΕΡΡΩΝ
ΣΤΕΦ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΕΠΙΚΟΙΝΩΝΙΩΝ
Αντικειμενοστραφής προγραμματισμός

Εργαστήριο 11

(***Αντικείμενα σε αρχεία***)

Να γραφεί πρόγραμμα όπου θα ορίζεται η προηγούμενη κλάση Student και το αρχείο σπουδαστών που δημιουργήθηκε.

Να γραφούν οι εξής επιπλέον συναρτήσεις-μέλη της κλάσης Student:

- **bool searchFile(FILE *sf, float arm)**
  Η συνάρτηση θα δέχεται έναν αριθμό μητρώου και θα τον αναζητά στο αρχείο σπουδαστών. Αν τον βρίσκει, θα επιστρέφει μία λογική τιμή true, αλλιώς θα επιστρέφει την τιμή false.

- **void updateGrade(FILE *sf)**
  Η συνάρτηση θα διαβάζει από το πληκτρολόγιο τον αριθμό μητρώου ενός σπουδαστή και καλώντας τη συνάρτηση searchFile() θα τον αναζητεί στο αρχείο. Αν η αναζήτηση είναι επιτυχής, τότε θα ζητείται από το χρήστη να πληκτρολογήσει ένα νέο βαθμό και ο νέος αυτός βαθμός θα καταχωρείται στο αρχείο στη θέση του παλιού βαθμού του συγκεκριμένου σπουδαστή.

Στο τέλος, το πρόγραμμα να εμφανίζει τα περιεχόμενα του αρχείου –με κλήση της συνάρτησης readStudents() – για να γίνεται επιβεβαίωση της ενημέρωσης.

Αντικείμενα σε αρχεία - Ενημέρωση αντικειμένου

Στο πρόγραμμα που ακολουθεί γίνεται αναφορά και επεξεργασία του αρχείου ***person.dat*** που δημιουργήθηκε στο προηγούμενο εργαστήριο.

```
#include <iostream.h>
#include <fstream.h>
#include <stdio.h>
#include <conio.h>
class Person
{
  private:
      int am;
      char name[20];
      int age;
  public:
      void printData()
      {
         cout << "A.M.: " << am << endl;
         cout << "Name: " << name << endl;
         cout << "Age: " << age << endl;
      }
      void updateAge(int age1)
      {
          age = age1;
      }
      bool  equalAm(int am1)
      {
          if (am == am1)
              return  true;
          else
              return false;
      }
};

void main()
{
   Person  p;
   FILE *f;
   int armit;
   int  ilikia;
   bool found;
   int lenObj;

   f = fopen("person.dat", "rb+");
```

```
  cout << "Give person am for update:";
  cin >> armit;
  found = false;
  fread(&p, sizeof(p), 1, f);
  while (!feof(f) && found == false)
        {
           if (p.equalAm(armit))
              found = true;
           else
              fread(&p, sizeof(p), 1, f);
        }
  if (found)
     {
          cout << “The person was found:” << endl;
          p.printData();
          cout << endl << "Give new person age:";
          cin >> ilikia;
          p.updateAge(ilikia) ;
          lenObj = sizeof(p);
          fseek(f, -lenObj, 1);
          fwrite(&p, sizeof(p), 1, f);
     }
   else
     {
          cout << "No such person." << endl;
          getch();
     }
   fclose(f);
}
```

Στο παραπάνω πρόγραμμα γίνεται ενημέρωση του αρχείου με πληροφορίες ατόμων. Πιο συγκεκριμένα, δίνεται ο αριθμός μητρώου κάποιου ατόμου από το πληκτρολόγιο και αναζητείται στο αρχείο. Η προσπέλαση του αρχείου γίνεται με την εντολή fread(), η οποία διαβάζει ένα αντικείμενο τη φορά.

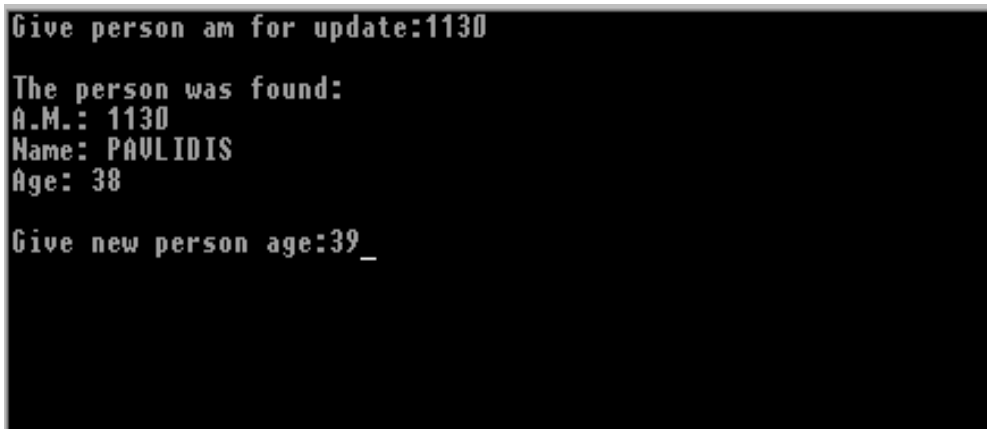
Αν ο αριθμός μητρώου βρεθεί, τότε ζητείται να πληκτρολογηθεί η νέα ηλικία του ατόμου. Καλώντας τη συνάρτηση updateAge() γίνεται τροποποίηση της ηλικίας του τρέχοντος αντικειμένου, δηλ. του ατόμου που μόλις διαβάστηκε από το αρχείο. Επειδή όμως ο δείκτης του αρχείου είναι τοποθετημένος στο επόμενο αντικείμενο, πρέπει να τον επανατοποθετήσουμε μία θέση πίσω και μετά να αποθηκεύσουμε εκ νέου το τρέχον αντικείμενο. Αυτό επιτυγχάνεται με τις παρακάτω εντολές:

lenObj = sizeof(p); *//υπολογίζει το μέγεθος του τρέχοντος αντικειμένου σε bytes*

fseek(f, -lenObj, 1); *//μετακινεί το δείκτη του αρχείου τόσα bytes από την τρέχουσα* θέση

fwrite(&p, sizeof(p), 1, f); *//γράφει το αντικείμενο στο αρχείο*

Έξοδος προγράμματος



Συμπληρωματικές ασκήσεις

Άσκηση 1

Να γραφεί πρόγραμμα όπου θα ορίζεται η κλάση Ergazomenos προηγούμενου εργαστηρίου και το αρχείο εργαζομένων που δημιουργήθηκε.

Επίσης, να γραφεί η εξής επιπλέον συνάρτηση-μέλος:

- **void searchEmployee(FILE *f, float arm)**
  Η συνάρτηση θα δέχεται έναν αριθμό μητρώου και θα τον αναζητά στο αρχείο εργαζομένων. Αν τον βρίσκει, τότε θα ζητείται από το χρήστη να πληκτρολογήσει ένα νέο ωρομίσθιο και θα το καταχωρεί στη θέση του παλιού.

Άσκηση 2

Να γραφεί πρόγραμμα όπου θα ορίζεται η κλάση Proion προηγούμενου εργαστηρίου και το αρχείο προϊόντων που δημιουργήθηκε.

Επίσης, να γραφεί η εξής επιπλέον συνάρτηση-μέλος:

- **void searchItem(FILE *f, float kodikos)**
  Η συνάρτηση θα δέχεται έναν κωδικό προϊόντος μητρώου και θα τον αναζητά στο αρχείο με τα προϊόντα. Αν τον βρίσκει, τότε θα ζητείται από το χρήστη να πληκτρολογήσει μία νέα τιμή μονάδος και θα την καταχωρεί στη θέση της παλιάς.

Σημ.:

Στο τέλος, και για τις δύο ασκήσεις, τα προγράμματα να εμφανίζουν τα περιεχόμενα των αρχείων, για να γίνεται επιβεβαίωση των ενημερώσεων.

ΤΕΙ ΣΕΡΡΩΝ
ΣΤΕΦ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΕΠΙΚΟΙΝΩΝΙΩΝ
Αντικειμενοστραφής προγραμματισμός

Εργαστήριο 12

(***Αντικείμενα σε αρχεία***)

Να γραφεί πρόγραμμα όπου θα ορίζεται η προηγούμενη κλάση Student και το αρχείο σπουδαστών που δημιουργήθηκε.

Να γραφούν οι εξής επιπλέον συναρτήσεις-μέλη της κλάσης Student:

- **void writeStudent(FILE *sf)**
  Η συνάρτηση θα διαβάζει τις πληροφορίες για ένα νέο σπουδαστή και θα τις καταχωρεί ως ένα αντικείμενο στο τέλος του αρχείου.

- **void deleteStudent(FILE *sf)**
  Η συνάρτηση θα διαβάζει από το πληκτρολόγιο τον αριθμό μητρώου ενός σπουδαστή και καλώντας τη συνάρτηση searchFile() θα τον αναζητεί στο αρχείο. Αν η αναζήτηση είναι επιτυχής, τότε θα διαγράφεται από το αρχείο, αλλιώς θα τυπώνεται ανάλογο μήνυμα.

- **void eraseFromFile(FILE *sf, int armit)**
  Η συνάρτηση θα καλείται από τη deleteStudents() και θα φέρνει σε πέρας τη διαγραφή. Η διαγραφή θα υλοποιείται με τη χρήση πίνακα.

Ακολούθως, να τυπώνεται το παρακάτω μενού για τη διαχείριση του προαναφερθέντος αρχείου:

1.Εισαγωγή νέου σπουδαστή

2.Διαγραφή σπουδαστή

3.Ενημέρωση βαθμολογίας σπουδαστή

4.Εμφάνιση αρχείου σπουδαστών

5.Έξοδος

Αντικείμενα σε αρχεία – Εισαγωγή/διαγραφή αντικειμένων

Εισαγωγή αντικειμένου στο τέλος του αρχείου

Στο παρακάτω πρόγραμμα πληκτρολογούνται τα στοιχεία ενός νέου ατόμου και καταχωρούνται σε ένα αντικείμενο με τη συνάρτηση *setData()*. Ακολούθως, μετακινείται ο δείκτης του αρχείου στο τέλος του αρχείου με την εντολή

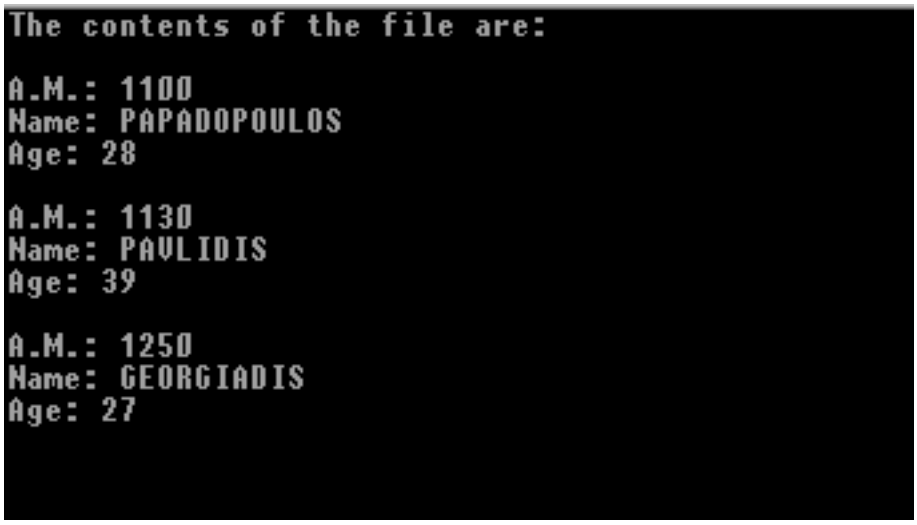
fseek(f, 0, 2);

και τέλος γράφεται το νέο αντικείμενο με την εντολή *fwrite()*.

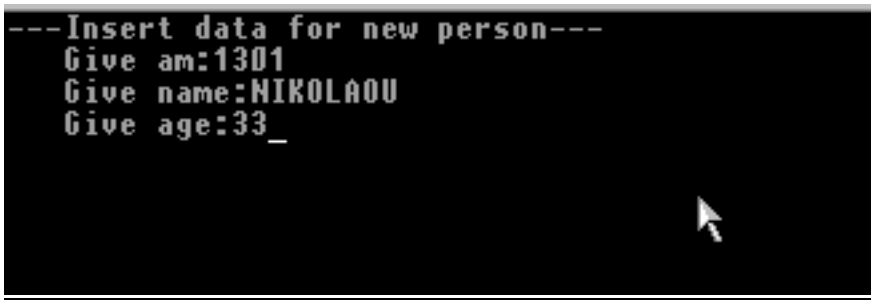
```
#include <iostream.h>
#include <fstream.h>
#include <stdio.h>
#include <conio.h>

class Person
{
  private:
      int am;
      char name[20];
      int age;
  public:
      void setData(int am1, char name1[], int age1)
      {
         am = am1;
         strcpy(name, name1);
         age = age1;
      }
      void printData()
      {
         cout << "A.M.: " << am << endl;
         cout << "Name: " << name << endl;
         cout << "Age: " << age << endl;
      }
 };
void main()
{
   Person  p;
   FILE *f;
   int armit;
   char onoma[20];
   int  ilikia;
```

```
   f = fopen("person.dat", "rb+");
   cout << "---Insert data for new person---" << endl;
   cout << "   Give am:";
   cin >> armit;
   cout << "   Give name:";
   cin >> onoma;
   cout << "   Give age:";
   cin >> ilikia;
   p.setData(armit, onoma, ilikia);

   fseek(f, 0, 2);
   fwrite(&p, sizeof(p), 1, f);
   fclose(f);
}
```

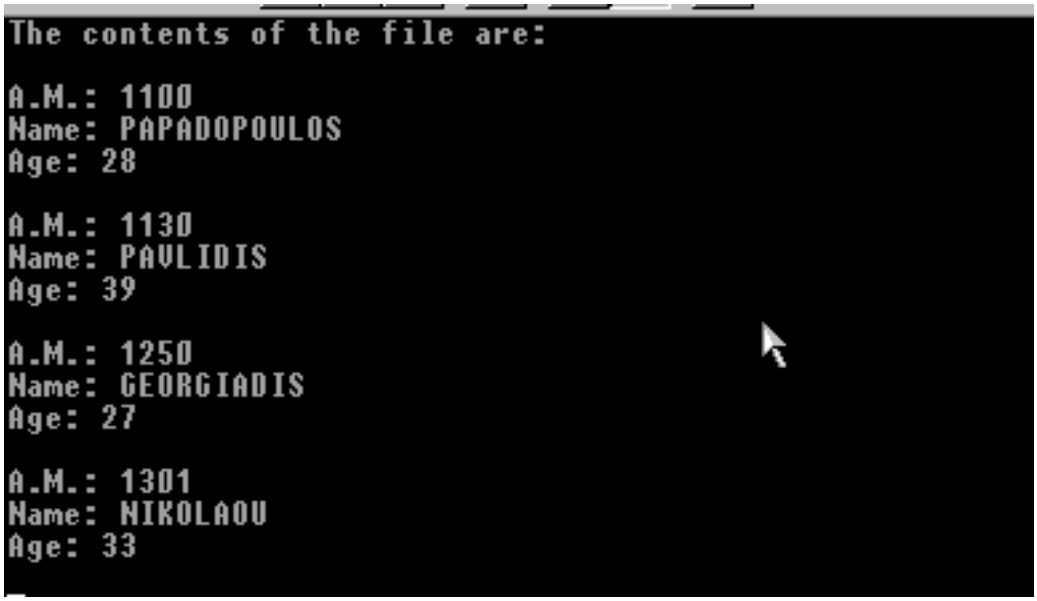
Περιεχόμενα του αρχείου πριν την εισαγωγή:



Έξοδος προγράμματος



Περιεχόμενα του αρχείου μετά την εισαγωγή:



Διαγραφή αντικειμένου από το αρχείο

Στο παρακάτω πρόγραμμα πληκτρολογείται ο αριθμός μητρώου ενός ατόμου και αναζητείται στο αρχείο. Αν βρεθεί, τότε διαγράφεται από το αρχείο. Η διαγραφή υλοποιείται με τη χρήση ενός πίνακα, όπου μετακινούνται – προσωρινά - όλα τα αντικείμενα από το αρχείο, πλην του αντικειμένου που διαγράφεται. Στο τέλος, καταστρέφεται το παλιό αρχείο με την εντολή

f = fopen("person.dat", "wb+");

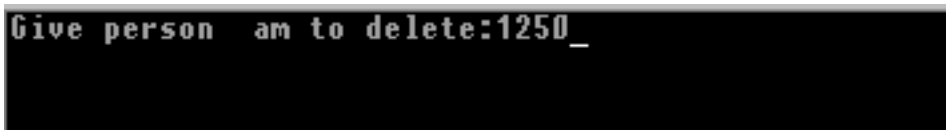
και ξαναδημιουργείται με τα δεδομένα του πίνακα.

```
#include <iostream.h>
#include <fstream.h>
#include <stdio.h>
#include <conio.h>

class Person
{
  private:
     int am;
     char name[20];
     int age;
  public:
     void setData(int am1, char name1[], int age1)
     {
        am = am1;
        strcpy(name, name1);
        age = age1;
     }
```

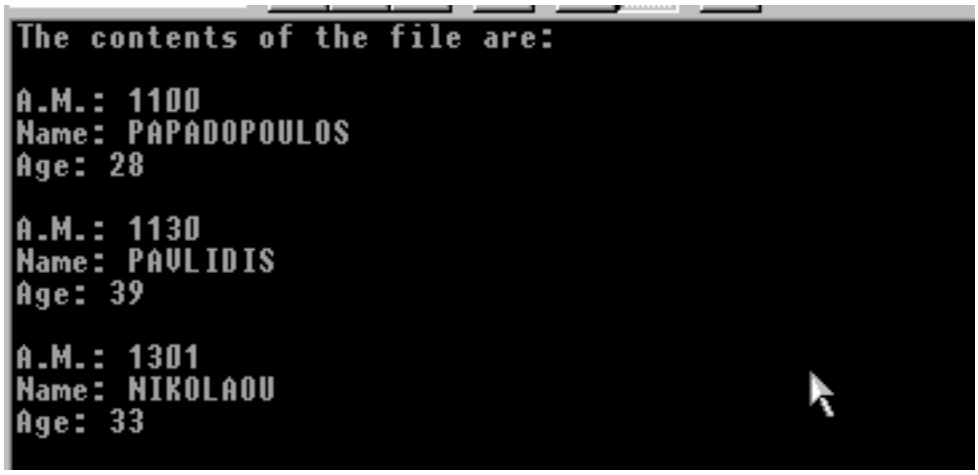
```
    void printData()
    {
       cout << "A.M.: " << am << endl;
       cout << "Name: " << name << endl;
       cout << "Age: " << age << endl;
    }
    bool  equalAm(int am1)
    {
        if (am == am1)
            return  true;
        else
            return false;
    }
};
void main()
{
   Person  p, pin[30];
   FILE *f;
   int armit, i=0, j;
   bool found;

   f = fopen("person.dat", "rb+");
   cout << "Give person  am to delete:";
   cin >> armit;
   found = false;
   fread(&p, sizeof(p), 1, f);
   while (!feof(f) && found == false)
        {
          if (p.equalAm(armit))
             found = true;
          else
             fread(&p, sizeof(p), 1, f);
        }
   if (found)
     {
         rewind(f);
         fread(&p, sizeof(p), 1, f);
         while (!feof(f))
              {
                 if (!p.equalAm(armit))
                     pin[i++] = p;
                 fread(&p, sizeof(p), 1, f);
              }
         fclose(f);
         f = fopen("person.dat", "wb+");
         for (j=0; j<i; j++)
              fwrite(&pin[j], sizeof(pin[j]), 1, f);
    }
```

```
  else
     {
          cout << "No such person." << endl;
          getch();
     }
  fclose(f);
}
```

Έξοδος προγράμματος



Περιεχόμενα του αρχείου μετά τη διαγραφή:



Συμπληρωματικές ασκήσεις

Άσκηση 1

Να γραφεί πρόγραμμα όπου θα γίνεται, μέσω ενός μενού, η διαχείριση του αρχείου εργαζομένων προηγούμενων ασκήσεων.

Άσκηση 2

Να γραφεί πρόγραμμα όπου θα γίνεται, μέσω ενός μενού, η διαχείριση του αρχείου προϊόντων προηγούμενων ασκήσεων.

Σημ.:

Τα μενού θα έχουν τις εξής επιλογές:

1.Εισαγωγή νέου αντικειμένου
2.Διαγραφή αντικειμένου
3.Ενημέρωση αντικειμένου
4.Εμφάνιση αρχείου αντικειμένων
5.Έξοδος

ΤΕΙ ΣΕΡΡΩΝ
ΣΤΕΦ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΕΠΙΚΟΙΝΩΝΙΩΝ
Αντικειμενοστραφής προγραμματισμός

Εργαστήριο 13

(***Αρχεία κεφαλίδων***)

Να γραφεί αρχείο κεφαλίδας με το όνομα "labstack.h" όπου θα ορίζεται μία κλάση με το όνομα myStack και τα δεδομένα-μέλη:

```
char S[N];
int top;
```

Επίσης, θα ορίζονται και οι εξής συναρτήσεις-μέλη:

- **bool stackEmpty()**
  Η συνάρτηση θα ελέγχει αν η στοίβα είναι άδεια και θα επιστρέφει μία ανάλογη λογική τιμή true ή false.
- **bool stackFull()**
  Η συνάρτηση θα ελέγχει αν η στοίβα είναι γεμάτη και θα επιστρέφει μία ανάλογη λογική τιμή true ή false.
- **void push(char x)**
  Η συνάρτηση θα δέχεται ένα χαρακτήρα και θα τον εισάγει στην κορυφή της στοίβας. Αν η στοίβα είναι γεμάτη θα τυπώνει ένα ανάλογο μήνυμα.
- **char pop()**
  Η συνάρτηση θα εξάγει το χαρακτήρα που βρίσκεται στην κορυφή της στοίβας. Αν η στοίβα είναι άδεια θα τυπώνει ένα ανάλογο μήνυμα.

Ακολούθως, να γραφεί πρόγραμμα όπου θα ενσωματώνεται το προηγούμενο αρχείο κεφαλίδας και θα γίνεται έλεγχος σε αριθμητικές παραστάσεις για τη σωστή χρήση παρενθέσεων. Η λειτουργία του προγράμματος θα έχει ως εξής:

Θα ζητείται από το χρήστη να πληκτρολογήσει μία αριθμητική παράσταση η οποία θα περιέχει και παρενθέσεις. Κάθε φορά που θα πληκτρολογείται μία αριστερή παρένθεση, θα τοποθετείται στη στοίβα. Όταν πληκτρολογείται μία δεξιά παρένθεση, θα εξάγεται μία αριστερή παρένθεση από την κορυφή της στοίβας. Το πρόγραμμα θα πρέπει να τυπώνει ένα κατάλληλο μήνυμα, για το αν οι παρενθέσεις χρησιμοποιήθηκαν σωστά ή όχι.

Π.χ

(a+b)*c+a*(d-e)	*Παράσταση σωστή.*
(((a*b)+(c*d)*e)+f)*g	*Παράσταση σωστή.*
a+b*(d-e))	*Παράσταση λάθος. Περισσότερες δεξιές παρενθέσεις.*
((a+b)*c+a*d	*Παράσταση λάθος. Περισσότερες αριστερές παρενθέσεις.*
(a+b)*c)+a*(d-e)	*Παράσταση λάθος. Περισσότερες δεξιές παρενθέσεις.*

Αρχεία κεφαλίδων

Στο παρακάτω αρχείο κεφαλίδας, με το όνομα stoiva.h, ορίζεται μία κλάση StaticStack με τις κατάλληλες συναρτήσεις για την υλοποίηση της δομής της στοίβας με χρήση πίνακα. Ως γνωστόν, η δομή της στοίβας ανήκει στην κατηγορία των δομών δεδομένων LIFO (Last In First Out). Το αρχείο αυτό θα ενσωματωθεί στο επόμενο πρόγραμμα για τη διαχείριση της δομής της στοίβας μέσα από ένα μενού επιλογών.

Stoiva.h

```
#include <iostream.h>
#define N 10
class StaticStack
{
   private:
      int S[N];
      int top;
   public:
      StaticStack()
      {
         top = -1;
      }
      bool push(int x)
      {
         if (top == N-1)
           {
            cout << "Stack full!!!" << endl;
            return false;
           }
         else
           {
            S[++(top)] = x;
            return true;
           }
      }
      bool pop(int *p)
      {
         if (top < 0)
           {
            cout << "Stack empty!!!" << endl;
            return false;
           }
         else
           {
            *p = S[(top)--];
            return true;
           }
      }
```

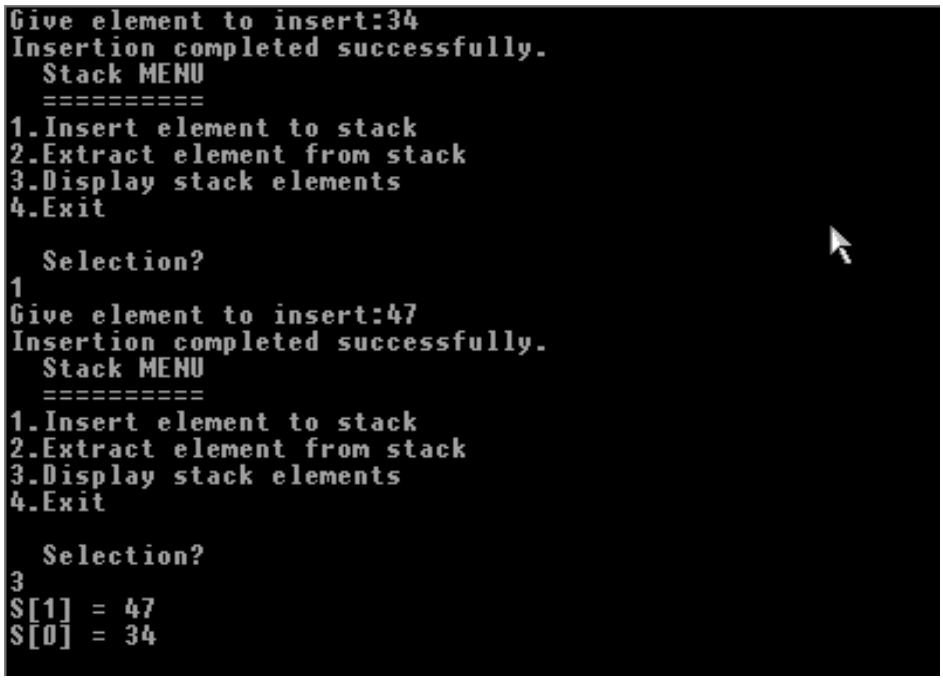
```
    void displayStack()
    {
       int i;
       if (top < 0)
          cout << "Stack empty. No elements to display." << endl;
       else
          for(i=top; i>=0; i--)
             cout << "S[" << i << "] = " << S[i] << endl;
       getch();
    }
};
```

Στο παρακάτω πρόγραμμα γίνεται ενσωμάτωση του προηγούμενου αρχείου κεφαλίδας, για τη διαχείριση της δομής της στοίβας με τη χρήση ενός μενού επιλογών:

```
#include "stoiva.h"    // ενσωμάτωση αρχείου κεφαλίδας
void eisagogi(StaticStack *s1);
void exagogi(StaticStack *s1);

void main()
{
   int epil;
   StaticStack stackObj;

   do
   {
      cout << "  Stack MENU" << endl;
      cout << "  ==========" << endl;
      cout << "1.Insert element to stack" << endl;
      cout << "2.Extract element from stack" << endl;
      cout << "3.Display stack elements" << endl;
      cout << "4.Exit" << endl;
      cout << endl << "  Selection?";
      cin >> epil;
      switch (epil)
      {
         case 1:
              eisagogi(&stackObj);
              break;
         case 2:
              exagogi(&stackObj);
              break;
         case 3:
              stackObj.displayStack();
              break;
      }
   }while (epil!=4);
}
```

```
void eisagogi(StaticStack *s1)
{
   int elem;
   bool flag;

   cout << "Give element to insert:";
   cin >> elem;
   flag = s1->push(elem);
   if (flag==true)
      cout << "Insertion completed successfully." << endl;
   else
      cout << "No more elements can be inserted." << endl;
   getch();
}

void exagogi(StaticStack *s1)
{
   int elem;
   bool flag;

   flag = s1->pop(&elem);
   if (flag==true)
      cout << "Element " << elem << " was successfully extracted." << endl;
   else
      cout << "There are no elements to be extracted." << endl;
   getch();
}
```

Έξοδος προγράμματος



ΤΕΙ ΣΕΡΡΩΝ
ΣΤΕΦ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΕΠΙΚΟΙΝΩΝΙΩΝ
Αντικειμενοστραφής προγραμματισμός

Εργαστήριο 14

(***Αρχεία κεφαλίδων***)

Να γραφεί αρχείο κεφαλίδας με το όνομα "labque.h" όπου θα ορίζεται μία κλάση με το όνομα myQueue και τα δεδομένα-μέλη:

int Q[N];
int front, rear;

Επίσης, θα ορίζονται και οι εξής συναρτήσεις-μέλη:

- **bool queueEmpty()**
  Η συνάρτηση θα ελέγχει αν η ουρά είναι άδεια και θα επιστρέφει μία ανάλογη λογική τιμή true ή false.
- **bool queueFull()**
  Η συνάρτηση θα ελέγχει αν η ουρά είναι γεμάτη και θα επιστρέφει μία ανάλογη λογική τιμή true ή false.
- **void enqueue(int x)**
  Η συνάρτηση θα δέχεται έναν ακέραιο και θα τον εισάγει στο τέλος της ουράς. Αν η ουρά είναι γεμάτη θα τυπώνει ένα ανάλογο μήνυμα.
- **int dequeue()**
  Η συνάρτηση θα εξάγει το πρώτο στοιχείο από την ουρά. Αν η στοίβα είναι άδεια θα τυπώνει ένα ανάλογο μήνυμα.

Ακολούθως, να γραφεί πρόγραμμα όπου θα ενσωματώνεται το προηγούμενο αρχείο κεφαλίδας και θα τυπώνεται ένα μενού διαχείρισης, για την εξυπηρέτηση πελατών που καταφθάνουν σε μία τράπεζα.

1. Άφιξη πελάτη
2. Αναχώρηση πελάτη
3. Κατάσταση ουράς

Κατά την άφιξη ενός πελάτη θα του αποδίδεται ένας αριθμός προτεραιότητας, ο οποίος και θα καταχωρείται στην ουρά. Η υλοποίηση αυτού του αριθμού να γίνει με μία στατική μεταβλητή, που θα έχει αρχική τιμή 1 και θα αυξάνεται αυτόματα κατά την άφιξη ενός πελάτη.

Αρχεία κεφαλίδων

Στο παρακάτω αρχείο κεφαλίδας, με το όνομα oura.h, ορίζεται μία κλάση StaticQueue με τις κατάλληλες συναρτήσεις για την υλοποίηση της δομής της ουράς με χρήση πίνακα. Ως γνωστόν, η δομή της ουράς ανήκει στην κατηγορία των δομών δεδομένων FIFO (First In First Out). Το αρχείο αυτό θα ενσωματωθεί στο επόμενο πρόγραμμα για τη διαχείριση της δομής της ουράς μέσα από ένα μενού επιλογών.

Oura.h

```
#include <iostream.h>
#include <conio.h>
#define N 10
class StaticQueue
{
   private:
      int Q[N];
      int front, rear;
   public:
      StaticQueue()
      {
         front = -1;
         rear = -1;
      }
      bool enqueue(int x)
      {
         if (rear == N-1)
            {
             cout << "Queue full!!!" << endl;
             return false;
            }
         else
            {
             Q[++(rear)] = x;
             return true;
            }
      }
      bool dequeue(int *p)
      {
         if (front == rear)
            {
             cout << "Queue empty!!!" << endl;
             return false;
            }
         else
            {
             *p = Q[++(front)];
             return true;
            }
      }
```

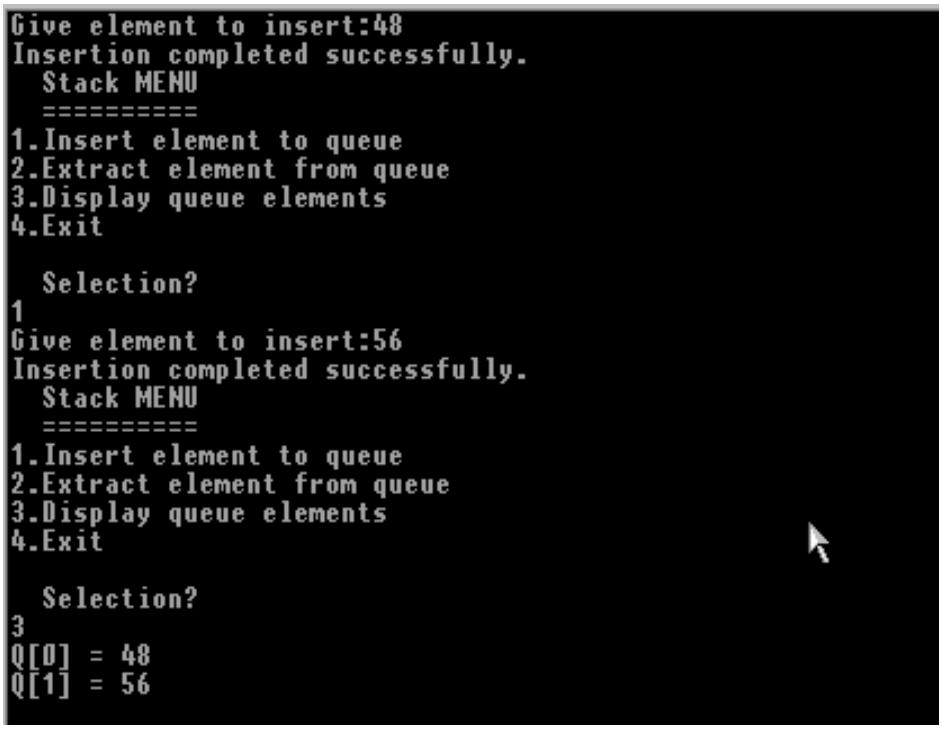
```
    void displayQueue()
    {
       int i;
       if (front == rear)
          cout << "Queue empty. No elements to display." << endl;
       else
          for(i=front+1; i<=rear; i++)
             cout << "Q[" << i << "] = " << Q[i] << endl;
       getch();
    }
};
```

Στο παρακάτω πρόγραμμα γίνεται ενσωμάτωση του προηγούμενου αρχείου κεφαλίδας, για τη διαχείριση της δομής της στοίβας με τη χρήση ενός μενού επιλογών:

```
#include "oura.h"    // ενσωμάτωση αρχείου κεφαλίδας
void eisagogi(StaticQueue *q1);
void exagogi(StaticQueue *q1);

void main()
{
   int epil;
   StaticQueue queObj;

   do
   {
      cout << "  Stack MENU" << endl;
      cout << "  ==========" << endl;
      cout << "1.Insert element to queue" << endl;
      cout << "2.Extract element from queue" << endl;
      cout << "3.Display queue elements" << endl;
      cout << "4.Exit" << endl;
      cout << endl << "  Selection?";
      cin >> epil;
      switch (epil)
      {
         case 1:
              eisagogi(&queObj);
              break;
         case 2:
              exagogi(&queObj);
              break;
         case 3:
              queObj.displayQueue();
              break;
      }
   }while (epil!=4);
}
```

```
void eisagogi(StaticQueue *q1)
{
   int elem;
   bool flag;

   cout << "Give element to insert:";
   cin >> elem;
   flag = q1->enqueue(elem);
   if (flag==true)
      cout << "Insertion completed successfully." << endl;
   else
      cout << "No more elements can be inserted." << endl;
   getch();
}

void exagogi(StaticQueue *q1)
{
   int elem;
   bool flag;

   flag = q1->dequeue(&elem);
   if (flag==true)
      cout << "Element " << elem << " was successfully extracted." << endl;
   else
      cout << "There are no elements to be extracted." << endl;
   getch();
}
```

Έξοδος προγράμματος



Συμπληρωματικές ασκήσεις

Άσκηση 1

Να γραφεί αρχείο κεφαλίδας με το όνομα "prosopiko.h" όπου θα ορίζεται η κλάση με το όνομα Ergazomenos, που περιγράφηκε σε προηγούμενα εργαστήρια.

Επίσης, θα ορίζονται και οι αντίστοιχες συναρτήσεις-μέλη για την:

- ***εισαγωγή***
- ***διαγραφή***
- ***μεταβολή***
- ***εμφάνιση***

αντικειμένων του αρχείου *employee.dat*.

Ακολούθως, να γραφεί πρόγραμμα όπου θα ενσωματώνεται το προηγούμενο αρχείο κεφαλίδας και θα πληκτρολογούνται ο αριθμός μητρώου και οι ώρες εργασίας για κάποιο εργαζόμενο και θα υπολογίζεται – με χρήση συνάρτησης – η αμοιβή του. Η διαδικασία θα επαναλαμβάνεται για πολλούς εργαζόμενους μέχρι να δοθεί 0 στον αριθμό μητρώου.

Άσκηση 2

Να γραφεί αρχείο κεφαλίδας με το όνομα "apothiki.h" όπου θα ορίζεται η κλάση με το όνομα Proion, που περιγράφηκε σε προηγούμενα εργαστήρια.

Επίσης, θα ορίζονται και οι αντίστοιχες συναρτήσεις-μέλη για την:

- ***εισαγωγή***
- ***διαγραφή***
- ***μεταβολή***
- ***εμφάνιση***

αντικειμένων του αρχείου *items.dat*.

Ακολούθως, να γραφεί πρόγραμμα όπου θα ενσωματώνεται το προηγούμενο αρχείο κεφαλίδας και θα πληκτρολογούνται ο κωδικός μητρώου και η ποσότητα για κάποιο προϊόν και θα υπολογίζεται – με χρήση συνάρτησης – η αξία του. Η διαδικασία θα επαναλαμβάνεται για πολλά προϊόντα μέχρι να δοθεί 0 στον κωδικό προϊόντος.