Αντικειμενοστραφής Προγραμματισμός

Ενδεκτικές ασκήσεις-απαντήσεις

Τσούλος Ιωάννης, Επίκουρος Καθηγητής Τμ. Μηχανικών Πληροφορικής Τ.Ε.

Άρτα, Μάιος 2015

Άδειες Χρήσης

Το παρόν εκπαιδευτικό υλικό υπόκειται σε άδειες χρήσης Creative Commons. Για εκπαιδευτικό υλικό, όπως εικόνες, που υπόκειται σε άλλου τύπου άδειας χρήσης, η άδεια χρήσης αναφέρεται ρητώς.

Χρηματοδότηση

Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα. Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο ΤΕΙ Ηπείρου**» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.

Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Πρώτο σετ Ασκήσεων C++

Ιωάννης Γ. Τσούλος

2015

Στο σημερινό σετ ασκήσεων θα παρουσιαστούν μια σειρά από ασκήσεις εισαγωγής στην C++, στις δομές ελέγχου και στους πίνακες της γλώσσας

1 Πρόγραμμα επεξεργασίας μισθοδοσίας

Να γραφεί πρόγραμμα το οποίο θα διαβάζει από το πληκτρολόγιο τους μισθούς N προσώπων, όπου N ακέραιος αριθμός που θα εισάγεται προηγουμένως. Το πρόγραμμα να εμφανίζει τον μέσο όρο της μισθοδοσίας.

```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5          double salary;
6          double avg_salary=0.0;
7          int counter;
8          int N;
9          cout<<"Doste␣N?";
10         cin>>N;
11         for(counter=1;counter<=N;counter=counter+1)
12         {
13                 cout<<"Doste␣mistho␣"<<counter<<"␣";
14                 cin>>salary;
15                 avg_salary=avg_salary+salary;
16         }
17         avg_salary=avg_salary/N;
18         cout<<"Mesos␣misthos:␣"<<avg_salary<<endl;
19         return 0;
20 }
```

Να γίνουν οι ακόλουθες διορθώσεις προσθήκες στο παραπάνω πρόγραμμα (με την σειρά που παρουσιάζεται παρακάτω):

1. Το πρόγραμμα διαβάζει από το πληκτρολόγιο τους μισθούς αλλά δεν κάνει κάποιο έλεγχο για το αν οι μισθοί είναι αρνητικοί. Να γίνει αλλαγή στο

πρόγραμμα ώστε αν κάποιος μισθός είναι αρνητικός δεν θα υπολογίζεται στο άθροισμα.

2. Αφού γίνει η παραπάνω αλλαγή να αλλάξετε το πρόγραμμά σας, ώστε αν κάποιος μισθός είναι αρνητικός να γίνεται επαναληπτικά ανάγνωσή του μέχρι ο χρήστης να εισάγει θετική μισθοδοσία.

3. Η εταιρεία αποφάσισε να δώσει μπόνους 100 ευρώ σε όσους μισθωτούς λαμβάνουν λιγότερα από 1000 ευρώ. Διορθώστε το πρόγραμμά σας, ώστε να γίνεται αύξηση της μισθοδοσίας κατά 100 ευρώ σε όσους λαμβάνουν λιγότερα από 1000 ευρώ.

4. Το πρόγραμμά σας να απαντά στην ερώτηση: πόσοι δεν πήραν αύξηση από την εταιρεία;

2 Πρόγραμμα επιλογής συνάρτησης

Να γραφεί πρόγραμμα το οποίο θα διαθέτει τις ακόλουθες συναρτήσεις:

1. Μια συνάρτηση υπολογισμού της δύναμης X^Y, όπου X και Υ ορίσματα της συνάρτησης.

2. Μια συνάρτηση υπολογισμού του παραγοντικού $X!$

Στην κυρίως συνάρτηση ο χρήστης εισάγει έναν ακέραιο στο διάστημα [0,2] υποχρεωτικά. Αν ο χρήστης δώσει 0 το πρόγραμμα σταματά, διαφορετικά αν δώσει 1 εισάγει πρώτα έναν δεκαδικό αριθμό X και μετά έναν δεκαδικό αριθμό Υ και υπολογίζεται η δύναμη με την βοήθεια της πρώτης συνάρτησης. Αν δώσει τον αριθμό 2 εισάγει ένα ακέραιο X και εμφανίζεται το παραγοντικό του αριθμού X. Στο τέλος το πρόγραμμα εμφανίζει πόσες φορές έγινε κλήση της πρώτης συνάρτησης και πόσες φορές της δεύτερης.

```
1  # include <iostream>
2  using namespace std;
3
4  double power(double x,double y)
5  {
6          double p=1.0;
7          int i;
8          for(i=1;i<=y;i++)
9                  p=p*x;
10         return p;
11 }
12
13 int factorial(int x)
14 {
15         int p=1;
16         int i;
```

```
17          for(i=1;i<=x;i++) p=p*i;
18          return p;
19  }
20
21  int readOption()
22  {
23          int option;
24          do
25          {
26                  cout<<"Dose epilogi ";
27                  cin>>option;
28          }while(option<0 || option>2);
29          return option;
30  }
31
32  int main()
33  {
34          int myoption;
35          double x,y;
36          int ix;
37          double mypower;
38          int myfactorial;
39          int counter1=0,counter2=0;
40          while((myoption=readOption())!=0)
41          {
42                  switch(myoption)
43                  {
44                  case 1:
45                          counter1++;
46                          cout<<"Dose x kai Y ";
47                          cin>>x>>y;
48                          mypower=power(x,y);
49                          cout<<"Dynami = "<<mypower<<endl;
50                          break;
51                  case 2:
52                          cout<<"Dose x ";
53                          cin>>ix;
54                          myfactorial=factorial(ix);
55                          cout<<"Paragontiko = "<<myfactorial<<endl;
56                          counter2++;
57                          break;
58                  }
59          }
60          cout<<"Counter1: "<<counter1<<endl;
61          cout<<"Counter2: "<<counter2<<endl;
62          return 0;
```

```
63 }
```

Να γίνουν οι ακόλουθες διορθώσεις/προσθήκες στο παραπάνω πρόγραμμα:

1. Η συνάρτηση του παραγοντικου να γίνει αναδρομική.
2. Η συνάρτηση της δύναμης να επιστρέφει -1 αν κάποιο από τα Χ ή Υ είναι αρνητικό.

3 Χρήση πινάκων

Να γραφεί πρόγραμμα για την ανάγνωση και αποθήκευση βαθμολογιών για 5 σπουδαστές σε έναν πίνακα δεκαδικών αριθμών. Το πρόγραμμα να εμφανίζει

1. Μέσο όρο στο συγκεκριμένο μάθημα.
2. Πλήθος σπουδαστών που έχουν προβιβάσιμο βαθμό.

```
1  # include <iostream>
2  using namespace std;
3
4  double  readGrade()
5  {
6          double b;
7          do
8          {
9                  cout<<"Dose␣bathmo␣(0-10)␣";
10                 cin>>b;
11         }while(b<0 || b>10);
12         return b;
13 }
14
15 void readLesson(double *x,int n)
16 {
17         int i;
18         for(i=0;i<n;i++)
19                 x[i]=readGrade();
20 }
21
22 double average(double *x,int n)
23 {
24         double s=0.0;
25         int i;
26         for(i=0;i<n;i++)
27                 s=s+x[i];
28         return s/n;
29 }
```

```
30
31 int       passed(double *x,int n)
32 {
33         int count=0;
34         int i;
35         for(i=0;i<n;i++)
36                 if(x[i]>=5) count++;
37         return count;
38 }
39
40 int main()
41 {
42         const int students=5;
43         double lesson[students];
44         double mesos;
45         int perasan;
46         readLesson(lesson,students);
47         mesos=average(lesson,students);
48         perasan=passed(lesson,students);
49         cout<<"Mesos oros: "<<mesos<<endl;
50         cout<<"Perasan    : "<<perasan<<endl;
51         return 0;
52 }
```

Να γίνουν οι ακόλουθες διορθώσεις/προσθήκες στο παραπάνω πρόγραμμα:

1. Να προστεθεί μια ακόμα συνάρτηση για την επιστροφή του αριθμού σπουδαστή (θέση στον πίνακα) με τον μεγαλύτερο βαθμό.

2. Αν θέλουμε να καταγράψουμε τις επιδόσεις των 5 σπουδαστών σε 3 μαθήματα τι αλλαγές χρειάζονται;

Δεύτερο σετ Ασκήσεων C++

Ιωάννης Γ. Τσούλος

2015

Στο σημερινό σετ ασκήσεων θα παρουσιαστούν μια σειρά από ασκήσεις για χρήση κατηγοριών σε προγράμματα.

1 Πρόγραμμα ημερομηνίας

Η επόμενη δήλωση της κατηγορίας date μπορεί να χρησιμοποιηθεί για αναπαράσταση ημερομηνιών:

```
1  #ifndef DATE_H_
2  #define DATE_H_
3
4  class date {
5  private:
6          int day,month,year;
7  public:
8          date();
9          void initDate(int d,int m,int y);
10         bool setDay(int d);
11         bool setMonth(int m);
12         bool setYear(int y);
13         int getDay();
14         int getMonth();
15         int getYear();
16         void printDate();
17         virtual ~date();
18 };
19
20 #endif /* DATE_H_ */
```

Η υλοποίηση της παραπάνω κατηγορίας έχει ως εξής:

```
1 #include <iostream>
2 using namespace std;
3 #include "date.h"
4 date::date()
```

```
5  {
6          day=1;
7          month=1;
8          year=1;
9  }
10
11 void date::initDate(int d,int m,int y)
12 {
13         if(!setDay(d)) day=1;
14         if(!setMonth(m)) month=1;
15         if(!setYear(y)) year=1;
16 }
17
18 bool     date::setDay(int d)
19 {
20         if(d<0 || d>31) return false;
21         day=d;
22         return true;
23 }
24
25 bool     date::setMonth(int m)
26 {
27         if(m<0 || m>12) return false;
28         month=m;
29         return true;
30 }
31
32 bool     date::setYear(int y)
33 {
34         if(year<0) return false;
35         year=y;
36         return true;
37 }
38
39 int              date::getDay()
40 {
41         return day;
42 }
43
44 int      date::getMonth()
45 {
46         return month;
47 }
48
49 int      date::getYear()
50 {
```

```
51          return year;
52  }
53
54  void date::printDate()
55  {
56          cout<<day<<"/"<<month<<"/"<<year<<endl;
57  }
58
59  date::~date()
60  {
61
62  }
```

Ένα απλό πρόγραμμα - οδηγός που αρχικοποιεί δύο αντικείμενα ημερομηνίας και τα εμφανίζει είναι το επόμενο:

```
1  #include "date.h"
2  #include <iostream>
3  using namespace std;
4
5  int main()
6  {
7          date d1;
8          date d2;
9          d1.initDate(10,11,2001);
10         d2.initDate(22,24,-1);
11         d1.printDate();
12         d2.printDate();
13         return 0;
14 }
```

Αφού εκτελέσετε με προσοχή το παραπάνω πρόγραμμα κάνετε τις απαραίτητες τροποποιήσεις ώστε:

1. Τα στοιχεία των ημερομηνιών d1 και d2 να διαβάζονται από το πληκτρολόγιο
2. Να προστεθεί μια ακόμα μέθοδος isLeap() στην κατηγορία date, που να επιστρέφει true αν η το έτος είναι δίσεκτο και false αν δεν είναι
3. Να προστεθεί και να δοκιμαστεί (μέσω της main) μια μέθοδος στην κατηγορία date με όνομα void nextDay(). Αυτή η μέθοδος θα υπολογίζει την επόμενη ημέρα και οι υπολογισμοί της θα αποθηκεύονται στα ιδιωτικά πεδία day, month, year.

2 Πρόγραμμα για σημεία

Η επόμενη κατηγορία δηλώνει ένα σημείο στον χώρο R^2 με συντεταγμένες X και Y.

```
1  #ifndef POINT_H_
2  #define POINT_H_
3
4  class Point {
5  private:
6          double x,y;
7  public:
8          Point();
9          Point(double mx,double my);
10         double getx();
11         double gety();
12         void move(double dx,double dy);
13         void print();
14         virtual ~Point();
15 };
16
17 #endif /* POINT_H_ */
```

Ο κώδικας για την υλοποίηση της κατηγορίας έχει ως ακολούθως:

```
1
2  #include "Point.h"
3  # include <iostream>
4  using namespace std;
5
6  Point::Point()
7  {
8          x=0.0;
9          y=0.0;
10 }
11
12
13 Point::Point(double mx,double my)
14 {
15         x = mx;
16         y = my;
17 }
18
19
20 double  Point::getx()
21 {
22         return x;
23 }
24
25
26 double  Point::gety()
27 {
```

```
28          return y;
29  }
30
31  void    Point::move(double dx,double dy)
32  {
33          x=x+dx;
34          y=y+dy;
35  }
36
37  void    Point::print()
38  {
39          cout<<"("<<x<<","<<y<<")"<<endl;
40  }
41
42  Point::~Point()
43  {
44
45  }
```

Μια ενδεικτική main() συνάρτηση όπου αρχικοποιούμε δύο αντικείμενα Point και τα εμφανίζουμε στην οθόνη είναι η επόμενη. Με βάση τα παραπάνω προσθέστε μια ακόμα μέθοδο στην κατηγορία Point

```
bool equals(Point other)
```

Η μέθοδος θα επιστρέφει true αν το τρέχον σημείο (συντεταγμένες x,y) και το other σημείο έχουν τις ίδιες συντεταγμένες και false σε άλλη περίπτωση.

Τρίτο σετ Ασκήσεων C++

Ιωάννης Γ. Τσούλος

2015

1 Κατηγορία προσώπων

Να γραφεί κατηγορία για την περιγραφή προσώπων. Στα ιδιωτικά πεδία της κατηγορίας πρέπει να περιλαμβάνονται τα ακόλουθα:

1. Όνομα
2. Επίθετο
3. Τηλέφωνο
4. Ηλικία

Στα δημόσια πεδία θα πρέπει να υπάρχουν οι μέθοδοι set και get για τα ιδιωτικά πεδία καθώς και δύο βοηθητικές μέθοδοι:

1. Μια μέθοδος που θα επιστρέφει αληθές αν το πρόσωπο είναι ενήλικο
2. Μια μέθοδος που θα τυπώνει στην οθόνη τα στοιχεία το προσώπου

Η κατηγορία Person έχει ως ακολούθως:

```
1  #ifndef PERSON_H_
2  #define PERSON_H_
3  # include <string>
4  using namespace std;
5
6  class Person {
7  private:
8          string name,lastname,telephone;
9          int age;
10 public:
11         Person();
12         void setName(string n);
13         void setLastname(string l);
14         bool setTelephone(string t);
15         bool setAge(int a);
```

```
16          bool isAdult();
17          void printDetails();
18          virtual ~Person();
19  };
20
21  #endif /* PERSON_H_ */
```

Η υλοποίηση της κατηγορίας είναι:

```
 1  #include "Person.h"
 2  #include <iostream>
 3  using namespace std;
 4
 5  Person::Person()
 6  {
 7          name="";
 8          lastname="";
 9          telephone="2681050500";
10          age=18;
11  }
12
13
14  void    Person::setName(string n)
15  {
16          name = n;
17  }
18
19  void    Person::setLastname(string l)
20  {
21          lastname = l;
22  }
23
24  bool    Person::setTelephone(string t)
25  {
26          int i;
27          //first rule
28          if(t.length()!=10) return false;
29          //second rule
30          if(t[0]=='0') return false;
31          for(i=0;i<t.length();i++)
32          {
33                  //third rule
34                  if(!(t[i]>='0' && t[i]<='9')) return false;
35          }
36          telephone = t;
37          return true;
38  }
```

```
39
40 bool     Person::setAge(int a)
41 {
42         if(age<0) return false;
43         age=a;
44         return true;
45 }
46
47 bool     Person::isAdult()
48 {
49         if(age>=18) return true;
50         else return false;
51 }
52
53 void     Person::printDetails()
54 {
55         cout<<"Person details"<<endl;
56         cout<<"Name:     \t"<<name<<endl;
57         cout<<"Lastname: \t"<<lastname<<endl;
58         cout<<"Telephone:\t"<<telephone<<endl;
59         cout<<"Age:      \t"<<age<<endl;
60 }
61
62 Person::~Person()
63 {
64
65 }
```

Τέλος μια ενδεικτική main για την προηγούμενη κατηγορία έχει ως ακολούθως:

```
1  #include "Person.h"
2  #include <stdlib.h>
3  int main()
4  {
5          Person giannis;
6          giannis.setName("Giannis");
7          giannis.setLastname("Tsoulos");
8          giannis.setTelephone("aaa2111");
9          giannis.setAge(38);
10         giannis.printDetails();
11         system("PAUSE");
12         return 0;
13 }
```

Με βάση τα παραπάνω κάνετε τις επόμενες αλλαγές:

1. Προσθέστε ακόμα ένα ιδιωτικό πεδίο για το email του προσώπου. Δημιουργήστε τις απαραίτητες set και get μεθόδους. Στην μέθοδο setEmail() να γίνει

έλεγχος αν το όρισμα που θα μπει στο πεδίο email είναι έγκυρο email (έχει μόνο ένα σύμβολο @ και αυτό δεν βρίσκεται ούτε στην αρχή ούτε στο τέλος του email).

2. Αλλάξτε την συνάρτηση δημιουργίας της κατηγορίας ώστε να διαβάζει και τα 5 πεδία από το πληκτρολόγιο.

3. Αλλάξτε την main() συνάρτηση ώστε να υπάρχουν τρία διαφορετικά αντικείμενα της κατηγορίας Person. Κάνετε τις αρχικοποιήσεις κάθε αντικειμένου και εμφανίστε τα στοιχεία του αντικειμένου με την μεγαλύτερη ηλικία.

2 Κατηγορία κύκλος

Να γραφεί κατηγορία για την περιγραφή κύκλων με τα ακόλουθα ιδιωτικά πεδία:

1. Κέντρο κύκλου (σημείο X0,Υ0)
2. Ακτίνα κύκλου

Στα δημόσια πεδία να περιλαμβάνονται τα ακόλουθα:

1. Συνάρτηση δημιουργίας
2. Μέθοδος επιστροφής του εμβαδού του κύκλου (πR^2)
3. Μέθοδος επιστροφής της περιμέτρου του κύκλου $(4\pi R)$
4. Μέθοδος κλιμάκωσης του κύκλου. Η μέθοδος αυτή δέχεται σαν όρισμα έναν παράγοντα X και πολλαπλασιάζει την ακτίνα με αυτόν τον παράγοντα.

Η δήλωση της κατηγορίας έχει ως ακολούθως:

```
1  #ifndef CIRCLE_H_
2  #define CIRCLE_H_
3
4  class Circle {
5  private:
6          double x0,y0,radious;
7  public:
8          Circle(double x,double y,double r);
9          void scale(double factor);
10         bool pointIn(double x,double y); //to be implemented
11         double getArea();
12         double getPerimeter();
13         ~Circle();
14 };
15
16 #endif /* CIRCLE_H_ */
```

Η υλοποίηση της κατηγορίας:

```
1  #include "Circle.h"
2  # include <math.h>
3  Circle::Circle(double x,double y,double r)
4  {
5          x0=x;
6          y0=y;
7          radious = fabs(r);
8  }
9
10 void     Circle::scale(double factor)
11 {
12         radious = fabs(factor) * radious;
13 }
14
15 bool     Circle::pointIn(double x,double y)
16 {
17         //to be implemented
18 }
19
20 double   Circle::getArea()
21 {
22         return M_PI * radious * radious;
23 }
24
25 double   Circle::getPerimeter()
26 {
27
28         return 4.0 * M_PI * radious;
29 }
30
31 Circle::~Circle()
32 {
33
34 }
```

Και τέλος μια ενδεικτική main:

```
1  # include "Circle.h"
2  # include <iostream>
3  using namespace std;
4  # include <stdlib.h>
5  int main()
6  {
7          Circle c1(10,10,20);
8          cout<<"C1 Area      "<<c1.getArea()<<endl;
9          cout<<"C1 Perimeter "<<c1.getPerimeter()<<endl;
10         c1.scale(2.0);
```

```
11          cout<<"C1 Area       "<<c1.getArea()<<endl;
12          cout<<"C1 Perimeter "<<c1.getPerimeter()<<endl;
13          system("PAUSE");
14          return 0;
15  }
```

Με βάση τα παραπάνω υλοποιήστε την μέθοδο bool pointIn(double x,double y); έτσι ώστε να επιστρέφει αληθές αν το σημείο x,y είναι μέσα στον κύκλο και ψευδές αλλιώς. Δοκιμάστε την μέθοδό σας στην main, όπου θα διαβάζετε δύο αριθμούς X,Υ και θα ελέγχετε αν αυτό το σημείο (X,Υ) είναι μέσα στον κύκλο C1 που υπάρχει στην main. Υπενθυμίζεται πως ένα σημείο (x, y) είναι μέσα στον κύκλο με κέντρο (x_0, y_0) και ακτίνα R αν ισχύει η σχέση

$$(x - x_0)^2 + (y - y_0)^2 \leq R^2$$

Τέταρτο σετ Ασκήσεων C++

Ιωάννης Γ. Τσούλος

2015

1 Κατηγορία Ημερομηνίας

Να γραφεί κατηγορία για την πλήρη περιγραφή ημερομηνιών. Στα ιδιωτικά πεδία πρέπει να περιλαμβάνονται τα ακόλουθα:

1. Ημέρα
2. Μήνας
3. Έτος
4. Πίνακας ακεραίων με τις μέρες κάθε μήνα.

Στα δημόσια πεδία τα ακόλουθα:

1. Τρεις υπερφορτωμένες συναρτήσεις δημιουργίας
2. Μέθοδοι set και get για ημέρα, μήνα, έτος με έλεγχο τιμών
3. Μια μέθοδος εμφάνισης της ημερομηνίας στην οθόνη
4. Μια μέθοδος που θα αλλάζει την ημερομηνία στην επόμενη ημέρα.
5. Μια μέθοδος που θα δέχεται σαν όρισμα αναφορά σε αντικείμενο Date και θα ελέγχει αν η τρέχουσα ημερομηνία και η ημερομηνία στην αναφορά είναι ίδιες.

Η περιγραφή της κατηγορίας έχει ως ακολούθως:

```
1  #ifndef DATE_H_
2  #define DATE_H_
3
4  class Date {
5  private:
6                  int monthdays[12];
7                  void fillTable();
8                  int day,month,year;
9  public:
10         //constructors
```

```
11          Date();
12          Date(int y);
13          Date(int d,int m,int y);
14
15          //set methods
16          bool setYear(int y);
17          bool setMonth(int m);
18          bool setDay(int d);
19
20          //get methods
21          int       getDay();
22          int       getMonth();
23          int       getYear();
24
25          //utility methods
26          bool      sameDate(Date &other);
27          void      nextDay();
28          void      printDate();
29          virtual ~Date();
30  };
31
32  #endif /* DATE_H_ */
```

Η υλοποίηση της κατηγορίας:

```
1   #include "Date.h"
2   # include <iostream>
3   using namespace std;
4
5   Date::Date()
6   {
7           fillTable();
8           day=1;
9           month=1;
10          year=1;
11  }
12
13  Date::Date(int y)
14  {
15          fillTable();
16          day=1;
17          month=1;
18          setYear(y);
19  }
20
21
22  Date::Date(int d,int m,int y)
```

```
23 {
24         fillTable();
25         setDay(d);
26         setMonth(m);
27         setYear(y);
28 }
29
30 void     Date::fillTable()
31 {
32         //oi meres tou mina gia kathe mina
33         monthdays[0]=31;
34         monthdays[1]=28;
35         monthdays[2]=31;
36         monthdays[3]=30;
37         monthdays[4]=31;
38         monthdays[5]=30;
39         monthdays[6]=31;
40         monthdays[7]=31;
41         monthdays[8]=30;
42         monthdays[9]=31;
43         monthdays[10]=30;
44         monthdays[11]=31;
45 }
46
47 bool Date::setYear(int y)
48 {
49         if(y<0) {year=1;return false;}
50         else
51         {
52                 year=y;
53                 return true;
54         }
55 }
56
57 bool Date::setMonth(int m)
58 {
59         if(m<0 || m>12) {month=1;return false;}
60         else
61         {
62                 month=m;
63                 return true;
64         }
65 }
66
67 bool Date::setDay(int d)
68 {
```

```
69          if(d<0 || d>31 || d>monthdays[month-1]) {day=1;return false;}
70          else
71          {
72                  day=d;
73                  return true;
74          }
75  }
76
77  int       Date::getDay()
78  {
79          return day;
80  }
81
82  int       Date::getMonth()
83  {
84          return month;
85  }
86
87  int       Date::getYear()
88  {
89          return year;
90  }
91
92  void      Date::nextDay()
93  {
94          ++day;
95          if(day>monthdays[month-1])
96          {
97                  month++;
98                  day=1;
99                  if(month>12)
100                 {
101                         month=1;
102                         year++;
103                 }
104         }
105 }
106
107 void      Date::printDate()
108 {
109         cout<<day<<"/"<<month<<"/"<<year<<endl;
110 }
111
112 bool      Date::sameDate(Date &other)
113 {
114         if(day==other.getDay() && month==other.getMonth() && year==other.getY
```

```
115                             return true;
116             else
117                             return false;
118 }
119
120 Date::~Date()
121 {
122
123 }
```

Τέλος μια ενδεικτική main συνάρτηση είναι η επόμενη:

```
 1 #include "Date.h"
 2 #include <iostream>
 3 using namespace std;
 4
 5 int main()
 6 {
 7             Date d1;
 8             Date d2(1987);
 9             Date d3(19,11,1922);
10             d1.printDate();
11             d2.printDate();
12             d3.printDate();
13             return 0;
14 }
```

Με βάση τα παραπάνω κάνετε τις ακόλουθες αλλαγές:

1. Αλλάξτε την υπερφορτωμένη συνάρτηση δημιουργίας Date() ώστε να διαβάζει την ημερομηνία από το πληκτρολόγιο. Αν δίνεται λανθασμένη είσοδος (πχ αρνητικό έτος) να γίνεται επαναληπτικά ανάγνωση.

2. Προσθέτε μια μέθοδο prevDay() που θα μειώνει την ημέρα κατά 1.

3. Προσθέστε μια μέθοδο με το όνομα bool before(Date &other), η οποία θα επιστρέφει αληθές αν η τρέχουσα ημερομηνία είναι πριν την ημερομηνία other και ψευδές αλλιώς.

4. Διορθώστε τον κώδικα όπου χρειάζεται ώστε να λαμβάνεται υπόψιν σας τα δίσεκτα έτη. Πιθανόν να χρειαστεί να αλλάξετε πάνω από 3 μεθόδους.

Πέμπτο σετ Ασκήσεων C++

Ιωάννης Γ. Τσούλος

2015

1 Κατηγορία μαθητών με δείκτες

Η επόμενη κατηγορία υλοποιεί μια κατηγορία για την αποθήκευση μαθητών ενός εξαμήνου με χρήση δεικτών. Στην συνέχεια δημιουργεί 5 μαθητές δυναμικά σαν δυναμικά πεδία πίνακα. Στα ιδιωτικά πεδία της κατηγορίας Student περιλαμβάνονται τα ακόλουθα:

1. Όνομα μαθητή
2. Επίθετο μαθητή
3. Αριθμός μαθημάτων
4. Πίνακας βαθμολογίας

Στα δημόσια πεδία περιλαμβάνονται

1. Μέθοδος δημιουργίας
2. Μέθοδοι set και get
3. Μέθοδος εμφάνισης του μαθητή
4. Μέθοδος διαγραφής

Η main() συνάρτηση θα δημιουργήσει 5 δυναμικά αντικείμενα της κατηγορίας Student και στην συνέχεια θα τα εμφανίσει στην οθόνη. Η δήλωση της κατηγορίας έχει ως ακολούθως:

```
1 #ifndef STUDENT_H_
2 #define STUDENT_H_
3
4 class Student {
5 private:
6         char *name,*lastname;
7         double *lessons;
8         int    countLessons;
9 public:
```

```
10          Student(char *n,char *l,int c);
11          void setLesson(int index,double value);
12          double getLesson(int index);
13          void printDetails();
14          char *getName();
15          char *getLastname();
16          void setName(char *n);
17          void setLastname(char *l);
18          ~Student();
19  };
20
21  #endif
```

Η υλοποίηση της κατηγορίας έχει ως ακολούθως:

```
1   #include "Student.h"
2   # include <string.h>
3   # include <iostream>
4   using namespace std;
5
6   Student::Student(char *n,char *l,int c)
7   {
8           setName(n);
9           setLastname(l);
10          if(c<0) countLessons=1; else countLessons=c;
11          lessons=new double[countLessons];
12  }
13
14  void    Student::setLesson(int index,double value)
15  {
16          if(index<0 || index>=countLessons)
17                  return;
18          if(value<0 || value>10)
19                  lessons[index]=0;
20          else
21                  lessons[index]=value;
22  }
23
24  double  Student::getLesson(int index)
25  {
26          if(index<0 || index>=countLessons) return 0;
27          return lessons[index];
28  }
29
30  void    Student::printDetails()
31  {
32          int i;
```

```
33          cout<<"** Student Details ***"<<endl;
34          cout<<"Name:    "<<name<<endl;
35          cout<<"Lastname:"<<lastname<<endl;
36          for(i=0;i<countLessons;i++)
37                  cout<<"Lesson: "<<(i+1)<<" Grade: "<<lessons[i]<<endl;
38  }
39
40  char    *Student::getName()
41  {
42          return name;
43  }
44
45  char    *Student::getLastname()
46  {
47          return lastname;
48  }
49
50  void    Student::setName(char *n)
51  {
52          name=new char[strlen(n)+1];
53          strcpy(name,n);
54  }
55
56  void    Student::setLastname(char *l)
57  {
58          lastname=new char[strlen(l)+1];
59          strcpy(lastname,l);
60  }
61
62  Student::~Student()
63  {
64          delete[] name;
65          delete[] lastname;
66          delete[] lessons;
67  }
```

Τέλος η συνάρτηση main() έχει ως εξής:

```
1  #include "Student.h"
2  #include <iostream>
3  using namespace std;
4  int main()
5  {
6          char myname[100];
7          char mylastname[100];
8          Student *myclass[5];
9          int i,j,nlessons;
```

```
10          for(i=0;i<5;i++)
11          {
12                  cout<<"Give details for student "<<(i+1)<<endl;
13                  cout<<"Give name "<<endl;
14                  cin>>myname;
15                  cout<<"Give lastname"<<endl;
16                  cin>>mylastname;
17                  cout<<"Give number of lessons "<<endl;
18                  cin>>nlessons;
19                  myclass[i]=new Student(myname,mylastname,nlessons);
20                  for(j=0;j<nlessons;j++)
21                  {
22                          double grade;
23                          cout<<"Give grade for lesson "<<(j+1)<<endl;
24                          cin>>grade;
25                          myclass[i]->setLesson(j,grade);
26                  }
27
28          }
29          for(i=0;i<5;i++)
30          {
31                  myclass[i]->printDetails();
32                  delete myclass[i];
33          }
34          return 0;
35  }
```

Με βάση τα παραπάνω κάνετε τις επόμενες τροποποιήσεις:

1. Να προστεθεί στα ιδιωτικά πεδία της Student και ο αριθμός μητρώου του σπουδαση. Να κάνετε όλες τις απαραίτητες αλλαγές στην μέθοδο δημιουργίας και να προσθέσετε και μεθόδους set και get για αυτό το πεδίο

2. Να προστεθεί μέθοδος average() στην κατηγορία Student που να υπολογίζει και να επιστρέφει τον μέσο όρο βαθμολογίας του σπουδαστή

3. Εμφανίστε στην main() συνάρτηση τους μέσους όρους βαθμολογίας για κάθε σπουδαστή

4. Εμφανίστε στην main() συνάρτηση τα στοιχεία του σπουδαστή (Όνομα, Επίθετο) με τον καλύτερο μέσο όρο

5. Να προστεθεί η μέθοδος bool isBetter(Student *other); Η μέθοδος αυτή δέχεται σαν όρισμα έναν δείκτη σε αντικείμενο Student και επιστρέφει αληθές αν ο τρέχων σπουδαστής έχει καλύτερο μέσο όρο από τον other και false σε άλλη περίπτωση.

6. Αλλάξτε τις δηλώσεις των πεδίων name,lastname από char * σε string

Έκτο σετ Ασκήσεων C++

Ιωάννης Γ. Τσούλος

2015

1 Κατηγορία βιβλίων

Να δημιουργηθεί κατηγορία βιβλίων με τα εξής ιδιωτικά πεδία:

1. Τίτλο βιβλίου
2. Συγγραφέα
3. Κωδικό
4. Αριθμό σελιδών

Στα δημόσια πεδία να υπάρχουν τα ακόλουθα

1. Δύο μέθοδοι δημιουργίας της επιλογής σας
2. Μέθοδοι set και get
3. Μέθοδος εκτύπωσης των στοιχείων
4. Μέθοδος διαγραφής που εμφανίζει τον τίτλο του βιβλίου που θα διαγραφεί.

Η κατηγορία βιβλίου είναι η ακόλουθη:

```
1  # include <string>
2  using namespace std;
3
4  class Book
5  {
6          private:
7                  string author,title;
8                  int pages;
9                  int code;
10         public:
11                 Book();
12                 Book(int c,string a,string t,int p);
13                 string getAuthor();
14                 string getTitle();
```

```
15                      int     getPages();
16                      int     getCode();
17                      void    setAuthor(string s);
18                      void    setTitle(string s);
19                      void    setPages(int p);
20                      void    setCode(int c);
21                      void    printDetails();
22                      ~Book();
23  };
```

Η υλοποίηση της κατηγορίας βιβλίου είναι η ακόλουθη:

```
1   # include "book.h"
2   # include <iostream>
3   using namespace std;
4   Book::Book()
5   {
6           setCode(0);
7           setPages(0);
8           setAuthor("");
9           setTitle("");
10  }
11
12  Book::Book(int c,string a,string t,int p)
13  {
14          setCode(c);
15          setPages(p);
16          setAuthor(a);
17          setTitle(t);
18  }
19
20  string  Book::getAuthor()
21  {
22          return author;
23  }
24
25  string  Book::getTitle()
26  {
27          return title;
28  }
29
30  int     Book::getPages()
31  {
32          return pages;
33  }
34
35  int     Book::getCode()
```

```
36 {
37         return code;
38 }
39
40 void    Book::setAuthor(string s)
41 {
42         author=s;
43 }
44
45 void    Book::setTitle(string s)
46 {
47         title=s;
48 }
49
50 void    Book::setPages(int p)
51 {
52         if(p<0) pages=0; else pages=p;
53 }
54
55 void    Book::setCode(int c)
56 {
57         if(c<0) code=0; else code=c;
58 }
59
60 void    Book::printDetails()
61 {
62         cout<<"Title: "<<title<<" Author: "<<author<<"\n"<<
63                 "Code: "<<code<<" Pages: "<<pages<<endl;
64 }
65
66
67 Book::~Book()
68 {
69         cout<<"Deleting book "<<title<<endl;
70 }
```

Μια ενδεικτική main συνάρτηση στην οποία γίνεται χρήση δεικτών και δυναμικών αντικειμένων βιβλίου είναι η ακόλουθη:

```
1 #include "book.h"
2 #include <iostream>
3 using namespace std;
4
5 int main()
6 {
7         Book a(100,"King","The Stand",800);
8         Book *pa=&a;
```

```
 9          Book *b=new Book(200,"Kazantzakis","Zorbas",500);
10          pa->printDetails();
11          pa->setPages(1000);
12          a.printDetails();
13          b->printDetails();
14          delete b;
15          return 0;
16  }
```

Με βάση τα παραπάνω κάνετε τις επόμενες προσθήκες:

1. Προσθέστε μια ακόμα συνάρτηση δημιουργίας Book(Book &other), στην οποία να γίνεται αντιγραφή κάθε πεδίου του αντικειμένου other στο τρέχον αντικείμενο. Για παράδειγμα θα οι σελίδες θα πρέπει να γίνουν ίσες με τις σελίδες του αντικειμένου Other.

2. Να προστεθεί η μέθοδος bool sameAuthor(Book *other) Η μέθοδος επιστρέφει αληθές αν το τρέχον βιβλίο και το other ανήκουν στον ίδιο συγγραφέα.

3. Να προστεθεί η μέθοδος bool equals(Book *other), που επιστρέφει αληθές αν το τρέχον βιβλίο και το other είναι πανομοιότυπα σε όλα τα πεδία της.

2 Βιβλιοθήκη

Με βάση την προηγούμενη κατηγορία να γίνει κατηγορία Library η οποία θα διαθέτει στα ιδιωτικά της πεδία

1. Έναν πίνακα δέκα δεικτών σε βιβλία Book

Στα δημόσια πεδία θα περιλαμβάνει

1. Μια μέθοδο δημιουργίας

2. Μια μέθοδο εισαγωγής νέου βιβλίου

3. Μια μέθοδο εμφάνισης όλων των βιβλίων

4. Μια μέθοδο διαγραφής βιβλίου με βάση τον κωδικό του

5. Μια μέθοδο αναζήτησης βιβλίου με βάση τον κωδικό του

Η δήλωσης της κατηγορίας έχει ως ακολούθως:

```
1  # include "book.h"
2  # define NBOOKS 10
3  class Library
4  {
5           private:
6                   Book *mybooks[NBOOKS];
7           public:
```

```
 8                       Library();
 9                       bool addBook(Book *b);
10                       void delBook(int code);
11                       Book *searchBook(int code);
12                       void     showBooks();
13                       ~Library();
14  };
```

Η υλοποίηση της κατηγορίας είναι:

```
 1  # include "library.h"
 2  # include <iostream>
 3  using namespace std;
 4
 5
 6  Library::Library()
 7  {
 8          int i;
 9          for(i=0;i<NBOOKS;i++) mybooks[i]=NULL;
10  }
11
12  bool     Library::addBook(Book *b)
13  {
14          int i;
15          for(i=0;i<NBOOKS;i++)
16                  if(mybooks[i]==NULL)
17                  {
18                          mybooks[i]=new Book(b->getCode(),
19                                  b->getAuthor(),b->getTitle(),
20                                  b->getPages());
21                          return true;
22                  }
23          return false;
24  }
25
26  void     Library::delBook(int code)
27  {
28          int i;
29          for(i=0;i<NBOOKS;i++)
30          {
31                  if(mybooks[i]!=NULL)
32                  {
33                          if(mybooks[i]->getCode()==code)
34                          {
35                                  delete mybooks[i];
36                                  mybooks[i]=NULL;
37                                  return;
```

```
38                              }
39                      }
40              }
41      }
42
43      Book    *Library::searchBook(int code)
44      {
45              int i;
46              for(i=0;i<NBOOKS;i++)
47              {
48                      if(mybooks[i]!=NULL)
49                      {
50                              if(mybooks[i]->getCode()==code)
51                                      return mybooks[i];
52                      }
53              }
54              return NULL;
55      }
56
57
58      void    Library::showBooks()
59      {
60              int i;
61              for(i=0;i<NBOOKS;i++)
62                      if(mybooks[i]!=NULL)
63                              mybooks[i]->printDetails();
64      }
65
66      Library::~Library()
67      {
68              int i;
69              for(i=0;i<NBOOKS;i++)
70                      if(mybooks[i]!=NULL) delete mybooks[i];
71      }
```

Τέλος μια ενδεικτική main συνάρτηση με χρήση μενού είναι η επόμενη:

```
1   #include "library.h"
2   #include <iostream>
3   using namespace std;
4
5   int menu()
6   {
7           int option;
8           do{
9                   cout<<"1-ADD_BOOK_\n";
10                  cout<<"2-SEARCH_FOR_A_BOOK\n";
```

```
11                      cout<<"3-SHOW BOOKS\n";
12                      cout<<"4-DELETE BOOK\n";
13                      cout<<"5-QUIT\n";
14                      cin>>option;
15              }
16              while(option<1 || option>5);
17              return option;
18      }
19
20      int main()
21      {
22              Library mylib;
23              string author,title;
24              int pages,code;
25              int option;
26              do
27              {
28                      option=menu();
29                      if(option==1)
30                      {
31                              cout<<"Enter author \n";
32                              cin>>author;
33                              cout<<"Enter title \n";
34                              cin>>title;
35                              cout<<"Enter code \n";
36                              cin>>code;
37                              cout<<"Enter number of pages \n";
38                              cin>>pages;
39                              Book p(code,author,title,pages);
40                              mylib.addBook(&p);
41                      }
42                      else
43                      if(option==2)
44                      {
45                              cout<<"Enter code \n";
46                              cin>>code;
47                              Book *p=mylib.searchBook(code);
48                              if(p!=NULL) p->printDetails();
49                              else cout<<"Book not found\n";
50                      }
51                      else
52                      if(option==3)
53                      {
54                              mylib.showBooks();
55                      }
56                      else
```

```
57                      if(option==4)
58                      {
59                              cout<<"Enter code \n";
60                              cin>>code;
61                              mylib.delBook(code);
62                      }
63              }while(option!=5);
64              return 0;
65      }
```

Με βάση τα παραπάνω να γίνουν οι επόμενες αλλαγές:

1. Να προσθέσετε μια μέθοδο int countBooks(string author) που θα επιστρέφει το πλήθος των βιβλίων του author
2. Να διορθώσετε την μέθοδο addBook(Book *p) ώστε αν υπάρχει ήδη το βιβλίο με κωδικό ίδιο με του βιβλίου p, να μην γίνεται προσθήκη του βιβλίου

Κληρονομικότητα και τελεστές

Ιωάννης Γ. Τσούλος

May 16, 2015

1 Το παράδειγμα των σχημάτων

Η κατηγορία Shape υλοποιεί αφηρημένα σχήματα. Στα δημόσια πεδία υπάρχουν οι υπερβατικές μέθοδοι area() και perimeter() για εμβαδό και περίμετρο αντίστοιχα καθώς και η μέθοδος printDetails() που εμφανίζει στοιχεία του σχήματος. Η δήλωση της κατηγορίας έχει ως ακολούθως:

```
1  # ifndef __SHAPE__H
2  # define __SHAPE__H
3  class Shape
4  {
5          public:
6                  virtual double area()=0;
7                  virtual double perimeter()=0;
8                  void printDetails();
9                  friend bool operator<=(Shape &a,Shape &b);
10 };
11 # endif
```

Ο κώδικας της κατηγορίας Shape έχει ως ακολούθως:

```
1  # include "shape.h"
2  # include <iostream>
3  using namespace std;
4  void Shape::printDetails()
5  {
6          cout<<"Area is      "<<area()<<endl;
7          cout<<"Perimeter is "<<perimeter()<<endl;
8  }
9
10 bool operator<=(Shape &a,Shape &b)
11 {
12         if(a.area()<=b.area()) return true;
13         else return false;
14 }
```

Με βάση την κατηγορία Shape ορίζουμε τις κατηγορίες Circle και Rectangle για την υλοποίηση κύκλων και ορθογωνίων αντίστοιχα. Η δήλωση της κατηγορίας Circle έχει ως ακολούθως:

```
1  # ifndef __CIRCLE__H
2  # define __CIRCLE__H
3  # include "shape.h"
4
5  class Circle: public Shape
6  {
7          private:
8                  double X,Y,R;
9          public:
10                 Circle(double a,double b,double c);
11                 virtual double area();
12                 virtual double perimeter();
13                 double getx();
14                 double gety();
15                 double getr();
16                 void setCenter(double a,double b);
17                 double setR(double a);
18 };
19 # endif
```

Η υλοποίηση της κατηγορίας Circle είναι η επόμενη:

```
1  # include <math.h>
2  # include "circle.h"
3  Circle::Circle(double a,double b,double c)
4  {
5          setCenter(a,b);
6          setR(c);
7  }
8
9  double  Circle::area()
10 {
11         return M_PI*R*R;
12 }
13
14 double  Circle::perimeter()
15 {
16         return 2.0 * M_PI * R;
17 }
18
19 double  Circle::getx()
20 {
21         return X;
```

```
22  }
23
24  double   Circle::gety()
25  {
26          return Y;
27  }
28
29  double   Circle::getr()
30  {
31          return R;
32  }
33
34  void     Circle::setCenter(double a,double b)
35  {
36          X=a;
37          Y=b;
38  }
39
40  double   Circle::setR(double a)
41  {
42          if(a<0)
43                  R=1.0;
44          else
45                  R=a;
46  }
```

Η κατηγορία Rectangle ορίζεται ως εξής:

```
1  # ifndef __RECTANGLE__H
2  # define __RECTANGLE__H
3  # include "shape.h"
4  class Rectangle :public Shape
5  {
6          private:
7                  double a;
8          public:
9                  Rectangle(double ma);
10                 virtual double area();
11                 virtual double perimeter();
12                 double geta();
13                 void seta(double ma);
14 };
15 # endif
```

και η υλοποίησή της είναι:

```
1  # include "rectangle.h"
2  Rectangle::Rectangle(double ma)
```

```
3  {
4          seta(ma);
5  }
6
7  double  Rectangle::area()
8  {
9          return a*a;
10 }
11
12 double  Rectangle::perimeter()
13 {
14         return 4*a;
15 }
16
17 double  Rectangle::geta()
18 {
19         return a;
20 }
21
22 void    Rectangle::seta(double ma)
23 {
24         if(ma<0)
25                 a=1;
26         else
27                 a=ma;
28 }
```

Τέλος μια ενδεικτική main() συνάρτηση που δημιουργεί έναν πίνακα σχημάτων και εμφανίζει τα στοιχεία ταξινομημένα ως προς το εμβαδόν (φιλικός τελεστής <=) είναι η επόμενη:

```
1  #include "circle.h"
2  #include "rectangle.h"
3  #include <iostream>
4  using namespace std;
5
6  int main()
7  {
8          Shape *shapes[10];
9          int i,j;
10         double a,b,c;
11         for(i=0;i<10;i++)
12         {
13                 if(i%2==0)
14                 {
15                         //make circle
16                         cout<<"Enter center of circle ";
```

```
17                              cin>>a>>b;
18                              cout<<"Enter R ";
19                              cin>>c;
20                              shapes[i]=new Circle(a,b,c);
21                      }
22                      else
23                      {
24                              //make rectangle
25                              cout<<"Enter a ";
26                              cin>>a;
27                              shapes[i]=new Rectangle(a);
28                      }
29              }
30              //sort shapes
31              for(i=0;i<10;i++)
32              {
33                      for(j=0;j<9;j++)
34                      {
35                              if((*shapes[j])<=(*shapes[j+1]))
36                              {
37                                      Shape *t=shapes[j];
38                                      shapes[j]=shapes[j+1];
39                                      shapes[j+1]=t;
40                              }
41                      }
42              }
43              //print shapes
44              for(i=0;i<10;i++)
45              {
46                      cout<<"Shape : "<<i<<endl;
47                      shapes[i]->printDetails();
48              }
49              //delete shapes
50              for(i=0;i<10;i++)
51                      delete shapes[i];
52              return 0;
53      }
```

Με βάση τα παραπάνω υλοποιήστε τα επόμενα:

1. Εμφανίστε στην συνάρτηση main() το μικρότερο σχήμα
2. Να προστεθεί η συνάρτηση δημιουργίας Circle(), στην οποία ο χρήστης εισάγει από το πληκτρολόγιο X, Y,R
3. Να προστεθεί η συνάρτηση δημιουργίας Rectangle(), στην οποία ο χρήστης είσαγει από το πληκτρολόγιο την πλευρά a

4. Να χρησιμοποιήστε τις παραπάνω συναρτήσεις δημιουργίας στην main()

2 Χρονικές στιγμές

Στην συνέχεια δημιουργείται μια κατηγορία για την περιγραφή χρονικών στιγμών (Time) Εκτός των άλλων πεδίων η κατηγορία περιλαμβάνει και δύο τελεστές μοναδιαίας αύξησης και μείωσης της χρονικής στιγμής. Η δήλωση της κατηγορίας Time έχει ως ακολούθως:

```
1  # ifndef __TIME__H
2  class Time
3  {
4          private:
5                  int hour,minute,second;
6          public:
7                  Time(int h,int m,int s);
8                  void details();
9                  Time& operator++();
10                 Time& operator--();
11 };
12
13 # define __TIME__H
14 # endif
```

Η υλοποίηση της κατηγορίας είναι:

```
1  # include <string.h>
2  # include "time.h"
3  # include <iostream>
4  using namespace std;
5
6  Time::Time(int h,int m,int s)
7  {
8          hour=h;
9          minute=m;
10         second=s;
11 }
12
13 void     Time::details()
14 {
15         cout<<hour<<":"<<minute<<":"<<second<<endl;
16 }
17
18 Time&    Time::operator++()
19 {
20         ++second;
21         if(second==60)
```

```
22          {
23                  minute++;
24                  second=0;
25                  if(minute==60)
26                  {
27                          minute=0;
28                          ++hour;
29                          if(hour==24) hour=0;
30                  }
31          }
32          return *this;
33  }
34
35  Time&   Time::operator--()
36  {
37          --second;
38          if(second<0)
39          {
40                  second=59;
41                  minute--;
42                  if(minute<0)
43                  {
44                          minute=59;
45                          hour--;
46                          if(hour<0) hour=23;
47                  }
48          }
49          return *this;
50  }
```

Τέλος μια ενδεικτική main() συνάρτηση είναι η επόμενη:

```
1  #include "time.h"
2  int main()
3  {
4          Time t1(23,59,59);
5          t1.details();
6          ++t1;
7          t1.details();
8          Time t2(0,0,0);
9          t2.details();
10         --t2;
11         t2.details();
12         return 0;
13 }
```

Με βάση τα παραπάνω να κάνετε τα ακόλουθα:

1. Να προστεθεί συνάρτηση δημιουργίας Time() στην οποία ο χρήστης θα εισάγει από το πληκτρολόγιο τα στοιχεία της χρονικής στιγμής. Άκυρες τιμές πχ. αρνητική ώρα δεν θα επιτρέπονται και ο χρήστης θα πρέπει να τις εισάγει ξανά.

2. Να δημιουργηθεί φιλικός τελεστής - ανάμεσα σε αντικείμενα Time. Το αποτέλεσμα θα είναι η διαφορά των δύο χρονικών στιγμών σε δευτερόλεπτα