ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ Α.Π.Θ.
ΕΡΓΑΣΤΗΡΙΟ C++
ΕΞΑΜΗΝΟ Γ – Ακαδημαϊκό Έτος 2006-2007

ΠΑΡΑΔΕΙΓΜΑ δυναμικής δέσμευσης και αποδέσμευσης μνήμης στη C++ μέσω των new και delete.

```
// create.cpp

#include <iostream>
using namespace std;

class values {
   public:
     values() : value1(0), value2(0)
     {
        count++;
        cout << "default constructing object " << count << endl;
     }
     ~values()
     {
        cout << "destructing! object " << count << endl;
        count--;
     }
     void setvalues(int i, int j) { value1 = i; value2 = j; }
     long getvalue1() const { return value1; }
     long getvalue2() const { return value2; }
   private:
     long value1;
     long value2;
     static int count;
};
int values::count = 0;

const int MAX_ELEMENTS = 20;

int main()
{
   int *intArray;
   intArray = new int[ MAX_ELEMENTS ];
   int i = 0;
```

```
    for(i = 0; i < MAX_ELEMENTS; i++ )
       intArray[i] = i;

    int *int_ptr = new int;
    *int_ptr = 1000;
    cout << "int_ptr = " << *int_ptr << endl;
    delete int_ptr;

    for(i = 0; i < MAX_ELEMENTS; i++ )
       cout << "intArray[i] = " << intArray[i] << endl;

    delete [] intArray;

    values *sArray = new values[ MAX_ELEMENTS ];
    for(i = 0; i < MAX_ELEMENTS; i++ )
    {
       sArray[i].setvalues(i, i * 10);
    }
    for(i = 0; i < MAX_ELEMENTS; i++ )
    {
       cout << "sArray[i].value1 = " << sArray[i].getvalue1();
       cout << ", sArray[i].value2 = " << sArray[i].getvalue2()
            << endl;
    }
    delete [] sArray;

    return( 0 );
}
```

Υπερφόρτωσης τελεστών

- Παράδειγμα Α)

```
// point.C
#include <cmath>
#include <iostream>
using namespace std;

class Point {
public:
   // constructors
   Point() : x(0), y(0) { } // προκαθορισμένος δομητής
   Point(double x1) { x = x1; y = 0; }  // δομητής με ένα όρισμα
   Point(double x1, double y1) { move(x1, y1); }
   Point(const Point & pt); // δομητής αντιγραφέας

   // overloaded operators
   bool operator==(const Point &p1);
   Point& operator=(const Point &p1);
   Point operator*(double n) const;
   Point operator-();

   // data accessors
   double getx() const { return x; }
   double gety() const { return y; }

   // data mutators
   void setx(double x1){ x = x1; }
   void sety(double y1){ y = y1; }
   void move(double x1, double y1) { x = x1; y = y1; }

   // computation and print functions
   void print() const { cout << "(" << x << "," << y << ")"; }
   double distance() const;

   inline static Point origin();
private:
   double x, y;
};

bool Point::operator==(const Point &p1)
{
    return ((getx() == p1.getx()) && (gety() == p1.gety()));
}

Point& Point::operator=(const Point &p1) // Τελεστής ανάθεσης τιμής!
{
```

```
    if (this != &p1){
       x = p1.x;
       y = p1.y;
    }
    return *this;
}
// Πολ/σμός των συντεταγμένων με n
Point Point::operator*(double n) const
{
    return Point(x*n,y*n);
}

Point Point::operator-() // Μοναδιαίος τελεστής!
{
    return Point(-x,-y);
}

double Point::distance() const
{
  return sqrt(getx() * getx() + gety() * gety());
}

inline Point Point::origin()
{
  return Point();
}

Point::Point(const Point & pt) : x(pt.x), y(pt.y)
{
  // Κενό σώμα!
}

int main()
{
    double n;
    Point p(20, 43);
    Point q;                                  // αρχικοποίηση q σε (0,0)
    Point r(20);
    Point s(p);                               // αρχικοποίηση με το p.

    cout << "p: " << p.getx() << ' ' << p.gety() << endl;

    q.move(30, 30);                           // move q

    cout << "q: " << q.getx() << ' ' << q.gety() << endl;

    q = p; // ανάθεση τιμής!
    cout << "q: " << q.getx() << ' ' << q.gety() << endl;
```

```
    if (q==p) cout << "Points q and p match!\n";

    p= p*2;
    p=-p;
    cout << "-p: " << p.getx() << ' ' << p.gety() << endl;

    s.print(); // εκτύπωση μέσω κλήσης συνάρτησης μέλους

    cin >> n;
    s.setx(n);
    cin >> n;
    s.sety(n);
    if (s == (Point::origin()))
       cout << "s is Origin\n";
    else
       cout << "Distance of s = " << s.distance() << endl;

    return 0;
}
```

ΑΣΚΗΣΕΙΣ

- Υπερφορτώστε τους δυαδικούς τελεστές: “+”, “-”, “+=”, “- =” ώστε να μπορούμε να έχουμε p1+p2, p1-p2, p1+=p2, p1 - = p2.
- Μπορείτε να υπερφορτώσετε τον τελεστή “<<” ώστε να μπορούμε μέσω αυτού να τυπώσουμε απ’ευθείας ένα σημείο;

ΠΑΡΑΔΕΙΓΜΑ Β)
Δίνεται η παρακάτω τάξη που ορίζει ένα ρολόϊ που μετρά από 00:00:00 έως 23:59:59.
Να συμπληρωθεί το παρακάτω πρόγραμμα με τις αναγκαίες συναρτήσεις ώστε να δουλεύει.

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

class mytime {
   public:
      void display() const           //format: 23:15:01
      {
        if(hours < 10)    cout << '0';
        cout << hours << ':';
        if(minutes < 10)  cout << '0';
        cout << minutes << ':';
        if(seconds < 10)  cout << '0';
        cout << seconds;
      }
   private:
      int hours;                     //0 to 23
      int minutes;                   //0 to 59
      int seconds;                   //0 to 59
};


int main()
{
    mytime t1(5,30,0);
    cout << "t1   : ";
    t1.display();

    mytime t2(t1);
    cout << "t2(t1) : ";
    t2.display();

    t1++;
    --t2;
    cout << "t1++ : ";
    t1.display();
    cout << "--t2 : ";
    t2.display();

    mytime t3 = t2 + t1*3;
    cout << "t3 = t2 + t1*3  : ";
    t3.display();

    t1 = t3 - t1;
```

```
    cout << "t1 = t3 - t1 : ";
    t1.display();

    return 0;
}
```

ΑΠΑΝΤΗΣΕΙΣ στην υπερφόρτωση συναρτήσεων

ΠΑΡΑΔΕΙΓΜΑ Α)

```
// point.C - private data members
#include <cmath>
#include <iostream>
using namespace std;

class Point {
public:
   typedef double value_type;
   // constructors
   Point() : x(0), y(0) { } // προκαθορισμένος δομητής
   Point(double x1) { x = x1; y = 0; }  // δομητής με ένα όρισμα
   Point(double x1, double y1) { move(x1, y1); }
   Point(const Point & pt); // δομητής αντιγραφέας

   // overloaded operators
   bool operator==(const Point &p1);
   Point& operator=(const Point &p1);
   Point operator*(double n) const;
   Point operator-();

   Point operator+(const Point &p1) const;
   Point operator-(const Point &p1) const;
   Point& operator+=(const Point &p1);
   Point& operator-=(const Point &p1);
   //ostream& operator << ( ostream&); // Αν ο τελεστής << είναι
μέλος της τάξης
   friend ostream& operator<<(ostream&, const Point&);

   // data accessors
   double getx() const { return x; }
   double gety() const { return y; }

   // data mutators
   void setx(value_type x1){ x = x1; }
   void sety(value_type y1){ y = y1; }
   void move(value_type x1, value_type y1) { x = x1; y = y1; }

   // computation and print functions
   void print() const { cout << "(" << x << "," << y << ")"; }
   value_type distance() const;

   inline static Point origin();
private:
   value_type x, y;
```

```
};

bool Point::operator==(const Point &p1)
{
    return ((getx() == p1.getx()) && (gety() == p1.gety()));
}

Point& Point::operator=(const Point &p1) // Τελεστής ανάθεσης τιμής!
{
   if (this != &p1){
      x = p1.x;
      y = p1.y;
   }
   return *this;
}
// Πολ/σμός των συντεταγμένων με n
Point Point::operator*(double n) const
{
   return Point(x*n,y*n);
}

Point Point::operator-() // Μοναδιαίος τελεστής!
{
   return Point(-x,-y);
}

////////////////////////////////////////////////
Point Point::operator+(const Point &p1) const
{
   return Point(x+p1.x, y+p1.y);
}
Point Point::operator-(const Point &p1) const
{
   return Point(x-p1.x, y-p1.y);
}
Point& Point::operator+=(const Point &p1)
{
   x+=p1.x;
   y+=p1.y;
   return *this;
}
Point& Point::operator-=(const Point &p1)
{
   x-=p1.x;
   y-=p1.y;
   return *this;
}
ostream& operator<<(ostream& os, const Point& pt)
{
```

```
    os << "(" << pt.x << "," << pt.y << ")" << endl;
     return os;
}
/*  ο τελεστής << ως συνάρτηση μέλος της τάξης!
ostream& Point::operator<< ( ostream& os ) {
    os << "(" << x << "," << y << ")" << endl;
    return os;
}
*/
////////////////////////////////////////////////

Point::value_type Point::distance() const
{
  return sqrt(getx() * getx() + gety() * gety());
}

inline Point Point::origin()
{
  return Point();
}

Point::Point(const Point & pt) : x(pt.x), y(pt.y)
{
  // Κενό σώμα!
}

int main()
{
   Point::value_type n;
   Point p(20, 43);
   Point q;                                  // αρχικοποίηση q σε (0,0)
   Point r(20);
   Point s(p);                               // αρχικοποίηση με το p.

   cout << "p: " << p.getx() << ' ' << p.gety() << endl;

   q.move(30, 30);                           // move q

   cout << "q: " << q.getx() << ' ' << q.gety() << endl;

   q = p; // ανάθεση τιμής!
   cout << "q: " << q.getx() << ' ' << q.gety() << endl;

   if (q==p) cout << "Points q and p match!\n";

   p= p*2;
   p=-p;
   cout << "-p: " << p.getx() << ' ' << p.gety() << endl;
```

```
   s.print(); // εκτύπωση μέσω κλήσης συνάρτησης μέλους

   p = q+r;
   cout << "p = q + r = " << p.getx() << ' ' << p.gety() << endl;
   r += p;
   cout << "r += p : " << r.getx() << ' ' << r.gety() << endl;
   p -= r;
   cout << "p -= r : " << p.getx() << ' ' << p.gety() << endl;
   r = r-p;
   cout << "r = r - p = " << r.getx() << ' ' << r.gety() << endl;

   cout << "p=" << p << "q=" << q << "r=" << r;
   // Αν ο τελεστής << είναι μέλος της τάξης Point τότε ισχύουν τα
παρακάτω:
   //cout << "p="; p << cout; cout << "q="; q << cout; cout << "r=";
r << cout;

   cin >> n;
   s.setx(n);
   cin >> n;
   s.sety(n);
   if (s == (Point::origin()))
      cout << "s is Origin\n";
   else
      cout << "Distance of s = " << s.distance() << endl;

   return 0;
}
```

ΠΑΡΑΔΕΙΓΜΑ Β)

```
// Προσοχή δε γίνεται έλεγχος για εισαγωγή έγκυρων τιμών!.
#include <iostream>
using namespace std;

class mytime {
   public:
      // constructors
      mytime() : hours(0), minutes(0), seconds(0)
      {  }
      mytime(int h, int m, int s) :  //3-arg constructor
           hours(h), minutes(m), seconds(s)
      {  }

      mytime(const mytime&);  // copy constructor

      // overloaded operators
      mytime operator+(const mytime& t1) const;
      mytime operator-(const mytime& t1) const;
      mytime operator++(int);
      const mytime& operator--();
      mytime operator*(int n);

      void display() const           //format: 23:15:01
      {
        if(hours < 10)    cout << '0';
        cout << hours << ':';
        if(minutes < 10)  cout << '0';
        cout << minutes << ':';
        if(seconds < 10)  cout << '0';
        cout << seconds << endl;
      }
   private:
      int hours;                     //0 to 23
      int minutes;                   //0 to 59
      int seconds;                   //0 to 59
};

mytime::mytime(const mytime& t1) :
          hours(t1.hours), minutes(t1.minutes), seconds(t1.seconds)
{
}

mytime mytime::operator +(const mytime& t1) const
{
     int tim = (hours+t1.hours)*3600 +
               (minutes+t1.minutes)*60 + seconds + t1.seconds;
```

```
    mytime temp;
    temp.hours = tim / 3600;
    tim %=3600;
    temp.minutes = tim / 60;
    temp.seconds = tim % 60;

    return temp;
}

mytime mytime::operator -(const mytime& t1) const
{
     int tim = (hours-t1.hours)*3600 +
               (minutes-t1.minutes)*60 + seconds - t1.seconds;
     if (tim < 0) return mytime(0,0,0);

    mytime temp;
    temp.hours = tim / 3600;
    tim %=3600;
    temp.minutes = tim / 60;
    temp.seconds = tim % 60;

    return temp;
}

mytime mytime::operator ++(int)  // post-increment
{
    mytime temp=*this;
    int tim = hours*3600 + minutes*60 + seconds;
    tim++;
    hours = tim / 3600;
    tim %= 3600;
    minutes = tim / 60;
    tim %= 60;
    seconds = tim;

    return temp;
}

const mytime& mytime::operator --()  // pre-increment
{
     int tim = hours*3600 + minutes*60 + seconds;
     tim--;
     if (tim <0) tim=0;
     hours = tim / 3600;
     tim %= 3600;
     minutes = tim / 60;
     tim %= 60;
     seconds = tim;
```

```
      return *this;
}
mytime mytime::operator *(int n)
{
      int tim = n*(hours*3600+ minutes*60 + seconds);
      mytime t;

      t.hours = tim / 3600;
      tim %= 3600;
      t.minutes = tim / 60;
      tim %= 60;
      t.seconds = tim;

      return t;
}

int main()
{
     mytime t1(5,30,0);
     cout << "t1    : ";
     t1.display();

     mytime t2(t1);
     cout << "t2(t1) : ";
     t2.display();

     t1++;
     --t2;
     cout << "t1++ : ";
     t1.display();
     cout << "--t2 : ";
     t2.display();

     mytime t3 = t2 + t1*3;
     cout << "t3 = t2 + t1*3  : ";
     t3.display();

     t1 = t3 - t1;
     cout << "t1 = t3 - t1 : ";
     t1.display();

     return 0;
}
```

ΣΗΜΕΙΩΣΗ

- Παρόμοια μπορούν να οριστούν και οι πράξεις: “t1--”, “++t1”.