Κλάσεις και αντικείμενα

Μεταβίβαση αντικειμένων σε συναρτήσεις

- Μεταβίβαση κατ’ αξία (by value)

Είναι ο προκαθορισμένος τρόπος μεταβίβασης παραμέτρων στη C++. Όταν μεταβιβάζουμε ένα αντικείμενο σε μια συνάρτηση, η αντίστοιχη παράμετρος πρέπει να είναι ένα αντικείμενο της ίδιας κλάσης. Οι τιμές των μεταβλητών – μελών του ορίσματος αντιγράφονται στις αντίστοιχες μεταβλητές – μέλη της παραμέτρου.

Στη παρακάτω συνάρτηση ogos() μεταβιβάζεται ένα αντικείμενο – ορθογώνιο παραλληλόγραμμο, το οποίο αντιστοιχεί στη μια πλευρά του παραλληλεπιπέδου καθώς και το ύψος του. Η συνάρτηση επιστρέφει ως τιμή τον όγκο του παραλληλεπιπέδου:

```
float ogos (rectangle rec, float ypsos)
{
   float ogos_rec;
   ogos_rec = rec.emvado() * ypsos;
   return ogos_rec;
}

int main()
{
 rectangle rec1(10,2);
 cout << "Εμβαδό παραλληλογράμμου="
        << rec1.emvado() << endl;
 cout << "Όγκος παραλληλεπιπέδου με ύψος 5 = "
        << ogos (rec1, 5) << endl;
 return 0;
}
```

Εμβαδό παραλληλογράμμου = 20
Όγκος παραλληλεπιπέδου με ύψος 5 = 100

- Επειδή η μεταβίβαση παραμέτρων έγινε κατ’ αξία, οι αλλαγές στις μεταβλητές – μέλη της παραμέτρου rec δεν επηρεάζουν τις αντίστοιχες μεταβλητές – μέλη του αντικειμένου ορίσματος με το οποίο κλήθηκε η συνάρτηση

- Μεταβίβαση κατ’ αναφορά (by reference)

Η μεταβίβαση παραμέτρων κατά την κλήση μιας συνάρτησης μπορεί να γίνει με τη χρήση αναφορικών παραμέτρων. Στη περίπτωση αυτή με τη τροποποίηση των τιμών των αναφορικών παραμέτρων της η συνάρτηση επηρεάζει τα αντίστοιχα ορίσματα με τα οποία κλήθηκε.

Μια αναφορική παράμετρος δηλώνεται με το πρόθεμα & πριν το όνομα της. Στη περίπτωση που η αναφορική παράμετρος είναι αντικείμενο μιας κλάσης, όποιες αλλαγές γίνουν στις μεταβλητές – μέλη αυτού του αντικειμένου επηρεάζουν τις αντίστοιχες μεταβλητές – μέλη του ορίσματος με το οποίο κλήθηκε η συνάρτηση.

Στη παρακάτω έκδοση της συνάρτησης ogos() η παράμετρος rec δηλώνεται ως αναφορική:

```
float ogos (rectangle &rec, float ypsos)
 {
   float ogos_rec;
   ogos_rec = rec.emvado() * ypsos;
   rec.set_ab(7,7);
   return ogos_rec;
 }
```

Εφαρμόζοντας τη μέθοδο set_ab() στο αντικείμενο rec αλλάζουμε τις διαστάσεις του σε 7x7. Η αλλαγή αυτή όμως τώρα επηρεάζει το αντίστοιχο αντικείμενο – όρισμα με το οποίο κλήθηκε η συνάρτηση

Συναρτήσεις που επιστρέφουν αντικείμενα

- Μια συνάρτηση μπορεί να επιστρέψει ως τιμή ένα αντικείμενο μιας κλάσης
- Η συνάρτηση tetragono() που ακολουθεί επιστρέφει ένα αντικείμενο – ορθογώνιο παραλληλόγραμμο κλάσης rectangle με ίσες πλευρές και εμβαδό όσο η τιμή της παραμέτρου με την οποία καλείται. Οι πλευρές του πρέπει να έχουν τιμή όσο η τετραγωνική ρίζα της παραμέτρου. Ο τύπος της συνάρτησης θα πρέπει να συμφωνεί με τον τύπο του αντικειμένου που επιστρέφει.

```
 rectangle tetragono (float emv)
{
  rectangle sq;
  sq.set_ab(sqrt(emv), sqrt(emv));
  return sq;
}
 int main()
 {
   rectangle rec1;
   rec1 = tetragono(100);
   cout << “Εμβαδό rec1 = “
            << rec1.emvado() << endl;
  return 0;
 }
```

- Η κλήση της συνάρτησης tetragono(100) επιστρέφει ως τιμή ένα αντικείμενο ορθογώνιο παραλληλόγραμμο με διαστάσεις 10x10. Η πρόταση rec1 = tetragono(100) καταχωρίζει τις τιμές των μεταβλητών – μελών του προσωρινού αντικειμένου που επιστρέφει η συνάρτηση στις αντίστοιχες μεταβλητές - μέλη του αντικειμένου rec1 της ίδιας κλάσης
- Παρά το γεγονός πως ο τύπος της συνάρτησης είναι rectangle η συνάρτηση tetragono() δεν είναι μέλος της κλάσης rectangle
- **Ο τύπος μιας συνάρτησης που επιστρέφει ως τιμή ένα αντικείμενο πρέπει να είναι ίδιος με την κλάση του αντικειμένου. Όταν επιστρέφει η συνάρτηση, δημιουργείται ένα προσωρινό αντικείμενο, του οποίου η τιμή θα πρέπει να ανατεθεί σε ένα άλλο αντικείμενο της ίδιας κλάσης για να μη χαθεί.**

Φίλιες (friend) συναρτήσεις και κλάσεις

- Πρόσβαση στα ιδιωτικά μέλη μιας κλάσης έχουν μόνο οι μέθοδοι της κλάσης
- Οποιαδήποτε συνάρτηση ενός προγράμματος μπορεί να γίνει φίλια συνάρτηση μιας κλάσης και να έχει πρόσβαση στα ιδιωτικά της μέλη, αρκεί να δηλωθεί μέσα στην κλάση (όπως και οι μέθοδοι της κλάσης) με το πρόθεμα **friend**
- Ας υποθέσουμε ότι θέλουμε να φτιάξουμε μια συνάρτηση η οποία δεν θα ανήκει στην κλάση rectangle και θα μηδενίζει τις διαστάσεις των αντικειμένων – ορθογωνίων παραλληλογράμμων αυτής της κλάσης, αναθέτοντας στις ιδιωτικές μεταβλητές plevra_a, plevra_b την τιμή 0. Η συνάρτηση αυτή θα πρέπει να δηλωθεί ως φίλια συνάρτηση της κλάσης rectangle για να μπορεί να έχει πρόσβαση στα ιδιωτικά της μέλη:

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

class rectangle
{
        float plevra_a;
        float plevra_b;
public:
        rectangle(float a, float b);
        rectangle();
        float emvado();
        void set_ab(float a, float b);
        friend void to_zero();
}rec1;
```

```
rectangle::rectangle(float a, float b)
{
        plevra_a = a;
        plevra_b = b;
}

rectangle::rectangle()
{
        plevra_a = 0;
        plevra_b = 0;
}

float rectangle::emvado()
{
    return plevra_a * plevra_b;
}

void rectangle::set_ab(float a, float b)
{
   plevra_a=a;
   plevra_b=b;
}

void to_zero()
{
        rec1.plevra_a=0;
        rec1.plevra_b=0;
}
```

```
int main()
{
        rec1.set_ab(10,2);
        cout  << "Εμβαδό rec1 πριν = "
                        << rec1.emvado() << endl;
        to_zero();
        cout  << "Εμβαδό rec1 μετά = "
                        << rec1.emvado() << endl;
    return 0;
}
```

- Συνήθως στις φίλιες συναρτήσεις μεταβιβάζουμε ως παράμετρο ένα ή περισσότερα αντικείμενα της κλάσης στην οποία έχουν οριστεί ως φίλιες
- Στον παρακάτω κώδικα, στη συνάρτηση to_zero() μεταβιβάζεται κατ’ αναφορά ένα αντικείμενο τύπου rectangle

```
#include <iostream>
using namespace std;
class rectangle
{
        float plevra_a;
        float plevra_b;
public:
        rectangle(float a, float b);
        rectangle();
        float emvado();
        void set_ab(float a, float b);
        friend void to_zero(rectangle &rec);
};
```

```
rectangle::rectangle(float a, float b)
{
		plevra_a = a;
		plevra_b = b;
}

rectangle::rectangle()
{
		plevra_a = 0;
		plevra_b = 0;

}

float rectangle::emvado()
{
    return plevra_a * plevra_b;
}

void rectangle::set_ab(float a, float b)
{
   plevra_a=a;
   plevra_b=b;
}

void to_zero(rectangle &rec)
{
		rec.plevra_a=0;
		rec.plevra_b=0;
}
```

```
int main()
{
   rectangle rec1;
            rec1.set_ab(8,7);
            cout      << "Εμβαδό rec1 πριν = "
                              << rec1.emvado() << endl;
            to_zero(rec1);
            cout      << "Εμβαδό rec1 μετά= "
                              << rec1.emvado() << endl;
  return 0;
}
```

Εμβαδό rec1 πριν = 56
Εμβαδό rec1 μετά = 0

- Η συνάρτηση to_zero() πρέπει να δηλωθεί ως φίλια της κλάσης rectangle για να μπορεί να έχει πρόσβαση στα μέλη plevra_a και plevra_b
- Η παράμετρος της συνάρτησης to_zero() πρέπει να είναι αναφορική για να μπορεί η συνάρτηση να μεταβάλλει τις τιμές του αντικειμένου – ορίσματος με το οποίο καλείται

- Φίλιες κλάσεις

Μια κλάση μπορεί να δηλωθεί ως φίλια μιας άλλη κλάσης με τον ίδιο τρόπο όπως δηλώνεται μια φίλια συνάρτηση της κλάσης

Όλες οι μέθοδοι μιας φίλιας κλάσης γίνονται φίλιες συναρτήσεις της αρχικής κλάσης και επομένως έχουν πρόσβαση στα ιδιωτικά της μέλη

Στο παράδειγμα που ακολουθεί η κλάση Α δηλώνεται ως φίλια κλάση της Β

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

class B
{
   friend class A;
private:
   int i;
   void display()
   {
      cout<<i<<endl;
   }

};

class A
{
public:
   void test(B obj_B)
   {
      obj_B.i = 10;
      obj_B.display();
   }
};

int main()
{
   B o1;
   A o2;
   o2.test(o1);
   return 0;
}
```

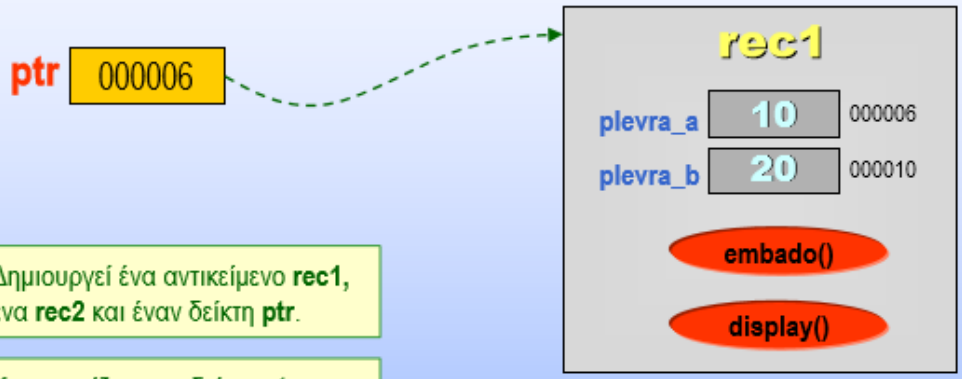
Δείκτες σε αντικείμενα

- Ένας δείκτης μπορεί να δείχνει σε οποιονδήποτε τύπο δεδομένων
- Η δήλωση ενός δείκτη σε αντικείμενα μιας κλάσης αλλά και η απόδοση τιμής σε αυτόν τον δείκτη γίνεται με τη χρήση των τελεστών * και &
- **Η πρόσβαση στα μέλη ενός αντικειμένου μέσω ενός δείκτη προς αυτό το αντικείμενο γίνεται με τη χρήση του τελεστή βέλους ->**

Δείκτες σε αντικείμενα και πρόσβαση στα μέλη μέσω δείκτη με χρήση του τελεστή αναφοράς *

Μια μεταβλητή δείκτη μπορεί να περιέχει τη **διεύθυνση** ενός **αντικειμένου**. Τότε λέμε ότι ο δείκτης **δείχνει** στο συγκεκριμένο αντικείμενο.

```
class rectangle
{
public:
    float plevra_a;
    float plevra_b;
    float emvado();
    void display();
};
.....
rectangle rec1,rec2,*ptr;
ptr = &rec1;
.....
(*ptr).plevra_a=10;
(*ptr).plevra_b=20;
(*ptr).display();
rec2=*ptr;
```



Δημιουργεί ένα αντικείμενο **rec1**, ένα **rec2** και έναν δείκτη **ptr**.

Καταχωρίζει στον δείκτη **ptr** τη διεύθυνση του **rec1**.

rec2

Η πρόσβαση στις μεταβλητές και συναρτήσεις-μέλη του αντικειμένου **rec1** μέσω του δείκτη **ptr** γίνεται με χρήση του **τελεστή αναφοράς** (*):
Η πρόταση **(*ptr).plevra_a=10** είναι ισοδύναμη με την πρόταση **rec1.plevra_a=10** και
Η πρόταση **(*ptr).display()** είναι ισοδύναμη με την πρόταση **rec1.display()**.
Η πρόταση **rec2=*ptr** είναι ισοδύναμη με την **rec2=rec1**!

Η πρόσβαση στο αντικείμενο μέσω ενός δείκτη που δείχνει στο αντικείμενο μπορεί να γίνει μέσω του τελεστή αναφοράς *. Η πρόταση ***ptr** είναι ισοδύναμη με τη πρόταση **rec1**!

```
int main()
{
  rectangle rec1;
  rectangle *ptr1;
  ptr1=&rec1;
  ptr1->set_ab(10,2);
  ptr1->xroma="κόκκινο";
  cout <<"Εμβαδό =" ptr1->embado() <<endl;
  return 0;
}
```

- Τα αντικείμενα μιας κλάσης έχουν μια συγκεκριμένη διεύθυνση και ένα μέγεθος όπως και οι μεταβλητές οποιουδήποτε άλλου τύπου δεδομένων. Η διεύθυνση ενός αντικειμένου είναι η διεύθυνση που καταλαμβάνει η πρώτη μεταβλητή μέλος του. Το μέγεθος ενός αντικειμένου είναι το άθροισμα των byte των μεταβλητών μελών του. Η διεύθυνση αποδίδεται από τον τελεστή & και το μέγεθος από τον sizeof

- Υποθέτουμε ότι η διεύθυνση του αντικειμένου rec1 είναι 8000 και ότι ο δείκτης ptr1 χρησιμοποιείται για την πρόσβαση σε αυτό το αντικείμενο. Επίσης η κλάση rectangle περιέχει 3 μεταβλητές των 4 bytes

&rec1 -> 8000

sizeof rec1 -> 12

sizeof (rectangle) -> 12

- Ο τελεστής βέλους -> χρησιμοποιείται για την πρόσβαση στα μέλη του αντικειμένου μέσω του δείκτη ptr1. Για παράδειγμα, η πρόταση ptr1->set_ab(10,2) εφαρμόζει τη μέθοδο set_ab() στο αντικείμενο που δείχνει ο δείκτης ptr1 καταχωρίζοντας σε αυτό τις διαστάσεις 10 και 2

Ο δείκτης **this**

- Μια μέθοδος καλείται πάντα σε συνδυασμό με κάποιο αντικείμενο με χρήση του τελεστή τελείας (.)
- Όταν εφαρμόζεται μια μέθοδος, στη μέθοδο μεταβιβάζεται αυτόματα ένας δείκτης με όνομα this ο οποίος δείχνει στο ίδιο το αντικείμενο στο οποίο εφαρμόστηκε η μέθοδος
- Ο δείκτης this έχει εμβέλεια μόνο μέσα στη μέθοδο και μπορεί να χρησιμοποιηθεί από τον κώδικα της μεθόδου για να έχει έμμεση πρόσβαση στα μέλη του αντικειμένου στο οποίο εφαρμόστηκε

```
void rectangle :: set_ab(float a, float b)
{
  this ->plevra_a = a;
  plevra_b = b;
}
```

- Στη περίπτωση απλών μεθόδων η χρήση του δείκτη **this** δεν είναι απαραίτητη αλλά ούτε και χρήσιμη. Στις μεθόδους υπερφόρτωσης τελεστών αρκετές φορές είναι αναπόφευκτη
- Η τιμή του δείκτη **this** δεν μπορεί να αλλάξει

Μέθοδοι δόμησης αντιγράφου

- Στη C++ τα νέα αντικείμενα δημιουργούνται με 2 τρόπους. Με τον άμεσο, όταν εκτελείται η πρόταση δήλωσης του αντικειμένου και με τον έμμεσο: κάθε φορά που ένα αντικείμενο μεταβιβάζεται κατ’ αξία ως όρισμα σε μια συνάρτηση η C++ δημιουργεί μέσα στη συνάρτηση ένα αντίγραφο του αντικειμένου – ορίσματος. Επίσης όταν μια συνάρτηση επιστρέφει ως τιμή ένα αντικείμενο η C++ δημιουργεί ένα προσωρινό αντικείμενο αντίγραφο του αντικειμένου που επιστρέφει η συνάρτηση

Μέθοδοι δόμησης αντιγράφου

- Στη περίπτωση που έχει οριστεί κάποια μέθοδος δόμησης για μια κλάση, αυτή εφαρμόζεται αυτόματα κάθε φορά που δημιουργείται ένα αντικείμενο με άμεσο τρόπο, δηλαδή με μια πρόταση δήλωσης. Στη περίπτωση που δημιουργείται ένα αντικείμενο με έμμεσο τρόπο δεν καλείται η μέθοδος δόμησης (αν υπάρχει) αλλά μια ειδική μέθοδος: η **μέθοδος δόμησης αντιγράφου** (copy constructor) της κλάσης
- Σε όλες τις περιπτώσεις έμμεσης δημιουργίας ενός αντικειμένου, το νέο αντικείμενο είναι αντίγραφο κάποιου υπάρχοντος αντικειμένου
- Αν δεν έχει οριστεί κάποια άλλη μέθοδος δόμησης αντιγράφου η C++ χρησιμοποιεί την προκαθορισμένη μέθοδο δόμησης αντιγράφου η οποία αντιγράφει το ένα αντικείμενο στο άλλο bit προς bit. Η προκαθορισμένη αυτή μέθοδος είναι διαθέσιμη για κάθε κλάση

Μέθοδοι δόμησης αντιγράφου

- Αν και τις περισσότερες φορές η προκαθορισμένη μέθοδος δόμησης αντιγράφου δημιουργεί ένα πιστό αντίγραφο, υπάρχουν περιπτώσεις όπως όταν μεταβιβάζουμε ή επιστρέφουμε αντικείμενα μιας κλάσης τα οποία χρησιμοποιούν δυναμικά κατανεμημένη μνήμη, όπου μπορούν να δημιουργηθούν προβλήματα. Σε αυτή την περίπτωση είναι προτιμότερη η χρήση μιας εξειδικευμένης μεθόδου δόμησης και ότι της προκαθορισμένης
- Μια μέθοδος δόμησης αντιγράφου χρησιμοποιείται για να αρχικοποιήσει ένα νέο αντικείμενο από ένα υπάρχον αντικείμενο

Στατικά μέλη μιας κλάσης

- Τόσο οι μεταβλητές – μέλη όσο και οι μέθοδοι μιας κλάσης μπορούν να δηλωθούν ως στατικές με το προσδιοριστικό static στην αρχή της πρότασης δήλωσης τους
- Στατικές μεταβλητές – μέλη: δεσμεύουν μνήμη μόνο μια φορά και είναι κοινόχρηστες σε όλα τα αντικείμενα της κλάσης. Μπορούν να είναι είτε δημόσιες είτε ιδιωτικές
- Στις στατικές μεταβλητές μέλη αποθηκεύουμε συνήθως καθολικές πληροφορίες που αφορούν όλα τα αντικείμενα της κλάσης

Στατικά μέλη μιας κλάσης

- Υποθέτουμε ότι στην κλάση rectangle θέλουμε να τηρούμε το πλήθος των αντικειμένων ορθογωνίων παραλληλογράμμων που έχουμε δημιουργήσει αλλά και τη συνολική τους επιφάνεια. Δεν θα είχε νόημα αν χρησιμοποιούσαμε δυο κανονικές μεταβλητές μέλη, αφού αυτές θα επαναλαμβάνονταν σε κάθε αντικείμενο. Για αυτό τον σκοπό δηλώνουμε τις δυο στατικές μεταβλητές μέλη plithos και epifania. Οι μεταβλητές αυτές καταλαμβάνουν χώρο στη μνήμη μόνο μια φορά για ολόκληρη την κλάση και είναι κοινόχρηστες από όλα τα αντικείμενα της κλάσης

Στατικά μέλη μιας κλάσης

- Κάθε αντικείμενο της κλάσης rectangle έχει πρόσβαση στα στατικά μέλη plithos και epifania όπως και στις κανονικές μεταβλητές. Η διαφορά είναι ότι με τις προτάσεις rec1.plithos και rec2.plithos αναφερόμαστε στην ίδια θέση μνήμης της στατικής μεταβλητής μέλους plithos που είναι κοινή για όλη τη κλάση. Στο παρακάτω πρόγραμμα η κλάση rectangle αποκτά τα δυο δημόσια στατικά μέλη και οι μέθοδοι δόμησης και αποδόμησης τροποποιούνται κατάλληλα ώστε να ενημερώνουν ανάλογα αυτές τις στατικές μεταβλητές μέλη

```
#include <iostream>
#include <string>
using namespace std;
class rectangle
{
        float plevra_a;
        float plevra_b;
public:
   string xroma;
   static int plithos;
   static float epifania;
        rectangle(float a, float b);
        rectangle();
        ~rectangle();
        float emvado() {return plevra_a * plevra_b;}
        void set_ab(float a, float b) {plevra_a=a; plevra_b=b;}
        void show();

};
```

```
rectangle::rectangle(float a, float b)
{
        plevra_a = a;
        plevra_b = b;
        plithos++;
        epifania=epifania+emvado();
}
rectangle::rectangle()
{
        plevra_a = 0;
        plevra_b = 0;
        plithos++;
        epifania=epifania+emvado();
}
rectangle::~rectangle()
{
        plithos--;
        epifania=epifania-emvado();
}

int rectangle::plithos=0;
float rectangle::epifania=0;
```

```
int main()
{
        rectangle rec1(5,8),rec2(4,7),rec3;
        cout << "Πλήθος :"<<rectangle::plithos;
        cout << "  Επιφάνεια :"<<rectangle::epifania<<endl;
        {
   rectangle rec4(10,3);
        cout << "Πλήθος :"<<rectangle::plithos;
        cout << "  Επιφάνεια :"<<rectangle::epifania<<endl;
   }
        cout << "Πλήθος :"<<rectangle::plithos;
        cout << "  Επιφάνεια :"<<rectangle::epifania<<endl;
        return 0;
}
```

```
Πλήθος = 3 Επιφάνεια = 68
Πλήθος = 4 Επιφάνεια = 98
Πλήθος = 3 Επιφάνεια = 68
```

- Όταν οι στατικές μεταβλητές μέλη είναι ιδιωτικές, η πρόσβαση σε αυτές γίνεται μόνο με τη χρήση δημόσιων μεθόδων και όχι άμεσα μέσω των αντικειμένων ή απευθείας από τη κλάση. Ο παρακάτω κώδικας χρησιμοποιεί τις δυο στατικές μεταβλητές μέλη ως ιδιωτικά μέλη της κλάσης rectangle

```
#include <iostream>
#include <string>
using namespace std;
class rectangle
{
        float plevra_a;
        float plevra_b;
        static int plithos;
   static float epifania;
public:
   string xroma;
        rectangle(float a, float b);
        rectangle();
        float emvado();
        void set_ab(float a, float b);
        void show();
        void show_total();

};
```

```
float rectangle::emvado()
{
        return plevra_a * plevra_b;
}
void rectangle::set_ab(float a, float b)
{
        plevra_a = a;
        plevra_b = b;
}
void rectangle::show()
{
        cout        <<"REC : "
                        << plevra_a << "x" << plevra_b << " Χρώμα:"<<xroma<<endl;
}
void rectangle::show_total()
{
        cout        <<"Πλήθος="
                        << plithos << "  Επιφάνεια=" << epifania<<endl;
}
rectangle::rectangle(float a, float b)
{
        plevra_a = a;
        plevra_b = b;
        plithos++;
        epifania=epifania+emvado();
}
rectangle::rectangle()
{
        plevra_a = 0;
        plevra_b = 0;
        plithos++;
        epifania=epifania+emvado();
}
```

```
int rectangle::plithos=0;
float rectangle::epifania=0;

int main()
{
        rectangle rec1(5,8),rec2(4,7),rec3;
   rec1.show_total();
   rec2.show_total();
   return 0;
}
```

```
Πλήθος=3 Επιφάνεια = 68
Πλήθος=3 Επιφάνεια = 68
```

- Στις ιδιωτικές στατικές μεταβλητές μέλη έχουμε πρόσβαση από οποιοδήποτε αντικείμενο της κλάσης αλλά μόνο με τη χρήση μιας δημόσιας μεθόδου. Από τη στιγμή που για να χρησιμοποιήσουμε μια δημόσια μέθοδο πρέπει να έχουμε ένα τουλάχιστον αντικείμενο δεν είναι δυνατή η πρόσβαση στις ιδιωτικές στατικές μεταβλητές μέλη απευθείας από την κλάση παρά μόνο μέσω ενός αντικειμένου της κλάσης

Συνοπτικά:

- Οι στατικές μεταβλητές – μέλη πρέπει να δηλωθούν ως καθολικές μεταβλητές της κλάσης για την οποία ορίζονται
- Η μνήμη για τις στατικές μεταβλητές – μέλη δεσμεύεται ακόμα και αν η κλάση δεν έχει κανένα αντικείμενο
- Σε όλες τις αριθμητικές στατικές μεταβλητές – μέλη ανατίθεται αρχική τιμή 0 εκτός αν ορίζεται διαφορετική αρχική τιμή στην πρόταση ορισμού τους

Στατικές μέθοδοι κλάσης

- Μια μέθοδος μπορεί να δηλωθεί ως στατική με το πρόθεμα static. Οι διαφορές μεταξύ στατικών και μη στατικών μεθόδων είναι οι ακόλουθες: 1) Μια στατική μέθοδος έχει πρόσβαση μόνο σε στατικά μέλη της κλάσης. Μια μη στατική μέθοδος έχει πρόσβαση τόσο στα στατικά όσο και στα μη στατικά μέλη της κλάσης 2)Μια στατική μέθοδος μπορεί να κληθεί ακόμη και αν η κλάση δεν έχει κανένα αντικείμενο. Μια μη στατική μέθοδος μπορεί να κληθεί μόνο σε σχέση με κάποιο αντικείμενο της κλάσης 3) Σε μια στατική μέθοδο δεν μεταβιβάζεται ο δείκτης **this**

Στατικές μέθοδοι κλάσης

- Δηλώνουμε μια μέθοδο ως στατική κυρίως όταν θέλουμε να έχουμε πρόσβαση σε αυτή, ακόμα και αν δεν έχει δημιουργηθεί κανένα αντικείμενο αυτής της κλάσης
- Στα διαγράμματα UML τα στατικά μέλη μιας κλάσης εμφανίζονται υπογραμμισμένα

```
#include <iostream>
#include <string>
using namespace std;
class rectangle
{
        float plevra_a;
        float plevra_b;
        static int plithos;
   static float epifania;
public:
   string xroma;
        rectangle(float a, float b);
        rectangle();
        float emvado();
        void set_ab(float a, float b);
        void show();
        static void show_total();
};
```

```
float rectangle::emvado()
{
        return plevra_a * plevra_b;
}
void rectangle::set_ab(float a, float b)
{
        plevra_a = a;
        plevra_b = b;
}
void rectangle::show()
{
        cout    <<"REC : "
                        << plevra_a << "x" << plevra_b << "
Χρώμα:"<<xroma<<endl;
}
void rectangle::show_total()
{
        cout    <<"Πλήθος="
                        << plithos << "  Επιφάνεια=" << epifania<<endl;
}
```

```
rectangle::rectangle(float a, float b)
{
        plevra_a = a;
        plevra_b = b;
        plithos++;
        epifania=epifania+emvado();
}
rectangle::rectangle()
{
        plevra_a = 0;
        plevra_b = 0;
        plithos++;
        epifania=epifania+emvado();
}

int rectangle::plithos=0;
float rectangle::epifania=0;

int main()
{
        rectangle rec1(5,8),rec2(4,7),rec3;
  rectangle::show_total();
  return 0;
}
```

Πλήθος = 3 Επιφάνεια =68

Στατικές μεταβλητές – μέλη μόνο για ανάγνωση

- Στη παρακάτω κλάση circle η μεταβλητή μέλος pi δηλώνεται ως ιδιωτική, στατική και με το πρόθεμα const μόνο για ανάγνωση. Ουσιαστικά με τον τρόπο αυτό δηλώνεται μια σταθερά με εμβέλεια σε όλες τις μεθόδους μιας κλάσης

```
#include <iostream>
#include <cmath>
using namespace std;
class circle
{
        static const double pi;
public:

        float aktina;
        circle(float r) {aktina=r;};
        float emvado() {return pi*pow(aktina,2);};
        float perimetros() {return 2*pi*aktina;};
};
const double circle::pi=3.14159265359;
int main()
{
        circle c1(5),c2(10);
   cout << c1.emvado()<<endl;
   cout << c2.perimetros()<<endl;
   return 0;
}
```

```
78.5398
62.8319
```

Η κλάση string

- Η κλάση string διατίθεται από την καθιερωμένη βιβλιοθήκη της C++ και δίνει τη δυνατότητα στη C++ να χειρίζεται τις συμβολοσειρές ως αντικείμενα
- Ένα αντικείμενο της κλάσης string μπορεί να αποθηκεύσει σύνολα χαρακτήρων με δυναμικό τρόπο
- Το μέγεθος ενός τέτοιου αντικειμένου καθορίζεται κατά το στάδιο της εκτέλεσης (run time) και όχι από το στάδιο της μεταγλώττισης όπως στη C
- Επιπλέον με τα αντικείμενα της κλάσης string μπορούν να χρησιμοποιηθούν οι τελεστές ανάθεσης, σύγκρισης και πρόσθεσης με αποτέλεσμα ο χειρισμός τους να γίνεται πολύ πιο εύκολος

Η κλάση string

#include <string>

string s1 = “Η C++”; ή **string** s1 (“Η C++”);

string s1, s2;

s2 = “ Γλώσσα C++”;

s3 = s1 +“ ” + s3;

cout << s3 << endl;

if (s1>s2) cout << “ΝΑΙ” << endl;

Η κλάση string

- Η χρήση των τελεστών ανάθεσης, πρόσθεσης και σύγκρισης από αντικείμενα της κλάσης string είναι εφικτή διότι οι τελεστές αυτοί έχουν υπερφορτωθεί για αυτή την κλάση ώστε να επιτελούν τις συγκεκριμένες λειτουργίες

Διαχείριση αντικειμένων της κλάσης string

Η διαχείριση των αντικειμένων της κλάσης string γίνεται μέσω των μεθόδων αυτής τη κλάσης. Στον πίνακα που ακολουθεί αναφέρονται διάφορες λειτουργίες που αφορούν αντικείμενα της κλάσης αυτής και τον τρόπο υλοποίησης. Υποθέτουμε ότι ισχύουν οι παρακάτω δηλώσεις:

```
string s, s1, s2;
char c, *cs;
int i, start, len, newSize;
bool b;
```

Τύπος	Σύνταξη	Περιγραφή
Δημιουργία Αντικειμένων κλάσης string		
	string s;	Δημιουργεί ένα αντικείμενο s της κλάσης string
	string s (s1);	Δημιουργεί ένα αντικείμενο s της κλάσης string, με αρχική τιμή το string s1
	string s(cs); string s (“C++”);	Δημιουργεί ένα αντικείμενο s κλάσης string με αρχική τιμή τη συμβολοσειρά που δείχνει ο cs ή τη συμβολοσειρά C++

Καταχώρηση, αλλαγή στοιχείων

s1 = s2;	Καταχωρίζει το s2 στο s1
s1 = cs;	Καταχωρίζει τη συμβολοσειρά που δείχνει ο cs στο s1
s1 = c;	Καταχωρίζει τον χαρακτήρα c στο s1
s[i] = c;	Καταχωρίζει τον χαρακτήρα c στη θέση i του συνόλου χαρακτήρων του s. Η πρώτη θέση ενός συνόλου χαρακτήρων είναι η θέση 0
s.at(i) = c;	Όπως η προηγούμενη με τη διαφορά ότι αν το i είναι εκτός των ορίων του string s δημιουργεί την εξαίρεση out_of_range
s.append(s2);	Προσθέτει το string s2 στο τέλος του s. (s = s+s2)
s.append(cs);	s = s + cs
s.assign(s2, start, len);	Καταχωρίζει το string s, len χαρακτήρες από το s2 ξεκινώντας από τη θέση start
s.clear();	Διαγράφει όλους τους χαρακτήρες από το s
s.replace(start, len, s1);	Αντικαθιστά len χαρακτήρες του s με τους χαρακτήρες του s1 ξεκινώντας από τη θέση start
s.insert(start, s2);	Παρεμβάλλει τους χαρακτήρες του s2 μέσα στο s ξεκινώντας από τη θέση start
s.erase(start, len);	Απομακρύνει len χαρακτήρες του s ξεκινώντας από τη θέση start
s.erase(start);	Απομακρύνει όλους τους χαρακτήρες του s ξεκινώντας από τη θέση strart
s1.swap(s2)	Αντιμεταθέτει τα περιεχόμενα των s1 και s2

Προσπέλαση		
*char	s.c_str();	Επιστρέφει τους ίδιους χαρακτήρες όπως το s τερματισμένους με τον χαρακτήρα null
string	s.substr(start, len);	Επιστρέφει len χαρακτήρες από το s ξεκινώντας από τη θέση start. Στη περίπτωση που παραληφθεί η δεύτερη παράμετρος επιστρέφει όλους τους χαρακτήρες από τη θέση start μέχρι το τέλος
char	s[i];	Επιστρέφει τον χαρακτήρα της i θέσης του s
char	s.at(1);	Επιστρέφει τον χαρακτήρα της i θέσης με τη διαφορά ότι στην περίπτωση που ο i είναι εκτός ορίων του s δημιουργεί την εξαίρεση out_of_range

Μέγεθος		
int	s.length();	Επιστρέφει το πλήθος των χαρακτήρων του αντικειμένου s
int	s.size();	Ίδια λειτουργία όπως η προηγούμενη
int	s.capacity();	Επιστρέφει το πλήθος χαρακτήρων που μπορεί να αποθηκεύσει το string s χωρίς ανακατανομή μνήμης
bool	s.empty();	Επιστρέφει τιμή true αν το αντικείμενο s δεν περιέχει κανένα χαρακτήρα
int	s.resize(newSize, c);	Αλλάζει το μέγεθος του s σε newSize συμπληρώνοντας τα κενά (αν χρειάζεται) με τον χαρακτήρα c
Εύρεση – Επιστρέφει ως τιμή τη θέση του εντοπισμού (string::npos – σε περίπτωση μη εντοπισμού)		
int	s.find(c);	Εντοπίζει την πρώτη εμφάνιση του c μέσα στο s
int	s.find(s1);	Εντοπίζει την πρώτη εμφάνιση του s1 μέσα στο s
int	s.rfind(s1);	Εντοπίζει την τελευταία εμφάνιση του s1 μέσα στο s
int	s.find_first_of(s1);	Εντοπίζει τον πρώτο χαρακτήρα του s ο οποίος περιλαμβάνεται στο σύνολο χαρακτήρων του s1
int	s.find_first_not_of(s1);	Εντοπίζει τον πρώτο χαρακτήρα του s ο οποίος δεν περιλαμβάνεται στο σύνολο χαρακτήρων του s1
int	s.find_last_of(s1);	Εντοπίζει τον τελευταίο χαρακτήρα του s ο οποίος περιλαμβάνεται στο σύνολο χαρακτήρων του s1
int	s.find_last_not_of(s1);	Εντοπίζει τον τελευταίο χαρακτήρα του s ο οποίος δεν περιλαμβάνεται στο σύνολο χαρακτήρων του s1

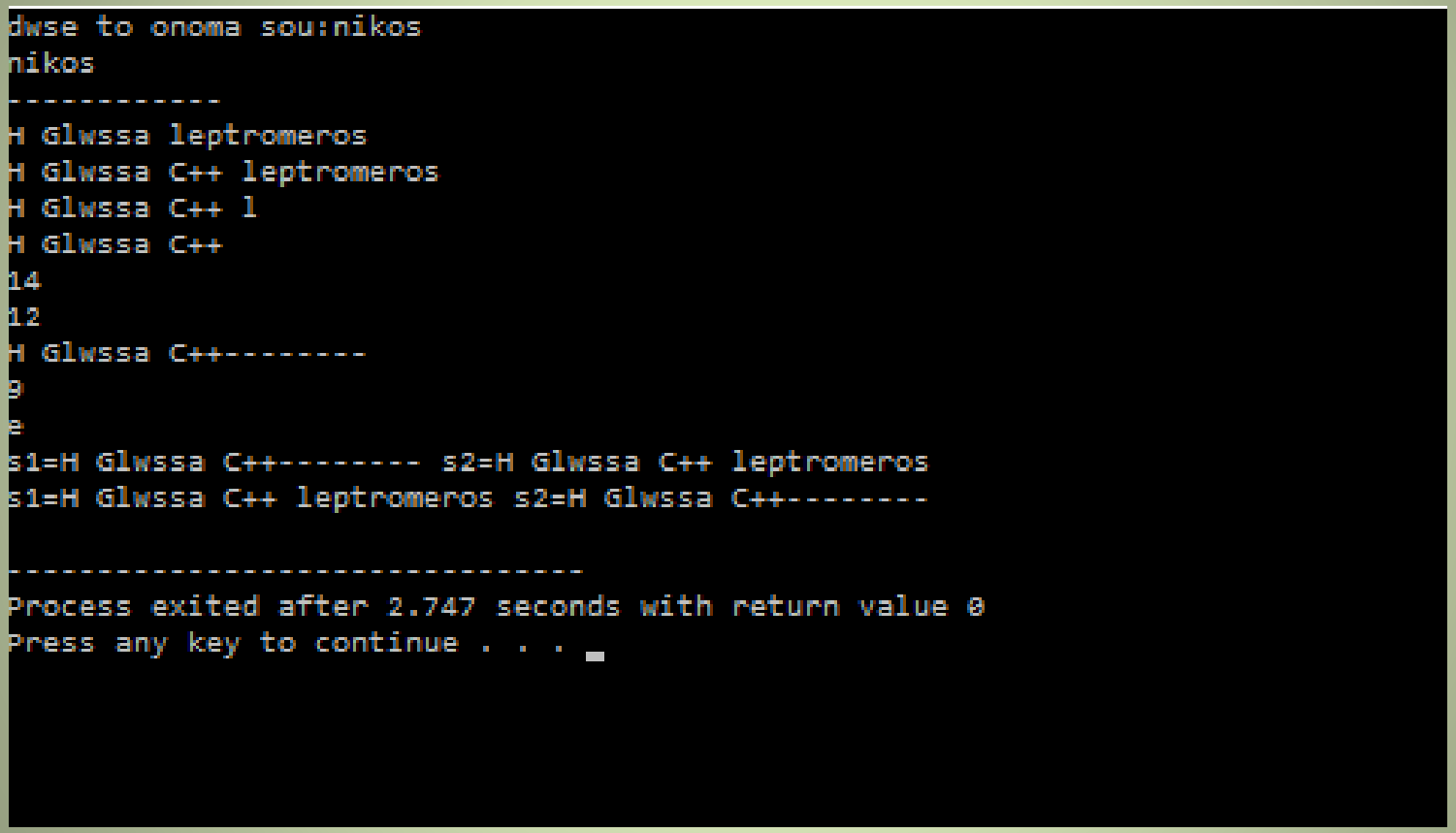
Σύγκριση		
int	s1.compare(s2);	Συγκρίνει αλφαβητικά τα s1 και s2 και επιστρέφει -1 αν s1<s2, 0 εάν s1=s2 και 1 εάν s1>s2
bool	s1 == s2 Επίσης > < >= <= !=	Οι τελεστές σύγκρισης έχουν υπερφορτωθεί για την κλάση string μπορούν να χρησιμοποιηθούν για τη σύγκριση αντικειμένων της
	Είσοδος / Έξοδος	
	cin >> s;	Καταχωρίζει στο string s το σύνολο χαρακτήρων που εισάγεται από το πληκτρολόγιο μέχρι να πατηθεί ένας λευκός χαρακτήρας
	getline(cin, s);	Διαβάζει από το πληκτρολόγιο οτιδήποτε μέχρι τον επόμενο χαρακτήρα αλλαγής γραμμής και το καταχωρίζει στο s
	cout << s;	Στέλνει τα περιεχόμενα του s στην οθόνη

- Το παρακάτω πρόγραμμα δείχνει τον τρόπο διαχείρισης των αντικειμένων της κλάσης string μέσω των μεθόδων που διαθέτει

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
        string s("C++ ");
        string s1="123456789", s2;
        int i;
        cout<<"Δώσε το όνομά σου:";
        getline(cin,s1);
        //Εμφανίζει το όνομα που πληκτρολογήθηκε
        cout<<s1<<endl<<"----------"<<endl;
        s2="Η γλώσσα ";
        s2.append("λεπτομερώς");  //Προσθέτει χαρακτήρες στο τέλος του s2
        cout<<s2<<endl;
        s2.insert(9,s);          //Παρεμβάλει τα περιεχόμενα του s στην 9η θέση του s2
        cout<<s2<<endl;
        s1=s2.substr(0,14);      //Καταχώριση των 14ων πρώτων χαρακτήρων του s2 στο s1
        cout<<s1<<endl;
        s1.erase(12);            //Διαγραφή των χαρακτήρων του s1 από τη θέση 12 μέχρι τέλος
        cout<<s1<<endl;
        cout<<s1.capacity()<<endl;          //Εμφανίζει τη χωρητικότητα του s1
```

```
        cout<<s1.size()<<endl;       //Εμφανίζει το πλήθος χαρακτήρων του s1
        //Μεταβάλει το μέγεθος του s1 σε 20 χαρακτήρες, γεμίζοντας τα κενά με παύλες '-'
        s1.resize(20,'-');
        cout<<s1<<endl;
        i=s2.find(s);          //Εντοπίζει τη θέση του s μέσα στο s2
        cout<<i<<endl;
        //Εντοπίζει τη θέση του πρώτου χαρακτήρα του s2 οποίος δεν υπάρχει στο s1
        i=s2.find_first_not_of(s1);
        cout << s2[i]<<endl; //Εμφανίζει αυτόν τον χαρακτήρα
        //Εμφανίζει τα περιεχόμενα του s1 και του s2
        cout<<"s1="<<s1<<" "<<"s2="<<s2<<endl;
        s1.swap(s2); //Αντιμεταθέτει τα περιεχόμενα των s1 και s2
        //Εμφανίζει τα περιεχόμενα του s1 και του s2
        cout<<"s1="<<s1<<" "<<"s2="<<s2<<endl;
        return 0;
}
```

- Τα αποτελέσματα του προηγούμενου προγράμματος:



- Το πρόγραμμα που ακολουθεί γεμίζει έναν πίνακα αντικειμένων κλάσης string με 10 φράσεις που πληκτρολογεί ο χρήστης και έπειτα εμφανίζει τον πρώτο και τον τελευταίο χαρακτήρα κάθε φράσης. Την εμφάνιση των χαρακτήρων αναλαμβάνει η συνάρτηση display() στην οποία μεταβιβάζεται ο πίνακας

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

void display(string o[]);

int main()
{
        string onom[10];
        int i;
        for (i=0;i<10;i++)
    getline(cin,onom[i]);
  display(onom);
        return 0;
}

void display(string o[])
{
  int i;
  for (i=0;i<10;i++)
    cout << o[i][0] << o[i][o[i].size()-1]<< endl;

}
```

Σύνθετες κλάσεις

- Μια μεταβλητή – μέλος μιας κλάσης μπορεί να είναι ένα αντικείμενο ή πίνακας αντικειμένων μιας οποιασδήποτε άλλης κλάσης
- Οι κλάσεις που περιέχουν τέτοιες μεταβλητές – μέλη λέγονται **σύνθετες κλάσεις** (compositions)
- Στο πρόγραμμα που ακολουθεί οι κλάσεις point και circle έχουν διαφοροποιηθεί ώστε να περιλαμβάνουν μεθόδους δόμησης και αποδόμησης. Επίσης στη κλάση point έχει προστεθεί η μέθοδος move() η οποία μετακινεί ένα σημείο σε μια νέα θέση επάνω στο καρτεσιανό επίπεδο και στη κλάση circle η μέθοδος info() η οποία εμφανίζει τα στοιχεία ενός αντικειμένου - κύκλου

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

class point
{
public:
		float x_pos;
		float y_pos;
		point();
		point(float x, float y);
		~point();
		void move(float dx, float dy);
};
point::point()
{
		x_pos=0;
		y_pos=0;
		cout<<"Ένα σημείο δημιουργήθηκε στο 0,0"<<endl;
}
point::point(float x, y)
{float
		x_pos=x;
		y_pos=y;
		cout<<"Ένα σημείο δημιουργήθηκε στο "<<x_pos<<","<<y_pos<<endl;
}
```

```
point::~point()
{
        cout<<"Το σημείο στο "<<x_pos<<","<<y_pos<<" καταστράφηκε"<<endl;
}
void point::move(float dx, float dy)
{
        x_pos=x_pos+dx;
        y_pos=y_pos+dy;
}
class circle
{
public:
        float aktina;
        point center;
        string xroma;
        circle();
        circle(float r);
        ~circle();
        void info();
};
circle::circle()
{
        aktina=0;
        xroma="";
        cout<<"Ένας κύκλος δημιουργήθηκε με ακτίνα "<<aktina<<endl;
}
```

```
circle::circle(float r)
{
        aktina=r;
        xroma="";
        cout<<"Ένας κύκλος δημιουργήθηκε με ακτίνα "<<aktina<<endl;
}
circle::~circle()
{
        cout<<"Ένας κύκλος με ακτίνα "<<aktina<<" καταστράφηκε"<<endl;
}
void circle::info()
{
        cout<<"Κύκλος "<<aktina<<" @ "<<center.x_pos<<","<<center.y_pos<<endl;
}

int main()
{
        circle c1,c2(10);
        c1.info();
        c2.center.x_pos=20;
        c2.center.y_pos=40;
        c2.info();
        c2.center.move(-5,30);
        c2.info();
        return 0;
}
```

Ένα σημείο δημιουργήθηκε στο 0,0
Ένας κύκλος δημιουργήθηκε με ακτίνα 0
Ένα σημείο δημιουργήθηκε στο 0,0
Ένας κύκλος δημιουργήθηκε με ακτίνα 10
Κύκλος 0 @ 0,0
Κύκλος 10 @ 20,40
Κύκλος 10 @ 15,70
Ένας κύκλος με ακτίνα 10 καταστράφηκε
Το σημείο στο 15,70 καταστράφηκε
Ένας κύκλος με ακτίνα 0 καταστράφηκε
Το σημείο στο 0,0 καταστράφηκε

Ένθετες Κλάσεις

- Μια ένθετη κλάση (nested class) είναι μια κλάση η οποία ορίζεται μέσα σε μια άλλη κλάση

```
class A
{
  public:
        class B
         {
            ..............
         };
        B obj1;
   }
```

Η κλάση B ορίζεται μέσα στην κλάση A. Είναι μια ένθετη κλάση της A και είναι διαθέσιμη μόνο μέσα στο χώρο εμβέλειας της A. Μέλη της κλάσης A μπορούν να είναι τύπου B και μέθοδοι της A μπορούν να χρησιμοποιούν την κλάση B

Το μέλος obj1 της κλάσης A είναι ένα αντικείμενο της κλάσης B

Ένθετες Κλάσεις

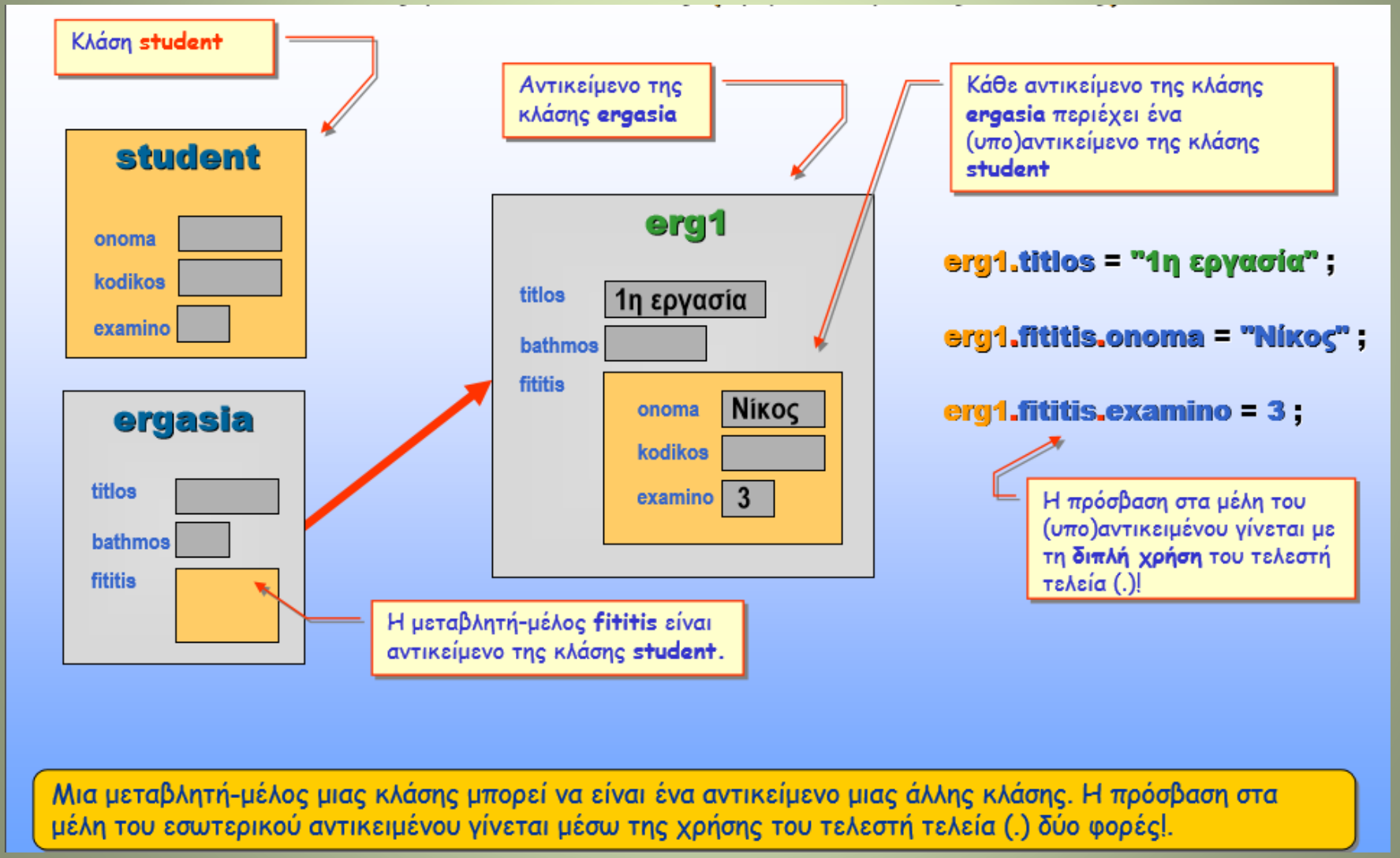
```
 int main()
{
   A o1;
   o1.obj.b1 = 4;
   ……..
   A :: B o2;
   o2.b1 = 5;
   ……..
 }
```

Το αντικείμενο o1 διαθέτει ένα αντικείμενο μέλος της κλάσης B. Στο μέλος b1 του obj1 καταχωρίζεται ο αριθμός 4

Δηλώνεται ένα αντικείμενο o2 κλάσης B η οποία ορίζεται μέσα στη κλάση A

Στο μέλος b1 του o2 καταχωρίζεται το 5

Ένθετες Κλάσεις



Ένθετες Κλάσεις

```
.....

class student
{
public:
    string onoma;
    string kodikos;
    int examino;
};

class ergasia
{
public:
    string titlos;
    float bathmos;
    student fititis;
};
.....
int main()
{
    ergasia first;
    first.titlos="1h ergasia";
    first.bathmos=8.5;
    first.fititis.onoma="Nikos";
    first.fititis.kodikos="ct08000";
    first.fititis.examino=3;
}
```

Η μεταβλητή-μέλος **fititis** είναι αντικείμενο της κλάσης **student**.

Η πρόσβαση στα μέλη του αντικειμένου **first** γίνεται με τη **απλή χρήση** του τελεστή τελεία (.)!

Η πρόσβαση στα μέλη του (υπο)αντικειμένου **fititis** γίνεται με τη **διπλή χρήση** του τελεστή τελεία (.)!

Ένθετες Κλάσεις

```
.....

class student
{
public:
    string onoma;
    string kodikos;
    int examino;
};

class ergasia
{
public:
    string titlos;
    float bathmos;
    student fititis[5];
};
.....
int main()
{
    ergasia first;
    first.titlos="1h ergasia";
    first.bathmos=8.5;
    .....
    first.fititis[2].onoma="Nikos";
    first.fititis[2].kodikos="ct08000";
    first.fititis[2].examino=3;
}
```

Κάθε εργασία μπορεί να γίνει το πολύ από 5 φοιτητές. Τώρα κάθε αντικείμενο κλάσης **ergasia** περιέχει 5 (υπο)αντικείμενα της κλάσης **student**!

Καταχώριση στοιχείων εργασίας

Καταχώριση στοιχείων για τον 3ο φοιτητή.

Ένθετες κλάσεις

- Στο παρακάτω παράδειγμα η κλάση point ορίζεται ως ένθετη της κλάσης circle. Η κλάση point ορίζεται ως ιδιωτική της circle οπότε είναι διαθέσιμη μόνο μέσα σε αυτή τη κλάση.
- Οι μέθοδοι της κλάσης circle παραμένουν ίδιες και επειδή είναι μέλη της έχουν φυσικά πρόσβαση στην ιδιωτική ένθετη κλάση point. Τώρα όμως η κλάση point είναι άγνωστη εκτός της circle και δεν μπορεί να χρησιμοποιηθεί από άλλες κλάσεις

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

class point
{
public:
		float x_pos;
		float y_pos;
		point();
		point(float x, float y);
		~point();
		void move(float dx, float dy);
};
point::point()
{
		x_pos=0;
		y_pos=0;
		cout<<"Ένα σημείο δημιουργήθηκε στο 0,0"<<endl;
}
point::point(float x, float y)
{
		x_pos=x;
		y_pos=y;
		cout<<"Ένα σημείο δημιουργήθηκε στο "<<x_pos<<","<<y_pos<<endl;
}
```

```
point::~point()
{
		cout<<"Το σημείο στο "<<x_pos<<","<<y_pos<<" καταστράφηκε"<<endl;
}
void point::move(float dx, float dy)
{
		x_pos=x_pos+dx;
		y_pos=y_pos+dy;
}
class circle
{
public:
		float aktina;
		point center;
		string xroma;
		circle();
		circle(float r);
		~circle();
		void info();
};
circle::circle()
{
		aktina=0;
		xroma="";
		cout<<"Ένας κύκλος δημιουργήθηκε με ακτίνα "<<aktina<<endl;
}
circle::circle(float r)
{
		aktina=r;
		xroma="";
		cout<<"Ένας κύκλος δημιουργήθηκε με ακτίνα "<<aktina<<endl;
}
```

```
circle::~circle()
{
	cout<<"Ένας κύκλος με ακτίνα "<<aktina<<" καταστράφηκε"<<endl;
}
void circle::info()
{
	cout<<"Κύκλος "<<aktina<<" @ "<<center.x_pos<<","<<center.y_pos<<endl;
}

int main()
{
	circle c1,c2(10);
	c1.info();
	c2.center.x_pos=20;
	c2.center.y_pos=40;
	c2.info();
	c2.center.move(-5,30);
	c2.info();
	return 0;
}
```

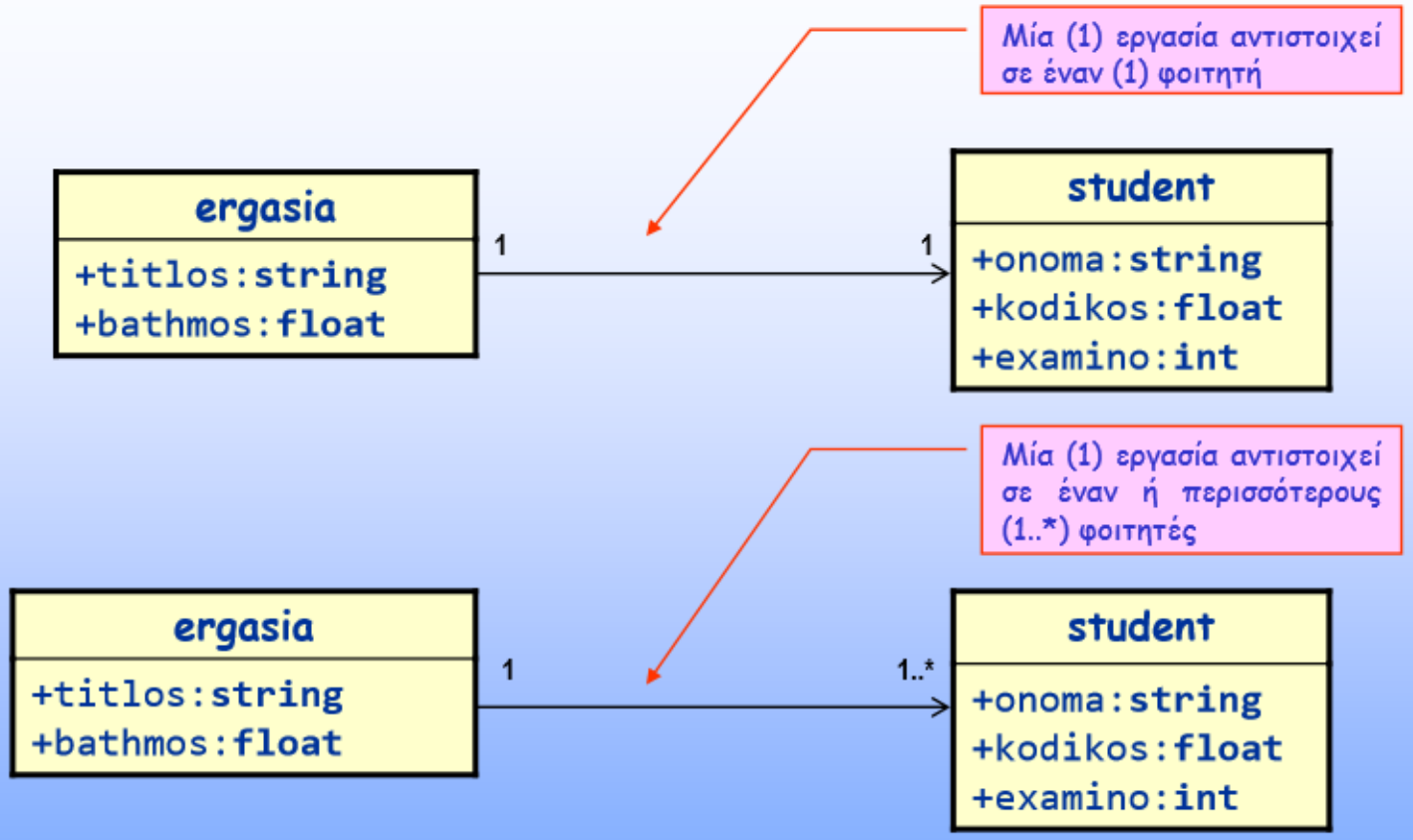
Προκαταβολική δήλωση κλάσης

- Κατά τον σχεδιασμό ενός προγράμματος αν θέλουμε να δηλώσουμε ένα αντικείμενο μιας κλάσης θα πρέπει πρώτα να έχουμε ορίσει την κλάση. Το ίδιο ισχύει και τα μέλη των κλάσεων. Σε διαφορετική περίπτωση θα δημιουργείτο πρόβλημα στον μεταγλωττιστή
- Η λύση είναι η προκαταβολική δήλωση (forward declaration) της κλάσης η οποία απλά γνωστοποιεί στον μεταγλωττιστή την ύπαρξή της. Ο ορισμός της κλάσης μπορεί να ακολουθεί σε οποιοδήποτε σημείο του προγράμματος. Η προκαταβολική δήλωση περιλαμβάνει μόνο το όνομα της κλάσης και όχι το σώμα της.

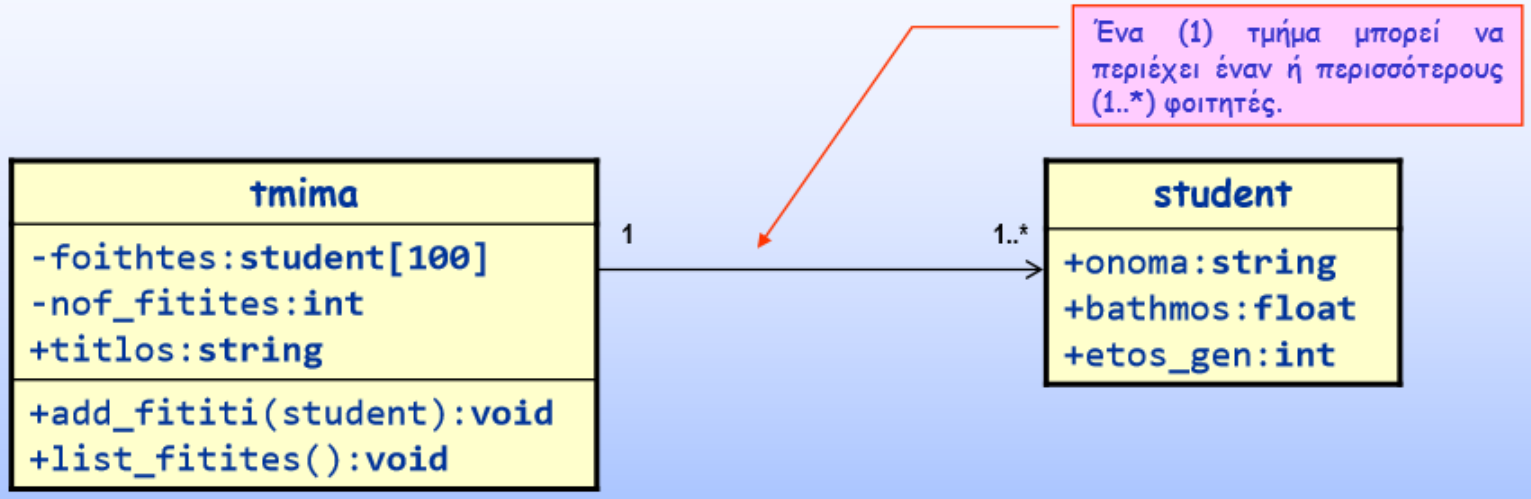
Συσχέτιση κλάσεων

- Τα διάφορα αντικείμενα (άρα και οι κλάσεις) είναι δυνατόν να σχετίζονται με διάφορους τρόπους μεταξύ τους
- Ο τρόπος αυτός συσχέτισης λέγεται συσχέτιση σύνθεσης (composition relationship) όπου ένα αντικείμενο αποτελεί τμήμα ενός πιο σύνθετου αντικειμένου. Σε αυτό τον τρόπο συσχέτισης τα δυο αντικείμενα συνδέονται με σχέση ζωής: όταν καταστραφεί ένα αντικείμενο της κλάσης δεν έχει νόημα να συνεχίσει να υπάρχει το μέλος – αντικείμενο της κλάσης που περιέχει

Απεικόνιση συσχέτισης σύνθεσης με UML



Στα διαγράμματα κλάσεων η συσχέτιση μεταξύ των κλάσεων συμβολίζεται με ένα απλό βέλος από τη μία κλάση στην άλλη.


Ένα (1) τμήμα μπορεί να περιέχει έναν ή περισσότερους (1..*) φοιτητές.
tmima
-foithtes:student[100]
-nof_fitites:int
+titlos:string
+add_fititi(student):void
+list_fitites():void
1
1..*
student
+onoma:string
+bathmos:float
+etos_gen:int

Λίστες αρχικοποίησης

- Οι λίστες αρχικοποίησης (initialization lists) είναι ένας διαφορετικός τρόπος ανάθεσης αρχικών τιμών στις μεταβλητές – μέλη μιας κλάσης από τις μεθόδους δόμησης της.
- Στο πρόγραμμα που ακολουθεί χρησιμοποιούνται λίστες αρχικοποίησης για την ανάθεση αρχικών τιμών από τις μεθόδους δόμησης. Η λίστα αρχικοποίησης παρεμβάλλεται αμέσως μετά από τις παραμέτρους μιας μεθόδου και πριν από το σώμα της, και ξεκινάει με μια άνω κάτω τελεία (:)

```
#include <iostream>
using namespace std;
class point
{
public:
        float x_pos;
        float y_pos;
        point();
        point(float x, float y);
};
point::point():x_pos(0),y_pos(0)
{
        cout<<"Ένα σημείο δημιουργήθηκε στο 0,0"<<endl;
}
point::point(float x, float y):x_pos(x), y_pos(y)
{
        cout<<"Ένα σημείο δημιουργήθηκε στο "<<x_pos<<","<<y_pos<<endl;
}

int main()
{
    point p1,p2(34,45);
    return 0;
}
```

```
Ένα σημείο δημιουργήθηκε στο 0,0
Ένα σημείο δημιουργήθηκε στο 34,45
```

Λίστες αρχικοποίησης

- Οι λίστες αρχικοποίησης περιλαμβάνουν τα ονόματα των μεταβλητών – μελών στις οποίες θέλουμε να αναθέσουμε αρχικές τιμές, διαχωρισμένα με ένα κόμμα. Οι αρχικές τιμές ορίζονται μέσα σε παρενθέσεις αμέσως μετά από το όνομα μεταβλητής – μέλους.

point :: point() : x_pos(0), y_pos(0)

Όταν κληθεί η συγκεκριμένη μέθοδος δόμησης θα τεθεί ως αρχική τιμή το 0 και στις δυο μεταβλητές σε αντίθεση με την παρακάτω μέθοδο όπου θα δοθούν οι τιμές x, y

point :: point(**float** x, **float** y) : x_pos(x), y_pos(y)

Λίστες αρχικοποίησης

- Στο πρόγραμμα που ακολουθεί τα σώματα των μεθόδων δόμησης δεν περιλαμβάνουν προτάσεις ανάθεσης τιμών στις μεταβλητές – μέλη και ορίζονται μέσα στην ίδια την κλάση
- Παρατηρούμε ότι ο ορισμός των μεθόδων δόμησης δεν τελειώνει με ερωτηματικό διότι ακολουθεί η κενή σύνθετη πρόταση {} του σώματος της μεθόδου

```
#include <iostream>
using namespace std;
class point
{
public:
        float x_pos;
        float y_pos;
        point():x_pos(0),y_pos(0){}
        point(float x, float y):x_pos(x), y_pos(y){}
        void info() {cout << "Σημείο στο "<<x_pos<<","<<y_pos<<endl;}
};

int main()
{
    point p1,p2(34,45);
    p1.info();
    p2.info();
    return 0;
}
```

```
#include <iostream>
using namespace std;
class point
{
public:
        float x_pos;
        float y_pos;
        point():x_pos(0),y_pos(0){}
        point(float x, float y):x_pos(x), y_pos(y){}
        void info() {cout << "Σημείο στο "<<x_pos<<","<<y_pos<<endl;}
};

int main()
{
    point p1,p2(34,45);
    p1.info();
    p2.info();
    return 0;
}
```

Λίστες αρχικοποίησης

- Οι λίστες αρχικοποίησης είναι ιδιαίτερα χρήσιμες και πολλές φορές απαραίτητες στην περίπτωση της σύνθεσης, όπου μια κλάση έχει ως μέλη αντικείμενα άλλων κλάσεων
- Οι λίστες αρχικοποίησης δίνουν τη δυνατότητα κατά τη δημιουργία ενός αντικειμένου μιας κλάσης Α, η δημιουργία ενός αντικειμένου κλάσης Β να γίνει με ορίσματα, ώστε να κληθεί άλλη μέθοδος δόμησης και όχι η προκαθορισμένη
- Στο πρόγραμμα που ακολουθεί οι μέθοδοι δόμησης και των δυο κλάσεων χρησιμοποιούν λίστες αρχικοποίησης

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

class point
{
		float x_pos;
		float y_pos;
public:
		point():x_pos(0),y_pos(0) {}
		point(float x, float y):x_pos(x),y_pos(y) {}
		void info() {cout << x_pos<<","<<y_pos<<endl;}
};
class circle
{
public:
		float aktina;
		point center;
		string xroma;
		circle():aktina(0),xroma("") {}
		circle(float r):aktina(r),xroma("Ìáýñï") {};
		circle(float r, float x, float y):aktina(r),center(x,y) {};
		void display()
		{
				cout <<"Êýêëïò "<<xroma<<" "<<aktina<<" óôï ";
				center.info();
		}
};
```

```
int main()
{
        circle c1,c2(5),c3(10,30,40);
        c1.display();
        c2.display();
        c3.display();
        return 0;
}
```

Κύκλος 0 στο 0,0
Κύκλος Μαύρο 5 στο 0,0
Κύκλος 10 στο 30,40

Παράδειγμα 1

- Η παρακάτω κλάση box χρησιμοποιείται για τη διαχείριση τρισδιάστατων κουτών με διαστάσεις x, y και z

```
#include <iostream>
using namespace std;
class box
{
public:
        float x;
        float y;
        float z;
        float volume()
        {
           return x*y*z;
   }
        void show();
};
float perimetros(box b)
{
        return 4*(b.x+b.y+b.z);
}
box new_double(box b)
{
        box temp;
        temp.x=2*b.x;
        temp.y=2*b.y;
        temp.z=2*b.z;
        return temp;
}
```

Η μεταβίβαση της παραμέτρου γίνεται κατ αξία. Υπολογίζεται και επιστρέφεται η περίμετρος του κουτιού

Δημιουργείται ένα νέο αντικείμενο – κουτί temp. Το νέο κουτί αποκτά διπλάσιες διαστάσεις από τις διαστάσεις του κουτιού της παραμέτρου και επιστρέφεται ως τιμή

```
void double_it(box &b)
{
		b.x=2*b.x;
		b.y=2*b.y;
		b.z=2*b.z;
}
void box::show()
{
   cout <<"x="<<x<<endl;
   cout <<"y="<<y<<endl;
   cout <<"z="<<z<<endl<<endl;
}

int main()
{
     box b1,b2;
     //ΣΤο αντικείμενο b1 βάζουμε διαστάσεις 3,4 και 5
     b1.x=3;
     b1.y=4;
     b1.z=5;
     b1.show();
     //Στο b2 καταχωρίζεται ενα αντικείμενο διπλάσιο του b1
     b2=new_double(b1);
     b2.show();
     //Διπλασιάζεται το ίδιο το b1
     double_it(b1);
     b1.show();
     return 0;
}
```

Η μεταβίβαση της παραμέτρου γίνεται κατ' αναφορά. Οι αλλαγές στα μέλη της παραμέτρου επηρεάζουν το όρισμα – αντικείμενο με το οποίο καλείται η συνάρτηση

Παράδειγμα 2

- Το παρακάτω πρόγραμμα χρησιμοποιεί μια στατική μεταβλητή – μέλος για να καταχωρίζει το πλήθος των αντικειμένων box

```
#include <iostream>
using namespace std;
class box
{
        float x;
        float y;
        float z;
        static int plithos;
public:
        box();
        ~box();
        void set_xyz(float a, float b, float c);
        void show();
        static void show_plithos(){cout<<"Πλήθος="<<plithos<<endl;}
};
box::box()
{
   plithos++;
}
```

Η στατική μεταβλητή plithos θα χρησιμοποιηθεί για την καταχώρηση του πλήθους των αντικειμένων box. Η μεταβλητή – μέλος θα πρέπει να δηλωθεί και εκτός της κλάσης ως καθολική

Η στατική μέθοδος show_plithos() εμφανίζει την τιμή της στατικής μεταβλητής plithos

```
box::~box()
{
   plithos--;
}
void box::set_xyz(float a, float b, float c)
{
    x=a;
    y=b;
    z=c;
}
void box::show()
{
   cout <<"x="<<x<<endl;
   cout <<"y="<<y<<endl;
   cout <<"z="<<z<<endl<<endl;
}

int box::plithos=0;

int main()
{
     box b1,b2,b3[5];
     //Στο αντικείμενο b1 βάζουμε διαστάσεις 3,4 και 5
     b1.set_xyz(3,4,5);
     box::show_plithos();
     return 0;

}
```

Ορισμός της στατικής μεταβλητής plithos. Μια στατική μεταβλητή – μέλος θα πρέπει να δηλωθεί και εκτός της κλάσης ως καθολική

Δημιουργούνται συνολικά 7 αντικείμενα τύπου box. Στη στατική μέθοδο show_plithos() έχουμε πρόσβαση απευθείας από την κλάση

Πλήθος = 7

Παράδειγμα 3

- Το παρακάτω πρόγραμμα διαχειρίζεται ομαδικές εργασίες φοιτητών. Για τον σκοπό αυτό χρησιμοποιεί δυο κλάσεις: την κλάση **ergasia** και την κλάση **my_date**. Για κάθε αντικείμενο εργασία καταχωρίζονται ο τίτλος, ο βαθμός, το πλήθος φοιτητών που μετέχουν, η ημερομηνία παράδοσης και τα ονόματα μέχρι 5 φοιτητών. Η ημερομηνία παράδοσης είναι ένα αντικείμενο my_date. Η κλάση my_date χρησιμοποιείται για την καταχώριση μιας πλήρους ημερομηνίας καθώς και για την εμφάνιση της. Δεν γίνεται έλεγχος ορθότητας της ημερομηνίας

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

class my_date
{
   int hmera;
   int minas;
   int etos;
public:
   void set_hmer(int h, int m, int e);
   void show();
};
void my_date::set_hmer(int h, int m, int e)
{
   hmera=h;
   minas=m;
   etos=e;
}
void my_date::show()
{
   cout<<hmera<<"/"<<minas<<"/"<<etos<<endl;
}
```

```
class ergasia
{
        float bathmos;
public:
   string titlos;
   int plithos_foititon;
        my_date paradosi;
        string onomata[5];
        ergasia();
        void add_onom(string onom);
        void set_bathmos(float b);
        void display();
};
ergasia::ergasia()
{
   titlos="";
   bathmos=-1;
   plithos_foititon=0;
}
```

```
void ergasia::add_onom(string onom)
{
   if (plithos_foititon<5)
   {
      onomata[plithos_foititon]=onom;
      plithos_foititon++;
      cout << "Το όνομα καταχωρήθηκε! Υπάρχουν ακόμα "<<5-plithos_foititon<<" θέσεις"<<endl;
   }
   else
      cout << "Το όνομα ΔΕΝ καταχωρήθηκε! Όλες οι θέσεις είναι συμπληρωμένες"<<endl;
}
void ergasia::set_bathmos(float b)
{
   bathmos=b;
}
void ergasia::display()
{
   int i;
   cout<<endl<<"Τίτλος:"<<titlos<<endl;
   cout<<"==============================="<<endl;
   cout<<"Ημερομηνία παράδοσης:";
   paradosi.show();
   cout<<"Βαθμός:"<<bathmos<<endl;
   cout <<endl<< "Φοιτητές:"<<plithos_foititon<<endl;
   cout<<"-------------------------------"<<endl;
   for (i=0;i<plithos_foititon;i++)
      cout<<onomata[i]<<endl;
}
```

```
int main()
{
   ergasia e1;
   e1.titlos="Κλάσεις και αντικείμενα";
   e1.add_onom("Αραχτός Νικόλαος");
   e1.add_onom("Ζαμαμφουτίδης Κώστας");
   e1.add_onom("Σπασικλίδης Τηλέμαχος");
   e1.paradosi.set_hmer(5,12,2014);
   e1.set_bathmos(8.5);
   e1.display();
   return 0;
}
```

- Τα αποτελέσματα του προγράμματος:

Το όνομα καταχωρήθηκε! Υπάρχουν ακόμα 4 θέσεις
Το όνομα καταχωρήθηκε! Υπάρχουν ακόμα 3 θέσεις
Το όνομα καταχωρήθηκε! Υπάρχουν ακόμα 2 θέσεις

Τίτλος: Κλάσεις και αντικείμενα
==========
Ημερομηνία παράδοσης : 5/12/2014
Βαθμός : 8.5

Φοιτητές: 3

Αραχτός Νικόλαος
Ζαμαμφουτίδης Κώστας
Σπασικλίδης Τηλέμαχος

Παράδειγμα 4

- Ο παρακάτω κώδικας παρουσιάζει το προηγούμενο πρόγραμμα αλλά με αρχικοποίηση των μεταβλητών μελών στις μεθόδους δόμησης μέσω λιστών αρχικοποίησης

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

class my_date
{
   int hmera;
   int minas;
   int etos;
public:
   my_date():hmera(0),minas(0),etos(0) {}
   my_date(int h, int m, int e):hmera(h),minas(m),etos(e) {}
   void set_hmer(int h, int m, int e);
   void show();
};

void my_date::set_hmer(int h, int m, int e)
{
   hmera=h;
   minas=m;
   etos=e;
}
void my_date::show()
{
   cout<<hmera<<"/"<<minas<<"/"<<etos<<endl;
}
```

```
class ergasia
{
		float bathmos;
public:
   string titlos;
   int plithos_foititon;
		my_date paradosi;
		string onomata[5];
		ergasia():titlos(""),bathmos(-1),plithos_foititon(0) {};
		ergasia(string t,int h, int m, int e):titlos(t),bathmos(-1),plithos_foititon(0),paradosi(h,m,e) {}
		void add_onom(string onom);
		void set_bathmos(float b);
		void display();
};

void ergasia::add_onom(string onom)
{
   if (plithos_foititon<5)
   {
      onomata[plithos_foititon]=onom;
      plithos_foititon++;
      cout << "Το όνομα καταχωρήθηκε! Υπάρχουν ακόμα "<<5-plithos_foititon<<" θέσεις"<<endl;
   }
   else
      cout << "Το όνομα ΔΕΝ καταχωρήθηκε! Όλες οι θέσεις είναι συμπληρωμένες"<<endl;
}
```

```
void ergasia::set_bathmos(float b)
{
   bathmos=b;
}
void ergasia::display()
{
   int i;
   cout<<endl<<"Τίτλος:"<<titlos<<endl;
   cout<<"================================"<<endl;
   cout<<"Ημερομηνία παράδοσης:";
   paradosi.show();
   cout<<"Βαθμός:"<<bathmos<<endl;
   cout <<endl<< "Φοιτητές:"<<plithos_foititon<<endl;
   cout<<"-------------------------------"<<endl;
   for (i=0;i<plithos_foititon;i++)
      cout<<onomata[i]<<endl;
}
int main()
{
   ergasia e1,e2("Λίστες αρχικοποίησης",5,2,2015);
   e1.display();
   e2.add_onom("Πινέζα Μαρία");
   e2.add_onom("Κοντοπίθογλου Αθανάσιος");
   e2.set_bathmos(6);
   e2.display();
   return 0;
}
```

```
Titlos:
================================
Hmerominia paradosis:0/0/0
Bathmos:-1

Foitites:0
--------------------------------
to onoma kataxorithike. Yparxoyn akoma 4 theseis
to onoma kataxorithike. Yparxoyn akoma 3 theseis

Titlos:Listes arxikopoiisis
================================
Hmerominia paradosis:5/2/2015
Bathmos:6

Foitites:2
--------------------------------
Pineza Maria
Kontopithoglou Athanasios

--------------------------------
Process exited after 0.2562 seconds with return value 0
Press any key to continue . . .
```