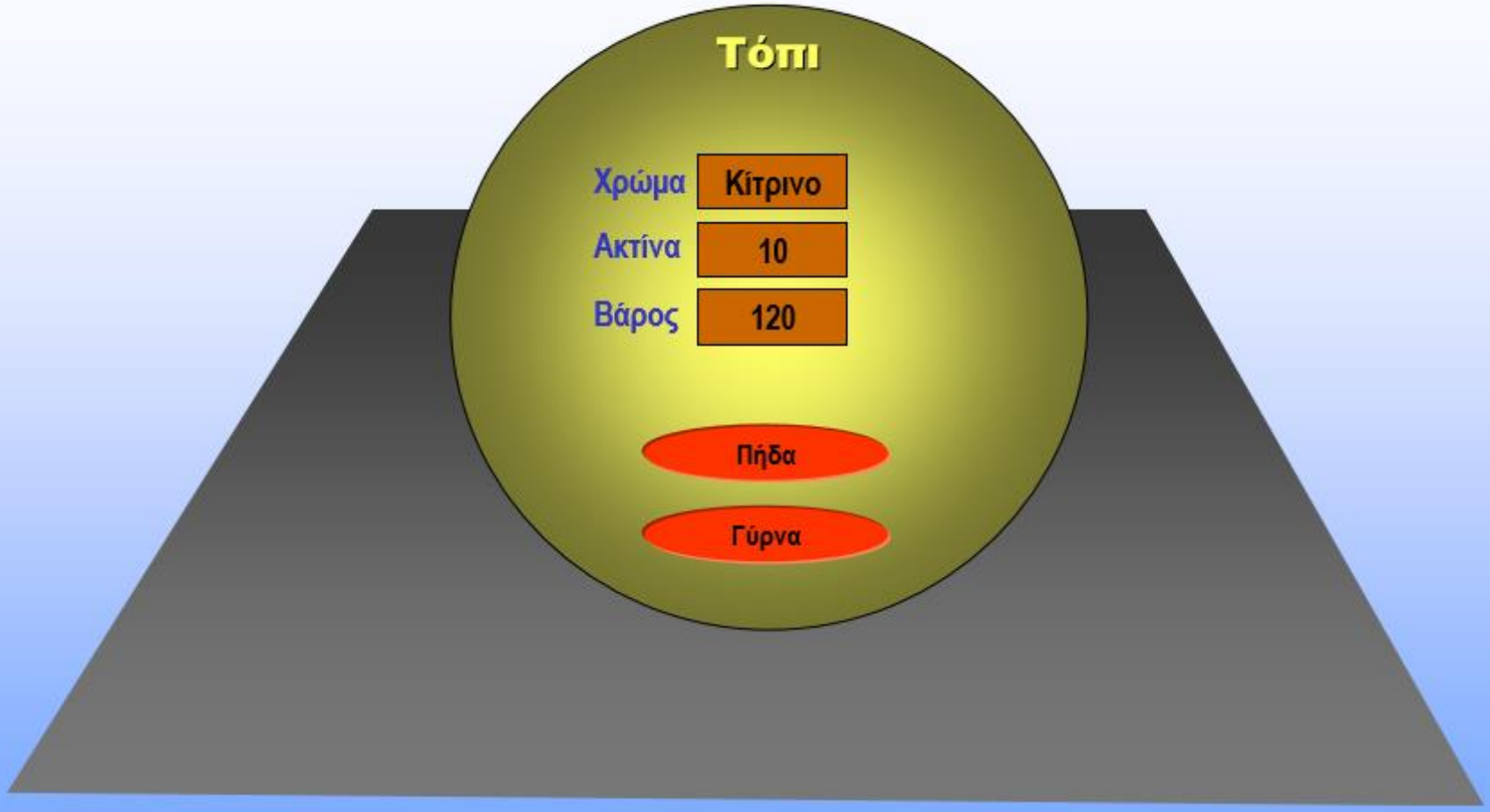
Κλάσεις και αντικείμενα

- Αντικείμενο είναι μια οντότητα η οποία διαθέτει χαρακτηριστικά τα οποία προσδιορίζουν την κατάσταση του και ενέργειες (μεθόδους) οι οποίες καθορίζουν τις διάφορες συμπεριφορές του
- Υποθετικά ένα τόπι είναι ένα αντικείμενο το οποίο έχει τα δικά του χαρακτηριστικά (χρώμα, ακτίνα, βάρος) και πραγματοποιεί ενέργειες όπως περιστροφή και αναπήδηση

Αντικείμενα



Αλλαγή χαρακτηριστικών αντικειμένου – επηρεάζεται η εμφάνιση του

Ενεργοποίηση λειτουργιών αντικειμένου – επηρεάζεται η συμπεριφορά του

Τόπι.Χρώμα=“Ροζ”;

Τόπι.Ακτίνα=15;

Τόπι.Βάρος=120;

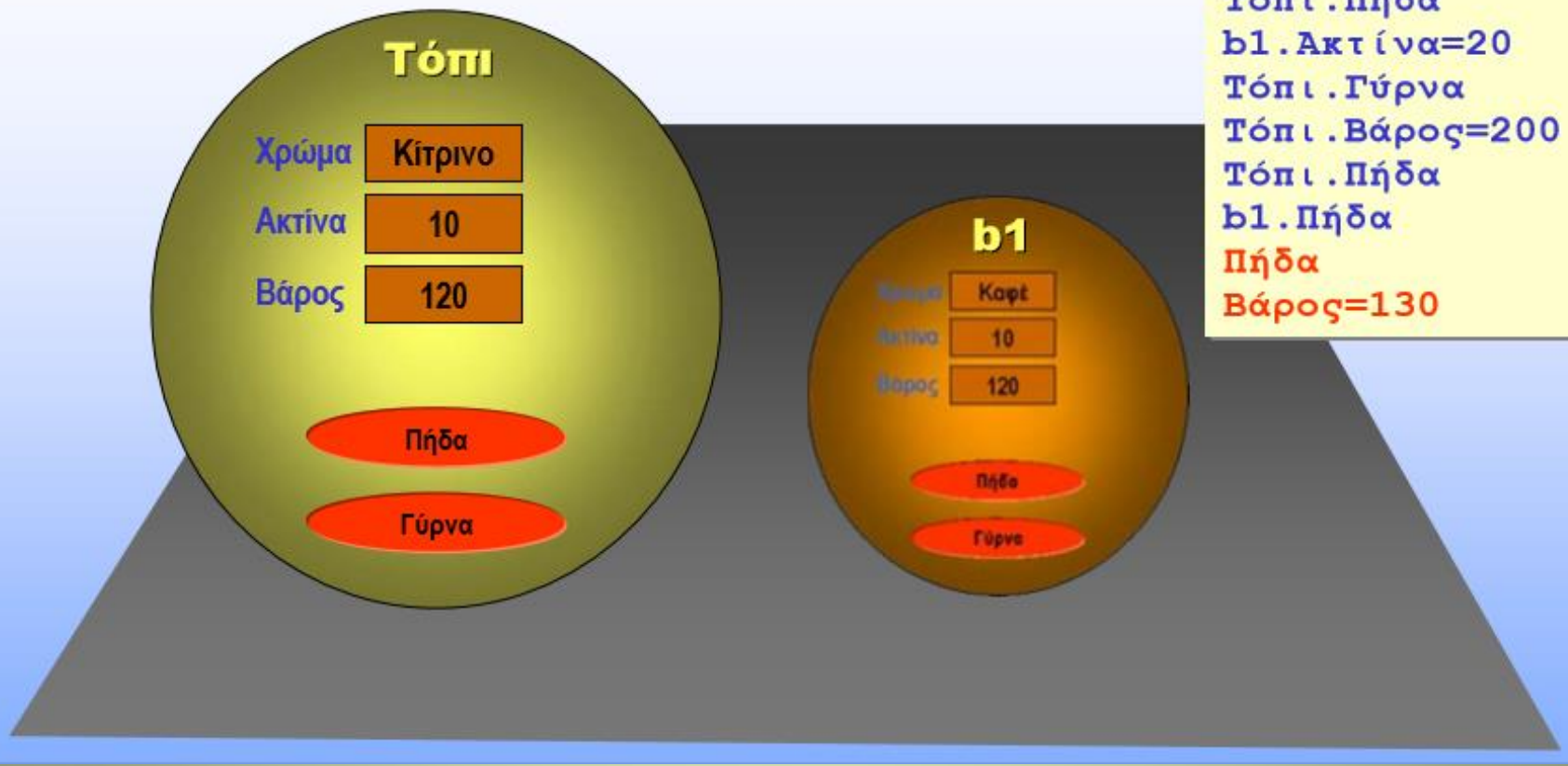
Τόπι.Πήδα;

Τόπι.Πήδα;

Τόπι.Γύρνα;

Η πρόσβαση στα χαρακτηριστικά και στις μεθόδους (ενέργειες) ενός αντικειμένου γίνεται με τον τελεστή της τελείας (όπως στα πεδία μιας δομής)

Αντικείμενα - αλλαγή χαρακτηριστικών/εφαρμογή μεθόδων

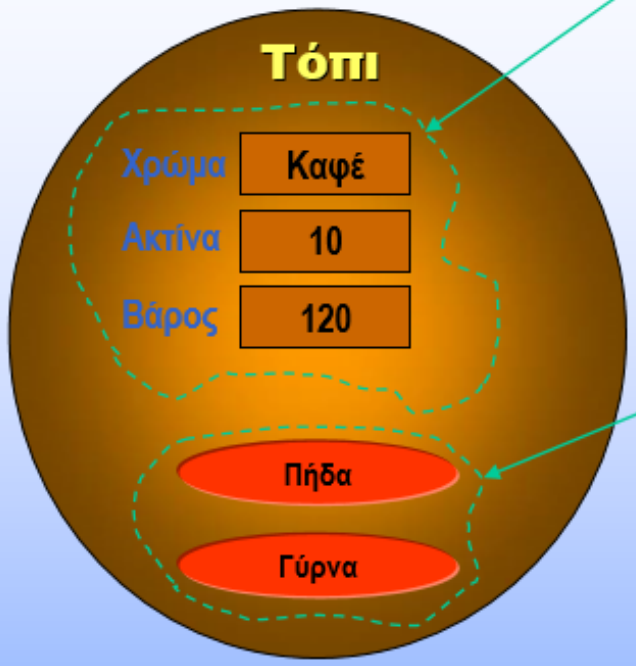



```
Τόπι.Χρώμα="Μπλε"
Τόπι.Ακτίνα=15
Τόπι.Πήδα
b1.Ακτίνα=20
Τόπι.Γύρνα
Τόπι.Βάρος=200
Τόπι.Πήδα
b1.Πήδα
Πήδα
Βάρος=130
```

Τα μέλη ενός αντικειμένου

- Ένα αντικείμενο διαθέτει μέλη στα οποία αποθηκεύονται τα χαρακτηριστικά ή η κατάσταση του. Τα μέλη αυτά στις γλώσσες αντικειμενοστρεφούς προγραμματισμού υλοποιούνται με μεταβλητές. Οι μεταβλητές αυτές καλούνται **μεταβλητές-μέλη** (data members)
- Ένα αντικείμενο διαθέτει μέλη με τα οποία υλοποιούνται διάφορες ενέργειες του. Τα μέλη αυτά στις γλώσσες αντικειμενοστρεφούς προγραμματισμού υλοποιούνται με συναρτήσεις. Οι συναρτήσεις αυτές καλούνται **συναρτήσεις-μέλη ή μέθοδοι** (function members ή methods)

Τα μέλη ενός αντικειμένου



Τιμές που επηρεάζουν τα **χαρακτηριστικά** του αντικειμένου.
Οι τιμές αυτές αποθηκεύονται στις **μεταβλητές-μέλη** (data members).

Ενέργειες που καθορίζουν τη **συμπεριφορά** του αντικειμένου.
Οι ενέργειες αυτές λέγονται **μέθοδοι** και υλοποιούνται με τις **συναρτήσεις-μέλη** (function members ή methods).

- Αν το τόπι ήταν αντικείμενο της C++ ο ακόλουθος κώδικας θα το έκανε να αναπηδήσει 4 φορές, να περιστραφεί κατά 270^{o} και στο τέλος να εμφανίσει στην οθόνη τον όγκο του:

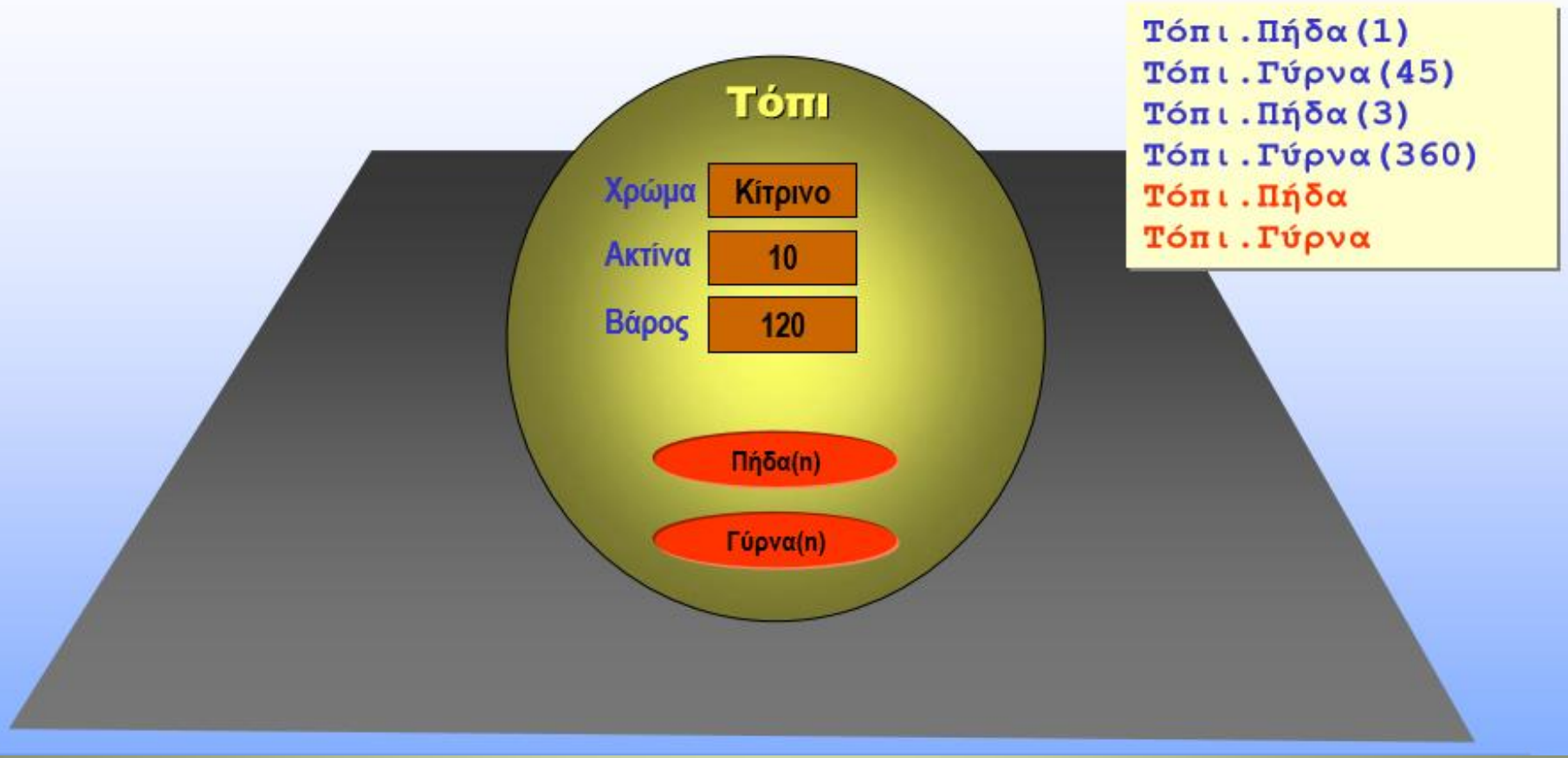
Τόπι.Πήδα(4);

Τόπι.Γύρνα(270);

cout <<Τόπι.Όγκος();

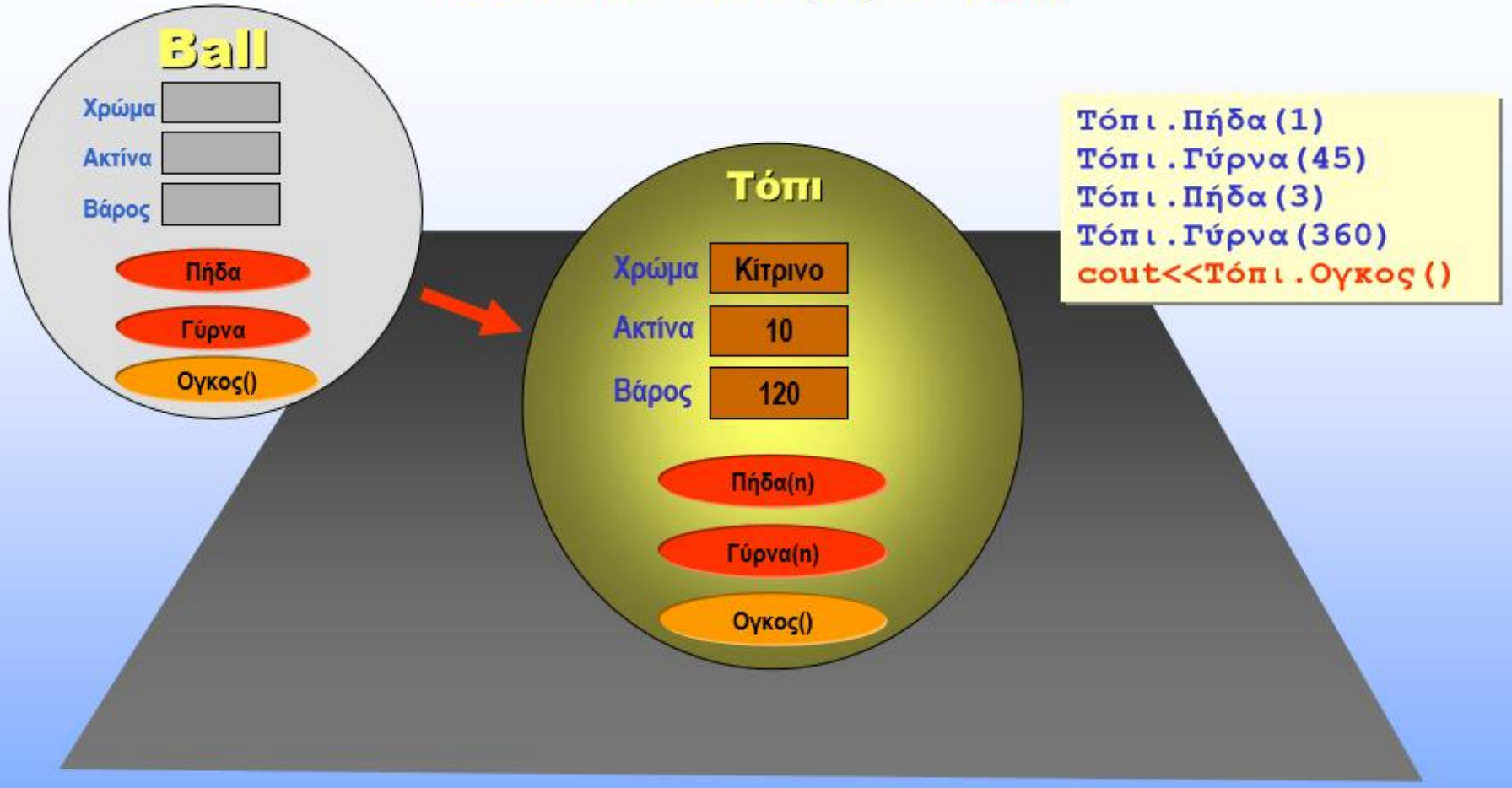
Οι μέθοδοι μπορούν να έχουν παραμέτρους, οι τιμές των οποίων επηρεάζουν την λειτουργία που πρόκειται να εκτελεστεί

Αντικείμενα - μέθοδοι με παραμέτρους



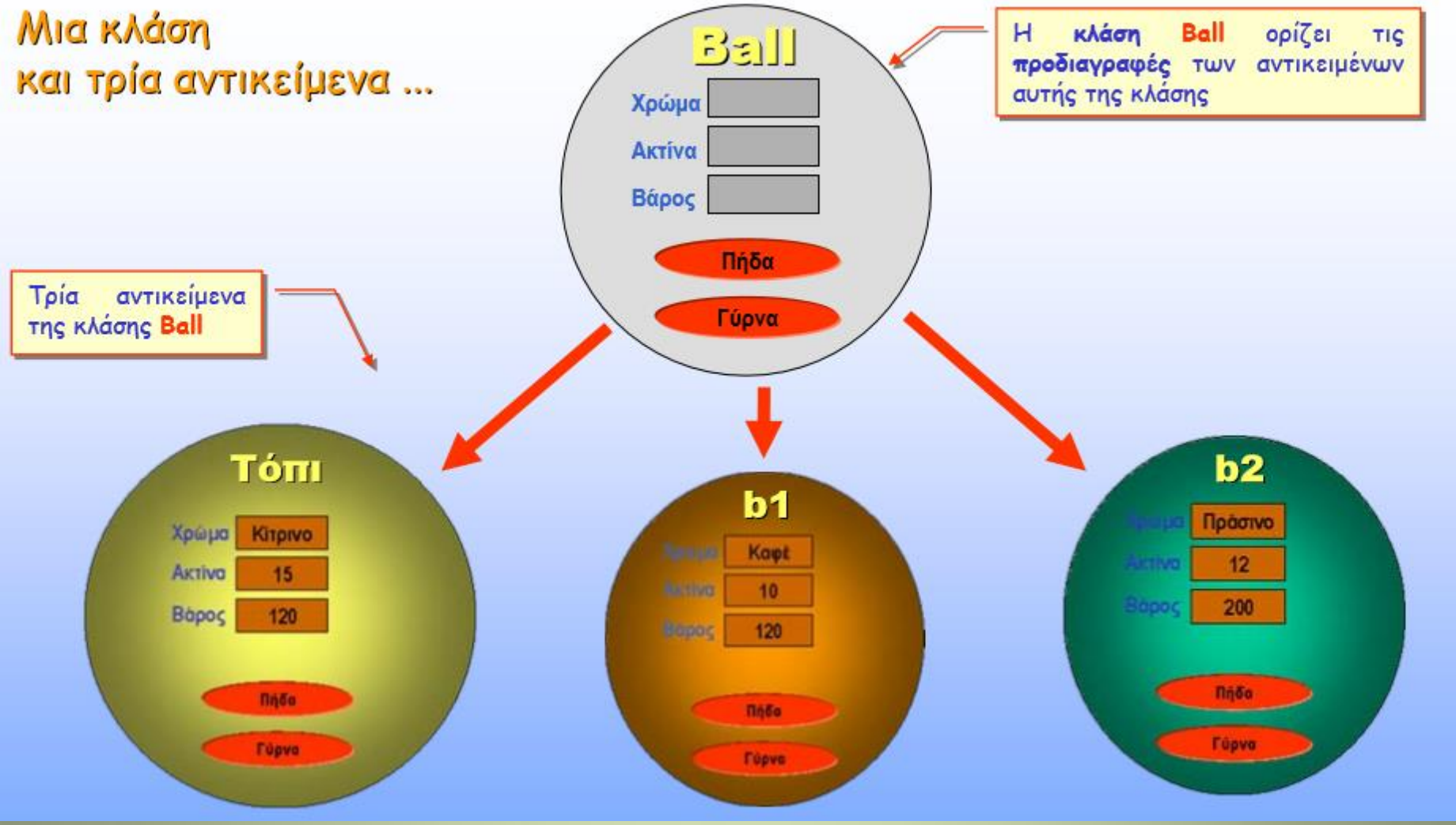
- Οι μέθοδοι μπορούν να επιστρέφουν τιμές. Για παράδειγμα η πρόταση **cout<<Τόπι.όγκος(3)** εμφανίζει στην οθόνη τον όγκο που έχει το τόπι με ακτίνα 3 καθώς ο όγκος μιας σφαίρας καθορίζεται από το μήκος της ακτίνας της $v = \frac{4}{3}\pi R^3$

Μέθοδοι που επιστρέφουν τιμές



- Με το σκεπτικό ότι το τόπι ανήκει σε μια ευρύτερη κατηγορία παιχνιδιών, τις μπάλες, οι οποίες μοιράζονται κοινά χαρακτηριστικά, θα μπορούσαμε να πούμε ότι η Μπάλα αποτελεί μια **κλάση**
- Η **κλάση** (class) είναι μια έννοια η οποία προσδιορίζει μια κατηγορία αντικειμένων και ταυτόχρονα περιγράφει τα κοινά χαρακτηριστικά τους
- Μια κλάση είναι το σύνολο των **προδιαγραφών** για τη κατασκευή αντικειμένων με συγκεκριμένα χαρακτηριστικά και λειτουργίες
- Ένα αντικείμενο είναι μια πραγματική υπόσταση μιας κλάσης, έχει όλα τα χαρακτηριστικά και τις λειτουργίες στην οποία ανήκει και μπορούμε να πούμε πως αποτελεί ένα **στιγμιότυπο** (instance) μιας κλάσης
- Κάθε αντικείμενο έχει **ταυτότητα** (identity) η οποία το διαφοροποιεί από τα υπόλοιπα αντικείμενα, ακόμα και αν τα χαρακτηριστικά τους έχουν ακριβώς τις ίδιες τιμές

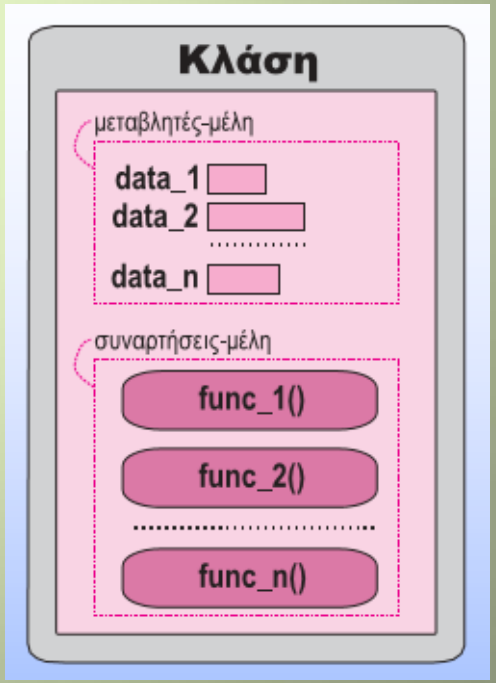
Μια κλάση και τρία αντικείμενα ...



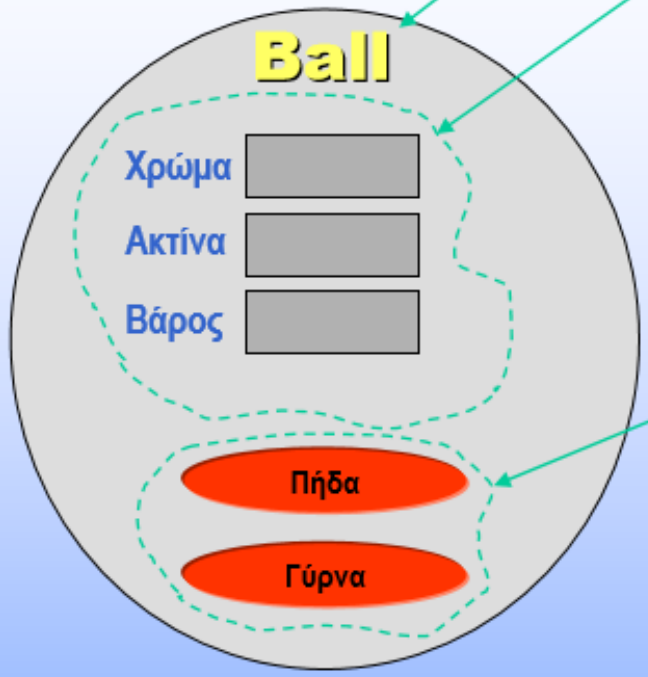
Ορισμός μιας κλάσης

- Μια κλάση αποτελείται από τις **μεταβλητές – μέλη** που προσδιορίζουν τα χαρακτηριστικά και την κατάσταση των αντικειμένων της κλάσης και τις **συναρτήσεις – μέλη** που προσδιορίζουν τις λειτουργίες που μπορούν να επιτελούν

class *όνομα_κλάσης*
{
Δηλώσεις μεταβλητών – μελών
Δηλώσεις συναρτήσεων – μελών
};



Μια κλάση ...

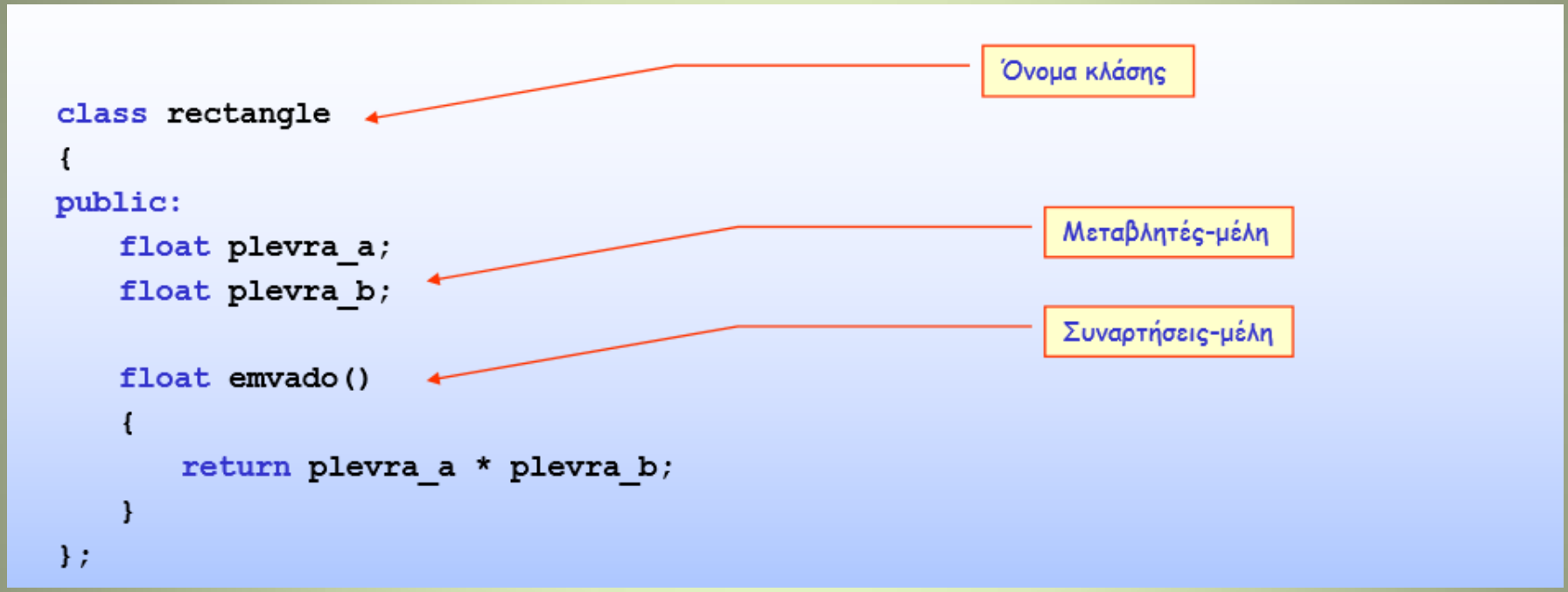


Όνομα κλάσης

Τιμές που επηρεάζουν τα **χαρακτηριστικά** του αντικειμένου.
Οι τιμές αυτές αποθηκεύονται στις **μεταβλητές-μέλη** (data members).

Ενέργειες που καθορίζουν τη **συμπεριφορά** του αντικειμένου.
Οι ενέργειες αυτές υλοποιούνται με τις **συναρτήσεις-μέλη** (function members ή methods).

Παράδειγμα κλάσης




```
class rectangle
{
public:
    float plevra_a;
    float plevra_b;

    float emvado()
    {
        return plevra_a * plevra_b;
    }
};
```

- Όταν μια συνάρτηση – μέλος μιας κλάσης ορίζεται έξω από την κλάση, τότε πριν από το όνομα της πρέπει να αναφέρεται το όνομα της κλάσης στην οποία ανήκει: **όνομα_κλάσης::όνομα_συνάρτησης()**
- ```
  class rectangle
  {
    ……
  };
    float rectangle :: emvado()
     {
        ……
      }
  ```
- Ο τελεστής :: ονομάζεται τελεστής επίλυσης εμβέλειας και χρησιμοποιείται όταν θέλουμε να αναφερθούμε σε μια οντότητα η οποία ορίζεται σε έναν διαφορετικό χώρο εμβέλειας
  ```

Συναρτήσεις – μέλη κλάσης

```
class rectangle
{
public:
    float plevra_a;
    float plevra_b;

    float emvado()
    {
        return plevra_a * plevra_b;
    }
    void display()
    {
        cout << "Orthogonio : " << plevra_a << "x"  << plevra_b << endl;
    }
};
```

Δηλώσεις περισσότερων συναρτήσεων-μελών μέσα στη κλάση

Συναρτήσεις – μέλη κλάσης

```
class rectangle
{
public:
    float plevra_a;
    float plevra_b;
    float emvado();
    void set_ab(float a, float b)
    void display();
};
float rectangle::emvado()
{
    return plevra_a * plevra_b;
}
float rectangle::set_ab(float a, float b)
{
    plevra_a = a;
    plevra_b = b;
}
void rectangle::display()
{
    cout << "Orthogonio : " << plevra_a << "x" << plevra_b << endl;
}
```

Δηλώσεις των συναρτήσεων-μελών μέσα στη κλάση

Ορισμός των συναρτήσεων-μελών έξω από τη κλάση.
Πριν από το όνομα πρέπει να αναφέρεται η κλάση στην οποία ανήκουν!

Ο συναρτήσεις-μέλη μπορεί να έχουν παραμέτρους όπως όλες οι συναρτήσεις!

Συναρτήσεις – μέλη κλάσης

Συναρτήσεις-μέλη κλάσης

```
class rectangle
{
public:
    float plevra_a;
    float plevra_b;

    float emvado();
};

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

Δήλωση της συνάρτησης-μέλος μέσα στη κλάση

Ορισμός της συνάρτησης-μέλος εκτός της κλάσης. Πριν από το όνομα της συνάρτησης προηγείται το όνομα της κλάσης και ο τελεστής επίλυσης εμβέλειας ::

Η συνάρτηση-μέλος έχει πρόσβαση στις μεταβλητές-μέλη της κλάσης

Ορισμός και χρήση αντικειμένων

```
class rectangle
{
   …….
 } rec1, rec2;
```

Ή

```
rectangle rec1, rec2;
```

Τα αντικείμενα rec1, rec2 της κλάσης rectangle μπορούν να δημιουργηθούν με τον ορισμό της κλάσης ή με ξεχωριστή πρόταση. Αν δηλωθούν πριν την main() έχουν καθολική εμβέλεια (1ος τρόπος), ενώ αν δηλωθούν μέσα στη main() έχουν τοπική εμβέλεια

Πρόσβαση στα μέλη ενός αντικειμένου

rec1.plevra_a = 10;

rec2.plevra_b =20;

Η πρόσβαση στα μέλη ενός αντικειμένου μιας κλάσης γίνεται μέσω του τελεστή τελεία (.)

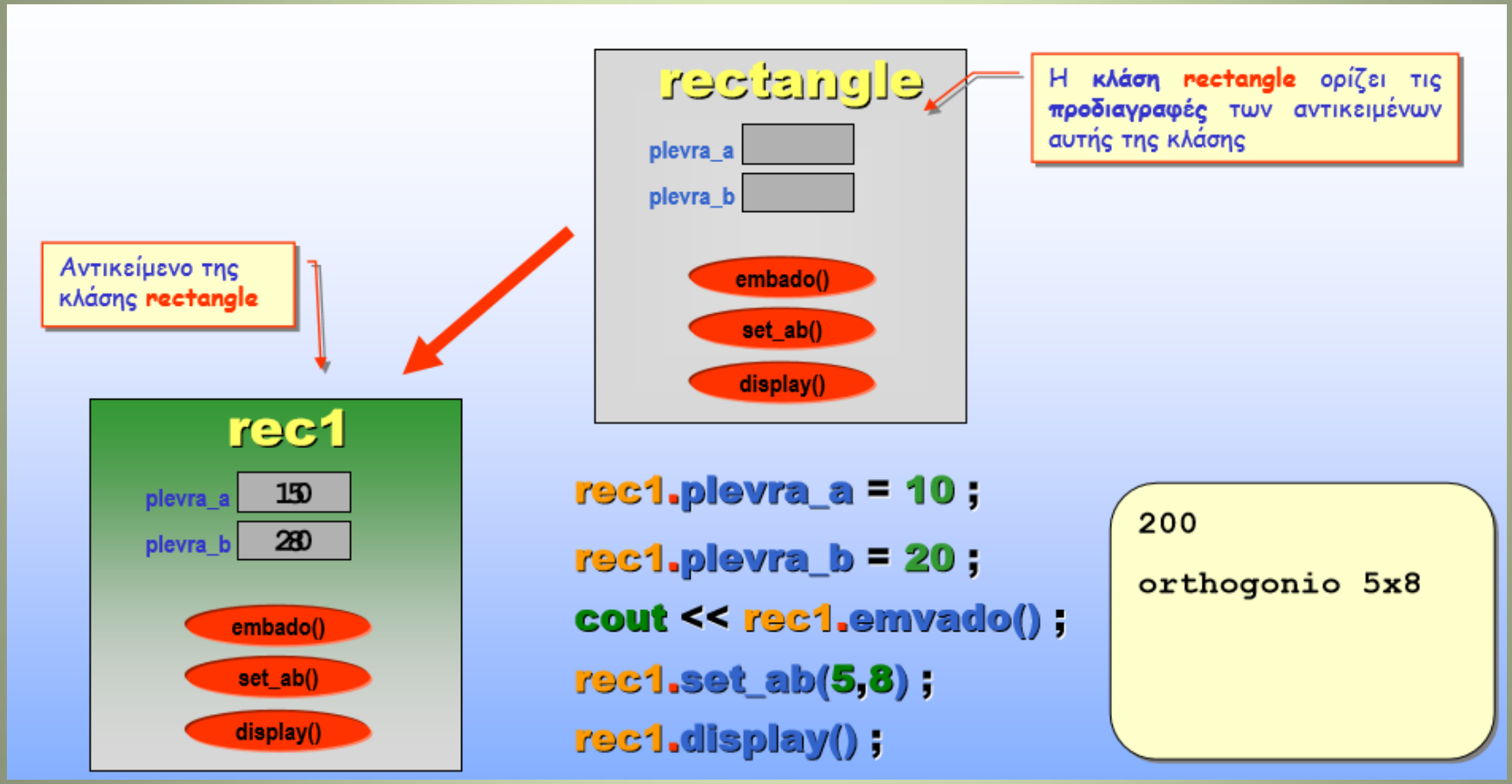
Μια συνάρτηση – μέλος λέγεται επίσης μέθοδος (method) της κλάσης και καλείται πάντα

Πρόσβαση στα μέλη ενός αντικειμένου

```
cout << rec1.emvado();
```

Στη παραπάνω πρόταση εφαρμόζουμε τη **μέθοδο** emvado() στο αντικείμενο rec1

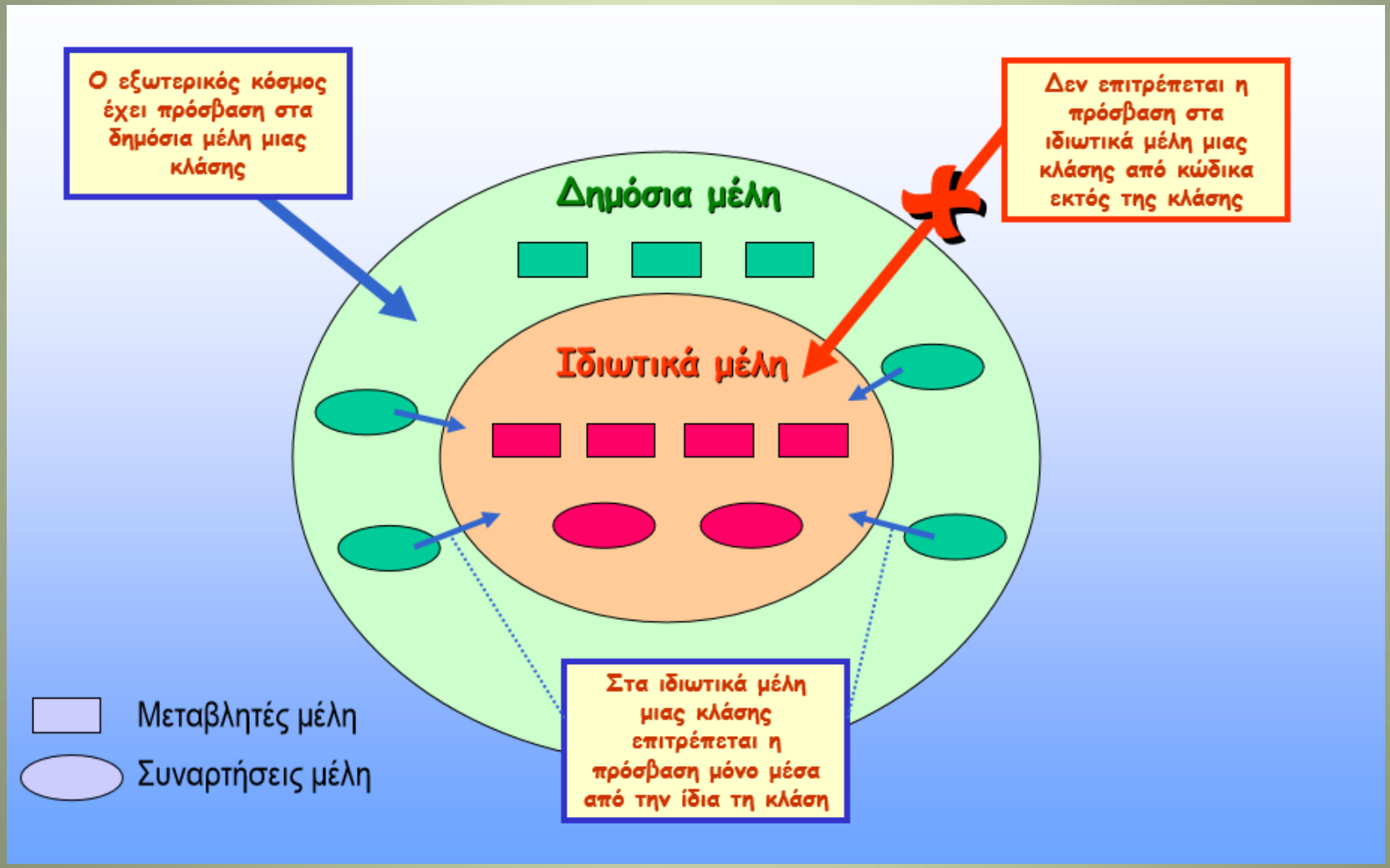
Πρόσβαση στα μέλη ενός αντικειμένου



Δημόσια και ιδιωτικά μέλη κλάσης

- Τα μέλη μιας κλάσης μπορεί να είναι ιδιωτικά (**private:**) ή δημόσια (**public:**)
- Τα **ιδιωτικά** μέλη είναι προσπελάσιμα μόνο από μεθόδους της κλάσης
- Στα **δημόσια** έχουν πρόσβαση ακόμα και συναρτήσεις οι οποίες δεν ανήκουν στην κλάση
- Εξορισμού τα μέλη μιας κλάσης είναι ιδιωτικά συνεπώς το προσδιοριστικό **private**: μπορεί να παραλειφθεί

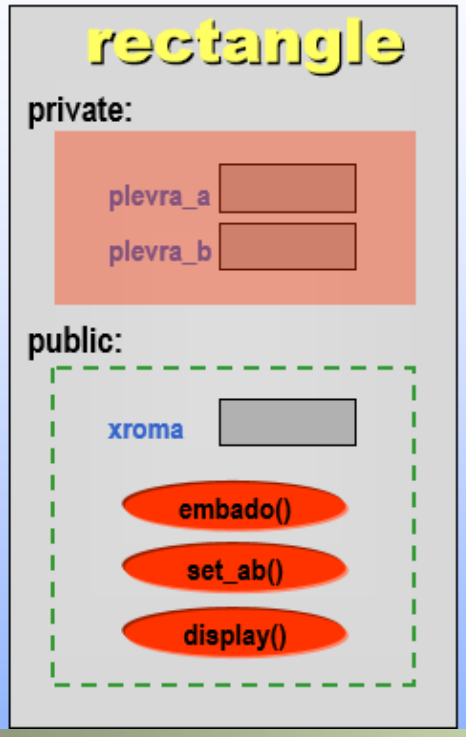
Δημόσια και ιδιωτικά μέλη κλάσης



Δημόσια και ιδιωτικά μέλη κλάσης

- Η συσκευασία των δεδομένων και των μεθόδων σε μια ενιαία οντότητα, δηλαδή σε ένα αντικείμενο, λέγεται **ενθυλάκωση**
- Η απόκρυψη των δεδομένων και των μεθόδων ενός αντικειμένου μέσω της ιδιωτικοποίησης τους οδηγεί στην **αφαιρετικότητα**
- Ένα αντικείμενο επικοινωνεί με το προγραμματιστικό του περιβάλλον μέσω ενός συστήματος **διεπαφής** (interface) αποκρύπτοντας τα υπόλοιπα χαρακτηριστικά του
- Η **διεπαφή** ενός αντικειμένου με το περιβάλλον του γίνεται μόνο μέσω των δημόσιων μελών του. Το σύστημα διεπαφής μπορεί να περιλαμβάνει τόσο μεταβλητές όσο και συναρτήσεις - μέλη

Ιδιωτικά μέλη μιας κλάσης




```
.....
rectangle rec1;
.....
rec1.plevra_a=10;
rec1.plevra_b=5;
rec1.set_ab(10,5);
rec1.display();
cout << rec1.embado();
.....
```

Δεν επιτρέπεται η πρόσβαση στα ιδιωτικά μέλη της κλάσης.

```
orthogonio 10x5
50
```

Παραδείγματα κλάσεων

```
class person
{
    string onoma;
    int ilikia;
    float varos;
    float ypsos;
};
```

Η κλάση αυτή δεν έχει δημόσια μέλη. Τα αντικείμενα της κλάσης δεν θα έχουν **καμία διεπαφή** με τον έξω κόσμο και θα είναι πρακτικά άχρηστα

```
class person
{
    float varos;
    float ypsos;
 public:
    int ilikia;
    string onoma;
};
```

Δεν υπάρχει τρόπος πρόσβασης στα ιδιωτικά μέλη της κλάσης. Θα έπρεπε να υπήρχαν συναρτήσεις-μέλη που να διαχειρίζονται τα μέλη αυτά.

```
class person
{
 public:
    string onoma;
    int ilikia;
    float varos;
    float ypsos;
}filos,pelatis;
```

Όλα τα μέλη της κλάσης είναι δημόσια και προσβάσιμα από κώδικα εκτός της κλάσης. Ταυτόχρονα με τον ορισμό της κλάσης δημιουργούνται και δύο αντικείμενα το filos και το pelatis.

Μέθοδοι δόμησης και αποδόμησης

- Όταν δημιουργούμε ένα νέο αντικείμενο πρέπει να δώσουμε τιμές στις μεταβλητές – μέλη του για να αποκτήσει μια συγκεκριμένη υπόσταση. Μέχρι τότε οι μεταβλητές – μέλη είναι κενές και περιέχουν **απροσδιόριστες** τιμές
- Στο παρακάτω τμήμα κώδικα δημιουργείται ένα αντικείμενο rec3 της κλάσης rectangle. Αν κληθεί η μέθοδος emvado() για αυτό το αντικείμενο το πιθανότερο θα είναι να επιστρέψει μια απροσδιόριστη τιμή:

Μέθοδοι δόμησης και αποδόμησης

```
 int main()
{
  rectangle rec3;
  cout << rec3.emvado();
  return 0;
 }
```

Η **μέθοδος δόμησης** (constructor method) είναι μια ειδική μέθοδος κλάσης η οποία καλείται αυτόματα με τη δημιουργία ενός αντικειμένου αυτής της κλάσης. Η μέθοδος δόμησης έχει το ίδιο όνομα με τη κλάση και δεν ανήκει σε κανένα τύπο

Μέθοδοι δόμησης και αποδόμησης

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

Η παραπάνω μέθοδος δόμησης δίνει αρχικές τιμές στις διαστάσεις κάθε νέου αντικειμένου της κλάσης rectangle

Μέθοδοι δόμησης και αποδόμησης

- Αντίστοιχα η μέθοδος αποδόμησης (ή συνάρτηση καταστροφής – destructor) εφαρμόζεται αυτόματα όταν καταστρέφεται ένα αντικείμενο
- Η μέθοδος αποδόμησης είναι μια ειδική μέθοδος κλάσης η οποία καλείται αυτόματα μόλις καταστρέφεται ένα αντικείμενο αυτής της κλάσης. Η μέθοδος αποδόμησης έχει το ίδιο όνομα με τη κλάση αλλά με το πρόθεμα ~ πριν από το όνομα της και επίσης δεν ανήκει σε κανένα τύπο

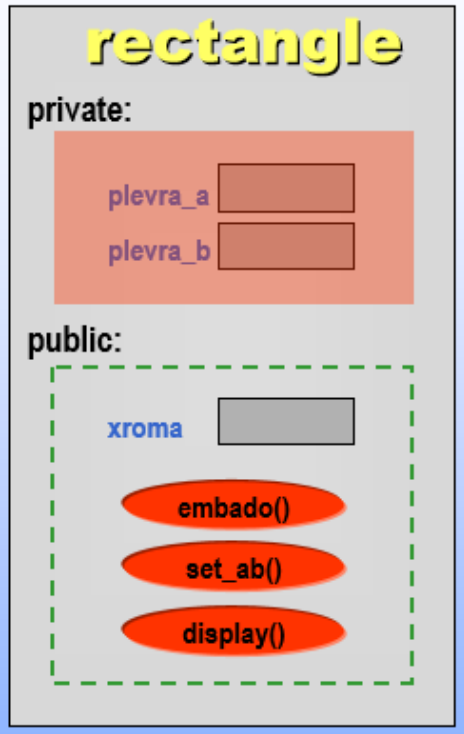
Μέθοδοι δόμησης και αποδόμησης

```
rectangle :: ~ rectangle()
{
  cout << “καταστράφηκε” << endl;
}
```

- Ένα αντικείμενο καταστρέφεται μόλις λήξει η εμβέλεια του

```
 int main()
{
   rectangle rec1; //δημιουργία αντικειμένου rec1
   {
      rectangle rec3;//δημιουργία αντικειμένου rec3
   }//καταστροφή αντικειμένου rec3
   rectangle rec2; //δημιουργία αντικειμένου rec2
   return 0;//καταστροφή αντικειμένων rec1 και rec2
}
```

Συναρτήσεις δόμησης και αποδόμησης




```
.....
rectangle rec1;
.....
rec1.display();
.....
```

Ερώτηση: Τι περιέχουν οι μεταβλητές-μέλη μετά από την δημιουργία ενός αντικειμένου?

Απάντηση: Έχουν απροσδιόριστες τυχαίες τιμές!

Το πρόβλημα αυτό λύνεται με χρήση συναρτήσεων δόμησης

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

Μια συνάρτηση **δόμησης** (constructor) έχει ίδιο όνομα με τη κλάση και εκτελείται αυτόματα με τη δημιουργία ενός αντικειμένου. Χρησιμοποιείται συνήθως για την απόδοση αρχικών τιμών στις μεταβλητές-μέλη ενός αντικειμένου.

```
rectangle::~rectangle()
```

Μια συνάρτηση **αποδόμησης** (destructor) έχει ίδιο όνομα με τη κλάση αλλά με πρόθεμα τον χαρακτήρα ~, και εκτελείται αυτόματα με την καταστροφή ενός αντικειμένου.

Προκαθορισμένες μέθοδοι δόμησης και αποδόμησης

- Για μια κλάση δεν είναι υποχρεωτικό να οριστούν μέθοδοι δόμησης και αποδόμησης
- Κάθε κλάση διαθέτει προκαθορισμένες μεθόδους δόμησης και αποδόμησης οι οποίες χρησιμοποιούνται σε περίπτωση όπου δεν έχουν οριστεί άλλες από τον προγραμματιστή
- Η προκαθορισμένη μέθοδος δόμησης δεν έχει παραμέτρους και απλώς δημιουργεί το αντικείμενο χωρίς άλλη ενέργεια και χωρίς να βάζει κάποιες τιμές στις μεταβλητές – μέλη του
- Η προκαθορισμένη μέθοδος αποδόμησης καταστρέφει το αντικείμενο και απελευθερώνει τη μνήμη που καταλάμβανε

Μέθοδοι δόμησης με παραμέτρους

```
rectangle rec (10,2);

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

Στη περίπτωση που η μέθοδος δόμησης έχει παραμέτρους τότε στη δήλωση των αντικειμένων θα πρέπει να χρησιμοποιούνται ορίσματα τα οποία θα μεταβιβάζονται στις αντίστοιχες παραμέτρους της μεθόδου δόμησης.
Εναλλακτικά: **rectangle** rec1 = rectangle (10,2);

Μέθοδοι δόμησης με παραμέτρους

- Ένα ολοκληρωμένο πρόγραμμα με χρήση συναρτήσεων δόμησης και αποδόμησης είναι το παρακάτω.

```cpp
#include <iostream>
#include <string>
using namespace std;
class rectangle
{
    float plevra_a;
    float plevra_b;
public:
    string xroma;
    rectangle(float a, float b);
    ~rectangle();
    float emvado();
    void set_ab(float a, float b);
};
float rectangle::emvado()
{
    return plevra_a * plevra_b;
}
void rectangle::set_ab(float a, float b)
{
    plevra_a = a;
    plevra_b = b;
}
rectangle::rectangle(float a, float b)
{
    plevra_a = a;
    plevra_b = b;
    cout    << "Δημιουργήθηκε ένα παραλληλόγραμμο " << a << "x" << b << endl;
}

rectangle::~rectangle()
{
    cout    << "Ένα παραλληλόγραμμο "
            << plevra_a << "x" << plevra_b << " καταστράφηκε" << endl;
}
int main()
{
    rectangle rec1(10,2),rec2(4,8);
    cout    << "Εμβαδό rec1=" << rec1.emvado() << endl
            << "Εμβαδό rec2=" << rec2.emvado() << endl;
    {
        rectangle rec3(3,7);
        cout    <<"Εμβαδό rec3="<<rec3.emvado()<<endl;
        rec3.set_ab(8,10);
        cout    <<"Εμβαδό rec3="<<rec3.emvado()<<endl;
    }
    return 0;
}
```

Αποτελέσματα προγράμματος

- Δημιουργήθηκε ένα παραλληλόγραμμο 10x2
- Δημιουργήθηκε ένα παραλληλόγραμμο 4x8
- Εμβαδό rec1= 20
- Εμβαδό rec2= 32
- Δημιουργήθηκε ένα παραλληλόγραμμο 3x7
- Εμβαδό rec3= 21
- Εμβαδό rec3= 80
- Ένα παραλληλόγραμμο 8x10 καταστράφηκε
- Ένα παραλληλόγραμμο 4x8 καταστράφηκε
- Ένα παραλληλόγραμμο 10x2 καταστράφηκε

Μέθοδοι δόμησης και αποδόμησης

- Δεν είναι υποχρεωτικό μια κλάση να έχει μεθόδους δόμησης και αποδόμησης
- Στην περίπτωση που δεν έχουν οριστεί μέθοδοι δόμησης και αποδόμησης χρησιμοποιούνται οι προκαθορισμένες μέθοδοι που δημιουργεί ο μεταγλωττιστής
- Μια κλάση μπορεί να έχει πολλές μεθόδους δόμησης αλλά μόνο μια αποδόμησης
- Οι μέθοδοι δόμησης και αποδόμησης δεν έχουν τύπο
- Οι μέθοδοι δόμησης και αποδόμησης πρέπει να είναι δημόσιες
- Οι μέθοδοι δόμησης μπορούν να έχουν παραμέτρους
- Οι μέθοδοι αποδόμησης δεν επιτρέπεται να έχουν παραμέτρους

Υπερφόρτωση μεθόδων

- Οι μέθοδοι μιας κλάσης μπορούν να **υπερφορτωθούν**
- Σε κάθε κλήση μιας υπερφορτωμένης μεθόδου, ο **μεταγλωττιστής** της C++ επιλέγει την κατάλληλη έκδοση της ανάλογα με τον τύπο και τον αριθμό των ορισμάτων της κλήσης

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

Όταν καλείται η υπερφορτωμένη μέθοδος set_ab() με δυο ορίσματα, τότε εκτελείται η παρούσα έκδοση και καταχωρίζει τις τιμές των ορισμάτων στις μεταβλητές – μέλη του αντικειμένου για το οποίο κλήθηκε

Όταν καλείται η υπερφορτωμένη μέθοδος με ένα όρισμα τότε εκτελείται η δεύτερη έκδοση της set_ab() η οποία διπλασιάζει τις διαστάσεις ενός αντικειμένου της κλάσης rectangle

Υπερφόρτωση μεθόδων δόμησης

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

Αυτή η υπερφορτωμένη μέθοδος δόμησης καλείται όταν δημιουργούμε ένα αντικείμενο τύπου rectangle χωρίς ορίσματα (δίνει μηδενικές διαστάσεις)

Οι παραπάνω υπερφορτωμένες μέθοδοι δόμησης δίνουν τη δυνατότητα να δηλώσουμε αντικείμενα με ορίσματα τις αρχικές τους διαστάσεις ή χωρίς καθόλου ορίσματα

Υπερφόρτωση συναρτήσεων δόμησης

```
class rectangle
{
public:
    float plevra_a;
    float plevra_b;
    rectangle();
    rectangle(float x);
    rectangle(float x, float y);
    float emvado();
    void display();
};
rectangle::rectangle()
{
    plevra_a=0;
    plevra_b=0;
}
rectangle::rectangle(float x)
{
    plevra_a=plevra_b=x;
}
rectangle::rectangle(float x, float y)
{
    plevra_a=x;
    plevra_b=y;
}
```

Δηλώσεις των υπερφορτωμένων μεθόδων δόμησης μέσα στη κλάση. Προσέχουμε ότι οι συναρτήσεις δόμησης δεν έχουν τύπο.

Ορισμός των μεθόδων δόμησης έξω από τη κλάση.

Η προκαθορισμένη μέθοδος δόμησης (χωρίς παράμετρο). Καταχωρίζει μηδενικές τιμές στις πλευρές του ορθογωνίου.

Μέθοδος δόμησης με μία παράμετρο. Καταχωρίζει την ίδια τιμή στις πλευρές του ορθογωνίου.

Μέθοδος δόμησης με δύο παραμέτρους. Καταχωρίζει τις τιμές των παραμέτρων στις πλευρές του ορθογωνίου.

Υπερφόρτωση μεθόδων δόμησης

- **ΠΡΟΣΟΧΗ**: Και οι δυο εκδόσεις της υπερφορτωμένης μεθόδου δόμησης πρέπει να δηλώνονται μέσα στο σώμα της κλάσης
- Ο μεταγλωττιστής επιλέγει την κατάλληλη έκδοση της μεθόδου δόμησης ανάλογα με τον αριθμό και τον τύπο των ορισμάτων που χρησιμοποιούνται κατά τη δήλωση του αντικειμένου
- Δεν επιτρέπεται η υπερφόρτωση των μεθόδων αποδόμησης

Πως καλούνται οι συναρτήσεις δόμησης

Οι συναρτήσεις δόμησης καλούνται **αυτόματα** κατά τη πρόταση δήλωσης ενός αντικειμένου. Στη περίπτωση υπερφόρτωσης της συνάρτησης δόμησης, ανάλογα με τον **τρόπο δήλωσης** του αντικειμένου καλείται και η κατάλληλη έκδοση της συνάρτησης δόμησης. **ΔΕΝ** υπάρχει άλλος τρόπος κλήσης μιας συνάρτησης δόμησης!

```
rectangle rec1;

rectangle rec2(10);

rectangle rec3(5,20);
```

Καλείται η **προκαθορισμένη** συνάρτηση δόμησης (χωρίς παράμετρο). Αν δεν υπάρχει δεν καλείται καμία συνάρτηση και δεν αποδίδονται αρχικές τιμές στις μεταβλητές μέλη του rec1.

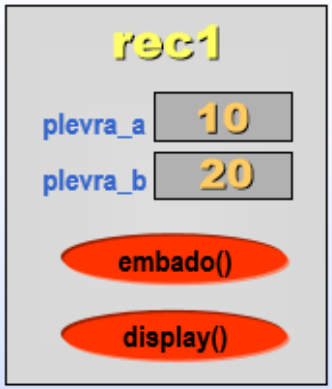
Καλείται η η **υπερφορτωμένη** έκδοση της συνάρτησης δόμησης (με μία παράμετρο) και δημιουργεί ένα τετράγωνο 10x10. Αν δεν υπάρχει, η μεταγλώττιση σταματάει με σχετικό μήνυμα λάθους.

Καλείται η η **υπερφορτωμένη** έκδοση της συνάρτησης δόμησης (με δύο παραμέτρους) και δημιουργεί ένα ορθογώνιο 5x20. Αν δεν υπάρχει, η μεταγλώττιση σταματάει με σχετικό μήνυμα λάθους.

Ανάθεση τιμών σε αντικείμενα

- Με καταχώριση τιμών στις μεταβλητές – μέλη του αντικειμένου με απευθείας πρόσβαση σε αυτές (αν είναι public)
- Μέσω μιας δημόσιας μεθόδου της κλάσης η οποία με τη σειρά της καταχωρίζει τιμές στις μεταβλητές – μέλη του αντικειμένου για το οποίο καλείται
- Μέσω μιας μεθόδου δόμησης η οποία καταχωρίζει αρχικές τιμές στις μεταβλητές μέλη του αντικειμένου αυτόματα με τη δημιουργία τους
- Μπορούμε επίσης να καταχωρίσουμε σε ένα αντικείμενο την τιμή ενός άλλου αντικειμένου, χρησιμοποιώντας τον τελεστή τιμής =. Σε αυτή την περίπτωση καταχωρίζονται οι τιμές των μεταβλητών – μελών του ενός αντικειμένου στο άλλο. Τα αντικείμενα πρέπει να ανήκουν στην ίδια κλάση ***αντικείμενο1 = αντικείμενο2***

Χρήση του τελεστή ανάθεσης σε αντικείμενα κλάσεων




```
αντικείμενο1 = αντικείμενο2 ;
```

Ο τελεστής ανάθεσης ίσον (=) χρησιμοποιείται για την καταχώριση των στοιχείων του αντικειμένου2 στο αντικείμενο1. Τα αντικείμενα πρέπει να είναι της ίδιας κλάσης!

```
rectangle rec1,rec2 ;

rec1.plevra_a = 10 ;

rec1.plevra_b = 20 ;

rec2 = rec1 ;

rec2.display() ;
```

```
orthogonio 10x20
```

- Ένα πεδίο μιας κλάσης μπορεί να είναι ένας πίνακας όπως φαίνεται στον παρακάτω κώδικα

```
class student
{
private:
    float bathmoi[5];
public:
    string onoma;
    string kodikos;
    int etos;
};

void student::get()
{
    int i;
    for (i=0;i<5;i++) cin>>bathmoi[i];
}

float student::mesos()
{
    int i;
    float ath=0;
    for (i=0;i<5;i++) ath=ath+bathmoi[i];
    return ath/5;
}
```

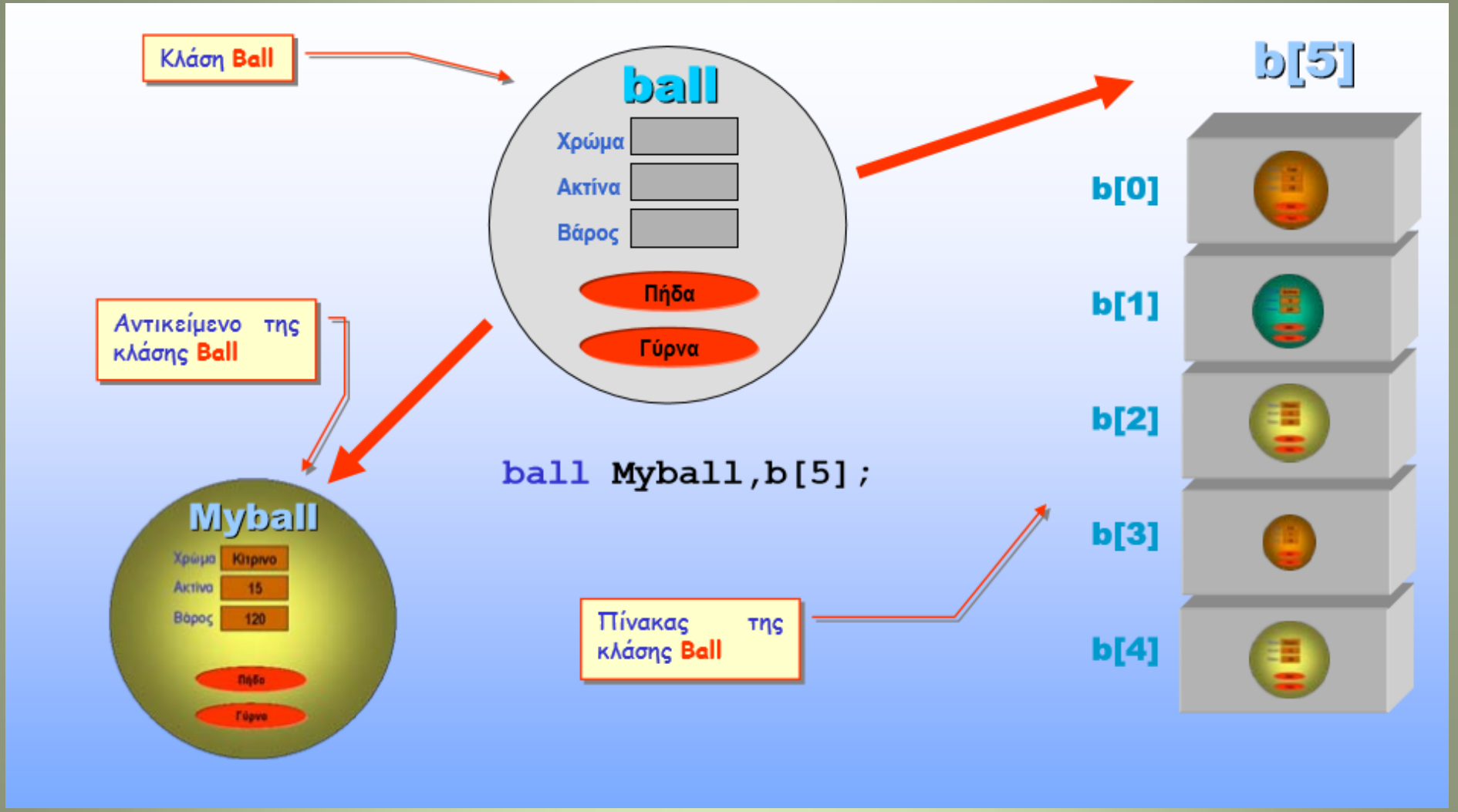
Συναρτήσεις-μέλη της κλάσης

```
#include <iostream>
using namespace std;
.....
int main()
{
    student s1;
    s1.onoma="Takis";
    s1.kodikos="ct81000";
    s1.etos=2012;
    s1.get();
    cout<<s1.mesos();
}
```

Πίνακες από αντικείμενα

- Παρόμοιοι με τους πίνακες από δομές
- Κάθε θέση μνήμης του πίνακα περιέχει ένα αντικείμενο με τα δεδομένα καταχωρισμένα στις μεταβλητές – μέλη του

```
rectangle rec[5];
rec[0].set_ab(10,2);
rec[1].set_ab(5,3);
……
rec[4].set_ab(7,9);
```


Κλάση Ball
ball
Χρώμα
Ακτίνα
Βάρος
Πήδα
Γύρνα
b[5]
b[0]
b[1]
b[2]
b[3]
b[4]
Αντικείμενο της κλάσης Ball
ball Myball,b[5];
Myball
Χρώμα
Κίτρινο
Ακτίνα
15
Βάρος
120
Πήδα
Γύρνα
Πίνακας της κλάσης Ball

Δημιουργία πίνακα της κλάσης rectangle

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

class rectangle
{
public:
    float plevra_a;
    float plevra_b;
    float emvado();
    void display();
};
.....
int main()
{
    int i;
    rectangle r[5];
    for (i=0;i<5;i++)
    {
        r[i].plevra_a=rand()%100;
        r[i].plevra_b=rand()%100;
    }
    for (i=0;i<5;i++)
        r[i].display();
    for (i=0;i<5;i++)
        cout << r[i].emvado()<<endl;
    return 0;
}
```

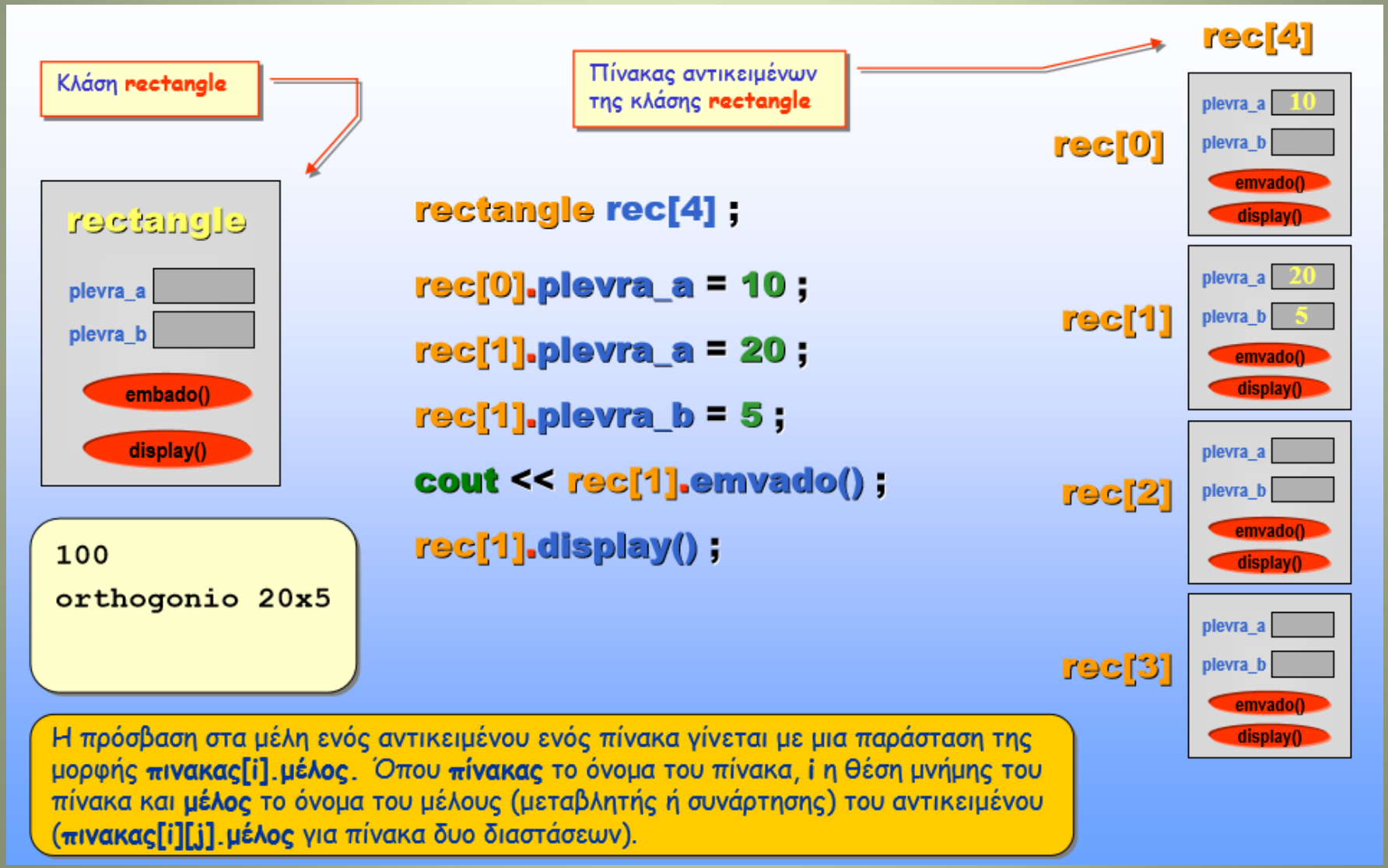
Δημιουργεί έναν πίνακα κλάσης **rectangle** με 5 αντικείμενα.

Καταχωρεί σε όλα τα αντικείμενα του πίνακα τυχαίες διαστάσεις.

Εμφανίζει τις διαστάσεις των αντικειμένων του πίνακα.

Εμφανίζει το εμβαδόν όλων των αντικειμένων του πίνακα.

Πρόσβαση στα μέλη ενός αντικειμένου σε πίνακα



Πίνακες από αντικείμενα

- Κάθε φορά που δημιουργείται ένας πίνακας αντικειμένων η μέθοδος δόμησης της κλάσης καλείται τόσες φορές όσες είναι οι θέσεις του πίνακα
- Κατά την καταστροφή του πίνακα η μέθοδος αποδόμησης της κλάσης καλείται τόσες φορές όσες είναι οι θέσεις του πίνακα
- Ένας πίνακας αντικειμένων μπορεί να μεταβιβαστεί ως όρισμα σε μια συνάρτηση

Μεταβλητές μέλη με αρχικές τιμές

```
#include <iostream>
#include <string>
using namespace std;
class rectangle
{
    float plevra_a=0;
    float plevra_b=0;
public:
    string xroma;
    rectangle();
    rectangle(float a, float b);
};
rectangle::rectangle()
{
    cout<<"Δημιουργήθηκε ένα παραλληλόγραμμο "<<plevra_a<<"x"<<plevra_b<<endl;
}
rectangle::rectangle(float a, float b)
{
    plevra_a = a;
    plevra_b = b;
    cout<<"Δημιουργήθηκε ένα παραλληλόγραμμο "<<plevra_a<<"x"<<plevra_b<<endl;
}
int main()
{
    rectangle rec1,rec2(4,8);
    return 0;
}
```

Μεταβλητές μέλη με αρχικές τιμές

- Σε κάθε νέο αντικείμενο της κλάσης rectangle οι μεταβλητές μέλη θα έχουν αρχική τιμή 0. Οι μέθοδοι δόμησης μπορούν να αλλάξουν αυτές τις αρχικές τιμές

Παραλληλόγραμμο 0x0
Παραλληλόγραμμο 4x8

Προκαθορισμένες τιμές στις παραμέτρους μεθόδων

- Οι μέθοδοι μιας κλάσης μπορεί να διαθέτουν παραμέτρους με **προκαθορισμένες** τιμές όπως και κάθε άλλη συνάρτηση της C++
- Η παρακάτω μέθοδος της κλάσης rectangle πολλαπλασιάζει τις πλευρές ενός αντικειμένου παραλληλογράμμου με διαφορετικές κλίμακες
- Οι κλίμακες αυτές μεταβιβάζονται στις παραμέτρους της μεθόδου
- Και για τις 2 παραμέτρους ορίζεται ως προκαθορισμένη τιμή το 1

Προκαθορισμένες τιμές στις παραμέτρους μεθόδων

```
  rectangle :: scale_ab(float kla=1, float klb=1)
{
  plevra_a=plevra_a*kla;
  plevra_b=plevra_b*klb;
}
```

Επομένως η παραπάνω μέθοδος μπορεί να κληθεί με κανένα, ένα ή δυο ορίσματα:

```
 rec1.scale_ab();
 rec1.scale_ab(2);
 rec1.scale_ab(2,4);
```

Προκαθορισμένες τιμές στις παραμέτρους μεθόδων

- Αντί να χρησιμοποιήσουμε προκαθορισμένες τιμές στις παραμέτρους της, θα μπορούσαμε να πετύχουμε το ίδιο αποτέλεσμα υπερφορτώνοντας τη μέθοδο scale_ab() τρεις φορές με διαφορετικό πλήθος παραμέτρων
- **ΠΡΟΣΟΧΗ**: Από τη στιγμή που δηλωθεί μια παράμετρος με προκαθορισμένη τιμή τότε όλες οι επόμενες παράμετροι επίσης πρέπει να έχουν προκαθορισμένη τιμή
- Προκαθορισμένες τιμές στις παραμέτρους μπορούν να έχουν και οι μέθοδοι δόμησης

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

class trigono
{
public:
    float basi;
    float ypsos;
    string xroma;
    trigono(int a, int b, string xr);
    void scale_it(int klimaka);
    void display();
};
trigono::trigono(int a=0, int b=0, string xr="Κόκκινο")
{
    basi=a;
    ypsos=b;
    xroma=xr;
}
void trigono::scale_it(int klimaka=1)
{
    basi=basi*klimaka;
    ypsos=ypsos*klimaka;
}
void trigono::display()
{
    cout<<"Χρώμα:"<<xroma<<endl<<"Βάση:"<<basi<<endl<<"Υψος:"<<ypsos<<endl;
}
int main()
{
    trigono tr1,tr2(10),tr3(9,5),tr4(4,7,"Πράσινο");
    tr1.display();
    tr2.display();
    tr3.display();
    tr4.display();
    return 0;
}
```

Χρώμα: Κόκκινο
Βάση: 0
Ύψος: 0
Χρώμα: Κόκκινο
Βάση:10
Ύψος: 0
Χρώμα: Κόκκινο
Βάση: 9
Ύψος: 5
Χρώμα: Πράσινο
Βάση: 4
Ύψος: 7

Κλάσεις και δομές

- Δομές και κλάσεις έχουν την ίδια **λειτουργικότητα**
- Και στις **δομές** μπορούμε να έχουμε **private** μέλη με τη μόνη διαφορά πως αν δεν προσδιορίσουμε κάτι διαφορετικό όλα τα μέλη μιας δομής είναι δημόσια
- Οι δομές προτείνονται για οντότητες οι οποίες χρησιμοποιούνται για **αποθήκευση** δημόσιων **δεδομένων** και δεν εμπεριέχουν καμιά λειτουργικότητα (POD – Plain Old Data)
- Όταν τα δεδομένα εμπεριέχουν μια **ιδιωτικότητα**, **μεθόδους** για την επεξεργασία τους ή θέλουμε να τα χρησιμοποιήσουμε ως βάση για τη δημιουργία άλλων οντοτήτων, τότε προτιμότερη είναι η χρήση **κλάσεων**

Αντικειμενοστρεφής προγραμματισμός

- Αντί να προσαρμόσουμε το πρόβλημα στη γλώσσα προγραμματισμού που χρησιμοποιούμε, προσαρμόζουμε τη γλώσσα στο πρόβλημα που θέλουμε να λύσουμε κατασκευάζοντας τα κατάλληλα εργαλεία
- Οι κλάσεις και τα αντικείμενα χρησιμοποιούνται ώστε να μοντελοποιήσουν δεδομένα και διαδικασίες του πραγματικού κόσμου

Διαδικαστική αντιμετώπιση

- Έστω ότι θέλουμε να κρατάμε τις διαστάσεις και τα χρώματα δυο σχημάτων ορθογωνίων παραλληλογράμμων και δυο κύκλων και επιπρόσθετα να υπολογίζουμε το εμβαδόν τους
- Το πρόγραμμα που ακολουθεί δείχνει την διαδικαστική αντιμετώπιση του προβλήματος

```cpp
/* Το ακλολουθο πρόγραμμα ακουλουθεί 'διαδικαστική' προσέγγιση για την διαχείριση
τεσσάρων αντικειμένων: Δυο ορθογωνίων παραλληλογράμμων (10x5 πράσινο και 7x8 κόκκινο) και δύο κύκλων
(ακτίνας 5 γαλάζιος και ακτίνας 4 κίτρινος*/

#include <iostream>
#include <string>
using namespace std;
float emvado_p(float x, float y);
float emvado_k(float ak);

int main()
{
    float p1a=10,p1b=5,p2a=7,p2b=8;  // Οι μεταβλητές στις ποίες αποθηκέυονται τα μήκη των πλευρών των παραλληλογράμμων
    float k1r=5,k2r=4;               // Οι μεταβλητές στις ποίες αποθηκέυονται τα μήκη των ακτίνων τοων κύκλων
    string xrp1="Πράσινο",xrp2="Κόκκινο",xrk1="Γαλάζιο",xrk2="Κίτρινο"; // Μεταβλητές στις ποίες αποθηκέυονται τα χρώματα των σχημάτων
    float emp1, emk1;
    emp1=emvado_p(p1a,p1b);
    emk1=emvado_k(k2r);
    cout << "Ορθογώνιο "<< xrp1 <<" "<<p1a<<"x"<<p1b<<" E="<<emp1<<endl; // Εμφανίζει τα στοιχεία του πρώτου παραλληλογράμμου
    cout << "Κύκλος "<< xrk2 <<" R="<<k2r<<" E="<<emk1<<endl;     // Εμφανίζει τα στοιχεία του δεύτερου κύκλου
    return 0;
}

float emvado_p(float x, float y) // Συνάρτηση υπολογισμού του εμβαδού ορθογωνίου παραλληλογράμμου
{
    return x*y;
}
float emvado_k(float ak)        // Συνάρτηση υπολογισμού του εμβαδού κύκλου
{
    return 3.14*ak*ak;
}
```

Αντικειμενοστρεφής αντιμετώπιση

- Χρησιμοποιούνται οι κλάσεις rectangle και circle. Στις μεταβλητές – μέλη των κλάσεων αποθηκεύονται οι διαστάσεις και το χρώμα των σχημάτων
- Κάθε κλάση διαθέτει μέθοδο για τον υπολογισμό του εμβαδού του σχήματος

```
/* Το ακλολουθο πρόγραμμα ακουλουθεί 'διαδικαστική' προσέγγιση για την διαχείριση
τεσσάρων αντικειμένων: Δυο ορθογωνίων παραλληλογράμμων (10x5 πράσινο και 7x8 κόκκινο) και δύο κύκλων
(ακτίνας 5 γαλάζιος και ακτίνας 4 κίτρινος*/

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

class rectangle
{
public:
    float plevra_a;
    float plevra_b;
    string xroma;
    float emvado()
    {
        return plevra_a * plevra_b;
    }
    void display()
    {
        cout << "Ορθογώνιο : "<<xroma<<" "<<plevra_a<<"x"<<plevra_b<<endl;
    }
};

class circle
{
public:
    float aktina;
    string xroma;
    float emvado()
    {
        return aktina*aktina*3.14;
    }
    void display()
    {
        cout << "Κύκλος : " << xroma << " R=" << aktina << endl;
    }
};
```

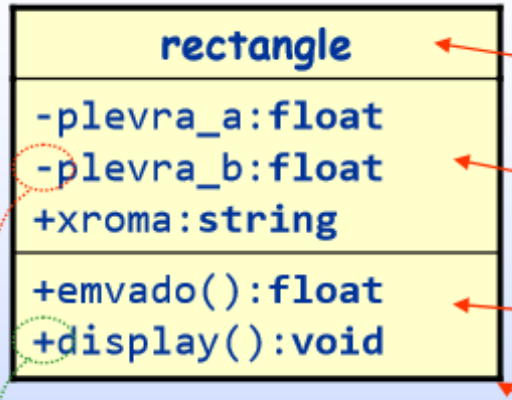
```
int main()
{
    int a,b;
    rectangle rec1,rec2;
    circle c1,c2;
    rec1.plevra_a=10;
    rec1.plevra_b=5;
    rec1.xroma="Πράσινο";
    rec2.plevra_a=7;
    rec2.plevra_b=8;
    rec2.xroma="Κόκκινο";
    c1.aktina=5;
    c1.xroma="Γαλάζιο";
    c2.aktina=4;
    c2.xroma="Κίτρινο";
    rec1.display();
    rec2.display();
    c1.display();
    c2.display();
    cout<<rec1.emvado()<<endl;
    cout<<c2.emvado()<<endl;
    return 0;

}
```

Απεικόνιση κλάσεων με τη UML

- Unified Modeling Language – Ενοποιημένη Γλώσσα Μοντελοποίησης
- Μας δίνει την δυνατότητα μέσω στοιχείων και διαγραμμάτων να απεικονίσουμε συστήματα αντικειμενοστρεφών εφαρμογών
- Ένα από τα εργαλεία που χρησιμοποιεί η UML είναι τα διαγράμματα κλάσεων τα οποία περιγράφουν τη δομή των κλάσεων μιας εφαρμογής καθώς και τη συσχέτιση των κλάσεων μεταξύ τους
- Μια κλάση σε ένα διάγραμμα κλάσεων απεικονίζεται ως ένα ορθογώνιο που αποτελείται από 3 οριζόντια τμήματα. Στο πάνω τμήμα αναφέρεται το όνομα της κλάσης, στο μεσαίο τμήμα τα χαρακτηριστικά της (μεταβλητές – μέλη) και στο κάτω τμήμα οι λειτουργίες της (μέθοδοι)
- Κάθε μεταβλητή – μέλος αναφέρεται ως ±όνομα: τύπος = αρχική τιμή (+ για public, - για private)
- Κάθε μέθοδος αναφέρεται ως ±όνομα (τύπος_παρ1, τύπος_παρ2…): τύπος

UML - Διαγράμματα κλάσεων



Όνομα κλάσης

Μεταβλητές-μέλη (data members)

Συναρτήσεις-μέλη - μέθοδοι (function members - methods)

Το - συμβολίζει ιδιωτικό μέλος

Το + συμβολίζει δημόσιο μέλος

Παρατηρούμε ότι στα διαγράμματα κλάσεων πρώτα αναφέρεται το όνομα του μέλους και μετά ο τύπος του.

UML Διαγράμματα Κλάσεων

- Στα διαγράμματα κλάσεων η συσχέτιση συμβολίζεται με ένα απλό βέλος από τη μια κλάση στην άλλη