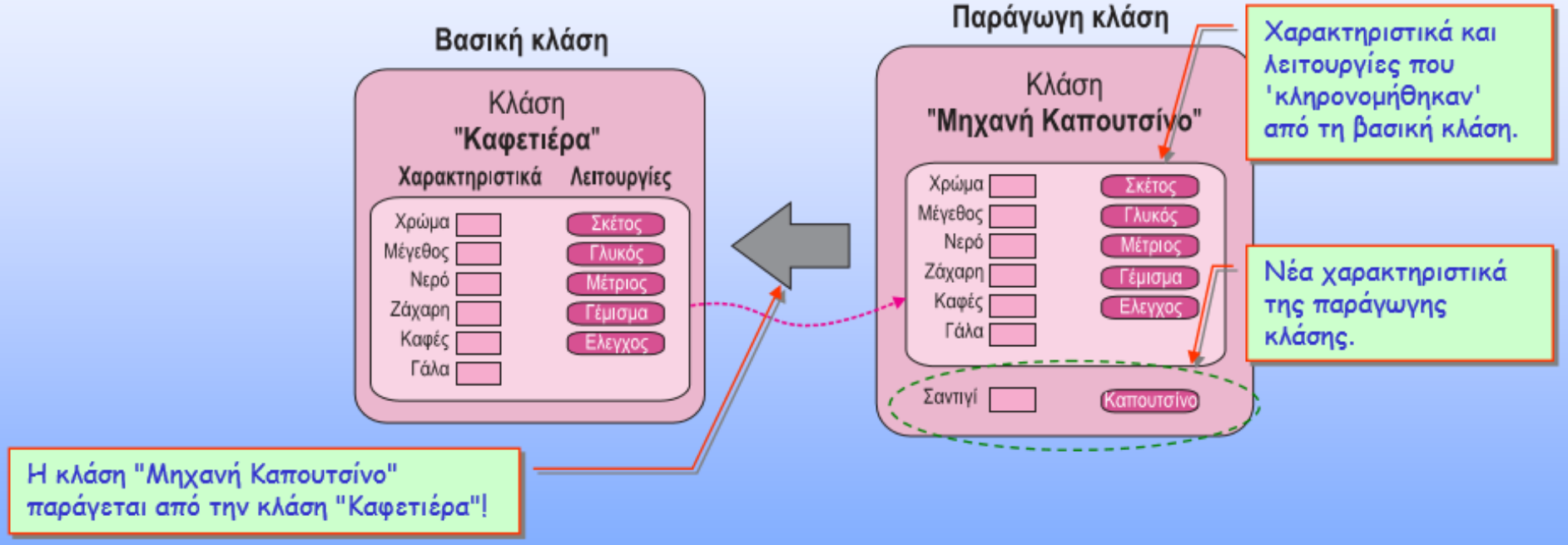
Κληρονομικότητα

Κληρονομικότητα (inheritance)

Η κληρονομικότητα (inheritance) αποτελεί ένα από τα πιο ισχυρά χαρακτηριστικά του αντικειμενοστρεφούς προγραμματισμού.
Η κληρονομικότητα δίνει τη δυνατότητα να παραχθεί μια νέα κλάση από μια υπάρχουσα κλάση.

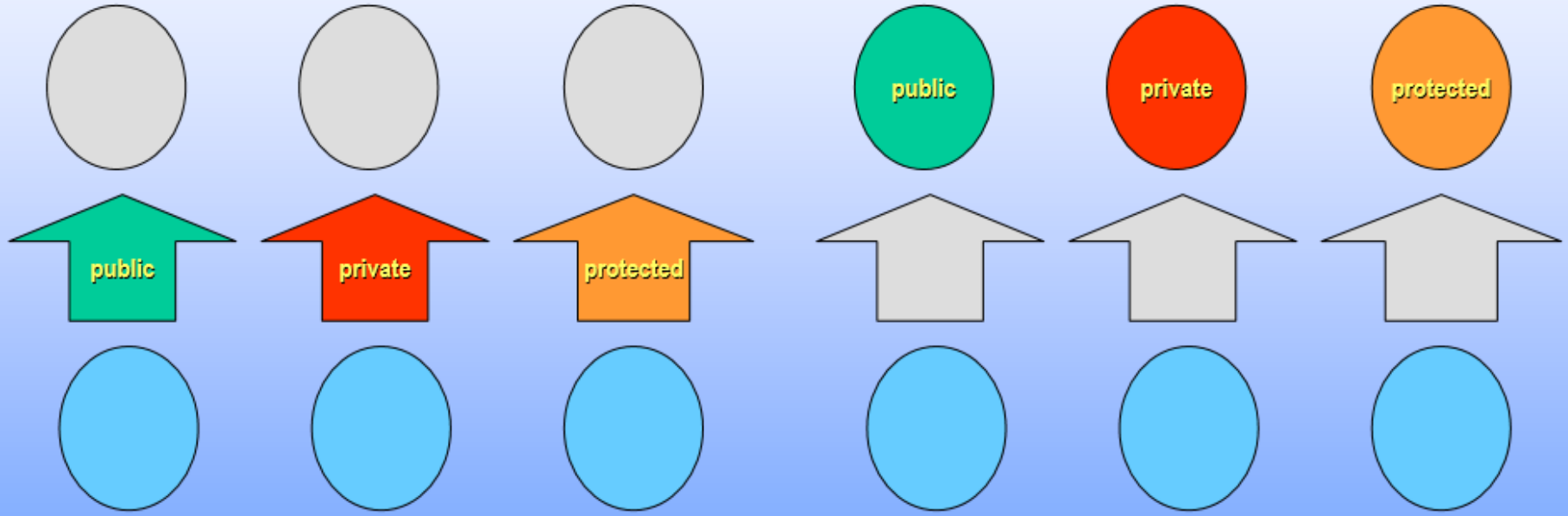


Η **παράγωγη** κλάση **κληρονομεί** όλα τα χαρακτηριστικά της **βασικής** κλάσης. Η παράγωγη κλάση μπορεί να τροποποιήσει τα χαρακτηριστικά που κληρονόμησε, αλλά να προσθέσει και νέα. Η βασική κλάση παραμένει αναλλοίωτη.

Κληρονομικότητα (inheritance)

Για να κατανοήσουμε τη κληρονομικότητα, πρέπει να καταλάβουμε ότι η πρόσβαση στα πράγματα που κληρονομήσαμε εξαρτάται από δύο παράγοντες:

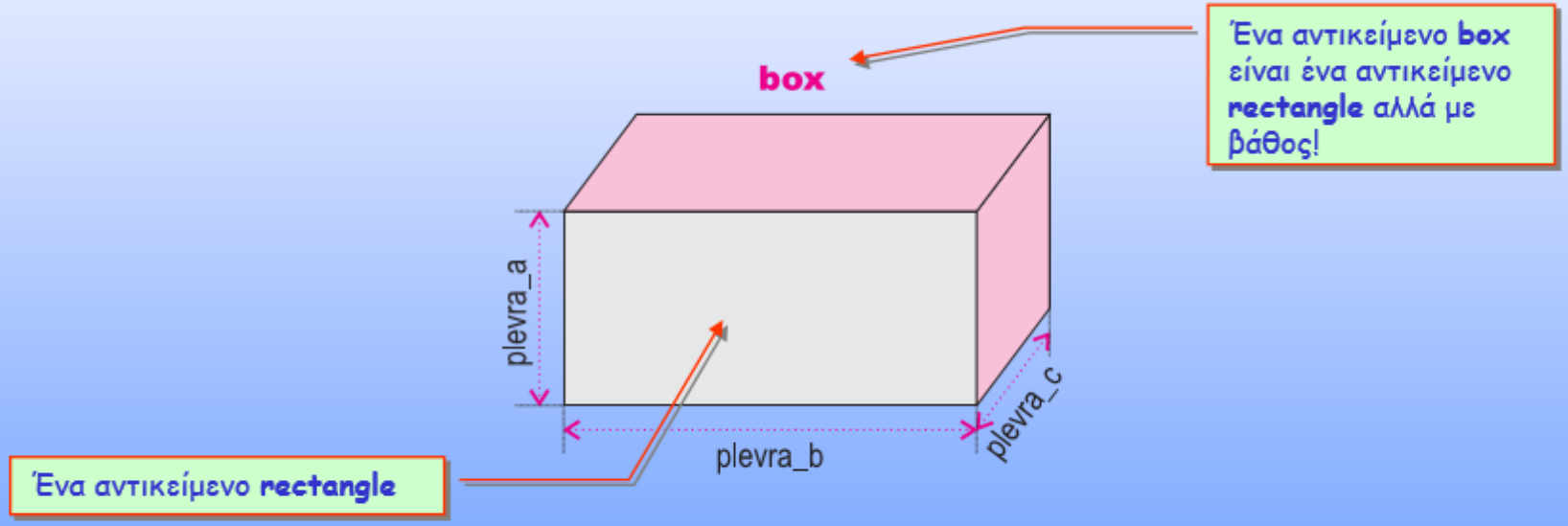
- **Με ποιο τρόπο τα κληρονομήσαμε**
- **Τι είναι αυτό που κληρονομήσαμε**



Μπορούμε να κληρονομήσουμε κάτι με δημόσια (public), ιδιωτική (private) ή προστατευμένη (protected) πρόσβαση. Αυτό που κληρονομούμε μπορεί να είναι δημόσιο, ιδιωτικό ή προστατευμένο. Υπάρχουν επομένως **εννέα διαφορετικοί συνδυασμοί κληρονομικότητας** (για τα προστατευμένα μέλη θα μιλήσουμε στη συνέχεια).

Κληρονομικότητα (inheritance)

Ας υποθέσουμε ότι θέλουμε να διαχειριστούμε ορθογώνια παραλληλόγραμμα τριών διαστάσεων τα οποία θα έχουν και βάθος. Σε αυτή την περίπτωση, εκτός από τις διαστάσεις plevra_a και plevra_b, θα πρέπει να κρατάμε και την τρίτη διάσταση του αντικειμένου plevra_c. Ο προγραμματιστής λοιπόν θα πρέπει να φτιάξει μια νέα κλάση box για την διαχείριση των τριδιάστατων αυτών ορθογωνίων παραλληλογράμμων. Η κλάση αυτή θα έχει μικρές διαφορές από την κλάση rectangle: Θα πρέπει να προστεθεί μία μεταβλητή-μέλος plevra_c, καθώς και οι απαραίτητες συναρτήσεις-μέλη για την διαχείριση των αντικειμένων τύπου box.



Η C++ δίνει τη δυνατότητα στον προγραμματιστή να δημιουργήσει τη νέα κλάση **box** από την ήδη υπάρχουσα **rectangle** κληρονομώντας ταυτόχρονα τα χαρακτηριστικά και τις λειτουργίες της.

Κληρονομικότητα (inheritance)

Παράγωγη κλάση

Προσδιοριστικό πρόσβασης κληρονομικότητας

Βασική κλάση

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

```
class box:public rectangle
{
    float plevra_c;
public:
    float ogos()
    {
        return emvado()*plevra_c;
    }
    void set_c(float c)
    {
        plevra_c=c;
    }
};
```

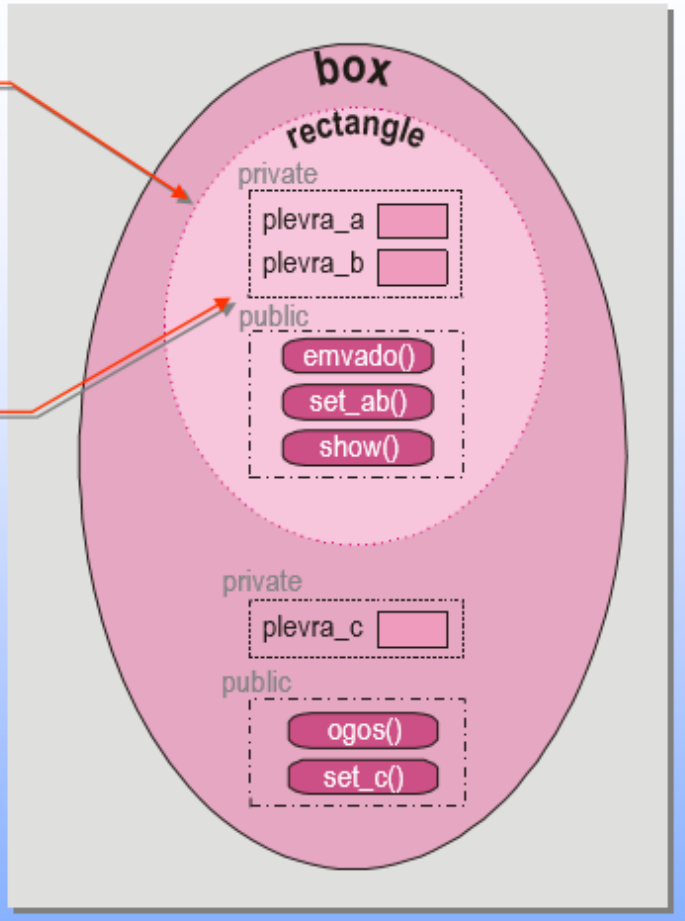
Η κλάση **box** παράγεται από τη κλάση **rectangle**

Στη δήλωση της νέας κλάσης μετά από το όνομά της ακολουθεί ο τελεστής :, **το προσδιοριστικό πρόσβασης** (στη συγκεκριμένη περίπτωση το **public**) και το όνομα της βασικής κλάσης. Το προσδιοριστικό πρόσβασης **προσδιορίζει τον τρόπο** με τον οποίο κληρονομούνται τα μέλη της βασικής τάξης.

Αντικείμενο της παράγωγης κλάσης

Κάθε αντικείμενο της κλάσης **box** περιέχει ένα (υπο)αντικείμενο της κλάσης **rectangle**.

Η πρόσβαση στα μέλη του (υπο)αντικειμένου εξαρτάται από το προσδιοριστικό πρόσβασης κληρονομικότητας.

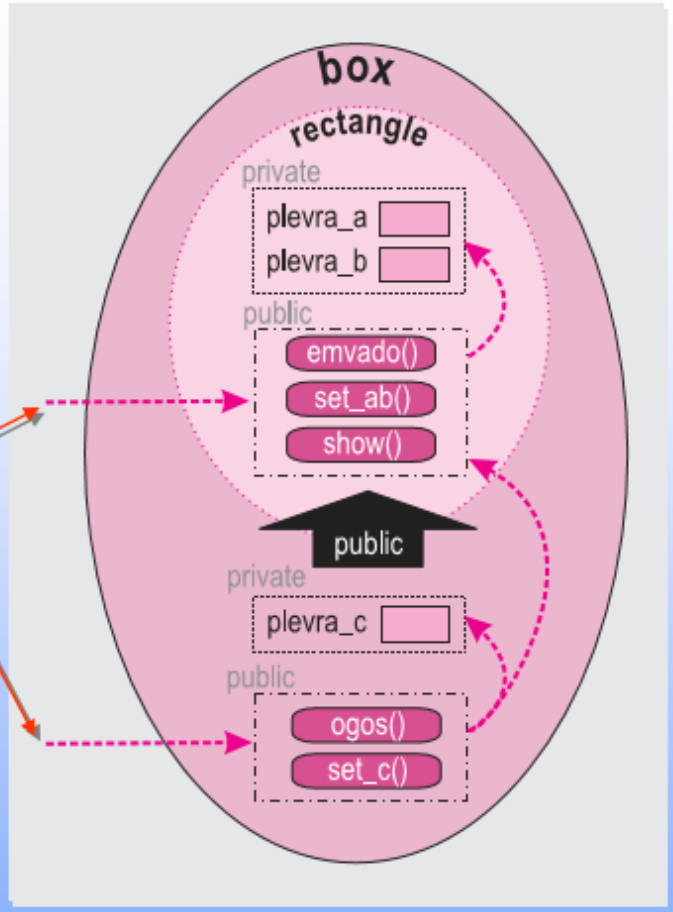


Κάθε αντικείμενο της παράγωγης κλάσης περιέχει ένα (υπο)αντικείμενο της βασικής κλάσης. Το αντικείμενο της παράγωγης κλάσης περιέχει επί πλέον τα νέα μέλη της κλάσης. Η πρόσβαση στα μέλη του (υπο)αντικειμένου που κληρονομήθηκε από τη βασική κλάση καθορίζεται από το προσδιοριστικό πρόσβασης κληρονομικότητας κατά τον ορισμό της παράγωγης κλάσης.

Κληρονομικότητα με δημόσια πρόσβαση (public) βασικής κλάσης

Η χρήση του προσδιοριστικού **public** σημαίνει ότι όλα τα δημόσια μέλη της βασικής κλάσης θα είναι και δημόσια μέλη της παράγωγης. Τα ιδιωτικά μέλη της βασικής κλάσης κληρονομούνται σαν ιδιωτικά, **χωρίς** όμως να είναι προσβάσιμα ούτε από την παράγωγη κλάση.

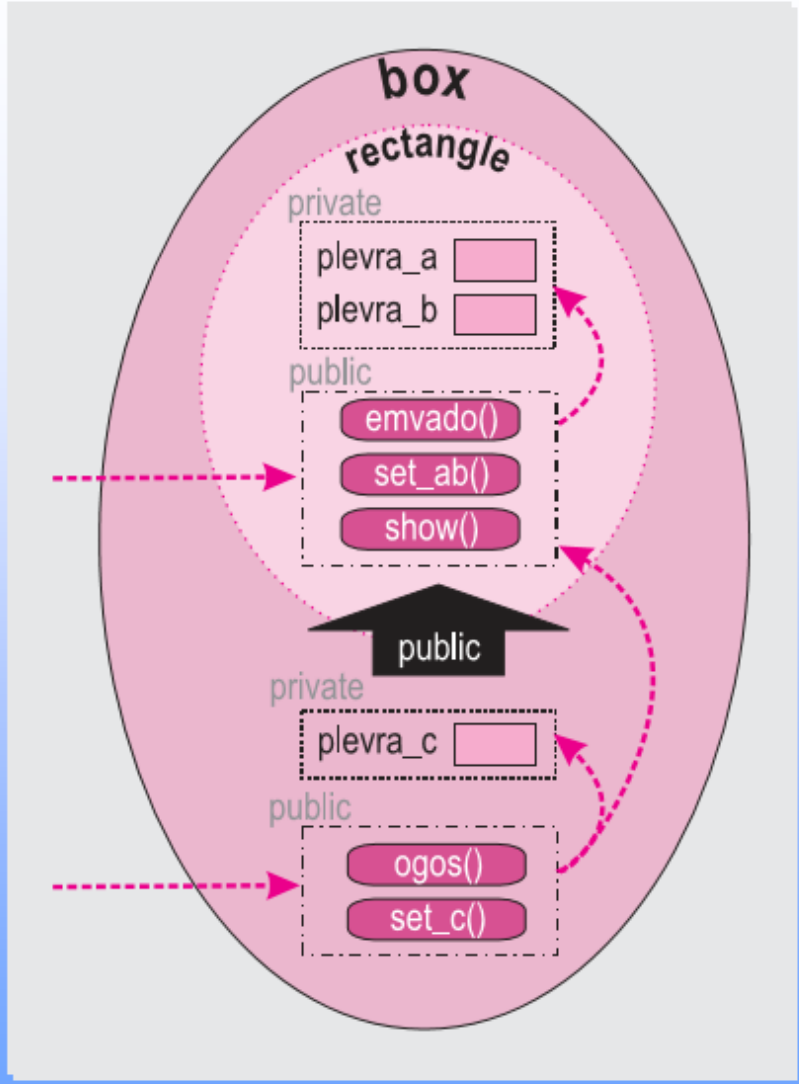
Τα **δημόσια** μέλη τόσο της **παράγωγης** όσο και τη **βασικής** κλάσης, είναι **προσβάσιμα** και από συναρτήσεις **εκτός της κλάσης**!



Η διεπαφή των αντικειμένων της κλάσης **box** με το προγραμματιστικό περιβάλλον γίνεται τόσο μέσω των **δημόσιων μελών** της κλάσης **box**, όσο και μέσω των **δημόσιων μελών** της κλάσης **rectangle** (υποδεικνύεται από τα βέλη εκτός της κλάσης).

Κληρονομικότητα με δημόσια πρόσβαση (public) βασικής κλάσης

- Η κλάση **box** έχει τρεις μεταβλητές-μέλη τις **plevra_a**, **plevra_b** και **plevra_c**. Οι δύο πρώτες έχουν κληρονομηθεί από την **rectangle** με ιδιωτική όμως πρόσβαση.
- Οι μεταβλητές-μέλη plevra_a και plevra_b έχουν κληρονομηθεί ως ιδιωτικές και οι νέες συναρτήσεις-μέλη της box δεν έχουν πρόσβαση σε αυτές.
- Οι συναρτήσεις-μέλη της box, ogos() και set_c() έχουν πρόσβαση στη μεταβλητή-μέλος plevra_c.
- Καμία από τις τρεις μεταβλητές μέλη δεν είναι προσβάσιμη από αντικείμενα της κλάσης box.
- Η κλάση box διαθέτει τώρα πέντε συναρτήσεις-μέλη: Τις emvado(), show() και set_ab() που κληρονόμησε από τη rectangle και τις δικές της ogos() και set_c(). Και οι πέντε συναρτήσεις-μέλη της κλάσης box είναι δημόσιες και προσβάσιμες από τα αντικείμενα της κλάσης.



Παράδειγμα

- Το παρακάτω πρόγραμμα χρησιμοποιεί ως βασική την κλάση **rectangle** από την οποία παράγεται η κλάση **rec3D** και δείχνει την απλή χρήση της κληρονομικότητας. Η βασική κλάση **rectangle** διαθέτει επιπλέον το προστατευόμενο μέλος **xroma** και η νέα κλάση **rec3D** το επίσης προστατευόμενο μέλος **color**

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

```
class rec3D:public rectangle
{
    float plevra_c;
protected:
    string color;
public:
    float ogos();
    void set_c(float c, string xr);
};
float rec3D::ogos()
{
    return emvado()*plevra_c;
}
void rec3D::set_c(float c, string xr)
{
    plevra_c=c;
    color=xr;
}
```

```
int main()
{
    rectangle a;
    rec3D b;
    a.set_ab(10,20,"Κόκκινο");
    cout << a.emvado() << endl;
    a.show();
    b.set_ab(23,3,"Μαύρο");
    b.set_c(4,"Πράσινο");
    cout<<"Όγκος="<<b.ogos()<<endl;
    cout<<"Εμβαδό="<<b.emvado()<<endl;
    return 0;
}
```

200
Κόκκινο 10x20
Όγκος=276
Εμβαδό=69

Προστατευμένα (protected) μέλη μιας κλάσης

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

Το προσδιοριστικό **protected**: καθορίζει ότι τα μέλη που ακολουθούν είναι προστατευμένα.

Τα προστατευμένα μέλη μιας κλάσης, όσον αφορά στην ίδια τη κλάση, συμπεριφέρνονται όπως ακριβώς και τα ιδιωτικά:
Στα προστατευμένα μέλη έχουν πρόσβαση μόνο οι συναρτήσεις-μέλη της κλάσης.

```
int main()
{
    rectangle r1;
    r1.set_ab(5,15);
    r1.plevra_a=10;
    r1.plevra_b=20;
    ....
    return 0;
}
```

ΛΑΘΟΣ προτάσεις! Δεν υπάρχει πρόσβαση στα protected μέλη!

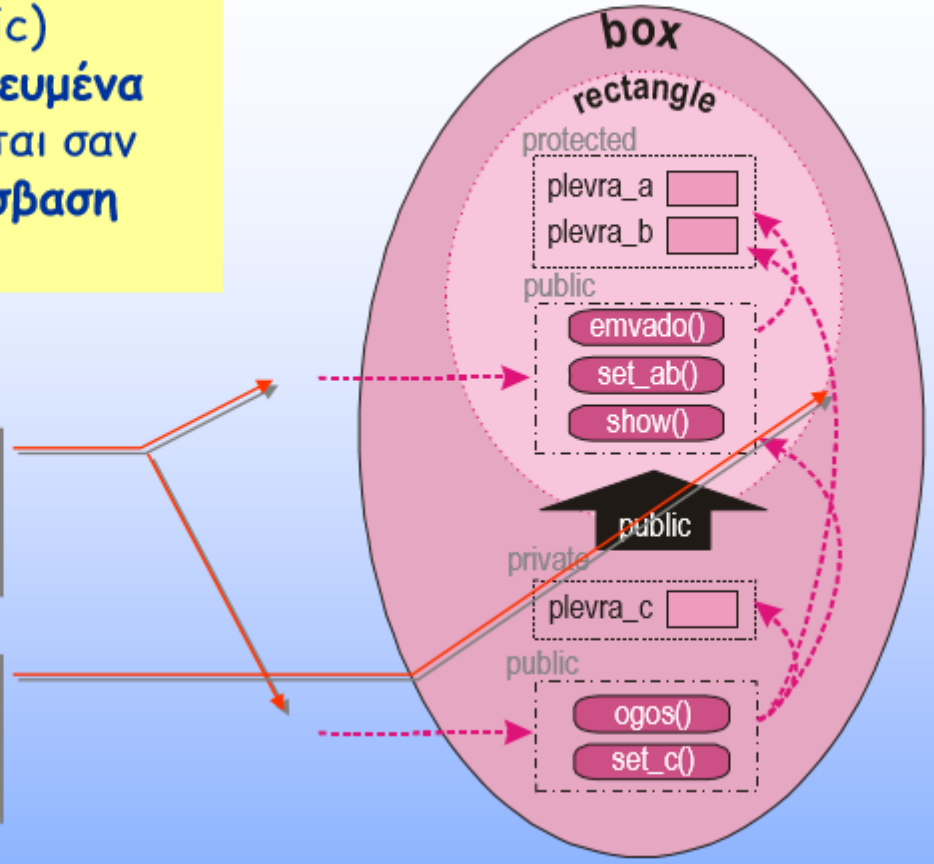
Η διαφορετικότητα των προστατευμένων μελών γίνεται εμφανής **μόνο** στην περίπτωση που η κλάση χρησιμοποιηθεί ως βασική κλάση για την παραγωγή μιας νέας κλάσης. Τα **προστατευμένα** μέλη της **βασικής** κλάσης **είναι προσβάσιμα** από την **παράγωγη** κλάση.

Κληρονομικότητα με δημόσια πρόσβαση (public) βασικής κλάσης

Στη κληρονομικότητα με δημόσια (puplic) πρόσβαση βασικής κλάσης, τα **προστατευμένα** μέλη της βασικής κλάσης κληρονομούνται σαν **ιδιωτικά**. Η παράγωγη κλάση **έχει πρόσβαση** στα **προστατευμένα** μέλη της βασικής.

Τα **δημόσια** μέλη τόσο της **παράγωγης** όσο και τη **βασικής** κλάσης, είναι **προσβάσιμα** και από συναρτήσεις **εκτός της κλάσης**!

Τα **προστατευμένα** μέλη της **βασικής** κλάσης, είναι **προσβάσιμα** από συναρτήσεις **της παράγωγης κλάσης**!



Κληρονομικότητα με δημόσια πρόσβαση (public) βασικής κλάσης

Ο πίνακας συνοψίζει όλες τις πιθανές περιπτώσεις στη περίπτωση κληρονομικότητας με δημόσια (public) πρόσβαση στη βασική κλάση.

Μέλη βασικής κλάσης	Πρόσβαση από αντικείμενα της βασικής κλάσης	Πρόσβαση από την παράγωγη κλάση	Πρόσβαση από αντικείμενα της παράγωγης κλάσης
public	ΝΑΙ	ΝΑΙ	ΝΑΙ
private	ΟΧΙ	ΟΧΙ	ΟΧΙ
protected	ΟΧΙ	ΝΑΙ	ΟΧΙ

Παράδειγμα

- Στο παρακάτω πρόγραμμα η δήλωση των μεταβλητών – μελών της βασικής κλάσης **rectangle** ως προστατευμένων δίνει τη δυνατότητα στην παράγωγη κλάση **rec3D** να τις προσπελάζει

```
#include <iostream>
using namespace std;
class rectangle
{
protected:
    float plevra_a;
    float plevra_b;
public:
    float emvado(){return plevra_a * plevra_b ;}
    void set_ab(float a, float b);
};
void rectangle::set_ab(float a, float b)
{
    plevra_a=a;
    plevra_b=b;
}
class rec3D:public rectangle
{
    float plevra_c;
public:
    float ogos(){ return plevra_a * plevra_b * plevra_c;}
    void set_c(float c){plevra_c=c;}
    void show() {cout<<plevra_a<<"x"<<plevra_b<<"x"<<plevra_c<<endl;}
};
```

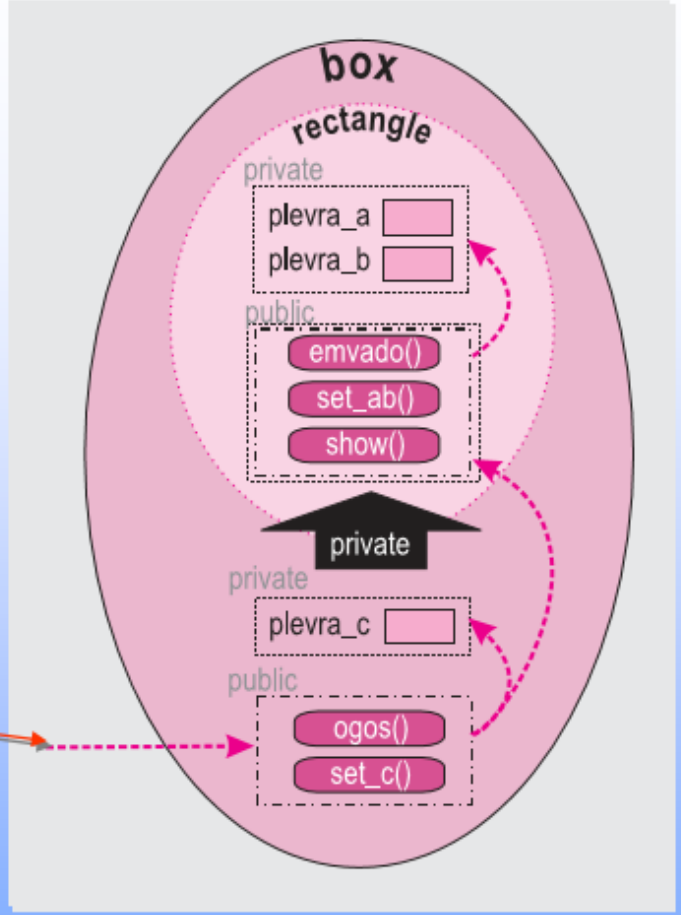
```
int main()
{
    rec3D b;
    b.set_ab(23,3);
    b.set_c(4);
    b.show();
    cout<<"Όγκος="<< b.ogos()<<endl;
    return 0;
}
```

23x3x4

Όγκος=276

Κληρονομικότητα με ιδιωτική (private) πρόσβαση βασικής κλάσης

Μια κλάση μπορεί να παραχθεί από μια βασική κλάση με ιδιωτική (**private**) πρόσβαση. Σε αυτή την περίπτωση όλα τα δημόσια και προστατευμένα μέλη της βασικής κλάσης γίνονται ιδιωτικά της παράγωγης κλάσης. Αυτό σημαίνει ότι ενώ είναι προσβάσιμα από τα υπόλοιπα μέλη της παράγωγης κλάσης, δεν είναι διαθέσιμα σε κώδικα εκτός της κλάσης. Η παράγωγη κλάση εξακολουθεί να μην έχει πρόσβαση στα ιδιωτικά μέλη της βασικής κλάσης.

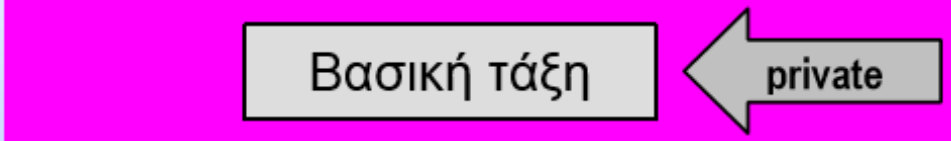


Μόνο τα **δημόσια** μέλη της **παράγωγης** κλάσης, είναι **προσβάσιμα** από συναρτήσεις **εκτός της κλάσης**!

Η διεπαφή των αντικειμένων της κλάσης **box** με το προγραμματιστικό περιβάλλον γίνεται μόνο μέσω των **δημόσιων μελών** της κλάσης **box**, όπως υποδεικνύεται από το βέλος εκτός της κλάσης.

Κληρονομικότητα με ιδιωτική (private) πρόσβαση βασικής κλάσης

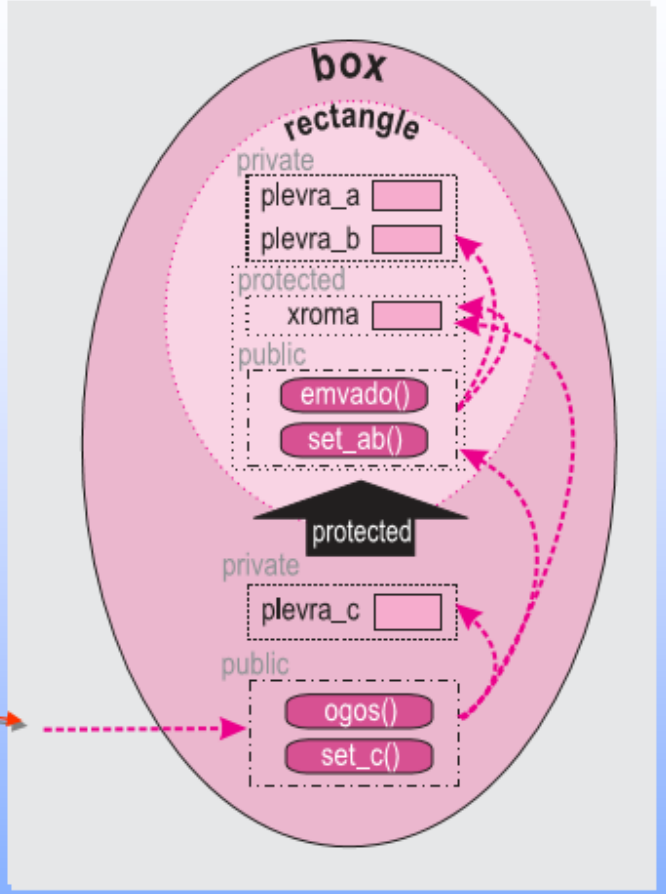
Ο πίνακας συνοψίζει όλες τις πιθανές περιπτώσεις στη περίπτωση κληρονομικότητας με ιδιωτική (private) πρόσβαση στη βασική κλάση.



Μέλη βασικής κλάσης	Πρόσβαση από αντικείμενα της βασικής κλάσης	Πρόσβαση από την παράγωγη κλάση	Πρόσβαση από αντικείμενα της παράγωγης κλάσης
public	ΝΑΙ	ΝΑΙ	ΟΧΙ
private	ΟΧΙ	ΟΧΙ	ΟΧΙ
protected	ΟΧΙ	ΝΑΙ	ΟΧΙ

Κληρονομικότητα με προστατευμένη (protected) πρόσβαση βασικής κλάσης

Όταν μια βασική κλάση κληρονομείται ως **προστατευμένη (protected)**, τότε όλα τα προστατευμένα και τα δημόσια μέλη της, γίνονται προστατευμένα μέλη της παράγωγης κλάσης. Η παράγωγη κλάση έχει πρόσβαση τόσο στα δημόσια όσο και στα προστατευμένα μέλη που κληρονόμησε. Η παράγωγη κλάση εξακολουθεί να μην έχει πρόσβαση στα ιδιωτικά μέλη της βασικής κλάσης.

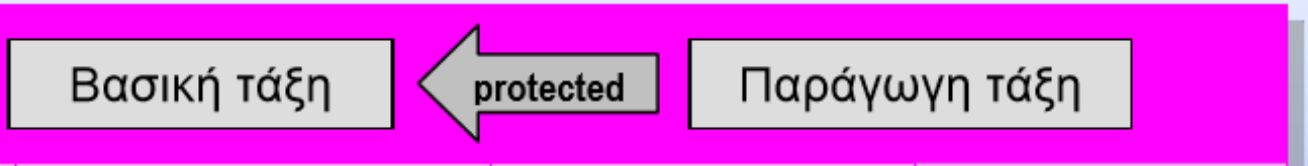


Μόνο τα **δημόσια** μέλη της **παράγωγης** κλάσης, είναι **προσβάσιμα** από συναρτήσεις **εκτός της κλάσης**!

Η διεπαφή των αντικειμένων της κλάσης **box** με το προγραμματιστικό περιβάλλον γίνεται μόνο μέσω των **δημόσιων μελών** της κλάσης **box**, όπως υποδεικνύεται από το βέλος εκτός της κλάσης.

Κληρονομικότητα με προστατευμένη (protected) πρόσβαση βασικής κλάσης

Ο πίνακας συνοψίζει όλες τις πιθανές περιπτώσεις στη περίπτωση κληρονομικότητας με προστατευμένη (protected) πρόσβαση στη βασική κλάση.



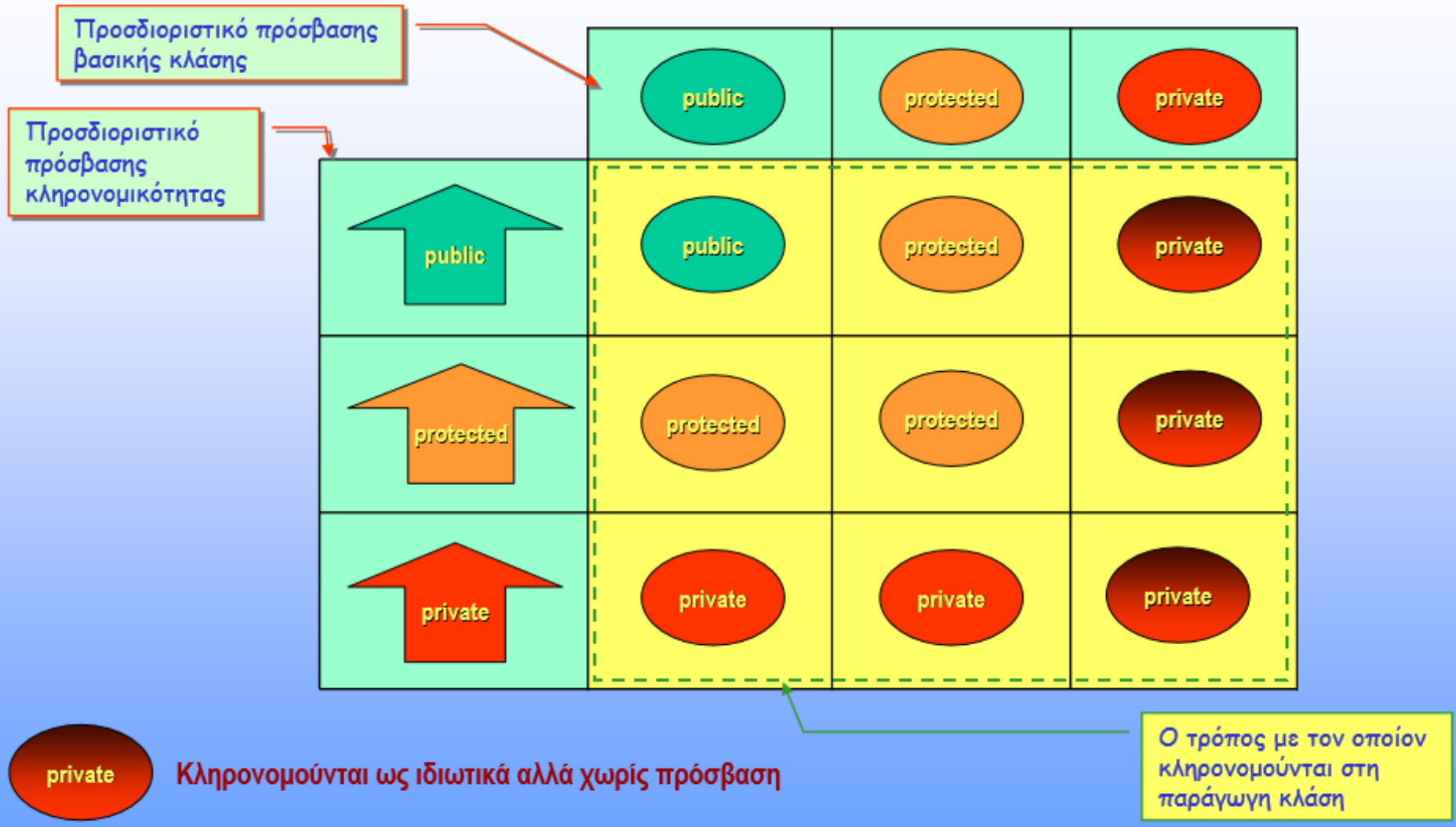
Μέλη βασικής κλάσης	Πρόσβαση από αντικείμενα της βασικής κλάσης	Πρόσβαση από την παράγωγη κλάση	Πρόσβαση από αντικείμενα της παράγωγης κλάσης
public	ΝΑΙ	ΝΑΙ	ΟΧΙ
private	ΟΧΙ	ΟΧΙ	ΟΧΙ
protected	ΟΧΙ	ΝΑΙ	ΟΧΙ

Οι πιθανές περιπτώσεις κληρονομικότητας

Προσδιοριστικό πρόσβασης κληρονομικότητας	Μέλη βασικής κλάσης	Κληρονομούνται από την παράγωγη κλάση ως ...
public	public	Δημόσια (public)
	private	Ιδιωτικά (private) χωρίς πρόσβαση
	protected	Προστατευμένα (protected)
private	public	Ιδιωτικά (private)
	private	Ιδιωτικά (private) χωρίς πρόσβαση
	protected	Ιδιωτικά (private)
protected	public	Προστατευμένα (protected)
	private	Ιδιωτικά (private) χωρίς πρόσβαση
	protected	Προστατευμένα (protected)

- Τα μέλη τα οποία έχουν κληρονομηθεί ως **δημόσια** είναι προσβάσιμα από τα υπόλοιπα μέλη της παράγωγης κλάσης, καθώς και από κώδικα εκτός της κλάσης.
- Τα μέλη τα οποία έχουν κληρονομηθεί ως **ιδιωτικά** είναι προσβάσιμα από τα υπόλοιπα μέλη της παράγωγης κλάσης, αλλά όχι από τον κώδικα εκτός της κλάσης.
- Τα μέλη τα οποία έχουν κληρονομηθεί ως **ιδιωτικά χωρίς πρόσβαση** δεν είναι προσβάσιμα από τα υπόλοιπα μέλη της παράγωγης κλάσης, ούτε και από τον κώδικα εκτός της κλάσης.
- Τα μέλη τα οποία έχουν κληρονομηθεί ως **προστατευμένα** είναι προσβάσιμα από τα υπόλοιπα μέλη της παράγωγης κλάσης, αλλά όχι από τον κώδικα εκτός της κλάσης.

Οι πιθανές περιπτώσεις κληρονομικότητας



Πρόσβαση από τα μέλη και τα αντικείμενα της παράγωγης κλάσης

```
class rectangle
{
private:
    float plevra_a;
    float plevra_b;
protected:
    string xroma;
public:
    float emvado()
    {
        return plevra_a * plevra_b;
    }
    void show()
    {
        cout<<plevra_a<<"x"
        <<plevra_b<<endl;
        cout<<xroma<<endl;
    }
    void set_ab(float a,float b,string x)
    {
        plevra_a=a;
        plevra_b=b;
        xroma=x;
    }
};
```

public

```
class box:public rectangle
{
    float plevra_c;
public:
    float ogos()
    {
        return emvado()*plevra_c;
    }
    void set_c(float c)
    {
        plevra_c=c;
    }
    void test()
    {
      ✘ plevra_a=10;
        xroma="red";
    }
};
```

```
int main()
{
    box b1;
    b1.ogos();
    b1.set_ab(10,20,"red");
    b1.set_c(30);
 ✘  b1.xroma="black";
    b1.show();
 ✘  b1.plevra_c=40;
    .....
}
```

Πρόσβαση από τα μέλη και τα αντικείμενα της παράγωγης κλάσης

```
class rectangle
{
private:
    float plevra_a;
    float plevra_b;
protected:
    string xroma;
public:
    float emvado()
    {
        return plevra_a * plevra_b;
    }
    void show()
    {
        cout<<plevra_a<<"x"
        <<plevra_b<<endl;
        cout<<xroma<<endl;
    }
    void set_ab(float a,float b,string x)
    {
        plevra_a=a;
        plevra_b=b;
        xroma=x;
    }
};
```

private

```
class box:private rectangle
{
    float plevra_c;
public:
    float ogos()
    {
        return emvado()*plevra_c;
    }
    void set_c(float c)
    {
        plevra_c=c;
    }
    void test()
    {
      plevra_a=10;
        xroma="red";
    }
};
```

```
int main()
{
    box b1;
    b1.ogos();
    b1.set_ab(10,20,"red");
    b1.set_c(30);
    b1.xroma="black";
    b1.show();
    b1.plevra_c=40;
    .....
}
```

Πρόσβαση από τα μέλη και τα αντικείμενα της παράγωγης κλάσης

```
class rectangle
{
private:
    float plevra_a;
    float plevra_b;
protected:
    string xroma;
public:
    float emvado()
    {
        return plevra_a * plevra_b;
    }
    void show()
    {
        cout<<plevra_a<<"x"
        <<plevra_b<<endl;
        cout<<xroma<<endl;
    }
    void set_ab(float a,float b,string x)
    {
        plevra_a=a;
        plevra_b=b;
        xroma=x;
    }
};
```

protected

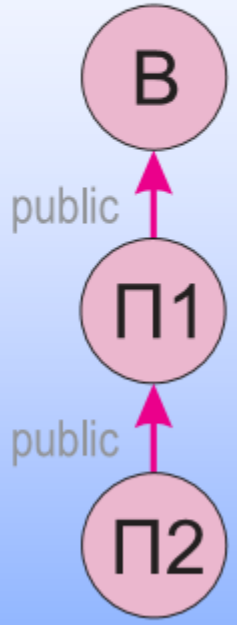
```
class box:protected rectangle
{
    float plevra_c;
public:
    float ogos()
    {
        return emvado()*plevra_c;
    }
    void set_c(float c)
    {
        plevra_c=c;
    }
    void test()
    {
      ✘ plevra_a=10;
        xroma="red";
    }
};
```

```
int main()
{
    box b1;
    b1.ogos();
 ✘  b1.set_ab(10,20,"red");
    b1.set_c(30);
 ✘  b1.xroma="black";
 ✘  b1.show();
 ✘  b1.plevra_c=40;
    .....
}
```

Διαδοχικά επίπεδα κληρονομικότητας

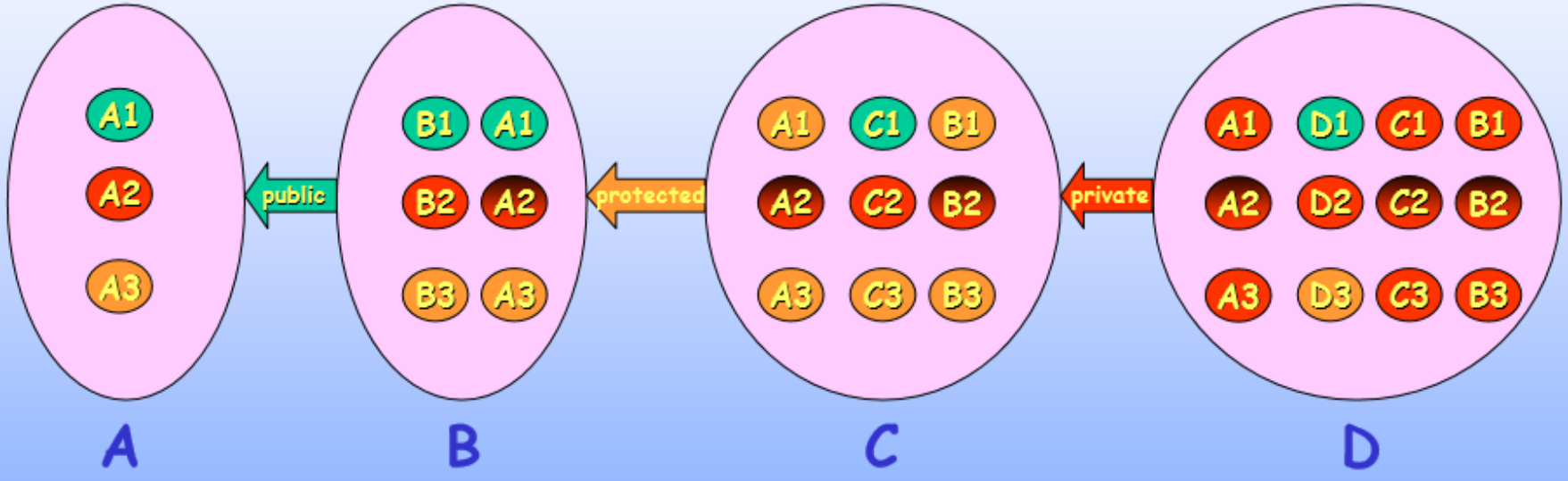
Αρχίζουμε να συζητάμε για διαδοχική κληρονομικότητα, από τη στιγμή που μια παράγωγη κλάση αποτελεί τη βασική κλάση για ένα επόμενο επίπεδο κληρονομικότητας.

Αν θεωρήσουμε ως βασική κλάση την κλάση Β, Π1 μια κλάση που παράγεται από τη Β και Π2 μια κλάση η οποία παράγεται από την Π1, το διπλανό γράφημα κληρονομικότητας δείχνει τη σχέση των τριών αυτών κλάσεων. Υπενθυμίζουμε ότι τα βέλη συμβολίζουν την έννοια "παράγεται από" και δείχνουν πάντα προς τη βασική κλάση. Επίσης στα βέλη μπορεί να σημειώνεται το είδος της πρόσβασης στη βασική κλάση.



Πολλαπλά επίπεδα κληρονομικότητας

Αρχίζουμε να συζητάμε για πολλαπλή κληρονομικότητα, από τη στιγμή που μια παράγωγη κλάση αποτελεί τη βασική κλάση για ένα επόμενο επίπεδο κληρονομικότητας.



Δημόσιο μέλος

Ιδιωτικό μέλος

Προστατευμένο μέλος

Ιδιωτικό μέλος χωρίς πρόσβαση από τα μέλη της κλάσης

Παράδειγμα

- Στο παρακάτω πρόγραμμα η κλάση **rec3D** παράγεται από την **rectangle** και η **box** από την **rec3D**. Η νέα κλάση **box** προσθέτει βάρος στα αντικείμενα – κουτιά μέσω της ιδιωτικής μεταβλητής – μέλους **varos**. Η κλάση διαθέτει επίσης δυο μεθόδους, μια για τη καταχώριση του βάρους και μια για τον υπολογισμό της περιμέτρου ενός αντικειμένου – κουτιού. Η πρόσβαση και για τα δυο επίπεδα κληρονομικότητας είναι δημόσια

```cpp
#include <iostream>
using namespace std;
class rectangle
{
protected:
    float plevra_a;
    float plevra_b;
public:
    float emvado(){return plevra_a * plevra_b;}
    void show() {cout<<plevra_a<<"x"<<plevra_b<<endl;}
    void set_ab(float a, float b){plevra_a=a; plevra_b=b;}
};
class rec3D:public rectangle
{
    float plevra_c;
public:
    float ogos(){return plevra_a * plevra_b * plevra_c;}
    void set_c(float c){plevra_c=c;}
    float get_c(){return plevra_c;}
    void show() {cout<<plevra_a<<"x"<<plevra_b<<"x"<<plevra_c<<endl;}
};
```

```
class box:public rec3D
{
    float varos;
public:
    void set_varos(float v){varos=v;}
    float perimetros()
    {
        return 4*(plevra_a+plevra_b+get_c());
    }
};
int main()
{
    rec3D b;
    box c;
    b.set_ab(23,3);
    b.set_c(4);
    b.show();
    c.set_ab(7,8);
    c.set_c(9);
    cout<<"Ογκος="<<c.ogos()<<endl;
    cout<<"Πρίμετρος="<<c.perimetros()<<endl;
    c.show();
    return 0;
}
```

23x3x4
Όγκος=504
Περίμετρος=96
7x8x9

Υποσκελισμός συναρτήσεων βασικής κλάσης (function override)

Όταν η παράγωγη κλάση διαθέτει μια συνάρτηση-μέλος με ίδιο όνομα με συνάρτηση-μέλος της βασικής κλάσης, τότε η συνάρτηση-μέλος της παράγωγης κλάσης **υποσκελίζει** (override) την αντίστοιχη συνάρτηση της βασικής κλάσης. Δηλαδή εκτελείται εκείνη αντί της αντίστοιχης συνάρτησης-μέλος της βασικής κλάσης.

Κληρονομικότητα και συναρτήσεις δόμησης και αποδόμησης

Όταν **δημιουργείται** ένα αντικείμενο μιας παράγωγης κλάσης, **πρώτα** καλείται η συνάρτηση **δόμησης** της **βασικής** κλάσης και **κατόπιν** της **παράγωγης** κλάσης. Όταν **καταστρέφεται** ένα αντικείμενο μιας παράγωγης κλάσης, **πρώτα** καλείται η συνάρτηση **αποδόμησης** της **παράγωγης** κλάσης και μετά της **βασικής**.

Στη περίπτωση που υπάρχουν **πολλά επίπεδα κληρονομικότητας**, όταν δημιουργείται ένα αντικείμενο της παράγωγης κλάσης, τότε οι συναρτήσεις δόμησης καλούνται με τη σειρά που έχουν παραχθεί οι κλάσεις (από την υψηλότερη προς της χαμηλότερη ιεραρχία), ενώ όταν καταστρέφεται το αντικείμενο, οι συναρτήσεις αποδόμησης καλούνται με την αντίστροφη σειρά.

Υπέρβαση μεθόδων (method override) της βασικής κλάσης

```
class rectangle
{
public:
    float plevra_a;
    float plevra_b;
    float emvado()
    {
        return plevra_a * plevra_b;
    }
    void show()
    {
        cout<<plevra_a<<"x"
        <<plevra_b<<endl;
    }
};
```

public

```
class box:public rectangle
{
public:
    float plevra_c;
    float ogos()
    {
        return emvado()*plevra_c;
    }
    void show()
    {
        cout<<plevra_a<<"x"
            <<plevra_b<<"x"
            <<plevra_c<<endl;
    }
};
```

Η μέθοδος show() της κλάσης box υπερβαίνει τη μέθοδο show() της rectangle

```
int main()
{
    box b1;
    b1.plevra_a=10;
    b1.plevra_b=20;
    b1.plevra_c=30;
    cout<<b1.emvado();
    b1.show();
    .....
}
```

Καλείται η μέθοδος emvado() της **βασικής** κλάσης.

Καλείται η μέθοδος show() της **παράγωγης** κλάσης.

Όταν η παράγωγη κλάση διαθέτει μια συνάρτηση-μέλος με ίδιο όνομα και αποτύπωμα με μια συνάρτηση-μέλος της βασικής κλάσης, τότε η συνάρτηση-μέλος της παράγωγης κλάσης υπερβαίνει (override) την αντίστοιχη συνάρτηση της βασικής κλάσης.

Πρόσβαση σε υπερβατικές μεθόδους της βασικής κλάσης

```
class rectangle
{
public:
    float plevra_a;
    float plevra_b;
    float emvado()
    {
        return plevra_a * plevra_b;
    }
    void show()
    {
        cout<<plevra_a<<"x"
        <<plevra_b<<endl;
    }
};
```

public

```
class box:public rectangle
{
public:
    float plevra_c;
    float ogos()
    {
        return emvado()*plevra_c;
    }
    void show()
    {
        rectangle::show();
        cout<<"x"<<plevra_c<<endl;
    }
};
```

Καλείται η μέθοδος show() της **βασικής** κλάσης **rectangle**.

```
int main()
{
    box b1;
    b1.plevra_a=10;
    b1.plevra_b=20;
    b1.plevra_c=30;
    b1.show();
    b1.rectangle::show();
    .....
}
```

Καλείται η μέθοδος show() της **παράγωγης** κλάσης.

Καλείται η μέθοδος show()

Στην περίπτωση της υπέρβασης μεθόδων, μια υπερβατική μέθοδος της βασικής κλάσης μπορεί να κληθεί χρησιμοποιώντας ως πρόθεμα το όνομα της βασικής κλάσης ακολουθούμενο από τον τελεστή επίλυσης εμβέλειας ::.

Παράδειγμα

- Στον κώδικα που ακολουθεί, η παράγωγη κλάση **rec3D** διαθέτει και μια μέθοδο **embado()** η οποία επιστρέφει το συνολικό εμβαδό (και των 4 επιφανειών) των τρισδιάστατων αντικειμένων τύπου **rec3D**. Η μέθοδος **embado()** υποσκελίζει την ομώνυμη μέθοδο της κλάσης **rectangle**

```cpp
#include <iostream>
using namespace std;
class rectangle
{
protected:
    float plevra_a;
    float plevra_b;
public:
    float emvado(){return plevra_a * plevra_b;}
    void show() {cout<<plevra_a<<"x"<<plevra_b<<endl;}
    void set_ab(float a, float b){plevra_a=a; plevra_b=b;}
};
class rec3D:public rectangle
{
    float plevra_c;
public:
    float emvado();
    float ogos(){return plevra_a * plevra_b * plevra_c;}
    void set_c(float c){plevra_c=c;}
    float get_c(){return plevra_c;}
    void show() {cout<<plevra_a<<"x"<<plevra_b<<"x"<<plevra_c<<endl;}
};
float rec3D::emvado()
{
    return plevra_a*plevra_b*2+plevra_b*plevra_c*2+plevra_a*plevra_c*2;
}
```

```
class box:public rec3D
{
    float varos;
public:
    void set_varos(float v){varos=v;}
    float perimetros()
    {
        return 4*(plevra_a+plevra_b+get_c());
    }
};
int main()
{
    rec3D b;
    b.set_ab(23,3);
    b.set_c(4);
    b.show();
    cout<<"ΕΜΒΑΔΟ-3Δ="<<b.emvado()<<endl;
    cout<<"ΕΜΒΑΔΟ-2Δ="<<b.rectangle::emvado()<<endl;
    return 0;
}
```

Παράδειγμα

- Το παρακάτω πρόγραμμα δείχνει τη σειρά με την οποία εκτελούνται οι μέθοδοι δόμησης και αποδόμησης τόσο της παράγωγης όσο και της βασικής κλάσης

```cpp
#include <iostream>
using namespace std;
class rectangle
{
    float plevra_a;
    float plevra_b;
public:
    rectangle() {cout<<"rectangle δημιουργήθηκε"<<endl;}
    ~rectangle() {cout<<"rectangle καταστράφηκε"<<endl;}
    void set_ab(float a, float b);
};
void rectangle::set_ab(float a, float b)
{
    plevra_a=a;
    plevra_b=b;
}
class rec3D:public rectangle
{
    float plevra_c;
public:
    rec3D() {cout<<"rec3D δημιουργήθηκε"<<endl;}
    ~rec3D() {cout<<"rec3D καταστράφηκε"<<endl;}
    void set_c(float c){plevra_c=c;}
};
```

```
int main()
{
    {
        rec3D b;
        b.set_ab(23,3);
        b.set_c(4);
    }
    return 0;
}
```

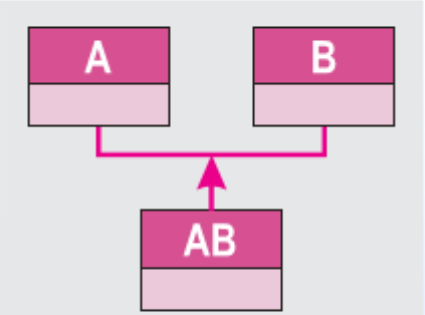
rectangle δημιουργήθηκε
rec3D δημιουργήθηκε
rec3D καταστράφηκε
rectangle καταστράφηκε

Πολλαπλή κληρονομικότητα

```
class A
{
    .....
}

class B
{
    .....
}

class AB : public A, public B
{
    .....
}
```



Η κλάση **AB** παράγεται από τις βασικές κλάσεις **A** και **B**.

Κάθε αντικείμενο της κλάσης **AB** περιέχει ένα υπο-αντικείμενο της κλάσης **A** και ένα της **B**.

Η πολλαπλή κληρονομικότητα αναφέρεται στην περίπτωση στην οποία μία κλάση παράγεται από δύο ή περισσότερες βασικές κλάσεις. Η πολλαπλή κληρονομικότητα δίνει τη δυνατότητα σε μια κλάση να κληρονομήσει τα χαρακτηριστικά και τις λειτουργίες από περισσότερες από μία κλάσεις.

Πολλαπλή κληρονομικότητα - παράδειγμα

```
class A
{
    int var_a;
public:
    void set_a(int x) {var_a=x;}
    void show_a() {cout<<"var_a="<<var_a<<endl;}
};
```

```
class B
{
    int var_b;
public:
    void set_b(int x) {var_b=x;}
    void show_b() {cout<<"var_b="<<var_b<<endl;}
};
```

public

public

```
class AB: public A, public B
{
public:
    void show_ab()
    {
     cout<<"object_ab"<<endl;
    }
};
```

Η κλάση AB κληρονομεί, με δημόσια πρόσβαση, όλες τις μεταβλητές και συναρτήσεις-μέλη των κλάσεων A και B.

```
int main()
{
    AB obj;          //δημιουργείται ένα αντικείμενο της κλάσης AB
    obj.set_a(23);   //καλείται η μέθοδος set_a() της κλάσης A
    obj.show_ab();   //καλείται η μέθοδος show_ab() της κλάσης AB
    obj.show_a();    //καλείται η μέθοδος show_a() της κλάσης A
    obj.set_b(11);   //καλείται η μέθοδος set_b() της κλάσης B
    obj.show_b();    //καλείται η μέθοδος show_b() της κλάσης B
    .....
}
```

Πολλαπλή κληρονομικότητα - παράδειγμα με ίδια μέλη

```
class A
{
    int var_a;
public:
    void set(int x) {var_a=x;}
    void show() {cout<<"var_a="<<var_a<<endl;}
};
```

```
class B
{
    int var_b;
public:
    void set(int x) {var_b=x;}
    void show() {cout<<"var_b="<<var_b<<endl;}
};
```

Οι δύο κλάσεις διαθέτουν μεθόδους με το ίδιο όνομα!

public

public

```
class AB: public A, public B
{
public:
   void show()
   {
     cout<<"object_ab"<<endl;
   }
};
```

Η κλάση AB διαθέτει τη μέθοδο show() η οποία υπερβαίνει τις μεθόδους show() των κλάσεων A και B.

```
int main()
{
    AB obj;          //δημιουργείται ένα αντικείμενο της κλάσης AB
    obj.A::set(23); //καλείται η μέθοδος set() της κλάσης A
    obj.show();      //καλείται η μέθοδος show() της κλάσης AB
    obj.A::show();  //καλείται η μέθοδος show() της κλάσης A
    obj.B::set(11); //καλείται η μέθοδος set() της κλάσης B
    obj.B::show();  //καλείται η μέθοδος show() της κλάσης B
    .....
}
```

Χρησιμοποιείται το **όνομα της κλάσης** και ο **τελεστής επίλυσης εμβέλειας**, για να προσδιορίσουμε τη μέθοδο που θα κληθεί.

Κληρονομικότητα (inheritance)

- Η κληρονομικότητα δίνει τη δυνατότητα να παράγουμε μια κλάση από μια ήδη υπάρχουσα κλάση.
- Η νέα κλάση που παράγεται λέγεται **παράγωγη** κλάση και η κλάση από την οποία προέκυψε **βασική** κλάση.
- Κάθε αντικείμενο μιας παράγωγης κλάσης εμπεριέχει ένα υποαντικείμενο της βασικής κλάσης.
- Μια παράγωγη κλάση μπορεί να αποτελέσει τη βασική κλάση για ένα επόμενο επίπεδο κληρονομικότητας.
- Το προσδιοριστικό πρόσβασης (public, private, protected) καθορίζει τον τρόπο με τον οποίο η παράγωγη κλάση κληρονομεί τα μέλη της βασικής.
- Το προσδιοριστικό πρόσβασης καθορίζει επίσης τον τρόπο με τον οποίο τα μέλη που κληρονομήθηκαν θα συμπεριφέρονται μέσα στην παράγωγη κλάση, αλλά και με ποιον τρόπο θα κληρονομηθούν σε ένα επόμενο επίπεδο κληρονομικότητας.
- Όταν η παράγωγη κλάση διαθέτει μια συνάρτηση-μέλος με ίδιο όνομα και αποτύπωμα (ίδιο τύπο, πλήθος και τύπο παραμέτρων) με μια συνάρτηση-μέλος της βασικής κλάσης, τότε η συνάρτηση-μέλος της παράγωγης κλάσης υπερβαίνει (override) την αντίστοιχη συνάρτηση της βασικής κλάσης.

Παράδειγμα

- Παρακάτω ορίζεται η κλάση **circle_based_shapes** για τη διαχείριση αντικειμένων τα οποία έχουν ως βάση έναν κύκλο. Δηλαδή οι κύκλοι, οι σφαίρες, οι κύλινδροι και οι κυκλικοί κώνοι. Θέλουμε να δημιουργήσουμε 2 νέες κλάσεις **sphere** και **cylinder** οι οποίες θα παράγονται από την παραπάνω κλάση και θα διαχειρίζονται 3D αντικείμενα σχήματος σφαίρας και κυλίνδρου αντίστοιχα. Κάθε νέα κλάση θα πρέπει να διαθέτει μια μέθοδο **embado()** η οποία θα υποσκελίζει την αντίστοιχη μέθοδος της **circle_based_shapes** και θα υπολογίζει την επιφάνεια του σχήματος. Επίσης και οι 2 κλάσεις θα διαθέτουν μια μέθοδο **ogos()** η οποία θα επιστρέφει τον όγκο του 3D σχήματος

```cpp
#include <iostream>
#include <cmath>
#define PI 3.141593

using namespace std;

class circle_based_shape
{
    float aktina;
public:
    float emvado(){return  pow(get_r(),2)*PI;}
    float perimetros(){return  2*get_r()*PI;}
    void set_r(float r){aktina=r;}
    float get_r() {return aktina;}
};

class sphere:public circle_based_shape
{
public:
    float emvado(){return  4*pow(get_r(),2)*PI;}
    float ogos(){return  4/3.0*pow(get_r(),3)*PI;}
};
```

```
class cylinder:public circle_based_shape
{
    float ypsos;
public:
    void set_ypsos(float y){ypsos=y;}
    float get_ypsos() {return ypsos;}
    float emvado(){return  2*PI*ypsos+2*PI*pow(get_r(),2);}
    float ogos(){return  ypsos*PI*pow(get_r(),2);}
};

int main()
{
    sphere ball;
    cylinder c1;
    ball.set_r(10);
    c1.set_r(5);
    c1.set_ypsos(10);
    cout<<"Εμβαδό επιφάνειας σφαίρας:"<<ball.emvado()<<" m2"<<endl;
    cout<<"Όγκος σφαίρας:"<<ball.ogos()<<" m3"<<endl;
    cout<<"Εμβαδό επιφάνειας κυλίνδρου:"<<c1.emvado()<<" m2"<<endl;
    cout<<"Όγκος κυλίνδρου:"<<c1.ogos()<<" m"<<endl;
    cout<<"Εμβαδό βάσης κυλίνδρου:"<<c1.circle_based_shape::emvado()<<" m2"<<endl;
    return 0;
}
```

Εμβαδό επιφάνειας σφαίρας: 1256.64m2
Όγκος σφαίρας: 4188.79m3
Εμβαδό επιφάνειας κυλίνδρου: 219.912m2
Όγκος κυλίνδρου: 785.389m3
Εμβαδό βάσης κυλίνδρου: 78.5398m2