Class Forwarding in C++

Class Forwarding

- Δύο κλάσεις, που η κάθε μία έχει κάποιο μέλος-δεδομένο που είναι δείκτης / αναφορά / αντικείμενο της άλλης κλάσης.
- Το πρόβλημα είναι με ποια σειρά θα γίνουν οι ορισμοί των κλάσεων.

Δείκτης / Αναφορά και στις δύο

```
class A
{
    ...
    B* b;
    ...
};
```

```
class B
{
    ...
    A* a;
    ...
};
```

Δείκτης / Αναφορά και στις δύο

- Ορίζουμε τις δύο κλάσεις με όποια σειρά θέλουμε και πριν τον πρώτο ορισμό βάζουμε μια δήλωση για την άλλη. Δηλαδή, έστω ότι πρώτα γράφουμε τον ορισμό της Α και μετά της Β, τότε πριν και από τους δύο ορισμούς, βάζουμε “class B;”

- Το παραπάνω λέγεται “class forwarding”

- Ενημερώνει τον μεταγλωττιστή ότι θα συναντήσει πιο κάτω κάποια κλάση με όνομα Β

- Το παραπάνω του αρκεί για να έχει δείκτες / αναφορές σε αντικείμενα τέτοιου τύπου

Δείκτης / Αναφορά στη μία μόνο

```
class A
{
    ...
    B* b;
    ...
};
```

```
class B
{
    ...
    A a;
    ...
};
```

Δείκτης / Αναφορά στη μία μόνο

- Σε αυτή την περίπτωση πρώτα πρέπει να οριστεί η κλάση που περιέχει τον δείκτη / αναφορά και μετά αυτή με το αντικείμενο
- Αυτό γιατί ο μεταγλωττιστής πρέπει να ξέρει τα πάντα για ένα τύπο, πριν φτιάξει αντικείμενα αυτού (δεν ισχύει το ίδιο για τους δείκτες)
- Για το παράδειγμα της προηγούμενης σελίδας, πριν και από τους δύο ορισμούς πρέπει να μπει το “class B;”

Αντικείμενα και στις δύο

```
class A
{
    ...
    B b;
    ...
};
```

```
class B
{
    ...
    A a;
    ...
};
```

Αντικείμενα και στις δύο

- Αυτή η περίπτωση δεν είναι επιτρεπτή στη C++
- Αντιστοιχεί σε μια έμεση μορφή αναδρομικών κλάσεων
- Ο μεταγλωττιστής θα έπρεπε να ξέρει πλήρως την κλάση B για να φτιάξει το μέλος στην A, αλλά για να ξέρει πλήρως την B θα έπρεπε να ξέρει πλήρως την A και ούτω καθεξής

Class Forwarding

- Σε όλες τις περιπτώσεις, πριν γίνει κλήση μιας συνάρτησης, ο μεταγλωττιστής πρέπει να ξέρει την δήλωση της
- Κάποιες συναρτήσεις-μέλη δεν μπορούν να οριστούν inline μαζί με την δήλωση της κλάσης, αλλά ξεχωριστά μετά

Class Forwarding (λάθος)

```
class A
{
    ...
    B* b;
    void foo(void)
    {
                b->lala();
    }
    ...
};
```

```
class B
{
    ...
    A* a;
    void lala(void) { }
    ...
};
```

Class Forwarding (σωστό)

```
class A
{
    ...
    B* b;
    void foo(void);
    ...
};
```

```
class B
{
    ...
    A* a;
    void lala(void) { }
    ...
};
```

```
void A::foo(void) { … }
```