Η ΓΛΩΣΣΑ C++

Μάθημα 11:

Εικονικές Μέθοδοι

Δημήτρης Ψούνης

www.psounis.gr

Περιεχόμενα Μαθήματος

Α. Θεωρία

1. Επαναορισμός Μεθόδων
   1. Υπέρβαση (overriding) Μεθόδων
   2. Δείκτες και Αναφορές σε Αντικείμενα
2. Εικονικές Μέθοδοι
   1. Ορισμός Μεθόδου Κλάσης ως εικονικής
   2. Παράδειγμα Ορισμού Εικονικής Μεθόδου
   3. Ισχύει μόνο για δείκτες/αναφορές!
   4. Διάχυση virtual στις υποκλάσεις
   5. Εικονικοί Καταστροφείς
3. Αφηρημένες Κλάσεις
   1. Αμιγώς Εικονικές Μέθοδοι
   2. Παράδειγμα

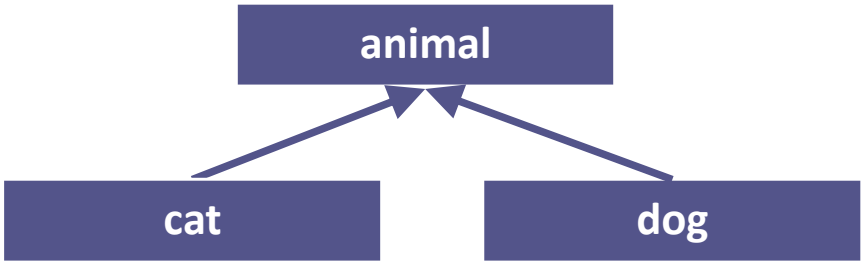
Β. Ασκήσεις

Α. Θεωρία

1. Επαναορισμός Μεθόδων

1. Υπέρβαση (overriding) Μεθόδων

- Μία παραγόμενη κλάση έχει το δικαίωμα να επαναορίσει μία μέθοδο της βασικής κλάσης.
- Ας επανέλθουμε σε μια απλή εκδοχή της οντολογίας των ζώων:



- κι έστω ότι η κλάση animal έχει μία μέθοδο make_sound().
  - Η γάτα και ο σκύλος έχουν δικαίωμα να επαναορίσουν τη μέθοδο ώστε να εξειδικεύουν τον ήχο τους.

- Αυτός είναι και ο πρώτος τρόπος για να επαναορίσουμε μια μέθοδο και αναφέρεται ως **υπέρβαση μεθόδου (method overriding)**

Α. Θεωρία

1. Επαναορισμός Μεθόδων

1. Υπέρβαση (overriding) Μεθόδων

- Η υλοποίηση των μεθόδων έχει ως εξής:

```
class animal {
 public:
   void make_sound();
   …
};

class dog: public animal {
 public:
   void make_sound();
    …
};

class cat: public animal {
 public:
   void make_sound();
    …
};
```

```
void animal::make_sound()
{
   cout<<"Animal sound"<<endl;
}
void dog::make_sound()
{
   cout<<"Arf arf"<<endl;
}
void cat::make_sound()
{
   cout<<"Meooooww"<<endl;
}
```

```
int main()
{
   dog piko(10,35,"Left Hill 154");
   cat kitty(6, 20);
   animal an(40, 80);
   piko.make_sound();
   kitty.make_sound();
   an.make_sound();
   return 0;
}
```

Ολοκληρωμένο το πρόγραμμα είναι το: «cpp11.method_overriding.cpp»

Α. Θεωρία

1. Επαναορισμός Μεθόδων

2. Δείκτες και Αναφορές σε Αντικείμενα

- Βλέπουμε τώρα μια ιδιαιτερότητα του ορισμού δεικτών (και αναφορών) σε αντικείμενα:

Ένας δείκτης σε αντικείμενο βασικής κλάσης:
- Μπορεί να δείχνει σε αντικείμενο παραγόμενης κλάσης

- Έτσι πέρα από την προφανή χρήση, π.χ.:

```
animal *bp = new animal();
bp->make_sound();
dog *dp = new dog();
dp->make_sound();
```

  - και έχουμε πρόσβαση στις μεθόδους που αντιστοιχούν στην κάθε κλάση

- Μπορούμε να ορίσουμε και δείκτη βασικής κλάσης που δείχνει σε αντικείμενο παραγόμενης κλάσης:

```
animal *bp = new dog();
bp->make_sound();
```

  - Ο δείκτης βασικής κλάσης που δείχνει σε αντικείμενο παραγόμενης κλάσης, έχει πρόσβαση μόνο στις μεθόδους της βασικής κλάσης.
- (Σημείωση: Δεν μπορούμε να κάνουμε το αντίθετο: Δείκτης παραγόμενης κλάσης να δείχνει σε αντικείμενο βασικής κλάσης)

Α. Θεωρία

1. Επαναορισμός Μεθόδων

2. Δείκτες και Αναφορές σε Αντικείμενα

- Με τη χρήση δεικτών βασικής κλάσης σε παραγόμενα αντικείμενα, μπορούμε να γράψουμε π.χ.:

```
int main()
{
   animal *bpd= new dog(10,35,"Left Hill 154");
   animal *bpc= new cat(6, 20);
   animal *bpa=new animal(40, 80);

   bpd->make_sound();
   bpc->make_sound();
   bpa->make_sound();

   delete bpd; delete bpc; delete bpa;
   return 0;
}
```

βλ: «cpp11.pointer_to_base_class1.cpp»

Σημαντικό:

- Βλέπουμε ότι με τον παραπάνω κώδικα έχουμε κατορθώσει να έχουμε πρόσβαση σε διαφορετικούς τύπους δεδομένων (cat, dog κ.λπ.) μέσω της βασικής κλάσης.

Α. Θεωρία

1. Επαναορισμός Μεθόδων

2. Δείκτες και Αναφορές σε Αντικείμενα

- Ωστόσο το πρόβλημα (που θέσαμε στην άσκηση 2 του προηγούμενου μαθήματος) δεν έχει ακόμη λυθεί, π.χ. ο ακόλουθος κώδικας:

```
int main()
{
   animal *pin[3];
   pin[0] = new dog(10,35,"Left Hill 154");
   pin[1] = new cat(6, 20);
   pin[2] = new animal(40, 80);
   for (int i=0; i<3; i++)
      pin[i]->make_sound();
   for (int i=0; i<3; i++)
      delete pin[i];
   return 0;
}
```

βλ: «cpp11.pointer_to_base_class2.cpp»

- Θα καλέσει για όλα τα αντικείμενα την make_sound της βασικής κλάσης
  - (ενώ θα θέλαμε να καλέσει την make_sound της παραγόμενης κλάσης)
- Η λύση που δόθηκε είναι οι εικονικές μέθοδοι!

Α. Θεωρία

2. Εικονικές Μέθοδοι

1. Ορισμός Μεθόδου Κλάσης ως εικονικής

- Ορίζουμε μία μέθοδο ως εικονική, θέτοντας τη λέξη – κλειδί virtual μπροστά της κατά τον ορισμό της στη βασική κλάση (μόνο στο πρωτότυπο της μεθόδου).

Μία **virtual μέθοδος** μίας βασικής κλάσης δίνει τη δυνατότητα:
- Να επαναορίσουμε τη μέθοδο στην παραγόμενη κλάση
- Και ορίζοντας έναν δείκτη βασικής κλάσης
  - και θέτοντας τον να δείχνει σε ένα αντικείμενο παραγόμενης κλάσης
  - να ενεργοποιείται η μέθοδος της παραγόμενης κλάσης

- Συνεπώς για τον επαναορισμό μιας μεθόδου κάνουμε τον εξής διαχωρισμό:
- Αν έχουμε έναν δείκτη βασικής κλάσης που δείχνει σε αντικείμενο παραγόμενης κλάσης:
  - Αν κληθεί η μέθοδος, τότε:
    - Αν αυτή δεν είναι virtual, καλείται η μέθοδος της βασικής κλάσης
    - Αν αυτή είναι virtual, καλείται η μέθοδος της παραγόμενης κλάσης

Α. Θεωρία

2. Εικονικές Μέθοδοι

2. Παράδειγμα Ορισμού Εικονικής Μεθόδου

- Συνεπώς, απλά θέτοντας ως virtual την επίμαχη μέθοδο:

```
class animal {
  public:
    ....
    virtual void make_sound();
    ...
};
```

```
int main()
{
    animal *pin[3];
    pin[0] = new dog(10,35,"Left Hill 154");
    pin[1] = new cat(6, 20);
    pin[2] = new animal(40, 80);

    for (int i=0; i<3; i++)
        pin[i]->make_sound();

   for (int i=0; i<3; i++)
        delete pin[i];
    return 0;
}
```

βλ: «cpp11.virtual_method.cpp»

- Θα καλέσει για κάθε αντικείμενο τη make_sound όπως έχει οριστεί στην παραγόμενη κλάση!

Α. Θεωρία

2. Εικονικές Μέθοδοι

3. Ισχύει μόνο για δείκτες/αναφορές!

- Προσοχή, ότι η παραπάνω τεχνική ισχύει μόνο όταν έχουμε πρόσβαση στα αντικείμενα μέσω δεικτών ή αναφορών. Π.χ. ο ακόλουθος κώδικας δεν έχει τα επιθυμητά αποτελέσματα:

```
int main()
{
   animal pin[3] = {
      dog(10,35,"Left Hill 154"),
      cat(6, 20),
      animal(40, 80)
   };

   for (int i=0; i<3; i++)
      pin[i].make_sound();

   return 0;
}
```

βλ: «cpp11.virtual_method_object.cpp»

Α. Θεωρία

2. Εικονικές Μέθοδοι

3. Ισχύει μόνο για δείκτες/αναφορές!

- Ενώ ο ακόλουθος κώδικας (πρόσβαση μέσω αναφοράς) όντως έχει τα επιθυμητά αποτελέσματα:

```
void f(animal &ob)
{
   ob.make_sound();
}

int main()
{
   dog d(10,35,"Left Hill 154");

   f(d);

   return 0;
}
```

βλ: «cpp11.virtual_method_reference.cpp»

Α. Θεωρία

2. Εικονικές Μέθοδοι

3. Ισχύει μόνο για δείκτες/αναφορές!

- Συνοψίζοντας:
  - Έστω ότι η βασική κλάση:
    - Έχει μία μέθοδο over() η οποία έχει οριστεί χωρίς virtual
    - Έχει μία μέδος virt() η οποία επαναορίζεται με virtual
  - Και η παραγόμενη κλάση επαναορίζει τις over() και virt()
  - Τότε ανάλογα με το πως έχουμε πρόσβαση σε ένα αντικείμενο της παραγόμενης κλάσης καλείται είτε η μέθοδος της βασικής είτε της παραγόμενης κλάσης και συγκεκριμένα:

Πρόσβαση	Κώδικας	X=over()	X=virt()
Αντικείμενο	Base ob = der(); ob.X();	**Base**	**Base**
Μέσω Δείκτη	Base *p = new der(); p->X();	**Base**	**Derived**
Μέσω Αναφοράς	Derived ob; Base &r =ob; r.X();	**Base**	**Derived**

Α. Θεωρία

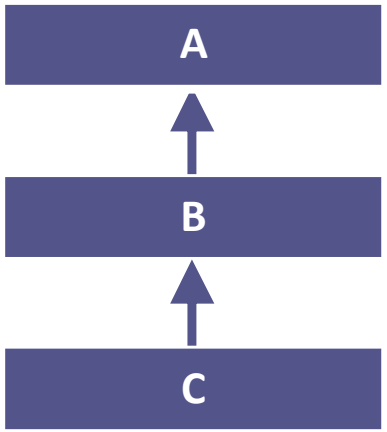
2. Εικονικές Μέθοδοι

4. Διάχυση virtual στις υποκλάσεις

- Σημαντικό!

Αν μία μέθοδος οριστεί virtual τότε παραμένει virtual σε κάθε παραγόμενη κλάση
- χωρίς να χρειάζεται να γράφουμε ρητά ότι παραμένει virtual στην παραγόμενη κλάση.

- π.χ. στην ακόλουθη ιεραρχία κλάσεων, εννοείται ότι σε όλες τις παραγόμενες κλάσεις θα παραμένει virtual, ακόμη και αν η διατύπωση είναι αυτή που φαίνεται δεξιά:




```
class A {
  public:
    virtual void f();
  ....
};
```

```
class B: public A {
  public:
    void f();
  ....
};
```

```
class C: public B {
  public:
    void f();
  ....
};
```

Σημείωση:
- Καλό θα είναι (για λόγους αναγνωσιμότητας του προγράμματος) να αναφέρουμε ότι η συνάρτηση είναι virtual και στις υποκλάσεις, εφόσον αυτές επαναορίζονται σε περαιτέρω υποκλάσεις.

Α. Θεωρία

2. Εικονικές Μέθοδοι

5. Εικονικοί Καταστροφείς

- Ας υποθέσουμε ότι έχουμε μία βασική κλάση Β και μία παραγόμενη αυτής D
- Για τους καταστροφείς τους:
  - Είδαμε στο προηγούμενο μάθημα, ότι όταν έχουμε πρόσβαση μέσω αντικειμένου, η εκτέλεση των destructors γίνεται όπως περιμέναμε:
    - πρώτα εκτελείται ο destructor της παραγόμενης και μετά της βασικής κλάσης
  - Θα δούμε τώρα τι γίνεται όταν έχουμε πρόσβαση μέσω δείκτη.

- Περίπτωση Α': Μέσω δείκτη στην παραγόμενη κλάση:

```
int main()
{
   dog *dp= new dog(10,35,"Left Hill 154");

   delete dp;

   return 0;
}
```

βλ: «cpp11.destructor_derived_pointer.cpp»

  - Η εκτέλεση των destructor γίνεται κανονικά.

Α. Θεωρία

2. Εικονικές Μέθοδοι

5. Εικονικοί Καταστροφείς

- Περίπτωση Β': Μέσω δείκτη στη βασική κλάση:

```
int main()
{
   animal *bp = new dog(10,35,"Left Hill 154");

   delete bp;

   return 0;
}
```

βλ: «cpp11.destructor_base_pointer.cpp»

- Παρατηρούμε ότι εκτελείται μόνο ο destructor της βασικής κλάσης
  - (Γίνεται ταίριασμα του τύπου δεδομένων του δείκτη με τον destructor που εκτελείται).

- Το πρόβλημα αυτό μπορεί να ξεπεραστεί, δηλώνοντας τον destructor ως virtual

Α. Θεωρία

2. Εικονικές Μέθοδοι

5. Εικονικοί Καταστροφείς

- Η δήλωση του constructor ως virtual πρέπει να γίνει στην βασική κλάση:

```
class animal {
  public:
    …
    virtual ~animal();
    …
};
```

```
int main()
{
    animal *bp = new dog(10,35,"Left Hill 154");

    delete bp;

    return 0;
}
```

βλ: «cpp11.virtual_destructor_base_pointer.cpp»

- Το πρόγραμμα πλέον έχει την επιθυμητή συμπεριφορά.

Συνεπώς για να δουλεύουν σωστά οι destructors με την κληρονομικότητα:

- Θα πρέπει ΠΑΝΤΑ να δηλώνουμε ως virtual τον destructor κλάσης που κληρονομείται από άλλες κλάσεις.

Α. Θεωρία

3. Αφηρημένες Κλάσεις

1. Αμιγώς Εικονικές Μέθοδοι

- Πολλές φορές οι υπερκλάσεις δεν έχουν πραγματικό νόημα.
- Για παράδειγμα, η έννοια «ζώο» ναι μεν είναι υπαρκτή
  - Αλλά δεν υπάρχει κάτι στον πραγματικό κόσμο, το οποίο να είναι αποκλειστικά και μόνο ζώο και να μην προσδιορίζεται από κάποια παραγόμενη κλάση (π.χ. σκύλος, γάτα κ.λπ.)
- Επίσης αυτό οδηγεί σε κάποιους παραλογισμούς.
  - π.χ. η μέθοδος make_sound() δεν είχε ιδιαίτερο νόημα.
    - και βάλαμε να βγάζει το μη ρεαλιστικό «animal sound»

- Για το λόγο αυτό ορίζονται οι αμιγώς εικονικές μέθοδοι (pure virtual):

Μία μέθοδος ορίζεται **αμιγώς εικονική**, θέτοντας την = 0 κατά τη δήλωσή της
- Η μέθοδος δεν απαιτεί πια ορισμό (δεν κατασκευάζουμε σώμα της)
- και πλέον δεν μπορούμε να κατασκευάσουμε αντικείμενα της κλάσης
  - Η κλάση γίνεται μία **«αφηρημένη» κλάση (abstract class)**

Α. Θεωρία

3. Αφηρημένες Κλάσεις

2. Παράδειγμα

- Δηλώνουμε την μέθοδο make_sound() ως εικονική και η κλάση γίνεται αφηρημένη:

```
class animal {
  public:
    …
    virtual void make_sound() = 0;
    …
};
```

```
int main()
{
    animal *bp = new dog(10,35,"Left Hill 154");

    delete bp;

    return 0;
}
```

βλ: «cpp11.virtual_destructor_base_pointer.cpp»

- Δεν μπορουμε να δηλώσουμε αντικείμενα, αλλά μπορούμε να δηλώσουμε δείκτη (ή αναφορά) βασικής κλάσης ώστε να δείχνει σε αντικείμενα παραγόμενων κλάσεων

Μία αφηρημένη κλάση (ή αμιγώς εικονική κλάση):

- Αν οι εικονικές της μέθοδοι δεν επαναοριστούν στην παραγόμενη κλάση, τότε και αυτή θα είναι αφηρημένη κλάση.
- Χρησιμεύουν και για να ορίσουμε διεπαφές (interfaces – προδιαγραφές που πρέπει να ακολουθούν όλες οι παραγόμενες κλάσεις)

Β. Ασκήσεις

Άσκηση 1.1: Magic The Gathering (συνέχεια…)

Επεκτείνουμε το πρόγραμμα (άσκηση 2.4, μάθημα 10):

- Ορίστε μία κλάση pool
  - Σε αυτήν θα πρέπει να αποθηκεύονται το πολύ 300 κάρτες
  - Θα χρησιμεύει για να αποθηκεύσουμε τις κάρτες οι οποίες υπάρχουν στο παιχνίδι.
  - Ο κατασκευαστής να αποθηκεύει σε κατάλληλη δομή τις κάρτες του προηγούμενου μαθήματος, καθώς και τις ακόλουθες κάρτες:

Β. Ασκήσεις

Άσκηση 1.1: Magic The Gathering (συνέχεια…)

- Ορίστε επίσης:
  - Μία μέθοδο: pick_random_card() η οποία θα επιστρέφει μία από τις κάρτες που έχουν αποθηκευτεί στην κλάση pool.

Β. Ασκήσεις

Άσκηση 1.2: Magic The Gathering (συνέχεια…)

- Ορίστε μία κλάση hand
  - Θα αποθηκεύει το πολύ 10 κάρτες
  - Έχει μία μέθοδο pick_card(pool &p, int i) η οποία θα επιλέγει τυχαία μία κάρτα και θα την τοποθετεί στο χέρι του παίκτη, στη θέση i.
  - Έχει μία μέθοδο fill_hand(pool &p) η οποία θα γεμίζει το χέρι του παίκτη με 7 τυχαίες κάρτες.
  - Ο κατασκευαστής να γεμίζει το χέρι του παίκτη
  - Έχει υπερφόρτωση του τελεστή εκτύπωσης.
- Αναδείξτε τη συμπεριφορά της κλάσης μέσω κατάλληλης συνάρτησης main.

Β. Ασκήσεις

Άσκηση 2.1: Κλάση Μάγος

Επεκτείνουμε την άσκηση 3 του μαθήματος 6

- Τροποποιήστε την κλάση «Μάγος»
- ώστε να έχει μία μέθοδο attack() η οποία σε κάθε γύρο
  - θα κάνει damage από 20 έως 30 (με το ραβδί του)

- Τροποποιήστε τη main ώστε σε κάθε γύρο:
  - Ο μάγος πρώτα θα επιτίθεται με το ραβδί του
  - Έπειτα θα επιλέγει τυχαία lightning ή fireball

Β. Ασκήσεις

Άσκηση 2.2: Μάγος εναντίον Yeti

Επεκτείνουμε την άσκηση 3 του μαθήματος 6

- Η κλάση humanoid κληρονομείται από την κλάση Yeti
- Ένα Yeti
  - επιτίθεται κάνοντας ζημιά από 3 έως 10
  - έχει υγεία ίση με 150

Αλλάξτε την main ώστε να προσομοιώνει τη μάχη του μάγου με 2 Yeti.

- Τα Yeti να είναι αποθηκευμένα σε έναν πίνακα 2 θέσεων
  - και να αποθηκεύονται μέσω δεικτών
- Ο Μάγος επιλέγει ένα από τα 2 yeti και του επιτίθεται μέχρι να το σκοτώσει
  - έπειτα επιλέγει το 2ο yeti και επιτίθεται σε αυτό

Β. Ασκήσεις

Άσκηση 2.3: Μάγος εναντίον Goblin

Συνεχίζουμε την επέκταση του προγράμματος:

- Η κλάση humanoid κληρονομείται από την κλάση Goblin
- Ένα Goblin
  - επιτίθεται κάνοντας ζημιά από 2 έως 5
  - έχει υγεία ίση με 80

Αλλάξτε την main ώστε να προσομοιώνει τη μάχη του μάγου με 2 Yeti και 2 Goblin.