Πανεπιστήμιο Αιγαίου

Μεθοδολογίες και Γλώσσες Προγραμματισμού Ι

Ειδικά Θέματα

Εργίνα Καβαλλιεράτου (kavallieratou@aegean.gr)

Μόνιμη Επίκουρος Καθηγήτρια

Τμήμα Μηχανικών Πληροφοριακών & Επικοινωνιακών Συστημάτων

Άδειες Χρήσης

- Το παρόν εκπαιδευτικό υλικό υπόκειται σε άδειες χρήσης Creative Commons.
- Για εκπαιδευτικό υλικό, όπως εικόνες, που υπόκειται σε άλλου τύπου άδειας χρήσης, η άδεια χρήσης αναφέρεται ρητώς.

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Αιγαίου**» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Σήμερα!

- ✓ Static Member
- ✓ Πρόσβαση χωρίς αντικείμενο
- ✓ private static member
- ✓ Static Member Functions
- ✓ Πρόσβαση συνάρτησης χωρίς αντικείμενο
- ✓ Ισότητα αντικειμένων
- ✓ Friend κλάσεις και συναρτήσεις

Static members

- ✓Τόσο οι μεταβλητές-μέλη όσο και οι συναρτήσεις μιας κλάσης μπορούν να δηλωθούν ως στατικές με το προσδιοριστικό static
- ✓Τα στατικά μέλη διαφοροποιούνται από τα υπόλοιπα όσον αφορά στον τρόπο κατανομής μνήμης που τους διατίθεται και στον τρόπο χρήσης τους

Static Member/1

```
#include <iostream.h>
class Cat
{
 public:
   Cat(int age):itsAge(age){HowManyCats++; }
   virtual ~Cat() { HowManyCats--; }
   virtual int GetAge() { return itsAge; }
   virtual void SetAge(int age) { itsAge = age; }
   static int HowManyCats;
private:
   int itsAge;
};
int Cat::HowManyCats = 0;
```

Static Member/2

```
int main()    {
      const int MaxCats = 5; int i;
      Cat *CatHouse[MaxCats];
      for (i = 0; i<MaxCats; i++)
        CatHouse[i] = new Cat(i);
      for (i = 0; i<MaxCats; i++)       {
        cout << "There are ";
        cout << Cat::HowManyCats;
        cout << " cats left!\n";
        cout << "Deleting the one which is ";
        cout << CatHouse[i]->GetAge();
        cout << " years old\n";
        delete CatHouse[i];
        CatHouse[i] = 0;    }
    return 0; }
```

Output

```
There are 5 cats left!
Deleting the one which is 0 years old
There are 4 cats left!
Deleting the one which is 1 years old
There are 3 cats left!
Deleting the one which is 2 years old
There are 2 cats left!
Deleting the one which is 3 years old
There are 1 cats left!
Deleting the one which is 4 years old
```

Static members

- ✓Οι στατικές μεταβλητές μιας κλάσης δεσμεύουν μνήμη μόνο μία φορά και τις «μοιράζονται» όλα τα αντικείμενα αυτής της κλάσης
- ✓Η δήλωση της στατικής μεταβλητής γίνεται στη δήλωση της κλάσης
- ✓Η αρχικοποίηση της τιμής της στατικής μεταβλητής γίνεται εκτός κλάσης με χρήση του τελεστή εμβέλειας

Πρόσβαση χωρίς αντικείμενο/1

```
#include <iostream.h>
class Cat
{
public:
   Cat(int age):itsAge(age){HowManyCats++; }
   virtual ~Cat() { HowManyCats--; }
   virtual int GetAge() { return itsAge; }
   virtual void SetAge(int age) { itsAge = age; }
   static int HowManyCats;
private:
   int itsAge;
};
int Cat::HowManyCats = 0;
```

Πρόσβαση χωρίς αντικείμενο/2

```
void TelepathicFunction();
   int main()   {
      const int MaxCats = 5; int i;
      Cat *CatHouse[MaxCats];
      for (i = 0; i<MaxCats; i++)     {
         CatHouse[i] = new Cat(i);
         TelepathicFunction();
      }
      for ( i = 0; i<MaxCats; i++)     {
         delete CatHouse[i];
         TelepathicFunction();
      }
      return 0;
   }
```

Πρόσβαση χωρίς αντικείμενο/3

```
void TelepathicFunction()
   {
      cout << "There are ";
      cout << Cat::HowManyCats << " cats alive!\n";
   }
```

Output

There are 1 cats alive!
There are 2 cats alive!
There are 3 cats alive!
There are 4 cats alive!
There are 5 cats alive!
There are 4 cats alive!
There are 3 cats alive!
There are 2 cats alive!
There are 1 cats alive!
There are 0 cats alive!

private static member/1

```
#include <iostream.h>
 class Cat
 {
 public:
    Cat(int age):itsAge(age){HowManyCats++; }
    virtual ~Cat() { HowManyCats--; }
    virtual int GetAge() { return itsAge; }
    virtual void SetAge(int age) { itsAge = age; }
    virtual int GetHowMany() { return HowManyCats; }
 private:
    int itsAge;
    static int HowManyCats;
 };
 int Cat::HowManyCats = 0;
```

private static member/2

```
int main()    {
   const int MaxCats = 5; int i;
   Cat *CatHouse[MaxCats];
   for (i = 0; i<MaxCats; i++)
      CatHouse[i] = new Cat(i);
   for (i = 0; i<MaxCats; i++)      {
      cout << "There are ";
      cout << CatHouse[i]->GetHowMany();
      cout << " cats left!\n";
      cout << "Deleting the one which is ";
      cout << CatHouse[i]->GetAge()+2;
      cout << " years old\n";
      delete CatHouse[i];
      CatHouse[i] = 0;      }
 return 0; }
```

Output

There are 5 cats left!
Deleting the one which is 2 years old
There are 4 cats left!
Deleting the one which is 3 years old
There are 3 cats left!
Deleting the one which is 4 years old
There are 2 cats left!
Deleting the one which is 5 years old
There are 1 cats left!
Deleting the one which is 6 years old

Static Member Functions/1

```
#include <iostream.h>
class Cat
{
public:
   Cat(int age):itsAge(age){HowManyCats++; }
   virtual ~Cat() { HowManyCats--; }
   virtual int GetAge() { return itsAge; }
   virtual void SetAge(int age) { itsAge = age; }
   static int GetHowMany() { return HowManyCats; }
private:
   int itsAge;
   static int HowManyCats;
};
```

Static Member Functions/2

```
int Cat::HowManyCats = 0;
void TelepathicFunction();
int main()    {
   const int MaxCats = 5;
   Cat *CatHouse[MaxCats]; int i;
   for (i = 0; i<MaxCats; i++)       {
      CatHouse[i] = new Cat(i);
      TelepathicFunction();     }
   for ( i = 0; i<MaxCats; i++)     {
      delete CatHouse[i];
      TelepathicFunction();     }
   return 0;   }
void TelepathicFunction()   {
   cout << "There are " << Cat::GetHowMany() << " cats alive!\n"; }
```

Output

There are 1 cats alive!
There are 2 cats alive!
There are 3 cats alive!
There are 4 cats alive!
There are 5 cats alive!
There are 4 cats alive!
There are 3 cats alive!
There are 2 cats alive!
There are 1 cats alive!
There are 0 cats alive!

Πρόσβαση συνάρτησης χωρίς αντικείμενο/1

```
class Cat
{
public:
static int GetHowMany() { return HowManyCats; }
private:
static int HowManyCats;
};
int Cat::HowManyCats = 0;
int main()
{
int howMany;
Cat theCat;
howMany = theCat.GetHowMany();
howMany = Cat::GetHowMany();     }
```

Χαρακτηριστικά static συναρτήσεων

- ✓ Μια στατική συνάρτηση μπορεί να έχει **πρόσβαση μόνο σε στατικά μέλη** της κλάσης. Μια μη-στατική συνάρτηση μπορεί να έχει πρόσβαση τόσο στα στατικά όσο και στα μη στατικά μέλη της κλάσης.
- ✓ Μια στατική συνάρτηση μπορεί να κληθεί ακόμη και αν η κλάση δεν έχει κανένα αντικείμενο.
- ✓ Σε μια στατική συνάρτηση δεν μεταβιβάζεται ο δείκτης **this**.

Ισότητα Αντικειμένων

Πότε δύο αντικείμενα είναι ίσα;

- ✓ Όταν έχουν τις ίδιες τιμές

```
int m, n;
if (m == n) . . .
```

- ✓ Όταν δείχνουν στις ίδιες τιμές
  - int* p;
  - int* q;
  - // . . .
  - if (*p = = *q) . . .

Ισότητα Αντικειμένων

- ✓ Όταν είναι οι ίδιοι δείκτες

```
int* p;
int* q;
// . . .
if (p = = q) . . .
```

- ✓ Όταν είναι οι ίδιες αναφορές; (σύγκριση διευθύνσεων μνήμης)

```
if (&r == &s)
```

Ισότητα Αντικειμένων

```
bool Point::is_equal(Point& b) {
  return x == b.x && y == b.y;
}
bool Rectangle::is_equal(Rectangle& b) {
  return left_top().is_equal(b.left_top())
    && right_bottom().is_equal(b.right_bottom());
}
```

Θα πρέπει πάντα η ισότητα μεταξύ αντικειμένων να ελέγχεται από συνάρτηση που γράφει ο προγραμματιστής.

Ισότητα Αντικειμένων /1

```
#include <iostream.h>
class CAT
{
    public:
        CAT();
        int GetAge() const { return *itsAge; }
        int GetWeight() const { return *itsWeight; }
        void SetAge(int age) { *itsAge = age; }
        CAT operator=(const CAT &);
    private:
        int *itsAge;
        int *itsWeight;
};
```

Ισότητα Αντικειμένων/2

```
CAT::CAT()    {
 itsAge = new int;
itsWeight = new int;
*itsAge = 5;
*itsWeight = 9;    }
CAT CAT::operator=(const CAT & rhs) {
 if (this == &rhs)
   return *this;
delete itsAge;
delete itsWeight;
itsAge = new int;
itsWeight = new int;
*itsAge = rhs.GetAge();
*itsWeight = rhs.GetWeight();
return *this; }
```

Ισότητα Αντικειμένων /3

```
int main()
{
    CAT frisky;
    cout << "frisky's age: " << frisky.GetAge() << endl;
    cout << "Setting frisky to 6...\n";
    frisky.SetAge(6);
    CAT whiskers;
    cout << "whiskers' age: " << whiskers.GetAge() << endl;
    cout << "copying frisky to whiskers...\n";
    whiskers = frisky;
    cout << "whiskers' age: " << whiskers.GetAge() << endl;
return 0;
}
```

```
frisky's age: 5
Setting frisky to 6...
whiskers' age: 5
copying frisky to whiskers...
whiskers' age: 6
```

Friend classes

- ✓Μία κλάση μπορεί να δηλώσει άλλες κλάσεις ως "friend classes".
- ✓Μια κλάση μπορεί να δηλώσει εξωτερικές συναρτήσεις ως "friends".
- ✓Οι friends έχουν πρόσβαση σε private μέλη

Δήλωση friend κλάσης

```
class X {
 friend class ExternalClass;
 ...
};
```

- ✓ Όλες οι συναρτήσεις μέλη της ExternalClass έχουν πρόσβαση στα μέλη της X

Δήλωση friend συνάρτησης

```
class X {
...
friend ret_type func_name (arguments);
...
};
ret_type func_name (arguments) {…}
```

- ✓ Ο ορισμός της συνάρτησης δεν περιέχει τη λέξη **friend**

Παράδειγμα friend συνάρτησης

```
class rectangle
{
private:
    float plevra_x;
    float plevra_y;
public:
    rectangle(float plevra_x, float plevra_y)
          {this>plevra_x=plevra_x; this->plevra_y=plevra_y;}
    rectangle() {this->plevra_x=0; this->plevra_y=0;}
    float emvado() {return plevra_x*plevra_y;}
    friend void to_zero(rectangle &);
};
```

Παράδειγμα friend συνάρτησης

```
void to_zero(rectangle &rec)
{
rec.plevra_x=0;
rec.plevra_y=0;
}

int main() {
rectangle rect(20, 10);
cout << "Εμβαδόν : " << rect.emvado() << endl;
to_zero(rect);
cout << "Εμβαδόν : " << rect.emvado() << endl;
}
```