Πανεπιστήμιο Αιγαίου

Μεθοδολογίες και Γλώσσες Προγραμματισμού Ι

Classes

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

Σημερινό Μάθημα

- ✓ Constructor και destructor
- ✓ Συναρτήσεις-μέλη const
- ✓ Inline συναρτήσεις
- ✓ Δηλώσεις κλάσεων
- ✓ Σύνθετες κλάσεις

Constructor και destructor

- ✓ Αν δεν δηλώσουμε constructor ή/και destructor ο compiler χρησιμοποιεί προκαθορισμένους.
- ✓ Οι προκαθορισμένοι constructor και destructor δεν κάνουν τίποτα και δεν επιστρέφουν τίποτα, ισοδυναμούν με:

cat() & ~cat()

- ✓ Όταν δηλώσουμε ένα αντικείμενο κλάσης καλούμε τον constructor
- ✓ Αν ο constructor δεν παίρνει παραμέτρους, μπορούμε να γράψουμε:

Cat Frisky;

- ✓ Συνίσταται όταν δηλώνουμε constructor να δηλώνουμε και destructor

Πρόγραμμα/1

```
#include <iostream.h>
class Cat {
    public:
        Cat(int initialAge);          // constructor
        ~Cat();                       // destructor
        int GetAge();
        void SetAge (int age);
        void Meow();
    private:
        int itsAge;
};
```

Πρόγραμμα/2

```
Cat::Cat(int initialAge) {
itsAge = initialAge;
}
Cat::~Cat(){
}
int Cat::GetAge()  {
    return itsAge;
}
void Cat::SetAge(int age) {
    itsAge = age;
}
void Cat::Meow() {
    cout << "Meow.\n";
}
```

Πρόγραμμα/3

```
int main() {
    Cat Frisky(5);
    Frisky.Meow();
    cout << "Frisky is a cat who is " ;
    cout << Frisky.GetAge() << " years old.\n";
    Frisky.Meow();
    Frisky.SetAge(7);
    cout << "Now Frisky is " ;
    cout << Frisky.GetAge() << " years old.\n";
    return 0;
}
```

Συναρτήσεις-μέλη const

- ✓ Με μία συνάρτηση const μέσα στην κλάση, πληροφορούμε τον compiler, ότι δεν αλλάζει τα δεδομένα της κλάσης

```
void SomeFunction() const;
int GetAge() const;
```

- ✓ Είναι καλή προγραμματιστική τεχνική να δηλώνουμε το μέγιστο δυνατό αριθμό const συναρτήσεων για προστασία

Λάθος Παράδειγμα/1

```
#include <iostream.h>
class Cat
{
  public:
     Cat(int initialAge);
     ~Cat();
     int GetAge() const;
     void SetAge (int age);
     void Meow();
  private:
    int itsAge;
};
```

Λάθος Παράδειγμα/2

```
Cat::Cat(int initialAge)   {
     itsAge = initialAge;
      cout << "Cat Constructor\n“;   }
Cat::~Cat()    {
      cout << "Cat Destructor\n“;   }
int Cat::GetAge() const  {
    return (itsAge++);  }
void Cat::SetAge(int age)  {
     itsAge = age; }
void Cat::Meow() {
     cout << "Meow.\n"; }
```

Inline συναρτήσεις

- ✓ Αν μας ενδιαφέρει η ταχύτητα εκτέλεσης και η συνάρτηση που θέλουμε να γράψουμε είναι απλή τότε μπορούμε να τη δηλώσουμε ως inline π.χ.

```
inline int square(int x) {return x*x;}
```

Inline συναρτήσεις

- ✓ Μπορούμε να γράψουμε εκτός κλάσης

```
inline int Cat::GetWeight(){
return itsWeight;
}
```

- ✓ Ή αν τη δηλώσουμε μέσα στην κλάση γίνεται αυτόματα inline

```
class Cat{
public:
int GetWeight() { return itsWeight; }    // inline
void SetWeight(int aWeight);
};
```

Δηλώσεις κλάσεων

- ✓ Συνηθίζεται να δηλώνουμε τις κλάσεις, καθώς και όλες τις δηλώσεις ενός προγράμματος σε ξεχωριστό αρχείο.
- ✓ Αυτό το αρχείο ονομάζεται header file, έχει το ίδιο όνομα με το αρχείο του προγράμματος και επέκταση .h ή .hpp
- ✓ Αν δηλώσουμε μία κλάση σε ξεχωριστό αρχείο, το αρχείο παίρνει το όνομα της κλάσης.
- ✓ Στο κυρίως αρχείο πρέπει να συμπεριλάβουμε τα header files με τις βιβλιοθήκες.

CAT.H

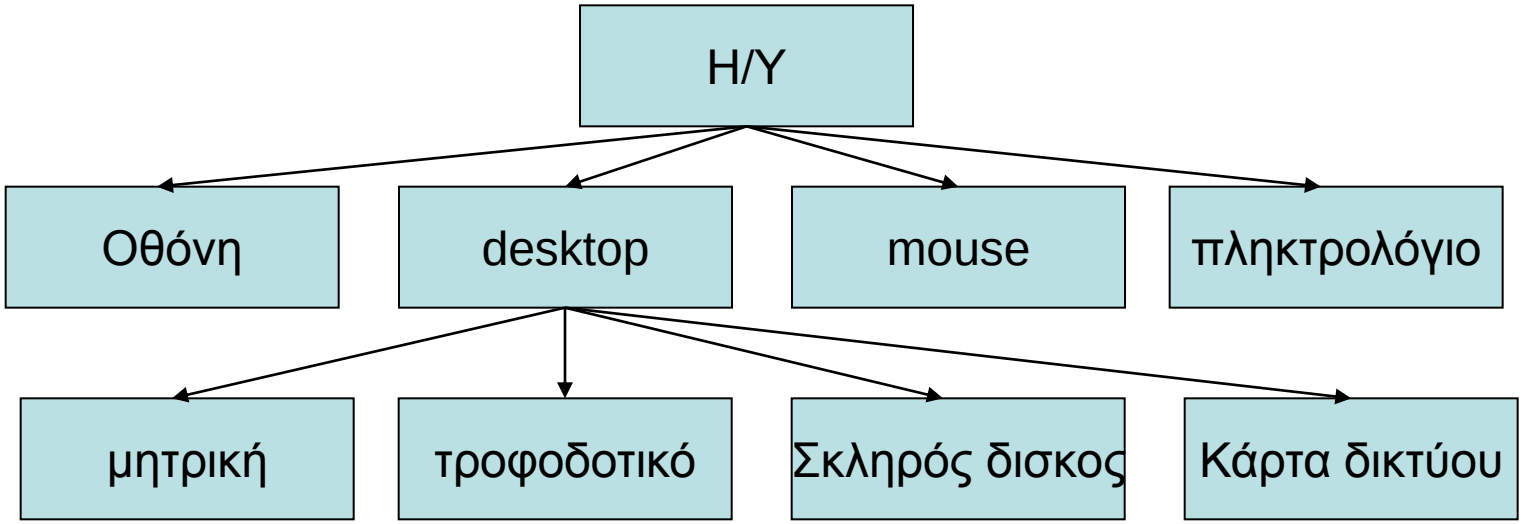
```
#include <iostream.h>
 class Cat
 {
 public:
    Cat (int initialAge);
   ~Cat();
    int GetAge() { return itsAge;}              // inline!
    void SetAge (int age) { itsAge = age;}    // inline!
    void Meow() { cout << "Meow.\n";}      // inline!
 private:
     int itsAge;
 };
```

CAT.CPP

```
#include "cat.h"
  Cat::Cat(int initialAge) {
      itsAge = initialAge; }
  Cat::~Cat() {
  }
int main() {
     Cat Frisky(5);
     Frisky.Meow();
     cout << "Frisky is a cat who is " ;
     cout << Frisky.GetAge() << " years old.\n";
     Frisky.Meow();
     Frisky.SetAge(7);
     cout << "Now Frisky is " ;
     cout << Frisky.GetAge() << " years old.\n";
     return 0; }
```

Σύνθετες κλάσεις

- ✓ Πρόκειται για κλάσεις, που στα δεδομένα τους συμπεριλαμβάνουν άλλες απλούστερες κλάσεις
- ✓ Αυτό δηλώνει μία σχέση μεταξύ των κλάσεων



Παράδειγμα

Θεωρούμε κλάση **τετραγώνου**, το οποίο ορίζεται μέσω δύο **σημείων**:

» Το πάνω αριστερά και

» το κάτω δεξιά

Το σημείο ορίζεται μέσω των συντεταγμένων του Χ και Υ.

RECT.H/1 (HPP?)

```
#include <iostream.h>
  class Point
  {
    // no constructor
    public:
      void SetX(int x) { itsX = x; }
      void SetY(int y) { itsY = y; }
      int GetX()const { return itsX;}
      int GetY()const { return itsY;}
    private:
      int itsX;
      int itsY;
  };
```

RECT.H/2

```
class  Rectangle {
    public:
      Rectangle (int top, int left, int bottom, int right);
      ~Rectangle () {}
      int GetTop() const { return itsTop; }
      int GetLeft() const { return itsLeft; }
      int GetBottom() const { return itsBottom; }
      int GetRight() const { return itsRight; }
      Point  GetUpperLeft() const { return itsUpperLeft; }
      Point  GetLowerLeft() const { return itsLowerLeft; }
      Point  GetUpperRight() const { return itsUpperRight; }
```

RECT.H/3

```
Point  GetLowerRight() const { return itsLowerRight; }
void SetUpperLeft(Point Location)  {itsUpperLeft = Location;}
void SetLowerLeft(Point Location)  {itsLowerLeft = Location;}
void SetUpperRight(Point Location)  {itsUpperRight = Location;}
void SetLowerRight(Point Location)  {itsLowerRight = Location;}
void SetTop(int top) { itsTop = top; }
void SetLeft (int left) { itsLeft = left; }
void SetBottom (int bottom) { itsBottom = bottom; }
void SetRight (int right) { itsRight = right; }
int GetArea() const;
```

RECT.H/4

```
private:
      Point  itsUpperLeft;
      Point  itsUpperRight;
      Point  itsLowerLeft;
      Point  itsLowerRight;
      int    itsTop;
      int    itsLeft;
      int    itsBottom;
      int    itsRight;
 };
```

RECT.CPP/1

```
#include "rect.h"
Rectangle::Rectangle(int top, int left, int bottom, int right)  {
        itsTop = top;
        itsLeft = left;
        itsBottom = bottom;
        itsRight = right;
       itsUpperLeft.SetX(left);
       itsUpperLeft.SetY(top);
       itsUpperRight.SetX(right);
       itsUpperRight.SetY(top);
       itsLowerLeft.SetX(left);
       itsLowerLeft.SetY(bottom);
       itsLowerRight.SetX(right);
       itsLowerRight.SetY(bottom);  }
```

RECT.CPP /2

```
int Rectangle::GetArea() const {
        int Width = itsRight-itsLeft;
        int Height = itsTop - itsBottom;
        return (Width * Height);
  }
int main()  {
        Rectangle MyRectangle (100, 20, 50, 80 );
        int Area = MyRectangle.GetArea();
        cout << "Area: " << Area << "\n";
        cout << "Upper Left X Coordinate: ";
        cout << MyRectangle.GetUpperLeft().GetX();
        return 0;  }
```

Παραδειγμα 1.1

```
#include <iostream>
class Rectangle {
    int width, height;
  public:
    Rectangle ();
    Rectangle (int,int);
    int area (void) {return (width*height);}
};
Rectangle::Rectangle () {
  width = 5;
  height = 5;
}
```

Παραδειγμα 1.2

```
Rectangle::Rectangle (int a, int b) {
  width = a;
  height = b;
}

int main () {
  Rectangle rect (3,4);
  Rectangle rectb;
  cout << "rect area: " << rect.area() << endl;
  cout << "rectb area: " << rectb.area() << endl;
  return 0;
}
```

Παραδειγμα 2.1

```
#include <iostream>
class Circle {
    double radius;
  public:
    Circle(double r) : radius(r) { }
    double area() {return radius*radius*3.14159265;}
};
class Cylinder {
    Circle base;
    double height;
public:
    Cylinder(double r, double h) : base (r), height(h) {}
    double volume() {return base.area() * height;}
};
```

Παραδειγμα 2.2

```
int main () {
  Cylinder foo (10,20);

  cout << "foo's volume: " << foo.volume() << '\n';
  return 0;
}
```

Παράδειγμα 3.1

```
#include <iostream>
class Dummy {
  public:
    static int n;
    Dummy () { n++; };
    ~Dummy () { n--; };
};
int Dummy::n=0;
```

Παράδειγμα 3.2

```
int main () {
  Dummy a;
  Dummy b[5];
  Dummy * c = new Dummy;
  cout << a.n << '\n';
  delete c;
  cout << Dummy::n << '\n';
  return 0;
}
```

Παράδειγμα 4.1 (λάθος)

```
#include <iostream>
class MyClass {
  public:
    int x;
    MyClass(int val) : x(val) {}
    int get() {return x;}
};
int main() {
  const MyClass foo(10);
  foo.x = 20;
  cout << foo.x << '\n';
  return 0;
}
```

Τι πάει στραβά εδώ;

```
class Square
{
public:
    int Side;
}
```

Γιατί δεν είναι χρήσιμο;

```
class Cat
{
    int GetAge()const;
private:
    int itsAge;
};
```

Βρείτε λάθη

```
class  TV
{
public:
   void SetStation(int Station);
   int GetStation() const;
private:
   int itsStation;
};
main()
{
   TV myTV;
   myTV.itsStation = 9;
   TV.SetStation(10);
   TV myOtherTv(2);
}
```