Πανεπιστήμιο Αιγαίου

Μεθοδολογίες και Γλώσσες Προγραμματισμού Ι

Συναρτήσεις & Κλάσεις

Εργίνα Καβαλλιεράτου (kavallieratou@aegean.gr)

Μόνιμη Επίκουρος Καθηγήτρια

Τμήμα Μηχανικών Πληροφοριακών & Επικοινωνιακών Συστημάτων

CC BY NC ND

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο

ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ & ΘΡΗΣΚΕΥΜΑΤΩΝ, ΠΟΛΙΤΙΣΜΟΥ & ΑΘΛΗΤΙΣΜΟΥ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ

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

Overloading class member συναρτήσεις/1

```
#include <iostream.h>
typedef unsigned short int USHORT;
enum BOOL { FALSE, TRUE};
class Rectangle {
public:
    Rectangle(USHORT width, USHORT height);
    ~Rectangle(){}
    void DrawShape() const;
    void DrawShape(USHORT aWidth, USHORT aHeight) const;
private:
    USHORT itsWidth;
    USHORT itsHeight; };
```

Overloading class member συναρτήσεις/2

```
Rectangle::Rectangle(USHORT width, USHORT height)  {
    itsWidth = width;
    itsHeight = height;  }

void Rectangle::DrawShape() const  {
    DrawShape( itsWidth, itsHeight); }

void Rectangle::DrawShape(USHORT width, USHORT height) const {
    for (USHORT i = 0; i<height; i++)  {
            for (USHORT j = 0; j< width; j++) {
                    cout << "*"; }
           cout << "\n"; }
}
```

Overloading class member συναρτήσεις/3

```
int main() {
    Rectangle theRect(30,5);
    cout << "DrawShape(): \n";
    theRect.DrawShape();
    cout << "\nDrawShape(40,2): \n";
    theRect.DrawShape(40,2);
    return 0;
}
```

Overloading class member συναρτήσεις/output

DrawShape():

****

****

****

****

****

DrawShape(40,2):

****

****

Overloading τους προκαθορισμένους τελεστές

- ✓ Μπορούμε να κάνουμε overloading του προκαθορισμένους τελεστές δηλώνοντας συναρτήσεις της μορφής

  returnType operator op (parameters)

- ✓ όπου, op ο τελεστής για overloading

  void operator++ ()

- ✓ Όταν υπερφορτώνεται ένας τελεστής, αποκτά διαφορετική σημασία μόνο για την κλάση για την οποία υπερφορτώνεται. Διατηρεί την αρχική του σημασία για τους υπόλοιπους τύπους δεδομένων.

Overloading τους προκαθορισμένους τελεστές

- ✓ Δεν μπορούν να υπερφορτωθούν όλοι οι τελεστές της C++
- ✓ Δεν μπορούμε να δημιουργήσουμε νέους τελεστές. Πρέπει να χρησιμοποιήσουμε τους υπάρχοντες
- ✓ Δεν μπορούμε να αλλάξουμε την προτεραιότητα των τελεστών

Overloading ++ (Prefix)/1

```
#include <iostream.h>
typedef unsigned short USHORT;
class Counter  {
public:
    Counter();
    ~Counter(){}
    USHORT GetItsVal()const { return itsVal; }
    void SetItsVal(USHORT x) {itsVal = x; }
    void Increment() { ++itsVal; }
    void operator++ () { ++itsVal; }
private:
    USHORT itsVal; };
Counter::Counter(): itsVal(0)  {};
```

Overloading ++(Prefix)/2

```
int main() {
    Counter i;
    cout << "The value of i is " << i.GetItsVal() << endl;
    i.Increment();
    cout << "The value of i is " << i.GetItsVal() << endl;
    ++i;
    cout << "The value of i is " << i.GetItsVal() << endl;
    return 0;
}
```

```
The value of i is 0
The value of i is 1
The value of i is 2
```

Υπερφόρτωση δυαδικών τελεστών

- ✓ Οι δυαδικοί τελεστές είναι αυτοί που έχουν δύο μέλη, ένα αριστερό και ένα δεξί (π.χ. /).
- ✓ Το αντικείμενο που καλεί τη συνάρτηση υπερφόρτωσης τελεστή είναι το αριστερό μέλος του τελεστή.
- ✓ Στη συνάρτηση μεταβιβάζεται ως παράμετρος το δεξί μέλος του τελεστή

Υπερφόρτωση δυαδικών τελεστών

```
rectangle rectangle::operator+ (rectangle op2)
{
rectangle tt;
// οι ιδιότητες στην κλάση rectangle είναι δηλωμένες ως public
tt.plevra_a = this->plevra_a+op2. plevra_a;
tt.plevra_b = this->plevra_b+op2. plevra_b;
return tt;
}
// Για παράδειγμα στην main μπορούμε να τον χρησιμοποιήσουμε ως:
rectangle rec1(10,2), rec2(5,7), rec3;
rec3=rec1 + rec2;
```

Υπερφόρτωση δυαδικών τελεστών

```
bool rectangle::operator> (rectangle op2)
{
if (this->emvado() > op2.emvado())
        return true;
else
        return false;
}

…
rectangle rec1(10,2), rec2(5,7);
if (rec2>rec1)  cout  << “rec2 > rec1” << endl;
```

Overloading ++ (Postfix)/1

```
#include <iostream.h>
typedef unsigned short USHORT;
class Counter  {
public:
    Counter();
    ~Counter(){}
    USHORT GetItsVal()const { return itsVal; }
    void SetItsVal(USHORT x) {itsVal = x; }
    void Increment() { ++itsVal; }
    int operator++ (int) { int temp=itsVal; ++itsVal; return temp;}
private:
    USHORT itsVal; };
Counter::Counter(): itsVal(0)  {};
```

Overloading ++ (Postfix)/2

```
int main() {
    Counter i;
    cout << "The value of i is " << i.GetItsVal() << endl;
    i.Increment();
    cout << "The value of i is " << i.GetItsVal() << endl;
    cout << "The value of i is " << i++ << endl;
    cout << "The value of i is " << i.GetItsVal() << endl;}
```

```
The value of i is 0
The value of i is 1
The value of i is 1
The value of i is 2
```

Παράδειγμα

Υπερφόρτωση του τελεστή συνάρτησης : αναφερόμαστε στα αντικείμενα μιας κλάσης σε μορφή συναρτήσεων

```
rectangle rectangle::operator() (float a, float b)
{
this->plevra_a=a;
this->plevra_b=b;
return *this;
}
…
rectangle rec1(10,2);
rec1(5, 9) // Προσοχή απλά θέτουμε διαστάσεις 5x9 στο αντικείμενο rec1.
```

Overloading τους προκαθορισμένους τελεστές

- ✓ Η υπερφόρτωση τελεστών μπορεί να υλοποιηθεί και με τη χρήση συναρτήσεων που δεν είναι μέλη μιας κλάσης.
- ✓ Συνήθως οι συναρτήσεις αυτές είναι friend συναρτήσεις της κλάσης για την οποία υπερφορτώνεται ο τελεστής.

Προκαθορισμένες Τιμές/1

```
#include <iostream.h>
typedef unsigned short int USHORT;
enum BOOL { FALSE, TRUE};
class Rectangle {
public:
    Rectangle(USHORT width, USHORT height);
     ~Rectangle(){}
    void DrawShape(USHORT aWidth, USHORT aHeight, BOOL
                                          UseCurrentVals = FALSE) const;
private:
    USHORT itsWidth;
    USHORT itsHeight; };
Rectangle::Rectangle(USHORT width, USHORT height):
                    itsWidth(width), itsHeight(height) {}
```

Προκαθορισμένες Τιμές/2

```
void Rectangle::DrawShape(USHORT width, USHORT height, BOOL
                                                UseCurrentValue ) const  {
    int printWidth;
    int printHeight;
    if (UseCurrentValue == TRUE) {
            printWidth = itsWidth;
            printHeight = itsHeight; }
    else {
            printWidth = width;
            printHeight = height; }
    for (int i = 0; i<printHeight; i++) {
            for (int j = 0; j< printWidth; j++) {
                    cout << "*"; }
            cout << "\n"; }
}
```

Προκαθορισμένες Τιμές/3

```
int main()  {
    Rectangle theRect(30,5);
    cout << "DrawShape(0,0,TRUE)...\n";
    theRect.DrawShape(0,0,TRUE);
    cout <<"DrawShape(40,2)...\n";
    theRect.DrawShape(40,2);
    return 0;
}
```

Προκαθορισμένες Τιμές/output

DrawShape(0,0,TRUE)...

****

****

****

****

****

DrawShape(40,2)...

****

****

Boolean τιμές

- ✓Η C++ θεωρεί μια τιμή FALSE όταν είναι μηδέν και TRUE όλες τις άλλες τιμές.

Overloading vs Default

Χρησιμοποιήστε Overloading συναρτήσεις αντί Προκαθορισμένες τιμές, όταν

- ✓ Δεν υπάρχει λογική προκαθορισμένη τιμή
- ✓ Χρειάζεστε διαφορετικές εκδοχές των διαδικασιών (αλγορίθμων)
- ✓ Χρειάζεστε διαφορετικές παραμέτρους εισόδου

Default Constructor

- ✓ Αν δεν δηλώσουμε constructor, ο compiler χρησιμοποιεί ένα default constructor, χωρίς παραμέτρους, που δεν κάνει τίποτα.
- ✓ Όταν ορίσουμε έναν οποιοδήποτε constructor ο default του compiler παύει να υπάρχει.
- ✓ Για αυτό αν θέλουμε να υπάρχει και constructor χωρίς παραμέτρους θα πρέπει να τον ορίσουμε εμείς.
- ✓ Εξ'ορισμού ένας constructor χωρίς παραμέτρους, ονομάζεται default constructor.
- ✓ Ο default constructor που ορίζουμε μπορεί να λειτουργεί όπως επιθυμούμε.

Constructor overloading/1

```
#include <iostream.h>
class Rectangle {
public:
    Rectangle();
    Rectangle(int width, int length);
    ~Rectangle() {}
    int GetWidth() const { return itsWidth; }
    int GetLength() const { return itsLength; }
private:
    int itsWidth;
    int itsLength; };
Rectangle::Rectangle() { itsWidth = 5;  itsLength = 10;  }
Rectangle::Rectangle (int width, int length)  {
    itsWidth = width; itsLength = length;  }
```

Constructor overloading/2

```
int main()  {
    Rectangle Rect1;
    cout << "Rect1 width: " << Rect1.GetWidth() << endl;
    cout << "Rect1 length: " << Rect1.GetLength() << endl;
    int aWidth, aLength;
    cout << "Enter a width: ";
    cin >> aWidth;
    cout << "\nEnter a length: ";
    cin >> aLength;
    Rectangle Rect2(aWidth, aLength);
    cout << "\nRect2 width: " << Rect2.GetWidth() << endl;
    cout << "Rect2 length: " << Rect2.GetLength() << endl;
    return 0
}
```

Constructor overloading/output

Rect1 width: 5

Rect1 length: 10

Enter a width: 20

Enter a length: 50

Rect2 width: 20

Rect2 length: 50

Αρχικοποίηση Αντικειμένων

- ✓ Οι Constructors περιλαμβάνουν δύο τμήματα:

  την αρχικοποίηση και το σώμα.

- ✓ Οι μεταβλητές μπορούν να πάρουν τιμή σε οποιοδήποτε από τα δύο τμήματα:
  - Είτε αρχικοποιώντας τους στο τμήμα αρχικοποίησης
  - Είτε δίνοντας τιμή στο σώμα του

```
CAT():
itsAge(5),              // λίστα αρχικοποίησης
itsWeight(8)
{}                      // σώμα constructor
```

Αρχικοποίηση Αντικειμένων

```
Rectangle::Rectangle():
itsWidth(5), itsLength(10)
{ };

Rectangle::Rectangle (int width, int length):
itsWidth(width), itsLength(length)
{ };
```

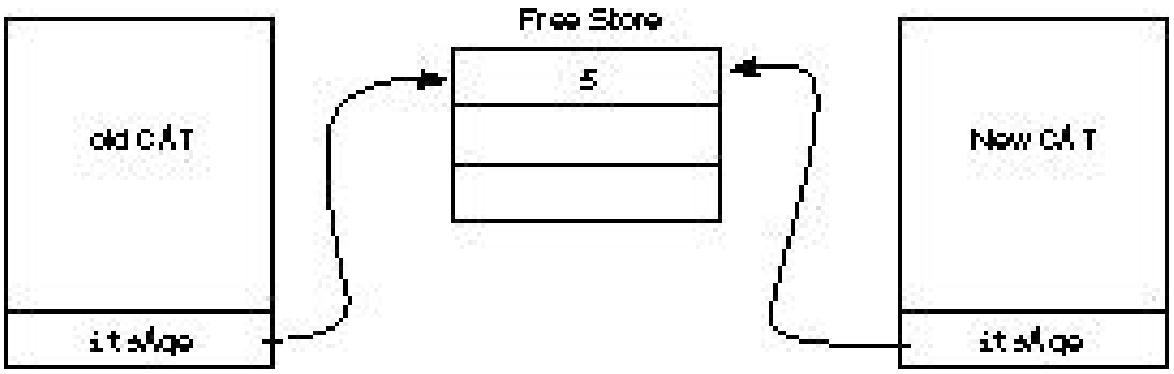
Copy Constructor

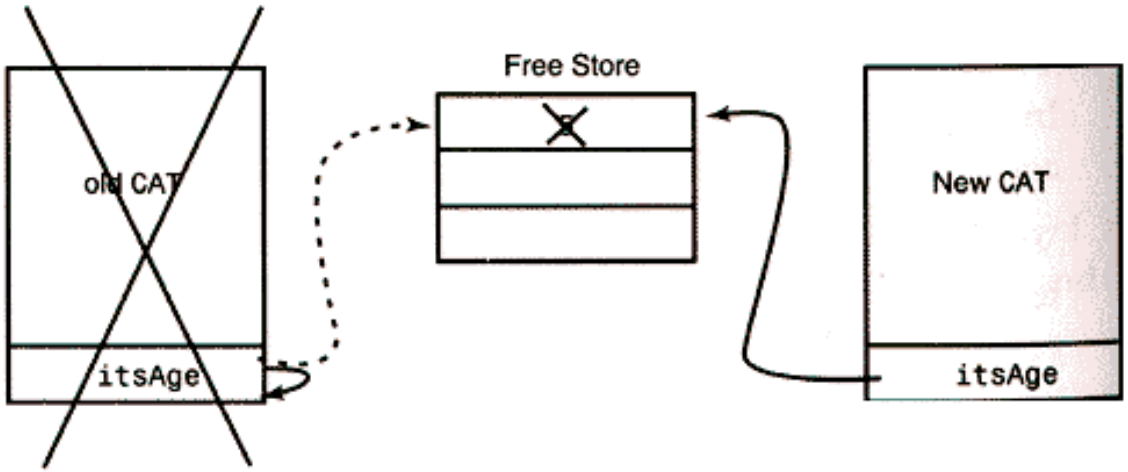
- ✓ Ο compiler διαθέτει τον Copy Constructor για τις περιπτώσεις που χρειαζόμαστε αντίγραφο αντικειμένου, πχ πέρασμα σε συνάρτηση
- ✓ Όλοι οι copy constructors παίρνουν ως παράμετρο μια αναφορά σε αντικείμενο της ίδιας κλάσης.

CAT(const CAT & theCat);

- ✓ Ο default copy constructor αντιγράφει κάθε δεδομένο του αντικειμένου-παραμέτρου στα δεδομένα ενός νέου αντικειμένου.
- ✓ Αυτό μπορεί να είναι πρόβλημα όταν τα δεδομένα είναι δείκτες καθώς θα δείχνουν και οι δύο στην ίδια θέση μνήμης αλλά θα πάψουν να υπάρχουν με την καταστροφή του αντίστοιχου αντικειμένου

Copy Constructor





Copy Constructor

- ✓ Η λύση είναι να δημιουργήσουμε το δικό μας copy constructor που θα δεσμεύσει την απαιτούμενη μνήμη εξ' αρχής πριν αντιγράψει τις τιμές σε νέα μνήμη.

Copy Constructor/1

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

Copy Constructor/2

```
CAT::CAT() {
    itsAge = new int;
    itsWeight = new int;
    *itsAge = 5;
    *itsWeight = 9; }
CAT::CAT(const CAT & rhs) {
    itsAge = new int;
    itsWeight = new int;
    *itsAge = rhs.GetAge();
    *itsWeight = rhs.GetWeight(); }
CAT::~CAT() {
    delete itsAge;
    itsAge = 0;
    delete itsWeight; itsWeight = 0; }
```

Copy Constructor/3

```
int main()  {
    CAT frisky;
    cout << "frisky's age: " << frisky.GetAge() << endl;
    cout << "Setting frisky to 6...\n";
    frisky.SetAge(6);
    cout << "Creating boots from frisky\n";
    CAT boots(frisky);
    cout << "frisky's age: " << frisky.GetAge() << endl;
    cout << "boots' age: " << boots.GetAge() << endl;
    cout << "setting frisky to 7...\n";
    frisky.SetAge(7);
    cout << "frisky's age: " << frisky.GetAge() << endl;
    cout << "boot's age: " << boots.GetAge() << endl;
    return 0;  }
```

Copy Constructor/output

frisky's age: 5

Setting frisky to 6...

Creating boots from frisky

frisky's age: 6

boots' age: 6

setting frisky to 7...

frisky's age: 7

boots' age: 6

Counter II/1

```
#include <iostream.h>
typedef unsigned short USHORT;
class Counter  {
public:
    Counter();
    ~Counter(){}
    USHORT GetItsVal()const { return itsVal; }
    void SetItsVal(USHORT x) {itsVal = x; }
    void Increment() { ++itsVal; }
private:
    USHORT itsVal;  };
Counter::Counter():  itsVal(0)  { };
```

Counter II/2

```
int main()  {
    Counter i;
    cout << "The value of i is " << i.GetItsVal() << endl;
    i.Increment();
    cout << "The value of i is " << i.GetItsVal() << endl;
    return 0;
}
```

```
The value of i is 0
The value of i is 1
```