Ελληνική Δημοκρατία
Τεχνολογικό Εκπαιδευτικό
Ίδρυμα Ηπείρου

Αντικειμενοστραφής Προγραμματισμός

Ενότητα 3 : **Κλάσεις**

Ιωάννης Τσούλος

Τμήμα Μηχανικών Πληροφορικής Τ.Ε

Αντικειμενοστραφής Προγραμματισμός

Ενότητα 3 : Κλάσεις

Ιωάννης Τσούλος
Επίκουρος Καθηγητής

Άρτα, 2015

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
ΕΣΠΑ 2007-2013
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

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 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.
- Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο ΤΕΙ Ηπείρου**» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.

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
ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Εισαγωγικά

- Η αντικειμενοστραφής πλευρά της C++ φαίνεται με την χρήση των **αντικειμένων** και των **κλάσεων** στις οποίες ανήκουν τα αντικείμενα.
- Όλα τα αντικείμενα που έχουν κοινά χαρακτηριστικά ανήκουν στην ίδια κλάση και κάθε ένα από αυτά λέγεται ότι είναι “**στιγμιότυπο**” (instance) της κλάσης.

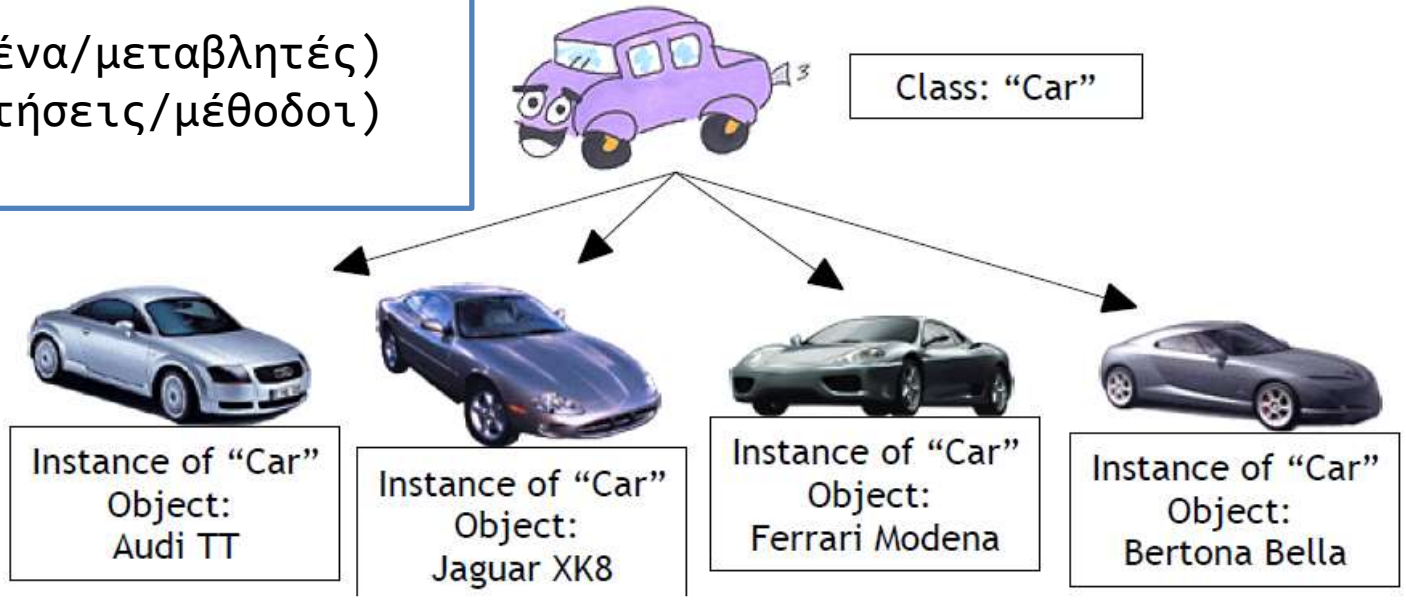
Παράδειγμα κλάσης

- **Κλάση "αυτοκίνητο":** Θεωρεί μια γενική εικόνα του αυτοκινήτου με κάποια χαρακτηριστικά που υπάρχουν σε όλα τα **στιγμιότυπα**.
- Συγκεκριμένα, η κλάση αυτοκίνητο πιθανώς να ορίζει ότι είναι τετράτροχο, έχει τιμόνι, ταχύτητες, πεντάλ και καθίσματα για τους οδηγούς αλλά δεν ορίζει με σαφήνεια το σχήμα ή τους μηχανισμούς όλων των εξαρτημάτων, ούτε τα χαρακτηριστικά της μηχανής (κυβικά, ίπποι, κύλινδροι, κλπ).
- Αυτά είναι χαρακτηριστικά που αφορούν το κάθε **αντικείμενο** ξεχωριστά (ή κάποια υποκατηγορία/υποκλάση της κλάσης "αυτοκίνητο").

Παράδειγμα κλάσης

- Στη C++ η κλάση δηλώνεται με τη λέξη `class`. Στο παράδειγμα του αυτοκινήτου, έστω ότι θέλουμε να δηλώσουμε μια κλάση με το όνομα Car:

```
class Car
{
    (δεδομένα/μεταβλητές)
    (συναρτήσεις/μέθοδοι)
};
```



member variables

- Τα **χαρακτηριστικά** του αντικειμένου είναι τα δεδομένα γύρω από τα οποία πρέπει να αναπτύξουμε το πρόγραμμά μας.
- Στο παράδειγμα με το αυτοκίνητο, τα **δεδομένα** αυτά είναι η γωνία στρέψης του τιμονιού, η πίεση που ασκούμε στα πεντάλ, η θέση του μοχλού ταχυτήτων και άλλα.
- Στον προγραμματισμό, αυτά τα δεδομένα θα πρέπει να μοντελοποιηθούν, δηλαδή να γίνει η αντιστοιχία τους σε μεταβλητές τις οποίες μπορούμε να επεξεργαστούμε στο πρόγραμμά μας.

member variables

```
class Car
{
    // Γωνία στρέψης του τιμονιού
    float steering_angle;
    // Ποσοστό πατήματος του γκαζιού (0 = καθόλου, 100 = τερματισμένο!)
    float gas_pedal;
    // Ποσοστό πατήματος του φρένου (0 = καθόλου, 100 = τερματισμένο!)
    float break_pedal;
    // Ποσοστό πατήματος του συμπλέκτη (0 = καθόλου,
    // 100 = τερματισμένο!)
    float clutch;
    // θέση της τρέχουσας ταχύτητα (πιθανές τιμές: 0, 1,2,3,4,5,
    // 0 = νεκρό, -1 = όπισθεν)
    int gear;
    // μεταβλητές που καθορίζουν την επιτάχυνση, την ταχύτητα του
    // αυτοκινήτου και τις στροφές του κινητήρα
    float acceleration, speed, rpm;
}
```

member variables

- Εδώ πρέπει να σημειωθεί ότι οι μεταβλητές της κλάσης έχουν διαφορετικές τιμές για κάθε αντικείμενο, και κάθε αντικείμενο έχει πρόσβαση μόνο στις δικές του μεταβλητές.
- Επίσης, ο τύπος δεδομένων που επιλέχθηκε για κάθε μεταβλητή εξαρτάται από το είδος της πληροφορίας που θα κρατάει αυτή η μεταβλητή. Για παράδειγμα, εφόσον η γωνία στρέψης του τιμονιού θα είναι σε μοίρες, είναι λογικό να χρησιμοποιήσουμε ένα τύπο που θα μπορεί να κρατήσει δεκαδικούς αριθμούς, όπως ο float.
- Αυτά, λοιπόν, είναι τα δεδομένα μας ομαδοποιημένα υπό την έννοια της κλάσης “Car”.
- Αυτή τη στιγμή δεν είναι τίποτε παραπάνω από μια ομάδα μεταβλητών χωρίς να έχουμε ορίσει τον τρόπο επεξεργασίας τους. Γι' αυτό είναι απαραίτητη η ύπαρξη του κατάλληλου interface (member methods) των δεδομένων με τον χρήστη.

member methods

- Οι μέθοδοι (methods) σε μια κλάση, δεν είναι παρά συναρτήσεις (υπορουτίνες) που προσφέρουν πρόσβαση στα δεδομένα του εκάστοτε αντικειμένου της κλάσης.
- Η αλληλεπίδραση του χρήστη με κάθε αντικείμενο γίνεται μέσω των μεθόδων της κλάσης, οι οποίες καθορίζουν και τον τρόπο χειρισμού των μεταβλητών του αντικειμένου.
- Στο παράδειγμά μας, οι μέθοδοι θα καθορίζουν με ποιον τρόπο θα στρίβουμε το τιμόνι, θα αυξάνουμε τη το γκάζι ή το φρένο, θα αλλάζουμε ταχύτητες αλλά και πώς θα μπορούμε να γνωρίζουμε την ταχύτητα του αυτοκινήτου, τις στροφές του κινητήρα δηλαδή πληροφορίες που δε μπορούμε να ελέγξουμε άμεσα.

member methods

Πιθανές μέθοδοι για την κλάση "Car"

```
// Αλλαγή της γωνία στρέψης του τιμονιού, <relative_angle> μοίρες
// σε σχέση με την τρέχουσα γωνία.
void turn_wheel(float relative_angle);
// Πάτημα πεντάλ γκαζιού
void press_gas_pedal(float amount);
// Πάτημα πεντάλ φρένου
void press_break_pedal(float amount);
// Πάτημα πεντάλ συμπλέκτη
void press_clutch_pedal(float amount);
// Αλλαγή της ταχύτητας. Επιστρέφει true αν η αλλαγή ήταν επιτυχής
// ή false αν ήταν ανεπιτυχής (π.χ. από 5 σε όπισθεν).
bool change_gear(int new_gear);
// προβολή της τρέχουσας ταχύτητας, επιτάχυνσης και στροφών του
// κινητήρα
float get_acceleration();
float get_speed();
float get_rpm();
```

member methods

- Οι παραπάνω μέθοδοι, όπου κρίνεται αναγκαίο επιστρέφουν κάποιο αποτέλεσμα, είτε ως δεκαδικό αριθμό (float) στην περίπτωση των μεθόδων `get_*()`, ή ως bool στην `change_gear()`.
- Ο τύπος void είναι ειδική περίπτωση που δεν επιστρέφει κάποιο αποτέλεσμα. Χρησιμοποιείται όταν μας ενδιαφέρει η εκτέλεση κάποιας διαδικασίας χωρίς όμως να είναι απαραίτητο να γνωρίζουμε το αποτέλεσμά της, ή μπορεί να μην επιστρέφει κάποιο αποτέλεσμα εξ' αρχής.

Υλοποίηση μιας μεθόδου

- Οι παραπάνω είναι απλώς οι δηλώσεις των μεθόδων, δηλαδή δεν περιέχουν καθόλου κώδικα και πρέπει να τις συμπεριλαμβάνουμε στον ορισμό της κλάσης.
- Για να είναι ολοκληρωμένος ο ορισμός μιας μεθόδου θα πρέπει να συνοδεύεται και από την υλοποίησή της (**implementation**).
- Η υλοποίηση συνιστάται να γίνεται σε ξεχωριστό αρχείο (το implementation αρχείο, με κατάληξη `.cpp`) , για παράδειγμα η υλοποίηση της `turn_wheel()` θα μπορούσε να είναι στο αρχείο `car.cpp`:

Υλοποίηση μιας μεθόδου

```
void Car::turn_wheel(float relative_angle)
{
    steering_angle += relative_angle;
    if (steering_angle <= -720.0)
        steering_angle = -720.0;
    if (steering_angle >= 720.0)
        steering_angle = 720.0;
}
```

- Προσθέσαμε και επιπλέον κώδικα ασφαλείας, ο οποίος απαγορεύει στο τιμόνι να κάνει περισσότερες από 2 στροφές αριστερά ή δεξιά.
- Το κάθε αντικείμενο **πρέπει** να θέσει τους δικούς του περιορισμούς και δικλείδες ασφαλείας με τη μορφή κώδικα στις μεθόδους.

Υλοποίηση μιας μεθόδου

- Στη C++, συνήθως η δήλωση μιας κλάσης, μεταβλητής ή συνάρτησης γίνεται σε ξεχωριστό αρχείο, το αρχείο κεφαλίδας του οποίου το όνομα λήγει σε `.h` ή `.hpp`, ενώ το αρχείο ορισμού -δηλαδή το αρχείο που περιέχει τον πηγαίο κώδικα ορισμού της αντίστοιχης κλάσης, μεταβλητής ή συνάρτησης- λήγει σε `.cpp` ή `.cxx`.
- Δηλαδή για την κλάση Car θα έχουμε δύο αρχεία, το αρχείο κεφαλίδας `Car.h` και το αρχείο ορισμού `Car.cpp`.
- Αυτός ο διαχωρισμός εξυπηρετεί τη γρήγορη χρήση της κλάσης μέσα σε άλλες κλάσεις ή συναρτήσεις.

Δημιουργία αντικειμένων

- Δημιουργία με δύο τρόπους, **στατικά και δυναμικά**.
- Η στατική δημιουργία γίνεται όπως και ο ορισμός κάποιας μεταβλητής ενώ η δυναμική γίνεται με τη χρήση της εντολής new της C++.

  new: δημιουργεί μια φυσική αναπαράσταση της κλάσης, ένα μοναδικό στιγμιότυπο, και αν είναι επιτυχής επιστρέφει ένα δείκτη (pointer) σε αυτό, διαφορετικά επιστρέφει μηδέν (0).
- Με αυτό το δείκτη μπορούμε να προσπελάσουμε το αντικείμενο με οποίον τρόπο θέλουμε (και εφόσον το επιτρέπει το ίδιο το αντικείμενο).

Δημιουργία αντικειμένων

- Για την κλάση Car η δημιουργία ενός αντικειμένου στατικά και δυναμικά γίνεται ως εξής (το αντικείμενο acar δημιουργείται στατικά, ενώ το anothercar δημιουργείται δυναμικά):

```
Car acar();
Car *anothercar = new Car();
```

- Αν ο constructor δεν παίρνει παραμέτρους μπορούμε να αποφύγουμε τις παρενθέσεις, δηλαδή το ακόλουθο είναι ισοδύναμο:

```
Car acar;
Car *anothercar = new Car;
```

Δημιουργία αντικειμένων

- Όσον αφορά το αντικείμενο anothercar, η δημιουργία του δεν είναι αναγκαίο να γίνει κατά την δήλωσή του. Το ίδιο αποτέλεσμα μπορούμε να έχουμε και με τις εντολές:

```
Car *anothercar;
anothercar = new Car;
```

Δημιουργία αντικειμένων

- Τα δεδομένα του κάθε αντικειμένου (οι μεταβλητές) αλλά και οι μέθοδοι της κλάσης που αντιστοιχούν στο αντικείμενο, μπορούν να προσπελλαστούν ως εξής:

```
// Η γωνία στρέψης του τιμονιού του acar
acar.steering_angle
// Η γωνία στρέψης του τιμονιού του anothercar
anothercar->steering_angle
// Εντολή στο acar να στρίψει δεξιά 13.4 μοίρες.
acar.turn(13.4);
// Επιστρέφει την τρέχουσα ταχύτητα του acar
float speed = acar.get_speed();
// Εντολή στο anothercar να στρίψει αριστερά 32 μοίρες
anothercar->turn(-32.0);
// Εντολή στο anothercar να βάλει όπισθεν
bool result = anothercar->ghange_gear(-1);
```

Constructors

- Όταν δημιουργείται ένα αντικείμενο (στατικά ή δυναμικά με την εντολή new), στην πραγματικότητα η C++, αφού δεσμεύσει την απαραίτητη μνήμη για το αντικείμενο, εκτελεί μια συγκεκριμένη μέθοδο της κλάσης, τον δημιουργό (constructor).
- Ο δημιουργός πραγματοποιεί τις απαραίτητες ενέργειες για να καταστεί κάποιο αντικείμενο έτοιμο προς χρήση. Αυτές μπορεί να είναι κάτι απλό (ρύθμιση μεταβλητών με αρχικές τιμές), ή πιο περίπλοκο (δημιουργία σύνδεσης με μια βάση δεδομένων, αρχικοποίηση των πινάκων SQL).
- Ο δημιουργός έχει τη μορφή μιας μεθόδου με το όνομα της κλάσης και χωρίς τύπο.

Constructors

- **πχ. πιθανός δημιουργός**

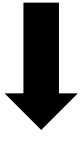
– Ο δημιουργός πραγματοποιεί την αρχικοποίηση (initialization) των μεταβλητών του αντικειμένου ώστε αυτό να είναι έτοιμο προς χρήση.

– Μπορούμε να έχουμε περισσότερους από έναν δημιουργούς, οι οποίοι να δέχονται διαφορετικές παραμέτρους ο καθένας.

```
Car::Car()
{
    steering_wheel = 0.0;
    gas_pedal = 0.0;
    break_pedal = 0.0;
    float clutch = 0.0;
    int gear = 0;
    acceleration = 0.0;
    speed = 0.0;
    rpm = 0.0;
}
```

Constructors

- Αν μπορούσαμε να ορίσουμε χαρακτηριστικά του κινητήρα (engine_cc: κυβικά, engine_hp: ίπποι) στη δημιουργία του αντικειμένου, θα μπορούσαμε να έχουμε έναν επιπλέον δημιουργό



constructor overloading

```
Car::Car(int cc, int hp)
{
    engine_cc = cc;
    engine_hp = hp;
    // υπόλοιπες εντολές
    αρχικοποίησης του αντικειμένου
}
```

Destructors

- Ό,τι αντικείμενο δημιουργούμε δυναμικά πρέπει να το καταστρέφουμε (εντολή delete) ελευθερώνοντας όλους τους πόρους που δεσμεύσαμε κατά την ύπαρξή του (κλείσιμο αρχείων, αποσύνδεση από βάσεις δεδομένων, τερματισμός threads, αποδέσμευση μνήμης, κλπ).
- Η καταστροφή του αντικειμένου γίνεται καλώντας μια μέθοδο που καλείται καταστροφέας (destructor).
- Το όνομα του destructor είναι το ίδιο με της κλάσης προπορευόμενο από τον τελεστή ~. Δηλαδή για την κλάση Car, το όνομα του καταστροφέα είναι ~Car().

Ο δείκτης αναφοράς this

- Μέσα σε μια μέθοδο, μπορούμε να χρησιμοποιήσουμε μια μεταβλητή της κλάσης απλώς με το όνομά της, αναφερόμενοι στην τιμή που έχει η μεταβλητή για το συγκεκριμένο αντικείμενο.
- Για το σκοπό αυτό μπορούμε να χρησιμοποιήσουμε τη μεταβλητή `this` που επιστρέφει πάντα ένα δείκτη στο τρέχον αντικείμενο (δηλαδή αυτό που καλεί την μέθοδο).

Ο δείκτης αναφοράς this

- Με το δείκτη `this`, η μέθοδος `turn_wheel()` που είδαμε παραπάνω μετασχηματίζεται ως εξής:

```
void turn_wheel(float relative_angle)
{
    this->steering_angle += relative_angle;
    if (this->steering_angle <= -720.0)
        this->steering_angle = -720.0;
    if (this->steering_angle >= 720.0)
        this->steering_angle = 720.0;
}
```

- Δείκτης `this`, ιδιαίτερα χρήσιμος ειδικά σε περιπτώσεις διαχείρισης περισσότερων από ένα όμοιων αντικειμένων στην ίδια μέθοδο.

Method Overloading

- Σε ένα πρόγραμμα, υπάρχει περίπτωση να χρειαστεί να εκτελέσουμε την ίδια διαδικασία με ελαφρώς διαφορετικά δεδομένα.
- Αυτό συνήθως σημαίνει ότι θα χρειαστεί να έχουμε διαφορετικές συναρτήσεις/μεθόδους για κάθε διαφορετική παράμετρο που δίνουμε.
- Για παράδειγμα, ας υποθέσουμε ότι ο χρήστης καλεί την μέθοδο `turn_wheel()` με παράμετρο ένα `int` αντί για `float`. Σε μια γλώσσα όπως η C θα έπρεπε να έχουμε δύο συναρτήσεις/μεθόδους, μια για κάθε διαφορετική παράμετρο και μάλιστα με διαφορετικό όνομα…

Method Overloading

```
void Car::turn_wheel_int(int relative_angle)
{
    this->steering_angle += (float) relative_angle;
    if (this->steering_angle <= -720.0)
        this->steering_angle = -720.0;
    if (this->steering_angle >= 720.0)
        this->steering_angle = 720.0;
}
void Car::turn_wheel_float(float relative_angle)
{
    steering_angle += relative_angle;
    if (steering_angle <= -720.0)
        steering_angle = -720.0;
    if (steering_angle >= 720.0)
        steering_angle = 720.0;
}
```

Method Overloading

- Η τεχνική του method overloading, επιτρέπει τον ορισμό διάφορων παραλλαγών μιας μεθόδου αναλόγως τις ζητούμενες παραμέτρους που δέχεται αυτή.

```
void Car::turn_wheel(float relative_angle)
{
    steering_angle += relative_angle;
    if (steering_angle <= -720.0)
        steering_angle = -720.0;
    if (steering_angle >= 720.0)
        steering_angle = 720.0;
}
void Car::turn_wheel(int relative_angle)
{
    turn_wheel((float) relative_angle);
}
```

Βιβλιογραφία

1. Εγχειρίδιο της C++, 2η Ελληνική έκδοση, Jesse Liberty, Γκιούρδας.
2. Μάθετε τη C++, 2η Ελληνική έκδοση, Jesse Liberty , Γκιούρδας.
3. Προγραμματισμός με τη γλώσσα C++ Μέρος Α, Αλεβίζος Θ., Έκδοση ΤΕΙ Καβάλας
4. C++ Αντικειμενοστραφής Προγραμματισμός Υπολογιστών Τομαράς Α., , Εκδόσεις Νέων Τεχνολογιών.
5. Ανακαλύψτε τη γλώσσα C, J. Purdum, Εκδόσεις Δίαυλος.
6. Εισαγωγή στο Συστηματικό Προγραμματισμό και στη γλώσσα C++, Σ. Μπαλτζής, εκδόσεις πανεπιστημίου Ιωαννίνων.
7. C++ From the beginning, Jan Skansholm, Addison Wesley.
8. The design and analysis of computer algorithms, A.V. AHO, J.E. HOPCROFT, J.D. ULLMANN, Addison Wesley 1974.
9. Structure and Interpretation of Computer Programs, H. ABELSON, G.J. SUSSMAN, J. SUSSMAN, MIT Press, Mc Graw Hill Book Company, 1985
10. The art of computer programming, D.E. KNUTH, Addison-Wesley.

Σημείωμα Αναφοράς

Έκδοση: 1.0 Άρτα, 2015. Διαθέσιμο από τη δικτυακή διεύθυνση:
http://eclass.teiep.gr/courses/COMP113/

Σημείωμα Αδειοδότησης

[1] http://creativecommons.org/licenses/by-nc-nd/4.0/deed.el

Τέλος Ενότητας

Επεξεργασία: Ευάγγελος Καρβούνης

Άρτα, 2015

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επενδύοντας στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ & ΘΡΗΣΚΕΥΜΑΤΩΝ, ΠΟΛΙΤΙΣΜΟΥ & ΑΘΛΗΤΙΣΜΟΥ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
ΕΣΠΑ 2007-2013
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Τέλος Ενότητας

Κλάσεις