

**Ανοικτά Ακαδημαϊκά Μαθήματα**

**Τεχνολογικό Εκπαιδευτικό Ίδρυμα Αθήνας**

**Αντικειμενοστρεφής Προγραμματισμός (Ε)**

**Ενότητα 2:** Κατασκευαστές - καταστροφείς

Κλειώ Σγουροπούλου

Τμήμα Μηχανικών Πληροφορικής ΤΕ

|  |  |
| --- | --- |
| Το περιεχόμενο του μαθήματος διατίθεται με άδεια Creative Commons εκτός και αν αναφέρεται διαφορετικά | 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. |

Περιεχόμενα

[Αντικειμενοστρεφής Προγραμματισμός- Εργαστήριο 2 1](#_Toc402941224)

[Κατασκευαστές - Καταστροφείς 1](#_Toc402941225)

[Πολυμορφισμός – Υπερφόρτωση συνάρτησης 3](#_Toc402941226)

[Αναφορές (references) 4](#_Toc402941227)

# Αντικειμενοστρεφής Προγραμματισμός- Εργαστήριο 2

## Κατασκευαστές - Καταστροφείς

Όταν δημιουργούμε ένα αντικείμενο είναι πολλές φορές απαραίτητο να δώσουμε κάποιες αρχικές τιμές στις μεταβλητές του. Ένας τρόπος για να γίνει αυτό είναι δημιουργώντας μια συνάρτηση που θα δώσει τιμές σε όλες τις παραμέτρους του αντικειμένου. Παράδειγμα μιας τέτοιας συνάρτησης είναι η συνάρτηση set\_values

της κλάσης Vehicle στην 3η άσκηση του 1ου Εργαστηρίου.

Ο πιο σωστός όμως τρόπος είναι η αρχικοποίηση του αντικειμένου μέσω μιας ειδικής συνάρτησης που ονομάζεται **κατασκευαστής ή συνάρτηση αρχικών συνθηκών (construction)**. Ο κατασκευαστής αποτελεί μέλος της κλάσης που ορίζει το αντικείμενο, έχει *το ίδιο όνομα με την κλάση* και εκτελείται αυτόματα με τη διακήρυξη του αντικειμένου. Δεν έχει τύπο επιστρεφόμενης τιμής και μπορεί να υπερφορτωθεί.

Σε αντίθεση με τις συναρτήσεις αρχικών συνθηκών οι συναρτήσεις **τελικών συνθηκών, καταστροφείς (destructors)**, εκτελούνται με την καταστροφή του αντικειμένου. Με το όρο «καταστροφή» εννοούμε τη στιγμή κατά την οποία η μνήμη που δεσμεύτηκε για την καταχώρηση των μεταβλητών του αντικειμένου αποδεσμεύεται από το πρόγραμμα (π.χ. με την έξοδο από μια συνάρτηση όπου το αντικείμενο ορίστηκε ως τοπική μεταβλητή). Οι καταστροφείς έχουν και αυτοί το ίδιο όνομα με την κλάση, ενώ πριν από το όνομα τίθεται ο χαρακτήρας ~. Ένα παράδειγμα χρήσης ενός καταστροφέα είναι όταν ένα αντικείμενο θα πρέπει να απελευθερώσει τη μνήμη που προηγούμενα είχε δεσμεύσει με μια συνάρτηση δυναμικής κατανομής μνήμης.

#include <iostream>

#include <string>

using namespace std;

class student

{

public:

string name;

int code;

student();

student(string \_name, int \_code);

~student();

void display();

};

student::student()

{

name = "N/A"; //not available

code = 0;

}

student::student(string \_name, int \_code)

{

name = \_name;

code = \_code;

}

student::~student()

{

cout << "destructor called" << endl;

}

void student::display()

{

cout << "Name: " << name << ", code: " << code << endl;

}

int main()

{

student s1;

student s2("John Doe", 2323);

s1.display();

s2.display();

system("pause");

return 0;

}

Στο προηγούμενο παράδειγμα υπάρχουν δύο κατασκευαστές: ο default κατασκευαστής που δίνει default αρχικές τιμές σε όλα τα αντικείμενα που δημιουργούνται από αυτόν, και ένας γενικός κατασκευαστής (general constructor) που μπορεί να δώσει την αρχική κατάσταση που θέλουμε εμείς.

Επίσης, ορίζεται και ο destructor της κλάσης student, ο οποίος όμως δεν επιτελεί κάποια σημαντική λειτουργία. Για τον λόγο αυτό θα μπορούσαμε και να τον παραλείψουμε. Θα δούμε στο 5ο εργαστήριο πότε μας είναι απαραίτητος

Επισημαίνεται τέλος ότι μπορούμε να συμπυκνώσουμε 2 κατασκευαστές σε έναν, όπως παρακάτω. Αν στην κλήση του κατασκευαστή δεν δοθούν παράμετροι, τότε χρησιμοποιούνται οι default τιμές.

student::student(string \_name = "N/A", int \_code=0)

{

name = \_name;

code = \_code;

}

Γενικότερα οι βασικοί τύποι των κατασκευαστών που παρέχει η γλώσσα C++ είναι:

|  |  |
| --- | --- |
| Τύπος | Γενική Μορφή |
| Default constructor | X(void)  X(a1=d1, a2=d2, …) |
| General constructor | X(a1, a2, a3, …) |
| Copy constructor | X(X&, a2=d2, …)  X(const X&, a2=d2, …) |
| Conversion constructor | X(T, a2=d2, …)  X(T&, a2=d2, …)  X(const T, a2=d2, …) X(const T&, a2=d2, …) |

#### Άσκηση 1:

Στην κλάση Vehicle που ορίσαμε στο Εργαστήριο 1, να υλοποιήσετε τους απαραίτητους constructors.

Υπενθυμίζεται ότι η κλάση Vehicle περιέχει τις εξής παραμέτρους:

* Μάρκα (π.χ. Toyota)
* Μοντέλο (π.χ. Avensis)
* Έτος κατασκευής (π.χ. 2007)
* Αριθμός πινακίδας
* Είδος (Ι.Χ, φορτηγό, μοτοσυκλέτα)

## Πολυμορφισμός – Υπερφόρτωση συνάρτησης

Με τον όρο πολυμορφισμό περιγράφουμε, μεταξύ άλλων, τη δυνατότητα μιας γλώσσας προγραμματισμού να χρησιμοποιεί το ίδιο συμβολικό όνομα για διαφορετικές χρήσεις. Η C++ ως αντικειμενοστρεφής γλώσσα προγραμματισμού δίνει δυνατότητα να έχουμε επικάλυψη των συναρτήσεων. Μπορούμε δηλαδή να χρησιμοποιήσουμε το ίδιο όνομα για να περιγράψουμε δυο ή περισσότερες συναρτήσεις.

Στο παράδειγμά μας ορίζουμε τρεις διαφορετικές συναρτήσεις abs(int x), abs(float x), abs(double x) με το ίδιο όνομα, κάθε μία από τις οποίες επιστρέφει την απόλυτη τιμή ενός int, ενός float και ενός double αντίστοιχα. Ο μεταγλωττιστής θα αντιλαμβάνεται ποια από τις τρεις πρέπει να χρησιμοποιηθεί κάθε φορά, ανάλογα με το πλήθος και *των τύπο* των ορισμάτων που δίνονται.

#include <iostream> using namespace std;

int abs(int a); double abs(double a);

int main()

{

int i = -100;

double j = 2.1345;

cout << abs(i) << endl; cout << abs(j) << endl; system("pause");

return 0;

}

int abs(int a)

{

if(a < 0)

return -a;

else

return a;

}

double abs(double a)

{

if(a < 0)

return -a;

else

return a;

}

#### Άσκηση 2:

Να προσθέσετε μία υπερφορτωμένη έκδοση της abs(), η οποία να δέχεται δύο ακέραιες παραμέτρους και να εμφανίζει την απόλυτη τιμή του αθροίσματός τους.

#### Άσκηση 3:

Να ορίσετε μία κλάση rectangle, η οποία να περιέχει τα παρακάτω μέλη:

α) το μήκος και το ύψος του ορθογωνίου (να είναι private)

β) μία συνάρτηση που να επιστρέφει το εμβαδό του ορθογωνίου.

γ) δύο υπερφορτωμένες εκδόσεις της συνάρτησης set\_dimensions(). Η πρώτη έκδοση να δέχεται δύο παραμέτρους (πλάτος και ύψος), ενώ η δεύτερη να δέχεται μόνο μια παράμετρο (το πλάτος και το ύψος θα είναι ίδια τετράγωνο).

Να γραφεί ένα πρόγραμμα, το οποίο να δημιουργεί ένα αντικείμενο της κλάσης rect, να καλεί τις δύο εκδόσεις της set\_dimensions() και να εμφανίζει κάθε φορά το εμβαδό του ορθογωνίου.

## Αναφορές (references)

Όταν καλούμε συναρτήσεις, οι τιμές των ορισμάτων τους μεταβιβάζονται σ’ αυτές ακολουθώντας τους κανόνες της κλήσης με τιμή (by value). Αυτό έχει ως συνέπεια οι τροποποιήσεις στα ορίσματα στο εσωτερικό των συναρτήσεων να αφορούν στα αντίγραφα των ορισμάτων και όχι στα αρχικά ορίσματα. Αν θέλουμε

η συνάρτηση να τροποποιήσει τις τιμές των μεταβλητών που δίνονται ως ορίσματα θαπρέπει να δώσουμε ορίσματα δεικτών στις μεταβλητές και να συντάξουμε ανάλογα τον κώδικα.

Η C++ μας παρέχει τη δυνατότητα να έχουμε μεταβίβαση ορισμάτων σε συναρτήσεις με αναφορά (by reference) και όχι με τιμή. Αυτό μπορούμε να το επιτύχουμε αν στη

διακήρυξη της συνάρτησης προτάξουμε τον τελεστή & στα ορίσματα που θέλουμε να καλέσουμε με αναφορά.

#### Παράδειγμα:

#include "stdafx.h"

#include <iostream>

using namespace std;

void increase(int &num) //reference to num

{

num++;

}

int main()

{

int a = 4;

for(int i = 0; i<5; i++)

{

increase(a);

cout << a << endl ;

}

system("pause");

return 0;

}

Επίσης, μπορούμε να δημιουργήσουμε αναφορά (reference) σε μια μεταβλητή, η οποία λειτουργεί σαν alias (ψευδώνυμο, άλλη ονομασία) της μεταβλητής. Αν αλλάξουμε την τιμή του reference, αλλάζει και η τιμή της μεταβλητής. Ουσιαστικά πρόκειται για την ίδια μεταβλητή, απλά με άλλο όνομα.

#### Παράδειγμα:

#include <iostream> using namespace std;

int main()

{

int a = 90;

int &a\_ref = a; //reference to variable "a"

cout << "Contents\n";

cout << a << " " << a\_ref <<endl; cout << "Addresses\n";

cout << &a << " " << &a\_ref <<endl;

system("pause"); return 0;

}

#### Άσκηση 4:

Θεωρήστε μια συνάρτηση swap, σκοπός της οποίας είναι η ανταλλαγή των τιμών δύο μεταβλητών. Αν η συνάρτηση αυτή κληθεί όπως παρακάτω, θα έχει επιτύχει τον

σκοπό της και γιατί;

#include <iostream> using namespace std;

void swap(int a, int b)

{

int temp = a; a = b;

b = a;

}

int main()

{

int a = 3;

int b = 4;

swap(a, b);

cout << a << ", " << b; system("pause");

return 0;

}

Διορθώστε την συνάρτηση (με pointers και με references) ώστε να εκτελεί τη προσδοκώμενη λειτουργία

|  |
| --- |
| **Ανοικτά Ακαδημαϊκά Μαθήματα**  **Τεχνολογικό Εκπαιδευτικό Ίδρυμα Αθήνας** |
| **Τέλος Ενότητας** |
| **Χρηματοδότηση**   * Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα. * Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο ΤΕΙ Αθήνας**» 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. |

**Σημειώματα**

**Σημείωμα Αναφοράς**

Copyright ΤΕΙ Αθήνας, Κλειώ Σγουροπούλου, 2014. Κλειώ Σγουροπούλου. «Αντικειμενοστρεφής Προγραμματισμός». Ενότητα 2: «Κατασκευαστές- καταστροφείς». Έκδοση: 1.0. Αθήνα 2014. Διαθέσιμο από τη δικτυακή διεύθυνση: [ocp.teiath.gr](https://ocp.teiath.gr/).

**Σημείωμα Αδειοδότησης**

Το παρόν υλικό διατίθεται με τους όρους της άδειας χρήσης Creative Commons Αναφορά, Μη Εμπορική Χρήση Παρόμοια Διανομή 4.0 [1] ή μεταγενέστερη, Διεθνής Έκδοση. Εξαιρούνται τα αυτοτελή έργα τρίτων π.χ. φωτογραφίες, διαγράμματα κ.λ.π., τα οποία εμπεριέχονται σε αυτό και τα οποία αναφέρονται μαζί με τους όρους χρήσης τους στο «Σημείωμα Χρήσης Έργων Τρίτων».

[](file:///C:\Users\pantelis\Downloads\%5b1%5d%20http:\creativecommons.org\licenses\by-nc-sa\4.0\)

[1] http://creativecommons.org/licenses/by-nc-sa/4.0/

Ως **Μη Εμπορική** ορίζεται η χρήση:

* που δεν περιλαμβάνει άμεσο ή έμμεσο οικονομικό όφελος από την χρήση του έργου, για το διανομέα του έργου και αδειοδόχο
* που δεν περιλαμβάνει οικονομική συναλλαγή ως προϋπόθεση για τη χρήση ή πρόσβαση στο έργο
* που δεν προσπορίζει στο διανομέα του έργου και αδειοδόχο έμμεσο οικονομικό όφελος (π.χ. διαφημίσεις) από την προβολή του έργου σε διαδικτυακό τόπο

Ο δικαιούχος μπορεί να παρέχει στον αδειοδόχο ξεχωριστή άδεια να χρησιμοποιεί το έργο για εμπορική χρήση, εφόσον αυτό του ζητηθεί.

**Διατήρηση Σημειωμάτων**

* Οποιαδήποτε αναπαραγωγή ή διασκευή του υλικού θα πρέπει να συμπεριλαμβάνει:
* Το Σημείωμα Αναφοράς
* Το Σημείωμα Αδειοδότησης
* Τη δήλωση Διατήρησης Σημειωμάτων
* Το Σημείωμα Χρήσης Έργων Τρίτων (εφόσον υπάρχει) μαζί με τους συνοδευόμενους υπερσυνδέσμους.