ΤΕΙ Λάρισας

Τμήμα **Τεχνολογίας Πληροφορικής & Τηλεπικοινωνιών**

Αντικειμενοστραφής προγραμματισμός **II**

C# (Sharp)

βιβλίο μελέτης θεωρίας – Ασκήσεις Πράξης

έκδοση 1.0.3

Λιόλιος Νικόλαος

Νεβράντζας Βάιος-Γερμανός

Λάρισα

Σεπτέμβριος **2013**

Ευρετήριο

Εισαγωγή

Τι χρειάζεται ένας φοιτητής για τη σωστή παρακολούθηση και συμμετοχή στα μαθήματα;

1. Σελίδα μαθήματος	Εγγραφή	Ο κάθε φοιτητής πρέπει να κάνει εγγραφή στη σελίδα του μαθήματος στην πλατφόρμα e-class **(χωρίς κωδικό)**. Η εγγραφή είναι υποχρεωτική και πρέπει να γίνει μέσα στις **2 πρώτες εβδομάδες** κάθε εξαμήνου.
2. Λογισμικό	**Visual C# 2010** Express Edition	Στο μάθημα κάνουμε χρήση της **Visual C# 2010 Express Edition**, την οποία μπορείτε να κάνετε **download** από το site της Microsoft: (http://www.microsoft.com/express/downloads/#2010-Visual-CS)
	απαιτήσεις	Πρέπει να κάνετε registration στο site της Microsoft για να δας δωθεί ένας **registration code**. Είναι **δωρεάν**.
	Οδηγίες χρήσης	Αναλυτικό **video** υπάρχει στη σελίδα του μαθήματος στο **e-class** στο υπομενού /εργαλείο "**Βίντεο**" .

Ασκήσεις Πράξης

1. Προϋποθέσεις για την συμμετοχή μίας ομάδας εργασίας στις εξετάσεις		Είναι υποχρεωτική η παρακολουθηση του **70%** των .
	τι σημαίνει "**επιτυχής παρακολούθηση**";	Θα σας δίνονται ανά τακτά διαστήματα ασκήσεις κατανόησης εννοιών. Για να θεωρηθεί μία παρακολούθηση επιτυχής θα πρέπει να επιλύσετε σωστά **την άσκηση αυτή.**
	σωστή επίλυση άσκησης	ο κώδικας να μην έχει **συντακτικά** και **λογικά** λάθη.

1

εισαγωγή στη C#

Τι θα δούμε σε αυτό το μάθημα

1. την ιστορία της C#
2. την δημιουργία της C#
3. αντικειμενοστρεφής προγραμματισμός
4. ενθυλάκωση
5. πολυμορφισμός
6. κληρονομικότητα
7. δημιουργία του πρώτου προγράμματος
8. ανάλυση του πρώτου προγράμματος
9. έλεγχος συντακτικών λαθών
10. παράμετροι στη Main()
11. παράδειγμα
12. ερωτήσεις κατανόησης

1.1

Η ιστορία της C#

Από τη C στη C++

Στα τέλη της δεκαετίας του '70 το μέγεθος των έργων που υλοποιούσαν οι προγραμματιστές με τη C έφτασε στα όρια του. Έτσι εμφανίστηκε ένας νέος τρόπος προγραμματισμού, ο αντικειμενοστρεφής προγραμματισμός (**Object Oriented Programming-OOP**) και μία νέα γλώσσα, η αντικειμενοστρεφής γλώσσα C++.

Από τη C++ στη Java

Λόγω της μεγάλης διάδοσης του internet οι προγραμματιστές επιθυμούσαν να μεταφέρουν τον κώδικά τους σε διαφορετικά περιβάλλοντα. Αυτό δεν μπορούσε να γίνει με τη C++, οπότε δημιουργήθηκε η αντικειμενοστρεφής γλώσσα προγραμματισμού **Java**, η οποία έλυνε 2 βασικά προβλήματα :

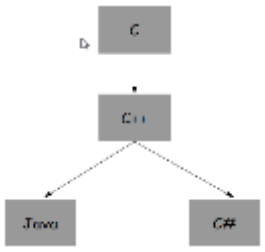
- **μεταφερσιμότητα**: μετάφραση του πηγαίου κώδικα στην ενδιάμεση γλώσσα bytecode και στη συνέχεια εκτέλεση αυτού του κώδικα στην εικονική μηχανή της Java (Java Virtual Machine-JVM).
- ο κώδικας μπορούσε να εκτελεστεί σε οποιαδήποτε περιβάλλον διέθετε εικονική μηχανή Java.
- **ασφάλεια**: επειδή ο JVM τρέχει τον κώδικα bytecode, έχει τον πλήρη έλεγχο του προγράμματος έτσι ώστε να μπορεί να αποτρέψει οποιαδήποτε κακόβουλη ενέργεια ενός Java προγράμματος.

Από τη C++ στη C#

Αν κ η Java έλυσε 2 βασικά προβλήματα, της λείπουν ακόμη 2 χαρακτηριστικά:

1. **διαγλωσσική διαλειτουργικότητα**: για τη δημιουργία μεγάλων κατανεμημένων συστημάτων λογισμικού χρειάζεται ο κώδικας που παράγεται από μία γλώσσα να συνεργάζεται με κώδικα που παράγει μια δεύτερη γλώσσα.
2. **πλήρης ενσωμάτωση με τα Windows**: η java, αν και μπορεί να εκτελεστεί κάτω από τα Windows, δεν συνδέεται στενά με αυτά, σε αντίθεση με τη C#.

δένδρο εξέλιξης



1.2

Η δημιουργία της C#

χαρακτηριστικά της C που κληρονόμησε η C#

- **σύνταξη**
- **λέξεις κλειδιά**
- **τελεστές**

σχέση Java και C#

- μεταφερτός κώδικας
- εκτέλεση προγραμμάτων σε ασφαλές περιβάλλον

σχέση της C# με το .NET Framework

- Το .NET Framework είναι το περιβάλλον χρόνου εκτέλεσης της C#.
- Οι βιβλιοθήκες της γλώσσας ορίζονται από το .NET Framework.

τι είναι το .NET Framework

ένα περιβάλλον που υποστηρίζει την ανάπτυξη και την εκτέλεση άκρως κατανεμημένων εφαρμογών που βασίζονται σε συστατικά. Επιτρέπει στις γλώσσες προγραμματισμού να συνεργάζονται και παρέχει χαρακτηριστικά ασφαλείας, μεταφερτότητας και ένα κοινό μοντέλο προγραμματισμού για τα Windows. Παρέχει τα εξής:

- σύστημα **Common Language Runtime**: διαχειρίζεται την εκτέλεση ενός προγράμματος σε C#.
- **Βιβλιοθήκη Κλάσεων**: παρέχει στα προγράμματα δυνατότητες πρόσβασης στο περιβάλλον χρόνου εκτέλεσης.

1.3

Αντικειμενοστρεφής Προγραμματισμός

ορισμός

τα αντικειμενοστρεφή προγράμματα οργανώνονται γύρω από τα δεδομένα, δηλαδή όταν προγραμματίζουμε στην ουσία ορίζουμε τα δεδομένα και τις ρουτίνες που επιτρέπεται να δρουν επί αυτών των δεδομένων. Έτσι, ένας τύπος δεδομένων ορίζει τι ακριβώς τύπου λειτουργίες μπορούν να εφαρμοστούν επί αυτών των δεδομένων.

χαρακτηριστικά

1. ενθυλάκωση

2. πολυμορφισμός

3.Κληρονομικότητα

1.4

Ενθυλάκωση

Ορισμός

είναι ένας μηχανισμός προγραμματισμού που συνδέει τον κώδικα με τα δεδομένα που χειρίζεται και τα κρατά και τα δύο ασφαλή από εξωτερικές παρεμβολές και κακομεταχείριση.

η ενθυλάκωση στη πράξη

ο κώδικας και τα δεδομένα συνδέονται κατά τέτοιο τρόπο ώστε να δημιουργείται ένα "μαύρο κουτί". Μέσα σε αυτό βρίσκονται όλα τα απαραίτητα δεδομένα και ο απαραίτητος κώδικας. Έτσι δημιουργείται ένα αντικείμενο. Μέσα σε αυτό, ο κώδικας και τα δεδομένα μπορεί να είναι ιδιωτικά (private) -που σημαίνει ότι μπορούν να προσπελαστούν μόνο από άλλα μέρη του αντικειμένου- ή δημόσια (public) -μπορούν να προσπελαστούν και από τμήμα του προγράμματος που βρίσκεται έξω από το αντικείμενο.

Βασική μονάδα ενθυλάκωσης

η βασική μονάδα ενθυλάκωσης είναι η **κλάση (class)**:
μία κλάση ορίζει τι μορφή θα έχει ένα αντικείμενό της, δηλαδή καθορίζει τα δεδομένα και τον κώδικα που θα δρα επί αυτών των δεδομένων. Τόσο τα δεδομένα όσο και ο κώδικας ονομάζονται *μέλη* της κλάσης. Κατά αντιστοιχία, τα δεδομένα ονομάζονται *μεταβλητές μέλους* ενώ ο κώδικας *μέθοδοι μέλους*.

1.5

Πολυμορφισμός

Ορισμός

είναι το χαρακτηριστικό που επιτρέπει να το ίδιο όνομα μεθόδου να προκαλεί την εκτέλεση διαφορετικού κώδικα ανάλογα με τον τύπο του αντικειμένου στο οποίο καλείται.

πλεονεκτήματα

- μείωση πολυπλοκότητας κώδικα

1.6

Κληρονομικότητα

Ορισμός

είναι η διαδικασία κατά την οποία ένα αντικείμενο μπορεί να πάρει τις ιδιότητες ενός άλλου αντικειμένου.

πλεονεκτήματα

- ιεραρχική δομή κλάσεων
- κάθε αντικείμενο πρέπει να ορίζει μόνο εκείνες τις ιδιότητες που το κάνουν μοναδικό μέσα στη κλάση του και όχι αυτές που κληρονομεί

1.7

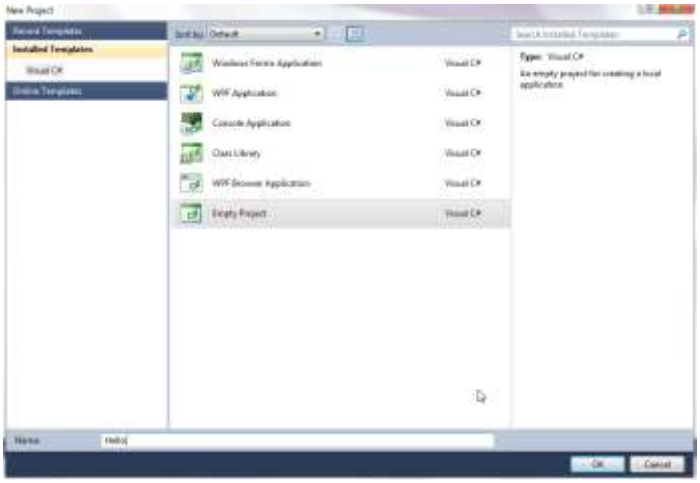
Δημιουργία του πρώτου προγράμματος

Μεταγλωττιστής C#

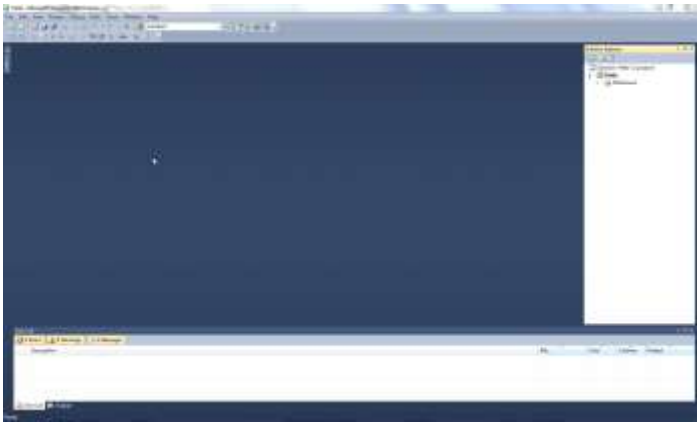
για να δημιουργήσετε , να μεταγλωττίσετε και να εκτελέσετε οποιοδήποτε πρόγραμμα C# θα χρειαστείτε τη ***Microsoft Visual C# 2010 Express Edition*** (δείτε και **Εισαγωγή.2**) αν έχετε λειτουργικό Microsoft Windows (Xp,Vista, 7.0) ή ***MonoDevelop*** αν έχετε λειτουργικό MacOS ή Linux.

Χρήση του περιβάλλοντος *Microsoft Visual C# 2010 Express Edition*

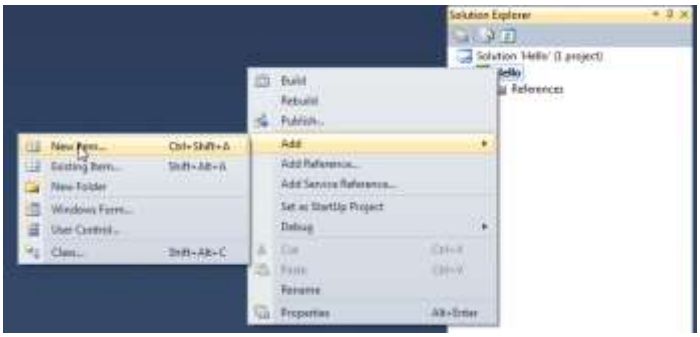
1. αρχικά πρέπει να δημιουργήσουμε ένα νέο, κενό έργο επιλέγοντας File | New Project:



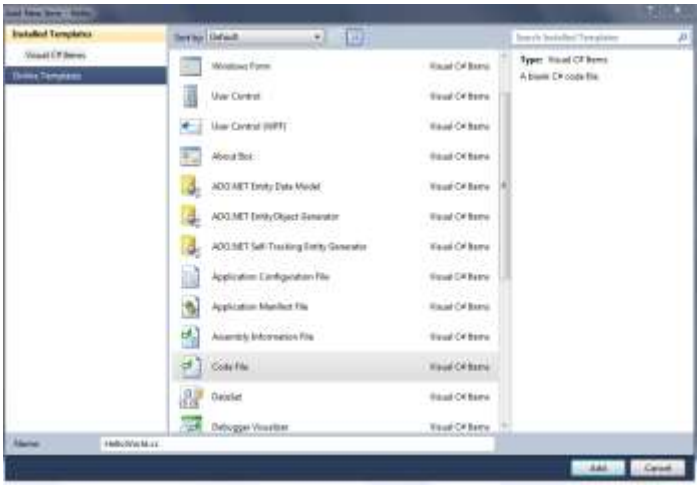
με το όνομα Hello. Πατάμε OK και βρισκόμαστε στο εξής σημείο:



Στο δεξί μέρος βλέπουμε το παράθυρο του Solution Explorer που μας δείχνει το project που δημιουργήσαμε και τι υπάρχει μέσα σε αυτό. Στη συνέχεια, για να προσθέσουμε ένα αρχείο πηγαίου κώδικα σε αυτό κάνουμε δεξί κλικ πάνω στο Hello μέσα στον Solution Explorer και επιλέγουμε Add | New Item... :



και καταλήγουμε στο παρακάτω παράθυρο διαλόγου:



στο οποίο επιλέγουμε Code File , για το οποίο δίνουμε όνομα HelloWorld.cs και πατάμε Add. Έτσι, η τελική μας εικόνα είναι η ακόλουθη:

Στο παράθυρο HelloWorld.cs γράφουμε το πρώτο μας πρόγραμμα.

Το πρώτο μας πρόγραμμα

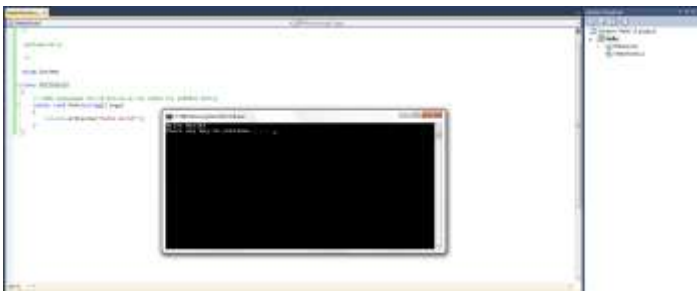
```
/*
 HelloWorld.cs
 */

using System;

class HelloWorld
{
    // Κάθε πρόγραμμα στη C# ξεκινά με την κλήση της μεθόδου Main()
    static void Main(string[] args)
    {
        Console.WriteLine("Hello World!");
        Console.ReadKey();
```

```
    }
}
```

Μεταγλώττιση του προγράμματος

Επιλέγουμε **Debug | Build Solution ή F6**

Εκτέλεση προγράμματος



Επιλέγουμε **Debug | Start Without Debugging ή Ctrl+F5**

οπότε θα πρέπει να δούμε την παρακάτω εικόνα:

1.8

Ανάλυση του πρώτου προγράμματος

**/*
HelloWorld.cs
*/
και
// Κάθε πρόγραμμα στη C# ξεκινά με την κλήση της μεθόδου Main()**

οι πρώτες 3 γραμμές κώδικα (με χρώμα πράσινο) αποτελούν σχόλια του προγράμματος. Η C# υποστηρίζει **2 είδη σχολίων**: α) το *σχόλιο παραγράφου* , που αρχίζουν με τους χαρακτήρες **/*** και τελειώνουν με τους ***/** και β) το *σχόλιο γραμμής* που αρχίζει με τους χαρακτήρες **//** .

using System;

δηλώνει ότι το πρόγραμμα χρησιμοποιεί τον **χώρο ονομάτων System** (χώρος ονομάτων= δηλωτική περιοχή), που σχετίζεται με τη βιβλιοθήκη κλάσεων του .NET Framework, η οποία είναι η βιβλιοθήκη που χρησιμοποιείται από την C#. Το **using** είναι λέξη-κλειδί που εισάγει το System στο πρόγραμμα. Κάθε εντολή τερματίζει με ελληνικό ερωτηματικό (;) .
Κάθε όνομα χώρου ονομάτων στη C# οφείλει να έχει **κεφαλαίο** το αρχικό γράμμα κάθε λέξης που περιέχει.

class HelloWorld

η βασική μονάδα ενθυλάκωσης είναι η **κλάση**. Έτσι, εδώ ορίζουμε την κλάση **HelloWorld**, της οποίας ο ορισμός αρχίζει με το αριστερό άγκιστρο **{** και τελειώνει με το δεξί άγκιστρο **}**.
Κάθε όνομα κλάσης στη C# οφείλει να έχει **κεφαλαίο** το αρχικό γράμμα κάθε λέξης που περιέχει.

static void Main(string[] args)

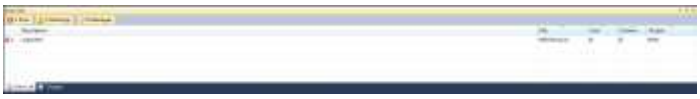
η γραμμή αυτή ορίζει τη **μέθοδο (συνάρτηση) Main()**. Από εδώ ξεκινά η εκτέλεση του προγράμματός μας. Μία μέθοδος που τροποποιείται από την **static** μπορεί να κληθεί *πριν* δημιουργηθεί ένα αντικείμενο της κλάσης της, κάτι το οποίο είναι απαραίτητο αφού η Main() καλείται κατά την εκκίνηση του προγράμματος. Η λέξη-κλειδί void δηλώνει ότι η συγκεκριμένη μέθοδος δεν επιστρέφει μία τιμή (μία μέθοδος μπορεί και να επιστρέφει τιμή μετά τη λειτουργία της). Τέλος, το **string[] args** αποτελεί όρισμα της Main() και συγκεκριμένα είναι ένας πίνακας που περιέχει αλφαριθμητικά και ονομάζεται args (περιέχει τα δεδομένα που διοχετεύονται στη βασική μέθοδο εξωτερικά).

Κάθε όνομα μεθόδου στη C# οφείλει να έχει **κεφαλαίο** το αρχικό γράμμα κάθε λέξης που περιέχει.

Console.WriteLine("Hello World!");	καλούμε την κλάση Console, η οποία ανήκει στον χώρο ονομάτων System και περιέχει μεθόδους σχετικά με την είσοδο και έξοδο στην κονσόλα. Από όλες αυτές τις μεθόδους επιλέγουμε την μέθοδο WriteLine() και της διοχετεύουμε ένα αλφαριθμητικό για να το εμφανίσει (διοχέτευση=όρισμα σε μέθοδο). αποτέλεσμα εκτέλεσης μεθόδου: εμφάνιση του μηνύματος Hello World! στην κονσόλα και αλλαγή γραμμής του κέρσορα.
Console.ReadKey();	Η μέθοδος Readkey() απλά κρατά τα αποτελέσματα του προγράμματος στην όθόνη μέχρι εμείς να πατήσουμε κάτι στο πληκτρολόγιο (οτιδήποτε).

1.9

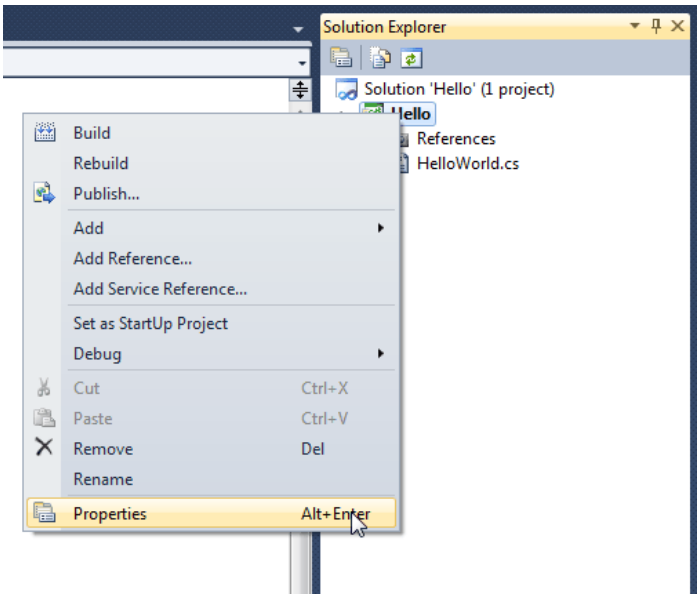
Έλεγχος συντακτικών σφαλμάτων

τι είναι συντακτικό λάθος;	λάθη που αφορούν τη συντακτική δομή της γλώσσας.
παράδειγμα	εάν πχ κατά λάθος παραλείψουμε ένα ερωτηματικό στο τέλος κλήσης της μεθόδου WriteLine() και πατήσουμε F6, θα πάρουμε τα εξής λάθη από τον μεταγλωττιστή:  στο κάτω μέρος της Visual C#.
τι κάνουμε σε αυτή τη περίπτωση;	αρχίζοντας από το *πρώτο λάθος*, προσπαθούμε να κατανοήσουμε το μήνυμα λάθους, πάμε στην αντίστοιχη γραμμή, το διορθώνουμε και πατάμε εκ νέου F6. Επαναλαμβάνουμε τη διαδικασία μέχρι να μην έχουμε κανένα συντακτικό λάθος.

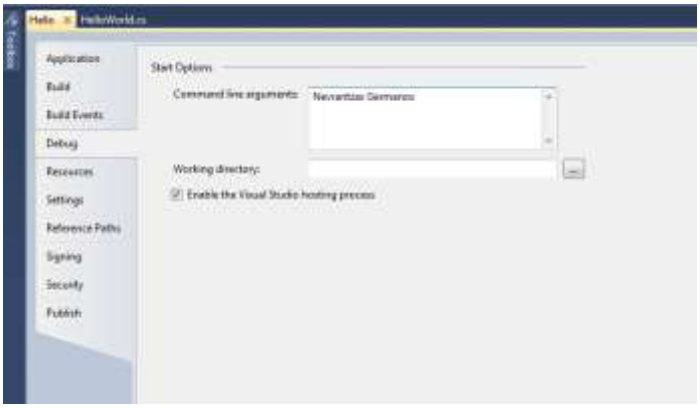
1.10

Παράμετροι στη Main()

ορισμός	είδαμε στο προηγούμενο παράδειγμα του HelloWorld.cs ότι μπορούμε να περάσουμε ως όρισμα στη βασική συνάρτηση έναν *πίνακα αλφαριθμητικών* με το όνομα **args**. Η δήλωση του πίνακα είναι η εξής: **string [] args** Ο πίνακας αυτός διοχετεύει στο πρόγραμμα (άρα αυτό μπορεί να τα χρησιμοποιήσει στη συνέχεια) ότι πληροφορία του δώσει ο χρήστης με *εξωτερικό τρόπο.*
διοχεύτεση ορισμάτων με εξωτερικό τρόπο	για να περάσουμε ορίσματα μέσω της μεθόδου Main() κάνουμε δεξί κλικ στο όνομα του project μέσα στο Solution Explorer και επιλέγουμε *Properties*:



στη συνέχεια πάμε στη καρτέλα *Debug* και στο πλαίσιο *Command line arguments* πληκτρολογούμε όλα τα ορίσματα που θέλουμε να περάσουμε στο πρόγραμμά μας (χωρισμένα με κενά):



και κλείνουμε το παράθυρο των ιδιοτήτων. Με αυτό τον τρόπο, όταν θα τρέξουμε το πρόγραμμα, θα έχουμε αποθηκεύσει στον πίνακα args και συγκεκριμένα στη θέση 0 το Nevrantzas , ενώ στη θέση 1 το Germanos.

τροποποίηση HelloWorld

έστω ότι θέλουμε να μας δώσει ο χρήστης με εξωτερικό τρόπο τα στοιχεία του , τα οποία στη συνέχεια θέλουμε να εκτυπώσουμε στην κονσόλα. Τροποποιήστε το αρχικό μας πρόγραμμα ως εξής:

```
/*
 HelloWorld.cs
 */

using System;

class HelloWorld
{
    // Κάθε πρόγραμμα στη C# ξεκινά με την κλήση της μεθόδου Main()
    static void Main(string[] args)
    {
        //εκτύπωση ονόματος
        Console.WriteLine(args[1]);
        //εκτύπωση επιθέτου
        Console.WriteLine(args[0]);
    }
}
```

Σε αυτή τη περίπτωση η μέθοδος WriteLine() έχει ως όρισμα μία μεταβλητή, οπότε δεν χρησιμοποιούμε τους χαρακτήρες " ".

1.11

Παράδειγμα

εκφώνηση
όνομα project:
Welcome
όνομα πηγαίου αρχείου:
WelcomeToCSharp.cs

Να γίνει πρόγραμμα στο οποίο ο φοιτητής θα δίνει το επίθετό του, το όνομά του και τον ΑΜ του και το πρόγραμμα θα εμφανίζει τα στοιχεία του με την εξής μορφή:

Επίθετο, Όνομα, ΑΜ: 1999

κώδικας

```
using System;

class Welcome
{
    static void Main(string[] args)
    {
        Console.WriteLine(args[1] + ", " + args[0] + ", AM:" +
args[2]);
        Console.ReadKey();
    }
}
```

1.12

Ερωτήσεις κατανόησης

1 είναι η C# case sensitive;

2 αφαιρέστε τα εξωτερικά ορίσματα που εισάγατε στο project Welcome. Ποιο θα είναι το πρόβλημα αν προσπαθήσουμε να εκτελέσουμε το πρόγραμμα μας; Εκτελέστε το πρόγραμμά σας.

3 στον κώδικα

```
using System;

class Welcome
{
    static void Main(string args[])
    {
        Console.WriteLine(args[1] + ", " + args[0] + ", AM:" +
args[2]);
        Console.ReadKey();
    }
}
```

ποιο είναι το λάθος;

4 στον κώδικα

```
using System;

static void Main(string []args)
{
    Console.WriteLine(args[1] + ", " + args[0] + ", AM:" +
args[2]);
    Console.ReadKey();
}
```

ποιο είναι το λάθος;

5 στον κώδικα

```
using System;

class Welcome
{
    void Main(string[] args)
    {
        Console.WriteLine(args[1] + ", " + args[0] + ", AM:" +
args[2]);
        Console.ReadKey();
    }
}
```

ποιο είναι το λάθος;

2

Δεδομένα, τελεστές, είσοδος/έξοδος

Τι θα δούμε σε αυτό το μάθημα

1. βασικοί τύποι δεδομένων
2. ακέραιοι
3. κινητής υποδιαστολής
4. ο τύπος decimal
5. χαρακτήρες
6. bool
7. string
8. χρήση μεταβλητών
9. άρρητη δήλωση μεταβλητών
10. σταθερές
11. τελεστές
12. casting ασύμβατων τύπων
13. είσοδος προγράμματος
14. έξοδος προγράμματος
15. σταθερές ανάποδης καθέτου
16. ερωτήσεις κατανόησης

2.1

Βασικοί τύποι δεδομένων

γιατί είναι σημαντικοί οι τύποι δεδομένων

η C# χρησιμοποιεί ισχυρό έλεγχο τύπων, το οποίο σημαίνει ότι όλες οι πράξεις ελέγχονται ως προς τη συμβατότητα των τύπων. Εάν υπάρχουν μη έγκυρες πράξεις τότε *δεν μεταγλωττίζονται*. Κέρδος; φυσικά η *αξιοπιστία* των προγραμμάτων μας.

κατηγορίες ενσωματωμένων τύπων δεδομένων

- **τύποι τιμών (value types):**
  περιέχει μία πραγματική τιμή, πχ 100, 25.9
- **τύποι αναφοράς (reference types):**
  περιέχει μία αναφορά προς την τιμή, συνήθως *κλάση*.

πίνακας τύπων τιμών

bool	τιμές true/false
byte	μη προσημασμένος ακέραιος 8-bit
char	χαρακτήρας
decimal	αριθμητικός τύπος για οικονομικούς υπολογισμούς
double	κινητής υποδιαστολής διπλής ακρίβειας
float	κινητής υποδιαστολής μονής ακρίβειας
int	ακέραιος
long	μεγάλος ακέραιος
sbyte	προσημασμένος ακέραιος 8-bit
short	μικρός ακέραιος
uint	μη προσημασμένος ακέραιος
ulong	μη προσημασμένος μεγάλος ακέραιος
ushort	μη προσημασμένος μικρός ακέραιος

string	αλφαριθμητικό

2.2

Ακέραιοι

ακέραιοι και bits

byte	8	[0,255]
sbyte	8	[-128,127]
short	16	[-32768,32767]
ushort	16	[0,65535]
int	32	[-214748□648,2147483647]
uint	32	[0,4294967295]
long	64	[-9223372036854775808,92233720368547758077]
ulong	64	[0,18446744073709551615]

παράδειγμα ορθής χρήσης ακεραίων

```
using System;

class Akeraios
{
    static void Main()
    {
        byte x;
        int sum;

        sum = 0;

        for (x = 1; x <= 100; x++)
            sum = sum + x;

        Console.WriteLine("To athroisma einai:" + sum);

        Console.ReadKey();
    }
}
```

2.3

Κινητής υποδιαστολής

είδη τύπων

η C# υποστηρίζει 2 τύπους κινητής υποδιαστολής:

- **float**
  μονής ακρίβειας, **32** bits, διάστημα τιμών [1.5E-45, 3.4E+38]
  πχ float x=1.2f;

- **double**

διπλής ακρίβειας, **64** bits, διάστημα τιμών [5E-324, 1.7E+308]
πχ double x=1.2;

Ο τύπος που χρησιμοποιούμε περισσότερο είναι ο *double*, γιατί πολλές από τις μαθηματικές συναρτήσεις της βιβλιοθήκης κλάσεων της C# χρησιμοποιούν τέτοιου τύπου τιμές.

παράδειγμα

```
using System;

class TyposDouble
{
    static void Main()
    {
        double x, y, z;

        x = 3;
        y = 4;

        z = Math.Sqrt(x * x + y * y);

        Console.WriteLine("Ypoteinousa:" + z);

        Console.ReadKey();
    }
}
```

2.4

Ο τύπος Decimal

bits και περιοχή τιμών

ο τύπος decimal χρησιμοποιεί **128 bits** και μπορεί να παραστήσει τιμές στο διάστημα **[1E-28, 7.9E+28]**. Μπορεί να παραστήσει με ακρίβεια μέχρι *28 δεκαδικά ψηφία* και είναι ιδιαίτερα χρήσιμος όταν έχουμε υπολογισμούς χρηματικών ποσών.

Σημείωση: ο τύπος decimal δεν υπάρχει στις υπόλοιπες αντικειμενοστρεφής γλώσσες προγραμματισμού.

παράδειγμα

```
using System;

class TyposDecimal
{
    static void Main()
    {
        decimal ypoloipo;
        decimal pososto;

        ypoloipo = 1000.0m;
        pososto = 0.09m;

        ypoloipo = ypoloipo * pososto + ypoloipo;

        Console.WriteLine("Neo ypoloipo:"+ypoloipo);

        Console.ReadKey();
    }
}
```

2.5

Χαρακτήρες

bits και περιοχή τιμών

η C# για την αναπαράσταση των χαρακτήρων χρησιμοποιεί ***Unicode***, άρα ο τύπος char είναι ένας ***μη προσημασμένος*** τύπος ***16 bit*** με περιοχή τιμών **[0,65535]**.

δήλωση μεταβλητής χαρακτήρα

```
char x='a';
```

2.6

Bool

ορισμός

αναπαριστά τιμές **αληθούς/ψευδούς** χρησιμοποιώντας τις δεσμευμένες λέξεις **true** και **false** αντίστοιχα.

παράδειγμα

```
using System;

class TyposBool
{
    static void Main()
    {
        bool b;

        b = false;

        if (b)
            Console.WriteLine("Executed");
        else
            Console.WriteLine("NotExecuted");

        Console.WriteLine("90>20 einai " + (90 > 20));

        Console.ReadKey();
    }
}
```

2.7

String

ορισμός

είναι ένα *σύνολο χαρακτήρων* που περικλείεται σε διπλές αποστρόφους.

δήλωση string

```
string x="tei larisas";
```

κυριολεκτικό αλφαριθμητικό (verbatim string)

περιέχει στην αρχή το σύμβολο **@** και ότι βρίσκεται μέσα στις αποστρόφους γίνεται αποδεκτό *χωρίς τροποοίηση από ειδικούς χαρακτήρες*.

πχ

```
string monopati=@"c:\hello.cs";
```

2.8

Χρήση μεταβλητών

αρχικοποίηση μεταβλητών

οι μεταβλητές μπορούν να αρχικοποιηθούν με 3 τρόπους:

1. **κατά τη δήλωση:**
   int x=3;
   float y=2.1f;

2. με **ανάθεση τιμής:**
   int x;
   x=3;

3. με **δυναμική αρχικοποίηση** (χρήση έκφρασης):
   double radius=5,height=5;
   double volume;
   volume=3.1419*radius*radius*height;

2.9

Άρρητη δήλωση τύπου μεταβλητών

τι σημαίνει "άρρητη δήλωση"; αποφασίζει ο *μεταγλωττιστής* τον **τύπο της μεταβλητής** με βάση την *τιμή* που χρησιμοποιείται για την αρχικοποίησή της.

πως μπορούμε να την υλοποιήσουμε; χρησιμοποιούμε την λέξη-κλειδί **var** και πρέπει <u>οπωσδήποτε</u> να την *αρχικοποιήσουμε*.

παραδειγμα στη δήλωση

var balance = 1370.14;

var radius = 10;

η μεταβλητή pi είναι τύπου *double* και η μεταβλητή radius είναι τύπου *int*.

περιορισμός μπορούμε να δηλώσουμε <u>*μία και μόνο μία*</u> άρρητη μεταβλητή σε κάθε εντολή. Η ακόλουθη δήλωση είναι λάθος:

var count=10,max=20;

2.10

Σταθερές

ορισμός **δεν** αλλάζει η τιμή τους σε όλη τη διάρκεια ζωής τους.

δήλωση σταθεράς **const** int x=3;

εμβέλεια σταθεράς

- **global**: όταν δηλωθεί εκτός της Main() αλλά εντός της κλάσης που την περιέχει.
- **local**: όταν δηλωθεί εντός της Main().

2.11

Τελεστές

αριθμητικοί τελεστές

+	πρόσθεση
-	αφαίρεση
*	πολλαπλασιασμός
/	διαίρεση

%	υπόλοιπο (ισχύει για μεταβλητές τύπου int και double)
++	προσαύξηση κατά 1
--	μείωση κατά 1

σχεσιακοί τελεστές

==	ίσο
!=	όχι ίσο
>	μεγαλύτερο
<	μικρότερο
<=	μικρότερο ή ίσο
>=	μεγαλύτερο ή ίσο

λογικοί τελεστές

&	AND
|	OR
^	XOR
||	βραχυκυκλωμένο OR (ο δεύτερος τελεστέος αποτιμάται όταν είναι απαραίτητο)
&&	βραχυκυκλωμένο AND (ο δεύτερος τελεστέος αποτιμάται όταν είναι απαραίτητο)
!	NOT

παράδειγμα βραχυκυκλωμένου τελεστή

```
using System;

class Vraxukuklwma
{
    static void Main()
    {
        int n, d;

        n = 10;
        d = 2;


        if (d != 0 && (n % d) == 0)
            Console.WriteLine(d + " is a factor of " + n);

        d = 0;

        if (d != 0 && (n % d) == 0)
            Console.WriteLine(d + " is a factor of " + n);

        Console.ReadKey();

        }
}
```

αν στο τελευταίο if είχαμε ως τελεστή το & τότε θα είχαμε διαίρεση με το 0!

2.12

Casting ασύμβατων τύπων

τι ακριβώς είναι το casting τύπων;

είναι μία εντολή προς τον μεταγλωττιστή να *μετατρέψει* μία έκφραση σ'ένα συγκεκριμένο τύπο, δηλαδή ζητά *ρητή μετατροπή τύπου*. Ο γενικός του τύπος είναι ο ακόλουθος:

(τύπος προορισμού) έκφραση

πχ

αν είχαμε δηλώσει 2 μεταβλητές με τον εξής τρόπο:

double x,y;

& θέλαμε το αποτέλεσμα της διαίρεσης x/y να είναι ακέραιο, τότε θα είχαμε το ακόλουθο casting:

(int)(x/y);

προσοχή: οι παρενθέσεις είναι *απαραίτητες*, διότι το αποτέλεσμα της έκφρασης (int)x/y; είναι τύπου double.

παράδειγμα

```
using System;

class Casting
{
    static void Main()
    {
        double x, y;
        byte b;
        int i;
        char ch;

        x = 10.0;
        y = 3.0;

        i = (int)(x / y);
        Console.WriteLine("Integer outcome of x / y: " + i);

        i = 100;
        b = (byte)i;
        Console.WriteLine("Value of b: " + b);

        i = 325;
        b = (byte)i;
        Console.WriteLine("Value of b: " + b);

        b = 88;
        ch = (char)b;

        Console.WriteLine("ch: " + ch);

        Console.ReadKey();
    }
}
```

2.13

Είσοδος προγράμματος

μέθοδος εισόδου πληροφορίας από το πληκτρολόγιο

1. ReadLine() :

ανήκει στην κλάση *Console*, διαβάζει από το πληκτρολόγιο & αλλάζει γραμμή.

2. Read();

ανήκει στην κλάση Console, διαβάζει έναν χαρακτήρα από το πληκτρολόγιο, τον μετατρέπει σε ακέραιο και το αποθηκεύει σε μία μεταβλητή τύπου int.

3. ReadKey();

περιμένει μέχρι να πατήσουμε έναν χαρακτήρα στο πληκτρολόγιο. αν της δώσουμε όρισμα true, τότε ο χαρακτήρας που πατάμε δεν εκτυπώνεται, αλλιώς εμφανίζεται στην οθόνη.

παραδείγματα

```
string myString=Console.ReadLine();
```

Μετατροπή εισόδου

(χρήση της μεθόδου ReadLine())

- ότι εισάγουμε από το πληκτρολόγιο θεωρείται μία *ακολουθία χαρακτήρων* (**string**)

- όταν εισάγουμε αριθμούς (οποιουδήποτε τύπου) θα πρέπει να τους ***μετατρέψουμε*** (από string) στον κατάλληλο τύπο δεδομένων για να μπορούμε στη συνέχεια να εκτελούμε πράξεις.

Μέθοδοι μετατροπής αλφαριθμητικού σε αριθμό (int, float, double)

1. Int32.Parse()

μετατροπή από string σε ακέραιο.

2. Double.Parse()

μετατροπή από string σε double.

3. Single.Parse()

μετατροπή από string σε float μονής ακρίβειας.

παράδειγμα

```
string myString = "1023";

int myInt = Int32.Parse( myString );
```

2.14

Έξοδος προγράμματος

Μέθοδοι εξόδου

1. WriteLine() :

ανήκει στην κλάση Console, γράφει στην οθόνη ότι βρίσκεται μέσα στα " " & αλλάζει γραμμή.

2.Write() :

ανήκει στην κλάση Console & γράφει στην οθόνη ότι βρίσκεται μέσα στα " ".

τρόποι εφαρμογής

- **με συνέννωση (+):**
  Console.WriteLine("Onoma: "+onoma+",Epitheto: "+epitheto);

- **με αλφαριθμητικό σταθερών και μεταβλητών:**
  Console.WriteLine("Onoma {0},Epitheto: {1}",onoma,epitheto);

μέσα στα άγκυστρα μπορούμε να τοποθετήσουμε χαρακτήρες φορμαρίσματος της εξόδου, πχ:

Console.WriteLine(" To apotelesma ths praxis 10/3 einai: {0,5:#.##}",10.0/3.0);

θα εμφανίσει:

To apotelesma ths praxis 10/3 einai: 3,33

σημείωση: ο αριθμός μετά το κόμμα δηλώνει τον *αριθμό των θέσεων* που θα χρησιμοποιηθούν για να εμφανιστεί το αποτέλεσμα (συμπεριλαμβάνονται η υποδιαστολή και τα δεκαδικά), ενώ μετά την άνω κάτω τελεία δηλώνουμε την *μορφή* που θα έχει ο αριθμός.

2.15

Σταθερές ανάποδης καθέτου

Ορισμός είναι οι χαρακτήρες που εισάγουμε μέσα σε μία ακολουθία χαρακτήρων & έχουν ειδική σημασία.

Είδη

1. \n : αλλαγή γραμμής
2. \r : αρχή της ίδιας γραμμής
3. \t : tab
4. \" : "
5. \' : '
6. \\ : \

2.16

Ερωτήσεις κατανόησης

1 Τι είναι ο τύπος χαρακτήρων (char) στη C# και σε τι διαφέρει από τις υπόλοιπες γλώσσες προγραμματισμού; Γιατί αυτή η διαφορά;

2 Μπορεί μία τιμή bool να έχει όποια τιμή θέλετε. Στη c μία μη μηδενική τιμή είναι true. Σωστό ή λάθος;

3 Τι λάθος/η υπάρχει/ουν στον παρακάτω κώδικα;

```
using System;

class Program
{
    static void Main(string[] args)
    {
        for (int i = 0; i < 10; i++)
        {
            int sum;

            sum = sum + i;
        }
        Console.WriteLine("Sum is: " , sum);
    }
}
```

4 Ποια η τιμή της μεταβλητής y στις ακόλουθες εκφράσεις, αν υποθέσουμε ότι η αρχική τιμή της μεταβλητής x είναι 10:

Y = ++x + 25;

Y = x+++ 25;

Y = (x++) + 25;

5 Είναι σωστός ο παρακάτω κώδικας; Αν ναι, πείτε τι θα εμφανίσει, αλλιώς βρείτε τα λάθη του:

```
using System;

class Program
{
    static void Main()
    {
        string onoma = Console.ReadLine();
        Console.WriteLine("To onoma sou einai " + onoma);

        Console.ReadKey(true);
    }
}
```

6 Είναι σωστός ο παρακάτω κώδικας; Αν ναι, πείτε τι θα εμφανίσει, αλλιώς βρείτε τα λάθη του:

```
using System;

const int N=10;

class Program
{
    static void main()
    {
        var x = 10.0;
        byte c;

        for (c = 0; c <= N; ++c)
            x = x + c;

        Console.WriteLine("x={0,10:###.00}",x);

        Console.ReadKey(true);
    }
}
```

7 Είναι σωστός ο παρακάτω κώδικας; Αν ναι, πείτε τι θα εμφανίσει, αλλιώς βρείτε τα λάθη του:

```
using System;

class Program
{
```

```
    static void Main()
    {
        double a, b;

        a = Double.Parse(Console.ReadLine());
        b = Double.Parse(Console.ReadLine());

        byte x = 1;

        x = (byte)(a + b);

        Console.WriteLine("x = " + x);

        Console.ReadKey(true);
    }
}
```

8 Ποια η τελική τιμή του x αν ο χρήστης δώσει ως είσοδο:

300

400

στο παρκάτω πρόγραμμα;

```
using System;

class Program
{

    static void Main()
    {
        int a, b;

        a = Int32.Parse(Console.ReadLine());
        b = Int32.Parse(Console.ReadLine());

        byte x = 1;

        x = (byte)(a + b);

        Console.WriteLine("x="+x);

        Console.ReadKey(true);
    }
}
```

2.17

Παραδείγματα

Να γίνει πρόγραμμα το οποίο

1. ο χρήστης να δίνει το όνομά του και το επίθετό του

2. το πρόγραμμα θα εκτυπώνει τα πλήρη στοιχεία του χρήστη

```
using System;
class Lab1Ex1
{
    public static void Main()
    {
        Console.WriteLine("Dwse to onoma sou");
        string toOnomaMou = Console.ReadLine();
        Console.WriteLine("Dwse to epitheto sou");
        string toEpithetoMou = Console.ReadLine();
        Console.WriteLine("Onoma:"+toOnomaMou+",epitheto:" +
toEpithetoMou);

Console.WriteLine("Onoma:{0},epitheto:{1}",toOnomaMou,toEpithetoMou;
        Console.ReadKey();
    }
}
```

Να γίνει πρόγραμμα το οποίο

```
using System;

class Lab1Ex2
```

1. ο χρήστης να δίνει 2 ακεραίους

2. το πρόγραμμα θα υπολογίζει & θα εκτυπώνει το άθροισμα τους

```
{
    public static void Main()
    {
        string strAr1, strAr2;

        Console.WriteLine("Dwse ton prwto akeraio:");
        strAr1 = Console.ReadLine();
        Console.WriteLine("Dwse ton deutero akeraio:");
        strAr2 = Console.ReadLine();

        int ar1, ar2;
        ar1 = Int32.Parse(strAr1);
        ar2 = Int32.Parse(strAr2);

        int apotelesma;
        apotelesma = ar1 + ar2;

        Console.WriteLine("{0}+{1}={2}", ar1, ar2, apotelesma);
        Console.ReadKey();
    }
}
```

Να γίνει πρόγραμμα το οποίο

1. ο χρήστης να δίνει ένα ποσό σε δολλάρια

2. το πρόγραμμα θα μετατρέπει τα δολλάρια σε ευρώ σύμφωνα με τον τύπο:

ευρώ = 0.70 * δολλάρια

```
using System;
class Lab1Ex3
{
    public static void Main()
    {
        const double ISOTIMIA = 0.75;

        string strDollars;
        Console.WriteLine("Dwse ta dollaria:");

        strDollars = Console.ReadLine();

        double dollars;
        dollars = Double.Parse(strDollars);

        Double euros;
        euros = ISOTIMIA * dollars;

        Console.WriteLine("{0} dollars = {1} euros", dollars,
euros);
        Console.ReadKey();
    }
}
```

Ασκήσεις Πράξης #2

Άσκηση 1

Να γίνει πρόγραμμα το οποίο

1. **ο χρήστης να δίνει 2 ακεραίους**
2. **το πρόγραμμα θα υπολογίζει & θα εκτυπώνει το άθροισμα τους**

Άσκηση 2

Να γίνει πρόγραμμα το οποίο

1. **ο χρήστης να δίνει ένα ποσό σε δολλάρια**
2. **το πρόγραμμα θα μετατρέπει τα δολλάρια σε ευρώ σύμφωνα με τον τύπο: ευρώ = 0.70 * δολλάρια**

Άσκηση 3

Να γίνει πρόγραμμα το οποίο

1. **ο χρήστης να δίνει το όνομά του και το επίθετό του**
2. **το πρόγραμμα θα εκτυπώνει τα πλήρη στοιχεία του χρήστη**

3

Δομές, ελέγχου & επανάληψης

Τι θα δούμε σε αυτό το μάθημα

1. δομές ελέγχου
    1. η δομή if
    2. η δομή switch
2. δομές επανάληψης
    1. η δομή while
    2. η δομή do…while
    3. η δομή for
3. break και continue
4. ερωτήσεις κατανόησης
5. ασκήσεις

3.1

Δομές ελέγχου

3.1.1 η δομή if

Η *σύνταξη* της δομής είναι η εξής:

```
if (συνθήκη ελέγχου)
{
ομάδα εντολών 1
}
else
{
ομάδα εντολών 2
}
```

Η *λειτουργία* της δομής είναι η εξής:
Πρώτα ελέγχεται η *συνθήκη*. Αν είναι **true** τότε εκτελείται η ομάδα εντολών 1 αλλιώς εκτελείται η ομάδα εντολών 2.
Η συνθήκη ελέγχου είναι συνήθως μία ισότητα ή ανισότητα. Η ισότητα εκφράζεται με το σύμβολο == και όχι με το απλό = (το οποίο σημαίνει "ανάθεση").

Σημαντικές επισημάνσεις:
Η δομή μπορεί να γίνει πιο απλή παραλείποντας το else ή ακόμα πιο σύνθετη εμπλουτίζοντας το else με εμφωλευμένο if...else... .

παράδειγμα

Παιχνίδι μαντέματος

```
using System;

class Guess2
{
   static void Main()
   {
      char ch, answer = 'K';

      Console.WriteLine("I'm thinking of a letter between A and Z.");
      Console.Write("Can you guess it: ");

      ch = (char)Console.Read(); // get the user's guess

      if (ch == answer)
         Console.WriteLine("** Right **");
      else
         Console.WriteLine("...Sorry, you're wrong.");

      Console.ReadKey(true);
   }
}
```

3.1.2 η δομή switch

Η *σύνταξη* της δομής είναι η εξής:

```
switch (επιλογέας)
{
        case τιμή1: εντολές;break;
        case τιμή2: εντολές;break;
        ...
```

```
        case τιμήΝ: εντολές;break;
        default:    εντολές;break;
}
```

Η *λειτουργία* της δομής είναι η εξής:
Υπολογίζεται η τιμή του **επιλογέα** στην αρχή της δομής. Ανάλογα με την τιμή του εκτελούνται οι εντολές του αντίστοιχου case & στη συνέχεια το break (το οποίο & τερματίζει το switch). Αν δεν ταιριάζει κανένα case με την τιμή του επιλογέα τότε εκτελούνται οι εντολές του default.

Σημαντικές επισημάνσεις:

- Ο επιλογέας πρέπει απαραίτητα να είναι μεταβλητή ή παράσταση βαθμωτού τύπου, δηλαδή να παίρνει διακριτές τιμές (**μεταβλητή τύπου: int, short, byte, char ή string**).
- είναι σφάλμα οι εντολές μίας case να συνεχίζονται στην επόμενη case (**κανόνας απαγόρευσης συνέχισης**), οπότε το break σε κάθε case είναι υποχρεωτικό. Το ίδιο ισχύει και για το default, αν και βρίσκεται στο τέλος της δομής switch.
- Μπορεί μία ομάδα από case να περιέχουν τις ίδιες εντολές προς εκτέλεση. Σε αυτή την περίπτωση αυτές γράφονται μία φορά στο τελευταίο κατά σειρά case ως εξής:

```
switch (a)
{
 case 'a':
 case 'A': Console.WriteLine("...");break;
 ...
 default: ...;break;
}
```

παράδειγμα

Να γίνει πρόγραμμα το οποίο:

να αναπαριστά την λειτουργία της **αριθμομηχανής τσέπης** ως εξής:

- ο χρήστης να δίνει την πράξη με εξωτερικό τρόπο με την ακόλουθη μορφή:
  2 + 3
- Ανάλογα με την πράξη να εμφανίζει το **αποτέλεσμα**.
- Σε περίπτωση **μή ορθής εισόδου** να βγάζει «Error»
- Σε περίπτωση **διαίρεσης με το 0** πάλι να εμφανίζει ανάλογο μήνυμα

```
using System;

class Calculator
{
  static void Main(string []args)
  {
    double a, b, apotelesma = 0.0;
    bool flag = false;

    a = Double.Parse(args[0]);
    b = Double.Parse(args[2]);

    switch (args[1])
    {
      case "+":
        apotelesma = a + b;
        break;
      case "-":
        apotelesma = a - b;
        break;
      case "*":
        apotelesma = a * b;
        break;
      case "/":
        if (b != 0.0)
          apotelesma = a / b;
        else
        {
          Console.WriteLine("Error");
          flag = true;
        }
        break;
      default:
        Console.WriteLine("Input Error!");
```

```
            flag = true;
            break;
        }
        if (!flag)
            Console.WriteLine("{0:.00}{1}{2:.00}={3:.00}", a, args[1], b, apotelesma);

        Console.ReadKey();
    }
}
```

3.2

Δομές επανάληψης

3.2.1 η δομή while

Η *σύνταξη* της δομής είναι η εξής:

while (συνθήκη)
{
…ομάδα εντολών…
}

Η *λειτουργία* της δομής είναι η εξής:
Αρχικά **ελέγχεται** η **συνθήκη** που βρίσκεται μέσα στην παρένθεση. Αν το αποτέλεσμα είναι **true** τότε **εκτελείται** η ομάδα εντολών μέσα στα άγκιστρα και στη συνέχεια ο έλεγχος **επανέρχεται** στην κορυφή του while για να γίνει εκ νέου έλεγχος της συνθήκης . Σε περίπτωση που η συνθήκη δεν ισχύει (false) η ομάδα εντολών **παρακάμπτεται**.

Σημαντικές επισημάνσεις:

- Ένα while μπορεί να **μην** εκτελεστεί καμία φορά κατά την εκτέλεση ενός **προγράμματος**.
- Οι μεταβλητές που συμμετέχουν στην συνθήκη θα πρέπει να **μεταβάλλονται** (η τιμή τους) μέσα στην ομάδα εντολών αλλιώς το while δεν θα τερματίσει ποτέ (**ατέρμων** βρόχος) !

παράδειγμα

Να γίνει πρόγραμμα το οποίο θα διαβάζει τις βαθμολογίες των μαθημάτων ενός φοιτητή (τέλος εισαγωγής με το -1), θα υπολογίζει & να εκτυπώνει τον **ΜΟ των μαθημάτων**.

σημείωση:

Τα στοιχεία του φοιτητή (Επίθετο, Όνομ,ΑΜ) να εισάγονται με *εξωτερικό τρόπο* στο πρόγραμμα & να εκτυπώνονται μαζί με τον ΜΟ.

```
using System;

class Program
{
    static void Main(string[] args)
    {
        int sum,
            i,
            vathmos = 0;
        float mesosOros;

        sum = 0;
        i = 0;

        while (vathmos != -1)
        {
            Console.WriteLine("Dwse ton {0} vathmo tou foiththt:", i + 1);

            vathmos = Int32.Parse(Console.ReadLine());

            if (vathmos >= 0 && vathmos <= 10)
            {
                sum = sum + vathmos;
                i = i + 1;
            }
        }

        mesosOros = (float)sum / i;
```

```
        Console.WriteLine("\nO foiththts {0}, {1} me AM {2} exei MO {3:00.00}", args[0], args[1], args[2], mesosOros);

        Console.ReadKey();
    }
}
```

3.2.2 Η *σύνταξη* της δομής είναι η εξής:

η δομή do...while

```
do
{
...ομάδα εντολών...

} while (συνθήκη) ;
```

Η *λειτουργία* της δομής είναι η εξής:
Αρχικά εκτελείται η ομάδα **εντολών** που βρίσκεται ανάμεσα στα άγκιστρα & στο **τέλος ελέγχεται** η συνθήκη που βρίσκεται μέσα στην παρένθεση. Αν το αποτέλεσμα είναι **true** τότε εκτελείται **εκ νέου** η ομάδα εντολών μέσα στα άγκιστρα. Σε διαφορετική περίπτωση το πρόγραμμα συνεχίζει στις **επόμενες** εντολές.

Σημαντικές επισημάνσεις:

- Ένα do... while θα εκτελεστεί **τουλάχιστον μία φορά**, σε αντίθεση με το απλό while.
- Οι μεταβλητές που συμμετέχουν στην συνθήκη θα πρέπει & εδώ να **μεταβάλλονται** (η τιμή τους) μέσα στην ομάδα εντολών αλλιώς το while δεν θα τερματίσει ποτέ (**ατέρμων** βρόχος) !

παράδειγμα

Να γίνει πρόγραμμα το οποίο να υπολογίζει το **άθροισμα**:

S= 1+3+5+...+N

όταν ο χρήστης **δίνει το N**

περιορισμοί για το N: θετικός & περιττός ακέραιος αριθμός.

```
using System;

class Athroisma
{
   const int ARXH = 1, VHMA = 2;

   static void Main()
   {
      int N;

      do
      {
         Console.WriteLine("Dwse to N:");
         N = Int32.Parse(Console.ReadLine());
      } while (N < 1 || N % 2 == 0);

      int S = 0, i = ARXH;

      do
      {
         S += i;
         i += VHMA;
      } while (i <= N);

      Console.WriteLine("S= " + S);
      Console.ReadKey();
   }
}
```

3.2.3 Η *σύνταξη* της δομής είναι η εξής:

η δομή for

```
for (αρχική τιμή;έλεγχος;ανανέωση)
{
...
}
```

Η *λειτουργία* της δομής είναι η εξής:

Αρχικά εκτελείται η ***αρχική τιμή***. Στην ουσία πρόκειται για μία **αρχικοποίηση ενός μετρητή** πριν την εκτέλεση του κώδικα που βρίσκεται ανάμεσα στα άγκιστρα. Στη συνέχεια ελέγχεται αν είναι αληθής η ***συνθήκη ελέγχου.*** Αν το αποτέλεσμα είναι **true** τότε εκτελούνται οι εντολές που βρίσκονται ανάμεσα στα άγκιστρα. Στο τέλος εκτελείται η ***έκφραση ανανέωσης*** & στη συνέχεια ελέγχεται εκ νέου η ***συνθήκη ελέγχου*** & η όλη διαδικασία επαναλαμβάνεται μέχρι να **αποτύχει** η συνθήκη ελέγχου.

Χρήση της for

1. Μπορεί να εκτελείται με **μείωση** του μετρητή:

```
using System;

class ParadeigmaFor
{
   static void Main(string[] args)
   {
      int i;

      for (i = 10; i > 0; i--)
         Console.WriteLine(i);

      Console.ReadKey();
   }
}
```

2. Μπορεί να εκτελείται με αύξηση ή μείωση του μετρητή κατά **οποιονδήποτε ακέραιο αριθμό**:

```
using System;

class ParadeigmaFor
{
   static void Main(string[] args)
   {
      int i,j=0;

      for (i = 2; i <= 100; i+=2)
         Console.WriteLine(++j+":"+i);

      Console.ReadKey();
   }
}
```

3. Μπορεί να εκτελείται με **γεωμετρική αύξηση ή μείωση** του μετρητή:

```
using System;

class ParadeigmaFor
{
   static void Main(string[] args)
   {
      double i;
      int j=0;

      for (i = 1000.0; i <= 3100.0; i*=1.05)
         Console.WriteLine(++j+":"+i);

      Console.ReadKey();
   }
}
```

4. Μπορεί να εκτελείται με **χρήση χαρακτήρων** αντί ακεραίων:

```
using System;
```

```
class ParadeigmaFor
{
   static void Main(string[] args)
   {
      char i;
      int j=0;

      for (i = 'A'; i <= 'Z'; i++)
         Console.WriteLine(++j+":"+i);

      Console.ReadKey();
   }
}
```

5. Κάποιες εκφράσεις μπορούν να **παραλείπονται**:

```
using System;

class ParadeigmaFor
{
   static void Main(string[] args)
   {
      char i='A';
      int j=0;

      for (; i <= 'Z'; )
         Console.WriteLine(++j+":"+i++);

      Console.ReadKey();
   }
}
```

6. Μπορούν να χρησιμοποιηθούν πολλαπλές μεταβλητές ελέγχου:

```
using System;

class Comma
{
  static void Main()
   {
            int i,j;

            for(i=0,j=10;i<j;i++,j--)
               Console.WriteLine("i and j:"+i+" "+j);

             Console.ReadKey();
    }
}
```

παράδειγμα

Να γίνει πρόγραμμα το οποίο να μετράει του χαρακτήρες που δίνει ο χρήστης από το πληκτρολόγιο

```
using System;

class ForTest
{
   static void Main()
   {
      int numChars = 0;

      Console.WriteLine("Grapse mia ekfrash:");

      for (int i = 0; (char)Console.Read() != '\n'; i++)
        numChars++ ;

      Console.WriteLine("Edwses " + numChars +" xarakthra/es");

      Console.ReadKey();
   }
```

```
}
```

3.3

break και continue

break

η *λειτουργία* της break είναι η εξής: όταν εκτελεστεί **τερματίζει** σε εκείνο το σημείο τον βρόχο , παρακάμπτοντας τον υπόλοιπο κώδικα μέσα στο σώμα του βρόχου και τον έλεγχο της συνθήκης. Ο έλεγχος περνά στην επόμενη εντολή μετά τον βρόχο.

παράδειγμα

```
using System;

class BreakTester
{
    static void Main(string[] args)
    {
        string output = "";
        int count;

        for (count = 1; count <= 10; count++)
        {
            if (count == 3)
                break;

            output += count + "*";
        }

        output += "\nBroke out of loop at count = " + count;

        Console.WriteLine("Apotelesma ekteleshs:\n"+output);
        Console.ReadKey();
    }
}
```

continue

η *λειτουργία* της continue είναι η εξής: οδηγεί τον κώδικα στην επόμενη επανάληψη , παρακάμπτοντας την κανονική δομή ελέγχου του βρόχου.

παράδειγμα

```
using System;

class ContDemo
{
  static void Main()
  {
    int i;

    // Print even number between 0 and 100.
    for (i = 0; i <= 100; i++)
    {

      // Iterate if i is odd.
      if ((i % 2) != 0)
        continue;

      Console.WriteLine(i);
    }

    Console.ReadKey();
  }
}
```

3.4

Ερωτήσεις κατανόησης

1 Προτείνετε αρχικές τιμές για τα x και y έτσι ώστε ο ακόλουθος κώδικας να εκτυπώσει error:

```
using System;

class Program
{
   static void Main()
   {
      int x=?, y=?,z=-3;

      bool done=true;

      if(x<10)
         if (y > 100){
            if (!done) x = z;
            else  y = z;
         }
         else
            Console.WriteLine("error");
   }
}
```

2 Στο παρακάτω απόσπασμα κώδικα, μετά την εκτέλεση της πρότασης break, τι εμφανίζεται;

```
for (i = 0; i < 10; i++)
 {
    while (running)
    {
       if (x < yield)
          break;
          //...
    }
    Console.WriteLine("after while");
 }

 Console.WriteLine("after for");
```

3 Τι εκτυπώνει ο παρακάτω κώδικας;

```
using System;

class Program
{
   static void Main()
   {
      for (int i = 0; i <= 10; i++)
      {
         Console.Write(i + " ");
         if ((i % 2) == 0)
            continue;
         Console.WriteLine();
      }
   }
}
```

4 Τι εκτυπώνει ο παρακάτω κώδικας; Υπάρχει κάτι παράξενο;

```
using System;

class Program
{
   static void Main()
   {
      for (int i = 0; i <= 10; i++)
      {
         Console.Write(i + " ");
         if ((i % 2) == 0)
            continue;
         else
            break;
         Console.WriteLine();
```

```
    }
  }
}
```

5 Συμπληρώστε τον ακόλουθο κώδικα έτσι ώστε αυτός να εκτυπώνει την ακολουθία 1 2 4 8 16 32...

```
for (int i = ?; i < 100; i+=?)
{
  Console.WriteLine(i + " ");
}
```

6 Εκτελούμε το παρακάτω πρόγραμμα και εισάγουμε από το πληκτρολόγιο τις τιμές 11 και 10 για τα x και y αντίστοιχα. Τι θα εκτυπωθεί;

```
using System;

class Program
{
  static void Main()
  {
    int x = Int32.Parse(Console.ReadLine());
    int y = Int32.Parse(Console.ReadLine());

    switch (x)
    {
      case 1:
      case 2: Console.WriteLine("1 ή 2"); break;
      case 7: Console.WriteLine("7"); break;
      case 11: Console.WriteLine("2999"); while(y<100) continue; break;
      default: Console.WriteLine("error"); break;
    }
  }
}
```

3.5

Ασκήσεις

Να γίνει πρόγραμμα το οποίο να επιλύει οποιαδήποτε εξίσωση 1ου βαθμού

ax+b=0

αν ο χρήστης εισάγει τις παραμέτρους α & β .

```
using System;

class PrwtovathmiaExiswsi
{
  static void Main()
  {
    string strA, strB;

    Console.WriteLine("Dwse tous syntelestes ths exiswsis:");

    strA = Console.ReadLine();
    strB = Console.ReadLine();

    float a, b;

    a = Single.Parse(strA);
    b = Single.Parse(strB);


    if (a == 0)
      if (b == 0)
        Console.WriteLine("H exiswsh {0}x+({1})=0 einai Aoristh.", a, b);
      else
        Console.WriteLine("H exiswsh {0}x+({1})=0 einai Adynath.", a, b);
    else
    {
      float x;

      x = -b / a;
```

```
        Console.WriteLine("H exiswsh {0}x+({1})=0 exei monadikh lush thn {2}.", a, b,
x);
    }

    Console.ReadKey();
  }
}
```

Να γίνει πρόγραμμα το οποίο:

να εξετάζει αν ένα έτος που δίνεται από τον χρήστη είναι δίσεκτο.

```
using System;

class DisektoEtos
{
    static void Main()
    {
        string strEtos;

        Console.WriteLine("Dwse to etos:");
        strEtos = Console.ReadLine();

        int etos;

        etos = Int32.Parse(strEtos);

        bool flag = false;

        if (etos % 100 == 0)
        {
            if (etos % 400 == 0)
                flag = true;
        }
        else
            if (etos % 4 == 0)
                flag = true;

        if (flag == true)
            Console.WriteLine("To etos {0} einai Disekto.",etos);
        else
            Console.WriteLine("To etos {0} den einai
Disekto.",etos);

        Console.ReadKey();
    }
}
```

Να γίνει πρόγραμμα το οποίο να υπολογίζει το **<u>άθροισμα</u>**:

S= 1+3+5+…+N

όταν ο χρήστης δίνει το N.

```
using System;

class Program
{
  const int ARXH = 1, VHMA = 2;
  static void Main()
  {
    int s = 0, N = -1, i = ARXH;

    while (N < 1 || N % 2 == 0)
    {
      Console.WriteLine("Dwse ton teliko oro N (thetikos & perittos):");
      N = Int32.Parse(Console.ReadLine());
    }

    while (i <= N)
    {
      s = s + i;
      i += VHMA;
    }

    Console.WriteLine("To athroisma ths akolouthias me arxiko oro {0} , vima {1} &
teliko oro {2} einai S={3}.", ARXH, VHMA, N, s);

    Console.ReadKey();
  }
}
```

Να γραφεί πρόγραμμα το οποίο να υπολογίζει το

```
using System;

class Athroisma
```

Ν παραγοντικό

με βάση τον τύπο

N!=1*2*...*(N-1)*N

περιορισμοί για το Ν: θετικός

```
{
   static void Main()
   {
      int N;

      do
      {
         Console.WriteLine("Dwse to N:");
         N = Int32.Parse(Console.ReadLine());
      } while (N < 1);

      int P = 1;

      for (int i = 1; i <= N; i++)
         P = P * i;

      Console.WriteLine(N + "!= " + P);
      Console.ReadKey();
   }
}
```

Να γίνει πρόγραμμα το οποίο να εξετάζει

αν ένας αριθμός (μεγαλύτερος της μονάδος) είναι πρώτος ή όχι.

Σημείωση:
Ένας **αριθμός Ν** ονομάζεται **πρώτος** αν ***δεν διαιρείται ακριβώς*** με κανέναν αριθμό στο διάστημα **[2,N-1]**.

```
using System;

class Prwtos
{
   static void Main()
   {
      int N;

      do
      {
         Console.Write("Dwse to N:");
         N = Int32.Parse(Console.ReadLine());
      } while (N < 2);

      bool bPrwtos = true;

      for (int i = 2; i < N; i++)
         if (N % i == 0)
         {
            bPrwtos = false;
            break;
         }

      if (bPrwtos)
         Console.WriteLine("O arithmos " + N + " einai prwtos!");
      else
         Console.WriteLine("O arithmos " + N + " den einai prwtos!");

      Console.ReadKey();
   }
}
```

4

Κλάσεις, Αντικείμενα & Μέθοδοι

Τι θα δούμε σε αυτό το μάθημα

1. Γενικά περί Κλάσεων
2. Δημιουργία μίας Κλάσης
3. Παράδειγμα: η κλάση Oxhma
4. Δημιουργία ενός αντικειμένου
5. Μέθοδοι
6. Κατασκευαστές
7. Καταστροφείς
8. Η λέξη-κλειδί this
9. Ερωτήσεις κατανόησης
10. Άσκηση

4.1

Γενικά περί Κλάσεων

ορισμός

κλάση είναι ένα *πρότυπο* που ορίζει την μορφή ενός *αντικειμένου*. Καθορίζει τα **δεδομένα** και τον **κώδικα** που θα δρα πάνω σε αυτά τα δεδομένα. Τα αντικείμενα είναι *στιγμιότυπα* της κλάσης στην οποία ανήκουν. Ότι ανήκει μέσα σε μία κλάση αποτελεί ***μέλος*** της κλάσης.

τι μπορεί να περιέχει μία κλάση;

οι περισσότερες κλάσεις περιέχουν **κώδικα** και **δεδομένα**.
Γενικότερα, μία κλάση μπορεί να έχει ως μέλη τα εξής:
μεταβλητές στιγμιοτύπου, στατικές μεταβλητές, σταθερές, μεθόδους, κατασκευαστές, καταστροφείς, δεικτοδότες, συμβάντα, τελεστές και ιδιότητες.

4.2

Δημιουργία μίας κλάσης

γενική μορφή μίας κλάσης

```
class ΌνομαΚλάσης{
//δήλωση μεταβλητών
προσπέλαση τύπος όνομα_μεταβλητής1;
προσπέλαση τύπος όνομα_μεταβλητής2;
...
προσπέλαση τύπος όνομα_μεταβλητήςΝ;

//δήλωση μεθόδων
προσπέλαση τύπος_επιστροφής όνομα_μεθόδου1 (παράμετροι){
//σώμα μεθόδου
}
προσπέλαση τύπος_επιστροφής όνομα_μεθόδου2 (παράμετροι){
//σώμα μεθόδου
}

...

προσπέλαση τύπος_επιστροφής όνομα_μεθόδουΝ(παράμετροι){
//σώμα μεθόδου
}
```

Καθοριστής προσπέλασης μέλους

ο καθοριστής προσπέλασης καθορίζει ποιος μπορεί να χρησιμοποιήσει την μεταβλητή ή τη μέθοδο μίας κλάσης. Υπάρχουν 4 καθοριστές προσπέλασης:

- **public** (δημόσια): το μέλος μπορεί να προσπελαύνεται από κάθε κώδικα που βρίσκεται μέσα σε ένα πρόγραμμα, φυσικά και εκτός της κλάσης.
- **private** (ιδιωτική): το μέλος μπορεί να προσπελαστεί μόνο από άλλα μέλη της κλάσης στην οποία βρίσκεται.
- **protected** (προστατευμένη)
- **internal** (εσωτερική)

Ο καθοριστής προσπέλασης είναι *προαιρετικός*. Αν λείπει, τότε το μέλος είναι private (ιδιωτικό).

4.3

Παράδειγμα: η κλάση Oxhma

γενικά

μία κλάση πρέπει να ορίζει μία και μόνο μία λογική οντότητα. Σε διαφορετική περίπτωση θα υπάρξουν σοβαρά προβλήματα στον χειρισμό του κώδικά σας.

Αν θέλουμε να έχουμε συγκεντρωμένα στοιχεία σχετικά με οχήματα , οπότε μας ενδιαφέρει να αποθηκεύσουμε πληροφορία σχετικά την μάρκα, με τον αριθμό επιβατών και την χωρητικότητα σε καύσιμο τότε μας βολεύει να ορίζουμε μία κλάση με το όνομα Vehicle

η κλάση Oxhma

```
class Oxhma{
public string marka;
public int epivates;
public int kausimo;
public double katanalwsh;
}
```

Η κλάση αυτή περιέχει μόνο δεδομένα, δηλαδή μεταβλητές. Τέτοιου τύπου μεταβλητές ονομάζονται μεταβλητές στιγμιοτύπου, με γενικό ορισμό τον εξής:

προσπέλαση τύπος όνομα_μεταβλητής;

Για να μπορούμε να χρησιμοποιήσουμε την κλάση θα πρέπει να ορίζουμε ένα στιγμιότυπό της, δηλαδή ένα αντικείμενο.

4.4

Δημιουργία ενός αντικειμένου

σύνταξη

ένα αντικείμενο της κλάσης Oxhma μπορεί να δημιουργηθεί ως εξής:

Oxhma ix = new Oxhma();

λειτουργία

αρχικά δηλώνουμε μία μεταβλητή με το όνομα ix τύπου κλάσης Oxhma. Αυτή η μεταβλητή δεν είναι από μόνη της ένα αντικείμενο. Το αντικείμενο δημιουργείται πραγματικά με τον τελεστή **new** και στο τέλος αναθέτουμε την μεταβλητή ix μία αναφορά στο αντικείμενο που μόλις δημιουργήθηκε στη μνήμη. Γι'αυτό το λόγο οι κλάσεις ονομάζονται *τύποι αναφοράς*.
Ο ***τελεστής new*** κατανέμει δυναμικά μνήμη για ένα αντικείμενο και επιστρέφει μία αναφορά προς αυτό, η οποία αποθηκεύεται σε μιά μεταβλητή.

τύποι αναφοράς

οι διαφορές μεταξύ ***τύπων τιμών*** και ***τύπων αναφοράς*** είναι μεγάλες. Μία μεταβλητή τύπου τιμής περιέχει *η ίδια την τιμή*, Για παράδειγμα, αν είχαμε την παρακάτω δήλωση

int x=10;

τότε η μεταβλητή x περιέχει πραγματικά τη τιμή 10. Σε αντίθεση, μία μεταβλητή τύπου αναφοράς όπως η μεταβλητή ix στη δήλωση

Oxhma ix = new Oxhma();

δεν περιέχει το αντικείμενο αλλά ***μία αναφορά προς το αντικείμενο.***
Επίσης, σε περίπτωση ανάθεσης, ενώ σε μεταβλητές τύπου τιμής απλά η μεταβλητή στο αριστερό μέρος δέχεται ένα αντίγραφο της τιμής της μεταβλητής στο δεξιό μέρος, στις μεταβλητές τύπου αναφοράς η μεταβλητή στο αριστερό μέρος αναφέρεται στο ίδιο αντικείμενο που αναφέρεται και η μεταβλητή στο δεξί μέρος:

Oxhma car1=new Oxhma();
Oxhma car2=car1;

Αυτό σημαίνει, ότι αν προσπαθήσουμε να αλλάξουμε τιμή σε οποιαδήποτε μεταβλητή-μέλος του αντικειμένου τότε η αλλαγή θα ισχύει και για τις δύο μεταβλητές.

Παράδειγμα κεφαλαίου

(χρήση της κλάσης Oxhma, η οποία περιέχει μόνο δεδομένα)

```
using System;

class Oxhma{
   public string marka;
   public int epivates;
   public int kausimo;
   public double katanalwsh;
}

class Program
{
   static void Main()
   {
      Oxhma minivan = new Oxhma();
      Oxhma sportscar = new Oxhma();

      Console.WriteLine("Dwse ta stoixeia gia to minivan:");
      Console.Write("Marka:");
      minivan.marka = Console.ReadLine();
      Console.Write("Arithmos epivatwn:");
      minivan.epivates = Int32.Parse(Console.ReadLine());
      Console.Write("Xwrhtikothta kausimou:");
      minivan.kausimo = Int32.Parse(Console.ReadLine());
      Console.Write("Katanalwsh kausimou (lt/km)");
      minivan.katanalwsh = Double.Parse(Console.ReadLine());

      Console.WriteLine("Dwse ta stoixeia gia to sport autokinhto:");
      Console.Write("Marka:");
      sportscar.marka = Console.ReadLine();
      Console.Write("Arithmos epivatwn:");
      sportscar.epivates = Int32.Parse(Console.ReadLine());
      Console.Write("Xwrhtikothta kausimou:");
      sportscar.kausimo = Int32.Parse(Console.ReadLine());
      Console.Write("Katanalwsh kausimou (lt/km)");
      sportscar.katanalwsh = Double.Parse(Console.ReadLine());

      double apostash1,apostash2;

      apostash1 = minivan.kausimo*100 / minivan.katanalwsh;
      apostash2 = sportscar.kausimo *100/ sportscar.katanalwsh;

      Console.WriteLine("To minivan einai markas {0}, metaferei {1} atoma kai exei autonomia kausimou {2:.00}
km.",minivan.marka,minivan.epivates,apostash1);
      Console.WriteLine("To sportscar einai markas {0}, metaferei {1} atoma kai exei autonomia kausimou {2:.00} km.", sportscar.marka,
sportscar.epivates, apostash2);

      Console.ReadKey();
   }
}
```

ο τελεστής τελεία (.)

για να προσπελάσουμε τα μέλη μίας κλάσης με χρήση ενός αντικειμένου της , χρησιμοποιούμε τον *τελεστή τελεία* . Ο τελεστής αυτός συνδέει το όνομα του αντικειμένου με το όνομα του μέλους:

αντικείμενο.μέλος

4.5 Μέθοδοι

ορισμός

οι μέθοδοι κλάσης είναι υπορουτίνες που *χειρίζονται τα δεδομένα της* και σε πολλές περιπτώσεις *παρέχουν πρόσβαση* σε αυτά. Ουσιαστικά, άλλα μέρη του

προγράμματος θα αλληλεπιδρούν με μία κλάση μέσω των μεθόδων της. Τέλος, μία μέθοδος υλοποιεί μία συγκεκριμένη λειτουργία, μπορεί να χρειάζεται ορίσματα για να δουλέψει και μετά το πέρας λειτουργίας της να επιστρέφει κάτι σε αυτόν που την κάλεσε.

σύνταξη η γενική μορφή μίας μεθόδου είναι η εξής:

προσπέλαση τύπος_επιστροφής όνομα_μεθόδου(παράμετροι){

//κώδικας

}

Η προσπέλαση καθορίζει ποιος μπορεί να χρησιμοποιεί αυτή τη μέθοδο, ο τύπος_επιστροφής καθορίζει τι τύπου είναι η πληροφορία που επιστρέφει η μέθοδος και εντός των παρενθέσεων υπάρχουν τα ονόματα και οι τύποι των ορισμάτων (χωρισμένα με κόμματα) που χρειάζεται η μέθοδος για να δουλέψει.

Παράδειγμα κεφαλαίου (συνέχεια)

Στο προηγούμενο παράδειγμα η βασική μέθοδος υπολόγιζε τις αποστάσεις που θα μπορούσε να διανύσει ένα minivan και ένα σπορ αυτοκίνητο. Αυτοί οι υπολογισμοί έχουν άμεση σχέση με τις μεταβλητές kausimo και katanalwsh , οι οποίες ενθυλακώνονται στην κλάση Oxhma. Προκύπτει εύλογα το συμπέρασμα ότι το πιο σωστό θα ήταν να ενθυλακώσουμε στη κλάση αυτή μία μέθοδο η οποία να υπολογίζει την απόσταση που μπορεί να διανύσει ένα όχημα:

```
using System;

class Oxhma{
   public string marka;
   public int epivates;
   public int kausimo;
   public double katanalwsh;

   public double Apostash()
   {
      return kausimo * 100 / katanalwsh;
   }
}

class Program
{
   static void Main()
   {
      Oxhma minivan = new Oxhma();
      Oxhma sportscar = new Oxhma();

      Console.WriteLine("Dwse ta stoixeia gia to minivan:");
      Console.Write("Marka:");
      minivan.marka = Console.ReadLine();
      Console.Write("Arithmos epivatwn:");
      minivan.epivates = Int32.Parse(Console.ReadLine());
      Console.Write("Xwrhtikothta kausimou:");
      minivan.kausimo = Int32.Parse(Console.ReadLine());
      Console.Write("Katanalwsh kausimou (lt/km)");
      minivan.katanalwsh = Double.Parse(Console.ReadLine());

      Console.WriteLine("Dwse ta stoixeia gia to sport autokinhto:");
      Console.Write("Marka:");
      sportscar.marka = Console.ReadLine();
      Console.Write("Arithmos epivatwn:");
      sportscar.epivates = Int32.Parse(Console.ReadLine());
      Console.Write("Xwrhtikothta kausimou:");
      sportscar.kausimo = Int32.Parse(Console.ReadLine());
      Console.Write("Katanalwsh kausimou (lt/km)");
      sportscar.katanalwsh = Double.Parse(Console.ReadLine());

      double apostash1,apostash2;
```

```
        apostash1 = minivan.Apostash();
        apostash2 = sportscar.Apostash();

        Console.WriteLine("To minivan einai markas {0}, metaferei {1} atoma kai exei autonomia kausimou {2:.00}
km.",minivan.marka,minivan.epivates,apostash1);
        Console.WriteLine("To sportscar einai markas {0}, metaferei {1} atoma kai exei autonomia kausimou {2:.00} km.", sportscar.marka,
sportscar.epivates, apostash2);

        Console.ReadKey();
    }
}
```

4.6

Κατασκευαστές

ορισμός ο κατασκευαστής είναι μία *μέθοδος*, η οποία αρχικοποιεί ένα αντικείμενο της κλάσης όταν αυτό δημιουργείται, δηλαδή δίνει αρχικές τιμές στα δεδομένα του. Έχει το ίδιο όνομα με την κλάση αλλά δεν έχουν ρητό επιστρεφόμενο τύπο.

σύνταξη η γενική μορφή ενός κατασκευαστή είναι η εξής:

προσπέλαση όνομα_κλάσης(παράμετροι){

//κώδικας

}

Συνήθως, η *προσπέλαση* είναι public για να μπορεί να κληθεί εκτός της κλάσης και μπορεί να περιέχει ή όχι ορίσματα.

Όλες οι κλάσεις έχουν κατασκευαστέςχωρίς ορίσματα, ακόμα και αν δεν δηλώσουμε ρητά εμείς κάποιον (σε αυτή τη περίπτωση το κάνει ο μεταγλωττιστής και αρχικοποιεί όλες τις μεταβλητές-μέλη με τις προεπιλεγμένες τιμές τους).

Παράδειγμα κεφαλαίου (συνέχεια)

Αν θέλαμε να προσθέσουμε στη κλάση Oxhma έναν κατασκευαστή, τότε αυτός θα μπορούσε να αρχικοποιεί τις μεταβλητές-μέλη ενός αντικειμένου με τις τιμές που δίνει ο χρήστης από το πληκτρολόγιο. Επίσης, μπορούμε να εισάγουμε και μία νέα μέθοδο, η οποία υπολογίζει πόσα λίτρα βενζίνης θα χρειαστεί ένα όχημα για να διανύσει συγκεκριμένα χιλιόμετρα:

```
using System;

class Oxhma{
    public string marka;
    public int epivates;
    public int kausimo;
    public double katanalwsh;

    public Oxhma(string m, int e, int k, double kt)
    {
        marka = m;
        epivates = e;
        kausimo = k;
        katanalwsh = kt;
    }

    public double Apostash()
    {
        return kausimo * 100 / katanalwsh;
    }

    public double YpologismosKausimou(double xiliometra)
    {
        return xiliometra * katanalwsh / 100;
```

```
    }
}

class Program
{
    static void Main()
    {
        string ma;
        int ep;
        int ka;
        double kat;

        Console.WriteLine("Dwse ta stoixeia gia to minivan:");
        Console.Write("Marka:");
        ma = Console.ReadLine();
        Console.Write("Arithmos epivatwn:");
        ep = Int32.Parse(Console.ReadLine());
        Console.Write("Xwrhtikothta kausimou:");
        ka = Int32.Parse(Console.ReadLine());
        Console.Write("Katanalwsh kausimou (lt/km)");
        kat = Double.Parse(Console.ReadLine());

        Oxhma minivan = new Oxhma(ma, ep, ka, kat);

        Console.WriteLine("Dwse ta stoixeia gia to sport autokinhto:");
        Console.Write("Marka:");
        ma = Console.ReadLine();
        Console.Write("Arithmos epivatwn:");
        ep = Int32.Parse(Console.ReadLine());
        Console.Write("Xwrhtikothta kausimou:");
        ka = Int32.Parse(Console.ReadLine());
        Console.Write("Katanalwsh kausimou (lt/km)");
        kat = Double.Parse(Console.ReadLine());

        Oxhma sportscar = new Oxhma(ma, ep, ka, kat);

        double apostash1,apostash2;

        apostash1 = minivan.Apostash();
        apostash2 = sportscar.Apostash();

        int xiliometraTaxidiou = 250;

        Console.WriteLine("To minivan einai markas {0} kai gia na dianysei {1} Km tha xreiastei {2:.0} Lt
venzinis",minivan.marka,xiliometraTaxidiou,minivan.YpologismosKausimou(xiliometraTaxidiou));
        Console.WriteLine("To sportscar einai markas {0} kai gia na dianysei {1} Km tha xreiastei {2:.0} Lt venzinis", sportscar.marka,
xiliometraTaxidiou, sportscar.YpologismosKausimou(xiliometraTaxidiou));

        Console.ReadKey();
    }
}
```

4.7

Καταστροφείς

ορισμός καταστροφέας είναι μία μέθοδος-μέλος, η οποία απελευθερώνει τη μνήμη που δεσμεύθηκε για τη δημιουργία ενός αντικειμένου (από τον τελεστή new). Στην ουσία καταστρέφει το αντικείμενο.

σύνταξη η γενική σύνταξη ενός καταστροφέα είναι η εξής:

~όνομα_κλάσης(){

//κώδικας

}

Ο καταστροφέας καλείται πριν από τη λεγόμενη *συλλογή απορριμάτων*, κάτι που

γίνεται στο παρασκήνιο (*χωρίς* τη συμμετοχή του προγραμματιστή), έτσι ώστε να απελευθερωθεί μνήμη.

4.8

Η λέξη-κλειδί this

ορισμός

η λέξη-κλειδί this αποτελεί μία *αναφορά* στο αντικείμενο στο οποίο δρα μία μέθοδος που έχει κληθεί. Έτσι, θα μπορούσε να χρησιμοποιηθεί για όλες τις μεταβλητές-μέλη μέσα στον κώδικα μίας κλάσης. Αυτό είναι αναγκαίο διότι η C# επιτρέπει τα ονόματα μίας μεθόδου να είναι τα ίδια με τα ονόματα των μεταβλητών (στιγμιοτύπου) της κλάσης. Έτσι, στον κατασκευαστή της κλάσης Oxhma θα μπορούσαμε να είχαμε τα εξής:

```
public Oxhma(string marka, int epivates, int kausimo, double katanalwsh)
{
    this. marka = marka;
    this.epivates = epivates;
    this.kausimo = kausimo;
    this.katanalwsh = katanalwsh;
 }
```

Σε αυτή τη περίπτωση οι τοπικές μεταβλητές *κρύβουν* τις μεταβλητές-μέλη της κλάσης και για να έχουμε πρόσβαση σε αυτές χρησιμοποιούμε τη λέξη this.

4.9

Ερωτήσεις κατανόησης

1 τι είναι κλάση και τι αντικείμενο;

2 Δείτε τον παρακάτω πρόγραμμα. Βρείτε τα λάθη του έτσι ώστε να μην υπάρχουν συντακτικά λάθη.

```
using System;

class Program
{
   class Foititis
   {
      string onomatepwnymo;
      string AM;
      float programmatismos1;
      float programmatismos2;
      float programmatismos3;

      float YpologismosMO()
      {
         return (programmatismos1 + programmatismos2 + programmatismos3) / 3;
      }
   }

   static void Main()
   {
      Foititis f1,f2 = new Foititis();

      Console.WriteLine("O mo gia ton 1o foithth einai:{0:.00}",f1.YpologismosMO());
      Console.WriteLine("O mo gia ton 2o foithth einai : {0:.00}".f2.YpologismosMO());

      Console.ReadKey();
   }
}
```

3 Γράψτε έναν κατασκευαστή για το πρόγραμμα στο ερώτημα 2, έτσι ώστε να δίνονται αρχικές τιμές σε όλες τις μεταβλητές στιγμιοτύπου. Θα αλλάζατε κάτι άλλο στο κυρίως πρόγραμμα;

4 Τι θα εκτυπώσει ο παρακάτω κώδικας;

```
using System;

class Klash
{
   public int a;
   public int b;

   public Klash()
   {
      a = 0;
      b = 0;
   }
}

class Program
{
  static void Main()
   {
      Klash x1 = new Klash();
      Klash x2 = x1;

      Console.WriteLine("Times tou antikeimenou x1:{0},{1}",x1.a,x1.b);
      Console.WriteLine("Times tou antikeimenou x2:{0},{1}", x2.a, x2.b);

      x2.b = 25;
      x1.a = 90;

      Console.WriteLine("Times tou antikeimenou x1:{0},{1}", x1.a, x1.b);
      Console.WriteLine("Times tou antikeimenou x2:{0},{1}", x2.a, x2.b);

      Klash x3 = new Klash();

      x2 = x3;

      Console.WriteLine("Times tou antikeimenou x1:{0},{1}", x1.a, x1.b);
      Console.WriteLine("Times tou antikeimenou x2:{0},{1}", x2.a, x2.b);
      Console.WriteLine("Times tou antikeimenou x3:{0},{1}", x3.a, x3.b);

      Console.ReadKey();
   }
}
```

5 Τι θα εκτυπώσει ο παρακάτω κώδικας;

```
using System;

class Factor
{
   // Determine if x is a factor of y.
   public bool IsFactor(int x, int y)
   {
      if ((y % x) == 0) return true;
      else return false;
   }
}

class IsFact
{
   static void Main()
   {
      Factor x = new Factor();

      if (x.IsFactor(2, 20)) Console.WriteLine("2 is factor");
      if (x.IsFactor(3, 20)) Console.WriteLine("this won't be displayed");

      Console.ReadKey();
   }
```

```
}
```

4.10

Άσκηση

Να γραφεί πρόγραμμα, το οποίο με τη χρήση της κλάσης Cycle, θα υπολογίζει και θα εκτυπώνει το εμβαδόν και την περίμετρο του κύκλου.

```
using System;

class Cycle
{
   //δεδομένα-μεταβλητές
   double aktina;

   //κατασκευαστής
   public Cycle(double aktina)
   {
      this.aktina = aktina;
   }

   //μέθοδοι
   public void SetAktina(double aktina)
   {
      this.aktina = aktina;
   }

   public double GetAktina()
   {
      return aktina;
   }

   public double YpologismosEmvadou()
   {
      return Math.PI * Math.Pow(aktina, 2.0);
   }

   public double YpologismosPerimetrou()
   {
      return 2 * Math.PI * aktina;
   }
}

class Program
{
   static void Main()
   {

      Cycle kiklos = new Cycle(0.0);

      do{
      Console.WriteLine("Dwse thn aktina tou kuklou:");
      kiklos.SetAktina(Double.Parse(Console.ReadLine()));
      }while(kiklos.GetAktina()<=0.0);

      Console.WriteLine("O kiklos me aktina {0:.0} exei emvadon {1:.0} kai perimetro
{2:.0}",
kiklos.GetAktina(),kiklos.YpologismosEmvadou(),kiklos.YpologismosPerimetrou());
      Console.ReadKey();
   }
}
```

5

Πίνακες & Strings

Τι θα δούμε σε αυτό το μάθημα

1. Γενικά περί Πινάκων
2. Μονοδιάστατοι Πίνακες
3. Δισδιάστατοι Πίνακες
4. Ακανόνιστοι Πίνακες
5. Η ιδιότητα Length
6. Υλοποίηση ουράς
7. String
8. Ο βρόχος foreach
9. Ερωτήσεις κατανόησης
10. Άσκηση

5.1

Γενικά περί Πινάκων

ορισμός πίνακας είναι μία *ομάδα μεταβλητών του ιδίου τύπου*, στις οποίες αναφερόμαστε με ένα *κοινό όνομα*. Το πλεονέκτημά τους είναι ότι μπορούμε να τους χειριστούμε εύκολα, λόγω της οργανωμένης δομής τους.

υλοποίηση η υλοποίηση των πινάκων στη *C#* διαφέρει από τη *C* διότι γίνεται ως ***αντικείμενα***, προσφέροντας αρκετά πλεονεκτήματα (μεγαλύτερη ευελιξία, συλλογή απορριμάτων)

5.2

Μονοδιάστατοι πίνακες

δήλωση οι πίνακες δηλώνονται αντίστοιχα με τα αντικείμενα, δηλαδή πρέπει να υλοποιηθούν 2 πράγματα, *δήλωση μεταβλητής* που θα αναφέρεται στον πίνακα και *δέσμευση μνήμης* για τον πίνακα. Ο γενικός τύπος είναι ο εξής:

τύπος[] όνομα_πίνακα= new τύπος [μέγεθος_πίνακα];

Για παράδειγμα, αν θέλαμε να δηλώσουμε έναν πίνακα για να αποθηκεύσουμε 5 βαθμολογίες φοιτητών και στη συνέχεια να γεμίσουμε αυτόν τον πίνακα, θα γράφαμε τα ακόλουθα:

```
using System;

class Program
{
   static void Main()
   {
      double[] vathmoi = new double[5];

      for (int i = 0; i < 5; i++)
         vathmoi[i] = Double.Parse(Console.ReadLine());

      Console.ReadKey();
   }
}
```

αρχικοποίηση ενός πίνακα η αρχικοποίηση ενός μονοδιάστατου πίνακα γίνεται ως εξής:

τύπος[] όνομα_πίνακα=**{τιμή1, τιμή2, ...,τιμήΝ}**;

Όταν αρχικοποιούμε τον πίνακα δεν χρειάζεται να χρησιμοποιήσουμε τον τελεστή new (δεν είναι λάθος αν το κάνουμε όμως).

Παράδειγμα

(ταξινόμηση πίνακα με τη μέθοδο bubble sort)

```
using System;
```

```
class Bubble
{
   static void Main()
   {
      int[] nums = { 99, -10, 100123, 18, -978, 5623, 463, -9, 287, 49 };
      int a, b, t;
      int size;
      size = 10;

      Console.Write("Original array is:");

      for (int i = 0; i < size; i++)
         Console.Write(" " + nums[i]);

      Console.WriteLine();

      // This is the bubble sort.
      for (a = 1; a < size; a++)
         for (b = size - 1; b >= a; b--)
         {
            if (nums[b - 1] > nums[b])
            {
               t = nums[b - 1];
               nums[b - 1] = nums[b];
               nums[b] = t;
            }
         }

      Console.Write("Sorted array is:");
      for (int i = 0; i < size; i++)
         Console.Write(" " + nums[i]);

      Console.ReadKey();
   }
}
```

5.3

Παράμετροι στη Main()

ορισμός

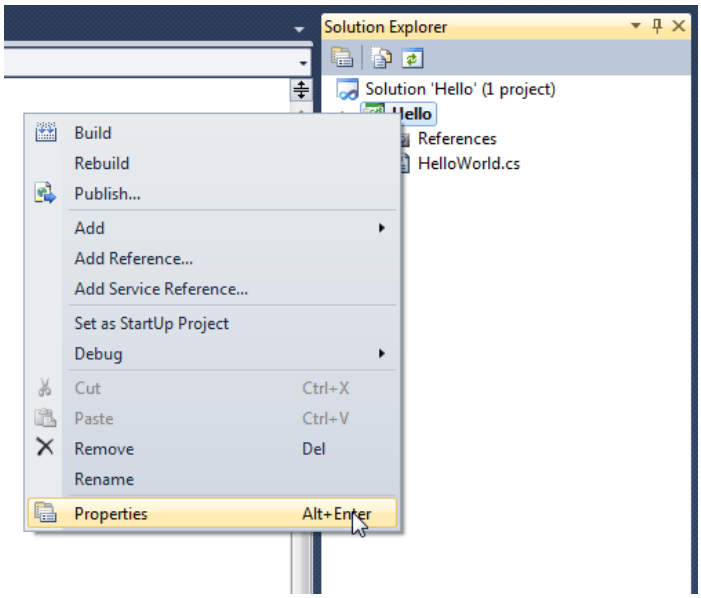
είδαμε στο προηγούμενο παράδειγμα του HelloWorld.cs ότι μπορούμε να περάσουμε ως όρισμα στη βασική συνάρτηση έναν *πίνακα αλφαριθμητικών* με το όνομα **args**. Η δήλωση του πίνακα είναι η εξής:

string [] args

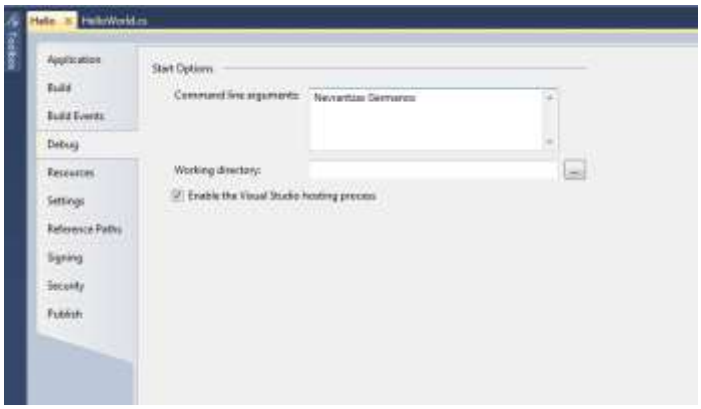
Ο πίνακας αυτός διοχετεύει στο πρόγραμμα (άρα αυτό μπορεί να τα χρησιμοποιήσει στη συνέχεια) ότι πληροφορία του δώσει ο χρήστης με *εξωτερικό τρόπο.*

διοχεύτεση ορισμάτων με εξωτερικό τρόπο

για να περάσουμε ορίσματα μέσω της μεθόδου Main() κάνουμε δεξί κλικ στο όνομα του project μέσα στο Solution Explorer και επιλέγουμε *Properties*:



στη συνέχεια πάμε στη καρτέλα *Debug* και στο πλαίσιο *Command line arguments* πληκτρολογούμε όλα τα ορίσματα που θέλουμε να περάσουμε στο πρόγραμμά μας (χωρισμένα με κενά):



και κλείνουμε το παράθυρο των ιδιοτήτων. Με αυτό τον τρόπο, όταν θα τρέξουμε το πρόγραμμα, θα έχουμε αποθηκεύσει στον πίνακα args και συγκεκριμένα στη θέση 0 το Nevrantzas , ενώ στη θέση 1 το Germanos.

τροποποίηση HelloWorld

έστω ότι θέλουμε να μας δώσει ο χρήστης με εξωτερικό τρόπο τα στοιχεία του , τα οποία στη συνέχεια θέλουμε να εκτυπώσουμε στην κονσόλα. Τροποποιήστε το αρχικό μας πρόγραμμα ως εξής:

```
/*
 HelloWorld.cs
 */

using System;

class HelloWorld
{
    // Κάθε πρόγραμμα στη C# ξεκινά με την κλήση της μεθόδου Main()
    static void Main(string[] args)
    {
        //εκτύπωση ονόματος
        Console.WriteLine(args[1]);
        //εκτύπωση επιθέτου
        Console.WriteLine(args[0]);
    }
}
```

Σε αυτή τη περίπτωση η μέθοδος WriteLine() έχει ως όρισμα μία μεταβλητή, οπότε δεν χρησιμοποιούμε

τους χαρακτήρες " ".

5.4

Παράδειγμα

εκφώνηση
όνομα project:
Welcome
όνομα πηγαίου αρχείου:
WelcomeToCSharp.cs

Να γίνει πρόγραμμα στο οποίο ο φοιτητής θα δίνει το επίθετό του, το όνομά του και τον ΑΜ του και το πρόγραμμα θα εμφανίζει τα στοιχεία του με την εξής μορφή:

Επίθετο, Όνομα, ΑΜ: 1999

κώδικας

```
using System;

class Welcome
{
    static void Main(string[] args)
    {
        Console.WriteLine(args[1] + ", " + args[0] + ", AM:" + args[2]);
        Console.ReadKey();
    }
}
```

5.5

Δισδιάστατοι Πίνακες

δήλωση

στους δισδιάστατους πίνακες η θέση ενός στοιχείου καθορίζεται από δύο δείκτες. Η γενική μορφή είναι η εξής:

τύπος[,] όνομα_πίνακα= new τύπος[μέγεθος1 , μέγεθος2];

Για παράδειγμα, αν θέλαμε να δηλώσουμε ένα πίνακα ακεραίων με διάσταση 10Χ20 τότε θα γράφαμε:

int[,] table=new int[10,20];

αρχικοποίηση

η αρχικοποίηση ενός δισδιάστατου πίνακα ακολουθεί τη λογική ενός μονοδιάστατου:

```
τύπος[ , ] όνομα_πίνακα={
                          {τιμή, τιμή,...,τιμή},
                          {τιμή, τιμή,...,τιμή},
                            ...
                          {τιμή, τιμή,...,τιμή},
                         };
```

5.6

Ακανόνιστοι Πίνακες

δήλωση

η C# επιτρέπει τη δήλωση και χρήση ενός πίνακα πινάκων, δηλαδή ενός πίνακα, του οποίου *το μήκος κάθε γραμμής δεν είναι ίδιο*. Η γενική μορφή δήλωσης είναι η εξής:

τύπος[][] όνομα_πίνακα=new τύπος[μέγεθος][];

Το μέγεθος καθορίζει τον αριθμό των γραμμών. Το μήκος κάθε σειράς δεν έχει

καθοριστεί, θα γίνει μεμονωμένα, πχ:

```
int[][] x=new int[3][];
x[0]=new int[2];
x[1]=new int[3];
x[2]=new int[4];
```

Έτσι, έχουμε δημιουργήσει τον εξής (σχηματικά) πίνακα:

x[0][0]	x[0][1]		
x[1][0]	x[1][1]	x[1][2]	
x[2][0]	x[2][1]	x[2][2]	x[2][3]

Παράδειγμα

```
using System;

class Jagged
{
   static void Main()
   {
      int[][] riders = new int[7][];
      riders[0] = new int[10];
      riders[1] = new int[10];
      riders[2] = new int[10];
      riders[3] = new int[10];
      riders[4] = new int[10];

      riders[5] = new int[2];
      riders[6] = new int[2];

      int i, j;

      // Fabricate some data.
      for (i = 0; i < 5; i++)
         for (j = 0; j < 10; j++)
            riders[i][j] = i + j + 10;
      for (i = 5; i < 7; i++)
         for (j = 0; j < 2; j++)
            riders[i][j] = i + j + 10;

      Console.WriteLine("Riders per trip during the week:");
      for (i = 0; i < 5; i++)
      {
         for (j = 0; j < 10; j++)
            Console.Write(riders[i][j] + " ");
         Console.WriteLine();
      }
      Console.WriteLine();

      Console.WriteLine("Riders per trip on the weekend:");
      for (i = 5; i < 7; i++)
      {
         for (j = 0; j < 2; j++)
            Console.Write(riders[i][j] + " ");
         Console.WriteLine();
      }

      Console.ReadKey();
   }
}
```

5.7

Η ιδιότητα Length

ορισμός εφόσον οι πίνακες υλοποιούνται ως αντικείμενα, μπορούμε να εφαρμόσουμε την ιδιότητα **Length**: η ιδιότητα αυτή φανερώνει τον αριθμό των στοιχείων που περιέχει ο πίνακας.

παράδειγμα

```
using System;

class LengthDemo
{
   static void Main()
   {
      int[] list = new int[10];
      int[,] twoD = new int[3, 4];
      int[] nums = { 1, 2, 3 };

      // A variable-length table.
      int[][] table = new int[3][];

      // Add second dimensions.
      table[0] = new int[] { 1, 2, 3 };
      table[1] = new int[] { 4, 5 };
      table[2] = new int[] { 6, 7, 8, 9 };

      Console.WriteLine("length of list is " + list.Length);
      Console.WriteLine("length of twoD is " + twoD.Length);
      Console.WriteLine("length of nums is " + nums.Length);
      Console.WriteLine("length of table is " + table.Length);
      Console.WriteLine("length of table[0] is " + table[0].Length);
      Console.WriteLine("length of table[1] is " + table[1].Length);
      Console.WriteLine("length of table[2] is " + table[2].Length);
      Console.WriteLine();

      // Use length to initialize list.
      for (int i = 0; i < list.Length; i++)
         list[i] = i * i;

      Console.Write("Here is list: ");
      // now use length to display list
      for (int i = 0; i < list.Length; i++)
         Console.Write(list[i] + " ");
      Console.ReadKey();
   }
}
```

```
length of list is 10
length of twoD is 12
length of nums is 3
length of table is 3
length of table[0] is 3
length of table[1] is 2
length of table[2] is 4

Here is list: 0 1 4 9 16 25 36 49 64 81
```

Το πρόγραμμα εκτυπώνει τα εξής:

5.8

Υλοποίηση ουράς με χρήση πίνακα

Άσκηση

Να γίνει πρόγραμμα το οποίο να υλοποιεί μία απλή ουρά με διεργασίες εισόδου και εξόδου στοιχείων.

```
using System;

class Queue
{
   //metavlhtes
   double[] oura;
   int put, get;

   //kataskeuasths
   public Queue(int megethos)
   {
      oura = new double[megethos + 1];
      put = get = 0;
   }

   //methodoi
   public void Put(double x)
   {
      if (put == oura.Length - 1)
      {
         Console.WriteLine("H oura einai gemath...");
         return;
      }
      put++;
      oura[put] = x;
   }

   public double Get()
   {
      if (get == put)
      {
         Console.WriteLine("H oura einai adeia...");
         return -1;
      }
      get++;
      return oura[get];
   }
}

class Program
{
   static void Main()
   {

      Console.WriteLine("Dwse to megethos ths ouras:");
      Queue ouraAkeraiwn = new Queue(Int32.Parse(Console.ReadLine()));

      int plithos;

      Console.Write("Posous vathmous tha dwseis? - ");
      plithos=Int32.Parse(Console.ReadLine());

      for (int i = 0; i < plithos; i++)
      {
         Console.Write("Dwse ton " + (i + 1) + " vathmo:");
         ouraAkeraiwn.Put(Int32.Parse(Console.ReadLine()));
      }

      Console.WriteLine("H oura sou periexei:");

      double number;

      for (int i = 0; i < plithos; i++)
      {
         number = ouraAkeraiwn.Get();
```

```
        if (number != -1)
            Console.WriteLine(number);
    }

    Console.ReadKey();
  }
}
```

5.9

String

υλοποίηση των string στη C#

η C# υποστηρίζει τον νέο τύπο δεδομένων string. Τα string είναι ***αντικείμενα***, δηλαδή όταν δηλώνουμε μία μεταβλητή τύπου string στην ουσία δημιουργούμε ένα αντικείμενο της κλάσης String. Έτσι, ο τύπος δεδομένων string είναι *τύπος αναφοράς.*

δημιουργία ενός string

υπάρχουν διάφοροι τρόποι για να δημιουργήσουμε ένα string:

- από ένα **κυριολεκτικό string**:
  string x="tei larisas";
- από έναν **πίνακα χαρακτήρων**:
  char[] x={'t','e','i'};
  string s=new string(x);
- από **άλλο string**:
  string s1="tei";
  string s2=s1;

πράξεις σε string

- **static string Copy(string s)** : επιστρέφει αντίγραφο του s. Προσοχή: είναι static, άρα δεν εφαρμόζεται σε αντικείμενο.
- **int CompareTo(string s)** : συγκρίνει το string για το οποίο καλείται με το όρισμα και επιστρέφει 0 εάν είναι ίσα, αλλιώς τιμή διαφορετική του μηδενός.
- **int IndexOf(string s)** : ψάχνει μέσα στο string για το οποίο καλείται για το string s και επιστρέφει τον δείκτη που θα τον βρει ή -1 αν δεν υπάρχει.
- **string Substring(int,int)** : επιστρέφει ένα υποstring, το οποίο αρχίζει από τον ακέραιο που υποδεικνύει το πρώτο όρισμα και έχει μέγεθος που υποδεικνύει το δεύτερο όρισμα.

παράδειγμα

```
using System;

class StrOps
{
  static void Main()
  {
    string str1 ="When it comes to .NET programming, C# is #1.";
    string str2 = string.Copy(str1);
    string str3 = "C# strings are powerful.";
    int result, idx;

    Console.WriteLine("Length of str1: " + str1.Length);

    // Display str1, one char at a time.
    for (int i = 0; i < str1.Length; i++)
      Console.Write(str1[i]);
    Console.WriteLine();

    if (str1 == str2)
      Console.WriteLine("str1 == str2");
    else
```

```
        Console.WriteLine("str1 != str2");

      if (str1 == str3)
        Console.WriteLine("str1 == str3");
      else
        Console.WriteLine("str1 != str3");

      result = str1.CompareTo(str3);
      if (result == 0)
        Console.WriteLine("str1 and str3 are equal");
      else if (result < 0)
        Console.WriteLine("str1 is less than str3");
      else
        Console.WriteLine("str1 is greater than str3");

      // Assign a new string to str2.
      str2 = "One Two Three One";

      idx = str2.IndexOf("One");
      Console.WriteLine("Index of first occurrence of One: " + idx);
      idx = str2.LastIndexOf("One");
      Console.WriteLine("Index of last occurrence of One: " + idx);

    }
}
```

5.10

Ο βρόγχος foreach

σύνταξη

η γενική σύνταξη του βρόχου είναι η εξής:

foreach(τύπος μεταβλητής **in** πίνακας)
{
  //κώδικας
}

Στην ουσία, δηλώνουμε μία νέα μεταβλητή (ιδίου τύπου με τα στοιχεία του πίνακα) η οποία διατρέχει τον πίνακα, στοιχείο προς στοιχείο.

παράδειγμα

```
using System;

class ForeachDemo
{
  static void Main()
  {
    int sum = 0;
    int[] nums = new int[10];

    // Give nums some values.
    for (int i = 0; i < 10; i++)
      nums[i] = i;

    // Use foreach to display and sum the values.
    foreach (int x in nums)
    {
      Console.WriteLine("Value is: " + x);
      sum += x;
    }
    Console.WriteLine("Summation: " + sum);
  }
}
```

5.11

Ερωτήσεις κατανόησης

1 Δηλώστε έναν πίνακα 10X9 με στοιχεία αλφαριθμητικά.

2 Η παρακάτω δήλωση είναι σωστή; αν ναι, τι θα εκτυπωθεί;

```
int[,] x = new int[,] {{3,2,5,4,2},{7,4,2} };
Console.WriteLine(x.Length);
```

3 Θέλετε να διατρέξετε ένα αλφαριθμητικό χαρακτήρα-χαρακτήρα. Ο παρακάτω κώδικας είναι σωστός;

```
 string x = "tei";

 foreach(string i in x)
    Console.WriteLine(i);
```

4 Τι θα εκτυπώσει ο παρακάτω κώδικας;

```
using System;

class Program
{
   static void Main()
   {
      char[] pinax = { 't', 'e', 'i', ' ', 'l',  'a', 'r', 'i', 's', 'a', 's' };
      string s1 = "tei";
      string s2 = new string(pinax);;
      string s3 = string.Copy(s1)+" hpeirou";

      string[] s_array = new string[3];

      s_array[0] = s1;
           s_array[1]=string.Copy(s2.Substring(4,7));
      s_array[2] = s2.Substring(0, 3);

      for (int i = 0; i < s_array.Length; i++)
         Console.WriteLine(s_array[i]);

     Console.ReadKey();
   }
}
```

5.12

Άσκηση

Κατεβάστε το αρχείο Lab5Exercise.cs από το e-class.

Συμπληρώστε την κλάση Foititis με βάση τις εντολές της μεθόδου Main()

```
using System;

class Foititis
{
   //dedomena-metavlites
   string onomatepwnumo;
   double[] vathmoi;

   //kataskeuastis

   //methodoi
   public void SetOnoma(string s)
   {
```

```
        onomatepwnumo = s;
    }

    public void SetPinaka(int n)
    {
        vathmoi = new double[n];
    }

    public void SetVathmousFoititi()
    {
        for (int i = 0; i < vathmoi.Length; i++)
            vathmoi[i] = Double.Parse(Console.ReadLine());
    }

    public double YpologismosMO()
    {
        double sum = 0.0;
        for (int i = 0; i < vathmoi.Length; i++)
            sum += vathmoi[i];

        return sum / vathmoi.Length;
    }

    public void EktypwshMO(){
        Console.WriteLine("{0:.00}", YpologismosMO());
    }
}


class Program
{
    static void Main()
    {

        Foititis f = new Foititis();

        Console.Write("Onoma foithth:");
        f.SetOnoma(Console.ReadLine());
        Console.Write("Plithos vathmwn:");
        f.SetPinaka(Int32.Parse(Console.ReadLine()));

        f.SetVathmousFoititi();
        f.EktypwshMO();

        Console.ReadKey();
    }
}
```

6

Μέθοδοι & Κλάσεις

Τι θα δούμε σε αυτό το μάθημα

---

1. Μεταβίβαση ορισμάτων σε μέθοδο
    1. μεταβίβαση τύπου τιμής
    2. μεταβίβαση αναφοράς τιμής-ref και out
    3. μεταβίβαση αναφοράς αντικειμένου
    4. μεταβίβαση αναφοράς μεταβλητών τύπου αναφοράς
2. Υπερφόρτωση
    1. υπερφόρτωση μεθόδων
    2. υπερφόρτωση κατασκευαστών
    3. κλήση υπερφορτωμένου κατασκευαστή μέσω του this
3. Ερωτήσεις κατανόησης

6.1

Μεταβίβαση ορισμάτων σε μέθοδο

6.1.1

μεταβίβαση τύπου τιμής

σε μία μέθοδο-μέλος μίας τάξης μπορούμε να μεταβιβάσουμε μία ***απλή μεταβλητή*** (τύπου τιμής). Σε αυτή τη περίπτωση στην ουσία μεταβιβάζουμε ένα αντίγραφο της πραγματικής τιμής (που είναι αποθηκευμένη στη μνήμη).
Για παράδειγμα, στον παρακάτω κώδικα οι τιμές των μεταβλητών a και b *δεν αλλάζουν*, παρόλο που μεταβιβάζονται στη μέθοδο-μέλος NoChange() μέσω του αντικειμένου ob. Αυτό που αλλάζει είναι μόνο οι τιμές των *εσωτερικών μεταβλητών i και j.*

παράδειγμα

```
using System;

class Test
{
   // This method causes no change to the arguments used in the call.
   public void NoChange(int i, int j)
   {
      i = i + j;
      j = -j;
   }
}

class CallByValue
{
   static void Main()
   {
      Test ob = new Test();

      int a = 15, b = 20;
      Console.WriteLine("a and b before call: " + a + " " + b);

      ob.NoChange(a, b);
      Console.WriteLine("a and b after call: " + a + " " + b);

      Console.ReadKey();
   }
}
```

6.1.2

μεταβίβαση αναφοράς τιμής

ref και out

σε μία μέθοδο μπορούμε να μεταβιβάσουμε *μεταβλητές τύπου τιμής με* ***αναφορά***. Αν το κάνουμε αυτό, τότε η μέθοδος θα αλλάξει και τις τιμές των ορισμάτων της κλήσης της.

Όταν επιθυμούμε απλά να *επηρεάσουμε τις πραγματικές παραμέτρους* που περνάμε ως ορίσματα σε μία μέθοδο τότε χρησιμοποιούμε τον *τροποποιητή* ***ref*** σε 2 σημεία του προγράμματός μας:

- *πριν* τη δήλωση του ορίσματος που θέλουμε να περνάει με αναφορά, στη δήλωση της μεθόδου μέσα στη κλάση.
- κατά τη κλήση της μεθόδου της κλάσης, *πριν* από το όρισμα.

Σημείωση: όταν ένα όρισμα μεταβιβάζεται με ref τότε ***πρέπει να έχει μία τιμή πριν από τη κλήση.*** Όταν θέλουμε να μεταβιβάσουμε μία μεταβλητή με αναφορά (*χωρίς όμως να της έχουμε αποδόσει ακόμα τιμή*) για να επιστρέψουμε μία τιμή από μία μέθοδο, τότε χρησιμοποιούμε τον *τροποποιητή* ***out*** στα ίδια σημεία με τον ref.

παράδειγμα

```
using System;

class SwapDemo
{
   // This method exchanges its arguments.
   public void Swap(ref int a, ref int b)
   {
      int t;

      t = a;
      a = b;
      b = t;
   }
}

class SwapIt
{
   static void Main()
   {
      SwapDemo ob = new SwapDemo();

      int x = 10, y = 20;
      Console.WriteLine("x and y before call: " + x + " " + y);

      ob.Swap(ref x, ref y);
      Console.WriteLine("x and y after call: " + x + " " + y);

      Console.ReadKey();
   }
}
```

παράδειγμα

```
using System;

class Rectangle
{
   int side1;
   int side2;

   public Rectangle(int i, int j)
   {
      side1 = i;
      side2 = j;
   }
   // Return area and determine if square.
   public int RectInfo(out int per)
   {
      per = 2 * (side1 + side2);

      return side1 * side2;
   }
}

class OutDemo
{
   static void Main(string [] args)
   {
      Rectangle rect = new Rectangle(Int32.Parse(args[0]),Int32.Parse(args[1]));
      int emvadon,perimetros;

      emvadon = rect.RectInfo(out perimetros);

      Console.WriteLine("Το ορθογώνιο έχει εμβαδόν "+emvadon+" και περίμετρο "+perimetros);

      Console.ReadKey();
```

```
    }
}
```

6.1.3

μεταβίβαση αναφοράς αντικειμένου

σε μία μέθοδο μπορούμε να μεταβιβάσουμε και ***μεταβλητές αντικειμένου***, δηλαδή στην ουσία μία αναφορά αντικειμένου. Σε αυτή τη περίπτωση, *μεταβιβάζουμε ένα αντίγραφο της αναφοράς* και οι αλλαγές που γίνονται σε αυτή δεν επηρεάζουν την αναφορά-όρισμα. Όμως, **και οι 2 αυτές αναφορές "δείχνουν" στην ίδια θέση μνήμης, δηλαδή οι αλλαγές θα επηρεάσουν το αντικείμενο τελικά.**
Για παράδειγμα, στο παρακάτω πρόγραμμα οι τιμές των μεταβλητών a και β θα αλλάξουν τιμή, αφού τις μεταβιβάζουμε στη μέθοδο Change() με χρήση του αντικειμένου ob.

παράδειγμα

```
using System;

class Test
{
   public int a, b;

   public Test(int i, int j)
   {
      a = i;
      b = j;
   }

   /* Pass an object. Now, ob.a and ob.b in object
     used in the call will be changed. */
   public void Change(Test ob)
   {
      ob.a = ob.a + ob.b;
      ob.b = -ob.b;
   }
}

class CallByRef
{
   static void Main()
   {
      Test ob = new Test(15, 20);
      Console.WriteLine("ob.a and ob.b before call: " + ob.a + " " + ob.b);

      ob.Change(ob);
      Console.WriteLine("ob.a and ob.b after call: " + ob.a + " " + ob.b);

      Console.ReadKey();
   }
}
```

6.1.4

μεταβίβαση αναφοράς

μεταβλητών τύπου αναφοράς

Οι τροποποιητές **ref** και **out** μπορούν να χρησιμοποιηθούν και σε ***μεταβλητές τύπου αναφοράς***. Σε αυτή τη περίπτωση, η μέθοδος επηρεάζει την αναφορά στην οποία αναφέρεται.

παράδειγμα

```
using System;

class Test
{
   public int a;

   public Test(int i)
   {
      a = i;
```

```
    }
    // This will not change the argument.
    public void NoChange(Test o)
    {
        Test newob = new Test(0);
        o = newob; // this has no effect outside of NoChange()
    }

    // This will change what the argument refers to.
    public void Change(ref Test o)
    {
        Test newob = new Test(0);
        o = newob; // this affects the calling argument.
    }
}

class CallObjByRef
{
    static void Main()
    {
        Test ob = new Test(100);
        Console.WriteLine("ob.a before call: " + ob.a);

        ob.NoChange(ob);
        Console.WriteLine("ob.a after call to NoChange(): " + ob.a);

        ob.Change(ref ob);
        Console.WriteLine("ob.a after call to Change(): " + ob.a);

        Console.ReadKey();
    }
}
```

6.2

Υπερφόρτωση

6.2.1

υπερφόρτωση μεθόδων

Στη C#, μέσα σε μία οποιαδήποτε κλάση, 2 ή περισσότερες μέθοδοι ***μπορούν να έχουν το ίδιο όνομα,*** αρκεί οι δηλώσεις των παραμέτρων τους να είναι *διαφορετικές*, έτσι ώστε να μπορεί ο μεταγλωττιστής να τις διαχωρίσει. Οι μέθοδοι αυτές λέγονται ***υπερφορτωμένες (πολυμορφισμός)***.

παράδειγμα

```
using System;

class Overload
{
    public void OvlDemo()
    {
        Console.WriteLine("No parameters");
    }

    // Overload OvlDemo for one integer parameter.
    public void OvlDemo(int a)
    {
        Console.WriteLine("One parameter: " + a);
    }

    // Overload OvlDemo for two integer parameters.
    public int OvlDemo(int a, int b)
    {
        Console.WriteLine("Two parameters: " + a + " " + b);
        return a + b;
    }

    // Overload OvlDemo for two double parameters.
    public double OvlDemo(double a, double b)
```

```
    {
        Console.WriteLine("Two double parameters: " + a + " " + b);
        return a + b;
    }
}

class OverloadDemo
{
    static void Main()
    {
        Overload ob = new Overload();
        int resI;
        double resD;

        // Call all versions of OvlDemo().
        ob.OvlDemo();
        Console.WriteLine();

        ob.OvlDemo(2);
        Console.WriteLine();

        resI = ob.OvlDemo(4, 6);
        Console.WriteLine("Result of ob.OvlDemo(4, 6): " + resI);
        Console.WriteLine();


        resD = ob.OvlDemo(1.1, 2.32);
        Console.WriteLine("Result of ob.OvlDemo(1.1, 2.2): " + resD);

        Console.ReadKey();
    }
}
```

6.2.2

υπερφόρτωση κατασκευαστών

εκτός από την υπερφόρτωση των απλών μεθόδων, μπορούμε να κάνουμε και ***υπερφόρτωση των κατασκευαστών μίας κλάσης***. Έτσι, μπορούμε να δημιοουργούμε αντικείμενα με διάφορους τρόπους.

παράδειγμα

```
using System;

class MyClass
{
    public int x;

    public MyClass()
    {
        Console.WriteLine("Inside MyClass().");
        x = 0;
    }

    public MyClass(int i)
    {
        Console.WriteLine("Inside MyClass(int).");
        x = i;
    }

    public MyClass(double d)
    {
        Console.WriteLine("Inside MyClass(double).");
        x = (int)d;
    }

    public MyClass(int i, int j)
    {
        Console.WriteLine("Inside MyClass(int, int).");
        x = i * j;
    }
}

class OverloadConsDemo
{
```

```
   static void Main()
   {
      MyClass t1 = new MyClass();
      MyClass t2 = new MyClass(88);
      MyClass t3 = new MyClass(17.23);
      MyClass t4 = new MyClass(2, 4);

      Console.WriteLine("t1.x: " + t1.x);
      Console.WriteLine("t2.x: " + t2.x);
      Console.WriteLine("t3.x: " + t3.x);
      Console.WriteLine("t4.x: " + t4.x);

      Console.ReadKey();
   }
}
```

6.2.3

κλήση υπερφορτωμένου κατασκευαστή

μέσω του this

όταν υπερφορτώνουμε κατασκευαστές έχουμε τη δυνατότητα ***ο ένας κατασκευαστής να καλεί τον άλλον***. Η γενική σύνταξη είναι η εξής:

όνομα_κατασκευαστή(παράμετροι-1) **: this**(*παράμετροι-2*){
//κώδικας
}

Όταν καλείται ένας τέτοιος κατασκευαστής, εκτελείται ***πρώτα*** ο κατασκευαστής που συμφωνεί με την λίστα παραμέτρων-2. Στη συνέχεια, αν υπάρχει κώδικας και για τον κατασκευαστή με τη λίστα παραμέτρων-1 εκτελείται και αυτός.

παράδειγμα

```
using System;

class XYCoord
{
   public int x, y;

   public XYCoord() : this(0, 0)
   {
      Console.WriteLine("Inside XYCoord()");
   }

   public XYCoord(XYCoord obj) : this(obj.x, obj.y)
   {
      Console.WriteLine("Inside XYCoord(XYCoord obj)");
   }

   public XYCoord(int i, int j)
   {
      Console.WriteLine("Inside XYCoord(XYCoord(int, int)");
      x = i;
      y = j;
   }
}

class OverloadConsDemo
{
   static void Main()
   {
      XYCoord t1 = new XYCoord();
      XYCoord t2 = new XYCoord(8, 9);
      XYCoord t3 = new XYCoord(t2);

      Console.WriteLine("t1.x, t1.y: " + t1.x + ", " + t1.y);
      Console.WriteLine("t2.x, t2.y: " + t2.x + ", " + t2.y);
      Console.WriteLine("t3.x, t3.y: " + t3.x + ", " + t3.y);

      Console.ReadKey();
   }
}
```

6.3

Ερωτήσεις κατανόησης

1 Είναι σωστός ο παρακάτω κώδικας;

```
class X{

        int meth(int a,int b){...}
        string meth(int a,int b){...}
```

2 Τι κάνουν οι ref και out; σε τι διαφέρουν;

3 Είναι σωστός ο παρακάτω κώδικας;

```
using System;

class Dokimes
{
   int metavliti;

   public void sunarthsh(ref int i)
   {
      i = i * 2;
   }
}
class Program
{
   static void Main()
   {
      int temp;
      Dokimes x = new Dokimes();

      x.sunarthsh(ref temp);

      Console.WriteLine(x.metavliti);

      Console.ReadKey();
   }
}
```

4 Είναι σωστός ο παρακάτω κώδικας;

```
using System;

class Dokimes
{
   int metavliti;

   public void sunarthsh(out int i)
   {
      i = i + 50;
   }

}

class Program
{
   static void Main()
   {
      int temp=20;
      Dokimes x = new Dokimes();

      x.sunarthsh(out temp);

      Console.WriteLine(temp);

      Console.ReadKey();
```

```
  }
}
```

5 Θέλουμε να δηλώσουμε έναν κατασκευαστή που θα καλεί τον ακόλουθο κατασκευαστή:

```
public Dokimes(string x)
{
   Console.WriteLine("hello" + x);
}
```

Γράψτε τον απαραίτητο κώδικα .

6.4

Άσκηση

Κατεβάσε το αρχείο Lab6Exercise.cs από το e-class. Συμπληρώστε κατάλληλα την κλάση Romvos.

7

Στατικά μέλη (static) & Ιδιότητες (get,set)

Τι θα δούμε σε αυτό το μάθημα

1. Στατικά μέλη κλάσης
    1. Ορισμός/δήλωση στατικού μέλους
    2. Κλήση στατικού μέλους
    3. χρήση στατικού μέλους
    4. περιορισμοί στατικής μεθόδου
    5. Στατικοί κατασκευαστές
2. Ιδιότητες get,set
    1. Ορισμός ιδιότητας
    2. Λειτουργία ιδιότητας
    3. Αυτόματα υλοποιούμενες ιδιότητες
3. Παράδειγμα
4. Άσκηση
5. Ερωτήσεις κατανόησης

7.1

Στατικά μέλη κλάσης

7.1.1

ορισμός/δήλωση στατικού μέλους

Συνήθως, κάθε μέλος κλάσης καλείται μέσω ενός *αντικειμένου* της ίδιας κλάσης. Θα μπορούσαμε, όμως, να ορίσουμε κάποιο μέλος της κλάσης έτσι ώστε να χρησιμοποιείται από μόνο του, δηλαδή *χωρίς την ανάγκη ενός αντικειμένου*. Για να το πετύχουμε αυτό, πρέπει να εισάγουμε στη δήλωση τη λέξη-κλειδί static:

```
class MyClass{
public static int x=0;
}
```

7.1.2

κλήση στατικού μέλους

Όταν ένα μέλος είναι static τότε αυτό μπορεί να προσπελαστεί πριν δημιουργήσουμε αντικείμενα της κλάσης του και φυσικά χωρίς τη χρήση αντικειμένου αλλά με απλή χρήση της κλάσης:

MyClass.x=100;

Αν ένα μέλος static είναι private, τότε μπορούμε να δημιουργήσουμε static μεθόδους get και set για να έχουμε πρόσβαση σε αυτό εξωτερικά (ή με static ιδιότητες get, set).

παράδειγμα

```
using System;

class StaticDemo
{
   // A static variable.
   public static int Val = 100;

   // A static method.
   public static int ValDiv2()
   {
      return Val / 2;
   }
}

class SDemo
{
   static void Main()
   {
      Console.WriteLine("Initial value of StaticDemo.Val is " + StaticDemo.Val);

      StaticDemo.Val = 8;
      Console.WriteLine("StaticDemo.Val is " + StaticDemo.Val);
      Console.WriteLine("StaticDemo.ValDiv2(): " + StaticDemo.ValDiv2());

      Console.ReadKey();
   }
}
```

Αυτό το πρόγραμμα εκτυπώνει:

```
Initial value of StaticDemo.Val is 100
StaticDemo.Val is 8
StaticDemo.ValDiv2(): 4
```

7.1.3

χρήση ενός στατικού μέλους

όταν δηλώνουμε ένα μέλος κλάσης ως στατικό, στην ουσία το κάνουμε **καθολικό**: για όλα τα αντικείμενα που θα δημιουργήσουμε, ***το στατικό μέλος είναι κοινό***.
Επίσης, πρέπει όλες οι στατικές μεταβλητές να έχουν **αρχική τιμή** (αν δεν το κάνουμε ρητά εμείς το κάνει ο μεταγλωττιστής με τις default τιμές).

7.1.4

περιορισμοί μίας στατικής μεθόδου

- μία μέθοδος static δεν έχει αναφορά this.
- μία μέθοδος static μπορεί να καλεί απευθείας μόνο άλλες μεθόδους static της κλάσης της.
- μία μέθοδος static μπορεί να προσπελαύνει απευθείας μόνο άλλα δεδομένα static της κλάσης της.

για παράδειγμα, ο παρακάτω κώδικας δεν θα περάσει από τον έλεγχο, γιατί η στατική μέθοδος ValDivDenom() προσπαθεί να χρησιμοποιήσει την μή στατική (στιγμιοτύπου) μεταβλητή Denom:

```
class StaticError
{
   public int Denom = 3; // a normal instance variable
   public static int Val = 1024; // a static variable

   /* Error! Can't directly access a non-static variable
      from within a static method. */
   public static int ValDivDenom()
   {
      return Val / Denom; // won't compile!
   }
}
```

7.1.5

Στατικοί κατασκευαστές

είναι δυνατό να δηλώσουμε έναν κατασκευαστή ως static. Ένας τέτοιος κατασκευαστής *καλείται αυτόματα όταν η κλάση φορτώνεται για πρώτη φορά*, πριν καν δημιουργηθούν αντικείμενα και φυσικά πριν κληθούν οι κατασκευαστές στιγμιοτύπου. Αρμοδιότητά του είναι να *αρχικοποιεί χαρακτηριστικά τύπου static* και όχι στιγμιοτύπου. **Δεν περιέχει τροποποιητές προσπέλασης και δεν μπορεί να κληθεί απευθείας από το πρόγραμμα**, πχ:

```
class Paradeigma
{
   public int met1 = 3;
   public static int met2 = 1024;

   static Paradeigma(int y)
   {
      met2 = y;
   }
}
```

7.2

Ιδιότητες get, set

7.2.1

ορισμός ιδιότητας

Σε μία κλάση, εκτός από μεταβλητές και μεθόδους, μπορούμε να δηλώσουμε ως μέλος και μία ***ιδιότητα***. *Μία ιδιότητα συνδυάζει μία μεταβλητή μέλος με δύο ειδικές μεθόδους προσπέλασής της και δεν δεσμεύει χώρο αποθήκευσης.*
Μία ιδιότητα αποτελείται από ένα *όνομα* και τους προσπελαστές ***get*** και ***set***.
Η γενική της μορφή φαίνεται παρακάτω:

τύπος **όνομα**{
get {

```
    //κώδικας
 }

set {
    //κώδικας
   }
}
```

7.2.2 λειτουργία ιδιότητας

Κάθε χρήση του ονόματος της ιδιότητας έχει ως αποτέλεσμα μία κλήση του κατάλληλου προσπελαστή:

- ο προσπελαστής **set** *δέχεται αυτόματα μία παράμετρο*, η τιμή της οποίας αποθηκεύεται στην default μεταβλητή **value** και η οποία περιέχει την τιμή που ανατίθεται στην ιδιότητα.
- ο προσπελαστής **get** *επιστρέφει την τιμή της μεταβλητής* που διαχειρίζεται και ο τύπος της πρέπει να συμφωνεί με τον τύπο της ιδιότητας.

παράδειγμα

```
using System;

class SimpProp
{
   int prop;

   public SimpProp()
   {
      prop = 0;
   }

   public int MyProp
   {
      get
      {
         return prop;
      }
      set
      {
         if (value >= 0) prop = value;
      }
   }
}

class PropertyDemo
{
   static void Main()
   {
      SimpProp ob = new SimpProp();

      Console.WriteLine("Original value of ob.MyProp: " + ob.MyProp);

      ob.MyProp = 100;
      Console.WriteLine("Value of ob.MyProp: " + ob.MyProp);

      Console.WriteLine("Attempting to assign -10 to ob.MyProp");
      ob.MyProp = -10;
      Console.WriteLine("Value of ob.MyProp: " + ob.MyProp);

      Console.ReadKey();
   }
}
```

Το συγκεκριμένο πρόγραμμα εκτυπώνει τα εξής:

```
Original value of ob.MyProp: 0
Value of ob.MyProp: 100
Attempting to assign -10 to ob.MyProp
Value of ob.MyProp: 100
```

7.2.3

αυτόματα υλοποιούμενες ιδιότητες

Είναι οι ιδιότητες στις οποίες *δεν ορίζουμε ρητά την μεταβλητή* που θα διαχειρίζεται η ιδιότητα. Ο μεταγλωττιστής αποφασίζει ποια μεταβλητή θα επηρεάσει. Η γενική μορφή αυτών των ιδιοτήτων είναι η εξής:

τύπος όνομα {get; set;}

Ενώ στις κανονικές ιδιότητες δεν είμαστε υποχρεωμένοι να γράψουμε και την set και την get, στις αυτόματες ιδιότητες πρέπει να υπάρχουν *και οι δύο*.

7.3

Παράδειγμα

Να γίνει πρόγραμμα το οποίο θα ζητάει από τον χρήστη τη πλευρά ενός τετραγώνου και θα εκτυπώνει το εμβαδόν του τετραγώνου. Στο τέλος θα πρέπει να εκτυπώνει το πλήθος των τετραγώνων για τα οποία υπολογίστηκε το εμβαδόν.

Program.cs

```
using System;

namespace Lab7
{
   class Program
   {
      static void Main()
      {
         string apantisi;

         do
         {
            Console.Write("Δώσε το μήκος της πλευράς του τετραγώνου:");
            Tetragwno sxhma = new Tetragwno(Double.Parse(Console.ReadLine()));

            Console.WriteLine("Το εμβαδόν του τετραγώνου με πλευρά {0:.0} είναι {1:.0}", sxhma.Pleura, sxhma.YpologismosEmvadou());

            Console.Write("Θα συνεχίσεις; ");
            apantisi = Console.ReadLine();

         } while (apantisi != "no");

         Console.WriteLine("Σύνολο σχημάτων:" + Tetragwno.GetCount());

         Console.ReadKey();
      }
   }
}
```

Tetragwno.cs

```
using System;

namespace Lab7
{
   class Tetragwno
   {
      //μεταβλητές
```

```
    private double pleura;
    private static int count;

    //κατασκευαστές
    static Tetragwno()
    {
       count=0;
    }

    public Tetragwno():this (0.0)
    {
    }

    public Tetragwno(double pleura)
    {
       count++;
       this.pleura = pleura;
    }

    //ιδιότητες
    public double Pleura
    {
       set
       {
          pleura = value;
       }
       get
       {
          return pleura;
       }
    }

    //μέθοδοι
    public static int GetCount()
    {
       return count;
    }

    public double YpologismosEmvadou()
    {
       return Math.Pow(pleura, 2.0);
    }

  }
}
```

7.4

Άσκηση

Κατεβάστε το αρχείο Lab7AskisiStart.zip . Αποσυμπιέστε στην επιφάνεια εργασίας και συμπληρώστε τον κώδικα της κλάσης Car που βρίσκεται στο αρχείο Car.cs

7.5

Ερωτήσεις κατανόησης

1 Τι θα εκτυπώσει το παρακάτω πρόγραμμα;

```
using System;

class metrStatic
{
   static int metr;
```

```
    public metrStatic()
    {
        metr = 3;
    }

    public void ayxise(int k)
    {
        metr = metr + k;
    }

    public static void ayxiseSt(int k)
    {
        metr = metr + k;
    }

    public int getMetr()
    {
        return metr;
    }

    public static int getMetrSt()
    {
        return metr;
    }
}

class Program
{
    static void Main()
    {
        metrStatic.ayxiseSt(15);
        metrStatic m = new metrStatic();
        m.ayxise(18);
        Console.WriteLine(m.getMetr());
        Console.WriteLine(metrStatic.getMetrSt());
    }
}
```

2 Τι θα εκτυπώσει το παρακάτω πρόγραμμα;

```
using System;

class metrStatic
{
    static int metr;

    static metrStatic()
    {
        metr = 3;
    }

    public void ayxise(int k)
    {
        metr = metr + k;
    }

    public static void ayxiseSt(int k)
    {
        metr = metr + k;
    }

    public int getMetr()
    {
        return metr;
    }

    public static int getMetrSt()
    {
        return metr;
    }
}
```

```
class Program
{
   static void Main()
   {
      metrStatic.ayxiseSt(15);
      metrStatic m = new metrStatic();
      m.ayxise(18);
      Console.WriteLine(m.getMetr());
      Console.WriteLine(metrStatic.getMetrSt());
   }
}
```

3

```
class metritis
{
   static int metr=23;
   public static metritis(int k)
   {  metr=k;  }
}
```

Ποια θα είναι η *αρχική τιμή του πεδίου metr* όταν κατασκευαστεί το πρώτο αντικείμενο; Υπάρχει λάθος στον κώδικα;

4 Τι θα εκτυπώσει το παρακάτω πρόγραμμα;

```
using System;

class metrStatic
{
   static int metr = 17;

   public metrStatic()
   {
      metr = 3;
   }

   static metrStatic()
   {
      metr = 13;
   }

   public void ayxise(int k)
   {
      metr = metr + k;
   }

   public static void ayxiseSt(int k)
   {
      metr = metr + k;
   }

   public int getMetr()
   {
      return metr;
   }

   public static int getMetrSt()
   {
      return metr;
   }
}

class Program
{
   static void Main()
   {
      metrStatic.ayxiseSt(15);
      metrStatic m = new metrStatic();
      m.ayxise(18);
      Console.WriteLine(m.getMetr());
      Console.WriteLine(metrStatic.getMetrSt());
   }
}
```

5 Υπάρχει λάθος στον παρακάτω κώδικα;

```
using System;

class metrStatic
{
   static int metr = 17;
   int x;

   public metrStatic()
   {
      metr = 3;
      x = 9;
   }

   static metrStatic()
   {
      metr = 13;
      x = 9;
   }

   public void ayxise(int k)
   {
      metr = metr + k;
   }

   public static int Metr
   {
      set
      {
         metr = metr + value;
      }
      get
      {
         return metr;
      }
   }
}

class Program
{
   static void Main()
   {
      metrStatic m = new metrStatic();
      m.ayxise(18);
      Console.WriteLine(metrStatic.Metr);
   }
}
```

8

Κληρονομικότητα

Τι θα δούμε σε αυτό το μάθημα

1: Γενικά

1.1: η έννοια της κληρονομικότητας

1.2: σύνταξη παραγόμενης κλάσης

2: Προσπέλαση και κληρονομικότητα μελών

2.1: προσπέλαση μελών κλάσης-βάσης

2.2: κατασκευαστές & κληρονομικότητα

3: Κληρονομικότητα & απόκρυψη ονόματος

3.1: τι είναι απόκρυψη ονόματος

3.2: προσπέλαση κρυμμένου ονόματος

4: Εικονικές μέθοδοι & υπερκάλυψη

4.1:τι είναι εικονική μέθοδος & πως ορίζουμε την υπερκάλυψη μεθόδου

5: αφαιρετικές κλάσεις και μέθοδοι

5.1: αφαιρετική μέθοδος

5.2: αφαιρετική κλάση

6: Άσκηση

8.1

Γενικά

8.1.1

Η έννοια της κληρονομικότητας

Η κληρονομικότητα είναι βασικό χαρακτηριστικό της C#. Έτσι, μπορούμε να δημιουργήσουμε μία *γενική κλάση* ***(κλάση βάσης)***, η οποία ορίζει κάποια γενικά χαρακτηριστικά (*μεταβλητές, μεθόδους, ιδιότητες* κτλ) και στη συνέχεια, όλα αυτά τα χαρακτηριστικά *να κληρονομούνται σε άλλες κλάσεις* ***(παραγόμενες κλάσεις)***, οι οποίες με τη σειρά τους να *προσθέτουν* στα γενικά χαρακτηριστικά της κλάσης-βάσης τα δικά τους, μοναδικά στοιχεία. Έτσι, μπορούμε να πούμε ότι η κάθε παραγόμενη κλάση είναι μία ειδική έκδοση της κλάσης-βάσης.

Σημείωση: κάθε παραγόμενη κλάση μπορεί να έχει *μία και μόνο μία κλάση-βάσης* (σε αντίθεση με τη C++).

8.1.2

σύνταξη παραγόμενης κλάσης

Όταν θέλουμε να δηλώσουμε ότι μία κλάση κληρονομεί τα χαρακτηριστικά μίας κλάσης-βάσης, αρκεί στη ***δήλωση της παραγόμενης κλάσης*** να δηλώσουμε την **κλάση-βάσης** με τον εξής τρόπο:

```
class όνομα_παραγόμενης_κλάσης : όνομα_κλάσης_βάσης
{
...
}
```

παράδειγμα

```
using System;


class TwoDShape
{
   public double Width;
   public double Height;

   public void ShowDim()
   {
      Console.WriteLine("Width and height are " + Width + " and " + Height);
   }
}

class Triangle : TwoDShape
{
   public string Style;

   public double Area()
   {
      return Width * Height / 2;
   }

   public void ShowStyle()
   {
      Console.WriteLine("Triangle is " + Style);
   }
}

class Shapes
{
   static void Main()
```

```
  {
    Triangle t1 = new Triangle();
    Triangle t2 = new Triangle();

    t1.Width = 4.0;
    t1.Height = 4.0;
    t1.Style = "isosceles";

    t2.Width = 8.0;
    t2.Height = 12.0;
    t2.Style = "right";

    Console.WriteLine("Info for t1: ");
    t1.ShowStyle();
    t1.ShowDim();
    Console.WriteLine("Area is " + t1.Area());

    Console.WriteLine();

    Console.WriteLine("Info for t2: ");
    t2.ShowStyle();
    t2.ShowDim();
    Console.WriteLine("Area is " + t2.Area());
    TwoDShape shape = new TwoDShape();

    shape.Width = 10;
    shape.Height = 20;

    shape.ShowDim();
    Console.ReadKey();
  }
}
```

Στο συγκεκριμένο παράδειγμα, η κλάση-βάσης είναι η TwoDShape ενώ η κλάση που κληρονομεί από αυτή είναι η Triangle (παραγόμενη κλάση). Έτσι κληρονομεί τις μεταβλητές Width και Height, καθώς και τη μέθοδο ShowDim() (μπορεί δηλαδή να τις χρησιμοποιήσει). Στη Main() ορίζουμε 2 αντικείμενα της παραγόμενης κλάσης και 1 αντικείμενο της κλάσης βάσης.
Το πρόγραμμα θα εκτυπώσει τα εξής:

```
Info for t1:
Triangle is isosceles
Width and height are 4 and 4
Area is 8

Info for t2:
Triangle is right
Width and height are 8 and 12
Area is 48
Width and height are 10 and 20
```

8.2

Προσπέλαση & κληρονομικότητα μελών

8.2.1 προσπέλαση μελών κλάσης-βάσης

Γενικά, μία παραγόμενη κλάση μπορεί να κληρονομήσει οποιοδήποτε χαρακτηριστικό της κλάση-βάσης το οποίο ***δεν είναι δηλωμένο ως private***. Τα private χαρακτηριστικά είναι ορατά μόνο μέσα στην κλάση που είναι δηλωμένα.
Άρα υπάρχουν 3 περιπτώσεις:

1. **public** χαρακτηριστικό της κλάσης βάσης: ***μπορούμε*** να έχουμε πρόσβαση σε αυτό από άλλη ανεξάρτητη κλάση **&** ***κληρονομείται*** το χαρακτηριστικό αυτό σε οποιαδήποτε παραγόμενη κλάση.
2. **private** χαρακτηριστικό της κλάσης βάσης: ***δεν μπορούμε*** να έχουμε πρόσβαση σε αυτό από άλλη ανεξάρτητη κλάση **&** ***δεν κληρονομείται*** το χαρακτηριστικό αυτό σε οποιαδήποτε παραγόμενη κλάση.
3. **protected** χαρακτηριστικό της κλάσης βάσης: ***δεν μπορούμε*** να έχουμε

πρόσβαση σε αυτό από άλλη ανεξάρτητη κλάση **αλλά *κληρονομείται*** το χαρακτηριστικό αυτό σε οποιαδήποτε παραγόμενη κλάση.

παράδειγμα

```
using System;

class B
{
   protected int i, j;

   public void Set(int a, int b)
   {
      i = a;
      j = b;
   }

   public void Show()
   {
      Console.WriteLine(i + " " + j);
   }
}

class D : B
{
   int k;
   public void SetK()
   {
      k = i * j;
   }

   public void Showk()
   {
      Console.WriteLine(k);
   }
}

class ProtectedDemo
{
   static void Main()
   {
      D ob = new D();

      ob.Set(2, 3); // OK, known to D
      ob.Show();    // OK, known to D

      ob.SetK();  // OK, part of D
      ob.Showk(); // OK, part of D

      Console.ReadKey();
   }
}
```

Αυτό το πρόγραμμα θα εκτυπώσει τα ακόλουθα:

```
2 3
6
Press any key to continue . . .
```

8.2.2

Κατασκευαστές & κληρονομικότητα

Όταν έχουμε κλάση-βάσης και παραγόμενες κλάσεις είναι δυνατόν να έχουμε *κατασκευαστές* και στις παραγόμενες κλάσεις. Αυτοί οι κατασκευαστές μπορούν να *αρχικοποιούν δημόσιες ή προστατευμένες μεταβλητές, τόσο δικές τους όσο και την κλάσης-βάσης.* Αν, όμως, η παραγόμενη κλάση **καλεί τον κατασκευαστή της κλάσης-βάσης** τότε για να του περάσει τα ορίσματα που χρειάζεται χρησιμοποιεί την λέξη-κλειδί **base**. Ιδού η σύνταξη:

παραγόμενος_κατασκευαστής(λίστα_παραμέτρων) : base(λίστα ορισμάτων){
//κώδικας
}

Ο παραγόμενος κατασκευαστής μεταβιβάζει στον κατασκευαστή της κλάσης-βάσης κάποιες από τις παραμέτρους του ως ορίσματα και ο αντίστοιχος κατασκευαστής εκτελείται. Στη συνέχεια, εκτελείται και κώδικας του παραγόμενου κατασκευαστή.

παράδειγμα

```
using System;

class TwoDShape
{
   double pri_width;
   double pri_height;

   // Default constructor.
   public TwoDShape()
   {
      Width = Height = 0.0;
   }

   // Specify Width and Height.
   public TwoDShape(double w, double h)
   {
      Width = w;
      Height = h;
   }

   // Construct object with equal width and height.
   public TwoDShape(double x)
   {
      Width = Height = x;
   }

   // Properties for Width and Height.
   public double Width
   {
      get { return pri_width; }
      set { pri_width = value < 0 ? -value : value; }
   }

   public double Height
   {
      get { return pri_height; }
      set { pri_height = value < 0 ? -value : value; }
   }

   public void ShowDim()
   {
      Console.WriteLine("Width and height are " + Width + " and " + Height);
   }
}
```

```
// A derived class of TwoDShape for triangles.
class Triangle : TwoDShape
{
   string Style;

   public Triangle()
   {
      Style = "null";
   }

   public Triangle(string s, double w, double h) : base(w, h)
   {
      Style = s;
   }

   // Construct an isosceles triangle.
   public Triangle(double x) : base(x)
   {
      Style = "isosceles";
   }

   public double Area()
   {
      return Width * Height / 2;
   }

   public void ShowStyle()
   {
      Console.WriteLine("Triangle is " + Style);
   }
}

class Shapes5
{
   static void Main()
   {
      Triangle t1 = new Triangle();
      Triangle t2 = new Triangle("right", 8.0, 12.0);
      Triangle t3 = new Triangle(4.0);

      t1 = t2;

      Console.WriteLine("Info for t1: ");
      t1.ShowStyle();
      t1.ShowDim();
      Console.WriteLine("Area is " + t1.Area());

      Console.WriteLine();

      Console.WriteLine("Info for t2: ");
      t2.ShowStyle();
      t2.ShowDim();
      Console.WriteLine("Area is " + t2.Area());

      Console.WriteLine();

      Console.WriteLine("Info for t3: ");
      t3.ShowStyle();
      t3.ShowDim();
      Console.WriteLine("Area is " + t3.Area());

      Console.WriteLine();

      Console.ReadKey();
   }
}
```

Το πρόγραμμα θα εκτυπώσει τα εξής:

```
Info for t1:
Triangle is right
Width and height are 8 and 12
Area is 48

Info for t2:
Triangle is right
Width and height are 8 and 12
Area is 48

Info for t3:
Triangle is isosceles
Width and height are 4 and 4
Area is 8
```

8.3

Κληρονομικότητα & απόκρυψη ονόματος

8.3.1

τι είναι απόκρυψη ονόματος

είναι δυνατόν μία παραγόμενη κλάση να ορίζει ένα μέλος, το οποίο *έχει το **ίδιο όνομα** μ'ένα μέλος της κλάσης-βάσης*. Όταν συμβαίνει αυτό, το μέλος της κλάσης βάσης **κρύβεται** μέσα στη παραγόμενη κλάση. Για να μην μας εμφανίσει προειδοποίηση ο μεταγλωττιστής θα πρέπει να εισάγουμε τη λέξη κλειδί **new** πριν το μέλος της παραγόμενης κλάσης, πχ:

```
using System;

class A
{
   public int i = 0;
}

class B : A
{
   new int i;

   public B(int b)
   {
      i = b;
   }

   public void Show()
   {
      Console.WriteLine("i in derived class: " + i);
   }
}

class NameHiding
{
   static void Main()
   {
      B ob = new B(2);

      ob.Show();
   }
}
```

8.3.2

προσπέλαση κρυμμένου ονόματος

Όταν Θέλουμε να έχουμε πρόσβαση σε ένα μέλος μία κλάσης βάσης *το οποίο είναι **κρυμμένο***, τότε χρησιμοποιούμε τη λέξη κλειδί **base** με την ακόλουθη μορφή:

base.μέλος

πχ

```
using System;
```

```
class A
{
   public int i = 0;
}

class B : A
{
   new int i;

   public B(int a, int b)
   {
      base.i = a;
      i = b;
   }

   public void Show()
   {

      Console.WriteLine("i in base class: " + base.i);

      Console.WriteLine("i in derived class: " + i);
   }
}

class UncoverName
{
   static void Main()
   {
      B ob = new B(1, 2);

      ob.Show();
   }
}
```

παράδειγμα

```
using System;

class Vehicle
{
   public int Passengers { get; protected set; }

   public int FuelCap { get; protected set; }

   public int Mpg { get; protected set; }

   // This is a constructor for Vehicle.
   public Vehicle(int p, int f, int m)
   {
      Passengers = p;
      FuelCap = f;
      Mpg = m;
   }

   // Return the range.
   public int Range()
   {
      return Mpg * FuelCap;
   }

   // Compute fuel needed for a given distance.
   public double FuelNeeded(int miles)
   {
      return (double)miles / Mpg;
   }

}

class Truck : Vehicle
{
```

```
    // Auto-implemented property for cargo capacity in pounds.
    public int CargoCap { get; protected set; }

    // This is a constructor for Truck.
    public Truck(int p, int f, int m, int c) : base(p, f, m)
    {
      CargoCap = c;
    }

}

class TruckDemo
{
  static void Main()
  {
    // Construct some trucks.
    Truck semi = new Truck(2, 200, 7, 44000);
    Truck pickup = new Truck(3, 28, 15, 2000);

    double gallons;
    int dist = 252;

    gallons = semi.FuelNeeded(dist);

    Console.WriteLine("Semi can carry " + semi.CargoCap + " pounds.");
    Console.WriteLine("To go " + dist + " miles semi needs " + gallons + " gallons of fuel.\n");

    gallons = pickup.FuelNeeded(dist);

    Console.WriteLine("Pickup can carry " + pickup.CargoCap + " pounds.");
    Console.WriteLine("To go " + dist + " miles pickup needs " + gallons + " gallons of fuel.");

    Console.ReadKey();
  }
}
```

8.4

Εικονικές μέθοδοι & υπερκάλυψη

8.4.1 τι είναι εικονική μέθοδος & πως ορίζουμε την υπερκάλυψη μεθόδου

είναι η μέθοδος που μπορεί να *ανακαθορίζεται σε μία ή περισσότερες παραγόμενες κλάσεις*, ενώ υπάρχει στην κλάση βάσης. Σε αυτή τη περίπτωση λέμε ότι η κάθε παραγόμενη κλάση μπορεί να έχει τη δική της έκδοση της εικονικής μεθόδου.
Για να δηλώσουμε μία μέθοδο ως εικονική στην κλάση βάσης, θα πρέπει να εισάγουμε τη λέξη **virtual** πριν από τη δήλωσή της στη κλάση βάσης, ενώ στη παραγόμενη κλάση θα πρέπει να εισάγουμε τη λέξη **override**. Η όλη διαδικασία ονομάζεται **υπερκάλυψη μεθόδου.**

Σημείωση:

- όταν υπερκαλύπτουμε μία μέθοδο το *όνομα*, ο *τύπος* και τα *ορίσματα* της μεθόδου πρέπει να είναι ακριβώς τα ίδια με την αρχική μέθοδο.
- μία εικονική μέθοδος δεν μπορεί να είναι **static** ή **abstract**.
- αν μία από τις παραγόμενες κλάσεις *δεν υπερκαλύπτει* μία εικονική μέθοδο, τότε χρησιμοποιείται η μέθοδος της κλάσης βάσης.
- μπορούμε να έχουμε και *εικονικές ιδιότητες*.

παράδειγμα

```
using System;
```

```
class Base
{
   // Create virtual method in the base class.
   public virtual void Who()
   {
      Console.WriteLine("Who() in Base");
   }
}

class Derived1 : Base
{
   // Override Who() in a derived class.
   public override void Who()
   {
      Console.WriteLine("Who() in Derived1");
   }
}

class Derived2 : Base
{
   // Override Who() again in another derived class.
   public override void Who()
   {
      Console.WriteLine("Who() in Derived2");
   }
}

class OverrideDemo
{
   static void Main()
   {
      Base baseOb = new Base();
      Derived1 dOb1 = new Derived1();
      Derived2 dOb2 = new Derived2();

      Base baseRef;

      baseRef = baseOb;
      baseRef.Who();

      baseRef = dOb1;
      baseRef.Who();

      baseRef = dOb2;
      baseRef.Who();
   }
}
```

παράδειγμα

```
using System;

class TwoDShape
{
   double pri_width;
   double pri_height;

   // A default constructor.
   public TwoDShape()
   {
      Width = Height = 0.0;
      Name = "null";
   }

   // Specify all information.
   public TwoDShape(double w, double h, string n)
   {
```

```
        Width = w;
        Height = h;
        Name = n;
    }

    // Construct object with equal width and height.
    public TwoDShape(double x, string n)
    {
        Width = Height = x;
        Name = n;
    }

    // Construct a copy of a TwoDShape object.
    public TwoDShape(TwoDShape ob)
    {
        Width = ob.Width;
        Height = ob.Height;
        Name = ob.Name;
    }

    // Properties for Width, Height, and Name.
    public double Width
    {
        get { return pri_width; }
        set { pri_width = value < 0 ? -value : value; }
    }

    public double Height
    {
        get { return pri_height; }
        set { pri_height = value < 0 ? -value : value; }
    }

    public string Name { get; set; }

    public void ShowDim()
    {
        Console.WriteLine("Width and height are " + Width + " and " + Height);
    }

    public virtual double Area()
    {
        Console.WriteLine("Area() must be overridden");
        return 0.0;
    }
}

// A derived class of TwoDShape for triangles.
class Triangle : TwoDShape
{
    string Style;

    // A default constructor.
    public Triangle()
    {
        Style = "null";
    }

    // Constructor that takes style, width, and height.
    public Triangle(string s, double w, double h) :  base(w, h, "triangle")
    {
        Style = s;
    }

    // Construct an isosceles triangle.
    public Triangle(double x) : base(x, "triangle")
    {
        Style = "isosceles";
    }

    // Construct a copy of a Triangle object.
    public Triangle(Triangle ob) : base(ob)
    {
        Style = ob.Style;
```

```
  }

  // Override Area() for Triangle.
  public override double Area()
  {
    return Width * Height / 2;
  }

  public void ShowStyle()
  {
    Console.WriteLine("Triangle is " + Style);
  }
}

// A derived class of TwoDShape for rectangles.
class Rectangle : TwoDShape
{
  // Constructor that takes width and height.
  public Rectangle(double w, double h) :  base(w, h, "rectangle")
{
}

  // Construct a square.
  public Rectangle(double x) :  base(x, "rectangle")
{
 }

  // Construct an object from an object.
  public Rectangle(Rectangle ob) : base(ob)
{
 }

  // Return true if rectangle is square.
  public bool IsSquare()
  {
    if (Width == Height) return true;
    return false;
  }

  // Override Area() for Rectangle.
  public override double Area()
  {
    return Width * Height;
  }
}

class DynShapes
{
  static void Main()
  {
    TwoDShape[] shapes = new TwoDShape[5];

    shapes[0] = new Triangle("right", 8.0, 12.0);
    shapes[1] = new Rectangle(10);
    shapes[2] = new Rectangle(10, 4);
    shapes[3] = new Triangle(7.0);
    shapes[4] = new TwoDShape(10, 20, "generic");

    for (int i = 0; i < shapes.Length; i++)
    {
      Console.WriteLine("object is " + shapes[i].Name);
      Console.WriteLine("Area is " + shapes[i].Area());

      Console.WriteLine();
    }
  }
}
```

8.5

8.5.1

αφαιρετική μέθοδος

όταν μία μέθοδος δηλώνεται αλλά *δεν υλοποιείται (δεν περιέχει εντολές)* σε μία κλάση, τότε λέμε ότι η μέθοδος αυτή είναι **αφαιρετική**. Για να δηλώσουμε μία τέτοια μέθοδο χρησιμοποιούμε την εξής μορφή;

abstract τύπος όνομα(παράμετροι);

8.5.2

αφαιρετική κλάση

όταν μία κλάση περιέχει *έστω μία αφηρημένη μέθοδο*, τότε πρέπει και αυτή να δηλωθεί ως abstract. Είναι προφανές ότι ***δεν μπορούμε να δηλώσουμε αντικείμενα*** μίας αφαιρετικής κλάσης.
Όταν μία παραγόμενη κλάση κληρονομεί μία αφαιρετική κλάση, τότε ***πρέπει να υλοποιεί όλες τις αφαιρετικές μεθόδους***. (εάν δεν το κάνει τότε πρέπει να δηλωθεί και αυτή με τη σειρά της ως abstract).

8.6

Άσκηση

```
using System;

abstract class TwoDShape
{
   double pri_width;
   double pri_height;

   // A default constructor.
   public TwoDShape()
   {
      Width = Height = 0.0;
      Name = "null";
   }

   // Parameterized constructor.
   public TwoDShape(double w, double h, string n)
   {
      Width = w;
      Height = h;
      Name = n;
   }

   // Construct object with equal width and height.
   public TwoDShape(double x, string n)
   {
      Width = Height = x;
      Name = n;
   }

   // Construct an object from an object.
   public TwoDShape(TwoDShape ob)
   {
      Width = ob.Width;
      Height = ob.Height;
      Name = ob.Name;
   }

   // Properties for Width and Height.
   public double Width
   {
      get { return pri_width; }
      set { pri_width = value < 0 ? -value : value; }
   }

   public double Height
   {
      get { return pri_height; }
      set { pri_height = value < 0 ? -value : value; }
   }
```

```
    public string Name { get; set; }

    public void ShowDim()
    {
        Console.WriteLine("Width and height are " +  Width + " and " + Height);
    }

    public abstract double Area();
}

// A derived class of TwoDShape for triangles.
class Triangle : TwoDShape
{
    string Style;

    // A default constructor.
    public Triangle()
    {
        Style = "null";
    }

    // Constructor that takes style, width, and height.
    public Triangle(string s, double w, double h) :   base(w, h, "triangle")
    {
        Style = s;
    }

    // Construct an isosceles triangle.
    public Triangle(double x) : base(x, "triangle")
    {
        Style = "isosceles";
    }

    // Construct an object from an object.
    public Triangle(Triangle ob) : base(ob)
    {
        Style = ob.Style;
    }

    // Override Area() for Triangle.
    public override double Area()
    {
        return Width * Height / 2;
    }

    public void ShowStyle()
    {
        Console.WriteLine("Triangle is " + Style);
    }
}

// A derived class of TwoDShape for rectangles.
class Rectangle : TwoDShape
{
    // Constructor that takes width and height.
    public Rectangle(double w, double h) :  base(w, h, "rectangle") { }

    // Construct a square.
    public Rectangle(double x) :  base(x, "rectangle") { }

    // Construct an object from an object.
    public Rectangle(Rectangle ob) : base(ob) { }

    // Return true if rectangle is square.
    public bool IsSquare()
    {
        if (Width == Height) return true;
        return false;
    }

    // Override Area() for Rectangle.
    public override double Area()
    {
        return Width * Height;
```

```
   }
}

class AbsShape
{
   static void Main()
   {
      TwoDShape[] shapes = new TwoDShape[4];

      shapes[0] = new Triangle("right", 8.0, 12.0);
      shapes[1] = new Rectangle(10);
      shapes[2] = new Rectangle(10, 4);
      shapes[3] = new Triangle(7.0);

      for (int i = 0; i < shapes.Length; i++)
      {
         Console.WriteLine("object is " + shapes[i].Name);
         Console.WriteLine("Area is " + shapes[i].Area());

         Console.WriteLine();
      }
   }
}
```

8.6

Ερωτήσεις κατανόησης

1 Δείτε τις παρακάτω κλάσεις. Ποιο είναι το λάθος;

```
using System;


namespace Paradeigma
{
   class A
   {
      int i, j;

      public A()
      {
         i = 0;
         j = 0;
         Console.WriteLine(i + "," + j);
      }

      public A(int i, int j)
      {
         this.i = i;
         this.j = j;
         Console.WriteLine(i + "," + j);
      }

   }

   class B : A
   {
      int w;

      public B()
      {
         w = 0;
         Console.WriteLine(w+","+i+","+j);
      }
   }
}
```

2 Δείτε τον παρακάτω κώδικα. Ποιο είναι το λάθος;

```
using System;
```

```
namespace Paradeigma
{
   class A
   {
      protected int i, j;

      public A()
      {
         i = 0;
         j = 0;
         Console.WriteLine(i + "," + j);
      }

      public A(int i, int j)
      {
         this.i = i;
         this.j = j;
         Console.WriteLine(i + "," + j);
      }

   }

   class B : A
   {
      int w;

      public B()
      {
         w = 0;
         Console.WriteLine(w+","+i+","+j);
      }

      public B(int x) : base(x)
      {

      }
   }
}
```

3 Έχετε τις παρακάτω κλάσεις:

```
using System;

namespace Paradeigma
{
   class AKlasi
   {
      protected int i, j;

      public AKlasi()
      {
         i = 0;
         j = 0;
         Console.WriteLine(i + "," + j);
      }

      public AKlasi(int i, int j)
      {
         this.i = i;
         this.j = j;
         Console.WriteLine(i + "," + j);
      }

   }

   class BKlasi : AKlasi
   {
      int w;

      public BKlasi()
      {
         w = 0;
         Console.WriteLine(w+","+i+","+j);
```

```
        }

        public BKlasi(int x)
            : base(x,0)
        {
            w = 0;
        }
    }
}
```

και σε ένα άλλο αρχείο το ακόλουθο πρόγραμμα:

```
using System;

namespace Paradeigma
{
    class Program
    {
        static void Main()
        {
            BKlasi antik = new BKlasi(1);

            AKlasi antik2 = new BKlasi();

            Console.ReadKey();
        }
    }
}
```

Είναι σωστό; Αν ναι, τι θα εκτυπώσει, αν όχι ποιο είναι το λάθος;

4 Έχετε τις ακόλουθες κλάσεις:

```
using System;

namespace Paradeigma
{
    abstract class AKlasi
    {
        protected int i, j;

        public AKlasi()
        {
            i = 0;
            j = 0;
            Console.WriteLine(i + "," + j);
        }

        public AKlasi(int i, int j)
        {
            this.i = i;
            this.j = j;
            Console.WriteLine(i + "," + j);
        }

        public abstract void Ektypwsh();

    }

     abstract class BKlasi : AKlasi
    {
        protected int w;

        public BKlasi()
        {
            w = 0;
            Console.WriteLine(w+","+i+","+j);
        }

        public BKlasi(int x)
            : base(x,0)
        {
            w = 0;
```

```
      }
   }

   class CKlasi : BKlasi
   {
      public override void Ektypwsh()
      {
         Console.WriteLine("i={0},j={1},w={2}",i,j,w);
      }
   }
}
```

και το παρακάτω πρόγραμμα:

```
using System;

namespace Paradeigma
{
   class Program
   {
      static void Main()
      {

         AKlasi antik = new CKlasi();

         antik.Ektypwsh();

         Console.ReadKey();
      }
   }
}
```

Τι θα εκτυπωθεί;

9

Ασκήσεις

Άσκηση 1

Παράδειγμα με abstract

```
using System;

abstract class publication
{
   protected string titlos;
   protected float timi;

   public publication(string s, float t)
   {
      titlos = s;
      timi = t;
   }
   abstract public void prnData();
}

class book : publication
{
   private int selides;

   public book(string s, float t, int sel)
      : base(s, t)
   {
      selides = sel;
   }
   public override void prnData()
   {
      Console.WriteLine(titlos + " " + timi + " " + selides);
   }
}

class cd : publication
{
   private float diarkeia;

   public cd(string s, float t, float dia)
      : base(s, t)
   {
      diarkeia = dia;
   }
   public override void prnData()
   {
      Console.WriteLine(titlos + " " + timi + " " + diarkeia);
   }
}

class efarmogi
{
   static void Main(string[] args)
   {

      publication[] publ = new publication[2];

      publ[0] = new book("The excaliber", (float)12.5, 323);
      publ[1] = new cd("Axion Esti", (float)18.25, (float)58.42);

      publ[0].prnData();
```

```
        publ[1].prnData();

        Console.ReadKey();
    }
}
```

Άσκηση 2

Παράδειγμα με κληρονομικότητα

```
using System;
class publication
{
    protected string titlos;
    protected float timi;

    public void InData(string s, float t)
    {
        titlos = s;
        timi = t;
    }
    public string getTitlos()
    {
        return titlos;
    }
    public float getTimi()
    {
        return timi;
    }
}

class book : publication
{
    private int selides;

    public void InData(string s, float t, int sel)
    {
        base.InData(s, t);
        selides = sel;
    }
    public void prnData()
    {
        Console.WriteLine(base.getTitlos() + " " + base.getTimi() + " " + selides);
    }
}

class cd : publication
{
    private float diarkeia;

    public void InData(string s, float t, float dia)
    {
        InData(s, t);
        diarkeia = dia;
    }
    public void prnData()
    {
        Console.WriteLine(getTitlos() + " " + getTimi() + " " + diarkeia);
    }
}

class efarmogi
{
    static void Main(string[] args)
    {
```

```
        book bi = new book();
        cd ka = new cd();

        bi.InData("The excaliber", (float)12.5, 323);
        ka.InData("Axion Esti", (float)18.25, (float)58.42);
        bi.prnData();
        ka.prnData();

        Console.ReadKey();
    }
}
```

Άσκηση 3

Παράδειγμα με πολυμορφισμό

```
using System;
class publication
{
    protected string titlos;
    protected float timi;

    public publication(string s, float t)
    {
        titlos = s;
        timi = t;
    }
    public void prnData()
    {
        Console.WriteLine(titlos + " " + timi);
    }
}

class book : publication
{
    private int selides;

    public book(string s, float t, int sel)
        : base(s, t)
    {
        selides = sel;
    }
    public void prnData()
    {
        Console.WriteLine(titlos + " " + timi + " " + selides);
    }
}

class cd : publication
{
    private float diarkeia;

    public cd(string s, float t, float dia)
        : base(s, t)
    {
        diarkeia = dia;
    }
    public void prnData()
    {
        Console.WriteLine(titlos + " " + timi + " " + diarkeia);
    }
}

class efarmogi
{
    static void Main(string[] args)
    {
        int i;
        publication[] publ = new publication[3];
```

```
        publ[0] = new publication("No title", 0F);
        publ[1] = new book("The excaliber", (float)12.5, 323);
        publ[2] = new cd("Axion Esti", (float)18.25, (float)58.42);

        for (i = 0; i < 3; i++) publ[i].prnData();
    }
}
```

Άσκηση Εργαστηρίου

Κλάση Sxhma:

```
using System;
using System.IO;

namespace Lab9Askisi
{
    class Sxhma
    {
        //metavlhtes
        protected string onomaSxhmatos;
        protected double emvadon;

        //kataskeuastes
        public Sxhma() { }
        public Sxhma(string onomaSxhmatos, double emvadon)
        {
            this.onomaSxhmatos = onomaSxhmatos;
            this.emvadon = emvadon;
        }

        //methodoi
        public virtual void YpologismosEmvadou()
        {
            Console.WriteLine("Το εμβαδόν του σχήματος {0} είναι: {1:0.00}", onomaSxhmatos, emvadon);
            string path=@"c:\Εμβαδόν\Εμβαδόν.txt";
            StreamWriter arxeio;
            if(!File.Exists(path))
                arxeio = File.CreateText(path);
            else
                arxeio = File.AppendText(path);
            arxeio.WriteLine("Το εμβαδόν του σχήματος {0} είναι: {1:0.00}", onomaSxhmatos, emvadon);
            arxeio.Close();
        }

        //idiothtes
        public string OnomaSxhmatos
        {
            set { onomaSxhmatos = value; }
            get { return onomaSxhmatos; }
        }
        public double Emvadon
        {
            set
            {
                if (value >= 0.0)
                    emvadon = value;
            }
            get { return emvadon; }
        }
    }
}
```

Κλάση Kiklos:

```
using System;

namespace Lab9Askisi
{
   class Kiklos:Sxhma
   {
      //metavlhtes
      protected double aktina;

      //kataskeuastes
      public Kiklos() { }
      public Kiklos(double aktina, string onomaSxhmatos, double emvadon)
         : base(onomaSxhmatos, emvadon)
      {
         this.aktina = aktina;
      }

      //methodoi
      public override void YpologismosEmvadou()
      {
         emvadon = Math.PI * Math.Pow(aktina, 2.0);
         base.YpologismosEmvadou();
      }

      //idiothtes
      public double Aktina
      {
         set
         {
            if (value >= 0.0)
               aktina = value;
         }
         get { return aktina; }
      }
   }
}
```

Κλάση Tetragwno:

```
using System;

namespace Lab9Askisi
{
   class Tetragwno:Sxhma
   {
      //metavlhtes
      protected double pleura;

      //kataskeuastes
      public Tetragwno() { }
      public Tetragwno(double pleura, string onomaSxhmatos, double emvadon)
         : base(onomaSxhmatos, emvadon)
      {
         this.pleura = pleura;
      }

      //methodoi
      public override void YpologismosEmvadou()
      {
         emvadon = Math.Pow(pleura, 2.0);
         base.YpologismosEmvadou();
      }

      //idiothtes
      public double Pleura
      {
         set
         {
            if (value >= 0.0)
               pleura = value;
         }
```

```
        get { return pleura; }
      }
   }
}
```

Κλάση Trigwno:

```
using System;

namespace Lab9Askisi
{
   class Trigwno:Sxhma
   {
      //metavlhtes
      protected double vash, ypsos;

      //kataskeustes
      public Trigwno() { }
      public Trigwno(double vash, double ypsos, string onomaSxhmatos, double emvadon)
         : base(onomaSxhmatos, emvadon)
      {
         this.vash = vash;
         this.ypsos = ypsos;
      }

      //methodoi
      public override void YpologismosEmvadou()
      {
         emvadon = vash * ypsos * 0.5;
         base.YpologismosEmvadou();
      }

      //idiothtes
      public double Vash
      {
         set
         {
            if (value >= 0.0)
               vash = value;
         }
         get { return vash; }
      }
      public double Ypsos
      {
         set
         {
            if (value >= 0.0)
               ypsos = value;
         }
         get { return ypsos; }
      }
   }
}
```

Πρόγραμμα (Main()):

```
using System;

namespace Lab9Askisi
{
   class Program
   {
      static void Main()
      {
         string epilogh;


            Console.WriteLine("Επιλογή σχήματος:");
            epilogh = Console.ReadLine();

            switch (epilogh)
            {
```

```
            case "τρίγωνο":
                Trigwno tr = new Trigwno();
                Console.WriteLine("Βάση του τριγώνου:");
                tr.Vash = Double.Parse(Console.ReadLine());
                Console.WriteLine("Ύψος του τριγώνου:");
                tr.Ypsos = Double.Parse(Console.ReadLine());
                tr.OnomaSxhmatos="Τρίγωνο";
                tr.YpologismosEmvadou();
                break;
            case "τετράγωνο":
                Tetragwno te = new Tetragwno();
                Console.WriteLine("Πλευρά τετραγώνου:");
                te.Pleura = Double.Parse(Console.ReadLine());
                te.OnomaSxhmatos = "Τετράγωνο";
                te.YpologismosEmvadou();
                break;
            case "κύκλος":
                Kiklos ki = new Kiklos();
                Console.WriteLine("Ακτίνα κύκλου:");
                ki.Aktina = Double.Parse(Console.ReadLine());
                ki.OnomaSxhmatos = "Κύκλος";
                ki.YpologismosEmvadou();
                break;
            default:
                Sxhma sx = new Sxhma();

                sx.OnomaSxhmatos = epilogh;
                Console.WriteLine("Εμβαδόν σχήματος:");
                sx.Emvadon = Double.Parse(Console.ReadLine());
                sx.YpologismosEmvadou();
                break;
        }

        Console.ReadKey(true);
    }
  }
}
```

Παράρτημα Α'

απαντήσεις ερωτήσεων κατανόησης

Lab1

1 ναι. προσοχή ειδικά στα πρώτα γραμματα κάθε λέξης σε ονόματα χώρου ονομάτων, κλάσεων και μεθόδων

2 προσπαθούμε να χρησιμοποιήσουμε ορίσματα που δεν έχουνε δωθεί αφού ο πίνακας args είναι άδειος. Αυτό είναι το αποτέλεσμα (exception):

Στην ουσία, το λαθος που μας εμφανίζει είναι ότι ο δείκτης του πίνακα args που χρησιμοποιούμε είναι μεγαλύτερος από τα πραγματικά στοιχεία του πίνακα. Πατάμε Shift+F5 για να σταματήσουμε την εκτέλεση του προγραμματός μας.

3 προφανώς οι αγκύλες στον ορισμό του πίνακα args πρέπει να είναι πριν το όνομα του πίνακα (σε αντίθεση με τη C). Να το λάθος:

Error List

1 Error | 0 Warnings | 0 Messages

	Description
1	Array type specifier, [], must appear before parameter name

4 δεν έχουμε ορίσει τη θεμελιώδη μονάδα ενθυλάκωσης: την κλάση. Από εκεί και πέρα εμφανίζονται πολλά λάθη που προέρχονται από τη βασική μας παράλειψη.

Error List

7 Errors | 0 Warnings | 0 Messages

	Description
1	Expected class, delegate, enum, interface, or struct
2	Identifier expected
3	Expected class, delegate, enum, interface, or struct
4	Identifier expected
5	Identifier expected
6	Identifier expected
7	Type or namespace definition, or end-of-file expected

Παρατηρήστε το πρώτο λάθος του μεταγλωττιστή:

5 η μέθοδος Main() δεν είναι static. Αυτό σημαίνει ότι θα έπρεπε να δημιουργήσουμε πρώτα ένα αντικείμενο της κλάσης Welcome και να την καλέσουμε με χρήση αυτού του αντικειμένου, κάτι που φυσικά είναι λάθος:

Lab2

1 αναπαριστά έναν χαρακτήρα μέσα σε μονά εισαγωγικά. Οι χαρακτήρες αυτοί είναι Unicode και όχι ASCII έτσι ώστε να σε πολλές γλώσσες.

2 λάθος. μία τιμή bool είναι true ή false.

3 υπάρχουν 2 λάθη:

- η μεταβλητή sum έπρεπε να δηλωθεί πριν από τη for
- στην ενοτλή εκτύπωσης το , έπρεπε να είναι +

4
36
35
35

5

```
To
onoma sou einai
Germanos
```

το πρόγραμμα είναι σωστό και αν δώσουμε ως όνομα το Germanos τότε θα εκτυπώσει:

6 υπάρχουν 2 λάθη:

- η σταθερά πρέπει να δηλωθεί εντός της κλάσης
- το όνομα της βασικής συνάρτησης πρέπει να είναι Main() και όχι main()

7 υπάρχει λάθος στον κώδικα:

η γραμμή x=a+b; έπρεπε να ήταν x=(byte)(a+b);

8

```
300
400
x=188
```

Η εξήγηση είναι η εξής:
300 και 400 μας κάνουν 700.Όμως, η μεταβλητή x είναι τύπου byte οπότε μπορεί να κρατήσει τιμές μέχρι το 255, η πρώτη τιμή που δεν μπορεί να κρατήσει είναι η 256.. Άρα, 700-256=444 αλλά και 444-256-188. Άρα το **188** είναι το πρώτο αποδεκτό νούμερο που μπορεί να κρατήσει η μεταβλητή x.

Lab3

1 ένας σωστός συνδυασμός τιμών θα ήταν :
X=0 και Y=0

2 μετά την εκτέλεση της break εμφανίζεται "after while"

3

0 1
2 3
4 5
6 7
8 9
10

4

0 1 | (ο τελευταίος χαρακτήρας είναι ο κέρσορας)

5

i=1 και i+=i

6

το πρόγραμμα θα εκτυπώσει την τιμή 2999 και στη συνέχεια θα κολλήσει.

Lab4

1

κλάση είναι ένα πρότυπο που ορίζει την μορφή ενός αντικειμένου. Καθορίζει τα δεδομένα και τον κώδικα που θα δρα πάνω σε αυτά τα δεδομένα. Τα αντικείμενα είναι στιγμιότυπα της κλάσης στην οποία ανήκουν.

2

- τα 2 αντικείμενα πρέπει να δηλωθούν ξεχωριστά.
- ο καθοριστής προσπέλασης της μεθόδου YpologismosMO πρέπει να τεθεί public.

3

```
public Foititis(string a, string b, float p1, float p2, float p3)
    {

        onomatepwnymo = a;

        AM = b;

        programmatismos1 = p1;

        programmatismos2 = p2;

        programmatismos3 = p3;

    }
```

Στη συνέχεια θα έπρεπε να δώσουμε ορίσματα στους κατασκευαστές κατά τη δήλωση των αντικειμένων.

4

```
Times tou antikeimenou x1:0,0
Times tou antikeimenou x2:0,0
Times tou antikeimenou x1:90,25
Times tou antikeimenou x2:90,25
Times tou antikeimenou x1:90,25
Times tou antikeimenou x2:0,0
Times tou antikeimenou x3:0,0
```

5

```
2 is factor
Press any key to continue . . .
```

Lab5

1 String[,] x=new string[10,9];

2 Υπάρχει φυσικά λάθος: ενώ πάμε να δηλώσουμε δισδιάστατο πίνακα σταθερής διάστασης, στην αρχικοποίηση δεν έχουμε τον ίδιο αριθμό στοιχείων σε κάθε γραμμή.

3 Όχι. Η μεταβλητή που χρησιμοποιούμε στη foreach θα έπρεπε να ήταν τύπου char, δηλαδή char i;

4 Θα εκτυπωθεί:
tei
larissas
tei

Lab6

1 όχι. οι μέθοδοι Meth() έχουν ίδιο πλήθος και είδος ορισμάτων, οπότε ο μεταγλωττιστής δεν μπορεί να διαχωρίσει ποια θα καλεί κάθε φορά.

2 παράγραφος 6.1.2.

3 όχι. τα λάθη είναι 2: δεν μπορούμε να έχουμε πρόσβαση στη metavliti αφού είναι private και δεν έχουμε αποδόσει τιμή στη μεταβλητή temp, την οποία την μεταβιβάζουμε στη synarthsh() με αναφορά.

4 όχι. η μεταβλητή i πρέπει να πάρει τιμή η οποία είναι ανεξάρτητη από τον εαυτό της.

5
```
public Dokimes():("")
  {

  }
```

Lab7

1
```
21
21
Press any key to continue . . .
```

2
```
36
36
Press any key to continue . . .
```

3 Ένας κατασκευαστής static δεν μπορεί να έχει ορίσματα.

4

```
21
21
Press any key to continue . . .
```

5 Ναι, υπάρχει λάθος: η γραμμή x=9; στον κατασκευαστή.

Lab8

---

1 Τα i,j δεν κληρονομούνται στη κλάση B.

2 Το λάθος βρίσκεται στον κατασκευαστή

```
public B(int x)
      : base(x)
```

ο οποίος καλεί τον κατασκευαστή της κλάσης *A* με ένα όρισμα , ο οποίος δεν υπάρχει!

3

```
1,0
0,0
0,0,0
```

4

```
0,0
0,0,0
i=0,j=0,w=0
```