Ευρετήριο εργαστηρίων

lab

1

εισαγωγή στη C#

Τι θα δούμε σε αυτό το εργαστήριο

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 :

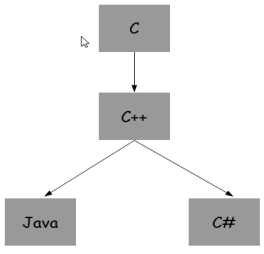
- **μεταφερσιμότητα:** μετάφραση του πηγαίου κώδικα στην ενδιάμεση γλώσσα bytecode και στη συνέχεια εκτέλεση αυτού του κώδικα στην εικονική μηχανή της Java (Java Virtual Machine-JVM).
- ο κώδικας μπορούσε να εκτελεστεί σε οποιαδήποτε περιβάλλον διέθετε εικονική μηχανή Java.
- **ασφάλεια:** επειδή ο JVM 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

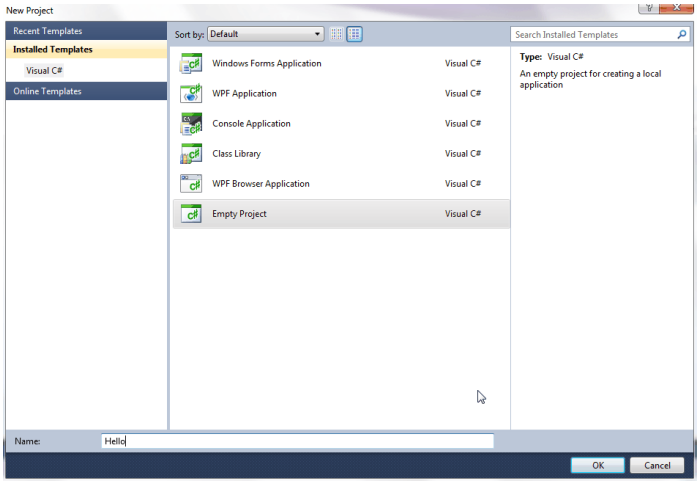
Δημιουργία του πρώτου προγράμματος

Μεταγλωττιστής C#

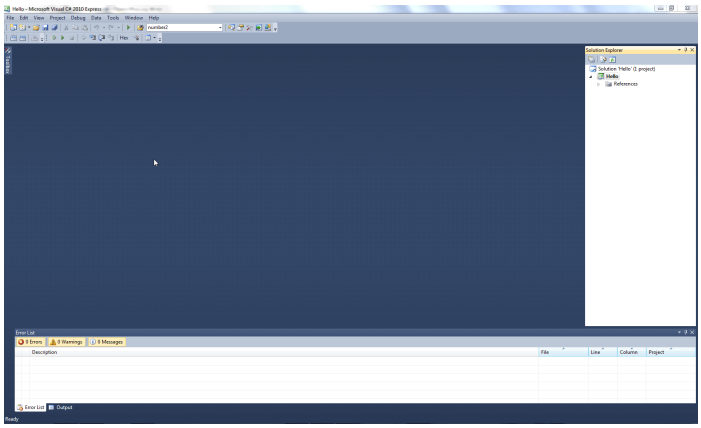
για να δημιουργήσετε , να μεταγλωττίσετε και να εκτελέσετε οποιοδήποτε πρόγραμμα C# θα χρειαστείτε τη ***Microsoft Visual C# 2010 Express Edition*** (δείτε και **Εισαγωγή.2**) αν έχετε λειτουργικό Microsoft Windows (Xp,Vista, 7.0) ή ***MonoDevelop*** αν έχετε λειτουργικό MacOS ή Linux.

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

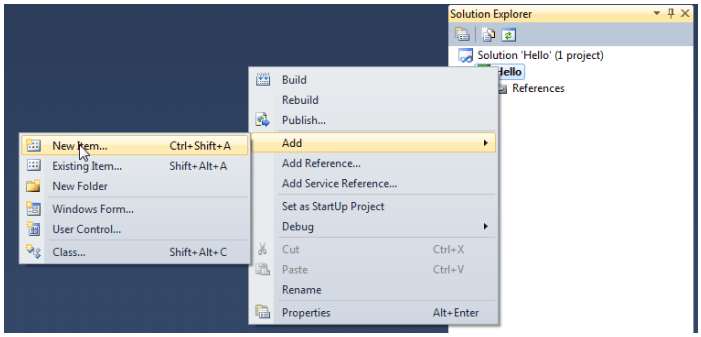
1. αρχικά πρέπει να δημιουργήσουμε ένα νέο, κενό έργο επιλέγοντας File | New Project:



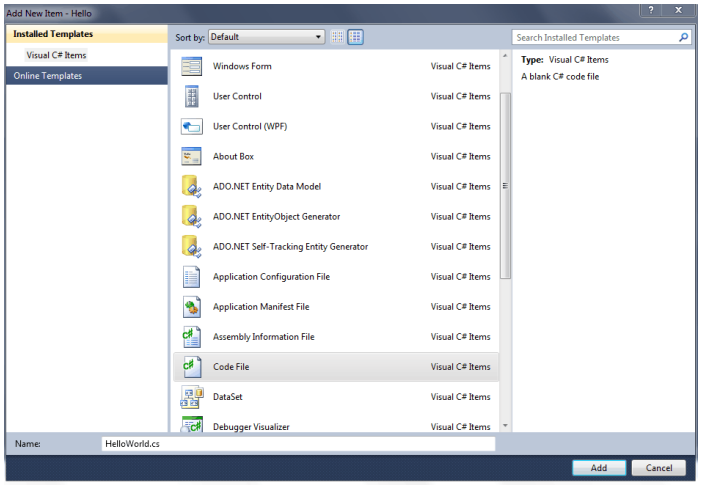
με το όνομα Hello. Πατάμε OK και βρισκόμαστε στο εξής σημείο:



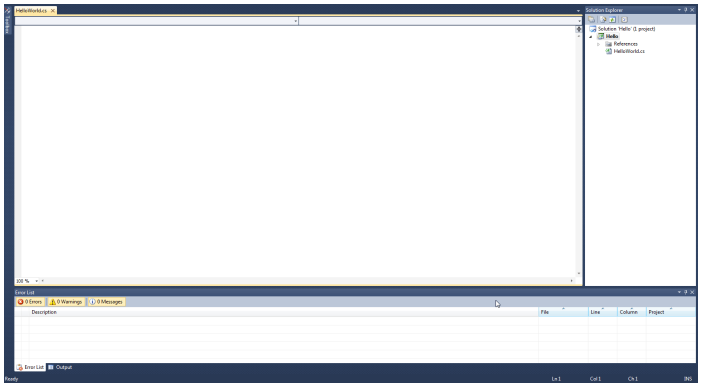
Στο δεξί μέρος βλέπουμε το παράθυρο του Solution Explorer που μας δείχνει το project που δημιουργήσαμε και τι υπάρχει μέσα σε αυτό. Στη συνέχεια, για να προσθέσουμε ένα αρχείο πηγαίου κώδικα σε αυτό κάνουμε δεξί κλικ πάνω στο Hello μέσα στον Solution Explorer και επιλέγουμε Add | New Item... :



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.

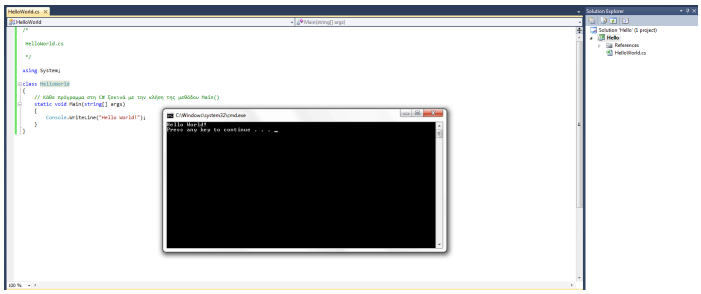
Το πρώτο μας πρόγραμμα

```
/*

 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!");

    }

}
```

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.

1.9

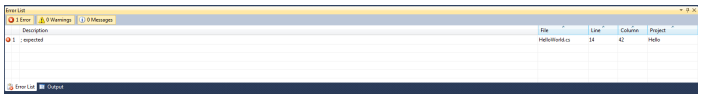
Έλεγχος συντακτικών σφαλμάτων

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

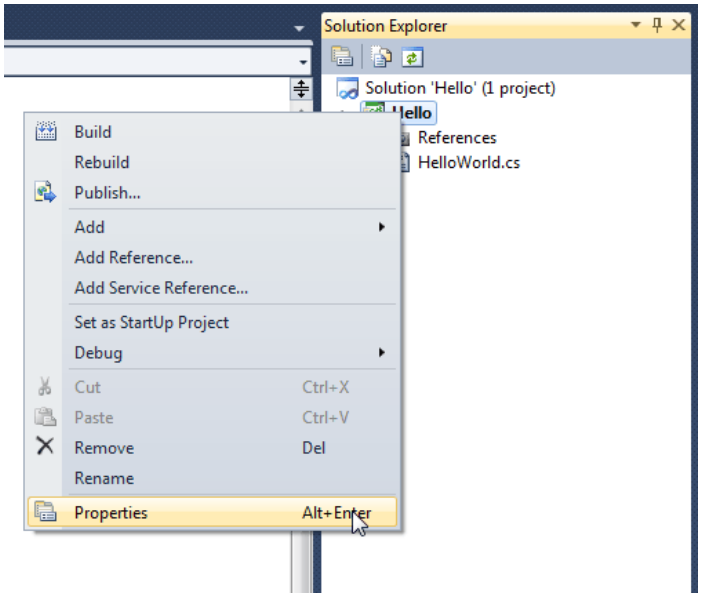
είδαμε στο προηγούμενο παράδειγμα του HelloWorld.cs ότι μπορούμε να περάσουμε ως όρισμα στη βασική συνάρτηση έναν *πίνακα αλφαριθμητικών* με το όνομα **args**. Η δήλωση του πίνακα είναι η εξής:

string [] args

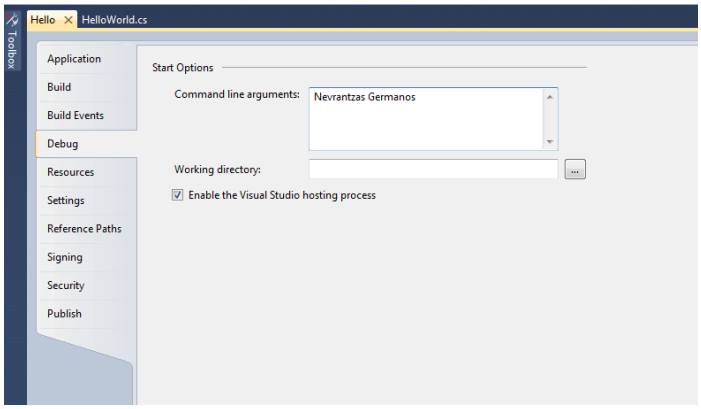
Ο πίνακας αυτός διοχετεύει στο πρόγραμμα (άρα αυτό μπορεί να τα χρησιμοποιήσει στη συνέχεια) ότι πληροφορία του δώσει ο χρήστης με *εξωτερικό τρόπο*.

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

κώδικας

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

```
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;

lab

2

Δεδομένα, τελεστές, είσοδος/έξοδος

Τι θα δούμε σε αυτό το εργαστήριο

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	[-2147483648,2147483647]
uint	32	[0,4294967295]
long	64	[-9223372036854775808,9223372036854775807]
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 τροποποίηση 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 οπωσδήποτε να την *αρχικοποιήσουμε*.

παραδειγμα

στη δήλωση

var pi=3.1419;

var radius=10;

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

περιορισμός

μπορούμε να δηλώσουμε *μία και μόνο μία* 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 κ 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 Μία τιμή bool μπορεί να έχει όποια τιμή θέλετε, επειδή 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()

    {

        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;

        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 χ 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

1. ο χρήστης να δίνει 2 ακεραίους

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

```
using System;

class Lab1Ex2

{

    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 = 0.82 * δολλάρια

```
using System;
class Lab1Ex3
{
    public static void Main()
    {
        const double ISOTIMIA = 0.82;

        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();
```

```
    }
}
```

lab

3

Δομές, ελέγχου & επανάληψης

Τι θα δούμε σε αυτό το εργαστήριο

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 *σύνταξη* της δομής είναι η εξής:

η δομή if

```
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 *σύνταξη* της δομής είναι η εξής:

η δομή switch

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

```
using System;
```

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

```
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 foiththh:", 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 δομή do...while

Η *σύνταξη* της δομής είναι η εξής:

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.

<u>Σημαντικές επισημάνσεις</u>:

- 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 δομή for

Η *σύνταξη* της δομής είναι η εξής:

```
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);
    }
 }
```

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 i;

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

      for (i = 0; (char)Console.Read() != '\n'; i++)
        ;
        Console.WriteLine("Edwses " + (i-1) +" 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()
   {

      int i;

      for (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()
   {

      int i;

      for (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, y;

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

άσκηση εργαστηρίου

Να γραφεί πρόγραμμα το οποίο να υπολογίζει το

N παραγοντικό

με βάση τον τύπο

N!=1*2*…*(N-1)*N

περιορισμοί για το N: θετικός

άσκηση εργαστηρίου

Να γίνει πρόγραμμα το οποίο να εξετάζει

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

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

άσκηση εργαστηρίου

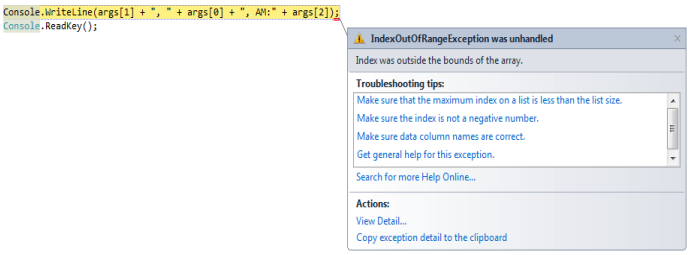
Παράρτημα Α'

απαντήσεις ερωτήσεων κατανόησης

Lab1

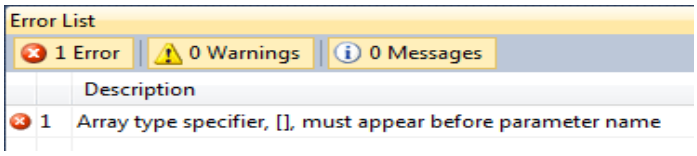
1 ναι. προσοχή ειδικά στα πρώτα γραμματα κάθε λέξης σε ονόματα χώρου ονομάτων, κλάσεων και μεθόδων

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

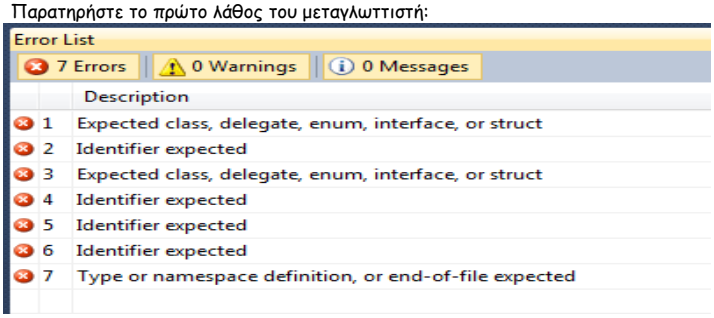


Στην ουσία, το λαθος που μας εμφανίζει είναι ότι ο δείκτης του πίνακα args που χρησιμοποιούμε είναι μεγαλύτερος από τα πραγματικά στοιχεία του πίνακα. Πατάμε Shift+F5 για να σταματήσουμε την εκτέλεση του προγραμματός μας.

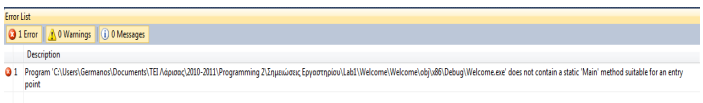
3 προφανώς οι αγκύλες στον ορισμό του πίνακα args πρέπει να είναι πριν το όνομα του πίνακα (σε αντίθεση με τη C). Να το λάθος:



4 δεν έχουμε ορίσει τη θεμελιώδη μονάδα ενθυλάκωσης: την κλάση. Από εκεί και πέρα εμφανίζονται πολλά λάθη που προέρχονται από τη βασική μας παράλειψη. 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 το πρόγραμμα είναι σωστό και αν δώσουμε ως όνομα το Germanos τότε θα εκτυπώσει:

```
To
onoma sou einai
Germanos
```

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.