https://rextester.com/

https://dotnetfiddle.net/

C#

Η C# είναι μια σχετικά καινούργια γλώσσα, ενώ έχει γίνει μια από τις πιο διαδεδομένες αντικειμενοστρεφείς γλώσσες προγραμματισμού.

Ο αντικειμενοστρεφής προγραμματισμός

- Ο αντικειμενοστρεφής προγραμματισμός καθιερώθηκε τη δεκαετία του 1990 (η γλώσσα που συνεισέφερε αρκετά σε αυτό ήταν η Java) αντικαθιστώντας σε μεγάλο βαθμό το παραδοσιακό πρότυπο του προστακτικού προγραμματισμού.
- Εδώ το βασικό στοιχείο του προγράμματος είναι το *αντικείμενο* (object) που συνδυάζει ιδιότητες και διαδικασίες που επενεργούν σε αυτές αλλάζοντας τα δεδομένα τους.
- Κάθε αντικείμενο αποθηκεύει δεδομένα σε μεταβλητές και απαντά σε μηνύματα εκτελώντας τις διαδικασίες που καλούνται και *μέθοδοι* (methods).
- Το αντικείμενο αποτελεί δεσμευμένο κομμάτι της μνήμης (και συγκεκριμένα του σωρού) και καλείται και *στιγμιότυπο* (instance), ενός σύνθετου *τύπου δεδομένων* (data type) που καλείται *κλάση* (class).
- Η κλάση είναι μία αυτοτελής και αφαιρετική αναπαράσταση κάποιας κατηγορίας αντικειμένων, είτε φυσικών αντικειμένων του πραγματικού κόσμου είτε νοητών, εννοιολογικών αντικειμένων και στην ουσία αποτελεί την προδιαγραφή των δεδομένων και των διαδικασιών που επιδρούν πάνω σε αυτά.

- Αν κανείς παρατηρήσει πως υλοποιούνται οι διάφορες εργασίες στον πραγματικό κόσμο, θα παρατηρήσει ότι αλληλεπιδρά σε έναν κόσμο που στρέφεται γύρω από αντικείμενα. Αν θέλει κάποιος να πάει σε ένα μέρος για παράδειγμα μπορεί να χρησιμοποιήσει το αυτοκίνητό του.
- Το αντικείμενο αυτοκίνητο αποτελείται από άλλα αντικείμενα, όπως η μηχανή που με τη σειρά της αποτελείται από άλλα αντικείμενα κ.ο.κ. καθένα από τα οποία αλληλεπιδρά με το άλλο για να πραγματοποιήσουν μια συγκεκριμένη εργασία, το σύνολο το οποίων όμως έχει σαν απώτερο στόχο να μετακινήσει το αυτοκίνητο από ένα σημείο σε ένα άλλο.

- στο παράδειγμα με το αυτοκίνητο, μπορούμε να ορίσουμε μία προδιαγραφή της μορφής και της συμπεριφοράς μιας κλάσης αντικειμένων (π.χ. μία κλάση "αυτοκίνητο"), ορίζοντας τόσο τις ιδιότητες της (π.χ. μία μεταβλητή "τρέχουσα ταχύτητα") όσο και τις πράξεις ή χειρισμούς επί αυτών των ιδιοτήτων (π.χ. μία διαδικασία "πάτημα του γκαζιού").
- Στο παράδειγμα αυτό κάθε πραγματικό αυτοκίνητο αναπαρίσταται ως ξεχωριστό, "φυσικό“ στιγμιότυπο αυτής της πρότυπης, ιδεατής κλάσης. Συνεπώς, αν υλοποιούσαμε αυτήν την προδιαγραφή στον υπολογιστή, τα αντικείμενα θα είναι αυτά που καταλαμβάνουν χώρο στη μνήμη του υπολογιστή ενώ οι κλάσεις αποτελούν απλώς τα "καλούπια" για τη δημιουργία τους.

- Οι αιτίες που ώθησαν στην ανάπτυξη του αντικειμενοστρεφούς προγραμματισμού είναι η ευκολία συντήρησης, οργάνωσης, χειρισμού και επαναχρησιμοποίησης κώδικα μεγάλων και πολύπλοκων εφαρμογών και επικράτησε επειδή μπορούσε να αντεπεξέλθει σε προγράμματα πολύ μεγαλύτερου όγκου και πολυπλοκότητας.

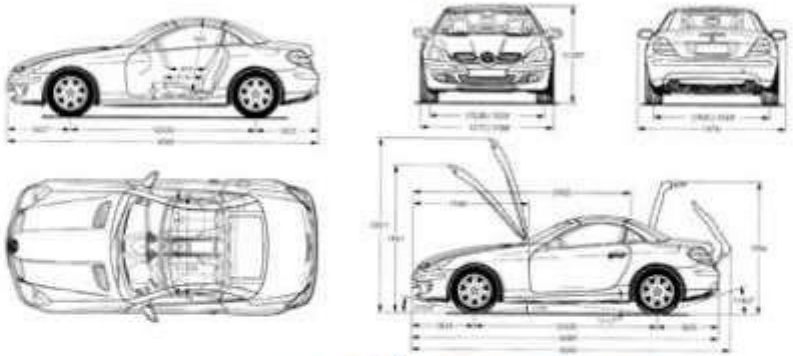
Ενθυλάκωση (encapsulation)

- Η ιδιότητα της ενθυλάκωσης αναφέρεται στην απόκρυψη των δεδομένων και της υλοποίησης ενός αντικειμένου και είναι γνωστή και ως *απόκρυψη δεδομένων* (data hiding).

- Τα δεδομένα και ο κώδικας όταν ενθυλακώνονται, αποκρύπτονται από την εξωτερική πρόσβαση. Όταν ένας εξωτερικός παρατηρητής βλέπει ένα αντικείμενο, μπορεί να το χρησιμοποιήσει μόνο μέσω της εξωτερικής *διασύνδεσής* του (interface), δηλαδή ό, τι είναι διαθέσιμο προς χρήση, ενώ οι εσωτερικές λεπτομέρειες είναι κρυμμένες και δεν μπορούν να χρησιμοποιηθούν.

- Έτσι τα ενθυλακωμένα δεδομένα δεν μπορούν να αλλάξουν και πιο συγκεκριμένα να ενημερωθούν άμεσα από εξωτερικά αντικείμενα. Για παράδειγμα η επιτάχυνση ενός αυτοκινήτου είναι διαθέσιμη για αλλαγή από το γκάζι (διασύνδεση) αλλά το πώς δουλεύει εσωτερικά η μηχανή για την πετύχει, είναι κρυφό από τον οδηγό.

Κλάση: **Αυτοκίνητο**
Ρόδες: 4
Πόρτες: 3 ή 5



Κλάση:
Mercedes SLK 2004
Ρόδες: 4
Πόρτες: 3
Μηχανή: 3.2L V6
Χρώμα:...
Καθίσματα:...

Αντικείμενο:
SLK-s
Ρόδες: 4
Πόρτες: 3
Μηχανή: 3.2L V6
Χρώμα: Ασημί
Καθίσματα: Δερμάτινα

Αντικείμενο:
SLK-b
Ρόδες: 4
Πόρτες: 3
Μηχανή: 3.2L V6
Χρώμα: Μπλε
Καθίσματα: Δερμάτινα

Αντικείμενο:
SLK-r
Ρόδες: 4
Πόρτες: 3
Μηχανή: 3.2L V6
Χρώμα: Κόκκινο
Καθίσματα: Δερματίνη

Εικόνα 1

Αφαίρεση δεδομένων (Abstraction)

- Η αφαίρεση δεδομένων είναι η αρχή που απαιτεί τα προγράμματα να μην λαμβάνουν υπόψη συγκεκριμένες υλοποιήσεις ή εσωτερικές αναπαραστάσεις αλλά να έχουν μια γενική αφηρημένη ιδέα των αντικειμένων που χρησιμοποιούν.
- Στον αντικειμενοστρεφή προγραμματισμό κάθε αντικείμενο μπορεί να ζητηθεί να παρέχει δεδομένα ή υπηρεσίες λαμβάνοντας μηνύματα/αιτήσεις.
- Τον αιτούντα δεν τον απασχολεί πως το αντικείμενο θα παράξει την δεδομένα αλλά τα δεδομένα καθαυτά. Το *πώς* - δηλαδή η υλοποίηση ή αλλιώς η δημιουργία της πληροφορίας - είναι μια εσωτερική για το αντικείμενο υπόθεση.
- Επομένως δίνεται έμφαση στο *τι* αντί για το *πώς* κατά την αλληλεπίδραση μεταξύ δυο αντικειμένων.

Κληρονομικότητα (inheritance)

- Τα αντικείμενα τείνουν να δομούνται βασισμένα σε άλλα αντικείμενα. Όταν δημιουργείται ένα νέο αντικείμενο (μέσα από μια κλάση), ουσιαστικά ορίζονται οι ιδιότητες που το κάνουν ξεχωριστό από τα υπόλοιπα αντικείμενα. Εξαιτίας της ομοιότητας πολλών αντικειμένων υπάρχει ένας μηχανισμός για τη μεταφορά των ιδιοτήτων κάποιων αντικειμένων σε άλλα αντικείμενα έτσι ώστε να αποφεύγεται ο άσκοπος ορισμός των χαρακτηριστικών σε κάθε αντικείμενο ξεχωριστά, αλλά και για να υπάρχει μια νοητή συγγένεια μεταξύ των αντικειμένων. Αυτός ο μηχανισμός καλείται *κληρονομικότητα*. Η κληρονομικότητα επιτρέπει στους προγραμματιστές την επαναχρησιμοποίηση των ορισμών δομών που έχουν οριστεί στο παρελθόν, μειώνοντας έτσι την εργασία κατά την ανάπτυξη προγραμμάτων.
- Πίσω στην Εικόνα 1, η κληρονομικότητα είναι αυτή που μεταφέρει τις ιδιότητες της γενικής κλάσης «Αυτοκίνητο» στην πιο ειδική κλάση Mercedes SLK 2004.

Πολυμορφισμός (polymorphism)

- Ο όρος πολυμορφισμός στην κυριολεξία αναφέρεται στην ιδιότητα του να έχει κάτι πολλαπλές μορφές. Στις γλώσσες προγραμματισμού τις περισσότερες φορές αναφέρεται στην ιδιότητα των μεθόδων να μπορούν να δέχονται και/ή να επιστρέφουν τιμές διαφορετικού τύπου.
- Για παράδειγμα μια μέθοδος που δέχεται ένα μόνο όρισμα είναι πολυμορφική αν μπορεί να δεχτεί παραμέτρους διαφορετικού τύπου. Ο πολυμορφισμός είναι διάχυτος στις αντικειμενοστρεφείς γλώσσες προγραμματισμού και συνδέεται αρκετά με την κληρονομικότητα καθώς ιδιότητες που κληρονομούνται μπορούν να έχουν διαφορετικές μορφές από το αντικείμενο πρόγονο στο αντικείμενο απόγονο.

Στον Πίνακα 1 συνοψίζονται τα κυριότερα χαρακτηριστικά των προαναφερθέντων προγραμματιστικών προτύπων και στον Πίνακα 2 δίνεται το κλασικό πρόγραμμα "hello world" σε χαρακτηριστικές γλώσσες των αντίστοιχων προγραμματιστικών προτύπων.

Πρότυπο	Βασικό στοιχείο	Πρόγραμμα	Εκτέλεση	Αποτέλεσμα
Προστακτικό	Εντολή	Ακολουθία Εντολών	Εκτέλεση εντολών	Τελική κατάσταση μνήμης
Συναρτησιακό	Συνάρτηση	Αποτίμηση συναρτήσεων	Τιμή της κύριας συνάρτησης	
Λογικό	Πρόταση	Αξιώματα και θεώρημα	Απόδειξη θεωρήματος	Επιτυχία ή αποτυχία της απόδειξης

Πίνακας 1 - Χαρακτηριστικά των βασικότερων προτύπων προγραμματισμού

Πίνακας 2 - Το "Hello World" σε γλώσσες από τα βασικότερα πρότυπα προγραμματισμού

Προστακτικό (C)	**#include <stdio.h>
	#include <stdlib.h>
	int main(void)
	{
	printf("Hello, imperative world\n");
	return EXIT_SUCCESS;
	}**

Συναρτησιακό (F#)	**let helloworld ruready = if ruready = 'y' then "hello functional world" else "stickin with c#";;**
Λογικό (Prolog)	**hello_world :- write('Hello Logic World!').**

Αντικειμενοστρεφές (C#)	**class Hello
	{
	static void Main()
	{
	System.Console.WriteLine("Hello Object World!");
	}
	}**

Ιστορικό εκδόσεων της C#

- Από την πρώτη της έκδοση το 2000, η γλώσσα C# έχει εξελιχθεί αρκετά. Αυτή η εξέλιξη οφείλεται εν μέρει στην προτίμηση που της δείχνουν η προγραμματιστές μεταξύ των γλωσσών στο .NET. Τα περισσότερα χαρακτηριστικά που προστίθενται αντικατοπτρίζονται στο .NET Framework αφού, όπως προαναφέρθηκε, η γλώσσα C# είναι αναπόσπαστο κομμάτι του.

Το πρώτο πρόγραμμα σε C#

Λίστα 2.1 – HelloWorld.cs

```
1. using System;
2. class Helloworld {
3. static void Main(String[] args) {
4. // Εμφάνιση χαιρετισμού στο παράθυρο της κονσόλας
5.Console.WriteLine("Hello World!");
6. }
7. }
```

Το πρώτο πρόγραμμα σε C#

- **Παρόλο που το πρόγραμμα είναι σύντομο, περιλαμβάνει μερικά σημαντικά στοιχεία που είναι κοινά σε όλα τα προγράμματα σε C#.**
- **Ας εξετάσουμε κάθε κομμάτι του προγράμματος ξεκινώντας με το όνομά του. Το όνομα ενός προγράμματος C# είναι αυθαίρετο. Αντίθετα με άλλες γλώσσες (όπως η Java) στις οποίες το όνομα του αρχείου είναι σημαντικό, στη C# δεν παίζει κάποιο ρόλο.**
  Ωστόσο καλό είναι να είναι όσο το δυνατόν περιγραφικό για το περιεχόμενό του. Πολλοί προγραμματιστές ακολουθούν τη σύμβαση της Java και ονομάζουν το αρχείο σύμφωνα με την κύρια κλάση που περιέχει. Επίσης τα προγράμματα C# έχουν κατάληξη .cs.

- **Στη γραμμή 1, ορίζεται ότι το πρόγραμμα χρησιμοποιεί το namespace System. Στη C# Το namespace System είναι ένα δεσμευμένο namespace για αντικείμενα της βιβλιοθήκης κλάσεων του .NET. Η δεσμευμένη λέξη using απλώς ορίζει ότι θα χρησιμοποιηθεί το namespace που ακολουθεί. Σε επόμενες ενότητες θα δούμε πως ορίζουμε τα δικά μας namespaces, μια διαδικασία που είναι απαραίτητη για τη σωστή οργάνωση των προγραμμάτων.**

Το πρώτο πρόγραμμα σε C#

- **Στη γραμμή 2 χρησιμοποιείται η δεσμευμένη λέξη class για να δηλωθεί ότι ορίζεται μια νέα κλάση. Μια κλάση είναι η βασική μονάδα ομαδοποίησης μεταβλητών και μεθόδων, δηλαδή των ιδιοτήτων και των συμπεριφορών μιας αφηρημένης έννοιας. Μετά τη λέξη class ακολουθεί το όνομα της κλάσης, στη συγκεκριμένη περίπτωση Helloworld. Τα περιεχόμενα της κλάσης εσωκλείονται σε αγκύλες ({}). Τα στοιχεία μεταξύ των αγκυλών καλούνται μέλη (members) μιας κλάσης.**

- **Στη γραμμή 3 ξεκινά η Main μέθοδος. Μια μέθοδος είναι η βασική μονάδα ομαδοποίησης εντολών και αντιστοιχεί στη συμπεριφορά μιας αφηρημένης έννοιας. Η μέθοδος Main είναι μια ειδική μέθοδος η οποία προσδιορίζει το σημείο έναρξης ενός προγράμματος.**
  Όλες οι εφαρμογές C# ξεκινάνε την εκτέλεσή τους καλώντας τη μέθοδο Main().

Το πρώτο πρόγραμμα σε C#

- Η παράμετρος της μεθόδου String[] args περιέχει τα ορίσματα γραμμών εντολών (command line arguments). Η δεσμευμένη λέξη static προσδιορίζει ότι η μέθοδος Main μπορεί να κληθεί χωρίς την αρχικοποίηση της κλάσης. οι static μέθοδοι δεν είναι συχνές σε μεγάλα προγράμματα. Ωστόσο η main πρέπει να είναι πάντα static γιατί τρέχει πριν το πρόγραμμα μπορέσει να δημιουργήσει αντικείμενα.
- Η πρώτη γραμμή μέσα στη main (γραμμή 4) είναι ένα σχόλιο. Το σχόλιο αυτό είναι προς όφελος του αναγνώστη του πηγαίου κώδικα για να εξηγήσει τι κάνει η επόμενη εντολή. Κείμενο που ξεκινάει από // αγνοείται από το μεταγλωττιστή.

Το πρώτο πρόγραμμα σε C#

- **Οι εντολές στο κυρίως μέρος (body) της μεθόδου Main (δηλαδή οι εντολές μεταξύ των αγκίστρων {}) εκτελούνται μία προς μία. Κάθε εντολή λήγει με ένα ελληνικό ερωτηματικό (;). Η μέθοδος έχει μόνο μια εντολή στη γραμμή 5. Η εντολή αυτή εκτυπώνει μια γραμμή κειμένου και συγκεκριμένα την "Hello, World!".**

- **Ωστόσο υπάρχουν πολλά μέρη όπου θα μπορούσε το πρόγραμμα να στείλει αυτό το κείμενο: σε ένα παράθυρο, σε ένα αρχείο ή σε έναν δικτυωμένο υπολογιστή στην άλλη άκρη του κόσμου. Χρειάζεται επομένως να ορίσουμε ότι ο προορισμός είναι η *κονσόλα* (Console) δηλαδή το παράθυρο της γραμμής εντολών.**

- **Το παράθυρο κονσόλας αναπαρίσταται στη C# με την κλάση Console. Για να χρησιμοποιήσουμε μια κλάση όπως την Console, προσδιορίζουμε τι θέλουμε να κάνουμε με αυτήν. Στην περίπτωση μας θέλουμε να εκτυπώσουμε μια γραμμή κειμένου. Η static μέθοδος WriteLine πραγματοποιεί αυτήν την ενέργεια. Δεν χρειάζεται να υλοποιήσουμε αυτήν τη μέθοδο καθώς είναι έτοιμη, απλώς πρέπει να την *καλέσουμε*.**

Όταν καλούμε μεθόδους στη C# πρέπει να προσδιορίσουμε τρία πράγματα:

1. **Την κλάση ή το αντικείμενο που θα χρησιμοποιήσουμε (σε αυτήν την περίπτωση την κλάση Console)**

2. **Το όνομα της μεθόδου που θα χρησιμοποιήσουμε (σε αυτή την περίπτωση την WriteLine).**

3. **Ένα ζευγάρι παρενθέσεων, που περιέχουν κάθε άλλη πληροφορία που χρειάζεται η μέθοδος (σε αυτήν την περίπτωση το κείμενο "Hello, World!"). Αυτή η πληροφορία καλείται παράμετρος της μεθόδου.**

Μια ακολουθία χαρακτήρων μέσα σε ανωφερή εισαγωγικά (quotation marks) - γνωστά και ως διπλές αποστρόφους - καλείται string. Πρέπει να εσωκλείουμε το περιεχόμενο του string σε εισαγωγικά για να μπορεί ο compiler να τα διακρίνει από τις υπόλοιπες λέξεις και κυρίως τις δεσμευμένες λέξεις της γλώσσας.

Συνδυάζοντας το παράδειγμα με όσα αναφέραμε μπορούμε να καταλήξουμε στη συνηθέστερη γενική δομή ενός προγράμματος κονσόλας σε C#:

```
//δήλωση namespace των οποίων τις κλάσεις θα χρησιμοποιήσουμε
using example namespace;
//δήλωση namespace στο οποίο θα ανήκουν οι παρακάτω κλάσεις
namespace anothernamespace;
//δήλωση της public κλάσης
class RandomClass {
static void main(String[] args) {
// Κλήση εντολών
}
}
```

On line compiler https://dotnetfiddle.net/

```
using System;

public class Program
{
    public static void MyMethod()    // Αν έχω μεθόδους
    {
            Console.WriteLine("H PROTI MOY METHODOS!");
    }

    public static void Main() // H main μου
    {
            MyMethod();
    }
}
```

Παράδειγμα

```
usingSystem;

Namespace MyNamespace
{
      classHelloworld
         {
            staticvoid Main()
             {
             Console.WriteLine("Hello World");
             Console.ReadKey();

             }
          }
}
```

Χώροι ονομάτων (Namespaces):

- Σε ένα C# πρόγραμμα χρησιμοποιούμαι Namespaces για την οργάνωση του κώδικα μας.
  Στο παραπάνω πρόγραμμα χρησιμοποιήσαμε τον χώρο ονομάτων System (γραμμή 1 του παραδείγματος) μέσο της λέξης κλειδιού using ,το οποίο περιλαμβάνει έτοιμες κλάσεις και μεθόδους του .NET με έτοιμο επαναχρησιμοποιήσιμο κώδικα!

  Ένα Namespace μπορεί να περιέχει τους εξής τύπους :

- • Namespaces
  • Classes
  • Delegates
  • Enums
  • Structs
  • Interfaces

- Για τη δημιουργία ενός δικού μας Namespace χρησιμοποιούμε τη δεσμευμένη λέξη-κλειδί namespace (γραμμή 2), έτσι η κλάση Helloworld (γραμμή 4) εμπεριέχεται στο MyNamespace που δημιουργήσαμε.
- Στη συνέχεια του κώδικα μπορούμε να δούμε πως μπορούμε να εμφανίσουμε στη γραμμή εντολών το μήνυμα "Hello World". Η μέθοδος Main() (γραμμή 6) περιέχει την εντολή Console.WriteLine("Hello World"); , η οποία εμφανίζει το τελικό μήνυμα.

ΠΑΡΑΤΗΡΉΣΕΙΣ:

- Η κλάση Console ανήκει στο χώρο ονομάτων System και η WriteLine() είναι μέθοδος της Console.
  Εάν δεν χρησιμοποιούσαμε την εντολή usingSystem; Η εντολή θα συντασσόταν
  System.Console.WriteLine("Hello Word");

- Χρησιμοποιούμε την τελεία ανάμεσα από Namespaces, κλάσεις και μεθόδους για να δείξουμε τι εμπεριέχεται σε τι. Πχ.

Ψευδώνυμα (Aliases):

- Μπορούμε να χρησιμοποιούμε ψευδώνυμα (Aliases) για να κάνουμε τον κώδικα μας ακόμα πιο σύντομο και να αποφύγουμε να γράφουμε μακροσκελείς εντολές. Πχ.

```
using Alias=System.Console;

    namespace MyNamespace
    {
    public class Helloworld
            {
            staticvoid Main()
                    {
                    Alias.WriteLine("Hello World");
                    Console.ReadKey();
                    }
            }
    }
```

Σχόλια:

- Μπορούμε να χρησιμοποιούμε σχόλια στον κώδικά μας κυρίως για να εξηγούμε κάποια σημεία του, όπως τη λειτουργία μιας κλάσης/μεθόδου που δημιουργήσαμε ,έτσι ώστε να γίνεται ο κώδικας εύκολος στην ανάγνωση τόσο για μας ,αν θελήσουμε κάποια στιγμή να τον ανατρέξουμε να δούμε τη λειτουργία του, αλλά και για κάποιο τρίτο πρόσωπο που θα χρειαστεί να τον δει.

  Τα σχόλια συντάσσονται χρησιμοποιώντας αυτόν τον συμβολισμό (//) ,εάν επρόκειτο για μία γραμμή σχολίου:
  //Αυτό είναι ένα σχόλιο

  ή αν το σχόλιο απαιτεί παραπάνω γραμμή με αυτόν τον τρόπο:
  /* Αυτό είναι
  ένα σχόλιο */

Δήλωση μεταβλητών, Ανάθεση-Σύγκριση τιμών & Εμφάνιση αποτελεσμάτων στην οθόνη:

- Σε ένα C# πρόγραμμα μπορούμε να δηλώσουμε μεταβλητές σε οποιοδήποτε σημείο του προγράμματος , πριν τις χρησιμοποιήσουμε για κάποια πράξη.

- Για να δηλώσουμε μία μεταβλητή ορίζουμε πρώτα τον τύπο της και ύστερα το όνομα της.
  πχ. int x; string y1; long z_2;

Δήλωση μεταβλητών, Ανάθεση-Σύγκριση τιμών & Εμφάνιση αποτελεσμάτων στην οθόνη (2)

- Για να κάνουμε ανάθεση τιμής σε μεταβλητές χρησιμοποιούμε το ίσον (=)
  πχ. x = 5; Ή η ίδια εντολή ως int x = 5; άν την δηλώναμε ταυτόχρονα μαζί με την
  ανάθεση της τιμής. char y = 'h'; Κτλ.

- Για σύγκριση μεταβλητών ισχύουν :
  >= μεγαλύτερο ή ίσο
  <= μικρότερο ή ίσο
  == ίσο
  != διάφορο

- Για να εμφανίσουμε το περιεχόμενο μεταβλητών στην οθόνη η C# χρησιμοποιεί τη μέθοδο WriteLine() με δύο τρόπους σύνταξης.
- Παρακάτω θα δούμε σε ένα απλό πρόγραμμα που εμφανίζει το άθροισα δύο ακεραίων τον τρόπο δήλωσης μεταβλητών, την ανάθεση τιμών σε αυτές και τέλος πώς να εμφανίζουμε μεταβλητές με τη χρήση της WriteLine() :

```
using System;
namespaceMyNamespace
{
class Sum_x1_x2
{
staticvoidMain()
{
// 1ος τρόπος δήλωσης στην αρχή του προγράμματος
int x1 =5;
int x2 =3;
int y = x1 + x2;
// 1ος τρόπος σύνταξης της WriteLine
Console.WriteLine(" Το αποτέλεσμα είναι : {0} ", y);
/* Το output είναι: Το αποτέλεσμα είναι : 8
  Στη θέση του συμβόλου {0} η WriteLine
  τοποθετεί την τιμή που έχει το y.Το μηδέν
  μέσα στα άγκιστρα δείχνει τη θέση

  της μεταβλητής μετά το κόμμα μέσα στη Writeline.
  Αν θέλαμε να εμφανίσουμε παραπάνω μεταβλητές
  θα κάναμε το εξής:
  Console.WriteLine("{0} + {1} = {2}",x1,x2,y);
  Το output εδώ θα ήταν : 5 + 3 = 8
  Στη θέση του {0} πάει η τιμή της x1, στο {1} η x2
  και στο {2} η y.          */
```

```
string z ="!!!";/*2ος τρόπος δήλωσης κατά τη ροή του
                                προγράμματος */
// 2ος τρόπος σύνταξης της WriteLine
Console.WriteLine("Το αποτέλεσμα είναι : {0}"+z ,y);
/* Γιατί +  z???
 Στη WriteLIne οτιδήποτε είναι μέσα σε εισαγωγικά
 αποτελεί string. Η μεταβλητή z είναι επίσης ένα
 string. Βλέπουμε λοιπόν ότι μπορούμε να προσθέσουμε
 δύο strings μαζί και το αποτέλεσμα θα έιναι
 το ένα string να "κολλήσει" πίσω από το άλλο.
 Το output εδώ θα δώσει : Το αποτέλεσμα είναι :
             */
Console.ReadKey();
}
}
}
```

ΠΑΡΑΤΗΡΗΣΕΙΣ:

- Η C# είναι Casesensitive ,
  Για να δηλώσουμε την Main() χρησιμοποιήσαμε κεφαλαίο Μ! Θα ήταν λάθος να λέγαμε static void main().
- ο compiler εξ ορισμού καταλαβαίνει ότι το βασικό πρόγραμμα βρίσκεται στην μέθοδο Main() και όχι στην main().
- Εννοείται πως int y ≠ int Y !

Keywords,

- Στη C# υπάρχουν δεσμευμένες λέξεις-κλειδιά.
  Δεν μπορούμε να δηλώσουμε πχ μια μεταβλητή με όνομα Console/int/void/class,
  όπως επίσης δεν μπορούμε να βάζουμε αριθμό μπροστά από μεταβλητή!

Σταθερές (Constans),

Η δήλωση των σταθερών στο C# πρόγραμμα όπως πχ. Η τιμή του π=3.14. γίνεται με αυτόν τον τρόπο :

- const double p = 3.14;

Στη WriteLine() υπάρχουν 2 τρόποι εμφάνισης μεταβλητών.

- 1ος τρόπος ως συνεχόμενο string
- 2ος τρόπος με int μεταβλητές :
  πχ.
  int x = 1;
  int y = 2;
  int sum = x + y;

- //1ος τρόπος
  Console.WriteLine(x + "+" + y + "=" + sum);
  // outuput 1+2=3

  //2ος τρόπος
  Console.WriteLine("{0}+{1}={2}",x,y,sum);
  //output 1+2=3

Datatypes (τύπους δεδομένων) C#

Integers (ακέραιοι)

Τύποι	.NET	Μέγεθος	Σειρά
sbyte	System.SByte	1 byte	-128 μέχρι 127
byte	System.Byte	1 byte	0 μέχρι 255
short	System.IUInt16	2 byte	-32,768 μέχρι 32,767
ushort	System.Uint16	2byte	0 μέχρι 65,535
int	System.Int32	4 byte	-2,147,483,648 μέχρι 2,147,483,647
uint	System.Uint32	4 byte	0 μέχρι 4,294,967,295
long	System. Int64	8 byte	-9,223,372,036,854,775,808 μέχρι 9,223,372,036,854,775,807
ulong	System.Uint64	8 byte	0 μέχρι 18,446,744,073,709,551,615

Floatingpoint Number (μεταβλητές κινητής υποδιαστολής)

Τύποι	.NET	Μέγεθος	Σειρά
Float	System.Single	4 byte	1.5×10^{-45} μέχρι 3.4×10^{38}
Double	System.Double	8 byte	5.0 x 10-324 μέχρι 1.7 x 10308
Decimal	System.Decimal	16 byte	1.0×10^{-28} μέχρι 7.9×10^{28}

- Οι μεταβλητές περιέχουν μια τιμή, η οποία μπορεί να μεταβληθεί κατά την διάρκεια εκτέλεσης του προγράμματος.
- Για να χρησιμοποιηθεί όμως μια μεταβλητή πρέπει πρώτα να δηλωθεί, να της δοθεί δηλαδή ένα όνομα και να επιλεγεί ο τύπος δεδομένων που θα αποθηκεύει. Η δήλωση γίνεται γράφοντας έναν τύπο δεδομένων ακολουθούμενο από ένα όνομα.

Παράδειγμα (Byte):

- **staticvoid** Main()
- {
- **byte**MyNumber=254;
- Console.WriteLine(MyNumber);//Εμφανίζει 254
- MyNumber++; // Συν 1 -> 255
- Console.WriteLine(MyNumber);//Εμφανίζει 255
- MyNumber++; // Συν 1 -> 0 (όριο 255)
- Console.WriteLine(MyNumber);//Εμφανίζει 0
- MyNumber++; // Συν 1 -> 1
- Console.WriteLine(MyNumber);//Εμφανίζει 1
- }

Παράδειγμα (int/float)

- **Με Int**
  static void Main()
- **//Σε περίπτωση που Δηλώσουμε 2 ακεραίους INT**
- **{ //Δηλώνουμε 2 ακεραίους INT**
- **int A =10;**
- **int B =3; //διαιρούμε 10/3**
- **Int apotelesma= A/B; // Μαςεμφανίζει 3 και Οχι 3,333**
- **Console.WriteLine(apotelesma);**
- **}**

Με Float

- **Static void Main()**
- **//Σε περίπτωση που Δηλώσουμε 2 ακεραίους Single**
- **{ //Δηλώνουμε 2 ακεραίους Single ή double ή float**
- **Single A =10;**
- **Single B =3;**
- **//διαιρούμε 10/3**
- **Single apotelesma= A/B;**
- **// Μας δίνει 3,333**
- **Console.WriteLine(apotelesma);**

}

Παράδειγμα (Int/Long):

```
Static void Main()
{
int x =10000;
int y =20000;
        //Σε αυτό οι μεταβλητές "int" μετατρέπονται σε "long"
   //Δεν χρειάζεται conversion από το 'int' σε
   // long',γίνεται αυτόματα.
int total;
total= x * y;
Console.WriteLine("Το αποτέλεσμα είναι:"+total);
}
```

Strings&chars (Συμβολοσειρές και χαρακτήρες)

- Συμβολοσειρά (**string**) είναι μια σειρά αλφαριθμητικών χαρακτήρων (γενικά εκτυπώσιμων συμβόλων ASCII). Όταν λέμε σειρά εννοούμε διαδοχικές θέσεις μνήμης που μπορούν να αντιμετωπισθούν σαν ένα σύνολο διατεταγμένων στοιχειωδών δεδομένων.
- Η συμβολοσειρά είναι ένας ειδικός τύπος πίνακα με δεδομένα του τύπου **char** και με τερματισμό μέσω του μηδενικού χαρακτήρα. Παράδειγμα:

```
Static void Main()
{
//το A = StudentGuru
String A="StudentGuru";
//B = με το πρώτο χαρακτήρα της μεταβλητής A άρα = "S"
char B = A[0];
Console.WriteLine(B);// εκτυπώνει στην κονσόλα.
Console.WriteLine(A);// εκτυπώνει στην κονσόλα.
//Τώρα το B = με το τρίτο χαρακτήρα της μεταβλητής A άρα ="u"
B=A[2];
Console.WriteLine(B);// εκτυπώνει στην κονσόλα.

}
```

Bool τιμές

Ο τύπος bool έχει δύο σταθερές τιμές:
την **true** και την **false**
Σημείωση: *true = αληθές και false = εσφαλμένο, ψευδές.*
Παράδειγμα:

```
staticvoid Main()
{
boolCheckNumber;

Console.WriteLine("Typosestudentguru");
stringnum=Console.ReadLine();
//εάνημεταβλητή = studentguru
if(num=="studentguru")
{
CheckNumber=true;//τότετοcheckNumber = true
}
else
{
CheckNumber=false;//διαφορετικάcheckNumber = false
}
Console.WriteLine(CheckNumber);
}
```

NullableTypes

- Ο όρος **Null** είναι μια ενδιαφέρουσα ιδέα στον κόσμο του προγραμματισμού και δεν είναι συνώνυμο με το «μηδέν» ή «κενό», αλλά «απροσδιόριστη».
- Ευτυχώς, η C # παρέχει ένα πολύ χρήσιμο εργαλείο:. NullableTypes δεδομένων. Αυτοί οι τύποι δεδομένων μπορούν να χρησιμοποιηθούν σε οποιοδήποτε σημείο του κώδικα όπου υπάρχει ανάγκη να έχουμε μια μεταβλητή που μπορεί να έχει αξίες.

Παράδειγμα Nullable types

```
staticvoid Main(string[]args)
{
int? num=null; //Nullable Types Σύνταξη
//Έλεγχος εάν η μεταβλητή num έχει τιμή
if(num.HasValue==true)
//αν ναι τύπωσε μας την τιμή
System.Console.WriteLine(num.Value);
        Else
//αν όχι τύπωσε NULL
System.Console.WriteLine("NULL");
}
```

Άμεση Μετατροπή

```
int? n =null;
//int m1 = n;       // Λάθος

int m2 =(int)n;
int m3 =n.Value;
```

Έμμεση μετατροπή

```
int? n1 =null;

int? n2;
n2 =10;//Έμμεση μερατροπή
```

“??” Τελεστές Nullable

```
int? c =null;
// d = c, και το c είναι null, το d = -1.
int? e =null;
int? f =null;
// g = e ή και f, εκτός εάν το e και το f = null, που για κάθε
   περίπτωση g = -1.
bool? b =null;
if(b)// Λάθος
{
}
```

Var Types

- Η μεταβλητή δεν είναι άλλο από μια ονομασμένη θέση μνήμης που μπορεί να περιέχει δεδομένα οποιουδήποτε είδους. Με τον τύπο Var δεν χρειάζεται να σκεφτούμε πολύ σκληρά το πώς θα δηλώσουμε την μεταβλητή μας. Η var είναι εύχρηστη, διότι δεν χρειάζεται να αποφασίσετε αν θα χρησιμοποιήσετε System.String ή System.Int32. Ο compiler καθορίζει τον τύπο για μάς.

Παράδειγμα VarTypes

```
staticvoid Main()
{
Var leksi= “studentGuru”;
Var Arithmos=21;
Var MegArithmo=219283746253;
Var dekadiko=2,19283746253;
}
```

Data Type Conversion and Format

- Η C# είναι μια γλώσσα που εξ ορισμού οτιδήποτε δεδομένα δέχεται από το χρήστη τα δέχεται ως strings! Για το λόγο αυτό έχουν σχεδιαστεί αρκετές μέθοδοι με τις οποίες μπορούμε να διαχειριστούμε και να μετατρέψουμε διαφορετικούς τύπους δεδομένων ανάλογα με το τι θέλουμε να κάνουμε.

Έστω ότι θέλουμε να προσθέσουμε δύο αριθμούς τους οποίους μας δίνει ο χρήστης...

```
namespaceMyNamespace{
publicclass Sum_x1x2 {
staticvoidMain() {  //Δηλώνουμε τις μεταβλητές ως ακέραιες
int x1 ;
int x2 ;
 Console.WriteLine("Δώσε δύο ακέραιες τιμές");
 //Με την Console.ReadLine() παίρνουμε τις τιμές
        x1=Console.ReadLine();<- ΛΑΘΟΣ
        X2=Console.ReadLine();<- ΛΑΘΟΣ
        Console.WriteLine("Το αποτέλεσμα είναι", x1+x2);
}
}
}
```

Γιατί ο compiler βγάζει λάθος;

- Όπως προαναφέραμε , οτιδήποτε τιμές διαβάζουμε από το χρήστη η C# τις δέχεται ως strings! Έτσι λοιπόν λέγοντας x1 = Console.ReadLine(); στην ουσία προσπαθούμε να βάλουμε ένα string μέσα σε μία int μεταβλητή!
- Για το λόγο αυτό υπάρχουν έτοιμες κάποιες μέθοδοι, οι οποίες λύνουν αυτό το πρόβλημα και θα τις δούμε παρακάτω.

Parse & Conversion Methods

- Οι parsemethods δέχονται ως είσοδο string τιμές και τις μετατρέπουν στον τύπο που θέλουμε. Έτσι στο παραπάνω πρόγραμμα για να πάρουμε τις ακέραιες τιμές θα πούμε :

  x1 = Int32.Parse(Console.ReadLine());
  x2 = Int32.Parse(Console.ReadLine());

- Με τον τρόπο μετατρέπουμε την string τιμή που έδωσε ο χρήστης μέσω της ReadLine() σε int τιμή και την αποθηκεύουμε στην μεταβλητή χ1 (και χ2 αντίστοιχα) .

Οι parsemethods συντάσσονται παρόμοια και για τους υπόλοιπους τύπους μεταβλητών.

- double.Parse() , για μετατροπή από string σε double.
- long.Parse() , για μετατροπή από string σε long κλπ.

ConversionMethods:

- Με τις συγκεκριμένες μεθόδους μπορούμε να μετατρέψουμε οποιονδήποτε τύπο μεταβλητής σε κάποιον άλλο.
  πχ int σε double , int σε float κλπ

  Με τον μόνο περιορισμό ότι δεν μπορούμε φυσικά να μετατρέψουμε ένα string,
  πχ. “Stavros” σε αριθμητικό τύπο (!)
  εκτός και αν αυτό το string περιέχει μόνο αριθμό,
  πχ. ‘’25” σε int μπορεί να γίνει.

Σύνταξη της Convert

- Η Convert είναι μέθοδος του namespaceSystem
  Αν θέλουμε πχ. Να μετατρέψουμε έναν int έστω x=5 σε double γράφουμε

- **double y = Convert.ToDouble(x);**
- Στη θέση της λέξης Double μπαίνουν και οι υπόλοιποι τύποι του .Net (Int32,Single κλπ)

Π Ρ Ο Σ Ο Χ Η!

- **Μετατροπές από double, float κλπ. σε int προκαλεί απώλειες δεδομένων!**

```
double z =5.4;
int a = Convert.ToInt32(z);
Console.WriteLine("ΑΠΟ DOUBLE ΣΕ INT");
Console.WriteLine("double {0} ",z);//output
Console.WriteLine("int {0} ",a);
/*output 5 (γίνεται στρογγυλοποίηση και απαλείφονται
  τα δεκαδικά*/
```

Casting

- Το casting στη C# βοηθάει στο να μετατρέψουμε έναν τύπο μιας μεταβλητής σε κάποιον άλλο στιγμιαία μέσα στο κώδικα. Αυτή η αλλαγή παύει να ισχύει μόλις εκτελεστεί η εντολή που περιέχει κάστα και η μεταβλητή αποκτά ξανά τον τύπο που είχε αρχικά.

Σύνταξη:

(float)x

η κάστα μπαίνει μπροστά από τη μεταβλητή (η μια αριθμητική πράξη πχ (float) x/y) με παρενθέσεις και μέσα βάζουμε τον τύπο που θέλουμε.

Παράδειγμα casting:

- Έστω b,c δύο ακέραιοι τους οποίους θέλω να διαιρέσω και να εμφανίσω.

```
int b =10;
int c =3;
Console.WriteLine("Αποτέλεσμα διαίρεσης {0}", b/c);//
    output 3 kai oxi 3,33333
```

Παράδειγμα casting(2)

- Αυτό συνέβη γιατί οι μεταβλητές b και c είναι ακέραιες. Επομένως η διαίρεση είναι ακέραια και απαλείφονται τα δεκαδικά. Αν θέλουμε κανονική διαίρεση θα έπρεπε να πούμε:

Console.WriteLine("Αποτέλεσμα διαίρεσης {0}",(**float**) b/c);// output 3,33333

- Το αποτέλεσμα της πράξης γίνεται float με τη χρήση της κάστας και είναι σαν να διαιρούνται πραγματικοί αριθμοί και όχι ακέραιοι. Το b και το c μετά την εκτέλεση της WriteLine εξακολουθούν να είναι int.

StringMethods

- ToString() Method
- Χρησιμοποιούμε την ToString() μέθοδο για να μετατρέψουμε μια μεταβλητή οποιουδήποτε τύπου ή ολόκληρη αριθμητική πράξη σε string.
- **Σύνταξη:** μεταβλητή.ToString();

```
int x =1;
int y =2;
string sum1 =(x + y).ToString();
Console.WriteLine(sum1);// output 3
//εδώ κάναμε το αποτέλεσμα της πρόσθεσης 2 ακεραίων
   string
string sum2 =x.ToString()+y.ToString();
Console.WriteLine(sum2);// output 12/*εδώ
   μετατρέψαμε ξεχωριστά τους ακεραίους σε strings και
   τα ενώσαμε*/
```

Σημείωση: τα x, y εξακολουθούν να είναι int μετά την εκτέλεση της ToString. Το αποτέλεσμα που δίνει η ToString αποθηκεύεται σε string μεταβλητές (sum1,sum2)

String.Format() :

- Η String.Format() ανήκει στο namespaceSystem και δέχεται σαν όρισμα ολόκληρη παράσταση όπως ακριβώς η WriteLine() και το αποτέλεσμα της είναι ένα string.

```
int x =1;
int y =2;
string output =String.Format("{0}+{1}={2}",x,y,x+y);
Console.WriteLine(output);//output 1+2=3
```

String.Compare()

- Τη χρησιμοποιούμε για να συγκρίνουμε δύο string μεταβλητές.

-Σύνταξη: **String.Compare(x,y)** ,όπου x , y strings.

Αν το x είναι μεγαλύτερο από το y η μέθοδος δίνει 1
Αν το x είναι μικρότερο από το y η μέθοδος δίνει -1
Τέλος αν είναι ίσα τα δύο strings δίνει 0

```
string x ="Maria";
string y ="Giorgos";
string z ="Kostas";
Console.WriteLine(String.Compare(y,x));//output -1
Console.WriteLine(String.Compare(x, z));//output 1
Console.WriteLine(String.Compare(x,"Maria"));//output 0
```

Trimming

- Εάν θέλουμε να απαλείψουμε τυχόν κενά που πληκτρολογεί ο χρήστης εισάγοντας ένα string υπάρχουν τρείς μέθοδοι και συντάσσονται ως εξής:

Μεταβλητή**. TrimStart()** , όπου απαλείφει τα κενά στην αρχή ενός string

Μεταβλητή**. TrimEnd()** , όπου απαλείφει τα κενά στο τέλος ενός string

Μεταβλητή**. Trim()** , όπου απαλείφει τα κενά στην αρχή και το τέλος ενός string.

TextCaseConversion

- Στη C# υπάρχουν δύο μέθοδοι οι οποίες μπορούν να μετατρέψουν τα γράμματα ενός string σε κεφαλαία ή μικρά ανάλογα με το πώς θέλουμε να εμφανιστούν

1) μεταβλητή. **ToLower**(); ,για μικρά
2) μεταβλητή. **ToUpper**(); ,για κεφαλαία

Έστω string x = “Hello World”

- Console.WriteLine(x.ToUpper());//output : HELLO WORLD
- Console.WriteLine(x.ToLower());//output : hello world