



C# .NET Core Πρωταρχικοί Τύποι Δεδομένων

Αθ. Ανδρούτσος



Τύποι Δεδομένων

Προγραμματισμός με C#.NET Core

- Η C# μας παρέχει πρωταρχικούς και σύνθετους τύπους δεδομένων
- Οι πρωταρχικοί τύποι λειτουργούν ως ψευδώνυμα (alias) των κλάσεων (structs) του .NET στις οποίες αντιστοιχούν. Άρα πρόκειται στην πραγματικότητα για κλάσεις, οπότε μπορούμε να κάνουμε **invoke methods** στους πρωταρχικούς τύπους
- Περιλαμβάνεται στους πρωταρχικούς τύπους και ο τύπος **decimal (128-bit)** που είναι **floating point** για μεγάλους αριθμούς ενώ δεν έχει τα προβλήματα αναπαράστασης και σύγκρισης που έχουν οι **float / double**



Πρωταρχικοί Τύποι Δεδομένων

Προγραμματισμός με C#.NET Core

C# type keyword	.NET type
bool	System.Boolean
byte	System.Byte
sbyte	System.SByte
char	System.Char
decimal	System.Decimal
double	System.Double
float	System.Single
int	System.Int32
uint	System.UInt32

long	System.Int64
ulong	System.UInt64
short	System.Int16
ushort	System.UInt16

- Πρόκειται για το βασικό *Type System* της C# και του .NET με τύπους δεδομένων για αριθμούς, τιμές αλήθειας και χαρακτήρες



Τύποι Δεδομένων

Πρωταρχικοί Τύποι

Προγραμματισμός με C#.NET Core

- **int** (32-bits) – byte (8 bits), short (16 bits), long (64 bits)
 - Για int: `Int32.MinValue` / `Int32.MaxValue`
- **char** (16-bits)
 - Α, ω, 2, !, Β, @, κλπ
- Δεκαδικοί αριθμοί **float/double/decimal**
 - 32/64/128 bits
- Τιμές αλήθειας (**bool**)
 - Αληθές, ψευδές



Int Subtypes

Προγραμματισμός με C#.NET Core

Τύπος	Μέγεθος - sizeof()	Range	Min / Max Values
int	32 bits	-2,147,483,648 to 2,147,483,647	Int32.MinValue / Int32.MaxValue
uint	32 bits	0 to 4,294,967,295	UInt32.MinValue / UInt32.MaxValue
sbyte	8 bits	-128 to 127	Sbyte.MinValue / Sbyte.MaxValue
byte	8 bits	0 to 255	Byte.MinValue / Byte.MaxValue
short	16 bits	-32,768 to 32,767	Int16.MinValue / Int16.MaxValue
ushort	16 bits	0 to 65,535	UInt16.MinValue / UInt16.MaxValue
long	64 bits	9,223,372,036,854,775,808 to 9,223,372,036,854,775,807	Int64.MinValue / Int64.MaxValue
ulong	64 bits	0 to 18,446,744,073,709,551,615	UInt64.MinValue / UInt64.MaxValue



Σύνθετοι Τύποι Reference Types

Προγραμματισμός με C#.NET Core

- Δείκτες στη δυναμική μνήμη (heap)
 - Αλφαριθμητικά (string)
 - Ακολουθίες χαρακτήρων
 - Πίνακες (Arrays)
 - Ακολουθίες δεδομένων ίδιου πρωταρχικού τύπου, π.χ. πίνακας Ακεραίων , `int[]`
 - Κλάσεις (Class) και τύπος `object`
 - Κλάσεις: Δεδομένα διαφορετικού πρωταρχικού τύπου, π.χ. Οδός (String), Αριθμός (Χαρακτήρας)



Nullable Types

Προγραμματισμός με C#.NET Core

- Η C# .NET 8.0 μας δίνει τη δυνατότητα να ορίζουμε πρωταρχικούς τύπους που να μπορούν να πάρουν την τιμή null ώστε να μπορούμε (ιδιαίτερα όταν εργαζόμαστε με ΒΔ) να εκχωρούμε τιμές null
 - `Nullable<int> num = null;`
 - `int? num = null;`



Placeholders

Προγραμματισμός με C#.NET Core

```
1 namespace HelloWorld
2 {
3     internal class Program
4     {
5         static void Main(string[] args)
6         {
7             int num = 10;
8             Console.WriteLine("num = {0}", num);
9         }
10    }
11 }
```

- Το {0} είναι placeholder και αντιστοιχεί στην 1^η παράμετρο μετά το κόμμα
- Αντίστοιχα τα {1} {2} αντιστοιχούν σε επόμενες παραμέτρους



Πρωταρχικοί Τύποι Δεδομένων

Προγραμματισμός με C#.NET Core

```
1  using System;
2
3  namespace IntLimits
4  {
5      /// <summary>
6      /// Prints the limits of int, byte, short, long,
7      /// uint, ushort, ulong, sbyte
8      /// </summary>
9      class Program
10     {
11         static void Main(string[] args)
12         {
13             Console.WriteLine($"int: {sizeof(int) * 8} bits\t{int.MinValue}\t{int.MaxValue}");
14             Console.WriteLine($"byte: {sizeof(byte) * 8} bits\t{Byte.MinValue}\t{Byte.MaxValue}");
15             Console.WriteLine($"short: {sizeof(short) * 8} bits\t{Int16.MinValue}\t{Int16.MaxValue}");
16             Console.WriteLine($"long: {sizeof(long) * 8} bits\t{Int64.MinValue}\t{Int64.MaxValue}");
17             Console.WriteLine($"uint: {sizeof(uint) * 8} bits\t{UInt32.MinValue}\t{UInt32.MaxValue}");
18             Console.WriteLine($"ushort: {sizeof(ushort) * 8} bits\t{UInt16.MinValue}\t{UInt16.MaxValue}");
19             Console.WriteLine($"ulong: {sizeof(ulong) * 8} bits\t{UInt64.MinValue}\t{UInt64.MaxValue}");
20             Console.WriteLine($"sbyte: {sizeof(sbyte) * 8} bits\t{SByte.MinValue}\t{SByte.MaxValue}");
21             Console.WriteLine($"float: {sizeof(float) * 8} bits\t{Single.MinValue}\t{float.MaxValue}");
22             Console.WriteLine($"double: {sizeof(double) * 8} bits\t{Double.MinValue}\t{double.MaxValue}");
23             Console.WriteLine($"decimal: {sizeof(decimal) * 8} bits\t{Decimal.MinValue}\t{decimal.MaxValue}");
24             Console.WriteLine($"char: {sizeof(char) * 8} bits");
25             Console.WriteLine($"boolean: {sizeof(bool) * 8} bits\t{true}\t{false}");
26         }
27     }
28 }
29
```



Μεταβλητές απλών τύπων

Προγραμματισμός με C#.NET Core

- Θέσεις μνήμης που περιέχουν την τιμή ενός συγκεκριμένου απλού τύπου δεδομένων
- Μία μεταβλητή αποτελείται από τον τύπο δεδομένων και το όνομά της (π.χ. `int sum`)

Μνήμη

sum	30

← `int sum = 30;`
Με αυτή την εντολή ο μεταγλωττιστής της C# **δεσμεύει στη μνήμη** χώρο για έναν **ακέραιο** και ονομάζει αυτό τον χώρο **sum**. Επίσης εκχωρεί την **τιμή 30**.

Ως χώρο που δεσμεύεται στη μνήμη **απεικονίζουμε συνήθως ένα κελί (byte)** της μνήμης, στην πραγματικότητα όμως δεσμεύονται περισσότερα bytes, ανάλογα με τον τύπο δεδομένων. Για παράδειγμα οι ακέραιοι (`int`) καταλαμβάνουν 4 bytes στη μνήμη (32-bits)



Μεταβλητές σύνθετων τύπων (1)

Προγραμματισμός με C#.NET Core

- Οι θέσεις μνήμης περιέχουν δείκτες ενός συγκεκριμένου τύπου δεδομένων
Ο δείκτης δείχνει σε θέση στη δυναμική μνήμη
- Ο σωρός (heap) περιέχει την πραγματική τιμή



```
string s = "Hello";
```

Με αυτή την εντολή ο μεταγλωττιστής της C# δεσμεύει στη μνήμη χώρο για έναν δείκτη σε `string` και ονομάζει αυτό τον χώρο **s**.

Επίσης εκχωρεί την τιμή **a9cf** που είναι η διεύθυνση μνήμης στο **heap** του **"Hello"**.



Μεταβλητές σύνθετων τύπων (2)

Προγραμματισμός με C#.NET Core

- Automatic Garbage Collection (Συλλογή σκουπιδιών)
- Τα strings είναι immutable. Αν δώσουμε νέα τιμή στο string `s`, τότε αλλάζει ο δείκτης και δείχνει στη νέα τιμή.
- Η παλιά τιμή στο heap πρέπει να γίνει delete

s Μνήμη

string@b8ff

Heap

Hello
Hello-2

s="Hello-2";

Με αυτή την εντολή ο μεταγλωττιστής της C# αλλάζει την τιμή του δείκτη να δείχνει στη νέα θέση.

Η προηγούμενη θέση μένει χωρίς αναφορά. Αν δεν διαγραφεί στο τέλος η δυναμική μνήμη θα γεμίσει



Strings Interning

Προγραμματισμός με C#.NET Core

- Η C# .NET χρησιμοποιεί *string interning* για να βελτιστοποιήσει την αποθήκευση string literals





Κυριολεκτικά - Literals

Προγραμματισμός με C#.NET Core

- bool: true/false
- int: 0xA8F1 δεκαεξαδικοί
 - long 20L
 - uint 100u, ulong 100ul
- float/double/decimal (0.0f/0.0d/0.0m)
 - $1.3e-5$ ($=1.3 \cdot 10^{-5}$)
- char 'x'
 - \u0064 unicode char
- string "alice"



Ακολουθίες διαφυγής Escape sequence (1)

Προγραμματισμός με C#.NET Core

- Ο χαρακτήρας \ επιτρέπει την εμφάνιση χαρακτήρων με ειδικό νόημα
 - \' - Μονό εισαγωγικό
 - \" – Διπλά εισαγωγικά
 - \\ - backslash
 - \n – νέα γραμμή
 - \t - tab



Ακολουθίες διαφυγής

Escape sequence (2)

Προγραμματισμός με C#.NET Core

- `string s1 = "Μου φώναξε: \"Bob\"";`
- `string path = "C:\\windows\\thanos";`
- `string path = @"C:\windows\thanos";`
- Το **@** **μπροστά από ένα string** προστάζει τον compiler να κάνει `interpret` το string literally και δεν κάνουμε `escape` σε αυτή την περίπτωση



Τελεστές (1)

Προγραμματισμός με C#.NET Core

- Οι τελεστές δίνουν τη δυνατότητα πράξεων μεταξύ μεταβλητών (τελεσταίων)
- Τύποι Τελεστών:
 - **Αριθμητικοί** (+, -, /, *, %, ++, --)
 - **Σύγκρισης** (<, <=, >, >=, !=, ==)
 - **Λογικοί** (&&, ||, !)
Λογικό ΚΑΙ, Λογικό Η, Λογική Άρνηση



Τελεστές (2)

Προγραμματισμός με C#.NET Core

- Δυαδικοί (&, |, ^, ~, <<, >> (^ XOR, ~ NOT, << shift left, >> shift right)
- Συνένωση strings (+)
- Τύποι: (type), typeof, sizeof
- Άλλοι: ., new, [], ? :, (), ??



Τελεστές (3)

Προγραμματισμός με C#.NET Core

- ?? (null coalescing operator, σαν τον τριαδικό αλλά εκχωρεί το αριστερό μέρος μόνο αν δεν είναι null)
 - `st = st1 ?? new Student();`
 - Δηλ. `st = (st1 != null) ? st1 : new Student();`



Bitwise operators

shift left right

Προγραμματισμός με C#.NET Core

- $1 \ll 3$
- $0000\ 0001 \rightarrow 0000\ 1000$
 - Ο νέος αριθμός είναι το 8
 - Είναι σαν να έγινε η πράξη $1 * 2^3$
- $16 \gg 2$
- $0001\ 0000 \rightarrow 0000\ 0100$
 - Ο νέος αριθμός είναι το 4
 - Είναι σαν να έγινε η πράξη $16 / 2^2$



Τριαδικός / Δυαδικός τελεστής ?:

Προγραμματισμός με C#.NET Core

- `int a = -12;`
`Console.WriteLine((a >= 0) ? a : -a);`



Προτεραιότητα τελεστών

Προγραμματισμός με C#.NET Core

$x * y, x / y, x \% y$	Multiplicative
$x + y, x - y$	Additive

- Στις παραστάσεις προτεραιότητα έχουν οι πολλαπλασιασμοί, διαιρέσεις, mod και μετά προσθέσεις, αφαιρέσεις
- Συνίσταται η χρήση παρενθέσεων για τον ορισμό προτεραιοτήτων



Associativity

Προγραμματισμός με C#.NET Core

- Σε ίσης προτεραιότητας τελεστές έχουμε αριστερό associativity, δηλαδή οι πράξεις εκτελούνται από αριστερά προς τα δεξιά



Type Casting (1)

Προγραμματισμός με C#.NET Core

- Έμμεσο
 - `int a = 5;`
 - `long myLong = a;`
- Άμεσο με `(type)` -υπάρχει περίπτωση απώλειας ακρίβειας, π.χ. `long -> int`
 - `long n = 10;`
 - `int a = (int) n;`
- suffix
 - `long n = 5.5L;`



Type Casting (2)

Προγραμματισμός με C#.NET Core

- Στις παραστάσεις όταν υπάρχει ένας `int` όλοι μετατρέπονται σε `int`
- Αν υπάρχει `long` όλα μετατρέπονται σε `long`
- Αν υπάρχει `float` ή `double` όλα γίνονται `promote` σε `float` ή `double`



Type Casting (3)

Προγραμματισμός με C#.NET Core

- Τα αριθμητικά literals αν είναι ακέραιοι, θεωρούνται int και αν είναι δεκαδικοί θεωρούνται double
- Literals μετατρέπουμε με type-suffix
 - L ή l για long
 - D ή d για double
 - F ή f για float
 - M ή m για decimal



Μετατροπή σε string

Προγραμματισμός με C#.NET Core

- `.ToString()`
- Έμμεσα μέσα στην `WriteLine()`
- Έμμεσα με τον τελεστή συνένωσης `+`
- Άμεσα με το `.ToString()`
 - `5.ToString()`



Μετατροπή από String

Προγραμματισμός με C#.NET Core

- **Parse()**
 - `int.Parse("100");`
 - `short.Parse("-100", NumberStyles.AllowLeadingSign);`
 - `int.Parse("1,000", NumberStyles.AllowThousands, new CultureInfo("en-US")); // returns 1000`
- **TryParse()** – Ελέγχει αν μπορεί να γίνει parse το string (αν είναι int) και επιστρέφει true/false. Η τιμή αν μπορεί να γίνει parse πάει στο num (out parameter)
 - `bool isParsable = int.TryParse("1000", out int num);`
- **Convert.ToCTSDDataType()**
 - `Convert.ToInt32("1234");`
 - `Convert.ToInt16("1s");` `// FormatException`



Έξοδος δεδομένων

Προγραμματισμός με C#.NET Core

```
1 namespace WriteSyntax
2 {
3     /// <summary>
4     /// C# syntax for output
5     /// </summary>
6     internal class Program
7     {
8         static void Main(string[] args)
9         {
10             int num = 10;
11             Console.WriteLine("num = " + num);
12             Console.WriteLine("num = {0}", num);
13             Console.WriteLine($"num = {num}");
14         }
15     }
16 }
```

- using System;
 - Console.Write()
 - Console.WriteLine()
- Placeholders / Interpolation
 - {0}, {1}, {2}, ...
 - Όπως στην printf() της Java
- Interpolation με \$ και {}



Μορφοποίηση - Alignment

Προγραμματισμός με C#.NET Core

`{index[,alignment][:formatString]}`

```
Console.WriteLine("{0,6}", 1234);  
Console.WriteLine("{0,6}", 12);  
Console.Write("{0,-6}", 123);  
Console.WriteLine("--end");
```

```
    1234  
      12  
123    --end
```



Μορφοποίηση Αριθμών

Προγραμματισμός με C#.NET Core

```
Console.WriteLine("{0:C2}", 123.456);
```

```
//Output: 123,46 лв.
```

```
Console.WriteLine("{0:D6}", -1234);
```

```
//Output: -001234
```

```
Console.WriteLine("{0:E2}", 123);
```

```
//Output: 1,23E+002
```

```
Console.WriteLine("{0:F2}", -123.456);
```

```
//Output: -123,46
```

```
Console.WriteLine("{0:N2}", 1234567.8);
```

```
//Output: 1 234 567,80
```

```
Console.WriteLine("{0:P}", 0.456);
```

```
//Output: 45,60 %
```

```
Console.WriteLine("{0:X}", 254);
```

```
//Output: FE
```

Τα σύμβολα εξαρτώνται από το locale (culture), δηλαδή τα settings του υπολογιστή, δηλ. όσο αφορά το σύμβολο χιλιάδων, δεκαδικών νόμισμα, κλπ



Localization

Προγραμματισμός με C#.NET Core

```
1 using System;
2 using System.Threading;
3 using System.Globalization;
4 namespace LocalSettingsDemo
5 {
6     0 references
7     class CultureInfoExample
8     {
9         0 references
10        static void Main()
11        {
12            DateTime d = new DateTime(2012, 02, 27, 17, 30, 22);
13            Thread.CurrentThread.CurrentCulture = CultureInfo.GetCultureInfo("en-US");
14            Console.WriteLine("{0:N}", 1234.56);
15            Console.WriteLine("{0:D}", d);
16            Thread.CurrentThread.CurrentCulture = CultureInfo.GetCultureInfo("el-GR");
17            Console.WriteLine("{0:N}", 1234.56);
18            Console.WriteLine("{0:D}", d);
19        }
20    }
21 }
```

C:\WIN

```
1,234.56
Monday, February 27, 2012
1.234,56
Δευτέρα, 27 Φεβρουαρίου 2012
Press any key to continue . . .
```




Είσοδος Δεδομένων

Προγραμματισμός με C#.NET Core

- Τι διαβάζουμε από την κονσόλα
 - **Μόνο Κείμενο (String)**
- Τους άλλους τύπους (int, float, κλπ) τους κάνουμε convert αφού κάνουμε parse το string
- `Console.ReadLine()` Διαβάζει μια γραμμή που τελειώνει με enter
- `Console.read()` Διαβάζει ένα χαρακτήρα και επιστρέφει το Unicode



Έλεγχος μετατροπής string σε int

Προγραμματισμός με C#.NET Core

- Όταν μετατρέπουμε ένα string σε ακέραιο με την `int.Parse(string)` ή `Int32.Parse(string)` ή `Convert.ToInt32(string)`
- Αν το string δεν είναι αριθμός δημιουργείται μία εξαίρεση. Για να ελέγχουμε το παραπάνω μπορούμε να χρησιμοποιούμε try-catch ή την `TryParse`

```
string str = Console.ReadLine();
int intValue;
bool parseSuccess = Int32.TryParse(str, out intValue);
Console.WriteLine(parseSuccess ?
    "The square of the number is " + intValue * intValue + "."
    : "Invalid number!");
```



Output (1)

```
1  using System;
2
3  namespace WriteDemo
4  {
5      /// <summary>
6      /// Prints the sum of two integers using
7      /// 1. Concat: + + + ...
8      /// 2. Placeholders: {0} {1} {2} ...
9      /// Interpolation: {num1} {num2} {result} ...
10     /// </summary>
11     class Program
12     {
13         static void Main(string[] args)
14         {
15             // Δήλωση και Αρχικοποίηση Μεταβλητών
16
17             int num1 = 10;
18             int num2 = 20;
19             int result = 0;
20
21             // Εντολές
22             result = num1 + num2;
23
24             // Εκτύπωση Αποτελεσμάτων
25             Console.WriteLine("To άθροισμα των " + num1 + " και " + num2 + " είναι " + result);
26             Console.WriteLine("To άθροισμα των {0} και {1} είναι {2}", num1, num2, result);
27             Console.WriteLine($"To άθροισμα των {num1} και {num2} είναι {result}");
28         }
29     }
30 }
31
```



Output (2)

- Όταν έχουμε να εκτυπώσουμε και αλφαριθμητικά και μεταβλητές μπορούμε να το κάνουμε με τρεις τρόπους
 - Concat με τον τελεστή +
 - Με placeholders μέσα σε {} ξεκινώντας από το 0
 - Με interpolation μεταβλητών μέσα σε {} με τον χαρακτήρα \$ στην αρχή της Write



Expressions

```
1  using System;
2
3  namespace ExpressionsDemo
4  {
5      /// <summary>
6      /// Shows examples of arithmetic operators:
7      /// +, -, *, /, %, ++, --, +=, -=, /=, *=,
8      /// </summary>
9      class Program
10     {
11         static void Main(string[] args)
12         {
13             int num1 = 1;
14             int num2 = 2;
15             int sum, sub, mul, div, mod, result1, result2;
16             int finalResult;
17
18             sum = num1 + num2;
19             sub = num1 - num2;
20             mul = num1 * num2;
21             div = num1 / num2;
22             mod = num1 % num2;
23
24             result1 = sum++;
25             result2 = ++sub;
26
27             //mul = mul + 2;
28             mul += 2;
29
30             finalResult = ((num1 + num2) * (num2 / (num1++))) % num2;
31
32             Console.WriteLine($"{finalResult}");
33             Console.WriteLine($"{sum}\t{sub}\t{mul}\t{div}\t{mod}");
34             Console.WriteLine($"{result1}\t{result2}");
35         }
36     }
37 }
38 }
```



Διάβασμα αριθμών και πράξεις

Προγραμματισμός με C#.NET Core

```
class ReadingNumbers
{
    static void Main()
    {
        Console.Write("a = ");
        int a = int.Parse(Console.ReadLine());

        Console.Write("b = ");
        int b = int.Parse(Console.ReadLine());

        Console.WriteLine("{0} + {1} = {2}", a, b, a + b);
        Console.WriteLine("{0} * {1} = {2}", a, b, a * b);

        Console.Write("f = ");
        double f = double.Parse(Console.ReadLine());
        Console.WriteLine("{0} * {1} / {2} = {3}",
            a, b, f, a * b / f);
    }
}
```



Typecast

```
1  using System;
2
3  namespace TypecastDemo
4  {
5      /// <summary>
6      /// Shows Typecast examples between int - long
7      /// </summary>
8      class Program
9      {
10         static void Main(string[] args)
11         {
12             int myInt = 10;
13             long myLong = 100_000_000_000_000L;
14             float myFloat = 10.65F; // Type Suffix, C# decimals are double
15             decimal myDecimal = 1200.67M;
16
17             //myLong = myInt;
18             // Console.WriteLine($"{myLong}");
19
20             myInt = (int) myLong;
21             Console.WriteLine($"{myInt}");
22             Console.WriteLine($"{myLong}");
23             Console.WriteLine($"{myFloat}");
24             Console.WriteLine($"{myDecimal}");
25         }
26     }
27 }
```



Double demo

Προγραμματισμός με C#.NET Core

```
1  using System;
2
3  namespace FloatDoubleDemo
4  {
5      class Program
6      {
7          static void Main(string[] args)
8          {
9              double d1 = 12.5;
10             double d2 = 0D;
11             double result = 0D;
12             double remaining = 0D;
13
14             result = d1 / d2;
15             remaining = d1 % d2;
16
17             Console.WriteLine($"Result: {result} -- Remaining: {remaining}");
18         }
19     }
20 }
21
```

- Η διαίρεση δίνει δεκαδικό και το mod όπως στους ακραίους



Ternary

```
1  using System;
2
3  namespace TernaryOperator
4  {
5      /// <summary>
6      /// Υπολογίζει το απόλυτο ενός αριθμού με τη
7      /// χρήση του τριαδικού τελεστή. Το απόλυτο
8      /// ενός θετικού αριθμού είναι ο ίδιος ο αριθμός,
9      /// ενώ ενός αρνητικού είναι ο αντίστοιχος θετικός
10     /// </summary>
11     class Program
12     {
13         static void Main(string[] args)
14         {
15             int inputNumber = 0;
16             int abs = 0;
17
18             Console.WriteLine("Δώστε ένα ακέραιο αριθμό");
19             inputNumber = int.Parse(Console.ReadLine());
20
21             abs = (inputNumber >= 0) ? inputNumber : -inputNumber;
22
23             Console.WriteLine($"Absolute of {inputNumber} = {abs}");
24         }
25     }
26 }
27
```



Ternary write

```
1  using System;
2
3  namespace TernaryWriteDemo
4  {
5      /// <summary>
6      /// Τριαδικός τελεστής μέσα σε printf
7      /// </summary>
8      class Program
9      {
10         static void Main(string[] args)
11         {
12             int inputNumber = 0;
13             int abs = 0;
14
15             Console.WriteLine("Δώστε ένα ακέραιο αριθμό");
16             inputNumber = int.Parse(Console.ReadLine());
17
18             Console.WriteLine("Absolute of {0} = {1}", inputNumber, (inputNumber >= 0) ? inputNumber : -inputNumber);
19         }
20     }
21 }
22
```



Read, convert & Ternary Op

Προγραμματισμός με C#.NET Core

```
4 {
5     /// <summary>
6     /// Shows how to read from std input (console)
7     /// with Console.ReadLine and int.Parse() to covert
8     /// to int.
9     ///
10    /// Console always returns strings!
11    /// </summary>
12    class Program
13    {
14        static void Main(string[] args)
15        {
16            int age = 0;
17            string s;
18
19            Console.WriteLine("Παρακαλώ δώστε την ηλικία σας");
20            age = int.Parse(Console.ReadLine());
21
22            Console.WriteLine("{0}", (age >= 12) ? "Non PG-12" : "PG-12");
23
24
25            //s = (age >= 12) ? "non PG-12" : "PG-12";
26            //Console.WriteLine(s);
27
28            /*
29            if (age >= 12)
30            {
31                Console.WriteLine("Non PG-12");
32            } else
33            {
34                Console.WriteLine("PG-12");
35            }
36            */
37
38            // Console.WriteLine($"Η ηλικία σας είναι: {age}");
39        }
40    }
41 }
```

- Ternary expands to if/else



Float double read

Προγραμματισμός με C#.NET Core

```
1  using System;
2
3  namespace FloatDoubleRead
4  {
5      class Program
6      {
7          static void Main(string[] args)
8          {
9              float f = 0F;
10             double d = 0D;
11
12             Console.WriteLine("Δώστε δύο δεκαδικούς");
13             f = float.Parse(Console.ReadLine());
14             d = double.Parse(Console.ReadLine());
15
16             Console.WriteLine($"f = {f:N2} \t d = {d:N2}");
17         }
18     }
19 }
20
```



TryParse

Προγραμματισμός με C#.NET Core

```
internal class Program
{
    static void Main(string[] args)
    {
        decimal d1 = 0m;
        decimal d2 = 0m;

        Console.WriteLine("Insert three nums");
        if (!decimal.TryParse(Console.ReadLine(), out d1))
        {
            Console.WriteLine("Error in input");
        }

        if (!decimal.TryParse(Console.ReadLine(), out d2))
        {
            Console.WriteLine("Error in input");
        }

        if (!decimal.TryParse(Console.ReadLine(), out decimal d3))
        {
            Console.WriteLine("Error");
        }

        Console.WriteLine($"d1: {d1}, d2: {d2}, d3: {d3}");
    }
}
```

- Η TryParse() μπορεί να χρησιμοποιηθεί για state testing αντί της try/catch
- Λαμβάνει δύο παραμέτρους, i) το input και ii) την μεταβλητή που θα αποθηκευτεί αν είναι έγκυρος ο αριθμός.
- Το out σημαίνει πως το d1 περνάει by reference



Παραδείγματα

Προγραμματισμός με C#.NET Core

1. Διαβάζει από τον χρήστη ένα ποσό σε Ευρώ και το μετατρέπει σε δολάρια USA. Η ισοτιμία είναι $1 \text{ Ευρώ} = 116 \text{ USA cents}$
2. Διαβάζει από τον χρήστη δευτερόλεπτα και τα μετατρέπει σε μέρες, ώρες, λεπτά, δευτερόλεπτα



Ενδεικτικές Λύσεις

Προγραμματισμός με C#.NET Core

- Προσπαθήστε πρώτα να αναπτύξετε μόνοι σας τα παραδείγματα
- Ακολουθούν ενδεικτικές λύσεις



Euros to dollars (1)

Προγραμματισμός με C#.NET Core

```
1  using System.Text;
2
3  namespace EurosToDollars
4  {
5      /// <summary>
6      /// Reads from standard input an integer that
7      /// represents an amount in euro and converts
8      /// to dollars and cents. Assume that the parity
9      /// rate is 1 Euro = 1.07 USD.
10     /// </summary>
11     internal class Program
12     {
13         static void Main()
14         {
15             const decimal RATE = 1.07m;
16             const int CENTS_PER_DOLLAR = 100;
17             decimal dollars;
18             decimal cents;
19
20             Console.WriteLine("Please insert the amount in Euros");
21             if (!decimal.TryParse(Console.ReadLine(), out decimal inputEuros))
22             {
23                 Console.WriteLine("Error in input");
24             }
25         }
26     }
27 }
```

- Στην TryParse μπορούμε άμεσα να δηλώσουμε την μεταβλητή εξόδου



Euros to dollars (2)

Προγραμματισμός με C#.NET Core

```
26 dollars = inputEuros * RATE;  
27 cents = dollars * CENTS_PER_DOLLAR % 100;  
28  
29 Console.OutputEncoding = Encoding.UTF8;  
30 Console.WriteLine($"{\u20AC {inputEuros:N2} equals to \u0024 {dollars:N2} {cents, 5:N2} \u0024 cents");  
31 }  
32 }  
33 }
```

- Με `Console.OutputEncoding` θέτουμε σε UTF-8 την έξοδο



Secs to pretty format (1)

Προγραμματισμός με C#.NET Core

```
1 namespace SecondsToDate
2 {
3     /// <summary>
4     /// Converts seconds to pretty date.
5     /// </summary>
6     internal class Program
7     {
8         static void Main(string[] args)
9         {
10             const double SEC_PER_MINUTE = 60D;
11             const double SEC_PER_HOUR = 60 * SEC_PER_MINUTE;
12             const double SEC_PER_DAY = 24 * SEC_PER_HOUR;
13             double days;
14             double hours;
15             double minutes;
16             double seconds;
17
18             Console.WriteLine("Please insert seconds");
19             if(!double.TryParse(Console.ReadLine(), out double inputSeconds))
20             {
21                 Console.WriteLine("Error in input");
22             }
```



Secs to pretty format (2)

Προγραμματισμός με C#.NET Core

```
23
24     days = inputSeconds / SEC_PER_DAY;
25     seconds = inputSeconds % SEC_PER_DAY;
26
27     hours = seconds % SEC_PER_HOUR;
28     seconds %= SEC_PER_HOUR;
29
30     minutes = seconds % SEC_PER_MINUTE;
31     seconds %= SEC_PER_MINUTE;
32
33     Console.WriteLine($"{inputSeconds} = {days:N0} days, {hours:N0} hours, " +
34         $"{minutes:N0} minutes, {seconds:N0} seconds");
35 }
36 }
37 }
```



Μικρές Εργασίες (1)

Προγραμματισμός με C#.NET Core

- Πρόβλημα: Γράψτε μια εφαρμογή κονσόλας που λειτουργεί ως απλός υπολογιστής. Η εφαρμογή πρέπει να:
 - Ζητά από τον χρήστη να εισάγει δύο αριθμούς (δεκαδικούς αριθμούς).
 - Εκτελεί βασικές πράξεις: πρόσθεση, αφαίρεση, πολλαπλασιασμό και διαίρεση.
 - Εμφανίζει τα αποτελέσματα για κάθε πράξη.
- Βήματα:
 - Χρησιμοποιήστε τον τύπο `double` για τους αριθμούς.



Μικρές Εργασίες (2)

Προγραμματισμός με C#.NET Core

- Γράψτε μια εφαρμογή κονσόλας που μετατρέπει θερμοκρασία από Κελσίου σε Φαρενάιτ ή αντίστροφα, ανάλογα με την είσοδο του χρήστη.
 - Ζητήστε από τον χρήστη να επιλέξει τύπο μετατροπής (C για Κελσίου σε Φαρενάιτ, F για Φαρενάιτ σε Κελσίου).
 - Ζητήστε από τον χρήστη να εισάγει την τιμή της θερμοκρασίας.
 - Εκτελέστε τη μετατροπή ανάλογα με την επιλογή:
 - Κελσίου σε Φαρενάιτ: $F = C \times 9/5 + 32$
 - Φαρενάιτ σε Κελσίου: $C = (F - 32) \times 5/9$
- Εμφανίστε τη μετατραπείσα θερμοκρασία.