CUPRINS

Introducere 7

De ce această carte? 8
Eficienţă maximă 8
Scurt Istoric 9
De ce C#? 9

Capitolul I : Să ne pregătim 11

.NET 12
Spaţii de nume, clase, metode 12
Visual Studio 15
New project 16

Capitolul II: Prima aplicaţie 25

ToolBox, proprietăţi şi evenimente 26
MessageBox 30
Prima aplicaţie 36

Capitolul III: Toolbox 39

Box 40
CheckBox 40
CheckedListBox 43
ComboBox 44
PictureBox 47
TextBox 50
Label, ProgressBar şi Radio Button 54
Label 54
ProgressBar 57
RadioButton 60

Containers 64
GroupBox *64*
Panel *65*
SplitContainer *65*
TabControl *66*
Menus & Toolbars 67
ContextMenuStrip *67*
MenuStrip *71*
StatusStrip *74*
Timer 75
Dialog 79
OpenFileDialog *79*
SaveFileDialog *83*

Capitolul IV : Elemente de proiect 87

Proprietăţiile şi evenimentele unei forme 88
„Add new item..." 91
AboutBox *93*
Bitmap File *96*
Class *96*
Code File *97*
Cursor file *97*
Custom Control *99*
Icon File *99*
Resource File *101*
Setting File *103*
Text File *103*
User Control *105*
Windows Form *106*
Comunicarea între forme 110

Capitolul V – Linii, forme geometrice şi alte elemente de grafică 117

Clasa *Graphics* 118

Clasele Pen şi Brush 118
Linii, forme geometrice şi alte elemente de grafică 119
Linie *121*
Şir de linii *123*
Poligon *125*
Dreptunghi *127*
Elipsă *128*
Sector de elipsă *130*
Arc *132*
Sfaturi şi trucuri 135
Anexa 1 139
Net Speed 139
Anexa 2 149
Cpu Meter 149
BIBLIOGRAFIE 153

Introducere

Câţi dintre voi nu v-a-ţi dori să puteţi trece dincolo de bariera coborâtă de tainele programării, să puteţi intra sub interfaţa programelor pe care le utilizaţi zi de zi, să le intelegeţi, să le manipulaţi şi să creaţi chiar voi propriile softuri, în funcţie de dorinţele şi necesităţiile dumneavoastră? Într-o singura idee, să puteţi folosi calculatorul personal în adevăratul sens al cuvîntului. Veţi rămane uimiţi cât de mic este pragul dintre calculator - un simplu obiect pe care îl folosiţi pentru a afla informaţii sau pentru a vă ocupa timpul liber şi calculator - o fereastră de oportunităţi în care orice gând, idee, dorinţă poate fi materializată printr-un simplu cod de program.

Câţi dintre voi nu v-aţi dorit să porniţi pe acest drum, să învăţaţi şi să creaţi, ca într-un final să deveniţi buni programatori, dar v-a-ţi lovit de obstacole puse chiar de cărţile de specialitate, care în ciuda faptului că au scop didactic, conţin explicaţii şi formulări *‚pompoase'* şi fără prea mult sens pentru voi, create de programatori de renume mondial. Se pare ca aceştia au neglijat faptul că textele lor se adresează persoanelor aflate în stadiul de iniţiere, fără prea multe cunoştinţe în domeniu, pentru care o explicaţie simplă şi modestă este mult mai semnificativă şi mai atrăgătoare decât una complicată, de zeci de rânduri, în care se folosesc numai termeni ştiinţifici, majoritatea ne mai auziţi şi ne mai folosiţi până în acel moment.

Această carte îşi propune să renunţe la ideile greoaie şi de neînţeles şi să aducă o abordare nouă, deschisă, care să apropie cititorul de lumea programării şi să îi stimuleze dorinţa de a asimila informaţii noi, de a învăţa şi, în sfârşit, de a programa cu C#.

De ce această carte?

Ca mulţi tineri din ziua de azi, m-am hotarât şi eu într-o zi frumoasa de primăvară, să învăţ să creez programe ca cele pe care le folosesc zi de zi. După o lunga analiză şi câteva zile de căutări şi documentări, am decis că limbajul C# este cel mai accesibil, cel mai bun şi cel mai uşor de înţeles, într-un cuvânt cel mai potrivit limbaj de programare pentru acest scop. Astfel, am reuşit să strâng un număr mare de cărţi de specialitate, toate promiţându-mi acelaşi lucru: să ma înveţe să programez cu C#.

Încă de la primele capitole (capitole ce ar trebui să fie atrăgătoare şi uşor de înţeles) am avut mereu impresia că deschid un dicţionar cu o limbă străină necunoscută, în care aproape toate traducerile şi explicaţiile lipseau. În acele momente, de la sentimentul de speranţă că următoarea carte mă va stimula şi îmi va reaprinde dorinţa de a cunoaşte, am ajuns la rutină, la dezamăgire şi într-un final am ajuns chiar în punctul de a dori să renunţ. Cu greutate, am trecut peste aceea dezamăgire şi încet, am reuşit să pun cap la cap informaţiile adunate din cărţi şi astfel să fac în sfârşit primii paşi cu C#. Pentru mine acesta a însemnat trecerea la nivelul al doilea în drumul devenirii unui programator.

Însă, această trecere nu a fost de ajuns. Văzând că majoritatea autorilor cărţilor din acest domeniu sunt oameni de ştiinţă şi distinşi profesori universitari, că limbajul dumnealor nu este cel mai apropiat de limbajul tinerilor, studenţilor şi chiar cel al elevilor dornici de cunoaştere, am hotărât, student fiind, să scriu această carte în spiritul în care mi-aş fi dorit eu să găsesc un manual nu cu mult timp în urmă.

Eficienţă maximă

Dacă vă aşteptaţi să învăţaţi C# doar citind această carte, veţi fi dezamăgiţi de rezultat. După cum ştiţi, în acest domeniu nimic nu se poate face fără efort, fără a porni un mediu de programare şi a tasta chiar dumneavoastră codurile de program pentru a le testa ulterior.

Fiecare capitol este uşor de citit şi de înţeles însă este recomandat ca cel putin jumătate din timpul acordat de dumneavoastră studiului cărţii să fie destinat

practicii. Chiar dacă vi se pare prea mult timp vă garantez că rezultatele vor fi peste aşteptări.

Din aceste motive cartea conţine foarte multe exemple (coduri sursă), pe care vă recomand să le scrieţi şi să le rescrieţi (copy/paste nu vă va ajuta la nimic), să le testaţi şi să le modificaţi după bunul plac.

Scurt Istoric

Ar fi neobişnuit ca o carte dedicată unui limbaj de programare să nu prezinte măcar un scurt istoric despre acesta. Aşa că, cu riscul de a plictisi, voi spune câteva idei despre apariţia şi dezvoltarea „tânărului" limbaj C#.

Apărut ca variantă beta în iunie 2000 şi apoi ca variantă oficială în primăvara anului 2002, C# (se pronunţă „see sharp") este un limbaj relativ „tânăr".

C# a fost creat de o echipă de programatori şi oameni de ştiinţă de la Microsoft, echipă din care face parte şi inginerul Anders Hejlsberg, cel care a participat la crearea altor produse şi limbaje celebre cum ar fi: Borland Turbo C++ şi Borland Delphi. Scopul acestei echipe a fost crearea unui limbaj de programare care să „colecteze" tot ce este mai bun de la limbajele deja existente şi abia apoi să adauge concepte noi pentru a crea ceva extraordinar.

După cum se vede în prezent, scopul celor de la Microsoft a fost atins, C# devenind unul dintre cele mai puternice, cele mai flexibile şi cele mai utilizate limbaje de programare.

De ce C#?

- **Flexibil**

În C# aproape că nu există limite pentru ceea ce puteţi face. Cu puţină muncă şi imaginaţie, puteţi crea de la cele mai simple programe de Windows până la cele mai complexe aplicaţii Web.

- **Uşor de folosit**

Dacă aveţi un minim de cunoştinţe despre limbajele C, C++ şi despre programarea orientată pe obiecte, veţi observa cât de uşoare şi cât de logice sunt funcţiile existente în C#.

De exemplu pentru a afişa mesajul „Salut” într-o fereastră de mesaj, tot ce trebuie să faceţi este să scrieţi funcţia următoare:

```
MessageBox.Show("Salut");
```

- **Puternic**

După cum am spus în paginile anterioare, C# este un limbaj de programare ce a preluat multe din trasaturile bune ale limbajelor existente în perioada când acesta a fost creat.

Capitolul I : Să ne pregătim

În cele ce urmează am presupus că dumneavoastră aveţi cunoştinţe minime de bază legate de C sau C++ şi de programarea orientată pe obiecte. În caz contrar însuşiţi-vă mai întâi aceste cunoştinte şi doar după aceea îndreptaţi-vă atenţia către C# deoarece majoritatea cărţilor dedicate acestui limbaj pleacă de la nivelul de bază.

Deşi poate fi plictisitor pentru unii dintre dumneavoastră, acest capitol conţine informaţii necesare (să le zicem pregătitoare) pentru a putea apoi să crem în sfârşit primul nostru program.

Am încercat să păstrez acest capitol cât mai scurt şi mai simplu, dar să conţină totuşi, toate informaţiile necesare pentru a pune bazele cunoştinţelor dumneavoastră de programare cu C#.

Obiective:

- Înţelegerea conceptului de .Net;
- Înţelegerea conceptelor: Namespace(spaţii de nume), clase, metode;
- Crearea unui nou proiect;
- Înţelegerea componentelor unui proiect.

Subcapitole:

1. .Net
2. Spaţii de nume, clase, metode
3. Visual Studio
4. New Project

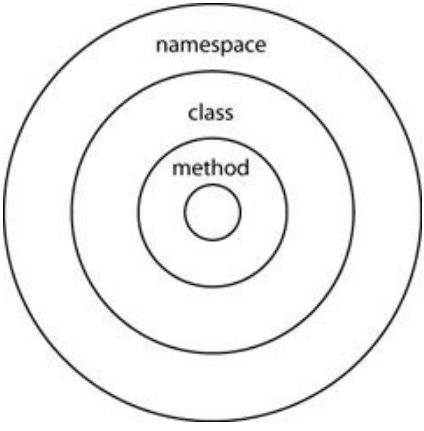
.NET

.NET este o platforma software, realizată la Microsoft, destinată dezvoltării de aplicaţii. Cele mai importante caracteristici ale acestei platforme sunt următoarele:

- Programatorii au la dispoziţie o serie de limbaje de programare de nivel înalt din care pot alege. Printre limbajele puse la dispoziţie de Microsoft se numără şi C#, fiind dezvoltat o dată cu platforma .NET.
- Pot fi dezvoltate o varietate largă de aplicaţii, de la programe pentru desktop până la aplicaţii pentru dispozitive mobile, aplicaţii şi servicii web sau servicii Windows, de la programe izolate şi până la sisteme distribuite de dimensiuni mari.
- Mediul de execuţie este strict controlat de un motor de execuţie, care oferă o serie de facilităţi ce ridică mult nivelul de calitate a aplicaţiilor (managementul automat al memoriei şi securitatea fiind primele două care ies în evidenţă).

Spaţii de nume, clase, metode

Când doriţi să creaţi un program va trebui mai întâi să creaţi un proiect. Fiecare proiect pe care îl veţi crea va fi împărţit pe mai multe nivele:



I. Spaţii de nume(Namespace)

Imaginaţi-vă aceste nivele ca fiind datele (adresa) de pe un colet. Când doriţi să trimiteţi coletul veţi scrie pe spatele lui destinaţia: oraşul, strada şi numărul (presupunând că veţi trimite coletul în ţară). În funcţie de aceste date, coletul dumneavoastră va fi trimis la adresa specificată.

Asemănător se întâmplă şi în proiectul dumneavoastră. Puteţi să priviţi un spaţiu de nume (in continuare vom folosi doar denumirea de namespace pentru spaţiu de nume) ca pe oraşul scris în adresa de destinaţie. Acesta reprezintă primul nivel în care coletul dumneavoastră va fi pus în maşina ce va transporta marfa în oraşul indicat.

Un namespace conţine una sau mai multe clase care conţin la la rîndul lor una sau mai multe metode.

Fie următoarele 2 namespace-uri:

```
namespace Name1
{
...
    public class Class1
    {
       ...
        public void ShowThat()
        {
            ....
        }
       ...
    }
...
}


şi


namespace Name2
{
...
    public class Class2
    {
       ...
        public void ShowThat()
        {
```

```
            ....
        }
        ...
    }
...
}
```

Pentru a folosi metoda *ShowThat ()* din clasa *Class1* din namespace-ul *Name1* va trebui sa specificaţi mai întâi că doriţi să folosiţi acel namespace (la începutul codului):

```
using Name1;
```

şi apoi să apelaţi metoda (în interiorul codului):

```
Name1.class1.ShowThat();
```

Vă întrebaţi probabil de ce am scris două namespace-uri şi nu unul. Să presupunem că în programul dumneavoastră aveţi nevoie la un moment dat de metoda *ShowThat()* din *Name1* şi apoi, după câteva lini de cod (nu contează câte sau unde doriţi să apelaţi aceste metode atât timp cât sunt în acelaşi proiect), aveţi nevoie şi de metoda *ShowThat()* din *Name2*. În acest caz veţi scrie în codul dumneavoastră următoarele:

```
{
        ...
        Name1.class1.ShowThat();
        ...
        Name2.class2.ShowThat();
        ...
}
```

Aceasta este principalul motiv pentru care se folosesc aceste nivele: previne confuzia între variabilele, funcţiile şi clasele cu aceleaşi nume şi este cu atât mai folositoare cu cât codul dumneavoastră este mai mare.

II. Clase

Mai ţineţi minte asemănarea făcută mai înainte dintre nivelele dintr-un proiect şi adresa de pe un colet? Daca da, va puteţi imagina acum o clasă ca fiind strada din adresa dată. În acest moment coletul se află deja la oficiul poştal

din oraşul destinaţie şi este preluat de poştaşul care corespunde străzii dumneavoastră.

Un namespace conţine, de regulă, mai multe clase. Aceste clase reprezintă în general obiecte şi conţin metode şi caracteristici specifice. Presupunând că dumneavoastră aveţi cunoştinţe legate de programarea orientată pe obiecte, acest aspect ar trebui să fie bine înţeles.

III. Metode

Continuând povestea noastră cu adresele şi coletul, am ajuns la ultimul nivel şi anume nivelul numărului casei de pe strada dată în adresă. În acest moment coletul este în geanta poştaşului aflat pe strada respectivă. Destinatarul primeşte coletul şi astfel am trecut prin toate cele trei nivele şi am reuşit să trimitem pachetul (ce reprezintă codul de instrucţiune) unde am dorit. Atenţie însă, deoarece am dorit să fac o legătură între aceste nivele şi un aspect legat de viaţa de zi cu zi (sistemul poştal), comparaţia este umpic forţată: O metodă nu este un obiect, este modul în care se interacţionează cu clasa şi cu instanţe ale ei.

O metodă conţine codul sursă format din instrucţiuni care spun calculatorului dumneavoastră ce sa facă.

Visual Studio

Toate explicaţiile din această carte au fost scrise pentru mediile de programare Microsoft Visual Studio.

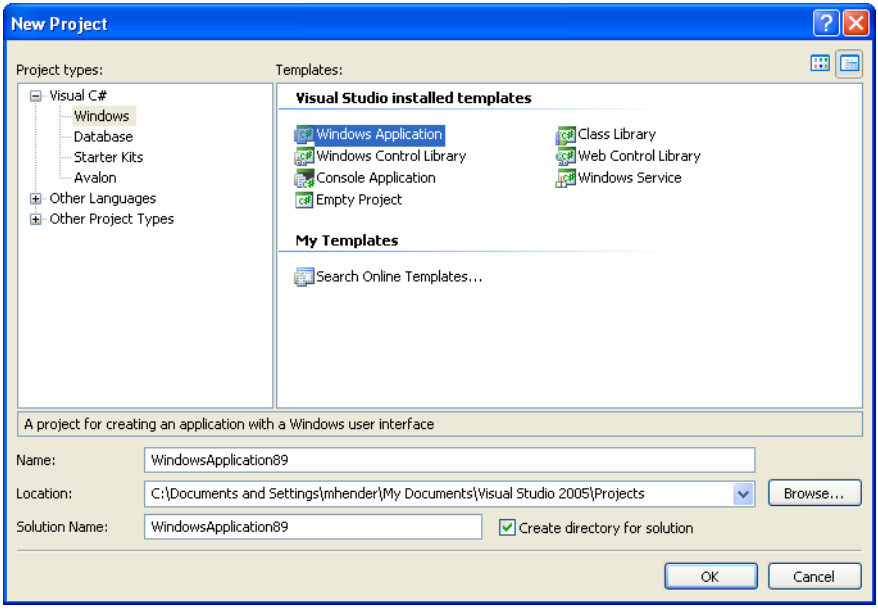
Toate exemplele din această carte sunt corecte şi au fost verificate cu Microsoft Visual Studio 2008 Express Edition, care poate fi descărcat gratuit de pe internet (adresa: http://www.microsoft.com/express/download/) . Variantele *professional* şi altle se pot achiziţiona contra cost.

New project

De fiecare dată când veţi începe scrierea unui nou program veţi creea un nou proiect. În Visual Studio aceasta se face astfel:

File->New->Project

După ce vă va apărea fereastra *New Project* introduceţi numele proiectului şi verificaţi ca tipul dumneavoastră de proiect (*Project type*:) să fie *Visual C# - > Windows* iar şablonul (*Templates*:) să fie selectat cel cu denumirea :*Windows Application* sau *Windows Form Application.*



Felicitări! Aţi creat primul dumneavoastră proiect. Presupunem că acesta se numeşte *WindowsApplication89.*