RODICA ELENA BADEA

PROGRAMARE IN C# PENTRU LICEU

"Lumea crede ca informatica este arta geniilor dar realitatea este chiar opusa, sunt multi oameni care construiesc dupa alti oameni, ca un zid de caramizi." - DONALD KNUTH

Rodica Elena BADEA

PROGRAMARE IN C# PENTRU LICEU

ISBN 978-606-94690-0-2

Editura EvoMind

2018

Cuprins

1. Platforma .NET

.NET este un cadru (Framework) de dezvoltare a produselor software ce permite realizarea, distribuirea şi rularea aplicaţiilor pentru sisteme de operare Microsoft şi aplicaţiilor WEB.

1.1 Tehnologia .NET

Tehnologia .NET înglobează mai multe tehnologii (ASP, XML, OOP, SOAP, WDSL, UDDI) şi limbaje de programare (VB, C++, C#, J#) asigurând atât portabilitatea codului compilat între diferite calculatoare cu sisteme Windows, cât şi reutilizarea codului în programe, indiferent de limbajul de programare utilizat.

1.2 Componenta .NET Framework

.NET Framework este o componentă livrată împreună cu sistemul de operare Windows. Reprezintă un mediu care permite dezvoltarea si rularea aplicațiilor si serviciilor Web, independent de platforma. Ea stă la baza tehnologiei .NET si reprezintă ultima interfața dintre aplicațiile .NET si sistemul de operare. In prezent este formata din 3 componente esențiale:

- Limbajele C#, VB.NET, C++ şi J#. Ele respectă nişte specificaţii OOP numite Common Type System (CTS)pentru a putea fi integrate pe platforma. Elementele lor de baza sunt : clase, interfeţe, delegări, tipuri valoare şi referinţă, iar ca mecanisme: moștenire, polimorfism și tratarea excepțiilor
- Common Language Runtime –platforma de executare a programelor utilizata de toate cele patru limbaje.
- Framework Class Library (FCL)-bibliotecile utilizate in realizarea aplicațiilor.

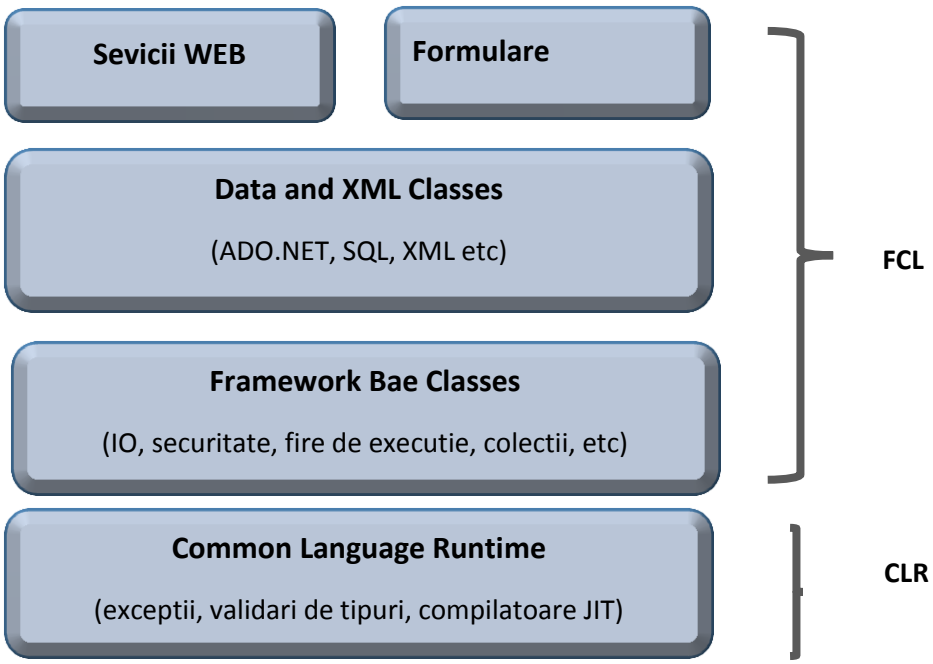
Dezvoltarea de aplicații pe platforma .NET se poate face prin intermediul următoarelor elemente :

- un set de limbaje (C#, Visual Basic .NET, J#, Managed C++, Smalltalk, Perl, Fortran, Cobol, Lisp, Pascal etc),
- un set de medii de dezvoltare (Visual Studio .NET, Visio),
- o bibliotecă de clase pentru crearea serviciilor Web, aplicațiilor Web şi aplicațiilor desktop Windows.

Când dezvoltăm aplicații .NET, putem utiliza:

- Servere specializate - un set de servere Enterprise .NET (din familia SQL Server 2000, Exchange 2000 etc), care pun la dispoziție funcții de stocare a bazelor de date, email, aplicații B2B (Bussiness to Bussiness – comerț electronic între partenerii unei afaceri).
- Servicii Web (în special comerciale), utile în aplicații care necesită identificarea utilizatorilor (de exemplu, .NET Passport - un mod de autentificare folosind un singur nume şi o parolă pentru toate site-urile vizitate)
- Servicii incluse pentru dispozitive non-PC (Pocket PC Phone Edition, Smartphone, Tablet PC, Smart Display, XBox, set-top boxes, etc.)

Arhitectura .NET Framework



Componenta .NET Framework este formată din compilatoare, biblioteci şi alte executabile utile în rularea aplicațiilor .NET.

Fișierele corespunzătoare se află, în general, în directorul C:\WINDOWS\Microsoft.NET\Framework\V2.0…. (corespunzător versiunii instalate).

Principalele caracteristici ale arhitecturi .NET

- Independenta dintre procesor si platforma
  In scrierea aplicațiilor programatorii nu trebuie sa fie preocupați de caracteristici hardware si software ale sistemului.
- Interoperabilitatea limbajelor
  Componentele aceleiași aplicații pot fi scrise in diverse limbaje suportate de platforma .NET.
- Managementul automat al memoriei
  Existenta mecanismului de *garbage collection* (alocarea si eliberarea memoriei in mod automat).
- Portabilitatea
  Un program scris pe platforma .NET poate rula pe orice sistem pe care aceasta platforma este instalata.
- Securitatea
  Este realizata pe platforma prin mecanismul de *tratare a exceptiilor.*

1.3 Compilarea programelor pe platforma .NET

Limbajele de programare pot fi limbaje interpretate si limbaje compilate. Programarea intr-un limbaj interpretat presupune scrierea codului si rularea. Pe parcursul rulării codul este schimbat in cod mașină si executat. Se pot înlătura erorile pe parcurs. Limbajul nu necesită timp de compilarea si legare. Marele dezavantaj: sunt limbaje lente (exemplu limbajul Visual Basic in momentul apariției).

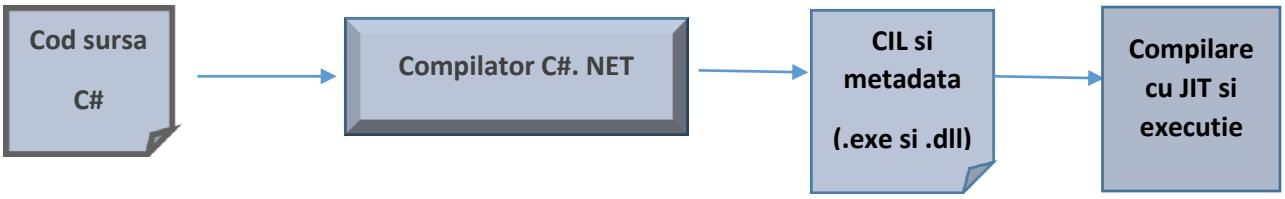
Codul scris intr-un limbaj compilat se numește cod sursa, este translatat de compilator in cod executabil. In cazul in care apar erori se reia compilarea. Exemplu limbajele C, C++.

Limbajul C# este un limbaj compilat. In comparative cu celelalte doua limbaje menționate mai sus in urma compilării codului se creează un fișier *assembly(* cu extensia .exe sau .dll). un astfel de fișier poate fi executat pe un sistem fără .NET. Fișierul conține un cod numit limbaj intermediar, CIL (Common Intermediate Language). CIL conține un set de instrucțiuni portabile, independente de platforma si procesor.

In momentul in care un program este executat, CRL activează compilatorul JIT(just in time). Acesta preia codul CIL si-l transforma in executabil.

Un program compilat in format CIL poate rula pe orice sistem pe care s-a instalat CRL.

Fișierul .exe(sau .dll) creat la compilare conține codul CIL si metadatele(datele utilizate de aplicație).



Procesul de compilare C# pe platforma .NET

De ce am alege .NET?

- ❖ Oferă instrumente pe care le putem folosi si in alte programe, permite accesul ușor la baze de date, realizarea elementelor grafice

- Dispune de instrumente(controale) pentru realizarea elementelor grafice in cadrul interfețelor. Controale ce sunt găsite in spațiul de nume System.Windows.Forms
- .NET vă oferă clase care efectuează majoritatea sarcinilor uzuale, reducând astfel timpul necesar dezvoltării aplicațiilor.

2. Introducere în limbajul C#

2.1 Caracteristici

Limbajul C# este unul simplu care folosește 80 de cuvinte cheie si 12 tipuri predefinite de date. Este foarte productive bazând-se pe concept modern de programare.

In echipa care a creat C# s-a aflat si autorul limbajului Turbo Pascal, Anders Hejlsbeng.Este un limbaj orientat pe obiect (OOP). Prin urmare se folosește de clase. Variabilele de tip clasa se numesc obiecte. Obiectele sunt uneori tangibile: ferestre, butoane sau netangibile de exemplu fire de execuție.

OOP are la baza următoarele concept fundamentale: încapsularea datelor, moștenirea si polimorfismul. Fiecare clasa are câmpuri, proprietății si metode.

C# permite programarea generica, diferita de OOP. Acest tip de programare are drept scop scrierea de cod independent de tipurile de date. Exemplu: O funcție primește drept parametru un sir de elemente de tip întreg si realizează sortarea acestora. Se poate realiza un algoritm ce sortează șirul fără sa tina cont de tipul elementelor, astfel se definește o funcție generica ce poate fi folosita pentru șiruri cu diverse tipuri de elemente.

C# folosește garbage collection pentru alocarea si eliberarea automata a zonei de memorie ocupata de obiecte.

2.2 Crearea aplicațiilor consola

Crearea aplicațiilor pe platforma .NET se poate face:

- Prin compilarea in linia de comanda
- Prin folosirea consolei

  Pentru prima varianta este necesara instalarea kit-ului de dezvoltare a aplicațiilor .NET Framework 4.5 Software Development Kit (SDK), oferit in

mod gratuit pentru Windows 8 (sau se alege varianta adecvata sistemului de operare folosit). Variantele de SDK sunt gratuite.

Codul poate fi scris in orice editor de texte si apoi compilat in linia de comanda.

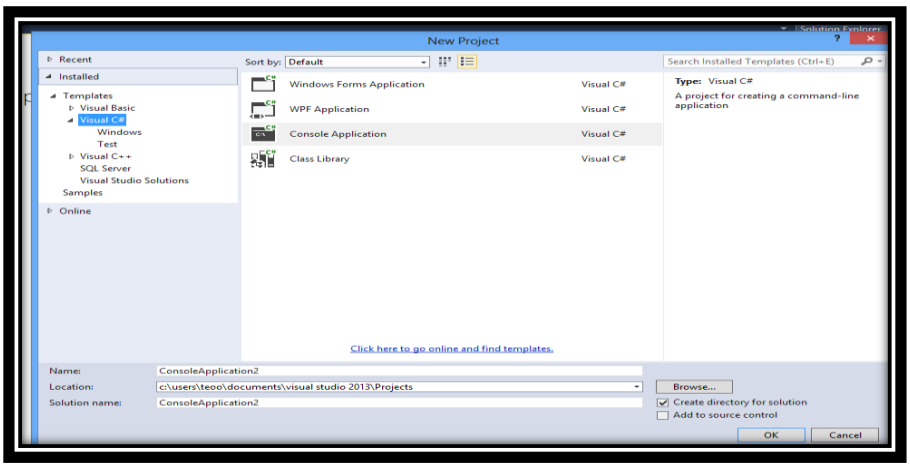
Pentru cea de-a doua modalitate, pe care o vom utiliza in continuare, trebuie sa instalam Visual Studio .NET sau Visual Studio Express 2013(sau alte versiuni).

Visual 2013 Express poate fi descărcat de pe site-ul Microsoft, varianta trial. Necesita existenta unui cont pe acest site, astfel încât după expirarea perioadei de 30 de zile se poate reactiva cu cerința de folosire in scopuri didactice. Va conține Visual C#, Visual Basic, Visual C++, SDK 4.5.1 si serverul de BD Microsoft SQL Server 2012.

Indiferent de varianta aleasa trebuie sa optam pentru instalarea componentei .NET Framework, Visual C#, MS SQL Server si MSDN (documentație).

După instalarea Visual Studio Express 2013 pentru crearea unei aplicații de tip consola vom urma pașii:

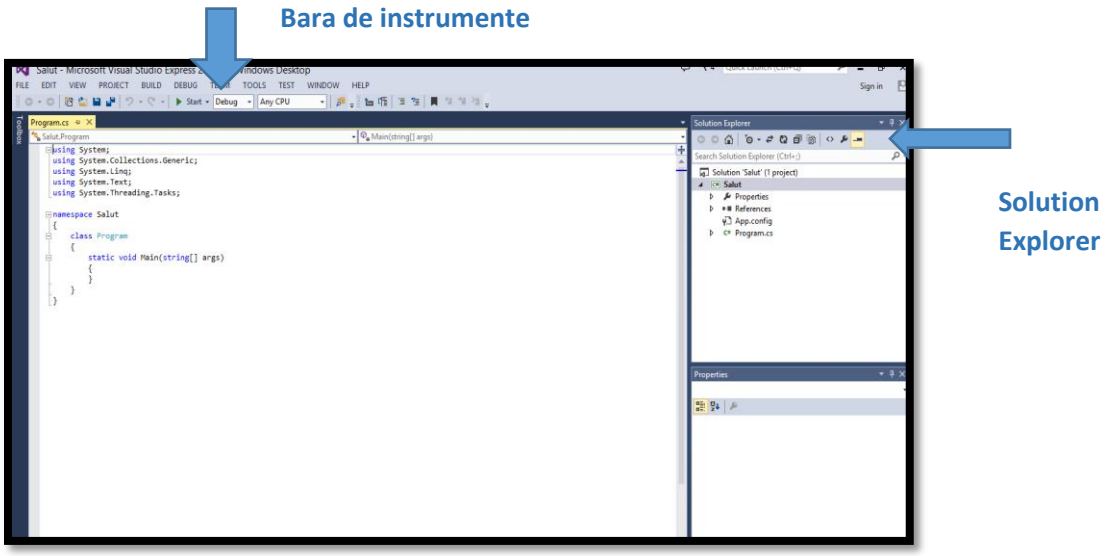
- Accesam din start VS Express 2013



- Selectam New Project->Visual C#-> Console Application, schimbam numele aplicației daca dorim.

Se va deschide o fereastra in care putem edita codul. Fereastra pune la dispoziție:

- Bara de instrumente (Toolbars) -conține icon-uri pentru operații de baza: crearea, salvarea, vizualizare, etc.
- Panoul Solution Explorer- afișează fișierele conținute, fișierul cu extensia *.cs este cel care conține sursa.



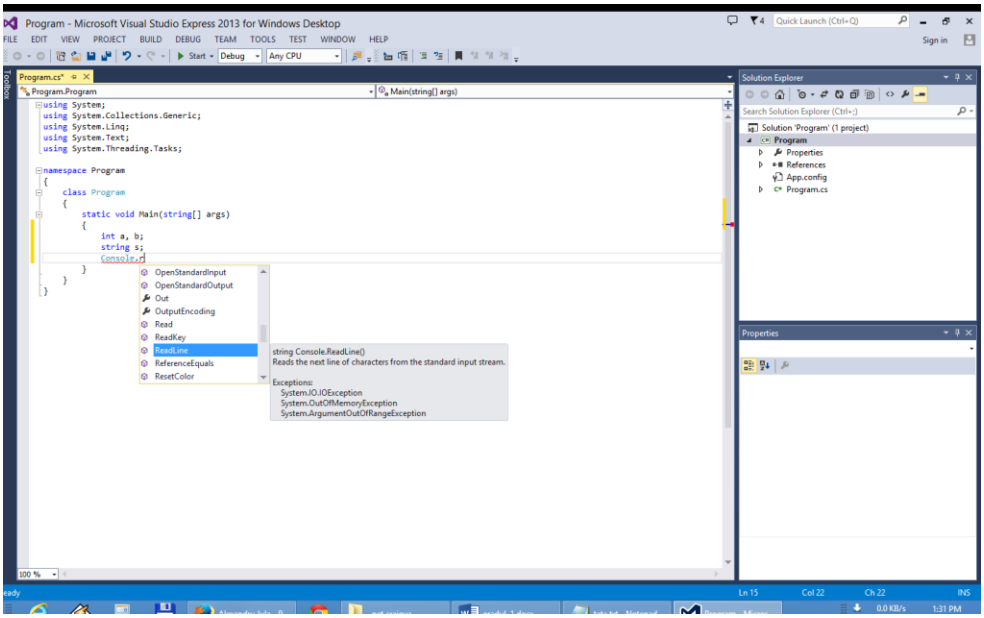
In fereastra deschisa adăugam o declarare de variabile locale si o afișare a textului “invat c#”.

Pentru compilare si rulare se folosește F5. Pentru compilare F6 se creează fișierul assembly.

Rularea programului se poate realiza in mai multe moduri:

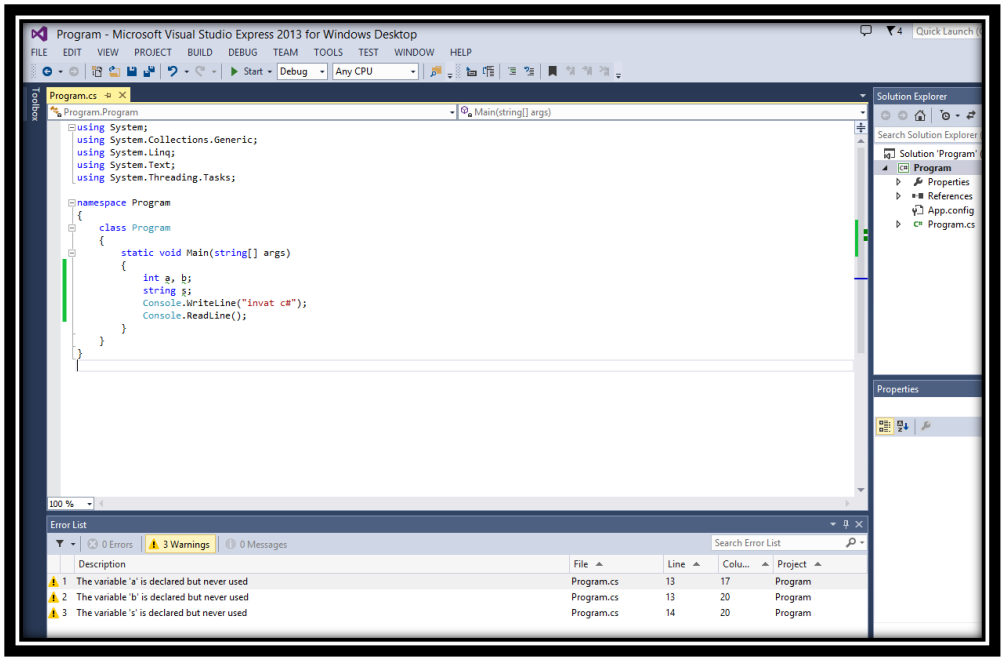
- CTRL+F5 fără asistenta de depanare
- F5, cu asistenta de depanare
- Pas cu pas (Step Into -F11, Step Over-F10)
- Rapida pana la un punct de întrerupere(F9-Toggle Breackpoint, F6- Start Debugging, Shift+F5-Stop Debugging)

O facilitate a editorului de cod este IntellSense, oferă o evidențiere a metodelor existente într-o lista din care se poate selecta cea dorita.



2.3 Structura unui program in C#

In urma creări aplicației de tip consola in VS Express 2013



Se evidențiază următoarea structura.

Prima linie using System reprezintă o directive ce specifica faptul ca se vor folosi clase incluse in spațiul de nume System.

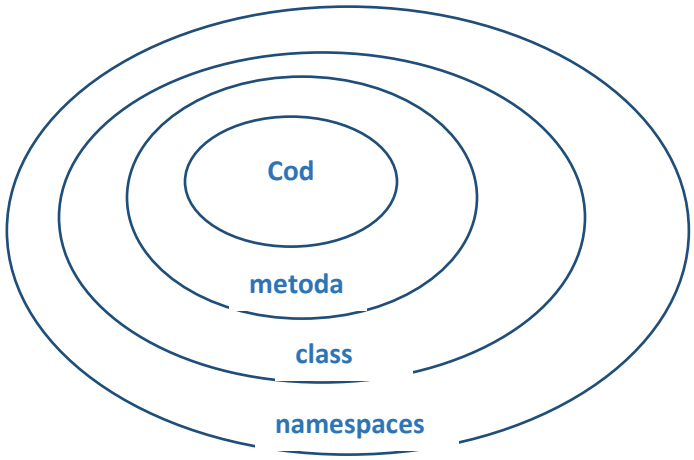
Un spațiu de nume reprezintă o colecție de tipuri sau o grupare de alte spatii de nume care pot fi utilizate in program.

Exemplu: pentru afișarea la consola folosim clasa Console din spațiul de nume System.

Orice cod este conținut de o clasa. In cazul nostru clasa se numește program. Aceasta clasa este un tip abstract. C# lucrează numai in clase. Nu exista variabile globale. Pentru specificarea clasei se folosește cuvântul cheie class.

Pentru afișare se apelează metoda WriteLine a clasei Console.

In C# un program poate fi privit având mai multe straturi. Codul se afla in interiorul metodelor. Metodele se afla in clase, clasele in namespaces-uri.



Punctul de intrare in aplicație este metoda Main, care poate prelua sau nu argumente in linia de comanda. In momentul execuției programului CLR(mediul de execuție) caută metoda Main. Un program poate sa conțină mai multe clase dar o singura metoda Main.

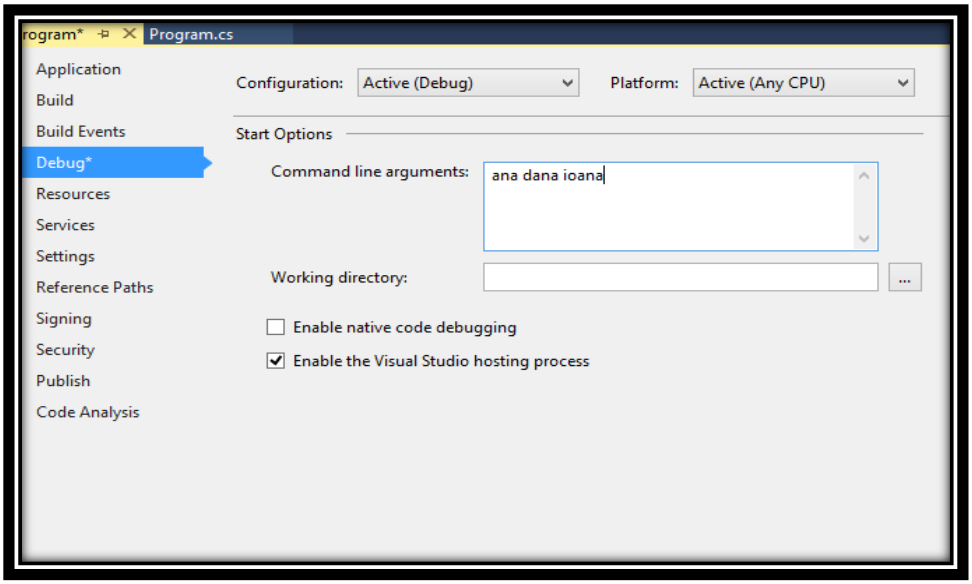
Următorul program afișează termeni șirului transmis in linia de comanda.

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

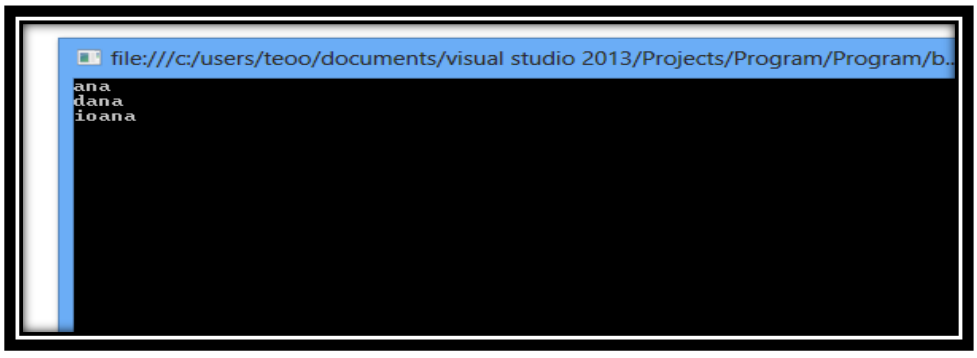
```
    {
        for (int i = 0; i < args.Length;i++ )
            Console.WriteLine(args[i]+" ");
        Console.ReadLine();
    }
  }
}
```

Pentru introducerea argumentelor vom folosi fereastra Solution Explorer-> Properties

> Debug ->Command line Argument si editam elementele separate prin spațiu.



La rulare rezulta:



In interiorul metodei se pot defini variabile locale, așa cum avem in exemplu anterior a si b de tipul int si s de tip string.

Formele funcției Main:

- static void Main(string[] args) {/*…*/} - metoda statica, care nu returnează nimic si care are drept parametru in linia de comanda un sir de string-uri.
- static int Main(string[] args) {/*…*/} - metoda statica, care returneaza un întreg (poate fi un cod de eroare) si care are drept parametru in linia de comanda un sir de string-uri.
- static void Main() {/*…*/} -- metoda statica, nu returnează nimic si nu are parametrii in linia de comanda
- static int Main() {/*….*/} - metoda statica, care returnează un întreg (poate fi un cod de eroare) , fără parametrii in linia de comanda.

  Cuvântul cheie static este folosit pentru a preciza o metoda static. O metoda de acest fel poate fi apelata in lipsa unui obiect de tipul clasei.

2.4 Sintaxa limbajului C#

Alfabetul limbajului C# este format din litere mici si mari ale alfabetului englez si simbolurile special. Vocabularul limbajului este format din: nume (identificatori), cuvinte cheie, separatori, delimitatori, expresii, comentarii.

2.4.1 Comentarii

Limbajul C# permite 3 tipuri de comentarii:

- pe o singura linie specificate prin //
- pe mai multe linii /*…*/
- document in format XML folosind ///.

2.4.2 Nume (identificatori)

Reprezintă numele folosite pentru variabile, clase, metode. Ele se construiesc după următoarele reguli:

- trebuie sa înceapă cu litera sau unul dintre cele doua simboluri “@”, “_”;
- al doilea caracter trebuie sa fie litera , cifra sau simbolul “_”;
- numele cuvintelor cheie nu pot fi folosite cu alte destinații, decât daca sunt precedate de “@”.

2.4.3 Cuvintele cheie

C# are următoarele cuvinte cheie:

abstract	as	base	break	bool
byte	case	catch	char	checked
class	const	continue	decimal	default
delegate	do	double	else	enum
event	explicit	extern	false	finally
fixed	float	for	goto	goto
if	implicit	in	int	interface
internal	is	lock	long	namespace
new	null	object	operator	out
override	params	private	protected	public
readonly	ref	return	sbyte	sealed
short	sizeof	stackalloc	static	string
struct	switch	this	throw	true
try	typeof	uint	ulong	unchecked
unsafe	ushort	using	virtual	void
volatile	while			

Cuvinte chei contextuale (dau semnificații specific codului)

ascending	by	descending	equals	from
get	group	into	join	let
on	orderby	partial	select	set
value	where	yield		

2.4.4 Constante

Prin folosirea la declararea lor se folosește cuvântul cheie *const.*

Exemplu: *const int x=90*; //inițializarea constantei este obligatorie.

Prin folosirea la declarare a cuvântului *readonly (*variabila este membra a claselor).

Exemplu: *readonly int x;*

readonly int x=34;

2.4.5 Expresii si operatori

Definiție. O expresie reprezintă o succesiune logica de operanzi si operatori. Operatorii reprezintă operațiile care se efectuează asupra operanzilor (valori).

Exista următoarele tipuri de operatori: unari, binari, ternari.

Ordinea de evaluare a expresiei este data de prioritatea operatorilor.

Prioritate	**Tip**	**Operatori**	**Asociativitate**
0	Primar	( ) [ ] f() . x++ x-- new typeof sizeof checked unchecked ->	→
1	Unar	+ - ! ~ ++x --x (tip) true false & sizeof	→
2	Multiplicativ	* / %	→
3	Aditiv	+ -	→
4	De deplasare	<< >>	→
5	Relaţional	< > <= >= is as	→
6	De egalitate	== !=	→
7	AND (SI) logic	&	→
8	XOR (SAU exclusive) logic	^	→
9	OR (SAU) logic	|	→
10	AND (SI) conditional	&&	→
11	OR (SAU)	||	→
12	Condiţional(ternar)	?:	←

13	atribuire simplă atribuire compusa	= *= /= %= += -= ^= &= <<= >>= |=	←

Tabel cu operatorii si prioritatea lor

Exemplu de folosire a operatorului ternar.

Sa se decida daca un număr citit este par sau impar.

Sintaxa: (condiție)?expresie_1: expresie_2.

Semnificație: Se evaluează condiția daca este adevărata se executa expresie_1 altfel expresie_2.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Program
{
   class Program
   {
      static void Main(string[] args)
      {
         int x;
         string rez;
         Console.WriteLine("Introduceti numarul=");
         x=int.Parse(Console.ReadLine());
         rez=(x%2==0)?"este un numar par":"este un numar impar";
         Console.WriteLine(rez);
         Console.ReadLine();
      }
   }
}
```

2.4.6 Conversia datelor

Ca si in limbajul C, exista doua tipuri de conversii:

- implicite- in mod automat de la un tip de date mai mic la unul mai mare.
- Explicit- folosid operatorul cast “()”. Conversie forțata.

  Exemplu:

Adunarea a doua numere de tip int si păstrarea rezultatului într-o variabila de tip long. Împărțirea a doua numere de tip întreg si obținerea unui rezultat real.

```
class Program
   {
      static void Main(string[] args)
      {
         int x,y;
         long r1;
         float r2;
         Console.WriteLine("Introduceti numarul x=");
         x=int.Parse(Console.ReadLine());
         Console.WriteLine("Introduceti numarul y=");
         y = int.Parse(Console.ReadLine());
         r1 = x + y;//conversie implicita;
         r2 = x / y;//nu s-a realizat decat conersia implicita a rezultatului
         Console.WriteLine(r2);
         r2 = (float)x / y;//conversie explicita a lui x la tipul float;
         Console.WriteLine(r2);
         Console.ReadLine();
      }
}
```

Conversia din tip numeric in sir de caractere si invers esste foarte folosita in limbajul C#.

Din număr in sir numeric se va face utilizând metoda *ToString* a clasei *Object*.

Din sir de caractere in număr, folosind metoda *Parse* a clasei *Object.*

sir → int int.Parse(sir) sau Int32.Parse(sir)

sir → long long.Parse(sir) sau Int64.Parse(sir)

sir →double double.Parse(sir) sau Double.Parse(sir)

sir →float float.Parse(sir) sau Float. Parse(sir)

Conversia boxing si unboxing se aplica in acest limbaj in care toate tipurile deriva din clasa Object, întrucât permite trecerea de la tipul valoare la tipul obiect si invers.

Exemplu de conversie boxing:

int a=14;

Object ob=(object)a;//boxing explicit

sau

object ob=a;//boxing implicit

Exemplu conversie unboxing

int i=20;

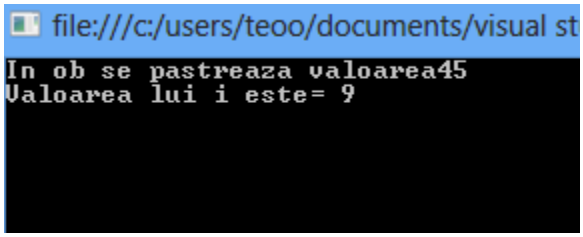
object ob=i; //boxing implicit

i=(int)ob;// unboxing explicit

Exemplu:

```
static void Main(string[] args)
      {
         int i = 45;
         object ob = i;//boxing implicit
         i = 9;
         Console.WriteLine("In ob se pastreaza valoarea"+ ob);
         Console.WriteLine("Valoarea lui i este= "+ i);
         Console.ReadLine();
      }
```

Dupa rulare se obține:



2.5 Tipuri de date

In limbajul C# se poate face o clasificarea a tipurilor de date in doua categorii:

- ❖ Tipuri valoare: simple predefinite, enumerare, structura;

- Tipuri referința: clase, delegate, interface, array;

Pentru declararea variabilelor de tip valoare se aloca spațiu de memorie in mod automat. Declararea variabilelor de tip referința nu implica in mod automat alocarea de spațiu.

Tipurile valoare diferă de tipurile referința prin faptul ca variabilele de tip valoare conțin datele obiectului, variabilele de tip referința conțin referințe la obiecte.

2.5.1 Tipuri predefinite

C# dispune de următoarele tipuri predefinite ilustrate in următorul tabel.

Tip	Descriere	Alias pentru tipul struct din spațiul de nume System
object	object	
string	Secvenţă de caractere Unicode	System.String
sbyte	tip întreg cu semn, pe 8 biţi	System.Sbyte
short	tip întreg cu semn, pe 16 biţi	System.Int16
int	tip întreg cu semn pe, 32 biţi	System.Int32
long	tip întreg cu semn, pe 64 de biţi	System.Int64
byte	tip întreg fărăsemn, pe 8 biţi	System.Byte
ushort	tip întreg fărăsemn, pe 16 biţi	System.Int16
uint	tip întreg fărăsemn, pe 32 biţi	System.Uint32
ulong	tip întreg fărăsemn, pe 64 biţi	System.Uint64
float	tip cu virgulămobilă, simplăprecizie, pe 32 biţi (8 pentru exponent, 24 pentru mantisă)	System.Single
double	tip cu virgulămobilă, dublăprecizie, pe 64 biţi (11 pentru exponent, 53 pentru mantisă)	System.Double

bool	tip boolean	System.Boolean
char	tip caracter din setul Unicode, pe 16 biţi	System.Char
decimal	decimal tip zecimal, pe 128 biţi (96 pentru mantisă), 28 de cifre semnificative	System.Decimal

2.5.2 Tipul enumerare

Se definește ca si in limbajul C de utilizator. Nu poate fi declarat abstract si nu poate fi derivat.

Cuvântul cheie folosit este *enum.* Tipul *enum* este derivat din clasa *System.Enum.*

Sintaxa: *[atribute] [modificatori] enum NumeEnumerare [:Tip]*

{

lista

}

Observații:

- Valoarea primului element din listă este 0, următorul element are valoarea mai mare cu o unitate decât precedentul.
- Valorile folosite pentru inițializări trebuie fie in domeniul de valori ale tipului *enum*

  Exemplu:

  enum O: byte

  {

  A=1

  }// eroare
- Nu se accepta referințe circulare

enum

{

a=b,

b}

- In lipsa specificației tipului *enum* acesta este int.
- Valoarea fiecărui câmp poate fi specificat explicit.

  Exemplu:

  enum Valori

  {

  a=1,

  b=3,

  c=a+b

  }

- Mai mulți membri pot avea aceeași valoare.

  Exemplu:

  enum E1

  {

  p=10,

  f=1,

  r=f

  }

- Este recomandat ca orice tip enum sa conțină un membru cu valoarea 0, pentru ca in anumite contexte valoarea implicita a unei variabile enum este 0.

- Tipul enumerare poate fi convertit către tipul de baza si înapoi, folosind o conversie explicita (cast).

Exemplu:

```
enum Valori
   {
   a=1,
   b=3,
   c=5
   }
Class Testare
 {
Public void Main()
    {
     Valori v=(Valori)5;
     int i=(int) v;
    }}
```

Exemplu de folosire a unei structure in cadrul unui program

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApplication3
{
   class Program
   {
      enum Luna
      {
         Ianuaria=1,
         Februarie,
         Martie,
         Aprile,
         Mai,
         Iunie,
         Iulie,
```

```
        August,
        Septembrie,
        Octombrie,
        Noiembrie,
        Decembrie
    }
    static void Main(string[] args)
    {
        Console.WriteLine("Luna iunie estr a " +(int) Luna.Iunie + " a anului");
        Console.ReadLine();
    }
}
        }
```

2.5.3 Tipul structura (struct)

Se declara prin utilizarea cuvântului struct. Sunt similare claselor dar le lipsește caracteristica de moștenire.

Exemplu:

```
using System;
        struct Elev
         {
         public string nume;
         public float nota;
          }
        Class Testare
           {
             static void Main()
               {
                Elev e=new Elev();
                e.nume=”Ionescu”;
                e.nota=”9.50”;
                Console.WriteLine(’’Elevul ’’+e.nume+’’ are nota ’’+e.nota);
```

```
            Console.ReadLine();
            }
    }
```

2.5.4 Tablouri unidimensionale

Tablourile sunt structuri de date dea același tip. Ele sunt date de tip referința. Se clasifica in doua tipuri: tablouri unidimensionale si tablouri multidimensionale.

Declararea tabloului unidimensional:

tip_elemente [] nume_tablou;

Exemplu:
int[] a;

Declararea nu asigura alocarea spațiului de memorie, pentru acest lucru se folosește operatorul new.

nume_tablou=new tip_elemente[n];

Indexarea elementelor începe de la 0.

Exemplu: programul construiește un tablou cu elemente de tip întreg pe care apoi îl afișează.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApplication3
{
    class Program
    {

        static void Main(string[] args)
        {
            int[] v;
            v = new int[10];
            for (int i = 0; i < v.Length; i++)
                v[i] = i * 3;
            for (int i = 0; i < v.Length; i++)
                Console.WriteLine(" "+v[i]);
            Console.ReadLine();
        }
    }
}
```

Tablourile sunt obiecte ale clasei Array, prin urmare ele pot folosi metode ale acestei clase (exemplu: metoda Lenght).

Tablourile unidimensionale pot fi inițializate cu valori de același tip cu elementele lor.

Exemplu:

int[] a=new int[] {7, 9, 5, 6}//tablou cu patru elemente

sau

int[] a={2, 3, 7, 8}

2.5.5 Tablourile multidimensionale rectangulare

Tablourile multidimensionale se împart in doua categorii: rectangulare si neregulare.

Tablourile rectangulare sunt acele care își păstrează numărul de elemente pentru o anumita dimensiune, ele au prin urmare o forma dreptunghiulara.

Declarare: tip_elemente[,] nume_tablou;

Creare: *nume_tablou=new tip_elemente[n,m];*

Exemplu: int[,] ma;

ma=new int[n,m];

Exemplu de initializare la declarare:

int[,] ma=new int[,]{{1,1},{2,3}}

Exemplu:

In programul urmator se creaza un tablou rectangular cu elemente a căror valoare este data de produsul liniei si coloanei pe care se afla (linia-dimensiunea1, coloana dimensiunea2).

```
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApplication3
{
    class Program
    {

        static void Main(string[] args)
        {
            int[,] m;
            m = new int[5,5];
            for (int i = 0; i < m.GetLength(0); i++)
                for (int j = 0; j < m.GetLength(1); j++)
                m[i,j] = i * j;
            for (int i = 0; i < m.GetLength(0); i++)
            {
                for (int j = 0; j < m.GetLength(1); j++)
                    Console.Write(" " + m[i,j]);
                Console.WriteLine();
            }
            Console.ReadLine();
        }}}
```

Se folosete metoda *GetLength(d)* a clasei *Array* pentru a determina numărul de elemente corespunzător dimensiuni d. Prima dimensiune se considera a fi cu valoarea 0.

2.5.6 Tablouri multidimensionale neregulate

Un tablou neregulat reprezintă un tablou de tablouri.

Declararea : *int[][] a;*

Accesarea elementelor se face prin *a[i][j]*.

Exemplu: Programul defineste un tablou neregular pe care îl afișează.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApplication3
{
    class Program
    {

        static void Main(string[] args)
        {
            int[][] a =
            {
                new int[] {3,6,8},
                new int[] {3,7,9,2,1},
                new int[] {1,2}
            };
            for (int i = 0; i < a.Length; i++)
            {
                for (int j = 0; j < a[i].Length; j++)
                    Console.Write(" " + a[i][j]);
                Console.WriteLine();
            }
            Console.ReadLine();
        }
    }
}
```

2.6 Instrucțiuni

Instrucțiunile limbajului C# sunt preluate din limbajul C, C++ cu mici modificări. Ele sunt de decizie, iterații si de control.

2.6.1 Instrucțiunea if

Sintaxa: *if (conditie)*

instructiune1

[else

instructiune2]

Semnificatie:

Se evaluează condiția, daca este adevărata se executa instructiune1, altfel daca exista se executa instructiune2.

Exemplu:

Se citește un număr n de la tastatura, sa se afișeze daca este par sau impar.

```
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApplication3
{
    class Program
    {

        static void Main(string[] args)
        {
            int n;
            Console.WriteLine("Introduceti numarul dorit=");
            n = int.Parse(Console.ReadLine());
            if (n%2==0)
                Console.WriteLine("Numarul este par");
            else
                Console.WriteLine("Numarul este impar");

            Console.ReadLine();
        }
    }
}
```

2.6.2 Instrucțiunea switch

Folosita pentru a evita if-urile imbricate.

Sintaxa*: switch (n)*

{

case val1:bloc_ instructiunea1;

break;

case val2: bloc_instructiunea2;

break;

……..

[default: bloc_instructiunel] }

Limbajul C# nu suporta căderea implicita la secțiunea următoare decât in cazuri vide. Forțarea treceri la un nou nivel folosind *goto case*.

Limbajul permite ca o variabila de tip sir de caractere sa fie comparata cu șiruri de caractere din *case*-uri.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApplication3
{
    class Program
    {

        static void Main(string[] args)
        {
            string s = "unu";
            switch(s)
            {
                case "unu": Console.WriteLine("Unu");
                            goto case "trei"; //folosit pentru caderea pe urmatoarea
ramura in lipsa break;
                case "doi":
                case "trei": Console.WriteLine("Trei");
                            break;
            }
            Console.ReadLine();
        }

          }
```

2.6.3 Instrucțiunea for

Este o instrucțiune repetitiva.

Sintaxa: for (expresie1;expresie2;expresie3)

{

bloc instrucțiune;

}

Semnificație: *expresie1* reprezintă etapa de inițializare si este prima care se executa. *Expresie2* reprezintă condiția de continuare sau nu a execuției blocului de instrucțiuni. In cazul in care se rezultatul in urma evaluării condiției este adevărat se executa *bloc instrucțiune* si apoi e*xpresie3*, care modifica pasul după care se reia evaluarea *expresie2* pana la obținerea unei valori de fals.

Exemplu: Se citește un număr natural n. sa se calculeze suma primelor n numere naturale.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApplication3
{
    class Program
    {

        static void Main(string[] args)
        {
            int n;
            Console.WriteLine("Introduceti n=");
            n = int.Parse(Console.ReadLine());
            int S = 0;
            for (int i = 0; i <= n;i++ )
                 S=S+i;
            Console.WriteLine("Suma este="+S);
                Console.ReadLine();
        }

    }
      }
```

2.6.4 Instructiunea foreach

Este folosita pentru iterarea printre elementele unui tablou sau ale unei colecții.

Sintaxa: *foreach (tip_identificator in expresie)*
bloc_instructiuni

Exemplu de folosire:

- *string s="Probleme"+ " de "+" rezolvat";*
  foreach (char c in s)
  Console.Write(c);
- *int[] a={10,11,19}*
  foreach (int x in a)
  Console.Write(x+" ");
  Console.WriteLine();

2.6.5 Instrucțiunea while

Este o instrucțiune repetitiva cu test inițial.

Sintaxa: *while (conditie)*
{
bloc_instructiuni;
}

Semnificatie: Se evaluează expresie, daca se obține o valoare de adevăr atunci se executa bloc instrucțiuni si se reia evaluarea expresiei. Structura se părăsește când se obține o valoare de fals pentru expresie.

Exemplu: Produsul primelor n numere naturale.

```
namespace ConsoleApplication4
{
    class Program
    {
        static void Main(string[] args)
        {
            int n;
            Console.WriteLine("n=");
            n = int.Parse(Console.ReadLine());
            long p = 1;int i=1;
            while (i <= n)
            {
                p = p * i;
                i++;
            }
            Console.WriteLine("p=" + p);
            Console.ReadLine();
        }
    }
       }
```

2.6.6 Instructiunea do-while

Sintaxa: *do*

{

bloc_instructiuni;

}

Semnificație:

Se executa blocul de instrucțiuni, după care se evaluează condiția. In cazul in care se obține o valoare de adevăr se reia etapa anterioara. Se părăsește structura in momentul in care valoarea expresiei devine falsa.

Exemplu: Sa se afișeze toate numerele care sunt palindrom mai mic decât un n citit.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

```
namespace ConsoleApplication4
{
    class Program
    {
        static void Main(string[] args)
        {
            int n;
            do
            {
                Console.WriteLine("n=");
                n = int.Parse(Console.ReadLine());
                if (n <= 0)
                    Console.WriteLine("nu s-a citi o valoare corecta pentru n");
            } while (n <= 0);
            int i=1;
            do
            {
                if (palindrom(i) == 1)
                 Console.Write(" " + i);
                i++;
            } while (i <= n);

            Console.ReadLine();
        }
        static int palindrom(int x)
        {
            int nr = x, nr1 = 0;
            while(nr>0)
            {
                nr1 = nr1*10 + nr % 10;
                nr = nr / 10;
            }
            if (x == nr1)
                return 1;
            else
                return 0;
        }

        }
       }
```

2.6.7 Instrucțiunea continue si break

Instrucțiune *continue* permite reluarea iterației celei mai apropiate structuri repetitive sau *switch*, iar *break* permite întreruperea iterației.

Exemplu:

Se citesc numere naturale si se calculează suma lor pana nu depășește o valoare n citita anterior. Citirea numerelor se oprește când s-au citit suficiente numere care sa satisfacă suma propusa.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApplication5
{
    class Program
    {
        static void Main(string[] args)
        {
            int n,nr;
            Console.WriteLine("n=");
            n = int.Parse(Console.ReadLine());
            int s = 0;
            for (; ; )
            {
                Console.WriteLine("nr=");
                nr = int.Parse(Console.ReadLine()); ;
                if (s + nr < n)
                {
                    s = s + nr;
                    continue;
                }
                else
                    break;
            }
            Console.WriteLine("suma la care s-a ajuns este " + s);
            Console.ReadLine();
        }

    }
      }
```

2.6.8 Instructiunea goto

Este folosita pentru realizarea de salturi in interiorul instrucțiunii *switch*.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

```
namespace @goto
{
    class Program
    {
        static void Main(string[] args)
        {
            int n;
            Console.WriteLine("alegeti una dintre variantele 1, 2 sau 3 ");
           do
           {
           Console.Write("varianta=");
            n=int.Parse(Console.ReadLine());
               switch (n)
               {
                   case 1:Console.WriteLine("Ati ales varianta 1");
                       goto case 4;
                   case 2: Console.WriteLine("Ati ales varianta 2");
                           break;
                   case 3: Console.WriteLine("Ati ales varianta 3");
                           break;
                   case 4: Console.WriteLine("Sunteti norocosi beneficiati si de varianta
4");
                           break;

               }
           }while (n!=1&& n!=2&& n!=3);
           Console.ReadLine();
        }

    }
      }
```

A doua varianta de folosire prin definirea unei etichete. Nu este recomandata deoarece ingreuneaza programele.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace eticheta
{
    class Program
    {
        static void Main(string[] args)
        {
            int i = 0,s=0;
        repeta:
            i++;
            s=s+i;
    if (i<5)
       goto repeta;
            Console.WriteLine("s="+s);
            Console.ReadLine();
```

```
    }
}
  }
```

3. Programare Orientata pe Obiect in C#

3.1 Noțiunea de OOP

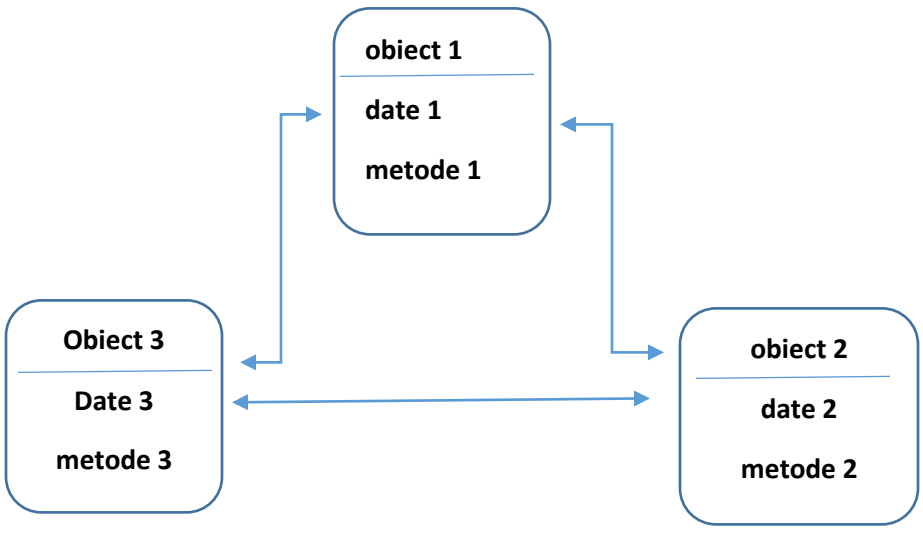
Programarea orientata pe obiect (OOP) este cel mai nou stil de programare.

In programarea procedurala se definesc tipuri de date si funcții care lucrează cu aceste tipuri, dar se păstrează o bariera intre acestea. Logica grupării acestora revine programatorului.

In OOP se definesc tipuri de date si metode care lucrează cu acestea, ele se grupează in același obiect.

Acest mod de organizare este net superior in conceperea aplicațiilor de mari dimensiuni, atunci când se lucrează in echipa.

Avantaje deosebite se obțin prin abstractizare. Programul este format dintr-un ansamblu de obiecte cu diverse proprietăți. Obiecte care interacționează intre ele.



3.2 Mecanismele fundamentale ale OOP

- Încapsularea.

Un obiect încapsulează date si cod care acționează asupra acestora. O regula de baza OOP este aceea ca o parte din membrii sa fie protejați, astfel încât sa nu fie accesați din afara, iar alți publici. Cei publici formează *interfața*.

- Abstractizarea

  Modelarea realității prin intermediul claselor care copiază caracteristicile obiectului real.

- Moștenirea

  Reprezintă mecanismul prin care o clasa deriva din alta clasa. Prin derivare înțelegem păstrarea comportamentului clasei si adăugarea de comportamente specifice noi clase.

  Exemplu: vrem sa construim un buton cu colturile rotunjite, luminos. Ne vom folosi de butonul existent (clasa Button din Forms) si vom adaugă codul cerut suplimentar.

  Principalul beneficiu al moștenirii este reutilizarea codului.

- Polimorfismul

  Reprezintă posibilitatea mai multor obiecte dintr-o ierarhie de clase de a utiliza metode cu același nume dar comportament diferit.

3.3 Noțiunea de clasa

Definiție : Clasa este o unitate logica care modelează un obiect real sau abstract. Ea încapsulează date si funcții care operează cu aceste date.

Datele pot fi câmpuri sau constante. Funcțiile pot fi: metode, proprietăți, constructori, destructori, operatori, evenimente.

Pentru definire se folosește cuvântul chei class.

Sintaxa:

```
class nume
        {
         //campuri
         [modificator acces] tip1 nume1;
         [modificator acces] tip2 nume2;
          ……..
          [modificator acces] tip n nume n;
           //metode
         [modificator acces] tip_returnat nume_metoda1 (Parametrii1)
         [modificator acces] tip_returnat nume_metoda2 (Parametrii2)
         ……...
         [modificator acces] tip_returnat nume_metoda n (Parametrii n)
    }
```

Modificatori de acces controlează accesul la membrii clasei.

Modificator acces	**Descriere**
public	Acces nelimitat
private	Acces limitat la propria clasa
protected	Acces limitat la propria clasa si la clasele derivate
internal	Acces limitat la programul care contine clasa
protected internal	Acces limitat la program si la clasele derivate

Pentru a crea un obiect dintr-o clasa se procedează conform exemplului de mai jos.

```
class A
  {
```

//membrii

}

A a=new A(); //crearea unui obiect de tip A

Accesarea membrilor unei clase se face prin folosirea operatorului „.”.

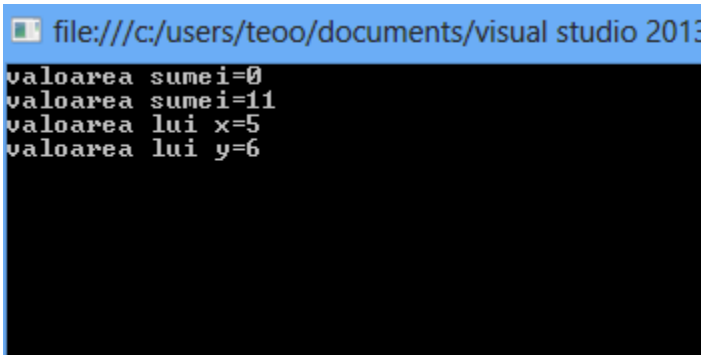
a.membru

Exemplu: programul următor definește 2 clase: clasa Numere si clasa Program. In clasa Program, in funcția Main se creează un obiect de tip Numere si apoi se apelează metodele clasei Numere.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace clasa1
{
  class Numere
  {
   private int x,y;//camp privat ce paote fi accesat doar din metodele sale
   public int Suma()
   {
       return x + y;
   }
    public void Init(int a, int b)
    {
        x=a;
        y=b;
    }
      public int Afisarex()
      {
          return x;
      }
      public int Afisarey()
      {
          return y;
      }
  }
    class Program
    {
        static void Main(string[] args)
        {
         Numere n=new Numere();
        // n.x=4;// campul x este privat si nu paote fi accesat din alta clasa.
         int s=n.Suma();//metode publice ce pot fi apelate
```

```
            Console.WriteLine("valoarea sumei="+s);
            n.Init(5,6);
            s=n.Suma();
            Console.WriteLine("valoarea sumei="+s);
            Console.WriteLine("valoarea lui x="+n.Afisarex());
            Console.WriteLine("valoarea lui y=" + n.Afisarey());
            Console.ReadLine();
        }

}
        }
```

Rezultatul:



3.1 Datele din interiorul unei clase

Datele din interiorul clasei sunt cunoscute ca variabile sau atribute. Tipul acestor date poate fi oricare dintre tipurile limbajului C#.

Declararea datelor. Sintaxa:

[modificator acces] tip_data nume;

Observații:

- In cazul in care modificatorul de acces lipsește se considera a fi private.
- Numele este un identificator stabilit de programator.

O clasificare a datelor membre poate fi considerata in: constante si câmpuri. Constantele sunt valori fixe declarate prin intermediul cuvântului cheie *const.*

Sintaxa:

[modificator acces] const tip identificator=expresieConstanta;

Tipul acestor date constante poate fi orice tip predefinit fără *object* si *enum.*

Exemplu:

```
class ExempluConstanta
  {
   public const int MAX=1000;
   public const string cale="\dosar\b";
   const double MIN=MAX/11.3;
  }
```

Campul reprezintă o variabila a clasei.

Sintaxa:

[modificator acces] tip identificator [=valoare];

Observații:

- Pe lângă modificatori de acces studiați anterior se mai adaugă si: *new, readonly, volatile, static;*
- După cerințe se poate face o inițializare explicita la declararea acestor câmpuri.

Exemplu:

```
class Data
  {
    public int nota;
    protected string nume;
    int id;//implicit este un camp privat
    static void Main()
```

```
Data o=new Data();
 o.id=13;
}
}
```

Un câmp de instanța reprezintă un câmp pentru care nu se specifica modificatorul static. In acest caz el se regăsește in orice obiect instanția al clasei.

o.id=13 //accesarea câmpului de instanța

Un câmp static este cel care folosește modificatorul *static.* Accesarea unui astfel de câmp din afara clasei se poate face doar prin numele clasei (câmpul aparține doar clasei).

```
class Data
  {
 public static n=4;
Static void Main()
{
        Data.n--;
       }
    }
```

Câmpurile readonly sunt specificate de modificatorul *readonly*. Inițializarea (modificarea valori) se face fie la declarare fie prin intermediul constructorului.

```
class Data
    {
     public readonly int x=14;
```

```
public readonly int y;
public class Data(int n)
  {
   this.y=n;
   }
}
```

Câmpurile volatile, folosesc modificatorul *volatile*. Astfel de câmpuri pot fi doar de următoarele tipuri.

- *byte, sbyte, short, ushort, int, uint, char, float, bool;*
- enumerare de tip *byte, sbyte, short, ushort, int, uint;*
- tip referința;

Inițializarea implicita a câmpurilor unei clase respecta valorile din tabel:

Tip	Valoare
numeric	0
bool	false
char	0
enum	0
referinta	null

3.5 Funcțiile clasei

Funcțiile din interiorul clasei pot sa fie de următoarele tipuri: constructorii, destructori, metode, proprietăți, evenimente, indexatori si operatori.

3.5.1 Constructori

Definiție: Constructorul este o funcție speciala care are rol in crearea si construirea obiectelor. La fiecare instanțiere a unui obiect dintr-o clasa se apelează constructorul clasei.

Observații:

- ❖ Toate clasele au constructori(cel puțin unul);
- ❖ Constructorii au același nume cu cel al clasei;
- ❖ Constructorii nu returnează nimic si pot avea modificatori de acces;
- ❖ Se pot defini proprii constructori, in lipsa lor se apelează cei impliciți;
- ❖ Un constructor implicit inițializează câmpurile clasei cu valori implicite;
- ❖ Constructorii pot fi declarați statici pentru inițializarea membrilor statici ai claselor.
- ❖ Intr-o clasa pot exista mai mulți constructori in funcție de setările dorite pentru câmpuri. Diferențierea dintre ei se face prin parametrii din lista (numărul acestora si/sau tipul acestora).

Sintaxa:

modificator_acces nume_constructor([parametri])([initializator]

[{

corp_constructor;

}]

Observații: *„initializator”* permite invocarea numelui unui constructor anume înaintea execuției corpului constructorului curent. El are doua forme:

base ([parametri]) si *this ([parametri])*. Daca nu este precizat in mod automat el este *base()*

Exemplu:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace punct
{
    class Punct
    {
        public double x;
        public double y;
        public Punct()
        {
            //constructor gol fara corp
        }
        public Punct(double a)
        {
            x = a;
        }
        public Punct(double a, double b)
        {
            x = a;
            y = b;
        }
        static void Main()
        {
            Punct A, B, C;
            A = new Punct();
            Console.WriteLine(A.x+" "+A.y);
            B=new Punct(7.5);
           Console.WriteLine(B.x+" "+B.y);
            C=new Punct(9,6);
           Console.WriteLine(C.x+" "+C.y);
            Console.ReadLine();
        }
    }
}
```

Exista si notiunea de constructor de copiere atunci când se dorește crearea de obiecte copii fidele ale altor obiecte.

Exemplu:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace copiere_constructor
{
    class Copie
    {
        private int a;
```

```
        public Copie( int n)
        {
            a = n;
        }
        public Copie(Copie C)
        {
            a = C.a;
        }
        public int Afis()
        {
            return a;
        }
        static void Main()
        {
            Copie C1 = new Copie(5);
            Copie C2 = new Copie(C1);
            Console.WriteLine(C2.Afis());
            Console.ReadLine();
        }
    }
}
```

3.5.2 Destructori

O clasa poate sa aiba un singur destructor si mai mulți constructori. Rolul destructorului este acela de a distruge obiectele(instanțele clasei).

Observații:

- ❖ Nu are parametrii
- ❖ Nu are modificator de acces
- ❖ Se apelează automat
- ❖ Nu poate fi moștenit
- ❖ Este precedat de caracterul „~”

Exemplu:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace destructor
{
    class Punct
    {
```

```
        public int x;
        public int y;
        public Punct(int a, int b)
        {
            x = a;
            y = b;
        }
        ~Punct()
        {
            Console.WriteLine("Apel destructor");
        }
    }
        class Program
        {
        static void Main()
        {
            Punct P = new Punct(10, 10);
            Console.WriteLine(P.x + " " + P.y);
            Console.ReadLine();
        }
    }
}  ??????????????????????
```

3.5.3 Metode

Sunt funcții care implementează acțiuni.

Sintaxa:

[modificator_acces] tip_returnat nume_metoda ([parametrii])

[{

corp_metoda;

}]

Observații:

- In lipsa modificatorului acesta este *private*
- Pe lângă modificatorii standard se adaugă si *new, static, virtual, sealed, averride, abstract, extern*
- Poate fi orice tip definit sau void când nu returnează nimic
- *nume_metoda* este un identificator. Atunci când definește doar un membru al interfeței este precedat de numele interfeței

[nume_interfata].[nume_metoda]

- lista de parametri este formata dintr-o succesiune de declarări de forma următoare:
  [atribut][modificator] tip_parametru nume_parametru

  Unde, modificator poate sa fie *ref (*parametri de intrare, parametri ieșire), *aut (*parametri de ieșire). In lipsa acestui modificator parametrii sunt considerați de intrare si se transmit prin valoare (la ieșire din metoda își pierd valoarea).

Numele metodei trebuie sa fie diferit de cel al altor membrii. Apelul metodelor se face folosind operatorul „ .”. Exista doua forme de apel:

[nume_obiect].[nume_metoda] → *pentru functii nestatice*
[nume_clasa].[nume_metoda] → *pentru functii statice*

Suprancarcarea metodelor.

Operația se numește overloading. Presupune ca doua sau mai multe metode sa aibă același nume in aceeași clasa. Metodele trebuie totuși sa difere prin numărul si/sau tipul parametrilor formali.

Exemplu:

```
class Supraincaracare
  {
    void M(int x) {}
    void M(char y){}//supraincaracare corecta;
    void M(int x, char y){}//supraincaracare corecta;
    int M(int x){}//incorect, acelasi tip de parametru formal)
    }
```

3.5.4 Proprietatii

Este o functie a clasei ce permite sa accesam sau/si sa modificam caracteristicile unui obiect al clasei.

Vin in sprijinul accesării câmpurilor private, ca si cum ar fi declarate publice. Nu se încalecă protecția acestor câmpuri.

Sintaxa:

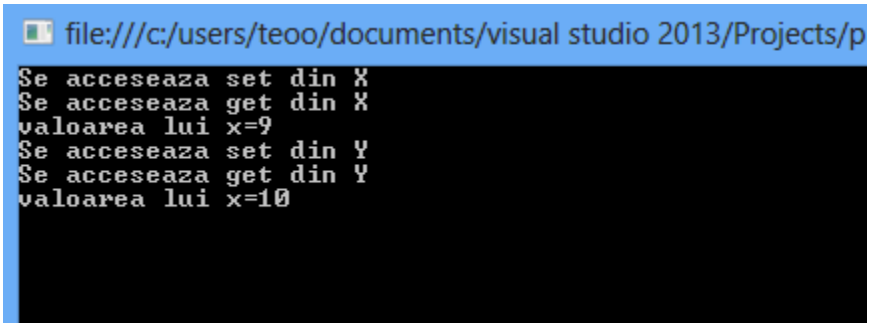
```
[atribut] modificator_acces tipReturnat numeProprietate
    {
      get{
          }
      set{
          }
     }
```

Observații:

- modificator acces, poate fi orice tip clasic de modificator la care se adaugă: *new, static, virtual, sealed, override, abstract, extern*.
- *tipReturnat,* poate fi orice tip al limbajului C#.
- *„get”* corespunde unei metode fără parametrii si returnează o valoare de același tip cu al proprietății.
- *„set”* corespunde unei metode cu un singur parametru al cărui tip este același cu tipul proprietății. Nu returnează nimic (*void).*
- Este modalitatea recomandata pentru accesarea câmpurilor si protejarea lor in același timp. Parametrul care se transmite in cazul „*set”* nu apare explicit in sintaxa, el este implicit identificat prin „*value”*.

Exemplu:

```
  using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace proprietatii
{
    class Punct
    {
        private double x;
        private double y;
        public double P1
        {
            get
            {
                Console.WriteLine("Se acceseaza get din X");
                return x;
            }
            set
            {
                Console.WriteLine("Se acceseaza set din X");
                x = value;
            }
        }
        public double P2
        {
            get
            {
                Console.WriteLine("Se acceseaza get din Y");
                return y;
            }
            set
            {
                Console.WriteLine("Se acceseaza set din Y");
                y = value;
            }
        }
    }

    class Program
    {
        static void Main()
        {
            Punct P = new Punct();
           // P.x = 3;//Error      1      'proprietatii.Punct.x' is inaccessible due to its
protection level- incorect
            P.P1 = 9;//se apeleaza set sin P1.
            Console.WriteLine("valoarea lui x="+P.P1);

            P.P2 = 10;//se apeleaza set din P2.
            Console.WriteLine("valoarea lui x=" + P.P2);
            Console.ReadLine();
        }
    }
}
```



- In interiorul lui set se poate plasa orice cod defensiv de protectie a câmpului asociat acestor proprietății.

```
namespace proprietatii
{
    class Punct
    {
        private double x;
        private double y;
        public double P1
        {
            get
            {
                Console.WriteLine("Se acceseaza get din X");
                return x;
            }
            set
            {
                Console.WriteLine("Se acceseaza set din X");
                if (value > 3000)
                    x = 3000;
                else
                x = value;
            }
                  }
          ……..

class Program
    {
        static void Main()
        {
            Punct P = new Punct();
           // P.x = 3;//Error      1      'proprietatii.Punct.x' is inaccessible due to its
protection level- incorect
            P.P1 = 3001;//se apeleaza set sin P1.
            Console.WriteLine("valoarea lui x="+P.P1);
}}
```

- Daca dorim sa asociem proprietatea unui câmp static trebuie sa o declaram statica.
- Nu pot fi supraîncărcați.

3.5.5 Evenimente si delegări

Evenimentele sunt membrii dintr-o clasa ce realizeaza notificari privind starea lor catre celelalte obiecte.

Clasa in care a avut loc evenimentul pune la dispoziția celorlalte clase un *delegat*, o referința către o funcție necunoscuta care va avea doar un antet specificat, urmând ca ea sa fie implementata la nivelul fiecărei clase interesata de eveniment.

Implementarea metodelor ce răspund la evenimente poarta numele de *„tratarea evenimentelor”.*

Sintaxa:

[atribut] [modificatorAcces] even tipDelegat nume

Unde,

- *modificatorAcces* sunt cei standard.
- *tipDelegat* este un tip de date derivat din clasa *Delegate* a spațiului *System*.

Definirea unui *tipDelegat* se realizează:

[atribut][modificatorAcces] delegate tipRezultat nume[listaParametri]

Exemplu:

Clasa Vector cu metodele: constructor Vector, Afisare, si antetul si o metoda delegate. Clasa Program in care se implementează metoda *delegate* si metoda *Main()* . Programul verifica daca vectorul este crescător sau descrescător.

Black sails

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace @delegate
{
    public delegate bool comparare(object t1, object t2);// antet functie delegate
    class Vector
    {
        public const int max = 6;
        public int[] v=new int[max];
        public Vector()
        {
            for(int i=0;i<max;i++)
            {
                Console.WriteLine("v["+i+"]=");
                v[i]=int.Parse(Console.ReadLine());
            }
        }

        public void Afis()
        {
          for(int i=0;i<max;i++)
              Console.WriteLine(" "+v[i]);
        }

        public bool verificare(comparare ok)//ok este delegare catre functie necunoscuta
        {
            for(int i=0;i<max-1;i++)
                if (!ok(v[i],v[i+1])) return false;
                    return true;
        }

    }

    class Program
    {   public static bool descrescator(object t1, object t2)
       {
        if ((int)t1>=(int)t2) return true;
        else
            return false;
       }

public static bool crescator(object t1, object t2)
       {
        if ((int)t1<=(int)t2) return true;
        else
            return false;
       }

        static void Main()
        {
            Vector a;
            a=new Vector();
            a.Afis();
            if (a.verificare(crescator))
```

```
                Console.WriteLine("vector crescator");
            if(a.verificare(descrescator))
              Console.WriteLine("vector descrescator");
            Console.ReadLine();
        }
    }
}
```

3.5.6 Operatori

Un operator ca membru al unei clase reprezinta o operatie care poate fi aplicata unei instanțe a clasei.

Operatorul ca membru reprezintă o supraîncărcare. C# permite redefinirea semnificației unui operator standard.

Sintaxa:

[atribut] modificatorOperator decalaratieOperator corpOperator

Observații:

- declararea operatorului trebuie sa fie publica sau statica
- nu se accepta drept parametrii decât cei valoare
- operatori unari au un singur parametru
- operatorii binari au doi parametrii
- operatorul „=” nu se supraîncarcă
- daca supraîncărcam „==” atunci suntem obligați sa supraîncărcam si „!=” (la fel in cazul „<”,”>” analog cu „<=”,”<=”.

Operatorii unari.

Sintaxa:

tip operator operatorUnar(tip operand)

{

}

Pot fi supraîncărcați următori operatori unari:+, -, !, ~, ++, --, true, false;

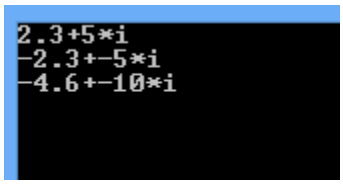
Observații:

- *tipOperand* trebuie sa fie de tipul clasei in care se definește operatorul unar
- operatorii +, -, !, ~ returnează orice tip de date
- operatorii ++, -- returnează un rezultat de tipul clasei
- operatorii true si false returnează tipul bool.

Exemplu: Clasa Complex in care vom supraîncarcă -,++.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace operatori_unari
{
    class Complex
    {
       public double x;
        public double y;
        public Complex(double a,double b)
        {
            x = a;
            y = b;
        }
        public static Complex operator -(Complex
        {
         a.x=-a.x;
         a.y=-a.y;
        return a;
         }
        public static Complex operator ++(Complex a)
        {
        a.x=2*a.x;
        a.y=2*a.y;
        return a;
        }
    }

    class Program
    {
        static void Main()
        {
             Complex C;
            C=new Complex(2.3,5);
            Console.WriteLine(C.x+"+"+C.y+"*i");
            C=-C;
            Console.WriteLine(C.x+"+"+C.y+"*i");
            C++;
            Console.WriteLine(C.x + "+" + C.y + "*i");
```

```
            Console.ReadLine(); }}}
```



Operatori binari.

Sintaxa:

tip operator operatorBinar (tip operand, tip operand)

{

}

Pot fi supraîncărcați operatorii binari: +, -, *, /, %, &, |, ^, <<, >>, ==, !=,<, >, <=, >=

Observație:

- cel puțin unul dintre *tip operand* trebuie sa fie de tipul clasei

Exemplu: Clasa *Fractie*, supraîncarcă operatorii: +, *, ==,!=

```
using System;
using System.Collections.Generic;

namespace fractie_supraincaracre
{
    class Fractie
    {
        public int x;
        public int y;
        public Fractie(int a, int b)
        {
            x = a;
            y = b;
        }
        public int cmmdc(int a, int b)
        {
            if (a == b)
                return a;
            else
                if (a > b)
                    return cmmdc(a - b, b);
```

```
            else
                return cmmdc(a, b - a);
        }
        public static Fractie operator +(Fractie f1, Fractie f2)
        {
        Fractie f = new Fractie(1,1);
        f.x=(f1.x*f2.y+f2.x*f1.y);
        f.y=f1.y*f2.y;
        int c=f.cmmdc(f.x,f.y);
        f.x=f.x/c;
        f.y=f.y/c;
        return f;
        }
    public static Fractie operator*(Fractie f1, Fractie f2)
        {
        Fractie f = new Fractie(1, 1);
        f.x=f1.x*f2.x;
        f.y=f1.y*f2.y;
        int c=f.cmmdc(f.x,f.y);
        f.x=f.x/c;
        f.y=f.y/c;
        return f;
        }
    public static bool operator ==(Fractie f1, Fractie f2)
    {
        if ((f1.x == f2.x) && (f1.y == f2.y))
            return true;
        else
            return false;
    }
    public static bool operator !=(Fractie f1, Fractie f2)
    {
        if (f1==f2)
            return false;
        else
            return true;
    }
    }

    class Program
    {
        static void Main()
        {
            Fractie F,G;
            F = new Fractie(5, 25);
            G = new Fractie(7, 100);
            F = F + G;
            Console.WriteLine(F.x + "/" + F.y);
            F = F * G;
            Console.WriteLine(F.x + "/" + F.y);
            if (F==G)
                Console.WriteLine("fractii egale");
            else
                Console.WriteLine("fractiile nu sunt egale");
            Console.ReadLine();
        }}}
```

Operatori de conversie.

Operatorii pot fi de conversie implicita sau explicita.

Sintaxa:

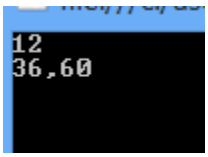
implicit operator tip (tip operand) {}

explicit operator tip (tip operand) {}

Se poate realiza conversia dintr-un tip de baza într-o clasa si invers sau dintr-un tip de clasa in alt tip de clasa.

Exemplu: Conversia dintr-un tip de clasa in alt tip de clasa.

```
using System;
using System.Threading.Tasks;

namespace ConsoleApplication7
{
    class C1
   {
    public int x;
    public C1(int a)
      {
       x=a;
      }
    public static explicit operator C2(C1 c1)
     {
     C2 c2=new C2(c1.x*3,c1.x*5);
    return c2;
     }
}
  class C2
{
 public int x;
 public int y;
 public C2(int a,int b)
 {
     x=a;
     y=b;
 }

}
    class Program
    {
        static void Main()
        {
            C1 c1=new C1(12);
            Console.WriteLine(c1.x);
            C2 c2=(C2)c1;
            Console.WriteLine(c2.x+","+c2.y);
            Console.ReadLine();
        }}}
```



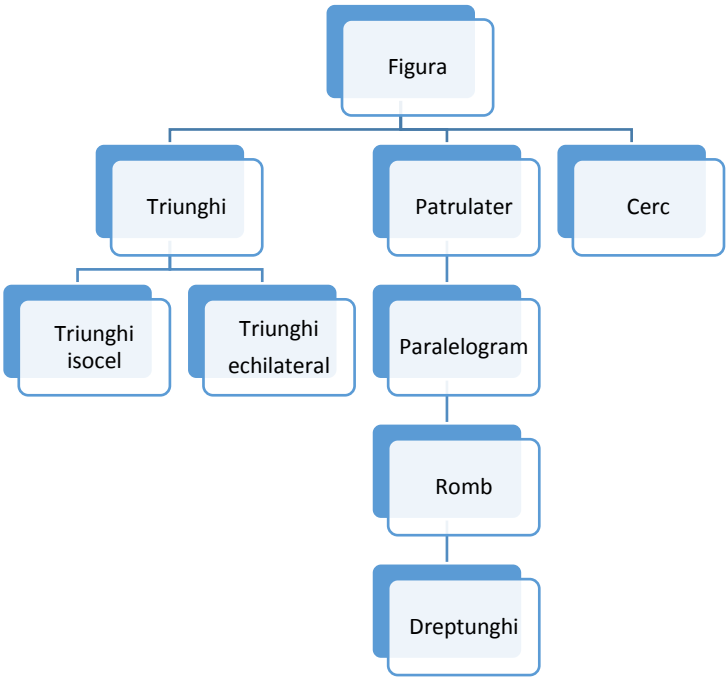
3.6 Moștenirea

3.6.1 Noțiuni generale

Este unul dintre principiile de baza ale OOP. Ea permite definirea unor noi clase pornind de la o clasa de baza, spunem ca o extinde.

Pe baza procesului de moștenire se pot crea ierarhii de clase.

Exemplu:



Ierarhie de clase

Esența moștenirii consta in a spune ca un obiect al clasei derivate este un obiect al clasei de baza.

Exemple:

Triunghiul isoscel este un *Triunghi*

Triunghiul este o *Figura*

Paralelogramul este un *Patrulater*

Extinderea presupune o *specializare*, iar moștenirea tuturor caracteristicilor clasei de baza presupune o *generalizare.*

3.6.2 Implementarea moștenirii

Fie clasa *Triunghi* care moștenește clasa *Figura:*

```
class Figura    //clasa de baza
    {
    //membrii clasei de baza;}
    class Triunghi: Figura //clasa care deriva din Figura
      {
    //membrii clasei triunghi
      }
```

Clasa *Triunghi* va include toti membrii clasei *Figura* si in plus proprii membrii (excepție constructorii clasei de baza nu pot fi moșteniți).

Observații:

- In C# toate clasele sunt derivate. Orice clasa deriva in mod direct sau indirect din clasa *object* (ea sta la baza tuturor ierarhilor de clase).
- O clasa nu poate moșteni decât o singura clasa
- Nu se accepta in C# moștenirea multipla
- Adâncimea ierarhiei de moștenire nu are limite

class B:A{}

class C:B{}

class D:C{}

……

3.6.3 Accesarea membrilor moșteniți

Un obiect al unei clase derivate moștenește toți membrii clasei de baza cu excepția constructorilor. Totuși prin intermediul *modificatorilor de acces* se poate restricționa accesul la uni dintre ei.

Modificatori de acces sunt: *public, private, protected, internal si protected internal*.

Observații:

- Metodele clasei derivate pot accesa membrii publici si protejați ai clasei de baza.
- Metodele unei clase pot accesa orice membru al clasei respective indiferent de nivelul de protecție
- In afara clasei de baza si a celei derivate se pot accesa numai membrii publici ai clasei de baza.
- Membrii unei clase marcați cu *internal* sunt vizibili pentru toate clasele din același fișier
- Membrii unei clase marcați cu *protected internal* sunt vizibili tuturor celor care o moștenesc.

Exemplu:

Accesul la membrii claselor Persoana si Elev in funcție de modificatorii de acces folosiți.

```
using System;
using System.Collections.Generic;
using System.Linq;
```

```
using System.Text;
using System.Threading.Tasks;

namespace acces_membrii
{
    class Persoana
    {
       private int nr_legitimatie;//camp privat
       protected string nume;  //camp protejat
       public int Nr_legitimatie//proprietate asociata campului privat
       {
           set
           {
               nr_legitimatie=value;
           }
           get
           {
            return nr_legitimatie;
           }
       }
       protected void Activitate()// o metoda protejata
       {
           Console.WriteLine(nume+"Persoana desfasoara o activitate");
       }
       }
       class Elev:Persoana
       {
        private string clasa;
        public string Clasa//proprietatea asociata campului privat clasa
        {
        set
           {
               clasa=value;
           }
           get
           {
            return clasa;
           }
        }
           public void ActivitateElev()
           {
           // nr_legitimatie=100;//incorect camp privat in clasa de baza;
            Nr_legitimatie=100;//corect acces pri proprietate;
            nume="Marinica";//corect acces la camp protejat din  clasa de baza;
            Activitate();//corect metoda protejata in clasa de baza;
            Console.WriteLine("Tocmai s-a apelat Activitate_elev cu legitimatia
"+Nr_legitimatie);
           }
        }

    class Program
    {
        static void Main()
        {
            Persoana P=new Persoana();
           // P.nr_legitimatie=300;//incorect in afara clasei de baza si cea derivata s-a
accesat un membru protejat
           // P.nume="Grigore";//incorect in afara clasei de baza si a clasei derivate s-a
incercat accesul la un membru protejat;
            P.Nr_legitimatie=300;//corect, acces prin proprietate publica
            //P.Activitate();//incorect acces prin metoda protejata
```

```
            ///accesul print-un obiect al clasei derivate
            Elev E=new Elev();
           // E.nr_legitimatie=300;//incorect in afara clasei de baza si cea derivata s-a
accesat un membru protejat
            //E.nume="Grigore";//incorect in afara clasei de baza si a clasei derivate s-a
incercat accesul la un membru protejat;
            E.Nr_legitimatie=300;//corect, acces prin proprietate publica
           // E.clasa="XB";//incorect in afara clasei derivate s-a incercat o accesare a
unui camp privat
            E.Clasa="XB";//corect in afara clsei s-a accesat o proprietate publica
            E.ActivitateElev();//corect s-a acesat o metoda publica a clasei derivate
            Console.ReadLine();
        }
    }
       }
```

3.6.4 Constructorii claselor derivate

Constructorii nu se mostenesc in clasele derivate dar se pot apela pentru construirea unor porțiuni de obiect.

Observații:

- La crearea unui obiect al unei clase derivate se apelează implicit constructorul clasei de baza, apoi se executa codul.
- Se poate realiza si un apel explicit folosind sintaxa:

  public ClasaDerivata: base()

  {

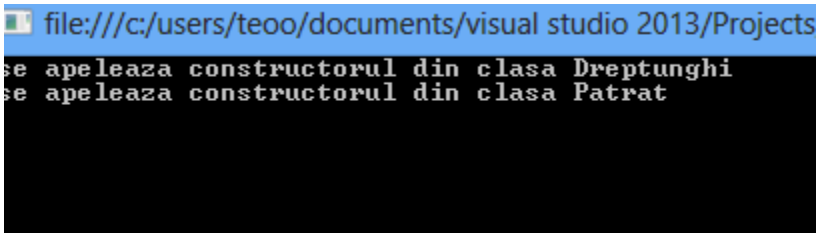
  }

Cazul devine obligatoriu atunci când clasa de baza are constructor cu parametrii.

Exemplu: Clasa de baza *Dreptunghi* si clasa derivata *Patrat*.

```
 Using System;
using System.Collections.Generic;
namespace constructori_derivati
{
    class Dreptunghi
    {
        public int l;
        public int L;
        public Dreptunghi()
```

```
        {
            Console.WriteLine("se apeleaza constructorul din clasa de Dreptunghi");
        }
    }
    class Patrat:Dreptunghi
    {
        public Patrat()
        {
            Console.WriteLine("se apeleaza constructorul din clasa Patrat");
            L = 1;
        }
    }
    class Program
    {
        static void Main()
        {
            Patrat P = new Patrat();
            Console.ReadLine();
        }
    }
}
```



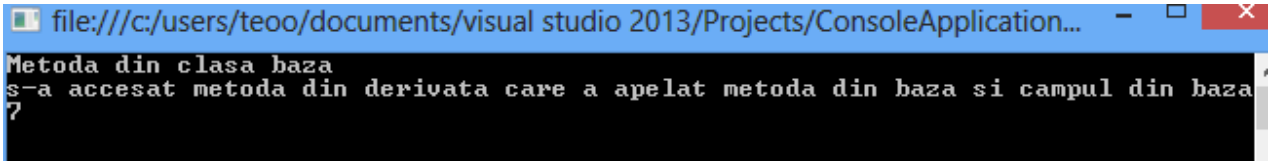
3.6.5 Membri ascunși

Exista posibilitatea ca unul dintre câmpurile clasei derivate sa aibă același nume cu unul dintre câmpurile clasei de baza(sau o metoda sa aibă aceeași signatura).in acest caz se spune ca membrii clasei de baza sunt ascunșii clasei moștenitoare. Pentru accesarea acestora membrii vor folosi cuvântul cheie *new* plasat in fata membrilor din clasa derivata si base la accesare.

Exemplu:

```
using System;
using System.Collections.Generic;
namespace ConsoleApplication8
{
    class Baza
    {
        public int x=7;
        public void Metoda()
        {
```

```
            Console.WriteLine("Metoda din clasa baza");
        }
    }
    class Derivata:Baza
    {
     new public int x=10;
     new public void Metoda()
     {
      base.Metoda();
      Console.WriteLine("s-a accesat metoda din derivata care a apelat metoda din baza si
campul din baza"+base.x);
     }
    }
    class Program
    {
        static void Main()
        {
            Derivata d=new Derivata();
            d.Metoda();
            Console.ReadLine();

        }
    }
}
```



3.7 Polimorfismul

Polimorfismul, drept concept fundamental al OOP, permite un comportament diferențiat al unei entități. Este caracteristica unei variabile de a putea referi obiecte de tipuri diferite.

Din punct de vedere al moștenirii, polimorfismul permite invocarea metodelor derivate cu ajutorul referințelor la clasa de baza.

3.7.1 Conversia referințelor

Putem transforma o referința la clasa derivata într-o referința la clasa de baza, prin intermediul operatorului de conversie.

Exemplu:

```
Using System;
using System.Collections.Generic;
namespace ConsoleApplication9
{
    class Baza
    {
        public void Afis()
        {
            Console.WriteLine("Afis() din Baza");
        }
    }
    class Derivata : Baza
    {
        new public void Afis()
        {
            Console.WriteLine("Afis() din derivata");
        }
    }
    class Program
    {
        static void Main()
        {
            Derivata D = new Derivata();
            D.Afis();
            Baza B = (Baza)D;//s-a realizat conversia referintei catre clasa de baza;
            B.Afis();
            Console.ReadLine();
        }
    }
}
```

3.7.2 Metode virtuale

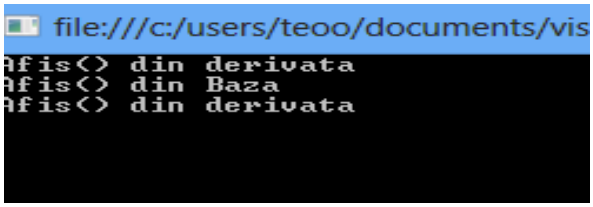
O referința la un obiect din clasa de baza nu poate invoca o metoda definita cu new.

Pentru realizarea polimorfismului bazat pe moștenire, trebuie ca in clasa de baza sa fie definit membrul *virtual*, iar la redefinirea membrului cu aceeași signatura in clasa derivata se adaugă *override*.

Exemplu:

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
namespace ConsoleApplication9
{
    class Baza
    {
```

```
        public virtual void Afis()
        {
            Console.WriteLine("Afis() din Baza");
        }
    }
    class Derivata : Baza
    {
        override public void Afis()
        {
            Console.WriteLine("Afis() din derivata");
        }
    }
    class Program
    {
        static void Main()
        {
            Derivata D = new Derivata();
            D.Afis();
            Baza B= new Baza();
            B.Afis();
            B = new Derivata();
            B.Afis();
            Console.ReadLine();
        }
    }
}
```

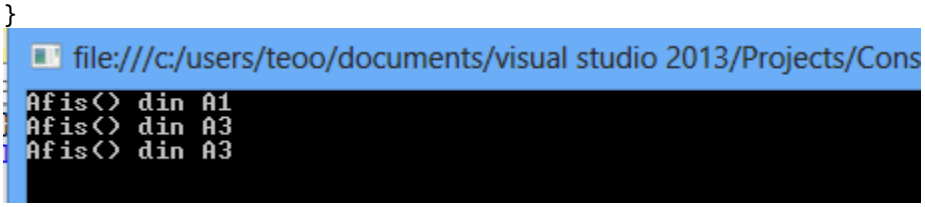


Pentru un lant de derivare pornind de la clasa de baza cu definirea unei metode virtuale, in clasele derivate putem sau nu sa redefinim metodele respective.

Exemplu:

```
using System;
using System.Threading.Tasks;
namespace ConsoleApplication10
{
    class A1
    {
        virtual public void Afis()
        {
            Console.WriteLine("Afis() din A1");
        }
    }
    class A2 : A1
    {
    }
    class A3 : A2
    {
     override  public void Afis()
        {
```

```
            Console.WriteLine("Afis() din A3");
            }
        }
    class A4 : A3
    {

    }

    class Program
    {
        static void Main()
        {
            A1 a = new A2();
            a.Afis();
            a = new A3();
            a.Afis();
            a = new A4();
            a.Afis();
            Console.ReadLine();
        }
    }
}
```



Observații:

- Membrii cum ar fi: metode, proprietăți, evenimente, indexatori pot fi declarați *virtual*.
- Câmpurile nu pot fi declarate virtual.
- Metodele *override* nu pot avea alt modificator de acces decât cel al metodei *virtual*.
- Metodele statice nu pot fi redefinite.

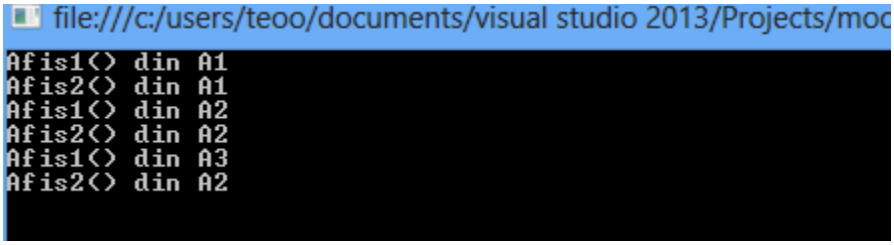
3.7.3 Modificatorul sealed

Plasat în fața unei metode sau a unei proprietății interzice moștenirea acestora în derivare (împiedica redefinirea lor).

Exemplu:

```
using System;
using System.Threading.Tasks;
```

```
namespace modisealed
{
    class A1
    {
        virtual public void Afis1()
        {
            Console.WriteLine("Afis1() din A1");
        }
        virtual public void Afis2()
        {
        Console.WriteLine("Afis2() din A1");
        }
    }
    class A2 : A1
    {
        override public void Afis1()
        {
            Console.WriteLine("Afis1() din A2");
        }
        sealed override public void Afis2()
        {
            Console.WriteLine("Afis2() din A2");
        }
    }
    class A3 : A2
    {
        override public void Afis1()
        {
            Console.WriteLine("Afis1() din A3");
        }
       /* override public void Afis2()//Error   1      'modisealed.A3.Afis2()': cannot
override inherited member 'modisealed.A2.Afis2()' because it is sealed
        {
            Console.WriteLine("Afis2() din A3");
        }*/
    }
    class A4 : A3
    {

    }
    class Program
    {
        static void Main()
        {
            A1 a = new A1();
            a.Afis1();
            a.Afis2();
             a = new A2();
            a.Afis1();
            a.Afis2();
             a = new A3();
            a.Afis1();
            a.Afis2();
            Console.ReadLine();
        }
    }
}
```

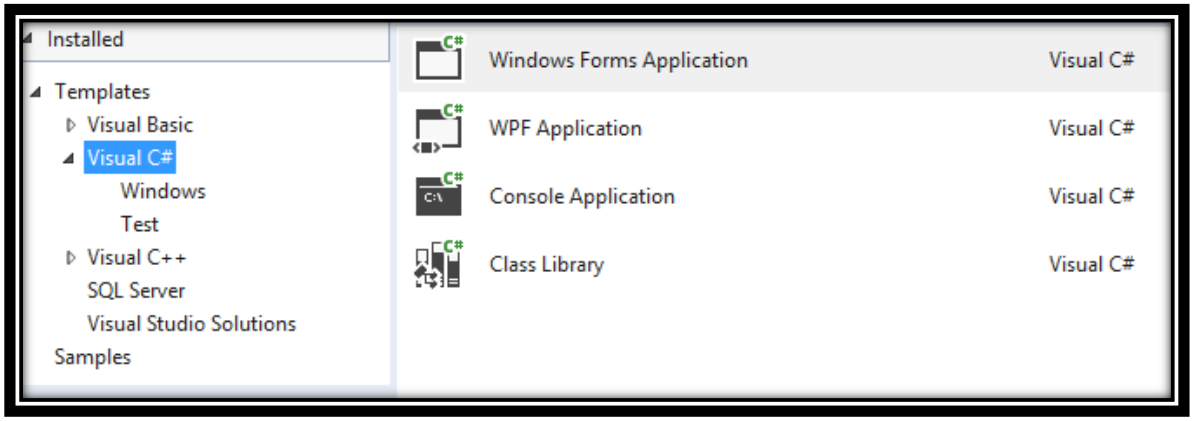


Utilitatea polimorfismului se rezumă la posibilitatea utilizări unei colecții de clase care derivă dintr-o clasă de bază, clase care utilizează metode declarate virtuale în clasa de bază.

4. Progrmare vizuala

Visual C# 3013 ofera posibilitatea crearii urmatoarelor tipuri de aplicatii:

- ❖ Windows Forms Application
- ❖ WPF Application
- ❖ Console Application
- ❖ Class Library



Primele doua varianta de aplicatii permit folosirea obiectelor grafice cu scopul realizarii interfetei cu utilizatorii.

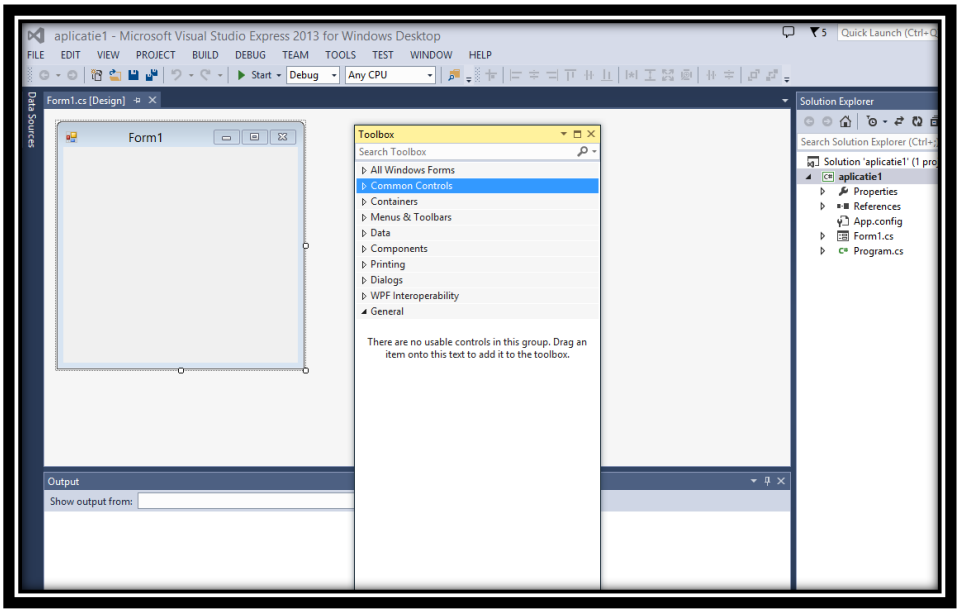
Printre elementele de interfata se folosesc controalele Windows. In realizarea de aplicatii ne vom concentra astfel pe functionalitatea acesteia, utilizand pentru partea de interfata elementele deja existente.

4.1 Crearea unei aplicatii Windows Forms

Pentru realizarea unei aplicatii de acest tip vom urma etapele:

- ❖ *New Project → Visual C# → Windows Forms Applications*
- ❖ Se introduce numele aplicatiei si se alege locul salvarii acesteia

- In fereastra *Solution Explorer* vom putea vizualiza toate fisierele proiectului. *Form1.cs* este formularul implicit care se deschide. *Program.cs* este fisierul corspunzator codului.
- Accesam *Toolbox* de unde putem sa selectam orice forma Windows dorita (control)



- Selectam control-ul dorit si il tragem cu mouse-ul pe formular.
- Stabilim proprietatile pentru control-ul stabilit, prin apasare click dreapta → *Properties*.
- Din caseta anterioara putem sa stabilim un eveniment, la alegerea acestuia se deschide metoda asociata evenimentului. Metoda ce trebuie completata.
- Pentru compilarea si rulare se apasa F5.

4.2 Controale Windows Forms

O clasificare a controlalelor dupa utilitatea lor se poate face astfel:

- Controale pentru editarea de text (exemplu: *TextBox*, *RichTextBox*)
- Controale pentru declansarea evenimentelor (exemplu: *Button*)
- Controale de afisare a informatiilor (exemplu: *Label*, *LinkLabel*)

- Controale pentru liste (exemplu: *ListView*, *ListBox*)
- Controale de tip container (*GroupBox*, *Panel*)
- Controlae de tip meniu (*MenuStrip*, *ToolStrip*)

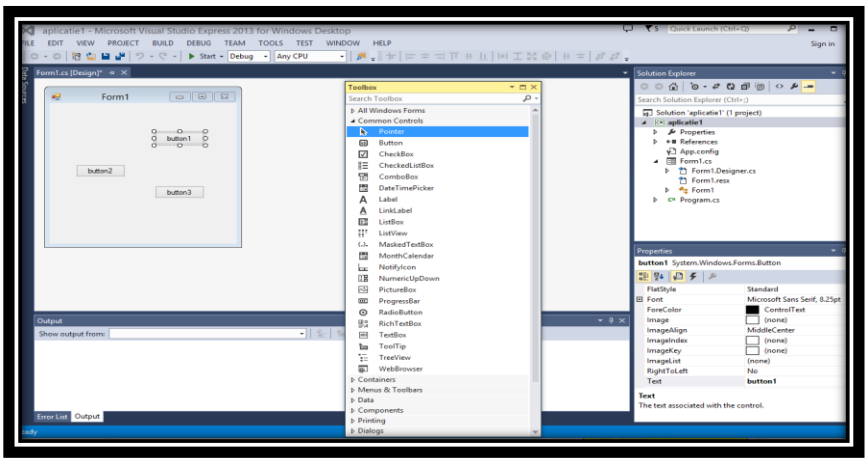
4.2.1 Controlul Button

Este cel mai des utilizat control. Pentru exemplificarea folosirii lui vom crea urmatoarea aplicatie:

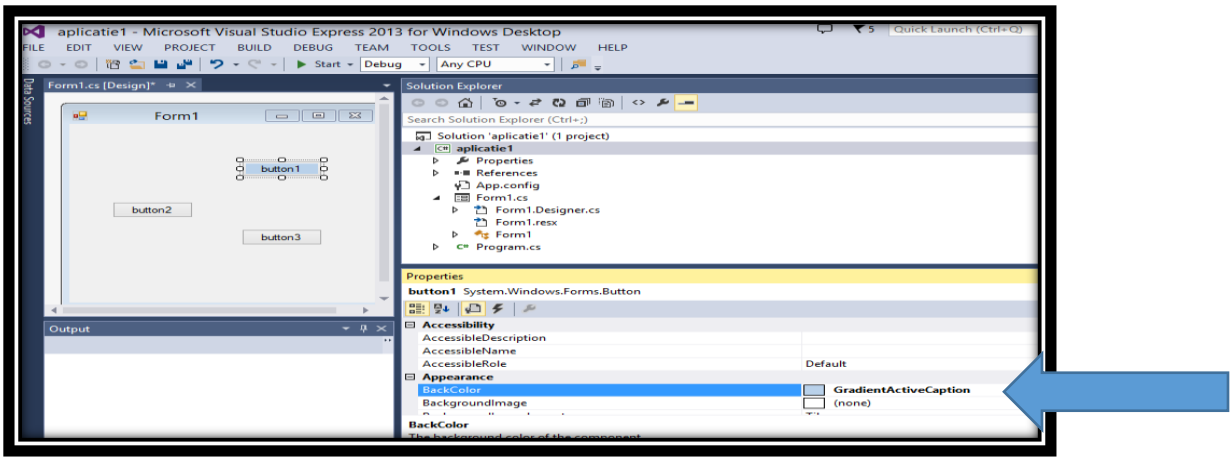
Intr-o fereastra se creeaza trei butoane, colorate initial in verde, cu nule B1, B2, B3. La efectuare click pe B1 acesta isi schimba culoarea in rosu, la parasirea suprafetei butonului acesta revine la galben. La efectuare click pe B2 acesta isi schimba culoarea in galben, la parasirea suprafetei butonului acesta revine la galben. La efectuare click pe B3 acesta isi schimba culoarea in albastru, la parasirea suprafetei butonului acesta revine la galben.

Etape:

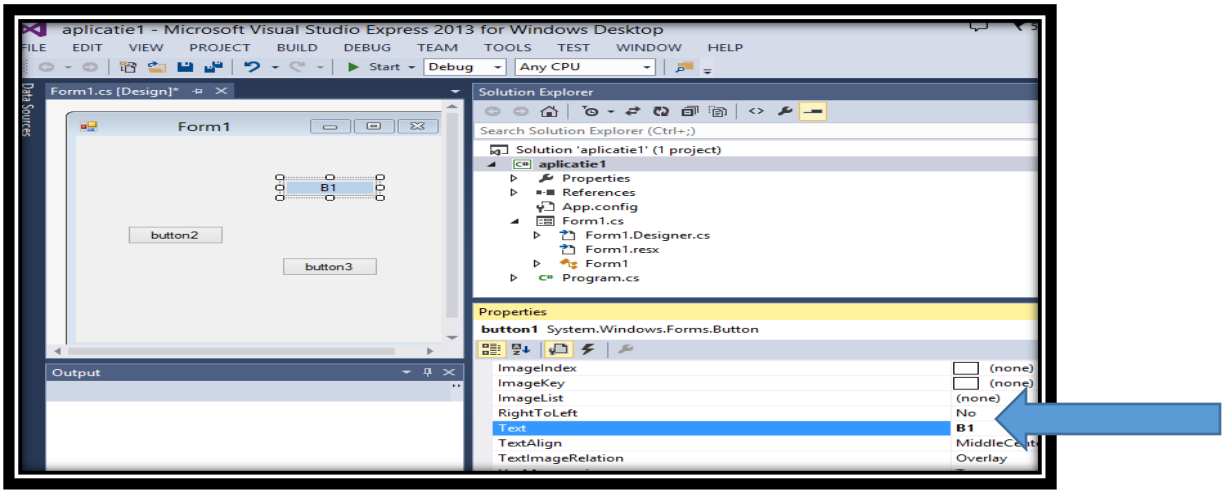
- Deschidem o noua aplicatie *WindowsForms*
- Din *Toolbox* tragem cu mouse-ul pe formular cele trei butoane.
- Pentru fiecare dintre butoane accesam *Properties (*click dreapta pe fiecare*)* pentru a selecta proprietatile dorite.



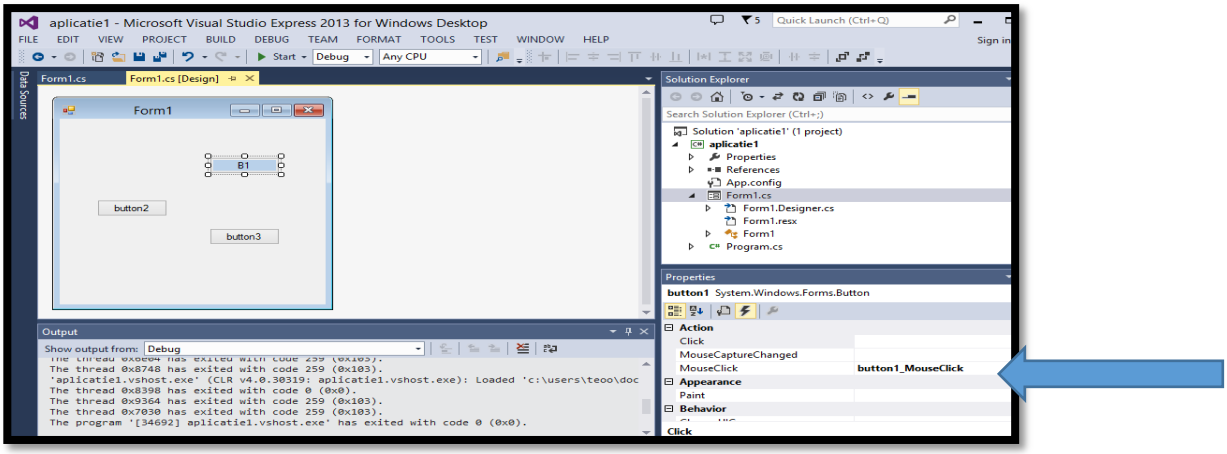
Stabilirea culorii:



Stabilirea etichetei de pe buton:



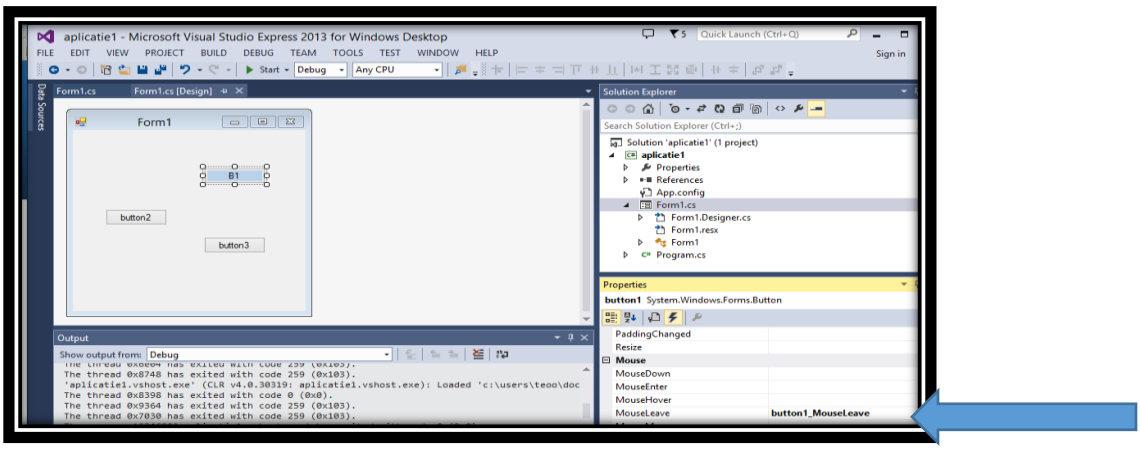
Stabilirea evenimentului pentru schimbarea culorii la efectuare click pe mouse:



Se deschide handler-ul corespunzator pentru introducerea codului:

```
private void button1_MouseClick(object sender, MouseEventArgs e)
      {
          button1.BackColor = Color.Red;}
```

Revenirea la culoarea anterioara dupa parasirea suprafetei butonului:

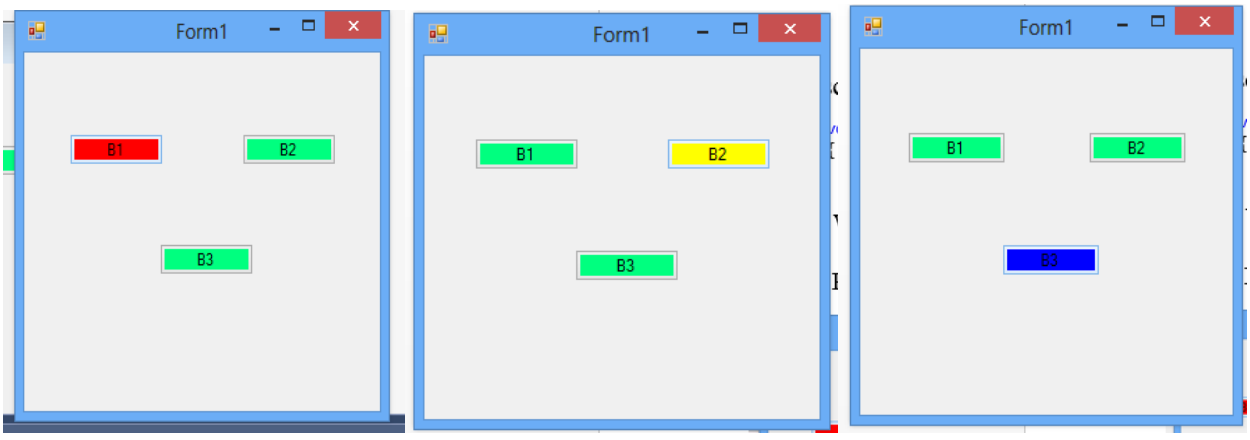


Se deschide handler-ul corespunzator pentru introducerea codului:

```
private void button1_MouseLeave(object sender, EventArgs e)
      {
          button1.BackColor = Color.Blue;
       }
```

La fel vom proceda si pentru celelalte doua butoane.

- Pentru vizualizarea rezultatului(compilare si rulare) apasa F5 si rezultatul:



4.2.2 Controalele Label si LinkLabel

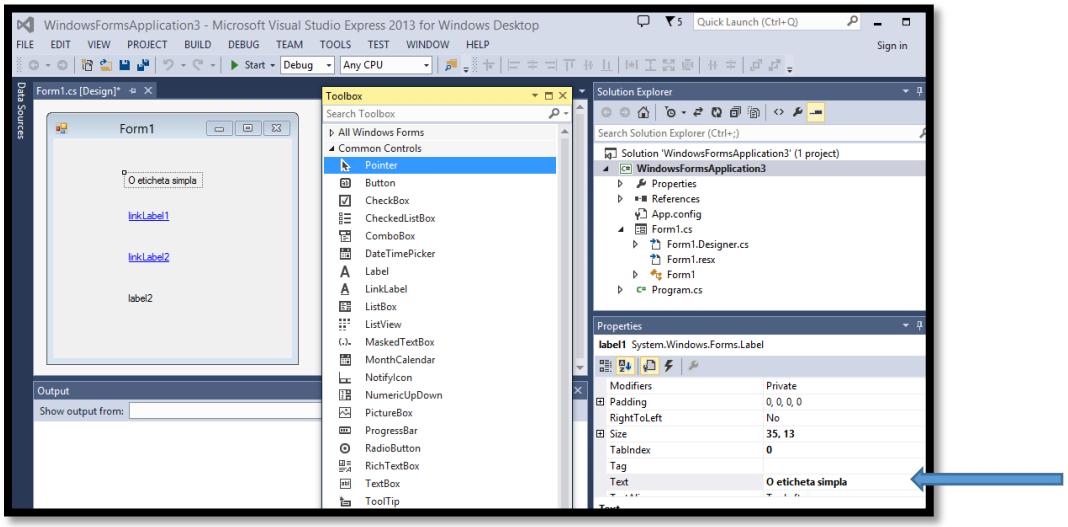
Sunt folosite cu scopul de afisare a informatiilor.

Exemplu: O fereastra cu o eticheta simpla, o eticheta de tip ancora catre site-ul www.yahoo.com si o eticheta mixta (simpla+ancora) catre www.google.com . La accesarea ancorei catre www.google.com se va face vizibila o alta eticheta initial ascunsa, cu mesajul „Tocmai ati accesat site-ul”.

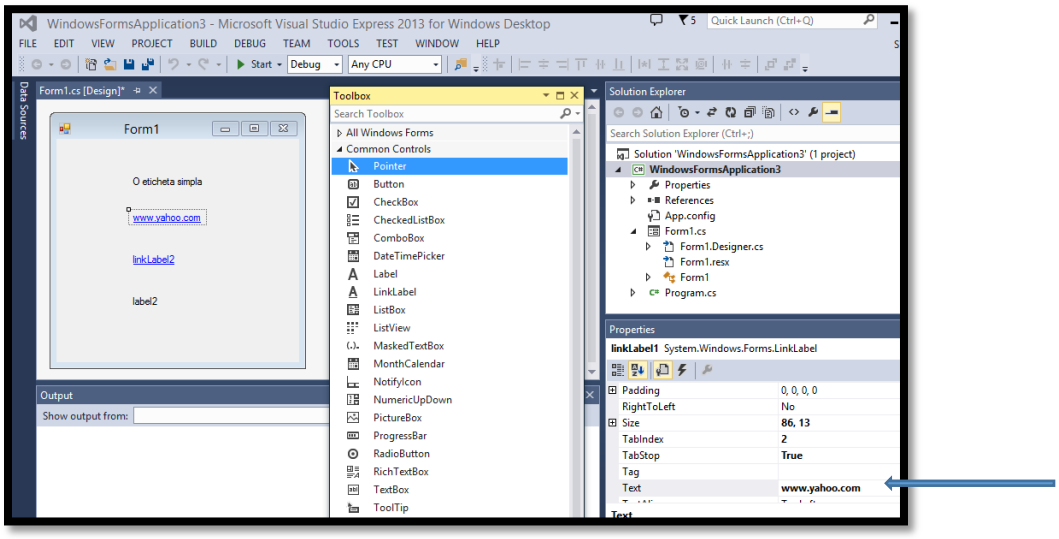
Etape:

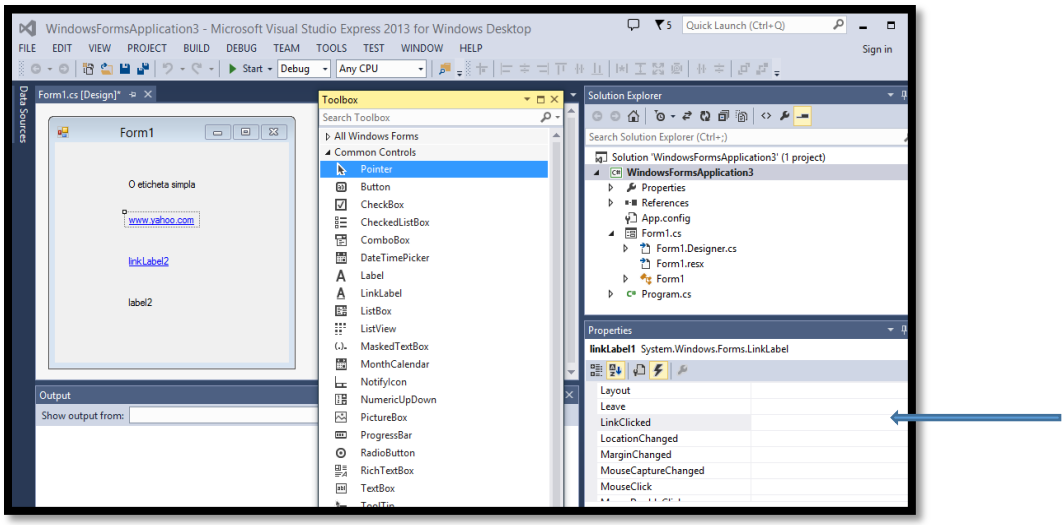
- Deschidem o noua aplicatie *WindowsForms*
- Din *Toolbox* selectam cele 2 *Label* si cele *2 LinkLabel* de care avem nevoie
- Pentru fiecare din etichete satbilim din *Properties* (click dreapta) setarile dorite. Pentru LinkLabel selctam si evenimetele.

Pentru prima eticheta simpla:



Pentru prima eticheta LinkLabel setam Text si evenimentul LinkClicked. In cadrul metodei care se deschide vom introduce codul corespunzator deschiderii www.yahoo.com.

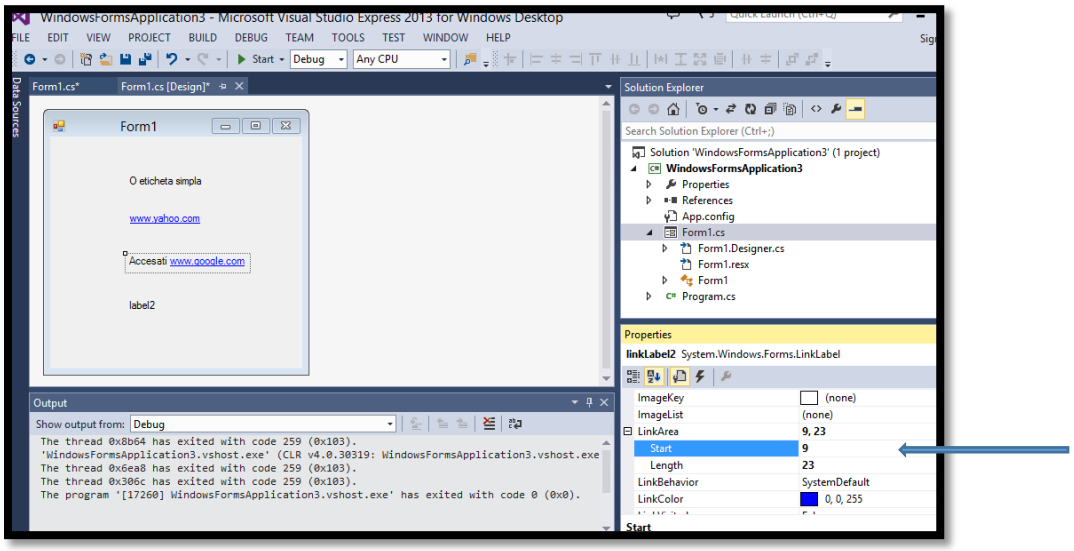


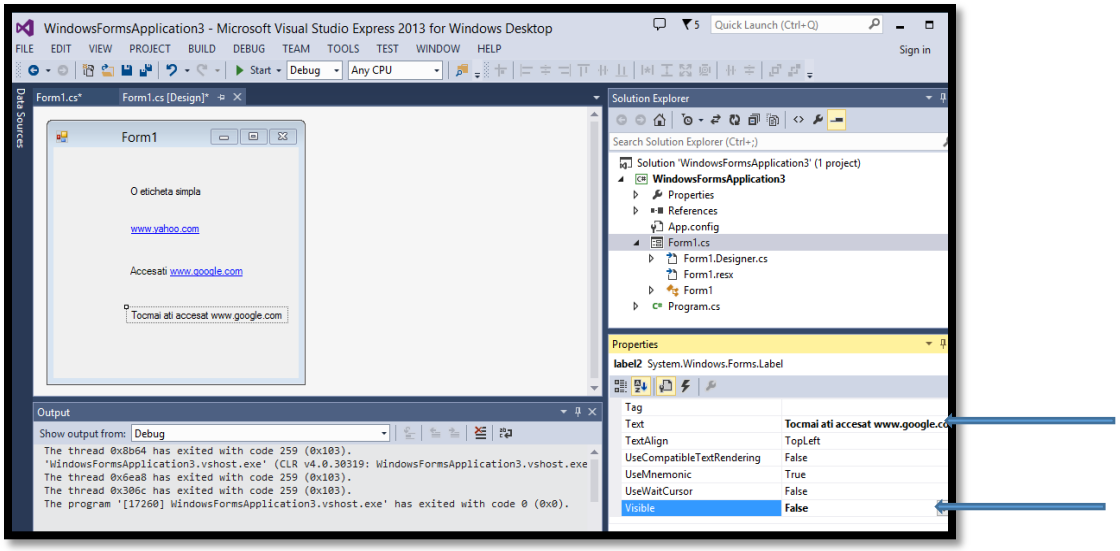



```
private void linkLabel1_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)
        {
            linkLabel1.LinkVisited = true;//schimba culaore link la vizitare
            System.Diagnostics.Process.Start("http://www.yahoo.com");//apelam metoda
pentru
            //deschidera site-ulul
        }
```

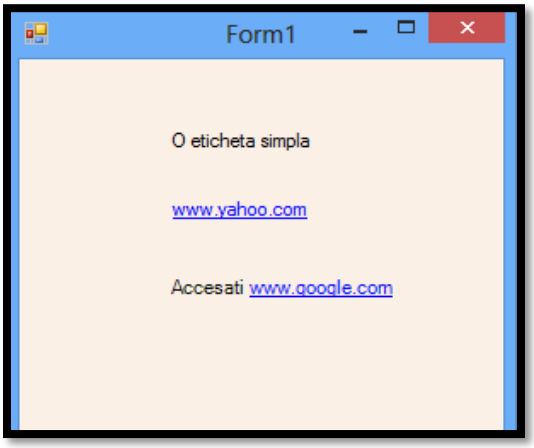
Pentru eticheta combinata pe langa introducerea Text, eveniment setam LinkArea sa inceapa cu pozitia 9.

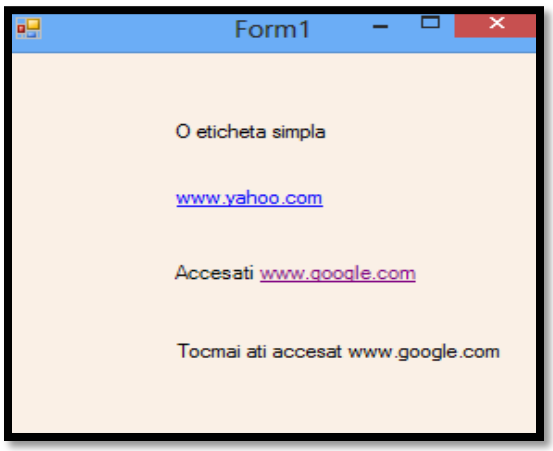
```
private void linkLabel2_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)
        {
            linkLabel2.LinkVisited = true;//schimba culaore link la vizitare
            System.Diagnostics.Process.Start("http://www.google.com");//apelam metoda
pentru
            //deschidera site-ulul
            label2.Visible = true;//devine vizibila eticheta a doua simpla;
        }
```


WindowsFormsApplication3 - Microsoft Visual Studio Express 2013 for Windows Desktop
Form1
O eticheta simpla
www.yahoo.com
Accesati www.google.com
label2
Solution Explorer
Properties
linkLabel2 System.Windows.Forms.LinkLabel
Start
Output


WindowsFormsApplication3 - Microsoft Visual Studio Express 2013 for Windows Desktop
Form1
O eticheta simpla
www.yahoo.com
Accesati www.google.com
Tocmai ati accesat www.google.com
Solution Explorer
Properties
label2 System.Windows.Forms.Label
Output

❖ La rulare:

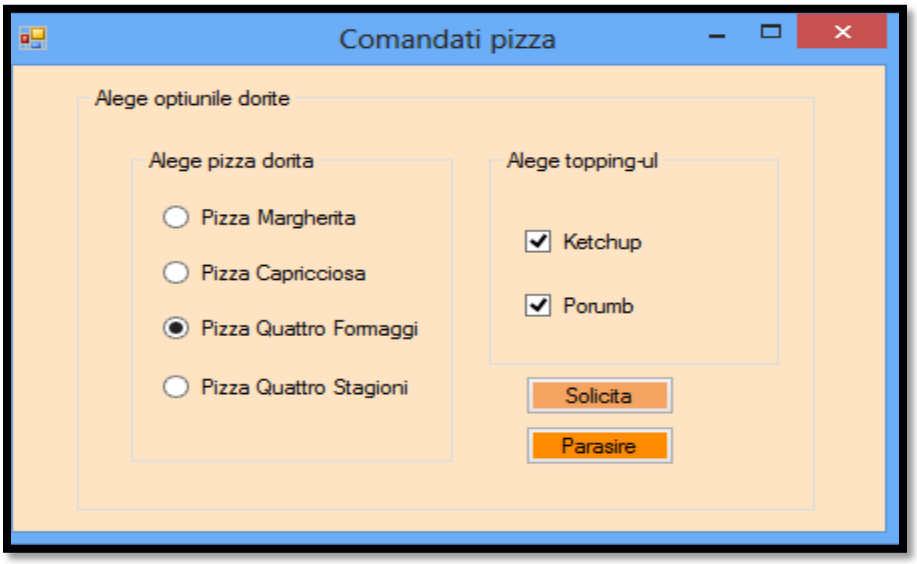

Form1
O eticheta simpla
www.yahoo.com
Accesati www.google.com


Form1
O eticheta simpla
www.yahoo.com
Accesati www.google.com
Tocmai ati accesat www.google.com

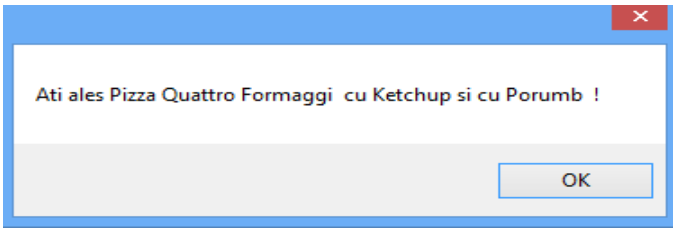
4.2.3 Controalele RadioButton, CheckBox si GroupBox

Deriva din clasa *Button*. Butoanele radio permit alegerea unei singure optiuni din mai multe, motiv pentru care ele se grupeaza obligatoriu in *container*.Butoanele de validare permit alegerea uneia sau mai multor optiuni (pot fi grupate sau nu in *container*).

Pentru exemplificarea folosiri lor realizam fereaastra de forma:

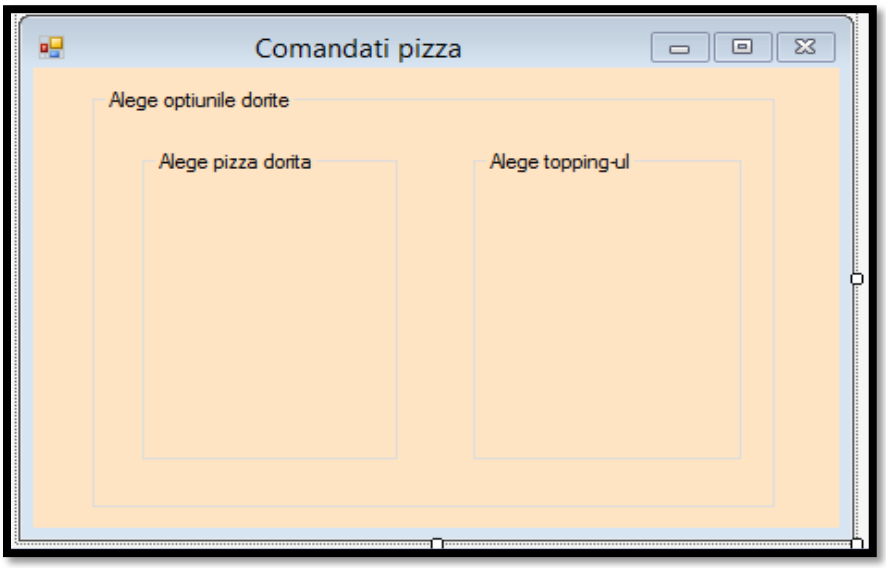


La selectia unui tip de pizza si unui toping adecvat sau amandoua si in urma apasari butonului cu eticheta „Selectie” ne v-a afisa mesajul:

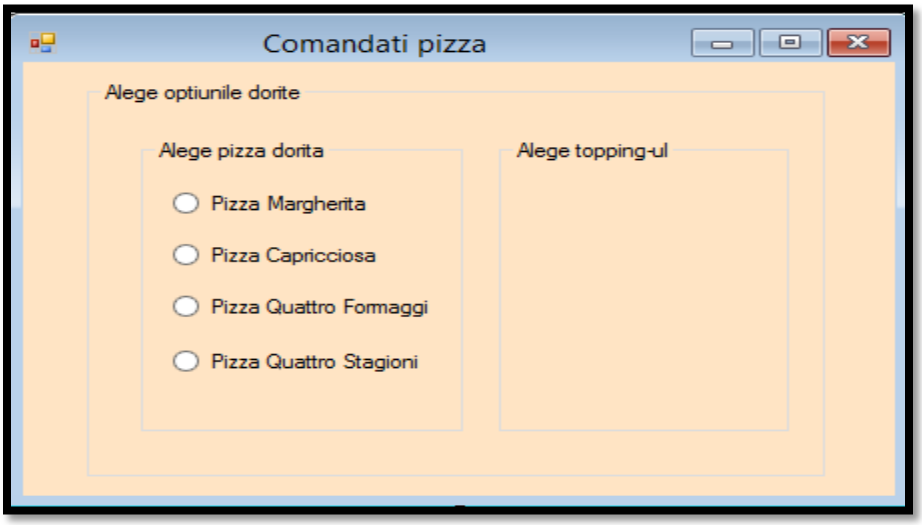


Etape:

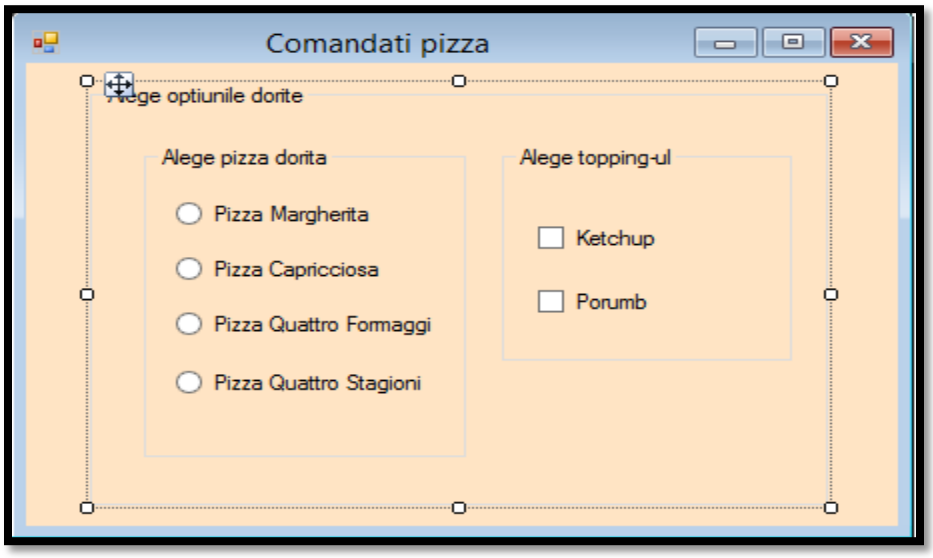
- Deschidem o noua aplicatie *WindowsForms*
- In formularul curent selectam din *Toolbox*, sectiunea *Containers* trei controale de tip *GroupBox*, pentru care stabilim din Properties campul Text denumirile dorite (*Alegeti optiunile dorite*, *Alegeti pizza*, *Alegeti topping*)



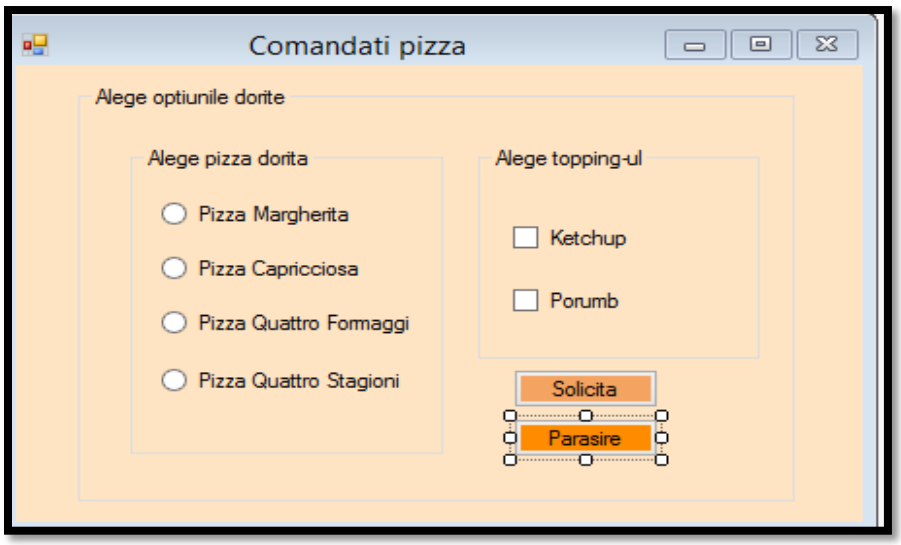
- In interiorul *GroupBox*-ului etichetat „*Alege pizza dorita*”, din *Toolbox* selectam 4 butoane radio pe care le etchetam din *Properties* → *Text* cu denumiri de pizza.



- In interiorul *GroupBox*-ului etichetat „*Alege topping-ul*”, din *Toolbox* selectam 2 butoane de verificare pe care le etchetam din *Properties* → *Text* cu denumiri de topping.



- Din Toolbox selectam doua controale de tip Button. Din Properties→Text le stabilim etichetarea „Solicita” si „Parasire”. Butonul „Solicita ” trebuie sa raspunda evenimentului ClickMouse prin afisarea unui mesaj cu optiunile dorite (selectate).



- Din acest motiv in Properties→ Events la evenimentul respectiv introducem codul:

```
private void button1_MouseClick(object sender, MouseEventArgs e)
  {
      string m = "Ati ales ";
```

```
        if (r1.Checked)
            m = m + r1.Text;
        if (r2.Checked)
            m = m + r2.Text;
        if (r3.Checked)
            m = m + r3.Text;
        if (r4.Checked)
            m = m + r4.Text;
        m = m + " cu ";
        int ok = 0;
        if (c1.Checked)
        {
            m = m + c1.Text;
            ok = 1;
        }
        if (c2.Checked)
            if (ok == 1)
                m = m + " si cu " + c2.Text;
            else
                m = m + c2.Text;
        MessageBox.Show(m + "  !");
    }
    }
```

- Pentru a parasi formularul de comanda si aplicatia in sine, asociem butonului „Parasire”, la evenimentul ClickMouse codul:

```
private void b2_MouseClick(object sender, MouseEventArgs e)
{
    Application.Exit();
}
```

4.2.4 Controlul TextBox si RichTextBox

Controlul TextBox este folosit pentru introducerea de date de dimensiuni mici.

Dintrele numeroasele proprietati pe care le detine amintim:

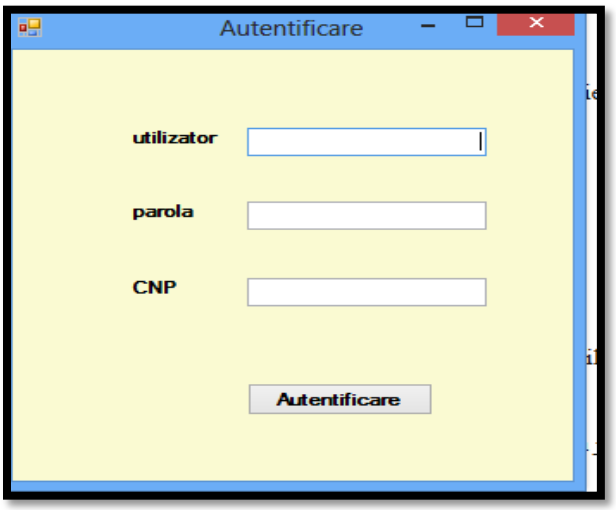
- Text – pentru editarea textului din control
- PasswordChar – folosita pe post de masca pentru parola
- Multiline – seteaza control-ul pe mai multe linii

Dintre evenimente:

- Click – se declanseaza la apasarea click pe control
- GetFocus – se declanseaza la accesarea control-ului
- Leave – se declanseaza la parasirea control-ului

Exemplificare a folosiri *TextBox*

Dorim sa construim un formular de identificare de forma:

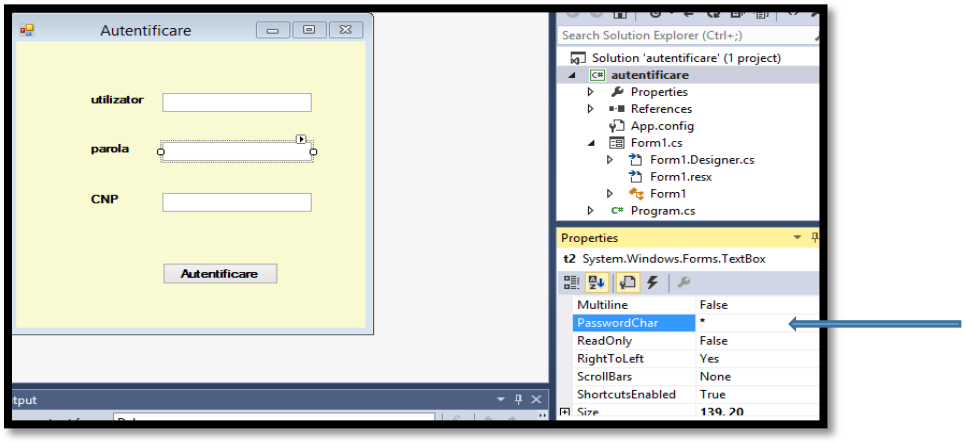


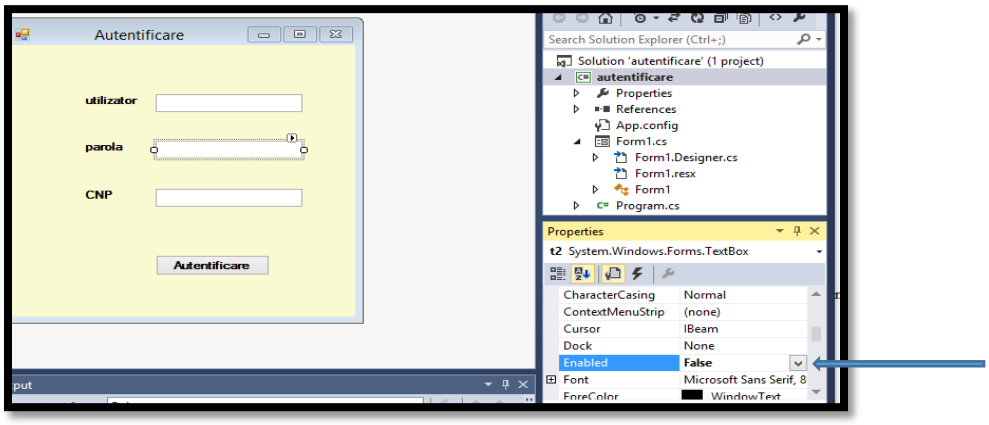
Contine trei controale de tip *TextBox* si un contro de tip *Button*. Campul etichetat „parola” este mascat cu „*” si initial inactiv. Campul CNP si butonul „Autentificare” sunt setate initial pe invizibil. La introducerea utilizatorului corect se activeaza campul etichetat „parola”. La introducerea unei parole corecte devin vizibile controalele: CNP si „Solicita”. Butonul va avea asociat la evenimentul click o rutina de verificare a corectitudini introduceri CNP-ului, in cazul unei operatii corecte se va afisa mesajul „Autentificare reusita”, in caz contrar „Autentificare nereusita”.

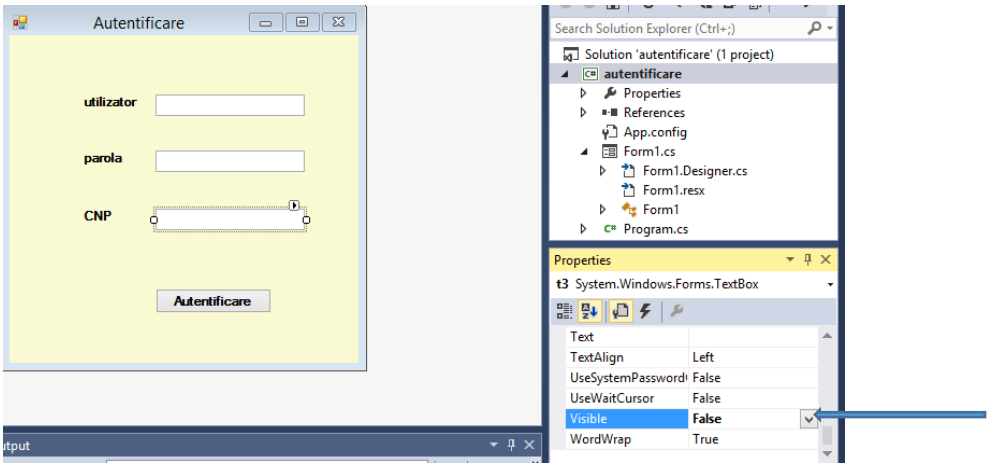
Etape:

- Deschidem o noua aplicatie *WindowsForms*
- In formularul curent din Toolbox selectam de trei ori control-ul *TextBox*, pentru ficare dintre ele un *Label* adecvat si un control de tip *Button*
- Stabilim din *Properties* pentru fiecare eticheta textul dorit, pentru controalele de tip *TextBox* completam proprietatea *Name* pentru editarea codului intr-un mod mai usor.

❖ Pentru a seta campul „parola” de tip masca „*”, din *Properties* selectam *PasswordChar* de tip „*”, *Enabled* de tip fals. Pentru campuri invizibile selectam proprietatea *Visible* pe valoarea de fals.

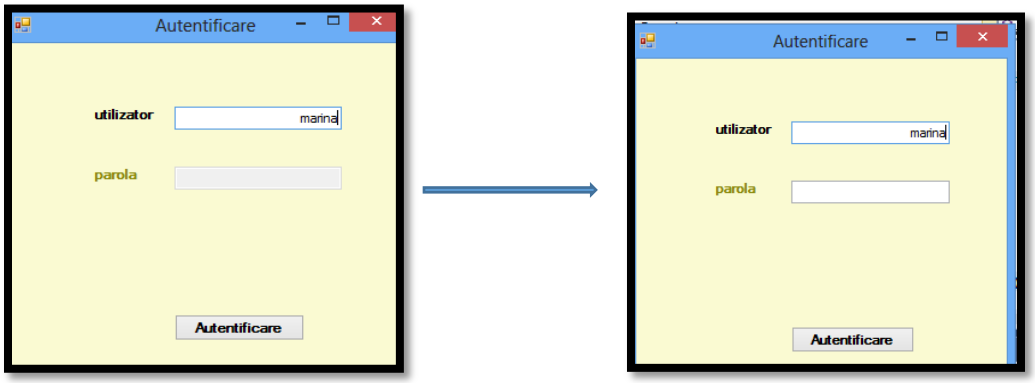






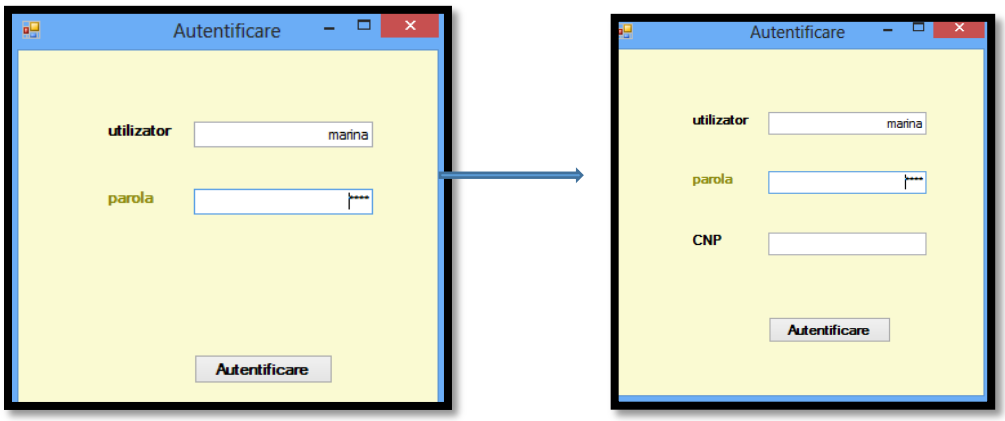
❖ Pentru aputea verifica introducerea utilizatorului corect putem asocia control-ului *t1*(numele stabilit pentru primul control) evenimentului *PreviewKeyDown*.

```
  private void t1_PreviewKeyDown(object sender, PreviewKeyDownEventArgs e)
{
    if (e.KeyCode == Keys.Tab)
        if (t1.Text == "marina")
            t2.Enabled = true;
        else
            MessageBox.Show("user incorect");

            }
```



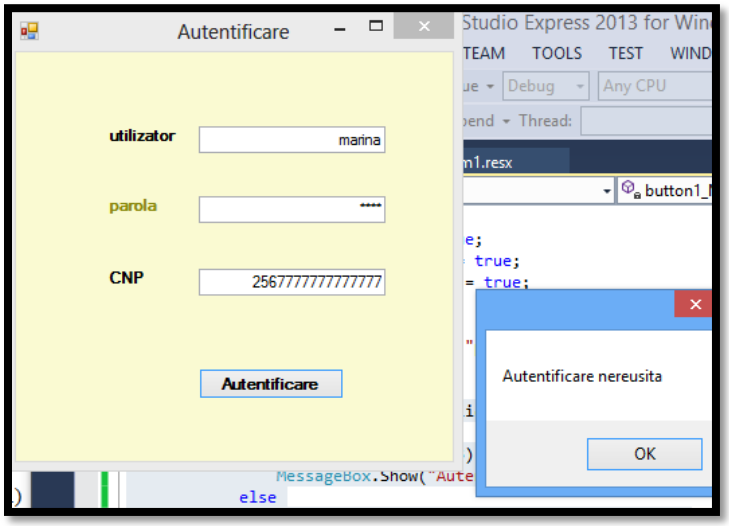
❖ Pentru verificarea parolei corecte din control-ul TextBox t2, asociam evenimentul *PreviewKeyDown* cu urmatorul cod editat.

```
    private void t2_PreviewKeyDown(object sender, PreviewKeyDownEventArgs e)
{

    if (e.KeyCode == Keys.Enter)
        if (t2.Text == "1234")
        {
            t3.Visible = true;
            label3.Visible = true;
            button1.Visible = true;
        }
        else
            MessageBox.Show("parola incorecta");
              }
```



❖ Control-ului de tip *Button* ii asociem evenimentul *MouseClick* cu urmatorul cod:

```
  private void button1_MouseClick(object sender, MouseEventArgs e)
{
    if (t3.Text.Length == 13)
        MessageBox.Show("Autentificare reusita");
    else
        MessageBox.Show("Autentificare nereusita");
            }
```



5. Invatarea prin Metoda Proiectului in Limbajul C#

5.1 Notiuni generale despre Metoda Proiectului

Învăţarea bazată pe proiecte este un model de instruire centrat pe elev. Acest tip de învăţare dezvoltă cunoştinţe şi capacităţi în domeniul informaticii prin sarcini de lucru extensive, care promovează investigaţia şi demonstraţiile autentice ale învăţării prin rezultate şi performanţe. Educaţia prin metoda proiectului este orientată de întrebări cheie ale curriculumului care fac legătura între standardele de performanţă (obiective de referinţă şi competenţe specifice), capacităţile cognitive de nivel superior ale elevilor şi contexte din viaţa reală. Unităţile de învăţare care utilizează metoda proiectului includ strategii de instruire variate, menite să îi implice pe elevi indiferent de stilul lor de învăţare. Disciplinele informatice, prin esenţa lor, conduc spre dezvoltarea de proiecte, ca o finalitate a procesului de predare-învăţare. Tehnologia este utilizată tot pentru a sprijini învăţarea şi documentarea în realizarea produsului finit. Pe întreg parcursul desfăşurării proiectului, sunt incluse diferite metode de evaluare pentru a asigura calitatea activităţilor de învăţare.

Proiectul are obiective operaţionale clare, care sunt în conformitate cu standardele de performanţă (obiectivele de referinţă şi competenţele specifice) şi se concentrează pe ceea ce trebuie să ştie elevii ca rezultat al activităţilor de învăţare. Concentrându-se pe obiective, profesorul defineşte în planul de evaluare modalităţile corespunzătoare prin care elevii demonstrează ceea ce au învăţat şi organizează activităţile de învăţare şi procesul de instruire. Activităţile proiectului au drept rezultat produsele elevilor şi performanţe legate de sarcini realizate de aceştia, precum prezentările convingătoare, care demonstrează că au înţeles obiectivele operaţionale şi standardele de performanţă.

Introducerea unei unităţi de învăţare bazate pe un proiect se realizează prin intermediul unor întrebări care exprimă idei importante şi durabile, cu un caracter transdisciplinar. Elevii sunt provocaţi să cerceteze mai în profunzime subiectul cu ajutorul întrebărilor de conţinut, care se concentrează pe obiectivele operaţionale şi pe standarde de performanţă. Există trei tipuri de întrebări cheie ale curriculumului: esenţiale, specifice unităţii de învăţare şi specifice conţinuturilor. Întrebările esenţiale au un caracter general şi sunt întrebări deschise care abordează idei importante şi concepte durabile pe care oamenii se străduiesc să le înţeleagă. Acestea depăşesc de multe ori graniţa unei singure discipline şi îi ajută pe elevi să vadă legătura dintre subiecte. Întrebările unităţii sunt direct legate de proiect şi sprijină investigaţiile cu privire la întrebarea esenţială. Acestea ajută la demonstrarea înţelegerii de către elevi a conceptelor de bază ale proiectului. Întrebările de conţinut au mai mult un caracter factual şi sunt conforme standardelor de performanţă.

Proiectele au relevanţă pentru viaţa elevilor şi pot implica reprezentanţi ai comunităţii sau experţi din exterior, care asigură un context pentru învăţare.

Cu ajutorul tehnologiei, elevii au un control mai mare asupra produselor finale, precum şi posibilitatea de a personaliza aceste produse. Elevii pot depăşi limitele sălii de clasă colaborând cu alţi elevi aflaţi la distanţă prin intermediul email-ului sau al propriilor site-uri sau prezentându-şi rezultatele învăţării cu ajutorul instrumentelor multimedia. Activităţile proiectului sprijină dezvoltarea atât a capacităţilor cognitive, cât şi a celor metacognitive, precum colaborarea, auto-monitorizarea, analiza datelor sau evaluarea informaţiilor. Pe parcursul proiectului, întrebările cheie ale curriculumului îi provoacă pe elevi să gândească şi să facă legătura cu concepte care contează în lumea reală.

Organizarea activităţilor de realizare a proiectelor presupune din partea profesorului următoarele activităţi:

- Stabilirea titlului: profesorul poate să decidă tema proiectului sau poate să permită elevilor să o facă
- Stabilirea grupelor de lucru: se va face de către profesor după consultarea prealabilă a elevilor
- Stabilirea timpului de lucru: profesorul trebuie să proiecteze atât timpul alocat elevilor pentru realizarea proiectului cât şi timpul pentru prezentarea şi evaluarea proiectelor
- Stabilirea obiectivelor şi a competenţelor vizate
- Ghidarea activităţii: presupune îndrumarea elevilor cu privire la rolul şi sarcinile de lucru ale fiecăruia, indicaţii la părţile pe care elevii nu ştiu să le dezvolte, indicarea de bibliografie suplimentară
- Evaluarea: profesorul decide criteriile după care vor fi evaluaţi elevii

Avantajele învăţării prin metoda proiectului sunt:

- Încurajarea spiritului investigativ şi a gândirii de nivel superior (Thomas, 1998)
- O participare mai bună, sporirea încrederii în sine şi ameliorarea atitudinii cu privire la învăţare (Thomas, 2000)
- Achiziţiile în domeniile cunoaşterii sunt egale sau mai bune decât cele generate de alte metode, iar elevii implicaţi în proiecte îşi asumă o responsabilitate mai mare în ceea ce priveşte propriul studiu decât pe parcursul activităţilor didactice tradiţionale (Boaler, 1999; SRI, 2000)
- Oportunităţi de formare a unor competenţe complexe, cum ar fi capacităţi de gândire de nivel superior, rezolvare de probleme, abilităţi de colaborare şi competenţe de comunicare (SRI)

- Accesul la o gamă mai largă de oportunităţi de învăţare în clasă, constituind o strategie de implicare a elevilor care provin din diverse medii culturale (Railsback, 2002).
- Învăţarea prin metoda proiectului este un model de instruire care implică elevii în investigarea unor probleme captivante. Proiectele care oferă mai multe oportunităţi de învăţare pot fi semnificativ diferite în ceea ce priveşte aria tematică sau scopul şi pot fi aplicate la clase diferite şi la mai multe niveluri de studiu. Proiectele angajează elevii în roluri active, cum ar fi: luarea deciziei, investigare; documentare.
- Proiectele servesc obiective operaţionale specifice, semnificative. Proiectele nu reprezintă abateri de la programa şcolară, activităţi suplimentare sau activităţi cu o temă comună. Curriculumul prin proiecte este orientat de întrebări importante care leagă obiectivele operaţionale şi gândirea de nivel superior a elevilor cu viaţa de fiecare zi. Elevii îşi asumă deseori roluri din viaţa reală şi trebuie să îndeplinească sarcini pline de semnificaţie.
- În timp ce lucrează la proiecte, elevii îşi dezvoltă competenţe pentru lumea reală, corespunzătoare secolului XXI - multe din acestea fiind solicitate de angajatorii din zilele noastre - cum ar fi capacitatea de a: colabora; lua decizii; avea iniţiativă; rezolva probleme complexe; comunică eficient.
- Leagă cunoştinţele disciplinare, de viaţa de fiecare zi a elevilor ;
- Permite abordări monodisciplinare, interdisciplinare, pluridisciplinare şi transdisciplinare;
- Se centrează pe interesele elevilor ;
- Ţine cont de talentele lor ;
- Poate servi nevoilor de învăţare identificate .

Din categoriile de proiecte care pot fi dezvoltate la disciplinele informatice amintim:

- ❖ Complemente de informatică (teme studiate la clasă care pot fi aprofundate sau extinse)
- ❖ Aplicaţii din viaţa cotidiană (baze de date)
- ❖ Probleme interdisciplinare
- ❖ Jocuri
- ❖ Softuri educaţionale
- ❖ Web design

La disciplina Informatică, metoda proiectului poate fi utilizată la toate clasele, indiferent de profil.

Predarea prin această metodă presupune proiectarea unităţii de învăţare, susţinută de elaborarea materialelor auxiliare şi instrumentelor de evaluare care se utilizează în procesul de instruire. Acestea sunt puse la dispoziţia elevului pentru a fi consultate şi utilizate pe tot parcursul învăţării. Profesorul are rol de facilitator, care oferă informaţia necesară elevilor, monitorizează activităţile, evaluează. Elevii sunt implicaţi activ în propria învăţare şi, în urma documentării, vor realiza produse informatice.

Metoda constă în utilizarea unui site de colaborare, în care se creează directoare specifice unităţilor de învăţare sau proiectelor cu temă dată. Profesorul încarcă pe platforma informatică bibliografia, suportul teoretic, cerinţele referitoare la întocmirea proiectului, instrumentele de evaluare. Elevii prelucrează informaţia şi elaborează proiectul. Pe tot parcursul desfăşurării unităţii de învăţare elevii au acces la toate materialele publicate pe site, precum şi la cerinţele descrise de către profesor. Ei postează pe site temele pentru acasă şi instrumentele de evaluare

completate, iar la final, după obţinerea acceptului profesorului, publică proiectul realizat.

În perioada de elaborare a proiectului, profesorul supervizează fiecare etapă, direct şi pe site-ul de colaborare. Materialele postate pot fi accesate de către oricare dintre utilizatorii platformei, ca exemple de bună practică, eficientizând astfel procesul de învăţare.

5.2 Studiu de caz

In continuarea metoda proiectului v-a fi aplicata pentru unitatea de invatare „*Dezvoltarea şi prezentarea unei aplicaţii în mediul vizual*”, la clasa a XII-a , 3 ore pe sptamana.

Porind de la planificarea calendaristica intocmita la inceputul anului scolar, s-a intocmit proiectarea unitatii de invatare respective.

Consideram ca aceasta unitate de invatare corepunde tuturor cerintelor implementarii metodei proiectului, intrucat presupune existenta unei baze solide in ceace priveste limbajul C#, Mediul Visual C# si cunoasterea notiunilor de baze de date adecvate (proiectarea unei baze de date, operatii specifice bazelor de date realizate in SQL).

Implementarea implica utilizarea unui site de colaborare, in care documentele ce vin in sprijinul realizarii proiectului vor fi incarcate de profesor, cu scopul de a fi accesibile oricand elevilor.

Comunicarea intre elevi si profesori, elevi-elevi, se va face fie frontal sau prin intermediul site-ului de colaborare, site-urilor de socializare, e-mail.

5.2.1 Planificarea calendaristica si proiectarea unitatii de invatare

Planificarea calendaristica este prezentata mai jos impreuna cu proiectarea unitatii de invatare.

Planificare calendaristică

Unitatea şcolară..... Profesor:
.....

Disciplina INFORMATICĂ Clasa a XII-a , 3 ore/săpt.

Planificare calendaristică

Programare orientată pe obiecte şi programare vizuală

Anul şcolar ……

Nr. Crt.	Unităţi de învăţare	Competenţe specifice	Conţinuturi	Nr. Ore	Săptămâna	Observaţii
1.	Programarea orientată pe obiecte	1.1. Analizarea unei probleme în scopul identificării şi clasificării datelor necesare 1.2. Identificarea relaţiilor dintre date 1.3. Identificarea modalităţilor adecvate de structurare a datelor care intervin intr-o problema 1.4. Utilizarea funcţiilor specifice de prelucrare a datelor structurate 3.1. Utilizarea instrumentelor de dezvoltare a unei aplicaţii	❖ Principiile programării orientate pe obiecte ❖ Structura unei aplicaţii orientată pe obiecte ❖ Clase şi obiecte ❖ Clase şi funcţii şablon ❖ Derivarea claselor ❖ Tratarea erorilor ❖ Polimorfism **Evaluare sumativă**	42 3	S1-S15	

Nr. Crt.	Unităţi de învăţare	Competenţe specifice	Conţinuturi	Nr. Ore	Săptămâna	Observaţii
2.	Programare vizuală	1.1 Analizarea unei probleme în scopul identificării şi clasificării datelor necesare 2.1. Identificarea tehnicilor de programare adecvate rezolvării unei probleme şi aplicarea creativă a acestora 2.2. Elaborarea strategiei de rezolvare a unei probleme 3.1. Utilizarea instrumentelor de dezvoltare a unei aplicaţii 3.2. Elaborarea şi realizarea unei aplicaţii folosind un mediu de programare specific	❖ Concepte de bază ale programării vizuale ❖ Prezentarea mediului de programare vizual ❖ Elemente de POO în context vizual ❖ Construirea interfeţei utilizator ❖ Accesarea şi prelucrarea datelor **Evaluarea sumativă**	42 3	S16-S30	

Nr. Crt.	Unităţi de învăţare	Competenţe specifice	Conţinuturi	Nr. Ore	Săptămâna	Observaţii
3.	Dezvoltarea şi prezentarea unei aplicaţii în mediul vizual	2.1. Identificarea tehnicilor de programare adecvate rezolvării unei probleme şi aplicarea creativă a acestora 2.2. Elaborarea strategiei de rezolvare a unei probleme 2.3. Analizarea comparativă a eficienţei diferitelor tehnici de rezolvare a problemei respective şi alegerea celei mai eficiente variante 3.1. Utilizarea instrumentelor de dezvoltare a unei aplicaţii 3.2. Elaborarea şi realizarea unei aplicaţii folosind un mediu de programare specific 3.3. Prezentarea unei aplicaţii	❖ Ciclul de viaţă al unui produs software (alegerea temei, definirea cerinţelor utilizator, modelarea soluţiei, implementarea şi testarea produsului) ❖ Modele arhitecturale ❖ Realizarea şi documentarea proiectului ❖ Prezentarea proiectului **Evaluarea sumativă**	15 3	S31-S36	

b) Proiectarea unităţii de învăţare

Liceul ..... Profesor…….

INFORMATICĂ Clasa a XII-a , 3 ore/săptămână

Proiectul unităţii de învăţare

An şcolar ……

Dezvoltarea si prezentarea unei aplicatii in mediul visual (18 ore)

Nr. Crt.	Conţinuturi	C.S.	Activităţi de învăţare	Resurse	Evaluare	Obs.
1	**Ciclul de viaţă al unui produs software (alegerea temei, definirea cerinţelor utilizator, modelarea soluţiei, implementarea şi testarea produsului)**	2.1 2.2 3.1	- Prezentarea unor situatii reale de utilizarea a bazelor de date - Formularea unor teme concrete care sa rezolve cerinte reale - Conectarea la BD in Visual studio C# 2013 - Operatii asupra bazelor de date	Expunerea Conversaţia Activitate frontală Manualul scolar Activitate frontala	Observarea sistematică Aplicatie practica	
2	**Modele arhitecturale**	2.3 2.2	- Conceperea ERD-ului asociat BD - Optimizarea modelului relational astfel incat sa corespunda cerintelor din viata reala. - Stabilirea cerintelor functionale si analiza acestora - Stabilirea constrangerilor de dezvoltare si temporale	Conversatie Modelare Activitate frontala	Investigaţia Observarea sistematică Aplicatie practica	

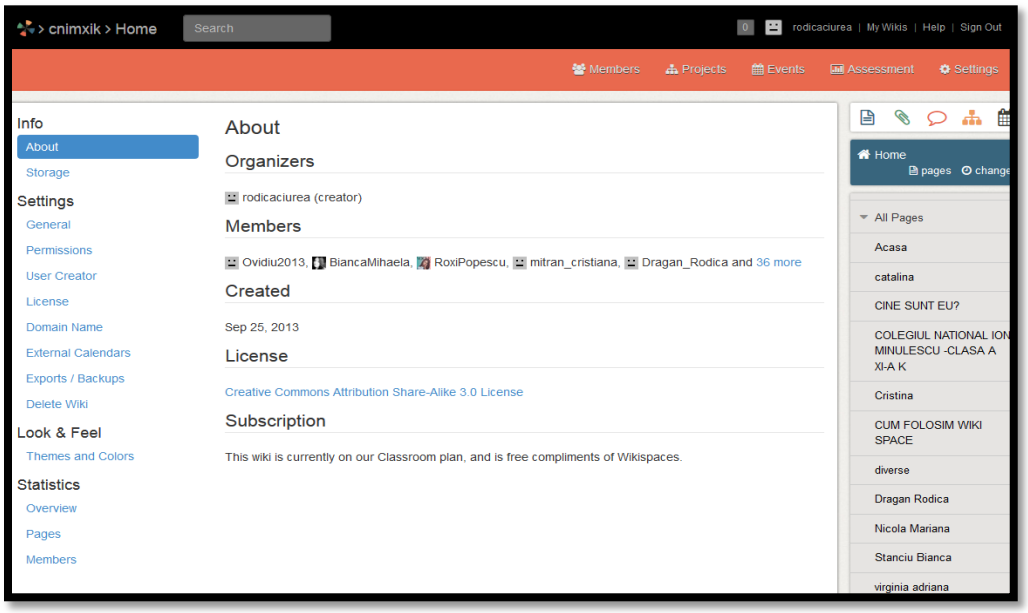
3	**Realizarea şi documentarea proiectului**	3.1 3.2	- proiectarea operatiilor de baza - realizarea interfetei - strangerea datelor din viata reala - testarea operatiilor implementate	Site-ul de colaborare Biblioteci virtuale Conversatia Fise de activitati practice	Investigaţia Observarea sistematică Aplicatie practica	
4	**Prezentarea proiectului**	3.3	- strangerea materialelor si informatiilor - stabilire structura prezentare - concluzii asupa obiectivelor principale ce trebuie atinse	Activitate frontala Conversatia Site-ul de colaborare	Observarea sistematică Investigaţia Aplicaţia practică	

5.2.3 Site-ul de colaborare

Inainte de demararea procesului de instruire, profesorul care va avea rol de facilitator pe parcursul procesului va construi, starange materialele si resursele adecvate.

Drept site de colaborare s-a ales varianta wiki. Wiki-urile reprezintă site-uri Web de colaborare online care pot fi setate pentru a fi editate de oricine sau doar de anumiţi utilizatori. Autorul unui wiki poate fi notificat cu privire la toate modificările şi poate urmări şi monitoriza dezvoltarea conţinutului site-ului.

Elevii au fos invitati sa colaboreze la acest site. Dupa acceptarea lor ca parteneri, au fost instruiti cu privire la utilizarea facilitatilor de care dispune.



S-au discutat problemeale vietii cotidiente care utilizeaza baze de date. Se pune elevilor întrebarea esenţială:„De ce trebuie implementata o aplicatie care sa gestioneze bazele de date?’’ În urma brainstormingului realizat, profesorul va afla ce cunoştinţe au elevii despre bazele de date.

Pe baza acestor discutii s-a stabilit de comun acord o lista cu teme ce modeleaza bazele de date utilizate in viata reala. Lista intocmita a fost postata pe site-ul de colaborare.

Pe baza discutiilor cu toata clasa asupra preferintelor in ceace priveste tema ce va fi aleasa se vor stabili grupurile de lucru. In alegerea temei un rol important il va avea si posibilitatea fiecarui elev de a interactiona in mod direct cu domeniul din care face parte problema propusa spre rezolvarea. Echipa proiectului trebuie sa fie echilibrata, construita pe baza abilitatilor si aptitudinilor fiecarui membru.

??

5.2.4. Temele propuse pentru proiect

Teme de proiecte propuse	
❖	Se considera baza de date cu studenti formata din urmatoarele tabele: Student, Adresa si Note. Tabelul Student contine urmatoarele campuri: Nr_Leg, Nume, Prenume, Sex, CNP. Tabelul Adresa contine campurile: Id_Adresa, CNP, Localitate, Strada, Judet. Tabelul Note contine campurile: CNP, Materie, Data, Nota
❖	Se considera baza de date cu, cartile dintr-o biblioteca formata din urmatoarele tabele: Carte, Domeniu si Imprumuturi. Tabelul Carte contine urmatoarele campuri: Nr_Inreg, Autor, Titlu, An_aparitie, Editura, Id_Domeniu. Tabelul Domeniu contine campurile: Id_Domeniu, Descriere. Tabelul Imprumuturi contine campurile: Nr_Inreg, Nume, Data_Imprumut, Data_Returnat, Adresa

❖	Se considera baza de date cu angajati formata din urmatoarele tabele: Angajat, Adresa si Sectie. Tabelul Angajat contine urmatoarele campuri: Nr_Leg, Nume, Prenume, Sex, CNP, Id_Sectie. Tabelul Adresa contine campurile: Id_Adresa, CNP, Localitate, Strada, Judet. Tabelul Sectie contine campurile: Id_Sectie, Denumire, Sef_Sectie
❖	Se considera baza de date cu abonati formata din urmatoarele tabele: Abonat, Adresa si Factura. Tabelul Abonat contine urmatoarele campuri: CNP, Nume, Prenume, Sex. Tabelul Adresa contine campurile: Id_Adresa, CNP, Localitate, Strada, Judet. Tabelul Factura contine campurile: Nr_Factura, Data, CNP, Suma
❖	Se considera baza de date cu produse formata din urmatoarele tabele: Produs, Detalii_Produs si Unitate_Masura. Tabelul Produs contine urmatoarele campuri: Id_Produs, Denumire, Firma_Producatoare. Tabelul Detalii_Produs contine campurile: Id_Detaliu, Id_Produs, Data_Fabricatie, Data_Expirare, Id_UM, Pret. Tabelul UM contine campurile: Id_UM, Descriere
❖	Se considera baza de date cu fotbalisti formata din urmatoarele tabele: Fotbalist, Pozitie si Echipa. Tabelul Fotbalist contine urmatoarele campuri: CNP, Nume, Prenume, Data_Nastere, Id_pozitie, Id_Echipa. Tabelul Pozitie contine campurile: Id_Pozitie, Descriere_Pozitie, Laterala. Tabelul Echipa contine campurile: Id_Echipa, Denumire, Adresa, Patron
❖	Se considera baza de date cu divizii de fotbal formata din urmatoarele tabele: Divizie, Clasament si Echipa. Tabelul Divizie contine urmatoarele campuri: Id_Divizie, Descriere, Nr_Echipe. Tabelul Clasament contine campurile: Id_Clasament, Id_Divizie, Pozitie_Clasament. Tabelul Echipa contine campurile: Id_Echipa, Denumire, Adresa, Id_Clasament, Nr_Puncte
❖	Se considera baza de date cu zborurile dintr-un aeroport formata din urmatoarele tabele: Avion, Ruta si Echipaj. Tabelul Avion contine urmatoarele campuri: Id_Avion, Nr_Locuri, Detalii, Producator, Id_Ruta. Tabelul Ruta contine campurile: Id_Ruta, Data_Plecare, Data_Sosire, Ruta. Tabelul Echipaj contine campurile: CNP, Nume, Id_Avion, Adresa, Calificare

❖	Se considera baza de date cu pacientii dintr-un spital formata din urmatoarele tabele: Pacient, Salon si Medic. Tabelul Pacient contine urmatoarele campuri: CNP, Nume, Prenume, Sex, CNP, Adresa, Data_Internarii, Id_Salon, Id_Medic. Tabelul Salon contine campurile: Id_Salon, Denumire, Nr_Paturi, Sef_Salon. Tabelul Medic contine campurile: Id_Medic, Nume, Functie, Specializare
❖	Se considera baza de date cu studenti dintr-un camin formata din urmatoarele tabele: Student, Facultate si Camin. Tabelul Student contine urmatoarele campuri: Nr_Leg, Nume, Prenume, Sex, CNP, Adresa, Id_Camin, Id_Facultate. Tabelul Camin contine campurile: Id_Camin, Nr_Locuri, Adresa, Reprezentant. Tabelul Facultate contine campurile: Id_Facultate, Denumire, Profil, Adresa, Decan
❖	Se considera baza de date cu orarul dintr-o facultate formata din urmatoarele tabele: Materie, Sali si Orar. Tabelul Materie contine urmatoarele campuri: Id_Materie, Denumire, Profesor, Asistent, Nr_Ore_Curs, Nr_Ore_Lab. Tabelul Sali contine campurile: Id_Sala, Denumire, Nr_Locuri, Responsabil. Tabelul Orar contine campurile: Ora_Inceput, Ora_Sfarsit, Id_Materie, Id_Sala

Fiecare echipa va fi formata din trei elevi. In cadrul acesteia rolurile si sarcinile fiecaruia vor fi stabilite din faza initiala. Un elev va fi desemnat responsabil cu strangerea (colectarea) informatiilor din viata cotidiana, altul cu intocmirea modelului de oraganizarea a informatiilor si construirea interfetei cu utilizatorul iar celalat elev cu implementarea metodelor ce rezolva cerintele de baza ale sistemului software. Ei vor colabora pentru realizarea fiecarei parti din proiect.

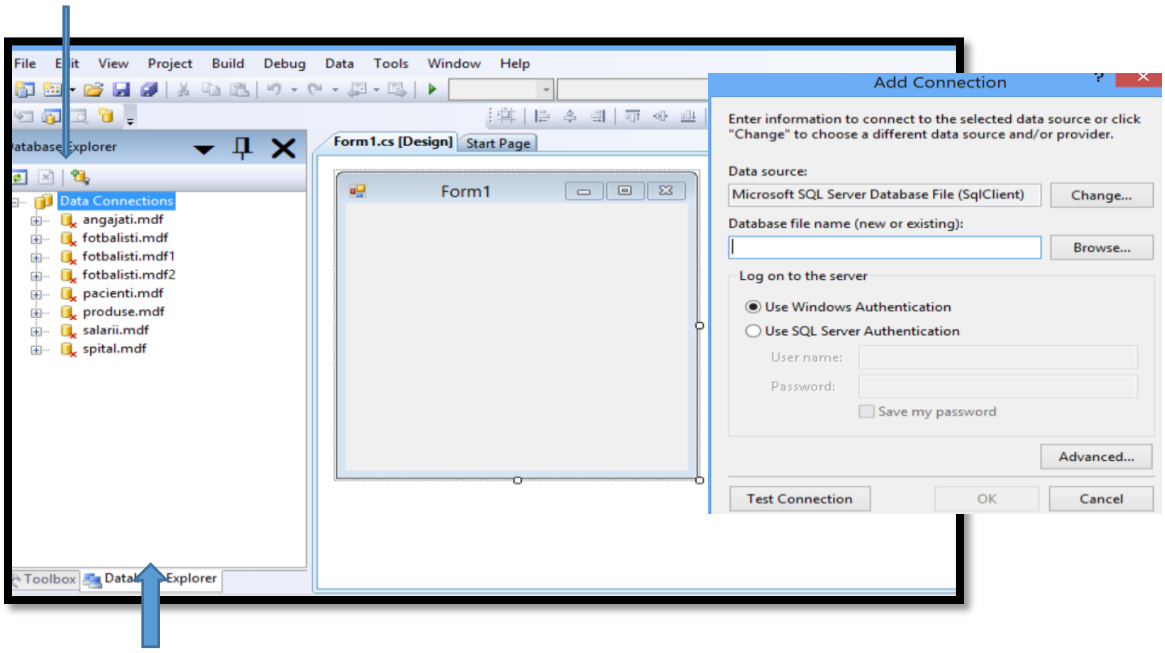
Se pleaca de la premisa ca au studiat anterior operatiile cu baze de date din SQL. Pentru a se obisnui cu implementarea acestora in mediul Visual C# 2013 elevii vor invata modalitatea de conectarea la BD in acest mediu visual.

5.2.5 Fisele practice din laborator

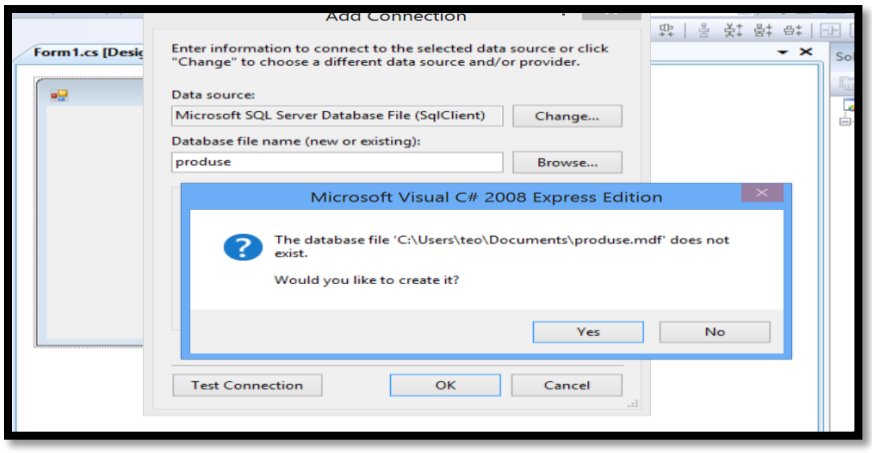
Fisa practica de lucru. Conectarea la BD in Microsoft Visual C# 2013

Etape:

- Se creeaza o noua aplicatie: *New Project* → *Windows Form Application*
- Se Selectati tabul "*Data Explorer*", apoi se face click dreapta pe "Data *Connection*" si se alegeboptiunea "*Add Connection*"

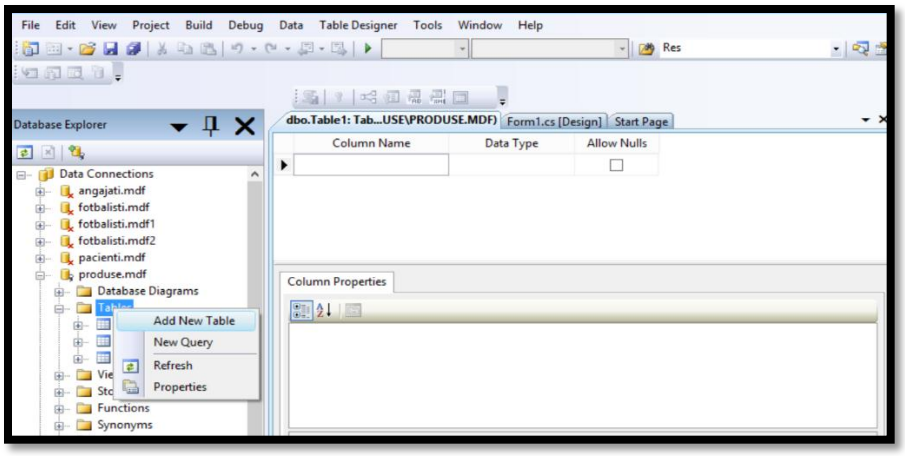


- Se introduce numele bazei de date care trebuie creata in campul "*Database file name (new or existing)"* si apoi selectati butonul "*yes*".

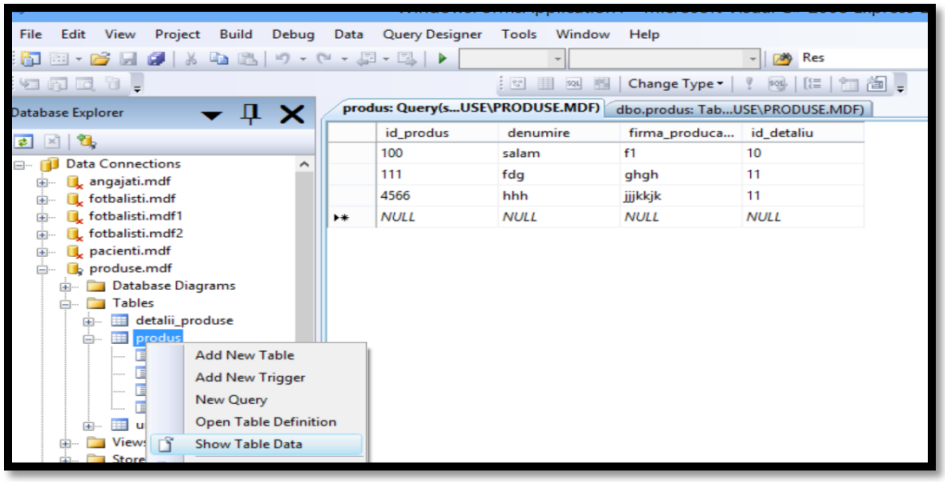


- Se expandeaza baza de date si se creaza structura tabelelor, click dreapta pe elemental "*Tables*" si optiunea "*Add New Table*" din meniul afisat.

Se creaza structura tabelelor dorite in aplicatie dupa care se populeaza cu date, apasand click dreapta pe numele tabelului si alegerea optiunii *Show*



❖ *Tabel Data.*



Respectiva fisa va fi lucrata in laboratorul de informatica sub directa indrumarea a profesorului. Pentru o reactualizarea a cunostiintelor in orice moment, fisa va fi postata si pe site-ul de colaborare.

In scopul realizari proiectului grupurile de elevi trebuie sa-si reactualizeze cunostiintele legate de folosirea bazelor de date in SQL, studiate anterior. In sprijinul acstui scop fisierul cu principalele comenzi SQL au fost postate pe site-ul de colaborare (se regaseste in **Anexa 1** a prezentei lucrari).

Intrucat proiectele ale caror teme au fost inspirate din viata cotidiana ele au relevanţă pentru viaţa elevilor şi pot implica reprezentanţi ai comunităţii sau

experţi din exterior, care asigură un context pentru învăţare. Elevii pot prezenta ceea ce au învăţat unui public real, pot face legătura cu resursele comunităţii, pot apela la experţi în domeniul respectiv sau pot comunica prin intermediul tehnologiei.o buna

Strangerea informatiilor in scpoul proiectarii aplicatiei implica o buna comunicare cu comunitatea si expertii din domeniu adecvat.

Cum se realizează relaţionarea dintre învăţarea bazată pe proiecte şi investigaţie? Investigaţia presupune o gamă largă de activităţi legate de curiozitatea noastră naturală în legătură cu lumea care ne înconjoară. În contextul educaţiei, investigaţia dobândeşte un înţeles specific. Profesorii care folosesc această tehnică drept strategie didactică îi încurajează în mod normal pe elevi să pună întrebări, să planifice şi să desfăşoare acţiuni de căutare de informaţii, să facă observaţii şi să reflecteze la ceea ce au descoperit. Totuşi, aceasta nu este o definiţie statică. Chiar şi la nivelul unei singure clase, activităţile de investigaţie se pot desfăşura după un continuum — între activităţi mai structurate dirijate de profesor şi activităţi deschise, mai libere, dirijate de interesele elevilor (Jarrett, 1997).

Ne-am putea gândi la învăţarea prin proiecte şi ca la o subcategorie a învăţării prin investigaţie. O analiză a cercetărilor efectuate în domeniul învăţării prin proiecte duce la concluzia că astfel de proiecte ce se concentrează pe aspecte sau probleme „îi conduc pe elevi spre întâlnirea (sau confruntarea) cu conceptele şi principiile centrale ale unei discipline” (Thomas, 2000, p. 3). Mai mult, activităţile centrale ale unui proiect implică investigaţia şi crearea de noi cunoştinţe de către elev (Thomas, 2000). Elevii pot să aleagă atunci când este vorba de conceperea propriului proiect, ceea ce le permite să-şi urmărească interesele şi să-şi manifeste curiozitatea. Răspunzând propriilor întrebări, elevii pot investiga subiecte care nu au fost identificate de profesor ca obiective ale procesului de învăţare.

In stadiul in care ne aflam cu implementarea metodei asupra unitati de invatare propusa am ajuns in punctul in care fiecare grup de elevi a investigat in viata

cotidiana universul ce trebuie proiectat. Informatiile au fost colectate si urmeaza a fi create structurile de date necesare implementarii modelului relational.
Date vor fi organizate in tabele (fiecarei entitati ii corespunde un tabel). Se vor stabili relatiile de baza intre tabele, constrangerile adecvate.
Mediile de comunicare on-line vor avea un rol important in comunicarea dintre profesor si elevi, in ceace priveste conceperea adecvata a structurii bazei de date.
Profesorii mai degrabă „antrenează" şi „modelează" şi vorbesc mai puţin. Ei trebuie să fie pregătiţi pentru a accepta „abaterile de la direcţie" care pot interveni pe parcursul desfăşurării unui proiect. Profesorii se pot afla în situaţia de a învăţa ei înşişi alături de elevi pe măsură ce proiectul se desfăşoară.
Provocările specifice cu care se pot confrunta profesorii includ:
* Recunoaşterea acelor situaţii care pot contribui la realizarea unor proiecte bune
* Structurarea problemelor ca oportunităţi de învăţare
* Colaborarea cu colegii pentru a dezvolta proiecte interdisciplinare
* „Administrarea" procesului de învăţare
* Integrarea corespunzătoare a tehnologiilor
* Conceperea unor metode şi instrumente de evaluare autentice
Dupa ce a fost stabilita structura fiecarui proiect si au fost reactualizate cunostiintele din Anexa 1(fisierul operatii de baza in SQL incarcate pe site-ul de colaborare) se va trece la o noua etapa de insusire de noi cunostinte si deprinderi cu tot colectivul clasei in laboratorul de informatica.
Elevii vor primi urmatoarea fisa de lucru.

Fisa de lucru.

- Populareа bazei de date „produse"(creata anterior) cu date conform schemei de mai jos:

Column Name	Data Type	Allow Nulls
id_detaliu	numeric(18, 0)	☐
data_fabricatie	datetime	☐
data_expirare	datetime	☐
id_um	numeric(18, 0)	☐
pret	numeric(18, 0)	☐

produse

Column Name	Data Type	Allow Nulls
id_produs	numeric(18, 0)	☐
denumire	nvarchar(50)	☐
firma_producatoare	nvarchar(50)	☐
id_detaliu	numeric(18, 0)	☐

unitati_masura

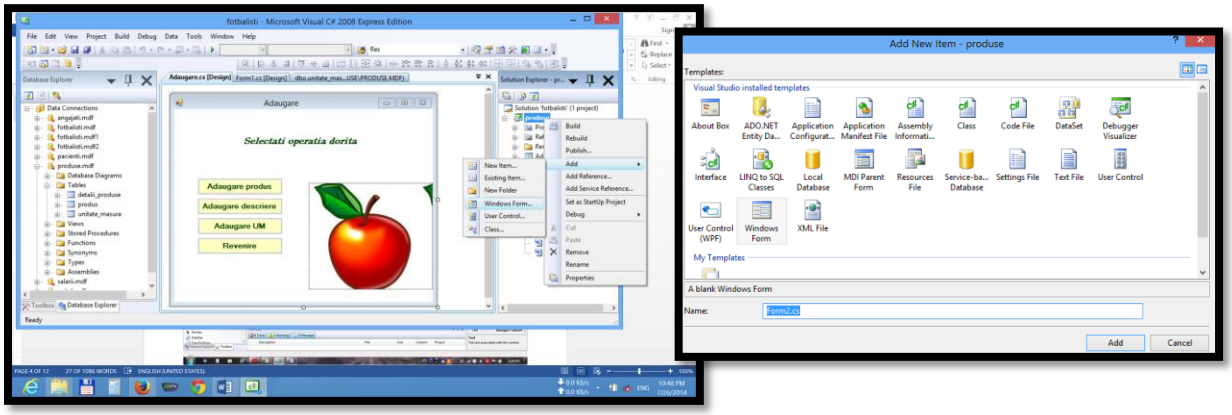
Column Name	Data Type	Allow Nulls
id_um	numeric(18, 0)	☐
descriere	nvarchar(50)	☑

❖ Cream formularul de pornirea, de unde avem posibilitatea accesari operatiilor de baza.

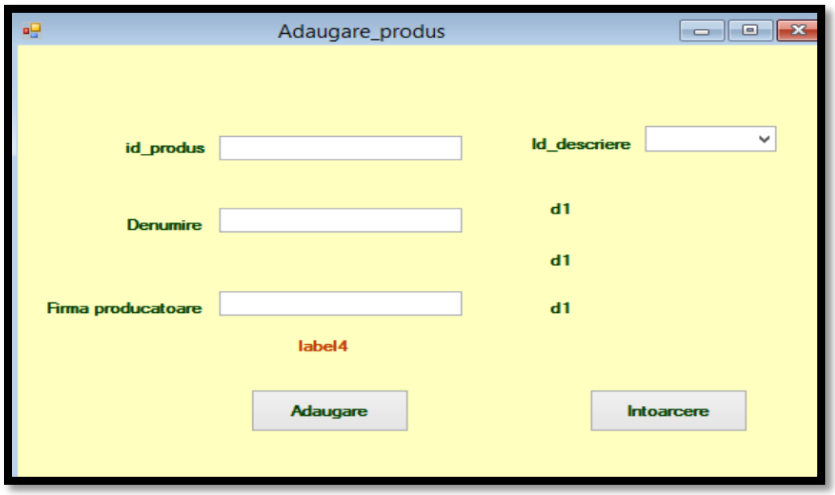
Pentru acest lucru din tabul “Solution Explorer” executam dublu-click pe formularul“Form1” si apoi adaugam 5 butoane: “Adaugare” (button1),

"Modificare" (button2), "Stergere" (button3), "Afisare" (button4), "Iesire" (button5), o eticheta „Selectati operatia dorita", o imagine adecvata.

- Se creaza cate un nou formular pentru fiecare actiune a butoanelor din formularul 1 prin selectarea numelelui de proiect din fereastra Solution Explorer, click dreapta → Add → Window Form.



- In formularul „Adaugare_produs" construim 3 controale de tip TextBox, un control ComboText, cele 8 etichete si doua controale de tip Button („Adaugare","Intoarecere").



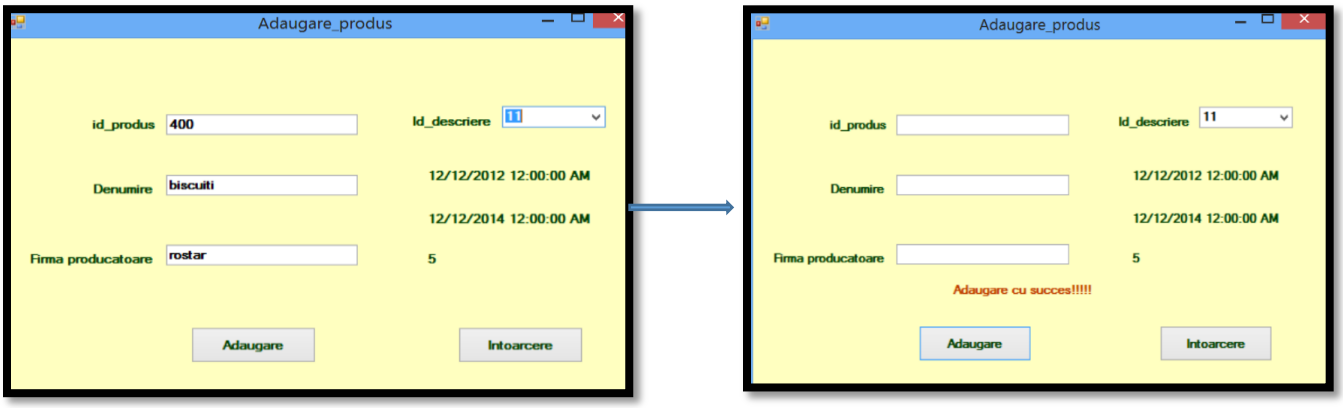
Butonului „Adaugare" ii atasam evenimentul dublu click in care editam urmatorul cod necesar adaugari unui nou produs in baza de date.

```
private void b1_Click(object sender, EventArgs e)
        {
            string insertsql;
```

```
            insertsql = "insert into produs (id_produs, denumire,
firma_producatoare, id_detaliu) values(" + tb11.Text + ",'" + tb22.Text + "','"
+ tb33.Text + "'," + cb1.Text + ")";
            try
            {
            SqlCommand cmd = new SqlCommand(insertsql, co);
            cmd.ExecuteNonQuery();
            tb11.Text = "";
            tb22.Text = "";
            tb33.Text = "";

            le.Text = "Adaugare cu succes!!!!! ";
            le.Visible = true;
            }
             catch(System.Data.SqlClient.SqlException)
             { le.Text="Atentie nu se poate adauga cheie primara duplicata!!!!!
";
             le.Visible=true;
             tb11.Text = "";
             tb22.Text = "";
             tb33.Text = "";

                  }
```

Anterior in constructorul clasei „*Adaugare_produs*” s-a realizat conectarea la baza de date si definirea listei corespunzatoare controlului *ComboText*, cu preluarea informatiilor din tabelul *detalii_produse*.

```
string s = "Data Source=.\\SQLEXPRESS;AttachDbFilename=D:\\net\\Ciurea Rodica
\\produse\\produse.mdf;Integrated Security=True;Connect Timeout=30;User
Instance=True";
            co = new SqlConnection(s);
            co.Open();
            string selectSQL = "SELECT id_detaliu FROM detalii_produse";
            SqlCommand cmd = new SqlCommand(selectSQL, co);
            SqlDataAdapter adapter = new SqlDataAdapter(cmd);
            DataSet ds = new DataSet();
            adapter.Fill(ds, "detalii_produse");
            foreach (DataRow r in ds.Tables["detalii_produse"].Rows)
            {
                cb1.Items.Add(r[0]);
            }
            if (cb1.Items.Count > 0)
                      cb1.SelectedIndex = 0;
```

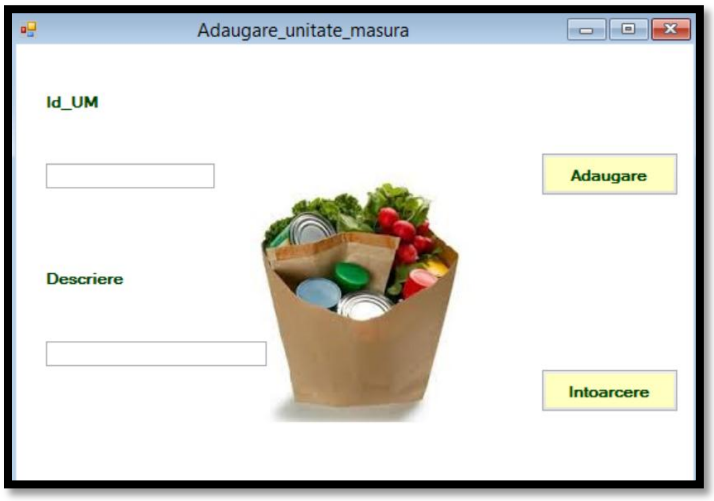


Asociem control-ului ComboText evenimentul SelectedIndexChanged in care editam codul urmator:

```
private void cb1_SelectedIndexChanged_1(object sender, EventArgs e)
        {

            string nrid = Convert.ToString(cb1.SelectedItem);
            if (nrid == null || nrid.Length == 0)
            {
                return;
            }

            string selectSQL = "SELECT * FROM detalii_produse WHERE
id_detaliu=" + nrid + "";
            SqlCommand cmd = new SqlCommand(selectSQL, co);
            SqlDataAdapter adapter = new SqlDataAdapter(cmd);
            DataSet ds = new DataSet();
            adapter.Fill(ds, "detalii_produse");
            DataRow r = ds.Tables["detalii_produse"].Rows[0];
            label6.Text = Convert.ToString(r["data_fabricatie"]);
            label7.Text = Convert.ToString(r["data_expirare"]);
            label8.Text = Convert.ToString(r["pret"]);


            label6.Visible = true;
            label7.Visible = true;
            label8.Visible = true;
```

La selectarea din lista aunui element se vor afisa prin intermediul etichetelor informatiile corespunzatoare descrierii produsului (initial aceste controale au fost setate pe invizibil).

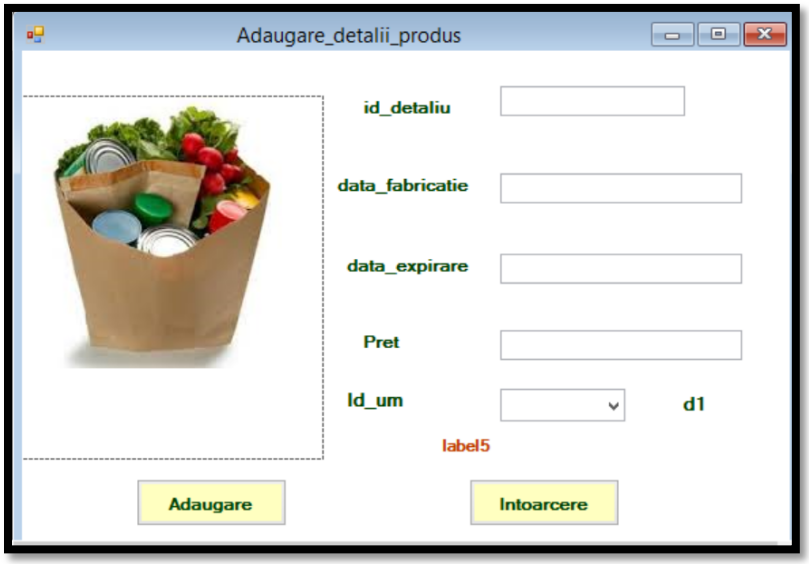
- In formularul „*Adaugare_unitate_masura*” se adauga doua controale de tip *TextBox (tb1, tb2)*, doua controale de tip *Label(label1, label2)* si doua controale de tip *Button(b1,b2)*.



Butonului „Adaugare” ii asociem evenimentul Click, caruia ii asociem urmatoru cod necesar introduceri inregistrarilor:

```
private void b1_Click(object sender, EventArgs e)
        {
            string s = "Data
Source=.\\SQLEXPRESS;AttachDbFilename=D:\\net\\Ciurea Marius-
Catalin\\produse\\produse.mdf;Integrated Security=True;Connect Timeout=30;User
Instance=True";
            //string s = "Data
Source=.\\SQLEXPRESS;AttachDbFilename=C:\\Users\\teo\\Documents\\fotbalisti.mdf
;Integrated Security=True;Connect Timeout=30;User Instance=True";
            SqlConnection co = new SqlConnection(s);
            co.Open();

            string insertsql;

            insertsql = "insert into unitate_masura(id_um, descriere) values("
+ tb1.Text + ",'" + tb2.Text + "')";
            try
            {
                SqlCommand cmd = new SqlCommand(insertsql, co);
                cmd.ExecuteNonQuery();
                tb1.Text = "";
                tb2.Text = "";


                le.Text = "Adaugare cu succes!!!!! ";
                le.Visible = true;
            }
            catch (System.Data.SqlClient.SqlException)
            {
                le.Text = "Atentie nu se poate adauga cheie primara
duplicata!!!!! ";
                le.Visible = true;
                tb1.Text = "";
                tb2.Text = ""; }}
```

❖ In formularul „Adaugare_detalii_produs”, se adauga cinci controale de tip TextBox (tb1, tb2, tb3, tb4), un control de tip ComboBox, sapte controale de tip Label, doua controale de tip Button (b1, b2).



Butonului „Adaugare”ii asociem evenimentul Click in care editam urmatorul cod necesar introducerii detaliilor legate de produse:

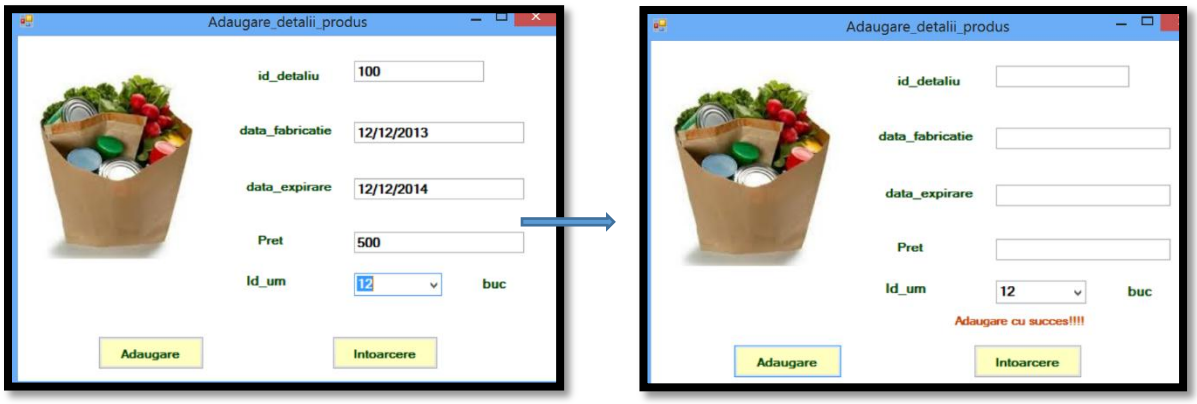
```
private void b1_Click(object sender, EventArgs e)
        {
            string insertsql;

            insertsql = "insert into detalii_produse(id_detaliu,
data_fabricatie, data_expirare,pret, id_um) values(" + tb1.Text + ",'" +
tb2.Text + "','" + tb3.Text + "'," + tb4.Text + ","+cb1.Text+")";

            try
            {
                SqlCommand cmd = new SqlCommand(insertsql, co);
                cmd.ExecuteNonQuery();
                l5.Text = "Adaugare cu succes!!!!";
                l5.Visible = true;
                tb1.Text = "";
                tb2.Text = "";
                tb3.Text = "";
                tb4.Text = "";

            }
            catch (System.Data.SqlClient.SqlException eb)
            {
                l5.Text = "posibila cheieie duplicata";
                l5.Visible = true;
                tb1.Text = "";
                tb2.Text = "";
                tb3.Text = "";
                tb4.Text = "";
            }         }
```

Anterior in constructorul clasei „Adaugare_detalii” s-a realizat conectarea la baza de date si definirea listei corespunzatoare controlului ComboText, cu preluarea informatiilor din tabelul detalii_produse.

```
string s = "Data Source=.\\SQLEXPRESS;AttachDbFilename=D:\\net\\Ciurea Rodics-
Elena\\produse\\produse.mdf;Integrated Security=True;Connect Timeout=30;User
Instance=True";
            co = new SqlConnection(s);
            co.Open();
            string selectSQL = "SELECT id_um FROM unitate_masura";
            SqlCommand cmd = new SqlCommand(selectSQL, co);
            SqlDataAdapter adapter = new SqlDataAdapter(cmd);
            DataSet ds = new DataSet();
            adapter.Fill(ds, "unitate_masura");
            foreach (DataRow r in ds.Tables["unitate_masura"].Rows)
            {
                cb1.Items.Add(r[0]);
            }
            if (cb1.Items.Count > 0)
              cb1.SelectedIndex = 0;
```

Asociem control-ului ComboText evenimentul SelectedIndexChanged in care editam codul urmator:

```
private void cb1_SelectedIndexChanged(object sender, EventArgs e)
        {
            string nrid = Convert.ToString(cb1.SelectedItem);
            if (nrid == null || nrid.Length == 0)
            {
                return;
            }
            string selectSQL = "SELECT * FROM unitate_masura WHERE id_um=" +
nrid + "";
            SqlCommand cmd = new SqlCommand(selectSQL, co);
            SqlDataAdapter adapter = new SqlDataAdapter(cmd);
            DataSet ds = new DataSet();
            adapter.Fill(ds, "unitate_masura");
            DataRow r = ds.Tables["unitate_masura"].Rows[0];
            label6.Text = Convert.ToString(r["descriere"]);


            label6.Visible = true;
              }
```



- In formularul „Modificare_produs” se adauga doua controale de tip ComboText (cb1, cb2), doua controale de tip TextBox (tb1,tb2), opt controale de tip Label si doua controale de tip Button (b1,b2).

Primului control de tip *ComboBox(cb1)* ii asociem un eveniment *SelectedIndex Changed* , al carui cod este de forma:

```
private void cb1_SelectedIndexChanged(object sender, EventArgs e)
        {

            string nri = Convert.ToString(cb1.SelectedItem);
            if (nri == null || nri.Length == 0)
            {
                return;
            }

            string selectSQL = "SELECT * FROM produs WHERE id_produs='" + nri +
"'";
            SqlCommand cmd = new SqlCommand(selectSQL, co);
            SqlDataAdapter adapter = new SqlDataAdapter(cmd);
            DataSet ds = new DataSet();
            adapter.Fill(ds, "produs");
            DataRow r = ds.Tables["produs"].Rows[0];
```

```
            tb1.Text = (string)r["denumire"];
            tb2.Text = (string)r["firma_producatoare"];
            cb2.SelectedItem = r["id_detaliu"];


            l2.Visible = true;
            l3.Visible = true;
            l4.Visible = true;
            l5.Visible = true;


            tb1.Visible = true;
            tb2.Visible = true;
            cb2.Visible = true;

            cb1.Enabled = false;

        }
```

Rolul metodei editate este de a prelua codurile de produse existente in tabela produse, si de a afisa detaliile legate de fiecare produs selectat in cele patru etichete care initial sunt invizibile.

Control de tip *ComboBox (cb2)* ii asociem un eveniment *SelectedIndex Changed* , al carui cod este de forma:

```
private void cb2_SelectedIndexChanged(object sender, EventArgs e)
        {
            string nrid = Convert.ToString(cb2.SelectedItem);
            if (nrid == null || nrid.Length == 0)
            {
                return;
            }

            string selectSQL = "SELECT * FROM detalii_produse WHERE
id_detaliu=" + nrid + "";
            SqlCommand cmd = new SqlCommand(selectSQL, co);
            SqlDataAdapter adapter = new SqlDataAdapter(cmd);
            DataSet ds = new DataSet();
            adapter.Fill(ds, "detalii_produse");
            DataRow r = ds.Tables["detalii_produse"].Rows[0];
            a1.Text ="fabricat la: "+ Convert.ToString(r["data_fabricatie"]);
            a2.Text = "expira la:"+Convert.ToString(r["data_expirare"]);
            a3.Text ="pret="+ Convert.ToString(r["pret"]);

            string nru = Convert.ToString(r["id_um"]);
            string selectSQL1 = "SELECT * FROM unitate_masura WHERE id_um=" +
nru+ "";
            SqlCommand cmd1 = new SqlCommand(selectSQL1, co);
            SqlDataAdapter adapter1 = new SqlDataAdapter(cmd1);
            DataSet ds1 = new DataSet();
            adapter1.Fill(ds1, "unitate_masura");
            DataRow r1 = ds1.Tables["unitate_masura"].Rows[0];
            a4.Text = "um:  "+Convert.ToString(r1["descriere"]);
```

```
            a1.Visible = true;
            a2.Visible = true;
            a3.Visible = true;
            a4.Visible = true;
        }
```

Butonului *„Modificare”* ii asociem evenimentul *Click*, pentru care se editeaza codul:

```
private void b1_Click(object sender, EventArgs e)
        {
            string sql = "update produs set denumire = '" + tb1.Text + "',
firma_producatoare='" + tb2.Text +"', id_detaliu = " + cb2.SelectedItem + "
where id_produs="+cb1.SelectedItem+"";
            SqlCommand cmd = new SqlCommand(sql, co);
            cmd.ExecuteNonQuery();
            le.Text = "Modificare cu succes!!!!! ";
            le.Visible = true;
            tb1.Text = "";
            tb2.Text = "";
            cb2.Text = "";

            cb1.Text = "";
            cb1.Enabled = true;
            a1.Visible = false;
            a2.Visible =false;
            a3.Visible = false;
            a4.Visible = false;
        }
```

Anterior in constructorul clasei „Modificare_produs” s-a realizat conectarea la baza de date.

```
            string s = "Data
Source=.\\SQLEXPRESS;AttachDbFilename=D:\\net\\Ciurea Rodica-
Elena\\produse\\produse.mdf;Integrated Security=True;Connect Timeout=30;User
Instance=True";
           // string s = "Data
Source=.\\SQLEXPRESS;AttachDbFilename=C:\\Users\\teo\\Documents\\fotbalisti.mdf
;Integrated Security=True;Connect Timeout=30;User Instance=True";
            co = new SqlConnection(s);
            co.Open();
            string selectSQL = "SELECT id_produs FROM produs";
            SqlCommand cmd = new SqlCommand(selectSQL, co);
            SqlDataAdapter adapter = new SqlDataAdapter(cmd);
            DataSet ds = new DataSet();
            adapter.Fill(ds, "produs");
            foreach (DataRow r in ds.Tables["produs"].Rows)
            {
                cb1.Items.Add(r[0]);
            }

            selectSQL = "SELECT id_detaliu FROM detalii_produse";
```

```
            cmd = new SqlCommand(selectSQL, co);
            adapter = new SqlDataAdapter(cmd);
            ds = new DataSet();
            adapter.Fill(ds, "detalii_produse");
            foreach (DataRow r in ds.Tables["detalii_produse"].Rows)
            {
                cb2.Items.Add(r[0]);
            }
```

- In formularul „*Stergere_produs*” se adauga un control *ComboText (cb1),* trei controale de tip *Label* si doua controale de tip *Button (b1,b2).*

In constructorul clasei *Stergere_produs* se creeaza conexiunea cu baza de date:

```
SqlConnection co;
        public Stergere_produs()
        {
            InitializeComponent();
            string s = "Data
Source=.\\SQLEXPRESS;AttachDbFilename=D:\\net\\Ciurea Marius-
Catalin\\produse\\produse.mdf;Integrated Security=True;Connect Timeout=30;User
Instance=True";
            co = new SqlConnection(s);
            co.Open();
            string selectSQL = "SELECT id_produs FROM produs";
            SqlCommand cmd = new SqlCommand(selectSQL, co);
            SqlDataAdapter adapter = new SqlDataAdapter(cmd);
            DataSet ds = new DataSet();
            adapter.Fill(ds, "produs");
            foreach (DataRow r in ds.Tables["produs"].Rows)
            {
                cb1.Items.Add(r[0]);
            }
            if (cb1.Items.Count > 0)
                    cb1.SelectedIndex = 0;
```

Controlului ComboText (cb1) ii asocie evenimentul *SelectedIndexChanged*, pentru care asociem codul:

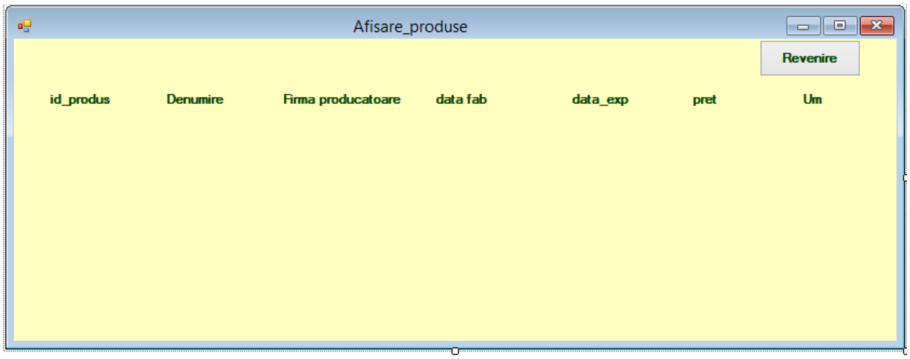
```
private void cb1_SelectedIndexChanged(object sender, EventArgs e)
        {
            string nri = Convert.ToString(cb1.SelectedItem);
            if (nri == null || nri.Length == 0)
            {
                return;
            }

            string selectSQL = "SELECT * FROM produs WHERE id_produs='" + nri +
"'";
            SqlCommand cmd = new SqlCommand(selectSQL, co);
            SqlDataAdapter adapter = new SqlDataAdapter(cmd);
            DataSet ds = new DataSet();
            adapter.Fill(ds, "produs");
            DataRow r = ds.Tables["produs"].Rows[0];
            l1.Text = (string)r["denumire"];
            l2.Text = (string)r["firma_producatoare"];
            l1.Visible = true;
            l2.Visible = true;
        }
        }
```

In urma codului lista derulanta va fi construita cu id-ul produsele din tabelul produse si prin intermediul etichetelor initial invizibile se afiseaza si celelalte campuri ale produsului.

Butonului „*Stergere*” ii asociem evenimentul *Click*, in care editam codul:

```
private void button1_Click(object sender, EventArgs e)
        {
            //SqlConnection co = new SqlConnection(@"Data
Source=.\SQLEXPRESS;AttachDbFilename=C:\Users\Jorhel\Documents\Salarii.mdf;Inte
grated Security=True;Connect Timeout=30;User Instance=True");
           // co.Open();
            string deletesql = "delete from produs where id_produs='" +
cb1.Text + "'";
            try
            {
                SqlCommand cmd = new SqlCommand(deletesql, co);
                cmd.ExecuteNonQuery();
                l1.Visible = false;
                l2.Visible = false;
                le.Text = "Stergere efectuata";
                le.Visible = true;
            }
            catch (System.Data.SqlClient.SqlException)
            { le.Text = "Nu s-a putut face stergerea!";
              le.Visible = true;
            }
              }
```

❖ In formularul „*Afisare_produse*” se adauga sapte controale de tip eticheta *(l1, l2, l3, l4, l5, l6)*.



In constructorul clasei „Afisare_p” se adauga codul necesar conectarii la baza de date, afisarii produselor cat si a detaliilor legate de acestea.

```
string s = "Data Source=.\\SQLEXPRESS;AttachDbFilename=D:\\net\\Ciurea Rodica-
Elena\\produse\\produse.mdf;Integrated Security=True;Connect Timeout=30;User
Instance=True";
            co = new SqlConnection(s);
            co.Open();
            string selectSQL = "SELECT id_produs, denumire, firma_producatoare,
data_fabricatie, data_expirare,pret, descriere FROM produs, detalii_produse,
unitate_masura where produs.id_detaliu=detalii_produse.id_detaliu and
detalii_produse.id_um=unitate_masura.id_um order by denumire";
            SqlCommand cmd = new SqlCommand(selectSQL, co);
            SqlDataAdapter adapter = new SqlDataAdapter(cmd);
            DataSet ds = new DataSet();
            adapter.Fill(ds, "produs");
            l1.Text += "\n" + "--------------" + "\n\n";
            l2.Text += "\n" + "-------------" + "\n\n";
            l3.Text += "\n" + "-------------" + "\n\n";
            l4.Text += "\n" + "-------------" + "\n\n";
            l5.Text += "\n" + "-------------" + "\n\n";
            l6.Text += "\n" + "------------" + "\n\n";
            l7.Text += "\n" + "------------" + "\n\n";
            foreach (DataRow r in ds.Tables["produs"].Rows)
            {
                l1.Text += r["id_produs"] + "\n" + "--------------" + "\n";
                l2.Text += r["denumire"] + "\n" + "--------------" + "\n";
                l3.Text += r["firma_producatoare"] + "\n" + "--------------" +
"\n";

            }
            DataSet ds1 = new DataSet();
            adapter.Fill(ds1, "detalii_produse");
            foreach (DataRow r1 in ds1.Tables["detalii_produse"].Rows)
            {
                l4.Text += r1["data_fabricatie"] + "\n" + "--------------" +
"\n";
                l5.Text += r1["data_expirare"] + "\n" + "--------------" +
"\n";
                l6.Text += r1["pret"] + "\n" + "--------------" + "\n";
            }
```

```
adapter.Fill(ds1, "unitate_masura");
foreach (DataRow r2 in ds1.Tables["unitate_masura"].Rows)
{

    l7.Text += r2["descriere"] + "\n" + "--------------" + "\n";
}
```

- Butoanele care presupun revenirea la formulare precedente pot sa aiba atasat un eveniment in al carui cod se editeaza:

```
        this.Close();
new Form1().Show();//se revine la primul formular.
```

5.2.6 Planul de evaluare.

Pe parcursul unitatii de invatare, profesorii aduna informatii despre modul cum invata elevii. Observa modul cum isi desfasoara activitatile in cadrul grupului, este atent la conversatiile purtate legate de modul de colaborare, adreseaza intrebari elevilor. Astfel el realizeaza o evaluare continuu, numita si formativa. Rolul acestor evaluarii este de a pastra linia corecta de a realiza sarcinile priectului, de a stimula si a determina sporirea increderii in sine a elevului.

Evauarea sumativa se realizeaza la sfarsitul unitatii de invatare si o fera informatii legate de dificultatile aparute pe parcursul invatarii, de zonele in care exista probleme.

Scopurie ce trebuie atinse in evaluarea formativa:

- Analiza nevoilor de invatare
- Incurajarea autonomiei in invatare si a colaborari
- Monitorizarea progresului
- Verificarea intelegerii si incurajarea metacognitiei

Scopul principal al evaluarii este:

❖ Demonstrarea intelegerii si competentelor formate

Planificarea corecta a evaluarii ne asigura o pastrare a focalizari pe obiectivele de invatare. Planul de evlauare include metode si instrumente de evaluare prin care se urmaresc asteparile cu privire la calitatea produselor si perfoeantelor elevilor.
In cadrul planului de evaluare se poate intocmi un grafic de timp pentru reprezentarea momentelor cand au loc evaluarile.

Grafic de timp pentru evaluare					
Inaintea inceperii activitatilor din proiect		Elevii lucreaza la proiect si finalizeaza sarcini de lucru		Dupa ce proiectul este finalizat	
Tehnica intrebarilor	Planificarea proiectului	Rezumate scrise	Evaluarea de grup si autoevaluarea	Rubrica de ziar	Proba de simulare
Jurnale	Graficul K-W-L	Lista de observatii	Rubrica de ziar	Graficul K-W-L	Eseu de reflectii
Test initial		Jurnale	Conferinte		
		Chestionare			

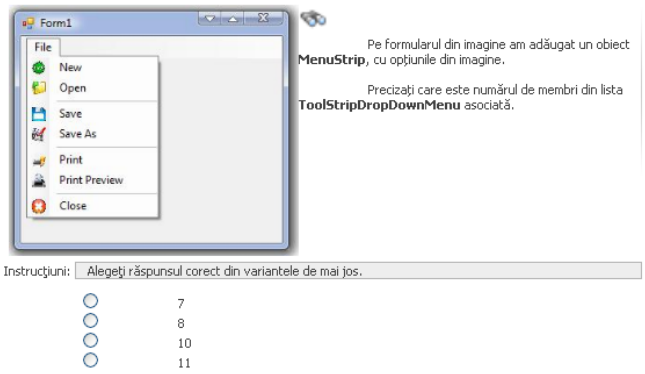
Inainte de demararea proiectului se realizeaza o evaluare initiala a cunostintelor pentru a avea reperele necesare procesului de instruire. In acest scop elevii vor primi in mod individual un test cu itemi diversi.

Nume si prenume_____
Clasa a XII-a_____

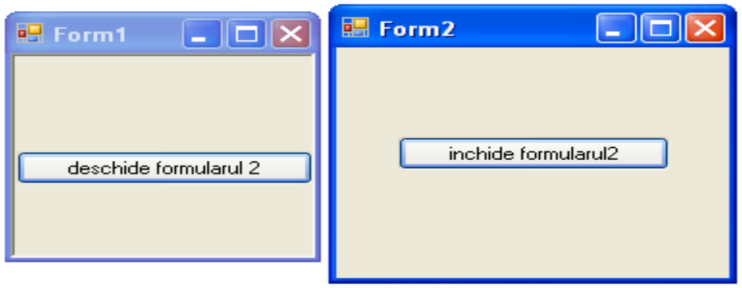
Evaluare initiala

1. Care este metoda corectă de a instanţia un obiect în C#?
   a. newObj = newObject();
   b. newObj = System.CreateObject(newObject);
   c. newObj = new newObject();
2. Ce face codul public class B : A { }
   a. va da o eroare

b. defineşte clasa B care va moşteni toate metodele clasei A
c. defineşte clasa B care va moşteni doar metodele publice ale clasei A

3. 5. Fişierele cu cod C# au extensia
   a. .#
   b. .csarp
   c. .cs
4. 

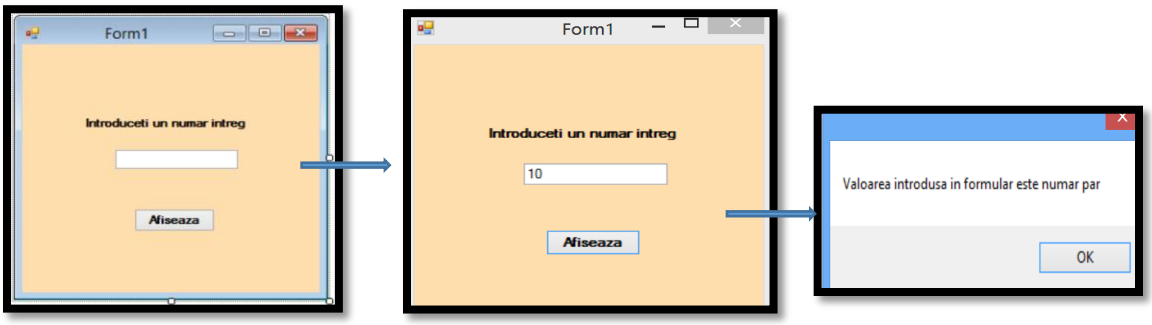
5. Private Button print = new button();
   a. Creeaza un control de tip buton
   b. Initializeaza un control de tip buton
   c. Instantiaza un control de tip buton
   d. a sib
   e. a si c
6. Pentru a bloca / debloca un obiect folosim metodele
   a. Lock şi Unlock
   b. Enter şi Exit
   c. Close şi Open
7. Completati codul pentru a deschide form2 si a inchide form1



private voidbutton1_Click(objectsender, EventArgse)
{
……

}

8. Pentru ca butonul *Afiseaza* sa raspunda la evenimentul click cu mesajul din casuta *„Valoarea introdusa in formular este numar par”* sau *„Valoarea introdusa in formular este numar impar”* in functie de valoarea introdusa in caseta *TextBox*, trebuie sa completam codul cu:




```
private void button1_Click(object sender, EventArgs e)
{
……
 }
```

9. Considerand entitatile PERSOANA si ADRESA , cand ne referim la domiciliul unei persoane, ce tip de relatie se poate stbili intre ele?
   a) unu la unu;
   b) unu la mai multi;
   c) mai multi la unu;
   d) mai multi la mai multi.
10. Care dintre urmatoarele entitati contine atribute nepotrivite:
   a) CLADIRE:adresa, etaj, numar, camera;
   b) CARTE: titlu, autor, data aparitiei, editura;
   c) ELEV: nume, data nasterii, loc de munca;
   d) ANGAJAT: nume, specializare, loc de munca.

11. Se da tabelul: CHITANTE (NrChitanta, Suma, DataChitanta, NrGhiseu). Care dintre urmatoarele interogari SQL are ca efect afisarea zilelor incare s-au emis mai mult de 2 chitante?

a) SELECT Nrchitanta, Suma, Data FROM CHITANTE WHERE COUNT(NrChitanta)>2
b) SELECT Nrchitanta, Suma, Data FROM CHITANTE HAVING COUNT(NrChitanta)>2
c) SELECT CHITANTE.Data FROM CHITANTE GROUP BY Data HAVING COUNT(NrChitanta)>2
d) SELECT COUNT(Data), COUNT(NrChitanta) FROM CHITANTE WHERE Data IS Unique

12. Se da tabelul: ANGAJATI (CodAngaj, Nume, DataAngajare, profesia, Salariu). Care dintre urmatoarele instructiuni SQL permite majorarea salariului cu 1% pentru salariatii angajati în anul 2005.

a) UPDATE Angajati SET Salariu = Salariu + 1% WHERE DataAngajare IN 2005
b) SELECT Salariu + 1%* Salariu FROM Angajati WHERE DataAngajare IN (#1/1/2005#,#31/12/2005#)
c) UPDATE Angajati SET Salariu = Salariu*1.01 WHERE YEAR(DataAngajare) = 2005
d) UPDATE Salariu FROM Angajati SET Salariu = Salariu * 101% WHERE DataAngajare BETWEEN #1/1/2005# AND #31/12/2005#

13. Care dintre urmatoarele variante este adecvata pentru inserarea unei DATE in tabel:
   a) insert into Employee(Start_Date) values (’05-FEB-2014’)
   b) insert into Employee(Start_Date) values (’FEB-05-2014’)
   c) ambele a si b
   d) nici o variant de raspuns

Unitatea de invatare „**Dezvoltarea si prezentarea unei aplicatii in mediul visual**” dispune de sase saptamani, fiecare pune la dispozitie trei ore de intalnire fata in fata.
K - W – L (Ştiu/ Vreau să ştiu/Am învăţat) - Situaţia de plecare (Stiu) este evidenţiată clar. În această etapă sunt scoase în la suprafaţă cunoştinţele şi deprinderile dobândite anterior şi care vor permite ca noile conţinuturi să fie relaţionate pe fundamentul celor cunoscute de elevi. Prima etapă, centrată pe întrebarea Ce ştiu despre subiect?, se suprapune momentului relativ formal de actualizare a cunoştinţelor anterioare (“idei-ancoră”).
În etapa a doua, centrată pe aspectul Ce vreau să ştiu?, elevii formulează întrebări referitoare la ceea ce ar dori să ştie despre subiectul abordat, listarea acestora făcându-se prin diferite proceduri
Consideram ca unitatea de invatare are in componenta sase lectii. Pentru fiecare dintre acestea elevul va trebui sa completeze diagrama K-W-L. La sfarsitul proiectului diagramele completate vor fi adaugate in portofoliu.
Motivul alegerii graficului KWL este că la finalul unitatii de învăţare elevul va avea o privire de ansamblu asupra progresului înregistrat la nivelul cunostinţelor.

Nume şi prenume _____

Clasa a XII –a _____

DIAGRAMA K-W-L

Completaţi la sfârşitul fiecărei lecţii următorul tabel :

Lecţia 1 . Modelarea vietii cotidiene prin intermediul bazelor de date.

Ce știu?	Ce aș vrea să știu?	Ce am învăţat ?
Ştiu că bazele de date sunt utilizate pentru pastrarea datelor ce modeleaza afaceri reale.	Cum extrag informatiile necesare construirii bazei de date.	Sa organizez corect informatiile in baza de date
	Care este modelul de organizare adecvat construirii aplicatiei	Baza de date sa raspunda cererilor din viata reala

Lecţia 2 . Conectarea la baza de date in mediul Visual C#.

Ce ştiu?	Ce aş vrea să ştiu?	Ce am învăţat ?

Lecţia 3 .Popularea bazei de date.

Ce ştiu?	Ce aş vrea să ştiu?	Ce am învăţat ?

Lecţia 4. Crearea interfetei aplicatiei prin exemple .

Ce ştiu?	Ce aş vrea să ştiu?	Ce am învăţat ?

Lecţia 5 . Folosirea operatiilor asupra bazelor de date in mediul Visual C#.

Ce ştiu?	Ce aş vrea să ştiu?	Ce am învăţat ?

Lecţia 6. Prezentarea aplicatiei .

Ce ştiu?	Ce aş vrea să ştiu?	Ce am învăţat ?

Chestionarea - Pe parcursul intregii unităţi se poate aplica chestionarea. Chestionarea este folosită pe parcursul întregii unităţi având rolul de a-i ajuta pe elevi să-şi dezvolte deprinderi cognitive de ordin superior şi pentru a-şi monitoriza permanent învăţarea. Pentru monitorizarea progresului, în cadrul grupului, profesorul va folosi un instrument de chestionare.

La sfarsitul fiecarei saptamani elevii or primi pe e-mail un chestionar pe care il vor completa. Datele vor fi colectate intr-o foaie de cacul de pe google drive si prin intermediul lor se va realiza un grafic privind eficienta metodei implementate.

Chestionar saptamana 1

- Apreciati pe o scara de la 1 la 4 importanta pastrarii informatiilor ce descriu o afacere in interiorul unei baze de date.
- Iti face placere sa aduni informatii legate de o afacere din viata cotidiana?
- Consideri ca impartirea pe grupe de lucru este o decizie buna?
- Rolul pe care l-ai primit in cadrul grupului te caracterizeaza.
- Preconizezi o buna colaborare cu echipa?
- Echipa voastra a reusit sa contacteze persoane implicate in afacerea reala pentru care trebuie sa implementati aplicatia?
- In ce statiu se afla echipa cu construirea structurii bazei de date?
- Reactualizarea cunostiintelor legate de operatiile de baza din SQL pe baza fisiei postate pe site-ul de colaborare intampina probleme?
- Vei putea participa la conferinta de pe Skype programata pentru aceasta saptamana?

Chestionarul final

După prezentarea proiectelor, eleviilor li se va aplica un chestionar cu următoarele întrebări:

- Cum ţi s-a părut lucrul în echipă?
- Cum te simţi mai bine lucrând: în grup sau în clasă individal?
- Ai participat cu idei in cadrul proiectului de grup?
- Dă un calificativ activităţii pe echipă?

  A. nesatisfăcător;
  B. satisfacator;
  C. bine;
  D. foarte bine

- Care au fost dificultăţile întâmpinate în realizarea acestui proiect?

In timpul eleborarii proiectului se va complete pentru fiecare grupa fisa de observatii:

Foaie de observaţie

Echipa _____ **Observator** _____

		Note
Obişnuinţe de lucru în grup	**Ce fac elevii:** • Lucrează individual sau colaborează cu alţii? • Încearcă să îi ajute pe ceilalţi , în ce feluri? • Incearcă să ceară sau să ofere ajutor? Dacă da, de la cine şi cui? • Participă şi în ce măsura la realizarea aplicaţiei? • Devin activ implicaţi şi se apucă de lucru?	
mat	**Elevii individual:** • Încearcă să explice punctul lor de vedere asupra aplicaţiei? • Îşi justifică părerile? • Acceptă si ţin cont de sugestiile şi ideile celorlalţi?	
Comunicarea în cadrul grupului	**Ce fac elevii:** • Vorbesc pentru a-şi clarifica ideile şi comunică cu ceilalţi? • Îşi asumă ambele roluri de prezentator şi *ascultător*? • Sunt capabili sa-şi susţină opiniile în faţa întregii clase? • Pot să exprime opinia grupului la fel de bine ca propriile păreri? • Sintetizează şi sumarizează ideile sale sau ale grupului?	

Cooperarea în cadrul grupului	**Ce face grupul:** • Distribuie sarcinile fiecărui membru al grupului? • Decide pentru alegerea unui leader de grup? • Realizează un plan pentru explicarea aplicaţiei proiect? • Se asigure că toţi membrii au înteles ce au de făcut? • Foloseşte timpul grupului într-un mod corespunzător? • Oferă susţinere fiecărui membru? • Urmăreşte activitatea de documentare şi realizare a proiectului	

In urma prezentarii aplicatiei create de catre fiecare grupa in fata intregului colectiv al clasei se vor completa urmatoarele formulare de feedback

Formular de feedback pentru prezentarea proiectului

(Peer Feedback)

Grupa 1 Prezentator: _____

Grupa 2 Evaluator: _____

1. Enumeraţi care vi s-au părut că sunt cele mai interesant părţi ale proiectului .

_____

_____

_____

_____

2. Care a fost şi cum a fost argumentată opinia grupului prezentator în cadrul dezbaterii?

_____

_____

_____

3, Ce sugetii aveţi pentru echipa prezentatoare astfel încât să-şi îmbunătăţească aplicaţia?

_____

4. Care este părerea voastră despre prezentarea proiectului? Explicaţi.

6. Concluzii

6.1. Posibilitatea predării limbajului C# la clasa a XII-a

"Informatica" desemneaza una dintre cele mai recente stiinte (termenul provine din Franta si este o combinatie intre cuvantul informatie si cuvantul automatica)
In vocabularul europenilor cuvantul "informatica" intra in anul 1966. Cu toate ingradile epocii comuniste in 1971, ministri Romaniei din acea vreme infinteaza liceele cu profil informatic.Acesta este momentul in care bazele informatiici sunt puse la punct in cadrul sistemului de invatamant preuniversitar românesc.
Initial ea a derivat din matematica, apoi a capatat o mare amploare cuprinzand diverse ramuri ale stiintei.
In contextul lucrarii de fata ne intereseaza analiza informaticii prin studiul algoritmilor si implementarii lor intr-un limbaj de programare si utilizarea lor in practica sub forma de aplicatii visuale.
Studiul informaticii in liceu respecta structura:

- în clasa a IX-a se studiază algoritmii elementari pentru specializarea matematica-informatica, iar la matematica-informatica intensiv se adauga implementarea lor într-un limbaj de programare;
- în clasa a X-a se insistă în special pe chestiuni mai strâns legate de limbajul de programare
- în clasa a XI-a se introduc câteva dintre metodele principale de programare (Backtracking, Divide et Impera, Alocare Dinamică şi Structuri de Date, Teoria Grafurilor)

Pana in anul 2007 programa scolara corespunzatoare clasei a XII-a prevedea studiul bazelor de date in FoxPro. Limbaj care a fost apreciat pana in 2000 cand evolutia a propulsat alte instrumente mai atractive.
In prezent disciplina Informatică – clasa a XII – a, specializarea matematică – informatică

(OM5959_Programa) este prevăzuta uramtoarea alocare de timp:

- ❖ pentru specializarea matematică-informatică: 4 ore/ săptămână, din care o oră pentru activităţi teoretice şi trei ore pentru activităţi practice;
- ❖ pentru specializarea matematică-informatică, intensiv informatică: 7 ore/ săptămână, din care două ore pentru activităţi teoretice şi cinci ore pentru activităţi practice.

Variante de studiu pentru specializarea matematică-informatică:

I. Baze de date (1 oră de teorie) + Sisteme de gestiune a bazelor de date (3 ore de activităţi practice).
II. Baze de date (1 oră de teorie) + Programare orientate pe obiecte si programare vizuală (3 ore de
activităţi practice).
III. Baze de date (1 oră de teorie) + Programare web (3 ore de activităţi practice).

Variante de studiu pentru specializarea matematică-informatică, intensiv informatică:
I. Baze de date (1 oră de teorie) + Sisteme de gestiune a bazelor de date (3 ore de activităţi practice)
+ Programare orientată pe obiecte si programare vizuală (1 oră de teorie + 2 ore de activităţi practice).
II. Baze de date (1 oră de teorie) + Programare web (1 oră de teorie si 2 ore de activităŃi practice) +
Programare orientată pe obiecte si programare vizuală (3 ore de activităţi practice).
III. Baze de date (1 oră de teorie) + Programare orientată pe obiecte si programare vizuală (1 oră de teorie
si 2 ore de activităţi practice) + Programare web (3 ore de activităţi practice).

IV. Baze de date (1 oră de teorie) + Sisteme de gestiune a bazelor de date (3 ore de activităţi practice) +

Programare web (1 oră de teorie + 2 ore de activităţi practice).

Studierea modulelor selectate se poate realiza secvenţial sau în paralel.

Conform programei scolare, ce ofera o mare flexibilitate in comparatie cu cea valabila perioadei pana in 2007 alegerea studiului Limbajului C# in cadrul Variantei II de studiu este o oportunitate.

Evolutia spectaculoasa a limbajelor orientate obiect, studiul limbajului C in clasele precedente ofera elevilor usurinta studiului limbajului C#.

Instructiunile pe care le foloseste C# sunt asemanatoare limbajului C.

Introducerea din mediul Visual ii determina pe elevii sa adopte un stil de invatare bazat pe descoperire si investigatie.

Atractivitatea oferita de lucru in mediul visual sporeste creativitatea si realizarea de aplicatii ce modeleaza probleme cotidiene.

Utilizarea bazelor de date create in SQL, ofera o puternica posibilitate de manipulare a datelor.

Limbajul C# contine mai multe facilitati novatoare, dintre care cele mai importante se refera la suportul incorporat pentru componente software. C# dispune de facilitati care implementeaza direct elementele carealcatuiesc componentele software, cum ar fi proprietatile, metodele si evenimentele.

Aşadar, limbajul C# are toate şansele de a fi un limbaj uşor de învăţat de către elevi.

Pentru o invatare mai usoara a limbajului este foarte indicată reluarea problemelor clasice şcolăreşti (de clasa a IX-a, a X-a şi a XI-a) cu mici precizări (acolo unde se poate, evident) care să ajute la înfrumuseţarea datelor de ieşire (folosirea de ferestre, controale de diverse tipuri)

6.2. Propunere de curs opţional „Programare într-un limbaj vizual”

Planurile - cadru de învăţământ pentru ciclul superior al liceului, filierele teoretică şi vocaţională, prevăd la toate profilurile şi specializările, un numar minim şi un numar maxim de ore pe săptămână pentru CDS.

Schemele orare ale tuturor claselor vor cuprinde, în mod obligatoriu, la toate profilurile şi specializările, cel puţin numărul minim de ore pe săptămână prevăzut pentru CDS în planul-cadru.

Reglementările în vigoare menţionează următoarele tipuri de opţionale:

1. de aprofundare

2. de extindere

3. ca disciplină nouă

4. integrat - la nivelul uneia sau mai multor arii curriculare

5. ca disciplină care apare în trunchiul comun la alte specialăzări

Opţionalul de aprofundare este derivat dintr-o disciplină studiată în trunchiul comun, care urmăreşte aprofundarea competenţelor din curriculumul nucleu prin noi unităţi de conţinut.

Opţionalul de extindere este derivat dintr-o disciplină studiată în trunchiul comun, care urmăreşte extinderea competenţelor generale din curriculumul nucleu prin noi competenţe specifice şi noi conţinuturi.

Opţionalul ca disciplină nouă introduce noi obiecte de studiu, în afara celor prevăzute în trunchiul comun la un anumit profil şi specializare, sau teme noi care nu se regăsesc în programele naţionale.

Opţionalul integrat introduce noi discipline structurate în jurul unei teme integratoare pentru o arie curriculară sau pentru mai multe arii curriculare.

Opţionalul ca disciplină care apare în trunchiul comun la alte specializări dispune de o programă deja elaborată la nivel central, ca programă obligatorie în cadrul unei anumite specializări.

Structura programei de opţional este aproximativ aceeaşi cu cea a programelor de trunchi comun elaborate la nivel naţional. În plus apare un Argument care motivează cursul propus, nevoile elevilor, ale comunităţii locale, formarea unor competenţe de transfer.

Elevii manifesta un grad mare de interes in folosirea elementelor visuale ale interfetei din Visual C#.

Un optional drept extindere la clasa a X-a in care algoritmii studiati in clasa a IX-a sunt transpusi intr-un limbaj de programare in mediul Visual, mareste atractivitatea catre domeniul programarii. Construirea unei interfete bazata pe obiecte deja existente ofera elevului o privirea asupra utilitatii activitati pe care o desfasoara.

Interdisciplinaritate poate fi usor de implementat prin abordarea unor probleme de genul:

- grafice matematice
- mici animatii care sa simuleze fenomene fizice
- probleme de logica matematica
- jocuri
- teste din diverse domenii

Prima parte a programei optionalului poate sa trateze facilitatile oferite de mediul visual si o privire comparativa a limbajul C# fata de limbajul C.

7. Bibliografie

1. Troelsen, Andrew, 2008, Pro C#2008 and the. Net 3.5 Platform, 4-th Edition.
2. Gălățan, Constantin, Gălățan, Susana, 2008, C# pentru liceu-Programare in Visual C#2008 Express Edition, Editura L&S Info-mat.
3. Marshall, Donis, 2006, Programming Microsoft Visula C# 2005:The Language, Microsoft Press.
4. Microssoft MSDN Express Libray 2010.
5. Niță, Adrian, Mariana, Niță, Olăroiu, Nicolae, Pintea, Rodica, Sichim, Cristina, Tarasă, Daniela, Tătăran, Mihai (coord.), Lupan, Nușa-Dumitriu (coord.), Jucovschi, Petru (coord.), 2008, Introducere in .Net Framework , Editura Byblos SRL.
6. Wright, Peter, 2006, Beginning Visual C# 2005 Express Edition: From Novice to Professional.
7. Chris, Pappas, William, Murray, 2004, C# pentru programarea Web, editura B.I.C. ALL.
8. Albahari, Ben, 2000, A Comparative Overview of C#, Genamics.
9. Petzold, Charles, 2003, Programare in Windows cu C#, Editura Teora.
10. Ferguson, Jeff, Patterson, Brian, Beres, Jason , Boutquin, Pierre, Gupta, Meeta, 2002, C# Bible, Wiley Publishing Inc..
11. Deitel, C# How to Program.
12. Masalagiu,Cristian, Asiminoaei, Ioan, 2004, Didactica predării informaticii, Editura Polirom.
13. Petre, Carmen, Popa, Daniela, Craciunoiu, Ștefania, Iliescu, Camelia, 2002, Metodica Predării Informatici și Tehnologiei Informației, Editura Arves.

Anexa 1

Operatii de baza in SQL

1. **SELECT**

SELECT este format din doua clauze:

- SELECT [DISTINCT] in cadrul acesteia se specifica numele coloanelor ce trebuie returnate.

Pentru a selecta toate coloanele se poate folosi simbolul asterisc *. Cuvântul cheie DISTINCT

adăugat după cuvântul cheie SELECT elimină rândurile duplicat din rezultatele înregistrării.

- FROM in care sunt specificate tabelele din care extragem .

Exemple:

- SELECT *nume, prenume, functie* FROM *angajati*

  Semnificatie: Selecteaza *nume, prenume, functie* pentru toate inregistrarile din tabelul *angajati*.

- SELECT *nume, salariu* FROM *angajat* WHERE *salariu* BETWEEN 2000 AND 2500

  Semnificatie: Selecteaza *numele* si *salariu* angajatilor pentru care *salariu* este cuprins intre 2000 si 2500

- SELECT *nume, functie, salariu, n_dept* FROM *angaja*t WHERE *functie = 'director'* OR (*functia = 'secretar'* AND *n_dept = 10*);

Semnificatie: Se afiseaza toti angajati (nume, functie, salariu, departament) din tabelul angajati care au functia director sau secretar in departamentul 10.

2. INSERT

Are drept rezultat inserarea de randuri noi intr-un tabel.

Variante:

- insereaza un singur rand

INSERT INTO nume_tabel [(lista_de_coloane)] VALUES (lista_de_valori);

Observaţie:

- lista de coloane este opţională, in cazul in care exista este incadrata in paranteze.

- cuvântul cheie NULL poate fi folosit în lista de valori pentru specificarea unei valori nule

pentru o coloană