**LICEUL TEORETIC ’’NICOLAE BALCESCU’’**

**PROIECT PENTRU OBŢINEREA ATESTĂRII PROFESIONALE ÎN INFORMATICĂ**

TITLUL LUCRĂRII:

**Masinuta**

*Elev:* **Orbai Andrei** – Clasa aXII-a MI2

*Profesor coordonator:* **Muntean Mihaela**

**Cluj-Napoca**

**Mai 2011**

***Cuprins***

1. Introducere
2. Generalităţi despre limbajul C#
3. Cerinţe hardware şi software
4. Structura şi conţinutul proiectului
5. Bibliografie

**Introducere**

Proiectul consta intr-o mica demonstratie a regulilor de circulatie si a sanctionarii acestora.

In Form Design Grafica-Masina(Meniu) au fost utilizate Labeluri (pentru informatii), Butoane (**Porneste, Oprire, Marsalier, Pilot Automat, Exit, Deseneaza**) care indeplinesc mai multe functii, 2 Timere si o imagine de fond.

****

**Generalitati despre Microsoft Visual Studio (C#)**

**Scurt istoric**

Lansat publicului in iunie 2000 si oficial in primavara anului 2002, C# este un limbaj de programare care combina facilitati testate de-a lungul timpului cu inovatii de ultim moment. Creatorii acestui limbaj au fost o echipa de la firma Microsoft condusa de Anders Hejlsberg. Desi limbajul este creat de Microsoft, acesta nu este destinat doar platformelor Microsoft.

Compilatoare C# exista si pentru alte sisteme precum Linux sau Macintosh. Creat ca instrument de dezvoltare pentru arhitectura .NET, limbajul ofera o modalitate facila si eficienta de a scrie programe pentru sistemul Windows, internet, componente software etc. C# deriva din doua dintre cele mai de succes limbaje de programare: C si C++. De asemenea, limbajul este o “ruda” apropiata a limbajului Java. Pentru o mai buna intelegere a limbajului C# este interesant de remarcat care este natura relatiilor acestuia cu celelalte trei limbaje mentionate mai sus. Pentru aceasta, vom plasa mai intai limbajul C# in contextual istoric determinat de cele trei limbaje.

**Limbajul C#**

Desi Java a rezolvat cu succes problema portabilitatii, exista unele aspecte care ii lipsesc. Una dintre acestea este interoperabilitatea limbajelor diferite, sau programarea in limbaj mixt (posibilitatea codului scris intr-un limbaj de a lucra in mod natural cu codul scris in alt limbaj). Interoperabilitatea limbajelor diferite este esentiala la cearea sistemelor software de dimensiuni mari.

Ca parte a ansamblului strategiei .NET, dezvoltata de Microsoft, la finele anilor ‟90 a fost creat limbajul C#. C# este direct inrudit cu C, C++ si Java. “Bunicul” limbajului C# este C-ul. De la C, C# mosteneste sintaxa, multe din cuvintele cheie si operatorii. De asemenea, C# construieste peste modelul de obiecte definit in in C++. Relatia dintre C# si Java este mai complicata. Java deriva la randul sau din C si C++. Ca si Java, C# a fost proiectat pentru a produce cod portabil. Limbajul C# nu deriva din Java. Intre C# si Java exista o relatie similara celei dintre “veri”, ele deriva din acelasi stramos, dar deosebinduse prin multe caracteristici importante.

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 care alcatuiesc componentele software, cum ar fi proprietatile, metodele si evenimentele. Poate cea mai importanta facilitate de care dispune C# este posibilitatea de a lucra intr-un mediu cu limbaj mixt.

**Principiile programarii orientate obiect**

Metodologiile de programare s-au modificat continuu de la aparitia calculatoarelor pentru a tine pasul cu marirea complexitatii programelor. Pentru primele calculatoare programarea se facea introducand instructiunile masina scrise in binar. Pe masura ce programele au crescut s-a inventat limbajul de asamblare, in care se puteau gestiona programe mai mari prin utilizarea unor reprezentari simbolice ale instructiunilor masina.

Cum programele continuau sa creasca, s-au introdus limbaje de nivel inalt, precum FORTRAN si COBOL, iar apoi s-a inventat programarea structurata. POO a preluat cele mai bune idei de la programarea structurata, combinanu-le cu concepte noi. A rezultat o modlitate diferita de a organiza un program. In fapt, un program poate fi organizat in doua moduri: in jurul codului (mod de lucru descris de sintagma “codul actioneaza asupra datelor”, valabil in cazul programarii structurate) sau in jurul datelor (abordare descrisa de sintagma “datele controleaza accesul la cod”, valabila in cazul programarii orientate obiect).

Toate limbajele POO au patru caracteristici comune: incapsularea, polimorfismul, mostenirea si reutilizarea.

**Incapsularea**

Incapsularea este un mecanism care combina codul si datele pe care le manipuleaza, mentinand integritatea acestora fata de interferenta cu lumea exterioara. Incapsularea mai este numita si realizarea de cutii negre, intrucat se ascunde functionalitatea proceselor. Cand codul si datele sunt incapsulate se creaza un obiect. In cadrul unui obiect, codul si datele pot fi publice sau private. Codul si datele private sunt accesibile doar in cadrul aceluiasi obiect, in timp ce codul si datele publice pot fi utilizate si din parti ale programului care exista in afara acelui obiect.

Unitatea fundamentala de incapsulare este clasa. Clasa specifica datele si codul care opereaza asupra datelor. O clasa defineste forma unui obiect. Sau altfel spus, o clasa reprezinta o matrita, iar un obiect reprezinta o instanta a clasei.Polimorfismul

Polimorfismul este calitatea care permite unei interfete sa aiba acces la un grup generic de actiuni. Termenul este derivat dintr-un cuvant grecesc avand semnificatia “cu mai multe forme”. Spre exemplu, sa presupunem ca avem o nevoie de o routina care sa returneze aria unei forme geometrice, care poate fi un triunghi, cerc sau trapez. Intrucat ariile celor trei forme se calculeaza diferit, routina trebuie sa fie adaptata la datele pe care le primeste incat sa distinga despre ce fel de forma este vorba si sa returneze rezultatul corect.

Conceptul de polimorfism este exprimat prin sintagma “o singura interfata mai mlte metode”.

**Mostenirea**

Mostenirea este procesul prin care un obiect poate dobandi caracteristicile altui obiect. Analogia cu conceptul de animal este elocventa. Spre exemplu, sa consideram o reptila. Aceasta are toate caracteristicile unui animal, insa in plus are si o alta caracteristica, si anume: sangele rece. Sa consideram un sarpe. Acesta este o reptila lunga si subtire care nu are picioare. Sarpele are toate caracteristicile unei reptile, insa poseda si propriile sale caracteristici. Asadar, un sarpe mosteneste caracteristicile unei reptile. O reptile mosteneste caracteristicile unui animal. Asadar, mecanismul mostenirii este cel care face posibil ca un obiect sa fie o instanta a unui caz mai general.

**Reutilizarea**

Atunci cand este creata o clasa, aceasta poate fi utilizata pentru a crea o multime de obiecte. Prin utilizarea mostenirii si incapsularii clasa amintita poate fi reutilizata. Nu mai este nevoie sa testam codul respectiv ci doar a il utlizam corect.

**Instructiuni:**

1. **Instructiunea IF**

Forma generala:

if (conditie)

{instructiune1;}

else

{instructiune2;}

-instructiune1 si instructiune2 pot fi orice fel

de instructiuni: de la instructiunea

vida(;),pana la blocuri complexe delimitate

cu {} sau alte instructiuni. In lipsa

delimitarii corespunzatoare,va apare un

mesaj de eroare! Se recomanda ca si o

singura instructiune sa fie delimitata de {}.

-daca expresia conditionala este adevarata se

executa instructiune1, altfel (daca exista)

se executa instructiune2.

-clauza else este optionala. In lipsa ei, daca

conditia s-a evaluat cu false, se trece la

urmatoarea instructiune din program.

1. **Instructiunea FOR**

Forma generala:

for (initializare; conditie; iteratie)

{instructiune}

instructiunea\_urmatoare

Initializarea este de regula o instructiune de atribuire care fixeaza valoarea initial a

variabilei de control al buclei. Conditia este o expresie de tip bool care stabileste

daca bucla continua ciclarea. Expresia iteratie stableste cantitatea cu care variabla

de control al buclei se modifica la fiecare repetare a buclei.

Una, doua sau toate trei dintre expresii pot lipsi, dar cei doi separatori sunt

obligatorii.

**Cerinte hardware si software**

* NET Framework (pentru desktop-ul Windows).:
  + [NET. 2.0](http://translate.googleusercontent.com/translate_c?hl=ro&langpair=en%7Cro&rurl=translate.google.ro&u=http://go.microsoft.com/fwlink/%3FLinkId%3D67329&usg=ALkJrhg8yf39jyVKT-1KqCe-v8V66jnrig)
  + . [NET Framework 3.0](http://translate.googleusercontent.com/translate_c?hl=ro&langpair=en%7Cro&rurl=translate.google.ro&u=http://go.microsoft.com/fwlink/%3FLinkId%3D102445&usg=ALkJrhh809oxlLFvRd3GLiD-BTq-RgNLXg)
  + [NET. 3.5](http://translate.googleusercontent.com/translate_c?hl=ro&langpair=en%7Cro&rurl=translate.google.ro&u=http://go.microsoft.com/fwlink/%3FLinkID%3D110520&usg=ALkJrhhCInlkqDU-ZZADpRYZ1xVEScdtCw)
  + [NET. 4](http://translate.googleusercontent.com/translate_c?hl=ro&langpair=en%7Cro&rurl=translate.google.ro&u=http://go.microsoft.com/fwlink/%3FLinkId%3D185209&usg=ALkJrhgsWz_35nhDkL6R9KKUhcYZjsMnsA)
* . NET Framework Compact (pentru Windows Mobile):
  + [Compact. NET Framework 2.0 SP2](http://translate.googleusercontent.com/translate_c?hl=ro&langpair=en%7Cro&rurl=translate.google.ro&u=http://go.microsoft.com/fwlink/%3FLinkId%3D102447&usg=ALkJrhiNBvMVlKjJbIcK3oi-RcFOW-P5oA)
  + [Compact NET Framework. 3.5](http://translate.googleusercontent.com/translate_c?hl=ro&langpair=en%7Cro&rurl=translate.google.ro&u=http://go.microsoft.com/fwlink/%3FLinkID%3D120927&usg=ALkJrhg-ARSxiSBwNgVgv_PUnskoJqrygQ)
* Microsoft Visual Studio 2008 sau versiuni mai recente.
  + Pentru a dezvolta aplicaţii pentru desktop-ul Windows, utilizarea Visual Studio 2010. Puteţi utiliza, de asemenea, Visual Studio 2008 SP1 pentru a dezvolta aplicaţii desktop pentru SQL Server Compact 3.5 SP2, cu toate acestea, Visual Studio 2010 oferă caracteristici noi, cum ar fi suportul pentru Transact-SQL Query Editor care nu sunt disponibile cu Visual Studio 2008 SP1.
  + Pentru a dezvolta aplicaţii pentru Windows Mobile, utilizarea Visual Studio 2008 SP1. Visual Studio 2010 nu are suport pentru dezvoltarea de aplicaţii mobile pentru SQL Server Compact 3.5.

Nu aveţi nevoie să instalaţi NET Framework separat. Dacă utilizaţi Visual Studio ca mediu de dezvoltare. Visual Studio instalează. NET Framework. Pentru informaţii despre cerinţele de sistem, a se vedea Visual Studio Books Online.

* Microsoft ActiveSync 4.0 sau versiuni mai recente.
* SQL Server 2008 R2 sau versiuni mai recente
* SQL Server 2008 sau versiuni mai recente
* SQL Server 2005 sau versiuni mai recente

Intel sau compatibil Pentium 600 MHz sau un procesor mai rapid (viteza procesorului recomandată este de 1 GHz sau mai rapid), 256 MB RAM minim (recomandat 512 MB RAM sau mai mult), spaţiu pe hard disk 250 MB

Pentru informaţii despre cerinţele de sistem de operare, consultaţi "Cerinţe hardware şi software pentru instalarea SQL Server 2008 R2" în SQL Server 2008 R2 Books Online.

* IIS 5.x sau versiuni mai recente

Suportate pe Windows Server 2003, Windows 2000 SP4 sau versiuni mai târziu, şi Windows XP.

120 MB de spatiu pe disc disponibil pe server.

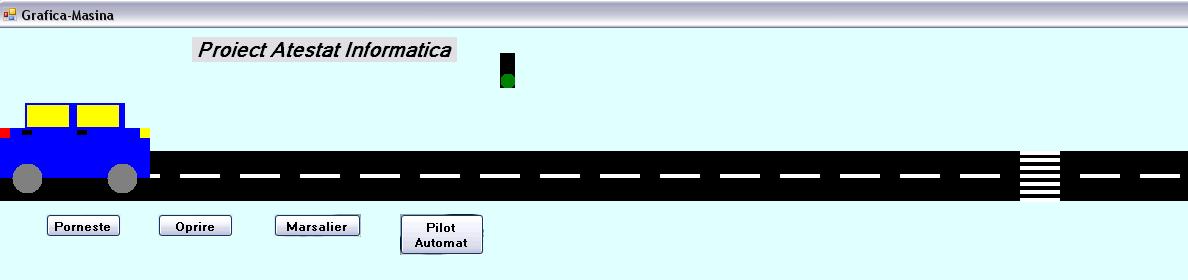
Pentru IIS 7.0, IIS 6.0 înapoi componente compatibilitate trebuie să fie instalat.

* Microsoft ActiveSync 4.0 sau versiuni mai recente

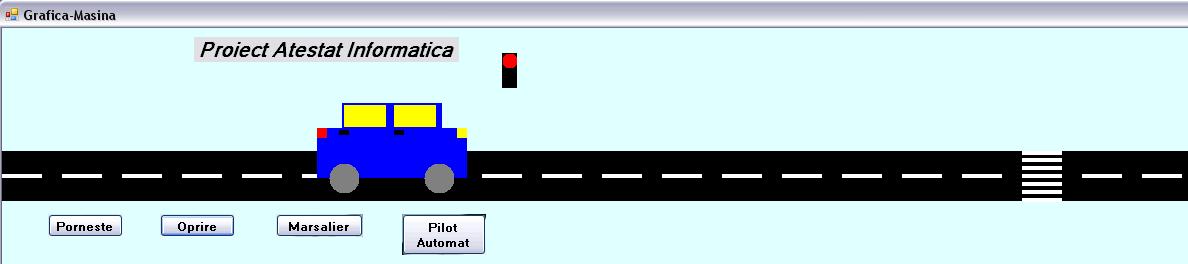
Necesară atunci când utilizaţi SQL Server 2008 R2 Management Studio pentru a gestiona baze de date SQL Server Compact pe dispozitivele conectate

Structura şi conţinutul proiectului

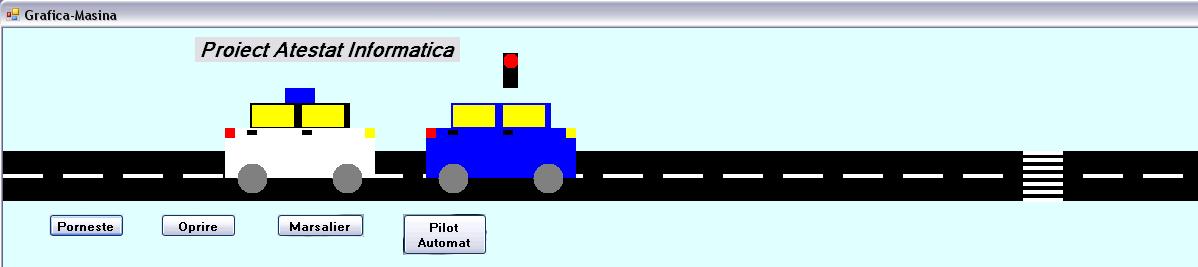
La rulare (**Deseneaza**) programul va desena masina albastra aflata pe un drum public. Pentru pornirea masinii se va apasa butonul **Porneste**.



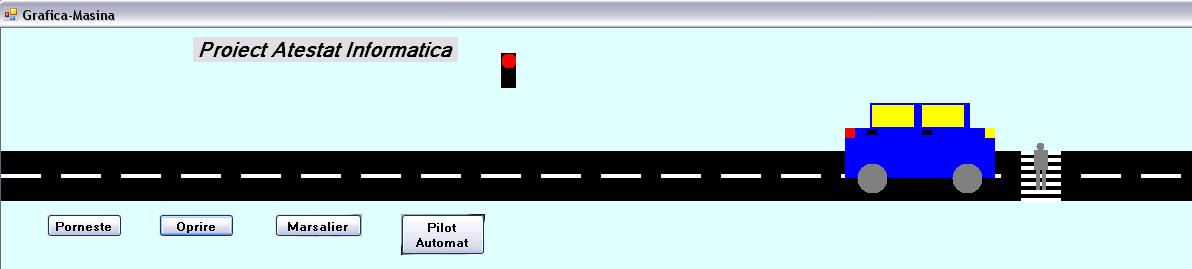
In dreptul semaforului daca este de culoarea rosie sunteti nevoit sa opriti inaintea acestuia, apasand butonul **Oprire**.



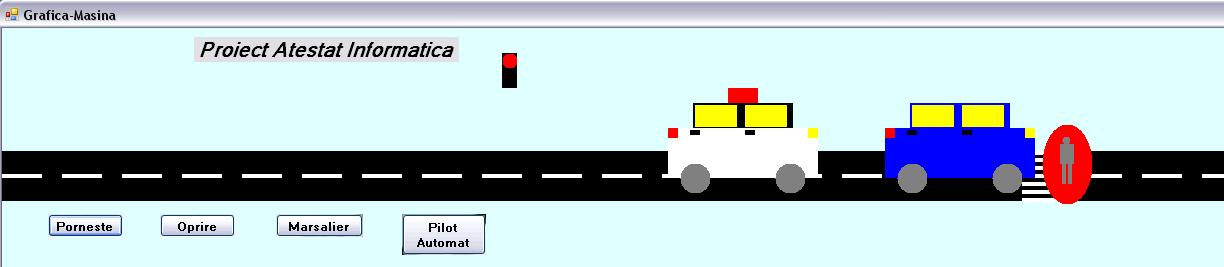
In cazul in care treceti pe rosu ve-ti fi oprit de politie.



Ajunsi in dreptul trecerii de pietoni v-om fi nevoiti sa acordam prioritate de trecere pietonilor.



In cazul in care lovim pietonul pe trecere v-a venii politia si ne v-a opri.



Pentru butonul **Pilot Automat** masina v-a actiona singura la oprirea pe culoarea rosie a semaforului, respectiv pornire pe culoarea verde si oprire la trecerea de pietoni. Este necesara apasarea butonului **Pornire** la inceputul rularii, iar apoi apasarea butonului **Pilot Automat** pentru aceasta functie. Pentru mersul cu spatele se va apasa butonul **Marsalier**.

**Codul Sursa al proiectului**

Declaratiile de la inceputul programului: initializarea valorilor intregi si a celor booleane(la inceput fiind initializate cu false).

public partial class Form1 : Form

{

int x, y, d, s, p, h, j, k;

bool pornire = false;

bool marsalier = false;

bool politie = false;

bool pilot = false;

In cadrul butonului **Deseneaza** am initializat cele 2 timeruri cu true.

private void button1\_Click(object sender, EventArgs e)

{

timer1.Enabled = true;

timer2.Enabled = true;

In butonul **Pornire** am initializat marsalier cu false, iar pornire cu true.

private void button3\_Click(object sender, EventArgs e)

{

marsalier = false;

pornire = true;

In butonul **Oprire** am initializat marsalier cu false si pornire cu false.

private void button4\_Click(object sender, EventArgs e)

{

pornire = false;

marsalier = false;

In butonul **Marsalier** am initializat pornire cu false iar marsalier cu true.

private void button5\_Click(object sender, EventArgs e)

{

pornire = false;

marsalier = true;

In butonul **Pilot Automat** am initializat pilot cu valoarea true.

private void button6\_Click(object sender, EventArgs e)

{

pilot = true;

**Timer 2**:

private void timer2\_Tick(object sender, EventArgs e)

{

y++;

p++;

**Timer 2:**

private void timer1\_Tick(object sender, EventArgs e)

{

Graphics g = this.CreateGraphics();

Brush b = new SolidBrush(Color.Black);

Brush b6 = new SolidBrush(Color.Gray);

Brush b1 = new SolidBrush(Color.Blue);

Brush b2 = new SolidBrush(Color.Yellow);

Brush b3 = new SolidBrush(Color.Red);

Brush b4 = new SolidBrush(Color.Coral);

Brush b5 = new SolidBrush(Color.Green);

Brush b7 = new SolidBrush(Color.White);

g.Clear(Color.LightCyan);

if (pornire == true) x++;

if (marsalier == true) x--;

g.FillRectangle(b, 0, 123, ClientSize.Width, 50);

for (d = 0; d < 30; d++) g.FillRectangle(b7, d \* 60, 146, 40, 4);

g.FillRectangle(b7, 1020, 138, 40, 4);

g.FillRectangle(b7, 1020, 130, 40, 4);

g.FillRectangle(b7, 1020, 123, 40, 4);

g.FillRectangle(b7, 1020, 154, 40, 4);

g.FillRectangle(b7, 1020, 162, 40, 4);

g.FillRectangle(b7, 1020, 170, 40, 4);

if ((x > 700) && (j<300))

{

j++;

g.FillRectangle(b6, 1035, 80 + j / 5, 10, 20);

g.FillRectangle(b6, 1035, 100 + j / 5, 4, 20);

g.FillRectangle(b6, 1041, 100 + j / 5, 4, 20);

g.FillRectangle(b6, 1040, 78 + j / 5, 1, 2);

g.FillRectangle(b6, 1045, 80 + j / 5, 2, 20);

g.FillRectangle(b6, 1033, 80 + j / 5, 2, 20);

g.FillEllipse(b6, 1035, 72 + j / 5, 8, 8);

k = j;

}

if ((pilot == true) && (x + 150 > 450) && (x + 150 < 500) && (s == 1)) pornire = false;

else if ((pilot == true) && (x + 150 > 450) && (x + 150 < 500) && (s == 2)) pornire = true;

if ((pilot == true) && (x + 150 > 970) && (x + 150 < 1020) && (j < 300)) pornire = false;

else if ((pilot == true) && (x + 150 > 970) && (x + 150 < 1020) && (j == 300)) pornire = true;

if ((x + 150 == 1033) && (k < 300))

{

pornire = false;

j = 400;

g.FillEllipse(b3, 1040, 60 + k / 5, 50, 80);

g.FillRectangle(b6, 1060, 80 + k / 5, 10, 20);

g.FillRectangle(b6, 1060, 100 + k / 5, 4, 20);

g.FillRectangle(b6, 1066, 100 + k / 5, 4, 20);

g.FillRectangle(b6, 1065, 78 + k / 5, 1, 2);

g.FillRectangle(b6, 1070, 80 + k / 5, 2, 20);

g.FillRectangle(b6, 1058, 80 + k / 5, 2, 20);

g.FillEllipse(b6, 1060, 72 + k / 5, 8, 8);

politie = true;

}

if ((x + 150 == 500) && (s == 1)) politie = true;

g.FillRectangle(b1, 0 + x, 100, 150, 50);

g.FillRectangle(b1, 25 + x, 75, 100, 25);

g.FillEllipse(b6, 12 + x, 135, 30, 30);

g.FillEllipse(b6, 107 + x, 135, 30, 30);

g.FillRectangle(b2, 140 + x, 100, 10, 10);

g.FillRectangle(b2, 27 + x, 77, 42, 22);

g.FillRectangle(b2, 77 + x, 77, 42, 22);

g.FillRectangle(b, 22 + x, 102, 10, 5);

g.FillRectangle(b, 77 + x, 102, 10, 5);

g.FillRectangle(b3, 0 + x, 100, 10, 10);

g.FillRectangle(b, 500, 25, 15, 35);

if (politie == true)

{

h++;

g.FillRectangle(b7, 0 + 3 \* h, 100, 150, 50);

g.FillRectangle(b, 25 + 3 \* h, 75, 100, 25);

if (p < 20) g.FillRectangle(b3, 60 + 3 \* h, 60, 30, 15);

else g.FillRectangle(b1, 60 + 3 \* h, 60, 30, 15);

if (p > 40) p = 0;

g.FillEllipse(b6, 12 + 3 \* h, 135, 30, 30);

g.FillEllipse(b6, 107 + 3 \* h, 135, 30, 30);

g.FillRectangle(b2, 140 + 3 \* h, 100, 10, 10);

g.FillRectangle(b2, 27 + 3 \* h, 77, 42, 22);

g.FillRectangle(b2, 77 + 3 \* h, 77, 42, 22);

g.FillRectangle(b, 22 + 3 \* h, 102, 10, 5);

g.FillRectangle(b, 77 + 3 \* h, 102, 10, 5);

g.FillRectangle(b3, 0 + 3 \* h, 100, 10, 10);

if ((h \* 3 + 150 == x - 50) ||

(h \* 3 + 150 == x - 51) ||

(h \* 3 + 150 == x - 52))

{

pornire = false;

marsalier = false;

h--;

//MessageBox.Show("ai trecut pe rosu. carnetul ti-a fost retinut");

}

}

if (y < 200)

{

g.FillEllipse(b3, 500, 25, 15, 15);

s = 1;

}

else

{

g.FillEllipse(b5, 500, 45, 15, 15);

s = 2;

}

if (y == 400) y = 0;

//Random h = new Random();

//Console.Beep(100 + h.Next(500), 100);

Butonul **Exit**( Iesire):

private void button2\_Click(object sender, EventArgs e)

{

this.Close();

Key Up/Key Down:

private void Form1\_KeyDown(object sender, KeyEventArgs e)

{

pornire = false;

private void Form1\_KeyUp(object sender, KeyEventArgs e)

{

pornire = true;

Bibliografie

<http://www.math.uaic.ro/~cgales/csharp/Curs1-11.pdf>

<http://www.google.ro/images>/

<http://ro.wikipedia.org/wiki/C_sharp>