

TUNEA MIHAIELA

CHIRĂ LILIANA

CARDAŞ CERASELA DANIELA

PROGRAMARE VIZUALĂ ÎN LIMBAJUL C# PRIN JOC

Fișe de laborator

Botoșani

2020

ISBN 978-973-0-32570-6

TUNEA MIHAIELA LILIANA CAR

ELA CHIR
CARDAS CERASELA DANIELA

PROGRAMARE VIZUALĂ ÎN LIMBAJUL C# PRIN JOC

Fise de laborator

Autori profesorii:

Cardaş Cerasela Daniela

Carpen Lenuța Manuela

Chira Liliana,

Giocaş Carmen Afrodita,

Principe Mihaela Liliana,

Asandului Andreea Roxana

Tunea Mihaela

ISBN 978-973-0-32570-6

Cuvânt înainte

Căutarea unui limbaj de programare perfect este ținta multor inovatori ai programării. În această domeniu *C# (CSharp)* este unul din cele mai evolute limbi și reprezintă pasul următor în evoluția limbajelor de programare.

Lucrarea de față prezintă aplicații de laborator pentru orele de Programare vizuală, realizate în *Visual Studio Express C# 2019*.

Fiecare fișă de laborator va descrie pașii de realizare a aplicației, codul sursă și imaginea formei finale.

Aplicațiile descrise în această lucrare sunt preluate de pe diverse site-uri de specialitate și adaptate cerințelor și nivelului claselor de liceu. De asemenea au fost incluse materiale didactice ale profesorilor de informatică din Botoșani.

Această lucrare vine în sprijinul profesorilor ce predau disciplina informatică și poate fi un instrument util chiar și celor care vor să învețe *Programarea Visuală cu C#*.

Vă dorim o programare plăcută și interesantă!

Cuprins

Cuprins	3
I. Instalarea aplicației Visual C#.....	5
II. Personalizarea ferestrei de lucru	7
III. Barele de instrumente.....	8
IV. Construirea aplicațiilor.....	12
V. Fișe de laborator	17
Fișa 1- Utilizarea etichetelor, a casetelor de text și a butoanelor	17
Fișa 2- Aplicații cu forme - “Ferestre pop-up”	22
Probleme propuse :	24
Aplicația 1- Password.....	24
Aplicația 2- Cifre ordonate	25
Fișa 3 - Utilizarea controalelor	27
Aplicația 1- Utilizarea controalelor Button, Radio, MessageBox	27
Aplicația 2- Afisarea imaginilor dintr-o listă de imagini	31
Aplicația 3- Utilizarea controlului CheckBox	32
Aplicația 4- Utilizarea controlului ListView.	33
Aplicația 5 – Controlul MonthCalendar	36
Fișa 4 - Utilizarea meniurilor	38
Aplicația 1- Formatarea textului din TextBox sau dintr-un label.....	38
Fișa 5 – Proiecte interdisciplinare	43
Aplicația 1- Convertor de lungimi (informatică-matematică-fizică)....	43
Aplicația 2- Convertor de lungimi (informatică-matematică).....	46
Aplicația 3- MiniCalculator (informatică-matematică).....	47
Aplicația 4- Graficul funcției de gradul II (informatică-matematică) ...	50
Aplicația 5- Calculator molecular (informatică-chimie)	55
Aplicația 6- Tabelul periodic al elementelor(informatică-chimie).....	57
Aplicația 7- Legile fizicii (informatică-fizică)	58
Aplicația 8- Calculul lungimii de undă (informatică-fizică)	64
Aplicația 9- Convertor de temperaturi (informatică-fizică)	65
Aplicația 10- Calculul vârstei (informatică-matematică)	66
Aplicația 11- Calculul vârstei în ani (informatică-matematică)	68

Aplicația 12- Corpuri geometrice (informatică-matematică)	69
Aplicația 13- Ceasuri	77
Aplicația 14- Ceas analogic	78
Aplicația 15- Crearea unui slideshow.....	81
Aplicația 16- Formular de autentificare.....	82
Aplicația 17- Generarea fractalilor	84
Fișa 6 – Jocuri	85
Jocul 1- Tic Tac Toe	85
Jocul 2- Hangman	88
Jocul 3- Battleship	91
Jocul 4- TRex	95
Jocul 5- Flappy Bird Game.....	98
Jocul 6- Memory Game	101
Jocul 7- HANGMAN	105
Jocul 8- Breakout Game	107
Jocul 9- Snaper Game.....	110
Jocul 10- Ferma animalelor	116
Jocul 11- Space Battle Game.....	118
Jocul 12- Save the Eggs Game	122
Jocul 13- Tetris Game.....	129
Joule 14 –Ghicește un număr generat random de computer.....	146
Jocul 15- Ghicește un număr (varianta cu cronometru)	148
Jocul 16-X și 0	150
Bibliografie	152
Webiografie	153

I. Instalarea aplicației Visual C#

Aplicația se instalează de la adresa

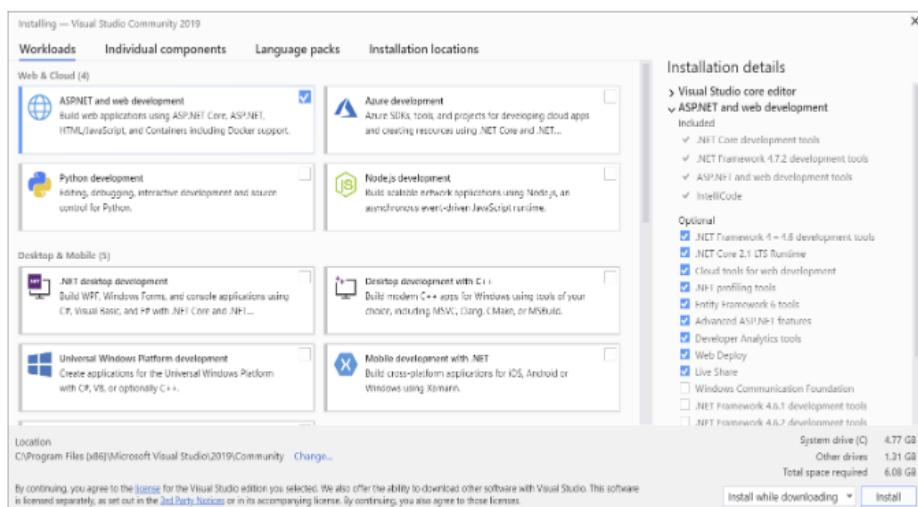
<https://docs.microsoft.com/en-us/visualstudio/install/install-visual-studio?view=vs-2019> urmând pașii indicați.

Pasul 1 – Înainte de a începe instalarea Visual Studio, verificați dacă specificațiile computerului dumneavoastră sunt compatibile cu acest program.

Pasul 2 - Descărcați Visual Studio - fișierul bootstrapper Visual Studio.

Pasul 3 – Instalați programul Visual Studio executând click pe fișierul bootstrapper Visual Studio Installer.

Pasul 4 – Selectați toate aplicațiile referitoare la C#.



Nu uitați să bifați și instalarea NetFramework-ului ce va permite lucrul cu platforma VisualStudio. Optional, puteți adăuga și componenta „...” pentru a putea include în proiecte și bazele de date.

Pasul 5 - Alegeti componente individuale (optional)

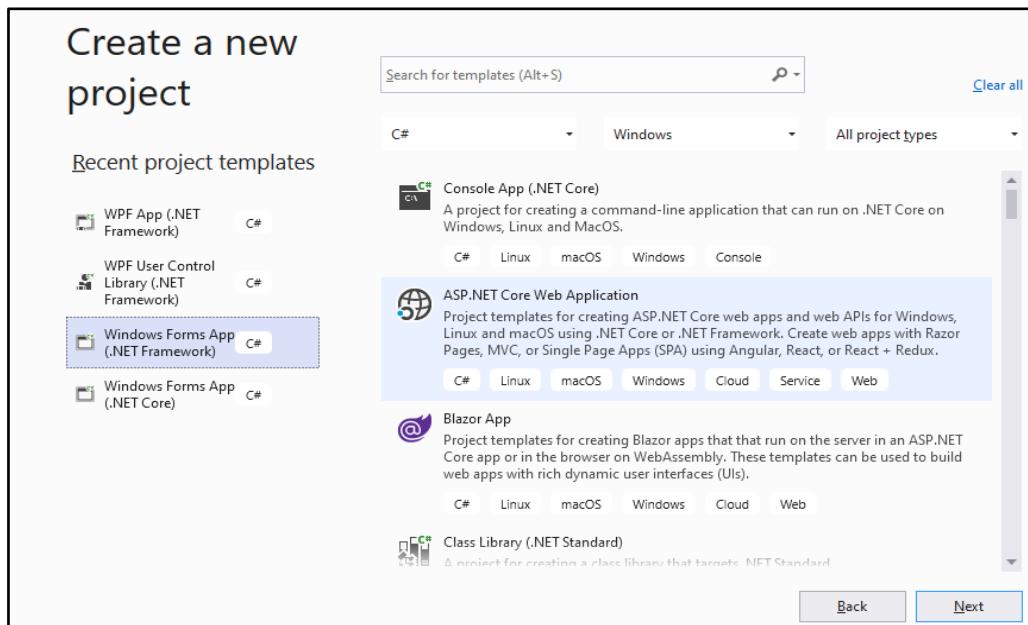
Pasul 6 - Instalarea pachetelor de limbi (optional), implicit aplicația este în engleză.

Pasul 7 - Selectați locația de instalare (optional), implicit se va instala pe partitura de Windows.

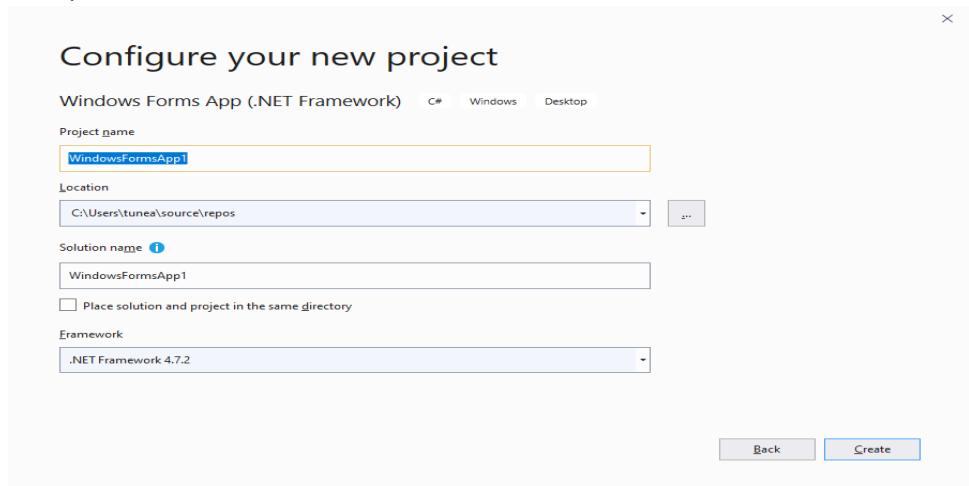
După finalizarea instalării Visual Studio, alegeti butonul Lansare, pentru a începe să lucrați cu Visual Studio.

În fereastra de pornire, alegeti „Create a New Project”.

Selectați tipul de aplicație pe care dorîți să o creați din lista de şablonă disponibile, în cazul nostru Windows Forms APP.

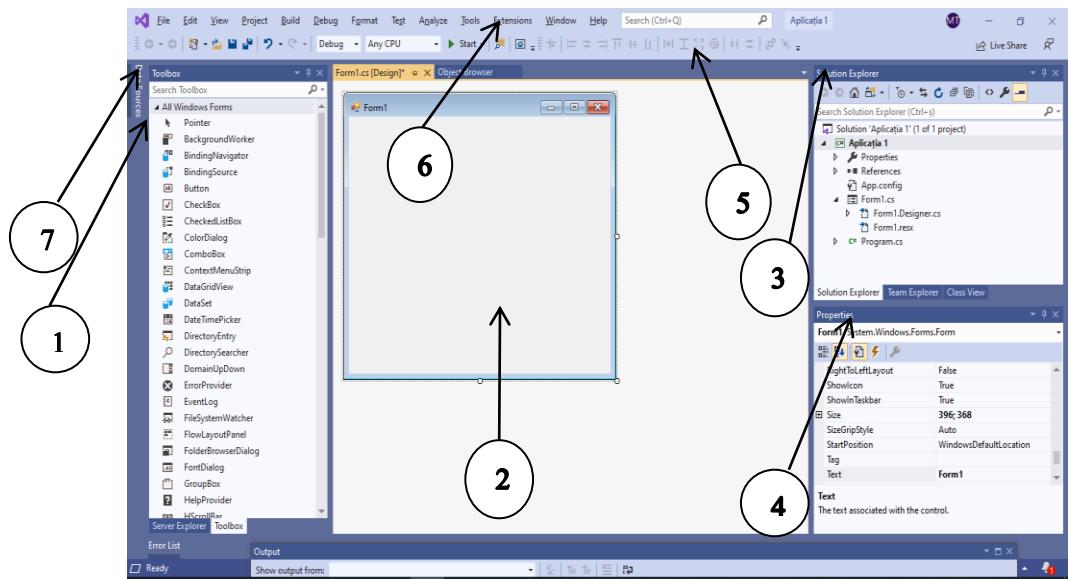


După accesarea butonului NEXT se va deschide fereastra pentru datele de identificare a proiectului (Numele proiectului, locația de salvare, etc) după care accesați butonul Create.



Visual Studio îți deschide noul proiect și ești gata de codare!

II. Personalizarea ferestrei de lucru



- 1- Fereastra Toolbox - Instrumentele de lucru**
- 2- Fereastra Windows Forms - Spațiul de lucru**
- 3- Fereastra Solution Explorer - Locul de unde se selectează modul de vizualizare al aplicației, codul sursă sau design dar și locul în care se pot adăuga mai ușor noi elemente (imagini, video sau sunet, clase, etc).**
- 4- Fereastra Properties – folosită pentru modificarea proprietăților obiectului selectat.**
- 5- Bara de unelte**
- 6- Bara de meniuri**
- 7- Toate ferestrele au în partea dreaptă o piuneză (**Pin**), care poziționată vertical fixează fereastra deschisă. În caz contrar fereastra se închide, retragându-se în partea stângă a mediului de programare.**

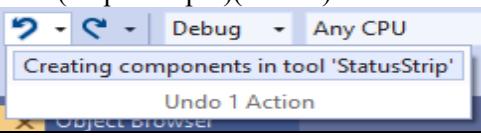
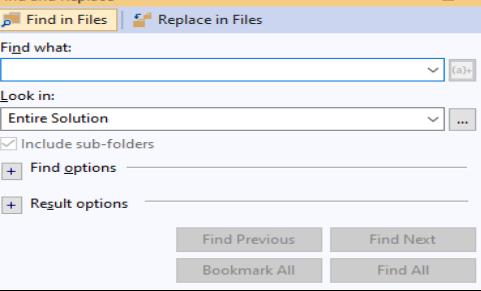
Orice fereastră poate fi aranjată în orice poziție dorită de utilizator, folosind operația drag-and-drop. În acest proces veți fi ghidat de săgețile care apar central și pe margini. De preferat ar fi ca aceste ferestre să rămână în pozițiile lor implicate.

III. Barele de instrumente

Implicit, la crearea unui proiect Windows Form, apare bara de instrumente standard:



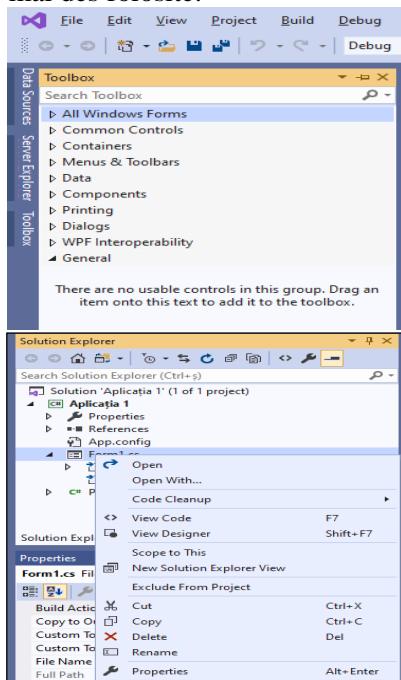
unde:

Icon	Semnificație
	Navigare înapoi sau înainte în cod sau fereastre (Ctrl+-) respectiv(Ctrl+Shift+-)
	Save(Ctrl+S) - Salveaza Forma1
	SaveAll(Ctrl+Shift+O)-Salvează tot proiectul
	Deschide fisierul(Ctrl+O)
	Proiect nou-(Ctrl+Shift+A)
	Cut
	Copy
	Paste
	Undo(un pas înapoi)(Ctrl+Z) 
	Redo (un pas înainte)(Ctrl+Y)
	Start debugging (F5) Compilează proiectul și îl lansează în execuție
	Find –Căutare și înlocuire(Ctrl+Shift+F) 
	Strat windows - Deschide prima fereastră de strat a aplicație Visual Studio

Barele de instrumente se folosesc atunci când dorim să acționăm asupra mai multor controale din fereastra noastră, și anume pentru: alinieri, spațieri, redimensionări, aducerea în față/spate a unora dintre controalele existente. Barele de instrumente se afișează acționând săgeată cu vârful în jos de la capătul celei existente  . Icon-urile aflate pe această bară sunt deosebit de sugestive pentru acțiunea pe care o realizează. De asemenea barele de instrumente se pot deplasa acționând capătul stâng al ei  și prin operația drag-and-drop se poziționează fie un lungul altă bare fie dedesubt sau deasupra.

În **Fereastra Toolbox** putem să deschidem una dintre opțiunile din fereastră apăsând semnul plus din față. De exemplu, dacă deschidem **All windows Forms** în fereastră apar toate tipurile de controale.

Dacă se selectează **Common Controls** în fereastră apar controalele cele mai des folosite.



Fereastra **Properties**, conține atât **proprietățile** cât și **evenimentele** atașate controalelor. Proprietățile controalelor, sunt moștenite sau suprascrise din clasa de bază **Control**. Proprietățile comune controalelor, proprietăți furnizate de către clasa **Control**, sunt ordonate alfabetic în fereastra **Properties**. Cele mai utilizate sunt prezentate în următorul tabel.

Orice control poate fi adus pe Formă prin operația drag-and-drop sau dublu click pe respectivul controller.

Fereastra **Solution Explorer**, din partea dreaptă se referă, printre altele la fereastra **Designer** sau la fereastra în care utilizatorul va scrie propriul cod.

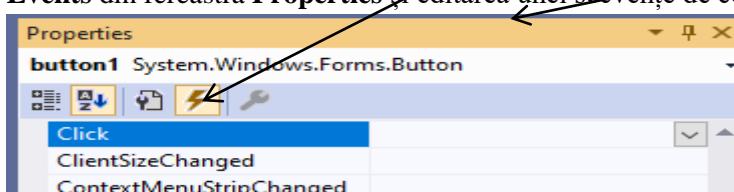
În cazul în care fereastra **Designer** este închisă, putem apela la opțiunea **Open** și aceasta va reapărea în fereastra centrală. Dacă dorim să vedem codul, apăsăm pe opțiunea **View Code**.

Dacă dorim să vizualizăm forma apăsăm pe opțiunea **View Designer**.

Proprietate	Descrierea proprietății
AutoSize	Stabilește dimensionarea automată a obiectului selectat (True/False)
AutoSizeMode	Permite creșterea automată a dimensiunii obiectului selectat (Grow Only) sau a creșterii și micșorării obiectului selectat selectat (Grow and Shrink)
BackColor	Stabilește culoarea de fundal a obiectului selectat
BackgroundImage	Permite stabilirea unei imagini de fundal pentru o fereastră, buton etc

Dock	Pozitionează obiectul(controlul) selectat la una dintre marginile ferestrei
Enabled	Permite obiectului selectat să i se atașeze un eveniment creat de utilizator
FlatAppeare	Obiectului selectat primește bordură colorată și de grosimea dorită. De asemenea poate fi activată schimbarea culorii la trecerea mouse-ului peste controler dând un efect dinamic.
Font	Stabilește tipul, dimensiunea fontului
ForeColor	Stabilește culorii textului
Image	Asociază o imagine pentru un obiect. De exemplu butonului i se poate asocia un personaj într-un joc.
Height	Stabilește înălțimea controlului selectat
Left	Stabilește distanța dintre marginea stângă a ferestrei până la obiect
Location	Stabilește locația obiectului pe fereastră
Name	Stabilește numele controler-ului pentru a fi vizibil în cod
Right	Stabilește distanța dintre marginea dreaptă a ferestrei până la obiect
Size	Stabilește dimensiunea obiectului
TabIndex	În funcție de valoarea introdusă se stabilește ordinea activării obiectului la apăsarea tastei TAB
TabStop	Stabilește dacă obiectul poate fi activat la apăsarea tastei TAB(True/False)
Text	Stabilește numele obiectului în fereastră. Se va insera numele ferestrei, a butonului, a GroupBox-ului sau se va introduce un text într-un label.
TextAlign	Stabilește alinierarea textului pe controler-ului pe verticală și pe orizontală.
TextImageRelation	Stabilește cum să “îmbrace” imagine controlul de exemplu Overlay
Visible	Stabilește vizibilitatea obiectului existent pe fereastră (True/False)
Width	Stabilește lățimea obiectului

Dinamica aplicațiilor vizuale se realizează prin aplicarea controlerului selectat acțiuni. De exemplu la acționarea tastei stângi al mouse-ului, personajul din fereastră să se deplaseze la stânga. Aceste evenimente se activează în fereastra **Events** din fereastra **Properties** și editarea unei sevențe de cod.



Clasa Control implementează o serie de evenimente predefinite. Generarea evenimentului se face dând dublu click pe evenimentul selectat. Automat se va crea o funcție în codul sursă. Ea va trebui completată cu codul specific.

În tabelul de mai jos sunt prezentate evenimentele predefinite ce se pot aplica obiectelor unei formei.

Eveniment	Descriere
Click	Evenimentul se activează dând click pe control
DragDrop	Evenimentul se activează la tragerea obiectului în fereastră
DoubleClick	Evenimentul se activează dând dublu click pe control
DragEnter	Evenimentul se activează prin operația drag-and-drop iar obiectul este pe control
DragLeave	Evenimentul se activează prin operația drag-and-drop iar obiectuliese din control
Enter	Evenimentul se activează prin apăsarea tastei Enter
KeyDown	Evenimentul se activează prin apăsarea unei taste stabilite de utilizator iar controlul e va deplasa în sus pe suprafață de formei. Se furnizează codul ASCII al tastei respective.
KeyPress	Evenimentul se activează prin apăsarea unei taste stabilite de utilizator pentru controlul specificat. Se furnizează codul ASCII al tastei respective.
KeyUp	Evenimentul se activează prin apăsarea unei taste stabilite de utilizator iar controlul se va deplasa în jos pe suprafață formei. Se furnizează codul ASCII al tastei respective.
MouseClick	Evenimentul se activează la click stânga al mouse-ului atunci când cursorul mouse-ului este pe obiect
MouseDown	Evenimentul se activează atunci când cursorul mouse-ului este pe obiect și se apasă un buton al mouse-ului
MouseUp	Evenimentul se activează atunci când se trece cu cursorul mouse-ului peste obiect și se eliberează butonul mouse-ului
MouseMove	Evenimentul se activează atunci când se trece cu cursorul mouse-ului peste obiect
Paint	Se generează evenimentul de desenare a obiectului
Validated	Se generează evenimentul, atunci când obiectul este activ. Se confirmă validarea obiectului
VisibleChanged	Generează vizibilitatea obiectului.

IV. Construirea aplicațiilor

A. Forme

O interfață grafică cu utilizatorul, GUI (Graphical User Interface), permite utilizatorului să interacționeze cu aplicațiile în mod vizual.

Un exemplu de GUI este o fereastră Internet Explorer care conține o *bară de meniuri* inclusiv opțiunile **File**, **Edit**, **View**, **Favorites**, **Tools** și **Help** sau un formular pentru completarea unei comenzi pe internet.

Interfețele GUI sunt construite cu ajutorul componentelor GUI numite și **controale**. O componentă GUI este un obiect cu care user-ul interacționează prin intermediul mouse-ului sau a tastaturii.

Câteva din controalele GUI uzuale sunt prezentate în continuare:

- **Label** - o zonă în care se afizează un text sau un icon needitabil
- **TextBox** - o zonă în care user-ul introduce date cu ajutorul tastaturii sau în care sunt afișate diferite informații
- **Button** - o zonă care declanșează un eveniment când este acționat
- **CheckBox** - un control GUI care poate fi selectat sau neselectat
- **ComboBox** - o listă derulantă (drop-down) din care user-ul poate selecta un item cu mouse-ul sau prin tastare într-o căsuță, dacă este permis
- **ListBox** - o zonă în care este afișată o listă de itemi din care user-ul poate selecta unul sau mai mulți făcând click cu mouse-ul
- **Panel** - un container în care pot fi plasate controale
- **ScrollBar** - bară derulantă

Window Forms & Formulare Windows

Formularele creează interfață GUI pentru aplicații. Un formular este un element grafic care apare pe desktop. Un formular poate fi un dialog, o fereastră **SDI** (Single Document Interface) sau o fereastră **MDI** (Multiple Document Interface).

Pentru unu formular stabilim proprietățile acestuia, adăugăm controale, setăm proprietățile acestora și le aplicăm evenimente (event handler).

Principalele proprietăți ale unui formular sunt:

- **AcceptButton** - ce buton va fi acționat când se apasă tasta Enter
- **AutoScroll** - apar bare de derulare (dacă este necesar pentru vizualizarea conținutului)
- **CancelButton** - ce buton va fi acționat dacă se apasă tasta Escape
- **FormBorderStyle** - chenarul formularului (none, single, 3D, sizeable)
- **Font** - fontul textului afișat în formular și, implicit, al controalelor ce vor fi adăugate formularului

Text - textul afișat în bara de titlu a formularului

Metodele uzuale ale unui formular sunt:

- **Close** - închide formularul și eliberează resursele alocate. Un formular închis nu poate fi redeschis.
- **Hide** - ascunde formularul
- **Show** - afișează formularul

Evenimente tipice pentru un formular:

- **Load** - intervine după ce formularul este afișat. Visual C# generează un event handler implicit când programatorul face dublu click pe formular în modul design.

Proprietățile, cel mai des aplicate, unui formular sunt:

- **Name** - Numele formularului pentru codul sursă,
- **Size** - Dimensionarea formularului ,
- **Text** -Titlul formularului,
- **BackColor** - Backgroundul formei () ,
- **Font** - Setarea fontului: tipul de font, dimensiune
- **ForeColor** - Culoarea textului controalelor inserate ulterior, dar la care vor fi aplicate noi proprietăți.

Prin crearea spațiului Form, proprietățile sale sunt moștenite de clasele derivate, obiectelor adăugate în ea ca de exemplu alte ferestre sau formulare (System.Windows.Forms.Form), controale: butoane (System.Windows.Forms.Button), casete de text (System.Windows.Forms.TextBox) etc.

Sunt moștenite, atributele vizuale ale ferestrei (stilul marginilor, culoare de fundal, etc.), metode care implementează anumite comportamente (*Show*, *Hide*, *Focus* etc.) și o serie de metode specifice (*handle*re) de tratare a evenimentelor (*Load*, *Click* etc.).

O formă poate fi activată cu *form.Show()* sau cu *form.ShowDialog()*, metoda a doua permite ca revenirea în forma anterioară, din care a fost activat noul formular să se facă numai după ce noul formular a fost închis.

Vizibilitatea unui formular poate fi setată folosind metodele *Hide* sau *Show*. Pentru a ascunde o formă putem folosi:

```
this.Hide(); //setarea vizibilității directe
this.Visible = false; //setarea vizibilității indirekte
```

Alte setări posibile din fereastra **Properties** pentru un **Form**:

- **StartPosition** stabilește poziția apariției ferestrei la rularea aplicației sau la activarea ei. Poziția poate fi setată:
 - **Manual** de către utilizator, care va seta locația cu **WindowsDefaultLocation**
 - Folosind **CenterScreen**, care stabilește ca de Form-ul să fie centrat pe desktop,
 - Folosind **WindowsDefaultLocation** se va stabili dimensiunea inițială și locația pentru forms (WindowsDefaultBounds)
 - Centrat pe forms-ul părinte care l-a afișat (**CenterParent**).
- **Location (X,Y)** reprezintă coordonatele colțului din stânga sus al form-ului relativ la colțul stânga sus al containerului. (Această proprietate e ignorată dacă StartPosition = Manual).
- Schimbarea locației form-ului poate fi tratată în evenimentele **Move** și **LocationChanged**.

Exemplu:

Locația form-ului poate fi stabilită relativ la desktop astfel:

```
private void Form1_Load(object sender, EventArgs e)
{
    this.Location = new Point(1, 1);
    this/DesktopLocation = new Point(1, 1);
}
```

- **Size (Width și Height)** stabilește schimbarea automată dimensiunilor Form-ului setate de utilizator prin evenimentele **Resize** sau **SizeChanged**. Chiar dacă proprietatea **Size** a formularului indică dimensiunea ferestrei.
- **MaximumSize și MinimumSize** restricționează dimensiunile unui Form.
De exemplu:

```
this.MinimumSize = new Size(200, 100);
//înălțimea ferestrei este 100 iar lățimea 200
this.MaximumSize = new Size(int.MaxValue, 300);
//înălțimea ferestrei poate fi oricăr dar lățimea este de 300
```

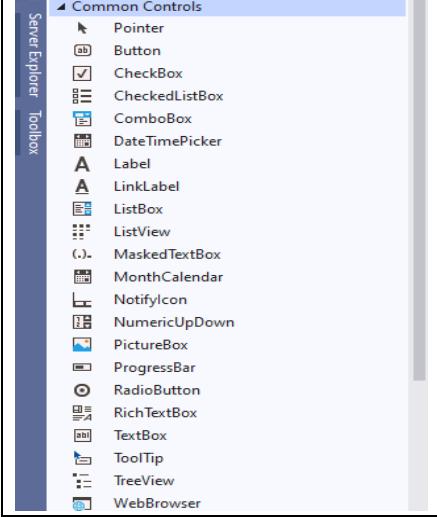
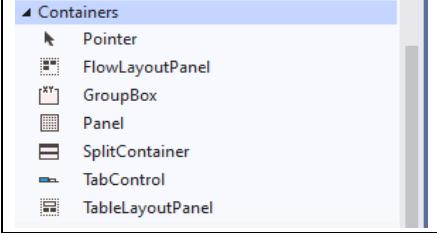
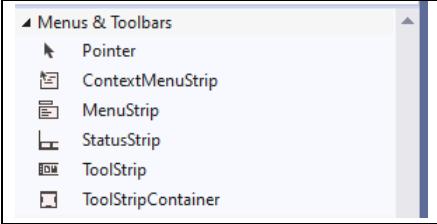
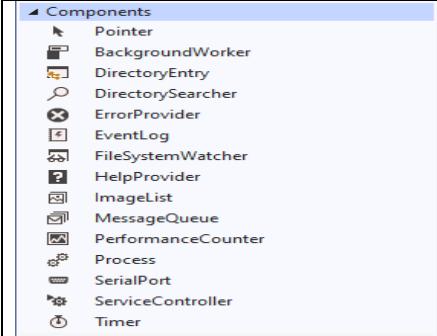
- **ControlBox** specifică dacă fereastra conține un icon buton de închidere al ferestrei și meniul System(Restore,Move,Size,Maximize,Minimize,Close).
- **HelpButton** specifică dacă butonul  va apărea sau nu lângă butonul de închidere al formularului (doar dacă MaximizeBox=false, MinimizeBox=false). Dacă utilizatorul apasă acest buton și apoi apasă oriunde pe formular va se declanșează evenimentul **HelpRequested (F1)**.
- **Icon** permite conectarea la un obiect de tip.icon și va fi folosit ca icon pentru form.
- **MaximizeBox și MinimizeBox** precizează dacă fereastra are sau nu butonul Maximize respectiv Minimize
- **ShowInTaskbar** precizează dacă fereastra apare în TaskBar atunci când form-ul este minimizat.
- **SizeGripStyle** specifică tipul pentru Size Grip care poate fi setat Auto,
Show, Hide și indică faptul că această form-ul poate fi redimensionată. 
- **TopMost** specifică dacă form-ul este afișat în fața tuturor celorlalte form-uri.
- **TransparencyKey** identifică o culoare care va deveni transparentă pe form.

B. Controale

Un Form reprezintă un control.

Un Form poate “găzdui” un container ce poate fi un Form sau un alt control.

Visual Studio .NET are o serie de controale standard, disponibile în **Toolbox** grupate după modul de utilizare:

	<ul style="list-style-type: none"> - Controale cele mai utilizate <ul style="list-style-type: none"> - Button - CheckBox - CheckedListBox - ComboBox - Label - LinkLabel - ListView - MaskedTextBox - MonthCalendar - NotifyIcon - NumericUpDown - PictureBox - ProgressBar - RadioButton - RichTextBox - TextBox - ToolTip - TreeView - WebBrowser
	Containere (găzduiește alte containere sau controale). De aici vom folosi GroupBox
	Meniuri și Toolbars des utilizate în Windows Form Apliction complexe. Vom utiliza MenuStrip .
	Timer -ul va fi folosit mai ales la jocuri.

Odată cu inserarea controlerelor se pot adăuga acestora evenimente ce creează funcții în codul sursă. Evenimentul se va alege din lista de evenimente specifice controlerului respectiv, din fereastra Events, din fereastra Properties, selectând butonul .

Un eveniment va fi generat efectuând dublu click în căsuța respectivă care va genera automat un nume pentru această funcție, ținând cont de numele control-ului și de numele evenimentului (de exemplu button1_Click).

Dacă în Designer efectuăm dublu clic pe un control, se va genera automat o funcție pentru evenimentul asociat controlului (pentru un buton evenimentul implicit este Click, pentru TextBox este TextChanged, pentru un formular este Load etc.).

Evenimentele cele mai des utilizate, ar fi:

- **Load** apare când formularul este pentru prima dată încărcat în memorie.
- **FormClosed** apare când formularul este închis.
- **FormClosing** apare când formularul se va închide ca rezultat al acțiunii utilizatorului asupra butonului Close (Dacă se setează CancelEventArgs.Cancel = True atunci se va opri închiderea formularului).
- **Activated** apare pentru formularul activ.
- **Deactivate** apare atunci când utilizatorul va da click pe alt form-ul aplicației.

Cele mai utilizate **Controale** adăugate în aplicațiile noastre sunt:

Nume control	Funcție control	Descriere
Button	buton	Acționat va realiza o acțiune, un eveniment descris printr-o secvență de instrucțiuni în codul sursă
MonthCalendar	calendar	Inserat într-o formă va afișa un calendarul lunii curente. Poate fi derulat către luni anterioare sau următoare cu ajutorul unor săgeți.
CheckBox	casetă de validare	Folosit pentru acțiuni de validare: Da/Nu sau include/exclude
Label	etichetă	Permite afișarea textelor explicative sau chiar a rezultatul unor instrucțiuni
ListBox	Casetă cu listă	Afișează o listă de articole creată de utilizator
PictureBox	Imagine	Se afișează imagini alese e utilizator
Pointer	Pointer	Este folosit pentru selectarea, mutarea sau redimensionarea unui control
RadioButton	Buton radio	Este utilizat pentru a selecta un singur element dintr-un grup de selecții.
TextBox	Casetă text	Este utilizat pentru afișarea textului generat de aplicație sau pentru a introduce date de către utilizator.

Cotainere

GroupBox	Este folosit în gruparea unor controale ca: listBox, checkBox, PictureBox
Panel	
TabelLayoutPanel	
MenuStrip	Inserează meniuri cu taburi

V. Fișe de laborator

Fișa 1- Utilizarea etichetelor, a casetelor de text și a butoanelor

Etichetele afișează un text read-only. Ele sunt definite în clasa **Label** care derivă din clasa **Control**.

O căsuță de text (**TextBox**) este o zonă în care textul poate fi afișat sau poate fi introdus prin intermediul tastaturii. O căsuță de text de tip **password** ascunde caracterele tastate de user, afișând în loc steluțe. Modificând proprietatea **PasswordChar** pentru o **textbox**, o transformăm într-o căsuță de parole și setăm caracterul ce va fi afișat.

Un **buton** este un control pe care userul îl acționează pentru a declanșa o acțiune specifică. Toate tipurile de butoane derivă din clasa **ButtonBase** din spațiul de nume **System.Windows.Forms**. Textul de pe suprafața unui buton se numește **eticheta butonului (button label)**.

Proprietățile unei **etichete (label)**:

- **Font** – fontul utilizat pentru textul etichetei
- **Text** – textul afișat de etichetă
- **TextAlign** – alinierea textului în cadrul etichetei (trei poziții pe orizontală: **left**, **center**, **right** și trei poziții pe verticală: **top**, **middle**, **bottom**).
- **Visible** – stabilește dacă eticheta este vizibilă (**true**) sau nu (**false**)
- **Anchor** – modul de ancorare în cadrul ferestrei atunci când ea este redimensionată
- **AutoSize** – dacă se redimensionează sau nu automat
- **ForeColor** – culoarea fontului
- **BorderStyle** – tipul chenarului aplicat etichetei

Proprietăți și evenimente pentru **TextBox**:

- **Anchor**, **Font**, **ForeColor**, **BorderStyle**, **Visible** – aceleași ca la toate controalele
- **BackColor** – culoarea fundalului căsuței
- **Multiline** – dacă este **true**, se pot introduce/afișa linii de text încheiate cu Enter
- **PasswordChar** – se stabilește caracterul afișat la introducerea unei parole în căsuță
- **ReadOnly** – dacă este **true**, căsuța este needitabilă
- **Text** – textul ce va fi afișat în interiorul căsuței de text
- **TextChanged** – eveniment (**event**) declanșat când user-ul modifică conținutul căsuței (adaugă sau șterge text)

Observație: Deși textbox-ul și alte controale pot răspunde la click-ul de mouse, ușual butonul este destinat acestui eveniment. Utilizați butoanele pentru a declanșa evenimente.

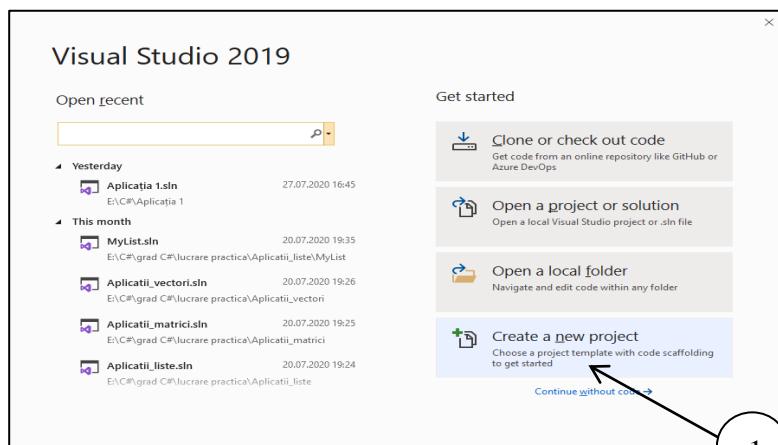
Proprietăți și evenimente specifice butoanelor (button):

- Toate proprietățile de la etichete și textbox cu aceeași semnificații
- Click – eveniment declanșant la acționarea butonului cu mouse-ul. Pentru a scrie event handler-ul asociat, în designer (proiectare) facem dublu click pe buton și se deschide fereastra de cod.

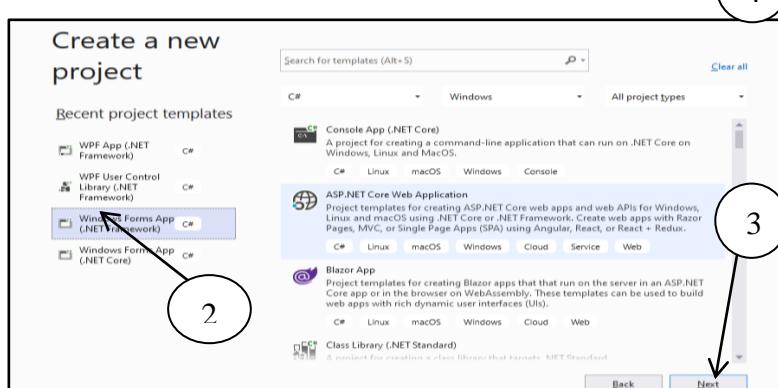
Aplicație

Momentan pentru a înțelege cum funcționează Visual Studio C# va trebui să prelucrăm variabile și probleme asemănătoare cu ceea ce lucram în C++. Propun pentru început rezolvarea problemei: “Afisarea numerelor întregi mai mici decât un număr dat”.

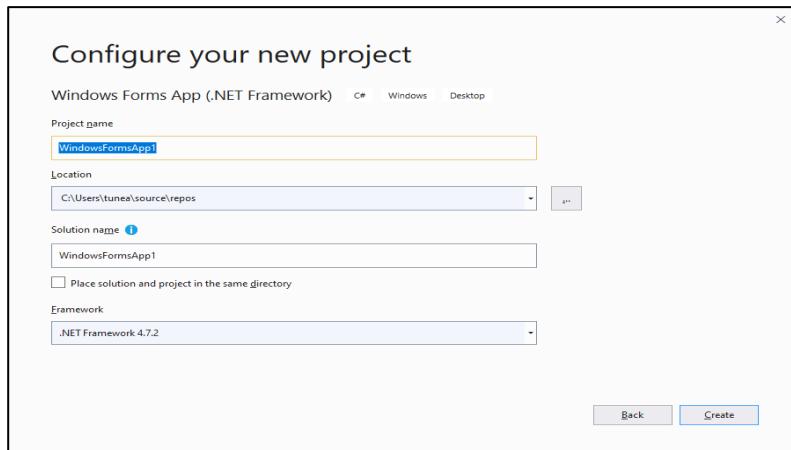
- Deschideți **C# 2019** → Create a new project → „**Windows Forms App(.NET Framework) C#**” → NEXT si în fereastra **Configure your new project** completați formularul:
- **Name project-** textul „**NumereIntregi**”,
- **Location-** Stabiliti locația unde doriti sa fie salvat proiectul.
- Apăsați butonul **Create**.



Pasul 1.



Pasul 2

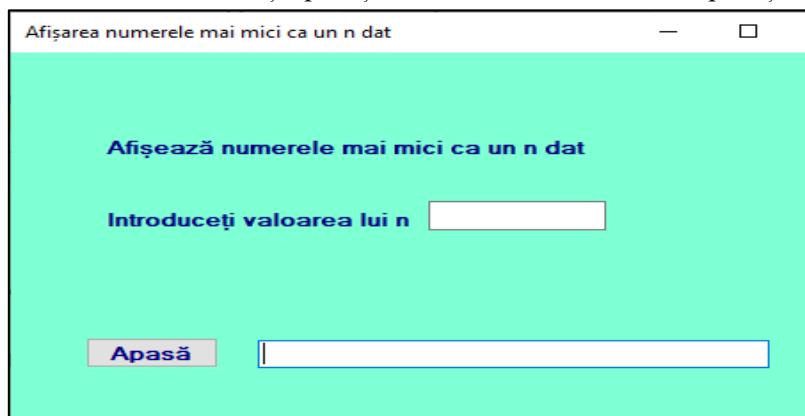


Pasul 3

Pasul 4 executați următoarele operații:

1. Selectați Form1. Din fereastra **Properties** stabiliți următoarele proprietăți pentru forma (dacă fereastra **Properties** nu este vizibilă, alegeti **View→Properties Window** sau F4):
 - **(Name): MyForm**
 - **BackColor: Aquamarine (din tab-ul WEB)**
 - **Font: Microsoft Sans Serif , 9.75 puncte, Bold**
 - **ForeColor: Navy (tabul WEB)**
 - **FormBorderStyle: FixedDialog**
 - **MaximizeBox: False**
 - **MinimizeBox: False** (fereastra nu va putea fi redimensionata de către user)
 - **Size: 500, 400**
 - **Text: Afișarea numerele mai mici ca un n dat**
2. Trageți din **Toolbox** pe forma un control de tip **Label** și setați-i, în fereastra **Properties**, următoarele proprietăți:
 - **(Name): TitleLabel**
 - **BorderStyle: Fixed3D**
 - **Location: 276; 16**
 - **TabIndex:1**
 - **Text: Dați valoarea lui n**
3. Trageți din **Toolbox** pe forma un alt control **Label**. Pentru acesta setați următoarele proprietăți:
 - **(Name): nLabel**
 - **BorderStyle: Fixed3D**
 - **Location: 178; 16**
 - **Text: Introduceti valoarea lui n**

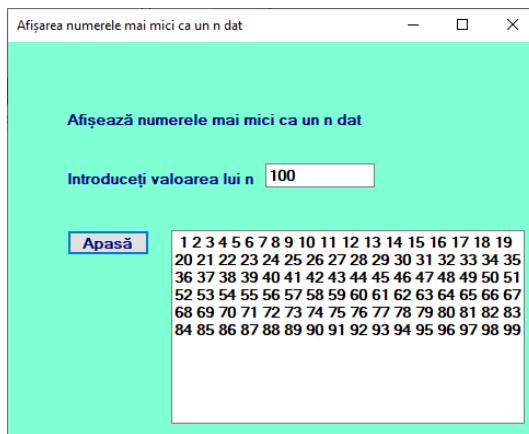
4. Trageți din **Toolbox** pe forma un control **TextBox** apoi setați următoarele proprietăți:
 - **(Name): nTextBox**
 - **LocationL 180, 120**
 - **Size: 100; 22**
5. Trageți pe forma din **Toolbox** un control **Button** apoi setați următoarele proprietăți:
 - **(Name): ApasaButton**
 - **Location: 32, 167**
 - **Size: 90, 20**
 - **Text: Apasă**
6. Trageți din **Toolbox** pe forma un control **Text Box** și setați-i următoarele proprietăți:
 - **(Name): RezultatTextBox**
 - **Location: 288; 21**
 - **ScrollBars: Both**
 - **Size: 370, 20**
 - **TextBoxTask: MultiLine**
7. Selectați **Debug→Start Debugging** sau **F5** sau săgeata verde din bara de instrumente și executați aplicația. Se va deschide fereastra aplicației.



8. Dați dublu click pe butonul “Apasă”. Se va crea un handler(o funcție) în codul sursă, în care completați următorul cod:

```
int n://se declara variabila intreaga n
private void Afiseaza_Click(object sender, EventArgs e)
{
    n = Int32.Parse(nTextBox.Text);
    //ce valoare text introduceti in caseta nTextBox pentru n se convertește la Int32 se declara
    //sirul de caractere s si se initializeaza cu sirul vid
    string s = " ";
    //parcurgem numerele de la 1 la n și le inseram in sirul s cu spatii intre ele
    for (int i = 1; i < n; i++)
        s += i + " ";
    RezultatTextBox.Text = s; //afisam sirul s in RezultatTextBox }
```

Rezultatul îl veți afla după apăsarea tastei F5.



Observații:

Se observă că valoarea intreagă introdusă pentru n în caseta text este de fapt un sir de caractere. Pentru a folosi operatorii aritmetici pentru n va trebui să transformăm sirul de caractere în număr întreg, folosind funcția de conversie Int32.Parse apoi rezultatul prelucrării, adică sirul s , va fi reținut într-un sir de caractere și afișat în caseta text RezultatTextBox.

Probleme propuse :

1. Afisarea numerelor pare până la un n dat
2. Afisarea numerelor divizibile cu 3 și 5 până la un n dat
3. Afisarea numerelor pătrate perfecte până la un n dat
4. Afisarea numerelor prime până la un n dat
5. Afisarea numerelor primelor 2 numere prime mai mici decât un n dat

Fişa 2- Aplicaţii cu forme - “Ferestre pop-up”

Ne propunem să realizăm legătura dintre două ferestre. Fiecare fereastră va avea câte un buton care accesat va realiza un eveniment.

1. Pentru început deschidem un nou proiect cu numele **“Ferestre pop-up”**.
2. Setaţi în modul de vizualizare design proprietăţile pentru Forms1.

Accesaţi fereastra **Properties** şi realizaţi următoarele::

- **(Name): PopForm**
 - **BackColor: Aquamarine (din tab-ul WEB)**
 - **Font: Microsoft Sans Serif , 9.75 puncte, Bold**
 - **ForeColor: Navy (tabul WEB)**
 - **FormBorderStyle: FixedDialog**
 - **Size: 329; 311**
 - **Text: Pop form**
3. Trageţi pe forma din **Toolbox** un control **Button** şi setaţi în fereastra **Properties** următoarele proprietăţi:
 - **(Name): PopButton**
 - **Autosize: True**
 - **Location: 329; 311**
 - **Text: Next**
 4. Alegeti din meniul **Project** opţiunea **Add Form** pentru a adăuga proiectului o nouă formă şi în fereastra de dialog care se deschide faceţi click pe butonul **Add**.
 5. În fereastra **Designer** este deschisa noua forma **Form2**. Faceţi clic pe ea şi în fereastra **Properties** setaţi:
 - **(Name): UpForm**
 - **BackColor: Orange (din tab-ul WEB)**
 - **Font: Microsoft Sans Serif , 9.75 puncte, Bold**
 - **ForeColor: Navy (tabul WEB)**
 - **FormBorderStyle: FixedDialog**
 - **Size: 310, 300**
 - **Text: Up Form**
 6. Trageţi pe forma un control **Button** şi setaţi-i următoarele proprietăti:
 - **(Name): UpButton**
 - **Autosize: True**
 - **Location: 55, 70**
 - **Text: Close**
 7. Faceţi dublu click pe butonul **Next** din **PopForm** şi în handler-ul creat scrieţi următorul cod:

```

private void PopButton_Click(object sender, EventArgs e)
{
    UpForm newForm = new UpForm(); //se aloca spatiu in memorie pt forma 2
    newForm.Show();
    //se afiseaza forma2
}

```

8. Faceți click pe butonul **Close** din forma **UpForm** și scrieți următorul cod în handler-ul nou creat:

```

private void UpButton_Click(object sender, EventArgs e)
{
    this.Close();
    //this reprezintă obiectul curent la care aplicăm acțiunea
    //sau doar close();
}

```

9. Inserați în forma 1 PopForm un control de tip calendar
 10. Inserați în forma 1 un Label în care afișați textul: Care este data curentă?
 11. Inserați în forma 2, UpForm, un Label în care afișați textul: Care este ora curentă?
 12. Inserați în **UpForm**, un **Label** în care să se afișeze ora curentă. Setați-i următoarele proprietăți:

AutoSize=True

BorderStyle=Fixed3D (tipul chenarului aplicat textului)

Font=Arial,26,Bold (fontul textului atașat)

Text="" (textul atașat etichetei și afișat în fereastra)

TextAlign=MiddleCenter (modul în care este aliniat textul în eticheta)

Visible=True (stabilește dacă obiectul este sau nu vizibil în fereastră)

13. Trageți în UpForm, un Timer din Toolbox → Components

14. Setați pentru timer, un eveniment din fereastra Properties →  unde evenimentul Tick numiți-l **aplCeas_Tick**

```

private void aplCeas_Tick(object sender, EventArgs e){
    DateTime DataCurenta = DateTime.Now;
    //Data curentă va fi reținuta în variabila DataCurenta de tip DateTime și se va
    //afișa ca text în labelCeas
    labelCeas.Text = DataCurenta.ToString("yyyy-MM-dd HH:mm:ss");
}

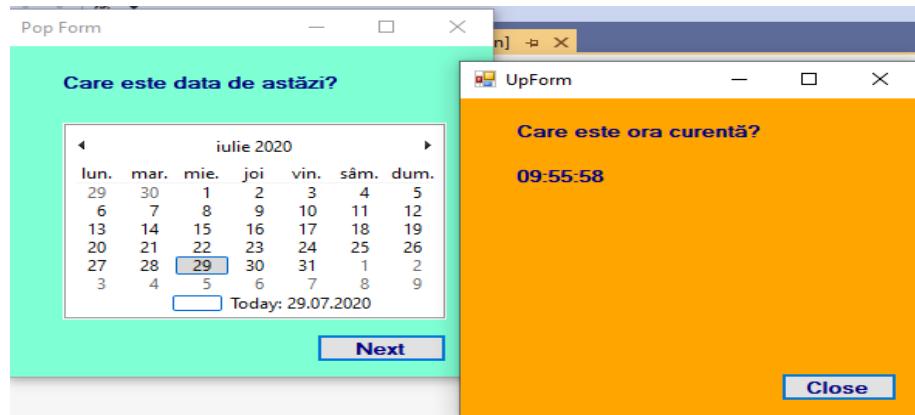
```

15. Accesați pentru timer, fereastra Properties  . Setați-i

Enable → True

Interval: 1000

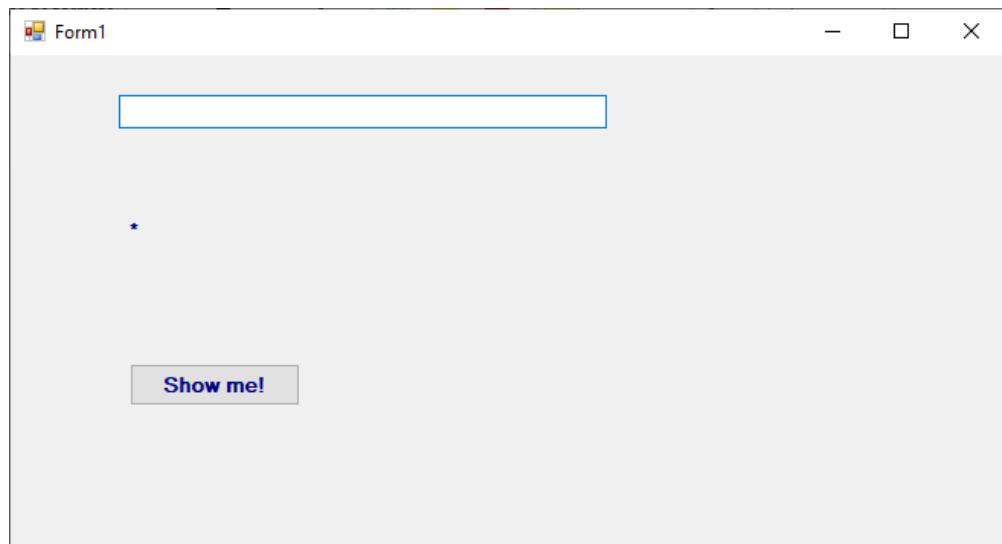
Pentru a vedea cum arată fereastra construită în realitate, apăsați tasta **F5** sau săgeata verde din bara de instrumente a mediului de programare.



Probleme propuse :

Aplicația 1- Password

Realizați aplicația pentru afișarea unei parole. Ea va conține trei controale: un label, un textBox și un button.



Proprietățile formei:

- **AutoScroll** – true
- **BackColor** – Gainsboro
- **Font** – Microsoft Sans Serif, Bold, 10
- **ForeColor** – Navy
- **Text** – Test

Inserați formaei: Label , TextBox, Button

Proprietăți textbox1:

- **Name** - **inputPasswordTextBox**
- **Anchor** – Top, Bottom, Left, Right
- **Font** – Microsoft Sans Serif, Bold, 10
- **ForeColor** – Navy

- **PasswordChar** – *
 - **Size** – 320;22
- Proprietăți label1:
- **Name** - **displayPasswordLabel**
 - **Text** – nimic
 - **Size** – 320;22
- Proprietăți button1:
- **Name** - **showPasswordButton**
 - **Size** – 180;37
 - **Text** – Show me!

În designer, faceți dublu click pe button ca să deschidă event handler-ul asociat și editați următorul cod:

```
private void showPasswordButton_Click(object sender, EventArgs e)
{
    displayPasswordLabel.Text = inputPasswordTextBox.Text;
}
```

Rulați aplicația și introduceți în căsuța de text un cuvânt și apoi apăsați butonul.

Aplicația 2- Cifre ordonate

Cerință: Se citește un număr natural n. Să se afișeze într-un nou form mesajul „da” în cazul în care cifrele numărului sunt ordonate descrescător respectiv mesajul „nu” într-un alt form, dacă cifrele numărului nu sunt ordonate.

Pentru aceasta vom deschide un proiect în WindowsFormApplication.

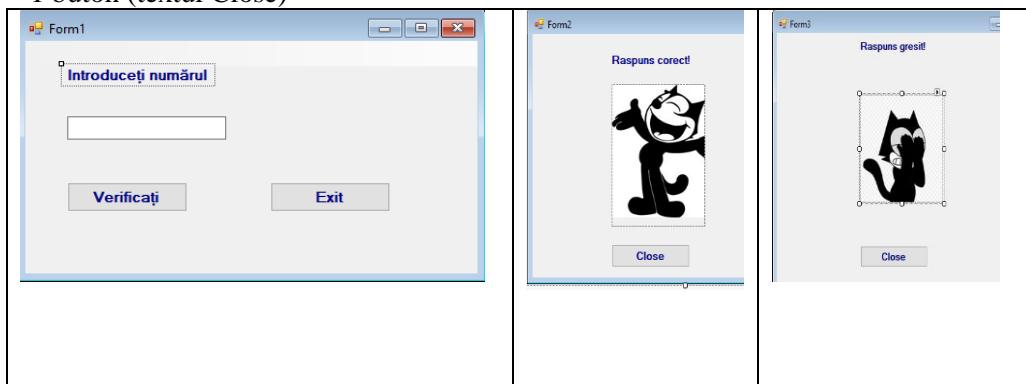
În fereastra curentă vom insera:

- 1 textBox – pentru n
- 1 Label cu textul – „Introduceți numărul”,
- 2 butoane cu textul – Verifica și Exit

Vom adăuga formei 1 încă 2 forme (Project→AddForm)

In fiecare formă se vor adăuga:

- 1 label. Pentru prima forma textul va fi „DA”, iar pentru a 2-a forma textul va fi „NU”
- 1 PictureBox – la care atașăm o imagine, în cazul nostru câte o pisică, pentru forma2 pisica câștigătoare, iar pentru forma3 pisica necâștigătoare (fereastra Properties→Image...→Loal source →Import→selectam imaginea corespunzătoare fiecărei forme)
- 1 buton (textul Close)



```

Declarăm cele 2 forme noi
private Form2 f1;
private Form3 f2;
Codul pentru butonul Verifica:
private void VerificaButton_Click(object sender, EventArgs e)
{
    int n,c;
    // ce valoare introducem in textBox1 convertim sла intreg si ova retine n
    n = Int32.Parse(textBox1.Text);
    //aflam prima cifra
    c = n % 10;
    n /= 10;
    while(n!=0)
    {
        if (n % 10 >= c)
            //verificam daca ultima cifra este > egala cu c
        {
            //daca da se deschide forma 3
            f2 = new Form3();
            f2.ShowDialog();
            break;
        }
        else
        {
            //daca cifra este mai mare se continua verificarea
            c = n % 10;
            n = n / 10;
        }
    }
    if(n==0)
        //daca am ajuns cu n =0 atunci cifrele sunt in ordine crescatoare
        //se deschide forma 2
    {
        f1 = new Form2();
        f1.ShowDialog();
    }
}

```

Pentru butonul **exit** codul

```

private void ExitButton_Click(object sender, EventArgs e)
{
    Application.Exit();
}

```

Pentru butoanele **close**

```

private void button1_Click(object sender, EventArgs e)
{
    //se inchide fereastra curenta
    Close();
}

```

Fişa 3 - Utilizarea controalelor

Butoanele radio (RadioButton) sunt asemănătoare cu căsuțele de validare deoarece au tot două stări: selectat și neselectat. Totuși, în mod normal, butoanele radio apar într-un grup în care doar un singur buton poate fi selectat la un moment dat. Selectând alt buton din grup, toate celelalte butoane vor fi în mod automat deselectate.

Observație

Utilizați RadioButtons când user-ul va selecta doar o opțiune din grup la un moment dat.

Utilizați CheckBoxes când user-ul poate selecta simultan mai multe opțiuni dintr-un grup.

Toate butoanele radio adăugate unei forme devin parte a aceluiași grup. Pentru a crea un nou grup de butoane trebuie să adăugați formei un obiect **GroupBox** din **ToolBox** și să inserați noile butoane radio în acest grup.

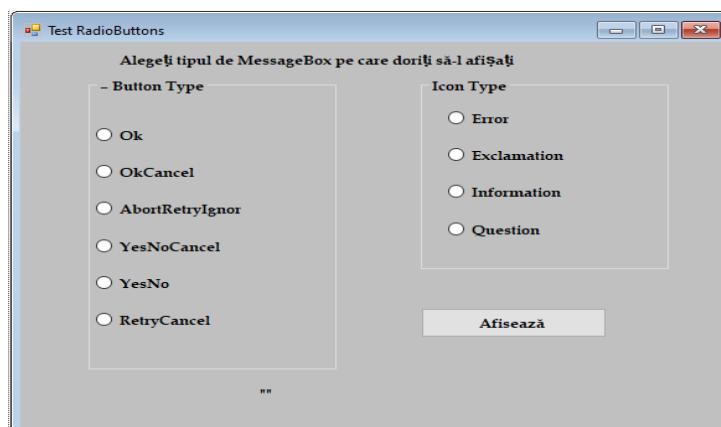
Proprietățile și evenimentele uzuale pentru RadioButton sunt:

- **Checked** – proprietate; indică dacă butonul este selectat implicit
- **Text** – proprietate; textul afișat în dreapta butonului radio (numit **eticheta** butonului)
- **Click** – eveniment survenit când user-ul face click pe buton. Event handler-ul este accesibil când faceți dublu click pe buton în modul design
- **CheckedChanged** – eveniment survenit ori de câte ori butonul își schimbă starea.

Aplicația 1- Utilizarea controalelor Button, Radio, MessageBox

Programul următor utilizează butoanele radio pentru a programa aspectul unui obiect MessageBox (fereastră/căsuță de mesaje). User-ul selectează cu ajutorul butoanelor radio butoanele și icon-ul care dorește să fie afișate în **MessageBox**, apoi acționează butonul pentru a afișa căsuța de mesaj. O etichetă plasată în colțul stânga jos afișează rezultatul din **MessageBox**, adică butonul apăsat de user pentru a închide căsuța.

Deschideți **C#** și creați un nou proiect **WindowsForm** cu numele **TestRadioButton**. În forma care se deschide adăugați din **ToolBox** următoarele obiecte:



Observație: obiectul **GroupBox** se găsește în **ToolBox** în tab-ul **Containers**. În fereastra **Properties** setați pentru obiectele proiectului următoarele proprietăți:

Fereastra Form1:	
Name - testRdBtnForm	
AutoScroll – true	
AutoSize – true	
AutoSizeMode - GrowAndShrink	
BackColor – Silver din tabul Web	
Font – Book Atiqua, Bold, 10	
Text – Test RadioButtons	
Grupul groupBox1:	
Name – typeGroup	Grupul groupBox2:
Text – Button Type	Name – iconGroup
Eticheta label1:	Eticheta label2:
Name – InfoLabel	Name – DisplayLabel
Text – Alegeți tipul de MessageBox pe care dorîți să-l afișați	Text – nimic("")
Butonul radioButton1:	
Name – OkRdBtn	Butonul radioButton7:
Text – OK	Name – ErrorRdBtn
Text – Error	Butonul radioButton8:
Butonul radioButton2:	
Name – OkCancelRdBtn	Name – ExclamRdBtn
Text – OKCancel	Text – Exclamation
Butonul radioButton3:	
Name – AbRetIgRdBtn	Butonul radioButton9:
Text – AbortRetryIgnore	Name – InfoRdBtn
Text – Information	Butonul radioButton10:
Butonul radioButton4:	
Name – YesNoCancelRdBtn	Name – QuestRdBtn
Text – YesNoCancel	Text – Question
Butonul radioButton5:	
Name – YesNoRdBtn	Buttonul button1
Text – YesNo	Name – DisplayBtn
Size – 150;30	
Butonul radioButton6:	
Name – RetCancelBtn	Text – Afiseaza
Text – RetryCancel	

Pentru a memora opțiunile user-ului, au fost create cele două grupuri de butoane: **typeGroup** și **iconGroup**.

Obiectul **iconGroup** este o enumerare a icon-urilor care vor fi afișate în **MessageBox**. Aceste icon-uri pot fi: **Asterisk**, **Error**, **Exclamation**, **Hand**, **Information**, **Question**, **Stop** și **Warning**. În acest proiect se folosesc numai iconurile **Error**, **Exclamation**, **Information** și **Question**.

Obiectul **typeGroup** este o enumerare a butoanelor care vor fi afișate în casuța **MessageBox** și acest proiect utilizează toate tipurile de combinații și anume: **OK**, **OKCancel**, **AbortRetryIgnore**, **RetryCancel**, **YesNo** și **YesNoCancel**.

Obiectul **MessageBox** este afișat prin apelul metodei **Show**:

MessageBox.Show(s,t,buttonType,iconType);

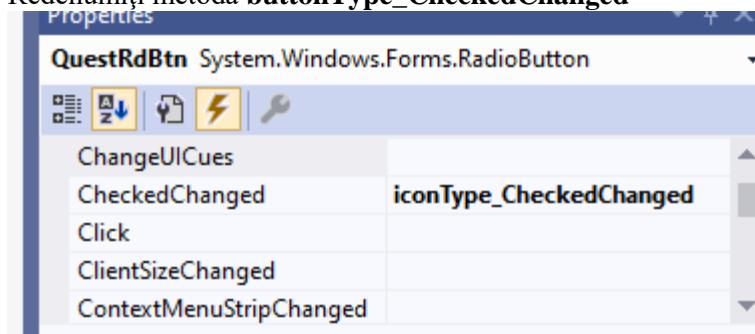
În această sintaxă, **s** este un sir de caractere care va fi afișat ca mesaj în interiorul căsuței, **t** este un sir de caractere care va fi afișat ca titlu în bara de titlu a căsuței, **buttonType** specifică ce butoane vor fi afișate în căsuți și **iconType** ce icon va fi afișat în căsuța de mesaje.

La apăsarea butonului **DisplayBtn**, căsuța de mesaje corespunzătoare opțiunilor alese va fi afișată și cu ajutorul etichetei **DisplayLabel** se va afișa ce buton din căsuță a fost apăsat de user.

Deoarece butoanele radio au fost grupate, există un singur administrator de evenimente pentru fiecare grup de butoane. Fiecare buton radio din grup generează un event **CheckedChanged** când este acționat.

Pentru a genera event handler-ul pentru un grup de butoane procedați astfel:

- Faceți dublu clic în designer pe primul buton din grupul **ButtonType** ca să se deschidă fereastra de cod
- Redenumiți metoda **buttonType_CheckedChanged**



- Setați event handler-ul **CheckedChanged** pentru fiecare buton radio din grupul **Button Type** la **buttonType_CheckedChanged** astfel: faceți un singur clic pe fiecare buton pe rand; în fereastra **Properties** faceți clic pe fulger ca să deschideți lista de event handler-e asociate și din lista derulantă selectați event handler-ul dorit ca în imaginea următoare
 - Procedați la fel pentru fiecare buton din grupul **Button Type**
- Creați un event handler pentru oricare buton din grupul Icon și redenumiți-l **iconType_CheckedChanged**.

Setați din fereastra **Properties** event handler-ul **CheckedChanged** al fiecărui buton la această denumire.

În acest moment acționarea unui buton din fiecare grup este rezolvată cu un singur event handler asociat. Va trebui să facem distincția între butoanele din același grup.

Fiecare event handler va compara argumentul **sender** (care este un obiect – cel care trimite) cu fiecare buton pentru a determina ce buton a fost selectat de user.

Pentru a putea utiliza aceste handleuri de grup, va trebui să creăm două obiecte în care să memorăm opțiunile utilizatorului. Pentru aceasta, faceți click în **Solution Explorer** pe codul ferestrei **Form1**.

Înaintea primului handler creat anterior, introduceți declarațiile:

```
private MessageBoxButtons buttonType = MessageBoxButtons.OK;
//ptr. primul grup de butoane//ptr. primul grup de butoane
private MessageBoxIcon iconType = MessageBoxIcon.Error;
//ptr.al doilea grup de butoane//ptr.al doilea grup de butoane
```

In fereastra de cod a event handler-ului **buttonType_CheckedChanged** introduceți următorul cod:

```
private void buttonType_CheckedChanged(object sender, EventArgs e)
{
    if (sender == OkRdBtn)
        buttonType = MessageBoxButtons.OK;
    else if (sender == OkCancelRdBtn)
        buttonType = MessageBoxButtons.OKCancel;
    else if (sender == AbRetIgRdBtn)
        buttonType = MessageBoxButtons.AbortRetryIgnore;
    else if (sender == YesNoCancelRdBtn)
        buttonType = MessageBoxButtons.YesNoCancel;
    else if (sender == YesNoRdBtn)
        buttonType = MessageBoxButtons.YesNo;
    else buttonType = MessageBoxButtons.RetryCancel;
}
```

Procedați analog și scrieți în fereastra de cod a event handler-ului **iconType_CheckedChanged** codul:

```
private void iconType_CheckedChanged(object sender, EventArgs e)
{
    if (sender == ErrorRdBtn)
        iconType = MessageBoxIcon.Error;
    else if (sender == ExclamRdBtn)
        iconType = MessageBoxIcon.Exclamation;
    else if (sender == InfoRdBtn)
        iconType = MessageBoxIcon.Information;
    else iconType = MessageBoxIcon.Question;
}
```

Faceți dublu click pe **DisplayBtn** și introduceți codul:

```
private void DisplayBtn_Click(object sender, EventArgs e)
{
    DialogResult resut = MessageBox.Show("Acesta este obiceiul  
tau MessageBox.", "CustomMessageBox", buttonType, iconType, 0, 0);
    switch(resut)
    {
        case DialogResult.OK: DisplayLabel.Text = " Ok a fost  
apăsat. "; break;
        case DialogResult.Cancel: DisplayLabel.Text = " Cancel  
a fost apăsat. "; break;
        case DialogResult.Abort: DisplayLabel.Text = " Abort a  
fost apăsat. "; break;
        case DialogResult.Retry: DisplayLabel.Text = " Retry a  
fost apăsat. "; break;
        case DialogResult.Ignore: DisplayLabel.Text = " Ignore  
a fost apăsat. "; break;
        case DialogResult.Yes: DisplayLabel.Text = " Yes a fost  
apăsat. "; break;
        case DialogResult.No: DisplayLabel.Text = " NO a fost  
apăsat. "; break;
    }
}
```

Event handler-ul **Click** al butonului **DisplayBtn** creează o căsuță de mesaje cu numele **Custom MessageBox** (afișat în bara de titlu) care afișează mesajul "**This is your custom MessageBox.**". Butoanele afișate în căsuță sunt stabilite prin obiectul **buttonType**, iar icon-ul afișat prin obiectul **iconType** programate anterior.

Rezultatul produs de un obiect **MessageBox** este o enumerare cu numele **DialogResult** (produsa de limbaj). Aceasta enumerare poate avea valorile: **Abort**, **Ignore**, **Yes**, **No**, **Cancel**, **Retry**, **Ok** sau **None**, adică corespunde tuturor butoanelor care ar putea fi afișate în **MessageBox**. Valoarea obiectului **DialogResult** poate fi testată (instrucțiunea **switch**) pentru a afla ce buton a acționat user-ul. În **DisplayLabel** se afișează mesajul care indică butonul apăsat.

Aplicația 2- Afișarea imaginilor dintr-o listă de imagini

Se va utiliza controlului **ImageList**. Acesta este un control care conține o listă de imagini, care poate fi setată la design (proprietatea **Images**).

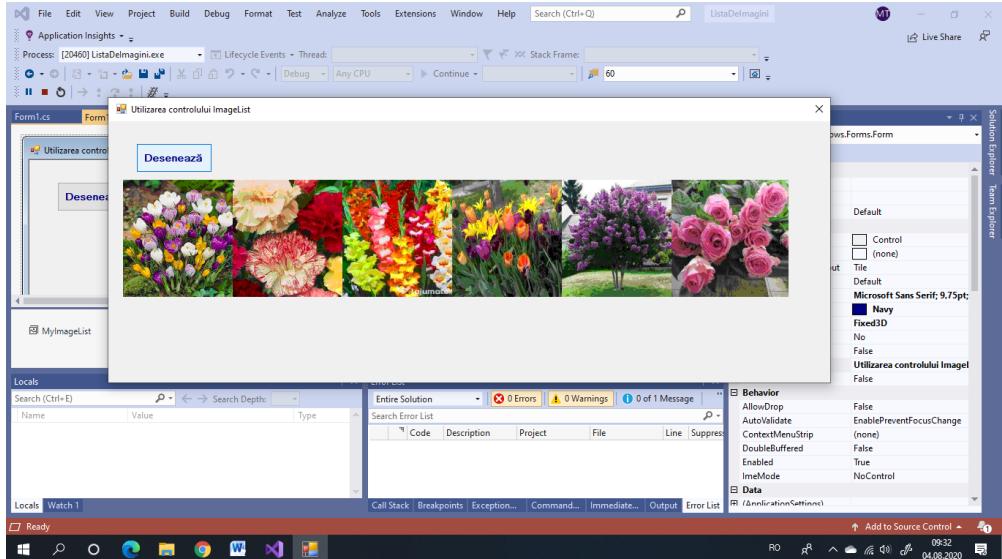
Controlul **ImageList** dispune de o metodă **Draw()** care permite desenarea imaginilor pe care le conține.

- Creați un proiect **Windows Forms** cu numele **UtilizareImageList**
- Pentru forma care se deschide setați proprietatile:
 - **(Name): mainForm**
 - **Font: Bold**
 - **FormBorderStyle: Fixed3D**
 - **MaximizeBox: False**
 - **MinimizeBox: False**
 - **Size: 800, 300**
 - **Text: Utilizarea controlului ImageList**
- Trageți pe forma un control **Button** și setați urmatoarele proprietăți:
 - **(Name): drawButton**
 - **AutoSize: True**
 - **Location: 38, 30**
 - **Text: Deseneaza**
- Trageți pe forma un control **ImageList** (din secțiunea **Components** din **Toolbox**) și setați urmatoarele proprietăți:
 - **(Name): myImageList**
 - **ImageSize: 150, 150** (dimensiunea de afisare a imaginilor din lista)
- Faceți clic pe proprietatea **Images** și apoi pe caseta cu trei puncte din dreapta pentru a deschide selectorul de imagini. În fereastra care se deschide faceți click pe butonul **Add** și încărcați cele cinci imagini pregătite în directorul proiectului. Când ati terminat faceți click pe butonul **OK**.

Pentru butonul **Desenează** aveți de introdus codul următor:

```
private void drawButton_Click(object sender, EventArgs e)
{
    //creeaza pentru forma un obiect Graphics (desen)
    Graphics grafic = this.CreateGraphics();
    // fiecare imagine din ImageList este desenata pe ecran cu metoda
    // Draw()
    for (int i = 0; i < MyImageList.Images.Count; i++)
        MyImageList.Draw(grafic, i * 150 + 20, 80, i);
    grafic.Dispose();
}
```

Imaginiile încărcate în **ImageList** sunt identificate prin numărul de ordine **i** (începând de la 0) și sunt desenate pe ecran începând din punctul de coordonate (**i*150+20, 80**) (orizontal, vertical). Reamintim că pentru proprietatea **ImageSize** a controlului **ImageList** au fost setate valorile **150, 150** (dimensiunea unei imagini).



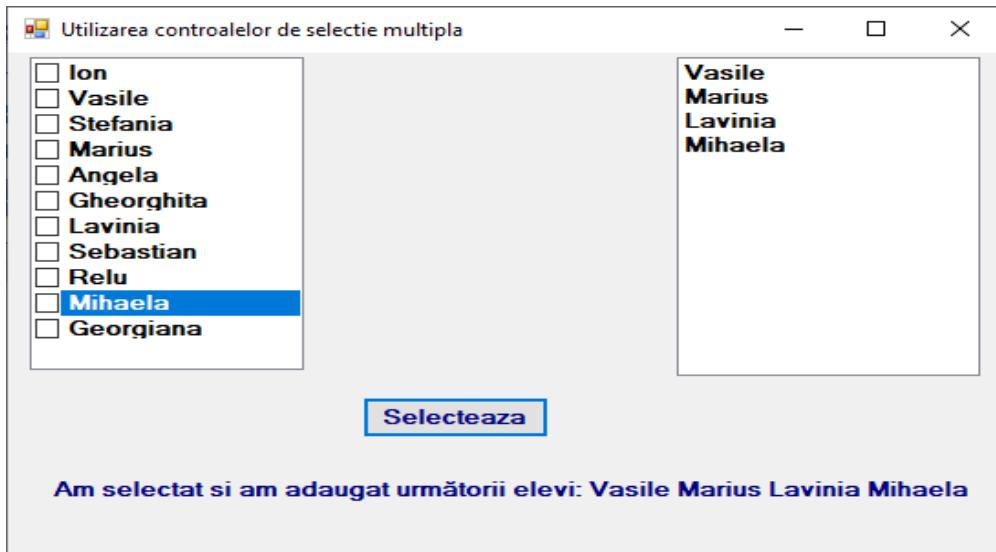
Aplicația 3- Utilizarea controlului **CheckedListBox**

Deschideți un nou proiect WindowsApplication cu numele **ControaleSelectieMultipla**. Pentru a utiliza controale pentru selecție și selecție multiplă vom adăuga formei un **CheckedListBox** care conține mai multe casete de validare și un **ListBox** care afișează mai mulți itemi de exemplu numele elevilor dintr-o clasă.

Vom adăuga formei un control **CheckedListBox** care va conține lista elevilor cu casete de validare pentru elev. La apăsarea unui buton, itemii bifati vor fi afișați într-un control **ListBox** și într-o etichetă **Label**.

- Pentru forma care se deschide setați următoarele proprietatea:
 - **(Name): mainForm**
 - **BackColor: Gainsboro** (tabul Web)
 - **Font: Microsoft Sans Serif 8 Bold**
 - **ForeColor: Navy** (tabul Web)
 - **FormBorderStyle: FixedDialog**
 - **MaximizeBox: False**
 - **MinimizeBox: False**
 - **Size: 560; 374**
 - **Text: Utilizarea controalelor de selecție multiplă**
- Adăugați formei un control **CheckedListBox** și setați următoarele proprietăți:
 - **(Name): inputCheckedListBox**
 - **CheckOnClick: True** (caseta este bifată cand facem clic pe ea)

- **ForeColor: Indigo**
- **Location: 30, 35**
- **Size: 140 110**
- În fereastra **Properties** faceți clic pe proprietatea **Items** și apoi pe caseta cu trei puncte din dreapta pentru a edita itemii care vor fi afișați în **CheckedListBox**. În fereastra de editare care se deschide, selectați itemii doriti ca în figura următoare:



Aplicația 4- Utilizarea controlului ListView.

ListView este folosit pentru a afișa o colecție de elemente în unul din cele 4 moduri (**Text**, **Text+Imagini mici**, **Imagini mari**, **Detalii**). Acesta este similar grafic cu ferestrele în care se afișează fișierele dintr-un anumit director din **Windows Explorer**. Fiind un control complex, conține foarte multe proprietăți, printre care:

- **View** (selectează modul de afișare (**LargeIcon**, **SmallIcon**, **Details**, **List**)),
- **LargeImageList**, **SmallImageList** (icon-urile de afișat în modurile **LargeIcon**, **SmallIcon**),
- **Columns** (utilizat doar în modul **Details**, pentru a defini coloanele de afișat), **Items** (elementele de afișat).

Aplicația de față are ca scop să se fișezeze într-un **ListView** o listă de elevi cu notele obținute la un test.. Proiectul conține clasa **Elev** pentru care se aplică proprietățile **Nume**, **Prenume** și **Nota** și cu metoda statică **CitesteElevi()** care returnează o listă de elevi (ne putem imagina că lista respectivă este citită dintr-o baza de date).

Metoda **SeteazaLista** pregătește lista pentru ca datele să fie afișate pe coloane și sortate alfabetic.

- Creați un proiect **Windows Forms** cu numele **UtilizareListView**
- Pentru forma care se deschide setați urmatoarele proprietăți:
 - **(Name): mainForm**
 - **AutoScroll: True**

- **Font: Bold**
 - **FormBorderStyle: FixedDialog**
 - **MaximizeBox: False**
 - **MinimizeBox: False**
 - **Size: 550, 340**
 - **Text: Utilizarea controlului ListView**
- Trageți pe formă un control **ListView** și setați următoarele proprietăți:
 - **(Name): eleviListView**
 - **BackColor: OldLace (tabul Web)**
 - **Dock : Fill (controlul va umple tot spațiul formei)**
- Alegeti din meniu **View→Code** sau apăsați tasta **F7** și adăugați codul metodei **SeteazaLista()** care stabilește opțiunile de afișare pentru controlul **ListView**

```
private void SeteazaLista()
{
    //adauga cele trei coloane cu titlu, latime si aliniere orizontala
    eleviListView.Columns.Add("Nume", 200, HorizontalAlignment.Left);
    eleviListView.Columns.Add("Prenume", 200, HorizontalAlignment.Left);
    eleviListView.Columns.Add("Nota", 200, HorizontalAlignment.Left);
        //modul de vizualizare este Detalii
    eleviListView.View = View.Details;
        //informatiile din lista sunt sortate alfabetic
    eleviListView.Sorting = SortOrder.Ascending;
    eleviListView.AllowColumnReorder = true;
}
```

- Alegeti din meniu **Project→Add Class** pentru a aduga proiectului o clasa si in fereastra de dialog care se deschide numiti clasa **Elev.cs**.
- In clasa creata scrieți următorul cod:

```
public static List<Elev> CitesteElevi()
{
    //se creeaza o lista cu elemente de tip Elev
    List<Elev> elevi = new List<Elev>();
    //lista este populata cu patru obiecte de tip Elev
//Informatiile despre elevi se seteaza cu ajutorul proprietatilor
    //Nume, Prenume si Nota declarate anterior
    elevi.Add(new Elev()
    {Nume = "Popescu", Prenume = "Daniel", Nota = 9});
    elevi.Add(new Elev()
    { Nume = "Danila", Prenume = "Elena", Nota = 10 });
    elevi.Add(new Elev()
    { Nume = "Pavel", Prenume = "Dan", Nota = 8 });
    elevi.Add(new Elev()
    { Nume = "Ciornei", Prenume = "Andrei", Nota = 7});
    //lista poate continua
    //dupa creare si populare, lista este returnata de catre metoda
    return elevi;
}
```

- Faceți dublu-clic pe forma **mainForm** pentru a deschide administratorul **Load** (implicit pentru forme). În administrator scrieți codul:

```

private void Form1_Load(object sender, EventArgs e)
{
    //incepe actualizarea informatiilor din ListView

    this.eleviListView.BeginUpdate();
    ListViewItem a;
    ListViewItem.ListViewSubItem b;

    //c este un obiect din multimea de elevi returnata de metoda
    //CitesteElevi() din clasa Elev
    foreach (Elev c in Elev.CitesteElevi())
    {
        //se creeaza un nou item pentru controlul ListView
        a = new ListViewItem();
        a.Text = c.Nume;
        //numele elevului din obiectul c este memorat in proprietatea Text
        b = new ListViewItem.ListViewSubItem();
        //se creeaza un subitem pentru controlul ListView
        b.Text = c.Prenume;
        //prenumele elev. este memorat in obiectul a proprietatea Text
        a.SubItems.Add(b);
        //se adauga prenumele la itemul a
        b = new ListViewItem.ListViewSubItem();
        //se creeaza un nou subitem b in care se memoreaza nota elevului
        b.Text = c.Nota.ToString();
        //nota se adauga in itemul a
        a.SubItems.Add(b);
        //itemul a este complet si se adauga in controlul ListView
        eleviListView.Items.Add(a);
    }
    this.eleviListView.EndUpdate();
    //s-a terminat actualizarea informatiilor din ListView
    SeteazaLista();
    //apeleaza metoda care stabileste optiunile de afisare pentru
    //ListView
}

```

Nume	Prenume	Nota
Ciornel	Andrei	7
Danila	Elena	10
Pavel	Dan	8
Popescu	Daniel	9

Aplicația 5 – Controlul MonthCalendar

Controlul **MonthCalendar** afișează un calendar prin care se poate selecta o dată (zi, luna, an) în mod grafic. Proprietățile importante sunt: **MinDate**, **MaxDate**, **TodayDate** ce reprezintă data minimă/maximă selectabilă și data curentă (care apare afișată diferențiat sau nu, în funcție de valorile proprietăților **ShowToday**, **ShowTodayCircle**).

Există două evenimente pe care controlul le expune: **DateSelected** și **DateChanged**. În rutinele de tratare a acestor evenimente, programatorul are acces la un obiect de tipul **DateRangeEventArgs** care conține proprietățile **Start** și **End** (reprezentând intervalul de timp selectat).

Formularul din aplicație conține un calendar pentru care putem selecta un interval de maximum 30 de zile, sunt afișate săptămânile și ziua curentă. Intervalul selectat se afișează prin intermediul unei etichete. Dacă se selectează o singura dată atunci aceasta va fi adăugată ca item într-un **ComboBox** (orice dată poate apărea cel mult o dată în listă).

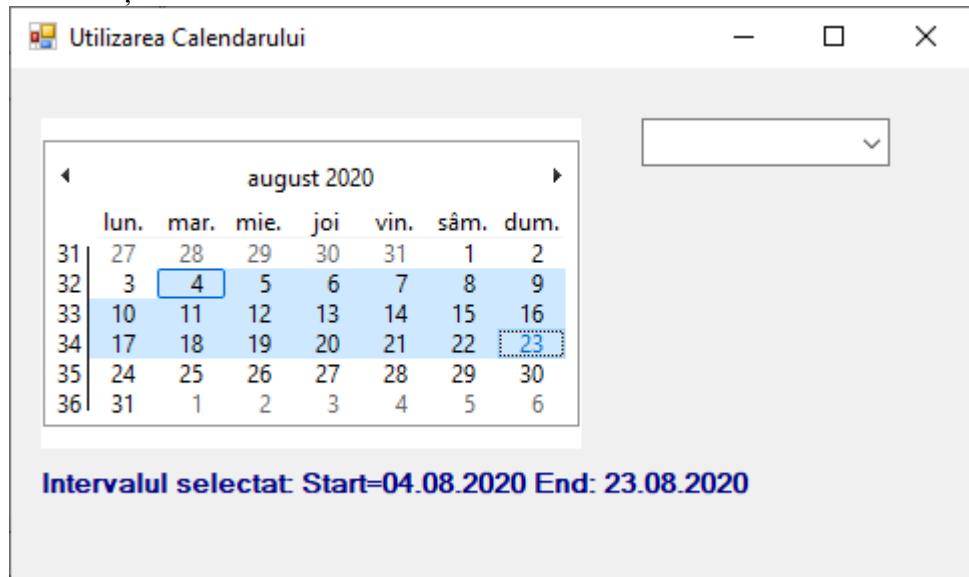
- Creati un proiect **Windows Forms** cu numele **UtilizareCalendar**
- Pentru forma care se deschide setati urmatoarele proprietati:
 - **(Name): mainForm**
 - **BackColor: Silver (tabul Web)**
 - **Font: Bold**
 - **ForeColor: Navy (Web)**
 - **FormBorderStyle: FixedSingle**
 - **Size: 445, 265**
 - **StartPosition: CenterScreen**
 - **Text: Utilizare Calendar**
- Trageți pe forma un control **MonthCalendar** și setați următoarele proprietăți:
 - **(Name): myMonthCalendar**
 - **BackColor: BlanchedAlmond (Web)**
 - **ForeColor: Navy**
 - **Location: 15, 25**
 - **MaxSelectionCount: 30 (numarul maxim de zile dintr-un interval selectat)**
 - **ShowTodayCircle: False**
 - **ShowWeekNumber: True**
 - **TitleBackColor: DarkMagenta (Web)**
- Trageți pe forma un control **ComboBox** și setați următoarele proprietăți:
 - **(Name): myComboBox**
 - **Locaton: 275, 25**
- Trageți pe forma un control **Label** și setați următoarele proprietăți:
 - **(Name): myLabel**
 - **Location: 15, 200**
- Selectați calendarul și în fereastra **Properties** selectați tabul **Events** (fulgerul galben). Din lista evenimentelor selectați **DateSelected** și faceți dublu-clic pentru a deschide administratorul asociat. Scrieți codul următor:

```

private void MyMonthCalendar_DateSelected(object sender,
DateRangeEventArgs e)
{
    //in eticheta se afiseaza intervalul selectat
    this.myLabel.Text = "Intervalul selectat: Start=" +
        e.Start.ToShortDateString() + " End: " +
        e.End.ToShortDateString();
    //daca este selectata o singura zi
    if (e.Start.ToShortDateString() ==
        e.End.ToShortDateString())
    {
        string x = e.Start.ToShortDateString();
        //respectiva data este adaugata in ComboBox, daca nu
        exists deja
        if (!myComboBox.Items.Contains(x))
            myComboBox.Items.Add(x);
    }
}

```

Se va afișa:



Fișă 4 - Utilizarea meniurilor

Aplicația 1- Formatarea textului din TextBox sau dintr-un label

Meniurile sunt utilizate pentru a oferi grupuri de comenzi înrudite pentru aplicațiile Windows. Deși aceste comenzi depind de aplicație, unele - cum ar fi **Open** și **Save** – sunt comune multor aplicații.

Fiecare opțiune de meniu poate avea o scurtătură – **ALT+litera subliniată** – spre exemplu **ALT+F** selectează opțiunea **File**. Pentru a crea o scurtătură trebuie să tastezi caracterul **&** în fața denumirii opțiunii (de exemplu **&File**). Opțiunile de meniu (**menu items**) generează evenimentul **Click** când sunt selectate.

-Deschideți C# și creați un proiect WindowsForm cu numele **test_meniuri**.

În fereastra, trageți din **ToolBox**, din secțiunea **Menus&Toolbars**, un obiect **menuStrip** și poziționați-l în colțul stânga sus al ferestrei. Adăugați un obiect **Label** în partea de jos a ferestrei și un **TextBox** care va trebui să-l setați **Multiline** și dimensionați-l astfel încât să umple fereastra.

Pentru **Form1**, din fereastra **Properties Window**, setați proprietățile:

Name – MenuForm

Backcolor – Control

Font – Microsoft Sans Serif, Bold, 10

Text – Utilizare meniuri

Pentru **Label1**:

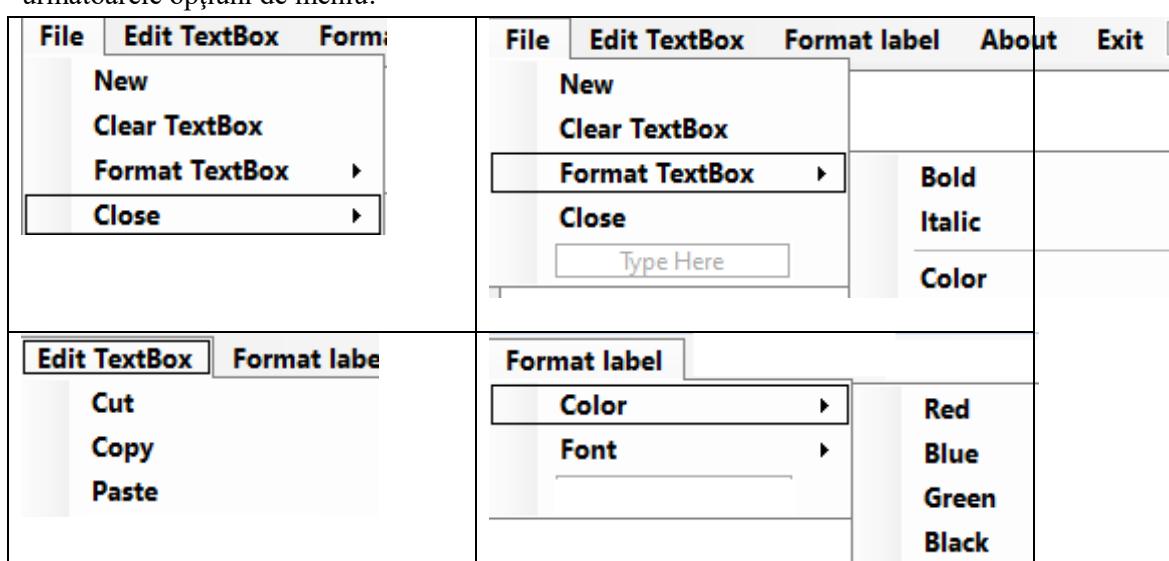
Name – displayLabel

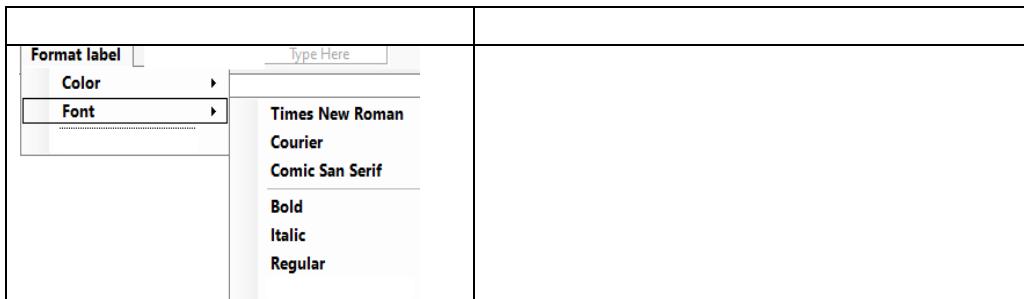
Font – Microsoft Sans Serif, Regular, 10

ForeColor – Control text (din System)

Text – Utilizați meniul Format ca să schimbiți aspectul acestui text

Trageți din fereastra **Toolbox**, din **Meniu&Toolbars**, un **MenuStrip** și introduceți următoarele opțiuni de meniu:





Faceți dublu click pe opțiunea **New** și introduceți codul:

```
private void newToolStripMenuItem_Click(object sender, EventArgs e)
{
    System.Diagnostics.Process.Start("notepad");
    //se va deschide aplicația NotePad
}
```

Faceți dublu-clic pe opțiunea **Close** și scrieți codul:

```
TextBox1.Clear(); //sterge textul afisat in caseta
TextBox1.Visible = false; //caseta redevine invizibila
```

Faceți dublu click pe opțiunea **Clear TextBox** și introduceți codul:

```
TextBox1.Clear()//se va sterge textul introdus în TextBox
```

În meniu **Format TextBox** conține opțiunile **Bold**, **Italic**, **Color** cu care vom forma textul selectat dinn **TextBox1**.

Faceți dublu click pe opțiunea **Bold** introduceți codul:

```
private void boldToolStripMenuItem_Click(object sender, EventArgs e)
{
boldToolStripMenuItem.Checked = boldToolStripMenuItem.Checked;
    TextBox1.Font = new Font(TextBox1.Font.FontFamily, 14,
    TextBox1.Font.Style ^ FontStyle.Bold);
}
```

De asemenea faceți dublu click pe opțiunea **Italic** introduceți codul:

```
private void italicToolStripMenuItem_Click(object sender, EventArgs e)
{
italicToolStripMenuItem.Checked = italicToolStripMenuItem.Checked;
    TextBox1.Font = new Font(TextBox1.Font.FontFamily, 14,
    TextBox1.Font.Style ^ FontStyle.Italic);}
```

Pentru colorarea textului selectat din **TextBox1** va trebui să faceți dublu click pe rând pe opțiunile **Red**, **Blue**, **Green** și **Black** și să introduceți la fiecare următoarele coduri:

```
private void redToolStripMenuItem1_Click(object sender, EventArgs e)
{
    ClearColor();
    TextBox1.ForeColor = Color.Red;
    blackItem.Checked = true;
}
```

```

private void blueToolStripMenuItem1_Click(object sender, EventArgs e)
{
    ClearColor();
    TextBox1.ForeColor = Color.Blue;
    blackItem.Checked = true;
}

private void greenToolStripMenuItem1_Click(object sender, EventArgs e)
{
    ClearColor();
    TextBox1.ForeColor = Color.Green;
    blackItem.Checked = true;
}

private void blackToolStripMenuItem1_Click(object sender, EventArgs e)
{
    ClearColor();
    TextBox1.ForeColor = Color.Black;
    blackItem.Checked = true;
}

```

Meniul **Edit** conține opțiunile **Cut**, **Copy**, **Paste**. Pentru manipularea textului selectat din **TextBox1** va trebui să faceți dublu click pe rând pe opțiunile **Cut**, **Copy**, **Paste** și să introduceți la fiecare următoarele coduri:

```

private void copyToolStripMenuItem_Click(object sender, EventArgs e)
{
    TextBox1.Copy();
}

private void cutToolStripMenuItem_Click(object sender, EventArgs e)
{
    TextBox1.Cut();
}

private void pasteToolStripMenuItem_Click(object sender, EventArgs e)
{
    TextBox1.Paste();
}

```

Meniul **Format Label este** compact și conține opțiunile de formatare a textului din **displayLabel**. Astfel în opțiunea color colorăm textul.

Pentru a putea colora textul din **displayLabel** va trebui să faceți dublu click pe rând pe opțiunile **Red**, **Blue**, **Green** și **Black** și să introduceți la fiecare următoarele coduri:

```

private void redToolStripMenuItem_Click(object sender, EventArgs e)
{
    ClearColor();
    displayLabel.ForeColor = Color.Red;
    redItem.Checked = true;
}

private void blueToolStripMenuItem_Click(object sender, EventArgs e)
{
    ClearColor();
    displayLabel.ForeColor = Color.Blue;
    redItem.Checked = true;
}

```

```

private void redToolStripMenuItem_Click(object sender, EventArgs e)
{
    ClearColor();
    displayLabel.ForeColor = Color.Green;
    redItem.Checked = true;
}

private void redToolStripMenuItem_Click(object sender, EventArgs e)
{
    ClearColor();
    displayLabel.ForeColor = Color.Black;
    redItem.Checked = true;
}

```

- Observați structura codului pentru obiectul **displayLabel** este aceeași ca și la obiectul **TextBox1**. Diferă doar locul de formatarea textului.

displayLabel.ForeColor = Color.Black;

sau

textBox1.ForeColor = Color.Black;

- Pentru a introduce linia separatoare între culori și fonturi, faceți click dreapta pe opțiunea **Bold - Insert – Separator**.
- Faceți click pe opțiunea **File** și setați proprietățile:

Name – fileItem

Font – Tahoma Bold 10 (automat, toate opțiunile din meniu **File** vor fi bold)

- Faceți click pe opțiunea **Format Label** și setați proprietățile:

Name – formatItem

Font – Tahoma Bold 10 (automat toate opțiunile din meniu **Format** vor fi boldite)

- Faceți click pe rând pe fiecare opțiune de meniu și redenumiți-le: **aboutItem, exitItem, colorItem, fontItem, blackItem, blueItem, redItem, greenItem, timesItem, courierItem , comicItem, boldItem și italicItem**.
- Vom adăuga administratori de evenimente pentru fiecare opțiune de meniu.

```

private void timestem_Click(object sender, EventArgs e)
{
displayLabel.Font = new Font("Times New Roman", 14,
displayLabel.Font.Style);
timesItem.Checked = true;
}

private void courierItem_Click(object sender, EventArgs e)
{
displayLabel.Font = new Font("Courier", 14, displayLabel.Font.Style);
timesItem.Checked = true;
}

private void comicItem_Click(object sender, EventArgs e)
{
displayLabel.Font = new Font("Comic San Serif", 14,
displayLabel.Font.Style);
timesItem.Checked = true;
}

```

```

private void boldItem_Click(object sender, EventArgs e)
{
    boldItem.Checked = !boldItem.Checked;
    displayLabel.Font = new Font(displayLabel.Font.FontFamily, 14,
    displayLabel.Font.Style ^ FontStyle.Bold);
}
private void italicItem_Click(object sender, EventArgs e)
{
    italicItem.Checked = !italicItem.Checked;
    displayLabel.Font = new Font(displayLabel.Font.FontFamily, 14,
    displayLabel.Font.Style ^ FontStyle.Italic);
}
private void regularItem_Click(object sender, EventArgs e)
{
    regularItem.Checked = regularItem.Checked;
    displayLabel.Font = new Font(displayLabel.Font.FontFamily, 14,
    displayLabel.Font.Style ^ FontStyle.Regular);
}

```

Faceți dublu click pe **About** și introduceți codul:

```

private void aboutItem_Click(object sender, EventArgs e)
{
    MessageBox.Show("Exemplu de utilizare a meniurilor",
    "Informatii", MessageBoxButtons.OK, MessageBoxIcon.Information);
}

```

Se va afișa o căsuță de mesaje cu informațiile de mai sus.

Faceți dublu click pe **Exit** și adăugați codul:

```

private void exitItem_Click(object sender, EventArgs e)
{
    Application.Exit();
}

```

Va avea ca efect închiderea aplicației. Se folosește exact la fel pentru a închide orice aplicație C#.

În fereastra de cod scrieți funcția care are rolul de a reseta toate culorile selectate anterior.

```

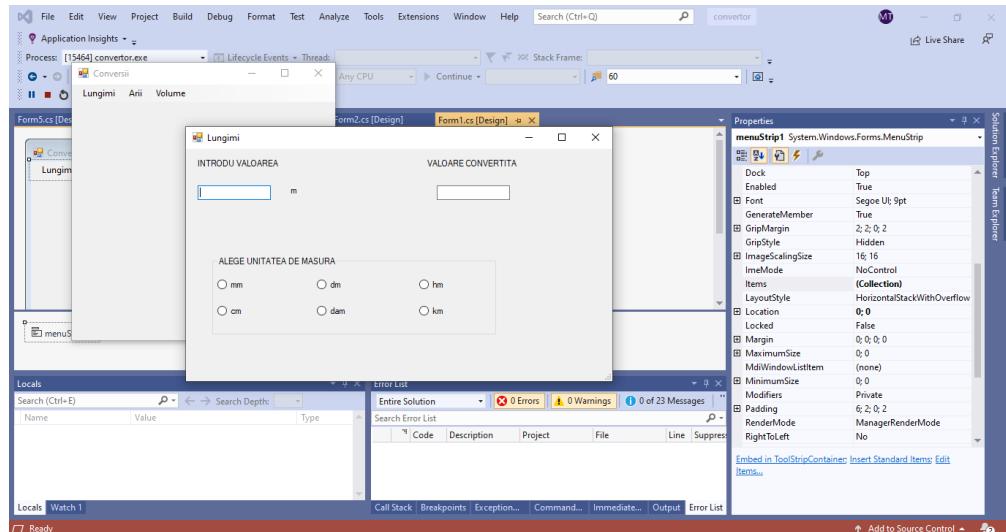
private void ClearColor()
{
    blackItem.Checked = false;
    blueItem.Checked = false;
    redItem.Checked = false;
    greenItem.Checked = false;
}

```

Fișă 5 – Proiecte interdisciplinare

Aplicația 1- Convertor de lungimi (informatică-matematică-fizică)

Folosind MeniuStrep și adăugând forme pentru fiecare conversie realizați aplicația din imaginea de mai jos.



Codul sursă pentru forma1

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace convertor
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        private Form2 f1;
        private Form3 f2;
        private Form4 f3;

        private void lungimiToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            f1 = new Form2();
            f1.ShowDialog();
        }
    }
}
```

```

        private void ariiToolStripMenuItem_Click(object sender,
EventArgs e)
{
    f2 = new Form3();
    f2.ShowDialog();
}

private void volumeToolStripMenuItem_Click(object sender,
EventArgs e)
{
    f3 = new Form4();
    f3.ShowDialog();
}
}
}

```

Codul sursa pentru forma 2:

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace convertor
{
    public partial class Form2 : Form
    {
        private double R;
        private double V;
        public Form2()
        {
            InitializeComponent();
        }

        private void radioButton2_CheckedChanged(object sender,
EventArgs e)
{
    V = double.Parse(textBox1.Text);
    R = V * 100;
    textBox2.Text = R.ToString();
}

        private void radioButton1_CheckedChanged(object sender,
EventArgs e)
{
    V = double.Parse(textBox1.Text);
    R = V * 1000;
    textBox2.Text = R.ToString();
}
}

```

```

private void radioButton3_CheckedChanged(object sender,
EventArgs e)
{
    V = double.Parse(textBox1.Text);
    R = V *10;
    textBox2.Text = R.ToString();
}

private void radioButton4_CheckedChanged(object sender,
EventArgs e)
{
    V = double.Parse(textBox1.Text);
    R = V / 10;
    textBox2.Text = R.ToString();
}

private void radioButton5_CheckedChanged(object sender,
EventArgs e)
{
    V = double.Parse(textBox1.Text);
    R = V / 100;
    textBox2.Text = R.ToString();
}

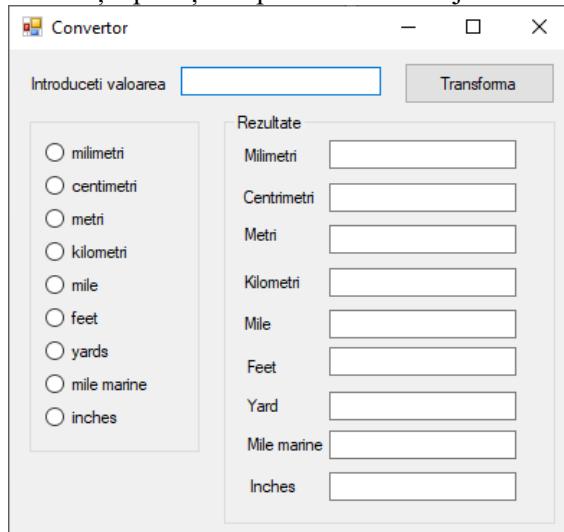
private void radioButton6_CheckedChanged(object sender,
EventArgs e)
{
    V = double.Parse(textBox1.Text);
    R = V/1000;
    textBox2.Text = R.ToString();
}
}

```

Completați proiectul.

Propuneri:

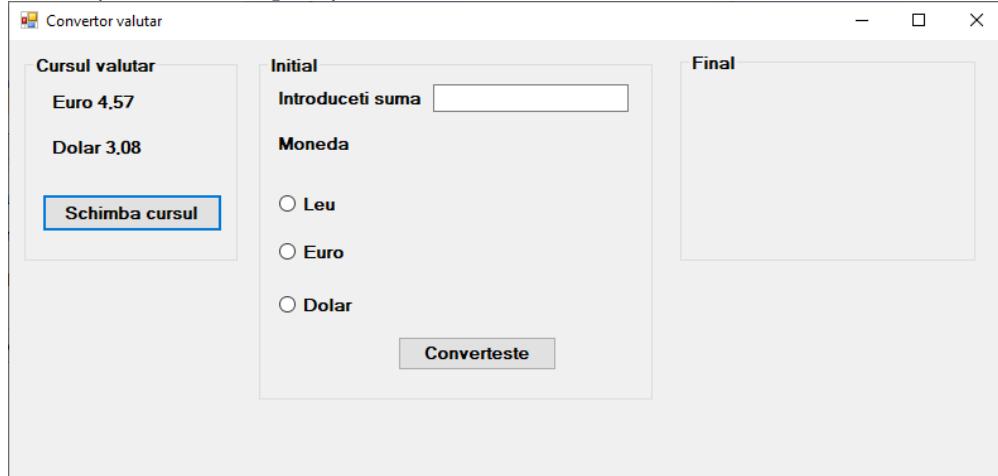
Realizați aplicația după forma de mai jos:



Aplicația 2- Convertor de lungimi (informatică-matematică)

Aplicația își propune conversia valutară lei-euro-dolar și invers.

Realizați următoarea aplicație:



Codul sursă este:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.IO;

namespace convertorvalutar
{
    public partial class Form1 : Form
    {
        public double ceuro, cdolar;
        public double suma;
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            StreamReader sr = new StreamReader("curs.txt");
            ceuro = double.Parse(sr.ReadLine());
            cdolar = double.Parse(sr.ReadLine());
            le.Text = "Euro " + ceuro;
            ld.Text = "Dolar " + cdolar;
            sr.Close();
        }
    }
}
```

```

private void b1_Click(object sender, EventArgs e)
{
    Form2 f2 = new Form2();
    f2.ShowDialog();
}

private void b2_Click(object sender, EventArgs e)
{
    StreamReader sr = new StreamReader("curs.txt");
    ceuro = double.Parse(sr.ReadLine());
    cdolar = double.Parse(sr.ReadLine());
    sr.Close();
    if (tb1.Text != "")
    {
        suma = double.Parse(tb1.Text);
        if (rb1.Checked == true) suma = suma * 1;
        if (rb2.Checked == true) suma = suma * ceuro;
        if (rb3.Checked == true) suma = suma * cdolar;
        l1.Text = suma.ToString() + " Lei";
        l2.Text = (suma / ceuro).ToString() + " Euro";
        l3.Text = (suma / cdolar).ToString() + " Dolari";
    }
    else MessageBox.Show("Introduceti suma");
}
}

```

Propuneri:

Completați aplicația și pentru alte valute.

Aplicația 3- MiniCalculator (informatică-matematică)

Realizați următoarea aplicație:

<pre> using System; using System.Collections.Generic; using System.ComponentModel; using System.Data; using System.Drawing; using System.Linq; using System.Text; using System.Windows.Forms; namespace AtestatVlad { public partial class MiniCalculatorForm : Form { public MiniCalculatorForm() { InitializeComponent(); } double op1; double op2; String s = ""; } } </pre>		<pre> private void Doibutton_Click(object sender, EventArgs e) { s+="2"; } </pre>
--	--	---

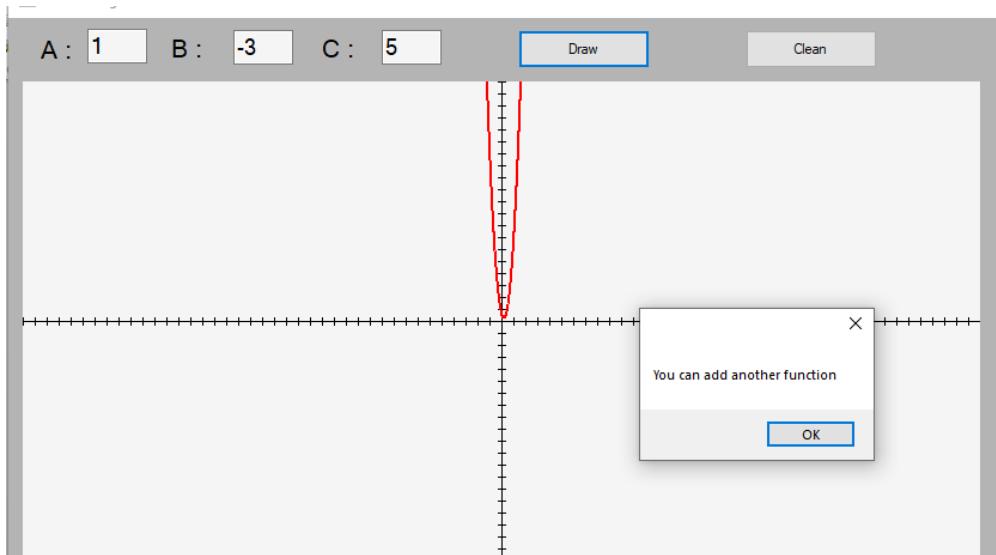
<pre> int operand = 0; private void Zerobutton_Click(object sender, EventArgs e) { s+="0"; TextBox1.Text=s; } private void Unobutton_Click(object sender, EventArgs e) { s+="1"; TextBox1.Text=s; } </pre>	<pre> TextBox1.Text=s; } private void Treobutton_Click(object sender, EventArgs e) { s+="3"; TextBox1.Text=s; } private void Patrobutton_Click(object sender, EventArgs e) { s+="4"; TextBox1.Text=s; } </pre>
<pre> private void Cincibutton_Click(object sender, EventArgs e) { s+="5"; TextBox1.Text=s; } private void Sasebutton_Click(object sender, EventArgs e) { s+="6"; TextBox1.Text=s; } private void Saptebutton_Click(object sender, EventArgs e) { s+="7"; TextBox1.Text=s; } private void Optbutton1_Click(object sender, EventArgs e) { s+="8"; TextBox1.Text=s; } private void Nouabutton_Click(object sender, EventArgs e) { s+="9"; TextBox1.Text=s; } private void </pre>	<pre> private void Egalbutton_Click(object sender, EventArgs e) { op2 = Double.Parse(s); s = ""; TextBox1.Text = s; if (operand == 1) s+= (double)op1 + (double)op2; else if (operand == 2) s+= (double)op1 - (double) op2; else if (operand == 3) s += (double)op1 * (double)op2; else { if(op2.CompareTo(0)==0) s="Err- Division by zero"; else s+=(double)op1/(double)op2; } TextBox1.Text=s; } private void virgulaButton_Click(object sender, EventArgs e) { s += ","; TextBox1.Text = s; } </pre>

<pre> Sumabutton_Click(object sender, EventArgs e) { operand = 1; op1 = Double.Parse(s); s = ""; TextBox1.Text = s; } private void Scaderebutton_Click(object sender, EventArgs e) { operand = 2; op1 = Double.Parse(s); s = ""; TextBox1.Text = s; } private void Produsbutton_Click(object sender, EventArgs e) { operand = 3; op1 = Double.Parse(s); s = ""; TextBox1.Text = s; } private void Impartirebutton_Click(object sender, EventArgs e) { operand = 4; op1 = Double.Parse(s); s = ""; TextBox1.Text = s; } private void Clearbutton_Click(object sender, EventArgs e) { s = ""; TextBox1.Clear(); } </pre>	<pre> private void Sqrtbutton_Click(object sender, EventArgs e) { operand = 5; op1 = Double.Parse(s); if (op1 >= 0) { double c = Math.Sqrt(op1); s = ""; s += c; } else s = "Err - sqrt from negative number"; TextBox1.Text = s; } private void signButton_Click(object sender, EventArgs e) { s += "-"; TextBox1.Text = s; } private void Exitbutton_Click(object sender, EventArgs e) { Close(); } </pre>
--	---

Propuneri:

Adăugați și alte funcții pentru aplicația MiniCalculator.

Aplicația 4- Graficul funcției de gradul II (informatică-matematică)



Realizați aplicația din imaginea de mai sus. Descoperiți cum a fost realizată studiind codul sursă de mai jos.

Codul sursă este:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace Graph2
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        int[] x = new int[1000];
        int[] y = new int[1000];
        int[] xl = new int[3] { 10, 280, 550 };
        int[] yl = new int[3] { 20, 50, 80 };
        Label[] lbl = new Label[10];
        Graphics g;
        Graphics g2;
        bool OK;
        Pen p = new Pen(Color.Black, 2);
        Pen p2 = new Pen(Color.Black, 1);
        Color axiscolor = Color.Black;
```

```

Color[] color = new Color[10] { Color.Red, Color.Green, Color.Orange,
Color.Cyan, Color.Blue, Color.DarkCyan, Color.Lime, Color.Purple,
Color.Pink, Color.Yellow };
// V-varful parabolei
int a, b, c, delta, Vx, Vy, i, j, col=-1 , k;
string f;

private void Form1_Load(object sender, EventArgs e)
{
    g = this.panel1.CreateGraphics();
    CreateLegend();
}
void DrawAxis()
{
    //desenarea axelor Ox si Oy
    p2.Color = axiscolor;
    g.DrawLine(p2, 0, 200, 800, 200);
    g.DrawLine(p2, 400, 0, 400, 400);
    for (i = 0; i <= 800; i += 10)
    {
        g.DrawLine(p2, i, 197, i, 203);
    }
    for (i = 0; i <= 400; i += 10)
    {
        g.DrawLine(p2, 397, i, 403, i);
    }
}

private void button1_Click(object sender, EventArgs e)
{

    CheckInput();
    if (col == 8)
    {
        OK = true;
    if (MessageBox.Show("Clean the sheet before adding anymore functions",
    "Information", MessageBoxButtons.OKCancel, MessageBoxIcon.Information)
== DialogResult.OK)
    {
        button2.PerformClick();
        k = 0;
        Controls.Remove(legend);
        CreateLegend();
    }
    else
        OK = false;
    }
    if (OK)
    {

f = "f(x) = " + a.ToString() + "*x^2 + " + b.ToString() + "x + " +
c.ToString();
        DrawAxis();
        a *= -1; b *= -1; c *= -1;
        delta = b * b - 4 * a * c;
    }
}

```

```

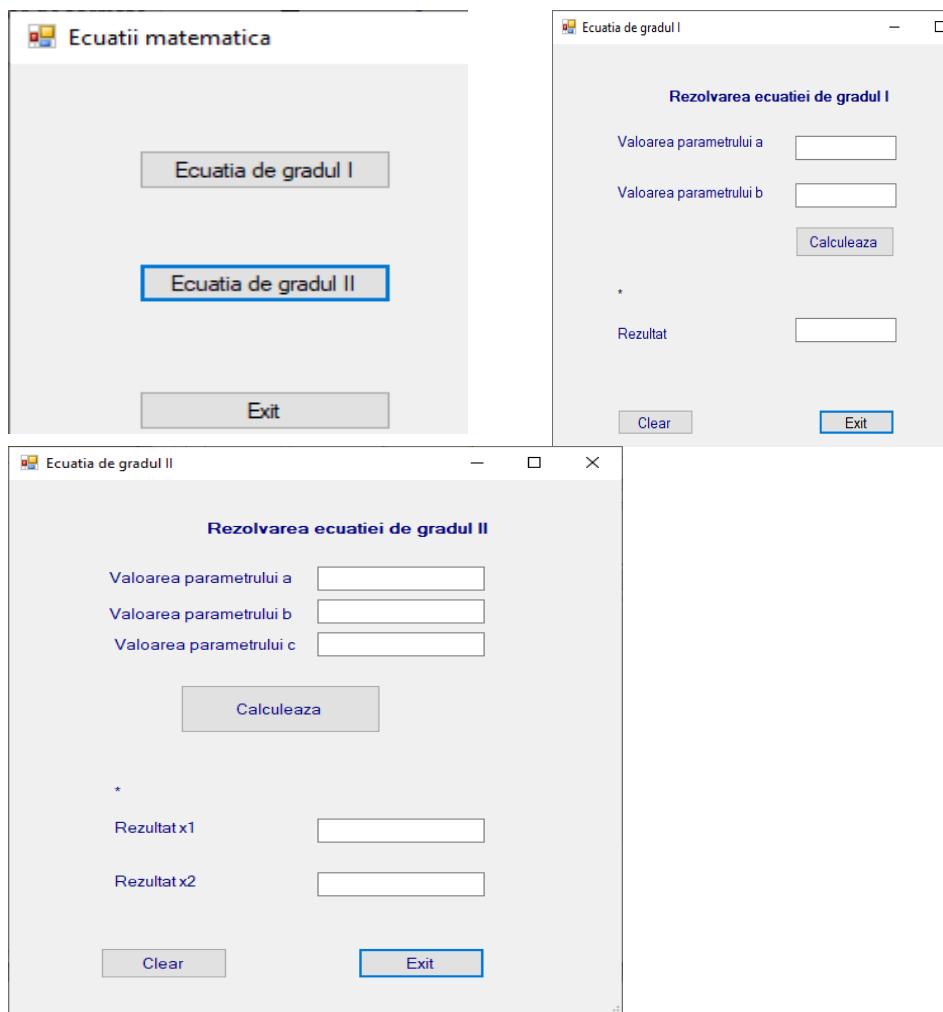
Vx = -b / (2 * a); Vy = delta / (4 * a);
for (i = Vx - 100, j = 1; i <= Vx + 100; i++, j++)
{
    x[j] = i + 400;
    y[j] = a * i * i + b * i + c + 200;
}
i = 1;
col++;
timer1.Start();
AddFunctionToLegend(f, color[col]);
}
else
    MessageBox.Show("Invalid imput", "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error);
}

private void timer1_Tick(object sender, EventArgs e)
{
    p.Color = color[col];
    if (i < 200)
        g.DrawLine(p, x[i], y[i], x[i + 1], y[i + 1]);
    else
{ timer1.Stop(); MessageBox.Show("You can add another function"); }
    i++;
}
private void button2_Click(object sender, EventArgs e)
{
    //stergere
    g.Clear(Color.WhiteSmoke);
    legend.Dispose();
    k = 0;
    DrawAxis();
    col = -1;
    CreateLegend();
}
void CheckImput()
{
    //verificarea datelor de intrare
    OK = true;
    try
    {
        a = Convert.ToInt32(textBox1.Text);
        b = Convert.ToInt32(textBox2.Text);
        c = Convert.ToInt32(textBox3.Text);
    }
    catch
    {
        OK = false;
    }
    if (textBox1.Text == "" || textBox1.Text == "0")
{
    textBox1.BackColor = Color.Red; OK = false; textBox1.Text = "";
}
    if (textBox2.Text == "")

```


Propuneri:

Realizați afișarea soluțiilor pentru rezolvarea ecuațiilor de gradul I și II după model.

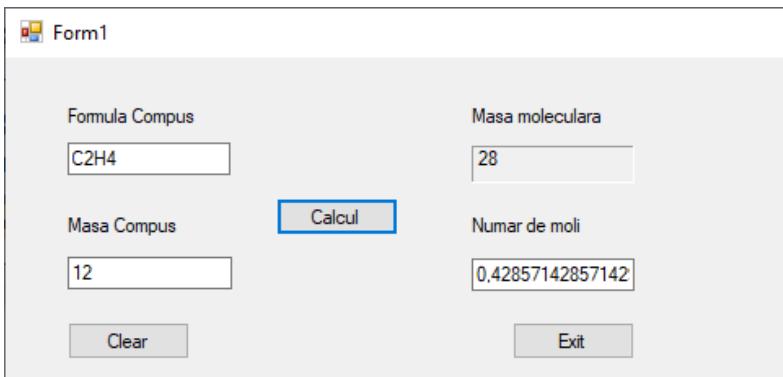
**Propuneri:**

Realizati o aplicatie WindowsForm care va contine o forma si controale specifice pentru rezolvarea urmatorului scenariu:

1. *Formular* de comanda de produse(Flori, pizza, carti, bilete la un spectacol etc) efectuata de un client si afisarea intr-un MessageBox a optiunilor alese de client;

Aplicația 5- Calculator molecular (informatică-chimie)

Realizați aplicația după forma de mai jos:



Realizați aplicația după forma de mai sus.

Codul sursă este:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace Calculator_molecular
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        private void buttonExit_Click(object sender, EventArgs e)
        {
            Close();
        }
        private void buttonClear_Click(object sender, EventArgs e)
        {
            textBoxFormula.Text = "";
            labelRezultat.Focus();
            textBoxMC.Text = "";
            textBoxNM.Text = "";
            labelRezultat.Text = "";
        }

        private void buttonCalculeaza_Click(object sender, EventArgs e)
        {
            int masa_moleculara = 0, m=0, p;
            string myString = textBoxFormula.Text.ToUpper();
```

```

        double NM;
        int i=0;
        while(i<myString.Length)
        {
            switch(myString[i])
            {
                case 'C':
                    m=12;
                    break;
                case 'H':
                    m=1;
                    break;
                case 'N':
                    m=14;
                    break;
                case 'O':
                    m=16;
                    break;
            }
            p=1;
            if (i + 1 < myString.Length)
            {
                if (myString[i + 1] != 'C' && myString[i + 1] != 'O' && myString[i + 1]
                != 'N' && myString[i + 1] != 'H')
                {
                    p = myString[i + 1] - '0';
                    i += 2;
                }
                else
                {
                    p = 1; i++;
                }
            }
            else
            {
                p = 1; i++;
            }
            masa_moleculara += m * p;
        }
        labelRezultat.Text = masa_moleculara.ToString();
        NM = double.Parse(textBoxMC.Text) / masa_moleculara;
        textBoxNM.Text = NM.ToString();
    }
private void textMasaFormula_TextChanged(object sender, EventArgs e)
{
    textBoxMC.Text = "";
    labelRezultat.Focus();
    labelRezultat.Text = "";
}
}

```

Aplicația 6- Tabelul periodic al elementelor(informatică-chimie)

Realizați aplicația de mai jos.

Pentru fiecare buton afișați informații despre elementul respectiv.



Codul sursă pentru elementul Hidrogen este:

```
private void button78_Click(object sender, EventArgs e)
{
    MessageBox.Show("Proprietati:\n Nemetal\nInclor\n ", "HIDROGEN");
}
```

Propuneri:

- Completati proiectul introducand pentru toate elementele tabelului informațiile specifice.
- Găsiți și alte metode de afișare a informațiilor pentru elementul selectat.

Aplicația 7- Legile fizicii (informatică-fizică)

Realizați aplicația de mai jos. Din meniu principal se poate alege căte o problemă referitoare la una din legile fizicii studiate.

Form1

Legea lui Ohm Legea Kirchhoff1 Legea lui Kirchhoff 2 Exit

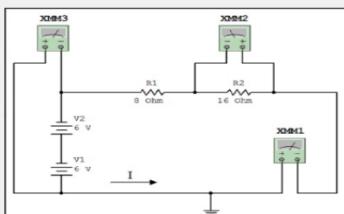
Legea lui Kirchhoff 1

Se dă:

R1	<input type="text"/>	Ohm
R2	<input type="text"/>	Ohm
V1	<input type="text"/>	V
V2	<input type="text"/>	V

Se cere:

U	<input type="text"/>	V
U2	<input type="text"/>	V
I	<input type="text"/>	A
RE	<input type="text"/>	Ohm



Zoom In Calculateaza Clear Exit
Zoom Out

Form2

Se aplică **prima lege a lui Kirchhoff** pentru:

nodul D:

$$I_4 - I_3 - I_8 = 0$$

$$I_3 = I_4 - I_8$$

nodul A:

$$I_7 - I_1 - I_6 = 0$$

$$I_6 = I_7 - I_1$$

Aplicam **a doua lege a lui Kirchhoff** pentru ochiul de retea ABCDFGA și alegem drept sens de parcurs sensul acelor de ceasornic și obținem:

$$I_1R_1 + I_2R_2 + I_3R_3 + I_4R_4 + I_5R_5 - I_6R_6 = E_1 - E_2 - E_3 + E_4$$

Zoom In Zoom Out

Zoom In Zoom Out

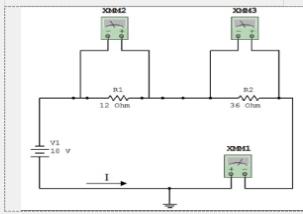
Form3

Se dă:

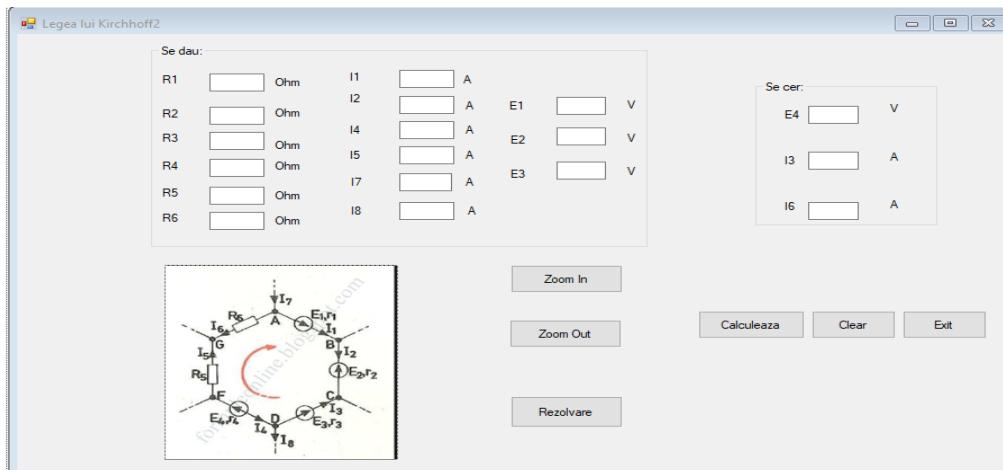
R1	<input type="text"/>	Ohm
R2	<input type="text"/>	Ohm
V1	<input type="text"/>	V

Se cere:

U1	<input type="text"/>	V
U2	<input type="text"/>	V
I	<input type="text"/>	A
RE	<input type="text"/>	Ohm



Zoom In Calculateaza Clear Exit
Zoom Out



Codul sursă pentru Legea lui Kirchhoff 2 ar fi:

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace WFA_Legile_fizicii
{
    public partial class FormKirchhoff2 : Form
    {
        private int zoom;
        private Form2 f2;

        public FormKirchhoff2()
        {
            InitializeComponent();
        }
        private void FormKirchhoff2_Load(object sender, EventArgs e)
        {
            Form f = new Form();
            f.MdiParent = this;
            f.Show();
        }
        private void btnExit_Click(object sender, EventArgs e)
        {
            Close();
        }
        private void btnClear_Click(object sender, EventArgs e)
        {
            textBoxR1.Text = "";
            textBoxR2.Text = "";
            textBoxR3.Text = "";
        }
    }
}

```

```

        textBoxR4.Text = "";
        textBoxR5.Text = "";
        textBoxR6.Text = "";
        textBoxI1.Text = "";
        textBoxI2.Text = "";
        textBoxI4.Text = "";
        textBoxI5.Text = "";
        textBoxI7.Text = "";
        textBoxI8.Text = "";
        textBoxI6.Focus();
        textBoxI3.Focus();
        textBoxE4.Focus();
        textBoxI6.Text = "";
        textBoxI3.Text = "";
        textBoxE4.Text = "";
    }
    private void btnCalculeaza_Click(object sender, EventArgs e)
    {
        double I3, I6, E4;
        double e1, e2, e3, e4, e5, e6;
        I3 = (double.Parse(textBoxI4.Text) - double.Parse(textBoxI8.Text));
        I6 = (double.Parse(textBoxI7.Text) - double.Parse(textBoxI1.Text));
        e1=(double.Parse(textBoxI1.Text) * double.Parse(textBoxR1.Text));
        e2=(double.Parse(textBoxI2.Text) * double.Parse(textBoxR2.Text));
        e3=(I3 *double.Parse(textBoxR3.Text));
        e4=(double.Parse(textBoxI4.Text) * double.Parse(textBoxR4.Text));
        e5=(double.Parse(textBoxI5.Text) * double.Parse(textBoxR5.Text));
        e6=(I6 * double.Parse(textBoxR6.Text));
        E4 = (double.Parse(textBoxE2.Text) + double.Parse(textBoxE3.Text) -
        double.Parse(textBoxE1.Text)+e1+e2-e3-e4+e5-e6);
        textBoxI3.Text = I3.ToString();
        textBoxI6.Text = I6.ToString();
        textBoxE4.Text = E4.ToString();
    }
    private void button1_Click(object sender, EventArgs e)
    {
        zoom = 6;
        int W = pictureBox2.Size.Width + zoom;
        int H = pictureBox2.Size.Height + zoom;
        pictureBox2.ClientSize = new Size(W, H);
    }
    private void button2_Click(object sender, EventArgs e)
    {
        zoom = -6;
        int W = pictureBox2.Size.Width + zoom;
        int H = pictureBox2.Size.Height + zoom;
        pictureBox2.ClientSize = new Size(W, H);
    }
    private void buttonRezolvare_Click(object sender, EventArgs e)
    {
        f2 = new Form2();
        f2.ShowDialog();
    }
}

```

Codul sursă pentru legea lui OHM este:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace WFA_Legile_fizicii
{
    public partial class Form2_Ohm : Form
    {
        private int zoom;
        public Form2_Ohm()
        {
            InitializeComponent();
        }
        private void Form2_Ohm_Load(object sender, EventArgs e)
        {
            Form f = new Form();
            f.MdiParent = this;
            f.Show();
        }
        private void button2_Click(object sender, EventArgs e)
        {
            zoom = -6;
            int W = pictureBox1.Size.Width + zoom;
            int H = pictureBox1.Size.Height + zoom;
            pictureBox1.ClientSize = new Size(W, H);
        }
        private void btnClear_Click(object sender, EventArgs e)
        {
            textBoxR1.Text = "";
            textBoxR2.Text = "";
            textBoxV1.Text = "";
            textBoxRE.Focus();
            textBoxI.Focus();
            textBoxU2.Focus();
            textBoxU1.Focus();

            textBoxRE.Text = "";
            textBoxI.Text = "";
            textBoxU2.Text = "";
            textBoxU1.Text = "";
        }
        private void btnCalculeaza_Click(object sender, EventArgs e)
        {
            double RE;
            double U1,U2,I;
```

```

        I =
(double.Parse(textBoxV1.Text))/(double.Parse(textBoxR1.Text)+double.Parse(textBoxR2.Text));
        U1 = I * double.Parse(textBoxR1.Text);
RE=(double.Parse(textBoxR1.Text)+double.Parse(textBoxR2.Text));
        U2=I*double.Parse(textBoxR2.Text);
        textBoxI.Text = I.ToString();
        textBoxRE.Text=RE.ToString();
        textBoxU1.Text=U1.ToString();
        textBoxU2.Text=U2.ToString();
    }
private void button1_Click_1(object sender, EventArgs e)
{
    zoom = 6;
    int W = pictureBox1.Size.Width + zoom;
    int H = pictureBox1.Size.Height + zoom;
    pictureBox1.ClientSize = new Size(W, H);
}
}
}
}

```

Codul sursa pentru Legea lui Kirchhoff 1 este:

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace WFA_Legile_fizicii
{
    public partial class FormKirchhoff1 : Form
    {

        private int zoom;
        public FormKirchhoff1()
        {
            InitializeComponent();
        }

        private void FormKirchhoff1_Load(object sender, EventArgs e)
        {
            Form f = new Form();
            f.MdiParent = this;
            f.Show();
        }

        private void button2_Click(object sender, EventArgs e)
        {
            zoom = -6;
            int W = pictureBox2.Size.Width + zoom;
            int H = pictureBox2.Size.Height + zoom;

```

```

        pictureBox2.ClientSize = new Size(W, H);
    }

    private void button1_Click(object sender, EventArgs e)
    {
        zoom = 6;
        int W = pictureBox2.Size.Width + zoom;
        int H = pictureBox2.Size.Height + zoom;
        pictureBox2.ClientSize = new Size(W, H);
    }
    private void btnClear_Click(object sender, EventArgs e)
    {
        textBoxR1.Text = "";
        textBoxR2.Text = "";
        textBoxV1.Text = "";
        textBoxRE.Focus();
        textBoxI.Focus();
        textBoxU2.Focus();
        textBoxU.Focus();
        textBoxRE.Text = "";
        textBoxI.Text = "";
        textBoxU2.Text = "";
        textBoxU.Text = "";
    }

    private void btnCalculeaza_Click(object sender, EventArgs e)
    {
        double RE;
        double U,U1,U2, I;

        U = double.Parse(textBoxV1.Text) +
double.Parse(textBoxV2.Text);
        I = U / (double.Parse(textBoxR1.Text) +
double.Parse(textBoxR2.Text));
        RE = (double.Parse(textBoxR1.Text) +
double.Parse(textBoxR2.Text));
        U1 = I * double.Parse(textBoxR1.Text);
        U2 = I * double.Parse(textBoxR2.Text);

        textBoxI.Text = I.ToString();
        textBoxRE.Text = RE.ToString();
        textBoxU.Text = U.ToString();
        textBoxU2.Text = U2.ToString();
    }
}
}

```

Propuneri:

- Compleați proiectul pentru celelalte forme la care nu am prezentat codul.
- Inserați după model și alte forme în care să folosiți legile fizicii.

Aplicația 8- Calculul lungimii de undă (informatică-fizică)

Realizați următoarea aplicație după forma de mai jos:



Codul sursă pentru butonul calculează este:

```
private void button1_Click(object sender, EventArgs e)
{
    double W, W1, W2;
    double electron;
    double m, h, e0, c, lam;
    electron = -1.6 * Math.Pow(10, -19);
    c=W1 = 3*Math.Pow(10, 8);
    m = 9.1 * Math.Pow(10, -31);
    e0 = 8.86 * Math.Pow(10, -12);
    h = 6.6 * Math.Pow(10, -34);
    W1 = (-1 * (Math.Pow(electron, 4) * m) / 8 * Math.Pow(e0,
2) * h) / Math.Pow(double.Parse(textBoxNR1.Text), 2);
    W2 = (-1 * (Math.Pow(electron, 4) * m) / 8 * Math.Pow(e0,
2) * h) / Math.Pow(double.Parse(textBoxNR2.Text), 2);
    W = W2 - W1;

    lam = (c / W) * h;
    labelRezultat.Text=lam.ToString();
}
```

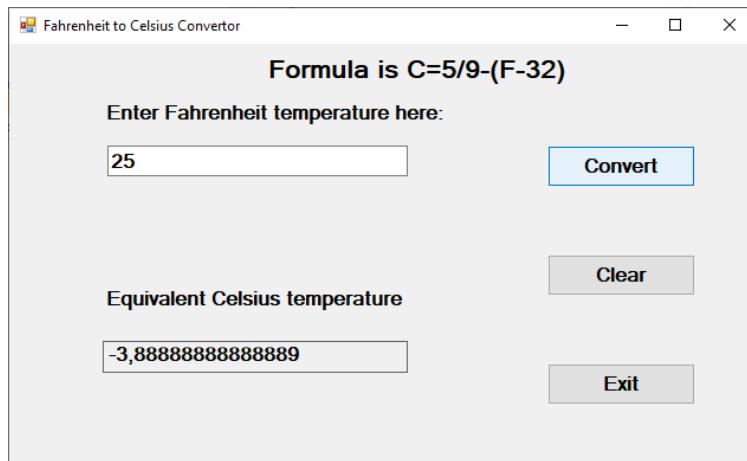
Propuneri:

Folosind Modulul “*Controale de tip : butoane, label, textbox, radiobutton,checkbox*” realizati o aplicatie WindowsForm care va contine o forma si controale specifice pentru rezolvarea urmatorului scenariu:

- 1.. *Formular* de inscriere pe site-ul tarom.ro si afisarea intr-un MessageBox a optiunilor alese de client;
3. *Formular* de inscriere pe site-ul unei agentii imobiliare pentru achitzionarea unui apartament si afisarea intr-un MessageBox a optiunilor alese de client;

Aplicația 9- Convertor de temperaturi (informatică-fizică)

Realizați următoarea aplicație după forma de mai jos:



Codul sursă pentru butonul **Convert** este:

```
private void btnConvert_Click(object sender, EventArgs e)
{
    double C;
    double F;
    F = double.Parse(textBoxFahrenheit.Text);
    C = 5.0 / 9.0 * (F - 32);
    labelCelcius.Text = C.ToString();
}
```

Pentru a șterge informația afișată funcția este:

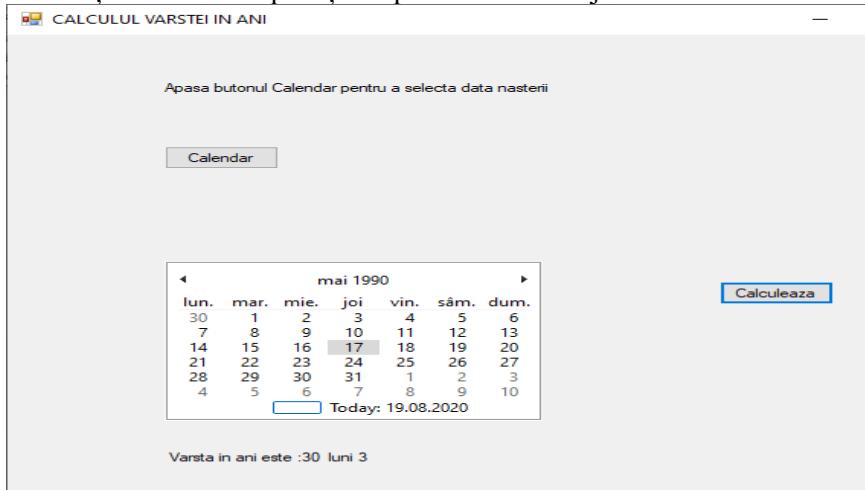
```
private void btnClear_Click(object sender, EventArgs e)
{
    labelCelcius.Text = "";
    textBoxFahrenheit.Text = "";
    textBoxFahrenheit.Focus();
}
```

Propuneri:

- Completăți codul sursă cu restul codului necesar funcționării aplicației.
- Adăugați aplicației și conversia inversă din Celsius în Fahrenheit a temperaturii.

Aplicația 10- Calculul vârstei (informatică-matematică)

Realizați următoarea aplicație după forma de mai jos:



Controlul **MonthCalendar** permite vizualizarea sau setarea informațiilor de tip data. Este un obiect de clasă **MonthCalendar**. Câteva proprietăți specifice ar fi:

- **CalendarDimensions** – stabilește dimensiunile controlului de tip **Calendar**;
- **FirstDayOfWeek** – stabilește prima zi din săptămână. Implicit este duminică;
- **MaxDate** – marchează ultima dată reprezentabilă pe calendar;
- **MinDate** – marchează prima dată reprezentabilă pe calendar;
- **SelectionRange** – marchează numărul maxim de zile ce pot fi selectate în control.
- **Start** – furnizează data de început a zonei selectate în control.
- **ShowToday** – precizează dacă data curentă este sau nu afișată în partea de jos a controlului.
- **TodayDate** – furnizează data curentă.

Proprietățile de tip dată, sunt manipulate prin intermediul unor obiecte de clasă **DateTime**. Câteva proprietăți ale acestei clase, mai des utilizate sunt:

- **Date** – furnizează data conținută în instanța curentă a obiectului;
- **Day** – furnizează numărul zilei din lună conținută în instanța curentă a obiectului;
- **DayOfWeek** – furnizează numărul zilei din săptămână conținută în instanța curentă a obiectului;
- **DayOfYear** – furnizează numărul zilei din an conținută în instanța curentă a obiectului;
- **Hour** – furnizează ora conținută în instanța curentă a obiectului;
- **Minute** – furnizează minutul conținută în instanța curentă a obiectului;
- **Month** – furnizează luna conținută în instanța curentă a obiectului;
- **Now** – furnizează un obiect care conține data și ora curentă a computerului, setate la nodul de reprezentare locală.
- **Second** – furnizează secunda conținută în instanța curentă a obiectului;

- **Today** – furnizează data curentă;
 - **Year** – furnizează anul conținut în instanța curentă a obiectului;
- Câteva evenimente produse controlul **MonthCalendar** sunt:
- **DateChanged** – se produce la schimbarea datei din calendar;
 - **DateSelected** – se produce atunci când utilizatorul realizează o selecție explicită cu mouse-ul a unor date;

Codul sursă este;

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace WindowsFormsApplication1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            monthCalendar1.Hide();
            label2.Hide();
            button2.Hide();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            monthCalendar1.Show();
            button2.Show();

        }
        private void button2_Click(object sender, EventArgs e)
        {
            DateTime data_nasterii =
monthCalendar1.SelectionRange.Start;
            var azi = DateTime.Today;
            int varsta = azi.Year - data_nasterii.Year;
            int nrluni = azi.Month - data_nasterii.Month;
            if (nrluni < 0)
                nrluni = 12 - data_nasterii.Month + azi.Month;
                if (data_nasterii.AddYears(varsta) > azi)
                    varsta--;
            label2.Text = "Varsta in ani este :" +
varsta.ToString()+" luni "+nrluni.ToString();
            label2.Show();        }    }}}
```

Aplicația 11- Calculul vârstei în ani (informatică-matematică)

Noțiuni : controlul **MonthCalendar**, ascunderea/reafisarea controalelor, lucrul cu date **DateTime**

COD SURA

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

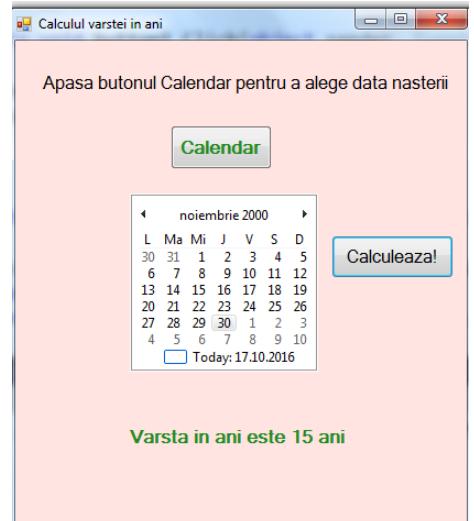
namespace WindowsFormsApplication1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            monthCalendar1.Hide();
            button2.Hide();
            label2.Hide();
        }

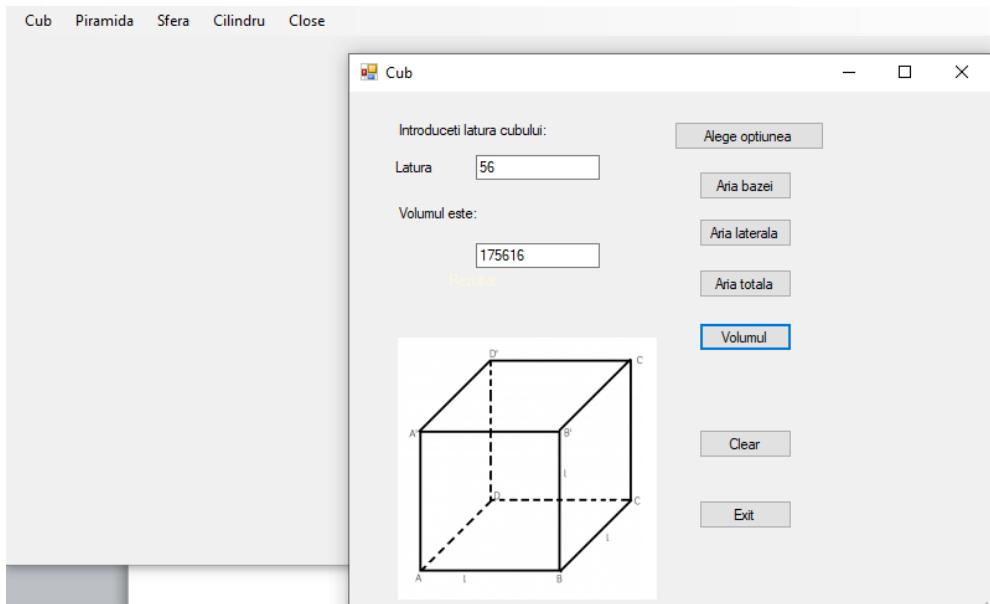
        private void button1_Click(object sender, EventArgs e)
        {
            monthCalendar1.Show();
            button2.Show();
        }

        private void button2_Click(object sender, EventArgs e)
        {
            DateTime data_nasterii = monthCalendar1.SelectionRange.Start;
            var azi = DateTime.Today;
            int varsta = azi.Year - data_nasterii.Year;
            if (data_nasterii.AddYears(varsta) > azi)
                varsta--;
            label2.Show();
        }

        label2.Text = "Varsta in ani este "+varsta.ToString()+" ani";
    }
}
```



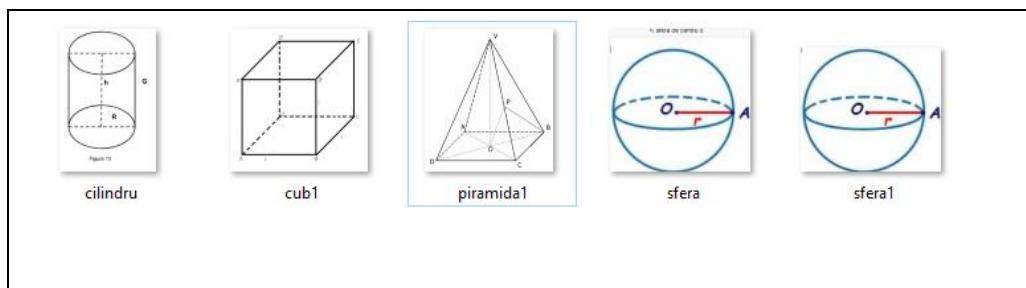
Aplicația 12- Corpuri geometrice (informatică-matematică)



Aplicația de mai sus conține un MeniuStreep cu cele 5 taburi la care am dat numele corpurilor geometrice prezentate în aplicație. Fiecare tab deschide o formă specifică corpului geometric ales.

În funcție de datele introduse pentru corpul geometric specificat se afișează rezultatele cerute.

Pentru aceasta aplicație am avut nevoie de imaginile unui cub, al unui cilindru, al unei piramide și al unei sfere ce sunt preluate de un PictureBox în fiecare formă.



Codul sursă pentru forma **Cub** este:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

```

using System.Windows.Forms;

namespace WindowsFormsCalcul_Corpuri_Geometrice
{
    public partial class Form2 : Form
    {
        double l,r;
        public Form2()
        {
            InitializeComponent();
        }

        private void ButtonArieC_Click(object sender, EventArgs e)
        {
//înainte de actionarea butonului Arie textBoxLaturaC
//este invisibil
            textBoxLaturaC.Visible = false;
//la actionarea butonului Arie textBoxLaturaC devine visibil
            textBoxLaturaC.Visible = true;
//valoarea introdusa in textBoxLaturaC care se considera text
//se converteste cu functia Parse, pentru noi la tipul double

            l = double.Parse(textBoxLaturaC.Text);
            r = l * l;//se calculeaza raza
            label3.Show();
            textBoxRezultatC.Show();//textBoxRezultatC apare pe forma
                //in label3 se afiseaza mesajul „Aria bazei este:”
            label3.Text = "Aria bazei este:";
//se converteste la tipul text rezultatul si se afiseaza in
//textBoxRezultatC
            textBoxRezultatC.Text = r.ToString();
            label2.Show();
        }

        private void Button4_Click(object sender, EventArgs e)
        {
            Close();
        }

        private void Button5_Click(object sender, EventArgs e)
        {
            label4.Show();
            buttonArieC.Show();
            button2.Show();
            button1.Show();
            buttonVolumC.Show();
            label4.Text = "Introduceti latura cubului:";
            textBoxLaturaC.Show();
            label1.Show();
        }

        private void Form2_Load(object sender, EventArgs e)
        {
            label3.Hide();
            label4.Hide();
            textBoxLaturaC.Hide();
            label1.Hide();
        }
    }
}

```

```

        textBoxRezultatC.Hide();
        label2.Hide();
        buttonArieC.Hide();
        button2.Hide();
        button1.Hide();
        buttonVolumC.Hide();
        label5.Hide();

    }
    private void Button2_Click(object sender, EventArgs e)
    {
        textBoxLaturaC.Visible = false;
        textBoxLaturaC.Visible = true;
        l = double.Parse(textBoxLaturaC.Text);
        r = 4*l * l;
        label3.Show();
        textBoxRezultatC.Show();
        label3.Text = "Aria laterală este:" ;
        textBoxRezultatC.Text = r.ToString();
        label2.Show();
    }
    private void Button1_Click(object sender, EventArgs e)
    {
        textBoxLaturaC.Visible = false;
        textBoxLaturaC.Visible = true;
        l = double.Parse(textBoxLaturaC.Text);
        r = 6*l * l;
        label3.Show();
        textBoxRezultatC.Show();
        label3.Text = "Aria totală este:" ;
        textBoxRezultatC.Text = r.ToString();
        label2.Show();
    }
    private void ButtonVolumC_Click(object sender, EventArgs e)
    {
        textBoxLaturaC.Visible = false;
        textBoxLaturaC.Visible = true;
        l = double.Parse(textBoxLaturaC.Text);
        r = l * l*l;
        label3.Show();
        textBoxRezultatC.Show();
        label3.Text = "Volumul este:" ;
        textBoxRezultatC.Text = r.ToString();
        label2.Show();
    }
    private void Button3_Click(object sender, EventArgs e)
    {
        textBoxLaturaC.Clear();
        textBoxRezultatC.Clear();
        label3.Visible = true;
    }
}

```

Codul sursă pentru forma **Piramidă** este:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace WindowsFormsCalcul_Corpuri_Geometrice
{
    public partial class Form3 : Form
    {
        public Form3()
        {
            InitializeComponent();
        }

        private void Form3_Load(object sender, EventArgs e)
        {
            label3.Hide();
            label4.Hide();
            textBoxLaturaP.Hide();
            label1.Hide();
            textBoxRezultatP.Hide();
            label2.Hide();
            button3.Hide();
            button2.Hide();
            button1.Hide();
            button4.Hide();
            button5.Hide();
            buttonATP.Hide();
            label5.Hide();
            textBoxInaltimeP.Hide();
            buttonVolumP.Hide();
        }

        private void Button6_Click(object sender, EventArgs e)
        {
            label4.Text = "Alegeti una din urmatoarele optiuni:";
            button1.Show();
            button2.Show();
            button3.Show();
            buttonATP.Show();
            button4.Show();
            button5.Show();
            button6.Show();
            buttonVolumP.Show();
            label4.Text = "Introduceti latura si inaltimea piramidei:";
            textBoxLaturaP.Show();
            label1.Show();
            label2.Show();
            textBoxInaltimeP.Show();
            label4.Show();
        }
    }
}
```

```

}

double lp, hp, rp,pb,ap,AL,Ab;//declaram variabilele

private void Button3_Click(object sender, EventArgs e)
{
    textBoxLaturaP.Visible = true;
    textBoxInaltimeP.Visible = true;
    lp = double.Parse(textBoxLaturaP.Text);
    hp = double.Parse(textBoxInaltimeP.Text);
    rp = lp * lp;
    Ab = rp;
    label3.Show();
    label2.Show();
    textBoxRezultatP.Show();
    label5.Visible = true;
    label5.Text = "Aria bazei piramidei este:";
    textBoxRezultatP.Text = rp.ToString();

}
private void Button4_Click(object sender, EventArgs e)
{

    textBoxLaturaP.Visible = true;
    textBoxInaltimeP.Visible = true;
    lp = double.Parse(textBoxLaturaP.Text);
    hp = double.Parse(textBoxInaltimeP.Text);
    rp = 4*lp;
    label3.Show();
    label2.Show();
    textBoxRezultatP.Show();
    label5.Visible = true;
    label5.Text = "Perimetru bazei piramidei este:";
    textBoxRezultatP.Text = rp.ToString();
}
private void Button5_Click(object sender, EventArgs e)
{
    textBoxLaturaP.Visible = true;
    textBoxInaltimeP.Visible = true;
    lp = double.Parse(textBoxLaturaP.Text);
    hp = double.Parse(textBoxInaltimeP.Text);
    pb = 4 * lp;
    ap = Math.Sqrt(hp * hp + (lp / 2) * (lp / 2));
    AL = pb * ap / 2;
    label3.Show();
    label2.Show();
    textBoxRezultatP.Show();
    label5.Visible = true;
    label5.Text = "Aria laterală a piramidei este:";
    textBoxRezultatP.Text = AL.ToString();
}
private void buttonATP_Click(object sender, EventArgs e)
{
    textBoxLaturaP.Visible = true;
    textBoxInaltimeP.Visible = true;
    lp = double.Parse(textBoxLaturaP.Text);
}

```

```

        hp = double.Parse(textBoxInaltimeP.Text);
        rp = AL+Ab;
        label3.Show();
        label2.Show();
        textBoxRezultatP.Show();
        label5.Visible = true;
        label5.Text = "Aria totala a piramidei este:";
        textBoxRezultatP.Text = rp.ToString();
    }
    private void buttonVolumP_Click(object sender, EventArgs e)
    {
        textBoxLaturaP.Visible = true;
        textBoxInaltimeP.Visible = true;
        lp = double.Parse(textBoxLaturaP.Text);
        hp = double.Parse(textBoxInaltimeP.Text);
        rp = Ab * hp/3;
        label3.Show();
        label2.Show();
        textBoxRezultatP.Show();
        label5.Visible = true;
        label5.Text = "Aria totala a piramidei este:";
        textBoxRezultatP.Text = rp.ToString();
    }
    private void Button2_Click(object sender, EventArgs e)
    {
        textBoxRezultatP.Clear();
        label3.Visible = true;
    }
    private void Button1_Click(object sender, EventArgs e)
    {
        Close();
    }
}
Codul sursă pentru forma Sferă este:
```

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace WindowsFormsCalcul_Corpuri_Geometrice
{
    public partial class Form4 : Form
    {
        public Form4()
        {
            InitializeComponent();
        }
        private void Form4_Load(object sender, EventArgs e)
        {
```

```

        textBoxRaza.Hide();
        label3.Hide();
        label4.Hide();
        label1.Hide();
        label2.Hide();
        button3.Show();
        button4.Hide();
        button5.Hide();
        textBoxRezultatC.Hide();
    }
    private void button3_Click(object sender, EventArgs e)
    {
        textBoxRaza.Show();
        label3.Show();
        label3.Text = "Introduceti raza sferei:";
        label1.Show();
        button4.Show();
        button5.Show();
    }
    double rz,pi=3.14,A,V;
    private void button1_Click(object sender, EventArgs e)
    {
        Close();
    }
    private void button2_Click(object sender, EventArgs e)
    {
        textBoxRezultatC.Clear();
        label4.Visible = true;
    }
    private void button4_Click(object sender, EventArgs e)
    {
        label2.Show();
        textBoxRaza.Visible = true;
        rz = double.Parse(textBoxRaza.Text);
        A = pi*rz*rz;
        textBoxRezultatC.Show();
        label4.Show();
        label4.Text = "Aria sferei este:";
        textBoxRezultatC.Text = A.ToString();
    }
    private void button5_Click(object sender, EventArgs e)
    {
        label2.Show();
        textBoxRaza.Visible = true;
        rz = double.Parse(textBoxRaza.Text);
        V = 4 * pi * rz * rz;
        textBoxRezultatC.Show();
        label4.Show();
        label4.Text = "Volumul sferei este:";
        textBoxRezultatC.Text = V.ToString();
    }
}
}

```

Codul sursă pentru formăl ce face legătură cu formele din meniu este:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
namespace WindowsFormsCalcul_Corpuri_Geometrice
{
    public partial class Form1 : Form
    {
        private Form2 f2;
        private Form3 f3;
        private Form4 f4;
        private Form5 f5;
        public Form1()
        {
            InitializeComponent();
        }
    private void CubToolStripMenuItem_Click(object sender, EventArgs e)
    {
        f2 = new Form2();
        f2.ShowDialog();
    }
    private void PiramidaToolStripMenuItem_Click(object sender, EventArgs e)
    {
        f3 = new Form3();
        f3.ShowDialog();
    }
    private void SferaToolStripMenuItem_Click(object sender, EventArgs e)
    {
        f4 = new Form4();
        f4.ShowDialog();
    }
    private void CiklindruToolStripMenuItem_Click(object sender, EventArgs e)
    {
        f5 = new Form5();
        f5.ShowDialog();
    }
    private void closeToolStripMenuItem_Click(object sender, EventArgs e)
    {
        Close();    }   }}
```

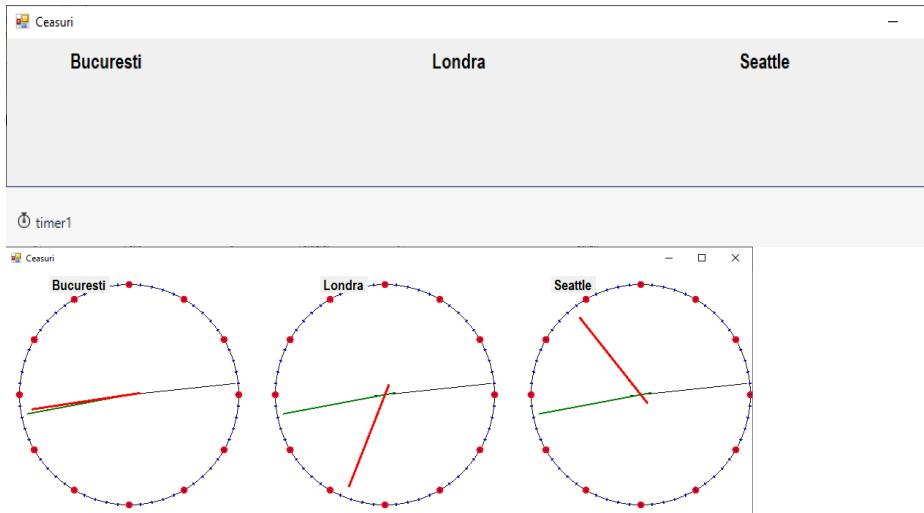
Propuneri:

1.Completați aplicația pentru forma Cilindru.

2.*Realizați un Formular* pentru conversia unui numar natural in baza 2 si in baza 8 si 16.

Aplicația 13- Ceasuri

Aplicația are ca scop afișarea orei pe glob simultan, în trei locații diferite.



Codul sursă este:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace ceasuri
{
    public partial class Form1 : Form
    {
        Graphics G;

        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Shown(object sender, EventArgs e)
        {
            G = this.CreateGraphics();
            timer1.Start();
        }

        private void timer1_Tick(object sender, EventArgs e)
        {
            G.Clear(Color.White); //Bucuresti
            ceas c1 = new ceas(170, 170, 150, DateTime.Now);
            c1.desenez(G); //Londra
            ceas c2 = new ceas(520, 170, 150, DateTime.Now.AddHours(-2));
            c2.desenez(G); //Seattle
            ceas c3 = new ceas(870, 170, 150, DateTime.Now.AddHours(-10));
            c3.desenez(G); } }
```

Aplicația 14- Ceas analogic

Realizați aplicația “Ceas analogic” după următoarea formă:

Codul sursă este:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

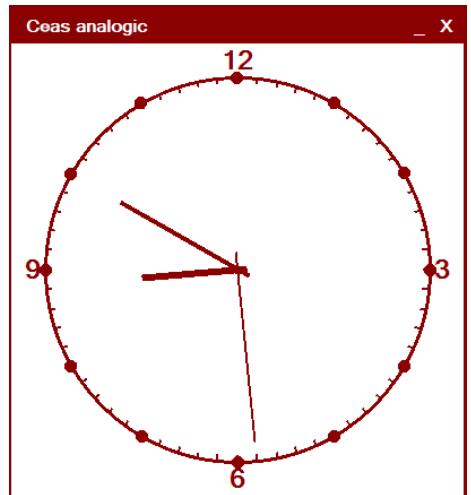
namespace ceas_analogic
{
    public partial class Form1 : Form
    {
        int w, h;
        float hh, mm,
ss;/ora,minut,secunda
        float r;//raza
        int x0, y0;//centrul
        Image img;
        Graphics g;
        System.Drawing.Font f;
        SizeF ms;

        public Form1()
        {
            InitializeComponent();
        }

        private void label1_Click(object sender, EventArgs e)
        {
            Application.Exit();
        }
        private void label3_Click(object sender, EventArgs e)
        {
            this.WindowState = FormWindowState.Minimized;
        }
        private void timer1_Tick(object sender, EventArgs e)
        {

            deseneaza();
        }
        private void Form1_Load(object sender, EventArgs e)
        {
            f = new System.Drawing.Font("Time New Roman", 18,
fontStyle.Bold);

            timer1.Start();
            deseneaza();
        }
    }
}
```



```

        }
        void deseneaza()
        {
            hh = DateTime.Now.Hour;
            mm = DateTime.Now.Minute;
            ss = DateTime.Now.Second;
            w = 400;
            h = 400;
            img = new Bitmap(w, h);
            g = Graphics.FromImage(img);
            x0 = w / 2;
            y0 = h / 2;
            r = 170;
            p.Location = new Point(2, 34);
            p.Width = w;
            p.Height = h;
            //sterg desenul
            g.Clear(Color.White);
            //cercul
            g.DrawEllipse(new Pen(Color.DarkRed,3), x0 - r, y0 - r, 2 * r, 2 * r);
            //orele 12, 6, 3, 9
            StringFormat sf = new StringFormat();
            sf.Alignment = StringAlignment.Center;
            ms = g.MeasureString("12", f);
            RectangleF r1 = new RectangleF(x0 - ms.Width / 2, y0 - r - ms.Height, ms.Width, ms.Height);
            g.DrawString("12", f, Brushes.DarkRed, r1, sf);
            ms = g.MeasureString("6", f);
            RectangleF r2 = new RectangleF(x0 - ms.Width / 2, y0 + r, ms.Width, ms.Height);
            g.DrawString("6", f, Brushes.DarkRed, r2, sf);
            ms = g.MeasureString("3", f);
            RectangleF r3 = new RectangleF(x0 + r, y0 - ms.Height / 2, ms.Width, ms.Height);
            g.DrawString("3", f, Brushes.DarkRed, r3, sf);
            ms = g.MeasureString("9", f);
            RectangleF r4 = new RectangleF(x0 - r - ms.Width, y0 - ms.Height / 2, ms.Width, ms.Height);
            g.DrawString("9", f, Brushes.DarkRed, r4, sf);
            //punctele minutelor
            for (int i = 0; i < 60; i++)
            {
                Point PP1 = new Point((int)(x0 + r * 0.97 * Math.Cos(i * Math.PI / 30)), (int)(y0 + r * 0.97 * Math.Sin(i * Math.PI / 30)));
                Point PP2 = new Point((int)(x0 + r * Math.Cos(i * Math.PI / 30)), (int)(y0 + r * Math.Sin(i * Math.PI / 30)));
                g.DrawLine(new Pen(Color.DarkRed, 2), PP1, PP2);
            }
            //punctele orelor
            for (int i = 0; i < 12; i++)
            {

```

```

Point Pp1 = new Point((int)(x0 + r * 0.92 * Math.Cos(i * Math.PI / 6)), (int)(y0 + r * 0.92 * Math.Sin(i * Math.PI / 6)));
Point Pp2 = new Point((int)(x0 + r * Math.Cos(i * Math.PI / 6)), (int)(y0 + r * Math.Sin(i * Math.PI / 6)));
//g.DrawLine(new Pen(Color.DarkRed, 4), Pp1, Pp2);
g.DrawEllipse(new Pen(Color.DarkRed, 5), Pp2.X-3,Pp2.Y-3,6,6);
}
hh = hh - 3 + mm / 60; mm = mm - 15; ss = ss -
15;
//orar
Point P1 = new Point((int)(x0 - r * 0.05 * Math.Cos(hh * Math.PI / 6)), (int)(y0 - r * 0.05 * Math.Sin(hh * Math.PI / 6)));
Point P2 = new Point((int)(x0 + r * 0.5 * Math.Cos(hh * Math.PI / 6)), (int)(y0 + r * 0.5 * Math.Sin(hh * Math.PI / 6)));
g.DrawLine(new Pen(Color.DarkRed, 6), P1, P2);
//minutar
Point P3 = new Point((int)(x0 - r * 0.07 * Math.Cos(mm * Math.PI / 30)), (int)(y0 - r * 0.07 * Math.Sin(mm * Math.PI / 30)));
Point P4 = new Point((int)(x0 + r * 0.7 * Math.Cos(mm * Math.PI / 30)), (int)(y0 + r * 0.7 * Math.Sin(mm * Math.PI / 30)));
g.DrawLine(new Pen(Color.DarkRed, 4), P3, P4);
//secundar
Point P5 = new Point((int)(x0 - r * 0.09 * Math.Cos(ss * Math.PI / 30)), (int)(y0 - r * 0.09 * Math.Sin(ss * Math.PI / 30)));
Point P6 = new Point((int)(x0 + r * 0.9 * Math.Cos(ss * Math.PI / 30)), (int)(y0 + r * 0.9 * Math.Sin(ss * Math.PI / 30)));
g.DrawLine(new Pen(Color.DarkRed, 2), P5, P6);
g.FillEllipse(Brushes.DarkRed, x0 - 4, y0 - 4, 8, 8);
this.Refresh();
}
private void p_Paint_1(object sender, PaintEventArgs e)
{
e.Graphics.DrawImage(img, 0, 0, w, h);
}  }}

```

Propuneri:

1. Realizați un *Formular* pentru calculul dobanzii unei sume ce trebuie împrumutata. Bancile la care se poate face împrumutul vor fi afisate intr-un grup cu butoane radio.

Aplicația 15- Crearea unui slideshow

Noțiuni : controalele **PictureBox** și **ImageList**, butoane de navigare în **ImageList**

Codul sursă:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace slideshow
{
    public partial class Form1 : Form
    {
        int x = 0;
        public Form1()
        {
            InitializeComponent();
        }

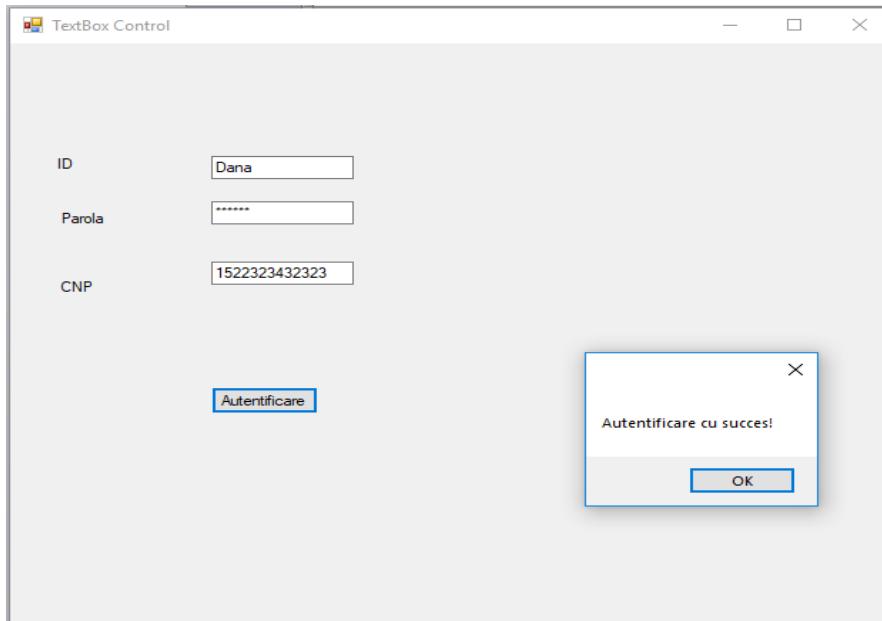
        private void btn_next_Click(object sender, EventArgs e)
        {
            x++;
            pictureBox1.Image = imageList1.Images[x % 4];
        }

        private void btn_prev_Click(object sender, EventArgs e)
        {
            x--;
            if(x < 0) x = 0;
            pictureBox1.Image = imageList1.Images[x % 4];
        }
    }
}
```



Aplicația 16- Formular de autentificare

Noțiuni : controlul **TextBox**, ascunderea/reafisarea controalelor, activarea/dezactivarea controalelor



Codul sursă:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
namespace TextBox
{
    public partial class TextBoxExample : Form
    {
        public TextBoxExample()
        {
            InitializeComponent();

            parolaTextBox.PasswordChar = '*';
            parolaTextBox.Enabled = false;
            CNPTextBox.Visible = false;
            cnplabel.Visible = false;
            AutentificareBtn.Visible = false;
```

```

        }
    private void idTextBox_PreviewKeyDown(object sender,
PreviewKeyDownEventArgs e)
{
    if (e.KeyCode == Keys.Tab)
        if (idTextBox.Text == "Dana")
            parolaTextBox.Enabled = true;
        else
            MessageBox.Show("Utilizatorul nu exista");
    }
    private void parolaTextBox_PreviewKeyDown(object
sender, PreviewKeyDownEventArgs e)
{
    if (e.KeyCode == Keys.Enter)
        if (parolaTextBox.Text == "parola")
        {
            cnplabel.Visible = true;
            CNPTextBox.Visible = true;
            AutentificareBtn.Visible = true;
            CNPTextBox.Focus();
        }
        else
        { MessageBox.Show("Parola incorecta");
            parolaTextBox.Clear();
        }
    }
private void CNPTextBox_KeyPress(object sender, KeyPressEventArgs e)
{
    if (e.KeyChar < 48 || e.KeyChar > 57)
        e.Handled = true;
}
private void AutentificareBtn_Click(object sender, EventArgs e)
{
    if (CNPTextBox.Text.Length != 13)
    {
        MessageBox.Show("CNP incorect, scrieti din nou!");
        CNPTextBox.Clear();
    }
    else
        MessageBox.Show("Autentificare cu succes!");      }
}
}

```

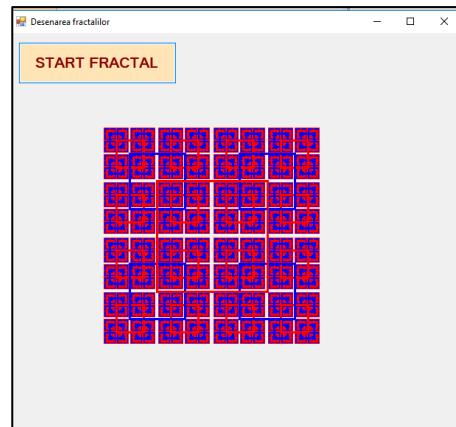
Aplicația 17- Generarea fractalilor

Noțiuni : Controlul Timer, obiecte grafice (Pen, Point)

COD SURSA

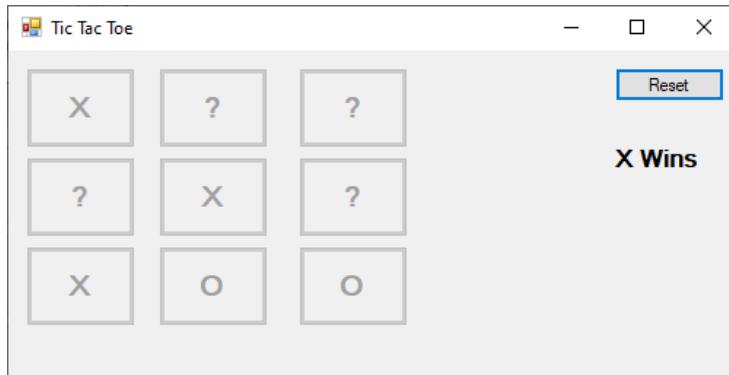
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace fractal
{
    public partial class Form1 : Form
    {
        int m=8;
        public Form1()
        {
            InitializeComponent();
        }
        void patrat(int n, int x, int y, int l)
        {
            if(n>1)
            {
                patrat(n - 1, x - l / 4, y - l / 4,1/2);
                patrat(n - 1, x +3* l / 4, y - l / 4,1/2);
                patrat(n - 1, x - l / 4, y+3* l / 4,1/2);
                patrat(n - 1, x +3* l / 4, y +3* l / 4,1/2);
            }
            Pen penc ;
            if (n % 2 == 0)
                penc = new Pen(Color.Red, 3);
            else
                penc = new Pen(Color.Blue, 3);
            Point[] p = new Point[4];
            p[0].X = x; p[0].Y = y;
            p[1].X = x; p[1].Y = y + 1;
            p[2].X = x + 1; p[2].Y = y + 1;
            p[3].X = x + 1; p[3].Y = y;
            Graphics g = this.CreateGraphics();
            g.DrawPolygon(penc,p);
        }
        private void timer1_Tick(object sender, EventArgs e)
        {
            if (m <= 8)
            {
                int x = 200, y = 200, l = 150;
                patrat(m, x, y, l);
                m = m + 1; } }
        private void btn_Start_Click(object sender, EventArgs e)
        { timer1.Start(); } }}
```



Fișă 6 – Jocuri

Jocul 1- Tic Tac Toe



Codul sursă este:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace TicTacToeGame
{
    public partial class Form1 : Form
    {
        Player currentPlayer;
        public Form1()
        {
            InitializeComponent();
        }
        public enum Player
        {
            X,
            O
        }
        private void buttonClick(object sender, EventArgs e)
        {
            var button = (Button)sender;
            currentPlayer = Player.X;
            button.Text = currentPlayer.ToString();
            button.Enabled = false;
            button.BackColor = System.Drawing.Color.LightGreen;
            Check();
            AiTimer.Start();

        }
    }
}
```

```

private void resetGame(object sender, EventArgs e)
{
    label1.Text = "??";
    foreach (Control x in this.Controls)
    {
        if (x is Button && x.Tag == "play")
        {
            ((Button)x).Enabled = true;
            ((Button)x).Text = "?";
            ((Button)x).BackColor = default(Color);
        }
    }
}
private void playAI(object sender, EventArgs e)
{
    foreach (Control x in this.Controls)
    {
        if (x is Button && x.Text == "?" && x.Enabled)
        {
            x.Enabled = false;
            currentPlayer = Player.O;
            x.Text = currentPlayer.ToString();
            x.BackColor =
System.Drawing.Color.LightGoldenrodYellow;
            AiTimer.Stop();
            Check();
            break;
        }
        else
        {
            AiTimer.Stop();
        }
    }
}
private void Check()
{
if (button1.Text == "X" && button2.Text == "X" && button3.Text == "X"
 ||
button4.Text == "X" && button5.Text == "X" && button6.Text == "X" ||
button7.Text == "X" && button8.Text == "X" && button9.Text == "X" ||
button2.Text == "X" && button5.Text == "X" && button8.Text == "X" ||
button3.Text == "X" && button6.Text == "X" && button9.Text == "X" ||
button1.Text == "X" && button5.Text == "X" && button9.Text == "X" ||
button3.Text == "X" && button5.Text == "X" && button7.Text == "X"
        )
    {
        WON();
        label1.Text = "X Wins";
    }
    else if(
button1.Text == "O" && button2.Text == "O" && button3.Text == "O" ||
button4.Text == "O" && button5.Text == "O" && button6.Text == "O" ||
button7.Text == "O" && button8.Text == "O" && button9.Text == "O" ||

```

```
button1.Text == "0" && button4.Text == "0" && button7.Text == "0" ||
button2.Text == "0" && button5.Text == "0" && button8.Text == "0" ||
button3.Text == "0" && button6.Text == "0" && button9.Text == "0" ||
button1.Text == "0" && button5.Text == "0" && button9.Text == "0" ||
button3.Text == "0" && button5.Text == "0" && button7.Text == "0" )
{
    WON();
    label1.Text = "O Wins";
}

}
private void WON()
{
    foreach (Control x in this.Controls)
    {
        if(x is Button && x.Tag=="play")
        {
            ((Button)x).Enabled = false;

            ((Button)x).BackColor = default(Color);
        }
    }
}
```

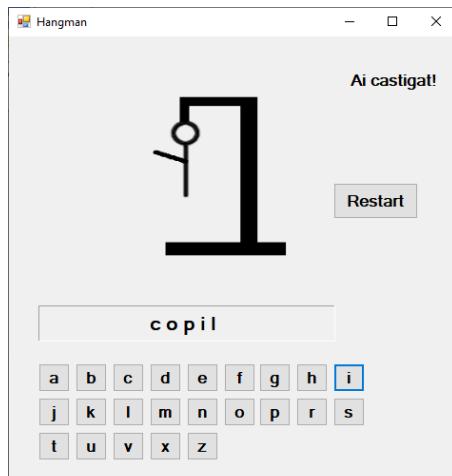
Propunerí:

1. Realizați un “Joc de atenie” – la incarcarea formei , se vor afisa 9 butoane cu numere generate aleator. Jucatorul trebuie sa apese butoanele in ordinea crescatoare a valorilor. La final, calculatorul afiseaza intr-un MessageBox timpul in care jucatorul a reusit sa apese corect cele 9 butoane (cu ajutorul unui control Timer) sau mesajul “Ai ratat, nu esti suficient de atent!”.
 2. Realizați jocul “Nu te supara frate ! » (problema „joc13” → .campion)

Jocul 2- Hangman

Hangman este un joc care constă în descoperirea unui cuvânt sau a unei propoziții și se joacă de obicei între doi jucători. Un jucător se gândește la un cuvânt, frază sau propoziție, iar celălalt încearcă să ghicească sugerând literele într-un număr limitat de pași.

Acest joc este util pentru îmbogățirea vocabularului și distractiv și se poate realiza ca o aplicație pe calculator în limbajul C#.



Aplicația conține un pictureBox, un label care v-a afișa rezultatul jocului, un buton de Restart, un textBox pentru afișarea cuvântului și butoanele ce conțin literele alfabetului.

Fiecare buton apăsat va afișa litera respectivă, dacă există, în toate locurile din cuvânt/propoziție.

Codul sursă este:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.IO;

namespace Hangman
{
    public partial class FormHangman : Form
    {
        private Bitmap[] hangImages = {
global::Hangman.Properties.Resources.Hangman_0,
            global::Hangman.Properties.Resources.Hangman_1,
            global::Hangman.Properties.Resources.Hangman_2,
            global::Hangman.Properties.Resources.Hangman_3,
            global::Hangman.Properties.Resources.Hangman_4,
            global::Hangman.Properties.Resources.Hangman_5,
            global::Hangman.Properties.Resources.Hangman_6 };
        private int wrongGuesses = 0;
        private string Current = "";
        private string[] words;
        private string copyCurrent = "";
```

```

public FormHangman()
{
    InitializeComponent();
}

private void loadwords()
{
    char[] delimiterChars = { ',' };
string[] readText = File.ReadAllLines("Commonwords2.txt");
    words = new string[readText.Length];
    int index = 0;
    foreach (string s in readText)
    {
        string[] line = s.Split(delimiterChars);
        words[index++] = line[1];
    }
}

private void setupwordChoise()
{
    wrongGuesses = 0;
    hangImage.Image = hangImages[wrongGuesses];
    int guessIndex = (new Random().Next(words.Length));
    Current = words[guessIndex];
    copyCurrent = "";

    for (int index = 0; index < Current.Length; index++)
    {
        copyCurrent += "_";
    }

    displayCopy();
}

private void displayCopy()
{

    lblShowWord.Text = "";
    for (int index = 0; index < copyCurrent.Length; index++)
    {
        lblShowWord.Text += copyCurrent.Substring(index, 1);
        lblShowWord.Text += " ";
    }
}

private void guessClick(object sender, EventArgs e)
{
    Button choise = sender as Button;
    choise.Enabled = true;
    if (Current.Contains(choise.Text))
    {
        char[] temp = copyCurrent.ToCharArray();

```

```

        char[] find = Current.ToCharArray();
        char guessChar = choise.Text.ElementAt(0);
        for (int index = 0; index < find.Length; index++)
        {
            if (find[index] == guessChar)
            {
                temp[index] = guessChar;
            }
        }
        copyCurrent = new string(temp);
        displayCopy();
    }
    else
    {
        wrongGuesses++;
    }
    if (wrongGuesses < 7)
    {
        hangImage.Image = hangImages[wrongGuesses];
    }
    else
    {
        lblResult.Text = "Ai pierdut!";
    }
    if (copyCurrent.Equals(Current))
    {
        lblResult.Text = "Ai castigat!";
    }
}

private void FormHangman_Load(object sender, EventArgs e)
{
    loadwords();
    setupwordChoise();
}

private void button19_Click(object sender, EventArgs e)
{
    setupwordChoise();
    lblResult.Text = "";
}
}

```

Jocul 3- Battleship



Un alt joc realizat pe calculator ce a pornit de la jocul cu hârtie și creion între 2 jucători este “Bătălia vapoarelor”. După cum observați în forma de mai sus cei doi jucători, calculatorul(Enemy) și player-ul nostru, este divizată în două secțiuni. Fiecare jucător are un număr de poziții în care își ascunde vapoarele și un afișaj de scor, o listă de selecție pentru propunerea poziției vasului inamic (Enter move sau Enemy move). Fiecare poziție are un nume de linie și coloană.

La final, după un număr limitat de alegeri, se afișează rezultatul jocului printr-un mesaj, într-un textBox.

Codul sursă este:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Diagnostics;

namespace BattleShipGame
{
    public partial class Form1 : Form
    {
        List<Button> playerPosition;//creează o listă pentru toate butoanele de poziție ale jucătorului
        List<Button> enemyPosition; //creează o listă pentru toate butoanele de poziție ale inamicului
        Random rand = new Random(); //crează o nouă instanță aleatoare pentru clasa numită rand
        int totalShips = 3;//initializarea numarului de vapoare ale jucatorului
        int totalenemy = 3;//initializarea numarului de vapoare ale inamicului
        int rounds = 10;//totalul de runde pe care le joaca fiecare avand 5
        int playerTotalScore = 0;//initializarea scorului jucatorului
        int enemyTotalScore = 0;//initializarea scorului inamicului

        public Form1()
        {
            InitializeComponent();
            loadButtons();//încărcați butoanele pentru inamic și jucător în sistem
        }
    }
}
```

```

attackButton.Enabled = true;//dezactivați butonul de atac a jucătorului
enemyLocationList.Text = null;//anulează caseta derulantă A locației inamicului
}
/*Mai jos este funcția evenimentului palyerPicsPosition.
Acestă funcție va rula de fiecare dată când jucătorul va face clic pe propriul buton.
În această funcție avem 2 instrucțiuni if.
Primul dacă verifica daca numărul totalShips are o valoare mai mare decât 0
ceea ce înseamnă că are mai mult de 1, atunci permite rularea instrucțiunilor din interior.*/
/*mai întâi va verifica ce buton a fost apasat și îl va stoca în variabila locală numită buton.
atunci vom dezactiva butonul și îl va devei tinta playership
daca am apasat unbuton se va schimba culoarea background-ului in albastru. Va fi usor de identificat
daca am lovit o nava iar numarul de nave va scadea
*/
/* al 2 lea if verifica daca totalul navelor este 0 permite butonului atac sa fie apasat.
si transforma bacgroundul in RED. Textul din label indica ca jucatoru poate selecteze 3 butoane.
Va tb sa selecteze din lista pozitia.
*/
private void palyerPicksPosition(object sender, EventArgs e)
{
    //aceasta fuctie permite jucatorului să atace 3 pozitii de pe mapa
    if (totalShips > 0)
    {
        //daca numarul total de vapoare este >0 inseapna ca am descoperit nave
        var button = (Button)sender;
        //orice buton apasat va tb dezactivat
        button.Enabled = false;
        //daca am nimerit o nava
        button.Tag = "Vaporul jucatorului";
        //si isi va schimba culoarea in albastru
        button.BackColor = System.Drawing.Color.Blue;
        totalShips--;/scade numarul navelor
    }
    else
    if (totalShips ==0)
    {
        //daca jucatorul a lovit toate navele
        attackButton.Enabled= true;
        //se activeaza butonul atac
        attackButton.BackColor = System.Drawing.Color.Red;

        //culoarea de fundal devine rosu
        helpText.Top = 55;
        //se muta textul help sus
        helpText.Left = 230;
        helpText.Text = "2) Alege din lista pozitia de atac";
    }
}
private void attackEnemyPosition(object sender, EventArgs e)
{
    //functia permite jucatorului sa faca mișcări pe locația inamicului,
    //trebuie să verifice dacă jucătorul poate alege o locație din listă
    if(enemyLocationList.Text!="")
    {
        //daca miscarea corespunzatoare este lovita
        //sedeclara variabila attacPos care va primi valoarea text, aleasa din lista
        var attackPos = enemyLocationList.Text;
        attackPos = attackPos.ToLower();
        //se converteste textul in litere mici
        int index = enemyPosition.FindIndex(a => a.Name == attackPos);
    }
}

```

```

//se cauta in pozitiile inamice valoarea alesa si i se atribuie variabilei index
if(enemyPosition[index].Enabled && rounds>0)
{
    rounds--;
    roundText.Text = "Runda " + rounds;

    if(enemyPosition[index].Tag=="vapor inamic")
    {
        //daca am ghicit locatia inamicului
        enemyPosition[index].Enabled = false;
        //se dezactiveaza butonul
        enemyPosition[index].BackgroundImage = Properties.Resources.fireIcon;
        //imaginea se scimba cu cea a exploziei
        enemyPosition[index].BackColor = System.Drawing.Color.DarkBlue;
        //imaginea de background devine albастar inchis
        playerTotalScore++;
        //creste scorul jucatorului
        playerScore.Text = "" + playerTotalScore;
        //se afiseaza valoarea text a scorului jucatorului
        enemyPlayTimer.Start();
        //restart timerul jucatorului
    }
    else
    {
        //daca jucatorul nu a nimerit nava inamica
        enemyPosition[index].Enabled = false;
        //dezactivam butonul
        enemyPosition[index].BackgroundImage = Properties.Resources.missIcon;
        //schimbam imaginea de background
        enemyPosition[index].BackColor = System.Drawing.Color.DarkBlue;
        //culoarea butonului devine albatru inchis
        enemyPlayTimer.Start();           //repornim timer-ul
    }
}
else
{
    //daca jucatorul nu a ales corect locatia din lista se afiseaza mesajul
    MessageBox.Show("Alege o alta locatie din lista.");
}
}

private void enemyAttackPlayer(object sender, EventArgs e)
{
    //aceasta functie este pentru CPU care alege pozitia de pe mapa jucatorului
    if (playerPosition.Count > 0 && rounds > 0)
    {
        rounds--;
        roundText.Text = "Runda " + rounds;
        int index = rand.Next(playerPosition.Count);
        if (playerPosition[index].Tag == "vaporul jucatorului")
        {
            playerPosition[index].BackgroundImage = Properties.Resources.fireIcon;
            enemyMoves.Text = "" + playerPosition[index].Text;

            playerPosition[index].Enabled = false;
            playerPosition[index].BackColor = System.Drawing.Color.DarkBlue;
            playerPosition.RemoveAt(index);
            enemyTotalScore++;
            enemyScore.Text = "" + enemyTotalScore;
        }
    }
}

```

```

        enemyPlayTimer.Stop();
    }
    else
    {
        playerPosition[index].BackgroundImage = Properties.Resources.missIcon;
        enemyMoves.Text = "" + playerPosition[index].Text;
        playerPosition[index].Enabled = false;
        playerPosition[index].BackColor = System.Drawing.Color.DarkBlue;
        playerPosition.RemoveAt(index);
        enemyPlayTimer.Stop();
    }
}
if (rounds < 1 || playerTotalScore > 2 || enemyTotalScore > 2)
{
    if (playerTotalScore > enemyTotalScore)
    {
        MessageBox.Show("Ai castigat");
    }
    if (playerTotalScore == enemyTotalScore)
    {
        MessageBox.Show("Nu a castigat nimeni. Mai incerca");
    }
    if (playerTotalScore < enemyTotalScore)
    {
        MessageBox.Show("Ai pierdut!");
    }
}
private void enemyPicksPositions(object sender, EventArgs e)
{
    int index = rand.Next(enemyPosition.Count);
    //aceasta functie este pentru CPU care alege pozitia de pe mapa jucatorului
    if (enemyPosition[index].Enabled == true && enemyPosition[index].Tag == null)
    {
        enemyPosition[index].Tag = "nava inamica";
        totalenemy--;
        Debug.WriteLine("Pozitia inamicului " + enemyPosition[index].Text);
    }
    else
    {
        index = rand.Next(enemyPosition.Count);
    }
    if(enemyTotalScore>1)
    {
        enemyPositionPicker.Stop();
    }
}
private void loadButtons()
{
    //Aceasta fc va incarca toate butoanele in lista derulanta
playerPosition = new List<Button> { w1, w2, w3, w4, x1, x2, x3, x4, y1, y2, y3, y4, z1, z2, z3, z4 };
enemyPosition = new List<Button> { a1, a2, a3, a4, b1, b2, b3, b4, c1, c2, c3, c4, d1, d2, d3, d4 };

    /*această buclă va parcurge fiecare dintre butoanele poziției inamice,
     apoi le va adăuga pe lista derulantă a locației inamicului pentru noi,
     va elimina și toate etichetele de pe butoanele locației inamicului */
    for(int i=0;i<enemyPosition.Count;i++)
    {
        enemyPosition[i].Tag = null;
        enemyLocationList.Items.Add(enemyPosition[i].Text);
    }
}
}

```

Jocul 4- TRex

Crearea unui joc de rulare fără sfârșit TRex care funcționează în studio vizual folosind C #(WindowsFormApplication C#) a fost pentru mine o încercare.

Am creat simularea saltului dinozaurului TRex, forța săriturii și gravitație.

În cadrul programului am utilizat diferite tipuri de date, cum ar fi numere întregi, booleane și siruri de caractere.

În cadrul animării mai multor obiecte am folosit cronometru (timer) și casete de imagine. De asemenea am utilizat funcții (event) pentru a activa anumite taste și anumite evenimente.

Am creat funcția de resetare a jocului iar un contor va menține scorul pe tot parcursul jocului.

Am calculat coliziunea dintre obiecte și imaginea jucătorului.



Codul sursă este:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace TRexGame
{
    public partial class Form1 : Form
    {
        bool jumping = false;
        int jumpSpeed = 10;
        int force = 12;
        int score = 0;
        int obstacleSpeed = 10;
        Random rnd = new Random();
        public Form1()
        {
            InitializeComponent();
            // resetGame(); // ruleaza functia reset
```

```

        }
    private void gameEvent(object sender, EventArgs e)
    {
        trex.Top += jumpSpeed;
        scoreText.Text = "Score: " + score;
        if (jumping && force < 0)
        {
            jumping = false;
        }
        if (jumping)
        {
            jumpSpeed = -12;
            force -= 1;
        }
        else
        {
            jumpSpeed = 12;
        }
        foreach (Control x in this.Controls)
        {
            if (x is PictureBox && x.Tag == "obstacle")
            {
                x.Left -= obstacleSpeed;
                if (x.Left + x.Width < -120)
                {
                    x.Left = this.ClientSize.Width + rnd.Next(200, 800);
                    score++;
                }
                if (trex.Bounds.IntersectsWith(x.Bounds))
                {
                    gameTimer.Stop();
                    trex.Image = Properties.Resources.dead;
                    scoreText.Text += " Apasa R pentru restart";
                }
            }
            if(trex.Top>=380 && !jumping)
            {
                force = 12;
                trex.Top = floor.Top - trex.Height;
                jumpSpeed = 0;
            }
            if(score>=10)
            {
                obstacleSpeed = 15;
            }
        }
    private void keyisdown(object sender, KeyEventArgs e)
    {
        if(e.KeyCode==Keys.Space && !jumping)
        {
            jumping = true;
        }
    }
}

```

```

        }
    }
    private void keyisup(object sender, KeyEventArgs e)
    {
        if (e.KeyCode == Keys.R)
        {   resetGame(); //ruleaza functia reset   }
        if(jumping)
        {
            jumping = false;
        }
    }
    public void resetGame()
    {
        force = 12;
        trex.Top = floor.Top - trex.Height;
        jumpSpeed = 0;
        jumping=false;
        score = 0;
        obstacleSpeed = 10;
        scoreText.Text = "Scor " + score;
        trex.Image = Properties.Resources.running;
        foreach(Control x in this.Controls)
        {
            if(x is PictureBox && x.Tag=="obstacle")
            {
                int position = rnd.Next(600, 1000);
                x.Left = 640 + (x.Left + position + x.Width * 3);
            }
        }
        gameTimer.Start();
    }
}

```

Propuneri:

Realizați:

1.Ghicitoarea unui numar de la 1 la 100

2. Puzzle (<http://jetgamedev.blogspot.ro/2012/05/lesson-0229-c-lab-4-create-image-puzzle.html>)

3. Joc de zaruri (<http://www.youtube.com/watch?v=vic52OZM6Ho>)

Jocul 5- Flappy Bird Game

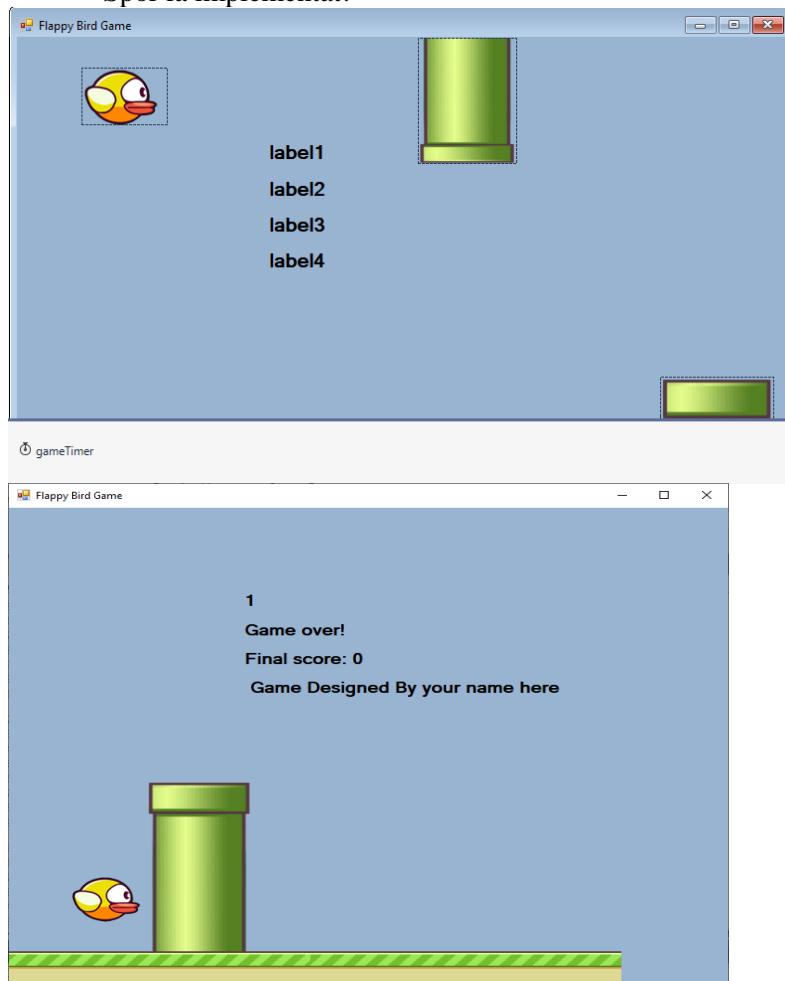
<https://www.mooict.com/create-flappy-bird-game-in-visual-studio-using-c/>

În jocul Flappy Bird a fost creat de către Dong Nguyen. Jocul a fost lansat pe 24 mai 2013, este 2D, cu grafici care amintesc de epoca de aur a jocurilor video. La câteva luni după lansare, Flappy Bird a ajuns în top-ul celor mai populare jocuri gratuite pentru mobil, având peste 50 de milioane de downloadări.

Flappy Bird este un joc cunoscut pentru nivelul de dificultate. Jocul, implică atingerea ecranului/apasarea tastei space/efectuarea click stanga pe ecranul dispozitivului dumneavoastră pentru a propulsa o pasăre mică printr-o serie de ţevi plasate strategic.

Zboară prin conducte fară a te lovi, într-un joc de tip arcade extrem de antrenant, care a născut fenomenul mondial Flappy Bird.

Spor la implementat!



Alicația conține 4 label-uri, 3 imagini ce reprezintă pasărea și tubulatura prin care va trebui ghidată pe suprafața scenei și un timer.

Ghidarea păsării pe suprafața de joc va trebui realizată cu ajutorul tastelor săgeți și a tastei space pentru salt. Aceasta presupune realizarea codului de manipulare al tastelor.

Codul sursă este:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace FlappyBirdGame
{
    public partial class Form1 : Form
    {
        bool jumping = false;
        int pipeSpeed = 5;
        int gravity = 5;
        int Inscore = 0;
        public Form1()
        {
            InitializeComponent();
            endText1.Text = "Game over!";
            endText2.Text = "Final score: " + Inscore;
            gameDesigner.Text = " Game Designed By your name here";
            endText1.Visible = false;
            endText2.Visible = false;
            gameDesigner.Visible = false;
        }

        private void gameTimer_Tick(object sender, EventArgs e)
        {
            pipeBottom.Left -= pipeSpeed;
            pipeTop.Left -= pipeSpeed;
            flappyBird.Top += gravity;
            scoreText.Text = "" + Inscore;

            if (pipeBottom.Left < -80)
            {
                pipeBottom.Left = 1000;
                Inscore += 1;
            }
            else
                if (pipeTop.Left < -95)
            {
                pipeBottom.Left = 1100;
                Inscore += 1;
            }

            if (flappyBird.Bounds.IntersectsWith(ground.Bounds))
            {
                endGame();
            }
        }
    }
}
```

```
        }
        else if
(flapBird.Bounds.IntersectsWith(pipeBottom.Bounds))
{
    endGame();
}
else if (flapBird.Bounds.IntersectsWith(pipeTop.Bounds))
{
    endGame();
}
}

private void GameKeyUp(object sender, KeyEventArgs e)
{
    if(e.KeyCode==Keys.Space)
    {
        jumping = false;
        gravity = -5;
    }
}

private void GameKeyDown(object sender, KeyEventArgs e)
{
    if (e.KeyCode == Keys.Space)
    {
        jumping = true;
        gravity = 5;
    }
}

private void endGame()
{
    gameTimer.Stop();
    endText1.Visible = true;
    endText2.Visible = true;
    gameDesigner.Visible = true;
}

}
```

Propuneri:

Realizați:

1. Convertor texte- mesaj vocal
(<http://www.youtube.com/watch?v=XKtNgVL4jLc>)
 2. Virus- blocarea mouse-ului
(<http://www.youtube.com/watch?v=F3Ny3AE8ZgE>)
 3. Ping pong (<http://www.youtube.com/watch?v=TdXZQNF373M>)
 4. Cartonase (problema „cartonase” → .campion)

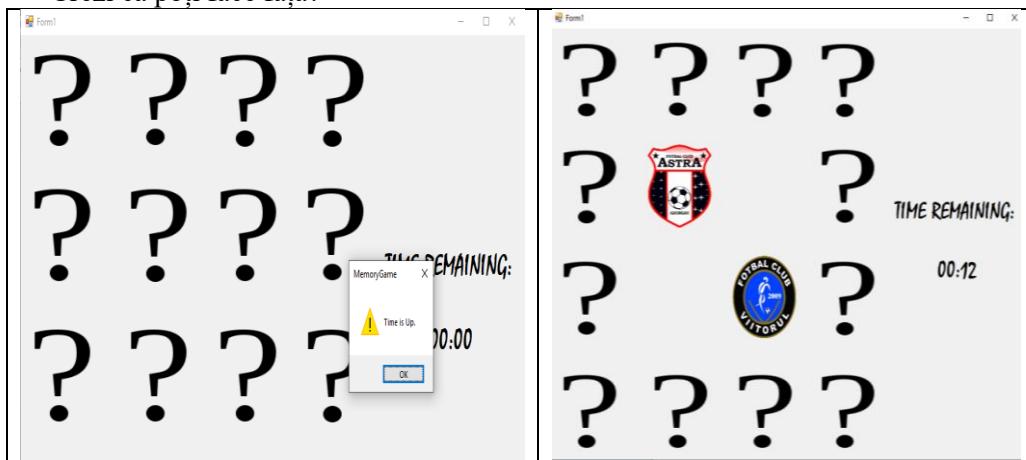
Jocul 6- Memory Game

Jocurile de memorie sunt foarte importante pentru păstrarea în formă a creierului. Ca orice mușchi, dacă nu este „lucrat”, creierul își pierde din capacitate.

Jocurile de memorie, simple la prima vedere, sunt deosebit de îndrăgite de copii întrucât, de multe ori, ei vor prelua conducerea atunci când vine vorba de memorie vizuală.

Jocul Memory Game își propune să testeze abilitatea de reținere a informațiilor, cât și viteza de gândire. 8 echipe de fotbal se ascund sub un vag semn de întrebare, iar tu ești provocat ca în timp de doar 30 de secunde să găsești dublura fiecareia.

Crezi că poti face față?



În această aplicație am folosit 16 PictureBox-uri, pentru cele 8 echipe de fotbal și dublura lor.

Jocul se porneste la pasarea butonului START si EXIT inchide fereastra.

Deși la început apare doar un semn al întrebării, la un simplu click, acesta va afișa stema echipei care se ascunde sub el.

De asemenea, am utilizat 2 label-uri, pentru ca jucatorul să fie informat cât timp (din totalul celor 30 de secunde) mai are la dispoziție pentru a termina jocul.

Dacă la al doilea click, jucătorul nimerește dublura stemei ce se ivește după primul click, atunci locurile în care cele două imagini se aflau va rămâne gol. În caz contrar, se vor afișa emblemele a două steme diferite, apoi se vor ascunde din nou, în același loc, sub semnul întrebării.

Codul sursă este:

```
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Linq;
using System.Windows.Forms;
using WindowsFormsApplication3.Properties;

namespace WindowsFormsApplication3
```

```

{
    public partial class GAME : Form
    {
        private bool _allowClick = true;
        private PictureBox _firstGuess;
        private readonly Random _random = new Random();
        private readonly Timer _clickTimer = new Timer();
        int ticks = 30;
        readonly Timer timer = new Timer { Interval = 1000 };

        public GAME()
        {
            InitializeComponent();
            SetRandomImages();
            HideImages();
            StartGameTimer();
            _clickTimer.Interval = 1000;
            _clickTimer.Tick += _clickTimer_Click;

        }
        private PictureBox[] PictureBoxes
        {
            get
            {
                return Controls.OfType<PictureBox>().ToArray();
            }
        }
        private static IEnumerable<Image> Images
        {
            get{
                return new Image[]
                {
                    Resources.img1,
                    Resources.img2,
                    Resources.img3,
                    Resources.img4,
                    Resources.img5,
                    Resources.img6,
                    Resources.img7,
                    Resources.img8
                };
            }
        }
    }

    private void StartGameTimer()
    {
        timer.Start();
        timer.Tick += delegate
        {
            ticks--;
            if (ticks == -1)
            {
                timer.Stop();
            }
        };
    }
}

```

```

MessageBox.Show("Time is Up.", "MemoryGame", MessageBoxButtons.OK,
MessageBoxIcon.Exclamation);
    ResetImages();
}
var time = TimeSpan.FromSeconds(ticks);
lblTime.Text = "00:" + time.ToString("ss");
};

private void ResetImages()
{
    foreach (var pic in PictureBoxes)
    {
        pic.Tag = null;
        pic.Visible = true;
    }

    HideImages();
    SetRandomImages();
    ticks = 30;
    timer.Start();
}

private void HideImages()
{
    foreach (var pic in PictureBoxes)
    {
        pic.Image = Resources.img0;
        // pic.Image = (Image)pic.Tag;
    }
}

private PictureBox GetFreeSlot()
{
    int num;
    do
    {
        num = _random.Next(0, PictureBoxes.Count());

    } while (PictureBoxes[num].Tag != null);
    return PictureBoxes[num];
}

private void SetRandomImages()
{
    foreach (var image in Images)
    {
        GetFreeSlot().Tag = image;
        GetFreeSlot().Tag = image;
    }
}

private void ClickImage(object sender, EventArgs e)
{
    if (!_allowClick) return;
    var pic = (PictureBox)sender;
}

```


Jocul 7- HANGMAN

Noțiuni : Crearea dinamică a butoanelor și label-urilor, obiecte Graphics, metode de desenare, Menustrip

COD SURSA

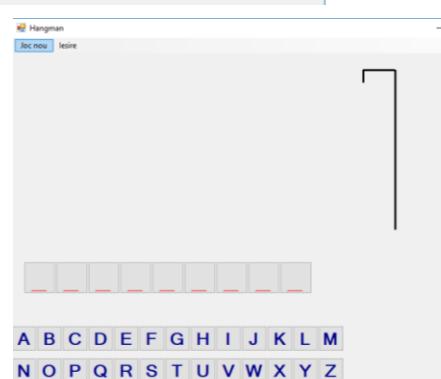
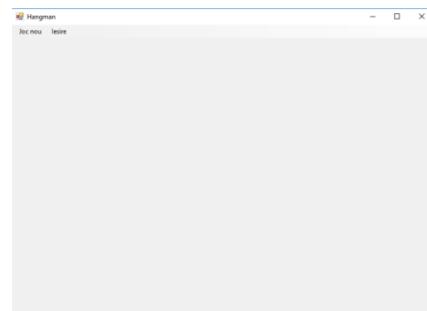
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace hangman
{
    public partial class Form1 : Form
    {
        Graphics g;
        string cuvant_de_ghicit =
"DRAGOBETE";
        int errors = 0; int rest_litere = 9;
        Pen penc = new Pen(Color.Black, 3);
        Button[] litera_cuv = new Button[9];

        public Form1()
        {
            InitializeComponent();
        }

        private void iesireToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            Application.Exit();
        }

        private void creare_litere()
        {
            Button[] litera = new Button[26];
            for(int i=0;i<26;i++)
            {
                litera[i] = new Button();
                litera[i].Text = Convert.ToString((char)('A' + i));
                litera[i].Size = new Size(40, 40);
                litera[i].Font = new Font(this.Font.Name, 20,
FontStyle.Bold);
                litera[i].ForeColor = Color.DarkBlue;
                if (i < 13)
                    litera[i].Location = new Point(i * 40, 450);
                else litera[i].Location = new Point((i-13) * 40, 500);
                litera[i].Click+=new EventHandler(litera_clic);
            }
            this.Controls.AddRange(litera);
        }
    }
}
```

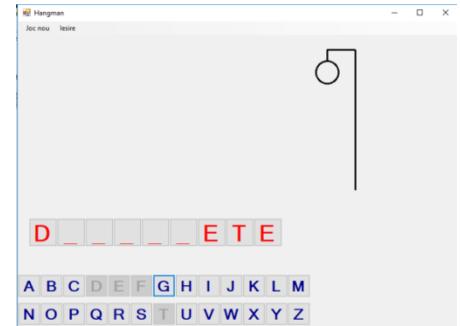


```

        }
    private void creare_litere_cuvant()
    {
        //Button[] litera_cuv = new Button[9];
        for (int i = 0; i < 9; i++)
        {
            litera_cuv[i] = new Button();
            litera_cuv[i].Text = "__";
            litera_cuv[i].Size = new Size(50, 50);
            litera_cuv[i].Font = new Font(this.Font.Name, 30,
FontStyle.Bold);
            litera_cuv[i].ForeColor = Color.Red;
            litera_cuv[i].Location = new Point(20+i *50, 350);
        }
        this.Controls.AddRange(litera_cuv);
    }
    private void jocNouToolStripMenuItem_Click(object sender,
EventArgs e)
{
    creare_litere();
    creare_litere_cuvant();
    g = this.CreateGraphics();
    g.DrawLine(penc, 600, 50, 600, 300);
    g.DrawLine(penc, 600, 50, 550, 50);
    g.DrawLine(penc, 550, 50, 550, 70);

}
private void desen(int errors)
{
    switch (errors)
    {
case 1: g.DrawEllipse(penc, 530, 70, 40, 40); break;//cap
case 2: g.DrawLine(penc, 550, 110, 550, 200); break;//trunchi
case 3: g.DrawLine(penc, 550, 130, 520, 150); break;//brat
case 4: g.DrawLine(penc, 550, 130, 580, 150); break;//brat
case 5: g.DrawLine(penc, 550, 200, 520, 250); break;//picior
case 6: g.DrawLine(penc, 550, 200, 580, 250); break;//picior
    }
}
private void litera_clic(object
sender,EventArgs e)
{
    Button litera = (Button)sender;
    int ok = 0;
    for(int i=0;i<9;i++)
{
if(cuvant_de_ghicit[i]==Convert.ToChar(litera.Text))
{
    ok = 1;
    litera_cuv[i].Text = litera.Text;
    rest_litere--;
}
if (ok == 0)
{
}
}
}

```



```

if(cuvant_de_ghicit[i]==Convert.ToChar(litera.Text))
{
    ok = 1;
    litera_cuv[i].Text = litera.Text;
    rest_litere--;
}
if (ok == 0)
{
}
}
}

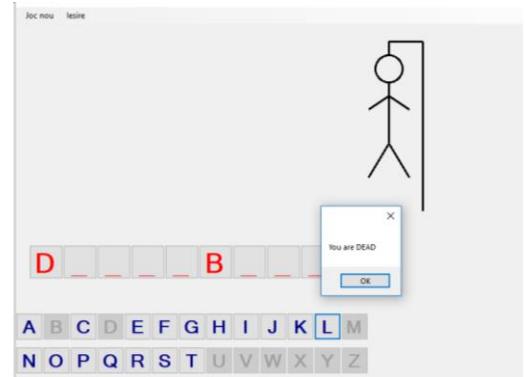
```

```

        errors++;
        if (errors == 8)
    MessageBox.Show("You are DEAD");
        else desen(errors);
    }
    litera.Enabled = false;
    if (rest_litere == 0)
    MessageBox.Show("Ai ghicit!!!");
}

}

```



Jocul 8- Breakout Game

Breakout este un joc de arcadă dezvoltat și publicat de Atari și lansat pe 13 mai 1976 și l-am realizat în limbajul C#.

Breakout Game constă într-un zid de cărămizi dispuse pe mai multe linii și divers colorate, aflate în partea de sus a ecranului cu scopul de a-l distrugă. O mingă se mișcă de sus în josul ecranului, ghidată de tastele săgeți și tasta space. Atunci când o cărămidă este lovită, mingea sare înapoi și cărămidă dispare. Jucătorul pierde o încercare atunci când mingea atinge partea de jos a ecranului. Pentru a preveni acest lucru, jucătorul are o paletă mobilă orizontală ghidată de tastele săgeți.

Manevrarea tastelor, efectul dispariției cărămizilor și să nu uităm afișarea scorului se realizează prin cod.



Codul sursă este:

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace BreakoutGame
{
    public partial class Form1 : Form
    {

        bool goRight;
        bool goLeft;
        int speed=10;
        int ballx = 5;
        int bally = 5;
        int score = 0;
        private Random rnd = new Random();
        public Form1()
        {
            InitializeComponent();
            foreach (Control x in this.Controls)
            {
                if(x is PictureBox && x.Tag == "block")
                {
                    Color randomColor = Color.FromArgb(rnd.Next(256), rnd.Next(256),
rnd.Next(256));
                    x.BackColor = randomColor;
                }
            }
        }

        private void keyisdown(object sender, KeyEventArgs e)
        {
            if(e.KeyCode==Keys.Left && player.Left>0)
            {
                goLeft = true;
            }
            if(e.KeyCode==Keys.Right && player.Left+player.Width<920)
            {
                goRight = true;
            }
        }

        private void keyisup(object sender, KeyEventArgs e)
        {
            if(e.KeyCode==Keys.Left)
            {
                goLeft = false;
            }
            if (e.KeyCode == Keys.Right)

```

```

        {
            goRight = false;
        }
    }

private void timer1_Tick(object sender, EventArgs e)
{
    ball.Left += ballx;
    ball.Top += bally;

    label1.Text = "Score:" + score;

    if (goLeft) { player.Left -= speed; }
    if (goRight) { player.Left += speed; }

    if (player.Left<1)
    {
        goLeft = false;
    }
    else if(player.Left+player.Width>920)
    {
        goRight = false;
    }
    if(ball.Left+ball.Width>ClientSize.Width || ball.Left<0)
    {
        ballx = -ballx;
    }
    if(ball.Top<0||ball.Bounds.IntersectsWith(player.Bounds))
    {
        bally = -bally;
    }
    if(ball.Top+ball.Height>ClientSize.Height)
    {
        gameOver();
    }
    foreach (Control x in this.Controls)
    {
        if (x is PictureBox && x.Tag == "block")
        {
            if(ball.Bounds.IntersectsWith(x.Bounds))
            {
                this.Controls.Remove(x);
                bally = -bally;
                score++;
            }
        }
    }
    if(score>34)
    {
        gameOver();
        MessageBox.Show("You Win");
    }
}
private void gameOver(){timer1.Stop();      }    }}}

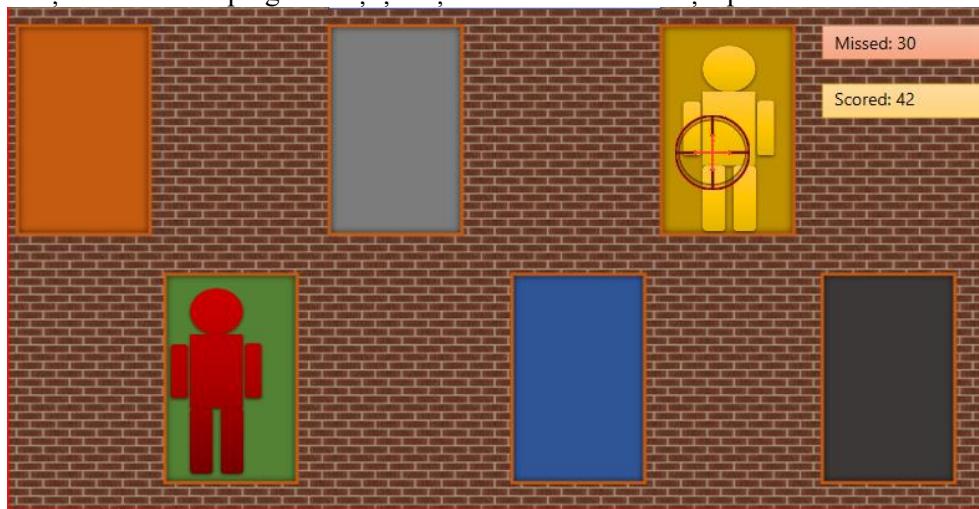
```

Jocul 9- Snaper Game

Snaper Game face parte din categoria jocurilor care reușește să capteze foarte ușor atenția jucătorilor, cu scopul de a obține scorul cel mai bun la țintă.

Îndemanarea, precizia, atenția cât și spiritul competitiv sunt foarte bine antrenate prin intermediul acestui joc.

Ești suficient de pregătit să-ți țintești adversarii? Distracție plăcută!



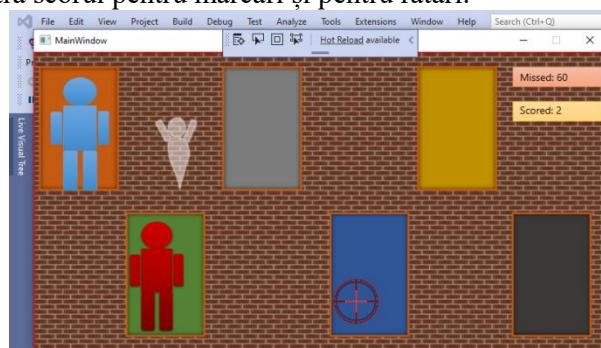
Pentru acest joc am avut nevoie de imagini cu: 4 manechine, 1 fantomă, 1 imagine de fundal și 1 imagine cu țintă.



După cum veți vedea imaginea urmează mouse-ul, putem face click pe imaginile manechin.

Imaginile manechinului schimbă pozițiile și se mișcă la întâmplare, făcând jocul ușor provocator.

Când faceți clic pe manechin, fantoma ei va începe să plutească încet și vom înregistra scorul pentru marcări și pentru ratări.



Codul sursă este:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using System.Windows.Threading;

namespace SnipeTheDummies
{

    public partial class MainWindow : Window
    {
        //declararea variabilelor
        ImageBrush backgroundImage = new ImageBrush();
        // Background image holder - imaginie de fundal
        ImageBrush ghostSprite = new ImageBrush();
        // ghost image holder - Imaginea fantomă
        ImageBrush aimImage = new ImageBrush();
        // imaginea cu ținta

        DispatcherTimer DummyMoveTimer = new DispatcherTimer();
        // Cronometrul miscarii imaginilor cu manechine
        DispatcherTimer showGhostTimer = new DispatcherTimer();
        // Cronometru pentru imaginile cu fantome

        int topCount = 0; // initializarea cronometrul de sus
        int bottomCount = 0; // initializarea -cronometrul de jos

        int score; // înregistrază scorul
        int miss; // înregistrează ratările

        List<int> topLocations;
        // lista locațiilor superioare
        List<int> bottomLocations; // lista locațiilor inferioare

        List<Rectangle> removeThis = new List<Rectangle>();
        // colector de resturi pentru acest joc

        Random rand = new Random(); // generează numere random
        public MainWindow()
        {
            InitializeComponent();
            this.Cursor = Cursors.None; // ascunde cursorul mouse-ului
```

```

// setează imaginea de fundal alocând parametrul imagebrush suprafetei
// de joc
// background - fundalul este preluat din directorul imagini
backgroundImage.ImageSource = new BitmapImage(new
Uri("pack://application:,,,/images/background.png"));
    MyCanvas.Background = backgroundImage;

// setează imaginea cu scorul pentru jucator din directorul imagini
scopeImage.Source = new BitmapImage(new
Uri("pack://application:,,,/images/sniper-aim.png"));

// - setează imaginea cu fantoma din directorul imagini
ghostSprite.ImageSource = new BitmapImage(new
Uri("pack://application:,,,/images/ghost.png"));
// setează cronometrul pentru mișcarea manechinelor
DummyMoveTimer.Tick += DummyMoveTick;
    DummyMoveTimer.Interval =
TimeSpan.FromMilliseconds(rand.Next(800, 2000));
    DummyMoveTimer.Start();

// setează cronometrul pentru animația cu fantoma
showGhostTimer.Tick += ghostAnimation;
showGhostTimer.Interval = TimeSpan.FromMilliseconds(20);
showGhostTimer.Start();

// adaugă locația superioară într-o lista (numerele sunt in pixeli)
topLocations = new List<int> { 23, 270, 540, 23, 270, 540
};

// adaugă locația inferioara într-o lista (numerele sunt in pixeli)
bottomLocations = new List<int> { 138, 128, 678, 138, 128,
678 };

}

private void ShootDummy(object sender, MouseButtonEventArgs e)
{
    if (e.OriginalSource is Rectangle)
    {
// dacă suprafața pe care dăm click este un dreptunghi, atunci acesta
// va trebui creat și va trebui să îl alocăm dreptunghiului care trimite
// acea acțiune
        Rectangle activeRec = (Rectangle)e.OriginalSource; // create the link between the sender rectangle

        MyCanvas.Children.Remove(activeRec);
// caută dreptunghiul și îl elimină de pe suprafața de joc

        score++; // add 1 to the score

        if ((string)activeRec.Tag == "top")
        {
            // if the rectangles tag was top - dacă
dreptunghiul activat a fost în partea superioară

```

```

        // scade o unitate din cronometrul superior
        topCount--;
    }
    else if ((string)activeRec.Tag == "bottom")
    {
// - dacă dreptunghiul activat a fost în partea inferioară

    // scade o unitate din cronometrul inferior
            bottomCount--;
    }

// creaza un nou dreptunghi cu fantoma
//width 60 pixels and height 100 pixels - lățime 60
pixeli și înălțime 100 pixeli
// va avea un tag numit fantomă și va fi umplut cu imaginea fantomei
    Rectangle ghostRec = new Rectangle
    {
        Width = 60,
        Height = 100,
        Fill = ghostSprite,
        Tag = "ghost"
    };

// o data ce dreptunghiul este configurat, va trebui să îi setam nouui
//obiect pozițiile x si y
// vom calcula locația click-ului de mouse și o vom adauga acolo
    Canvas.SetLeft(ghostRec, Mouse.GetPosition(MyCanvas).X - 40);
// seteaza poziția din stânga a dreptunghiului valoarea X a mouse-ului
    Canvas.SetTop(ghostRec, Mouse.GetPosition(MyCanvas).Y - 60);
// seteaza poziția superioara a dreptunghiului valoarea Y a mouse-ului

MyCanvas.Children.Add(ghostRec);
// adaugă noul dreptunghi pe suprafața de joc
    }
}

private void Canvas_MouseMove(object sender, MouseEventArgs e)
{
    // preia coordonatele x si y ale cursorului de la mouse
    System.Windows.Point position = e.GetPosition(this);
    double pX = position.X;
    double pY = position.Y;

    // seteaza înălțimea și lățimea cercului cursorului de la mouse
    Canvas.SetLeft(scopeImage, pX - (scopeImage.Width / 2));
    Canvas.SetTop(scopeImage, pY - (scopeImage.Height / 2));
}

private void ghostAnimation(object sender, EventArgs e)
{
    scoreText.Content = "Scored: " + score;
// alocă parametrul cu scor-ul textbox-ului în care acesta va fi afișat
    missText.Content = "Missed: " + miss;
// alocă parametrul cu ratările textbox-ului în care acesta va fi
//afișat
}

```

```

// execută un loop foreach pentru a verifica dacă există dreptunghiuri
// pe suprafața de joc
    foreach (var x in MyCanvas.Children.OfType<Rectangle>())
    {
        // dacă există un dreptunghi cu fantoma atașată
        if ((string)x.Tag == "ghost")
        {
            // animează-l pană în partea superioară a ecranului
            Canvas.SetTop(x, Canvas.GetTop(x) - 5);

            // dacă ajunge o fantoma la -180 pixeli de sus
            if (Canvas.GetTop(x) < -180)
            {
                // adaugă acea fantomă la colectorul de resturi
                removeThis.Add(x);
            }
        }
    }

    // o va sterge de pe ecran
    // Verifică numarul rămas de dreptunghiuri în lista de elemente active
    // sterge-le din listă folosind un foreach
    foreach (Rectangle y in removeThis)
    {
        MyCanvas.Children.Remove(y);
    }
}

private void DummyMoveTick(object sender, EventArgs e)
{
    removeThis.Clear(); // sterge colectorul de resturi când se încarcă

    // acest loop for each va verifica dacă avem vreun
    // dreptunghi pe suprafața de joc. Dacă avem, trebuie să le eliminăm

    foreach (var i in MyCanvas.Children.OfType<Rectangle>())
    {
        // verifică dacă există dreptunghiuri care au tag-ul superior sau
        // inferior atașat la ele
        if ((string)i.Tag == "top" || (string)i.Tag == "bottom")
        {
            removeThis.Add(i); // adaugă elementele în lista removeThis

            topCount--; // scade o unitate din parametrul topCount
            bottomCount--; // scade o unitate din parametrul bottomCount
            miss++; // adaugă-le la parametrul miss
        }
    }
    // dacă parametrul topcount este mai mic de 3
    if (topCount < 3)
    {
        // executa funcția ShowDummies
    }
}

```

```

        ShowDummies(topLocations[rand.Next(0, 5)], 35, rand.Next(1, 4),
        "top");
                topCount++; // contorizeaza scorul
            }

// dacă parametrul bottomcount este mai mic de 3
if (bottomCount < 3)
{
    // executa functia ShowDummies
ShowDummies(bottomLocations[rand.Next(0, 5)], 230, rand.Next(1, 4),
"bottom");
}
}

private void ShowDummies(int x, int y, int skin, string tag)
{
// crează o nouă imagine pentru fundalul imaginii manechinelor
ImageBrush dummyBackground = new ImageBrush();
// oricare dintre manechine va fi returnat de această funcție
// ii va schimba fundalul iar fiecare număr va fi monitorizat și va fi
//atribuit un caz pentru a răspunde
switch (skin)
{
    case 1:
        dummyBackground.ImageSource = new BitmapImage(new
Uri("pack://application:,,,/images/dummy01.png"));
        break;
    case 2:
        dummyBackground.ImageSource = new BitmapImage(new
Uri("pack://application:,,,/images/dummy02.png"));
        break;
    case 3:
        dummyBackground.ImageSource = new BitmapImage(new
Uri("pack://application:,,,/images/dummy03.png"));
        break;
    case 4:
        dummyBackground.ImageSource = new BitmapImage(new
Uri("pack://application:,,,/images/dummy04.png"));
        break;
}
// - crează un nou dreptunghi
// acest dreptunghi va avea eticheta în aceasta funcție
// lățime 80 pixeli și înălțime 155 pixeli
// culoarea de umplere a acestui dreptunghi va fi culoarea manechinului
Rectangle newRec = new Rectangle
{
    Tag = tag,
    Width = 80,
    Height = 155,
    Fill = dummyBackground
};
Canvas.SetTop(newRec, y); // pozitionează valoarea y a dreptunghiului
Canvas.SetLeft(newRec, x); // position the rectangles x position

MyCanvas.Children.Add(newRec);
// adaugă noul dreptunghi pe suprafața de joc } } }

```

Jocul 10- Ferma animalelor



Proiectul Ferma Animalelor și-a propus folosirea imaginilor, textelor și a sunetelor în descrierea animalelor, folsind ApplicationWindowsForms din limbajul C#.

Cu ajutorul cursorului de mouse selectăm un animăluț din fermă iar sunetul specific va fi auzit și se va afișa un text descriptiv.

Proiectul poate fi folosit ca material didactic al lecțiilor preșcolarilor sau a celor din clasele primare și adaptat după caz, de exemplu, la orele de biologie.

Aplicația “Ferma Animalelor”, este alcătuită din imagini mici, ale celor 8 animaluțe, în cazul nostru, decupate din imaginea mare și suprapuse peste imaginea selectată.

La trecerea cursorului peste animăluț, de exemplu peste CAL, se aude sunetul specific și numele animalului selectat.



Codul sursă este:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Media;
```

```

namespace Ferma
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void pictureBox2_MouseHover(object sender, EventArgs e)
        {
            ToolTip cal = new ToolTip();
            cal.SetToolTip(pictureBox2, "Cal");

            SoundPlayer player = new SoundPlayer();
            player.Stream = Properties.Resources.Horse;
            player.Play();
        }

        private void pictureBox3_MouseHover(object sender, EventArgs e)
        {
            ToolTip vaca = new ToolTip();
            vaca.SetToolTip(pictureBox3, "Vaca");

            SoundPlayer player = new SoundPlayer();
            player.Stream = Properties.Resources.Cow;
            player.Play();
        }

        private void pictureBox4_MouseHover(object sender, EventArgs e)
        {
            ToolTip gaina = new ToolTip();
            gaina.SetToolTip(pictureBox4, "GAINA");
            SoundPlayer player = new SoundPlayer();
            player.Stream = Properties.Resources.gaina1;
            player.Play();
        }
    }
}

```

Propuneri:

Completați aplicația pentru celelalte elemente.

Jocul 11- Space Battle Game

Space Battle Shooter este un joc ce include un câmp de luptă în care o navă spațială se poate mișca doar lateral și trebuie să împuște alte nave spațiale inamice care vin din sens opus cu diferite viteze.

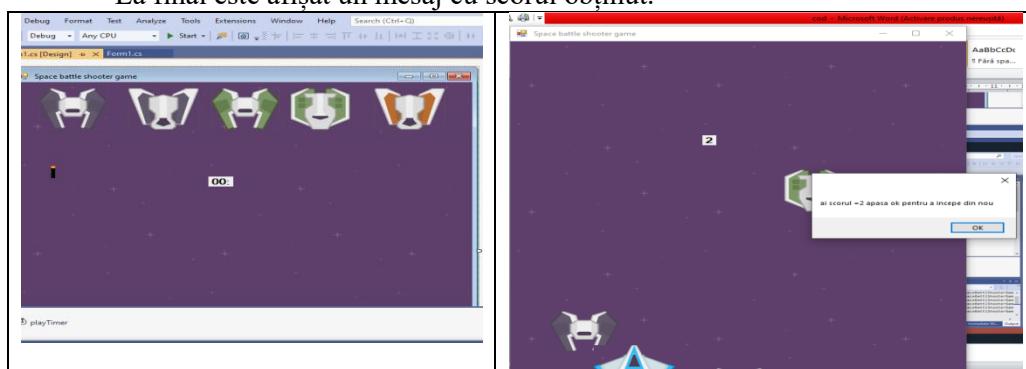
Acesta constă în faptul că jucătorul se poate mișca de la stânga la dreapta și invers și poate arunca gloanțe folosind tasta SPACE.

Jucătorul poate împușca cu un singur gloanț pe rând (cât timp gloanțul se află pe ecran nu mai poate împușca încă o dată), gloanțul are direcția în sus, iar fiecare inamic se deplasează în jos, dacă gloanțul nimerește inamicul, el va reaparea în partea de sus a scenei și va cobora.

Dacă inamicul ajunge la finalul platformei, va reaparea în partea de sus a scenei și va cobora din nou în jos. De fiecare dată când gloanțul nimerește inamicul se va adăuga un punct la scor, iar scorul va fi reînnoit pe ecran.

Vor fi folosite imagini pentru nava jucătorului, cele 5 nave spațiale inamice și imaginea gloanțului. Vom ataşa jocului nelipsitul timer.

La final este afișat un mesaj cu scorul obținut.



Codul sursă

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace spaceBattlShooterGame
{
    public partial class Form1 : Form
    {
        int moveLeft = 0;
        int enemyMove = 5;
        Random rnd = new Random();
        int bulletSpeed = 5;
        bool shooting = false;
```

```

int score = 0;
public Form1()
{
    InitializeComponent();
    enemy1.Top = -500;
    enemy2.Top = -900;
    enemy3.Top = -1300;
    enemy4.Top = -1700;
    enemy5.Top = -2100;

    bullet.Top = -100;
    bullet.Left = -100;

}

private void keisdown(object sender, KeyEventArgs e)
{
    if (e.KeyCode == Keys.Left)
        if (player.Location.X < 0)
        {
            moveLeft = 0;
        }
        else
        {
            moveLeft = -5;
        }

    else if (e.KeyCode == Keys.Right)

        if (player.Location.X > 512)
        {
            moveLeft = 0;
        }

        else
        {
            moveLeft = 5;
        }

    else if (e.KeyCode == Keys.Space)
    {
        if (shooting == false)
        {
            bulletSpeed = 8;
            bullet.Left = player.Left + 50;
            bullet.Top = player.Top;
            shooting = true;
        }
    }
}

private void keyisup(object sender, KeyEventArgs e)

```

```

    {
        if (e.KeyCode == Keys.Left)
        {
            moveLeft = 0;
        }
        else
        {
            if (e.KeyCode == Keys.Right)
            {
                moveLeft = 0;
            }
        }
    }

    private void playTimer_Tick(object sender, EventArgs e)
    {
        player.Left += moveLeft;
        bullet.Top -= bulletSpeed;
        enemy1.Top += enemyMove;
        enemy2.Top += enemyMove;
        enemy3.Top += enemyMove;
        enemy4.Top += enemyMove;
        enemy5.Top += enemyMove;
        scoreText.Text = "" + score;
        if (enemy1.Top == 660 || enemy2.Top == 660 || enemy3.Top == 660 ||
            enemy4.Top == 660 || enemy5.Top == 660)
        {
            gameOver();
        }
        if (shooting && bullet.Top < 0)
        {
            shooting = false;
            bulletSpeed = 0;
            bullet.Top = -100;
            bullet.Left = -100;
        }
        enemyHit();
    }
    private void enemyHit()
    {
        if (bullet.Bounds.IntersectsWith(enemy1.Bounds))
        {
            score += 1;
            enemy1.Top = -500;
            int ranP = rnd.Next(1, 300);
            enemy1.Left = ranP;
            shooting = false;
            bulletSpeed = 0;
            bullet.Top = -100;
            bullet.Left = -100;
        }
        Else
        {
            if (bullet.Bounds.IntersectsWith(enemy2.Bounds))
            {
                score += 1;
                enemy2.Top = -900;
                int ranP = rnd.Next(1, 400);
                enemy2.Left = ranP;
                shooting = false;
                bulletSpeed = 0;
                bullet.Top = -100;
                bullet.Left = -100;
            }
        }
        else
        {
            if (bullet.Bounds.IntersectsWith(enemy3.Bounds))
        }
    }
}

```

```

    {
        score += 1;
        enemy3.Top = -1300;
        int ranP = rnd.Next(1, 500);
        enemy3.Left = ranP;
        shooting = false;
        bulletSpeed = 0;
        bullet.Top = -100;
        bullet.Left = -100;
    }
    else
    if (bullet.Bounds.IntersectsWith(enemy4.Bounds))
    {
        score += 1;
        enemy4.Top = -1700;
        int ranP = rnd.Next(1, 600);
        enemy4.Left = ranP;
        shooting = false;
        bulletSpeed = 0;
        bullet.Top = -100;
        bullet.Left = -100;
    }
    else
    if (bullet.Bounds.IntersectsWith(enemy5.Bounds))
    {
        score += 1;
        enemy5.Top = -2100;
        int ranP = rnd.Next(1, 700);
        shooting = false;
        enemy5.Left = ranP;
        bulletSpeed = 0;
        bullet.Top = -100;
        bullet.Left = -100;
    }
}
private void gameOver()
{
    playTimer.Enabled = false;
    MessageBox.Show("ai scorul =" + score + " apasa ok
pentru a incepe din nou");
    score = 0;
    scoreText.Text = "0";
    enemy1.Top = -500;
    enemy2.Top = -900;
    enemy3.Top = -1300;
    enemy4.Top = -1700;
    enemy5.Top = -2100;

    bullet.Top = -100;
    bullet.Left = -100;
    playTimer.Enabled = true;      }    }}}
```

Jocul 12- Save the Eggs Game



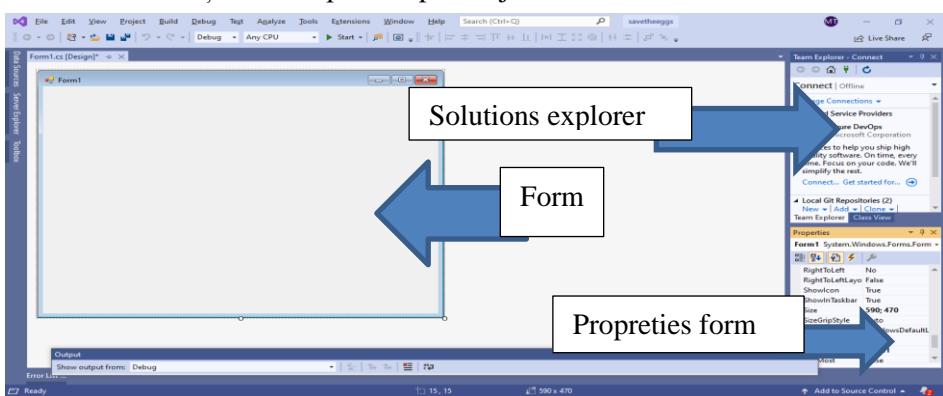
Save The Eggs Game este un joc dezvoltat de compania Namco. A fost lansat în 1980 și a devenit imediat foarte popular.

În **Save The Eggs Game**, jucătorul trebuie să salveze cât mai multe ouă. Un contor reține numărul de ouă salvate, iar un alt contor reține numărul de ouă sparte. Ideea acestui joc este ca jucătorul să prindă obiectele care apar din în partea de sus a ecranului. Jucătorul va putea să se deplaceze la stânga și la dreapta, iar jucătorul va putea, de asemenea, să prindă elementele (ouăle) care cad sau se sparg dacă acestea ajung în partea dejos a ecranului sub formă de picături(ouă sparte).

De asemenea, după un număr de obiecte salvate, viteza apariției obiectelor crește, iar picăturile vor devini mai dese, ceea ce va face jocul mai dificil. Am creat acest joc în Visual Studio C# 2019.

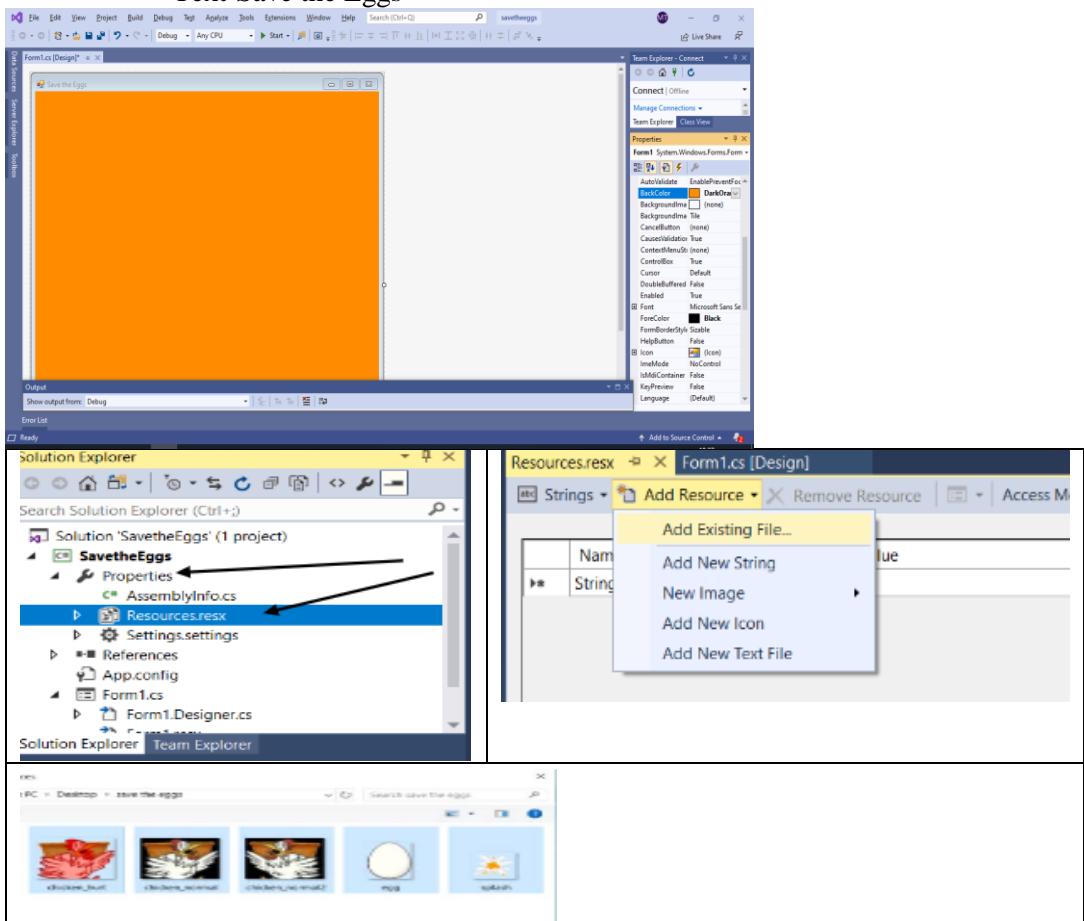
Obiectivele aplicației sunt:

- Utilizarea componentelor Windows Form Application Visual Studio C#, cum ar fi casetele de imagine, etichetele și cronometrele
- Utilizarea clasei aleatorii pentru a genera numere aleatorii și a le aloca locațiilor obiectului
- Utilizarea mai multor evenimente în Visual Studio
- Accelerarea jocului odată ce o condiție a fost îndeplinită
- Păstrați scorul pentru capturi și ratări
- Animați obiectul odată ce elementul este ratat
- Crearea funcțiilor de repornire pentru joc



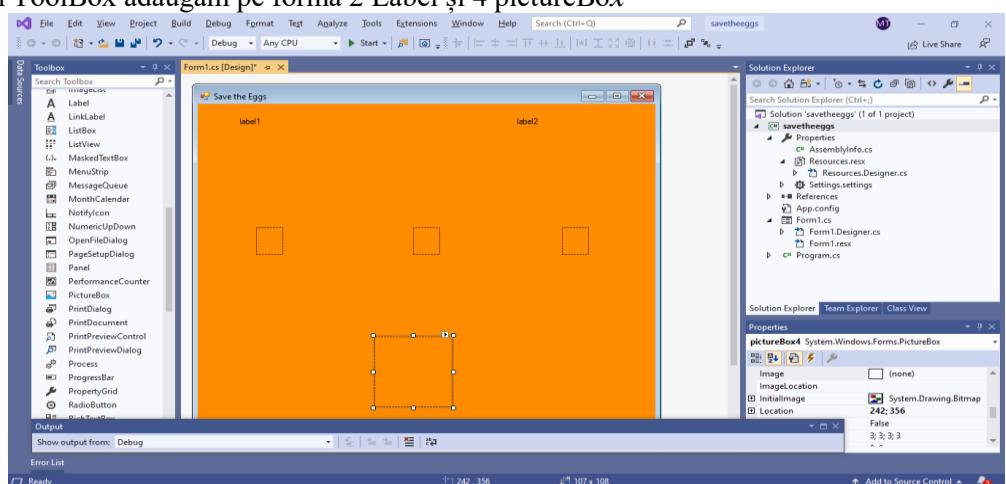
Proprietățile formei:

- Back Color -255, 192,128 – (Dark Orange)
- Size-628,741
- Text-Save the Eggs



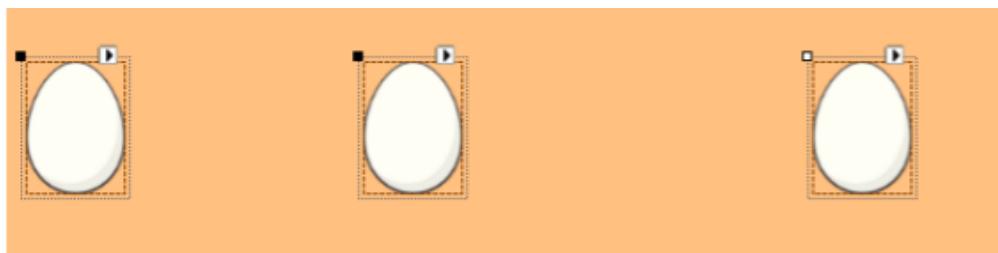
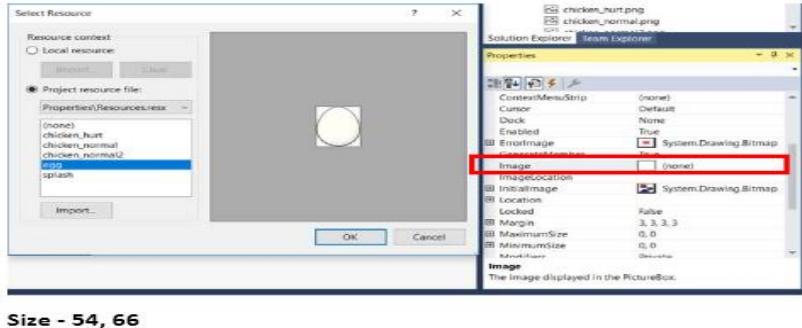
Importăm în resursele proiectului imaginile de mai sus.

Din ToolBox adăugăm pe formă 2 Label și 4 pictureBox



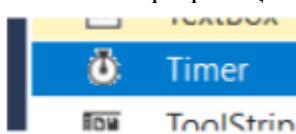
-Selectăm cele 3 picturebox și aplicăm proprietățile:

- Size Mode-Stretch Image
- Tag-Eggs
- Image Egg



Proprietățile celor 3 pictureBoxuri:

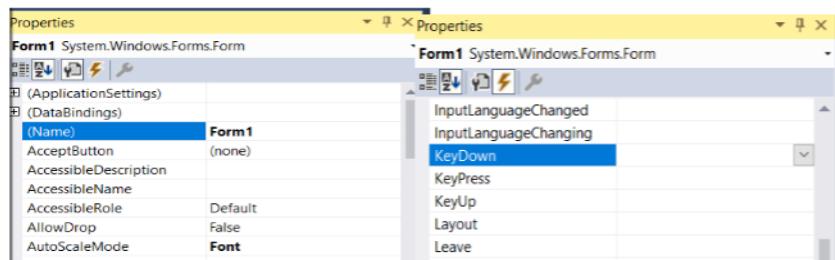
- Name -chicken
 - Image-Chicken_normal2
 - Size -93,83
 - SizeMode-Stretch Image
- Adăugăm un Timer formei din Toolbox cu următoarele proprietăți:



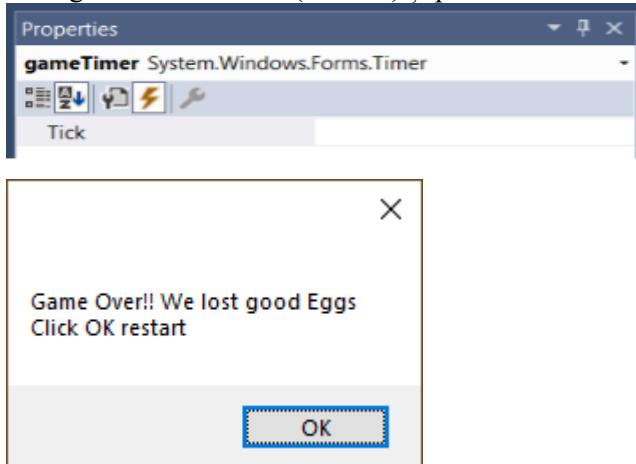
Properties	
gameTimer	System.Windows.Forms.Timer
Enabled	True
GenerateMember	True
Interval	20
Modifiers	Private
Tag	

-Schimbăm numele timerului în gameTimer și intervalul activat - 20 milisecunde

-Adăugăm tastele functionale: săgeată stânga și dreapta
Selectăm Forma apoi aplicăm următoarele proprietăți keyDown și KeyUp



- Se va crea handler pentru cele 2 key: **keyisdown** si **keyisup**
- Adăugăm un eveniment (hendler) și pentru timer tick- gameTick



Pornind de la declararea variabilelor și apoi adăugarea codurilor fiecărui eveniment construim codul sursă al proiectului.

Codul Sursă

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
namespace SaveTheEggs
{
    public partial class Form1 : Form
    {
        bool goleft;
//variabila care verifica daca jucatorul poate muta la stanga
        bool goright;
//variabila care verifica daca jucatorul poate muta la dreapta
        int speed = 5;//valoarea initiala vitezei cu care cad ouale
        int score = 0;//valoarea initiala a scorului
        int missed = 0;//valoarea initiala pentru ouale pierdute
        Random rndy = new Random(); //valoarea oarecare pentru locatia Y
        Random rndx = new Random(); //valoarea oarecare pentru locatia X
    }
}

```

```

    PictureBox splash = new PictureBox();
//creaza dinamic noi stropi

    public Form1()
    {
        InitializeComponent();
        reset(); //apeleaza resetarea jocului
    }
//functia reset() repositioneaza elementele pe suprafata jocului
    private void keyisdown(object sender, KeyEventArgs e)
    {
        if (e.KeyCode == Keys.Left)
        {
//daca se apasa tasta sageata stanga jucatorul se deplaseaza spre
//stanga
            goleft = true;

        }
        if (e.KeyCode == Keys.Right)
        {
//daca se apasa tasta dreapta stanga jucatorul se deplaseaza spre
//dreapta
            goright = true;
        }
    }

    private void keyisup(object sender, KeyEventArgs e)
    {
//in acest caz se face inactiva tasta stanga sau dreapta dupa caz
        if (e.KeyCode == Keys.Left)
        {
//daca se apasa tasta sageata stanga jucatorul se deplaseaza spre
//stanga
            goleft = false;
        }
        if (e.KeyCode == Keys.Right)
        {
//daca se apasa tasta dreapta stanga jucatorul se
//deplaseaza spre dreapta

            goright = false;
        }
    }

    private void gameTick(object sender, EventArgs e)
    {
        label1.Text = "Eggs Caught: " + score;
//se afiseaza scorul oualelor salvate
        label2.Text = "Eggs Missed: " + missed;
//se afiseaza scorul oualelor pierdute
        if (goleft == true && Chichen.Left > 0)
        {
// Daca se acceseaza sageata stanga se deplaseaza gaina cu 12 pixeli
            Chichen.Left -= 12;
//se schimba imaginea gainii cu fata catre stanga
        }
    }
}

```

```

        Chichen.Image = Properties.Resources.chicken_normal2;
    }
    if (goright == true && Chichen.Left + Chichen.Width <
this.ClientSize.Width)
    {
// Daca se acceseaza sageata dreapta se deplaseaza
//gaina catre dreapta cu 12 pixeli
        Chichen.Left += 12;
        //se schimba maginea gainii cu fata catre stanga
        Chichen.Image = Properties.Resources.chicken_normal;
    }
    //verificam daca gana nu a depasit suprafata ferestrei
    foreach (Control X in this.Controls)
    {
        if (X is PictureBox && X.Tag == "Eggs")
        {
//daca se coboara un ou si a fost prins creste viteza
            X.Top += speed;
            //daca oul ajunge pe podea si nu a fost prins apare
//imaginea stropilor. Oul s-a spart
            if (X.Top + X.Height > this.ClientSize.Height)
            {
                splash.Image = Properties.Resources.splash;
                //setam locatia oualui care se va sparge
                splash.Location = X.Location;
                //reinem pozitia oualui
                splash.Height = 59;
                //setam culoarea transparenta a imaginii oualui pierdut
                splash.BackColor = System.Drawing.Color.Transparent;
                //adaugam imaginea picaturii pe forma
                this.Controls.Add(splash);
                //pozitionam imaginea picaturii
                X.Top = rndy.Next(80, 300);
                X.Left = rndx.Next(5, this.ClientSize.Width - X.Width);
                //creste contorul oualelor sparte
                missed++;
                // imaginea vizibila
                Chichen.Image = Properties.Resources.chicken_hurt;
            }

            //daca oul a fost atins de gaina
            //ambele imagini se intersecțează
            if (X.Bounds.IntersectsWith(Chichen.Bounds))
            {
                //repozitionam oul aleator
                X.Top = rndy.Next(100, 300) * -1;
                X.Left = rndx.Next(5, this.ClientSize.Width - X.Width);
                //creste scorul oualelor salvate
                score++;

            }
            if (score >= 20)
        //daca scorul depaseste 20 puncte creste viteza de coborare a
        //oualelor
            speed = 10;
    }
}

```

```

        }
        if (missed > 8)
        {
            gameTimer.Stop();
            //se afiseaza mesajul
            MessageBox.Show("Game Over!! We lost good Eggs" + "\r\n" + "Click OK
restart");
            //daca se apasa ok se restarteaza jocul
            reset();
        }
    }
}

private void reset()
{
    //verificam toate elementele jocului
    foreach (Control X in this.Controls)
    {
        if (X is PictureBox && X.Tag == "Eggs")
        {
            X.Top = rndy.Next(100, 300) * -1;
            X.Left = rndx.Next(5, this.ClientSize.Width - X.Width);

        }
    }
    Chichen.Left = this.ClientSize.Width / 2;
    Chichen.Image = Properties.Resources.chicken_normal2;
    score = 0;
    missed = 0;
    speed = 5;
    goleft = false;
    goright = false;
    gameTimer.Start();
}
}

```

Jocul 13- Tetris Game

Tetris este un joc video de tip puzzle, programat inițial de inginerul Rusiei Sovietice Alexey Pajitnov. Prima versiune care a putut fi folosită a apărut la data de 6 Iunie 1984.

Scopul jocului este de a uni piesele și de a forma linii pe orizontală. Acest joc a apărut inițial pe calculatoarele de tip arcade în SUA, dar și pe consolele de tip „Game Boy”.

De-a lungul timpului au apărut numeroase versiuni, jocul fiind adaptat pentru fiecare tip de consolă video sau sistem de operare, precum: Microsoft, Android, X-Box etc.

Descrierea funcțiilor utilizate în implementare:

1. **private bool gameActive** este un boolean folosit pentru a detecta dacă jocul este activ sau nu.
2. **private bool nextShapeDrawed** este un boolean folosit pentru a observa dacă următoare piesă de Tetromino a apărut pe ecran.
3. **private bool bottomCollided, leftCollided si rightCollided** sunt 3 booleeni care indică dacă piesele de tetris s-au unit între ele sau nu.
4. **private bool isGameOver** observă dacă jocul s-a încheiat sau nu.
5. **private int gameSpeed** este viteza jocului
6. **private int levelScale** va preciza nivelul la care a ajuns jucătorul și va crește cu o unitate după fiecare 60 de secunde.

Pentru a determina forma și culoarea pieselor de tetris folosim:

1. **private static Color O_TetrominoColor** = Colors.GreenYellow;
2. **private static Color I_TetrominoColor** = Colors.Red;
3. **private static Color T_TetrominoColor** = Colors.Gold;
4. **private static Color S_TetrominoColor** = Colors.Violet;
5. **private static Color Z_TetrominoColor** = Colors.DeepSkyBlue;
6. **private static Color J_TetrominoColor** = Colors.Cyan;
7. **private static Color L_TetrominoColor** = Colors.LightSeaGreen;

I. Verificare coliziunilor

```
private bool rotationCollided(int _rotation)
{
    if ( checkCollided(0, currentTetrominoWidth - 1))
        { return true; }//Coliziune jos
    else if (checkCollided(0, - (currentTetrominoWidth - 1))) { return true; }
// Coliziune sus
    else if (checkCollided(0, -1)) { return true; }// Top collision
    else if (checkCollided(-1, currentTetrominoWidth - 1)){ return true; }
// Coliziune stanga
    else if (checkCollided(1, currentTetrominoWidth - 1)) { return true; }
// Coliziune dreapta
    return false;
}

// verifică dacă s-au unit în alte forme
```

II. private void TetroCollided()

```

    {
        bottomCollided = checkCollided(0, 1);
        leftCollided = checkCollided(-1, 0);
        rightCollided = checkCollided(1, 0);
    }
}

```

III. Verificare daca jocul trebuie oprit

```

private void shapeStoped() {
    timer.Stop();
    playSound(0);
    if (downPos <= 2)
    {
        gameOver();
        return;
    }
}

```

IV. Stergerea unei linii in cazul completarii acestia

```

if (squareCount == gridColumn)
{
    playSound(1);
    deleteLine(row);
    scoreTxt.Text = getScore().ToString();
    checkComplete();
}

```



Codul sursă:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using System.Windows.Threading;
namespace Tetris
{
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    ///

```

```

public partial class MainWindow : Window
{
    private const int GAMESPEED = 700;
    // Lista pentru adaugarea a doua sunete
    List<System.Media.SoundPlayer> soundList=new
    List<System.Media.SoundPlayer>();
    DispatcherTimer timer;
    Random shapeRandom;
    private int rowCount = 0;
    private int columnCount = 0;
    private int leftPos = 0;
    private int downPos = 0;
    private int currentTetrominoWidth;
    private int currentTetrominoHeigth;
    private int currentShapeNumber;
    private int nextShapeNumber;
    private int tetrisGridColumn;
    private int tetrisGridRow;
    private int rotation = 0;
    private bool gameActive = false;
    private bool nextShapeDrawed = false;
    private int[,] currentTetromino = null;
    private bool isRotated = false;
    private bool bottomCollided = false;
    private bool leftCollided = false;
    private bool rightCollided = false;
    private bool isGameOver = false;
    private int gameSpeed;
    private int levelScale = 60;
    // la fiecare 60 de secunde creste nivelul pana la 10
    private double gameSpeedCounter=0;
    private int gameLevel=1;
    private int gameScore = 0;
    private static Color O_TetrominoColor =
    Colors.GreenYellow;
    private static Color I_TetrominoColor = Colors.Red;
    private static Color T_TetrominoColor = Colors.Gold;
    private static Color S_TetrominoColor =
    Colors.Violet;
    private static Color Z_TetrominoColor =
    Colors.DeepSkyBlue;
    private static Color J_TetrominoColor = Colors.Cyan;
    private static Color L_TetrominoColor =
    Colors.LightSeaGreen;
    List<int> currentTetrominoRow = null;
    List<int> currentTetrominoColumn = null;

```

```

        // culoarea pentru forma de tetromino
        Color[] shapeColor = {
O_TetrominoColor,I_TetrominoColor,
T_TetrominoColor,S_TetrominoColor,
Z_TetrominoColor,J_TetrominoColor,
L_TetrominoColor
};

        string[] arrayTetrominos = { "", "O_Tetromino"
, "I_Tetromino_0",
"T_Tetromino_0", "S_Tetromino_0",
"Z_Tetromino_0", "J_Tetromino_0",
"L_Tetromino_0"
};

#region Array of tetrominos shape
        // aranjarea formelor de tetris
        //---- tetromino patrat-----
public int[,] O_Tetromino = new int[2, 2] { { 1, 1 },
// * *
{ 1, 1 } }; // * *

//---- tetromino in forma de I-----
public int[,] I_Tetromino_0 = new int[2, 4] { { 1, 1,
1, 1 }, { 0, 0, 0, 0 } }; // * * * *

public int[,] I_Tetromino_90 = new int[4, 2] { { 1, 0
}, // *
{ 1, 0 }, // *
{ 1, 0 }, // *
{ 1, 0 } }; // *

        //---- Tetromino in forma de T -----
public int[,] T_Tetromino_0 = new int[2, 3] { { 0, 1, 0 },
// *
{ 1, 1, 1 } }; // * * *

public int[,] T_Tetromino_90 = new int[3, 2] { { 1, 0 },
// *
{ 1, 1 }, // *
{ 1, 0 } }; // *

```

```

public int[,] T_Tetromino_180 = new int[2, 3]
{{1,1,1},
// * * *
{0,1,0}}; //      *
public int[,] T_Tetromino_270 = new int[3, 2] {{0,1},
{1,1},
{0,1}};
-----tetromino in forma de S-----
public int[,] S_Tetromino_0 = new int[2, 3] {{0,1,1
{1,1,0}};
public int[,] S_Tetromino_90 = new int[3, 2] {{1,0},
{1,1},{0,1}};
-----Tetromino in forma de Z-----
public int[,] Z_Tetromino_0 = new int[2, 3] {{1,1,0},
{0,1,1}};
public int[,] Z_Tetromino_90 = new int[3, 2] {{0,1},
{1,1}, {1,0}};
-----Tetromino in forma de J-----
public int[,] J_Tetromino_0 = new int[2, 3] {{1,0,0
{1,1,1}};           public int[,] J_Tetromino_90 =
new int[3, 2] {{1,1},
{1,0}, {1,0}};
           public int[,] J_Tetromino_180 = new int[2, 3]
{{1,1,1}, {0,0,1}};

public int[,] J_Tetromino_270 = new int[3, 2] {{0,1},
{0,1}, {1,1 }};
----- Tetromino in forma de L-----
           public int[,] L_Tetromino_0 = new int[2, 3]
{{0,0,1}, {1,1,1}};
public int[,] L_Tetromino_90 = new int[3, 2] {{1,0},
{1,0}, {1,1}};
public int[,] L_Tetromino_180 = new int[2, 3]
{{1,1,1}, {1,0,0}};

public int[,] L_Tetromino_270 = new int[3, 2] {{1,1
{0,1},{0,1 }};
           public object Task { get; private set; }
#endregion
public MainWindow()
{
    InitializeComponent();
    gameSpeed = GAMESPEED;
//eveniment creat pentru apasarea tastelor
    KeyDown += MainWindow_KeyDown;
// timpul initial
    timer = new DispatcherTimer();
}

```

```

        timer.Interval = new TimeSpan(0, 0, 0, 0,
gameSpeed); // 700 millisecond
        timer.Tick += Timer_Tick;
tetrisGridColumn =
tetrisGrid.ColumnDefinitions.Count;
tetrisGridRow = tetrisGrid.RowDefinitions.Count;
shapeRandom = new Random();
currentShapeNumber = shapeRandom.Next(1, 8);
nextShapeNumber = shapeRandom.Next(1, 8);
nextTxt.Visibility = levelTxt.Visibility=
GameOverTxt.Visibility = Visibility.Collapsed;
                // adaugarea a doua sunete in lista
soundList.Add(new
System.Media.SoundPlayer(Properties.Resources.collide
d));
soundList.Add(new
System.Media.SoundPlayer(Properties.Resources.deleteL
ine));
    }

// eveniment cheie pentru forma tetrominelor dar si
//pentru rotatia acestora
private void MainWindow_KeyDown(object sender,
KeyEventArgs e)
{
    if (!timer.IsEnabled) { return; }
    switch (e.Key.ToString())
    {
        case "Up":
            rotation += 90;
        if (rotation > 270) { rotation = 0; }
            shapeRotation(rotation);
            break;
        case "Down":
            downPos++;
            break;
        case "Right":
// verifica daca piesele s-au unit
            TetroCollided();
        if (!rightCollided) { leftPost++; }
            rightCollided = false;
            break;
        case "Left":
// verifica daca piesele s-au unit
            TetroCollided();
            if (!leftCollided) { leftPost--; }
            leftCollided = false;
            break;
    }
}

```

```

        }
        moveShape();
    }

    // rotatia Tetrominelor
    private void shapeRotation(int _rotation)
    {
        // verifica daca s-au unit
        if (rotationCollided(rotation))
        {
            rotation -= 90;
            return;
        }

        if
(arrayTetrominos[currentShapeNumber].IndexOf("I_") == 0)
        {
if (_rotation > 90) { _rotation = rotation = 0; }
            currentTetromino =
getVariableByString("I_Tetromino_" + _rotation);
        }
        else if
(arrayTetrominos[currentShapeNumber].IndexOf("T_") == 0)
        {
            currentTetromino =
getVariableByString("T_Tetromino_" + _rotation);
        }
        else if
(arrayTetrominos[currentShapeNumber].IndexOf("S_") == 0)
        {
if (_rotation > 90) { _rotation = rotation = 0; }
            currentTetromino =
getVariableByString("S_Tetromino_" + _rotation);
        }
        else if
(arrayTetrominos[currentShapeNumber].IndexOf("Z_") == 0)
        {
if (_rotation > 90) { _rotation = rotation = 0; }
            currentTetromino =
getVariableByString("Z_Tetromino_" + _rotation);
        }
        else if
(arrayTetrominos[currentShapeNumber].IndexOf("J_") == 0)
        {

```

```

currentTetromino = getVariableByString("J_Tetromino_"
+ _rotation);
}
else if
(arrayTetrominos[currentShapeNumber].IndexOf("L_") ==
0)
{
currentTetromino = getVariableByString("L_Tetromino_"
+ _rotation);
}
else if
(arrayTetrominos[currentShapeNumber].IndexOf("O_") ==
0)
// nu se roteste
{
    return;
}

isRotated = true;
addShape(currentShapeNumber, leftPos, downPos);
}
// miscarea pieselor in jos odata cu ticalitul
//ceasului
private void Timer_Tick(object sender, EventArgs e)
{
    downPos++;
    moveShape();
    if (gameSpeedCounter >= levelScale)
    {
        if (gameSpeed >= 50)
        {
            gameSpeed -= 50;
            gameLevel++;
        levelTxt.Text = "Level: " + gameLevel.ToString();
        }
        else { gameSpeed = 50; }
        timer.Stop();
    timer.Interval = new TimeSpan(0, 0, 0, 0, gameSpeed);
        timer.Start();
        gameSpeedCounter = 0;
    }
    gameSpeedCounter += (gameSpeed/1000f);

}
// metoda de oprire a butonului start
private void Button_Click_1(object sender,
RoutedEventArgs e)
{

```

```

        if(isGameOver)
        {
            tetrisGrid.Children.Clear();
            nextShapeCanvas.Children.Clear();
            GameOverTxt.Visibility = Visibility.Collapsed;
            isGameOver = false;
        }
        if(!timer.IsEnabled)
        {
            if (!gameActive) { scoreTxt.Text = "0"; leftPos = 3;
            addShape(currentShapeNumber, leftPos); }
            nextTxt.Visibility = levelTxt.Visibility =
            Visibility.Visible;
            levelTxt.Text = "Level: " + gameLevel.ToString();
            timer.Start();
            startStopBtn.Content = "Stop Game";
            gameActive = true;
        }
        else
        {
            timer.Stop();
            startStopBtn.Content = "Start Game";
        }
    }
    // adaugarea unei noi piese in joc
    private void addShape(int shapeNumber,int _left=0,int
    _down=0)
    {
        // inlaturarea pozitiei piesei anterioare
        removeShape();
        currentTetrominoRow = new List<int>();
        currentTetrominoColumn = new List<int>();
        Rectangle square=null;
        if (!isRotated)
        {
            currentTetromino = null;
        currentTetromino =
        getVariableByString(arrayTetrominos [shapeNumber].ToString() );
        }
        int firstDim = currentTetromino.GetLength(0);
        int secondDim = currentTetromino.GetLength(1);
        currentTetrominoWidth = secondDim;
        currentTetrominoHeigth = firstDim;
        // doar pentru Tetromino in forma de I
        if (currentTetromino == I_Tetromino_90)
        {
            currentTetrominoWidth = 1;
        }
    }

```

```

        else if (currentTetromino ==
I_Tetromino_0) { currentTetrominoHeigth = 1; }
            for (int row=0;row < firstDim;row++)
            {
                for (int column=0; column < secondDim; column++)
                {
                    int bit = currentTetromino[row, column];
                    if (bit == 1 )
                    {
square = getBasicSquare(shapeColor[shapeNumber - 1]);

tetrisGrid.Children.Add(square);
                    square.Name = "moving_" +
Grid.GetRow(square)+"_"+Grid.GetColumn(square);
                    if (_down >=
tetrisGrid.RowDefinitions.Count-
currentTetrominoHeigth)
                    {
_down = tetrisGrid.RowDefinitions.Count -
currentTetrominoHeigth;
                    }
                    Grid.SetRow(square, rowCount + _down);
                    Grid.SetColumn(square, columnCount + _left);
                    currentTetrominoRow.Add(rowCount + _down);
                    currentTetrominoColumn.Add(columnCount + _left);

                    }
                    columnCount++;
                }
                columnCount = 0;
                rowCount++;
            }
            columnCount = 0;
            rowCount = 0;
            if (!nextShapeDrawed)
            {
                drawNextShape(nextShapeNumber);
            }
        }
// adaugarea unei noi forme intr-o locatie
private void moveShape()
{
    leftCollided = false;
    rightCollided = false;

    // verifica daca s-au unit
    TetroCollided();
    if (leftPos > (tetrisGridColumn -
currentTetrominoWidth))
    {

```

```

leftPos = (tetrisGridColumn - currentTetrominoWidth);
        }
        else if (leftPos < 0) { leftPos = 0; }

        if (bottomCollided)
        {
            shapeStoped();
            return;
        }
addShape(currentShapeNumber, leftPos, downPos);
}

// verifica unirea daca
private bool rotationCollided(int _rotation)
{
    if (checkCollided(0, currentTetrominoWidth - 1)) { return true; } //Coliziune jos
    else if (checkCollided(0, -(currentTetrominoWidth - 1))) { return true; } // Coliziune sus
    else if (checkCollided(0, -1)) { return true; } // Top collision
    else if (checkCollided(-1, currentTetrominoWidth - 1)) { return true; } // Coliziune stanga
    else if (checkCollided(1, currentTetrominoWidth - 1)) { return true; } // Coliziune dreapta
    return false;
}
// verifica daca s-au unit in alte forme
private void TetroCollided()
{
    bottomCollided = checkCollided(0, 1);
    leftCollided = checkCollided(-1, 0);
    rightCollided = checkCollided(1, 0);
}
//verifica daca s-au unit
private bool checkCollided(int _leftRightOffset,
int _bottomOffset)
{
    Rectangle movingSquare;
    int squareRow = 0;
    int squareColumn = 0;
    for (int i = 0; i <= 3; i++)
    {
        squareRow = currentTetrominoRow[i];
        squareColumn = currentTetrominoColumn[i];
        try
        {
movingSquare = (Rectangle)tetrisGrid.Children

```

```

        .Cast<UIElement>()
.FIRSTOrDefault(e => Grid.GetRow(e) == squareRow +
_bottomOffset && Grid.GetColumn(e) ==
squareColumn+_leftRightOffset);
if (movingSquare != null)
{
    if (movingSquare.Name.IndexOf("arrived") == 0)
    {
        return true;
    }
}
catch { }
}

if(downPos > (tetrisGridRow -currentTetrominoHeigth))
{ return true; }
return false;
}

// Extrage o piesa noua de tetromino in
//nextShapeCanvas
private void drawNextShape(int shapeNumber)
{
    nextShapeCanvas.Children.Clear();
    int[,] nextShapeTetromino = null;
nextShapeTetromino =
getVariableByString(arrayTetrominos[shapeNumber]);
int firstDim = nextShapeTetromino.GetLength(0);
int secondDim = nextShapeTetromino.GetLength(1);
    int x = 0;
    int y = 0;
    Rectangle square;
    for (int row = 0; row < firstDim; row++)
    {
        for (int column = 0; column < secondDim; column++)
        {
            int bit = nextShapeTetromino[row, column];
if (bit == 1)
{
    square = getBasicSquare(shapeColor[shapeNumber-1]);
}

nextShapeCanvas.Children.Add(square);
    Canvas.SetLeft(square, x);
    Canvas.SetTop(square, y);
}
x += 25;
}
x = 0;
y += 25;
}
}

```

```

        nextShapeDrawed = true;
    }

// Aceasta metoda este apelata cand piese s-a unit
// sau a atins josul careului
private void shapeStoped()
{
    timer.Stop();
    playSound(0);
    //conditia pentru ca jocul sa se termine
    if (downPos <= 2)
    {
        gameOver();
        return;
    }

    int index = 0;
    while (index < tetrisGrid.Children.Count)
    {
        UIElement element = tetrisGrid.Children[index];
        if (element is Rectangle)
        {
            Rectangle square = (Rectangle)element;
            if (square.Name.IndexOf("moving_") == 0)
            {
                // Inlocuirea numelui pieselor din "in miscare" in
                // "ajunse"
                string newName= square.Name.Replace("moving_",
                "arrived_");
                square.Name=newName;
            }
            index++;
        }
        // verifica daca linia e completa si muta
        // piesele cu un nivel mai jos
        checkComplete();
        reset();
        timer.Start();
    }
}

// metoda de a verifica daca o linie este completa
private void checkComplete()
{
    int gridRow =
tetrisGrid.RowDefinitions.Count;
    int gridColumn =
tetrisGrid.ColumnDefinitions.Count;
    int squareCount = 0;
}

```

```

        for (int row = gridRow; row >= 0; row--)
        {
            squareCount = 0;
        for (int column = gridColumn; column >= 0; column--)
        {
            Rectangle square;
            square = (Rectangle)
tetrisGrid.Children
            .Cast<UIElement>()
            .FirstOrDefault(e => Grid.GetRow(e) == row &&
Grid.GetColumn(e) == column);
            if (square != null)
            {
                if (square.Name.IndexOf("arrived") == 0)
                {
                    squareCount++;
                }
            }
        }

// Daca squareCount == gridColumn inseamna //ca linia
//este completa si trebuie stearsa
        if (squareCount == gridColumn)
        {
            playSound(1);
            deleteLine(row);
            scoreTxt.Text =
getScore().ToString();
            checkComplete();
        }
    }
    // sterge linia de patrate
private void deleteLine(int row)
{
    // Delete complete line
for(int i=0;i<tetrisGrid.ColumnDefinitions.Count;i++)
{
    Rectangle square;
    try
    {
        square =
(Rectangle)tetrisGrid.Children
        .Cast<UIElement>()
        .FirstOrDefault(e => Grid.GetRow(e) == row &&
Grid.GetColumn(e) == i);

tetrisGrid.Children.Remove(square);
    }
}

```

```

        catch { }

    }

    // muta cu un nivel in jos restul pieselor
    foreach (UIElement element in
tetrisGrid.Children)
{
    Rectangle square =
(Rectangle)element;
    if(square.Name.IndexOf("arrived")==0 &&
Grid.GetRow(square)<=row)
    {
        Grid.SetRow(square,
Grid.GetRow(square) + 1);
    }
}
// modifica scorul
private int getScore()
{
    gameScore += 50 * gameLevel;
    return gameScore;
}

// resetarea
private void reset()
{
    downPos = 0;
    leftPos = 3;
    isRotated = false;
    rotation = 0;
    currentShapeNumber = nextShapeNumber;
    if (!isGameOver) {
addShape(currentShapeNumber, leftPos); }
    nextShapeDrawed = false;
    shapeRandom = new Random();
    nextShapeNumber = shapeRandom.Next(1, 8);
    bottomCollided = false;
    leftCollided = false;
    rightCollided = false;
}
// resetarea jocului cand s-a terminat
private void gameOver()
{
    isGameOver = true;
    reset();
    startStopBtn.Content = "Start Game";
    GameOverTxt.Visibility = Visibility.Visible;
    rowCount = 0;
}

```

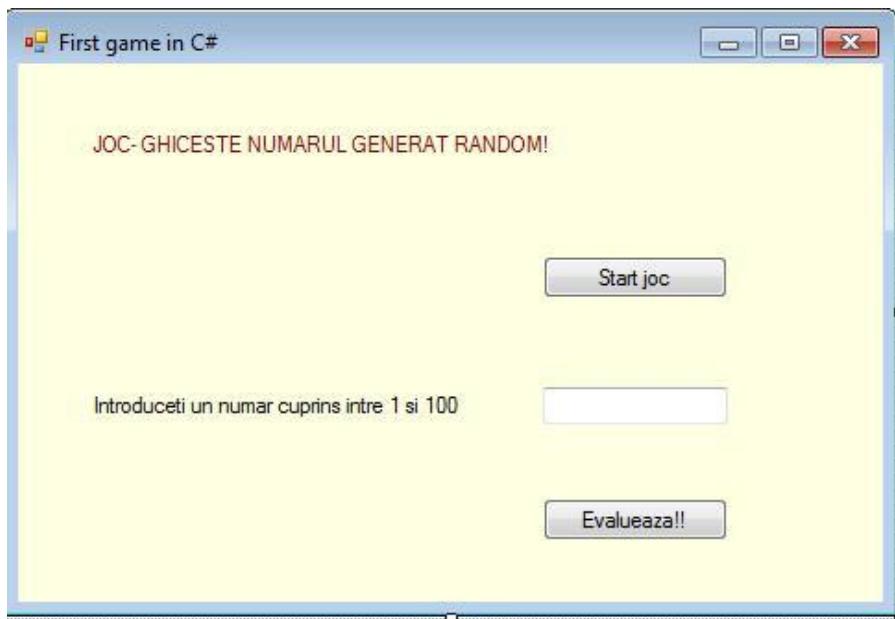
```

        columnCount = 0;
        leftPos = 0;
        gameSpeedCounter = 0;
        gameSpeed = GAMESPEED;
        gameLevel = 1;
        gameActive = false;
        gameScore = 0;
        nextShapeDrawed = false;
        currentTetromino = null;
        currentShapeNumber = shapeRandom.Next(1, 8);
        nextShapeNumber = shapeRandom.Next(1, 8);
        timer.Interval = new TimeSpan(0, 0, 0, 0, gameSpeed);
    }
    // Inlaturarea formei din rand
    private void removeShape()
    {
        int index = 0;
        while (index < tetrisGrid.Children.Count)
        {
            UIElement element = tetrisGrid.Children[index];
            if (element is Rectangle)
            {
                Rectangle square = (Rectangle)element;
                if (square.Name.IndexOf("moving_") == 0)
                {
                    tetrisGrid.Children.Remove(element);
                    index = -1;
                }
                index++;
            }
        }
    }
    // crearea patratelului din forma pieselor de Tetris
    private Rectangle getBasicSquare(Color rectColor)
    {
        Rectangle rectangle = new Rectangle();
        rectangle.Width = 25;
        rectangle.Height = 25;
        rectangle.StrokeThickness = 1;
        rectangle.Stroke = Brushes.White;
        rectangle.Fill =
        getGradientColor(rectColor);
        return rectangle;
    }
    // culoarea de baza pentru patratelele normale
    private LinearGradientBrush getGradientColor( Color clr)
    {
        LinearGradientBrush gradientColor = new
        LinearGradientBrush();
        gradientColor.StartPoint = new Point(0, 0);
        gradientColor.EndPoint = new Point(1, 1.5);
    }
}

```

```
GradientStop black = new GradientStop();
black.Color = Colors.Black;
black.Offset = -1.5;
gradientColor.GradientStops.Add(black);
GradientStop other = new GradientStop();
other.Color = clr;
other.Offset = 0.70;
gradientColor.GradientStops.Add(other);
return gradientColor;
}
// Accesarea variabilei dupa numele subramurii
private int[,] getVariableByString(string variable)
{
    return
(int[,])this.GetType().GetField(variable).GetValue(th
is);
}
// Pornirea sunetului. index=0 e pentru alaturarea
//pieselor. wav si index=1 pentru stergerea liniei,
deleteLine.wav.
private void playSound(int index)
{
    soundList[index].Play();
} } }
```

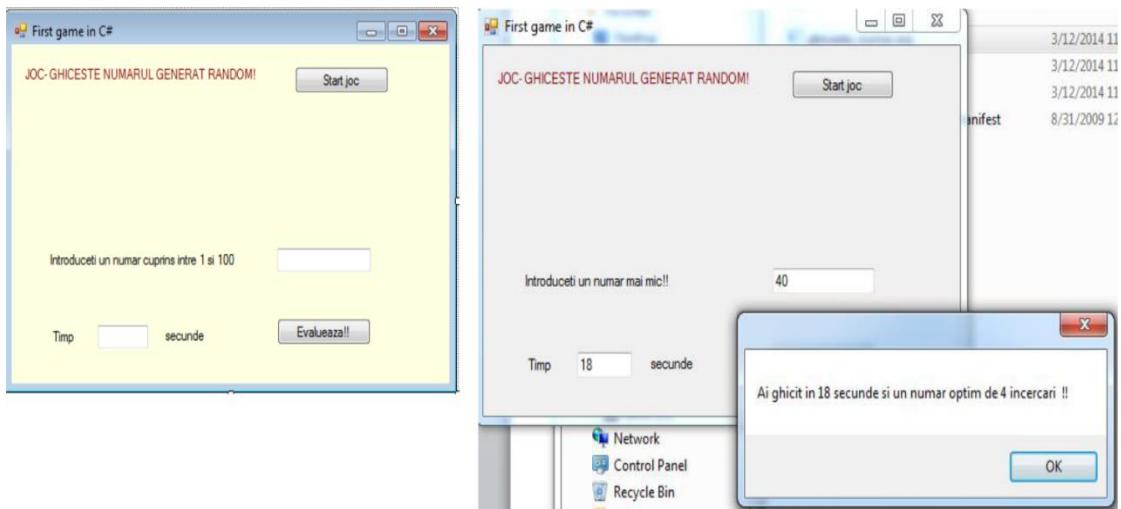
Joule 14 –Ghicește un număr generat random de computer



```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
namespace ghiceste_numar
{
    public partial class Form1 : Form
    {
        int nr_de_ghicit,nr_optim_incercari=7;
        //nr de incercari maxim= log in baza //2 din valoarea maxima,100
        int nr_incercari=0;
        private void win()
        {
            if(nr_incercari<=nr_optim_incercari)
                MessageBox.Show("Ai ghicit intr-un numar optim de " +
                    Convert.ToString(nr_incercari) + " incercari !!");
            else
                MessageBox.Show("Ai ghicit intr-un numar nepermis de mare de " +
                    Convert.ToString(nr_incercari) + " incercari !!");
        }
        public Form1()
        { InitializeComponent(); }
```

```
private void Form1_MouseHover(object sender, EventArgs e)
{ this.BackColor = Color.Beige; }
private void Form1_MouseLeave(object sender, EventArgs e)
{ this.BackColor = SystemColors.Control; }
private void button2_Click(object sender, EventArgs e)
{ Random rand = new Random();
nr_de_ghicit = rand.Next(1, 100);
}
private void button1_Click(object sender, EventArgs e)
{ int nr = Convert.ToInt32(textBox1.Text);
if (nr == nr_de_ghicit)
win();
else
{
nr_incercari++;
if (nr < nr_de_ghicit)
label1.Text = "Introduceti un numar mai mare!!";
else
label1.Text = "Introduceti un numar mai mic!!";
textBox1.Clear(); }
}
}}
```

Jocul 15- Ghicește un număr (varianta cu cronometru)



Form1 : Form

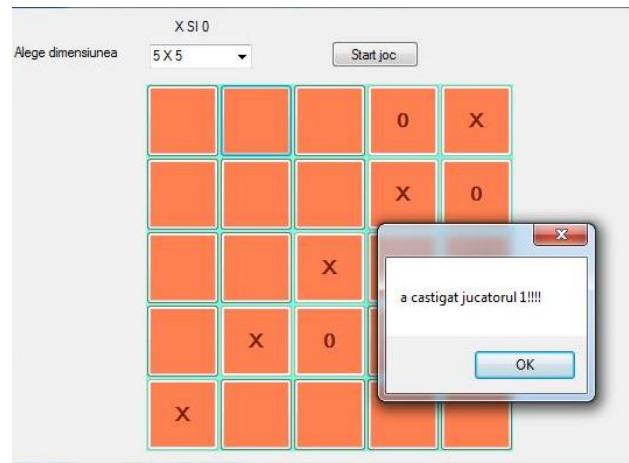
```
{  
    int nr_de_ghicit, nr_optim_incercari=7; //nr de incercari  
    maxim= log in baza 2 din valoarea maxima, 100  
    int nr_incercari=0;  
    int durata_joc = 0;  
    //form methods  
    public Form1()  
    {  
        InitializeComponent();  
    }  
    private void Form1_Load(object sender, EventArgs e)  
    {  
        label1.Hide(); label3.Hide(); label4.Hide();  
        textBox2.Hide(); textBox1.Hide(); button1.Hide();  
    }  
    private void Form1_MouseHover(object sender, EventArgs e)  
    {  
        this.BackColor = Color.Beige;  
    }  
    private void Form1_MouseLeave(object sender, EventArgs e)  
    {  
        this.BackColor = SystemColors.Control;  
    }  
    //my methods
```

```

private void win()
{
    if (nr_incercari <= nr_optim_incercari)
        MessageBox.Show("Ai ghicit in "+durata_joc.ToString()+" secunde si un numar optim de " +
            Convert.ToString(nr_incercari) + " incercari !!");
    else
        MessageBox.Show("Ai ghicit in "+durata_joc.ToString()+" secunde si un numar nepermis de mare de " +
            Convert.ToString(nr_incercari) + " incercari !!");
}
private void button2_Click(object sender, EventArgs e)//start joc
{
    Random rand = new Random();
    nr_de_ghicit = rand.Next(1, 100);
    timer1.Start();
    label1.Show(); label3.Show(); textBox2.Show();
    label4.Show();
    textBox1.Show(); button1.Show();
    textBox1.Focus();
}
private void timer1_Tick(object sender, EventArgs e)
{
    durata_joc++;
    textBox2.Text = durata_joc.ToString();
}
private void button1_Click(object sender, EventArgs e)//evaluateaza
{
    int nr = Convert.ToInt32(textBox1.Text);
    if (nr == nr_de_ghicit)
    {
        win();
        timer1.Stop();
    }
    else
    {
        nr_incercari++;
        if (nr < nr_de_ghicit)
            label1.Text = "Introduceti un numar mai mare!!";
        else
            label1.Text = "Introduceti un numar mai mic!!";
        textBox1.Clear(); } } } }
```

Jocul 16-X și 0

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
namespace xsi0
{
    public partial class Form1 : Form
    {
        int n,clic;
        Button[,] btn;
        bool finish_joc(int i, int j)
        {
            int s = 0;
            string r = btn[i, j].Text;
            for (int k = 0; k < n; k++)
                if (btn[i, k].Text == r)
                    s++;
            if (s == n)
                return true;//linie
            s = 0;
            for (int k = 0; k < n; k++)
                if (btn[k, j].Text == r)
                    s++;
            if (s == n)
                return true;//coloana
            if (i == j)//diagonala principala
            {
                s = 0;
                for (int k = 0; k < n; k++)
                    if (btn[k, k].Text == r) s++;
                if (s == n) return true; }
            if (i == n-j-1)//diagonala secundara
            {
                s = 0;
                for (int k = 0; k < n; k++)
                    if (btn[k, n - k - 1].Text == r)
                        s++;
                if (s == n) return true;
            }
            return false;
        }
        public Form1()
        { InitializeComponent(); }
        private void Form1_Load(object sender, EventArgs e)
        { panel1.Hide(); }
        private void btn_click(object sender, EventArgs e)
```



```

{
Button butonclicuit = (Button)sender;
clic++;
butonclicuit.Font = new Font(FontFamily.GenericSansSerif,
12.0F, FontStyle.Bold);
if (clic % 2==1)
{
butonclicuit.ForeColor = Color.Black;
butonclicuit.Text = "X";
}
else
{
butonclicuit.ForeColor = Color.Red;
butonclicuit.Text = "0";
}
butonclicuit.Enabled = false;
Point p = butonclicuit.Location;
int x = p.X, y = p.Y,i,j;
i = x / (315 / n);
j = y / (315 / n);
if (finish_joc(i,j)&&clic%2==1)
MessageBox.Show("a castigat jucatorul 1!!!!");
if (finish_joc(i,j) && clic % 2 == 0)
MessageBox.Show("a castigat jucatorul 2!!!!");
}
private void init_joc(int n) {
int i,j;
btn=new Button[n,n];
for (i = 0; i < n; i++)
for (j = 0; j < n; j++)
{
btn[i,j] = new Button();
//btn[i, j].Name = "btn" + i.ToString()+j.ToString();
btn[i,j].BackColor = Color.Coral;
btn[i,j].ForeColor = Color.DarkGreen;
btn[i,j].Location = new Point(i * (315/n), j * (315/n));
btn[i,j].Size = new Size(315 / n, 315 / n);
btn[i,j].Click += new EventHandler(btn_click);
panel1.Controls.Add(btn[i,j]);
}
clic = 0;
}
private void btn_start_Click(object sender, EventArgs e)
{ panel1.Show(); panel1.BackColor = Color.Aquamarine;
if(comboBox1.SelectedIndex==0)
n=3;
if (comboBox1.SelectedIndex == 1)
n = 5;
if (comboBox1.SelectedIndex == 2)
n = 7;
init_joc(n);
}
}
}

```

Bibliografie

1. Conger, David, *Programarea în C#*, Editura All, 2008
2. Schildt, Herbert, *C#*, Editura Teora, 2002
3. Pappas,Chris H.; Murray, William H., *C# pentru programarea Web*, Editura All, 2002
4. Reynolds-Haertle, Robin A., *Programare orientată pe obiecte cu Microsoft Visual Basic.Net și Microsoft Visual C#.Net*, Editura Teora, 2002
5. Ioan Jinga, Elena Istrate, *Manual de pedagogie*, București, Editura „ALL Educational”, 2001
6. Ioan Cerghit, *Metode de învățământ*, Iași, Editura „Polirom”, 2006
7. Carmen Petre, Ștefania Crăciunoiu, Daniela Popa, Carmen Iliescu, *Metodica predării informaticii și tehnologiei informației*, Craiova, Editura „Arves”, 2002.
8. Adrian Adăscăliței, *Instruire asistată de calculator*, Iași, Editura „Polirom”, 2007
9. Laura Grindei, Bogdan Orza, Aurel Vlaicu, *Tehnologii multimedia cu aplicații interactive în eLearning*, Cluj Napoca, Editura „Albastră”, 2007.
10. Ioan Cerghit, *Sisteme de instruire alternative și complementare-structuri, stiluri și strategii*, București, Editura „Aramis Print SRL ”, 2002.
11. Mihaela Brut, *Instrumente pentru eLearning, ghidul informatic al profesorului modern*, Iași, Editura „Polirom”, 2006.
12. Prof. Radu Jugureanu, *AEL eContent Manager*, SIVECO România.- www.siveco.ro
13. Ionescu, Miron; Radu, Ioan, *Didactica modernă*, Cluj-Napoca, Editura Dacia,2001,
14. R. Stradling , *Să înțelegem istoria secolului XX* , Editura Sigma, București, 2002
15. M. Jalobeanu , WWW în învățământ. Instruirea prin Internet , Cluj Napoca, 2001
16. Ana-Maria Suduc, Mihai Bîzoi, Gabriel Gorghi, *Tehnici informaționale computerizate- Aplicații destinate cadrelor didactice- Târgoviște*, Editura „Bibliotheca ”, 2008.
17. Mihai Bîzoi, Ana-Maria Suduc, *Bazele programării orientate pe obiecte- Aplicații în limbajul SMAL TALC*, Târgoviște, Editura „Bibliotheca ”, 2008.
18. I. Neacșu. *Instruire și învățare*. Editura științifică și enciclopedică, București, 1990.
19. Ion T.Radu. *Evaluarea în procesul didactic*. E.D. P., București, 2004.
20. M. Ionescu. *Didactica modernă*. Editura „Dacia”, Cluj-Napoca 2004.
21. M. Ionescu. *Lecția între proiect și realizare*. Editura “Dacia”, Cluj-Napoca, 1982.

22. Novak Andrei, *Metode statistice în pedagogie și psihologie*. E.D.P., București, 1997.
23. O. Oprea. *Tehnologia instruirii didactice*. Editura Didactică și Pedagogică, București 1979.
24. P. Mureșan. *Învățarea eficientă și rapidă*. Editura “Ceres”. București 1990

Webiografie

1. [http://depmath.ulbsibiu.ro/chair2/craciunas/soft %20educational.pdf](http://depmath.ulbsibiu.ro/chair2/craciunas/soft%20educational.pdf)
2. www.advancedelearning.com
3. www.didactic.ro/files/12/proiectdeprogramaoptional.doc
4. <https://www.mooict.com/>
5. <http://msdn.microsoft.com>
6. <http://ro.wikipedia.org>
7. <http://www.tutorialspoint.com/>
8. <https://www.mooict.com/c-tutorial-create-a-breakout-game-in-visual-studio/>
9. <http://stackoverflow.com>
10. <https://youtu.be/tIA9RcpPfgc>
11. <http://www.tutorialspoint.com/>
12. <https://www.mooict.com/c-tutorial-create-a-save-the-eggs-item-drop-game-in-visual-studio/>
13. <https://www.mooict.com/c-tutorial-create-a-full-space-invaders-game-using-visual-studio/>
14. - <http://netcode.ru/dotnet/?artID=6778>
15. - <https://hugogiraudel.com/2012/12/20/tetris>
16. <https://www.mooict.com/c-tutorial-create-a-t-rex-endless-runner-game-in-visual-studio/>
17. <https://www.mooict.com/c-tutorial-create-a-breakout-game-in-visual-studio/>
18. <https://www.mooict.com/c-tutorial-create-a-superhero-memory-game/>

Căutarea unui limbaj de programare perfect este ținta multor inovatori ai programării. În această domeniu *C# (CSharp)* este unul din cele mai evoluate limbi și reprezintă pasul următor în evoluția limbajelor de programare.

Lucrarea de față prezintă aplicații de laborator pentru orele de Programare vizuală, realizate în *Visual Studio Express C# 2019*.

Fiecare fișă de laborator va descrie pașii de realizare a aplicației, codul sursă și imaginea formei finale.

Aplicațiile descrise în această lucrare sunt preluate de pe diverse site-uri de specialitate și adaptate cerințelor și nivelului claselor de liceu. De asemenea au fost incluse materiale didactice ale profesorilor de informatică din Botoșani.

Această lucrare vine în sprijinul profesorilor ce predau disciplina informatică și poate fi un instrument util chiar și celor care vor să învețe *Programarea Visuală cu C#*.

Vă dorim o programare plăcută și interesantă!

Botoșani

2020

