Lucrarea 3

- **Obiective**

Obiectivele acestei lucrări sunt prezentarea unei aplicații C++ și a unei aplicații C# , având ca temă sărbătorile de iarnă.

Aplicația C++ va trebui să conțină cel puțin o instrucțiune repetitivă imbricată într-o altă instrucțiune repetitivă, iar aplicația din WFA C# va folosi instrucțiuni repetitive pentru a desena o ghirlandă plină cu decorațiuni de Crăciun.

Ghirlanda va trebui să respecte următoarele cerințe:

-se va defini un set propriu de figuri create prin instrucțiuni repetitive, care vor alcătui ornamentele; în acest caz va fi vorba de globuri hand-made;

-figurile din set vor fi afișate aleator;

-la modificarea dimensiunii ferestrei se vor afișa aleator alte globuri din setul definit;

-dacă se modifică lățimea ferestrei, figurile trebuie să umple tot ecranul;

-dacă se modifică înălțimea ferestrei, figurile trebuie să se rescaleze.

- **Codul Sursa al aplicației C#WFA**

```
namespace Lucrarea3_V2
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        int alfa_g;// unghiul in grade
        double pi = System.Math.PI;
```

```
void F1(int x0)
{
    int x, y;
    int y0 = Height / 3;
    int xc = x0 + y0 / 2;
    int yc = y0*3/2;
    System.Drawing.Graphics Desen;
    System.Drawing.Pen Creion_Green, Creion_Red;
    Desen = CreateGraphics();
    Creion_Green = new System.Drawing.Pen(System.Drawing.Color.Green, 2);
    Creion_Red = new System.Drawing.Pen(System.Drawing.Color.Red, 2);
    Desen.DrawLine(Creion_Green, x0 + y0 / 2, 10, x0 + y0 / 2, y0);
    for (alfa_g = 0; alfa_g < 360; alfa_g += 6)
    {
        double alfa_r = 2 * pi * alfa_g / 360;// unghiul in radiani
        x = System.Convert.ToInt32(xc + y0 / 2 * System.Math.Cos(alfa_r));
        y = System.Convert.ToInt32(yc - y0 / 2 * System.Math.Sin(alfa_r));
        Desen.DrawLine(Creion_Red, xc, yc, x, y);
    }
}

void F2(int x0)
{
    int x, y;
    int y0 = this.Height / 3;
    int xc = x0 + y0 / 2;
    int yc = y0 * 3 / 2;
    System.Drawing.Graphics Desen;
    System.Drawing.Pen Creion_Green, Creion_Pink, Creion_Gray;
    Desen = CreateGraphics();
    Creion_Green = new System.Drawing.Pen(System.Drawing.Color.Green, 2);
    Creion_Pink = new System.Drawing.Pen(System.Drawing.Color.Pink, 3);
    Creion_Gray = new System.Drawing.Pen(System.Drawing.Color.Gray, 2);
    Desen.DrawEllipse(Creion_Gray, x0, y0, y0, y0);
    Desen.DrawLine(Creion_Green, x0 + y0 / 2, 10, x0 + y0 / 2, y0);
    for (alfa_g = 0; alfa_g < 360; alfa_g += 30)
    {
        double alfa_r = 2 * pi * alfa_g / 360;// unghiul in radiani
        x = System.Convert.ToInt32(xc + y0 / 2 * System.Math.Cos(alfa_r));
        y = System.Convert.ToInt32(yc - y0 / 2 * System.Math.Sin(alfa_r));
        Desen.DrawLine(Creion_Pink, xc, yc, x, y);
    }
}

void F3(int x0)
{
    int x, y;
    int y0 = this.Height / 3;
    int xc = x0 + y0 / 2;
    int yc = y0 * 3 / 2;
    System.Drawing.Graphics Desen;
    System.Drawing.Pen Creion_Green, Creion_Gold;
    Desen = CreateGraphics();
    Creion_Green = new System.Drawing.Pen(System.Drawing.Color.Green, 2);
    Creion_Gold = new System.Drawing.Pen(System.Drawing.Color.Gold, 3);
    Desen.DrawEllipse(Creion_Green, x0, y0, y0, y0);
    Desen.DrawLine(Creion_Green, x0 + y0 / 2, 10, x0 + y0 / 2, y0);
    for (alfa_g = 0; alfa_g < 360; alfa_g += 9)
    {
```

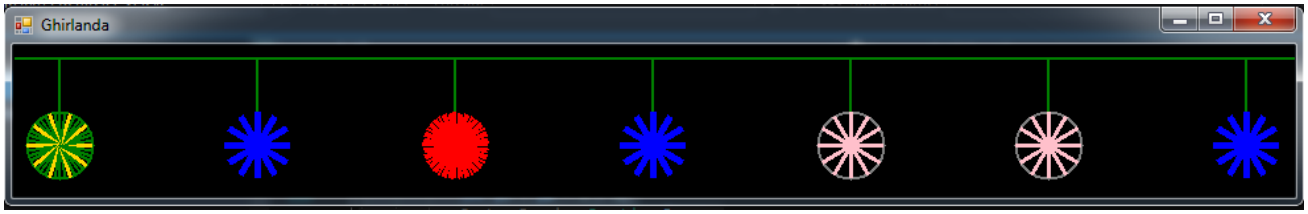
```
        double alfa_r = 2 * pi * alfa_g / 360;// unghiul in radiani
        x = System.Convert.ToInt32(xc + y0 / 2 * System.Math.Cos(alfa_r));
        y = System.Convert.ToInt32(yc - y0 / 2 * System.Math.Sin(alfa_r));
        if (alfa_g % 36 == 0)
            Desen.DrawLine(Creion_Gold, xc, yc, x, y);
        else
            Desen.DrawLine(Creion_Green, xc, yc, x, y);
    }
}

void F4(int x0)
{
    int x, y;
    int y0 = this.Height / 3;
    int xc = x0 + y0 / 2;
    int yc = y0 * 3 / 2;
    System.Drawing.Graphics Desen;
    System.Drawing.Pen Creion_Green, Creion_Blue;
    Desen = CreateGraphics();
    Creion_Green = new System.Drawing.Pen(System.Drawing.Color.Green, 2);
    Creion_Blue = new System.Drawing.Pen(System.Drawing.Color.Blue, 5);
    Desen.DrawLine(Creion_Green, x0 + y0 / 2, 10, x0 + y0 / 2, y0);
    for (alfa_g = 0; alfa_g < 360; alfa_g += 30)
    {
        double alfa_r = 2 * pi * alfa_g / 360;// unghiul in radiani
        x = System.Convert.ToInt32(xc + y0 / 2 * System.Math.Cos(alfa_r));
        y = System.Convert.ToInt32(yc - y0 / 2 * System.Math.Sin(alfa_r));
        Desen.DrawLine(Creion_Blue, xc, yc, x, y);
    }
}

void F5(int x0)
{
    int x, y;
    int y0 = this.Height / 3;
    int xc = x0 + y0 / 2;
    int yc = y0 * 3 / 2;
    System.Drawing.Graphics Desen;
    System.Drawing.Pen Creion_Green,Creion_Orange, Creion_Gray;
    Desen = CreateGraphics();
    Creion_Green = new System.Drawing.Pen(System.Drawing.Color.Green, 2);
    Creion_Orange = new System.Drawing.Pen(System.Drawing.Color.Orange, 3);
    Creion_Gray = new System.Drawing.Pen(System.Drawing.Color.LightGray, 3);
    Desen.DrawEllipse(Creion_Gray, x0, y0, y0, y0);
    Desen.DrawLine(Creion_Green, x0 + y0 / 2, 10, x0 + y0 / 2, y0);
    for (alfa_g = 60; alfa_g < 360; alfa_g += 120)
    {
        double alfa_r = 2 * pi * alfa_g / 360;// unghiul in radiani
        x = System.Convert.ToInt32(xc + y0 / 2 * System.Math.Cos(alfa_r));
        y = System.Convert.ToInt32(yc - y0 / 2 * System.Math.Sin(alfa_r));
        Desen.DrawLine(Creion_Gray, xc, yc, x, y);
        Desen.DrawLine(Creion_Orange, xc+10, yc+10, x, y);
    }
}
```

```
 private void Form1_Paint(object sender, PaintEventArgs e)
        {

            System.Drawing.Graphics Desen;
            System.Drawing.Pen Creion_Green;
            Desen = CreateGraphics();
            Creion_Green = new System.Drawing.Pen(System.Drawing.Color.Green, 2);
            Desen.Clear(BackColor);
            Desen.DrawLine(Creion_Green, 1, 10, this.Width, 10);

            int k = 10;
            Random rd = new Random();
            int rand_num = rd.Next(3, 10);
            do
            {
                rand_num = rd.Next(3, 10);
                if (rand_num % 5 == 0) { F1(k); }
                else if (rand_num % 5 == 1) { F2(k); }
                else if (rand_num % 5 == 2) { F3(k); }
                else if (rand_num % 5 == 3) { F4(k); }
                else if (rand_num % 5 == 4) { F5(k); }
                k += 150;
            }
            while (k <this.Width);
        }
    }
}
```

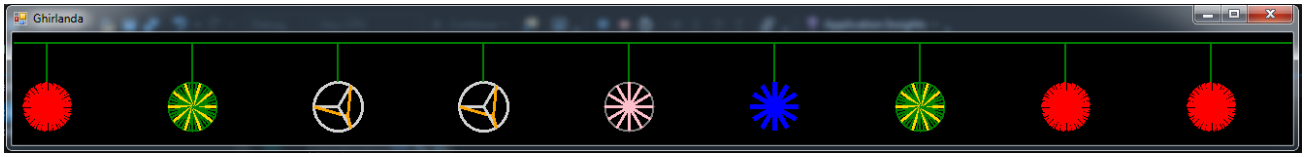
- Explicații:
  Programul e constituit din cinci subprograme, reprezentând fiecare formă în parte, și programul principal, Form1_Paint ce va apela funcțiile la îndeplinirea condițiilor de mai sus. Fiecare funcție deține un set propriu de elemente grafice, acestea formând globul dorit, dar și "ața" de care se prinde pe ghirlandă. Prin modificări la nivelul instrucțiunii repetitive, modelele globurilor variază. Fiecare funcție are nevoie la apelare de un număr întreg, x0, ce va determina distanța dintre forme. În Form1_Paint, distanța e reprezentată de o variabilă k, care se va incrementa cu 150 de pixeli, până la umplerea ferestrei cu podoabe. Globurile vor apărea aleator prin utilizarea unui număr random, rand_num, ce are valori între 3 și 10. În cadrul instrucțiunii do while se va genera un număr aleator și, prin împărțirea sa la numărul de forme existente, în funcție de cifra restului, instrucțiunea decisivă va alege funcția/globul pe care îl va utiliza pentru desen.

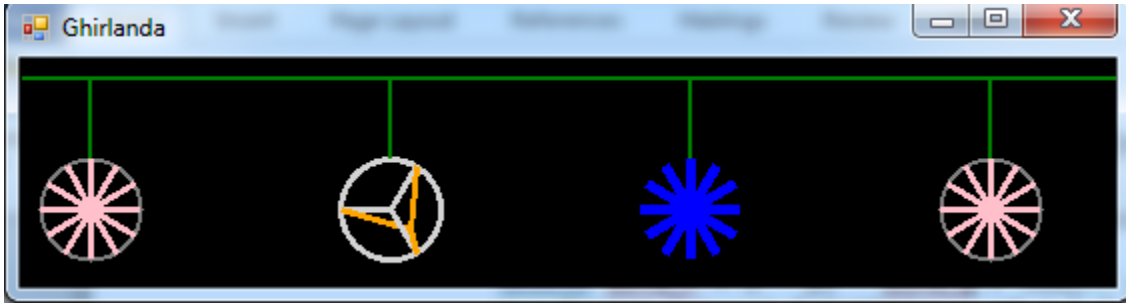
- **<u>Capturi de ecran din timpul rulării programului</u>**

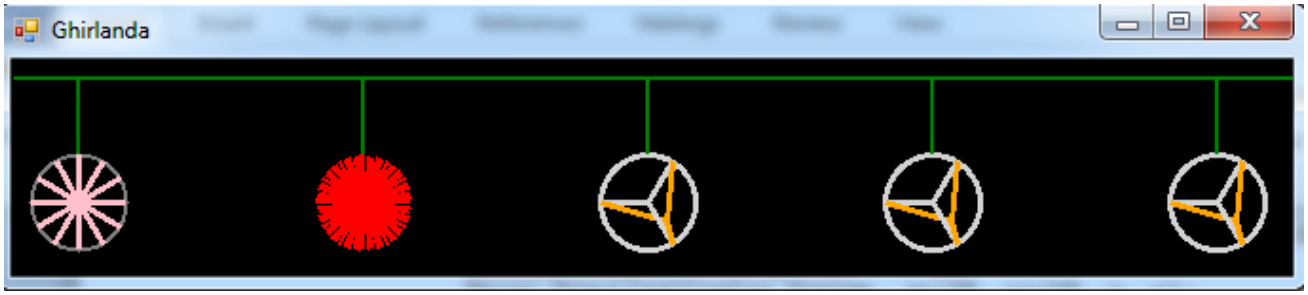
Prima deschidere a programului, generarea primelor forme random



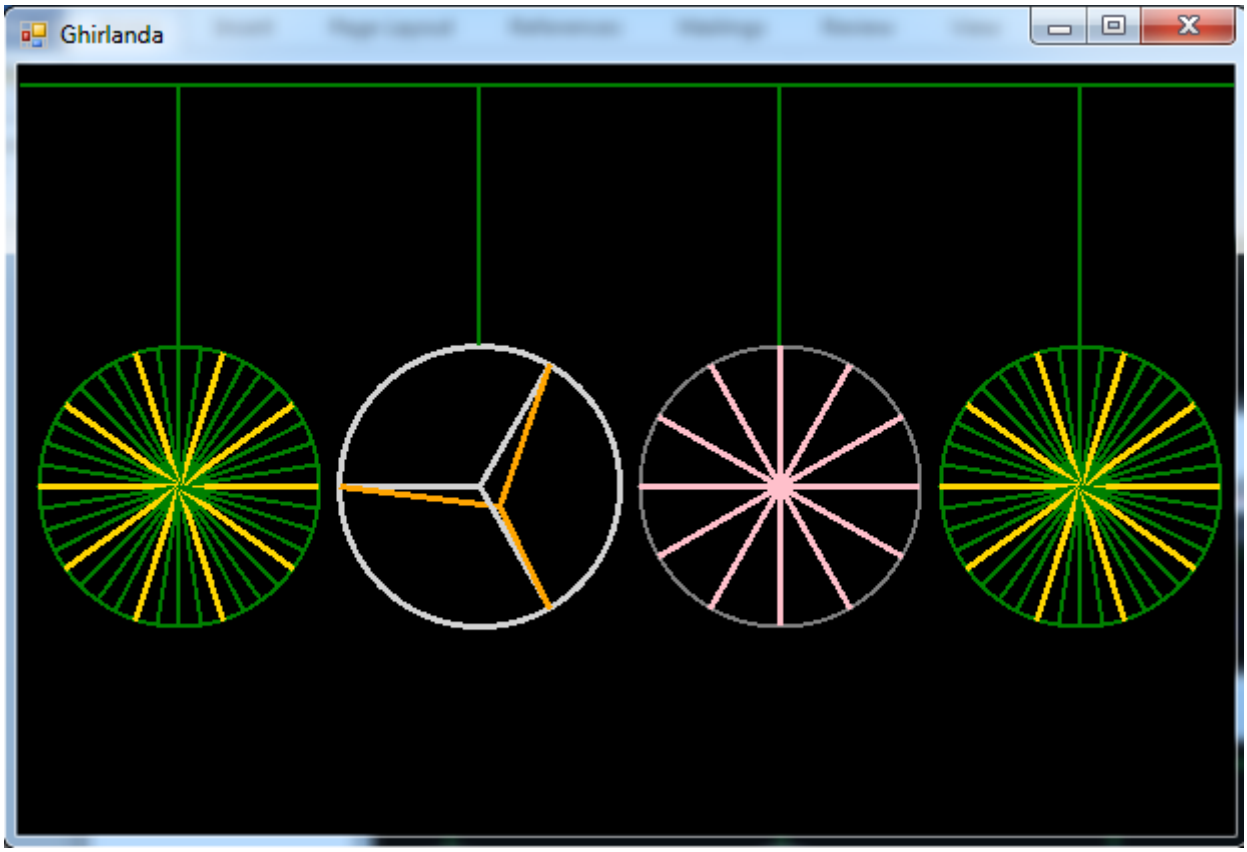
Modificând lățimea ferestrei se generează un nou set de globuri ce umple formul

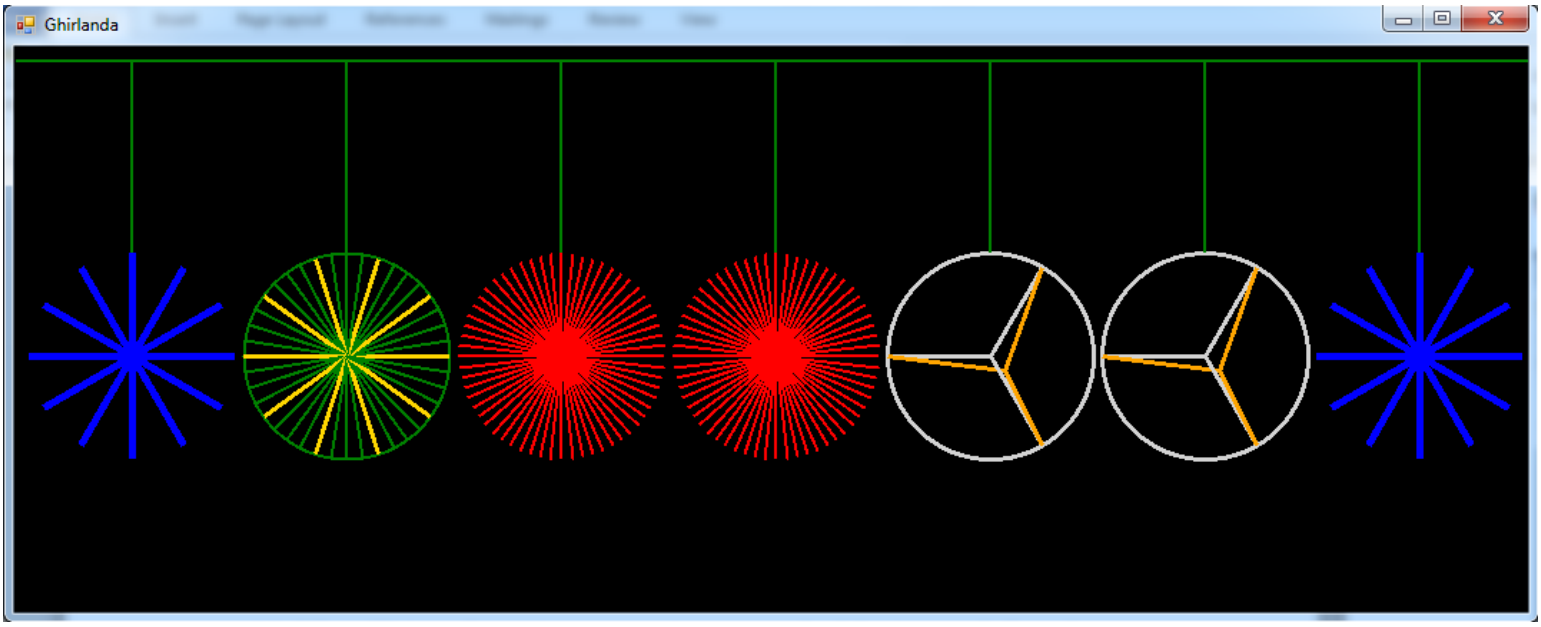






Odată cu modificarea înălțimii ferestrei, globurile se vor redimensiona





- **Codul Sursă al aplicației C++**

```
#include<iostream>
using namespace std;

int main() {
    int n, i, j;
    cin >> n;
    cout << "\n\n";

    for (i = 1; i <= n; ++i) {
        for (j = 1; j <= n; ++j)
            if (i >= j)
                cout << "*";
            else
                cout << " ";

        for (j = 1; j <= n; ++j)
            cout << " ";

        for (j = 1; j <= n; ++j)
            if (j > n - i)
                cout << "*";
            else
                cout << " ";

        cout << "\n";
    }
    for (i = 1; i <= n; ++i) {
        for (j = 1; j <= n; ++j)
            cout << " ";

        for (j = 1; j <= n; ++j)
            cout << "*";

        for (j = 1; j <= n; ++j)
            cout << " ";

        cout << "\n";
    }
    for (i = 1; i <= n; ++i) {
        for (j = 1; j <= n; ++j)
            if (j <= n - i + 1)
                cout << "*";
            else
                cout << " ";

        for (j = 1; j <= n; ++j)
            cout << " ";

        for (j = 1; j <= n; ++j)
            if (i <= j)
                cout << "*";
            else
                cout << " ";

        cout << "\n";
    }
    cin.ignore();
```

```
cin.get();
return 0;
   }

   /*
   Pentru această cerință, am decis să reprezint primul program “complex” realizat în
   liceu, și anume MORIȘCA. Această figură formată din patru triunghiuri dreptunghice cu
   un pătrat în centrul lor reprezenta coșmarul fiecărui mic programator la început de
   drum, mai exact în clasa a 9-a. De dragul vremurilor bune, aduc acest omagiu bietei
   moriști, care la momentul respectiv făcea furori printre elevii necunoscători în
   tainele informaticii.
   În acest program, singura sarcină a utilizatorului este aceea de a tasta un număr ce
   va reprezenta dimensiunea fiecărui element specific formei.

   P.S. Vă rog să mă scuzați pentru paranteza narativă, amintirile au pus stăpânire pe
   mine. o(╥﹏╥)o o(╥﹏╥)o
   */
```

- **Capturi de ecran din timpul rulării programului**