

## חומרים שהוכנו על-ידי משתתפי קורס מורים מובילים תשע"ה

ניתן להשתמש בחומרים לצורך הוראה בלבד.

**לא ניתן לפרסם את החומרים או לעשות בהם כל שימוש מסחרי**

ללא קבלת אישור מראש מצוות הפיתוח

**כתיבה ועריכה:**

**דפנה לוי-רשתי, חיה עידן**



## בית ספר תיכון עירוני מקיף ע"ש גולדווטר

רח' חטיבת הנגב, ת.ד. 595 אילת 88104, טלפון: 08-6372106, פקס: 08-6370181, מזכירות כללית: 08-6372105

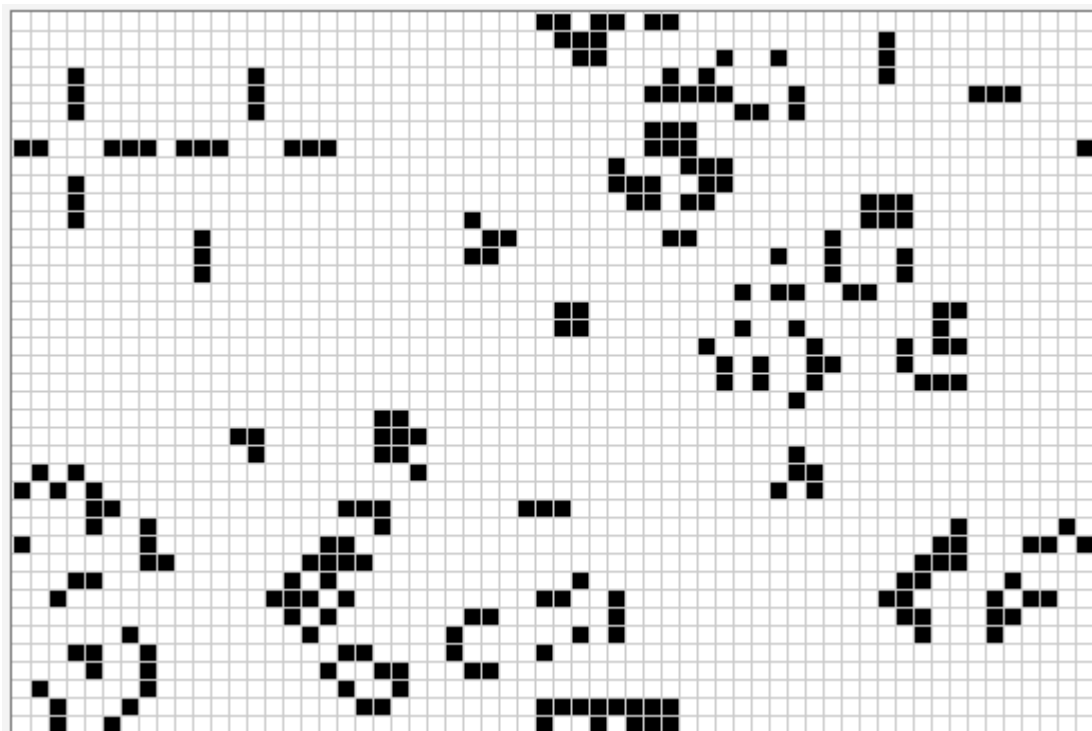


משרד החינוך  
מחוז דרום

מעבדה לסיכום נושא מערך דו מימדי של עצמים

משחק החיים

המעבדה מיועדת לתלמידים אשר סיימו את פרק הלימוד של מערך דו מימדי לפי התוכנית החדשה. ליצירת עניין, קיים שימוש בגרפיקה וב- windowsForm.



עבודה זו עוסקת במשחק החיים, שהוא אחת הגרסאות המפורסמות ביותר לאוטומט תאי דו-מימדי.

### 1. השכלה כללית - למורה

ליבו של המדע המודרני, שתחילתו לפני כ-300 שנים, נעוץ בניסיון לבסס את המודלים המדעיים על משוואות מתמטיות. משוואה היא אכן דבר נוח מאוד כאשר מדובר במערכת פשוטה יחסית, אך כאשר מדובר במערכת מורכבת שהתהליכים בתוכה מצריכים רמה מסוימת של חישוב, הניסיון לתאר את ההתנהגות במספר חישובים קטן, נועד לכישלון.

במקום להסתכל על מערכת ולנסות לזהות את המשוואה המתמטית שמתארת אותה, יש לאתר תוכנית תייצר את התנהגות המערכת. תוכנית כזו מכונה אוטומט תאי (Cellular Automata).

אוטומט תאי הוא מודל הנחקר במסגרת תורת החישוביות, מתמטיקה וביולוגיה תיאורטית. הוא כולל סריג של תאים, שלכל אחד מהם מספר סופי של מצבים. הזמן במודל הוא בדיד, ומצבו של כל תא בזמן נתון  $t$  הוא פונקציה של מצבו ומצב תאים אחרים (שכניו של התא הנתון) בזמן  $t-1$ . פונקציה זו חלה על כל התאים, כלומר כל התאים משתנים על-פי אותה מערכת של כללים. כל הפעלה של הפונקציה על כל התאים בסריג יוצרת דור חדש.

האוטומט התאי מדגים כיצד ממספר מצומצם של חוקים ניתן ליצור תופעות מורכבות ובלתי צפויות וכיצד מכמות מועטה של אינפורמציה ניתן לייצר תהליכי התפתחות ו"חיים" דינאמיים.

אוטומטים ראשוניים מהסוג הזה נבנו מתוך מחשבה לחקות חיים בעזרת תוכנית מחשב, ובהם כל משבצת על המסך סימנה "תא", ולכן הם זכו לשם אוטומטים תאיים.

## 2. משחק החיים

בשנת 1970 המציא המתמטיקאי ג'ון קונווי את האוטומט התאי המפורסם ביותר, משחק החיים. משחק החיים זכה לפופולריות רבה מאוד בקרב מתמטיקאים, מתכנתים וחובבי שעשועי מתמטיקה, במידה רבה בזכות סדרת מאמרים מאת מרטין גרדנר אודותם, שפורסמה בירחון American Scientific. בין השאר התגלו באוטומט זה צורות המראות התנהגויות מורכבות ומעניינות<sup>1</sup>.

התגלה שהאוטומט שקול למכונת טיורינג, כלומר כל חישוב שניתן לבצע על מחשב כלשהו ניתן לביצוע בעזרת משחק החיים, הועלו השערות בקשר לשאלה האם אכן ניתן לייצר חיים, והאם ניתן לייצר יצורים חושבים בעזרת המשחק. השימושים הורחבו גם לנושאים שונים בביולוגיה, פיסיקה, כימיה, כאוס ועוד.

המשחק מתחיל ממצב התחלתי כלשהו, היכול להיקבע על ידי המשתמש. כלל תא ערך בינארי חי או מת. מצב לוח מכונה "דור".

**בכל "דור" נולדים, מתים ושורדים תאים, לפי הכללים הבאים:**

**לידה** – תא מת ייוולד מחדש אם יש לו בדיוק שלושה שכנים חיים.

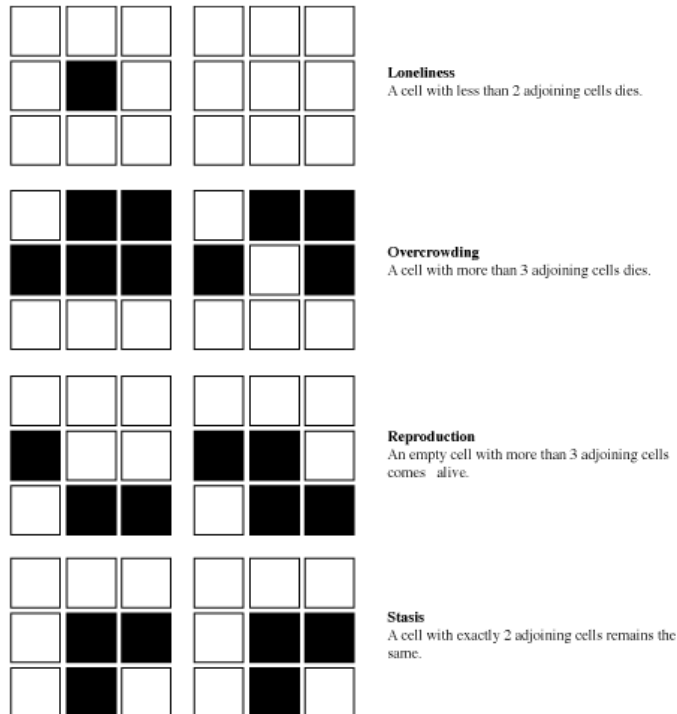
**מוות** – תא חי ימות בשני מקרים:

- אם יש לו שכן חי אחד או פחות – התא ימות מבדידות,

**או**

<sup>1</sup> <http://www.math.com/students/wonders/life/life.html>

- אם יש לו ארבעה שכנים חיים או יותר – התא ימות מצפיפות יתר.
- הישרדות** – תא חי ישרוד אם יש לו שניים או שלושה שכנים חיים. ברור כי תא מת יישאר מת אם לא הייתה לו סיבה להיוולד.



(לתלמיד)

1. מהם העצמים להם אנו זקוקים במשחק? פרט את העצמים ותכונותיהם. תכנן את המחלקות הרלבנטיות.

(למורה)

תכנון מחלקת לוח: CellsGrid

```
public class CellsGrid
{
    private int rowsNum;
    private Cell[,] world;
    private int generationNum;
    ...}

```

בחרנו לייצג את העולם כמטריצה ריבועית של תאים. מכאן נגזרו תכונות המחלקה: גודל המערך, המערך עצמו, ומספר הדור. תאי המטריצה קיבלו מספר סידורי בודד, בנוסף למספר שורה ומספר עמודה, לתרגול עבודה במטריצה ולגישה קלה יותר ללוח הגרפי, לדוגמה בלוח 4\*4:

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

### תכנון מחלקת תא: Cell

לעצם מטיפוס תא, 4 תכונות: מספר שורה, מספר עמודה, מספר סידורי - מטיפוס שלמים, ותכונה בוליאנית, האם התא חי או מת. לבד מפעולות מאחזרות, יש צורך בפעולה קובעת המחיה/ממיתה את התא, ובפעולה חישובית, האם התא יחיה או ימות בדור הבא, על פי מספר שכניו.

```
public class Cell
{
    private int rowIndex;
    private int colIndex;
    private bool isAlive;
    private int serial;
    ...}

```

(לתלמיד)

2. א. כתוב את מחלקת Cell. תשומת ליבך לפעולה הבונה, פעולות מאחזרות וקובעות (האם צריך כאלו לכל התכונות?).
- ב. הוסף פעולה המקבלת את מספר השכנים של התא, ומחשבת האם התא יחיה או ימות בדור הבא. הפעולה תחזיר אמת אם התא יהיה חי, אחרת תחזיר שקר.
3. במחלקת CellsGrid, כתוב את תכונות המחלקה ואת הפעולות הבאות:
  - א. פעולה בונה המקבלת את גודל העולם (אורך הצלע) ובונה עולם דור ראשון, בו מספר התאים החיים הוא בין חמישית מספר התאים בעולם לשליש מספר התאים בעולם. התאים החיים ימוקמו רנדומלית בעולם.
  - ב. פעולה בונה המקבלת את הדור הקודם (עצם מטיפוס CellsGrid) ובונה את הדור הבא, על פי כללי המשחק.
  - ג. פעולה המקבלת תא ומחזירה את מספר שכניו החיים.

(למורה – ניתן לתת לתלמיד ממשק למימוש: )

	פעולה בונה. הפעולה מקבלת את גודל העולם (אורך הצלע) ובונה את הדור הראשון.
	פעולה בונה. הפעולה מקבלת עצם מטיפוס CellsGrid המייצג את הדור הקודם, ובונה עצם המייצג את הדור שאחריו.
	פעולה המחזירה את מספר התאים החיים בדור הראשון. מספר התאים החיים יהיה בין חמישית לשליש מספר התאים בעולם.
	פעולה המפזרת את התאים החיים בדור הראשון באופן רנדומלי במטריצת העולם.
	פעולה המקבלת תא ומחזירה את מספר השכנים החיים שיש לו.
	פעולות קובעות/מאחזרות (האם יש צורך לכל התכונות?)

(לתלמיד)

4. תכנן ובנה את הטופס הראשי - חלון המשחק הראשי. שנה את שם הטופס ל:

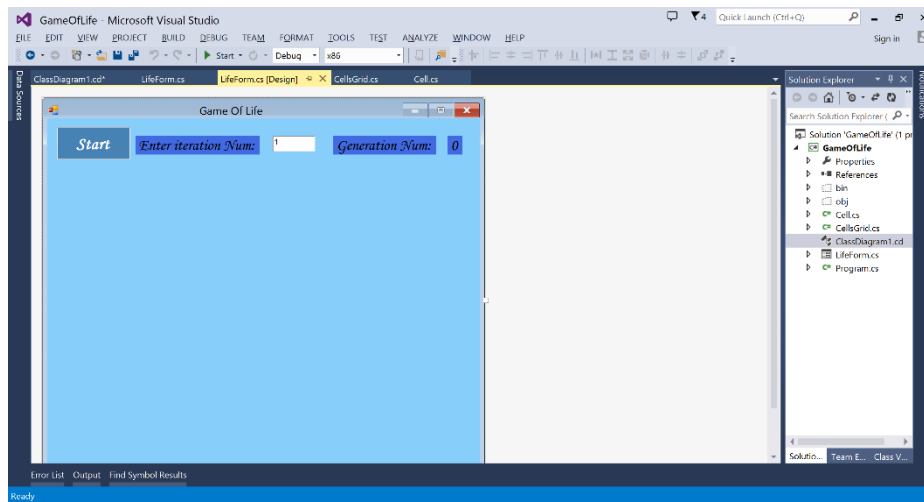
**LifeForm.**

לטופס יש להוסיף (באמצעות גרירה) את הפקדים הבאים:

- התחלת משחק : כפתור התחל – אשר יתחיל את המשחק
- מספר דורות לחישוב קדימה : תיבת טקסט – בה יקליד המשתמש את מספר הדורות הרצוי לו, ותווית אשר תבקש מהמשתמש להכניס את מספר הדורות המבוקש. ערך ברירת המחדל בתיבת הטקסט יהיה 1.
- דור נוכחי : שתי תוויות : האחת תציין כי התווית השניה מציגה את מספר הדור והאחרת מציגה את מספר הדור.

יש להשאיר מקום מספיק לשרטוט העולם : שרטוט זה לא יעשה על ידי גרירה, אלא בבנייה ישירה במחלקת הטופס.

עצב את הטופס והפקדים כרצונך (צבעים, פונטים, כותרות וכו').



(למורה)

אם הכיתה אינה מכירה ה- `WindowsForm` – ניתן להוריד מהרשת את המצגת של זהבה יעקובסון: תכנות חלונאי.

(לתלמיד)

5. בטופס הראשי, יש להוסיף את תמונות התאים. ניתן לבחור בכל פקד מתאים. אנו בחרנו לשרטט כל תא ככפתור, שצבעו נגזר מחייו. אם התא חי – צבע הכפתור יהיה שחור. אם מת – צבע הכפתור יהיה לבן. לא הגדרנו, כמובן, פעולת קליק לכפתורים המסמנים תאים.

תכנות הטופס:

גודל העולם

קואורדינטה X של "תמונת" התא הראשון

קואורדינטה Y של "תמונת" התא הראשון

גודל "תמונת" התא

עצם מטיפוס `CellsGrid` – המייצג את הדור הנוכחי

מערך דו מימדי של פקדים מטיפוס כפתורים, המייצגים את הלוח הגרפי

נתונה הפעולה `BuildBtns()`, אשר בונה את מטריצת הכפתורים המייצגים את התאים:

```
private void BuildBtns()
{
    Cell[,] world = this.cg.GetWorld();
    this.graficWorld = new System.Windows.Forms.Button[rowsNum, rowsNum];
}
```

```

int n = 0;

for (int i = 0; i < rowNum; i++)
{
    for (int j = 0; j < rowNum; j++)
    {
        graficWorld[i, j] = new System.Windows.Forms.Button();
        graficWorld[i, j].Width = picSide; // Width of button
        graficWorld[i, j].Height = picSide; // Height of button
        graficWorld[i, j].Left = xPos + picSide * j;
        graficWorld[i, j].Top = yPos + picSide * i;
        graficWorld[i, j].Tag = n++;

        if (world[i, j].GetIsAlive()) graficWorld[i, j].BackColor = Color.Black;
        else graficWorld[i, j].BackColor = Color.White;

        graficWorld[i, j].Enabled = false;
        graficWorld[i, j].Visible = true;

        this.Controls.Add(graficWorld[i, j]);
    }
}
}

```

- א. כתוב את תכונות הטופס.
- ב. כתוב את הפעולה הבונה של הטופס, המשתמשת בפעולה `BuildBtns()`, לבניית הלוח הגרפי.
- ג. כתוב את הפעולה `CangeBtns()`, המשנה את טקסט תווית מספר הדור לדור הנוכחי, ומשנה את צבע התאים בהתאם לדור הנוכחי.
- ד. הוסף את פעולת הקליק לכפתור ההתחל. בלחיצה על כפתור זה, יחושבו מספר דורות בהתאם לכתוב בתאיבט הטקסט שבטופס. לכל דור חדש, יש לייצר עצם חדש, ולעדכן את הלוח, תוך שימוש בפעולת `CangeBtns()`.  
זכור להשתמש בפעולות:

```

this.Refresh();
System.Threading.Thread.Sleep(2000);

```

- ה. כתוב את הפעולה הבונה של הטופס, המשתמשת בפעולה `BuildBtns()`, לבניית הלוח הגרפי.



6. במחלקת Program, בנה עצם חדש מטיפוס CellsGrid, והעבר אותו כפרמטר בפעולה .Run

התוכנית המלאה:

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

namespace GameOfLife
{
    static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            CellsGrid cg = new CellsGrid(50);
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new LifeForm(cg));
        }
    }
}
//-----

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace GameOfLife
{
    public class Cell
    {
        private int rowIndex;
        private int colIndex;
        private bool isAlive;
        private int serial;

        public Cell(int rowsNum, int row, int col, bool alive)
        {
```

```
        this.rowIndex = row;
        this.colIndex = col;
        this.serial = rowNum * row + col;
        this.isAlive = alive;
    }

    public int GetRow()
    {
        return this.rowIndex;
    }

    public int GetCol()
    {
        return this.colIndex;
    }

    public int GetSerial()
    {
        return this.serial;
    }

    public bool GetIsAlive()
    {
        return this.isAlive;
    }

    public bool WillBeAlive(int neighborsNum)
    {
        if (neighborsNum == 3)
            return true;
        else
            if ((neighborsNum == 2) && this.isAlive)
                return true;
            else
                return false;
    }

    public void SetIsAlive(bool boolVal)
    {
        this.isAlive = boolVal;
    }
}
//-----
```

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace GameOfLife
{
    public class CellsGrid
    {
        private int rowsNum;
        private Cell[,] world;
        private int generationNum;

        public CellsGrid(int rowsNum) // first generation CellsGrid
        {
            this.rowsNum = rowsNum;
            this.generationNum = 0;
            Cell[,] world = new Cell[rowsNum, rowsNum];

            for (int i = 0; i < rowsNum; i++)
            {
                for (int j = 0; j < rowsNum; j++)
                {
                    Cell c = new Cell(rowsNum, i, j, false);
                    world[i, j] = c;
                }
            }
            this.PutBugsRnd(world);
        }

        private int bugsNum()
        {
            int num1 = (rowsNum * rowsNum) / 5;
            int num2 = (rowsNum * rowsNum) / 3;
            Random rnd = new Random();
            return rnd.Next(num1, num2);
        }

        private void PutBugsRnd(Cell[,] cells)
        {
            int bugs;
            bugs = this.bugsNum();
            Random rnd = new Random();

            while (bugs > 0)

```

```

    {
        int i = rnd.Next(0, rowsNum-1);
        int j = rnd.Next(0, rowsNum-1);

        if (!(cells[i, j].GetIsAlive()))
        {
            cells[i, j].SetIsAlive(true);
            bugs--;
        }
    }
    this.world = cells;
}

public CellsGrid(CellsGrid cg) // next generation CellsGrid
{
    this.rowsNum = cg.rowsNum;
    this.generationNum = cg.GetGenerationNum() + 1;
    Cell[,] cells = new Cell[rowsNum, rowsNum];

    for (int i = 0; i < rowsNum; i++)
    {
        for (int j = 0; j < rowsNum; j++)
        {
            Cell oldCell = cg.GetWorld()[i, j];
            int count = cg.neighborsCount(oldCell);

            Cell c = new Cell(rowsNum, i, j, oldCell.WillBeAlive(count));
            cells[i, j] = c;
        }
    }
    this.world = cells;
}

public int GetRowsNum()
{
    return this.rowsNum;
}

public void SetRowsNum(int r)
{
    this.rowsNum = r;
}

public int GetGenerationNum()
{
    return this.generationNum;
}

public void SetGenerationNum(int gn)
{

```

```

        this.generationNum = gn;
    }

    public Cell[,] GetWorld()
    {
        return this.world;
    }

    public int neighborsCount(Cell c)
    {
        int count = 0;
        int row = c.GetRow();
        int col = c.GetCol();

        for (int i = row - 1; i <= row + 1; i++)
        {
            for (int j = col - 1; j <= col + 1; j++)
            {
                if ((i >= 0) && (i < this.rowsNum) &&
                    (j >= 0) && (j < this.rowsNum) &&
                    (!((i == row) && (j == col))))
                {
                    if (this.world[i, j].GetIsAlive()) count++;
                }
            }
        }
        return count;
    }
}
//-----

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 GameOfLife
{
    public partial class LifeForm : Form
    {
        int rowsNum;
        int xPos = 20;
        int yPos = 70;
    }
}

```

```

int picSide = 15;
CellsGrid cg;
Button[,] graphicWorld;

public LifeForm()
{
    InitializeComponent();
}

public LifeForm(CellsGrid cg)
{
    this.rowsNum = cg.GetRowsNum();
    this.cg = cg;
    BuildBtns();
    InitializeComponent();
}

//The following procedure to create matrix of buttons:
private void BuildBtns()
{
    Cell[,] world = this.cg.GetWorld();
    this.graphicWorld = new System.Windows.Forms.Button[rowsNum, rowsNum];
    int n = 0;

    for (int i = 0; i < rowsNum; i++)
    {
        for (int j = 0; j < rowsNum; j++)
        {
            // Initialize one variable
            graphicWorld[i, j] = new System.Windows.Forms.Button();
            graphicWorld[i, j].Width = picSide; // Width of button
            graphicWorld[i, j].Height = picSide; // Height of button
            graphicWorld[i, j].Left = xPos + picSide * j;
            graphicWorld[i, j].Top = yPos + picSide * i;
            graphicWorld[i, j].Tag = n++;

            if (world[i, j].GetIsAlive())
                graphicWorld[i, j].BackColor = Color.Black;
            else
                graphicWorld[i, j].BackColor = Color.White;

            graphicWorld[i, j].Enabled = false;
            graphicWorld[i, j].Visible = true;

            this.Controls.Add(graphicWorld[i, j]);
        }
    }
}

```

```
private void ChangeBtns()
{
    this.numLbl.Text = cg.GetGenerationNum().ToString();
    this.Refresh();
    for (int i = 0; i < rowsNum; i++)
    {
        for (int j = 0; j < rowsNum; j++)
        {
            if (this.cg.GetWorld()[i,j].GetIsAlive())
                graphicWorld[i, j].BackColor = Color.Black;
            else
                graphicWorld[i, j].BackColor = Color.White;

            graphicWorld[i, j].Visible = true;
        }
    }
}

private void button1_Click(object sender, EventArgs e)
{
    int iters = Int32.Parse(this.iterTb.Text);
    for (int i = 0; i < iters; i++)
    {
        CellsGrid newCg = new CellsGrid(cg);
        this.cg = newCg;
        this.ChangeBtns();
        this.Refresh();
        System.Threading.Thread.Sleep(2000);
    }
    if (button1.Text == "Start")
        button1.Text = "Next";
}
}
```