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

    int BoardSize;
    int TriedMoves;

    private void btnGo_Click(object sender, EventArgs e)
    {
      try
      {
        BoardSize = int.Parse(txtBoardSize.Text);
      }
      catch
      {
        BoardSize = 8;
        txtBoardSize.Text = BoardSize.ToString();
      }

      int[,] Board = new int[BoardSize, BoardSize];

      Position InitialPosition;
      InitialPosition.row = 0;
      InitialPosition.col = 0;

      int DoneMoves = 0;
      TriedMoves = 0;

      Position CurrentPosition = InitialPosition;
      MarkMove(Board, CurrentPosition, ref DoneMoves);

      DateTime dtStart = DateTime.Now;
      MoveHorse(Board, CurrentPosition, DoneMoves);
      DateTime dtFinish = DateTime.Now;

      ShowBoard(Board);

      MessageBox.Show("Elapsed time: " + dtFinish.Subtract(dtStart).ToString());
      //MessageBox.Show("Tried moves: " + TriedMoves.ToString());
    }

    private void ShowBoard(int[,] Board)
    {
      this.SuspendLayout();
      for (int r = 0; r < BoardSize; r++)
      {
        for (int c = 0; c < BoardSize; c++)
        {
          string TextBoxName = "txt" + r.ToString() + "_" + c.ToString();
          TextBox t = (TextBox)this.Controls[TextBoxName];
          if (t == null)
          {
            t = new TextBox();
            t.Name = TextBoxName;
            t.Size = new Size(26, 22);
            t.Location = new Point(10 + c * 26, 50 + r * 22);
            t.TextAlign = HorizontalAlignment.Center;
            this.Controls.Add(t);
          }
          if (Board[r, c] != 0)
          {
            t.Text = String.Format("{0:D2}  ", Board[r, c]);
            t.BackColor = Color.LightGreen;
          }
          else
          {
            t.Text = "";
            t.BackColor = Color.White;
          }

        }
      }
      this.ResumeLayout();
      Application.DoEvents();
    }

    private void MarkMove(int[,] Board, Position MovePosition, ref int DoneMoves)
    {
      DoneMoves++;
      Board[MovePosition.row, MovePosition.col] = DoneMoves;

      //ShowBoard(Board);
      //System.Threading.Thread.Sleep(2000);
    }

    private void UnmarkMove(int[,] Board, Position MovePosition, ref int DoneMoves)
    {
      Board[MovePosition.row, MovePosition.col] = 0;
      DoneMoves--;
    }

    private bool MoveHorse(int[,] Board, Position CurrentPosition, int DoneMoves)
    {
      if (DoneMoves == BoardSize * BoardSize)
        return true;
      int[,] Offset = new int[8, 2] { { 2, 1 }, { 1, 2 }, { -1, 2 }, { -2, 1 }, { -2, -1 }, { -1, -2 }, { 1, ↙
   -2 }, { 2, -1 } };

      // con questa sequenza di mosse ci mette circa 2 minuti
      //int[,] Offset = new int[8, 2] { { 1, 2 }, { 2, 1 }, { 2, -1 }, { 1, -2 }, { -1, -2 }, { -2, -1 }, {  ↙
  -2, -1 }, { -1, 2 } };

      for (int i = 0; i < 8; i++)
      {
        Position NextPosition = CurrentPosition;
        NextPosition.col += Offset[i, 0];
        NextPosition.row += Offset[i, 1];
        if (IsValid(Board, NextPosition))
        {
          MarkMove(Board, NextPosition, ref DoneMoves);

          if (MoveHorse(Board, NextPosition, DoneMoves))
            return true;
          UnmarkMove(Board, NextPosition, ref DoneMoves);
          TriedMoves++;   // Conta tutte le mosse valide
        }
        //TriedMoves++;   // Conta tutte le mosse (valide e non)
      }
      if (TriedMoves % 10000 == 0)
        ShowBoard(Board);
      return false;
    }

    private bool IsValid(int[,] Board, Position ProposedPosition)
    {
      if (ProposedPosition.row < 0 || ProposedPosition.row >= BoardSize || ProposedPosition.col < 0 ||       ↙
  ProposedPosition.col >= BoardSize)
        return false;
      if (Board[ProposedPosition.row, ProposedPosition.col] != 0)
        return false;
      return true;
    }
    struct Position
    {
      public int row;
      public int col;
    }

  }
}
```