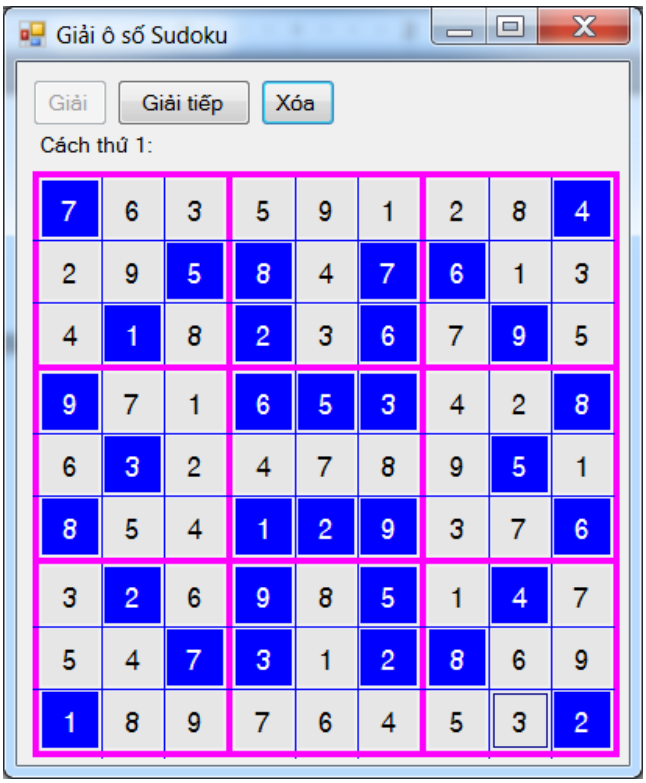
MÔN : LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG
Bài thực hành số 7.2 : Xây dựng chương trình giải ô số Sudoku

I. Mục tiêu :

- Giúp SV làm quen với qui trình thiết kế trực quan 1 ứng dụng Dialog Based.
- Giúp SV làm quen với việc dùng lại linh kiện phần mềm.
- Giúp SV thấy cụ thể cấu trúc ứng dụng cấu thành từ các đối tượng.
- Giúp SV thấy sự tương tác giữa các đối tượng.

II. Nội dung :

- Xây dựng chương trình nhỏ chạy ở chế độ đồ họa cho phép user nhập các giá trị vào 1 số ô Sudoku rồi giải. Cửa sổ ứng dụng có dạng sau :



III. Chuẩn đầu ra :

- Thành thạo việc xây dựng 1 ứng dụng theo qui trình thiết kế trực quan.
- Thành thạo việc dùng lại linh kiện phần mềm có sẵn, thấy rõ cấu trúc phầm mềm và sự tương tác giữa các đối tượng trong phần mềm.
- Thành thạo việc viết code thay đổi kích thước và vị trí các đối tượng giao diện khi cửa sổ chứa chúng bị thay đổi.

IV. Phân tích :

Chương trình chứa 1 đối tượng Form, Form chứa các đối tượng bên trong :
- 3 đối tượng Button
- 1 đối tượng Label
- 1 đối tượng ma trận Sudoku

Ta có thể thiết kế trực quan Form với các đối tượng co sẵn như Button, Label cho dễ. Riêng ma trận Sudoku sẽ được lập trình động theo trạng thái từng thời điểm của nó.

V. Qui trình :

1. Chạy VS .Net, chọn menu File.New.Project để hiển thị cửa sổ New Project.

2. Mở rộng mục Visual C# trong TreeView "Project Types", chọn mục Window, chọn icon "Windows Application" trong listbox "Templates" bên phải, thiết lập thư mục chứa Project trong listbox "Location", nhập tên Project vào textbox "Name:" (thí dụ Form_Sudoku), click button OK để tạo Project theo các thông số đã khai báo.
3. Form đầu tiên của ứng dụng đã hiển thị trong cửa sổ thiết kế, việc thiết kế form là quá trình lặp 4 thao tác tạo mới/xóa/hiệu chỉnh thuộc tính/tạo hàm xử lý sự kiện cho từng đối tượng cần dùng trong form.
4. Nếu cửa sổ ToolBox chưa hiển thị chi tiết, chọn menu View.Toolbox để hiển thị nó (thường nằm ở bên trái màn hình). Click chuột vào button  (Auto Hide) nằm ở góc trên phải cửa sổ ToolBox để chuyển nó về chế độ hiển thị thường trực.
5. Duyệt tìm phần tử Button (trong nhóm Common Controls), chọn nó, dời chuột về góc trên trái của form và vẽ nó với kích thước mong muốn. Xem cửa sổ thuộc tính của Button vừa vẽ (thường ở góc dưới phải màn hình), duyệt tìm và hiệu chỉnh thuộc tính Text = "Giải", duyệt tìm và thay đổi thuộc tính (Name) = btnGiai.
6. Lặp lại bước 5 hai lần nữa để vẽ 2 button ("Giải tiếp", btnGiaitiep) và ("Xóa", btnXoa).
7. Duyệt tìm phần tử Label (trong nhóm Common Controls), chọn nó, dời chuột về vị trí ngay dưới các button và vẽ nó với kích thước mong muốn. Hiệu chỉnh thuộc tính (Name) = lblMesg.
8. Dời chuột về button btnGiai, ấn kép chuột vào nó để tạo hàm xử lý sự kiện Click chuột cho button, cửa sổ mã nguồn sẽ hiển thị để ta bắt đầu viết code cho hàm. Cách tổng quát để tạo hàm xử lý sự kiện là chọn đối tượng btnGiai, cửa sổ thuộc tính của nó sẽ hiển thị, click icon  để hiển thị danh sách các sự kiện của đối tượng, duyệt tìm sự kiện quan tâm (Click), ấn kép chuột vào comboBox bên phải sự kiện Click để máy tạo tự động hàm xử lý cho sự kiện này. Cửa sổ mã nguồn sẽ hiển thị khung sườn của hàm vừa được tạo với thân rỗng, ta viết thân cho hàm này sau.
9. Dời chuột về cửa sổ "Solution Explorer" (thường nằm ở phía trên phải màn hình), duyệt tìm phần tử Form1.cs, ấn phải chuột trên nó để hiển thị menu lệnh, chọn lệnh "View Designer" để hiển thị lại cửa số thiết kế của Form. Thực hiện lại bước 7 trên button btnGiaitiep để tạo hàm xử lý sự kiện Click chuột cho button này.
10. Tiếp tục tạo hàm xử lý sự kiện Click chuột cho button btnXoa.
11. Dời chuột về cửa sổ "Solution Explorer", duyệt tìm phần tử Form1.cs, ấn phải chuột trên nó để hiển thị menu lệnh, chọn lệnh "View Designer" để hiển thị lại cửa số thiết kế của Form. Chọn Form này rồi lần lượt tạo hàm xử lý sự kiện KeyDown, MouseDown, Paint cho Form.
12. Dời chuột về đầu class Form1 rồi viết tiếp đoạn code sau phục vụ việc giải ô số Sudoku :

```
//=================================================
// phần code độc lập với giao diện
//=================================================
    //định nghĩa kiểu miêu tả thông tin về 1 ô số
    struct Cell
    {
       public bool fix;   //fix = true : giá trị ô cố định
       public sbyte value;
    };
    //định nghĩa các thuộc tính dữ liệu
    const int LEN = 9;
    Cell[,] matran = new Cell[LEN, LEN];
    int h, c;  //tọa độ ô đang xử lý
```

```
int cachso;

//hàm kiểm tra ô (h,c) có thể chứa giá trị val không ?
bool Testvitri(int h, int c, sbyte val)
{
   int h1, c1;
   int i, j;
   matran[h, c].value = 0;
   if (val > 9) return false;
   // xem có trùng với cell nào ở hàng h ?
   for (c1 = 0; c1 < LEN; c1++)
      if (matran[h, c1].value == val) return false;
   // xem có trùng với cell nào ở cot c ?
   for (h1 = 0; h1 < LEN; h1++)
      if (matran[h1, c].value == val) return false;
   // xem có trùng với cell trong vùng 3x3 ?
   h1 = h / 3 * 3;
   c1 = c / 3 * 3;
   for (i = h1; i < h1 + 3; i++)
      for (j = c1; j < c1 + 3; j++)
         if (matran[i, j].value == val) return false;
   //chứa kết quả vào cell tương ứng
   matran[h, c].value = val;
   return true;
}

//hàm tìm trị phù hợp cho ô h,c ?
//trả về TRUE nếu được, FALSE nếu không
bool timtri(int h, int c)
{
   sbyte val;
   sbyte d;
   if (matran[h,c].fix) return true;
   val = matran[h, c].value; val++;
   matran[h,c].value = 0;
   for (d = val; d <= 9; d++) if (Testvitri(h, c, d)) return true;
   return false;
}

//hàm lùi về ô ngay trước ô h,c cần xừ lý tiếp
//trả về TRUE nếu lùi được, FALSE nếu không lùi được
bool Back(ref int h, ref int c)
{
   while (true)
   {
      if (c > 0) c--;
      else
      {
         c = LEN - 1; h--;
      }
      if (h < 0) return false; //hết cách
      if (matran[h, c].fix == false) return true;
```

```
        }
    }


//=============================================================
// phần code phục vụ giao diện với người dùng
//=============================================================
//Định nghĩa các hằng cần dùng
    int yStart = 70;    //top của bảng Sukodu
    int xStart = 10;    //left của bảng Sukodu
    int wSeparator = 4; //khoảng hở giữa các vùng
    const int WCELL = 40;   //độ rộng từng ô
    const int HCELL = 40;   //độ cao từng ô
    //tạo pen với màu Blue, nét vẽ 2 pixel
    Pen pen = new Pen(Color.FromArgb(255, 0, 255), 1);
    //tạo brush với màu đỏ, tô đặc
    Brush brush = new SolidBrush(Color.FromArgb(255, 0, 255));
    Graphics g;
    int xcur, ycur; //ô nhập liệu

    //hàm hiển thị nội dung ô row,col
    private void OutXY(int row, int col)
    {
        //tạo đối tượng font chữ cần dùng
        Font myFont = new Font("Helvetica", 11);
        //tạo biến miêu tả chế độ canh giữa khi xuất chuỗi
        StringFormat format1 = new StringFormat(StringFormatFlags.NoClip);
        format1.Alignment = StringAlignment.Center;
        //tính tọa độ x,y trên form của ô row,col
        int x, y;
        x = xStart + col * WCELL + (col / 3 + 1) * wSeparator;
        y = yStart + row * HCELL + (row / 3 + 1) * wSeparator;
        //thiết lập màu nền và màu chữ cho ô
        Color bColor, fColor;
        if (matran[row,col].fix)
        {
            bColor = Color.FromArgb(0, 0, 255);
            fColor = Color.FromArgb(255, 255, 255);
        }
        else
        {
            bColor = Color.FromArgb(230, 230, 230);
            fColor = Color.FromArgb(0, 0, 0);
        }
        pen.Color = fColor;
        brush = new SolidBrush(bColor);
        //tô nền cho ô
        g.FillRectangle(brush, x + 2, y + 2, WCELL - 4, HCELL - 4);
        if (matran[row,col].value > 0) // hiển thị ô hợp lệ
            g.DrawString(matran[row,col].value.ToString(), myFont, new SolidBrush(fColor), x +
WCELL / 2, y + 8, format1);
        if (matran[row,col].value == -1) //hiển thị ô không hợp lệ
```

```
        {
            g.DrawString("?", myFont, System.Drawing.Brushes.Red,x + WCELL / 2, y + 8,
format1);
        }
        if (row == ycur && col == xcur)
        {
            //vẽ cursor nhập liệu ở ô hiện hành
            pen.Color = Color.DarkBlue;
            g.DrawRectangle(pen,x + 3, y + 3, WCELL - 6, HCELL - 6);
        }
    }

    private void Giai() {
        //bắt đầu tìm trị cho ô trên trái
        cachso = 0;
        h = 0; c = 0;
        //cố gắng tìm 1 cách sắp xếp các ô số
        Tim1cach();
        btnGiai.Enabled = false;
        btnGiaitiep.Enabled = true;
    }

    //hàm cố gắng tìm 1 cách sắp xếp các ô số
    void Tim1cach()
    {
        while (h<LEN) { //cần tìm trị cho ô h,c
            //tìm trị cho ô h,c
            if (timtri(h,c)) { //nếu tìm được
                //tiếp tục ô kế tiếp
                if (++c == LEN) { c = 0; ++h; }
                continue;
            }
            //nếu tìm không được trị cho ô h,c
            matran[h, c].value = 0;
            if (Back(ref h, ref c)) continue;
            //hết cách --> dừng chương trình
            lblMesg.Text = "Hết cách rồi";
            return;
        }
        // tìm được cách sắp toàn bộ các ô số
        cachso++;
        lblMesg.Text = "Cách thứ " + cachso + ":";
        this.Invalidate();
        return;
    }

    //hàm tìm cách giải kế tiếp
    private void Giaitiep()
    {
        if (!Back(ref h, ref c))
            //hết cách
            lblMesg.Text = "Không còn cách nào khác nữa.";
```

```
        else
        {
          Tim1cach();
        }
    }

    //hàm reset ma trận Sudoku về trạng thái ban đầu
    private void XoaSudoku()
    {
        int h, c;
        //lặp reset từng cell
        for (h = 0; h < LEN; h++)
            for (c = 0; c < LEN; c++) {
                matran[h, c].fix = false;
                matran[h, c].value = 0;
            }
        lblMesg.Text = "Hãy nhập số vào các ô :";
        btnGiaitiep.Enabled = false;
        btnGiai.Enabled = true;
        Invalidate();
    }
```

13. Viết code cho các hàm xử lý sự kiện vừa tạo ra trong các bước trước như sau (lưu ý chỉ copy thân hàm và dán vào khung sườn của hàm đã được máy tạo sẵn) :

```
    //hàm khởi tạo Form
    public Form1()
    {
        InitializeComponent();
        int h, c;
        //phân phối vùng nhớ cho ma trận Sukodu
        for (h = 0; h < LEN; h++)
            for (c = 0; c < LEN; c++)
                matran[h, c] = new Cell();
        //hiệu chỉnh lại kích thước Form để chứa vừa ma trận TextBox
        this.Size = new Size(9 * WCELL + xStart * 2 + 14 + 4 * wSeparator, 9 * HCELL + yStart
+ 10 + 40 + 4 * wSeparator);
        //thiết lập ô chứa cursor nhập liệu
        xcur = ycur = LEN / 2;
        //cho phép Form xử lý sự kiện phím
        this.KeyPreview = true;
        XoaSudoku();
    }

    //hàm xử lý Click chuột trên button Giai
    private void btnGiai_Click(object sender, EventArgs e)
    {
        Giai();
    }

    //hàm xử lý Click trên button Giaitiep
    private void btnGiaitiep_Click(object sender, EventArgs e)
    {
        Giaitiep();
```

```
}


//hàm xử lý Click trên button Xoa
private void btnXoa_Click(object sender, EventArgs e)
{
   XoaSudoku();
}

//hàm xử lý Click chuột trên Form
private void Form1_MouseDown(object sender, MouseEventArgs e)
{
   //xác định tọa độ chuột
   int x = e.X, y =e.Y;
   //kiểm tra chuột có nằm trên ma trận Sudoku ?
   if (xStart > x || x >xStart + LEN * WCELL + 4*wSeparator ||
      yStart > y || y > yStart+ LEN * HCELL + 4*wSeparator) {
      //nếu nằm ngoài ma trận thì không làm gì
      return;
   }
   //xác định ô được focus
   xcur = (x - xStart) /WCELL;
   if (xcur == LEN) xcur--;
   ycur = (y - yStart) /HCELL;
   if (ycur == LEN) ycur--;
   //vẽ lại Form
   this.Invalidate();
   return;
}

//hàm xử lý ấn phím trên Form
private void Form1_KeyDown(object sender, KeyEventArgs e)
{
   if (Keys.D0<= e.KeyCode && e.KeyCode <=Keys.D9) { //các phím số 0-9
      sbyte d = (sbyte)(e.KeyCode -Keys.D0);
      if (d == 0) { //phím xóa ô
         matran[ycur, xcur].fix = false;
         matran[ycur, xcur].value = d;
      }
      else if (Testvitri(ycur, xcur, d)) {//hợp lệ
         matran[ycur,xcur].fix = true;
         matran[ycur,xcur].value = d;
      } else { //không hợp lệ
         matran[ycur,xcur].value = -1;
         matran[ycur,xcur].fix = false;
         lblMesg.Text = "Hãy sửa giá trị ô màu đỏ vì bị lỗi.";
      }
   }
   //kiểm tra các phím điều khiển
   switch (e.KeyCode) {
      case Keys.Up:
         if (ycur==0) return;
```

```
            ycur--; break;
        case Keys.Down:
            if (ycur==LEN) return;
            ycur++; break;
        case Keys.Left:
            if (xcur==0) return;
            xcur--; break;
        case Keys.Right:
            if (xcur==LEN) return;
            xcur++; break;
        case Keys.Enter:
            Giai(); break;
        case Keys.Delete:
            XoaSudoku(); break;
        case Keys.Space:
            Giaitiep(); break;
        case Keys.Escape:
            this.Close(); break;
        default:
            break;
    }
    Invalidate();
}

//hàm vẽ lại Form
private void Form1_Paint(object sender, PaintEventArgs e)
{
    int x1,y1,x2,y2;
    int row, col;
    g = e.Graphics;
    //thiết lập màu vẽ
    for (row = 0; row < LEN; row++)
        for (col = 0; col < LEN; col++) OutXY(row, col);

    //thiết lập màu Magenta cho pen
    pen.Color = Color.FromArgb(255, 0, 255);
    //tạo brush với màu Magenta, tô đặc
    brush = new SolidBrush(Color.FromArgb(255, 0, 255));
    //vẽ các đường lưới dọc phân ô và phân vùng ma trận Sudoku
    y1 = yStart;
    y2= yStart+ LEN * HCELL + 4*wSeparator;
    for (col=0;col<=LEN;col++) {
        //xác định tọa độ x của đường lưới
        x1 = x2 = xStart + col*WCELL+ (col/3+1)*wSeparator;
        if ((col-col/3*3)==0)  //lưới phân vùng
            g.FillRectangle(brush, x1 - wSeparator, y1, wSeparator, y2 - yStart);
        else { //lưới phân ô
            pen.Color = Color.FromArgb(0, 0, 255);
            g.DrawLine(pen, x1, y1, x2, y2);
        }
    }
    //vẽ các đường lưới ngang phân ô và phân vùng ma trận Sudoku
```

```
        x1 = xStart;
        x2 = xStart + LEN * WCELL + 3 * wSeparator;
        for (row=0;row<=LEN;row++) {
          //xác định tọa độ y của đường lưới
          y1 = y2 = yStart + row * HCELL + (row / 3 + 1) * wSeparator;
          if ((row - row / 3 * 3) == 0) //lưới phân vùng
             g.FillRectangle(brush, x1, y1 - wSeparator, x2 - xStart, wSeparator);
          else { //lưới phân ô
           pen.Color = Color.FromArgb(0, 0, 255);
           g.DrawLine(pen, x1, y1, x2, y2);
          }
        }
      }
```

14. Chọn menu Debug.Start Debugging để dịch và chạy thử ứng dụng.
15. Khi Form chương trình hiển thị, hãy click chuột vào từng ô cần nhập số rồi gỏ phím số cần nhập. Sau khi đã nhập xong các ô số của bài toán, bạn click button Giai và xem đáp án có đúng với yêu cầu. Nếu muốnn xem đáp án khác, bạn ấn button "Giải tiếp", máy sẽ cố gắng tìm phương án khác, nếu có nó hiển thị lên, nếu không nó báo hết cách.
16. Muốn giải ô số Sudoku khác, bạn click button "Xóa" để máy khởi động lại từ đầu. Hãy thực hiện lại bước 15.