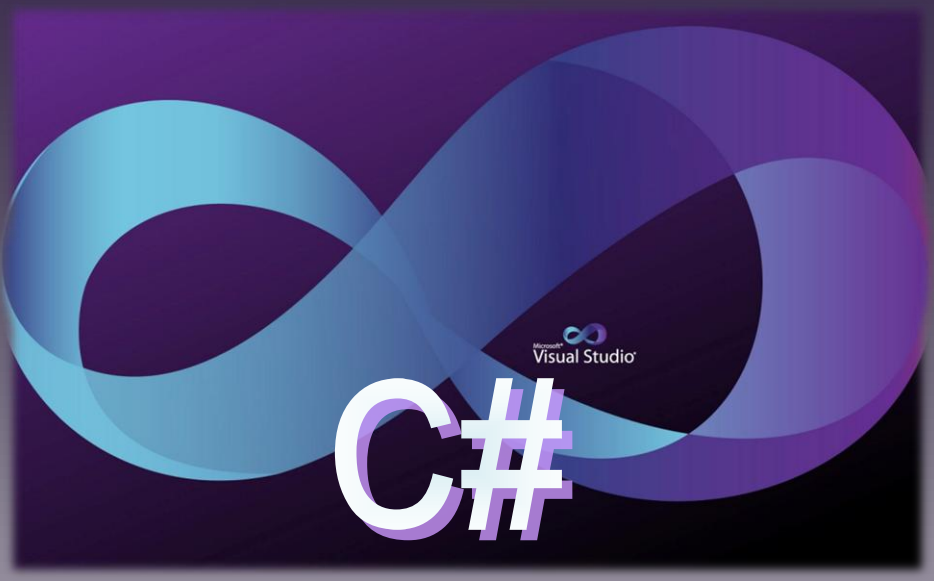
ISTITUTO TECNICO INDUSTRIALE
G. M. ANGIOY
SASSARI



CORSO DI PROGRAMMAZIONE

USO DEL DATAGRIDVIEW PER VISUALIZZARE ARRAY

DISPENSA 05.05

05-05_DataGridView[ver_15]

Versione del: 07/11/2015
Revisione numero: 15

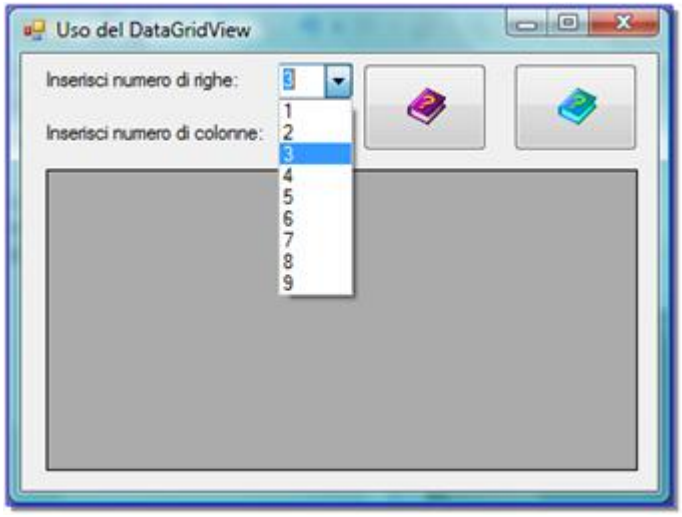
Prof. Andrea Zoccheddu
Dipartimento di Informatica

DIPARTIMENTO
INFORMATICA E TELECOMUNICAZIONI

DATAGRIDVIEW

INTRODUZIONE AL CONTROLLO DATAGRIDVIEW

PROGETTO GUIDATO



- Si prepari un Form1 simile al seguente:
- Ci sono due ComboBox con degli elenchi da 1 a 9; puoi usare la proprietà `Items` per modificare gli elenchi; imposta la proprietà DropDownStyle a `DropDownList`
- Ci sono due pulsanti; puoi usare la proprietà Image se vuoi modificarli simili alla figura
- Imposta le seguenti proprietà del DataGridView (il controllo si trova nella paletta degli strumenti, nella categoria Dati):
- AutoSizeColumnsMode → `AllCells`
- AutoSizeRowsMode → `AllCells`
- RowHeadersVisible → `False`
- ColumnsHeadersVisible → `False`
- Preparare le variabili globali seguenti

VC#

```
int Nrighe, Ncolonne;
int[,] matrix;
```

- Doppio clic su `button1` e associarvi:

VC#

```
//acquisisce numero righe matrice
Nrighe = 1 + comboBox1.SelectedIndex;
//acquisisce numero sue colonne
Ncolonne = 1 + comboBox2.SelectedIndex;
if (Nrighe > 0 && Ncolonne > 0)
{
      //dimensiona la matrice
      matrix = new int[Nrighe, Ncolonne];
      Random dado = new Random();
      for (int r = 0; r < Nrighe; r++)
            for (int c = 0; c < Ncolonne; c++)
                  //inizializza ogni cella di matrix
                  matrix[r, c] = dado.Next(10, 100);
```

```
      //predispone aspetto del dataGridView1
      dataGridView1.RowCount = Nrighe;
      //predispone aspetto del dataGridView1
      dataGridView1.ColumnCount = Ncolonne;
      for (int r = 0; r < Nrighe; r++)
            for (int c = 0; c < Ncolonne; c++)
                  //copia i valori nel controllo
                  dataGridView1.Rows[r].Cells[c].Value = matrix[r, c];
      //elimina la selezione da qualsiasi cella
      dataGridView1.ClearSelection();
}
else
      //messaggio di errore
      MessageBox.Show("devi selezionare righe e colonne!");
```

- Doppio clic su `button2` e associare il seguente codice:

VC#

```
dataGridView1.RowHeadersVisible = !dataGridView1.RowHeadersVisible;
dataGridView1.ColumnHeadersVisible = !dataGridView1.ColumnHeadersVisible;
```

- Prova il progetto

IL CONTROLLO DATAGRIDVIEW

SCOPO DEL CONTROLLO

Il DataGridView è un controllo visuale che serve per mostrare dati in forma tabellare. Il controllo quindi dispone di proprietà e metodi per gestire righe, colonne e celle. Il controllo è progettato per ricevere dati anche da altre fonti, come i database (vedi MS Access ovvero il modulo 5 ECDL base), ma al momento si studierà solamente, ed in modo superficiale, l'uso del controllo per mostrare dati delle matrici e dei vettori.

PROPRIETÀ E METODI DEL CONTROLLO

Il controllo dispone di molte proprietà e metodi, ma ne vedremo solo alcuni:

Proprietà	Valore	Scopo
ColumnCount	int	Indica il numero di colonne da mostrare
RowCount	int	Indica il numero di righe da mostrare
ColumnHeadersVisible	bool	Se è False, nasconde le intestazioni di colonna (prima riga)
RowHeadersVisible	bool	Se è False, nasconde le intestazioni di riga (prima colonna)
AutoSizeColumnsMode	oggetto	Indica come determinare la larghezza delle colonne
AutoSizeRowsMode	oggetto	Indica come determinare la altezza delle righe
ReadOnly	bool	Se è True, impedisce all'utente di modificare il controllo
Rows	oggetto	Indica le righe del controllo, permette l'accesso alle celle
Columns	oggetto	Indica le colonne del controllo, permette l'accesso alle celle

AllowUserToAddRows	bool	Se è False, impedisce di aggiungere righe al controllo
AllowUserToDeleteRows	bool	Se è False, impedisce di eliminare righe dal controllo
AllowUserToOrderColumns	bool	Se è False, impedisce di ordinare in base ai valori di colonna
Metodo	**Nota**	**Scopo**
ClearSelection()	metodo	Pulisce qualsiasi selezione del dataGrid

ACCESSO ALLE CELLE

Analizziamo un po' meglio l'accesso alle celle del controllo. Un primo passo consiste nell'usare la proprietà `Rows` che indica la collezione di tutte le righe del controllo:

```vc#
dataGridView1.Rows
```

poiché questa proprietà è una collezione è possibile usarla COME SE fosse un vettore, ovvero è possibile con un indice (nell'esempio r) un suo elemento:

```vc#
dataGridView1.Rows[r]
```

che indica la riga numero r del controllo; questa riga è un oggetto con relative proprietà e metodi; per esempio dispone dei metodi `Add` (aggiungere una riga), `Remove` ed `Insert` (inserisci una riga in un determinato indice):

```vc#
dataGridView1.Rows.Add("daino", "gatto", "pollo", "serpe");
dataGridView1.Rows.Insert(0, "ape", "bue", "cane", "dodo");
```

Tramite la proprietà Rows si può accedere alla proprietà `Cells` che fa riferimento alle celle della riga; in particolare:

```vc#
dataGridView1.Rows[r].Cells[c]
```

indica la cella numero c della riga r, ovvero coincide con la cella di coordinate [r,c]. Questo elemento (la cella intesa come oggetto) è ricco di proprietà ed utilizzi. Ne vedremo solo alcuni, cercando di fornire spunti interessanti per scenari di sviluppo:

Value	Indica il valore della cella
Esempio:	`dataGridView1.Rows[2].Cells[2].Value = "cane";`

Style.BackColor	Indica il colore della cella
Esempio:	`dataGridView1.Rows[2].Cells[2].Style.BackColor = Color.Yellow;`

PROGETTO GUIDATO

- Si prepari un Form1 simile alla figura, con cinque pulsanti, con il testo vuoto e le icone impostate con la proprietà Image caricata (sei libero di porre le immagini preferite);

- Dichiara le seguenti variabili globali

VC#

```
int giocatore = 1;
//i giocatori saranno due,
contrassegnati 0 e 1
int punti1 = 0;
int punti0 = 0;
int difficoltà = 5;
```

- Cerca nella finestra delle Proprietà, la paletta degli Eventi (c'è un fulmine nel pulsante che le fa comparire) ; cerca l'evento CellClick e fai doppio clic nello spazio accanto; deve comparire un gestore simile al seguente:

VC#

```
private void dataGridView1_CellClick
    (object sender, DataGridViewCellEventArgs e)
{
     |
}
```

- modifica il gestore inserendo il seguente codice:

VC#

```
private void dataGridView1_CellClick
                              (object sender, DataGridViewCellEventArgs e)
{
   //valori di inizio gioco
   int giocatore = 1; //i giocatori saranno due, contrassegnati 0 e 1
   int punti1 = 0;
   int punti0 = 0;
   button2.Enabled = false;
   button3.Enabled = false;
```

```
   //imposta dimensioni tavola
   dataGridView1.RowCount = difficoltà;
   dataGridView1.ColumnCount = difficoltà;
   dataGridView1.RowHeadersWidth = 25;
   dataGridView1.ColumnHeadersHeight = 25;
   dataGridView1.SelectionMode = DataGridViewSelectionMode.CellSelect;
   dataGridView1.AutoSizeColumnsMode =
                        DataGridViewAutoSizeColumnsMode.AllCellsExceptHeader;
   dataGridView1.AutoSizeRowsMode =
                        DataGridViewAutoSizeRowsMode.AllCellsExceptHeaders;
```

```
    dataGridView1.RowHeadersVisible = false;
    dataGridView1.ColumnHeadersVisible = false;
    //scrive valori nella tavola
    Random valori = new Random();
    for (int r = 0; r < difficoltà; r++)
       for (int c = 0; c < difficoltà; c++)
          dataGridView1.Rows[r].Cells[c].Value = valori.Next(0, 10);
    dataGridView1.ClearSelection();
    button1.Enabled = false;
}
```

- Associa al primo pulsante il codice:

VC#

```
int r = e.RowIndex;
int c = e.ColumnIndex;
int punti = Convert.ToInt16(dataGridView1.Rows[r].Cells[c].Value);
giocatore = (1 + giocatore) % 2; //alterna 0 e 1
if ((dataGridView1.Rows[r].Cells[c].Style.BackColor == Color.Red)
            || dataGridView1.Rows[r].Cells[c].Style.BackColor == Color.Aqua)
                        punti = 22; //il giocatore corrente perderà!!!
```

```
if (giocatore == 0)
{
      dataGridView1.Rows[r].Cells[c].Style.BackColor = Color.Red;
      dataGridView1.ClearSelection();
      punti0 += punti;
      if (punti0 > 21)
      {
            dataGridView1.Enabled = false;
            MessageBox.Show("Vince il Blu");
            button1.Enabled = true;
      }
      else if (punti0 == 21)
      {
            dataGridView1.Enabled = false;
            MessageBox.Show("Vince il Rosso");
            button1.Enabled = true;
      }
}
```

```
else //giocatore == 1
{
      dataGridView1.Rows[r].Cells[c].Style.BackColor = Color.Aqua;
      dataGridView1.ClearSelection();
      punti1 += punti;
      if (punti1 > 21)
      {
            dataGridView1.Enabled = false;
            MessageBox.Show("Vince il Rosso");
            button1.Enabled = true;
      }
      else if (punti1 == 21)
      {
            dataGridView1.Enabled = false;
            MessageBox.Show("Vince il Blu");
            button1.Enabled = true;
      }
}
```

- Associa al secondo pulsante il codice:

VC#

```
difficoltà++;
button1.Text = "" + difficoltà;
```

- Associa al terzo pulsante il codice:

VC#

```
difficoltà--;
button1.Text = "" + difficoltà;
```

- Associa al quarto pulsante il codice:

VC#

```
string s = "devi fare 21 col tuo colore";
MessageBox.Show(s);
```

- Associa al quinto pulsante il codice:

VC#

```
//nessun codice per ora
```

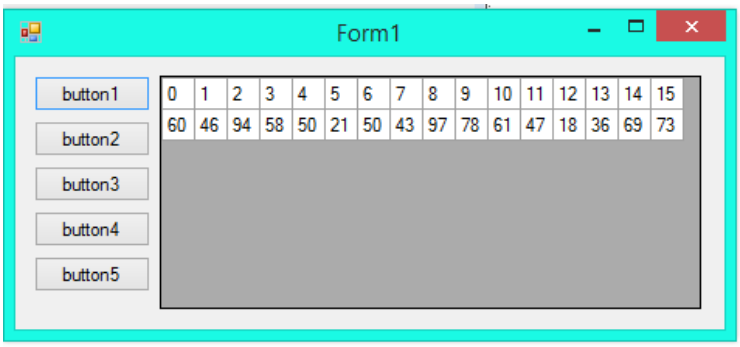
- Hai capito lo scopo del gioco? Prova il progetto …

ESERCIZI

ESERCIZI SU: MATRICI E DATAGRIDVIEW

ESERCIZIO 1. VETTORI DI INTERI

Dichiara un vettore globale



`Button1` crea un vettore di 16 celle con valori casuali

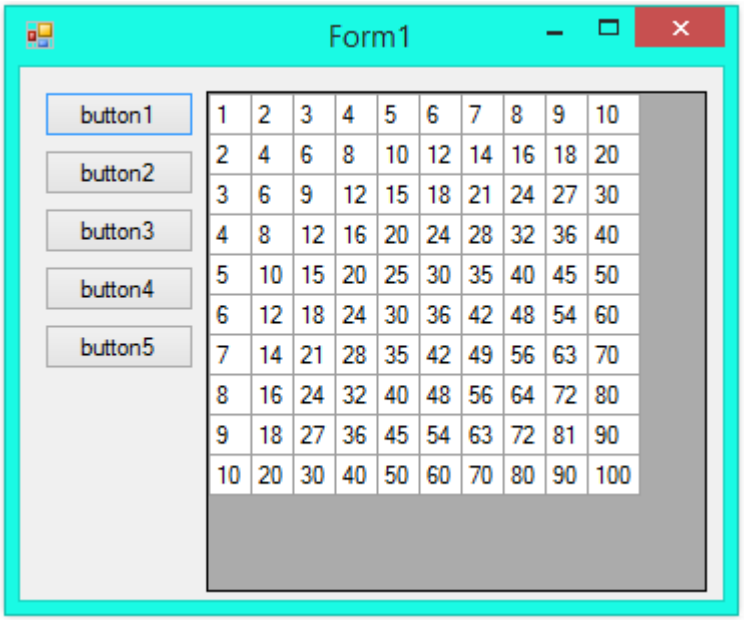
`Button2` mostra il vettore nel DataGridView (vedi figura)

`Button3` cerca nel vettore il valore massimo e lo sostituisce con 0; poi mostra il vettore nel DataGridView

`Button4` cerca nel vettore il valore minimo e lo sostituisce con 99; poi mostra il vettore nel DataGridView

`Button5` cerca nel vettore tutti i valori inferiori alla media e li sostituisce con 50; poi mostra il vettore nel DataGridView

ESERCIZIO 2. TABELLINE



`Button1` visualizza la classica tabellina delle moltiplicazioni nel DataGridView (vedi figura)

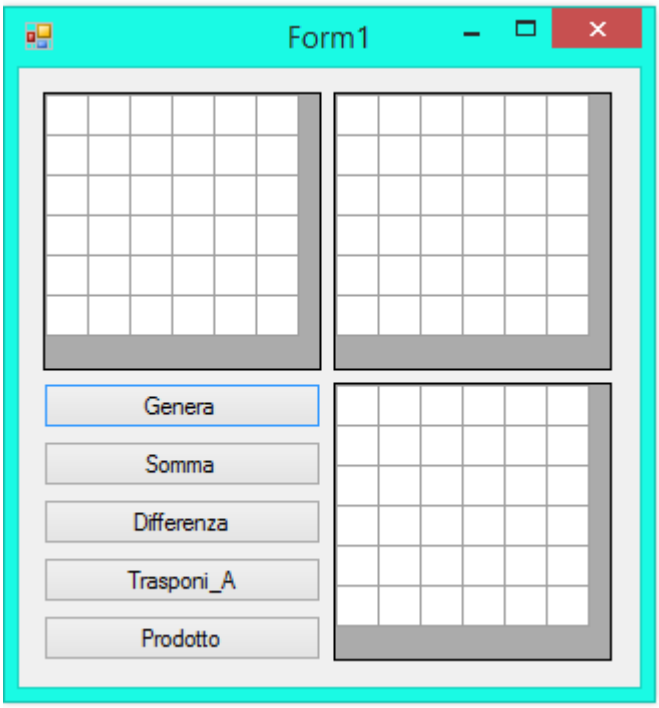
`Button2` simile al precedente, ma invece dei prodotti mostra e somme dei numeri

`Button3` in ogni cella mostra le coordinate di riga e colonna separate da un punto e virgola (es. **3;7**)

`Button4` scrive 0 in tutte le celle, tranne che sulla diagonale principale dove scrive 1

`Button5` scrive "" in tutte le celle, tranne che sulla diagonale principale dove scrive "♥"

ESERCIZIO 3. ALGEBRA MATRICIALE



`Genera` crea e visualizza due matrici e le visualizza nelle prime due DataGridView

`Somma` calcola la somma delle due matrici e visualizza la matrice somma nel terzo DataGridView

`Differenza` calcola la sottrazione delle due matrici e visualizza la matrice differenza nel terzo DataGridView

`Differenza` calcola la trasposta della prima matrice e visualizza la matrice trasposta nel terzo DataGridView

`Prodotto` calcola il prodotto delle due matrici e visualizza la matrice prodotto nel terzo DataGridView

ESERCIZIO 4. MEMORY

Il gioco del Memory prevede una matrice quadrata di ordine pari che contiene coppie di caratteri. La matrice NON viene visualizzata. Quando il giocatore sceglie la prima casella del DataGridView il programma mostra il contenuto della cella; quando sceglie una seconda casella può individuare la coppia (che resterà visualizzata) oppure trova carte diverse (ed un terzo clic nasconde di nuovo i valori scoperti); il programma tiene conto dei tentativi fatti. Il gioco finisce individuando tutte le coppie nascoste.

Pulsante 1: inizio del gioco, preparazione della matrice, nasconde i dati;

Pulsante 2: mostra i tentativi fatti finora;

Pulsante 3: abbandono del gioco;

Pulsante 4: chiude il gioco.

ESERCIZIO 5. BATTAGLIA NAVALE

Il gioco riguarda un solo giocatore contro il computer. Il computer nasconde 8 sottomarini (da una casella ciascuno) in una griglia di 36 celle marine. All'inizio le celle sono celesti. L'utente tenta di colpirli con un clic sulla casella; se c'è il sottomarino, la cella si colora di rosso (esplosione), altrimenti di bianco(schiuma). Il gioco tiene il conto dei tentativi. Quando il giocatore trova tutti i sottomarini, mostra il numero dei tentativi fatti.

Pulsante 1: inizio del gioco, preparazione della matrice, nasconde i dati;

Pulsante 2: mostra i tentativi fatti finora;

Pulsante 3: abbandono del gioco;

Pulsante 4: chiude il gioco.

ESERCIZIO 6. ARTIFICIERE

È un gioco per un solo giocatore. Il computer nasconde, in una griglia di 36 celle, 9 mine (ciascuna occupa una casella). All'inizio le celle sono bianche. L'utente deve sminare il terreno, evitando di incappare in una mina: con un clic sulla casella rivela se è libera; se c'è una bomba si perde subito; altrimenti la casella diventa verde (libera) e mostra il numero delle bombe che si trovano in tutte le caselle adiacenti (per un lato o per uno spigolo).

Il giocatore vince quando ha liberato tutte le caselle senza mine.

Pulsante 1: inizio del gioco, preparazione della matrice, nasconde i dati;

Pulsante 2: un messaggio informa delle caselle libere restanti;

Pulsante 3: aumenta la difficoltà +1 mina;

Pulsante 4: diminuisce la difficoltà -1 mina;

SOMMARIO