NT106.P22.ANTT - LẬP TRÌNH MẠNG CĂN BẢN

BÁO CÁO BÀI TẬP TUẦN 02

***

Thông tin cá nhân:

Họ và tên	MSSV	Email
Nguyễn Đình Khang	23520694	23520694@gm.uit.edu.vn

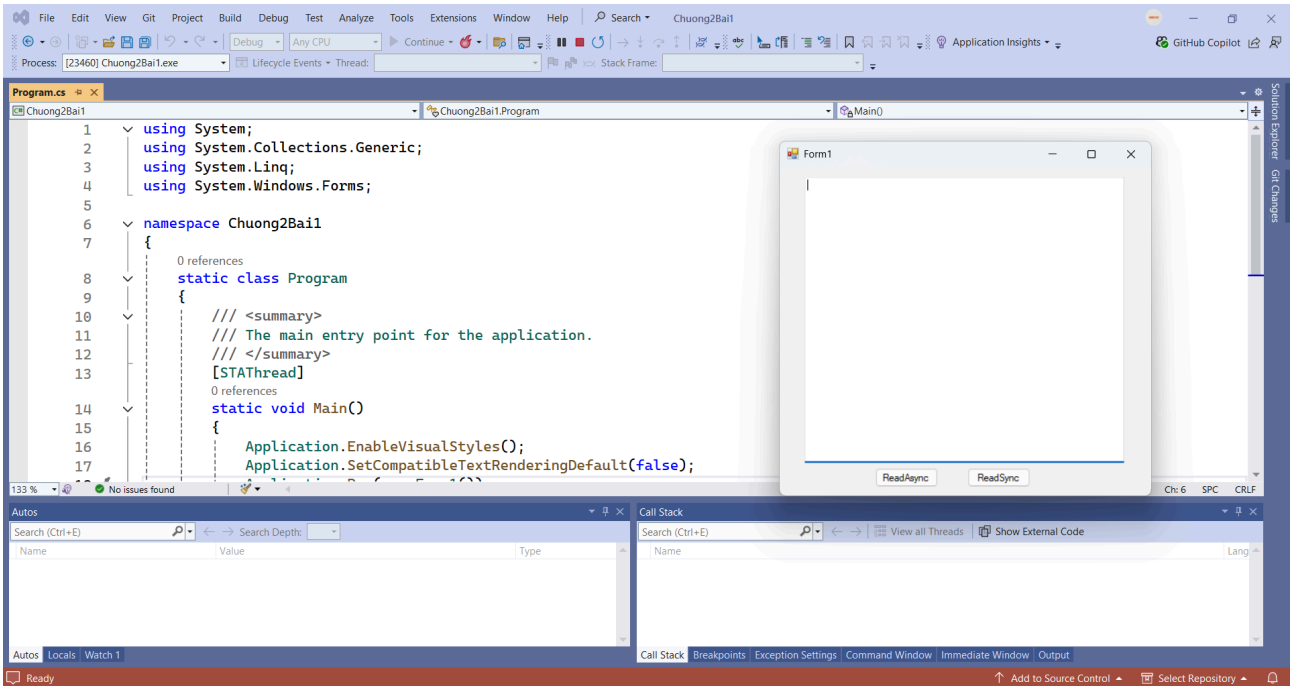
BÀI LÀM

I. BÀI TẬP 01

- Khi thực hiện chạy chương trình Chuong2Bai1, ứng dụng cho phép người dùng đọc nội dung từ một file và hiển thị lên giao diện, với hai chế độ đọc chính là đồng bộ (synchronous) và bất đồng bộ (asynchorous).

- Tại file Program.cs:

    + Đây là chương trình chịu trách cho việc khởi động giao diện chính (Form1).



+ Application.SetCompatibleTextRenderingDeFault(false) và Application.EnableVisualStyles() đảm bảo giao diện hiển thị đúng và tương thích với hệ điều hành.

+ Application.Run(new Form1()) khởi tạo và chạy form chính.

- Tại file Form1.Designer.cs

+ tbResults: Chức năng tạo một Textbox để hiển thị nội dung file.

```
41          // tbResults
42          //
43          this.tbResults.Location = new System.Drawing.Point(30, 12);
44          this.tbResults.Multiline = true;
45          this.tbResults.Name = "tbResults";
46          this.tbResults.Size = new System.Drawing.Size(381, 341);
47          this.tbResults.TabIndex = 0;
48          //
```

+ btnReadAsync và btnReadSync: Đây là hai nút bấm dùng để đọc file theo hai chế độ.

- btnReadAsync: Đọc file theo chế độ bất đồng bộ.

```
49          // btnReadAsync
50          //
51          this.btnReadAsync.Location = new System.Drawing.Point(114, 359);
52          this.btnReadAsync.Name = "btnReadAsync";
53          this.btnReadAsync.Size = new System.Drawing.Size(75, 23);
54          this.btnReadAsync.TabIndex = 1;
55          this.btnReadAsync.Text = "ReadAsync";
56          this.btnReadAsync.UseVisualStyleBackColor = true;
57          this.btnReadAsync.Click += new System.EventHandler(this.btnReadAsync_Click);
58          //
```

- btnReadSync: Đọc file theo chế độ đồng bộ.

```
59          // btnReadSync
60          //
61          this.btnReadSync.Location = new System.Drawing.Point(224, 359);
62          this.btnReadSync.Name = "btnReadSync";
63          this.btnReadSync.Size = new System.Drawing.Size(75, 23);
64          this.btnReadSync.TabIndex = 2;
65          this.btnReadSync.Text = "ReadSync";
66          this.btnReadSync.UseVisualStyleBackColor = true;
67          this.btnReadSync.Click += new System.EventHandler(this.btnReadSync_Click);
68          //
```

+ openFileDialog: Chức năng dùng để tạo hộp thoại cho người dùng chọn file để tải lên từ máy tính.

```
36          //
37          // openFileDialog
38          //
39          this.openFileDialog.FileName = "openFileDialog1";
40          //
```

- Tại file Form1.cs

+ InitializeComponent() là phương thức khởi tạo được gọi để thiết lập giao diện từ Form1.Designer.cs

```
1 reference
public Form1()
{
    InitializeComponent();
}
```

+ Sự kiện btnReadAsync_Click:

- openFileDialog có chức năng giúp người dùng chọn file từ máy tính.
- Tạo FileStream với chế độ bất đồng bộ (true) và kích thước bộ đệm là 4096 byte.
- Sau đó, khởi động đọc file bất đồng bộ bằng BeginRead, truyền dữ liệu thông qua fileContents và gọi fs_StateChanged sau khi hoàn tất.

```
1 reference
private void btnReadAsync_Click(object sender, EventArgs e)
{
    openFileDialog.ShowDialog();
    callback = new AsyncCallback(fs_StateChanged);
    fs = new FileStream(openFileDialog.FileName, FileMode.Open,
    FileAccess.Read, FileShare.Read, 4096, true);
    fileContents = new Byte[fs.Length];
    fs.BeginRead(fileContents, 0, (int)fs.Length, callback, null);
}
```

+ Phương thức fs_StateChanged:

- Được sử dụng để kiểm tra nếu quá trình đọc file hoàn tất (IsCompleted).
- Chuyển đổi mảng byte fileContest thành chuỗi UTF-8 và gọi InfoMessage để hiển thị lên tbResutls.
- Sau đó, đóng FileStream.

```
1 reference
private void fs_StateChanged(IAsyncResult asyncResult)
{
    if (asyncResult.IsCompleted)
    {
        string s = Encoding.UTF8.GetString(fileContents);

        InfoMessage(s);
        //tbResults.Text = Encoding.UTF8.GetString(fileContents);

        fs.Close();
    }
}
```

+ Tại phương thức InfoMessage: Có chức năng kiểm tra InvokeRequired để đảm bảo quá trình cập nhật tbResults trên thread UI (thread an toàn cho giao diện).

```
3 references
public void InfoMessage(String info)
{
    if (tbResults.InvokeRequired)
    {
        InfoMessageDel method = new InfoMessageDel(InfoMessage);
        tbResults.Invoke(method, new object[] { info });
        return;
    }

    tbResults.Text = info;
}
```

+ Sự kiện btnReadSync_Click:

- Có openFileDialog dùng để chọn file từ máy tính.
- Tạo một thread mới (thdSyncRead) và gọi phương thức syncRead trên thread đó.

```
1 reference
private void btnReadSync_Click(object sender, EventArgs e)
{
    openFileDialog.ShowDialog();
    Thread thdSyncRead = new Thread(new ThreadStart(syncRead));
    thdSyncRead.Start();
}
```

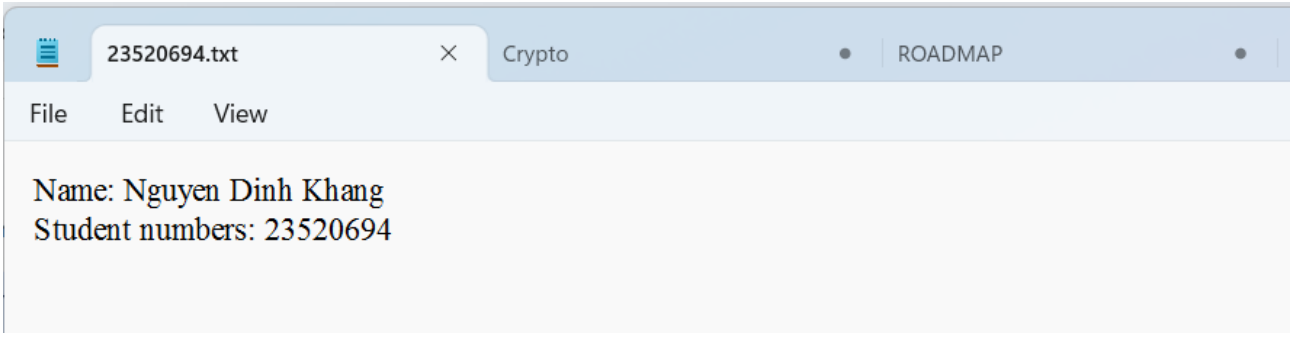
+ Phương thức syncRead:

- Mở FileStream với chế độ OpenOrCreate (tạo file mới nếu không tồn tại).
- Đọc toàn bộ nội dung vào mảng byte fileContents.
- Chuyển đổi thành chuỗi UTF-8 và gọi InfoMessage để hiển thị.
- Đóng FileStream.

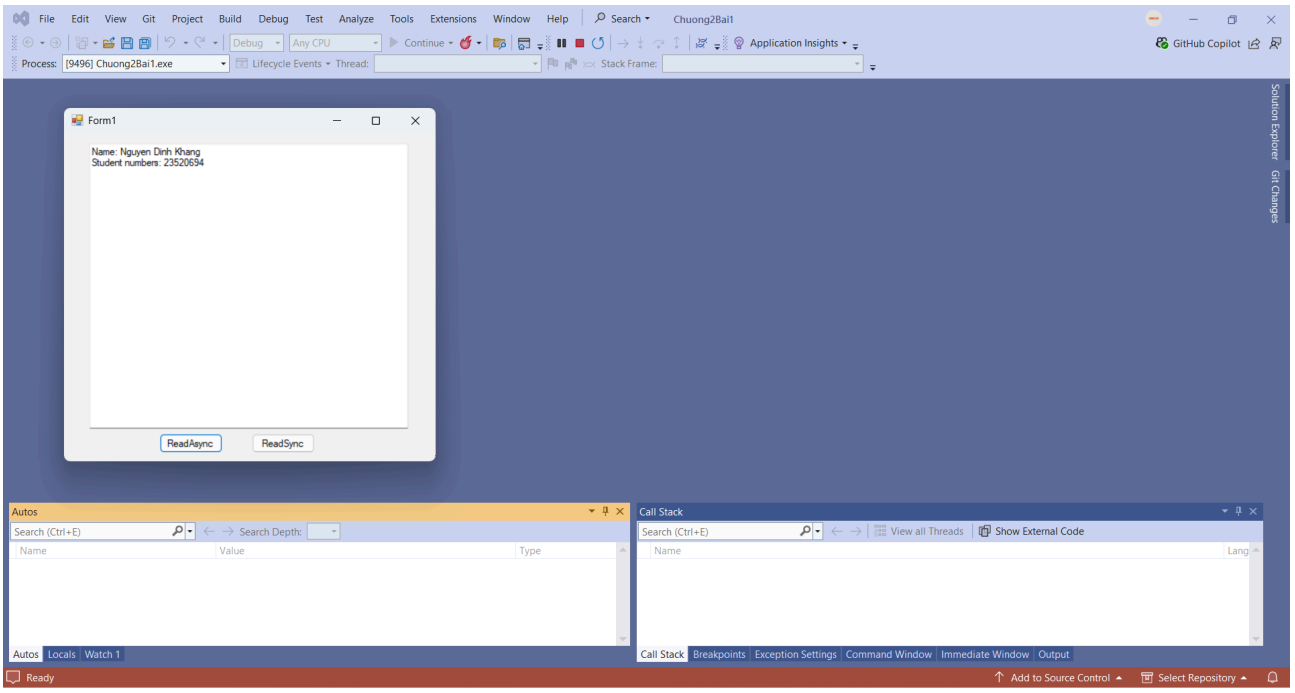
```
1 reference
public void syncRead()
{
    //OpenFileDialog ofd = new OpenFileDialog();
    //ofd.ShowDialog();
    //openFileDialog.ShowDialog();
    FileStream fs;
    try
    {
        fs = new FileStream(openFileDialog.FileName, FileMode.OpenOrCreate);
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
        return;
    }
    fs.Seek(0, SeekOrigin.Begin);
    byte[] fileContents = new byte[fs.Length];
    fs.Read(fileContents, 0, (int)fs.Length);
    //tbResults.Text = Encoding.UTF8.GetString(fileContents);
    string s = Encoding.UTF8.GetString(fileContents);

    InfoMessage(s);
    fs.Close();
}
}
}
```

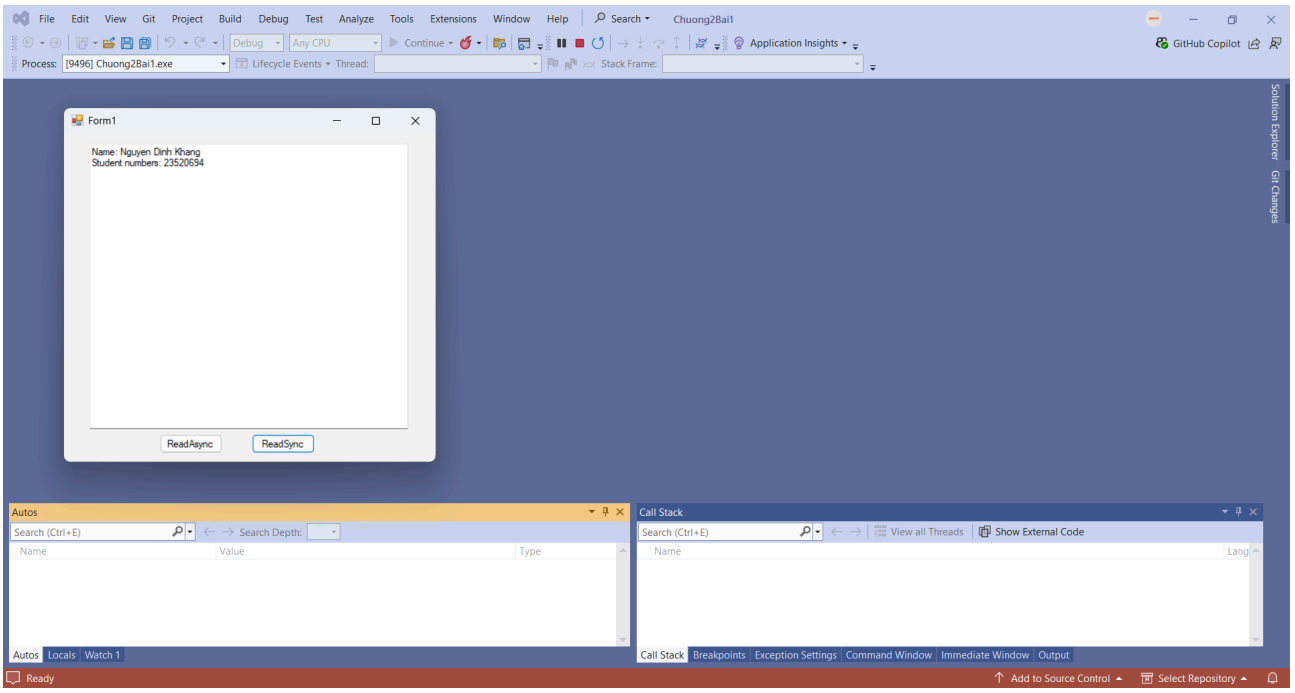
- Về kết quả hiển thị: Thực hiện đọc nội dung bên trong file 23520694.txt thông qua quá trình hai nút chức năng:



+ Giao diện khi thực hiện tương tác với giao diện thông qua nút ReadAsync (Đọc bất đồng bộ)



+ Giao diện khi thực hiện tương tác với giao diện thông qua nút ReadSync (Đọc đồng bộ)



- Nhận xét:

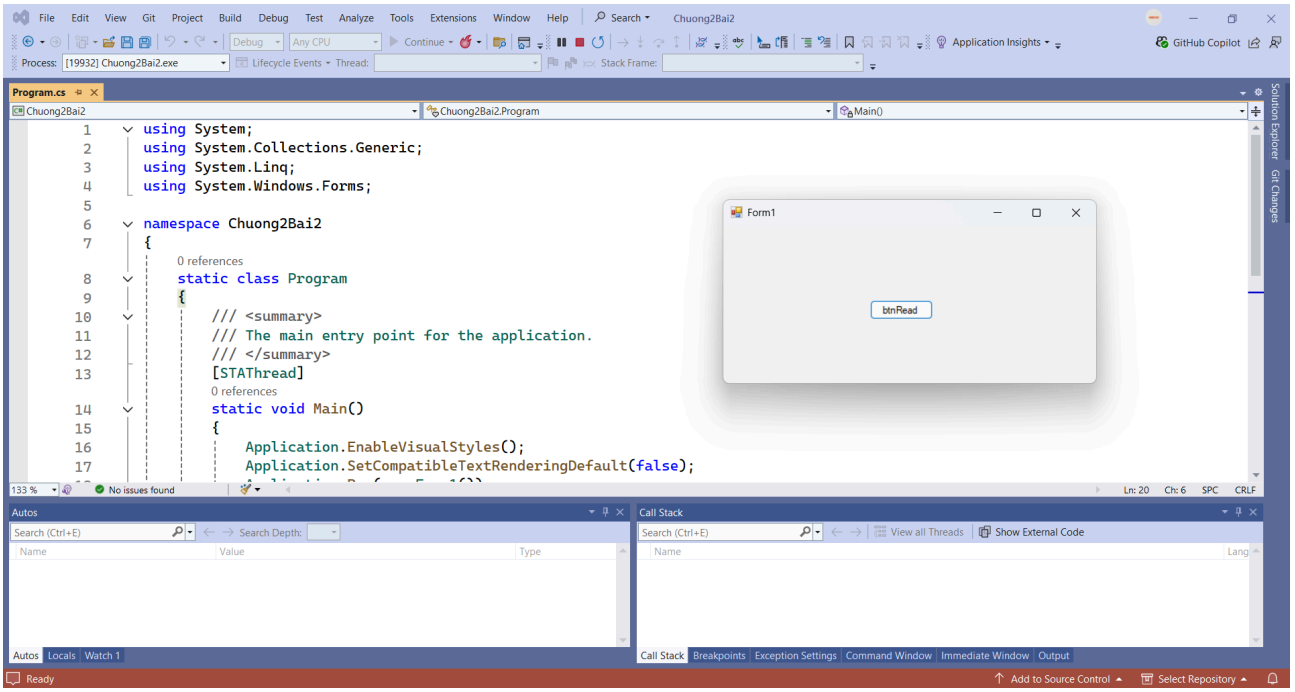
+ Đối với file test có kích thước khoảng 1KB, khi thực hiện thử nghiệm với ReadSync hoặc ReadAsync, ta quan sát thời gian đọc và trạng thái giao diện, nhận thấy thời gian đọc gần như ngay lập tức, không có sự khác biệt quá lớn khi so sánh giữa hai chức năng đọc đồng bộ và bất đồng bộ.

II. BÀI TẬP 02

- Khi thực hiện chạy chương trình Chuong2Bai2, ứng dụng cho người dùng biết tổng số hàng hiện có trong một file dữ liệu đầu vào.

- Tại Program.cs:

+ Đây là chương trình chịu trách cho việc khởi động giao diện chính (Form1).



+ Application.EnableVisualStyles(): Có chức năng kích hoạt các kiểu giao diện trực quan (visual styles) để giao diện trông hiện đại và phù hợp với hệ điều hành Windows.

+ Application.SetCompatibleTextRenderingDefault(false): Có chức năng thiết lập chế độ vẽ văn bản mặc định, đảm bảo tương thích với các phiên bản cũ của Windows Forms.

+ Application.Run(new Form1()): Có chức khởi tạo một instance của Form1 và chạy ứng dụng, hiển thị giao diện chính (như hình).

- Tại Form1.Designer.cs:

+ Phương thức InitializeComponent: Có chức năng dùng để thiết lập các thuộc tính của btnRead (như vị trí, kích thước, sự kiện,...). Ngoài ra, phương thức này còn có SuspendLayout() và ResumeLayout() giúp tối ưu hóa quá trình vẽ giao diện khi thêm controls.

```
29      private void InitializeComponent()
30      {
31          this.btnRead = new System.Windows.Forms.Button();
32          this.SuspendLayout();
33          //
34          // btnRead
35          //
36          this.btnRead.Location = new System.Drawing.Point(175, 89);
37          this.btnRead.Name = "btnRead";
38          this.btnRead.Size = new System.Drawing.Size(75, 23);
39          this.btnRead.TabIndex = 0;
40          this.btnRead.Text = "btnRead";
41          this.btnRead.UseVisualStyleBackColor = true;
42          this.btnRead.Click += new System.EventHandler(this.btnRead_Click);
43          //
44          // Form1
45          //
46          this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
47          this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
48          this.ClientSize = new System.Drawing.Size(445, 188);
49          this.Controls.Add(this.btnRead);
50          this.Name = "Form1";
51          this.Text = "Form1";
52          this.ResumeLayout(false);
53
54      }
```

+ Phương thức Dispose: Có chức năng dùng để giải phóng tài nguyên khi form bị đóng, đặc biệt là đối tượng components nếu tồn tại.

```
14      protected override void Dispose(bool disposing)
15      {
16          if (disposing && (components != null))
17          {
18              components.Dispose();
19          }
20          base.Dispose(disposing);
21      }
22
```

- Tại Form1.cs:

+ Phương thức khởi tạo (Form1): InitializeComponent() được gọi để thiết lập giao diện từ Form1.Designer.cs

```
        1 reference
15      public Form1()
16      {
17          InitializeComponent();
18      }
19
20
```

+ Sự kiện btnRead_Click:

```
1 reference
21  private void btnRead_Click(object sender, System.EventArgs e)
22  {
23      OpenFileDialog ofd = new OpenFileDialog();
24      ofd.ShowDialog();
25      FileStream fs = new FileStream(ofd.FileName, FileMode.OpenOrCreate);
26      StreamReader sr = new StreamReader(fs);
27      int lineCount = 0;
28      while (sr.ReadLine() != null)
29      {
30          lineCount++;
31      }
32      fs.Close();
33      MessageBox.Show("There are " + lineCount + " lines in " + ofd.FileName);
34  }
35  }
36  }
37
```

- Tạo và hiển thị OpenFileDialog:
  - OpenFileDialog ofd = new OpenFileDialog(): Có chức năng tạo một hộp thoại cho phép người dùng chọn file từ hệ thống.
  - ofd.ShowDialog(): Hiển thị hộp thoại và chờ người dùng chọn file từ máy tính hoặc hủy bỏ.

```
23      OpenFileDialog ofd = new OpenFileDialog();
24      ofd.ShowDialog();
```

- Đọc file:
  - FileStream fs = new FileStream(ofd.FileName, FileMode.OpenOrCreate): Có chức năng tạo một luồng file (FileStream) để đọc hoặc tạo file (nếu không tồn tại) từ đường dẫn được chọn.
  - StreamReader sr = new StreamReader(fs): Có chức năng tạo một đối tượng StreamReader để đọc văn bản từ file (dữ liệu đầu vào của người dùng).
  - int lineCount = 0: Khởi tạo biến đếm số dòng.
  - while (sr.Readline() != null) { lineCount++ }: Có chức năng đọc từng dòng của file bằng ReadLine() và tăng biến lineCount cho đến khi hết file (trả về null).

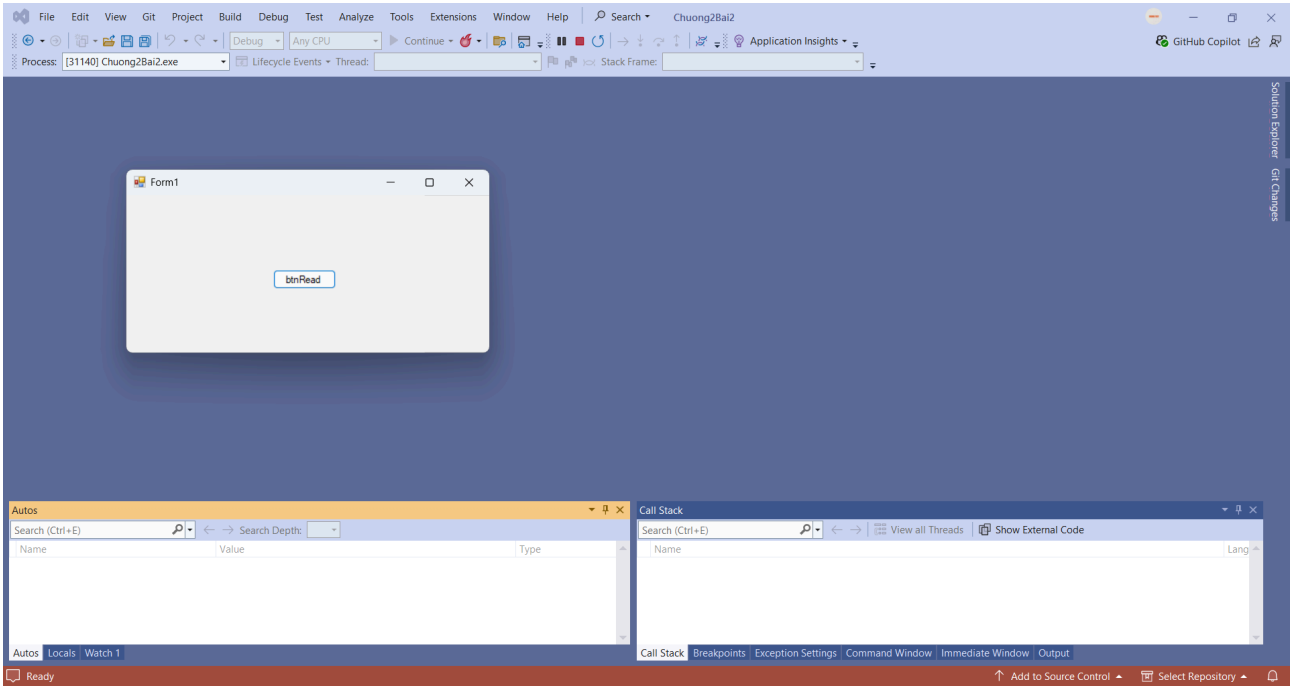
```
25      FileStream fs = new FileStream(ofd.FileName, FileMode.OpenOrCreate);
26      StreamReader sr = new StreamReader(fs);
27      int lineCount = 0;
28      while (sr.ReadLine() != null)
29      {
30          lineCount++;
31      }
```

- Đóng tài nguyên và hiển thị kết quả:
    - fs.Close(): Đóng luồng file để giải phóng tài nguyên.
    - MessageBox.Show(“There are ” + lineCount + “ lines in ” + ofd.FileName): Có chức năng hiển thị một hộp thoại thông báo số dòng và tên file.

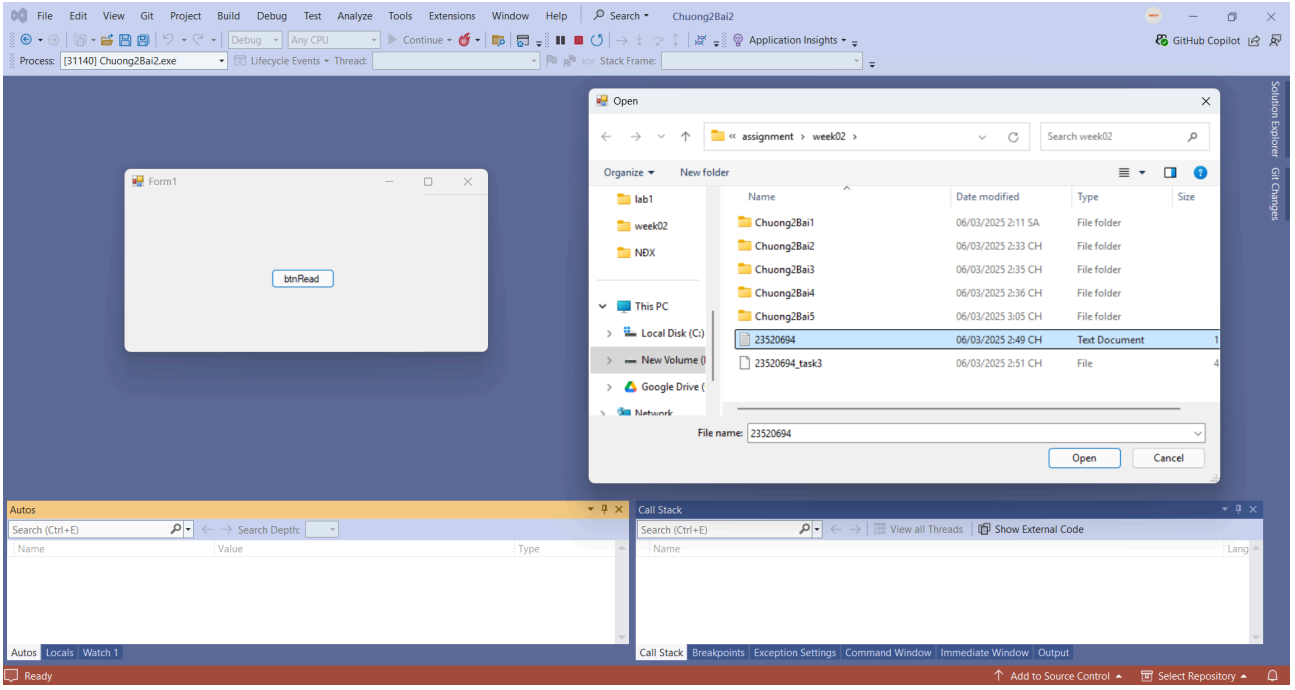
```
32                  fs.Close();
33                  MessageBox.Show("There are " + lineCount + " lines in " + ofd.FileName);
34              }
35          }
36      }
37
```

- Cách thức vận hành của chương trình:

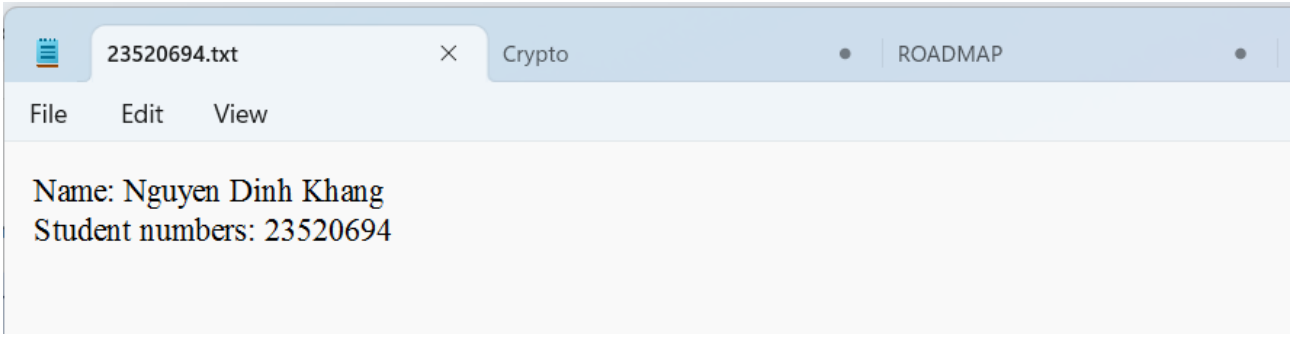
+ Khi chạy chương trình Chuong2Bai2, chương trình sẽ hiển thị ra Form như hình.



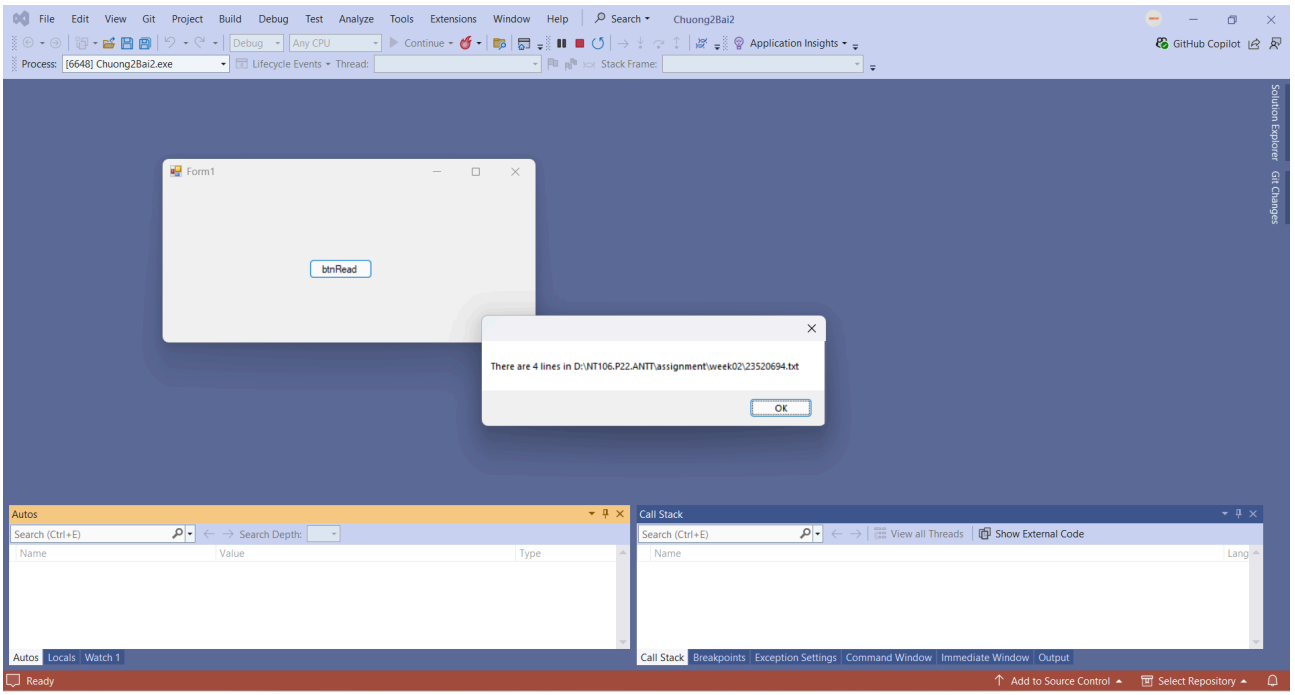
+ Khi người dùng nhấn nút “btnRead”, chương trình sẽ mở ra hộp thoại chọn file.



+Thực hiện đếm số dòng trong file 23520694.txt, nội dung file như sau:



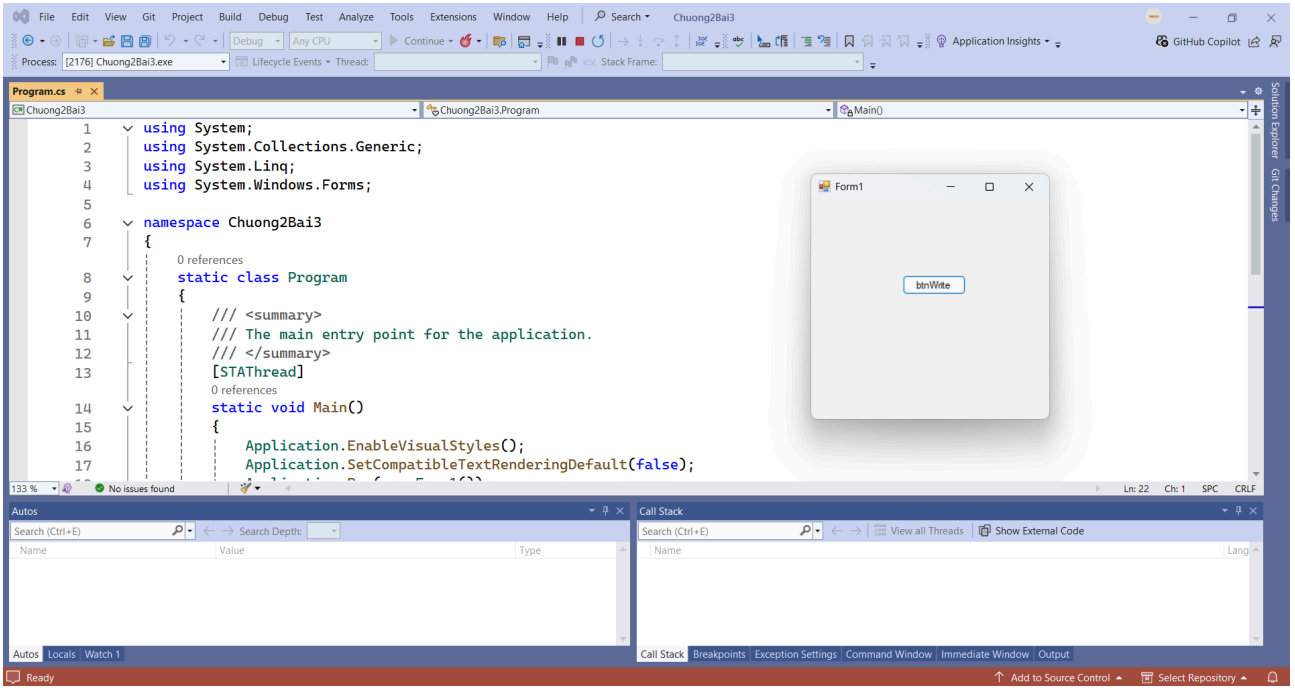
+ Sau khi chọn file. chương trình sẽ thực hiện đọc toàn bộ nội dung file và đếm số lượng dòng, rồi hiển thị kết quả thông qua MessageBox.



III. BÀI TẬP 03

- Tại Program.cs

+ Đây là chương trình chịu trách cho việc khởi động giao diện chính (Form1).



+ Application.EnableVisualStyles(): Có chức năng kích hoạt các kiểu giao diện trực quan để giao diện trông hiện đại hơn.

+ Application.SetCompatibleTextRenderingDefault(false): Đặt chế độ vẽ văn bản mặc định, đảm bảo tương thích với các phiên bản cũ của Windows Forms.

+ Application.Run(new Form1()): Có chức năng khởi tạo và chạy form chính Form1, hiển thị giao diện ra màn hình.

- Tại Form1.Designer.cs

+ Phương thức InitializeComponent: Có chức năng thiết lập thuộc tính của btnWrite như vị trí, kích thước, sự kiện,... Ngoài ra, SuspendLayout() và ResumeLayout() có chức năng giúp tối ưu hóa quá trình vẽ giao diện khi thêm các controls.

```
29      private void InitializeComponent()
30      {
31          this.btnWrite = new System.Windows.Forms.Button();
32          this.SuspendLayout();
33          //
34          // btnWrite
35          //
36          this.btnWrite.Location = new System.Drawing.Point(109, 90);
37          this.btnWrite.Name = "btnWrite";
38          this.btnWrite.Size = new System.Drawing.Size(75, 23);
39          this.btnWrite.TabIndex = 0;
40          this.btnWrite.Text = "btnWrite";
41          this.btnWrite.UseVisualStyleBackColor = true;
42          this.btnWrite.Click += new System.EventHandler(this.btnWrite_Click);
43          //
44          // Form1
45          //
46          this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
47          this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
48          this.ClientSize = new System.Drawing.Size(284, 262);
49          this.Controls.Add(this.btnWrite);
50          this.Name = "Form1";
51          this.Text = "Form1";
52          this.ResumeLayout(false);
53
54      }
```

+ Phương thức Dispose: Có chức năng giải phóng tài nguyên khi form bị đóng, đặc biệt là đối tượng components nếu tốn tài.

```
14      protected override void Dispose(bool disposing)
15      {
16          if (disposing && (components != null))
17          {
18              components.Dispose();
19          }
20          base.Dispose(disposing);
21      }
22
```

- Tại Form1.cs

+ Phương thức khởi tạo InitializeComponent() được gọi để thiết lập giao diện từ Form1.Designer.cs

+ Sự kiện btnWrite_Click:

```
1 reference
20      private void btnWrite_Click(object sender, EventArgs e)
21      {
22          SaveFileDialog sfd = new SaveFileDialog();
23          sfd.ShowDialog();
24          FileStream fs = new FileStream(sfd.FileName, FileMode.CreateNew);
25          BinaryWriter bw = new BinaryWriter(fs);
26          int[] myArray = new int[1000];
27          for (int i = 0; i < 1000; i++)
28          {
29              myArray[i] = i;
30              bw.Write(myArray[i]);
31          }
32          bw.Close();
33      }
34  }
35  }
36
```

- Tạo và hiển thị SaveFileDialog:
  - SaveFileDialog sfd = new SaveFileDialog(): Có chức năng tạo một hộp thoại cho phép người dùng chọn vị trí và tên file để lưu.
  - sfd.ShowDialog(): Hiển thị hộp thoại và chờ người dùng chọn vị trí hoặc hủy bỏ.

```
22          SaveFileDialog sfd = new SaveFileDialog();
23          sfd.ShowDialog();
```

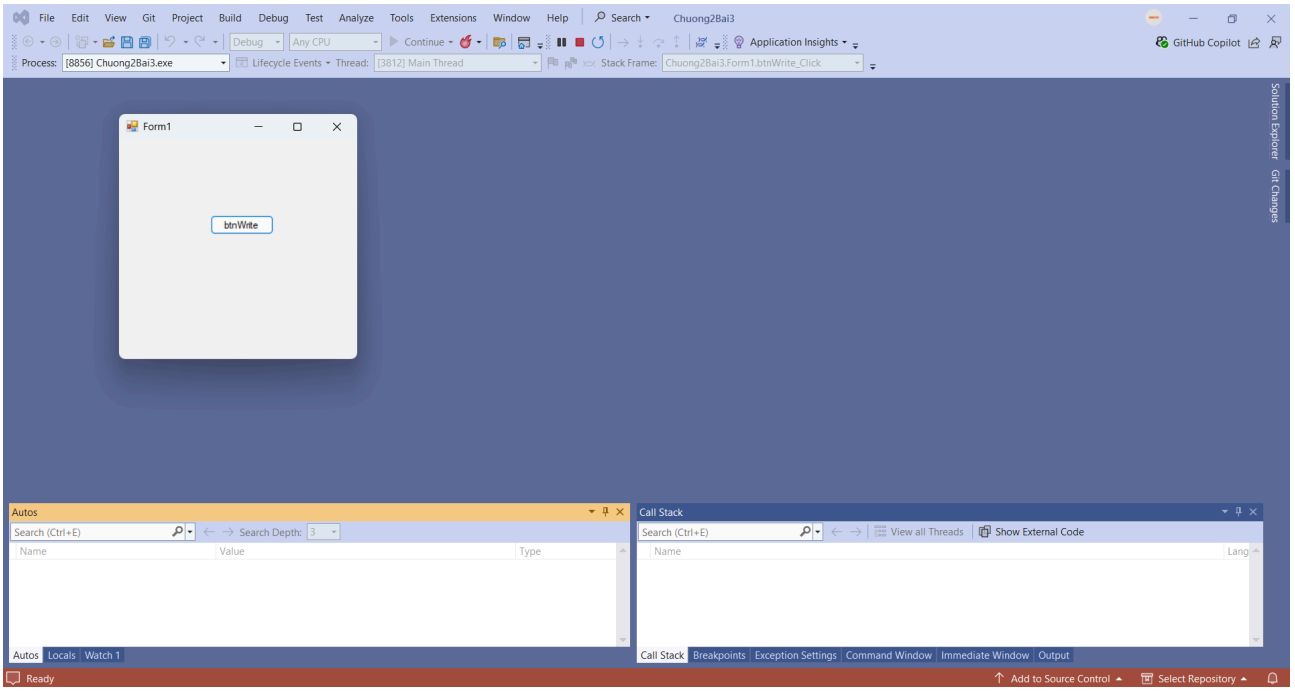
- Ghi file:
    - FileStream fs = new FileStream(sfd.FileName, FileMode.CreateNew): Có chức năng tạo một luồng file (FileStream) để ghi dữ liệu, với FileMode.CreateNew tạo file mới (thất bại nếu file đã tồn tại).
    - BinaryWriter bw = new BinaryWriter(fs): Tạo một đối tượng BinaryWriter để ghi dữ liệu nhị phân vào luồng file.
    - int[] myArray = new int[1000]: Khởi tạo một mảng nguyên (int) có 1000 phần tử.
    - for(int i = 0; i < 1000; i++) { myArray[i] = i; bw.Write(myArray[i]); }: Có chức năng gán giá trị i cho mỗi phần tử trong mảng và ghi từng giá trị vào file dưới dạng nhị phân.

```
24          FileStream fs = new FileStream(sfd.FileName, FileMode.CreateNew);
25          BinaryWriter bw = new BinaryWriter(fs);
26          int[] myArray = new int[1000];
27          for (int i = 0; i < 1000; i++)
28          {
29              myArray[i] = i;
30              bw.Write(myArray[i]);
31          }
```

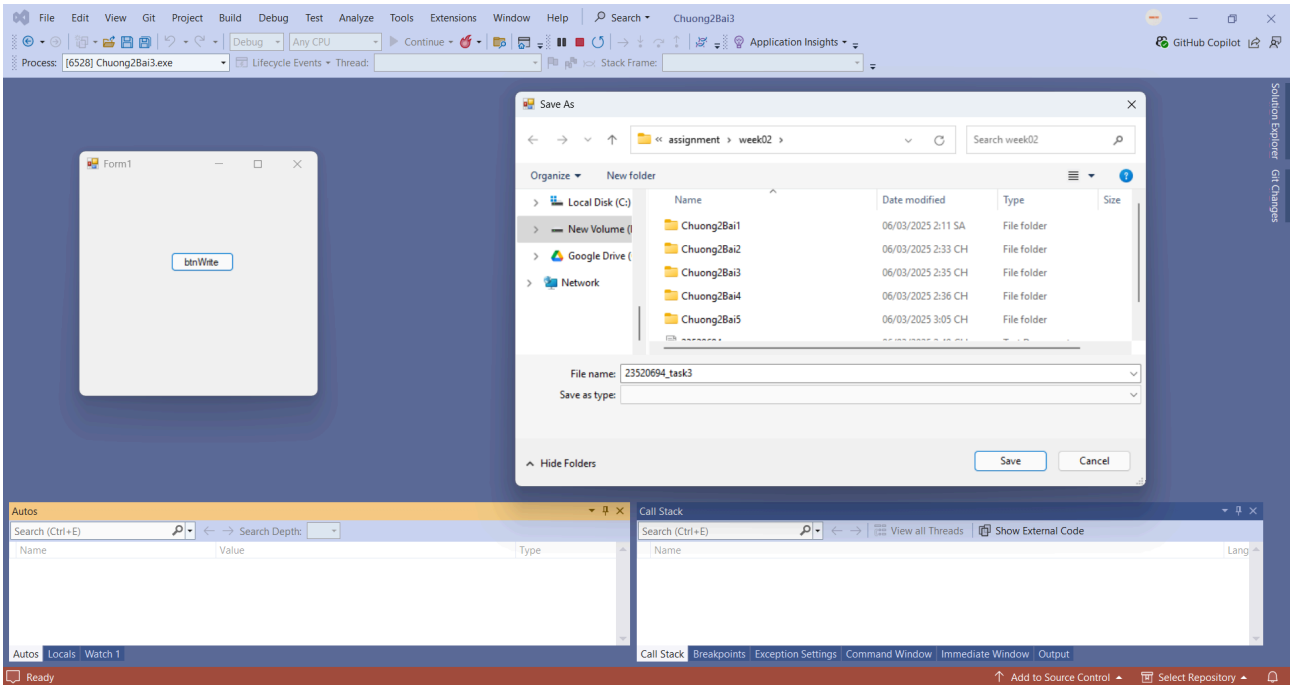
- Đóng tài nguyên:
    - btw.Close(): Đóng BinaryWriter, tự đóng file FileStream liên kết.

- Cách thức vận hành của chương trình:

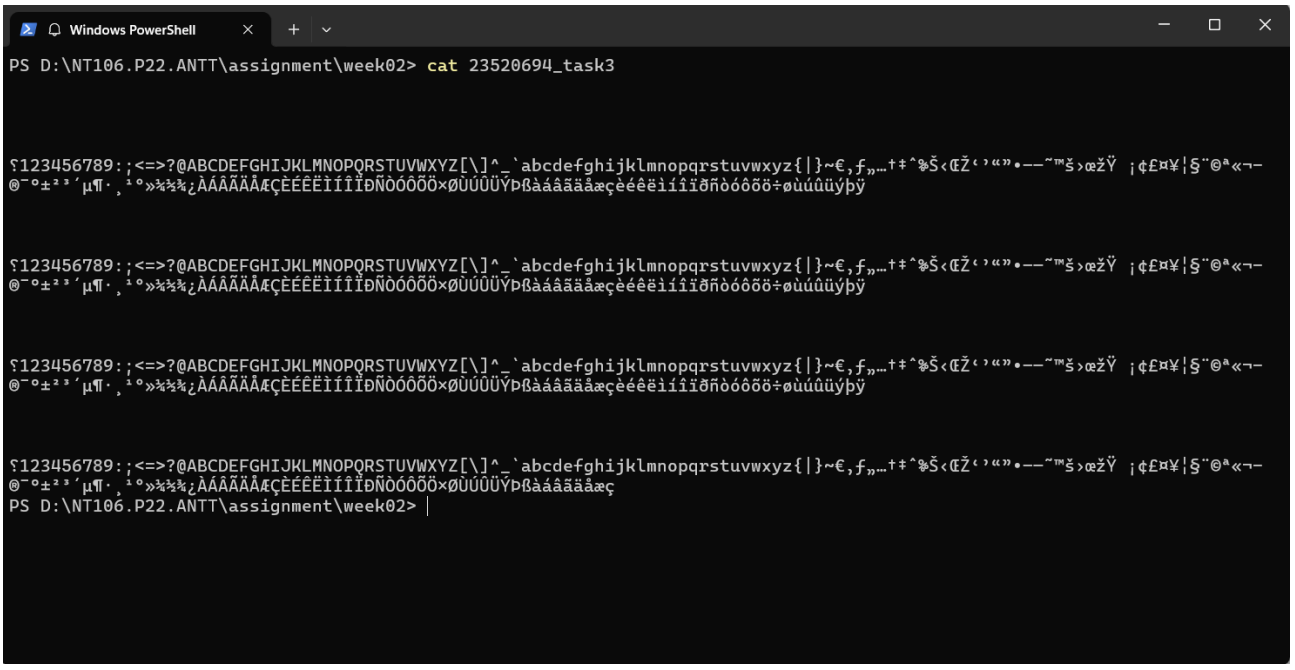
+ Khi chạy chương trình Chuong2Bai3, chương trình sẽ hiển thị ra Form như hình.



+ Khi người dùng nhất nút “btnWrite”, chương trình sẽ mở hộp thoại SaveFileDialog. Sau đó, người dùng chọn vị trí và tên file mà mình muốn lưu (ở đây, người dùng chọn lưu file với tên gọi 23520694_task3).



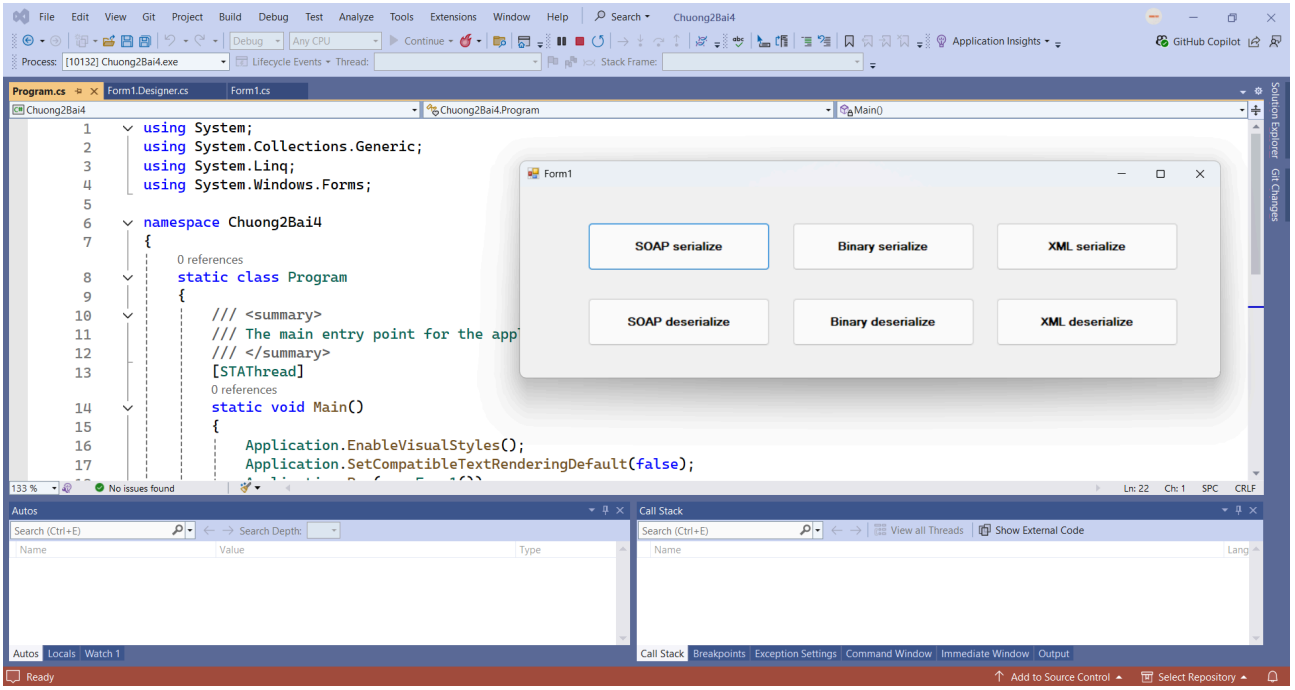
+ Tiếp sau đó, chương trình sẽ tạo một file mới, tạo mảng gồm 1000 số nguyên tố (từ 0 đến 999), và ghi từng số vào file dưới dạng nhị phân. Sau khi ghi xong, file được đóng và quá trình hoàn tất (không có thông báo xác nhận). Nội dung của file khi thực hiện lệnh “cat 23520694_task3”.



III. BÀI TẬP 04

- Tại Program.cs

+ Đây là chương trình chịu trách cho việc khởi động giao diện chính (Form1).




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

namespace Chuong2Bai4
{
    0 references
    static class Program
    {
        /// <summary>
        /// The main entry point for the app
        /// </summary>
        [STAThread]
        0 references
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
```

+ Application.EnableVisualStyles(): Có chức năng kích hoạt các kiểu giao diện trực quan.

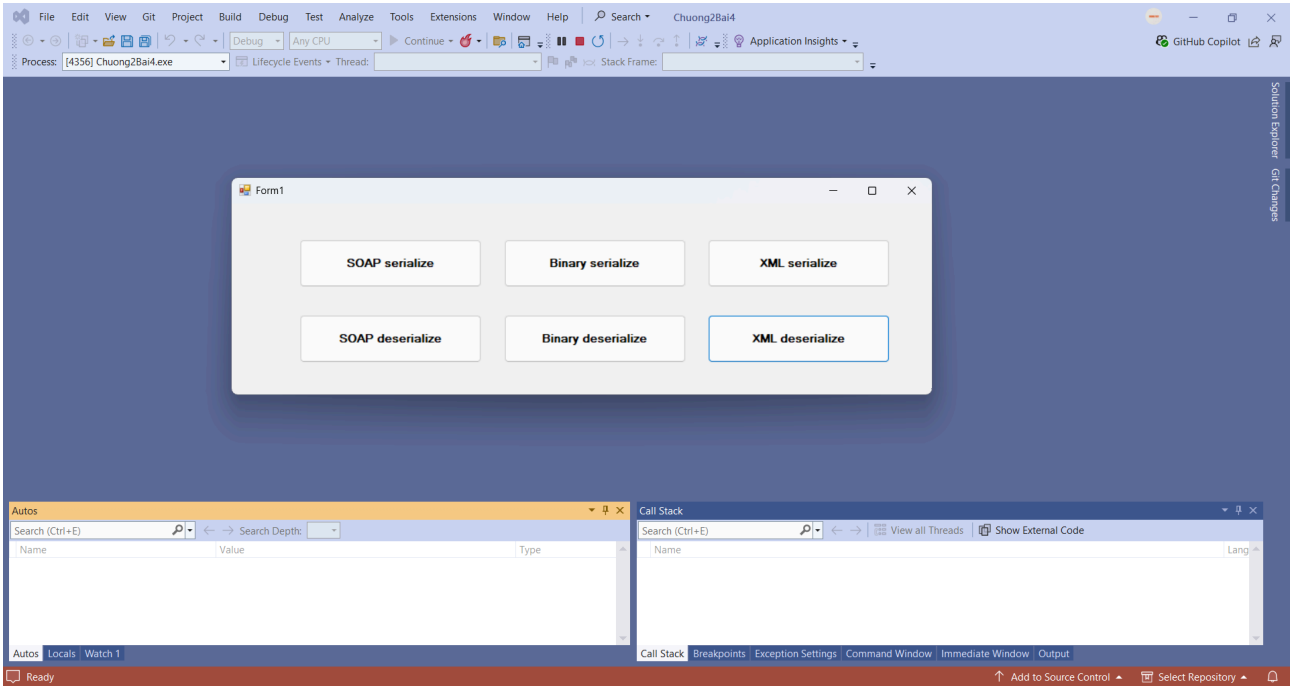
+ Application.SetCompatibleTextRederingDefault(false): Được sử dụng nhằm mục đích đặt chế độ vẽ văn bản mặt định, đảm bảo tương thích với các phiên bản cũ của Windows Forms.

+ Application.Run(new Form1()): Có chức năng khởi tạo và chạy form chính Form1, hiển thị giao diện chính.

- Tại Form1.Designer.cs

+ Ta có các thành phần giao diện (controls):

- button1: “SOAP serialize” - Tuần tự hóa đối tượng thành file SOAP.
- button2: “SOAP deserialize” - Giải tuần tự hóa từ file SOAP.
- button4: “Binary serialize” - Tuần tự hóa đối tượng thành file nhị phân.
- button3: “Binary deserialize” - Giải tuần tự hóa từ file nhị phân.
- button6: “XML serialize” - Tuần tự hóa đối tượng thành file XML.
- button5: “XML deserialize” - Giải tuần tự hóa từ file XML.



+ Phương thức InitializeComponent: Có chức năng thiết lập các thuộc của các nút (vị trí, kích thước và sự kiện). Ngoài ra, còn có SuspendLayout() và ResumeLayout() có chức năng tối ưu hóa quá trình vẽ giao diện.

- Tại Form1.cs

+ Sự kiện button1_Click (SOAP Serialize):

```
1 reference
23  private void button1_Click(object sender, EventArgs e)
24  {
25      company Vendor = new company();
26      company Buyer = new company();
27      lineItem Goods = new lineItem();
28      purchaseOrder po = new purchaseOrder();
29      Vendor.name = "Acme Inc.";
30      Buyer.name = "Wiley E. Coyote";
31      Goods.description = "anti-RoadRunner cannon";
32      Goods.quantity = 1;
33      Goods.cost = 599.99;
34      po.items = new lineItem[1];
35      po.items[0] = Goods;
36      po.buyer = Buyer;
37      po.vendor = Vendor;
38      SoapFormatter sf = new SoapFormatter();
39      FileStream fs = File.Create("..\\po.xml");
40      sf.Serialize(fs, po);
41      fs.Close();
42  }
43
```

- Tạo một đối tượng purchaseOrder với dữ liệu mẫu (Vendor: “Acme Inc”, Buyer: “Wiley E.Coyote”, Item: “anti-RoadRunner cannon”, Quantity: 1, Cost: 599.99).
- Bên cạnh đó, sử dụng SoapFormatter để tuần tự hóa đối tượng thành file SOAP (po.xml).
- Cuối cùng là ghi vào file và đóng luồng.

+ Sự kiện button2_Click (SOAP Deserialize):

```
1 reference
44  private void button2_Click(object sender, EventArgs e)
45  {
46      SoapFormatter sf = new SoapFormatter();
47      FileStream fs = File.OpenRead("..\\po.xml");
48      purchaseOrder po = (purchaseOrder)sf.Deserialize(fs);
49      fs.Close();
50      MessageBox.Show("Customer is " + po.buyer.name +
51          "\nVendor is " + po.vendor.name + ", phone is " + po.vendor.phone +
52          "\nItem is " + po.items[0].description + " has quantity " +
53          po.items[0].quantity.ToString() + ", has cost " + po.items[0].cost.ToString());
54  }
55
```

- Đọc file po.xml và giải tuần tự hóa thành đối tượng purchaseOrder.
- Hiển thị thông tin đối tượng qua MessageBox (tên khách hàng, nhà cung cấp, mô tả mặt hàng, số lượng, giá).

+ Sự kiện button4_Click (Binary Serialize):

```
1 reference
private void button4_Click(object sender, EventArgs e)
{
    company Vendor = new company();
    company Buyer = new company();
    lineItem Goods = new lineItem();
    purchaseOrder po = new purchaseOrder();
    Vendor.name = "Microsoft Inc.";
    Buyer.name = "Bill Gate";
    Goods.description = "anti-RoadRunner cannon";
    Goods.quantity = 1;
    Goods.cost = 599.99;
    po.items = new lineItem[1];
    po.items[0] = Goods;
    po.buyer = Buyer;
    po.vendor = Vendor;
    BinaryFormatter bf = new BinaryFormatter();
    FileStream fs = File.Create("..\\po_bin.txt");
    bf.Serialize(fs, po);
    fs.Close();
    MessageBox.Show("Serialize succesful!", "Info");
}
```

- Tạo một đối tượng purchaseOrder với dữ liệu khác (Vendor: “Microsoft Inc”, Buyer: “Bill Gates”, Item: “anti-RoadRunner cannon”).
- Sử dụng BinaryFormatter để tuần tự hóa đối tượng thành file nhị phân (po_bin.txt).
- Hiển thị thông báo “Serialize successful!”.

+ Sự kiện button3_Click (Binary Deserialize):

```
1 reference
private void button3_Click(object sender, EventArgs e)
{
    BinaryFormatter bf = new BinaryFormatter();
    FileStream fs = File.OpenRead("..\\po_bin.txt");
    purchaseOrder po = (purchaseOrder)bf.Deserialize(fs);
    fs.Close();
    MessageBox.Show("Customer is " + po.buyer.name);
}
```

- Đọc file po_bin.txt và giải tuần tự hóa thành đối tượng purchaseOrder.
- Hiển thị tên khách hàng qua MessageBox.

+ Sự kiện button6_Click (XML Serialize):

```
1 reference
private void button6_Click(object sender, EventArgs e)
{
    company Vendor = new company();
    company Buyer = new company();
    lineItem Goods = new lineItem();
    purchaseOrder po = new purchaseOrder();
    Vendor.name = "Microsoft Inc.";
    Buyer.name = "Paul Allen";
    Goods.description = "Microsoft Office 2010";
    Goods.quantity = 5;
    Goods.cost = 1500;
    po.items = new lineItem[1];
    po.items[0] = Goods;
    po.buyer = Buyer;
    po.vendor = Vendor;
    XmlSerializer xs = new XmlSerializer(po.GetType());
    FileStream fs = File.Create("..\\po1.xml");
    xs.Serialize(fs, po);
    fs.Close();
}
```

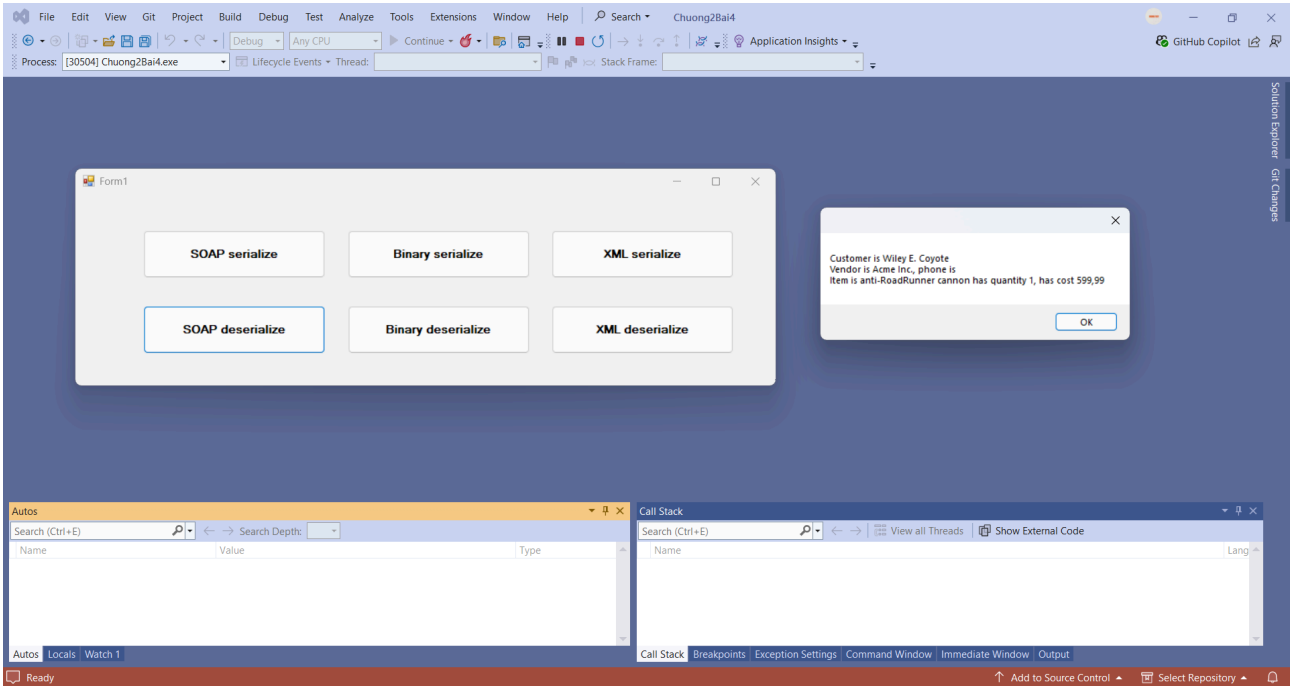
- Tạo một đối tượng purchaseOrder với dữ liệu khác (Vendor: "Microsoft Inc.", Buyer: "Paul Allen", Item: "Microsoft Office 2010", Quantity: 5, Cost: 1500).
- Sử dụng XmlSerializer để tuần tự hóa đối tượng thành file XML (po1.xml).
- Ghi vào file và đóng luồng.

+ Sự kiện button5_Click (XML Deserialize):

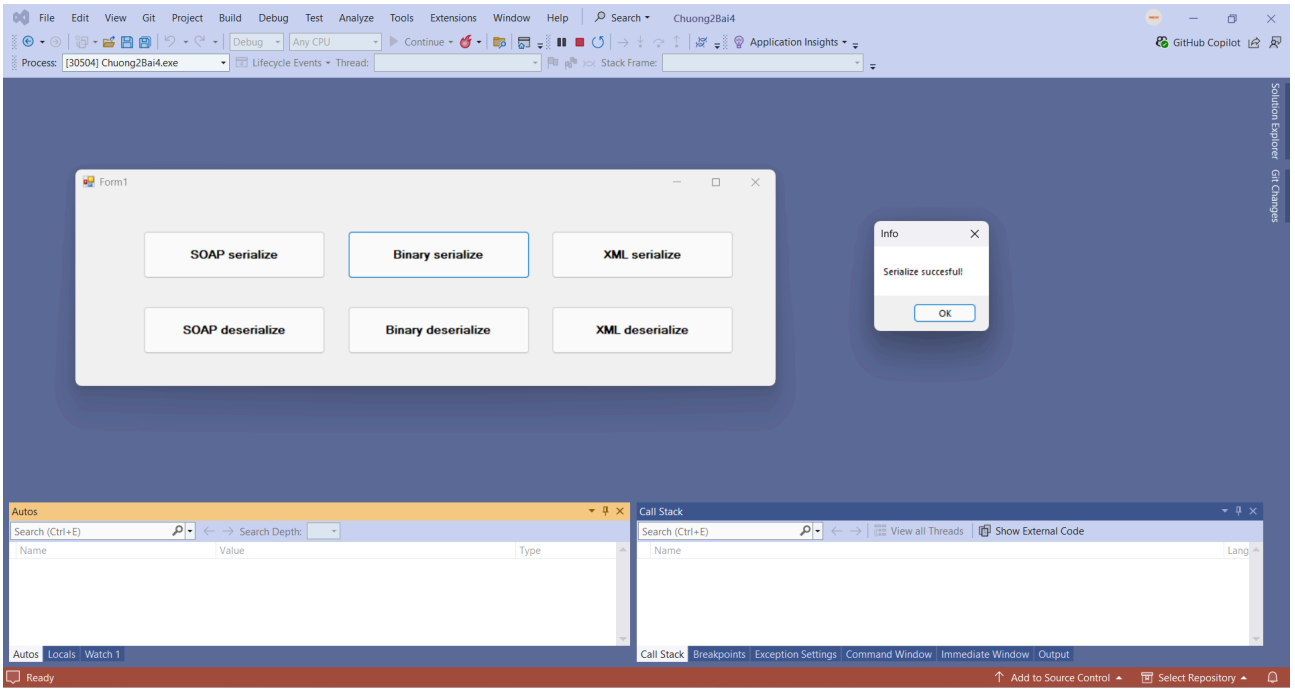
```
private void button5_Click(object sender, EventArgs e)
{
    purchaseOrder po = new purchaseOrder();
    XmlSerializer xs = new XmlSerializer(po.GetType());
    FileStream fs = File.OpenRead("..\\po1.xml");
    po = (purchaseOrder)xs.Deserialize(fs);
    fs.Close();
    MessageBox.Show("Customer is " + po.buyer.name +
        "\nVendor is " + po.vendor.name + ", phone is " + po.vendor.phone +
        "\nItem is " + po.items[0].description + " has quantity " +
        po.items[0].quantity.ToString() + ", has cost " + po.items[0].cost.ToString());
}
}
```

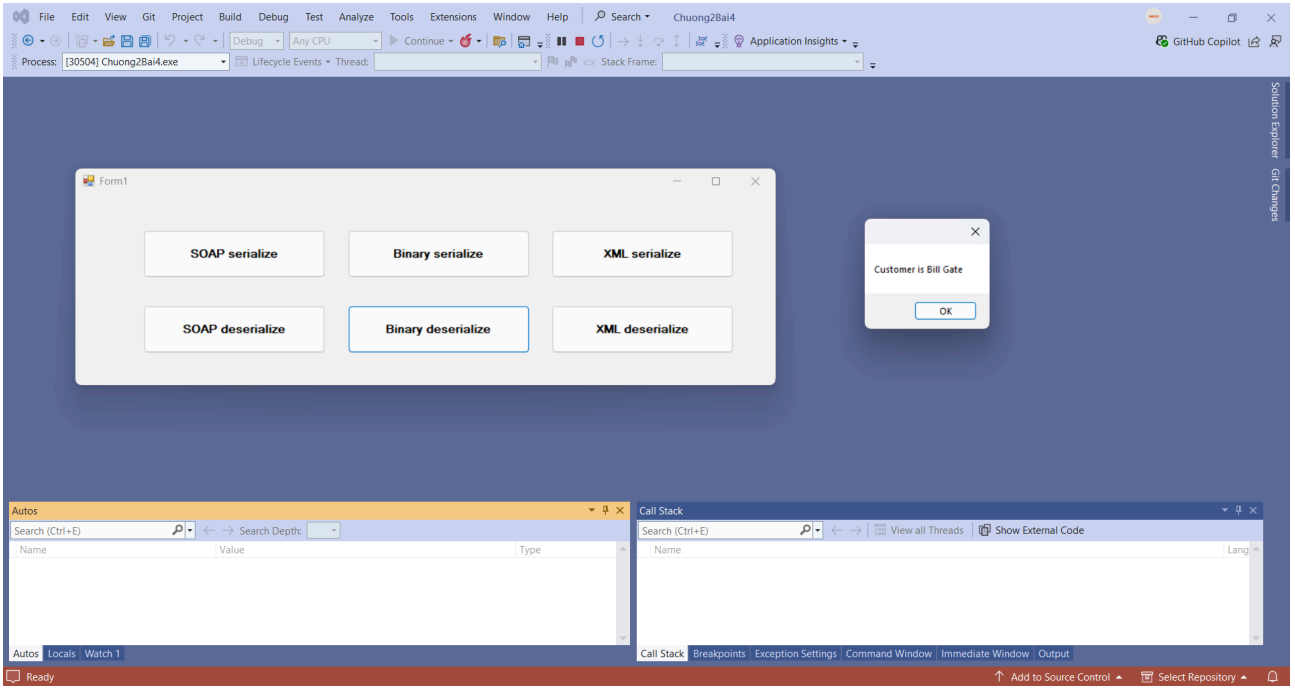
- Đọc file po1.xml và giải tuần tự hóa thành đối tượng purchaseOrder.
- Hiển thị thông tin đối tượng qua MessageBox.

- Cách thức vận hành của chương trình:
    + SOAP Serialization/Deserialization:
        - Serialize: Tạo đối tượng purchaseOrder, tuần tự hóa thành định dạng SOAP (XML có cấu trúc đặc biệt), lưu vào po.xml.
        - Deserialize: Đọc file SOAP, tái tạo đối tượng, và hiển thị thông tin.



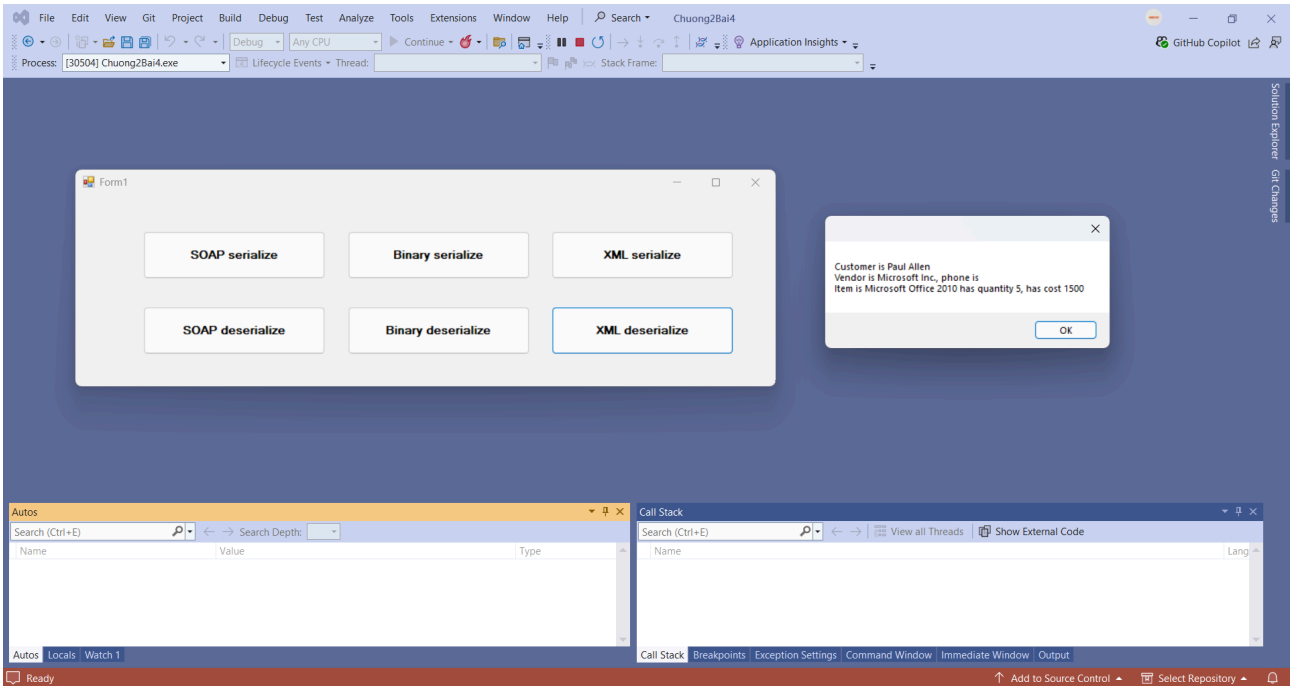
+ Binary Serialization/Deserialization:
    - Serialize: Tạo đối tượng purchaseOrder, tuần tự hóa thành định dạng nhị phân, lưu vào po_bin.txt.
    - Deserialize: Đọc file nhị phân, tái tạo đối tượng, và hiển thị tên khách hàng.





+ XML Serialization/Deserialization:

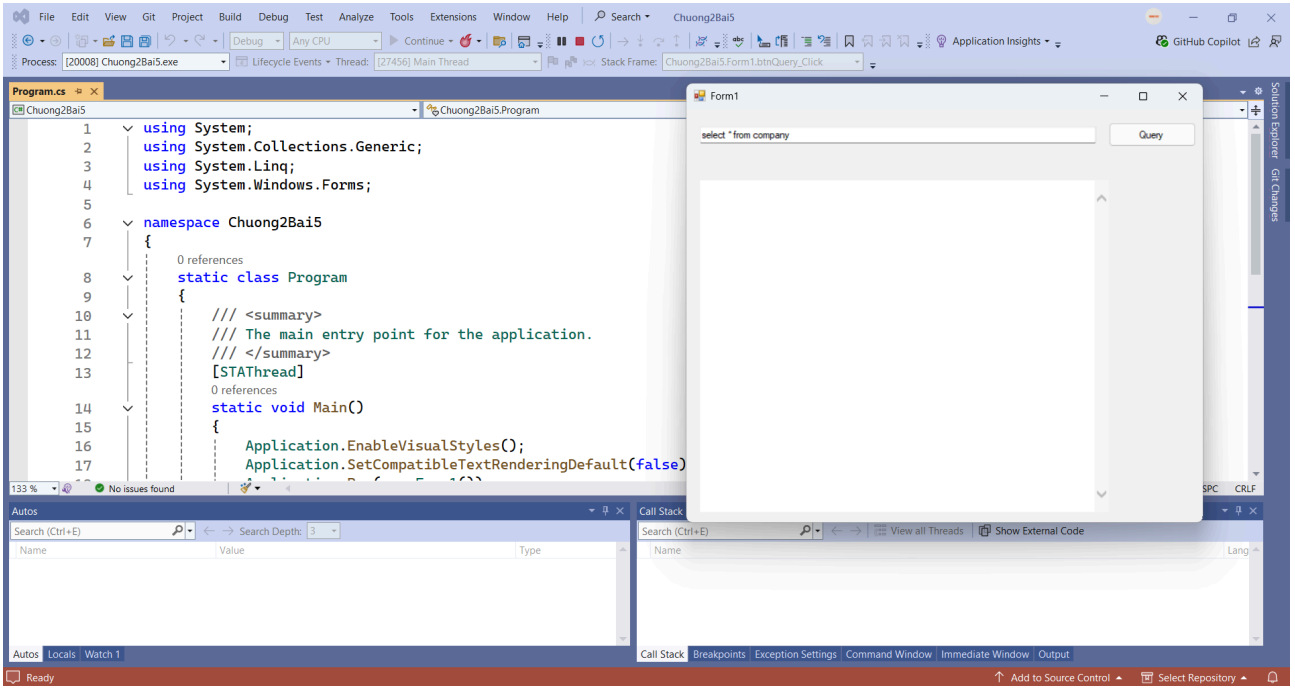
- Serialize: Tạo đối tượng purchaseOrder, tuần tự hóa thành định dạng XML, lưu vào po1.xml.
- Deserialize: Đọc file XML, tái tạo đối tượng, và hiển thị thông tin.



III. BÀI TẬP 05

- Tại Program.cs

+ Đây là chương trình chịu trách cho việc khởi động giao diện chính (Form1).



- Tại Form1.Designer.cs

+ Phương thức InitializeComponent: Có chức năng thiết lập các thuộc tính của các controls (vị trí, kích thước, sự kiện, v.v.). Ngoài ra, còn có SuspendLayout() và ResumeLayout() có chức năng tối ưu hóa quá trình vẽ giao diện.

- Tại Form1.cs

+ InitializeComponent() là phương thức khởi tạo được dùng để thiết lập giao diện từ Form1.Designer.cs

```
1 reference
public Form1()
{
    InitializeComponent();
}
```

+ Sự kiện Form1_Load:

```
1 reference
private void Form1_Load(object sender, EventArgs e)
{
    webBrowser.Navigate("about:blank", null, null, null);
}
```

- webBrowser.Navigate("about:blank", null, null, null): Có chức năng khởi động trình duyệt web với trang trống (about:blank) để tránh lỗi khi chưa có dữ liệu.

+ Sự kiện btnQuery_Click:

```
1 reference
private void btnQuery_Click(object sender, EventArgs e)
{
    //Access 2007 connection
    string szDSN = "Provider=Microsoft.ACE.OLEDB.16.0;" +
        "Data Source=D:/NT106.P22.ANTT/assignment/week02/Chuong2Bai5/Chuong2Bai5/Test1.mdb;" +
        "Persist Security Info=False";
    OleDbConnection DSN = new OleDbConnection(szDSN);
    XmlSerializer xs = new XmlSerializer(typeof(DataSet));
    DataSet ds = new DataSet();
    DSN.Open();
    OleDbCommand odbc = new OleDbCommand(tbSQL.Text, DSN);
    OleDbDataAdapter odda = new OleDbDataAdapter(odbc);
    odda.Fill(ds, "sql");
    TextWriter tw = new StreamWriter("..\\sql.xml");
    xs.Serialize(tw, ds);
    tw.Close();
    DSN.Close();
    webBrowser.Navigate("D:/NT106.P22.ANTT/assignment/week02/Chuong2Bai5/Chuong2Bai5/bin/sql.xml",
                                                                                    null, null, null);
}
}
}
```

- Kết nối cơ sở dữ liệu:
  - string szDSN: Chuỗi kết nối đến cơ sở dữ liệu Access 2007, sử dụng Microsoft.ACE.OLEDB.16.0 với đường dẫn file D:/NT106.P22.ANTT/assignment/week02/Chuong2Bai5/Chuong2Bai5/Test1.mdb.
  - OleDbConnection DSN = new OleDbConnection(szDSN): Có chức năng tạo kết nối đến cơ sở dữ liệu.

```
//Access 2007 connection
string szDSN = "Provider=Microsoft.ACE.OLEDB.16.0;" +
    "Data Source=D:/NT106.P22.ANTT/assignment/week02/Chuong2Bai5/Chuong2Bai5/Test1.mdb;" +
    "Persist Security Info=False";
OleDbConnection DSN = new OleDbConnection(szDSN);
```

- Thực hiện truy vấn:
  - DSN.Open(): Mở kết nối.
  - OleDbCommand odbc = new OleDbCommand(tbSQL.Text, DSN): Có chức năng tạo lệnh SQL từ nội dung ô tbSQL.
  - OleDbDataAdapter odda = new OleDbDataAdapter(odbc): Có nhiệm vụ thực hiện việc tạo adapter để điền dữ liệu vào DataSet.
  - odda.Fill(ds, "sql"): Có chức năng thực thi truy vấn và điền kết quả vào DataSet với tên bảng "sql".

```
DSN.Open();
OleDbCommand odbc = new OleDbCommand(tbSQL.Text, DSN);
OleDbDataAdapter odda = new OleDbDataAdapter(odbc);
odda.Fill(ds, "sql");
```

- Tuần tự hóa kết quả:
    - XmlSerializer xs = new XmlSerializer(typeof(DataSet)): Thực hiện tạo đối tượng để tuần tự hóa DataSet thành XML.
    - TextWriter tw = new StreamWriter("..\\sql.xml"): Có nhiệm vụ tạo file sql.xml trong thư mục cha.
    - xs.Serialize(tw, ds): Ghi DataSet thành file XML.
    - tw.Close();: Đóng luồng ghi.

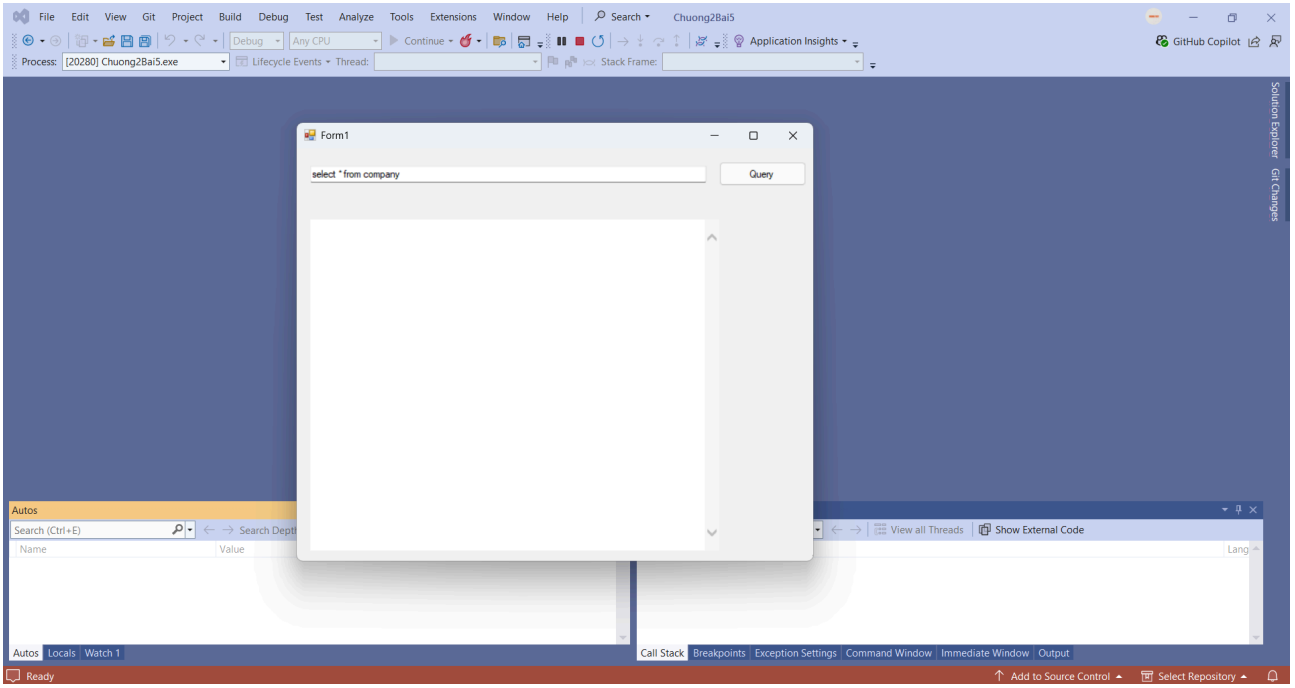
```
41          XmlSerializer xs = new XmlSerializer(typeof(DataSet));
42          TextWriter tw = new StreamWriter("..\\sql.xml");
43          xs.Serialize(tw, ds);
44          tw.Close();
```

- Hiển thị kết quả:
    - DSN.Close(): Đóng kết nối cơ sở dữ liệu.
    - webBrowser.Navigate("D:/Baitap/C-Sharp/Network Programming/Chuong2Bai5/Chuong2Bai5/bin/sql.xml", null, null, null): Điều hướng trình duyệt web đến file XML để hiển thị nội dung.

```
43          DSN.Close();
44          webBrowser.Navigate("D:/NT106.P22.ANTT/assignment/week02/Chuong2Bai5/Chuong2Bai5/bin/sql.xml",
45                                                                                  null, null, null);
46      }
```

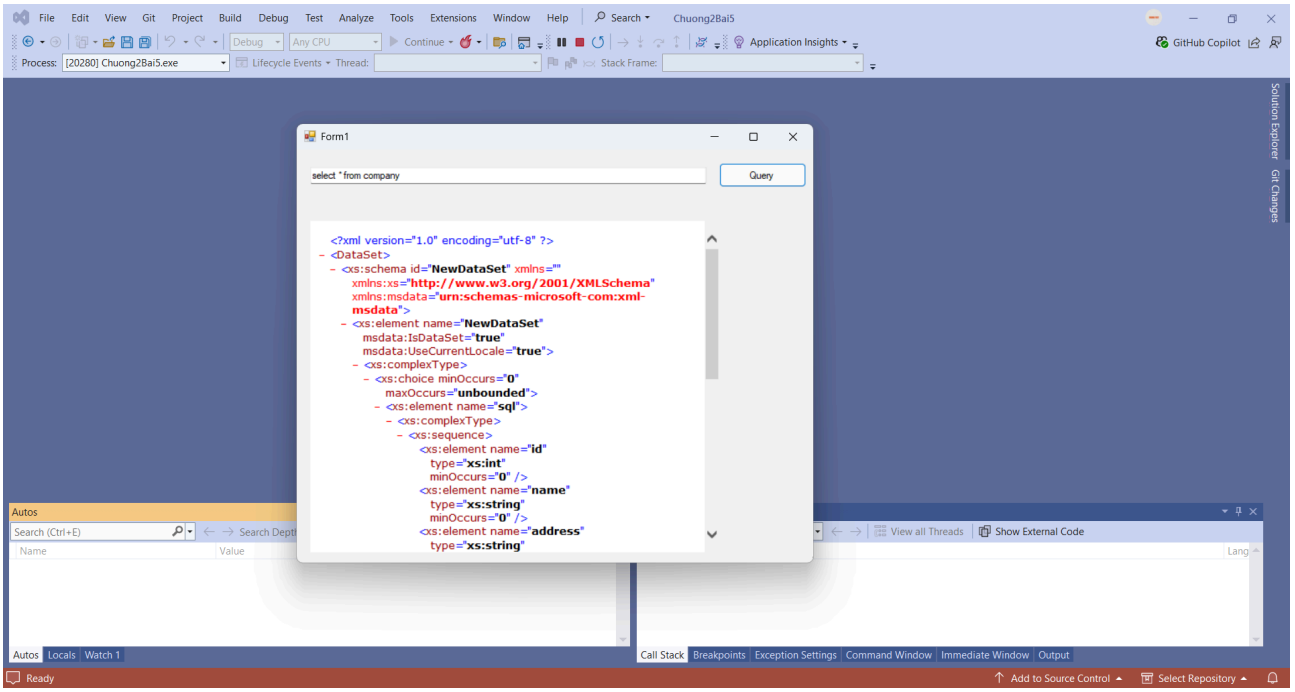
- Cách thức vận hành chương trình

+ Khởi động chương trình: Form hiển thị với ô tbSQL chứa câu lệnh mặc định "select * from company", nút "Query", và trình duyệt web trống.



+ Thực hiện truy vấn: Người dùng nhấn "Query", chương trình kết nối đến file cơ sở dữ liệu Access Test1.mdb. Sau đó, chương trình sẽ thực thi câu lệnh SQL từ tbSQL, lấy dữ liệu từ bảng (ví dụ: company), và lưu vào DataSet.

+ Xuất kết quả thành XML: Kết quả từ DataSet được tuần tự hóa thành file sql.xml trong thư mục cha. File XML được tạo với cấu trúc mô tả dữ liệu truy vấn.



+ Hiển thị: Trình duyệt webBrowser mở file sql.xml để hiển thị nội dung dưới dạng định dạng XML có thể đọc được. Bên dưới là nội dung của file XML sau khi thực hiện xong chương trình.

```
<?xml version="1.0" encoding="utf-8" ?>
- <DataSet>
- <xs:schema id="NewDataSet" xmlns=""
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:msdata="urn:schemas-microsoft-com:xml-msdata">
- <xs:element name="NewDataSet" msdata:IsDataSet="true"
msdata:UseCurrentLocale="true">
- <xs:complexType>
- <xs:choice minOccurs="0" maxOccurs="unbounded">
- <xs:element name="sql">
- <xs:complexType>
- <xs:sequence>
<xs:element name="id" type="xs:int" minOccurs="0"
/>
<xs:element name="name" type="xs:string"
minOccurs="0" />
```

```
  <xs:element name="address" type="xs:string"
minOccurs="0" />
  </xs:sequence>
  </xs:complexType>
  </xs:element>
  </xs:choice>
  </xs:complexType>
  </xs:element>
  </xs:schema>
- <diffgr:diffgram
xmlns:msdata="urn:schemas-microsoft-com:xml-msdata"
xmlns:diffgr="urn:schemas-microsoft-com:xml-diffgram-v1">
- <NewDataSet>
- <sql diffgr:id="sql1" msdata:rowOrder="0">
  <id>1</id>
  <name>Wiley E coyote</name>
  <address>sandy desert</address>
  </sql>
- <sql diffgr:id="sql2" msdata:rowOrder="1">
  <id>2</id>
  <name>Acme corp.</name>
  <address>big city</address>
  </sql>
  </NewDataSet>
  </diffgr:diffgram>
  </DataSet>
```

--Hết--