

CHƯƠNG 7: ĐIỀU KHIỂN DIALOG VÀ PHƯƠNG THỨC MESSAGE

7.1. Lớp MessageBox

Lớp *MessageBox* giúp hiển thị một hộp thoại trên màn hình. Khi hộp thoại xuất hiện thì người dùng bắt buộc phải thao tác trên hộp thoại trước thì mới có tiếp tục thực hiện công việc trên ứng dụng. Việc hiển thị như thế hữu ích khi muốn cảnh báo một lỗi hoặc hướng dẫn cho người dùng. Sử dụng lớp *MessageBox* thì cần khai báo không gian tên:

System.Windows.Forms

Để hiển thị hộp thoại, người dùng chỉ cần sử dụng phương thức tĩnh `Show()` của lớp *MessageBox* mà không cần phải có bất cứ khai báo nào để tạo đối tượng lớp *MessageBox*.

➤ Phương thức `Show()`:

```
DialogResult Show(string text, string caption,  
                  MessageBoxButtons buttons,  
                  MessageBoxIcon icon,  
                  MessageBoxDefaultButton defaultButton,  
                  MessageBoxOptions options);
```

- *Text*: Chuỗi hiển thị trên hộp thoại *MessageBox*
- *Caption*: Chuỗi hiển thị trên thanh titlebar của hộp thoại *MessageBox*
- Kiểu liệt kê *MessageBoxIcon* gồm các giá trị xác định biểu tượng hiển thị trên *MessageBox*:

```
public enum MessageBoxIcon  
{  
    Asterisk = 0x40,  
    Error = 0x10,  
    Exclamation = 0x30,  
    Hand = 0x10,  
    Information = 0x40,  
    None = 0,  
    Question = 0x20,  
    Stop = 0x10,  
    Warning = 0x30  
}
```

- Kiểu liệt kê *MessageBoxDefaultButtons* gồm các giá trị xác định nút được *Focus* trên *MessageBox*:

```
public enum MessageBoxDefaultButton
{
    Button1 = 0,
    Button2 = 0x100,
    Button3 = 0x200
}
```

- Kiểu liệt kê *MessageBoxButtons* gồm các giá trị xác định nút hiển thị trên *MessageBox*:

```
public enum MessageBoxButtons
{
    OK,
    OKCancel,
    AbortRetryIgnore,
    YesNoCancel,
    YesNo,
    RetryCancel
}
```

- Kiểu liệt kê *MessageBoxOption* gồm các giá trị xác định nút hiển thị bên phải hoặc bên trái của *MessageBox*:

```
public enum MessageBoxOptions
{
    DefaultDesktopOnly = 0x20000,
    RightAlign = 0x80000,
    RtlReading = 0x100000,
    ServiceNotification = 0x200000
}
```

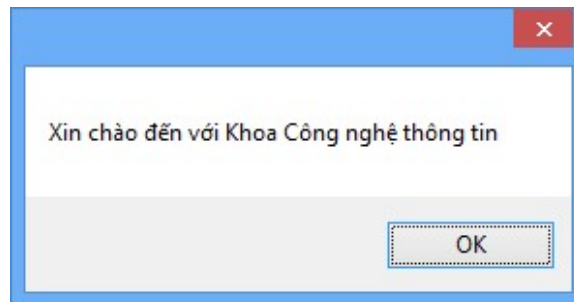
Lưu ý: Khi hộp thoại hiển thị từ phương thức *Show()* có sử dụng tham số *MessageBoxButton* thì phương thức này sẽ trả về một giá trị có kiểu *DialogResult*. *DialogResult* không phải là lớp mà là một kiểu liệt kê bao gồm các giá trị:

```
public enum DialogResult
{
    None,
    OK,
    Cancel,
    Abort,
    Retry,
    Ignore,
    Yes,
    No
}
```

Lập trình viên có thể sử dụng *DialogResult* để xác định xem người dùng đã nhấn nút nào trên hộp thoại *MessageBox*.

Ví dụ 7.1: Hiển thị các dạng khác nhau của hộp thoại *MessageBox*.

- Hiển thị hộp thoại hiển thị dòng thông báo đơn giản như hình 7.1:

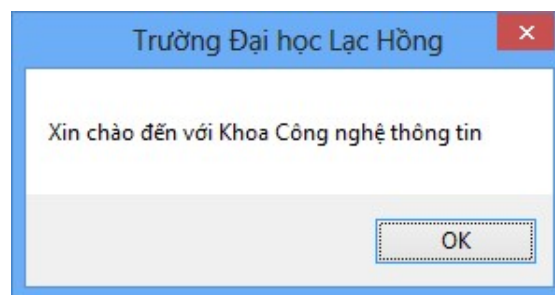


Hình 7.1: Hộp thoại *MessageBox* đơn giản

Mã lệnh:

```
MessageBox.Show("Xin chào đến với Khoa Công nghệ thông tin");
```

- Hiển thị hộp thoại với dòng thông báo trên *MessageBox* và dòng tiêu đề trên titlebar của hộp thoại như hình 7.2.

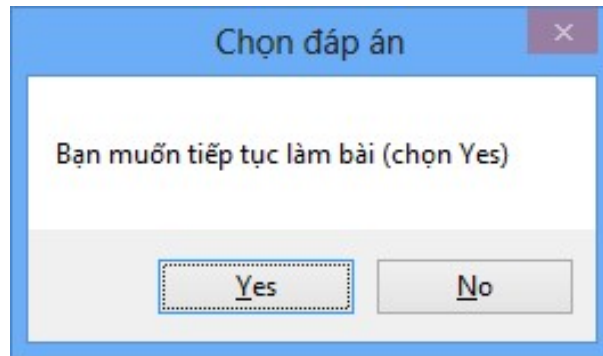


Hình 7.2: Hộp thoại *MessageBox* với dòng tiêu đề trên titlebar

Mã lệnh:

```
MessageBox.Show("Xin chào đến với Khoa Công nghệ thông tin",  
                "Trường Đại học Lạc Hồng");
```

- Hiện thị hộp thoại với hai nút Yes/ No như hình 7.3. Muốn hiện thị nút trong hộp thoại cần sử dụng tham số thuộc kiểu *MessageBoxButton*, đồng thời để biết người dùng nhấn nút Yes hay No thì cần sử dụng biến thuộc kiểu *DialogResult* để nhận kết quả trả về từ hộp thoại.

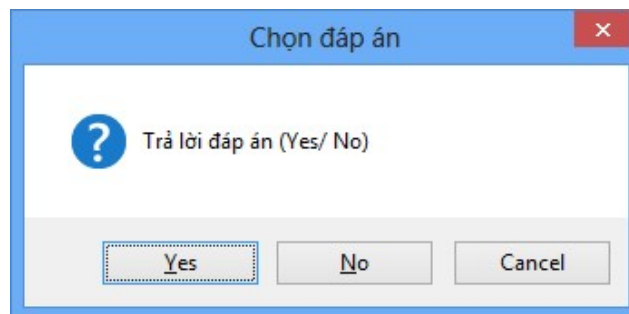


Hình 7.3: Hộp thoại *MessageBox* với hai nút Yes/ No

Mã lệnh:

```
DialogResult r = MessageBox.Show("Bạn muốn tiếp tục làm bài",  
                                "Chọn đáp án",  
                                MessageBoxButtons.YesNo);  
if (r == DialogResult.Yes)  
    MessageBox.Show("Bạn đã chọn Yes");  
else  
    MessageBox.Show("Bạn đã chọn No");
```

- Hiện thị hộp thoại có 3 nút Yes/ No/ Cancel và có thêm biểu tượng dạng câu hỏi cạnh chuỗi hiển thị trên *MessageBox* như hình 7.4.

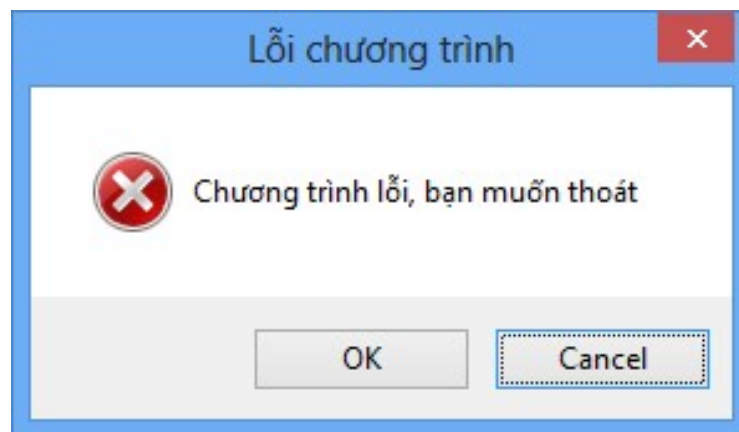


Hình 7.4: Hộp thoại *MessageBox* với biểu tượng dạng câu hỏi

Mã lệnh:

```
DialogResult r = MessageBox.Show("Trả lời đáp án (Yes/ No)",  
                                "Chọn đáp án",  
                                MessageBoxButtons.YesNoCancel,  
                                MessageBoxIcon.Question);  
  
if (r == DialogResult.Yes)  
    MessageBox.Show("Bạn đã chọn Yes");  
else  
    if(r==DialogResult.No)  
        MessageBox.Show("Bạn đã chọn No");  
    else  
        MessageBox.Show("Bạn đã thoát chương trình");
```

- Hiện thị hộp thoại có 2 nút Ok/ Cancel và có thêm biểu tượng dạng lỗi cạnh chuỗi hiển thị trên *MessageBox* như hình 7.5. Nút mặc định được Focus khi hộp thoại hiển thị là nút Cancel.



Hình 7.5: Hộp thoại *MessageBox* báo lỗi

Mã lệnh:

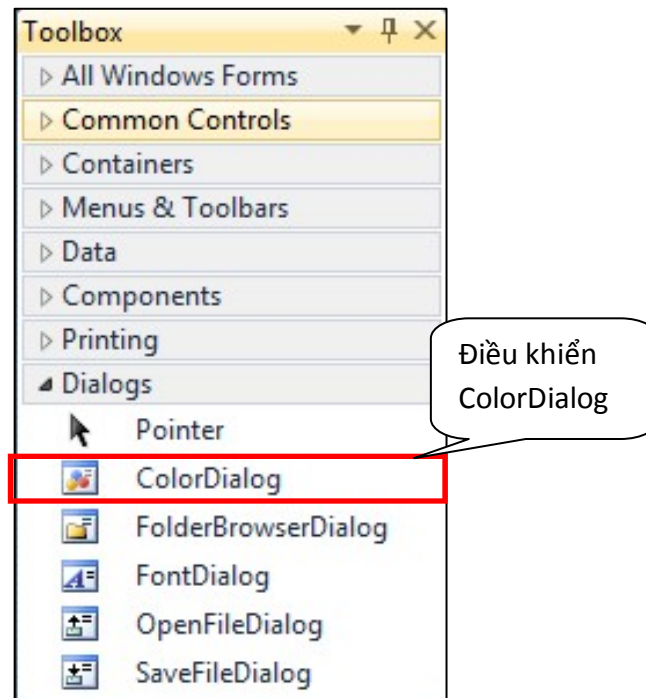
```
DialogResult r = MessageBox.Show("Chương trình lỗi, bạn muốn  
                                thoát", "Lỗi chương trình",  
                                MessageBoxButtons.OKCancel,  
                                MessageBoxIcon.Error,  
                                MessageBoxDefaultButton.Button2);  
  
if (r == DialogResult.OK)  
    MessageBox.Show("Bạn đã chọn OK, thoát chương trình");  
else  
    if(r==DialogResult.No)  
        MessageBox.Show("Bạn đã chọn Cancel, tiếp tục chạy  
                        chương trình");
```

7.2. Điều khiển ColorDialog

Để thiết lập màu cho các điều khiển, C# hỗ trợ lớp *ColorDialog* để người dùng sử dụng và hiển thị trực quan qua một hộp thoại gọi là *ColorDialog Box*.

ColorDialog Box chứa một danh sách các màu sắc được xác định cho hệ thống hiển thị. Người dùng có thể lựa chọn hoặc tạo ra một màu sắc đặc biệt từ danh sách, sau đó áp dụng khi thoát khỏi hộp thoại.

ColorDialog nằm trong nhóm Dialog của cửa sổ Toolbox như hình 7.6.



Hình 7.6: Điều khiển ColorDialog trong cửa sổ Toolbox

Lập trình có thể tạo ra đối tượng *ColorDialog* bằng cách kéo điều khiển vào form từ cửa sổ Toolbox hoặc sử dụng mã lệnh tạo đối tượng và hiển thị *ColorDialog Box* bằng cách gọi phương thức *ShowDialog()* để hiển thị hộp thoại như hình 7.7.

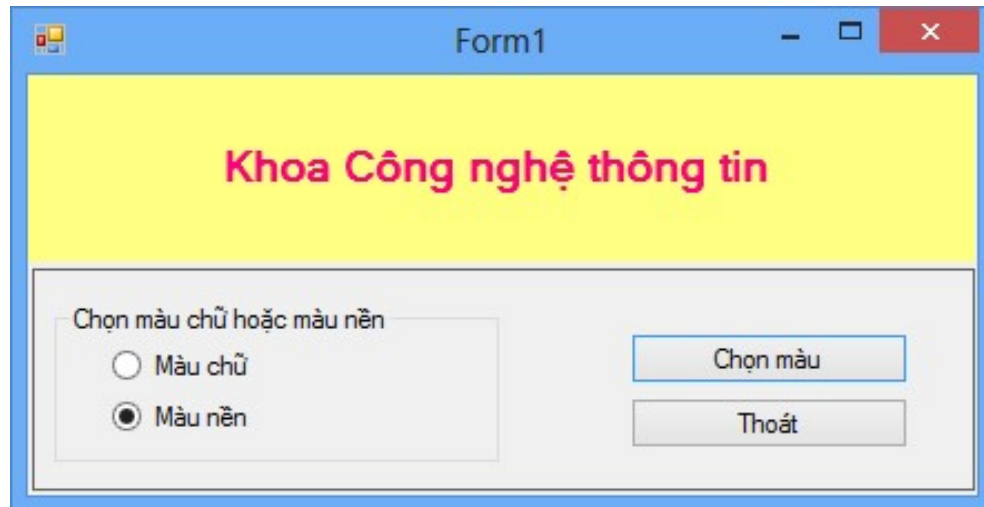
Mã lệnh:

```
ColorDialog clbox= new ColorDialog();  
clbox.ShowDialog();
```



Hình 7.7: Giao diện ColorDialog Box

Ví dụ 7.2: Viết chương trình chọn màu chữ và màu nền cho Label có giao diện như hình 7.8.



Hình 7.8: Giao diện form lựa chọn màu cho Label

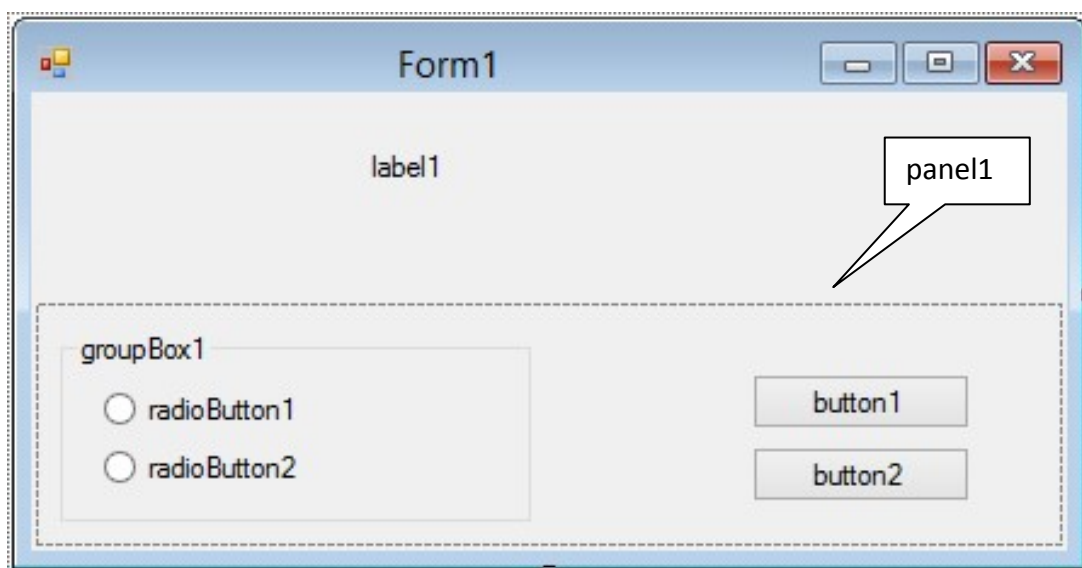
Yêu cầu:

Người dùng chọn thay đổi màu nền hoặc màu chữ của Label có dòng chữ “Khoa Công nghệ thông tin” bằng cách chọn một trong hai *RadioButton*: “Màu chữ”, “Màu nền”.

Sau khi đã có lựa chọn, người dùng nhấn vào nút “Chọn màu” để hiện thị *ColorDialog Box* và chọn màu muốn thay đổi cho *Label*.

Hướng dẫn:

Bước 1: Thiết kế giao diện ban đầu: Thêm các điều khiển *Label*, *Button*, *Panel*, *GroupBox* và *RadioButton* từ cửa sổ Toolbox vào form như hình 7.9.



Hình 7.9: Giao diện form sau khi thêm điều khiển

Bước 2: Thiết lập giá trị thuộc tính cho điều khiển trong cửa sổ Properties

- label1:
 - Thuộc tính *Text*: “Khoa Công nghệ thông tin”
 - Thuộc tính *Font Size*: 14
 - Thuộc tính *Font style*: Bold
 - Thuộc tính *AutoSize*: False
 - Thuộc tính *Size*: 428, 86
 - Thuộc tính *TextAlign*: MiddleCenter
- panel1:
 - Thuộc tính *BorderStyle*: FixedSingle
- groupBox1:
 - Thuộc tính *Text*: “Chọn màu chữ hoặc màu nền”
- radioButton1:
 - Thuộc tính *Text*: “Màu chữ”
 - Thuộc tính *Name*: radMauChu
- radioButton2:
 - Thuộc tính *Text*: “Màu nền”

Thuộc tính *Name*: radMauNen

- button1:

Thuộc tính *Text*: “Chọn màu”

Thuộc tính *Name*: btnChonMau

- button2:

Thuộc tính *Text*: “Thoát”

Thuộc tính *Name*: btnThoat

Bước 3: Viết mã lệnh cho các điều khiển

- Sự kiện *Click* nút btnThoat:

```
private void btnThoat_Click(object sender, EventArgs e)
{
    Close();
}
```

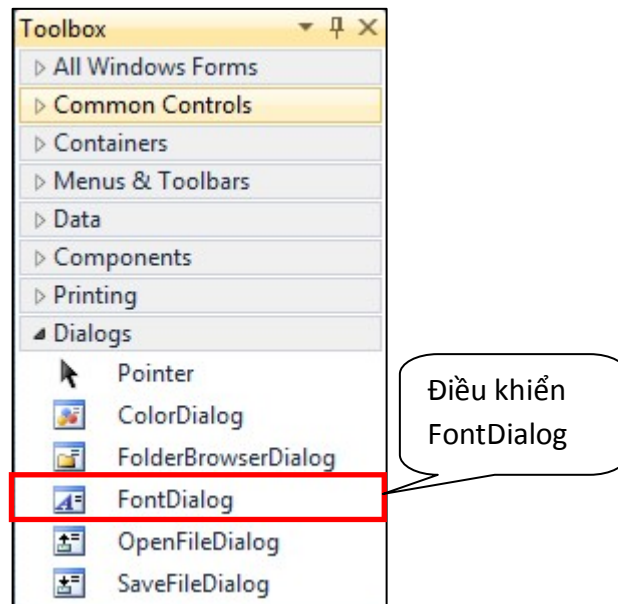
- Sự kiện *Click* nút btnChonMau:

```
private void btnChonMau_Click(object sender, EventArgs e)
{
    ColorDialog clDLog = new ColorDialog();
    DialogResult rs = clDLog.ShowDialog();
    if (rs == DialogResult.OK)
    {
        if (radMauChu.Checked == true)
        {
            lblHienThi.ForeColor = clDLog.Color;
        }
        if (radMauNen.Checked == true)
        {
            lblHienThi.BackColor = clDLog.Color;
        }
    }
}
```

7.3. Điều khiển FontDialog

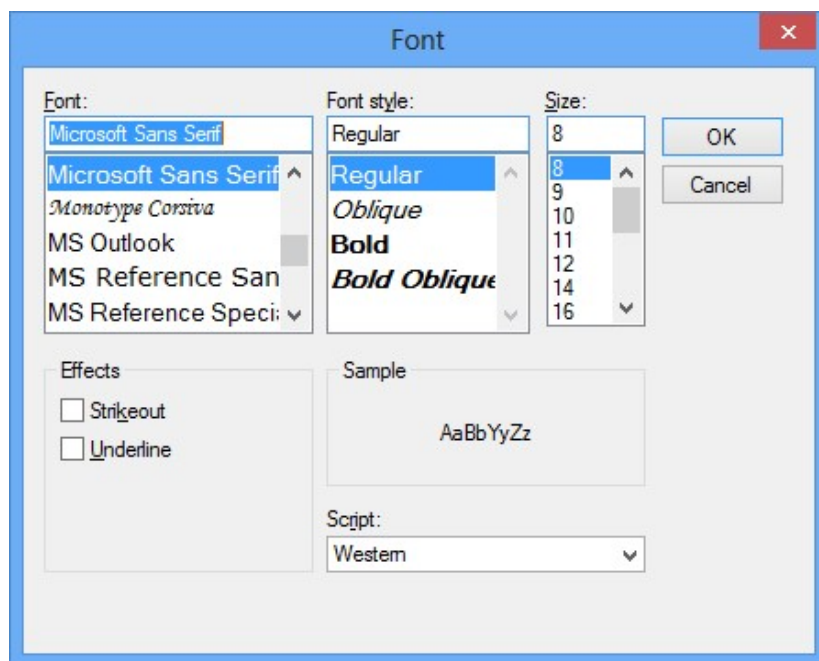
Tương tự như *ColorDialog Box*, để thiết lập font chữ hiển thị, C# cũng hỗ trợ hộp thoại gọi là *FontDialog Box*. *FontDialog Box* sẽ hiển thị một danh sách các font chữ hiện đang được cài đặt trên hệ thống; Cho phép người dùng lựa chọn các thuộc tính cho một font chữ hợp lý, như kiểu font chữ, kích cỡ chữ, hiệu ứng,

FontDialog nằm trong nhóm Dialogs của cửa sổ Toolbox như hình 7.10.



Hình 7.10: Điều khiển *FontDialog* trong cửa sổ Toolbox

Lập trình có thể tạo ra đối tượng *FontDialog* bằng cách kéo điều khiển vào form từ cửa sổ Toolbox hoặc sử dụng mã lệnh tạo đối tượng và hiển thị *FontDialog Box* bằng cách gọi phương thức `ShowDialog()` để hiển thị hộp thoại như hình 7.11.

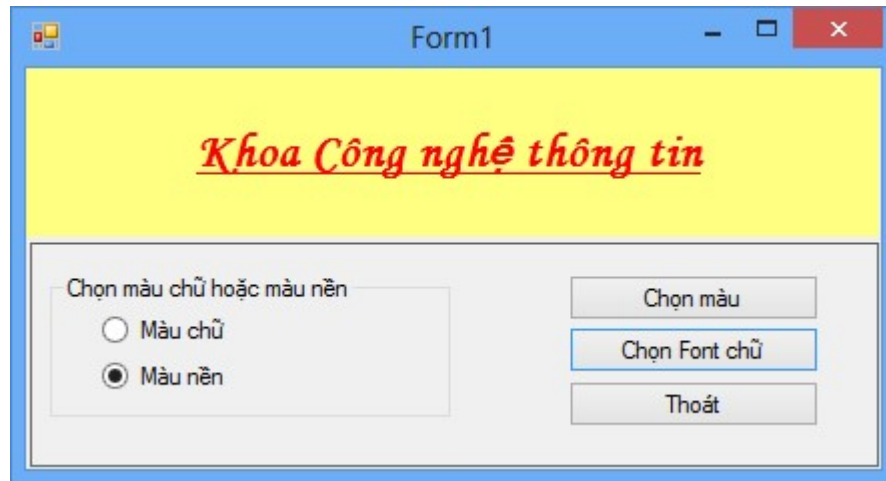


Hình 7.11: Giao diện *FontDialog Box*

Mã lệnh hiển thị *FontDialog Box*:

```
FontDialog fontbox= new FontDialog();  
fontbox.ShowDialog();
```

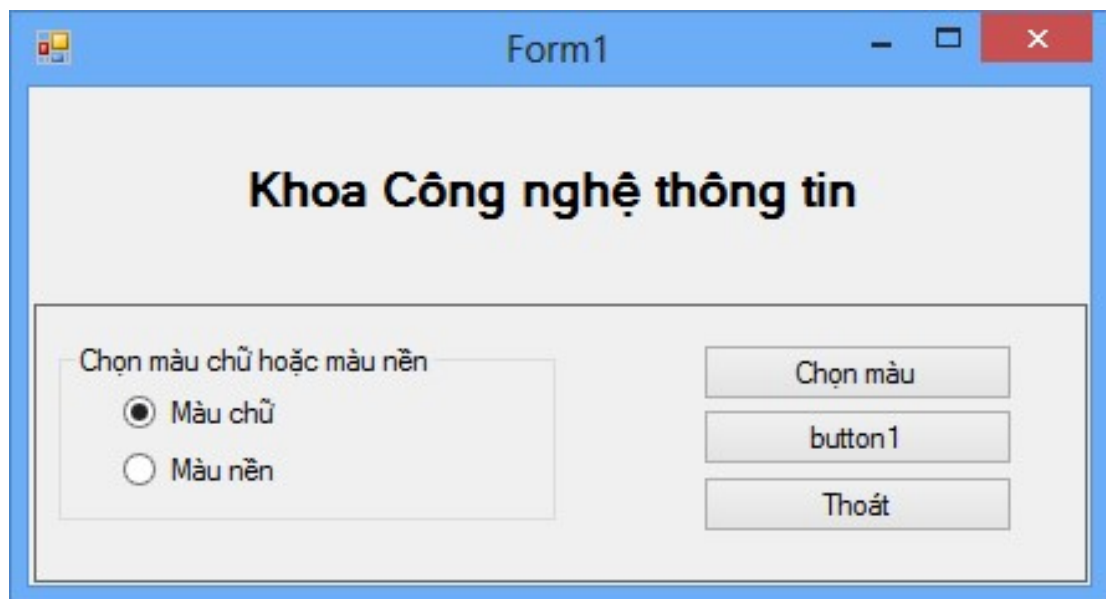
Ví dụ 7.3: Dựa vào ví dụ 7.2. Thêm nút lệnh “Chọn Font chữ” như hình 7.12



Hình 7.12: Giao diện form sau khi thêm nút “Chọn Font chữ”

Yêu cầu: Khi người dùng nhấn vào nút “Chọn Font chữ” sẽ hiển thị hộp thoại *FontDialog Box* cho phép người dùng thay đổi font chữ, kiểu chữ và kích thước của dòng chữ “Khoa Công nghệ thông tin”.

Bước 1: Thêm một điều khiển *Button* từ cửa sổ Toolbox vào frm như hình 7.13



Hình 7.13: Giao diện form sau khi thêm Button

Bước 2: Thiết lập giá trị thuộc tính cho nút button1:

Thuộc tính *Name*: btnChonFont

Thuộc tính *Text*: “Chọn Font chữ”

Bước 3: Viết mã lệnh cho sự kiện.

- Khai báo biến fontbox có kiểu *FontDialog*:

```
FontDialog fontbox = new FontDialog();
```

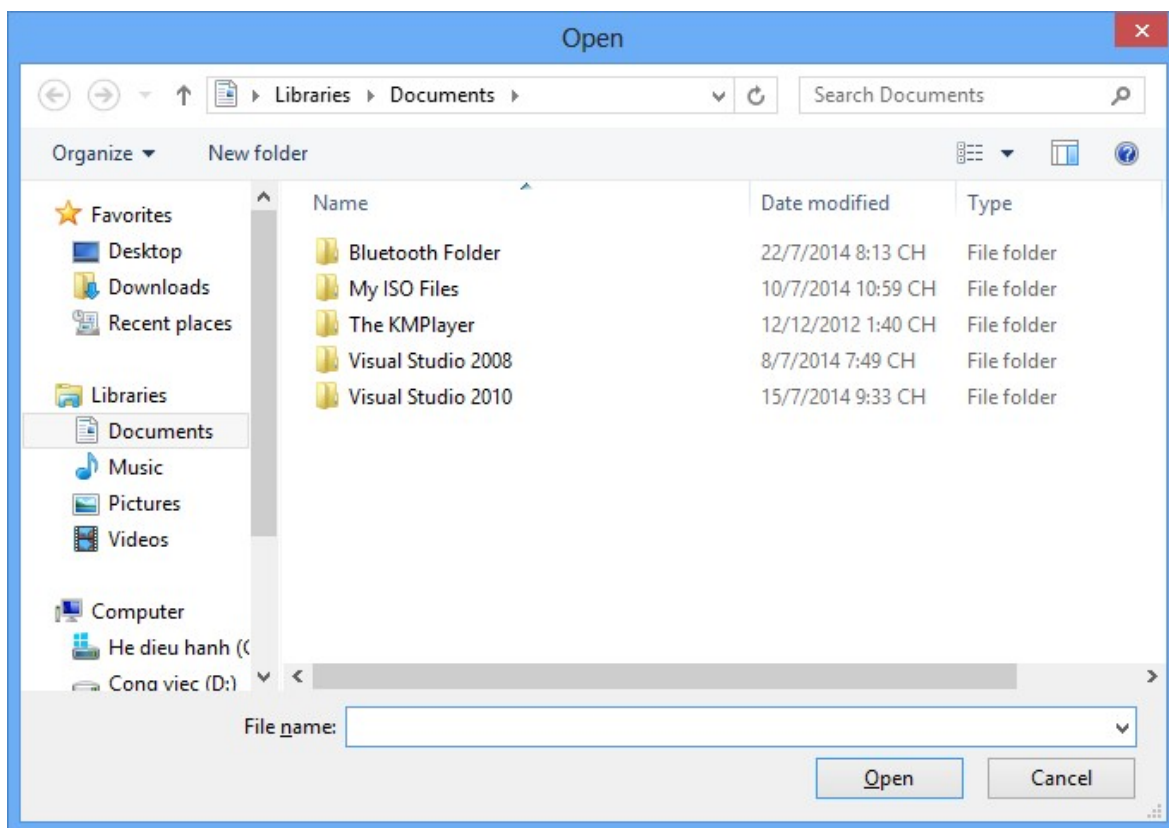
- Sự kiện *Click* nút btnChonFont

```
private void btnChonFont_Click(object sender, EventArgs e)
{
    DialogResult rs = fontbox.ShowDialog();
    if (rs == DialogResult.OK)
    {
        lblHienThi.Font = fontbox.Font;
    }
}
```

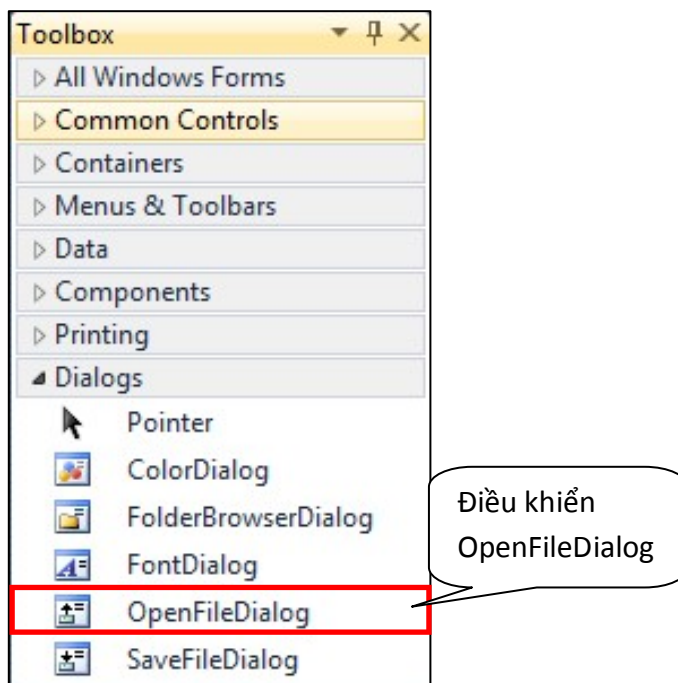
7.4. Điều khiển OpenFileDialog

Điều khiển *OpenFileDialog* cho phép hiển thị hộp thoại *OpenFileDialog* để lấy về ổ đĩa, tên thư mục, tên tập tin đối với một tập tin hay thư mục đang tồn tại. Thông thường các ứng dụng Windows Forms sử dụng hộp thoại *OpenFileDialog* để mở một tập tin nào đó lưu trên bộ nhớ máy tính.

Hộp thoại *OpenFileDialog* có giao diện như hình 7.14 và đặt trong nhóm Dialogs của cửa sổ Toolbox như hình 7.15.



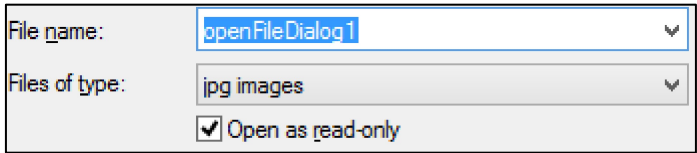
Hình 7.14: Hộp thoại OpenFileDialog



Hình 7.15: Điều khiển OpenFileDialog trong cửa sổ Toolbox

- Một số thuộc tính thường dùng của *OpenFileDialog*:

Thuộc tính	Mô tả
<i>Title</i>	Thiết lập chuỗi hiển thị trên titlebar của hộp thoại OpenFileDialog
<i>Filter</i>	<p>Thuộc tính nhận giá trị là một chuỗi. Chuỗi này chứa các đuôi mở rộng của tập tin, giúp lọc ra trên hộp thoại OpenFileDialog chỉ hiển thị đúng loại đuôi có trong chuỗi.</p> <p>Ví dụ: Thiết lập chuỗi chỉ hiển thị file hình ảnh có đuôi JPG và đuôi GIF:</p> <p style="text-align: center;">“jpg images *.jpg gif images *.gif”</p> <p>Thiết lập chuỗi hiển thị tất cả những gì có trong thư mục trên hộp thoại:</p> <p style="text-align: center;">“All *.*”</p>
<i>FileName</i>	Thuộc tính trả về đường dẫn cùng tên tập tin được chọn
<i>FileNames</i>	<p>Thuộc tính trả về một mảng các đường dẫn cùng tên tập tin được chọn.</p> <p>Lưu ý: Thuộc tính thường được sử dụng khi thuộc tính MultiSelect được thiết lập là True</p>
<i>FilterIndex</i>	<p>Thuộc tính sử dụng cùng với thuộc tính Filter. Có ý nghĩa sẽ ưu tiên lọc loại tập tin nào trước trong chuỗi Filter.</p> <p>Ví dụ: Chuỗi Filter lọc hai loại tập tin JPG và GIF:</p> <p style="text-align: center;">“jpg images *.jpg gif images *.gif”</p> <p>Nếu muốn hộp thoại khi hiển thị lọc hình có đuôi JPG đầu tiên thiết lập thuộc tính FilterIndex có giá trị 1.</p> <p>Nếu muốn hộp thoại khi hiển thị lọc hình có đuôi GIF đầu tiên thiết lập thuộc tính FilterIndex có giá trị 2.</p>
<i>InitialDirectory</i>	Giá trị thuộc tính là một đường dẫn đến một thư mục hoặc ổ đĩa. Khi hộp thoại OpenFileDialog được mở lên sẽ hiển thị nội dung của đường dẫn này đầu tiên.
<i>RestoreDirectory</i>	Mang hai giá trị True hoặc False, cho biết hộp thoại có trả lại đường dẫn thư mục vừa mở trước đó.

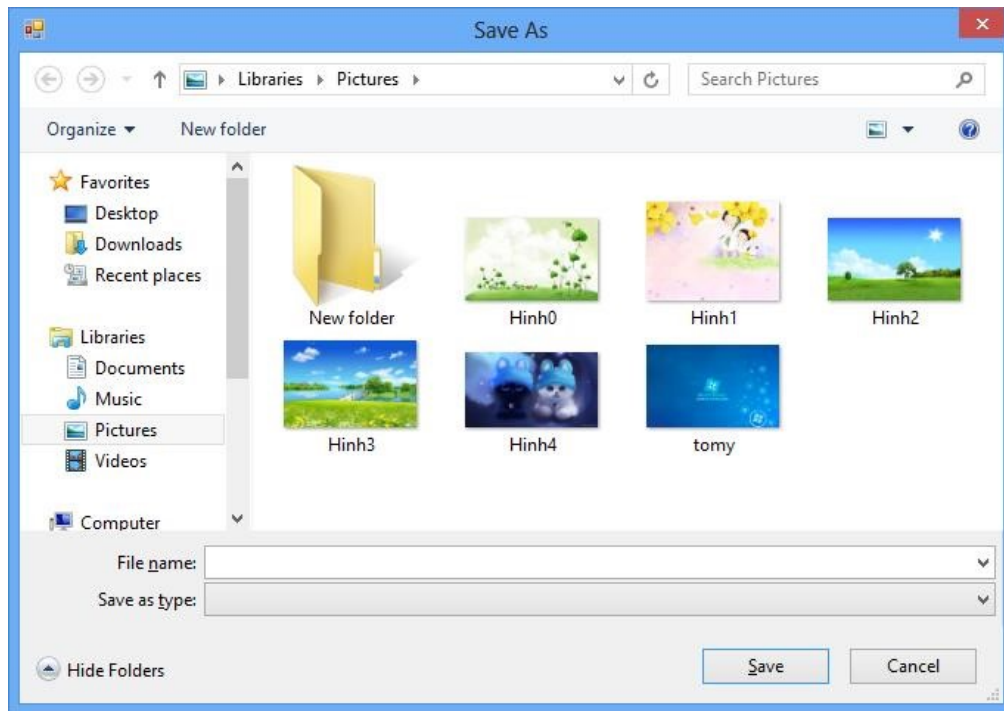
	Lưu ý: Thuộc tính chỉ có hiệu lực khi thuộc tính InitialDirectory mang giá trị rỗng
<i>MultiSelect</i>	Mang giá trị True hoặc False. <ul style="list-style-type: none"> - Nếu là True: Cho phép người dùng chọn nhiều tập tin hoặc thư mục - Nếu là False: Chỉ được chọn 1 tập tin hoặc một thư mục
<i>CheckFileExists</i>	Mang hai giá trị True hoặc False. Nếu là True sẽ cho phép hộp thoại hiển thị một cảnh báo nếu người dùng chỉ định một tập tin không tồn tại và nhấn nút Open.
<i>ReadOnlyChecked</i>	Mang giá trị True hoặc False. <ul style="list-style-type: none"> - Nếu là True: Hiển thị một CheckBox với tiêu đề Open as Read Only phía dưới ComboBox Files Of Types  <ul style="list-style-type: none"> - Nếu là False: Không hiển thị CheckBox. <p>Lưu ý: Thuộc tính chỉ có hiệu lực khi thuộc tính ShowReadOnly mang giá trị True</p>
<i>AddExtension</i>	Mang hai giá trị True hoặc False. <ul style="list-style-type: none"> - Nếu là True sẽ cho phép thêm vào tên mở rộng (jpg, gif, ..) vào tập tin. - Nếu là False: Không cho phép
<i>DefaultExt</i>	Thêm tên mở rộng (.jpg, .gif, ...) cho tập tin nếu người dùng không cung cấp tên mở rộng.

➤ Một số phương thức thường dùng của *OpenFileDialog*:

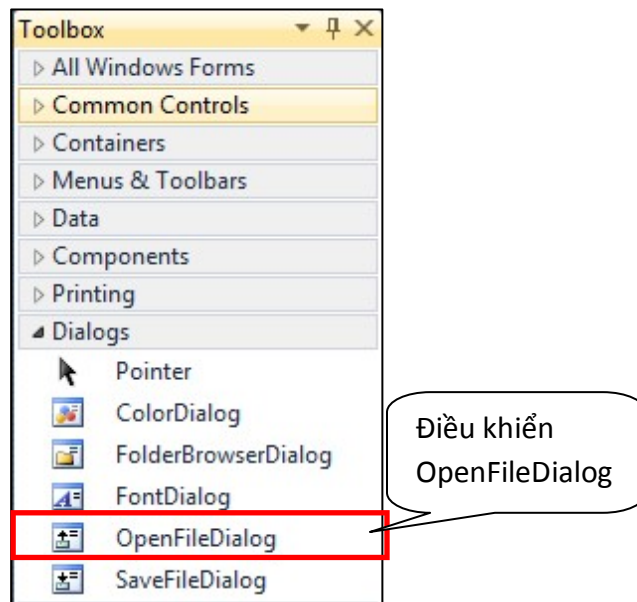
Phương thức	Mô tả
<i>Reset()</i>	Thiết lập giá trị các thuộc tính trở lại giá trị mặc định
<i>ShowDialog()</i>	Hiển thị hộp thoại, bắt buộc người dùng phải thao tác và đóng hộp thoại lại mới có thể thực hiện công việc khác.

7.5. Điều khiển SaveFileDialog

Điều khiển *SaveFileDialog* cho phép hiển thị hộp thoại *SaveFileDialog* để người dùng lựa chọn đường dẫn đến một thư mục hoặc ổ đĩa trên hệ thống để ghi một tập tin mới hoặc ghi đè tập tin đã có. Hộp thoại *SaveFileDialog* có giao diện như hình 7.16 và đặt trong nhóm Dialogs của cửa sổ Toolbox như hình 7.17.



Hình 7.16: Giao diện hộp thoại *SaveFileDialog*



Hình 7.17: Điều khiển *SaveFileDialog* trong cửa sổ Toolbox

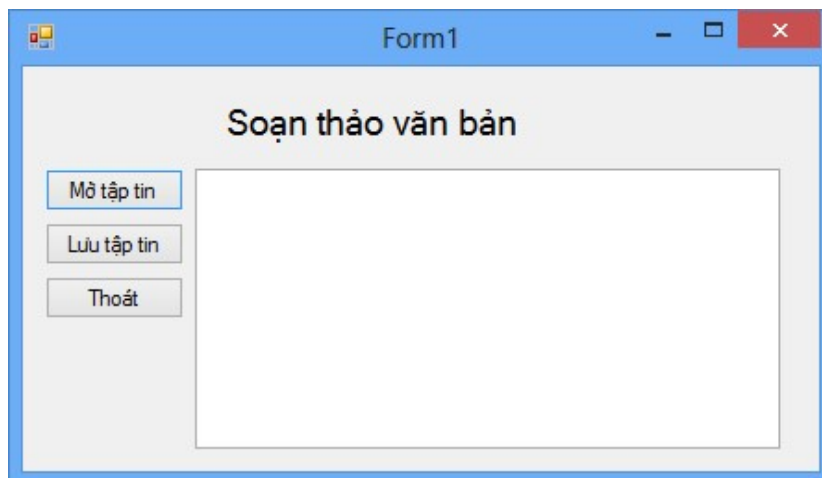
- Một số thuộc tính thường dùng của *SaveFileDialog*:

SaveFileDialog và *OpenFileDialog* đều thừa kế từ lớp cơ sở là *FileDialog* do đó *SaveFileDialog* có nhiều thuộc tính giống như *OpenFileDialog*. Một số thuộc tính *OpenFileDialog* có nhưng *SaveFileDialog* không có như: *Multiselect*, *ReadOnlyChecked*, *ShowReadOnly*. Do là hộp thoại giúp lưu tập tin do đó *SaveFileDialog* hỗ trợ hai thuộc tính mới là: *CreatePrompt* và *OverwritePrompt*.

Thuộc tính	Mô tả
<i>Title</i>	Thiết lập chuỗi hiển thị trên titlebar của hộp thoại <i>OpenFileDialog</i>
<i>Filter</i>	Thuộc tính nhận giá trị là một chuỗi. Chuỗi này chứa các đuôi mở rộng của tập tin, giúp lọc ra trên hộp thoại <i>OpenFileDialog</i> chỉ hiển thị đúng loại đuôi có trong chuỗi. Ví dụ: Thiết lập chuỗi chỉ hiển thị file hình ảnh có đuôi JPG và đuôi GIF: “jpg images *.jpg gif images *.gif” Thiết lập chuỗi hiển thị tất cả những gì có trong thư mục trên hộp thoại: “All *.*”
<i>FileName</i>	Thuộc tính trả về đường dẫn cùng tên tập tin được chọn
<i>FilterIndex</i>	Thuộc tính sử dụng cùng với thuộc tính <i>Filter</i> . Có ý nghĩa sẽ ưu tiên lọc loại tập tin nào trước trong chuỗi <i>Filter</i> . Ví dụ: Chuỗi <i>Filter</i> lọc hai loại tập tin JPG và GIF: “jpg images *.jpg gif images *.gif” Nếu muốn hộp thoại khi hiển thị lọc hình có đuôi JPG đầu tiên thiết lập thuộc tính <i>FilterIndex</i> có giá trị 1. Nếu muốn hộp thoại khi hiển thị lọc hình có đuôi GIF đầu tiên thiết lập thuộc tính <i>FilterIndex</i> có giá trị 2.
<i>InitialDirectory</i>	Giá trị thuộc tính là một đường dẫn đến một thư mục hoặc ổ đĩa. Khi hộp thoại <i>OpenFileDialog</i> được mở lên sẽ hiển thị nội dung của đường dẫn này đầu tiên.
<i>RestoreDirectory</i>	Mang hai giá trị <i>True</i> hoặc <i>False</i> , cho biết hộp thoại có

	trả lại đường dẫn thư mục vừa mở trước đó. Lưu ý: Thuộc tính chỉ có hiệu lực khi thuộc tính InitialDirectory mang giá trị rỗng
<i>CheckFileExists</i>	Mang hai giá trị True hoặc False. Nếu là True sẽ cho phép hộp thoại hiển thị một cảnh báo nếu người dùng chỉ định một tập tin không tồn tại và nhấn nút Open.
<i>CheckPathExists</i>	Mang hai giá trị True hoặc False. Nếu là True sẽ kiểm tra đường dẫn tới tập tin có hợp lệ hay không.
<i>AddExtension</i>	Mang hai giá trị True hoặc False. - Nếu là True sẽ cho phép thêm vào tên mở rộng (jpg, gif, ..) vào tập tin. - Nếu là False: Không cho phép
<i>DefaultExt</i>	Thêm tên mở rộng (.jpg, .gif, ...) cho tập tin nếu người dùng không cung cấp tên mở rộng.
<i>OverwritePrompt</i>	Mang hai giá trị True và False. Nếu là True sẽ hiện cảnh báo khi người dùng ghi đè vào một tập tin đã tồn tại
<i>CreatePrompt</i>	Mang hai giá trị True và False. Nếu là True sẽ hiện thông báo khi người dùng lưu một tập tin mới

Ví dụ 7.4: Viết chương trình soạn thảo văn bản đơn giản sử dụng *OpenFileDialog* để mở tập tin và *SaveFileDialog* để lưu tập tin. Chương trình có giao diện như hình 7.18.



Hình 7.18: Giao diện chương trình soạn thảo văn bản

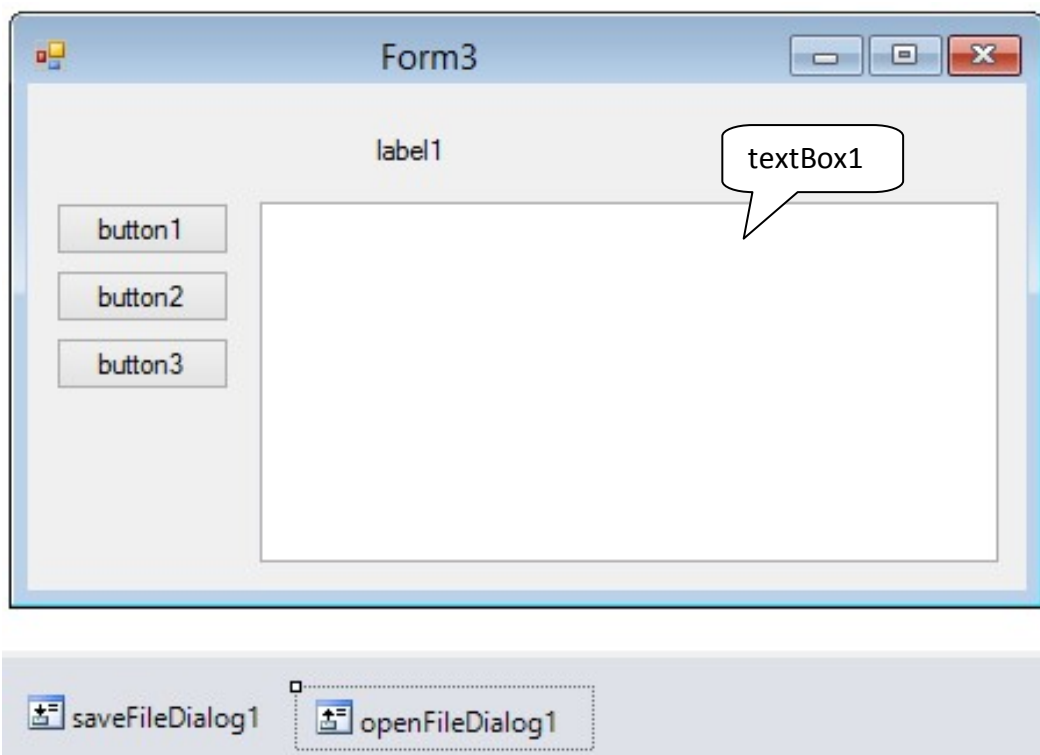
Yêu cầu:

Người dùng nhấn nút “Mở tập tin” để mở một tập tin có định dạng .txt. Nội dung của tập tin này sẽ hiển thị trên *TextBox* (chỉ cho phép hiển thị tập tin .txt)

Người dùng nhấn nút “Lưu tập tin” để lưu nội dung vừa soạn thảo trong *TextBox* vào một thư mục nào đó trên hệ thống.

Hướng dẫn:

Bước 1: Thiết kế giao diện chương trình. Lập trình viên thêm các điều khiển: *Label*, *TextBox*, *Button*, *OpenFileDialog* và *SaveFileDialog* từ cửa sổ Toolbox vào form như hình 7.19



Hình 7.19: Giao diện form soạn thảo văn bản khi thêm điều khiển

Bước 2: Thiết lập giá trị thuộc tính cho điều khiển trong cửa sổ Properties

- label1:
Thuộc tính *Text*: “Soạn thảo văn bản”
Thuộc tính *Font Size*: 14
- button1:
Thuộc tính *Name*: btnMoTapTin
Thuộc tính *Text*: “Mở tập tin”
- button2:

Thuộc tính *Name*: btnLuuTapTin

Thuộc tính *Text*: “Lưu tập tin”

- button3:

Thuộc tính *Name*: btnThoat

Thuộc tính *Text*: “Thoát”

- textbox1:

Thuộc tính *Name*: txtNoiDung

Thuộc tính *MultiLine*: True

Thuộc tính *Size*: 93, 55

- openFileDialog1:

Thuộc tính *RestoreDirectory*: True

Thuộc tính *Title*: “Mở tập tin Text để soạn thảo”

Thuộc tính *Filter*: “File Text|*.txt”

- saveFileDialog1:

Thuộc tính *RestoreDirectory*: True

Thuộc tính *Title*: “Lưu tập tin Text”

Thuộc tính *Filter*: “File Text|*.txt”

Thuộc tính *OverwritePrompt*: True

Bước 3: Viết mã lệnh cho các điều khiển

- Sự kiện *Click* của nút btnMoTapTin:

```
private void btnMoTapTin_Click(object sender, EventArgs e)
{
    if (openFileDialog1.ShowDialog() == DialogResult.OK)
    {
        string tentaptin = openFileDialog1.FileName;
        StreamReader rd = new StreamReader(tentaptin);
        txtNoiDung.Text = rd.ReadToEnd();
        rd.Close();
    }
}
```

- Sự kiện *Click* của nút btnThoat:

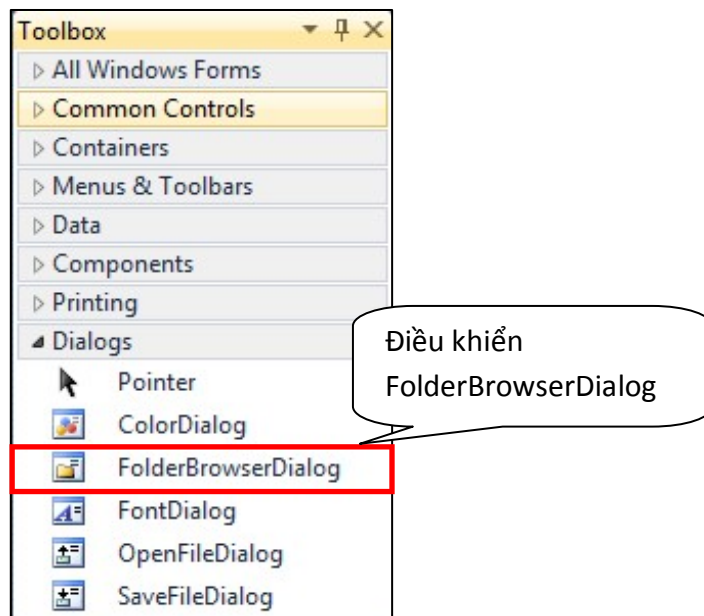
```
private void btnThoat_Click(object sender, EventArgs e)
{
    Close();
}
```

- Sự kiện *Click* của nút `btnLuuTapTin`:

```
private void btnLuuTapTin_Click(object sender, EventArgs e)
{
    if (saveFileDialog1.ShowDialog() == DialogResult.OK)
    {
        string noidung = txtNoiDung.Text;
        string tentaptin = saveFileDialog1.FileName;
        StreamWriter wt = new StreamWriter(tentaptin);
        wt.Write(noidung);
        wt.Close();
    }
}
```

7.6. Điều khiển `FolderBrowserDialog`

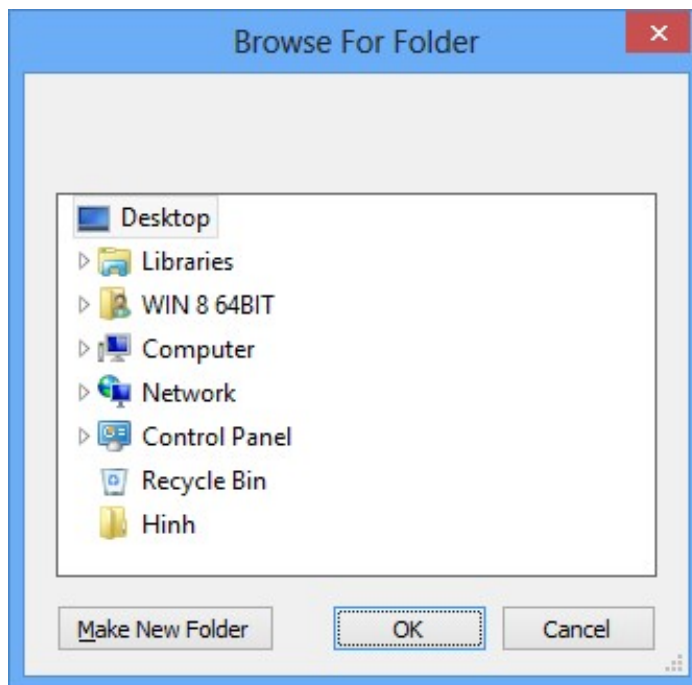
Điều khiển *FolderBrowserDialog* cho phép người dùng chọn một thư mục nào đó đang tồn tại trong hệ thống. Ngoài ra trong quá trình chọn người dùng cũng có thể tạo thư mục mới trên thư mục đang chọn. Điều khiển *FolderBrowserDialog* nằm trong nhóm Dialogs của cửa sổ Toolbox như hình 7.20



Hình 7.20: Điều khiển *FolderBrowserDialog* trong cửa sổ Toolbox

Sự khác biệt cơ bản *FolderBrowserDialog* và *OpenFileDialog* là *FolderBrowserDialog* chỉ cho phép chọn thư mục mà không phải là chọn tập tin và mở như *OpenFileDialog*.

Lập trình viên có thể tạo đối tượng *FolderBrowserDialog* bằng cách kéo điều khiển từ cửa sổ Toolbox vào form hoặc viết mã lệnh để tạo đối tượng *FolderBrowserDialog* và hiển thị hộp thoại *FolderBrowserDialog* bằng phương thức *ShowDialog*. Giao diện hộp thoại *FolderBrowserDialog* như hình 7.21




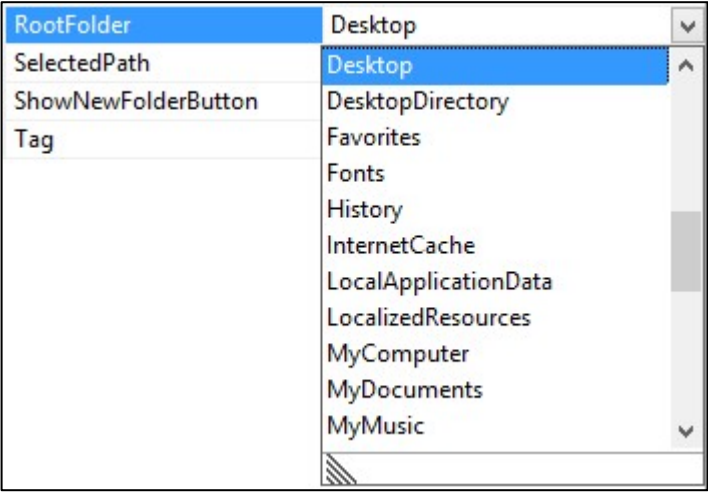
Hình 7.21: Giao diện hộp thoại *FolderBrowserDialog*

Mã lệnh tạo đối tượng và hiển thị hộp thoại *FolderBrowserDialog*:

```
FolderBrowserDialog fd = new FolderBrowserDialog();
fd.ShowDialog();
```

➤ Một số thuộc tính thường dùng của *FolderBrowserDialog*:

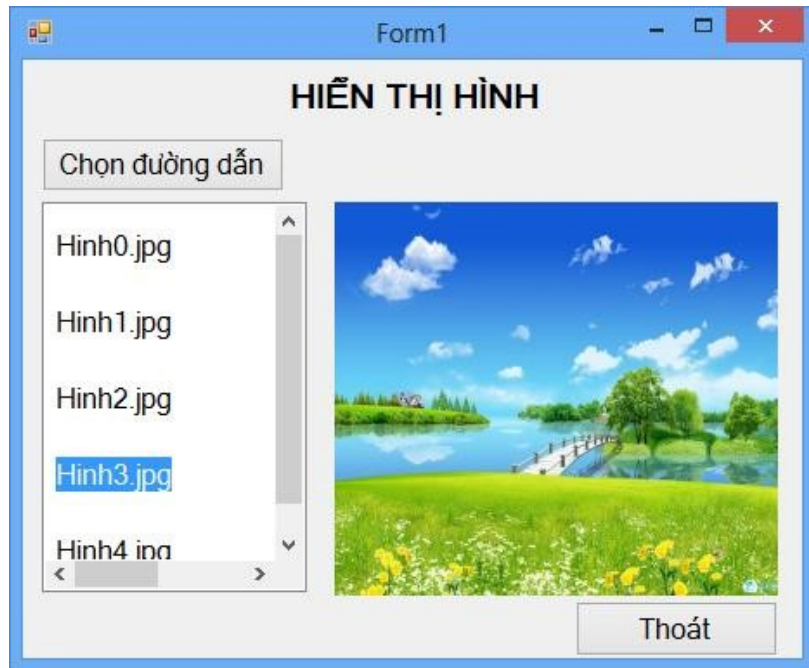
Thuộc tính	Mô tả
<i>Name</i>	Thiết lập tên cho <i>FolderBrowserDialog</i>
<i>SelectedPath</i>	Trả về đường dẫn cùng tên của thư mục mà người dùng chọn
<i>Description</i>	Thiết lập chuỗi mô tả hộp thoại, chuỗi mô tả này được hiển thị dưới titlebar của <i>FolderBrowserDialog</i>
<i>ShowNewFolderButton</i>	Mang hai giá trị True hoặc False. - Nếu là True: Sẽ xuất hiện nút lệnh “Make New Folder” phía dưới bên trái nút lệnh “OK”. Nút

	<p>lệnh này cho phép người dùng tạo một thư mục mới bên trong thư mục đang chọn.</p>  <p>- Nếu là False: Không hiển thị nút lệnh “Make New Folder”</p>
<i>RootFolder</i>	<p>Thiết lập thư mục mặc định sẽ mở ra khi hiển thị hộp thoại <i>FolderBrowserDialog</i>. Các vị trí thư mục mặc định sẽ mở đã được thiết lập sẵn để lập trình viên lựa chọn.</p> 

Ví dụ 7.5: Viết chương trình hiển thị hình ảnh có trên hệ thống máy tính. Giao diện chương trình như hình 7.21.

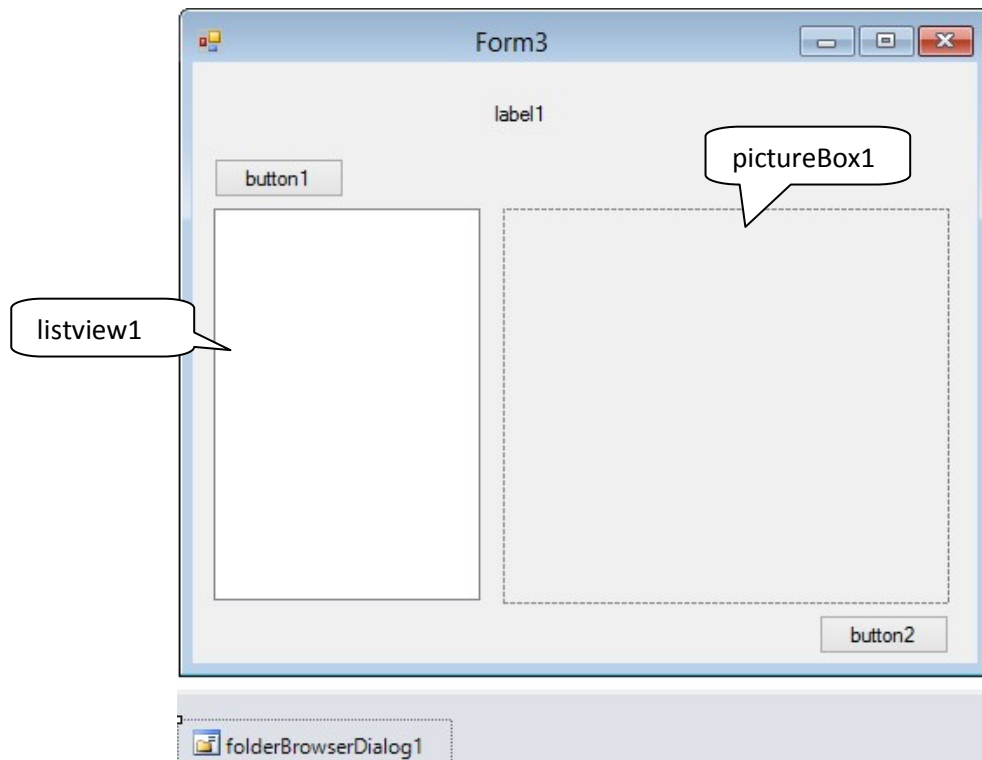
Yêu cầu:

Chương trình cho phép người dùng chọn đường dẫn đến thư mục chứa hình. Một *ListView* trên form sẽ hiển thị danh sách tên các hình có đuôi tập tin dạng .JPG, Khi người dùng chọn một hình trên *ListView* thì hình đó sẽ tương ứng hiển thị trên *PictureBox*.



Hình 7.21: Giao diện chương trình hiển thị hình ảnh

Bước 1: Thiết kế giao diện chương trình. Thêm các điều khiển: *Label*, *Button*, *PictureBox*, *ListView* và *FolderBrowserDialog* vào form như hình 7.22.



Hình 7.22: Giao diện form sau khi thêm điều khiển

Bước 2: Thiết lập giá trị thuộc tính cho các điều khiển trong cửa sổ Properties

- label1:
Thuộc tính *Text*: “HIỂN THỊ HÌNH”
Thuộc tính *Font Size*: 14
Thuộc tính *Font Style*: Bold
- button1:
Thuộc tính *Text*: “Chọn đường dẫn”
Thuộc tính *Name*: btnChon
- button2:
Thuộc tính *Text*: “Thoát”
Thuộc tính *Name*: btnThoat
- listView1:
Thuộc tính *Name*: listPath
Thuộc tính *View*: Tile
Thuộc tính *MultiSelect*: False
Thuộc tính *AllowDrop*: True
- pictureBox1:
Thuộc tính *Name*: picboxHienThi
Thuộc tính *SizeMode*: StretchImage
- folderBrowserDialog1:
Thuộc tính *Name*: myFolderBrowser

Bước 3: Viết mã lệnh cho các điều khiển

- Sự kiện *Click* của nút btnChon:

```
private void btnChon_Click(object sender, EventArgs e)
{
    DialogResult dr = myFolderBrowser.ShowDialog() ;
    string[] path =
    Directory.GetFiles(myFolderBrowser.SelectedPath, "*.jpg");
    if (dr == DialogResult.OK)
    {
        foreach (string i in path)
        {
            int vt = i.LastIndexOf('\\');
            listPath.Items.Add(i.Substring(vt+1, i.Length - vt-1));
        }
    }
}
```

- Sự kiện *Click* của nút btnThoat:

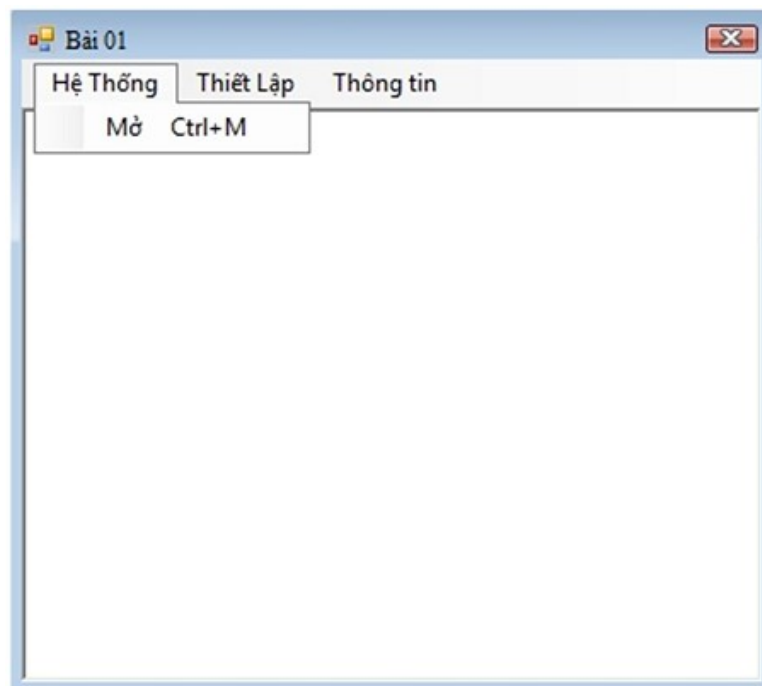
```
private void btnThoat_Click(object sender, EventArgs e)
{
    Close();
}
```

- Sự kiện *SelectedIndexChanged* của nút listPath:

```
private void listPath_SelectedIndexChanged(object sender,
EventArgs e)
{
    pictureBoxHienThi.ImageLocation =
        folderBrowserDialog1.SelectedPath
        + "\\\" + listPath.FocusedItem.Text;
}
```

7.7. Bài tập cuối chương

Câu 1: Thiết kế chương trình xử lý văn bản có giao diện như hình 7.23.



Hình 7.23: Giao diện chương trình xử lý văn bản

Xử lý menu “Hệ Thống”:

- Mở (Ctrl+M): dùng OpenFileDialog để mở một file text bất kỳ hiện có trên ổ cứng máy tính và hiển thị nội dung của file text đó. Nếu không có sinh viên tự tạo ra một file text với nội dung bất kỳ.

Xử lý menu “Thiết Lập”:

- Đổi (Ctrl+D): để viết IN kí tự đầu của mỗi từ và cắt bỏ những kí tự trắng hiện có trong đoạn văn bản đang hiển thị.
- Font (Ctrl+F): dùng FontDialog và ColorDialog để thay đổi font, size, color của vùng văn bản đang được chọn khối trong khung hiển thị.

Xử lý menu “Thông tin”:

- Sinh viên: Hiển thị MessageBox bao gồm các thông tin Họ tên – Mã số sinh viên của sinh viên.

Gợi ý: khoảng trắng thừa là những khoảng trắng đứng đầu và cuối đoạn văn bản, những khoảng trắng đứng sau một khoảng trắng khác, những khoảng trắng đứng trước các dấu ngắt câu như dấu phẩy “,” dấu chấm “.” dấu chấm phẩy “;” dấu hai chấm “:” dấu hỏi “?” dấu chấm than “!”

Câu 2: Thiết kế chương trình tính toán có giao diện như hình 7.24.

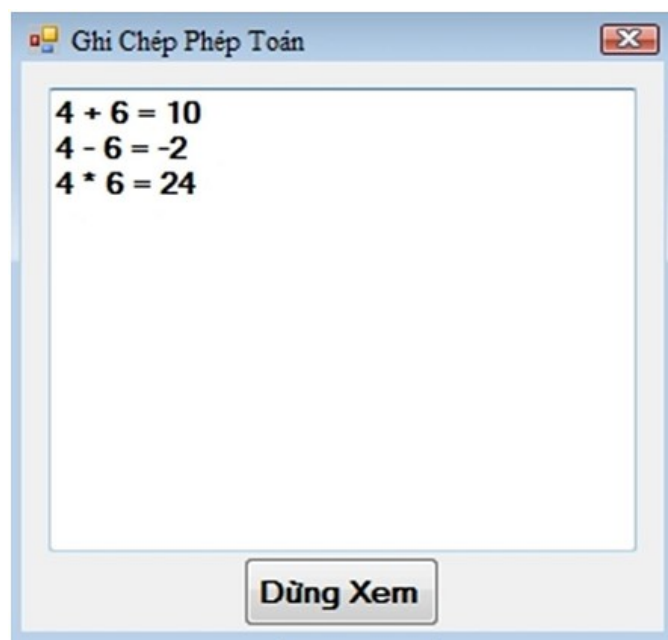


Hình 7.24: Giao diện chương máy tính

Yêu cầu:

- Xử lý sự kiện điều khiển form hình 7.24:

- Khi chưa nhấn lên Button “On/Off” một lần nào hoặc số lần kích lên nút là chẵn thì máy tính trong điều kiện không sẵn sàng sử dụng, nghĩa là không cho tác động lên các thành phần khác trên form máy tính.
- Mặc định, RadioButton “Có” đang được chọn.
- Bẫy lỗi nhập, chỉ cho nhập số vào hai TextBox txtSoA và txtSoB.
- TextBox kết quả không cho người dùng nhập liệu mà chỉ để xuất kết quả của phép toán khi người dùng nhấn các Button “Cộng”, “Trừ”, “Nhân”, “Chia” tương ứng. Khi kích nút phép toán nào thì nút đang được chọn sẽ đổi màu nền. Nếu kích lên nút phép toán khác thì nút phép toán đã được kích trước đó sẽ trở về màu nền ban đầu.
- Khi RadioButton “Có” đang được chọn, cứ mỗi phép toán ra được kết quả thì ghi toàn bộ phép toán đó thành một dòng dạng Giá trị A Tên phép toán Giá trị B = Kết quả tính vào tập tin GhiText.txt (xem hình 7.25), đồng thời, cho tác động Button “Xem File”. File GhiText.txt phải đặt cùng vị trí với file chạy (.exe) của chương trình.
- Khi chọn RadioButton “Không” thì tập tin GhiText.txt hiện có sẽ bị xóa, đồng thời, không cho tác động lên Button “Xem File”.
- Kích Button “Xem File” cho phép hiển thị form ghi chép phép toán (hình 7.25)
- Kích Button “Làm lại” sẽ xóa sạch nội dung các ô nhập, ô kết quả và nội dung tập tin GhiText.txt, chờ thực hiện bài toán mới



Hình 7.25: Giao diện form ghi chép phép toán

- Xử lý sự kiện điều khiển form hình 7.25:
 - Khi load form, xuất hiện OpenFileDialog cho chọn tập tin GhiText.txt và hiển thị nội dung file lên form.
 - Button “Dừng Xem” thì đóng form ghi chép phép toán để trở về màn hình form máy tính.

Câu 3: Thiết kế chương trình tính toán Fibonacy có giao diện như hình 7.26.

Hình 7.26: Giao diện chương trình tính toán Fibonacy

Yêu cầu:

- Button “Mở/Tắt”: Khi chưa kích lên nút một lần nào hoặc số lần kích lên nút là chẵn thì tắt máy, nghĩa là không cho tác động lên các thành phần khác trên form. Nếu số lần kích lên nút là lẻ thì mở máy, nghĩa là cho tác động lên các thành phần còn lại trên form.
- Nhập và tính kết quả:
 Cách tính giá trị từng số hạng trong dãy Fibonacci: $F_0=1, F_1=1, F_n=F_{n-1}+F_{n-2}$ (với $n \geq 2$)
 - Hình 7.27: Giao diện ban đầu khi chưa nhập số n, chờ nhập số n. Bẫy lỗi nhập, chỉ cho nhập vào một số n thỏa $2 \leq n \leq 10$.

Hình 7.27: Giao diện ban đầu khi chưa nhập n

- Hình 7.28: khi vừa nhập vào số n thỏa điều kiện, hiển thị giá trị của F_0 và F_1 . Đồng thời, điền thêm giá trị n vào nhãn của số hạng F đang để trống.

Hình 7.28: Giao diện khi nhập n thỏa điều kiện

- Hình 7.29: khi nhấp chọn Button “=”, in giá trị số hạng F_n vào ô kết quả. Khi đã in kết quả thì không cho thay đổi giá trị trong ô nhập n .

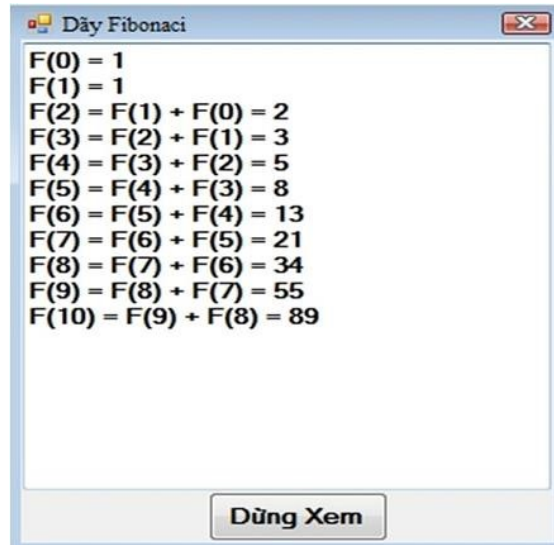
Hình 7.29: Kết quả tính Fibonacci khi $n = 5$

- GroupBox “Ghi từng số hạng vào file văn bản”:
 - Khi radioButton “Có” đang được chọn, cứ mỗi giá trị n nhập hợp lệ và ra được kết quả thì ghi vào tập tin GhiChep.txt đầy đủ từng số hạng từ $F_0 \rightarrow F_n$, mỗi số

hạng ghi thành một dòng dạng giống như trong Hình 2, đồng thời, cho tác động Button “Xem File”.

(Tập tin GhiChep.txt phải đặt cùng vị trí với file chạy (.exe) của chương trình.)

- Khi chọn vào radioButton “Không” thì tập tin GhiChep.txt hiện có sẽ bị xóa, đồng thời, không cho tác động lên Button “Xem File”.
- Nhấn chọn Button “Xem File” cho hiển thị form Dãy Fibonacci trên form như hình 7.30.



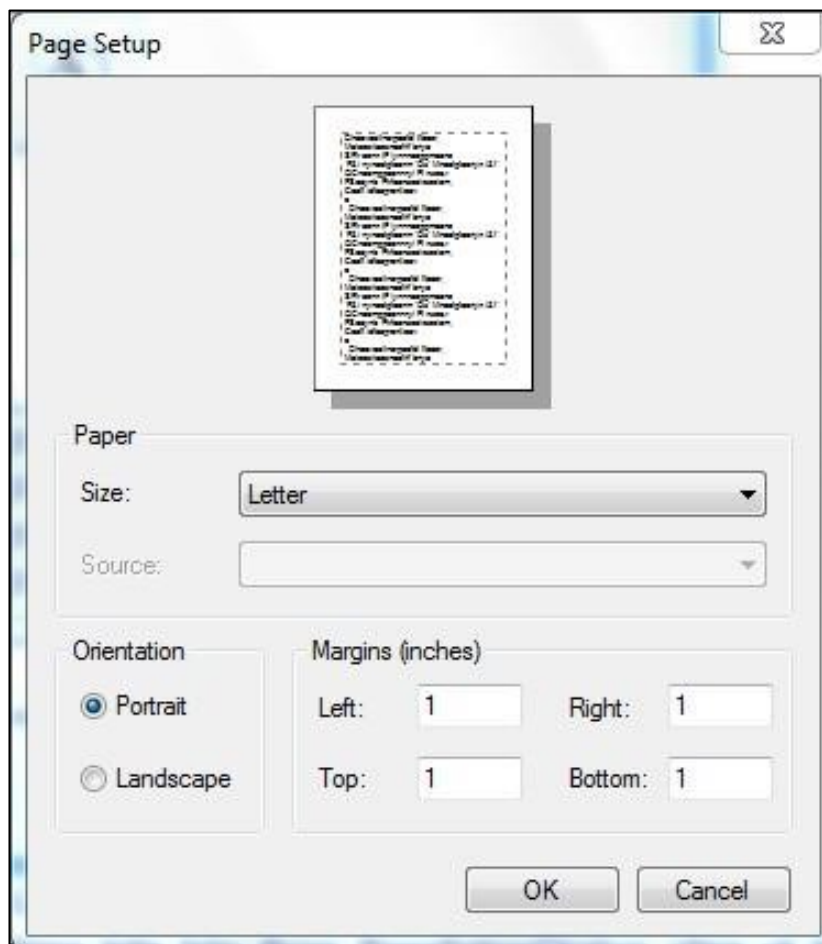
- Hình 7.30: Giao diện form hiển thị kết quả Fibonacy từ F_0 đến F_{10}

- Xử lý form Dãy Fibonacci:
 - Khi load form, xuất hiện OpenFileDialog cho chọn tập tin GhiChep.txt và hiển thị nội dung tập tin lên form.
 - Nhấn chọn Button “Dừng Xem” thì đóng form Dãy Fibonacci để trở về form hình 7.26.
- Nhấn chọn Button “Tiếp tục” xóa sạch nội dung các ô nhập, ô xuất kết quả và nội dung tập tin GhiChep.txt và chờ nhập số n mới.

CHƯƠNG 8: LÀM VIỆC VỚI ĐIỀU KHIỂN IN ẤN

8.1. Điều khiển *PageSetupDialog*

Điều khiển *PageSetupDialog* được hiển thị dạng hộp thoại khi chương trình thực thi. Việc hiển thị dạng hộp thoại cho phép người dùng dễ dàng thay đổi các thiết lập về trang như: canh lề, định hướng, kích thước trang, Hộp thoại *PageSetupDialog* có giao diện như hình 8.1.



Hình 8.1: Giao diện hộp thoại *PageSetupDialog*

Những thay đổi của người dùng trên hộp thoại *PageSetupDialog* cũng sẽ làm thay đổi các giá trị tương ứng trên hộp thoại *PrintDocument*. Do đó việc thiết lập các giá trị trong hộp thoại *PageSetupDialog* cũng ảnh hưởng đến hình dạng của trang khi in ấn.

Hiển thị hộp thoại *PageSetupDialog* sử dụng phương thức *ShowDialog()*. Tuy nhiên, điểm khác biệt là để có thể gọi được phương thức *ShowDialog()* thì cần phải thực hiện một thao tác trước là gán một đối tượng thuộc lớp *PrintDocument* cho thuộc tính