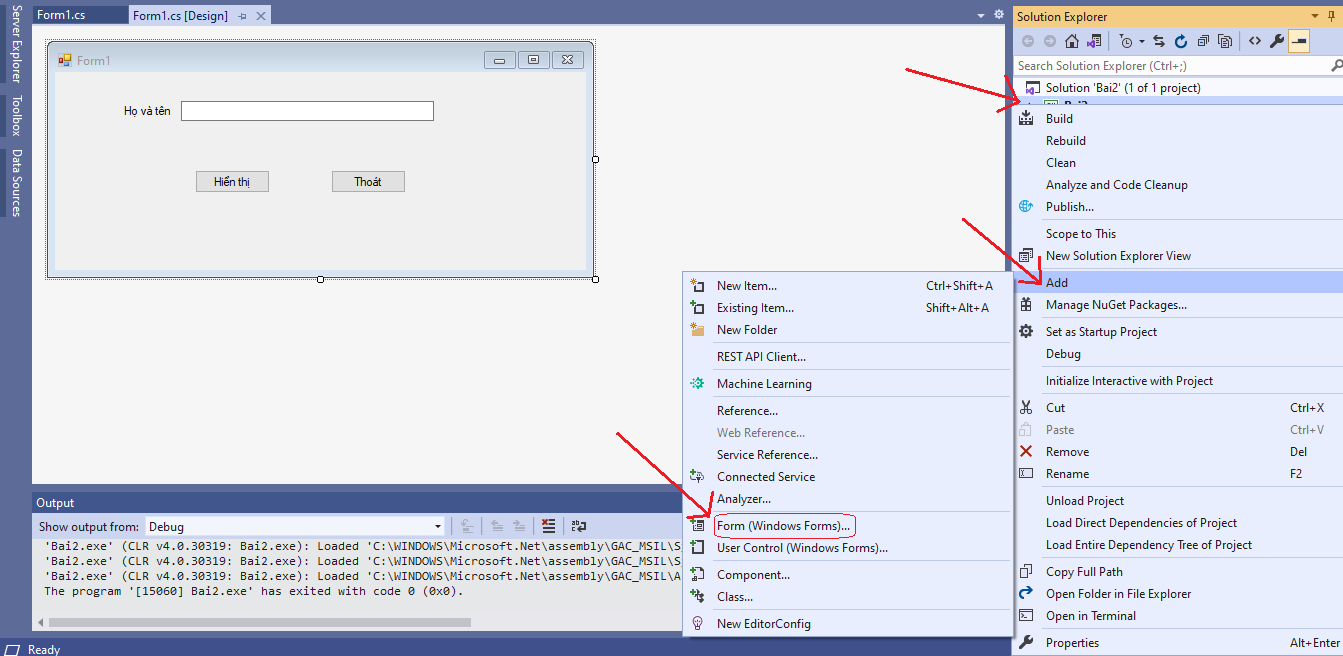
1. Giới thiệu winform

* Winform là thuật ngữ mô tả một ứng dụng được viết dùng .NET FrameWorrk và có giao diện người dùng Windows Forms.
* Mỗi màn hình windows cung cấp một giao diện giúp người dùng giao tiếp với ứng dụng. Giao diện này được gọi là giao diện đồ họa (GUI) của ứng dụng.
* Là các ứng dụng windows chạy trên máy tính như : Microsoft, Word, Excel, Access, Calculator, yahoo, Mail…

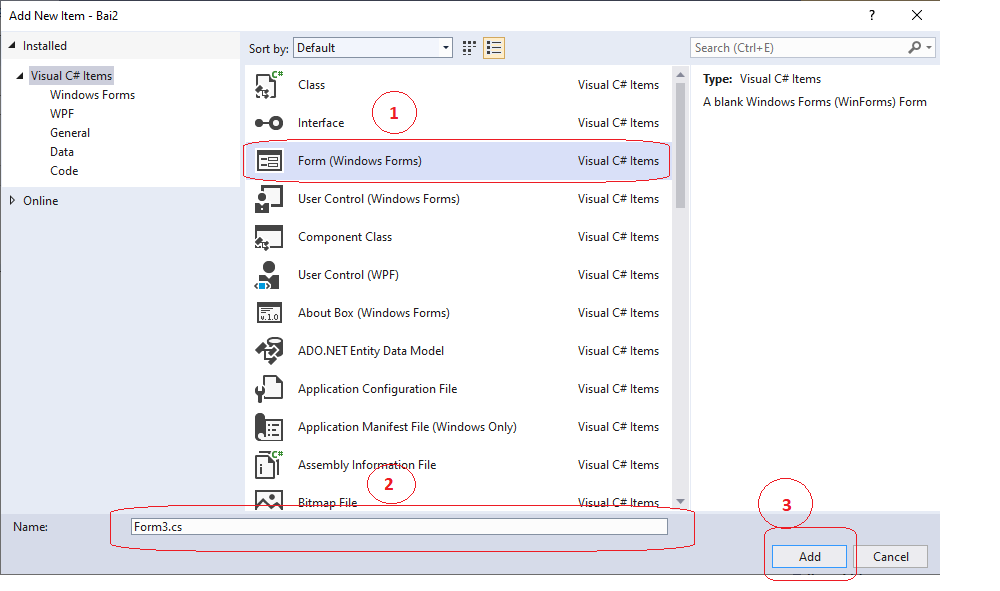
1. Windown form
   1. thêm, xóa 1 form trong project

a, thêm

* bước 1: vào menu Project/Add Windows Form howcj menu Project/Add New Item



* bước 2: thiết lập trên cửa sổ Add New Item:



**\***Thêm một Form có sẵn vào project: Ta cũng có thể thêm một Form đã có sẵn bằng cách:

Bước 1: Vào menu *Project*| *Add Existing Item.*

Bước 2: Theo đường dẫn, chọn Form cần add

Bước 3: Nhấn nút Add

b, Xóa một Form khỏi project

Để xóa một Form đang có trong dự án:

Bước 1: *Nhấn chuột phải*vào Form cần gỡ bỏ (ở cửa sổ Solution Explorer).

Bước 2: Chọn *Delete*trong menu hiện ra

* 1. một số thuộc tính

| **Thuộc tính** | **Diễn giải** |
| --- | --- |
| **Text** | Tiêu đề (titlebar) của Form |
| **WindowState** | Trạng thái thể hiện của Form (normal, minimized,maximized) |
| **StartPosition** | Vị trí Form hiển thị khi chương trình gọi đến Form |
| **TopMost** | Form có chọn hiện ưu tiên trong các action Form hay không |
| **Locked** | Khóa vị trí các controls trong quá trình thiết kế |
| **Icon** | Chọn biể tượng cho Form |
| **ControlBox** | Mang giá trị True hoặc False. Nếu thiết lập thuộc tính là False thì sẽ loại bỏ các nút minimize và nút maximize trên Form |
| **MinimizeBox** | Có hiện nút thu nhỏ của Form |
| **MaximizeBox** | Có hiện nút phóng to của Form |
| **BackColor** | Lựa chọn màu nên cho Form |
| **BackGroundImage** | Xác định file hình làm hình nền cho Form |
| **FormBoderStype** | Lựa chọn kiểu đường viền cho Form |
| **ForeColor** | Chọn màu chữ cho Form |
| **Font** | Lựa chọn font chữ, font size, font style cho Form |
| **Cusor** | Thiết lập hình dạng con trỏ khi di chuyển con trỏ vào Form |
| **IsMDIContainer** | Mang giá trị True hoặc False; True: Form ở dạng MDI Form (Form cha), False: Form ở dạng bình thường |
| **BackgroundImageLayout** | Thiết lập việc hiển thị hình vừa thêm trong thuộc tính BackGroundImage sẽ hiển thị trên Form ở dạng: bình thường (None), giữa (Center),... |
| **AcceptButton** | Giá trị thuộc tính này nhận là tên của một Button trên Form. Khi đó thay vì nhấp chuột vào Button để thực thi thì người dùng có thể nhấn phím Enter trên bàn phím |
| **CancleButton** | Giá trị thuộc tính này nhận là tên của một Button trên Form. Khi đó thay vì nhấp chuột vào Button để thực thi thì người dùng có thể nhấn nút Escape trên bàn phím |
| **Keypreview** | Cho phép Form nhận các giá trị từ bàn phím, mặc định là False (không nhận) |

* 1. 1 số sự kiện thông dụng

| **Tên sự kiện** | **Diễn giải** |
| --- | --- |
| **FormLoad** | Xảy ra khi Form bắt đầu chạy, ta thường dùng sự kiện Load để khởi tạo các giá trị ban đầu |
| **FormClosing** | Xảy ra trước khi đóng Form, ta thường dùng sự kiện này để giải phóng tài nguyên hệ thống |
| **KeyDown** | Xảy ra khi một phím được nhấn trên Form. muốn cho sự kiện này xảy ra ta phải đặt thuộc tính KeyPreview có giá trị là True, để lấy mã của phím được nhấn (keyCode) |
| **MouseClick** | Xảy ra khi nhấn một trong ba nút của chuột: chuột trái, chuột phải và chuột giữa |
| **BackColorChanged** | Giá trị thuộc tính BackColor thay đổi |
| **ForeColorChanged** | Giá trị thuộc tính ForeColor bị thay đổi |
| **Click** | Nhấp chuột vào vùng làm việc của Form |
| **DoubleClick** | Nhấp đúp chuột vào vùng làm việc của Form |
| **FormClosed** | Form đã được đóng hoàn toàn |
| **KeyDown** | Phím được nhấn xuống |
| **KeyUp** | Phím được thả ra |
| **KeyPress** | Phím được nhấn xuống và thả ra |
| **MouseEnter** | Chuột nằm trong vùng thấy được của Form |
| **MouseHover** | Chuột nằm trong vùng hiển thị một khoảng thời gian |
| **MouseDown** | Nhấn chuột trên vùng hiển thị của Form |
| **MouseLeave** | Chuột ra khỏi vùng thấy được của Form |
| **MouseMove** | Chuột được di chuyển trên Form |
| **Move** | Form được di chuyển |
| **Resize** | Form bị thay đổi kích thước |
| **TextChanged** | Giá trị của thuộc tính Text bị thay đổi |

* 1. Một số phương thức
* **Show()**: Sử dụng để hiển thị Form.
* **ShowDialog()**: Tương tự như Show() nhưng Form hiển thị bằng phương thức ShowDialog() sẽ bắt buộc người dùng phải thao tác cho tới khi đóng Form (khi chưa đóng Form thì không được thao tác với Form khác).
* **Hide()**: Ẩn Form, tương tự như việc thiết lập thuộc tính Visible = False.
* **Close()**: Đóng Form.
  1. Phân loại form

Trong Form sẽ được phân làm 3 loại Form:

* **Form bình thường** (*Normal Form*): Một Form bình thường khi được gọi hiển thị bên trong một Form cha và được chỉ định Form cha chứa nó là Form nào thông qua thuộc tính **MdiParent**khi đó sẽ trở thành Form con.
* **Form cha** (*MdiParent Form*): Là Form có thể chứa các Form khác bên trong nó, để một Form trở thành Form cha ta cần thiết lập thuộc tính **isMdiContainer**của Form có giá trị là True.
* **Form con** (*MdiChildren Form*)

**Gọi hiển thị Form**

Gọi hiển thị Form2 từ Form1 dạng: Form2 là Normal Form.

|  |  |
| --- | --- |
| 1  2 | Form frm = new Form2();  frm.Show(); |

Gọi hiển thị Form2 thì Form1 dạng: Form2 là Form con.

|  |  |
| --- | --- |
| 1  2  3 | Form frm = new Form2();  frm.MdiParent = this;  rfm.Show(); |

**Đóng Form, dừng chương trình**

Để đóng Form hoặc dừng chương trình ta sử dụng 2 dòng lệnh sau:

* **This.Close()**: Đóng Form hiện hành
* **Application.Exit()**: Đóng chương trình

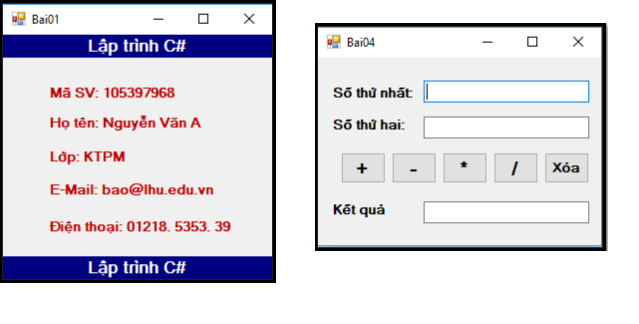
1. Các điều khiển
   1. label – button – textbox

**Label**thường được dùng hiển thị thông tin chỉ đọc và thường sử dụng kèm với các điều khiển khác để mô tả chức năng.

**Một số thuộc tính thường dùng của Label**

| **Thuộc tính** | **Mô tả** |
| --- | --- |
| BorderStyle | Thiết lập đường viền |
| Font | Thiết lập font chữ hiển thị trên Label |
| Text | Nội dung hiển thị trên Label |
| Backcolor | Màu nền của Label |
| ForeColor | Thiết lập màu chữ hiển thị trên Label |
| Name | Tên của Label |
| TextAlign | Canh lề nội dung của Label (Left | Center | Right) |
| Visible | Ẩn hoặc hiện Label |

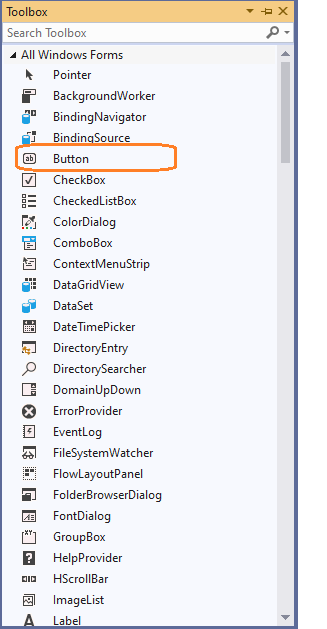
**Ví dụ:**



Các dòng chữ ở hai ví dụ trên đều sử dụng Label để mô tả, sử dụng thuộc tính Text trong Label để có thể đặt nội dung tùy ý.

2. Button

Button là điều khiển tạo giao diện nút lệnh trên Form, khi người dùng nhấn chuột vào nút lệnh thì chương trình sẽ thực hiện một hành động nào đó. Button được đặt trong nhóm Common Controls của cửa sổ Toolbox.



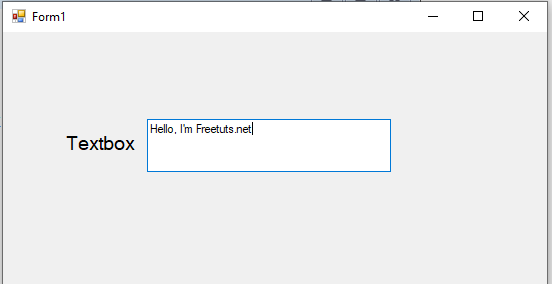
Công dụng của Button:

* Dùng để thực thi lệnh.
* Khi nhấp chuột lên Buton, chương trình nhận được tính hiệu Click và lệnh được thi hành.

| **Thuộc tính** | **Mô tả** |
| --- | --- |
| Name | Đặt tên cho nút lệnh |
| Text | Nội dung hiển thị lên nút nhấn |
| Visible | Ẩn, hiện nút nhấn |
| Enable | Cho phép/ không cho phép tương tác với nút lệnh |
| Font | Chỉ đinh kiểu chữ, kích cỡ chữ hiển thị |
| Image | Hình ảnh hiển thị trên nút lệnh |
| ForeColor | Màu chữ hiển thị trên nút lệnh |
| BackColor | Màu của nút lệnh |
| TabIndex | Chỉ định thứ tự tab của các Button trên Form |
| **Sự kiện** | **Diễn giải** |
| Click | Sự kiện nhấn chuột vào Button |
| TextChanged | Giá trị thuộc tính Text bị thay đổi |

3. Textbox

Trong phần này chúng ta sẽ tìm hiểu về công dụng của Textbox là gì, cũng như các thuộc tính và các phương thức thường dùng trong Textbox.



**Công dụng của Textbox:**

* Dùng để trình bày văn bản và cho phép người dùng được thay đổi nội dung văn bản.
* Công dụng chính là cho người dùng nhập văn bản.
* Ngoài các thuộc tính chung của các control, thì các bạn cần lưu ý thêm một số thuộc tính bên dưới:

| **Thuộc tính** | **Diễn giải** |
| --- | --- |
| PasswordChar | Ký tự thay thế khi nhập vào TextBox |
| Multiline | Cho phép Textbox có nhiều dòng hay không |
| ScrollBars | Thanh cuộn (None / Horizontal / Vertical / Both) |
| MaxLength | Quy định chuỗi Max sẽ nhập vào Textbox, mặc định là 32767 |
| Focus | Textbox sẵn sàng được tương tác bởi người dùng |

**Một số sự kiện thường dùng của Textbox:**

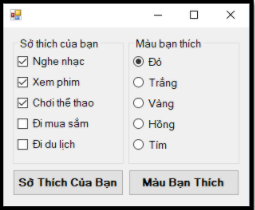
| **Sự kiện** | **Diễn giải** |
| --- | --- |
| KeyDown | Thực hiện công việc nào đó khi một phím được nhấn xuống |
| KeyUp | Thực hiện công việc nào đó khi một phím được thả ra |
| KeyPress | Xảy ra khi người dùng nhấn một phím và thả ra. Mỗi sự kiện KeyPress cho ta một cặp sự kiện KeyDown và KeyUp |
| TextChanged | Giá trị của thuộc tính Text bị thay đổi |
| MouseEnter | Chuột nằm trong vùng thấy được của Textbox |
| MouseHover | Chuột nằm trong vùng hiển thị một quảng thời gian |
| MouseLeave | Chuột ra khỏi vùng nhập liệu của Textbox |
| MouseMove | Chuột được di chuyển trên Textbox |

**Một số phương thức của Textbox:**

| **Phương thức** | **Diễn giải** |
| --- | --- |
| Clear() | Xóa tất cả chuỗi hiển thị trong Textbox |
| Cut() | Di chuyển phần nội dung bôi đen của chuỗi |
| Paste() | Dán phần nội dung được chọn của chuỗi |
| Copy() | Sao chép phần nội dung được bôi đen của chuỗi |
| Undo() | Khôi phục thao tác trước |
| Select() | Chọn một phần nội dung của chuỗi trong Textbox |
| SelectAll() | Chọn tất cả nội dung của chuỗi trong Textbox |
| DeselectAll() | Bỏ chọn chuỗi trong Textbox |

* 1. checkbox – radiobutton

**Checkbox**là điều khiển cho phép người dùng chọn hoặc không chọn trên Form, người dùng có thể chọn một hoặc nhiều giá trị cùng lúc.

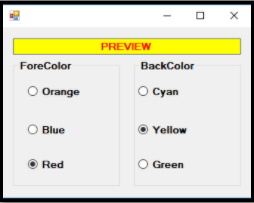


Chúng ta có một số thuộc tính của Checkbox dưới đây:

| **Thuộc tính** | **Mô tả** |
| --- | --- |
| Checked | Giá trị True là được chịn, nếu thiết lập False thì không được chọn |
| CheckState | Thường dùng để kiểm tra tình trạng Checkbox có được chọn hay không. Mang 3 giá trị UnChecked, Checked và Indeterminate.  - Checked: Điều khiển đang được chọn  - UnChecked: Điều khiển không được chọn  - Indeterminate: Điều khiển ở trạng thái không hoạt động |
| AutoCheck | Mang giá trị True hoặc False, nếu giá trị True thì cho phép người dùng nhấp chuột để chọn, nếu là False thì không cho phép người dùng nhấp chuột chọn |
| Text | Chuỗi văn bản hiển thị bên cạnh Checkbox |
| ThreeState | Mang giá trị True hoặc False; nếu là True thì cho phép Checkbox có 3 trạng thái: Checked, UnChecked, Indeterminate |
| RightToLeft | Mang giá trị Yes hoặc No; cho biết chuỗi văn bản hiển thị (thuộc tính Text) nằm bên trái hay bên phải của Checkbox |

2. RadioButton

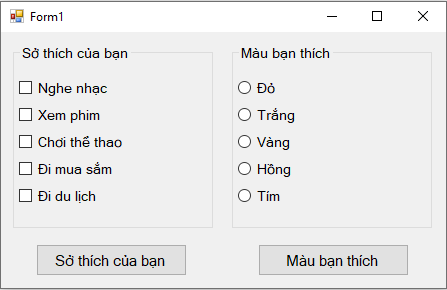
**RadioButton** tương tự như Checkbox là điều khiển cho phép người dùng chọn hoặc không chọn, điểm khác biệt là với RadioButton người dùng chỉ có thể chọn một giá trị trong một nhóm.



Cả hai điều khiển Checkbox và RadioButton đều có chung một sự kiện đó chính là **CheckedChanged**, xảy ra khi thay đổi trạng thái chọn của Checkbox, RadioButton.

Trong ví dụ này mình sẽ thực hiện một viết một ứng dụng sử dụng Checkbox và RadioButton, cụ thể như sau:

Thực hiện thiết kế giao diện cho Form như hình dưới đây:



Xử lý một số sự kiện sau:

* Khi click vào nút "**Sở thích của bạn**" thì những Checkbox được chọn bởi người dùng sẽ hiển thị trên hộp thoại MessageBox. Nếu không có Checkbox nào được chọn thì thông báo cho người dùng biết.
* Khi click vào nút "**Màu bạn thích**" thì RadioButton được chọn bởi người dùng sẽ được hiển thị trên hộp thoại MessageBox.

Chúng ta sẽ lần lượt thực hiện các yêu cầu nhé. Đầu tiên sẽ tạo giao diện cho Form giống như ứng dụng mẫu.

* Hai GroupBox (*ở bài sau mình sẽ hướng dẫn chi tiết*)
* 5 Checkbox với 5 sở thích tương ứng
* 5 RadioButton với 5 màu tương ứng
* 2 Button

Các bạn sẽ thay đổi một số thuộc tính như: Text, name, font,... cho giống với Form mẫu.

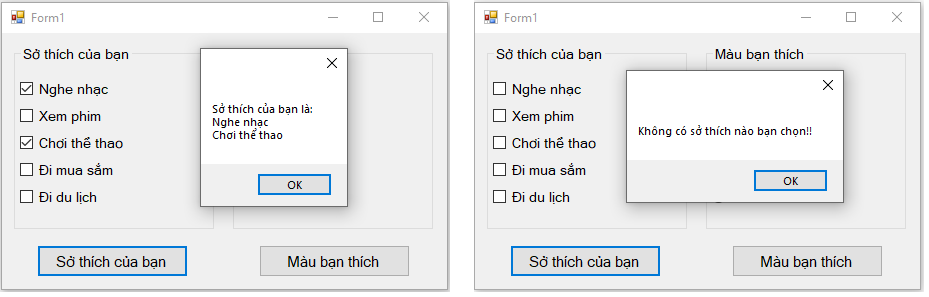
Sau khi thiết kế xong giao diện cho Form, bây giờ đến lúc các bạn sử lý sự kiện cho hai nút Button. Sự kiện đầu tiên là khi các bạn Click vào nút "**Sở thích của bạn**" thì các Checkbox được chọn sẽ hiển thị trên MessageBox.

Các bạn Double Click vào Button cần xử lý để viết sự kiện. Chúng ta sẽ sử dụng thuộc tính Checked để kiểm tra xem Checkbox có được chọn hay không bằng cách cho **Checked == true.**

Ta sẽ tạo một biến **String str**để lưu các Checkbox được chọn, nếu Checked == true thì thêm vào str.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33 | private void btn\_sothich\_Click(object sender, EventArgs e)          {              string str = "";              if(chkbox\_muasam.Checked == true)              {                  str = chkbox\_muasam.Text + "\n";              }              if(chkbox\_nghenhac.Checked == true)              {                  str = str + chkbox\_nghenhac.Text + "\n";              }              if(chkbox\_thethao.Checked == true)              {                  str = str + chkbox\_thethao.Text + "\n";              }              if(chkbox\_xemphim.Checked == true)              {                  str = str + chkbox\_xemphim.Text + "\n";              }              if(chxbox\_dulich.Checked == true)              {                  str = str + chxbox\_dulich.Text;              }                if(str.Length > 0)              {                  MessageBox.Show("Sở thích của bạn là: \n" + str);              }              else              {                  MessageBox.Show("Không có sở thích nào bạn chọn!!");              }          } |

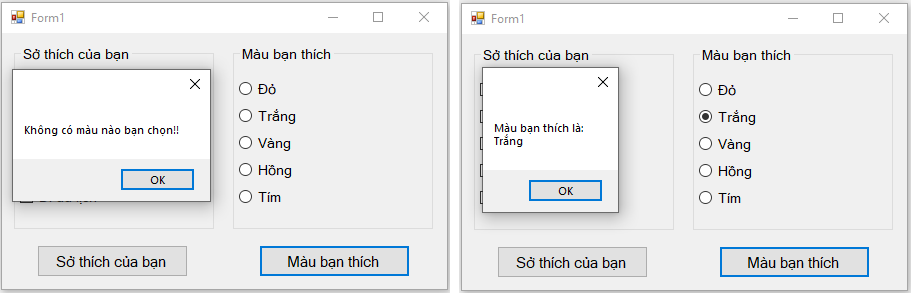
**Kết quả:**



Tiếp đến ta sẽ xử lý sự kiện khi Click vào nút "**Màu bạn thích**", tương tự như nút "**Sở thích của bạn**" ta sử dụng thuộc tính Checked để kiểm tra xem RadioButton có được chọn, nếu được chọn thì hiển thị ra MessageBox.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33 | private void btn\_mauthich\_Click(object sender, EventArgs e)          {              string str = "";              if (rdbtn\_do.Checked == true)              {                  str = rdbtn\_do.Text + "\n";              }              if (rdbtn\_hong.Checked == true)              {                  str = rdbtn\_hong.Text + "\n";              }              if (rdbtn\_tim.Checked == true)              {                  str = rdbtn\_tim.Text + "\n";              }              if (rdbtn\_trang.Checked == true)              {                  str = rdbtn\_trang.Text + "\n";              }              if (rdbtn\_vang.Checked == true)              {                  str = rdbtn\_vang.Text;              }                if (str.Length > 0)              {                  MessageBox.Show("Màu bạn thích là: \n" + str);              }              else              {                  MessageBox.Show("Không có màu nào bạn chọn!!");              }          } |

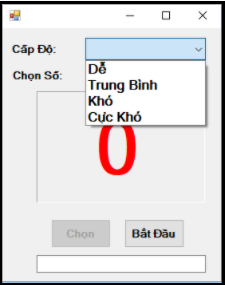
**Kết quả:**



* 1. combobox – listbox

1. ComboBox

ComboBox được dùng để hiển thị một danh sách những mỗi lần người dùng chỉ có thể chọn một lựa chọn, có thể nhập mới.



**Một số thao tác với ComboBox:**

* **Add()**: Thêm một mục chọn vào cuối danh sách ListBox.
* **Insert()**: Chèn thêm mục chọn vào vị trí **i**.
* **Count**: Trả về số muc chọn hiện đang có.
* **Item()**: Trả về mục chọn ở vị trí thứ **i**.
* **Remove()**: Bỏ mục chọn.
* **RemoveAt()**: Bỏ mục chọn ở vị trí thứ **i**.
* **Contains()**: Trả về True nếu có mục chọn trong danh sách, trả vể False nếu không có mục chọn trong danh sách.
* **Clear**: Xóa tất cả các mục chọn.
* **IndexOf()**: Trả về vị trí mục chọn trong danh sách, nếu không tìm thấy sẽ trả về -1.

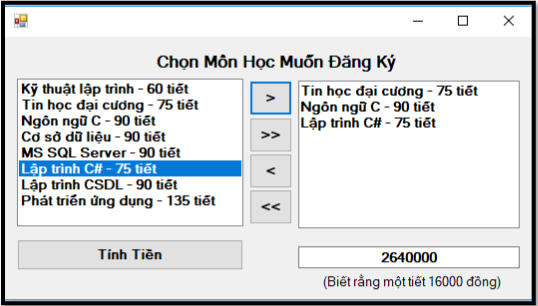
**Một số thuộc tính thường dùng:**

| **Thuộc tính** | **Mô tả** |
| --- | --- |
| **Text** | Trả về nội dung dòng dữ liệu đang hiển thị trên ComboBox |
| **DropDownStyle** | Quy định định dạng của ComboBox, nhận một tring các giá trị:  - Simple: hiển thị theo dạng ListBox + TextBox có thể chọn dữ liệu từ ListBox hoặc nhập mới vào TextBox  - DropDownList: Chỉ cho phép chọn dữ liệu trong ComboBox  - DropDown: Giá trị mặc định, có thể chọn hoặc nhập mới mục dữ liệu vào ComboBox |
| **Items** | Trả về các mục chứa trong ComboBox |
| **DropDownHeight** | Thiết lập chiều cao tối đa khi sổ xuống của ComboBox |
| **DropDownWidth** | Thiết lập độ rộng của mục chọn trong ComboBox |
| **SelectedIndex** | Lấu chỉ số mục được chọn, chỉ số mục đầu tiên là 0 |
| **SelectedItem** | Trả về mục được chọn |
| **SelectedText** | Lấy chuỗi hiển thị của mục chọn trên ComboBox |
| **DataSource** | Chọn tập dữ liệu điền vào ComboBox. Tập dữ liệu có thể là mảng, chuỗi, ArrayList,... |
| **DisplayMember** | Gán dữ liệu thành viên sẽ hiển thị trên ComboBox |
| **ValueMember** | Thuộc tính này chỉ định dữ liệu thành viên sẽ cung cấp giá trị cho ComboBox |
| **SelectedValue** | Trả về giá trị của mục chọn (ValueMember) nếu ComboBox có liên kết dữ liệu. Nếu không liên kết dữ liệu hoặc ValueMember không được thiết lập thì giá trị SelectedValue là giá trị chuỗi của thuộc tính SelectedItem |

Trong **ComboBox**có một sự kiện là ***SelectedIndexChanged***, sự kiện này xảy ra khi thay đổi mục chọn trong ComboBox.

2. ListBox

**ListBox**được dùng để hiển thị một danh sách các lựa chọn, người dùng có thể chọn một hoặc nhiều lựa chọn cùng lúc.



**Một số thao tác với ListBox:**

* **Add()**: Thêm một mục chọn vào cuối danh sách **ListBox**.
* **Insert()**: Chèn thêm mục vào vào vị trí **i**.
* **Count**: Trả về số mục chọn hiện đang có.
* **Item()**: Trả về mục chọn ở vị trí **i**.
* **Remove()**: Bỏ mục chọn.
* **RemoveAt()**: Bỏ mục chọn tại vị trí **i**.
* **Contains()**: Trả về True nếu có mục chọn trong danh sách và trả về False nếu không có mục chọn trong danh sách.
* **Clear**: Xóa tất cả các mục chọn.
* **IndexOf()**: Trả về vị trí mục chọn trong danh sách, nếu không tìm thấy sẽ trả về -1.

**Một số thuộc tính của ListBox thường dùng:**

| **Thuộc tính** | **Mô tả** |
| --- | --- |
| **DataSource** | Chọn tập dữ liệu điền vào ListBox. Tập dữ liệu có thể là mảng, chuỗi, ArrayList,... |
| **DisplayMember** | Dữ liệu thành viên sẽ được hiển thị trên ListBox |
| **ValueMember** | Thuộc tính này chỉ định dẽ liệu thành viên sẽ cung cấp giá trị cho ListBox |
| **SelectedValue** | Trả về giá trị của mục chọn nếu ListBox có liên kết dữ liệu. Nếu không liên kết với dữ liệu hoặc thuộc tính ValueMember không được thiết lập thì giá trị thuộc tính SelectedValue là giá trị chuỗi của thuộc tính SelectedItem |
| **Items** | Các mục chứa trong ListBox |
| **SelectedItem** | Trả về mục được chọn |
| **SelectedIndex** | Lấy chỉ số mục được chọn, chỉ số mục chọn đầu tiên là 0 |
| **SelectionMode** | Cho phép chọn một hoặc nhiều dòng dữ liệu trên ListBox, bao gồm:  - *One*: Chỉ chọn một giá trị  - *MultiSimple*: Cho phép chọn nhiều, chọn bằng cách Click vào mục chọn, bỏ chọn bằng cách Click vào mục đã chọn  - *MultiExtended*: Chọn nhiều bằng cách nhấn kết hợp với Shift hoặc Ctrl |
| **SelectedItems** | Được sử dụng khi SelectionMode là MultiSimple hoặc MultiExtended. Thộc tính SelectedItems chứa các chỉ số của các dòng dữ liệu được chọn |
| **SelectedItems** | Được sử dụng khi SelectionMode là MultiSimple hoặc MultiExtended. Thuộc tính SelectedItems chứa các chỉ số của các dòng dữ liệu được chọn |

Trong ListBox có một sự kiện được sử dụng rất nhiều đó chính là ***SelectedIndexChanged***, sự kiện này xảy ra khi thay đổi mục chọn trong ListBox.

* 1. progressbar – timer
  2. listview
  3. tree view
  4. datetimepicker – monthcalendar
  5. menustrip
  6. dialog file