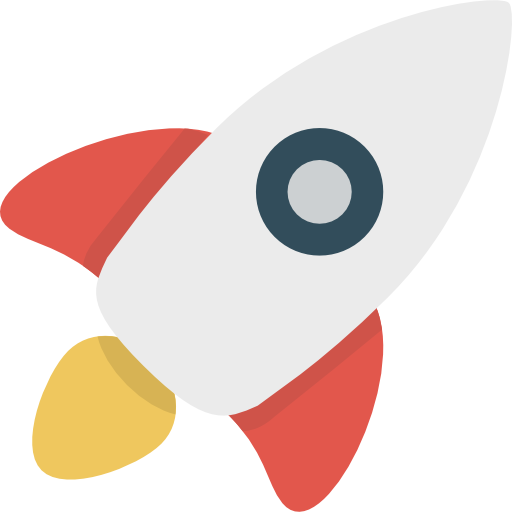
**Đại học Khoa học Tự Nhiên – TPHCM**

****

**LẬP TRÌNH QUẢN LÝ ỨNG DỤNG**

**Giáo viên phụ trách: Ngô Chách Đức**

🙠 🕮 🙢

Thành viên nhóm:

1. **Ngô Ngọc Thiện 1461641**
2. **Hồ Tuấn Sang 1461576**
3. **Dương Gia Hoà 1461350**
4. **Đỗ Nguyễn Minh Luân 1461458**

**Câu 1:** Phân biệt khái niệm Trường dữliệu (Field) và Thuộc tính (Property) trong.NET Framework.

Trường dữ liệu (Field)

* Field là thuộc tính nằm trong lớp, có thể là private,public ,protected.Nhưng field thường để dạng private để đảm bảo cho việc dữ liệu trong lớp được đảm bảo. Đảm bảo tính đóng gói và tính giấu kín thông tin thuộc tính bên trong lớp.

Thuộc tính (Property)

* Property vẫn là thuộc tính trong lớp, nhưng thuộc tính này có kiểu truy xuất là public.
* Property sẽ bao bọc field nó cần biểu diễn. Thuộc tính Property sẽ cho phép gán hoặc lấy dữ liệu của field từ bên ngoài. Ngoài ra:
  + Property còn có lợi ích ở việc ràng buộc dữ liệu khi nhập vào field bằng cách cài đặt điều kiện dữ liệu vào trong phương thức set của Property, từ đó đảm bảo cho đối tượng của lớp không bị hư hại (lớp chứa dữ liệu bị sai).
  + Khi chỉ muốn 1 trường dữ liệu chỉ được đọc mà không được ghi, thì trong thuộc tính chỉ cần cài đặt phương thức get().
* Do Property dạng public nên ta có thể truy xuất bên ngoài Class. Nhưng mục đích của Property là bao bọc field để nên những thay đổi của Field sẽ thông qua Property nên Field sẽ được bảo mật hơn.

Câu 2: So sánh giữa Cấu trúc(Structure) và Lớp (Class) trong .NET Framework? Đồng thời so sánh với Struct trong C/C++

Cấu trúc(Structure)

* Cấu trúc là một kiểu dữ liệu do người dùng tự định nghĩa (user-defined data type) dùng để tập hợp các thành phần dữ liệu có quan hệ với nhau thành một cấu trúc duy nhất. Structure tương tự như class chẳng hạn nó có các fields và các methods,…
* Structure mang kiểu giá trị(value type).
* Khi một cấu trúc được tạo ra, biến mà các cấu trúc được phân công giữ dữ liệu thực tế của cấu trúc. Khi các cấu trúc được gán cho một biến mới, nó được sao chép. Các biến mới và biến ban đầu do đó có hai bản sao riêng biệt của cùng một dữ liệu. Thay đổi thực hiện được một bản sao không ảnh hưởng đến các bản sao khác.
* Structure không được khai báo Constructor và Destructor.
* Structure dùng cho các kiểu hạng nhẹ và thời gian tồn tại trong bộ nhớ ngắn chủ yếu là dữ liệu mà không được dự định sẽ được sửa đổi sau khi được tạo ra cấu trúc.
* Structure không có kế thừa và đa hình
* Ngoài ra tầm vực của các biến của Structure luôn là public nên không đảm báo về mặt bảo mật dữ liệu.

struct ExampleStruct

{

public ExampleStruct(string a)

{

}

public void method()

{

}

public int datamember;

}

Lớp (Class)

* Class mang kiểu tham chiếu(reference type).
* Khi một đối tượng của lớp được tạo ra, biến mà đối tượng được phân công giữ chỉ có một tham chiếu tới bộ nhớ mà. Khi các đối tượng tham chiếu được gán cho một biến mới, các biến mới đề cập đến các đối tượng ban đầu. Thay đổi thực hiện thông qua một biến đổi được phản ánh trong các biến khác bởi vì họ đều đề cập đến cùng một dữ liệu.
* Có tính chất hướng đối tượng.
* Class được dùng cho các đối tượng lớn và thời gian tồn tại trong bộ nhớ lâu.
* Class có thể khai báo Constructor và Destructor.
* Được sử dụng để mô hình hành vi phức tạp hơn, hoặc dữ liệu được dự định sẽ được sửa đổi sau khi một đối tượng lớp được tạo ra.
* Các biến của Class có thể giới hạn tầm vực như là private, public, protected.

Structure C/C++

* Giống như với class C++ nhưng tất cả các biến trong struct đều có thể truy cập vì nó là public
* Các biến đều là mặc định là public nhưng cũng có thể truy cập để sửa đổi
* Struct là thuộc dạng value type
* Shông có default constructor và default destructor

struct PERSON {

int age;

long ss;

float weight;

char name[25];

}

**Note**: khi truyền 1 cấu trúc vào 1 hàm thì chương trình tạo ra 1 bản sao đối tượng và truyền vào. Còn khi truyền 1 class vào hàm thì chương trình sẽ truyền con trỏ đối tượng vào và mọi thay đổi sẽ ảnh hưởng trực tiếp đến bản gốc.

Câu 3: Ý nghĩa của các thành phần của cấu trúc DateTime

|  |  |
| --- | --- |
| **Fields** | |
| **MaxValue** | Thể hiện giá trị lớn nhất có thể của hàm DateTime  (Max là 3:59:59.9999999 UTC, December 31, 9999) đúng chính xác 100 nano giây trước khi qua ngày 00:00:00 UTC, January 1, 10000 |
| **MinValue** | Thể hiện giá trị nhỏ nhất có thể của hàm DateTime  (Min là 00:00:00.0000000 UTC, January 1, 0001) |
| **Properties** | |
| **Date** | Lấy giá trị date (dd-mm-yyyy) của 1 thể hiện của đối tượng DateTime |
| **Day** | Lấy ngày của tháng của 1 thể hiện của đối tượng DateTime |
| **DayOfWeek** | Lấy ngày của tuần của 1 thể hiện của đối tượng DateTime |
| **DayOfYear** | Lấy ngày của năm của 1 thể hiện của đối tượng DateTime |
| **Hour** | Lấy giờ của 1 thể hiện của đối tượng DateTime |
| **Millisecond** | Lấy mili giây của 1 thể hiện của đối tượng DateTime |
| **Minute** | Lấy phút của 1 thể hiện của đối tượng DateTime |
| **Month** | Lấy tháng của 1 thể hiện của đối tượng DateTime |
| **Now** | Lấy DateTime hiện trên hệ thống. |
| **Second** | Lấy giây của 1 thể hiện của đối tượng DateTime |
| **TimeOfDay** | Lấy bộ giờ (hh:mm:ss) của 1 thể hiện của đối tượng DateTime |
| **ToDay** | Lấy Time được cài trên hệ thống. |
| **Year** | Lấy năm của1 thể hiện của đối tượng DateTime |
| **Methods** | |
| **AddDays** | Cộng thêm ngày vào 1 thể hiện của đối tượng DateTime |
| **AddHours** | Cộng thêm giờ vào 1 thể hiện của đối tượng DateTime |
| **AddMilliseconds** | Cộng thêm mili giây vào 1 thể hiện của đối tượng DateTime |
| **AddMinutes** | Cộng thêm phút. vào 1 thể hiện của đối tượng DateTime |
| **AddMonths** | Cộng thêm tháng. vào 1 thể hiện của đối tượng DateTime |
| **AddYears** | Cộng thêm năm vào 1 thể hiện của đối tượng DateTime |
| **Compare** | Compare(Date1, Date2)  Kết quả trả ra:  + Nhỏ hơn 0 khi Date1 sớm hơn Date2  + Bằng 0 khi Data1 chính là Date2  + Lớn hơn 0 khi Date1 trễ hơn Date2 |
| **CompareTo** | So sánh thể hiện hiện tại với 1 thể hiện khác  Date1.Compareto(Date2)  + Nhỏ hơn 0 nếu Date1 sớm hơn Date2  + Lơn hơn 0 nếu Date1 trễ hơn Date2 hoặc Date2 là null  + Bằng 0 nếu Date1 và Date2 bằng nhau |
| **DaysInMonth** | Ngày trong tháng |
| **IsLeapYear** | Là năm nhuận. |
| **Subtract** | Trừ khoảng thời gian (- 1 TimeSpan). |
| **ToLongDateString** | Long date pattern: "dddd, MMMM dd, yyyy"  Long date string: "Wednesday, May 16, 2001" |
| **ToLongTimeString** | Long time pattern: "h:mm:ss tt"  Long time string: "3:02:15 AM" |
| **ToShortDateString** | Short date pattern: "M/d/yyyy"  Short date string: "5/16/2001" |
| **ToShortTimeString** | Short time pattern: "h:mm tt"  Short time string: "3:02 AM" |

**Câu 4:**

**Cách sử dụng của phương thức** [**ToString(String, IFormatProvider)**](https://msdn.microsoft.com/en-us/library/d830955a(v=vs.110).aspx)

* Phương thức này sẽ nhận 2 Parameter là:
* String (chuỗi định dạng để tùy chỉnh lại dạng đầu ra của chuỗi): vd: "#" => 1234.5678 ("#####") -> 1235
* IformatProvider (1 đối tượng cung cấp các định dạng theo từng văn hóa (vietnam, english , chinese,...)):

public interface IFormatProvider {

object GetFormat(Type formatType);

}

Ta thấy IFormatProvider chính là class Kế thừa đa hình từ các Class được quy định chuẩn chẳng hạn CultureInfo("en-US"),

CultureInfo("fr-FR"),

CultureInfo("es-ES"),

CultureInfo("de-DE"),

**Ví dụ: Giá trị trả về của hàm này là 1 chuỗi mới được định dạng theo quy định chuẩn số và theo từng văn hóa**

Dim cultures() As CultureInfo = {CultureInfo.CreateSpecificCulture("en-US"),

CultureInfo.CreateSpecificCulture("fr-FR"),

CultureInfo.CreateSpecificCulture("es-ES")}

Dim positiveNumber As Integer = 1679

Dim negativeNumber As Integer = -3045

Dim specifiers() As String = {"G", "C", "D8", "E2", "F", "N", "P", "X8"}

For Each specifier As String In specifiers

For Each culture As CultureInfo In Cultures

' Display values with "G" format specifier.

Console.WriteLine("{0} format using {1} culture: {2, 16} {3, 16}", \_

specifier, culture.Name, \_

positiveNumber.ToString(specifier, culture), \_

negativeNumber.ToString(specifier, culture))

Next

**Kết quả:**

'E2 format using en-US culture: 1.68E+003 -3.05E+003

'E2 format using fr-FR culture: 1,68E+003 -3,05E+003

'E2 format using es-ES culture: 1,68E+003 -3,05E+003

* **Người dùng còn có thể sử dụng hệ thống cung cấp định dạng chuẩn**

**Standard Numeric Format String:**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  | | --- | --- | --- | --- | | **Format specifier** | **Name** | **Description** | **Examples** | | "C" or "c" | Currency | Result: A currency value.  Supported by: All numeric types.  Precision specifier: Number of decimal digits.  Default precision specifier: Defined by[NumberFormatInfo.CurrencyDecimalDigits](https://msdn.microsoft.com/en-us/library/system.globalization.numberformatinfo.currencydecimaldigits(v=vs.110).aspx).  More information: [The Currency ("C") Format Specifier](https://msdn.microsoft.com/en-us/library/dwhawy9k(v=vs.110).aspx#CFormatString). | 123.456 ("C", en-US) -> $123.46  123.456 ("C", fr-FR) -> 123,46 €  123.456 ("C", ja-JP) -> ¥123  -123.456 ("C3", en-US) -> ($123.456)  -123.456 ("C3", fr-FR) -> -123,456 €  -123.456 ("C3", ja-JP) -> -¥123.456 | | "D" or "d" | Decimal | Result: Integer digits with optional negative sign.  Supported by: Integral types only.  Precision specifier: Minimum number of digits.  Default precision specifier: Minimum number of digits required.  More information: [The Decimal("D") Format Specifier](https://msdn.microsoft.com/en-us/library/dwhawy9k(v=vs.110).aspx#DFormatString). | 1234 ("D") -> 1234  -1234 ("D6") -> -001234 | | "E" or "e" | Exponential (scientific) | Result: Exponential notation.  Supported by: All numeric types.  Precision specifier: Number of decimal digits.  Default precision specifier: 6.  More information: [The Exponential ("E") Format Specifier](https://msdn.microsoft.com/en-us/library/dwhawy9k(v=vs.110).aspx#EFormatString). | 1052.0329112756 ("E", en-US) -> 1.052033E+003  1052.0329112756 ("e", fr-FR) -> 1,052033e+003  -1052.0329112756 ("e2", en-US) -> -1.05e+003  -1052.0329112756 ("E2", fr\_FR) -> -1,05E+003 | | "F" or "f" | Fixed-point | Result: Integral and decimal digits with optional negative sign.  Supported by: All numeric types.  Precision specifier: Number of decimal digits.  Default precision specifier: Defined by[NumberFormatInfo.NumberDecimalDigits](https://msdn.microsoft.com/en-us/library/system.globalization.numberformatinfo.numberdecimaldigits(v=vs.110).aspx).  More information: [The Fixed-Point ("F") Format Specifier](https://msdn.microsoft.com/en-us/library/dwhawy9k(v=vs.110).aspx#FFormatString). | 1234.567 ("F", en-US) -> 1234.57  1234.567 ("F", de-DE) -> 1234,57  1234 ("F1", en-US) -> 1234.0  1234 ("F1", de-DE) -> 1234,0  -1234.56 ("F4", en-US) -> -1234.5600  -1234.56 ("F4", de-DE) -> -1234,5600 | | "G" or "g" | General | Result: The more compact of either fixed-point or scientific notation.  Supported by: All numeric types.  Precision specifier: Number of significant digits.  Default precision specifier: Depends on numeric type.  More information: [The General ("G") Format Specifier](https://msdn.microsoft.com/en-us/library/dwhawy9k(v=vs.110).aspx#GFormatString). | -123.456 ("G", en-US) -> -123.456  -123.456 ("G", sv-SE) -> -123,456  123.4546 ("G4", en-US) -> 123.5  123.4546 ("G4", sv-SE) -> 123,5  -1.234567890e-25 ("G", en-US) -> -1.23456789E-25  -1.234567890e-25 ("G", sv-SE) -> -1,23456789E-25 | | "N" or "n" | Number | Result: Integral and decimal digits, group separators, and a decimal separator with optional negative sign.  Supported by: All numeric types.  Precision specifier: Desired number of decimal places.  Default precision specifier: Defined by[NumberFormatInfo.NumberDecimalDigits](https://msdn.microsoft.com/en-us/library/system.globalization.numberformatinfo.numberdecimaldigits(v=vs.110).aspx).  More information: [The Numeric ("N") Format Specifier](https://msdn.microsoft.com/en-us/library/dwhawy9k(v=vs.110).aspx#NFormatString). | 1234.567 ("N", en-US) -> 1,234.57  1234.567 ("N", ru-RU) -> 1 234,57  1234 ("N1", en-US) -> 1,234.0  1234 ("N1", ru-RU) -> 1 234,0  -1234.56 ("N3", en-US) -> -1,234.560  -1234.56 ("N3", ru-RU) -> -1 234,560 | | "P" or "p" | Percent | Result: Number multiplied by 100 and displayed with a percent symbol.  Supported by: All numeric types.  Precision specifier: Desired number of decimal places.  Default precision specifier: Defined by[NumberFormatInfo.PercentDecimalDigits](https://msdn.microsoft.com/en-us/library/system.globalization.numberformatinfo.percentdecimaldigits(v=vs.110).aspx).  More information: [The Percent ("P") Format Specifier](https://msdn.microsoft.com/en-us/library/dwhawy9k(v=vs.110).aspx#PFormatString). | 1 ("P", en-US) -> 100.00 %  1 ("P", fr-FR) -> 100,00 %  -0.39678 ("P1", en-US) -> -39.7 %  -0.39678 ("P1", fr-FR) -> -39,7 % | | "R" or "r" | Round-trip | Result: A string that can round-trip to an identical number.  Supported by: [Single](https://msdn.microsoft.com/en-us/library/system.single(v=vs.110).aspx), [Double](https://msdn.microsoft.com/en-us/library/system.double(v=vs.110).aspx), and [BigInteger](https://msdn.microsoft.com/en-us/library/system.numerics.biginteger(v=vs.110).aspx).  Precision specifier: Ignored.  More information: [The Round-trip ("R") Format Specifier](https://msdn.microsoft.com/en-us/library/dwhawy9k(v=vs.110).aspx#RFormatString). | 123456789.12345678 ("R") -> 123456789.12345678  -1234567890.12345678 ("R") -> -1234567890.1234567 | | "X" or "x" | Hexadecimal | Result: A hexadecimal string.  Supported by: Integral types only.  Precision specifier: Number of digits in the result string.  More information: [The HexaDecimal ("X") Format Specifier](https://msdn.microsoft.com/en-us/library/dwhawy9k(v=vs.110).aspx#XFormatString). | 255 ("X") -> FF  -1 ("x") -> ff  255 ("x4") -> 00ff  -1 ("X4") -> 00FF | | Any other single character | Unknown specifier | Result: Throws a [FormatException](https://msdn.microsoft.com/en-us/library/system.formatexception(v=vs.110).aspx) at run time. |  | |

**Ví dụ:**

int value = -16325;

string specifier;

// Use standard numeric format specifier.

specifier = "G";

Console.WriteLine("{0}: {1}", specifier, value.ToString(specifier));

// Displays: G: -16325

specifier = "C";

Console.WriteLine("{0}: {1}", specifier, value.ToString(specifier));

// Displays: C: ($16,325.00)

specifier = "D8";

Console.WriteLine("{0}: {1}", specifier, value.ToString(specifier));

// Displays: D8: -00016325

specifier = "E4";

Console.WriteLine("{0}: {1}", specifier, value.ToString(specifier));

// Displays: E4: -1.6325E+004

specifier = "e3";

Console.WriteLine("{0}: {1}", specifier, value.ToString(specifier));

// Displays: e3: -1.633e+004

specifier = "F";

Console.WriteLine("{0}: {1}", specifier, value.ToString(specifier));

// Displays: F: -16325.00

specifier = "N";

Console.WriteLine("{0}: {1}", specifier, value.ToString(specifier));

// Displays: N: -16,325.00

specifier = "P";

Console.WriteLine("{0}: {1}", specifier, (value/100000).ToString(specifier));

// Displays: P: -16.33 %

specifier = "X";

Console.WriteLine("{0}: {1}", specifier, value.ToString(specifier));

// Displays: X: FFFFC03B

// Use custom numeric format specifiers.

specifier = "0,0.000";

Console.WriteLine("{0}: {1}", specifier, value.ToString(specifier));

// Displays: 0,0.000: -16,325.000

specifier = "#,#.00#;(#,#.00#)";

Console.WriteLine("{0}: {1}", specifier, (value\*-1).ToString(specifier));

// Displays: #,#.00#;(#,#.00#): 16,325.00

* **Ngoài ra Người dùng còn có thể sử định dạng tự định nghĩa phương thức toString(String) cho các Class khác bằng cách quy định String input để trả ra output là gì.**

**Câu 5:** Trình bày một số Thuộc tính, Phương thức thông dụng của lớp System.String và lớp System.Collections.Generic.List (Of T). Đối với mỗi phương thức, hãy cho ví dụcách sửdụng

* **Cấu trúc của class String:**

Public NotInheritable Class String

Implements IComparable, ICloneable, IConvertible, IEnumerable,

IComparable(Of String), IEnumerable(Of Char), IEquatable(Of String)

|  |  |  |
| --- | --- | --- |
| String | | |
| Phương thức | Ý nghĩa | Ví dụ |
| Clone() | Trả về 1 tham chiếu đến chuỗi ban đầu | Puclic Function Clone As Object |
| Compare(String, Int32, String, Int32, Int32) | So sánh các chuỗi con trong 2 chuỗi lớn và trả về 1 số để xác định vị trí tương đối của 2 chuỗi con theo thứ tự sắp xếp | Dim str1 As [String] = "machine"  Dim str2 As [String] = "device"  Dim str As [String]  Dim result As Integer  Console.WriteLine()  Console.WriteLine("str1 = '{0}', str2 = '{1}'", str1, str2)  result = [String].Compare(str1, 2, str2, 0, 2)  str = IIf(result < 0, "less than", IIf(result > 0, "greater than", "equal to"))  Console.Write("Substring '{0}' in '{1}' is ", str1.Substring(2, 2), str1)  Console.Write("{0} ", str)  Console.WriteLine("substring '{0}' in '{1}'.", str2.Substring(0, 2), str2)  Result:  str1 = 'machine', str2 = 'device'  Substring 'ch' in 'machine' is less than substring 'de' in 'device'. |
| Compare(String, Int32, String, Int32, Int32, Boolean) | So sánh các chuỗi con trong 2 chuỗi lớn, bỏ qua các trường hợp chữ hoa hoặc chú ý đến trường hợp chữ hoa ,và trả về 1 số để xác định vị trí tương đối của 2 chuỗi con theo thứ tự sắp xếp | Dim str1 As [String] = "MACHINE"  Dim str2 As [String] = "machine"  Dim str As [String]  Dim result As Integer  Console.WriteLine()  Console.WriteLine("str1 = '{0}', str2 = '{1}'", str1, str2)  Console.WriteLine("Ignore case:")  result = [String].Compare(str1, 2, str2, 2, 2, True)  str = IIf(result < 0, "less than", IIf(result > 0, "greater than", "equal to"))  Console.Write("Substring '{0}' in '{1}' is ", str1.Substring(2, 2), str1)  Console.Write("{0} ", str)  Console.WriteLine("substring '{0}' in '{1}'.", str2.Substring(2, 2), str2)  Result:  str1 = 'MACHINE', str2 = 'machine'  Substring 'CH' in 'MACHINE' is equal to substring 'ch' in 'machine'. |
| [Compare(String, Int32, String, Int32, Int32, Boolean, CultureInfo)](https://msdn.microsoft.com/en-us/library/3h045cx3(v=vs.110).aspx) | So sánh các chuỗi con trong 2 chuỗi lớn, bỏ qua hoặc chú ý đến các trường hợp dựa theo thông tin văn hóa cụ thể đc truyền vào, và trả về 1 số để xác định vị trí tương đối của 2 chuỗi con theo thứ tự sắp xếp | Dim str1 As [String] = "MACHINE"  Dim str2 As [String] = "machine"  Dim str As [String]  Dim result As Integer  Console.WriteLine()  Console.WriteLine("str1 = '{0}', str2 = '{1}'", str1, str2)  Console.WriteLine("Ignore case, Turkish culture:")  result = [String].Compare(str1, 4, str2, 4, 2, True, New CultureInfo("tr-TR"))  str = IIf(result < 0, "less than", IIf(result > 0, "greater than", "equal to"))  Console.Write("Substring '{0}' in '{1}' is ", str1.Substring(4, 2), str1)  Console.Write("{0} ", str)  Console.WriteLine("substring '{0}' in '{1}'.", str2.Substring(4, 2), str2)  Result:  str1 = 'MACHINE', str2 = 'machine'  Substring 'IN' in 'MACHINE' is less than substring 'in' in 'machine'. |
| Compare(String, String) | So sánh 2 chuỗi khác nhau và trả về 1 số để xác định vị trí tương đối của 2 chuỗi đó theo thứ tự sắp xếp | Dim s1 As String = "ani" + ChrW(&h00AD) + "mal"  Dim s2 As String = "animal"  Console.WriteLine("Comparison of '{0}' and '{1}': {2}",  s1, s2, String.Compare(s1, s2))  Result: Comparison of 'ani-mal' and 'animal': 0 |
| [Compare(String, String, Boolean)](https://msdn.microsoft.com/en-us/library/zkcaxw5y(v=vs.110).aspx) | So sánh 2 object xem có nằm trước hay nằm sau hay xuất hiện tại vị trí nào đó với 1 chuỗi cụ thể | Dim strFirst As String = "Goodbye"  Dim strSecond As String = "Hello"  Dim strThird As String = "a small string"  Dim strFourth As String = "goodbye"  Dim cmpVal As Integer = str1.CompareTo(str2)  If cmpVal = 0 Then  ' The values are the same.  Return "The strings occur in the same position in the sort order."  ElseIf cmpVal < 0 Then  Return "The first string precedes the second in the sort order."  Else  Return "The first string follows the second in the sort order."  End If |
| [Concat(IEnumerable<String>)](https://msdn.microsoft.com/en-us/library/dd784338(v=vs.110).aspx) | Nối các thành phần thuộc dạng Ienmerable kiểu String | Dim maxPrime As Integer = 100  Dim primeList As IEnumerable(Of String) = GetPrimes(maxPrime)  Console.WriteLine("Primes less than {0}:", maxPrime)  Console.WriteLine(" {0}", String.Concat(primeList)) |
| [Concat(Object)](https://msdn.microsoft.com/en-us/library/khca9w90(v=vs.110).aspx) | Tạo ra 1 chuỗi mới từ 1 object cụ thể | Dim i As Integer = - 123  Dim o As [Object] = i  Dim objs() As [Object] = {-123, -456, -789}  Console.WriteLine("Concatenate 1, 2, and 3 objects:")  Console.WriteLine("1) {0}", [String].Concat(o))  Console.WriteLine("2) {0}", [String].Concat(o, o))  Console.WriteLine("3) {0}", [String].Concat(o, o, o)) |
| [Concat(Object[])](https://msdn.microsoft.com/en-us/library/k9c94ey1(v=vs.110).aspx) | Tạo 1 chuỗi mới từ các thành phần trong mảng object | Dim t1 As New Test1()  Dim t2 As New Test2()  Dim i As Integer = 16  Dim s As String = "Demonstration"  Dim o As Object() = {t1, i, t2, s}  Console.WriteLine(String.Concat(o)) |
| [Concat(String, String)](https://msdn.microsoft.com/en-us/library/a6d350wd(v=vs.110).aspx) | Nối 2 chuỗi | Dim fName As String = "Simon"  Dim mName As String = "Jake"  Dim lName As String = "Harrows"  Console.WriteLine("Welcome to this page, '{0}'!", \_  String.Concat(String.Concat(fName, mName), lName)) |
| [Concat(String[])](https://msdn.microsoft.com/en-us/library/0wkb0y3w(v=vs.110).aspx) | Nối mảng chuỗi | Dim s As String() = { "hello ", "and ", "welcome ", "to ",  "this ", "demo! "}  ' Put all the strings together.  Console.WriteLine(String.Concat(s)) |
| [Contains(String)](https://msdn.microsoft.com/en-us/library/dy85x1sa(v=vs.110).aspx) | Trả về giá trị xác định chuỗi con có đc chứa trong chuỗi hay không | Dim s As String = "This is a string."  Dim sub1 As String = "this"  Console.WriteLine("Does '{0}' contain '{1}'?", s, sub1)  Dim comp As StringComparison = StringComparison.Ordinal  Console.WriteLine(" {0:G}: {1}", comp, s.Contains(sub1, comp)) |
| [Copy(String)](https://msdn.microsoft.com/en-us/library/system.string.copy(v=vs.110).aspx) | Tạo 1 chuỗi mới có giá trị giống với chuỗi ban đầu | Dim str1 As String = "abc"  Dim str2 As String = "xyz"  Console.WriteLine("After String.Copy...")  str2 = String.Copy(str1)  Console.WriteLine("str1 = '{0}'", str1)  Console.WriteLine("str2 = '{0}'", str2) |
| [CopyTo(Int32, Char[], Int32, Int32)](https://msdn.microsoft.com/en-us/library/system.string.copyto(v=vs.110).aspx) | Copy 1 số lượng các kí tự tại 1 vị trí trong chuỗi đến 1 vị trí khác trong mảng các kí tự Unicode | Dim strSource As String = "changed"  Dim destination As Char() = {"T"c, "h"c, "e"c, " "c, "i"c, "n"c, "i"c, \_  "t"c, "i"c, "a"c, "l"c, " "c, "a"c, "r"c, "r"c, "a"c, "y"c}    ' Embed the source string in the destination string  strSource.CopyTo(0, destination, 4, strSource.Length) |
| [EndsWith(String)](https://msdn.microsoft.com/en-us/library/2333wewz(v=vs.110).aspx) | Xác định phần kết thúc của chuỗi có trùng với chuỗi khác hay ko | Dim strings() As String = { "This is a string.", "Hello!",  "Nothing.", "Yes.", "randomize" }  For Each value In strings  Dim endsInPeriod As Boolean = value.EndsWith(".")  Console.WriteLine("'{0}' ends in a period: {1}",  value, endsInPeriod)  Next |
| [EndsWith(String, Boolean, CultureInfo)](https://msdn.microsoft.com/en-us/library/t9h2fbth(v=vs.110).aspx) | Xác định phần kết thúc của chuỗi có trùng với chuỗi khác hay không khi sử dụng so sánh tùy theo culture | Dim msg1 As String = "Search for the target string ""{0}"" in the string ""{1}""." & vbCrLf  Dim msg2 As String = "Using the {0} - ""{1}"" culture:"  Dim msg3 As String = " The string to search ends with the target string: {0}"  Dim result As Boolean = False  Dim msg1 As String = "Search for the target string ""{0}"" in the string ""{1}""." & vbCrLf  Dim msg2 As String = "Using the {0} - ""{1}"" culture:"  Dim msg3 As String = " The string to search ends with the target string: {0}"  Dim result As Boolean = False  Dim ci As CultureInfo  Dim capitalARing As String = "Å"  Dim xyzARing As String = "xyz" & "å"  ci = New CultureInfo("en-US")  result = xyzARing.EndsWith(capitalARing, False, ci)  Console.WriteLine(msg3, result) |
| [Equals(Object)](https://msdn.microsoft.com/en-us/library/fkfd9eh8(v=vs.110).aspx) | So sánh 2 object có cùng dạng, cùng giá trị hay không | Dim sb As New StringBuilder("abcd")  Dim str1 As [String] = "abcd"  Dim str2 As [String] = Nothing  Dim o2 As [Object] = Nothing  Console.WriteLine(" Is str1 equal to sb?: {0}", str1.Equals(sb)) |
| [Equals(String)](https://msdn.microsoft.com/en-us/library/858x0yyx(v=vs.110).aspx) | So sánh chuỗi ban đầu có cùng giá trị với chuỗi đc truyền vào hay không | Dim sb As New StringBuilder("abcd")  Dim str1 As [String] = "abcd"  Dim str2 As [String] = Nothing  Dim o2 As [Object] = Nothing  Console.WriteLine(" Is str1 equal to str2?: {0}", str1.Equals(str2)) |
| [Equals(String, String)](https://msdn.microsoft.com/en-us/library/1hkt4325(v=vs.110).aspx) | So sánh 2 chuỗi có cùng giá trị hay không | Dim sb As New StringBuilder("abcd")  Dim str1 As [String] = "abcd"  Dim str2 As [String] = Nothing  Dim o2 As [Object] = Nothing  Console.WriteLine(" Is str1 equal to str2?: {0}", [String].Equals(str1, str2)) |
| Format(IFormatProvier, String, Object) | Thay thế các mục định dạng hoặc các mục trong một chuỗi chỉ định với chuỗi đại diện của các đối tượng tương ứng. Một thông tin định dạng nguồn cung cấp tham số nền văn hóa cụ thể. | Dim acctNumber As Integer = 79203159  Console.WriteLine(String.Format(New CustomerFormatter, "{0}", acctNumber)) |
| Format(IFormatProvier, String, Object, Object) | Thay thế các mục định dạng trong một chuỗi chỉ định với chuỗi đại diện của hai đối tượng quy định.Một thông tin định dạng nguồn cung cấp tham số nền văn hóa cụ thể. |  |
| Format(IFormatProvier, String, Object, Object, Object) | Thay thế các mục định dạng trong một chuỗi chỉ định với chuỗi đại diện của ba đối tượng quy định.Một nền văn hóa cụ thể nguồn cung cấp thông tin định dạng. | Dim n As Integer = 10  Dim value As Double = 16.935  Dim day As DateTime = Date.Now  Dim provider As New InterceptProvider()  Console.WriteLine(String.Format(provider, "{0:N0}: {1:C2} on {2:d}", n, value, day)) |
| Format(IFormatProvier, String, Object()) | Thay thế các mục định dạng trong một chuỗi quy định với cơ quan đại diện chuỗi của các đối tượng tương ứng trong một mảng được chỉ định. Một thông tin định dạng nguồn cung cấp tham số nền văn hóa cụ thể. | Dim values() As Object = { 1603, 1794.68235, 15436.14 }  Dim result As String  For Each value In values  result = String.Format("{0,12:C2} {0,12:E3} {0,12:F4} {0,12:N3} {1,12:P2}",value, CDbl(value) / 10000) |
| Format(String, Object) | Thay thế một hoặc nhiều mục định dạng trong một chuỗi chỉ định với chuỗi đại diện của một đối tượng cụ thể. | Dim pricePerOunce As Decimal = 17.36d  Dim s As String = String.Format("The current price is {0} per ounce.",  pricePerOunce)  Result: The current price is 17.36 per ounce. |
| Format(String, Object, Object) | Thay thế các mục định dạng trong một chuỗi chỉ định với chuỗi đại diện của hai đối tượng quy định. | Dim s As String = String.Format("At {0}, the temperature is {1}°C.",  Date.Now, 20.4)  ' Output similar to: 'At 4/10/2015 9:29:41 AM, the temperature is 20.4°C.' |
| Format (String, Object, Object, Object) | Thay thế các mục định dạng trong một chuỗi chỉ định với chuỗi đại diện của ba đối tượng quy định | Dim formatString As String = " {0,10} ({0,8:X8})" + vbCrLf + \_  "And {1,10} ({1,8:X8})" + vbCrLf + \_  " = {2,10} ({2,8:X8})"  Dim value1 As Integer = 16932  Dim value2 As Integer = 15421  Dim result As String = String.Format(formatString, \_  value1, value2, value1 And value2) |
| Format (String, Object ()) | Thay thế các mục định dạng trong một chuỗi chỉ định với chuỗi đại diện của một đối tượng tương ứng trong một mảng được chỉ định. | Imports System.Collections.Generic  Module Example  Public Sub Main()  Dim rnd As New Random()  Dim numbers(3) As Integer  Dim total As Integer = 0  For ctr = 0 To 2  Dim number As Integer = rnd.Next(1001)  numbers(ctr) = number  total += number  Next  numbers(3) = total  Dim values(numbers.Length - 1) As Object  numbers.CopyTo(values, 0)  Console.WriteLine("{0} + {1} + {2} = {3}", values)  End Sub  End Module |
| GetEnumerator() | Lấy một đối tượng có thể lặp qua các ký tự riêng lẻ trong chuỗi này. | Module Example  Sub Main()  EnumerateAndDisplay("Test Case")  EnumerateAndDisplay("This is a sentence.")  EnumerateAndDisplay("Has" & vbTab & "two" & vbTab & "tabs")  EnumerateAndDisplay("Two" & vbLf & "new" & vbLf & "lines")  End Sub  Sub EnumerateAndDisplay(phrase As String)  Console.WriteLine("The characters in the string ""{0}"" are:", phrase)  Dim charCount As Integer = 0  Dim controlChars As Integer = 0  Dim alphanumeric As Integer = 0  Dim punctuation As Integer = 0  For Each ch In phrase  Console.Write("'{0}' ", If(Not Char.IsControl(ch), ch,  "0x" + Convert.ToUInt16(ch).ToString("X4")))  If Char.IsLetterOrDigit(ch) Then  alphanumeric += 1  Else If Char.IsControl(ch) Then  controlChars += 1  Else If Char.IsPunctuation(ch) Then  punctuation += 1  End If  CharCount += 1  Next  Console.WriteLine()  Console.WriteLine(" Total characters: {0,3}", CharCount)  Console.WriteLine(" Alphanumeric characters: {0,3}", alphanumeric)  Console.WriteLine(" Punctuation characters: {0,3}", punctuation)  Console.WriteLine(" Control Characters: {0,3}", controlChars)  Console.WriteLine()  End Sub  End Module |
| GetHashCode() | Trả về mã đại diện cho chuỗi. | Module GetHashCode  Sub Main()  DisplayHashCode("")  DisplayHashCode("a")  DisplayHashCode("ab")  DisplayHashCode("abc")  DisplayHashCode("abd")  DisplayHashCode("abe")  DisplayHashCode("abcdef")  DisplayHashCode("abcdeg")  DisplayHashCode("abcdeh")  DisplayHashCode("abcdei")  DisplayHashCode("Abcdeg")  DisplayHashCode("Abcdeh")  DisplayHashCode("Abcdei")  End Sub  Sub DisplayHashCode(Operand As String)  Dim HashCode As Integer = Operand.GetHashCode()  Console.WriteLine("The hash code for ""{0}"" is: 0x{1:X8}, {1}",  Operand, HashCode)  End Sub  End Module |
| GetType() | Lấy các kiểu của trường hợp hiện tại | Module Example  Public Sub Main()  Dim n1 As Integer = 12  Dim n2 As Integer = 82  Dim n3 As Long = 12  Console.WriteLine("n1 and n2 are the same type: {0}",  Object.ReferenceEquals(n1.GetType(), n2.GetType()))  Console.WriteLine("n1 and n3 are the same type: {0}",  Object.ReferenceEquals(n1.GetType(), n3.GetType()))  End Sub  End Module  ' The example displays the following output:  ' n1 and n2 are the same type: True  ' n1 and n3 are the same type: False |
| GetTypeCode() | Trả về kiểu code cho lớp chuỗi | Imports System  Class Sample  Public Shared Sub Main()  Dim str As [String] = "abc"  Dim tc As TypeCode = str.GetTypeCode()  Console.WriteLine("The type code for '{0}' is {1}, which represents {2}.", \_  str, tc.ToString("D"), tc.ToString("F"))  End Sub 'Main  End Class |
| IndexOf(Char) | Báo cáo chỉ số không dựa trên sự xuất hiện đầu tiên của ký tự Unicode được quy định trong chuỗi này. | Public Module Example  Public Sub Main()  ' Create a Unicode string with 5 Greek Alpha characters.  Dim szGreekAlpha As New String(ChrW(&H0391), 5)  ' Create a Unicode string with 3 Greek Omega characters.  Dim szGreekOmega As String = ChrW(&H03A9) + ChrW(&H03A9)+  ChrW(&H03A9)  Dim szGreekLetters As String = String.Concat(szGreekOmega, szGreekAlpha, \_  szGreekOmega.Clone())  ' Display the entire string.  Console.WriteLine(szGreekLetters)  ' The first index of Alpha.  Dim iAlpha As Integer = szGreekLetters.IndexOf(ChrW(&H0391))  ' The first index of Omega.  Dim iomega As Integer = szGreekLetters.IndexOf(ChrW(&H03A9))  Console.WriteLine("First occurrence of the Greek letter Alpha: Index {0}",  ialpha)  Console.WriteLine("First occurrence of the Greek letter Omega: Index {0}",  iomega)  End Sub  End Module |
| IndexOf(Char, Int32) | Báo cáo chỉ số không dựa trên sự xuất hiện đầu tiên của ký tự Unicode được quy định trong chuỗi này. Các tìm kiếm bắt đầu tại một vị trí ký tự quy định | Module Sample  Sub Main()  Dim br1 As String = "0----+----1----+----2----+----3----+----4----+----5----+----6----+-"  Dim br2 As String = "0123456789012345678901234567890123456789012345678901234567890123456"  Dim str As String = "Now is the time for all good men to come to the aid of their party."  Dim start As Integer  Dim at As Integer  Console.WriteLine()  Console.WriteLine("All occurrences of 't' from position 0 to {0}.", str.Length - 1)  Console.WriteLine("{1}{0}{2}{0}{3}{0}", Environment.NewLine, br1, br2, str)  Console.Write("The letter 't' occurs at position(s): ")  at = 0  start = 0  While start < str.Length AndAlso at > -1  at = str.IndexOf("t"c, start)  If at = -1 Then  Exit While  End If  Console.Write("{0} ", at)  start = at + 1  End While  Console.WriteLine()  End Sub  End Module |
| IndexOf(Char, Int32, Int32) | Báo cáo chỉ số không dựa trên sự xuất hiện đầu tiên của ký tự được quy định trong trường hợp này. Các tìm kiếm bắt đầu tại một vị trí ký tự quy định và kiểm tra một số quy định | Dim startIndex As Integer = - 1  Dim hitCount As Integer = 0  ' Search for all occurrences of the target.  While True  startIndex = searched.IndexOf( \_  target, startIndex + 1, \_  searched.Length - startIndex - 1)  ' Exit the loop if the target is not found.  If startIndex < 0 Then  Exit While  End If  Console.Write("{0}, ", startIndex)  hitCount += 1  End While |
| IndexOf(String) | Báo cáo chỉ số không dựa trên sự xuất hiện đầu tiên của chuỗi quy định trong trường hợp này. | Dim softHyphen As String = ChrW(&h00AD)  Console.WriteLine(s1.IndexOf(softHyphen + "n"))  ----> vi tri 1 |
| IndexOf(String, Int32) | Báo cáo chỉ số không dựa trên sự xuất hiện đầu tiên của chuỗi quy định trong trường hợp này. Các tìm kiếm bắt đầu tại một vị trí ký tự quy định. | Dim searchString As String = Chrw(&h00AD) + "m"  Dim s1 As String = "ani" + ChrW(&h00AD) + "mal"  Console.WriteLine(s1.IndexOf(searchString, 2))  ---->>> 4 |
| IndexOf(String, Int32, Int32) | Báo cáo chỉ số không dựa trên sự xuất hiện đầu tiên của chuỗi quy định trong trường hợp này. Các tìm kiếm bắt đầu tại một vị trí ký tự quy định và kiểm tra một số quy định của vị | Dim searchString As String = Chrw(&h00AD) + "m"  Dim s1 As String = "ani" + ChrW(&h00AD) + "mal"  Dim s2 As String = "animal"  Console.WriteLine(s1.IndexOf(searchString, 2, 4))  Console.WriteLine(s2.IndexOf(searchString, 2, 4))  -->>>4  -->>>3 |
| IndexOf (String, Int32, Int32, StringComparison) | Báo cáo chỉ số không dựa trên sự xuất hiện đầu tiên của chuỗi xác định trong hiện tại Chuỗi đối tượng. Các thông số xác định vị trí tìm kiếm bắt đầu trong chuỗi hiện tại, số lượng ký tự trong chuỗi hiện tại để tìm kiếm, và các loại tìm kiếm để sử dụng cho các chuỗi quy định. | Dim searchString As String = Chrw(&h00AD) + "m"  Dim s1 As String = "ani" + ChrW(&h00AD) + "mal"  Dim s2 As String = "animal"  Console.WriteLine(s1.IndexOf(searchString, 2, 4, StringComparison.CurrentCulture))  Console.WriteLine(s1.IndexOf(searchString, 2, 4, StringComparison.Ordinal))  Console.WriteLine(s2.IndexOf(searchString, 2, 4, StringComparison.CurrentCulture))  Console.WriteLine(s2.IndexOf(searchString, 2, 4, StringComparison.Ordinal)) |

**Cấu trúc của Class List(Of T)**

<SerializableAttribute>

Public Class List(Of T)

Implements IList(Of T), ICollection(Of T), IEnumerable(Of T), IEnumerable,

IList, ICollection, IReadOnlyList(Of T), IReadOnlyCollection(Of T)

|  |  |  |
| --- | --- | --- |
| **List(Of T)** | | |
| **Phương thức** | **Ý nghĩa** | **Ví dụ** |
| Add(T) | Thêm 1 đối tượng vào cuối danh sách, đổi tượng thuộc kiểu danh sách đã khởi tạo (T). | Dim DanhSachLop As New List(Of String)  DanhSachLop.Add("14CK2")  DanhSachLop.Add("14CK3") |
| AddRange(Ienumerable(OfT)) | Thêm một dẩy các đối tượng xác đinh vào cuối danh sách. | Dim danhsach() As String = { "14CK2", \_  "14CK3**"**}  Dim DanhSachLop As New List(Of String)(danhsach) dinosaurs.AddRange(dinosaurs) |
| AsReadOnly() | Trả ra một danh sách chỉ đọc mà không được sửa. | Dim DanhSachLop As New List(Of String)  DanhSachLop.Add("14CK2")  DanhSachLop.Add("14CK3")  Dim DanhSachChiDoc As List(Of String) = DanhSachLop.AsReadOnly() |
| BinarySearch(T) | Tìm kiếm nhị phân sử dụng so sánh mặc định, yêu cầu danh sách đã được sắp xếp và trả ra chỉ số của đối tượng trong danh sách. | Dim DanhSachLop As New List(Of String)  DanhSachLop.Add("14CK2")  DanhSachLop.Add("14CK3")  DanhSachLop.Sort() ‘sap xep danh sach.  Dim index As Integer = DanhSachLop.BinarySearch("14CK") |
| BinarySearch(T, Icomparer(OfT)) | Tìm kiếm nhị phân sử dụng so sánh đề ra, yêu cầu danh sách đã được sắp xếp và trả ra chỉ số của đối tượng trong danh sách. | Dim DanhSachLop As New List(Of String)  DanhSachLop.Add("14CK2")  DanhSachLop.Add("14CK3")  Dim comparer As New Comperer ‘La mot Lop comparer bat ki su dung interface Icomparer.  DanhSachLop.Sort(dc)  Dim index As Integer = DanhSachLop.BinarySearch("14CK", comparer) |
| BinarySearch(Int32, Int32, T, Icomparer(OfT)) | Tìm kiếm nhị phân sử dụng so sánh đề ra trên một phạm vi đã được sắp xếp trong mảng và trả ra chỉ số của đối tượng trong danh sách. | Dim DanhSachLop As New List(Of String)  DanhSachLop.Add("14CK2")  DanhSachLop.Add("14CK3")  Dim comparer As New Comperer ‘La mot Lop comparer bat ki su dung interface Icomparer.  DanhSachLop.Sort(dc)  Dim index As Integer = DanhSachLop.BinarySearch(0, 1, "14CK", comparer) |
| Clear() | Xóa tất cả phần tử có trong danh sách. | Dim DanhSachLop As New List(Of String)  DanhSachLop.Add("14CK2")  DanhSachLop.Add("14CK3")  DanhSachLop.Clear() |
| Contains(T) | Xác định một đối tượng có chứa trong danh sách không. | Dim DanhSachLop As New List(Of String)()  DanhSachLop.Add("14CK2") DanhSachLop.Add(("14CK3")  If DanhSachLop.Contains("14CK3") Then  Console.WriteLine("Co lop 14CK3")  Else  Console.WriteLine("Khong co lop 14CK3")  End If |
| ConvertAll(OfTOutput)(Converter(OfT,Toutput)) | Chuyển các phần tử trong dánh sách sang một kiểu khác và trả ra danh sách các phần tử với kiểu dữ liệu mới. | Dim lpf As New List(Of PointF)  lpf.Add(New PointF(27.8, 32.62))  lpf.Add(New PointF(99.3, 147.273))  Dim lp As List(Of Point) = lpf.ConvertAll( \_  New Converter(Of PointF, Point)(AddressOf PointFToPoint)) |
| CopyTo(T()) | Sao chép toàn bộ phần tử có trong danh sách đến một mảng khác tương thích. | Dim DanhSachLop As New List(Of String)()  DanhSachLop.Add("14CK2") DanhSachLop.Add(("14CK3")  Dim DanhSachSaoChep(14) As String  DanhSachLop.CopyTo(DanhSachSaoChep) |
| CopyTo(T(), Int32) | Sao chép một mảng vào danh sách với vị trí bắt đầu trong danh sách. | Dim DanhSachLop As New List(Of String)()  DanhSachLop.Add("14CK2") DanhSachLop.Add(("14CK3")  Dim DanhSachSaoChep(14) As String  DanhSachLop.CopyTo(DanhSachSaoChep,1) |
| CopyTo(Int32, T(), Int32, Int32) | Sao chép các phần tử trong phạm vi đặt ra của mảng vào vị trí danh sách. | Dim DanhSachLop As New List(Of String)()  DanhSachLop.Add("14CK2") DanhSachLop.Add(("14CK3")  Dim DanhSachSaoChep(14) As String  DanhSachLop.CopyTo(0,DanhSachSaoChep,0,2) |
| Equals(Object) | So sánh bằng của đối tượng hiện tại với một đối tượng khác. | Dim hocsinhA As New String(“An”)  Dim hocsinhB As New String(“Binh”)  If hocsinhA.Equals(hocsinhB) Then  // to do  End If |
| Exists(Predicate(OfT)) | Xác định xem danh sách có chứa đối tượng được quy định bỡi một điều kiện đặt ra. | Dim dinosaurs As New List(Of String)  dinosaurs.Add("Compsognathus")  dinosaurs.Add("Amargasaurus")  dinosaurs.Add("Oviraptor")  Console.WriteLine(vbLf & \_  "Exists(AddressOf EndsWithSaurus): {0}", \_  dinosaurs.Exists(AddressOf EndsWithSaurus))  Private Shared Function EndsWithSaurus(ByVal s As String) \_  As Boolean  Return s.ToLower().EndsWith("saurus")  End Function |
| Finalize() | Giúp List giải phóng tài nguyên (trao trã tài nguyên). |  |
| Find(Predicate(OfT)) | Tìm kiếm một đối tượng trong danh sách kèm với kiểu tìm kiếm và trả ra kết quả đầu tiên là kiểu T. | Private IDToFind As String = "bk109"  Public Books As New List(Of Book)  Sub Main()  FillList()  Dim result As Book = Books.Find(AddressOf FindID)  If result IsNot Nothing Then  Console.WriteLine("Found: ")  Else  Console.WriteLine("Not found: ")  End If  Private Function FindID(ByVal bk As Book) As Boolean  If bk.ID = IDToFind Then  Return True  Else  Return False  End If  End Function |
| FindAll(Predicate) | Tìm kiếm một đối tượng trong danh sách kèm với kiểu tìm kiếm và trả ra tất cả kết quả tìm thấy kiểu T. | Tương tự Find(Predicate(OfT)) nhưng sẽ trả về một danh sách kết quả tìm được. |
| FindIndex(Int32, Predicate(OfT)) | Tìm kiếm một đối tượng trong danh sách với phạm vi từ một chỉ số đến chỉ số cuối cùng của danh sách kèm điều kiện và trả ra kết quả đầu tiên là chỉ số. | Dim employees As New List(Of Employee)()  employees.AddRange( { New Employee() With { .Name = "Frank", .Id = 2 },  New Employee() With { .Name = "Jill", .Id = 3 },  New Employee() With { .Name = "Dave", .Id = 5 },  New Employee() With { .Name = "Jack", .Id = 8 },  New Employee() With { .Name = "Judith", .Id = 12 },  New Employee() With { .Name = "Robert", .Id = 14 },  New Employee() With { .Name = "Adam", .Id = 1 } } )  employees.Sort()  Dim es As New EmployeeSearch("J")  Dim index As Integer = employees.FindIndex(4, AddressOf es.StartsWith)  Console.WriteLine("Starting index of'J': {0}",  If(index >= 0, index.ToString(), "Not found")) |
| FindIndex(Predicate(OfT)) | Tìm kiếm một đối tượng trong danh sách kèm điều kiện và trả ra kết quả đầu tiên là chỉ số. | Tương tự như FindIndex(Int32, Predicate(OfT)) nhưng đối với hàm này sẽ tìm trên toàn bộ danh sách. |
| FindLast(Predicate(OfT)) | Tìm kiếm một đối tượng trong danh sách kèm điều kiện và trả ra kết quả cuối cùng kiểu T. | Tương tự như Find(Predicate(OfT)) nhưng sẽ trả ra kết quả tìm kiếm cuối cùng. |
| FindLastIndex(Int32, Predicate(OfT)) | Tìm kiếm một đối tượng trong danh sách với phạm vi từ đối tượng đầu tiên tới đối tượng có chỉ số truyền vào kèm điều kiện và trả ra kết quả cuối cùng là chỉ số. | Tương tự như FindIndex(Int32, Predicate(OfT)) nhưng kết quả tìm kiếm là kết quả cuối cùng |
| FindLastIndex(Predicate(OfT)) | Tìm kiếm một đối tượng trong danh sách kèm điều kiện và trả ra kết quả cuối cùng là chỉ số. | Tương tự như FindIndex(Predicate(OfT)) nhưng hàm này sẽ trả ra kết quả tìm kiếm cuối cùng. |
| ForEach(Action(OfT)) | Thực hiện một hành động lên mỗi đối tượng có trong danh sách. | Dim DanhSachLop As New List(Of String)  DanhSachLop.Add("14CK3")  DanhSachLop.Add("14CK4")  DanhSachLop.Add("14CK5")  For Each lop As String In DanhSachLop  Console.WriteLine(lop)  Next |
| [GetRange(Int32, Int32)](https://msdn.microsoft.com/en-us/library/21k0e39c(v=vs.110).aspx) |  | List T:  'Brachiosaurus  'Amargasaurus  'Amargasaurus  'Tyrannosaurus  'Deinonychus  'Velociraptor  'Mamenchisaurus  T.GetRange(2, 3).ToArray  'Amargasaurus  'Tyrannosaurus  'Deinonychus |
| [GetType()](https://msdn.microsoft.com/en-us/library/system.object.gettype(v=vs.110).aspx) | Trả ra Type của List |  |
| [IndexOf(T)](https://msdn.microsoft.com/en-us/library/e4w08k17(v=vs.110).aspx) | Trả ra vị trí đầu tiên của phần tử T trong List | Dim dinosaurs As New List(Of String)  dinosaurs.Add("Tyrannosaurus")  dinosaurs.Add("Amargasaurus")  dinosaurs.Add("Mamenchisaurus")  dinosaurs.Add("Brachiosaurus")  dinosaurs.Add("Deinonychus")  dinosaurs.Add("Tyrannosaurus")  dinosaurs.Add("Compsognathus")  Console.WriteLine(vbLf & \_  "IndexOf(""Tyrannosaurus""): {0}", \_  dinosaurs.IndexOf("Tyrannosaurus"))  'IndexOf("Tyrannosaurus"): 0  Console.WriteLine(vbLf & \_  "IndexOf(""Tyrannosaurus"", 3): {0}", \_  dinosaurs.IndexOf("Tyrannosaurus", 3))  'IndexOf("Tyrannosaurus", 3): 5  Console.WriteLine(vbLf & \_  "IndexOf(""Tyrannosaurus"", 2, 2): {0}", \_  dinosaurs.IndexOf("Tyrannosaurus", 2, 2))  'IndexOf("Tyrannosaurus", 2, 2): -1 |
| [IndexOf(T, Int32)](https://msdn.microsoft.com/en-us/library/s8t42k5w(v=vs.110).aspx) | Trả ra vị trí đầu tiên của phần tử T trong List bắt đầu tại index |
| [IndexOf(T, Int32, Int32)](https://msdn.microsoft.com/en-us/library/dy8zse0c(v=vs.110).aspx) | Trả ra vị trí đầu tiên của phần tử T trong List bắt đầu tại index đếm sang count phần tử |
| [Insert(Int32, T)](https://msdn.microsoft.com/en-us/library/sey5k5z4(v=vs.110).aspx) | Chén 1 phần tử vào List ở vị trí chỉ định. | List T  'Brachiosaurus  'Amargasaurus  'Mamenchisaurus  T.Insert(1,”Mamenchisaurus “)  Result:  'Brachiosaurus  'Mamenchisaurus  'Amargasaurus  'Mamenchisaurus |
| [InsertRange(Int32, IEnumerable(Of T))](https://msdn.microsoft.com/en-us/library/884ee1fz(v=vs.110).aspx) | Chèn các phần tử của một List khác vào List (Of T) ở vị trí chỉ định. | List T  'Brachiosaurus  'Amargasaurus  'Mamenchisaurus  'List input  'Brachiosaurus  'Amargasaurus  'Mamenchisaurus  'InsertRange(1, input)  'Brachiosaurus  'Amargasaurus  'Brachiosaurus  'Amargasaurus  'Mamenchisaurus  'Mamenchisaurus |
| [LastIndexOf(T)](https://msdn.microsoft.com/en-us/library/dsxfcba1(v=vs.110).aspx) | Trả ra vị trí cuối cùng của phần tử T trong List | Dim dinosaurs As New List(Of String)  dinosaurs.Add("Tyrannosaurus")  dinosaurs.Add("Amargasaurus")  dinosaurs.Add("Mamenchisaurus")  dinosaurs.Add("Brachiosaurus")  dinosaurs.Add("Deinonychus")  dinosaurs.Add("Tyrannosaurus")  dinosaurs.Add("Compsognathus")  Console.WriteLine(vbLf & \_  "LastIndexOf(""Tyrannosaurus""): {0}", \_  dinosaurs.LastIndexOf("Tyrannosaurus"))  'LastIndexOf("Tyrannosaurus"): 5  Console.WriteLine(vbLf & \_  "LastIndexOf(""Tyrannosaurus"", 3): {0}", \_  dinosaurs.LastIndexOf("Tyrannosaurus", 3))  'LastIndexOf("Tyrannosaurus", 3): 0  Console.WriteLine(vbLf & \_  "LastIndexOf(""Tyrannosaurus"", 4, 4): {0}", \_  dinosaurs.LastIndexOf("Tyrannosaurus", 4, 4))  'LastIndexOf("Tyrannosaurus", 4, 4): -1 |
| [LastIndexOf(T, Int32)](https://msdn.microsoft.com/en-us/library/s088xd6w(v=vs.110).aspx) | Trả ra vị trí cuối cùng của phần tử T trong List bắt đầu tại index |
| [LastIndexOf(T, Int32, Int32)](https://msdn.microsoft.com/en-us/library/haast0bw(v=vs.110).aspx) | Trả ra vị trí cuối cùng của phần tử T trong List bắt đầu tại index đếm sang count phần tử |
| [MemberwiseClone()](https://msdn.microsoft.com/en-us/library/system.object.memberwiseclone(v=vs.110).aspx) | Tạo ra 1 bản sao của List |  |
| [Remove(T)](https://msdn.microsoft.com/en-us/library/cd666k3e(v=vs.110).aspx) | Xóa phần tử đầu tiên là T trong List(Of T) | List T:  ' 1234  ' 1333  ' 1834  ' 1434  ' 1444  ' 1634  ' T.Remove(1834)  ' 1234  ' 1333  ' 1434  ' 1444  ' 1634 |
| [RemoveAll(Predicate(Of T))](https://msdn.microsoft.com/en-us/library/wdka673a(v=vs.110).aspx) | Xóa toàn bộ phần tử vi phạm quy định Predicate(Of T) | private static bool EndsWithSaurus(String s)  {  return s.ToLower().EndsWith("saurus");  }  Hàm trả ra: true nếu tận cùng là saurus ,không thì false  List T:  Compsognathus  Amargasaurus  Oviraptor  Velociraptor  Deinonychus  Dilophosaurus  Gallimimus  Triceratops  T.RemoveAll(EndsWithSaurus)  Result:  Compsognathus  Oviraptor  Velociraptor  Deinonychus  Gallimimus  Triceratops |
| [RemoveAt(Int32)](https://msdn.microsoft.com/en-us/library/5cw9x18z(v=vs.110).aspx) | Xóa ở vị trí chỉ định trong List | List T:  'ID: 1234 Name: crank arm  'ID: 1334 Name: chain ring  'ID: 1834 Name: brake lever  'ID: 1434 Name: regular seat  'ID: 1444 Name: banana seat  'ID: 1634 Name: shift lever  T.RemoveAt(3)    'ID: 1234 Name: crank arm  'ID: 1334 Name: chain ring  'ID: 1834 Name: brake lever  'ID: 1444 Name: banana seat  'ID: 1634 Name: shift lever |
| [RemoveRange(Int32, Int32)](https://msdn.microsoft.com/en-us/library/y33yd2b5(v=vs.110).aspx) | Xóa dãy phần tử trong List bắt đầu tại vị trí index đếm sang count phần tử | 'List T  'Brachiosaurus  'Amargasaurus  'Mamenchisaurus  T.RemoveRange(1, 1)  'Brachiosaurus  'Amargasaurus |
| [Reverse()](https://msdn.microsoft.com/en-us/library/b0axc2h2(v=vs.110).aspx) | Đảo ngược thứ tự các phần tử trong toàn bộ List | dinosaurs.Add("Pachycephalosaurus")  dinosaurs.Add("Parasauralophus")  dinosaurs.Add("Mamenchisaurus")  dinosaurs.Add("Amargasaurus")  dinosaurs.Add("Coelophysis")  dinosaurs.Add("Oviraptor")  Result dinosaurs. [Reverse()](https://msdn.microsoft.com/en-us/library/b0axc2h2(v=vs.110).aspx)  'Oviraptor  'Coelophysis  'Amargasaurus  'Mamenchisaurus  'Parasauralophus  'Pachycephalosaurus  Result dinosaurs.Reverse(1, 4)  'Oviraptor  'Parasauralophus  'Mamenchisaurus  'Amargasaurus  'Coelophysis  'Pachycephalosaurus |
| [Reverse(Int32, Int32)](https://msdn.microsoft.com/en-us/library/hf2ay11y(v=vs.110).aspx) | Đảo ngược thứ tự các phần tử tại ví trí index đếm sang count phần tử |
| [Sort()](https://msdn.microsoft.com/en-us/library/b0zbh7b6(v=vs.110).aspx) | Sắp xếp List theo quy định mặc định | List T:  Samuel Dakota Koani Saya Vanya Yiska Yuma Jody Nikita  Result:  T.Sort(): Dakota Jody Koani Nikita Samuel Saya Vanya Yiska Yuma  Sort theo Alpha-Beta |
| [Sort(Comparison(Of T))](https://msdn.microsoft.com/en-us/library/w56d4y5z(v=vs.110).aspx) | Sắp xếp các phần tử trong danh sách bằng cách sử dụng quy định [System.Comparison(Of T)](https://msdn.microsoft.com/en-us/library/tfakywbh(v=vs.110).aspx). | Ta cần viết Comparison là cách thức quy định Sort.  Chẳng hạn:  Ta có List T  Before sort:  ' ID: 1434 Name: regular seat  ' ID: 1234 Name: crank arm  ' ID: 1634 Name: shift lever  ' ID: 1334 Name:  ' ID: 1444 Name: banana seat  ' ID: 1534 Name: cassette  Hàm Conparison quy định Sort bằng Name theo Alpha-Beta  T.Sort(Comparion(Of T))  After sort by name:  ' ID: 1334 Name:  ' ID: 1444 Name: banana seat  ' ID: 1534 Name: cassette  ' ID: 1234 Name: crank arm  ' ID: 1434 Name: regular seat  ' ID: 1634 Name: shift lever |
| [Sort(IComparer(Of T))](https://msdn.microsoft.com/en-us/library/234b841s(v=vs.110).aspx) | Sắp xếp các phần tử trong danh sách bằng cách sử dụng quy định IComparer | Public Class DinoComparer  Implements IComparer(Of String)  Public Function Compare(ByVal x As String, \_  ByVal y As String) As Integer \_  Implements IComparer(Of String).Compare  If x Is Nothing Then  If y Is Nothing Then  ' If x is Nothing and y is Nothing, they're  ' equal.  Return 0  Else  ' If x is Nothing and y is not Nothing, y  ' is greater.  Return -1  End If  Else  ' If x is not Nothing...  '  If y Is Nothing Then  ' ...and y is Nothing, x is greater.  Return 1  Else  ' ...and y is not Nothing, compare the  ' lengths of the two strings.  '  Dim retval As Integer = \_  x.Length.CompareTo(y.Length)  If retval <> 0 Then  ' If the strings are not of equal length,  ' the longer string is greater.  '  Return retval  Else  ' If the strings are of equal length,  ' sort them with ordinary string comparison.  '  Return x.CompareTo(y)  End If  End If  End If  End Function  End Class  Sau đó dùng IComparer này để Sort List theo cú pháp:  Chẳng hạng List T  Dim dc As New DinoComparer   * T.Sort(dc) |
| [Sort(Int32, Int32, IComparer(Of T))](https://msdn.microsoft.com/en-us/library/8ce6t5ad(v=vs.110).aspx) | Sắp xếp các phần tử bắt đầu từ index đếm sang count phần tử trong 1 chuỗi phần tử bằng cách sử dụng quy định IComparer | Tương tự như trên nhưng chỉ Sort 1 phần trong List  Chẳng hạn: T.Sort(2,3,dc) Sort từ phần tử thứ 2 đến phần tử 2+3=5 bằng IComparer |
| [ToArray()](https://msdn.microsoft.com/en-us/library/x303t819(v=vs.110).aspx) | Copy các phần tử của List(Of T) vào Array mới | 'Brachiosaurus  'Amargasaurus  'Amargasaurus  'Tyrannosaurus  'Deinonychus  'Velociraptor  'Mamenchisaurus  dinosaurs.GetRange(2, 3).ToArray  '  'Amargasaurus  'Tyrannosaurus  'Deinonychus |
| [ToString()](https://msdn.microsoft.com/en-us/library/system.object.tostring(v=vs.110).aspx) | Trả về một chuỗi đại diện cho đối tượng hiện tại. (Thừa kế từ Object.) | Dim values() As Integer = { 1, 2, 4, 8, 16, 32, 64, 128 }  Console.WriteLine(values.ToString())  Dim list As New List(Of Integer)(values)  Console.WriteLine(list.ToString())  ' The example displays the following output:  ' System.Int32[]  'System.Collections.Generic.List`1[System.Int32] |
| [TrimExcess()](https://msdn.microsoft.com/en-us/library/ms132207(v=vs.110).aspx) | Xác định số lượng phần tử thực tế có trong danh sách(Of T). Nếu số lượng đó ít hơn giá trị ngưỡng. Để giúp làm giảm thiểu tổn thất bộ nhớ. | List T:  ID: 1234 Name: crank arm  ID: 1334 Name: chain ring  ID: 1434 Name: seat  ID: 1534 Name: cassette  ID: 1634 Name: shift lever  Capacity: 8  Count: 5  T.TrimExcess()  Result:  Capacity: 5  Count: 5 |
| [TrueForAll(Predicate(Of T))](https://msdn.microsoft.com/en-us/library/kdxe4x4w(v=vs.110).aspx) | Xác định xem mọi phần tử trong danh sách (Of T) phù hợp với các điều kiện quy định.  - True nếu tất cả phần tử trong List phù hợp với các điều kiện  -False nếu có bất cứ phần tử nào trong List không thỏa. | Private Shared Function EndsWithSaurus(ByVal s As String) \_  As Boolean  Return s.ToLower().EndsWith("saurus")  End Function  End Class  Nếu tận cùng là saurus thì true ngược lại là false  List T:  'Compsognathus  'Amargasaurus  'Oviraptor  'Velociraptor  'Deinonychus  'Dilophosaurus  'Gallimimus  'Triceratops  T.TrueForAll(AddressOf EndsWithSaurus): False |
| [GetHashCode()](https://msdn.microsoft.com/en-us/library/system.object.gethashcode(v=vs.110).aspx) | Trả ra mã định dạng của List |  |