

LẬP TRÌNH GIAO DIỆN

Nguyễn Thị Mai Trang

Nguyễn Thị Mai Trang

1

1

Chương 6

Mảng (Array) Chuỗi (String)

2

Mục tiêu

- Mô tả, khởi tạo và sử dụng thành thạo kiểu dữ liệu mảng trong lập trình
- Mô tả, khởi tạo và sử dụng thành thạo các kiểu dữ liệu tập hợp trong .Net
- Mô tả, khởi tạo và sử dụng thành thạo kiểu dữ liệu chuỗi trong lập trình
- Sử dụng thành thạo lớp StringBuilder khi thao tác động với chuỗi

Nguyễn Thị Mai Trang

3

3

6.1 Mảng

1. Giới thiệu về mảng
2. Khai báo
3. Làm việc với mảng
4. Truyền mảng cho phương thức
5. Mảng nhiều chiều
6. Các lớp tập hợp trong VS.Net

Nguyễn Thị Mai Trang

4

4

6.1.1 Giới thiệu về mảng

- Mảng là một tập hợp có thứ tự của những đối tượng có cùng một kiểu dữ liệu.
- Các phần tử trong mảng được truy xuất theo tên và vị trí, chỉ số bắt đầu bằng zero.

– Ví dụ:
mảng số nguyên
có tên là c,
có 7 phần tử:

Mảng c có 7 phần tử

c[0]	5
c[1]	8
c[2]	12
c[3]	-7
c[4]	6
c[5]	15
c[6]	20

Chỉ số mỗi phần tử trong mảng

Nguyễn Thị Mai Trang

5

5

Giới thiệu về mảng (tt)

- Mảng là kiểu dữ liệu tham chiếu, được xem là một đối tượng bao gồm các phương thức, thuộc tính.
- Có nhiều loại mảng: mảng một chiều, mảng nhiều chiều, ...
- Mảng là đối tượng của lớp System.Array.
- Các thuộc tính cơ bản của class Array:
 - Length: thuộc tính chiều dài của mảng
 - Rank: thuộc tính số chiều của mảng

Nguyễn Thị Mai Trang

6

6

Giới thiệu về mảng (tt)

- Các phương thức cơ bản của class Array:
 - BinarySearch(): tìm kiếm trên mảng một chiều đã sắp thứ tự.
 - Clear(): xóa tất cả các phần tử của mảng.
 - Copy(): sao chép một vùng của mảng vào mảng khác.
 - Reverse(): đảo thứ tự của các thành phần trong mảng một chiều.
 - Sort(): sắp xếp giá trị trong mảng một chiều (đối với các kiểu dữ liệu định nghĩa sẵn).
 - GetLowerBound(): trả về cận dưới của chiều xác định trong mảng.
 - GetUpperBound(): trả về cận trên của chiều xác định trong mảng.
 - SetValue(): thiết lập giá trị cho phần tử mảng.

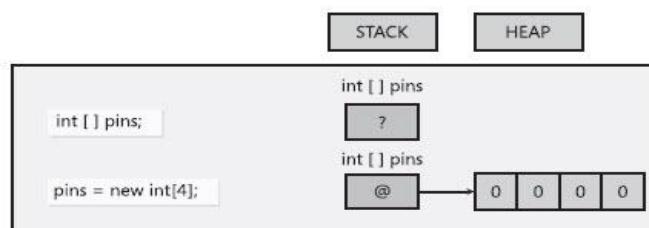
Nguyễn Thị Mai Trang

7

7

6.1.2 Khai báo mảng

- **Cú pháp:** `type[] array_name;`
 - Ví dụ: `int [] pins;`
- Khai báo và cấp phát vùng nhớ cho mảng với từ khóa **new**:
 - Ví dụ: `int [] pins= new int [4];`



Nguyễn Thị Mai Trang

8

8

Khai báo mảng (tt)

- Khai báo và khởi tạo các phần tử mảng:

– Ví dụ:

```

• string [ ] arrColors = { "Red", "Green", "Blue" };
• int [ ] pins = new int [4] { 9, 3, 7, 2 };
• Random r = new Random ();
  int [ ] pins = new int [4] { r.Next() % 10,
                              r.Next() % 10,
                              r.Next() % 10,
                              r.Next() % 10 };

```

6.1.3 Làm việc với mảng

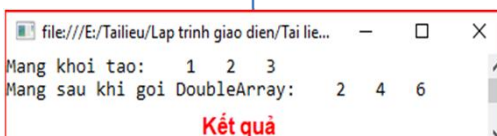
- **Xác định số phần tử mảng:** sử dụng thuộc tính Length
- **Sắp xếp mảng:** nếu các thành phần của mảng là kiểu định nghĩa trước (predefined types), ta có thể sắp xếp tăng dần bằng cách gọi phương thức static **Array.Sort()**
- **Duyệt mảng:**
 - Duyệt mảng dựa vào chỉ số như C++
 - Duyệt mảng dùng lệnh foreach

6.1.4 Truyền mảng cho phương thức

- Mảng luôn được truyền bằng tham chiếu.

```
static void Main(string[] args) {
    int[] arrInt = { 1, 2, 3 };
    Console.WriteLine("Mang khoi tao: ");
    foreach (int n in arrInt)
        Console.WriteLine("{0,4}", n);
    DoubleArray(arrInt);
    Console.WriteLine("\nMang sau khi gọi DoubleArray: ");
    foreach (int n in arrInt)
        Console.WriteLine("{0,4}", n);
    Console.ReadLine();
}

static void DoubleArray (int [] arr) {
    for (int i = 0; i < arr.Length; i++)
        arr[i] = arr[i] * 2;
}
```



```
file:///E:/Tai lieu/Lap trinh giao dien/Tai lie...
Mang khoi tao:    1    2    3
Mang sau khi gọi DoubleArray:    2    4    6
Kết quả
```

Nguyễn Thị Mai Trang

11

11

Truyền mảng cho phương thức (tt)

- Truyền mảng bình thường:
 - Sự thay đổi giá trị các phần tử mảng trong phương thức sẽ **ảnh hưởng** đến đối tượng ngoài phương thức
 - Sự thay đổi tham chiếu của biến mảng trong phương thức sẽ **không ảnh hưởng** đến đối tượng ngoài phương thức
- Truyền mảng với ref: sự thay đổi tham chiếu của biến mảng trong phương thức sẽ ảnh hưởng đến đối tượng ngoài phương thức.

Nguyễn Thị Mai Trang

12

12

6.1.5 Mảng nhiều chiều

- Mảng cần phải có hai hoặc nhiều chỉ số mới xác định được một phần tử của mảng được gọi là mảng nhiều chiều, phổ biến nhất là mảng hai chiều.
- Mảng hai chiều là mảng cần hai chỉ số để xác định được một phần tử.
- Mảng hai chiều được chia thành hai loại:
 - Mảng hình chữ nhật
 - Zagged array

Nguyễn Thị Mai Trang

13

13

Mảng nhiều chiều (tt)

- **Mảng hình chữ nhật:**
 - Có dạng bảng, trong đó các hàng có cùng kích thước (có cùng số cột).
 - Mỗi phần tử trong mảng được xác định qua hai chỉ số: hàng và cột theo quy ước: chỉ số thứ nhất là hàng, chỉ số thứ hai là cột, đều bắt đầu = 0

	cột 0	cột 1	cột 2	cột 3
hàng 0	a[0, 0]	a[0, 1]	a[0, 2]	a[0, 3]
hàng 1	a[1, 0]	a[1, 1]	a[1, 2]	a[1, 3]
hàng 2	a[2, 0]	a[2, 1]	a[2, 2]	a[2, 3]

Tên mảng

Chỉ số hàng

Chỉ số cột

Nguyễn Thị Mai Trang

14

14

Mảng nhiều chiều (tt)

- Khai báo mảng hai chiều
 - Cú pháp: `type[,] array-name;`
 - Ví dụ:


```
int [,] arrInt = new int [2,3];
int [,] arrInt = {{1, 2},{3,4}};
int [,] arrInt = new int [,] {{1,2},{3,4},{5,6},{7,8}};
string[,] arrString = {"Lennon","John",
                        {"McCartney","Paul"},
                        {"Harrison","George"},
                        {"Starkey","Richard"}};
```

Nguyễn Thị Mai Trang

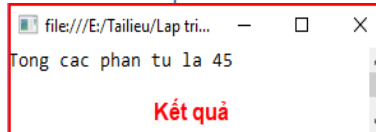
15

15

Mảng nhiều chiều (tt)

- Duyệt mảng hai chiều:
 - Sử dụng phương thức `GetLength (n)` để truy xuất số phần tử của mỗi chiều.
 - Sử dụng hai vòng lặp `for` để duyệt qua các hàng và cột

```
static void Main(string[] args) {
    int[,] arrInt = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
    int sum = 0;
    for (int i = 0; i < arrInt.GetLength(0); i++)
        for (int j = 0; j < arrInt.GetLength(1); j++)
            sum += arrInt[i, j];
    Console.WriteLine("Tong cac phan tu la {0}", sum);
    Console.ReadLine();
}
```



Nguyễn Thị Mai Trang

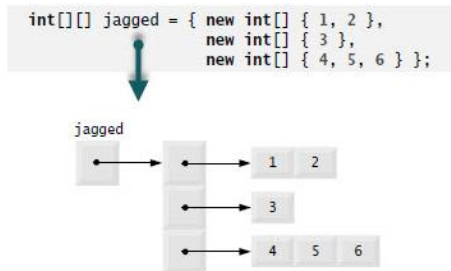
16

16

Mảng nhiều chiều (tt)

• Zagged array:

- Là mảng hai chiều, trong đó mỗi hàng là một mảng
- Số phần tử trong mỗi hàng không bằng nhau.
- Các hàng phải được khai báo tường minh.



Nguyễn Thị Mai Trang

17

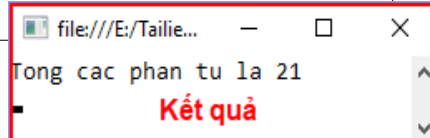
17

Mảng nhiều chiều (tt)

• Duyệt mảng Zagged array:

```
static void Main(string[] args)
{
    int[][] zaggedInt = {new int [] {1, 2},
                        new int [] {3},
                        new int [] { 4, 5, 6}};

    int sum = 0;
    foreach (var row in zaggedInt)
        foreach (var number in row)
            sum += number;
    Console.WriteLine("Tổng các phần tử là {0}", sum);
    Console.ReadLine();
}
```



Nguyễn Thị Mai Trang

18

18

6.1.6 Các lớp tập hợp thông dụng

- Mảng là kiểu dữ liệu cho phép chúng ta lưu trữ một tập hợp các phần tử một cách khá hiệu quả.
- Tuy nhiên, mảng cũng có một số hạn chế:
 - Kích thước của mảng không thể thay đổi khi chương trình thực thi
 - Các phần tử lưu trong mảng phải có cùng kiểu dữ liệu
 - Khi thêm hoặc xóa các phần tử trong mảng, ta phải duyệt và dời các phần tử liên quan
 - ...

Nguyễn Thị Mai Trang

19

19

Các lớp tập hợp thông dụng (tt)

- .NET framework cung cấp một thư viện các lớp với các phương thức cho phép thao tác với tập hợp một cách dễ dàng như:
 - Lớp ArrayList
 - Lớp Hashtable
 - Lớp SortedList
 - Lớp Queue
 - Lớp Stack
 - ...
- Các lớp trên nằm trong namespace **System.Collections**

Nguyễn Thị Mai Trang

20

20

ArrayList

- Là lớp lưu trữ tập hợp các đối tượng theo kiểu mảng, các phần tử được truy xuất thông qua chỉ số.
- Kích thước của ArrayList có thể thay đổi lúc chương trình thực thi.
- ArrayList có thể lưu các phần tử thuộc các kiểu dữ liệu khác nhau.
- ArrayList cho phép lưu giá trị null cũng như giá trị trùng lặp.
- Một số thuộc tính của lớp ArrayList:
 - Count: số phần tử trong danh sách
 - Capacity: số phần tử mà ArrayList có thể chứa

Nguyễn Thị Mai Trang

21

21

ArrayList (tt)

- Một số phương thức của lớp ArrayList:
 - Add: thêm một phần tử vào cuối danh sách.
 - Insert: chèn một phần tử vào danh sách tại vị trí được chỉ định.
 - Remove: xóa phần tử khỏi danh sách.
 - RemoveAt: xóa phần tử khỏi danh sách theo vị trí.
 - Contains: kiểm tra một phần tử có thuộc danh sách hay không (true: có, false: không).
 - IndexOf: trả về chỉ số của một phần tử trong danh sách.
 - Sort: sắp xếp danh sách đối với những kiểu dữ liệu định nghĩa trước.

Nguyễn Thị Mai Trang

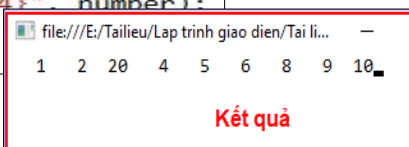
22

22

ArrayList (tt)

- Ví dụ sử dụng ArrayList:

```
static void Main(string[] args)
{
    ArrayList arrInt = new ArrayList();
    for (int i = 1; i <= 10; i++)
        arrInt.Add(i);
    arrInt.Remove(3);
    arrInt.RemoveAt(5);
    arrInt.Insert(2, 20);
    foreach (int number in arrInt)
        Console.WriteLine("{0,4}", number);
    Console.ReadLine();
}
```



Kết quả

Nguyễn Thị Mai Trang

23

23

SortedList - Hashtable

- **SortedList:**

- Là lớp danh sách kiểu từ điển, trong đó mỗi phần tử chứa trong nó được xác định thông qua hai trường Key và Value.
- Các phần tử chứa trong SortedList sẽ tự động được xếp thứ tự dựa vào trường Key

- **Hashtable:**

- Tương tự SortedList, nhưng các phần tử không được tự động sắp xếp do đó thao tác trên Hashtable nhanh hơn so với SortedList.

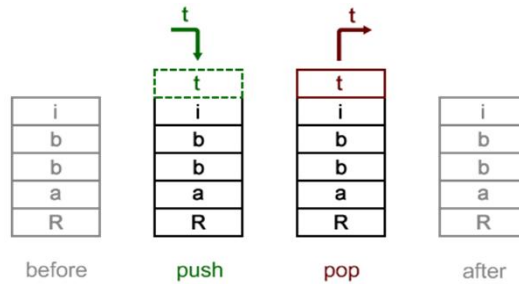
Nguyễn Thị Mai Trang

24

24

Stack

- Là loại danh sách mà các phần tử được tổ chức như một ngăn xếp theo thứ tự LIFO (**L**ast **I**n **F**irst **O**ut)
- Thao tác thêm phần tử vào Stack và lấy phần tử ra khỏi Stack đều được thực hiện tại một đầu của danh sách.



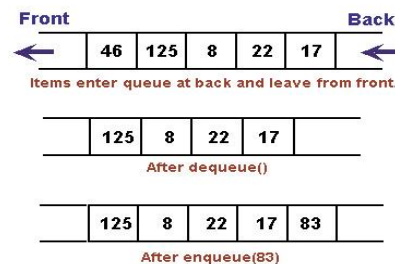
Nguyễn Thị Mai Trang

25

25

Queue

- Là loại danh sách mà các phần tử được tổ chức như một hàng theo thứ tự FIFO (**F**irst **I**n **F**irst **O**ut)
- Các phần tử được thêm vào cuối danh sách và được lấy ra từ đầu danh sách



Nguyễn Thị Mai Trang

26

26

6.2 Chuỗi (String)

1. Giới thiệu về chuỗi
2. String constructor
3. Các thuộc tính của lớp String
4. Các phương thức của lớp String
5. Các thao tác với chuỗi
6. Lớp StringBuilder

27

6.2.1 Giới thiệu về chuỗi

- Chuỗi là một dãy các ký tự unicode liên tiếp nhau trong bộ nhớ và không thể thay đổi.
- → Các phương thức áp dụng lên chuỗi không làm thay đổi nội dung bản thân chuỗi gốc mà chỉ có thể trả về một chuỗi mới.
- Trong .NET, chuỗi là kiểu dữ liệu tham chiếu, mỗi chuỗi là một đối tượng của lớp string.

28

Giới thiệu về chuỗi (tt)

- Lớp System.String có nhiều phương thức static giúp thao tác và xử lý chuỗi một cách dễ dàng.

```
public static int Compare(String strA, int indexA, String strB, int indexB);
public static int CompareOrdinal(String strA, String strB);
public static int CompareOrdinal(String strA, int indexA, String strB, int indexB);
public static String Concat(IEnumerable<String> values);
public static String Concat(params object[] args);
public static String Copy(String str);
public static bool Equals(String a, String b);
public static bool Equals(String a, String b, StringComparison comparisonType);
public static String Format(String format, params object[] args);
public static String Format(String format, object arg0);
public static String Concat(object arg0, object arg1, object arg2);
public static String Concat(object arg0, object arg1, object arg2, object arg3);
public static String Concat(String str0, String str1, String str2);
public static String Concat<T>(IEnumerable<T> values);
```

Nguyễn Thị Mai Trang

29

29

6.2.2 String constructor

- Phương thức khởi tạo của lớp string:

```
public String(char[] value);
public String(sbyte* value);
public String(char* value);
public String(char c, int count);
public String(char[] value, int startIndex, int length);
public String(sbyte* value, int startIndex, int length);
public String(char* value, int startIndex, int length);
public String(sbyte* value, int startIndex, int length, Encoding enc);
```

Nguyễn Thị Mai Trang

30

30

6.2.3 Các phương thức của lớp String

- Phương thức static (public static):
 - Compare: so sánh 2 chuỗi, trả về giá trị nguyên cho biết chuỗi thứ nhất lớn hơn, nhỏ hơn hay bằng chuỗi thứ hai.
 - Concat: trả về một chuỗi là kết quả của việc nối hai chuỗi.
 - Copy: trả về một bản sao của một chuỗi.
 - Equals: so sánh hai chuỗi, trả về giá trị true/false cho biết hai chuỗi có/không bằng nhau.
 - Format: trả về một chuỗi đã được định dạng xuất.
 - Join: trả về một chuỗi được ghép nối từ một mảng chuỗi.

Nguyễn Thị Mai Trang

31

31

Các phương thức của lớp String (tt)

- Phương thức thành viên của lớp:
 - CompareTo: so sánh một chuỗi với chuỗi khác, trả về giá trị nguyên, cho biết chuỗi đó lớn hơn, nhỏ hơn hay bằng một chuỗi khác.
 - EndsWith: trả về giá trị true/false, xác định chuỗi kết thúc của một chuỗi.
 - StartsWith: trả về giá trị true/false, xác định chuỗi bắt đầu của một chuỗi.
 - Equals: so sánh một chuỗi với chuỗi khác, trả về giá trị true/false cho biết chuỗi đó có/không bằng với một chuỗi khác.
 - Insert: trả về một chuỗi là kết quả sau khi chèn vào chuỗi một chuỗi khác.
 - Remove: trả về một chuỗi sau khi xóa các ký tự trong chuỗi.

Nguyễn Thị Mai Trang

32

32

Các phương thức của lớp String (tt)

- Phương thức thành viên (tt):
 - LastIndexOf: trả về vị trí tìm thấy của ký tự trong chuỗi bắt đầu từ cuối chuỗi.
 - Split: trả về một mảng chuỗi sau khi đã tách chuỗi đó dựa vào các ký tự đánh dấu.
 - SubString: trả về một chuỗi con trích từ một chuỗi.
 - ToCharArray: trả về một mảng ký tự sau khi sao chép một số ký tự từ chuỗi sang mảng.
 - ToLower: trả về chuỗi ký tự thường từ một chuỗi.
 - ToUpper: trả về chuỗi ký tự hoa từ một chuỗi.

Nguyễn Thị Mai Trang

33

33

Các phương thức của lớp String (tt)

- Phương thức thành viên (tt):
 - Trim: trả về chuỗi đã được cắt bỏ khoảng trắng hoặc các ký tự chỉ định ở đầu và cuối một chuỗi.
 - TrimStart: trả về chuỗi đã được cắt bỏ khoảng trắng hoặc các ký tự chỉ định ở đầu một chuỗi.
 - TrimEnd: trả về chuỗi đã được cắt bỏ khoảng trắng hoặc các ký tự chỉ định ở cuối một chuỗi.

Nguyễn Thị Mai Trang

34

34

6.2.4 Các thao tác với chuỗi

- So sánh chuỗi
 - Sử dụng toán tử so sánh bằng (==)
 - Sử dụng phương thức Equal
 - `bool bRes = s1.Equal (s2);`
 - `bRes = true` nếu `s1 = s2`
 - `bRes = false` nếu `s1 != s2`
 - Sử dụng phương thức static `String.Equals`
 - `bool bRes = String.Equals(s1, s2);`

Các thao tác với chuỗi (tt)

- So sánh chuỗi (tt)
 - Sử dụng phương thức `CompareTo`
 - `int result = s1.CompareTo (s2);`
 - `result < 0` nếu `s1 > s2`
 - `result = 0` nếu `s1 = s2`
 - `result > 0` nếu `s1 < s2`
 - Sử dụng phương thức static `String.Compare`
 - `int result = String.Compare (s1, s2);`
 - Ví dụ: Xem tài liệu học tập Lập trình giao diện

Các thao tác với chuỗi (tt)

- Trích chuỗi con: sử dụng phương thức SubString
 - SubString (int index): trả về chuỗi con bắt đầu từ vị trí index
 - SubString (int index, int length): trả về chuỗi con gồm length ký tự bắt đầu từ vị trí index
- Ví dụ: Xem tài liệu học tập Lập trình giao diện

Các thao tác với chuỗi (tt)

- Nối hai chuỗi
 - Sử dụng toán tử +
 - Sử dụng String.Concat (string s1, string s2)
 - Ví dụ: Xem tài liệu học tập Lập trình giao diện

Các thao tác với chuỗi (tt)

- Tìm ký tự, tập ký tự, chuỗi con
 - IndexOf: vị trí xuất hiện đầu tiên của một chuỗi con hoặc ký tự trong chuỗi.
 - IndexOfAny: vị trí xuất hiện đầu tiên của một hoặc một tập ký tự trong chuỗi từ một giới hạn cho trước.
 - LastIndexOf: vị trí xuất hiện cuối cùng của một chuỗi con hoặc ký tự trong chuỗi.
 - LastIndexOfAny: vị trí xuất hiện cuối cùng của một hoặc một tập ký tự trong chuỗi.
 - Ví dụ: Xem tài liệu học tập Lập trình giao diện

Các thao tác với chuỗi (tt)

- Thay thế chuỗi con trong chuỗi
 - Replace(string oldValue, string newValue)
- Loại bỏ khoảng trắng / ký tự trong chuỗi
 - Trim: trả về chuỗi đã loại bỏ ở đầu và cuối chuỗi các ký tự trong mảng tham số, nếu không có tham số thì loại bỏ các ký tự trắng.
 - TrimStart: trả về chuỗi đã loại bỏ ở đầu chuỗi các ký tự trong mảng tham số, nếu không có tham số thì loại bỏ các ký tự trắng.
 - TrimEnd: trả về chuỗi đã loại bỏ ở cuối chuỗi các ký tự trong mảng tham số, nếu không có tham số thì loại bỏ các ký tự trắng.
- Các ví dụ: Xem tài liệu học tập Lập trình giao diện

Các thao tác với chuỗi (tt)

- Loại bỏ chuỗi con trong chuỗi
 - Remove (int index): trả về chuỗi đã loại bỏ các ký tự bắt đầu từ vị trí index.
 - Remove (int index, int count): trả về chuỗi đã loại bỏ count ký tự bắt đầu từ vị trí index.
- Tách chuỗi vào mảng: sử dụng phương thức Split
- Ráp chuỗi từ mảng: sử dụng phương thức Join
- Các ví dụ: Xem tài liệu học tập Lập trình giao diện

Nguyễn Thị Mai Trang

41

41

6.2.5 Lớp StringBuilder

- Cho phép tạo, thao tác và xử lý đối với chuỗi khi chương trình thực thi
- Các phương thức khởi tạo:

```
public StringBuilder();
public StringBuilder(int capacity);
public StringBuilder(string value);
public StringBuilder(string value, int capacity);
public StringBuilder(int capacity, int maxCapacity);
public StringBuilder(string value, int startIndex, int length, int capacity);
```

Nguyễn Thị Mai Trang

42

42

Lớp StringBuilder (tt)

- Tạo đối tượng StringBuilder :
 - `StringBuilder sb1 = new StringBuilder();`
 - `StringBuilder sb2 = new StringBuilder(100);`
 - `string s = "This is a string";`
 - `StringBuilder sb3 = new StringBuilder(s);`
 - `StringBuilder sb4 = new StringBuilder(s, 4);`

Lớp StringBuilder (tt)

- Một số thuộc tính của lớp StringBuilder:
 - **Lenght**: chiều dài hiện tại của chuỗi
 - **Capacity**: dung lượng tối đa mà chuỗi có thể chứa.
- Một số phương thức của lớp StringBuilder
 - **Append**: nối một chuỗi vào cuối chuỗi
 - **AppendFormat**: định dạng chuỗi khi nối vào
 - **Insert**: chèn một chuỗi
 - **Replace**: thay thế tất cả thể hiện của một ký tự bằng các ký tự khác
 - **EnsureCapacity**: thiết lập lại kích thước hiện tại mà đối tượng StringBuilder có thể chứa.

Lớp StringBuilder (tt)

- Thay đổi kích thước chuỗi khi chương trình thực thi
 - EnsureCapacity
- Nối chuỗi:
 - Append
 - AppendFormat
- Chèn, xóa, thay thế chuỗi
 - Insert
 - Remove
 - Replace
- Các ví dụ: Xem tài liệu học tập Lập trình giao diện