

LẬP TRÌNH GIAO DIỆN

Nguyễn Thị Mai Trang

1

1

Chương 6

Mảng (Array) Chuỗi (String)

2

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

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

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

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

6

6

Giới thiệu về mảng (tt)

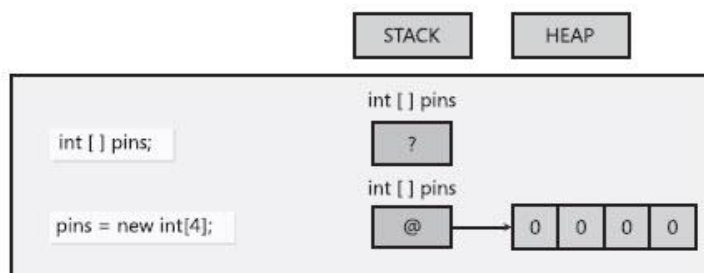
- Các phương thức cơ bản của class Array:
 - BinarySearch(): tìm kiếm trên mảng một chiều đã sắp thứ tự.
 - Clear(): xóa tất cả các phần tử của mảng.
 - Copy(): sao chép một vùng của mảng vào mảng khác.
 - Reverse(): đảo thứ tự của các thành phần trong mảng một chiều.
 - Sort(): sắp xếp giá trị trong 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.

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];`



8

8



Khai báo mạng (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 };

9

9



6.1.3 Làm việc với mảng

- **Xác định số phần tử mảng:** sử dụng thuộc tính Length

- Ví dụ: `int size = arrayInt.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()**

– Ví dụ:

- `int [] arrayInt = { 5, 7, 3, 8, 2 };`
- `Array.Sort (arrayInt);`
- Kết quả: thứ tự các phần tử trong mảng `arrayInt` sẽ là 2, 3, 5, 7, 8

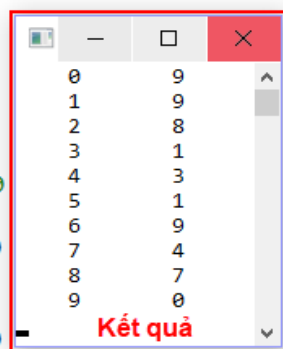
10

10

Làm việc với mảng (tt)

- Duyệt mảng dựa vào chỉ số như C++:

```
0 references
static void Main(string[] args)
{
    const int SIZE = 10;
    int [] arrInt = new int[SIZE];
    Random rand = new Random();
    //gán các giá trị ngẫu nhiên nhỏ hơn 10
    //cho các phần tử mảng
    for (int i = 0; i < arrInt.Length; i++)
        arrInt[i] = rand.Next() % 10;
    // in mảng
    for (int i = 0; i < arrInt.Length; i++)
        Console.WriteLine("{0,5}{1,8}", i, arrInt[i]);
    Console.ReadLine();
}
```



0	9
1	9
2	8
3	1
4	3
5	1
6	9
7	4
8	7
9	0

Kết quả

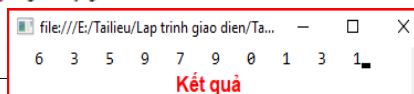
11

11

Làm việc với mảng (tt)

- Duyệt mảng dùng lệnh foreach

```
static void Main(string[] args)
{
    const int SIZE = 10;
    int [] arrInt = new int[SIZE];
    Random rand = new Random();
    //gán các giá trị ngẫu nhiên nhỏ hơn 10
    //cho các phần tử mảng
    for (int i = 0; i < arrInt.Length; i++)
        arrInt[i] = rand.Next() % 10;
    // in mảng
    foreach (int n in arrInt)
        Console.Write("{0,4}", n);
    Console.ReadLine();
}
```



6	3	5	9	7	9	0	1	3	1
---	---	---	---	---	---	---	---	---	---

Kết quả

12

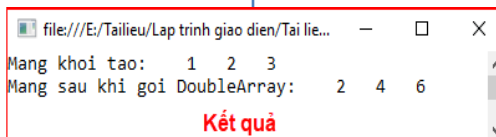
12

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 khai 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;
}
```



Kết quả

13

13

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
 - Ví dụ:

```
static void DoubleArray (int [] arr) {
    for (int i = 0; i < arr.Length; i++)
        arr[i] = arr[i] * 2;
    arr = new int [] { 10, 11, 12 };
}
```

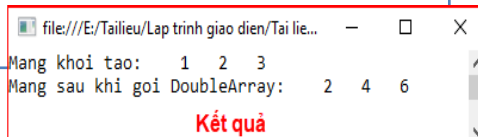
14

14

Truyền mảng cho phương thức (tt)

- Truyền mảng bình thường:

```
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();
}
```



Kết quả

15

15

Truyền mảng cho phương thức (tt)

- Truyền mảng với ref: sự thay đổi tham chiếu của biến mảng trong phương thức sẽ ảnh hưởng đến đối tượng ngoài phương thức.

Ví dụ:

```
static void DoubleArray (ref int [] arr)
{
    for (int i = 0; i < arr.Length; i++)
        arr[i] = arr[i] * 2;
    arr = new int [] { 10, 11, 12 };
}
```

16

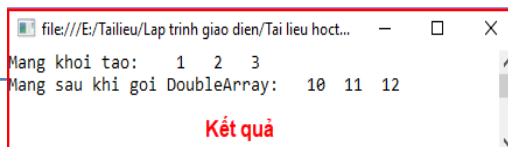
16

Truyền mảng cho phương thức (tt)

- Truyền mảng với ref:

Ví dụ (tt):

```
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(ref arrInt);
    Console.WriteLine("\nMang sau khi gọi DoubleArray: ");
    foreach (int n in arrInt)    Console.WriteLine("{0,4}", n);
    Console.ReadLine();
}
```



Kết quả

17

17

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
 - Jagged array

18

18

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

19

19

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"}};
```

20

20

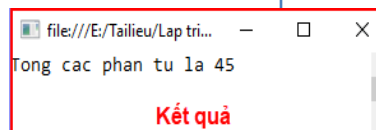
Mảng nhiều chiều (tt)

• 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("Tổng các phần tử là {0}", sum);
    Console.ReadLine();
}
```



21

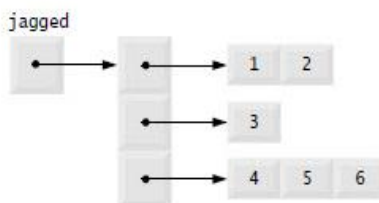
21

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.

```
int[][] jagged = { new int[] { 1, 2 },
                  new int[] { 3 },
                  new int[] { 4, 5, 6 } };
```



22

22

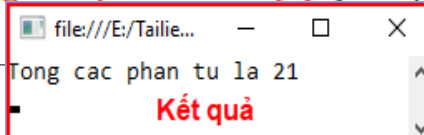
Mảng nhiều chiều (tt)

- 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("Tong cac phan tu la {0}", sum);
    Console.ReadLine();
}
```



Tong cac phan tu la 21
Kết quả

23

23

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
 - ...

24

24

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**
→ Phải khai báo: **using System.Collections;**

25

25

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

26

26

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.

27

27

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();
}
```

file:///E:/Tai lieu/Lap trinh giao dien/Tai li...

1 2 20 4 5 6 8 9 10

Kết quả

28

28

List

- Là lớp triển khai các giao diện `ICollection`, `IEnumerable`
- Quản lý danh sách các đối tượng cùng kiểu
- Các phương thức thao tác trên List tương tự như `ArrayList`:
 - `Add`, `AddRange`, `Insert`, `RemoveAt`, `Remove`,...
- Ví dụ:
 - Tạo danh sách List rỗng với các phần tử có kiểu `int`:
`var arrInt = new List<int>();`
 - Tạo danh sách List có khởi gán giá trị:
`var arrInt = new List<int>() {4, 2, 5, 7};`

29

29

List (tt)

- Tạo danh sách List chứa các đối tượng:

```
class Circle{
    double x;
    double y;
    double radius;
    public Circle(double x, double y, double r)
    {
        this.x = x;
        this.y = y;
        radius = r;
    }
}
```

```
var arrCircle = new List<Circle>() {
    new Circle(100, 50, 30),
    new Circle(150, 60, 50)
};
```

30

30

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.

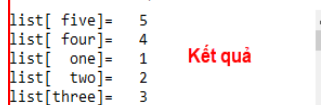
31

31

SortedList (tt)

• Ví dụ sử dụng SortedList

```
static void Main(string[] args)
{
    string[] arrNumbers = { "one", "two", "three", "four", "five" };
    SortedList list = new SortedList();
    for (int i = 0; i < arrNumbers.Length; i++)
    {
        string key = arrNumbers[i];
        list[key] = i + 1; Thêm phần tử vào list
    }
    foreach (DictionaryEntry entry in list) duyet list
        Console.WriteLine ("list[{0},5]= {1,3}", entry.Key, entry.Value);
    Console.ReadLine();
}
```



file:///E:/Tailieu/Lap tri... - □ ×

```
list[ five]= 5
list[ four]= 4
list[ one]= 1
list[ two]= 2
list[ three]= 3
```

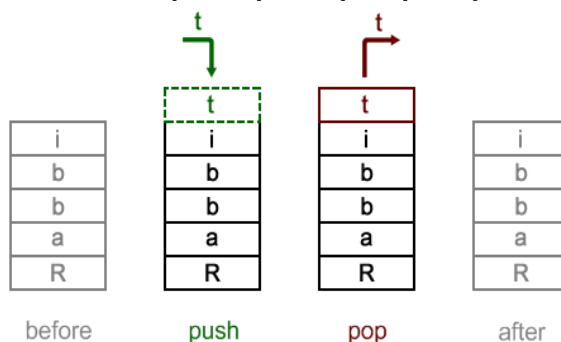
Kết quả

32

32

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.



33

33

Stack

```
static void Main(string[] args)
{
    string[] arrNumbers = { "one", "two", "three", "four", "five" };
    Stack stack = new Stack();
    foreach (string number in arrNumbers)
        stack.Push(number); //thêm phần tử vào stack
    Console.WriteLine("Cac phan tu trong stack: ");
    foreach (string item in stack) //duyet, in stack
        Console.Write("{0,6}", item);
    string lastitem = stack.Pop().ToString(); //lay phan tu tu stack
    Console.WriteLine("\nPhann tu lay ra tu stack:{0,6}", lastitem);
    Console.WriteLine("Cac phan tu con lai trong stack: ");
    foreach (string item in stack) //duyet, in cac phan tu con lai
        Console.Write("{0,6}", item);
    Console.ReadLine();
}
```

```
file:///E:/Taiieu/Lap trinh giao dien/Tai lieu hoc tap/Chuong ...
Cac phan tu trong stack:  five four three  two  one
Phann tu lay ra tu stack: five
Cac phan tu con lai trong stack:  four three  two  one
```

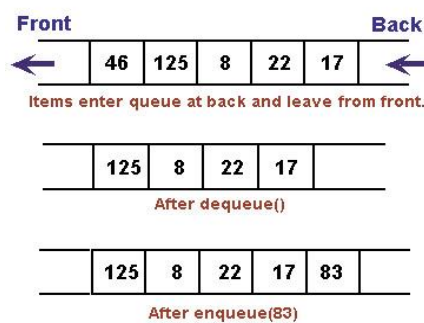
Kết quả

34

34

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



35

35

Queue

```
static void Main(string[] args)
{
    string[] arrNumbers = { "one", "two", "three", "four", "five" };
    Queue queue = new Queue();
    foreach (string number in arrNumbers)
        queue.Enqueue(number); //thêm phần tử vào queue
    Console.Write("Cac phan tu trong queue: ");
    foreach (string item in queue) //duyet, in quene
        Console.Write("{0,6}", item);
    string lastitem = queue.Dequeue().ToString(); //lay phan tu tu queue
    Console.WriteLine("\nPhan tu lay ra tu queue:{0,6}", lastitem);
    Console.Write("Cac phan tu con lai trong queue: ");
    foreach (string item in queue) //duyet, in cac phan tu con lai
        Console.Write("{0,6}", item);
    Console.ReadLine();
}
```

file:///E:/Tai lieu/Lap trinh giao dien/Tai lieu hoc tap/Chuong 6... — □ ×

Cac phan tu trong queue: one two three four five

Phan tu lay ra tu queue: one

Cac phan tu con lai trong queue: two three four five

Kết quả

36

36

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

37

37

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.

38

38

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, String str3);
public static String Concat<T>(IEnumerable<T> values);
```

39

39

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);
```

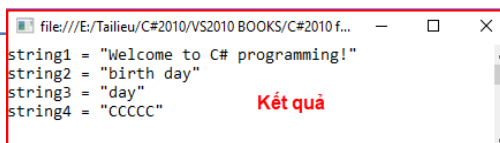
40

40

String constructor (tt)

• Ví dụ

```
public static void Main( string[] args )
{
    char[] arrChar = {'b', 'i', 'r', 't', 'h', ' ', 'd', 'a', 'y'};
    string string0 = "Welcome to C# programming!";
    string string1 = string0;
    string string2 = new string(arrChar);
    string string3 = new string(arrChar, 6, 3 );
    string string4 = new string( 'C', 5 );
    Console.WriteLine( "string1 = " + "\"" + string1 + "\"\n" +
        "string2 = " + "\"" + string2 + "\"\n" +
        "string3 = " + "\"" + string3 + "\"\n" +
        "string4 = " + "\"" + string4 + "\"\n" );
}
```



```
file:///E:/Tailieu/C#2010/VS2010 BOOKS/C#2010 f...
string1 = "Welcome to C# programming!"
string2 = "birth day"
string3 = "day"
string4 = "CCCCC"
```

Kết quả

41

41

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.

42

42

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.

43

43

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.

44

44

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.

45

45

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);`

46

46

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);`

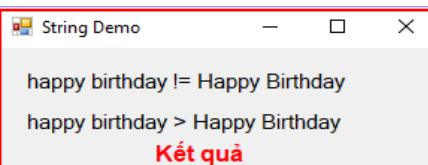
47

47

Các thao tác với chuỗi (tt)

– Ví dụ về so sánh chuỗi

```
private void CompareString()
{
    string s1 = "happy birthday";
    string s2 = "Happy Birthday";
    bool equal = s1.Equals(s2);
    int cmp = s1.CompareTo(s2);
    if (equal)
        label1.Text = String.Format("{0} = {1}", s1, s2);
    else
        label1.Text = String.Format("{0} != {1}", s1, s2);
    if (cmp == 0)
        label2.Text = String.Format("{0} = {1}", s1, s2);
    else if (cmp == -1)
        label2.Text = String.Format("{0} > {1}", s1, s2);
    else
        label1.Text = String.Format("{0} < {1}", s1, s2);
}
```



48

48

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

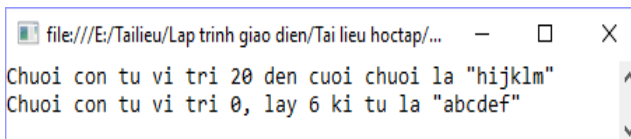
49

49

Các thao tác với chuỗi (tt)

– Ví dụ sử dụng SubString

```
public static void Main( string[] args )
{
    string letters = "abcdefghijklmabcdefghijklm";
    Console.WriteLine("Chuoi con tu vi tri 20 den cuoi chuoi la \"
        + letters.Substring(20) + \"");
    Console.WriteLine("Chuoi con tu vi tri 0, lay 6 ki tu la \"
        + letters.Substring(0, 6) + \"");
    Console.ReadLine();
}
```



50

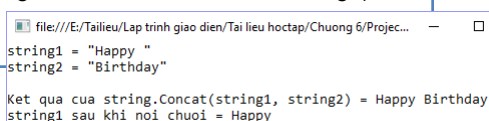
50

Các thao tác với chuỗi (tt)

- Nối hai chuỗi

- Sử dụng toán tử +
- Sử dụng `String.Concat (string s1, string s2)`

```
public static void Main(string[] args) {
    string string1 = "Happy ";
    string string2 = "Birthday";
    Console.WriteLine("string1 = \"\" + string1 + "\"\n\" +
        "string2 = \"\" + string2 + "\"");
    Console.WriteLine(
        "\nKet qua cua string.Concat(string1, string2) = " +
        string.Concat(string1, string2));
    Console.WriteLine("string1 sau khi noi chuoai = " + string1);
    Console.ReadLine();
}
```



```
file:///E:/Tailieu/Lap trinh giao dien/Tai lieu hoc tap/Chuong 6/Projec...
string1 = "Happy "
string2 = "Birthday"
Ket qua cua string.Concat(string1, string2) = Happy Birthday
string1 sau khi noi chuoai = Happy
```

51

51

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.

52

52

Các thao tác với chuỗi (tt)

– Ví dụ sử dụng IndexOfAny:

```
const string value1 = "Darth is my enemy.";
const string value2 = "Visual Basic is hard.";

int index1 = value1.IndexOfAny(new char[] { 'e', 'B' });
string s1 = value1.Substring(index1);
//s1 = "enemy."
int index2 = value2.IndexOfAny(new char[] { 'e', 'B' });
string s2 = value2.Substring(index2);
//s2 = "Basic is hard."
```

53

53

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.

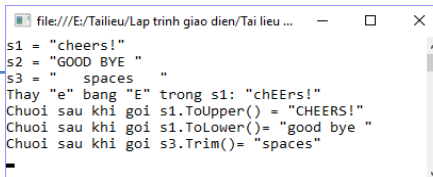
54

54

Các thao tác với chuỗi (tt)

– Ví dụ

```
public static void Main(string[] args) {
    string s1 = "cheers!";
    string s2 = "GOOD BYE ";
    string s3 = " spaces ";
    Console.WriteLine("s1 = \"\" + s1 + "\"\n"
        + "s2 = \"\" + s2 + "\"\n" + "s3 = \"\" + s3 + "\"");
    Console.WriteLine("Thay 'e' bang 'E' trong s1: \"\"
        + s1.Replace('e', 'E') + "\"");
    Console.WriteLine("Chuoi sau khi gọi s1.ToUpper() = \"\"
        + s1.ToUpper()
        + "\"\nChuoi sau khi gọi s1.ToLower() = \"\"
        + s2.ToLower() + "\"");
    Console.WriteLine("Chuoi sau khi gọi s3.Trim() = \"\"
        + s3.Trim() + "\"");
    Console.ReadLine();
}
```



```
file:///E:/Tailieu/Lap trinh giao dien/Tai lieu ...
s1 = "cheers!"
s2 = "GOOD BYE "
s3 = " spaces "
Thay "e" bang "E" trong s1: "cheErs!"
Chuoi sau khi gọi s1.ToUpper() = "CHEERS!"
Chuoi sau khi gọi s1.ToLower() = "good bye "
Chuoi sau khi gọi s3.Trim() = "spaces"
```

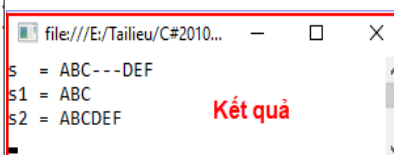
55

55

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.

```
public static void Main( string[] args
{
    string s = "ABC---DEF";
    string s1 = s.Remove(3);
    string s2 = s.Remove(3, 3);
    Console.WriteLine("s = {0}", s);
    Console.WriteLine("s1 = {0}", s1);
    Console.WriteLine("s2 = {0}", s2);
    Console.Read();
} // end Main
```



```
file:///E:/Tailieu/C#2010...
s = ABC---DEF
s1 = ABC
s2 = ABCDEF
Kết quả
```

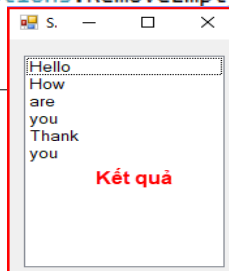
56

56

Các thao tác với chuỗi (tt)

- **Tách chuỗi vào mảng:** sử dụng phương thức Split

```
private void SplitString()
{
    string str = "Hello! How are you?.Thank you! ";
    char[] token = { ' ', '\t', '?', '.', '!' };
    string[] arrStr = str.Split(token,
                               StringSplitOptions.RemoveEmptyEntries);
    listBox1.Items.AddRange(arrStr);
}
```



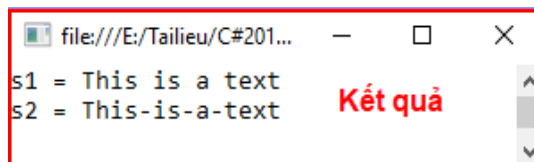
57

57

Các thao tác với chuỗi (tt)

- **Ráp chuỗi từ mảng:** sử dụng phương thức Join

```
public static void Main( string[] args )
{
    string[] arrString = { "This", "is", "a", "text" };
    string s1 = String.Join(" ", arrString);
    string s2 = String.Join("-", arrString);
    Console.WriteLine("s1 = {0}", s1);
    Console.WriteLine("s2 = {0}", s2);
    Console.Read();
} // end Main
```



58

58

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);
```

59

59

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);`

60

60

Lớp StringBuilder (tt)

- Một số thuộc tính của lớp StringBuilder:
 - Length: 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.

61

61

Lớp StringBuilder (tt)

- Thay đổi kích thước chuỗi khi chương trình thực thi
 - Sử dụng phương thức **EnsureCapacity**

```
StringBuilder buffer = new StringBuilder("Hello, how are you?");
Console.WriteLine( "buffer = " + buffer +
    "\nLength = " + buffer.Length +
    "\nCapacity = " + buffer.Capacity);
// thiết lập lại thuộc tính capacity
buffer.EnsureCapacity(75);
Console.WriteLine( "New capacity = " + buffer.Capacity );
// thiết lập lại thuộc tính Length
buffer.Length = 10;
Console.WriteLine( "New length = " + buffer.Length );
// use StringBuilder indexer
Console.Write("buffer = ");
//duyet, in ra từng ký tự trong buffer
for ( int i = 0; i < buffer.Length; i++ )
    Console.Write( buffer [i] );
```

```
file:///E:/TaiLieu/C#2...
buffer = Hello, how are you?
Length = 19
Capacity = 19
New capacity = 75
New length = 10
buffer = Hello, how_
```

Kết quả

62

62

Lớp StringBuilder (tt)

- Nối chuỗi: sử dụng phương thức **Append**

```
object objectValue = "hello";
string stringValue = "good bye";
char[] characterArray = { 'a', 'b', 'c', 'd', 'e', 'f' };
bool booleanValue = true;
char characterValue = 'Z';
int integerValue = 7;
StringBuilder buffer = new StringBuilder();
buffer.Append(objectValue); buffer.Append(" ");
buffer.Append(stringValue); buffer.Append(" ");
buffer.Append(characterArray); buffer.Append(" ");
buffer.Append(characterArray, 0, 3); buffer.Append(" ");
buffer.Append(booleanValue); buffer.Append(" ");
buffer.Append(characterValue); buffer.Append(" ");
buffer.Append(integerValue);
Console.WriteLine("buffer = " + buffer.ToString() + "\n");
```

buffer = hello good bye abcdef abc True Z 7

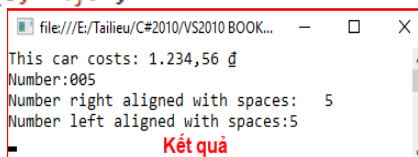
63

63

Lớp StringBuilder (tt)

- Nối chuỗi: sử dụng phương thức **AppendFormat**

```
StringBuilder buffer = new StringBuilder();
string string1 = "This {0} costs: {1:C}. \n";
object[] objectArray = new object[2];
objectArray[0] = "car";
objectArray[1] = 1234.56;
buffer.AppendFormat(string1, objectArray);
string string2 = "Number:{0:d3}. \n" +
    "Number right aligned with spaces:{0, 4}. \n" +
    "Number left aligned with spaces:{0, -4}.";
buffer.AppendFormat(string2, 5);
Console.WriteLine(buffer.ToString());
```



Kết quả

64

64

Lớp StringBuilder (tt)

- Insert, Remove

```
object objectValue = "hello";
string stringValue = "good bye";
char[] characterArray = { 'a', 'b', 'c', 'd', 'e', 'f' };
bool booleanValue = true;
char characterValue = 'K';
int integerValue = 7;
StringBuilder buffer = new StringBuilder();
buffer.Insert(0, objectValue); buffer.Insert(0, " ");
buffer.Insert(0, stringValue); buffer.Insert(0, " ");
buffer.Insert(0, characterArray); buffer.Insert(0, " ");
buffer.Insert(0, booleanValue); buffer.Insert(0, " ");
buffer.Insert(0, characterValue); buffer.Insert(0, " ");
buffer.Insert(0, integerValue);
Console.WriteLine("buffer after Inserts: \n" + buffer + "\n");
buffer.Remove(4, 4);
Console.WriteLine("buffer after Removes:\n" + buffer);
```

```
file:///E:/Tailieu/C#2010/VS2010 BO...
buffer after Inserts:
7 K True abcdef good bye hello
buffer after Removes:
7 K abcdef good bye hello
```

Kết quả

65

65

Lớp StringBuilder (tt)

- Replace

```
StringBuilder builder1 =
    new StringBuilder("Happy Birthday Jane");
StringBuilder builder2 =
    new StringBuilder("good bye greg");
Console.WriteLine("Before replacements:" +
    builder1.ToString() + "\n" + builder2.ToString());
builder1.Replace("Jane", "Greg");
builder2.Replace('g', 'G', 0, 5);
Console.WriteLine("After replacements:" +
    builder1.ToString() + "\n" + builder2.ToString());
```

```
file:///E:/Tailieu/C#2010/VS2010 BOOKS/C#201...
Before replacements:Happy Birthday Jane
good bye greg
After replacements:Happy Birthday Greg
Good bye greg
```

Kết quả

66

66