

Bài 7: BITARRAY TRONG C#

Xem bài học trên website để ủng hộ Kteam: [BitArray trong C#](#)

Mọi vấn đề về lỗi website làm ảnh hưởng đến bạn hoặc thắc mắc, mong muốn khóa học mới, nhằm hỗ trợ cải thiện Website. Các bạn vui lòng phản hồi đến Fanpage [How Kteam](#) nhé!

Dẫn nhập

Ở các bài học trước, chúng ta đã cùng nhau tìm hiểu về [QUEUE TRONG C#](#). Hôm nay chúng ta sẽ cùng tìm hiểu về **BitArray trong C#**.

Nội dung

Để đọc hiểu bài này tốt nhất các bạn nên có kiến thức cơ bản về các phần:

- [BIẾN](#) , [KIỂU DỮ LIỆU](#), [TOÁN TỬ](#) trong C#
- [CÂU ĐIỀU KIỆN](#) trong C#
- Cấu trúc cơ bản của [VÒNG LẶP](#), [HÀM](#) trong C#
- [MẢNG](#) trong C#
- [LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG TRONG C#](#)

Trong bài học này, chúng ta sẽ cùng tìm hiểu các vấn đề:

- BitArray là gì?
- Một số thuộc tính và phương thức hỗ trợ sẵn trong BitArray.

BitArray là gì?

BitArray là một Collections giúp quản lý, lưu trữ một danh sách các bit (0 hoặc 1), được biểu diễn như kiểu **Boolean** (kiểu luận lý). Trong đó **true** biểu thị cho bit 1 và **false** biểu thị cho bit 0.

Nó được sử dụng khi ta cần lưu trữ danh sách các bit mà chưa biết trước số lượng. Ta có thể truy cập đến các phần tử trong **BitArray** thông qua chỉ số như **ArrayList**.

Sẽ có bạn thắc mắc **sao không dùng mảng các đối tượng kiểu bool mà lại dùng BitArray?**

Thì câu trả lời là **BitArray** giúp **tiết kiệm bộ nhớ hơn** rất nhiều:

- Mặc dù kiểu **bool** chỉ lưu 2 giá trị **true** hoặc **false** nhưng lại tốn đến **1 bytes** cho mỗi biến kiểu **bool**.
- Trong khi đó mỗi phần tử trong **BitArray** tốn đúng **1 bit** để lưu trữ.

Trước khi sử dụng ta cần khởi tạo vùng nhớ bằng toán tử **new**.

Lưu ý: là ta không thể khởi tạo 1 BitArray rỗng!

Đầu tiên có thể khởi tạo BitArray và cho biết số phần tử ban đầu của BitArray:

```
/*  
 * Khởi tạo 1 BitArray có 10 phần tử.  
 * Mỗi phần tử có giá trị mặc định 0 (false).  
 */  
BitArray MyBA = new BitArray(10);
```

Nếu bạn không muốn giá trị mặc định là **false** thì bạn có thể chỉ định giá trị mặc định thông qua constructor:

```
/*  
 * Khởi tạo 1 BitArray có 10 phần tử.  
 * Mỗi phần tử có giá trị mặc định 1 (true).  
 */  
BitArray MyBA2 = new BitArray(10, true);
```

Có thể khởi tạo một BitArray từ một mảng **bool** có sẵn:

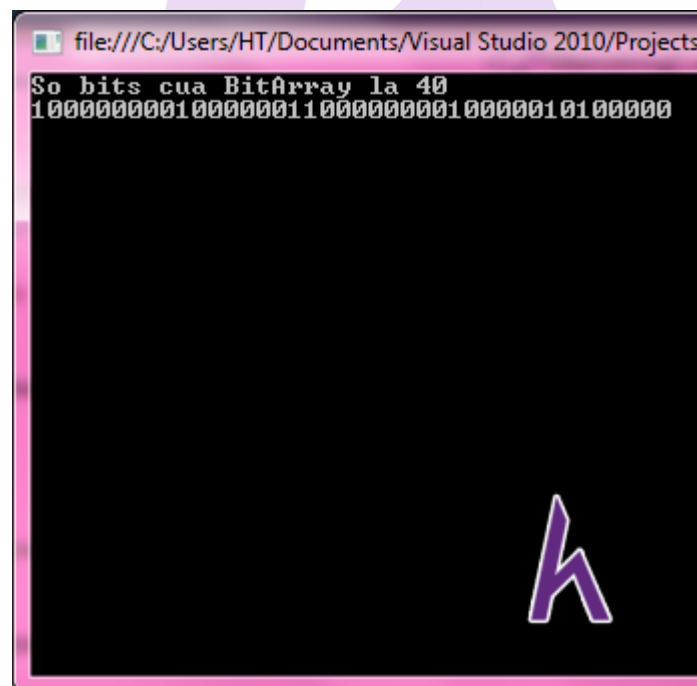
```
/*
 * Khởi tạo 1 BitArray từ một mảng bool có sẵn.
 */
bool[] MyBools = new bool[5] { true, false, true, true, false };
BitArray MyBA3 = new BitArray(MyBools); // 1 0 1 1 0
```

Hoặc khởi tạo một **BitArray** từ một mảng byte có sẵn:

```
/*
 * Khởi tạo 1 BitArray từ một mảng byte có sẵn.
 */
byte[] MyBytes = new byte[5] { 1, 2, 3, 4, 5 };
BitArray MyBA4 = new BitArray(MyBytes);
```

Đối với trường hợp này ta có thể cách lưu trữ của nó như sau:

- Đầu tiên ta đã biết **1 byte = 8 bits**.
- Khi đó trình biên dịch sẽ chuyển các **số kiểu byte** sang dạng **8 bits** và lưu lần lượt vào BitArray. Như vậy trường hợp trên sẽ có **5 số kiểu byte** tương đương với **40 bits** sẽ được lưu vào BitArray.
- Thử in giá trị các phần tử BitArray trên để kiểm chứng:



Để dễ nhìn thì cứ in được **8 bits** thì mình sẽ xuống dòng. Bạn lưu ý đây chỉ là cách mình hiển thị lên màn hình thôi chứ bên trong vẫn là 1 mảng các bit duy nhất:



Gợi ý: Bạn có thể thử đổi các bit trên xem có giống 5 số kiểu byte ban đầu không nhé!

Tương tự bạn cũng có thể khởi tạo một **BitArray** từ một mảng các số nguyên **int** có sẵn:

```
/*  
 * Khởi tạo 1 BitArray từ một mảng int có sẵn.  
 */  
int[] MyInts = new int[5] { 1, 2, 3, 4, 5 };  
BitArray MyBA5 = new BitArray(MyInts);
```

Trường hợp này tương tự như trên nhưng lúc này kiểu **int** chiếm **4 bytes** nên ta sẽ đổi **4 bytes = 32 bits**. Và Trình biên dịch sẽ chuyển mỗi số nguyên **int** sang **32 bits** và lưu vào **ArrayList**:

TÊN PHƯƠNG THỨC	Ý NGHĨA
And(BitArray Value)	Thực hiện phép toán AND bit giữa dãy bit hiện tại với dãy bit Value và trả về 1 BitArray là kết quả của phép toán trên.
Clone()	Tạo 1 bản sao từ BitArray hiện tại.
CopyTo(Array array, int Index)	Thực hiện sao chép tất cả phần tử trong BitArray sang mảng một chiều array từ vị trí Index của array.
Get(int Index)	Trả về giá trị của bit tại vị trí Index trong BitArray.
Not()	Trả về 1 BitArray là kết quả của phép toán NOT trên dãy bit hiện tại.

Lưu ý:

- Các phép toán **AND, OR, NOT, XOR** đã được trình bày trong bài [TOÁN TỬ TRONG C#](#) các bạn có thể xem lại.
- Các phép toán **AND, OR, NOT, XOR** phải được thực hiện trên 2 **BitArray** có **cùng độ dài** nếu không sẽ báo lỗi.
- Các phép toán **AND, OR, NOT, XOR** sẽ làm thay đổi cả BitArray gọi nó.
Ví dụ:

```
BitArray A = new BitArray(5);
BitArray B = new BitArray(5, true);
A.And(B);
```

Thì kết quả của phép **AND** sẽ được cập nhật giá trị vào BitArray A.

Một ví dụ đơn giản về sử dụng BitArray

Một chương trình đơn giản về BitArray:

```
// Khởi tạo 1 BitArray từ mảng bool có sẵn
bool[] MyBool2 = new bool[5] { true, false, true, true, false };
BitArray MyBA6 = new BitArray(MyBool2);

// Khởi tạo 1 BitArray có 2 phần tử và có giá trị mặc định là 1 (true)
bool[] MyBool3 = new bool[] { false, true, true, false, false };
BitArray MyBA7 = new BitArray(MyBool3);

Console.Write(" Gia tri cua MyBA6: ");
PrintBits(MyBA6, 5);

Console.Write(" Gia tri cua MyBA7: ");
PrintBits(MyBA7, 5);

Console.WriteLine(" Thuc hien cac phep toan AND, OR, NOT, XOR tren MyBA6 va
MyBA7: ");

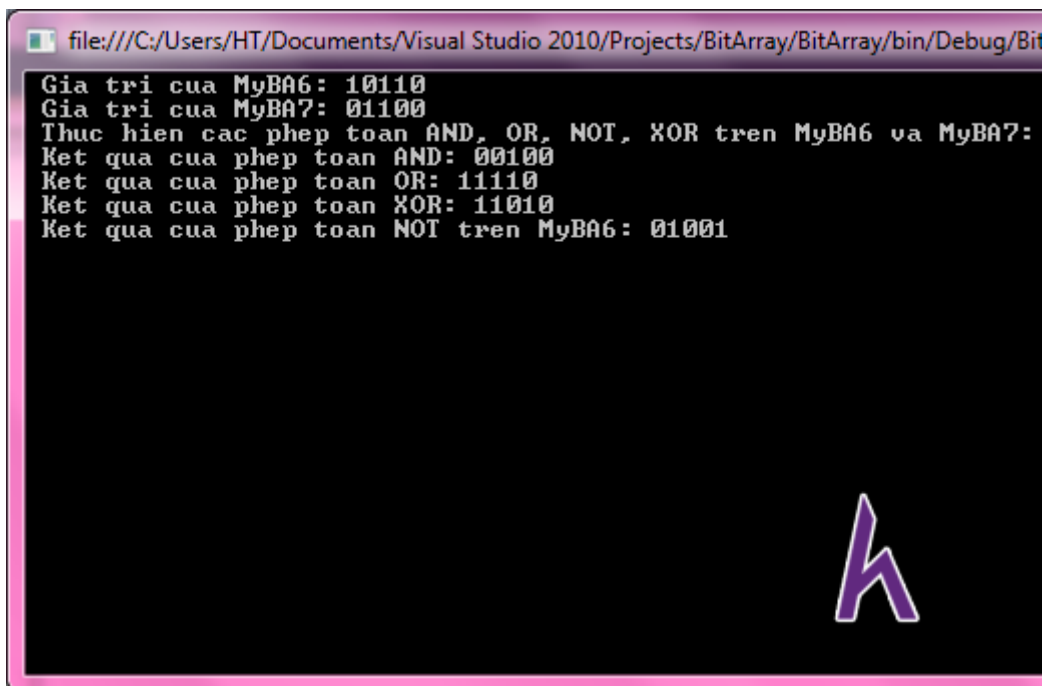
// thực hiện sao chép giá trị của MyBA6 ra để không làm thay đổi nó
BitArray AndBit = MyBA6.Clone() as BitArray;
AndBit.And(MyBA7);
Console.Write(" Ket qua cua phep toan AND: ");
PrintBits(AndBit, 5);

BitArray OrBit = MyBA6.Clone() as BitArray;
OrBit.Or(MyBA7);
Console.Write(" Ket qua cua phep toan OR: ");
PrintBits(OrBit, 5);

BitArray XorBit = MyBA6.Clone() as BitArray;
XorBit.Xor(MyBA7);
Console.Write(" Ket qua cua phep toan XOR: ");
PrintBits(XorBit, 5);

BitArray NotBit = MyBA6.Clone() as BitArray;
NotBit.Not();
Console.Write(" Ket qua cua phep toan NOT tren MyBA6: ");
PrintBits(NotBit, 5);
```

Kết quả khi chạy đoạn chương trình trên:



```
file:///C:/Users/HT/Documents/Visual Studio 2010/Projects/BitArray/BitArray/bin/Debug/Bit
Gia tri cua MyBA6: 10110
Gia tri cua MyBA7: 01100
Thuc hien cac phep toan AND, OR, NOT, XOR tren MyBA6 va MyBA7:
Ket qua cua phep toan AND: 00100
Ket qua cua phep toan OR: 11110
Ket qua cua phep toan XOR: 11010
Ket qua cua phep toan NOT tren MyBA6: 01001
```

Các bạn có thể tự kiểm tra lại xem có đúng không nhé!

Kết luận

Nội dung bài này giúp các bạn nắm được:

- BitArray là gì?
- Một số thuộc tính và phương thức hỗ trợ sẵn trong BitArray.

Bài học sau chúng ta sẽ cùng tìm hiểu về [GENERIC TRONG C#](#).

Cảm ơn các bạn đã theo dõi bài viết. Hãy để lại bình luận hoặc góp ý của mình để phát triển bài viết tốt hơn. Đừng quên **"Luyện tập – Thử thách – Không ngại khó"**.