

## 1. Type Safety in C#

- تعني أن المترجم يمنع استخدام أنواع بيانات غير متوافقة.
- مثال:

```
int x = "hello"; // Error: can't assign string to int
```

- الغرض منها: تقليل الأخطاء وقت التشغيل (Runtime).

## ✓ 2. Bitwise Operators

تُستخدم هذه العمليات للتعامل مع البتات (bits) مباشرة في الأعداد الصحيحة.

المُعامل	الاسم	المثال	النتيجة	الشرح
&	AND (و)	5 & 3	1	يحتفظ بالبتات المشتركة فقط.
	OR (أو)	5   3	7	
^	XOR (أو الحصري)	5 ^ 3	6	يضع 1 إذا كان البتتين مختلفين.
~	NOT (عكس البتات)	~5	-6	يعكس كل البتات (0 → 1 والعكس).
<<	Left Shift	5 << 1	10	(×2). يحرك البتات لليسار.
>>	Right Shift	5 >> 1	2	(÷2). يحرك البتات لليمين.

## 3. System.OverflowException & checked / unchecked

Problem: Overflow يحصل , int عند تجاوز الحد الأقصى لقيمة نوع مثل

checked

- يرصد الخطأ ويرمي استثناء:

checked

```
{  
    int x = int.MaxValue + 1; // Exception!  
}
```

unchecked



- يتجاهل الخطأ ويكمل:

unchecked

```
{  
    int x = int.MaxValue + 1; // No error, but wrong value  
}
```

---

#### 4. Hashing vs Encoding

Feature	Hashing	Encoding
Purpose	Data integrity (e.g., passwords)	Data transmission/format (e.g., URLs)
Reversible	 No	 Yes
Examples	SHA256, MD5	Base64, UTF-8
Use case	Verify if data is unchanged	Safely store/transfer text

---

#### 5. Garbage Collector in C# (Not Shell Book style )

- What it does:  
Automatically frees up memory by deleting unused objects.
- When:  
No reference is pointing to an object → GC cleans it.
- Benefits:
  - Prevents memory leaks.
  - No need for manual delete.

- **Forcing it:**

**GC.Collect(); // Not recommended unless necessary**

- **GC works in generations:**
  - **Gen 0: Short-lived objects (like temp variables)**
  - **Gen 1: Medium-lived**
  - **Gen 2: Long-lived (like static fields)**