* **Diff Between endl & \n in c++ :**

1. **\n**
   * معناها "new line" = سطر جديد.
   * وظيفتها إنها **تنزل المؤشر للسطر الجديد فقط**.
   * **ما بتعمل flush (إرسال إجباري) للـ buffer**.

1. **endl**
   * معناها "end line" = نهاية السطر.
   * وظيفتها إنها **تنزل للسطر الجديد + تعمل flush للـ buffer** (يعني تجبر النظام يطبع المحتوى فورًا).

### 🔹 متى تستخدم كل واحد؟

* **استخدم \n**:
  + إذا كنت بتطبع نصوص عادية أو بتعمل حلقات طباعة كثيرة (أسرع من endl   
    لانه مافي flush
  + مثال: طباعة بيانات كبيرة بسرعة.
* **استخدم endl**:
  + إذا كنت **بدك تتأكد إن النص انطبع مباشرة** (مهم في الـ debugging أو عند التعامل مع ملفات/شبكات).
  + مثال: طباعة رسالة خطأ لازم تبين فورًا.
* **استخدم endl + \n** :
  + إذا كان في على فرض 100 جملة بدي اطبعها لو استخدمت \n ==🡺حيكون بطيء
  + إذا كان في على فرض 100 جملة بدي اطبعها لو استخدمت \endl ==🡺حيكون بطيء
  + الحل 🡺 اعمل اول 50 \n ->بعدها اعمل endl -> عشان يعمل fluch buffer

وال 50 الباقيين اعمل نفس الحركة

.

* **Literals and Escape sequences literals**
* **Literals** are fixed values written directly in a program. They represent constant data.
  + **Numeric literals:** 10, 3.14
  + **Character literals:** 'A', '\n'
  + **String literals:** "Hello", "123"
  + **Boolean literals:** true, false
  + **Null literal:** null (in some languages)
* **Escape Sequences** are special codes used inside **character or string literals** to represent characters that cannot be typed directly. They always start with \.
  + Examples:
    - \n → newline
    - \t → tab
    - \" → double quote
    - \\ → backslash
* **Note:** Escape sequences are also considered **literals** (character or string literals written in a special form).

.

* **Variables**

**C++ Data Types**

**│**

**├── 1. Fundamental (Primitive)**

**│ │**

**│ ├── Integral**

**│ │ ├── int**

**│ │ ├── short int**

**│ │ ├── long int**

**│ │ ├── unsigned int**

**│ │ ├── char**

**│ │ └── wchar\_t (wide character)**

**│ │**

**│ ├── Floating**

**│ │ ├── float**

**│ │ └── double**

**│ │**

**│ ├── Boolean**

**│ │ └── bool**

**│ │**

**│ └── Void**

**│ └── void**

**│**

**├── 2. Derived**

**│ ├── Array**

**│ ├── Pointer**

**│ ├── Reference**

**│ └── Function**

**│**

**└── 3. User-defined**

**├── struct**

**├── class**

**├── union**

**├── enum**

**└── typedef / using**

.

* **Datatype sizes**

| **Data Type** | **Size (in bytes)** | **Range (approximate)** |
| --- | --- | --- |
| **bool** | 1 | true / false |
| **char** | 1 | -128 to 127 (signed) 0 to 255 (unsigned) |
| **wchar\_t** | 2 or 4 (depends on compiler) | 0 to 65,535 (if 2 bytes) |
| **char16\_t** | 2 | 0 to 65,535 |
| **char32\_t** | 4 | 0 to 4,294,967,295 |
| **short** | 2 | -32,768 to 32,767 |
| **unsigned short** | 2 | 0 to 65,535 |
| **int** | 4 | -2,147,483,648 to 2,147,483,647 |
| **unsigned int** | 4 | 0 to 4,294,967,295 |
| **long** | 4 (on 32-bit), 8 (on 64-bit systems) | -2,147,483,648 to 2,147,483,647 (if 4 bytes) |
| **unsigned long** | same as long | 0 to 4,294,967,295 (if 4 bytes) |
| **long long** | 8 | -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807 |
| **unsigned long long** | 8 | 0 to 18,446,744,073,709,551,615 |
| **float** | 4 | ~ ±3.4e38 (7 decimal digits precision) |
| **double** | 8 | ~ ±1.7e308 (15 decimal digits precision) |
| **long double** | 8, 12, or 16 (depends on compiler) | even larger range (~18-19 digits precision) |
| **void** | no storage | no value |
| **string** | 12 |  |

* **Type Modifiers**

**We can modify some of the fundamental data types by using type modifiers . There are 4 type modifiers in C++ , they are :**

**1-signed**

**2-unsigned**

**3-short**

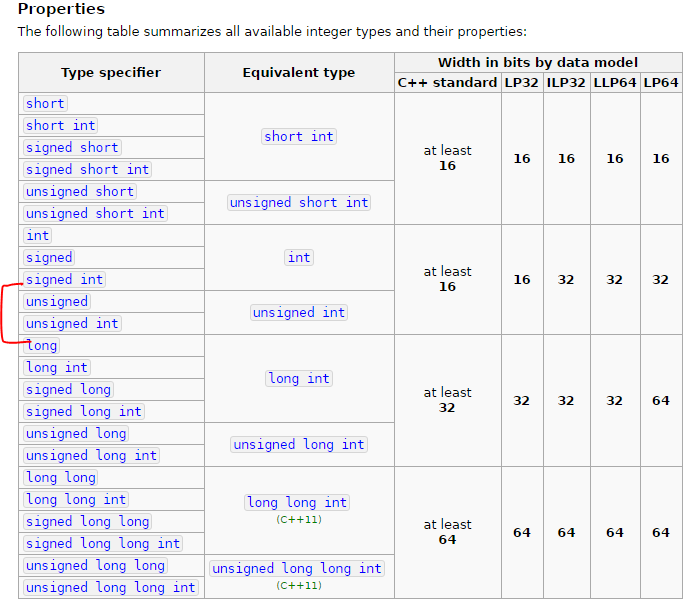
**4-long**

* **Note :** 
  + - **We can only modify the following datatypes with the type modifiers:**

**1 – int**

**2- double**

**3-char**



منطقيا : عمر الشخص 🡺 مش رح يتجاوز 200 سنة  
 فاذا عرفته ب int  
 بتكون تعمل جريمة بحق البشرية