要知道 `_BitInt(n)` 是甚麼我肯定要先看看reference

因此首先，我先打開了C23 的reference
https://en.cppreference.com/w/c/23

並找到了 `_BitInt(n)` 所列的位置
https://en.cppreference.com/w/c/language/arithmetic_types

reference裡面是這樣說明的：

- `_BitInt(n)` (also accessible as `signed _BitInt(n)`), the bit-precise signed integer types (where `n` is replaced by an integer constant expression denoting the precise width (including the sign bit), that cannot be larger than `BITINT_MAXWIDTH` from `<limits.h>`)

看到這個解釋的時候，我第一個想法是`_BitInt(n)` 是可以透過告訴電腦我現在要一個 n bit 大小的integer，例如 `_BitInt(64)` 就會等於一個 int64_t

但我後來查閱了一個blog，裡面提到了 `_BitInt(n)` 更多的特性和用途
https://blog.tal.bi/posts/c23-bitint/

他和 C++ 的bitset 最大的差別在於 `_BitInt(n)` 可以做數學符號的運算 (+ - * /)，而 bitset 只能做位元的操控

另一方面，在一些特定的演算法或是function 中，透過 `_BitInt(n)` 可以十分有效的改善一個演算法或是 function 的使用空間。例如加密、圖像、IP、嵌入系統，不再受限於原本基礎的那幾個資料型態。

雖然在現在的個人電腦中記憶體容量甚大，但在一些嵌入式系統或是特定用途的電子系統中，記憶體空間其實還是很寶貴的。因此這時 `_BitInt(n)` 在特定情況下可以很好的取代 int 和 long，節省了更多空間和開發的可能。

而且因為有了 `_BitInt(n)`，因此可以讓開發者在設定上更為彈性。

我原本以為只是一個可以自己設定要多大的資料型態，但後來發現其實這可以影響到很多開發的層面和記憶體空間的調配。