Summary Guide

What is the purpose of typedef?

The typedef keyword is used in C to create an alias for existing data types. This improves code readability, reduces repetition, and simplifies the syntax when dealing with complex types like structs, enums, and unions.

How are bit fields declared and what are their size limitations?

Bit fields are declared within a struct using the syntax: unsigned int field_name : number_of_bits;. The size limitations depend on the underlying type, usually int, and the maximum number of bits it can represent (typically 32 bits on most systems).

What happens if a bit field overflows?

If a value larger than what a bit field can hold is assigned, the extra bits are truncated, and only the least significant bits are stored. This can lead to unexpected behavior or data loss.

How is typedef used with complex types like structs and unions?

Typedef allows you to assign a new name to a struct or union type, enabling you to use that name without needing to repeat the full declaration. For example: typedef struct { int x; int y; } Point; allows you to declare variables as Point p1; instead of struct Point p1;.

What is the default underlying type of an enum?

The default underlying type of an enum in C is int. Each enumerator is assigned an integer value starting from 0 unless explicitly defined otherwise.

How is a union different from a struct?

In a struct, each member has its own memory, and the total size is the sum (plus padding). In a union, all members share the same memory location, and the size of the union is equal to the size of its largest member.

When is using a union more memory-efficient?

Using a union is more memory-efficient when you need to store multiple types in the same memory space but will only use one type at a time. It helps reduce memory usage, especially in embedded systems or memory-constrained environments.