Types of Pointers in C

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Pointer Type | Description | Dos | Don'ts | Example |
| Null Pointer | A pointer that does not point to any valid memory location. | - Always initialize pointers to `NULL` before use. | - Never dereference a null pointer. | int\* ptr = NULL; if (ptr != NULL) {  printf("%d", \*ptr); } |
| Void Pointer | A pointer that can point to any data type. | - Use type casting when dereferencing. | - Avoid dereferencing without typecasting. | void\* ptr; int x = 10; ptr = &x; printf("%d", \*(int\*)ptr); |
| Dangling Pointer | A pointer that points to a memory location that has been deallocated. | - Set pointers to `NULL` after freeing memory. | - Never dereference a dangling pointer. | int\* ptr = malloc(sizeof(int)); free(ptr); ptr = NULL; |
| Function Pointer | A pointer that points to a function. | - Use for callbacks or implementing function tables. | - Avoid confusing function signatures with incorrect typecasting. | int add(int a, int b) { return a + b; } int (\*ptr)(int, int) = add; printf("%d", ptr(3, 4)); |
| Pointer to Pointer | A pointer that stores the address of another pointer. | - Useful for dynamic memory allocation and multi-level pointer access. | - Be cautious when dereferencing multiple levels. | int x = 5; int\* ptr = &x; int\*\* ptr\_to\_ptr = &ptr; printf("%d", \*\*ptr\_to\_ptr); |
| Wild Pointer | A pointer that points to an unknown or uninitialized memory location. | - Initialize all pointers before use. | - Never leave pointers uninitialized or point them to random memory. | int\* ptr; // Uninitialized pointer - unpredictable behavior |
| Const Pointer | A pointer to a constant value. The pointer itself can be changed, but the data it points to cannot. | - Use for read-only data to prevent accidental modification. | - Do not attempt to modify the data pointed to by a const pointer. | const int\* ptr = &x; // \*ptr = 10; // error, data is constant |
| Pointer to Constant Data | A pointer that points to a constant data, but the pointer can be modified to point to other addresses. | - Useful for pointing to constant data or data in a read-only segment. | - Don't attempt to modify the data being pointed to. | int x = 10; const int\* ptr = &x; ptr = &x; // Valid, but can't modify \*ptr |