Dangling vs Wild Pointers (C++) – Quick Revision Sheet

Core Definitions

Dangling Pointer - Pointer that refers to memory whose **lifetime has ended** - Memory was **valid earlier**, now **invalid** - Dereferencing → **Undefined Behavior (UB)**

Wild Pointer - Pointer that is **declared but not initialized** - Points to **garbage / random memory** - Dereferencing → **Undefined Behavior (UB)**

Key Difference (Interview Gold)

> **Wild pointers are born bad. Dangling pointers become bad.**

Aspect	Dangling Pointer	Wild Pointer
Initialized?	Yes	No
Ever valid?	Yes	No
Root cause	Lifetime ended	Garbage value
Debug difficulty	High	Medium

Common Causes

Dangling Pointer

- `delete` called but pointer reused
- Returning address of local variable
- Object goes out of scope
- Multiple pointers, one deletes memory
- STL container reallocation (`vector`)

Wild Pointer

- Declared but not initialized
- Forgot to assign before dereference

Classic Code Examples

Dangling Pointer

```
int* p = new int(10);
delete p;
*p = 5;   // ❌ UB (dangling)
```

```
int* foo() {
    int x = 10;
    return &x;  // ❌ dangling
}
```

Wild Pointer

```
int* p;
*p = 10;  // ❌ UB (wild)
```

Prevention Techniques

General Rules

- Always initialize pointers
- Never use pointers after `delete`
- Avoid raw owning pointers

Best Practices

- Set to `nullptr` after delete
- Prefer **values** or **references**
- Use **RAII / Smart Pointers**

```
std::unique_ptr<int> p = std::make_unique<int>(10);
```

Smart Pointer Notes (SDE-2+)

- `unique_ptr` → single ownership
- `shared_ptr` → reference counting
- `weak_ptr` → breaks cycles

⚠ Still unsafe if raw pointer from `.get()` outlives owner

Tools to Detect Bugs

- AddressSanitizer (`-fsanitize=address`)
- Valgrind
- Clang-Tidy / Static Analysis

Rapid Interview Q&A

Q: Can a pointer be both wild and dangling?
A: No

Q: Is deleting `nullptr` safe?
A: Yes

Q: Which is harder to debug?
A: Dangling pointer

One-Line Answers (Memorize)

- **Dangling Pointer:** Pointer to memory whose lifetime has ended
- **Wild Pointer:** Uninitialized pointer pointing to random memory

---

✅End of 1-page revision sheet