이 메모는 명확하게 **개발자의 철학, 경험에 기반한 직관, 리팩토링 중의 맥락적 해석**, 그리고 **실전에서 체득한 안정성 중심 설계 원칙**을 기록한 *기술 회고 겸 설계 원리의 기초 메모*로 보입니다.

형식은 자유로운 **비공식 메모**지만, 내용적으로는 다음 3가지 범주로 구조화할 수 있습니다:

**🧱 1. 개발 철학 및 저수준 이해 기반 자원 관리관**

**🔸 주요 메시지:**

* 포인터를 가상 주소로 이해하고 있으며, "소유권" 개념을 실무적으로 받아들임
* 스마트 포인터는 얕은 복사 시 **소유권 이전 vs 공유**의 차이를 명확히 분리
* 소유권 개념은 **책임과 자원 반납의 명확화**를 위한 수단으로 수용
* GC는 “있다고 나쁜 것은 아니나, **믿고 방기하면 안 된다**”는 태도

**🔍 해석:**

이는 전통적인 **RAII 기반 설계 철학**을 지향하며, 소유권과 책임을 명시적으로 코드화해야 안정성이 높아진다는 명확한 가치 판단을 내포하고 있음.

**🏗️ 2. 리팩토링 히스토리 및 클래스 분류 체계 정리**

**🔸 분류 기준:**

| **구분** | **주요 클래스** | **특징** |
| --- | --- | --- |
| 전기형 | CUserBitmap, CUserBitmapProcess | 클래스화 시도, 절차적 코드 기반, 동적 메모리 흔적 |
| 중기형 | CBitmapAction + 상속 구조 | 공통 기능 부모화, 상속 구조 정립 시기 |
| 후기형 | CColorConv, CBlobColoring, CBasicBone | STL 도입, 벡터 대체, 소유권 제어 클래스 도입 |

**🔍 해석:**

"전기형 → 중기형 → 후기형"이라는 리팩토링 단계 구분은, 개발자가 소스 구조의 복잡도와 설계 수준을 내적으로 분류한 결과입니다. 특히 후기형은 STL 기반 리팩토링과 책임 분리 구조를 반영한 구조로 인식됩니다.

**⚙️ 3. 코딩 원칙과 안전성 철학**

**🔸 강조된 원칙들:**

* 디버깅 쉬운 구조 > 성능
* 재귀 호출 및 goto → 지양
* 명확한 소유권, 네임스페이스 분리, 전역 변수 회피
* 함수는 최소 bool로 성공/실패 표현
* "자원은 쓰고 나면 반드시 반납"
* 소멸자에서 자원 정리 보장
* 구조보다 중요한 것은 **안정성**
* "무조건 실패해도, 다른 시스템은 살려놔야 한다"

**🔍 해석:**

**현장 중심의 보수적이고 강력한 안전 중심 철학**이며, 추상 이론보다는 실전에서 생긴 문제에 대해 반복 학습을 통해 체화된 경험 중심 태도입니다. "죽더라도 조용히 혼자 죽어야 한다"는 표현은 다중 시스템에서의 *격리성과 복구 가능성*에 대한 명확한 철학입니다.

**🧭 이 메모가 갖는 의미**

| **항목** | **해석** |
| --- | --- |
| 목적 미확정 | 설계 문서로 쓸 의도는 아니었음. **마인드 정리 혹은 철학 회고에 가깝다** |
| 형식 | 내러티브 + 단문 중심의 자유 메모. 그러나 내용은 철저하게 실용주의적, 고안정성 중심 |
| 위치 | 개발자가 중간 이상 수준의 리팩토링 설계 철학에 도달하고 있는 **전이적 단계**에서 작성된 것으로 보임 |

**✍️ 만약 정리해서 활용한다면?**

1. \*\*"개발자 철학/개발 규약 메모"\*\*로 별도 문서화
2. **클래스별 구조 변화 요약표 + 적용 설계 원칙 매핑**
3. 내부 위키나 팀 설계 원칙 초안으로 일부 활용 가능