Day-2

C++ Philosophy

제로 오버헤드 원칙 - 성능적 측면

- 사용하지 않는 것에 대해서는 비용을 지불하지 않는다
- 실제로 사용하는 것이 있다면, 수작업으로 코드를 작성한 것 만큼 효율적이어야 한다.

정적으로 타입이 지정된 언어 - 타입 안정성

- 런타임이 아닌 컴파일 타임에 많은 버그를 잡아냄
  - 이는 많은 상용 어플리케이션에서 중요한 고려사항
- 타입 어노테이션으로 코드 가독성 향상
  - 변수나 상수를 선언할 때 타입을 명시적으로 선언해줌으로써 어떤 타입의 값이 저장될 것인지 알려주는 방법
- 컴파일러 최적화 및 런타임 효율성 향상
- 사용자가 직접 타입 시스템을 정의할 수 있음

추가

- 프로그래밍 모델
  - 구획화, 실제 문제를 해결하는 경우에만 기능만 추가하고, 완전한 제어를 허용합니다.
- 예측 가능한 런타임(제약 조건 하에서)
  - 가비지 컬렉션 없음, 동적 타이핑 없음 → 실시간 시스템
- 낮은 리소스
  - 낮은 메모리 및 에너지 소비 → 제한된 하드웨어 플랫폼
- 정적 분석에 적합 → 안전이 중요한 소프트웨어
- 이식성 → 최신 C++ 표준은 이식성이 뛰어남

C++은 누구를 위한 언어?

- 하드웨어를 매우 잘 사용하고 추상화를 통해 복잡성을 관리하려는 사람들을 위한 언어
- C++은 모든 사람을 위한 언어가 아님
- 어떤 종류의 정밀도를 목표로 하는 전문가를 위한 도구로 만들어짐