Day-3

C++ Weaknesses

C++이 어려운 이유

C는 발에 총을 쏘기 쉽고, C++는 더 어렵지만 총을 쏘면 다리 전체가 날아갑니다.

C++ 사용의 문제점은... 이미 이 언어에는 무엇이든 하기 전에 모든 것을 알아야 한다는 경향이 강하다는 것입니다.

20년 동안 C++를 사용해왔음에도 불구하고 처음으로 오류나 경고 없이 사소하지 않은 코드 덩어리를 컴파일할 때면 의심이 듭니다. 일반적으로 좋은 징조는 아닙니다.

C++의 단점

하위 호환성(Backward-compatibility)

- 기술 및 컴퓨터 분야에서 새 제품이 이전 제품을 염두에 두고 만들어진 제품에서 별도의 수정 없이 그대로 쓰일 수 있는 것을 뜻함
- C++은 C와의 하위 호환성에 집중
- 따라서 C++은 아래 서술한 단점이 존재함
    - 종종 C에서 비롯된 위험한 기본값과 구조는 제거하거나 변경할 수 없음
    - 개발진의 노력에도 불구하고, 새로운 기능에는 많은 사용자 경험 후에야 명백히 밝혀진 결함이 존재, 수정이 불가능함
    - C++ 사용은 생산성이나 표현력에서 거의 이득이 없고 코드 명확성에 막대한 대가를 치르는 반면 개발자의 인지적 부담은 점점 더 커지고 있음

C++의 비평과 대체

- epoch
    - 이전 버전과 호환되는 언어 진화 메커니즘

    - C++의 문제를 해결하는 동시에 C++의 이전 버전과 이후 버전의 호환성을 유지하는 메커니즘
- C++의 목표와 우선순위
    - 언어 목표
        1. 성능이 중요한 소프트웨어
        2. 소프트웨어와 언어의 진화
        3. 간단하고, 읽고, 이해하고, 작성하기 쉬운 코드
        4. 실질적인 안전 보장 및 테스트 메커니즘
        5. 빠르고 확장 가능한 개발
        6. 최신 하드웨어 아키텍처, OS 플랫폼 및 환경
    - 비목표
        1. 안정적인 언어 및 라이브러리 ABI
        2. 이전 버전 또는 이후 버전과의 호환성
        3. 소스 코드나 재구축 기능이 없는 레거시 컴파일 라이브러리
        4. 기존 컴파일 및 연결 모델 지원
- Carbon Language
    - 구글에서 개발 중인 다목적 범용 프로그래밍 언어
    - C++을 대체하거나 기존의 레거시 C++코드와의 상호 운용성 달성을 목표로 함
- Circle C++ 컴파일러
- Cppfront
    - C++을 10배 더 간단하고 안전하게 만들 수 있을까?
    - 실험적인 컴파일러
    - 현재 매우 불완전한 상태

Rust

- C++의 대체재로써 등장
- C++과 마찬가지로 성능과 zero-abstract, 오버헤드에 중점을 둠
- 메모리 버그와 같은 C++에 영향을 미치는 많은 취약점을 방지하도록 설계됨

- 플랫폼 간 호환성을 촉진시킴

Zig(2016)

- C를 대체할 수 있는 최소한의 오픈소스 프로그래밍 언어
- C와 완벽하게 상호 운용되도록 만들어짐
- C/C++ 컴파일러 포함

왜 새로운 언어로 전환하는 것이 어려울까?

- 완벽한 언어는 없음, 항상 새로운 '빛나는' 언어가 존재함
- Alignment : 모든 개발자가 새로운 언어로 전환하도록 강제함
- 상호 운용성
  - 수천억 줄의 기존 코드
  - 심각한 설계 제약을 부과하는 C 및 C++ 코드와 상호 운용 되어야 함
- 에코시스템 : 지난 40년 동안 개발된 도구 및 라이브러리 부족
- 시간 및 비용