1) RUST는 C/C++의 수행속도에 비견되는 속도를 가진 함수형 언어다. 이 언어의 특성을 요약하고, Python등 기존 언어와 비교하라

Rust는 병렬처리가 잘되고 속도가 빠르다. C/C++에 버금가는 함수형 언어이다.

Python 은 개념을 쉽게 이해할 수 있게 짤 수 있다. 전공자가 아닌 사람도 사용하기 비교적 쉽고, 여러 분아에 활용될 수 있다. 반면에 Rust는 세련된 에러 처리나 소유권 등의 개념 등이 이해하고 구현하기 어렵다.

Rust와 마찬가지로 Python 도 objected oriented 언어이다. (객체 지향 언어는 abstraction, encapsulation, inheritence, polymorphism의 4가지 특징을 가진다.)

python는 인터프리터 언어인 반면에 Rust는 컴파일 기반의 언어이다.(컴파일 언어가 일반적으로 인터프리터를 이용하여 실행시키는 것보다 훨씬 빠르게 동작한다.

**Rust의 핵심은 메모리 오류를 없애면서 C/C++에 버금가는 빠른 속도를 내는 것이다.**

메모리 오류를 없애기 위해서 lifetime과 소유권 개념이 중요하다. lifetime으로 C/C++에서 dangling pointer에 의한 memory leak와 같은 것을 방지할 수 있다.

또한, 빠른 속도를 내기 위해서 java의 static과 같은 시스템의 퍼포먼스에 악영향을 주는 기능(Static은 Garbage Collecter의 관리 영역 밖에 존재하므로 Static을 자주 사용하면 프로그램의 종료시까지 메모리가 할당된 채로 존재하므로 자주 할당하면 시스템의 퍼포먼스에 악영향을 준다.)이 다른 언어와 비교해서 현저히 적다.

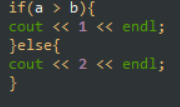
2) 첨부한 파일 내용에 따라 자기가 짠 프로그램에 그런 유형의 오류가 있는 예를 보이면서 스스로 어떻게 프로그램을 짤지 설명하라.

- 1. 베끼기(생략) 베끼면 당장 코드를 짜는데 편리하고, 그 코드를 이용하여 시간을 단축시킬 수 있으나 그 코드에 문제가 생기게 되면 유지보수에 더 많은 시간이 걸릴 수 있다. 베낄때도 이해하고 베껴야 한다.

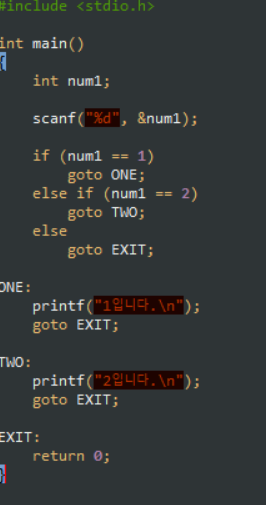
- 2. functional code

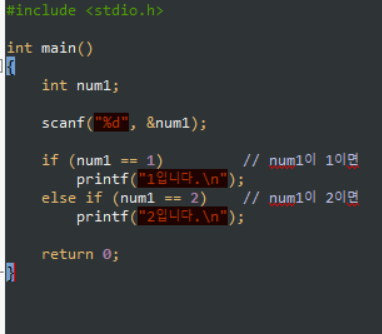
main()에 그냥 다 넣은 코드들을 함수형 코드로 묶어주어 함수를 호출하는 방식으로 사용하게 되면, 코드를 짜는 입장에서는 번거로울 수 있으나, 이후에 문제가 있는 함수나 클래스를 변경하는 방식을 통해서 유지보수에 더 나은 코드가 된다.

- 3. 비표준적 띄어쓰기. (indent 안맞추는 것)

이런 식으로 코드를 짜면 알아보기 힘들다. 코드가 서로 구별되게 indent를 맞춰줘야 한다.

- 4. goto 사용. 무분별한 goto 사용은 프로그램이 뒤죽박죽 되게 만들어서 코드를 이해하는 데나, 유지보수하는데 모두 어렵다.

와 같이 goto로 뒤죽박죽 되는 코드를

와 같이 가독성도 좋고 프로그램의 동작도 복잡하지 않게 바꿔서 써야 한다.

- 5. 유형을 선언하지 않기

유형을 선언하지 않게 되면 해당 개체가 어떤 기능을 수행하고 코드 내에서 어떤 역할을 하는지 알기 어렵고, 그 결과 유지보수, 가독성에 문제가 생긴다.

- 6. 요요코드 작성

주로 객체지향에서 발생하는 문제인데. 상속에 대해 너무 길고 복잡하게 하면 프로그래머가 이러한 코드를 왔다갔다하면서 파악하기 어렵고 그에 따라 유지보수하기도 어렵다.

이를 극복하기 위해서 상속을 너무 깊게(길고 복잡하게) 사용하지 않고 얕고 알아보기 쉽게 써서 유지보수가 쉽고 가독성이 좋게 코드를 작성해야 한다.

- 7. 나만의 데이터 구조 작성

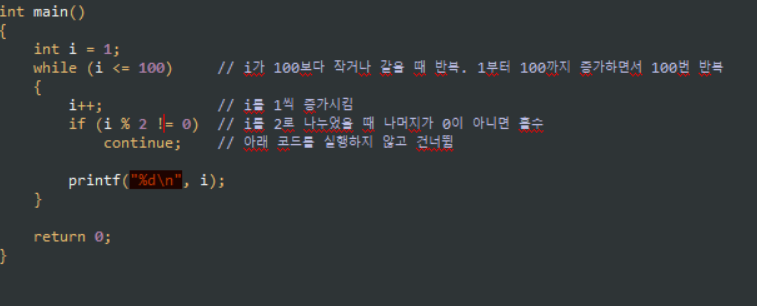
데이터 구조 과정에 숙달되지 않은 사용자가 라이브러리에 존재한 데이터구조를 활용해서 코드를짜지 않고 데이터 구조를 스스로 만들면, 협업하는 사람들에게 혼란을 줄 수 있고, 속도, 안정성에 대한 문제가 발생할 수 있다.

되도록이면 라이브러리에 있는 코드를 사용하고, 사용자가 만든 데이터 구조를 활용하기 위해서는 안정성이 보장되어야 한다.

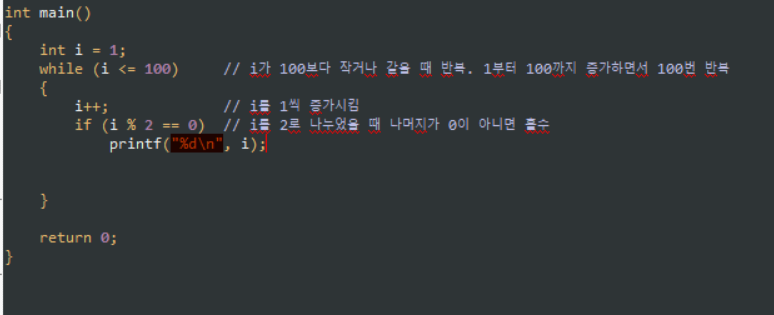
- 8. 구식 루프

루프를 이용하면 무한 루프와 같은 함정에 빠질 가능성도 존재하고, 이중 루프를 넘어가면 안정성에도 문제가 생길 수 있기 때문에. 루프를 사용하는 대신에 다른 템플릿이나 테이블을 사용하는게 더 가독성이 좋고 편리할 수 있다.

- 9. 중간에 루프에서 이탈하기.

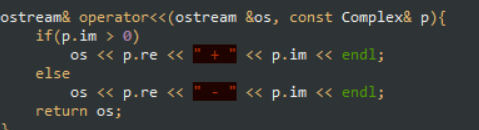


의 코드 대신에



이런 식으로 continue를 쓰지 않고도 사용할 수 있다. 코드의 가독성이 떨어지고 더 복잡한 코드가 될 수 있기 때문에 continue, return 과 같은 코드의 불변성을 변화시킬 수 있는 코드는 최대한 작성을 지양해야 한다.

- 10. 연산자와 함수 재정의



이렇게 << 연산을 바꿔서 출력을 바꾸는 기능이나 +, \*와 같은 연산자를 재정의해서 (2 + 3i) \* (4 + 3i)와 같은 연산을 더 쉽게 사용할 수 있으나, 이것은 기존의 연산자를 바꾸는 것이므로 문제가 될 수 있다. 자신 혼자만 프로그래밍을 하는 것이 아니라 협업이 요구되는 상황에서는 코드가 팀원들에게 헷갈리게 작용할 수 있다. 협의가 된 것으로 바꾸거나 개인적으로는 바꾸지 않는게 좋다.

느낀 점)

1 - Rust의 특징에 대해 중점적으로 기술하였습니다. 마침 Rust를 공부하는 스터디를 하는 도중이었는데 배운 기본 지식을 다시 한번 되새기는 시간이 되었습니다.

2 – 유지보수에 대한 이에 대한 개념을 어느정도는 알고 있었으나 다시금 되새기는 시간이 되었고, 명확하게 정리가 되었습니다.