게임 알고리즘 보고서

*“Queue in Rust & C++”*

201513071\_김시하

1. 문제 내용
2. 문제 분석
3. 주요 코드
4. 실행 결과
5. Error Log
6. 결론 & 후기

문제 내용

시작에 앞서 문제에 대하여 서술하겠음

1. C++ 에서 템플릿을 사용하여 큐를 만들어라

|  |  |
| --- | --- |
| 목적 | 구현 세부 내용 |
| 템플릿 큐 | 큐에 템플릿 기능을 적용하여 추론 기능을 만들어라 |

문제 분석

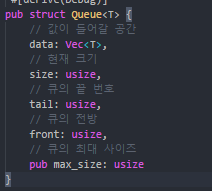
|  |  |
| --- | --- |
| 이름 | |
| C++ 템플릿 큐 | Rust 템플릿 큐 |
| 분석 | |
| 자료형에 자유로운 큐를 만들고  Int, float, string 형을 응용하라 | 자료형에 자유로운 큐를 만들고  Int, float, string 형을 응용하라 |

본 문서에선 두 언어에서 구현된 Queue 를 토대로

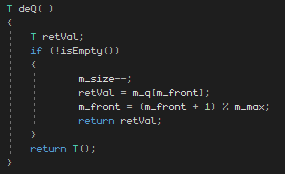
속도 비교를 위해 시간 측정 코드가 있음

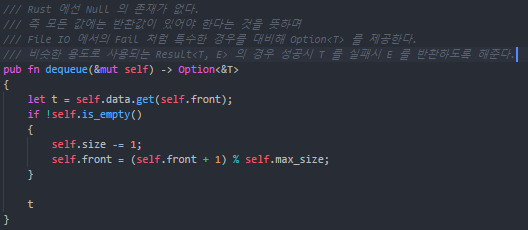
두 큐 모두 링 구조를 가지고 있다.

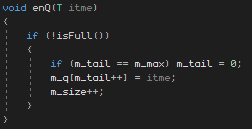
주요 코드

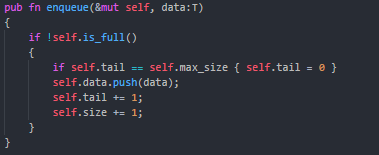


큐의 구조체







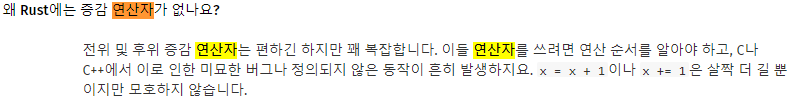


러스트에는 모종의 이유로 전위연산 후위연산의 개념이 없다.

러스트 문서에서는 해당 연산자가 프로그래머의 머리를 어지럽게 만들고

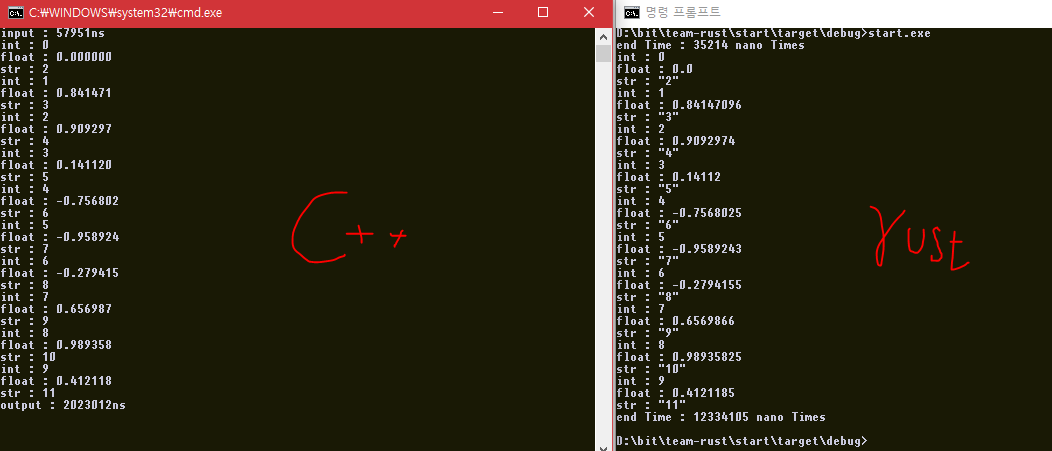
생각을 불편하게 만들어 예기치 못한 버그를 만들 수 있다고 판단하여 만들지 않았다고

적혀있다.



실행 결과

해당 장에서는 C++ 과 Rust 간의 속도에 대하여 알아보겠음



둘의 비교 결과가 상이한 것의 원인을 분석해보겠다.

C++ 의 경우 대입 연산자가 이동이 아닌 복사의 개념으로 동작한다는 것을 인지하여야 한다.

이는 버퍼를 복사하는 대신 이전한다는 개념에 더 가까우며

Rust 의 경우 소유권 이라는 개념을 사용하여 이전이 아닌 소유권의 이동으로 써 값은 항시

그 자리에 남아있어 추가적인 연산이 적어 OS 단에서의 오버헤드가 적어지는 이점이 있다.

때문에 값을 집어넣는 Enqueue 시 Rust 쪽에서 조금 더 빠르게 움직인다.

다음은 출력 시간을 비교할 것 인데

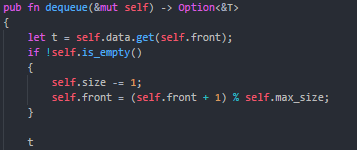
출력시간의 경우 C++ 이 더 빠르다.

이는 값이 배열에서 빠져나가는 과정을 보면 이해가 되는데

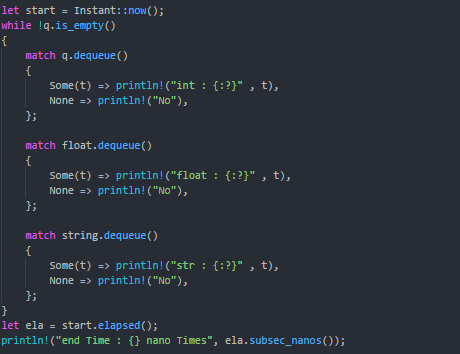
Rust 에서 오버헤드가 적은 비용으로 enqueue 를 진행했으니

당연히 dequeue 도 빨라야 하는 것 이라고 생각 할 수 있다.

물론 dequeue 에서 , 그러니까



함수 자체는 바르게 돌지만 이를 출력하는 과정인



해당 코드의 비용이 비싼 것 이다.

본인의 경우 해당 코드를 더 간단히 쓸 수 있었지만 Rust 가 지향하는

방식을 따랐다.

Error Log

.

결론 & 후기

결론

템플릿 큐를 구현하고 수행 시간을 알아내어 C++ 과 Rust 의 성능 비교를 해보았다.

후기

둘의 속도를 비교 한다는 게 마치 어릴 때 RC카 가지고 누구 차가 더 빠른지 내기 하는 느낌이라 재미있는 것 같다. 특히 Rust 가 기존에 배워오던 언어들과 너무 다른 지향점을 가지고 만들어져 더욱 흥미롭게 다가온다. 방금 위에서 본 enqueue 의 결과만 봐도 ..