STL 과제 1

학번 : 2014180026

이름 : 원동욱

목차

1. 과제 해결 방법
   1. 친구들과 토론
   2. 코딩으로 구현
   3. 교수님의 수업내용 적용 후 코드 정리
2. 요구 사항 중 구현하지 못한 내용과 어려웠던 점
   1. 구현하지 못한 내용
   2. 어려웠던 점
      1. 자료들을 어떻게 저장을 할까?
      2. Sort()으로 인한 프로그램의 속도 저하
      3. 정규 분포, 파일 입출력에 대한 문제
      4. 코드를 해석, 수정에 대한 문제
      5. 자잘한 실수
3. 과제를 하면서 느낀 점
   1. 구현 방식의 차이
   2. 저장 방법의 중요성
   3. 프로그램 만들 때의 마음 가짐
   4. 깃의 중요성

[ 과제 해결방법 ]

1. 친구들과 토론

기본적으로 자료형의 이름을 통일하는 것부터 시작했다. 그리고 게임을 표현하는 방법에 대해 애기 하였는데 플레이어를 나타낼 수 있는 class와 그 class를 이용해 게임의 기능을 구현하는 class로 게임을 표현 하기로 했다.

그 다음 문제로 거론 된 것은 플레이어 십만명을 어떻게 저장 하는가 였다. 가장 쉬운 배열 이야기가 먼저 나왔었지만 스택이 버틸 수 없다는 우려에 벡터를 쓰기로 결론을 지었다.

플레이어를 저장을 합의를 하고 나니 랭킹을 어떻게 할 지 이야기가 나왔는데 크게 3가지 의견이 나왔었다. 매번 소트를 돌리자는 의견은 연산이 너무 많이 일어 날 거 같다는 의견이 나와서 바로 취소 되었고 쿠키 훈련소 점수로 정렬된 벡터와 떼탈출 점수로 정렬된 벡터를 동시에 보유하는 방안과 인덱스만 기억해서 찾아가는 것 중 구현 되는 걸로 하기로 하였다.

마지막 문제로는 플레이어 1000명을 어떻게 중복 없이 뽑아내는지 대한 문제 였는데 해석을 중복을 허용하고 10000번만 플레이를 하는 것으로 해서 그냥 반복 10000 의견과 플레이한 배열을 가지고 그만큼 검사를 하는 방법, 플레이어가 자신이 플레이 된 지를 알고 있어서 플레이가 되지 않았을 경우에만 플레이하도록 하는 방법이 거론 되었는데 시간이 늦어서 결론은 내리지 못하고 끝났다.

1. 혼자 1차적인 코딩을 해봄

인덱스 값으로 랭크를 찾아간다는 방식으로 먼저 구현을 했지만 소트를 할 때마다 저장한 배열의 위치들이 바뀌고 이로 인해 원하던 값이 나오지 않는다는 걸 깨닫고. 이를 해결 하기 위해 데이터를 가리키는 포인터들을 저장하는 것으로 저장 방법을 바꾸었다

플레이 할 때 마다 소트를 쓰면 플레이 할 때 조금 시간 걸리는 것이 마음에 안 들어서 쿠키 훈련소 탈출/때탈출로 정렬된 포인터 벡터를 만들어서 소트가 조금이라도 빠르게 돌아가도록 하였다.

중간에 랭크 표시를 쉽게 하기 위해서 플레이어에 랭크를 추가하고 플레이어 만명을 중복없이 선택하기 위해서 플레이 여부를 확인하는 변수도 추가되었다. 하지만 정규 분포를 사용 할 때 일어나는 오류를 잡지 못하여 균일 분포로 설정하였고 파일 입출력의 사용방법을 몰라서 적용시키지 못하였다.

1. 수업내용에서 배운 내용들을 추가 및 코드 정리

수업에서 벡터를 파일 입출력 하는 방법과 랜덤에 대한 설명을 듣고 이를 프로그램에 적용시켰고 적용시키려고 보니 프로그램을 해석하고 수정하기 힘들어서 전체적으로 프로그램을 정리하고 적용시키고 과제를 마쳤다.

[ 요구사항 중 구현하지 못한 내용과 어려웠던 점 ]

1. 구현 못한 점 : 없음
2. 어려웠던 점
   1. 자료형들을 어떻게 저장할까?

처음에 자료형을 어떻게 저장 할 지를 몰라서 많이 고민했다. 게임을 하면서도 고민하고 친구네에 인쇄하러 가면서도 고민하고 고민하다 포인터를 사용하면 해볼까? 라는 생각이 떠올랐고 처음에는 unique\_ptr로 구현했었는데 복사도 안되고 값의 공유도 안되어서 shared\_ptr로 바꿔서 프로그램을 구현했더니 원하는 대로 구현이 되어서 사용하였다. 하지만 shared\_ptr에 적응하는 것도 쉽지는 않았다.

* 1. Sort()으로 인한 프로그램의 속도 저하

처음에 & 안 쓰고 프로그램을 만들었었는데 생성자 만드는데 3초가 넘어가는 거 보고 놀라서 어디가 문제인지 찾다가 &를 쓰지 않은 것을 깨닫고 수정하니 생성 시간이 절반으로 감소하였다. 나중에 릴리즈 모드를 사용하는 걸 적용시키니 더 빨라졌다.

* 1. 정규 분포,랜덤, 파일 분포에 대한 문제

정규 분포는 int값 인자를 넣을 수 없다는 걸 모르고 계속 int 넣어서 오류가 나서 설정을 못했었고 랜덤 값이 계속 똑같은 순서대로 출력되고 파일을 어떻게 입출력 해야 할지 몰라서 찾아봤었는데 못 찾아서 너무 답답했었다.

* 1. 코드를 해석, 수정 할 수가 없음.

코드가 일정 수가 넘어가니 수정하는 위치를 찾기에도 한참이고 정리와 주석이 없으니 어제 했던 코드를 어떻게 했는지도 기억이 안 나서 결국 코드를 쭉 읽으면서 정리했다.

* 1. 자잘한 실수

갑자기 소트가 값을 변경하지 않아서 온갖 변수를 고쳐가면서 찾았었는데 람다함수에서 비교 문이 p1 < p1 이런 식 이였다. 이런 자잘하게 실수 경우가 컴파일러 오류도 발생하지 않고 한 글자 실수 인 경우가 다반사라 찾기도 어려웠다.

[ 과제를 하면서 느낀 점 ]

1. 구현 방식의 차이

친구들과 토론하면서 할 때 3명~4명이서 했는데 애들이 구현하는 방법이 다 달라서 놀라게 되었고 여러 사람이 모여서 토론하니 내가 생각 하지 못한 부분을 볼 수 있는 것을 알 수 있었다.

2. 저장 방식의 중요성

처음에 자료들을 어떻게 저장 할 지에 대한 문제가 정하는게 정말 큰 일 이였다. 처음에 저장을 잘못 설정하니 코드를 만드는 진도가 정말 1도 나가지 않았었다. 반면 정확하게 저장 방식을 설정하니 그 다음부터는 속도가 났었다.

3. 프로그램 만들 때의 마음 가짐

급하게 코드를 짜려고 하면 정말 죽도 밥도 안된다는 걸 느낄 수 있었다. 처음에 클레스를 만들 때 아무 생각 없이 실행 안 해 가면서 만들었다가 결국 프로그램이 죽어버리고 처음부터 조금씩 다시 만들어서 간신히 프로그램이 돌아가게 하였다. 한번에 실력보다 너무 많이 만들면 고생한다는 걸 느끼게 되었다.

프로그램을 만들 때 적당히 휴식이 필요하다는 걸 느꼈다. 계속 컴퓨터 화면만 보고 있어도 원하는 코드가 만들어지지 않는다는 걸 느꼈다. 오히려 잠깐 밖에 걸어 다닐 때 해결법을 많이 얻을 수 있었다.

1. 깃의 중요성

헝가리안 언어를 수정 했을 뿐인데 프로그램이 죽어버리고 alt + z 해도 복구되지 않아 정신 붕괴 할 뻔 했지만 깃이 2번 복구 시켜서 살았다. 깃 최고…

1. 복사는 비싸다

옛날에 const &를 왜 쓰는지 궁금 하였는데 이번에 sort를 돌리면서 왜 쓰는 알았다.. 복사는 너무 비싼 연산이다. 복사보단 move가 move 보다 emplace가.. 수가 쌓이면 부담이 너무 커진다.

1. 알고리즘의 중요성
   1. 알고리즘에 따라 프로그램이 얼마나 빨라 질 수 있는지 직접 눈으로 보고 체감 할 수 있었었다. 그리고 좋은 알고리즘 만드는 건 진짜 힘들다는 것도 느꼈다