

C++ 프로그래밍 및 실습

TFT Guide

최종 보고서

제출일자:2024/12/22

제출자명:김현규

제출자학번:214014

1. 프로젝트 목표 (16 pt)

1) 배경 및 필요성 (14 pt)

이 프로젝트를 시작하게 된 가장 큰 이유는 제가 좋아하는 게임인 TFT 때문입니다. 프로젝트 이름인 'TFT Guide'도 이에서 유래했습니다. 간단히 말하자면, TFT는 "리그 오브 레전드"의 IP(Intellectual Property)를 활용한 체스류 오토배틀러 게임으로, 전략 시뮬레이션 장르에 속합니다. 이 게임은 다양한 챔피언들을 이용해 팀을 구성하고, 자동으로 전투를 진행하면서 최적의 전략을 찾아내는 것이 핵심입니다. 사용자들은 각기 다른 챔피언 조합, 아이템 선택, 그리고 전투 대형 등을 통해 게임에서 승리를 거두기 위해 전략적 사고를 요구받습니다.

TFT는 게임의 깊이와 다양성으로 인해 많은 사용자들에게 사랑받고 있지만, 그만큼 진입 장벽이 높은 게임이기도 합니다. 특히 초보자들이 게임의 기본 전략을 제대로 이해하고 따르기 어려운 경우가 많습니다. 전략적인 요소가 많다 보니 어떤 챔피언을 선택해야 할지, 어떤 아이템을 사용할지, 그리고 어떤 조합이 가장 효과적인지에 대한 고민이 끊이지 않습니다.

이 프로젝트의 필요성은 바로 이러한 초보자들이 게임을 더 쉽게 이해하고 즐길 수 있도록 돕기 위함입니다. TFT Guide 프로그램은 초보자들이 게임의 전략적 요소를 더 쉽게 이해하고, 아이템 선택 및 조합 구성에서 더 나은 선택을 할 수 있도록 돕는 것을 목표로 합니다. 이를 통해 사용자들이 자신의 실력을 향상시키고 게임에서 더 높은 승률을 기록할 수 있도록 지원하고자 합니다. 예를 들어, 특정 상황에서 어떤 아이템이 가장 효과적인지, 어떤 챔피언 조합이 현재 강력한지 등의 정보를 제공함으로써 사용자들이 보다 효율적인 전략을 구사할 수 있게 돕습니다.

또한, 이 프로그램은 단순히 초보자들만을 위한 것이 아니라, 중급자나 고급자들에게도 유용한 도구가 될 수 있습니다. TFT는 주기적인 패치로 인해 최적의 전략도 달라지기 마련입니다. TFT Guide는 최신 패치를 반영한 정보를 제공함으로써 모든 레벨의 사용자들이 항상 최적의 전략을 구사할 수 있도록 도와줍니다. 이를 통해 사용자는 게임의 변화에 신속하게 적응하고, 더 나은 결과를 얻을 수 있게 됩니다.

결론적으로, TFT Guide는 TFT에서 실력 향상을 원하는 모든 사용자들에게 필요한 프로그램입니다. 초보자들에게는 게임의 기본적인 전략을 이해하고 적용하는 데 도움을 주며, 중급자나 고급자들에게는 최신 메타에 맞춘 최적의 전략을 제시함으로써 게임의 재미를 극대화할 수 있도록 지원합니다. 이러한 프로그램을 통해 사용자들이 게임을 보다 깊이 이해하고, 더 큰 만족감을 느끼며 즐길 수 있기를 기대합니다.

2) 프로젝트 목표

이 프로젝트의 목표는 사용자가 TFT에서 각 상황에 맞는 최적의 전략을 쉽게 선택할 수 있도록 돕는 프로그램을 개발하는 것입니다. TFT에는 다양한 요소들이 존재합니다. 챔피언의 스킬, 시너지, 아이템, 증강체 등 승률을 높이는 데 필요한 많은 요소들이 게임 내에서 중요한 역할을 합니다. 이 프로그램은 어떤 챔피언에게 어떤 아이템을 장착해야 가장 승률이 좋은지, 현재 시너지에 맞는 최적의 증강체가 무엇인지 등을 분석하고 추천함으로써 사용자에게 유용한 정보를 제공합니다.

구체적으로, 사용자가 게임 도중 어려움을 느낄 때 어떤 선택이 가장 유리한지를 명확하게 안내하고, 이를 통해 승률을 높일 수 있는 기회를 제공합니다. 예를 들어, 특정 라운드에서 어떤 아이템이 가장 효과적인지, 어떤 챔피언 조합이 메타에서 강력한지 등에 대한 정보를 실시간으로 제공하여 사용자들이 더 나은 결정을 할 수 있도록 돕습니다.

이 프로그램의 최종 목표는 모든 레벨의 사용자들이 게임의 복잡한 전략 요소를 보다 쉽게 이해하고 적용함으로써, 게임을 더 재미있게 즐길 수 있도록 돕는 것입니다. 초보자에게는 기본적인 전략 이해를 돕고, 중급자나 고급자에게는 최신 메타에 맞는 전략을 제공하여 게임에서 더 큰 성취감을 느낄 수 있도록 지원하는 것을 목표로 합니다.

3) 차별점

기존에 TFT를 즐기는 사람들이 참고하는 사이트는 주로 '롤체지지'라고 불리는 사이트가 있습니다. 이곳에서는 한 조합에 대해 하나의 구성만 추천해주고 있습니다. 또한 그 조합에 맞는 아이템과 증강체를 추천해주고 있지만, 어느 상황에서 어떤 선택이 좋은지에 대한 구체적인 정보는 부족합니다. 예를 들어, 각 조합의 승률, 사람들이 얼마나 기용하는지, 그리고 어떤 상황에서 가장 효과적인지에 대한 정보가 충분하지 않습니다.

TFT Guide는 기존 사이트와 차별화된 기능을 제공합니다. 저는 추천 조합에 따른 아이템과 증강체는 물론이고, 픽률과 승률, 추천 배치도, 그리고 어느 타이밍에 어떤 조합으로 이끌어 나가야 하는지에 대한 구체적인 정보를 제공하여 사용자가 더 나은 결정을 내릴 수 있도록 돕고자 합니다. 예를 들어, 특정 타이밍에 어떤 아이템을 장착하고 어떤 조합을 완성해야 하는지에 대한 세부적인 정보와 이에 따른 승률 및 배치도까지 모두 추천해줄 계획입니다. 이를 통해 사용자들이 게임에서 더 많은 전략적 선택을 할 수 있도록 돕고, 게임을 더욱 재미있게 즐길 수 있도록 지원하는 것이 목표입니다.

4) 사용 설명서

```
덱 추천 프로그램입니다. 옵션을 선택하세요:
1. 최고 승률 덱 추천
2. 모든 덱 정보 보기
3. 챔피언 기반 덱 추천
4. 아이템 추천
5. 종료
선택: █
```

옵션 선택은 숫자 1 ~ 5를 입력하면 됩니다. 1 ~ 2번의 기능은 숫자만 입력하면 문제 없이 수행되지만, 챔피언과 아이템은 오타가 발생하면 기능이 제대로 수행되지 않습니다.

```
추천받고 싶은 챔피언 이름을 입력하세요: Ryze
챔피언 Ryze과(와) 관련된 덱:
덱 이름: 차원문 덱
승률: 15.3%
평균 등수: 4.31
Top 4 비율: 52.5%
사용 비율: 0.3
티어: C
구성 챔피언: Ryze / Taric / TahmKench
추천 아이템: staff / mana essence / chain vest / magic cloak
-----
```

챔피언의 종류는 10개로 이루어져 있습니다.

챔피언 : "Ryze", "Taric", "TahmKench", "Gwen", "Fiora", "Rakan", "Varus", "Xerath", "Kalista", "Milio"

챔피언은 작성된 것과 동일하게 입력하지 않으면 실행되지 않습니다.

```
추천받고 싶은 아이템 이름을 입력하세요: staff
아이템 staff에 맞는 챔피언:
Ryze
Gwen
Xerath
```

아이템은 6종류로 이루어져 있습니다.

아이템 : "staff", "mana essence", "chain vest", "giant belt", "sword", "bow"

아이템도 챔피언처럼 작성된 것과 동일하게 입력하지 않으면 실행되지 않습니다.

2. 기능 계획

1) 최고 승률의 덱 추천 기능

- 현재 가장 승률과 평균 순위가 높은 덱을 순서대로 추천해주는 기능.

(1) 상위 티어 게이머들의 데이터 수집

- 상위 티어 게이머들의 게임 데이터를 수집하여 어떤 덱이 승률, 평균 순위 그리고 사용 횟수 등을 제공.

(2) 덱 추천도 기능

- 수집된 데이터를 이용하여 각 덱의 성능에 따라 S, A, B, C, D 랭크로 제공.

2) 시각적 배치도 추천 기능

- 사용자가 게임 내에서 챔피언들을 어떻게 배치해야 될지에 대한 시각적인 배치도를 제공.

(1) 상황별 배치도 제공

- 게임 라운드에 따라 가장 강한 덱 구성과 구성된 챔피언에 특성에 맞는 배치도를 추천해줌.

3) 개인화된 전략 피드백 기능

- 사용자의 과거 게임 데이터를 분석하여 반복적인 실수를 파악하고, 이를 줄이기 위한 전략적 피드백을 제공

(1) 반복적인 실수 분석

- 사용자가 게임에서 자주 저지르는 실수를 분석하여 피드백을 제공.

(2) 유사 취향의 상위 플레이어의 전략 추천 기능

- 사용자와 유사한 전략을 지닌 상위 플레이어들과 사용자의 다른 점을 비교하여 피드백해주고, 상위 랭크 플레이어의 전략을 추천

4) 최고 승률의 증강체 추천 기능

- TFT는 2-1, 3-2, 4-2 라운드에 증강체라는 특수 기능을 선택할 수 있는데 랜덤하게 나오는 3가지 중에 한개를 골라야 함. 그 3가지 중에서 가장 높은 승률을 가지는 증강체와 그에 잘 어울리는 덱을 추천해주는 기능.

(1) 증강체의 승률 분석

- 각 증강체의 승률, 평균 순위 등을 분석하여 각 증강체에 랭크를 정하여 추천해줌.

(2) 증강에 어울리는 덱 추천

- 증강체는 공격력을 올려주거나, 주문력을 올려주는 등 각 덱에 따라 어울리는 증강체가 있고, 아닌 경우가 있음. 그렇기에 증강체에 따라 승률이 더 올라가는 덱들이 존재함. 그러한 추천 덱을 제공.

5) 아이템 추천 기능

- 챔피언마다 스킬이 다르기 때문에 그에 맞는 아이템도 각자 다름. 이를 가장 승률이 좋은 아이템으로 추천해주는 기능.

(1) 아이템에 걸맞는 챔피언 추천 기능

- 각 아이템을 사용했을 때 가장 높은 승률과 좋은 순위를 갖는 챔피언을 추천.

(2) 챔피언에 걸맞는 아이템 추천 기능

- 각 챔피언에 대해서 가장 높은 승률과 좋은 순위를 기록한 아이템을 추천.

3. 진척사항

1) 기능 구현

(1) 사용자 입력 처리 : 사용자에게 1 ~ 5번을 입력받아 각각 "최고 승률 덱 추천", "모든 덱 정보 보기", "챔피언 기반 덱 추천", "아이템 추천", "종료"를 수행합니다.

- 입출력

- choice : 사용자가 선택한 메뉴 옵션(정수)입니다.

- sub_choice : 아이템 추천과 관련된 서브메뉴에서 선택한 옵션(정수)입니다.

- 설명

- 사용자는 반복문에서 계속 옵션을 선택하며, 종료를 선택할 때까지 반복됩니다.

- 사용자가 선택할 수 있는 메뉴는 1 ~ 5번까지 존재합니다.

1. 최고 승률 덱 추천 : RecommendDeck(decks) 함수를 실행합니다.

2. 모든 덱 정보 보기 : ViewDeckDetails(decks) 함수를 실행합니다.

3. 챔피언 기반 덱 추천 : RecommendDeckByChampion(decks) 함수를 실행합니다.

4. 아이템 추천 : sub_choice에 따라 RecommendItemsByChampion(championItems) 또는 RecommendChampionsByItem(itemChampions) 함수를 실행합니다.

5. 덱에 맞는 증강체 추천 : RecommendAugmentsForDeck(decks, decks_augment) 함수를 실행합니다.

6. 종료 : 프로그램을 종료합니다.

- 잘못된 입력에 대한 처리를 포함하여, 사용자가 올바른 입력을 할 때까지 반복되도록 설계 되었습니다.

- 적용된 배운 내용

- 조건문인 if문을 사용했습니다.

- 코드 스크린샷

```
239     int choice;
240     while (true) {
241         cout << "덱 추천 프로그램입니다. 옵션을 선택하세요:\n";
242         cout << "1. 최고 승률 덱 추천\n";
243         cout << "2. 모든 덱 정보 보기\n";
244         cout << "3. 챔피언 기반 덱 추천\n";
245         cout << "4. 아이템 추천\n";
246         cout << "5. 덱에 맞는 중강체 추천\n";
247         cout << "6. 종료\n";
248         cout << "선택: ";
249         cin >> choice;
250
251         if (choice == 1) {
252             RecommendDeck(decks);
253         } else if (choice == 2) {
254             ViewDeckDetails(decks);
255         } else if (choice == 3) {
256             RecommendDeckByChampion(decks);
257         } else if (choice == 4) {
258             int sub_choice;
259             cout << "1. 챔피언에 맞는 아이템 추천\n";
260             cout << "2. 아이템에 맞는 챔피언 추천\n";
261             cout << "선택: ";
262             cin >> sub_choice;
263
264             if (sub_choice == 1) {
265                 unordered_map<string, vector<string>> champion_items = {
266                     {"Ryze", {"staff", "mana essence"}},
267                     {"Taric", {"chain vest", "magic cloak"}},
268                     {"TahmKench", {"giant belt"}},
269                     {"Gwen", {"staff"}},
270                     {"Fiora", {"sword"}},
271                     {"Rakan", {"giant belt"}},
272                     {"Varus", {"sword", "mana essence"}},
273                     {"Xerath", {"staff", "mana essence"}},
274                     {"Kalista", {"bow"}},
275                     {"Milio", {"mana essence"}}
276                 };
277                 RecommendItemsByChampion(champion_items);
278             } else if (sub_choice == 2) {
279                 unordered_map<string, vector<string>> item_champions = {
280                     {"staff", {"Ryze", "Gwen", "Xerath"}},
281                     {"mana essence", {"Ryze", "Varus", "Xerath", "Milio"}},
282                     {"chain vest", {"Taric"}},
283                     {"giant belt", {"TahmKench", "Rakan"}},
284                     {"sword", {"Fiora", "Varus"}},
285                     {"bow", {"Kalista"}}
286                 };
287                 RecommendChampionsByItem(item_champions);
288             } else {
289                 cout << "잘못된 입력입니다. 다시 시도하세요." << endl;
290             }
291         } else if (choice == 5) {
292             int deck_choice;
293             cout << "중강체를 추천받고 싶은 덱을 선택하세요:\n";
294             for (size_t i = 0; i < decks.size(); ++i) {
295                 cout << i + 1 << ". " << decks[i].GetName() << "\n";
296             }
297             cout << "선택: ";
298             cin >> deck_choice;
299
300             if (deck_choice >= 1 && deck_choice <= decks.size()) {
301                 RecommendAugmentsForDeck(decks[deck_choice - 1], deck_augments);
302             } else {
303                 cout << "잘못된 입력입니다. 다시 시도하세요." << endl;
304             }
305         } else if (choice == 6) {
306             cout << "프로그램을 종료합니다." << endl;
307             break;
308         } else {
309             cout << "잘못된 입력입니다. 다시 시도하세요." << endl;
310         }
311     }
312
313     return 0;
314 }
```

(2) 최고 승률 덱 추천 : 현재 등록된 덱들의 승률의 덱을 추천합니다. 또한 덱의 이름, 구성 챔피언, 추천 아이템, 승률 등 세부 정보를 출력합니다.

- 입출력

- 입력: `vector<Deck>& decks`— 등록된 덱들의 정보를 담고 있는 벡터.
- 출력: 콘솔에 추천 덱의 정보가 출력됩니다. 여기에는 덱의 이름, 승률, 평균 등수, Top 4 비율, 구성 챔피언, 추천 아이템 등이 포함됩니다.

- 설명

- 이 함수는 주어진 덱 리스트에서 가장 높은 점수를 가진 덱을 추천합니다.
- 빈 덱 리스트 처리: 만약 덱 리스트가 비어 있는 경우, "덱 정보가 없습니다."라고 출력하고 함수를 종료합니다.
- 정렬(sort): 덱 리스트를 점수를 기준으로 내림차순 정렬한 후, 승률이 가장 높은 덱을 추천합니다.
- 세부 정보 출력: 정렬된 첫 번째 덱의 세부 정보를 출력하여 사용자가 해당 덱에 대한 모든 주요 정보를 알 수 있도록 합니다.

- 적용된 배운 내용

- Deck 클래스의 멤버 함수 `GetWinRate()`와 `PrintDeckInfo()`를 호출하였습니다.

- 코드 스크린샷

```
139 // 최고 승률 덱 추천 기능 구현 함수 정의
140 void RecommendDeck(vector<Deck>& decks) {
141     if (decks.empty()) {
142         cout << "덱 정보가 없습니다." << endl;
143         return;
144     }
145     sort(decks.begin(), decks.end(), [](const Deck& a, const Deck& b) {
146         return a.GetScore() > b.GetScore();
147     });
148     cout << "추천 덱 (승률 기준):\n";
149     decks[0].PrintDeckInfo();
150 }
151
```

(3) 덱 정보 조회 : 사용자에게 모든 덱의 이름을 출력해주고, 세부정보를 조회하고 싶은 덱의 번호를 입력받아 덱을 출력합니다.

- 입출력

- 입력:const vector<Deck>& decks— 등록된 덱들의 정보를 담고 있는 벡터.
- 출력:콘솔에 모든 덱의 세부 정보가 출력됩니다. 여기에는 덱의 이름, 승률, 구성 챔피언, 추천 아이템 등이 포함됩니다.

- 설명

- 함수는 등록된 덱이 없는 경우를 먼저 확인하고, 덱이 없을 경우 "덱 정보가 없습니다."라는 메시지를 출력한 후 함수를 종료합니다.
- 만약 덱이 있다면, 등록된 모든 덱의 정보를 출력합니다. 각 덱의 정보는 PrintDeckInfo()함수를 사용해 출력됩니다.
- 이를 통해 사용자는 전체 등록된 덱의 구성, 승률, 추천 아이템 등 다양한 정보를 알 수 있습니다.

- 적용된 배운 내용

- 조건문(if)와 반복문(for)을 사용하고, Deck 클래스에서 PrintDeckInfo()를 호출했습니다.

- 코드 스크린샷

```
123 // 모든 덱 정보 보기 기능 구현 함수 정의
124 void ViewDeckDetails(const vector<Deck>& decks) {
125     if (decks.empty()) {
126         cout << "덱 정보가 없습니다." << endl;
127         return;
128     }
129     for (const auto& deck : decks) {
130         deck.PrintDeckInfo();
131     }
132 }
```

(4) 챔피언 기반 덱 추천 : 사용자가 특정 챔피언 이름을 입력하면 해당 챔피언이 포함된 덱을 검색하여 사용자에게 덱을 추천해주고, 덱에 대한 세부정보를 출력합니다.

- 입출력

입력:

- 사용자로부터 특정 챔피언 이름(champion)을 입력받습니다.
- 이 입력은 덱 목록에서 특정 챔피언을 포함한 덱을 찾기 위해 사용됩니다.

출력:

- 해당 챔피언이 포함된 덱의 정보를 출력합니다.
- 덱의 이름, 승률, 구성된 챔피언들, 추천 아이템 등이 포함된 상세 정보가 콘솔에 표시됩니다.

- 설명

- 프로그램은 사용자가 입력한 챔피언 이름을 기준으로 덱 목록에서 해당 챔피언을 포함한 덱을 찾습니다.
- 없는 경우 처리: 만약 해당 챔피언이 포함된 덱을 찾지 못할 경우, "해당 챔피언을 포함한 덱 정보가 없습니다."라는 메시지를 출력합니다.
- 찾는 경우 처리: 덱을 찾으면 PrintDeckInfo()를 호출하여 덱의 세부 정보를 출력합니다.

- 적용된 배운 내용

- 조건문, 반복문, 클래스에서 함수 호출을 사용했습니다.

- 코드 스크린샷

```
134 // 챔피언 기반 덱 추천 가능 구현 함수 정의
135 void RecommendDeckByChampion(const vector<Deck>& decks) {
136     string champion;
137     cout << "추천받고 싶은 챔피언 이름을 입력하세요: ";
138     cin.ignore();
139     getline(cin, champion);
140
141     bool found = false;
142     for (const auto& deck : decks) {
143         if (find(deck.GetChampions().begin(), deck.GetChampions().end(), champion) != deck.GetChampions().end()) {
144             deck.PrintDeckInfo();
145             found = true;
146         }
147     }
148     if (!found) {
149         cout << "해당 챔피언을 포함한 덱 정보가 없습니다." << endl;
150     }
151 }
152
```

(5) 아이템 추천 : 사용자에게 아이템 추천을 위한 두 가지 선택지를 제공하는데, 챔피언에 맞는 아이템을 추천 받는 것과 아이템에 걸맞는 챔피언을 추천받는 것 중 사용자에게 입력을 받아 아이템이나 챔피언을 추천합니다.

- 입출력

입력:

- 챔피언 이름 (champion): 특정 챔피언에 맞는 아이템을 추천받고자 할 때 사용됩니다.
- 아이템 이름 (item): 특정 아이템에 적합한 챔피언을 추천받고자 할 때 사용됩니다.

출력:

- 아이템 추천: 사용자 입력에 따라 특정 챔피언에 맞는 추천 아이템 목록을 출력합니다.
- 챔피언 추천: 입력된 특정 아이템에 적합한 챔피언을 추천합니다.

- 설명

- 사용자는 먼저 챔피언 기반 아이템 추천 또는 아이템 기반 챔피언 추천 중 하나를 선택합니다.
- 챔피언 기반 아이템 추천 시, 챔피언 이름을 입력받고 그에 맞는 아이템 목록을 출력합니다.
- 반대로 아이템 기반 챔피언 추천 시, 아이템 이름을 입력받고 그 아이템에 적합한 챔피언들을 출력합니다.
- 각각의 선택지에 대해 해당 정보를 찾지 못할 경우, "해당 정보가 없습니다."라는 메시지를 출력합니다.

- 코드 스크린샷

```
153 // 챔피언에 맞는 아이템 추천 기능 구현 함수 정의
154 void RecommendItemsByChampion(const unordered_map<string, vector<string>>& champion_items) {
155     string champion;
156     cout << "추천받고 싶은 챔피언 이름을 입력하세요: ";
157     cin.ignore();
158     getline(cin, champion);
159
160     auto it = champion_items.find(champion);
161     if (it != champion_items.end()) {
162         cout << "챔피언 " << champion << "에 추천되는 아이템:\n";
163         for (const auto& item : it->second) {
164             cout << item << "\n";
165         }
166     } else {
167         cout << "해당 챔피언에 대한 아이템 정보가 없습니다." << endl;
168     }
169     cout << "-----\n";
170 }
171
172 // 아이템에 맞는 챔피언 추천 기능 구현 함수 정의
173 void RecommendChampionsByItem(const unordered_map<string, vector<string>>& item_champions) {
174     string item;
175     cout << "추천받고 싶은 아이템 이름을 입력하세요: ";
176     cin.ignore();
177     getline(cin, item);
178
179     auto it = item_champions.find(item);
180     if (it != item_champions.end()) {
181         cout << "아이템 " << item << "에 맞는 챔피언:\n";
182         for (const auto& champion : it->second) {
183             cout << champion << "\n";
184         }
185     } else {
186         cout << "해당 아이템에 대한 챔피언 정보가 없습니다." << endl;
187     }
188     cout << "-----\n";
189 }
```

(6) 증강체 추천 : 이 기능은 사용자에게 덱에 적합한 증강체를 추천하는 기능입니다. 사용자가 선택한 덱에 대해, 해당 덱에 어울리는 증강체의 목록을 출력합니다.

- 입출력

입력: const Deck& deck, const unordered_map<string, vector<string>>& deck_augments

- 덱(deck): 사용자로부터 선택된 덱.
- 덱과 관련된 증강체 정보를 저장한 unordered_map자료구조(deck_augments).

출력:

- 덱 이름과 함께 해당 덱에 추천되는 증강체들을 출력합니다.
- 만약 해당 덱에 대한 증강체 정보가 없을 경우, 적절한 메시지를 출력합니다.

- 설명

- 이 함수는 특정 덱을 입력받아 그 덱에 적합한 증강체의 목록을 출력합니다.
- 증강체 존재 확인: 먼저 unordered_map에서 해당 덱의 이름을 찾아 증강체 정보를 가져옵니다.
- 출력: 해당 덱의 증강체 정보가 존재하면, 리스트로 출력하며, 그렇지 않으면 "해당 덱에 대한 증강체 정보가 없습니다."라고 출력합니다

- 코드 스크린샷

```
191 // 덱에 어울리는 증강체 추천 기능 구현 함수 정의
192 void RecommendAugmentsForDeck(const Deck& deck, const unordered_map<string, vector<string>>& deck_augments) {
193     auto it = deck_augments.find(deck.GetName());
194     if (it != deck_augments.end()) {
195         cout << "덱 " << deck.GetName() << "에 추천되는 증강체:\n";
196         for (const auto& augment : it->second) {
197             cout << augment << "\n";
198         }
199     } else {
200         cout << "해당 덱에 대한 증강체 정보가 없습니다." << endl;
201     }
202     cout << "-----\n";
203 }
204
```

(7) 클래스 :

item 클래스

- 입출력

- 입력: 아이템 이름(name)과 유형(type)을 받아 객체를 생성합니다.
- 출력: PrintItemInfo() 함수를 통해 아이템의 이름과 유형을 출력합니다.

- 설명

멤버 변수:

- name: 아이템의 이름을 저장.
- type: 아이템의 유형을 저장.
- 생성자: Item(string n, string t)를 통해 아이템의 이름과 유형을 초기화합니다.

멤버 함수:

- GetName() 및 GetType(): 각각 아이템의 이름과 유형을 반환합니다.
- PrintItemInfo(): 아이템의 이름과 유형을 콘솔에 출력합니다.

Deck 클래스

- 입출력

- 입력: 덱 이름(name), 승률(win_rate), 평균 등수(avg_placement), Top 4 비율(top4_rate), 사용 비율(usage_count), 구성 챔피언 목록(champions)을 받아 객체를 생성합니다.
- 출력: PrintDeckInfo()함수를 통해 덱의 세부 정보를 출력합니다. 여기에는 덱의 이름, 승률, 평균 등수, 챔피언 구성, 추천 아이템 등이 포함됩니다.

- 설명

멤버 변수:

- name, win_rate, avg_placement, top4_rate, usage_count, champions, recommended_items등 덱에 관한 다양한 정보를 저장합니다.

생성자:

- Deck(string n, double wr, double ap, double t4r, double uc, vector<string> champs)를 통해 덱의 주요 정보를 초기화합니다.

멤버 함수:

- GetName(), GetWinRate(), GetAvgPlacement()등은 덱의 정보를 반환합니다.
- SetScore(int new_score): 덱의 점수를 설정합니다.
- AddRecommendedItem(const Item& itm): 추천 아이템을 덱에 추가합니다.
- PrintDeckInfo(): 덱의 모든 정보를 포맷에 맞춰 출력합니다.

- 코드 스크린샷

```
20 // 아이템 클래스 정의
21 class Item {
22 private:
23     string name;
24     string type;
25
26 public:
27     Item(string n, string t) : name(n), type(t) {}
28
29     string GetName() const { return name; }
30     string GetType() const { return type; }
31
32     void PrintItemInfo() const {
33         cout << "아이템 이름: " << name << ", 유형: " << type << "\n";
34     }
35 };
36
```

```
43 // 덱 클래스 정의
44 class Deck {
45 private:
46     string name;
47     double win_rate;
48     double avg_placement;
49     double top4_rate;
50     double usage_count;
51     int score;
52     vector<string> champions;
53     vector<Item> recommended_items;
54
55 public:
56     // 생성자
57     Deck(string n, double wr, double ap, double t4r, double uc, vector<string> champs)
58         : name(n), win_rate(wr), avg_placement(ap), top4_rate(t4r), usage_count(uc), champions(champs), score(0) {}
59
60     // 멤버 변수 접근을 위한 getter 함수
61     string GetName() const { return name; }
62     double GetWinRate() const { return win_rate; }
63     double GetAvgPlacement() const { return avg_placement; }
64     double GetTop4Rate() const { return top4_rate; }
65     double GetUsageCount() const { return usage_count; }
66     int GetScore() const { return score; }
67     const vector<string>& GetChampions() const { return champions; }
68     const vector<Item>& GetRecommendedItems() const { return recommended_items; }
69
70     // 점수 설정 메서드
71     void SetScore(int new_score) {
72         score = new_score;
73     }
74
75     void SetScore(){
76         // 가중치 설정
77         const double win_rate_weight = 0.4;
78         const double avg_placement_weight = 0.3;
79         const double top4_rate_weight = 0.2;
80         const double usage_count_weight = 0.1;
81
82         // 기준값
83         const double max_win_rate = 100.0; // 승률 최대값 100%
84         const double min_avg_placement = 1.0; // 평균 등수 최소값 1.0
85         const double max_top4_rate = 100.0; // Top4 비율 최대값 100%
86         const double max_usage_count = 10.0; // 사용 비율 최대값 (가정)
87
88         // 점수 계산 (0 ~ 100 범위로 조정)
89         double win_rate_score = (win_rate / max_win_rate) * 100 * win_rate_weight;
90         double avg_placement_score = ((4.0 - avg_placement) / (4.0 - min_avg_placement)) * 100 * avg_placement_weight;
91         double top4_rate_score = (top4_rate / max_top4_rate) * 100 * top4_rate_weight;
92         double usage_count_score = (usage_count / max_usage_count) * 100 * usage_count_weight;
93
94         // 총 점수 계산
95         score = static_cast<int>(win_rate_score + avg_placement_score + top4_rate_score + usage_count_score);
96     }
97 }
```

```

97
98 // 추천 아이템 추가 메서드
99 void AddRecommendedItem(const Item& itm) {
100     recommended_items.push_back(itm);
101 }
102
103 // 덱 정보 출력 메서드
104 void PrintDeckInfo() const {
105     string tier;
106     if (score >= 13) {
107         tier = "S";
108     } else if (score >= 9) {
109         tier = "A";
110     } else if (score >= 6) {
111         tier = "B";
112     } else {
113         tier = "C";
114     }
115     cout << "덱 이름: " << name << "\n";
116     cout << "승률: " << win_rate << "%\n";
117     cout << "평균 등수: " << avg_placement << "\n";
118     cout << "Top 4 비율: " << top4_rate << "%\n";
119     cout << "사용 비율: " << usage_count << "\n";
120     cout << "티어: " << tier << "\n";
121     cout << "구성 챔피언: ";
122     for (size_t i = 0; i < champions.size(); ++i) {
123         cout << champions[i];
124         if (i < champions.size() - 1) {
125             cout << " / ";
126         }
127     }
128     cout << "\n추천 아이템: ";
129     for (size_t i = 0; i < recommended_items.size(); ++i) {
130         cout << recommended_items[i].GetName();
131         if (i < recommended_items.size() - 1) {
132             cout << " / ";
133         }
134     }
135     cout << "\n-----\n";
136 }
137 };

```

(8) 데이터 처리 :

- 입출력

입력:

- decks.txt파일을 통해 덱 데이터를 불러옵니다.
- 코드 내부에 저장된 추천 아이템 및 증강체 데이터를 사용합니다.

출력:

- 이 데이터는 이후 사용자가 덱 정보를 요청하거나 추천을 받을 때 사용되며, main()함수 내에서 데이터를 초기화하고 설정합니다.

- 설명

덱 데이터 초기화:

- vector<Deck> decks를 통해 기본 덱 정보를 설정합니다. 각 덱의 이름, 승률, 평균 등수, Top 4 비율, 사용 비율, 구성 챔피언 등이 저장됩니다.
- 덱 객체는 각종 데이터를 텍스트 파일을 통해 초기화되며, 이 객체들은 이후 다양한 함수에서 활용됩니다.

추천 아이템 추가:

- unordered_map<string, vector<Item>> deck_items는 덱 이름과 관련된 추천 아이템을 매핑합니다.
- 반복문을 통해 각 덱의 이름을 deck_items에서 찾아 추천 아이템을 덱에 추가하는 로직을 구현합니다.
- auto& deck : decks로 덱 리스트를 순회하며, 각 덱의 이름을 사용해 추천 아이템을 추가합니다.

덱별 증강체 추천 데이터 정의:

- unordered_map<string, vector<string>> deck_augments는 각 덱과 해당 덱에 맞는 증강체들을 매핑합니다.
- 예를 들어, "차원문 덱"에 대해 "작은 친구들", "차원문 문장" 등의 증강체가 추천됩니다.
- 이 데이터는 이후 증강체 추천 기능에서 사용됩니다.

- 적용된 배운 내용

- 14주차에 배운 텍스트 파일 입출력을 추가 적용했습니다.

-코드 스크린샷

```
361 int main() {
362     // txt 파일에서 덱 데이터 읽기
363     string filename = "decks.txt";
364     vector<Deck> decks = LoadDecksFromFile(filename);
365
366     if (decks.empty()) {
367         cout << "덱 데이터가 없습니다." << endl;
368         return 0;
369     }
370
371     unordered_map<string, unordered_map<string, pair<int, int>>> deck_positions = {
372         {"차원문 덱", {{{"Ryze", {4, 1}}, {"Tatic", {1, 3}}, {"TahmKench", {1, 5}}}},
373         {"달콤술사 전사 덱", {{{"Gwen", {1, 2}}, {"Fiora", {1, 3}}, {"Rakan", {1, 4}}}},
374         {"아르카나 폭파단 덱", {{{"Varus", {4, 1}}, {"Xerath", {4, 7}}, {"TahmKench", {1, 4}}}},
375         {"요정 섀도자 덱", {{{"Kalista", {4, 1}}, {"Rakan", {1, 2}}, {"Milio", {4, 3}}}}
376     };
377
378     // 추천 아이템 추가
379     unordered_map<string, vector<Item>> deck_items = {
380         {"차원문 덱", {Item("staff", "magic"), Item("mana essence", "magic")}},
381         {"달콤술사 전사 덱", {Item("sword", "physical"), Item("chain vest", "armor")}},
382         {"아르카나 폭파단 덱", {Item("staff", "magic"), Item("giant belt", "health")}},
383         {"요정 섀도자 덱", {Item("bow", "ranged"), Item("mana essence", "magic")}}
384     };
385
386     for (auto& deck : decks) {
387         auto it = deck_items.find(deck.GetName());
388         if (it != deck_items.end()) {
389             for (const auto& itm : it->second) {
390                 deck.AddRecommendedItem(itm);
391             }
392         }
393     }
394
395     // 덱별 증강체 추천 데이터 정의
396     unordered_map<string, vector<string>> deck_augments = {
397         {"차원문 덱", {"작은 친구들", "차원문 문장"}},
398         {"달콤술사 전사 덱", {"캐러멜을 바른 아늑함", "달콤술사 문장", "큰 꾸러미"}},
399         {"아르카나 폭파단 덱", {"아르카나 전달자", "아르카나 문장", "단결된 의지"}},
400         {"요정 섀도자 덱", {"왕실 근위대", "판도라의 아이템", "요정 문장"}}
401     };
402 }
```

9) 덱 데이터 파일 읽기

- 입출력

- 입력 : 텍스트 파일(decks.txt)의 경로(filename)를 받아 덱 정보를 읽어옵니다.
- 출력 : 덱 정보를 담은 `vector<Deck>` 반환합니다.

- 설명

- `ifstream`을 사용하여 파일을 연 후, 파일을 열지 못하면 에러 메시지를 출력하고 빈 덱 벡터를 반환합니다.
- 파일의 각 줄을 읽어와 `stringstream`을 이용해 데이터를 처리합니다.
- `stringstream::str()`를 기준으로 데이터를 분리하여 덱 이름, 승률, 평균 등수, Top 4 비율 등을 추출합니다.
- `stringstream::str()`함수를 사용하여 문자열 데이터를 숫자로 변환합니다.
- 챔피언 목록은 슬래시(/)로 구분되어 있으며, 이를 `getline`을 사용해 벡터(`vector<string>`)로 분리합니다.
- 분리된 데이터를 기반으로 `Deck`객체를 생성한 후, 벡터에 추가합니다.

- 적용된 배운 내용 (예: 반복문, 조건문, 클래스, 함수, 포인터 등)

14주차에 배운 파일 입출력을 적용했습니다.

- 코드 스크린샷

```
211 // 파일에서 덱 데이터를 읽는 함수
212 vector<Deck> LoadDecksFromFile(const string& filename) {
213     vector<Deck> decks;
214     ifstream file(filename);
215
216     if (!file.is_open()) {
217         cerr << "파일을 열 수 없습니다: " << filename << endl;
218         return decks;
219     }
220
221     string line;
222     while (getline(file, line)) {
223         stringstream ss(line);
224         string name, win_rate_str, avg_placement_str, top4_rate_str, usage_count_str, champions_str;
225
226         // 데이터를 ','를 기준으로 분리
227         getline(ss, name, ',');
228         getline(ss, win_rate_str, ',');
229         getline(ss, avg_placement_str, ',');
230         getline(ss, top4_rate_str, ',');
231         getline(ss, usage_count_str, ',');
232         getline(ss, champions_str, ',');
233
234         // 문자열을 숫자로 변환
235         double win_rate = stod(win_rate_str);
236         double avg_placement = stod(avg_placement_str);
237         double top4_rate = stod(top4_rate_str);
238         double usage_count = stod(usage_count_str);
239
240         // 챔피언을 '/' 기준으로 분리
241         vector<string> champions;
242         stringstream champs_ss(champions_str);
243         string champion;
244         while (getline(champs_ss, champion, '/')) {
245             champions.push_back(champion);
246         }
247
248         // Deck 객체 생성 후 벡터에 추가
249         decks.emplace_back(name, win_rate, avg_placement, top4_rate, usage_count, champions);
250     }
251
252     file.close();
253     return decks;
254 }
255
```

10) 개인 피드백 제공

- 입출력

입력 :

- filename: 플레이 데이터가 저장된 파일 경로
- decks: 덱 정보를 담은 vector<Deck>

출력 :

- 챔피언과 아이템의 매칭 결과를 분석하여 출력합니다.
- 매칭되지 않은 실수를 요약하여 출력합니다.

- 설명

- 입력 파일을 열지 못할 경우 에러 메시지를 출력하고 함수 종료시킵니다.
 - 각 줄을 읽어 덱 이름, 챔피언 이름, 아이템 이름을 쉼표(,)를 기준으로 분리합니다.
 - deck_name과 일치하는 덱을 decks에서 검색. 존재하지 않으면 오류 메시지를 출력합니다.
 - 덱의 챔피언 목록에서 champion_name이 존재하는지 확인합니다.
 - 덱의 추천 아이템 목록에서 item_name이 존재하는지 확인합니다.
 - 챔피언이나 아이템이 매칭되지 않았을 경우, mistake_summary 맵에 해당 덱의 실수를 누적합니다.
 - 분석된 실수 데이터를 요약하여 콘솔에 출력합니다.
- 적용된 배운 내용 (예: 반복문, 조건문, 클래스, 함수, 포인터 등)
- 조건문 및 반복문을 사용하였습니다.

- 코드 스크린샷

```
256 //개인 피드백 제공
257 void AnalyzeMistakesWithDeck(const string& filename, const vector<Deck>& decks) {
258     ifstream file(filename);
259     if (!file.is_open()) {
260         cerr << "사용자 데이터를 열 수 없습니다: " << filename << endl;
261         return;
262     }
263
264     unordered_map<string, int> mistake_summary;
265     string line;
266     while (getline(file, line)) {
267         stringstream ss(line);
268         string deck_name, champion_name, item_name;
269
270         // CSV 데이터 읽기
271         getline(ss, deck_name, ',');
272         getline(ss, champion_name, ',');
273         getline(ss, item_name, ',');
274
275         // 디버깅: 읽은 데이터 확인
276         cout << "읽은 데이터 -> 덱: " << deck_name << ", 챔피언: " << champion_name << ", 아이템: " << item_name << endl;
277
278         // 덱 확인
279         auto it = find_if(decks.begin(), decks.end(), [&deck_name](const Deck& d) {
280             return d.GetName() == deck_name;
281         });
282
283         if (it != decks.end()) {
284             const Deck& deck = *it;
285
286             // 챔피언 확인
287             bool champion_found = find(deck.GetChampions().begin(), deck.GetChampions().end(), champion_name) != deck.GetChampions().end();
288             cout << "챔피언 매칭 결과: " << (champion_found ? "매칭됨" : "매칭되지 않음") << endl;
289
290             // 아이템 확인
291             bool item_found = false;
292             for (const auto& item : deck.GetRecommendedItems()) {
293                 if (item.GetName() == item_name) {
294                     item_found = true;
295                     break;
296                 }
297             }
298             cout << "아이템 매칭 결과: " << (item_found ? "매칭됨" : "매칭되지 않음") << endl;
299
300             // 결과 저장
301             if (!champion_found) {
302                 mistake_summary["챔피언 잘못 선택"]++;
303             }
304             if (!item_found) {
305                 mistake_summary["아이템 잘못 선택"]++;
306             }
307         } else {
308             mistake_summary["알 수 없는 덱"]++;
309             cout << "알 수 없는 덱: " << deck_name << endl;
310         }
311     }
312
313     // 결과 출력
314     cout << "실수 분석 결과:\n";
315     if (mistake_summary.empty()) {
316         cout << "실수가 없습니다." << endl;
317     } else {
318         for (const auto& entry : mistake_summary) {
319             cout << entry.first << " - " << entry.second << "회\n";
320         }
321     }
322     cout << "-----\n";
323
324     file.close();
325 }
```

11) 시각적 배치도 제공

- 입출력

- 입력 : unordered_map<string, pair<int, int>> positions: 챔피언 이름과 해당 위치 정보를 포함한 맵.
- 출력 : 콘솔에 챔피언 이름이 배치된 4x7 배치도를 출력합니다.

- 설명

- rows = 4, cols = 7크기의 2차원 벡터 board를 생성하여 빈 칸(" ")으로 초기화합니다.
- positions맵을 순회하며 각 챔피언의 위치 정보를 보드에 반영합니다.
- 위치 정보를 1기반 인덱스에서 0기반 인덱스로 변환 후, 유효성 검사를 합니다.
- champion.substr(0, 3)를 사용해 챔피언 이름의 앞 3글자만 저장했습니다.
- 이중 반복문을 통해 board의 내용을 출력합니다.
- 각 행의 데이터를 출력한 뒤 줄바꿈을 추가했습니다.
- 출력이 끝난 후 구분선("-----")을 추가했습니다.

- 코드 스크린샷

```
327 // 시각적 배치도 출력 함수
328 void DisplayBoard(const unordered_map<string, pair<int, int>>& positions) {
329     const int rows = 4, cols = 7; // 4x7 그리드
330     vector<vector<string>> board(rows, vector<string>(cols, " "));
331
332     for (const auto& [champion, pos] : positions) {
333         int x = pos.first - 1; // 1-based to 0-based indexing
334         int y = pos.second - 1;
335         if (x >= 0 && x < rows && y >= 0 && y < cols) {
336             board[x][y] = champion.substr(0, 3); // 챔피언 이름 줄임
337         }
338     }
339
340     cout << "\n----- 시각적 배치도 ----- \n";
341     for (const auto& row : board) {
342         for (const auto& cell : row) {
343             cout << cell << " ";
344         }
345         cout << "\n";
346     }
347     cout << "----- \n";
348 }
```

2) 테스트 결과

(1) 최고 승률 덱 추천

덱 추천 프로그램입니다. 옵션을 선택하세요:

1. 최고 승률 덱 추천
2. 모든 덱 정보 보기
3. 챔피언 기반 덱 추천
4. 아이템 추천
5. 덱에 맞는 증강체 추천
6. 개인 피드백 제공
7. 배치도 추천
8. 종료

선택: 1

추천 덱 (승률 기준):

덱 이름: 차원문 덱

승률: 15.3%

평균 등수: 4.31

Top 4 비율: 52.5%

사용 비율: 0.3

티어: S

구성 챔피언: Ryze / Taric / TahmKench

추천 아이템: staff / mana essence

(2) 덱 정보 조회

덱 추천 프로그램입니다. 옵션을 선택하세요:

1. 최고 승률 덱 추천
2. 모든 덱 정보 보기
3. 챔피언 기반 덱 추천
4. 아이템 추천
5. 덱에 맞는 증강체 추천
6. 개인 피드백 제공
7. 배치도 추천
8. 종료

선택: 2

덱 이름: 차원문 덱

승률: 15.3%

평균 등수: 4.31

Top 4 비율: 52.5%

사용 비율: 0.3

티어: S

구성 챔피언: Ryze / Taric / TahmKench

추천 아이템: staff / mana essence

덱 이름: 달콤술사 전사 덱

승률: 16.5%

평균 등수: 4.43

Top 4 비율: 50.1%

사용 비율: 0.3

티어: A

구성 챔피언: Gwen / Fiora / Rakan

추천 아이템: sword / chain vest

덱 이름: 요정 세도자 덱

승률: 14.1%

평균 등수: 4.38

Top 4 비율: 52.2%

사용 비율: 0.48

티어: A

구성 챔피언: Kalista / Rakan / Milio

추천 아이템: bow / mana essence

덱 이름: 아르카나 폭파단 덱

승률: 13%

평균 등수: 4.66

Top 4 비율: 46.4%

사용 비율: 0.17

티어: B

구성 챔피언: Varus / TahmKench / Xerath

추천 아이템: staff / giant belt

(3) 챔피언 기반 덱 추천

추천받고 싶은 챔피언 이름을 입력하세요: Ryze 덱 이름: 차원문 덱 승률: 15.3% 평균 등수: 4.31 Top 4 비율: 52.5% 사용 비율: 0.3 티어: S 구성 챔피언: Ryze / Taric / TahmKench 추천 아이템: staff / mana essence -----	추천받고 싶은 챔피언 이름을 입력하세요: Taric 덱 이름: 차원문 덱 승률: 15.3% 평균 등수: 4.31 Top 4 비율: 52.5% 사용 비율: 0.3 티어: S 구성 챔피언: Ryze / Taric / TahmKench 추천 아이템: staff / mana essence -----
추천받고 싶은 챔피언 이름을 입력하세요: TahmKench 덱 이름: 차원문 덱 승률: 15.3% 평균 등수: 4.31 Top 4 비율: 52.5% 사용 비율: 0.3 티어: S 구성 챔피언: Ryze / Taric / TahmKench 추천 아이템: staff / mana essence ----- 덱 이름: 아르카나 폭파단 덱 승률: 13% 평균 등수: 4.66 Top 4 비율: 46.4% 사용 비율: 0.17 티어: B 구성 챔피언: Varus / TahmKench / Xerath 추천 아이템: staff / giant belt -----	추천받고 싶은 챔피언 이름을 입력하세요: Rakan 덱 이름: 달콤술사 전사 덱 승률: 16.5% 평균 등수: 4.43 Top 4 비율: 50.1% 사용 비율: 0.3 티어: A 구성 챔피언: Gwen / Fiora / Rakan 추천 아이템: sword / chain vest ----- 덱 이름: 요정 쇠도자 덱 승률: 14.1% 평균 등수: 4.38 Top 4 비율: 52.2% 사용 비율: 0.48 티어: A 구성 챔피언: Kalista / Rakan / Milio 추천 아이템: bow / mana essence -----
추천받고 싶은 챔피언 이름을 입력하세요: Fiora 덱 이름: 달콤술사 전사 덱 승률: 16.5% 평균 등수: 4.43 Top 4 비율: 50.1% 사용 비율: 0.3 티어: A 구성 챔피언: Gwen / Fiora / Rakan 추천 아이템: sword / chain vest -----	추천받고 싶은 챔피언 이름을 입력하세요: Gwen 덱 이름: 달콤술사 전사 덱 승률: 16.5% 평균 등수: 4.43 Top 4 비율: 50.1% 사용 비율: 0.3 티어: A 구성 챔피언: Gwen / Fiora / Rakan 추천 아이템: sword / chain vest -----
추천받고 싶은 챔피언 이름을 입력하세요: Kalista 덱 이름: 요정 쇠도자 덱 승률: 14.1% 평균 등수: 4.38 Top 4 비율: 52.2% 사용 비율: 0.48 티어: A 구성 챔피언: Kalista / Rakan / Milio 추천 아이템: bow / mana essence -----	추천받고 싶은 챔피언 이름을 입력하세요: Milio 덱 이름: 요정 쇠도자 덱 승률: 14.1% 평균 등수: 4.38 Top 4 비율: 52.2% 사용 비율: 0.48 티어: A 구성 챔피언: Kalista / Rakan / Milio 추천 아이템: bow / mana essence -----
추천받고 싶은 챔피언 이름을 입력하세요: Varus 덱 이름: 아르카나 폭파단 덱 승률: 13% 평균 등수: 4.66 Top 4 비율: 46.4% 사용 비율: 0.17 티어: B 구성 챔피언: Varus / TahmKench / Xerath 추천 아이템: staff / giant belt -----	추천받고 싶은 챔피언 이름을 입력하세요: Xerath 덱 이름: 아르카나 폭파단 덱 승률: 13% 평균 등수: 4.66 Top 4 비율: 46.4% 사용 비율: 0.17 티어: B 구성 챔피언: Varus / TahmKench / Xerath 추천 아이템: staff / giant belt -----

잘못 입력 시

추천받고 싶은 챔피언 이름을 입력하세요: dd
해당 챔피언을 포함한 덱 정보가 없습니다.

(4) 아이템 추천

- 챔피언 입력 시

덱 추천 프로그램입니다. 옵션을 선택하세요:
1. 최고 승률 덱 추천
2. 모든 덱 정보 보기
3. 챔피언 기반 덱 추천
4. 아이템 추천
5. 종료
선택: 4
1. 챔피언에 맞는 아이템 추천
2. 아이템에 맞는 챔피언 추천
선택: 1

추천받고 싶은 챔피언 이름을 입력하세요: Ryze
챔피언 Ryze에게 추천되는 아이템:
staff
mana essence

추천받고 싶은 챔피언 이름을 입력하세요: Taric
챔피언 Taric에게 추천되는 아이템:
chain vest
magic cloak

추천받고 싶은 챔피언 이름을 입력하세요: TahmKench
챔피언 TahmKench에게 추천되는 아이템:
giant belt

추천받고 싶은 챔피언 이름을 입력하세요: Gwen
챔피언 Gwen에게 추천되는 아이템:
staff

추천받고 싶은 챔피언 이름을 입력하세요: Fiora
챔피언 Fiora에게 추천되는 아이템:
sword

추천받고 싶은 챔피언 이름을 입력하세요: Rakan
챔피언 Rakan에게 추천되는 아이템:
giant belt

추천받고 싶은 챔피언 이름을 입력하세요: Varus
챔피언 Varus에게 추천되는 아이템:
sword
mana essence

추천받고 싶은 챔피언 이름을 입력하세요: Xerath
챔피언 Xerath에게 추천되는 아이템:
staff
mana essence

추천받고 싶은 챔피언 이름을 입력하세요: Kalista
챔피언 Kalista에게 추천되는 아이템:
bow

추천받고 싶은 챔피언 이름을 입력하세요: Milio
챔피언 Milio에게 추천되는 아이템:
mana essence

- 아이템 입력 시

추천받고 싶은 아이템 이름을 입력하세요: staff 아이템 staff에 맞는 챔피언: Ryze Gwen Xerath	추천받고 싶은 아이템 이름을 입력하세요: mana essence 아이템 mana essence에 맞는 챔피언: Ryze Varus Xerath Milio
------------------------------------------------------------------------------	------------------------------------------------------------------------------------------------------

추천받고 싶은 아이템 이름을 입력하세요: chain vest 아이템 chain vest에 맞는 챔피언: Taric	추천받고 싶은 아이템 이름을 입력하세요: giant belt 아이템 giant belt에 맞는 챔피언: TahmKench Rakan
-----------------------------------------------------------------------	------------------------------------------------------------------------------------

추천받고 싶은 아이템 이름을 입력하세요: sword 아이템 sword에 맞는 챔피언: Fiora Varus	추천받고 싶은 아이템 이름을 입력하세요: bow 아이템 bow에 맞는 챔피언: Kalista
----------------------------------------------------------------------	-----------------------------------------------------------

- 잘못 입력 시

```

덱 추천 프로그램입니다. 옵션을 선택하세요:
1. 최고 승률 덱 추천
2. 모든 덱 정보 보기
3. 챔피언 기반 덱 추천
4. 아이템 추천
5. 덱에 맞는 증강체 추천
6. 개인 피드백 제공
7. 배치도 추천
8. 종료
선택: 4
1. 챔피언에 맞는 아이템 추천
2. 아이템에 맞는 챔피언 추천
선택: 3
잘못된 입력입니다. 다시 시도하세요.
    
```

```

덱 추천 프로그램입니다. 옵션을 선택하세요:
1. 최고 승률 덱 추천
2. 모든 덱 정보 보기
3. 챔피언 기반 덱 추천
4. 아이템 추천
5. 덱에 맞는 증강체 추천
6. 개인 피드백 제공
7. 배치도 추천
8. 종료
선택: 4
1. 챔피언에 맞는 아이템 추천
2. 아이템에 맞는 챔피언 추천
선택: 2
추천받고 싶은 아이템 이름을 입력하세요: dd
해당 아이템에 대한 챔피언 정보가 없습니다.
-----
    
```

(5) 증강체 추천

선택: 5
증강체를 추천받고 싶은 덱을 선택하세요:
1. 차원문 덱
2. 달콤술사 전사 덱
3. 아르카나 폭파단 덱
4. 요정 채도자 덱
선택: 1
덱 차원문 덱에 추천되는 증강체:
작은 친구들
차원문 문장

선택: 5
증강체를 추천받고 싶은 덱을 선택하세요:
1. 차원문 덱
2. 달콤술사 전사 덱
3. 아르카나 폭파단 덱
4. 요정 채도자 덱
선택: 2
덱 달콤술사 전사 덱에 추천되는 증강체:
캐러멜을 바른 아늑함
달콤술사 문장
큰 꾸러미

선택: 5
증강체를 추천받고 싶은 덱을 선택하세요:
1. 차원문 덱
2. 달콤술사 전사 덱
3. 아르카나 폭파단 덱
4. 요정 채도자 덱
선택: 3
덱 아르카나 폭파단 덱에 추천되는 증강체:
아르카나 전달자
아르카나 문장
단결된 의지

선택: 5
증강체를 추천받고 싶은 덱을 선택하세요:
1. 차원문 덱
2. 달콤술사 전사 덱
3. 아르카나 폭파단 덱
4. 요정 채도자 덱
선택: 4
덱 요정 채도자 덱에 추천되는 증강체:
왕실 근위대
판도라의 아이템
요정 문장

잘못 입력 시

선택: 5
증강체를 추천받고 싶은 덱을 선택하세요:
1. 차원문 덱
2. 달콤술사 전사 덱
3. 아르카나 폭파단 덱
4. 요정 채도자 덱
선택: 5
잘못된 입력입니다. 다시 시도하세요.

(6) 개인 피드백 제공

```
선택: 6
사용자 데이터 파일 이름을 입력하세요: mistake.txt
읽은 데이터 -> 덱: 차원문 덱, 챔피언: Ryze, 아이템: staff
챔피언 매칭 결과: 매칭됨
아이템 매칭 결과: 매칭됨
읽은 데이터 -> 덱: 달콤술사 전사 덱, 챔피언: Varus, 아이템: sword
챔피언 매칭 결과: 매칭되지 않음
아이템 매칭 결과: 매칭됨
읽은 데이터 -> 덱: 아르카나 폭파단 덱, 챔피언: Varus, 아이템: bow
챔피언 매칭 결과: 매칭됨
아이템 매칭 결과: 매칭되지 않음
읽은 데이터 -> 덱: 차원문 덱, 챔피언: Taric, 아이템: mana essence
챔피언 매칭 결과: 매칭됨
아이템 매칭 결과: 매칭됨
실수 분석 결과:
아이템 잘못 선택 - 1회
챔피언 잘못 선택 - 1회
-----
```

잘못 입력 시

```
덱 추천 프로그램입니다. 옵션을 선택하세요:
1. 최고 승률 덱 추천
2. 모든 덱 정보 보기
3. 챔피언 기반 덱 추천
4. 아이템 추천
5. 덱에 맞는 증강체 추천
6. 개인 피드백 제공
7. 배치도 추천
8. 종료
선택: 6
사용자 데이터 파일 이름을 입력하세요: d
사용자 데이터를 열 수 없습니다: d
```

(7) 시각적 배치도 제공

<pre>선택: 7 배치도를 추천받고 싶은 덱을 선택하세요: 1. 차원문 덱 2. 달콤술사 전사 덱 3. 아르카나 폭파단 덱 4. 요정 채도자 덱 선택: 1 차원문 덱 배치도 추천: ----- 시각적 배치도 ----- . . Tar . Tah Ryz -----</pre>	<pre>선택: 7 배치도를 추천받고 싶은 덱을 선택하세요: 1. 차원문 덱 2. 달콤술사 전사 덱 3. 아르카나 폭파단 덱 4. 요정 채도자 덱 선택: 2 달콤술사 전사 덱 배치도 추천: ----- 시각적 배치도 ----- . Gwe Fio Rak -----</pre>
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

<p>선택: 7 배치도를 추천받고 싶은 덱을 선택하세요:</p> <ol style="list-style-type: none"> 1. 차원문 덱 2. 달콤술사 전사 덱 3. 아르카나 폭파단 덱 4. 요정 색도자 덱 <p>선택: 3 아르카나 폭파단 덱 배치도 추천:</p> <p>----- 시각적 배치도 -----</p> <pre> . . . Tah Var Xer </pre> <p>-----</p>	<p>선택: 7 배치도를 추천받고 싶은 덱을 선택하세요:</p> <ol style="list-style-type: none"> 1. 차원문 덱 2. 달콤술사 전사 덱 3. 아르카나 폭파단 덱 4. 요정 색도자 덱 <p>선택: 4 요정 색도자 덱 배치도 추천:</p> <p>----- 시각적 배치도 -----</p> <pre> . Rak Kal . Mil </pre> <p>-----</p>
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

잘못 입력 시

```

선택: 7
배치도를 추천받고 싶은 덱을 선택하세요:
1. 차원문 덱
2. 달콤술사 전사 덱
3. 아르카나 폭파단 덱
4. 요정 색도자 덱
선택: 5
잘못된 입력입니다. 다시 시도하세요.

```

(8) 종료 입력 시

```

덱 추천 프로그램입니다. 옵션을 선택하세요:
1. 최고 승률 덱 추천
2. 모든 덱 정보 보기
3. 챔피언 기반 덱 추천
4. 아이템 추천
5. 덱에 맞는 증강체 추천
6. 개인 피드백 제공
7. 배치도 추천
8. 종료
선택: 8
프로그램을 종료합니다.
PS C:\Users\1\Desktop\c++\project\CPP2409-P>

```

4. 계획 대비 변경 사항

3-2) 유사 취향의 상위 플레이어의 전략 추천 기능

- 사용자와 유사한 전략을 지닌 상위 플레이어들과 사용자의 다른 점을 비교하여 피드백해주고, 상위 랭크 플레이어의 전략을 추천
- 삭제
- 사유 : 유사 취향의 상위 플레이어의 전략 추천 기능 = 추천 아이템 / 추천 증강체와 동일한 기능이라 추가하지 않았습니다.

4-1) 증강체의 승률 분석

- 각 증강체의 승률, 평균 순위 등을 분석하여 각 증강체에 랭크를 정하여 추천 해줌.
- 삭제
- 사유 : 게임의 승률, 평균 순위 등은 주로 텍이나 챔피언에 따라 정해지고 증강체는 추가되는 +@의 느낌이기에 데이터가 올바르게 정해지지 않습니다. 그렇기에 삭제했습니다.

5. 느낀 점

이번 TFT Guide를 진행하며 첫 개인 프로젝트이기에 중간중간 난관에 봉착했을 때 매우 큰 스트레스를 받기도 했지만, 결국 완성본을 보고 나니 감회가 새로웠습니다. 저는 본 수업 이전에 프로그래밍 언어를 모두 시험을 위해 이론으로만 열심히 공부했기에 전혀 좋은 실력이 아니었습니다. 하지만 C++ 수업을 통해 실습 및 과제, 그리고 개인 프로젝트를 수행하며 아직 부족하지만 반년 전 저보다 확실히 실력이 늘었다는 것을 느낄 수 있었습니다.

개인적으로 TFT Guide를 만들면서 객체지향 프로그래밍의 중요성과 파일 활용으로 인한 편리함 등이 와닿았습니다. Deck이나 Item 클래스를 활용해 데이터를 구조화하면서 객체지향 프로그래밍의 강점을 체감했고, 데이터를 캡슐화하고 클래스 내부에서 로직을 처리하도록 설계함으로써 코드의 재사용성과 가독성을 높일 수 있었습니다.

파일은 제가 아무래도 통계를 기반으로 여러 가지 텍을 추천하는 프로그램을 만들다 보니 필수불가결한 부분이었는데, 수업 마지막쯤에 배워 이를 적용할 수 있었습니다. 파일 입력을 통해 프로그램이 데이터를 동적으로 불러오고 처리하도록 구현하면서 실제 데이터를 다루는 데 필요한 여러 가지 기술을 익힐 수 있었습니다.

그리고 제가 어려움을 겪었던 부분은 매우 많았지만, 그 과정에서 디버깅의 중요성을 깨닫게 되었습니다. 대표적으로는 마지막쯤에 개인 피드백 전략을 추가하면서 데이터와의 매칭이 하나도 수행되지 않아 디버깅을 통해 결국 오류를 수정하고 문제를 해결할 수 있었습니다.

한편으로는 아쉬운 부분도 많이 남습니다. 결코 짧은 시간은 아니었지만, 학기 중이기에 온전히 시간을 쏟을 수 없었던 부분이 있었습니다. 가장 아쉬운 부분은 인터페이스가 유저 친화적이지 못했다는 점입니다. 시각적으로 더 보기 편하게 만들 수 있었을 것 같은데 아직 제 능력으로는 부족했습니다. 그 부분이 상당히 아쉽습니다.

이번 프로젝트는 C++의 다양한 기능과 객체지향적 설계의 강점을 실습하고, 프로그램 개발의 첫 단계를 경험할 수 있는 소중한 기회였습니다. 이러한 경험은 앞으로 더 복잡한 문제를 해결하거나 실질적인 응용 프로그램을 설계하는 데 큰 밑거름이 될 것 같습니다!