

1. 서론

1-1. 프로젝트 목적 및 배경

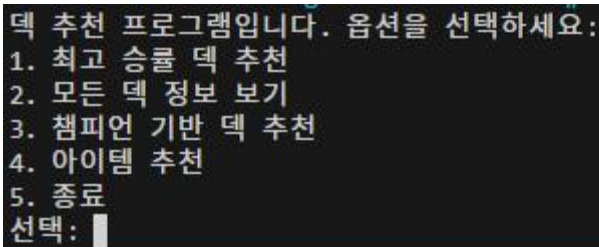
: TFT Guide는 TFT를 즐기는 사용자들에게 덱 선택과 아이템 구성을 효과적으로 도와주기 위해 만들기 시작 프로그램입니다. 이 프로그램은 덱의 승률, 평균 등수, 구성 챔피언 등을 바탕으로 플레이어에게 가장 효율적인 덱과 아이템을 추천하는 역할을 합니다.

1-2. 목표

: 최고 승률의 덱 추천 기능, 아이템 추천 기능 구현

1-3. 사용 설명서

:



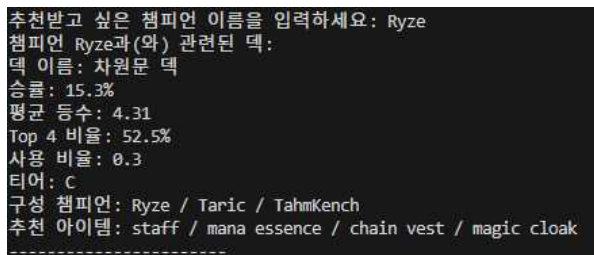
```
덱 추천 프로그램입니다. 옵션을 선택하세요:
1. 최고 승률 덱 추천
2. 모든 덱 정보 보기
3. 챔피언 기반 덱 추천
4. 아이템 추천
5. 종료
선택: █
```

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

챔피언의 종류는 10개로 이루어져 있습니다.

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

챔피언은 작성된 것과 동일하게 입력하지 않으면 실행되지 않습니다.

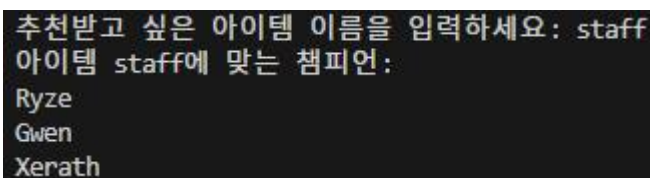


```
추천받고 싶은 챔피언 이름을 입력하세요: Ryze
챔피언 Ryze과(와) 관련된 덱:
덱 이름: 차원문 덱
승률: 15.3%
평균 등수: 4.31
Top 4 비율: 52.5%
사용 비율: 0.3
티어: C
구성 챔피언: Ryze / Taric / TahmKench
추천 아이템: staff / mana essence / chain vest / magic cloak
-----
```

아이템은 6종류로 이루어져 있습니다.

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

아이템도 챔피언처럼 작성된 것과 동일하게 입력하지 않으면 실행되지 않습니다.



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

2. 요구사항

2-1. 사용자 요구사항

: 사용자는 TFT Guide 프로그램을 통해 덱 정보를 확인하고, 아이템 추천을 받을 수 있습니다. 사용자는 터미널에서 다양한 옵션을 선택하여 덱의 구성, 승률, 추천 아이템 등을 조회할 수 있습니다.

2-2. 기능 계획

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 수행.

② 최고 승률 덱 추천 : 현재 등록된 덱들의 승률, 평균 등수, Top 4 비율을 점수로 환산하여 가장 높은 점수의 덱을 추천. 또한 덱의 이름, 구성 챔피언, 추천 아이템, 승률 등 세부 정보를 출력.

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

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

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 추천.

2-3. 함수 계획

① 메인 함수

: 사용자에게 계속해서 입력을 받아, 사용자의 선택에 따라 적절한 함수를 호출하여 각 기능을 수행합니다. 주요 기능으로는 덱 추천, 덱 정보 보기, 챔피언 기반 덱 추천, 아이템 추천, 프로그램 종료 등이 있습니다.

② 덱 추천 함수 (recommendDeck)

: 모든 덱의 승률, 평균 등수, Top 4 비율을 기준으로 각 덱의 점수를 계산합니다. 또한 계산된 점수를 기준으로 덱을 정렬하여 사용자가 선택할 수 있도록 추천합니다.

③ 덱 정보 출력 함수 (viewDeckDetails)

: 사용자에게 모든 덱의 이름을 출력하여 선택할 수 있도록 하고, 사용자가 선택한 덱의 승률, 평균 등수, 구성 챔피언, 추천 아이템 등 상세 정보를 출력합니다.

④ 챔피언 기반 덱 추천 함수 (recommendDeckByChampion)

: 사용자가 챔피언 이름을 입력하면, 해당 챔피언이 포함된 덱을 검색하여 출력합니다. 또한 덱의 이름, 승률, 구성 챔피언, 추천 아이템 등을 출력하여 사용자가 참고할 수 있도록 합니다.

⑤ 아이템 추천 함수 (recommendItemsByChampion, recommendChampionsByItem)

: recommendItemsByChampion은 사용자가 입력한 챔피언에 맞는 아이템들을 추천하고, recommendChampionsByItem은 사용자가 입력한 아이템에 적합한 챔피언들을 추천합니다.

⑥ 아이템 추가 함수 (addItemToDecks)

: 초기 덱 데이터에 각 덱에 적합한 추천 아이템들을 추가합니다.

⑦ 덱 정보 출력 함수 (printDeckInfo)

: 각 덱의 세부 정보를 출력합니다.

3. 설계 및 구현

3-1. 기능 별 구현 사항:

①에 대한 구현

1. 스크린샷

```
int choice;
while (true) {
    cout << "덱 추천 프로그램입니다. 옵션을 선택하세요:\n";
    cout << "1. 최고 승률 덱 추천\n";
    cout << "2. 모든 덱 정보 보기\n";
    cout << "3. 챔피언 기반 덱 추천\n";
    cout << "4. 아이템 추천\n";
    cout << "5. 종료\n";
    cout << "선택: ";
    cin >> choice;

    if (choice == 1) {
        recommendDeck(decks);
    } else if (choice == 2) {
        viewDeckDetails(decks);
    } else if (choice == 3) {
        recommendDeckByChampion(decks);
    } else if (choice == 4) {
        int subChoice;
        cout << "1. 챔피언에 맞는 아이템 추천\n";
        cout << "2. 아이템에 맞는 챔피언 추천\n";
        cout << "선택: ";
        cin >> subChoice;

        if (subChoice == 1) {
            recommendItemsByChampion(championItems);
        } else if (subChoice == 2) {
            recommendChampionsByItem(itemChampions);
        } else {
            cout << "잘못된 입력입니다. 다시 시도하세요." << endl;
        }
    } else if (choice == 5) {
        cout << "프로그램을 종료합니다." << endl;
        break;
    } else {
        cout << "잘못된 입력입니다. 다시 시도하세요." << endl;
    }
}
```

2. 입력 (블록/함수에 입력되는 변수, 값들과 설명)

- choice : 사용자가 선택한 메뉴 옵션(정수)입니다.
- subChoice : 아이템 추천과 관련된 서브메뉴에서 선택한 옵션(정수)입니다.

3. 반환값 (함수의 경우 작성)

- 반환값이 없습니다.

4. 결과 (블록/함수가 종료된 결과)

- choice에 따라 선택된 함수가 실행되며, 해당 함수의 결과에 따라 정보를 출력합니다.
- choice가 5이면 프로그램이 종료됩니다.

5. 설명 (코드 내 작동 순서, 내용 등 추가 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 : subChoice에 따라 recommendItemsByChampion(championItems) 또는 recommend ChampionsByItem(itemChampions) 함수를 실행합니다.
 5. 종료 : 프로그램을 종료합니다.
- 잘못된 입력에 대한 처리를 포함하여, 사용자가 올바른 입력을 할 때까지 반복되도록 설계되었습니다.

②에 대한 구현

1. 스크린샷

```
// 덱 추천 가능 구현 함수 정의
void recommendDeck(vector<Deck>& decks) {
    // 덱 점수를 초기화
    for (auto& deck : decks) {
        deck.setScore(0);
    }

    // 승률, 평균 등수, Top 4 비율을 기준으로 각각 순위를 매겨 점수를 부여
    // 승률 기준 점수 부여
    sort(decks.begin(), decks.end(), [](const Deck& a, const Deck& b) {
        return a.getWinRate() > b.getWinRate();
    });
    for (int i = 0; i < decks.size(); ++i) {
        decks[i].setScore(decks[i].getScore() + (decks.size() - i));
    }

    // 평균 등수 기준 점수 부여 (낮은 순위가 좋은 점수)
    sort(decks.begin(), decks.end(), [](const Deck& a, const Deck& b) {
        return a.getAvgPlacement() < b.getAvgPlacement();
    });
    for (int i = 0; i < decks.size(); ++i) {
        decks[i].setScore(decks[i].getScore() + (decks.size() - i));
    }

    // Top 4 비율 기준 점수 부여
    sort(decks.begin(), decks.end(), [](const Deck& a, const Deck& b) {
        return a.getTop4Rate() > b.getTop4Rate();
    });
    for (int i = 0; i < decks.size(); ++i) {
        decks[i].setScore(decks[i].getScore() + (decks.size() - i));
    }

    // 최종 점수를 기준으로 덱 정렬
    sort(decks.begin(), decks.end(), [](const Deck& a, const Deck& b) {
        return a.getScore() > b.getScore();
    });

    // 추천 덱 출력
    cout << "추천 덱 (종합 기준):\n";
    decks[0].printDeckInfo();
}
```

2. 입력 (블록/함수에 입력되는 변수, 값들과 설명)
 - decks : 덱들의 벡터, 각 덱에는 승률, 평균 등수, top4 비율 등의 정보가 들어있습니다.
3. 반환값 (함수의 경우 작성)
 - 반환 값이 없습니다.
4. 결과 (블록/함수가 종료된 결과)
 - 각 덱의 점수가 적절하게 설정되어, 최종적으로 가장 점수가 높은 덱을 출력합니다.
5. 설명 (코드 내 작동 순서, 내용 등 추가 설명)
 - 덱 점수 초기화 : for 루프를 사용하여 모든 덱의 점수를 0으로 설정합니다. 0으로 설정하

지 않으면 점수가 중복으로 들어가 여러 번 추천했을 때 점수가 정확히 측정되지 않아 추가했습니다.

- 승률 기준 정렬 및 점수 부여 : 덱들의 승률(`getWinRate()`)을 기준으로 내림차순으로 정렬합니다. 정렬된 결과에서 덱의 점수를 차등 부여 형식($4 - I$)으로 부여합니다.
- 평균 등수 기준 정렬 및 점수 부여 : 덱들의 평균 등수 (`getAvgPlacement()`)를 기준으로 등수는 낮은게 높은 점수를 얻어야 하니 오름차순으로 정렬합니다. 정렬된 결과에 따라 덱들의 점수를 부여합니다.
- Top 4 비율 기준 정렬 및 점수 부여 : 각 덱들의 Top 4 비율 (`getTop4Rate()`) 기준으로 내림차순으로 정렬하고, 결과에 따른 점수를 부여합니다.
- 점수 합산 최종 정렬 : 앞서 계산한 점수를 통해 최종 점수(`getScore()`)를 기준으로 정렬합니다.
- 추천 덱 출력 : 정렬 후 가장 점수가 높은 덱의 정보를 출력합니다.

③에 대한 구현

1. 스크린샷

```
// 모든 덱의 정보를 출력하는 함수 정의
void viewDeckDetails(const vector<Deck>& decks) {
    int deckChoice;
    cout << "모든 덱의 종류:\n";
    for (int i = 0; i < decks.size(); ++i) {
        cout << i + 1 << ". " << decks[i].getName() << "\n";
    }
    cout << "-----\n";
    cout << "보고 싶은 덱의 번호를 입력하세요: ";
    cin >> deckChoice;

    if (deckChoice >= 1 && deckChoice <= decks.size()) {
        decks[deckChoice - 1].printDeckInfo();
    } else {
        cout << "잘못된 입력입니다. 다시 시도하세요." << endl;
    }
}
```

2. 입력 (블록/함수에 입력되는 변수, 값들과 설명)

- decks : 덱들의 벡터, 각 덱에는 승률, 평균 등수, top4 비율 등의 정보가 들어있습니다.

3. 반환값 (함수의 경우 작성)

- 반환 값이 없습니다.

4. 결과 (블록/함수가 종료된 결과)

- 사용자가 선택한 덱의 정보를 출력합니다.

5. 설명 (코드 내 작동 순서, 내용 등 추가 설명)

- 모든 덱 이름 출력 : for 루프를 사용하여 decks에 포함된 모든 덱의 이름을 화면에 출력합니다. $I + 1$ 를 사용해 1부터 시작하는 번호를 출력하고, 각 덱의 이름은 `getName()`을 호출

하여 가져옵니다.

- 사용자 입력 받기 : 덱의 번호를 입력받고, 그 값을 deckChoice에 저장합니다.
- 유효성 검사 및 정보 출력 : deckChoice가 유효한 범위인 1부터 deck.size() 까지인지를 검사합니다. 유효한 경우 사용자가 선택한 덱의 정보를 출력합니다. 유효하지 않은 번호가 입력된 경우 잘못된 입력이라고 메시지를 출력합니다.

④에 대한 구현

1. 스크린샷

```
void recommendDeckByChampion(const vector<Deck>& decks) {
    string champion;
    cout << "추천받고 싶은 챔피언 이름을 입력하세요: ";
    cin >> champion;

    cout << "챔피언 " << champion << "과(와) 관련된 덱:\n";
    for (const auto& deck : decks) {
        const auto& champions = deck.getChampions();
        if (find(champions.begin(), champions.end(), champion) != champions.end()) {
            deck.printDeckInfo();
        }
    }
}
```

2. 입력 (블록/함수에 입력되는 변수, 값들과 설명)

- decks : 덱들의 벡터, 각 덱에는 승률, 평균 등수, top4 비율 등의 정보가 들어있습니다.
- champion : 사용자가 입력한 챔피언 이름으로 문자열입니다.

3. 반환값 (함수의 경우 작성)

- 반환 값이 없습니다.

4. 결과 (블록/함수가 종료된 결과)

- 사용자가 입력한 챔피언과 관련된 덱 정보를 출력합니다.

5. 설명 (코드 내 작동 순서, 내용 등 추가 설명)

- 사용자 입력 받기 : 챔피언 이름을 사용자로부터 입력받아 champion 변수에 저장합니다.
- 관련 덱 검색 및 출력 : decks 벡터에서 모든 덱을 반복하면서, 각 덱에 포함된 챔피언 정보 deck.getChampions()를 통해 가져옵니다. find 함수를 사용해 사용자가 입력한 챔피언 이름이 덱의 챔피언 목록에 포함되어 있는 지 확인합니다. 만약 해당 챔피언이 덱에 포함되어 있다면, 그 덱의 정보를 deck.printDeckInfo()를 이용하여 출력합니다.

⑤에 대한 구현

1. 스크린샷

```
// 챔피언을 입력받아 해당 챔피언에 맞는 아이템을 추천하는 함수 정의
void recommendItemsByChampion(const unordered_map<string, vector<string>>& championItems) {
    string champion;
    cout << "추천받고 싶은 챔피언 이름을 입력하세요: ";
    cin >> champion;

    auto it = championItems.find(champion);
    if (it != championItems.end()) {
        cout << "챔피언 " << champion << "에게 추천되는 아이템:\n";
        for (const auto& item : it->second) {
            cout << item << "\n";
        }
    } else {
        cout << "해당 챔피언에 대한 아이템 정보가 없습니다." << endl;
    }
}

// 아이템을 입력받아 해당 아이템에 맞는 챔피언을 추천하는 함수 정의
void recommendChampionsByItem(const unordered_map<string, vector<string>>& itemChampions) {
    string item;
    cout << "추천받고 싶은 아이템 이름을 입력하세요: ";
    cin.ignore();
    getline(cin, item);

    auto it = itemChampions.find(item);
    if (it != itemChampions.end()) {
        cout << "아이템 " << item << "에 맞는 챔피언:\n";
        for (const auto& champion : it->second) {
            cout << champion << "\n";
        }
    } else {
        cout << "해당 아이템에 대한 챔피언 정보가 없습니다." << endl;
    }
}
```

2. 입력 (블록/함수에 입력되는 변수, 값들과 설명)

- recommendItemsByChampion : championItems는 챔피언 이름을 키로 하고 추천 아이템 벡터를 값으로 가지는 자료형입니다.
- recommendChampionsByItem : itemChampions는 아이템 이름을 키로 하고 추천 아이템 벡터를 값으로 가지는 자료형입니다.

3. 반환값 (함수의 경우 작성)

- 반환 값이 없습니다.

4. 결과 (블록/함수가 종료된 결과)

- recommendItemsByChampion 함수는 사용자가 입력한 챔피언에 대한 추천 아이템 목록을 출력합니다.
- recommendChampionsByItem 함수는 사용자가 입력한 아이템에 적합한 챔피언 목록을 출력합니다.

5. 설명 (코드 내 작동 순서, 내용 등 추가 설명)

- 사용자 입력 받기 : 첫 번째로 recommendItemsByChampion은 추천하고 싶은 챔피언 이름을 사용자로부터 입력받아 champion 변수에 저장합니다. 두 번째로 recommendChampionsByItem은 추천하고 싶은 아이템 이름을 사용자로부터 입력받아 item 변수에 저장합니다.
- 추천 항목 찾기 및 출력 : recommendItemsByChampion은 championItems에서 입력된

챔피언 이름을 찾고, 해당 챔피언이 존재하면 추천 아이템 목록을 출력합니다. 만약 존재 하지 않는다면 해당 챔피언에 대한 추천 아이템이 없다는 메시지를 출력합니다.

recommendChampionsByItem은 itemChampions에서 입력된 아이템 이름을 찾고, 해당 아이템이 존재하면 이에 어울리는 챔피언 목록을 출력합니다. 아이템이 존재하지 않으면 앞선 방식과 동일하게 아이템에 대한 추천 챔피언 정보가 없다는 메시지를 출력합니다.

클래스 대한 추가 설명

1. 스크린샷

```
// 아이템 클래스 정의
class Item {
private:
    string name;
    string type;

public:
    Item(string n, string t) : name(n), type(t) {}

    string getName() const { return name; }
    string getType() const { return type; }

    void printItemInfo() const {
        cout << "아이템 이름: " << name << ", 유형: " << type << "\n";
    }
};
```

```
// 덱 클래스 정의
class Deck {
private:
    string name;
    double winRate;
    double avgPlacement;
    double top4Rate;
    double usageCount;
    int score;
    vector<string> champions;
    vector<Item> recommendedItems;

public:
    // 생성자
    Deck(string n, double wr, double ap, double t4r, double uc, vector<string> champs)
        : name(n), winRate(wr), avgPlacement(ap), top4Rate(t4r), usageCount(uc), champions(champs), score(0) {}

    // 멤버 변수 접근을 위한 getter 함수
    string getName() const { return name; }
    double getWinRate() const { return winRate; }
    double getAvgPlacement() const { return avgPlacement; }
    double getTop4Rate() const { return top4Rate; }
    double getUsageCount() const { return usageCount; }
    int getScore() const { return score; }
    const vector<string>& getChampions() const { return champions; }
    const vector<Item>& getRecommendedItems() const { return recommendedItems; }

    // 점수 설정 메서드
    void setScore(int newScore) {
        score = newScore;
    }
};
```

```

// 추천 아이템 추가 메서드
void addRecommendedItem(const Item& item) {
    recommendedItems.push_back(item);
}

// 덱 정보 출력 메서드
void printDeckInfo() const {
    string tier;
    if (score >= 12) {
        tier = "S";
    } else if (score >= 9) {
        tier = "A";
    } else if (score >= 6) {
        tier = "B";
    } else {
        tier = "C";
    }
    cout << "덱 이름: " << name << "\n";
    cout << "승률: " << winRate << "\n";
    cout << "평균 등수: " << avgPlacement << "\n";
    cout << "Top 4 비율: " << top4Rate << "\n";
    cout << "사용 비율: " << usageCount << "\n";
    cout << "티어: " << tier << "\n";
    cout << "구성 챔피언: ";
    for (size_t i = 0; i < champions.size(); ++i) {
        cout << champions[i];
        if (i < champions.size() - 1) {
            cout << " / ";
        }
    }
    cout << "\n추천 아이템: ";
    for (size_t i = 0; i < recommendedItems.size(); ++i) {
        cout << recommendedItems[i].getName();
        if (i < recommendedItems.size() - 1) {
            cout << " / ";
        }
    }
    cout << "\n-----\n";
}
};

```

2. 입력 (블록/함수에 입력되는 변수, 값들과 설명)

- 생성자 Item(string n, string t) : n은 아이템의 이름, t는 아이템의 유형입니다.
- 생성자 Deck(string n, double w, double ap, double tr, double uc, vector<string> champs): n은 덱의 이름, w는 덱의 승률, ap는 평균 등수, tr은 Top4비율, uc는 사용 횟수, champs는 덱에 포함된 챔피언의 목록입니다.
- addRecommendedItem(const Item& item) : item은 추천 아이템 객체입니다.
- setScore(int newScore) : newScore는 덱의 점수입니다.

3. 반환값 (함수의 경우 작성)

- 아이템 클래스의 getName(), getType() : 각각 아이템의 이름과 유형을 반환합니다.
- printItemInfo() : 반환값이 없으며, 아이템의 이름과 유형을 출력합니다.
- deck 클래스의 getName(), getWinRate(), getAvgPlacement() 등 : 각 덱의 속성 값을 반환합니다.
- printDeckInfo() : 반환값이 없으며, 덱의 정보를 출력합니다.

4. 결과 (블록/함수가 종료된 결과)

- printItemInfo() : 함수 호출 시, 아이템의 이름과 유형이 출력됩니다.
- printDeckInfo() : 함수 호출 시, 덱의 이름, 승률, 평균 등수 등 덱의 세부 정보가 출력됩니다.
- setScore(int newScore) : 덱의 점수가 업데이트됩니다.
- addRecommendedItem(const Item& item) : 함수 호출 시, 덱에 추천 아이템이 추가됩니다.

5. 설명 (코드 내 작동 순서, 내용 등 추가 설명)

- Item 클래스는 아이템의 이름과 유형을 저장하고, 이를 가져오거나 출력할 수 있는

printItemInfo() 함수를 제공합니다.

- Deck 클래스는 덱의 여러 속성들을 저장하며, 이러한 속성들을 가져오거나 출력하는 기능을 제공합니다.
- SetScore는 덱의 점수를 설정합니다.
- addRecommendedItem(const Item& item)는 덱의 추천 아이템 목록에 새로운 아이템을 추가합니다.
- printDeckInfo()는 점수에 따라 덱의 티어를 설정하고 이를 출력합니다. 또한 덱의 이름, 승률, 평균 등수, 사용 횟수 등을 순서대로 출력합니다. 포함된 챔피언 목록과 추천된 아이템 목록을 반복문을 통해 출력하여 사용자가 덱에 대한 상세한 정보를 쉽게 확인할 수 있도록 합니다.

데이터 처리에 대한 추가 설명

1. 스크린샷

```
// 모든 덱에 기본 아이템 추천 추가 함수 정의
void addItemsToDecks(vector<Deck>& decks) {
    // 아이템 목록 정의
    vector<Item> items = {
        Item("sword", "sword"),
        Item("staff", "staff"),
        Item("mana essence", "mana item"),
        Item("giant belt", "health item"),
        Item("chain vest", "armor"),
        Item("magic cloak", "magic resistance"),
        Item("bow", "attack speed item")
    };

    // 각 덱에 추천 아이템 추가
    for (auto& deck : decks) {
        if (deck.getName() == "차원문 덱") {
            deck.addRecommendedItem(items[1]); // 마법 지팡이
            deck.addRecommendedItem(items[2]); // 마나의 정수
            deck.addRecommendedItem(items[4]); // 식사술 조끼
            deck.addRecommendedItem(items[5]); // 마법 저항 망토
        } else if (deck.getName() == "달콤술사 전사 덱") {
            deck.addRecommendedItem(items[0]); // 룬소드
            deck.addRecommendedItem(items[1]); // 마법 지팡이
            deck.addRecommendedItem(items[3]); // 거인의 허리띠
        } else if (deck.getName() == "아르카나 폭파단 덱") {
            deck.addRecommendedItem(items[0]); // 룬소드
            deck.addRecommendedItem(items[2]); // 마나의 정수
            deck.addRecommendedItem(items[4]); // 식사술 조끼
            deck.addRecommendedItem(items[5]); // 마법 저항 망토
        } else if (deck.getName() == "요정 색도자 덱") {
            deck.addRecommendedItem(items[6]); // 곡궁
            deck.addRecommendedItem(items[3]); // 거인의 허리띠
        }
    }
}
```

```

// 덱 데이터
vector<Deck> decks = {
    {"차원문 덱", 15.3, 4.31, 52.5, 0.30, {"Ryze", "Taric", "TahmKench"}},
    {"딜공술사 전사 덱", 16.5, 4.43, 50.1, 0.30, {"Gwen", "Fiora", "Rakan"}},
    {"아르카나 폭파단 덱", 13.0, 4.66, 46.4, 0.17, {"Varus", "TahmKench", "Xerath"}},
    {"요정 설토자 덱", 14.1, 4.38, 52.2, 0.48, {"Kalista", "Rakan", "Milio"}}
};

// 덱에 아이템 추가
addItemToDecks(decks);

// 챔피언별 아이템 추천 데이터 정의
unordered_map<string, vector<string>> championItems = {
    {"Ryze", {"staff", "mana essence"}},
    {"Taric", {"chain vest", "magic cloak"}},
    {"TahmKench", {"giant belt"}},
    {"Gwen", {"staff"}},
    {"Fiora", {"sword"}},
    {"Rakan", {"giant belt"}},
    {"Varus", {"sword", "mana essence"}},
    {"Xerath", {"staff", "mana essence"}},
    {"Kalista", {"bow"}},
    {"Milio", {"mana essence"}}
};

// 아이템별 챔피언 추천 데이터 정의
unordered_map<string, vector<string>> itemChampions = {
    {"staff", {"Ryze", "Gwen", "Xerath"}},
    {"mana essence", {"Ryze", "Varus", "Xerath", "Milio"}},
    {"chain vest", {"Taric", "아르카나 폭파단 덱의 챔피언"}},
    {"giant belt", {"TahmKench", "Rakan", "요정 설토자 덱의 챔피언"}},
    {"sword", {"Fiora", "Varus"}},
    {"bow", {"Kalista"}}
};

```

2. 입력 (블록/함수에 입력되는 변수, 값들과 설명)

- decks : 덱들의 벡터, 각 덱에는 승률, 평균 등수, top4 비율 등의 정보가 들어있습니다.

3. 반환값 (함수의 경우 작성)

- 반환 값이 없습니다.

4. 결과 (블록/함수가 종료된 결과)

- 각 덱에 추천 아이템들이 추가됩니다.

5. 설명 (코드 내 작동 순서, 내용 등 추가 설명)

- 아이템 목록 정의 : 여러 아이템을 정의한 vector를 생성합니다. 각 아이템은 이름과 유형을 가지고 있습니다.
- 각 덱에 추천 아이템 추가 : 모든 덱을 for 루프로 순회하면서 각 덱의 이름을 기준으로 조건문을 사용해 특정 아이템을 추가합니다.

4. 테스트

① 에 대한 결과

```
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: 1
추천 덱 (종합 기준):
덱 이름: 차원문 덱
승률: 15.3%
평균 등수: 4.31
Top 4 비율: 52.5%
사용 비율: 0.3
티어: A
구성 챔피언: Ryze / Taric / TahmKench
추천 아이템: staff / mana essence / chain vest / magic cloak
-----
```

② 에 대한 결과

```
-----
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: 2
모든 덱의 종류:
1. 차원문 덱
2. 요정 색도자 덱
3. 달콤술사 전사 덱
4. 아르카나 폭파단 덱
-----
보고 싶은 덱의 번호를 입력하세요: 1
덱 이름: 차원문 덱
승률: 15.3%
평균 등수: 4.31
Top 4 비율: 52.5%
사용 비율: 0.3
티어: A
구성 챔피언: Ryze / Taric / TahmKench
추천 아이템: staff / mana essence / chain vest / magic cloak
-----
```

```
-----
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: 2
모든 덱의 종류:
1. 차원문 덱
2. 요정 색도자 덱
3. 달콤술사 전사 덱
4. 아르카나 폭파단 덱
-----
보고 싶은 덱의 번호를 입력하세요: 2
덱 이름: 요정 색도자 덱
승률: 14.1%
평균 등수: 4.38
Top 4 비율: 52.2%
사용 비율: 0.48
티어: B
구성 챔피언: Kalista / Rakan / Milio
추천 아이템: bow / giant belt
-----
```

```
-----
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: 2
모든 덱의 종류:
1. 차원문 덱
2. 요정 색도자 덱
3. 달콤술사 전사 덱
4. 아르카나 폭파단 덱
-----
보고 싶은 덱의 번호를 입력하세요: 3
덱 이름: 달콤술사 전사 덱
승률: 16.5%
평균 등수: 4.43
Top 4 비율: 50.1%
사용 비율: 0.3
티어: B
구성 챔피언: Gwen / Fiora / Rakan
추천 아이템: sword / staff / giant belt
-----
```

```
-----
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: 2
모든 덱의 종류:
1. 차원문 덱
2. 요정 색도자 덱
3. 달콤술사 전사 덱
4. 아르카나 폭파단 덱
-----
보고 싶은 덱의 번호를 입력하세요: 4
덱 이름: 아르카나 폭파단 덱
승률: 13%
평균 등수: 4.66
Top 4 비율: 46.4%
사용 비율: 0.17
티어: C
구성 챔피언: Varus / TahmKench / Xerath
추천 아이템: sword / mana essence / chain vest / magic cloak
-----
```


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: 2

모든 덱의 종류:

1. 차원문 덱
2. 요정 색도자 덱
3. 달콤술사 전사 덱
4. 아르카나 폭파단 덱

보고 싶은 덱의 번호를 입력하세요: 5

잘못된 입력입니다. 다시 시도하세요.

③ 에 대한 결과

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: A

구성 챔피언: Ryze / Taric / TahmKench

추천 아이템: staff / mana essence / chain vest / magic cloak

추천받고 싶은 챔피언 이름을 입력하세요: Taric

챔피언 Taric과(와) 관련된 덱:

덱 이름: 차원문 덱

승률: 15.3%

평균 등수: 4.31

Top 4 비율: 52.5%

사용 비율: 0.3

티어: A

구성 챔피언: Ryze / Taric / TahmKench

추천 아이템: staff / mana essence / chain vest / magic cloak

추천받고 싶은 챔피언 이름을 입력하세요: TahmKench

챔피언 TahmKench과(와) 관련된 덱:

덱 이름: 차원문 덱

승률: 15.3%

평균 등수: 4.31

Top 4 비율: 52.5%

사용 비율: 0.3

티어: A

구성 챔피언: Ryze / Taric / TahmKench

추천 아이템: staff / mana essence / chain vest / magic cloak

덱 이름: 아르카나 폭파단 덱

승률: 13%

평균 등수: 4.66

Top 4 비율: 46.4%

사용 비율: 0.17

티어: C

구성 챔피언: Varus / TahmKench / Xerath

추천 아이템: sword / mana essence / chain vest / magic cloak

추천받고 싶은 챔피언 이름을 입력하세요: Gwen

챔피언 Gwen과(와) 관련된 덱:

덱 이름: 달콤술사 전사 덱

승률: 16.5%

평균 등수: 4.43

Top 4 비율: 50.1%

사용 비율: 0.3

티어: B

구성 챔피언: Gwen / Fiora / Rakan

추천 아이템: sword / staff / giant belt

추천받고 싶은 챔피언 이름을 입력하세요: Fiora

챔피언 Fiora과(와) 관련된 덱:

덱 이름: 달콤술사 전사 덱

승률: 16.5%

평균 등수: 4.43

Top 4 비율: 50.1%

사용 비율: 0.3

티어: B

구성 챔피언: Gwen / Fiora / Rakan

추천 아이템: sword / staff / giant belt

```
추천받고 싶은 챔피언 이름을 입력하세요: Rakan
챔피언 Rakan과(와) 관련된 덱:
덱 이름: 요정 색도자 덱
승률: 14.1%
평균 등수: 4.38
Top 4 비율: 52.2%
사용 비율: 0.48
티어: B
구성 챔피언: Kalista / Rakan / Milio
추천 아이템: bow / giant belt
-----
덱 이름: 달콤술사 전사 덱
승률: 16.5%
평균 등수: 4.43
Top 4 비율: 50.1%
사용 비율: 0.3
티어: B
구성 챔피언: Gwen / Fiona / Rakan
추천 아이템: sword / staff / giant belt
-----
```

```
추천받고 싶은 챔피언 이름을 입력하세요: Varus
챔피언 Varus과(와) 관련된 덱:
덱 이름: 아르카나 폭파단 덱
승률: 13%
평균 등수: 4.66
Top 4 비율: 46.4%
사용 비율: 0.17
티어: C
구성 챔피언: Varus / TahmKench / Xerath
추천 아이템: sword / mana essence / chain vest / magic cloak
-----
```

```
추천받고 싶은 챔피언 이름을 입력하세요: Xerath
챔피언 Xerath과(와) 관련된 덱:
덱 이름: 아르카나 폭파단 덱
승률: 13%
평균 등수: 4.66
Top 4 비율: 46.4%
사용 비율: 0.17
티어: C
구성 챔피언: Varus / TahmKench / Xerath
추천 아이템: sword / mana essence / chain vest / magic cloak
-----
```

```
추천받고 싶은 챔피언 이름을 입력하세요: Kalista
챔피언 Kalista과(와) 관련된 덱:
덱 이름: 요정 색도자 덱
승률: 14.1%
평균 등수: 4.38
Top 4 비율: 52.2%
사용 비율: 0.48
티어: B
구성 챔피언: Kalista / Rakan / Milio
추천 아이템: bow / giant belt
-----
```

```
추천받고 싶은 챔피언 이름을 입력하세요: Milio
챔피언 Milio과(와) 관련된 덱:
덱 이름: 요정 색도자 덱
승률: 14.1%
평균 등수: 4.38
Top 4 비율: 52.2%
사용 비율: 0.48
티어: B
구성 챔피언: Kalista / Rakan / Milio
추천 아이템: bow / giant belt
-----
```

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. 종료
선택: 3
추천받고 싶은 챔피언 이름을 입력하세요: dd
챔피언 dd과(와) 관련된 덱:
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 에 대한 결과 - 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. 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. 종료
선택: 4
1. 챔피언에 맞는 아이템 추천
2. 아이템에 맞는 챔피언 추천
선택: 1
추천받고 싶은 챔피언 이름을 입력하세요: dd
해당 챔피언에 대한 아이템 정보가 없습니다.

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: 2
추천받고 싶은 아이템 이름을 입력하세요: d
해당 아이템에 대한 챔피언 정보가 없습니다.

⑤ 에 대한 결과

```
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 : 5
프로그램을 종료합니다.
PS C:\Users\1\Desktop\c++\project\CPP2409-P>
```

5. 결과 및 결론

5-1. 프로젝트 결과 : 이 프로젝트를 통해 챔피언과 아이템, 덱의 정보를 기반으로 간단한 추천 시스템을 구현했습니다. 사용자가 원하는 챔피언이나 아이템을 입력하면, 적절한 덱이나 조합 아이템을 추천할 수 있게 되었습니다. 덱의 성능에 따라 최적의 덱을 선정하고 이를 바탕으로 아이템을 추천하는 기능도 구현했습니다.

5-2. 느낀 점 : 9주차에 배운 클래스는 프로젝트에서 가장 중요한 부분이었습니다. 덱, 아이템, 챔피언과 같은 개념을 클래스로 정의함으로써 코드의 재사용성과 가독성이 크게 향상되었습니다. 클래스는 데이터와 기능을 하나로 묶어 주었고, 이를 통해 프로그램을 더 체계적이고 직관적으로 작성할 수 있었습니다. 10주차에 배운 벡터는 이 프로젝트에서 매우 유용하게 활용되었습니다. 덱, 챔피언, 아이템과 같은 데이터를 효율적으로 저장하고 접근하는 데 벡터의 기능이 큰 역할을 했습니다. 벡터를 사용하여 데이터를 순회하고, 필요한 정보들을 쉽게 추가하고 관리할 수 있었습니다. 벡터의 동적 배열 특성을 이해함으로써 데이터를 효율적으로 처리하는 법을 배울 수 있었습니다. 하지만 아쉽게도 처음 구상한 것과 다르게 흘러간 부분도 꽤 있었습니다. 파이썬에서는 데이터 파일을 바로 읽을 수 있는 방법을 알고 있었기에 c++에서도 비슷하게 할 수 있을 것이라고 생각했는데, 쉽게 되지는 않았습니다. 그래서 결국 데이터 값을 실수형으로 집어 넣었던 것이 아쉽게 느껴집니다. 아직 프로젝트가 중간 과정이기에 추후에 더 보완하고 싶습니다.