

C++ 프로젝트 및 실습

메뉴 추천 어플

진척 보고서 #3

제출일자: 2024-12-15

제출자명: 임연우

제출자학번: 234106

1. 프로젝트 목표 (16 pt)

1) 배경 및 필요성 (14 pt)

(본문: 12pt)

매 끼니마다 무엇을 먹어야 할지, 어디서 먹어야 할지 라는 고민을 수차례 반복함. 이 문제는 혼자일 때나, 여러 명일 때나, 쉽게 해결되지 않는 문제임. 이 문제를 해결하기 위해서 음식과 식당을 추천해주는 프로그램이 필요함.

2) 프로젝트 목표

무엇을 먹고싶은지 사용자의 니즈를 파악해 음식 메뉴를 추천하고, 사용자에게 적합한 식당을 추천하는 프로그램을 만드는 것

3) 차별점

기존 프로그램들은 식당과 음식들을 그저 나열하는데 그침. 여기서 나아가 사용자 선호도 우선의 순위 목록을 나열함으로써, 메뉴를 선택함에 도움을 주는 차별점을 가짐

2. 기능 계획

1) 전남대 근처 식당들의 행렬 지도 표시

- 설명 : 간단하게 표시된 형렬의 맵을 구현해 간략한 위치 확인

(1) 세부 기능 1 : 맵의 표시된 위치 확인

- 설명 : 맵의 좌표를 검색하면 무엇이 있는지 출력

(2) 세부 기능 2 : 내 위치 표시

- 설명 : 내 위치를 입력 후 배열 지도에 표시

2) 음식점 정보 검색

- 설명 : 음식점의 목록과 특정 음식점의 정보 열람

3) 먹고 싶은 음식 추천

- 설명 : 먹고 싶은 음식의 특징을 입력하면 그에 해당하는 음식을 추천해줌

4) 먹고 싶은 음식의 식당 추천

- 설명 : 먹고 싶은 음식을 입력하면 그 음식을 파는 가게를 알려줌

(1) 세부 기능 1: 음식점을 사용자 지정 기준으로 나열

- 설명 : 가격, 거리, 평점, 등의 기준을 설정해서 사용자가 원하는 식당을
고를 수 있도록 도움을 줌

3. 진척사항

1) 기능 구현

(1) 식당 지도 표시

```
while(true){
    cout << "-----" << endl;
    cout << "[기능 메뉴]" << endl
        << "1. 지도 보기" << endl
        << "2. 식당 정보" << endl
        << "3. 음식 추천받기" << endl
        << "4. 식당 추천받기" << endl
        << "5. 종료" << endl
        << "당신의 입력: ";

    cin >> menu;
    cout << endl;

    switch (menu) {
        case 1:
            user_locate = InputUserLocate(max_x, max_y);

            PrintMap(max_x, max_y, user_locate, restaurantList);

            cout << "원하는 추가 기능 선택" << endl
                << "1. 사용자 위치 재설정하기" << endl
                << "2. 좌표 검색하기" << endl
                << "3. 나가기" << endl
                << "당신의 입력: " ;

            while (true) {
                cin >> first_num;
                if (first_num != 1 && first_num != 2 && first_num != 3 ){
                    continue;
                }
                cout << endl;
                break;
            }
        }
    }
```

- 입력: max_x, max_y : 지도의 최대 크기

first_num : 사용자의 추가 기능 입력을 저장

- 결과: 기능 메뉴에서 하고자하는 기능 선택 - 1번 선택

사용자의 위치를 입력받고 지도 표시

추가적으로 필요한 기능을 선택

- 설명: InputUserLocate(max_x, max_y) 함수로 위치를 입력받고,
PrintMap(max_x, max_y, user_locate, restaurantList) 함수로 지도 그리기
first_num으로 추가 기능에 대한 입력을 받고 입력이 적절한지 if문으로 확인
적절하지 않으면 while문으로 적절할 때까지 반복
- 적용된 배운 내용: 반복문, 조건문, 스위치문, 함수,

```
if (first_num == 1) {  
    user_locate = InputUserLocate(max_x, max_y);  
    break;  
}
```

- 입력: max_x, max_y : 지도의 최대 크기
first_num : 사용자의 추가 기능 입력을 저장
- 결과: 사용자 위치 재설정
- 설명: first_num == 1 이면 실행
InputUserLocate(max_x, max_y) 함수를 재 호출하여 새로 입력
- 적용된 내용: 조건문, 함수

```

else if (first_num == 2) {
    vector<int> searchLocate;
    found = false;

    cout << "정보를 원하는 ";
    searchLocate = InputUserLocate(max_x, max_y);
    for (const auto& restaurant : restaurantList) {
        if (searchLocate[0] == restaurant.getLocation()[0],
            searchLocate[1] == restaurant.getLocation()[1]) {
            restaurant.Print_infor();
            found = true;
        }
    }
    if (found == false) {
        cout << "이 좌표에는 식당이 없습니다." << endl;
    }
    break;
}

```

- 입력: max_x, max_y : 지도의 최대 크기

first_num : 사용자의 추가 기능 입력을 저장

restaurantList : 모든 식당이 저장된 벡터<Restaurant> 배열

restaurant : 식당의 이름이 저장됨

- 결과: 좌표를 입력하여 해당 좌표에 어떤 식당이 있는지 확인

- 설명: first_num == 2 이면 실행

InputUserLocate(max_x, max_y) 함수로 좌표를 입력받아 해당 좌표에 어떤 식당이 있는지 for문으로 모든 식당을 비교하며 확인

식당이 있으면 Print_infor() 함수로 식당 정보 출력하고 found에 true저장

식당이 없으면(found가 false이면) 식당이 없음을 알림

- 적용된 내용: 조건문, 반복문, 배열, 함수, 포인터

```

else if (first_num == 3) {
    cout << "[기능메뉴 4. 지도보기]를 종료합니다" << endl;
    break;
}
else {
    cout << "잘못된 입력입니다" << endl;
    continue;
}

```

- 입력: first_num : 사용자의 추가 기능 입력을 저장

- 결과: first_num == 3이면 지도보기 종료

그 외 잘못된 입력이면 다시 돌아가기

- 설명: 3을 입력하면 break로 종료

그 외 입력은 잘못된 입력이므로 continue로 돌아가기

{ vector<int> InputUserLocate(int max_x, int max_y) 함수

```

vector<int> InputUserLocate(int max_x, int max_y) {
    int user_locateX = 0, user_locateY = 0;

    cout << "위치(x, y)를 입력하십시오(0~10): " << endl;
    cin >> user_locateX >> user_locateY;
    if ( 0 <= user_locateX && user_locateX <= max_x &&
        0 <= user_locateY && user_locateY <= max_y ) {
        return {user_locateX, user_locateY};
    }
    else {
        cout << "잘못된 입력입니다. 다시 입력하십시오." << endl;
        return InputUserLocate(max_x, max_y);
    }
}

```

- 입력: max_x, max_y : 지도의 최대 크기

- 결과: 사용자 입력으로 위치를 반환

- 설명: 좌표를 입력받아 user_locateX, user_locateY 에 저장하고,

좌표에 이상이 없으면 반환

이상이 있으면 함수 재귀호출하여 다시 입력

- 적용된 내용: 조건문, 함수

void PrintMap(int max_x, int max_y, vector<int> user_locate, vector<Restaurant>& restaurantList) 함수

```
void PrintMap(int max_x, int max_y, vector<int> user_locate, vector<Restaurant>& restaurantList){
    for (int i = 0; i < max_y; i++) {
        cout << "-----" << endl;
        for (int j = 0; j < max_x; j++){
            bool found = false;

            cout << "|" ;
            for (const auto& restaurant : restaurantList){
                if (restaurant.getLocation()[0] == j && restaurant.getLocation()[1] == i){
                    found = true;
                    break;
                }
            }
            if (found == true && user_locate[0] == j && user_locate[1] == i) {
                cout << "L&U";
            }
            else if (found == true){
                cout << " L ";
            }
            else if (user_locate[0] == j && user_locate[1] == i) {
                cout << " U ";
            }
            else {
                cout << "   ";
            }
        }
        cout << "|" << endl;
    }
    cout << "-----" << endl;
}
```

- 입력: max_x, max_y : 지도의 최대 크기

first_num : 사용자의 추가 기능 입력을 저장

restaurantList : 모든 식당이 저장된 벡터<Restaurant> 배열

restaurant : 식당의 이름이 저장됨

- 결과: 지도와 지도의 구성요소를 출력

- 설명: 반복문을 통해 지도의 틀을 출력하고,

현재 출력 중인 좌표와 식당의 좌표를 비교해 식당의 위치를 표시

사용자 위치와 식당의 위치가 겹칠 때, 식당만 있을 때, 사용자만 있을 때를 구별하여 표시

- 적용된 내용: 반복문, 조건문, 배열, 함수, 포인터

(2) 음식점 정보 검색

```
int main(){

    int num = 1;
    int menu;
    int restauNum;
    int foodNum;
    bool foodFound = false;
    string desiredFeature;

    while(num){
        cout << "-----" << endl;
        cout << "[기능 메뉴]" << endl
            << "1. 식당 정보" << endl
            << "2. 음식 추천받기" << endl
            << "3. 식당 추천받기" << endl
            << "4. 종료" << endl
            << "실행할 기능을 선택하세요: ";

        cin >> menu;
        cout << endl;
    }
```

- 입력: num : while문의 반복을 조절
menu : 이 다음의 스위치문의 분기 설정
- 결과: 기능 메뉴 인터페이스 출력
- 설명: while문으로 반복하면서 프로그램 실행
- 적용된 배운 내용: 반복문

case 1 실행 - 특정 음식점 정보 검색

```
switch (menu) {  
    //식당 정보  
    case 1:  
        DisplayRestaurantList();  
  
        cout << "어떤 식당의 정보를 원하십니까?" << endl  
            << "번호를 입력하십시오: ";  
  
        cin >> restauNum;  
        cout << endl;  
  
        DisplayRestaurant(restauNum);  
  
        cout << "음식 정보를 원하십니까?" << endl  
            << "1: 1번 음식" << endl  
            << "2: 2번 음식" << endl  
            << "3: 종료" << endl  
            << "원하는 서비스를 선택하세요: ";  
  
        cin >> foodNum;  
        cout << endl;  
  
        DisplayFoodInfo(restauNum, foodNum);  
  
        break;  
}
```

- 입력: menu : 스위치문 케이스 설정

restauNum : 원하는 식당에 해당하는 번호

foodNum : 원하는 음식에 해당하는 번호

- 결과: 식당 목록 출력

원하는 식당이 있는지 질문 출력

식당을 선택한 뒤 그 식당 정보 출력

음식의 정보를 원하는지, 어떤 음식을 원하는지 질문 출력

원하는 음식의 정보 출력

설명: void DisplayRestaurantList() 함수를 호출해 식당의 목록을 출력

void DisplayRestaurant(int restauNum) 함수를 호출해 식당의 정보를 출력

void DisplayFoodInfo(int restauNum, int foodNum) 함수를 호출해 음식의 정보를
출력

- 적용된 배운 내용: 스위치, 함수

{ } void DisplayRestaurantList() 함수 구현

```
//식당의 목록을 출력하는 함수
void DisplayRestaurantList() {
    int num = 1;

    cout << "식당 목록:" << endl;
    for (const auto& restaurant : restaurantList) {
        cout << num << ". " << restaurant.GetRestauName() << endl;
        num += 1;
    }
    cout << endl;
}
```

- 입력: num : 나열되는 식당들에 번호를 부여하여 같이 출력됨

restaurantList : 모든 식당이 저장된 벡터<Restaurant> 배열

restaurant : 식당의 이름이 저장되어 출력됨

- 결과: 반복문으로 리스트의 모든 식당들의 이름이 번호가 매겨져 함께 출력

- 설명: vector<Restaurant> 타입의 리스트의 데이터가 restaurant에 저장

반복할 때마다 +1 되는 num 출력

restaurant가 Restaurant 클래스이므로, 해당 클래스 안에 정의된

string GetRestaurantName() 함수를 호출하여 이름 출력

- 적용된 배운 내용: 반복문, 배열, 함수, 포인터

{{ }} vector<Restaurant> restaurantList 벡터 배열

```
//식당 목록 벡터
vector<Restaurant> restaurantList = {woulmidang, miss420, speedbanjum,
joseon, shinsacheon};
```

- 적용된 배운 내용: 배열, 벡터

(3) 구현한 기능 이름 : 먹고싶은 음식 추천

case 2 실행 - 먹고 싶은 음식 추천받기

```
//음식 추천받기
case 2:
    DisplayFeatures();
    cout << "원하는 음식의 특징을 입력하십시오: " ;
    cin >> desiredFeature;
    cout << endl;

    foodFound = false;
    for (const auto& restaurant : restaurantList) {
        foodFound = restaurant.RecommendFood(desiredFeature) ;
    }

    if (!foodFound) {
        cout << "특징에 맞는 음식이 없습니다." << endl;
    }

    break;
```

- 입력: desiredFeature : string 타입의 특징을 입력하여 저장
foodFound : 특징에 맞는 음식이 있는지 확인하는 변수
- 결과: 특징 목록을 출력
특징 목록 중 원하는 특징을 입력, 저장
음식점을 돌며 원하는 음식이 있으면 이름을 출력
원하는 음식이 없으면 "맞는 음식이 없음" 출력
- 설명: void DisplayFeatures() 함수를 호출해 모든 특징 목록을 출력
원하는 특징을 입력받아 desiredFeature에 저장
restaurantList에서 식당을 추출해 restaurant에 저장
restaurant는 Restaurant클래스이므로 해당 클래스에 정의된
bool RecommendFood(const string& desiredFeature) 함수를 호출
함수 호출의 결과로 이름을 출력하고 함수 반환값을 foodFound에 저장
원하는 음식이 없으면(= foodFound가 false이면) "맞는 음식이 없음" 출력
- 적용된 배운 내용: 스위치, 반복문, 조건문, 함수, 포인터

{ } void DisplayFeatures()함수

```
//특징 목록을 출력하는 함수
void DisplayFeatures() {
    cout << "모든 특징 목록:" << endl;
    for (const auto& feature : featureList) {
        cout << feature << " ";
    }
    cout << endl;
}
```

- 입력: featureList : 모든 특징 목록

feature의 데이터를 하나씩 저장

- 결과: 특징 목록에 들어있는 모든 특징을 출력

- 설명: vector<string> 타입의 특징 목록을 for문으로 하나씩 추출하여 출력

- 적용된 배운 내용: 반복문, 배열, 함수, 포인터

{{ }} featureList 특징 목록 벡터 배열

```
//특징 목록 벡터
vector<string> featureList = {"korean", "chinese", "japanese",
                             "vietnamese", "italian", "spicy", "sour"};
```

- 적용된 배운 내용: 배열, 벡터

{ } bool RecommendFood(const string& desiredFeature) 함수

```
//음식이 원하는 음식인지 확인하고, 맞으면 이름을 출력하는 함수
bool RecommendFood(const string& desiredFeature) const {
    bool foodFound = false;

    if (food1.HasFeature(desiredFeature)) {
        cout << food1.GetFoodName() << endl;
        foodFound = true;
    }

    if (food2.HasFeature(desiredFeature)) {
        cout << food2.GetFoodName() << endl;
        foodFound = true;
    }

    return foodFound;
}
```

- 입력: desiredFeature : 입력받은 string 타입의 원하는 특징

foodFound : 원하는 음식이 있는지 없는지 확인하는 값

-결과: 원하는 특징의 음식이 있으면 그 음식 이름을 출력

함수 종료 시 foodFound 값 true 반환

원하는 특징의 음식이 없으면 종료 시 false 반환

- 설명: 현재 확인 중인 식당에 파는 음식 food1, food2이 Food 클래스이므로

해당 클래스에 정의된 bool HasFeature(const string& desiredFeature) 함수를

호출

함수 호출의 반환값이 true이면 Food 클래스에 정의된 string GetFoodName() 함수를

호출하여 이름 출력

food1, food2 모두 진행하고 if문이 실행됐으면 true 반환

- 적용된 배운 내용: 조건문, 함수, 포인터

{{ }} bool HasFeature(const string& desiredFeature) 함수

```
//원하는 특징을 음식이 가지고 있는지 확인하는 함수
bool HasFeature(const string& desiredFeature) const {
    for (const auto& foodFeature : features) {
        if (desiredFeature == foodFeature) {
            return true;
        }
    }
    return false;
}
```

- 입력: desiredFeature : 입력받은 string 타입의 원하는 특징
features : Food 클래스에 내재된 vector<string> 타입 벡터 배열: 특징을 저장
foodFeature : features에서 하나씩 저장받음
- 결과: desiredFeature과 foodFeature가 같으면 true 반환, 다르면 false 반환
- 설명: 반복문으로 돌면서 음식이 가지고 있는 모든 feature를 desiredFeature과 비교
비교 값에 따라 bool 값 반환
- 적용된 배운 내용: 반복문, 조건문, 배열, 함수, 포인터

{{ }} string GetFoodName()

```
//음식 이름을 반환하는 함수
string GetFoodName() const {
    return name ;
}
```

- 음식 이름을 반환
- 적용된 배운 내용: 함수

(4) 구현한 기능 이름 : 먹고 싶은 음식의 식당 추천

case 3 실행 - 식당 추천

```
case 3: //식당 추천
    cout << "어떤 식당을 추천받고 싶으십니까?" << endl
        << "1. 특정 음식을 파는 식당" << endl
        << "2. 원하는 특징의 음식을 파는 식당" << endl
        << "당신의 입력: ";
    cin >> first_num ;
    cout << endl;
```

- 설명: case 3의 목록 설명

원하는 목록을 first_num에 저장

first_num에 따라 목록 실행

1이면 정확한 음식을 가진 식당을 찾아줌

2면 특정 스타일의 음식을 가진 식당을 찾아줌

first_num = 1일 때

```
if (first_num == 1) {
    cout << "원하는 음식을 입력하십시오" << endl
    |   << "당신의 입력: ";
    cin >> first_string;
    cout << endl;

    // 특정 음식을 찾는 식당 검색
    food_found = false;
    for (const auto& restaurant : restaurantList) {
        if (restaurant.HasRecommendFood(first_string)) {
            cout << restaurant.GetRestauName() << "에서 " << first_string
            |   << "을(를) 판매합니다." << endl;
            food_found = true;
        }
    }
    if (!food_found) {
        cout << "해당 음식을 판매하는 식당이 없습니다." << endl;
    }
}
```

- 입력: first_num : case3의 세부목록 지정

first_string : 이곳에 음식의 이름을 저장

restaurantList : 모든 식당이 저장된 벡터<Restaurant> 배열

restaurant : for문으로 식당을 하나씩 입력받음

food_found : 식당을 돌면서 일치하는 음식이 있으면 true 반환

- 결과: 찾는 음식이 있으면 판매하는 식당 출력

음식이 없으면 없다는 문장 출력

- 설명: 음식 이름을 입력받아서 각 식당에 HasRecommendFood 함수 호출하여 bool값 반환

참이면 식당 이름과 함께 안내 출력하고 food_found에 참 입력

찾는 음식이 없으면 food_found가 그대로 false이므로 없다는 안내 문구 출력

- 적용된 배운 내용: 조건문, 배열, 함수, 포인터

first_num = 2일 때

```
} else if (first_num == 2) { // 원하는 특징의 음식을 파는 식당 추천
    recommendedRestaurantList.clear();

    DisplayFeatures();
    cout << "원하시는 음식의 특징을 입력하십시오:" << endl
    | << "당신의 입력: ";
    cin >> first_string;
    cout << endl;

    // 특징을 기반으로 식당 검색
    food_found = false;
    for (const auto& restaurant : restaurantList) {
        if (restaurant.HasRecommendFood(first_string)) {
            recommendedRestaurantList.push_back(restaurant.GetRestauName());
            cout << restaurant.GetRestauName() << endl;
            food_found = true;
        }
    }
    cout << "에서 " << first_string << " 특징을 가진 음식을 판매합니다." << endl;
```

```
else {
    cout << "해당 특징의 음식을 판매하는 식당이 없습니다." << endl;
}
```

- 입력: first_num : case3의 세부목록 지정

first_string : 이곳에 음식의 이름의 특징을 저장

restaurantList : 모든 식당이 저장된 벡터<Restaurant> 배열

restaurant : for문으로 식당을 하나씩 입력받음

food_found : 식당을 돌면서 일치하는 음식이 있으면 true 반환

- 결과: 추천 음식 리스트 초기화

DisplayFeatures 함수로 입력가능한 특징 목록 출력

HasRecommendFood 함수로 특정 음식을 찾으면 추천 식당 리스트에 추가하고

식당 이름 출력, food_found = true 입력

- 설명: clear를 이용해서 추천 식당 리스트를 초기화

DisplayFeatures 함수로 특징 목록 출력

for문으로 돌면서 restaurantList의 모든 식당을 순회

HasRecommendFood 함수로 음식이 있는지 확인하고

있으면 recommendedRestaurantList 추천 식당 목록에 추가

식당 이름 출력

없으면 없다는 설명 출력

- 적용된 배운 내용: 조건문, 반복문, 배열, 함수, 포인터

food_found == true, 찾는 음식이 있을 때

```
if (food_found){
    cout << "새로운 기준에 따라 목록을 정렬하시겠습니까?" << endl
        << "1. 별점" << endl
        << "2. 가격" << endl
        << "3. 거리" << endl
        << "4. 아니오" << endl
        << "당신의 입력: ";
    cin >> align;

    if (align == 1) cout << "별점이 높은 순으로 정렬합니다." << endl;
    else if (align == 2) cout << "평균 가격이 싼 순으로 정렬합니다." << endl;
    else if (align == 3) cout << "거리가 가까운 순으로 정렬합니다." << endl;
    else if (align == 4) {
        cout << "기능 메뉴로 돌아갑니다." << endl;
        break;
    }
    else {
        cout << "잘못된 입력입니다." << endl;
        break;
    }

    PrintSortedRestaurants(recommendedRestaurantList, align, restaurantList, user_locate);

    break;
}
```

- 입력: food_found : 식당을 돌면서 일치하는 음식이 있으면 true 반환

align : 목록 정렬 방식 선택을 위한 입력값을 저장

- 결과: 원하는 정렬 방식을 선택하고

정렬된 목록을 출력

- 설명: food_found가 참이면 정렬 방식 목록을 안내

align을 입력 받고 PrintSortedRestaurants 함수 호출

- 적용된 배운 내용: 조건문, 함수

```
{ } void PrintSortedRestaurants(vector<string>& recommendedRestaurantList,  
  
int align, const vector<Restaurant>& restaurantList, vector<int> user_locate)
```

```
void PrintSortedRestaurants(vector<string>& recommendedRestaurantList, int align,  
const vector<Restaurant>& restaurantList, vector<int> user_locate) {  
    if (recommendedRestaurantList.empty()) {  
        return;  
    }  
  
    int bestIndex = 0;  
  
    for (int i = 1; i < recommendedRestaurantList.size(); ++i) {  
        const string& current = recommendedRestaurantList[i];  
        const string& best = recommendedRestaurantList[bestIndex];  
  
        if (align == 1) { // 별점 기준  
            float currentRating = 0, bestRating = 0;  
            for (const auto& restaurant : restaurantList) {  
                if (restaurant.GetRestauName() == current) currentRating = restaurant.getRating();  
                if (restaurant.GetRestauName() == best) bestRating = restaurant.getRating();  
            }  
            if (currentRating > bestRating) bestIndex = i;  
        }  
    }  
}
```

align = 1일 때

- 입력: recommendedRestaurantList : 추천 식당 리스트

align : 어떤 기준으로 정렬할 것인지에 대한 방식 선택

restaurantList : 모든 식당이 저장된 리스트

user_locate : 사용자의 위치가 저장된 배열

bestIndex : 최고의 값을 가르키는 인덱스 저장

currentRating, bestRating : 반복하면서 최고를 찾기위해 비교하는 변수

- 결과: 모든 식당을 돌면서 기준에 대해 가장 좋은 인덱스를 찾아 저장

- 설명: 초기값(index 0)을 가장 좋은 식당으로 설정

for문으로 레스토랑 순회

현재 확인하고 있는 식당의 별점을 getRating 함수로 확인

현재 확인하고 있는 식당이 최고의 식당이면 bestIndex 갱신

- 적용된 배운 내용 : 조건문, 반복문, 함수

```

else if (align == 2) { // 평균 가격 기준
    double currentPrice = 0, bestPrice = 0;
    for (const auto& restaurant : restaurantList) {
        if (restaurant.GetRestauName() == current) {
            currentPrice = (restaurant.getFoodPrice(1) + restaurant.getFoodPrice(1)) / 2;
        }
        if (restaurant.GetRestauName() == best) {
            bestPrice = (restaurant.getFoodPrice(1) + restaurant.getFoodPrice(1)) / 2;
        }
    }
    if (currentPrice < bestPrice) bestIndex = i;
}

```

align = 2 일 때

- 입력: currentPrice, bestPrice : 반복하면서 최고를 찾기위해 비교하는 변수

bestIndex : 최고의 값을 가르키는 인덱스 저장

- 결과: 모든 식당을 돌면서 기준에 대해 가장 좋은 인덱스를 찾아 저장
- 설명: 초기값(index 0)을 가장 좋은 식당으로 설정

for문으로 레스토랑 순회

현재 확인하고 있는 식당의 음식 가격을 getFoodPrice함수로 확인하고 평균을 냄

현재 확인하고 있는 식당이 최고의 식당이면 bestIndex 갱신

- 적용된 배운 내용 : 조건문, 반복문, 함수

```

else if (align == 3) { // 거리 기준
    int currentDist = 0, bestDist = 0;
    for (const auto& restaurant : restaurantList) {
        if (restaurant.GetRestauName() == current) {
            currentDist = (restaurant.getLocation()[0] - user_locate[0])^2 +
                           (restaurant.getLocation()[1] - user_locate[1])^2 ;
        }
        if (restaurant.GetRestauName() == best) {
            bestDist = (restaurant.getLocation()[0] - user_locate[0])^2 +
                       (restaurant.getLocation()[1] - user_locate[1])^2;
        }
    }
    if (currentDist < bestDist) bestIndex = i;
}
else {
    return;
}

```

align = 3일 때

- 입력: currentDist , bestDist : 반복하면서 최고를 찾기위해 비교하는 변수

user_locate : 사용자 위치가 저장된 배열

bestIndex : 최고의 값을 가르키는 인덱스 저장

- 결과: 모든 식당을 돌면서 기준에 대해 가장 좋은 인덱스를 찾아 저장
- 설명: 초기값(index 0)을 가장 좋은 식당으로 설정

for문으로 레스토랑 순회

현재 확인하고 있는 식당의 거리를 사용자가 입력한 현재 위치와 getLocation함수로 확인한 식당 위치를 피타고라스 공식으로 이용해 거리 구하기

현재 확인하고 있는 식당이 최고의 식당이면 bestIndex 갱신

- 적용된 배운 내용 : 조건문, 반복문, 배열, 함수

```

// 가장 우선순위가 높은 항목 출력
cout << "- " << recommendedRestaurantList[bestIndex] << endl;

// 해당 항목 제거
recommendedRestaurantList.erase(recommendedRestaurantList.begin() + bestIndex);

// 나머지 항목에 대해 재귀 호출
PrintSortedRestaurants(recommendedRestaurantList, align, restaurantList, user_locate

```

- 입력: bestIndex : 위에서 각 기준으로 구한 최고의 식당을 가르키는 인덱스

recommendedRestaurantList : 추천 식당 배열

- 결과: 최고의 식당을 차례대로 출력

- 설명: 최고의 식당을 출력

그 식당을 추천 식당 목록에서 삭제

남은 식당 중에서 PrintSortedRestaurants함수 호출

그 다음 최고의 식당 출력

추천 식당 목록이 빌 때 까지 출력

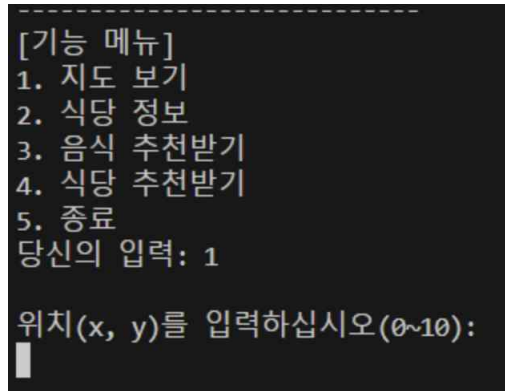
- 적용된 배운 내용 : 배열, 함수

2) 테스트 결과

(1) 지도 표시

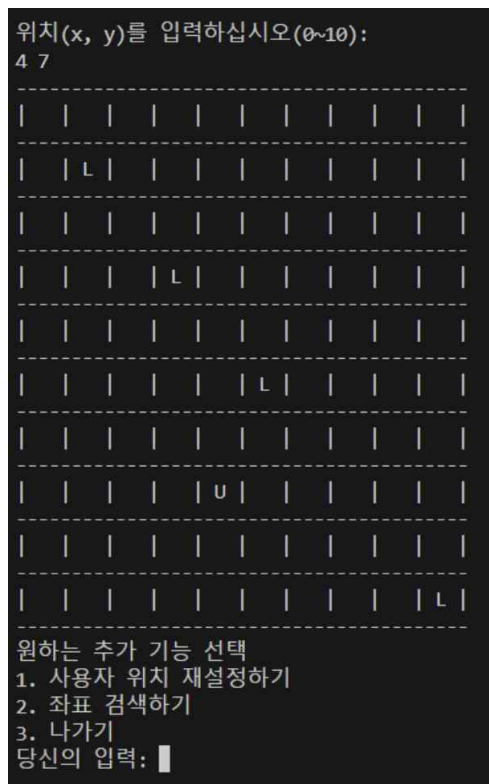
-설명: 기능메뉴에서 1번 선택

- 테스트 결과 스크린샷



- 설명: 위치 좌표 입력

- 테스트 결과 스크린샷



- 설명: 추가 기능 선택 - 1번

좌표를 입력 후 적용하고 기능 메뉴 복귀

- 테스트 결과 스크린샷

```
원하는 추가 기능 선택
1. 사용자 위치 재설정하기
2. 좌표 검색하기
3. 나가기
당신의 입력: 1

위치(x, y)를 입력하십시오(0~10):
4 8
-----
[기능 메뉴]
1. 지도 보기
2. 식당 정보
3. 음식 추천받기
4. 식당 추천받기
5. 종료
당신의 입력: █
```

- 설명: 추가 기능 선택 - 2번

지도를 확인하여 원하는 위치의 좌표 입력

입력된 좌표에 위치한 식당의 정보 출력

기능 메뉴 복귀

- 테스트 결과 스크린샷

```
원하는 추가 기능 선택
1. 사용자 위치 재설정하기
2. 좌표 검색하기
3. 나가기
당신의 입력: 2

정보를 원하는 위치(x, y)를 입력하십시오(0~10):
1 1
식당 이름: joseonjjambong
평점: 4.6
대표 메뉴: 1.joseonjjambong, 2.whitejjambong
위치: 1, 1

-----
[기능 메뉴]
1. 지도 보기
2. 식당 정보
3. 음식 추천받기
4. 식당 추천받기
5. 종료
당신의 입력: █
```

- 설명: 빈공간의 좌표를 입력

- 테스트 결과 스크린샷

```
원하는 추가 기능 선택
1. 사용자 위치 재설정하기
2. 좌표 검색하기
3. 나가기
당신의 입력: 2

정보를 원하는 위치(x, y)를 입력하십시오(0~10):
5 7
이 좌표에는 식당이 없습니다.
```

(2) 음식점 정보 검색

- 설명 : 프로그램 시작 시, 기능 메뉴 출력
- 테스트 결과 스크린샷

```
PS C:\CPP2409-P> & 'c:\Users\user2\.vscode\extensions\ms-vscode.cpptools-1.22
.11-win32-x64\debugAdapters\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-M
IEngine-In-4fwo51v1.1cf' '--stdout=Microsoft-MIEngine-Out-ni4irr2p.wag' '--std
err=Microsoft-MIEngine-Error-4t5ibk2z.kcv' '--pid=Microsoft-MIEngine-Pid-5whma
efv.ge2' '--dbgExe=C:\msys64\ucrt64\bin\gdb.exe' '--interpreter=mi'
-----
[기능 메뉴]
1. 식당 정보
2. 음식 추천받기
3. 식당 추천받기
4. 종료
실행할 기능을 선택하세요: █
```

- 설명 : 원하는 기능의 번호 입력 - 1번 입력

식당 목록을 출력하며 선택 대기

- 테스트 결과 스크린샷

```
실행할 기능을 선택하세요: 1

식당 목록:
1. woulmidang
2. misssaigon
3. speedbanjum
4. joseonjjambbong
5. shinsacheon

어떤 식당의 정보를 원하십니까?
번호를 입력하십시오: █
```

- 설명 : 원하는 식당 선택 - 1번

선택한 식당의 정보 출력

- 테스트 결과 스크린샷

```
번호를 입력하십시오: 1

식당 이름: woulmidang
평점: 5
대표 메뉴: 1.buncha, 2.pho
위치: 0, 0

음식 정보를 원하십니까?
1: 1번 음식
2: 2번 음식
3: 종료
원하는 서비스를 선택하세요: █
```

- 설명 : 원하는 식당 선택 - 2번

선택한 식당의 정보 출력

- 테스트 결과 스크린샷

```
어떤 식당의 정보를 원하십니까?
번호를 입력하십시오: 2

식당 이름: misssaigon
평점: 4.2
대표 메뉴: 1.friedrice, 2.pho
위치: 0, 0

음식 정보를 원하십니까?
1: 1번 음식
2: 2번 음식
3: 종료
원하는 서비스를 선택하세요: █
```

- 설명 : 원하는 서비스 선택 - 1번

1번 음식 설명 후 다시 기능 메뉴로 복귀

- 테스트 결과 스크린샷

```
원하는 서비스를 선택하세요: 1

이름: buncha
가격: 1.4
특징: Vietnamese sour noodle

-----
[기능 메뉴]
1. 식당 정보
2. 음식 추천받기
3. 식당 추천받기
4. 종료
실행할 기능을 선택하세요: █
```

- 설명 : 원하는 서비스 선택 - 2번

- 테스트 결과 스크린샷

```
원하는 서비스를 선택하세요: 2

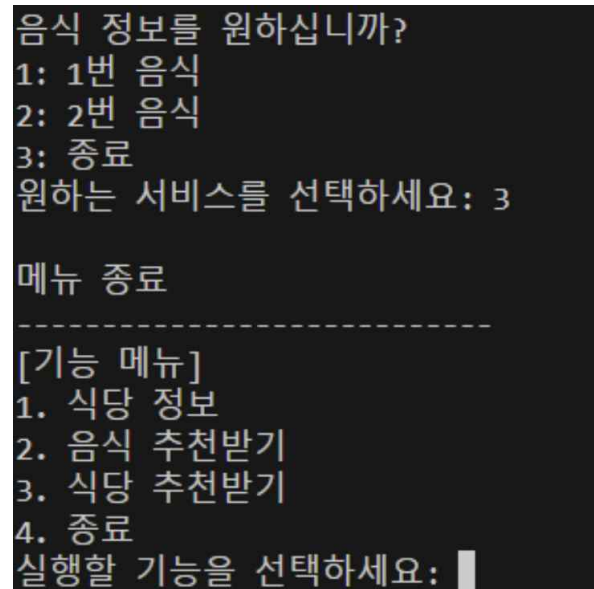
이름: pho
가격: 1.1
특징: Vietnamese beef noodle

-----
[기능 메뉴]
1. 식당 정보
2. 음식 추천받기
3. 식당 추천받기
4. 종료
실행할 기능을 선택하세요: █
```

- 설명 : 원하는 서비스 선택 - 3번

메뉴 종료 후 기능 메뉴 복귀

- 테스트 결과 스크린샷



```
음식 정보를 원하십니까?  
1: 1번 음식  
2: 2번 음식  
3: 종료  
원하는 서비스를 선택하세요: 3  
  
메뉴 종료  
-----  
[기능 메뉴]  
1. 식당 정보  
2. 음식 추천받기  
3. 식당 추천받기  
4. 종료  
실행할 기능을 선택하세요: |
```

(2) 먹고 싶은 음식 추천

- 설명 : 기능 2번 음식 추천 받기 선택

선택 가능한 특징 목록 출력

- 테스트 결과 스크린샷

```
[기능 메뉴]
1. 식당 정보
2. 음식 추천받기
3. 식당 추천받기
4. 종료
실행할 기능을 선택하세요: 2

모든 특징 목록:
korean chinese japanese vietnamese italian spicy sour
원하는 음식의 특징을 입력하십시오: 
```

- 설명 : 특징 입력 - korean

해당하는 음식 없으면, 다시 기능 메뉴 복귀

- 테스트 결과 스크린샷

```
모든 특징 목록:
korean chinese japanese vietnamese italian spicy sour
원하는 음식의 특징을 입력하십시오: korean

특징에 맞는 음식이 없습니다.
-----
[기능 메뉴]
1. 식당 정보
2. 음식 추천받기
3. 식당 추천받기
4. 종료
실행할 기능을 선택하세요: 
```


- 설명 : 특징 입력 - chinese

선택한 특징을 가지는 음식 출력

- 테스트 결과 스크린샷

```
원하는 음식의 특징을 입력하십시오: chinese  
  
jjajang  
jjambbong  
jjambbong  
whitejjambbong  
maratang  
guabaorou  
-----  
[기능 메뉴]  
1. 식당 정보  
2. 음식 추천받기  
3. 식당 추천받기  
4. 종료
```

(3) 먹고 싶은 음식의 식당 추천

현재 위치(x, y)를 입력하십시오(0~10):

설명 : 프로그램 시작하면 위치 입력

결과:

```
현재 위치(x, y)를 입력하십시오(0~10):
14 5
잘못된 입력입니다. 다시 입력하십시오.
현재 위치(x, y)를 입력하십시오(0~10):
-3 4
잘못된 입력입니다. 다시 입력하십시오.
현재 위치(x, y)를 입력하십시오(0~10):
4 6
-----
[기능 메뉴]
1. 식당 정보
2. 음식 추천받기
3. 식당 추천받기
4. 종료
당신의 입력: █
```

설명 : 기능 메뉴 입장

잘못 입력하면 다시 입력

기능 메뉴에서 3번 입력 (식당 추천 받기)

```
[기능 메뉴]
1. 식당 정보
2. 음식 추천받기
3. 식당 추천받기
4. 종료
당신의 입력: 3

어떤 식당을 추천받고 싶으십니까?
1. 특정 음식을 파는 식당
2. 원하는 특징의 음식을 파는 식당
당신의 입력: █
```

설명 : 확실한 특정 음식을 파는 식당을 원할 땐 1번

원하는 특징을 가진 음식을 파는 식당을 원할 땐 2번

1번을 눌렀을 때

```
어떤 식당을 추천받고 싶으십니까?
1. 특정 음식을 파는 식당
2. 원하는 특징의 음식을 파는 식당
당신의 입력: 1

원하는 음식을 입력하십시오
당신의 입력: pho

woulmidang에서 pho을(를) 판매합니다.
misssaigon에서 pho을(를) 판매합니다.
-----
[기능 메뉴]
1. 식당 정보
2. 음식 추천받기
3. 식당 추천받기
4. 종료
당신의 입력: █
```

설명: 특정 음식 'pho'를 입력했을 때, 'pho'를 파는 가게들을 출력

2번을 눌렀을 때

```
어떤 식당을 추천받고 싶으십니까?
1. 특정 음식을 파는 식당
2. 원하는 특징의 음식을 파는 식당
당신의 입력: 2

모든 특징 목록:
korean chinese japanese vietnamese italian spicy sour
원하시는 음식의 특징을 입력하십시오:
당신의 입력: chinese

speedbanjum
joseonjjambbong
shinsacheon
에서 chinese 특징을 가진 음식을 판매합니다.
새로운 기준에 따라 목록을 정렬하시겠습니까?
1. 별점
2. 가격
3. 거리
4. 아니오
당신의 입력: █
```

설명 : 입력할 수 있는 특징 목록을 보여주고 그 중 하나 입력

특징 'chinese'를 입력했을 때 chinese 특징을 가진 음식을 파는 식당을 찾아
출력

그리고 기준에 따라 정렬할 것인지 질문

1번-별점 을 눌렀을 때

```
에서 chinese 특징을 가진 음식을 판매합니다.  
새로운 기준에 따라 목록을 정렬하시겠습니까?  
1. 별점  
2. 가격  
3. 거리  
4. 아니오  
당신의 입력: 1  
별점이 높은 순으로 정렬합니다.  
- joseonjjambbong  
- speedbanjum  
- shinsacheon  
-----  
[기능 메뉴]  
1. 식당 정보  
2. 음식 추천받기  
3. 식당 추천받기  
4. 종료  
당신의 입력: █
```

설명 : 별점 순으로 정렬해서 출력

2번-가격 을 눌렀을 때

```
새로운 기준에 따라 목록을 정렬하시겠습니까?  
1. 별점  
2. 가격  
3. 거리  
4. 아니오  
당신의 입력: 2  
평균 가격이 싼 순으로 정렬합니다.  
- speedbanjum  
- joseonjjambbong  
- shinsacheon  
-----  
[기능 메뉴]  
1. 식당 정보  
2. 음식 추천받기  
3. 식당 추천받기  
4. 종료  
당신의 입력: █
```

설명 : 평균 가격이 싼 순으로 정렬해서 출력

3번-거리 을 눌렀을 때

```
새로운 기준에 따라 목록을 정렬하시겠습니까?
1. 별점
2. 가격
3. 거리
4. 아니오
당신의 입력: 3
거리가 가까운 순으로 정렬합니다.
- speedbanjum
- joseonjjambbong
- shinsacheon
-----
[기능 메뉴]
1. 식당 정보
2. 음식 추천받기
3. 식당 추천받기
4. 종료
당신의 입력: |
```

설명 : 거리가 가까운 순으로 정렬해서 출력

4. 계획 대비 변경 사항

1) 지도 관련 기능 합병

- 이전

기능1. 전남대 근처 음식점들을 행렬 맵으로 변환

기능2. 전남대 근처 음식점들을 행렬 맵으로 변환

기능3. 지도 표시

- 이후

기능1. 전남대 근처 식당들의 행렬 지도 표시와 세부 기능으로 분류

- 사유

: 한 기능에 포함될수 있는 작은 기능이기 때문에 이편이 간결함.

2) 가게 추천 기능 합병

- 이전

4) 먹고 싶은 음식의 식당 추천

- 설명 : 먹고 싶은 음식을 입력하면 그 음식을 파는 가게를 알려줌

5) 음식점을 가격, 거리, 평점 등의 기준으로 나열

- 설명 : 가격, 거리, 평점, 등의 기준을 설정해서 사용자가 원하는 식당을
고를 수 있도록 도움을 줌

- 이후

4) 먹고 싶은 음식의 식당 추천

- 설명 : 먹고 싶은 음식을 입력하면 그 음식을 파는 가게를 알려줌

(1) 세부 기능 1: 음식점을 사용자 지정 기준으로 나열

- 설명 : 가격, 거리, 평점, 등의 기준을 설정해서 사용자가 원하는 식당을
고를 수 있도록 도움을 줌

- 사유

본래의 의도가 별개의 기능이 아니라 연속된 기능임

이렇게 적는 것이 이해가 원활하고 간결함.

5. 프로젝트 일정

(진행한 작업과 진행 중인 작업 등을 표기)

업무		11/3	11/10	11/17
제안서 작성		완료			
기능2	세부기능1		완료		
기능3			완료		
기능4				진행 중	
기능1	세부기능1	진행 중			

업무		11/17	11/25	11/30	12/15
기능4		완료			
기능4	세부기능1		완료		
기능1	세부기능1			완료	
기능1	세부기능2			완료	