냉장고 현황판 프로그램

진척 보고서 3

제출일자: 2024.12.15

과목: C++프로그래밍및실습

학과: 소프트웨어공학과

학번: 214270

이름: 현경서

목차

1. 서론
   1) 배경 및 필요성
   2) 프로젝트 목표
   3) 차별점

2. 기능 계획
   1) 음식 목록 추가/제거
   2) 유통기한 임박 음식 출력
   3) 추천 메뉴 출력

3. 진척사항
   1) 추가 라이브러리 설치
   2) 기본 환경 구현
   3) 기능 구현
   4) 테스트 결과
   5) 개선 사항

4. 계획 대비 변경 사항
   1) 음식 목록 파일 확장자 변경
   2) 언어 설정 추가

5. 프로젝트 일정

1. 서론

1) 배경 및 필요성

냉장고에 있는 음식을 제때 처리하지 못하거나 어떤 음식이 있는지 잊어버리는 경우가 있어서 냉장고 현황판이 있어야겠다는 생각을 했다. 이번 프로젝트 통해서 이걸 프로그램으로 만들어보려 한다.

2) 프로젝트 목표

냉장고 안 음식 리스트를 csv 파일에 저장하고 이를 불러와서 유통기한이 임박한 음식과 추천 메뉴를 알려준다.

3) 차별점

냉장고 현황판을 프로그램으로 만든다면 아날로그로 사용할 때 보다 더 많은 기능을 사용할 수 있다. 구현하려는 기능 중 추천 메뉴 출력 기능은 아날로그 현황판에서는 사용할 수 없는 기능이다. 아날로그 현황판에서는 현황판의 음식을 보고 직접 검색해서 메뉴를 찾아내야 하지만 프로그램의 경우 알아서 메뉴를 알려준다. 냉장고에 남아있는 음식과 식사 메뉴 고민을 한번에 해결해 줄 수 있다.

또한 유통기한이 임박한 음식을 확인할 때 아날로그 현황판의 경우 직접 확인해야 하지만 프로그램의 경우 분류해서 알려주기 때문에 더 편하다고 할 수 있다.

2. 기능 계획

1) 음식 목록 추가/제거

음식 목록은 csv 파일에 저장해서 관리하며 프로그램은 csv 파일을 불러와서 목록을 변경한다. 새로운 음식을 추가하거나 이미 소비한 음식을 제거할 수 있다.

2) 유통기한 임박 음식 출력

csv 파일에는 음식의 이름, 개수, 유통기한을 저장한다. 유통기한이 얼마 남지 않은 식품들을 확인할 수 있다.

3) 메뉴 출력

csv 파일에 있는 음식들을 기반으로 추천 메뉴를 출력한다. openai api를 이용한다.

3. 진척사항

1) 추가 라이브러리 설치

① openai cpp

I. 설명

- https://github.com/olrea/openai-cpp
- C++에서 openai api를 사용하기 위한 라이브러리

II. 설치 및 적용

- include/openaicpp 폴더를 프로젝트 폴더에 복사한다.
- tasks.json 파일의 "args" 항목에 다음 옵션을 추가하여 헤더파일 경로를 설정한다.
  - "-I", "${fileDirname}₩₩include₩₩openai"

```
"-I",
"${fileDirname}\\include\\openai",
```

- food_manager.cpp 파일에서 #include "openai.hpp"로 헤더파일을 불러온다.

② curl

I. 설명

- https://curl.se/windows/
- openai cpp 라이브러리를 사용하려면 curl이 필요하다

II. 설치 및 적용

- curl for 64-bit를 다운받은 후 프로젝트 폴더에 압축을 푼다.
- tasks.json 파일의 "args" 항목에 다음 옵션을 추가하여 헤더파일 및 라이브러리 경로를 설정한다.
  - "-I", "${fileDirname}₩₩curl-8.11.0_1-win64-mingw₩₩include",
  - "-L", "${fileDirname}₩₩curl-8.11.0_1-win64-mingw₩₩lib"
  - "-lcurl"

```
"-I",
"${fileDirname}\\curl-8.11.0_1-win64-mingw\\include",
"-L",
"${fileDirname}\\curl-8.11.0_1-win64-mingw\\lib",
"-lcurl"
```

- curl-8.11.0_1-win64-mingw/bin 폴더 내의 libcurl-x64.dll 파일을 프로젝트 폴더에 복사한다. 실행 파일 디렉토리나 환경변수 path 경로에 libcurl-x64.dll 파일이 포함되어야 하기 때문이다. 사용자 입장에서 프로그램을 사용하기 위해 환경변수를 설정하는 것은 매우 불편한 일이므로 실행 파일 디렉토리에 파일을 복사한다.

③ nlohmann/json

I. 설명

- https://github.com/nlohmann/json
  api 요청이나 응답은 json으로 주고받기 때문에 json을 처리하기 위한 라이브러리가 필요하다.

II. 설치 및 적용

- openai cpp 라이브러리에 포함되어 있으므로 별도의 설치 및 적용이 필요하지 않다.
- json 객체 생성을 편하게 하기위해 다음 코드를 food_manager.cpp에 추가한다.

```
using json = nlohmann::json;
```

2) 기본 환경 구현

- 음식 정보 구조체

```
// 음식 정보 구조체
struct FoodInfo{
    string name;
    int count;
    int date; // YYYYMMDD
};
```

- 음식 목록 파일 경로 전역 상수로 선언

```
const string file_dir = "food_list.csv";
```

- 음식 목록 불러오기

```
vector<FoodInfo> ReadFoodListCsv(){
    vector<FoodInfo> food_list;
    FoodInfo food;
    string elem;
    ifstream file(file_dir);

    int i = 0; // while내 인덱스를 위한 변수
    while (file >> elem){
        // ',' 제거
        size_t pos = elem.find(',');
        if (pos != string::npos)
            elem.replace(pos, 1, "");
```

```
            // 요소 저장
            switch (i % 3){
                case 0:
                    food.name = elem;
                    break;
                case 1:
                    food.count = stoi(elem);
                    break;
                case 2:
                    food.date = stoi(elem);
                    food_list.push_back(food);
                    break;
            }
            i += 1;
        }
        file.close();

        return food_list;
    }
```

- csv파일을 읽어서 구조체 벡터에 저장하고 이를 반환한다.
- main() 함수에서 ReadFoodListCsv() 함수를 호출하여 csv의 내용을 벡터에 저장한다.
- 파일의 내용이 바뀔 때마다 파일을 다시 불러오는 대신, 벡터를 생성하여 파일과 벡터를 동시에 변경하는 방식으로 구현한다.

- 구조체 원소에 대한 사용자 입력 함수

```
string cinName(){
    string name;
    cout << "음식 이름: ";
    cin >> name;
    return name;
}
int cinCount(){
    int count;

    while(1){
        cout << "개수: ";
        cin >> count;
        if (cin.fail() || cin.peek() != '\n'){
            cout << "정수를 입력하십시오" << endl;
            cin.clear();
            cin.ignore(numeric_limits<streamsize>::max(), '\n');
            continue;
        }
        break;
    }
```

```
    return count;
}
int cinDate(){
    int date;
    int current_date = getCurrentDate();

    while(1){
        cout << "유통기한: ";
        cin >> date;
        if (cin.fail() || cin.peek() != '\n'){
            cout << "날짜 포맷에 맞춰 입력하십시오(YYYYMMDD)" << endl;
            cin.clear();
            cin.ignore(numeric_limits<streamsize>::max(), '\n');
            continue;
        }
        if (countDigits(date) != 8){
            cout << "날짜 포맷에 맞춰 입력하십시오(YYYYMMDD)" << endl;
            continue;
        }
        if (date < current_date){
            cout << "유통기한이 이미 지났습니다" << endl;
            continue;
        }
        break;
    }

    return date;
}
```

I. 입력

II. 결과

- 사용자 입력 반환

III. 설명

- 정수 입력의 경우 정수가 아닌 값이 입력되었는지 확인이 필요하다. 정수가 아닌 값이 입력되면 다음 입력에서 오류가 발생하므로, 이를 방지하기 위해 입력이 정수가 아니면 입력 버퍼를 완전히 비우고 다시 입력 받는다.
- 날짜 입력의 경우 날짜 포맷을 맞춰서 입력 받아야 하며 유통기한이 이미 지났는지 확인해야 한다.
- 사용자의 입력을 받을 때 확인해야 할 부분이 많으므로 이를 함수로 구현하여 입력이 필요한 부분에서 코드를 더 간결하게 작성할 수 있게 하였다.
- 이름 입력의 경우 확인해야 할 부분은 없지만 다른 두 원소의 입력이 함수로 구현되어 있으므로 이름 입력도 함수로 구현하였다.

- 자릿수 반환 함수

```
int countDigits(int number){
    int count = 0;
    while (number != 0) {
        number /= 10;
        count++;
    }
    return count;
}
```

I. 입력
   - number: 정수

II. 결과
   - number의 자릿수 반환

III. 설명
   - 정수의 자릿수가 필요한 경우가 있어서 구현하였다.

- 현재 날짜 반환 함수

```
int getCurrentDate(){
    int current_date;

    time_t raw_time;
    struct tm* time_info;
    raw_time = time(NULL);
    time_info = localtime(&raw_time);

    int current_y = time_info->tm_year + 1900;
    int current_m = time_info->tm_mon + 1;
    int current_d = time_info->tm_mday;

    current_date = current_y*10000 + current_m*100 + current_d;
    return current_date;
}
```

I. 입력

II. 결과
   - 현재 날짜 반환

III. 설명
   - 유통기한 임박 음식을 확인할 때 현재 날짜가 필요해서 구현하였다.

- openai api 연결 함수

```
bool ConnectApi(string key){
    // api 연결
    // 인터넷 연결이 없으면 예외를 발생시킨다
    // 잘못된 key를 입력하면 예외를 발생시킨다.
    openai::start(key);
    try{
        auto chat = openai::chat().create(R"(
        {
```

```
                "model": "gpt-4o-mini",
                "messages":[{"role":"user", "content":"test"}],
                "max_tokens": 10,
                "temperature": 0
            }
            )"_json);
        }
        catch(runtime_error const& e){
            cerr << e.what() << endl;
            return false;
        }
        return true;
    }
```

I. 입력
   - key: openai api key

II. 결과
   - OpenAI 인스턴스 생성
   - key의 유효성과 인터넷 연결에 대한 true/false 반환

III. 설명
   - 입력 받은 key로 openai::start() 함수를 호출하여 OpenAI 클래스의 static 인스턴스를 생성하고 반환한다.
   - static으로 선언되었기 때문에 인스턴스는 프로그램이 실행하는 동안 단 한 번만 생성되며 그 이후의 호출에서는 최초 생성된 객체를 반환한다. 이는 OpenAI 클래스가 싱글턴 패턴임을 알 수 있다.
   - 또한 static으로 선언되었기 때문에 ConnectApi() 함수가 종료된 후에도 OpenAI 인스턴스는 메모리에서 유지되고 있으며 프로그램 종료 시까지 사라지지 않는다.
   - openai::chat().create() 함수를 호출하여 api 요청을 보내고 응답을 받는다. 이때 입력 받은 key가 유효하지 않거나 인터넷 연결이 없을 경우 runtime error를 발생시킨다. 이를 예외처리 하여 runtime error가 발생한 경우 오류 코드를 출력하고 false를 반환한다.
   - api 요청을 보내고 성공적으로 응답을 받은 경우 true를 반환한다.
   - main() 함수에서 ConnectApi() 함수를 호출하여 api에 연결(유효한 OpenAI 인스턴스를 생성)하고 오류가 발생한 경우 프로그램을 종료시킨다.

- gpt-4o-mini 대답 생성 함수

```
string GetResponse(string system_prompt, string query){
    json payload = {
        {"model", "gpt-4o-mini"},
        {"messages", {
            {{"role", "system"}, {"content", system_prompt}},
            {{"role", "user"}, {"content", query}}
        }},
        {"max_tokens", 1000},
        {"temperature", 0}
    };
    json chat = openai::chat().create(payload);
    string response = chat["choices"][0]["message"]["content"];
    return response;
}
```

I. 입력
   - system_prompt: 시스템 프롬프트
   - query: 질문

II. 결과
   - response: gpt의 대답

III. 설명
   - 시스템 프롬프트와 질문을 입력 받고 gpt에게 전달하여 대답을 받는다.
   - api 응답은 json으로 오기 때문에 json에서 우리가 필요한 대답 부분만 추출하여 반환한다.

3) 기능 구현

① 파일 존재 확인

```
bool isFile(){
    fstream file(file_dir);
    if (!file.is_open()){
        cout << "음식 목록 파일을 찾을 수 없습니다" << endl;
        file.close();
        return false;
    }
    else{
        file.close();
        return true;
    }
}
```

I. 입력

II. 결과
   - 파일 유무에 대한 true/false 출력

III. 설명

- 전역 상수로 선언한 경로에 파일이 존재하지 않을 경우 오류 메시지를 출력하고 false를 반환한다.
- 파일이 존재할 경우 true를 반환한다.

② 기능 출력

```
void printFunction(){
    cout << "1. 현재 음식 보기" << endl;
    cout << "2. 음식 추가" << endl;
    cout << "3. 음식 제거" << endl;
    cout << "4. 유통기한 임박 음식 보기" << endl;
    cout << "5. 추천 메뉴 보기" << endl;
    cout << "6. 프로그램 종료" << endl;
}
```

I. 입력

II. 결과

- 사용할 수 있는 기능 출력

III. 설명

- 기능이 너무 많아서 사용자가 원할 때만 기능을 볼 수 있게 하기 위해 함수로 구현하였다.

③ 기능 선택

```
    printFunction();
    while (1){
        // 입력
        int choice; // 사용자 입력을 저장하는 변수
        cout << "사용할 기능을 선택하십시오(기능 보기: 0): ";

        cin >> choice;
        if (cin.fail() || cin.peek() != '\n'){
            cout << "정수를 입력하십시오" << endl;
            cin.clear();
            cin.ignore(numeric_limits<streamsize>::max(), '\n');
            continue;
        }
        if (choice == 0){
            printFunction();
        }
        else if (choice == 1){
            printFood(food_list);
        }
        else if (choice == 2){
            addFood(food_list);
        }
        else if (choice == 3){
            deleteFood(food_list);
        }
```

```
        else if (choice == 4){
            printExpiringFood(food_list);
        }
        else if (choice == 5){

        }
        else if (choice == 6){
            break;
        }
        else{
            cout << "잘못된 입력입니다" << endl;
        }
    }

    cout << "프로그램을 종료합니다" << endl;
    return 0;
}
```

I. 입력
   - 사용자 입력

II. 결과
   - 사용자 입력에 따른 함수 호출

III. 설명
   - 맨 처음에는 기능을 출력하고 그 다음 입력부터는 사용자가 요구하지 않는 한 기능을 출력하지 않는다.
   - 사용자 입력이 정수이므로 정수가 아닌 입력에 대한 검사와 처리를 한다.
   - 사용자의 입력이 0인 경우 기능을 출력한다.
   - 사용자의 입력이 1인 경우 printFood() 함수를 호출하여 음식 목록을 출력한다.
   - 사용자의 입력이 2인경우 addFood() 함수를 호출하여 음식을 추가한다.
   - 사용자의 입력이 3인경우 deleteFood() 함수를 호출하여 음식을 제거한다.
   - 사용자의 입력이 6인경우 프로그램을 종료한다.
   - 그 외의 입력의 경우 오류 메시지를 출력하고 다시 입력 받는다.

④ 음식 목록 출력

```
void printFood(vector<FoodInfo> list){
    int max_len = 0; // 이름 최대 길이
    int max_digits = 0; // 개수 최대 자릿수
    int date_size = 8; // 날짜 길이 (8로 고정)

    for (auto food : list){
        // 이름 길이
        int len = food.name.length();
        if (len > max_len)
            max_len = len;
```

```
        // 개수 자릿수
        int digits = countDigits(food.count);
        if (digits > max_digits)
            max_digits = digits;
    }

    cout << "|-";
    for (int i = 0; i < max_len; i++) cout << '-';
    cout << "-|-";
    for (int i = 0; i < max_digits; i++) cout << '-';
    cout << "-|-";
    for (int i = 0; i < date_size; i++) cout << '-';
    cout << "-|" << endl;

    for (auto food : list){
        cout << "| " << food.name;
        for (int i = 0; i < max_len - food.name.length(); i++) cout << ' ';
        cout << " | " << food.count;
        for (int i = 0; i < max_digits - countDigits(food.count); i++) cout << ' ';
        cout << " | " << food.date << " |" << endl;

        cout << "|-";
        for (int i = 0; i < max_len; i++) cout << '-';
        cout << "-|-";
        for (int i = 0; i < max_digits; i++) cout << '-';
        cout << "-|-";
        for (int i = 0; i < date_size; i++) cout << '-';
        cout << "-|" << endl;
    }
}
```

I. 입력
   - list: 음식 목록 벡터

II. 결과
   - 음식 목록 출력

III. 설명
   - 이름의 길이와 개수의 자릿수가 음식마다 모두 다르기 때문에 길이와 자릿수에 대한 최댓값을 구한다.
   - 최댓값을 기준으로 구분선을 출력하고 음식 정보도 같이 출력해서 전체 표를 출력한다.

⑤ 음식 추가

```
void addFood(vector<FoodInfo>& list){
    string name;
    int count;
    int date;

    // 사용자 입력 받기
    name = cinName();
    count = cinCount();
    date = cinDate();

    // 파일에 입력
    ofstream file(file_dir, ios::app);
```

```
    file << '\n' << name << ", " << count << ", " << date;
    file.close();

    // 벡터에 추가
    FoodInfo line;
    line.name = name;
    line.count = count;
    line.date = date;
    list.push_back(line);

    cout << name << " " << count << "개가 추가되었습니다." << endl;
}
```

I. 입력
   - list: 음식 목록 벡터
   - 사용자 입력

II. 결과
   - 벡터와 파일에 음식 추가

III. 설명
   - 사용자의 입력을 받아서 벡터와 파일에 새로운 음식을 추가하고 메시지를 출력한다.

⑥ 음식 제거

```
void deleteFood(vector<FoodInfo>& list){
    string name;
    int count;

    // 사용자 입력 받기
    name = cinName();
    count = cinCount();

    // 벡터에서 제거
    vector<FoodInfo>::iterator iter = list.begin();
    vector<FoodInfo>::iterator list_size = list.end();
    for (auto& food : list){
        if (name.compare(food.name) == 0){
            // 입력한 수량이 현재 수량보다 많으면 오류 출력
            if (count > food.count){
                cout << "입력한 수량이 현재 수량을 초과합니다" << endl;
                return;
            }
            else if (count < food.count){
                food.count -= count;
                break;
            }
            else{ // count == food.count
                list.erase(iter);
                break;
            }
        }
        iter += 1;
    }
```

```
    // 입력한 음식이 벡터 내에 없으면 오류 출력
    if (iter == list_size){
        cout << "해당 음식이 없습니다" << endl;
        return;
    }

    // 파일에서 제거
    // 제거한 벡터를 파일에 덮어쓰기
    ofstream file(file_dir);
    for (auto food : list){
        file << food.name << ", " << food.count << ", " << food.date << '\n';
    }
    file.close();

    cout << name << " " << count << "개가 제거되었습니다." << endl;
}
```

I. 입력
   - list: 음식 목록 벡터
   - 사용자 입력

II. 결과
   - 벡터와 파일에서 음식 제거

III. 설명
   - 사용자로부터 이름과 개수를 입력 받는다.
   - 입력한 개수가 현재 수량보다 많으면 오류를 출력한다.
   - 입력한 개수가 현재 수량과 같으면 입력한 음식을 벡터에서 제거한다.
   - 입력한 개수가 현재 수량보다 적으면 입력한 음식의 수량을 입력한 개수만큼 감소시킨다.
   - 만약 입력한 이름과 같은 이름의 음식이 벡터에 없으면 오류를 출력한다.
   - 변경된 벡터를 파일에 덮어써서 변경 사항을 저장한다.

⑦ 유통기한 임박 음식 출력

```
void printExpiringFood(vector<FoodInfo>& list){
    int current_date = getCurrentDate();
    int threshold = 3; // 유통기한 임박 기준

    // 유통기한 임박 음식 벡터를 생성하여 printFood() 함수로 보내준다
    vector<FoodInfo> expiring_food_list;
    for (auto food : list){
        if (food.date - threshold <= current_date)
            expiring_food_list.push_back(food);
    }
    printFood(expiring_food_list);
}
```

I. 입력
   - list: 음식 목록 벡터

II. 결과

- 유통기한 임박 음식 출력

III. 설명

- 음식 목록에서 유통기한이 얼마 남지 않은 음식들을 출력한다.
- getCurrentDate() 함수로 얻은 현재 날짜를 기준으로 유통기한이 3일 남은 음식들을 새로운 벡터에 저장하고 그 벡터를 printFood() 함수로 보내서 유통기한 임박 음식 목록을 출력한다.

⑧ 추천 메뉴 출력

```
void RecommendMenu(vector<FoodInfo>& list){
    string system_prompt = "영어로 음식 목록을 입력할테니까
                            터미널 환경에서 출력될거야. 적
                            밥과 면은 있다고 가정해.";

    // 질문 작성
    // 시스템 프롬프트가 적용되어 있으므로 음식 목록만 전달한
    string query;
    for (auto food : list){
        query += food.name;
        query += " ";
    }

    // 질문 보내고 대답 받기
    string response = GetResponse(system_prompt, query);

    // 대답 출력
    cout << response << endl;
}
```

```
string system_prompt = "영어로 음식 목록을 입력할테니까
이 음식들로 할 수 있는 요리와 간단한 설명만 한글로
출력해. 터미널 환경에서 출력될거야. 적절한 구분선을 같이
출력하고 마크다운 문법은 쓰지마. 밥과 면은 있다고
가정해.";
```

I. 입력

- list: 음식 목록 벡터

II. 결과

- 추천 메뉴 출력

III. 설명

- 시스템 프롬프트를 작성해서 음식 이름들 만으로 추천 메뉴를 받을 수 있게 한다.

- 문자열 query에 음식 이름들을 저장하고 GetResponse() 함수에 system_prompt와 query를 전달하여 추천 메뉴를 받고 이를 출력한다.

4) 테스트 결과

① 파일 존재 확인

```
PS C:\code\CPP2409-P>  & 'c:\Users\u
icrosoft-MIEngine-In-ux5zcmjc.y2f' '
IEngine-Pid-u1oo5jh3.xmd' '--dbgExe=
음식 목록 파일을 찾을 수 없습니다
프로그램을 종료합니다
PS C:\code\CPP2409-P>
```

② 기능 출력

```
PS C:\code\CPP2409-P>  & 'c:\Users\user\.vscod
icrosoft-MIEngine-In-tl1f2eeh.5sq' '--stdout=M
IEngine-Pid-pbvkrcsh.mdd' '--dbgExe=C:\msys64\
1. 현재 음식 보기
2. 음식 추가
3. 음식 제거
4. 유통기한 임박 음식 보기
5. 추천 메뉴 보기
6. 프로그램 종료
```

③ 기능 선택

```
사용할 기능을 선택하십시오(기능 보기: 0):
```

④ 음식 목록 출력

```
사용할 기능을 선택하십시오(기능 보기: 0): 1
|---------|----|----------|
| egg     | 10 | 20241111 |
|---------|----|----------|
| lettuce | 10 | 20241105 |
```

```
|---------|----|----------|
| sausage | 6  | 20241111 |
|---------|----|----------|
| meat    | 2  | 20241113 |
|---------|----|----------|
사용할 기능을 선택하십시오(기능 보기: 0):
```

⑤ 음식 추가

```
사용할 기능을 선택하십시오(기능 보기: 0): 2
음식 이름: fish
개수: 4
유통기한: 20241115
fish 4개가 추가되었습니다.
사용할 기능을 선택하십시오(기능 보기: 0):
```

```
사용할 기능을 선택하십시오(기능 보기: 0): 1
|---------|----|----------|
| egg     | 10 | 20241111 |
|---------|----|----------|
| lettuce | 10 | 20241105 |
|---------|----|----------|
| sausage | 6  | 20241111 |
|---------|----|----------|
| meat    | 2  | 20241113 |
|---------|----|----------|
| fish    | 4  | 20241115 |
|---------|----|----------|
사용할 기능을 선택하십시오(기능 보기: 0):
```

food_list.csv

```
egg, 10, 20241111
lettuce, 10, 20241105
sausage, 6, 20241111
```

```
4   meat, 2, 20241113
5
6   fish, 4, 20241115
```

- 음식 제거 기능에서 벡터를 파일에 덮어쓸 때 반복문으로 "{이름}, {개수}, {날짜}₩n"을 파일에 쓰기 때문에 마지막에 빈 줄이 하나 나온다. 이 상태에서 음식 추가 기능으로 새로운 음식을 추가하면 위의 이미지와 같이 한 줄이 비어진 채로 음식이 추가되지만 이는 이후에 파일을 읽거나 쓸 때 아무 문제도 일으키지 않는다.

⑥ 음식 제거

```
사용할 기능을 선택하십시오(기능 보기: 0): 3
음식 이름: fish
개수: 1
fish 1개가 제거되었습니다.
사용할 기능을 선택하십시오(기능 보기: 0):
```

```
사용할 기능을 선택하십시오(기능 보기: 0): 1
|---------|----|----------|
| egg     | 10 | 20241111 |
|---------|----|----------|
| lettuce | 10 | 20241105 |
|---------|----|----------|
| sausage | 6  | 20241111 |
|---------|----|----------|
| meat    | 2  | 20241113 |
|---------|----|----------|
| fish    | 3  | 20241115 |
|---------|----|----------|
사용할 기능을 선택하십시오(기능 보기: 0):
```

```
food_list.csv
1   egg, 10, 20241111
```

```
lettuce, 10, 20241105
sausage, 6, 20241111
meat, 2, 20241113
fish, 3, 20241115
```

```
사용할 기능을 선택하십시오(기능 보기: 0): 3
음식 이름: fish
개수: 3
fish 3개가 제거되었습니다.
사용할 기능을 선택하십시오(기능 보기: 0):
```

```
사용할 기능을 선택하십시오(기능 보기: 0): 1
|---------|----|----------|
| egg     | 10 | 20241111 |
|---------|----|----------|
| lettuce | 10 | 20241105 |
|---------|----|----------|
| sausage | 6  | 20241111 |
|---------|----|----------|
| meat    | 2  | 20241113 |
|---------|----|----------|
사용할 기능을 선택하십시오(기능 보기: 0):
```

food_list.csv

```
egg, 10, 20241111
lettuce, 10, 20241105
sausage, 6, 20241111
meat, 2, 20241113
```

⑦ 유통기한 임박 음식 출력

```
사용할 기능을 선택하십시오(기능 보기: 0): 1
|---------|----|----------|
| egg     | 10 | 20241124 |
|---------|----|----------|
| lettuce | 10 | 20241116 |
|---------|----|----------|
| sausage | 6  | 20241121 |
|---------|----|----------|
| meat    | 2  | 20241117 |
|---------|----|----------|
| potato  | 10 | 20241120 |
|---------|----|----------|
| fish    | 4  | 20241118 |
|---------|----|----------|
사용할 기능을 선택하십시오(기능 보기: 0): 4
|---------|----|----------|
| lettuce | 10 | 20241116 |
|---------|----|----------|
| meat    | 2  | 20241117 |
|---------|----|----------|
| fish    | 4  | 20241118 |
|---------|----|----------|
사용할 기능을 선택하십시오(기능 보기: 0): 6
프로그램을 종료합니다
PS Z:\CPP2409-P> date

2024년 11월 15일 금요일 오전 1:03:18


PS Z:\CPP2409-P>
```

⑧ 추천 메뉴 출력

```
사용할 기능을 선택하십시오(기능 보기: 0): 1
|---------|----|----------|
| egg     | 10 | 20241124 |
|---------|----|----------|
| fish    | 4  | 20241126 |
|---------|----|----------|
| onion   | 4  | 20241124 |
|---------|----|----------|
| cheeze  | 10 | 20241212 |
|---------|----|----------|
| sausage | 6  | 20241130 |
|---------|----|----------|
```

```
사용할 기능을 선택하십시오(기능 보기: 0): 5
---------------------------------------
1. 계란 볶음밥
   - 밥에 계란과 양파를 넣고 볶아 간단하게 만든 볶음밥입니다.

2. 치즈 오믈렛
   - 계란과 치즈, 양파를 섞어 팬에 부쳐 만든 부드러운 오믈렛입니다.

3. 소시지 파스타
   - 면에 소시지와 양파를 볶아 만든 간단한 파스타 요리입니다.

4. 생선 구이
   - 생선을 양념하여 구운 후, 계란과 함께 곁들여 먹는 요리입니다.

5. 치즈 생선 스테이크
   - 생선에 치즈를 얹어 오븐에 구워 만든 고소한 요리입니다.

6. 소시지와 양파 스튜
   - 소시지와 양파를 함께 끓여 만든 따뜻한 스튜입니다.
---------------------------------------
사용할 기능을 선택하십시오(기능 보기: 0):
```

5) 개선 사항

① 추천 메뉴 퀄리티 향상

I. 현황 및 문제점

- gpt-4o-mini가 낮은 퀄리티의 추천 메뉴를 제공하는 문제가 있었다.

II. 개선 방안

- 시스템 프롬프트를 더 자세하게 작성하여 추천 메뉴의 퀄리티를 높이고 출력 포맷을 일정하게 유지할 수 있도록 하였다.

```
"당신은 고급 요리 전문가입니다. 주어진 재료를 활용해 대중적이고 복잡한 요리를 제안하십시오.  \
요리는 다음 조건을 충족해야 합니다:  \
1. 대중적이고 유명한 요리를 제안하십시오.  \
2. 추가로 필요한 재료를 제안할 수 있습니다.  \
3. 모든 재료를 활용할 필요는 없습니다.  \
4. 다양한 메인 요리를 제안하십시오.  \
제안된 레시피는 고급 레스토랑이나 요리 경연 대회에서 사용할 수 있는 수준이어야 합니다.  \
요리의 이름, 재료, 간단한 설명만 출력하십시오.  \
터미널 환경에서 출력될 것을 생각해서 마크다운 문법은 사용하지 마시오.  \
출력 포멧:\n\
----------------------------------------\n\
요리 이름:...\n\
재료:...\n\
설명:...\n\
마지막에 출력 포멧에서 쓰인 구분선을 출력하십시오.";
```

III. 결과

```
사용할 기능을 선택하십시오(기능 보기: 0): 5
----------------------------------------
요리 이름: 퓨전 치즈 오믈렛과 소시지 스시 롤
재료: 계란, 생선, 양파, 치즈, 소시지, 김(해조류), 아보카도, 참기름, 소금, 후추
설명: 부드러운 계란과 치즈가 어울린 오믈렛을 구워 특제 소시지를 곁들여 풍미를 극대화합니다. 이와 함께
신선한 생선과 아보카도, 양파를 넣어 만든 스시 롤로 식사를 완성합니다. 참기름으로 고소한 맛을 더해 특별
한 퓨 전 요리를 즐길 수 있습니다.
----------------------------------------
요리 이름: 생선 소시지 스튜
재료: 생선, 소시지, 양파, 계란, 화이트 와인, 크림, 파슬리, 소금, 후추
설명: 신선한 생선과 소시지를 주재료로 한 고소한 스튜로, 양파와 화이트 와인으로 향미를 더하고 크림으로
부드러운 식감을 부여합니다. 마지막에 삶은 계란과 파슬리를 얹어 식감을 살리는 풍부한 맛의 요리입니다.

----------------------------------------
요리 이름: 양파 치즈 소시지 파이
재료: 소시지, 양파, 치즈, 계란, 밀가루(파이 크러스트를 위한), 우유, 소금, 후추
설명: 얇은 파이 크러스트에 풍성한 소시지와 양파를 볶아 치즈와 함께 채워 구워낸 파이로, 겉은 바삭하고
속은 부드러운 맛이 어우러지는 매력적인 요리입니다. 계란과 우유로 부드러움을 더하여 고급스러운 디너에
안성맞춤입니다.
----------------------------------------
요리 이름: 생선 소시지와 치즈를 곁들인 비스켓
재료: 생선, 소시지, 치즈, 양파, 계란, 우유, 밀가루
설명: 부드러운 비스켓 위에 생선과 소시지, 캐러멜라이즈한 양파를 얹고, 치즈를 녹여서 만든 요리입니다.
각 재료의 조화가 이루어져 독특하고 풍미 가득한 좌식을 제공하는 고급스러운 음식입니다.
----------------------------------------
사용할 기능을 선택하십시오(기능 보기: 0):
```

② 함수 헤더파일로 분리

I. 현황 및 문제점

- 프로그램 내 함수가 너무 많아서 유지 보수를 위해 여러 헤더파일에 나누어서 저장하였다.

II. 개선 방안

- 선언부는 include 폴더 안에, 구현부는 food_manager.cpp와 같은 위치에 저장한다. 헤더파일을 정상적으로 불러오기 위해 tasks.json 파일의 "args" 항목에 "-I", "${fileDirname}₩₩include"를 추가하고 구현 파일을 포함하여 빌드하기 위해 "${file}"대신 "${fileDirname}₩₩**.cpp"를 사용한다.

```
    "-g",
    // "${file}",
    "${fileDirname}\\**.cpp",
    "-o",
    "${fileDirname}\\${fileBasenameNoExtension}.exe",
    "-I",
    "${fileDirname}\\include",
```

- 함수들을 여러 곳에 분리해서 저장하면 구조체를 정의한 FoodInfo.hpp 파일이 여러 곳에서 include 되는데 이때 구조체가 중복 정의되어 오류를 발생시킨다. 헤더파일에 include guard를 추가하여 중복 정의를 방지하고 다른 헤더파일에도 모두 include guard를 추가하여 예기치 않은 오류가 발생하는 것을 방지한다.

```
#ifndef FOODINFO_HPP
#define FOODINFO_HPP

#include <string>
using namespace std;

// 음식 정보 구조체
struct FoodInfo{
    string name;
    int count;
    int date; // YYYYMMDD
};

#endif
```

III. 결과

```
MINGW64:/z/cpp2409-P
ku927@DESKTOP-F7K3Q07 MINGW64 /z/cpp2409-P (main)
$ ls -l | grep .cpp
-rw-r--r-- 1 ku927 197609    1218 Nov 26 10:05 fileUtils.cpp
-rw-r--r-- 1 ku927 197609    2910 Nov 26 11:07 food_manager.cpp
-rw-r--r-- 1 ku927 197609    6150 Nov 26 10:05 functions.cpp
-rw-r--r-- 1 ku927 197609    1393 Nov 26 10:49 inputUtils.cpp
-rw-r--r-- 1 ku927 197609    1240 Nov 26 10:06 openaiUtils.cpp
-rw-r--r-- 1 ku927 197609     588 Nov 26 10:05 utils.cpp

ku927@DESKTOP-F7K3Q07 MINGW64 /z/cpp2409-P (main)
$ |
```

```
MINGW64:/z/cpp2409-P/include
ku927@DESKTOP-F7K3Q07 MINGW64 /z/cpp2409-P/include (main)
$ ls -l | grep .hpp
-rw-r--r-- 1 ku927 197609 208 Nov 26 10:36 FoodInfo.hpp
-rw-r--r-- 1 ku927 197609 141 Nov 26 10:38 constants.hpp
-rw-r--r-- 1 ku927 197609 177 Nov 26 10:37 fileUtils.hpp
-rw-r--r-- 1 ku927 197609 386 Nov 26 10:39 functions.hpp
-rw-r--r-- 1 ku927 197609 243 Nov 26 10:40 inputUtils.hpp
-rw-r--r-- 1 ku927 197609 190 Nov 26 10:41 openaiUtils.hpp
-rw-r--r-- 1 ku927 197609 165 Nov 26 10:41 utils.hpp

ku927@DESKTOP-F7K3Q07 MINGW64 /z/cpp2409-P/include (main)
$
```

③ 한글 입출력

I. 현황 및 문제점

- 음식 목록에 음식을 추가할 때 음식 이름을 한글로 입력하면 한글이 깨지는 문제가 있었다.
- 한글을 포함하여 음식 목록을 출력할 때 한글의 바이트 수가 영어와 다르고 출력 칸 수도 영어와 달라서 구분선이 흐트러지는 문제가 있었다.

II. 개선 방안

- wstring을 이용하여 wcin을 해도 제대로 한글을 입력받지 못해서 utf16으로 입력을 받아 wstring에 저장하는 함수부터 구현하였다.

```
wstring InputUtf16(size_t bufferSize) {
    wstring buffer(bufferSize, L'\0'); // 버퍼
    DWORD char_num = 0; // 읽은 문자 수

    HANDLE stdinHandle = GetStdHandle(STD_INPUT_HANDLE); // 콘솔 입력 핸들
    if (stdinHandle == INVALID_HANDLE_VALUE) {
        return L"";
    }

    ReadConsoleW(stdinHandle, buffer.data(), bufferSize - 1, &char_num, NULL); // 콘솔 입력 읽기
    buffer.resize(char_num); // 읽은 문자 수만큼 크기 조정

    // '\r' 또는 '\n' 제거
    if (char_num > 0 && buffer[char_num - 1] == L'\n') {
        buffer.resize(char_num - 1);
        char_num -= 1;
    }
    if (char_num > 0 && buffer[char_num - 1] == L'\r') {
        buffer.resize(char_num - 1);
        char_num -= 1;
    }

    return buffer;
}
```

- 받은 입력을 구조체에 저장하기 위해 wstring을 string으로 변환하는 함수를 구현하였다.

```
string WstringToString(const wstring& wstr){
    wstring_convert<codecvt_utf8_utf16<wchar_t>> converter;
    return converter.to_bytes(wstr);
}
```

- 이를 바탕으로 CinName() 함수를 재조정하였다.

```
string CinName(){
    wstring name_w;
    string name;
    cout << "음식 이름: ";
    name_w = InputUtf16(256);
    name = WstringToString(name_w);
    return name;
}
```

- 이렇게 하면 string에 한글이 정상적으로 저장되며 벡터와 파일에서 한글을 사용할 때 문제가 발생하지 않는다.

- 영어는 string에 저장되면 한 글자당 1바이트씩 차지하지만 한글은 한 글자당 3바이트씩 차지한다. 즉, string.length()를 사용하면 영어의 경우 글자 수가 반환되고, 한글의 경우 글자 수 * 3이 반환된다. 하지만 영어나 한글이 wstring에 저장되고 wstring.length()를 사용한

다면 모두 글자 수가 반환된다. 정리하면 문자열 내의 한글 문자의 개수를 a, 영어 문자의 개수를 b라 할 때, string.length()는 3a+b를 반환하고 wstring.length()는 a+b를 반환한다.

- 콘솔 출력의 경우 영어는 한 글자당 한 칸 차지하지만, 한글은 한 글자당 두 칸 차지한다. 즉, 한글과 영어로 이루어진 문자열이 콘솔에 출력될 때 2a+b칸을 차지한다.
- 콘솔 출력 칸인 2a+b를 구하기 위해 3a+b와 a+b의 평균을 이용했다. wstring.length()의 값을 구하기 위해 string을 wstring으로 변환하는 함수부터 구현하였다.

```
wstring StringToWstring(const string& str){
    wstring_convert<codecvt_utf8<wchar_t>> converter;
    return converter.from_bytes(str);
}
```

- string.length()와 wstring.length()의 평균을 반환하여 콘솔 출력 칸을 구하는 함수를 구현하였다.

```
int GetOutputLength(const string& str){
    wstring wstr = StringToWstring(str);
    return (str.length() + wstr.length()) / 2;
}
```

- 이를 바탕으로 PrintFoods() 함수를 재조정하였다.

```
        // 이름 길이
        int len = GetOutputLength(food.name);
        if (len > max_len)
            max_len = len;
```

```
        cout << "| " << food.name;
        for (int i = 0; i < max_len - GetOutputLength(food.name); i++) cout << ' ';
        cout << " | " << food.count;
        for (int i = 0; i < max_digits - CountDigits(food.count); i++) cout << ' ';
        cout << " | " << food.date << " |" << endl;
```

III. 결과

```
사용할 기능을 선택하십시오(기능 보기: 0): 1
|-----------|------|----------|
| 음식      | 개수 | 유통기한 |
|-----------|------|----------|
| egg       | 10   | 20241202 |
|-----------|------|----------|
| onion     | 4    | 20241127 |
|-----------|------|----------|
| chicken   | 1    | 20241128 |
|-----------|------|----------|
| pork      | 1    | 20241128 |
|-----------|------|----------|
| tofu      | 1    | 20241130 |
```

```
|-----------|------|----------|
| 크래미    | 2    | 20241212 |
|-----------|------|----------|
| egg샐러드 | 1    | 20241203 |
|-----------|------|----------|
사용할 기능을 선택하십시오(기능 보기: 0):
```

food_list.csv > data

```
egg, 10, 20241202
onion, 4, 20241127
chicken, 1, 20241128
pork, 1, 20241128
tofu, 1, 20241130
크래미, 2, 20241212
egg샐러드, 1, 20241203
```

4. 계획 대비 변경 사항

1) 음식 목록 파일 확장자 변경

- ReadFoodListCsv() 함수를 간결하게 작성하기 위해서 음식 목록 파일의 확장자를 csv가 아닌 bin으로 변경하였다. 데이터를 불러올 때 콤마를 제거하는 작업을 없애고 file >> name >> count >> date 처럼 한번에 불러오게 하였다. 또한 bin 확장자로 저장해서 사용자의 직접 접근을 어렵게 하였다. csv를 사용하지 않으니 함수의 이름도 ReadFoodList()로 변경하였다.

```
vector<FoodInfo> ReadFoodList(){
    vector<FoodInfo> food_list;
    ifstream file(food_list_dir, ios::binary);

    string name;
    int count;
    int date;
    while (file >> name >> count >> date){
```

```
        FoodInfo food = {name, count, date};
        food_list.push_back(food);
    }

    file.close();

    return food_list;
}
```

2) 언어 설정 추가

I. 설명

- 다양한 언어를 지원하기 위해 언어 설정을 추가하였다.
- 번역의 기준은 영어로 놓고 각 언어에 대해 영어 문장과 그에 대응하는 번역 문장을 json 형식으로 저장하였다.
- 프로그램에서 json 파일을 불러와서 STL의 unordered_map에 저장하고 문자열을 출력할 때마다 이것을 사용한다.
- 관련 객체나 함수들은 language.hpp와 language.cpp에서 관리한다.

II. 구현

① 언어 파일

```
locales
  {} en.json
  {} ko.json
```

```
locales > {} ko.json > Load the food list
{
    "locale": "ko_KR.UTF-8",
    "language": "한글",

    "Enter openai api key: ": "openai api key를 입력하시오: ",
    "Connect to openai api": "openai api에 연결합니다",
    "Load the food list": "음식 목록을 불러옵니다",

    "Select the desired feature (View features: 0): ": "사용할 기능을 선택하십시오 (기능 보기: 0): ",
    "Please enter an integer": "정수를 입력하십시오",
    "Invalid input": "잘못된 입력입니다",
    "Exit the program": "프로그램을 종료합니다",
```

② 언어 파일 저장 객체

```
extern unordered_map<string, string> language_pack;
```

③ 언어 파일 불러오기

```
json ReadJson(string dir){
    ifstream json_file(dir);
    json js = json::parse(json_file);
    return js;
}
```

- fileUtils.cpp에 json 파일을 읽는 ReadJson() 함수를 구현해 놓았다.

```
void LoadLanguagePack(string language){
    string dir = "locales/.json";
    dir.insert(8, language);
    IsFile(dir);
    json lang_json = ReadJson(dir);
    for (auto& [key, value] : lang_json.items()) {
        language_pack[key] = value.get<string>();
    }
}
```

- ReadJson() 함수로 언어 파일을 불러와서 json타입(nlohmann::json 타입) 객체에 저장한 후 이를 다시 unordered_map 타입의 객체인 language_pack에 저장한다.
- json 타입이 아닌 unordered_map 타입을 쓰는 이유는 언어 파일이 중첩된 json 형태가 아닌 단일 레벨의 key-value 구조이며 속도와 메모리 사용량 측면에서 unordered_map이 더 좋기 때문이다.

④ 설정 기능 추가

```
        else if (choice == 6){
            ChangeSettings();
        }
```

- main() 함수에서 6을 입력 받을 경우 ChangeSettings() 함수를 호출한다.

```
void ChangeSettings(){
    while (1){
        cout << language_pack["--Settings--"] << endl;
        cout << language_pack["0. Back"] << endl;
        cout << language_pack["1. Language"] << endl;

        int choice;
        choice = CinInteger(language_pack["Select: "]);
        if (choice == 0){
            return;
        }
        else if (choice == 1){
            SetLanguage();
        }
        else{
            cout << language_pack["Invalid input"] << endl;
        }
    }
}
```

- 0을 입력 받을 경우 함수를 종료함으로써 뒤로 가기 기능을 구현했다.
- 1을 입력 받을 경우 SetLanguage() 함수를 호출한다.

```
void SetLanguage(bool init){
    string language;
    if (init == true) LoadLanguagePack("en");

    while (1){
        if (init != true){
            cout << language_pack["--Language--"] << endl;
            cout << language_pack["0. Back"] << endl;
        }
        cout << "1. English" << endl;
        cout << "2. Korean" << endl;

```

```
        int choice;
        choice = CinInteger(language_pack["Select: "]);
        if (choice == 0){
            if (init == true){
                cout << language_pack["invalid input"] << endl;
                continue;
            }
            return;
        }
        else if (choice == 1){
            language = "en";
            break;
        }
        else if (choice == 2){
            language = "ko";
            break;
        }
        else{
            cout << language_pack["invalid input"] << endl;
            continue;
        }
    }

    // 설정 저장
    json settings = ReadJson("settings.json");
    settings["language"] = language;
    WriteJson("settings.json", settings);

    // 언어 팩 불러오기
    LoadLanguagePack(language);
    // locale 설정
    string locale = language_pack["locale"];
    setlocale(LC_ALL, locale.c_str());

    cout << language_pack["Language settings have been completed"] << endl;
}
```

- ChangeSettings() 함수와 마찬가지로 0을 입력 받았을 경우 함수를 종료한다.
- 입력에 따라 언어를 설정하고 이를 fileUtils.cpp에 구현한 WriteJson() 함수를 이용하여 settings.json에 저장한다.

==========

```
void WriteJson(string dir, json js){
    ofstream json_file(dir);
    json_file << js;
}
```

```
{} settings.json > ...
{"language":""}
```

==========

- 변경된 언어에 맞는 파일을 다시 불러와서 language_pack에 저장한다.

⑤ 초기 설정

- SetLanguage() 함수에서 bool 형 변수 init은 초기 설정 여부를 확인하기 위한 변수로써 language.hpp에 기본값을 false로 구현해 놓았다.

==========

```
void SetLanguage(bool init = false);
```

==========

- 만약 아직 설정 값이 없는 초기 설정이라면 임시로 영어 json 파일을 사용하고 뒤로 가기를 막아놓는다.

```
    // 설정 불러오기
    json settings = ReadJson(settings_dir);
    string language = settings["language"];
    if (language == ""){
        SetLanguage(true);
    }
    else{
        // 언어 팩 불러오기
        LoadLanguagePack(language);
        // locale 설정
        string locale = language_pack["locale"];
        setlocale(LC_ALL, locale.c_str());
    }
```

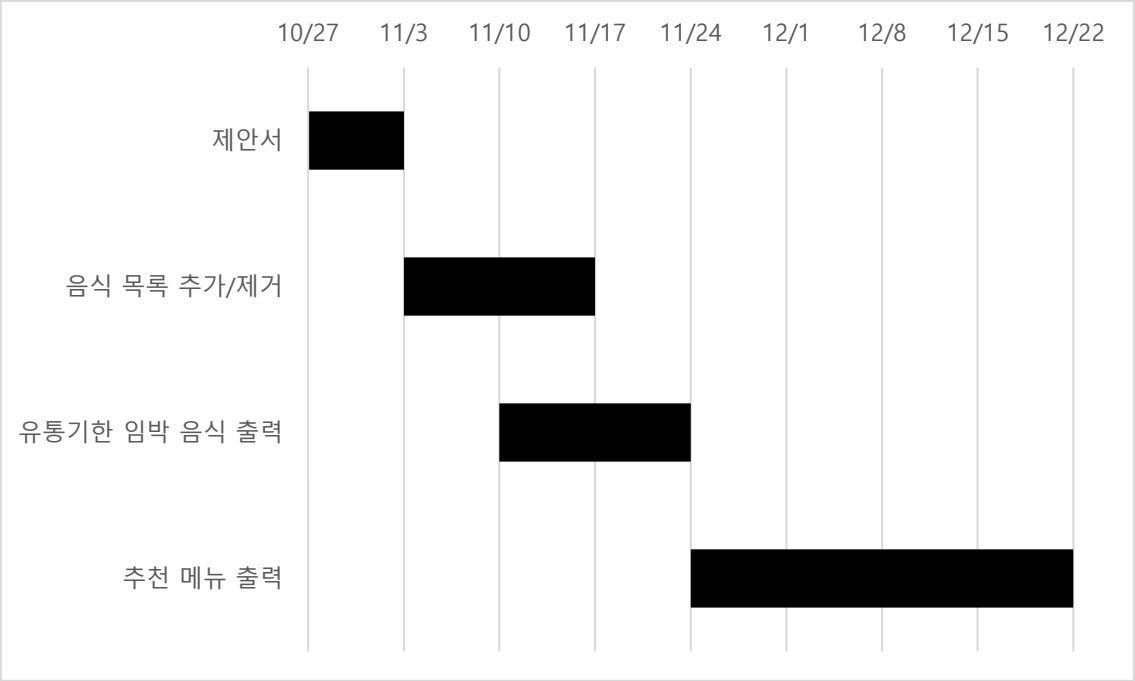
- main() 함수가 실행되면 openai api에 연결하기 전에 설정을 불러오며 언어 설정이 안되어 있을 경우 SetLanguage() 함수를 init = true로 해서 호출한다.
- 언어 설정이 되어 있다면 설정된 언어의 json 파일을 불러와서 language_pack에 저장한다.

⑥ language_pack 사용

```
void PrintFunctions(){
    cout << language_pack["1. View current foods"] << endl;
    cout << language_pack["2. Add a food"] << endl;
    cout << language_pack["3. Remove a food"] << endl;
    cout << language_pack["4. View foods nearing expiration"] << endl;
    cout << language_pack["5. View recommended menu"] << endl;
    cout << language_pack["6. Settings"] << endl;;
    cout << language_pack["7. Exit program"] << endl;
}
```

- 결과적으로 language_pack에는 key로 영어 문장, value로 번역된 문장이 저장된다. 따라서 위와 같이 영어 문장인 key값으로 language_pack에 접근하여 번역 문장을 얻을 수 있으며 cout으로 이를 출력한다.
- language_pack의 구현으로 각 언어에 대한 json 파일만 만들면 쉽게 언어를 추가할 수 있다.

5. 프로젝트 일정


10/27
11/3
11/10
11/17
11/24
12/1
12/8
12/15
12/22
제안서
음식 목록 추가/제거
유통기한 임박 음식 출력
추천 메뉴 출력