C++프로그래밍및실습

재료 관리 및 재료 기반 레시피 추천 프로그램 개발 보고서

최종 보고서

제출일자: 2023.12.24
제출자명: 김민호
제출자학번: 200244

1. 프로젝트 목표

1) 배경 및 필요성

현대 사회에서는 모두가 빠르게 움직이고, 시간이 제한적입니다. 이러한 환경에서 음식을 요리하고 다양한 레시피를 찾는 것은 쉬운 일이 아닙니다. 만들고 싶은 음식이 있어도 재료가 없다면 그 재료를 사기 위해 마트에 장을 보며 일일이 재료를 구매해야 하고, 음식을 다 먹은 후 남은 재료들을 처리하는 것도 여간 번거로운 일이 아닙니다. 이 재료들을 다 먹지 못해 버리는 일도 허다합니다. 그런데 만약 우리에게 주어진 재료만으로 요리를 할 수 있게 도와주는 앱이 있다면 이러한 불편한 점들이 해결될 것입니다. 또한 재료를 간편하게 관리할 수 있다면 더 좋을 것입니다. 이것은 0소비자에게는 시간과 에너지를 절약하고 재료 소비와 활용을 최적화하는데 도움을 줄 것입니다. 뿐만 아니라 판매자 입장에서도 소비자들의 구매를 더욱 유발하고 활성화할 것이기 때문에 유익할 것입니다. 또, 환경적으로도 재료 낭비로 인한 음식물 쓰레기 문제 해결에도 도움을 줄 것으로 예상합니다. 이 프로그램은 소비자, 판매자, 환경적인 측면에서 모두 유익한 프로그램이 될 것입니다.

2) 프로젝트 목표

재료 관리와 재료 기반 레시피 추천 프로그램을 개발하여 사용자들에게 시간과 에너지를 절약하고 재료 소비와 활용을 최적화하는데 도움을 주는 것을 목표로 합니다.

3) 차별점

기존 프로그램인 '만개의 레시피'에서는 '냉장고 파먹기' 라는 창에서 원하는 재료를 직접 입력해야 한다. 이 창에서 검색하는 것은 일반적인 창에서 검색하는 것과 차이가 거의 없다. 또한 재료 선호도를 입력 받는 기능이 없다. 레시피에 대한 평점 입력 시스템도 단순하다. 재료를 관리하는 기능 또한 없다. 반면 우리가 개발하려는 프로그램에서는 구매한 재료가 자동적으로 저장되고 사용자의 입력에

의해 간편하게 관리된다. 또한 냉장고 속 재료를 기반으로 추천하는 기능과, 선호 재료를 기반으로 추천하는 기능을 갖추고 있다. 평점을 세분화하여 (난이도, 조리 시간, 맛, 가성비) 평점을 기반으로 검색할 수 있다.

2. 기능 계획

1) 기능 1 재료 구매 내역 분석 및 재료 저장

(1) 세부 기능 1 : 제품명 분석 및 재료 분류

- 제품명을 분석하여 제품들을 각 재료별로 분류한다.

예상 알고리즘 : 제품명 마지막 단어를 추출 후 재료로 분류, 분류 빈도가 낮을시 단어 내에서 문자 분석 후 해당 문자를 포함하며 분류된 빈도가 높은 재료로 분류

더 섬세한 알고리즘은 인공지능으로 해결 가능

(2) 세부 기능 2 : 앱 내 구매 시 '재료'를 구매한 목록(별칭 : 냉장고)에 저장

- 앱 내 구매 시 '재료'를 구매한 목록(별칭 : 냉장고)에 저장한다.

(3) 세부 기능 3 : 외부에서 구매 시 구매 내역을 분석하여 '재료'를 냉장고에 저장

- 외부에서 구매 시 구매 내역을 분석하여 '재료'를 냉장고에 저장한다.

2) 기능 2 재료 기반 레시피 추천

- 레시피 추천을 선택하면 재료에 기반한 레시피를 추천하는 창을 출력한다.

(1) 세부 기능 1 : 냉장고 재료 기반 레시피 추천

- 냉장고 재료만으로 요리 가능(대체 식재료도 가능)한 레시피를 추천한다.

- 냉장고 재료가 다수 포함된 레시피를 추천한다.

(2) 세부 기능 2 : 선호 재료 기반 레시피 추천

- '선호 재료'가 포함된 레시피를 추천한다.

추천 순서 : 1. 냉장고 재료 2. 선호도

3) 기능 3 레시피 검색

- 레시피 검색을 선택하면 레시피를 검색하는 창을 시작한다.

- 재료 기반

- 요리명 기반

- 카테고리 기반(ex. 쉬운, 간단한, 빠른, 가성비, 주말, 자취...)

- 평점 : 정렬 기준으로 변경

4) 기능 4 : 항목별 평점 수집

-레시피를 읽으면

- 난이도, 조리 시간, (맛, 가성비) 의 항목별 별점을 입력받는다.

(1) 세부 기능 1 : 항목별 평점 수집

- 난이도, 조리 시간, (맛, 가성비) 의 항목별 별점을 입력받는다.

(2) 세부 기능 2 : 포인트 적립 기능

- 모든 별점을 입력받으면 사용자에게 5포인트를 적립한다.

항목, 별점, 포인트

레시피별로 항목별 별점 입력받기

저장하기

조건 : 모든 별점이 입력 완료되면 포인트 적립

5) 기능 5 : 재료 관리

- 레시피 조리 시 , 혹은 재료 화면에서 재료를 선택하면 기본적으로 '선호 재료' 출력, 냉장고 재료는 추가로 '재료 소진', '재료 소진 임박' 출력, 냉장고에 없는 재료는 '장바구니에 담기', '바로 구매' 출력. 위 선택지 중 선택한다.

(1) 세부 기능 1: 재료 선호도 관리 기능

- '선호 재료' 선택 시, 해당 재료를 '선호 재료'로 지정, 선호 재료 기반 레시피 추천 기능에서 사용된다.

(2) 세부 기능 2: 재료 소진 관리 기능

- '재료 소진 임박' 선택 시, 구매할 건지 문구 출력 & 재료를 '소진 임박 재료'로 지정', 구매 시 지정 해제한다.

- '재료 소진' 선택 시 구매할 건지 문구 출력 & 선호 재료가 아니라면냉장고 재료에서 제거한다.

%% 여기서 재고상태가 0인 재료는 선호재료밖에 안남게 된다.

(3) 세부 기능 3: 요리에 필요한 재료 구매 기능

- '바로 구매'를 선택 시 구매 확인 창을 출력 후 구매한다.

- '장바구니에 담기'를 선택 시 재료를 '장바구니'에 저장한다.

- 장바구니를 선택하면 장바구니 목록을 출력하고, 바로 구매를 선택 시 구매 확인 창을 출력 후 구매한다.

6) 기능 6 : 페이지 관리

- 페이지 클래스와 메서드를 정의하고,

기능 1 ~ 5를 메뉴로 입력받아 입력에 따른 페이지를 열고 관리한다.

세부기능 1: 메인 메뉴 페이지 열기

- 기능 1 ~ 5를 메뉴로 입력받아 입력에 따른 페이지를 연다.

세부기능 2: 레시피 추천 페이지 열기

- 메인메뉴에서 2를 입력하면 기능 2를 수행하는 페이지를 연다.

세부기능 3: 레시피 검색 페이지

- 메인메뉴에서 3을 입력하면 기능 3을 수행하는 페이지를 연다.

세부기능 4: 냉장고 페이지

- 메인메뉴에서 4를 입력하면 기능 4를 수행하는 페이지를 연다.

세부기능 5: 장바구니 페이지

- 메인메뉴에서 4를 입력하면 기능 4를 수행하는 페이지를 연다.

세부기능 6: 마이페이지

- 메인메뉴에서 5를 입력하면 기능 5를 수행하는 페이지를 연다.

희망 기능 : 소비 기한 관리 기능

3. 진척사항

1) 기능 구현

기능 4) 항목별 평점 수집

- 입력 : 평점을 저장할 레시피 인스턴스, 평가 항목별 평점

- 출력 : 레시피 인스턴스에 평점 저장, 유저의 포인트 5 적립

- 설명

InputRating() 메서드

1. 각 항목별로 평점을 입력받는다.

2. 입력받을 평점은 1~5사이이므로 입력받은 평점이 범위를 벗어나면 모든 항목의 평점을 0으로 저장한 후 함수를 종료한다.

3. 입력 평점이 범위 이내에 있다면 각 항목의 평점을 레시피 인스턴스에 저장한다.

4. 모든 입력을 정상적으로 마쳤다면 모든 항목의 평점이 0이 아닐 것이므로 첫 항목의 평점이 0이 아닐 때, 유저에게 포인트 5점을 적립한다.

- **적용된 배운 내용** : 반복문, 조건문, 제어구문, 벡터, 배열, 전역변수, 함수, 클래스, 멤버 변수, 멤버 함수, 접근 지시자, 헤더파일, 논리 연산, 입출력, 상수, 설정자

- 코드 스크린샷

```
48  void Recipe::InputRating() {
49      int rating;
50      string category[CATEGORY_SIZE] = { "난이도", "조리 시간", "맛", "가성비" };
51      cout << "--------------------------------" << endl;
52      cout << "평점을 입력해주세요(1 ~ 5)" << endl;
53      for (string ctg : category) {
54          cout << ctg << " : ";
55          cin >> rating;
56          if (rating < 1 || 5 < rating) {
57              cout << "입력 범위를 벗어났습니다." << endl;
58              for (string ctg : category) {
59                  SetRating(ctg, 0);
60              }
61              break;
62          }
63          SetRating(ctg, rating);
64      }
65      if (GetRating(category[0]) != 0) {
66          user.PlusPoint(5);
67      }
68  }
```

기능 5) 재료 관리

재료 페이지(세부 기능 1,2,3 통합)

- 입력 : 선택한 재료

- 출력 : 기본 메뉴, 제목, 선호 재료 상태, 재고 상태, 재료 구매 메뉴 출력, 선택한 재료의 멤버 변수 갱신, 입력받은 메뉴에 따른 페이지 생성, 페이지 삭제

- 설명

1. 선택한 재료가 '선호 재료'라면 첫번째 메뉴를 '선호 재료 헤제', 아니라면 '선호 재료로 지정'으로 한다.

2. 선택한 재료의 선호 재료 상태와 선호 재료 상태 갱신 메뉴를 출력한다.

3. 선택한 재료의 재고 상태와 재고 상태 갱신 메뉴를 출력한다.

4. 재료 구매 페이지들로 이동할 수 있는 메뉴를 출력한다.

5. 메뉴를 입력받는다.

6. 메뉴 1 ~ 4번은 선택한 재료의 멤버 변수를 갱신하는 메뉴로 해당 멤버 변수를 갱신한다.

7. 메뉴 5번을 입력받으면, 해당 재료의 in_basket을 true로 갱신한 후, 장바구니 페이지를 '열린 페이지 목록'에 추가한다.

8. 메뉴 6번을 입력받으면 해당 재료의 구매 페이지를 추가한다.

9. -1,과 0을 입력받으면 각각 현재 재료페이지를 삭제, 메인페이지를 추가한다.

10. 잘못 입력시 메시지를 출력한다.

- **적용된 배운 내용** :

1. 접근자, 설정자

2. 포인터 타입 벡터 정적 멤버 변수(vector <Ingredient*> Ingredient::fridge)

3. 범위 기반 for 문, for문

4. if, else 문, switch문

- 코드 스크린샷

```cpp
vector <Ingredient*> Ingredient::fridge;
//Ingredient Class Method
Ingredient::Ingredient(string name) {
    this->name = name;
    in_basket = false;
    purchased = false;
    preference = false;
    stock_state = 2;

}
string Ingredient::GetName() { return name; }
void Ingredient::SetName(string name) { this->name = name; }
bool Ingredient::GetPreference() { return preference; }
void Ingredient::SetPreference(bool preference) { this->preference = preference; }
int Ingredient::GetStockState() { return stock_state; }
void Ingredient::SetStockState(int stock_state) { this->stock_state = stock_state; }

bool Ingredient::GetInBasket() { return in_basket; }
void Ingredient::SetInBasket(bool in_basket) { this->in_basket = in_basket; }
bool Ingredient::GetPurchased() { return purchased; }
void Ingredient::SetPurchased(bool purchased) { this->purchased = purchased; }
```

```cpp
IngredientPage::IngredientPage(Ingredient* ingredient) : Page("재료 관리") {
    vector <string > menus { "선호 재료로 지정", "재료 추가", "재료 소진 임박", "재료 소진", "장바구니에 담기", "바로 구매" };
    for (string menu : menus) {
        menu_list.push_back(menu);
    }
    this->ingredient = ingredient;
    if (ingredient->GetPreference() == true) {
        cout << menu_list.size();
        menu_list[0] = "선호 재료 해제";
    }
}
void IngredientPage::UpdatePage() {
    if (ingredient->GetPreference() == true) {
        cout << menu_list.size();
        menu_list[0] = "선호 재료 해제";
    }
    else menu_list[0] = "선호 재료로 지정";
}
void IngredientPage::PrintMenu(const vector <string>&basic_menu_list) {
    int i = 0;
    cout << endl << endl;
    cout << "--------------------------------" << endl;
    for (string basic_menu : basic_menu_list) {
        cout << basic_menu << "\t";
    }
    cout << endl;
    cout << title << endl << endl;
    cout << ingredient->GetName() << endl;
    if (ingredient->GetPreference() == true) {
        cout << endl << "선호 재료" << endl;
    }
    else cout << "        "<< endl;
    cout << i + 1 << ", " << menu_list[i] << endl;
    i++;
    cout <<endl << "재고 상태 : ";
    switch (ingredient->GetStockState()) {
        case 0:
            cout << "소진 (없음)" << endl;
        case 1:
            cout << "소진 임박" << endl;
        case 2:
            cout << "충분함" << endl;
    }
    for (int j = 0; j < 3; j++) {
        cout << i + 1 << ", " << menu_list[i] << endl;
        i++;
    }
    cout << endl << "재료 구매" << endl;
    for (int j = 0; j < 2; j++) {
        cout << i + 1 << ", " << menu_list[i] << endl;
        i++;
    }
    cout << endl;
}
```

```cpp
int IngredientPage::OpenPage() {
    /*

    */
    PrintMenu();
    int command = InputMenu("메뉴 선택 : ");
    switch (command) {
    case 1:
        if (menu_list[0] == "선호 재료로 지정") {
            ingredient->SetPreference(true);
        }
        else {
            ingredient->SetPreference(false);
        }
        break;
    case 2:
        ingredient->SetStockState(2);
        break;
    case 3:
        ingredient->SetStockState(1);
        break;
    case 4:
        ingredient->SetStockState(0);
        break;
    case 5:
        // 장바구니에 담고 장바구니 페이지 열기
        ingredient->SetInBasket(true);
        opened_pages.push_back(new BasketPage());
        break;
    case 6:
        // 구매 페이지 열기
        opened_pages.push_back(new BuyPage(ingredient));
        break;
    case 0:
        opened_pages.push_back(new MainPage());
        break;
    case -1:
        DeletePage();
        break;
    default:
        cout << "잘못된 메뉴를 선택하였습니다." << endl;
    }
    return 0;
}
```

기능 6) 페이지 관리

기본 시스템(main 함수)

- 입력 : 입력받은 메뉴

- 출력 : 메인 페이지 열기, 입력 메뉴에 따른 페이지 열기, 프로그램 종료

- 설명

1. 프로그램 시작시 '열린 페이지 목록(Page::opened_pages)'에 메인 페이지를 추가한다.

2. 열린 페이지 목록은 유저가 열어놓은 페이지들을 저장하는 벡터이다. 이 목록의 마지막 페이지를 현재 데이터를 바탕으로 업데이트한다.

3. 마지막 페이지를 연다.

4. 페이지에서 프로그램 종료 명령(1)을 받으면, 데이터를 저장하고, 프로그램을 종료한다.

5. 종료 명령을 받지 않았다면, 프로그램이 종료될 때까지 2번, 3번 과정(마지막 페이지 업데이트, 열기)을 반복한다.

- 적용된 배운 내용 :

1. 함수(ReadIngredientData(), SaveIngredientData())

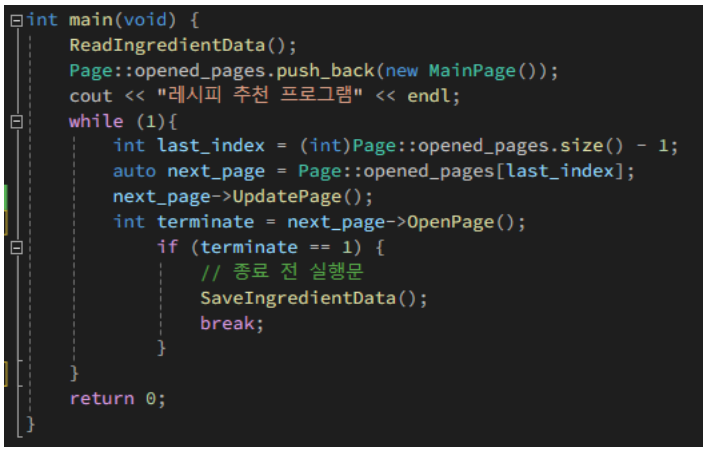
2. 정적 멤버 변수(opened_pages)

3. 벡터(push_back(), size())

4. 클래스 포인터의 멤버 함수 호출(->)

5. 무한 루프(while(1))

- 코드 스크린샷



```
void ReadIngredientData(){
    ifstream igd_data{ "ingredient_data.csv" };
    if (!igd_data.is_open()) {
        std::cerr << "Error" << std::endl;
        return;
    }

    string line;
    while (getline(igd_data, line)) {
        Ingredient* pigd = new Ingredient();
        istringstream iss{ line };
        string data;
        int i = 0;
        while (getline(iss, data, ',')) {
            switch (i % 5) {
            case 0:
                pigd->SetName(data);
                break;
            case 1:
                pigd->SetInBasket(stoi(data));
                break;
            case 2:
                pigd->SetPurchased(stoi(data));
                break;
            case 3:
                pigd->SetPreference(stoi(data));
                break;
            case 4:
                pigd->SetStockState(stoi(data));
                break;
            }
            i++;
        }
        Ingredient::fridge.push_back(pigd);
    }

    igd_data.close();
}
```

```
void SaveIngredientData(){
    ofstream igd_data{ "ingredient_data.csv" };
    if (!igd_data.is_open()) {
        std::cerr << "Error" << std::endl;
        return;
    }
    for (Ingredient* pigd : Ingredient::fridge) {
        string name;
        int in_basket, purchased, preference, stock_state;

        name = pigd->GetName();
        in_basket = pigd->GetInBasket();
        purchased = pigd->GetPurchased();
        preference = pigd->GetPreference();
        stock_state = pigd->GetStockState();
        igd_data << name << ",";
        igd_data << in_basket << "," << purchased << ",";
        igd_data << preference << "," << stock_state << "," << endl;
    }
    igd_data.close();
}
```

페이지 출력

- 입력 : 페이지 제목, 기본 메뉴 목록(벡터), 메뉴 목록(벡터), 메뉴가 없을 경우의 출력 메시지

- 출력 : 페이지 출력

- 설명

1. 기본 메뉴를 출력한다.

2. 페이지의 제목을 출력한다.

3. 메뉴가 존재하지 않는다면 없음 메시지를 출력한다.

4. 메뉴가 존재하면 메뉴 목록들을 출력한다.

- 적용된 배운 내용

1. 생성자

2. cout, cin, 이스케이프 시퀀스(escape sequence)(₩t)

3. 범위 기반 for문

4. % 연산자

메뉴 입력

- 입력 : 입력 메시지

- 출력 : 메뉴 입력 기능

- 설명 : 입력 메시지 출력 후, 메뉴 번호를 입력받아 입력 번호를 반환한다.

- 적용된 배운 내용 : cout, cin, 접근자와 const 키워드를 이용한 매개변수

- 코드 스크린샷

```cpp
//Page Class Method
Page::Page(string title, vector <string> menu_list) {
    this->title = title;
    // UpdatePage()
    // PrintMenu()
    // OpenPage()
    // InOpenedPages
    for (string menu : menu_list) {
        this->menu_list.push_back(menu);
    }
}
//vector <string> menu_list
void Page::UpdatePage() {}

void Page::PrintMenu(string none_message, const vector <string>& basic_menu_list) {
    cout << endl << endl;
    cout << "--------------------------------" << endl;
    int i = 1;
    for (string basic_menu : basic_menu_list) {
        cout << basic_menu << "\t";
        if (i % 2 == 0) {
            cout << endl;
        }
        i++;
    }
    cout << endl << endl;
    cout << title << endl << endl;

    if (menu_list.size() == 0) {
        cout  << none_message << endl << endl;
        return;
    }
    i = 1;
    for (string menu : menu_list) {
        cout << " " << i << ". " << menu << endl;
        i++;
    }
    cout << endl;
}


int Page::InputMenu(string message) {
    int command = 0;

    cout << message;
    cin >> command;
    return command;
}
```

세부 기능 1) 메인페이지 열기

- 입력 : 페이지 제목, 기본 메뉴 목록(벡터), 메뉴 목록(벡터), 입력받은 메뉴

- 출력 : 페이지 출력, 메뉴 입력, 입력에 따른 페이지 생성, 삭제, 프로그램 종료 메시지 전달

- 설명

1. 메인 페이지를 출력한다.

2. 메뉴를 입력 받는다.

3. 입력이 3이면 메뉴 번호 3번, 냉장고 페이지를 '열린 페이지 목록'에 추가한다.

4. 입력이 4이면 메뉴 번호 4번, 장바구니 페이지를 '열린 페이지 목록'에 추가한다.

5. 입력이 5이면 메뉴 번호 5번, 마이페이지를 '열린 페이지 목록'에 추가한다.

6. 입력이 -1이면 현재 페이지(메인 페이지)를 '열린 페이지 목록'에서 삭제한다.

7. 입력이 0이면 main함수에 프로그램 종료 메시지(1)을 반환한다.

8. 이외의 값이 입력되면 "잘못된 메뉴를 선택하였습니다."를 출력한다.

- 적용된 배운 내용

1. 멤버 함수 사용(PrintMenu(), InputMenu()

2. switch문

3. 동적 메모리 (new FridgePage(), BasketPage(), MyPage())

4. 상속, 부모 클래스 생성자 호출

5. 가상 함수(MainPage::UpdatePage(), MainPage::OpenPage())

- 코드 스크린샷

```cpp
// 메인 페이지
MainPage::MainPage() : Page("메인 메뉴", { "레시피 검색", "레시피 추천", "냉장고", "장바구니", "마이페이지" }) {}

void MainPage::UpdatePage() {}

int MainPage::OpenPage() {
    PrintMenu("", { "- 1. 이전으로", "0. 프로그램 종료" });
    int command = InputMenu("메뉴 선택 : ");
    //Opened_pages.push_back(main_page);
    //main_page.EnterMenu();
    //레시피 추천
    //레시피 검색
    //냉장고
    //재료 구매
    //마이페이지

    switch (command) {

    case 1:
        //opened_pages.push_back(new SearchPage());
        break;
    case 2:
        //레시피 추천 알고리즘 관련 함수
        //opened_pages.push_back(new RecommendPage());
        break;
    case 3:
        opened_pages.push_back(new FridgePage());
        break;
    case 4:
        opened_pages.push_back(new BasketPage());
        break;
    case 5:
        opened_pages.push_back(new MyPage());
        break;
    case -1:
        DeletePage();
        break;
    case 0:
        return 1;
    default:
        cout << "잘못된 메뉴를 선택하였습니다." << endl;
    }
    return 0;
}
```

세부 기능 3) 레시피 추천 페이지 열기

- 입력 : 레시피 제목 데이터, 레시피 (필요한)재료 데이터

- 출력 : 추천 페이지 출력(기본 메뉴, 페이지 제목, 추천 레시피 출력), 입력에 따른 페이지 생성, 삭제

- 설명

1. 추천 레시피를 추천 알고리즘을 통해 얻는다.

3. 레시피의 번호를 선택하면 선택한 레시피의 레시피 페이지를 opened_pages에 추가한다.

4. 0, -1을 입력하면 '메인페이지 열기', '이전으로'기능을 수행한다.

- 적용된 배운 내용 :

1. 오버라이딩

2. 입, 출력(cout, cin)

3. 동적 메모리

- 코드 스크린샷

```
RecommendPage::RecommendPage() :Page("레시피 추천") {
    vector <string> recipe_title_list = RecommendRecipe();
    for (string recipe_title : recipe_title_list) {
        menu_list.push_back(recipe_title);
    }
}
void RecommendPage::UpdatePage() {
    menu_list.clear();
    vector <string> recipe_title_list = RecommendRecipe();
    for (string recipe_title : recipe_title_list) {
        menu_list.push_back(recipe_title);
    }
}

int RecommendPage::OpenPage() {

    PrintMenu();
    int command = InputMenu("레시피 선택 : ");
    // 선택한 레시피의 레시피 페이지 열기
    if (0 < command && command <= 10) {
        opened_pages.push_back(new RecipePage(menu_list[command - 1]));
    }
    // 메인 페이지 열기
    else if (command == 0) {
        opened_pages.push_back(new MainPage());
    }
    else if (command == -1) {
        DeletePage();
        return 0;
    }
    else {
        cout << "잘못된 메뉴를 선택하였습니다." << endl;
    }
    return 0;
}
```

세부 기능 3) 검색 페이지 열기

- 입력 : 레시피 제목 데이터, 레시피 (필요한)재료 데이터, 검색 텍스트

- 출력 : 검색 페이지 출력(기본 메뉴, 페이지 제목, 검색된 레시피 출력), 입력에 따른 페이지 생성, 삭제

- 설명

1. 검색 텍스트를 입력 받는다.

2. 검색 텍스트를 기반으로 레시피를 검색하여 레시피를 메뉴로 출력한다.

3. 레시피의 번호를 선택하면 선택한 레시피의 레시피 페이지를 opened_pages에 추가한다.

4. 0, -1을 입력하면 '메인페이지 열기', '이전으로'기능을 수행한다.

- 적용된 배운 내용 :

1. 오버라이딩

2. 입, 출력(cout, cin)

3. 동적 메모리

- 코드 스크린샷

```cpp
SearchPage::SearchPage() : Page("레시피 검색") {}
void SearchPage::UpdatePage() {}
int SearchPage::OpenPage() {
    // 레시피 검색
    cout << endl << endl;
    cout << "--------------------------------" << endl;
    cout << title << endl << endl << "검색 : ";
    string search_text;
    cin >> search_text;
    // 검색 단어를 기반으로 레시피 메뉴 출력
    menu_list = SearchRecipe(search_text);
    PrintMenu();
    int command = InputMenu("레시피 선택 : ");
    // 선택한 레시피의 레시피 페이지 열기
    if (0 < command && command <= menu_list.size()) {
        opened_pages.push_back(new RecipePage(menu_list[command - 1]));
    }
    else if (command == 0) {
        opened_pages.push_back(new MainPage());
    }
    else if (command == -1) {
        DeletePage();
        return 0;
    }
    else {
        cout << "잘못된 메뉴를 선택하였습니다." << endl;
    }
    return 0;
}
```

세부 기능 3) 레시피 페이지 열기

- 입력 : 레시피 제목

- 출력 : 레시피 페이지 출력(기본 메뉴, 재료, 요리 과정, 평점 출력), 입력에 따른 페이지 생성, 삭제

- 설명

1. 레시피 제목을 바탕으로 레시피 페이지의 레시피 멤버 변수를 초기화한다.

2. 레시피 멤버 변수를 바탕으로 레시피 제목, 레시피 재료, 요리 과정, 평점을 출력한다.

3. 재료 번호를 선택하면 선택한 재료의 재료 페이지를 opened_pages에추가한다.

4. 0, -1을 입력하면 '메인페이지 열기', '이전으로'기능을 수행한다.

5. -2를 입력하면

- 적용된 배운 내용 :

1. 오버라이딩

2. 멤버 변수 초기화(recipe(TitleToReciepe(recipe_title)))

- 코드 스크린샷

```
RecipePage::RecipePage(string recipe_title) :Page("필요한 재료"), recipe(TitleToRecipe(recipe_title)) {
    // 메뉴 : 필요한 재료
    vector <string> ingredients = recipe.GetIngredients();
    for (string ingredient : ingredients) {
        menu_list.push_back(ingredient);
    }
}
 void RecipePage::UpdatePage(){}

int RecipePage::OpenPage(){
    // 페이지 제목 : 레시피 제목
    cout << endl << endl;
    cout << "------------------------------" << endl;
    cout << recipe.GetTitle();
    // 필요한 재료들 출력
    PrintMenu();
    // 요리 과정 출력
    cout << endl << "요리 과정" << endl << endl;
    int i = 1;
    vector <string> process_list = recipe.GetProcessList();
    for (string process : process_list) {
        cout << " " << i << "." << process << endl;
        i++;
    }
    // 평점 출력
    map <string, int> ratings = recipe.GetRatings();
    cout << endl << "평점" << endl;
    vector <string> category_list { "맛", "가성비", "조리시간", "난이도" };
    for (int i = 0; i < CATEGORY_SIZE; i++) {
        string category = category_list[i];
        cout << " " << category << ":" << ratings[category] << endl;
    }
    // 메뉴로 재료들을 입력
    cout << endl << "-2. 평점 입력" << endl << endl;
    int command = InputMenu("\n재료 선택 : ");
    if (1 <= command && command <= menu_list.size()) {
        Ingredient* ingredient_to_watch = NameToIngredient(menu_list[command - 1]);
        opened_pages.push_back(new IngredientPage(ingredient_to_watch));
    }
    else if (command == 0) { opened_pages.push_back(new MainPage()); }
    else if (command == -1) {
        DeletePage();
        return 0;
    }
    //  -2 입력시 평점 변경
    else if (command == -2) {
        recipe.InputRating();
    }
    else {
        cout << "잘못된 메뉴를 선택하였습니다." << endl;
    }
    return 0;
}
```

세부기능 4) 냉장고 페이지 열기

- 입력 : 냉장고 속 재료(Ingredient::fridge)

- 출력 : 냉장고 페이지 출력(기본 메뉴, 페이지 제목, 냉장고 재료 메뉴 출력), 입력 메뉴에 따른 재료 페이지 생성, 페이지 삭제

- 설명

1. 냉장고 속 재료들의 이름을 메뉴 목록에 저장한다.

2. 냉장고 페이지를 출력한다.(기본 메뉴, 페이지 제목, 냉장고 속 재료)

3. 세부 정보를 볼 재료를 입력받는다.

4. 냉장고 재료 번호를 입력받으면 해당 재료를 '재료 페이지'의 인수로 전달한 뒤, '재료 페이지'를 '열린 페이지 목록'에 추가한다.

5. 입력이 0이면 '메인 페이지'를 '열린 페이지 목록'에 추가한다.

6. 입력이 -1이면 '열린 페이지 목록'의 마지막 페이지(냉장고 페이지)를 삭제한다.

7. 이외의 번호 입력시 "잘못된 메뉴를 선택하였습니다."를 출력한다.

- **적용된 배운 내용**

1. 범위 기반 for 문

2. 정적 멤버 변수 (Ingredient::fridge)

3. 포인터 (Ingredient* ingredient, ingredient_to_watch)

4. 생성자, 접근자(GetName())

5. 벡터(menu_list.push_back(), .clear(), size())

6. 논리 연산자(&&), 관계 연산자 (<=)

7. 상속, 부모 클래스 생성자 호출

8. 가상 함수(FridgePage::UpdatePage(), FridgePage::OpenPage())

- 코드 스크린샷

1. 냉장고 재료의 이름을 저장할 벡터를 선언한다.

2. 냉장고페이지를 생성한다.

3. 재료가 저장되어있는 냉장고 벡터에서 재료를 가져와 이름을 선언한 벡터에 추가한다.

4. 기본메뉴와 추가 메뉴를 출력한다.

5. 메뉴를 입력받는다.

```
FridgePage::FridgePage() : Page("냉장고") {
    for (Ingredient* ingredient : Ingredient::fridge) {
        string ingredient_name = ingredient->GetName();
        menu_list.push_back(ingredient_name);
    }
}
void FridgePage::UpdatePage() {
    menu_list.clear();
    for (Ingredient* ingredient : Ingredient::fridge) {
        string ingredient_name = ingredient->GetName();
        menu_list.push_back(ingredient_name);
    }
}

int FridgePage::OpenPage() {
    PrintMenu("냉장고가 비어있습니다.");
    int command = InputMenu("재료 선택 : ");
    if (1 <= command && command <= Ingredient::fridge.size()) {
        Ingredient* ingredient_to_watch = Ingredient::fridge[command - 1];
        opened_pages.push_back(new IngredientPage(ingredient_to_watch));
    }
    else if (command == 0) { opened_pages.push_back(new MainPage()); }
    else if (command == -1) {
        DeletePage();
        return 0;
    }
    else {
        cout << "잘못된 메뉴를 선택하였습니다." << endl;
    }
    return 0;
}
```

세부기능 5) 장바구니 페이지

- 입력 : 냉장고 속 재료(Ingredient::fridge)

- 출력 : 장바구니 페이지 출력(기본 메뉴, 페이지 제목, 구매 메뉴 출력), 재료 멤

버 변수 갱신, 페이지 삭제

- 설명

1. 냉장고 재료을 중 in_basket이 true인 재료들만 basket_list에 추가한다.

2. basket_list의 재료들의 이름을 menu_list에 추가한다.

3. 장바구니 페이지를 출력한다.(기본 메뉴, 페이지 제목, 장바구니 재료)

4. 장바구니에 담긴 상품의 구매 의사를 묻고 입력받는다.

5. Y(y)를 출력하면 장바구니의 재료들의 상태를 in_basket = false, purchased = true로 갱신하고 구매 완료 메시지를 출력한다.

6. N(n) 입력 시 마지막 페이지(장바구니 페이지)를 '열린 페이지 목록'에서 삭제한다.

7. 이외의 입력에 대하여 '잘못 입력' 메시지를 출력한다.

- **적용된 배운 내용**

- 코드 스크린샷

1. 범위 기반 for 문

2. 정적 멤버 변수 (Ingredient::fridge)

3. 포인터 (Ingredient* ingredient)

4. 생성자, 접근자(GetInBasket()), 설정자(SetInBasket(), SetPurchased())

5. 벡터(menu_list.push_back(), .clear(), size())

6. 논리 연산자(||)

7. 상속, 부모 클래스 생성자 호출

8. 가상 함수(BasketPage::UpdatePage(), BaksetPage::OpenPage())

9. 오버라이딩 (BasketPage::InputMenu())

```cpp
BasketPage::BasketPage() : Page("장바구니") {
    for (Ingredient* ingredient : Ingredient::fridge) {
        bool in_basket = ingredient->GetInBasket();
        if (in_basket == true) { basket_list.push_back(ingredient); }
    }
    for (Ingredient* ingredient : basket_list) {
        string ingredient_name = ingredient->GetName();
        menu_list.push_back(ingredient_name);
    }
}
void BasketPage::UpdatePage(){
    basket_list.clear();
    menu_list.clear();
    for (Ingredient* ingredient : Ingredient::fridge) {
        bool in_basket = ingredient->GetInBasket();
        if (in_basket == true) { basket_list.push_back(ingredient); }
    }
    for (Ingredient* ingredient : basket_list) {
        string ingredient_name = ingredient->GetName();
        menu_list.push_back(ingredient_name);
    }
}
char BasketPage::InputMenu(string message) {
    char command = 0;
    cout << message;
    cin >> command;
    return command;
}
int BasketPage::OpenPage() {
    PrintMenu("장바구니에 담긴 상품이 없습니다.", {});
    char command = InputMenu("장바구니에 담긴 상품들을 바로 구매하시겠습니까? (Y / N) : ");
    if (command == 'Y' || command == 'y') {
        for (Ingredient* ingredient : basket_list) {
            ingredient->SetPurchased(true);
            ingredient->SetInBasket(false);
        }
        cout << "구매가 완료되었습니다." << endl;
    }
    else if (command == 'N' || command == 'n'){
        opened_pages[0]->DeletePage();
        return 0;
    }
    else {
        cout << "잘못된 메뉴를 선택하였습니다." << endl;
    }
    return 0;
}
```

세부 기능 5) 구매 페이지

- 입력 : 구매할 재료

- 출력 : 구매 페이지 출력(기본 메뉴, 페이지 제목, 구매 메뉴 출력), 재료 멤버

변수 갱신, 페이지 삭제

- 설명

1. 구매할 재료와 재료의 이름을 각각 item_to_buy와 menu_list에 추가한다.

2. 구매 페이지를 출력한다.(기본 메뉴, 페이지 제목, 구매할 재료)

3. 해당 상품의 구매 의사를 묻고, 입력받는다.

4. Y(y)입력시 해당 재료의 Purchased = true로 갱신하고 구매 완료 메시지를 출력한 후, 구매 페이지를 삭제한다.

5. N(n)입력시 '열린 페이지 목록'에서 구매 페이지를 삭제한다.

6. 이외의 문자 입력 시 '잘못 입력' 메시지를 출력한다.

- 적용된 배운 내용

1. call by reference(Ingredient* item_to_buy)

2. 클래스 포인터에서 접근자, 생성자 호출(item_to_buy->GetName(), SetPurchased())

5. 벡터(menu_list.push_back())

6. 논리 연산자(||)

7. 상속, 부모 클래스 생성자 호출

8. 가상 함수(BuyPage::UpdatePage(), BuyPage::OpenPage())

9. 오버라이딩 (BuyPage::InputMenu())

상속, 부모 클래스의 생성자 호출

- 코드 스크린샷

```cpp
BuyPage::BuyPage(Ingredient* item_to_buy) : Page("바로 구매") {
    this->item_to_buy = item_to_buy;
    menu_list.push_back(item_to_buy->GetName());
}
void BuyPage::UpdatePage(){}
char BuyPage::InputMenu(string message){
    char command = 0;
    cout << message;
    cin >> command;
    return command;
}
int BuyPage::OpenPage() {
    PrintMenu("",{});
    char command = InputMenu("위 상품을 바로 구매하시겠습니까? (Y / N) : ");
    if (command == 'Y' || command == 'y') {
        item_to_buy->SetPurchased(true);
        cout << "구매가 완료되었습니다." << endl;
        DeletePage();
    }
    else if (command == 'N' || command == 'n') {
        DeletePage();
    }
    else {
        cout << "잘못된 메뉴를 선택하였습니다." << endl;
    }
    return 0;
}
MyPage::MyPage() : Page("마이 페이지") {}
```

세부 기능 6) 마이페이지

- 입력 : 유저 인스턴스(전역 변수)

- 출력 : 마이페이지 출력(기본 메뉴, 페이지 제목, 유저의 포인트 출력), 입력 메뉴에 따른 페이지 열기(메인 페이지), 페이지 삭제(마이페이지)

- 설명

1. 구매 페이지와 유저의 포인트를 출력한다.

2. 0을 입력하면 메인페이지를 '열린 목록 페이지'에 추가한다.

3. -1을 입력하면 마이페이지를 삭제한다.

- **적용된 배운 내용**

1. 상속, 부모 클래스의 생성자 호출

2. 가상 함수(MyPage::OpenPage())

4. 오버라이딩(MyPage::PrintMenu())

5. 범위 기반 for 문

- 코드 스크린샷

```
MyPage::MyPage() : Page("마이 페이지") {}


void MyPage::PrintMenu(const vector <string>& basic_menu_list) {
    cout << endl << endl;
    cout << "--------------------------------" << endl;
    int i = 1;
    for (string basic_menu : basic_menu_list) {
        cout << basic_menu << "\t";
        if (i % 2 == 0) {
            cout << endl;
        }
        i++;
    }
    cout << "\n\n" << endl;
    cout << title << endl << endl;

    cout << "포인트 : " << user.GetPoint() << endl << endl;
}
int MyPage::OpenPage() {
    PrintMenu();
    int command = InputMenu();
    switch (command) {
    case 0:
        opened_pages.push_back(new MainPage());
        break;
    case -1:
        DeletePage();
        break;
    }
    return 0;
}
```

기능 2) 재료 기반 레시피 추천

기본 시스템(main 함수) 0

- 입력 : 입력받은 메뉴

- 출력 : 메인 페이지 열기, 입력 메뉴에 따른 페이지 열기, 프로그램 종료

- 설명

- 적용된 배운 내용 :

- 코드 스크린샷

기능 3) 레시피 검색

레시피 제목 데이터, 레시피 (필요한)재료 데이터 로드

- 입력 : 레시피 제목 파일, 레시피 (필요한)재료 파일

- 출력 : 레시피 제목 데이터, 레시피 (필요한)재료 데이터

- 설명

1. 레시피 제목, 레시피 재료가 포함된 파일을 로드한다.

2. 각 줄의 ','기준으로 첫번째 텍스트는 제목으로 recipe_name_list에 추가한다.

3. 다음 텍스트들은 재료들로 ingredients에 추가한다.

4. 레시피 제목과 해당하는 ingredients들을 map으로 묶어 ingredients_data에 추가한다.

5. 레시피 제목들(recipe_name_list)과 레시피 재료들(ingredients_data)데이터를 반환한다.

- 적용된 배운 내용

1. 파일 입출력, 파일 닫기

2. cerr

3. vector, map

- 코드 스크린샷

```
vector <string> ReadRecipeTitleDate() {
    ifstream recipe_contents_data{ "recipe_contents_data.csv" };
    if (!recipe_contents_data.is_open()) {
        std::cerr << "Error" << std::endl;
    }
    vector <string> recipe_name_list;
    string line;
    while (getline(recipe_contents_data, line)) {
        istringstream iss{ line };
        string recipe_name;
        getline(iss, recipe_name, ',');
        recipe_name_list.push_back(recipe_name);
    }
    recipe_contents_data.close();
    return recipe_name_list;
}
map <string, vector <string>> ReadRecipeIngredientsData() {
    ifstream recipe_ingredients_data{ "recipe_ingredients_data.csv" };
    if (!recipe_ingredients_data.is_open()) {
        std::cerr << "Error" << std::endl;
    }
    map <string, vector <string>> ingredients_data;
    string line;
    while (getline(recipe_ingredients_data, line)) {
        istringstream iss{ line };
        string data;
        getline(iss, data, ',');
        string recipe_title = data;
        vector <string> ingredients;
        while (getline(iss, data, ',')) {
            if (data == "") break;
            ingredients.push_back(data);
        }
        ingredients_data[recipe_title] = ingredients;
    }
    recipe_ingredients_data.close();
    return ingredients_data;
}
```

레시피 검색

- 입력 : 레시피 제목 데이터, 레시피 (필요한)재료 데이터, 검색 텍스트

- 출력 : 검색된 레시피 리스트

- 설명

1. 레시피 제목 데이터, 레시피 (필요한)재료 데이터를 로드한다.

2. 각 레시피에 대하여 레시피 제목ㅁ에 검색 텍스트를 포함하면 search_list에 추가한다.

3. search_list에 추가되지 않은 레시피에 대하여 해당 레시피의 재료 중 검색 텍스트를 포함하는 레시피를 search_list에 추가한다.

4. search_list를 반환한다.

- 적용된 배운 내용

1. 함수 호출

2. vector, map

3. for, if문, 범위 기반 for 문

4. 문자열 관련 메소드(size(), substr())

5. 명시적 형변환 ((int))

- 코드 스크린샷

```
vector <string> SearchRecipe(string search_text) {
    vector <string> search_list;
    vector <string> recipe_title_list = ReadRecipeTitleDate();
    map <string, vector<string>> recipe_ingredients_data = ReadRecipeIngredientsData();
    // 각 레시피에 대하여 레시피 제목에 검색 단어를 포함하면 search_list에 추가
    for (string recipe_title : recipe_title_list) {
        if (CheckIncludeText(recipe_title, search_text)) {
            search_list.push_back(recipe_title);
        }
    }
    // 각 레시피에 대하여
    for (const auto& title_ingredients : recipe_ingredients_data) {
        string recipe_title = title_ingredients.first;
        vector <string> ingredients = title_ingredients.second;
        bool is_in_list = false;
        // 해당 레시피가 search_list에 추가되었다면 해당 레시피에 대한 검색 종료
        for (string searched_title : search_list) {
            if (recipe_title == searched_title) {
                is_in_list = true;
                break;
            }
        }
        // 해당 레시피의 재료 중에 검색 단어를 포함하는 레시피를 search_list에 추가
        if (is_in_list == false) {
            for (string ingredient : ingredients) {
                if (CheckIncludeText(ingredient, search_text)) {
                    search_list.push_back(recipe_title);
                }
            }
        }
    }
    return search_list;
}
```

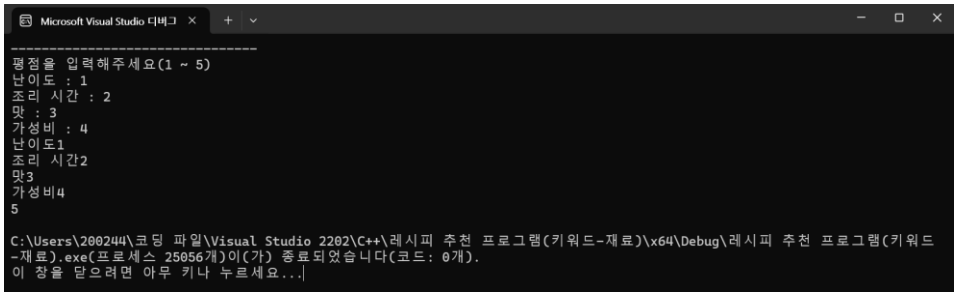
```
bool CheckIncludeText(string text, string check_text) {
    for (int i = 0; i <= (int) text.size() - (int) check_text.size(); i++) {
        size_t start_index = i;
        size_t len_text = check_text.size();
        string sub_text = text.substr(start_index, len_text);
        if (sub_text == check_text) return true;
    }
    return false;
}
```

2) 테스트 결과

(1) 항목별 평점 수집

- 설명 : recipe 인스턴스에 InputRating()메서드를 사용했을 때, 엽력화면을 정상적으로 출력하는지 확인하고, rating 멤버 변수에 항목별 평점이 정상적으로 저장되었는지 확인하고, 유저의 포인트가 정상적으로 적립되었는지 확인한다.

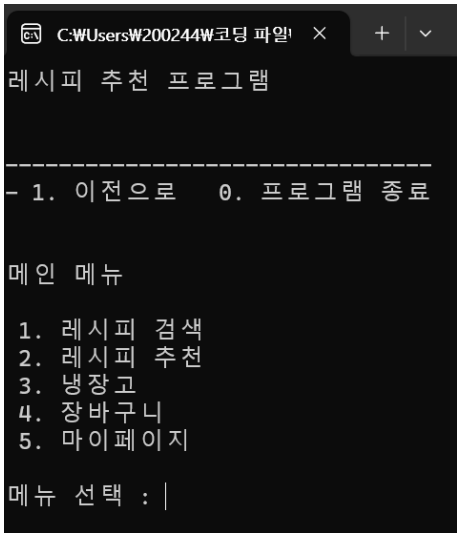
- 테스트 결과 스크린샷



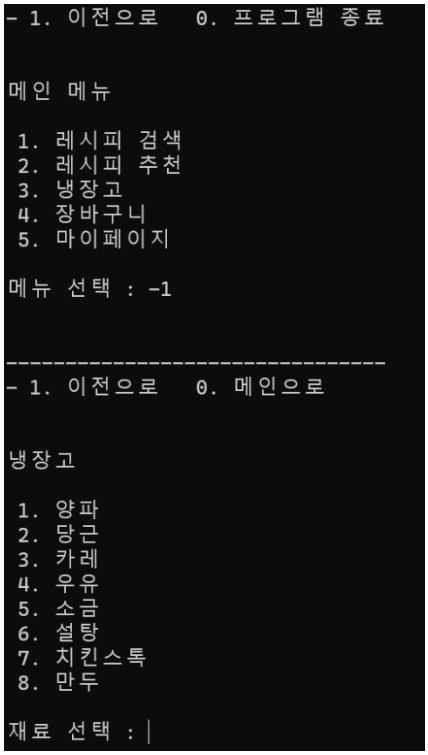
기능 7) 페이지 관리

(1) 메인 페이지 열기

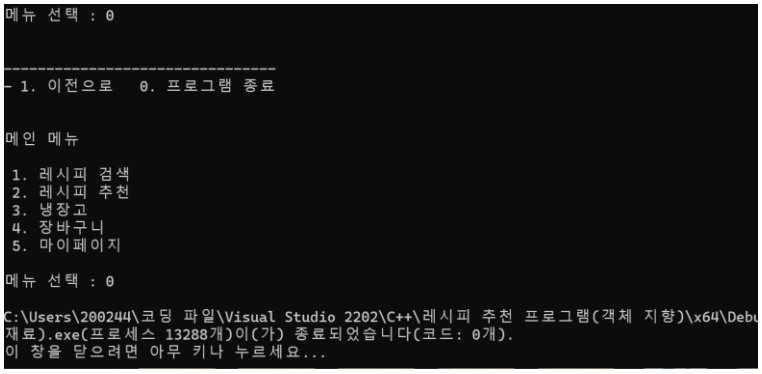
테스트 기능 : 프로그램 시작 시 메인 페이지 (기본 메뉴, '메인 메뉴' 제목, 메인 메뉴)가 정상적으로 출력



테스트 기능 : 메인 페이지에서 -1 입력 시 이전 페이지로 돌아감



테스트 기능 : 메인 페이지에서 0 입력 시 프로그램 종료



(2) 냉장고 페이지 열기

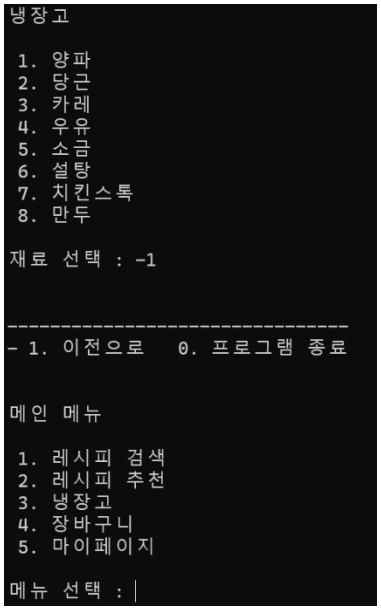
테스트 기능 : 메인 페이지에서 3 입력 시 냉장고 페이지로 이동, 페이지에 기본 메뉴, 페이지 제목, 냉장고 재료 출력, 재료 입력

```
메뉴 선택 : 3


--------------------------------
- 1. 이전으로   0. 메인으로


냉장고

 1. 양파
 2. 당근
 3. 카레
 4. 우유
 5. 소금
 6. 설탕
 7. 치킨스톡
 8. 만두

재료 선택 :
```

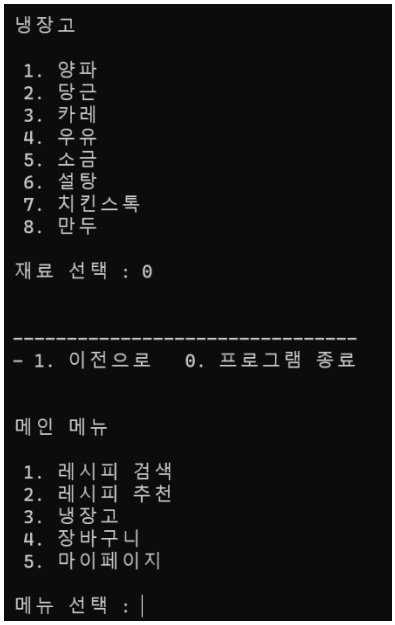
테스트 기능 : 재료 번호 입력 시 해당 재료 페이지로 이동

```
재료 선택 : 1


--------------------------------
- 1. 이전으로   0. 메인으로
재료 관리

양파

1, 선호 재료로 지정

재고 상태 : 소진 (없음)
소진 임박
충분함
2, 재료 추가
3, 재료 소진 임박
4, 재료 소진

재료 구매
5, 장바구니에 담기
6, 바로 구매

메뉴 선택 :
```

테스트 기능 : 냉장고 페이지에서 -1 입력 시 이전 페이지로 이동

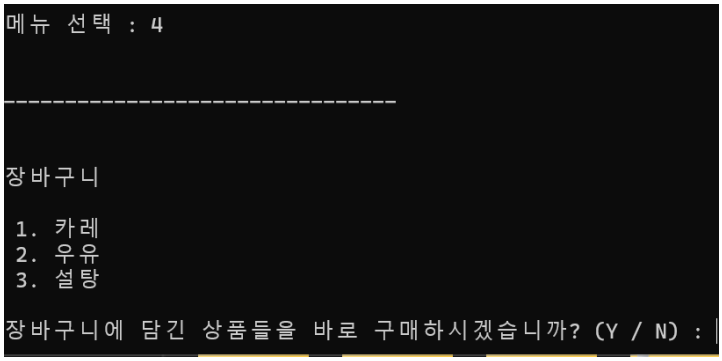


테스트 기능 : 냉장고 페이지에서 0 입력 시 메인 페이지로 이동

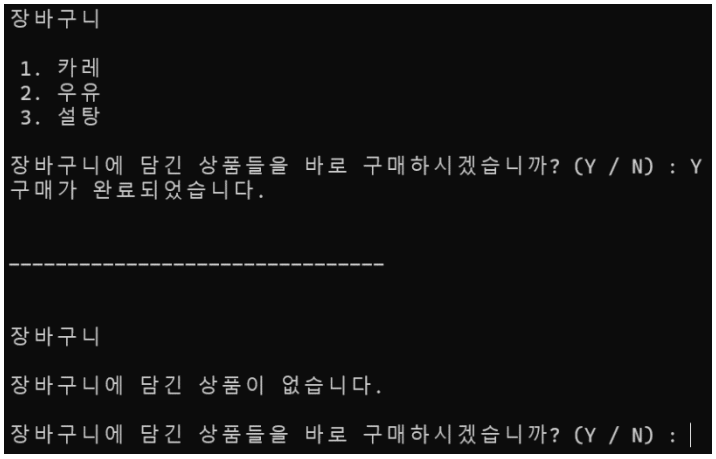


(3) 재료 구매 페이지 열기

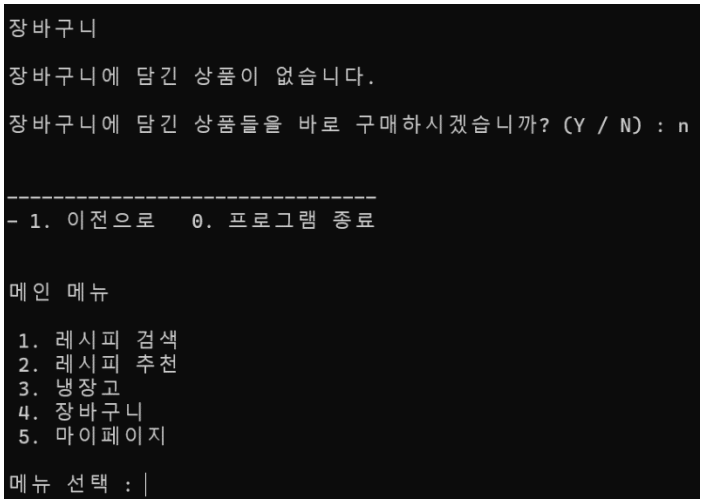
테스트 기능 : 메인 페이지에서 4 입력 시 장바구니 페이지로 이동, 페이지에 페이지 제목, 장바구니 리스트 출력, 구매 의사 입력



테스트 기능 : Y(y) 입력 시 해당 상품을 구매, (이후 장바구니가 비게 됨.)



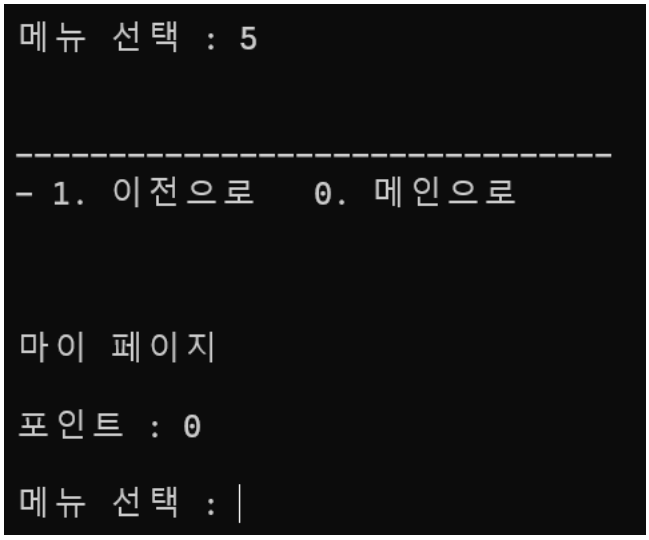
테스트 기능 : N(n) 입력 시 이전 페이지로 이동



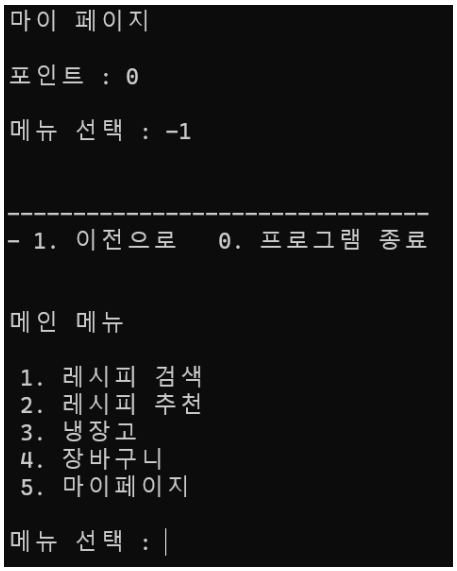
(4) 마이페이지 열기

테스트 기능 : 메인 페이지에서 5 입력 시 마이페이지로 이동, 마이페이지에 기본

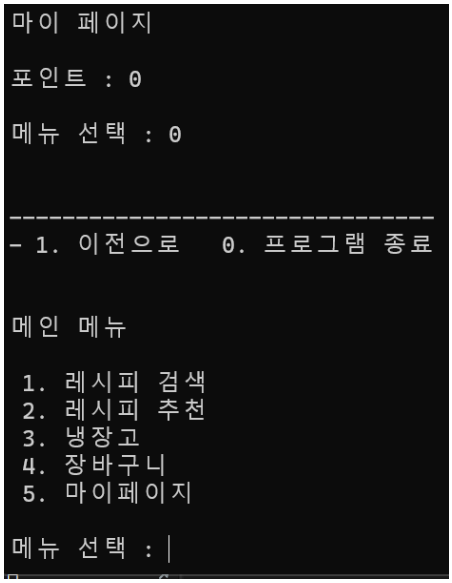
메뉴, 페이지 제목, 포인트 출력, 메뉴 입력



테스트 기능 : 마이 페이지에서 -1 입력 시 이전 페이지로 이동

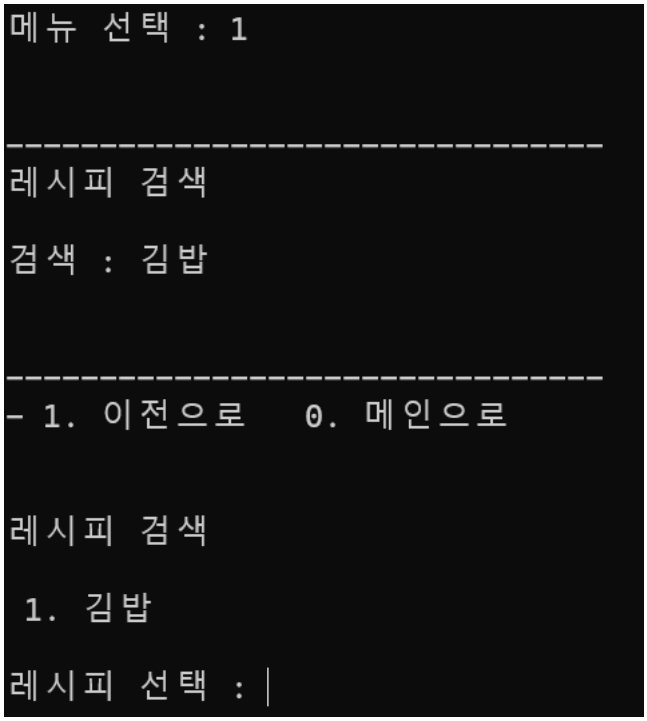


테스트 기능 : 마이 페이지에서 0 입력 시 메인 페이지로 이동

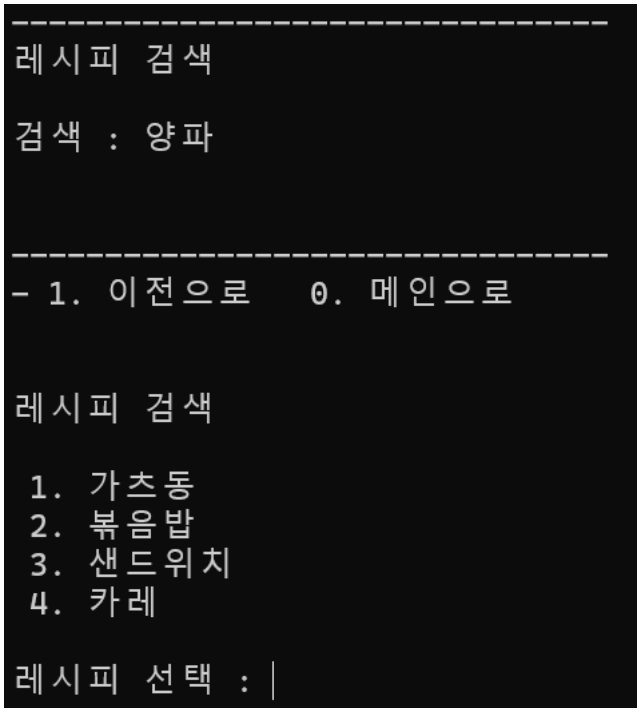


(5) 레시피 검색 페이지 열기

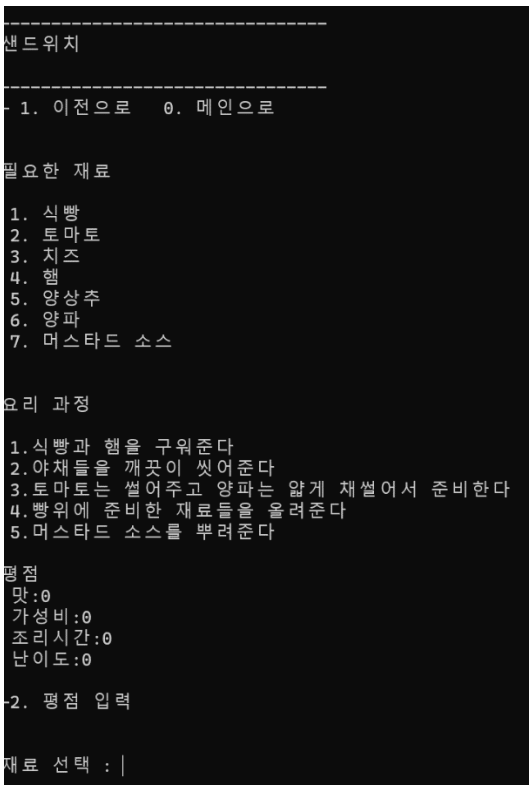
테스트 기능 : 요리명 검색 시 해당 레시피 제목 출력



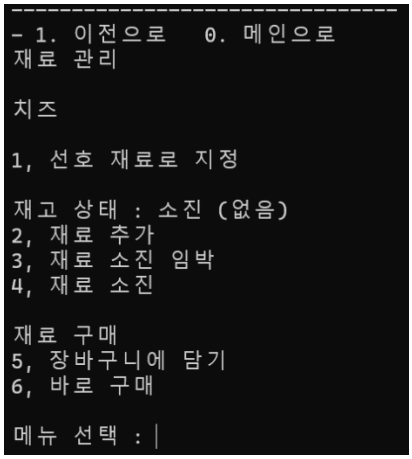
테스트 기능 : 재료명 검색 시 해당 재료를 포함한 레시피 제목 출력



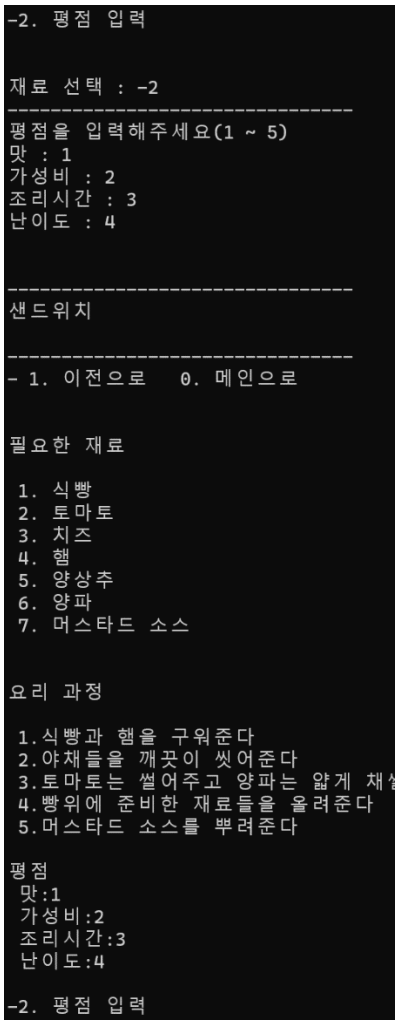
테스트 기능 : 레시피의 번호 입력 시 해당 레시피의 재료, 과정, 평점 출력



테스트 기능 : 재료 번호 입력 시 해당 레시피의 재료 페이지 출력



테스트 기능 : -2 입력시 평점 입력 페이지 출력, 평점 입력 후 평점 업데이트

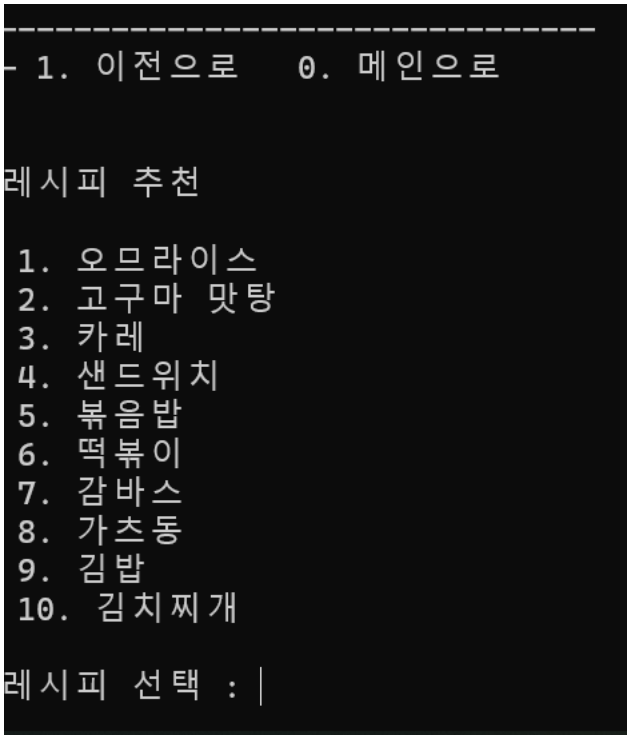


(6) 레시피 추천 페이지 열기

테스트 기능 : 레시피를 냉장고 기반 재료, 선호도 기반으로 추천하였는지 확인

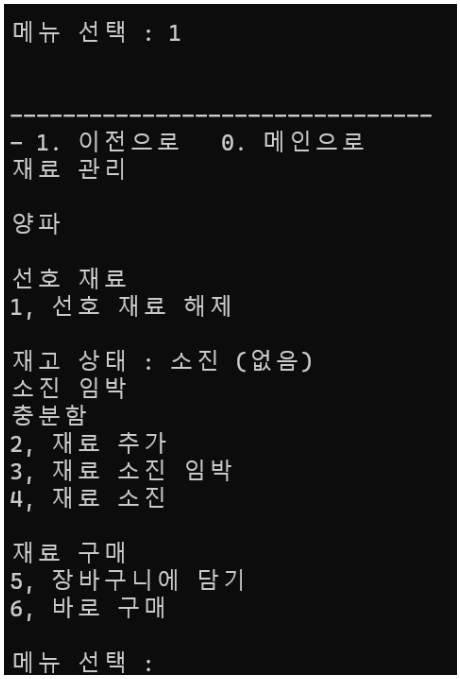
현재 냉장고에는 당근, 우유, 소금, 설탕, 떡이 있다.

이 재료들을 바탕으로 오므라이스, 고구마 맛탕이 가장 구매해야 하는 재료가 적기 때문에 오므라이스가 1번, 2번, 카레는 구매해야 하는 재료는 비교적 많지만 냉장고 재료로 충분히 만들 수 있으므로 3번이다.

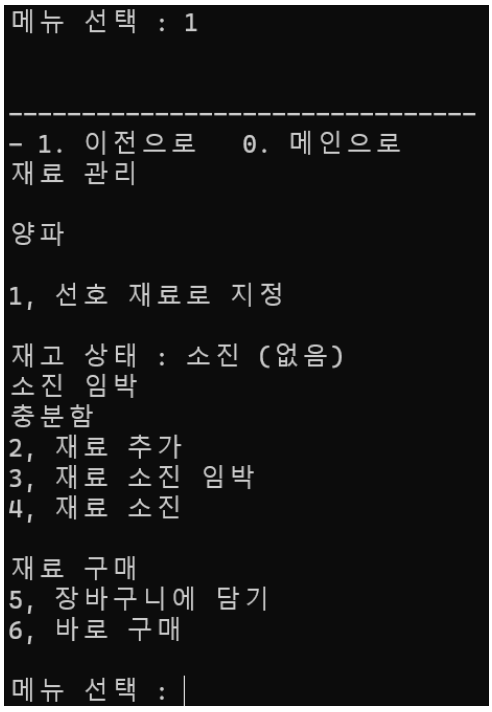


기능 6) 재료 관리(재료 페이지)

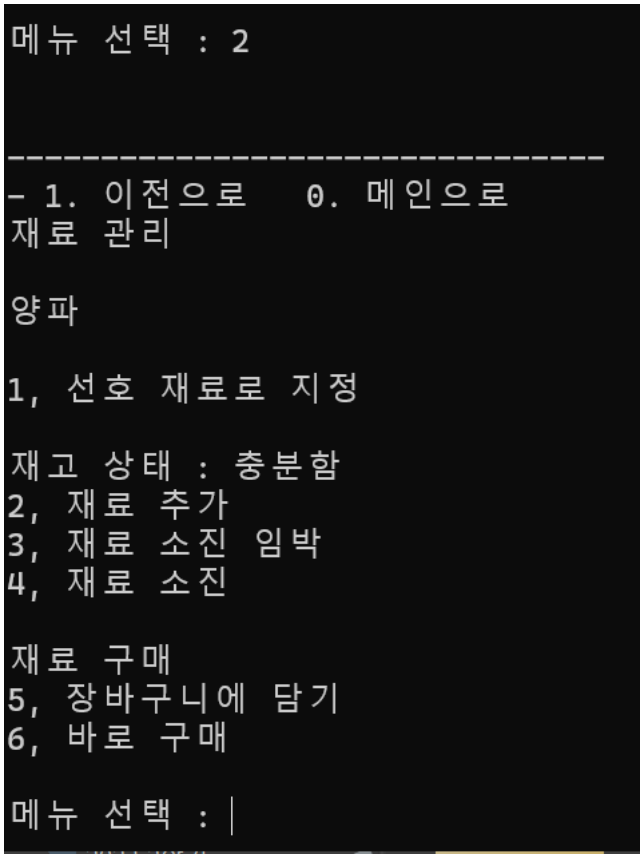
테스트 기능 : 선호 재료로 지정 선택시 (1) 선호 재료로 지정되고, 선호 재료 해제 출력



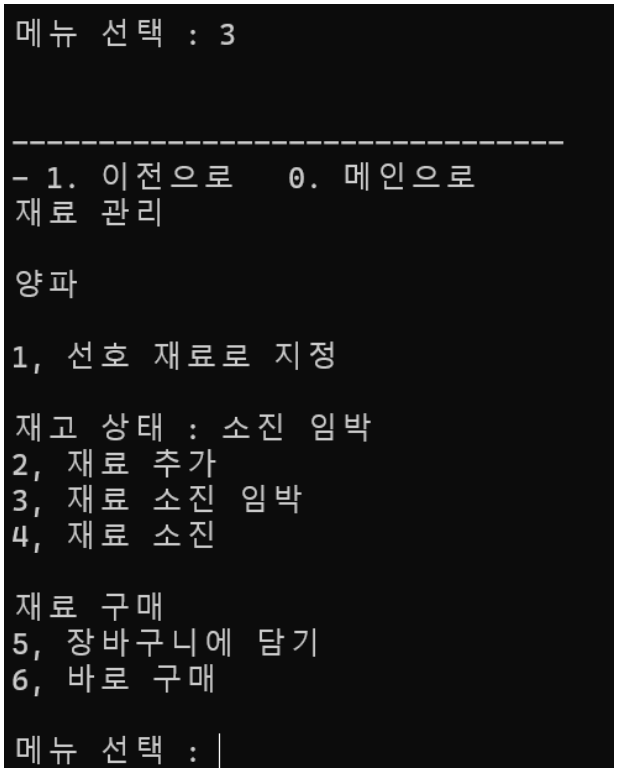
테스트 기능 : 선호 재료 해제 선택시 (1) 선호 재료에서 해제, 선호 재료로 지정 출력



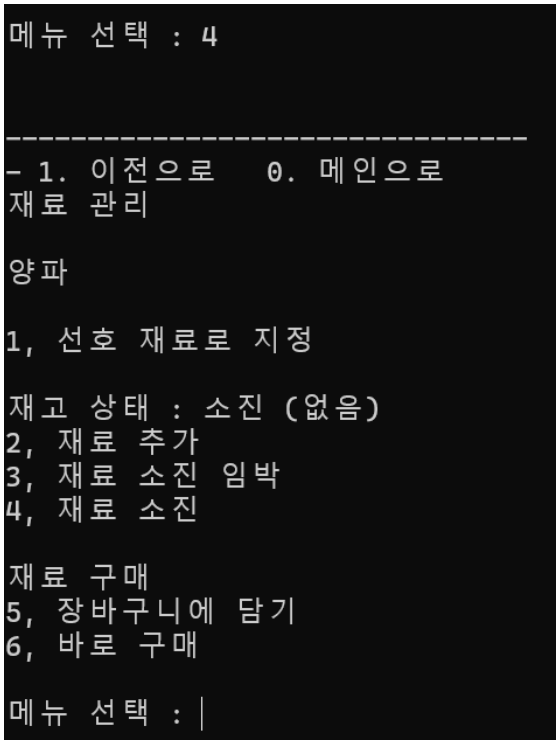
테스트 기능 ; 메뉴 2번 선택시 재고 상태가 충분함으로 변경



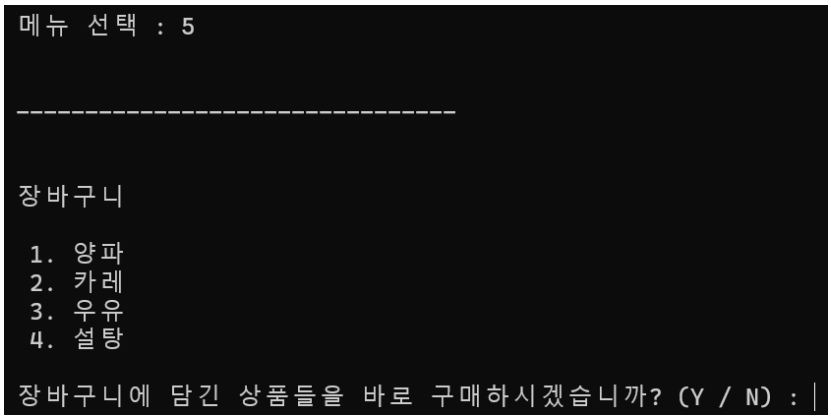
테스트 기능 : 메뉴 3번 선택 시 재고 상태가 재료 소진 임박으로 변경



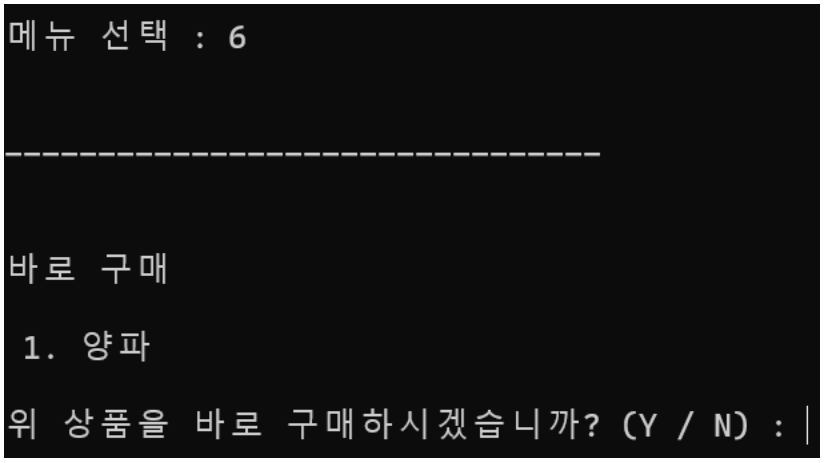
테스트 기능 : 메뉴 4번 선택 시 재고 상태가 소진(없음)으로 변경



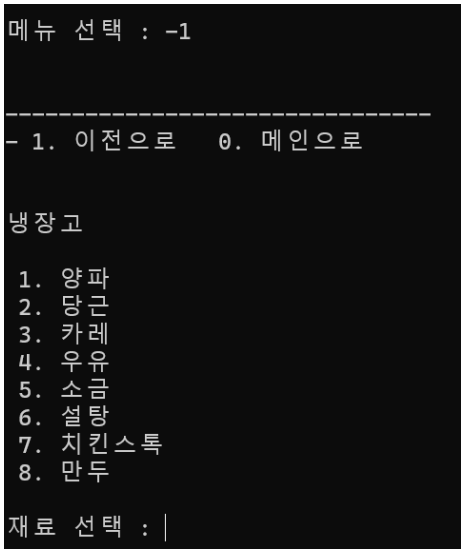
테스트 기능 : 메뉴 5번 선택 시 '양파'를 장바구니에 담고(장바구니 메뉴에 추가), 장바구니 페이지로 이동



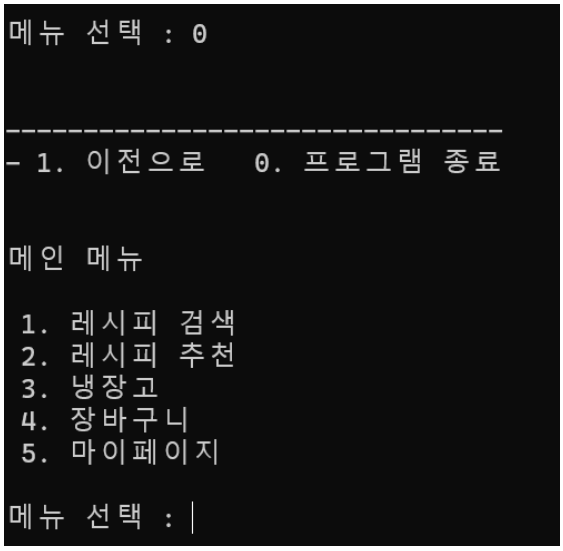
테스트 기능 : 메뉴 6번 선택 시 양파의 바로 구매 페이지로 이동



테스트 기능 : 메뉴 -1 선택 시 이전 페이지로 이동



테스트 기능 : 메뉴 0 선택 시 메인 페이지로 이동



4. 계획 대비 변경 사항

1) 기능 1 삭제

- 이전

기능1 구현

-이후

기능1 구현 취소

-사유

아직 외부에서 정보를 수집할 수 있는 환경이 조성되지 않아 구체적인 코드로 구현하기 힘듦

2) 페이지 관리 기능 추가

- 이전

페이지를 관리하는 클래스나 함수에 대한 계획 없었음.

- 이후

페이지 클래스와 함수 정의

- 사유

이 프로그램은 페이지에서 입력을 받아 새로운 페이지를 출력하고, 정보를 출력해야 하므로 꼭 필요한 기능이라고 생각하여 기능계획에 추가하였다.

2) 기능 3 레시피 검색 기준 일부 삭제

- 이전

요리명, 재료명, 카테고리 기반, 평점 기반으로 기능을 분리하여 검색

- 이후

요리명, 재료명 기반으로 통합 검색

카테고리 기반, 평점 기반 검색 기능 삭제

- 사유

다른 앱을 사용해 본 결과 카테고리 기반, 평점 기반으로 검색하는 것은 유용한 정보를 얻기 힘들기 때문에 두 기능은 삭제, 검색 조건이 줄어들었으므로 요리명, 재료명 기반 검색은 통합하여 검색하는 것으로 변경

2) 기능 5 재료 관리 기능 메뉴 출력 조건 변경

- 이전

재고 상태에 따라 재고 상태 갱신 메뉴 1개, 재료 구매 메뉴 1개를 출력

- 이후

재고 상태와 관계없이 재고 상태 갱신 메뉴 3개, 재료 구매 메뉴 2개 출력, 선호 재료 상태에 따라 지정, 해체로 메뉴 생성

- 사유

이전의 메뉴 출력 조건은 개발자가 사용자의 선택을 제한할 수 있는 여지가 있다. 하지만 사용자의 선택은 예측할 수 없다. 재료가 조금 남았어도 다시 재료를 살 수도 있는 것이고, 충분한 재료를 까먹고 다 쓸때까지 재료의 상태를 갱신하지 않을 수도 있다. 또 냉장고에 있는 재료라도 더 사고 싶을 수도 있다. 따라서 모든 메뉴를 선택할 수 있도록 변경하였다.

5. 느낀 점

처음으로 제대로 된 프로그램을 만들어보는 프로젝트였다. 내가 생각한 기능 계획을 세울 때에는 클래스에 대하여 아무것도 모른 채로 계획을 세웠었다. 그래서 처음 코드를 작성할 때에도 함수 위주의 절차 지향적인 프로그램을 만들고 있었다. 그러나 클래스를 배우고 나서는 이 때 세운 계획이 실제 코드를 작성하는 데에 불편함을 많이 준다는 것을 깨달았다. 어떤 기능을 만들기 위해서는 그보다 더 기초적인 기능들을 만들어야 하는데, 처음 세운 계획은 기능들을 작은 단위들로 나누지 않고, 생각나는 대로 적어서 이를 기능 단위들로 분리하는 것이 많이 힘들었다. 또, 프로그래밍 언어를 새롭게 배우면서 알게 되는 개념들이 바로바로 프로젝트를 진행하는데 적용되다 보니 원래 코드를 대폭 수정해야 하는 일도 잦았고, 개념을 배우면서 코드를 짜는 것이 힘들었다. 처음에 만든 함수를 전부 클래스화 하고, 포인터와 참조자로 매개변수들을 수정해야 하고, 배열을 벡터로 바꾸고, 일일이 변수를 선언해 초기화를 하는 것을 동적 메모리 생성으로 간단히 해결하는 등의 과정 중에 생각한 것을 코드로 짜다 보니 여러 세부적인 오류들이 발생하였다. 이 오류들은 프로그래밍 문법에서 그 문법의 세부적인 특성 때문에 발생하는 오류들이 많았다. 예를 들어, 변수의 size와 관련된 반환 함수들의 반환형은 size_t라는 양의 정수형 타입이라는 것이라던가, map 변수를 범위 기반 for 문으로 불러올 때 pair 타입으로 반환하여 .first와 .second를 사용하여 key와 value를 받아야 하는 것, 동적 메모리의 초기화 방법 등을 새로 배울 수 있었다. 잘 모르는 것들과 오류들을 구글 검색이나 chat GPT를 활용하면서 많이 배울 수 있었다. 이렇게 찾아보면서 강의에서 배운 게 다가 아니라 프로그래밍 언어가 내부에서 어떻게 동작하는지 내부 구조까지 정확히 알아야 내가 원하는 대로 코드를 짤 수 있겠다는 생각이 들었다. 또, 비슷한 오류들이 계속 반복되어 오류들을 검색할 때 블로그에서 이런 오류들에 대한 글을 확인하여 고칠 수 있었다. 이걸 보면서 나도 이런 구체적인 문법이나 오류 내용들을 기록하여 학습 해야겠다는 생각이 들었다. 프로그램을 계속 짜다 보니 점점 프로그래밍에 적합한 사고 과정이 무엇인지 스스로 체득하고 있는 기분이 들었다. 주석을 어떤 때에 달아야 하는 지 잘 몰랐는데, 클래스나 함수에 마우스를 댔을 때 나오는 설명을 보고 이

클래스를 다시 사용할 때 주석을 보고 사용하기 쉽게 하기 위한 것이라는 걸 깨닫게 되었다. 이전에는 어떤 코드가 좋은 코드이고, 어떤 구조로 프로그램을 짜야 할 지 감이 잘 안 잡혔지만, 크드를 만들면서 서로 분리 가능한 기능들은 분리하여 함수화하고, 코드 가독성과 확장성도 계속 고려하다 보니, 이런 것들은 코드를 많이 짜봐야 체득할 수 있겠다고 생각했다. 코드를 짜다 보면 이 과목 이외에 배운 내용들도 활용할 수 있었다. 특히 알고리즘 과목이 생각보다 적용이 많이 되는 게 느껴졌다. 강의 내용 중에 클러스터링 기법도 ai에서 활용되는 것을 알게 되었고, 이 프로그램에서도 안정성이 있는 삽입 정렬이 필요하여 이를 활용하였다. 이 외에도 특정 조건에 부합하는 상황을 탐색하기 위해 사용하는 for문들이 정렬에서 탐색하는 방법이랑 비슷하게 느껴져서 알고리즘도 프로그래밍에 밀접환 관련이 있다는 것을 깨닫고 공부의 필요성을 느끼게 되었다. 이번 학기에는 프로젝트를 2개씩 진행해야 해서, 바빴지만 곧 다가오는 겨울방학에는 이번 프로젝트를 통해 배운 내용들을 다시 정리해보고, 문법, 오류들에 대해 공부하면서 알고리즘 과목도 다시 복습해서 기초를 다져놓으려고 한다. 이번 과목들은 배우는게 빡빡해서 전반적으로 어려웠지만, 그만큼 많이 배울 수 있었고, 더 알차게 배운 것 같아 너무 좋았다.