

C++ 프로그래밍 및 실습

카페 자동화 시스템

최종 보고서

제출일자: 2023.12.24(일)

제출자명: 김보민

제출자학번: 215848

1. 프로젝트 목표

1) 배경 및 필요성

카페 메뉴의 수가 증가하면서 메뉴 선택 시 무엇을 먹을지 고민하는 고객들의 수도 증가함. 이러한 고객들의 고민을 조금이라도 간소화하고자 단계적 메뉴 선택에 관한 자동화 시스템이 필요함. 또한 이러한 자동화 시스템이 잘 갖추어진다면 직원이 고객의 주문을 잘못 받는 실수를 없앨 수 있음.

카페에 사람이 많을 경우, 고객들이 앉을 자리를 못 찾아 다시 돌아가는 경우가 발생함. 이를 효율적으로 관리하고자 고객들이 앉을 좌석을 선택하고, 현재 남은 여석을 보여주는 자동화 시스템이 필요함.

2) 프로젝트 목표

카페 메뉴를 큰 카테고리(음료, 디저트)에서 점차 작은 카테고리(커피, 에이드, 차, 프라푸치노 등)로 줄여 나가며, 가장 작은 카테고리 속 메뉴를 제시하는 프로그램을 만드는 것을 목표로 함. 즉 고객의 선호도를 단계적으로 선택하게 하여 맞춤 메뉴를 제시하는 것을 목표로 함.

카페에서 음식을 섭취하고자 하는 고객들을 대상으로 카페 내 앉을 좌석을 선택하도록 하는 것을 목표로 함. 또한 고객들이 카페에서 나갈 때, 해당 고객이 앉았던 자리는 다시 여석으로 반환되도록 하고자 함.

3) 차별점

기존 카페들은 많은 메뉴들을 그냥 나열하여 보여주지만 함. 이는 고객들이 메뉴를 선택하는데 있어 어려움을 줄 수 있음. 따라서 우리는 고객들이 메뉴를 정할 때, 큰 카테고리를 먼저 선택하게 하고, 선택한 카테고리 속 또다른 작은 카테고리를 선택하게 하며, 이를 반복하여 고객이 원하는 메뉴를 추천하는 것에 있어 기존 카페들과 차별점이 있음.

또한 우리는 카페 내 좌석 발권 시스템을 이용하여 카페의 여석을 관리하고 고객

들의 편의를 높이는 것에 있어 기존 카페들과 차별점이 있음.

2. 기능 계획

1) 기능 1: 카페 메뉴 단계별 제시

- 설명: 카페 메뉴를 음료, 디저트와 같이 큰 카테고리로 나누고, 이들의 하위 또한 여러 카테고리로 나누어, 고객들이 선택할 수 있는 메뉴의 가지 수를 줄여 나감.

(1) 세부 기능 1: 카페의 메뉴와 해당 가격을 제시

- 설명: 전체 메뉴와 가격을 저장하는 코드 생성. 카테고리별 메뉴를 저장할 코드 생성.

(이때 배열 또는 vector, map 중 하나를 사용할 예정)

(2) 세부 기능 2: 메뉴의 단계적 선택

- 설명: if문을 통해 고객이 선택한 큰 카테고리의 하위 카테고리 제시, 이를 고객이 원하는 메뉴를 얻을 때까지 진행.

(3) 세부 기능 3: 추가 토핑 선택과 매장 내 섭취 여부 선택, 결제 가격 제시

- 설명: 토핑(샷, 휘핑크림, 펄 등) 하나 당 300~800원이 추가 됨. Take out을 선택할 시 5% 할인. 이러한 토핑과 매장 내 섭취 여부를 반영한 최종 결제 가격 제시.

2) 기능 2: 카페 내 좌석 선택

- 설명: 메뉴를 선택할 때, 매장 내 섭취를 선택한 고객들을 대상으로 어느 자리에 착석할 것이지를 택하도록 함.

(1) 세부 기능 1: 현재 카페 내 좌석과 여석 제시

- 설명: 좌석을 제시하기 위한 2차원 배열 생성. 각각의 좌석에 좌표를 부여하여 고객이 해당 자리를 찾기 쉽게 함. 고객이 선택한 자리는 표식을 부여하여 다른 고객들이 선택할 수 없도록 함. ("□":선택가능, "■":선택불가능)

(1) 세부 기능 2: 카페에서 음식을 섭취하고 나가는 고객들의 자리를 여석으로 전

환

- 설명: 고객이 카페에서 섭취를 완료하고 나갈 때, 자신이 앉은 자리의 좌표를 입력하고 나가도록 함. 해당 자리의 '사용불가' 표식이 다시 '사용가능' 표식으로 전환.

3. 기능 구현

(1) 카페 메뉴와 해당 가격 제시

- 입력:

Floor f1(1층 좌석), Floor f2(2층 좌석), string choice(선택), vector<string> v1(선택지)

- 출력:

- "1. 메뉴 & 가격 2. 주문 3. 주문 목록 4. 결제 5. 카페 내 여석 6. 퇴실 7. 종료" 선택지 출력
- v1선택지(선택지의 숫자 문자열을 모아둔 벡터)에 없는 문자열을 (choice에) 입력 시, "잘못된 입력입니다. 양식에 맞게 다시 입력해 주세요."라 출력.
- choice에 "1"을 입력할 시, 전체 메뉴와 가격 출력.

- 설명

- UpdateSeat(f1, f2)를 통해 좌석 txt 파일을 읽어와 현재 고객들이 사용 중인 좌석을 업데이트 함.
- SelectChoice(choice) 함수를 통해 choice에 문자열을 입력 받아, CheckInput(v1, choice)를 호출하여 선택지에 존재하는 입력인지 확인. 만약 선택지에 없는 입력일 시, 에러를 던짐. catch(exception& e)가 실행되고 잘못된 입력이라 출력하고, 다시 choice 값을 입력 받음.
- choice에 "1"이 입력되면 TotalMenuPrint()가 호출되어 전체 메뉴와 가격을 출력함.

- 적용된 배운 내용

클래스(상속), 함수, 예외처리, 벡터, while/for 반복문, 조건문

- 코드 스크린샷

```
int main() {
    vector<Menu*> m = { &coffee, &non_coffee, &tea, &smoothie, &frappe, &ade, &dessert };
    int currentMenu:

    vector<FinalOrder> basket: // 주문 예정 메뉴들을 담을 (주문 목록)

    while (1) {
        Floor1 f1;
        Floor2 f2;

        UpdateSeat(f1, f2); // txt파일을 읽어와 사용중인 좌석 업데이트

        bool back = false; // back이 true가 되면 처음으로 돌아옴

        string choice;
        SelectChoice(choice);

        // 1. 메뉴와 가격 출력
        if (choice == "1") {
            TotalMenuPrint();
        }

        // 기능 1
        // try-catch문을 이용한 입력 예외 체크
        void CheckInput(vector<string> v, string input) {
            int count = 0;
            for (string& s : v) {
                if (s == input) count++;
            }
            if (count == 0) {
                throw exception("잘못된 입력입니다. 양식에 맞게 다시 입력해주세요.");
            }
        }

        // 잘못 입력 시 출력
        void PrintError() {
            cout << endl << "잘못된 입력입니다. 양식에 맞게 다시 입력해주세요." << endl;
        }

        // 처음으로 돌아갈 시 출력
        void PrintReset() {
            cout << endl << "처음 화면으로 돌아갑니다." << endl;
        }

        // 띄어쓰기가 없는 문자열로 반환
        string NonSpace(string str) {
            string result;
            for (char ch : str) {
                if (ch != ' ') result += ch;
            }
            return result;
        }

        // 처음에 나오는 선택
        void SelectChoice(string& choice) {
            try {
                cout << endl << "(숫자를 입력하십시오)" << endl;
                cout << "1. 메뉴 & 가격 2. 주문 3. 주문 목록 4. 결제 5. 카페 내 여석 6. 퇴실 7. 종료" << endl;
                cout << " >> ";
                getline(cin, choice);
                choice = NonSpace(choice);
                vector<string> v1 = { "1", "2", "3", "4", "5", "6", "7" }; // 첫 번째 선택지 벡터
                CheckInput(v1, choice); // 선택 & 선택지에 맞지 않은 입력 시 error
            }
            catch (exception& e) { // 잘못된 입력으로 다시 선택
                cout << endl << e.what() << endl;
                SelectChoice(choice);
            }
        }

        // 전체 메뉴 출력
        void TotalMenuPrint() {
            coffee.Print();
            non_coffee.Print();
            tea.Print();
            smoothie.Print();
            frappe.Print();
            ade.Print();
            dessert.Print();
        }
    }
}
```

```

// 선택된 좌석 불러오기
void UpdateSeat(Floor1& f1, Floor2& f2) {
    ifstream is_1{ "seats_floor_1.txt" };
    string s_1;
    while (is_1 >> s_1) {
        f1.SelectSeats(s_1);
    }
    is_1.close();

    ifstream is_2{ "seats_floor_2.txt" };
    string s_2;
    while (is_2 >> s_2) {
        f2.SelectSeats(s_2);
    }
    is_2.close();
}

```

(2) 주문을 받고 주문 목록에 저장

- 입력:

string choice(선택), string order(주문선택), vector<string> v2(주문 선택지), string category(음료 카테고리 선택), vector<string> v3(음료 카테고리 선택지), string menu(카테고리 속 메뉴 선택), int currentMenu(현재 카테고리를 숫자화), vector<Menu*> m(여러 카테고리를 가리키는 포인터 벡터), Cafe selectMenu(선택한 메뉴의 Cafe 객체), string subMenu(서브메뉴 선택), string ice_hot(음료의 온도 선택), vector<string> toppings(추가할 토핑들을 저장할 벡터), FinalOrder f(주문을 저장할 객체)

- 출력:

- choice에 "2"를 입력할 시, "1. 음료 2. 디저트 3. 처음으로 돌아가기" 선택지가 출력됨.

해당 선택지(v2)에 없는 문자열을 (order에) 입력 시, "잘못된 입력입니다. 양식에 맞게 다시 입력해 주세요."라 출력.

- order에 "3"을 입력할 시, "처음 화면으로 돌아갑니다."가 출력되고, 가장 처음의 선택지인 "1. 메뉴 & 가격 2. 주문 3. 주문 목록 4. 결제 5. 카페 내 여석 6. 퇴실 7. 종료"가 출력되어 다시 choice에 값을 입력 받음.

- order에 "1"을 입력할 시, 음료들의 카테고리 "1. 커피 2. 논커피 3. 티 4. 스무디 5. 프라페 6. 에이드 7. 처음으로 돌아가기"가 출력.

해당 카테고리(v3)에 없는 문자열을 (category에) 입력 시, "잘못된 입력입니다. 양식에 맞게 다시 입력해 주세요."라 출력.

category에 입력된 문자에 따라 카테고리 속 메뉴들 출력. (ex. "1" -> 커피 메뉴들 출력, "4" -> 스무디 메뉴들 출력)

출력된 메뉴 중 주문하고자 하는 메뉴를 menu에 입력 받음.

menu에 "0"을 입력할 시, "처음 화면으로 돌아갑니다."라 출력되고, 다시 맨 처음의 선택지로 돌아감.

menu에 카테고리 속 메뉴들의 이름을 정확하게 입력하지 않았을 때, "잘못된 입력입니다. 양식에 맞게 다시 입력해 주세요."라 출력.

입력된 menu에 서브 메뉴가 존재할 시, 해당 서브 메뉴들 출력.

subMenu에 서브 메뉴들의 이름을 정확하게 입력하지 않았을 때, "잘못된 입력입니다. 양식에 맞게 다시 입력해 주세요."라 출력.

subMenu에 "0"을 입력할 시, "처음으로 돌아갑니다."라 출력.

currentMenu가 0(커피), 1(논커피), 2(티)인 것은 차가운 음료를 먹을지 뜨거운 음료를 먹을지 물어보는 선택지 출력하여, ice_hot에 문자열을 입력받음. 이외의 것들(스무디, 프라페, 에이드)은 ice_hot에 "ICE"가 저장됨.

ice_hot에 "0"을 입력할 시, "처음 화면으로 돌아갑니다."라 출력.

선택지 외의 다른 문자를 (ice_hot에) 입력 시, "잘못된 입력입니다. 양식에 맞게 다시 입력해 주세요."라 출력.

토피ng 선택지 "1. Size UP(+800원) 2. 샷 추가(+500원) 3. 펄 추가(+500원) 4. 바닐라시럽 추가(+300원) 5. 휘핑크림 추가(+300원) 6. NO" 출력.

토피ng 선택지 외의 다른 문자를 입력할 시 또는 6을 입력하지 않을 시, 오류의 이유와 함께 "토피ng 선택을 리셋합니다. 다시 입력해주세요."가 출력.

올바른 토피ng 선택 이후 추가한 토피ng이 출력되고, 앞에서 선택한 메뉴를 주문 목록에 담았다고 출력.

· order에 "2"를 입력할 시, 디저트에 속하는 메뉴들 출력.

menu에 "0"을 입력할 시, "처음 화면으로 돌아갑니다."라 출력.

menu에 카테고리 속 메뉴들의 이름을 정확하게 입력하지 않았을 때, "잘못된 입력입니다. 양식에 맞게 다시 입력해 주세요."라 출력.

입력된 menu에 서브 메뉴가 존재할 시, 해당 서브 메뉴들 출력.

subMenu에 서브 메뉴들의 이름을 정확하게 입력하지 않았을 때, "잘못된 입력입니다. 양식에 맞게 다시 입력해 주세요."라 출력.

subMenu에 "0"을 입력할 시, "처음으로 돌아갑니다."라 출력.

subMenu에 올바른 입력이 들어가면, 앞에서 선택한 메뉴를 주문 목록에 담았다고 출력.

- 설명

- choice에 "2"가 입력될 시 실행됨.

- SelectOrder(order) 함수를 통해 order에 문자열을 입력 받아, CheckInput(v2, order)을 호출하여 선택지에 존재하는 입력인지 확인. 만약 선택지에 없는 입력일 시, 에러를 던짐. catch(exception& e)가 실행되고 잘못된 입력이라 출력하고 다시 order 값을 입력 받음.

- order에 "3"이 입력되면 PrintReset() 함수를 호출해 처음으로 돌아간다고 출력하고, continue를 통해 가장 바깥의 while문의 첫 하위 문장(처음 선택지)으로 이동함.

- order에 "1"이 입력되면 음료 카테고리가 출력됨. 이때 SelectCategory(category) 함수를 통해 category에 음료 카테고리를 선택해 입력 받는데, CheckInput(v3, drink)를 호출하여 선택지에 존재하는 입력인지 확인. 만약 선택지에 없는 입력일 시, 에러를 던짐. catch(exception& e)가 실행되고 잘못된 입력이라 출력 & 다시 drink 값을 입력 받음.

입력된 category값에 따라 currentMenu에 상수값을 할당하여 m[currentMenu]가 선택된 음료 카테고리를 가리키게 함. m[currentMenu]->Print()를 호출하여, 해당 카테고리 속 메뉴들을 출력함.

SelectMenu(m[currentMenu]->menu, menu, selectMenu) 함수를 사용해 menu에 값을 입력 받고, 해당 메뉴의 Cafe객체를 selectMenu에 저장함. 이때 메뉴 선택지에서 벗어난 입력을 하면 올바른 입력이 아니라 출력되고, SelectMenu() 함수가 다시 호출되어 menu에 값을 다시 입력 받음.

If(!selectMenu.GetSubMenu().empty())를 통해 선택한 메뉴의 서브 메뉴가 존재하는지 확인.

서브 메뉴가 존재한다면 selectMenu.PrintSub()를 호출하여 해당 서브 메뉴를 출력함. SelectSubMenu(selectMenu, subMenu) 함수를 사용해 subMenu에 값을 입력 받음. 이때 서브 메뉴 선택지에서 벗어난 입력을 하면 올바른 입력이 아니라 출력되고,

SelectSubMenu() 함수가 다시 호출되어 subMenu에 값을 다시 입력 받음.

SelectIceHot(currentMenu, ice_hot) 함수 호출을 통해 ice_hot에 값을 입력 받음. 이때 currentMenu가 0, 1, 2일 때만 ice_hot에 값을 입력 받을 수 있음. currentMenu가 3, 4, 5일 때는 ice_hot 값이 항상 "ICE"가 됨.

currentMenu 가 0, 1, 2 일 때 ice_hot 에 "0"이나 "ICE", "HOT"이 아닌 다른 문자열을 입력했다면, 잘못된 입력이라 출력되고 SelectIceHot() 함수가 다시 호출되어 ice_hot 에 다시 값을 입력 받음.

menu, taste, ice_hot에 "0"을 입력할 시, PrintReset()과 continue를 통해 가장 처음 선택지로 돌아가게 함.

서브 메뉴가 존재할 때는 FinalOrder f에 FinalOrder(category, menu, subMenu, ice_hot, selectMenu.GetPrice())를 저장하고, 서브 메뉴가 없을 때는 FinalOrder(category, menu, ice_hot, sub.GetPrice())를 저장함.

SelectToppings(toppings, f)를 통해 토핑을 추가함. 함수를 실행하면 토핑 선택지가 출력되고 함수 안에 있는 지역 변수 inputTopping에 원하는 토핑을 입력 받는데, getline(cin, inputTopping)을 통해 'Wn' 입력 전까지의 문자열을 inputTopping에 저장함. inputTopping에는 띄어쓰기로 구분된 토핑 번호들이 적혀있음. 따라서 vector<string> vectorTopping = SplitStringBySpace(inputTopping)을 통해 inputTopping의 띄어쓰기로 구분된 한 문장의 문자열들을 단어로 쪼개어 vectorTopping에 저장.

vectorTopping을 살펴보아, "6"이 존재하지 않을 시 "입력하신 숫자에 6이 없습니다. 토핑 선택을 리셋합니다. 다시 입력해주세요."가 출력됨. vectorTopping에서 "6" 전에 나온 문자열들로만 이루어진 set<string> setTopping을 생성. 이는 중복된 토핑 선택을 무시하기 위함.

setTopping을 살펴보아, 토핑 선택지에 없는 문자열이 존재할 때 "올바르지 않은 입력이 존재합니다. 입력하신 토핑 선택이 리셋됩니다. 다시 입력해주세요."가 출력됨.

입력 받은 setTopping과 pair<string, int>& toppingInfo = options[a]을 통해, 토핑 이름은 vector<string> toppings에 저장되고, 토핑 가격은 FinalOrder f 객체에 추가됨.

토핑 추가까지 완료되었으면 해당 주문을 vector<FinalOrder> basket에 push_back(f) 하여 주문 목록에 추가함.

(orser == "1"과 겹침)

· order에 "2"가 입력되면 currentMenu에 6을 할당하고, m[currentMenu]->Print()를 통해 디저트 카테고리 속 메뉴들이 출력됨. 이때 SelectMenu(m[currentMenu]->menu, menu, selectMenu) 함수를 사용해 menu에 값을 입력 받고, 해당 메뉴의 Cafe 객체를 selectMenu에 저장함. 이때 메뉴 선택지에서 벗어난 입력을 하면 올바른 입력이 아니라 출력되고, SelectMenu() 함수가 다시 호출되어 menu에 값을 다시 입력 받음.

If(!selectMenu.GetSubMenu().empty())를 통해 선택한 메뉴의 서브 메뉴가 존재하는지 확인.

서브 메뉴가 존재한다면 selectMenu.PrintSub()를 호출하여 해당 서브 메뉴를 출력함. SelectSubMenu(selectMenu, subMenu) 함수를 사용해 subMenu에 값을 입력 받음. 이때 서브 메뉴 선택지에서 벗어난 입력을 하면 올바른 입력이 아니라 출력되고, SelectSubMenu() 함수가 다시 호출되어 subMenu에 값을 다시 입력 받음.

menu, subMenu에 "0"을 입력할 시, back = true로 설정하여 가장 처음 while문으로 돌아가게 함.

서브 메뉴가 존재할 땐 FinalOrder f에 FinalOrder(menu, subMenu, selectMenu.GetPrice())를 저장하고, 서브 메뉴가 없을 때는 FinalOrder(menu, sub.GetPrice())를 f에 저장함. 해당 주문을 vector<FinalOrder> basket에 push_back(f) 하여 주문 목록에 추가함.

- 적용된 배운 내용

클래스(상속), 함수, 예외처리, 벡터, while/for 반복문, 조건문, map, set

- 코드 스크린샷

```
    }  
    // 2. 주문  
    else if (choice == "2") {  
        string order;  
        SelectOrder(order);  
    }
```

```

// 음료 & 디저트 주문
if (order == "3") { PrintReset(); continue; }
else {
    string category;
    if (order == "1") { // 음료
        SelectCategory(category);

        if (category == "7") { // 처음으로 돌아가기
            PrintReset();
            continue;
        }
        else {
            if (category == "1") currentMenu = 0;
            else if (category == "2") currentMenu = 1;
            else if (category == "3") currentMenu = 2;
            else if (category == "4") currentMenu = 3;
            else if (category == "5") currentMenu = 4;
            else if (category == "6") currentMenu = 5;
        }
    }
    else { // order == "2", 디저트
        currentMenu = 6;
    }
}

```

```

m[currentMenu]->Print();

// 메뉴 선택
string menu;
Cafe selectMenu = m[currentMenu]->menu[0];
SelectMenu(m[currentMenu]->menu, menu, selectMenu);

if (menu == "0") { // 처음으로 돌아가기
    PrintReset(); continue;
}

string subMenu; // 서브 메뉴
if (!selectMenu.GetSubMenu().empty()) {
    cout << endl;
    selectMenu.PrintSub();

    SelectSubMenu(selectMenu, subMenu); // 서브 메뉴 선택

    if (subMenu == "0") {
        PrintReset();
        continue; // 처음 화면으로 돌아가기
    }
}
}

```

```

FinalOrder f = FinalOrder();
if (order == "1") {
    string ice_hot;
    SelectIceHot(currentMenu, ice_hot); // ICE / HOT 선택

    if (ice_hot == "0") {
        PrintReset();
        continue; // 처음 화면으로 돌아가기
    }

    vector<string> toppings;
    if (!selectMenu.GetSubMenu().empty()) {
        f = FinalOrder(category, menu, subMenu, ice_hot, selectMenu.GetPrice());
    }
    else {
        f = FinalOrder(category, menu, ice_hot, selectMenu.GetPrice());
    }
    SelectToppings(toppings, f);
}
else { // order == 2
    if (!selectMenu.GetSubMenu().empty()) {
        f = FinalOrder(menu, subMenu, selectMenu.GetPrice());
    }
    else {
        f = FinalOrder(menu, selectMenu.GetPrice());
    }
}
basket.push_back(f);
cout << endl << " == ";
f.PrintFinal();
cout << "음(물) 구매 예정 목록에 담았습니다." << endl;
cin.clear();
}
}

```

```

// 주문 선택
void SelectOrder(string& order) {
    try {
        cout << endl << "무엇을 주문하시겠습니까? (숫자를 입력하십시오)" << endl;
        cout << "1. 음료 2. 디저트 3. 처음으로 돌아가기" << endl;
        cout << " >> ";
        getline(cin, order);
        order = NonSpace(order);
        vector<string> v2 = { "1", "2", "3" }; // 선택지를 저장한 벡터
        CheckInput(v2, order); // 선택 & 선택지에 맞지 않은 입력 시 error
    }
    catch (exception& e) { // 잘못된 입력으로 다시 선택
        cout << endl << e.what() << endl;
        SelectOrder(order);
    }
}

```

```

// 음료 카테고리 선택
void SelectCategory(string& category) {
    try {
        cout << endl << "(숫자를 입력하십시오)" << endl;
        cout << "1. 커피 2. 녹차 3. 티 4. 스무디 5. 프라페 6. 에이드 7. 처음으로 돌아가기" << endl;
        cout << " >> ";
        getline(cin, category);
        category = NonSpace(category);
        vector<string> v3 = { "1", "2", "3", "4", "5", "6", "7" }; // 선택지를 저장한 벡터
        CheckInput(v3, category); // 선택 & 선택지에 맞지 않은 입력 시 error
    }
    catch (exception& e) { // 잘못된 입력으로 다시 선택
        cout << endl << e.what() << endl;
        SelectCategory(category);
    }
}

```

```

// 메뉴 선택
void SelectMenu(vector<Cafe> c, string& menu, Cafe& selectMenu) {
    cout << "원하는 메뉴를 선택하십시오 (문자를 입력하십시오)" << endl;
    cout << "(0을 입력하면 처음으로 돌아갑니다)" << endl;
    cout << " >> ";
    getline(cin, menu);
    menu = NonSpace(menu);

    if (menu == "0") { return; }

    int count = 0;
    for (auto& e : c) {
        if (e.GetName() == menu) {
            selectMenu = e;
            count++;
        }
    }
    if (count == 0) {
        PrintError(); cout << endl;
        SelectMenu(c, menu, selectMenu);
    }
}

```

```

// 서브 메뉴(맛) 존재 시 선택
void SelectSubMenu(Cafe& c, string& subMenu) {
    cout << "해당 메뉴에서 원하는 맛을 선택하십시오 (문자를 입력하십시오)" << endl;
    cout << "(0을 입력하면 처음으로 돌아갑니다)" << endl;
    cout << " >> ";
    getline(cin, subMenu);
    subMenu = NonSpace(subMenu);

    if (subMenu == "0") { return; }

    int count = 0;
    for (string s : c.GetSubMenu()) {
        if (s == subMenu) count++;
    }
    if (count == 0) {
        PrintError(); cout << endl;
        SelectSubMenu(c, subMenu);
    }
}

```

```

// 음료의 Ice/Hot 선택
void SelectIceHot(int currentMenu, string& ice_hot) {
    if (currentMenu == 0 || currentMenu == 1 || currentMenu == 2) {
        cout << endl << "ICE / HOT 중 하나를 선택하십시오. (대문자/소문자 상관없이 영어로 입력하십시오)" << endl;
        cout << "(0을 입력하면 처음으로 돌아갑니다)" << endl;
        cout << ">> ";
        getline(cin, ice_hot);
        ice_hot = NonSpace(ice_hot);

        ice_hot = toUpperStr(ice_hot);
        if (ice_hot == "0") { return; }

        if (ice_hot != "ICE" && ice_hot != "HOT") {
            PrintError();
            SelectIceHot(currentMenu, ice_hot);
        }
    }
    else ice_hot = "ICE";
}

```

```

// 문자열의 모든 문자를 대문자로 바꾸는 함수
string toUpperStr(string str) {
    for (char& c : str)
        c = toupper(c);
    return str;
}

```

```

// 띄어쓰기가 포함된 문자열에서 띄어쓰기가 없는 문자들의 집합으로 만들
vector<string> SplitStringBySpace(string str) {
    vector<string> result;
    string temp;
    for (char ch : str) {
        if (ch == ' ') {
            if (!temp.empty()) {
                result.push_back(temp);
                temp.clear();
            }
        }
        else {
            temp += ch;
        }
    }
    if (!temp.empty()) {
        result.push_back(temp);
    }
    return result;
}

```

```

// 음료의 토핑 선택
void SelectToppings(vector<string>& toppings, FinalOrder& f) {
    map<string, pair<string, int>> options = {
        {"1", {"Size UP", 800}},
        {"2", {"샷 추가", 500}},
        {"3", {"펄 추가", 500}},
        {"4", {"휘핑크림 추가", 300}},
        {"5", {"바닐라시럽 추가", 300}}
    };
    set<string> setTopping;

    cout << endl << "*** 6을 입력하면 종료되고, 6 이후의 입력은 무시됩니다. 중복된 토핑 선택은 한 번만 적용됩니다.*** " << endl;
    while (1) {
        cout << endl << "원하는 추가 옵션을 모두 입력하십시오. (숫자를 입력하십시오)" << endl;
        cout << "1, Size UP(+800원) 2, 샷 추가(+500원) 3, 펄 추가(+500원) 4, 바닐라시럽 추가(+300원) 5, 휘핑크림 추가(+300원) 6, NO" << endl;
        cout << ">> ";

        bool check = true; // check == false가 될 경우, 토핑 선택 다시 실행

        string inputTopping;
        getline(cin, inputTopping);
        vector<string> vectorTopping = SplitStringBySpace(inputTopping);

        int count = 0;
        for (auto& a : vectorTopping) {
            if (a == "6") { count++; break; }
            else setTopping.insert(a);
        }
        if (count == 0) {
            cout << endl << "입력하신 숫자에 6이 없습니다. 토핑 선택을 리셋합니다. 다시 입력해주세요." << endl;
            check = false;
        }
        else {
            for (auto& a : setTopping) {
                if (options.find(a) == options.end()) {
                    cout << endl << "올바르지 않은 입력이 존재합니다. 입력하신 토핑 선택이 리셋됩니다. 다시 입력해주세요." << endl;
                    check = false; break;
                }
            }
        }
        if (check == false) {
            vectorTopping.clear(); setTopping.clear();
            continue;
        }
    }
}

```

```

    }
    else break;
}

for (auto& a : setTopping) {
    pair<string, int>& toppingInfo = options[a];
    string toppingName = toppingInfo.first;
    int price = toppingInfo.second;
    toppings.push_back(toppingName);
    f.IncreasePrice(price);
}
f.SetTopping(toppings);

if (!toppings.empty()) {
    cout << endl << "선택하신 옵션은 ";
    f.PrintTopping();
    cout << "입니다." << endl;
}
else {
    cout << endl << "토핑을 추가하지 않았습니다." << endl;
}
}

```

(3) 주문 목록 관리 (삭제 & 출력)

- 입력:

string choice(선택), vector<FinalOrder> basket(주문 목록이 저장됨), string orderList(주문 목록 삭제/출력/처음으로 돌아가기 선택), int del_Index(삭제할 주문 목록 인덱스)

- 출력:

- choice에 "3"을 입력할 시, "1. 주문 목록 삭제 2. 현재 주문 목록 3. 처음으로 돌아가기"가 출력됨.

- orderList에 "1"을 입력할 시, 현재 주문 목록을 출력함. 현재 주문 목록이 비어있으면 "현재 삭제할 수 있는 목록이 없습니다."고 출력.

현재 주문 목록이 비어있지 않다면, 주문 목록들을 중 삭제하고자 하는 목록을 물어봄.

del_Index에 0이 입력될 시, "주문 목록을 삭제하지 않고 처음으로 돌아갑니다."라 출력.

del_Index에 삭제하고자 하는 주문 목록의 인덱스가 입력될 시, 해당 주문 목록이 삭제되었다고 출력됨.

del_Index에 올바르지 않은 값이 입력될 시, "잘못된 입력으로 삭제를 실패하였습니다. 삭제하고자 하는 올바른 목록의 값을 다시 입력해주세요."가 출력.

- orderList에 "2"를 입력할 시, 현재 주문 목록이 비어있으면 "(없음)"이 출력. 현재 주문 목록이 비어있지 않다면, 입력된 전체 주문 목록이 출력됨.

- orderList 에 "3"을 입력하면, "처음으로 돌아갑니다."가 출력.

- 설명

· choice 에 "3"을 입력할 시 실행됨. SelectOrderList(orderList)를 통해 주문 목록을 삭제할 것인지, 주문 목록을 출력할 것인지, 처음으로 돌아갈 것인지를 선택함. 이때 CheckInput(v, orderList)를 이용해 orderList 에 올바르게 입력한지 확인. 만약 에러가 던져지면, SelectOrderList() 함수 내부의 catch 문이 실행되어 잘못 입력되었다고 출력되고, SelectOrderList()가 다시 호출되어 orderList 에 다시 값을 입력 받음.

· orderList 에 "1"을 입력하면, PrintOrderList(basket)을 통해 현재 주문 목록이 출력됨. 이때 주문 목록이 비어있으면 현재 삭제할 수 있는 목록이 없다고 출력됨.

현재 주문 목록이 비어있지 않다면, DeleteOrderList(basket) 함수가 호출됨. 현재 삭제하고자 하는 목록의 인덱스를 del_Index 에 입력함. del_Index 의 값이 0 일 때, 주문 목록을 삭제하지 않고 처음으로 돌아감. del_Index 의 값이 주문 목록에 없을 때, PrintDeleteError()을 호출하여 잘못된 입력으로 삭제를 실패했다고 출력하고 continue 를 통해 del_Index 에 다시 값을 입력 받음. del_Index 의 값이 주문 목록에 존재할 때, 해당 주문 목록을 삭제함.

· orderList 에 "2"를 입력하면, PrintOrderList(basket)이 호출되어 현재 주문 목록을 출력함.

· orderList 에 "3"을 입력하면, PrintReset() 함수를 호출해 처음으로 돌아간다고 출력하고, continue 를 통해 가장 바깥의 while 문의 첫 하위 문장으로 이동함.

- 적용된 배운 내용

클래스(상속), 함수, 예외처리, 벡터, while 반복문, 조건문

- 코드 스크린샷

```
// 주문 목록
else if (choice == "3") {
    string orderList;
    SelectOrderList(orderList);

    if (orderList == "1") { // 주문 목록 삭제
        PrintOrderList(basket);
        if (basket.empty()) {
            cout << endl << "현재 삭제할 수 있는 목록이 없습니다." << endl;
            continue;
        }
        else DeleteOrderList(basket);
    }
    else if (orderList == "2") { // 현재 주문 목록 출력
        PrintOrderList(basket);
    }
    else { PrintReset(); continue; } //orderList == "3"일 때 처음으로 돌아가기
}
```

```

// 주문 목록을 볼지 삭제할지 선택
void SelectOrderList(string& orderList) {
    cout << endl << "(숫자를 입력하십시오)" << endl;
    cout << "1. 주문 목록 삭제 2. 현재 주문 목록 3. 처음으로 돌아가기" << endl;
    cout << " >> ";
    getline(cin, orderList);
    orderList = NonSpace(orderList);

    vector<string> v = { "1", "2", "3" };
    try {
        CheckInput(v, orderList);
    }
    catch (exception& e) {
        cout << endl << e.what() << endl;
        SelectOrderList(orderList);
    }
}

```

```

// 주문 목록 출력
void PrintOrderList(vector<FinalOrder>& basket) {
    int n = 0;
    cout << endl << "--- 주문 목록 ---" << endl;
    if (basket.empty()) {
        cout << " >> (없음)" << endl;
    }
    else {
        for (auto& e : basket) {
            cout << " >> " << ++n << " , ";
            e.PrintFinal();
            cout << endl;
        }
    }
}

```

```

// 주문 목록 삭제 시, 목록을 잘못 입력하였을 때 출력
void PrintDeleteError() {
    cout << endl << "잘못된 입력으로 삭제를 실패하였습니다. 삭제하고자 하는 올바른 목록의 값을 다시 입력해주세요." << endl;
}

```

```

// 주문 목록 삭제
void DeleteOrderList(vector<FinalOrder>& basket) {
    string del_Index;
    while (1) {
        cout << endl << "몇 번째 목록을 삭제하시겠습니까?" << endl;
        cout << "(0을 입력하면 처음으로 돌아갑니다)" << endl;
        cout << " >> ";

        getline(cin, del_Index);
        stringstream ss(del_Index);
        int index;
        ss >> index;

        bool check = true;

        if (del_Index.empty()) { PrintDeleteError(); continue; }

        if (index == 0) {
            if (del_Index == "0") {
                cout << endl << "주문 목록을 삭제하지 않고 처음으로 돌아갑니다." << endl;
                return;
            }
            else { PrintDeleteError(); continue; }
        }

        if (index < 1 || index > basket.size()) { PrintDeleteError(); continue; }

        if (del_Index == to_string(index)) {
            cout << endl << " >> ";
            basket[index - 1].PrintFinal();
            cout << "을 삭제합니다." << endl;
            basket.erase(basket.begin() + index - 1);
            break;
        }
        else { PrintDeleteError(); continue; }
    }
}

```

(4) 결제 & 매장 내 섭취 여부 선택 & 매장 내 섭취 시 애플 좌석 선택

- 입력:

string choice(선택), vector<FinalOrder> basket(주문 목록), string pay_ok(결제 여부), string

take(매장 내 섭취 여부), string pay_method(결제 수단), string floor(앉을 층), string seat(앉을 좌석), Floor* currentFloor(선택한 층의 객체를 가리킴), set<string> list(txt 파일에 적힌 좌석 리스트), set<string> updateList(선택된 좌석을 제외한 좌석 리스트)

- 출력:

- choice에 "4"가 입력될 때, 현재 주문 목록이 비어있으면 "주문 목록이 비어있습니다."라고 출력.

- 현재 주문 목록이 비어있지 않으면 결제 금액을 출력하고 결제 여부(pay_ok)를 물어봄.

- pay_ok 에 "네"나 "아니요"가 아닌 다른 문자열을 입력하면, 잘못된 입력으로 다시 입력하라고 출력됨.

- pay_ok 에 "아니요"가 입력되면 "처음 화면으로 돌아갑니다."가 출력.

- pay_ok 에 "네"가 입력되면, 음식을 매장에서 섭취할 건지 아니면 포장할 건지 여부(take)를 물어봄.

- take 에 "1"이나 "2"가 아닌 다른 문자열을 입력하면, 잘못된 입력으로 다시 입력하라고 출력됨.

- take 에 "2"를 입력하면, 최종 금액에서 5% 할인된 금액이 출력되고, 여러 결제 수단(pay_method)들을 제시함.

pay_method 에 "1", "2", "3", "4", "5", "6"이 아닌 다른 문자가 입력되면, 잘못된 입력으로 다시 입력하라고 출력됨.

pay_method 에 "6"이 입력되면, 결제를 취소하고 처음으로 돌아간다고 출력됨.

pay_method 에 "1", "2", "3", "4", "5"가 입력되면, 최종 주문 목록과 입력된 문자에 맞는 결제 수단으로 결제가 완료되었다고 출력되고 감사 인사말(종료 인사)이 출력됨.

- take 에 "1"이 입력되면, 앉을 층(floor)을 선택하라고 출력됨.

floor 에 "0", "1", "2"가 아닌 다른 문자를 입력하면, 잘못된 입력으로 다시 입력해달라고 출력됨.

floor 에 "0"이 입력되면, 매장 내 섭취 여부(take)에 대한 출력으로 돌아감.

floor 에 "1"이나 "2"가 입력되면, 입력된 층의 좌석(여석)과 해당 좌석들 중 앉고 싶은 좌석(seat)을 입력하라고 출력됨.

seat 에 "0"을 입력하는 경우, 층(floor)을 다시 선택하라고 출력됨.

seat 에 입력한 문자가 선택한 층에 유효하지 않은 좌석일 경우, 이미 선택된 좌석 또는 잘못된 입력으로 다시 입력하라고 출력됨.

seat 에 입력한 문자가 선택한 층에 유효한 좌석일 경우, 선택한 좌석이 '사용중'이라는 표식으로 전환되고 해당 좌석이 선택되었다고 출력됨. 그 후 결제 수단(payment_method)을 선택하라고 출력됨.

payment_method 에 "1", "2", "3", "4", "5", "6"이 아닌 다른 문자가 입력되면, 잘못된 입력으로 다시 입력하라고 출력됨.

payment_method 에 "6"이 입력되면, 결제를 취소하고 처음으로 돌아간다고 출력됨.

payment_method 에 "1", "2", "3", "4", "5"가 입력되면, 최종 주문 목록과 입력된 문자에 맞는 결제 수단으로 결제가 완료되었다고 출력되고 감사 인사말(종료 인사)이 출력됨.

- 설명

- choice 에 "4"가 입력되면 실행됨. 이때 basket 에 주문 목록이 비어 있으면, continue 를 통해 결제를 하지 않고 처음 화면으로 돌아감.

- basket 에 주문 목록이 존재하면, PrintOrderList(basket)을 통해 현재 주문 목록을 출력하고, FinalPrice(basket)을 통해 주문 목록인 basket 에 담긴 FinalOrder 객체들의 가격의 합(결제 금액)을 출력함.

SelectYes_No(payment_ok) 함수를 통해 결제 여부를 선택함. payment_ok 에 "네"나 "아니요"가 아닌 다른 문자가 입력되면, 해당 입력은 잘못되었다고 출력되며 SelectYes_No() 함수를 다시 호출하여 payment_ok 에 다시 값을 입력함.

payment_ok 에 "아니요"가 입력되면 PrintReset() 함수를 호출해 처음 화면으로 돌아간다고 출력하고, continue 를 통해 가장 바깥의 while 문 내부의 첫 문장으로 돌아감.

payment_ok 에 "네"가 입력되면 SelectTake_out_In(take)를 통해 매장 내 섭취 여부를 선택함. take 에 "1"이나 "2"가 아닌 다른 문자가 입력되면, 해당 입력은 잘못되었다고 출력되며 SelectTake_out() 함수를 다시 호출하여 take 에 다시 값을 입력함.

- take 에 "2"가 입력되면 sumPrice(총 결제 금액) * 0.95 를 통해 최종 금액에서 5%를 할인해 줌. SelectPaymentMethod(payment_method, basket)를 통해 결제 수단을 선택함. 이때 CheckInput(v, payment_method) 함수를 통해, payment_method 에 입력한 값이 "1", "2", "3", "4", "5",

"6"이 아닌 다른 값일 때 오류를 던짐. 던져진 오류는 SelectPayMethod() 함수 내부의 catch 문으로 가 오류의 원인이 출력되고, SelectPayMethod()를 다시 호출하여 pay_method 에 다시 값을 입력 받음.

pay_method 의 값이 "6"일 때, back = true 로 설정해 처음 화면으로 돌아감.

pay_method 의 값이 "1", "2", "3", "4", "5" 중 하나일 때, 현재 주문 목록이 출력되고 선택한 결제 수단으로 결제가 완료되었다고 출력됨. 이후에 존재하는 break 2 개로 인해 while 문을 완전히 다 빠져나감. (종료)

- take 에 "1"이 입력되면 SelectFloor(floor)를 통해 앉고자 하는 층을 선택하고, currentFloor->PrintFloor()을 통해 선택한 층의 좌석을 출력함. SelectSeat(floor, seat, currentFloor)를 통해 앉을 좌석을 선택함. 이들의 작동 방식 또한 위의 함수들과 같이 잘못된 입력이 들어가면 자신을 재귀 호출하여 올바른 형식의 값이 floor 와 seat 에 들어가도록 함. 이때 floor 에 "0"이 입력되면 매장 내 섭취 여부(take) 선택으로 돌아가고, seat 에 "0"이 입력되면 앉고자 하는 층 선택으로 돌아감.

seat 에 올바른 좌석이 입력되면 currentFloor->SelectSeats(seat)를 통해 선택한 좌석의 표식을 "□"에서 "■"로 변경함. AddSeat(floor, seat) 함수를 호출해 해당 좌석을 txt 파일에 추가함. currentFloor->PrintFloor()를 통해 선택한 좌석의 표식과 이름이 변경되었다는 것을 보여줌.

SelectPayMethod(pay_method, basket)을 통해 결제 수단을 선택함. 이때 CheckInput(v, pay_method) 함수를 통해, pay_method 에 입력한 값이 "1", "2", "3", "4", "5", "6"이 아닌 다른 값일 때 오류를 던짐. 던져진 오류는 SelectPayMethod() 함수 내부의 catch 문으로 가 오류의 원인이 출력되고, SelectPayMethod()를 다시 호출하여 pay_method 에 다시 값을 입력 받음.

pay_method 의 값이 "1", "2", "3", "4", "5" 중 하나일 때, 현재 주문 목록이 출력되고 선택한 결제 수단으로 결제가 완료되었다고 출력됨. 이후에 존재하는 break 2 개로 인해 while 문을 완전히 다 빠져나감. (종료)

pay_method 의 값이 "6"일 때, CheckSeats(floor, seat, list)를 통해 txt 파일에서 좌석 정보를 읽어와서 list 에 저장함. for 문과 if 문을 이용하여 list 에 적힌 좌석들 중 위에서 선택한 좌석의 이름을 제외한 다른 좌석들의 이름만 updateList 에 저장함. currentFloor->ResetSeats(seat)를 통해 전에 선택한 좌석을 다시 "■"에서 "□"로 변경함.

UpdateExitSeat(floor, updateList, f1, f2)를 통해 선택했던 좌석이 제거된 리스트를 다시 txt 파일에 덮어씀. 그 후 back 에 true 값을 줘서 처음으로 돌아가게 함.

· take_select == true 가 되면(floor 에 "0" 입력) take out 여부 선택으로 돌아감. back == true 가 되면(pay_method 에 "6" 입력) 처음 화면으로 돌아감. 이들이 모두 설정되지 않았으면(결제 진행 시) 두 번의 break 문을 만나 종료됨.

- 적용된 배운 내용

클래스(상속), 함수, 예외처리, 벡터, while 반복문, 조건문, set

- 코드 스크린샷

```
// 결제
else if (choice == "4") {
    if (basket.empty()) { // 주문 목록이 비어있을 때, 결제 취소 & 처음으로 돌아감
        cout << endl;
        cout << "주문 목록이 비어있습니다." << endl;
        continue;
    }
    else { // 주문 목록이 비어있지 않을 때

        PrintOrderList(basket); // 현재 주문 목록을 출력

        cout << endl;
        cout << "현재 결제 금액은 총 " << FinalPrice(basket) << "원 입니다." << endl;

        string pay_ok; // 결제 여부
        SelectYes_No(pay_ok); // 결제 여부 선택
        if (pay_ok == "아니요") { PrintReset(); continue; } // 결제 취소 시 처음으로 돌아감
        else { // 결제 진행
            while (1) {
                bool take_select = false; // take_select == true가 되면 take out 여부 다시 선택
                string take; // take out 할 건지 or take in 할 건지
                string pay_method; // 지불 수단
                SelectTake_out_In(take); // take out 여부 결정

                double sumPrice = FinalPrice(basket);
                if (take == "2") { // take out 할 때
                    sumPrice *= 0.95; // 최종 금액의 5% 할인
                    cout << endl << "Take Out 할인으로 현재 결제 예정 금액은 " << sumPrice << "원 입니다." << endl;
                    SelectPayMethod(pay_method, basket); // 결제 수단 선택

                    if (pay_method == "6") back = true; // 결제 취소시 처음으로 돌아감
                    break;
                }
            }
        }
    }
}

else { // take in 할 때
    string floor; // 선택한 층을 저장할 변수
    string seat; // 선택한 좌석을 저장할 변수
    Floor+ currentFloor = &f1;
    while (1) {
        SelectFloor(floor); // 층 선택
        if (floor == "0") { // take out 여부를 다시 선택
            take_select = true;
            break;
        }
        else if (floor == "1") currentFloor = &f1;
        else currentFloor = &f2;

        currentFloor->PrintFloor(); // 선택한 층의 좌석 출력

        SelectSeat(floor, seat, currentFloor); // 앉을 좌석 선택
        if (seat == "0") continue; // 층 선택으로 돌아감
        else {
            currentFloor->SelectSeats(seat); // 선택한 좌석 변경 "□"-"■"
            AddSeat(floor, seat); // 선택한 좌석 txt 파일에 추가

            currentFloor->PrintFloor(); // 선택된 좌석이 변경된 (해당 층의) 전체 좌석 출력
            cout << endl << "~~~" << floor << "층의 " << seat << " 좌석이 선택되었습니다." << endl;

            SelectPayMethod(pay_method, basket); // 결제 수단 선택
        }
    }
}
```



```

// 매장 내 선택시, 앉을 층 선택
void SelectFloor(string& floor) {
    cout << endl << "*** 2층은 공부를 위한 장소로 다른 손님들에게 피해가 가지 않도록 조용히 해주세요 ***" << endl;
    cout << endl << "1층, 2층 중 원하는 층을 선택하십시오. (숫자를 입력하십시오)" << endl;
    cout << "(Take Out을 원하는 경우, 0을 입력하십시오, 다시 뒤로 돌아갑니다.)" << endl;
    cout << " >> ";
    getline(cin, floor);
    floor = NonSpace(floor);

    if (floor != "1" && floor != "2" && floor != "0") {
        cout << endl << "잘못된 입력입니다. 양식에 맞게 다시 입력해주세요." << endl;
        SelectFloor(floor);
    }
}

```

```

// 매장 내 선택시, 앉을 좌석 선택
void SelectSeat(string floor, string& seat, Floor* currentFloor) {
    cout << endl << floor << "층에서 원하는 좌석을 입력해주세요. (대문자/소문자 상관없음)" << endl;
    cout << "(층을 바꾸고자 하는 경우 0을 입력하십시오.)" << endl;
    cout << " >> ";
    getline(cin, seat);
    seat = NonSpace(seat);
    seat = toUpperStr(seat);

    bool check;
    bool checkSelected;
    if (seat == "0") return;

    check = currentFloor->CheckSeats(seat); // 선택된 좌석이 유효한지 확인

    if (check == true) return;
    else {
        checkSelected = currentFloor->CheckSelected(seat); // 선택한 좌석이 다른 고객이 사용중인 좌석인지 확인
        if (checkSelected == true)
            cout << endl << "이미 선택된 좌석입니다. 다른 좌석을 선택해주세요." << endl;
        else
            cout << endl << "올바르지 않는 좌석 입력입니다. 다시 입력해주세요." << endl;
        SelectSeat(floor, seat, currentFloor);
    }
}

```

```

// 매장 내 선택시, 앉을 좌석 txt파일에 추가
void AddSeat(string floor, string seat) {
    if (floor == "1") {
        ofstream os_1c{ "seats_floor_1.txt", ios::out | ios::app };
        if (os_1c.is_open()) {
            os_1c << seat << endl;
            os_1c.close();
        }
    }
    else {
        ofstream os_2c{ "seats_floor_2.txt", ios::out | ios::app };
        if (os_2c.is_open()) {
            os_2c << seat << endl;
            os_2c.close();
        }
    }
}

```

```

// 입력한 좌석이 존재하는지 확인
bool CheckSeats(string floor, string seat, set<string>& s) {
    bool check = false;
    if (floor == "1") {
        ifstream is_1{ "seats_floor_1.txt" };
        string s_1;
        while (is_1 >> s_1) {
            s.insert(s_1);
        }
        is_1.close();
    }
    else {
        ifstream is_2{ "seats_floor_2.txt" };
        string s_2;
        while (is_2 >> s_2) {
            s.insert(s_2);
        }
        is_2.close();
    }
    for (string s : s) {
        if (s == seat) {
            check = true;
            break;
        }
    }

    return check;
}

```

```

// 매장에서 퇴실할 시, 퇴석할 좌석 txt파일에서 제거
void UpdateExitSeat(string floor, set<string>& selectedSeats, Floor1& f1, Floor2& f2) {
    if (floor == "1") {
        ofstream os_1B{ "seats_floor_1.txt", ios::trunc };
        for (string s : selectedSeats) {
            os_1B << s << endl;
        }
        os_1B.close();
    }
    else {
        ofstream os_2B{ "seats_floor_2.txt", ios::trunc };
        for (string s : selectedSeats) {
            os_2B << s << endl;
        }
        os_2B.close();
    }
    UpdateSeat(f1, f2);
}

```

(5) 카페 내 좌석(여석) 출력

- 입력: Floor f1(1층 좌석), Floor f2(2층 좌석), string choice(선택)

- 출력:

· choice에 "5"를 입력할 시, 현재 카페 내 좌석들과 그들의 사용 가능 여부가 출력됨.

- 설명

· 가장 바깥의 while문에서 처음 실행되어지는 UpdateSeat(f1, f2)를 통해 좌석 txt 파일을 읽어와 현재 고객들이 사용 중인 좌석을 업데이트 함.

· choice에 "5"가 입력되면, f1.PrintFloor()와 f2.PrintFloor()가 호출되어 위에서 업데이트한 좌석들을 출력함.

- 적용된 배운 내용

클래스(상속), 함수, while/for 반복문, 조건문

- 코드 스크린샷

```

// 현재 카페 내 좌석(여석) 출력
else if (choice == "5") {
    f1.PrintFloor();
    cout << endl;
    f2.PrintFloor();
}

void Floor1::PrintFloor() {
    cout << endl << " <!출> " << endl;
    cout << "-----" << endl;
    for (char i = '1'; i <= '5'; i++) {
        string n = ""; n += i;
        if (n == "2" || n == "3" || n == "4")
            PrintSeats(n, 1);
        else
            PrintSeats(n);
        PrintNaming(n);
        PrintSeats(n);
    }
}

```

-> Floor1 클래스의 PrintFloor()함수

```

void Floor2::PrintFloor() {
    cout << endl << " <2층> -Study Only- " << endl;
    cout << "-----" << endl;
    for (char i = '1'; i <= '7'; i++) {
        string n = ""; n += i;
        cout << endl;
        PrintNaming(n);
        PrintSeats(n);
    }
}

```

-> Floor2 클래스의 PrintFloor()함수

(6) 퇴실

- 입력:

Floor f3(1층 좌석), Floor f3(2층 좌석), string choice(선택), string exit_floor(퇴실할 층), string exit_seat(퇴석할 좌석), set<string> seatList(사용중인 좌석 리스트), set <stirring> selectedSeats(사용중인 좌석들에서 퇴석할 좌석을 제외한 리스트)

- 출력:

- choice에 "6"을 입력할 시, 앉았던 층수(exit_floor)를 입력해주라고 출력됨.
- exit_floor에 "0"이나 "1", "2"를 제외한 다른 입력이 들어올 시, 잘못된 입력으로 다시 입력해 주라고 출력됨.
- exit_floor에 "0"을 입력할 시, 퇴실을 취소하고 처음 화면으로 돌아간다고 출력됨.
- exit_floor에 "1"이나 "2"를 입력할 경우, 입력한 층에 대한 좌석이 출력되는데 이때 고객들이 사용중인 좌석들의 이름도 보이게 출력됨. 출력된 좌석 중 앉았던 좌석(exit_seat)을 입력해 달라고 출력됨.
- exit_seat에 사용중인 좌석이 아닌 다른 입력이 들어올 시, 해당 좌석에 대한 정보가 없어 다시 입력해 주라고 출력됨.
- exit_seat에 "0"을 입력할 시, 앉았던 층 선택(exit_floor)으로 돌아간다고 출력됨.
- exit_seat에 앉았던 좌석에 대한 올바른 정보를 입력할 시, 선택한 층의 선택한 좌석에서 퇴석하겠다고 출력되고 프로그램이 종료됨.

- 설명

- choice에 "6"이 입력되면 실행됨. 일단 SelectedSeatUpdate(f3, f4)을 통해 고객들이 사용중인 좌석들의 이름이 보이도록 f3과 f4에 업데이트 함.

- SelectExitFloor(exit_floor)을 통해 퇴실할 층을 exit_floor에 입력 받음. exit_floor에 "0", "1", "2"가 아닌 다른 문자가 입력될 시, 잘못된 입력으로 다시 입력하라는 코멘트를 출력하고 SelectExitFloor()이 다시 호출되어 exit_floor에 값을 입력 받음.
- exit_floor에 "0"이 입력된 경우, 퇴실을 취소한다고 출력되고 PrintReset()을 호출해 처음 화면으로 돌아간다고 출력하고 back에 true값을 부여함. 문장의 후반에 back이 true일 때 continue를 통해 가장 바깥의 while문으로 돌아가게 함.
- exit_floor에 "1"이나 "2"가 입력된 경우, current_exitFloor에 해당 층의 객체 포인터를 할당함. current_exitFloor->PrintFloor()을 통해 입력된 층의 좌석들을 출력함.
- SelectExitSeats(exit_floor, exit_seat, seatList)을 통해 퇴석할 좌석을 exit_seat에 입력 받고, SelectExitSeats() 함수 내부의 CheckSeats() 함수 호출을 통해 seatList에 현재 고객들이 사용중인 좌석들을 저장함. exit_seat에 사용중인 좌석이 아닌 다른 입력이 들어올 시, 잘못된 입력으로 다시 입력하라는 코멘트가 출력되고 SelectExitSeats()이 다시 호출되어 exit_seat에 값을 입력 받음.
- exit_seat에 "0"이 입력되면, continue를 통해 퇴실할 층을 선택하는 단계로 돌아감.
- exit_seat에 퇴석할 좌석에 대한 올바른 정보가 입력되면, for문과 if문을 이용해 seatList에 적힌 적힌 좌석들 중 퇴석할 좌석의 이름을 제외한 다른 좌석들의 이름만 selectedSeats에 저장됨. current_exitFloor->ResetSeats(exit_seat)를 통해 퇴석할 좌석을 "■"에서 "□"로 변경함. UpdateExitSeat(exit_floor, selectedSeats, f3, f4)를 통해 퇴석할 좌석이 제거된 리스트(selectedSeats)를 다시 txt 파일에 덮어씀. PrintExit(exit_floor, exit_seat)를 통해 퇴석할 층과 좌석, 감사의 인사말이 출력됨. 그 후 두 번의 break문을 만나 프로그램이 종료됨.

- 적용된 배운 내용

클래스(상속), 함수, 예외처리, while/for 반복문, 조건문, set

- 코드 스크린샷

```

// 퇴실
else if (choice == "6") {
    Floor1 f3; Floor2 f4;
    set<string> seatList; // 사용중인 좌석들 (예정)
    set<string> selectedSeats; // 사용중인 좌석들에서 퇴석할 좌석을 제외한 리스트 (예정)

    Floor* current_exitFloor = &f3;

    SelectedSeatUpdate(f3, f4); // 선택된 좌석의 이름이 보이도록 업데이트
    while (1) {
        string exit_floor; // 퇴석할 좌석의 층
        SelectExitFloor(exit_floor); // 퇴석할 좌석의 층 선택

        if (exit_floor == "0") { // 퇴실 취소, 처음으로 돌아감
            cout << endl << "퇴실을 취소합니다.";
            PrintReset(); back = true;
            break;
        }
        else if (exit_floor == "1") current_exitFloor = &f3;
        else current_exitFloor = &f4;

        current_exitFloor->PrintFloor();

        string exit_seat; // 퇴석할 좌석
        SelectExitSeats(exit_floor, exit_seat, seatList); // 퇴석할 좌석 선택, 현재 사용중인 좌석들 리스트 seatList에 저장

        if (exit_seat == "0") continue; // 퇴석할 좌석의 층 선택으로 돌아감
        else {
            for (string s : seatList) { // 퇴석할 좌석을 제외한 사용중인 좌석들 리스트 selectedSeats에 저장
                if (s == exit_seat)
                    selectedSeats.insert(s);
            }
            current_exitFloor->ResetSeats(exit_seat); // 퇴석할 좌석을 빈 좌석으로 업데이트 "■"->"□"

            UpdateExitSeat(exit_floor, selectedSeats, f3, f4); // 퇴석할 좌석이 제거된 리스트를 다시 txt 파일에 저장 (덮어쓰기)
            PrintExit(exit_floor, exit_seat); // 퇴석할 층, 좌석, 감사의 인사말 출력
            break;
        }
        if (back == true) continue; // 처음으로 돌아감
        else break; // 종료
    }
}

```

```

// 선택된 좌석 이름이 보이게 불러오기
void SelectedSeatUpdate(Floor1& f1, Floor2& f2) {
    ifstream is_1{"seats_floor_1.txt"};
    string s_1;
    while (is_1 >> s_1) {
        f1.ExitSeats(s_1);
    }
    is_1.close();

    ifstream is_2{"seats_floor_2.txt"};
    string s_2;
    while (is_2 >> s_2) {
        f2.ExitSeats(s_2);
    }
    is_2.close();
}

```

```

// 매장에서 퇴실할 시, 퇴석할 층 선택
void SelectExitFloor(string& floor) {
    cout << endl << "음료/디저트를 잘 즐기셨습니까? 없었던 층수를 입력해 주세요. (숫자만 입력하십시오)" << endl;
    cout << "(취소하고자 하는 경우 0을 입력하십시오.)" << endl;
    cout << " >> ";
    getline(cin, floor);
    floor = NonSpace(floor);

    if (floor == "0") return;
    if (floor != "1" && floor != "2") {
        cout << endl << "잘못된 입력입니다. 다시 입력해 주세요." << endl;
        SelectExitFloor(floor);
    }
}

```

```

// 매장에서 퇴실할 시, 퇴석할 좌석 선택
void SelectExitSeats(string floor, string& seat, set<string>& seatList) {
    cout << endl << "없었던 좌석을 입력해 주세요. (대문자/소문자 상관없음)" << endl;
    cout << "(층을 바꾸고자 하는 경우 0을 입력하십시오.)" << endl;
    cout << " >> ";
    getline(cin, seat);
    seat = NonSpace(seat);
    seat = toUpperStr(seat);

    bool check;
    if (seat == "0") return;

    check = CheckSeats(floor, seat, seatList);

    if (check == false) {
        cout << endl << seat << " 좌석에 대한 정보가 없습니다. 다시 입력해 주세요." << endl;
        SelectExitSeats(floor, seat, seatList);
    }
}

```

```

// 입력한 좌석이 존재하는지 확인
bool CheckSeats(string floor, string seat, set<string>& s) {
    bool check = false;
    if (floor == "1") {
        ifstream is_1{ "seats_floor_1.txt" };
        string s_1;
        while (is_1 >> s_1) {
            s.insert(s_1);
        }
        is_1.close();
    }
    else {
        ifstream is_2{ "seats_floor_2.txt" };
        string s_2;
        while (is_2 >> s_2) {
            s.insert(s_2);
        }
        is_2.close();
    }
    for (string s : s) {
        if (s == seat) {
            check = true;
            break;
        }
    }
    return check;
}

// 퇴실시 출력
void PrintExit(string floor, string seat) {
    cout << endl << floor << "층의 " << seat << " 좌석에서 퇴석하겠습니다." << endl;
    cout << "저희 카드를 찾아와 주셔서 감사합니다. 안녕히 가세요." << endl;
}

// 매장에서 퇴실할 시, 퇴석할 좌석 txt파일에서 제거
void UpdateExitSeat(string floor, set<string>& selectedSeats, Floor1& f1, Floor2& f2) {
    if (floor == "1") {
        ofstream os_1B{ "seats_floor_1.txt", ios::trunc };
        for (string s : selectedSeats) {
            os_1B << s << endl;
        }
        os_1B.close();
    }
    else {
        ofstream os_2B{ "seats_floor_2.txt", ios::trunc };
        for (string s : selectedSeats) {
            os_2B << s << endl;
        }
        os_2B.close();
    }
    UpdateSeat(f1, f2);
}

```

(7) 종료

- 입력: string choice(선택)

- 출력: choice에 "7"을 입력할 시, "종료합니다. 다음에 또 찾아와 주세요." 출력.

- 설명

· choice에 "7"이 입력되면 "종료합니다. 다음에 또 찾아와 주세요."가 출력되고, break문을 만나 while문을 빠져나가게 되고 프로그램이 종료됨.

- 적용된 배운 내용

while 반복문, 조건문

- 코드 스크린샷

```

    }
    // choice == "7" 종료
    else {
        cout << endl << "종료합니다, 다음에 또 찾아 주세요. :D" << endl;
        break;
    }
}
return 0;
}

```

4. 테스트 결과

(1) 카페 메뉴와 해당 가격 제시

- 설명

· "1. 메뉴 & 가격 2. 주문 3. 주문 목록 4. 결제 5. 카페 내 여석 6. 퇴실 7. 종료"가 출력되는 가장 처음 화면임. 이때 "1", "2", "3", "4", "5", "6", "7" 외의 다른 문자를 입력할 시 "잘못된 입력입니다. 양식에 맞게 다시 입력하세요"가 출력됨. 이것 외에도 다른 선택지에서도 제시한 선택 문항 외의 다른 문자를 입력할 시 잘못된 입력으로 다시 입력해주는 문구가 출력됨.

· 처음 화면에서 1을 입력 시, 카페의 전체 메뉴와 그 가격을 출력함.

- 테스트 결과 스크린샷

```

(숫자를 입력하시오)
1. 메뉴 & 가격 2. 주문 3. 주문 목록 4. 결제 5. 카페 내 여석 6. 퇴실 7. 종료
>> jvh

```

잘못된 입력입니다. 양식에 맞게 다시 입력해주세요.

```

(숫자를 입력하시오)
1. 메뉴 & 가격 2. 주문 3. 주문 목록 4. 결제 5. 카페 내 여석 6. 퇴실 7. 종료
>> 1

```

<< COFFEE >>

에스프레소	2000원
아메리카노	2500원
카푸치노	3000원
카페라떼	3000원
헤이즐넛라떼	3500원
바닐라라떼	3500원
카라멜마끼아또	3500원
카페모카	3500원
돌체라떼	3700원

<< NON_COFFEE >>

곡물라떼	3300원
고구마라떼	3500원
초코라떼	3500원
그린티라떼	3500원

(2) 주문을 받고 주문 목록에 저장

- 설명

· 처음 화면에서 2를 입력하면 주문을 위한 프로그램이 시작됨. 음료와 디저트 중에 선택함. 그러면 음료는 카테고리, 메뉴, 서브메뉴(존재 시), 온도, 토핑을 선택할 수 있고, 디저트는 메뉴, 서브메뉴(존재 시)를 선택할 수 있음. 이들의 올바른 선택이 완료되면 주문 목록에 선택한 것이 업로드됨.

- 테스트 결과 스크린샷

```
(숫자를 입력하시오)
1. 메뉴 & 가격 2. 주문 3. 주문 목록 4. 결제 5. 카페 내 여석 6. 퇴실 7. 종료
>> 2

무엇을 주문하시겠습니까? (숫자를 입력하시오)
1. 음료 2. 디저트 3. 처음으로 돌아가기
>> 3

처음 화면으로 돌아갑니다.

(숫자를 입력하시오)
1. 메뉴 & 가격 2. 주문 3. 주문 목록 4. 결제 5. 카페 내 여석 6. 퇴실 7. 종료
>>
```

```
(숫자를 입력하시오)
1. 메뉴 & 가격 2. 주문 3. 주문 목록 4. 결제 5. 카페 내 여석 6. 퇴실 7. 종료
>> 2

무엇을 주문하시겠습니까? (숫자를 입력하시오)
1. 음료 2. 디저트 3. 처음으로 돌아가기
>> 1

(숫자를 입력하시오)
1. 커피 2. 논커피 3. 티 4. 스무디 5. 프라페 6. 에이드 7. 처음으로 돌아가기
>> 1

<< COFFEE >>
-----
    에스프레소    2500원
    아메리카노    2500원
    카푸치노      3000원
    카페라떼      3000원
    헤이즐넛라떼   3500원
    바닐라라떼     3500원
    카라멜마끼아또 3500원
    카페모카      3500원
    돌체라떼      3700원
-----

원하는 메뉴를 선택하시오 (문자를 입력하시오. 띄어쓰기 없이)
(0을 입력하면 처음으로 돌아갑니다)
>> 0

처음 화면으로 돌아갑니다.

(숫자를 입력하시오)
1. 메뉴 & 가격 2. 주문 3. 주문 목록 4. 결제 5. 카페 내 여석 6. 퇴실 7. 종료
>>
```

무엇을 주문하시겠습니까? (숫자를 입력하십시오)

1. 음료 2. 디저트 3. 처음으로 돌아가기

>> 1

(숫자를 입력하십시오)

1. 커피 2. 논커피 3. 티 4. 스무디 5. 프라페 6. 에이드 7. 처음으로 돌아가기

>> 2

<< NON_COFFEE >>

곡물라떼	3300원
고구마라떼	3500원
딸기라떼	3500원
초코라떼	3500원
그린티라떼	3500원
민트초코라떼	3500원

원하는 메뉴를 선택하십시오 (문자를 입력하십시오. 띄어쓰기 없이)

(0을 입력하면 처음으로 돌아갑니다)

>> 초코라떼

ICE / HOT 중 하나를 선택하십시오. (대문자로 입력하십시오)

(0을 입력하면 처음으로 돌아갑니다)

>> ICE

원하는 추가 옵션을 모두 입력하십시오. (숫자를 입력하십시오. 6을 입력하면 종료됩니다.)

1. Size UP(+800원) 2. 샷 추가(+500원) 3. 휘핑크림 추가(+300원) 4. 바닐라시럽 추가(+300원) 5. 펄 추가(+500원) 6. NO

>> 1 2 3 6

선택하신 옵션은 'Size UP, 샷 추가, 휘핑크림 추가'입니다.

** 논커피 - 초코라떼(ICE) - 5100원 (Size UP/샷 추가/휘핑크림 추가)을 구매 예정 목록에 담았습니다.

(숫자를 입력하십시오)

1. 메뉴 & 가격 2. 주문 3. 주문 목록 4. 결제 5. 카페 내 여석 6. 퇴실 7. 종료

>> 2

무엇을 주문하시겠습니까? (숫자를 입력하십시오)

1. 음료 2. 디저트 3. 처음으로 돌아가기

>> 2

<< DESSERT >>

쿠키	1500원	
마카롱	2000원	(초코/바닐라/쿠앤크/녹차/딸기/카라멜)
스콘	2500원	
머핀	2500원	
와플	3000원	
크로플	3300원	
조각케이크	4700원	(생크림/초코/치즈/고구마/티라미슈)

원하는 메뉴를 선택하십시오 (문자를 입력하십시오. 띄어쓰기 없이)

(0을 입력하면 처음으로 돌아갑니다)

>> 마카롱

-> 초코 / 바닐라 / 쿠앤크 / 녹차 / 딸기 / 카라멜

해당 메뉴에서 원하는 맛을 선택하십시오 (문자를 입력하십시오)

(0을 입력하면 처음으로 돌아갑니다)

>> 바닐라

** 디저트 - 마카롱 - 바닐라 - 2000원을 구매 예정 목록에 담았습니다.

(숫자를 입력하십시오)

1. 커피 2. 논커피 3. 티 4. 스무디 5. 프라페 6. 에이드 7. 처음으로 돌아가기
>> 1

<< COFFEE >>

에스프레소	2500원
아메리카노	2500원
카푸치노	3000원
카페라떼	3000원
헤이즐넛라떼	3500원
바닐라라떼	3500원
카라멜마끼아또	3500원
카페모카	3500원
돌체라떼	3700원

원하는 메뉴를 선택하십시오 (문자를 입력하십시오. 띄어쓰기 없이)

(0을 입력하면 처음으로 돌아갑니다)

>> americano

잘못된 입력입니다. 양식에 맞게 다시 입력해주세요.

원하는 메뉴를 선택하십시오 (문자를 입력하십시오. 띄어쓰기 없이)

(0을 입력하면 처음으로 돌아갑니다)

>> 아메리카노

ICE / HOT 중 하나를 선택하십시오. (대문자로 입력하십시오)

(0을 입력하면 처음으로 돌아갑니다)

>> 0

처음 화면으로 돌아갑니다.

(숫자를 입력하십시오)

1. 메뉴 & 가격 2. 주문 3. 주문 목록 4. 결제 5. 카페 내 여석 6. 퇴실 7. 종료

무엇을 주문하시겠습니까? (숫자를 입력하십시오)

1. 음료 2. 디저트 3. 처음으로 돌아가기

>> 1

(숫자를 입력하십시오)

1. 커피 2. 논커피 3. 티 4. 스무디 5. 프라페 6. 에이드 7. 처음으로 돌아가기

>> 1

<< COFFEE >>

에스프레소	2500원
아메리카노	2500원
카푸치노	3000원
카페라떼	3000원
헤이즐넛라떼	3500원
바닐라라떼	3500원
카라멜마끼아또	3500원
카페모카	3500원
돌체라떼	3700원

원하는 메뉴를 선택하십시오 (문자를 입력하십시오. 띄어쓰기 없이)

(0을 입력하면 처음으로 돌아갑니다)

>> 아메리카노

ICE / HOT 중 하나를 선택하십시오. (대문자로 입력하십시오)

(0을 입력하면 처음으로 돌아갑니다)

>> ICE

원하는 추가 옵션을 모두 입력하십시오. (숫자를 입력하십시오. 6을 입력하면 종료됩니다.)

1. Size UP(+800원) 2. 샷 추가(+500원) 3. 휘핑크림 추가(+300원) 4. 바닐라시럽 추가(+300원) 5. 펄 추가(+500원) 6. NO

>> 2 4 6

선택하신 옵션은 '샷 추가, 바닐라시럽 추가'입니다.

** 커피 - 아메리카노(ICE) - 3300원 (샷 추가/바닐라시럽 추가)을 구매 예정 목록에 담았습니다.

(3) 주문 목록 관리 (삭제 & 출력)

- 설명

· 처음 화면에서 3을 입력하면 주문 목록에 관해 관리할 수 있음. 여기에서 1을 입력하면 주문 목록을 삭제할 수 있고, 2를 입력하면 현재 주문 목록을 출력할 수 있고, 3을 입력하면 다시 처음 화면(선택)으로 돌아감.

· 주문 목록 삭제를 원할 시, 주문 목록이 존재해야 함. 주문 목록이 비어 있으면 1을 입력(주문 목록 삭제를 선택)해도 다시 처음 화면으로 돌아감.

- 테스트 결과 스크린샷

```
(숫자를 입력하십시오)
1. 메뉴 & 가격 2. 주문 3. 주문 목록 4. 결제 5. 카페 내 여석 6. 퇴실 7. 종료
>> 3

(숫자를 입력하십시오)
1. 주문 목록 삭제 2. 현재 주문 목록 3. 처음으로 돌아가기
>> 1

현재 주문 목록
>> (없음)

현재 삭제할 수 있는 목록이 없습니다.

(숫자를 입력하십시오)
1. 메뉴 & 가격 2. 주문 3. 주문 목록 4. 결제 5. 카페 내 여석 6. 퇴실 7. 종료
>>
```

```
(숫자를 입력하십시오)
1. 메뉴 & 가격 2. 주문 3. 주문 목록 4. 결제 5. 카페 내 여석 6. 퇴실 7. 종료
>> 3

(숫자를 입력하십시오)
1. 주문 목록 삭제 2. 현재 주문 목록 3. 처음으로 돌아가기
>> 3

처음 화면으로 돌아갑니다.

(숫자를 입력하십시오)
1. 메뉴 & 가격 2. 주문 3. 주문 목록 4. 결제 5. 카페 내 여석 6. 퇴실 7. 종료
>>
```

```
(숫자를 입력하십시오)
1. 메뉴 & 가격 2. 주문 3. 주문 목록 4. 결제 5. 카페 내 여석 6. 퇴실 7. 종료
>> 3

(숫자를 입력하십시오)
1. 주문 목록 삭제 2. 현재 주문 목록 3. 처음으로 돌아가기
>> 2

현재 주문 목록
>> 1. 돈키호테 - 초코라떼(ICE) - 5100원 (Size UP/샷 추가/휘핑크림 추가)
>> 2. 디저트 - 마카롱 - 바닐라 - 2000원
>> 3. 티 - 홍차 - 얼그레이(HOT) - 2500원
>> 4. 디저트 - 와플 - 3000원
>> 5. 커피 - 아메리카노(ICE) - 3300원 (샷 추가/바닐라시럽 추가)
```



```

(숫자를 입력하시오)
1. 주문 목록 삭제 2. 현재 주문 목록 3. 처음으로 돌아가기
>> 1

현재 주문 목록
>> 1. 논커피 - 초코라떼(ICE) - 5100원 (Size UP/샷 추가/휘핑크림 추가)
>> 2. 디저트 - 마카롱 - 바닐라 - 2000원
>> 3. 티 - 홍차 - 얼그레이(HOT) - 2500원
>> 4. 디저트 - 와플 - 3000원
>> 5. 커피 - 아메리카노(ICE) - 3300원 (샷 추가/바닐라시럽 추가)

몇 번째 목록을 삭제하시겠습니까?
(0을 입력하면 처음으로 돌아갑니다)
>> 2

>> 디저트 - 마카롱 - 바닐라 - 2000원을 삭제합니다.

```

```

(숫자를 입력하시오)
1. 주문 목록 삭제 2. 현재 주문 목록 3. 처음으로 돌아가기
>> 1

현재 주문 목록
>> 1. 논커피 - 초코라떼(ICE) - 5100원 (Size UP/샷 추가/휘핑크림 추가)
>> 2. 티 - 홍차 - 얼그레이(HOT) - 2500원
>> 3. 디저트 - 와플 - 3000원
>> 4. 커피 - 아메리카노(ICE) - 3300원 (샷 추가/바닐라시럽 추가)

몇 번째 목록을 삭제하시겠습니까?
(0을 입력하면 처음으로 돌아갑니다)
>> 0

주문 목록을 삭제하지 않고 처음으로 돌아갑니다.

(숫자를 입력하시오)
1. 메뉴 & 가격 2. 주문 3. 주문 목록 4. 결제 5. 카페 내 여석 6. 퇴실 7. 종료
>>

```

(4) 결제 & 매장 내 섭취 여부 선택 & 매장 내 섭취 시 앉을 좌석 선택

- 설명

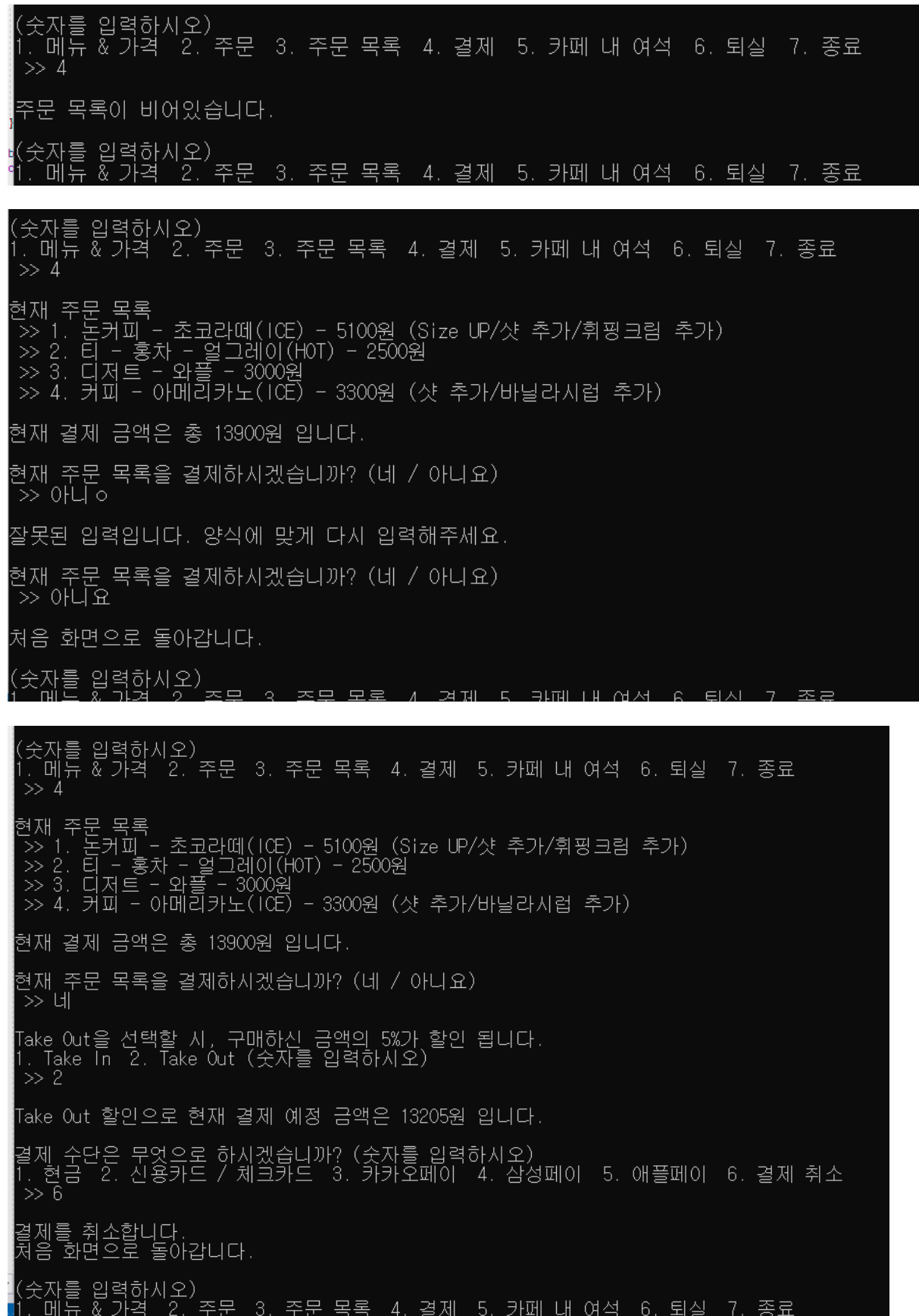
· 처음 화면에서 4가 입력되면 결제를 진행함. 일단 현재 주문 목록과 최종 결제 금액을 출력하고, 결제를 계속해서 진행할 것인지를 물어봄.

· 결제를 진행한다고 선택하면 포장을 할 건지, 아님 매장 내에서 먹을 건지를 물어봄. 포장을 선택하면 결제 금액의 5%를 할인해주고, 결제 수단을 선택해 결제를 마침.

매장 내에서 먹는다고 선택하면, 고객이 앉고자하는 층수와 좌석을 입력받아 해당 좌석이 유효한지 확인하고 고객에게 좌석을 할당함. 그 후 결제 수단을 선택하고 결제를 마침.

· 결제를 취소하는 경우를 제외하고, 올바른 결제 수단을 선택해 결제를 마치면 프로그램도 종료됨.

- 테스트 결과 스크린샷



Take Out을 선택할 시, 구매하신 금액의 5%가 할인 됩니다.

1. Take In 2. Take Out (숫자를 입력하십시오)

>> 1

** 2층은 공부를 위한 장소로 다른 손님들에게 피해가 가지 않도록 조용히 해주세요 **

1층, 2층 중 원하는 층을 선택하십시오. (숫자를 입력하십시오)

(Take Out을 원하는 경우, 0을 입력하십시오. 다시 뒤로 돌아갑니다.)

>> 1

<1층>

```

  ■ ■   □ □   □ □   □ □   ■ ■
[ - ] [ B ] [ C ] [ D ] [ - ]
  ■ ■   □ □   □ □   □ □   ■ ■

  □   □   □   ■   □   □
[ F ] [ G ] [ H ] [ - ] [ J ] [ K ]
  □   □   □   ■   □   □

  ■ ■ ■   □ □ □   □ □ □   ■ ■ ■
[ - ] [ M ] [ N ] [ - ]
  ■ ■ ■   □ □ □   □ □ □   ■ ■ ■

  □   ■   □   □   □   □
[ P ] [ - ] [ R ] [ S ] [ T ] [ U ]
  □   ■   □   □   □   □

  ■ ■   □ □   □ □   ■ ■   □ □
[ - ] [ W ] [ X ] [ - ] [ Z ]
  ■ ■   □ □   □ □   ■ ■   □ □
```

1층에서 원하는 좌석을 입력해주세요. (대문자로 입력하십시오)

(층을 바꾸고자 하는 경우 0을 입력하십시오.)

>> 0

<2층> -Study Only-

```

[ -- ] [ B1 ] [ C1 ] |   | [ D1 ] [ -- ] [ F1 ] [ G1 ]
  ■   □   □   |   |   □   ■   □   □

[ A2 ] [ B2 ] [ C2 ] |   | [ D2 ] [ E2 ] [ F2 ] [ -- ]
  □   □   □   |   |   □   □   □   ■

[ A3 ] [ B3 ] [ C3 ] |   | [ D3 ] [ E3 ] [ F3 ] [ G3 ]
  □   □   □   |   |   □   □   □   □

[ A4 ] [ B4 ] [ -- ] |   | [ D4 ] [ E4 ] [ -- ] [ G4 ]
  □   □   ■   |   |   □   □   ■   □

[ A5 ] [ -- ] [ C5 ] |   | [ -- ] [ E5 ] [ F5 ] [ G5 ]
  □   ■   □   |   |   ■   □   □   □

[ A6 ] [ B6 ] [ C6 ] |   | [ D6 ] [ E6 ] [ -- ] [ G6 ]
  □   □   □   |   |   □   □   ■   □

[ -- ] [ B7 ] [ C7 ] |   | [ D7 ] [ -- ] [ F7 ] [ G7 ]
  ■   □   □   |   |   □   ■   □   □
```

2층에서 원하는 좌석을 입력해주세요. (대문자로 입력하십시오)

(층을 바꾸고자 하는 경우 0을 입력하십시오.)

>> Aq

올바르지 않는 좌석 입력입니다. 다시 입력해주세요.

2층에서 원하는 좌석을 입력해주세요. (대문자로 입력하십시오)

(층을 바꾸고자 하는 경우 0을 입력하십시오.)

>> A1

이미 선택된 좌석입니다. 다른 좌석을 선택해주세요.

2층에서 원하는 좌석을 입력해주세요. (대문자로 입력하십시오)

(층을 바꾸고자 하는 경우 0을 입력하십시오.)

>>

2층에서 원하는 좌석을 입력해주세요. (대문자로 입력하시오)
(층을 바꾸고자 하는 경우 0을 입력하시오.)

>> B1

<2층> -Study Only-

[--] ■	[--] ■	[C1] □		[D1] □	[--] ■	[F1] □	[G1] □
[A2] □	[B2] □	[C2] □		[D2] □	[E2] □	[F2] □	[--] ■
[A3] □	[B3] □	[C3] □		[D3] □	[E3] □	[F3] □	[G3] □
[A4] □	[B4] □	[--] ■		[D4] □	[E4] □	[--] ■	[G4] □
[A5] □	[--] ■	[C5] □		[--] ■	[E5] □	[F5] □	[G5] □
[A6] □	[B6] □	[C6] □		[D6] □	[E6] □	[--] ■	[G6] □
[--] ■	[B7] □	[C7] □		[D7] □	[--] ■	[F7] □	[G7] □

~~ 2층의 B1 좌석이 선택되었습니다.

결제 수단은 무엇으로 하시겠습니까? (숫자를 입력하시오)

1. 현금 2. 신용카드 / 체크카드 3. 카카오페이 4. 삼성페이 5. 애플페이 6. 결제 취소

>>

1층에서 원하는 좌석을 입력해주세요. (대문자로 입력하시오)
(층을 바꾸고자 하는 경우 0을 입력하시오.)

>> B

<1층>

[-] ■ ■	[-] ■ ■	[C] □ □	[D] □ □	[-] ■ ■	
[F] □	[G] □	[H] □	[-] ■	[J] □	[K] □
[-] ■ ■ ■	[M] □ □ □	[N] □ □ □	[-] ■ ■ ■		
[P] □	[-] ■	[R] □	[S] □	[T] □	[U] □
[-] ■ ■	[W] □ □	[X] □ □	[-] ■ ■	[Z] □ □	

~~ 1층의 B 좌석이 선택되었습니다.

결제 수단은 무엇으로 하시겠습니까? (숫자를 입력하시오)

1. 현금 2. 신용카드 / 체크카드 3. 카카오페이 4. 삼성페이 5. 애플페이 6. 결제 취소

>> 2

-- 주문 메뉴 --

현재 주문 목록

>> 1. 돈커피 - 초코라떼(ICE) - 5100원 (Size UP/샷 추가/휘핑크림 추가)

>> 2. 티 - 홍차 - 얼그레이(HOT) - 2500원

>> 3. 디저트 - 와플 - 3000원

>> 4. 커피 - 아메리카노(ICE) - 3300원 (샷 추가/바닐라시럽 추가)

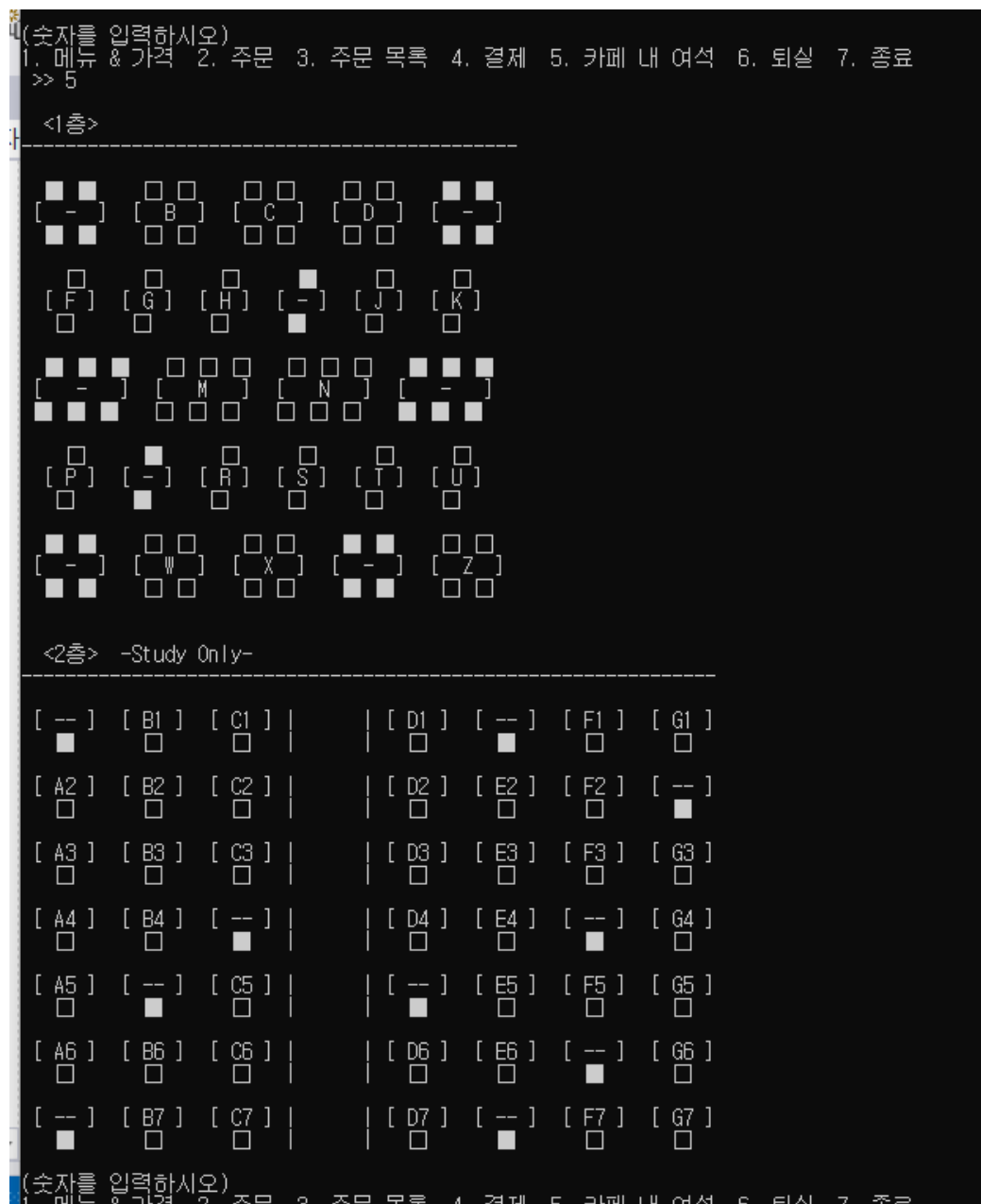
신용카드 / 체크카드(으)로 결제가 완료 되었습니다. 메뉴가 만들어질 때까지 잠시만 기다려 주세요. :)

(5) 카페 내 좌석(여석) 출력

- 설명

- 처음 화면에서 5를 입력하면 현재 카페 내 사용할 수 있는 좌석이 출력됨. 이때 색칠된 네모는 다른 고객이 이미 사용중인 좌석이라는 의미이고, 비어있는 네모는 사용 가능한 좌석이라는 의미임.

- 테스트 결과 스크린샷



(6) 퇴실

- 설명

- 처음 화면에 6이 입력되면 퇴실할 수 있음. 퇴실을 원하는 고객이 앉은 층수와 좌석을 입력하여 해당 좌석에서 나갈 수 있음. 퇴실 작업이 완료되면 프로그램은 종료됨.

- 테스트 결과 스크린샷

```
(숫자를 입력하시오)
1. 메뉴 & 가격 2. 주문 3. 주문 목록 4. 결제 5. 카페 내 여석 6. 퇴실 7. 종료
>> 6

음료/디저트를 잘 즐기셨습니까? 앉았던 층수를 입력해 주세요. (숫자만 입력하시오)
(취소하고자 하는 경우 0을 입력하시오.)
>> 0

퇴실을 취소합니다.
처음 화면으로 돌아갑니다.

(숫자를 입력하시오)
1. 메뉴 & 가격 2. 주문 3. 주문 목록 4. 결제 5. 카페 내 여석 6. 퇴실 7. 종료
```

```
(숫자를 입력하시오)
1. 메뉴 & 가격 2. 주문 3. 주문 목록 4. 결제 5. 카페 내 여석 6. 퇴실 7. 종료
>> 6

음료/디저트를 잘 즐기셨습니까? 앉았던 층수를 입력해 주세요. (숫자만 입력하시오)
(취소하고자 하는 경우 0을 입력하시오.)
>> 2

<2층> -Study Only-
-----
[ A1 ] [ B1 ] [ C1 ] | [ D1 ] [ E1 ] [ F1 ] [ G1 ]
[ ■ ] [ □ ] [ □ ] | [ □ ] [ ■ ] [ □ ] [ □ ]

[ A2 ] [ B2 ] [ C2 ] | [ D2 ] [ E2 ] [ F2 ] [ G2 ]
[ □ ] [ □ ] [ □ ] | [ □ ] [ □ ] [ □ ] [ ■ ]

[ A3 ] [ B3 ] [ C3 ] | [ D3 ] [ E3 ] [ F3 ] [ G3 ]
[ □ ] [ □ ] [ □ ] | [ □ ] [ □ ] [ □ ] [ □ ]

[ A4 ] [ B4 ] [ C4 ] | [ D4 ] [ E4 ] [ F4 ] [ G4 ]
[ □ ] [ □ ] [ ■ ] | [ □ ] [ □ ] [ ■ ] [ □ ]

[ A5 ] [ B5 ] [ C5 ] | [ D5 ] [ E5 ] [ F5 ] [ G5 ]
[ □ ] [ ■ ] [ □ ] | [ ■ ] [ □ ] [ □ ] [ □ ]

[ A6 ] [ B6 ] [ C6 ] | [ D6 ] [ E6 ] [ F6 ] [ G6 ]
[ □ ] [ □ ] [ □ ] | [ □ ] [ □ ] [ ■ ] [ □ ]

[ A7 ] [ B7 ] [ C7 ] | [ D7 ] [ E7 ] [ F7 ] [ G7 ]
[ ■ ] [ □ ] [ □ ] | [ □ ] [ ■ ] [ □ ] [ □ ]

앉았던 좌석을 입력해 주세요. (대문자로 입력하시오)
(층을 바꾸고자 하는 경우 0을 입력하시오.)
>> A5

A5 좌석에 대한 정보가 없습니다. 다시 입력해주세요.

앉았던 좌석을 입력해 주세요. (대문자로 입력하시오)
(층을 바꾸고자 하는 경우 0을 입력하시오.)
>> F6

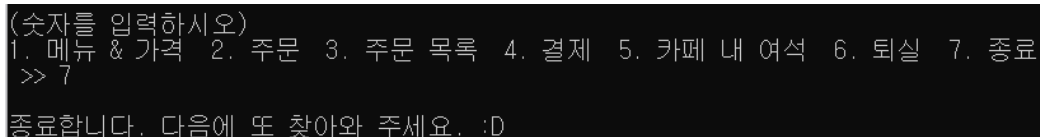
2층의 F6 좌석에서 퇴석하겠습니다.
저희 카페를 찾아와 주셔서 감사합니다. 안녕히 가세요.
```

(7) 종료

- 설명

· 처음 화면에서 7이 입력되면 "종료합니다. 다음에 또 찾아와 주세요."가 출력되고, 프로그램이 종료됨.

- 테스트 결과 스크린샷



```
(숫자를 입력하시오)
1. 메뉴 & 가격  2. 주문  3. 주문 목록  4. 결제  5. 카페 내 여석  6. 퇴실  7. 종료
>> 7

종료합니다. 다음에 또 찾아와 주세요. :D
```

5. 계획 대비 변경 사항

- (없음)

6. 느낀점

여태껏 학교를 다니면서 여러 코딩 실습 수업을 들었지만, 대부분의 수업이 별로 복잡하지 않은 코드를 짜보는 것이어서 그런지, 직접 프로그램의 방향성을 계획하고 코드로 작성해야 하는 해당 프로젝트가 좀 힘들었던 것 같다.

코드를 작성하면서 내가 원하는 출력이 나오지 않은 경우도 종종 있었고, 괜찮은 것 같은 코드도 조금 수정하려다가 갈아엎고 다시 작성한 경우도 많다.

이 프로젝트를 하면서 가장 힘들었던 부분은 cin을 통해 사용자 입력을 받을 때, 코드에서 원하는 타입이랑 사용자가 입력한 타입이 다를 경우 생기는 오류에 대한 처리가 가장 어려웠다. 원래 나는 cin을 통해 사용자에게 선택지에 대한 입력을 받을 때, int 타입으로 받으려고 코드를 작성했는데 그러면 사용자가 정수가 아닌 문자들을 입력한다면 오류가 나서 프로그램을 진행하는데 있어 많은 방해가 되어, 후에 사용자에게 입력 받는 값들은 대부분 string 타입으로 바꾸었다. 그런데 이렇게 바꾸고 보니 string 타입은 부등호를 사용할 수 없다는 단점이 존재했고, 따라서 입력 받은 문자열들을 일일이 '==' 또는 '!='를

통해 비교를 해야 돼 좀 간결하지 않은 코드가 된 것 같다. 후에 오류 처리에 대해 배우고 입력들을 int 타입으로 다시 바꾸어 작성해볼까 했더니, 더욱 길고 복잡한 코드가 되고 생각보다 잘 작동하지 않고 오류가 종종 생겨, int 타입의 입력 받기를 포기하고 다시 string 타입으로 돌아왔다.

이 밖에도 여러 시행착오가 존재했고 이들을 혼자서 해결해 나가는 과정이 힘들었다. 그래도 프로젝트를 통해 복잡한 코드를 직접 짜보고, 해당 코드가 어떻게 돌아가야 하는지 생각하는 과정에서 c++에 대한 지식뿐만 아니라 프로그램이라는 작업에 대해 많은 것을 배운 것 같다.