C++프로그래밍 및 실습

카페 자동화 시스템

진척 보고서 #1

제출일자: 2023.11.26(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 보여주기만 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 500원이 추가 됨. Take out을 선택할 시 500원 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 '사용 중'이라는 표식을 부여.

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

- 설명: 고객이 카페에서 섭취를 완료하고 나갈 때, 자신이 앉은 자리의 좌표를 입력하고 나가도록 함. 해당 자리의 '사용 중' 표식이 다시 좌표로 전환.

3. 진척사항

1) 기능 구현

(1) 카페 메뉴와 해당 가격 제시

- 입출력

입력: 사용자가 입력한 값을 갖는 request 변수, vector<Cafe> 카테고리들.

출력: 카테고리별 메뉴와 그 가격, 서브 메뉴를 출력함.

- 설명

카페 메뉴와 그 가격이 출력되기 위해서는 request 변수에 1이 입력되어야 함. request의 값이 1이 되면 전체 메뉴를 출력하는 TotalPrintMenu() 함수가 실행되고, 이 함수 안에 있는 카테고리별 메뉴를 출력할 PrintMenuCategory() 함수들이 실행됨

PrintMenuCategory() 함수에는 Cafe 객체에 정의된 PrintMenu() 함수를 이용하여 출력을 진행함.

Cafe 객체에 정의된 PrintMenu() 함수에는 각 메뉴들의 출력 형식과 서브 메뉴들의 출력 방법에 대해 정의되어 있음.

여러 함수들을 통해 각 카테고리별로 정리된 메뉴와 가격이 출력됨.

- 적용된 배운 내용

vector, class, 함수, 조건문, for/while 반복문

- 코드 스크린샷

```
void Cafe::PrintMenu() {
    cout << endl;
    cout << setw(18) << menu << setw(10) << price << "원";
    if (!subMenu.empty()) {
        cout << setw(5) << "(";
        for (auto p = subMenu.begin(); p != subMenu.end(); ++p) {
            if (p == subMenu.end() - 1) { cout << *p << ")"; }
            else cout << *p << "/";
        }
    }
}
```

Cafe 클래스에 정의된 함수: 메뉴들을 출력할 형식들에 대해 정의되어 있음

```
// 커피
vector<Cafe> coffee{ Cafe("에스프레소", 2500), Cafe("아메리카노", 2500),
                     Cafe("카푸치노", 3000), Cafe("카페라떼", 3000), Cafe("헤이즐넛라떼", 3500),
                     Cafe("바닐라라떼", 3500), Cafe("카라멜마끼아또", 3500),
                     Cafe("카페모카", 3500), Cafe("돌체라떼", 3700) };

// non 커피
vector<Cafe> non_coffee{ Cafe("곡물라떼", 3300), Cafe("고구마라떼", 3500), Cafe("딸기라떼", 3500),
                         Cafe("초코라떼", 3500), Cafe("그린티라떼", 3500), Cafe("민트초코라떼", 3500) };

// 차(tea)
vector<Cafe> tea{ Cafe("허브티", 2500), Cafe("홍차", 2500),
                  Cafe("복숭아티", 3000), Cafe("허니과일티", 3500) };

// 스무디
vector<Cafe> smoothie{ Cafe("스무디", 4000), Cafe("요거트스무디", 4300) };

// 프라페
vector<Cafe> frappe{ Cafe("자바칩프라페", 4500), Cafe("쿠키프라페", 4500),
                     Cafe("민트초코프라페", 4500), Cafe("그린티프라페", 4500) };

// 에이드
vector<Cafe> ade{ Cafe("자몽에이드", 4000), Cafe("레몬에이드", 4000), Cafe("유자에이드", 4000),
                  Cafe("청포도에이드", 4000), Cafe("패션후르츠에이드", 4000) };

// 디저트
vector<Cafe> dessert{ Cafe("쿠키", 1500), Cafe("마카롱", 2000),
                      Cafe("스콘", 2500), Cafe("머핀", 2500), Cafe("와플", 3000),
                      Cafe("크로플", 3300), Cafe("조각케이크", 4700) };
```

전역변수로 선언된 vector<Cafe>

```
void PrintMenuCategory(int s, string category, vector<Cafe>& items) {
    cout << setw(s) << "<< " << category << " >>";
    if (category == "SMOOTHIE" || category == "FRAPPE" || category == "ADE") {
        cout << " ICE ONLY" << endl;
    }
    else cout << endl;
    cout << "-----------------------------------";
    for (auto& item : items) {
        item.PrintMenu();
    }
    cout << endl;
    cout << "-----------------------------------" << endl;
    cout << endl;
}

void TotalMenuPrint() {
    // 전체 메뉴 출력
    PrintMenuCategory(17, "COFFEE", coffee);
    PrintMenuCategory(15, "NON COFFEE", non_coffee);
    PrintMenuCategory(19, "TEA", tea);
    PrintMenuCategory(11, "SMOOTHIE", smoothie);
    PrintMenuCategory(12, "FRAPPE", frappe);
    PrintMenuCategory(13, "ADE", ade);
    PrintMenuCategory(17, "DESSERT", dessert);
}
```

카테고리별 메뉴와 전체 메뉴를 출력하기 위한 함수

```
int main() {

    // Tea의 subMenu
    vector<string> herb{ "페퍼민트", "캐모마일", "로즈마리" };
    tea[0].SetSubMenu(herb);

    vector<string> black{ "다즐링", "얼그레이" };
    tea[1].SetSubMenu(black);

    vector<string> fruit{ "자몽", "레몬", "유자" };
    tea[3].SetSubMenu(fruit);

    // Smoothie의 SubMenu
    vector<string> plain{ "딸기", "망고", "키위" };
    smoothie[0].SetSubMenu(plain);

    vector<string> yogurt{ "딸기", "망고", "블루베리", "플레인" };
    smoothie[1].SetSubMenu(yogurt);

    // Dessert의 SubMenu
    vector<string> macaron{ "초코", "바닐라", "쿠앤크", "녹차", "딸기", "카라멜" };
    dessert[1].SetSubMenu(macaron);

    vector<string> cake{ "생크림", "초코", "치즈", "고구마", "티라미슈" };
    dessert[6].SetSubMenu(cake);

    vector<FinalOrder> bucket;  // 구매 예정 메뉴들을 담음
```

서브 메뉴를 갖는 것들은 각자의 Cafe 객체에 서브 메뉴 추가

```
while (1) {
    int request;
    bool back = false;

    cout << endl;
    cout << "(숫자를 입력하시오)" << endl;
    cout << "1. 메뉴 & 가격  2. 주문  3. 구매 예정 목록  4. 결제  5. 종료" << endl;
    cout << " >> ";
    cin >> request;

    if (request == 1) {
        TotalMenuPrint();
    }
```

사용자가 1을 입력할 시, 메뉴와 가격 출력

(2) 주문을 받아 구매 예정 목록에 저장

- 입출력

입력: 사용자가 입력한 값을 갖는 request 변수, 주문에 관한 order 변수, 주문할 음료의 카테고리를 갖는 drink 변수, 주문할 메뉴의 이름을 갖는 menu 변수, 서브메뉴를 선택하기 위한 last 변수, 음료의 온도를 선택하기 위한 ice_hot 변수, 추가 옵션을 위한 input 변수.

출력: 사용자가 선택한 값들에 대한 내용들을 출력함.

음료/디저트 중 선택, 원하는 메뉴 선택, 서브 메뉴 선택, 음료의 온도 선택, 추가 옵션 선택.

Ex) request = 2, order = 1, drink = 1, menu = "아메리카노", ice_hot = "ICE", input = "1 2 6"

-> 커피 – 아메리카노(ICE) – 2500원 (Size UP/샷 추가)

- 설명

주문을 위한 코드로, 여러 선택지가 등장하며 해당 선택지에 사용자가 원하는 값들을 입력하고, 최종적으로 올바른 입력은 FinalOrder 객체로 형성되어 vector<FinalOrder> bucket에 push됨.

- 적용된 배운 내용

vector, class, 함수, 조건문, for/while 반복문

- 코드 스크린샷

```
}
else if (request == 2) {
    int order;

    while (1) {
        cout << endl;
        cout << "무엇을 주문하시겠습니까? (숫자를 입력하시오)" << endl;
        cout << "1. 음료  2. 디저트  3. 처음으로 돌아가기" << endl;
        cout << " >> ";
        cin >> order;

        if (order > 3 || order < 1) { PrintError(); continue; }
        else break;
    }
```

```
            if (drink == 7) { cout << endl; back = true; }
            else {
                vector<Cafe> c = Order(drink);
                string menu;

                while (1) {
                    cout << endl;
                    cout << "해당 메뉴에서 원하는 음료를 선택하시오 (문자를 입력하시오. 띄어쓰기 없이)" << endl;
                    cout << " >> ";
                    cin >> menu;

                    int count = 0;
                    for (auto& e : c) {
                        if (e.GetName() == menu) {
                            count++;
                        }
                    }

                    if (count == 0) {
                        PrintError(); continue;
                    }
                    else break;
                }

                string last;
                bool found = false;
                string ice_hot;
                for (auto& e : c) {
                    if (e.GetName() == menu) {
                        Cafe s = e;

                        if (!s.GetSubMenu().empty()) {
                            cout << endl;
                            s.PrintSub();

                            while (1) {
                                cout << "해당 메뉴에서 원하는 맛을 선택하시오 (문자를 입력하시오)" << endl;
                                cout << " >> ";
                                cin >> last;

                                int count = 0;
                                for (auto& t : s.GetSubMenu()) {
                                    if (t == last) {
                                        count++;
                                    }
                                }

                                if (count == 0) {
                                    PrintError(); continue;
                                }
                                else break;
                            }

                            while (1) {
                                if (drink == 1 || drink == 2 || drink == 3) {
                                    cout << endl << "ICE / HOT 중 하나를 선택하시오. (대문자로 입력하시오)" << endl;
                                    cout << " >> ";
                                    cin >> ice_hot;

                                    if (ice_hot == "ICE" || ice_hot == "HOT") {
                                        break;
                                    }
                                    else {
                                        PrintError(); continue;
                                    }
                                }
                                else { ice_hot = "ICE"; break; }
                            }

                            for (auto& t : s.GetSubMenu()) {
                                if (t == last) {
                                    vector<string> topping;
                                    FinalOrder a(drink, menu, last, ice_hot, e.GetPrice());
                                    SelectToppings(topping, a);
                                    a.SetTopping(topping);

                                    bucket.push_back(a);
                                    cout << endl << " ++ ";
                                    a.PrintFinal();
                                    cout << "을 구매 예정 목록에 담았습니다." << endl;
                                    found = true;
                                    break;
                                }
                            }
                        }
```

```
                    else {
                        while (1) {
                            if (drink == 1 || drink == 2 || drink == 3) {
                                cout << endl << "ICE / HOT 중 하나를 선택하시오. (대문자로 입력하시오)" << endl;
                                cout << " >> ";
                                cin >> ice_hot;

                                if (ice_hot == "ICE" || ice_hot == "HOT") {
                                    break;
                                }
                                else {
                                    PrintError(); continue;
                                }
                            }
                            else { ice_hot = "ICE"; break; }
                        }

                        cout << endl;
                        vector<string> topping;
                        FinalOrder b(drink, menu, ice_hot, e.GetPrice());
                        SelectToppings(topping, b);
                        b.SetTopping(topping);

                        bucket.push_back(b);
                        cout << endl << " ** ";
                        b.PrintFinal();
                        cout << "을 구매 예정 목록에 담았습니다." << endl;
                        found = true;
                    }
                    break;
                }
            }
            if (!found) { PrintError(); continue; }
        }
    }
```

```
    ,
    else if (order == 2) {
        PrintMenuCategory(17, "DESSERT", dessert);
        string menu;

        while (1) {
            cout << endl;
            cout << "해당 메뉴에서 원하는 디저트를 선택하시오 (문자를 입력하시오. 띄어쓰기 없이)" << endl;
            cout << " >> ";
            cin >> menu;

            int count = 0;
            for (auto& e : dessert) {
                if (e.GetName() == menu) {
                    count++;
                }
            }

            if (count == 0) {
                PrintError(); continue;
            }
            else break;
        }
```

```
            string last;
            bool found = false;
            for (auto& e : dessert) {
                if (e.GetName() == menu) {
                    Cafe s = e;

                    if (!s.GetSubMenu().empty()) {
                        cout << endl;
                        s.PrintSub();

                        while (1) {
                            cout << "해당 메뉴에서 원하는 맛을 선택하시오 (문자를 입력하시오)" << endl;
                            cout << " >> ";
                            cin >> last;

                            int count = 0;
                            for (auto& t : s.GetSubMenu()) {
                                if (t == last) {
                                    count++;
                                }
                            }

                            if (count == 0) {
                                PrintError(); continue;
                            }
                            else break;
                        }
```

```
                    for (auto& t : s.GetSubMenu()) {
                        if (t == last) {
                            cout << endl;
                            FinalOrder a(menu, last, e.GetPrice());
                            bucket.push_back(a);
                            cout << " ** ";
                            a.PrintFinal();
                            cout << "을 구매 예정 목록에 담았습니다." << endl;
                            found = true;
                            break;
                        }
                    }
                }
                else {
                    cout << endl;
                    FinalOrder b(menu, e.GetPrice());
                    bucket.push_back(b);
                    cout << " ** ";
                    b.PrintFinal();
                    cout << "을 구매 예정 목록에 담았습니다." << endl;
                    found = true;
                }
                break;
            }
        }
        if (!found) { PrintError(); continue; }
    }
    else { cout << endl; back = true; }
}
```

```
void SelectToppings(vector<string>& toppings, FinalOrder& f) {
    vector<int> addT;

    while (1) {
        bool error = false;
        int six_count = 0;

        int input;
        cout << endl << "원하는 추가 옵션을 모두 입력하시오. (숫자를 입력하시오. 6을 입력하면 종료됩니다.)" << endl;
        cout << "1. Size UP(+800원) 2. 샷 추가(+500원) 3. 휘핑크림 추가(+300원) 4. 바닐라시럽 추가(+300원) 5. 펄 추가(+500원) 6. NO" << endl;
        cout << " >> ";
        while (1) {
            cin >> input;
            if (input == 6) break;
            addT.push_back(input);
        }

        for (auto& e : addT) {
            if (e < 1 || e > 6) {
                error = true;
                break;
            }
        }

        if (error == true) {
            PrintError();
            addT.clear();
            continue;
        }
        else break;
    }
```

```
    for (auto& e : addT) {
        if (e == 1) {
            toppings.push_back("Size UP");
            f.IncreasePrice(800);
        }
        else if (e == 2) {
            toppings.push_back("샷 추가");
            f.IncreasePrice(500);
        }
        else if (e == 3) {
            toppings.push_back("휘핑크림 추가");
            f.IncreasePrice(300);
        }
        else if (e == 4) {
            toppings.push_back("바닐라시럽 추가");
            f.IncreasePrice(300);
        }
        else if (e == 5) {
            toppings.push_back("펄 추가");
            f.IncreasePrice(500);
        }
        else {
            if (!toppings.empty()) {
                cout << "추가 옵션 선택을 종료합니다. 선택하신 옵션은 ";
                f.PrintTopping();
                cout << "입니다." << endl;
            }
            else {
                cout << "추가적으로 선택하지 않았습니다." << endl;
            }
        }
    }
}
```

추가 옵션 선택을 위한 함수

(3) 구매 예정 목록을 출력하고 삭제

- 입출력

입력: 사용자가 입력한 값을 갖는 request 변수, 구매 예정 목록에 대해 실행 사항을 부여할 orderList 변수, 주문을 통해 저장된 vector<FinalOrder> bucket, 삭제할 목록의 순서를 나타내는 del_Index 변수.

출력: 사용자가 선택한 값들에 대한 내용들을 출력함. orderList = 1일 때, 원하는 목록을 삭제할 수 있음. orderList = 2일 때, 현재 주문 예정 목록을 출력함.

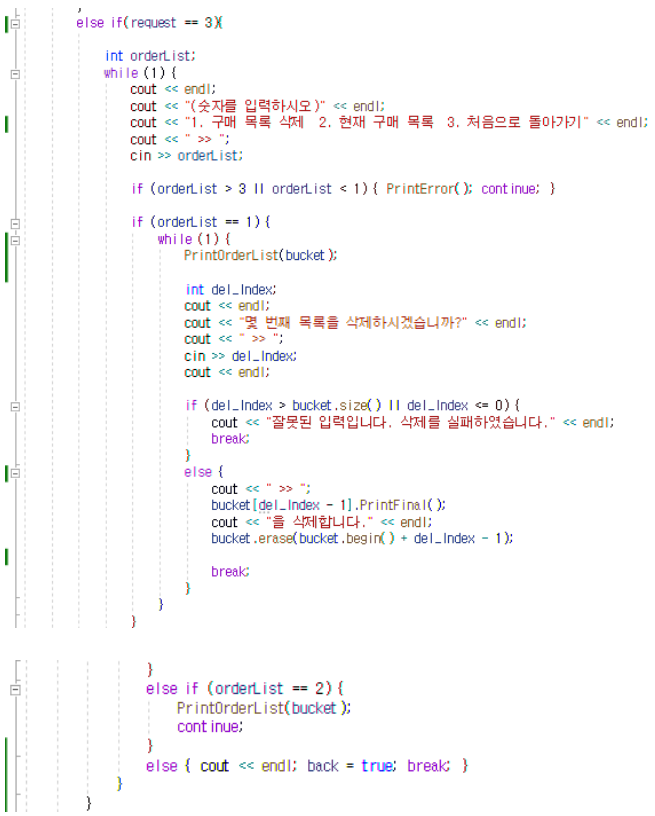
- 설명

위에서 주문을 통해 bucket에 저장한 주문 목록들을 볼 수 있고, 원하는 목록을 삭제할 수 있다.

- 적용된 배운 내용

vector, class, 함수, 조건문, for/while 반복문

- 코드 스크린샷



```
        ,
    else if(request == 3){

        int orderList;
        while (1) {
            cout << endl;
            cout << "(숫자를 입력하시오)" << endl;
            cout << "1. 구매 목록 삭제  2. 현재 구매 목록  3. 처음으로 돌아가기" << endl;
            cout << " >> ";
            cin >> orderList;

            if (orderList > 3 || orderList < 1) { PrintError(); continue; }

            if (orderList == 1) {
                while (1) {
                    PrintOrderList(bucket);

                    int del_Index;
                    cout << endl;
                    cout << "몇 번째 목록을 삭제하시겠습니까?" << endl;
                    cout << " >> ";
                    cin >> del_Index;
                    cout << endl;

                    if (del_Index > bucket.size() || del_Index <= 0) {
                        cout << "잘못된 입력입니다. 삭제를 실패하였습니다." << endl;
                        break;
                    }
                    else {
                        cout << " >> ";
                        bucket[del_Index - 1].PrintFinal();
                        cout << "을 삭제합니다." << endl;
                        bucket.erase(bucket.begin() + del_Index - 1);

                        break;
                    }
                }
            }
```

```
            }
            else if (orderList == 2) {
                PrintOrderList(bucket);
                continue;
            }
            else { cout << endl; back = true; break; }
        }
    }
```

(4) 구매 예정 목록을 통해 지불할 가격을 구하고 결제 수단 선택

- 입출력

입력: 사용자가 입력한 값을 갖는 request 변수, 주문 목록이 저장된 vector<FinalOrder> bucket,

결제 여부를 나타내는 pay 변수, 포장을 할 것인지 여부를 가지는 take 변수, 결제 수단을 나타내는 method 변수.

출력: 주문 목록에 존재하는 주문들의 가격의 합을 출력함. 결제 여부를 물어봄. 매장 내 섭취 여부를 물어봄. 결제 수단을 물어봄.

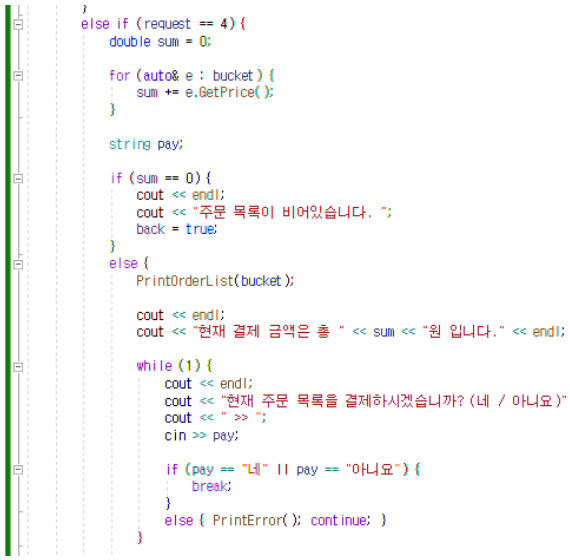
- 설명

최종 지불할 금액을 출력하고 결제 수단을 선택해 결제할 수 있음. 포장 시 전체 가격의 5%를 할인해주는 혜택이 존재.

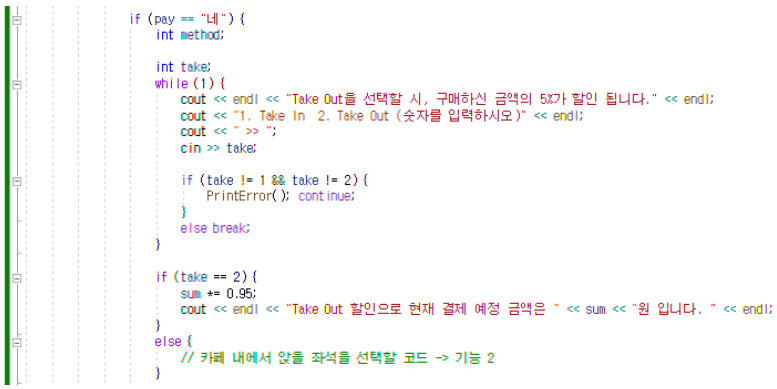
- 적용된 배운 내용

vector, class, 함수, 조건문, for/while 반복문

- 코드 스크린샷



```
}
else if (request == 4) {
    double sum = 0;

    for (auto& e : bucket) {
        sum += e.GetPrice();
    }

    string pay;

    if (sum == 0) {
        cout << endl;
        cout << "주문 목록이 비어있습니다. ";
        back = true;
    }
    else {
        PrintOrderList(bucket);

        cout << endl;
        cout << "현재 결제 금액은 총 " << sum << "원 입니다." << endl;

        while (1) {
            cout << endl;
            cout << "현재 주문 목록을 결제하시겠습니까? (네 / 아니요)" << endl;
            cout << " >> ";
            cin >> pay;

            if (pay == "네" || pay == "아니요") {
                break;
            }
            else { PrintError(); continue; }
        }
```



```
        if (pay == "네") {
            int method;

            int take;
            while (1) {
                cout << endl << "Take Out을 선택할 시, 구매하신 금액의 5%가 할인 됩니다." << endl;
                cout << "1. Take In  2. Take Out (숫자를 입력하시오)" << endl;
                cout << " >> ";
                cin >> take;

                if (take != 1 && take != 2) {
                    PrintError(); continue;
                }
                else break;
            }

            if (take == 2) {
                sum += 0.95;
                cout << endl << "Take Out 할인으로 현재 결제 예정 금액은 " << sum << "원 입니다. " << endl;
            }
            else {
                // 카페 내에서 앉을 좌석을 선택할 코드 -> 기능 2
            }
```

```
            while (1) {
                cout << endl;
                cout << "결제 수단은 무엇으로 하시겠습니까?(숫자를 입력하시오)" << endl;
                cout << "1. 현금  2. 신용카드 / 체크카드  3. 카카오페이  4. 삼성페이  5. 애플페이  6. 결제 취소" << endl;
                cout << " >> ";
                cin >> method;
                cout << endl;

                if (method < 1 || method > 6) {
                    PrintError(); continue;
                }
                else break;
            }

            if (method == 6) {
                cout << "결제를 취소합니다. ";
                back = true;
            }
            else {
                string m;
                if (method == 1) m = "현금";
                else if (method == 2) m = "신용카드 / 체크카드";
                else if (method == 3) m = "카카오페이";
                else if (method == 4) m = "삼성페이";
                else  m = "애플페이";

                cout << m << "(으)로 결제가 완료 되었습니다. 메뉴가 만들어질 때까지 잠시만 기다려 주세요. :)" << endl;
                break;

            }
        }
        else cout << endl; back = true;
    }
}
```

2) 테스트 결과

(1) 카페 메뉴와 해당 가격 제시

- 설명

전체 카페 메뉴와 그 가격, 서브 메뉴를 카테고리별로 출력.

- 테스트 결과 스크린샷

```
(숫자를 입력하시오)
1. 메뉴 & 가격  2. 주문  3. 구매 예정 목록  4. 결제  5. 종료
 >> 1
                << COFFEE >>
-----------------------------------
        에스프레소       2500원
        아메리카노       2500원
          카푸치노       3000원
          카페라떼       3000원
     헤이즐넛라떼       3500원
        바닐라라떼       3500원
    카라멜마끼아또       3500원
          카페모카       3500원
          돌체라떼       3700원
-----------------------------------

             << NON COFFEE >>
-----------------------------------
          곡물라떼       3300원
        고구마라떼       3500원
          딸기라떼       3500원
          초코라떼       3500원
        그린티라떼       3500원
      민트초코라떼       3500원
-----------------------------------

                 << TEA >>
-----------------------------------
            허브티       2500원    (페퍼민트/캐모마일/로즈마리)
            홍차       2500원    (다즐링/얼그레이)
          복숭아티       3000원
        허니과일티       3500원    (자몽/레몬/유자)
-----------------------------------
```

```
       << SMOOTHIE >> ICE ONLY
-----------------------------------
          스무디       4000원    (딸기/망고/키위)
      요거트스무디      4300원    (딸기/망고/블루베리/플레인)
-----------------------------------

        << FRAPPE >> ICE ONLY
-----------------------------------
      자바칩프라페       4500원
        쿠키프라페       4500원
    민트초코프라페       4500원
      그린티프라페       4500원
-----------------------------------

         << ADE >> ICE ONLY
-----------------------------------
        자몽에이드       4000원
        레몬에이드       4000원
        유자에이드       4000원
      청포도에이드       4000원
   패션후르츠에이드      4000원
-----------------------------------

            << 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)
-----------------------------------
```

(2) 주문을 받아 구매 예정 목록에 저장

- 설명

단계적 선택을 통한 주문. 이때 한 주문은 구매 예정 목록에 저장됨.

- 테스트 결과 스크린샷

```
(숫자를 입력하시오)
1. 메뉴 & 가격  2. 주문  3. 구매 예정 목록  4. 결제  5. 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
-----------------------------------


해당 메뉴에서 원하는 음료를 선택하시오 (문자를 입력하시오. 띄어쓰기 없이)
 >> 바닐라라떼

ICE / HOT 중 하나를 선택하시오. (대문자로 입력하시오)
 >> ICE


원하는 추가 옵션을 모두 입력하시오. (숫자를 입력하시오. 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 6

 ** 커피 - 바닐라라떼(ICE) - 4800원 (Size UP/샷 추가)을 구매 예정 목록에 담았습니다.
```

```
(숫자를 입력하시오)
1. 메뉴 & 가격  2. 주문  3. 구매 예정 목록  4. 결제  5. 종료
 >> 2

무엇을 주문하시겠습니까? (숫자를 입력하시오)
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)
-----------------------------------


해당 메뉴에서 원하는 디저트를 선택하시오 (문자를 입력하시오. 띄어쓰기 없이)
 >> 마카롱

 -> 초코 / 바닐라 / 쿠앤크 / 녹차 / 딸기 / 카라멜

해당 메뉴에서 원하는 맛을 선택하시오 (문자를 입력하시오)
 >> 카라멜

 ** 디저트 - 마카롱 - 카라멜 - 2000원을 구매 예정 목록에 담았습니다.
```

(3) 구매 예정 목록을 출력하고 삭제

- 설명

주문을 통해 저장된 목록들을 삭제, 관리할 수 있음.

- 테스트 결과 스크린샷

```
(숫자를 입력하시오)
1. 메뉴 & 가격  2. 주문  3. 구매 예정 목록  4. 결제  5. 종료
 >> 3

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

현재 구매 예정 목록
 >> 1. 커피 - 바닐라라떼(ICE) - 4800원 (Size UP/샷 추가)
 >> 2. 디저트 - 마카롱 - 카라멜 - 2000원

몇 번째 목록을 삭제하시겠습니까?
 >> 2

 >> 디저트 - 마카롱 - 카라멜 - 2000원을 삭제합니다.
```

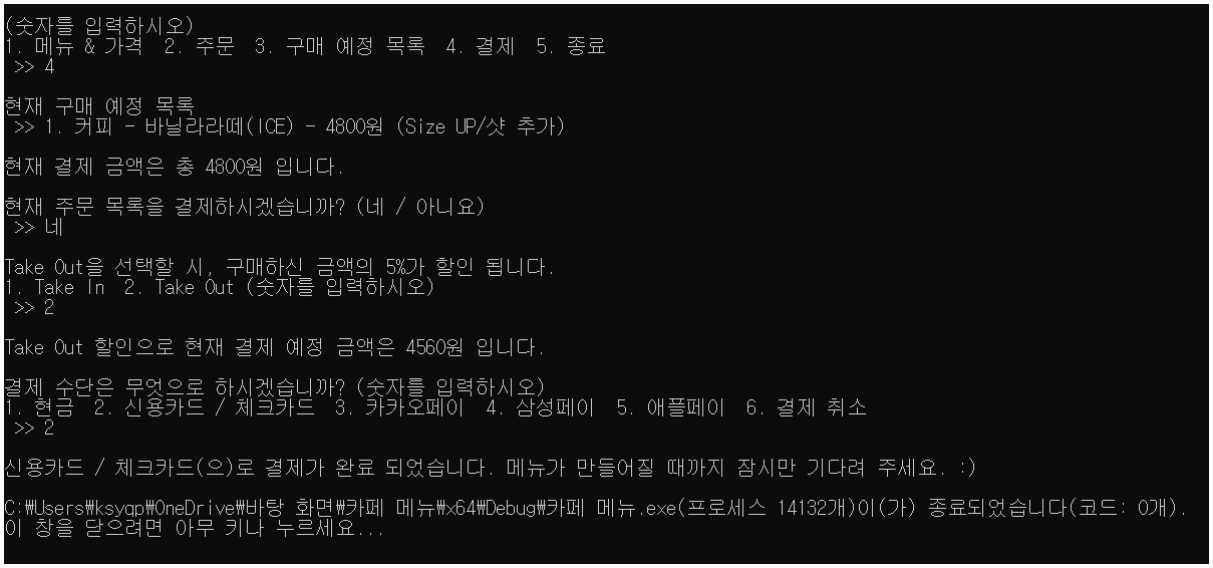
```
(숫자를 입력하시오)
1. 구매 목록 삭제  2. 현재 구매 목록  3. 처음으로 돌아가기
 >> 2

현재 구매 예정 목록
 >> 1. 커피 - 바닐라라떼(ICE) - 4800원 (Size UP/샷 추가)

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

다시 처음으로 돌아갑니다.
```

(4) 구매 예정 목록을 통해 지불할 가격을 구하고 결제 수단 선택

- 설명

지불해야할 전체 가격을 출력하고, 결제 수단을 선택하여 결제를 진행함. 이때 포장 구매 시 5% 할인 혜택이 존재.

- 테스트 결과 스크린샷



4. 계획 대비 변경 사항

(없음. 이전과 동일하게 진행할 예정)

5. 프로젝트 일정

업무		11/3	11/12	11/19	11/26	12/17	12/22
제안서 작성		완료					
기능1	세부 기능1	완료					
	세부 기능2		완료				
	세부 기능3			완료			
기능2	세부 기능1				---------->		
	세부 기능2					------->	

최종 프로그램 & 보고서					**---------->**