C++ 프로그래밍 및 실습

쿠폰 자동 지급

최종 보고서

제출일자: 2023/12/24
제출자명: 박지성
제출자학번: 200791

1. 프로젝트 목표

1) 배경 및 필요성

온라인 매장이 발달함으로써, 소비자들이 예전처럼 물건을 오프라인으로 구매하는 추세가 적어지고, 많은 사람들이 온라인 매장을 이용하기 시작함. 때문에 온라인 매장은 더 많은 소비자를 끌어들이기 위해서 여러 컨텐츠나 이벤트를 적극 활용 중임. 따라서, 소비자의 더 많은 소비를 증진시키고, 더 나은 소비 여건을 만족시키기 위해서 고객의 수요에 맞춘 이벤트 프로그램이 필요함.

2) 프로젝트 목표

고객의 과거 주문 내역에 따라 자주 주문했던 테마를 참고하여 할인 쿠폰을 자동 지급하는 프로그램을 만드는 것을 목표로 함. 또한, 등급에 따라 쿠폰의 할인율, 개수 등을 차등으로 지급하는 것까지 고려함.

3) 차별점

기존의 쿠폰 지급 방식은 고객 각각의 취향에 따라 지급되는 것이 아니라, 하나의 테마를 할인하는 방식으로 모두에게 같은 쿠폰을 일괄 지급하는 방식임. 이러한 방식은 그 테마를 좋아하는 고객에게만 유용한 할인 쿠폰 방식이었을 수 있기 때문에, 고객 각각의 구매 취향에 따라서 자신에게 맞는 할인 쿠폰을 지급하는 방식으로 개선해보려고 한다. 또, 매장 이용 빈도수에 따라 등급을 매기고, 그 등급에 따라서 쿠폰의 할인율, 쿠폰의 개수를 다르게 하는 기존의 방식은 그대로 채용한다.

2. 기능 계획

1) 기능 1 (고객의 구매 내역에 맞는 할인 쿠폰 지급)

① 고객의 구매 내역 수집
1) 위 프로그램에서는 데이터베이스 등을 이용할 수 없으므로, 구매 내역은 수집보다는 저장되어 있는 내역으로 사용함.

② 구매 내역에 따라서 각각에 맞는 테마 분류
1) 구매 내역에 따라 소비자를 지정된 테마로 분류 작업

③ 테마에 따라서 맞는 쿠폰 지급

2) 기능 2

① 고객의 구매 빈도, 구매한 양 등을 수집
1) 기능 1-1과 동일

② 구매 빈도, 구매한 양에 따라 등급 분류 및 지정
1) 구매 횟수가 일정 구간에 도달하면 쿠폰을 지급 하는 방식 (counter)
2) 등급에 도달하면 쿠폰 지급과 등급 도달 문구를 출력
3) 현재 customer 의 보유 쿠폰 개수 출력

3. 기능 구현

1. 대소문자 구분 없이 문자열 비교

- 입출력 : 매개변수: str1, str2 (비교하려는 두 문자열)

- 설명 : 대소문자를 구분하지 않고, 두 문자열을 비교하여 동일한지 여부를 참과 거짓으로 반환. 각 문자를 소문자로 변환한 후에 시작.

- 코드 스크린샷

```
// 대소문자 구분 없이 두 문자열을 비교하는 함수
bool caseInsensitiveCompare(const string& str1, const string& str2) {
    return equal(str1.begin(), str1.end(), str2.begin(), str2.end(), [](char a, char b) {
        return tolower(a) == tolower(b);
        });
}
```

2. 테마와 고객 구매 횟수 체크

- 입출력 : customerName - 구매 내역을 조회할 특정 고객의 이름

theme - 구매 내역을 조회할 특정 테마

- 설명 : 특정 고객이 특정 테마를 몇 번 구매했는 지 세는 함수. 입력으로 주어진 고객 이름과 테마에 대해 대소문자를 구분하지 않고 구매 내역 확인 후 일치하면 구매 횟수 증가

- 코드 스크린샷

```
// 특정 테마와 고객에 대한 구매 횟수를 세는 함수
int countTheme(const string& customerName, const string& theme) {
    int count = 0;
    for (int i = 0; i < numPurchases; ++i) {
        if (caseInsensitiveCompare(customerNames[i], customerName) && caseInsensitiveCompare(themes[i], theme)) {
            count++;
        }
    }
    return count;
}
```

3. 구매 내역을 배열에 추가

- 매개변수: customerName - 구매한 고객의 이름
  item - 구매한 상품의 이름
  theme - 구매한 상품의 테마
- 설명 : 새로운 구매 내역을 배열에 추가.
- 코드 스크린샷

```
// 구매 내역을 데이터 배열에 추가하는 함수
void addPurchase(const string& customerName, const string& item, const string& theme) {
    if (numPurchases < MAX_PURCHASES) {
        customerNames[numPurchases] = customerName;
        items[numPurchases] = item;
        themes[numPurchases] = theme;
        couponCounts[numPurchases] = 0;
        numPurchases++;
    }
    else {
        cout << "더 이상 구매 내역을 추가할 수 없습니다." << endl;
    }
}
```

4. 구매 내역 출력

- 입출력 : customerName - 구매한 고객의 이름
- 설명 : 특정 고객의 구매 내역을 출력.

- 코드 스크린샷

```
// 특정 고객의 구매 내역을 출력하는 함수
void printCustomerPurchases(const string& customerName) {
    cout << customerName << "님의 구매 내역" << endl;
    for (int i = 0; i < numPurchases; ++i) {
        if (caseInsensitiveCompare(customerNames[i], customerName)) {
            cout << "구매했던 상품: " << items[i] << " / 테마: " << themes[i];
            if (couponCounts[i] > 0) {
                cout << " / 할인 쿠폰 지급: " << couponCounts[i] << "장";
            }
            cout << endl;
        }
    }
    cout << endl;
}
```

5. 테마에 대한 할인 쿠폰 지급

- 입출력 : theme - 할인 쿠폰을 지급할 특정 테마

couponCount - 지급할 할인 쿠폰의 개수

- 설명 : 모든 고객에게 특정 테마에 대한 할인 쿠폰을 지급

- 코드 스크린샷

```
// 모든 고객에게 특정 테마에 대한 할인 쿠폰을 지급하는 함수
void grantCouponForThemeToAllCustomers(const string& theme, int couponCount) {
    for (int i = 0; i < numPurchases; ++i) {
        if (caseInsensitiveCompare(themes[i], theme)) {
            couponCounts[i] += couponCount;
            cout << customerNames[i] << "님에게 " << theme << " 테마에 대한 할인 쿠폰 " << couponCount << "장을 지급했습니다." << endl;
        }
    }
}
```

6. 고객에 대한 할인 쿠폰 지급

- 입출력 : customerName - 할인 쿠폰을 지급할 특정 고객의 이름

couponCount - 지급할 할인 쿠폰의 개수

- 설명 : 특정 고객에게 할인 쿠폰을 지급. 지급된 할인 쿠폰 개수와 함께 해당 내역을 출력.

- 코드 스크린샷

```
// 특정 고객에 대한 할인 쿠폰을 지급하는 함수
void grantCouponForCustomer(const string& customerName, int couponCount) {
    for (int i = 0; i < numPurchases; ++i) {
        if (caseInsensitiveCompare(customerNames[i], customerName)) {
            couponCounts[i] += couponCount;
            cout << customerName << "님에게 할인 쿠폰 " << couponCount << "장을 지급했습니다." << endl;
        }
    }
}
```

4. 테스트 결과

(1) 저장된 고객의 이름 입출력

- 설명 : 고객의 이름을 입력 (ex. 권성준, 이현인, 황건하)

- 테스트 결과 스크린샷

```
고객 이름을 입력하세요: 이현인
```

(2). 입력한 고객의 구매내역 출력

- 설명 : 특정 고객에 따라 저장된 구매내역을 출력해서 사용자에게 보여줌.

- 테스트 결과 스크린샷

```
이현인님의 구매 내역
구매했던 상품: 쌀 20kg / 테마: 식품
```

(3). 할인 쿠폰 지급

- 설명 : 기본적으로 구매 내역이 존재하는 고객에게는 할인 쿠폰 지급

- 테스트 결과 스크린샷

(4). 테마에 맞는 할인 쿠폰 자동 지급

- 설명 : 구매내역에 어떤 테마가 존재하는 지를 확인한 후, 테마에 맞는 할인 쿠폰 지급

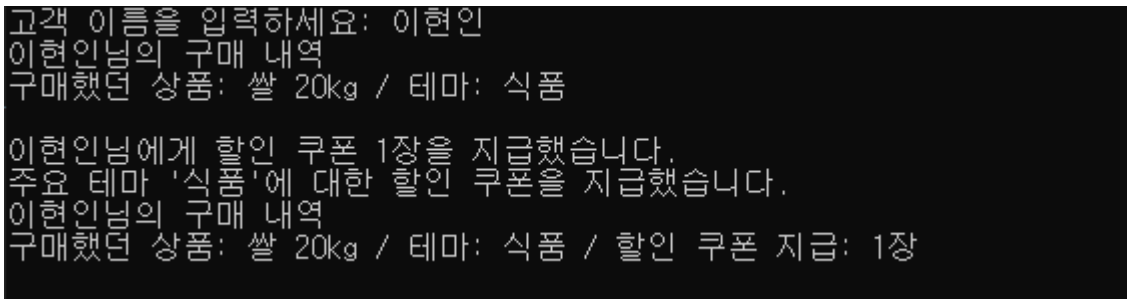
- 테스트 결과 스크린샷

(5). 쿠폰이 지급 되었는 지 재확인

- 설명 : 할인 쿠폰 지급이 되었는 지 재확인 하는 출력문

- 테스트 결과 스크린샷

(6). 전체 스크린샷



- 설명 : "purchaseData.cpp" 에 있는 특정 고객을 입력하면, 특정 고객에 맞는 쿠폰 지급

5. 계획 대비 변경 사항

- 등급제로 쿠폰을 지급하는 방식을 구현하지 못했고, 쿠폰에 대한 할인율을 나누는 것도 구현하지 못했음. 헤더 파일로 나누면서, 프로그램을 만드는 방식에 익숙하지 않은 점도 있었고, 데이터베이스를 배열로 표현하려는 시도 자체가 새로운 도전이었지만, 쉽지 않았다. 가능하면, 사용자에 많은 구매내역을 추가하여 구매내역의 테마중 가장 많이 나온 테마에 대해서 쿠폰을 지급하는 방식으로 만들어 보고 싶었는데, 마지막까지 잘 되지 않아서, 결국 구매내역을 하나만 추가하는 방식으로 간단하게 만들었다.

6. 느낀점

- 장기적인 시간을 두고 프로그램 하나를 깎는다는 것이 얼마나 쉽지 않은 일인지를 깨달았다. 그만큼 주제 선정에도 신중해야 했고, 굳이 수업시간에 배웠던 모든 것들을 사용하려고 시도하지 않았어도 됐을텐데, 사소한 것에 집착하여 너무 수준에 맞지 않는 프로그램을 짜려고 한 잘못이 있었던 것 같다. 앞으로도 학년이 올라가면서, 더 많은 프로젝트 수업을 진행하게 될 것이고, 지금 내가 했던 프로그램은 쉬웠다, 라고 생각될 단계도 언젠가 오겠지만, 그런 단계가 아직 오기 전에 이렇게 수업적으로, 스스로 한 번 해볼 수 있었던 것에 대해 긍정적으로 평가한다.

이렇게 당당하게 말할 정도로 프로그램에 신경을 쓰지 않은 것도 사실이지만, 개인적으로 수업적인 면만 봤을 때, 다른 공대 수업보다 훨씬 스스로 발전할 수 있는 여지를 많이 주는 수업이라고 생각한다. 비교하는 것은 아니지만, 모든 수업에 만족할 수 있는 것은 아니라고 생각하기에, 수업에 많은 준비를 해주시고, 학생들을 신경써주는 점에서 이 수업은 좋은 수업이었다.