C++프로그래밍및실습

물품 대여 관리 시스템

최종 보고서

제출일자: 2023-12-23

이름: 안호균

학번: 190025

1. 프로젝트 목표

1) 배경 및 필요성

저는 현재 소프트웨어공학과 학생회의 일원으로서 활동하고 있습니다. 소프트웨어공학과 학생회는 학생들의 편의성 증진을 위해 여러가지 학생 사업을 진행하는데, 그 중 하나로 학과 학생들에게 우산, 보조배터리, 돗자리와 같이 학생들이 특정 상황에 필요 할만한 물품을 대여해주는 '물품 대여 사업'이 있습니다.

하지만 현재 물품의 대여 및 반환을 기록하는 일련의 과정은 모두 수기로 작성되고 있어, 이로 인해 발생하는 여러가지 문제점들이 있었습니다. 물품을 대여한 학생들에 대한 정보 관리, 대여 가능한 물품의 수량 관리, 물품 분실 및 파손 등의 상태 관리, 대여 가능한 물품이 더 이상 남아있지 않을 때 학생들이 이를 인지하지 못하고 방문을 하는 경우 등 여러 문제가 발생했으며, 무엇보다 물품의 대여 및 반납 기록을 관리하면서 발생하는 학생회 인원의 인력 및 시간적 낭비가 지속적으로 발생했습니다.

이러한 배경 속에서 학생회를 위한 효율성을 높인 자동화된 대여 물품 관리 시스템을 고안하게 되었습니다. 이를 통해 물품의 대여 및 반납 과정을 체계적으로 관리하고, 물품의 상태를 관리하며, 대여한 학생의 정보를 정확하게 관리하게 될 것이라는 기대를 가지고 있습니다.

2) 프로젝트 목표

1. 체계적인 물품 관리

- 물품 관리 시스템의 도입으로, 학생회가 보유하고 있는 모든 대여 물품의 현황을 실시간으로 파악하고 관리할 수 있도록 합니다.
- 물품의 추가, 대여, 반환 및 현황 조회 등의 기능을 통합하여 제공하여 관리 과정의 복잡성을 최소화합니다.

2. 높은 데이터의 정확성 및 투명성 확보

- 대여 및 반환 기록을 시스템화 하여 기존의 수기 작성 방식에 비해 오류를 줄이고 데이터 안전성을 높일 수 있으며, 물품 대여자 정보를 기록으로 남김으로써 해당 물품 관련 정보의 투명성을 제공하여 불필요한 오해나 실수를 방지합니다.

3. 관리 편의성 제고

- 학생회 관리 인원이 시스템을 통해 대여 가능한 물품의 수량, 상태 등의 정보를 손쉽게 확인하고, 대여 및 반환 절차를 간편하게 관리할 수 있도록 합니다.

4. 장기적인 데이터 관리 및 활용

- 시스템의 기록된 데이터를 파일에 저장함으로써 장기적으로 보관하며, 학생들의 물품 대여 추세를 파악하여 물품의 수량 추가 여부를 판단하는데 활용할 수 있습니다.

3) 차별점

소프트웨어공학과 학생회의 물품 대여 관리 시스템은 학생회의 기능적 필요성에 의해 최적화되어 설계되었습니다. 특히, 이 시스템은 물품의 대여 및 반환 정보를 파일로 저장하여 기록하므로, 학생회 내외부에 투명하게 정보 제공이 가능하며, 이를 통해 발생할 수 있는 오해나 갈등을 미연에 방지할 수 있습니다. 또한, 본 시스템을 통해 데이터 분석이 가능합니다. 학생회에서는 이를 활용해 대여 추세를 분석할 수 있으며, 이를 기반으로 물품의 수량 추가 여부를 판단할 수 있으며, 수요 분석을 기반으로 앞으로 추가하면 좋을 물품을 결정하는 데에도 활용할 수 있습니다.

2. 기능 계획

1. **물품 추가**

    - 학생회 관리자는 새로운 대여 물품을 시스템에 등록할 수 있습니다. 물품 이름, 총 재고량 등 필요한 정보를 입력하여 추가합니다.

2. **물품 대여**

    - 학생들이 원하는 물품을 대여할 수 있습니다. 물품을 대여할 때 소프트웨어공학과 소속임을 확인할 학과 정보, 학번, 이름을 함께 입력하여 대여 기록을 남깁니다.

3. **물품 반납**

- 대여한 물품을 반납하는 기능입니다. 반납 시 대여자의 학번을 입력 받아 반납 대상자를 파악하고, 해당 물품의 재고를 복구합니다.

4. 모든 물품 리스트 보기

- 등록된 모든 물품의 목록을 조회할 수 있습니다. 각 물품의 총 수량, 재고량, 대여 중인 수량 등 상세한 정보가 함께 표시됩니다.

5. 대여 중인 물품 리스트 보기

- 현재 대여 중인 물품의 목록과 대여자 정보를 조회할 수 있습니다. 학생회 관리자는 이 기능을 통해 대여 현황을 확인할 수 있습니다.

6. 파일에 저장하기

- 현재 시스템의 데이터, 즉 물품 정보와 대여 정보 및 물품 재고를 파일로 저장할 수 있습니다. 이를 통해 데이터 백업과 이후 데이터 복구가 가능합니다.

7. 파일에서 불러오기

- 저장된 파일을 불러와 시스템의 데이터를 복구할 수 있습니다. 예기치 못한 상황에서 데이터 손실이 발생했을 때, 이 기능을 활용하여 데이터를 복원합니다.

8. 프로그램 종료

- 물품 대여 관리 프로그램을 안전하게 종료합니다.

3. 기능 구현

(1) 라이브러리, 네임스페이스, 클래스 선언 및 정의

```
#include <iostream>
#include <vector>
#include <string>
#include <chrono>
#include <iomanip>
#include <fstream>
#include <sstream>

using namespace std;
using namespace chrono;
```

- **라이브러리**: 입출력, 벡터, 문자열, 시간 요소, 유효 숫자 포맷팅, 파일 입출력을 위해 필요한 라이브러리들을 포함시킵니다.
- **네임스페이스:** 편의를 위해 std와 chrono의 네임스페이스를 사용합니다.

```
void viewItems();

// Item 클래스: 물품 정보를 저장한다.
class Item {
public:
    string name;  // 물품 이름
    int totalQuantity;  // 물품의 총 수량
    int availableQuantity;  // 현재 대여 가능한 물품 수량

    // 생성자: 물품 이름과 수량을 초기화한다.
    Item(string n, int q) : name(n), totalQuantity(q), availableQuantity(q) {}

    // 생성자2
    Item(string n, int totalQ, int availableQ) : name(n), totalQuantity(totalQ), availableQuantity(availableQ) {}
};

// Renter 클래스: 대여자 정보를 저장한다.
class Renter {
public:
    string department;  // 대여자의 학과
    string studentID;  // 대여자의 학번
    string studentName;  // 대여자의 이름
    string itemName;  // 대여한 물품 이름
    system_clock::time_point rentedTime;  // 대여한 시간
    system_clock::time_point dueTime;  // 반납 예정 시간
    bool isPenalized;  // 연체로 인한 패널티 여부
    system_clock::time_point penaltyEnd;  // 패널티 종료 시간


    // 생성자: 대여자 정보와 대여 시간, 반납 예정 시간을 초기화한다.
    Renter(string dept, string id, string name, string item, system_clock::time_point rented, system_clock::time_point due)
        : department(dept), studentID(id), studentName(name), itemName(item), rentedTime(rented), dueTime(due) {}
};


vector<Item> items;  // 모든 물품을 저장하는 벡터
vector<Renter> renters;  // 모든 대여자를 저장하는 벡터
vector<Renter> penalizedRenters;  // 연체자를 저장하는 벡터
```

- **입출력**:

1. name(Item): 물품 이름
2. totalQuantity(Item): 물품의 총 수량
3. availableQuantity(Item): 현재 대여 가능한 물품 수량
4. department(Renter): 대여자의 학과
5. studentID(Renter): 대여자의 학번
6. studentName(Renter): 대여자의 이름
7. itemName(Renter): 대여한 물품의 이름
8. rentedTime(Renter): 대여한 시간
9. dueTime(Renter): 반납 예정 시간
10. isPenalized: 연체로 인한 패널티 여부
11. penaltyEnd: 패널티 종료 시간

- **설명**: 물품 정보를 저장하는 'Item' 클래스와 물품 대여자의 정보를 저장하는 'Renter' 클래스를 선언하고 정보를 관리하기 위한 변수 및 함수를 선언합니다.

- **적용된 배운 내용**:

1. **클래스**:
   - Item과 Renter는 클래스를 활용하여 데이터와 관련 기능을 하나의 단위로 묶어 표현합니다.
   - Item 클래스는 물품에 대한 정보를, Renter 클래스는 물품을 대여한 사람의 정보를 저장합니다.
2. **멤버 변수**:
   - Item 클래스는 name, totalQuantity, availableQuantity와 같은 멤버 변수를 가지고 있고, 이는 각 물품의 이름, 총 수량, 사용 가능한 수량을 표현합니다.

- Renter 클래스는 department, studentID, studentName, itemName, rentedTime, dueTime, isPenalized, penaltyEnd를 멤버 변수로 가지고 있어, 대여자의 정보 및 대여된 물품과 대여 시간, 반납 예정 시간을 표현 및 연체 패널티를 관리합니다.

3. **생성자**:

   - 두 클래스 모두 생성자를 정의하여 객체가 생성될 때 멤버 변수들을 초기화합니다.

   - 예를 들어, Item 클래스의 생성자는 물품의 이름과 수량을 인자로 받거나 물품의 이름과 총 수량, 남은 수량을 인자로 받아 객체를 초기화합니다.

4. **벡터**:

   - vector<Item>과 vector<Renter>는 표준 라이브러리의 일부인 벡터를 사용합니다.

   - 이 벡터들은 Item 객체와 Renter 객체를 동적으로 저장하는데 사용되며, 물품 목록과 대여자 목록을 관리하는 데 사용됩니다.

5. **기타 C++ 표준 라이브러리 기능**:

   - string은 문자열을 저장하고 처리하는 데 사용됩니다.

   - 배운 내용은 아니지만, C++ 표준 라이브러리 중 chrono 라이브러리는 시간 관련 기능을 제공하여, 대여 시간과 반납 시간을 처리하는 데 사용됩니다.

(2) 관리자 인증 기능

```
// 관리자 인증 함수
bool authenticateAdmin() {
    string password;
    cout << "관리자 비밀번호를 입력하세요: ";
    cin >> password;

    const string adminPassword = "admin123";  // 정적 비밀번호 생성
    return password == adminPassword;
}
```

- **입출력**:

1. password: 입력된 비밀번호
2. adminPassword: 관리자 비밀번호

- **설명**: 이 기능은 사용자가 프로그램의 관리자로 인증하기 위해 비밀번호를 입력하는 과정을 구현합니다. 사용자가 올바른 비밀번호를 입력하면 시스템에 접근할 수 있는 권한을 얻게 됩니다.

- **적용된 배운 내용**:

1. **함수**:
    - authenticateAdmin은 관리자 인증을 처리하는 독립적인 기능을 수행하는 별도의 함수입니다.
2. **입출력**:
    - cin을 사용해 사용자로부터 비밀번호를 입력 받습니다. cout을 사용해 인증 성공 또는 실패 메시지를 출력합니다.

(3) 물품 추가 기능

```
// 물품을 추가하는 함수
void addItem() {
    string name;
    int quantity;

    cout << "₩n[물품 추가]₩n";
    cout << "물품 이름: ";
    cin >> name;

    // 같은 이름의 물품이 있는지 검사
    for (const auto& item : items) {
        if (item.name == name) {
            cout << "오류: 이미 같은 이름의 물품이 있습니다." << endl;
            return; // 함수 종료
        }
    }

    cout << "물품 수량: ";
    cin >> quantity;

    // 수량이 0인지 검사
    if (quantity <= 0) {
        cout << "오류: 물품 수량은 0보다 커야 합니다." << endl;
        return; // 함수 종료
    }

    // 같은 이름의 물품이 없고 수량이 0보다 크면 추가
    items.push_back(Item(name, quantity));
    cout << "물품이 추가되었습니다." << endl;
}
```

- **입출력**:

1. name: 물품 이름
2. quantity: 물품의 수량

- **설명**: 이 기능은 관리자가 새로운 물품을 시스템에 추가할 수 있도록 합니다. 사용자는 물품의 이름과 수량을 입력하며, 이 정보는 items 벡터에 저장됩니다. 물품 이름이 중복되거나 수량이 0 이하인 경우에는 오류 메시지를 출력하고 추가를 거부합니다.

- **적용된 배운 내용**:

1. **함수**:
    - addItem은 물품을 추가하는 별도의 함수로 구현되어 있으며, 사용자의 입력을 처리합니다.
2. **조건문**:
    - 입력 받은 물품의 이름이 기존에 items 벡터에 존재하는지 확인합니다. 입력 받은 수량이 0보다 큰지 검사합니다.

3. **반복문**:
   - for 반복문을 사용하여 items 벡터를 순회하며 중복된 물품 이름을 검사합니다.
4. **벡터**:
   - items는 vector<Item> 타입으로, 시스템에 등록된 물품들을 동적으로 관리합니다.
5. **입출력**:
   - cin을 사용하여 물품의 이름과 수량을 입력 받습니다. cout을 사용하여 물품 추가 결과를 사용자에게 알립니다.

(4) 물품 수정 기능

```
// 물품 수량을 수정하는 함수
void modifyItem() {
    string name;
    int newQuantity;

    cout << "₩n[물품 수정]₩n";
    // 현재 물품 리스트 출력
    viewItems();
    cout << endl;
    cout << "수정할 물품 이름: ";
    cin >> name;

    for (auto& item : items) {
        if (item.name == name) {
            cout << "현재 총 수량: " << item.totalQuantity << ", 현재 남은 수량: " << item.availableQuantity << endl;
            cout << "새로운 총 수량 (대여 중인 수량을 포함): ";
            cin >> newQuantity;

            if (newQuantity < item.totalQuantity - item.availableQuantity) {
                cout << "오류: 새로운 총 수량은 현재 대여 중인 수량보다 작을 수 없습니다." << endl;
                return;
            }

            // 남은 수량을 조정한다.
            item.availableQuantity = newQuantity - (item.totalQuantity - item.availableQuantity);
            item.totalQuantity = newQuantity;

            cout << "물품 정보가 수정되었습니다." << endl;
            return;
        }
    }

    cout << "해당 이름의 물품을 찾을 수 없습니다." << endl;
}
```

- **입출력**:

1. name: 물품 이름
2. newQuantity: 새로운 수량

- **설명**: 이 기능은 사용자가 시스템에 등록된 물품의 수량을 수정할 수 있게 합니다. 사용자는 수정할 물품의 이름과 새로운 수량을 입력합니다. 새로운 수량이 현재 대여 중인 수량보다 작으면 오류 메시지를 출력하고 수정을 거부합니다.

- **적용된 배운 내용**:

1. **함수**:
   - modifyItem은 물품의 수량을 수정하는 별도의 함수로 구현되어 있습니다.
2. **입출력**:
   - cin과 cout을 사용하여 사용자로부터 수정할 물품 이름을 입력 받고 처리 결과를 출력합니다.
3. **반복문**:
   - for 반복문을 통해 items 벡터 내의 물품들을 순회하며 입력된 이름과 일치하는 물품을 찾습니다.
4. **조건문**:
   - if 조건문으로 새로운 수량이 유효한지 확인합니다.
5. **벡터**:
   - items는 vector<Item> 타입으로, 시스템에 등록된 물품들을 관리합니다.

(5) 물품 삭제 기능

```
// 물품을 삭제하는 함수
void deleteItem() {
    string name;
    cout << "₩n[물품 삭제]₩n";
    // 현재 물품 리스트 출력
    viewItems();
    cout << endl;
    cout << "삭제할 물품 이름: ";
    cin >> name;

    for (auto it = items.begin(); it != items.end(); ++it) {
        if (it->name == name) {
            items.erase(it);
            cout << "물품이 삭제되었습니다." << endl;
            return;
        }
    }

    cout << "해당 이름의 물품을 찾을 수 없습니다." << endl;
}
```

- **입출력**:

1. name: 물품 이름

- **설명**: 이 기능은 사용자가 시스템에 등록된 물품을 삭제할 수 있게 합니다. 사용자는 삭제할 물품의 이름을 입력합니다. 삭제할 물품이 리스트에 있다면 물품을 삭제하고 찾을 수 없다면 삭제를 하지 않습니다.

- **적용된 배운 내용**:

1. **함수**:
   - deleteItem은 물품을 삭제하는 별도의 함수로 구현되어 있습니다.
2. **입출력**:
   - cin과 cout을 사용하여 사용자로부터 삭제할 물품 이름을 입력 받고 처리 결과를 출력합니다.
3. **반복문**:
   - for 반복문을 통해 items 벡터 내의 물품들을 순회하며 입력된 이름과 일치하는 물품을 찾습니다.
4. **조건문**:
   - if 조건문으로 입력된 물품이 존재하는지 확인합니다.
5. **벡터**:
   - items는 vector<Item> 타입으로, 시스템에 등록된 물품들을 관리합니다.

(6) 물품 관리 함수

```
// 물품을 관리(추가, 수정, 삭제)하는 함수
void manageItems() {
    int choice;
    while (true) {
        cout << "₩n[물품 관리]" << endl;
        cout << "1. 물품 추가" << endl;
        cout << "2. 물품 수정" << endl;
        cout << "3. 물품 삭제" << endl;
        cout << "0. 메인 메뉴로 돌아가기" << endl;
        cout << "선택: ";
        cin >> choice;

        switch (choice) {
        case 1: addItem(); break;
        case 2: modifyItem(); break;
        case 3: deleteItem(); break;
        case 0: return;
        default: cout << "잘못된 선택입니다." << endl;
        }
    }
}
```

- **입출력**:

1. choice: 사용자의 기능 선택

- **설명**: 이 기능은 사용자가 사용하고 싶은 기능의 번호를 입력 받습니다. 물품 추가, 수정, 삭제 중 하나의 기능을 입력 받거나 메인 메뉴로 돌아갑니다.

- **적용된 배운 내용**:

1. **함수**:
    - manageItems는 물품의 수량을 수정하는 별도의 함수로 구현되어 있습니다.
2. **입출력**:
    - cin과 cout을 사용하여 사용자의 선택을 입력 받고 처리 결과를 출력합니다.
3. **반복문**:
    - while 반복문을 통해 특정 조건을 만족할 때까지 반복문을 반복합니다.
4. **조건문**:
    - switch 조건문으로 사용자의 기능 선택을 처리합니다.

(7) 물품 대여 기능

```
// 물품을 대여하는 함수
void rentItem() {
    string department, studentID, studentName, itemName;

    cout << "₩n[물품 대여]₩n";
    cout << "학과: ";
    cin >> department;
    cout << "학번: ";
    cin >> studentID;
    cout << "이름: ";
    cin >> studentName;
    // 대여자의 편의를 위한 현재 물품 리스트 출력
    viewItems();
    cout << endl;
    cout << "대여하려는 물품 이름: ";
    cin >> itemName;

    bool itemFound = false;
    for (auto& item : items) {
        if (item.name == itemName) {
            itemFound = true;
            if (item.availableQuantity > 0) {
                item.availableQuantity--;

                auto now = system_clock::now();
                auto due = now + hours(48); // 반납 예정 시간을 48시간 후로 설정
                // auto due = now + seconds(30); // 반납 예정 시간 작동 테스트용 코드(30초로 설정)

                // 시간 출력을 위한 코드
                time_t dueTime_t = system_clock::to_time_t(due);
                struct tm dueTm;
                localtime_s(&dueTm, &dueTime_t);

                // 대여자가 연체 패널티를 받은 경우 대여 해주지 않음
                for (auto& renter : penalizedRenters) {
                    // 시간 출력을 위한 코드
                    time_t penaltyEndTime_t = system_clock::to_time_t(renter.penaltyEnd);
                    struct tm penaltyEndTm;
                    localtime_s(&penaltyEndTm, &penaltyEndTime_t);

                    if (renter.studentID == studentID && now < renter.penaltyEnd) {
                        cout << "연체로 인해 대여가 금지되었습니다." << endl;
                        cout << "대여 가능 시간: " << put_time(&penaltyEndTm, "%Y-%m-%d %H:%M:%S") << endl;
                        return;
                    }
                }

                renters.push_back(Renter(department, studentID, studentName, itemName, now, due));
                cout << "대여 완료! 반납 예정 시간: " << put_time(&dueTm, "%Y-%m-%d %H:%M:%S") << endl;
                return;
            }
            else {
                cout << "죄송합니다. 현재 해당 물품은 대여할 수 없습니다. (재고 부족)" << endl;
                return;
            }
        }
    }

    if (!itemFound) {
        cout << "해당 물품을 찾을 수 없습니다." << endl;
    }
}
```

- **입출력**:

1. department: 대여자의 학과
2. studentID: 대여자의 학번
3. studentName: 대여자의 이름

4. itemName: 대여한 물품의 이름
5. itemFound: 아이템 발견 여부
6. now: 현재 시각
7. due: 반납 예정 시간

- **설명**: 이 기능은 사용자가 시스템에 등록된 물품을 대여할 수 있게 합니다. 사용자는 자신의 학과, 학번, 이름 및 대여할 물품의 이름을 입력합니다. 시스템은 입력된 정보를 기반으로 물품의 대여 가능 여부를 확인하고 처리합니다. 연체 패널티가 있는 사용자는 대여를 거부합니다.

- **적용된 배운 내용**:

1. **함수**:
   - rentItem은 물품 대여 처리를 담당하는 별도의 함수로 구현됩니다.
2. **입출력**:
   - cin과 cout을 사용하여 사용자로부터 대여하려는 물품 이름을 입력 받고 연체 여부를 확인 후 대여 결과를 출력합니다.
3. **반복문**:
   - for 반복문을 사용하여 items 벡터 내의 물품을 찾고 penalizedRenters 벡터를 순회합니다.
4. **조건문:**
   - if 조건문으로 대여 가능 여부와 연체 패널티 여부를 검사합니다.
5. **벡터**:
   - items와 renters는 각각 물품과 대여자 정보를 저장하는 벡터입니다.

(8) 물품 반납 기능

```
// 물품을 반납하는 함수
void returnItem() {
    string studentID, itemName;

    cout << "₩n[물품 반납]₩n";
    cout << "학번: ";
    cin >> studentID;
    cout << "반납하려는 물품 이름: ";
    cin >> itemName;

    auto now = system_clock::now(); // 반납 시각 저장

    // 대여자 목록에서 해당 대여자 찾기
    for (size_t i = 0; i < renters.size(); ++i) {
        if (renters[i].studentID == studentID && renters[i].itemName == itemName) {
            if (now > renters[i].dueTime) {
                cout << "물품 반납이 연체되어 7일간 대여가 금지됩니다." << endl;
                renters[i].isPenalized = true;
                renters[i].penaltyEnd = now + hours(168);  // 현재로부터 7일 후

                // 시간 출력을 위한 코드
                time_t penaltyEndTime_t = system_clock::to_time_t(renters[i].penaltyEnd);
                struct tm penaltyEndTm;
                localtime_s(&penaltyEndTm, &penaltyEndTime_t);

                cout << "대여 가능 시간: " << put_time(&penaltyEndTm, "%Y-%m-%d %H:%M:%S") << endl;
                penalizedRenters.push_back(renters[i]);  // 연체자 목록에 추가
            }

            for (auto& item : items) {
                if (item.name == itemName) {
                    item.availableQuantity++;
                    cout << "반납 완료!" << endl;
                }
            }

            renters.erase(renters.begin() + i);  // 대여자 목록에서 제거
            return;
        }
    }
    cout << "반납할 물품을 찾을 수 없습니다." << endl;
}
```

- **입출력**:

1. studentID: 반납자의 학번
2. itemName: 반납하려는 물품의 이름
3. now: 현재 시각
4. dueTime: 반납 예정 시간

- **설명**: 이 기능은 사용자가 대여한 물품을 시스템에 반납할 수 있게 하는 기능입니다. 사용자는 반납을 위해 학번, 반납하려는 물품 이름을 입력하고, 대여자 목록에서 해당 물품을 대여한 사용자를 찾았다면, 해당 물품의 수량을 원복시키고, 반납 완료 메세지를 출력하며 사용자를 대여자 목록에서 제거합니다. 정보가 일치하지 않다면 반납할 물품을 찾을 수 없음을 알립니다. 또한 연체되어 반납을 하였다면 7일간 대여를 금지시킵니다.

- **적용된 배운 내용**:

1. **함수**:
   - returnItem은 물품 반납 처리를 담당하는 별도의 함수로 구현되어 있습니다. 이 함수는 사용자로부터 반납할 물품의 이름과 대여자 정보를 입력받아 처리합니다.
2. **입출력**:
   - cin과 cout을 사용하여 사용자로부터 학번, 반납하려는 물품의 이름을 입력받고 처리 결과를 화면에 출력합니다.
3. **반복문**:
   - for 반복문은 renters 벡터를 순회하며 해당 학번을 가진 대여자와 일치하는 물품을 찾습니다.
4. **조건문**:
   - if 조건문은 대여자의 대여 정보가 입력된 정보와 일치하는지 검사합니다.
   - 연체자라면 반납을 금지시키는 패널티를 부여합니다.
5. **벡터**:
   - renters 벡터는 현재 대여 중인 모든 물품과 대여자의 정보를 저장합니다. 이 벡터에서 해당 대여자를 찾아 정보를 업데이트하거나 제거합니다.
   - items 벡터는 등록된 모든 물품의 정보를 저장하며, 반납된 물품의 수량을 업데이트합니다.
   - 반납처리가 완료되면, renters 벡터에서 해당 대여자 정보를 erase 메서드를 사용하여 제거합니다.
   - 연체자라면 연체자 목록에 추가합니다.
6. **클래스의 객체 접근**:
   - Item 클래스와 Renter 클래스의 객체에 접근하여 물품의 수량을 수정하거나 대여자 정보를 업데이트합니다.

(9) 모든 물품의 목록을 보여주는 함수

```
// 모든 물품의 목록을 보여주는 함수
void viewItems() {
    cout << "₩n[모든 물품 리스트]₩n";
    for (const auto& item : items) {
        cout << item.name << " - 총 수량: " << item.totalQuantity << ", 남은 수량: " << item.availableQuantity
            << ", 대여 중: " << item.totalQuantity - item.availableQuantity << endl;
    }
}
```

- **입출력**:
  - 없음 (함수 내부에서 items 벡터의 데이터를 직접 사용)
- **설명**: 이 함수는 시스템에 등록된 모든 물품의 목록을 보여줍니다. items 벡터의 각 요소를 순회하며 물품의 이름, 총 수량, 남은 수량, 대여 중인 수량을 출력합니다.
- **적용된 배운 내용**:

1. **반복문**:
   - 이 함수에서는 for 반복문을 사용하여 items 벡터에 저장된 모든 Item 객체를 순회합니다. 이를 통해 저장된 모든 물품의 정보를 순차적으로 접근하고 출력합니다.
2. **벡터**:
   - items는 Item 객체들을 저장하는 vector<Item> 타입의 컨테이너입니다. 벡터를 사용함으로써 동적 크기 조정 및 인덱스 기반 접근이 가능해집니다.
3. **입출력**:
   - cout을 사용하여 사용자에게 총 수량, 남은 수량, 대여 중인 수량에 대한 정보를 제공합니다.
4. **클래스와 객체**:

- Item 클래스의 각 인스턴스는 물품의 이름, 총 수량, 사용 가능한 수량을 나타냅니다. 클래스의 각 객체는 이러한 데이터를 캡슐화하고 관리합니다.

(10) 대여 정보를 보여주는 함수

```
// 현재 대여 중인 물품과 대여자 정보를 보여주는 함수
void viewRenters() {
    cout << "₩n[대여 중인 물품 리스트]₩n";
    for (const auto& renter : renters) {
        time_t rentedTime_t = system_clock::to_time_t(renter.rentedTime);
        struct tm rentedTm;
        localtime_s(&rentedTm, &rentedTime_t);

        time_t dueTime_t = system_clock::to_time_t(renter.dueTime);
        struct tm dueTm;
        localtime_s(&dueTm, &dueTime_t);

        cout << "학과: " << renter.department
            << ", 학번: " << renter.studentID
            << ", 이름: " << renter.studentName
            << ", 대여 물품: " << renter.itemName << endl;
        cout << "대여 시간: " << put_time(&rentedTm, "%Y-%m-%d %H:%M:%S")
            << ", 반납 예정 시간: " << put_time(&dueTm, "%Y-%m-%d %H:%M:%S") << endl;
    }
}
```

- **입출력**:
  - 없음 (함수 내부에서 items 벡터의 데이터를 직접 사용)
- **설명**: 이 함수는 현재 대여 중인 모든 물품과 대여자의 정보를 보여줍니다. 대여자의 학과, 학번, 이름, 대여한 물품 이름, 대여 시간, 반납 예정 시간 등을 출력합니다.
- **적용된 배운 내용**:

1. **반복문**:
   - 함수는 for 반복문을 사용하여 renters 벡터에 저장된 모든 Renter 객체를 순회합니다. 이를 통해 현재 대여 중인 물품과 대여자 정보를 순차적으로 처리합니다.
2. **벡터**:
   - renters는 Renter 객체들을 저장하는 vector<Renter> 타입의 컨테이너입니다. 벡터의 유연성 덕분에 대여자 목록을 효과적으로 관리할 수 있습니다.
3. **입출력**:
   - cout을 사용하여 대여 정보를 사용자에게 제공합니다. 표준 출력 스트림을 통해 텍스트 정보를 콘솔에 출력합니다.
4. **클래스와 객체**:

   - Renter 클래스의 각 인스턴스는 대여자의 정보와 대여한 물품, 대여 시간 및 반납 예정 시간을 나타냅니다. 객체 지향 프로그래밍에서 클래스는 이러한 데이터와 기능을 캡슐화하는 데 사용됩니다.

(11) 물품 리스트를 파일에 저장하는 함수

```
// 물품 리스트를 파일에 저장하는 함수
void saveItemsToFile() {
    ofstream outFile("items.txt");
    for (const auto& item : items) {
        outFile << item.name << "," << item.totalQuantity << "," << item.availableQuantity << "₩n";
    }
    outFile.close();
    cout << "물품 리스트가 저장되었습니다." << endl;
}
```

- **입출력**:

1. outFile: 파일 출력 스트림, items.txt 파일에 데이터를 작성
2. name: 물품의 이름
3. totalQuantity: 물품의 총 수량
4. availableQuantity: 사용 가능한 물품 수량

- **설명**: 이 함수는 시스템에 저장된 모든 물품 정보를 items.txt 파일에 저장합니다. 파일 출력 스트림 ofstream을 이용하여 파일에 접근하고, items 벡터에 저장된 각 Item 객체의 데이터를 파일에 작성합니다.

- **적용된 배운 내용**:

1. **파일 입출력**:
   - ofstream을 사용하여 파일에 데이터를 씁니다. items.txt 파일에 물품 리스트를 저장하기 위해 사용됩니다.
2. **반복문**:
   - items 벡터에 저장된 모든 Item 객체를 순회하여, 각 객체의 정보를 파일에 기록합니다.
3. **벡터**:
   - items 벡터는 저장된 Item 객체들의 집합으로, 파일에 저장될 데이터를 제공합니다.
4. **문자열 처리**:
   - 각 Item 객체의 필드(물품 이름, 수량 등)를 문자열로 변환하여 파일에 쓰는 과정을 포함합니다.

(12) 대여자 리스트를 파일에 저장하는 함수

```
// 대여자 리스트를 파일에 저장하는 함수
void saveRentersToFile() {
    ofstream outFile("renters.txt");
    for (const auto& renter : renters) {
        outFile << renter.department << "," << renter.studentID << "," << renter.studentName << ","
            // 시간 데이터를 저장하기 위한 코드
            << renter.itemName << "," << duration_cast<seconds>(renter.rentedTime.time_since_epoch()).count() << ","
            << duration_cast<seconds>(renter.dueTime.time_since_epoch()).count() << "\n";
    }
    outFile.close();
    cout << "대여자 리스트가 저장되었습니다." << endl;
}
```

- **입출력**:

1. outFile: 파일 출력 스트림, renters.txt 파일에 데이터를 작성
2. renters 벡터의 각 Renter 객체의 필드:
3. department: 대여자의 학과
4. studentID: 대여자의 학번
5. studentName: 대여자의 이름
6. itemName: 대여한 물품 이름
7. rentedTime: 대여 시간
8. dueTime: 반납 예정 시간

- **설명**: 이 함수는 시스템에 저장된 모든 대여자 정보를 renters.txt 파일에 저장합니다. renters 벡터에 저장된 각 Renter 객체의 데이터를 파일 출력 스트림 ofstream을 통해 파일에 작성합니다.

- **적용된 배운 내용**:

1. **파일 입출력**:
   - ofstream 객체를 사용하여 renters.txt 파일에 대여자 정보를 기록합니다. 이를 통해 데이터를 영구적으로 저장하여 관리할 수 있습니다.
2. **반복문**:
   - renters 벡터의 각 Renter 객체를 순회하여, 그 정보를 파일에 저장합니다.
3. **벡터**:

- renters 벡터는 Renter 객체들의 집합으로, 파일에 저장될 데이터를 제공합니다.

(13) 물품 리스트를 파일에서 불러오는 함수

```
// 물품 리스트를 파일에서 불러오는 함수
void loadItemsFromFile() {
    ifstream inFile("items.txt");
    if (!inFile.is_open()) {
        cerr << "파일을 열 수 없습니다." << endl;
        return;
    }

    string line;
    while (getline(inFile, line)) {
        stringstream ss(line);
        string name, totalStr, availableStr;
        getline(ss, name, ',');
        getline(ss, totalStr, ',');
        getline(ss, availableStr);
        int total = stoi(totalStr);
        int available = stoi(availableStr);
        items.push_back(Item(name, total, available));
    }

    inFile.close();
}
```

- **입출력**:

1. inFile: 파일 입력 스트림, items.txt 파일로부터 데이터를 읽음

- **설명**: 이 함수는 items.txt 파일로부터 물품 리스트를 읽어와 items 벡터에 저장합니다. 파일 입력 스트림 ifstream을 이용하여 파일로부터 물품의 이름, 총 수량, 사용 가능한 수량을 읽어오고, 이를 Item 객체로 변환하여 items 벡터에 추가합니다.

- **적용된 배운 내용**:

1. **파일 입출력**:

   - ifstream을 사용하여 파일로부터 데이터를 읽어옵니다. items.txt 파일에서 물품 리스트를 불러오는 데 사용됩니다.

2. **반복문**:

   - 파일의 각 줄을 순회하며 물품 정보를 읽고 처리합니다.

3. **벡터**:

- 파일로부터 읽은 데이터는 items 벡터에 저장되며, 이를 통해 프로그램에서 쉽게 접근하고 사용할 수 있습니다.

(14) 대여자 리스트를 파일에서 불러오는 함수

```
// 대여자 리스트를 파일에서 불러오는 함수
void loadRentersFromFile() {
    ifstream inFile("renters.txt");
    if (!inFile.is_open()) {
        cerr << "파일을 열 수 없습니다." << endl;
        return;
    }

    string line;
    while (getline(inFile, line)) {
        stringstream ss(line);
        string dept, id, name, itemName, rentedSecStr, dueSecStr;
        getline(ss, dept, ',');
        getline(ss, id, ',');
        getline(ss, name, ',');
        getline(ss, itemName, ',');
        getline(ss, rentedSecStr, ',');
        getline(ss, dueSecStr);
        long long rentedSec = stoll(rentedSecStr);
        long long dueSec = stoll(dueSecStr);
        time_point<system_clock> rented = system_clock::from_time_t(rentedSec);
        time_point<system_clock> due = system_clock::from_time_t(dueSec);
        renters.push_back(Renter(dept, id, name, itemName, rented, due));
    }

    inFile.close();
}
```

- **입출력**:

1. inFile: 파일 입력 스트림, renters.txt 파일로부터 데이터를 읽음

- **설명**: 이 함수는 renters.txt 파일로부터 대여자 정보를 읽어와 renters 벡터에 저장합니다. 파일 입력 스트림 ifstream을 이용하여 파일로부터 대여자의 학과, 학번, 이름, 대여한 물품 이름, 대여 시간, 반납 예정 시간 등을 읽어오고, 이를 Renter 객체로 변환하여 renters 벡터에 추가합니다.

- **적용된 배운 내용**:

1. **파일 입출력**:
    - ifstream 객체를 사용하여 renters.txt 파일로부터 대여자 정보를 읽어옵니다. renters.txt 파일에서 대여자 리스트를 불러오는 데 사용됩니다.
    -
2. **반복문**:
    - 파일의 각 줄을 순회하며 대여자 정보를 읽고 처리합니다.
3. **벡터**:
    - 파일에서 읽은 대여자 데이터는 renters 벡터에 저장됩니다. 이를 통해 프로그램 내에서 대여자 정보에 쉽게 접근할 수 있습니다.

(15) 메인 함수

```
// 메인 함수: 사용자 인터페이스와 프로그램의 메인 루프를 담당
int main() {
    bool isAuthenticated = false;  // 관리자 인증 여부를 저장하는 플래그

    // 최초 접속 시 관리자 인증
    while (!isAuthenticated) {
        isAuthenticated = authenticateAdmin();
        if (!isAuthenticated) {
            cout << "인증 실패: 올바른 비밀번호를 입력하세요." << endl;
        }
        else {
            cout << "인증 완료. 시스템에 접속합니다." << endl;
            break;
        }
    }

    while (true) {
        cout << "₩n물품 관리 프로그램" << endl;
        cout << "1. 물품 관리" << endl;
        cout << "2. 물품 대여" << endl;
        cout << "3. 물품 반납" << endl;
        cout << "4. 모든 물품 리스트 보기" << endl;
        cout << "5. 대여 중인 물품 리스트 보기" << endl;
        cout << "6. 파일에 데이터 저장하기" << endl;
        cout << "7. 파일에서 데이터 불러오기" << endl;
        cout << "0. 종료" << endl;
        cout << "선택: ";

        int choice;
        cin >> choice;

        switch (choice) {
        case 1: manageItems(); break;
        case 2: rentItem(); break;
        case 3: returnItem(); break;
        case 4: viewItems(); break;
        case 5: viewRenters(); break;
        case 6:
            saveItemsToFile();
            saveRentersToFile();
            break;
        case 7:
            loadItemsFromFile();
            loadRentersFromFile();
            cout << "데이터를 불러왔습니다." << endl;
            break;
        case 0: // 데이터를 자동 저장하고 종료
            saveItemsToFile();
            saveRentersToFile();
            return 0;
        default: cout << "잘못된 선택입니다." << endl;
        }
    }

    return 0;
}
```

- **입출력**:

1. choice: 사용자가 선택할 기능의 번호

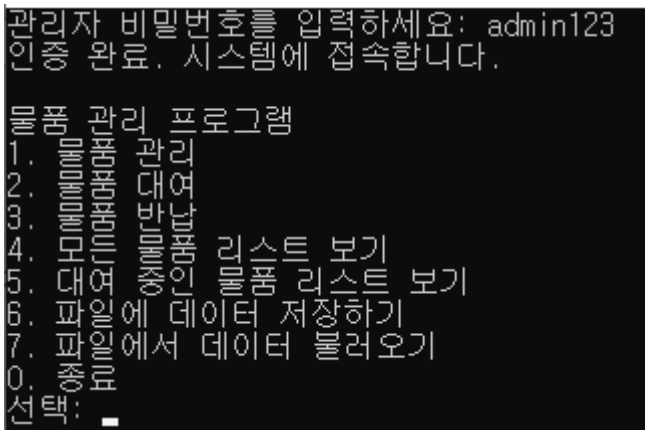
- **설명**: 메인 함수는 프로그램의 사용자 인터페이스 기능을 합니다. 사용자의 선택에 따라 앞서 정의한 기능별 함수들을 실행하도록 합니다.

- **적용된 배운 내용**:

1. **함수**:
   - 앞서 구현한 기능별 함수들을 사용자의 선택에 따라 실행합니다.
2. **반복문**:
   - while(true) 반복문을 사용하여 프로그램을 종료하는 명령이 올 때까지 무한 반복을 합니다.
3. **입출력**:
   - cout을 사용하여 사용자의 기능 선택을 유도하는 사용자 인터페이스를 출력합니다.
4. **조건문**:
   - switch 문을 이용하여 사용자가 원하는 기능을 숫자로 선택하도록 합니다.

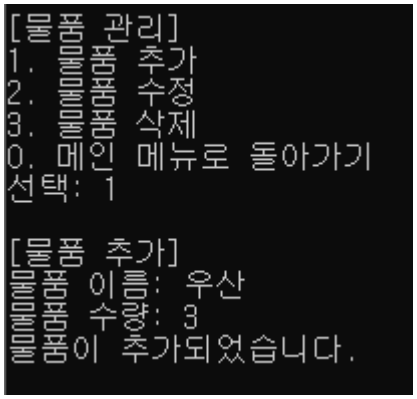
4. 테스트 결과

(1) 관리자 인증 기능



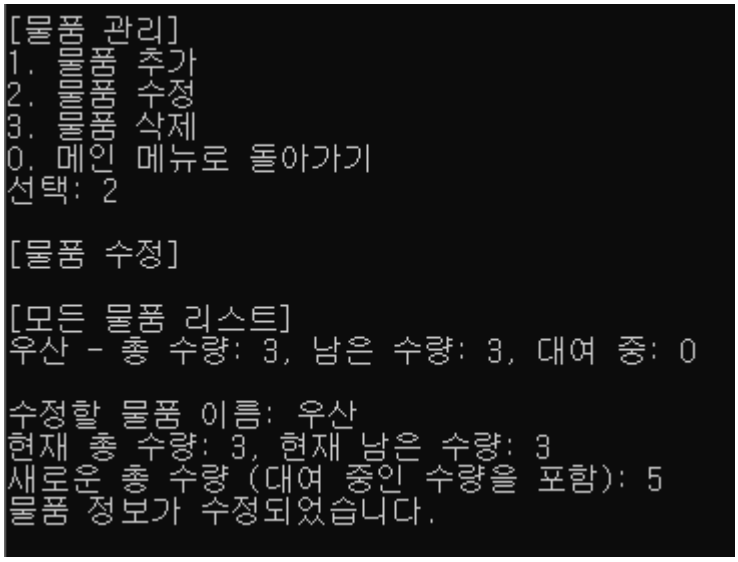
- **설명**: 이 기능은 사용자가 프로그램의 관리자로서 인증을 받기 위해 비밀번호를 입력하는 과정을 처리합니다. 올바른 비밀번호를 입력하면 사용자는 시스템에 접근할 수 있는 권한을 획득합니다.

(2) 물품 추가 기능



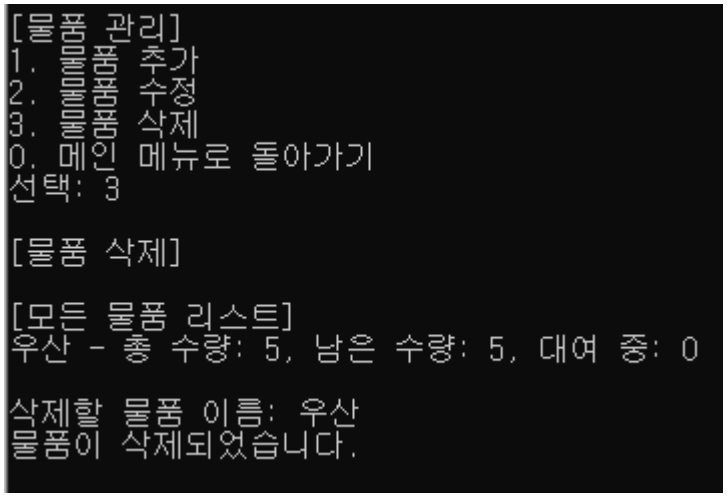
- **설명**: 관리자는 새로운 물품을 시스템에 추가할 수 있습니다. 사용자는 물품의 이름과 수량을 입력하고, 이 정보는 시스템의 items 벡터에 저장됩니다. 이미 존재하는 물품 이름이 입력되거나 수량이 0 이하인 경우, 오류 메시지가 표시되고 추가가 거부됩니다.

(3) 물품 수정 기능



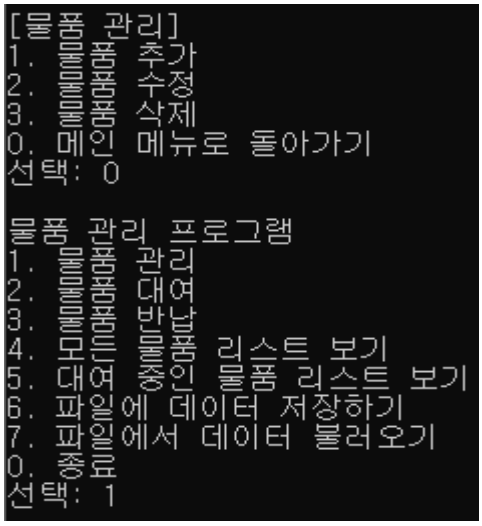
- **설명**: 이 기능을 통해 관리자는 기존에 등록된 물품의 수량을 수정할 수 있습니다. 수정할 물품의 이름과 새로운 수량을 입력받고, 새로운 수량이 현재 대여 중인 수량보다 적지 않은지 확인합니다. 유효한 변경이면 물품 정보가 업데이트됩니다.

(4) 물품 삭제 기능



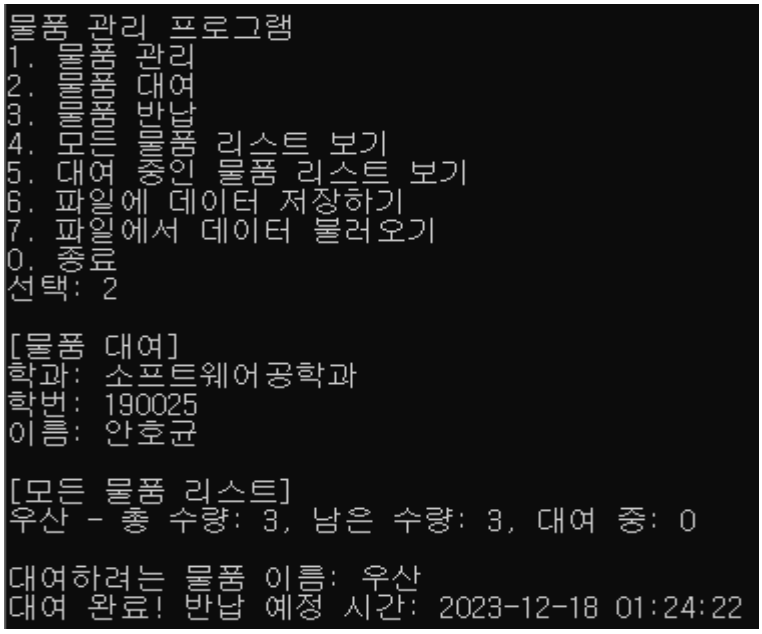
- **설명**: 관리자는 시스템에 등록된 물품을 삭제할 수 있습니다. 삭제할 물품의 이름을 입력받아 해당 물품이 목록에 존재하는 경우, 그 물품을 삭제합니다. 존재하지 않는 물품을 삭제하려고 할 때는 오류 메시지가 표시됩니다.

(5) 물품 관리 함수



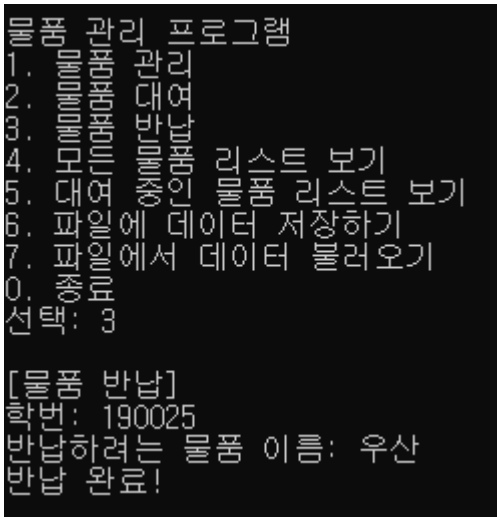
- **설명**: 이 함수는 물품 추가, 수정, 삭제와 같은 관리 작업을 선택할 수 있는 메뉴를 제공합니다. 사용자는 표시된 옵션 중에서 선택하여 해당 작업을 수행할 수 있습니다.

(6) 물품 대여 기능



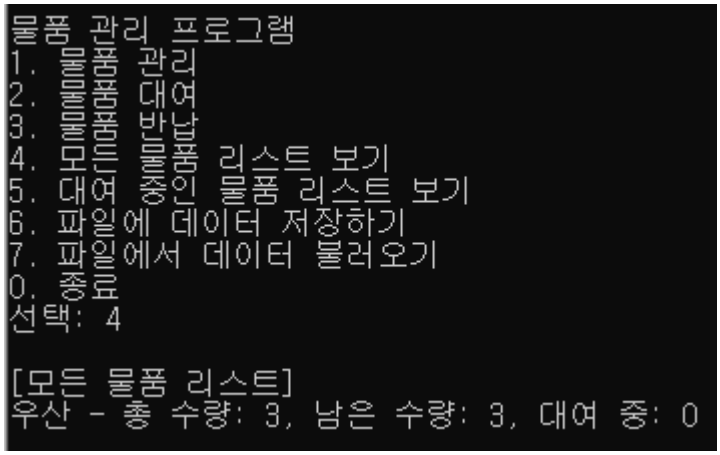
- **설명**: 사용자는 시스템에 등록된 물품을 대여할 수 있습니다. 이 기능에서는 대여자의 학과, 학번, 이름 및 대여할 물품의 이름을 입력받습니다. 물품이 사용 가능하고 대여자가 연체 패널티를 받지 않았다면 대여가 승인됩니다.

(7) 물품 반납 기능



- **설명**: 대여한 물품을 시스템에 반납하는 기능입니다. 사용자는 학번과 반납할 물품의 이름을 입력합니다. 반납 시 연체 여부를 확인하고, 연체된 경우 패널티가 부과됩니다.

(8) 모든 물품의 목록을 보여주는 함수



- **설명**: 이 함수는 시스템에 등록된 모든 물품의 목록을 출력합니다. 각 물품의 이름, 총 수량, 현재 사용 가능한 수량 등의 정보를 포함합니다.

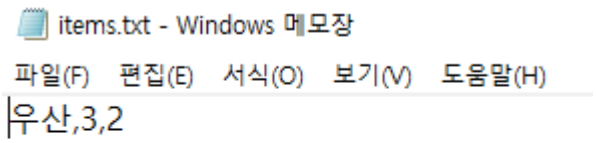
(9) 대여 정보를 보여주는 함수

```
물품 관리 프로그램
1. 물품 관리
2. 물품 대여
3. 물품 반납
4. 모든 물품 리스트 보기
5. 대여 중인 물품 리스트 보기
6. 파일에 데이터 저장하기
7. 파일에서 데이터 불러오기
0. 종료
선택: 5

[대여 중인 물품 리스트]
학과: 소프트웨어공학과, 학번: 190025, 이름: 안호균, 대여 물품: 우산
대여 시간: 2023-12-16 01:25:26, 반납 예정 시간: 2023-12-18 01:25:26
```

- **설명**: 현재 대여 중인 모든 물품과 대여자의 정보를 출력하는 기능입니다. 대여자의 학과, 학번, 이름, 대여한 물품, 대여 시간, 반납 예정 시간 등이 표시됩니다.

(10) 물품 리스트를 파일에 저장하는 함수

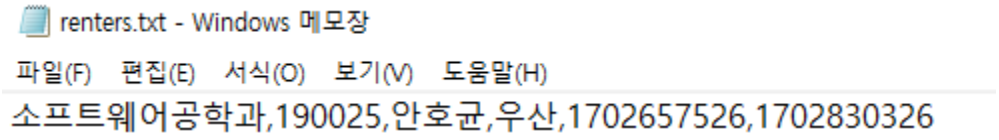
```
물품 관리 프로그램
1. 물품 관리
2. 물품 대여
3. 물품 반납
4. 모든 물품 리스트 보기
5. 대여 중인 물품 리스트 보기
6. 파일에 데이터 저장하기
7. 파일에서 데이터 불러오기
0. 종료
선택: 5

[대여 중인 물품 리스트]
학과: 소프트웨어공학과, 학번: 190025, 이름: 안호균, 대여 물품: 우산
대여 시간: 2023-12-16 01:25:26, 반납 예정 시간: 2023-12-18 01:25:26
```



- **설명**: 이 기능을 통해 현재 시스템에 등록된 모든 물품 정보를 items.txt 파일에 저장합니다. 물품의 이름, 총 수량, 현재 사용 가능한 수량이 파일에 기록됩니다.

(11) 대여자 리스트를 파일에 저장하는 함수

```
물품 관리 프로그램
1. 물품 관리
2. 물품 대여
3. 물품 반납
4. 모든 물품 리스트 보기
5. 대여 중인 물품 리스트 보기
6. 파일에 데이터 저장하기
7. 파일에서 데이터 불러오기
0. 종료
선택: 5

[대여 중인 물품 리스트]
학과: 소프트웨어공학과, 학번: 190025, 이름: 안호균, 대여 물품: 우산
대여 시간: 2023-12-16 01:25:26, 반납 예정 시간: 2023-12-18 01:25:26
```



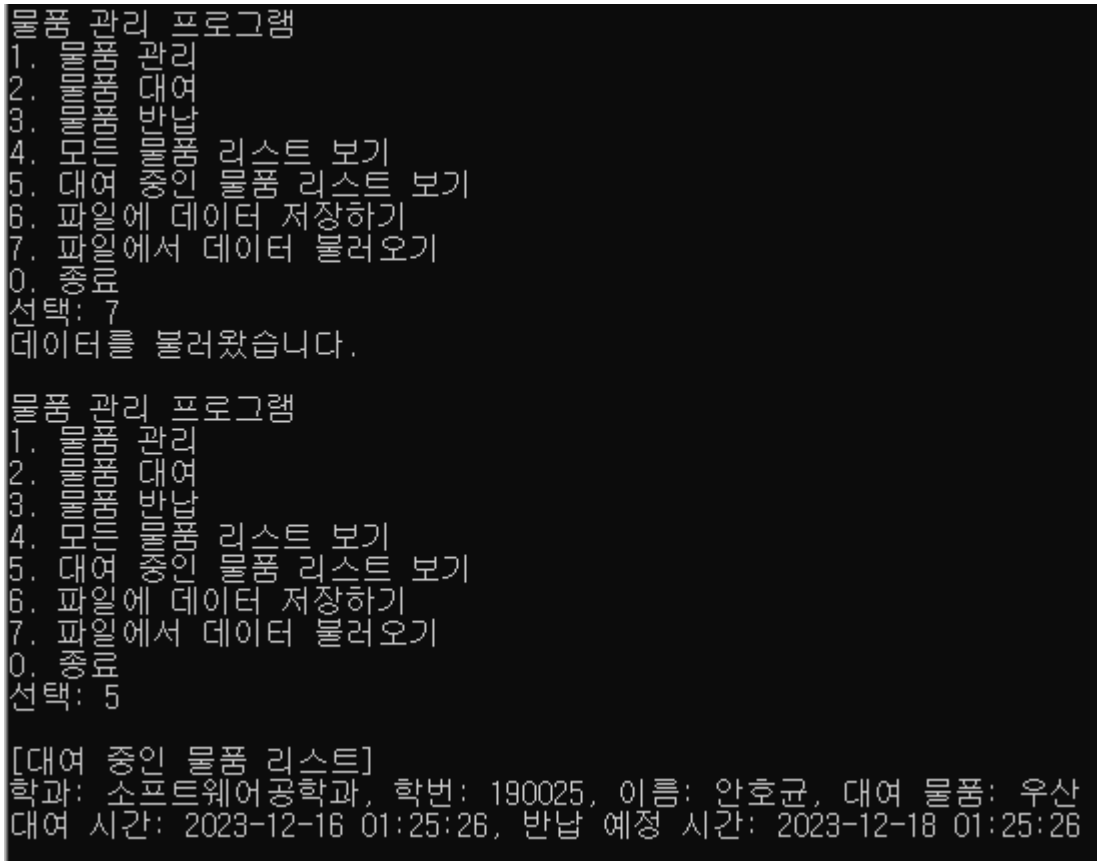
- **설명**: 시스템에 등록된 모든 대여자 정보를 renters.txt 파일에 저장합니다. 대여자의 학과, 학번, 이름, 대여한 물품, 대여 시간, 반납 예정 시간 등의 정보가 파일에 기록됩니다.

(12) 물품 리스트를 파일에서 불러오는 함수

```
물품 관리 프로그램
1. 물품 관리
2. 물품 대여
3. 물품 반납
4. 모든 물품 리스트 보기
5. 대여 중인 물품 리스트 보기
6. 파일에 데이터 저장하기
7. 파일에서 데이터 불러오기
0. 종료
선택: 7
데이터를 불러왔습니다.

물품 관리 프로그램
1. 물품 관리
2. 물품 대여
3. 물품 반납
4. 모든 물품 리스트 보기
5. 대여 중인 물품 리스트 보기
6. 파일에 데이터 저장하기
7. 파일에서 데이터 불러오기
0. 종료
선택: 4

[모든 물품 리스트]
우산 - 총 수량: 3, 남은 수량: 2, 대여 중: 1
```

- **설명**: 이 기능은 items.txt 파일에서 물품 리스트를 읽어와 시스템에 등록합니다. 파일에서 읽은 각 물품 정보는 Item 객체로 변환되어 items 벡터에 추가됩니다.

(13) 대여자 리스트를 파일에서 불러오는 함수



- **설명**: renters.txt 파일에서 대여자 정보를 읽어와 시스템에 등록하는 기능입니다. 파일에서 읽은 각 대여자 정보는 Renter 객체로 변환되어 renters 벡터에 추가됩니다.

5. 계획 대비 변경 사항

- 변경 사항 없이 계획대로 수행하였습니다.

6. 느낀점

수업에서 배운 내용을 기반으로 실제 문제 해결에 적용해본 이번 프로젝트 경험은, 저에게 매우 유의미한 학습 경험이 되었습니다. 이론적인 지식을 실제 겪고 있는 문제 해결에 적용해보는 일은 기존 다른 수업에서는 겪어보지 못했던 방식이기 때문입니다. 프로젝트에 착수하기 전, 저는 자신감과 함께 몇 가지 의문을 가지고 있었습니다. "과연 수업에서 배운 이론적 지식이 실제 상황에서 얼마나 유용하게 적용될 수 있을까?" 또한, "단순히 예제 코드를 통해 배운 내용과 이를 실제 응용 프로그램으로 구현하는 것 사이에는 어떤 차이가 있을까?"라는 생각이 들었습니다.

이 프로젝트를 진행하며, 저는 이러한 질문들에 대한 답을 찾을 수 있었습니다. 우선, 수업에서 배운 객체지향 프로그래밍은 프로젝트의 기초를 다지는 데 큰 도움이 되었습니다. 클래스와 객체를 사용하여 물품, 대여자, 관리 시스템 등을 표현하고, 이들 간의 상호작용을 구현하는 과정에서, 수업 시간에 배운 이론이 실제로 어떻게 적용되는지를 명확하게 이해할 수 있었습니다.

또한, 복잡한 문제를 해결하기 위해 필요한 알고리즘과 데이터 구조에 대한 이해가 심화되었습니다. 예를 들어, 대여 가능한 물품의 수량을 관리하고, 대여 및 반납 과정을 효율적으로 처리하기 위해 필요한 로직을 구현하는 과정에서, 수업에서 배운 배열, 리스트, 조건문, 반복문 등의 개념을 실제로 적용해보았습니다. 이러한 경험은 저에게 프로그래밍 능력을 단순히 '코드를 작성하는 것'에서 '문제를 해결하는 것'으로 확장하는 데 큰 도움이 되었습니다.

프로젝트를 통해 깨달은 다른 한 가지는, 실제 응용 프로그램 개발은 단순한 코드 작성을 넘어서는 많은 고려사항이 있다는 것입니다. 사용자 인터페이스 디자인, 사용자 경험, 데이터 보안 및 무결성 등의 추가적인 측면을 고려해야 했습니다. 이 과정에서 프로그래밍 기술 뿐만 아니라, 소프트웨어 개발의 다른 중요한 측면들에 대해서도 배울 수 있었습니다.

마지막으로, 이 프로젝트는 저의 프로그래밍에 대한 열정을 더욱 고취시켰습니다. 실제 문제를 해결할 수 있는 능력이 있다는 것을 깨달았을 때, 앞으로의 학습에 대한 강한 동기부여를 받을 수 있었습니다. 또한 프로젝트를 통해 배운 기술과 지식이 실질적인 가치를 창출할 수 있다는 것을 확인했으며, 앞으로도 이러한 기술을 활용하여 사회에 기여하고자 합니다.