C프로그래밍및실습

도서관 관리 시스템

진척 보고서 #2

제출일자: 2023-12-10
제출자명: 송재호
제출자학번: 185132

1. 프로젝트 목표

1) 배경 및 필요성

도서관은 도서의 정보를 확인, 대출, 등의 여러가지 기능을 함, 하지만 서적의 양이 많아짐에 따라, 사람이 직접 관리하는 것에 한계가 있음 이에 자동화 시스템이 필요하다고 생각한다. 고객이 원하는 분야의 서적들을 보여주고 각각의 서적을 선택한다면, 서적의 기본적인 정보들을 출력해준다면, 고객들은 직접 도서관에 방문하지 않아도 원하는 서적을 도서관에서 보유중인지 확인하거나, 원하는 분류만 선택을 한다면 서적의 목록을 보고 선택을 할 수 있을 것 이다.

2) 프로젝트 목표

많은 양의 서적들을 분야별로 분류하여 저장, 고객이 원하는 분야의 책 목록을 보여주고, 각 분야별에서 원하는 서적을 선택하면 서적의 정보를 출력, 고객이 서적을 대출한다면, 반납 예정일을 통보, 반납일의 일정 기간 전에 반납 요청 알림을 보낸다.

3) 차별점

기존 도서관 관리 시스템은 서적의 정보를 간략하게 보여주고 대출시 반납예정일만을 알려주었지만, 반납예정일의 일정 기간 전에 알림이 감.

2. 기능 계획

1) 분야 목록 출력 기능

- 설명: 원하는 분야를 선택하거나 입력한다면 같은 분야의 서적들의 목록을 출력하여 보여준다. 이러한 목록들은 아직 원하는 서적을 생각해 놓지 못한 고객들에게 서적 추천 기능의 역할을 할 수 있을 것 이다.

(1) 세부 기능

해당 기능을 사용하기 이전에 보유하고 있는 분류의 목록을 출력한다.

(2) 세부 기능

분야 선택 시 문자열 배열을 활용하여 분야별 도서 목록을 출력하고자 한다.

2) 대출, 반납 기능

- 설명: 원하는 도서가 있을 경우 대출 기능을 선택하여 대출을 하고 반납하는 기능이다.

(1) 세부 기능

대출 신청 시 해당 서적을 대출의 배열에 저장하는 기능

(2) 세부 기능

반납을 완료 할 경우 다시 서적을 원래 배열에 저장하는 기능

3) 관리자 암호 기능

- 설명: 도서 정보 입력에 있어서 접근 권한 제한이 없다면 잘못된 정보를 입력 받을 수 있다.

(1) 세부 기능

도서 정보 입력 전에 암호를 입력받아 암호가 일치 할 때 정보입력 기능을 하도록 하는 기능.

3. 진척사항

1) 기능 구현

(1) 도서 대출 기능

- 입출력

추가한 구조체: struct loan_book : 대출한 도서를 저장하는 구조체(도서명, 대출수량)

입력 :

struct library lib[]:구조체 library 의 배열 lib[] 로 도서의 정보를 저장

struct loan_book loanBook [] : 구조체 loan_book 의 배열 loanBook[] 로 대출 도서의 정보를 저장

char bk_nm[30] : 입력받은 도서명을 임시로 저장할 문자열

loanBook [*loanCount].book_name :구조체 배열 loanBook [] 중 도서의 이름을 저장하는 배열

loanBook [*loanCount].quantity :구조체 배열 loanBook [] 중 도서의 수량을 저장하는 배열

int loanCount : 대출 도서의 종류 수의 변수

int found : 입력받은 도서명이 저장되어있는 도서명인지 확인하기 위한 변수

출력 :void 함수이기에 반환값은 없으나 기존에 저장한 도서가 없을 때, 저장하지 않은 도서를 대대출하려 할 때, 대출을 완료했을때의 각각 안내 문구를 출력.

- 설명 : 대출하고자 하는 도서명을 입력받는다. 이때 이전에 입력받은 도서의 정보가 없다면, "현재 도서관에 저장되어 있는 도서가 없습니다."라는 안내 문구를 출력하고, 입력받은 도서명과 일치하는 도서명이 없다면, "도서명을 잘못 입력하셨습니다. 프로그램 처음으로 돌아갑니다."라는 안내문구를 출력한다.

대출하려는 도서명과 일치하는 도서명을 반복문과 strcmp 함수를 사용하여 찾아내고 일치하는 도서의 수량을 1줄이고 해당 도서의 이름을 loanBook [*loanCount].book_name 에 저장하고 . loanBook[*loanCount].quantity 의 값을 1추가 한다. 이때 loanBook [*loanCount].quantity 의 값을 이전에 초기화 시켜주어야한다.

- 적용된 배운 내용

반복문, 함수, 포인터, 배열, 구조체 조건문, scanf_s , printf 문, 함수

- 코드 스크린샷

```c
void borrowBook(struct library lib[], int count, struct loan_book loanBook[],
                int* loanCount) {
    char bk_nm[30];
    int found = 0;
    if (count == 0) {
        printf("현재 도서관에 저장되어 있는 도서가 없습니다.₩n");
        return;
    }// 도서관에 저장한 도서가 없을때

    printf("대출하려는 도서명을 입력하시오: ");
    scanf_s("%s", bk_nm, (int)sizeof(bk_nm));

    for (int i = 0; i < count; i++) {
        if (strcmp(bk_nm, lib[i].book_name) == 0) {
            if (lib[i].quantity == 0) {
                printf("해당 도서는 전권 대출중입니다. ₩n"); // 수량이 0일때
            } else {
                strcpy_s(loanBook[*loanCount].book_name, 20, lib[i].book_name);
                loanBook[*loanCount].quantity = 0;  // 대출 수량 초기화
                loanBook[*loanCount].quantity += 1;

                lib[i].quantity -= 1;

                (*loanCount)++;

                printf("대출 완료₩n");
            }
            found = 1;
            break;
        }
    }

    if (!found) {
        printf(
            "도서명을 잘못 입력하셨습니다.₩n"
            "프로그램 처음으로 돌아갑니다₩n₩n");
    }
}
```

(2) 대출 도서 출력 기능

- 입출력

입력:

struct loan_book loanBook[]: 구조체 loan_book 의 배열 loanBook [] 로 대출 도서의 정보를 저장

int loanCount : 대출 도서의 종류 수의 변수

loanBook [*loanCount].book_name :구조체 배열 loanBook [] 중 도서의 이름을 저장하는 배열

loanBook [*loanCount].quantity :구조체 배열 loanBook [] 중 도서의 수량을 저장하는 배열

출력: void 함수라 반환값은 없으나, 대출한 도서명의 목록을 출력한다.

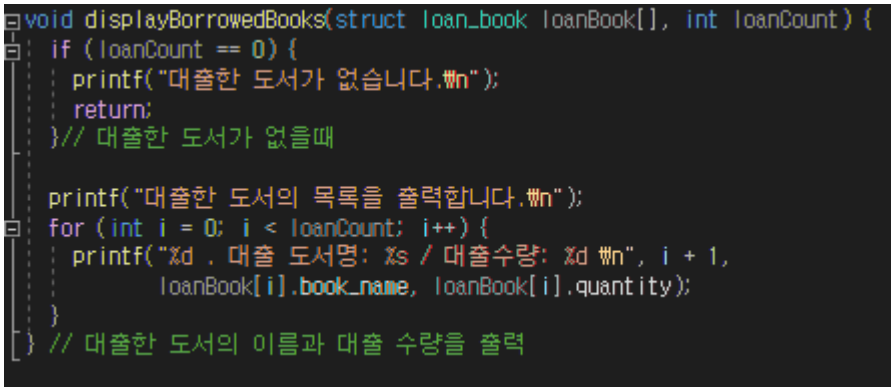
- 설명

대출중인 도서를 확인하기 위해 만든 코드이며, 대출중인 도서의 이름(loanBook [*loanCount].book_name)과 각각의 수량(loanBook [*loanCount].quantity)을 출력한다. 대출한 도서가 없을 경우 "대출한 도서가 없습니다."라는 안내문구를 출력한다.

- 적용된 배운 내용

구조체, 배열, printf , 반복문, 조건문, 함수

- 코드 스크린샷



```
void displayBorrowedBooks(struct loan_book loanBook[], int loanCount) {
    if (loanCount == 0) {
        printf("대출한 도서가 없습니다.₩n");
        return;
    }// 대출한 도서가 없을때

    printf("대출한 도서의 목록을 출력합니다.₩n");
    for (int i = 0; i < loanCount; i++) {
        printf("%d . 대출 도서명: %s / 대출수량: %d ₩n", i + 1,
               loanBook[i].book_name, loanBook[i].quantity);
    }
} // 대출한 도서의 이름과 대출 수량을 출력
```

(3) 대출한 도서 반납 기능

- 입출력

입력:

struct library lib[]:구조체 library 의 배열 lib[] 로 도서의 정보를 저장

struct loan_book loanBook [] : 구조체 loan_book 의 배열 loanBook[] 로 대출 도서의 정보를 저장

char bk_nm[30] : 입력받은 도서명을 임시로 저장할 문자열

loanBook [*loanCount].book_name :구조체 배열 loanBook [] 중 도서의 이름을 저장하는 배열

loanBook [*loanCount].quantity :구조체 배열 loanBook [] 중 도서의 수량을 저장하는 배열

int loanCount : 대출 도서의 종류 수의 변수

int found : 입력받은 도서명이 대출한 도서명과 일치하는지 확인하기 위한 변수

- 설명

반납할 도서를 입력받을 때 대출한 도서가 없다면 "반납할 도서가 없습니다"라는 안내 문구를 출력한다. 입력받은 도서명과 일치하는 도서명을 loanBook[i].book_name 에서 찾기 위해 반복문과 조건문과 strcmp를 사용한다. 일치하는 도서명을 찾는다면 구조체 배열 loanBook []에서 해당 인덱스를 삭제하고 다음 인덱스를 당겨온다. 대출한 도서를 반납했기 때문에 loanCount 를 1 감소 시키고 해당 도서명과 일치하는 기존 도서관 구조체의 수량 (lib[k].quantity)을 1 증가 시킨다

- 적용된 배운 내용

반복문, 조건문, printf(), scanf_s문, 함수, 구조체, 배열

- 코드 스크린샷

```c
// 도서 반납 함수
void returnBook(struct library lib[], int count, struct loan_book loanBook[],
                int* loanCount) {
    char bk_nm[30];
    int found = 0;

    if (*loanCount == 0) {
        printf("반납할 도서가 없습니다.₩n");
        return;
    }

    printf("반납하려는 도서명을 입력하시오: ");
    scanf_s("%s", bk_nm, (int)sizeof(bk_nm));

    for (int i = 0; i < *loanCount; i++) {
        if (strcmp(bk_nm, loanBook[i].book_name) == 0) {
            for (int j = i; j < *loanCount - 1; j++) {
                strcpy_s(loanBook[j].book_name, 20, loanBook[j + 1].book_name);
                loanBook[j].quantity = loanBook[j + 1].quantity;
            }
            // 도서 반납 = 대출 구조체에서 삭제

            (*loanCount)--;
            // 대출 수량 감소

            for (int k = 0; k < count; k++) {
                if (strcmp(bk_nm, lib[k].book_name) == 0) {
                    lib[k].quantity += 1;
                    break;
                }
            }
            // 반납 도서 수량 증가
            printf("도서 반납 완료₩n");
            found = 1;
            break;
        }
    }

    if (!found) {
        printf(
            "대출 기록에 해당 도서가 없습니다.₩n"
            "프로그램 처음으로 돌아갑니다₩n₩n");
    }
}
```

(4) 헤더파일 : fun.h 와 fun.c

- 입출력

입력, 출력

기존의 진척 보고서#1의 기능들 (도서 정보 입력 함수, 보유 장르 출력 함수, 장르별 도서 목록 출력 함수, 보유 도서 종류 수 출력 함수) 와 동일하다

- 설명

기존 진척보고서 #1에서 구현한 기능들을 fun.h fun.c로 헤더파일화 시킨다.

- 적용된 배운 내용

헤더파일

- 코드 스크린샷

fun.h

```
#pragma once
#define PASSWORD 123 // 관리자 암호

// 함수 원형 선언
/*
도서 정보입력
보유 장르 출력
장르별 도서 목록
보유 도서종류수
*/
void addBook(struct library lib[], int* count);
void displayGenres(struct library lib[], int count);
void displayBooksByGenre(struct library lib[], int count);
void displayBookCount(struct library lib[], int count);
```

fun.c

```c
#pragma once
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "fun.h"
#include "struct_library.h"

// 도서 추가 함수, 관리자 선택지(관리자 암호 123추가)
void addBook(struct library lib[], int* count) {
    int password;
    printf("관리자 암호를 입력하시오 : ");
    scanf_s("%d", &password);

    if (password == PASSWORD) {
        printf("도서의 이름을 입력하시오 : ");
        scanf_s("%s", lib[*count].book_name, (int)sizeof(lib[*count].book_name));

        printf("작가의 이름을 입력하시오 : ");
        scanf_s("%s", lib[*count].author, (int)sizeof(lib[*count].author));

        printf("도서의 장르를 입력하시오 : ");
        scanf_s("%s", lib[*count].fields, (int)sizeof(lib[*count].fields));

        printf("도서의 출판사를 입력하시오 : ");
        scanf_s("%s", lib[*count].publisher, (int)sizeof(lib[*count].publisher));

        printf("전체 페이지를 입력하시오 : ");
        scanf_s("%d", &lib[*count].pages, (int)sizeof(lib[*count].pages));

        printf("도서의 가격을 입력하시오 :  ");
        scanf_s("%f", &lib[*count].price, (int)sizeof(lib[*count].price));

        printf("도서의 수량을 입력하시오 :  ");
        scanf_s("%d", &lib[*count].quantity, (int)sizeof(lib[*count].quantity));

        printf("₩n도서 추가 완료!₩n₩n");
        (*count)++;
    } else {
        printf("올바르지 못한 암호입니다.₩n₩n");
    }
}// 도서 정보 입력  -> 도서명 작가명 장르 출판사 페이지 가격 수량
```

```c
// 보유 장르 확인 함수
void displayGenres(struct library lib[], int count) {
    printf("보유하고 있는 장르 목록₩n");

    char printedGenres[20][20];
    int printedCount = 0;

    for (int j = 0; j < count; j++) {
        int genrePrinted = 0;
        for (int k = 0; k < printedCount; k++) {
            if (strcmp(lib[j].fields, printedGenres[k]) == 0) {
                genrePrinted = 1;
                break;
            }
        }

        if (!genrePrinted) {
            printf("%d번 장르: %s₩n", printedCount+1, lib[j].fields);
            strcpy_s(printedGenres[printedCount], 20, lib[j].fields);
            printedCount++;
        }
    }
} // 입력한 도서의 종류수만큼 반복 장르를 출력한 장르에 저장,  출력한적 없는 장르  = 출력, 출력한 장르 수 증가
```

```c
// 장르별 도서 확인 함수
void displayBooksByGenre(struct library lib[], int count) {
    char field_nm[30];
    int found = 0;

    printf("장르명을 입력하시오 : ");
    scanf_s("%s", field_nm, (int)sizeof(field_nm));

    for (int i = 0; i < count; i++) {
        if (strcmp(field_nm, lib[i].fields) == 0) {
            printf("장르 %s에 속한 도서를 출력합니다. \n", field_nm);
            printf("-------------------\n");
            printf("도서명 : %s \n", lib[i].book_name);
            printf("작가명 : %s \n", lib[i].author);
            printf("도서 출판사 : %s \n", lib[i].publisher);
            printf("도서 페이지 : %d \n", lib[i].pages);
            printf("도서 가격 % f \n", lib[i].price);
            printf("보유수량 % d \n", lib[i].quantity);
            if (lib[i].quantity == 0) {
                printf("대출불가!\n");
            } else {
                printf("대출가능!\n");
            }
            printf("-------------------\n\n");
            found = 1;
        }
    }

    if (!found) {
        printf(
            "장르를 잘못 입력하셨습니다.\n"
            "프로그램 처음으로 돌아갑니다\n\n");
    }
}// 입력한 장르와 일치하는 도서의 정보를 출력

// 도서 종류 수 확인 함수
void displayBookCount(struct library lib[], int count) {
    printf("\n총 도서의 종류:  %d\n", count);
}
```

```c
// 도서 반납 함수
void returnBook(struct library lib[], int count, struct loan_book loanBook[],
                int* loanCount) {
    char bk_nm[30];
    int found = 0;

    if (*loanCount == 0) {
        printf("반납할 도서가 없습니다.₩n");
        return;
    }

    printf("반납하려는 도서명을 입력하시오: ");
    scanf_s("%s", bk_nm, (int)sizeof(bk_nm));

    for (int i = 0; i < *loanCount; i++) {
        if (strcmp(bk_nm, loanBook[i].book_name) == 0) {
            for (int j = i; j < *loanCount - 1; j++) {
                strcpy_s(loanBook[j].book_name, 20, loanBook[j + 1].book_name);
                loanBook[j].quantity = loanBook[j + 1].quantity;
            }
            // 도서 반납 = 대출 구조체에서 삭제

            (*loanCount)--;
            // 대출 수량 감소

            for (int k = 0; k < count; k++) {
                if (strcmp(bk_nm, lib[k].book_name) == 0) {
                    lib[k].quantity += 1;
                    break;
                }
            }
            // 반납 도서 수량 증가
            printf("도서 반납 완료₩n");
            found = 1;
            break;
        }
    }

    if (!found) {
        printf(
            "대출 기록에 해당 도서가 없습니다.₩n"
            "프로그램 처음으로 돌아갑니다₩n₩n");
    }
}
```

(5) 헤더파일: brrow.c 와 brrow.h

- 입출력

이번 진척 보고서의 구현한 기능(1),(2),(3) 와 같음

- 설명

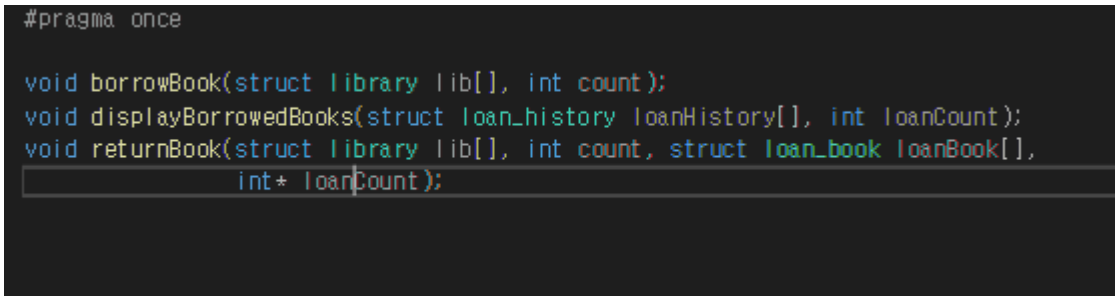
도서 대출 기능, 대출 도서 목록 출력 기능, 도서 반납 기능의 함수들을 헤더파일화 시켰다.

- 적용된 배운 내용

헤더 파일

- 코드 스크린샷

brrow.h



brrow.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "struct_library.h"
#include "brrow.h"

// 도서 대출 함수
void borrowBook(struct library lib[], int count, struct loan_book loanBook[],
                int* loanCount) {
    char bk_nm[30];
    int found = 0;
    if (count == 0) {
        printf("현재 도서관에 저장되어 있는 도서가 없습니다.₩n");
        return;
    }// 도서관에 저장한 도서가 없을때

    printf("대출하려는 도서명을 입력하시오: ");
    scanf_s("%s", bk_nm, (int)sizeof(bk_nm));

    for (int i = 0; i < count; i++) {
        if (strcmp(bk_nm, lib[i].book_name) == 0) {
            if (lib[i].quantity == 0) {
                printf("해당 도서는 전권 대출중입니다. ₩n"); // 수량이 0일때
            } else {
                strcpy_s(loanBook[*loanCount].book_name, 20, lib[i].book_name);
                loanBook[*loanCount].quantity = 0;  // 대출 수량 초기화
                loanBook[*loanCount].quantity += 1;

                lib[i].quantity -= 1;

                (*loanCount)++;

                printf("대출 완료₩n");
            }
            found = 1;
            break;
        }
    }

    if (!found) {
        printf(
            "도서명을 잘못 입력하셨습니다.₩n"
            "프로그램 처음으로 돌아갑니다₩n₩n");
    }
}
```

```c
// 대출 도서 출력함수
void displayBorrowedBooks(struct loan_book loanBook[], int loanCount) {
    if (loanCount == 0) {
        printf("대출한 도서가 없습니다.₩n");
        return;
    }// 대출한 도서가 없을때

    printf("대출한 도서의 목록을 출력합니다.₩n");
    for (int i = 0; i < loanCount; i++) {
        printf("%d . 대출 도서명: %s / 대출수량: %d ₩n", i + 1,
               loanBook[i].book_name, loanBook[i].quantity);
    }
} // 대출한 도서의 이름과 대출 수량을 출력
```

```c
// 도서 반납 함수
void returnBook(struct library lib[], int count, struct loan_book loanBook[],
                int* loanCount) {
    char bk_nm[30];
    int found = 0;

    if (*loanCount == 0) {
        printf("반납할 도서가 없습니다.₩n");
        return;
    }

    printf("반납하려는 도서명을 입력하시오: ");
    scanf_s("%s", bk_nm, (int)sizeof(bk_nm));

    for (int i = 0; i < *loanCount; i++) {
        if (strcmp(bk_nm, loanBook[i].book_name) == 0) {
            for (int j = i; j < *loanCount - 1; j++) {
                strcpy_s(loanBook[j].book_name, 20, loanBook[j + 1].book_name);
                loanBook[j].quantity = loanBook[j + 1].quantity;
            }
            // 도서 반납 = 대출 구조체에서 삭제

            (*loanCount)--;
            // 대출 수량 감소

            for (int k = 0; k < count; k++) {
                if (strcmp(bk_nm, lib[k].book_name) == 0) {
                    lib[k].quantity += 1;
                    break;
                }
            }
            // 반납 도서 수량 증가
            printf("도서 반납 완료₩n");
            found = 1;
            break;
        }
    }

    if (!found) {
        printf(
            "대출 기록에 해당 도서가 없습니다.₩n"
            "프로그램 처음으로 돌아갑니다₩n₩n");
    }
}
```

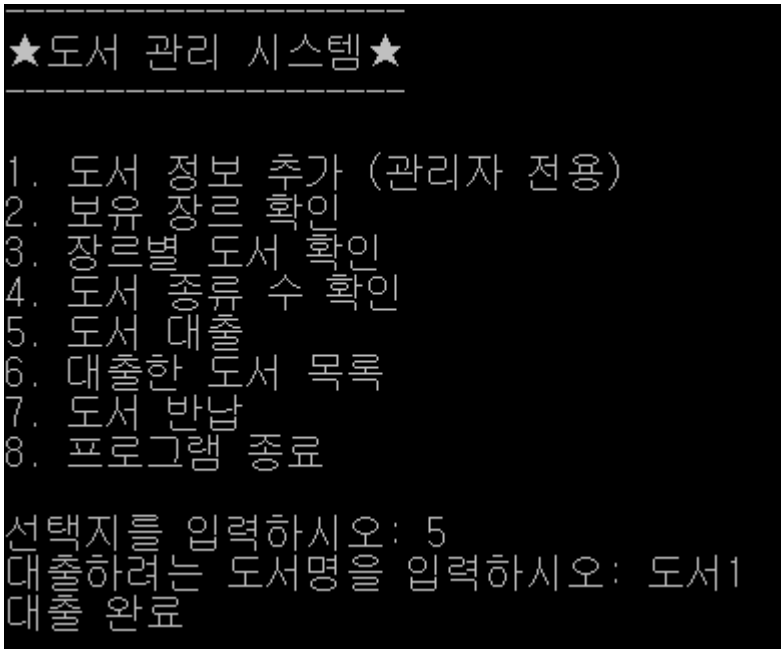
2) 테스트 결과

(1) 도서 대출

- 설명

입력 받은 도서명과 일치하는 도서를 대출한다 이때 한권씩 대출하기 때문에 기존에 있던 도서의 수량을 1 감소 시킨다.

- 테스트 결과 스크린샷

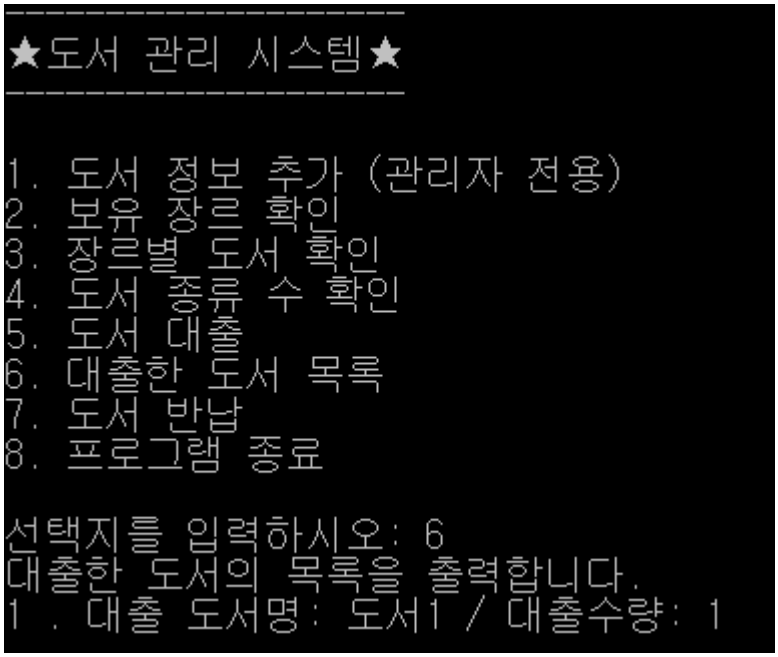


(2) 대출한 도서 목록 출력 기능

- 설명

현재 대출중인 도서의 이름과 대출 수량을 출력한다

- 테스트 결과 스크린샷

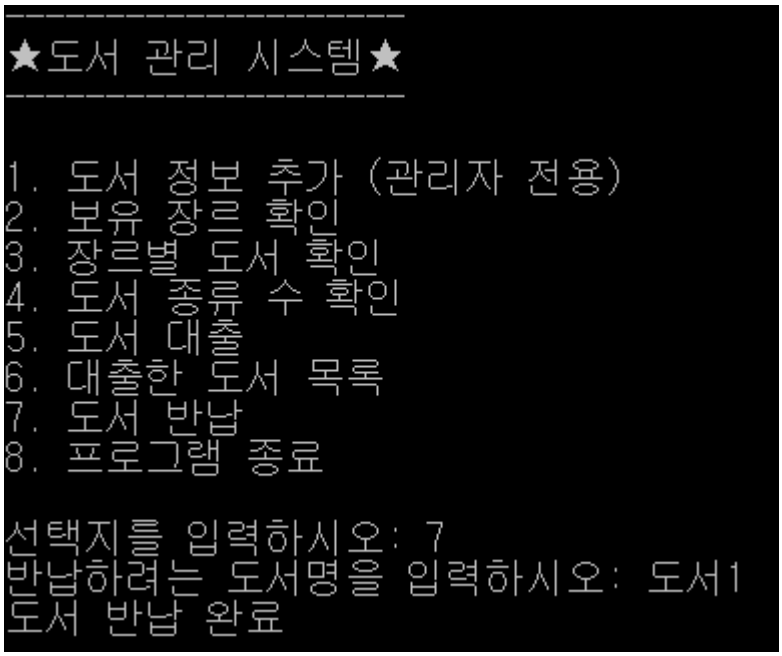


(3) 대출 도서 반납 기능

- 설명

입력받은 도서명과 일치하는 도서를 반납하여 대출중인 도서 구조체에서 삭제하고 기존 구조체의 수량을 1증가시킨다.

- 테스트 결과 스크린샷

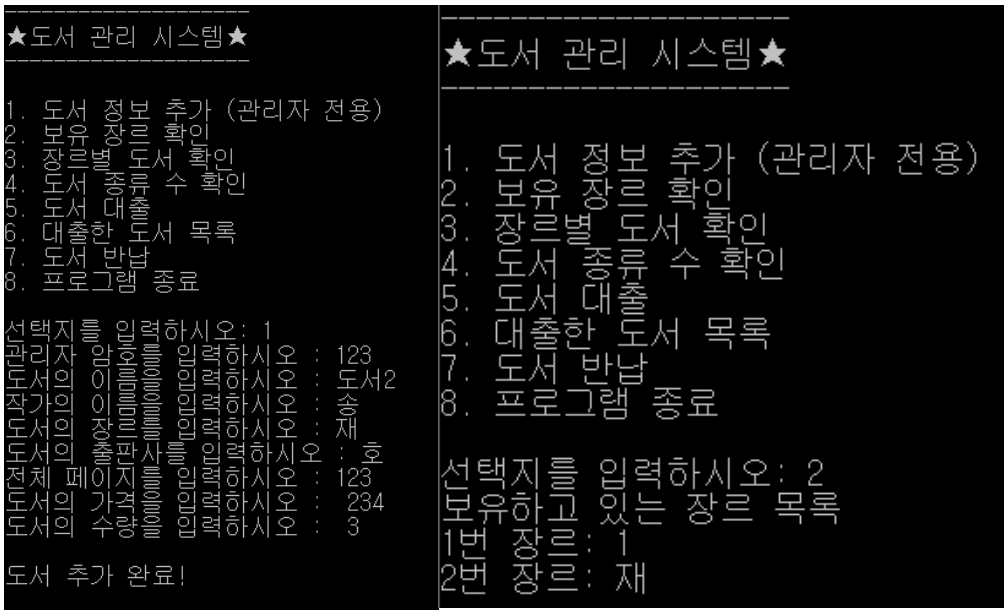


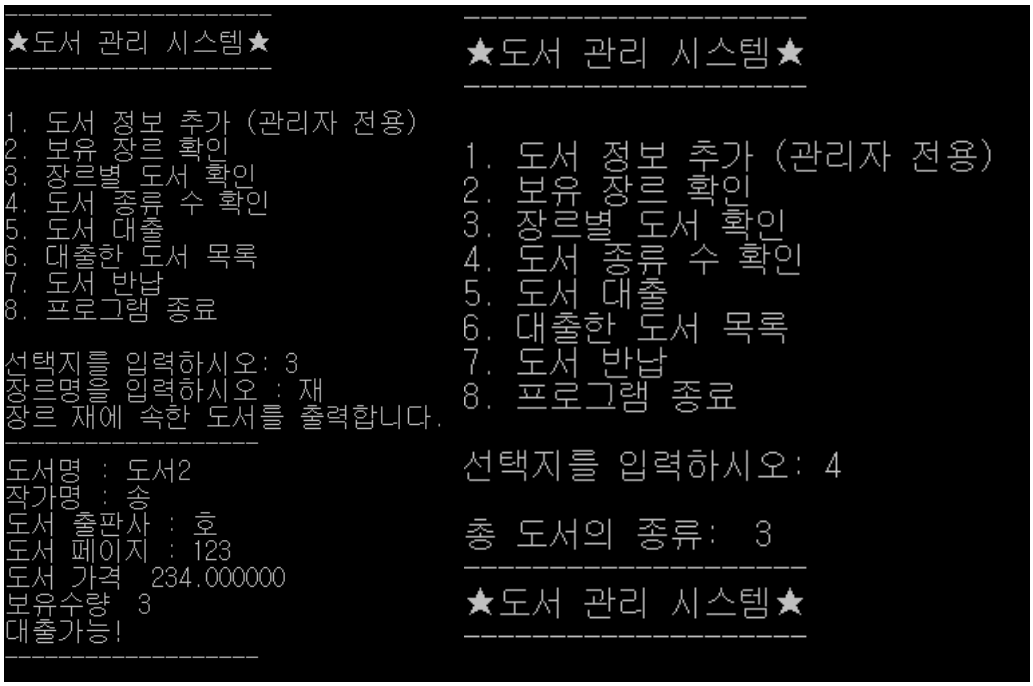
(4) 헤더파일화

- 설명

기존에 있던 함수의 기능들과 새로만든 대출, 반납, 대출중인 파일의 목록을 출력하는 기능 함수들을 헤더파일화 시킨다..

- 테스트 결과 스크린샷





-기능 5,6,7은 테스트 (1),(2),(3)에서 검증완료

4. 계획 대비 변경 사항

1) 헤더 파일화

- 이전: main 함수에 모든 함수, 구조체 선언, 정의

- 이후: 정보입력과 출력함수들을 fun.h로 대출,반납 함수를 brrow.h로 헤더 파일화

- 사유: main 함수가 너무 길어짐에 따라 관리가 힘들다.

5. 프로젝트 일정

업무		11/3	11/13	11/30	12/15	12/20
제안서 작성		완료				
기능1	세부기능1	완료				
	세부기능2	완료				
기능2	세부기능1		완료			
	세부기능2		완료			
기능3	세부기능1			완료		
추가 기능 구현					--------→	

완료 기능

기능 1 : 장르 출력, 장르별 도서 출력

기능 2 : 대출, 반납

기능 3 : 관리자 암호

추가 기능 구현 계획

- 구조체들의 엑셀화
- 알람 기능