C프로그래밍및실습

도서관 관리 시스템

최종 보고서

제출일자: 2023-12-24
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) 도서 정보 저장 기능

- 설명: 관리자가 사용하는 기능으로 도서의 정보를 입력하는 기능이다. 도서 정보의 오류를 막기위해서 관리자만이 도서의 정보를 입력할 수 있다.

(1) 세부 기능

도서의 정보를 입력하기전에 암호를 입력 받고 관리자 암호와 일치한다면 도서의

정보를 입력할 수 있게 하는 기능.

(2) 세부 기능

도서의 정보(도서의 이름, 작가의 이름, 장르명, 출판사, 페이지, 가격, 수량)을 입력 받는다

2) 분야 목록 출력 기능

- 설명: 원하는 장르를 선택하거나 입력한다면 같은 장르의 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 장르의 목록을 출력한다.

(2) 세부 기능

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

3) 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 저장한다.

(2) 세부 기능

반납을 완료 할 경우 해당 서적을 대출의 배열에서 삭제한다.

4) 도서 정보 파일 저장 함수

- 설명: 지금까지 실행한 도서의 정보(기존 도서, 대출한 도서)들을 텍스트 파일이나 엑셀 파일로 저장하는 기능

(1) 세부 기능

텍스트 파일이나 엑셀 파일 중 원하는 파일 형식을 지정하면 해당 파일로 도서의 정보를 저장한다.

3. 기능 구현

1) 기능 구현

(1) 관리자 암호 기능

- 입출력

입력 :

#define PASSWORD 123 전처리기 지시문을 사용하여 미리 정해놓은 관리자 암호.

int password 입력받을 관리자 암호를 저장하기 위한 변수.

출력 :

도서 정보 입력 함수의 일부분이라 반환값은 없으나 "관리자 암호를 입력하시오." 또는 암호를 잘못 입력하였을 경우 "올바르지 못한 암호입니다."라는 안내문구를 출력한다.

- 설명 : 도서의 정보를 입력할 때 관리자가 아닌 이용자의 도서 정보를 막기위한 보안 장치이다.

- 적용된 배운 내용

전처리기, 조건문 ,printf, scanf_s

- 코드 스크린샷

```
void addBook(struct library lib[], int* count) {
    int password;
    printf("관리자 암호를 입력하시오 : ");
    scanf_s("%d", &password);

    if (password == PASSWORD) {
```

암호가 일치하면 다음 코드 실행, 불일치하면

```
else {
printf("올바르지 못한 암호입니다.₩n₩n");
```

(도서 정보 입력 기능 속 기능이라 스크린샷이 짤려있음)

(2) 도서 정보 입력 함수.

- 입출력

입력:

struct library 도서 구조체이며 도서명, 작가명, 장르명, 출판사명, 수량, 가격, 페이지를 저장할 수 있다.

char book_name[20] 도서관 구조체에서 도서명을 저장하는 문자열 배열.

char author[20] 도서관 구조체에서 작가명을 저장하는 문자열 배열.

char fields[20] 도서관 구조체에서 장르명을 저장하는 문자열 배열.

char publisher[20] 도서관 구조체에서 출판사명을 저장하는 문자열 배열.

int quantity 도서관 구조체에서 수량을 저장하는 변수.

int pages 도서관 구조체에서 페이지를 저장하는 변수.

float price 도서관 구조체에서 가격을 저장하는 변수.

getchar() 버퍼 삭제

```
scanf_s("%[^₩n]", lib[*count].book_name, (int)sizeof(lib[*count].book_name))
```

- 도서관 구조체배열(lib)에서 책 이름(lib[*count].book_name)을 저장

```
scanf_s("%[^₩n]", lib[*count].author, (int)sizeof(lib[*count].author))
```

- 도서관 구조체배열(lib)에서 작가 이름(lib[*count].author)을 저장

```
scanf_s("%[^₩n]", lib[*count].fields, (int)sizeof(lib[*count].fields))
```

- 도서관 구조체배열(lib)에서 장르 이름(lib[*count].fields)을 저장

```
scanf_s("%[^₩n]", lib[*count].publisher,(int)sizeof(lib[*count].publisher))
```

- 도서관 구조체배열(lib)에서 출판사 이름(lib[*count].publisher)을 저장

```
scanf_s("%d", &lib[*count].pages, (int)sizeof(lib[*count].pages))
```

- 도서관 구조체배열(lib)에서 도서 페이지(lib[*count].pages)을 저장

```
scanf_s("%f", &lib[*count].price, (int)sizeof(lib[*count].price))
```

- 도서관 구조체배열(lib)에서 도서 가격(lib[*count].price)을 저장

```
scanf_s("%d", &lib[*count].quantity,(int)sizeof(lib[*count].quantity))
```

- 도서관 구조체 배열(lib)에서 도서 수량(lib[*count].quantity)을 저장

*count 도서의 종류 수를 나타내는 변수를 나타내는 포인터

출력: void함수이기 때문에 반환 값은 없으나, 도서의 정보를 입력할 때 각각의 안내문구를 출력하고 잘못 입력한다면, 오류 문구를 출력한다.

- 설명

도서의 정보(도서명, 작가명, 장르, 출판사명, 도서 페이지, 도서 가격, 도서 수량)을 입력 받고 도서관 구조체에 저장한다. 페이지, 가격, 수량을 입력할 때 잘못 입력하면 다시 처음으로 돌아간다.

- 적용된 배운 내용

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

- 코드 스크린샷

```c
void addBook(struct library lib[], int* count) {
    int password;
    printf("관리자 암호를 입력하시오 : ");
    scanf_s("%d", &password);

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

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

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

        printf("도서의 출판사를 입력하시오 : ");
        scanf_s("%[^\n]", lib[*count].publisher,
                (int)sizeof(lib[*count].publisher) );
        getchar();
        printf("전체 페이지를 입력하시오 : ");
        if (scanf_s("%d", &lib[*count].pages, (int)sizeof(lib[*count].pages)) !=1) {
            printf("페이지는 숫자로 입력해주세요.\n");
            while (getchar() != '\n');
            return;
        }

        printf("도서의 가격을 입력하시오 :  ");
        if (scanf_s("%f", &lib[*count].price, (int)sizeof(lib[*count].price)) !=1) {
            printf("가격은 숫자로 입력해주세요.\n");
            while (getchar() != '\n');
            return;
        }

        printf("도서의 수량을 입력하시오 :  ");
        if (scanf_s("%d", &lib[*count].quantity,(int)sizeof(lib[*count].quantity)) != 1) {
            printf("수량은 숫자로 입력해주세요.\n");
            while (getchar() != '\n');
            return;
        }
        printf("\n도서 추가 완료!\n\n");
        (*count)++;
    } else {
        printf("올바르지 못한 암호입니다.\n\n");
    }
```

(3) 보유 장르 출력 기능

- 입출력

입력:

`struct library lib[]` 구조체 `library` 배열

`int count` 도서 종류의 수를 저장하는 변수

`int printedCount` 출력된 장르의 수를 저장하는 변수.

`int genrePrinted` 이전에 출력한적이 있는 장르인지 확인하기 위한 변수.

```
strcpy_s(printedGenres[printedCount], 20, lib[j].fields);
```

- 도서관 구조체배열(`lib`)에서 장르 이름(`lib[*count].fields`)을 `printedGenres[printedCount]`에 복사하여 저장

출력:

void함수이기 때문에 반환값은 없으나, 입력하기전에 안내문구와 보유중인 장르를 출력한다.

- 설명

보유중인 장르를 반복문을 통해 출력한다. 조건문을 통해 이전에 출력한적 있는지 확인하고, 이전에 출력한적 없는 장르만 출력한다. 장르를 출력하고 출력한 장르명 출력 장르명 배열에 저장하고 출력한 장르의 수를 1증가시킨다.

- 적용된 배운 내용

조건문, 반복문, 함수, scanf_s, printf, 문자열, 구조체, 구조체 배열

- 코드 스크린샷

```
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
```

(4) 장르별 도서 정보 확인 기능

- 입출력

입력:

`char field_nm[30]` 장르의 이름을 입력받는 문자열.

`int found` 이전에 출력했는지 확인하기위한 변수.

`if (strcmp(field_nm, lib[i].fields) == 0`

- 입력한 장르가 기존에 저장되어있는 장르와 일치하는지 확인한다.

`char book_name[20]` 도서관 구조체에서 도서명을 저장하는 문자열 배열.

`char author[20]` 도서관 구조체에서 작가명을 저장하는 문자열 배열.

`char fields[20]` 도서관 구조체에서 장르명을 저장하는 문자열 배열.

`char publisher[20]` 도서관 구조체에서 출판사명을 저장하는 문자열 배열.

`int quantity` 도서관 구조체에서 수량을 저장하는 변수.

`int pages` 도서관 구조체에서 페이지를 저장하는 변수.

`float price` 도서관 구조체에서 가격을 저장하는 변수.

출력: void함수이기 때문에 반환 값은 없지만, 입력 받은 장르에 속하는 도서의 정보를 출력한다.

- 설명

장르를 입력받고 입력받은 장르와 동일한 장르에 속하는 도서의 정보(장르 제외)를 출력한다. 장르를 잘못 입력하면, 오류 문구를 출력한다. 대출 가능 여부도 함께 출력한다.

- 적용된 배운 내용

문자열, 함수, 조건문, 반복문, 구조체, 구조체 배열, scanf_s, printf

- 코드 스크린샷

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

  printf("장르명을 입력하시오 : ");
  scanf_s(" %[^\n]", field_nm, (int)sizeof(field_nm));
  getchar();

  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(
        "\n장르를 잘못 입력하셨습니다.\n\n"
        "프로그램 처음으로 돌아갑니다\n\n");
  }
}// 입력한 장르와 일치하는 도서의 정보를 출력
```

(5) 도서의 종류 수 확인

- 입출력

입력:

`struct library lib[]` 구조체 `library` 배열

`int count` 도서 종류의 수를 저장하는 변수

출력: void함수이기 때문에 반환 값은 없으나 도서의 종류 수를 출력한다.

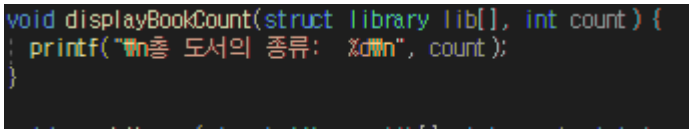
- 설명

보유 중인 도서의 종류 수를 확인한다. 도서의 정보가 저장되었는지 간단하게 확인하기 위한 기능이다.

- 적용된 배운 내용

반복문, printf, 구조체, 구조체 배열, 함수

- 코드 스크린샷



(6) 도서 대출 기능

- 입출력

입력:

struct loan_book 대출한 도서의 정보를 저장하는 구조체(도서명, 수량)

struct library lib[] 구조체 library 배열

struct loan_book loanBook[] 구조체 loan_book 배열

int count 도서 종류의 수를 저장하는 변수

scanf_s("%[^₩n]", bk_nm, (int)sizeof(bk_nm)) 대출하려는 도서명을 입력받는다.

getchar() 버퍼 삭제

lib[i].quantity 도서관 구조체 배열(lib)에서 수량을 저장하는 변수.

loanBook[*loanCount].book_name [20] 대출한 도서 구조체(loanBook)에서 도서명을 저장하는 문자열 배열

loanBook[*loanCount].quantity 대출한 도서 구조체(loanBook)에서 수량을 저장하는 변수

*loanCount 대출한 도서의 권수를 저장하는 변수를 나타내는 포인터

found 입력한 도서명이 올바른 도서명인지 확인하기 위한 변수

출력 : void함수이기 때문에 반환값을 없으나 각각의 오류 문구와 안내문구를 출력한다.

- 설명

도서관 구조체에 저장되어 있는 도서가 있는지 확인하고 없다면 "현재 도서관에 저장되어 있는 도서가 없습니다."라는 안내문구를 출력한다. 저장되어있는 도서가 있다면 대출하고자 하는 도서명과 일치하는 도서를 찾고 해당 도서의 수량에서 1을 감소시키고 대출 도서 구조체 배열에 해당 도서명을 복사하여 붙여넣고, 대출 도서 수량을 1 증가 시킨다.

- 적용된 배운 내용

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

- 코드 스크린샷

```
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: ");
    getchar();
    scanf_s("%[^₩n]", 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");
    }
}
```

(7) 대출 중인 도서 확인 기능

- 입출력

입력:

`struct loan_book` 대출한 도서의 정보를 저장하는 구조체(도서명, 수량)

`struct library lib[]` 구조체 `library` 배열

`struct loan_book loanBook[]` 구조체 `loan_book` 배열

`int count` 도서 종류의 수를 저장하는 변수

`int loanCount`대출한 도서의 권수를 저장하는 변수

`loanBook[*loanCount].book_name [20]` 대출한 도서 구조체(`loanBook`)에서 도서명을 저장하는 문자열 배열

`loanBook[*loanCount].quantity` 대출한 도서 구조체(`loanBook`)에서 수량을 저장하는 변수

출력: void함수라 반환 값은 없으나 대출한 도서가 없는 상황에서 오류 문구와 대출한 도서의 목록을 출력한다.

- 설명

대출한 도서가 없을 때 이 기능을 실행하면 "대출한 도서가 없습니다."라는 오류 문구를 출력하고 대출한 도서가 있다면 대출한 도서 목록을 "도서명 / 대출 수량: "의 형식으로 출력한다.

- 적용된 배운 내용

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

- 코드 스크린샷

```c
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
```

(8) 도서 반납 기능

- 입출력

입력:

struct loan_book 대출한 도서의 정보를 저장하는 구조체(도서명, 수량)

struct library lib[] 구조체 library 배열

struct loan_book loanBook[] 구조체 loan_book 배열

char bk_nm[30] 반납하려고 하는 도서명을 저장할 문자열 배열

scanf_s("%[^₩n]", bk_nm, (int)sizeof(bk_nm)) 반납하려는 도서명을 입력받는다.

getchar() 버퍼 삭제

lib[i].quantity 도서관 구조체 배열(lib)에서 수량을 저장하는 변수.

loanBook[*loanCount].book_name [20] 대출한 도서 구조체(loanBook)에서 도서명을 저장하는 문자열 배열

loanBook[*loanCount].quantity 대출한 도서 구조체(loanBook)에서 수량을 저장하는 변수

*loanCount 대출한 도서의 권수를 저장하는 변수를 나타내는 포인터

found 입력한 도서명이 올바른 도서명인지 확인하기 위한 변수

출력: void함수이기 때문에 반환 값은 없으나, 반납 도서 명을 입력할 때, 잘못 입력하였을때, 대출한 도서가 없을 때, 각각의 오류 문구과 안내문구를 출력한다.

- 설명

대출한 도서가 없다면 "반납할 도서가 없습니다."라는 오류 문구를 출력하고, 대출한 도서가 있다면, 반납하려는 도서 명을 입력 받는다. 입력 받은 도서 명과 일치하는 도서명을 가지는 대출 도서 구조체 배열에서의 인덱스를 삭제한다. 그리고 다음 인덱스를 앞쪽으로 당겨온다. 반납한 도서명과 일치하는 도서관 구조체의 인덱스의 수량을 1 증가시킨다.

- 적용된 배운 내용

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

- 코드 스크린샷

```c
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: ");
    getchar();
    scanf_s("%[^₩n]", bk_nm, (int)sizeof(bk_nm));
    getchar();

    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");
    }
```

(9) 도서 정보 파일 저장 기능

- 입출력

입력:

struct library 도서 구조체이며 도서명, 작가명, 장르명, 출판사명, 수량, 가격, 페이지를 저장할 수 있다.

char book_name[20] 도서관 구조체에서 도서명을 저장하는 문자열 배열.

char author[20] 도서관 구조체에서 작가명을 저장하는 문자열 배열.

char fields[20] 도서관 구조체에서 장르명을 저장하는 문자열 배열.

char publisher[20] 도서관 구조체에서 출판사명을 저장하는 문자열 배열.

int quantity 도서관 구조체에서 수량을 저장하는 변수.

int pages 도서관 구조체에서 페이지를 저장하는 변수.

float price 도서관 구조체에서 가격을 저장하는 변수.

File_Type 도서관 구조체를 저장하는 파일 형식을 저장하는 변수

FILE* file 파일 포인터

int input_output 파일 형식을 지정하기위한 변수

fprintf(file, "도서명,작가명,장르,출판사,수량,페이지,가격₩n") 해당 파일에 ("도서명,작가명,장르,출판사,수량,페이지,가격₩n")을 저장한다.

fopen_s(&file, File_Type, "w") 해당 파일을 '쓰기'모드로 열람한다.

fprintf(file, "%s,%s,%s,%s,%d,%d,%.2f₩n", lib[i].book_name, lib[i].author,lib[i].fields, lib[i].publisher, lib[i].quantity, lib[i].pages,lib[i].price) 해당 파일에 도서의 이름, 작가명, 장르, 출판사, 수량, 페이지, 가격을 저장한다.

출력: void함수이기에 반환 값은 없으나, 파일을 열 수 없을 때 오류 문구와, 파일 형식을 잘못 입력하였을 때, 오류문구를 출력하고 선택한 파일 형식으로 도서관 구조체 배열을 저장한다.

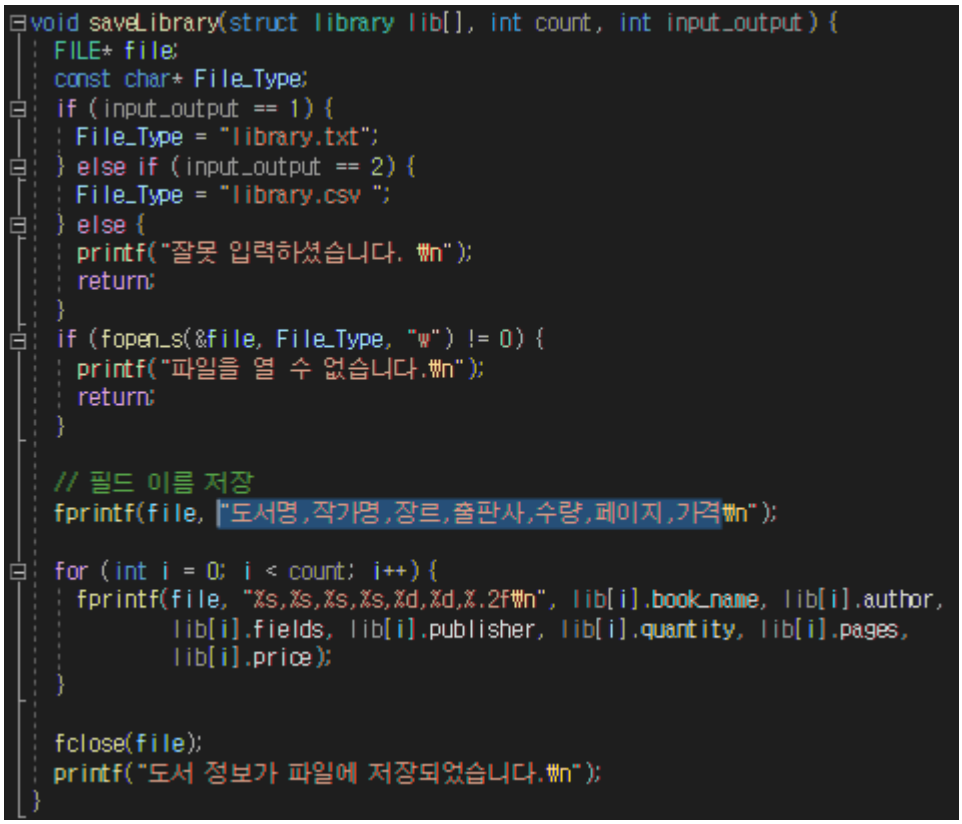
- 설명

도서 정보 입력 기능을 실행하여 생성한 도서관 구조체 배열을 텍스트 파일과 엑셀 파일 중 선택한 파일 형식으로 저장한다. 이때 파일 머리말로 ("도서명,작가명,장르,출판사,수량,페이지,가격")을 먼저 저장한다. 파일을 열 수 없거나, 정해놓은 파일 형식을 제외한 다른 형식의 파일을 입력할 경우 오류 문구를 출력한다.

- 적용된 배운 내용

파일 입출력, 함수, 구조체 배열, 조건문, 반복문, scanf_s, printf, 문자열, 포인터

- 코드 스크린샷



```
void saveLibrary(struct library lib[], int count, int input_output) {
    FILE* file;
    const char* File_Type;
    if (input_output == 1) {
        File_Type = "library.txt";
    } else if (input_output == 2) {
        File_Type = "library.csv ";
    } else {
        printf("잘못 입력하셨습니다. ₩n");
        return;
    }
    if (fopen_s(&file, File_Type, "w") != 0) {
        printf("파일을 열 수 없습니다.₩n");
        return;
    }

    // 필드 이름 저장
    fprintf(file, "도서명,작가명,장르,출판사,수량,페이지,가격₩n");

    for (int i = 0; i < count; i++) {
        fprintf(file, "%s,%s,%s,%s,%d,%d,%.2f₩n", lib[i].book_name, lib[i].author,
                lib[i].fields, lib[i].publisher, lib[i].quantity, lib[i].pages,
                lib[i].price);
    }

    fclose(file);
    printf("도서 정보가 파일에 저장되었습니다.₩n");
}
```

(10) 대출 도서 정보 파일 저장 기능

- 입출력

입력:

`struct loan_book` 대출한 도서의 정보를 저장하는 구조체(도서명, 수량)

`struct loan_book loanBook[]` 구조체 `loan_book` 배열

`loanBook[*loanCount].book_name [20]` 대출한 도서 구조체(`loanBook`)에서 도서명을 저장하는 문자열 배열

`loanBook[*loanCount].quantity` 대출한 도서 구조체(`loanBook`)에서 수량을 저장하는 변수

`File_Type` 도서관 구조체를 저장하는 파일 형식을 저장하는 변수

`FILE* file` 파일 포인터

`int input_output` 파일 형식을 지정하기위한 변수

출력: void함수이기에 반환 값은 없으나, 파일을 열 수 없을 때 오류 문구와, 파일 형식을 잘못 입력하였을 때, 오류문구를 출력하고 선택한 파일 형식으로 대출 도서 구조체 배열을 저장한다.

- 설명

도서 대출 기능을 실행하여 생성한 대출 도서 구조체 배열을 텍스트 파일과 엑셀 파일중 선택한 파일 형식으로 저장한다. 이때 파일 머리말로 ("도서명,수량")을 먼저 저장한다. 파일을 열 수 없거나, 정해놓은 파일 형식을 제외한 다른 형식의 파일을 입력할 경우 오류 문구를 출력한다.

- 적용된 배운 내용

파일 입출력, 함수, 구조체 배열, 조건문, 반복문, scanf_s, printf, 문자열, 포인터

- 코드 스크린샷

```c
void saveLoanBooks(struct loan_book loanBook[], int loanCount,
                   int input_output) {
    FILE* file;
    const char* File_Type;
    if (input_output == 1) {
        File_Type = "loan_books.txt";
    } else if (input_output == 2) {
        File_Type = "loan_books.csv";
    } else {
        printf("잘못 입력하셨습니다. ₩n");
        return;
    }
    if (fopen_s(&file, File_Type, "w") != 0) {
        printf("파일을 열 수 없습니다.₩n");
        return;
    }

    fprintf(file, "도서명,수량₩n");

    for (int i = 0; i < loanCount; i++) {
        fprintf(file, "%s,%d₩n", loanBook[i].book_name, loanBook[i].quantity);
    }

    fclose(file);
    printf("대출 도서 정보가 파일에 저장되었습니다.₩n");
}
```

(10) 헤더파일화 (1) – fun.h , fun.c

- 입출력

기능1,2,3,4,5,9,10 의 기능들을 fun.h ,fun.c 로 헤더파일화 시켰다.

- 설명

대출, 반납의 기능을 제외한 기능1,2,3,4,5,9,10 을 fun.h로 헤더파일화 시켰다.

- 적용된 배운 내용

헤더파일, 구조체, 구조체 배열, 문자열, 조건문, 반목문, 파일 입출력, 포인터, 배열, 전처리기, 함수

- 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);
void saveLoanBooks(struct loan_book loanBook[], int loanCount,int input_output);
void saveLibrary(struct library lib[], int count, int input_output);
```

fun.h

```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 : ");
        getchar();
        scanf_s("%[^\n]", lib[*count].book_name, (int)sizeof(lib[*count].book_name));
        getchar();

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

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

        printf("도서의 출판사를 입력하시오 : ");
        scanf_s("%[^\n]", lib[*count].publisher,
                (int)sizeof(lib[*count].publisher) );
        getchar();
        printf("전체 페이지를 입력하시오 : ");
        if (scanf_s("%d", &lib[*count].pages, (int)sizeof(lib[*count].pages)) !=1) {
            printf("페이지는 숫자로 입력해주세요.\n");
            while (getchar() != '\n');
            return;
        }
```

```c
        printf("도서의 가격을 입력하시오 :  ");
        if (scanf_s("%f", &lib[*count].price, (int)sizeof(lib[*count].price)) !=1) {
            printf("가격은 숫자로 입력해주세요.\n");
            while (getchar() != '\n');
            return;
        }

        printf("도서의 수량을 입력하시오 :  ");
        if (scanf_s("%d", &lib[*count].quantity,(int)sizeof(lib[*count].quantity)) != 1) {
            printf("수량은 숫자로 입력해주세요.\n");
            while (getchar() != '\n');
            return;
        }
        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 정보 확인 함수
void displayBooksByGenre(struct library lib[], int count) {
    char field_nm[30];
    int found = 0;

    printf("장르명을 입력하시오 : ");
    scanf_s(" %[^₩n]", field_nm, (int)sizeof(field_nm));
    getchar();

    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(
            "₩n장르를 잘못 입력하셨습니다.₩n₩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 정보 파일로 저장 함수
void saveLibrary(struct library lib[], int count, int input_output) {
    FILE* file;
    const char* File_Type;
    if (input_output == 1) {
        File_Type = "library.txt";
    } else if (input_output == 2) {
        File_Type = "library.csv ";
    } else {
        printf("잘못 입력하셨습니다. \n");
        return;
    }
    if (fopen_s(&file, File_Type, "w") != 0) {
        printf("파일을 열 수 없습니다.\n");
        return;
    }

    // 필드 이름 저장
    fprintf(file, "도서명,작가명,장르,출판사,수량,페이지,가격\n");

    for (int i = 0; i < count; i++) {
        fprintf(file, "%s,%s,%s,%s,%d,%d,%.2f\n", lib[i].book_name, lib[i].author,
                lib[i].fields, lib[i].publisher, lib[i].quantity, lib[i].pages,
                lib[i].price);
    }

    fclose(file);
    printf("도서 정보가 파일에 저장되었습니다.\n");
}
```

```c
// 대출 도서 정보 파일로 저장 함수
void saveLoanBooks(struct loan_book loanBook[], int loanCount,
                   int input_output) {
    FILE* file;
    const char* File_Type;
    if (input_output == 1) {
        File_Type = "loan_books.txt";
    } else if (input_output == 2) {
        File_Type = "loan_books.csv";
    } else {
        printf("잘못 입력하셨습니다. \n");
        return;
    }
    if (fopen_s(&file, File_Type, "w") != 0) {
        printf("파일을 열 수 없습니다.\n");
        return;
    }

    fprintf(file, "도서명,수량\n");

    for (int i = 0; i < loanCount; i++) {
        fprintf(file, "%s,%d\n", loanBook[i].book_name, loanBook[i].quantity);
    }

    fclose(file);
    printf("대출 도서 정보가 파일에 저장되었습니다.\n");
}
```

(11) 헤더파일화 (1) – borrow.h , borrow.c

- 입출력

기능 6,7,8 을 borrow.h , borrow.c로 헤더파일화 시켰다

- 설명

대출기능과 반납기능을 헤더파일화 시켰다

- 적용된 배운 내용

헤더파일, 구조체, 구조체 배열, 문자열, 조건문, 반목문, 파일 입출력, 포인터, 배열, 함수

- 코드 스크린샷

borrow.h

```
#pragma once

void borrowBook(struct library lib[], int count);
void displayBorrowedBooks(struct loan_history loanHistory[], int loanCount);
void returnBook(struct library lib[], int count, struct loan_book loanBook[],
                int* loanCount);
```

borrow.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "struct_library.h"
#include "borrow.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: ");
    getchar();
    scanf_s("%[^₩n]", 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: ");
    getchar();
    scanf_s("%[^₩n]", bk_nm, (int)sizeof(bk_nm));
    getchar();

    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");
    }
}
```

(11) 헤더파일화 (3) – struct_library.h

- 입출력

struct library 도서 구조체이며 도서명, 작가명, 장르명, 출판사명, 수량, 가격, 페이지를 저장할 수 있다.

char book_name[20] 도서관 구조체에서 도서명을 저장하는 문자열 배열.

char author[20] 도서관 구조체에서 작가명을 저장하는 문자열 배열.

char fields[20] 도서관 구조체에서 장르명을 저장하는 문자열 배열.

char publisher[20] 도서관 구조체에서 출판사명을 저장하는 문자열 배열.

int quantity 도서관 구조체에서 수량을 저장하는 변수.

int pages 도서관 구조체에서 페이지를 저장하는 변수.

float price 도서관 구조체에서 가격을 저장하는 변수.

struct loan_book 대출한 도서의 정보를 저장하는 구조체(도서명, 수량)

loanBook[*loanCount].book_name [20] 대출한 도서 구조체(loanBook)에서 도서명을 저장하는 문자열 배열

loanBook[*loanCount].quantity 대출한 도서 구조체(loanBook)에서 수량을 저장하는 변수

- 설명

코드에서 사용하는 구조체(도서관 구조체, 대출한 도서 구조체)를 헤어파일화 시켰다.

- 적용된 배운 내용

헤더파일, 구조체, 배열, 문자열

- 코드 스크린샷

```
#pragma once
// 도서관 구조체(도서명 작가명 장르 출판사 수량 페이지 가격)
struct library {
    char book_name[20];
    char author[20];
    char fields[20];
    char publisher[20];
    int quantity;
    int pages;
    float price;
};

// 대출한 도서 정보 구조체(도서명 수량 )
struct loan_book {
    char book_name[20];
    int quantity;

};
```

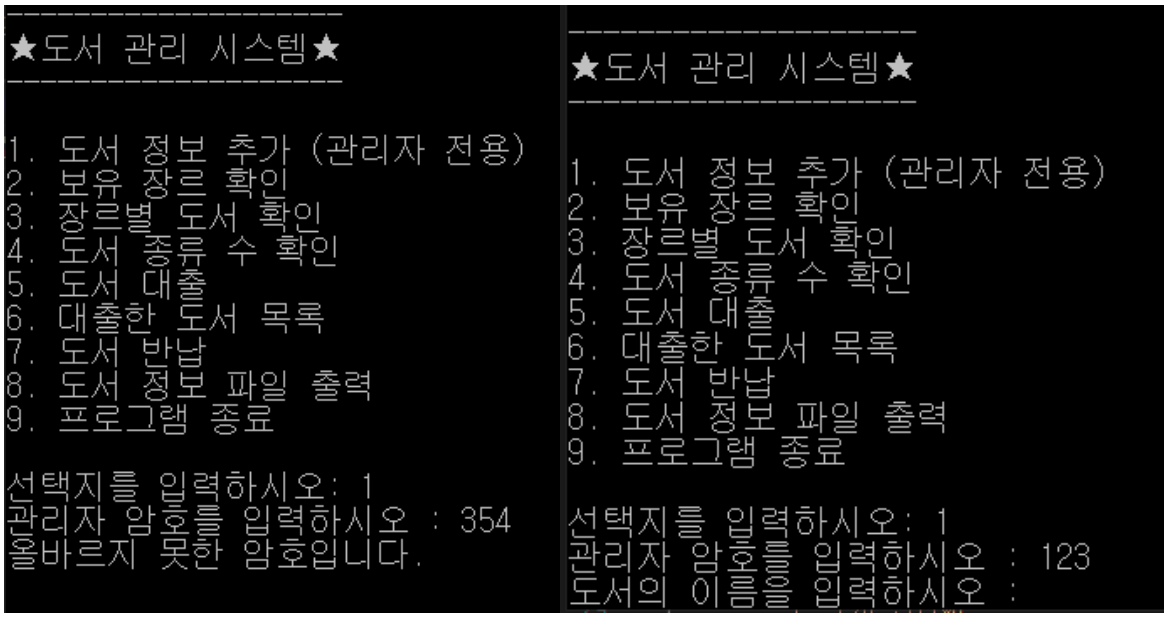
4. 테스트 결과

(1) 관리자 암호 입력 기능

- 설명

도서의 정보를 입력하려면 관리자 암호를 입력해야한다.

- 테스트 결과 스크린샷

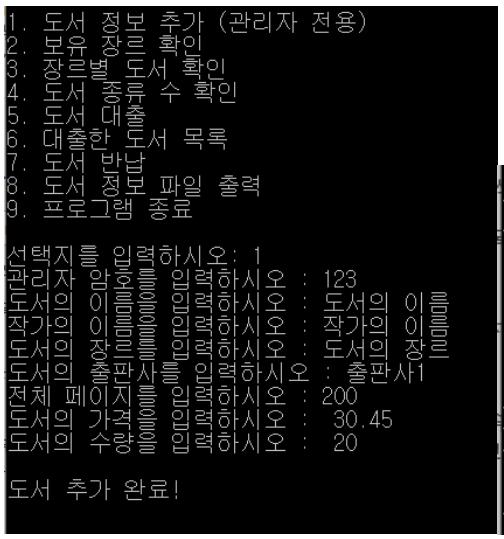


(2) 도서 정보 입력 기능

- 설명

도서의 정보를 입력한다 도서의 정보에는 도서명, 작가명, 장르, 출판사, 페이지, 수량, 가격이 있다. 수량과 가격, 페이지에 잘못된 값을 저장하려고 하면 오류 문구를 출력하고 처음으로 돌아간다.

- 테스트 결과 스크린샷

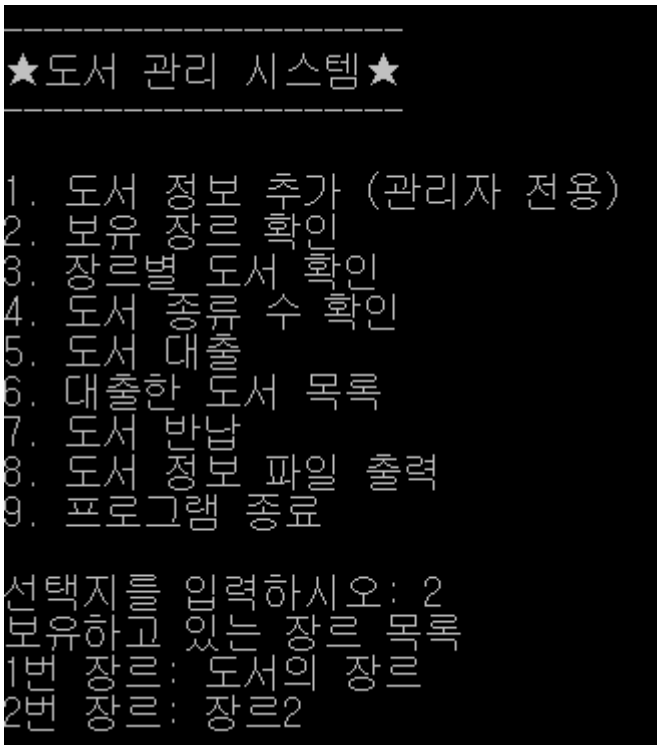


(3) 보유 장르 출력 기능

- 설명

보유 중인 장르를 출력해준다.

- 테스트 결과 스크린샷

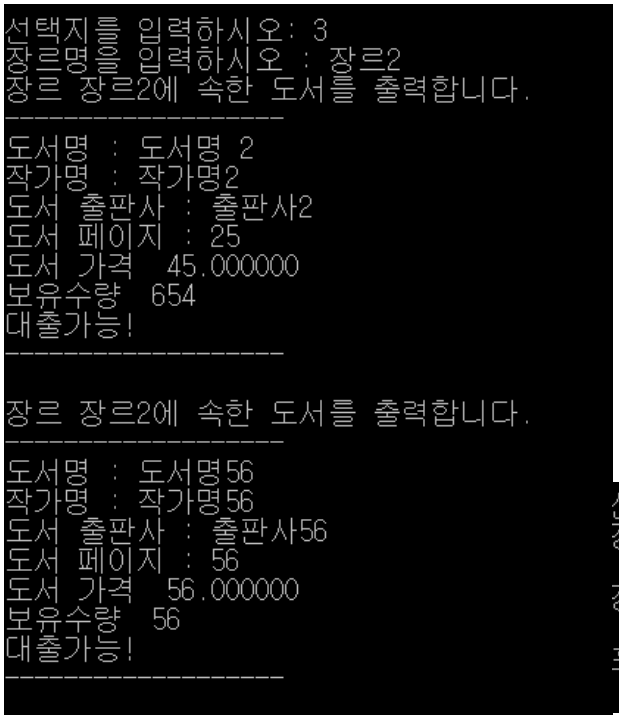


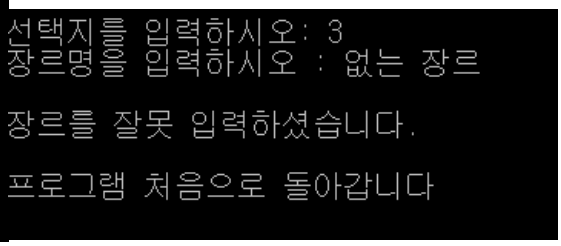
(4) 장르별 도서 정보 출력 기능

- 설명

입력한 장르의 도서 정보를 출력한다. 잘못 입력하면 오류 문구를 출력한다. 대출가능 여부도 함께 출력한다.

- 테스트 결과 스크린샷



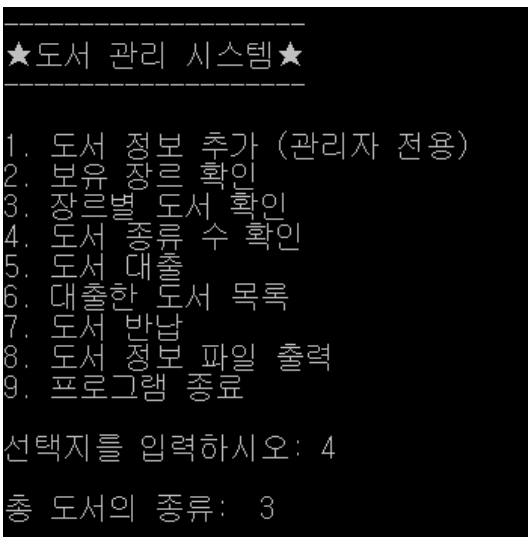


(5) 보유 도서 종류 수 확인 기능

- 설명

보유중인 도서의 종류 수를 출력한다.

- 테스트 결과 스크린샷

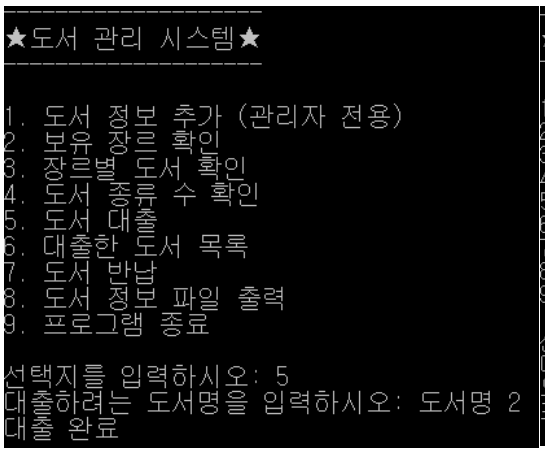


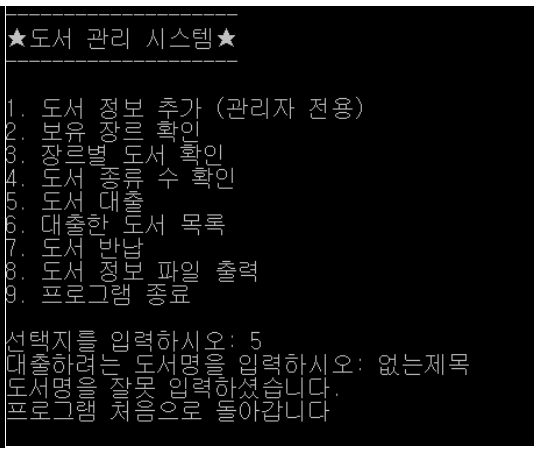
(6) 도서 대출 기능

- 설명

보유중인 도서를 대출하며 보유중인 도서가 없거나, 잘못된 도서명을 입력하면 오류문구를 출력한다.

- 테스트 결과 스크린샷



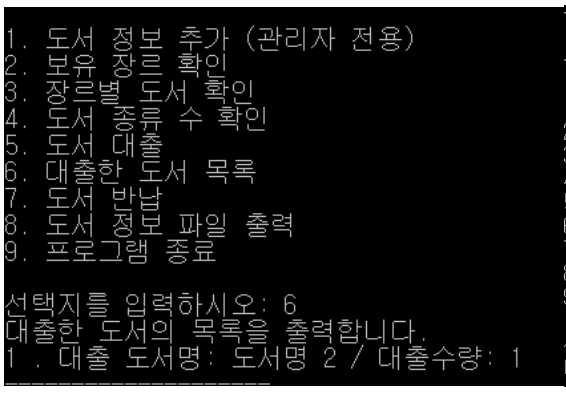


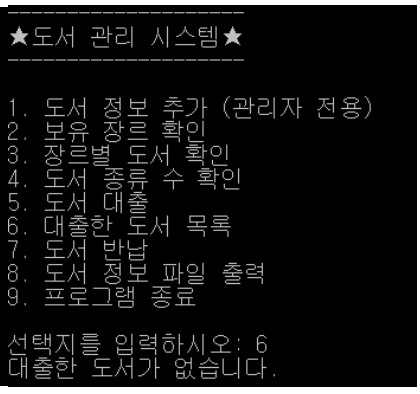
(7) 대출중인 도서 목록 출력 기능

- 설명

대출중인 도서의 목록을 출력한다. 대출중인 도서가 없을경우 오류문구를 출력한다.

- 테스트 결과 스크린샷



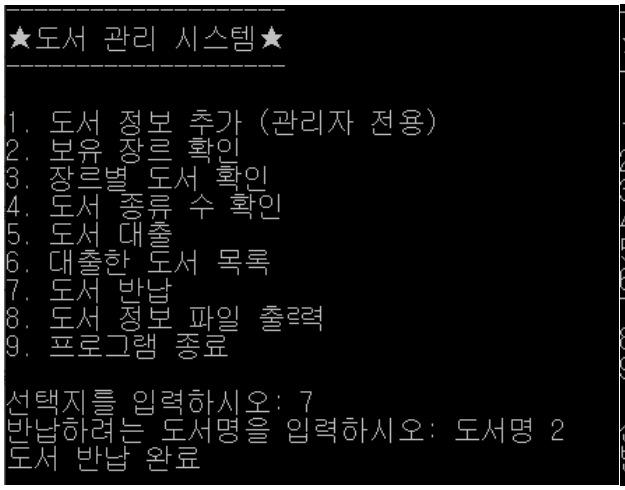


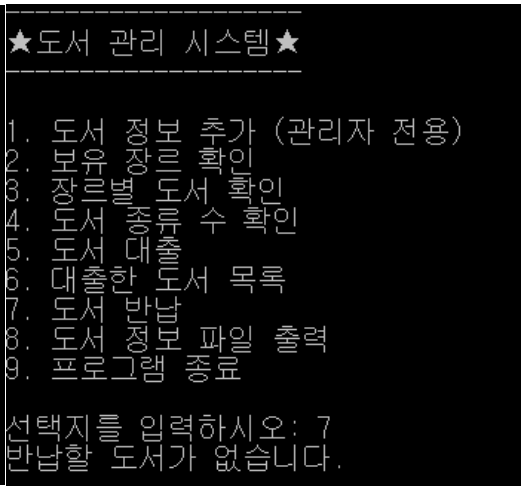
(8) 도서 반납 기능

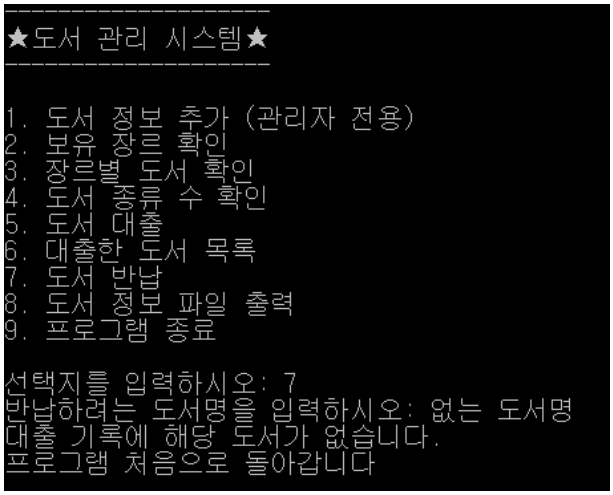
- 설명

대출한 도서중 입력한 도서를 반납한다. 대출한 도서가 없거나 잘못된 도서명을 입력한다면 오류 문구를 출력한다.

- 테스트 결과 스크린샷





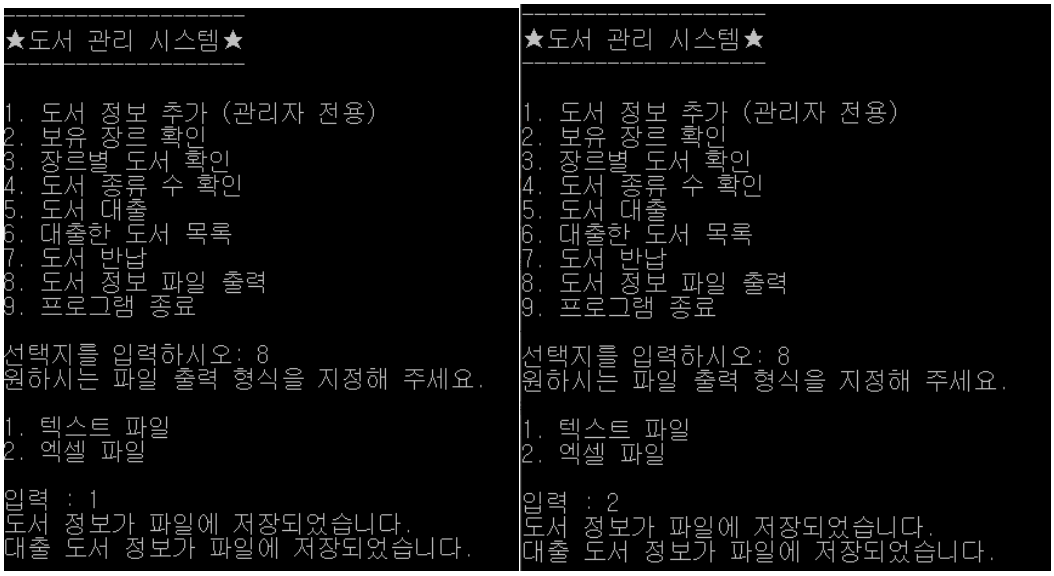


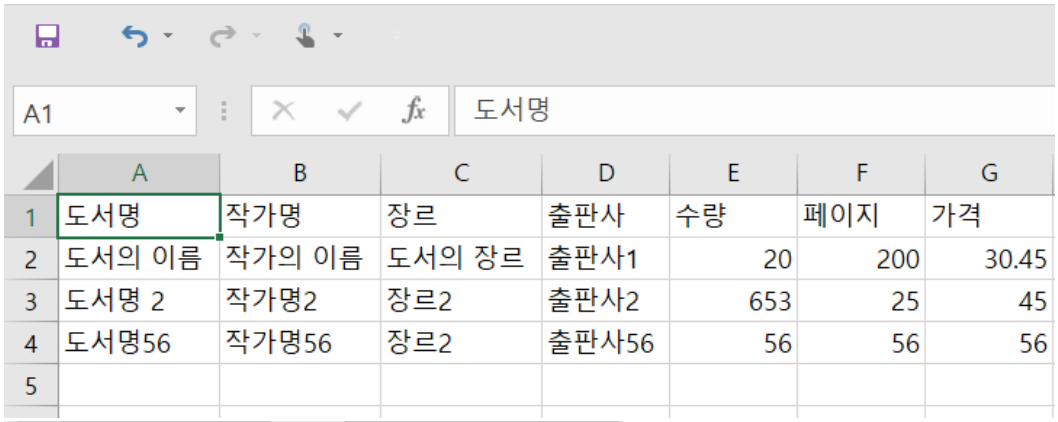
(9) 도서 정보 파일 저장기능

- 설명

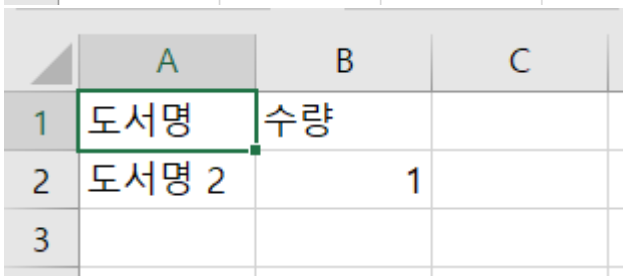
저장한 도서의 정보와 대출 도서 정보를 원하는 파일 형식으로 저장한다.

- 테스트 결과 스크린샷





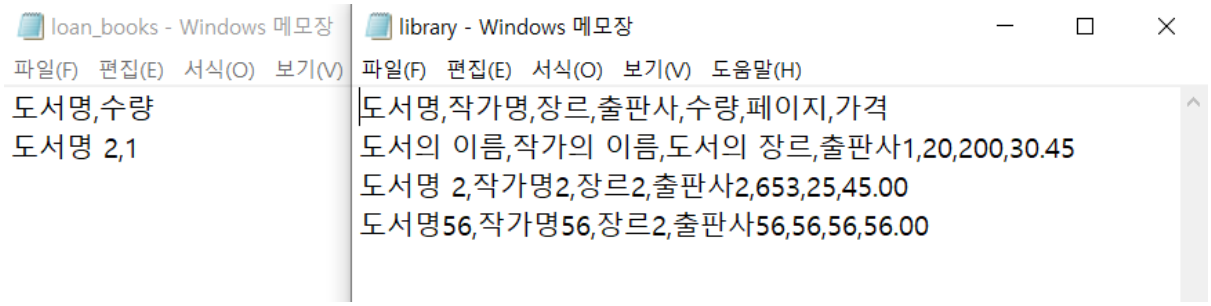
	A	B	C	D	E	F	G
1	도서명	작가명	장르	출판사	수량	페이지	가격
2	도서의 이름	작가의 이름	도서의 장르	출판사1	20	200	30.45
3	도서명 2	작가명2	장르2	출판사2	653	25	45
4	도서명56	작가명56	장르2	출판사56	56	56	56
5							



	A	B	C
1	도서명	수량	
2	도서명 2	1	
3			

상 : 도서관 구조체

하 : 대출한 도서 구조체



좌 : 대출한 도서 구조체

우 : 도서관 구조체

5. 계획 대비 변경 사항

1) 헤더 파일화

1. 소스코드가 길어지고 파일이 많아짐에 따라 관리가 어려워졌다.

2. 핵심기능을 하는 코드들만 소스 파일에 작성이 가능하다.

3. 특정 기능을 하는 함수, 구조체들의 재사용 쉽다.

- 위와 같은 3가지 이유로 대출,반납의 기능을 하는 함수들 , 사용하는 구조체들, 나머지 기능들을 묶어 헤더 파일화 시켰습니다..

2) 관리자 암호

도서의 정보를 일반 사용자가 입력하지 못하고 관리자 암호를 아는 관리자만이 도서의 정보를 입력 할 수 있다.

- 일반 사용자가 도서의 정보를 입력하다 잘못된 정보를 입력하는 것을 방지하고자 관리자 암호를 추가하였습니다.

3) 도서관 도서 정보, 대출 도서 정보들을 파일로 저장

도서관 도서 정보 구조체와 대출 도서 정보 구조체를 지정한 파일 형식(엑셀, 텍스트) 중 원하는 파일 형식으로 저장 할 수 있다.

- 저장한 도서 정보와 대출한 도서 정보를 한눈에 보기 쉽게 원하는 파일로 저장하는 기능을 추가하였습니다.

4) 반납 알림기능 삭제

도서를 대출하고 반납 기간의 일정시간전에 알림을 보내는 기능을 삭제

- 해당 기능을 구현하려면 프로그램을 계속 항시 실행하여야 하고 수업내용에서 구현하기에 어려움을 느꼈습니다. 해당 기능은 c언어를 통해 구현하기보다는 프로그램을 통해 엑셀파일로 저장하여 엑셀의 기능이나 다른 프로그램의 기능을 통해 구현하는게 더 효율적이라는 생각을 하여 삭제하였습니다.

6. 느낀점

- 도서관 관리 프로그램을 만들면서 처음 계획했던 기능들보다 많은 기능들이 생겼고 처음 생각했던 기능들도 코드를 짜면서 예상했던 코드보다 더 복잡해졌습니다. 기존에 예상하였을 때 단순히 배열 만을 통해 만들려고 하였던 도서관 도서 정보다 대출 정보 모두 구조체가 거의 필수적이었고 해당 구조체를 통해 구조체 배열을 만들고 대출, 반납의 기능에서 구조체 배열 일부를 수정, 복사 삭제 등의 과정을 수행하면서 구조체 배열을 다루는 숙련도가 향상되었습니다.

제안서에서 처음 제안하였던 알림 기능은 코딩을 할수록 코드 속에서 c언어로만 구현하기에는 비효율적이라는 생각이 들어 삭제하였습니다. 대출 기능과 time.h를 연계하여 쉽게 작성할 수 있을 거라고 생각하였는데 일정 시간을 측정하는 기능만을 찾아볼 수 있어 어려움을 겪고 다른 외부 프로그램을 사용해야 한다는 결론을 얻었습니다.

수업을 통해 배운 여러 가지 기능들을 실제로 제가 프로그램을 코딩을 작성하면서 적용하다 보니 중간고사 시험 이전보다 c언어 숙련도가 크게 상승하였다고 자신할 수 있을 정도의 성장하였습니다. 다음에 제가 프로그래밍을 할 때 이러한 경험이 큰 도움이 될 것 같습니다.