
# 도서관리 프로그램 작성

| | |
|---|---|
| 과목명 | 프로그래밍실습2 |
| 교수님 | 박유현 |
| 학 과 | 창의소프트웨어공학부 |
| 분 반 | 002 |
| 학 번 | 20193148 |
| 이 름 | 황진주 |
| 날 짜 | 2019.11.18 |


東義大學校
DONG-EUI UNIVERSITY

# 목차

## 〈 과제 〉

주제 : 구조체를 이용하여 도서관리 프로그램 작성하기

조건 :

. 책 정보 : 책제목, 페이지 수, 가격, 출판 날짜, 저자

. 도서관리 기능 : 책 정보 입력/삭제/검색/출력

. 도서관의 책은 10권 이내로 제한함

## 〈 프로그램 소개 〉

함수로 나눔에도 코드의 길이가 길어져 사용의 어려움이 있어, 기능에 따라 함수를 구분하여 헤더 파일로 나누었습니다. 프로그램은 큰 틀로 정보 저장/관리, 출력, 탐색으로 나누었고 7개의 헤더 파일과 1개의 소스파일로 작성되었습니다.

### 제시된 기능의 구현

※ 굵기 처리 된 내용은
제 프로그램만의 차별성을 나타냅니다.

정보 저장 :
도서 정보는 제목, 저자, 페이지 수, 가격, 출판 일을 구조체에 저장하였습니다.
구조체 배열로 재 선언 한 후 기본 정보들을 입력해두었습니다.

입력 :
입력을 통해 프로그램 동작 방향을 지정했습니다. getline을 사용하여 띄어쓰기가 있는 문자열도 받을 수 있도록 하였습니다.

삭제 :
clear()을 이용하여 책의 제목을 지웁니다. 제목이 없으면 정보가 없는 것으로 판단하고 그 정보의 인덱스 번호를 가져옵니다. 가져온 인덱스를 정보 유무를 저장하는 bool배열(presenceOfInfo)의 값을 false로 변경하여 정보가 없음을 알 수 있도록 하였습니다.

검색 :
프로그램 작성에서 가장 공을 들인 기능입니다. 모든 문자열이 같은 경우만이 아닌 **일부 문자가 같은 경우에도 같음을 알아**내 그 인덱스를 정보가 일치함을 저장하는 int배열(matchPoint)에 저장하여 값이 같음을 표기하였습니다. 'KMP(Knuth-Morris-Pratt)' 문자열 검색 알고리즘을 통해 검색을 하여, strcmp를 사용하는 경우보다 높은 탐색도를 보이도록 하였습니다.

출력 :
프로그램 실행 시 화면이 작아 원하는 출력 형태가 나오지 않아 system("mode con cols=180 lines=40");를 이용하여 **콘솔 크기를 설정**하였습니다. 목록 출력 시 입력 위치의 균일함을 위하여 width(), left, setw()을 활용해 **왼쪽 정렬**을 하여 출력하였습니다. 입력 값과 **중요 내용은 색을 다르게 설정**해 구분을 두어 가독성을 높였습니다. 가나다 순 정렬, 페이지 많은 순으로, 페이지 적은 순으로, 가격 높은 순으로, 가격 낮은 순으로 **목차를 정렬 기능**을 지원해 원하는 정보를 쉽게 얻도록 하였습니다.

〈 프로그램 파일 구성 〉

| 이름 | 기능 |
|---|---|
| main.cpp | 프로그램 총 동작 지시 |
| mode.h | 실행 타입 설정 |
| findSting.h | 문자 탐색 |
| bookData.h | 데이터 저장 위치 |
| searchBook.h | 도서 검색 동작시의 입출력 |
| manageData.h | 데이터 변경(추가/삭제) |
| printInformation.h | 각종 출력 |
| sortData.h | 데이터 정렬 |

〈 작성 코드 〉

main.cpp

```
#include "bookData.h"
#include "searchBook.h"
#include "mode.h"

using namespace std;

#define COMMENT ((int)4294967296+1)
#define endl "\n"

bool pyeonHae;


int main() {
	// 페이지수, 가격, 출간일, 책이름, 작가, 출판사
	// 콘솔 크기 조정
	system("mode con cols=180 lines=40");
	while (COMMENT) {
		setColor(white, black);		// 색상 초기화
		system("cls");				// 화면 초기화
		printMode(checkOverflow());	// 모드 출력

		checkOverflow();			// 정보 입력 확인 bool배열 정돈 및 추가 가능 여부 확인
		int mode = selectMode();	// 사용 모드 선택

		if (mode == 1) {			// 1. 도서 검색
			searchInfo();
			exit();
		}
		else if (mode == 2) {		// 2. 도서 추가
			addBook();
			exit();
		}
		else if (mode == 3) {		// 3. 도서 삭제
			deleteBook();
			exit();
		}
```

```
                else if (mode == 4) {              // 4. 도서 전체 출력
                        printBasic();
                        sortString(checkOverflow());
                        printAllList();
                        selectsort();
                        exit();
                }
        }
}
```

## mode.h

```
#include "bookData.h"
#include "sortData.h"

int selectMode();

// 모드 선정
int selectMode() {

	while (true) {
		string inputMode;
		setColor(lightPurple, black);
		cin >> inputMode;
		setColor(white, black);

		// 허용 범위 : 0 < temp < 버튼의 갯수
		// 예외 : 없음
		int mode = exceptionHandling(inputMode, 0, 4, INT_MIN);
		if (mode) {
			return mode;
		}
	}
}

// 정렬법 선택
void selectsort() {

	while (true) {
		string inputMode;

		cout << "\n\n0. 메인으로  1. 가나다 순 정렬  2.페이지 많은 순으로  3.페이지 적은 순으로  4.가격 높은 순으로  5.가격 낮은 순으로\n\n";

		setColor(lightPurple, black);
		cin >> inputMode;
		setColor(white, black);

		// 허용 범위 : 0 < temp < 버튼의 갯수
		// 예외 : 없음
```

```
                int mode = exceptionHandling(inputMode, -1, 5, INT_MIN);

                if (mode == 0) {        // 종료
                        return;
                }
                else if ( mode == 1) {  // 가나다 순
                        sortString(checkOverflow());
                        system("cls");
                        printBasic();
                        printAllList();
                }
                else if (mode == 2) {   // 페이지 높은 순
                        sortPageDown(checkOverflow());
                        system("cls");
                        printBasic();
                        printAllList();
                }
                else if (mode == 3) {   // 페이지 낮은 순
                        sortPageUp(checkOverflow());
                        system("cls");
                        printBasic();
                        printAllList();
                }
                else if (mode == 4) {   // 가격 높은 순
                        sortPriceDown(checkOverflow());
                        system("cls");
                        printBasic();
                        printAllList();
                }
                else if (mode == 5) {   // 가격 높은 순
                        sortPriceUp(checkOverflow());
                        system("cls");
                        printBasic();
                        printAllList();
                }
        }
}
```

## findSting.h

```
#pragma once
#include "bookData.h"


using namespace std;

/*
	목적 : 이미 탐색해서 일치하지 않음을 안 인덱스를 뛰어넘음으로 시간 낭비를 줄임
	매개변수 : 입력값
	반환 : 입력 가능한 공간의 인덱스
*/
vector<int> getpos(string p) {
	int m = (int)p.size(), j = 0;
	vector<int> pos(m, 0);
	for (int i = 1; i < m; i++) {
		while (j > 0 && p[i] != p[j]) {
			j = pos[j - 1];
		}
		if (p[i] == p[j]) {
			pos[i] = ++j;
		}
	}
	return pos;
}

/*
	목적 : 이미 탐색해서 일치하지 않음을 안 인덱스를 뛰어넘음으로 시간 낭비를 줄임
	매개변수 : 비교 대상, 입력값
	반환 : 입력 가능한 공간의 인덱스
*/
vector<int> kmp(string s, string p) {
	vector<int> ans;
	auto pos = getpos(p);
	int n = (int)s.size(), m = (int)p.size(), j = 0;
	for (int i = 0; i < n; i++) {
		while (j > 0 && s[i] != p[j]) {
			j = pos[j - 1];
		}
		if (s[i] == p[j]) {
```

```
                        if (j == m - 1) {
                                ans.push_back(i - m + 1);
                                j = pos[j];
                        }
                        else {
                                j++;
                        }
                }
        }
        return ans;
}

/*
        목적 : 검색어와 일치하는 책 찾기
        매개변수 : 입력 검색어, 검색 타입( 책 이름으로 검색, 작가 이름으로 검색)
        반환 : 일치하는 값이 있는 경우 그 값의 인덱스를 matchPoint배열의 인덱스로 사용하여 그 값을 참으로 만들어줌.
*/
void getMatch(string input, int type) {
        for (int i = 0; i < sizeof(presenceOfInfo); i++) {
                matchPoint[i] = 0;
        }

        // 책 이름 검색시
        if (type == 1) {
                for (int i = 0; i < sizeof(presenceOfInfo); i++) {
                        auto match = kmp(LB[i].name, input);
                        matchPoint[i] = (int)match.size();
                }
        }
        // 작가 이름 검색시
        else if (type == 2) {
                for (int i = 0; i < sizeof(presenceOfInfo); i++) {
                        auto match = kmp(LB[i].author, input);
                        matchPoint[i] = (int)match.size();        // 사이즈
                }
        }
}
```

## searchBook.h

```
#include "bookData.h"
#include "findString.h"
#include "manageData.h"

#pragma once
using namespace std;

int search() {
	return 0;
}

void searchInfo() {

	// 검색 타입 출력
	printSerchType();

	string input;

	int type = 1;

	while(true) {

		cin >> input;

		// 허용 범위 : 0 < input <= 버튼 수
		// 예외 : 인트최솟값
		type = exceptionHandling(input, 0, 2, INT_MIN);
		if (type) {
			break;
		}
	}

	if (type == 1) {
		cout << "\t검색하실 ";
		setColor(lightYellow, black);
		cout << "도서의 이름";
		setColor(white, black);
		cout << "을 입력해주세요 : ";
	}
```

```
        else {
                cout << "\t검색하실 ";
                setColor(lightYellow, black);
                cout << "작가의 이름";
                setColor(white, black);
                cout << "을 입력해주세요 : ";
        }

        cin >> input;                       // 검색어 입력
        getMatch(input, type);  // 입력 도서 검색
        printBasic();

        int cnt = 0;
        for (int i = 0; i < sizeof(presenceOfInfo); i++) {
                // 검색어가 일치된 값은 출력
                if (matchPoint[i]) {
                        printBookInfo(i);
                }
                else {
                        cnt++;
                }
        }


        if (cnt == sizeof(presenceOfInfo)) {     // 검색 되지 않은 값이 도서 수와 같
은 경우 (책이 없음)
                cout << "\t입력한 도서가 존재하지 않습니다.\n";
        }
        else {
                cout << "\t총 ";
                setColor(lightYellow, black);
                cout << sizeof(presenceOfInfo) - cnt;
                setColor(white, black);
                cout << "건의 도서가 검색되었습니다.\n";
        }
}
```

## manageData.h

```
#include "bookData.h"
#include "printInformation.h"

void addBook();
void deleteBook();
int checkOverflow();
int findNull();
int exceptionHandling(string, int, int, int);

// 도서 추가
void addBook() {
	if (checkOverflow() != sizeof(presenceOfInfo)) {
		int index = findNull()-1;
		presenceOfInfo[index] = true;

		// 페이지수, 가격, 출간일, 책이름, 작가
		cout << "\n\n\t책 ";                                  // 입력 내용 출력
: 제목
		setColor(lightGreen, black);
		cout << "제목";
		setColor(white, black);
		cout << "을 입력해주세요. : ";

		cin.ignore(50, '\n');                          // 입력 : 제목
		setColor(lightSkyblue, black);
		getline(cin, LB[index].name);
		setColor(white, black);


		cout << "\n\t책의 ";                                  // 입력 내용 출력
: 저자
		setColor(lightGreen, black);
		cout << "저자";
		setColor(white, black);
		cout << "를 입력해주세요. : ";



		setColor(lightSkyblue, black);   // 입력 : 저자
		getline(cin, LB[index].author);
```

```
                setColor(white, black);

                string temp;                                    // 입력 내용 출력
: 출간일
                cout << "\n\n\t책의 ";
                setColor(lightGreen, black);
                cout << "출간일";
                setColor(white, black);
                cout    <<    "을    입력해주세요.모르는    경우    '1'을    입력해주세
요.\n\t(YYYYMMDD ex - 20190402) : ";



                while (true) {                                  // 입력 : 저자 ,
예외처리
                        setColor(lightSkyblue, black);
                        cin >> temp;
                        setColor(white, black);

                        // 허용 범위 : 표기 기준 (YYYYMMDD)가 가능한 정수 < temp
< 2060년 이전 기간
                        // 예외 : 날짜를 모르는 경우
                        int    output    =    exceptionHandling(temp,    9999999,
20600000, 1);
                        if (output) {
                                LB[index].publicationDate = output;
                                break;
                        }
                }

                cout << "\n\n\t책의 ";                  // 입력 내용 출력 : 가격
                setColor(lightGreen, black);
                cout << "가격(1000원 이상)";
                setColor(white, black);
                cout << "을 입력해주세요. 모르는 경우 '1'을 입력해주세요. : ";

                while (true) {                                  // 입력 : 가격 ,
예외처리
                        setColor(lightSkyblue, black);
                        cin >> temp;
                        setColor(white, black);
```

```
                        // 허용 범위 : 0 < temp < int최댓값
                        // 예외 : 가격을 모르는 경우
                        int output = exceptionHandling(temp, 999, INT_MAX, 1);
                        if (output) {
                                LB[index].price = output;
                                break;
                        }
                }

                cout << "\n\n\t책의 ";                    // 입력 내용 출력 : 페이지
수
                setColor(lightGreen, black);
                cout << "페이지 수";
                setColor(white, black);
                cout << "를 입력해주세요. 모르는 경우 '1'을 입력해주세요. : ";


                while (true) {                                  //  입력  :  페이지
수 , 예외처리
                        setColor(lightSkyblue, black);
                        cin >> temp;
                        setColor(white, black);
                        // 허용 범위 : 0 < temp < int최댓값
                        // 예외 : 페이지 수를 모르는 경우
                        int output = exceptionHandling(temp, 0, INT_MAX, 1);
                        if (output) {
                                LB[index].page = output;
                                break;
                        }
                }

                setColor(green, black);
                cout << "\n\n\n\t입력하신 정보가 저장되었습니다.";
                setColor(white, black);
        }
        else {
                setColor(red,black);
                cout << "\n\t입력 가능 공간이 부족합니다.\n\n";
                cout << "\t기존 책을 삭제 후 진행해주세요\n";
```

```
                setColor(white, black);
        }

}


/*
        목적 : 입력값 예외처리 및 인트형 변환
        매개변수 : 입력값, 허용 최솟값, 허용 최댓값, 예외 인정 값
        반환 :
                올바른 값 입력시 - 입력한 값을 인트로 변환한 값
                올바르지 않은 값 - flase
*/
int exceptionHandling(string input, int range1, int range2, int exception) {
        int mode = atoi(input.c_str());
        if ((range1 < mode && mode <= range2) || (mode == exception)) {
                return mode;
        }
        else {
                cout << "\n\t\t\t\t올바른 값을 입력해주세요.\n";
                cout << "\t\t\t\t\t재 입력 : ";
                return false;
        }
}

// 삭제
void deleteBook() {
        system("cls");
        string input;
        printDividingLine(80);
        cout << "목록에 있는 책 중 삭제할 책의 번호를 입력해주세요. : ";
        printDividingLine(80);
        printAllList();                                                 //    모든
책 목록 출력

        setColor(skyblue,black);
        cout << "삭제할 책의 번호 : ";
        int index;
        while(true) {                                                   // 입력 :
삭제할 책의 번호 , 예외처리
```

```
                cin >> input;
                setColor(white, black);

                // 허용 범위 : 0 < index < 책 리스트 목록 수
                // 예외 : 인트최댓값
                index = exceptionHandling(input, 0, sizeof(presenceOfInfo),
INT_MIN);       // 예외 처리
                if (index) {
                        if (presenceOfInfo[index-1]) {
                                break;
                        }
                        cout << "\n\t\t\t\t올바른 값을 입력해주세요.\n";
                        cout << "\t\t\t\t재 입력 : ";
                }
        }

        cout << "\n도서 ";                                        // 삭제 완료 알림
        setColor(lightPurple, black);
        cout << LB[index - 1].name;
        setColor(white, black);
        cout << "(이)가 삭제되었습니다.\n";

        (LB[index-1].name).clear();                    // 내용 삭제
        presenceOfInfo[index - 1] = false;      // 삭제 표기

}

// 오버플로우 체크
int checkOverflow() {
        int cnt = 0;
        for (int i = 0; i < sizeof(presenceOfInfo); i++) {

                if ((LB[i].name).length() != 0) {  // 내용이 비어있지 않으면
                        presenceOfInfo[i] = true;               //  내용이  있음을
표기
                }
                else {
// 그렇지 않다면 (내용이 없다면)
                        presenceOfInfo[i] = false;              //  내용이  없음을
표기
```

```
                }

                if (presenceOfInfo[i]) {                    // 내용이 있다면 카운트 증
가
                        cnt++;
// 보유 책 체크용
                }
        }
        if (cnt == sizeof(presenceOfInfo)) {     // 내용을 모두 채움
                return cnt;
        }
        return cnt;
}


/*
        목적 : 책을 입력할 공간 찾기
        매개변수 :
        반환 : 입력 가능한 공간의 인덱스
*/
int findNull() {
        for (int i = 0; i < sizeof(presenceOfInfo); i++) {
                if (!presenceOfInfo[i]) {
                        return i+1;
                }
        }
        return sizeof(presenceOfInfo);
}
```

## printInformation.h

```
#include "bookData.h"

#pragma once
#define SHORTLINE 23
#define LONGLINE 80

using namespace std;

void printDividingLine(int);
void printBookInfo(int);
void printMode(int);
void printMode(int);
void printAllList();
void exit();

// 검색 형식 출력
void printSerchType() {
	system("cls");
	printDividingLine(SHORTLINE + 7);
	cout << "\t원하시는 검색 형식의 번호를 입력해주세요";
	printDividingLine(SHORTLINE + 7);
	cout << "\t\t1. 책 이름으로 검색\n\n";
	cout << "\t\t2. 작가 이름으로 검색\n\n";
}

// 메인 메뉴 출력
// 매개 변수 : 현재 보유 도서 수
void printMode(int book) {
	printDividingLine(SHORTLINE);
	setColor(skyblue, black);
	cout << "\t\t즐겨 찾기 목록\n\n";
	setColor(white, black);
	cout << "\t\t담긴 도서 수 : ";
	setColor(lightYellow, black);
	cout << book;
	setColor(white, black);
	printDividingLine(SHORTLINE);

	cout << "\t\t1) 도서 검색\n\n";
```

```
        cout << "\t\t2) 도서 추가\n\n";
        cout << "\t\t3) 도서 삭제\n\n";
        cout << "\t\t4) 모든 도서 보기\n\n\n\n";

        cout << "\t0이용할 기능의 ";
        setColor(lightYellow, black);
        cout << "번호";
        setColor(white, black);
        cout << "를 입력해주세요.\n\n";
        cout << "\t>> ";

}

// 목록 구분 출력
void printBasic() {
        system("cls");
        printDividingLine(LONGLINE);
        cout << "\t\t도서 목록";
        printDividingLine(LONGLINE);
        setColor(skyblue, black);
        cout << "    도서 번호"
                << "            도서 이름"
                << "\t\t\t\t\t\t       저자"
                << "\t\t  페이지"
                << "\t\t가격"
                << "\t\t     출간일" << endl << endl;
        setColor(white, black);
}

// 책 정보 출력
// 매개변수 : 출력할 정보의 인덱스
void printBookInfo(int index) {

        cout.width(7);

        cout << "    " << left << setw(15)
                << index + 1 << left << setw(55)
                << (LB + index)->name << left << setw(20)
                << (LB + index)->author << left << setw(20);
```

```
	int data;

	data = LB[index].page;
	if (data != 1) {
		cout << data << left << setw(20);
	}
	else {
		cout << "( 미기입 )" << left << setw(20);
	}

	data = LB[index].price;
	if(data != 1) {
		cout << data;
	}
	else {
		cout << "( 미기입 )";
	}

	data= LB[index].publicationDate;
	if (data != 1) {
		cout << data / 10000 << "년 "
			<< (data % 10000) / 100 << "월 "
			<< (data % 100) << "일" << endl;
	}
	else {
		cout << "( 미기입 )" << endl;
	}
	cout << endl;
}

// 구분선 출력
void printDividingLine(int lineLength) {
	setColor(red,black);
	cout << "\n\n";
	while (lineLength--) {
		cout << "♥";
	}
	cout << "\n\n";
	setColor(white, black);
}
```

```
// 입력된 모든 리스트 출력
void printAllList() {
	library LB[10];
	for (int i = 0; i < sizeof(presenceOfInfo); i++) {
		if (presenceOfInfo[i]) {
			printBookInfo(i);
		}
	}
}

// 종료
void exit() {
	cout << "\n\n >> 키 입력시 메인 화면으로 돌아갑니다....";
	cin.ignore(50, '\n');
	cin.get();
}
```

## bookData.h

```
#include <iostream>
#include <iomanip>
#include <Windows.h>
#include <cstring>
#include <string>
#include <vector>
#include <limits>

#define endl "\n"

#pragma once

bool presenceOfInfo[10];
int matchPoint[sizeof(presenceOfInfo)];

// 색상 정의
enum color {
	black, blue, green, skyblue, red,
	purple, yellow, white, grey, lightBlue,
	lightGreen, lightSkyblue, lightRed,
	lightPurple, lightYellow, darkWhite
};

// 색 설정
void setColor(int color, int bgcolor) {

	color &= 0xf;
	bgcolor &= 0xf;
	SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE),  (bgcolor
<< 4) | color);

}

// 도서 정보 구조체
struct library {
	int page;
	int price;
	int publicationDate;
	std::string name;
```

```
        std::string author;
};

// 기본 입력값
library LB[10] = {
                {300, 12500, 20191130, "김한규의 국가(2019)", "김한규"},
                {348, 48200, 20130325, "이상한 나라의 밍키", "김밍키"},
                {172, 14500, 20190126, "현수막개론", "낙타"},
                {1568, 18700, 20040211, "잠 못 이루는 협곡에", "승미"},
                {593, 34000, 20080402, "진주의 파티플랜", "나방신"},
                {120, 8000, 1, "경주, 역사의 중심에 서다. - 개정판", "동건"},
                {312, 11000, 20170328, "단체로 무친 사회 - 반사회적 행동의 목격",
"이승훈"},
};
```

## sortData.h

```
#include "bookData.h"

// 페이지수, 가격, 출간일, 책이름, 작가, 출판사
/*

	정렬
		- 오름? 내림?
			- 오름 : 가나다, 페이지, 가격
			- 내림 : 가격, 페이지 (미정)

		- 데이터 비교 함수
			- 복사본 가져 와서 인덱스로 인덱스
			- 인덱스 반환

		- 데이터 변경 함수
			- 주소값을 통해 변경
			- 책이름(string)
			- 작가  (string)
			- 페이지수(int)
			- 출간일(int)
			- 가격  (int)

*/


void changeValue(int);

// 목록 중간에 삭제돼서 빈 것이 있으면 소팅 어떻게 해요?
void sortPageDown(int size) {

	// for문 크기 : checkOverflow
	// 삭제 체크 : presenceOfInfo
	// 값 변경 : 인덱스 전송!

	for (int i = 0; i < size; i++) {
		int temp = 0;
		for (int k = 0; k < size; ++k) {
			for (int a = 0; a < size - 1; ++a) {
				if (!presenceOfInfo[a + 1] || !presenceOfInfo[a]) {
```

```
                                        continue;
                                }
                                if (LB[a + 1].page > LB[a].page) {
                                        changeValue(a);
                                }
                        }
                }
        }
}

void sortPageUp(int size) {

        // for문 크기 : checkOverflow
        // 삭제 체크 : presenceOfInfo
        // 값 변경 : 인덱스 전송!

        for (int i = 0; i < size; i++) {
                int temp = 0;
                for (int k = 0; k < size; ++k) {
                        for (int a = 0; a < size - 1; ++a) {
                                if (!presenceOfInfo[a + 1] || !presenceOfInfo[a]) {
                                        continue;
                                }
                                if (LB[a + 1].page < LB[a].page) {
                                        changeValue(a);
                                }
                        }
                }
        }
}

void sortPriceDown(int size) {

        // for문 크기 : checkOverflow
        // 삭제 체크 : presenceOfInfo
        // 값 변경 : 인덱스 전송!

        for (int i = 0; i < size; i++) {
                int temp = 0;
                for (int k = 0; k < size; ++k) {
```

```
                        for (int a = 0; a < size - 1; ++a) {
                                if (!presenceOfInfo[a + 1] || !presenceOfInfo[a]) {
                                        continue;
                                }
                                if (LB[a + 1].price > LB[a].price) {
                                        changeValue(a);
                                }
                        }
                }
        }
}

void sortPriceUp(int size) {

        // for문 크기 : checkOverflow
        // 삭제 체크 : presenceOfInfo
        // 값 변경 : 인덱스 전송!

        for (int i = 0; i < size; i++) {
                int temp = 0;
                for (int k = 0; k < size; ++k) {
                        for (int a = 0; a < size - 1; ++a) {
                                if (!presenceOfInfo[a + 1] || !presenceOfInfo[a]) {
                                        continue;
                                }
                                if (LB[a + 1].price < LB[a].price) {
                                        changeValue(a);
                                }
                        }
                }
        }
}



void sortString(int size) {

        for (int i = 0; i < size; i++) {
                int temp = 0;
                for (int k = 0; k < size; ++k) {
```

```
                        for (int a = 0; a < size - 1; ++a) {
                                if (!presenceOfInfo[a + 1] || !presenceOfInfo[a]) {
                                        continue;
                                }
                                if (LB[a].name.compare(LB[a + 1].name) > 0) {
                                        changeValue(a);
                                }
                        }
                }
        }
}

// 페이지수, 가격, 출간일, 책이름, 작가, 출판사
void changeValue(int index) {
        std::string tempS;
        int tempI;

        // 페이지 값 교체
        tempI = LB[index].page;
        LB[index].page = LB[index + 1].page;
        LB[index + 1].page = tempI;

        // 가격 값 교체
        tempI = LB[index].price;
        LB[index].price = LB[index + 1].price;
        LB[index + 1].price = tempI;

        // 출간일 값 교체
        tempI = LB[index].publicationDate;
        LB[index].publicationDate = LB[index + 1].publicationDate;
        LB[index + 1].publicationDate = tempI;

        // 책이름 값 교체
        tempS = LB[index].name;
        LB[index].name = LB[index + 1].name;
        LB[index + 1].name = tempS;

        // 작가 값 교체
        tempS = LB[index].author;
        LB[index].author = LB[index + 1].author;
```

```
        LB[index + 1].author = tempS;
}
```