

쉽게 사용할 수 있고 정보 안정성이 높은

재고관리 프로그램

2089055 조창희





목차



01. 프로그램 소개



02. 모드 소개



03. 시연



04. 코드설명



05. 기대 효과



프로그램 소개



재고관리 프로그램

```
C:\Users\조창희\source\repos\Onefile\Debug\Onefile.exe
배재상사 Fast System v1.0
=====
1. 재고 검색
2. 전체 재고 현황
3. 재고 정보 수정
4. 재고 정보 추가
5. 재고 정보 삭제
6. 재고 개수 수정
7. 종료
=====
정수값을 입력하시오: _
```

- 수 많은 양의 재고를 전산화 시켜 효율적으로 관리할 수 있는 프로그램
- 직관적인 구조와 간편하게 관리할 수 있는 설계로 누구나 어려움 없이 재고관리 현황을 등록·편집·삭제할 수 있음
- 프로그램이 종료되어도 파일이 저장되어 있어 다시 실행이 가능하여 안정성을 높임
- 편집, 삭제 시 해당 재고의 정보를 보여주고 수정할 것인지 다시 의사를 물음으로써 실수 방지 및 정보 안정성 향상
- 상품번호를 Primary Key로 보고, 중복된 상품번호가 있다면 등록을 거절해 정보 신뢰성을 높임



모드 소개



1. 재고 검색

```
1. 상품번호로 검색
2. 상품명으로 검색
=====
정수값을 입력하시오: 1
=====
검색할 상품번호: 1001
상품번호: 1001
상품명: 더블에이
단가: 5000
위치: D-30
개수: 5
=====
```

```
검색할 상품번호: 10012
10012의 데이터를 찾을 수 없습니다.
```

검색한 정보의 자료가 없는 경우

```
정수값을 입력하시오: 3
=====
올바른 명령어를 다시 입력해 주세요.
1. 상품번호로 검색
2. 상품명으로 검색
=====
정수값을 입력하시오:
```

명령어를 제대로 입력하지 않은 경우

- 상품번호나 상품명으로 등록된 정보를 확인할 수 있는 기능
- 등록된 정보가 없으면 데이터를 찾을 수 없다는 메시지를 띄움
- 명령어가 바르지 않다면 다시 명령어를 입력하도록 요청



모드 소개



2. 전체 재고 현황

전체 재고 현황입니다.

| 순번 | 상품번호 | 상품명 | 단가 | 수량 | 위치 |
|----|-------|-----------|-------|----|------|
| 1 | 1001 | 더블에이 | 5000 | 5 | D-30 |
| 2 | 1002 | 아두이노 알람시계 | 10000 | 1 | A-10 |
| 3 | 2003 | 풀 | 700 | 40 | B-10 |
| 4 | 20021 | 지우개 | 500 | 20 | B-10 |

총 4 종류의 상품의 데이터 출력 완료

1개 이상의 데이터가 등록되어 있을 시 출력 현황

- 등록된 자료를 표로 보여주어 가독성을 높임
- 중간에 자료가 삭제되면 순번은 자동으로 당겨져서 반영됨
- 등록된 데이터가 전혀 없으면 **출력할 데이터가 없습니다.** 메시지를 띄움



모드 소개



3. 재고 정보 수정

수정할 상품번호를 입력해 주세요.: 1001

=====

수정할 상품의 현재 정보는 다음과 같습니다.

상품번호: 1001

상품명: 더블에이

단가: 5000

위치: D-30

개수: 5

=====

수정을 진행하겠습니까?(예:1, 아니오:2):

수정을 진행하겠습니까?(예:1, 아니오:2): 1

수정하고자 하는 상품번호: 1003

수정하고자 하는 상품명: 더블비

수정하고자 하는 단가: 6000

수정하고자 하는 위치: D-30

수정하고자 하는 개수: 6

성공적으로 수정되었습니다.

수정이 성공적으로 진행된 경우

(정보가 등록되어 있지 않다면, 데이터가 없다고 뜸)

- 수정할 상품번호를 누르면 먼저 그에 대한 정보를 출력해 주고, 수정할 것인지 의사를 묻음
- 수정을 취소한다면 **수정이 취소되었습니다.** 메시지를 출력함.
- 수정을 하면 (2)전체 재고 현황에서는 순번이 그대로 유지된 채 재고의 정보만 바뀜



모드 소개



4. 재고 정보 추가

```
상품번호: 1234  
상품명: 스케치북  
가격: 1000  
위치: C-10  
개수: 3  
저장되었습니다!
```

```
상품번호: 1234  
상품명: 스텐드  
가격: 12000  
위치: A-10  
개수: 2  
이미 같은 상품번호로 등록된 상품이 있어 등록할 수 없습니다.
```

상품번호가 Primary Key의 역할을 하기 때문에 같은 상품번호는 등록 불가

- 상품에 대한 정보를 입력해 줌으로써 데이터를 저장 가능
- 상품번호가 Primary Key의 역할을 하기 때문에 다른 정보는 달라도 같은 상품번호는 등록할 수 없음



모드 소개



5. 재고 정보 삭제

```

삭제할 상품번호를 입력해 주세요.: 1234
=====
삭제할 상품의 현재 정보는 다음과 같습니다.
상품번호: 1234
상품명: 스케치북
단가: 1000
위치: C-10
개수: 3
=====
삭제를 진행하겠습니까?(예:1, 아니오:2):
  
```

```

삭제를 진행하겠습니까?(예:1, 아니오:2): 1
성공적으로 삭제되었습니다.
  
```

```

삭제를 진행하겠습니까?(예:1, 아니오:2): 2
삭제가 취소되었습니다.
  
```

삭제 여부를 묻고, 다른 정보이거나
잘못 입력했을 시 취소 가능

- 재고관리 특성상 재고 정보는 유지하되, 개수만 수정해야 할 일이 많음
- 개수만 수정할 상품번호를 입력하면 그 상품번호의 정보를 먼저 띄어 주고 삭제 여부를 묻음, 개수는 늘이거나 줄일 수 있음
- 업무를 효율화하고 다시 한 번 정보확인과 여부를 묻음으로써 정보의 안정성 확보



모드 소개



6. 재고 개수 수정

```
수정할 상품번호를 입력해 주세요.: 1003
=====
개수를 수정할 상품의 현재 정보는 다음과 같습니다.
상품번호: 1003
상품명: 더블비
단가: 6000
위치: D-30
개수: 6
=====
수정을 진행하겠습니까?(예:1, 아니오:2): 1
```

```
수정을 진행하겠습니까?(예:1, 아니오:2): 1
수정하고자 하는 개수: 10
성공적으로 수정되었습니다.
```

예를 입력하면 개수만 입력 받고 수정한다.

```
수정을 진행하겠습니까?(예:1, 아니오:2): 2
수정이 취소되었습니다.
```

아니오를 입력하면 수정을 취소한다.

- 삭제할 상품번호를 입력하면 그 상품번호의 정보를 먼저 띄어 주고 삭제 여부를 물음
- 상품번호는 Primary Key의 역할이고, 상품의 정보와 삭제 여부를 한 번 더 띄우면서 정보 안정성을 높임



시연



코드 설명(전처리기와 main 함수)

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <string.h>

#define SIZE 100

#define COMPANY_NAME "배재상사"
#define PROGRAM_NAME "Fast System"
#define PROGRAM_VER "v1.0"

typedef struct item {
    char number[SIZE]; // 상품번호
    char name[SIZE]; // 상품명
    int price; // 금액
    char place[SIZE]; // 위치
    int count; // 개수
} ITEM;

void menu();
ITEM get_record();
void print_record(ITEM data);
void add_record(FILE* fp);
void search_record(FILE* fp);
void update_record(FILE* fp);
void load_all_record(FILE* fp);
void remove_record(FILE* fp);
void update_only_count(FILE* fp);
```

```
int main(void)
{
    FILE* fp;
    int select;
    if ((fp = fopen("items.dat", "a+")) == NULL) {
        fprintf(stderr, "정보가 입력된 파일을 열 수 없습니다.");
        exit(1);
    }
}
```

```
#ifdef COMPANY_NAME
    printf("%s ", COMPANY_NAME);
#endif

#ifdef PROGRAM_NAME
    printf("%s", PROGRAM_NAME);
#else
    printf("재고관리 시스템");
#endif

#ifdef PROGRAM_VER
    printf(" %s\n", PROGRAM_VER);
#else
    printf("\n");
#endif
```

회사명, 프로그램명, 버전별로 바로 상단에 표시할 수 있도록
전처리기 지시어를 사용해 줌
(정의되어 있지 않으면 재고관리 시스템이 뜸)

배재상사 Fast System v1.0

```
while (1) {
    menu();
    printf("정수값을 입력하시오: ");
    scanf("%d", &select);
    printf("=====\n");

    switch (select) {
        case 1: //재고 검색
            search_record(fp);
            break;
        case 2: // 재고 전체 현황
            load_all_record(fp);
            break;
        case 3: // 재고 수정
            update_record(fp);
            break;
        case 4: // 재고 추가
            add_record(fp);
            break;
        case 5: // 재고 정보 삭제
            remove_record(fp);
            break;
        case 6:
            update_only_count(fp);
            break;
        case 7: // 종료
            fclose(fp);
            return 0;
    }

    if (!(select > 0 && select < 8))
        printf("올바른 명령어를 입력해 주세요!\n");
}
```



코드 설명(정보를 입출력하는 함수, 메뉴)

```
ITEM get_record()
```

```
{  
    ITEM data;  
    getchar();  
    printf("상품번호: ");  
    gets_s(data.number, SIZE);  
    printf("상품명: ");  
    gets_s(data.name, SIZE);  
    printf("가격: ");  
    scanf("%d", &data.price);  
    getchar();  
    printf("위치: ");  
    gets_s(data.place, SIZE);  
    printf("개수: ");  
    scanf("%d", &data.count);  
    getchar();  
    return data;  
}
```

```
void menu()
```

```
{  
    printf("=====\n");  
    printf(" 1. 재고 검색\n 2. 전체 재고 현황\n 3. 재고 정보 수정\n 4. 재고 정보 추가\n 5. 재고 정보 삭제\n 6. 재고 개수 수정\n 7. 종료\n");  
    printf("=====\n");  
}
```

데이터를 출력할 때

데이터를 입력 받을 때

다른 함수에서
호출되어 양식으로
사용됨

```
void print_record(ITEM data)
```

```
{  
    printf("상품번호: %s\n", data.number);  
    printf("상품명: %s\n", data.name);  
    printf("단가: %d\n", data.price);  
    printf("위치: %s\n", data.place);  
    printf("개수: %d\n", data.count);  
}
```

```
상품번호: 1002  
상품명: 아두이노 알람시계  
단가: 10000  
위치: 1  
개수: 4
```



코드 설명(재고 추가)

```
void add_record(FILE* fp)
```

```
{  
    ITEM getdata;  
    ITEM data;  
    getdata = get_record();  
    FILE* fp1;  
    fseek(fp, 0, SEEK_SET);  
    int status = 0;  
  
    if ((fp1 = fopen("temp.dat", "w")) == NULL)  
    {  
        fprintf(stderr, "파일 temp.dat를 열 수 없습니다");  
        exit(1);  
    }  
}
```

```
while (fread(&data, sizeof(data), 1, fp) == 1) {  
    if (strcmp(data.number, getdata.number) == 0)  
    {  
        printf("이미 같은 상품번호로 등록된 상품이 있어 등록할 수 없습니다.\n");  
        status = 1;  
    }  
    fwrite(&data, sizeof(data), 1, fp1);  
}  
if (status == 0)  
{  
    fwrite(&getdata, sizeof(getdata), 1, fp1);  
    printf("저장되었습니다!\n");  
}  
fclose(fp1);  
fclose(fp);
```

```
fclose(fp1);  
fclose(fp);
```

```
remove("items.dat");  
rename("temp.dat", "items.dat");
```

```
if ((fp = fopen("items.dat", "a+")) == NULL)  
{  
    fprintf(stderr, "삭제 처리 후 파일 items.dat를 열 수 없습니다");  
    exit(1);  
}
```

temp.dat에 items.dat를 집어 넣은 후
기존 items.dat를 삭제하고
temp.dat를 items.dat로 바꿔 줌

주 저장 파일(items.dat)에서 1개의 재고 데이터를
매번 받아와 입력한 양식과 비교하며 임시 저장소인
temp.dat에 저장

같은 상품번호가 있으면 입력한 파일을 저장 못하게
상태변수 지정(상품번호: Primary Key)



코드 설명(재고 검색)

```
void search_record(FILE* fp)
{
    char name[SIZE];
    ITEM data;
    int status = 0;

    fseek(fp, 0, SEEK_SET);
    getchar();
    while (1)
    {
        int user = 0;
        printf("1. 상품번호로 검색\n2. 상품명으로 검색\n");
        printf("=====\n");
        printf("정수값을 입력하시오: ");
        scanf("%d", &user);
        getchar();
        printf("=====\n");
        if (user == 1)
        {
            printf("검색할 상품번호: ");
            gets_s(name, SIZE);
            while (!feof(fp)) {
                fread(&data, sizeof(data), 1, fp);
                if (strcmp(data.number, name) == 0) {
                    print_record(data);
                    status = 1;
                    break;
                }
            }
            break;
        }
    }
}
```

```
else if (user == 2)
{
    printf("검색할 상품명: ");
    gets_s(name, SIZE);
    while (!feof(fp)) {
        fread(&data, sizeof(data), 1, fp);
        if (strcmp(data.name, name) == 0) {
            print_record(data);
            status = 1;
            break;
        }
    }
    break;
}
else
    printf("올바른 명령어를 다시 입력해 주세요.\n");
}
if (status == 0)
    printf("%s의 데이터를 찾을 수 없습니다.\n", name);
}
```

처음에 fseek를 이용해 파일 처음에 커서를 맞춘 후 한 개의 자료 크기만큼 호출해서 비교



코드 설명(재고 검색-1)

```
void update_record(FILE* fp)
{
    char name[SIZE] = "";
    ITEM data;
    FILE* fp1;
    int user = 0;
    int user_status = 0;
    int status = 0;
    if ((fp1 = fopen("temp.dat", "w")) == NULL)
    {
        fprintf(stderr, "파일 temp.dat를 열 수 없습니다");
        exit(1);
    }
    getchar();
    printf("수정할 상품번호를 입력해 주세요.: ");
    gets_s(name, SIZE);

    fseek(fp, 0, SEEK_SET);

    while (fread(&data, sizeof(data), 1, fp) == 1) {
        if (strcmp(data.number, name) == 0)
        {
            if (user_status == 0)
            {
                printf("=====\n");
                printf("수정할 상품의 현재 정보는 다음과 같습니다.\n");
                print_record(data);
                printf("=====\n");
                while (user_status == 0)
                {
                    printf("수정을 진행하겠습니까?(예:1, 아니오:2): ");
                    scanf("%d", &user);
                    getchar();
```

```
                if (user == 1 || user == 2)
                {
                    user_status = 1;
                    break;
                }
                else
                    printf("올바른 명령어를 다시 입력해 주세요!\n");
            }
        }
        if (user == 1 && status == 0)
        {
            printf("수정하고자 하는 상품번호: ");
            gets_s(data.number, SIZE);
            printf("수정하고자 하는 상품명: ");
            gets_s(data.name, SIZE);
            printf("수정하고자 하는 단가: ");
            scanf("%d", &data.price);
            getchar();
            printf("수정하고자 하는 위치: ");
            gets_s(data.place, SIZE);
            printf("수정하고자 하는 개수: ");
            scanf("%d", &data.count);
            status = 1;
        }
        else if (user == 2)
            status = 2;
    }
    fwrite(&data, sizeof(data), 1, fp1);
}
```

파일에서 한 개씩 읽어 상품번호를 비교 후
맞는 자료가 있으면 데이터 출력



코드 설명(재고 검색-2)

```
if (status == 1)
    printf("성공적으로 수정되었습니다.\n");
else if (status == 2)
    printf("수정이 취소되었습니다.\n");
else
    printf("수정할 데이터를 찾을 수 없습니다. %s은(는) 저장되어 있지 않습니다.\n", name);

fclose(fp1);
fclose(fp);

remove("items.dat");
rename("temp.dat", "items.dat");

if ((fp = fopen("items.dat", "a+")) == NULL)
{
    fprintf(stderr, "수정 처리 후 파일 items.dat를 열 수 없습니다");
    exit(1);
}
}
```

```
검색할 상품번호: 10012
10012의 데이터를 찾을 수 없습니다.
```




코드 설명(전체 재고 현황)

```
void load_all_record(FILE* fp)
{
    int id = 0;
    int task = 0;
    int status = 0;
    ITEM data;
    fseek(fp, 0, SEEK_SET);

    while (!feof(fp) && fread(&data, sizeof(data), 1, fp) == 1)
    {
        if (task == 0)
        {
            printf("전체 재고 현황입니다.\n");
            printf("===== \n");
            printf(" | 순번 | 상품번호 | 상품명 | 단가 | 수량 | 위치 | \n");
            printf(" |-----+-----+-----+-----+-----+-----+ \n");
            task = 1;
        }
        id++;
        printf(" | %-5d | %-16s | %-26s | %-8d | %-10d | %-11s | \n", id, data.number, data.name, data.price, data.count, data.place);
        status = 1;
    }

    if (status == 1)
    {
        printf(" |-----+-----+-----+-----+-----+-----+ \n");
        printf("총 %d 종류의 상품의 데이터 출력 완료\n", id);
    }
    else
        printf("출력할 데이터가 없습니다.\n");
}
```

순번은 저장되어 있는 순서대로 지정
(수정시에는 안 변하나, 삭제시에는 바뀜)

데이터 출력의 표시형식을 지키기 위해
%-5d 같은 형식을 사용함

전체 재고 현황입니다.

| 순번 | 상품번호 | 상품명 | 단가 | 수량 | 위치 |
|----|-------|-----------|-------|----|------|
| 1 | 1001 | 더블에이 | 5000 | 5 | D-30 |
| 2 | 1002 | 아두이노 알람시계 | 10000 | 1 | A-10 |
| 3 | 2003 | 풀 | 700 | 40 | B-10 |
| 4 | 20021 | 지우개 | 500 | 20 | B-10 |

총 4 종류의 상품의 데이터 출력 완료



코드 설명(재고 정보 삭제)

```
void remove_record(FILE* fp)
{
    char name[SIZE] = "";
    ITEM data;
    FILE* fp1;
    int status = 0;
    int user_status = 0;
    int user = 0;
    if ((fp1 = fopen("temp.dat", "w")) == NULL)
    {
        fprintf(stderr, "파일 temp.dat를 열 수 없습니다.");
        exit(1);
    }
    fseek(fp, 0, SEEK_SET);
    getchar();
    printf("삭제할 상품번호를 입력해 주세요.: ");
    gets_s(name, SIZE);

    while (fread(&data, sizeof(data), 1, fp) == 1)
    {
        if (strcmp(data.number, name) == 0)
        {
            printf("=====\n");
            printf("삭제할 상품의 현재 정보는 다음과 같습니다.\n");
            print_record(data);
            printf("=====\n");
            while (user_status == 0)
            {
                printf("삭제를 진행하겠습니까?(예:1, 아니오:2): ");
                scanf("%d", &user);
                getchar();
                if (user == 1)
                {
                    status = 1;
                    user_status = 1;
                    break;
                }
            }
        }
    }
}
```

```
        else if (user == 2)
        {
            status = 2;
            user_status = 1;
            fwrite(&data, sizeof(data), 1, fp1);
            break;
        }
        else
            printf("올바른 명령어를 다시 입력해 주세요!\n");
    }
    else
        fwrite(&data, sizeof(data), 1, fp1);
}

if (status == 1)
    printf("성공적으로 삭제되었습니다.\n");
else if (status == 2)
    printf("삭제가 취소되었습니다.\n");
else
    printf("삭제할 데이터를 찾을 수 없습니다. %s은(는) 저장되어 있지 않습니다.\n", name);

fclose(fp1);
fclose(fp);

remove("items.dat");
rename("temp.dat", "items.dat");

if ((fp = fopen("items.dat", "a+")) == NULL)
{
    fprintf(stderr, "삭제 처리 후 파일 items.dat를 열 수 없습니다.");
    exit(1);
}
```



코드 설명(재고 개수 수정-1)

```
void update_only_count(FILE* fp)
{
    char name[SIZE] = "";
    ITEM data;
    FILE* fp1;
    int user = 0;
    int user_status = 0;
    int status = 0;
    if ((fp1 = fopen("temp.dat", "w")) == NULL)
    {
        fprintf(stderr, "파일 temp.dat를 열 수 없습니다.");
        exit(1);
    }
    getchar();
    printf("수정할 상품번호를 입력해 주세요.: ");
    gets_s(name, SIZE);

    fseek(fp, 0, SEEK_SET);

    while (fread(&data, sizeof(data), 1, fp) == 1) {
        if (strcmp(data.number, name) == 0)
        {
            if (user_status == 0)
            {
                printf("=====\n");
                printf("개수를 수정할 상품의 현재 정보는 다음과 같습니다.\n");
                print_record(data);
                printf("=====\n");
                while (user_status == 0)
                {
                    printf("수정을 진행하겠습니까?(예:1, 아니오:2): ");
                    scanf("%d", &user);
                    getchar();
```

```
                    if (user == 1 || user == 2)
                    {
                        user_status = 1;
                        break;
                    }
                    else
                        printf("올바른 명령어를 다시 입력해 주세요!\n");
                }
            }
            if (user == 1 && status == 0)
            {
                printf("수정하고자 하는 개수: ");
                scanf("%d", &data.count);
                status = 1;
            }
            else if (user == 2)
                status = 2;
        }
        fwrite(&data, sizeof(data), 1, fp1);
    }
    if (status == 1)
        printf("성공적으로 수정되었습니다.\n");
    else if (status == 2)
        printf("수정이 취소되었습니다.\n");
    else
        printf("수정할 데이터를 찾을 수 없습니다. %s은(는) 저장되어 있지 않습니다.\n", name);
}
```

입력된 재고의 정보를 받아온
상태에서 재고 개수만
대입해 주고 파일에 써줌



코드 설명(재고 개수 수정-2)

```
fclose(fp1);  
fclose(fp);  
  
remove("items.dat");  
rename("temp.dat", "items.dat");  
  
if ((fp = fopen("items.dat", "a+")) == NULL)  
{  
    fprintf(stderr, "수정 처리 후 파일 items.dat를 열 수 없습니다");  
    exit(1);  
}  
}
```

```
수정을 진행하겠습니까?(예:1, 아니오:2): 1  
수정하고자 하는 개수: 10  
성공적으로 수정되었습니다.
```



기대 효과

- 직관적이어서 누구나 빠르게 사용할 수 있고, 효율적으로 수많은 재고들을 관리할 수 있다.
- 수정, 삭제를 실행하기 전에 해당하는 재고에 대한 정보를 보여주고, 다시 여부를 물음으로써 실수로 인한 데이터 손실을 예방할 수 있다.
- 상품번호를 Primary Key 처럼 활용함으로써 중복 등록을 예방해 데이터의 객관성을 유지할 수 있다.
- 상품의 개수만을 수정하는 기능으로 빠르게 재고 현황을 수정, 반영할 수 있다.
- 전체 재고 현황표를 보며 현재 남아있는 재고와 더 주문해야 할 재고를 미리 준비 할 수 있다.
- 매 작업 시마다 데이터 파일이 저장되고 있어 애기치 못한 자료 손실을 예방할 수 있다.



감사합니다