

File

C 중급 세미나 - 김혜윤 -

Contents

1. 파일 분리하기
 - 컴파일 과정 / 전처리기 / 헤더 파일
2. main 함수 인자
3. 파일 입출력
 - 스트림 / 파일 입출력 / 파일 위치 지시자 / fscanf 함수
4. 실습

파일 분리하기

컴파일 과정

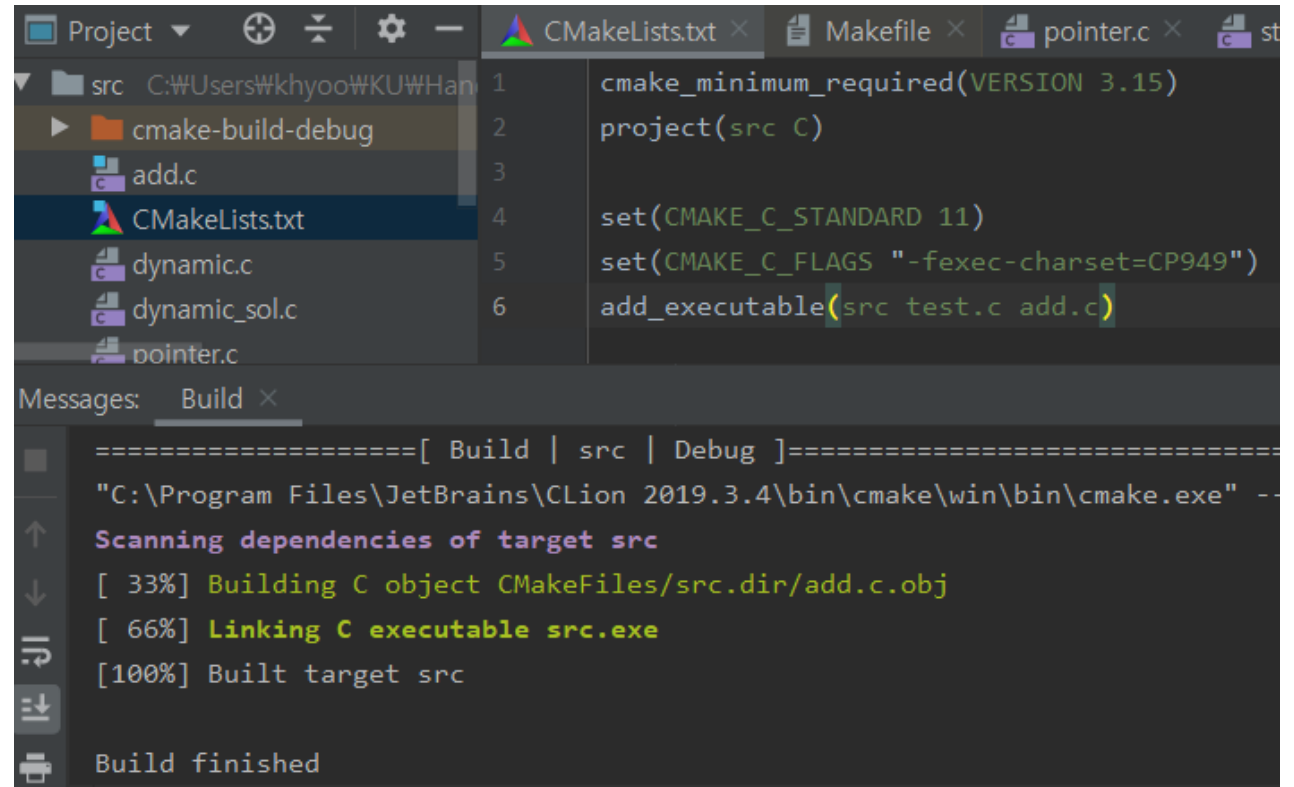
test.c

```
#include <stdio.h>
int add(int a, int b);

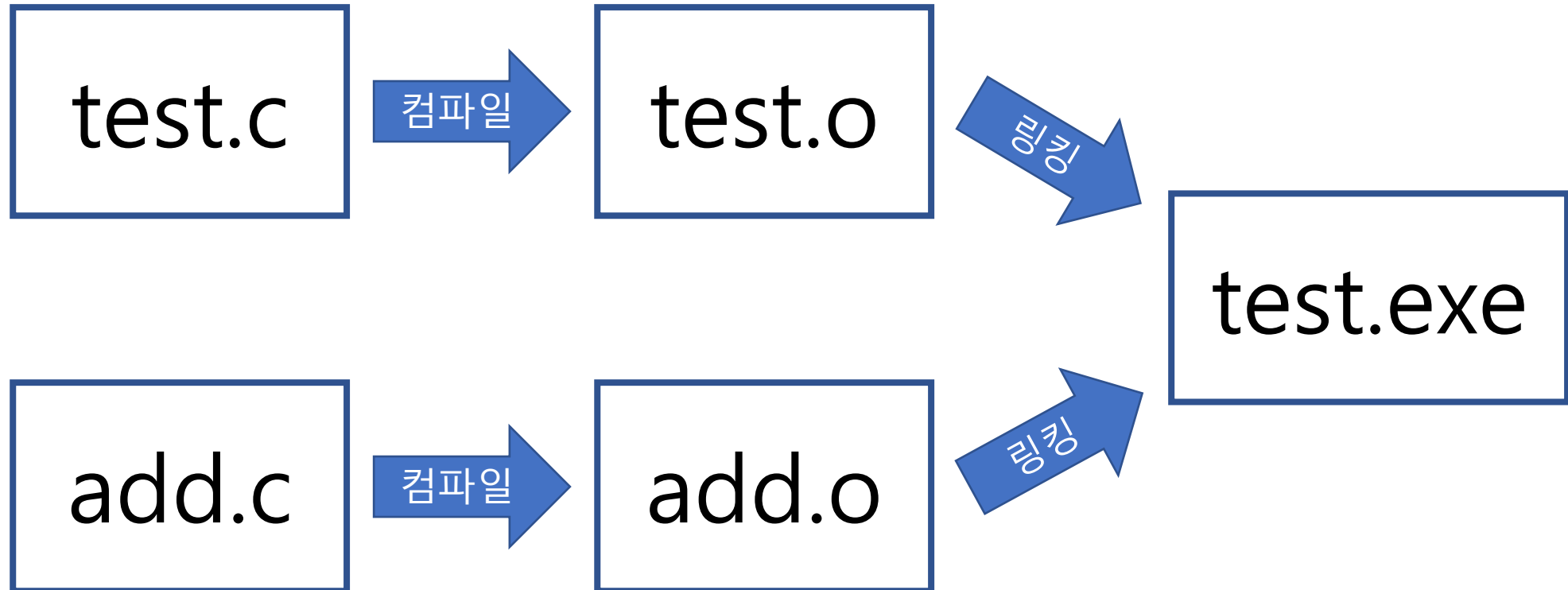
int main() {
    printf("%d", add(3,5));
    return 0;
}
```

add.c

```
int add(int a, int b){
    return a + b;
}
```



컴파일 과정



전처리기 / 헤더 파일

test.c

```
#include <stdio.h>
#include "add.h"

int main() {
    printf("%d", add(3,5));
    return 0;
}
```

add.h

```
int add(int a, int b);
```

add.c

```
int add(int a, int b){
    return a + b;
}
```

- 컴파일 전 전처리기에서 #명령어 처리
- 헤더파일 내용 include

전처리기 / 헤더 파일

```
#include "func.h"

int add(int a, int b){
    func();
    return a+b;
}

void func(){
}
```

- 다른 함수가 필요하면 원형 선언 필요
-> 헤더파일 사용하기
- 보통은 #include <stdio.h> 필요

main 함수 인자

main 함수 인자

```
#include <stdio.h>

int main(int argc, char** argv) {
    printf("인자 개수 argc: %d\n", argc);
    for (int i = 0; i < argc; ++i)
        printf("*argv[%d]: %s\n", i, argv[i]);
    return 0;
}
```

- argc: 인자 개수
디폴트 1개(프로그램 실행 경로)
- argv: char* 배열을 가리키는 char**
- argv[i]: 인자(문자열)을 가리키는 char*

```
C:\Users\khyoo\KU\HandS\2020\Seminar2020\C_Intermediate\src\cmake-build-debug>src.exe
```

```
인자 개수 argc: 1
```

```
*argv[0]: src.exe
```

```
C:\Users\khyoo\KU\HandS\2020\Seminar2020\C_Intermediate\src\cmake-build-debug>src.exe 인자1 인자2
```

```
인자 개수 argc: 3
```

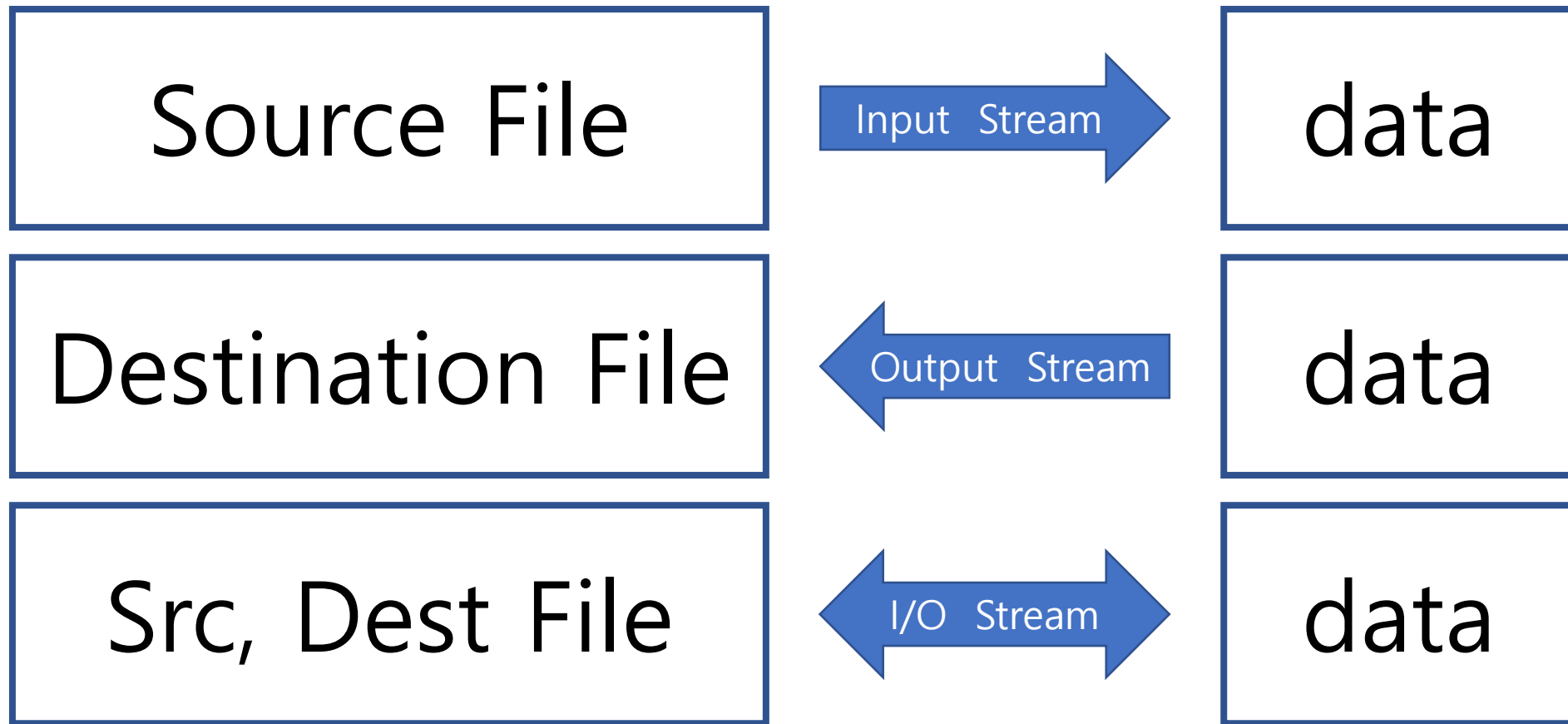
```
*argv[0]: src.exe
```

```
*argv[1]: 인자1
```

```
*argv[2]: 인자2
```

파일 입출력

스트림 데이터를 주고받는 통로



파일 출력

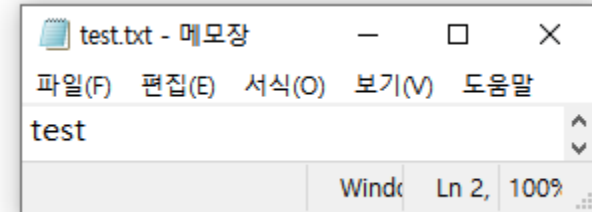
```
#include <stdio.h>

int main() {
    FILE *fp;
    fp = fopen("test.txt", "w");

    if (fp == NULL) {
        printf("Error\n");
        return 0;
    }

    fputs("test\n", fp);

    fclose(fp);
    return 0;
}
```



- FILE *fp: 파일 구조체
- fopen("file_name", "w"): 출력만 가능한 스트림을 가리키는 포인터 리턴

파일 입력

```
#include <stdio.h>
int main() {
    FILE *fp = fopen("test.txt", "r");
    char buf[20];
    if (fp == NULL) {
        printf("ERROR\n");
        return 0;
    }
    fgets(buf, 20, fp);
    printf("test.txt : %s \n", buf);
    fclose(fp);
    return 0;
}
```

test.txt : test

- fopen("file_name", "r"): 입력만 가능한 스트림을 가리키는 포인터 리턴
- fgets(buf, 20, fp): buf[20]에 \n나올 때까지 20바이트 읽어서 저장

파일 위치 지시자

```
#include <stdio.h>

int main() {
    FILE *fp = fopen("test.txt", "r");
    char a, buf[20];

    while ((a = fgetc(fp)) != EOF)
        printf("%c", a);

    fseek(fp, 0, SEEK_SET);
    fgets(buf, 20, fp);
    printf("\nfgests: %s", buf);
    fclose(fp);
    return 0;
}
```

test

fgests: test

- fseek(fp, offset, origin)
파일 위치 지시자를 origin에서
offset만큼 이동시킴
- SEEK_SET, SEEK_CUR, SEEK_END
파일 시작, 현재 위치, 파일 끝

↓
위치 지시자를 파일 시작위치로

t

e

s

t

파일 입출력

```
#include <stdio.h>

int main() {
    FILE *fp = fopen("test.txt", "r+");
    char a, buf[20];

    while ((a = fgetc(fp)) != 'e')
        printf("%c", a);
    fseek(fp, 0, SEEK_CUR);
    fputs("EEEE", fp);
    fseek(fp, 0, SEEK_SET);
    fgets(buf, 20, fp);
    printf("\nfgests: %s", buf);
    fclose(fp);
    return 0;
}
```

```
t
fgests: teEEEE
```

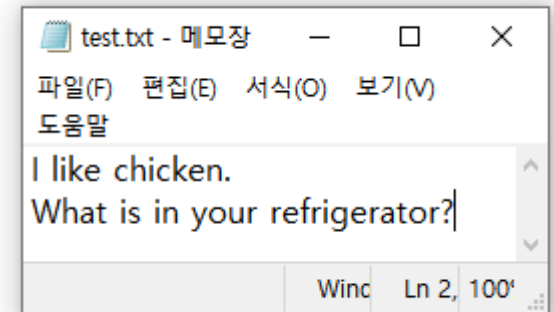
- 입출력 전환할 때 fseek이나 fflush 필수
- r+: 읽기+쓰기
- w+: 쓰기+읽기
파일 존재 -> 내용 지움
파일 존재 X -> 파일 생성
- a, a+, t, b 등

fscanf 함수

```
#include <stdio.h>

int main() {
    FILE *fp = fopen("test.txt", "r");
    char buf[20];
    if (fp == NULL) {
        printf("ERROR\n");
        return 0;
    }
    while (fscanf(fp, "%s", buf) != EOF)
        fprintf(stdout, "%s\n", buf);
    fclose(fp);
    return 0;
}
```

```
I
like
chicken.
What
is
in
your
refrigerator?
```



- 파일 끝 -> EOF 리턴
- fscanf(fp, "%s", buf):
띄어쓰기, \t, \n이 나올 때까지 읽음

실습

실습 코드 설명 기본 변수 및 구조체

```
void main(int argc, char** argv) {  
    int choice; // 사용자가 선택한 메뉴  
    int food_num = 0; // 현재 식재료 개수  
    int date[3] = { 2020, 3, 15 }; // 오늘 날짜  
    Foods foods[5];  
    for (int i = 0; i < 5; ++i) {  
        foods[i].pfood = (Foods*) malloc(sizeof(Food) * 2);  
        foods[i].capacity = 2;  
        foods[i].Foodnum = 0;  
    }  
    if (argc > 1) // 이전 data 불러오기  
        load_food_data(foods, &food_num, argv[1]);  
}
```

실습 코드 설명 main 함수

```
while (1) {
    printf("행동을 선택하세요 \n");
    printf("1. 식재료 추가하기 \n");
    printf("2. 현재 있는 식재료 보여주기 \n");
    printf("3. 유통기한 지난 식재료 보여주기 \n");
    printf("4. 식재료 개수 증감 \n");
    printf("5. 데이터 저장하고 프로그램 종료 \n");

    printf("번호 입력 : ");
    scanf("%d", &choice);
    printf("\n");
    switch (choice) {
        case 1: add_food(foods, &food_num); break; /* 식재료 추가 */
        case 2: show_food(foods, food_num); break; /* 현재 식재료 보여주기 */
        case 3: show_expired_food(foods, food_num, date); break; /* 유통기한 지난 식재료 보여주기 */
        case 4: change_food_num(foods, &food_num); break; /* 식재료 개수 증감 */
    }
    printf("\n");
    if (choice == 5) {
        save_food_data(foods, food_num); /* 데이터 저장하고 프로그램 종료 */
        break;
    }
}
```

실습 구현 내용 #1, 2, 3, 4

```
void add_food(Foods* pfoods, int* food_num) {  
    /** Your Code Here **/  
    printf("add food\n"); // delete  
}  
  
void show_food(Foods* pfoods, int food_number) {  
    /** Your Code Here **/  
    printf("show food\n"); // delete  
}  
  
void show_expired_food(Foods* pfoods, int food_number, int date[]) {  
    /** Your Code Here **/  
    printf("show expired food\n"); // delete  
}  
  
void change_food_num(Foods* pfoods, int* food_number) {  
    /** Your Code Here **/  
    printf("change_food_num\n"); // delete  
}
```

이전과 같습니다
수정 할 필요 X

실습 구현 내용 #5

번호 입력 :1

식재료 종류 (0: Meat, 1: Dairy, 2: Fruit, 3: Bread, 4: Sweets):0

식재료 이름:Pork

Pork를 냉장고에 넣었습니다.

식재료 유통기한(ex. 2020 1 23):2020 7 10

Pork의 유통기한은 2020년 7월 10일입니다. 상하기 전에 빨리 먹으세요.^^

식재료 개수:500

번호 입력 :1

식재료 종류 (0: Meat, 1: Dairy, 2: Fruit, 3: Bread, 4: Sweets):2

식재료 이름:Salad

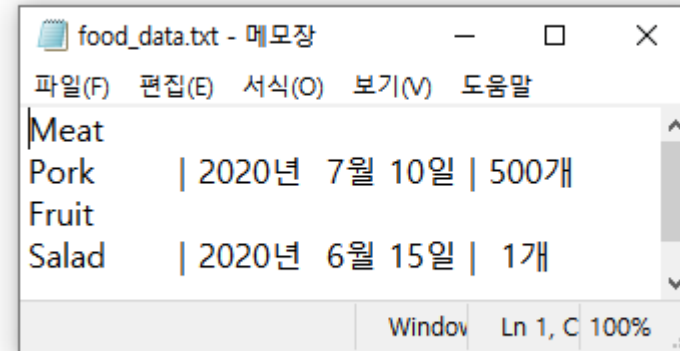
Salad를 냉장고에 넣었습니다.

식재료 유통기한(ex. 2020 1 23):2020 6 15

Salad의 유통기한은 2020년 6월 15일입니다. 상하기 전에 빨리 먹으세요.^^

식재료 개수:1

```
void save_food_data (Foods* pfoods, int* food_number) {  
    /** Your Code Here **/  
    printf("save_food_data\n"); // delete  
}
```



행동을 선택하세요

1. 식재료 추가하기
2. 현재 있는 식재료 보여주기
3. 유통기한 지난 식재료 보여주기
4. 식재료 개수 증감
5. 데이터 저장하고 프로그램 종료

번호 입력 :5

식재료가 없어서 데이터를 저장하지 않습니다.

행동을 선택하세요

1. 식재료 추가하기
2. 현재 있는 식재료 보여주기
3. 유통기한 지난 식재료 보여주기
4. 식재료 개수 증감
5. 데이터 저장하고 프로그램 종료

번호 입력 :5

데이터가 저장되었습니다. 프로그램을 종료합니다.

- 현재 식재료 정보를 저장하는 함수
- 저장 파일 이름, 저장 형식 자유

실습 구현 내용 #6

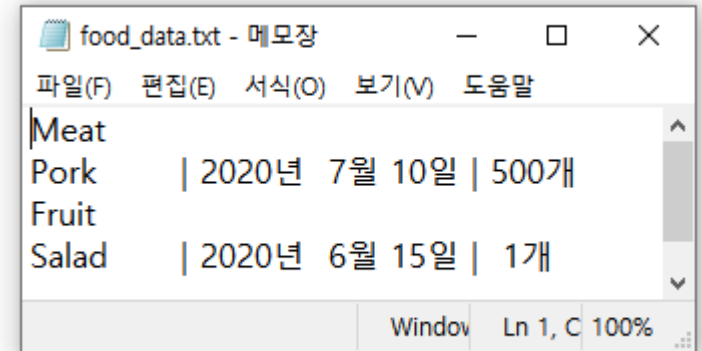
```
if (argc > 1) // 이전 data 불러오기
    load_food_data(foods, &food_num, argv[1]);
```

```
C:\Users\khyoo\KU\HandS\2020\Seminar2020\C_Intermediate\src\cmake-build-debug>src.exe food_data.txt
loading food_data.txt...

<< 자취생 냉장고 관리 프로그램 >>

행동을 선택하세요
1. 식재료 추가하기
2. 현재 있는 식재료 보여주기
3. 유통기한 지난 식재료 보여주기
4. 식재료 개수 증감
5. 데이터 저장하고 프로그램 종료
번호 입력 : 2

Meat
1: Pork      | 2020년 7월 10일 | 500개
Fruit
2: Salad     | 2020년 6월 15일 | 1개
```



- 이전 데이터를 불러와서 구조체에 저장함
- 2번째 인자를 인수로 넣어줌

```
void load_food_data(Foods* pfoods, int* food_number, char* filename){
    /** Your Code Here ***/
    printf("load_food_data\n"); // delete
}
```