2017년 2학기

객체지향프로그래밍

CSED232

Assignment #1

담당교수 : 윤은영

학번 : 20160074

학과 : 화학공학과

이름 : 고진민

POVIS ID : eric9709

--------------------------------------------------------------------------------------------------------------

<명예서약>

나는 이 프로그래밍 과제를 다른 사람의 부적절한 도움 없이 완수하였습니다.

--------------------------------------------------------------------------------------------------------------

1. **프로그램 개요**

이 프로그램은 “피자가게 메뉴 관리 및 주문 시스템”이다. 프로그램은 구체적으로 전체 메뉴의 조회 / 주문 / 메뉴 추가 / 메뉴 삭제 / 메뉴 검색 / 인기메뉴 추천 / 종료 기능을 포함한다.

피자의 구성 요소를 토핑, 도우, 에지의 세 종류로 나누어 각각을 class로 만들고, 각 객체로부터 생성된 인스턴스를 같은 class에 속하는 것들끼리 linked list로 구현한다. 토핑, 도우, 에지의 linked list 구현을 위한 list class도 추가로 정의된다. 각 멤버 변수는 모두 private이고 접근은 public 메소드로 접근한다.

메뉴별 간단한 기능은 다음과 같다. 조회는 전체 메뉴를 출력한다. 메뉴 추가는 토핑 / 도우 / 에지 중 하나를 선택해 해당 클래스의 정보를 입력해 메뉴를 추가한다. 메뉴 삭제는 이미 있는 메뉴의 이름을 입력받아 일치하면 삭제를, 아니라면 에러 메시지를 띄운다. 메뉴 검색은 세 종류의 클래스 중 하나를 선택해 세부 조건에 맞는 정보를 입력하고 해당 정보에 부합하는 메뉴를 찾아 출력한다. 인기메뉴 추천은 팔린 횟수가 가장 많은 메뉴를 클래스별로 하나씩 출력하되, 횟수가 같으면 먼저 추가된 메뉴가 출력된다. 종료는 프로그램을 종료한다.

1. **Class 구성**

|  |  |
| --- | --- |
| **Dough** | **DoughList** |
| - string name  - string comment  - int sell\_count,  - int extra\_cost  - float thickness  - Dough\* next | - Dough\* head  - int size |
| +Dough()  +void set\_name(string newname)  +void set\_thickness(float newthickness)  +void set\_extra\_cost(int newextra\_cost)  +void inc\_sellcount()  +void set\_comment(string newcomment)  +void set\_pointer(Dough \*newnext)  +string get\_name()  +float get\_thickness()  +int get\_extra\_cost()  +int get\_sell\_count()  +string get\_comment()  +Dough\* get\_pointer() | +DoughList()  +void add(Dough\* dough)  +int del(string menuname)  +Dough\* pick(string menuname)  +void show\_all\_menu()  +void show\_most\_popular()  +void search\_by\_thick(float hvalue, float lvalue) |
| **Edge** | **EdgeList** |
| - string name  - string mousse[3]  - string comment  - int sell\_count  - int extra\_cost  - Edge\* next | - Edge\* head  - int size |
| +Edge();  +void set\_name(string newname)  +void set\_extra\_cost(int newextra\_cost)  +void inc\_sellcount()  +void set\_comment(string newcomment)  +void set\_pointer(Edge \*newnext)  +string get\_name()  +string\* get\_mousse()  +int get\_extra\_cost()  +int get\_sell\_count()  +string get\_comment()  +Edge\* get\_pointer() | +EdgeList()  +void add(Edge\* edge)  +int del(string menuname)  +Edge\* pick(string menuname)  +void show\_all\_menu()  +void show\_most\_popular()  +void search\_by\_mousse(string moussename) |

|  |  |
| --- | --- |
| **Topping** | **ToppingList** |
| - string name  - string ingredient[10]  - string comment  - int sell\_count,  - int extra\_cost  - Topping\* next | - Topping\* head  - int size |
| +Topping()  +void set\_name(string newname)  +void add\_ingredient(string\* newingredient)  +void set\_price(int newprice)  +void inc\_sellcount()  +void set\_comment(string newcomment)  +void set\_pointer(Topping \*newnext)  +string get\_name()  +int get\_price()  +int get\_sell\_count()  +string get\_comment()  +Topping\* get\_pointer() | +ToppingList();  +void add(Topping\* topping);  +int del(string menuname);  +Topping\* pick(string menuname);  +void show\_all\_menu();  +void show\_most\_popular();  +void search\_by\_ingred(string ingname);  +void search\_by\_price(float hprice, float lprice); |

Dough, Edge, Topping 클래스의 경우 기본적으로 매뉴얼에 주어진 멤버 변수들을 사용했고, 여기에 linked list 구조를 구현하기 위한 node 형태를 만들기 위해 각 클래스의 주소값을 저장할 수 있는 private 포인터 변수인 next 변수를 정의했다. 또한 이 포인터에 접근하기 위해 주소값을 입력받아 next에 입력받는 public인 set\_pointer 함수와 next에 저장된 주소값을 반환하는 함수인 get\_pointer 함수를 추가로 사용했다.

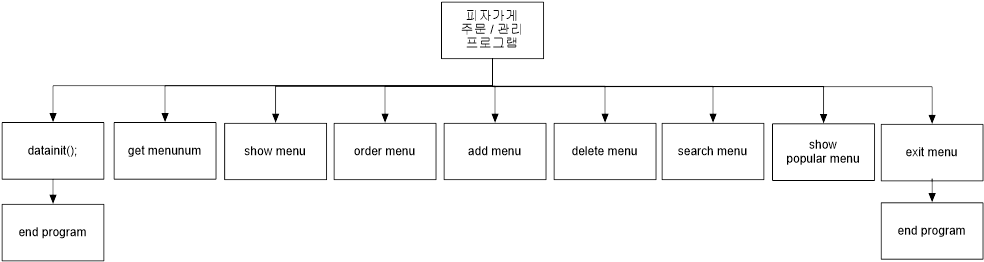
DoughList, EdgeList, ToppingList 클래스의 경우 매뉴얼에 나와있는 멤버 변수와 메소드 외에는 구현한 내용이 없다.

여섯 종류의 클래스 모두에 대해 생성자(constructor)를 이용해 멤버 변수 중 초기화가 필요한 변수들을 초기화했다. Linked list의 구현에 필요한 포인터 변수는 공통적으로 NULL로, list 클래스들의 경우에는 size를 0으로, 그 외의 세 클래스들은 sell\_count 변수와 ingredient, mousse 배열을 0으로 초기화했다.

각 함수의 구체적인 기능과 구현은 소스 코드 및 프로그램 설명 부분에서 진행할 것이다.

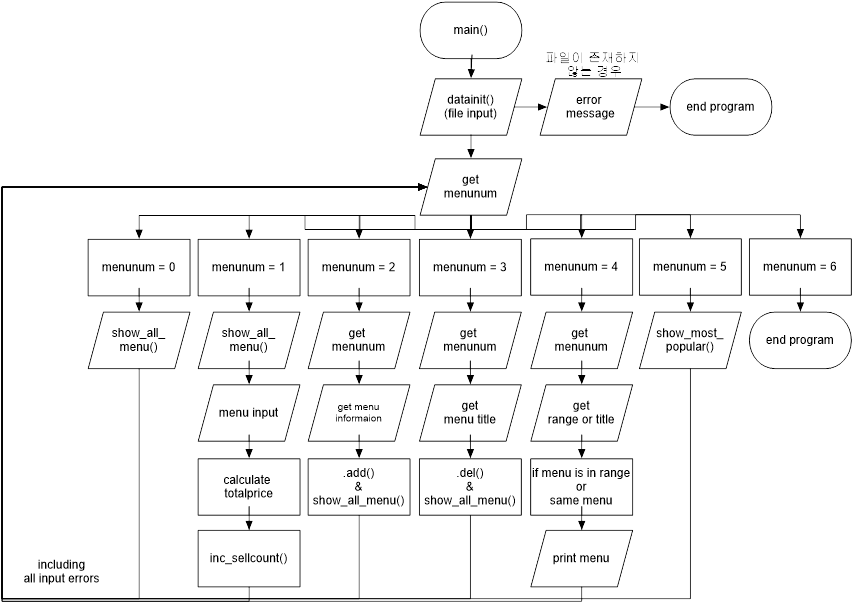
1. **구조도 및 알고리즘**

이 프로그램의 전체적인 structure chart는 다음과 같다.



<전체 프로그램의 structure chart>

main 함수의 대략적인 알고리즘은 아래 그림과 같다.



<main 함수의 flow chart>

1. **소스 코드 및 프로그램 설명**

main 함수를 제외한 6개의 .cpp 파일에 대한 헤더 파일은 class의 정의 뿐이고 기타 메소드의 구현은 .cpp 파일에 진행되어 있으므로 헤더 파일에 대한 설명은 생략하도록 한다.

**<main.cpp>**

핵심이 되는 코드라고 할 수 있는 main.cpp 코드이다. 이 파일에서는 크게 3개의 함수를 사용하는데, main 함수와 printMenu, datainit 함수가 그것이다.

**printMenu 함수**는 입력값과 반환값이 없고, 초기 메뉴 화면을 출력하는 기능을 가지는 함수이다.

**datainit 함수**는 main 함수에서 선언한 세 doughList, edgeList, toppingList 클래스의 인스턴스의 주소값을 입력받고, 반환값은 정수형이다. 이 함수는 입력받은 인스턴스의 주소값을 토대로 linked list 형태로 init\_menu.txt 파일로부터 파싱한 데이터를 연결하는 초기화 함수이다. 먼저 변수와 파일을 읽을 때 사용할 스트림을 선언한다. 이후 init\_menu.txt 파일을 읽어온다. 이 때 해당 파일이 없는 경우 에러 메시지를 출력한 뒤 0을 반환한다. 파일이 있는 경우 파싱 작업을 시작한다. 파싱 작업은 getline 함수를 이용해 .txt 파일의 내용을 line by line으로 읽어들인 뒤, split 하는 과정을 거쳐서 원하는 데이터를 인스턴스에 입력 후 list에 링크하는 과정을 반복한다. 이 과정은 while 루프를 통해서 이루어진다. 먼저 입력받은 내용이 “-토핑” 과 같은 클래스를 구분하는 문자열인 경우인지를 판단한다. 클래스 구분 문자열이라면 해당 클래스의 인스턴스에 데이터를 입력할 수 있도록 조건문을 찾아 이동한다. 만약 그렇지 않은 인스턴스에 들어갈 내용이라면 파싱한 데이터를 입력받을 인스턴스 하나를 동적할당 받은 뒤, split 과정을 수행한다. Split 과정은 strtok 함수를 이용하게 되는데, 우리가 처음 defalut로 사용할 txt 파일에서는 탭(\t)과 콤마(,)로 데이터 구분이 이루어지는 점을 이용해 두 가지의 토큰으로 splitting 과정을 수행해야 한다. Topping 부분의 데이터를 예로 들어보자면, 처음에 탭으로 split한 문자열은 이름으로, 그 다음에 split되는 문자열은 아직 쪼개지지 않은 ingredient의 전체 문자열로, 그 다음에는 price로, 마지막에는 comment로 문자열을 쪼갰다. 이후 **ingredient의 경우 콤마를 token으로 생각해 splitting 과정을 다시 한 번 수행**해 이 때 쪼개져 나오는 문자열들은 NULL로 초기화된 ingredient[10] 배열에 순서대로 하나씩 대입한다. 이는 **Edge 인스턴스의 mousse**도 비슷한 과정을 거친다. 이외에도 유의할 점은, price와 extra\_cost, thickness의 경우 string으로 처음 데이터가 받아지지만 이는 **atoi 혹은 atof 함수를 이용하여 int 혹은 float와 같은 적절한 변수형으로 변환** 후 대입되어야 한다. 이러한 과정을 거쳐 전체 파일의 모든 클래스에 대해 파싱 작업이 끝나면 열었던 파일을 close하고, 0을 return한다.

한 가지 구현상에서 참고할 점은 **Edge 인스턴스의 경우 mousse에 입력되는 값이 아예 없는 경우가 존재해서 예외처리를 해 주었지만 Topping 인스턴스의 ingredient의 경우 이러한 상황을 배제**했다. 기본적으로 피자에 토핑이 없는 경우는 생각하지 않기로 했다. 비슷한 이유로 **이름, 추가비용, 코멘트 등에서 입력값이 존재하지 않는 경우는 존재하지 않는다고 판단**, 구현을 진행했다. Mousse에 대한 예외처리는 find 함수를 통해서 진행했는데, find 함수를 통해서 처음 탭이 나오는 index와 두 번째로 탭이 나오는 index를 비교해서 둘의 차이가 1이라면, 즉 탭 다음 바로 탭이 나온다면 탭을 이용한 split 과정을 한 번 생략하고 파싱을 진행한다. 그렇지 않은 경우라면 mousse 부분에 데이터가 존재한다는 뜻이므로 그대로 splitting을 진행한다.

**main 함수**는 골격이 되는 메뉴를 구현한 함수이다. 먼저 변수를 선언하고, datainit 함수를 이용해 데이터를 파싱한다. 만일 파일이 존재하지 않으면(반환값이 0이면) 즉시 프로그램을 종료한다. 그렇지 않다면(반환값이 1이면) 메뉴 루프로 진입하게 된다. 메뉴 루프로 진입한 후에는 printMenu 함수를 이용해 메뉴를 출력하고, getline 함수를 이용해서 메뉴 번호를 입력받는다. 메뉴 번호를 입력받는 과정에서 getline 함수를 사용한 이유는 ‘잘못된 입력’의 범위를 ‘띄어쓰기가 포함된 문자열 혹은 문자 입력이 없는 상태에서의 엔터 입력’까지로 생각했기 때문이다. 이 때문에 getline 함수를 이용해 0~6을 제외한 문자열이 입력되면 모두 잘못된 입력으로 간주하도록 구현했다.

0을 입력받으면 각 list 클래스에 정의된 show\_all\_menu 함수를 호출해 모든 메뉴를 토핑, 도우, 에지 순으로 한 번씩 출력하게 된다. 이후 메뉴 루프 처음으로 이동.

1을 입력받으면 show\_all\_menu 함수를 호출해 모든 메뉴를 한 번씩 출력한 뒤 토핑, 도우 에지 순으로 메뉴 이름을 하나씩 입력받는다. 이 때, 매 입력 때마다 메뉴가 존재하는지 확인해 그렇지 않으면 에러 메시지를 출력하고 메뉴 루프의 처음으로 돌아간다. 이는 list 클래스의 pick 함수를 이용해서 구현할 수 있는데, 반환값이 NULL이라면 메뉴가 존재하지 않는다고 판단할 수 있다. 올바른 입력이 이루어졌다면 pick 함수에서 반환된 주소를 이용해 price와 extra\_cost를 get 함수를 이용해 받아올 수 있고, 이들을 합해 총액을 출력한다. 마지막으로 inc\_sellcount 함수를 이용해서 각 list에서 팔린 메뉴의 sell\_count를 1씩 올려준다. 이후 메뉴 루프 처음으로 이동.

2를 입력받으면 메뉴를 추가하는 작업을 거치게 된다. 먼저 어떤 클래스의 메뉴를 추가할 것인지 입력받는다. 예외 처리는 메인 메뉴와 같은 방법으로 이루어진다. 클래스의 종류가 결정되면 입력되는 해당 클래스의 인스턴스 하나를 동적할당 받는다. 이후 각 클래스의 멤버 변수에 해당하는 내용을 순서대로, 그리고 init\_menu.txt와 같은 형태로 입력한다(ingredient와 mousse는 콤마로 구분되어 입력). Topping의 ingredient와 Edge의 mousse는 입력값이 없어도 에러 메시지를 출력하지 않고 비어있는 상태로 진행할 수 있게 구현했다. 또한 이 둘은 문자열을 입력받은 뒤 strtok를 이용해 split 과정을 거친 뒤 배열에 차례대로 입력하게 된다. 이외의 멤버 변수에 해당하는 경우는 입력값이 없으면 잘못된 입력으로 간주하고 에러 메시지 출력 후 루프의 처음으로 돌아간다. 입력받은 메뉴가 이미 존재하는 경우 메뉴가 이미 존재한다는 메시지를 출력하고 루프의 처음으로 돌아간다. 올바른 입력값이 주어졌다면 동적할당된 인스턴스에 각 데이터를 입력하고(필요하다면 atoi 혹은 atof와 같은 변환 과정을 거친 후) 그 인스턴스를 list 클래스의 add 함수를 이용해 list의 마지막에 링크한다. 이후 메뉴 루프 처음으로 이동.

3을 입력받으면 기존에 있던 메뉴를 삭제하게 된다. 먼저 어떤 클래스의 메뉴를 삭제할 것인지 입력받는다. 예외 처리는 메인 메뉴와 같은 방법으로 이루어진다. 클래스의 종류가 결정되면 해당 클래스의 메뉴 이름을 입력받는다. 이 때 list 클래스의 del 함수를 이용한다. 만약 메뉴가 존재하면 삭제 성공 메시지 출력 후 해당 클래스의 전체 메뉴를 한 번 출력하고 루프의 처음으로 돌아간다. 입력받은 이름의 메뉴가 없다면 에러메시지 출력 후 루프의 처음으로 돌아간다. del 함수를 이용하면 함수 내부에서 알아서 메뉴의 존재 여부를 판단하고 삭제 과정을 거치게 된다.

4를 입력받으면 메뉴를 검색하게 된다. 먼저 어떤 클래스의 메뉴를 검색할 것인지 입력받는다. 예외 처리는 메인 메뉴와 같은 방법으로 이루어진다. 클래스가 정해지면 검색 조건에 맞게 데이터를 입력받는다. 이름이면 하나를, 범위라면 하한과 상한을 순서대로 입력한다. 입력이 업는 경우는 오류 메시지 출력 후 루프의 처음으로 이동한다. 입력이 올바른 경우 list 클래스의 search 계열 함수를 이용해서 해당 입력값을 만족하는 메뉴들을 출력하도록 한다. 이후 루프의 처음으로 돌아간다.

5를 입력받으면 토핑, 도우, 에지 순으로 sell\_count가 가장 높은 메뉴를 출력한다. List 클래스의 함수 show\_most\_popular 함수를 이용하고, 만약 sell\_count가 같다면 먼저 등록된 메뉴를 먼저 출력한다. 출력 후에는 루프의 처음으로 돌아간다.

6을 입력받으면 메뉴 루프를 이탈하게 된다. 루프를 이탈하면 바로 프로그램이 종료된다.

**<Dough.cpp, Edge.cpp, Topping.cpp>**

이 세 .cpp 파일은 크게 차이가 없고 메소드나 멤버 변수 한 두개 정도만 차이가 있으므로 한 번에 설명하도록 하겠다. 공통으로 존재하는 메소드 함수를 먼저 설명하겠다.

set\_name 함수는 string형 값을 입력받아 멤버 변수인 name에 값을 입력한다.

set\_price 혹은 set\_extra\_cost 함수는 정수형 값을 입력받아 멤버 변수 price 혹은 extra\_cost 변수에 값을 입력한다.

inc\_sellcount는 입력/반환값 없이 멤버 변수 sell\_count의 값을 1 증가시킨다.

set\_comment는 string형 값을 입력받아 멤버 변수인 comment에 값을 입력한다.

set\_pointer는 각 클래스의 주소값에 해당하는 값을 입력받아 그 값을 멤버 변수이자 포인터 변수인 next에 입력한다.

get\_name은 각 클래스의 string형 name 멤버 변수를 반환한다.

get\_price 혹은 get\_extra\_cost는 각 클래스의 정수형 멤버 변수인 price나 extra\_cost의 값을 반환한다.

get\_sell\_count는 각 클래스의 정수형 멤버 변수인 sell\_count의 값을 반환한다.

get\_pointer는 각 클래스의 포인터형 멤버 변수인 next의 값을 반환한다.

Edge 클래스의 add\_mousse는 배열의 주소를 입력받아 해당 배열을 mousse 멤버 배열 변수로 복사해 저장한다.

Edge 클래스의 get\_mousse는 mousse의 주소값을 반환한다.

Dough 클래스의 set\_thickness는 float형 값을 입력받아 멤버 변수 thickness에 대입한다.

Dough 클래스의 get\_thickness는 멤버 변수 thickness의 값을 반환한다.

Topping 클래스의 add\_ingredient는 배열의 주소를 입력받아 해당 배열을 ingredient 멤버 배열 변수로 복사해 저장한다.

Topping 클래스의 get\_ingredient는 ingredient의 주소값을 반환한다.

**<DoughList.cpp, EdgeList.cpp, ToppingList.cpp>**

이 세 파일 역시 search 계열 함수를 제외하면 모두 같은 기능을 하는 함수들이 공통적으로 포함되어 있으므로 한 번에 같이 설명하겠다.

add 함수는 list에 추가할 인스턴스의 주소값을 입력받아 linked list 형태로 구현된 list의 마지막에 해당 인스턴스(노드)를 추가하는 함수이다. List가 비어있는 경우는 list의 head가 바로 인스턴스를 가리키며, 그렇지 않은 경우는 list의 마지막까지 포인터를 이동시켜서 인스턴스를 링크한다.

del 함수는 string형의 메뉴 이름을 입력받아 해당 이름을 가지는 인스턴스를 linked list에서 제외하고 동적할당된 메모리를 해제한다. 메뉴 삭제에 성공한 경우는 1을, 그렇지 못한 경우는 0을 반환한다. 먼저 head가 NULL인 list가 비어있는 상황은 0을 반환하고 끝. 메뉴가 처음과 일치하는 경우와 그 외의 경우를 나누어 list의 끝까지 이동하면서 이름을 확인하고 이름이 일치하는 메뉴를 찾으면 해당 메뉴를 list에서 제외하고 할당된 메모리를 해제한다.

pick 함수는 string형의 메뉴 이름을 입력받아 존재하면 인스턴스의 주소를 반환하고 그렇지 않은 경우는 NULL을 반환한다. 기작 자체는 del 함수와 같으며 이름이 일치하는 부분을 찾았을 때 해당 인스턴스의 주소값을 반환한다는 점만 다르다.

show\_all\_menu 함수는 해당 클래스의 메뉴 전체를 출력하는 함수이다. pWalk 포인터를 list의 처음부터 끝까지 이동시키며 하나씩 전부 출력한다.

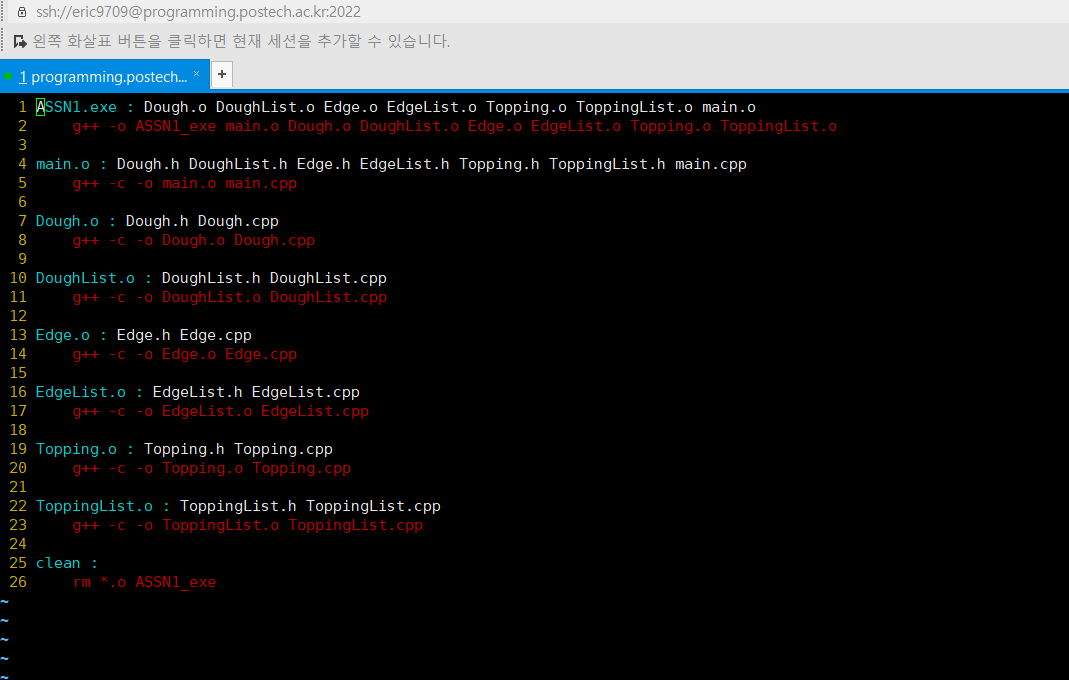
show\_most\_popular 함수는 해당 클래스에서 sell\_count가 가장 큰 인스턴스를 출력한다. 만약 sell\_count가 같다면 먼저 등록된, 즉 list의 머리에 가까운 인스턴스가 출력된다. 이는 역시 pWalk 포인터가 처음부터 끝까지 이동하지만 get\_sell\_count 함수를 호출해 계속해서 최대값을 비교하고 갱신한다. 갱신될 때마다 갱신되는 인스턴스의 주소를 저장했다가 pWalk가 끝에 도달하면 해당 주소의 인스턴스에 해당하는 메뉴를 출력한다.

search 계열 함수는 string형 이름 혹은 두 정수 혹은 float형 값을 입력받아 범위 내에 존재하거나 이름이 일치하는 메뉴를 모두 출력한다. 역시 pWalk 포인터가 link의 처음부터 끝까지 이동하지만 이번에는 조건에 맞는 인스턴스들만 출력하게 된다.

1. **프로그램 실행**

리눅스 환경은 학교 실습 서버를 이용했다. 이번 과제에 사용한 소스 파일들은 모두 /home/std/eric9709/CSED232/ASSN1/test 위치에 저장되어 있다.

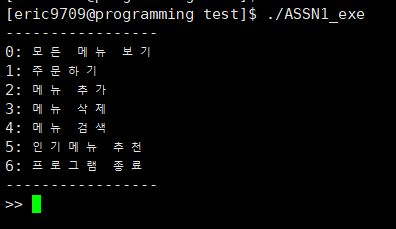
아래 그림처럼 Makefile을 만들고, make 명령어를 통해서 컴파일을 진행하면 컴파일이 완료된다.

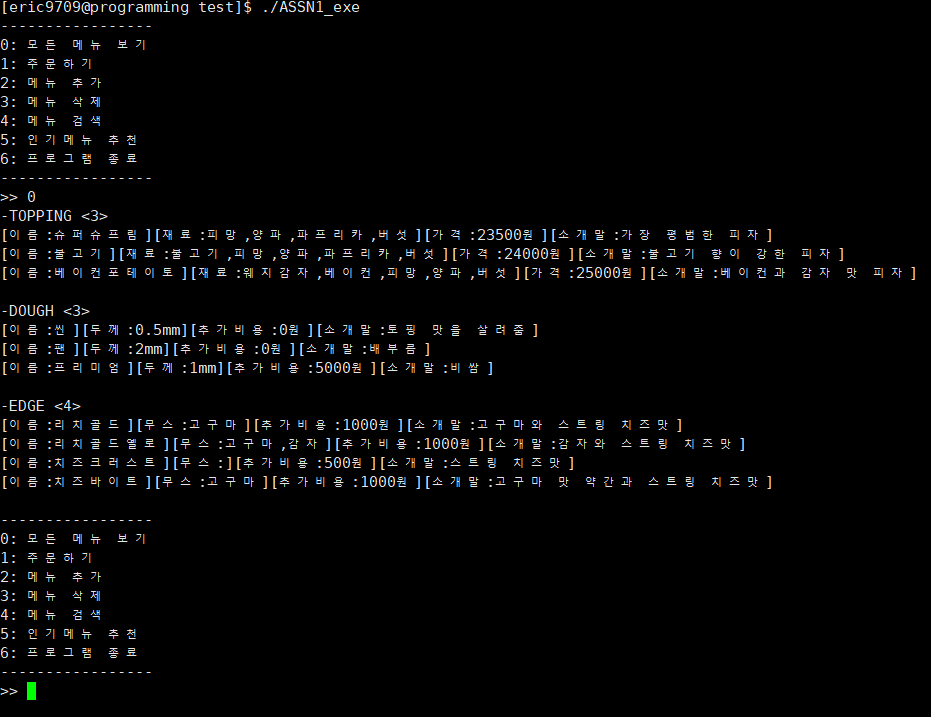




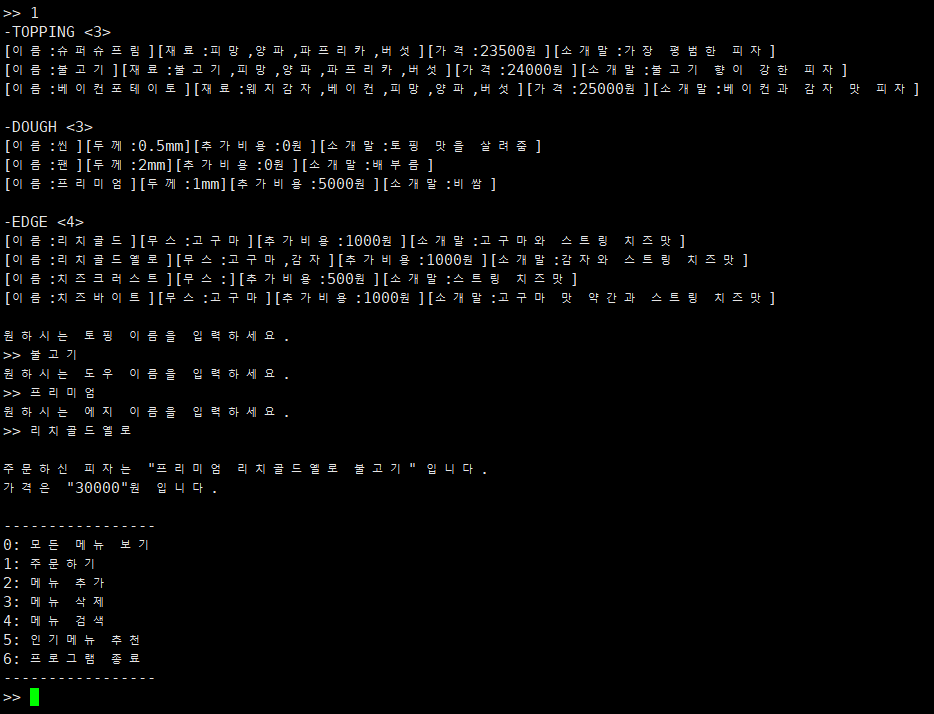
<Makefile과 이를 이용해 make 명령어로 컴파일을 완료한 모습>

컴파일을 진행하고, ./ASSN1\_exe 명령어로 실행 파일을 실행하면 아래와 같이 초기 메뉴 화면이 나오게 된다.

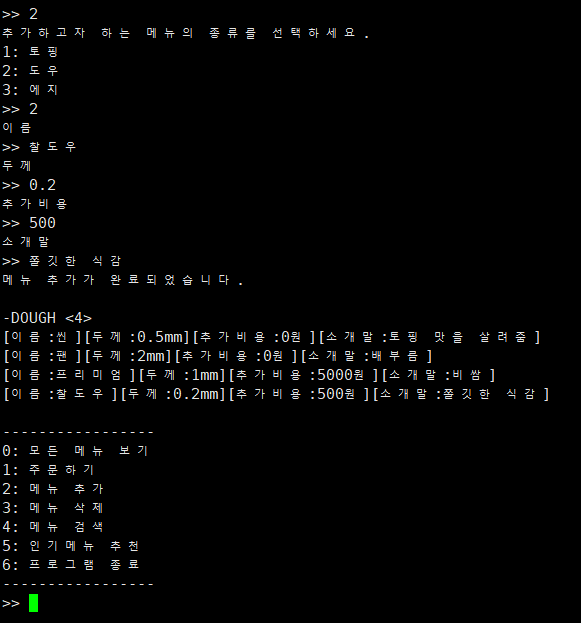




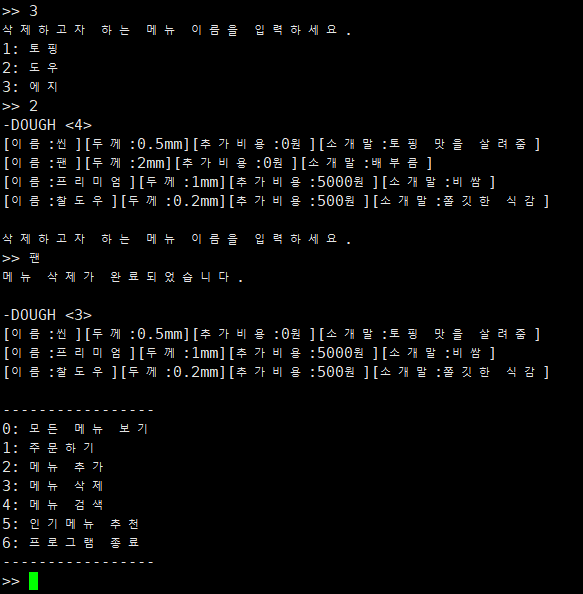
<0을 입력한 모습>



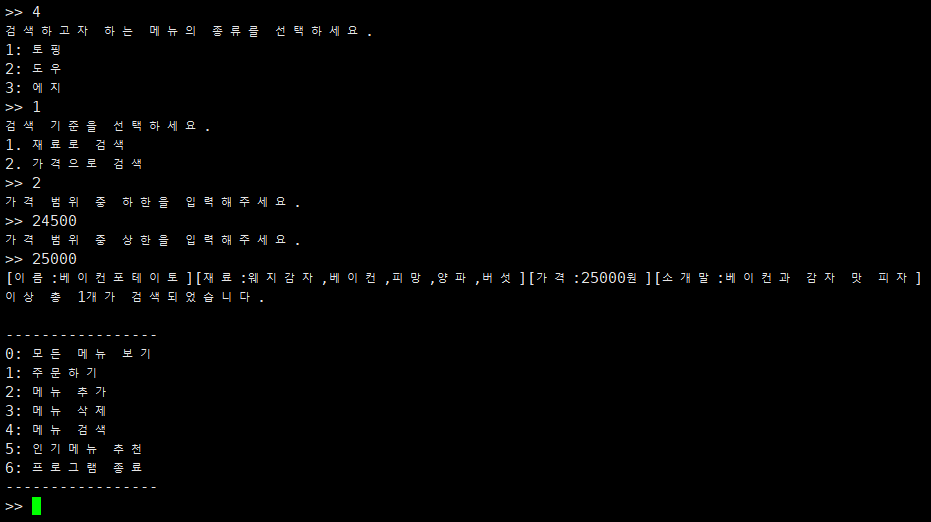
<1을 입력한 후 불고기, 프리미엄, 리치골드옐로를 입력한 모습>



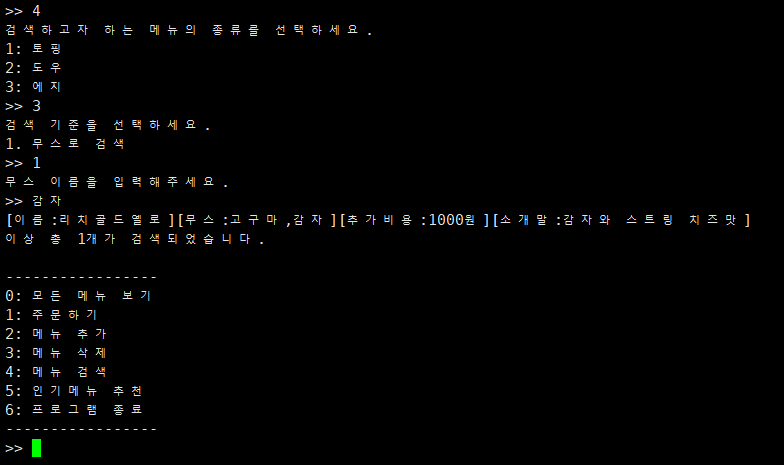
<2를 입력한 이후 새로운 메뉴를 추가한 모습>



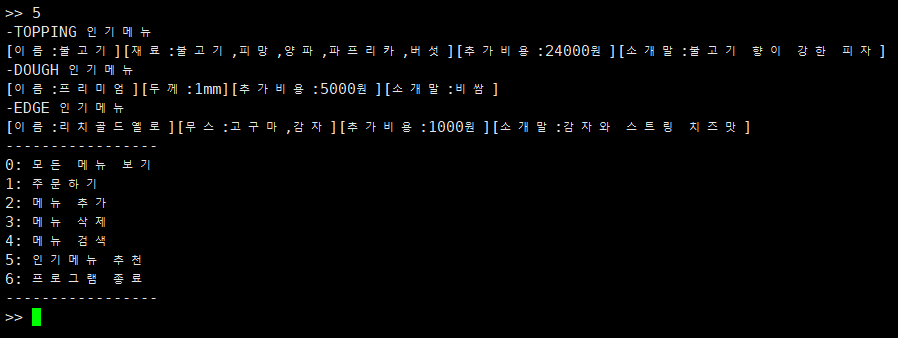
<3을 입력한 이후 도우에서 팬을 삭제한 모습>



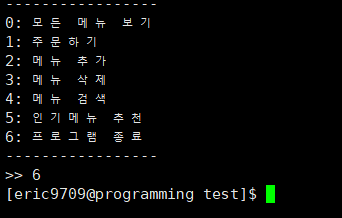
<4번 메뉴에서 토핑을 가격 범위로 검색한 모습>



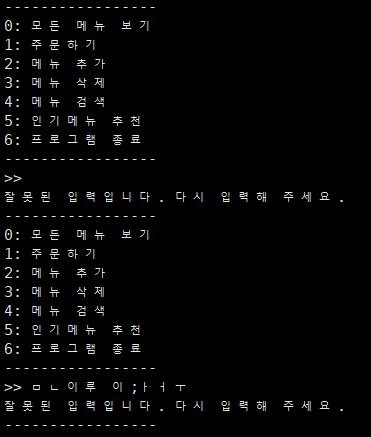
<4번 메뉴에서 에지를 무스로 검색한 모습>



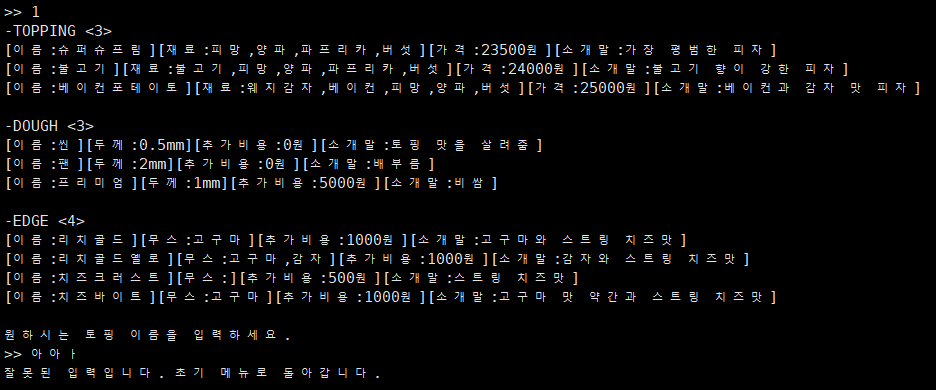
<위에서 주문한 값을 토대로 5번 메뉴를 실행한 모습 – 주문한 메뉴가 가장 인기있다>



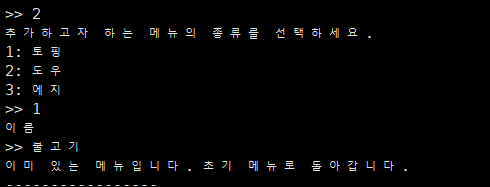
<6번 메뉴를 실행하여 프로그램이 종료된 모습>



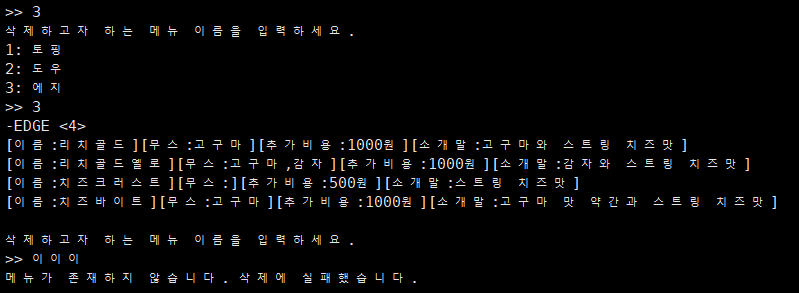
<입력 예외처리 – 입력 없는 엔터와 숫자가 아닌 띄어쓰기가 포함된 문자열의 입력>



<입력 예외처리 – 주문 시 없는 메뉴를 입력한 경우>



<입력 예외처리 – 2번 메뉴에서 이미 있는 메뉴를 입력한 경우>



<입력 예외처리 – 3번 메뉴에서 없는 메뉴를 삭제하려 하는 경우>

1. **토론**

* 데이터 파싱 함수 구현 과정에서 find 함수를 이용하는 과정에서 VS에서는 문제가 없었는데 linux에서는 이상한 문제가 발생했다. getline으로 받아온 문자열을 한 번 일일이 (int)형으로 건드리지 않으면 index를 사용해서 find 함수를 사용하게 되면 에러가 난다.
* 전반적으로 데이터 파싱이 너무 어려웠다. C++은 다른 언어와는 다르게 스플릿 함수가 없어서 사용자가 직접 구현을 해야 하는데, 나는 strtok 함수와 find 함수를 혼용했다. 이 과정에서 VS에서는 되지만 linux에서는 세그멘테이션 에러가 나는 때가 종종 있었기 때문에 많이 난감했었다. 그럼에도 불구하고 다행히 파싱을 구현하는 데에는 성공했다.
* 파일 파싱 말고도 메뉴 구현이 관건이었는데, 이 부분은 오히려 한 번에 성공해서 추가적인 예외처리를 제외하고는 크게 어려움이 없었다. 다만 입력 예외 처리를 위해 처음에는 cin 함수를 사용했으나 띄어쓰기가 포함된 문자열에도 대응하기 위해서 getline 함수로 바꾸는 과정이 있었다. 이 과정이 조금 힘들었고 이를 진행하면서 switch 문도 전부 if else문으로 바꿔야 했기에 시간이 조금 걸렸었다.

1. **결론**

* 이번 과제에서 요구한 메뉴 구현과 메뉴 처리, 그리고 데이터 파싱 모두를 성공적으로 구현했다.
* Class의 개념을 이용할 수 있었고, 파일을 분할해 프로그램을 구현할 수 있었으며 linked list를 이용해 프로그햄을 구현할 수 있었다.
* 과제를 구현하면서 다양한 함수(strtok / .c\_str() / find 등)를 사용하면서 익힐 수 있었다.
* 리눅스 환경에서 구현을 진행하면서 Makeflie을 만드는 방법을 익힐 수 있었다. 또한 이를 이용해 make 명령어로 컴파일링까지 실행할 수 있었다.

1. **개선 방향**

* 메뉴 추가 부분에서 금액이나 추가요금 부분에 숫자가 아닌 문자열이 입력되는 경우는 예외처리를 진행하지 못했다. 이 부분은 입력받은 문자열을 char\*로 변환한 뒤 하나하나 아스키코드와 비교해 숫자만 존재하는지 판단하는 방법으로 입력 오류를 판단할 수 있을 것이라 생각한다.
* 실수형 데이터를 입력받아야 하는 부분에서 문자열을 실수가 아닌지 어떤지를 인식할 수 있게 구현 가능할 것이다.
* 최대값과 최소값을 입력하는 부분에서 하한이 상한보다 크면 에러 메시지를 출력할 수 있게 구현할 수 있을 것이다.