

CSE2035 (C프로그래밍)

설계 프로젝트: 쇼핑몰 시스템 구현

서강대학교 컴퓨터공학과 박수현 (20181634)

담당교수: 서강대학교 컴퓨터공학과 김지환

1 설계 문제 및 목표

본 프로젝트에서는 C언어를 활용하여 사용자와 관리자의 관점에서 쇼핑몰 시스템을 구현한다.

1.1 현실적 제한조건

다음과 같은 기능을 구현한다.

공통 기능: 로그인하기

관리자 기능: 회원 정보 조회, 상품 정보 등록, 상품 정보 삭제, 상품 정보 수정, 상품 정보 및 구매 내역 조회, 상품 정보 및 구매 내역 통계, 배송 내역 수정

사용자 기능: 회원가입, 회원 정보 수정, 가상계좌 입금 및 출금, 상품 검색 및 조회, 상품 구매, 구매 내역 조회, 배송 내역 조회, 회원 탈퇴

2 요구사항

2.1 합성

이 프로젝트는 크게 사용자 로그인이 필요한 기능(이하 사용자 기능), 관리자 로그인이 필요한 기능(이하 관리자 기능), 로그인이 필요하지 않은 기능으로 나눌 수 있다. 로그인을 하지 않은 상태에서는 쇼핑몰에 등록되어 있는 상품을 이름, 카테고리, 가격대로 검색하며 둘러볼 수 있다.

사용자로 로그인할 경우 다음과 같은 기능을 제공한다.

회원 가입: 사용자가 새로운 회원으로 가입할 수 있는 기능이다. 추가적으로 ID 형식 및 중복 검사, 비밀번호 확인 일치 검사를 수행한다.

회원 정보 수정 및 회원 탈퇴: 사용자가 회원 정보를 수정할 수 있는 기능이다. 이름, 비밀번호, 주소를 수정하거나 쇼핑몰에서 탈퇴할 수 있는 기능을 제공한다.

가상계좌 입금 및 출금: 사용자가 가상계좌 잔액을 편집할 수 있는 기능이다. 가상계좌에 입금하거나 출금할 수 있는 기능을 제공한다.

상품 검색 및 조회: 사용자가 상품을 검색하고 조회할 수 있는 기능이다. 사용자에게는 현재 판매 중인 상품만 표시하도록 했으며, 이름, 카테고리, 가격대 검색을 통해 원하는 상품을 필터할 수 있는 기능을 제공한다. 또한 상품 조회 페이지에서 상품 구매 페이지로 이동할 수 있다.

상품 구매: 사용자가 상품을 구매할 수 있는 기능이다. 가상계좌 잔액을 확인해 구매 후 잔액 혹은 부족한 금액을 계산해 주고, 잔액이 부족할 경우 가상계좌 입금 페이지로 이동할 수 있는 기능도 제공한다.

구매 내역 조회: 사용자가 구매 내역을 조회할 수 있는 기능이다. 전체 구매 내역을 조회할 수 있다. 구매 내역마다 상품명과 카테고리, 가격이 표시된다.

배송 내역 조회: 사용자가 배송 내역을 조회할 수 있는 기능이다. 전체 구매 내역을 조회할 수 있다.

또한, 관리자로 로그인할 경우 다음과 같은 기능을 제공한다.

회원 정보 조회: 관리자가 전체 회원 정보를 조회할 수 있다.

상품 정보 등록: 관리자가 새로운 상품 정보를 등록할 수 있다. 여러 항목을 연속해서 등록할 수 있는 기능도 제공한다.

상품 정보 삭제: 관리자가 기존 상품 정보를 상품 코드를 기반으로 삭제할 수 있다. 여러 항목을 연속해서 삭제할 수도 있다.

상품 정보 수정: 관리자가 기존 상품 정보를 수정할 수 있다. 이름, 카테고리, 가격, 상품 상태를 새롭게 정할 수 있다.

상품 조회 및 통계: 관리자가 상품 정보와 통계를 조회할 수 있다. 상품은 이름, 카테고리, 가격대로 필터할 수 있으며, 각각 상품에 대해 판매량 조회가 가능하다.

구매 내역 및 통계: 관리자가 구매 내역과 통계를 조회할 수 있다. 구매 내역은 구매자의 ID와 상품 코드, 배송 상태로 필터할 수 있다.

배송 상태 수정: 관리자가 구매 내역의 배송 상태를 수정할 수 있다. 구매 내역은 구매자의 ID와 상품 코드, 배송 상태로 필터할 수 있다.

2.2 분석

메서드들은 여러 소스에 나뉘어 존재한다.

소스	역할
main.c	프로그램의 시작점. 프로그램에 필요한 파일들을 점검하고 프로그램을 시작한다.
language.c	프로그램에 등장하는 문자열들을 정의한다.
util/util_file.c	파일 입출력에 사용되는 유ти리티 메서드들의 집합이다.
util/util_io.c	사용자 입출력에 사용되는 유ти리티 메서드들의 집합이다.
util/util_string.c	문자열 처리에 사용되는 유ти리티 메서드들의 집합이다.
types/linked_list.c	연결 리스트(LinkedList)의 구현과 연결 리스트 수정에 필요한 메서드들의 집합이다.
types/member.c	사용자 구조체(Member)와 사용자 구조체 연결 리스트 (MemberList)의 구현을 주로 하여 사용자 구조체 연결 리스트 수정, 파싱(parsing)과 시리얼라이징(serializing), 그리고 불러오기와 저장에 필요한 메서드들의 집합이다.
types/order.c	주문 정보 구조체(OrderHistory)와 주문 정보 구조체 연결 리스트(OrderList)의 구현을 주로 한다. 기타 구현되어 있는 메서드들은 types/member.c와 크게 다르지 않다.
types/product.c	상품 정보 구조체(Product)와 상품 정보 구조체 연결 리스트 (ProductList)의 구현을 주로 한다. 기타 구현되어 있는 메서드들은 types/member.c와 크게 다르지 않다.
flow/main_menu.c	프로그램의 메인 메뉴.
flow/search_product_page.c	상품 정보 둘러보기 페이지.
flow/user/login_page.c	사용자 로그인 페이지.
flow/user/register_page.c	사용자 회원가입 페이지.
flow/user/user_page.c	사용자 기능들을 제공하는 메서드들의 집합이다.
flow/admin/admin_login_page.c	관리자 로그인 페이지.
flow/admin/admin_page.c	관리자 기능들을 제공하는 메서드들의 집합이다.

main.c main.c는 프로그램의 시작점이다.

int main(): 프로그램의 작동에 필요한 파일들을 점검하고 프로그램을 시작한다.

void initialize(): 프로그램의 작동에 필요한 파일들이 없을 경우 새로 생성한다.

language.c language.c는 프로그램에 등장하는 문자열들을 정의한다.

struct Language: 프로그램에 필요한 문자열들을 정의한다.

Language lang_korean(): 프로그램에 필요한 한국어 문자열들을 제공한다.

util/util_file.c util/util_file.c는 파일 입출력에 사용되는 유ти리티 메서드들의 집합이다.

int file_exists(char *filename): 이름이 filename인 파일이 존재하는지 확인한다.

void file_write(char *filename, char *contents): 이름이 filename인 파일을 새로 만들고 contents를 적는다.

void file_writeln(char *filename, char *contents): 이름이 filename인 파일을 새로 만들고 contents를 적는다. 마지막에 개행 문자(0x0A)를 추가한다.

© 2018 Suhyun Park

All Rights Reserved.

util/util_io.c util/util_io.c는 사용자 입출력에 사용되는 유ти리티 메서드들의 집합이다.

void scan_valid_id(char *temp, char *error_message, char *retry_message): 유효한 ID를 temp에 입력받는다. ID는 영어 대소문자와 숫자, 언더스코어(_)로만 구성되는 4~16자리 문자열이다. 유효하지 않은 값이 입력될 경우 error_message를 출력한다.

void scan_valid_string(char *temp, char *error_message, char *retry_message): 유효한 CSV 셀을 temp에 입력받는다. 쉼표(,) 혹은 따옴표(")를 포함한 값이 입력될 경우 error_message를 출력한다.

int scan_valid_boolean(char *error_message, char *retry_message): 유효한 0 혹은 1 값을 입력받는다. 대소문자를 구별하지 않고 y, 1, true, yes 중 하나가 입력될 경우 1, n, 0, false, no 중 하나가 입력될 경우 0을 반환한다.

int scan_valid_int(int min, int max, char *error_message, char *retry_message): min과 max 사이의 정수를 입력받는다. 범위를 초과하는 값이 입력될 경우 error_message를 출력한다.

util/util_string.c util/util_string.c는 문자열 처리에 사용되는 유ти리티 메서드들의 집합이다.

#define CLEAR_INPUT_BUFFER: 현재 인풋 버퍼를 초기화한다. scanf를 사용한 후에 fgets를 사용하면 의도한 대로 동작하지 않는데, 이 현상을 해결해 주기 위한 매크로 함수이다.

void string_itoa(char *str, int value): value를 문자열로 변환한다.

void string_to_lowercase(char *str): str 내의 문자들을 전부 소문자로 변환한다.

void string_to_uppercase(char *str): str 내의 문자들을 전부 대문자로 변환한다.

void string_invalid(char *str): str가 유효한 CSV 셀인지 검사한다. 쉼표(,) 혹은 따옴표("")를 포함하거나 빈 값일 경우 0, 아닐 경우 1이다.

void string_id_invalid(char *str): str가 유효한 ID인지 검사한다. 영어 대소문자와 숫자, 언더스코어(_)로만 구성되는 4-16자리 문자열일 경우 1, 아닐 경우 0이다.

char *string_trim(char *str): str 앞뒤의 공백을 제거한다.

void string_mask(char *str, char c): str 내의 모든 문자를 c로 교체한다.

© 2018 Suhyun Park
All Rights Reserved.

types/linked_list.c types/linked_list.c는 연결 리스트(LinkedList)^[1]의 구현과 연결 리스트 수정에 필요한 메서드들의 집합이다.

struct LinkedList: 연결 리스트 구조체이다.

enum GenericType: 연결 리스트 관련 함수 사용에 필요한 타입 변수들이다. 자세한 내용은 2.3에서 후술한다.

LinkedList *list_create(): 새로운 LinkedList를 생성한다.

void list_append(LinkedList *list, enum GenericType type, void *node): list 맨 뒤에 node를 삽입한다.

void list_removeLast(LinkedList *list, enum GenericType type): list의 마지막 노드를 제거한다.

void *list_get(LinkedList *list, enum GenericType type, int idx): list의 idx번째 노드를 찾는다.

list_set(LinkedList *list, enum GenericType type, void *value, int idx): list의 idx 번째 노드의 값을 value로 교체한다.

void list_free(LinkedList *list, enum GenericType type): list를 메모리에서 제거한다.

types/member.c types/member.c는 사용자 구조체(Member)와 사용자 구조체 연결 리스트(MemberList)의 구현을 주로 하여 사용자 구조체 연결 리스트 수정, 파싱(parsing)과 시리얼라이징(serializing), 그리고 불러오기와 저장에 필요한 메서드들의 집합이다.

enum MemberState: 사용자 상태를 나타내는 enumerable이다.

struct Member: 사용자 구조체이다.

struct MemberNode: MemberList를 이루는 노드 구조체이다.

#define null_member: null을 의미하는 Member이다.

void member_serialize(char *str, Member member): member를 CSV 형식으로 시리얼라이즈 한다.

Member parse_member(char *str): CSV 형태의 문자열을 str을 Member로 파싱한다.

MemberList *member_list_create(): 새로운 MemberList를 생성한다.

void member_list_append(MemberList *list, Member member): list 맨 뒤에 member를 삽입 한다.

void member_list_removelast(MemberList *list): list의 마지막 노드를 제거한다.

Member member_list_get(MemberList *list, int idx): list의 idx번째 Member를 찾는다.

void member_list_set(MemberList *list, Member member, int idx): list의 idx번째 Member의 값을 member로 교체한다.

int member_id_exists(MemberList *list, char *id): list에 ID id를 사용하는 Member가 존재하는지 확인한다.

Member member_with_id(MemberList *list, char *id): list에서 ID가 id인 Member를 찾아 반환한다. 없을 경우 null_member를 대신 반환한다.

MemberList *member_get_all(): customer.csv를 파싱해 저장된 사용자들을 전부 불러온다.

void member_print_all(MemberList *list, Language lang): lang 언어를 사용해 list에 저장 된 사용자 정보를 전부 출력한다.

void member_put_all(MemberList *list): customer.csv에 list에 존재하는 모든 사용자 정보를 저장한다. 기존에 존재하던 파일을 덮어쓴다.

types/order.c types/order.c는 구매 내역 구조체(OrderHistory)와 구매 내역 구조체 연결 리스트(OrderList)의 구현을 주로 하여 구매 내역 구조체 연결 리스트 수정, 파싱(parsing)과 시리얼라이징(serializing), 그리고 불러오기와 저장에 필요한 메서드들의 집합이다.

enum ParcelState: 배송 상태를 나타내는 enumerable이다.

struct OrderHistory: 구매 내역 구조체이다.

struct OrderNode: OrderList를 이루는 노드 구조체이다.

#define null_order: null을 의미하는 OrderHistory이다.

void order_serialize(char *str, OrderHistory orderHistory): orderHistory를 CSV 형식으로 시리얼라이즈한다.

OrderHistory parse_order(char *str): CSV 형태의 문자열을 str을 OrderHistory로 파싱한다.

OrderList *order_list_create(): 새로운 OrderList를 생성한다.

void order_list_append(OrderList *list, OrderHistory order): list 맨 뒤에 order를 삽입한다.

void order_list_removeLast(OrderList *list): list의 마지막 노드를 제거한다.

OrderHistory order_list_get(OrderList *list, int idx): list의 idx번째 OrderHistory를 찾는다.

OrderHistory order_list_get_by_idx(OrderList *list, int idx): list에서 주문번호가 idx인 OrderHistory를 찾아 반환한다. 없을 경우 null_order를 대신 반환한다.

void order_list_set(OrderList *list, OrderHistory order, int idx): list의 idx번째 OrderHistory의 값을 order로 교체한다.

OrderList *order_query_by_member_id(OrderList *list, int member_id): list에서 주문자의 회원번호가 member_id인 OrderHistory들을 찾아 새로운 OrderList를 생성해 반환한다.

OrderList *order_query_by_product_id(OrderList *list, int product_id): list에서 제품 번호가 product_id인 OrderHistory들을 찾아 새로운 OrderList를 생성해 반환한다.

OrderList *order_query_undelivered(OrderList *list): list에서 아직 배송 출발하지 않은 OrderHistory들을 찾아 새로운 OrderList를 생성해 반환한다.

OrderList *order_get_all(): history.csv를 파싱해 저장된 구매 내역을 전부 불러온다.

void order_print_all(OrderList *list, Language lang): lang 언어를 사용해 list에 저장된 구매 내역 정보를 전부 출력한다.

void order_print(OrderHistory order, Language lang): order를 lang 언어로 출력한다.

void order_put_all(OrderList *list): history.csv에 list에 존재하는 모든 구매 내역 정보를 저장한다. 기존에 존재하던 파일을 덮어쓴다.

types/product.c types/product.c는 상품 구조체(Product)와 상품 구조체 연결 리스트(ProductList)의 구현을 주로 하여 상품 구조체 연결 리스트 수정, 파싱(parsing)과 시리얼라이징(serializing), 그리고 불러오기와 저장에 필요한 메서드들의 집합이다.

enum ProductState: 상품 판매 상태를 나타내는 enumerable이다.

struct Product: 상품 구조체이다.

struct ProductNode: ProductList를 이루는 노드 구조체이다.

#define null_product: null을 의미하는 Product이다.

void product_serialize(char *str, Product product): product를 CSV 형식으로 시리얼라이즈한다.

Product parse_product(char *str): CSV 형태의 문자열을 str을 Product로 파싱한다.

ProductList *product_list_create(): 새로운 ProductList를 생성한다.

void product_list_append(ProductList *list, Product product): list 맨 뒤에 product를 삽입한다.

void product_list_removelast(ProductList *list): list의 마지막 노드를 제거한다.

Product product_list_get(ProductList *list, int idx): list의 idx번째 Product를 찾는다.

void product_list_set(ProductList *list, Product product, int idx): list의 idx번째 Product의 값을 product로 교체한다.

Product product_list_get_by_code(*ProductList *list, int code***):** list에서 상품코드가 code인 Product를 찾아 반환한다. 없을 경우 null_order를 대신 반환한다.

ProductList *product_query_by_name(*ProductList *list, char *keyword, int query_ready***):** list에서 이름에 keyword를 포함하는 Product들을 찾아 새로운 ProductList를 생성해 반환한다. query_ready가 1일 경우 판매 준비 중인 상품도 쿼리하고, 0일 경우 판매 비 중인 상품은 쿼리하지 않는다.

ProductList *product_query_by_category(*ProductList *list, char *keyword, int query_ready***):** list에서 카테고리가 keyword와 일치하는 Product들을 찾아 새로운 ProductList를 생성해 반환한다. query_ready가 1일 경우 판매 준비 중인 상품도 쿼리하고, 0일 경우 판매 비 중인 상품은 쿼리하지 않는다.

ProductList *product_query_by_price(*ProductList *list, int min, int max, int query_ready***):** list에서 가격이 min 이상 max 이하인 Product들을 찾아 새로운 ProductList를 생성해 반환한다. query_ready가 1일 경우 판매 준비 중인 상품도 쿼리하고, 0일 경우 판매 비 중인 상품은 쿼리하지 않는다.

ProductList *product_query_available(*ProductList *list, int query_ready***):** list에서 삭제되지 않은 Product들을 찾아 새로운 ProductList를 생성해 반환한다. query_ready가 1일 경우 판매 준비 중인 상품도 쿼리하고, 0일 경우 판매 비 중인 상품은 쿼리하지 않는다.

ProductList *product_get_all(): product.csv를 파싱해 저장된 상품들을 전부 불러온다.

void product_print_all(*ProductList *list, Language lang***):** lang 언어를 사용해 list에 저장된 상품 정보를 전부 출력한다.

void product_admin_print_all(*ProductList *list, Language lang***):** lang 언어를 사용해 list에 저장된 상품 정보를 현재 상품 상태와 함께 전부 출력한다.

void product_put_all(*ProductList *list***):** product.csv에 list에 존재하는 모든 상품 정보를 저장한다. 기존에 존재하던 파일을 덮어쓴다.

int product_sales(*Product product***):** product의 판매량을 반환한다.

flow/main_menu.c flow/main_menu.c는 프로그램의 메인 메뉴이다.

void main_menu(): 메인 메뉴.

flow/search_product_page.c flow/search_product_page.c는 상품 정보 둘러보기 페이지이다.

void search_product_page(): 상품 둘러보기 페이지.

flow/user/login_page.c flow/user/login_page.c는 사용자 로그인 페이지이다.

void login_page(): 사용자 로그인 페이지.

flow/user/register_page.c flow/user/register_page.c는 사용자 회원가입 페이지이다.

void register_page(): 사용자 회원가입 페이지.

flow/user/user_page.c flow/user/user_page.c는 사용자 기능들을 제공하는 메서드들의 집합이다.

void user_page(Member *session): 현재 로그인한 사용자의 사용자 페이지이다.

int user_edit_credentials(Member *session): 현재 로그인한 사용자의 회원 정보를 수정하는 페이지이다. 현재 사용자가 탈퇴할 경우 1을, 아닐 경우 0을 반환한다.

void user_edit_balance(Member *session): 현재 로그인한 사용자의 가상계좌 입출금 페이지이다.

void user_search_product(Member *session): 현재 로그인한 사용자가 쇼핑몰에 등록된 상품을 검색하는 페이지이다.

void user_product_list(Member *session, ProductList *products): 현재 로그인한 사용자가 쇼핑몰에 등록된 상품을 검색한 후 그 검색 결과를 보여주는 페이지이다.

void user_lookup_product(Member *session, Product product): 현재 로그인한 사용자가 개별 상품을 조회하는 페이지이다.

void user_purchase_product(Member *session, Product product): 현재 로그인한 사용자가 상품을 구매하는 페이지이다.

void user_search_orders(Member *session): 현재 로그인한 사용자의 주문을 조회하는 페이지이다.

void user_search_parcels(Member *session): 현재 로그인한 사용자의 주문들의 배송 상태를 조회하는 페이지이다.

flow/admin/admin_login_page.c flow/admin/admin_login_page.c는 관리자 로그인 페이지이다.

void admin_login_page(): 관리자 로그인 페이지.

flow/admin/admin_page.c flow/admin/admin_page.c는 관리자 기능들을 제공하는 메서드들의 집합이다.

void admin_page(): 관리자 페이지이다.

void admin_lookup_customers(): 관리자가 전체 회원 정보를 조회하는 페이지이다.

void admin_add_or_delete_products(): 관리자가 상품을 새로 등록하거나 기존 상품을 삭제하는 페이지이다.

void admin_add_product(): 관리자가 상품을 새로 등록하는 페이지이다.

void admin_remove_product(): 관리자가 상품을 삭제하는 페이지이다.

void admin_edit_products(): 관리자가 상품 편집을 위해 전체 상품을 조회하는 페이지이다.

void admin_edit_product(Product product): 관리자가 단일 상품을 편집하는 페이지이다.

void admin_search_product(): 관리자가 상품을 검색하는 페이지이다.

void admin_product_list(ProductList *products): 관리자가 검색한 상품들의 목록을 조회하는 페이지이다.

void admin_lookup_product(Product product): 관리자가 단일 상품을 조회하는 페이지이다.

void admin_lookup_orders(): 관리자가 구매 내역을 검색하는 페이지이다.

void admin_order_list(OrderList* orders): 관리자가 검색한 주문 내역들의 목록을 조회하는 페이지이다.

void admin_lookup_parcel_status(): 관리자가 배송 내역을 조회하기 위해 구매 내역을 검색하는 페이지이다.

void admin_order_list_select_to_edit(OrderList* orders): 관리자가 배송 상태를 편집하기 위해 검색된 구매 내역들을 조회하는 페이지이다.

void admin_edit_parcel_status(OrderHistory order): 관리자가 단일 구매 내역의 배송 상태를 편집하는 페이지이다.

2.3 제작

2.2에서 충분히 설명된 간단한 메서드들은 2.2에서의 설명을 그대로 적었다.

main.c main.c는 프로그램의 시작점이다.

int main(): 프로그램의 작동에 필요한 파일들을 점검하고 프로그램을 시작한다.

void initialize(): 프로그램의 작동에 필요한 파일들이 없을 경우 새로 생성한다. 파일 존재 여부는 util_file.c에 있는 file_exists() 메서드로 확인하며, customer.csv, product.csv, history.csv 가 각각 존재하지 않을 경우 새로 생성된다.

© 2018 Suhyun Park
All Rights Reserved.

struct Language: 프로그램에 필요한 문자열들을 정의한다. Language는 많은 char*를 포함하고 있는 하나의 거대한 구조체로서 2.7에서 후술할 확장성을 위해 만들어졌다.

Language lang_korean(): 프로그램에 필요한 한국어 문자열들을 제공한다.

util/util_file.c util/util_file.c는 파일 입출력에 사용되는 유틸리티 메서드들의 집합이다.

int file_exists(char *filename): 이름이 filename인 파일이 존재하는지 확인한다. 파일을 읽으려고 시도했을 때 생성된 FILE*이 NULL일 경우 파일이 존재하지 않는다고 가정한다.

void file_write(char *filename, char *contents): 이름이 filename인 파일을 새로 만들고 contents를 적는다.

void file_writeln(char *filename, char *contents): 이름이 filename인 파일을 새로 만들고 contents를 적는다. 마지막에 개행 문자(0x0A)를 추가한다.

util/util_io.c `util/util_io.c`는 사용자 입출력에 사용되는 유ти리티 메서드들의 집합이다.

void scan_valid_id(char *temp, char *error_message, char *retry_message): 유효한 ID를 `temp`에 입력받기 위해, 입력받은 값이 `string_id_invalid()`일 경우 ID 포맷에 맞는 값을 계속 요구한다.

void scan_valid_string(char *temp, char *error_message, char *retry_message): 유효한 CSV 셀을 `temp`에 입력받기 위해, 입력받은 값이 `string_invalid()`일 경우 CSV 셀 포맷에 맞는 값을 계속 요구한다.

int scan_valid_boolean(char *error_message, char *retry_message): 유효한 0 혹은 1 값을 입력받기 위해, 일단 입력받은 값을 `string_to_lowercase()`를 이용해 소문자로 전부 바꾸고, y, 1, true, yes 중 하나 혹은 n, 0, false, no 중 하나와 일치하는지 확인한다. 일치하는 값이 없을 경우 포맷에 맞는 값을 계속 요구한다.

int scan_valid_int(int min, int max, char *error_message, char *retry_message): `min`과 `max` 사이의 정수를 입력받는다. 범위를 초과하는 값이 입력될 경우 범위 안의 값을 계속 요구한다.

© 2018 Suhyun Park
All Rights Reserved.

util/util_string.c `util/util_string.c`는 문자열 처리에 사용되는 유ти리티 메서드들의 집합이다.

#define CLEAR_INPUT_BUFFER: 현재 인풋 버퍼를 초기화한다^[2]. 버퍼에 남은 `char`가 `n`이나 `EOF`가 아닐 때까지 문자를 계속 읽어들인다.

void string_itoa(char *str, int value): `value`를 문자열로 변환해 `str`로 반환한다. 일단 정수의 10진수 표현의 크기를 계산해 포인터를 그만큼 뒤로 옮기고, 포인터의 위치에 `value % 10`을 채우고 `value`의 값을 10으로 나누고 포인터를 왼쪽으로 한 칸 미는 것을 반복하여 문자열에 남은 빈 공간을 전부 채워 넣는다. 시간 복잡도는 $O(\log(value))$ 이다.

void string_to_lowercase(char *str): `str` 내의 문자들을 전부 소문자로 변환한다. `str` 내의 모든 문자에 대해서 만약 어떤 문자가 대문자 A와 대문자 Z 사이에 있다면 `0x20`을 더해 소문자로 만든다. 시간 복잡도는 $O(|str|)$ 이다.

void string_to_uppercase(char *str): `str` 내의 문자들을 전부 대문자로 변환한다. `str` 내의 모든 문자에 대해서 만약 어떤 문자가 소문자 a와 소문자 z 사이에 있다면 `0x20`을 빼 대문자로 만든다. 시간 복잡도는 $O(|str|)$ 이다.

void string_invalid(char *str): str가 유효한 CSV 셀인지 검사한다. str 내의 모든 문자에 대해서 만약 어떤 문자가 쉼표 혹은 마침표라면 1을 반환하고, 그런 문자가 하나도 없다면 0을 반환한다. 시간 복잡도는 $O(|str|)$ 이다.

void string_id_invalid(char *str): str가 유효한 ID인지 검사한다. str의 길이가 4부터 16 까지의 범위를 벗어날 경우 1을 반환하고, 아닐 경우 모든 문자에 대해 그 문자가 A와 Z, 혹은 a와 z, 혹은 0과 9 사이에 있거나 -(언더스코어)인지 확인한다. 모든 문자가 이 조건을 남족할 경우 0, 하나라도 만족하지 않을 경우 1을 반환한다. 시간 복잡도는 $O(|str|)$ 이다.

char *string_trim(char *str): str 앞뒤의 공백을 제거한다. 앞 부분이 공백이 아닐 때까지 str의 포인터를 뒤로 밀고, 새로운 end 포인터를 선언해 str의 맨 뒤로 위치하게 한 후, 뒷 부분이 공백이 아닐 때까지 end를 앞으로 밀고 그 뒷 위치에 NUL을 놓는다. 시간 복잡도는 $O(|str|)$ 이다.

void string_mask(char *str, char c): str 내의 모든 문자를 c로 교체한다. 모든 문자에 대해 수행한다. 시간 복잡도는 $O(|str|)$ 이다.

types/linked_list.c types/linked_list.c는 연결 리스트(LinkedList)^[1]의 구현과 연결 리스트 수정에 필요한 메서드들의 집합이다.

struct LinkedList: 연결 리스트 구조체이다. head, tail, size로 구성되어 있다. 노드는 모두 void* 형이다.

enum GenericType: 연결 리스트 관련 함수 사용에 필요한 타입 변수들이다. C11 표준에서는 제네릭 프로그래밍(Generic Programming)을 사용할 수 있으나 C99 표준에는 없기에 제네릭 프로그래밍을 흉내내고자 도입한 enum이다. LinkedList 관련 메서드들에 이 enum을 넘겨주면 메서드들이 노드의 자료형을 파악해 관련된 동작을 수행한다.

LinkedList *list_create(): 크기가 0인 새로운 LinkedList를 생성한다. head와 tail은 NULL로, size는 0으로 초기화된다.

void list_append(LinkedList *list, enum GenericType type, void *node): list 맨 뒤에 node를 삽입한다. head와 tail이 모두 NULL일 경우 head와 tail을 모두 node로 설정하고, 아닐 경우 현재 tail의 링크를 새 node로 설정한 뒤 tail을 node로 다시 설정한다. 이렇게 할 경우 마지막에서 두 번째에 있는 노드의 링크는 마지막에서 첫 번째에 있는 노드를 참조하게 되며 성공적으로 연결 리스트에 새 노드를 삽입하게 된다. 시간 복잡도는 $O(1)$ 이다.

void list_removelast(LinkedList *list, enum GenericType type): list의 크기가 0이 아니라면 마지막 노드를 제거한다. tail의 전 노드를 접근할 수 있는 방법이 없으므로 head에서부터 마지막에서 두 번째에 있는 노드까지를 순회한 뒤 맨 마지막 노드를 메모리에서 삭제하고, tail을 마지막에서 두 번째에 있었던 노드로 바꿔 준다. 시간 복잡도는 $O(|list|)$ 이다.

void *list_get(LinkedList *list, enum GenericType type, int idx): list의 idx번째 노드를 찾는다. head부터 차례차례 순회하는 방식으로 idx번째 노드를 찾을 수 있다. 시간 복잡도는 $O(idx)$ 이다.

list_set(LinkedList *list, enum GenericType type, void *value, int idx): list의 idx 번째 노드의 값을 value로 교체한다. head부터 차례차례 순회하는 방식으로 idx번째 노드에 접근해 값을 바꿀 수 있다. 시간 복잡도는 $O(idx)$ 이다.

void list_free(LinkedList *list, enum GenericType type): list를 메모리에서 제거한다. head부터 차례차례 순회하나, 접근 직후에 free()를 호출하면 다음 노드에 접근할 수 있는 링크까지도 사라져 버리므로 curr와 prev 두 개의 포인터를 활용하여 순회하면서 모든 노드를 메모리에서 제거하였다. 시간 복잡도는 $O(|list|)$ 이다.

types/member.c types/member.c는 사용자 구조체(Member)와 사용자 구조체 연결 리스트(MemberList)의 구현을 주로 하여 사용자 구조체 연결 리스트 수정, 파싱(parsing)과 시리얼라이징(serializing), 그리고 불러오기와 저장에 필요한 메서드들의 집합이다.

enum MemberState: 사용자 상태를 나타내는 enumerable이다. 탈퇴 상태일 경우 Unregistered, 가입 상태일 경우 Registered의 값을 가질 수 있다.

struct Member: 사용자 구조체이다. _idx는 회원번호, id는 ID, password는 비밀번호, name은 이름, address는 주소, balance는 가상계좌 잔액, state는 멤버 상태를 저장하고 있다.

struct MemberNode: MemberList를 이루는 노드 구조체이다. 값 member와 다음 노드로의 링크 next_ptr를 저장하고 있다.

#define null_member: null을 의미하는 Member이다. _idx가 -1인 것이 특징이다. 다른 메서드들에서 null 멤버를 반환했는지를 확인하려면 _idx가 -1인지를 확인하면 된다.

void member_serialize(char *str, Member member): member를 CSV 형식으로 시리얼라이즈한다. strcat, string_itoa를 활용하여 str에 덮어씌워 Call by reference로 반환한다.

Member parse_member(char *str): strtok과 atoi를 활용하여 CSV 형태의 문자열을 str을 Member로 파싱한다.

MemberList *member_list_create(): 새로운 MemberList를 생성한다. types/linked_list.c의 list_create()와 동치이다.

void member_list_append(MemberList *list, Member member): list 맨 뒤에 member를 삽입한다. 노드를 하나 생성하고 list_append(list, MemberType, node)를 호출한다.

void member_list_removelast(MemberList *list): list의 마지막 노드를 제거한다. list_removelast(list, MemberType)를 호출한다.

Member member_list_get(MemberList *list, int idx): list의 idx번째 Member를 찾는다. list_get(list, MemberType, idx)를 호출하고 이를 MemberNode*로 캐스팅한 후 값을 가져온다.

void member_list_set(MemberList *list, Member member, int idx): list의 idx번째 Member의 값을 member로 교체한다. list_set(list, MemberType, &member, idx)를 호출한다.

int member_id_exists(MemberList *list, char *id): list에 ID id를 사용하는 Member가 존재하는지 확인한다. head부터 순회해 ID가 id인 노드가 하나라도 존재하면 1을, 그렇지 않을 경우 0을 반환한다. 시간 복잡도는 $O(|list| |id|)$ 이다.

Member member_with_id(MemberList *list, char *id): list에서 ID가 id인 Member를 찾아 반환한다. head부터 순회해 ID가 id인 노드가 하나라도 존재하면 그 노드의 값을, 그렇지 않을 경우 null_member을 반환한다. 시간 복잡도는 $O(|list| |id|)$ 이다.

MemberList *member_get_all(): customer.csv를 파싱해 저장된 사용자들을 전부 불러온다. fgets를 사용해 줄마다 parse_member()를 하면서 연결 리스트의 마지막에 추가해가는 방식이다.

void member_print_all(MemberList *list, Language lang): lang 언어를 사용해 list에 저장된 사용자 정보를 전부 출력한다. head부터 순회하면서 출력한다. 시간 복잡도는 $O(|list|)$ 이다.

void member_put_all(MemberList *list): customer.csv에 list에 존재하는 모든 사용자 정보를 저장한다. 기존에 존재하던 파일을 덮어쓴다. 연결 리스트를 전부 순회하면서 각각의 노드에 member_serialize()를 한 것을 파일 맨 뒤에 붙여나가는 방식이다.

types/order.c types/order.c는 구매 내역 구조체(OrderHistory)와 구매 내역 구조체 연결 리스트(OrderList)의 구현을 주로 하여 구매 내역 구조체 연결 리스트 수정, 파싱(parsing)과 시리얼라이징(serializing), 그리고 불러오기와 저장에 필요한 메서드들의 집합이다.

enum ParcelState: 배송 상태를 나타내는 enumerable이다. 배송 준비 중일 경우 DeliverReady, 배송 중일 경우 Delivering, 배송이 완료되었을 경우 Delivered의 상태를 가질 수 있다.

struct OrderHistory: 구매 내역 구조체이다. _idx는 주문번호, product_idx는 상품코드, member_idx는 회원번호, state는 배송 상태를 저장하고 있다.

struct OrderNode: OrderList를 이루는 노드 구조체이다. 값 order와 다음 노드로의 링크 next_ptr를 저장하고 있다.

#define null_order: null을 의미하는 OrderHistory이다. _idx가 -1인 것이 특징이다. 다른 메서드들에서 null 주문을 반환했는지를 확인하려면 _idx가 -1인지를 확인하면 된다.

void order_serialize(char *str, OrderHistory orderHistory): orderHistory를 CSV 형식으로 시리얼라이즈한다. strcat, string_itoa를 활용하여 str에 덮어씌워 Call by reference로 반환한다.

OrderHistory parse_order(char *str): strtok과 atoi를 활용하여 CSV 형태의 문자열을 str을 OrderHistory로 파싱한다.

OrderList *order_list_create(): 새로운 OrderList를 생성한다. types/linked_list.c의 list_create()와 동치이다.

void order_list_append(OrderList *list, OrderHistory order): list 맨 뒤에 order를 삽입한다. 노드를 하나 생성하고 list_append(list, OrderType, node)를 호출한다.

void order_list_removelast(OrderList *list): list의 마지막 노드를 제거한다. list_removelast(list, OrderType)를 호출한다.

OrderHistory order_list_get(OrderList *list, int idx): list의 idx번째 OrderHistory를 찾는다. list_get(list, OrderType, idx)를 호출하고 이를 OrderNode*로 캐스팅한 후 값을 가져온다.

OrderHistory order_list_get_by_idx(OrderList *list, int idx): list에서 주문번호가 idx인 OrderHistory를 찾아 반환한다. head부터 순회해 주문번호가 idx인 노드가 하나라도 존재하면 그 노드의 값을, 그렇지 않을 경우 null_order를 반환한다. 시간 복잡도는 $O(|list||id|)$ 이다.

void order_list_set(OrderList *list, OrderHistory order, int idx): list의 idx번째 OrderHistory의 값을 order로 교체한다. list_set(list, OrderType, &order, idx)를 호출한다.

OrderList *order_query_by_member_id(OrderList *list, int member_id): list에서 주문자의 회원번호가 member_id인 OrderHistory들을 찾는다. 연결 리스트를 새로 만들어서 head부터 순회해 주문자의 회원번호가 member_id인 노드들을 새 연결 리스트의 뒤에 추가하고, 순회가 끝나면 새로 만들어진 연결 리스트를 반환한다. 시간 복잡도는 $O(|list||id|)$ 이다.

OrderList *order_query_by_product_id(OrderList *list, int product_id): list에서 제품번호가 product_id인 OrderHistory들을 찾는다. 연결 리스트를 새로 만들어서 head부터 순회해 제품번호가 product_id인 노드들을 새 연결 리스트의 뒤에 추가하고, 순회가 끝나면 새로 만들어진 연결 리스트를 반환한다. 시간 복잡도는 $O(|list||id|)$ 이다.

OrderList *order_query_undelivered(OrderList *list): list에서 아직 배송 출발하지 않은 OrderHistory들을 찾는다. 연결 리스트를 새로 만들어서 head부터 순회해 상태가 DeliverReady인 노드들을 새 연결 리스트의 뒤에 추가하고, 순회가 끝나면 새로 만들어진 연결 리스트를 반환한다. 시간 복잡도는 $O(|list||id|)$ 이다.

OrderList *order_get_all(): history.csv를 파싱해 저장된 구매 내역을 전부 불러온다. fgets를 사용해 줄마다 parse_order()를 하면서 연결 리스트의 마지막에 추가해가는 방식이다.

void order_print_all(OrderList *list, Language lang): lang 언어를 사용해 list에 저장된 구매 내역 정보를 전부 출력한다.

void order_print(OrderHistory order, Language lang): order를 lang 언어로 출력한다. head 부터 순회하면서 출력한다. 시간 복잡도는 $O(|list|)$ 이다.

void order_put_all(OrderList *list): history.csv에 list에 존재하는 모든 구매 내역 정보를 저장한다. 기존에 존재하던 파일을 덮어쓴다. 연결 리스트를 전부 순회하면서 각각의 노드에 order_serialize()를 한 것을 파일 맨 뒤에 붙여나가는 방식이다.

types/product.c types/product.c는 상품 구조체(Product)와 상품 구조체 연결 리스트(ProductList)의 구현을 주로 하여 상품 구조체 연결 리스트 수정, 파싱(parsing)과 시리얼라이징(serializing), 그리고 불러오기와 저장에 필요한 메서드들의 집합이다.

enum ProductState: 상품 판매 상태를 나타내는 enumerable이다. 상품이 삭제되어 있을 경우 Unavailable, 판매 준비 중일 경우 Ready, 판매 중일 경우 Available의 상태를 가질 수 있다.

struct Product: 상품 구조체이다. _idx는 상품코드, name은 상품 이름, category는 카테고리, price는 가격, state는 판매 상태를 저장하고 있다.

struct ProductNode: ProductList를 이루는 노드 구조체이다. 값 product와 다음 노드로의 링크 next_ptr를 저장하고 있다.

#define null_product: null을 의미하는 Product이다. _idx가 -1인 것이 특징이다. 다른 메서드들에서 null 상품을 반환했는지를 확인하려면 _idx가 -1인지를 확인하면 된다.

void product_serialize(char *str, Product product): product를 CSV 형식으로 시리얼라이즈한다. strcat, string_itoa를 활용하여 str에 덮어씌워 Call by reference로 반환한다.

Product parse_product(char *str): strtok과 atoi를 활용하여 CSV 형태의 문자열을 str을 Product로 파싱한다.

ProductList *product_list_create(): 새로운 ProductList를 생성한다. types/linked_list.c의 list_create()와 동치이다.

void product_list_append(ProductList *list, Product product): list 맨 뒤에 product를 삽입한다. 노드를 하나 생성하고 list_append(list, ProductType, node)를 호출한다.

void product_list_removelast(ProductList *list): list의 마지막 노드를 제거한다. list_removelast(list, ProductType)를 호출한다.

Product product_list_get(ProdutList *list, int idx): list의 idx번째 Product를 찾는다. list_get(list, ProductType, idx)를 호출하고 이를 ProductNode*로 캐스팅한 후 값을 가져온다.

void product_list_set(ProdutList *list, Product product, int idx): list의 idx번째 Product의 값을 product로 교체한다. list_set(list, ProductType, &product, idx)를 호출한다.

Product product_list_get_by_code(ProdutList *list, int code): list에서 상품코드가 code인 Product를 찾아 반환한다. 없을 경우 null_order를 대신 반환한다. head부터 순회해 상품코드가 code인 노드가 하나라도 존재하면 그 노드의 값을, 그렇지 않을 경우 null_product를 반환한다. 시간 복잡도는 $O(|list| |keyword|)$ 이다.

ProdutList *product_query_by_name(ProdutList *list, char *keyword, int query_ready): list에서 이름에 keyword를 포함하는 Product들을 찾는다. query_ready가 1일 경우 판매 준비 중인 상품도 쿼리하고, 0일 경우 비 중인 상품은 쿼리하지 않는다. 연결 리스트를 새로 만들어서 head부터 순회해 상품 이름에 keyword를 포함하는 노드들을 새 연결 리스트의 뒤에 추가하고, 순회가 끝나면 새로 만들어진 연결 리스트를 반환한다. 시간 복잡도는 $O(|list| |keyword|)$ 이다.

ProdutList *product_query_by_category(ProdutList *list, char *keyword, int query_ready): list에서 카테고리가 keyword와 일치하는 Product들을 찾는다. query_ready가 1일 경우 판매 준비 중인 상품도 쿼리하고, 0일 경우 비 중인 상품은 쿼리하지 않는다. 연결 리스트를 새로 만들어서 head부터 순회해 카테고리가 keyword와 일치하는 노드들을 새 연결 리스트의 뒤에 추가하고, 순회가 끝나면 새로 만들어진 연결 리스트를 반환한다. 시간 복잡도는 $O(|list| |keyword|)$ 이다.

ProdutList *product_query_by_price(ProdutList *list, int min, int max, int query_ready): list에서 가격이 min 이상 max 이하인 Product들을 찾는다. query_ready가 1일 경우 판매 준비 중인 상품도 쿼리하고, 0일 경우 판매 비 중인 상품은 쿼리하지 않는다. 연결 리스트를 새로 만들어서 head부터 순회해 가격이 min 이상 max 이하인 노드들을 새 연결 리스트의 뒤에 추가하고, 순회가 끝나면 새로 만들어진 연결 리스트를 반환한다. 시간 복잡도는 $O(|list|)$ 이다.

ProdutList *product_query_available(ProdutList *list, int query_ready): list에서 삭제되지 않은 Product들을 찾는다. query_ready가 1일 경우 판매 준비 중인 상품도 쿼리하고, 0일 경우 판매 비 중인 상품은 쿼리하지 않는다. 연결 리스트를 새로 만들어서 head부터 순회해 삭제되지 않은 상품 정보를 포함하는 노드들을 새 연결 리스트의 뒤에 추가하고, 순회가 끝나면 새로 만들어진 연결 리스트를 반환한다. 시간 복잡도는 $O(|list|)$ 이다.

ProdutList *product_get_all(): product.csv를 파싱해 저장된 상품들을 전부 불러온다. fgets를 사용해 줄마다 parse_product()를 하면서 연결 리스트의 마지막에 추가해가는 방식이다.

void product_print_all(ProdutList *list, Language lang): lang 언어를 사용해 list에 저장된 상품 정보를 전부 출력한다. head부터 순회하면서 출력한다. 시간 복잡도는 $O(|list|)$ 이다.

void product_admin_print_all(ProductList *list, Language lang): lang 언어를 사용해 list에 저장된 상품 정보를 현재 상품 상태와 함께 전부 출력한다. head부터 순회하면서 출력한다. 시간 복잡도는 $O(|list|)$ 이다.

void product_put_all(ProductList *list): product.csv에 list에 존재하는 모든 상품 정보를 저장한다. 기존에 존재하던 파일을 덮어쓴다. 연결 리스트를 전부 순회하면서 각각의 노드에 product_serialize()를 한 것을 파일 맨 뒤에 붙여나가는 방식이다.

int product_sales(Product product): product의 판매량을 반환한다. order_get_all()을 통해 전체 주문 리스트를 받아온 후 찾고자 하는 상품의 주문들을 head부터 순회하면서 찾고, 개수를 반환한다. 시간 복잡도는 $O(|orders|)$ 이다.

flow/main_menu.c flow/main_menu.c는 프로그램의 메인 메뉴이다.

void main_menu(): 메인 메뉴이다. 1에서 5 중 하나의 숫자를 입력받아 다음과 같은 동작을 실행한다.

입력값	동작
1	로그인 페이지 - login_page()
2	회원 가입 페이지 - register_page()
3	상품 검색 페이지 - search_product_page()
4	관리자 로그인 페이지 - admin_login_page()
5	프로그램 종료

flow/search_product_page.c flow/search_product_page.c는 상품 정보 둘러보기 페이지이다.

void search_product_page(): 상품 둘러보기 페이지이다. 1에서 5 중 하나의 숫자를 입력받아 다음과 같은 동작을 실행한다.

입력값	동작
1	이름으로 상품 검색 - <code>product_query_by_name()</code>
2	카테고리로 상품 검색 - <code>product_query_by_category()</code>
3	가격대로 상품 검색 - <code>product_query_by_price()</code>
4	전체 상품 보기 - <code>product_query_available()</code>
5	이전 화면으로 돌아가기

가격대 검색에서, 가격 범위의 최소값과 최대값을 각각 min, max 라고 하면, $0 \leq min \leq max \leq 2 \times 10^9$ 가 되도록 입력받는다.

이전 화면으로 돌아가지 않을 경우 검색된 상품 목록을 출력하고, 사용자가 이전 화면으로 돌아간다는 명령을 할 때까지 반복해서 숫자를 입력 받아 위와 같은 동작을 수행한다.

flow/user/login_page.c `flow/user/login_page.c`는 사용자 로그인 페이지이다.

void login_page(): 사용자 로그인 페이지이다. 아이디와 비밀번호를 입력받고, `member_with_id()`로 입력받은 ID를 갖는 회원을 탐색하고, 그 회원의 비밀번호와 입력받은 비밀번호를 대조한다. 일치하는 경우 탐색한 회원을 세션으로 하는 유저 페이지를 실행하고, 일치하지 않는 경우 3번까지 재시도하며 3번째에도 일치하지 않을 경우에는 메인 메뉴로 돌아간다.

뒤에 등장하는 모든 유저 페이지 관련 메서드들은 변수로 세션을 받게 되는데, 이는 현재 로그인한 유저의 정보를 쉽게 가져오고 조작하기 위함이다.

flow/user/register_page.c `flow/user/register_page.c`는 사용자 회원가입 페이지이다.

void register_page(): 사용자 회원가입 페이지이다. 이름, ID, 비밀번호, 주소를 입력 받아 회원 목록에 새 회원을 추가한다. 모든 문자열은 `util/util_io.c`에 존재하는 유효한 문자열 입력 함수로 입력받는다. 또한 회원 정보를 쿼리해 이미 존재하는 ID를 입력했을 경우 재입력을, 비밀번호 확인이 일치하지 않을 경우 재입력을 요구한다.

가입이 성공적으로 완료되면 회원의 가상계좌 잔액이 0으로, 회원의 상태가 '가입됨'(Registered)으로 초기화된다. 이후에는 새로 생성한 회원을 회원 리스트에 추가해 저장하고, 회원을 세션으로 하는 유저 페이지를 실행한다.

flow/user/user_page.c `flow/user/user_page.c`는 사용자 기능들을 제공하는 메서드들의 집합이다.

void user_page(Member *session): 현재 로그인한 사용자의 사용자 페이지이다. 1에서 6 중 하나의 숫자를 입력받아 다음과 같은 동작을 실행한다.

입력값	동작
1	회원 정보 수정 페이지 - <code>user_edit_credentials()</code>
2	가상계좌 잔액 편집 페이지 - <code>user_edit_balance()</code>
3	상품 검색 페이지 - <code>user_search_product()</code>
4	주문 검색 페이지 - <code>user_search_orders()</code>
5	배송 상태 조회 페이지 - <code>user_search_parcels()</code>
6	프로그램 종료

회원 정보 수정 페이지는 콜백(callback) 형식으로 작동한다. 회원 정보 수정 페이지에서 반환한 값이 1일 경우에는 현재 회원이 회원 정보 수정 페이지에서 탈퇴했음을 의미하므로, 즉시 프로그램을 종료한다.

사용자가 프로그램을 종료한다는 명령을 할 때까지 반복해서 숫자를 입력받아 위와 같은 동작을 수행한다. 이는 `while` 루프와 `exit_flag`으로 동작한다. 후술할 반복 수행 메서드들의 대부분을 이렇게 동작하도록 설계했다.

int user_edit_credentials(Member *session): 현재 로그인한 사용자의 회원 정보를 수정하는 페이지이다. 현재 사용자가 탈퇴할 경우 1을, 아닐 경우 0을 반환한다. 현재 회원 정보를 비밀번호만 `string_mask()`으로 가려서 출력하고, 1에서 5 중 하나의 숫자를 입력받아 다음과 같은 동작을 실행한다.

입력값	동작
1	이름 수정
2	비밀번호 수정
3	주소 수정
4	회원 탈퇴
5	이전 화면으로 돌아가기

모든 문자열은 `util/util_io.c`에 존재하는 유효한 문자열 입력 함수로 입력받는다. 새 정보를 입력받으면 현재 세션에 반영하며 `customer.csv`에도 새 정보를 덮어씌워 저장한다.

void user_edit_balance(Member *session): 현재 로그인한 사용자의 가상계좌 입출금 페이지이다. 1에서 3 중 하나의 숫자를 입력받아 다음과 같은 동작을 실행한다.

입력값	동작
1	가상계좌 입금 페이지
2	가상계좌 출금 페이지
3	이전 화면으로 돌아가기

사용자 계좌 잔액 정보는 int 형이므로 이론적으로 $2^{31} - 1 = 2,147,483,647$ 원^[3] 까지 입금할 수 있고, 이를 초과하면 오버플로(overflow) 오류가 일어나 잔액을 예측할 수 없게 된다. 따라서 입금할 경우 입금 후 잔액이 2,000,000,000 원이 될 때까지만 입금이 가능하도록 입금 금액을 제한한다. 출금할 경우에는 현재 잔액보다 많은 금액을 출금할 수 없도록 했다.

사용자가 이전 화면으로 돌아간다는 명령을 할 때까지 반복해서 숫자를 입력받아 위와 같은 동작을 수행한다.

void user_search_product(Member *session): 현재 로그인한 사용자가 쇼핑몰에 등록된 상품을 검색하는 페이지이다. 1에서 5 중 하나의 숫자를 입력받아 다음과 같은 동작을 실행한다.

입력값	동작
1	이름으로 상품 검색 - product_query_by_name()
2	카테고리로 상품 검색 - product_query_by_category()
3	가격대로 상품 검색 - product_query_by_price()
4	전체 상품 보기 - product_query_available()
5	이전 화면으로 돌아가기

가격 범위의 최소값과 최대값을 각각 min, max 라고 하면, $0 \leq min \leq max \leq 2 \times 10^9$ 가 되도록 입력받는다. 검색된 상품은 queried에 새 리스트로 메모리에 저장되고, user_product_list(session, queried)으로 넘겨주게 된다.

사용자가 이전 화면으로 돌아간다는 명령을 할 때까지 반복해서 숫자를 입력받아 위와 같은 동작을 수행한다.

void user_product_list(Member *session, ProductList *products): 현재 로그인한 사용자가 쇼핑몰에 등록된 상품을 검색한 후 그 검색 결과를 보여주는 페이지이다. 다른 함수가 넘겨준 products를 우선 product_print_all(products, lang)로 출력하고, 숫자를 하나 입력받는다. 입력받은 숫자가 0일 경우 이전 화면으로 돌아가고, 출력한 상품 코드 중에 입력한 숫자가 있을 경우

`product_list_get_by_code(products, code)`으로 해당 상품정보를 리스트에서 가져온 뒤 `user_lookup_product(session, selected)`를 통해 자세한 상품 정보를 로드한다.

사용자가 이전 화면으로 돌아간다는 명령을 할 때까지 반복해서 숫자를 입력받아 위와 같은 동작을 수행한다.

void user_lookup_product(Member *session, Product product): 현재 로그인한 사용자가 개별 상품을 조회하는 페이지이다. 상품 정보를 출력하고 1 또는 2의 숫자를 입력받아 1의 경우 상품 구매 페이지로 이동하고, 2의 경우 이전 화면으로 돌아간다. 사용자가 이전 화면으로 돌아간다는 명령을 할 때까지 반복해서 숫자를 입력받아 위와 같은 동작을 수행한다.

void user_purchase_product(Member *session, Product product): 현재 로그인한 사용자가 상품을 구매하는 페이지이다.

가상계좌 잔액이 상품 가격보다 크거나 같을 경우, 구매 후의 가상계좌 잔액을 출력하고 1과 2 중 하나의 값을 입력받는다. 1을 입력받았을 경우 상품을 구매한다. 2를 입력받았을 경우 이전 화면으로 돌아간다.

가상계좌 잔액이 상품 가격보다 작을 경우, 부족한 금액 출력하고 1과 2 중 하나의 값을 입력받는다. 1을 입력받았을 경우 가상계좌 입출금 페이지로 이동한다. 2를 입력받았을 경우 이전 화면으로 돌아간다. 사용자가 이전 화면으로 돌아간다는 명령을 할 때까지 반복해서 숫자를 입력받아 위와 같은 동작을 수행하므로 가상계좌 잔액을 충전한 후에 상품을 구매할 수 있다.

상품을 구매한 경우, 현재 세션의 가상계좌 잔액에서 상품 가격만큼의 금액을 차감하고 `customer.csv`에 저장한다. 또한 전체 주문 목록의 맨 뒤에 새 주문을 추가하고 `history.csv`에 저장한다.

void user_search_orders(Member *session): 현재 로그인한 사용자의 주문을 조회하는 페이지이다. `order_query_by_member_id(orders, session->_idx)`로 현재 세션의 모든 주문을 새로운 연결 리스트로 가져오고, 그 리스트의 `head`부터 순회하면서 모든 주문을 출력한다. 어떤 상품을 주문했는지 쉽게 알아볼 수 있게 `product_list_get_by_code(products, x->order.product_idx)`를 이용해 상품 이름도 가져온다.

void user_search_parcels(Member *session): 현재 로그인한 사용자의 주문들의 배송 상태를 조회하는 페이지이다. `order_query_by_member_id(orders, session->_idx)`로 현재 세션의 모든 주문을 새로운 연결 리스트로 가져오고, 그 리스트의 `head`부터 순회하면서 모든 주문의 배송 상태를 출력한다.

`flow/admin/admin_login_page.c` `flow/admin/admin_login_page.c`는 관리자 로그인 페이지이다.

void admin_login_page(): 관리자 로그인 페이지이다. 아이디와 비밀번호를 입력받고, 아이디가 admin, 비밀번호가 password인지 확인한다. 일치하는 경우 관리자 페이지를 실행하고, 일치하지 않는 경우 3번까지 재시도하며 3번째에도 일치하지 않을 경우에는 메인 메뉴로 돌아간다.

flow/admin/admin_page.c flow/admin/admin_page.c는 관리자 기능들을 제공하는 메서드들의 집합이다.

void admin_page(): 관리자 페이지이다. 1에서 7 중 하나의 숫자를 입력받아 다음과 같은 동작을 실행한다.

입력값	동작
1	회원 정보 조회 페이지 - admin_lookup_customers()
2	상품 등록 및 삭제 페이지 - admin_add_or_delete_products()
3	상품 수정 페이지 - admin_edit_products()
4	상품 조회 및 통계 페이지 - admin_search_product()
5	구매 내역 조회 및 통계 페이지 - admin_lookup_orders()
6	배송 내역 수정 페이지 - admin_lookup_parcel_status()
7	프로그램 종료

void admin_lookup_customers(): 관리자가 전체 회원 정보를 조회하는 페이지이다. member_get_all()으로 전체 회원 정보를 가져와 member_print_all(members, lang)으로 출력한다.

void admin_add_or_delete_products(): 관리자가 상품을 새로 등록하거나 기존 상품을 삭제하는 페이지이다. 1에서 3 중 하나의 숫자를 입력받아 다음과 같은 동작을 실행한다.

입력값	동작
1	상품 등록 - admin_add_product()
2	상품 삭제 - admin_remove_product()
3	이전 화면으로 돌아가기

사용자가 이전 화면으로 돌아간다는 명령을 할 때까지 반복해서 숫자를 입력받아 위와 같은 동작을 수행한다.

void admin_add_product(): 관리자가 상품을 새로 등록하는 페이지이다. 상품 이름, 카테고리, 1 과 2×10^9 사이의 가격을 입력받아 새 상품 product를 만든다. 유효한 1 혹은 0 값을 scan_valid_

`boolean()`으로 입력받아 즉시 판매할지도 선택할 수 있다. 새로 만든 상품은 전체 상품 리스트에 추가되고 `product.csv`에 추가된다. 이후에 유효한 1 혹은 0 값을 입력받아 상품을 계속 추가할지, 혹은 그만 추가할지 결정한다.

void admin_remove_product(): 관리자가 상품을 삭제하는 페이지이다. 상품 전체 리스트를 출력하고, 수를 입력받는다. 입력받은 수가 0일 경우 이전 화면으로 돌아가고, 목록에 존재하는 상품코드일 경우 해당 상품을 찾아 `state`를 `Unavailable`로 바꾸고 `product.csv`에 반영한다. 이후에 유효한 1 혹은 0 값을 입력받아 상품을 계속 삭제할지, 혹은 그만 삭제할지 결정한다.

void admin_edit_products(): 관리자가 상품 편집을 위해 전체 상품을 조회하는 페이지이다. 상품 전체 리스트를 출력하고, 수를 입력받는다. 입력받은 수가 0일 경우 이전 화면으로 돌아가고, 목록에 존재하는 상품코드일 경우 `admin_edit_product(selected)`을 호출해 개별 상품 편집 페이지로 들어간다.

void admin_edit_product(Product product): 관리자가 단일 상품을 편집하는 페이지이다. 1에서 5 중 하나의 숫자를 입력받아 다음과 같은 동작을 실행한다.

입력값	동작
1	이름 수정
2	카테고리 수정
3	가격 수정
4	판매 상태 수정
5	이전 화면으로 돌아가기

모든 문자열은 `util/util_io.c`에 존재하는 유효한 문자열 입력 함수로 입력받는다. 새 정보를 입력받으면 현재 수정하고 있는 항목에 반영하며 `product.csv`에도 새 정보를 덮어씌워 저장한다.

void admin_search_product(): 관리자가 상품을 검색하는 페이지이다. 1에서 5 중 하나의 숫자를 입력받아 다음과 같은 동작을 실행한다.

입력값	동작
1	이름으로 상품 검색 - <code>product_query_by_name()</code>
2	카테고리로 상품 검색 - <code>product_query_by_category()</code>
3	가격대로 상품 검색 - <code>product_query_by_price()</code>
4	전체 상품 보기 - <code>product_query_available()</code>
5	이전 화면으로 돌아가기

가격 범위의 최소값과 최대값을 각각 min, max 라고 하면, $0 \leq min \leq max \leq 2 \times 10^9$ 가 되도록 입력받는다. 검색된 상품은 queried에 새 리스트로 메모리에 저장되고, admin_product_list(queried)으로 넘겨주게 된다.

관리자가 이전 화면으로 돌아간다는 명령을 할 때까지 반복해서 숫자를 입력받아 위와 같은 동작을 수행한다.

void admin_product_list(ProductList *products): 관리자가 검색한 상품들의 목록을 조회하는 페이지이다. 다른 함수가 넘겨준 products를 우선 product_admin_print_all(products, lang)로 출력하고, 숫자를 하나 입력받는다. 입력받은 숫자가 0일 경우 이전 화면으로 돌아가고, 출력한 상품 코드 중에 입력한 숫자가 있을 경우 product_list_get_by_code(products, code)으로 해당 상품정보를 리스트에서 가져온 뒤 admin_lookup_product(selected)를 통해 자세한 상품 정보를 로드한다.

void admin_lookup_product(Product product): 관리자가 단일 상품을 조회하는 페이지이다. 상품코드와 이름, 카테고리, 판매 상태, 가격, 판매량을 출력한다. 0을 입력해 이전 화면으로 돌아갈 수 있다.

void admin_lookup_orders(): 관리자가 구매 내역을 검색하는 페이지이다. 1에서 5 중 하나의 숫자를 입력받아 다음과 같은 동작을 실행한다.

입력값	동작
1	사용자 ID로 주문 검색 - order_query_by_member_id()
2	상품코드로 주문 검색 - order_query_by_product_id()
3	미배송 주문 보기 - order_query_undelivered()
4	전체 주문 보기 - order_get_all()
5	이전 화면으로 돌아가기

검색된 주문들은 queried에 새 리스트로 메모리에 저장되고, admin_order_list(queried)으로 넘겨주게 된다.

void admin_order_list(OrderList* orders): 관리자가 검색한 주문 내역들의 목록을 조회하는 페이지이다. order_print_all(orders, lang)과 동치이다.

void admin_lookup_parcel_status(): 관리자가 배송 내역을 조회하기 위해 구매 내역을 검색하는 페이지이다. 1에서 5 중 하나의 숫자를 입력받아 다음과 같은 동작을 실행한다.

입력값	동작
1	사용자 ID로 주문 검색 - <code>order_query_by_member_id()</code>
2	상품코드로 주문 검색 - <code>order_query_by_product_id()</code>
3	미배송 주문 보기 - <code>order_query_undelivered()</code>
4	전체 주문 보기 - <code>order_get_all()</code>
5	이전 화면으로 돌아가기

검색된 주문들은 queried에 새 리스트로 메모리에 저장되고, `admin_order_list_select_to_edit(queried)`으로 넘겨주게 된다. 관리자가 이전 화면으로 돌아간다는 명령을 할 때까지 반복해서 숫자를 입력받아 위와 같은 동작을 수행한다.

`void admin_order_list_select_to_edit(OrderList* orders)`: 관리자가 배송 상태를 편집하기 위해 검색된 구매 내역들을 조회하는 페이지이다. 쿼리된 주문 리스트를 출력하고, 수를 입력받는다. 입력받은 수가 0일 경우 이전 화면으로 돌아가고, 목록에 존재하는 주문번호일 경우 `admin_edit_parcel_status(selected)`을 호출해 개별 주문 편집 페이지로 들어간다.

`void admin_edit_parcel_status(OrderHistory order)`: 관리자가 단일 구매 내역의 배송 상태를 편집하는 페이지이다. 1에서 4 중 하나의 숫자를 입력받아 다음과 같은 동작을 실행한다.

입력값	동작
1	현재 주문의 배송 상태를 '배송 준비 중' (DeliverReady) 으로 설정
2	현재 주문의 배송 상태를 '배송 중' (Delivering) 으로 설정
3	현재 주문의 배송 상태를 '배송 완료' (Delivered) 으로 설정
4	이전 화면으로 돌아가기

새 주문 상태를 입력받으면 현재 수정하고 있는 항목에 반영하며 `history.csv`에도 새 정보를 덮어씌워 저장한다.

2.4 시험

별도의 자동화된 Unit testing은 진행하지 않았으며, 모든 시험은 수동으로 진행되었다.

신규 사용자의 상품 구매 : 회원 가입 – 상품 검색 – 상품 선택 – 가상계좌 입금 – 상품 구매.

모든 메뉴에서 메뉴에 표시되지 않은 수를 입력할 경우 오류 메시지를 출력하고 재입력을 요구해야 한다.

회원 가입 시 아이디는 4~16자의 영어 대소문자, 숫자, 언더스코어(_)의 조합으로, 이외 정보들은 따옴표(“)와 쉼표(,)가 들어가지 않은 문자열로 입력받는다. 조건에 맞지 않는 문자열을 입력했을 경우 재입력을 요구한다. 비밀번호와 비밀번호 확인이 일치하지 않을 경우, 아이디가 이미 존재할 경우에도 재입력을 요구한다.

상품 선택 시 검색된 상품 리스트에 없는 상품코드를 선택할 경우 재입력을 요구한다.

가상계좌 잔액보다 상품 가격이 더 비쌀 경우 가상계좌 입금 메뉴가, 아닐 경우 상품 구매 메뉴를 출력한다. 가상계좌 입금 시 $(2 \times 10^9 - (\text{현재 잔액}))$ 원을 초과한 금액을 입금 시도할 경우 오류 메시지를 출력하고 재입력을 요구한다. 상품 구매 이후 가상계좌 잔액이 상품 가격만큼 차감된 것을 확인한다.

기존 사용자의 구매/배송 내역 조회 : 로그인 – 구매 내역 조회 – 배송 내역 조회.

구매 내역과 배송 내역이 올바르게 출력되는지 확인한다.

기존 사용자의 회원 정보 변경 : 로그인 – 회원 정보 수정 – 비밀번호 수정 – 이전 화면 – 회원 정보 수정

기존 비밀번호와 길이가 다른 새로운 비밀번호를 입력하고 이전 화면으로 돌아갔다가 다시 회원 정보 수정 페이지로 돌아왔을 때 새 비밀번호가 잘 반영되어 있는지 확인한다. 기존 비밀번호가 틀렸을 경우에는 재입력을 요구한다.

기존 사용자의 탈퇴 : 로그인 – 회원 정보 수정 – 회원 탈퇴.

로그인을 3번 실패할 경우 메인 메뉴로 돌아와야 한다. 회원 정보 수정에서 회원 탈퇴를 선택할 경우 정말로 탈퇴할 것인지 한 번 더 확인해야 하며 탈퇴 이후에는 그 회원의 아이디로 로그인할 수 없어야 한다.

관리자의 상품 등록 및 삭제 : 관리자 로그인 – 상품 등록 및 삭제 – 상품 등록 – 이전 화면 – 상품 조회 및 통계 보기 – 전체 상품 보기 – 이전 화면 – 상품 등록 및 삭제 – 상품 삭제 – 이전 화면 – 상품 조회 및 통계 보기 – 전체 상품 보기.

관리자가 아이디 admin, 비밀번호 password으로 로그인할 수 있는지 확인하고, 상품 정보가 잘 등록 되는지 확인한다. 음수의 가격 혹은 2×10^9 를 초과하는 가격이 입력될 경우에는 재입력을 요구한다. 전체 상품 조회를 했을 때 해당 상품이 조회되어야 한다.

상품이 삭제된 후 전체 상품 조회를 했을 때 해당 상품이 조회되지 않아야 한다.

관리자의 배송 상태 수정 : 관리자 로그인 – 배송 내역 수정.

관리자가 사용자가 구매한 상품의 배송 내역을 수정한다. 배송 내역을 수정할 때 1~4의 숫자만 입력 받아야 하며 다른 숫자를 입력받을 경우 경고문을 표시하고 재입력을 요구해야 한다.

2.5 평가

신규 사용자의 상품 구매 : 회원가입 – 상품 검색 – 상품 선택 – 가상계좌 입금 – 상품 구매.

===== SG-MALL에 오신 것을 환영합니다. =====
1. 회원 로그인
2. 회원 가입
3. 상품 검색
4. 관리자 로그인
5. 종료
선택: 2

===== 회원 가입 =====
1. 이름: 사용자
2. 아이디: shiftph
3. 이미 사용 중인 아이디입니다.
4. 아이디: customer
5. 비밀번호: password
6. 비밀번호 확인: passwords
7. 비밀번호가 일치하지 않습니다.
8. 비밀번호 확인: password
9. 주소: 서울시 양천구 신정동
10.

===== 사용자님, SG-MALL에 오신 것을 환영합니다. =====
1. 회원 정보 수정
2. 가상계좌 입금 및 출금
3. 상품 검색
4. 구매 내역 조회
5. 배송 내역 조회
6. 로그아웃 및 종료
선택: 3

===== 상품 검색 =====
1. 이름으로 검색하기

- 3 2. 카테고리로 검색하기
- 4 3. 가격으로 검색하기
- 5 4. 전체 상품 보기
- 6 5. 이전 화면으로 돌아가기

7 선택: 4

1 상품코드: 1

2 상품명: 티모

3 카테고리: 식용품

4 가격: 13500원

5 -----
6 상품코드: 4

7 상품명: 베인

8 카테고리: 생활용품

9 가격: 56000원

10 -----
11 0. 이전 화면으로 돌아가기

12 선택: 4

© 2018 Suhyun Park
All Rights Reserved.

1 상품코드: 4

2 상품명: 베인

3 카테고리: 생활용품

4 가격: 56000원

5 -----
6 1. 구매하기

7 2. 이전 화면으로 돌아가기

8 선택: 1

1 내 가상계좌 잔액: 0원

2 상품 가격: 56000원

3 필요 금액: 56000원

4 -----
5 1. 가상계좌 입금하고 구매하기

6 2. 이전 화면으로 돌아가기

7 선택: 1

1 ===== 가상계좌 입금 및 출금 =====

2 내 가상계좌 잔액: 0원

3 -----
4 1. 입금하기
5 2. 출금하기
6 3. 이전 화면으로 돌아가기
7 선택: 5
8 입력 범위를 벗어났습니다.
9 선택: 1

1 입금 금액: 9999999999
2 입력 범위를 벗어났습니다.
3 입금 금액: 320000
4 입금이 완료되었습니다.

1 ----- 가상계좌 입금 및 출금 -----
2 내 가상계좌 잔액: 320000원
3 -----
4 1. 입금하기
5 2. 출금하기
6 3. 이전 화면으로 돌아가기
7 선택: 3

1 내 가상계좌 잔액: 320000원
2 상품 가격: 56000원
3 구매 후 잔액: 264000원
4 -----
5 1. 구매하기
6 2. 이전 화면으로 돌아가기
7 선택: 1
8 구매가 완료되었습니다.

신규 사용자의 상품 구매는 문제 없이 이루어졌다. 잘못된 값의 입력을 방지하기 위한 정책도 잘 지켜졌다. 연결 리스트로 구현되어 있어 새로운 주문을 추가하는 것은 상수 시간($O(1)$)에 해결할 수 있었으나, 안타깝게도 파일로 저장할 때 이를 전부 덮어씌우는 것은 선형 시간($O(|list|)$)이 걸렸다. `fopen("customer.csv", "a")`를 사용해 파일 맨 뒤에 한 줄을 추가하는 메서드를 따로 짰다면 더 효율적이었을 것이다.

기존 사용자의 구매/배송 내역 조회 : 로그인 – 구매 내역 조회 – 배송 내역 조회.

===== SG-MALL에 오신 것을 환영합니다. =====

1. 회원 로그인

2. 회원가입

3. 상품 검색

4. 관리자 로그인

5. 종료

선택: 1

===== 회원 로그인 =====

아이디: customer

비밀번호: password

회원 아이디 또는 비밀번호가 일치하지 않습니다.

아이디: customer

비밀번호: password

===== 사용자님, SG-MALL에 오신 것을 환영합니다. =====

1. 회원 정보 수정

2. 가상계좌 입금 및 출금

3. 상품 검색

4. 구매 내역 조회

5. 배송 내역 조회

6. 로그아웃 및 종료

선택: 4

© 2018 Suhyun Park
All Rights Reserved.

===== 구매 내역 =====

주문번호: 5

상품코드: 4

상품명: 베인

카테고리: 생활용품

가격: 56000원

===== 사용자님, SG-MALL에 오신 것을 환영합니다. =====

1. 회원 정보 수정

2. 가상계좌 입금 및 출금

3. 상품 검색

4. 구매 내역 조회

- 6 5. 배송 내역 조회
7 6. 로그아웃 및 종료
8 선택: 5

1 ===== 배송 내역 =====
2 주문번호: 5
3 상품코드: 4
4 상품명: 베인
5 배송 상태: 배송 준비
6 -----

구매 내역과 배송 내역이 올바르게 출력되었다.

기존 사용자의 회원 정보 변경 :로그인 – 회원 정보 수정 – 비밀번호 수정 – 이전 화면 – 회원 정보 수정

- 1 ===== SG-MALL에 오신 것을 환영합니다. =====
2 1. 회원 로그인
3 2. 회원 가입
4 3. 상품 검색
5 4. 관리자 로그인
6 5. 종료
7 선택: 1

1 ===== 회원 로그인 =====
2 아이디: customer
3 비밀번호: password

- 1 ===== 사용자님, SG-MALL에 오신 것을 환영합니다. =====
2 1. 회원 정보 수정
3 2. 가상계좌 입금 및 출금
4 3. 상품 검색
5 4. 구매 내역 조회
6 5. 배송 내역 조회
7 6. 로그아웃 및 종료
8 선택: 1

===== 회원 정보 수정 =====

1. 이름: 사용자
 2. 비밀번호: *****
 3. 주소: 서울시 양천구 신정동
 4. 회원 탈퇴
 5. 이전 화면으로 돌아가기
- 선택: 2
- 현재 비밀번호: **passwor**
비밀번호가 일치하지 않습니다.
- 현재 비밀번호: **password**
새 비밀번호: **newpassword**,
잘못된 입력입니다.
- 새 비밀번호: **newpassword**
새 비밀번호 확인: **newpassword**
새 비밀번호가 저장되었습니다.

===== 회원 정보 수정 =====

1. 이름: 사용자
 2. 비밀번호: *****
 3. 주소: 서울시 양천구 신정동
 4. 회원 탈퇴
 5. 이전 화면으로 돌아가기
- 선택: 5

© 2018 Suhyun Park
All Rights Reserved.

===== 사용자님, SG-MALL에 오신 것을 환영합니다. =====

1. 회원 정보 수정
 2. 가상계좌 입금 및 출금
 3. 상품 검색
 4. 구매 내역 조회
 5. 배송 내역 조회
 6. 로그아웃 및 종료
- 선택: 1

===== 회원 정보 수정 =====

1. 이름: 사용자
2. 비밀번호: *****
3. 주소: 서울시 양천구 신정동
4. 회원 탈퇴

6 5. 이전 화면으로 돌아가기
7 선택:

비밀번호 변경 화면에서 기존 비밀번호와 일치하지 않는 비밀번호를 입력했을 경우 경고 메시지가 나왔다. 새 비밀번호에 허용되지 않은 문자인 쉼표를 입력했더니 경고 메시지가 나왔다. 비밀번호 변경에 성공한 후 이전 화면으로 나갔다가 다시 들어와도 변경된 비밀번호가 제대로 유지되었다. 따라서 세션 형태로 현재 로그인된 유저를 관리하는 것은 안정적이라고 평가할 수 있을 것이다.

기존 사용자의 탈퇴 :로그인 – 회원 정보 수정 – 회원 탈퇴.

1 ===== SG-MALL에 오신 것을 환영합니다. =====
2 1. 회원 로그인
3 2. 회원 가입
4 3. 상품 검색
5 4. 관리자 로그인
6 5. 종료
7 선택: 1

1 ===== 회원 로그인 =====
2 아이디: **asdf**
3 비밀번호: **asdf**
4 회원 아이디 또는 비밀번호가 일치하지 않습니다.
5 아이디: **asdf**
6 비밀번호: **asdf**
7 회원 아이디 또는 비밀번호가 일치하지 않습니다.
8 아이디: **asdf**
9 비밀번호: **asdf**
10 회원 아이디 또는 비밀번호가 일치하지 않습니다.
11 확인 후 다시 로그인을 시도하여 주세요.

1 ===== SG-MALL에 오신 것을 환영합니다. =====
2 1. 회원 로그인
3 2. 회원 가입
4 3. 상품 검색
5 4. 관리자 로그인
6 5. 종료
7 선택: 1

```
1 ===== 회원 로그인 =====  
2 아이디: customer  
3 비밀번호: newpassword
```

```
1 ===== 사용자님, SG-MALL에 오신 것을 환영합니다. =====  
2 1. 회원 정보 수정  
3 2. 가상계좌 입금 및 출금  
4 3. 상품 검색  
5 4. 구매 내역 조회  
6 5. 배송 내역 조회  
7 6. 로그아웃 및 종료  
8 선택: 1
```

```
1 ===== 회원 정보 수정 =====  
2 1. 이름: 사용자  
3 2. 비밀번호: *****  
4 3. 주소: 서울시 양천구 신정동  
5 4. 회원 탈퇴  
6 5. 이전 화면으로 돌아가기  
7 선택: 4  
8 정말 탈퇴하시겠습니까? (Y/n): y  
9 탈퇴가 완료되었습니다.
```

(프로그램 재실행)

```
1 ===== SG-MALL에 오신 것을 환영합니다. =====  
2 1. 회원 로그인  
3 2. 회원 가입  
4 3. 상품 검색  
5 4. 관리자 로그인  
6 5. 종료  
7 선택: 1
```

```
1 ===== 회원 로그인 =====  
2 아이디: customer  
3 비밀번호: newpassword
```

4 회원 아이디 또는 비밀번호가 일치하지 않습니다.
5 아이디:

로그인을 3번 실패할 경우 메인 메뉴로 돌아왔다. 회원 탈퇴를 선택할 경우 적절한 확인 절차를 거쳤다. 탈퇴 이후에는 그 회원의 아이디로 로그인할 수 없었다.

관리자의 상품 등록 및 삭제 : 관리자 로그인 – 상품 등록 및 삭제 – 상품 등록 – 이전 화면 – 상품 조회 및 통계 보기 – 전체 상품 보기 – 이전 화면 – 상품 등록 및 삭제 – 상품 삭제 – 이전 화면 – 상품 조회 및 통계 보기 – 전체 상품 보기.

1 ===== SG-MALL에 오신 것을 환영합니다. =====
2 1. 회원 로그인
3 2. 회원 가입
4 3. 상품 검색
5 4. 관리자 로그인
6 5. 종료
7 선택: 4

1 ===== 관리자 로그인 =====
2 아이디: admin
3 비밀번호: password
All Rights Reserved.

1 ===== SG-MALL 관리자 페이지 =====
2 1. 회원 정보 조회
3 2. 상품 등록 및 삭제
4 3. 상품 정보 수정
5 4. 상품 조회 및 통계 보기
6 5. 구매 내역 및 통계 보기
7 6. 배송 내역 수정
8 7. 로그아웃 및 종료
9 선택: 2

1 ===== 상품 등록 및 삭제 =====
2 1. 상품 추가
3 2. 상품 삭제
4 3. 이전 화면으로 돌아가기
5 선택: 1

```
1 ===== 상품 등록 =====
2 상품명: 새 상품
3 카테고리: 테스트
4 가격: -300
5 입력 범위를 벗어났습니다.
6 가격: 2000000001
7 입력 범위를 벗어났습니다.
8 가격: 314159
9 상품이 등록되어 현재 판매 준비 중입니다.
10 상품을 즉시 판매 상태로 설정할까요? (Y/n): Y
11 다른 상품을 계속 등록할까요? (Y/n): n
```

```
1 ===== 상품 등록 및 삭제 =====
2 1. 상품 추가
3 2. 상품 삭제
4 3. 이전 화면으로 돌아가기
5 선택: 3
```

```
1 ===== SG-MALL 관리자 페이지 =====
2 1. 회원 정보 조회
3 2. 상품 등록 및 삭제
4 3. 상품 정보 수정
5 4. 상품 조회 및 통계 보기
6 5. 구매 내역 및 통계 보기
7 6. 배송 내역 수정
8 7. 로그아웃 및 종료
9 선택: 4
```

```
1 ===== 상품 검색 =====
2 1. 이름으로 검색하기
3 2. 카테고리로 검색하기
4 3. 가격으로 검색하기
5 4. 전체 상품 보기
6 5. 이전 화면으로 돌아가기
7 선택: 4
```

```
1 상품코드: 1
2 상품명: 새 상품
```

3 카테고리: 테스트

4 가격: 314159원

5 상품 상태: 판매 중

6 -----

7 0. 이전 화면으로 돌아가기

8 선택: 0

1 ===== 상품 검색 =====

2 1. 이름으로 검색하기

3 2. 카테고리로 검색하기

4 3. 가격으로 검색하기

5 4. 전체 상품 보기

6 5. 이전 화면으로 돌아가기

7 선택: 5

1 ===== SG-MALL 관리자 페이지 =====

2 1. 회원 정보 조회

3 2. 상품 등록 및 삭제

4 3. 상품 정보 수정

5 4. 상품 조회 및 통계 보기

6 5. 구매 내역 및 통계 보기

7 6. 배송 내역 수정

8 7. 로그아웃 및 종료

9 선택: 2

1 ===== 상품 등록 및 삭제 =====

2 1. 상품 추가

3 2. 상품 삭제

4 3. 이전 화면으로 돌아가기

5 선택: 2

1 ===== 상품 삭제 =====

2 상품코드: 1

3 상품명: 새 상품

4 카테고리: 테스트

5 가격: 314159원

6 상품 상태: 판매 중

7 -----

8 0. 이전 화면으로 돌아가기

9 선택: 1

10 상품이 삭제되었습니다.

11 다른 상품을 계속 삭제할까요? (Y/n): n

1 ===== 상품 등록 및 삭제 =====

2 1. 상품 추가

3 2. 상품 삭제

4 3. 이전 화면으로 돌아가기

5 선택: 3

1 ===== SG-MALL 관리자 페이지 =====

2 1. 회원 정보 조회

3 2. 상품 등록 및 삭제

4 3. 상품 정보 수정

5 4. 상품 조회 및 통계 보기

6 5. 구매 내역 및 통계 보기

7 6. 배송 내역 수정

8 7. 로그아웃 및 종료

9 선택: 4

© 2018 Suhyun Park

All Rights Reserved.

1 ===== 상품 검색 =====

2 1. 이름으로 검색하기

3 2. 카테고리로 검색하기

4 3. 가격으로 검색하기

5 4. 전체 상품 보기

6 5. 이전 화면으로 돌아가기

7 선택: 4

1 일치하는 결과가 없습니다.

2 0. 이전 화면으로 돌아가기

3 선택:

관리자는 아이디 admin, 비밀번호 password으로 로그인할 수 있었고, 상품 정보는 잘 등록되었다. 음수의 가격 혹은 2×10^9 를 초과하는 가격이 입력될 경우 재입력이 요구되었다. 전체 상품 조회를 했을 때 해당 상품이 조회되었고, 삭제했을 경우 조회되지 않았다.

관리자의 배송 상태 수정 : 관리자 로그인 – 배송 내역 수정.

===== SG-MALL에 오신 것을 환영합니다. =====
1. 회원 로그인
2. 회원 가입
3. 상품 검색
4. 관리자 로그인
5. 종료
6. 선택: 4
7.

===== 관리자 로그인 =====
1. 아이디: admin
2. 비밀번호: password
3.

===== SG-MALL 관리자 페이지 =====
1. 회원 정보 조회
2. 상품 등록 및 삭제
3. 상품 정보 수정
4. 상품 조회 및 통계 보기
5. 구매 내역 및 통계 보기
6. 배송 내역 수정
7. 로그아웃 및 종료
선택: 6

===== 배송 상태 수정 =====
1. 구매자 ID로 조회
2. 상품 코드로 조회
3. 미배송된 주문 조회
4. 전체 주문 조회
5. 이전 화면으로 돌아가기
선택: 3

1. 주문번호: 3 / 상품번호: 4
2. 주문번호: 5 / 상품번호: 4
3. -----
4. 0. 이전 화면으로 돌아가기
5. 선택: 5

1 주문번호: 5
2 상품코드: 4
3 회원번호: 8
4 아이디: shiftptsh
5 이름: 박수현
6 주소: 서울시 양천구 신정동
7 배송 상태: 배송 준비

8 새 배송 상태:
9 1. 배송 준비
10 2. 배송 중
11 3. 배송 완료
12 4. 이전 화면으로 돌아가기
13 선택: 5
14 입력 범위를 벗어났습니다.
15 선택: 3
16 배송 상태가 수정되었습니다.
17

===== 배송 상태 수정 =====
1 1. 구매자 ID로 조회
2 2. 상품 코드로 조회
3 3. 미배송된 주문 조회
4 4. 전체 주문 조회
5 5. 이전 화면으로 돌아가기
6 선택: 3
7

1 주문번호: 3 / 상품번호: 4
2 -----
3 0. 이전 화면으로 돌아가기
4 선택:

배송 내역을 수정할 때 1~4 사이의 숫자가 아닌 숫자를 입력했더니 경고문이 표시되었다. 배송 내역을 미배송에서 배송 완료로 수정한 후 다시 미배송 주문 리스트를 쿼리할 경우 해당 주문이 나타나지 않았다.

2.6 보건 및 안정

이 프로그램에서 발생할 수 있는 대부분의 오류는 입력 포맷 오류 – 즉 사용자가 잘못된 형식의 문자열을 입력하는 경우이다. 이런 오류는 util_io.c의 입력 함수들을 사용하는 것으로 대부분 해결되었다. 일단 사용자가 너무 많은 글자를 입력할 경우 앞의 1,000 글자만 받도록 한정하였으며, 2.2에 적힌 대로 모든 함수가 구현되어 있어 이 유ти리티 메서드를 다른 메서드들에서 가져다 쓰기만 하면 올바른 포맷의 문자열을 입력받음이 보장되게 함으로서 해결했다.

코드에서 상수가 아닌(mutable) 전역 변수는 최대한 사용하지 않았다. 상수가 아닌 전역 변수의 경우 어떤 위치에서든 값이 변경될 수 있는데, 프로젝트가 커지면서 이 값이 변경되는 기준이 일관적이지 않을 수 있고 그 변경 시점을 점점 알기 힘들어지기 때문이다. 이로 인해 발생할 수 있는 잠재적 오류들을 전역 변수를 사용하지 않음으로서 줄일 수 있었다.

2.7 생산성과 내구성

사용자와 관리자 등이 쇼핑몰과 상호작용하는 기능들은 flow에, 입출력이나 문자열 처리, 파일 입출력을 도와주는 유ти리티 함수들은 util에, 연결 리스트(LinkedList)나 직접 정의한 자료형과 그와 관련된 함수들은 types에 구현해 소스를 체계적이고 효율적으로 관리할 수 있게 하였다.

C는 OOP(Object Oriented Programming) 패러다임과 제네릭 프로그래밍(Generic Programming)이 적용된 언어는 아니다. 하지만 유연한 설계와 유지보수를 위해 OOP 패러다임과 제네릭 프로그래밍을 최대한 흉내내려 노력했다. 가령, LinkedList 구조체는 노드로 void* 형식을 가지며, 특정 위치에 노드를 삽입하거나 수정하거나 삭제하는 등의 기본적인 메서드만을 구현했다. typedef를 통해 정의된 MemberList, ProductList, OrderList는 기본적으로는 LinkedList와 서로 같지만 types/*.c에 각각에 맞는 메서드들을 따로 정의하였고, LinkedList의 메서드들에 타입 변수 – 언어적 한계로 이 프로젝트에서는 enum – 를 같이 넘겨주도록 했다. 아쉽게도 접근 제한자와 비슷한 언어 기능인 inline 키워드는 컴파일러마다 구현이 달라서 사용하지 않았지만, 메서드 이름에 접두사를 붙임으로서 코드 작성 중에 구조체 간의 혼동이 최대한 일어나지 않도록 하였다. 이는 OOP와 제네릭 프로그래밍의 기조를 흉내낸 것으로, LinkedList가 C++ STL의 std::list에 대응한다면 MemberList는 C++ STL의 std::list<Member>에 대응하게 되는 것이다.

소스 코드는 쉬운 유지보수를 위해 최대한 추상화하였다. 예를 들어 연결 리스트는 노드를 기반으로 구현되어야 하지만 사용자와 관리자가 쇼핑몰과 상호작용하는 메서드들에서는 노드를 (직접적으로는) 전혀 만들거나 참조하지 않았다. 대신 types/*.c에 정의된 함수들만 노드를 만들거나 참조하도록 했다. 이는 노드 생성과 삭제 등의 관리를 일관적으로 할 수 있게 하여 잘못된 구현이 발생할 확률을 줄이게 되고, 결과적으로 훨씬 안정적인 프로그램을 구현할 수 있도록 해 준다.

마지막으로 생성되는 CSV 파일을 제외하고 프로그램에 등장하는 모든 문자열은 language.c에서 관리하도록 해 메시지를 수정해야 할 일이 있다면 코드를 전부 읽으면서 메시지를 찾지 않아도 한 파일에서 쉽게 수정할 수 있게 하였다. 또한 지원하는 언어를 코드 내에서 새로운 Language 생성만으로 쉽게 확장할 수 있게 하여 향후 국제화 혹은 다국어화할 수 있는 기반을 마련했다.

2.8 산업 표준

프로젝트 코드는 전부 ISO/IEC 9899:1999 (C99) 표준^[3]을 따른다. 프로젝트 코드를 통해 빌드한 프로그램이 생성하는 파일은 모두 IETF RFC 4180 (CSV) 표준^[4]을 따른다.

C99 표준을 벗어난 언어 기능이나 헤더 파일은 사용하지 않았으므로 C99 표준을 지원하는 컴파일러로 프로젝트 코드를 빌드할 경우 특정 운영체제에 의존하지 않는다.

소스 코드 내의 모든 메서드는 Doxygen으로 파싱할 수 있는 형태로 문서화되어 있다. Doxygen은 코드를 자동으로 문서화하는 사실상의 표준(de facto) 소프트웨어이다^[5].

3 기타

3.1 환경 구성

이 프로젝트는 Ubuntu 16.04.2 LTS에서 gcc 5.4.0과 GNU Make 4.1를 이용하여 컴파일되었다.

gcc 버전 정보(gcc -v)는 다음과 같다.

```
Using built-in specs.
COLLECT_GCC=gcc
COLLECT_LTO_WRAPPER=/usr/lib/gcc/x86_64-linux-gnu/5/lto-wrapper
Target: x86_64-linux-gnu
Configured with: ../src/configure -v --with-pkgversion='Ubuntu 5.4.0-6ubuntu1
~16.04.10' --with-bugurl=file:///usr/share/doc/gcc-5/README.Bugs --enable
-languages=c,ada,c++,java,go,d,fortran,objc,obj-c++ --prefix=/usr --
program-suffix=-5 --enable-shared --enable-linker-build-id --libexecdir=/
usr/lib --without-included-gettext --enable-threads=posix --libdir=/usr/
lib --enable-nls --with-sysroot=/ --enable-clocale=gnu --enable-libstdcxx
-debug --enable-libstdcxx-time=yes --with-default-libstdcxx-abi=new --
enable-gnu-unique-object --disable-vtable-verify --enable-libmpx --enable
-plugin --with-system-zlib --disable-browser-plugin --enable-java-awt=gtk
--enable-gtk-cairo --with-java-home=/usr/lib/jvm/java-1.5.0-gcj-5-amd64/
jre --enable-java-home --with-jvm-root-dir=/usr/lib/jvm/java-1.5.0-gcj-5-
amd64 --with-jvm-jar-dir=/usr/lib/jvm-exports/java-1.5.0-gcj-5-amd64 --
with-arch-directory=amd64 --with-ecj-jar=/usr/share/java/eclipse-ecj.jar
--enable-objc-gc --enable-multiarch --disable-werror --with-arch-32=i686
--with-abi=m64 --with-multilib-list=m32,m64,mx32 --enable-multilib --with
-tune=generic --enable-checking=release --build=x86_64-linux-gnu --host=
x86_64-linux-gnu --target=x86_64-linux-gnu
Thread model: posix
gcc version 5.4.0 20160609 (Ubuntu 5.4.0-6ubuntu1~16.04.10)
```

Make 버전 정보(make -v)는 다음과 같다.

```
GNU Make 4.1
Built for x86_64-pc-linux-gnu
Copyright (C) 1988-2014 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
```

개발 과정에서 운영체제는 macOS 10.14.1 Mojave를 사용하였고, IDE는 JetBrains 사의 CLion 2018.3을 사용하였다. CLion의 버전 정보는 다음과 같다.

```
CLion 2018.3
Build #CL-183.4284.156, built on November 24, 2018
JRE: 1.8.0_152-release-1343-b15 x86_64
JVM: OpenJDK 64-Bit Server VM by JetBrains s.r.o
macOS 10.14.1
```

3.2 참고 사항

특별히 참고할 만한 사항은 없다.

3.3 팀 구성

개인 프로젝트로 진행하였다.

3.4 수행기간

2018. 11. 14 (수) – 2018. 12. 14 (금)

참고문헌

- [1] Cormen, Thomas H.; Leiserson, Charles E.; Rivest, Ronald L.; Stein, Clifford. *Introduction to Algorithms*. MIT Press. (2003) pp. 205–213, 501–505.
- [2] *How to clear input buffer in C?*, Stack Overflow, <https://stackoverflow.com/questions/7898215/how-to-clear-input-buffer-in-c> (2018년 12월 18일에 조회.)
- [3] ISO/IEC 9899:1999: Programming languages – C, International Organization for Standardization, Geneva, Switzerland. (1999) <http://www.open-std.org/jtc1/sc22/WG14/www/docs/n1256.pdf> (2018년 12월 12일에 조회.)
- [4] IETF RFC 4180: CSV – Comma Separated Values, The Internet Society. (2005) <https://tools.ietf.org/html/rfc4180> (2018년 12월 12일에 조회.)
- [5] Dimitri van Heesch, *Doxygen*. <https://github.com/doxygen/doxygen/blob/master/README.md> (2018년 12월 13일에 조회.)

© 2018 Suhyun Park
All Rights Reserved.