2조 완료 설계서

주차관리해조

고석환, 송대근 이광진

주제 : 주차관리 시스템

백화점이나 마트 등 흔히 마주칠 수 있는 주차장 관리 시스템을 손으로 직접 구현해보고, 더 나은 알고리즘을 생각하여 좀 더 향상할 수 있는 방안을 모색해봤다.

폴더 구조

▼ mobis_pjs_team2

▼ dat

- car_set_list.csv
- car_set.csv
- car_set.dat
- iparking_state.dat
- oparking_state.dat

▼ include

▼ array.h

- int arrayCreate(LPARRAY* lppArray);
- int arraySize(LPC_ARRAY lpArray);
- int arrayGetAt(LPC_ARRAY lpArray, int nPos, LPDATA* lpValue);
- int arraySetAt(LPARRAY lpArray, int nPos, const LPDATA lpValue);
- int arrayAdd(LPARRAY lpArray, const LPDATA lpValue);
- int arrayRemoveAt(LPARRAY lpArray, int nPos);

- int arrayDestroy(LPARRAY lpArray);

▼ car_positioning.h

- int car_positioning(Car_state *car);

▼ display_parking_lot.h

- int clear_buffer(void);
- int init_parking_lot();
- int display_parking_lot(LPARRAY lp_IArray);

▼ enter_exit_manage.h

- typedef struct Person
- typedef struct Location
- typedef struct Time
- typedef struct Car_state
- int enter_exit_time(Car_state *car);
- int car_state_append(LPARRAY lpInput, LPARRAY lpCarset, int car_num);
- int car_state_remove(LPARRAY lpInput, LPARRAY lpOutput, LPARRAY lpCarset, int car_num);
- int disp_car_state(Car_state *lpcar);
- int save(LPARRAY lpiArray, LPARRAY lpoArray);

▼ fee_calculate.h

- int calc_diff_min(Car_state *lpcar, int *ret_diff);
- int fee_calculate(Car_state *lpOcar);
- int get_car_plate_to_put_fee(LPARRAY lp_input_car_Array, char *car_plate_num);

▼ long_term_parking_list.h

- int long_term_parking_list(LPARRAY lpArray, int sel_date, int year, int mon, int date);

▼ parking_history.h

- int parking_history(LPARRAY lp_input_car_Array, LPARRAY lp_output_car_Array);

▼ parking_status.h

- int inputline(FILE* fp, char** str);
- int info_car_num(LPARRAY lpArray,char *car_num, int *flag);*
- *int info_car_loc(LPARRAY lpArray, char* car_loc, int *flag);
- int parking_check(LPARRAY lpArray);

▼ save_load.h

- int car_set_save(Car_state *lpcar, int size);
- int car_set_load(LPARRAY lpArray);
- int input_car_load(LPARRAY lpArray);
- int output_car_load(LPARRAY lpArray);
- int input_car_save(LPARRAY lpArray);
- int output_car_save(LPARRAY lpArray);
- int car_set_load_csv(LPARRAY lpArray);

▼ src

- array.c
- car_positioning.c
- display_parking_lot.c
- enter_exit_manage.c
- fee_calculate.c
- long_term__parking_list.c
- main.c
- parking_history.c
- parking_status.c
- save_load.c

자료구조

Array 활용

먼저, array.c에 선언되어있는 LPAARAY의 구조체를 다음과 같은 변수를 3개 선언한다.

LPARRAY lp_input_car_Array = NULL;
LPARRAY lp_output_car_Array = NULL;
LPARRAY lp_car_set_Array = NULL;

arrayCreate() 로 메모리를 할당한다. 그 후

car_set_load_csv(lp_car_set_Array)
input_car_load(lp_input_car_Array)
output_car_load(lp_output_car_Array)

로 Array에 바이너리파일에 있는 내용을 저장한다.

car_set_load_csv는 car_set.csv의 데이터의 내용을,
input_car_load는 iparking_state.dat의 데이터의 내용을,
output_car_load는 oparking_state.dat의 내용을 각각 저장한다.

위의 과정이 프로그램이 실행되면 자동으로 Array에 저장된다.
입, 출차 내역을 실시간으로 보장하고 데이터의 변경, 추가, 삭제를 용이하도록 위와 같이 구성하였다.

car_set.csv의 내용을 불러온 lp_car_set_Array에 저장된 데이터는 다음과 같다.

```
차량정보:(sonata, red,111가 1111), 주차위치:(B1,2,2), 주차요금:0.100000원 , 인적사항:(1등 , 01011111111)
입차시각:(0.0.0 0h:0m:0s), 출차시각:(0.0.0 0h:0m:0s)
차량정보:( G70,blue,222가 2222), 주차위치:(B1,3,3), 주차요금:0.100000원 , 인적사항:(2등 , 01022222222)
입차시각:(0.0.0 0h:0m:0s), 출차시각:(0.0.0 0h:0m:0s)
```

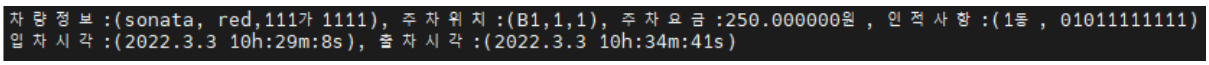
아직 주차장에 들어오지 않은, 일반적인 차량의 데이터다. 시나리오 전개 상 위의 데이터를 활용하여 입차, 출차 시나리오를 구성하고자 한다.

iparking_state.dat의 내용을 불러온 lp_input_car_Array 데이터는 다음과 같다.

```
차량정보:( G70,blue,222가 2222), 주차위치:(B1,1,1), 주차요금:0.100000원 , 인적사항:(2등 , 01022222222)
입차시각:(2022.3.3 10h:30m:47s), 출차시각:(2022.3.3 10h:30m:47s)
```

출차하지 않았으므로, 임의로 입차시각과 출차시각을 같게 했다.
 이 데이터는 현재 주차장에 있는 차량을 나타낸것이다.

oparking_state.dat의 내용을 불러온 lp_output_car_Array 데이터는 다음과 같다.



주차요금이 추가되었다. 그리고 출차 명령을 실행한 시간을 출차시간에 업데이트했다.

위의 데이터는 출차 명령어를 실행하면 lp_input_car_Array 에서의 데이터를 삭제하고 oparking_state.dat에 저장한다. 그 데이터를 불러와 lp_output_car_Array 에 저장했다.

위와 같이, Array 자료구조를 활용했다.

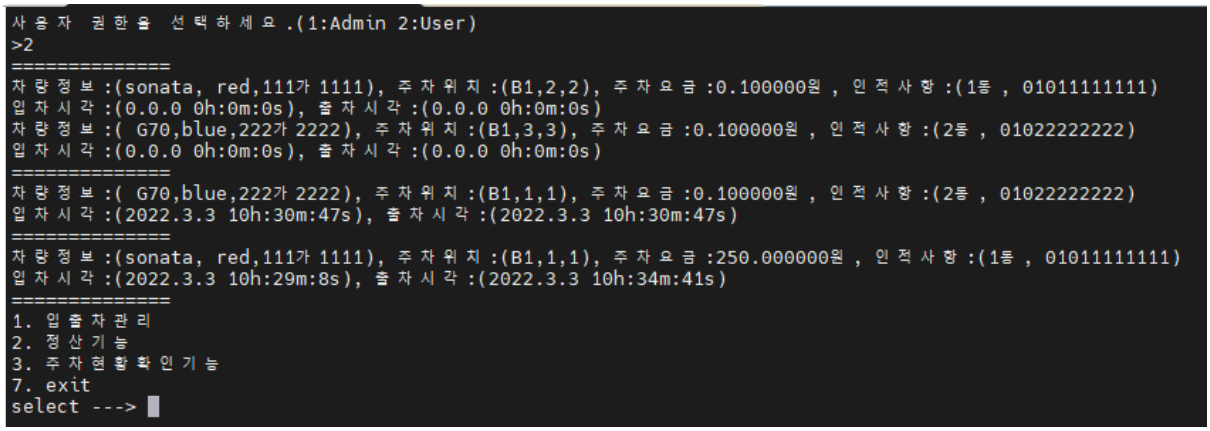
linkedList 활용

Linkedlist로 Queue를 만들어 출구와 가장 가까운 주차장 자리를 안내해주는 알고리즘을 만들었다
출입구로부터 가장 가까운 위치의 자리를 Queue에 넣고 해당 자리가 비어있으면, 그 위치를 알려주는 알고리즘이다.

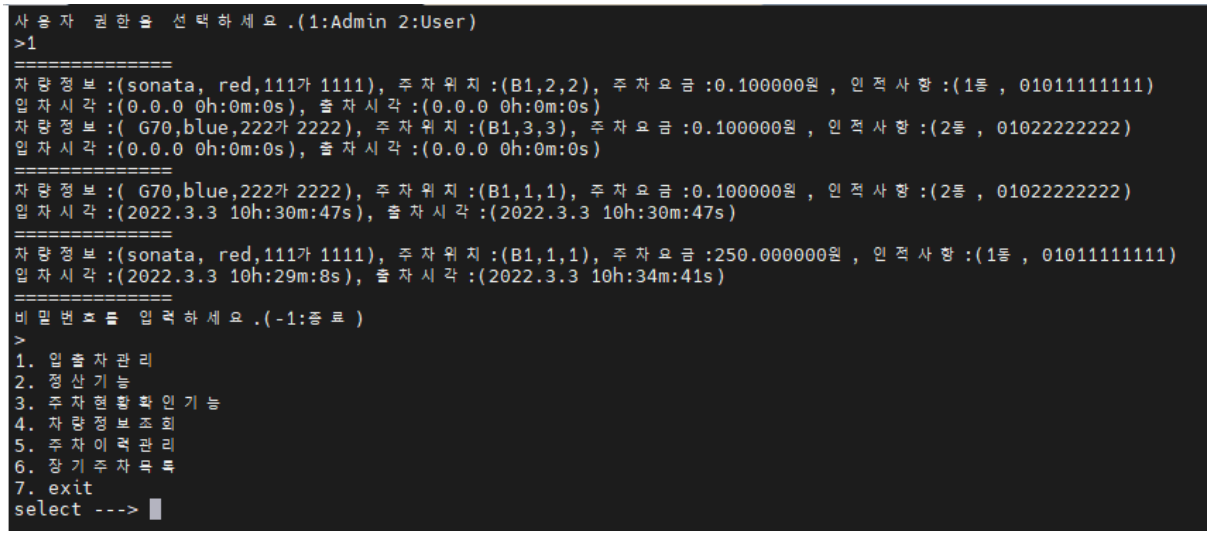
UI 구성

주차장의 모습처럼 UI를 구성했다. 입차 시 해당하는 차 번호가 좌표에 맞게 저장된다.

Admin, User인지 접근 권한을 묻고 접근 권한에 따라 사용할 수 있는 메뉴를 나뉘었다.

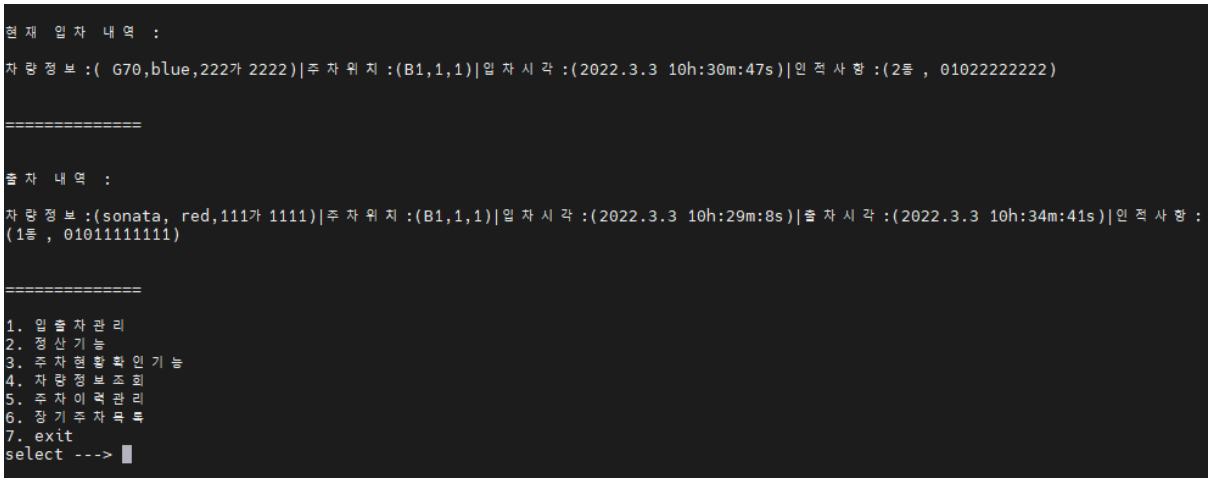


(User로 접근했을 때)



(Admin으로 접근했을 때)

switch문으로 숫자를 입력받아 그에 해당하는 문구를 출력하도록 UI를 구성했다.



Admin의 권한으로 주차이력관리를 실행하니, 현재 입차되어있는 차량 정보와 이미 출차한 모든 차량의 대한 내용을 출력한다.

시나리오

현재 주차장에서 흔히 볼 수 있는 주차장 프로그램은 별도의 센서를 통해 입, 출차가 자동화 되었지만, 본 시나리오에서 사용자가 임의로 입력하여 시나리오를 진행한다고 가정하겠다.

프로그램 실행 시, 초기 메뉴 접근 권한은 크게 2가지로 나뉜다.

1. 관리자 모드(Admin)
2. 일반 유저 모드(User)

초기 프로그램에 진입 시 유저 모드와 관리자 모드 중 선택하라는 선택지가 등장한다. 이때 접근 권한에 따라 보이는 메뉴가 달라진다.

1. 관리자 모드로 실행 시,
    1. 입, 출차 관리
    2. 정산 기능
    3. 주차현황 확인 기능
    4. 차량 정보 조회
    5. 주차 이력 관리
    6. 장기 주차 목록
    7. 빈공간 추천
    8. exit
2. 일반 유저 모드로 실행 시

    입, 출차 관리
    2. 정산기능
    3.주차현황관리 기능
    7. 빈공간 추천
    8. exit

1. 입, 출차 관리(car_state_append, car_state_remove)

enter_exit_manage.c에서 정의.

int car_state_append(LPARRAY lpInput, LPARRAY lpCarset, int car_num);
int car_state_remove(LPARRAY lpInput, LPARRAY lpOutput, LPARRAY lpCarset, int car_num);

---

int car_state_append(LPARRAY lpInput, LPARRAY lpCarset, int car_num);

car_state_append의 매개변수는 Array의 구조체로 프로그램을 실행할 때 Array를 할당 받은 데이터를 집어넣는다.

먼저 lpCarset에 있는 데이터에 대해 car_num으로 접근하여 arrayGetAt이라는 함수로 주소를 가져온다.
그 후 lpinput 구조체와 비교하여 해당 차량이 이미 lpinput 데이터에 존재하는지 중복 여부를 확인한다.
존재하지 않는다면 arrayAdd함수로 lpinput에 해당 데이터를 append한다.

int car_state_remove(LPARRAY lpInput, LPARRAY lpOutput, LPARRAY lpCarset, int car_num);

먼저 lpCarset에 있는 데이터에 대해 car_num으로 접근하여 arrayGetAt이라는 함수로 주소를 가져온다.
input과 마찬가지로, 그 후 lpinput 구조체와 비교하여 해당 차량이 이미 lpinput 데이터에 존재하는지 확인한다.
지우기 전에, tmp_lpcar 구조체에 주소를 가져오고, lpOcar에 깊은 복사를 실행한다. 그 후 lpinput에 있는 해당 데이터를 제거한다.
lpOcar에 있는 구조체에 출차 시간을 갱신하고 lpOutput에 arrayAdd 함수로 추가한다. 이 때, fee_calculator로 입차시간과 출차시간을 비교하여 저장한다.

2. 정산 기능(car_state_append, car_state_remove)

fee_calculate.c에서 정의.

int calc_diff_min(Car_state *lpcar, int *ret_diff);
int fee_calculate(Car_state *lpOcar);
int get_car_plate_to_put_fee(LPARRAY lp_input_car_Array, char *car_plate_num);

---

int calc_diff_min(Car_state *lpcar, int *ret_diff);

현재시각과 lpcar의 입차 시간을 계산하여 ref_diff에 저장한다.

int fee_calculate(Car_state *lpOcar);

해당 차량의 주인이 임직원인지 아닌지 파악 후, 임직원이면 50%할인 금액을 적용한다.

int get_car_plate_to_put_fee(LPARRAY lp_input_car_Array, char *car_plate_num);

만약 입차 목록 중 현재까지의 주차금액을 알고 싶다면,

이 함수로 lp_input_car_Array 내 에 있는 차 중 car_plate_num과 일치한 차에 대해 주차 금액을 알려준다.

3. 주차현황 확인 기능(display_parking_lot)

display_parking_lot.c 에서 정의.

int clear_buffer(void);
int init_parking_lot();
int display_parking_lot(LPARRAY lp_IArray);

---

int clear_buffer(void);

buffer에 쌓인 데이터를 초기화한다.

int init_parking_lot();

주차장을 초기화한다.

int display_parking_lot(LPARRAY lp_IArray);

매개변수로 lp_IArray를 받아온다. Car_state 의 포인터인 tmp에 locaton.floor, location.row, location.cal을 저장하고 해당하는 parking__lot 배열에 tmp를 저장한다.

4. 차량 정보 조회(parking_check)

parking_status.c에서 정의.

int parking_status_check(LPARRAY lpArray, Car_state parking_lot[3][10][10])
int inputline(FILE* fp, char** str);
int info_car_num(LPARRAY lpArray,char *car_num, int *flag)
int info_car_loc(LPARRAY lpArray, char* car_loc, int *flag);
int parking_check(LPARRAY lpArray);

---

int inputline(FILE* fp, char str);**

입력받은 버퍼 끝에 '\n' 처리하는 함수

int info_car_num(LPARRAY lpArray,char *car_num, int *flag)

해당 차량이 주차장 내에 존재하는지 확인하는 함수이다. 자동차 번호를 입력받고, lpArray 내에 번호가 존재하면 출력하는 함수이다.

int info_car_loc(LPARRAY lpArray, char* car_loc, int *flag);

원하는 주차장 구역의 위치에 차량이 존재하는지 확인하는 함수이다. 층, 행, 열 을 입력받고, 해당하는 parking_lot이 비어있지않으면 해당 차량번호를 출력하는 함수이다.

int parking_check(LPARRAY lpArray);

차량 정보를 출력하는 함수. 차량 번호 혹은 위치를 받아 사용자가 원하는 데이터를 출력한다.

5. 주차 이력 관리(parking_history)

parking_history.c 에서 정의

int parking_history(LPARRAY lp_input_car_Array, LPARRAY lp_output_car_Array)

---

int parking_history(LPARRAY lp_input_car_Array, LPARRAY lp_output_car_Array)

lp_input_car_Array, lp_output_car_Array의 두가지를 받아온다.

먼저 lp_input_car_Array의 차를 출력한다. 이는 현재 주차장에 있는 차량을 출력한다는 뜻이다.

그리고 lp_output_car_Array의 차를 출력한다. 이는 모든 기간 동안 출차한 차량의 이력을 출력한다.

6. 장기 주차 목록(long_term_parking_list)

long_term_parking_list.c에서 정의.
int long_term_parking_list(LPARRAY lpArray, int sel_date, int year, int mon, int date)
int time_calc(Car_state *lpcar, Time now, int sel_date, int year, int mon, int date)

먼저 메인문에서 일수로 받을지, 날짜로 받을지를 정한다.

다음과 같은 목록 중 검색하고 싶은 양식을 선택해주세요.
1. 일 수 (예 : 500일 이상으로 장기 주차한 차량에 대해 검색)
2. 날짜 (예 : 2020년 3월 1일 이전에 입차했던 차량에 대해 검색)

일 수로 받을 시 sel_date라는 변수에 원하는 일수를 기입하고, 날짜로 받을 시 year, mon, date라는 각각의 변수에 원하는 년도, 월, 일을 저장한다.

그리고 long_term_parking_list 함수를 호출한다.

int long_term_parking_list(LPARRAY lpArray, int sel_date, int year, int mon, int date)

매개변수는 각각 lpArray, sel_date, year, mon, date이다.

Array의 사이즈만큼 for문을 돌아 입차된 차량의 입차시각과 현재시각을 time_calc함수를 통해 계산한다. 만약 일 수나 지정한 기간보다 길다면, 해당 차량을 출력한다.

int time_calc(Car_state *lpcar, int sel_date, int year, int mon, int date)

lpcar 내의 입차시각과 sel_date 또는 year, mon, date와의 차이를 계산하여 결과값을 리턴해주는 함수이다.

7. 빈공간 추천(empty_space_recom)

empty_space_recom.c 에서 정의.
int empty_space_recom(Car_state parking_lot[3][3][10])
void init_queue(struct Queue *queue)
void enqueue(struct Queue *queue, int r, int c)
void dequeue(struct Queue *queue, int *rPtr, int *cPtr)

---

int empty_space_recom(Car_state parking_lot[3][3][10])

먼저 주차하고 싶은 층을 입력받는다. 그리고 가장 가까운 자리인 [0][5]을 queue에 넣고, 그 위치를 기준으로 삼아 주변 위치를 계속적으로 queue에 넣는다. 만약 주차되어있지 않는 자리를 발견하면 해당 자리를 추천한다.

7. exit(save)

save_load.c에 정의

int save(LPARRAY lpiArray, LPARRAY lpoArray)

프로그램을 종료하는 함수이다. 종료시 *.dat 파일에 바이너리 형태로 저장한다.

int save(LPARRAY lpiArray, LPARRAY lpoArray)

lpiArray와lpoArray 를 각각 put_car_save(lpiArray), output_car_save(lpoArray) 의 함수의 인자로 전달한다. 그리고 iparking_state.dat과 oparking_state.dat에 각각 바이너리 형태로 저장 후 종료한다.