

기말고사 대체 개인 프로젝트 - 중간 #03

<스마트 주차 관리 시스템>

234042 황선하

① 프로젝트 주제

- 차량의 크기와 특성에 따라 주차 공간을 최적화하고, 주차장의 혼잡도를 관리할 수 있는 스마트 주차 시스템을 개발하는 프로젝트입니다. 이를 통해 사용자는 차량 유형에 맞는 자리를 추천받아 더욱 편리하게 주차할 수 있습니다.

② 프로그램 실행 방법

- 처음 출력된 주차장의 상태를 확인한다.
- 주차할 차량의 종류, 혹은 옵션을 입력한다.
- 추천된 자리 목록을 확인하고 주차할 행과 열을 입력한다. (ex. 1 2)
- 주차를 완료한 후 현 상태의 주차장을 확인한다.

③ 현재까지 구현한 바

- (1st_Report)
- 시작 시 map 출력
- 사용자로부터 종류, 옵션 입력받기
- 대형차: 두 자리에 걸쳐, 한 대 주차 가능
- 그 외 차량: 한 칸 당 한 대 주차 가능
- 전기차: 충전 여부를 입력받고 이에 따라 자리 추천
- 추천 자리는 최대 5개까지
- (2nd_Report)
- 오토바이: 한 자리에 두 대까지 주차 가능
- 이미 오토바이가 한 칸에 한 대가 차 있는 경우 나중 오토바이가 입력되었을 때 그 자리가 우선시되어 추천
- 경차자리가 충분할 경우 5개의 추천 목록을 모두 경차자리로

- 이미 채워진 자리는 다른 차량으로 채워지지 않도록
- 자리(특성) 표시와 주차 표시를 구분하여 출력
- (3rd_Report)
- 전용자리에 다른 차량이 주차할 수 없도록
- 주차장의 모든 칸이 포화되었을 때 이를 알리고 프로그램 종료
- 잘못된 입력일 경우 경고문 출력
- 해당 차량의 추천 자리가 없을 경우 안내문 출력
- 추천 자리를 선택하지 않았을 경우 안내문 출력

④ 코드 설명

```
#include <iostream>
#include <string>
#include <vector>
using namespace std;

// 주차 공간의 종류 정의
const int EMPTY = 0;           // 비어있는 공간
const int MOTORBIKE = 1;       // 오토바이 공간
const int COMPACT = 2;         // 경차 공간
const int REGULAR = 3;         // 일반차 공간
const int LARGE = 4;           // 대형차 공간
const int ELECTRIC = 5;        // 전기차 공간
```

- 설명:
 - 각 차량별로 상태를 표시, 숫자를 통해 상태를 표시한다.
 - 이는 변경되지 않는 사항이기 때문에 const로 선언한다.
- 사용된 개념과 주차:
 - 2주차: constant(상수)

```
// 주차 상태 설정
const int OCCUPIED = 1; // 주차된 상태
const int VACANT = 0; // 비어있는 상태
```

- 설명:
 - 주차장의 칸이 어떤 상태인지 1(주차완료), 0(빈 칸)으로 나타낸다.
 - 이는 변경되지 않는 사항이기 때문에 const로 선언한다.

- **사용된 개념과 주차:**
 - 2주차: constant(상수)
-

```
// 주차장 크기 설정
const int rows = 4; // 주차장의 세로 길이
const int cols = 10; // 주차장의 가로 길이
```

- **설명:**
 - 주차장의 크기(칸 수)를 설정, 행과 열로 나타낸다.
 - 이는 변경되지 않는 사항이기 때문에 const로 선언한다.

- **사용된 개념과 주차:**
 - 2주차: constant(상수)
-

```
// 2차원 벡터 선언(주차 상태 표시)
vector<vector<int>> parking; // 각 칸의 주차 공간 타입 저장
vector<vector<int>> motorbike_count; // 각 칸에 주차된 오토바이 개수 저장
vector<vector<int>> parking_status; // 각 칸의 주차 상태 저장
```

- **설명:**
 - 주차장을 나타내기 위한 2차원 벡터 선언(parking)
 - 각 칸에 있는 오토바이의 개수 벡터 선언(motorbike_count)
 - 각 칸의 주차 상태 벡터 선언(parking_status)
 - 0대, 1대, 2대 주차된 오토바이 표현(motorbike_count)
 - 빈칸인지 채워진 칸인지 표현(parking_status)

- **사용된 개념과 주차:**
 - 10주차: vector(2차원 벡터)
-

```
// 주차 공간 타입에 대한 심볼 설정
char ParkingSymbol (int status, int type, int motorbike_count = 0) {
    if (type == MOTORBIKE) {
        if (motorbike_count == 2) return 'M'; // 두 대 주차된 경우
        else if (motorbike_count == 1) return 'm'; // 한 대 주차된 경우
    }

    if (status == OCCUPIED) { // 주차된 상태일 경우 대문자로 표시
        switch (type) {
            case COMPACT: return 'C';
            case REGULAR: return 'R';
            case LARGE: return 'L';
            case ELECTRIC: return 'E';
            default: return ' ';
        }
    }
    else { // 비어 있는 상태일 경우 소문자로 표시
        switch (type) {
            case COMPACT: return 'c';
            case ELECTRIC: return 'e';
            default: return ' ';
        }
    }
}
}
```

- 반환값:

- 각 차량별, 점유별로 부여된 Symbol
- 오토바이 한 대-m, 오토바이 두 대-M, 경차-C, 일반차-R, 대형차-L, 전기차-E, 빈자리-' '
- (비어있는 특수자리 표현)경차 전용 자리-c, 전기차 충전 자리-e

- 설명:

- 들어온 type이 MOTORBIKE인 경우 각 자리의 오토바이 개수를 확인하고 한 대일 경우 m, 두 대일 경우 M으로 반환
- 주차된 상태의 경우 대문자로 반환
- 비어있는 상태 + 특수 자리일 경우 소문자로 반환
- EMPTY(0)의 경우 공백 symbol할당

- 사용된 개념과 주차:

- 4주차: Condition(switch)
- 4주차: Condition(if, else if)
- 6주차/7주차: function(char 함수)

```

// 초기 주차 공간 세팅
void SetParking() {
    for (int i = 0; i < ROWS; i++) {
        for (int j = 0; j < COLS; j++) {
            parking[i][j] = EMPTY; // 초기 상태를 빈공간으로
            parking_status[i][j] = VACANT; // 주차되지 않은 상태로 저장
        }
    }

    // 왼쪽, 오른쪽 끝 자리를 경차 전용자리로
    for (int i = 0; i < ROWS; ++i) {
        parking[i][0] = COMPACT;
        parking[i][COLS - 1] = COMPACT;
    }

    // 마지막 행의 2~5열(4칸)을 전기차 충전자리로
    for (int i = 1; i < 5; ++i) {
        parking[ROWS - 1][i] = ELECTRIC;
    }
}

```

- 반환값:

- 주차자리 기준 별 C(경차자리) or E(전기차 충전 자리)로 채워진 주차장
- 모든 칸에 대해 초기 상태 설정(빈 공간)(주차되지 않은 상태)

- 설명:

- 4x10크기의 주차장에서 왼쪽, 오른쪽 끝 자리는 경차 전용자리로 설정
- 4x10크기의 주차장에서 마지막 행의 2~5까지의 4칸을 전기차 충전자리로 설정
- 벡터로 선언된 parking을 통해 각 특정 자리에 특징 추가(경차 or 전기차)
- 전체 주차장을 돌면서 상태 설정(EMPTY, VACANT)

- 사용된 개념과 주차:

- 6/7주차: function(void 함수)
- 10주차: vector(2차원 벡터의 인덱스 접근)

```

// 주차 상태 맵 출력 함수
void DisplayMap() {
    // 가로 경계선
    int width = COLS * 4 + 1;
    string line;
    for (int i = 0; i < width; ++i) {
        line += '-';
    }

    // 각 자리에 대한 주차 자리(상태) 출력
    // '\'를 통해 구분
    for (int i = 0; i < ROWS; ++i) {
        cout << line << endl;
        for (int j = 0; j < COLS; ++j) {
            // 공간에 맞는 symbol 가져오기
            // 오토바이의 개수에 따른 symbol 가져오기
            char symbol = ParkingSymbol(parking_status[i][j], parking[i][j], motorbike_count[i][j]);
            cout << "| " << symbol << " ";
        }
        cout << "|" << endl;
    }
    cout << line << endl;
}

```

- 반환값:

- 특수한 자리와 현재 주차된 상태가 나타난 주차장 출력

- 설명:

- '-'를 통해 가로로 행 분리
- 'w'를 통해 세로로 열 분리
- 행과 열만큼 반복하여 parking의 각 자리에 나타난 값(주차 자리 or 상태 표시)에 맞는 symbol 출력

- 사용한 개념과 주차:

- 6/7주차: function(void 함수)
- 7주차: string(문자열)
- 5주차: Loop(for)(중첩 for)
- 10주차: vector(2차원 벡터의 인덱스 접근)

```

// 차량 타입에 따른 주차 공간 추천 함수
vector<pair<int, int>> recommend_spots(int type, bool charging = false, int max = 5) {
    vector<pair<int, int>> recommend; // 추천되는 주차 공간을 저장할 벡터
    vector<pair<int, int>> compact_only; // 경차 전용 자리 벡터
    vector<pair<int, int>> half_full; // 이미 오토바이가 한 대 있는 공간을 저장할 벡터
    vector<pair<int, int>> empty; // 비어 있는 공간을 나타내는 벡터

    // 주차 자리를 순차적으로 훑으며 탐색
    // 일정 개수를 채울 경우 탐색 중단 후 반환
    for (int i = 0; i < ROWS; ++i) {
        for (int j = 0; j < COLS; ++j) {
            // 이미 점유된 자리는 무시
            if (parking_status[i][j] == OCCUPIED)
                continue;

            // 오토바이일 경우, 오토바이의 개수(0 or 2)에 따라 추천
            if (type == MOTORBIKE) {
                if (motorbike_count[i][j] == 1 && parking[i][j] == MOTORBIKE) {
                    // 이미 한 대가 주차된 공간인 경우 높은 우선순위를 가진 벡터에 추가
                    half_full.push_back({i, j});
                }
                else if (motorbike_count[i][j] == 0 && parking[i][j] == EMPTY) {
                    // 빈 공간인 경우 낮은 우선순위를 가진 벡터에 추가
                    empty.push_back({i, j});
                }
            }

            // 경차일 경우 우선적으로 경차 전용 자리를 확인
            else if (type == COMPACT) {
                if (parking[i][j] == COMPACT) {
                    compact_only.push_back({i, j});
                }
                else if (parking[i][j] == EMPTY) {
                    empty.push_back({i, j});
                }
            }

            // 전기차일 경우, 충전 여부에 따라 추천
            else if (type == ELECTRIC) {
                if (charging && parking[i][j] == ELECTRIC) {
                    recommend.push_back({i, j}); // 충전이 필요한 경우 전기차 자리 추천
                }
                else if (!charging && parking[i][j] == EMPTY) {
                    recommend.push_back({i, j}); // 충전이 아닌 경우 일반 자리 추천
                }
            }

            // 일반 차량 주차 공간 추천
            else if (type == COMPACT && (parking[i][j] == COMPACT || parking[i][j] == EMPTY))
                recommend.push_back({i, j});
            else if (type == LARGE && (j < COLS - 1 && (parking[i][j] == EMPTY && parking[i][j + 1] == EMPTY)))
                recommend.push_back({i, j});
            else if (type == REGULAR && parking[i][j] == EMPTY)
                recommend.push_back({i, j});
        }
    }
}

```

```

// 우선순위에 따라 정렬
if (type == COMPACT) {
    recommend.insert(recommend.end(), compact_only.begin(), compact_only.end());
    recommend.insert(recommend.end(), empty.begin(), empty.end());
}
else if (type == MOTORBIKE) {
    recommend.insert(recommend.end(), half_full.begin(), half_full.end());
    recommend.insert(recommend.end(), empty.begin(), empty.end());
}

// 최대 개수를 초과한 경우 조정
if (recommend.size() > max) {
    recommend.resize(max);
}
return recommend;
}

```

- 반환값:

- 추천자리로 채워진 벡터 반환

- 설명:

- 벡터 함수 안에 벡터를 생성하여 조건에 해당하는 주차 자리를 recommend 벡터에 push
- 자리 추천 시, 5자리 중 우선하여 자리를 추천하기 위해 경차 전용 자리 벡터(compact_only)(half_full) 선언(경차입력-경차 전용자리 우선, 오토바이 입력-오토바이 한 대 위치한 자리 우선)
- 이미 채워진 자리(OCCUPIED)의 경우 넘어가기
- 행과 열만큼 반복하여 각 차량 종류별로 조건을 확인한 후 그 자리를 추천 목록에 추가
- 전기차: 전기차이며 + 충전을 원하고 + 자리가 전기차 전용인 경우
- 전기차: 전기차이며 + 충전을 원하지 않고 + 자리가 비어있는 경우
- 경차: 경차이며 + 자리가 경차 전용이고(부족하면 일반 자리) + 자리가 비어있는 경우
- 대형차: 대형차이며 + 열이 마지막 위치보다 1 왼쪽이고 + 자리와 그 옆자리가 비어있는 경우
- 일반차: 일반차이며 + 자리가 비어있는 경우
- 오토바이: 오토바이이며 + 자리가 비어있는 경우(한 대만 채워진 경우)
- 벡터에 값이 5개까지 쌓이면 탐색을 종료하고 이 자리 목록을 반환한다.
- 오토바이와 경차의 경우 5개의 추천 자리를 우선순위에 따라 insert (벡터의 끝에)오토바이-한 대 주차된, 빈자리, 경차-경차 전용, 일반자리

- type외의 인자값을 미리 지정하여, 조건이 필요없는 차량의 사용을 용이화

- **사용한 개념과 주차:**

- 5주차: Loop(중첩 for)
- 10주차: vector(2차원 벡터의 인덱스 접근)
- 10주차: vector(2차원 벡터)
- 10주차: vector(2차원 벡터의 값 선언(push_back))
- 10주차: vector(2차원 벡터의 값 선언(insert))
- 6/7주차: function(함수)
- 4주차: Condition(if, else if)

```
// 차량 주차 함수
void park(int row, int col, int type) {
    // 입력 범위 확인
    if (row < 0 || row >= ROWS || col < 0 || col >= COLS) {
        cout << "주차장을 벗어난 위치입니다." << endl;
        return; // 잘못된 범위
    }

    // 이미 점유된 자리인지 확인
    if (parking_status[row][col] == OCCUPIED) {
        cout << "해당 자리는 이미 점유 중입니다." << endl;
        return; // 이미 점유된 자리
    }

    // 전용 주차 공간 확인 및 경고문 출력
    if (parking[row][col] == COMPACT && type != COMPACT) {
        cout << "이 자리는 경차 전용 자리입니다. 다른 차량은 주차할 수 없습니다." << endl;
        return; // 경차가 아닌 차량의 전용 자리 주차 방지
    }
    if (parking[row][col] == ELECTRIC && type != ELECTRIC) {
        cout << "이 자리는 전기차 충전 전용 자리입니다. 다른 차량은 주차할 수 없습니다." << endl;
        return; // 전기차가 아닌 차량의 전용 자리 주차 방지
    }
}
```

```

// 대형 차량일 경우 두 칸 확인 및 처리
if (type == LARGE) {
    if (col < COLS - 1 && parking_status[row][col] == VACANT && parking_status[row][col + 1] == VACANT) {
        // 채워짐 처리 후
        parking_status[row][col] = OCCUPIED;
        parking_status[row][col + 1] = OCCUPIED;
        // 주차
        parking[row][col] = LARGE;
        parking[row][col + 1] = LARGE;
    }
}
// 오토바이 주차 처리
else if (type == MOTORBIKE) {
    motorbike_count[row][col]++; // 채워진 오토바이 수 증가
    if (motorbike_count[row][col] == 2) {
        // 채워짐 처리 후
        parking_status[row][col] = OCCUPIED;
        // 주차
        parking[row][col] = MOTORBIKE;
    }
    // 오토바이 한 대 주차
    parking[row][col] = MOTORBIKE;
}
else {
    // 다른 차량 주차 처리
    parking_status[row][col] = OCCUPIED;
    parking[row][col] = type;
}

DisplayMap(); // 주차 후 맵 출력
}

```

- 반환값:

- Parking 벡터에서 해당하는 행과 열에 해당하는 차량 저장
- 차량 대입 후 현재 맵 출력
- 선택된 자리를 OCCUPIED 처리
- 잘못된 범위 or 이미 채워진 자리를 선택할 경우 경고문 출력
- 전용 주차 공간을 선택할 경우 경고문 출력

- 설명:

- 대형차(LARGE)의 경우 입력된 칸 옆까지 2자리를 차지
- 오토바이(MOTORBIKE)의 경우 채워진 오토바이의 수에 따라 주차
- 그 외의 경우 입력된 행과 열에 해당하는 칸만 차량 symbol값 대입
- 대입이 완료된 후에는 주차장의 모습 출력
- 주차하려는 자리의 상태와 차량의 type을 확인 후, 차량을 주차할 수 없는 공간일 경우 경고문 출력

- 사용된 개념과 주차:

- 6/7주차: function(void 함수)
- 4주차: Condition(if, else)

```
// 주차장이 포화 상태인지 확인하는 함수
bool IsParkingFull() {
    for (int i = 0; i < ROWS; i++) {
        for (int j = 0; j < COLS; j++) {
            if (parking_status[i][j] == VACANT) {
                return false; // 한 칸이라도 빈칸이 있으면 포화 false
            }
        }
    }
    return true; // 모든 칸이 주차 완료 상태
}
- }
```

- 반환값:

- 주차장의 모든 칸이 점유된 경우 -> true
- 주차장에 빈 칸이 존재하는 경우 -> false

- 설명:

- 주차장의 전체 행렬을 돌면서 주차 상태가 VACANT인 칸이 있는지 확인
- 한 칸이라도 발견되면 -> false, 발견되지 않으면 -> true

- 사용된 개념과 주차:

- 6/7주차: function(void 함수)
- 4주차: Condition(if)
- 5주차: Loop(중첩 for)

-----main문 시작-----

```
int main() {
    // 상태 대입
    // 행의 크기만큼 열을 제작하여 push
    // 비어있는 상태(EMPTY)로 초기화
    for (int i = 0; i < ROWS; ++i) {
        vector<int> row(COLS, EMPTY); // 주차 공간 타입 초기화
        vector<int> status_row(COLS, VACANT); // 주차 상태 초기화
        vector<int> motorbike_row(COLS, 0); // 오토바이 개수 초기화
        parking.push_back(row);
        parking_status.push_back(status_row);
        motorbike_count.push_back(motorbike_row);
    }
    SetParking(); // 초기 설정
    DisplayMap(); // 맵 출력
- }
```

- 설명:

- 반복문을 통해 주차장의 상태를 나타내는 parking 초기화
- 행만큼 반복하여 EMPTY(symbol)로 채워진 열을 반환
- 이를 행 하나하나에 push하여 parking 채우기

- 초반 설정으로서 모든 칸은 EMPTY가 된다.
- 반복문을 통해 주차 상태를 나타내는 status_row 초기화
- 행만큼 반복하여 VACANT(빈)로 채워진 열을 반환
- 이를 행 하나하나에 push하여 parking_status 채우기
- 반복문을 통해 오토바이의 개수를 나타내는 motorbike_row 초기화
- 행만큼 반복하여 0(0대)으로 채워진 열을 반환
- 이를 행 하나하나에 push하여 motorbike_count 채우기
- 이후 setParking함수를 통해 특수한 자리 설정
- 완성된 parking에 따른 주차장의 모습 출력

- 사용된 개념과 주차:

- 10주차: vector(2차원 벡터)
- 5주차: Loop(for)
- 10주차: vector(2차원 벡터의 값 선언(push_back))

```
// mud game처럼 종료신호를 받기 전까지 무한 루프
while(true) {
    // #0. 주차장의 포화상태 확인하기
    if (IsParkingFull()) {
        cout << "주차장의 모든 칸이 포화 상태입니다. 시스템을 종료합니다." << endl;
        break;
    }
}
```

- 설명:

- 무한루프를 도는 while을 통해 프로그램 동작
- 프로그램 종료는 break 사용
- IsParkingFull()함수를 통해 주차장 칸의 포화 상태 확인
- 포화 상태일 경우 안내문을 출력하고 프로그램 종료

- 사용한 개념과 주차:

- 5주차: Loop(while)
- 4주차: Condition(if, else if)

```
// #1. 명령어 입력받기
string command;
cout << "차량의 종류나 옵션을 입력해주세요 (motorbike, compact, regular, large, electric, map, exit) : ";
cin >> command;
```

- 설명:
 - String command를 통해 사용자의 차량, 옵션 입력 받고 변수 저장
 - 사용한 개념과 주차:
 - 7주차: 문자열(string)
-

```
// #2. exit 명령어가 들어온 경우
if (command == "exit")
|   break; // 프로그램 종료
// #3. map 명령어가 들어온 경우
else if (command == "map") {
|   displayMap(); // 현재 상태의 map 출력
|   continue; // 다음 명령어 받기
}
```

- 설명:
 - 입력된 명령어가 exit인 경우 break를 통해 while문 탈출 후 프로그램 종료
 - 입력된 명령어가 map인 경우 현재 주차장의 모습을 출력하는 displayMap 출력 후 while문의 위로(다시 명령어 받기)(continue)
 - 사용한 개념과 주차:
 - 4주차: Condition(if, else if)
-

```
// #4. 전기차의 충전 선택 확인
bool charging = false; // 기본 상태는 충전x
char answer;
if (command == "electric") {
|   cout << "충전을 원하십니까? (y/n) : ";
|   cin >> answer;
|   if (answer == 'y') {
|       |   charging = true; // 충전을 원할 경우 TRUE
|   }
}
}
```

- 설명:
 - 전기차의 경우 충전 확인을 위한 추가 질문이 필요
 - Charging을 통해 충전여부 확인(t-충전, f-충전x)
 - 사용자의 대답이 y인 경우 true로, n인 경우 false로 설정
- 사용한 개념과 주차:
 - 4주차: Condition(중첩 if)

```

// #5. 차량 타입에 따른 주차 공간 추천
vector<pair<int, int>> recommended_spots;    // 추천된 주차 공간(주소)를 저장할 벡터

// 각 차량 타입을 확인하고 이에 따라 자리 추천
if (command == "motorbike") {
    recommended_spots = recommend_spots(MOTORBIKE);
}
else if (command == "compact") {
    recommended_spots = recommend_spots(COMPACT);
}
else if (command == "regular") {
    recommended_spots = recommend_spots(REGULAR);
}
else if (command == "large") {
    recommended_spots = recommend_spots(LARGE);
}
// 충전이 필요한 전기차의 경우
else if (command == "electric" && charging) {
    recommended_spots = recommend_spots(ELECTRIC, true);
}
// 충전이 필요없는 전기차의 경우
else if (command == "electric") {
    recommended_spots = recommend_spots(ELECTRIC);
}
// 잘못된 입력의 경우
else {
    cout << "잘못된 입력입니다." << endl;
    continue;
}

```

- 설명:

- 각 차량종류 별 추천받은 자리를 저장하기 위한 벡터 선언
- 입력받은 명령어를 확인하여 이에 따른 recommendSpots함수의 추천 자리를 받고 이를 벡터에 저장
- 전기차의 경우 충전여부(charging)를 확인하여 필요한 경우 charging 파라미터에 true 조건을 추가하여 함수 실행
- 잘못된 입력을 받은 경우 안내문을 출력하고 while문의 맨 위로(재입력)

- 사용한 개념과 주차:

- 10주차: vector(2차원 벡터)
 - 4주차: Condition(if, else if)
-

```

// #6. 추천된 자리 출력 및 주차
if (recommended_spots.empty()) {
    cout << "해당 차량의 추천 자리가 없습니다." << endl;
    continue;
}

cout << "추천 주차 자리 : ";
for (auto spot : recommended_spots) {
    cout << "(" << spot.first << ", " << spot.second << ") ";
}
cout << endl;

// 자리 선택
int select_r, select_c;
cout << "주차할 자리를 선택하세요 (행, 열) : ";
cin >> select_r >> select_c;

// 추천 자리를 선택했는지 확인
bool is_recommended_spot = false;
for (auto spot : recommended_spots) {
    if (spot.first == select_r && spot.second == select_c) {
        is_recommended_spot = true;
        break;
    }
}

// 추천 자리를 선택하지 않은 경우 경고문 출력
if (!is_recommended_spot) {
    cout << "추천 자리가 아닙니다." << endl;
}

// 차량의 종류에 맞게 주차 처리
if (command == "motorbike") {
    park(select_r, select_c, MOTORBIKE);
}
else if (command == "compact") {
    park(select_r, select_c, COMPACT);
}
else if (command == "regular") {
    park(select_r, select_c, REGULAR);
}
else if (command == "large") {
    park(select_r, select_c, LARGE);
}
else if (command == "electric") {
    park(select_r, select_c, ELECTRIC);
}
}

```

- 설명:

- 각 차량종류별 추천받은 자리를 담은 벡터(1개라도 채워진)를 출력
- 이에 따른 사용자의 선택을 입력받기(select_r, select_c)
- 입력된 command를 확인하고 이에 따른 symbol로 parking에 값 초기화
- park함수를 통해 해당 행과 열의 parking symbol 변경(' ' -> 차량)
- 추천 자리 vector(recommended_spots)가 비어있을 경우 이를 알리고 while 문의 맨위로(재입력)
- for문을 통해 선택한 자리와 추천된 자리를 비교 -> 일치하지 않는 경우 안내문 출력(추천 자리 선택 -> true, 추천x 자리 선택 -> false)

- 사용한 개념과 주차:

- 4주차: Condition(중첩 if, else if)
- 5주차: Loop(for)

⑤ 실행 결과

- compact

```
| c | | | | | | | | c |
-----
| c | | | | | | | | c |
-----
| c | | | | | | | | c |
-----
| c | e | e | e | e | | | | c |
-----
```

차량의 종류나 옵션을 입력해주세요 (motorbike, compact, regular, large, electric, map, exit) : compact
추천 주차 자리 : (0, 0) (0, 9) (1, 0) (1, 9) (2, 0)
주차할 자리를 선택하세요 (행, 열) : 0 9

```
| c | | | | | | | | c |
-----
| c | | | | | | | | c |
-----
| c | | | | | | | | c |
-----
| c | e | e | e | e | | | | c |
-----
```

- motorbike


```

| c |   |   |   |   |   |   |   |   | c |
| c |   |   |   |   |   |   |   |   | c |
| c |   |   |   |   |   |   |   |   | c |
| c | e | e | e | e |   |   |   |   | c |

```

차량의 종류나 옵션을 입력해주세요 (motorbike, compact, regular, large, electric, map, exit) : motorbike
 추천 주차 자리 : (0, 1) (0, 2) (0, 3) (0, 4) (0, 5)
 주차할 자리를 선택하세요 (행, 열) : 0 3

```

| c |   |   | m |   |   |   |   |   | c |
| c |   |   |   |   |   |   |   |   | c |
| c |   |   |   |   |   |   |   |   | c |
| c | e | e | e | e |   |   |   |   | c |

```

차량의 종류나 옵션을 입력해주세요 (motorbike, compact, regular, large, electric, map, exit) : motorbike
 추천 주차 자리 : (0, 3) (0, 1) (0, 2) (0, 4) (0, 5)
 주차할 자리를 선택하세요 (행, 열) : 0 3

```

| c |   |   | M |   |   |   |   |   | c |
| c |   |   |   |   |   |   |   |   | c |
| c |   |   |   |   |   |   |   |   | c |
| c | e | e | e | e |   |   |   |   | c |

```

- regular

```

| c |   |   | M |   |   |   |   |   | c |
| c |   |   |   |   |   |   |   |   | c |
| c |   |   |   |   |   |   |   |   | c |
| c | e | e | e | e |   |   |   |   | c |

```

차량의 종류나 옵션을 입력해주세요 (motorbike, compact, regular, large, electric, map, exit) : regular
 추천 주차 자리 : (0, 1) (0, 2) (0, 4) (0, 5) (0, 6)
 주차할 자리를 선택하세요 (행, 열) : 0 6

```

| c |   |   | M |   | R |   |   |   | c |
| c |   |   |   |   |   |   |   |   | c |
| c |   |   |   |   |   |   |   |   | c |
| c | e | e | e | e |   |   |   |   | c |

```

- large

c			M			R			c
c									c
c									c
c	e	e	e	e					c

차량의 종류나 옵션을 입력해주세요 (motorbike, compact, regular, large, electric, map, exit) : large
 추천 주차 자리 : (0, 1) (0, 4) (0, 7) (1, 1) (1, 2)
 주차할 자리를 선택하세요 (행, 열) : 1 2

c			M			R			c
c		L	L						c
c									c
c	e	e	e	e					c

- electric(충전 o)

c			M			R			c
c		L	L						c
c									c
c	e	e	E	e					c

차량의 종류나 옵션을 입력해주세요 (motorbike, compact, regular, large, electric, map, exit) : electric
 충전을 원하십니까? (y/n) : y
 추천 주차 자리 : (3, 1) (3, 2) (3, 4)
 주차할 자리를 선택하세요 (행, 열) : 3 1

c			M			R			c
c		L	L						c
c									c
c	E	e	E	e					c

- electric(충전 x)

```
-----
| c |   |   | M |   |   | R |   |   | c |
-----
| c |   | L | L |   |   |   |   |   | c |
-----
| c |   |   |   |   |   |   |   |   | c |
-----
| c | E | e | E | e |   |   |   |   | c |
-----
차량의 종류나 옵션을 입력해주세요 (motorbike, compact, regular, large, electric, map, exit) : electric
충전을 원하십니까? (y/n) : n
추천 주차 자리 : (0, 1) (0, 2) (0, 4) (0, 5) (0, 7)
주차할 자리를 선택하세요 (행, 열) : 0 7
-----
| c |   |   | M |   |   | R | E |   | c |
-----
| c |   | L | L |   |   |   |   |   | c |
-----
| c |   |   |   |   |   |   |   |   | c |
-----
| c | E | e | E | e |   |   |   |   | c |
-----
```

- map출력

```
-----
| c |   |   | M |   |   | R | E |   | c |
-----
| c |   | L | L |   |   |   |   |   | c |
-----
| c |   |   |   |   |   |   |   |   | c |
-----
| c | E | e | E | e |   |   |   |   | c |
-----
차량의 종류나 옵션을 입력해주세요 (motorbike, compact, regular, large, electric, map, exit) : map
-----
| c |   |   | M |   |   | R | E |   | c |
-----
| c |   | L | L |   |   |   |   |   | c |
-----
| c |   |   |   |   |   |   |   |   | c |
-----
| c | E | e | E | e |   |   |   |   | c |
-----
```

- 모든 칸 포화

```
-----
| C | L | L | M | L | L | R | E | R | C |
-----
| C | R | L | L | R | L | L | L | L | C |
-----
| C | L | L | L | L | L | L | R | R | C |
-----
| C | E | E | E | E | R | L | L | R | C |
-----
주차장의 모든 칸이 포화 상태입니다. 시스템을 종료합니다.
```

- 추천 자리 없음

```

| c | L | L | M | L | L | R | E |   | c |
| c |   | L | L | R | L | L | L | L | c |
| c | L | L | L | L | L | L | R |   | c |
| c | E | e | E | e |   | L | L |   | c |

```

차량의 종류나 옵션을 입력해주세요 (motorbike, compact, regular, large, electric, map, exit) : large
 해당 차량의 추천 자리가 없습니다.

차량의 종류나 옵션을 입력해주세요 (motorbike, compact, regular, large, electric, map, exit) : █

- 추천 이외의 자리

```

| c |   |   | M |   |   | R | E |   | c |
| c |   | L | L |   |   |   |   |   | c |
| c |   |   |   |   |   |   |   |   | c |
| c | E | e | E | e |   |   |   |   | c |

```

차량의 종류나 옵션을 입력해주세요 (motorbike, compact, regular, large, electric, map, exit) : regular

추천 주차 자리 : (0, 1) (0, 2) (0, 4) (0, 5) (0, 8)

주차할 자리를 선택하세요 (행, 열) : 1 4

추천 자리가 아닙니다.

```

| c |   |   | M |   |   | R | E |   | c |
| c |   | L | L | R |   |   |   |   | c |
| c |   |   |   |   |   |   |   |   | c |
| c | E | e | E | e |   |   |   |   | c |

```

- 잘못된 입력(오타)

```

| c |   |   | M |   |   | R | E |   | c |
| c |   | L | L | R |   |   |   |   | c |
| c |   |   |   |   |   |   |   |   | c |
| c | E | e | E | e |   |   |   |   | c |

```

차량의 종류나 옵션을 입력해주세요 (motorbike, compact, regular, large, electric, map, exit) : bike
 잘못된 입력입니다.

차량의 종류나 옵션을 입력해주세요 (motorbike, compact, regular, large, electric, map, exit) : █

- 잘못된 입력(범위)

차량의 종류나 옵션을 입력해주세요 (motorbike, compact, regular, large, electric, map, exit) : regular

추천 주차 자리 : (0, 1) (0, 2) (0, 4) (0, 5) (0, 8)

주차할 자리를 선택하세요 (행, 열) : 12 3

추천 자리가 아닙니다.

주차장을 벗어난 위치입니다.

차량의 종류나 옵션을 입력해주세요 (motorbike, compact, regular, large, electric, map, exit) : █

- 잘못된 입력(점유)

차량의 종류나 옵션을 입력해주세요 (motorbike, compact, regular, large, electric, map, exit) : regular
추천 주차 자리 : (0, 1) (0, 2) (0, 4) (0, 5) (0, 8)
주차할 자리를 선택하세요 (행, 열) : 0 3
추천 자리가 아닙니다.
해당 자리는 이미 점유 중입니다.
차량의 종류나 옵션을 입력해주세요 (motorbike, compact, regular, large, electric, map, exit) : █

- 잘못된 입력(전용 자리)

차량의 종류나 옵션을 입력해주세요 (motorbike, compact, regular, large, electric, map, exit) : regular
추천 주차 자리 : (0, 1) (0, 2) (0, 4) (0, 5) (0, 8)
주차할 자리를 선택하세요 (행, 열) : 0 9
추천 자리가 아닙니다.
이 자리는 경차 전용 자리입니다. 다른 차량은 주차할 수 없습니다.
차량의 종류나 옵션을 입력해주세요 (motorbike, compact, regular, large, electric, map, exit) : regular
추천 주차 자리 : (0, 1) (0, 2) (0, 4) (0, 5) (0, 8)
주차할 자리를 선택하세요 (행, 열) : 3 2
추천 자리가 아닙니다.
이 자리는 전기차 충전 전용 자리입니다. 다른 차량은 주차할 수 없습니다.
차량의 종류나 옵션을 입력해주세요 (motorbike, compact, regular, large, electric, map, exit) : █

⑥ 코드 문제점

- 1. 안내문(경고문)간의 우선순위가 존재하지 않아, 여러 문장이 동시에 출력된다.

차량의 종류나 옵션을 입력해주세요 (motorbike, compact, regular, large, electric, map, exit) : regular
추천 주차 자리 : (0, 1) (0, 2) (0, 4) (0, 5) (0, 8)
주차할 자리를 선택하세요 (행, 열) : 0 3
추천 자리가 아닙니다.
해당 자리는 이미 점유 중입니다.
차량의 종류나 옵션을 입력해주세요 (motorbike, compact, regular, large, electric, map, exit) : █

차량의 종류나 옵션을 입력해주세요 (motorbike, compact, regular, large, electric, map, exit) : regular
추천 주차 자리 : (0, 1) (0, 2) (0, 4) (0, 5) (0, 8)
주차할 자리를 선택하세요 (행, 열) : 0 9
추천 자리가 아닙니다.
이 자리는 경차 전용 자리입니다. 다른 차량은 주차할 수 없습니다.
차량의 종류나 옵션을 입력해주세요 (motorbike, compact, regular, large, electric, map, exit) : regular
추천 주차 자리 : (0, 1) (0, 2) (0, 4) (0, 5) (0, 8)
주차할 자리를 선택하세요 (행, 열) : 3 2
추천 자리가 아닙니다.
이 자리는 전기차 충전 전용 자리입니다. 다른 차량은 주차할 수 없습니다.
차량의 종류나 옵션을 입력해주세요 (motorbike, compact, regular, large, electric, map, exit) : █

차량의 종류나 옵션을 입력해주세요 (motorbike, compact, regular, large, electric, map, exit) : regular
추천 주차 자리 : (0, 1) (0, 2) (0, 4) (0, 5) (0, 8)
주차할 자리를 선택하세요 (행, 열) : 12 3
추천 자리가 아닙니다.
주차장을 벗어난 위치입니다.
차량의 종류나 옵션을 입력해주세요 (motorbike, compact, regular, large, electric, map, exit) : █

⑦ 개선할 점

- ~~오토바이는 한 자리에 2대까지 주차할 수 있다.~~
- ~~이미 오토바이가 한 칸에 한 대가 차 있는 경우 나중 오토바이가 입력되었을 때 그 자리가 우선시된다.~~
- ~~경차자리가 충분할 경우 5개의 추천 목록은 모두 경차자리로 채운다.~~
- ~~이미 채워진 자리는 다른 차량으로 다시 채워지지 않는다.~~
- ~~자리(특성) 표시와 주차 표시는 구분하여 나타낸다.~~
- ~~추천 외의 자리를 선택할 경우 이에 대한 안내 문구를 출력한다.~~
- ~~다른 차는 전용자리를 선택할 수 없도록 한다.~~
- ~~주차장의 모든 칸이 포화되었을 때 이를 알리고 프로그램 종료한다.~~
- ~~잘못된 입력일 경우 경고문 출력한다.~~
- ~~해당 차량의 추천 자리가 없을 경우 안내문을 출력한다.~~
- ~~핵심 문제에 대한 안내문(경고문)만 출력한다.~~
- ~~최대한 대형차의 자리가 많이 남도록 자리를 추천한다.~~