2048 퍼즐게임 보고서

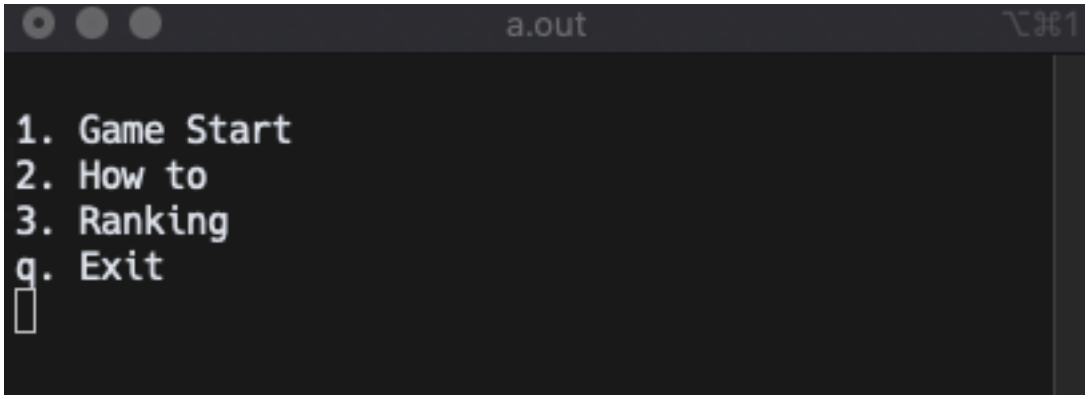
2015-11923 최한결

개발환경:
OS: Mac OS catalina
compiler version: Apple clang version 11.0.0 (clang-1100.0.33.16)

- window wsl(ubuntu) gcc 8.3.0환경에서 시간 출력 기능 정상작동(실시간 업데이트) 안 되는 것을 확인함.

How to build and run this project:
main.c 가 있는 폴더에서 clang main.c 명령어 입력 후 ./a.out 명령어 실행

How to play the game:
우선 게임에서 게임 종료 후 이름 입력할 때를 제외하고는 buffer를 사용하지 않고 키를 입력 받는다.



게임을 실행할 경우 다음과 같은 메뉴가 나오며 각 메뉴에 해당하는 키를 입력하면 각 메뉴가 실행된다. 1,2,3,q를 제외한 input이 들어올 경우 Invalid input이라는 문구가 뜬다.
1을 눌러 게임을 실행시키면 게임을 실행할 수 있다. WASD를 이용해 조작할 수 있고 q를 누르면 메뉴로 빠져나올 수 있다. 시간이 10분이 지나거나 더 이상 움직일 수 없을 경우 게임오버, 2048을 완성시키면 게임 클리어된다.

메뉴 2번을 누르면 간략한 설명이 나온다.
메뉴 3번을 누를 경우 다음과 같이 순위가 나온다.
이 때 순위는 result가 win일 경우, time이 짧을 수록, move가 적을 수록, max_combo가 높을 수록 높은 순위가 매겨진다.

rank	name	result	score	time(sec)	move	max_combo
1.	hank	win	19556	232	741	24
2.	first	lose	2900	22	207	11
3.	first	lose	2900	22	207	11

How to implement several features:

시간을 표시할 때 입력이 없어도 시간이 실시간으로 줄어들게 하고 싶었다. 그래서 무한 루프를 돌면서 시간을 계속 표시하도록 설정하고 _kbhit()라는 함수를 삽입해 키보드를 입력할 경우 관련된 logic이 수행되도록 하여 semi-asynchronous하게 동작하도록 만들었다. 또 시간 update하는 loop가 너무 많이 동작할 필요가 없다고 판단해 usleep(10000)을 사용했다.
랜덤으로 빈 공간에 2 또는 4를 삽입하는 과정에서는 game_board 각 원소의 주소를 갖는 25크기의 1차원 배열 empty_square_pointer와 빈 공간의 크기를 나타내는 empty_square_number를 이용해 빈 공간을 알 수 있도록 한 후 *(empty_square_pointer[rnad()%empty_square_number])와 같이 빈 공간 접근 후 이를 2 또는 4로 업데이트했다. Empty_square_pointer와 Empty_square_number는 각 칸이 업데이트 될 때마다 같이 업데이트 해주었다.

command, merge, move 함수는 wasd 각 입력이 들어왔을 때 실행된다.
우선 merge 함수는 재귀적으로 호출되는 함수이고 어떤 숫자로부터 입력된 방향키와 반대 방향으로 숫자를 탐색 후 같은 숫자가 있을 경우 merge, 다른 숫자가 있을 경우 숫자 방향키 반대방향 쪽 바로 옆칸으로 이동하는 함수이다. 움직임 혹은 merge가 일어날 경우 1을 반환한다.
Move함수는 방향키 방향으로부터 첫 원소부터 방향키 방향으로 옮기는 함수이다. 마찬가지로 move가 일어날 경우 1을 반환한다.

EX). 0 4 2 2 상황에서 a가 입력된 경우:
제일 왼쪽 칸에서 move호출 -> merge 호출 -> 4 0 2 2 -> 첫째 칸 merge 다시 호출 -> 4 2 0 2 -> 두번째 칸 move 호출 -> 두번째 칸 merge 호출 -> 4 4 0 0

```
171 int merge(int row, int col, int search_row, int search_col, int row_dir, int col_dir, unsigned int value, int *merge_time) {
172     if (search_col > 4 || search_col < 0 || search_row > 4 || search_row < 0) {
173         return 0;
174     } else if (game_board[search_row][search_col] == 0) {
175         return merge(row, col, search_row + row_dir, search_col + col_dir, row_dir, col_dir, value, merge_time);
176     } else {
177         unsigned int temp = game_board[search_row][search_col];
178         if (temp == value) {
179             add_empty_square(search_row, search_col);
180             game_board[row][col] = value * 2;
181             score += value * 2;
182             (*merge_time)++;
183         } else if (value == 0) {
184             add_empty_square(search_row, search_col);
185             game_board[row][col] = temp;
186             remove_empty_square(row, col);
187         } else if (search_row == row + row_dir && search_col == col + col_dir) {
188             return 0;
189         } else {
190             add_empty_square(search_row, search_col);
191             game_board[row + row_dir][col + col_dir] = temp;
192             remove_empty_square(row + row_dir, col + col_dir);
193         }
194     }
195     return 1;
196 }
197
198 unsigned int move(int row, int col, int row_dir, int col_dir, int *merge_time) {
199     int new_row = row + row_dir;
200     int new_col = col + col_dir;
201     unsigned int value = game_board[row][col];
202     unsigned int is_moved = merge(row, col, new_row, new_col, row_dir, col_dir, value, merge_time);
203     if (value == 0) {
204         value = game_board[row][col];
205         if (merge(row, col, new_row, new_col, row_dir, col_dir, value, merge_time) == 1) {
206             move(new_row, new_col, row_dir, col_dir, merge_time);
207             is_moved = 1;
208         }
209     } else {
210         is_moved = is_moved | move(new_row, new_col, row_dir, col_dir, merge_time);
211     }
212     return is_moved;
213 }
```