# Sokoban developer documentation.

## Basic working principles:

- First of all, the program should bring the user to the main menu which user can make a decision what to do next such as: play game, how to play, top 5 player, credits and exit.
- If the player clicks on play game, the game should show all the available map and let the user choose. After that the program can bring the user to the game. If the player makes other choice, this program just prints the information that the user want into the screen.
- If the player choose exit, the program should be terminated.
- Every window has a back option, which will bring the user back to the previous window or until the main menu window.
- While playing a game, the program should always show the user how many steps they have already taken so far.
- While playing a game, if the user can't finish the game and want to exit from the game only (not the whole program), the user can exit by click ESC and enter.
- If The user can finish the game, and the counted steps are less than the top 5 recorded score, then the program will ask the user for a name and update the top 5 players with this new player.
- If the user can finish the game but the counted steps are more that the top 5 recorded scores, then the program should just finish the game and be able to bring user back to the main menu.

## Functions and algorithms:

- **Int get_destination(char*)**

  This function take an array of character (which is a map) as an input and has return type is integer.

  The function scans the whole array for the destination, when found the destination, save the index of array of that destination into the array destination[] which is design to save the destination. And count the number of destination found. After finish this process for the whole map, return the number of destination found.

- **void print_map(char*)**

  This function take the array of character (which is a map) as an input and print the whole map. The function doesn't return any thing.

- **void get_position(char*, int*, int*)**

  This function takes an array of character (which is a map) and two address of an integers (which is a x and y position of the player character), the function doesn't return anything.

This function will scan through the whole map and when it found the player position ‘P’ on the map save that position into 2 pointers of integer (so here is the position of the player character).

- **void move_char(char*,int, int, int, int*)**

  This function has 5 parameters: array of character (map), integer (pos_x), integer (pos_y), integer (a number which will determine whether player wanted to move in which direction) and a pointer to integer (to count the step in each move). The function doesn’t return anything.

  This function will check whether the position that player is going to is hitting is a wall, pushing a box, or entering an empty space. If player is pushing a box then check further if after pushing, the box will be in the destination or an empty space, if it is a destination then replace that destination by a completed destination, replace the player character by empty space, replace the box by player character and increment the step by 1. And does the same for other cases as well. If the player is hitting the wall or can’t push the box due to there is some obstracle then do nothing.

- **char* get_map(char*, int)**

  This function has 2 input parameters: array of character (map) and integer (number which indicate which map is chosen). This function will return the pointer to the array of character.

  First, the function will consider which file is said to be load from the file by the number that is given to the function, then open the corresponding file, read each character and copy it into the array of character that is given to the function. After finish return the address that points to the first character of this array.

- **int main_menu(void)**

  This function takes nothing as input parameter and have return type is integer.

  This function just print the main menu to the user and wait for a choice that user is going to make, when the user enter his/her choice (which is integer), the program return this number.

- **int game_play(char*)**

  This function take an array of character as an input parameter and has return type is integer.

  This function will print the map with user’s step so far, then ask the user for which direction user want to move the player character, when the user enter some direction or control the character, move_char function will modify the map as what the user has entered, check if all destinations are filled with the box or not, if no then repeat the previous process.

If all the destinations are filled with boxes then finish the game. After that this function will load the top player's scores from the file and check if the step that this user took is better than any of the recorded score, if no then the program can just bring the user back to the main menu.

If the new score is better than any recorded scores, The program ask the user for his/her name, create a new struct player, store this name and this score into this struct and replace this with the highest recorded score and also shift down all other recorded score that need to be changed

- **int select_map(void)**

  This function take nothing as input parameter and return integer.

  This function will load the existing maps from the file and print them all to the screen, ask the user which map they want to choose, when the user enter the number of the map, this function return this number.

- **void get_howToPlay(void)**

  This function takes nothing as input parameter and return nothing as well.

  The function just loads "howToPlay.txt" file and print how to play the game to the screen. Then wait for the user to go back to the main menu.

- **void get_topPlayer(void)**

  This function takes nothing as input parameter and return nothing as well.

  The function just loads "topPlayer.txt" file and print the top 5 players so far to the screen. Then wait for the user to go back to the main menu.

- **void get_Credits(void)**

  This function takes nothing as input parameter and return nothing as well.

  The function just loads "tCredits.txt" file and print the credits of this program to the screen. Then wait for the user to go back to the main menu.

- **void read_file(char*)**

  This function take the array of character as an input parameter, the function return nothing.

  This function will have the array of character which will be consider as the name of the file that it needs to open, then open this file, print everything inside the file to the screen and wait for the user to read the text or some information that the function printed, when the user has done reading the user can type back option to exit this window or go back.

## Implementations:

- To store the user with his/her best score, we will introduce a struct to create our own data type which can be store the player's name and score.
- Every time when the user needs to make a decision, we will have a function for it, the program should call the function then the function prints what user need to choose, get the input from the user and return the number which is corresponding to what the user have chosen to the previous function that call this function to be use further. Such as: in main menu, selecting a map, etc.
- When we enter the game, the basic idea of the gameplay is:
    - Read the selected map from the map00.txt, store the map into a dynamically growing array.
    - Scan for the whole map for the destination first and store their position into one array. This array will be checked in each move whether in that move the player has successfully finish the game or not, if not continue the game, if yes stop the game.
    - In this program, we will store a whole map in a 1D array of character, we can access to each element of the map by simply the index of the array.
    - Print the map (including player position, box, wall) into the screen.
    - to move player character around the map is just to modify the map that we have copied in each move then print it again.
    - After each move increase the steps that the user have taken and always check the array that store the position of the destination, if they are all finish then stop the game.
    - After the game is stop successfully, open the "top_5_player.txt" file and get the recorded scores into the array of struct. If the new scores that the current user did is better than any of the recorded scores, create a new struct, ask the current user for his/her name and store this name with his/her score into the new struct that we just created, then replace this struct with the struct in array that need to be replace. After that write this array of struct down into "top_5_player.txt" file.