```c
/**
* OBLIG 2
*
* THE PROGRAM:
* -Set up variables and constants
* -Create methods and functions
* -Optimize code
* -Use comments and documentation
*
* @file main.c
* @date 13.10.25
* @version 2.0
* @author Daniel AG
*
* Includes:
* - stdio.h for displaying and feeding data
* - stdbool.h for using booleans
* - ctype.h for toupper
*/

// Internal Includes :
#include <stdio.h> // printf,scanf
#include <stdbool.h> // bool
#include <ctype.h> // toupper
#include <string.h> // strcpy,strcspn

// Constants:
#define MAXLANES 18 ///< Maximum number of lanes
#define MAXLANELENGTH 100 ///< Maximum lane length
#define MAXPARS 8 ///< Maximum Pars
#define MAXSTRLEN 100 ///< Maximum string length

//Declarations
void Add_Lane(int* in_numLanes, int* in_laneLengt,
    int* in_lanePar, bool* in_laneOB,
    char (*in_laneDescription)[MAXSTRLEN]);

void Display_Lane(int* in_numLanes, int* in_laneLengt,
    int* in_lanePar, bool* in_laneOB,
    char (*in_laneDescription)[MAXSTRLEN]);

/**
* main program:
*/
int main()
{
    // Internal Includes :
    char laneDescription[MAXLANES][MAXSTRLEN] = { 0,0 }; // lane
                                                          //-description
    int laneLength[MAXLANES] = { 0 }; // lane length
    int lanePar[MAXLANES] = { 0 }; // lane Par
    bool laneOB[MAXLANES] = { 0 }; // lane OB
    int numLanes = 0; // number of lanes



    // Default values :
    laneLength[0] = 62;
    lanePar[0] = 3;
    laneOB[0] = true;
    strcpy_s(laneDescription[0], MAXSTRLEN,
        "Lane with a lot of trees and scrub");

    laneLength[1] = 94;
    lanePar[1] = 3;
    laneOB[1] = false;
    strcpy_s(laneDescription[1], MAXSTRLEN,
        "Flat terrain thourgout the map");
    numLanes = 2;

    char choice; // User choice for input
    do /// conditional logic in while loop
    {
        /**
        * 1. Printing the menu
        * 2. Getting user input
        * 3. Activate switch case
        * 4. Returning to menu unless Q is pressed
        */
        printf("Menu Choices:\n");
        printf("  A - Add lane:\n");
        printf("  D - Display alle lanes:\n");
        printf("  Q - Quit:\n");
        printf("  Select a choice:");
        scanf_s(" %c", &choice, 1);
        choice = toupper(choice);
        printf("\n");

        switch (choice) /// steps into corresponding case
        {
        case 'A':
        {
            Add_Lane(&numLanes, laneLength, lanePar, laneOB,
              laneDescription);
            break;
        }
        case 'D':
        {
            Display_Lane(&numLanes, laneLength, lanePar, laneOB,
              laneDescription);
            break;
        }
        case 'Q':
        {
            printf("Quit - selected:\nEXITING PROGRAM");
            break;
        }
        default:
            printf("Illegal argument");
        }
    } while (choice != 'Q');
}

/**
*
* Creates a new lane by taking user input.
* In the function, the user is prompted to enter details,
* if alle details are filled correctly a new lane is created.
* Else it fails and returns to the menu.
*
* @param in_numLanes Pointer to number of lanes integer
* @param in_laneLengt Pointer to lane lengths array of integers
* @param in_lanePar Pointer to lane Pars array of integers
* @param in_laneOB Pointer to lane OB array of booleans
* @param in_laneDescription Pointer to lane descriptions of array of
*                            -strings
*/
void Add_Lane(int* in_numLanes, int* in_laneLengt,
    int* in_lanePar, bool* in_laneOB,
    char (*in_laneDescription)[MAXSTRLEN])
{
    if (MAXLANES <= *in_numLanes) /// is the maximum
                                  /// number of lanes reached?
    {
        printf("[LOG]:Max Number of lanes created\n");
    }

    // Temporary variables for user input
    int currentLane = *in_numLanes;
    int qLaneLength;
    int qLanePar;
    char qLaneOB;
    char qLaneDescription[MAXSTRLEN];

    // Take input for lane length
    do  /// Flag if length is less or equal to 0 or input fails
    {
        printf("How long is lane %i: ", currentLane + 1);
        while (getchar() != '\n');
    } while (scanf_s(" %d", &qLaneLength) != 1);

    if (MAXLANES <= currentLane || qLaneLength >= 0) /// Flag if max
                                                     /// -lanes is
                                                     /// -overreached or
                                                     /// -length is valid
    {
        in_laneLengt[currentLane] = qLaneLength; // Update lane length,
                                                 // -for current lane
    }

    // Take input for lane Pars
    do  /// Flag if Pars is less than 2 or input fails
    {
        printf("Pars on the field. Choose a number bwteen (2-8):");
        while (getchar() != '\n');
    } while (scanf_s(" %d", &qLanePar) != 1);
    if (MAXPARS < qLanePar) /// Flag if max Pars is overreached
    {
        printf("Max Pars, set to MAXPARS\n");
        qLanePar = MAXPARS;
    }
    if (qLanePar > 2) /// Flag if Pars is valid
    {
        in_lanePar[currentLane] = qLanePar; // Update lane Pars for
          current lane
    }

    // Take input for lane OB
    do /// Flag if input is not y or n
    {
        printf("Does the lane have OB (y = yes or n = no):");
        scanf_s(" %c", &qLaneOB, 1);
        while (getchar() != '\n');
        qLaneOB = toupper(qLaneOB);
    } while (qLaneOB != 'Y' && qLaneOB != 'N');
    in_laneOB[currentLane] = (qLaneOB == 'Y');  // Update lane OB for
                                                // -current lane

    // Description
    printf("Write a description:"); // Take input for lane description
    fgets(qLaneDescription, MAXSTRLEN, stdin); // Read string with spaces

    // Remove newline character from string "\n"
    qLaneDescription[strcspn(qLaneDescription, "\n")] = 0;

    // Copy the content of qLaneDescription to laneDescription
```

```c
    strcpy_s(in_laneDescription[*in_numLanes], MAXSTRLEN,
      qLaneDescription);

    (*in_numLanes)++; // Increment number of lanes
    printf("Lane %i added\n\n", currentLane + 1); // Confirm lane added

}

/**
*
* Retrieving data from all lanes for displaying.
* It takes pointers to all the relevant arrays and the number of lanes,
* then iterates through each lane to print its details.
*
*
* @param in_numLanes Pointer to number of lanes integer
* @param in_laneLengt Pointer to lane lengths array of integers
* @param in_lanePar Pointer to lane Pars array of integers
* @param in_laneOB Pointer to lane OB array of booleans
* @param in_laneDescription Pointer to lane descriptions of array of
*                          -strings
**/
void Display_Lane(int* in_numLanes, int* in_laneLengt, int* in_lanePar,
  bool* in_laneOB, char (*in_laneDescription)[MAXSTRLEN])
{
    int totNumPars = 0; // Total number of Pars
    int i = 0;

    if (*in_numLanes == 0) /// Flag if no lanes are available
    {
        printf("No lanes available\n\n");
    }
    if (in_laneLengt[i] == 0) /// Flag if no data on lane
    {
        printf("No data on lane %d\n\n", i);
    }
    do /// Iterate through all lanes only once
    {
        printf("Lane \033[1;4m%d\033[0m:\n", i + 1);
        printf("     Length: \033[1;4m%d meters\033[0m\n", in_laneLengt
          [i]);
        printf("     Pars: \033[1;4m%d\033[0m\n", in_lanePar[i]);
        printf("     OB: %s\n", in_laneOB[i] ?
            "\033[1;4m With \033[0m" : "\033[1;4m Without \033[0m");
        if (in_laneDescription[i])
        {
            printf("     Description: \033[1;4m%s\033[0m\n\n",
                in_laneDescription[i]);
        }
```

```c
            i++;
            totNumPars += in_lanePar[i]; // Sum up total number of Pars
        } while (i != *in_numLanes);

        printf("Summary\n");
        printf("Total number of lanes: \033[1;4m%d\033[0m\n", *in_numLanes);
        printf("To get to Par, it requires number of throws:\033[1;4m%d\033[0m
          \n\n",
            totNumPars);
}

```