```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
```

```
49      int laneLength[MAXLANES] = { 0 }; // lane length
50      int lanePar[MAXLANES] = { 0 }; // lane Par
51      bool laneOB[MAXLANES] = { 0 }; // lane OB
52      int numLanes = 0; // number of lanes
53
54
55
56      // Default values :
57      laneLength[0] = 62;
58      lanePar[0] = 3;
59      laneOB[0] = true;
60      strcpy_s(laneDescription[0], MAXSTRLEN,
61          "Lane with a lot of trees and scrub");
62
63      laneLength[1] = 94;
64      lanePar[1] = 3;
65      laneOB[1] = false;
66      strcpy_s(laneDescription[1], MAXSTRLEN,
67          "Flat terrain thourgout the map");
68      numLanes = 2;
69
70      char choice; // User choice for input
71      do /// conditional logic in while loop
72      {
73          /**
74          * 1. Printing the menu
75          * 2. Getting user input
76          * 3. Activate switch case
77          * 4. Returning to menu unless Q is pressed
78          */
79          printf("Menu Choices:\n");
80          printf("  A - Add lane:\n");
81          printf("  D - Display alle lanes:\n");
82          printf("  Q - Quit:\n");
83          printf("  Select a choice:");
84          scanf_s(" %c", &choice, 1);
85          choice = toupper(choice);
86          printf("\n");
87
88          switch (choice) /// steps into corresponding case
89          {
90          case 'A':
91          {
92              Add_Lane(&numLanes, laneLength, lanePar, laneOB,
                  laneDescription);
93              break;
94          }
95          case 'D':
96          {
```

```
97              Display_Lane(&numLanes, laneLength, lanePar, laneOB,
                  laneDescription);
98              break;
99          }
100         case 'Q':
101         {
102             printf("Quit - selected:\nEXITING PROGRAM");
103             break;
104         }
105         default:
106             printf("Illegal argument");
107         }
108     } while (choice != 'Q');
109 }
110
111 /**
112 *
113 * Creates a new lane by taking user input.
114 * In the function, the user is prompted to enter details,
115 * if alle details are filled correctly a new lane is created.
116 * Else it fails and returns to the menu.
117 *
118 * @param in_numLanes Pointer to number of lanes integer
119 * @param in_laneLengt Pointer to lane lengths array of integers
120 * @param in_lanePar Pointer to lane Pars array of integers
121 * @param in_laneOB Pointer to lane OB array of booleans
122 * @param in_laneDescription Pointer to lane descriptions of array of
    strings
123 */
124 void Add_Lane(int* in_numLanes, int* in_laneLengt,
125     int* in_lanePar, bool* in_laneOB,
126     char (*in_laneDescription)[MAXSTRLEN])
127 {
128     if (MAXLANES <= *in_numLanes) // is the maximum number of lanes
        reached?
129     {
130         printf("[LOG]:Max Number of lanes created\n");
131     }
132
133     // Temporary variables for user input
134     int currentLane = *in_numLanes;
135     int qLaneLength;
136     int qLanePar;
137     char qLaneOB;
138     char qLaneDescription[MAXSTRLEN];
139
140     // Take input for lane length
141     do  /// Flag if length is less or equal to 0 or input fails
142     {
```

```
143         printf("How long is lane %i: ", currentLane + 1);
144         while (getchar() != '\n');
145     } while (scanf_s(" %d", &qLaneLength) != 1);
146
147     if (MAXLANES <= currentLane || qLaneLength >= 0) /// Flag if max
        lanes is
148                                                      /// -overreached or
149                                                      /// -length is valid
150     {
151         in_laneLengt[currentLane] = qLaneLength; // Update lane length,
152                                                  // -for current lane
153     }
154
155     // Take input for lane Pars
156     do  /// Flag if Pars is less than 2 or input fails
157     {
158         printf("Pars on the field. Choose a number bwteen (2-8):");
159         while (getchar() != '\n');
160     } while (scanf_s(" %d", &qLanePar) != 1);
161     if (MAXPARS < qLanePar) /// Flag if max Pars is overreached
162     {
163         printf("Max Pars, set to MAXPARS\n");
164         qLanePar = MAXPARS;
165     }
166     if (qLanePar > 2) /// Flag if Pars is valid
167     {
168         in_lanePar[currentLane] = qLanePar; // Update lane Pars for
            current lane
169     }
170
171     // Take input for lane OB
172     do /// Flag if input is not y or n
173     {
174         printf("Does the lane have OB (y = yes or n = no):");
175         scanf_s(" %c", &qLaneOB, 1);
176         while (getchar() != '\n');
177         qLaneOB = toupper(qLaneOB);
178     } while (qLaneOB != 'Y' && qLaneOB != 'N');
179     in_laneOB[currentLane] = (qLaneOB == 'Y');  // Update lane OB for
        current lane
180
181     // Description
182     printf("Write a description:"); // Take input for lane description
183     fgets(qLaneDescription, MAXSTRLEN, stdin); // Read string with spaces
184
185     // Remove newline character from string "\n"
186     qLaneDescription[strcspn(qLaneDescription, "\n")] = 0;
187
188     // 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]);
        }
```

```
234         i++;
235         totNumPars += in_lanePar[i]; // Sum up total number of Pars
236     } while (i != *in_numLanes);
237 
238     printf("Summary\n");
239     printf("Total number of lanes: \033[1;4m%d\033[0m\n", *in_numLanes);
240     printf("To get to Par, it requires number of throws:\033[1;4m%d\033[0m ↵
          \n\n",
241         totNumPars);
242 }
243 
```