Fundamentals of Programming II

Help for the Practice (Part I)

Index

1. Introduction

- ✓ This presentation is to help on the development of the part I of the practice.
- ✓ Needed: some practical work on **lists**, **modules** and **matrixes**.
- ✓ ¡Practice with exercises if you have not done it yet!
- ✓ This help is optional. You do not need to follow this help in order to solve the part I of the practice.

2. Planning

- ✓ The planning for the laboratory sessions is as follows:
  - 26/2: Mine Module (start)
  - 4/3: Mine Module (end)
  - 11/3: Game Module (start)
  - 18/3: Game Module (end)
  - 25/3: Main Program.
  - 1/4: Final Testing and Debugging.

3. Project and Modules

- ✓ Create a **new Project** for the Part I of the Practice in Visual Studio.
- ✓ The different modules are incorporated by creating the **header files** (.h files) and **implementation files** (.cpp files) in the corresponding directories of the solution explorer (right window of Visual Studio).
- ✓ Start working on Part I by adding the following C++ files into the corresponding directories: **mine.h** and **mine.cpp**.
- ✓ In *mine.cpp*, use the *#include* directive for including *mine.h*.
- ✓ Do the same with the **game** module, considering also that this new module uses the mine module.

4. Implementation of the Mine Module

- ✓ You must include in the file *mine.h* the declarations of constants and types of the mine module, as already introduced in the assignment:

```
const int MAX = 50;
typedef enum tCell { FREE, SOIL, GEM, STONE, WALL, DYNAMITE, EXIT,
MINER };
typedef tCell tPlane[MAX][MAX];
typedef struct {
   tPlane mine_plane;
   int row; int column;
   int nrows; int ncolumns;
} tMine;
typedef char tPlaneCharacters[3 * MAX][3 * MAX];
typedef int tPlaneColours[3 * MAX][3 * MAX];
```

- ✓ The prototypes of the subprograms requested in the assignment are as follows :

```
void loadMine(ifstream& file, tMine& mine); // reads data from a file, count the number of gems it contains and saves the miner's position

void draw1_1(const tMine& mine); // draws the mine at 1:1 scale

void draw1_3(const tMine& mine); // draws the mine at 1:3 scale and uses the following function draw3x3

void draw3x3(tCell cell, tPlaneCharacters &characters, tPlaneColours &colours, int i, int j); // draws the squares enlarged three times. Specifically, the cell serves to update the plane of characters and colours in the coordinates i, j.

void backgroundColour(int colour); // establishes the background colour to color with the foreground colour white. You must use this function before drawing a cell and then use it again to return background colour to black (0).
```

- Consider also implementing the following functions for your solution:

```
tCell load_cell(char c); // returns one of the values of tCell
depending on the value of char c.

void character_colour(tCell cell, char& character, int& numColour);
// returns the character and the numColour of the tCell cell. This
function is used for cells where the same character is repeated:
FREE, STONE, SOIL and WALL.
```

- Implement the *load_cell* function with an appropriate switch.
- Implement the *loadMine* procedure using two nested for loops and calling the *load_cell* function.
- Once you read the number of rows, and the number of columns in the first line of the file (see the assignment), you must read one character for the end of the line. The same occurs when you read each line of the characters. You must read the end of the line.

- ✓ Implement *character_colour* procedure using a switch.
- ✓ Implement the *draw1_1* procedure, declaring the following local variables:

```
char plane_Characters[MAX][MAX];
int plane_Colours[MAX][MAX];
```

- ✓ For implementing this procedure, consider the following pseudocode:

```
Traverse the plane of the mine
   Depending on the value of mine.mine_plane[i][j]
      Put the appropriate character in plane_Characters[i][j]
      Put the appropriate colour in plane_Colours[i][j]
      Use character_colour when appropriate
system("cls")
Traverse the plane of the mine
   Call backgroundColour using plane_Colours
   Display plane_Characters using setw
   Call backgroundColour with 0
```

- ✓ Add a **main.cpp** file in the source code of your project. This file is for implementing the main program of your solution.
- ✓ Test your code loading the file *1.txt* and drawing this same mine using the *draw1_1* procedure.

- ✓ Implement *draw3x3* procedure using the *character_colour* function which returns the character and the colour of a cell knowing that they are repeated in a 3x3 matrix.
- ✓ For implementing the *draw3x3* procedure, consider the following pseudocode:

```
Traverse a 3x3 matrix
    Call the character_colour function
    Update the characters plane parameter with the character found.
    Update the colours plane parameter with the colour found.
```

- Implement the *draw1_3* procedure, declaring the following local variables:

```
tPlaneCharacters plane_Characters;
tPlaneColours plane_Colours;
```

- For implementing this procedure, consider the following pseudocode:

```
Traverse the plane of the mine
   Depending on the value of mine.mine_plane[i][j]
      Put the appropriate characters in plane_Characters
      Put the appropriate colours in plane_Colours
      Use draw3x3 when appropriate
system("cls")
Traverse the plane of the mine
   Call backgroundColour using plane_Colours
   Display plane_Characters using setw
   Call backgroundColour with 0
```

- ✓ When you put appropriate characters in *plane_Characters*, you might traverse a 3x3 matrix as follows:

```
plane_Characters[i * 3][j * 3] = 'M';
plane_Characters[i * 3][j * 3 + 1] = 'I';
plane_Characters[i * 3][j * 3 + 2] = '-';
plane_Characters[i * 3 + 1][j * 3] = '-';
plane_Characters[i * 3 + 1][j * 3 + 1] = 'N';
plane_Characters[i * 3 + 1][j * 3 + 2] = 'E';
plane_Characters[i * 3 + 2][j * 3] = 'R';
plane_Characters[i * 3 + 2][j * 3 + 1] = '-';
plane_Characters[i * 3 + 2][j * 3 + 2] = '-';
```

- ✓ The background colours that we have used in our demo and videos are:

```
Miner: 7.
Dynamite: 9.
Gem: 2.
Exit: 1.
Free: 4.
Stone: 4.
Soil: 4.
Wall: 4.
```

- ✓ Change your **main program** and test your new code loading the file *1.txt* and drawing this same mine using the *draw1_3* procedure.