LW: Dungeon Crawler

For the labwork, you are responsible for writing the preliminary game logic of a dungeon crawler computer game, which is a type of game where you control an in-game character who is navigating a fantasy dungeon environment. Your co-workers are still working on the graphics separately from the logic, so you are working on a text-based game prototype for developing and debugging your game logic code.

A grade of complete on this labwork requires 35 of 40 points. Work on your code on Mimir with others in your groups, but submit individually.

# Objectives

* Properly create and store values into a 2D dynamic array using loop iteration.
* Code and call functions with pass by reference.
* Update variables after being passed into functions as references.

# Input Map Text File

This labwork will involve reading a text file that contains information of the dungeon map's internal representation. This text file consists of three parts:

* Line 1: Map Dimensions. This line contains two values for representing the map's width and height, respectively.
* Line 2: Player Starting Location. This line contains two values for representing the player's starting x-coordinate position and y-coordinate position, respectively.
* Lines 3+: These lines contain the individual tile information of the dungeon map as numerical values.
  + The first number in Line 3 represents the map tile at (0, 0), where the first value is the x-coordinate position and the second value is the y--coordinate position.
  + Refer to the tile status constants in helper.h for more details.

The following is an example text file for a 3×3 tile representation of a dungeon map.

|  |
| --- |
| 3 3  1 2  3 0 0  2 2 0  0 0 0 |

* Line 1; The map has a width of 3 and a height of 3.
* Line 2: The player will start at map location (1, 2), where 1 is the x-coordinate location and 2 is the y-coordinate location.
* Lines 3-5: The map's internal representation for each map tile.
* Note #1: Line breaks for lines 3 and after are purely for aesthetic purposes, so you must not assume that line breaks represent the actual map dimensions.
* Note #2: Since you are working with a visual coordinate map, you will once again set up your 2D dynamic array as column-first order like for the Seam Carving homeworks.

The collection of map text files that are used in the Mimir test cases are located on the class's Google Shared Drive under: Labwork → LW: Dungeon Crawler → Sample Map Files

# Step 0. Download and Set Up

1. Download and extract the starter code into your programming environment. Confirm that the starter code consists of the following files:
   1. main.cpp: You must edit this file for the labwork.
   2. helper.cpp: You must not edit this file.
   3. helper.h: You must not edit this file, but you must read it.
2. Compile and run the initial state of the starter code with the following command line.
   1. g++ -std=c++17 -Wall -Wextra -pedantic -Weffc++ -g main.cpp helper.cpp
3. Confirm that the initial state of the starter code executes without errors and with only warnings related to unused variables.
4. Read code documentation of and familiarize with the constants in helper.h file. These constants will be used in editing the main.cpp file.
   1. Tile Status Constants: these constants hold values for representing the tile status on the dungeon map.
      1. TILE\_OPEN
      2. TILE\_PLAYER
      3. TILE\_PILLAR
      4. TILE\_EXIT
   2. Movement Status Flag Constants: these constants hold values for representing the player's movement status flags.
      1. STATUS\_MOVE
      2. STATUS\_STAY
      3. STATUS\_EXIT
   3. User Keyboard Input Constants: these constants hold values for representing the user's keyboard inputs
      1. MOVE\_UP
      2. MOVE\_LEFT
      3. MOVE\_DOWN
      4. MOVE\_RIGHT
      5. INPUT\_QUIT

# Step 1. Write Code for createMap(...) Function

1. Open main.cpp.
2. Go to the main() function and make a note of the following local variables in the first line of that method: width, height, currX, and currY. You will be using these local variables for Step 1.
3. Go to the comment for:
   1. **STEP 1: Write code for createMap(...) function here**.
4. Create the function signature (i.e., the function's return type, identifier, and parameter list) below this comment for creating the dungeon map.
   1. The function's identifier (i.e., the name of the function will be called: createMap
   2. The function's return type will be a double int pointer (i.e., a pointer to an array of int pointers, aka a pointer to a pointer to an int).
   3. The function's parameter list will consist of the following variables:
      1. the file name as a const string
      2. the reference to the map's width as an int
      3. the reference to the map's height as an int
      4. the reference to the current player's x-position as an int
      5. the reference to the current player's y-position as an int

The remaining sub-steps will be written inside the function.

1. Read the file name of a text file containing the map information. Also check to make sure that the file opened successfully. If the file did not open successfully, then:
   1. output the following error message as a cout statement, where <filename> is the name of the file:
      1. ERROR: unable to open: <filename>
   2. close the file reader
   3. return a null pointer
2. Read the first first two lines and store them in their respective variables.
3. Create a new map as a 2D dynamic int array based on the stored map dimensions.
4. Read the remaining lines and store these values into the newly-created map.
5. Store the player's starting location into the map with the stored starting position.
6. Close the file reader and return the map.
7. Go to the main() function and uncomment the lines after completing Step 1. These lines are located before and after the while-loop.

# Step 2. Write Code for updateNextPosition(...) Function

1. Open main.cpp.
2. Go to the main() function and make a note of the following local variables in that method: map, width, height, input, currX, currY, nextX, and nextY. You will be using these local variables for Step 2.
3. Go to the comment for:
   1. **STEP 2: Write code for updateNextPosition(...) function here.**
4. Create the function signature (i.e., the function's return type, identifier, and parameter list) below this comment for creating the dungeon map.
   1. The function's identifier (i.e., the name of the function will be called: updateNextPosition
   2. The function will return an int.
   3. The function's parameter list will consist of the following variables:
      1. the map as an 2D int dynamic array
      2. the map's width as an int
      3. the map's height as an int
      4. the user's keyboard input as a char
      5. the reference to the player's current x-coordinate position as an int
      6. the reference to the player's current y-coordinate position int
      7. the reference to the player's next y-coordinate position int
      8. the reference to the player's next y-coordinate position int
5. Set the player's next position to the player's current position.
6. Calculate the player's position based on the user's keyboard input. Refer to the user's keyboard input constants in helper.h.
7. Check if the player's next position for the following conditions:
   1. If the position is out of bounds, then reset the player's next position to the player's current position. Finally, return the player's movement status as staying.
   2. If the position is located at a pillar, then reset the player's next position to the player's current position. Finally, return the player's movement status as staying.
   3. If the position is located at an exit, then return the player's movement status as escaping.
   4. Otherwise, return the player's movement status as moving.
8. Go to the main() function and uncomment the lines after completing Step 2. These lines are located inside the while-loop.

# Step 3. Write Code for updateMap(...) Function

1. Open main.cpp.
2. Go to the main() function and make a note of the following local variables in that method: map, currX, currY, nextX, and nextY. You will be using these local variables for Step 3.
3. Go to the comment for:
   1. **STEP 3: Write code for updateMap(...) function here.**
4. Create the function signature (i.e., the function's return type, identifier, and parameter list) below this comment for creating the dungeon map.
   1. The function's identifier (i.e., the name of the function will be called: updateMap
   2. The function will return nothing..
   3. The function's parameter list will consist of the following variables:
      1. the map as an 2D int dynamic array
      2. the reference to the player's current x-coordinate position as an int
      3. the reference to the player's current y-coordinate position int
      4. the reference to the player's next y-coordinate position int
      5. the reference to the player's next y-coordinate position int

The remaining sub-steps will be written inside the function.

1. Go to the player's current position in the map and clear the tile as open.
2. Update the player's current position to the player's next position.
3. Go to the player's updated current position in the map and mark the tile as the player.
4. Go to the main() function and uncomment the lines after completing Step 3. These lines are located inside the while-loop.

# [CHALLENGE] Write Code for updateStatistics(...) Function

1. Open main.cpp.
2. Go to the main() function and add the following local variables: numMoves, numUps, numDowns, numLefts, and numRights. You will be using these local variables for this Challenge Step.
   1. Hint: Due to how the code updates the player's movement, you may also want to consider initializing the values of nextX and nextY to -1 to simplify the next sub-steps.
3. Above the main() function, create the function signature using the local variables from sub-step 2 and from any other relevant local variables for updating the following statistics:
   1. numMoves: the total number of moves, both actual moves (e.g., walking left to a new tiel) and stationary moves (e.g., standing still due to being blocked by a pillar)
   2. numUps, numDowns, numLefts, numRights: the number of success up, down, left, and right moves, respectively
4. In the main() function, call the updateStatistics() function after updating the player's next position.
5. In the main() function, add an if-statement to display the statistics information whenever the user inputs the key for displaying the in-game statistics information (refer to helper.h for the appropriate constant's name). The following is an example output of the statistics, where the bold red numbers are example outputs.

|  |
| --- |
| Enter command (w,a,s,d: move, q: quit): i  --- Statistics Information ---  - Moves: **4**  - Ups: **1**  - Downs: **0**  - Lefts: **2**  - Rights: **0**  Enter command (w,a,s,d: move, q: quit): |