ALGORITHMS & DATA STRUCTURES SEMESTER PROJECT

“Video game map creation simulation”

This idea of the present project is original: I came up with it, including the text of the problem and its premise.

SCARLAT Horia-Mihai

423G – ETTI – IETTI

2022 - 2023

Premise:

You are a video game developer at your favorite studio and have been tasked with creating the layout of the map in a game you’ve been working on, using names and IDs for each location. They are fed to you by another developer tasked with this, who decided that the best way to do this would be using a simple linked list. Obviously, all these locations of the level have certain names and number IDs assigned to them. They must be considered when designing the map. The only condition that you’ve been told is that the sections “Inner Chamber” and “Outer Chamber” must be the first and respectively the last locations you need to consider.

Thinking about it, you’ve decided that the best way to go about this is to assign these types of data in a tree (chain) manner in the system memory so that they can be accessed faster. Coincidentally, where they are assigned relative to their parent (either left or right) will also be their “real-time” locations in the game, relative to the first location when entering the level (i.e., “Inner Chamber”). Luckily, you don’t have any further criteria after which you need to guide yourself in terms of the position of these locations, so they are all arbitrary.

(IDs are to be randomly generated using the randomisation functions. Keep in mind that “Inner Chamber” and “Outer Chamber” must be the first and respectively last entries as they are the first and respectively last locations in the map.)

Description of the problem

The code begins by creating 3 additional files to our main one: auxArrays.h, rand.h, structPrint.h. The first important one that will be discussed is structPrint.h:

This file contains all the declarations of the nodes (both simple linked list and tree) associated pointers and variables for the ID (integer) and name (string). Additionally, a preorder print function for the tree is present, which also contains a long list of if-conditionals that have certain indentations associated with each level of the tree/chain (will improve upon in future). The print function (printTreePreorder()) will then be called recursively for the left side and respectively right side of the tree.

“rand.h” is another file which contains a copied algorithm for generating either 1 or 0 (1 having a 75% chance of generation, while 0 having a 25% percent of generation). This necessary to decide (in a random manner) whether a node will be added to the left or the right of a respective node.

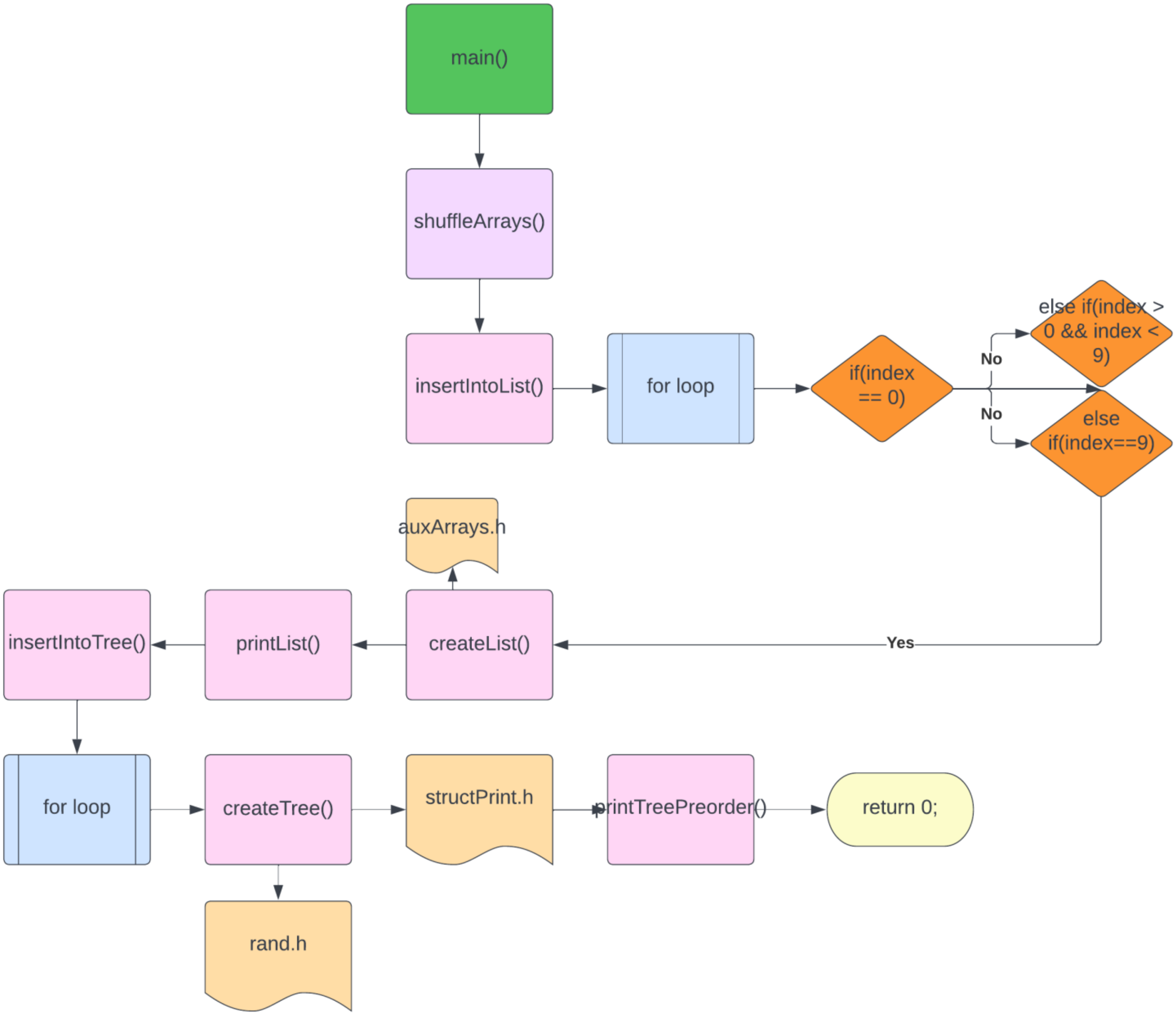
“auxArrays.h” is a file which contains three arrays which contain the names, numbers from 0 to 7 and IDs associated with each node of the data structures. They are then shuffled in a shuffleArrays() function which will make sure to maintain the random character of our problem.

The main file, “SemesterProjectSDA.cpp” then begins with the creation function for the list where data is inserted to the respective pointers. The insertIntoList() function makes sure, employing a while loop, that “inner_chamber” and “outer_chamber” are at the beginning, respectively the end of the list, to respect the problem’s request, heavily employing the use of the shuffled arrays.

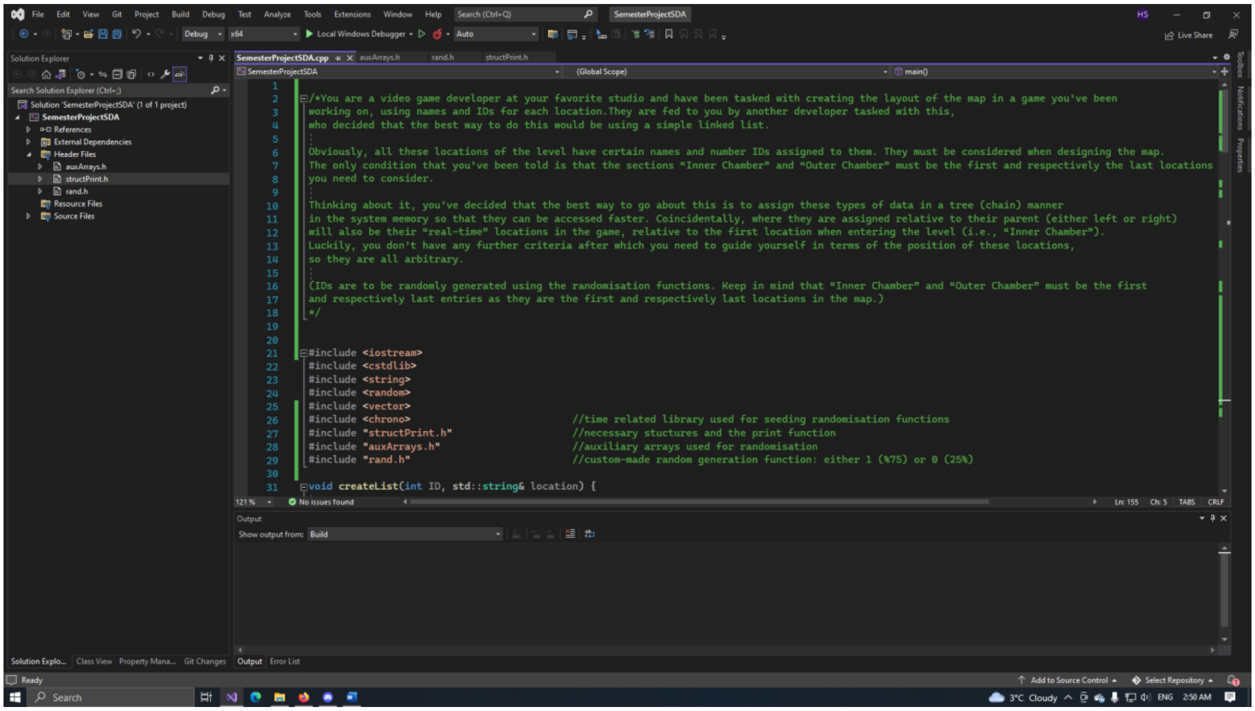
Following these, we have a printList() function which does what its tile says, recursively, moving a temp pointer variable until the end of the list. Creation of the tree (chain) happens in the createTree() function, which has parameters ID and location, which are passed in a for loop which parses the linked list. In this function, the rand.h header shines, make use of it to randomize the location of the regions of the supposed map. insertIntoTree() is the function in which the linked list is parsed in a for loop and the parameters (ID and name of location) is passed to the createTree() function.

The main file closes with the main(), which calls upon all the necessary functions, cleanly.

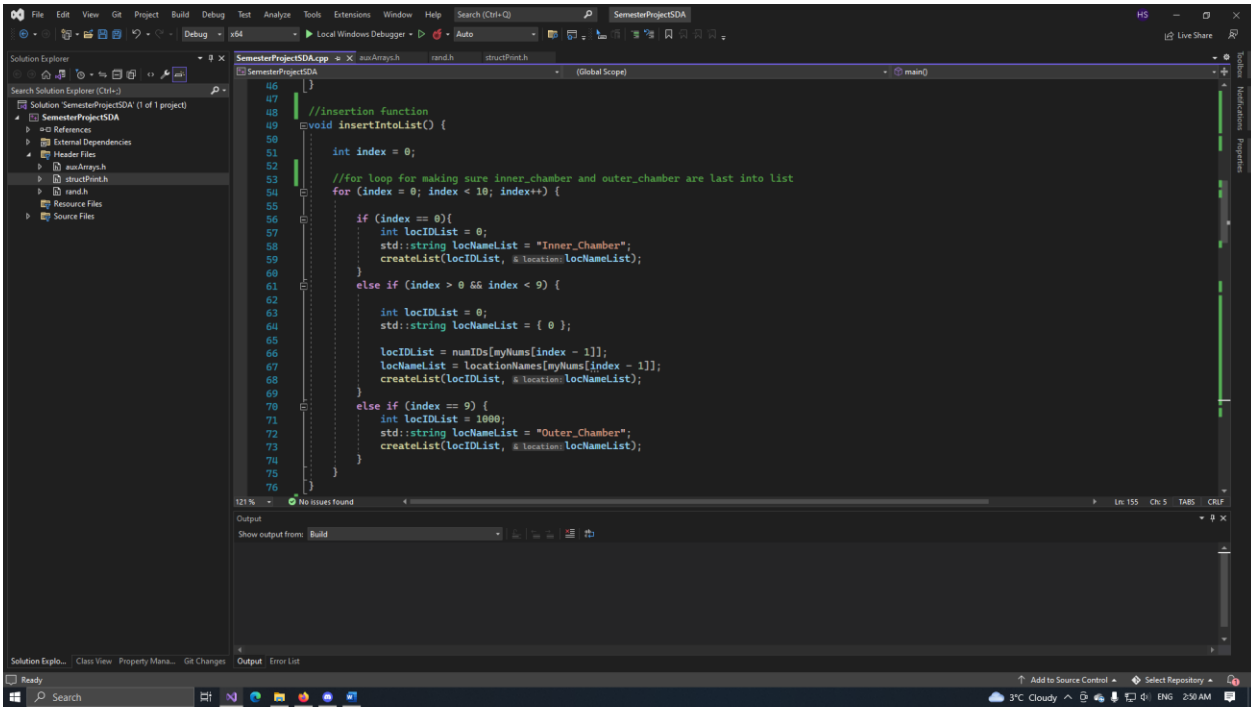
Workflow


main()
shuffleArrays()
insertIntoList()
for loop
if(index == 0)
No
else if(index > 0 && index < 9)
No
else if(index==9)
Yes
auxArrays.h
createList()
printList()
insertIntoTree()
for loop
createTree()
rand.h
structPrint.h
printTreePreorder()
return 0;

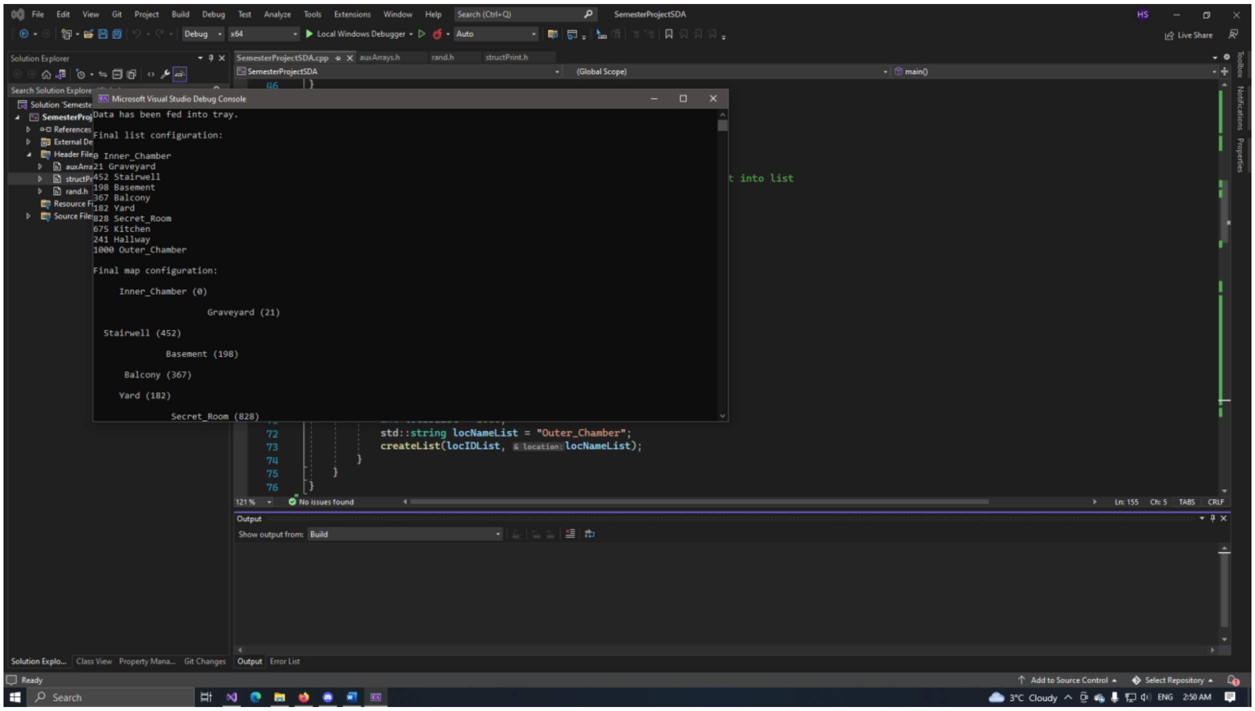
Code screenshots



```cpp
/*You are a video game developer at your favorite studio and have been tasked with creating the layout of the map in a game you've been
working on, using names and IDs for each location.They are fed to you by another developer tasked with this,
who decided that the best way to do this would be using a simple linked list.

Obviously, all these locations of the level have certain names and number IDs assigned to them. They must be considered when designing the map.
The only condition that you've been told is that the sections "Inner Chamber" and "Outer Chamber" must be the first and respectively the last locations
you need to consider.

Thinking about it, you've decided that the best way to go about this is to assign these types of data in a tree (chain) manner
in the system memory so that they can be accessed faster. Coincidentally, where they are assigned relative to their parent (either left or right)
will also be their "real-time" locations in the game, relative to the first location when entering the level (i.e., "Inner Chamber").
Luckily, you don't have any further criteria after which you need to guide yourself in terms of the position of these locations,
so they are all arbitrary.

(IDs are to be randomly generated using the randomisation functions. Keep in mind that "Inner Chamber" and "Outer Chamber" must be the first
and respectively last entries as they are the first and respectively last locations in the map.)
*/


#include <iostream>
#include <cstdlib>
#include <string>
#include <random>
#include <vector>
#include <chrono>                           //time related library used for seeding randomisation functions
#include "structPrint.h"                    //necessary stuctures and the print function
#include "auxArrays.h"                      //auxiliary arrays used for randomisation
#include "rand.h"                           //custom-made random generation function: either 1 (%75) or 0 (25%)

void createList(int ID, std::string& location) {
```

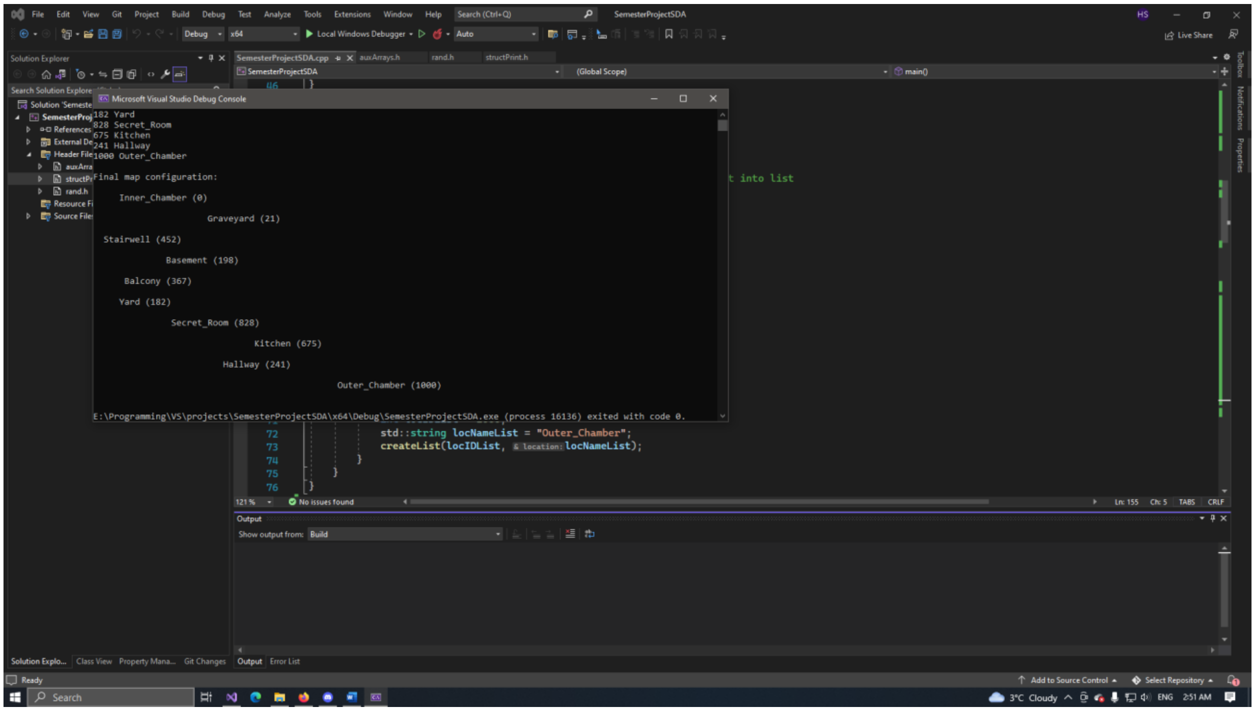


```cpp
}

//insertion function
void insertIntoList() {

    int index = 0;

    //for loop for making sure inner_chamber and outer_chamber are last into list
    for (index = 0; index < 10; index++) {

        if (index == 0){
            int locIDList = 0;
            std::string locNameList = "Inner_Chamber";
            createList(locIDList, locNameList);
        }
        else if (index > 0 && index < 9) {

            int locIDList = 0;
            std::string locNameList = { 0 };

            locIDList = numIDs[myNums[index - 1]];
            locNameList = locationNames[myNums[index - 1]];
            createList(locIDList, locNameList);
        }
        else if (index == 9) {
            int locIDList = 1000;
            std::string locNameList = "Outer_Chamber";
            createList(locIDList, locNameList);
        }
    }
}
```



```
Data has been fed into tray.

Final list configuration:

0 Inner_Chamber
21 Graveyard
452 Stairwell
198 Basement
367 Balcony
182 Yard
828 Secret_Room
675 Kitchen
241 Hallway
1000 Outer_Chamber

Final map configuration:

     Inner_Chamber (0)

                      Graveyard (21)

  Stairwell (452)

              Basement (198)

      Balcony (367)

     Yard (182)

               Secret_Room (828)
```

```
            std::string locNameList = "Outer_Chamber";
            createList(locIDList, locNameList);
        }
    }
}
```



```
182 Yard
828 Secret_Room
675 Kitchen
241 Hallway
1000 Outer_Chamber

Final map configuration:

     Inner_Chamber (0)

                      Graveyard (21)

  Stairwell (452)

              Basement (198)

      Balcony (367)

     Yard (182)

               Secret_Room (828)

                               Kitchen (675)

                         Hallway (241)

                                               Outer_Chamber (1000)

E:\Programming\VS\projects\SemesterProjectSDA\x64\Debug\SemesterProjectSDA.exe (process 16136) exited with code 0.
```

```
            std::string locNameList = "Outer_Chamber";
            createList(locIDList, locNameList);
        }
    }
}
```