Program name: Final Project A Text-based Game
Author: Xiaoying Li
Date: 8/16/2019
Description: Design Description & Test Table & Reflection of Final Project A Text-based Game

● Design

1. Main

The main function controls the entire program.

Ask user whether he/she wants to start the game or not.

If the user decided not to start the game, quit the program.

Otherwise, start the game.

Create 6 Space objects with the structure of linked space.

Set player at the start room.

Let player walk through the first 5 rooms.

After the player walks through the first 5 rooms, if the number of steps is no more than 10, let user enter the end room.

Otherwise, game over.

After the game ends, ask the user whether he/she wants to play again.

If the user decided to play again, start the game again.

Otherwise, quit the program.

Loop the game until the user decided to quit.

2. Menu

The menu function prints menu options to the screen for the user, and after the user makes a choice, verify the user's input, and return the value back to the program.

(1) startMenu:

When the project starts, ask user whether he/she wants to start the game or not.

If the user decided to start the game, return true.

Otherwise, return false.

(2) endMenu:

When the game ends, ask the user whether he/she wants to play the game again.

If the user decided to play the game again, return true.

Otherwise, return false.

3. Space

The Space class is an abstract base class, which represents the space the player can be in, will be inherited by all the rooms' own subclass.

It has following protected data members:

4 Space pointers: top, right, left, and bottom.

Room number <int>;

Player’s choice to go to which room <int>;

A bool referring to whether the room has been visited.

It has following public member functions:

Set function for 4 Space pointers;

Get function for Space pointers left and right;

Get function for room number and room option;

Function implements room’s special action (pure virtual function);

Destructor.

4. RoomStart

The RoomStart class inherits from Space class, and contains all information of the start room.

Its private data members are inherited from Space class.

It has following public member functions:

Default constructor:

initializes all data members when a RoomStart object is created.

Override action() function that takes in a vector of string as container for the player to carry items and returns nothing. It implements the special action for the player to interact with in the start room.

Destructor.

5. RoomBetween

The RoomBetween class inherits from Space class, and contains all information of rooms between the start room and end room.

Its private data members are inherited from Space class, and an array of string contains all items to collect

It has following public member functions:

Default constructor:

initializes all data members when a RoomBetween object is created.

Override action() function that takes in a vector of string as container for the player to carry items and returns nothing. It implements the special action for the player to interact with in the rooms between the start room and end room.

Destructor.

6. RoomEnd

The RoomEnd class inherits from Space class, and contains all information of the end room, which refer to ouside.

Its private data members are inherited from Space class.

It has following public member functions:

Default constructor:

initializes all data members when a RoomEnd object is created.

Override action() function that takes in a vector of string as container for the player to carry items and returns nothing. It implements the special action for the player to interact with in the end room (outside).

Destructor.

7. Validation

The validation function takes in a string and pass it by reference. The function makes sure the user's input is an integer. Afterwards, it returns the integer the user inputted.

```
Loop until the user inputs an integer.
    When a string inputted, detect whether an extraction has failed and fix it.
        If an extraction has failed, ask the user to enter again.
    If extraction succeeds, then make sure no extraneous input.
        Then detect if every character of the string is digit.
            If every character of the string is digit, then the string is an integer.
            Otherwise, ask the user to enter again.
    If the input string is an integer, transfer it to an integer, and return the integer.
```

The validation of the range of every integer returned should be validated after the integer returns to the program.

- **Test Plan**

This test plan does not include test cases for input validation, startMunu() and endMenu(), since they have been tested at Project1. This test plan only verifies if the program meets all given specifications and works well at all cases.

Test Scope	Description	Expected Results
class RoomStart	Select to start Ant Moving Game.	Correctly output game start sentence and rules of the game. Correctly execute the special action of start room: ask player to fix the light bulb. Notify the user this room has already been explored when user enters this room again, and automatically set player to second room.
class RoomBetween	Program automatically sets player to the second room after user successfully completed the special action of start room.	From room 2 to room 5, there is an item for picking up in each room. Program should ask user if he wants to pick up the item in each room. If the item in a room has been picked up, then that room will be considered as visited. When user comes into the room again, program should notify user this room has been visited, and there's nothing to pick up. If the user entered a room but didn't pick up the item, the room will still be

		considered as not visited. When user goes to that room again, program should ask if user wants to pick up the item. From room 2 to room 5, the special action is opening a door to next room or last room. If user selects to go to next room, program should set the user to next room. If user selects to go back to previous room, program should set user to previous room, and output correct statement according to if the user has picked up the item in that room.
class RoomEnd	In room 5, player selects to enter next room, and the totally opened doors less than 10 times.	Program should check if player picked up all the 4 items. If yes, then executes the special action of end room: Ask player to return items to ant. If player didn’t pick up all 4 items, program should tell user game failed, not executing the special action of end room.
step limit	Before enters the end room, user has already used all 10 chances to open doors in between rooms.	When the 11th time user intends to open a door, program should output game over, and not allow user to enter end room.
main.game()		Program should run smoothly, the transition between rooms should be executed correctly. All the requirements should be fulfilled.,

- **Reflection**

1. Design changes I have made:

First of all, in my initial design, I tried to write 6 derived classes that are derived from the Space class. Each room will have its own derived class. However, after I actually started to code, I realized this design is extremely time-consuming. Since I need to complete this program in very limited time, I decided to combine 4 between rooms to 1 class.

Secondly, the pure virtual function action() took in 3 parameters at the beginning, including an integer to record open-door times, and a bool to determine if the game is over. But I found this design is redundant, because the integer step showing the open-door limit can be fulfilled in the flow of the whole game, and the bool to determine game over is unnecessary since there is no other factors can lead to game over than reaching step limits, thus, the bool's function can be replaced by step limit. Therefore, I deleted these two parameters from action(), only kept vector of string which represents container.

Thirdly, since the option of choosing which room to enter only exists in room 2 to room 5, roomOption was a data member only in class roomBetween. But this made coding the entire game flow in main.game() very complicated, because roomOption didn't exist in start room and end room. When the program needed to use roomOption to determine set player to which room, all the situations related to start room and end room had to be listed specifically, increasing the complexity of code and may cause errors. In this case, I changed roomOption into a data member of class Space, and inherited by all the derived classes of class Space. Since start room only connected to room 2, I set the roomOption of startroom as only able to enter next room, which is 1. Similarly, end room's roomOption was set to 2. Thus, program can use roomOption to determine which room the player will be set to.

2. How I solved problems encountered:

The biggest challenge I encountered during final project was I did not understand linked structure very well, and I had a hard time connecting the classes for each room by linked structure. To solve this problem, I learned the content of week 6 on Canvas again, and reviewed my code of Lab6. Luckily I solved this problem.

After I finished coding of the whole project, there's a lot of memory leak occurred during test. Learning destructor related knowledge online helped me fixed this issue.

3. What I learned for this project:

This project helped me have a good grasp of linked structure and destructor related knowledge. In many projects and labs, I noticed my code has memory leak issue. Although I successfully fixed these problems, the process was very hard for me. I think I still lack an in-depth understanding of memory leak. In the break after this quarter, I'll use spare time to acquire more memory leak related knowledge.

Due to my personal issue, I didn't have much time for this project. Although I fulfilled the requirement of this project, the design and code is still superficial. So the importance of reasonably scheduling time is another lesson I learned from this project.