```
#include <iostream>
#include <vector>
using namespace std;


/*
The Room class is used to organize data for an individual room.
This includes the room's name and description, and exits that
lead to other rooms.
*/
class Room {
public:
    // class fields (variables)
    int id;                             // each room has a
numerical ID
    string name;                        // short name
    string description;                 // full description
    int north, south, east, west; // store the IDs of the
adjacent rooms

    // class methods (functions)
    Room(){
        // default constructor -- doesn't do anything at the
moment

    };

    Room(int id, string name, string description) {
        // alternate constructor - sets up the room all at once
        this->id = id;
        this->name = name;
        this->description = description;
    };

    void describe() {
        // describe the room
        cout << name << endl;
        cout << description << endl;
    }

};

/*
The Game class serves to organize all the information we need
for our game.
It contains a vector of Rooms (the room list), as well as the
current player
location.

Because we have an ID for each room, we can simply store
locations as ints.

To keep things more organized, we've made separate functions
```

```
for setting up
the game, and playing the game.

For your assignment, you should use this structure with your
own room names,
descriptions, and "map" of how they are organized. You should
have 5-7 rooms.

Finally, using the method discussed here, move your player
through the rooms,
and use describe() to print out the output -- giving a tour of
the game.

*/
class Game {
public:

    vector<Room> roomList;      // stores all the Room objects
    int          playerLocation;// ID of the room the player
is in

    const int    NUM_ROOMS = 4; // determines the vector size
    // we define the room IDs using const ints in order to
make the code more readable.
    const int NOWHERE       = 0;    // I just used this to
indicate when something's broken.
    const int LIVING_ROOM   = 1;
    const int DINING_ROOM   = 2;
    const int KITCHEN       = 3;

    Game() {
        // constructor
        roomList.resize(NUM_ROOMS);   // set up the vector to
be the right size
        playerLocation = LIVING_ROOM; // start the player on
the map
    }
    setup() {

        // build the rooms and connect them.
        // In English:
        // The dining room is north of the living room. (and
vice versa)
        // The kitchen is east of the dining room. (and vice
versa)

        // I build the first room the long way, with the
default constructor.
        Room livingRoom;
        livingRoom.name = "Living Room";
        livingRoom.id = LIVING_ROOM;
        livingRoom.description = "A simple living room with
carpet and a couch.";
```

```
        livingRoom.north = DINING_ROOM;
        roomList.at(LIVING_ROOM) = livingRoom;

        // for the rest of the rooms I use the constructor,
since it's quicker.
        Room diningRoom = Room(DINING_ROOM, "Dining Room",
"Dusty furniture fills the room.");
        diningRoom.south = LIVING_ROOM;
        diningRoom.east = KITCHEN;
        roomList.at(DINING_ROOM) = diningRoom;

        Room kitchen = Room(KITCHEN, "kitchen", "A small,
dimly lit kitchen.");
        kitchen.west = DINING_ROOM;
        roomList.at(KITCHEN) = kitchen;

        // last room is Room 0 - nowhere. I put this here
        // for debugging purposes. If you somehow end up in
room zero,
        // it'll be obvious it wasn't supposed to happen.
        //Room nowhere = Room(NOWHERE, "Nowhere", "It is dark.
You are lost.");
        //roomList.at(NOWHERE) = nowhere;

    }

    void tour() {
        int newLocation; // a placeholder used to update
location

        // this is the "gameplay" -- walking through the rooms
        Room currentRoom = roomList.at(playerLocation);
        currentRoom.describe();

        // move to a new room
        cout << endl << "walking north..."  << endl;
        // i want to go to the room north of here -- that ID
is stored in north
        newLocation = currentRoom.north;
        // update the player location ID, then get the new room
        playerLocation = newLocation;
        currentRoom = roomList.at(playerLocation);
        // finally, describe the room
        currentRoom.describe();

        // move to the final room
        cout << endl << "walking east..." << endl;
        newLocation = currentRoom.east;
        playerLocation = newLocation;
        currentRoom = roomList.at(playerLocation);
        currentRoom.describe();

        // testing the nowhere "fall back" code
```

```
        /*
        cout << endl << "walking north, no room here..." <<
endl;
        newLocation = currentRoom.north;
        playerLocation = newLocation;
        currentRoom = roomList.at(playerLocation);
        currentRoom.describe();
        */

    }



};


int main()
{
    // since the logic of our game is contained within the
Game class,
    // all we do in main() is get things started
    Game game;
    cout << "Setting up the game" << endl;
    game.setup();
    cout << "Starting the tour" << endl;
    game.tour();
    cout << "Game tour finished" << endl;

    return 0;
}
```