C# Hunt the Wumpus

Hunt the Wumpus is logic game in which you have to locate and kill a monster called the Wumpus. This early game served as the prototype for text-adventures of the late seventies and eighties. Version are available for your phone.

Concepts, keywords you will use...
for loops, do-while loops, break, continue

Estimated completion time: 1-2 days

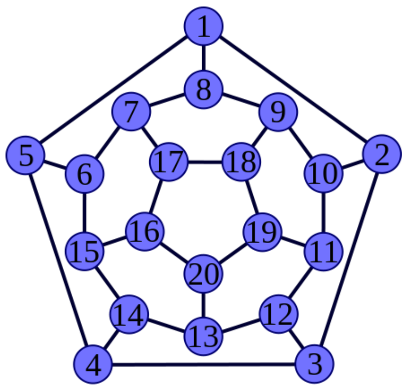
About the game

Hunt the Wumpus is a classic game that goes back to 1973 (which raises the question what computer it was written on). To goal of the game is to locate and kill a monster known as the wumpus which lives in a network of twenty caves.   To win, you must figure out which cave the wumpus is in and shoot an arrow into it. Unfortunately, you only have one arrow, so your shot needs to be right the first time. Besides the Wumpus, there are other hazards in the caves as well: pits and vampire bats. Here are the complete rules.

- If you enter the Wumpus' cave, it will kill you and eat you.
- If you shoot at the Wumpus' and miss, it will kill you and eat you.
- If you fall into a pit, you die.
- If you are next to the Wumpus, you will smell the Wumpus.
- If you are next to a bottomless pit, you will feel a draft.
- If you are next to bats, you will hear squeaks.
- If you enter a room which has bats, the bats will fly you to a different room

The Map

The image on the right shows how the rooms are hooked together. If for example, the player were in room one and felt a draft, the pit could be in rooms 2, 5, or 8. It could even be in two of those since there are two pits! Determining where things are requires moving around the rooms and using the process of elimination.



It is extremely helpful to have this map and take notes on it as you play the game.

A note about Console Applications

This project is implemented as a Console application, meaning it runs in a Cmd prompt. You print to the console, you can use the command…

Console.WriteLine("your message");

Two functions have also been provided for you to allow you to read input from the console. ReadChar() and ReadInt().

For example, to read a character you would type:

```
char ch = ReadChar();   //get a char from user, store it in ch
```

To read an integer…

```
int n = ReadInt();   //get an int from user, store it in n
```

Getting started

- There is a project folder in T:\C#\Wumpus. Copy this folder to your X: drive.

- Launch the task list from the View menu to see what you have to complete.

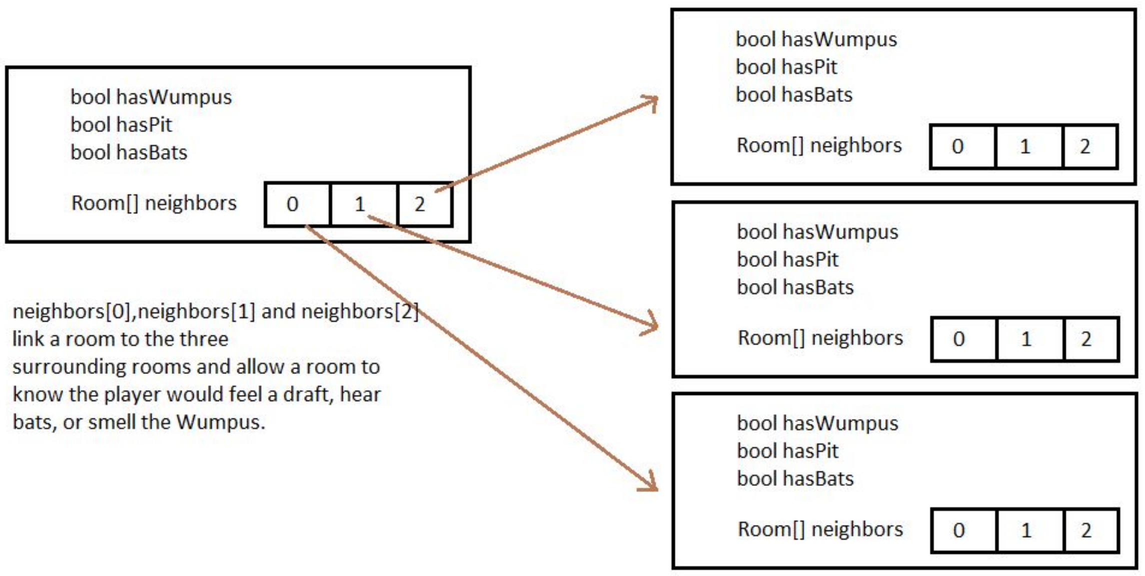
The Room class

The game space is modelled as an array of Room objects. Each Room knows whether it has a pit, bats, or the Wumpus. Each room also knows which three Rooms are adjacent to it. It stores these in an array.

You need to implement the following 3 functions.

HasDraft()	returns true if any of the neighbor’s hasPit variable is true (otherwise return false)
HearsBats()	returns true if any of the neighbor’s hasBats variable is true (otherwise return false)
CanSmellWumpus()	returns true if any of the neighbor’s hasWumpus variable is true (otherwise return false)

The diagram below shows how the rooms are connected and how you can implement these functions using the array, *neighbors.*


bool hasWumpus
bool hasPit
bool hasBats
Room[] neighbors
0
1
2
neighbors[0],neighbors[1] and neighbors[2] link a room to the three surrounding rooms and allow a room to know the player would feel a draft, hear bats, or smell the Wumpus.

The Game Class

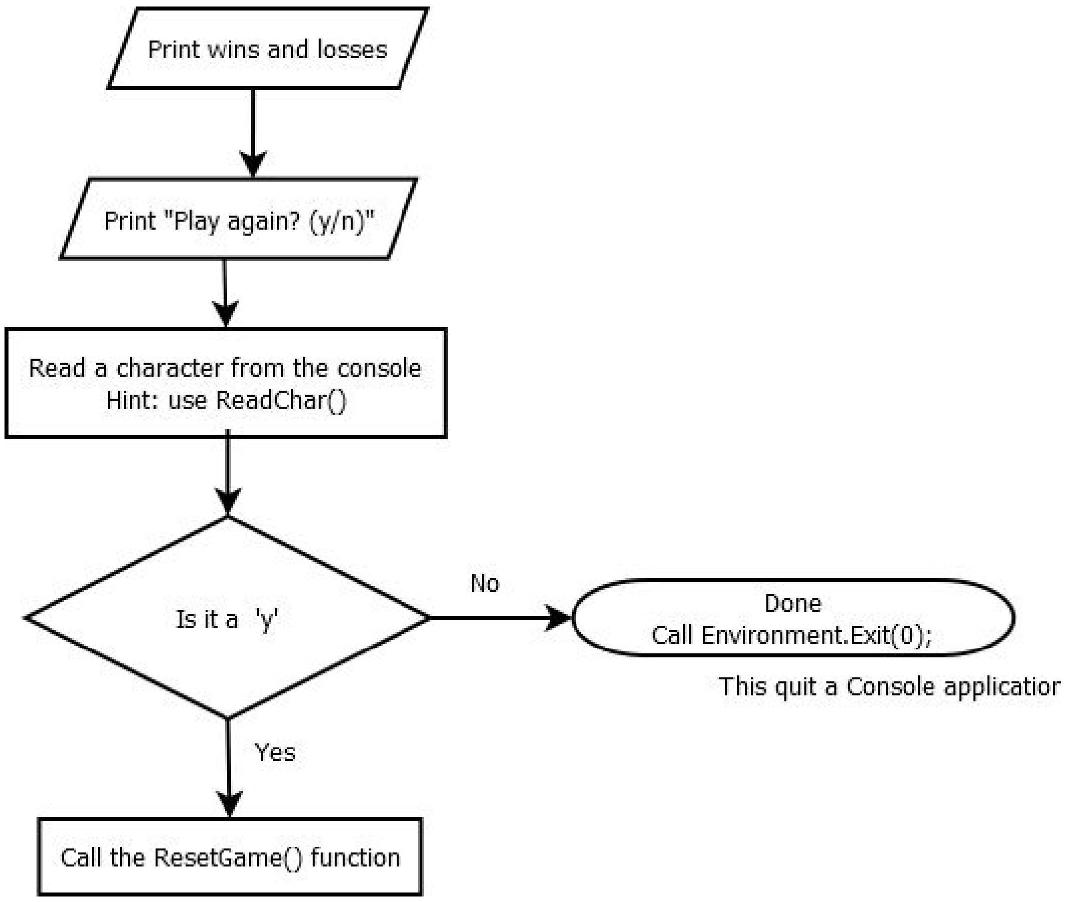
This class has been partially implemented for you. You need to complete the following functions.

> ResetGame()
> Run()
> Move(int newRoom);
> Shoot(int newRoom);
> PlayAgain();

The ResetGame() function

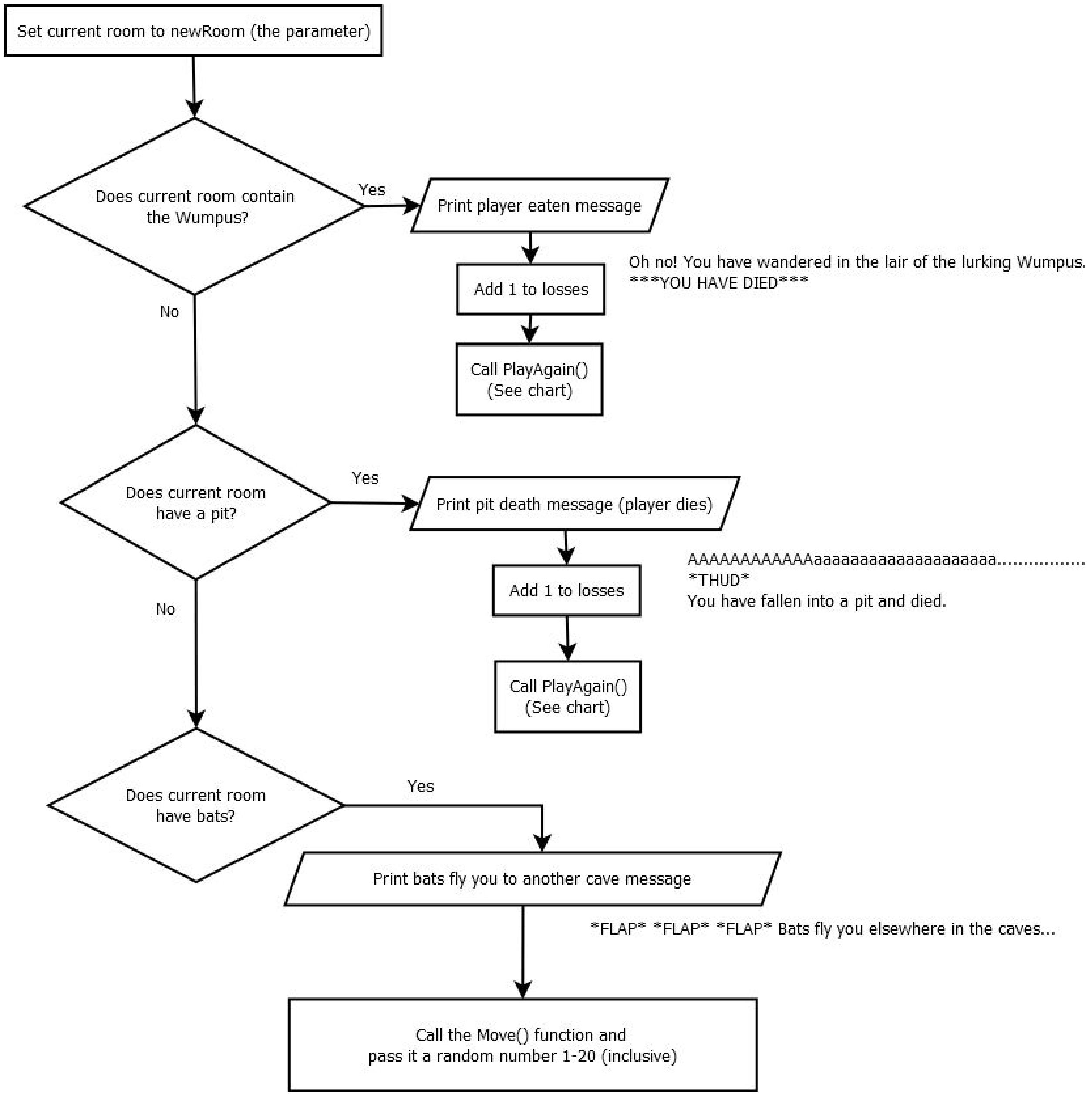
This function loops over all the rooms and calls each one's Clear function to unset all the public variables to false (hasPit, hasBats, hasWumpus). Then it randomly places two pits, two bats, and one wumpus. The comments in the code will guide you through what to do.

The PlayAgain() function


Print wins and losses
Print "Play again? (y/n)"
Read a character from the console
Hint: use ReadChar()
Is it a 'y'
No
Done
Call Environment.Exit(0);
This quit a Console applicatior
Yes
Call the ResetGame() function

The Move() function

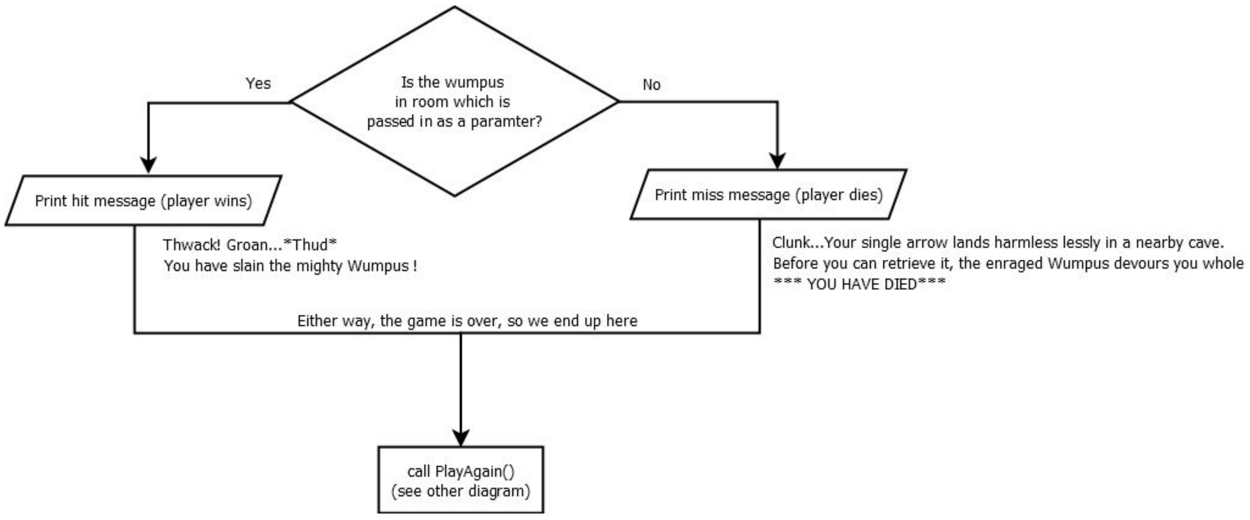
The Move() function takes a parameter indicating what room to move into. It checks to see if the player is eaten by the Wumpus, falls into a pit, or is carried elsewhere by bats. If bats carry the player elsewhere, the Move() function will be called again using a random number as the new room.


Set current room to newRoom (the parameter)
Does current room contain the Wumpus?
Yes
Print player eaten message
Add 1 to losses
Call PlayAgain() (See chart)
Oh no! You have wandered in the lair of the lurking Wumpus.
YOU HAVE DIED
No
Does current room have a pit?
Yes
Print pit death message (player dies)
Add 1 to losses
Call PlayAgain() (See chart)
AAAAAAAAAAAAaaaaaaaaaaaaaaaaaaa................
THUD
You have fallen into a pit and died.
No
Does current room have bats?
Yes
Print bats fly you to another cave message
FLAP *FLAP* *FLAP* Bats fly you elsewhere in the caves...
Call the Move() function and pass it a random number 1-20 (inclusive)

The Shoot() function

The shoot function needs to ask the player for a room, validate the room is in fact one of the neighboring rooms, then check to see if the wumpus is in that room. If it, is the player wins, otherwise the player dies.

Here is a flow chart showing what the function needs to do...

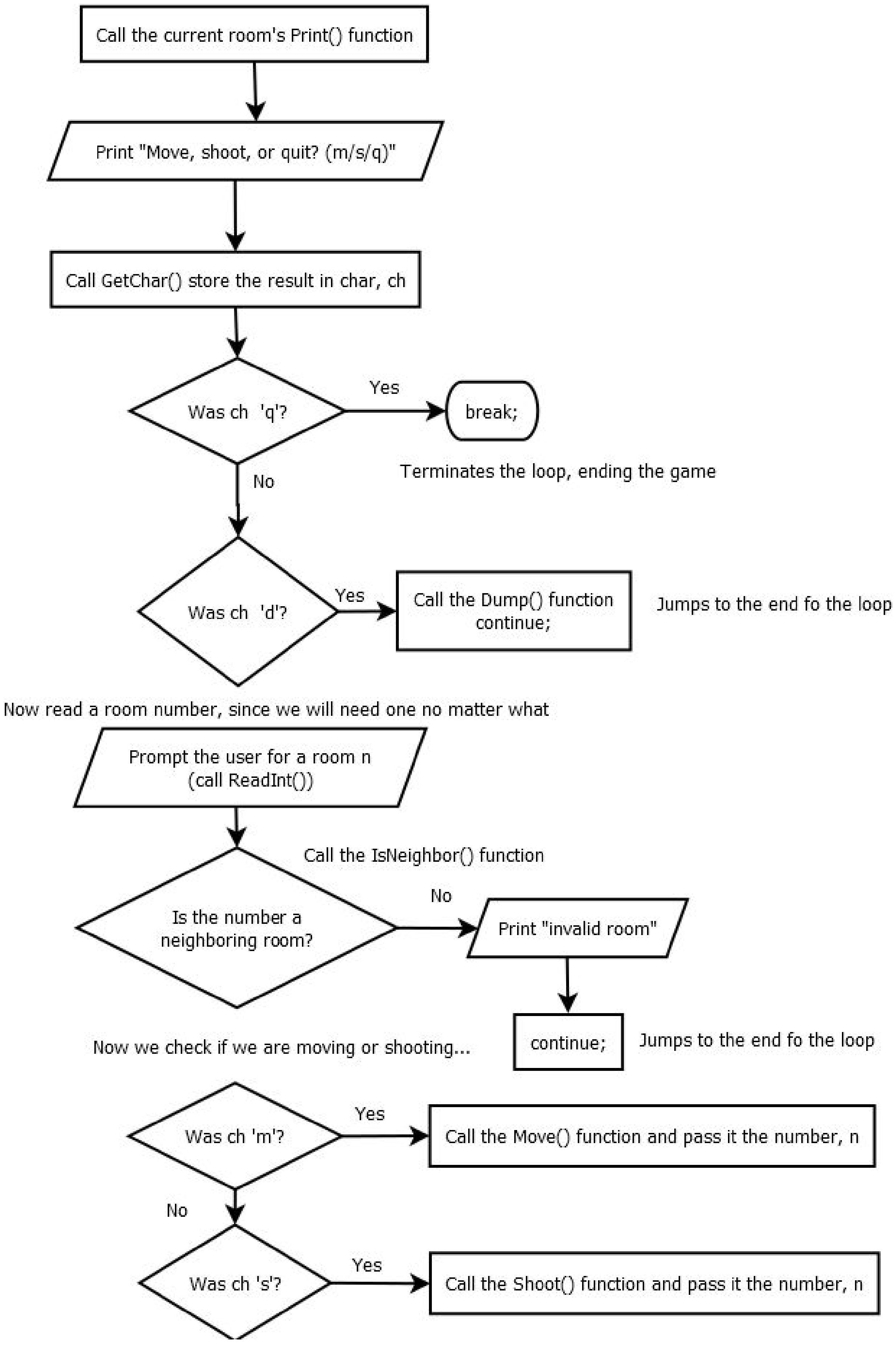

Is the wumpus in room which is passed in as a paramter?
Yes
No
Print hit message (player wins)
Thwack! Groan...*Thud*
You have slain the mighty Wumpus !
Print miss message (player dies)
Clunk...Your single arrow lands harmless lessly in a nearby cave.
Before you can retrieve it, the enraged Wumpus devours you whole
*** YOU HAVE DIED***
Either way, the game is over, so we end up here
call PlayAgain()
(see other diagram)

The Run() function

Once you have the Move() and Shoot() functions. You can implement the Run() function of the Game class. This function ties everything together. There is a built-in cheat. If your enter ‘d’, you should call the Dump() function, which prints out the contents of the rooms.

Note: all of this code is inside a while (true) loop so the user is repeatedly prompted for input

Put all of this inside a while(true) loop


Call the current room's Print() function
Print "Move, shoot, or quit? (m/s/q)"
Call GetChar() store the result in char, ch
Was ch 'q'?
Yes
break;
Terminates the loop, ending the game
No
Was ch 'd'?
Yes
Call the Dump() function
continue;
Jumps to the end fo the loop
Now read a room number, since we will need one no matter what
Prompt the user for a room n
(call ReadInt())
Call the IsNeighbor() function
Is the number a neighboring room?
No
Print "invalid room"
continue;
Jumps to the end fo the loop
Now we check if we are moving or shooting...
Was ch 'm'?
Yes
Call the Move() function and pass it the number, n
No
Was ch 's'?
Yes
Call the Shoot() function and pass it the number, n

Enjoy playing your game.

Variations

Crooked arrow variation
If the player fires into a room that doesn't have the Wumpus, use a while loop to keep moving the arrow into a neighboring room until it hits the player or Wumpus. If the arrow hits the player, print...

You fall to the ground, struck by your own arrow. As you lie helpless, the Wumpus rushes to your side, not to help you, but to eat you!
****YOU HAVE DIED****

Moving wumpus variation
If the player fires into a room that doesn't have the Wumpus, move the wumpus to new room (print that the startled wumpus has moved). If the new room happens to be the player's room, the player is eaten.