The Game of 24

Course: Computer Programming (EE3031)
Written Report of Homework 2
高泊天 (B09209023)

A. Introduction -- Reasons for making this version

In Homework 1, I have written a program for the game of 24. However it was an incomplete version with a lot of missing features. Therefore, I have decided to continue the work on making it a complete game. Also, as I have planned previously, I wanted to work on the advanced version of the Game of 24, which was also played between me and my classmates back in my junior and senior school years. I am naming the new advanced version the Game of X.

B. Game rules -- How does the Game of X work?

In this new mode, the rules are basically the same as the Game of 24 where players use '+,−,×,÷' for calculation and each number can only be used once. However, the target is not limited to 24 only, but it can be any integer between [24,99]. This increases the difficulty of the game greatly and gives a larger variety and more changes to the previous game. For example, if the initial numbers are '3,5,12,13' and the given target number is '50', the solution will be $5 \times 13 - 12 - 3 = 60$.

C. Updates -- Improvements made in this version

In this version, I have added a stopwatch from System.Diagnostics to record the players' used time to finish solving the required number of questions. The timer starts once the player selects the game mode and stops when the player finishes entering the last operation of the last question. This is useful to determine whether the player performed well in that run. Also, players can compete with each other in order to determine who is the better player.

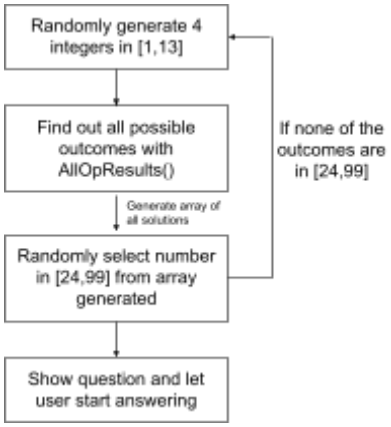
In terms of entering the answer, several improvements have been made as well. First, players can enter each operation equation at once. For example, if the player wants to add '3' and '5', they can enter '3,+,5' at once, with ',' between numbers and operation signs. This allows players to think more before pressing enter compared to the previous system where players have to press enter after every number/character.

Second, players can enter ‘r’ to reset the question. Considering that the new mode may be too difficult for many players, I have added a feature so that when players face questions that they cannot solve after a long period of time, they can get a new question. However, they will get a penalty of 10 seconds, which will be shown at the end of the game.

In terms of the methods of determining valid question sets, I have re-written a completed set of possible outcomes. Considering that each operation requires 2 numbers and each of them may perform 6 different operations (a+b, a-b, b-a, a*b, a/b, b/a). Then we can ensure that all possibilities are considered.

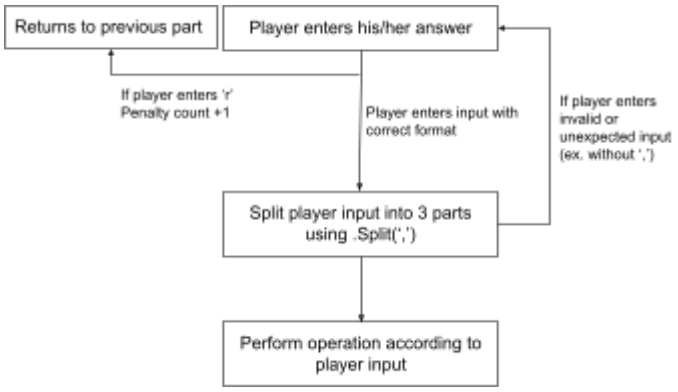
D. Coding -- How does the program make them work (with Flow Chart)

1. Selecting the target number



The method used here can ensure that each question has an answer. AllOpResults() allows the program to obtain an array including all possible outcomes by just entering the array of 4 integers. The program performs 142 different equations and 24 arrangements of the 4 integers. Each of them are stored in an array which is returned to the main function.

2. Allowing user to reset question and new input method



Here I used .Split() to split the player’s input into 2 int types and 1 char type. First, determine if the player enters ‘r’, if yes, return to the previous step. Then, split the input into an array with 3 elements. If the resulting array has more or less than 3 elements, it will be considered as an invalid input and the player will be sent back to the step of input to avoid error. For penalty count, an integer pen is used to record the number of reset the player did.

3. Stopwatch and penalty time

I used System.Diagnostics which included a class for stopwatch for timing. Stopwatch starts once the user enters ‘2’ or ‘3’ when choosing gamemode and ends when the loop ends, which is when the player finishes the last question correctly. Then, the screen shows the used time and total penalty time (10*pen). The player then can use it to compete with others.

E. Possible modifications -- What can be improved in the future

There are still several spots for improvements in this updated version. Also, I would like to add a few features to make it look like an actual app game.

First, re-writing the original 64 equations into AllOpResults() reduces the length of the program by a lot without needing to include 2 big paragraphs of exact same code. However, for others it may still be hard to quote for other usage as the equations are typed out one by one. It may be re-written into loops to consider each operation possibilities in all 3 steps.

Moreover, the method of judging valid inputs is still not complete. Considering that the user might still enter inputs like ‘a,+,b’ where a and b are letters and cannot be converted into integers, the game will still crash. However, it is still a rare specific case, unless the player intends to break the game, if not it should not happen very often.

Finally, to make it look like an actual game, I would like to add some graphics to the game. In order to make it similar to the actual game played with poker, I might add some techniques from Object-Orientation. Ultimately allowing players to select cards or drag them to perform operations.

F. Conclusion

This current version provides basically a complete gameplay experience for players. Even though the difficulty might be a bit high for starters, the new ‘reset’ feature provides an alternative method and strategy for them. The visual features can also be added if graphics are made as well.

G. References

1. Stopwatch Class (2021) from: https://docs.microsoft.com/en-us/dotnet/api/system.diagnostics.stopwatch?view=net-5.0

2. String.Split Method (2021) from: https://docs.microsoft.com/en-us/dotnet/api/system.string.split?view=net-5.0