Pass the Pigs

Mignot Mesele

January 1, 2024

Purpose

The purpose of this program is to simulate a simplified version of the game pass the pig. Players go through a series of rotations in which the program it simulates rolling the pig and gives the player the current roll which has an outcome of sides, trotters. If the roll is anything other than the sides of the pig then the program continues to "roll the pig", otherwise the current player's turn is over and it moves to the next person. This also applies to when it rolls a side after continuing to roll other outcomes. The first user to get 100 points ends the program indicating the winner.

How to Use the Program

To use the program you'd have to use make pig to create an exceutable version of pig.c. To run it on the terminal or command line, you'd run the line "./pig". To get the file "pig" you need to have the file, Makefiles, which gives you tools to make pig.c into an executable. You also need to have the file "names.h" since it contains the names of users in an array and defines the max amount of players which I use in pig.c. when you're finish, make sure the use make clean to remove the files that the make command creates.

Program Design

The program starts with the user input prompt using scanf to read the input. If the input is anything other than 2-10, then it gives an error message and uses 2 instead. Then it prompts the seed to be used with srandom later in the program. It also checks if the input is anything other than an integer and give an error message if true, using 2023. After getting the amount of users along with including names.h, an array is created with names of the players in names.h. Next, an enumeration is created with all the names of the Positions and uses typedef with the enumeration to create the positions data type. This creates constants under the names of each position and they have the data type positions. Next, an array is made with the Positions data type to put all the positions in an array to be able to make checking what position was rolled by the user easier. Then a passing pig algorithm that consists of looping through players and their turns until someone rolls a side position and until someone wins. I'll have functions that utilizes random and srandom to get a value from 0-6 to be used as an index for the positions array, returning the position and put into another function that returns the points earned and is given to the player. Lastly the program prints the winner if the winner is found by creating a condition in the inner loop that checks if the current player's score is 100 or greater and assigning a value to a variable, which is checked on the outer loop and breaks if it satisfies the condition and ends the program after printing the winner prompt.

Data Structures

Array is used in this program to keep track of all the names which makes it easier to distribute points and check who past 100 points. To shuffle through a specific name, the index would be given from a separate function that loops through users.

Enum is used to create the 5 of the 7 outcomes because of the probability of 2/7 that side and jowler has. This makes the sides into constants so they don't get modified in any way. the enumeration is also used

with typedef to put each constant under the type "position". They're then put into another array to get access to each position when running the program.

Algorithms

```
pass the pig algorithm
    infintely loops from 0 to the amount of players-1{
        loops infinitely until the current player gets a side position{
            if the position checker function returns the value of a side position then{
                break out the second loop which ends turn
                }
            else {
                check what position it is and increment current player's points
            }
            if the current player's points exceed or equal 100 then{
                assign the winner of the game
            }
        if the current player is the winner{
            break from the main loop
        }
    }
```

Function Descriptions

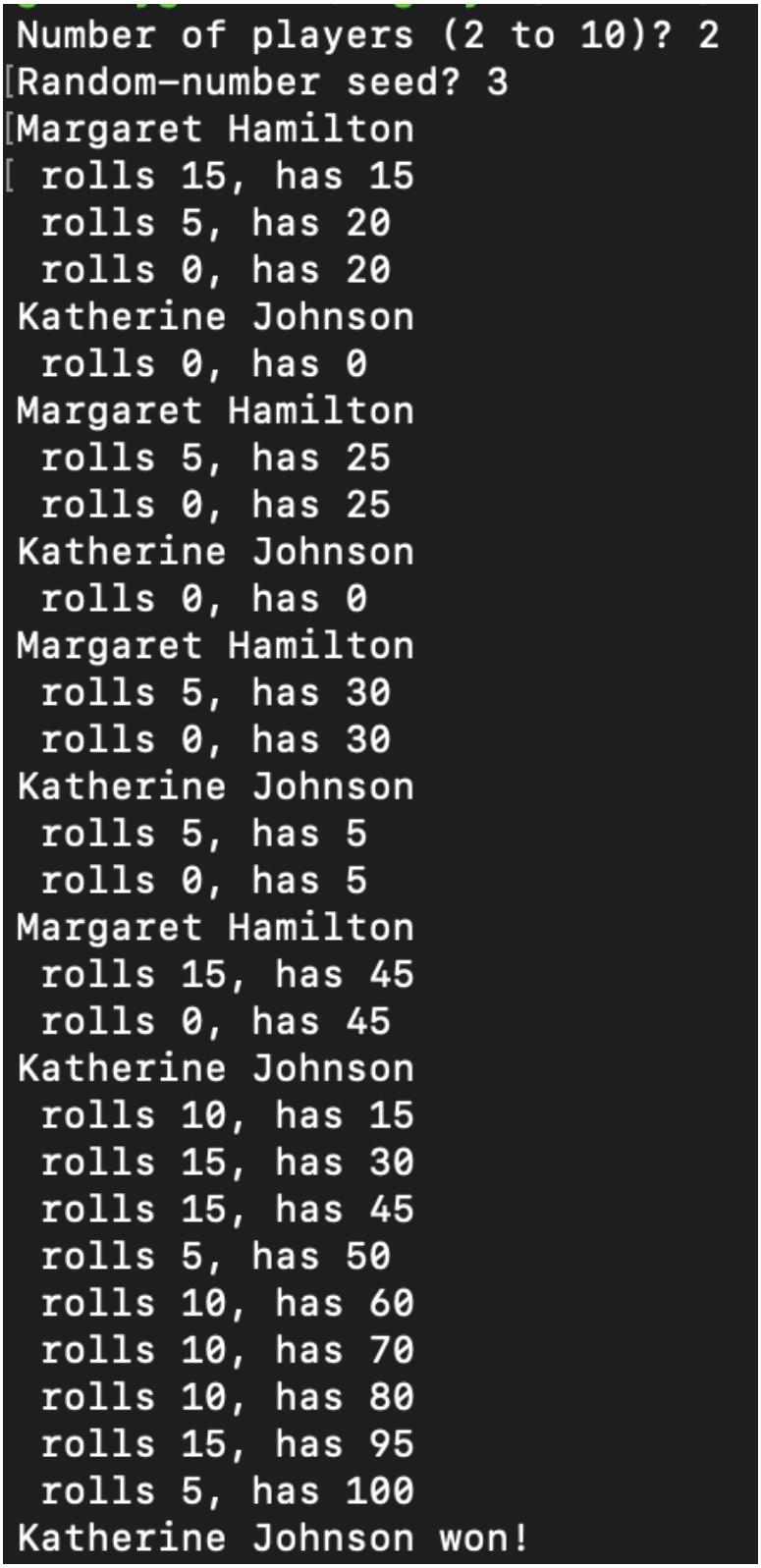
Main:
Main function is where the main implementation of the program happens in which the code starts and ends from. The inputs for main is the amount of players through the prompt given for the user to answer. The output of main is the winner prompt giving the name and amount of the player who reaches to 100 points first.

roll the pig: This function is used to create a roll from 0 to 6 to indicate which position that the user rolled. It doesn't use any inputs from main. It declares an int variable named roll and is assigned the output of generating a random number mod 7, getting a number from 0 to 6. The function returns roll.

position checker: this function is used to get the amount of points the user earns based on what position the user rolled. This function has an input(positions) which was created in main to hold the position value, from enum assigning each position a value starting from 0, from using the outcome of roll the pig as an index for the pig array that has all the positions. It takes the position value and is put through conditions checking if it equaled a certain value which then returns a certain amount of points. For example if positions was 2, then it would fulfill the else if (positions == 2) statement and return 10. This is then used to update players' scores.

Results

It ran as expected. The program asked the user to input the amount of players and if they chose anything other than 2-10 then it would default at 2. It then prompts the seed for the PRNG and if they put anything other than a number, then it defaults at 2023. After that the program went through every person and repeated, printing what they rolled and their current score was, until someone won. Example of the results is shown in the picture below with the inputs 2 and 3.



Some of errors that I ran into were the total points of the first player being added with the points from other players rolls, which was fixed by changing the declaration of the array that contains the scores from "player score board[]=" to player score board[MAX PLAYERS] since the code was creating pointers and

not actual values,and the other error was that the program was ending after one loop and declaring player zero as the winner, even though they had 25 points, which was fixed by deleting winner=1 that was after player=zero(the reason that the program loops through players) because this would decide that the current player, which would be player zero as the winner.