Assignment 1 - Pass the Pigs Design

Sean Carlyle

September 27 2021

1 Introduction

This is my design for the pass the pigs game in C. It chooses to abstract much of the repeated code since leaving it un-abstracted would make it very hard to understand. The only inputs by the user are the seed and number of players. The seed should be an unsigned integer and the number of players should be between two and ten. The expected output is a game of Pass the Pig where a player rolls a pig in different positions to gain points. Each player has their own turn rolling different positions, a players turn only stops when they roll the pig on the side or win. Rolling on the side gives zero points, upright gives ten points, snout gives fifteen points and ear gives five points. The first player to a hundred points wins the game.

2 Pseudocode with explanations

The code starts with **main** which asks how many players are playing and then checks whether the user inputted a number between two and ten. If they input the amount of players correctly, the program just calls the get seed function and grabs the seed. Then it calls the game function, which takes the seed and player input then plays the game. If the user inputs something that's not between two and ten, it sends a error message and sets the player input to two. It then gets the seed and calls the game function just like before. The reason I decided to make getting the seed and playing the game functions is mainly for readability. We are using almost the exact same code between

the if/else statement that separates whether the user inputted a valid user number, so it makes sense to just abstract these processes. It makes it so the code is less cluttered.

Include any header files

Declare function **game** and **get seed**

Declare **main**

Declare variables

Ask how many players?

Get fixed-width integer input

If the input is less than two or greater than ten

Print invalid message and make input two

Get the seed from **get seed** and call **game** function with input two and seed

Else

Get the seed from **get seed** and call **game** function with user input and seed

Make sure to return something

The **get seed** function is pretty simple and just checks whether an inputted seed by the user is an unsigned integer. If the seed is an unsigned integer, it returns the seed. If the seed is not an unsigned integer, **get seed** sets the seed to two thousand twenty one and sends an error message. After, it returns the seed.

Declare function **get seed** which takes no input

Declare variables

Ask user for random seed

Get fixed-width integer seed

If seed is not an unsigned integer

Print invalid seed and make seed two-thousand twenty-one

Return seed

Else

Return user inputted seed

The **game** function is the function that actually prints out and plays the pass the pig game. It starts off by enumerating constants to each position that the pig can roll. Then we create an array with each element in the array corresponding to a pig position, some positions are assigned to more than one element depending on how likely it is to roll them in the game. Side has a two out of seven chance and the same with ear. Every other position has a one out of seven chance. We also set the seed for random here and make a score array for each player. I thought that this was the best way to keep the score because we will be iterating through the players and if the score is an array with the same length each iteration can correspond to the same players score. So, we start a loop that doesn't end until the game ends. In this loop we again loop this time iterating through every player. This way we can go through every players turn, and if the game hasn't ended it will restart at the first players turn. To exit the player iteration loop we have to ask if the game is marked done after every turn. We start the first players turn, then create a loop that only ends when the player either wins or rolls the pig on the side. So we roll a number between zero and six since the probability of rolling each side adds up to seven, hence why our pig array has seven elements. We then check which roll the player rolled by comparing the element of the roll in the pig array to the enumerated possible rolls. For example, if we rolled a two we check what two is in the pig array. We do this by comparing the second element in the pig array with the enumerated positions like side or back since we enumerated each possible pig position with numbers zero to four. We add the score for that position to the total and keep on going until the player either hits a side or wins. If the player wins then their score and name is copied down and printed by the game function once exiting all loops. This design is better than rolling a number between one and seven because it only requires to check whether the number in the pig array at that roll value is equal to a number zero to four since their are only five total pig positions and some repeat in the array. It is also better for readability since it makes the code compare positions to positions rather than arbitrary numbers. In the other design we would have to compare what was rolled to numbers zero through six to find out what was rolled.

Declare function **game** which takes input as players and seed

Assign each position the pig can fall in to numbers zero through four

Declare a pig array with each slot equal to a position the pig can fall

Set seed for random

Declare helper variables

Create an array the size of amount of players called score and fill it with zero's

While the game hasn't ended

    Loop through every players turn

        If game is marked as done then exit loop

        Print out who is rolling currently

        While player still rolling

            Roll a number between zero and six

            With the pig array, use the random number to see which way the pig fell

            Print out the specific text of which position their pig landed

            Add the specific amount of points to the players score

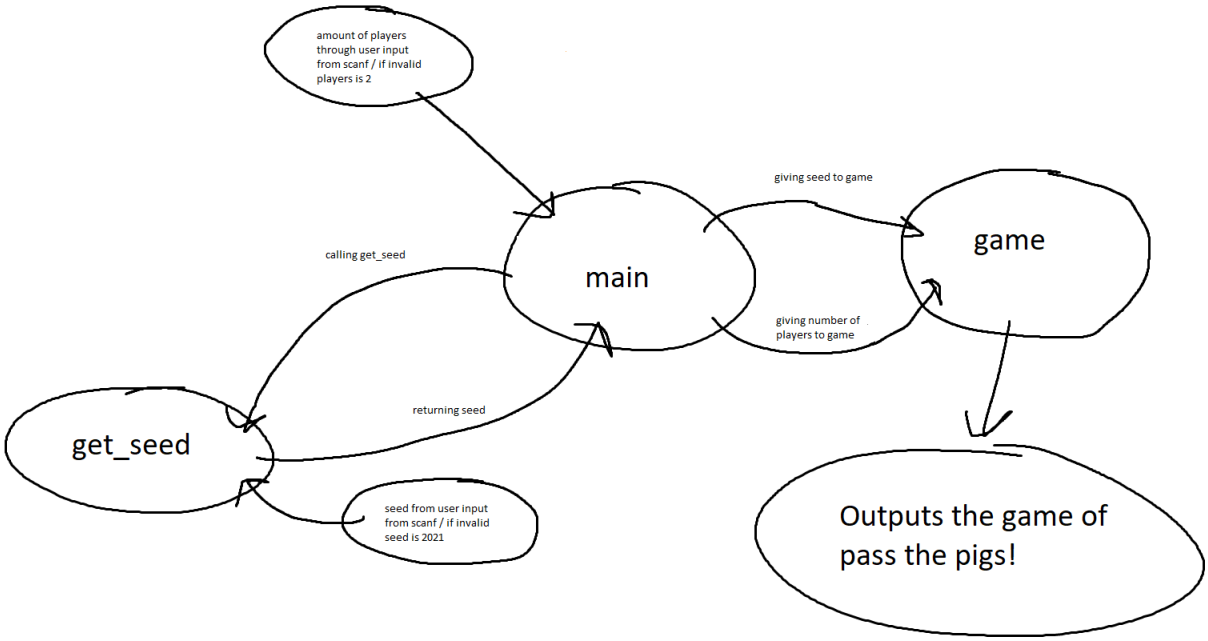
            If they land on side, break out of the while loop and start the next players turn

            If the players score goes over one hundred, save them and their score and exit

Print out who won the game and by how many points

Make sure to return something to main

3 A diagram of how program works



4 Error Handling

The error handling for my program is quite simple. For the amount of players input, if scanf fails to successfully match and assign any input values then an invalid message pops up and sets the amount of players to two. The same is with the seed input, if scanf doesn't recognize any successful integer inputs it reads an error message. Then it sets the seed to a default of two thousand twenty one and keeps running the program. The one problem with the error handling is due to a problem with scanf. If a player inputs a non-integer value for number of players, then the next scanf will not be read due to the previous one having filled the input line with non-integers. So in this case, it will default both player and seed to their default values. This could be a problem but the example code answer doesn't address this issue either. It also doesn't stop the program from running so it is not a huge error.