Assignment 1 – Pass the Pigs

Vanessa Huynh

CSE 13S – Fall 2023

Purpose

The purpose of this assignment is to design a simple version of the Pass the Pig game in C programming language. The primary objective of this game is to simulate a game for n players, such that $2 \leq n \leq 10$. Each player takes turns to play and roll a dice and earn points according to the positions that the pig lands on. Each position has a set score and Next Action associated with it. When a player reaches 100 scores or higher, declare the winner or "won" as a congratulatory message to end the game. If not, continue with the next player until there is a winner. This game will include some basic programming facilities such as handling user input, primitive types, arrays, loops, and conditionals.

How to Use the Program

Firstly, copy the names.h file and Makefile in the resources file to the asgn1 file in the cse13s directory. This is for our pig.c file (where we designed the program) to access player names in names.h file and to access the "make" command that is created in Makefile.

After designing the program in pig.c, we want to compile it by typing "make" in the asgn1 working directory, and run. "make" will run all of the compilation steps needed for this pig.c to work. This Makefile is powerful as it allows us to process the program more conveniently than writing a lengthy compilation command line. If we have something like this, we don't need to worry about mistyping as a typo of a letter can ruin the whole program. Makefile is necessary especially when having a complex program that needs to manage more dependencies and take care of format or clean the project file.

Here is the Makefile that contains the compilation type, the command line that checks for all of the flags, other necessary files and their dependencies, and how to format needed file to design this program:

```
CC = clang
CFLAGS = -Wall -Wextra -Wstrict-prototypes -Werror -pedantic
.PHONY: all clean format
all: pig

pig: pig.o
	$(CC) $(CFLAGS) $< -o $@
pig.o: pig.c names.h
	$(CC) $(CFLAGS) -c pig.c
format:
	clang-format -i -style=file *.[ch]
clean:
	rm -f pig *.o
```

After compiling, run "./pig" to test out the program in pig.c file. Then create input1.txt file that contains the number of player and number of seed. After that, run "./pig <input1.txt >output1.txt" to take the input in input1.txt to run in pig.c, get the output and store in output1.txt. We do the same thing but running the

input1.txt with pig_ref, a reference program to get the expected output by running "./pig_ref <input1.txt >expect1.txt". Finally, compare the output of our program with the expected output by running "diff expect1.txt output1.txt".

Program Design

Data Structures

- enum: or enumeration is used for defining the position of the pigs including SIDE, RAZORBACK, TROTTER, SNOUTER, and JOWLER. When defining these positions in the enum, when we call the position index, we can access the name of that index, which is also the position name.
- const Position pig[7]: This array stores the possible outcomes of a pig roll. The array assigns the position of a pig to each value of a pseudorandom number from 0 to 6.
- const char *player_name[]: This array stores the players' names.
- const char *player_names[MAX_PLAYERS]: This array stores the names of the players. It has a size of MAX_PLAYERS and the maximum number of players depends on the size of the names.h file.
- int player_scores[MAX_PLAYERS]: This array stores the scores of the players. It has the same size as the player_names array, allowing us to associate each score with a specific player's name.

Algorithms

Here are some crucial algorithms for this game:

Validate user input for the number of players and the number of random seeds. The pseudocode for checking valid seeds is similar to this.

```
prompt user to input valid num_player
    store the input value
check if input is valid or not{
        fprintf(stderr, "error message");
        set default num_players if not valid
```

Pseudocode for making an array to store the player score associated with each player:

```
Declare an array of strings called player_names with a size of MAX_PLAYERS
Declare an array of integers called player_scores with a size of MAX_PLAYERS

For each player from 0 to (num_players - 1):
    Set the name of the player from the predefined player_name array as player_names[player].
    Initialize the score of the player to 0 in player_scores[player]
```

Here is the pseudocode for the game loop:

```
Set the winner to -1

While winner is -1:
    For each player from 0 to num_players-1:
        Print the name of the current player from player_name array
        Generate a random roll between 0 and 6 and store it in roll
        Get the pig's position using pig[roll] and store it in a position variable
```

```
        Update the current player's score by adding a new score to current player_scores[player]
        print out "rolls X, has Y" after every roll

        Check if the current player's score is greater than or equal to 100:
            Set winner to the current player
            Break out of the loop

        Check if the pig's position is SIDE:
            Exit the loop

    If player is not -1:
        Exit the game loop.
```

Function Descriptions

There is only one main() function that stores all of the code to run this program. It includes defined variables, condition statements to check for input validation, an array to store players' names and scores, a while loop to design the game loop, and several if statements to check for winner, position, and exit the game.

Results

My program works as expected. The prompts for user to input num_players and number of random seeds take in input values if they are valid. If not, the program takes the default values of num_players = 2, or 2023 seeds to run the program if either of the input is not valid.

When entering the game loop, the game loop is able to print out the name of the current player and the result "rolls X, has Y" after every roll. Further, the score from previous turns is added when the player gets a new turn. It keeps adding the score and prints out the progress of the current player until either the player wins, or the pig lands on SIDE to make a new turn for the next player.

Once the winner is found, the program declares "<winner name>won!", then exits the game loop. The whole program ends.

References

[1] "Simple Do While Loop Using While(True);" Stack Overflow, stackoverflow.com/questions/14934507/simple-do-while-loop-using-whiletrue.

[2] "What's the Point of Using 'While (True) ...'?" Stack Overflow, stackoverflow.com/questions/3947055/whats-the-point-of-using-while-true.

[3] GeeksforGeeks. "C Arrays." GeeksforGeeks, Aug. 2023, www.geeksforgeeks.org/c-arrays.

The last page contains the comparison of outpu1.txt and expect1.txt of Results section.

```
vanessah@vaness:~/cse13s/asgn1$ cat output1.txt
Number of players (2 to 10)? Random-number seed? Margaret Hamilton
 rolls 15, has 15
 rolls 5, has 20
 rolls 0, has 20
Katherine Johnson
 rolls 0, has 0
Margaret Hamilton
 rolls 5, has 25
 rolls 0, has 25
Katherine Johnson
 rolls 0, has 0
Margaret Hamilton
 rolls 5, has 30
 rolls 0, has 30
Katherine Johnson
 rolls 5, has 5
 rolls 0, has 5
Margaret Hamilton
 rolls 15, has 45
 rolls 0, has 45
Katherine Johnson
 rolls 10, has 15
 rolls 15, has 30
 rolls 15, has 45
 rolls 5, has 50
 rolls 10, has 60
 rolls 10, has 70
 rolls 10, has 80
 rolls 15, has 95
 rolls 5, has 100
Katherine Johnson won!
```

(a) outpu1.txt

```
vanessah@vaness:~/cse13s/asgn1$ cat expect1.txt
Number of players (2 to 10)? Random-number seed? Margaret Hamilton
 rolls 15, has 15
 rolls 5, has 20
 rolls 0, has 20
Katherine Johnson
 rolls 0, has 0
Margaret Hamilton
 rolls 5, has 25
 rolls 0, has 25
Katherine Johnson
 rolls 0, has 0
Margaret Hamilton
 rolls 5, has 30
 rolls 0, has 30
Katherine Johnson
 rolls 5, has 5
 rolls 0, has 5
Margaret Hamilton
 rolls 15, has 45
 rolls 0, has 45
Katherine Johnson
 rolls 10, has 15
 rolls 15, has 30
 rolls 15, has 45
 rolls 5, has 50
 rolls 10, has 60
 rolls 10, has 70
 rolls 10, has 80
 rolls 15, has 95
 rolls 5, has 100
Katherine Johnson won!
```

(b) expect1.txt