GAME 1: CONSOLE GAMES

UFCFWA-30-1 GAMES IN C++

Worksheet 1 deadline: 20:00 8th November 2020 (GitHub classroom will take your last commit at this point)

You can get worksheets signed off in a session before this date. We **strongly** suggest getting worksheets checked off as you complete them. Depending on feedback you may need to undertake further work.

To introduce you to basic C++ functionality we will create two console games. We will create a Guess the Number game and Rock Paper Scissors where you play against the computer.

We will be discussing concepts needed to complete this worksheet in the lectures over the next few weeks. Work through the worksheet as you understand the topics covered.

GITHUB CLASSROOM FOR WORKSHEET 1

To create your repository for worksheet 1, you will need to access the GitHub classroom 'assignment'. Follow this link to create your Console Games repository: https://classroom.github.com/a/QQZAHLqc

As a reminder:

- This link will take you to the classroom assignment. If you are not already linked to our GiC++ classroom, it will ask you to **link yourself with your Student User ID number, it is important that you do this!**
- Then follow through the instructions to create your own personal repository for Console Games.
- Now your remote repository has been created so you can commit your code here as you work on Worksheet 1.
- Remember you now need to **clone** the repository so you have your local copy to work on.

More information about GitHub repositories can be found in the 'Git: GitHub classroom and GitKraken' worksheet on Blackboard. ***It is recommended that you complete the Git worksheet first so that you understand the set up.***

PART 1: GUESS THE NUMBER

SEQUENTIAL PROGRAMMING

1. For the basis of a Guess the Number game, we would like you to create a program in which the computer generates a number between 1 and 10, the computer asks the user to guess the number and the user inputs one attempt at guessing the number.

    - First write a flow chart or some pseudocode to outline your program, problem analysis notes and techniques from the lecture may help. Here is a short explanation about pseudocode and also flowcharts: https://www.bbc.co.uk/bitesize/guides/z3bq7ty/revision/2

    - Implement the C++ code needed to output text asking the player for a value and then read in the value entered by the user.

        - Reference on reading in from the command line: https://en.cppreference.com/w/cpp/io/cin

BRANCHING

2. Next, you will need to use input and output functionality along with `if` and `else` statements.

    Implement the code detailed in the steps below:
    - Generate a random number between 1 and 10, assign it to an integer variable named target_number. For information on how to generate a random number in C++: http://en.cppreference.com/w/cpp/numeric/random/srand
    - Output a line of code asking the player to guess a number
    - Read in the player input
    - Check if the player input is equal to the target_number?
        - If it is then tell the player they have won
    - Check if the player input less than the target_number?
        - If it is then tell the player
    - Else the player input must be greater than the target_number
        - Tell the player

LOOPING

3. Adjust your Guess the Number game so that the player is able to keep making guesses until they get the correct answer. Use a `while` loop to provide this functionality - this will be your guess the number game loop.
    - Create a variable that will store a true/false value. This variable will keep track of when the game is running, so you might call the variable `is_running`.
    - Initialise the value of `is_running` to be `true`.
    - You want this variable to remain `true` until the game ends, in this case when someone wins the game. Decide when you need to assign this variable to `false` and add the code to the appropriate place.

    Tips
    - Think about the positioning of the `while` loop and the condition. What code do you want the program to keep repeating?
    - If you are not sure, write some pseudocode or draw a flow diagram to show what you want to happen.

4. Add a `for` loop so that the user has a maximum of 5 guesses to get the number
    - If the user doesn't guess correctly in these 5 guesses then output a 'You lose' message, along with the correct number.
    - Whether the player wins or loses ask if the user wants to play again. Hint: you can use your `is_running` variable here to continue or stop the game depending on the user response. So you will need to move your code so that the game doesn't necessarily stop on a win.
    - Be careful to get your `for` loop and your `while` loop nested together correctly.

PART 2: ROCK PAPER SCISSORS

This section of the tutorial is to help you learn about reading input from file and about using the `char` datatype by creating a Rock Paper Scissors game.

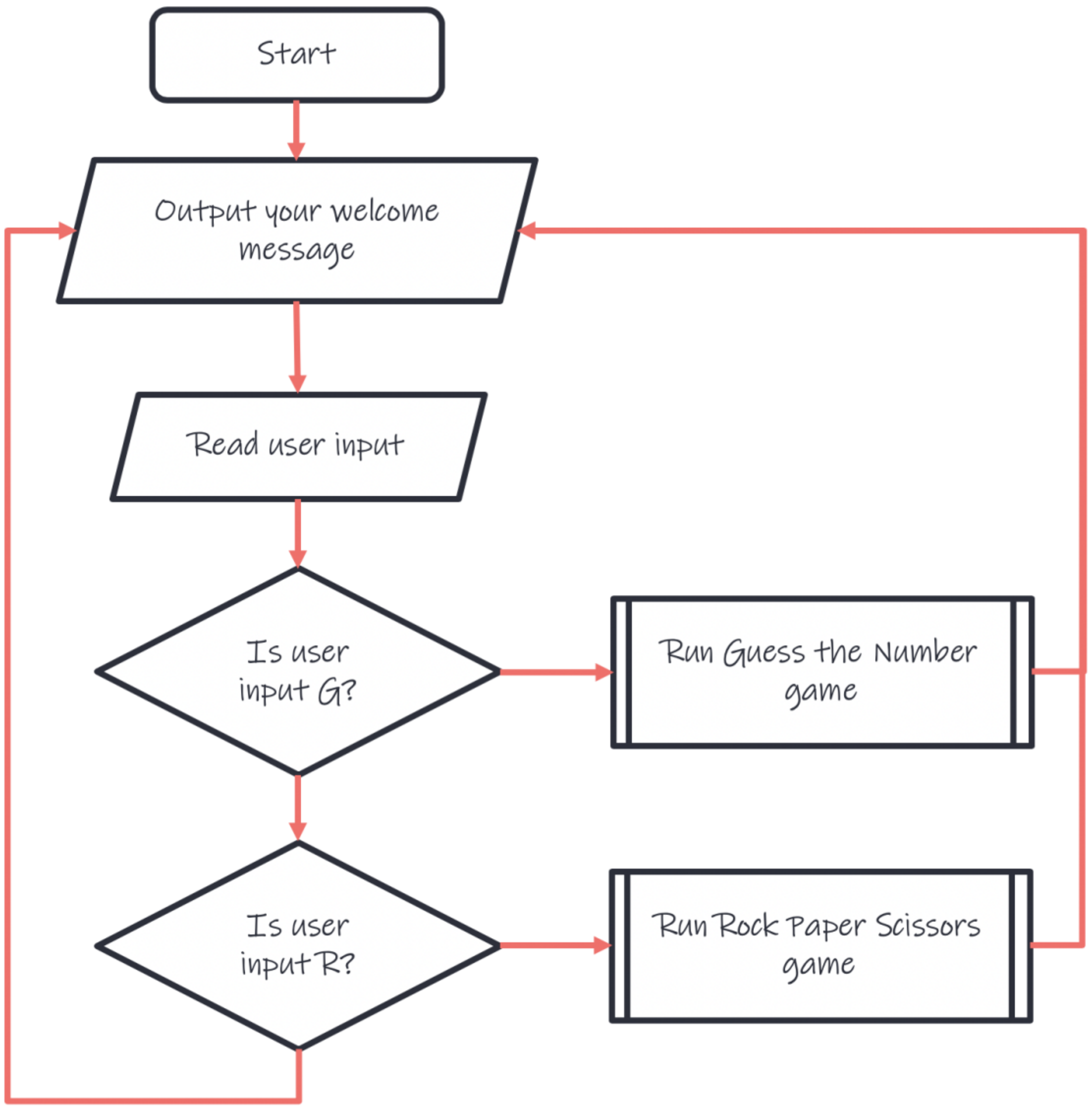
If you don't know the game: https://en.wikipedia.org/wiki/Rock–paper–scissors

Basic rules: Rock beats scissors, paper beats rock, scissors beat paper.

ADDING TO YOUR GAME LOOP

1. Add an option at the beginning of your program to choose whether you play Guess the Number or Rock Paper Scissors. This should sit inside a main game loop. Your game should play like this:

- Display a welcome message such as: "`Hello, welcome to console games. Press G to play Guess the number and press R to play Rock Paper Scissors`"
- If the user presses G then run the Guess the Number code your wrote in part 1.
- If the user presses R then… (you will insert the code into this section based on the tasks below)
- Once you finish playing either game then the game loop will loop back ask which game you want to play next (your initial welcome message).


Start
Output your welcome message
Read user input
Is user input G?
Run Guess the Number game
Is user input R?
Run Rock Paper Scissors game

FILE INPUT

2. To help you create your Rock Paper Scissors game:
   - RPS.txt (file on Blackboard) contains 10 computer turns.

- Open the file and then check that the file has opened successfully (if you're not sure look at the lecture slides)
- Read in the first computer turn value and store it in a variable.
- Get the player to enter their value and store that in a variable.
- Check whether the player or the computer win.
- Create variables `comp_win_tally` and `player_win_tally` and record who wins the point. You will need these tallies for the next section.
- Output the result to the console.

FUNCTIONS

We have a lecture on functions in week 3. If you are not yet familiar with functions, have a look here: http://www.learncpp.com/cpp-tutorial/14-a-first-look-at-functions/

You'll notice your code is getting quite long and it becomes more difficult to find the area you are working on. Functions will help you to separate your code into logical sections, as well as allowing you to easily repeat functionality.

2. Adjust your Rock Paper Scissors game so that the code to play a turn is put into a function of its own. This will make it easier for you to extend your code to play more than one turn. The function signature should look like this:

   ```
   int playTurn(char computer_value)
   ```

   It should take the computer value that was read from the file as a parameter (i.e. the letter it reads from the file each time). It should ask the player for their guess and then work out whether the player or computer won that turn. It should return an integer representing who wins (e.g. -1 for computer win, 0 for a draw and 1 for player win).

3. Now adjust your Rock Paper Scissors game so that the game has 5 turns. Add a `for` loop and call your function from within that loop.
   - Use the tallies that you were asked to create in the last section to store who wins each turn.
   - After the 5 turns output who won Rock Paper Scissors and by how much.

OPTIONAL: MORE LOOPS FOR RANDOMLY GENERATED PLAY

- Add an option at the beginning of your game for the user to choose whether you play using file input or a random number generator to create the computer's values. Using the random number generator means you can play for as long as you like, rather than being limited to the number in the file.

- Implement the code to generate random computer guesses for If the player selects the random number generator,
    - Tip: Generate a 1, 2 or 3 using the random number generator and allocate these values to be either rock, paper or scissors. This number then represents the computer's turn.
    - Add a `for` loop so that the program will continue to loop as long as the player wants.
    - Make sure you add an option for the player to input Q when they want to stop the game. At that point output the tally of who won and by how much.

OPTIONAL: ADDING STRATEGY

- Instead of just generating a random number for the computer's guess, try to apply some intelligence to the computer's guesses. See if you can make the game more difficult for a player to win.