Programming Project Report Richard Stone N0782914

Random Story Generator

Contents

[Specification 3](#_Toc7027956)

[Design 4](#_Toc7027957)

[Menu function 4](#_Toc7027958)

[Flow Diagram 5](#_Toc7027959)

[Name Class 6](#_Toc7027960)

[Name Generator 6](#_Toc7027961)

[Add To File 6](#_Toc7027962)

[File Length 7](#_Toc7027963)

[Random Number Generator 7](#_Toc7027964)

[Story Class 7](#_Toc7027965)

# Specification

The program should function as a random story generator, generating silly stories that should be fairly coherent. The primary function of the program will be the story generator, this will generate a story of a size determined by the user between the options of short and long. Once selected, a story of the specified length will be generated. The program will also generate character names, one for the protagonist and one for another character, who may be an antagonist or other relation to the protagonist. The name will be generated in the format [name] the [title], for example, “Richard the Stupid”. This will be generated from two text files, one containing all names and one containing all titles, the program will randomly pick one from each file and deliver it in the desired format. Another function will be the ability to view the number of names in the file, alongside the number of titles in the file. The user will also be able to input either a name or a title to the text document, which will be saved forever, unless it is deleted manually from the file.

The user will interact with the program via command prompt, loading the program from a .exe file. The menu will be text based and will rely upon the user’s input to be navigated, this input will be given via numbers which will correspond to their choice to proceed. For example, at the start the user will be presented with a choice of “1. Generate story 2. Add to files 3. Show file info 4. Exit program” The user will then enter one of those numbers, which will bring the user to their desired function. The program will then deliver the desired story into the console and will prompt the user to enter whether they would like the story to be saved. If the user does want the story to be saved, it will be saved to a rich text document with a name of their choosing. Rich text format has been chosen because this will preserve formatting, whereas Notepad which is the default for .txt will not. Therefore, .rtf will open by default in either WordPad or Microsoft Word, and therefore will have superior formatting.

After any function is completed, the program will allow the user to enter whether they would like to run the program again. If the user selects yes, they will be brought back to the start of the program and be prompted with the same initial question. If the user selects no, the program will shut down successfully. The program should also successfully validate any user input and will account for all forms of input.

# Design

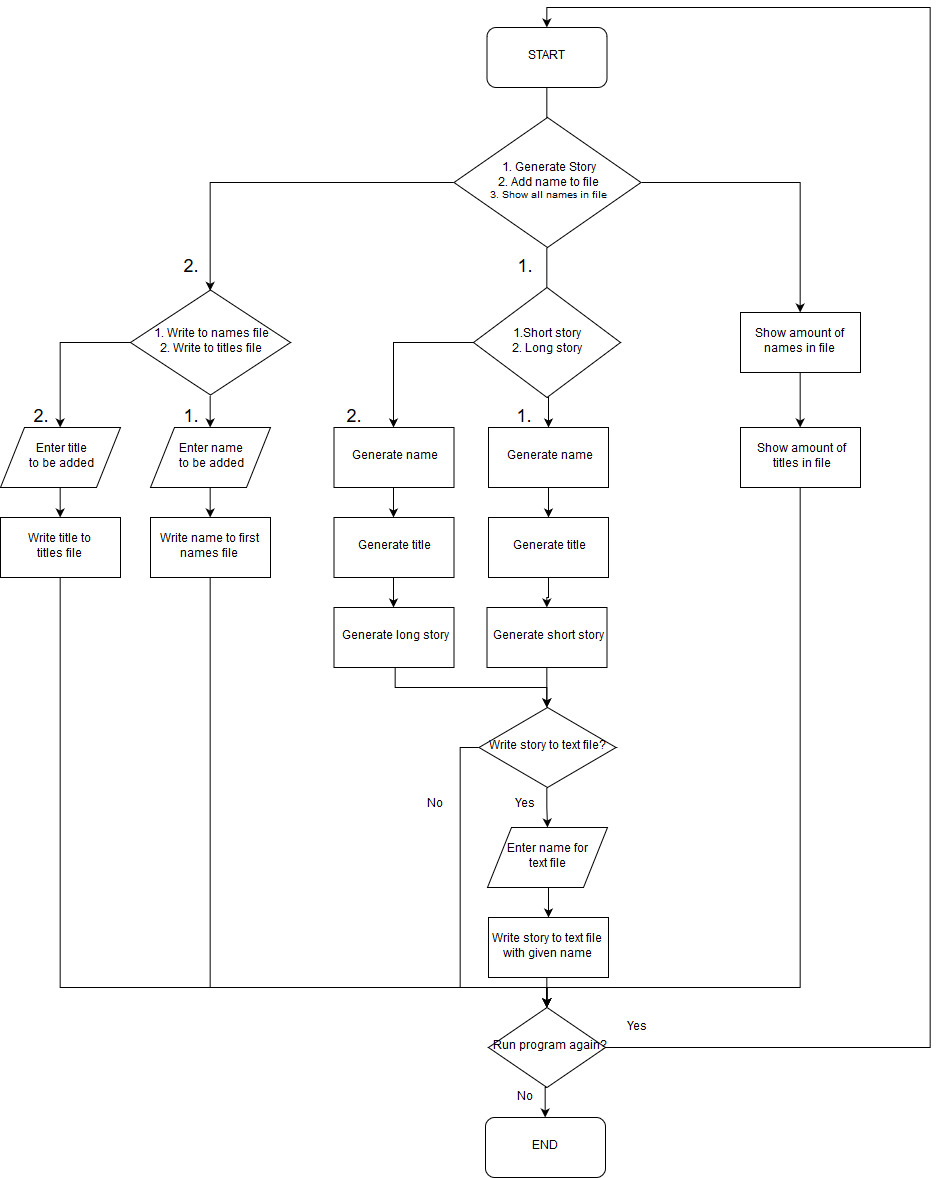
To function correctly, the program will require a certain amount of programming concepts to be incorporated. This will span from very basic concepts such as if statements, for loops, and while loops, to more complex concepts such as classes. Each main procedure of the program will have its own function, and each relevant function will be grouped together under a class. The main two classes will be a name to be generated, and a story to be generated. Under the name class, a function to select the name from the file must be created, a function to write more names into the file, and a function to determine how many names currently exist in the text file.

Within the story class, a function for each section of the story will be generated, an additional function will be used to ‘stitch together’ each fragment of the story in the correct order. Another function must be written to then write the story to a file, should the user wish to do this. Outside of any class, the menu function will be created. This will be the function for the primary navigation of the program as described in the specification. For the purpose of tidiness, the program will also be separated into header files and separate program files.

## Menu function

The menu function will exist independently from any class and will be the function called within the main function, from this function all other functions and classes will be defined. This is primarily for the neatness of the program and for efficiency. The menu will be navigated via simple string-based input in the form of numbers, the user will be presented with their options and must then input the corresponding number to their choice. The function itself will therefore primarily be made up of if statements including validation if the user does not enter a valid input choice. The menu, and overall program will be structured similarly to the presented flow diagram on the next page.

## Flow Diagram



## Name Class

The Name class will include the following functions:

* Name Generator
* Add To File
* File Length

\*\*Upon revisiting, a new function was deemed necessary:

* Random Number Generator

### Name Generator

The name generator will very simply generate a random number between 0 and the amount of lines in the file. It will generate this random number, then assign the corresponding value within the text file to a string variable. This will be done for both the first name and the title, for instance the generator might produce the first name “Andrew” and the title “The Smelly”, the function will then join them together into a single string and be returned as “Andrew The Smelly”.

\*\*Upon revisiting, this function was modified slightly to only return one name at a time, therefore the name and title must be generated separately. This is so that the name “Andrew” can be used within the final story without the title. It was also modified to use the random number generator function.

### Add To File

The add to file function will prompt the user to enter which file they would like to add a name or title to, between the choices of names file and titles file. It will then prompt the user to enter what name or title they would like to add and append it onto the end of the document on a new line. This name should then be able to be read back and used in the program alongside all the other names and should be accurately reflected in the file length function.

### File Length

The file length function will be used to determine how many items are in the file by incrementing a counter whilst iterating through the file. For instance, if a file holds 12 text items, each on their own line, the file length function should return the integer value 12. This function will be used when the user branches to the view file information from the menu. This will be applied to both the names file and the titles file.

### Random Number Generator

Due to the inefficiency and ineffectiveness of the built in rand() and srand() function, a new random number generator was deemed necessary . This random number generator allows a random number to be generated between 0 and any given max value. This will be used in conjunction with the file length in order to generate each name.

## Story Class

The story class will inherit from Name in order to use the random number generator to select each sentence from its array. The story class will contain functions for each segment of the story, each function will contain an array of relevant sentences. For instance, the introduction function would contain introductory sentences, such as “once upon a time”. The story class will also include a function to write the story to a file, should the user request this.

# Testing

|  |  |
| --- | --- |
| **Test** | **Expected Outcome** |
| Enter a value other than the parameters specified in each menu | The program should return an error and reset. |
| Enter an input into either the name file or the titles file | The program should input the name into the selected file, and the file length function should reflect this. |
| Select yes when saving to a text file and give the file a name | The program should create a file with the specified name and place it in the directory. If the file name already exists, it will be replaced. |
| Select yes when asked to repeat the program | The program should restart. |
| Select the generate story option at the start. | The generate story branch should begin, after selecting a story length, the story should then generate. |
| Select the add to files option at the start. | The add to files branch should begin, after selecting which file and entering the data, this should be added to the file. |
| Select the view file information option at the start. | The file information branch should begin, showing the user the length of each file. |
| Select the exit program option at the start. | The program should close. |