Project Report

Yusuf Aygün
2020400033

14.03.2024

1 Introduction

This program serves as an interpreter designed to manage character inventories in the context of J.R.R. Tolkien's fictional universe. The functionalities offered by the interpreter include processing and responding to specific types of sentences and questions, managing inventory data, and executing operations through a fictional world.

2 Workflow Overview

The program continuously reads input from the user until an "exit" command is encountered. The input is first parsed by a function called *'parse_sentence'*. Then it is checked for any invalidations by a function called *'initial_valid_check'*, this function do not check all the conditions for a sentence to be invalid, the input can still be invalid even it passed the initial check, more checks are done in executing sentences. After initial check, the input is executed according to its type, input can either be in the form of sentences or a question. Here's how the program handles each type of input:

2.1 Input Types and Processing

The inputs are executed differently after deciding whether the input is a question or sentence by comparing the last token with '?'.

- **Sentences**: Sentences include actions such as buying, selling, or moving items and characters; and conditions such as checking characters inventory or location. Each sentence follows strict syntax rules, and valid inputs result in changes to the internal state of character inventories or locations. Invalid sentences are flagged without causing state changes. If the sentence is invalid, the program outputs 'INVALID' and executes nothing, else it outputs 'OK'.
- **Questions**: The program answers questions about the total number of specific items, the location of characters, the presence of characters in specific locations, and comprehensive inventory details of a character. If the sentence is invalid, the program outputs 'INVALID'.

2.2 Answering Questions

If the sentence is a question, we enter a function called *'answer_question'* to answer the questions. This function checks the question sentence according to its question word and creates a response

if there is no invalid situation. It checks the sentence for possible invalidations and returns -1 if the sentence is invalid. If there are no problems, it prints the answer of the question and returns 0.

2.3 Executing Sentences

If the sentence is not a question, we enter a function called *'execute_sentences'* to execute actions. There may be multiple sentences in one input, so we run a while loop to execute all the sentences. First we start by checking whether the input contains 'if', if it does not contain it, that means the sentence contains just action sentences, execute the actions one by one after splitting them, else first split the condition sentences and check if the conditions are satisfied, than execute the sentences if so. After executing one full sentence, continue in while loop to execute other sentences, keep track of the start index of the sentences by using an int variable called *'start_index'*.

- **Splitting Sentences**: Split the sentences to get just one meaningful condition sentence or one action sentence, use functions called *'split_conditions'* and *'split_actions'*. These functions iterate through all tokens, starting from start index, to find the first condition or action word in the sentence, and then they calculate the total word count for the specific sentences according to their descriptions. After calculating, they allocate memory for the splitted sentences and assign pointers to the tokens of the sentences. Then they return a pointer to the pointers array of the executable sentence.
- **Checking Conditions**: After splitting, the condition sentences are checked via a function called *'condition_check'*, this function checks if the conditions are satisfied for different kind of conditions according to description of the conditions. Returns -1 if conditions are not satisfied and 0 if the condition is true.
- **Executing Actions**: The action sentences also executed by a similar function called *'execute_action'*, this function also checks the sentence according to its action word and executes the action if there is no invalid situation. It checks the sentence for possible invalidations and returns -1 without executing anything if the sentence is invalid. If there are no problems, it executes the action and returns 0.

3 Other Functionalities

To maintain the smooth workflow, the program uses different functions and logics.

- **Structures**: There are three structures in the program to keep track of Subjects, Items and Locations. These structures keep the needed data for their types. Also there are some functions are used in the program to manipulate and control these structures such as get functions and create functions, besides the functions, the program also uses 2 global arrays to hold the info for all subjects and locations, and an array for each subject to keep track of its items.
- **Other Functions**: The program also contains different function for executing actions, answering questions, controlling and parsing the input, and some others for different purposes. All functions have some control mechanism for errors and invalidations and they ensure that the program runs smoothly.