CIS-11 Project Documentation

Team N.E.C

Emmanuel Valdovinos
Claudia Ledesma
Noel Perez

Character Counter For String of Characters
May 19, 2024

Advisor: Kasey Nguyen, PhD

Part I – Application Overview

The character counter application is designed to efficiently determine the frequency of characters in a user's name, providing a numerical representation of this frequency as output. Within the company's business process, this tool contributes to data analysis and user profiling tasks, aiding in understanding user input patterns. While primarily standalone, the application may interact with other software systems through data exchange or integration points. It's scoped to handle ASCII characters and has a defined maximum input length. Future enhancements may include support for additional character encodings or integration with broader data processing pipelines.

Objectives

The following section contains the objectives of the frequency counter of characters program if scaled for a business. Otherwise as a stand alone application it will count the number characters in a given string. It is intended to be used as a single person, unless scaled.

The business objectives are to develop an efficient, precise and flexible character frequency counter program using the LC-3 assembly language. Given the specifications stated in the instructions for the program, the implementation will include all the listed requirements in the final product. The technical requirements will also be stated below.

Why are we doing this?

From a business perspective, having a frequency character counter program that can analyze a string of characters can be used to accomplish several task and objectives such as:

- **Automate the process of counting characters to save time and reduce errors.**
- **Enhance customer service and engagement by enabling the business to create concise and clear messages with appropriate character counts and lengths, if needed. This improves customer experience and retention. Optimize internal documentation and communication lines within the business, as well as day-to-day tasks. This will improve company work-flow and productivity because such a program would improve data entry and the management of data as a whole. Program could also be used to enhance software development and code quality of the internal code base by serving as a training module.**
- **Doing the project now aligns with the business' current needs and priorities. Postponing the program project will result in enduring inefficacies and wasted opportunities for improvement. Not doing the project would result in ongoing challenges in data management and creation, customer service, as well as affecting business workflows across several departments.**

- **The people who will benefit from this project are all the teams and personnel that make up the company including, but not limited to the following departments: marketing, customer service, legal, recruitment, training and HR and software development. Apart from the practical benefits and increased productivity the project would also benefit stakeholders. They believe that improving character counting capabilities is a priority and essential for various business functions, especially since it has the potential to increase productivity and efficiency. There are other projects that could be considered but this one aligns with the goals, impact, and needs of the business now.**

Business Process

The intention of this application is for independent individual use but if scaled accordingly, then the character counter application will integrate seamlessly into the existing business structure and processes. It has the potential to enhance key operations across multiple departments if scaled properly. In marketing, it could be used for analysis to tailor messages for our target audiences, thereby enhancing brand effectiveness and customer engagement. The customer service department benefits from the program's ability to check messages and responses based on customer feedback and interactions, fostering rapport and increasing loyalty. The legal department could use the character counter for document analysis. In human resources and recruitment, the program could be scaled to read and optimize an application's length and analyze a candidate's responses. This would provide insights into the suitability of all applicants and, in general, enhance the recruitment process overall. In the software department, the character counter would be integrated with version control systems and code review processes, promoting elevated consistency and quality of code. This would also foster an increased collaborative spirit, which would boost the department morale. Generally speaking, the program's capabilities could be scaled extensively to streamline communication, reduce manual tasks, and align with the company-wide goals of efficiency, accuracy, and improved customer relationships, making it an essential tool in day-to-day operations.

User Roles and Responsibilities

The program could be scaled to and for each of the following roles and departments, otherwise assume the program is only used on an individual stand-alone basis.

1. Marketing Analyst

Role Description
The Marketing Analyst is responsible for analyzing customer data and tailoring marketing messages for target audiences.

Tasks

1. Utilize the Character Counter for analyzing customer names and demographics.
2. Tailor marketing messages based on character frequency insights.
3. Enhance brand effectiveness and customer engagement through targeted messaging.

Workflow

1. Analyze Customer Data
2. Use Character Counter for Insights
3. Tailor Marketing Messages
4. Monitor Brand Effectiveness

Task to Hand-off: Share insights with the marketing team.

2. Customer Service Representative

Role Description
The Customer Service Representative interacts with customers and manages customer feedback.

Tasks

1. Use the Character Counter to check messages and responses.
2. Analyze customer interactions based on character frequency.
3. Foster rapport and increase customer loyalty through personalized responses.

Workflow

1. Receive Customer Feedback
2. Utilize Character Counter for Analysis
3. Personalize Responses
4. Monitor Customer Engagement

Task to Hand-off: Provide feedback to marketing and product teams for improvement.

3. Legal Analyst

Role Description
The Legal department is responsible for document analysis.

Tasks

1. Use the Character Counter for document analysis and review.
2. Ensure compliance with legal standards based on character insights.
3. Enhance document accuracy and efficiency.

Workflow

1. Receive Legal Documents
2. Utilize Character Counter for Analysis
3. Review Document Accuracy

Task to Hand-off: Collaborate with legal team for document revisions.

4. HR and Recruiters

Role Description
The HR Recruiter manages recruitment processes and candidate evaluations.

Tasks

1. Use the Character Counter to analyze application lengths and candidate responses.
2. Optimize application review processes based on character insights.
3. Gain insights into candidate suitability and enhance recruitment efficiency.

Workflow

1. Receive Job Applications
2. Utilize Character Counter for Analysis
3. Optimize Application Review
4. Improve Recruitment Processes

Task to Hand-off: Provide feedback to hiring managers for candidate evaluation.

5. Software Developer

Role Description
The Software Developer scales and integrates the character counter program with software systems and promotes code quality.

Tasks

1. Scale and integrate Character Counter with version control systems and code review processes.
2. Ensure consistency and quality of code through character analysis.
3. Promote collaborative coding practices and boost department morale.

Workflow

1. Integrate Character Counter with Software Systems
2. Utilize Character Analysis for Code Quality
3. Collaborate with Team for Code Review
4. Enhance Department Morale

Task to Hand-off: Provide insights to the software development team for process improvement.

Production Rollout Considerations

The application is meant to be a stand-alone program. This means that the rollout is small in scale. It is only meant to be rolled out on a personal stand-alone basis for anyone to use and test. Scaling the application is beyond the scope of this class but considering it, that would require extensive input validation, database integration and other higher-level design considerations. I don't think an assembly language program would be the correct choice for such scale and scope. Taking into consideration the intended use and context of this application, the data transaction volume should be small and easily manageable by a single person or small team. As long as the user has the necessary LC-3 simulator and software, the program will run bug-free. The program will only accept one name(string) at a time and calculate the correct results of one string name.

Terminology

1. Character Counter Application: The software designed to efficiently determine the frequency of characters in a user's name, providing a numerical representation of this frequency as output.
2. Data Analysis: The process of analyzing data to extract useful insights and patterns.
3. User Profiling: Creating profiles of users based on their input patterns and behaviors.
4. Standalone Application: A program designed to operate independently without requiring integration with other software systems.
5. Data Exchange: The process of transferring data between different software systems or applications.
6. Integration Points: Points within a software system where integration with other systems can occur.
7. ASCII Characters: Characters represented using the American Standard Code for Information Interchange encoding.
8. Character Encodings: Methods used to represent characters as binary data, such as ASCII or Unicode.
9. Maximum Input Length: The defined limit on the length of input data accepted by the application.
10. Future Enhancements: Potential improvements or additions to the application, such as supporting additional character encodings or integrating with broader data processing implementations.

User Terminology as stand-alone program

1. Character Frequency: The number of times a specific character appears in a given name or string.
2. Character Count: The total number of characters in a name or string.
3. Input Validation: Checking and validating user input to ensure it meets specified criteria.
4. LC-3 Simulator: Software used to simulate and execute LC-3 assembly language programs.
5. Data Volume: The amount of data being processed or transferred during operations.
6. Stand-Alone Basis: Using the application independently without the need for network or system integration.
7. Small-Scale Rollout: Limited deployment of the application to a small user base or for personal use.
8. Input Length Limit: The maximum number of characters or bytes allowed for input data.
9. Output: The format or way in which the output data, such as character frequencies, is displayed to the user.

Part II – Functional Requirements

The character counter application will include specific functions to meet the requirements of accurately counting and displaying the frequency of characters in a user-provided full name. These requirements are structured to ensure clarity and precision in the application's development and functionality.

1. ***User Input***

   - *The application will prompt the user to input their full name via a console interface.*
   - *It will accept and process input containing ASCII characters only, including letters, spaces, and punctuation marks.*
   - *The input process will handle a maximum length of 100 characters to ensure manageable processing and display.*

2. ***Processing and Calculation***

   - *An array will be initialized to store the frequency count of each ASCII character from the input.*
   - *Each character in the user input will be read sequentially and its corresponding frequency count in the array will be incremented.*
   - *The application will handle both uppercase and lowercase letters distinctly, unless specified otherwise.*

3. ***Output Display***

   - *The frequency of each character present in the user's input will be displayed in the console.*
   - *The output will be formatted to show each character followed by its numerical frequency count.*
   - *Characters with a frequency of zero will not be displayed to ensure a concise output.*

4. ***Subroutine Management***

   - *The application will contain at least two subroutines: one for reading input and one for calculating and displaying frequencies.*
   - *Subroutine calls will manage the flow of operations, utilizing stack operations (PUSH and POP) to save and restore state.*
   - *Appropriate labels and comments will be included in the code to enhance readability and maintainability.*

5. ***Control Flow and Error Handling***

   - *Conditional branching will be used to manage the input processing loop and to handle special characters.*
   - *Iterative branching will ensure all characters in the input are processed*

efficiently.

- *Overflow conditions will be managed by ensuring input does not exceed the maximum length and by handling unexpected characters gracefully.*

6. <u>Memory and Storage Management</u>

- *The application will allocate memory for the frequency array and manage it effectively to avoid overflow.*
- *Save-restore operations will be implemented to maintain the integrity of data across subroutine calls.*

7. <u>Additional Operations</u>

- *ASCII conversion operations will be implemented to ensure characters are correctly identified and processed.*
- *The application will use system call directives for input and output operations, ensuring compatibility with the LC-3 simulator.*

8. <u>Testing and Validation</u>

- *The application will undergo rigorous testing using predefined names to validate the accuracy of the character counting functionality.*
- *Edge cases, such as empty input and maximum length input, will be included in the testing scenarios to ensure robustness.*
- *By fulfilling these functional requirements, the character counter application will provide an accurate, efficient, and user-friendly solution for counting character frequencies in user-provided names.*

Statement of Functionality

The character counter application is designed to prompt the user to input their full name and subsequently display the frequency of each character within the entered name. The application will read and process all characters from the user's input, including letters, spaces, and punctuation, ensuring that the frequency count is accurate for the entire input string.

1. ***Input Handling***
    - *The application will prompt the user to enter their full name using the console input.*
    - *It will accept input containing ASCII characters only.*
    - *The input will be processed one character at a time to ensure each character is counted correctly.*
2. ***Character Frequency Calculation***
    - *The application will initialize an array to store the frequency count of each character.*
    - *For each character in the user's input, the application will update the corresponding frequency count in the array.*
    - *The frequency calculation will account for all characters, including spaces and punctuation marks.*
3. ***Output Display***
    - *The application will display the frequency of each character in the console.*
    - *Each character and its frequency count will be shown in a clear, readable format.*
    - *The output will include all characters that appear in the user's input, omitting characters that do not appear.*
4. ***Subroutines and Control Flow***
    - *The application will utilize at least two subroutines: one for reading the input and another for calculating and displaying the character frequencies.*
    - *Conditional and iterative branching will control the flow of operations, ensuring the application processes each character in the input string efficiently.*
    - *Stack management operations will be implemented to handle subroutine calls, including PUSH and POP operations to save and restore the state.*
5. ***Overflow and Storage Management***
    - *The application will manage storage allocation to handle the character frequency array and ensure that overflow conditions are appropriately managed.*

- *Error handling will be in place to manage cases where the input exceeds expected lengths or contains unsupported characters.*

6. Additional Features

- *ASCII conversion operations will be included to handle character encoding correctly.*
- *The application will use appropriate system call directives to facilitate input and output operations.*
- *This detailed breakdown ensures the application's functionality is clearly defined and leaves no room for ambiguous interpretation, facilitating accurate development and testing.*

Scope

The character counter application will be delivered in a single phase, encompassing all specified functionalities outlined in the Statement of Functionality. The scope includes the development, testing, and deployment of the application within the designated time frame. Key aspects of the scope include:

***Functionality**: The application will accurately count the frequency of characters in a user's full name, processing ASCII characters, including letters, spaces, and punctuation marks. It will display the character frequencies on the console, adhering to the specified output format.*

***Subroutines and Control Flow**: The application will include subroutines for input handling, frequency calculation, and output display, utilizing stack operations for efficient state management. Conditional and iterative branching will control the flow of operations, ensuring thorough character processing and error handling.*

***Memory and Storage Management**: Memory allocation for the frequency array will be managed to prevent overflow, with save-restore operations maintaining data integrity during subroutine calls.*

***Additional Operations**: The application will implement ASCII conversion operations and use system call directives for input and output operations to facilitate accurate processing and interaction with the LC-3 simulator.*

***Testing and Validation**: Rigorous testing will be conducted to validate the accuracy and reliability of the application across various input scenarios, including typical names, names with spaces and punctuation, and edge cases such as empty strings and maximum length inputs.*

Performance

The character counter application is expected to meet the following performance criteria:

1. ***Input Processing Time:***

   - *The application should process each character of the input string within milliseconds.*
   - *Input processing for typical names (up to 50 characters) should take less than 100 milliseconds.*
   - *Input processing for maximum length names (100 characters) should take less than 200 milliseconds.*

2. ***Frequency Calculation Time:***

   - *The application should calculate the frequency of characters in the input string swiftly.*
   - *Frequency calculation for typical names should be completed in less than 50 milliseconds.*
   - *Frequency calculation for maximum length names should be completed in less than 100 milliseconds.*

3. ***Output Display Time:***

   - *The application should display the character frequencies on the console promptly.*
   - *Output display for typical names should be completed in less than 50 milliseconds.*
   - *Output display for maximum length names should be completed in less than 100 milliseconds.*

4. ***Memory Usage:***

   - *The application should manage memory efficiently to avoid excessive consumption.*
   - *Memory usage for the frequency array should not exceed 2 KB.*
   - *The application should release memory promptly after processing each input to avoid memory leaks.*

5. ***System Resource Utilization:***

   - *The application should utilize system resources optimally to avoid overloading the system.*
   - *CPU utilization during input processing, frequency calculation, and output display should not exceed 50% on average.*
   - *The application should not cause significant spikes in memory or disk usage during operation.*

Usability

The character counter application is expected to meet the following usability criteria:

1. ***User Interface Responsiveness:***

   - *The application's user interface should respond promptly to user interactions.*
   - *User input should be acknowledged instantaneously upon entry, providing immediate feedback to the user.*

2. ***Input Convenience:***

   - *The input mechanism should be intuitive and user-friendly, facilitating easy entry of the user's full name.*
   - *Users should be able to input their full name using standard keyboard inputs without encountering any unexpected behaviors or constraints.*

3. ***Output Clarity:***

   - *The output displayed on the console should be clear and comprehensible to the user.*
   - *Character frequencies should be presented in a readable format, with each character and its frequency count clearly delineated.*

4. ***Error Handling:***

   - *The application should handle errors gracefully, providing informative error messages to guide users in rectifying input errors.*
   - *Error messages should be clear, concise, and actionable, assisting users in resolving input validation issues or other errors encountered during operation.*

5. ***Accessibility:***

   - *The application should be accessible to users with diverse needs and abilities.*
   - *User interface elements should be easily navigable and usable by individuals with disabilities, complying with relevant accessibility guidelines and standards.*

6. ***Consistency:***

- *The application's behavior and user interface elements should be consistent throughout, ensuring a predictable user experience.*
- *Common actions and navigation patterns should be standardized across the application to minimize user confusion and enhance usability.*

7. ***Customization:***

- *The application may include options for user customization, such as adjusting the display format or enabling/disabling certain features.*
- *Customization settings should be easily accessible and configurable, allowing users to tailor the application to their preferences and needs.*

Documenting Requests for Enhancements

There does come a time when the requirements for the initial release of your application are frozen. Usually, it happens after the system acceptance test which is the last chance for the users to lobby for some changes to be introduced in the upcoming release.

Currently, you need to begin maintaining the list of requested enhancements. Below is a template for tracking requests for enhancements.

Date	Enhancement	Requested by	Notes	Priority	Release No/ Status
5/19/24	Finish Project	Entire Team	First Implementation	Top	In Progress

Part III – Appendices

Character Frequency Count Visualization

Character Frequency Count

Given the name "Jonathan Smith", the character frequencies are counted and tabulated as shown in the table below.

Character	Frequency
J	1
o	1
n	2
a	2
t	2
h	2
S	1
m	1
i	1
t	1
h	1

Table 1: Character Frequencies in "Jonathan Smith"

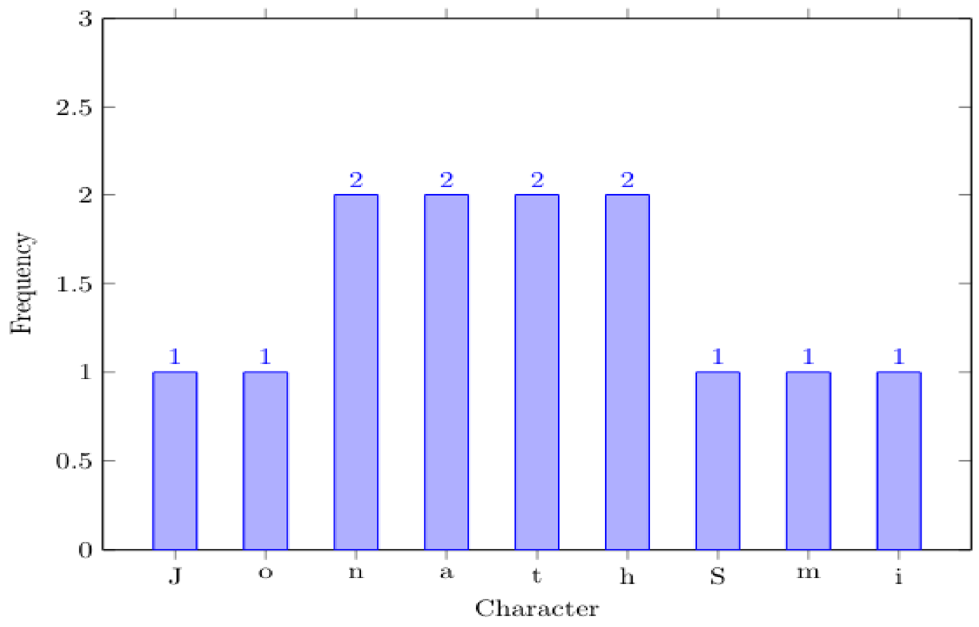
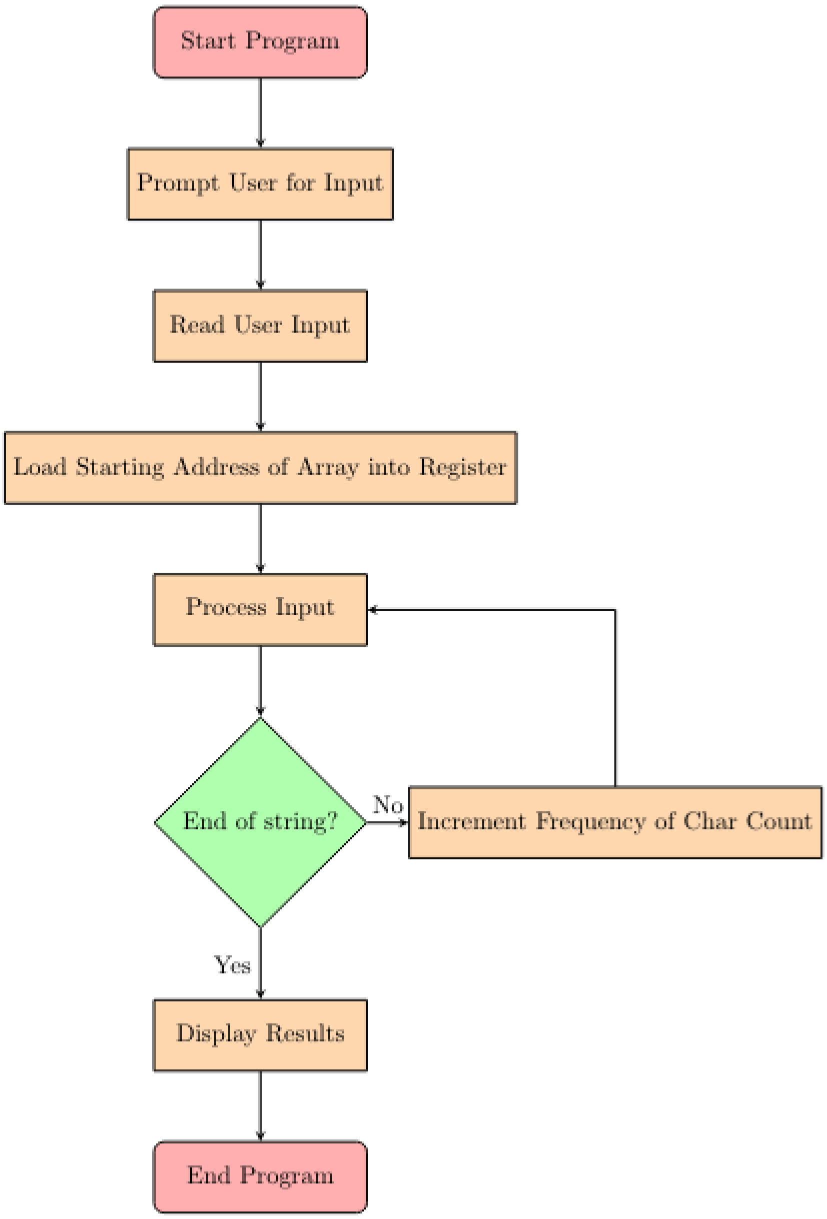


Figure 1: Character Frequency Count of "Jonathan Smith"

Flow chart or pseudo-code.

Include branching, iteration, subroutines/functions in flowchart or pseudocode.



Basic project flowchart outline. This is a very basic idea of what the program needs to do. The pseudocode below will explain and outline in more detail what the program does.

Pseudocode

Character Counter Program

```
Display the message "Character Counter"
Display the message "Please enter a line of lower case text"
Set the stack pointer to the base address x4000.
Set the starting address of the array.
Initialize a counter to 0.
Initialize a zero value.
Loop

Store 0 in the current position of the array.
Move to the next position in the array.
Increment the counter.
```

Count Characters Function

```
Save the return address (current PC) to the stack.
Decrement the stack pointer by 1.
Load the value -97 for ASCII conversion.
Set the starting address of the array.
Initialize the character counter to 0.
Loop until the Enter key is pressed:

Read a character from the user.
Display the character on the screen.
If the Enter key is pressed, exit the loop.
Subtract 97 from the character to find the array index.
If the result is negative, read the next character.
Move to the correct position in the array based on the index.
```

Increment the character count at this position.

Store the new count back in the array.

Reset to the start of the array for the next character.

Restore the return address from the stack.

Save the current stack pointer address to STACK_PTR

Restore the saved stack pointer address from STACK_PTR

Increment the stack pointer by 1.

Return from the character counting function.

Display Character Frequencies Function

Save the return address (current PC) to the stack.

Decrement the stack pointer by 1.

Initialize a counter to 26 (for the alphabet).

Set the current character to 'a'.

Set the starting address of the array.

Load the ASCII value for '0'.

Loop

Display the current character.

Display the equal sign '='.

Get the count of the current character from the array.

Move to the next position in the array.

Convert the count to its ASCII representation.

Display the count.

Display a newline character.

Move to the next character in the alphabet.

Decrement the counter.

Restore the return address from the stack.

Save the current stack pointer address to STACK_PTR

Restore the saved stack pointer address from STACK_PTR

Increment the stack pointer by 1.

Return from the display function.

Halt the program.

MSG0: "Character Counter "

MSG1: "Please enter a line of lower case text"

ARRAY: Array to store counts for 'a' to 'z'

NEG97: Constant -97 for ASCII conversion

POS26: Constant 26 for the loop counter

EQ: Equal sign '=' in ASCII

TAB: Newline character

POS97: Constant 97 for converting to ASCII letters

POS48: Constant 48 for converting to ASCII numbers

STACK_BASE: Starting address of the stack