PROJECT REPORT

Course Code: CSEG1032

Course Title: Programming in C

Project Title: Multi-Currency Exchange Rate Viewer (C Project)

Student Name: Siddhant Sharma

Roll Numbers: 590023054

Semester: I

ABSTRACT: An Interactive Exchange Rate Viewer

This document represents a comprehensive overview of a C-language project designed for visualizing currency exchange rates. It outlines the problem definition, system architecture, implementation specifics, testing methodologies and avenues for future development. The primary aim is to provide the students and instructors with a detailed reference for reviewing the code and understanding the underlying concepts of C programming in practical application.

Project Overview

This project delivers an interactive console-based application for displaying and analyzing historical currency exchange rate data. Leveraging fundamental C programming constructs, it provides users with the ability to query exchange rates for specific currencies, periods, and perform basic comparative analysis.

Currency exchange rates fluctuate regularly based on economic, political, and market conditions. Managing, analyzing, and retrieving historical exchange rate data is essential for financial decision-making. This project, Currency Exchange Rate Analysis System, is designed to read exchange rate data from CSV files and allow users to:

- Display exchange rates for a particular country code for an entire year.
- Display exchange rate for a specific month.
- Analyze exchange rate trends between two years.
- Store exchange rates of all countries for a specific year or month into an output text file.
- Display country names using a separate Country Information CSV.

This project uses concepts of file handling, string manipulation, user-defined functions, structures, and data parsing in the C language.

Problem Definition: Navigating Currency Volatility

In an increasingly globalized economy, understanding and tracking currency exchange rates is paramount for various stakeholders, including international businesses, investors, and individuals engaging in foreign transactions. The challenge lies in efficiently accessing, processing and presenting this often-voluminous data in a user-friendly format. Manual analysis of raw exchange rate data can be time-consuming and prone to errors.

Current Challenges Identified:

1. Data Accessibility

Most of the time, raw exchange rate data are stored in massive CSV or text-based files, which generally make it hard for regular users to come up with any meaningful insights. Going through such data without an appropriate user interface is very time-consuming. Users may find it difficult to fetch specific currency figures fast and in a proper manner.

2. Information Overload

Exchange rate datasets might have hundreds of currency pairs, prices for different time frames, and other meta-information. Such an excessive amount of data could bewilder the users who just need a few numbers for trips, businesses, or conversions. A simplified version is needed to get rid of the noise and provide a more focused output.

3. Trend Analysis

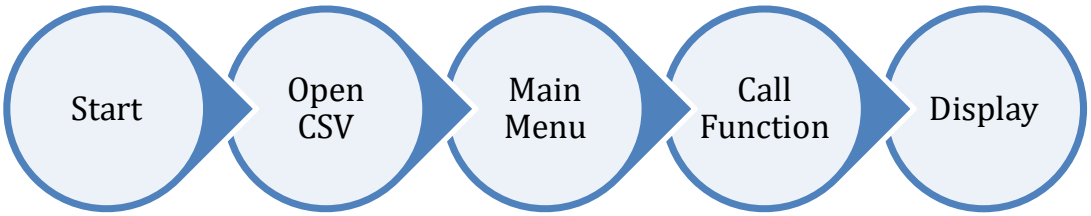
Knowing the exchange rates that have changed over time involves having such tools that can identify patterns rather than just values. In the absence of automation, users have to manually look for comparisons of the old values, which is an inefficient process and, furthermore, they are prone to make mistakes. The lack of easily understandable trend indicators hinders financial decision-making.

4. Customisation

Most of the raw exchange-rate datasets lack the provision of user-specific options for filtering or formatting.

System Design: Modular Architecture for Clarity

The system is designed with a modular approach to ensure maintainability, readability and ease of debugging. Each core functionality is encapsulated within dedicated functions, promoting a clear separation of concerns. The primary input for the system is a CSV file containing historical exchange rate data, with the first column typically representing the data and subsequent columns representing currency codes and their respective rates.



Key Components and their interactions:

- **Data Input:** The system ingests exchange rate data from a CSV file. The OpenCSVFile function handles file opening and error checking.
- **CSV Parsing:** The parseCSV function is critical for breaking down each line of the CSV file into individual data points(columns), enabling structured access to date and currency rates.
- **User Interface & Navigation:** A main menu loop allows user to select various operations. Functions like Display, DisplayColor and ChecksScreenCount manage console output, colourisation and pagination for a better user experience.
- **Data Processing Functions:** Each analysis or display task (e.g.,ShowRatesForYear) has its own dedicated function, which queries the parsed based on user criteria.
- **Output:** Results are displayed to the console or written to external files as specified by the user's choice.

Module Breakdown

1. Reading Country Information

Function: ReadCountries()

- Reads "All-Countries.csv"
- Stores country code, country name, and description in the Currency-Info structure.
- Allows conversion from currency code → country name.

2. Display Exchange Rates for a Year

Function: ShowRatesForYear(FILE *fp)

- Prompts for country code and year.
- Locates the corresponding column in the CSV.
- Displays all 12 months of that year with their exchange rates.
- Example Output:

Exchange Rates for USD in 2023

2023-01: 82.34

2023-02: 82.12

...

3. Display Exchange Rate for a Specific Month

Function: ShowRatesForMonth(FILE *fp)

- Input:
- Country Code (USD/INR/AED etc.)
- Month (YYYY-MM)
- Displays single month’s exchange rate.

4. Analyze Country's Exchange Rate Between Two Years

Function: AnalyzeCountryBetweenYears(FILE *fp)

Features:

- Reads data between given start and end years.
- Calculates:
- Minimum rate
- Maximum rate
- Average rate
- Displays records for all months in the range.
- Example:
- Summary for EUR (2021–2023)
- Minimum : 0.84
- Maximum : 0.93
- Average : 0.88

5. Store Rates for All Countries in a Year/Month

Function: StoreRatesForAllCountriesInYear(FILE *fp, int MonthorYear)

- Writes exchange rate data of all countries into a text file.
- Two modes:
- Year Mode (MonthorYear = 4) → match YYYY
- Month Mode (MonthorYear = 7) → match YYYY-MM
- Output saved in:
- C:/Personal/Siddhant/ExchangeRate_Year.txt

Example Output:

- Currency Rate Country
- USD 82.30 -> United States
- EUR 89.40 -> Europe

Data Structures Used

1. **Structure: CurrencyInfo**

```
typedef struct {
    char code[10];
    char country[50];
    char desc[100];
} CurrencyInfo;
```

Used to store:

- Currency code
- Country name
- Description

2. **Arrays**
    - structCountry[MAX_COUNTRIES]

    Stores all country information.

3. **2D Arrays**
    - columns[MAX_COLS]

    Used to parse CSV columns.

Algorithm

- Show the user all the currencies that can be chosen.
- Take the user input for the selection.
- Locate the exchange rate from the given values or a file by using fopen() and fclose() if necessary.

- Output the result in a user-friendly manner together with the conversion value.

- Based on the user's choice, either return to the primary menu or quit the program.

File Operations

The program uses:

- fopen()
- fgets()
- rewind()
- fprintf()
- fclose()

CSV files:

1. **Exchange Rates CSV File**

   Contains:

Date,USD,INR,AED,EUR

2023-01,82.3,1,3.67,0.84

2. **All-Countries.csv**

   Contains:

USD,United States,Dollar

INR,India,Rupee

Implementation Details

Data Structure Implementation:

A user-defined structure CurrencyInfo containing currency code, country name, and description.

```
typedef struct
{
    char code[4];
    char country[200];
    char desc[50];
} CurrencyInfo;
```

Function Implementation :

A user defined function which prints the message in a particular color.

```
void Display(const char *msg)
{
//    screenRowCount++;
    switch(colorchoice)
    {

        case 1: // Blue
        printf("\033[1;34m%s\n\033[0m", msg);
        break;
        case 2: // Cyan
        printf("\033[1;36m%s\n\033[0m", msg);
        break;
        case 3: // Mangeta
        printf("\033[1;35m%s\n\033[0m", msg);
        break;

        default:
        printf("\033[1;37m%s\n\033[0m", msg);
    }
}
```

Testing & Result

1. Choose the function to be invoked

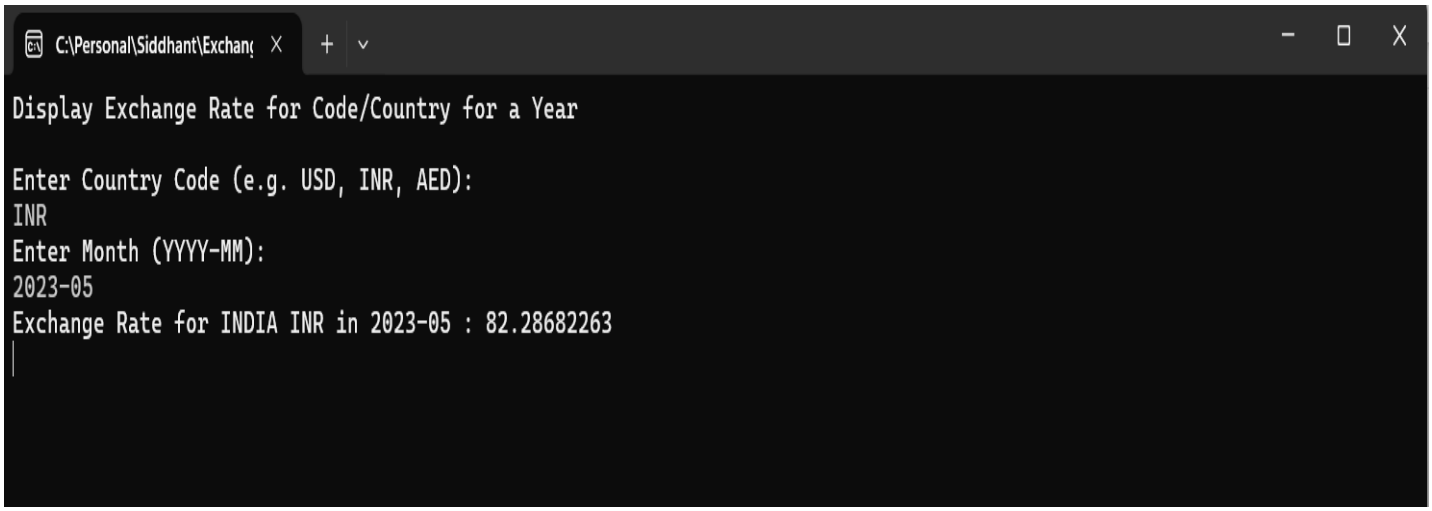
```
int main()
{
    char choice;
    char msg[100] = "You selected option:";
    FILE *fp;
    const char *filePath= "C:/Personal/Siddhant/ExchangeRate/rates.csv";

ReadCountries();
    while(1==1)
    {
        ClearScreen();
        Display("============================================");
        Display("              Exchange Rate Menu");
        Display("============================================");
        Display("1.  Display Exchange Rate for Code/Country for a month: yyyy-mm"); // 1 RECORD
        Display("2.  Display Exchange Rate for Code/Country for a Year: yyyy"); // 12 RECORD
        Display("3.  Display Exchange Rate for ALL Countries for a Year: YYYY");
        Display("4.  Display Exchange Rate for ALL for a month: yyyy-mm"); // show data for particular Year/Month
        Display("5.  Analyze Exchange Rate for Code/Country between Years: YYYY-YYYY");
        Display("6.  Change Color");
        Display("0.  Exit");
        Display("============================================");

        Display("Enter your choice: ");
        scanf("%c", &choice);
```

2. The function invoked

```
void ShowRatesForMonth(FILE *fp)
{
    char line[1024];
    char *columns[MAX_COLS];
    int targetCol = -1;
    char message[200];
    char country[10], month[10];
    ClearScreen();
    Display(ShowRatesForMonth_Title);
    Display("\nEnter Country Code (e.g. USD, INR, AED): ");
    scanf("%s", country);
    strupr(country);

    Display("Enter Month (YYYY-MM): ");
    scanf("%s", month);

    rewind(fp);  // Go back to the top of the CSV file

    // ========== Read Header Row to Locate Country Column ==========
    if (!fgets(line, sizeof(line), fp))
    {
        Display("Error reading header row.\n");
        return;
    }
    line[strcspn(line, "\r\n")] = '\0';
```

```c
if (targetCol == -1)
{
    sprintf(message, "Country code '%s' not found.\n", country);
    Display(message);
    return;
}

// ========= Search CSV Rows for the exact month =========
while (fgets(line, sizeof(line), fp)) {
    line[strcspn(line, "\r\n")] = '\0';

    int count = parseCSV(line, columns, MAX_COLS);
    if (count <= targetCol) continue;

    // date is in columns[0], e.g. "2024-03"
    if (strcmp(columns[0], month) == 0)
    {

        sprintf(message, "Exchange Rate for %s %s in %s : %s", GetCountryByCode(country), country, month, columns[targetCol]);
        //printf("\nExchange Rate for %s in %s : %s\n", country, month, columns[targetCol]);
        Display(message);
        return;  // stop after finding the month
    }
}
```

3. Result



Conclusion

The project successfully demonstrates how C can be used to process large CSV files, analyze financial data, and produce meaningful reports. It provides efficient search, filtering, and statistical calculation capabilities.

This Currency Exchange Rate Analysis System can be extended further for banking, finance, or economic forecasting applications.

We can improve/enhance the project by incorporating:

- Add GUI using GTK/C++ Qt.
- Add graphs for currency trend.
- Other input files
- Make paths configurable.
- Add error logs.

References:

- "The C Programming Language" — Kernighan & Ritchie
- GCC Documentation
- Currency data from Open-Source datasets
- Country & Currency list (source Wikipedia: ISO 4217 - Wikipedia)
- K&R: The C Programming Language, Kernighan & Ritchie
- CSV file format references and parsing tips: RFC 4180