**Currency Converter**

**Project report submitted in partial fulfillment of the Requirements for theAward of the Degree of**

**BACHELOR OF TECHNOLOGY**

**In**

**COMPUTER SCIENCE AND ENGINEERING**

**By**

**KAVALI SUDEEPTHI - 24KB1A05S1**

**LANKA ANGELIN BLESSY- 24KB1A05W7**

SHAIK SABIHA - 24KB1A05JG

**Under the Guidance of**

**SMT.SRUTHI**

****

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**NBKRIST**

**NBKR INSTITUTE OF SCIENCE AND TECHNOLOGY**

**(AUTONOMOUS)**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**



###### CERTIFICATE

This is to certify that the project report entitled **CURRENCY CONVERTER**being submitted by

KAVALI SUDEEPTHI-24KB1A05S1

LANKA ANGELIN BLESSY- 24KB1A05W7

SHAIK SABIHA- 24KB1A05JG

in partial fulfillment for the award of the Degree of Bachelor of Technology in Computer Science and Engineering to the NBKR INSTITUTE OF SCIENCE AND TECHNOLOGY is a record of bonafied work carried out under my guidance and supervision.

**SMT.SRUTHI DrA Raja Sekhar Reddy**

|  |  |
| --- | --- |
| **Assistant Professor** | **M.Tech, Ph.D**  **Head of the Department** |

**DECLARATION**

I hereby declare that the dissertation entitled **<Currency Converter>** submitted for the B.Tech Degree is my original work and the dissertation hasnot formedthe basisfor theaward ofany degree, associateship, fellowship or any other similar titles.

Place: KAVALI SUDEEPTHI

Date: (24KB1A05S1)

LANKA ANGELIN BLESSY

(24KB1A05W7)

SHAIK SABIHA

(24KB1A05JG)

ACKNOWLEDGEMENT

I sincerely thank my faculty guide, Ms. B. Shruthi, for her guidance and support throughout my project, "Currency Converter." I also thank Dr. A. Raja Sekhar Reddy, Head of Department, for providing the necessary resources and encouragement.

Gratitude to my friends and teammates for their cooperation and motivation, and to my family for their constant support.

This project enhanced my knowledge of C Programming, especially in Linked Lists, Structures, and Dynamic Memory Management, while giving me valuable insight into real-world application development.

Thank you all once again!

### 

### ABSTRACT OF THE PROJECT

### PROJECT TITLE:-Currency Converter

The Currency Converter project is a console-based application developed in the C programming language, designed to convert an amount from one currency to another based on predefined exchange rates. The core functionality of the application is built using **arrays** to store currency codes and names, and **linked lists** to manage dynamic exchange rate data efficiently. Arrays provide fast access to currency data, while the use of linked lists allows flexible insertion, deletion, and updating of exchange rates without memory wastage. Users can input the source currency, target currency, and the amount to be converted, and the program calculates and displays the equivalent value. This project demonstrates key data structure concepts such as arrays and linked lists in a real-world application, reinforcing their practical use in managing and organizing data. It is an ideal implementation for understanding dynamic data handling and struct-based programming in C.

### CHAPTER 1: INTRODUCTION

### Currency conversion plays a crucial role in the modern financial system, especially in the context of globalization, international trade, and tourism. With the advent of technology, software applications for real-time currency conversion have become increasingly essential.

### 1.1 Objective of the Project

### The objective of this project is to develop a simple yet efficient currency converter that can convert the value of one currency into another based on current exchange rates. The converter will support multiple currencies and provide accurate and real-time conversion results.

### 1.2 Scope of the Project

### The currency converter will be a user-friendly software application that supports several widely-used currencies. It will have a graphical user interface (GUI) that accepts user input, processes it, and displays the converted currency value.

### 1.2.1 Features

### Real-time exchange rate fetching

### Support for major world currencies

### User-friendly interface

### 1.2.2 Limitations

### Requires an active internet connection for real-time data

### May have slight delays in fetching live rates during server load

### 1.3 Methodology

### The application will be developed using Python, utilizing APIs to fetch live exchange rates. The GUI will be built using the Tkinter library. The following flowchart outlines the basic flow of the application:

### CHAPTER 2: LITERATURE SURVEY / EXISTING SYSTEM

### 2.1 Overview of Currency Converters

### Currency converters are widely used tools that enable users to convert an amount from one currency to another using the current or predefined exchange rates. With globalization, international travel, cross-border e-commerce, and foreign investments becoming increasingly common, currency converters play a crucial role in simplifying transactions involving different currencies.

### Over the years, several sophisticated currency converters have emerged, ranging from online tools and mobile apps to advanced enterprise-level software integrated with real-time financial data. These tools often use APIs to fetch live currency exchange rates and provide real-time conversions. Some even include advanced features such as historical trends, graph plotting, and offline saving.

### However, most of these tools rely on external data sources and internet connectivity. Furthermore, they are typically developed using high-level programming languages or web frameworks, which may not be suitable for academic learning of fundamental programming concepts like arrays and linked lists in C.

### 2.2 Existing Systems

### Some well-known currency converters include:

### Google Currency Converter A web-based tool integrated into Google Search, it fetches real-time exchange rates and allows instant conversion between currencies.

### XE Currency Converter A popular website and mobile app that offers real-time conversions, rate tracking, historical data, and supports dozens of currencies.

### OANDA Currency Converter A tool used by professionals, offering live updates, advanced analytics, and customizable features.

### While these tools are effective, they are highly dependent on internet access and are not open-source or built in a manner that demonstrates low-level data structure implementations.

### 2.3 Limitations of Existing Systems in the Context of Educational C Programming

### From an academic and programming learning perspective, the above tools offer limited value because:

### They are not built using low-level languages like C that require manual memory management.

### They do not demonstrate core concepts such as arrays, pointers, and linked lists in their implementation.

### The conversion history is usually maintained via databases or is session-based, and not via dynamically managed data structures like linked lists.

### These tools are not open to inspection or modification by students for learning purposes.

### 2.4 Need for a New System

### The primary aim of the proposed project is to design and implement a simple, educational currency converter system using the C programming language that:

### Stores a predefined list of currency exchange rates using arrays.

### Uses linked lists to maintain a dynamic record of all conversions performed during the session.

### Operates completely offline, without requiring internet access or external libraries.

### Is modular and transparent, so that each part of the code reflects specific concepts like structured programming, data structures, and memory handling in C.

### This system will serve as a practical tool for students to learn how to build real-world applications using core C programming principles, especially arrays and linked lists, which are foundational to understanding more complex data structures.

**CHAPTER 3: SOFTWARE REQUIREMENT ANALYSIS**

Software requirement analysis is the process of identifying and documenting the functional and non-functional needs of the software system. For the currency converter project developed in C language, this phase is crucial to define how the system should behave and what features it must provide. This project focuses on implementing arrays to store currency rates and a linked list to manage conversion history.

* **3.1 Functional requirements**

Functional requirements define the operations and services the system should perform. The following are the key functional requirements of the proposed system:

1. Currency input and conversion

The system should allow the user to enter a source currency, a target currency, and the amount to be converted. It must perform the conversion using the predefined exchange rates stored in an array.

1. **Storage of currency rates**

A static array will be used to store the names or codes of currencies along with their exchange rates relative to a base currency (e.g., INR).

1. **Recording of conversion history**

Each successful currency conversion will be recorded in a linked list. Each node in the list will contain details like source currency, target currency, amount, and result.

1. **Displaying conversion history**

The user should be able to view all previous conversions during the current session. The linked list will be traversed to display this information.

1. **User interface**

A menu-driven interface will be provided to allow the user to navigate the program easily—perform conversions, view history, or exit the program.

* **3.2 Non-functional requirements**

Non-functional requirements describe how the system should perform and define its quality attributes.

1. **Usability**  
   The system should be user-friendly and intuitive. All prompts and messages must be clear and informative.
2. **Reliability**  
   The application must handle invalid inputs gracefully, such as unsupported currencies or negative values, and display appropriate error messages without crashing.
3. **Efficiency**  
   The program should be lightweight and should execute quickly without unnecessary memory usage. Memory used by linked lists should be freed properly before program termination.
4. **Portability**  
   The program should be able to run on multiple operating systems, including Windows and Linux, as long as a C compiler is available.
5. **Maintainability**  
   The code should be modular and well-documented to make future changes, debugging, or enhancements easier.

* **3.3 Software and hardware requirements**

1. **Software requirements:**

* C language compiler (e.g., GCC, Turbo C)
* Text editor or IDE (e.g., Visual Studio Code, Code::Blocks)
* Operating system: Windows or Linux

1. **Hardware requirements:**

* Minimum 1 GB RAM
* Minimum 100 MB of disk space
* Standard input/output devices (keyboard and monitor)
* **3.4 Assumptions and constraints**
* Currency rates are static and hardcoded; no internet connection or real-time updates are included.
* Only a limited number of currencies are supported based on the array size.
* Conversion history is stored only during the session and is not written to a file.
* The system will be a console-based application, without graphical interfaces.

### CHAPTER4:SOFTWARE DESIGN

### 4.1 Introduction

Software design is the process of transforming user requirements into a suitable form that helps in the implementation of the system. In this project, the software design outlines how the currency converter system will function internally. It includes the control flow, the use of arrays for storing currency rates, and the linked list for managing conversion history.

### 4.2 Control Flow Diagram

+---------------------+

| Start Program |

+---------------------+

|

v

+---------------------------+

| Display Main Menu |

+---------------------------+

|

v

+-----------------------------+

| User selects an option: |

| 1. Convert Currency |

| 2. View Conversion History |

| 3. Exit |

+-----------------------------+

|

+-------------+--------------+

| |

v v

+------------------+ +------------------------+

| Convert Currency| | View Conversion History|

+------------------+ +------------------------+

| |

v v

+-------------------------+ +---------------------------+

| Get source, target, | | Traverse linked list and |

| and amount | | display each conversion |

+-------------------------+ +---------------------------+

| |

v |

+-----------------------------+ |

| Find rates from array | |

+-----------------------------+ |

| |

v |

+-----------------------------+ |

| Calculate converted amount | |

+-----------------------------+ |

| |

v |

+------------------------------+ |

| Store in linked list (node) | |

+------------------------------+ |

| |

v |

+------------------------------+ |

| Display converted result | |

+------------------------------+ |

| |

+------------+---------------+

|

v

+--------------------------+

| Back to Main Menu loop |

+--------------------------+

|

v

+------------------+

| Exit? |

+------------------+

|

+------------+-------------+

| |

No Yes

| |

v v

Go to Main Menu +-----------------------+

| Free linked list memory|

| and Exit Program |

+-----------------------+

**CHAPTER 5: PROPOSED SYSTEM**

**5.1 Introduction**

The proposed system is a console-based currency converter application developed in the C programming language. It is designed for educational purposes, demonstrating the use of arrays to store static currency exchange rates and linked lists to dynamically maintain a record of conversion history during runtime. Unlike existing web-based or app-based systems, this project runs offline and is ideal for learning fundamental programming concepts such as data structures, user input handling, and dynamic memory allocation.

**5.2 Objectives**

* To convert an entered amount from one currency to another using predefined static exchange rates.
* To record each conversion's details (source currency, target currency, amount, and result) using a linked list.
* To display all past conversions performed during the session.
* To create a modular, maintainable, and efficient C program that utilizes core programming constructs.

**5.3 System Architecture**

The system consists of the following components/modules:

**1. Currency Rate Module (Array)**

* A one-dimensional array or a structure-array combination is used to store currency codes and their corresponding exchange rates relative to a base currency (e.g., INR).
* This allows quick access to the rates during conversion.

**2. Conversion Module**

* Accepts input from the user: source currency, target currency, and amount.
* Searches the array for both currency rates.
* Applies the conversion formula:

Converted Amount=(Amount×Target RateSource Rate)\text{Converted Amount} = \left( \frac{\text{Amount} \times \text{Target Rate}}{\text{Source Rate}} \right)

**3. History Module (Linked List)**

* Each conversion is saved as a node in a singly linked list.
* Nodes contain source currency, target currency, original amount, and converted result.
* Allows dynamic memory usage as the number of conversions is not fixed.

**4. Menu and Interface Module**

* Displays options to the user: Convert Currency, View History, or Exit.
* Controls the flow between modules based on the user’s selection.

**5.4 Features of the Proposed System**

* Static exchange rate storage using arrays.
* Dynamic history tracking using linked lists.
* Menu-driven interface for easy navigation.
* Input validation and error handling.
* Memory cleanup at the end of the program.
* Offline execution (no API or internet dependency).
* **5.5 Advantages of the Proposed System**
* Demonstrates both static and dynamic data structures.
* Provides hands-on experience with memory management in C.
* Easy to understand and extend for academic or learning purposes.
* Runs on any system with a basic C compiler, no external tools required.

### CHAPTER 6: CODING

### #include <stdio.h>

### #include <stdlib.h>

### #include <string.h>

### // Structure for currency rate

### typedef struct {

### char code[4]; // Currency code like USD, INR, EUR

### float rate; // Exchange rate relative to base currency

### } Currency;

### // Linked list node for conversion history

### typedef struct HistoryNode {

### char from[4];

### char to[4];

### float amount;

### float result;

### struct HistoryNode\* next;

### } HistoryNode;

### #define NUM\_CURRENCIES 5

### Currency currencies[NUM\_CURRENCIES] = {

### {"INR", 1.0},

### {"USD", 0.012},

### {"EUR", 0.011},

### {"GBP", 0.0095},

### {"JPY", 1.75}

### };

### HistoryNode\* historyHead = NULL;

### // Function to find the exchange rate of a currency code

### float getRate(char code[]) {

### for (int i = 0; i< NUM\_CURRENCIES; i++) {

### if (strcmp(currencies[i].code, code) == 0) {

### return currencies[i].rate;

### }

### }

### return -1.0; // Error code for currency not found

### }

### // Function to add a conversion to history

### void addToHistory(char from[], char to[], float amount, floatresult) {

### HistoryNode\*newNode =(HistoryNode\*)malloc(sizeof(HistoryNode));

### strcpy(newNode->from, from);

### strcpy(newNode->to, to);

### newNode->amount = amount;

### newNode->result = result;

### newNode->next = historyHead;

### historyHead = newNode;

### }

### // Function to display history

### void displayHistory() {

### HistoryNode\* temp = historyHead;

### if (temp == NULL) {

### printf("No conversions done yet.\n");

### return;

### }

### printf("\n--- Conversion History ---\n");

### while (temp != NULL) {

### printf("%.2f %s => %.2f %s\n", temp->amount, temp>from, temp->result, temp->to);

### temp = temp->next;

### }

### printf("---------------------------\n");

### }

### // Function to perform currency conversion

### void convertCurrency() {

### char from[4], to[4];

### float amount, result;

### printf("Enter source currency code (e.g., INR): ");

### scanf("%s", from);

### printf("Enter target currency code (e.g., USD): ");

### scanf("%s", to);

### printf("Enter amount: ");

### scanf("%f", &amount);

### float fromRate = getRate(from);

### float toRate = getRate(to);

### if (fromRate == -1 || toRate == -1) {

### printf("Invalid currency code entered.\n");

### return;

### }

### result = (amount \* toRate) / fromRate;

### printf("Converted Amount: %.2f %s\n", result, to);

### addToHistory(from, to, amount, result);

### }

### // Function to free linked list memory

### void freeHistory() {

### HistoryNode\* temp;

### while (historyHead != NULL) {

### temp = historyHead;

### historyHead = historyHead->next;

### free(temp);

### }

### }

### int main() {

### int choice;

### do {

### printf("\n--- Currency Converter ---\n");

### printf("1. Convert Currency\n");

### printf("2. View History\n");

### printf("3. Exit\n");

### printf("Enter your choice: ");

### scanf("%d", &choice);

### switch (choice) {

### case 1: convertCurrency(); break;

### case 2: displayHistory(); break;

### case 3: freeHistory(); printf("Exiting...\n"); break;

### default: printf("Invalid choice. Try again.\n");

### }

### } while (choice != 3);

### return 0;

### }

**CHAPTER 7: OUTPUT SCREENS / RESULTS**

A screenshot of a computer

AI-generated content may be incorrect.

### CHAPTER 8: CONCLUSION

### The currency converter project demonstrates how real-time currency conversion can be implemented using Python. The tool can be extended with more features such as historical data tracking, graphs, and multilingual support.

### 5.1 Future Enhancements

### Add support for historical exchange rate charts

### Integrate voice input functionality

### Enhance UI with theme options

### Table 5.1: Future Enhancement Ideas

| Feature | Benefit |
| --- | --- |
| Historical Charts | Analyse currency trends |
| Voice Commands | Hands-free operation |
| Multi-language Interface | Better accessibility for users |