**Project Report**

**Project Name:** Calculator-X (Advanced Calculator)

**University Name:** FAST-NUCES, Karachi

**Course Code:** Bushra Sattar

**Date of Submission:** December 6, 2024

**Project Team**

* **Leader**: Muhammad Raza Mustafa (Roll No: 24k-1017, Section: BSC-1H)
* **Member 1**: Muhib Ahmed Khan (Roll No: 24k-0505, Section: BSC-1H)
* **Member 2**: Abdullah Razzaq (Roll No: 24k-0691, Section: BSC-1H)

**Introduction**

The **Calculator-X** project is an advanced computational tool designed to perform a wide range of mathematical operations and data manipulations efficiently and accurately. The project combines **innovative features**, **user-friendly design**, and **modular programming** to create a robust solution for various computational needs.

This project stands out due to its capability to handle large-scale operations, real-time input validation, and dynamic navigation. The implementation emphasizes **scalability**, **maintainability**, and **ease of use**, reflecting the team’s dedication and technical expertise.

The primary goals of **Calculator-X** are:

1. **Showcase advanced programming techniques** to enhance efficiency and modularity.
2. **Develop a versatile, user-friendly tool** for diverse computational tasks.
3. **Ensure scalability and maintainability**, allowing for future upgrades and easy expansion.

**The Idea Behind:**

The core idea behind **Calculator-X** was to create a single platform that simulates various types of calculators, all in one tool. Instead of relying on a GUI, we focused on developing a **command-line interface (CLI)** that offers a rich and interactive experience. Our primary goal was to maintain the simplicity and efficiency of a CLI while enhancing its functionality and user engagement.

We aimed to provide an all-encompassing tool that can handle a wide range of calculations, from basic arithmetic to more complex operations, such as large factorials and large-digit multiplications. Despite the limitations of a CLI, we ensured that each feature was well-organized, efficient, and easy to use.

By focusing on a clean, intuitive interface and implementing dynamic input validation, **Calculator-X** delivers enhanced user experience. The result is a powerful and fully functional calculator that pushes the boundaries of what is possible within the constraints of a command-line environment.

**User Interface:**

The User Interface (UI) of **Calculator-X** is thoughtfully designed to deliver seamless and engaging user experience. Despite being a command-line application, its intuitive navigation, clear structure, and dynamic feedback make it easy and efficient to use.

1. **Arrow-Based Navigation**

* The menu utilizes **up and down arrow keys** for changing the selection.
* Users can enter a selected option by pressing the **Enter key** or the **right arrow key**.
* To go back to the previous menu, the **left arrow key** is used. However, when performing certain calculations, such as addition or subtraction, users must complete at least one calculation before returning, as arrow keys are temporarily disabled during the process.
* This navigation system eliminates the need for typing commands manually, making interactions faster and more intuitive.
* The system dynamically highlights the selected menu option, ensuring clarity and guiding the user effectively.

1. **Dynamic Input Validation**

In designing **Calculator-X**, we accounted for keyboards that include many unnecessary keys, which can potentially interfere with calculations. To address this, we implemented a robust **input validation feature** that ensures only the required values are accepted, depending on the operation. This includes validating:

* **Data types**: Integers, floats, binary, or hexadecimal etc. as needed.
* **Sign**: Positive or negative numbers.

**Error Feedback**



* Incorrect inputs turn the prompt **red**, immediately signaling the error.
* Any invalid input is **cleared entirely**, prompting the user to re-enter valid data.
* The system continues rejecting invalid inputs until correct data is entered.



**Success Feedback**

* Valid inputs are highlighted in **green**, providing clear visual confirmation of accuracy.



**Overflow Control**

* The system incorporates **overflow control**, which disables the keyboard after a specified number of digits (as determined by the user). This prevents unintentional input overflow and ensures the calculations remain within manageable limits.

1. **Color Coding**

The UI employs a **consistent color scheme** to differentiate between sections, inputs, and results. This makes the interface visually appealing and ensures that users can easily follow the workflow. For this reason, we have used 18 different colors.



1. **Menu Structure**

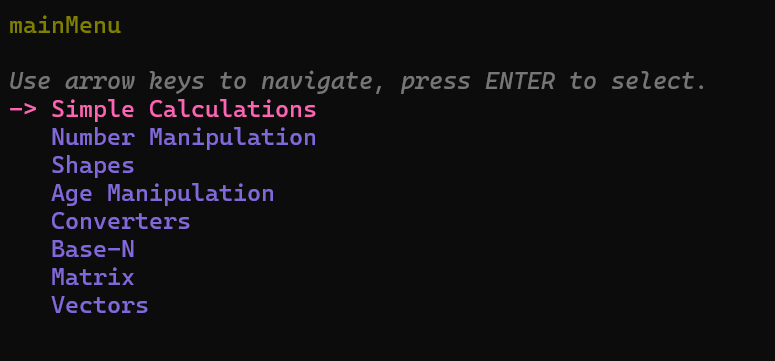
* A hierarchical menu system organizes all features, allowing users to navigate seamlessly to their desired functionality.
* Submenus are provided for specific sections, such as **unit conversions** and **shape calculations**, simplifying access to related operations.
* The **current directory location** is displayed at the top of the menu, assisting users in forward and backward navigation within the menu structure.

By combining these elements, the UI of **Calculator-X** offers a balance between functionality and usability, ensuring that users can perform even complex calculations with ease and confidence.

**Features:**

1. **Main Menu**

The **Main Menu** of **Calculator-X** is designed as a hierarchical system, offering intuitive navigation to access a diverse range of computational features. It organizes the following modules effectively:



Each module is implemented in a modular way, ensuring clear separation of concerns, with individual .c and .h files managing their functionalities.

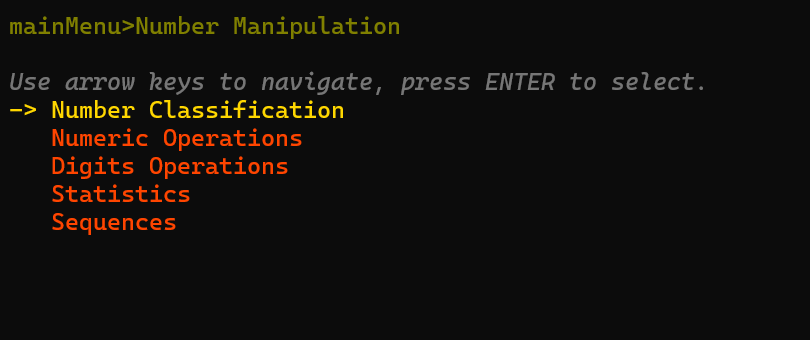
1. **Simple Calculator**

The **Simple Calculator** module, managed in **simpleCalculator.c** and **simpleCalculator.h,** provides essential arithmetic operations with added functionality:

* **Addition**: Handles integer and floating-point numbers seamlessly.
* **Subtraction**: Supports differences with precision, irrespective of input size.
* **Multiplication**: Capable of handling **20-digit by 20-digit multiplications**, ensuring precision and accuracy even for large numbers.
* **Division**: Includes robust validation to prevent division by zero. If the divisor is zero, the result is displayed as **"inf" (infinity)** for clarity.
* **Remainder (Modulus)**: Accepts only integers to maintain the integrity of the operation.
* **Power Function**: Computes powers up to **99**, accommodating most user needs for exponential calculations.
* **Square Root Function**: Computes roots up to **99**, maintaining precision for large and small numbers alike.

1. **Number Manipulation**

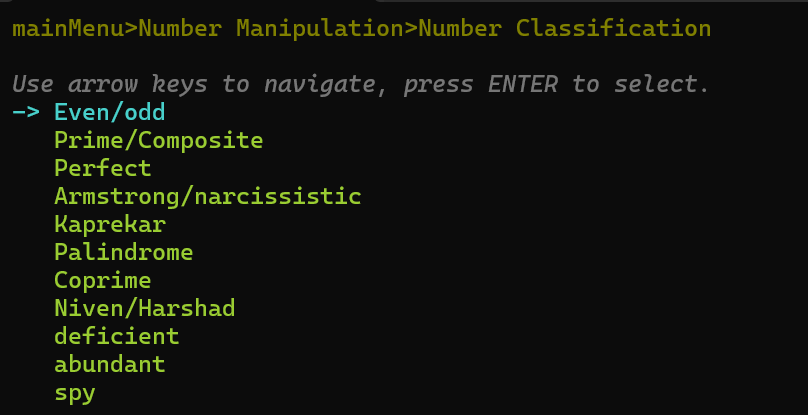
The **Number Manipulation** module is a versatile section of **Calculator-X**, containing five subcategories:



This module is implemented and managed under **numberManipulation.c** and **numberManipulation.h**, ensuring a clean and organized codebase.

**1. Number Classification**

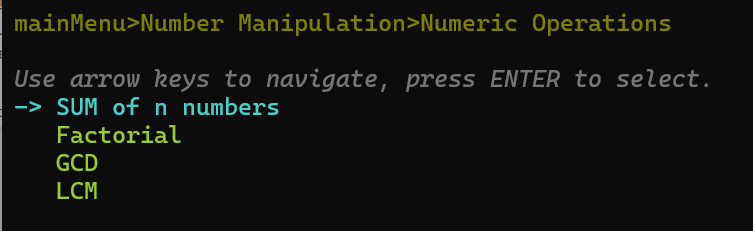
The **Number Classification** subcategory identifies different types of numbers, featuring **11 distinct number identifiers**:



For user convenience, the module also includes **definitions** of each type, ensuring clarity and usability for users who may not be familiar with these terms.

**2. Numeric Operations**

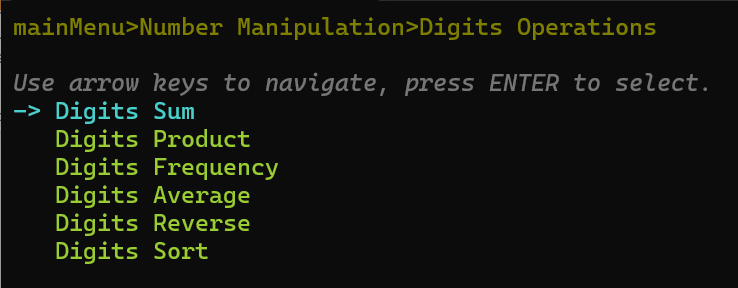
This subcategory supports four types of operations:



* **Sum of N Numbers**: Capable of calculating the sum of **99 distinct 10-digit numbers**, offering exceptional scalability for large datasets.
* **Factorial**: Computes factorials up to **999!** Utilizing optimized algorithms to handle large numbers without performance degradation.
* **GCD (Greatest Common Divisor)**: Efficiently finds the largest divisor common to two integers.
* **LCM (Least Common Multiple)**: Determines the smallest multiple common to two integers, with precision.

**3. Digit Operations**

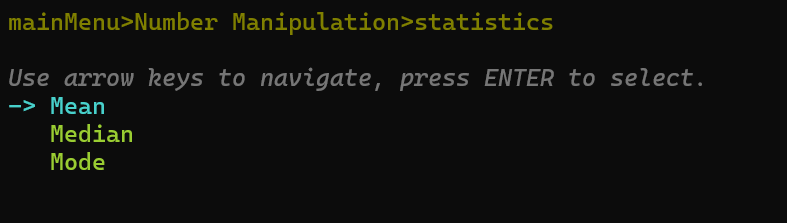
The **Digit Operations** subcategory offers six functionalities and can handle numbers up to **115 digits long**, ensuring support for extensive computations:



* **Count Digits**: Counts the total number of digits in a given number.
* **Sum of Digits**: Calculates the sum of all digits in the number.
* **Product of Digits**: Computes the product of all digits in number.
* **Reverse Digits**: Outputs the number with its digits reversed.
* **Sort Digits in Ascending Order**: Rearranges the digits of the number in **ascending order**, providing clarity for digit-based analysis.
* **Digit Frequency**: Analyzes and displays the **frequency of each digit (0–9)** within the number, offering detailed insights into its composition.

**4. Statistics**

The **Statistics** subcategory efficiently manages operations for up to **99 distinct 10-digit numbers**, making it highly scalable for large datasets. It includes:

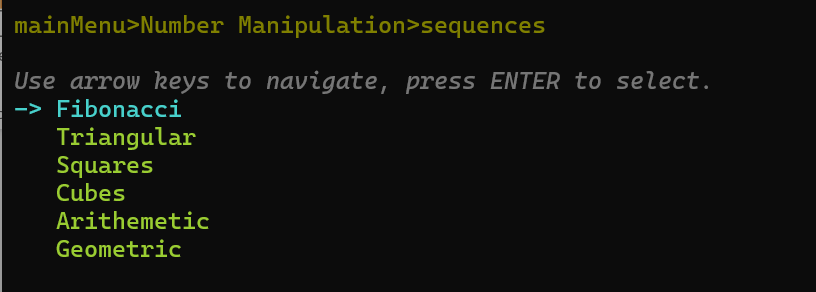


* **Mean**: Calculates the average of the input dataset.
* **Median**: Determines the middle value in the dataset.
* **Mode**: Identifies the most frequently occurring number(s) in the dataset.

These functionalities are optimized for accuracy and performance, even with extensive numerical inputs.

**5. Sequences**

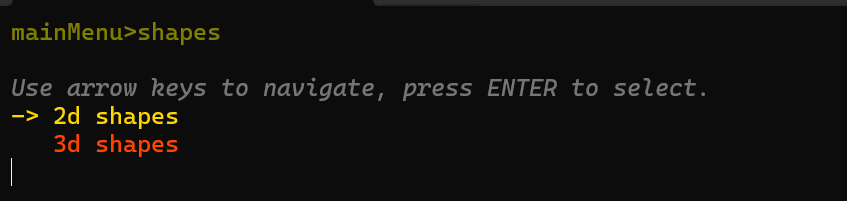
The **Sequences** subcategory features six types of number series:



* The first 4 sequences (**Fibonacci**, **Triangular**, **Squares, and Cubes**) can generate up to **999 terms**, ensuring a comprehensive output for mathematical and analytical purposes.
* The remaining sequences (**Arithmetic and Geometrics**) are implemented to handle numbers efficiently and display precise results tailored to user inputs.

1. **Shapes**

The **Shapes** module of **Calculator-X** is designed to perform advanced geometric calculations and visualizations. It is divided into two subcategories:



The module is implemented across **four files**:

* **shapes.c and shapes.h**: Handle the implementation of shape-related calculations.
* **shapesIllustration.c and shapesIllustration.h**: Focus on illustrating the shapes to enhance user comprehension.

This modular approach ensures clarity and maintainability of the code.

**1. 2D Shapes**

**A screenshot of a computer

Description automatically generated**The **2D Shapes** subcategory features a total of **14 shapes**, offering a comprehensive range of geometric computations. The supported shapes are:

**Features for 2D Shapes**

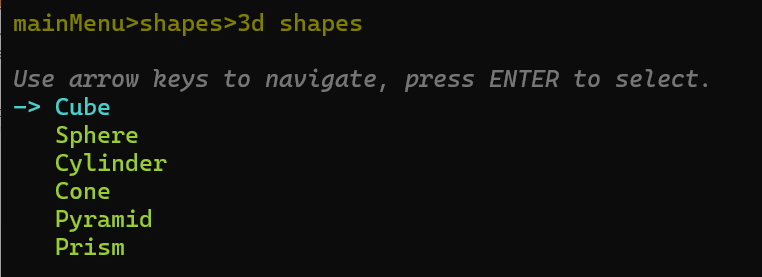
The **2D Shapes** section includes a wide variety of geometric shapes, offering a comprehensive set of functionalities for each.

1. **Circle**
   * Area
   * Circumference (Perimeter)
2. **Square**
   * Area
   * Perimeter
   * Diagonal
3. **Rectangle**
   * Area
   * Perimeter
   * Diagonal
4. **Triangle**
   * Area
   * Perimeter
   * Semi-Perimeter
   * Inradius
   * Circumradius
5. **Trapezoid**
   * Area
   * Perimeter
6. **Parallelogram**
   * Area
   * Perimeter
7. **Rhombus**
   * Area
   * Perimeter
8. **Kite**
   * Area
   * Perimeter
   * Diagonals
9. **Dart**
   * Area
   * Perimeter
   * Diagonals
10. **Ellipse**
    * Area
    * Approximate Perimeter
    * Perimeter
    * Apothem
11. **Sector of a Circle**
    * Area
    * Arc Length
    * Chord Length
12. **Segment of a Circle**
    * Area
    * Arc Length
    * Chord Length
13. **Annulus (Ring)**
    * Area
    * Circumference
14. **Regular Polygon**
    * Area
    * Perimeter
    * Apothem
    * Interior Angle
    * Exterior Angle

The illustration module (shapesIllustration.c) helps visualize these shapes, adding clarity to the calculations.

**2. 3D Shapes**

The **3D Shapes** subcategory supports **6 shapes**, covering essential solid figures used in various fields. The supported shapes are:

****

**Features for 3D Shapes**

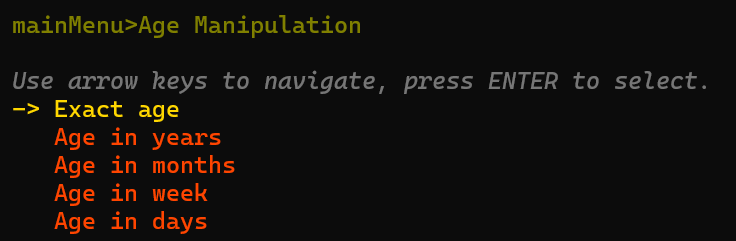
The **3D Shapes** section handles various solid figures, offering calculations for their surface areas and volumes.

1. **Cube**
   * Surface Area
   * Volume
2. **Sphere**
   * Surface Area
   * Volume
3. **Cylinder**
   * Lateral Surface Area
   * Total Surface Area
   * Volume
4. **Cone**
   * Lateral Surface Area
   * Total Surface Area
   * Volume
5. **Pyramid**
   * Lateral Surface Area
   * Total Surface Area
   * Volume (specific types, e.g., Square, Triangular)
6. **Prism**
   * Lateral Surface Area
   * Total Surface Area
   * Volume (specific types, e.g., Rectangular, Triangular)

The **Shapes Illustration Module** provides detailed visual representations of the shapes, making the calculations more intuitive and user-friendly.

1. **Age Manipulation**

The **Age Manipulation** module in Calculator-X is a powerful tool with five distinct features. This module is managed by ageManipulation.c and ageManipulation.

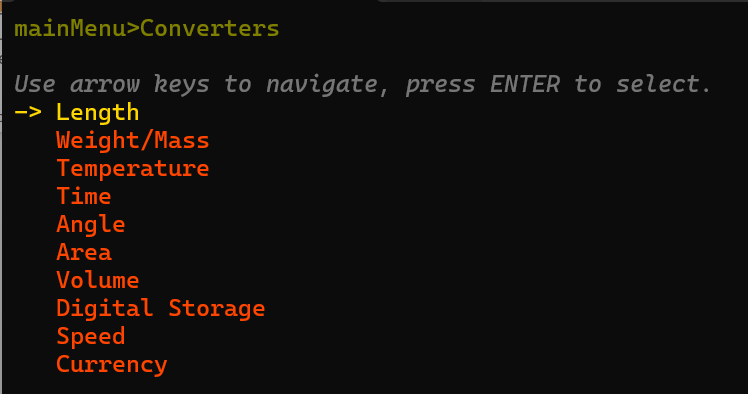


**Key Functionalities**

* **Real-Time Date Integration**:  
  This module works with the current system date in real-time, ensuring accurate calculations without requiring manual date inputs except for the user’s date of birth.
* **Validation Checker**:  
  It includes a robust validation system to prevent users from entering a date later than today's date, ensuring logical and accurate inputs.
* **Accurate Age Calculation**:  
  The Exact Age Calculator accounts for every detail, such as leap years, variable month lengths, and time zones, to provide a precise calculation of the user’s age down to the day.

1. **Converters**

The **Converters** module in Calculator-X is a versatile feature that allows users to seamlessly convert between different units. It supports 10 types of unit converters:



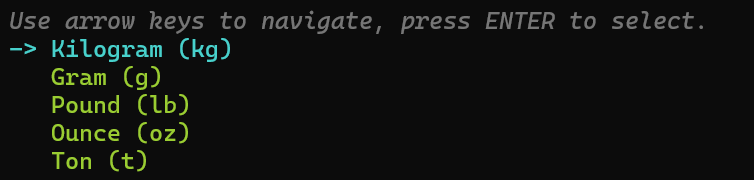
1. **Length Converter**

**A black screen with white text

Description automatically generated**The Length Converter supports 6 units, and all units are interconvertible:

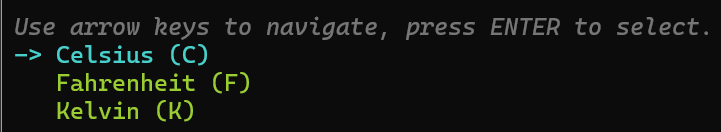
1. **Weight Converter**

The Weight Converter supports 5 units, ensuring comprehensive coverage for various applications:

****

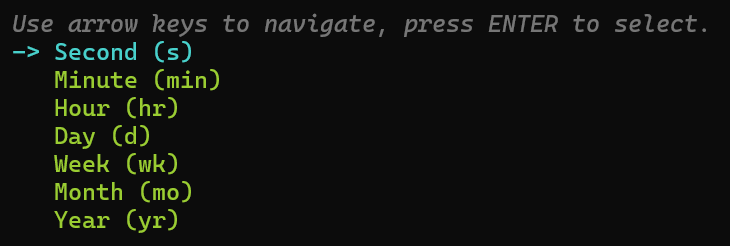
1. **Temperature Converter**

The Temperature Converter supports 3 widely used temperature scales:

****

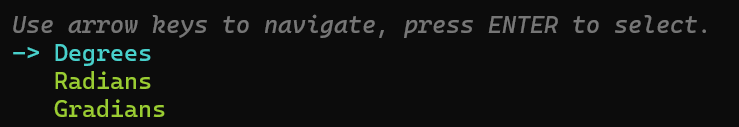
1. **Time Converter**

The Time Converter offers 7 units for converting time measurements:

****

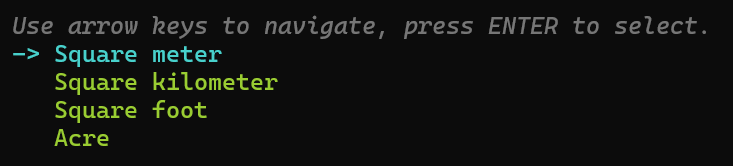
1. **Angle Converter**

The Angle Converter supports 3 widely used angle units:

****

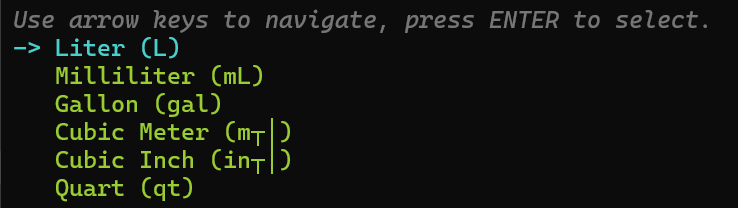
1. **Area Converter**

The Area Converter provides 4 units for converting surface areas:

****

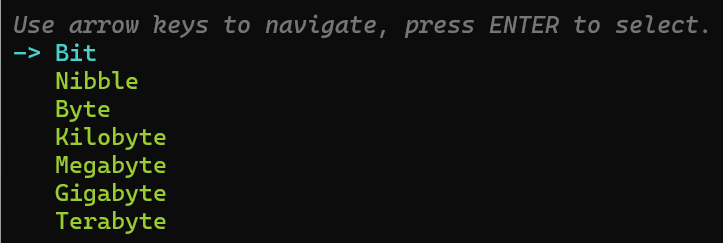
1. **Volume Converter**

The Volume Converter handles 6 units to accommodate various volumetric measurements:

****

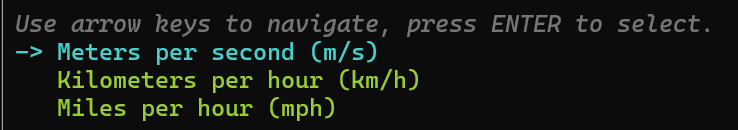
1. **Digital Storage Converter**

The Digital Storage Converter supports 7 units commonly used in computing:

****

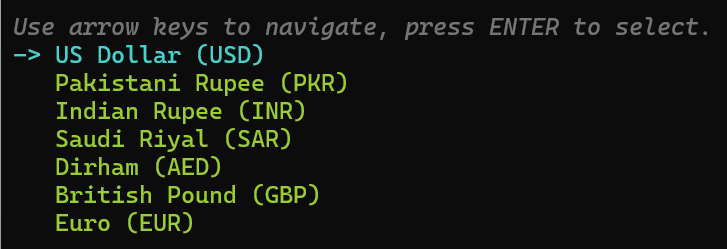
1. **Speed Converter**

The Speed Converter supports 3 units for speed conversions:



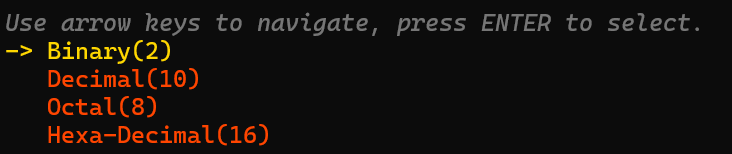
1. **Currency Converter**

The Currency Converter supports 7 major currencies for financial transactions and exchange rates. But it is not live.



1. **Base-N Converter**

The **Base-N Converter** module offers seamless interconversion between four commonly used numeral systems:



It is implemented in **baseN.c** and **baseN.h**, ensuring a modular and organized structure for efficient computation.

This module provides robust functionality by supporting numbers with up to **20 digits**, accommodating a wide range of numerical values and use cases.

**Validation and Error Handling**

To ensure accurate results, the module includes a validation mechanism to prevent the entry of invalid characters for the selected numeral system. For example:

* **Binary** allows only 0 and 1.
* **Octal** accepts digits from 0 to 7.
* **Decimal** permits numbers are from 0 to 9.
* **Hexadecimal** accepts 0-9 and A-F.

**Accuracy in Conversion**

The **Base-N Converter** guarantees precision in all conversions by adhering to strict algorithmic checks and transformations. Each conversion is carried out systematically, ensuring no loss of data or rounding errors, even with the maximum **20-digit** input.

**Development Approach**

1. **Modular Design**

The project strictly adheres to modular programming principles, where each feature is encapsulated in separate .c and .h files. This ensures:

* **Scalability**: New features can be integrated effortlessly without disrupting the existing structure.
* **Readability**: Each module is self-contained, making the codebase easier to debug and comprehend.
* **Maintainability**: Bug fixes or updates can be applied to individual modules without impacting others.

1. **File Organization**

The project maintains a well-structured file hierarchy for better manageability:

* **Core Functionalities**:
  + **simpleCalculator.c:** Handles arithmetic operations.
  + **numberManipulation.c:** Implements number-based tools and utilities.
  + **matrixOperations.c:** Manages matrix-related computations.
  + **vectors.c:** Manages vector related operations
* **Utilities**:
  + **inputValidator:** Ensures accurate inputs and provides real-time error feedback.
  + **list.c:** Implements arrow-based menu navigation and colors for an intuitive user interface.
  + **shapesIllustator.c:** For shapes illustration
* **Specialized Features**:
  + **shapes.c:** Handles computations for 2D and 3D shapes.
  + **baseN.c:** Implements numeral system conversions.
  + **ageManipulation.c:** Manages age-related calculations.
  + **converters.c:** For unit conversions

1. **Build System**

The project leverages a **makefile** for seamless compilation and linking of all modules. The makefile facilitates:

* **Streamlined Builds**: Automates the compilation process.
* **Efficient Testing**: Reduces manual efforts during development and testing cycles.

1. **UI and Code Presentation**

Just as we have focused on creating an intuitive and user-friendly interface, equal emphasis has been placed on **code presentation**:

* **Comprehensive Comments**: The codebase includes detailed comments, guiding developers and maintainers through each segment.
* **Relatable Names**: Variables and functions are named using clear, descriptive terms to enhance code readability.
* **Modifiability**: Thanks to the modular structure, the code can be easily extended or adjusted for future requirements.

1. **Efficiency and Optimization**

* **Recursive Approach**: Recursive methods ensure elegant and concise implementations.
* **Performance**: Fast and efficient algorithms have been chosen to handle each calculation, ensuring optimal runtime even for large inputs.
* **Use of header files**: Use of same name header files make code more efficient
* **Scalability**: The code structure supports complex operations (e.g., handling up to 20-digit numbers) without compromising accuracy or performance.

1. **Key Takeaway**

The combination of a **user-focused UI**, a **modular structure**, and **efficient algorithms** ensures that the project not only meets but exceeds the standards of functionality, maintainability, and performance.

**Achievements**

The **Calculator-X** project is a reflection of the team’s dedication, technical expertise, and innovative approach. Key achievements include:

* **Comprehensive Modular Design**: Seamlessly integrating multiple features, each managed through well-structured .c and .h files.
* **Advanced Computational Capabilities**: Successfully implementing operations like **999! factorial**, **20-digit by 20-digit multiplications**, and handling **115-digit numbers** with precision.
* **Dynamic and User-Centric UI**: Providing real-time feedback with dynamic input validation, error handling, and an intuitive CLI interface enhanced by arrow-key navigation.
* **Efficient Algorithms**: Employing optimized and recursive approaches for faster computations across diverse functionalities.
* **Versatility in Features**: Offering a wide array of tools, from unit converters to shape calculators, all within a single cohesive platform.
* **Precision and Reliability**: Consistently delivering accurate results without any deviation, even when compared with physical or online calculators.

**Comparison**

**Calculator-X** has been rigorously tested against both **online calculators** and **physical calculators**. Not only has it matched their precision in every case, but it has also surpassed them in certain scenarios.

**Advantages over Online Calculators**:

* **Offline Functionality**: Unlike online calculators, **Calculator-X** does not require an internet connection, ensuring accessibility anytime and anywhere.
* **Comprehensive Features**: It integrates numerous functionalities—such as number manipulation, unit conversions, and geometric calculations—into a single platform, which is often unavailable in online tools.

**Advantages over Physical Calculators**:

* **Enhanced Capabilities**: Supports large-scale computations (e.g., 999! factorial, 20-digit numbers), which physical calculators cannot handle.
* **Dynamic Feedback**: Provides real-time error handling and input validation, features absent in most physical calculators.
* **Better Organization**: Offers structured menus, making complex operations more accessible and intuitive.

By excelling in precision, functionality, and user experience, **Calculator-X** proves to be a superior alternative to both online and physical calculators.

**Future Work**

Looking ahead, the **Calculator-X** project aims to expand its functionality and enhance user experience with the following additions:

1. **Filing System**:
   * Implement a feature for storing the calculation history, allowing users to review previous results.
2. **Reuse of Previous Results**:
   * Allow users to use previously calculated answers directly in new operations, streamlining workflows.
3. **Live Currency Conversion**:
   * Integrate a live API for real-time currency exchange rates, enhancing the currency converter module.
4. **Additional Converters**:
   * Expand the list of converters to include categories like:
     + Force (e.g., Newtons, Dynes)
     + Torque (e.g., Newton-meters, Pound-feet)
     + Electrical units (e.g., Volts, Amperes, Ohms).
   * Add more units in existing converters
5. **Expanded Number Manipulation**:
   * Add support for identifying more number types (e.g. Evil numbers).
6. **Enhanced Sequences**:
   * Introduce new sequences such as Lucas Numbers, Catalan Numbers, or Pascal’s Triangle computations.
7. **Advanced Features**:
   * Add modules for polynomial equation solving.
   * Introduce tools like a **Health Calculator** (e.g., BMI, calorie tracker).
8. **Improved Graphics**:
   * Enhance shape illustrations with more detailed visuals and step-by-step construction guides.
9. **Statistical Tools**:
   * Incorporate advanced statistical operations like variance, standard deviation, and regression analysis.
10. **Support for Multivariable Functions**:
    * Develop tools to solve equations or perform operations involving multiple variables.
11. **Transition to C++**
    * Transform the project from **C to C++** to leverage the advantages of **object-oriented programming (OOP)**. This shift will enable more powerful computations, better code organization, and enhanced scalability for future development.

These future enhancements will make **Calculator-X** an even more powerful, versatile, and indispensable computational tool.

**Conclusion**

The **Calculator-X** project successfully achieves its objectives of creating an advanced, feature-rich, and user-friendly calculator. By integrating modular programming principles, efficient algorithms, and a dynamic CLI-based user interface, the project sets a high standard for computational tools.

This project not only fulfills its intended scope but also showcases the potential for future growth. It highlights the power of structured design and technical rigor in building scalable and reliable software solutions. With its adaptability and maintainability, **Calculator-X** is a step forward in computational tool innovation.