University of Engineering and Technology (UET) Mardan

Department of Computer Science - Section A

Semester 1

Number System Converter and Calculator

Language: C

Type: Console-based Application

Team Members:

- **Ubaid Ullah** (Reg No. 24MDBCS0553)
- **Haris Bashir** (Reg No. 24MDBCS0544)
- **Muhammad Awais Ghani** (Reg No. 24MDBCS0565)

Date: 4/1/2025

---

1. Introduction

This project focuses on creating a **Number System Converter and Calculator** that can convert between different number systems (decimal, binary, octal, and hexadecimal) and perform basic arithmetic operations such as addition, subtraction, multiplication, division, power, square root, and modulo. The project is implemented in the C programming language and aims to provide a useful tool for both learning and practical applications related to number systems and mathematical operations.

---

2. Project Objective

The goal of this project is to develop a program that:

1. Converts numbers between different number systems (decimal, binary, octal, and hexadecimal).
2. Performs basic arithmetic operations such as addition, subtraction, multiplication, and division.
3. Handles advanced operations like exponentiation, square roots, and modulo calculations.
4. Provides a user-friendly, text-based interface with input validation to prevent errors during runtime.

3. Key Features and Functions

The program will include the following features implemented as functions in C:

1. **Menu Display:**
    - Displays a menu with options for number system conversions and arithmetic operations.
2. **Number System Conversion:**
    - Converts decimal numbers to binary, octal, and hexadecimal formats.
    - Converts binary, octal, and hexadecimal numbers back to decimal.
3. **Arithmetic Operations:**
    - Supports basic arithmetic operations such as addition, subtraction, multiplication, and division.
    - Handles advanced operations like power, square root, and modulo calculations.
4. **Input Validation:**
    - Ensures that the user inputs valid numbers for each operation and conversion.
    - Handles invalid inputs gracefully without causing the program to crash.
5. **Error Handling:**
    - Prevents issues like division by zero or invalid number system digits.
6. **Exit Option:**
    - Allows the user to exit the program or go back to the main menu after performing operations.

---

4. Algorithm Details

Main Program Logic:

1. **Start Program:**
    - The program begins by displaying a welcome message and the main menu.
2. **User Input:**
    - The program prompts the user to enter a choice from the main menu.
    - Depending on the user's choice, the appropriate submenu for conversions or arithmetic operations is displayed.

3. **Perform Conversion:**
    - If the user selects a conversion, the program asks for the number and the system to convert to/from (e.g., decimal to binary).
    - The program performs the conversion and displays the result.
4. **Perform Arithmetic:**
    - If the user selects arithmetic operations, the program asks for two numbers and performs the operation (e.g., addition or multiplication).
5. **Error Handling:**
    - The program checks for invalid inputs and handles errors like division by zero or invalid number system digits.
6. **Repeat or Exit:**
    - After completing an operation, the user is given the option to return to the main menu or exit the program.

Conversion Logic:

1. **Input:**
    - The program prompts the user to input a number in a specified format (e.g., decimal or binary).
2. **Fetch:**
    - The number is fetched from the user input and validated.
3. **Conversion:**
    - The appropriate conversion algorithm is applied based on the user's choice (e.g., converting a decimal to binary by repeatedly dividing the number by 2).
4. **Output:**
    - The result of the conversion is displayed to the user.

---

5. Tools and Technologies

- **Programming Language:** C
- **File Handling:** Not used in this project, as the program only interacts with the user via the console.

---

6. Programming Concepts Used

During the implementation of this project, several key programming concepts were applied to build a functional and user-friendly application:

1. **Functions**:
    - We used functions to modularize the program. Each operation (conversion or arithmetic) is encapsulated in its own function, making the program more organized and easier to debug.
2. **Conditionals (if-else and switch statements)**:
    - Conditional statements allowed us to direct the flow of the program based on user input. For example, based on the user's choice of conversion or arithmetic operation, the appropriate function is called.
3. **Loops (while and for loops)**:
    - We used loops to repeatedly ask the user for input and process the conversion (e.g., repeatedly dividing a number by 2 for binary conversion) and arithmetic operations.
4. **Arrays**:
    - Arrays were used to store the results of number system conversions (e.g., binary numbers). This was particularly useful for binary and octal conversions, where numbers are built incrementally.
5. **String Handling**:
    - Functions like strlen () and scanf () were used to handle user input for number systems like binary, hexadecimal, and octal. This was necessary because user input in these formats is processed as strings before being converted.
6. **Mathematical Functions**:
    - The pow () and sqrt () functions were used for exponentiation and square root calculations, making it easier to implement these operations without manually coding the logic.
7. **Error Handling**:
    - We incorporated error handling using conditional checks to ensure that the user input is valid. For example, division by zero was checked, and invalid number system digits were handled gracefully.

---

7. What We Learned

1. **Understanding Number Systems**:
    - By implementing the conversions between different number systems (decimal, binary, octal, hexadecimal), we gained a deeper understanding of how these systems work and how to manipulate data at a low level.
2. **Function Decomposition**:
    - The importance of dividing a large problem into smaller, manageable functions became clear. Each function had a specific role, making the code easier to maintain and test.
3. **Input Validation and Error Handling**:
    - We learned how important it is to validate user input and handle errors properly. This ensures that the program behaves predictably and doesn't crash due to invalid input or unexpected errors.
4. **Modular Design**:
    - The project reinforced the importance of writing modular code that can be easily expanded or modified. If we wanted to add more number systems or arithmetic operations in the future, it would be relatively simple to do so without affecting other parts of the program.
5. **Practical Application of Mathematical Operations**:
    - Implementing complex mathematical operations like power and square root in C gave us practical experience with built-in functions like pow () and sqrt (), and understanding how to use them efficiently.

Conclusion

This project will provide a comprehensive tool for converting between number systems and performing basic arithmetic operations. The modular design ensures that the program is easy to expand or modify in the future.

Best Regards,

Semester 1, Section A

Department of Computer Science

UET Mardan