Number Conversion Program

# Introduction

## Purpose

The purpose of this document is to provide an overview of the design and functionality of a command-line program for converting numbers between different numerical bases (Decimal, Binary, Octal, and Hexadecimal).

## Scope

This program allows users to input numbers in any of the supported bases and convert them to any other supported base. It includes validation for input numbers and handles both integer and fractional conversions.

## Audience

This document is intended for developers and maintainers who need to understand the architecture, functionality, and usage of the number conversion program.

# System Overview

## System Architecture

The program is designed as a console-based application using C++. It employs a modular structure with functions for input handling, validation, conversion, and output.

## Functional Components

* **Input Handling:** Functions to read user inputs for number types (Decimal, Binary, Octal, Hexadecimal).
* **Validation:** Functions to validate the format and correctness of input numbers based on their respective bases.
* **Conversion:** Functions to convert numbers between Decimal, Binary, Octal, and Hexadecimal bases, handling both integer and fractional parts.
* **User Interface:** Console-based interface to interact with the user, displaying prompts, input fields, and conversion results.

## Non-functional Requirements

* **Performance:** Efficient handling of number conversions for typical use cases.
* **Usability:** Intuitive user interface with clear prompts and error messages.
* **Reliability:** Robust input validation to prevent program crashes due to invalid inputs.

# Detailed Design

## Modules and Functions

### ReadUserOption()

* + - * Purpose: Reads and validates the user's choice for number base conversion.
      * Input: None
      * Output: Returns the user's choice as an integer.

### IsDecimalNumber(string Number)

* + - * Purpose: Validates if a string represents a Decimal number.
      * Input: String representation of the number.
      * Output: Returns true if the input string is a valid Decimal number, otherwise false.

### IsBinaryNumber(string Number)

* + - * Purpose: Validates if a string represents a Binary number.
      * Input: String representation of the number.
      * Output: Returns true if the input string is a valid Binary number, otherwise false.

### IsOctalNumber(string Number)

* + - * Purpose: Validates if a string represents an Octal number.
      * Input: String representation of the number.
      * Output: Returns true if the input string is a valid Octal number, otherwise false.

### IsHexaDecimalNumber(string Number)

* + - * Purpose: Validates if a string represents a Hexadecimal number.
      * Input: String representation of the number.
      * Output: Returns true if the input string is a valid Hexadecimal number, otherwise false.

### IsValidInput(string Number, enNumberConversion ConversionType)

* + - * Purpose: Validates if a given string input is valid for a specified number base.
      * Input: String representation of the number, Enum specifying the number base.
      * Output: Returns true if the input string is valid for the specified base, otherwise false.

### ReadDecimalNumber(), ReadBinaryNumber(), ReadOctalNumber(), ReadHexaDecimalNumber()

* + - * Purpose: Functions to read user input for numbers in Decimal, Binary, Octal, and Hexadecimal formats respectively.
      * Input: None
      * Output: Returns a struct (stNumberThatWillConvert) containing the validated number in the specified base.

### ShowConversionOptions(string Message)

* + - * Purpose: Displays conversion options to the user and reads their choice.
      * Input: Message to display as a prompt.
      * Output: Returns the user's selected conversion type (enNumberConversion).

### Conversion Functions (DecimalToBinary, BinaryToDecimal, DecimalToHexadecimal, HexadecimalToDecimal, DecimalToOctal, OctalToDecimal)

* + - * Purpose: Convert numbers between different bases (Decimal, Binary, Octal, Hexadecimal).
      * Input: String representation of the number in one base.
      * Output: String representation of the number in the desired base.

### PrintResult(stOperationInfo OperationInfo)

* + - * Purpose: Displays the converted result to the user.
      * Input: Struct (stOperationInfo) containing conversion details and result.
      * Output: Prints the converted result to the console.

### PerformConversionOption(stOperationInfo OperationInfo)

* + - * Purpose: Orchestrates the conversion process by interacting with the user, performing conversions, and returning the result.
      * Input: Struct (stOperationInfo) containing conversion details.
      * Output: Returns updated struct (stOperationInfo) with conversion results.

### Start()

* + - * Purpose: Main function to start the program, handles the main loop for repeated conversions until the user chooses to exit.
      * Input: None
      * Output: None

# Implementation Details

## Programming Language

The program is implemented in C++.

## Libraries Used

* <iostream>: Input and output operations.
* <vector>, <string>: Data structures and string manipulation.
* <iomanip>, <cctype>, <cmath>: Formatting output, character operations, and mathematical functions.

## Error Handling

* Invalid inputs are handled with error messages and prompts for re-entry.
* Edge cases such as negative numbers, floating-point numbers, and non-numeric characters are validated and handled appropriately.

# User Interaction

## Input Requirements

Users are prompted to enter numbers in Decimal, Binary, Octal, or Hexadecimal formats. Invalid inputs prompt for re-entry.

## Output Format

Converted results are displayed in the chosen base format.

# Conclusion

## Summary

The number conversion program provides a robust solution for converting between different numerical bases with efficient validation and clear user interaction.

## Future Enhancements

* Support for additional numerical bases or custom base conversions.
* Graphical user interface (GUI) for improved usability.
* Performance optimizations for large number inputs.