Devbhasha Sample Programs Collection

This document contains a collection of sample programs written in Devbhasha, a programming language inspired by Indian mythology and culture. These examples demonstrate the various features and capabilities of the language.

Table of Contents

1. Hello World

```
// Basic Hello World program in Devbhasha

dharma HelloWorld {
    lekha("Namaste, World!");
}
samapti
```

2. Addition of Two Numbers

```
// Program to add two numbers input by the user

dharma Addition {
    // Variable declarations
    prakruti sankhya number1;
    prakruti sankhya number2;
    prakruti sankhya sum;

    // Get user input
    lekha("Enter first number: ");
    number1 = pathana();

    lekha("Enter second number: ");
    number2 = pathana();

    // Calculate sum
    sum = number1 + number2;

    // Display result
    lekha("Sum of " + number1 + " and " + number2 + " is: " + sum);
}
samapti
```

3. Basic Arithmetic Operations

```
// Program demonstrating basic arithmetic operations

dharma ArithmeticOperations {
    prakruti sankhya a = 20;
    prakruti sankhya b = 5;

    lekha("Arithmetic Operations on " + a + " and " + b);
    lekha("Addition: " + (a + b));
    lekha("Subtraction: " + (a - b));
    lekha("Multiplication: " + (a * b));
    lekha("Division: " + (a / b));
    lekha("Modulus: " + (a % b));

    // Increment and decrement operations
    prakruti sankhya c = a;
    c++;
    lekha("After increment: " + c);

    c--;
    lekha("After decrement: " + c);

    // Compound assignment operators
    prakruti sankhya d = 10;
    d += 5;  // d = d + 5
    lekha("After += 5: " + d);

    d -= 3;  // d = d - 3
    lekha("After -= 3: " + d);

    d *= 2;  // d = d * 2
    lekha("After *= 2: " + d);

    d /= 4;  // d = d / 4
    lekha("After /= 4: " + d);
}
samapti
```

4. Conditional Statements

```
// Program demonstrating conditional statements

dharma ConditionalStatements {
    prakruti sankhya age;

    lekha("Enter your age: ");
    age = pathana();

    // Simple if-else statement
    yadi (age >= 18) {
        lekha("You are an adult.");
    } anyatha {
        lekha("You are a minor.");
    }

    // Nested if-else statement
    yadi (age < 13) {
        lekha("You are a child.");
    } athava (age < 20) {
        lekha("You are a teenager.");
    } athava (age < 60) {
        lekha("You are an adult.");
    } anyatha {
        lekha("You are a senior citizen.");
    }

    // Boolean logic combinations
    prakruti sankhya income;
    lekha("Enter your monthly income: ");
    income = pathana();

    yadi (age > 18 && income > 30000) {
        lekha("You are eligible for a credit card.");
    } anyatha {
        lekha("You are not eligible for a credit card.");
    }

    yadi (age > 60 || income > 100000) {
        lekha("You qualify for special services.");
    }
}
samapti
```

5. While Loop - Sum of Numbers

```
// Program to calculate sum of first n natural numbers using while loop

dharma WhileLoopSum {
    prakruti sankhya n;
    prakruti sankhya sum = 0;
    prakruti sankhya i = 1;

    lekha("Enter a positive integer: ");
    n = pathana();

    yadi (n <= 0) {
        lekha("Please enter a positive number!");
    } anyatha {
        // Calculate sum using while loop
        yavat (i <= n) {
            sum += i;
            i++;
        }

        lekha("Sum of first " + n + " natural numbers is: " + sum);
    }
}
samapti
```

6. Do-While Loop - Number Guessing

```
// Simple number guessing game using do-while loop

dharma NumberGuessing {
    prakruti sankhya secretNumber = 42;
    prakruti sankhya guess;
    prakruti sankhya attempts = 0;

    lekha("Welcome to the Number Guessing Game!");
    lekha("I'm thinking of a number between 1 and 100.");

    kuru {
        lekha("Enter your guess: ");
        guess = pathana();
        attempts++;

        yadi (guess < secretNumber) {
            lekha("Too low! Try again.");
        } athava (guess > secretNumber) {
            lekha("Too high! Try again.");
        } anyatha {
            lekha("Congratulations! You've guessed the number in " + attempts + "
attempts.");
        }
    } yavat (guess != secretNumber);
}
samapti
```

7. For Loop - Multiplication Table

```
// Program to generate multiplication table using for loop

dharma MultiplicationTable {
    prakruti sankhya number;
    prakruti sankhya limit;

    lekha("Enter a number: ");
    number = pathana();

    lekha("Enter the limit: ");
    limit = pathana();

    lekha("Multiplication Table for " + number + ":");

    chakra (prakruti sankhya i = 1; i <= limit; i++) {
        lekha(number + " × " + i + " = " + (number * i));
    }
}
samapti
```

8. Function - Prime Number Check

```
// Program with a function to check if a number is prime

dharma PrimeCheck {
    prakruti sankhya number;

    lekha("Enter a positive integer: ");
    number = pathana();

    yadi (number <= 1) {
        lekha(number + " is not a prime number.");
    } anyatha {
        yadi (isPrime(number)) {
            lekha(number + " is a prime number.");
        } anyatha {
            lekha(number + " is not a prime number.");
        }
    }
}
samapti

// Function to check if a number is prime
karma satya isPrime(sankhya num) {
    // Check for divisibility from 2 to sqrt(num)
    chakra (prakruti sankhya i = 2; i * i <= num; i++) {
        yadi (num % i == 0) {
            lautana nahi;  // Not prime
        }
    }

    lautana haan;  // Prime
}
```

9. Function with Return Value - Factorial

```
// Program to calculate factorial using a recursive function

dharma FactorialCalculator {
    prakruti sankhya number;

    lekha("Enter a non-negative integer: ");
    number = pathana();

    yadi (number < 0) {
        lekha("Factorial is not defined for negative numbers.");
    } anyatha {
        prakruti sankhya result = factorial(number);
        lekha("Factorial of " + number + " is: " + result);
    }
}
samapti

// Recursive function to calculate factorial
karma sankhya factorial(sankhya n) {
    yadi (n <= 1) {
        lautana 1;
    } anyatha {
        lautana n * factorial(n - 1);
    }
}
```

10. Function with Multiple Parameters - Greatest Common Divisor

```
// Program to find GCD (Greatest Common Divisor) of two numbers

dharma GCDCalculator {
    prakruti sankhya num1;
    prakruti sankhya num2;

    lekha("Enter first number: ");
    num1 = pathana();

    lekha("Enter second number: ");
    num2 = pathana();

    prakruti sankhya result = gcd(num1, num2);
    lekha("GCD of " + num1 + " and " + num2 + " is: " + result);
}
samapti

// Function to calculate GCD using Euclidean algorithm
karma sankhya gcd(sankhya a, sankhya b) {
    yadi (b == 0) {
        lautana a;
    } anyatha {
        lautana gcd(b, a % b);
    }
}
```

11. Array Declaration and Manipulation

```
// Program demonstrating array operations

dharma ArrayOperations {
    // Array declaration and initialization
    prakruti sankhya[] numbers = [10, 20, 30, 40, 50];

    // Print array elements
    lekha("Array elements:");
    chakra (prakruti sankhya i = 0; i < 5; i++) {
        lekha("Element at index " + i + ": " + numbers[i]);
    }

    // Calculate sum of array elements
    prakruti sankhya sum = 0;
    chakra (prakruti sankhya i = 0; i < 5; i++) {
        sum += numbers[i];
    }
    lekha("Sum of array elements: " + sum);

    // Find maximum element
    prakruti sankhya max = numbers[0];
    chakra (prakruti sankhya i = 1; i < 5; i++) {
        yadi (numbers[i] > max) {
            max = numbers[i];
        }
    }
    lekha("Maximum element: " + max);

    // Modify array elements
    lekha("\nMultiplying each element by 2:");
    chakra (prakruti sankhya i = 0; i < 5; i++) {
        numbers[i] *= 2;
        lekha("Element at index " + i + " now: " + numbers[i]);
    }
}
samapti
```

12. String Manipulation

```
// Program demonstrating string operations

dharma StringOperations {
    prakruti shabda greeting = "Namaste";
    prakruti shabda name;

    lekha("Enter your name: ");
    name = pathana();

    // String concatenation
    prakruti shabda message = greeting + ", " + name + "!";
    lekha(message);

    // String length
    lekha("Length of your name: " + name.length());

    // Convert to uppercase
    lekha("Your name in uppercase: " + name.toUppercase());

    // Check if string contains a substring
    lekha("Enter a character to check in your name: ");
    prakruti akshara char = pathana();

    yadi (name.contains(char)) {
        lekha("Your name contains the character '" + char + "'.");
    } anyatha {
        lekha("Your name does not contain the character '" + char + "'.");
    }

    // Substring
    yadi (name.length() > 3) {
        prakruti shabda firstThree = name.substring(0, 3);
        lekha("First three characters of your name: " + firstThree);
    }
}
samapti
```

13. Map (Dictionary) Usage

```
// Program demonstrating map (dictionary) operations

dharma MapOperations {
    // Create a map of Indian states and their capitals
    prakruti kosha<shabda, shabda> stateCapitals = {
        "Gujarat": "Gandhinagar",
        "Maharashtra": "Mumbai",
        "Tamil Nadu": "Chennai",
        "Karnataka": "Bengaluru",
        "Kerala": "Thiruvananthapuram"
    };

    // Accessing map values
    lekha("Capital of Gujarat: " + stateCapitals["Gujarat"]);

    // Adding new key-value pair
    stateCapitals["Rajasthan"] = "Jaipur";

    // Displaying all key-value pairs
    lekha("\nIndian States and their Capitals:");
    chakra (prakruti shabda state : stateCapitals.keys()) {
        lekha(state + ": " + stateCapitals[state]);
    }

    // Check if key exists
    prakruti shabda stateToCheck;
    lekha("\nEnter a state name to check: ");
    stateToCheck = pathana();

    yadi (stateCapitals.containsKey(stateToCheck)) {
        lekha("Capital of " + stateToCheck + " is " + stateCapitals[stateToCheck]);
    } anyatha {
        lekha("Information not available for " + stateToCheck);
    }

    // Remove a key-value pair
    prakruti shabda stateToRemove;
    lekha("\nEnter a state name to remove: ");
    stateToRemove = pathana();

    yadi (stateCapitals.containsKey(stateToRemove)) {
        stateCapitals.remove(stateToRemove);
        lekha(stateToRemove + " has been removed from the map.");
    } anyatha {
        lekha(stateToRemove + " is not in the map.");
    }
}
samapti
```

14. Nested Loops - Pattern Printing

```
// Program to print a pyramid pattern using nested loops

dharma PyramidPattern {
    prakruti sankhya rows;

    lekha("Enter the number of rows: ");
    rows = pathana();

    lekha("Pyramid Pattern:");

    chakra (prakruti sankhya i = 1; i <= rows; i++) {
        // Print spaces
        chakra (prakruti sankhya j = 1; j <= rows - i; j++) {
            lekha(" ");
        }

        // Print stars
        chakra (prakruti sankhya k = 1; k <= 2 * i - 1; k++) {
            lekha("*");
        }

        // Move to next line
        lekha("\n");
    }
}
samapti
```

15. Error Handling

```
// Program demonstrating error handling

dharma ErrorHandlingDemo {
    prakruti sankhya a;
    prakruti sankhya b;
    prakruti sankhya result;

    lekha("Division with Error Handling");
    lekha("Enter first number (a): ");
    a = pathana();

    lekha("Enter second number (b): ");
    b = pathana();

    prayatna {
        // Check for division by zero
        yadi (b == 0) {
            // Throw an error
            throw "GanitaDosha";
        }

        result = a / b;
        lekha("Result of a / b = " + result);

    } dosha (GanitaDosha error) {
        lekha("Error: Cannot divide by zero!");

    } dosha (AnyaDosha error) {
        lekha("An unknown error occurred: " + error.message);

    } antima {
        lekha("Division operation completed. Thank you!");
    }
}
samapti
```

These sample programs cover a wide range of programming concepts and features available in the Devbhasha language. Feel free to modify and experiment with these examples to better understand how Devbhasha works.