Devbhasha: The Divine Programming Language

Introduction

Devbhasha ("Language of the Gods" in Sanskrit) is a programming language inspired by Indian mythology, culture, and history. This language aims to make programming accessible while celebrating India's rich heritage through its syntax and structure. Devbhasha blends modern programming concepts with traditional Indian philosophical and mythological elements.

Philosophy

Devbhasha follows principles derived from ancient Indian texts:

- **Clarity** (Spashta): Code should be clear and understandable
- **Purpose** (Lakshya): Every element should serve a defined purpose
- **Harmony** (Sanyoga): Different parts of the program should work together seamlessly
- **Elegance** (Soundarya): The language should be aesthetically pleasing to write and read

Program Structure

Every Devbhasha program follows a structure similar to ancient Sanskrit texts, with a clear beginning and ending:

```
dharma ProgramName {
    // Main code here
}
samapti
```

The keyword `dharma` marks the beginning of a program (comparable to "main" in other languages), and `samapti` marks its conclusion.

Basic Syntax

Statements and Blocks

All statements in Devbhasha end with a semicolon (`;`). Code blocks are enclosed in curly braces `{ }`.

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

Data Types

Devbhasha includes the following primary data types:

Type	Keyword	Description	Example
Boolean	`satya`	Truth values	`haan` (true), `nahi` (false)
Integer	`sankhya`	Whole numbers	`108`, `-42`
Float	`dashamika`	Decimal numbers	`3.14`, `-0.5`
Character	`akshara`	Single characters	`'a'`, `'म'`
String	`shabda`	Text values	`"Namaste"`
Null	`shunya`	Empty value	`shunya`

Variables and Constants

Variable Declaration

Variables are declared using the `prakruti` keyword, followed by the data type and variable name:

```
prakruti shabda devName = "Vishnu";
prakruti sankhya yugas = 4;
prakruti satya isImmortal = haan;
```

Variables can also be declared without initialization:

```
prakruti shabda playerName;
playerName = "Arjuna";
```

Constants

Constants are declared using the `nitya` keyword:

```
nitya sankhya avatars = 10;
nitya shabda epicName = "Mahabharata";
```

Operators

Arithmetic Operators

Operator	Keyword	Description	Example
+	`yoga`	Addition	`a + b`
-	`viyoga`	Subtraction	`a - b`
*	`guna`	Multiplication	`a * b`
/	`bhaga`	Division	`a / b`
%	`shesha`	Modulus	`a % b`
++	`vriddhi`	Increment	`a++`
--	`hrasa`	Decrement	`a--`

Comparison Operators

Operator	Keyword	Description	Example
==	`samana`	Equal to	`a == b`
!=	`asamana`	Not equal to	`a != b`
>	`adhika`	Greater than	`a > b`
<	`nyuna`	Less than	`a < b`
>=	`adhikasamana`	Greater than or equal to	`a >= b`
<=	`nyunasamana`	Less than or equal to	`a <= b`

Logical Operators

Operator	Keyword	Description	Example
&&	`cha`	Logical AND	`a && b`
||	`va`	Logical OR	`a || b`
!	`na`	Logical NOT	`!a`

Assignment Operators

Operator	Description	Example
=	Assignment	`a = b`
+=	Add and assign	`a += b`
-=	Subtract and assign	`a -= b`
*=	Multiply and assign	`a *= b`
/=	Divide and assign	`a /= b`
%=	Modulus and assign	`a %= b`

Control Structures

Conditional Statements

```
yadi (condition) {
    // Code if condition is true
} athava (anotherCondition) {
    // Code if another condition is true
} anyatha {
    // Code if no conditions are true
}
```

Example:

```
yadi (age > 100) {
    lekha("You are a centenarian!");
} athava (age > 60) {
    lekha("You are a senior citizen.");
} anyatha {
    lekha("You are young!");
}
```

Switch Statement

```
vikalpa (expression) {
    sthiti value1:
        // Code for value1
        virama;
    sthiti value2:
        // Code for value2
        virama;
    apavada:
        // Default code
}
```

Loops

For Loop

```
chakra (initialization; condition; increment) {
    // Repeating code
}
```

Example:

```
chakra (prakruti sankhya i = 0; i < 10; i++) {
    lekha("Iteration " + i);
}
```

While Loop

```
yavat (condition) {
    // Repeating code
}
```

Example:

```
prakruti sankhya count = 0;
yavat (count < 5) {
    lekha("Count: " + count);
    count++;
}
```

Do-While Loop

```
kuru {
    // Code to execute
} yavat (condition);
```

Example:

```
prakruti sankhya attempt = 0;
kuru {
    lekha("Attempt " + attempt);
    attempt++;
} yavat (attempt < 3);
```

Break and Continue

- `virama`: Break statement
- `agre`: Continue statement

Example:

```
chakra (prakruti sankhya i = 0; i < 10; i++) {
    yadi (i == 5) {
        virama; // Exit the loop
    }
    yadi (i % 2 == 0) {
        agre; // Skip to next iteration
    }
    lekha("Processing " + i);
}
```

Functions

Functions are declared using the `karma` keyword:

```
karma functionName(parameters) {
    // Function body
    lautana value; // Return statement
}
```

Example:

```
karma calculateSum(sankhya a, sankhya b) {
    lautana a + b;
}

// Function call
prakruti sankhya result = calculateSum(5, 3);
```

Void Functions

Functions that don't return a value use the `shunya` keyword:

```
karma shunya greet(shabda name) {
    lekha("Namaste, " + name + "!");
}
```

Arrays and Collections

Arrays

```
prakruti dataType[] arrayName = [value1, value2, ...];
```

Example:

```
prakruti shabda[] pandavas = ["Yudhishthira", "Bhima", "Arjuna", "Nakula", "Sahadeva"];
prakruti sankhya[] numbers = [1, 2, 3, 4, 5];
```

Accessing array elements:

```
prakruti shabda firstPandava = pandavas[0];
```

Maps (Dictionaries)

```
prakruti kosha<keyType, valueType> mapName = {key1: value1, key2: value2};
```

Example:

```
prakruti kosha<shabda, sankhya> devaShakti = {"Indra": 100, "Agni": 85, "Vayu": 90};
```

Accessing map elements:

```
prakruti sankhya indraStrength = devaShakti["Indra"];
```

Comments

```
// Single line comment

/*
   Multi-line
   comment
*/
```

Input/Output

- `pathana`: Input command
- `lekha`: Output command

Example:

```
lekha("Enter your name:");
prakruti shabda userName = pathana();
lekha("Namaste, " + userName + "!");
```

Classes and Objects

```
varga ClassName {
    // Properties
    prakruti dataType propertyName;

    // Constructor
    nirmata(parameters) {
        // Initialization code
    }

    // Methods
    karma methodName(parameters) {
        // Method body
    }
}
```

Creating an object:

```
prakruti ClassName objectName = nava ClassName(parameters);
```

Example:

```
varga Deva {
    prakruti shabda name;
    prakruti sankhya power;

    nirmata(shabda devName, sankhya devPower) {
        swayam.name = devName;
        swayam.power = devPower;
    }

    karma shabda getDescription() {
        lautana swayam.name + " has power level " + swayam.power;
    }
}

prakruti Deva vishnu = nava Deva("Vishnu", 1000);
lekha(vishnu.getDescription());
```

Inheritance

```
varga ChildClass prachina ParentClass {
    // Child class members
}
```

Example:

```
varga Avatar prachina Deva {
    prakruti shabda incarnation;

    nirmata(shabda name, sankhya power, shabda form) {
        param.nirmata(name, power);
        swayam.incarnation = form;
    }
}
```

Error Handling

```
prayatna {
    // Code that might cause an error
} dosha (ErrorType error) {
    // Error handling code
} antima {
    // Code that always executes
}
```

Example:

```
prayatna {
    prakruti sankhya result = 10 / 0;
} dosha (MathError error) {
    lekha("Math error: " + error.message);
} dosha (AnyError error) {
    lekha("Unknown error: " + error.message);
} antima {
    lekha("Error handling complete.");
}
```

Modules and Imports

```
// Importing a module
aahvana ModuleName;

// Exporting from a module
pradana karma functionName() {
    // Function code
}
```

Standard Library

Devbhasha includes modules representing various aspects of Indian mythology:

- `Ganita`: Math functions
- `Kala`: Date and time functions
- `Bhasha`: String manipulation
- `Sangraha`: Collection utilities
- `Vyavastha`: System operations

Example:

```
aahvana Ganita;

dharma MathDemo {
    prakruti dashamika pi = Ganita.PI;
    prakruti sankhya result = Ganita.power(2, 10);
    lekha("Result: " + result);
}
samapti
```

Sample Program: Fibonacci Series

```
dharma Fibonacci {
    lekha("Fibonacci Series Generator");
    lekha("How many terms? ");
    prakruti sankhya terms = pathana();

    prakruti sankhya a = 0;
    prakruti sankhya b = 1;

    lekha(a);
    lekha(b);

    chakra (prakruti sankhya i = 2; i < terms; i++) {
        prakruti sankhya next = a + b;
        lekha(next);
        a = b;
        b = next;
    }
}
samapti
```

Appendix: Cultural References

Many keywords in Devbhasha are derived from Sanskrit terminology:

- `dharma` (धर्म): Duty, purpose, law
- `karma` (कर्म): Action, function, work
- `yadi` (यदि): If
- `chakra` (चक्र): Wheel, cycle
- `varga` (वर्ग): Class, category
- `lekha` (लेख): Write, document
- `pathana` (पठन): Reading, study

This language specification document serves as a comprehensive guide to programming in Devbhasha. As the language evolves, additional features may be added to enhance its capabilities while maintaining its cultural authenticity.