Iamstuck Language Documentation

Introduction

"Iamstuck" is a programming language designed for simplicity and ease of use, with a primary focus on providing intuitive mechanisms for working with containers. Drawing inspiration from C++ and Python, Iamstuck offers a set of keywords, operators, and constructs to facilitate common programming tasks.

Keywords

The following keywords are available in the "Iamstuck" language:

- `DEF`: Used to define functions.
- `RETURN`: Used to return a value from a function.
- `IF`, `ELIF`, `ELSE`: Conditional statements.
- `FOR`: Looping construct for iterating over a range or container.
- `PRINT`, `PRINTLN`, `PRINTF`: Output statements for printing to the console.
- `PASS`: Placeholder statement indicating no operation.
- `STACK`, `QUEUE`, `DEQUE`: Container types for stacks, queues, and deques.
- `WHILE`: Looping construct for a while loop.

Operators

Arithmetic Operators

- `+`: Addition
- `-`: Subtraction
- `*`: Multiplication
- `/`: Division
- `%`: Modulus

Assignment Operators

- `=`: Assign
- `+=`: Add and assign
- `-=`: Subtract and assign
- `*=`: Multiply and assign
- `/=`: Divide and assign
- `%=`: Modulus and assign

Increment and Decrement Operators

- `++`: Increment
- `--`: Decrement

Logical Operators

The language supports the following logical operators:

- `AND`: Logical AND

- `OR` : Logical OR
- `NOT` : Logical NOT
- `XOR` : Logical XOR

Functions

Functions are defined using the `DEF` keyword. The name of the function must not be "main" due to translation constraints.

Example:

```
DEF void myFunction(int x, string s):
    // Function body
END
```

Control Flow

Conditional Statements

Conditional statements use `IF` , `ELIF` , and `ELSE` :

```
IF condition:
    // Code block
ELIF another_condition:
    // Code block
ELSE:
    // Code block
END
```

Loops

For Loop

```
FOR i IN RANGE(start, end, step):
    // Code block
END
```

While Loop

```
WHILE condition:
    // Code block
END
```

Printing to Console

The language provides multiple ways to print to the console:

- `PRINT(args)` : Prints without a newline.
- `PRINTLN(args)` : Prints with a newline.

- `PRINTF(format, args)` : Prints using a format string.

Data Structures

The language supports three container types:

- `STACK<T>` : A stack of data type `T` .
- `QUEUE<T>` : A queue of data type `T` .
- `DEQUE<T>` : A deque of data type `T` .

Containers Methods and Operations

Common to All Containers

Methods

- `bool empty()` : Returns `true` if the container is empty; otherwise, returns `false` .
- `int size()` : Returns the size of the container.
- `T back()` : Returns the last element in the container.
- `void print()` : Prints all elements in the container.

Operators

- `T& operator[](int index)` : Returns a reference to the element at the specified index.

`Queue` Methods and Operators

Additional Methods

- `void push(T val)` : Adds the element `val` to the back of the queue.
- `T pop_front()` : Removes and returns the element from the front of the queue.
- `T front()` : Returns the element at the front of the queue.

`Stack` Methods and Operators

Additional Methods

- `void push(T val)` : Adds the element `val` to the back of the stack.
- `T pop()` : Removes and returns the element from the top of the stack.

`Deque` Methods and Operators

Additional Methods

- `void push_front(T val)` : Adds the element `val` to the front of the deque.
- `void push(T val)` : Adds the element `val` to the back of the deque.
- `T pop_front()` : Removes and returns the element from the front of the deque.
- `T pop()` : Removes and returns the element from the back of the deque.
- `T front()` : Returns the element at the front of the deque.

Sample Codes

```
// Tower of Hanoi

DEF void tower_of_hanoi(int n, char from, char to, char aux):
    IF n == 1:
        PRINTF("Move disk 1 from rod {} to rod {}\n", from, to);
        RETURN;
    END
    tower_of_hanoi(n - 1, from, aux, to);
    PRINTF("Move disk {} from rod {} to rod {}\n", n, from, to);
    tower_of_hanoi(n - 1, aux, to, from);
END

int n = 3;
tower_of_hanoi(n, 'A', 'C', 'B');
```

Console Output Examples

```
// Different ways to print to console

int x = 2;
string s = "ola";

PRINT("x = ", x, ", s = ", s, "\n");
PRINTLN("x = ", x, ", s = ", s);
PRINTF("x = {}, s = {}\n", x, s);
```

Looping Constructs

```
// Examples of FOR and WHILE loops

FOR i IN RANGE(5):
    PRINT(i);
END

FOR i IN RANGE(1, 10):
    PASS;
END

FOR j IN RANGE(100, -1, -1):
    PRINT(i * 2);
END

STACK<int> stack;

FOR s IN stack:
    s += 1;
END

int i = 10;

WHILE i > 0:
    i -= 1;
END
```

Notes

Ensure correct usage of syntax and indentation for proper execution.
The language supports a variety of common programming constructs for versatile use.

Avoid naming functions as "main" for compatibility with C++ translation.