

# XYZ Language Specification

## Compilers Project

### Authors

Vishal Vijay Devadiga (CS21BTECH11061)

Satpute Aniket Tukaram (CS21BTECH11056)

Mahin Bansal (CS21BTECH11034)

Harshit Pant (CS21BTECH11021)

# Table of Contents

- Table of Contents
- Introduction
  - What is XYZ?
  - Why XYZ?
- Language Specifications
  - Data Types
    - \* Primitive Data Types
      - Integer:
      - Character:
      - Float:
      - Boolean:
    - \* Composite Data Types
      - Sets:
      - Structs:
    - \* Comments
  - Operators
    - \* Arithmetic Operators
    - \* Logical Operators
    - \* Comparison Operators
    - \* Assignment Operators
    - \* Misc Operators
    - \* Operator Precedence
  - Control Flow
    - \* If-Else
    - \* Loops
  - Constants
  - Keywords
  - Identifiers
  - Statements
- References

# Introduction

What is XYZ?

Why XYZ?

# Language Specifications

XYZ follows, making it easier for programmers to pick up XYZ easily and keep their focus on the logic rather than XYZ.

- XYZ is a **statically typed** language
- XYZ is a **strongly typed** language
- XYZ is a **procedural** language
- XYZ is case sensitive.

XYZ does not support Object Oriented Programming(**OOPs**).

## Data Types

XYZ uses common data types found in most programming languages.

### Primitive Data Types

**Integer:** Signed Integers are represented by the `int_x` keyword, where `x` is the number of bits used to represent the integer. XYZ supports 8, 16, 32 and 64 bit integers.

Unsigned Integers are represented by the `uint_x` keyword, where `x` is the number of bits used to represent the integer. XYZ supports 8, 16, 32 and 64 bit integers.

**Character:** Characters are represented by the `char` keyword. XYZ supports 8 bit characters.

**Float:** Floats are represented by the `float_x` keyword, where `x` is the number of bits used to represent the float. XYZ supports 32 and 64 bit floats.

**Boolean:** Booleans are represented by the `bool` keyword, which is similar to the `bool` keyword in C, C++, Java and Python.

### Composite Data Types

**Sets:** Sets are collections of elements of the same data type. XYZ supports two types of sets: Ordered Sets and Unordered Sets.

- Ordered Sets are represented by the `o_set` keyword.
- Unordered Sets are represented by the `u_set` keyword.

**Structs:** Structs are represented by the `struct` keyword. Structs can contain any data type supported by XYZ.

## Comments

XYZ has only one type of comment, that can act as both single line and multi line comments. The comment starts with `<!--` and ends with `--!>`. Below is an example of a comment:

```
<!-- This is a comment --!>
```

```
<!-- This is a  
multi line comment --!>
```

## Operators

Operators supports by XYZ are similar to the operators supported by C.

### Arithmetic Operators

Operator	Description	Associativity
+	Addition	Left to Right
-	Subtraction	Left to Right
*	Multiplication	Left to Right
/	Division	Left to Right
%	Modulo	Left to Right

### Logical Operators

Operator	Description	Associativity
&&	Logical AND	Left to Right
	Logical OR	Left to Right
!	Logical NOT	Right to Left

### Comparison Operators

Operator	Description	Associativity
==	Equal to	Left to Right
!=	Not equal to	Left to Right
>	Greater than	Left to Right
<	Less than	Left to Right
>=	Greater than or equal to	Left to Right

Operator	Description	Associativity
<=	Less than or equal to	Left to Right

## Assignment Operators

Operator	Description	Associativity
=	Assignment	Right to Left
+=	Addition Assignment	Right to Left
-=	Subtraction Assignment	Right to Left
*=	Multiplication Assignment	Right to Left
/=	Division Assignment	Right to Left
%=	Modulo Assignment	Right to Left
&=	Logical AND Assignment	Right to Left
\ =	Logical OR Assignment	Right to Left

## Misc Operators

Operator	Description	Associativity
.	Access Struct Member	Left to Right
[]	Access Set Element	Left to Right

## Operator Precedence

Operator	Description
()	Parentheses
!	Logical NOT
*, /, %	Multiplication, Division, Modulo
+, -	Addition, Subtraction
>, <, >=, <=	Comparison
==, !=	Equality
&&	Logical AND
\ \	Logical OR
=	Assignment
+=, -=, *=, /=, %=, &=, \ =	Assignment

## Control Flow

XYZ enforce the use of curly braces for all control flow statements. XYZ does not support the use of indentation for control flow statements. XYZ supports the following control flow statements:

### If-Else

Below is the syntax for the if-else statement:

```
if (condition) {  
    statement;  
} elif {  
    statement;  
} else {  
    statement;  
}
```

### Loops

XYZ only supports the `while` loop. Below is the syntax for the `while` loop:

```
while (condition) {  
    statements;  
}
```

## Constants

Constants are represented by the `const` keyword. Constants can be of any data type supported by XYZ.

## Keywords

Keyword	Description
<code>int_x</code>	Integer
<code>uint_x</code>	Unsigned Integer
<code>char</code>	Character
<code>float_x</code>	Float
<code>bool</code>	Boolean
<code>const</code>	Constant
<code>struct</code>	Struct
<code>o_set</code>	Ordered Set

Keyword	Description
<code>u_set</code>	Unordered Set
<code>if</code>	If
<code>elif</code>	Else If
<code>else</code>	Else
<code>while</code>	While
<code>break</code>	Break
<code>continue</code>	Continue
<code>return</code>	Return
<code>true</code>	True
<code>false</code>	False
<code>&lt;!--</code>	Start of comment
<code>--!&gt;</code>	End of comment

## Identifiers

XYZ uses the following rules for identifiers:

- Identifiers can only contain alphanumeric characters and underscores.
- Identifiers cannot start with a number.
- Identifiers cannot be a keyword.
- Identifiers cannot contain spaces.
- Identifiers cannot contain special characters.

Regular Expressions for Identifiers:

```
[a-zA-Z_][a-zA-Z0-9_]*
```

## Statements



## References