Compilers Project

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:
    - Comments
  - Operators
    - Arithmetic Operators
    - Logical Operators
    - Comparison Operators
    - Assignment Operators
    - Operator Precedence
  - Control Flow
    - If-Else
    - Loops
  - Constants
  - Keywords
  - Identifiers
- References

Introduction

What is XYZ?

Why XYZ?

Language Specifications

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

- XYZ is a **statically typed** language
- XYZ is a **strongly typed** language
- XYZ is a **procedural** language

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

Data Types

The language 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. The language 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. The language supports 8, 16, 32 and 64 bit integers.

Character: Characters are represented by the `char` keyword. The language 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. The language 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.

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 the language 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
`<=`	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

Operator	Description	Associativity
`*=`	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

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. The language does not support the use of indentation for control flow statements. The language 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 the language.

Keywords

Keyword	Description
`int_x`	Integer
`uint_x`	Unsigned Integer
`char`	Character
`float_x`	Float
`bool`	Boolean
`const`	Constant
`if`	If
`elif`	Else If
`else`	Else
`while`	While
`break`	Break
`continue`	Continue
`return`	Return
`true`	True
`false`	False
`<!--`	Start of comment
`--!>`	End of comment

Identifiers

The language 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 are case sensitive.
- Identifiers cannot contain special characters.

Regular Expressions for Identifiers:

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

References