Computer

Creators:

Braedon Behnke (bcb346@nau.edu)
Eli Barela (emb538@nau.edu)
Aadarsha Bastola (ab3824@nau.edu)

1. Introduction

Computer is a simple **Procedure Oriented**, English-like programming language designed for beginners. As such, it is meant for smaller projects and isn't expected to be utilized in real-world applications outside of education.

Computer uses a syntax that is easy to understand and read, making it ideal for those who are just starting to learn programming. Source Code for the language will be saved as **.compute** files.

2. Translator

Computer will be an **interpreted language**. Since this language is primarily meant for beginners, using an interpreter makes it very easy to run code without needing to compile the code. Just write the code and start the program!

This also makes it easier to debug mistakes, as the interpreter can tell you exactly where something went wrong in the code.

3. Syntax

Computer's syntax is very close to English language structure, this makes it easy to read and write code. Similar to Python, indents are used to distinguish code blocks instead of brackets like C or Java.

Comments are ignored by the interpreter and are followed by `//`

4. Data Types

INT: Integer Data type, Stores integer values (numerical value without decimal point)

Example: 1, 2, -10, 1000

STR: String Data type, stores sequence of characters

Example: "Ron", "Hello World"

CHAR: Character data type, stores a single character

Example: 'a', 'z'

BOOL: Boolean Data type, stores **True** or **False** values

DOUBLE: Double Data Type, stores numerical values with decimal points

Example: 10001.2, 22.1

5. Creating and Declaring Variables

Variables are created with the keyword **`CREATE`** followed by the data type **`(INT, STR, BOOL, DOUBLE).`**

Examples:

```
    CREATE INT integerVariable IS 100
    CREATE STR stringVariable IS “Hello World!”
    CREATE DOUBLE doubleVariable IS 1992.26
    CREATE BOOL booleanVariable IS True
```

Variables that are not assigned any values will have **NULL** value by default:

```
    CREATE INT integerVariable
    CREATE STR stringVariable
    CREATE DOUBLE doubleVariable
    CREATE BOOL booleanVariable
```

A **DYNAMIC** variable can be declared which can store any kind of supported data type, however once one data type is stored in the variable it can not store other kind of data type:

```
    CREATE DYNAMIC booleanVariable
```

An **ARRAY** of any **INT, STR or DOUBLE** can be created.

```
    CREATE STR stringArray IS [“This”, “is”, “the”, “best”,
                                                  “language”]
```

6. Operators

6.1 Arithmetic Operators:

Operator	Operation	Example
+	Addition	sum IS var1 + var2
-	Subtraction	diff IS var2 - var1
*	Multiplication	prod IS var1 * var2
/	Division	div IS var2 / var1
%	Modulus	mod IS mod1 % mod2
^	Power	result IS base ^ power

6.2 Logical Operators:

Operator	Operation	Example

`AND`	Logical And	`var1 AND var2`
`OR`	Logical Or	`var2 OR var1`
`NOT`	Logical Not	`NOT(var1)`

6.3 Relational Operators:

Operator	Operation	Example
`EQUALS`	Equals to	`var1 EQUALS var2`
`NOTEQUALTO`	Not Equals to	`var1 NOTEQUALTO var2`
`LESSTHAN`	Less Than	`var1 LESSTHAN var2`
`GREATERTHAN`	Greater Than	`var1 GREATERTHAN var2`
`LESSTHANOREQUAL`	Less than or equals to	`var1 LESSTHANOREQUAL var2`
`GREATERTHANOREQUAL`	Greater than or equals to	`var1 GREATERTHANOREQUAL var2`

6.4 Assignment Operators

Operator	Operation	Example
`IS`	Assignment (var1 = var2)	`var1 IS var2`

7. Control Structures

`IF ELSE:` If statement runs the code if a condition is met. **`IF`** is followed by a condition which is followed by a semicolon, the code chunk to be run if the condition is met is indented in the next line.

```
IF Condition:
    // Code to execute if the condition is met
```

```
     ELSE:
          // Code to execute if the condition is not met
```

Example:

```
     IF var1 NOTEQUALSTO var2:
          PRINT "Var1 is not equal to var2"
     ELSE:
          PRINT "Var1 is equal to var2"
```

8. Loops

8.1 For Loop:

FOR Loop Allows to execute a block of code repeatedly for a specified number of times or until a certain condition is met.

```
     FOR variable IS initialvalue TO finalValue STEP step:
          // Code to be repeated
```

(Step is Optional, can be used if a programmer wants to, default value is 1)

Example:

```
     FOR index IS 0 TO 100 STEP 5:
          PRINT(index)
```

8.2 While Loop:

WHILE Loop Allows to execute a block of code repeatedly for a infinite number of times or until a certain condition is met.

```
     WHILE variable1 EQUALS variable2
          // Code to be repeated
```

Example:

```
     WHILE index LESSTHAN 10:
          PRINT index
          index IS index + 1
```

9. Functions

Functions are defined by **COMPUTER** keyword followed by function name in uppercase followed by the parameters for the function.

```
     COMPUTER FUNCTION_NAME parameter1, parameter2:
        RETURN functionReturn
```

Example:

```
    COMPUTER ADD_NUMBERS number1, number2:
        RETURN number1 + number2
```

10. Program Example

Program to check and print if a number is even or odd numbers between 1 to 20:

```
COMPUTER START
    CREATE INT index IS 0
    CREATE INT range IS 20

    FOR index TO range:
        ODD_OR_EVEN index


COMPUTER ODD_OR_EVEN testNumber:
    IF testNumber % 2 EQUALS 0:
        PRINT testNumber + " is an EVEN Number"
    ELSE:
        PRINT testNumber + " is a Odd Number"
```