Algonquin College Logo

# SCHOOL OF ADVANCED TECHNOLOGY

### ICT - Applications & Programming

### Computer Engineering Technology – Computing Science



A11

Language Specification

Lab Professor / Lab Session:

[Paulo Sousa /302]

Team:

[Jwell Lalakiya] - Id: [041056229] / [Priya Patel] - Id: [041084491]

Language Name [DefaultX]

|  |  |
| --- | --- |
| **Part**  **1** | **Language User Reference** |

* 1. **User Manual**

**Element 1: Name / Extension**

**Name:** DefaultX

**Extension: “**.dfx”

**DefaultX** is an open-source programming language that is similar to programming language “Go”. Creating this Go-inspired language allows us to build on the success and familiarity of Go while introducing new features or improvements. DefaultX allows programmers to explore alternative language features, syntax, or paradigms while maintaining a familiar foundation of Go.

**Element 2 – Comments**

**Single Line Comment:** (/ /)

**Multi Line Comment:** (//\*.….\*//)

**Element 3 – Keywords**

* *break*
* *default*
* *func*
* *case*
* *struct*
* *if / else*
* *package*
* *switch*
* *const*
* *type*
* *continue*
* *for*
* *import*
* *return*
* *var*

**Element 4 – Datatypes**

**Integer:**

* Syntax: int32
* Size: 4 bytes (32 bits)

**Float (Single Precision):**

* Syntax: float32
* Size: 4 bytes (32 bits)

**Float (Double Precision):**

* Syntax: float64
* Size: 8 bytes (64 bits)

**Strings:**

* Syntax: string

**Boolean:**

* Syntax: bool
* Size: 1 byte

**Element 5 – Variables**

**Integer:** var [variable name] **int32** = [integer value].

**Boolean:** var [variable name] **bool** = [true/false]

**String:** var [variable name] **string** = [string value]

**Floating:** var [variable name] **float64** = [double precision value]

var [variable name] **float32** = [single precision value]

**Element 6 – Methods / Functions**

**Method Declaration:**

**func** methodName(parameters) **return\_type** {

**// Method implementation**

}

**Element 7 - Commands**

**Attribution / assignment**:

* A programmer can assign a value to any variale using “**Assignment opperator (=)**”.
* Our language does support casting but it will be only limited to similar type. For example, a programmer cannot cast from “**int**” to “**string**”.
* All the arithmetic operation will be performed using regular **arithmetic operators** i.e. **(+,-,/,%,\*)**
* You can join two strings using “**+**” operator if both the operands are string.

**Selection:**

In our language conditions there are two conditional statements:

* **If/else:**

**if (condition)** {

//block of code

} **else if (condition)** {

// block of code

} **else** {

//block of code

}

* **Example:**

**if age < 18 {**

**fmt.Println("You are a minor.")**

**} else if age >= 18 && age < 65 {**

**fmt.Println("You are an adult.")**

**} else {**

**fmt.Println("You are a senior citizen.")**

**}**

**Conditions** usually work when there are two operands and one comparison operator in between. So basically, the operands are compared bassed on the comparison operator used in between. Following are the comparision operator that can be used in our language:

* “==” / “!=”
* “>”/”<”
* “>=”/”<=”

Programmer can also combime two conditions using logical operators. Following are the list of logical operators:

* AND
* OR
* NOT

**Interaction:**

Our language has **for loop** as well as **for each** loop to loop through array elements.

**for** ([*initialize loop variable*];[*exit condition*];[*increment/decrement*]){

}

**Input**:  
Programmers can use “**Scanf**” to read an input from the keyboard depending on the format specifier.

***fmt.*Scanf(“[*format specifier*]”,&[*vairable name to be stored in*]);**

**Output:**

Programmers can use “**Println**” to read an input from the keyboard depending on the format specifier.

***fmt.*Println(“[*String*/*format specifier*]”,[*vairable name the value is stored in*]);**

**Element 7 – Proper elements**

We are planning to add **exception handling** **(Try-catch)** in our language which is not in Go nor in C language

|  |  |
| --- | --- |
| **Part**  **2** | **Language Comparison** |

**Comparing with C language**

**Differences**

|  |  |  |
| --- | --- | --- |
|  | * C has a traditional syntax while DefaultX has more of a non conventional syntax while defining variables and functions. For example: In DefaultX programmer has to write the data type at the end after decarling the variables. * C does not have exception handling while DefaultX has exceptional handling. (Try-Catch) |  |

**Advantages / Disadvantages (in comparison with C)**

|  |  |  |
| --- | --- | --- |
|  | Advantages:   * DefaultX has error handling while C doesn’t have anything similar * DefaultX has a unique syntax which makes reading the syntax for someone who is new to programming easier.   Disadvantages:   * DefaultX has a non conventional syntax which makes it harder for someone which strong programming backgroud to learn this syntax. |  |

**Comparing with another language**

**Language Name: Java**

**Differences**

|  |  |  |
| --- | --- | --- |
|  | * Java is Object-Oriented language while DefaultX is Procedure-Oriented language. * Java has a traditioinal syntax while DefaultX has more of a non conventional syntax while defining variables and functions. For example: In DefaultX programmer has to write the data type at the end after decarling the variables. |  |

**Advantages / Disadvantages (in comparison with this second language)**

|  |  |  |
| --- | --- | --- |
|  | Advantages:   * DefaultX has a unique syntax which makes reading the syntax for someone who is new to programming easier. * In Java, you need to write more code to achieve the same functionality that can be achieved with less code in Go. This makes Java code harder to read and maintain, especially for larger projects.   Disadvantages:   * Java has an extensive ecosystem of libraries, frameworks, and tools. DefaultX, on the other hand, is relatively new and its ecosystem is still developing. |  |

|  |  |
| --- | --- |
| **Part**  **3** | **Architectural Questions** |

**Advantages**

The goal of my language is to make a language is to make something simile to Go Language and learn what the obstacles I will face when creating a new language. My language will definetly be usefull to for GUI developers and game developers.

**Strategy: C Implementation**

* We are planning to implement “string” datatype using “Character Array/pointer”. As C doesn’t have String data type, we can use pointers or character array to implement those while passing in a function or printing them*.*
* C does not have native support for try-catch blocks like some other programming languages such as C++ or Java. However, you can implement a form of error handling in C using standard C features like setjmp and longjmp. This is a way to simulate try-catch behavior.

Here's a basic example:

#include <stdio.h>

#include <setjmp.h>

jmp\_buf exception\_env;

void throw\_exception() {

longjmp(exception\_env, 1);

}

int main() {

if (setjmp(exception\_env) == 0) {

// Try block

printf("Trying something...\n");

throw\_exception(); // Simulate an exception

printf("This won't be executed.\n");

} else {

// Catch block

printf("Caught an exception!\n");

}

printf("Program continues after the 1294try-catch block.\n");

return 0;

}

* While printing numerical values using “**fmt.Println()”** we will need to convert numerical data into character/string data. For example:- If we want to print an integer/float variable using “**fmt.Println()**” we need to convert the value stored in variable into string behind the scenes.[Your ideas about how to identify scope (ex: blocks between conditionals or functions)]
* In our language the scope management is done using “{ }”.

**References**

[1] <https://go.dev/ref/spec>

[2] <https://chat.openai.com>

Algonquin College

Fall, 2023