Algonquin College Logo

# SCHOOL OF ADVANCED TECHNOLOGY

### ICT - Applications & Programming

### Computer Engineering Technology – Computing Science



A11

Language Specification

Lab Professor / Lab Session:

[Paulo Sousa / 011 / 012]

Team:

[Shreya Gulati] - Id: [041076329] / [Dhruv Hapani] - Id: [041080213]

Language Name [Gojo]

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

* 1. **User Manual**

**Element 1: Name / Extension**

*The name of our language is "Gojo".*

*The filename extension for the Gojo language is ".gj".*

*The Go language served as a major source of inspiration for Gojo. The design tenets of Go, which place an emphasis on clarity and readability, served as inspiration for this project. These guidelines make it easier for developers to comprehend and improve the codebase.*

**Element 2 – Comments**

*Gojo allows you to write both single-line and multi-line comments (using // and /\* \*/, respectively).*

**Element 3 – Keywords**

*default func interface case if for*

*go map struct else type import*

*goto package switch const continue return*

*These keywords, which are a part of the syntax of the Go language, are used in particular ways in the language.*

**Element 4 – Datatypes**

*Different data types, including texts, floating-point numbers, and integers, are used to represent different sorts of values in the Gojo programming language.*

1. *Integers:*

* *int8: 8 bits (1 byte), range from -128 to 127 (signed), or 0 to 255 (unsigned).*
* *int16: 16 bits (2 bytes), range from -32,768 to 32,767 (signed), or 0 to 65,535 (unsigned).*
* *int32 (also known as rune): 32 bits (4 bytes), range from -2,147,483,648 to 2,147,483,647 (signed), or 0 to 4,294,967,295 (unsigned).*
* *int64: 64 bits (8 bytes), range from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807 (signed), or 0 to 18,446,744,073,709,551,615 (unsigned).*

1. *Floating-Points Numbers:*

* *float32: 32 bits (4 bytes), approximately 7 decimal digits of precision.*
* *float64 (also known as float): 64 bits (8 bytes), approximately 15 decimal digits of precision.*

1. *Strings:*

* *Strings are sequences of Unicode characters.*
* *Variable-sized, based on the number of characters and UTF-8 encoding.*

**Element 5 – Variables**

1. *Integers Numbers:*

* *The var keyword, followed by the variable name and its data type, can be used to define a variable that stores an integer. You can use types like int, int8, int16, int32, or int64 for integers.*
* *Example: var myInteger int*

*myInteger = 42*

1. *Floating-Point Numbers:*

* *Similar to integers, you can define variables to hold floating-point numbers. Common types includes ‘float32’ and ‘float64’.*
* *Example: var myFloat float64*

*myFloat = 3.14159*

1. *Text:*

* *To define variables to hold text or strings, you can use the string data type.*
* *Example: var myText string*

*myText = “Hello, World”*

**Element 6 – Methods / Functions**

1. *Functions:*

* *A function in Gojo is a reusable block of code that performs a specific task.*
* *Functions can accept zero or more parameters, perform computations, and return a result.*
* *Example: func FunctionName(parameters) returnType{*

*//Function Body*

*return result*

*}*

1. *Methods:*

* *Methods in Gojo are functions associated with a specific data type, known as the method receiver.*
* *They allow you to define behavior or operations that are closely related to the data type.*
* *Example: func (receiver ReceiverType) MethodName(parameters) returnType {*

*//Method body*

*Return result*

*}*

**Element 7 - Commands**

* ***Attribution / assignment****:*

1. *Assignment Operator(=):*

* *The assignment operator = is used to assign a value to a variable.*
* *Example: var myVariable int*

*myVariable = 42*

1. *Short Variable Declaration(:=):*

* *The short variable declaration := allows you to declare and assign a value to a variable in a single line.*
* *Example: myVariable := 42*

1. *Casting:*

* *Gojo has explicit type conversion, often referred to as casting. You can convert between compatible types using a type conversion expression.*
* *Example: var myInt int = 42*

*var myFloat float64 = float64(myInt)*

1. *Math Operations:*

* *Gojo supports standard mathematical operations for numeric types. You can use operators like +, -, \*, /, and % (for remainder) to perform arithmetic operations.*
* *Example: a := 10*

*b := 5*

*sum := a + b*

*difference := a - b*

*product := a \* b*

*quotient := a / b*

*remainder := a % b*

1. *String Concatenation:*

* *Gojo allows you to concatenate strings using the + operator*
* *Example: firstName := "John"*

*lastName := "Doe"*

*fullName := firstName + " " + lastName*

* ***Selection****: In the Gojo programming language, you can implement if-style logic using the if statement for conditional execution of code blocks. Go also supports else if and else clauses for more complex conditional branching. Additionally, Go provides a switch statement for multi-way conditionals.*

1. *‘if’ statement:*

* *The basic structure of an ‘if’ statement in Gojo is*

*if condition {*

*// Code to execute when the condition is true*

*} else {*

*// Code to execute when the condition is false*

*}*

1. *Logical Operators:*

* *&& (logical AND): Returns ‘true’ if both conditions are ‘true’.*
* *|| (logical OR): Returns ‘true’ if at least one condition is ‘true’.*
* *! (logical NOT): Negates the condition.*
* *Example: a := true*

*b := false*

*if a && b {*

*// This block won't execute since both conditions are not true.*

*fmt.Println("Both conditions are true.")*

*} else if a || b {*

*// This block will execute since one of the conditions is true.*

*fmt.Println("At least one condition is true.")*

*} else {*

*// This block won't execute since neither condition is true.*

*fmt.Println("No conditions are true.")*

*}*

1. *Comparison Operators:*

* *== (equals)*
* *!= (not equals)*
* *< (less than)*
* *<= (less than or equal to)*
* *>= (greater than or equal to)*
* *Example:*

*x := 10*

*y := 5*

*if x == y {*

*fmt.Println("x is equal to y.")*

*} else if x > y {*

*fmt.Println("x is greater than y.")*

*} else {*

*fmt.Println("x is less than y.")*

*}*

1. *‘switch’ Statement:*

* *The switch statement in Gojo allows you to evaluate an expression against multiple possible values. It provides an elegant way to handle multi-way conditionals.*
* *Example: day := "Monday"*

*switch day {*

*case "Monday":*

*fmt.Println("It's Monday!")*

*case "Tuesday":*

*fmt.Println("It's Tuesday!")*

*default:*

*fmt.Println("It's some other day.")*

*}*

* ***Interaction****: In the Gojo programming language, you can handle looping using various constructs, primarily the for loop. Gojo's for loop is flexible and can be used for different types of loops, including the classic for loop, for range loop, and while-like loop.*

1. *Classic ‘for’ loop:*

* *The classic for loop in Gojo is similar to loops in many other programming languages. It has three components: initialization, condition, and post-expression.*
* *.Example: for initialization; condition; post-expression {*

*// Loop body*

*}*

1. *‘for’ Range Loop:*

* *The for range loop is used to iterate over elements of an array, slice, string, map, or channel.*
* *Example: for index, element := range collection {*

*// Loop body*

*}*

1. *‘for’ loop as ‘while’ -like loop:*

* *Gojo doesn't have a while loop in the traditional sense, but you can achieve similar behavior using the for loop with just a condition.*
* *Example: for condition {*

*}*

* ***Input****: In the Gojo programming language, you can read input from the keyboard (standard input) using the ‘fmt’ package's ‘Scan’ functions. The most commonly used function for reading strings from the keyboard is ‘fmt.Scanln’.*
* *Example: package main*

*import (*

*"fmt"*

*)*

*func main() {*

*var input string*

*fmt.Print("Enter a string: ")*

*fmt.Scanln(&input)*

*fmt.Printf("You entered: %s\n", input)*

*}*

* ***Output****: In Gojo, you can display output on the screen (standard output) using the fmt package's Print and Printf functions. These functions allow you to format and print data to the console.*

1. *Printing a String:*

* *Use ‘fmt.Print’ or ‘fmt.Println’ to display a string to the console. The difference between ‘Print’ and ‘Println’ is that ‘Println’ adds a newline character (\n) after the output.*

1. *Printing Variables:*

* *You can print the values of variables using fmt.Print or fmt.Println. To include variable values in the output, use placeholders and provide the variables as arguments to the Printf function.*

1. *Formatting Output:*

* *Gojo's ‘Printf’ function allows you to format output using format verbs (e.g., %s, %d, %f, etc.) and format flags (e.g., "%d" for decimal, "%x" for hexadecimal, "%f" for floating-point, etc.).*

1. *Print Special Characters:*

* *You can also print special characters, such as newline (\n) or tab (\t), to format your output.*

*Functions: [Function definition: parameters and returning types]*

*o What will be the syntax for making a function or subroutine?*

*o How will it take parameters?*

*o How will it return results?*

**Element 7 – Proper elements**

*[Include specific features / elements to be included in your language]*

* *Ternary By adding features that the Go language does not have, we hope to improve our language, Gojo. The Ternary Operator is one such feature. The beautiful implementation of this operator in Python, which enables short and understandable conditional formulations, served as our model. A ternary operation in Python can be expressed as follows:*

x = 10

message = "Even" if x % 2 == 0 else "Odd”

*The general syntax for Python’s ternary operator is:*

value\_if\_true if condition else value\_if\_false

*We intend to give developers a more expressive and condensed way to write conditional logic by using a similar strategy in Gojo, improving the code's general readability and maintainability.*

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

**Comparing with C language**

**Differences**

|  |  |  |
| --- | --- | --- |
|  | **1. Syntax and Simplicity**  Gojo: Gojo is created with readability and simplicity in mind. Compared to C, its syntax is simpler and shorter. It does away with several C features like pointer arithmetic and header files that can cause complexity and errors.  C: C is renowned for its simple syntax, but low-level features like pointers and manual memory management can make it harder to learn.  **2. Memory Management**  Gojo: Gojo has automatic garbage collection for memory, lowering the possibility of memory leaks and streamlining memory handling.  C: Memory allocation and deallocation are explicitly controlled by developers in C, which can result in efficient memory consumption but also increases the possibility of memory-related errors.  **3. Concurrency and Parallelism**  Gojo: Gojo makes it simpler to develop concurrent programmes by providing built-in support for concurrency via goroutines and channels. It excels at handling several jobs at once.  C: C lacks native capability for parallelism or concurrency. Concurrency can be more difficult to achieve and frequently requires platform-specific libraries.  **4. Compilation and Execution**  Gojo: Gojo uses a speedy compilation method, which leads to a faster execution of code. It directly compiles to machine code, doing away with the requirement for a middle step.  C: Unlike Go's compilation, C code normally goes through a compilation and linking procedure that can be slower. It gives the user additional power over the memory arrangement. |  |

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

|  |  |  |
| --- | --- | --- |
|  | Advantages of Go Compared to C:   1. **Simplicity and Readability:**   Gojo: Gojo was created with readability and simplicity in mind. Code writing and maintenance are made simpler by its clear and straightforward syntax.  C: Although C has a simple syntax, low-level features like pointers and manual memory management can make it harder to read.   1. **Error Handling:**   Gojo: Gojo promotes idiomatic error management with the use of return values that show errors. Its distinct approach to error checking makes error management simpler.  C: Due to the lack of specified error types, C relies on error codes and return values for error handling, which can be error-prone.   1. **Standard Libraries:**   Gojo: Gojo includes a robust standard library that covers a variety of features, including as file processing, networking, and the web. It promotes the usage of popular libraries for everyday tasks.  C: Compared to Go, C has a smaller standard library, and programmers frequently use third-party libraries for a variety of tasks.  Disadvantages of Go Compared to C:   1. **Low-Level Control:**   Gojo: Since Gojo encapsulates a lot of low-level information, it is less suitable for projects that need for precise hardware and memory control.  C: Where precise control is crucial, C shines in system programming, embedded systems, and device drivers.   1. **Performance:**   Gojo: While Gojo provides good performance, C code's lower level nature sometimes allows it to be optimised to run more quickly.  C: C enables programmers to optimise programmes for maximum performance by giving them fine-grained control over memory and hardware.   1. **Portability:**   Gojo: Gojo is made to be cross-platform compatible, but with good planning, C code can be highly portable.  C: C code is easily portable to different platforms, but it may take more extra work to make sure it is compatible. |  |

**Comparing with another language**

**Language Name: Python**

Python shines in ease of use, versatility, and a wide range of available libraries

**Differences**

|  |  |  |
| --- | --- | --- |
|  | 1. **Syntax:**   Gojo: Gojo is a statically typed, compiled language that places a strong emphasis on readability and simplicity. It lacks some of the syntactic capabilities present in dynamically typed languages and employs curly brackets for code blocks.  Python: Python is an interpreted language with dynamic typing that is renowned for its clear and expressive syntax. Code chunks are defined by indentation (whitespace).   1. **Ecosystem:**   Gojo: Gojo's ecosystem is expanding and is focused on system-level programming, microservices, and web development. It features a sizable standard library.  Python: Python has a developed and wide-ranging ecosystem for many different fields, such as web development, data research, and automation. It boasts a huge selection of external libraries and frameworks.   1. **Static vs. Dynamic Typing:**   Gojo: Gojo implements static typing, which can increase code stability by catching type problems at compile time.  Python: Python uses dynamic typing, which offers flexibility but necessitates extensive testing to identify type-related problems.   1. **Performance:**   Gojo: Gojo is renowned for its fast runtime and efficient compilation. It generates native machine code upon compilation, enabling quick execution.  Python: Python is an interpreted language, which means it often moves more slowly than Go. It can be enhanced using C/C++ modules, though, for tasks that demand performance. |  |

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

|  |  |  |
| --- | --- | --- |
|  | Advantages of Go Compared to Python:   1. **Strong Typing:**   Gojo: Gojo enforces strong typing, which aids in the compile-time detection of type-related issues.  Python: Because Python is dynamically typed, type problems can only be found during runtime.   1. **Static Binaries:**   Gojo: Gojo generates statically linked binaries, streamlining distribution and deployment.  Python: Python code frequently needs an interpreter and dependencies, which complicates deployment.   1. **Performance:**   Gojo: Gojo is a statically typed, compiled language that frequently executes more quickly than Python, which is an interpreted language.  Python: Python can be slower for some compute-intensive activities due to the interpreted nature of the language.  Disadvantages of Go Compared to Python:   1. **Generics (as of knowledge cutoff in September 2021):**   Gojo: The lack of generic functionality in Gojo could result in code duplication.  Python: Thanks to features like type hinting, Python offers improved support for generic programming.   1. **Lack of Dynamic Features:**   Gojo: Gojo's emphasis on simplicity and static type may prevent it from being used in some fields where dynamic features are required.  Python: Metaprogramming and flexible coding are made possible by Python's dynamic nature.   1. **Slower Development (for some use cases):**   Gojo: Because of its more rigid type system, writing code in Gojo could take longer at first.  Python: Python's dynamic typing can make prototyping go more quickly. |  |

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

**Advantages**

*The primary objective of our language, Gojo, is to offer a straightforward and effective language for creating dependable and scalable software. It tries to overcome the difficulties that developers encounter with regard to concurrency, performance, and language complexity. Go's benefits are most apparent in web development, cloud services, microservices, and system-level programming.*

*We tried to keep it as simple and fun as It could be so that it can be used for highest efficiency.*

**Strategy: C Implementation**

*[How your language can be implemented in C – ex: datatypes]*

* *Lexical analysis, syntax analysis, semantic analysis, code generation, and the conversion of Gojo data types to C data types are all processes in the process of implementing a portion of Gojo in C. Important factors to take into account include memory management, concurrent translation, using the standard library, aligning error systems, testing, and potential optimisations.*

*The inherent differences between the two languages, notably their memory and concurrency frameworks, make it a difficult task to completely translate the Gojo language to C.*

*[Your ideas about how to identify elements from language]*

* *The procedure includes lexical analysis to tokenize the code, syntax analysis to generate an abstract syntax tree, semantic analysis to check for correctness, and code generation to create C code in order to identify the components in Gojo code for a Gojo-to-C transpiler. The transpiler interprets "write to the console" commands like fmt.Println as function calls and maps them to their C equivalents. Text literals like "this is going to get printed" are translated to C character arrays and are considered as string literals. Text literals, variables, and function calls are processed correctly and written to the console in C thanks to the transpiler, which creates C code that mimics the behaviour of the original Gojo code.*

*[Your ideas about how to identify scope (ex: blocks between conditionals or functions)]*

* *Curly brackets are used to denote code blocks in Gojo. The beginning of a block is indicated by the opening brace, and its conclusion by the closing brace. Although it has no bearing on scope, indentation is advised for readability.*

*For controlling looped code, Gojo uses for loops enclosed in braces. Here's an example:*

*for i := 0; i < 5; i++ {*

*// Loop block*

*fmt.Println(i)*

*}*

*In this illustration, each time the loop iterates, the code contained in the loop block (between and), is looped through.*

**References**

|  |  |
| --- | --- |
|  | * *ChatGPT* |

Algonquin College

Fall, 2023