Algonquin College Logo

# SCHOOL OF ADVANCED TECHNOLOGY

### ICT - Applications & Programming

### Computer Engineering Technology – Computing Science



A11

Language Specification

Lab Professor / Lab Session:

[Prof. Sarah Khan / Session 303]

Team:

[Juhyeon Oh] - Id: [041023645] / [Taeyoung You] - Id: [041079981]

Due Date: Jan 24th, 2025

Language Name [FlowCode]

***This template is suggested (not mandatory) to answer A11 Specification.***

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

* *Note: Read the specification and use this template to create your language. The red comments below can be removed in your submission.*
  1. **User Manual**

**Element 1: Name / Extension**

* *Language Name: FlowCode*
* *Suffix: .fc*

**Element 2 – Comments**

*Comments: \*\* This is single line comment \*\**

*\*\* This is*

*Multi line*

*comment \*\**

* *The usage of single and multiple lines of comments is the same.*

**Element 3 – Keywords**

*Keyword list:*

* *Control Flow: if, then, elif, else, endif, repeat, check, break, continue*
* *Function: return, end, void*
* *Data types: int, string, bool, double*
* *Others: datatype[ ](array), cons, declaration, begin, import*

**Element 4 – Datatypes**

*[Datatypes]:*

* *Integer: int (i.e. int a, b, sum)*
* *String: string (i.e. string fruit = ‘Apple’)*
* *Boolean: bool (i.e. bool b = true)*
* *Double: double (i.e. double num)*

**Element 5 – Variables**

1. ***Variables***

***[Definition] =>*** *Variables in FlowCode are used to store data. All variables must be statically declared at the beginning of a program or block using the declaration keyword. Variables are not implicitly created during assignment; they must be explicitly declared with their data type before use. They must start with a lowercase letter or an underscore (\_) and can include letters, numbers, and underscores. Variable names are case-sensitive and cannot contain spaces or special characters (except for underscore(\_)).*

***[sequence of letters] =>*** *[a-z\_][A-Za-z0-9\_]\**

***[Examples]:***

*\*\* Valid variables \*\**

*declaration;*

*int x = 10*

*string fruit = ‘Apple’*

*bool flag = true*

*double d = 5.5*

*end;*

*\*\* Invalid variables \*\**

*int 1user = 10 \*\* Cannot start with a number \*\**

*string sweet-fruit = ‘strawberry’ \*\* Cannot include hyphen (-) \*\**

*string sour fruit = ‘lemon’ \*\* Cannot include spaces \*\**

*\*\* Error: all variables must be declared first. \*\**

1. ***Constants(cons)***

***[Definition] =>*** *Constants in FlowCode are immutable values defined using the `cons` keyword. The data type (e.g., int, double, string) must be specified explicitly, and the constant name should be unique and descriptive. In addition, it should start with upper-case letter or underscore(\_).*

***[sequence of letters] =>*** *[A-Z\_][A-Za-z0-9\_]\**

***[Examples]:***

*\*\* Valid constants \*\**

*declaration:*

*cons int MAX\_STUDENTS = 100*

*cons double PI = 3.14159*

*cons string FRUITS = ‘Apple’*

*end;*

*\*\* Invalid constants \*\**

*cons int 1STUDENT = 50 \*\* Cannot start with a number cons \*\**

*double PI VALUE = 3.14 \*\* Cannot include spaces \*\**

*cons string fruit = ‘Apple’ \*\* Use uppercase for constant names*

*\*\* Error: all variables must be declared first. \*\**

**Element 6 – Methods / Functions**

1. ***Functions***

* *Definition: In FlowCode, functions are defined using the* ***() keyword*** *after the function name, followed by the return type (e.g., void, int, double, etc.), the function name, and parameters. The function name should start with an uppercase letter. The function body is enclosed between : and end;. Functions can optionally return a value using the return keyword*
* *Syntax:*

*\*\* definition of function \*\**

*[return\_type] [function\_name]([parameters]):*

*\*\* function body \*\**

*end;*

* *Example:*

1. *Void function:*

*void SampleFunc():*

*\*\* code \*\**

*end;*

1. *Function with Return value:*

*int SumFunc(int a, int b):*

*return a + b*

*end;*

1. *Function with No Parameters:*

*string Hello():*

*return ‘Hello, World!’*

*end;*

1. *Function with Multiple Parameters:*

*double CalculateArea(double width, double height):*

*return width \* height*

*end;*

1. ***Function Call***

* *Definition: In FlowCode, functions are called using the function name followed by a colon (:) and their parameters, if any. A semicolon (;) is always required at the end of a function call to signify its completion. This makes function calls distinct from other statements in the language.*
* *Syntax:*

*\*\* function call \*\**

*[function\_name]: [parameters]; \*\* Void function*

*[variable] = [function\_name]: [parameters]; \*\* Function with return value \*\**

* *Example:*

1. *Calling a Void Function:*

*sampleFunc: void;*

1. *Calling a Function with Return Value:*

*sum = sumFunc: a, b;*

**Element 7 - Commands**

***- Variable Assignment:***

*In FlowCode, variables are assigned values using the* ***= operator****. All variables must be declared in the declaration block before they can be assigned a value.*

*Example:*

*declaration:*

*int a, b*

*string str*

*end;*

*a = 10 \*\* Assigning an integer value to `a`\*\**

*str = ‘hello’ \*\* Assigning a string value to `str`\*\**

***- Casting:***

*FlowCode allows implicit type casting. Explicit casting is required when converting between incompatible data types (e.g., int to double, or string to int).*

*Example:*

*declaration:*

*double d*

*int x*

*end;*

*x = 5*

*d = x \*\* implicitly converting int to double \*\**

***-* *Math Operations:***

*FlowCode supports basic arithmetic operations (+, -, \*, /, %, ^). Operations must involve compatible data types.,*

*Example:*

*declaration:*

*int num1, num2, sum*

*end;*

*num1 = 5*

*num2 = 10*

*sum = num1 + num2 \*\* Addition \*\**

***- String Concatenation:***

*Strings can be concatenated using* ***$ keyword within double-quotation****, allowing merging of string literals and variables.*

*Example:*

*declaration:*

*string firstName, lastName, fullName*

*end;*

*firstName = ‘John’*

*lastName = ‘Doe’*

*fullName = “$firstName $lastName” \*\* Merging strings with a space \*\**

* ***Selection***
* ***If-style:*** *FlowCode uses structured if statements for conditionals, clearly marked by if, then and terminated with endif;. The syntax supports elif and else for additional conditions. Boolean operators like and, or, not and xor are used for logical operations, and standard comparison operators (<, >, <=, >=, ==, !=) are supported for conditions.*
* ***Syntax:***

*\*\* if statement \*\**

*if condition then:*

*\*\* code \*\**

*endif;*

*elif condition then:*

*\*\* code \*\**

*endif;*

*else:*

*\*\* code \*\**

*endif;*

* ***Boolean Operations and Conditions \*\****
  + - ***Logical Operators****:*
      * *and: Logical AND*
      * *or: Logical OR*
      * *xor: Logical XOR*
      * *not: Logical NOT*
    - ***Supported Comparison Operators:***
      * *< : Less than*
      * *> : Greater than*
      * *<=: Less than or equal*
      * *>=: Greater than or equal*
      * *==: Equal to*
      * *!=: Not equal to*
* ***Interaction****:*
* ***Repetition:*** *FlowCode uses the repeat construct for loops, clearly marked with repeat: and terminated by util condition;. The syntax ensures clarity by explicitly defining the exit condition.*
* ***Syntax:***

*\*\* repeat statement \*\**

*check condition:*

*\*\* code \*\**

*repeat;*

* ***Input***
* *In FlowCode, user input is handled using the* ***input:*** *keyword. Multiple variables can be assigned values from a single line of input, separated by spaces. This allows for concise and flexible input handling.*
* ***Syntax:***

*Input: variable;*

*Input: variable1 variable2 variable3;*

* ***Output****:*
* *Output in FlowCode is handled using the* ***output:*** *keyword. It supports displaying strings, variables, and expressions. The* ***$*** *symbol can be used within double-quoted strings to embed variable values directly, simplifying string interpolation.*
* ***Syntax:***

*Output: 'string'; \*\* Single-quoted strings (static text). \*\**

*Output: expression;\*\* Outputs the result of an expression. \*\**

*Output: "string with $var."; \*\* Double-quoted strings with variable interpolation. \*\**

* ***Functions****:*
* *In FlowCode, functions are defined using the* ***() keyword****, followed by the return type, function name, and parameters. The function name should start with an uppercase letter. Functions that do not return a value use the void type, while those that return a value must explicitly specify the return type. Parameters are declared with their types and listed inside parentheses. The function body is enclosed between : and end;.*

**Element 7 – Proper elements**

* ***Use default package:*** *Ensure that some internal libraries, such as the "flowio" package, are already included but need to be explicitly called when used.*
* ***Mandatory Imports:*** *Import statements may be required for default Input/Output functionality.*

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

**Comparing with C language**

**Differences**

|  |  |  |
| --- | --- | --- |
|  | **FlowCode** | **C Language** |
| **Data Types** | - All variables are statically declared in the declaration section.  - Does not support pointers.  - Memory management is implicit and simplified. | - Variables can be declared anywhere in the code (since C99).  - Supports pointers for direct memory access.  - Provides greater flexibility but adds complexity. |
| **Statements** | No header files.  - All code must be written within a begin and end block.  - Ensures a clear structure. | - Requires header files.  - Explicit function declarations are necessary before use.  - Increases complexity for beginners |
| **Operations** | - No pointers or direct memory access.  - Simplified syntax.  - Focuses on high-level operations like Input:, Output:, and math functions. | - Supports low-level operations like pointer arithmetic and direct memory manipulation.  - Provides more control but has a steeper learning curve. |
| **Orthogonality** | - Only one way to perform key actions (e.g., loops use repeat exclusively, Input: and Output: use predefined keywords). | - Offers multiple ways to perform the same task (e.g., while, for, and do-while loops). |
| **Easy to use** | - Designed for simplicity.  - Pseudocode-like syntax ideal for education and beginners. | - Requires understanding of low-level concepts like memory allocation and pointers.  - Challenging for beginners. |

|  |  |
| --- | --- |
|  |  |

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

|  |  |  |
| --- | --- | --- |
|  | **Advantages of FlowCode**   1. **Simple Syntax**: Pseudocode-like structure; no pointers, header files, or manual memory management. 2. **Structured Flow**: begin/end blocks ensure clear program organization. 3. **Static Declaration**: All variables are declared upfront, reducing runtime errors. 4. **Built-in Input/Output**: Default commands Input: Output:) require no extra libraries. 5. **Education-Friendly**: Ideal for beginners; easy to learn and use.   **Disadvantages of FlowCode**   1. **Limited Scalability**: Not suitable for large or complex projects. 2. **No Pointers**: Simplifies memory management but restricts advanced functionality. 3. **Static Variables Only**: No dynamic memory allocation, less flexibility. 4. **Less Versatile**: Only one loop type (repeat), fewer options for advanced use. 5. **Lower Performance**: Not optimized for computation-heavy tasks compared to C. |  |

**Comparing with Java (or another language)**

**Language Name: Python**

**Differences**

|  |  |  |
| --- | --- | --- |
|  | **FlowCode** | **Python** |
| **Static vs Dynamic Typing** | Statically typed: Variables must be declared with a type in the declaration block. | - Dynamically typed: Variables are assigned, and their types are inferred at runtime. |
| **Execution Speed** | - Compiled: Potentially faster execution compared to Python. | - Interpreted: Generally slower due to runtime type inference and overhead. |
| **Object-Oriented Programming(OOP)** | - Does not support OOP; focuses on procedural programming. | - Fully supports OOP, including classes, inheritance, and polymorphism. |
| **Code Structure** | - Requires all code to be within begin and end blocks for clarity and structure. | - Allows code to be written anywhere, without strict structural constraints. |

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

|  |  |  |
| --- | --- | --- |
|  | **Advantages of FlowCode (Compared to Python)**   1. **Simpler Syntax**:  * Designed to mimic pseudocode, making it easier for beginners to learn and use.  1. **Faster Execution**:  * Being a compiled language, FlowCode can outperform Python in terms of execution speed for basic tasks.  1. **Structured Coding**:  * Enforces a clear program flow with begin and end blocks, improving readability and reducing errors.  1. **Education-Oriented**:  * Ideal for teaching programming basics without overwhelming learners with advanced features like OOP.  1. **Static Typing**:  * Catches type errors at compile-time, ensuring fewer runtime bugs.   **Disadvantages of FlowCode (Compared to Python)**   1. **Limited Flexibility**:  * Requires strict structure and static typing, which can feel restrictive compared to Python's dynamic and flexible nature.  1. **No OOP Support**:  * Lacks advanced features like classes and objects, making it unsuitable for complex, scalable applications.  1. **Reduced Ecosystem**:  * Unlike Python, FlowCode has no extensive libraries or frameworks for domains like data science, machine learning, or web development.  1. **Scalability**:  * Better suited for small, educational projects rather than large-scale software development.  1. **Development Speed**:  * Python’s dynamic nature and rich libraries enable faster development for complex tasks compared to FlowCode’s manual structure. |  |

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

**Advantages**

***Goals and Purpose:***

* ***Educational Focus****: FlowCode is designed for beginners, focusing on simplicity, readability, and ease of learning.*
* ***Structured Logic****: Inspired by pseudocode, FlowCode enforces structured programming principles through blocks (begin, end) and clear syntax. It helps beginners access algorithm thinking*
* ***Efficient Execution****: As a compiled language, FlowCode aims for faster execution than interpreted languages like Python.*

***Advantages:***

1. ***Easy to Use:*** *The language is simple and uses clear, easy-to-understand commands like input: for getting user input and output: for showing results. This makes it perfect for beginners.*
2. ***Safe and Reliable****: Since you must define the type of every variable (like int or string) before using it, the language helps catch mistakes early, before running the program.*
3. ***Ready-to-Use Features****: Many common functions (like printing, math operations, and loops) are already built into the language, so you don’t need to import extra libraries to use them.*
4. ***Consistent and Clear:*** *There’s only one way to do things like loops or conditionals, making the language easier to learn and write without confusion.*

***Challenges:***

1. *How to handle auto-casting while considering data type sizes and floating-point numbers.*
2. *How to correctly designate the appropriate format specifiers.*

**Strategy: C Implementation**

* *Data types in C:*

*int -> C’s int (4 bytes)*

*double -> C’s double (8 bytes)*

*string -> C’s char[] (dynamic size)*

*bool -> C’s \_Bool or int*

1. *How this language should be built (using C language)*
   * + *Determine the necessary data types for the new language and map each type to its corresponding C data type.*
2. *What could be your strategy to identify the elements of the language (“tokens”)*
   * + *Define identifiers, operators, constants, keywords, reserved words, and delimiters, and create a token stream container to manage them.*
3. *How to define a scope for iteration/conditionals*
   * + *Use within : to start the scope and end; to close it.*

***Note 1: C Datatypes***

*Remember that you are implementing your language in ANSI C. For this reason, you cannot create arbitrarily your language (from scratch). You need to use what is already provided by C Compiler. For this reason, think about using and defining the language obeying the datatypes.*

**FINAL SUGGESTIONS**

*Here some ideas to think about your language....*

* *Don't make this assignment harder than it needs to be on yourself. Focus on making the syntax for your language that meets our requirements. Worry about extra features later.*
* *Don’t worry if your new language winds up having really difficult parts. You'll be allowed to change your language as you go along, as long as you make "patch notes" to explain those changes. We'll tell you about this later.*
* *Finally, think about creating an “master-piece”: until now, you have used several languages. And if you have conditions to define yours, how it could be?*

**References**

|  |  |
| --- | --- |
|  | *Wikipedia Contributors. “Pseudocode.” Wikipedia, Wikimedia Foundation, 9 Oct. 2019, en.wikipedia.org/wiki/Pseudocode.*  *Wikipedia Contributors. “Flowchart.” Wikipedia, Wikimedia Foundation, 29 Aug. 2019, en.wikipedia.org/wiki/Flowchart.*  *Wikipedia Contributors. “ALGOL.” Wikipedia, Wikimedia Foundation, 14 Sept. 2019, en.wikipedia.org/wiki/ALGOL.*  *Neso Academy. “Lexical Analyzer – Tokenization.” YouTube, 14 Apr. 2022, www.youtube.com/watch?v=MZ9NZdZteG4&list=PLBlnK6fEyqRjT3oJxFXRgjPNzeS-LFY-q&index=6. Accessed 17 Jan. 2025.* |

**Github Link**

|  |  |
| --- | --- |
|  | Repository: <https://github.com/TaeyoungYou/Flowcode>  Github project: <https://github.com/users/TaeyoungYou/projects/6> |

Algonquin College

Winter, 2025