**Structured Language**

Variable and function names cannot consist of the following:

1. Numbers (e.g. one, two, ten)
2. Variable types (e.g. String, character, integer)
3. Comparison operators (e.g. less than, greater than, less than equal, not equal)
4. Logical operators (e.g. and, or)
5. Unary operators (“minus minus”, “plus plus”, “not”, “negative”)
6. Any keywords (e.g. "array", "begin", "call", "character", "condition", "create", "declare", "divide", "double", "else", "end", "equal", "float", "for", "function", "greater", "if", "index", "integer", "less", "long", "loop", "minus", "modulo", "not", "parameter", "plus", "return", "size", "string", "than", "then", "times", "type", "void", "while", "with")

**Declare variable**

Syntax: declare <variable type> <variable name> end declare  
Example: declare integer max end declare  
Code: int max;

Syntax: declare <variable type> <variable name> equal <literal> end declare  
Example: declare integer max equal five end declare  
Code: int max = 5;

**Declare array**

Syntax: declare <variable type> array <variable name> size <number> end declare  
Example: declare integer array sequence size ten end declare  
Code: int sequence[10];  
Optional: Can add in “with” before “size” keyword (i.e. declare integer array sequence ***with*** size ten end declare)

**Variable assignment**

Syntax: <variable name> equal <literal> end equal  
Example: max equal ten end equal  
Code: max = 10;

**Array assignment**

Syntax: <variable name> array index <number> equal <literal> end equal  
Example: sequence array index three equal four end equal  
Code: sequence[3] = 4;

**If – then – else**

Note 1: Logical “and” and logical “or” is not implemented yet.  
Note 2: <statements> refer to any other statement, e.g. variable declaration statement.  
Note 3: When dealing with nested if-loops, use end if to close the if loop

Syntax: begin if <statements> then <statements> end if  
Example: begin if b less than two then b equal one end equal end if  
Code: if (b < 2) { b = 1; }

Syntax: begin if <statements> then <statements> else <statements> end if  
Example: begin if b less than two then b equal one end equal else a equal one end equal end if  
Code: if (b < 2) { b = 1; } else { a = 1; }

**Declare function**

Syntax: create function <function name> return type <variable type> <0 or more: parameter <variable type> <variable name>> begin <statements> end function  
Example: create function hit return type void parameter float damage begin end function  
Code: void hit(float damage){ }  
Optional: Can add in “with” before “return type” keyword and/or “parameter” keyword

More examples:

create function find maximum with return type integer with parameter integer length with parameter integer array numbers begin end function

create function main with return type void begin declare integer a equal one end declare end function

**Call a function**

Syntax: call function <function name> <0 or more: parameter <variable name>> end function   
Example: call function search parameter lower parameter higher end function  
Code: search(lower,higher);  
Optional: Can add in “with” before “parameter” keyword

**While loops**

Syntax: while <variable name> begin <statements> end while   
Example: while is cold begin end while  
Code: while (isCold) { }

Syntax: while <variable name> <comparison operator> <variable/literal> begin <statements> end while   
Example: while counter less than two begin counter equal counter minus one end equal end while  
Code: while (counter < 2) { counter = counter - 1; }

**For loops**

Syntax: for loop condition <variable name> equal <literal> condition <variable name> <comparison operator> <literal> condition <variable name> <operator> begin <statements> end for loop   
Example: for loop condition i equal one condition i  
less than length condition i plus plus begin end for loop  
Code: for (i = 1; i < length; i++) { }

**Literals**

Integer: Just verbalize the number (e.g. ten 🡪10, twenty two 🡪 22)

Float: Verbalize number, using “point” for decimal place (e.g. thirty four point two 🡪 34.2)

Character: Say “Character” followed by an alphabet. (e.g. character x 🡪 ‘x’)

String: Syntax: string <string content> end string (e.g. string hello end string 🡪 “hello”)

Note: There is no string variables in C, but we can pass string literals into function calls like printf.