

## Pseudocode Translation Quick Reference Guide

Category	Pseudocode	C/C++ Source Code Equivalent
(Basic) Data types	Integer Floating Point Character	int, long float, double char
Module declarations	<b>moduleName</b> (parameter list if any) Statement(s) <b>END</b>  <i>NB: Parameter list should also indicate whether the parameter(s) is being received or returned e.g.</i>  SubAlgoritmTotal(receives num1 and num2, returns total)	<b>void</b> moduleName(parameter list if any) { Statement(s); }  OR datatype moduleName(parameter list if any) { Statement(s); <b>return value;</b> }
Variable declaration	Let <i>variableName1</i> store a <i>datatype</i> value	<i>datatype</i> variableName1;
Constant declaration	CONSTNAME = value  <i>NB: It is convention to use all uppercase characters when naming constants to distinguish them from variables.</i>	Const <i>datatype</i> CONSTNAME = value;
Assigning Values	<b>SET</b> variableName <b>TO</b> value OR VariableName = value	VariableName = value;
Conditional Operators	operand1 = operand2 operand1 <b>NOT</b> = operand2 operand1 < operand2 operand1 > operand2 operand1 <= operand2 operand1 >= operand2  <i>NB: The alternative is to express the operators in text e.g.</i> <i>operand1 greater than or equal to operand2</i>	(operand1 == operand2) (operand1 != operand2) (operand1 < operand2) (operand1 > operand2) (operand1 <= operand2) (operand1 >= operand2)
Logical Operators	operand1 <b>AND</b> operand2 operand1 <b>OR</b> operand2 <b>NOT</b> operand1	(operand1 && operand2) (operand1    operand2) !(operand1)
Selection	<b>IF</b> (condition) <b>THEN</b> True statement(s) block <b>ELSE</b> False statement(s) block <b>ENDIF</b>	if (condition) { True statement(s) block; } else { False statement(s) block; }
Case/Switch	<b>CASE OF</b> single_variable value_1 : statement block_1 value_2 : statement block_2 ... value_n : statement block_n value_other : statement block_other <b>ENDCASE</b>	<b>switch</b> (single_variable) { <b>case</b> value_1 : statement(s); <b>break;</b> <b>case</b> value_2 : statement block_2; <b>break;</b> ... <b>case</b> value_n : statement block_n; <b>break;</b> <b>default</b> value_other : statement block_other; }

WHILE-DO Repetition	<b>DOWHILE</b> (condition) statement(s) <b>ENDDO</b>	<b>while</b> (condition) { statement(s); }
DO-WHILE Repetition	<b>DO</b> statement(s) <b>WHILE</b> (condition)	<b>do</b> { statement(s); } <b>while</b> (condition);
Repeat-Until Repetition	<b>Repeat</b> statement(s) <b>Until</b>	<b>do</b> { statement(s); } <b>while</b> !(condition);
For loop Repetition	<b>DO</b> counter = begin <b>TO</b> end statement(s) <b>ENDDO</b>	<b>for</b> (counter=begin; counter<=end; counter++) { statement(s); }
Record Structures	recordName fieldname1 fieldname2 ... fieldnameN	<b>struct</b> recordName { datatype fieldname1; datatype fieldname2; ... datatype fieldnameN; }
Sequential Files - READING	Initial processing <b>READ</b> variableName <b>DOWHILE NOT EOF</b> ... <b>READ</b> variableName <b>ENDDO</b> Final processing	<b>#include &lt;fstream.h&gt;</b> ... <b>ifstream</b> inFileHandle; ... inFileHandle. <b>open</b> ("sourcefile"); <b>while</b> (!inFileHandle. <b>eof</b> ()) { ... inFileHandle >> variableName; } inFileHandle. <b>close</b> ();
Sequential Files - WRITING	<b>WRITE</b> variableName	<b>#include &lt;fstream.h&gt;</b> ... <b>ofstream</b> outFileHandle; ... outFileHandle. <b>open</b> ("sourcefile"); ... outFileHandle << variableName << " "; ... outFileHandle. <b>close</b> ();
DECLARING 1D	<b>SET</b> arrayName(maxNumElements)	datatype arrayName[maxNumElements];
Arrays Declaring 2D	<b>SET</b> arrayName(rows, columns)	datatype arrayName[rows][columns];
Array Processing ASSIGNING A VALUE	arrayName(index) = value	arrayName[index] = value;
Array Processing READING A VALUE	VariableName = arrayName(index)	variableName = arrayName[index];

*Adapted from Simple Programming Design, 4<sup>th</sup> Edition, by Lesley Anne Robertson*