RAPTOR Syntax and Semantics  By Lt Col Schorsch

Program - an ordered collection of instructions that, when executed, causes the computer to behave in a predetermined manner.

Variable - A variable names a memory location. By using that variable’s name you can store data to or retrieve data from that memory location.

A variable has 4 properties:  a name,  a set of values,  a notation for literals of those values,  and operations and functions which can be performed on those values.

RAPTOR has two simple data types: Number and String (Array data types are described later)

<table>
<thead>
<tr>
<th>Type name</th>
<th>Literal Values</th>
<th>Operations grouped from lowest to highest precedence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>-32, 0, 1, 49, etc. -2.1, 3.1415, etc.</td>
<td>[=, &lt;, &lt;&gt;, &gt;=, /=, ! =, ], +, [/ , rem, mod], [**]</td>
</tr>
<tr>
<td>String</td>
<td>&quot;Hello&quot;, &quot;Bob&quot;, etc.</td>
<td>[=, &lt;, &lt;&gt;, &gt;=, /=, ! =, ], +</td>
</tr>
</tbody>
</table>

The following operators are only used in decisions (see Selection and Iteration)

<table>
<thead>
<tr>
<th>Relational operators:</th>
<th>and, or, not, xor</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;, &lt;=, &gt;, &gt;=</td>
<td>True and True True</td>
</tr>
<tr>
<td>=, =, !=</td>
<td>True and False False</td>
</tr>
<tr>
<td>&lt;, &lt; &gt;, &gt;=</td>
<td>False and False False</td>
</tr>
<tr>
<td>and, or, not, xor</td>
<td>False and True False</td>
</tr>
</tbody>
</table>

Function - A function performs a computation on data and returns a value.

Functions use parentheses to indicate their data (i.e. sqrt(4.7), sin(2.9), etc.)

Basic math: sqrt, log, abs, ceiling, floor
- sqrt returns the square root, ex sqrt(4) is 2
- log returns the natural logarithm, ex log(1) is 1
- abs returns the absolute value, ex abs(-9) is 9
- ceiling rounds up to a whole number, ex ceiling(3.14159) is 4

Trigonometry: sin, cos, tan, cot, arcsin, arccos, arctan, arccot
- Angles are in radians, ex sin(pi) is 0.
- arcsin and arccos are the two parameter versions of those functions.
- arctan and arccot are written in RAPTOR as arctan(X/Y).

Miscellaneous: Length, Of, Random
- Length returns the number of characters in a string ex Name ← "Stuff" followed by Length_Of(Name) is 5
- Random (Random * X + Y extends the range by X and shifts it by Y)

Procedure Call - A procedure is a set of executable statements that have been given a name.

Calling a procedure executes the statements associated with that procedure.

Procedure_name (Parameter 1, Parameter 2, etc.)

the number and order of parameters in the call must match the expected number and order.

The data types of the parameters in the call must match the expected data types of the parameters.

Procedure parameters can be used to give (supply) a procedure with data or can accept (receive) data.

Parameters must be variables if they receive a value.

Parameters can be an expression (computation), variable or literal if they supply a value.

Procedure_Name(P1, P2)
RAPTORGraph is a collection of procedures and functions that a RAPTOR programmer can use to create a graphics window, draw and animate graphical objects in that window, and interact with the graphics window using the keyboard and mouse.

**Procedure calls** occur only in call symbols.

**Function calls** return a value and therefore can occur anywhere a value can occur. (i.e. in assignment, decision, and output statements and as procedure call parameters.)

### Graphic window opening and closing procedures
- Open_Graph_Window( X_Size, Y_Size )
- Close_Graph_Window

### Graphic window “size” functions
- Get_Max_Width -> returns available screen pixel width
- Get_Max_Height -> returns available screen pixel height
- Get_Window_Width -> returns current window pixel width
- Get_Window_Height -> returns current window pixel height

### Drawing procedures
- Put_Pixel( X, Y, Color )
- Draw_Line( X1, Y1, X2, Y2, Color )
- Draw_Box( X1, Y1, X2, Y2, Color, Filled/Unfilled )
- Draw_Circle( X, Y, Radius, Color, Filled/Unfilled )
- Draw_Ellipse( X1, Y1, X2, Y2, Color, Filled/Unfilled )
- Draw_Arc( X1, Y1, X2, Y2, StartX, StartY, EndX, EndY, Color )
- Clear_Window( Color )
- Flood_Fill( X, Y, Color )
- Display_Text( X, Y, String Expression, Color )
- Display_Number( X, Y, Number Expression, Color )

### Mouse input procedures
- Wait_for_Mouse_Button( Which_Button )
- Get_Mouse_Button( Which_Button, X, Y )

### Mouse input functions
- Mouse_Button_Pressed( Which_Button ) -> returns True / False
- Get_Mouse_X -> returns X coordinate of mouse location
- Get_Mouse_Y -> returns Y coordinate of mouse location

### Keyboard input procedure
- Wait_For_Key

### Keyboard input functions
- Key_Hit -> returns True / False (whether a key was pressed)
- Get_Key -> returns the numeric ASCII value of the pressed key
- Get_Key_String -> returns a string value of the pressed key

### Graphics window query function
- Get_Pixel( X, Y ) -> returns the number code for the color of the pixel at (X, Y)

### How to animate an object in RAPTORGraph
Place the following inside of a loop
- Draw some an object relative to an X,Y point with the drawing procedures
- Delay_For some small time period
- Draw the object again in white (i.e. erase it)
- Update the X,Y point where you are drawing by some small offset
Decision - A decision is part of a Selection or Iteration (loop) statement. A decision symbol (its value during execution) determines which way execution will continue. Use relational operators (and logical operators) to get a Boolean value for the decision.

Relational: =, <, <=, >, >=, /=, !=

Logical: and, or, not, xor

Selection Statement - A selection statement is used to decide whether or not to do something, or to decide which of several things:

If the Boolean Expression is TRUE, execute the left hand path
otherwise execute the right hand path

If the value of the variable GPA is greater than 3.0
then execute the statement
Put("Dean’s List")
otherwise do nothing

If a student’s GPA is less than 2.0
then execute the statement
Put("Academic Probation")
otherwise execute the statement
Put("Cadet in good standing")

Iteration Statement (loop statement) – An Iteration statement enables a group of statements to be executed more than once. Use I.T.E.M (Initialize, Test, Execute, and Modify) to ensure your loop (and loop control variable) are correct.

A Sentinel Controlled Loop repeats its statements until the sentinel value is entered.

A Counter Controlled Loop repeats its statements a fixed number of times. This executes the loop 100 times because of the decision: Count >= 100.

This last example requires several decision statements as there are several decisions (more than two possible paths). The code assigns a nominal “grade” based on a student’s GPA. The “pattern” of these selection statements is called cascading selections.
Array variable - Array variables are used to store many values (of the same type) without having to have many variable names. Instead of many variables names a count-controlled loop is used to gain access (index) the individual elements (values) of an array variable.

RAPTOR has one and two dimensional arrays of numbers. A one dimensional array can be thought of as a sequence (or a list). A two dimensional array can be thought of as a table (grid or matrix).

To create an array variable in RAPTOR, use it like an array variable. i.e. have an index, ex. Score[1], Values[x], Matrix[3,4], etc.

All array variables are indexed starting with 1 and go up to the largest index used so far. RAPTOR array variables grow in size as needed.

The assignment statement

\[ \text{GPAs}[24] \leftarrow 4.0 \]

assigns the value 4.0 to the 24th element of the array GPas. If the array variable GPAs had not been used before then the other 23 elements of the GPAs array are initialized to 0 at the same time. i.e. The array variable GPAs would have the following values:

1 2 3 4 ...
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

The initialization of previous elements to 0 happens only when the array variable is created. Successive assignment statements to the GPAs variable affect only the individual element listed. For example, the following successive assignment statements

\[ \text{GPAs}[20] \leftarrow 1.7 \]
\[ \text{GPAs}[11] \leftarrow 3.2 \]

would place the value 1.7 into the 20th position of the array, and would place the value 3.2 into the 11th position of the array. i.e.

\[ \text{GPAs}[20] \leftarrow 1.7 \]
\[ \text{GPAs}[11] \leftarrow 3.2 \]

An array variable name, like GPAs, refers to ALL elements of the array. Adding an index (position) to the array variable enables you to refer to any specific element of the array variable.

Two dimensional arrays work similarly, i.e. Table[7,2] refers to the element in the 7th row and 2nd column.

Individual elements of an array can be used exactly like any other variable. E.g. the array element GPAs[5] can be used anywhere the number variable X can be used.

The Length_Of function can be used to determine (and return) the number of elements that are associated with a particular array variable.

For example, after all the above, Length_Of (GPAs) is 24.