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: (1) a name, (2) a memory location, (3) a data type, (4) a value. You can assign a value to a variable using an assignment statement (see below). RAPTOR variables are declared on first use, they must be assigned a value on first use and based on that value it’s data type will be Number, String, or an Array of Numbers.

Data Type - A Data Type is the name for a group of data values with similar properties. A Data Type has 4 properties: (1) a set of values, (2) a notation for those values, (3) operations and functions which can be performed on those values.

Assignment Statement - An assignment statement is used to evaluate an expression and store the results in a variable. The expression is on the right hand side of the assignment operator, =

Operator – An operator directs the computer to perform some computation on data. Operators are placed between the data (operands) being operated on (i.e. X / 3, Y + 7, N < M, etc.)

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.).

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.

Variable Syntax

Assignment Statement

Operator Syntax

Data Type Syntax

Function Syntax

Procedure Call Syntax

Symbols:

- Assignment
- Call
- Input
- Output
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

### 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

### 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
- **Mouse_Button_Released( Which_Button )** -> returns True / False
- **Get_Mouse_X** -> returns X coordinate of mouse location
- **Get_Mouse_Y** -> returns Y coordinate of mouse location

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

### RAPTORGraph Colors
- Black, Blue, Green, Cyan, Red, Magenta, Brown, Light_Gray, Dark_Gray, Light_Blue, Light_Green, Light_Cyan, Light_Red, Light_Magenta, Yellow, White

### 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
**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 Condition Controlled Loop (basic loop) repeats its statements until a condition (the decision statement) becomes true.
The validation loop above will continue to execute until the user enters a number between 1 and 10.
Number is the loop control variable.

A Count Controlled Loop repeats its statements a fixed number of times.
(This executes the loop 100 times because of the decision: \( \text{Count} \geq 100 \)).
The count controlled loop above executes exactly 10 times (it displays the numbers 1 through 10 and the squares of those numbers). \( \text{Count} \) is the loop control variable.

**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")

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

$$GPAs[24] ← 4.0$$

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:

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>…</th>
<th>…</th>
<th>23</th>
<th>24</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The initialization of previous elements to 0 happens only when the array variable is created. Successful assignment statements to the GPAs variable affect only the individual element listed.

For example, the following successive assignment statements

$$GPAs[20] ← 1.7$$
$$GPAs[11] ← 3.2$$

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. $GPAs[20] ← 1.7$

1 2 3 4 ... ... 23 24

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0.00

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.