**Core Requirements**

|  |  |  |  |
| --- | --- | --- | --- |
| **Variables** |  |  |  |
| Numeric | String | Array |  |
| Int - 4 byte integer  Float - 8 byte floating pt (double) | String   * Values not reused * Mutable * character is one byte   Bool: 1 byte T or F | * [] * Homogeneous * Int, Float, Bool or String * Allocated Size * Highest element populated | We will support explicit declarations. |
| **Constants** |  |  |  |
| Type Examples  Int 100, -5, -1, 30, 0  Float 0.5, -5.0, 0.0, 1.25, 123.45  We are not handling E. | String Constants:  "Can't"  'Bob said, "Hi"'  'Pete said, "I can\'t"'  "\t\tEnd\n"  "2017-05-02"  Bool Constants: T, F |  |  |
| **Comments** |  |  |  |
|  |  |  | // until end of line |
| **Operators** |  |  |  |
| Numeric | String | Array | Generic (dependent on left operand) |
| + binary plus  - binary minus, unary minus  \* multiplication  / division  ^ exponentiation  The data type of the expression operation is based on the datype of the left operand (or the single operand for unary). | # concatenate  [0] first character  [*-*1] last character  [*subscript*] character at *subscript*  must check boundaries  Note: if doing only prefix, use  [*string* *subscript*] | [0] first element  [*-*1] last element  [*subscript*] element at *subscript*    must check boundaries (even when doing the compiler)  Note: if doing only prefix, use  [*array* *subscript*] | == equals  < less than  > greater than  >= greater than or equal to  <= less than or equal to  != not equal  Note: that a numeric comparison is different from a string comparison (e.g., "3" > "25" whereas 3 < 25)  The data type of the comparison is based on the first operand. |
| **logical operators**  and  or  not |  |  |  |
| **Assignment**  iVal = pi \* 2 \* radius;  A Float value is truncated when it is assigned to an Int. | = copies the value  sVal = first # " " # last;  sVal = iVal + 5;  Assignment of a numeric to a string will use coercion rules.  Values may be truncated depending on the declared size of the target. | array = array;  array = 0;  Arrays support array to array and scalar assignment- assign that value to every element.  Assignment of an array of a different length will vary dependent on flexible requirements. |  |
| **Functions** |  |  |  |
| Built-in: | Built-in:  LENGTH(*string*)  returns the number of characters in *string.*  SPACES(*string)*  returns T if the string is empty or only contains spaces.  MAXLENGTH(*string*)  returns the declared length of the string which is the maximum number of characters. | Built-in:  ELEM(*array*)  returns the subscript of highest populated element + 1  MAXELEM(*array*)  declared number of elements |  |
| **Control Flow Statements** |  |  |  |
| for *cv* = *sv* to *condition* by *incr*:  *body*  endfor; | for *char* in *string*:  *body*  endfor; | for *item* in *array*:  *body*  endfor; |  |
|  |  |  | while *condition* :  *body*  endwhile; |
|  |  |  | if *condition* :  *body*  endif; |
|  |  |  | if *condition* :  *body*  else  *body*  endif; |
| **Print** |  |  |  |
| Assume coercion rules. | print (*strExpr, ...*); | n/a | print one space between items listed  assume \n after print |
| **Datatype Coercion** |  |  |  |
|  | Int to string  Double to string (assume %.2lf) |  |  |