Inform in four minutes	Use parentheses () to control the order of evaluation. Arithmetic/logical expressions support these operators: p + q addition	Named word values, unchanging at run-time, which are by default initialised to zero: Constant constant;
A quick reference to the Inform programming language Inform is copyright © 2002 by Graham Nelson http://www.gnelson.demon.co.uk/ This guide is copyright © 2002 by Roger Firth http://www.firthworks.com/roger/ Version 1.3 (March 2002)	p - qsubtraction $p + q$ subtraction $p * q$ multiplication p / q integer division $p % q$ remainder $p++$ increments p , evaluates to original value	Constant constant = expr; Standard constants are true (1), false (0) and nothing (0) also NULL (-1). To define a constant (unless it already exists): Default constant expr;
	$\begin{array}{llllllllllllllllllllllllllllllllllll$	•• Variables and Arrays ••••••••••••••••••••••••••••••••••••
The road to brevity is via imprecision and through solecism – refer to the <i>Inform Designer's Manual</i> for the definitive story.	$ \begin{array}{ll} \rho \& q & \text{bitwise AND} \\ \rho \mid q & \text{bitwise OR} \end{array} $	are by default initialised to zero. A global variable is a single word:
•• Literals ••••••••••••••••••••••••••••••••••••	<pre>~p bitwise NOT (inversion) Conditional expressions return true (1) or false (0);</pre>	Global <i>variable</i> ; Global <i>variable</i> = <i>expr</i> ;
A Z-code word literal uses sixteen bits (whereas a Glulx word has thirty-two bits). A byte literal is always eight bits.	q may be a list of alternatives q1 or q2 or qN: p == q p is equal to q	A word array is a set of global words accessed using <i>array</i> >0, <i>array</i> >1, <i>array</i> >(<i>N</i> -1):
• Decimal: -32768 to 32767 Hexadecimal: \$0 to \$FFFF Binary: \$\$0 to \$\$1111111111111	$p \rightarrow q \qquad p \text{ is equal to } q$ $p \rightarrow q \qquad p \text{ is n't equal to } q$ $p \rightarrow q \qquad p \text{ is greater than } q$	Array array> N; Array array> expr1 expr2 exprN; Array array> "string";
 Action: ##Look Character: 'a' Dictionary word: 'aardvark' (up to nine characters 	$\rho < q$ ρ is less than q $p >= q$ ρ is greater than or equal to q $\rho <= q$ ρ is less than or equal to q	A table array is a set of global words accessed using <i>array</i> >1, <i>array</i> >2, <i>array</i> > <i>N</i> , with <i>array</i> >0 initialised to <i>N</i> :
significant); use circumflex "^" to denote apostrophe. Plural word: 'aardvarks//p' Single-character word: "a" (name property only) or 'a//'	p ofclass qobject p is of class qp in qobject p is a child of object qp notin qobject p isn't a child of object q	Array array table N; Array array table expr1 expr2 exprN; Array array table "string";
 String: "aardvark's adventure" (maximum around 4000 characters); can include special values including: ^ newline 	p provides qobject p provides property qp has qobject p has attribute qp hasnt qobject p hasn't attribute q	A byte array is a set of global bytes accessed using array->0, array->1, array->(N-1): Array array -> N:
~ quotes " " " @@64 at sign "@"	Boolean expressions return true (1) or false (0):	Array array -> expr1 expr2 exprN; Array array -> "string";
@@92 backslash "\" @@94 circumflex "^" @@126 tilde "~"	$p \&\& q$ both p and q are true (non-zero) $p \mid \mid q$ either p or q is true (non-zero) $\sim p$ p is false (zero)	A string array is a set of global bytes accessed using <i>array</i> ->1, <i>array</i> ->2, <i>array</i> ->N, with <i>array</i> ->0 initialised to <i>N</i> :
 a with a grave accent "à", et al pound sign "£", et al 	To return -1 , 0 or 1 based on unsigned comparison: UnsignedCompare(p , q)	Array <i>array</i> string N; Array <i>array</i> string <i>expr1 expr2 exprN</i> ; Array <i>array</i> string " <i>string</i> ";
@00 @31 low string 031	To return true if object q is a child or grand-child or of p: IndirectlyContains(p,q)	In all these cases, the characters of the initialising <i>string</i> ar unpacked to the individual word/byte elements of the array
The identifier of an Inform <i>constant</i> , <i>variable</i> , <i>array</i> , <i>class</i> , <i>object</i> , <i>property</i> , <i>attribute</i> , <i>routine</i> or <i>label</i> . Up to 32 characters: alphabetic (case not significant), numeric and underscore, with the first character not a digit.	To return a random number 1 <i>N</i> , or one from a list of constant values: random(<i>N</i>) random(<i>value</i> , <i>value</i> , <i>value</i>)	See also Objects (for property variables) and Routines (for local variables).

•• Classes and Objects •••••••• •• Manipulating the object tree ••••••• •• Routines ••••• To declare a *class* – a template for a family of objects – A routine can have up to 15 **local variables**: word values To change object relationships at run-time: which are private to the routine and which by default are set where the optional (N) limits instances created at run-time: move *object* to *parent object*; to zero on each call. Recursion is permitted. remove *object*; Class class(N)class class class ... class A standalone routine: To return the parent of an object (or nothing): has attr def attr def ... attr def • has a name, by which it is called using *routine(*); can parent(object) with prop def, also be called indirectly using To return the first child of an object (or nothing): . . . indirect(*routine*, *a*1, *a*2, ... *a*7) prop def; child(*object*) • can take arguments, using routine(a1, a2, ... a7), To declare an *object*; "Object" can instead be a *class*, the To return the adjacent child of an object's parent (or whose values initialise the equivalent local variables remaining four header items are all optional. and arrows nothing): (->, -> ->, ...) and *parent object* are incompatible: returns true at the final "1" sibling(object) [routine Object arrows object "ext name" parent object To return the number of child objects directly below an local var local var ... local var; class class class ... class statement; has attr def attr def ... attr def object: statement: with prop def, children(object) . . . statement: prop def; •• Message passing ••••• 1: The class, has and with (and also the rarely-used private) To a class: A routine **embedded** as the value of an object property: segments are all optional, and can appear in any order. class.remaining() • has no name, and is called when the property is invoked; To determine an object's class as one of Class, Object, *class*.create() can also be called explicitly using object.property() Routine, String (or nothing): class.destroy(object) • accepts arguments only when called explicitly metaclass(object) class.recreate(object) returns false at the final "1" class.copy(to object, from object) **has segment**: Each *attr def* is either of: To an object: property [attribute local var local var ... local var; ~attribute object.property(a1,a2, ... a7) statement: To change attributes at run-time: To a routine: statement; give object attr def attr def ... attr def; routine.call(a1,a2, ... a7) . . . statement; with/private segments: Each prop_def declares a variable To a string: (or word array) and can take any of these forms (where a string.print() Routines return a single value, when execution reaches the *value* is an expression, a string or an embedded routine): string.print to array(array) final "]" or an explicit return statement: property return *expr*; property value •• Statements ••••••••• property value value ... value return: Each *statement* is terminated by a semi-colon ";". rtrue; A property variable is addressed by *object.property* (or A statement block is a single statement or a series of rfalse: within the object's declaration as self. property). *statements* enclosed in braces { . . . }. To define a dummy standalone routine with *N* local variables Multiple *values* create a property array; in this case An exclamation "!" starts a comment - rest of line ignored. (unless it already exists): object. #property is the number of **bytes** occupied by the A common statement is the assignment: array, the entries can be accessed using Stub routine N: variable = expr; object.&property-->0, object.&property-->1, ..., and *object.property* refers to the value of the first entry. There are two forms of multiple assignment: A property variable inherited from an object's class is variable = variable = ... = expr; addressed by *object.class::property*; this gives the variable = expr, variable = expr, ...; original value prior to any changes within the object. 2

2

•• Flow control •••••••• •• Displaying information ••••••••• •• Uncommon and deprecated statements •• To output a list of values: To execute statements if *expr* is true; optionally, to execute To jump to a labelled statement: other statements if *expr* is false: print value, value, ... value; jump *label*; if (expr) To output a list of values followed by a newline, then return .label: statement: statement block true from the current routine: To terminate the program: if (*expr*) print ret value, value, ... value; statement block quit; If the first (or only) *value* is a string, "print ret" can be else To save and restore the program state: statement block omitted: save label; To execute statements depending on the value of *expr*: "string", value, ... value; Each *value* can be an expression, a string or a rule. switch (expr) { restore *label*: value: statement; ... statement; An **expression** is output as a signed decimal value. To output the Inform compiler version number: value: statement; ... statement; A string in quotes "..." is output as text. inversion: default: statement; ... statement; A **rule** is one of: To accept data from the current input stream: (number) expr the *expr* in words read text array parse array routine; where each *value* can be given as: (char) *expr* the *expr* as a single character To assign to one of 32 'low string' variables: constant the string at the addr (string) addr string N "string"; lo constant to hi constant (address) *addr* the dictionary word at the *addr* constant, constant, ... constant Lowstring *string var* "*string*"; the external (short) name of the *object* (name) *object* string *N* string var: the short name preceded by "a/an" (a) object •• Loop control •••••••••• the short name preceded by "the" (the) *object* To execute statements while *expr* is true: the short name preceded by "The" (The) *object* while (*expr*) (routine) value the output when calling routine(value) statement block To execute statements until *expr* is true: To output a newline character: do new line; statement block To output multiple spaces: until (*expr*) spaces *expr*; To execute statements while a variable changes: To output text in a display box: for (set var : loop while expr : update var) box "string" "string" ... "string"; statement block To change from regular to fixed-pitch font: To execute statements for all defined objects: font off; objectloop (variable) statement block . . . font on: To execute statements for all objects selected by *expr*: To change the font attributes: objectloop (expr starting with variable) style bold; ! use one or more of these statement block style underline; To jump out of the current innermost loop or switch: style reverse: 1 break: style roman; To immediately start the next iteration of the current loop: continue:

3

3

```
•• Verbs and Actions ••••••••••
                                                        •• Other useful directives ••••••••
                                                                                                                •• File structure ••••••••
To specify a new verb:
                                                        To include a directive within a routine definition [...].
                                                                                                                A minimal source file:
                                                        insert a hash "#" as its first character.
 Verb 'verb' 'verb' ... 'verb'
                                                                                                                 Constant Story "MYGAME";
        * token token ... token -> action
                                                                                                                 Constant Headline "^My first Inform game.^";
                                                        To conditionally compile:
       * token token ... token -> action
                                                                                                                 Constant MANUAL PRONOUNS:
                                                         Ifdef name:
                                                                            ! use any one of these
                                                         Ifndef name:
       * token token ... token -> action:
                                                                                                                 Include "Parser":
                                                         Iftrue expr:
                                                                                                                 Include "VerbLib":
where instead "Verb" can be "Verb meta", "action" can be
                                                         Iffalse expr:
"action reverse"; tokens are optional and each is one of:
                                                              . . .
                                                                                                                 [ Initialise; location = study; "^Hello!^"; ];
                                                         Ifnot:
                that literal word
'word'
                                                                                                                 Class
                                                                                                                          Room
w1'/w2'/\ldots any one of those literal words
                                                         Endif:
                                                                                                                          description "A bare room."
                                                                                                                   with
               an object with that attribute
attribute
                                                                                                                   has
                                                                                                                          light:
                                                        To display a compile-time message:
                an object with animate attribute
creature
                                                         Message "string";
                an object held by the player
held
                                                                                                                 Class
                                                                                                                          Furniture
               an object in scope
                                                                                                                   with before [; Take,Pull,Push,Pushdir:
noun
                                                        To include the contents of a file, searching the Library path:
                                                                                                                               print ret (The) self,
noun=routine an object for which routine returns true
                                                         Include "source file";
                                                                                                                                   " is too heavy for that."; ]
scope=routine an object in this re-definition of scope
                                                        To include the contents of a file in the same location as the
                                                                                                                   has
                                                                                                                          static supporter;
multiheld
                one or more objects held by the player
                                                        current file:
                one or more objects in scope
multi
                                                                                                                 Room
                                                                                                                          study "Your study";
                                                         Include ">source file";
               as multi, omitting the specified object
multiexcept
                                                        To specify that a library routine is to be replaced:
                                                                                                                 Furniture "writing desk" study
               as multi, omitting those in specified object
multiinside
                                                                                                                   with name 'writing' 'desk' 'table';
                                                         Replace routine:
                any text
topic
                any number
                                                        To set the game's release number (default is 1), serial
number
                                                                                                                 Object -> -> axe "rusty axe"
                                                        number (default is today's vvmmdd) and status line format
                                                                                                                   with name 'rusty' 'blunt' 'axe' 'hatchet',
routine
               a general parsing routine
                                                                                                                          description "It seems old and blunt.";
                                                        (default is score):
To add synonyms to an existing verb:
                                                         Release expr:
                                                                                                                 Include "Grammar";
 Verb 'verb' 'verb' ... = 'existing verb':
                                                         Serial "vymmdd";
                                                         Statusline score:
To modify an existing verb:
                                                                                                                •• Compiler •••••••••
                                                         Statusline time;
 Extend 'existing verb' last
                                                                                                               To compile (on a PC, use "infrmw32" at the DOS prompt):
                                                        To declare a new attribute common to all objects:
       * token token ... token -> action
                                                                                                                 inform commands source file
       * token token ... token -> action
                                                         Attribute attribute:
                                                                                                                Useful commands include:
                                                        To declare a new property common to all objects:
        * token token ... token -> action:
                                                                                                                -~S
                                                                                                                               disable both Strict checks and Debug tools
                                                         Property property;
where instead "Extend" can be "Extend only" and "last"
                                                         Property property expr;
                                                                                                                -~SD
                                                                                                                               disable Strict checks, enable Debug tools
can be omitted, or changed to "first" or "replace"
                                                                                                                -X
                                                                                                                               enable Infix debugger
To explicitly trigger a defined action (both noun and second
                                                        •• Uncommon and deprecated directives •••
                                                                                                                               output all game text to file (for spell-check)
                                                                                                                -r
are optional, depending on the action):
                                                        You're unlikely to need these; look them up if necessary.
                                                                                                                               display game's size and other statistics
                                                                                                                – s
 <action noun second>:
                                                                                                                               display game's memory map
                                                         Abbreviate "string" "string" ... "string";
                                                                                                                - Z
To explicitly trigger a defined action, then return true from
                                                                                                                               compile in Version 8 format (default is v5)
                                                                                                                -v8
                                                         End:
the current routine:
                                                                                                                               search for Included files in these directories
                                                                                                               +dir,dir,...
                                                         Import variable variable ... variable;
 <<action noun second>>;
                                                         Link "compiled file";
                                                                                                                To display full compiler help, type:
                                                         Switches list of compiler switches;
                                                                                                                 inform -h -h1 -h2
                                                         System file;
```