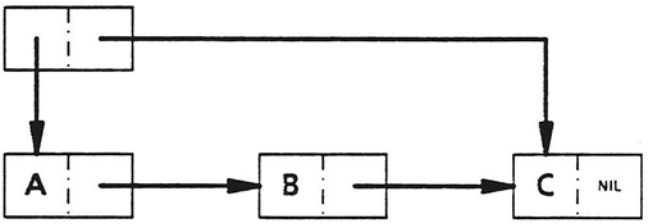
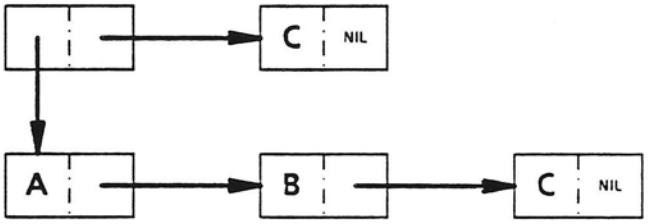
CIRCLPRINT

HPRINT is designed primarily for dumping circular or reentrant list structures (as well as other data structures for which READ is not an inverse of PRINT) so that they can be read back in by Interlisp. The CIRCLPRINT package is designed for printing circular or reentrant structures so that the user can look at them and understand them.

A reentrant list structure is one that contains more than one occurrence of the same EQ structure. For example, TCONC makes use of reentrant list structure so that it does not have to search for the end of the list each time it is called. Thus, if *X* is a list of three elements, (A B C), being constructed by TCONC, the reentrant list structure used by TCONC for this purpose is:

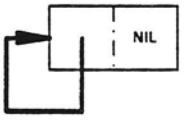


This structure would be printed by PRINT as ((A B C) C). Note that PRINT would produce the same output for the nonreentrant structure:

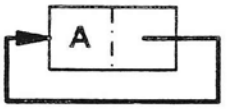


In other words, PRINT does not indicate the fact that portions of the structure in the first figure are identical. Similarly, if PRINT is applied to a circular list structure (a special type of reentrant structure) it will never terminate.

For example, if PRINT is called on the structure:



it will print an endless sequence of left parentheses, and if applied to:



will print a left parenthesis followed by an endless sequence of *A*'s.

The function CIRCLPRINT described below produces output that will exactly describe the structure of any circular or reentrant list structure. This output may be in either single- or double-line format. Below are a few examples of the expressions that CIRCLPRINT would produce to describe the structures discussed above.

First figure, single-line:

((A B *1* C)1)

First figure, double-line:

((A B C) 1)

1

Third figure, single-line:

(*1* 1)

Third figure, double-line:

(1)

1 .

Fourth figure, single-line:

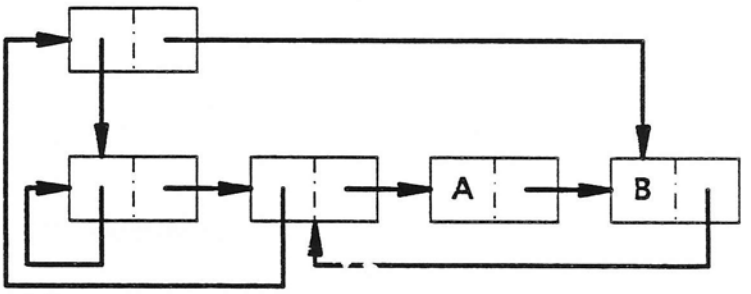
(*1* A . 1)

Fourth figure, double-line:

(A . 1)

1

The more complex structure:



is printed as follows:

Single-line:

(*2* (*1* 1 *3* 2 A *4* B . 3) . 4)

Double-line:

((1 2A B . 3) . 4)

21 3 4

In both formats, the reentrant nodes in the list structure are labeled by numbers. (A reentrant node is one that has two or more pointers coming into it.) In the single-line format, the label is printed between asterisks at the beginning of the node (list or tail) that it identifies. In the double-line format, the label is printed below the beginning of the node it identifies. An occurrence of a reentrant node that has already been identified is indicated by printing its label in brackets.

(CIRCLPRINT *LIST PRINTFLG RLKNT*) [Function]

Prints an expression describing *LIST*. If *PRINTFLG* = NIL, double-line format is used, otherwise single-line format. CIRCLPRINT first calls CIRCLMARK, and then calls either RLPRIN1 (if *PRINTFLG* = T) or RLPRIN2 (if *PRINTFLG* = NIL). Finally, RLRESTORE is called to restore *LIST* to its unmarked state. Returns *LIST*.

(CIRCLMARK *LIST RLKNT*) [Function]

Marks each reentrant node in *LIST* with a unique number, starting at *RLKNT* plus one (or one, if *RLKNT* is NIL). Value is *RLKNT*. Marking *LIST* physically alters it. However, the marking is performed undoably. In addition, *LIST* can always be restored by specifically calling RLRESTORE.

(RLPRIN1 *LIST*) [Function]

Prints an expression describing *LIST* in the single-line format. Does not restore *LIST* to its unCIRCLMARKed state. *LIST* must previously have been CIRCLMARKed, or an error is generated.

(RLPRIN2 *LIST*) [Function]

Same as RLPRIN1, except that the expression describing *LIST* is printed in the double-line format.

(RLRESTORE *LIST*) [Function]

Physically restores list to its original, unmarked state.

Note that the user can mark and print several structures that together share common substructures, e.g., several property lists, by making several calls to CIRCLMARK, followed by calls to RLPRIN1 or RLPRIN2, and finally to RLRESTORE.

(CIRCLMAKER *LIST*) [Function]

LIST may contain labels and references following the convention used by CIRCLPRINT for printing reentrant structures in single-line format, e.g., (*1* . 1). CIRCLMAKER performs the necessary RPLACAs and RPLACDs to make *LIST* correspond to the indicated structure. Value is (altered) *LIST*.

(CIRCLMAKER1 *LIST*) [Function]

Does the work for CIRCLMAKER. Uses free variables LABELST and REFLST. LABELST is a list of dotted pairs of labels and

corresponding nodes. REFLST is a list of nodes containing references to labels not yet seen. CIRCLMAKER operates by initializing LABELST and REFLST to NIL, and then calling CIRCLMAKER1. It generates an error if REFLST is not NIL when CIRCLMAKER1 returns. The user can call CIRCLMAKER1 directly to "connect up" several structures that share common substructures, e.g., several property lists.

READAIS

AIS (array of intensity samples) is a format for color and gray-level images. The following functions allow reading and writing of AIS files from Lisp. Note: The reading of an AIS file is done with the normal interrupts off.

(AISBLT *SourceAisFile SourceX SourceY DestinationBitmap DestinationX DestinationY Width Height How Filter Nbits LoBitAddress*) [Function]

Puts the image in an AIS file into a bit map. AISBLT checks the sample size of the AIS file and the number of bits per pixel of the *DestinationBitmap* and performs the required reduction (if any). *SourceX* and *SourceY* give the left and bottom coordinates in the source file of the image to be read (default to (0,0)). *DestinationBitmap* can be a bit map, a color bit map, or a window. If it is NIL, a window will be created of the proper size.

How indicates what method of reduction is to be used if the sample size of the AIS file is larger than the number of bits per pixel in the destination bit map. The recognized methods are TRUNCATE (use the high-order bits) and FSA (use the Floyd-Steinberg dithering algorithm). The default when going to a 1 bpp bit map is FSA; otherwise it is TRUNCATE.

Filter if non-NIL should be an array that will be used to filter the samples read from the AIS file. If *Filter* is given and a sample point of intensity *N* is read from the file, (ELT *Filter N*) is used to determine the bits for the destination. The function SMOOTHHIST described below is one way of getting a filter that balances the contrast in an image.

Nbits and *LoBitAddress* allow an image to be read into one or more "planes" of a color bit map. *Nbits* tells how many bits are to be taken from each image sample, and *LoBitAddress* indicates the lowest bit within each pixel that the *Nbits* bits are to go. (Bit address zero is the leftmost or highest-order bit. For a four-bit-per-pixel bit map, three would be the lowest-order bit.) This is used by SHOWCOLORAIS to put the different planes of a color image into the bit map.

(SHOWCOLORAIS *BaseFile ColorMapInfo How ColorBM SourceX SourceY DestinationX DestinationY Width Height*) [Function]

Reads a color image from three AIS files into a color bit map. The three color files are obtained by concatenating the strings "-red.ais", "-green.ais", and "-blue.ais"onto the end of *BaseFile*. If *ColorMapInfo* is a list of three small integers, it indicates how many of the bits in the destination are allocated to each color.

For example, if *ColorBM* is a four-bit-per-pixel color bit map and *ColorMapInfo* is (1 2 1), one bit (bit zero) will be allocated to the red image, two bits (bits one and two) will be allocated to the green image, and one bit (bit three) will be allocated to the blue image.

ColorBM is the color bit map the image will be stored into. It defaults to (COLORSCREENBITMAP).

How, SourceX, SourceY, DestinationX, DestinationY, Width, and Height are as described in AISBLT.

An experimental feature that is available only when going to 8 bpp color bit map: if *ColorMapInfo* is a color map, each pixel will be determined by finding the color in *ColorMapInfo* that is closest to the 24 bits of color information read from the three image files. (This takes a long time.) The function COLOR.DISTANCE (red green blue redentry greenentry blueentry) is called to calculate the distance by which "closest" color is determined.

Note: Large portions of SHOWCOLORAIS run with the standard interrupts turned off.

(THREECOLORMAP *RedBits GreenBits BlueBits Nbits*) [Function]

Returns a color map that assumes the *Nbits* bits are to be treated as three separate color planes with *RedBits* bits being in the red plane, *GreenBits* bits being in the green plane, and *BlueBits* bits being in the blue plane. Within each plane, the colors are uniformly distributed over the intensity range 0 to 255.

(WRITEAIS *ColorBM File Region*) [Function]

Writes the region *Region* of the color bit map *ColorBM* onto the file *File* in AIS format. This provides an efficient way of saving color or gray- level images.

(AISHISTOGRAM *File Region*) [Function]

Returns a histogram array of the region *Region* in the AIS file *File*. The histogram array has as its *N*th element the number of pixels in the region that have intensity *N*.

(GRAPHAISHISTOGRAM *Histogram W*) [Function]

Draws a graph of a histogram array in the window *W*. If *W* is NIL, a window is created.

(SMOOTHEDFILTER *Histogram*) [Function]

Returns a "filter" array that maximally distributes the intensities values contained in *Histogram*. The filter array can be passed to AISBLT to change the contrast of the image being read.