```
MM       MM       OOOOOO       DDDDDDDD       EEEEEEEEEE     LL
MM       MM       OOOOOO       DDDDDDDD       EEEEEEEEEE     LL
MMMM   MMMM     OO      OO     DD      DD     EE             LL
MMMM   MMMM     OO      OO     DD      DD     EE             LL
MM  MM  MM     OO      OO     DD      DD     EE             LL
MM  MM  MM     OO      OO     DD      DD     EE             LL
MM       MM     OO      OO     DD      DD     EEEEEEEE       LL
MM       MM     OO      OO     DD      DD     EEEEEEEE       LL
MM       MM     OO      OO     DD      DD     EE             LL
MM       MM     OO      OO     DD      DD     EE             LL
MM       MM     OO      OO     DD      DD     EE             LL
MM       MM     OO      OO     DD      DD     EE             LL
MM       MM       OOOOOO       DDDDDDDD       EEEEEEEEEE     LLLLLLLLLL
MM       MM       OOOOOO       DDDDDDDD       EEEEEEEEEE     LLLLLLLLLL



LL               PPPPPPPP       TTTTTTTTTT                       11
LL               PPPPPPPP       TTTTTTTTTT                       11
LL               PP      PP         TT                         1111
LL               PP      PP         TT                         1111
LL               PP      PP         TT                           11
LL               PP      PP         TT                           11
LL               PPPPPPPP           TT                           11
LL               PPPPPPPP           TT                           11
LL               PP                 TT                           11
LL               PP                 TT                           11
LL               PP                 TT             ....          11
LL               PP                 TT             ....          11
LLLLLLLLLL       PP                 TT             ....      111111
LLLLLLLLLL       PP                 TT             ....      111111



*START* Job ARIADN Req #326 for EGB    Date 15-Feb-84 15:33:09 Monitor: //, TOPS
File RM:<JIM.DOC>MODEL.LPT.1, created:  9-Aug-83 16:21:48
        printed: 15-Feb-84 15:33:40
Job parameters: Request created:15-Feb-84 15:33:05   Page limit:81   Forms:NORMAL  A
File parameters: Copy: 1 of 1   Spacing:SINGLE   File format:ASCII   Print mode:ASCI
```

Editor Model

1. Introduction

In the interest of simplifying the user's notion of how the editor works, the following is an attempt to define the user-apparent types and their operations in a consistent manner. While this may make some operations less convenient than a hand-tailored approach, it seems likely to make the system more generally understandable.

2. Types

Any editing session involves a variety of data structures. The following is an attempt to describe each of the types and how it relates to others in the system.

Object — A basic component of the underlying system. Objects are typed. Each type has type-specific editor component called an Object Editor. The object editor interfaces between items of the type and the rest of the editor.

Also used for any entity that constitutes a complete, identifiable sub-tree of an environment object. Examples of Ada objects are packages, procedures, statements, expressions, identifiers. For English, examples are paragraphs, sentences and words. Objects that do not contain other objects are called "leaf objects". Objects that are known outside of the editor are called "environment objects" where the distinction is important.

Selection — A distinguished object or set of objects.

Image — The human-readable representation of an object. The user is encouraged to think of the image of an object as being the visible representative of the object. At whatever level of detail the image represents the object, changes to the object must be reflected on the image (this is most easily accomplished by allowing only one image per underlying object, but this is an implementation issue, not a part of the definition). The visible representation of the image consists of an array (positive) of lines, each an array (positive) of character positions.

Position — A particular (line, column) coordinate on an image, window, or screen.

Window — A bounded rectangular portion of a screen. Windows are normally associated with images..

Superwindow — A collection of windows that are manipulated together, typically consisting of a major window and its associated

banner and command window(s). A superwindow corresponds loosely to the user notion of window. Where possible without introducing confusion, window will be used in place of superwindow.

Screen — A representation of what the user sees on the terminal screen. Made up of a ring of stacked, overlapping, (opaque) windows.

Cursor — A distinguished position on a window, image or screen.

Point — The user-visible (flashing) screen cursor.

Character — The contents of a particular image position. Any particular character has an ASCII value, a usage (self-representing, ellipsis, prompt), and a font. Also used to indicate single-character granularity of position operations.

Word — A sequence of one or more characters in the image, delimited by one or more characters that do not normally occur in the content (as opposed to punctuation) of the object. For Ada and English, examples are space, comma, period and parentheses.

Ring — A collection of entities of a particular type (position, object, window) that is arranged to provide stack-like push and and non-destructive pop. That is, pop really rotates the ring and push inserts the item into the ring at the current (top) position. Rings are of finite size (push can cause ring members to be displaced), the size of each being determined by its expected use.

3. Character and Object Editing

The existence of an object structure makes it possible to provide a number of operations that depend on the type-specific structure of the particular object. This mechanism not only allows simple manipulation of large sections of text, it allows precautions to be taken to preserve the semantic integrity of the objects. On the other hand, much of the content of these objects is simply characters that the user may want or need to change independent of the object structure. As a result, editing takes place at two distinct, but highly related levels, character and object.

Both object and character operations are useful; both are supported for their strengths. The following sections discuss character operations, object operations and their relationship.

3.1. Character Operations

The character-oriented view of the entities in the system is:

Character	Basic component, a single column's contents.
Word	One or more contiguous "content" characters.
Line	Zero or more characters arrayed horizontally.
Pane	One or more lines; the contents of a window.
Image	The entirety of a view of an object; one or more panes.
Screen	All of the characters visible to the user.

Note that this arrangement is loosely hierarchical, i.e. image is made up of panes, which is made up of lines, which is (sort of) made up of words, which are made up of characters. Words do not fit as neatly as the others, but are an attractive user notion. Images are really composed of lines, but a pane is convenient since it captures the set of lines that is visible. It is called a pane to focus on the text that appears in the window, rather than the window itself. We will use the notion of hierarchical containment in constructing a uniform set of operations that span the various natural granularities of the display.

Consicer the following primitive character operations.

- Position (Up, Down, Left, Right)
- Delete (Up, Down, Left, Right, All)

3.1.1. Positioning

All of the following positioning commands move the image cursor to a position that is computed on the basis of the unit specified. Appropriate actions are taken to keep the image cursor visible on the window.

Character Position up (down, left, right) character causes motion in the quarter plane, moving one character position up (down, left, right). Position up (left) stops at the first column (line), but proceeds indefinitely down (right).

Word Position left (right) word moves a word or part of a word to the left (right), stopping at the left (right) end of the word. Position up (down) word moves to the word above (below). The last word of line N is adjacent to first word of line N+1.

Line Position left (right) line moves to the left (right) end of the line. Position up (down) line moves to the first (last) line of the pane.

Pane Position left (right) pane moves to the left (right) end of the pane. Position up (down) pane moves to the previous (next) pane of the image.

Image Position left (right) image moves to column 1 (the last column of) the current line. Position up (down) image moves to the first (last) position of the image.

Screen Position screen operations work just like position character operations, except that motion is on the screen rather than on an image.

3.1.2. Deletion

Deletion operations follow the notions of left and right from positioning very closely. The principal difference is that while quarter-plane motion makes it possible to get to any logical position in the image, strict quarter-plane deletion makes it impossible to change the number of lines in the image. Delete all operations are intended to delete all of the characters falling within the specified region.

Character Delete up (down, left, right) character corresponds to moving the point as with position and deleting the character(s) between its final and initial positions. The sole difference being that there is an implied position between lines that is deleted by delete left from column 1 and delete right from columns at or to the right of the end of the line.

Delete all character is the same as delete right character.

Word Delete left (right) word deletes from the point to the position a word or part of a word to the left (right). Delete up (down) word deletes from the point to the word above (below) it. The last word of line N is adjacent to first word of line N+1.

Line Delete left (right) line deletes from the point to the beginning (end) of the line. Delete up (down) line deletes from the point to the end of the previous (start of the next) line. Note that this differs from the pane-orientation of motion.

Delete all line removes the entire line and its contents.

Pane Delete left (right) pane deletes from the point to the left (right) edge of the pane. Delete up (down) pane deletes from the point to the beginning (end) of the pane. Delete all pane deletes all of the lines on the pane.

Image Delete left (right) image deletes from the point to column 1 (last column of the current line. Delete up (down) image deletes from the point to the start (end) of the image. Delete all image deletes all of the

Screen There are no distinct screen deletion operations.

3.2. Object Operations

Object operations differ from character operations in two important ways:

1. The user can (must) specify the scope of the object of interest.
2. Since the user has specified an object, the system can treat it as an object, not just a set of contiguous characters.

At any particular time, an image cursor is positioned at a particular character, word, line, pane and image; it is also positioned within some number of objects. While it is possible to compute which objects the cursor is within by climbing the object tree to its root, there is no fixed-depth notion of object containment analogous to character, line, image. Thus, where it is possible to determine the meaning of "delete line" on the basis of a single image position, it is not possible to determine the meaning of "delete object" without determining which of the containing objects is meant.

Need description of object tree somewhere in here.

For any particular image, there is a corresponding root object. For any particular point in the image, there is a containment tree reaching back to the root. Unfortunately, adjacent image positions can have very different containment trees, with one contained directly in the root object and the other a part of a deeply nested object. For example, punctuation and white-space typically falls outside of local contexts. We justify ignoring this complication by noting that every position has a smallest enclosing object and the positions of most active user interest are "well-behaved".

We will use the term Object Cursor to indicate the object equivalent of the image cursor, that is, the extent and location of the current object.

Object cursor motions follow the object tree. Left (right) moves the cursor to the previous (next) object at the same logical level. Up moves the cursor to the parent of the current object; this can also be viewed as increasing the size of the object cursor. Down moves the object cursor to one of the children of the current object cursor; either the child used to select the parent or the first child.

There are delete current and previous object operations that do the "obvious" things.

Object cursors are also used for selection and moving/copying objects as described below.

3.3. Object and Character Cursors

As described above, there are two user-visible cursors for each window. For the moment, we will assume that there is only one window containing the current object and point cursors. We also ignore the distinction between the image and screen cursors. The following are characteristics of the two types of cursors and their relationship.

1. Although the point cursor seems to rest on top of a single character, it actually represents a point between two characters.

2. The point cursor is represented on the screen by the physical cursor.

3. The object cursor either represents the null object positioned at the point cursor or the contents of an object.

4. The object cursor is represented by a (somehow) highlighted region of the window that covers all of the visible characters of the represented object.

5. The point cursor is always visible on the current window.

6. When an object cursor is created or altered, the point cursor is within the object cursor. The point cursor's position within the object cursor only changes as necessary, i.e. if the object cursor is moved so that the point cursor is no longer contained, the point cursor moves to the beginning of the object cursor created.

7. The object cursor is unchanged by movement of the point cursor.

8. There exists an operation to convert the point cursor into an object cursor containing only a prompt for the type of item that can be inserted at the current cursor position.

3.4. Selection and Region Operations

A basic notion of the editor is the ability to designate a region of interest and perform operations on it. Deletion has been discussed above in the context of both object deletion and text area deletion. Two additional operations, Move and Copy, are also commonly useful for both character and object editing. A salient characteristic of both move and copy is the need for two kinds of information: what to move (copy) and to where. For both character and object editing, there are deletion operations that are self-contained, e.g. delete the current

object or delete to the end of the current line. The only movement operations of comparable "simplicity" would be of the form transpose this x with the previous x.

The basic move paradigm is:

1. Select a region or object.

    a. The object cursor designates a region.

        More complicated (possibly discontiguous) regions can be built up

    b. For character operations, the designation of a region proceeds by marking one end of the selection, moving the cursor and forming a selection from the region between the marked position and the current cursor.

2. Designate a position to which to move the selection.

3. Request that the selected region or object be moved.

Some types of selections can only be moved as objects; others only as text. For object operations, the point cursor may be an ambiguous or inappropriate insertion point. For such cases, the opening of an insertion point and positioning of the point cursor inside of it serves to remove the ambiguity or at least make the user aware of the logical characteristics of the target position. If the user doesn't open an insertion point, one will be opened prior to attempting the move.

Note that despite the parallelism of this description, move character region and move object can have very different effects and limitations. The intent is to promote the use of object operations where possible and provide fine-grain character operations as less convenient alternatives for when no object-oriented operation allows the required flexibility.

3.5. Compound Regions

The object cursor and character selections described above are merely convenient special cases of more general selections.

1. General selections are composed of a ring of simple selections.

2. General selections also form a ring.

3. Each simple selection is added to the ring of general selections.

4. The user can construct general selections by coalescing the two top selections of the selection ring into the new top element.

5. The top of the selection ring (of each type) is highlighted.

6. For object cursor operations, the current object cursor is the member of the selection containing the point cursor or the null object associated with the point cursor. This means that moving the point cursor away from the object cursor leaves the object cursor (and other parts of the current object cursor) selected until the next object cursor operation.

7. There is a null selection that can be placed at the top of the ring to remove highlighting from any element.

8. Operations that deal with a ring do so by iterating over the members of the ring and performing the operation on each component. The order of the iteration is the order in which the items were added to the ring.

Conversion operations exist for making a character (object) selection into the closest corresponding object (character) selection. In the case of object to character conversion, the region of text can be exact. For character to object, the smallest enclosing list of compatible objects is chosen. There is a good chance that users would prefer to have a single, less-specific set of operations that perform the conversions automatically and/or complain of unreasonable operations.

3.6. Rings

The basic operations on rings are push, pop, next and previous. For specific purposes, ring-specific operations can be defined that act upon the top two (N?) elements. An example is the combine the top two elements operation on selection rings.

4. Windows, Superwindows and Screens

1. The content of an object is associated with its images.

2. In order to see an image, it must be represented on a window.

3. Windows are grouped into superwindows to make visual correspondence easier for the user.

4. Superwindows are a part of a screen. Information can be shared across screens by having distinct windows on different screens referencing the same image.

5. Command windows are associated with a particular window as long as it contains a particular image.

6. Each image has attributes that govern how it is edited. These include overwrite, and changed, parsed, committed status.

7. Windows form a ring within a screen; screens form a ring within a session.

4.1. Screens

1. Screens have functional characteristics (program development, mail, debugging) and specific names to distinguish among multiple screens for the same function.

2. Screens are for user convenience in managing screen clutter between activities that are viewed as distinct. The available functions are not limited by the screen, but by the permissible operations on whatever objects are being edited.

3. A new screen starts out with a set of default windows (possibly depending on its function, eventually depending on user-controllable parameters). A typical empty screen might be similar to a start-up screen on Red, consisting of a system status window and an object window with associated command windows, banners, etc.

4.2. Windows

A number of different kinds of windows are maintained:

1. Object windows. Windows in which the user edits objects.
2. Command windows. Ada windows intended for immediate execution. The major distinction between these and object windows is intent. Depending on user preference, command windows may have command characteristics that are identical to Ada windows or very different ones.
3. Banners. Signpost windows attached to object or other major windows. Intially, probably not editable.
4. System windows. Windows used by to communicate editor and system information. Principal difference from object windows is persistence and usage. System windows may disappear or scroll themselves as needed. Examples are status, help and buffer-directory windows.

Table of Contents