```
-- StreamIO.Mesa  Edited by Sandman on May 12, 1978  3:09 PM

DIRECTORY
  IODefs: FROM "iodefs" USING [
    Control A, Control H, Control Q, Control R, Control V, Control W, Control X, CR,
    DEL, ESC, NumberFormat, SP],
  StreamDefs: FROM "streamdefs" USING [
    ClearCurrentLine, ClearDisplayChar, GetDefaultDisplayStream,
    GetDefaultKey, StreamHandle],
  StringDefs: FROM "stringdefs" USING [
    AppendChar, AppendNumber, StringToNumber];

DEFINITIONS FROM StreamDefs, IODefs;

StreamIO: PROGRAM
  IMPORTS StreamDefs, StringDefs EXPORTS IODefs SHARES IODefs, StreamDefs = PUBLIC

BEGIN

 -- the main body

Input, Output: StreamHandle;

BeginLine: PRIVATE BOOLEAN ← TRUE;

GetInputStream: PROCEDURE RETURNS [StreamHandle] =
  BEGIN
  RETURN[Input]
  END;

GetOutputStream: PROCEDURE RETURNS [StreamHandle] =
  BEGIN
  RETURN[Output]
  END;

SetInputStream: PROCEDURE [s: StreamHandle] =
  BEGIN
  Input ← s
  END;

SetOutputStream: PROCEDURE [s: StreamHandle] =
  BEGIN
  Output ← s;
  BeginLine ← TRUE;
  END;


-- Character operations

ReadChar: PROCEDURE RETURNS [CHARACTER] =
  BEGIN
  RETURN[Input.get[Input]];
  END;


WriteChar: PROCEDURE [c:CHARACTER] =
  BEGIN
  Output.put[Output, c];
  BeginLine ← c = CR;
  END;


-- Reading Strings

ReadString: PROCEDURE [s:STRING, t:PROCEDURE[CHARACTER]RETURNS[BOOLEAN]] =
  BEGIN
  WriteChar[ReadEditedString[s,t,TRUE]];
  END;

ReadID: PROCEDURE [s:STRING] =
  BEGIN
  [] ← ReadEditedString[s,atomfound,TRUE];
  END;

ReadLine: PROCEDURE [s:STRING] =
  BEGIN
```

```
  [] ← ReadEditedString[s,crfound,TRUE];
  WriteChar[CR];
  END;

crfound: PRIVATE PROCEDURE [c:CHARACTER] RETURNS [BOOLEAN] =
  BEGIN RETURN [c = CR] END;

atomfound: PRIVATE PROCEDURE [c:CHARACTER] RETURNS [BOOLEAN] =
  BEGIN RETURN [IF c = SP OR c = CR THEN TRUE ELSE FALSE] END;

Rubout: SIGNAL = CODE;
LineOverflow: SIGNAL [s: STRING] RETURNS [ns: STRING] = CODE;

ReadEditedString: PROCEDURE [s:STRING, t:PROCEDURE [CHARACTER] RETURNS [BOOLEAN], newstring:BOOLEAN] RE
**TURNS[CHARACTER] =
  BEGIN
  c: CHARACTER;
  i: CARDINAL;
  state: {TrailingInvisible, Visible, LeadingInvisible};
  c ← Input.get[Input];
  IF newstring THEN
    IF c = ESC THEN
      BEGIN  WriteString[s];  c ← Input.get[Input];  END
    ELSE s.length ← 0;
  UNTIL t[c] DO SELECT c FROM
    DEL => SIGNAL Rubout;
    ControlA, ControlH =>
      BEGIN
      IF s.length > 0 THEN
        BEGIN
        WITH Output SELECT FROM
          Display => ClearDisplayChar[Output,s[s.length-1]];
          ENDCASE => Output.put[Output,c];
        s.length ← s.length-1;
        END;
      END;
    ControlW, ControlQ =>
      BEGIN  -- text to be backed up is of the form
      -- ...<LeadingInvisible><Visible><TrailingInvisible>
      -- the <Visible> and <TrailingInvisible> are to be removed.
      state ← TrailingInvisible;
      FOR i DECREASING IN [0..s.length) DO
        SELECT s[i] FROM
          IN ['A..'Z], IN ['a..'z], IN ['0..'9] =>
            IF state = TrailingInvisible THEN state ← Visible;
          ENDCASE =>
            IF state = Visible THEN state ← LeadingInvisible;
        IF state = LeadingInvisible THEN GO TO Done;
        WITH Output SELECT FROM
          Display => ClearDisplayChar[Output,s[i]];
          ENDCASE => Output.put[Output,c];
        REPEAT
          Done => s.length ← i+1;
          FINISHED => s.length ← 0;
        ENDLOOP;
      END;
    ControlR =>
      BEGIN
      WriteChar[CR];
      WriteString[s];
      END;
    ControlX =>
      BEGIN
      WITH Output SELECT FROM
        Display => ClearCurrentLine[Output];
        ENDCASE => Output.put[Output,c];
      s.length ← 0;
      END;
    ControlV =>
      BEGIN
      WHILE s.length >= s.maxlength DO
        s ← SIGNAL LineOverflow[s];
        ENDLOOP;
      s[s.length] ← c ← Input.get[Input];
      s.length ← s.length+1;
      WriteChar[c];
```

```
      END;
    ENDCASE =>
      BEGIN
      WHILE s.length >= s.maxlength DO
        s ← SIGNAL LineOverflow[s];
        ENDLOOP;
      s[s.length] ← c;
      s.length ← s.length+1;
      WriteChar[c];
      END;
    c ← Input.get[Input];
    ENDLOOP;
  RETURN[c];
  END;


-- Writing Strings

WriteString: PROCEDURE [s:STRING] =
  BEGIN
  i:CARDINAL;
  FOR i IN [0..s.length) DO
    Output.put[Output,s[i]];
    ENDLOOP;
  IF s.length # 0 THEN BeginLine ← s[s.length-1] = CR;
  END;

WriteLine: PROCEDURE [s:STRING] =
  BEGIN
  WriteString[s];
  WriteChar[CR];
  END;

NewLine: PROCEDURE RETURNS[BOOLEAN] =
  BEGIN RETURN[BeginLine] END;
```

```
-- Numerical i/o

ReadNumber: PROCEDURE [default: UNSPECIFIED, radix: CARDINAL]
  RETURNS [UNSPECIFIED] =
  BEGIN
  s: STRING ← [10];
  IF radix = 10 AND LOOPHOLE[default, INTEGER] < 0 THEN
    BEGIN default ← -default; s[0] ← '-; s.length ← 1 END;
  StringDefs.AppendNumber[s,default,radix];
  IF radix = 8 THEN StringDefs.AppendChar[s,'B];
  [] ← ReadEditedString[s, atomfound, TRUE];
  RETURN[StringDefs.StringToNumber[s,radix]];
  END;

ReadDecimal: PROCEDURE RETURNS [INTEGER] =
  BEGIN
  s: STRING ← [10];
  [] ← ReadEditedString[s, atomfound, TRUE];
  RETURN [StringDefs.StringToNumber[s,10]]
  END;

ReadOctal: PROCEDURE RETURNS [UNSPECIFIED]=
  BEGIN
  s: STRING ← [10];
  [] ← ReadEditedString[s, atomfound, TRUE];
  RETURN [StringDefs.StringToNumber[s,8]]
  END;

OutNumber: PROCEDURE
  [stream: StreamHandle, val: INTEGER, format: NumberFormat] =
  BEGIN
  i: CARDINAL;
  neg: BOOLEAN ← FALSE;
  fill: CHARACTER ← (IF format.zerofill THEN '0 ELSE ' );
  s: STRING ← [10];

  IF val<0 AND ~format.unsigned THEN BEGIN val←-val; neg←TRUE END;
  StringDefs.AppendNumber[s,val,format.base];
  i ← s.length;
  IF neg THEN
    BEGIN
    i ← i + 1;
    IF format.zerofill THEN BEGIN stream.put[stream,'-]; neg←FALSE END;
    END;
  THROUGH (i..format.columns] DO stream.put[stream,fill] ENDLOOP;
  IF neg THEN stream.put[stream,'-];
  FOR i IN [0..s.length) DO stream.put[stream,s[i]] ENDLOOP;
  RETURN
  END;

WriteNumber: PROCEDURE [v: UNSPECIFIED, f: NumberFormat] =
  BEGIN
  OutNumber[Output,v,f];
  BeginLine ← FALSE;
  RETURN
  END;

WriteDecimal: PROCEDURE [n: INTEGER] =
  BEGIN
  WriteNumber[n,NumberFormat[10,FALSE,FALSE,0]];
  RETURN
  END;

WriteOctal: PROCEDURE [n: UNSPECIFIED] =
  BEGIN
  WriteNumber[n,NumberFormat[8,FALSE,TRUE,0]];
  IF n ~IN[0..7] THEN WriteChar['B];
  RETURN
  END;

Input ← StreamDefs.GetDefaultKey[!ANY => CONTINUE];
Output ← StreamDefs.GetDefaultDisplayStream[!ANY => CONTINUE];

END.
```