N. Wirth
26.6.73

Implementations of PASCAL in systems using character sets without control characters*

In implementations based on character sets lacking the character eol (end of line), a text file x is considered as a sequence of lines, and each line being a sequence of characters (like a file of files).

The procedures put(x) and get(x) operate within a component of the file x, i.e. within a line. Upon generation of a text, a line is terminated by calling the (new) procedure

putln(x)

In reading, calling get(x) when x↑ represents the last character of a line, causes the (new) predicate

eol(x) (end-of-line)

to become true. The value of x↑ resulting from this operation is not defined. Reading of the next line is initiated by calling the (new) procedure

getln(x)

Note the analogy of the following program schemas valid for implementations using the control character eol, and the new line control procedures respectively:

writing a text x :

```
repeat                                  repeat
    repeat P(x↑); put(x)                    repeat P(x↑); put(x)
    until p;                                until p;
       x↑ := eol; put(x)                    putln(x)
until q                                 until q
```

reading a text x :

```
while ¬eof(x) do                        while ¬eof(x) do
begin                                   begin
     while x↑ ≠ eol do                       while ¬eol(x) do
     begin Q(x↑); get(x)                     begin Q(x↑); get(x)
     end;                                    end;
     R; get(x)                               R; getln(x)
end                                     end
```

* for instance, CDC SCOPE 3.4 with 64-character sets

For the standard text files input and output, two new standard procedures are introduced:

```
readln   ≡   getln(input)
writeln  ≡   putln(output)
```

The following two program schemas demonstrate their use. Again, the analogous schemas for implementations offering the control character eol are shown in correspondence.

Writing the text "output":

```
repeat                                  repeat
    repeat P(ch); write(ch)                 repeat P(ch); write(ch)
    until p;                                until p;
    write(eol)                              writeln
until q                                 until q
```

Reading the text "input":

```
while ¬eof(input) do                    while ¬eof(input) do
begin read(ch);                         begin
      while ch ≠ eol do                       while ¬eol(input) do
      begin Q(ch); read(ch)                   begin read(ch); Q(ch)
      end;                                    end;
      R                                       R; readln
end                                     end
```

The greatest imcompatibility between two corresponding schemas occurs in reading the file input. This unfortunate discrepancy is due to the circumstances, that read(ch) stands for the two statements

```
ch := input↑; get(input)
```

that by definition in implementations without control characters the condition eol(input) = true prohibits the application of get(input) , and that therefore the function eol(input) must be evaluated before the first call of read(ch) . Note that in these implementations, eol does never occur as the value of input↑ ; instead, if eol(x) = true , the value x↑ is some printable character, but it is not defined which one. Neither must the value eol occur as an argument to the standard procedure write.