

In-class Exercise

- Find as many problems as you can with the following specification for filling text
 - It was originally developed by Peter Naur as part of his description of the stepwise refinement approach to program development

Naur's Original Problem Statement

Given a text consisting of words separated by BLANKS or by NL (newline) characters, convert it to a line-by-line form in accordance with the following rules:

1. line breaks must be made only where the given text has BLANK or NL
2. each line is filled as far as possible, as long as
3. no line will contain more than MAXPOS characters

Problems - 1

1. No definition of *line*; no statement of the relationship of *line* to the newline character
2. If the input text does not end with an NL, then (presumably) it is not a line; hence, it is meaningless to talk of it being converted on a *line-by-line* basis
3. More generally, there is no description of the structure of the input file; no definition in terms of lines. There is also no definition of the output file structure
4. There is no prescription about what to do about multiple blanks and at least two interpretations are possible. There is no rule about zero length words and in fact no definition of *word*

Problems - 2

5. There is no prescription about what to do about multiple newlines
6. Nothing is stated about restrictions on the input file; specifically, what to do if an input line has a word containing $>$ MAXPOS characters
7. There is no prescription about what to do if the first characters in the input are one or more BLANKs
8. There is no rule considering the case where multiple BLANKs end the file or come just before a NL that ends the file

Another Try

- In the context of program testing, Susan Gerhardt and John Goodenough devised the following version

Gerhardt And Goodenough - 1

The program's input is a stream of characters whose end is signaled with a special end-of-text character, ET. There is exactly one ET character in each input stream.

Characters are classified as

- break characters--BL (blank) and NL (newline);
- nonbreak characters--all others except ET;
- the end-of-text indicator--ET

A *word* is a nonempty sequence of nonbreak characters. A *break* is a sequence of one or more break characters. Thus, the input can be viewed as a sequence of words separated by breaks, with possibly leading and trailing breaks, and ending with ET

Gerhardt And Goodenough - 2

The program's output should be the same sequence of words as in the input with the exception that an oversized word (i.e., a word containing more than MAXPOS characters, where MAXPOS is a positive integer) should cause an error exit from the program (i.e., a variable, Alarm, should have the value TRUE), up to the point of an error, the program's output should have the following properties:

1. A new line should start only between words and at the beginning of the output text, if any.
2. A break in the input is reduced to a single break character in the output.
3. As many words as possible should be placed on each line (i.e., between successive NL characters).
4. No line may contain more than MAXPOS characters (words and BLs).

Problems With The G&G Spec - 1

1. Use of both of the phrases *non-empty* and *one or more*. Why not avoid potential confusion by using just one phrase? Same criticism holds for the terms *stream* and *sequence*
2. *Output text, if any* is an example of remorse. A qualification to the definition of output text is added long after it is first used
3. *Line* is used before it is defined. The definition is incorrect (it defines a line as being between NL's which ignores text before the first NL). The meaning of NL and its relation to the concept of *line* is left implicit. *Alarm* is a program concept instead of a problem concept. Behavior of Alarm is not specified if MAXPOS is never exceeded. The *point of error* is not defined

Problems With The G&G Spec - 2

4. Contradictory definitions of input (*stream of characters* and *sequence of words separated by breaks*)
5. ET is a machine dependent (program domain) concept. It is used three times before it is defined. The output file does not contain ET which is either a bug in the spec or a significant non-uniformity
6. The phrase *trailing blanks ending with ET* is confusing
7. MAXPOS is used before it is defined

Bertrand Meyer's Approach

- Bertrand Meyer has addressed the difficulties in specifying the text-fill problem mathematically. His approach is to define the problem in terms of sequences, sets of sequences, functions and relations. Left undefined is the concepts of CHAR, other than it is a set that contains at least the two elements *blank* and *new_line*. The set BREAK_CHAR is defined to be the set $\{\overline{blank}, \overline{new_line}\}$
- Both the input and the output are sequences of CHARs. The main idea is to look at sets of subsequences that are related to the input by various rules and to select only those subsequences that obey other rules. Any element of this set (of subsequences) is a valid output of the program

Meyer's Approach - 2

- Among the properties of Meyer's solution are that, if more than one output satisfies the spec, the spec does not prefer one to another. This is called a *non-deterministic spec*. The advantage of non-determinism is that it gives more freedom to developers.
- The spec also does not constrain the solution by discussing program details. Thus, if the input contains a word longer than MAXPOS, no sequence of CHARs will satisfy the spec in relationship to it. How the error is indicated is up to the implementation (or another part of the spec)
- After constructing a mathematical solution, Meyer then expresses the solution in unambiguous structured English, as presented on the next slide.

Meyer's Final Spec

Given are a non-negative integer MAXPOS and a character set including two "break characters", *blank* and *new_line*

The program shall accept as input a finite sequence of characters and produce as output a sequence of characters satisfying the following conditions:

- It only differs from the input by having a single break character whenever the input has one or more break characters;
- any MAXPOS + 1 consecutive characters include a *new_line*;
- the number of *new_line* characters is minimal

If (and only if) an input sequence contains a group of MAXPOS + 1 consecutive nonbreak characters, there exists no such output. In this case, the program shall produce the output associated with the initial part of the sequence, up to and including the MAXPOS-th character of the first such group, and report the error