MAV '87 ?

RECURSION - SHOULD IT BE AVOIDED OR DOES IT LIBERATE?

Liddy Nevile
Australian Council for Educational Research

The question in the title often arises when a student who has some understanding of recursion is told, yet again, to write a program in BASIC which will do this (or that) and that most of the marks will be given for the algorithm and the desk-check. Teachers have even found it wise to avoid giving marks for the program because students might copy form someone else.

This is not a small problem. Teachers in Victorian schools are busy integrating computer programming into their mathematics curricula, but not easily. There are too many languages, too many dialects even, for the teachers to be able to set tasks which they can be sure will be appropriate. Students have home computers; they are often very good programmers; and they may well have different ways of understanding programming fror their teachers. Teachers are used to being accountable for their work, and are placed in an unenviable position by the new 'rules' of the examination system which make them responsible for part of the students' final marks - such teachers cannot be expected to take risks.

One aspect of programming which has caused particular concern is recursion. In programming terms, it is elegant, economical, and a device which those who have mastered it would hesitate to disregard when they feel it is called for in solving a problem. Indeed, they may, given free choice, choose problems just because they are suitable for solution by the writing of recursive procedures.

From a mathematics point of view, this can cause some difficulties for the teachers. Why?

First we need to know what is being considered when the word recursion is used; how does recursion relate to mathematics, to computation, and what 'value' might understanding recursion have for students.

Recursion

A dictionary 'might' say recursion is recursion - that is a recursive definition.

In practice, the definition of recursion depends upon its context; whether it is representing the product or the process - the static or the dynamic.

In practice, the definition of recursion depends upon its context; whether it is representing the product or the process - the static or the dynamic.

Archimedes is said to be one of the first to use recursion when he defined pi as the limit of a recurrent sequence, but Boyle was one of the first to use the term (Kilpatrick, 1984, p. 9).

Kilpatrick wrote (1984):

> Recursion has no generally accepted meaning; it is applied to several related concepts: recursion relations, recursive functions, recursive procedures, and recursive conditional expressions (McCarthy, 1983). One can think of recursion as a method of defining a function 'by specifying each of its values in terms of previously defined values, and possibly using other defined functions' (Cutland, 1980, p. 32). Or one can think somewhat more generally of a recursive function or procedure as one that calls itself (Cooper and Clancy, 1982, p. 236).
>
> This self-referential aspect of recursion is its most intriguing feature. The Dictionary of Scientific and Technical terms (1978) defines recursion as 'a technique in which an apparently circular process is used to perform an iterative process'. More broadly still, Hofstadter (1979) characterizes recursion as 'nesting, and variations on nesting' (p. 127). For Hofstadter, recursion is a metaphor for organizing the world). (p. 9)

It is the 'apparently circular' feature of the recursive definition which often makes it expressive, and which is considered here as one of its significant liberating features. In the example below, procedures which will produce the nth element of the Fibonacci series are considered. The main one is recursive, it is a definition which depends upon its own meaning, it is defined in terms of itself (or self-image). This can make writing definitions easy, but is not normally acceptable in either mathematics or English.

Abelson and Sussman have written (1985, p. 97):

> Recursion is a normal tool for dealing with tree structures, since we can often reduce operations on trees to operations on their branches, which reduce in turn to operations on the branches of the branches, and so on, until we reach the leaves of the branches.

This is what happens in the case of the Fibonacci series. A complex tree structure underlies the apparently simple definition (see below).

Having chosen a suitable description of the series, the procedures can be written without complete specification (in mathematical terms) and from then on the relevant procedures (processes) can be treated as if they are objects, the element itself.

Having processes which immediately become objects is powerful and relevant to the question of liberation of the user. It is possible for the user to build another level of abstraction into a problem solving exercise. If a problem can be reduced in this way, the solving process can often become one of successive definition, until there is no longer a problem.

When the solving of the problem becomes the process of successively redefining the problem, at the end there is no explicit algorithm which can be handed in with the program.

When the computer performs the computation, another problem arises for the teacher and student. Because recursion is a very messy process, 'playing computer' is not a very satisfactory exercise, often producing pages and pages of workings which are neither instructive nor useful.

Each time the recursive procedure calls the (recursive) procedure it creates another copy of itself, stores the remaining part of itself, and proceeds to compute the new procedure, which in turn probably calls for a new copy of itself ... until the STOP rule is reached, when all the stored unfinished work can be slowly recovered and completed. This process can be so demanding on computer memory that often recursion which is almost iteration is required (see below).

So a student who chooses to solve a problem using recursion in this way is likely to fail according to marks given for the original problem set (by the teacher above): there will be no algorithm, no sensible desk check, despite the elegant and efficient solution of the problem.

Recursion and Mathematics

Abelson and Sussman note (1985):

> despite the enormous amount of work in programming-language semantics, the notion of a data object, which is so important and pervasive in modern programming practice, does not have a completely satisfactory mathematical treatment. (p. 82)

But there is a beginning. Kilpatrick noted (1984):

> Hilton (1984) has observed that the computer is changing mathematics in several ways, including giving a new prominence to iteration theory. Maurer (1983) argues that teachers should present 'the modern precise idea of an algorithm, and some of its particular techniques such as recursion, as among the great ideas in human intellectual history'. (p. 161). (p. 9)

Abelson and Sussman commented (1985):

> In testing the primality of very large numbers chosen at random, the chance of stumbling upon a value that fools the Fermat test is less than the chance that cosmic radiation will cause the computer to make an error in carrying out a 'correct' algorithm. Considering an algorithm to be inadequate for the first reason but not for the second illustrates the difference between mathematics and engineering. (p. 49)

This presents a challenge to mathematicians, and therefore mathematics educators.

There are other attributes to recursion which are likely to be useful to students and which are not normally employed by school students. Among these is the inductive process; real mathematicians use it frequently. The notion of limit is another: the problem is which comes first, for instance limits or recursion? Another problem of course relates to how these concepts can be learned.

Significant among the utility aspects of recursion must be the production of stunning graphics. Fractals are but one example:

Tessellations take on a new interest when they become Escher-like.

In dealing with these issues, the student practices a number of skills which are mathematical, and further, experiences some of the aesthetics of mathematics which are usually only enjoyed by real mathematicians.

Kilpatrick raises the issue of students' reflection on their work. If they are to learn about learning, with recursion they will have a tool with which to do this, for it is a good tool for thinking about learning and knowledge development - a recursive operation.

Conclusion

If it is true that students in schools could enjoy activities which are closely linked to recursion, and it is true that there is a lot of mathematical experience implicit in computation using recursion, then shouldn't recursion be something which we endeavour to teach to students, rather than something to be avoided?

An Example

The Fibonacci series provides a suitable example of recursion and its role for our purposes.

Any element in the series is the sum of the last two preceding elements, and of course each of those is the sum of the two proceeding it:

term n is (term n-1 + term n-2)

Alternatively,

term n, where the first two terms are a and b

is the same as

term n-1, where the first two terms are b and a+b

is the same as

term n-2, where the first two terms are a+b and a+2b

is the same as ...

If the second definition is chosen,

```
TO FIB.SERIES :A :B :N
IF :N = 1 [OUTPUT :B STOP]
FIB.SERIES :B :A+:B :N-1
END
```

is its Logo representation (definition), and the element can be obtained with the aid of a small procedure which 'produces' it (makes it into an object):

```
TO FIB :N
OUTPUT FIB.SERIES 0 1 :N
```