Tasting the flavour of programming

(On the occasion of a talk to be given to Finnish students visiting the THE on wednesday 27 march 1985.)

It is a pleasure to me to have the opportunity to address young students from abroad, especially when those students are from Finland. Partially due to the circumstance that our countries are both very small and partially due to the circumstance that Finland is located in such a far corner of Western Europe, Eindhoven University has a much more intimate contact with the Helsinki Universities than with -for instance- the famous Sorbonne. I regard your visit to Eindhoven as a continuation of that relationship and I bid you welcome.

*

Over the last decade programming has presented itself as a new mathematical discipline. Since I do not expect you to be familiar with that discipline and since this is a very short talk, I shall restrict myself to sketching you the flavour of the discipline (at least of some part of it). For that purpose I shall show you two examples of program development, one very simple example (to warm up) and one less simple example (to make a point).

+

The simple exercise deals with the derivation of a program that -when executed- generates a table of the first 100 cubes of natural numbers. My guess is that nobody will object to the program

$$n := 0$$
$$;\ \underline{\text{do}}\ n \neq 100 \rightarrow f{:}(n) = n^3;\ n := n+1\ \underline{\text{od}}\ .$$

This program is correct because it maintains (the truth of) the condition

$$P0: \quad 0 \leq n \leq 100 \ \wedge\ (\underline{A}i : 0 \leq i < n : f(i) = i^3)\ ,$$

so that upon termination, when $n = 100$, $P0$ precisely states what we wanted.

If we would leave it at this, the example would be a little bit too poor. For the sake of the argument we impose the restriction that the program be expressed in terms of no other arithmetic operations than additive operations. For the program we already have this means that we have to eliminate the now inadmissible expression n^3. We propose to do so by insisting -in addition to $P0$- on (the truth of)

$$P1: \quad x = n^3 .$$

This gives us as a next version of the program, for instance,

$$n := 0;\ x := 0$$
$$;\ \underline{\text{do}}\ n \neq 100 \rightarrow f{:}(n) = x\ ;\ x := (n+1)^3;\ n := n+1\ \underline{\text{od}}$$

Now it seems that we have made things worse: the inadmissible expression n^3 has been exchanged for the more complicated, equally inadmissible, expression $(n+1)^3$. However, we achieved that the assignment to f became admissible. And we proceed, with the promise that with respect to this assignment the program has its eventual shape.

The strategy for dealing with the inadmissible $(n+1)^3$ is standard: if the machine cannot perform the calculation, we have to do it ourselves. Hence, we start calculating,

$$\begin{array}{ll} & (n+1)^3 \\ = & \quad \{\text{algebra}\} \\ & n^3 + 3.n^2 + 3.n + 1 \\ = & \quad \{\text{on account of P1}\} \\ & x + 3.n^2 + 3.n + 1 \quad , \end{array}$$

and, hence, we may replace the inadmissible $x := (n+1)^3$ by the still inadmissible $x := x + 3.n^2 + 3.n + 1$. But we made progress: we exchanged the cubic expression for a quadratic one. By insisting -in addition to P0 and P1- on (the truth of)

P2: $y = 3.n^2 + 3.n + 1$,

we find as a next version of the program

```
   n:= 0; x:= 0; y:= 1
 ; do n ≠ 100
   → f:(n)= x ; x:= x+y
     ; y:= 3.(n+1)² + 3.(n+1) + 1
     ; n:= n+1
   od .
```

In the meantime, you will have recognized the pattern: what we just did to transform the assignment to x into an admissible one will be repeated for y. Because

$$
\begin{array}{ll}
 & 3.(n+1)^2 + 3.(n+1) + 1 \\
= & \quad \{\text{algebra}\} \\
 & 3.n^2 + 3.n + 1 + 6.n + 6 \\
= & \quad \{\text{on account of P2}\} \\
 & y + 6.n + 6 ,
\end{array}
$$

we are suggested to insist on (the truth of)

P3: $z = 6.n + 6$

as well.

I leave the last step to you and give you the ultimate program:

```
   n := 0; x := 0; y := 1; z := 6
 ; do  n ≠ 100
   →   f:(n) = x
     ; x := x + y
     ; y := y + z
     ; z := z + 6
     ; n := n + 1
   od .
```

The technique used in the derivation of the above program is absolutely standard. People familiar with that technique produce this program, without hesitation and without thinking, in less than half a minute. I suggest that, when you are back home again, you pose the problem to one of your colleagues, who is probably not familiar with the technique. Then you have a big chance that you find him hesitating or thinking or produce a more complicated program (perhaps incorrect), or combinations of these. This, of course, is not amazing because the availability of a mathematical technique enhances — if the technique is well-chosen — one's mathematical abilities (whatever this may be). If you are not struck by the difference you could expose your colleague to the following problem, which is a crime to many professional programmers.

+

For a sequence $f(i: 0 \le i < N)$, $N \ge 0$, of integers the "segmentsum" $S(p,q)$ is defined for all p, q in the range $0 \le p \le q \le N$ by

$$S(p,q) = (\underline{S}\, i: p \le i < q: f(i)) .$$

We wish to construct a program that computes the minimum value of a segmentsum, i.e. that computes the value x that satisfies

$$x = (\underline{\mathrm{MIN}}\, p,q: 0 \le p \le q \le N: S(p,q)) .$$

Again, my guess is that nobody will object to the program

```
  n := 0; x := 0
; do n ≠ N
  →  x := (MIN p,q: 0 ≤ p ≤ q ≤ n+1 : S(p,q))
   ; n := n+1
  od .
```

This program is correct because it maintains

$P0: \quad 0 \le n \le N \;\wedge\; x = (\underline{\text{MIN}}\, p,q : 0 \le p \le q \le n : S(p,q))$,

so that upon termination, when $n = N$, $P0$ precisely states what we wanted.

<u>Remark</u> "Nobody will object" is perhaps a too bold statement since you don't know the definition of <u>MIN</u>. I trust however, you made an intelligent guess.
(End of Remark.)

The assignment to x is so horribly inadmissible that we rashly start to calculate for ourselves:

$$
\begin{array}{ll}
 & (\underline{\text{MIN}}\, p,q : 0 \le p \le q \le n+1 : S(p,q)) \\
= & \qquad \{\text{splitting the domain}\} \\
 & (\underline{\text{MIN}}\, p,q : 0 \le p \le q \le n : S(p,q)) \\
 & \underline{\text{min}} \\
 & (\underline{\text{MIN}}\, p : 0 \le p \le n+1 : S(p,n+1)) \\
= & \qquad \{\text{on account of } P0\} \\
 & x \;\underline{\text{min}}\; (\underline{\text{MIN}}\, p : 0 \le p \le n+1 : S(p,n+1))
\end{array}
$$

Therefore, a next version of the program is

```
   n := 0; x := 0
 ; do  n ≠ N
   →   x := x min (MIN p: 0 ≤ p ≤ n+1 : S(p, n+1))
     ; n := n+1
   od .
```

Again we made progress, because the horribly inadmissible expression has been exchanged for a still horribly inadmissible, yet simpler, expression: we lost a dummy.

As in the first example we could try to insist on (the truth of)

$$y = (\underline{\mathrm{MIN}}\ p : 0 \le p \le n+1 : S(p, n+1)) ,$$

but that is impossible because $S(p, n+1)$ is not defined for all values in the range of n, viz. not for $n = N$. Instead we insist on

$$P1: \quad y = (\underline{\mathrm{MIN}}\ p : 0 \le p \le n : S(p, n)) .$$

The next version then becomes

```
   n := 0 ; x := 0 ; y := 0
 ; do n ≠ N
   → y := (MIN p: 0 ≤ p ≤ n+1 : S(p, n+1))
     ; x := x min y
     ; n := n+1
   od .
```

<u>Note</u> Notice that, thus far, the sequence f has not yet entered the game.
(End of Note.)

Again, we calculate:

$$
\begin{array}{ll}
 & (\underline{\mathrm{MIN}}\ p : 0 \le p \le n+1 : S(p, n+1)) \\
= & \quad \{ \text{splitting the domain} \} \\
 & (\underline{\mathrm{MIN}}\ p : 0 \le p \le n : S(p, n+1))\ \underline{\min}\ S(n+1, n+1) \\
= & \quad \{ \text{definition of } S \} \\
 & (\underline{\mathrm{MIN}}\ p : 0 \le p \le n : S(p, n) + f(n))\ \underline{\min}\ 0 \\
= & \quad \{ \text{addition distributes over } \underline{\mathrm{MIN}} \} \\
 & ((\underline{\mathrm{MIN}}\ p : 0 \le p \le n : S(p, n)) + f(n))\ \underline{\min}\ 0 \\
= & \quad \{ \text{according to } P1 \} \\
 & (y + f(n))\ \underline{\min}\ 0 .
\end{array}
$$

And here is our final program:

$$
\begin{array}{l}
n := 0;\ x := 0;\ y := 0 \\
;\ \underline{\text{do}}\ n \neq N \\
\quad \rightarrow\ y := (y + f(n))\ \underline{\text{min}}\ 0 \\
\qquad ;\ x := x\ \underline{\text{min}}\ y \\
\qquad ;\ n := n+1 \\
\underline{\text{od}}\ .
\end{array}
$$

Ain't it a Beauty.

*

I hope that I made you taste the flavour of programming, and I hope it tasted well. But before we leave I would like to make one more remark. You saw me using non-conventional notations for more or less conventional notions. What you saw was the top of an iceberg. Machines derive their usefulness from the fact that they do <u>precisely</u> what we instruct them to do. If we want to use them well, we have to learn how to conduct our reasoning power much more precisely, and hence much more effectively, than traditional mathematics is able to supply. As such, I am convinced that Computing Science will have a much deeper influence on mathematics than the current generation of mathematicians is willing to accept. I hope that <u>you</u> change gears.

A nice stay, and a good trip back home.

W.H.J. Feijen,
26 March 1985