

FORTRAN DAYS

John Stuckey

I feel younger than I am, and I am younger than I look. Or at least that is my conceit.

I am by no means one of the pioneers of computing *per se*, having started using computers only 39 years ago, yet when I got into it, computing services was a not-yet-recognized field. People were still impressed to know that one “knew about computers.”

I came back from overseas in 1967, after ending my short and very junior career with the foreign service in disagreement with U.S. Southeast Asia policy, returning to the University of Kansas, where I could hope that I was remembered as someone brighter than my grades suggested. It worked, and I managed to get admitted to graduate study in political science there and, two years later, at the University of Michigan.

My wife at the time got an operator’s job in the Computing Center at Kansas to help support us. Those computer things seemed pretty cool, so I registered for a no-credit course in FORTRAN programming.

It changed my life.

I mean, it wasn’t easy. You punched your instructions onto those notorious (“Do not fold, spindle, or mutilate”) 80-column cards and submitted them, along with any also-punched data you might hope to analyze, at the Input Counter. There they were stacked along with the other “jobs” waiting to be submitted to the awesome digital wizard somewhere back behind. One could later call a recorded message line to find out when to come, hopefully, to claim the results. “Estimated turn-around is 24-36 hours.” “Estimated turn-around is 6-8 hours.” That gave one an idea of when to come retrieve one’s set of punched cards and ensuing printout.

And to see, almost inevitably, what brainless typo had wrecked the entire run. And then to launch the process again, hoping to progress at least a few lines further into the virgin code. Mozart is said to have written perfect first-draft scores that needed no correction. That’s not the sort of FORTRAN I wrote.

The bane of my programming life, other than simple typos, was the infinite loop – a set of instructions that told the computer to do something over and over until some end condition was reached but all too often managed to define the end condition so that it was never reached. That sounds absurdly stupid, and I cannot argue that it was not, yet I suspect that I was not the only one who managed to inflict that fatality, repeatedly, on his programs. One had to estimate, for processor-allotment purposes, the maximum amount of highly-rationed and valuable time one was requesting for one's "job." Much as I hated estimating that limit, I was not infrequently glad it was there, for if I was dumb enough to program an infinite loop, the computer was more than brainless enough to do exactly what I had instructed, over and mindlessly over until it had exhausted the limit. In its simplest prototype, an infinite loop consisted of something like

```
1 J=3
2 K=4
3 I=J+K
4 IF (I .GT. 5) GO TO 3
5 etc.
```

where your brain was saying "13" at the end of the fourth statement while your fingers typed "3". Even at the slow speeds of 1960s processors, the computer could do statements 3 and 4 over and over, a whole lot of times per second, to no useful purpose whatsoever.

For many years now, using computers has nearly universally meant "running" programs written by others. Perhaps even being one of thousands or millions who run those programs. In the "old days," one nearly always ran, or tried to run, programs one had written oneself, often for a single, specific inquiry. It was a humbling experience.

Back then, I could make FORTRAN do wonderful tricks, purposes for which other languages would eventually be devised. For my wife's dissertation I wrote programs to enable her to edit the text and generate the glossary of the pre-modern-English translations of 12th-century Latin manuscripts, including alphabetic characters that did not survive into modern English and were therefore not present among printable characters. FORTRAN was not intended for such text-manipulation purposes, but it was the language I knew. We argued with the Graduate School at Michigan over the submitted

format of the dissertation, ultimately persuading those in charge of such details that multiple identical xerographic manuscripts was *at least* as good as the “original” and “carbon copies” which had been rigidly required for many decades.

Those programming skills were valuable for at least 3-4 years, until they were rendered obsolete. Today, I have a hard time explaining to computer users even what it means to “write a program,” let alone why I still hesitate to begin a line in an e-mail message or any other context with a number.

I’m willing to bet that many of you have much better stories of your early experiences with computers. Being freed from the punched card or paper tape by your first teletype, allowing you to type instructions that were processed in real time. Tricking the computer to do what you required. Silly things either you or the computer did. Getting your first acoustic coupler, giving you mobility of access. Seeing/using colors and graphics for the first time. Care to share some of those experiences? Send me some of your memories, and although I can’t promise to publish them all, I’ll bet we can share some amusing stories. And tell me, if you know, why it should still makes me nervous to start a line of text, in the middle of a paragraph, with the numeral “1.”

Send any of your anecdotes or reminiscences to me ubiquity@acm.org before the end of January, 2007. If I may identify you in passing your stories on, please give me explicit permission to do so. Let’s make *Ubiquity* a little more interactive.

John Stuckey wrestled with computers and tried to help users as an IT administrator at Michigan, Carnegie-Mellon, Northeastern, Washington and Lee, and the Wissenschaftszentrum Berlin. Since retiring from full-time employment earlier this year he is writing, consulting, and traveling. He’s also a frequent reviewer and associate editor for Ubiquity.

Source: <http://acm.org/ubiquity>