Who Can Be Programmers?

Computing practice and theory is based on the very hidden and hard-to-reveal assumption that engineers, mathematicians, and computer scientists are the only ones who will write a program or contribute to a software system. When the computing age began, no one but scientists and the military even remotely conceived of the utility of programs and programming. Today, almost every business and human pursuit is built on computing and digital technology. Artists, craftspeople, writers, fishermen, farmers, tightrope walkers, bankers, children, carpenters, singers, dentists, and animals depend on computing and most of the people I mentioned want to have a say in how such software works, looks, and behaves. Many of them would program if it were possible.

Richard Gabriel, "Mob Software"

Spirits of the Computer

We are about to study the idea of a *computational process.* Computational processes are abstract beings that inhabit computers. As they evolve, processes manipluate other abstract things called *data*. The evolution of a process is directed by a pattern of rules called a *program*. People create programs to direct processes. In effect, we conjure the spirits of the computer with our spells.

A computational process is indeed much like a sorcerer's idea of a spirit. It cannot be seen or touched. It is not composed of matter at all. However, it is very real. It can perform intellectual work. . . .

Abelson and Sussman, *Structure and Interpretation of Computer Programs*

Spirits of the Computer (cont'd)

The programs we use to conjure processes are like a sorcerer's spells. They are carefully composed from symbolic expressions in arcane and esoteric *programming languages* that prescribe the tasks we want our processes to perform.

A computational process, in a correctly working computer, executes programs precisely and accurately. Thus, like the sorcerer's apprentice, novice programmers must learn to understand and to anticipate the consequences of their conjuring. Even small errors (usually called *bugs* or *glitches*) in programs can have complex and unanticipated consequences.

Abelson and Sussman, *Structure and Interpretation of Computer Programs*

The Compulsive Programmer

The computer programmer. . . is a creator of universes for which he alone is the lawgiver. So, of course, is the designer of any game, . . . [but computer] systems so formulated and elaborated *act out* their programmed scripts. They compliantly objey their laws and vividly exhibit their obedient behavior. No playwright, no stage directory, no empoeror, however powerful, has ever excercised such absolute authority to arrange a stage or a field of battle and to command such unswervingly dutiful actors or troops.

One would have to be astonished if Lord Acton's observation that power corrupts were not to apply in an environment in which omnipotence is so easily achievable. It does apply.

Joseph Weizenbaum, *Computer Power and Human Reason*

The Compulsive Programmer (cont'd)

Wherever computer centers have become established. . . bright young men of disheveled appearance, often with sunken glowing eyes, can be seen sitting at computer consoles, their arms tensed and waiting to fire their fingers. . . their attention seems to be as rivited as a gambler's on the rolling dice. . . .

Their rumpled clothes, their unwashed and unshaven faces, and their uncombed hair all testify that they are ob livious to their bodies and to the world in which they move. They exist, at least when so engaged, only through and for the computers. These are computer bums, compulsive programmers.

Joseph Weizenbaum, *Computer Power and Human Reason*

Reflection of God's Creativity

I think people who write programs do have a glimmer of extra insight into the nature of God. . . because creating a program often means that you have to create a small universe.

Don Knuth, *Things a Computer Scientist Rarely Talks about*

The Joys of the Craft

Why is programming fun? What delights may its practitioner expect as his reward?

First is the sheer joy of making things. As the child delights in his mud pie, so the adult enjoys building things, especially things of his [or her] own design. I think this delight must be an image of God's delight in making things, a delight shown in the distinctness and newness of each leaf and each snowflake.

Fred Brooks, *The Mythical Man-Month*

The Joys of the Craft (cont'd)

Second is the pleasure of making things that are useful to other people...

Third is the fascination of fashioning complex puzzle-like objects of interlocking moving parts and watching them work in subtle cycles, playing out the consequences of principles built in from the beginning...

Fourth is the joy of always learning, which springs from the nonrepeating nature of the task...

Fred Brooks, *The Mythical Man-Month*

The Joys of the Craft (cont'd)

Yet the Finally, there is the delight of working in such a tractable medium. The programmer, like the poet, works only slightly removed from pure thought-stuff. He [or she] builds castles in the air, from air, creating by exertion of the imagination. Few media of creation are so flexible, so easy to polish and rework, so readily capable of realizing grand conceptual structures. (As we shall see later, this very tractability has its own problems.)

Fred Brooks, *The Mythical Man-Month*

The Joys of the Craft (cont'd)

Yet the program construct, unlike the poet's words, is real in the sense that it moves and works, producing visible outputs separate from the construct itself. It prints results, draws pictures, produces sounds, moves arms. The magic of myth and legend has come true in our time. One types the correct incantation on a keyboard, and a display screen comes to life, showing things that never were or could be.

Programming is fun because it gratifies creative longings built deep within us. . .

Fred Brooks, *The Mythical Man-Month*