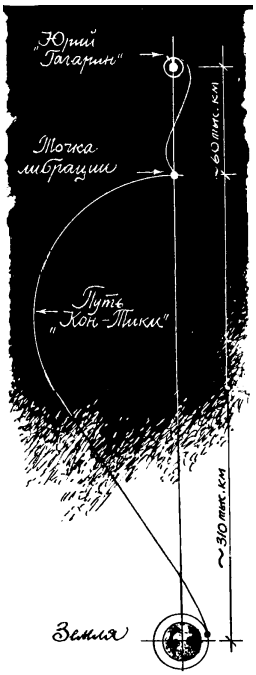
2 Greybeard's Luck

a sermon by the Rt. Revd. Dr. Pastor Manul Laphroaig

My first computer was not a computer; rather, it was a "programmable micro-calculator." By the look of it, it was macro rather than micro, and could double as a half-brick in times of need. It had to be plugged in pretty much most of the time (these days, I have a phone like that), and any and all programs had to be punched in every time it lost power for some reason. It sure sounds like five miles uphill in the snow, both ways, but in fact it was the most wondrous thing ever.



The programmable part was a stack machine with a few additional named memory registers. Instructions were punched on the keyboard; besides the stack reverse Polish arithmetic, branches, and a couple of conditionals, there was a command for pushing a keyed-in number on top of the stack. That was my first read-eval-print loop, and it was amazing. Days were spent entering some numbers, hitting go, observing the output, and repeating over and over. (A trip from the Moon base back to Earth took almost a year, piece by piece. A sci-fi monthly published a program for each trajectory, from lift-off to refueling at a Lagrange point, and finally atmospheric braking and the perilous final landing on good old Earth.)

You see, I understood everything about that calculator: the stack, the stop-and-wait for the input, reading and writing registers (that is, pushing the numbers in them on top of the stack or copying the top of the stack into them), the branches and the loops. There was never a question how any operation worked: I always knew what registers were involved, and had to know this in order to program anything at all. No detail of the programming model could be left as "magic" to "understand later"; no vaguely understood part could be left glossed over to "do real work now." There were no magical incantations to cut-and-paste to make something work without understanding it.

I did not recognize how lucky I had been until, many years later, I decided to take up "real" industrial programming, which back then meant C++. Suddenly my head was full of Inheritance, Overloading, Encapsulation, Polymorphism, and suchlike things, all with capital letters. I learned their definitions, pasted large blocks of code, and enthusiastically puzzled over tricky questions from these Grand Principles of Object Oriented Programming such as, "if a virtual function is also overloaded, which version will be called?" In retrospect, my time would have been better spent researching whether Superman would win over Batman.

At about the same time I learned about New Math. It was born of the original Sputnik Moment and was the grand idea to reform the teaching of mathematics to school children so that they would make better Sputniks, and faster. The earth-bound kind of arithmetic that was useful in a shop class would be replaced by the deeper, space-age kind.

That Sputnik must have carried a psychotronic weapon. There is no other sane explanation for why the schooling of American engineers—those who launched the same kind of satellite just four months later—suddenly wasn't deemed good enough. A whole industry arose to print new, more expensive textbooks, with Ph.D.s in space-age math education to match; teachers were told to abandon the old ways and teach to the new standards. Perfectly numerate parents could no longer comprehend the point of grade school arithmetic homework.

Suddenly, adding numbers mattered less than knowing that Addition was Commutative; as a result, school children learned about Commutativity but could no longer actually add numbers. They couldn't add numbers in their heads or on paper, let alone multiply them. Shop class became the only place in school where one could actually learn about fractions—not that they were Rational Numbers, but how to actually measure things with them, and why. College students thought an algebraic equation was harder if it contained fractions.

Knowledge of math was measured by remembering special words, rather than a show of skill. You see, a skill always involves a lot of tricks; they may be nifty, but they are also too technical and who has time for

that in this space age? Important Concepts, on the other hand, are nicely general, and you can have middle schoolers saying things straight out of the graduate program within a few weeks! Is that not Progress? Indeed, only one other Wonder of Progress can stand close to New Math: the way that children are locked in a room with a literate adult for most of the day, for years, and still emerge unable to read. People couldn't pull that off in the Dark Ages; this takes Science to organize.

What came after New Math was even worse. Some of the school children who could barely count but knew the Important Concepts became teachers and teachers of teachers. Others realized that despite all the Big Ideas the skill of math was vanishing. They saw the fruits of Big Idea pushers dismissing drill; they concluded that drill was the key to the skill. So subsequent reforms barreled between repetitive, senseless rote and more Capital Letter Words. These days it seems that Discovery, Higher Order, Critical Thinking are in fashion, which means children must waste days of school time "discovering" Pi and suchlike, working through countless vaguely defined steps, only to memorize whatever the teacher would tell them these activities meant in the end. Now we have the worst of all: wasted time and boredom without any productive skill actually learned. The only thing than can be learned in such a class is helplessness and putting up with pretentious waste of time, or worse!, mistaking this for actual math.

I was beginning to feel pretty helpless in the world of C++ Important Concepts of Object Oriented Programming. I was yearning for my old calculator, where I did not have to learn a magical order of mystery buttons to press in order to get the simplest program to work. Having had a book fetish since childhood, I hoped for a while that I just hadn't found the right one to Unleash or Dummify myself in 21 Days. I was like a school child who could hardly suspect that the latest textbook with brightly colored pictures is full of vague unmathematical crap that would horrify actual mathematicians. (More likely, such mathematicians of ages past would run the textbook authors through in a proper duel.)

Then one day that world was blown to bits. Polymorphism and Inheritance blew up when I saw a vtable. After that, function name mangling was a brief mop-up operation that took care of Overloading. Suddenly, the Superman-vs-Batman contests and other C++ language-lawyer interview fare became trivial. It was just as simple as my calculator; in fact, it was simpler because it did not have the complexity of managing a tiny amount of memory.

There is an old name for what people do with Big Ideas and Important Concepts that are so important that you cannot hope to have their internal workings understood without special training by special people. It is called *worshiping idols*, and what we ought to do with idols is to smash them to bits.

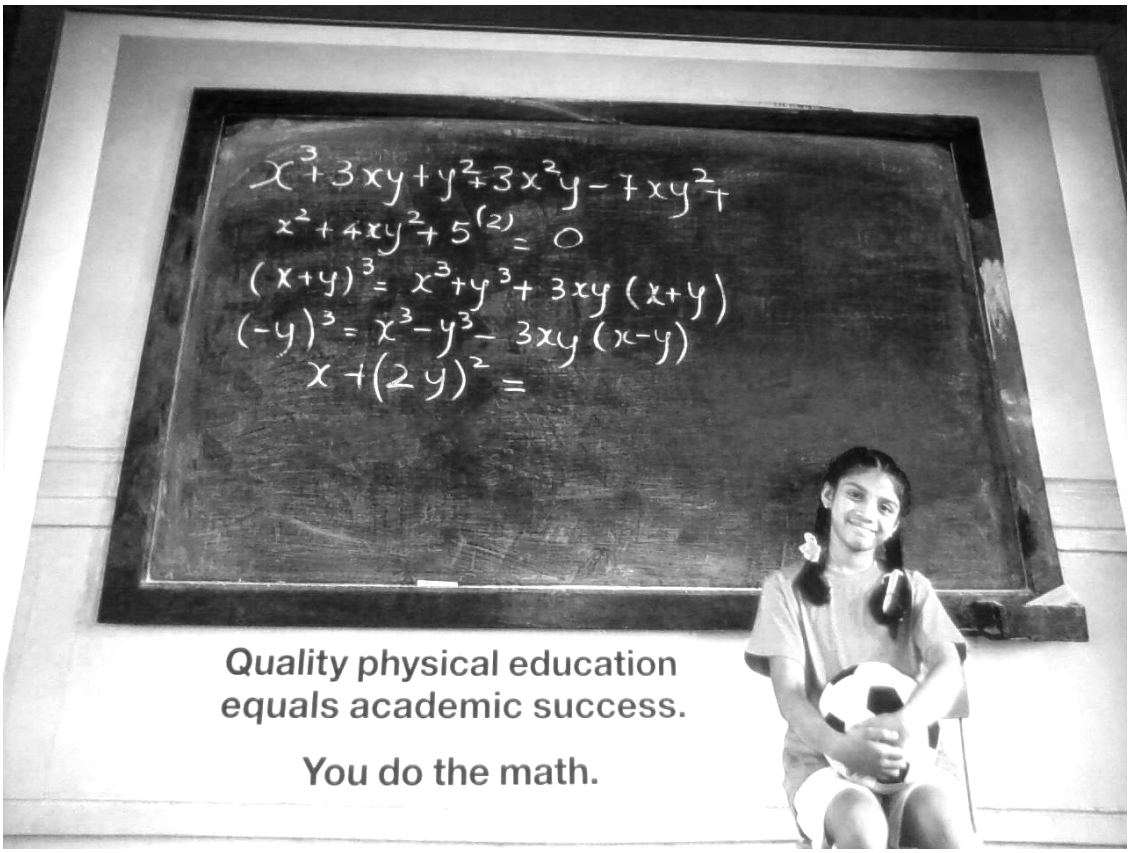
And if the bits do not make sense, then the whole of a Most Modern Capitalized Fashion does not make sense, and the special people are merely priests promising that supplicating the idol will improve your affairs. Not that anything is wrong with priests, but idols teach no skills, and if your trust is in your skill, then you should seek a different temple and a different augur. Or, better yet, build your own damned bird-feeder!

– — — – — — — — — — – — — — – — – — — – — – — — – — — —

Verily I say to you that when they keep uttering some words in such a way that you hear Capital Letters, look 'em in the eye and ask 'em: "how does this work?" Also remember that "I don't really know" is an acceptable answer, and the one who gives it is your potential ally.

I was brought to a place where they worshiped idols called Commutativity and Associativity, or else Inheritance and Polymorphism, and where they made sacrifices of their children's time to these idols. They made many useless manuscripts that would break a mule's back but which these children had to carry to and from school. And making a whip of cords, I drove them all out of the temple, screaming "This is a waste of time and paper! Trees will grow back hundredfold if you let them alone, for nature cannot be screwed, but who will restore to the old the lost time of their youth?"

They taught, "Lo this is Commutative and Higher Order, or else this is a Reference, and this is a Pointer." And when I asked them, "How do you add numbers, and how does your linker work?", they demurred and spoke of Abstraction and Patterns. Verily I tell you, if you don't know how to do your Abstractions on paper and what they compile into, you are worshiping idols and wasting your time. And if you teach that to children, you are sacrificing their time and their minds to your graven images. Repent and smash your graven idols to bits, and teach your children about the smashing and the bits and the bytes instead, for these are the only skills that matter!



Seriously, try to do the math.