Programmable Calculators

Adrian Kosmaczewski

2021-01-01

As far as I can remember, I have always been fascinated by computers. But in the crisis-ridden Argentina of the 1980s, buying one was beyond the reach of my mother. Actually, we did not even have a telephone back then.

The economy did not allow for much freedom of movement; the national currency was extremely devaluated, and was losing value every minute. Three different monetary units in a couple of years (the “Peso Ley 18’888”, the “Peso Argentino”, and the “Austral”) could not stop inflation, unbound and uncontrollable, from eating savings, sanity and happiness.

Thankfully, those were the best years of Argentine football. But that is another story.

My explorations within the realm of computers were limited to three simple sources of information: a couple of books, the Commodore 64 of one of my friends, and my beloved programmable calculators.

Books

My grandmother was a mathematician. Actually, she was one of the first female mathematicians ever to graduate in the mid 1920s from the University of Geneva. At home we had lots of science books; I have been exposed to technical literature since the very beginning. Most of it in German, French or English, neither of which I could read back then.

(Funny story, my grandmother was born merely three days after John Von Neumann, and both in the same Empire; she was born in Philippopolis, Bulgaria, while he was born in Budapest, Hungary. But I digress.)

One day, my mother, who was learning to use computers at her job, brought home my first computer book: a copy of the Spanish version of the "MS-DOS Users' Guide" by Paul Hoffman and Tamara Nicoloff, published in 1984 by McGraw Hill.

(She was also learning Lotus 1-2-3 and dBase, but I do not remember her having anything else than a notepad with written notes about those.)

Actually, that MS-DOS guide was not even a real book, but a Xerox'd copy of the actual thing. This was the best my mum could afford. I still have it.

OK, that actually was not my first computer book. My mum had worked at IBM back in the 60s and 70s, and we had a sizeable amount of IBM Mainframe brochures at home. I wish I had kept them.

The MS-DOS book is actually still useful; well, maybe only if you use FreeDOS.

Commodore 64

In 1984, during my 5th grade of primary school, a friend in my class named Hernán Poletti got a Commodore 64 as a gift by her mother. It was, as far as I can tell, the only Commodore 64 in the neighbourhood, and needless to say, my friend became almost overnight the most popular person in the known universe.

Back then my only approach to programming consisted in reading code in programming magazines, of which a few were available in Argentina back then. My mother would find them in some newsstands in downtown Buenos Aires, to my greatest delight. But I had never seen code running; I had no idea how those lines of text could do anything cool on the screen of a computer monitor.

The arrival of that Commodore 64 changed it all; for hours in a row, we would type those lines of code on that clunky and clicky keyboard, and then we would save them in the cassette tape recorder. Yes, kids; back then, there were not even diskettes to save stuff into. We were using good old audio cassettes. As simple as that.

Later on, typing the `LOAD` and `RUN` commands on the screen would (sometimes) make the program actually do something on the screen. Most of the time it did not; and unfortunately, my debugging skills were not enough to solve the problems in those programs. I have never known if the bugs were due to my typing skills or to errors in the source code of the magazine; that point will remain in obscurity forever.

Casio fx-180P

All of this preparation was fruitful; my first exposure to programming happened around 1986, when I received a Casio fx-180P calculator for my birthday, as a present from a distant European relative. I remember looking at this thing in dismay at first, with keys labeled with strange signs, like logarithms, sine, cosine, and other mathematical notations that I had never seen before.

Even worse, the manual was only in French and German. Let us just say that my discovery process of this calculator was slow and painful.

What puzzled me the most was that the keyboard had a button labelled `RUN` on the lower right side; and that was enough to lighten my curiosity. Could this be a small computer? I had seen that same button in the keyboards on computer shops, such as the Sinclair ZX Spectrum or the Talent MSX.

Could this calculator be a computer, albeit a small one?

Much to my dismay, however, most of the functionality on this machine remained obscure. Until one day, during a rainy Saturday in Buenos Aires in 1989, I wrote my first program.

It was a simple program that allowed me to calculate the roots of the quadratic equation:

$$ax^2 + bx + c = 0$$

Which requires a well known formula:

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

The program, once run, would ask me for the three values `a`, `b` and `c`, and would proceed to give me the two possible roots, one after the other. The whole operation would mean just hitting the `P1` key, which triggered the execution of the program; entering the three values, followed by the `RUN` button, and then the first value would appear. Pressing `RUN` a second time would show the second root of the equation.

Was it cheating? Of course it was. I hope Ms Elisa Quastler, my math teacher back in Argentina, will not be too angry to discover that this was the reason I could finish some tests in only a fraction of the time it took others.

First Program

Writing this small program was quite an exercise in itself, from a programming perspective at least. The Casio fx-180P, originally produced in 1980, had a (very) limited memory, and could only accomodate up to 38 instructions; each instruction being roughly equivalent to a single keystroke. It contained six addressable registers (known as `K1` to `K6`), plus the standard “memory” accessed through the classic `M+`, `M-`, `MR` and `MC` buttons.

For reference, the instruction `KIN` reads a value into a register, while, predictably enough, `KOUT` does the opposite. `DISP` stops the execution of the program to show an intermediate value.

The program would simply simulate the sequence of steps that a human being would carry, storing the three coefficients in three of those addressable registers, then performing the operations in roughly this order:

```
KIN 1       # a --> stored in register 1
KIN 2       # b --> stored in register 2
KIN 3       # c --> stored in register 3
4
x
KOUT 1
x
KOUT 3
=
KIN 4       # 4ac --> stored in register 4
KOUT 2
x^2
-
KOUT 4
√
KIN 5       # sqrt(b*b - 4ac) --> stored in register 5
2
x
KOUT 1
=
KIN 6       # 2a --> stored in register 6
b
±           # Negative b
+
KOUT 5
=
/
KOUT 6
=
DISP        # First result displayed, hit `RUN` to continue
b
±           # Negative b
-
KOUT 5
=
/
KOUT 6
=
```

38 Steps

And here it is; probably written around June 1989, my first program, for the Casio fx-180P. It used all of the 38 steps available in the machine. It could not be longer than this, and maybe I could have “refactored” it, to make it shorter or faster, but for my particular needs, it was more than enough.

There was a big issue, though; if

$$b^2 < 4ac$$

then the program crashed; the Casio fx-180P did not have support for complex numbers.

HP 48GX

During my days as a Physics student in the University of Geneva I bought an HP 48GX graphing calculator; compared to my previous Casio, this was a beast. And it is still quite an impressive piece of hardware, 27 years later.

For those wondering, yes, it is almost the same one used by Spiderman’s father in one of the movies; that one was an HP-48G, not the GX. The difference being that the GX had an extension bay for memory and program cards, as well as 128K of RAM instead of 32.

Oh, and did I mention it has an infrared port which can send and receive data using the Kermit protocol? I still have the connection cable, with its RS-232 adaptor, allowing it to talk to any standard IBM PC of its era.

Le sigh.

Quadratic Functions in RPL

The same program as above, but this time in the RPL language:

```
«
'C' STO 'B' STO 'A' STO
B 2 ^ 4 A C * * - √ 'E' STO
B NEG E + 2 A * /
B NEG E - 2 A * /
»
```

Although this might seem esoteric at first, it is very easy to understand once you know that RPL is a stack-based programming language (as is the whole calculator, by the way). In the code above, we just push values to the stack, and any binary operations will simply use the first and second values of the stack to perform its calculation.

The `STO` operation saves whatever value is in the second position of the stack to the variable whose name is in the first position.

As you might expect, the `√` operation is the square root.

To execute this program, save it into a variable, push the values `a`, `b` and `c` (in this order) on the stack, and press the key that corresponds to the program. The values that appear after the execution are the much required solutions to the problem.

And of course, the biggest advantage of the HP 48GX compared to the Casio program above is that it has support for complex numbers, which is not the case for the previous program. This means that our little RPL program can solve all cases.

Coda

As I was finishing the preparation of this article, I found out there is a Texas Instrument calculator that runs Python natively.