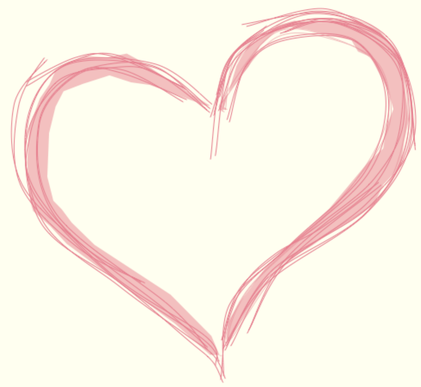


The Art of Computer Science

by Sandy Urazayev

Who is Sandy?

I'm a senior studying Computer Science, Mathematics, and Literature.

I was introduced to programming when I was 13 through Pascal.

I tried myself in industry, research, had a startup with my friend, what now?

With the pandemic starting,
felt incredibly lonely.

Started reading again and fell
in love with literature.

Went on a little break from the
School of Engineering to pursue
my new literature major.

Now I'm back in Computer Science
and I was asking myself...

What is Computer Science?

What does it mean to
be a Computer Scientist?

What is our core identity?

Apparently, no one knows.

Join me one a philosophical adventure on finding out

Who are we?

Computer Science is
engineering, right?

Humanities, maybe?

"Actual" engineers call us fake engineers as we don't engage in classical engineering ways

Then we should be mathematicians!

So, we're not engineers, nor mathematicians, nor humanities?

We do some small part
of it, but nowhere near to
the discipline as a whole

We do have some similarities,
but still farther away than
engineering or mathematics

Let's zoom out

Science and Art

We hear arts and we think of "fine arts", such as painting and sculpture, but it originated with a very different meaning.

Going back to Latin roots, artis/ars meant "skill". Corresponding Greek τέχνη was the root for both "technology" and "technique"

Funnily enough, "science"
meant the same

Distinction between "art" and "science"
has been growing steadily, "science" being
used to stand for knowledge, and "art"
for the application of knowledge

In example, science of astronomy
was the basis for the art of navigation

Similar distinction we draw between
"science" and "engineering"

Science primarily and wholly relies on real-life observations to draw conclusions

One problem has "one correct way" of solving it, as that's the way nature is

$$v_f^2 = v_i^2 - 2g\Delta x$$

$$\Delta E = \Delta K$$

$$d = t/2 * (v_f + v_i)$$

"nature dictates"

The difference between art and science is that science is what we understand well enough to explain to a computer. Art is everything else.

-- Donald Knuth

Back to Computer Science...

Is Computer Science a "science"?

In the grand scale of things,
does this apply to us?

Is there a single "correct" and "accepted" way of writing any program for any given problem?

Absolutely not.

There are as many ways to write
a program as there are foods to eat

One's way is affected by experience,
taste, aesthetic, and MORE!

Sounds like more of
an art to me, doesn't it?

Computer Science is Art!!!

Well, not purely!

We still focus on application-based problems rather than ones of the soul

We came... nowhere?

On the contrary, we realize it might be naive and foolish to try and categorize ourselves into small pretty boxes

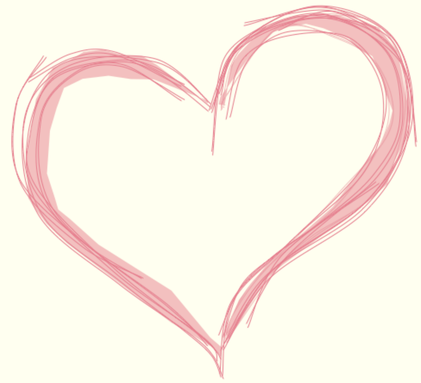
It echoes every field, deserving
a separate discipline of its own

Especially in the Information Age

I wish to view Computer Science
from the lens of science and art

Walking that narrow, yet gentle
line right in-between, in balance

We finally return to our title



The Art of Computer Science

Some programs are elegant, some are exquisite, some are sparkling. My claim is that it is possible to write grand programs, noble programs, truly magnificent ones!

-- Donald Knuth

How do we find that
delicate balance between
"honing our art to perfection",
yet staying "practical" and
"realistic" to the real-world?

How do we find that
delicate balance between
"honing our art to perfection",
yet staying "practical" and
"realistic" to the real-world?

We should try to struck the balance of creating something we feel is beautiful and can deliver the greatest good

Program can be "good" for many reasons. Firstly, it should work correctly. Secondly, it is easy to change when we need to update it or adapt it

It is true art to be build a program that is graceful in its user interactions and can provide meaningful messages on errors and unexpected scenarios.

Like in any other art form, it takes a non-trivial amount of time to develop good taste and intuition for efficiency, optimizations, and structure.

Knowing history is paramount, names like:
Donald Knuth, Alan Turing, Brian Kernighan,
Dennis Ritchie, Ken Thompson, John von
Neumann, Grace Hopper, Margaret Hamilton

Paradigms and methodologies like:
Imperative, Declarative/Functional,
Object-Oriented, Literate, Array-Based,
Structured, Generic, Agent-Oriented, etc.

The question probably follows...

Sandy, why should I care about arts, and history, and philosophies? I'm okay with just making fat corporate stacks

The question probably follows...

Sandy, why should I care about arts, and history, and philosophies? I'm okay with just making fat corporate stacks

That is a completely valid way of thinking and viewing life and

Please let me entertain a thought of why I think it should matter

Those who don't know history are doomed to repeat it.

-- Edmund Burke

Knowing and respecting the ones who
came before us and built the foundation
we stand on today will help us to
build the foundation of tomorrow
for the ones that will follow us

It gives more perspective into what's been tried and worked (or failed), building confidence in yourself and what you can do

Many starting computer scientists and junior programmers are impressionable and susceptible to many weakly structured arguments and points

PHP sucks!!

HTML is not a language

Golang is for bad programmers

Rust is too hard and for elitists

C++ is the worst language ever created

(maybe true)

Do listen what others say, but
come to your own conclusions by
finding support in personal experience
and conclusions drawn from history

We talked about Computer Science as an Art, because it applies accumulated knowledge to the world. Programmers who subconsciously view themselves as artists will enjoy what they do and will do it better

-- Donald Knuth

No matter what

Live YOUR life. Be true to yourself