Computational Thinking

The simple way to solve problems

Computational Thinking (CT) is not just a way of thinking used to solve computer programming problems, it is useful in solving all sorts of problems. Solutions arrived at using CT are usually generic, in other words, the solution can be re-used to solve many similar problems.

Let’s have a look at the steps involved in solving a problem with CT.

- Analyse (think about the problem)
  - What are the constraints, parameters, and assumptions?
  - What are the factors and variables within the problem?
  - What is the objective?
  - What will the solution address?
- Gather information
- Break the problem down into small pieces
- Solve the small pieces
- Put the pieces together in the correct order

Suppose we want to tell a robot how to load a shipping container with boxes which are all the same size.

Think about some of the things we need to know (gather information)…

What are the small problems that must be solved and in what order should they be solved?

Ask yourself – are there any limiting factors or constraints?

Assumptions: “boxes are all the same size.”

- Size is by dimensions, not volume.

Factors

- What are the interior dimensions of the container?
- What are the exterior dimensions of the boxes?
- What is the weight of each box?

Constraints

- Is there a limit to the number of boxes that can be stacked on top of each other?
- How much weight can the container hold?
- Is there an up/down orientation to the boxes? Must the box height be vertically oriented or can be the box be shipped in any orientation?

Looking at the results of our information gathering we can see that there are possible constraints that must be considered before we begin any calculations involving size.

1. Is there a correct side up?
2. Can we stack as many boxes as will fit, or is there a max number of boxes in a stack?
3. Is there a limit to the weight allowed in the container?

If there is not a correct side up, the calculation involving size becomes much more complex, so for today I am going to leave that for you to think about after seeing the logical order of steps involved in solving the big problem.

Here are our first two small solutions:

1. If there is a limit to the number of boxes that can be stacked, we must remember that when making our calculations, and check the results of calculating how many boxes in a stack against the stack limit.

2. When we have finished calculating how many boxes will fit inside the container we must check to ensure that we are not going to overload it.

Now That we know the how to manage our constraints, we can go ahead and calculate the best way to load the boxes on the container.

Do two calculations (W = width, L – length; B = box, and C = container):

- `(W of C / W of B) * (L of C / L of B)`
- `(L of C / W of B) * (W of C / L of B)`

Whichever result is greater is the correct loading pattern to give the robot, along with the number of boxes in a stack (another calculation, with a possible max number) and the maximum number of boxes by weight.

In conclusion we can see that using CT we:

- Thought about the problem
- Considered constraints and limitations
- Broke the problem into small pieces
- Found a number of small solutions
- Put the small pieces together to create the large solution
- Designed a generic solution that can be used to instruct our robot how to load any container