Theory: Components of computational thinking

12221 users solved this topic. Latest completion was about 2 hours ago.

5 minutes 2 / 2 problems solved

Start practicing

Computational thinking is a set of skills that helps you come up with a generalizable solution to a problem and solve it using a computer.
Let's elaborate on the main foundations of this skill, namely: **abstraction, decomposition, pattern recognition**, and **evaluation**.

§1. Decomposition

Complex problems are easier to solve when we break them down into separate steps and subproblems. We do this all the time in our daily life as well. For example, if your goal is to make a pizza, you'd need to find a recipe, shop for all the relevant ingredients, heat the oven, make the dough, chop the toppings and finally bake the pizza for an appropriate amount of time. Breaking down a problem like that helps to focus on solving each subproblem separately (and maybe even reusing solutions other people came up with) and avoid being overwhelmed by the requirements.

§2. Extracting and Generalizing patterns

Once the problem is decomposed our next step would be to look for patterns in the smaller subproblems. A pattern is a set of features in a task that is shared among more than one instance of the problem. For example, if you need to cook Pepperoni pizza, you'd need to make a flat pizza bottom. The bottom is a pattern shared among all the pizzas, so you can generalize this pattern and if you wish to later make Margherita you can reuse all the information and steps you took for preparing Pepperoni pizza.

Pattern recognition is based on the five key steps:

- Identify common elements in problems or systems
- Identify and Interpret common differences in problems or systems
- Identify individual elements within sub-problems
- Describe patterns that have been identified
- Make predictions based on identified patterns.

§3. Abstraction

Abstraction helps us generalize the solution by figuring out a model of a situation. We get rid of all the unnecessary details to focus on what's actually important. One can say that a programming language is an abstraction. You use variables to perform operations instead of listing all the data these variables stand for every time.

It's exactly the abstraction that helps us come up with a generalizable solution without writing a separate program for each variation of the problem.

If we come back to pizza, and try to apply abstraction we would say that for pizza to be a pizza it needs flat dough, sauce, and toppings.

The shape of the pizza and the fact that it has cheese would be unnecessary details since it is possible to imagine square pizza with no cheese.

Therefore if we abstract our pizza like a flat dough open pie with toppings and sauce we would be able to write one program to perform operations on all the pizzas, no matter their taste, shape, and vegan friendliness.

§4. Evaluation

Current topic:

✓ Components of computational thinking 7★ ···

Topic depends on:

✓ Intro to computational thinking 7★ ···

Topic is required for:

Natural language processing ···

✓ Functional decomposition 7★ ···

Table of contents:

Once you found a solution to your problem it is helpful to go through an evaluation checklist. At times your solution won't work and this checklist can also help you to find out where you've gone wrong.

Make sure your solution is:

- Easy to understand. Usually that is the case if the problem is decomposed well.
- Complete. Your solution covers all the aspects of the problem. Remember, the devil is in the corner cases.
- Efficient. Make sure you are making the best of the tools you have, in case of programming it can regard using appropriate syntax constructions. Speed is one of the most important markers of the solution's efficiency and becomes incredibly important when writing production code.

Applying the concepts we discussed is helpful to develop a more efficient and structured problem-solving process. After all, the ability to solve problems is what makes a programmer out of a person who knows a programming language.

Report a typo

886 users liked this theory. **16** didn't like it. **What about you?**

😍 🙂 😐 🙁 😡

Start practicing

Comments (17) Hints (1) Useful links (0) Show discussion