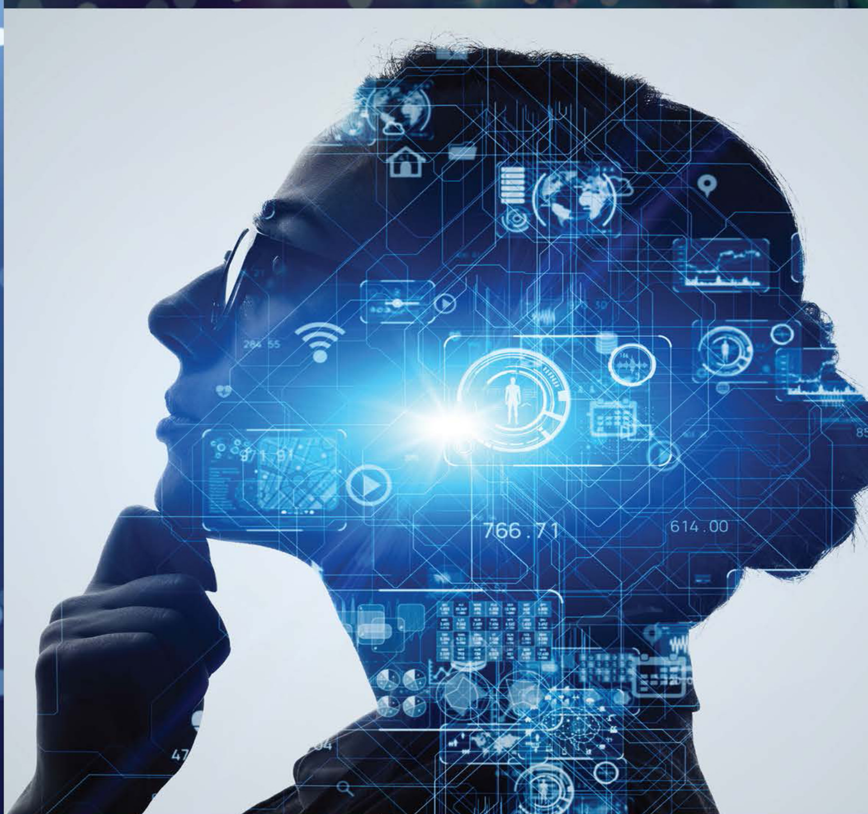
BECOMING A COMPUTATIONAL THINKER

Success in the Digital Age

Paul S. Wang



FREE INSTRUCTOR & STUDENT RESOURCES

CRC Press
Taylor & Francis Group
A CHAPMAN & HALL BOOK

Becoming a Computational Thinker

Becoming a Computational Thinker: Success in the Digital Age has a single purpose: to help everyone become computational thinkers. Computational thinking (CT) is thinking informed by the digital age, and a computational thinker is someone who can apply that thinking everywhere and anywhere. Through practical examples and easy-to-grasp terminology, this book is a guide to navigating the digital world and improving one's efficiency, productivity, and success immediately.

Given their pervasiveness, knowledge and experience of computation are a cornerstone of productivity, and improved thinking will only lead to advances in every aspect of one's life. In this way, CT can be thought of as the mutual reinforcement of thinking and knowledge of computation in the digital age. Comprising a rich collection of self-contained articles that can be read separately, and illustrated by pictures, images and article-end crossword puzzles, this book is an engaging and accessible route to 'Becoming a Computational Thinker' and achieving 'Success in the Digital Age'.

Aimed at the general reader, this book provides insights that can be applied across the full spectrum of industries and practices, helping readers to not only adapt and function in the digital world but also take advantage of new technologies and even innovate new ways of doing things.

Paul S. Wang is an author, computer scientist, researcher, consultant and academic. He is a PhD and faculty member of MIT with over 40 years of experience in teaching and book publishing. His current interest is in introducing computational thinking to all.

Becoming a Computational Thinker

Success in the Digital Age

Paul S. Wang

CRC Press
Taylor & Francis Group
Boca Raton London New York

CRC Press is an imprint of the
Taylor & Francis Group, an **informa** business
A CHAPMAN & HALL BOOK

First edition published 2024
by CRC Press
2385 NW Executive Center Drive, Suite 320, Boca Raton FL 33431

and by CRC Press
4 Park Square, Milton Park, Abingdon, Oxon, OX14 4RN

CRC Press is an imprint of Taylor & Francis Group, LLC

ISBN: 978-1-032-56899-7 (hbk)
ISBN: 978-1-032-56898-0 (pbk)
ISBN: 978-1-003-43768-0 (ebk)

DOI: 10.1201/b23382

Typeset in Latin Modern Font
by KnowledgeWorks Global Ltd.

Publisher's note: This book has been prepared from camera-ready copy provided by the authors.
Access additional online resources at https://computize.org/CTer/

Contents

Preface

Hello, welcome to the delightful pages of

Becoming a Computational Thinker: Success in the Digital Age

The book has a single purpose: to help everyone become a ***computational thinker***. A computational thinker is someone who has acquired computational thinking (CT) and can apply that thinking everywhere, every day. Reading the book, you can better navigate the digital world and improve your efficiency, productivity, and success immediately.

You'll find many interesting and practical articles on CT. We call them *CT Articles*. Each CT Article is short, sweet, and an independent self-contained unit. Each CT Article is written in plain language and amply illustrated by pictures and images. To enjoy the book, you don't need a background in computing, just a sense of curiosity.

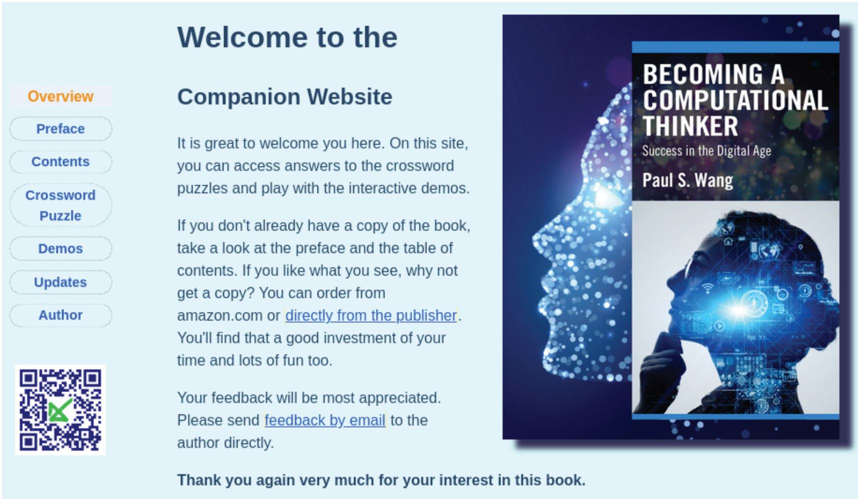
Collectively, the CT Articles paint an increasingly complete picture of computing and digital technologies as well as mental skills inspired by them. You can pick a CT Article that strikes your fancy and read it. There is no need to stick to the sequential ordering. But it is good to start at the beginning.

You'll find the information revealing, interesting, and practical. The CT Articles' applications can be found everywhere, every day. Simply allow the CT Articles to guide your digital journey to better understanding, sharper mental skills, wiser decisions, and more success in everything you do.

To make the book more useful, there is a companion website at `computize.org/CTer` where you can find answers to article-end crossword puzzles, interactive demos, and more.

Companion Website QR Code

So go ahead, kick back, get a cup of tea or coffee, and start reading.



Companion Website: computize.org/CTer

Acknowledgments

I'd like to thank my dear wife Jennifer Wang (葛孝薇) who encouraged me to embark on this project, read all drafts, and provided great feedback, sometimes with specific ideas and wording changes. I am very grateful to her.

Some CT Articles are derived from my CT blog. Some blog articles have also appeared in the online *aroundKent* magazine published by Matt Keffer of Kent Ohio. I am grateful to him for his encouragement and support. The crossword puzzles have been made using CrosswordLabs.com, an excellent website for creating and customizing crossword puzzles.

I'd also like to thank Prof. Michael Mikusa and Mark Weiseman for reviewing the book proposal. I express my deep appreciation to the editors at CRC Press, Randi (Cohen) Slack, Solomon Pace-McCarrick and Michele Dimont, for their help and guidance, and thank the copyeditor Samar Haddad for a careful and thorough job.

Paul S. Wang, Kent, Ohio

王士弘

Computational Thinking: An Introduction

Computers brought us the digital revolution. People in the digital age have such wonderful things as laptops, smartphones, satellite navigation, drones, the Internet, the web, social media, streaming video, artificial intelligence (AI), and more. We are experiencing the *fourth industrial revolution* brought by significant applications of digital technologies and ever-increasing automation in all parts of our economy.

Such changes are tremendously beneficial as well as challenging. For example, we can *ask Google* or *chatGPT* any questions we may have on just about any subject and usually get answers instantly. The Internet spans the globe and brings all parts of the world within instant reach. But in the global village, rumors and fake news can spread quickly. Worse yet, hackers also could steal sensitive information or even hold our computers for ransom.

As modern citizens, we want to take advantage of digital technologies while wisely mitigate the complications and risks brought by them. That's where *computational thinking* (CT) can come into play. When we acquire CT and become *computational thinkers*, we can navigate the "digital sea" more effectively and apply CT every day to make ourselves more successful.

What Is Computational Thinking?

FIGURE 1: *Computational Thinking*

Thinking is and has always been a mental activity informed by experience, knowledge, and logic. Computational thinking (CT) is thinking informed by the computing age.

Knowledge and experience of computation will enhance thinking for all people in the digital age (Figure 1). And improved thinking will also lead to advances in every aspect of our lives, including computing as well as other areas of study.

Thus, computational thinking is the result of mutual reinforcement between thinking and computational knowledge in the digital age .

Specifically, *CT is the mental skill and orientation to apply fundamental concepts and reasoning, derived from modern computing and digital technologies, in all areas, including day-to-day activities.*

CT involves our understanding of computing, its advantages, limitations, and potential problems. CT also encourages us to keep asking questions such as, "*What if we automate this?*", "*What instructions and precautions would we need if we were asking young children to do this?*", "*How efficient is this?*", and "*What can go wrong with this?*".

CT can expand our minds, help us solve problems, increase efficiency, avoid mistakes, and anticipate pitfalls, as well as interact and communicate better with others, people, or machines. A *computational thinker* (CTer) is anyone who has acquired the ability to use and apply CT. A CTer can not only adapt and function well in the digital world, but also take good advantage of new technologies and even innovate and create new ways to do things. A CTer strives to apply CT everywhere everyday.

A Powerful Way of Thinking

OK great, CT is important. But what exactly are the concepts and methodologies it provides? Here is a list of some main aspects of CT:

- Simplification through abstraction—Abstraction is a technique to reduce complexity by ignoring unimportant details and focusing on what matters (Figure 2). For example, a driver views a car in terms of how to drive it and ignores how it works or is built. A user cares only about which mouse button to click and keys to press and generally overlooks how computers work internally.

- Power of automation—Arranging matters so they become routine and easy to automate. Working out a systematic procedure, an algorithm,

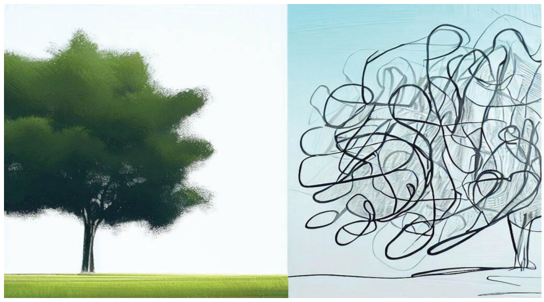


FIGURE 2: *Abstraction Is Commonplace*

for carrying out recurring tasks can significantly increase efficiency and productivity.

- Iteration and recursion—Ingeniously reapplying the same successful techniques and repeatedly executing the same set of steps to solve problems.

- An eye and a mind for details—Small things such as characters in uppercase versus lowercase or with an extra space can make all the difference. Any piece of data may be subject to interpretation depending on the context. You need eyes of an eagle, mind of a detective, and a careful and meticulous approach. Overlooking anything can and will lead to failure.

- Precision in communication—Try telling the computer to do what you mean and not what you say `;-)`. You need to spell it out precisely and completely. Don't spare any details. Vagueness is not tolerated. And contexts must be made explicit.

- Logical deductions—"Cold logic" rules. Causes will result in consequences, whether you like it or not. There is no room for wishful or emotional thinking. Don't we all wish some of this seeps into such things as our politics?

- Breaking out of the box—A computer program executes code to achieve any task. Unlike humans, especially experts, it does not bring experience or expertise to bear. Coding a solution forces us to think at a dumb computer's level (as if talking to a one-year-old) and get down to basics (Figure 3). This way, we will naturally need to think outside any "boxes."

- Anticipating problems—Automation relies on preset conditions. All possible exceptions must be met with prearranged contingencies. Ever said "I'll take care of that later"? Because there is a chance you might forget,

FIGURE 3: *Breaking out of the Box*

according to CT, you should have a contingency plan ready in case you do forget. Otherwise, you have set a trap for yourself.

These are just some of the main ideas. CT offers you many more concepts and ways to think that can be just as, if not more, important. You'll find these CT principles highlighted in articles throughout this book.

Computational Thinkers Computize

In the author's textbook *From Computing to Computational Thinking* (CRC Press, 2015), a new word was introduced.

Definition: **computize**, verb. To apply computational thinking. To view, consider, analyze, design, plan, work, and solve problems from a computational perspective.

When considering, analyzing, designing, formulating, or devising a solution/answer to some specific problem, computizing becomes an important additional dimension of deliberation.

Therefore, a computational thinker is anyone who understands and applies CT, who computizes as a rule, and who can take better advantages of new technologies in the digital age.

CT Protects against Disasters!

People say, "hindsight is 20/20." But, since automation must deal with all possible applications in the future, we must ask "what if" questions and take into account all conceivable scenarios and eventualities. Let's look at a specific example. Hurricane Sandy was one of the deadliest and most destructive hurricanes in US history (Figure 4).

FIGURE 4: *NYC Subway Flooding (Hurricane Sandy)*

With CT at multiple levels, dare we say that many of the disasters from Sandy might have been substantially reduced?

- The New York City subway entrances and air vents are at street level. What if streets are flooded? What if flood water enters the subway?
- What if we need to fight fires in a flooded area? Do we have fire boats in addition to fire trucks? Do we have firefighters trained for boats?
- Most portable emergency power generators run on gasoline. What happens if gas runs out and gas stations are flooded?
- What if the drinking water supply stops? Can we provide emergency water from fire hydrants? In that case, can we use a mobile contraption that connects to a hydrant, purifies the water, and provides multiple faucets?
- What if emergency power generators are flooded? Should we waterproof generators in designated at-risk buildings?
- What if cell towers lose power? How hard is it to deploy portable or airborne (drone?) cell towers in an emergency?
- What if we simulate storm damage with computer modeling and find out ahead of time what to prepare for?

So let's computize at multiple levels and do our best to get 20/20 hindsight beforehand.

CT and DNA: a Success Story

In computing, performing the same set of steps repeatedly to achieve a certain goal is a technique known as *iteration* which is an important concept in CT.

Iteration of a process has led to the invention of the *polymerase chain reaction* (PCR), a technique in molecular biology to generate thousands to millions of copies of a particular DNA sequence.

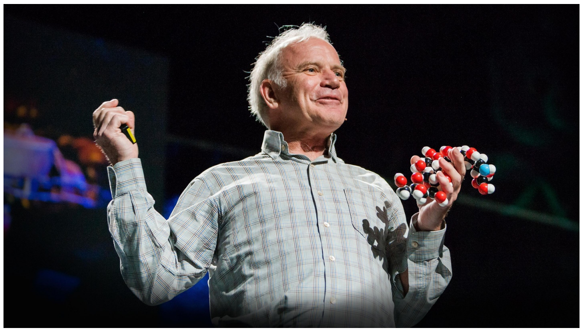


FIGURE 5: *Dr. Kary Mullis, TED Talk, 2009*

Developed by Dr. Kary Mullis (Figure 5) in 1983, PCR is now indispensable in medical and biological research and applications, including DNA testing and genetic fingerprinting. The impact of automated PCR is huge and far-reaching. Mullis was awarded the 1993 Nobel Prize in Chemistry for his part in the invention of PCR (Figure 6).

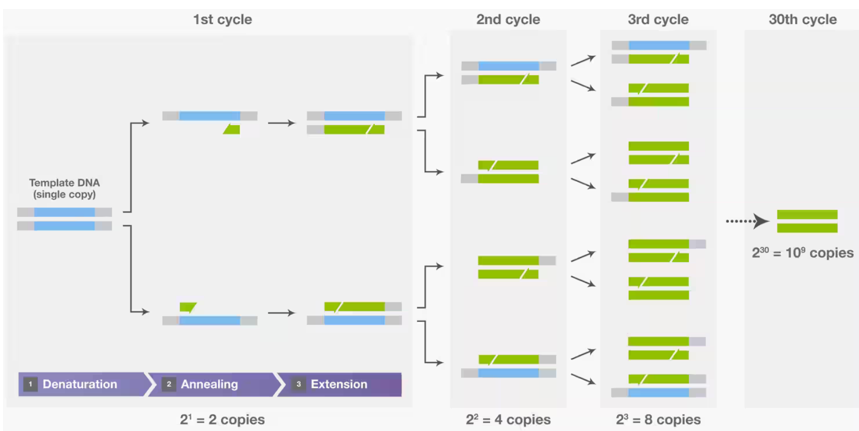


FIGURE 6: *Polymerase Chain Reaction (PCR)*

In recounting his invention, Dr. Mullis wrote in his book *Dancing Naked in the Mind Field*:

> *I knew computer programming, and from that I understood the power of a reiterative mathematical procedure. That's where you apply some process to a starting number to obtain a new number, and then you apply the same process to the new number, and so on. If the process is multiplication by two, then the result of many cycles*

> *is an exponential increase in the value of the original number: 2 becomes 4 becomes 8 becomes 16 becomes 32 and so on.*
>
> *If I could arrange for a short synthetic piece of DNA to find a particular sequence and then start a process whereby that sequence would reproduce itself over and over, then I would be close to solving my problem.*

At the time of the invention, the "polymerase" and other related DNA duplication techniques were already known. It was the "chain reaction" part that was missing. Well, we have Dr. Mullis and his computational thinking to thank for the invention.

And what a significant invention! *The New York Times* described it as "highly original and significant, virtually dividing biology into the two epochs of before P.C.R. and after P.C.R.". Today, DNA technologies have been applied widely and changed our lives in countless ways, in medical science, law enforcement, biology, and so on.

Furthermore, major breakthroughs attributable to CT also include weather modeling and prediction, machine learning in Artificial Intelligence (AI) and many other disciplines. Such advances help motivate CT in all areas.

Problem-Solving

A central idea in CT is problem-solving. Some CT experts would even say "CT is nothing but problem-solving." Not everyone is willing to go that far, But, it is certainly true that problem-solving is at the center of CT.

The usual or colloquial meaning of '*problem*' is often a certain difficulty to overcome when trying to do something. But, from the CT point of view, **almost anything one does becomes a problem of how to apply CT to it**.

Thus, to a CTer, a problem can be any or a combination of these: a project, task, mission, goal, challenge, difficulty, effect, outcome, or purpose. And the meaning of *solving that problem* becomes "finding a systematic, ingenious, step-by-step, effective, and efficient way to achieve, perform, or overcome it." Such solutions can potentially be automated using well-designed algorithms or procedures.

Here we define a new word *problemize*:

Definition: **problemize**, verb. To consider or treat something as a problem of how to apply CT to it.

Realizing the power of CT, CTers have the attitude to problemize, no matter how annoying and tedious it may seem.

CT Article 1, "*Everyday Computational Thinking Can Save Lives*" will discuss how to *problemize daily activities* such as leaving the house and going to bed. Hence, we can say that CTers tend to problemize all the time.

Time Well Spent

One does not become a computational thinker overnight. But reading CT Articles here is definitely a direct and enjoyable approach. As history has demonstrated time and again, a person who is better educated in the next dominating technology and who can absorb new thinking into common sense will have a significant competitive edge. You know we are talking about a computational thinker.

Happy Reading!

1

Everyday Computational Thinking Can Save Lives

Algorithms and their design and implementation are critical to modern computing.

Today we live in a global village. The Internet, the web, and the computer in its many different forms are providing instant communication across vast distances and changing almost every aspect of our lives. All the wonderful changes have been made possible by ingenious algorithms running on computing machines.

Here we'll see what an algorithm is and how flowcharts can help us envision procedures and formulate step-by-step solutions to problems. The knowledge is basic and key to computational thinking (CT).

Also in this CT Article, we will explain how algorithm-inspired CT can be applied by everyone in everyday situations and how it can make important differences and even save lives.

What Is an Algorithm?

Remember the hugely successful movie *The Social Network* (2010)? It told the story of Mark Zuckerberg and how he created Facebook. The motion picture also introduced the term *algorithm* to much of the world for the first time.

Simply put, *an algorithm is a step-by-step procedure designed to perform a particular task or to solve a specific problem.* Such a procedure forms the basis for writing a program that will perform the procedure on a computer. Thus, algorithms are fundamental to computing and therefore important for computational thinkers.

The origin of the word "algorithm" traces back to the surname Al-Khwārizmī of a Persian mathematician (780–850 CE), who was well known for his work on algebra and arithmetic with Indian numbers (now known as Arabic numbers). The modern-day meaning of algorithm in mathematics and computer science relates to an effective step-by-step procedure to solve a given class of problems or to perform certain tasks or computations (Figure 1). Specifically, a procedure becomes an algorithm if it satisfies all of the following criteria:

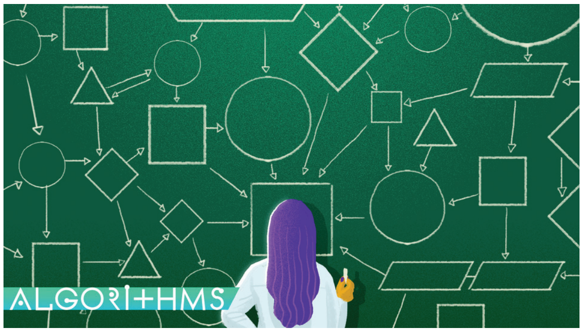


FIGURE 1: *Creating an Algorithm*

- **Finiteness**: The procedure consists of a finite number of steps and will always terminate in finite time.
- **Definiteness**: Each step is precisely, rigorously, and unambiguously specified.
- **Input**: The procedure receives certain data (or none) as input before it starts. Possible values for the data may vary within limitations.
- **Output**: The procedure produces results as its output.
- **Effectiveness**: Each operation in the procedure is basic and clearly doable.

For a given problem, there usually are multiple algorithms for its solution. The design and analysis of algorithms are central to computer science and programming.

What do algorithms have to do with everyday computational thinking? Good question. Well, it has to do with setting goals, devising concrete steps to achieve them, anticipating problems, and arranging solutions in advance. That's pretty important for everything every day, right?

Flowcharts

An algorithm is basically a procedure to achieve a certain goal.

A *flowchart* presents a procedure visually with words and diagrams. For any procedure, we can use a flowchart to plan the sequences of steps, to refine the solution logic, and to indicate how to handle different possibilities. Here is a simple flowchart (Figure 2) for the task of "getting up in the morning." We begin at the `Start` and follow the arrows to each next step. A diamond shape is used to indicate a fork in the path. Which way to turn depends on the conditions indicated. Obviously, we use diamond shapes to indicate

FIGURE 2: *A Wake-Up Flowchart*

possibilities. The snooze option leads to a branch that repeats some steps. In programming, such a group of repeating steps is called a *loop*. The procedure ends when the person finally climbs out of bed.

As another example, let's look at a flowchart for troubleshooting a lamp (Figure 3). The very first step after `Start` is significant. Although the purpose

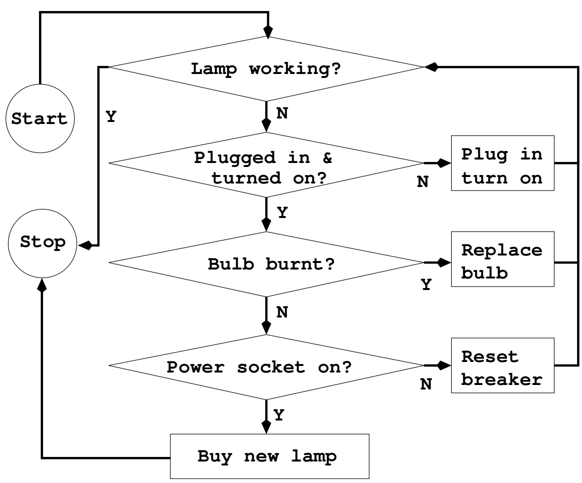


FIGURE 3: *Lamp Fixing*

of the procedure is to troubleshoot a lamp, we, nonetheless, make no implicit assumption that the lamp is not working. Without this step at the beginning, the procedure would potentially troubleshoot a perfectly good lamp, and, worse yet, would decide to replace it with a new lamp!

Each of the next three steps tests for a particular problem and makes a fix. Then the same procedure is reiterated by going back to step one to determine if the lamp is now working. This flowchart is a bit more complicated. Yet, it is worth careful examination (Figure 4) by following the steps. That is also a good way to get into the head of a programmer.

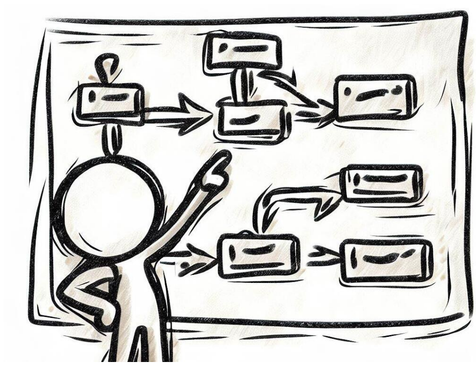


FIGURE 4: *Following a Flowchart*

All you need is pencil and paper to start drawing your own flowcharts. Try it and you may find it not so difficult. To make nice looking flowcharts, you can find many tools on your computer as well as online.

Correctly setting goals and guarding against potential problems are important aspects of devising a procedure. We want to avoid fussy, vague, confused, wishful, emotional, impulsive, optimistic, or pessimistic thinking. We want to practice CT any time and anywhere we can. We want to set clear goals, have a sequence of steps to achieve them, anticipate problems, and prepare solution plans in advance.

CT and Driving

Take driving a car for example. What is the goal? It is to get to a destination safely. It is not enjoying the sound system, watching the scenery, or engaging in conversation, although we have nothing against any of that as long as it does not get in the way of safe driving. Texting and driving is never safe.

Stopped at a traffic light, we wait for the light to turn green. But, we may need to run the red light if an 18-wheeler is about to crash into us from behind. That means we need to be checking our rearview mirror while waiting for the green light. When the light turns green, do we blindly rush into the intersection? What if a car is running the tail end of the yellow light or the red light?

Thus, the goal is not to obey traffic signals, but to ensure safety. In the United States, a car crash kills a person every 12 minutes on average. If you are thinking straight, is a car a fun machine or a dangerous one? CT can keep you focused on the goals, make you pay attention to details, plan for contingencies, and shield you from distractions. CT can save the day, and perhaps even your life!

FIGURE 5: *Preflight Checklist*

Now let's apply CT to the task of "getting ready to drive a car" and write down an algorithm-inspired predrive checklist like the one used by airline pilots (Figure 5).

1. Am I ready to leave? Forgot to bring anything?
2. Walk around the car, check windows, tires, lights, back seat, and any objects and activities near the car.
3. Get in the car, foot on brake, close and lock all doors.
4. Adjust seat and steering column positions as needed. Check positions of all rear-view mirrors, buckle up.
5. Check the instrumentation panel, pay attention to the fuel level.
6. If necessary, familiarize yourself with the controls for lights, turn signals, wipers, heat/AC, and emergency signal. Make sure they are working properly.
7. Release the hand brake, start the engine, shift gear.
8. Make sure the gear is in D or R as intended, then start driving.

Airlines have developed rigorous preflight checklists for safety. Incidents, sometimes fatal, happen when pilots and crew, failing computational thinking, do not follow the exact procedure. The same goes for doctors and nurses in hospitals, especially in operating rooms.

More Everyday Applications

Driving a car is not the only thing we do every day. Let's see how CT can be used to our advantage in other common activities.

We see an algorithm has a well-defined beginning and ending. This leads to the following CT principle:

> **CT concept–*Get ready before you start***: *If you are not prepared, forget something you need, or get underway in a hurry, you may often find yourself in trouble or even danger.*

Thus, when you leave home, work, school, or any other place, you need to get ready before you go! You don't want to leave anything behind or forget to bring anything you need later. Ever heard of people going to a show but forgot to bring tickets? In computing when you begin a procedure or algorithm, you need to make sure to set up the correct context. The same goes for daily living.

FIGURE 6: *Working in the Yard*

Leaving one place for another is the most frequent context change we do, and if we pay attention we can avoid trouble and be safer too. For example, when you go outdoors to work in the yard or garden (Figure 6), don't forget to wear sunglasses, hat, gloves, long pants, and working shoes. Apply sun screen lotion if necessary too. Oh, bring your cellphone with you in case someone might call. Surely you can think of many other daily situations where getting ready before you start becomes such good advice. How about forgetting to charge your phone or gas up your car before a trip? It can mean life or death indeed.

CT also encourages symmetric thinking. Getting ready at the beginning also reminds us to do the same at the end. When you leave a restaurant or some other such place, how often have you left your sunglasses, umbrella, hat, gloves, pen, or water bottle behind (Figure 7)? After finishing a call, you

FIGURE 7: *Sir, Your Water Bottle?*

might put the cellphone down casually? Put it back where it belongs. How many times have you misplaced your phone and can't find it? How about your scissors or another tool (Figure 8), car keys, sunglasses, even wallet? Hence, we have the symmetric CT principle:

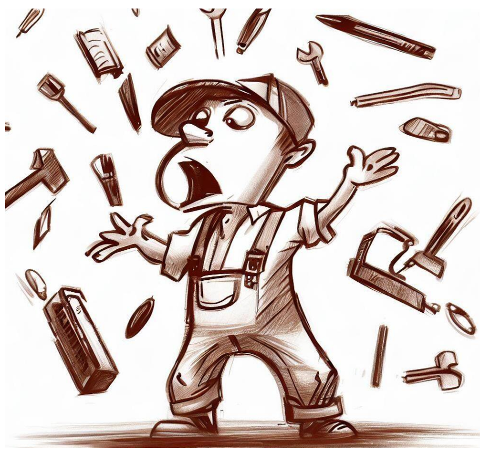


FIGURE 8: *Where Is That Tool?*

> **CT concept–*Get set after the end***: *Returning things to their undisturbed state can help a lot in the future.*

All this CT leads to a good habit:

> **CT concept–*Checklisting***: *Write down a complete and detailed list for starting, performing, and finishing important tasks.*

Even a shopping list can be a good thing everyday.
We all go to bed at night and wake up in the morning every single day.

There are two chances per day to practice leaving one state and entering another. So get into a restful state for a good night's sleep and wake up to an energetic state ready to face a new day. Who said we couldn't practice computational thinking everyday?

A flowchart or an algorithm has another characteristic, namely sequencing of the steps. This leads to:

> **CT concept–*Activity sequencing***: *The order in which things got done matters.*

We all have loads of things to do every day, but we can become much more efficient if we plan ahead and decide what to do first and what next. Doing things in a logical order can make life go smoothly indeed.

Here we mentioned just several CT ideas inspired by a few computational techniques such as flowcharts and algorithms. As you know more about digital and computing technologies, by reading CT Articles here, for example, you'll discover many other CT ideas making you an even better computational thinker and helping you succeed in the digital age.

You Can Do It!

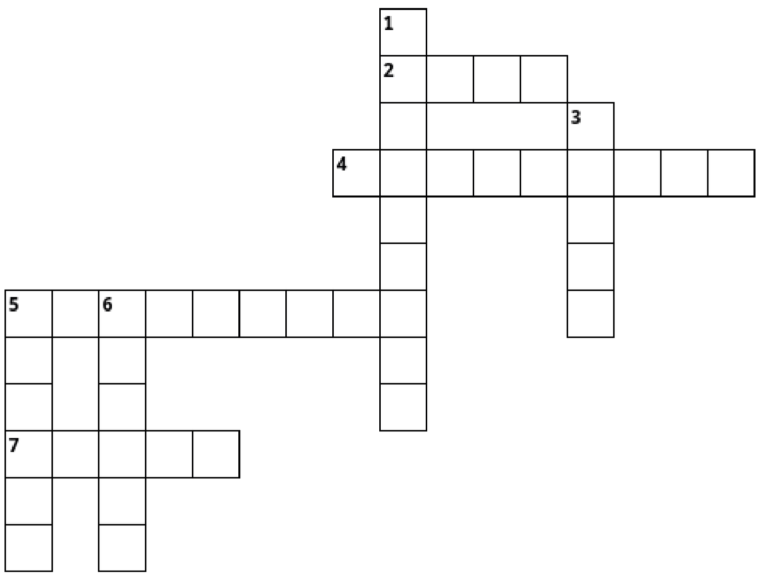
We have seen a sampling of computational thinking, inspired by algorithms and how computers follow step-by-step instructions. More importantly, we see how CT can be applied in everyday situations. We see the advantages CT can bring if applied well.

For most people, CT does not come naturally. Research has shown that snap judgment based on intuition and experience is the norm. But, by keeping CT in mind and consciously making it an added dimension to our deliberations, we can make it work for us.

By being creative, everyone can derive benefits from CT every single day. As a result, our community, even the entire society, will be better off by becoming more efficient and effective.

CT Crossword Puzzle

Basic Concepts



Across

2. Repeating a set of steps

4. Applying CT

5. Chart of a procedure

7. Data received

Down

1. Step-by-step procedure

3. In the beginning

5. At the end

6. Data produced