The IB Computer Science Textbook

Last Edit: March 2, 2018
Contributing Authors: See Annex A

This textbook was written with the express purpose of educating and helping International Baccalaureate students. The content within this textbook is comprised of user-submitted information, verified and validated through peer-checking and online research. Use it well.

This document and other Bible documents are maintained on the r/IBO Discord server. https://discord.gg/IBO for more information.

To submit to this textbook or other documents within the Bible, please see https://github.com/pants1/CS-Bible.

Contents

Foreword

The IB tries to hit home with the fact that students must be using '**Computational Thinking**'. This is within every mark scheme, in the subject guide, in the IA, and in the EE. This is something you need to display whenever you are tested on your Computer Science knowledge. So, remember this.
Computational Thinking.
Computational thinking is a problem-solving methodology that is applicable across a range of subject disciplines and underpins this course. The six principles of computational thinking are:

- Thinking procedurally
- Thinking logically
- Thinking ahead
- Thinking concurrently
- Thinking abstractly
- Thinking recursively

(From the IB Computer Science subject guide)

1 System fundamentals

1.1 Systems in organisations

Topic Overview.
Computer system fundamentals help with understanding how a complex network of computers and people who operate in organizations work. Some of the most important aspects of this topic are strategies & planning, managing wide-scale software development, and system back-ups for data safety. The topic of system fundamentals also outlines components of a computer, ethics for computer scientists, and human interactions with machines. These are only outlines of what is to come in this topic.

What is a system? The beauty of metaphors.
Firstly, **we must define what a system is. A system is an interconnected set of parts that work together to perform a common goal.** Systems are usually organized and follow certain methods every time they operate. One simple example of a system may be your bag. Firstly, think about the components of your bag: zippers, compartments, straps, handles, soft outer material etc. So, we can observe that these components work together towards a common goal: storing and transporting relatively small items. They're all very different things but we associate them with each other because they work systematically in our everyday lives. This is a very simple example of a system, but as systems get more complex and advanced they tend to use levels of *abstraction* (term for later on in the topic).

Because this is *computer* science and we have established what a system is, we must then define **computer systems**. Going from bags to computers is a huge step, but this is what this topic takes us through (bags not included though). Understanding the fundamentals of a system and relating them together is what helps engineers build such complex systems, it's all about working from the bottom up. **Computer systems are machines made of hardware and software working synchronously to receive, process, display, and output information.** Though these are not requirements for a computer to be a computer. For example, a toaster does not process or output information, but it runs on a chip we would call a computer.

What is abstraction?
Layers of abstraction are very important in computer science and any field of engineering. They are important because they save time and effort which is not always needed. Layers of abstraction are when you build upon lower-level more fundamental concepts to higher-level more brief concepts. Going back to the bag system, instead of saying the system with zippers, compartments, straps, we call it a bag. And anyone who knows what a bag is already understands what is going on. Imagine having to explain what bag is every time you need it or talk about it, it wastes a lot of time and effort with information you do not need! Which is why levels of abstraction are important for computer scientists when they talk about systems as a whole.

1.2 System design basics

2 Computer organization

3 Networks

4 Computational thinking, problem-solving and programming

5 Abstract data structures (HL Only)

6 Resource management (HL Only)

7 Control (HL Only)

8 Option A: Databases

9 Option B: Modelling and simulation

10 Option C: Web science

11 Option D: Object-oriented programming (OOP)

12 Case Studies

13 General Help

https://www.sharelatex.com/learn/

A Contributors

Pants#0001 (Curator)
nullptr#1380
9frazz#5253
batmansmaster#289