|  |  |
| --- | --- |
|  | Concept Assignment 2  PLTW Computer Science CSP Core Training |

# Abstraction

|  |  |
| --- | --- |
|  | Learning Objectives |

LO2.1 While working through Activity 4.1.2, the teacher will:

* Relate the changes in voltage states (i.e., on and off) of transistors to their representations in binary: the bits 1 and 0.
* Illustrate how combinations of switches can form logical units called gates, and that combinations of gates can perform computation.
* Show that assemblies of logic gates are the substance of integrated circuits (ICs).
* Identify where bits, switches, gates and ICs fall on the ladder of abstraction.

LO2.2 While working through the content of Lesson 1.2, the teacher will:

* Translate decimal numbers to binary (and vice versa) to demonstrate that base-10 numbers can be represented as bits (1s and 0s), and that a series of bits can represent a base-10 number.
* Explain how a series of bits can represent a symbol, such as a key on a keyboard.
* Recognize that one hexadecimal digit replaces four binary digits and understand why this is useful.
* Appreciate that colors can be represented by a series of hexadecimal digits.
* Assign a data type to a variable based upon its role in the program: Boolean for true/false, lists for aggregated data, integers for whole numbers.
* Connect the use of decimal, binary, hexadecimal, data types, and data roles to the ladder of abstraction.
* Differentiate variable type from variable role.

LO2.3 Based on a review of sections 1.1.5 and work in section 1.2.4, the teacher will:

* State the eight variable roles and give examples of how each might be used in a program, procedure, or function.
* Categorize the usage of variables, by role, within a program written by someone else.
* Communicate with others regarding the use of variables using the nomenclature of variable roles.

|  |  |
| --- | --- |
|  | Introduction: Abstraction Defined |

In computer science, as well as in other fields, abstraction is a technique for simplifying complex systems. It works by ignoring or removing less relevant detail, maintaining focus on essential elements of the overall task or the task at hand. Computer scientists speak of moving up and down the ladder of abstraction: “going up the ladder” means hiding more and more complexity, whereas “moving down the ladder” means revealing more and more complexity. This technique allows one to understand the need and placement of specific detail. It allows the practitioner to grasp the big picture first before filling in all the details. Included as one of the Seven Big Ideas in AP’s Computer Science Principles course, abstraction is defined by the College Board as:

… a process, a strategy, and the result of reducing detail to focus on concepts relevant to understanding and solving problems.1

Specific types of abstractions examined and used in this course include hardware, data, procedural, problem-solving, and model abstraction. This day will focus on hardware, data, and procedural abstraction. Throughout the course, we will refer to and use abstraction. In later activities, we’ll examine problem-solving and model abstraction in more detail.

|  |  |
| --- | --- |
|  | AP CSP Enduring Understandings (EU) and Learning Objectives (LO) |

Abstraction is the second Big Idea in the AP Computer Science Principles (CSP) course. Our curriculum will address the following Enduring Understandings (EU) and Learning Objectives (LO) on Day Two:

* Describe the variety of abstractions used to represent data. CSP.LO2.1.1 [P3]
* At one of the lowest levels of abstraction, digital data is represented in binary (base 2) using only combinations of the digits 0 and 1. CSP.EK.2.1.1E
* The interpretation of a binary sequence depends on how it is used. CSP.EK.2.1.2D
* Explain how binary sequences are used to represent digital data. CSP.LO2.1.2 [P5]
* A sequence of bits may represent instructions or data. CSP.EK.2.1.2E
* Use multiple levels of abstraction to write programs. CSP.LO2.2.2 [P3]
* Binary data is processed by physical layers of computing hardware, including gates, chips, and components. CSP.EK.2.2.3E
* A logic gate is a hardware abstraction that is modeled by a Boolean function. CSP.EK.2.2.3F
* A chip is an abstraction composed of low-level components and circuits that perform a specific function. CSP.EK.2.2.3G
* Hardware is built using multiple levels of abstraction, such as transistors, logic gates, chips, memory, motherboards, special purposes cards, and storage devices. CSP.EK.2.2.3I

|  |  |
| --- | --- |
|  | Part 1: Hardware Abstraction |

Abstraction is so ubiquitous in our lives that much of the complexity in what we see and do is hidden from us. In your role as a computer science teacher, you will uncover much of this complexity for your students. You will be climbing down the ladder of abstraction, going “under the hood” to examine and understand how things work: in this case, the creation of computing’s foundational elements. The artful use of abstraction can make the process of understanding easier and more effective, which is why the topic is so important.

In this section, we uncover the generation of bits (1s and 0s) at the hardware level. Some may find the material in this unit unfamiliar. Rather than being at a disadvantage, such individuals have an opportunity to examine what effective pedagogy looks like. In conveying new and strange concepts to anxious learners, pay particular attention to the manner of language used to instruct, the nonverbal signs used to encourage and/or support students, and the eagerness or reluctance with which participants address confusion.

1. Your Master Teacher will give a high-level overview of the concepts investigated in Activity 4.1.2 (AND, NAND, OR, NOR and Inverter gates work, discrete electrical components).
2. Review the components available in the electronic kit and complete a circuit on a breadboard as directed by your Master Teacher.

|  |
| --- |
| Submission Item |
| 1. How does the concept of abstraction relate to hardware in computers? |

|  |  |
| --- | --- |
|  | Part 2: Data Abstraction |

All digital data is represented in 0s and 1s. The same 0s and 1s can represent many types of data, including:

* an integer
* a rational number, also known as a **floating-point** value or float
* a string of characters
* a list or a tuple
* a color
* an image
* a sound
* a video

One or more **data representations** are commonly used to encode each of these types of data. A particular encoding must be used to represent data in 0s and 1s, and the same encoding must be used to retrieve the data from the 0s and 1s. Computer programming languages take care of these underlying details for each type of data representation. We say that a language offers a **data abstraction** for each type so that the programmer doesn’t have to worry about the details.

However, there are many reasons that the programmer and the user need to understand some of the details. Integers can have maximum and minimum values before “rolling over.” Floating-point numbers have limited precision and can accumulate rounding errors. String encodings vary between programs, so cutting from one program and pasting into another program can result in strange characters. Images, sound, and video offer trade-offs between resolution and file size for storage and transmission.

1. Work through Activity 1.2.1. You may complete the activity without building blocks, but using them favors the kinesthetic learner and may provide a more concrete experience for those who need or want that. Be sure to emerge from the activity with the ability to convert between binary and decimal bases.

|  |
| --- |
| Submission Item |
| 1. Answer question 17 a and b (Part V) and Conclusion Question 1 from Activity 1.2.1. |

1. Activity 1.3.1 illustrates how binary sequences are used as microprocessor commands. The data representation shown in the activity is at a much lower level of abstraction than a Word document or a Scratch program. Assembly language programmers live at this level of abstraction.

Activities 1.1.5 and 1.1.6, Variable Roles I and II, introduce the concept of variable roles with Scratch. The eight variable roles highlighted in the course are: **Fixed**, **Most Recent**, **Accumulator**, **Aggregator**, **Stepper**, **Walker**, **Best-So-Far**, and **One-Way Flag**.

While many professional programmers make it through their entire careers without formally classifying variables, your job as a computer science teacher is made easier and more effective by using this tool of abstraction. Once committed to memory, these variable roles evoke an associated context and, usually, a variable type, such as integer, Boolean, or list. That association can speed the design of a procedure or help students understand how a particular variable functions in existing code. Activity 1.2.4 further develops understanding of variable roles using a pre-made project in MIT App Inventor. In this activity, the context is the evolution of a mobile app called bounceBall.

1. Download and examine several versions of the program provided and examine its variables and associated roles. Refer to the Teacher Guide and use the answer key to see how different parts of the code are identified as having different variable roles.

|  |
| --- |
| Submission Item |
| 1. Explain the eight variable roles that have been discussed. This can be as simple as a list with definitions or as elaborate as a graphical representation of each role. Create something that you can use in your classroom to help your students understand the roles of variables. |

|  |  |
| --- | --- |
|  | Part 3: Considering Classroom Implications |

* What can you, as the teacher, do to promote a growth mindset in your classroom?
* What types of speech, behavior, or nonverbal language can support your intentions? What kinds of things undermine those intentions?
* Metaphor is a powerful pedagogical tool. Think of a metaphor that highlights or illustrates the difference between an accumulator and an aggregator, and between a walker and a stepper.
* Writer’s block is a condition that affects writers. A similar complaint is not knowing where to start in the construction of a program. How is abstraction helpful here? What must the teacher focus on to take advantage of this tool?

|  |
| --- |
| Submission Item |
| 1. Write a reflection about the things you learned today. Consider highlighting new things you’ve learned, items you need to consider for implementing this in your classroom, and ideas and suggestions you’ve heard from others. Use the questions above as prompts, but don’ot feel limited or constrained by just those questions. |

1 AP Computer Science Principles Curriculum Framework, p. 14