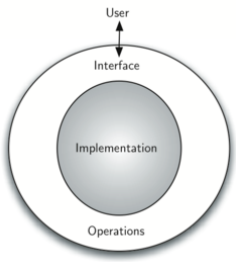
1.5. Why Study Data Structures and Abstract Data Types?

To manage the complexity of problems and the problem-solving process, computer scientists use abstractions to allow them to focus on the “big picture” without getting lost in the details. By creating models of the problem domain, we are able to utilize a better and more efficient problem-solving process. These models allow us to describe the data that our algorithms will manipulate in a much more consistent way with respect to the problem itself.

Earlier, we referred to procedural abstraction as a process that hides the details of a particular function to allow the user or client to view it at a very high level. We now turn our attention to a similar idea, that of **data abstraction**. An **abstract data type**, sometimes abbreviated **ADT**, is a logical description of how we view the data and the operations that are allowed without regard to how they will be implemented. This means that we are concerned only with what the data is representing and not with how it will eventually be constructed. By providing this level of abstraction, we are creating an **encapsulation** around the data. The idea is that by encapsulating the details of the implementation, we are hiding them from the user’s view. This is called **information hiding**.

Figure 2 shows a picture of what an abstract data type is and how it operates. The user interacts with the interface, using the operations that have been specified by the abstract data type. The abstract data type is the shell that the user interacts with. The implementation is hidden one level deeper. The user is not concerned with the details of the implementation.



(../_images/adt.png)

Figure 2: Abstract Data Type

The implementation of an abstract data type, often referred to as a **data structure**, will require that we provide a physical view of the data using some collection of programming constructs and primitive data types. As we discussed earlier, the separation of these two perspectives will allow us to define the complex data models for our problems without giving any indication as to the details of how the model will actually be built. This provides an **implementation-independent** view of the data. Since there will usually be many different ways to implement an abstract data type, this implementation independence allows the programmer to switch the details of the implementation without changing the way the user of the data interacts with it. The user can remain focused on the problem-solving process.

You have attempted 1 of 1 activities on this page

(WhatIsProgramming.html) **Next Section - 1.6. Why Study Algorithms? (WhyStudyAlgorithms.html)**

(WhatIsProgramming.html)

(WhyStudyAlgorithms.html)

✔ **Completed. Well Done!**