1. Data Structures and Algorithms :: Introduction to Data Structures

PROGRAMS = DATA STRUCTURES + ALGORITHMS

Computer programs and data go together, i.e., we cannot have one without the other. Consequently, we must have ways in programs to store data, manipulate it, display it, transmit it, receive it and so on.

A **data structure** is a particular "way" of storing data in memory (or on disk, but we will only concern ourselves with memory) so that we may efficiently operate on that data.

An array is the simplest data structure in most programming languages, including Java. Arrays have their uses but in general they are sufficient for only a limited subset of problems.

Consequently, over the past 70-some years that programmers have been writing computer programs, numerous types of data structures have been created for many different types of programming problems. These data structures are the primary tools in a programmers toolbox. Every program—other than the simplest types of programs you learn to write when you are first learning programming—uses data structures.

1. Data Structures and Algorithms :: Introduction to Data Structures (continued)

The field of data structures is so large that specific courses in all computer science degree programs are devoted to it. CSE205 serves as an introduction to data structures and in this course we will study four of the most commonly used types of data structures:

Linked lists
Stacks
Queues
Trees

1. Data Structures and Algorithms :: Introduction to Algorithms

In computing, an **algorithm** is a sequence of instructions written in some language that when executed solves a problem or performs some task in a finite amount of time. Implicit in this definition is the requirement that an algorithm must terminate.

The classic algorithm analogy is a cooking recipe. If we wish to make a chocolate cake—we are doing it the old-fashioned way rather than adding water to a box mix—then there specific steps that must be followed in a prescribed order. These steps involve ingredients, which are analogous to data in a computer program. Assuming we use the proper ingredients and follow the recipe exactly, then at the end we will have produced a (very yummy) chocolate cake.

However, failure to use the proper ingredients (e.g., using baking soda rather than baking powder), skipping a step (not putting the cake batter in the oven to bake), or completing steps out of order (adding the eggs to the mix before breaking the eggs and discarding the shells), then we are going to end up with something that is not, in any way, an acceptable chocolate cake.

Algorithms are the same way. If we correctly employ an algorithm on incorrect data (the wrong ingredients) then the output from the algorithm will be useless (we refer to this as the **garbage-in-garbage-out** or **GIGO** principle in programming).

On the other hand, if the data not garbage, but we do not write the code to properly implement the algorithm (skipping steps or performing steps out of order) then the output from the algorithm will again be useless.

1. Data Structures and Algorithms :: Operations on Data Structures

Data structures and algorithms go together (like chocolate and peanut butter). A data structure details the way we store the data and the operations we can perform to use that particular data structure.

An algorithm describes how we performs those operations to use the data structure, so during our discussion of data structures, we will also be examining algorithms for using those structures.

In general, the three primary operations that we perform on all data structures are:

1. Add data to the structure.
2. Remove data from the structure.
3. Search the structure for specific data.

Most data structures will support operations other than these, but these are the "big three".

In conclusion, at a certain level a computer program is nothing more than an algorithm that accepts data as input, stores and transforms that data in various ways, and produces data as output.