Data structures, arrays, and modularizing

ASSIGN BUSTER

In a computer, “ the data structure becomes an object that includes data and functions” (Robertson, 2007). It stores and organizes the data in a computer. Data structures make it possible to sort through large databases and internet indexing services. These would include online libraries, such EBSCOhost through universities, and search websites, such asGoogle.

These would also include member profiles such as Facebook. Arrays “ provide the programmer with a way of organising a collection of homogeneous data items (that is, items that have the same type and the same length) into a single data structure. An array, then, is a data structure that is made up of a number of variables all of which have the same data type” (Robertson, 2007). For example, there are 40 students in the first grade. A single variable name such as ‘ first’ can be used with all 40 first grade students. “ The individual data items that make up the array are referred to as the elements of the array. Elements in the array are distinguished from one another by the use of an index or subscript, enclosed in parentheses, following the array name” (Robertson, 2007), such as ‘ first (8)’.

This subscript refers to the eighth student in the first grade. The smallest element of an array must also be the size of the largest element, so in using the months of the year as an example, the names of the shorter months must be padded with blank spaces. The month of May must be as long as the month of September. Modularization breaks down the complexity of programming. That means that the complex problem gets divided into smaller parts, thus creating subtasks or functions from the major tasks. The subtasks can then be further divided into smaller subtasks, as long as each

task or subtask is independent of all others. “ There are a number of benefits from using modular design.

• Ease of understanding: each module should perform just one function. • Reusable code: modules used in one program can also be used in other programs. • Elimination of redundancy: using modules can help to avoid the repetition of writing out the same segment of code more than once. Efficiency of maintenance: each module should be self-contained and have little or no effect on other modules within the program” (Robertson, 2007). For the “ ease of understanding, each module should perform one function” (Robertson, 2007). An example is when a computer prompts for the number of drinks from the coffee bar, sorts them by type of drink, and then displays a chart to a screen or output. This task may repeat itself throughout the morning rush hour and can be created as a subtask.

For “ reusable code, the modules used in one program can also be used in other programs” (Robertson, 2007). Once a module has completed testing and is essentially error-free, it can be used in other programs instead of rewriting or creating new code. For an example, a second coffee bar may use the code for afternoon business hours. For “ elimination of redundancy, using modules can help to avoid the repetition of writing out the same segment” (Robertson, 2007). Using a module shortens the length of code when many portions may actually be repetitious. For example, the library may need to monitor how many books onscienceare checked out during the month of the science fair. Naturally, other books would be checked out during that same time and those need to be monitored as well.

Instead of writing the whole code to ask if a book was on science or something else, a module would shorten the code. For “ efficiency of maintenance, each module should be self-contained and have little or no effect on other modules within the program” (Robertson, 2007). The modules should only perform one task and not be dependent on another module. If a module is modified or removed, it will not have a potentially damaging effect on other modules in the software. Programming can range from something simple like maintaining a log of exercise hours completed during the week or as complex as operating the space shuttle where lives are dependent upon perfect code. These are some of the steps taken to ease the process.

Reference Robertson, L.

A. (2007). Simple Program Design. A Step by Step Approach, Fifth Edition. cGraw-Hill.. Retrieved from University of Phoenix eBook Collection database.