1.4. What Is Programming?

Programming is the process of taking an algorithm and encoding it into a notation, a programming language, so that it can be executed by a computer. Although many programming languages and many different types of computers exist, the important first step is the need to have the solution. Without an algorithm there can be no program.

Computer science is not the study of programming. Programming, however, is an important part of what a computer scientist does. Programming is often the way that we create a representation for our solutions. Therefore, this language representation and the process of creating it becomes a fundamental part of the discipline.

Algorithms describe the solution to a problem in terms of the data needed to represent the problem instance and the set of steps necessary to produce the intended result. Programming languages must provide a notational way to represent both the process and the data. To this end, languages provide control constructs and data types.

Control constructs allow algorithmic steps to be represented in a convenient yet unambiguous way. At a minimum, algorithms require constructs that perform sequential processing, selection for decision-making, and iteration for repetitive control. As long as the language provides these basic statements, it can be used for algorithm representation.

All data items in the computer are represented as strings of binary digits. In order to give these strings meaning, we need to have **data types**. Data types provide an interpretation for this binary data so that we can think about the data in terms that make sense with respect to the problem being solved. These low-level, built-in data types (sometimes called the primitive data types) provide the building blocks for algorithm development.

For example, most programming languages provide a data type for integers. Strings of binary digits in the computer's memory can be interpreted as integers and given the typical meanings that we commonly associate with integers (e.g. 23, 654, and -19). In addition, a data type also provides a description of the operations that the data items can participate in. With integers, operations such as addition, subtraction, and multiplication are common. We have come to expect that numeric types of data can participate in these arithmetic operations.

The difficulty that often arises for us is the fact that problems and their solutions are very complex. These simple, language-provided constructs and data types, although certainly sufficient to represent complex solutions, are typically at a disadvantage as we work through the problem-solving process. We need ways to control this complexity and assist with the creation of solutions.

You have attempted 1 of 1 activities on this page

(WhatIsComputerScience.html)
(WhatIsComputerScience.html)

Next Section - 1.5. Why Study Data Structures and Abstract Data Types?
(WhyStudyDataStructuresandAbstractDataTypes.html)

(WhyStudyDataStructuresandAbstractDataTypes.html)

✔ **Completed. Well Done!**

© Copyright 2014 Brad Miller, David Ranum. Created using Runestone (http://runestoneinteractive.org/) 4.0.0.

(WhatIsComputerScience.html)

Next Section - 1.5. Why Study Data Structures and Abstract Data Types? (WhyStudyDataStructuresandAbstractDataTypes.html)