WebLab

Module 1.5 – Introduction to Programming

Module Summary

This module introduces students to the basic concepts of writing programs to solve algorithms using computers. We will cover flowcharts, pseudocode, variables and data types, operators, conditional statements, loops, functions and arrays.

Expectations

At the end of this module students will be able to:

1. Decompose complex problems into a series of small steps
2. Illustrate algorithms using flowcharts
3. Translate flowcharts into programs
4. Properly use variables, operators and keywords which make up basic computer code
5. Read and understand the Javascript programming language

Submodules

Thinking Like A Programmer – "Nowhere am I so desperately needed as among a shipload of illogical humans" - Spock

This section will focus on deconstructing a problem into its constituent elements. Talking to a computer is like talking to a three-year-old. You need to use short words, be very precise in what you say and keep your instructions simple. Computers actually have very little built-in ability. As programmers, we need to learn to work within the limitations of what the computer can handle.

Flowcharts – a developer's tool

A flowchart is a visual mechanism that developers can use when solving logic problems. It illustrates the flow of the code and makes it easy to pick out the different steps which need to be programmed into the solution.

Pseudocode – an agnostic language for communicating with other developers

Psuedocode is another tool used by developers during the process of creating algorithms. Writing psuedocode is a preliminary step in coding. While psuedocode is not readable by computers directly, it is a great tool for communicating with other programmers and is easily converted into any programming or scripting language.

Variables – say hello to my new piece of data named "George"

The data used by programs is stored within the computer using variables. A variable is a named memory location where data can be stored and retrieved within a program simply by referring to the assigned name.

Comparison Operators – evaluating differences

Comparison operators are used to evaluate differences and similarities between two pieces of data. You can see if two data elements are the same, different, larger or smaller.

Conditional Statements – making decisions

Computers are excellent at evaluating and comparing data and then making decisions based on the result. The process of making these decisions is called a conditional statement. In programming this is represented by the keywords **if** and **else** and is sometimes referred to as an if-then-else statement.

Loops – again, and again, and again

Oftentimes instructions need to be repeated more than once, such as when processing a set of data. Programmers refer to these structures as loops. The three basic looping structures are **while**, **do**..**while** and **for** loops.

Simple Data Types – are you my type?

In our world, data comes in many different forms. We are used to dealing with names, phone numbers, addresses, images, and many other forms of data. Within a computer, data is stored and processed according to its type. Numbers are handled one way and names another. We will look at the various ways that data is stored and handled within a program.

Operators – crunching the numbers

Operators are the symbols used to refer to the functions that computers use to manipulate data. When we all studied math in school we were introduced to the mathematical operators for addition (+) and subtraction (-). This section will introduce all of the operators that are used by Javascript to manipulate the various data types.

Functions – better living through code partitioning

Programmers don't like to have to write the same code more than once. However, many programming solutions require similar pieces of code that have been written for other programs. For instance, when handling input data from a user, it is usually necessary to validate the input to make sure that the user has input what was expected. When writing code, it is a good idea to capture these reusable pieces into separate blocks of code called functions. Good coding practice calls for the use of functions to separate out any section of code that can represent a self-contained solution to a problem. The two main purposes of creating functions are for reusability and abstraction.

Arrays – containers for storing groups of related data

Many problem solutions process lots of related information. For example, the roommate algorithm that you looked at in module one needed to process lots of data from many individual students. Rather than having a different variable name for each individual student, we can group together related data into a storage device known as an array which give us a convenient way to refer to the data and access individual pieces of it.

Objects – like arrays, but different

We all know what objects are because we are surrounded by them. In computer science, the term object refers to a particular structure that contains both data and behavior. Behavior refers to the methods used to interact with the data. Consider a book for example. A book has data (all the information that is printed on its pages) and it has behavior. A book's behavior might be described using methods such as “open”, “turn to page”, “search index for a term”, “close”, etc.