1.11 Introduction summary

This chapter's key points included:

* A program consists of instructions (aka statements) that execute one at a time, to get input, process data, and put output.
* A program uses variables to hold data, which may change ("vary").
* A flowchart depicts a program graphically, with a node for each statement.
* A program can output a variable's value, or a string literal consisting of characters (including a newline character).
* A program can contain comments, which are for humans only and ignored when the program runs.
* Most whitespace (regular spaces or newlines) is ignored by a program, but good practice uses whitespace in a consistent way.
* The information age is quite new in human history. Changes are rapid.
* Computers surround us and computer numbers and usage continue to grow.
* Inside a computer, all data (characters, numbers, and more) is represented as bits: 0's and 1's.
* Programming is largely about problem solving, namely creating a methodical solution to a given task.
* Careers in computing are numerous, highly-rated, and growing. Non-computing jobs may benefit from programming.
* Pseudocode is an informal textual representation of a program intended for easy human understanding.

2.16 Variables/Assignments Summary

This chapter's key points included:

* A variable declaration declares a new variable, specifying the variable's name and type.
* An assignment statement assigns the variable on the left-side of the = with the current value of the right-side expression.
* An expression is a combination of items, like variables, literals, operators, and parentheses, that evaluates to a value.
* A name created by a programmer for an item like a variable or function is called an identifier, which must follow certain rules to be valid. Programmers typically follow identifier naming conventions that are defined by their company, team, teacher, etc.
* An expression is evaluated using precedence rules that follow the evaluation order of standard mathematics.
* Incremental development is the process of writing and and testing a small amount of code, then writing and testing a small amount more (an incremental amount), and so on.
* A variable declared as type float stores a floating-point number, which is a real number, like 98.6, 0.0001, or -666.667.
* A programmer should choose a variable's type based on the type of value held. Integer variables are typically used for values that are counted. Floating-point variables are typically used for values that are measured or when dealing with fractions of countable items, such as the average number of cars per household.
* Programming languages typically have built-in functions to perform common operations needed by programmers, such as performing mathematical operations like square root or raising a number to a power.
* A function is a list of statements executed by invoking the function's name, with such invoking known as a function call.
* Programming languages typically have built-in functions for generating random numbers. The integers generated by a random number generator are known as pseudo-random. "Pseudo" means "not actually, but having the appearance of". Internally, the RandomNumber() function has an equation to compute the next "random" integer from the previous one.
* When the operands of / are both integers, the operator performs integer division, which does not generate any fraction.
* For integer division, the second operand of / or % must never be 0, because division by 0 is mathematically undefined. A divide-by-zero error occurs at runtime if a divisor is 0, causing a program to terminate.
* A type conversion is a conversion of one data type to another, such as an integer to a float.
* zyFlowchart, and other programming languages, automatically performs several common conversions between integer and float types, and such automatic conversion is known as implicit conversion.
* If a programmer needs to explicitly convert an item's type, the programmer can use a type cast to converts value of one type to another type.
* The modulo operator (%) evaluates to the remainder of the division of two integer operands.
* A constant is a named value item that holds a value that cannot change. Constants are commonly used in programs to hold the value of mathematical constants or a value that should not change during the program's execution.

3.11 Branches summary

This chapter's key points included:

* In a flowchart, a decision creates two branches, one for when the decision's expression is true (the if branch), another when false (the else branch).
* If-else branches have statements in each branch. An if branch has no statements in the else branch.
* If-elseif branches have cascaded decisions along the false branches. Only one true branch can execute.
* A branch can itself have a decision, known as nested branches.
* Multiple if branches can be created, which are independent, so more than one true branch can execute.
* Valid equality and relational operators are ==, !=, <, <=, >, >=.
* If-elseif branches are commonly used to detect ranges, with the lower end of the range implicit.
* Logical operators are: and, or, not.
* In an expression, operators are evaluated in a specific order based on precedence rules (just like in math).
* Because floating-point numbers aren't represented exactly, they shouldn't be compared for equality (using ==). Instead, they can be compared for "close enough".
* Branches in pseudocode use words like if, else, and elseif, and a branch's statements appear indented starting on a next line.