Terms used in Structured Programming in C++. (Definitions may be different in other languages.)
Some of these definitions are specific to C++, for example, the definition of 'variable' changes in other languages.

An **algorithm** is a finite sequence of discrete, well-defined steps that will always give a solution to a problem. Alternately, you could say that an algorithm is a step-by-step procedure to solve a problem in a finite amount of time.

A **program** is an algorithm translated into a form that a computer can execute.

A **variable** is a named location in RAM that holds a value which can change as a program executes.

A **data type** is a specification of the domain and operations of a variable of its type.

The **domain** of a data type is the set of legal values for a variable of that data type.

The **operations** of a data type is the set of legal actions on a variable of that data type.

A **compiler** is a program that translates human-readable source code into machine language object code. This produces an **object file** containing the machine language instructions for the program, which must then be linked (by the **linker**) to the libraries (e.g., iostream) to produce an executable program.

A **stream** is a one-directional flow of character data going either into or out of a running program (a **process**).

Type Casting is the explicit change of data type during expression evaluation.

Type Coercion is the implicit change of data type during expression evaluation.

An **L-value** is an expression that can legally be placed the left side of an assignment = operator. It evaluates to a place in memory (an address) whose contents can be modified.

An **R-value** is an expression that can be legally placed on the right side of an assignment = operator. It always evaluates to some type of value (although that value may be an address).

A **Control Structure** is a programming construct that determines the sequence of execution of statements in a program.

Sequential Execution is the fundamental control structure: e.g., statement are executed in the sequence they are written unless another control structure (or flow modifier like break or continue) changes that sequence.

A **Structured Program** is a program written using the built-in control structures of a language, without goto. It sometimes referred to as a "goto-less" program.

Computer Science is the study of how to solve problems by use of machines.

A **Statement Block** is either a single statement or a compound statement.

A **Compound Statement** is a group of zero or more statements between curly braces {}.

A **string** is a group of characters treated as a unit. C++ supports two types of strings, C-strings and C++ strings.

A **C-string** is an array of char with its data terminated by a NUL byte (ASCII code zero).

A C++ string is a class type (the **string** class). It contains data you cannot directly access except through the string class' public interface, a set of class member functions (methods) that let you access or modify the data. C++ strings are not C-strings. The two types of strings in C++ are not the same and not compatible. Don't worry about what a class is too much for now, you'll see this in CSCI-15.

Whitespace characters are characters that do not light up pixels on the screen (Or make them dark if against a white background). Some whitespace characters are space, tab, newline and carriage return.

A **Counting Loop** is a loop that has the property that you can determine in advance how many times it will iterate (execute its body).

An **Event Loop** is a loop in which you can't know how many times it will iterate by examining the code in advance.

Abstraction is the separation of the essential logical properties of an object or action from the implementation details of that object or action. Informally, you could say *defer the details*.

In mathematics, a **function** is a set of points called the domain; a set of points called the range; and a set of rules for transcribing points in the domain to points in the range. This definition is pretty useless in introductory computer science classes, although it matters later on.

A better definition of a **function** in introductory computer science courses is that a function is a group of statements that performs a single task, all of that task, and nothing but that task.

An **argument** is an expression, the value of which is copied into its corresponding parameter at the time the function is invoked.

A **parameter** is a variable, local to its function, which is initialized to the value of its corresponding argument at the time the function is invoked.

A **file** is a named collection of related data stored on a secondary storage device.

An **array** is a homogeneous collection of data elements guaranteed to be contiguous in RAM.

A **struct** is a heterogeneous collection of data fields not guaranteed to be contiguous in RAM.

A **sentinel** is a value that marks the end of input data. It is usually not considered part of the input data itself.

A **pointer** is a variable that contains an address.

A **macro** is a fragment of code which has been given a name. Whenever the name is used, it is replaced by the contents of the macro. (You could think of a macro as a type of substitution cypher: in the compiler, the macro is replaced by its expansion before code generation occurs.)