#include <iostream> - input/output (console usage) | #include <iomanip> - manipulation | #include<string> - can use string varaibles | #include <cmath> - allows mathematical library functions | #include <fstream> - file operaions (file input/output)

**Chapter 1:** -- **CPU – Central Processing Unit** – Part of the computer that runs programs. **Main Memory** (aka Random Access Memory, or RAM)– Where computer stores a program while the program is running. **Secondary Storage** – Holds date for long period of time. **Input Devices** – Device that collects info and sends it to the computer. **Output Devices** – computer sends info to output device, which formats and presents it. **System software** (operating systems, utility programs, software development tools) – Programs that control and manage the basic operations of a computer. **Application software** – Programs that make a computer useful for everyday tasks**. Program** – A set of instructions a computer follows in order to perform a task. **Programming language** – special language used to write computer programs. **Integrated Development Environments (IDEs)** – IDEs consist of a text editor, compiler, debugger, and other utilities integrated into a package with a single set of menus. **Key Words** – reserved and cannot be used for anything other than their designated purposes. Always lower case. (ex. using, namespace, int, double). **Programmer Defined Identifiers** – Not part of C++ language, but rather are names made up by the programmer. (variables are examples of these kind of words) **Operators** – Perform operations on pieces of data known as operands. (ex. + - / \* ==). **Line** – a single line as it appears on the program. **Statement** – a complete instruction that causes the computer to perform some action. **Variable** – a names storage location in the computer’s memory for holding a piece of information.

**Chapter 2** -- #include directive – causes the contents of another file to be inserted into the program.

**Variable definition** – tells the compiler the variable’s name and the type of data it will hold. (int number; int is the type of date (integer) and number is the variable’s name. **Initializing a variable** – assigning a value to it for the first time. **Literal** – piece of data that is written directly into a program’s code. (i.e. number = 100) **Char data type** – used to store individual characters. Can only hold one character at a time.

#include <string> - string class, needed to use string type variables (string movieTitle;)

**Boolean variables** – set to either true or false.

**Chapter** 3 - **Expression** – a programming statement that has a value. Usually consists of an operator and operands. (sum = 21 + 3)

**Overflow** – When a variable is assigned a value that is too large in range for that variable’s data type.

**Underflow** - When a variable is assigned a value that is too small in range for that variable’s data type.

**Type casting** – allows you to perform manual data type conversion. (val = static\_cast<int>(number);

**Multiple assignments** – to assign the same value to several variables with one statement (a = b = c = 5)

**Combined assignment operator** – number = number + 1;

**Formatting** – the way a value is printed

**Setprecision Manipulator** – floating point values may be rounded to a number of significant digits

**Showpoint manipulator** – will add on zeros to data types that can support them but don’t have them in the value (i.e. double x = 440 can be made to show 440.00 with showpoint. << showpoint <<

**Left and Right manipulators** - << left << << right <<

**getline function** – reads an entire line, including leading and embedded spaces, and stores it in a string object

**Chapter** 4 - Relational **Operators** – relational operators allow you to compare numeric and char values and determine whether one is greater than, less than, equal to, or not equal to another.

**If statement** – can cause other statements to execute only under certain conditions.

**Sequence structure** – the statements are executed in sequence, without branching in another direction.

**Decision structure** – executes some statements only under certain circumstances (if statement). S specific action is taken only when specific condition exists.

**If/else statement** – execute one group of statements if expression true, another group if false.

**Nested if statements** – to test more than one condition, an if statement can be nested inside another.

**If/else if statement** – tests a series of conditions. (if, else if, else if, else). Simpler than nested if/else.

**Flags** – A flag is a Boolean or integer variable that signals when a condition exists.

**Logical operators** – logical operators connect two or more relational expressions into one or reverse the logic of an expression. && (and), || (or), ! (not)

**Input validation** – process of inspecting data given to a program by the user and determining if valid.

**Conditional operator** – works list if/else statement – x<0 ? y=10: z=20 (if x<0 then y=10, else z = 20) Switch statement – lets the value of a variable or an expression determine where the program branches

**Variable scope** – the scope of a variable is limited to the block in which it is defined.

**Variables with same name** – When a block is nested inside another block, a variable defined in the inner block may have the same name as a variable defined in the outer block. As long as the variable in the inner block is visible, however, the variable in the outer block will be hidden.

**Chapter** 5 - Increment **and decrement operators** – (++ and --) Increase or decrease by one.

**Loop** – control statement that causes a statement or group of statements to repeat.

**While loop** – expression is tested, if true, the statement is executed. Expression tested. (pretest loop)

**Pretest loop** – tests its expression before each iteration.

**Counter** – a variable that is regularly incremented or decremented each time a loop reiterates.

**Do-While loop** – posttest loop, always performs at least one iteration.

**Posttest loop** – the loop’s expression is tested after each iteration.

**For loop** – ideal for performing a known number of iterations. Pretest loop.

**Conditional loop** – executes as long as a particular condition exists.

**Count-controlled loop** – repeats a specific number of times.

**Sentinel** – a special value that marks the end of a list of values.

**File** – when a program needs to save data for later use, it writes the data in a file. Data can then be used at a later time.

**Text file** – contains data that has been encoded as text, using a scheme such as ASCII or Unicode.

**Binary file** – contains data that has not been converted to text. You cannot view the contents of a binary file with a text editor.

**File access methods** – sequential access and direct access. Sequential access files you access data from the beginning of a file to the end of a file. Have to read all data that comes before the data you want to read. Random access file (aka direct access file) you can jump directly to any piece of data in the file without reading the data that comes before it.

**File name and file name extension**. Johnson.txt – Johnson would be file name and .txt would be the extension.

Ifstream inputFile; (next line) inputFile.open(“Customers.txt”); this opens a file for input. | inputFile.close(); closes a file

outputFile << “I love C++ programming\n”; writes data to a file. | inputeFile >> name; reads data from a file.

**Break statement in nested loop** – only interrupts the loop in which it is placed. (break statement causes loop to terminate early)

**Continue statement** – causes the current iteration of the loop to end immediately. Stops current iteration and begins the next.

**Chapter** 6 - Function **call** – a statement that causes a function to execute. A function definition contains the statements that make up the function.

**Function prototype** – eliminates the need to place a function definition before all calls to the function.

**Passing data by value** – When an argument is passed into a parameter, only a copy of the argument’s value is passed. Changes to the parameter do not affect the original statement.

**Return statement** – the return statement causes a function to end immediately.

**Returning a value from a function** – a function may send a value back to the part of the program that called the function.

**Local and Global variables** – a local variable is defined inside a function and is not accessible outside the function. A global variable is defined outside all functions and is accessible to all functions in its scope.

**Reference variables** – when used as parameters they allow a function to access the parameter’s original argument. Changes to the parameter are also made to the argument.

**Stub** – a dummy function that is called instead of the actual function it represents.

**Driver** – a program that tests a function simply by calling it. If the function accepts arguments, the driver passes test data.