Chapter 1 : Getting Started (NOTES)

argument : Value passed to a function.
Assignment : Obliterates an object's current value and replaces that value by a new one.
block : Sequence of zero or more statements enclosed in curly braces.
Buffer : A region of storage used to hold data. IO facilities often store input (or output) in a buffer and read or write the buffer independently from actions in the program. Output buffers can be explicitly flushed to force the buffer to be written. By default, reading cin flushes cout; cout is also flushed when the program ends normally.
endl, which is a special value called a manipulator. Writing endl has the effect of ending the current line and flushing the buffer associated with that device. Flushing the buffer ensures that all the output the program has generated so far is actually written to the output stream, rather than sitting in memory waiting to be written.
built-in type : Type, such as int, defined by the language.
A **type** defines both the contents of a data element and the operations that are possible on those data. The data our programs manipulate are stored in variables and every variable has a type. When the type of a variable named v is T, we often say that "v has type T" or, interchangeably, that "v is a T."
cerr : ostream object tied to the standard error, which often writes to the same device as the standard output. By default, writes to cerr are not buffered. Usually used for error messages or other output that is not part of the normal logic of the program.
string literal : Sequence of zero or more characters enclosed in double quotes ("a string literal").
character string literal : Another term for string literal.
cin : istream object used to read from the standard input.
Class : Facility for defining our own data structures together with associated operations. The class is one of the most fundamental features in C++. Library types, such as istream and ostream, are classes.
class type : A type defined by a class. The name of the type is the class name.
clog : ostream object tied to the standard error. By default, writes to clog are buffered. Usually used to report information about program execution to a log file.
comments : Program text that is ignored by the compiler. C++ has two kinds of comments: single-line and paired. Single-line comments start with a //. Everything from the // to the end of the line is a comment. Paired comments begin with a /* and include all text up to the next */.
Condition : An expression that is evaluated as true or false. A value of zero is false; any other value yields true.
cout : ostream object used to write to the standard output. Ordinarily used to write the output of a program.
curly brace : Curly braces delimit blocks. An open curly ({) starts a block; a close curly (}) ends one.
data structure : A logical grouping of data and operations on that data.
edit-compile-debug : The process of getting a program to execute properly.
end-of-file : System-specific marker that indicates that there is no more input in a file.
Expression : The smallest unit of computation. An expression consists of one or more operands and usually one or more operators. Expressions are evaluated to produce a result. For example, assuming i and j are ints, then i + j is an expression and yields the sum of the two int values.
for statement : Iteration statement that provides iterative execution. Often used to repeat a calculation a fixed number of times.
Function : Named unit of computation. A function definition has four elements: a return type, a function name, a (possibly empty) parameter list enclosed in parentheses, and a function body.
function body : Block that defines the actions performed by a function.
function name : Name by which a function is known and can be called.
Header : Mechanism whereby the definitions of a class or other names are made available to multiple programs. A program uses a header through a #include directive.

if statement : Conditional execution based on the value of a specified condition. If the condition is true, the if body is executed. If not, the else body is executed if there is one.
Initialize : Give an object a value at the same time that it is created.
iostream : Header that provides the library types for stream-oriented input and output.
istream : Library type providing stream-oriented input.
ostream : Library type providing stream-oriented output.
library type: Type, such as istream, defined by the standard library.
main : Function called by the operating system to execute a C++ program. Each program must have one and only one function named main.
manipulator : Object, such as std::endl, that when read or written "manipulates" the stream itself.
member function : Operation defined by a class. Member functions ordinarily are called to operate on a specific object
method : Synonym for member function.
namespace : Mechanism for putting names defined by a library into a single place. Namespaces help avoid inadvertent name clashes. The names defined by the C++ library are in the namespace std.
std : Name of the namespace used by the standard library. std::cout indicates that we're using the name cout defined in the std namespace.
parameter list : Part of the definition of a function. Possibly empty list that specifies what arguments can be used to call the function.
return type : Type of the value returned by a function.
source file :Term used to describe a file that contains a C++ program. On most systems, the name of a source file ends with a suffix, which is a period followed by one or more characters. The suffix tells the system that the file is a C++ program. Different compilers use different suffix conventions; the most common include .cc, .cxx, .cpp, .cp, and .C.
standard error : Output stream used for error reporting. Ordinarily, the standard output and the standard error are tied to the window in which the program is executed.
standard input: Input stream usually associated with the window in which the program executes.
standard output : Output stream usually associated with the window in which the program executes.
standard library : Collection of types and functions that every C++ compiler must support. The library provides the types that support IO. C++ programmers tend to talk about "the library," meaning the entire standard library. They also tend to refer to particular parts of the library by referring to a library type, such as the "iostream library," meaning the part of the standard library that defines the IO classes.
statement : A part of a program that specifies an action to take place when the program is executed. An expression followed by a semicolon is a statement; other kinds of statements include blocks and if, for, and while statements, all of which contain other statements within themselves.
uninitialized variable: Variable that is not given an initial value. Variables of class type for which no initial value is specified are initialized as specified by the class definition. Variables of built-in type defined inside a function are uninitialized unless explicitly initialized. It is an error to try to use the value of an uninitialized variable. Uninitialized variables are a rich source of bugs.
variable :A named object.
while statement : Iteration statement that provides iterative execution so long as a specified condition is true. The body is executed zero or more times, depending on the truth value of the condition.

LIST OF OPERATORS

( ) Operator	Call Operator	A pair of parentheses "()" following a function name. The operator causes a function to be invoked. Arguments to the function may be passed inside the parentheses.
++ Operator	Increment Operator	Adds 1 to the operand; ++i is equivalent to i = i + 1.

+= Operator	Compound Assignment Operator	Compound assignment operator that adds the right-hand operand to the left and stores the result in the left-hand operand; a += b is equivalent to a = a + b.
. Operator	Dot Operator	Left-hand operand must be an object of class type and the right-hand operand must be the name of a member of that object. The operator yields the named member of the given object.
:: Operator	Scope Operator	Among other uses, the scope operator is used to access names in a namespace. For example, std::cout denotes the name cout from the namespace std.
= Operator	Assignment Operator	Assigns the value of the right-hand operand to the object denoted by the left-hand operand
-- Operator	Decrement operator	Subtracts 1 from the operand; --i is equivalent to i = i - 1.
<< Operator	Output operator	Writes the right-hand operand to the output stream indicated by the left-hand operand: cout << "hi" writes hi to the standard output. Output operations can be chained together: cout << "hi" << "bye" writes hibye.
>> Operator	Input operator	Reads from the input stream specified by the left- hand operand into the right-hand operand: cin >> i reads the next value on the standard input into i. Input operations can be chained together: cin >> i >> j reads first into i and then into j.
== Operator	equality operator	Tests whether the left-hand operand is equal to the right-hand operand
!= Operator	inequality operator	Tests whether the left-hand operand is not equal to the right-hand operand
<= Operator	less-than-or-equal operator	Tests whether the left-hand operand is less than or equal to the right-hand operand
< operator	less-than operator	Tests whether the left-hand operand is less than the right-hand operand
>= operator	Greater-than-or-equal operator	Tests whether the left-hand operand is greater than or equal to the right-hand operand.
> operator	Greater-than operator	Tests whether the left-hand operand is greater than the right-hand operand.

#include : Directive that makes code in a header available to a program

Standard Input and Output Objects :
The library defines four IO objects. To handle input, we use an object of type istream named cin (pronounced see-in). This object is also referred to as the standard input. For output, we use an ostream object named cout (pronounced see-out). This object is also known as the standard output. The library also defines two other ostream objects, named cerr and clog (pronounced see-err and see-log, respectively). We typically use cerr, referred to as the standard error, for warning and error messages and clog for general information about the execution of the program. Ordinarily, the system associates each of these objects with the window in which the program is executed. So, when we read from cin, data are read from the window in which the program is executing, and when we write to cout, cerr, or clog, the output is written to the same window.

while

```
while (condition) {
    //statements if condition is TRUE
```

```
        }
```
for
```
        for (initialise ; condition ; expression ) {
                //statements if condition is True
                //once run it will run expression then re check condition
                // repeat
        }
```
if
```
        if (condition) {
        // condition TRUE run this block
        } else {
        // condition FALSE run this block.
        //not required to write this block (it will not run FALSE block and continue).
        }
```
std::min : returns minimum value
```
        std::min (data1, data2, …);
```
std::max : returns maximum value
```
        std::max(data1,data2,…);
```