C++ Command List

Compiling

To compile a C++-program, you can use either g++ or c++.

g++ -o executable filename.out sourcefilename.cc

c++ -o executable filename.out sourcefilename.cc

e.g. g++ -o C++sample inout.out C++sample inout.cc

For the following commands you can find at the end of this summary sample programs.
Each command in C++ is followed by ";" . Carriage return has no meaning in C++.

Comments

Essential for the writing of clear programs are comments, which are explanations for your program, and which are ignored by the compiler.

/* . . . */ puts the text . . . into comment (more than one line possible)

// . . . puts text . . . for rest of same line into comment

Data Types

Data Type	Declaration (Example)	Assignment (Example)
Integer	short i1, i2; int i1, i2; long i1, i2; unsigned i1, i2; unsigned long i1, i2;	i1 = 3;

Real	float = f1, f2;
	double = f1, f2;
	long double = f1, f2;
		float n = 3.14E-10;
		double n = 4.201 or 4.2E-2;
		long double n = 3.65E-4932;

Single Character (LETTER)	char c1, c2;	c1 = 'R';
String of Character (WORD)	string s1, s2;	s1 = "Word";
Logical (Boolean)[True/False]	bool b1, b2;	b1 = true; b2 = false;

Input And Output

Input/Output With Screen:

To be able to use the following commands you need to write

#include <iostream>

using namespace std; at the beginning of your program:

output:

cout << "string of characters";
cout << variable; << endl;

input:

cin >> variable;

Input/Output With Files:

To be able to use the following commands you need to write

#include <fstream> ----------at the beginning of your program:

output:

ofstream outfilevariable ("outputfilename",ios::out);

outfilevariable << . . . ----------will write into file with name outputfilename

input:

ifstream infilevariable ("inputfilename",ios::in);

infilevariable >> . . . ----------will read from file with name outputfilename

Arithmetic Calculations

Operations: + − ∗ /

Functions:

To be able to use all of the following functions you need to to write at the beginning of your program:

#include <cmath>

#include <cstdlib>

Function	Example
pow(x,y)	x^y
sin(x)	
cos(x)	
tan(x)	
asin(x)	sin−1 (x) in range [−π/2, π/2]
acos(x)	cos−1 (x) in range [0, π]
atan(x)	tan−1 (x) in range [−π/2, π/2]
sinh(x)	
cosh(x)	
tanh(x)	
exp(x)	e^x
log(x)	ln(x)
sqrt(x)	√ x
fabs(x)	|x|
floor(x)	Largest integer not greater than x; EXAMPLE: floor(5.768) = 5
ceil(x)	Smallest integer not less than x; EXAMPLE: ceil(5.768) = 6
fmod(x,y)	floating-point remaind of x/y , with the same sign as x
x % y	remainder of x/y, x & y are ints. EXAMPLE: 7%5 = 2

Decision Statements

Comparison Operators:

Operator	Equivalent	Example
==	=	i1 == i2; ----------1 == 1
!=	Not =	i1 != i2; ----------1 != 2
>	>	i1 > i2; -------- 2 > 1;
<	<	i1 < i2 ---------- 1 < 2;
>=	≥	i1 >= i2; ---------- 3 ≥ 2;
<=	≤	i1 <= i2; ---------- 2 ≤ 2;
&&	and	(i1 <= i3)&& (i1 != i4)
||	and/or	(i1 <= i3) || (i1 != i4)

Statements: IF, IF/ELSE, IF/ELSE IF/ELSE, SWITCH/CASE, SWITCH/CASE/DEFAULT

IF

```
if( condition ) {
statements
}
```

IF / ELSE

```
if( condition ) {
statements
}
else {
statements
}
```

IF / ELSE IF / ELSE

```
if( condition ) {
statements
}
else if {statements}
else {statements}
```

SWITCH / CASE

```
switch ( casevariable ) {
case value1a:
case value1b:
{statements}
break;
```

SWITCH / CASE / DEFAULT

```
switch ( casevariable ) {
case value2a:
case value2b:
{statements}
break;
default:
{statements}
}
```

Loops / Repetitions: (WHILE, FOR, DO / WHILE)

WHILE

```
while (conditions) {
statements
}
```

FOR

```
for ( init; conditions; update ){
statements
}
```

DO / WHILE

```
do {
statements these statements are done before the while statement check
} while ( condition) ;
```

Functions: FunctionType FunctionName(Parameters)

A function is a set of commands. It is useful to write a user-defined function, i.e. your own function, whenever you need to do the same task many times in the program. All programs start execution at the function **main**. For Functions you need steps 1-3:

Step 1: At the beginning of your program you need to declare any functions: ***function type function name (types of parameter list);***

example 1: double feetinchtometer (int,double);
example 2: void metertofeetinch (double, int&, double&);

The function type declares which variable is passed back from the function (**void** means none via the function type). The variables without "**&**" are all input parameters, i.e. only passed to the function and are not changed within the function. The variables with "**&**" may be passed both to and from the function and may be changed in the function.

Step 2: In the program you use the function with:
function_name (parameter_list);

example 1: feetinchtometer(5.0,3.2);
example 2: metertofeetinch(1.3,feet,inch);

Step 3: After main { . . . } define your function:
function_type function_name (parameter types and names) { declarations and statements }

example 1: double feetinchtometer(int feet, double inch){ . . .};
example 2: void metertofeetinch (double m, int& feet, double& inch){. . .};

Arrays

Data Type	Declaration (Example)	Assignment (Example)

Array	int street[100];
	double matrix[7];
	etc.
		street[0] = 1; street[51] = 49;
		matrix[6] = 3.2;
		etc.

Multidimensional Array	int lattice[10] [10];
	double wateruse[5] [3] [2];
	etc.
		lattice[0] [1] = 1;
		wateruse[3] [1] [0] = 1.2;
		etc.

Escape Sequences pg.22

Using **character constants** such as a single character or a character string in a program, certain characters are problematic.
Escape sequence is an indirect way of specifying a Special character, and it always begins with a backslash.

Escape Sequence	Control Character
PUT ESCAPE SEQUENCES IN THIS COMMAND `std::cout << "" << std::endl;`	
\n	New Line
\t	New Tab
\v	Vertical Tab
\b	Backspace
\r	Carriage Return
\f	Form Feed
\a	Alert/Bell
PROBLEM CHARACTER	
\\	Backslash
\'	Single Quote
\"	Double Quote

EXAMPLE

```
std::cout << "\"Least \'said\' \\\n\t\tsoonest \'mended\'.\"" << std::endl;
```