Summary of basic C++-commands

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;`
		`f1 = 3.2; f2 = 2.1E-3;`

single character string of characters	`char c1,c2;` `string s1,s2;`	`c1 = 'R'` `s1 = "Farmer"`
logical	`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: $+ \quad - \quad * \quad /$

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>
```

`pow(x,y)`	x^y
`sin(x)`	
`cos(x)`	
`tan(x)`	
`asin(x)`	$\sin^{-1}(x)$ in range $[-\pi/2, \pi/2]$
`acos(x)`	$\cos^{-1}(x)$ in range $[0, \pi]$
`atan(x)`	$\tan^{-1}(x)$ in range $[-\pi/2, \pi/2]$
`sinh(x)`	
`cosh(x)`	
`tanh(x)`	
`exp(x)`	e^x
`log(x)`	$\ln(x)$
`sqrt(x)`	$\sqrt{x}$
`fabs(x)`	$\lvert x \rvert$
`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 remainder of x/y with the same sign as x
`x % y`	remainder of x/y, both x and y integers

Decision Statements

Comparison Operators:

`==`	$=$	example:	`i1 == i2`
`!=`	$\neq$		`i1 != i2`
`>`	$>$		`i1 > i2`
`<`	$<$		`i1 < i2`
`>=`	$\geq$		`i1 >= i2`
`<=`	$\leq$		`i1 <= i2`
`&&`	and		`(i1 != i2)&& (i1 == i3)`
`||`	and/or		`(i1 == i2) || (i1 == i3)`

Statements:

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

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

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

```
switch ( casevariable ) {
case value1a:
case value1b:
  {statements}
  break;
case value2a:
case value2b:
  {statements}
  break;
default:
  {statements}
}
```

Repetitions

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

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

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

functions

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 (actual_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]=0; street[50]=7;`
		`matrix[6]=3.2;`
		etc.

multidim. array	`int lattice[10][10];`
	`double wateruse[5][3][2];`
	etc.
		`lattice[0][1]=1;`
		`wateruse[3][1][0]=1.2;`
		etc.