C++ Quick Reference

Commonly Used C++ Data Types

Data Type	Description
`char`	Character
`unsigned char`	Unsigned Character
`int`	Integer
`short int`	Short integer
`short`	Same as `short int`
`unsigned short int`	Unsigned short integer
`unsigned short`	Same as `unsigned short int`
`unsigned int`	Unsigned integer
`unsigned`	Same as `unsigned int`
`long int`	Long integer
`long`	Same as `long int`
`unsigned long int`	Unsigned long integer
`unsigned long`	Same as `unsigned long int`
`float`	Single precision floating point
`double`	double precision floating point
`long double`	Long double precision floating point

Forms of the ***if*** *Statement*

Simple if / **Example**

```
if (expression)              if (x < y)
   statement;                   x++;
```

if/else / **Example**

```
if (expression)              if (x < y)
   statement;                   x++;
else                         else
   statement;                   x--;
```

if/else if / **Example**

```
if (expression)              if (x < y)
   statement;                   x++;
else if (expression)         else if (x < z)
   statement;                   x--;
else                         else
   statement;                   y++;
```

To conditionally-execute more than one statement, enclose the statements in braces:

Form / **Example**

```
if (expression)              if (x < y)
{                            {
   statement;                   x++;
   statement;                   cout << x;
}                            }
```

Web Sites

For the *Starting Out with C++* Companion Web Site
www.pearsonhighered.com/gaddis
For Addison-Wesley Computing
www.pearsonhighered.com/cs

Commonly Used Operators

Assignment Operators

`=`	Assignment
`+=`	Combined addition/assignment
`-=`	Combined subtraction/assignment
`*=`	Combined multiplication/assignment
`/=`	Combined division/assignment
`%=`	Combined modulus/assignment

Arithmetic Operators

`+`	Addition
`-`	Subtraction
`*`	Multiplication
`/`	Division
`%`	Modulus (remainder)

Relational Operators

`<`	Less than
`<=`	Less than or equal to
`>`	Greater than
`>=`	Greater than or equal to
`==`	Equal to
`!=`	Not equal to

Logical Operators

`&&`	AND
`||`	OR
`!`	NOT

Increment/Decrement

`++`	Increment
`--`	Decrement

Conditional Operator ?:

Form:

```
expression ? expression : expression
```

Example:

```
x = a < b ? a : b;
```

The statement above works like:

```
if (a < b)
   x = a;
else
   x = b;
```

The ***while*** *Loop*

Form: / **Example:**

```
while (expression)       while (x < 100)
   statement;               cout << x++ << endl;

while (expression)       while (x < 100)
{                        {
   statement;               cout << x << endl;
   statement;               x++;
}                        }
```

*The do-****while*** *Loop*

Form: / **Example:**

```
do                       do
   statement;               cout << x++ << endl;
while (expression);      while (x < 100);

do                       do
{                        {
   statement;               cout << x << endl;
   statement;               x++;
} while (expression);    } while (x < 100);
```

C++ Quick Reference (continued)

The `for` *Loop*

Form:

```
for (initialization; test; update)
   statement;

for (initialization; test; update)
{
   statement;
   statement;
}
```

Example:

```
for (count = 0; count < 10; count++)
   cout << count << endl;

for (count = 0; count < 10; count++)
{
   cout << "The value of count is ";
   cout << count << endl;
}
```

The `switch/case` *Construct*

Form:

```
switch (integer-expression)
{
   case integer-constant:
      statement(s);
      break;
   case integer-constant:
      statement(s);
      break;
   default :
      statement;
}
```

Example:

```
switch (choice)
{
   case 0 :
      cout << "You selected 0.\n";
      break;
   case 1 :
      cout << "You selected 1.\n";
      break;
   default :
      cout << "You did not select 0 or 1.\n";
}
```

Using `cout`

Requires `<iostream>` header file.

Commonly used stream manipulators

Name	Description
`endl`	advances output to the beginning of the next line.
`fixed`	sets fixed point notation
`left`	sets left justification
`right`	sets right justification
`setprecision`	sets the number of significant digits
`setw`	sets field width
`showpoint`	forces decimal point & trailing zeros to display

Example:

```
cout << setprecision(2) << fixed
     << left << x << endl;
```

Member functions for output formatting

Name	Description
`.precision`	sets the number of significant digits
`.setf`	sets one or more `ios` flags
`.unsetf`	clears one or more `ios` flags
`.width`	sets field width

Example:

```
cout.precision(2);
```

Using `cin`

Requires `<iostream>` header file

Commonly used stream manipulators

Name	Description
`setw`	sets field width

Member functions for specialized input

Name	Description
`.getline`	reads a line of input as a C-string
`.get`	reads a character
`.ignore`	ignores the last character entered
`.width`	sets field width

Some Commonly Used Library Functions

Name	Description
(The following require <cstdlib>)	
`atof`	Converts C-string to float
`atoi`	Converts C-string to int
`atol`	Converts C-string to long int
`rand`	Generates a pseudo-random number
`srand`	Sets seed value for random numbers
(The following require <cctype>)	
`islower`	Returns true if char argument is lowercase
`isupper`	Returns true if char argument is uppercase
`tolower`	Returns the lowercase equivalent of the char argument
`toupper`	Returns the uppercase equivalent of the char argument
(The following require <cmath>)	
`pow`	Raises a number to a power
`sqrt`	Returns square root of a number
(The following require <cstring>)	
`strcat`	Appends a C-string to another C-string
`strcpy`	Copies a C-string
`strlen`	Returns the length of a C-string