

C++ Quick Reference ©2000 Scott/Jones Inc.

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) statement; | if (x < y) x++; |
| if/else | Example |
| if (expression) statement; else statement; | if (x < y) x++; else x--; |
| if/else if | Example |
| if (expression) statement; else if (expression) statement; else statement; | if (x < y) x++; else if (x < z) x--; else y++; |

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

| | |
|---|--|
| Form | Example |
| if (expression) { statement; statement; } | if (x < y) { x++; cout << x; } |

Web Sites

For the *Starting Out with C++ Series*

www.GaddisBooks.com

For Scott/Jones Publishing Company

www.ScottJonesPub.com

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:

```
while (expression)
    statement;
```

Example:

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

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

The do-while Loop

Form:

```
do
    statement;
while (expression);
```

Example:

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

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

C++ Quick Reference (continued) ©2000 Scott/Jones Inc.

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.

Member functions for output formatting

| Name | Description |
|-------------------------|---------------------------------------|
| <code>.precision</code> | sets the number of significant digits |
| <code>.setf</code> | sets one or more ios flags |
| <code>.unsetf</code> | clears one or more ios flags |
| <code>.width</code> | sets field width |

Commonly used stream manipulators

| Name | Description |
|---------------------------|---------------------------------------|
| <code>setprecision</code> | sets the number of significant digits |
| <code>setw</code> | sets field width |
| <code>setiosflags</code> | sets one or more ios flags |

ios flags commonly used with *cout*

| Name | Description |
|------------------------------|--|
| <code>ios::dec</code> | sets decimal notation |
| <code>ios::fixed</code> | sets fixed point notation |
| <code>ios::hex</code> | sets hex notation |
| <code>ios::left</code> | sets left justification |
| <code>ios::oct</code> | sets octal notation |
| <code>ios::right</code> | sets right justification |
| <code>ios::showpoint</code> | forces decimal point & trailing zeros to display |
| <code>ios::showpos</code> | causes + to display in front of positive numbers |
| <code>ios::scientific</code> | sets scientific notation |
| <code>ios::uppercase</code> | Cause the E in E-notation numbers to display in uppercase. |

Using *cin*

Requires `<iostream>` header file.

Member functions for input formatting

| Name | Description |
|-----------------------|------------------------------------|
| <code>.getline</code> | reads a line of input |
| <code>.get</code> | reads a character |
| <code>.ignore</code> | ignores the last character entered |
| <code>.width</code> | sets field width |

Commonly used stream manipulators

| Name | Description |
|-------------------|------------------|
| <code>setw</code> | sets field width |

Some Commonly Used Library Functions

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