{}

[Further Syntax]

Description

Curly braces (also referred to as just "braces" or as "curly brackets") are a major part of the C++ programming language. They are used in several different constructs, outlined below, and this can sometimes be confusing for beginners.  
An opening curly brace { must always be followed by a closing curly brace }. This is a condition that is often referred to as the braces being balanced. The Arduino IDE (Integrated Development Environment) includes a convenient feature to check the balance of curly braces. Just select a brace, or even click the insertion point immediately following a brace, and its logical companion will be highlighted.

Beginner programmers, and programmers coming to C++ from the BASIC language often find using braces confusing or daunting. After all, the same curly braces replace the RETURN statement in a subroutine (function), the ENDIF statement in a conditional and the NEXT statement in a FOR loop.

Unbalanced braces can often lead to cryptic, impenetrable compiler errors that can sometimes be hard to track down in a large program. Because of their varied usages, braces are also incredibly important to the syntax of a program and moving a brace one or two lines will often dramatically affect the meaning of a program.

Example Code

The main uses of curly braces are listed in the examples below.

**Functions**

void myfunction(datatype argument) {

// any statement(s)

}

**Loops**

while (boolean expression) {

// any statement(s)

}

do {

// any statement(s)

} while (boolean expression);

for (initialisation; termination condition; incrementing expr) {

// any statement(s)

}

**Conditional Statements**

if (boolean expression) {

// any statement(s)

}

else if (boolean expression) {

// any statement(s)

}

else {

// any statement(s)

}