The Structure of `for` Loops

A `for` loop is also used to repeat a statement or block of statements as long as a condition is true. The `for` construct in C++ is a loop with initialization, conditional, and incrementation/decrementation clauses. It has the following syntax:

```
for([I]; [C]; [N])
{
  statements--for single statement, the braces may be omitted
}
```

where `I` = initialization statement
`C` = conditional statement that evaluates to some value
`N` = incrementation/decrementation statement

When the `for` statement is first reached by the interpreter, the initialization statement is executed. If the conditional statement is true (nonzero), then the statements are executed, after which the incrementation statement is executed. If the condition is still true, the statements and incrementation are performed again. If the condition is still true, the statements and incrementation are performed again. The statements and incrementation continue to occur as long as the statement is true. Consider the following example:

```
for(int n = 0; n < 5; n++)
{
  cout << "C++" << endl;
}
```

This loop will output the string `C++` five times. The initialization clause sets the integer variable `n` to zero; the condition specifies that the loop should continue while `n` is less than 5; the incrementation clause increases the value of `n` by 1 on each iteration. `n` is referred to as the *index variable.*

The Structure of `do-while` Loops

A `do-while` loop is used to execute a block of statements as long as a given condition becomes true. Equivalently, it is used to repeat statements *until* a condition becomes false. The statements are always executed at least once. The syntax is:

```
do
{
  statement(s);

}while(condition);
```

For example, the following block will prompt the user for a number until he or she enters **`-1`**:

```
do
{
  cout << "Enter a number: ";
  cin >> num;
}while (num != -1);
```

The Structure of `while` Loops

A `while` loop is used to repeat a statement or block of statements as long as a condition is true. It is similar to the `do-while` construct, except that the condition in a `while` loop is tested at the beginning of the loop. The syntax is:

```
while(condition)
{
  statements--braces optional for a single statement
}
```

For example, the fragment below continues to get input until the user enters **`25`**.

```
int n = 0;
while(n != 25)
{
  cout << "Please enter a number: ";
  cin >> n;
}
```

Note: The difference between a `do-while` loop and a `while` loop is that `do-while` loop will run at least once before testing to continue. The `while` loop tests before it even runs the loop once.