

Schleifen

<code>while (...) {...}</code>	while-Schleife
<pre>// Compute number of binary digits for input > 0 unsigned int bin_digits = 0; unsigned int input; std::cin >> input; assert(input > 0); while (input > 0) { input /= 2; ++bin_digits; }</pre>	

<code>do {...} while (...);</code>	do-Schleife
<p>Der Unterschied zur <code>while</code>-Schleife ist, dass der Rumpf der <code>do</code>-Schleife mindestens einmal ausgeführt wird. Sie hat ein <code>“;”</code> am Schluss.</p>	
<pre>int input; do { std::cout << "Enter negative number: "; std::cin >> input; } while (input >= 0); std::cout << "The input was: " << input << "\n";</pre>	

<code>break</code>	Schleife abbrechen
<pre>int input, n; std::cin >> input >> n; // Divide input by n numbers [Note: integer-division]. // Stop if 0 is entered. for (int i = 1; i <= n; ++i) { int k; std::cin >> k; if (k == 0) break; // go straight to Output input /= k; } // Output std::cout << input << " remains\n";</pre>	

<code>continue</code>	zur nächsten Iteration springen
<p>Bei der for-Schleife wird das Inkrement noch ausgeführt.</p>	
<pre>int input, n; std::cin >> input >> n; // Divide input by n numbers [Note: integer-division]. // Skip entered 0's. for (int i = 1; i <= n; ++i) { int k; std::cin >> k; if (k == 0) continue; // go straight to ++i input /= k; } // Output std::cout << input << " remains\n";</pre>	

Datentypen

<code>float</code>	Datentyp für Zahlen mit Nachkommastellen (32 Bit)
Literal: ohne Exponent: <code>288.18f</code> , mit Exponent: <code>0.28818e3f</code>	
Der Modulo-Operator <code>%</code> existiert für <code>float</code> nicht.	
<pre>float a = 288.18f; float b = 0.28818e3f / a; // computations work as expected float c; std::cin >> c; // float user input</pre>	

<code>double</code>	größerer Datentyp für Zahlen mit Nachkommastellen (64 Bit)
Literal: ohne Exponent: <code>288.18</code> , mit Exponent: <code>0.28818e3</code>	
Unterschied zu <code>float</code> : <code>double</code> ist genauer (grössere Präzision und grösseres Exponenten-Spektrum), braucht aber mehr Platz im Speicher (<code>float</code> : 32 Bit, <code>double</code> : 64 Bit). Der Modulo-Operator <code>%</code> existiert für <code>double</code> nicht.	
<pre>double a = 288.18; double b = 0.28818e3 / a; // computations work as expected double c; std::cin >> c; // double user input</pre>	