

Generell

<code>{ ... }</code>	Block
<p>Blöcke spielen eine grosse Rolle, wenn es darum geht, wo im Programm eine Variable gültig ist. So ist eine Variable ab ihrer Deklaration bis hin zum Ende des Blocks, in dem sie definiert wurde potentiell gültig.</p>	
<pre>int main () { unsigned int a; std::cin >> a; if (a < 4) { std::cout << a << " "; // a exists in nested blocks int b = 18; std::cout << b << " "; // b exists here too } else { std::cout << b << " "; // Error: b not declared yet int b = 11; } std::cout << a << " "; // a still exists here std::cout << b << "\n"; // Error: b does not exist anymore return 0; }</pre>	

Datentypen

<code>void</code>	Datentyp für Funktion ohne Rückgabe .
<p><code>void</code>-Funktionen haben keinen Rückgabewert, aber sinnvollerweise einen Effekt (z.B. Textausgabe im Beispiel unten).</p>	
<pre>void print_message (const int a, const double b) { std::cout << "Your assets: " << a << "\n" << "Your interest: " << b << "\n"; }</pre>	

Funktionen

Funktion	Selbstständiger Codeabschnitt
<p>Wichtige Befehle:</p> <p>Definition: <code>int my_fun (bool arg1, float arg2) {...}</code> Deklaration: <code>int my_fun (bool arg1, float arg2);</code> Rückgabe: <code>return my_val;</code> Aufruf: <code>my_fun(true, 3.75f)</code></p> <p>Der Rückgabewert wird immer zum Rückgabetyt konvertiert.</p> <p>Wir verwenden Funktionsdeklarationen dazu, um den Gültigkeitsbereich der zugehörigen Funktionsdefinition zu vergrössern (Siehe Beispiel unten).</p> <p>Bei der Funktionsdeklaration spielt es keine Rolle, ob die Argumente const sind oder nicht.</p> <p>Die Funktionsdefinition ist auch eine Funktionsdeklaration.</p> <p>Jede Funktion, die nicht den Rückgabetyt <code>void</code> hat, muss ein return haben.</p>	
<pre>int foo (unsigned int a); void bar (unsigned int a) { std::cout << foo(--a) << "\n"; // Note: foo is defined below. Thus without the // declaration above, bar would not "know about" // foo. } int foo (const unsigned int a) { bar(a); return a+1; } ...</pre>	

Programmier-Befehle - Woche 05

<pre>// PRE: ... // POST: ...</pre>	Funktionsbeschreibung
<p>PRE-/POST-Conditions gehören vor jede Funktionsdefinition ausser der <code>main</code>-Funktion. (In diesen Programmier-Befehlszusammenfassungen werden sie aber manchmal aus Platzgründen weggelassen.)</p>	
<pre>// PRE: - // POST: return value is a^4 int power_4 (const unsigned int a) { return a*a*a*a; } // PRE: a != 0 // POST: return value is true if and only if a divides 8765 bool divides_8765 (const int a) { return 8765 % a == 0; }</pre>	

Standardbibliothek

<code>std::pow</code>	Potenzieren
Erfordert: <code>#include <cmath></code>	
<pre>double a = std::pow(2.5, 2); // Computes 2.5 ^ 2 == 6.25</pre>	

<code>std::sqrt</code>	Quadratwurzel
Erfordert: <code>#include <cmath></code>	
IEEE 754 garantiert, dass der (mathematisch) exakte Wert auf die nächste repräsentierbare Zahl gerundet wird.	
<pre>double a = std::sqrt(14.0625); // Result: 3.75</pre>	