

# Programmier-Befehle - Woche 3

## Datentypen

<code>bool</code>	Datentyp für Wahrheitswerte
Literal: <code>true</code> , <code>false</code>	
<pre>bool t = true; if (!t == false)     std::cout &lt;&lt; "This is output!\n"; if (t)     std::cout &lt;&lt; "This is output as well!\n";</pre>	

## Operatoren

<code>&amp;&amp;</code>	Logisches UND
Präzedenz: 6 und Assoziativität: links	
<b>Kurzschluss-Auswertung:</b> <code>&amp;&amp;</code> wertet <i>immer</i> den <i>linken</i> Operanden zuerst aus. Ist dieser <i>falsch</i> (also <i>false</i> ), so wird der <i>rechte</i> Operand <i>nicht mehr</i> ausgewertet.	
<pre>if (3 &gt; 2 &amp;&amp; 10 &gt; 11)    // no short circuit evaluation     std::cout &lt;&lt; "Of course not!\n";  int a = 3; if (false &amp;&amp; ++a &lt; 2)    // short circuit evaluation     std::cout &lt;&lt; "Of course not!\n";  std::cout &lt;&lt; a &lt;&lt; "\n"; // Output: 3</pre>	

<code>  </code>	Logisches ODER
-----------------	----------------

( ... )

# Programmier-Befehle - Woche 3

( ... )

Präzedenz: 5 und Assoziativität: links

**Kurzschluss-Auswertung:** `||` wertet auch *immer* den *linken* Operanden zuerst aus. Ist dieser *wahr* (also `true`), so wird der *rechte* Operand *nicht mehr* ausgewertet.

```
if (8 < 3 || -2 > -5)    // no short circuit evaluation
    std::cout << "Yes!\n";

int a = 3;
if (3 < 8 || ++a < 2)    // short circuit evaluation
    std::cout << "Yes!\n";

std::cout << a << "\n"; // Output: 3
```

!

Logisches NICHT

Präzedenz: 16 und Assoziativität: rechts

```
int a;
int b;
int c;
std::cin >> a >> b >> c; // read three int values from user

if ( ! (a <= b && c <= b) )
    std::cout << "b is not max!\n";
```

<

strikt kleiner

Präzedenz: 11 und Assoziativität: links

Sonst gibt es noch:

>     **strikt grösser**  
<=    **kleiner gleich**  
>=    **grösser gleich**

( ... )

# Programmier-Befehle - Woche 3

( ... )

```
int a;
int b;
std::cin >> a >> b; // read two int values from user

if (a < b)
    std::cout << "a smaller than b\n";
```

**==**

exakt gleich

Präzedenz: 10 und Assoziativität: links

Sonst gibt es noch:

**!=**      **ungleich**

```
int a;
int b;
std::cin >> a >> b; // read two int values from user

if (a == b)
    std::cout << "a is equal to b\n";
```

## Schleifen

**for (...)** {...}

for-Schleife

Wenn man eine leere Condition als Abbruchbedingung angibt, so wird diese als *wahr* interpretiert.

```
unsigned int n;
std::cin >> n;

// Compute 1 + 2 + 3 + ... + n
unsigned int sum = 0;
for (unsigned int i = 1; i <= n; ++i)
    sum += i;
std::cout << "1 + 2 + ... + n = " << sum << "\n";
```

## Generell

<code>if-else</code>	bedingtes Ausführen von Code
Der <code>else</code> -Teil ist optional.	
<pre>int a; int b; std::cin &gt;&gt; a &gt;&gt; b; // read two int values from user  // if a &lt; b then output 3; otherwise output 8 if (a &lt; b)     std::cout &lt;&lt; 3 &lt;&lt; "\n"; else     std::cout &lt;&lt; 8 &lt;&lt; "\n";</pre>	

<code>assert</code>	sofortiges Stoppen des Programms bei Verletzung einer Bedingung (zu Testzwecken)
Erfordert: <code>#include&lt;cassert&gt;</code>	
Wenn das fertige Programm veröffentlicht werden soll, kann man die <code>assert</code> -Befehle bequem deaktivieren.	
<pre>int a; int b; std::cin &gt;&gt; a &gt;&gt; b; // read two int values from user assert(b != 0); // prevent division by 0 std::cout &lt;&lt; a / b &lt;&lt; "\n";</pre>	