

Datentypen

<code>bool</code>	Datentyp für Wahrheitswerte
Literal: <code>true</code> , <code>false</code>	
<pre>// The following loop reads in numbers until a number < 3 // is entered. for (bool another = true; another;) { int a; std::cin >> a; if (a < 3) // Note: a < 3 is of type bool another = false; }</pre>	

<code>const ...</code>	<i>Schreibzugriff</i> auf Variable verboten
Gemeint ist natürlich der Schreibzugriff <i>nach</i> der Initialisierung.	
<pre>int a = 3; const int b = 4; a = 5; // valid b = 3; // not valid since b is const int c = -2 * b; // valid since just WRITE-access is forbidden // by "const"</pre>	

Operatoren

<code>&&</code>	Logisches UND
Präzedenz: 6 und Assoziativität: links	
Kurzschluss-Auswertung: <code>&&</code> wertet <i>immer</i> den <i>linken</i> Operanden zuerst aus. Ergibt dieser <code>false</code> , so wird der <i>rechte</i> Operand <i>nicht mehr</i> ausgewertet.	

(...)

Programmier-Befehle - Woche 3

(...)

```
if (3 > 2 && 10 > 11)    // no short circuit evaluation
    std::cout << "Of course not!\n";

int a = 3;
if (false && ++a < 2)    // short circuit evaluation
    std::cout << "Of course not!\n";

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

||

Logisches ODER

Präzedenz: 5 und Assoziativität: links

Kurzschluss-Auswertung: || wertet auch *immer* den *linken* Operanden zuerst aus. Ergibt dieser *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,b,c;
std::cin >> a >> b >> c; // read three int values from user

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

Programmier-Befehle - Woche 3

<code><</code>	strikt kleiner
Präzedenz: 11 und Assoziativität: links	
Sonst gibt es noch: <code>></code> strikt grösser <code><=</code> kleiner gleich <code>>=</code> grösser gleich	
<pre>int a,b; std::cin >> a >> b; // read two int values from user if (a < b) std::cout << "a smaller than b\n";</pre>	

<code>==</code>	exakt gleich
Präzedenz: 10 und Assoziativität: links	
Sonst gibt es noch: <code>!=</code> ungleich	
<pre>int a,b; std::cin >> a >> b; // read two int values from user if (a == b) std::cout << "a is equal to b\n";</pre>	

Schleifen

<code>for (...) {...}</code>	for-Schleife
Wenn man eine leere Condition als Abbruchbedingung angibt, so wird diese als true interpretiert.	

(...)

Programmier-Befehle - Woche 3

(...)

```
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 << "Intermediate result: " << sum << "\n";
}
```

Generell

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

<code>assert</code>	sofortiges Stoppen des Programms zu Testzwecken
Erfordert: <code>#include <cassert></code>	
Wenn das fertige Programm veröffentlicht werden soll, kann man die <code>assert</code> -Befehle bequem deaktivieren (siehe <code>assertion2.cpp</code>).	
<pre>int a,b; std::cin >> a >> b; // read two int values from user assert(b != 0); // prevent division by 0 std::cout << a / b << "\n";</pre>	