Datentypen

`bool`	Datentyp für **Wahrheitswerte**
Literal: `true`, `false`	

```
for (bool another = true; another;) {
    int a;
    std::cin >> a;
    if (a < 3)      // Note: a < 3 is of type bool
        another = false;
}
```

Operatoren

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

```
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;

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**

(...)

(...)

```
int a,b;
std::cin >> a >> b;

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

==	**exakt gleich**

Präzedenz: 10 und Assoziativität: `links`

Sonst gibt es noch:
`!=` **ungleich**

```
int a,b;
std::cin >> a >> b;

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

Generell

if-else	**bedingtes Ausführen** von Code

Der `else`-Teil ist optional.

```
int a,b;
std::cin >> a >> b;

// if a < b then output "Ja"; otherwise output "Nein"
if (a < b) {
    std::cout << "Ja\n";
} else {
    std::cout << "Nein\n";
}
```

`assert`	**sofortiges Stoppen** des Programms zu Testzwecken
Erfordert: `#include <cassert>`	

```
int a,b;
std::cin >> a >> b;
assert(b != 0); // prevent division by 0
std::cout << a / b << "\n";
```

Schleifen - Teil 1

`for (...) {...}`	`for`**-Schleife**
Wenn man eine leere Condition als Abbruchbedingung angibt, so wird diese als `true` interpretiert.	

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

unsigned int sum = 0;
for (unsigned int i = 1; i <= n; ++i) {
    sum += i;
    std::cout << "Intermediate result: " << sum << "\n";
}
```