C++ Cheat-Sheet Lite

https://grader.soi.ch/workshop/

Eingabe/Ausgabe

```
#include <soi>

signed main() {
  int a = read_int();
  int b = read_int();
  int answer = a + b;
  print(answer);
}
```

- `read_int()` liest die nächste Zahl von der Eingabe.
- `print(x)` schreibt die Zahl x in die Ausgabe.
- `print("hallo")` schreibt "Hallo" in die Ausgabe.
- `print(1, 2, 3, 4)` schreibt die 4 Zahlen "1 2 3 4" auf die Ausgabe.

Debugging

```
int x=4;
dbg(x);
dbg(x + 3);
```

Gibt aus:

```
[task.cpp:5 (main)] x = 4 (int)
[task.cpp:6 (main)] x + 3 = 7 (int)
```

"task.cpp": Datei, "main": Funktion, "x" bzw "x+3": Ausdruck, "4" bzw "7" Wert, "int": Typ.

Variablen

Variablen müssen mit einem Buchstaben anfangen und können danach aus Buchstaben, Zahlen und Unterstrichen bestehen. Alle Variablen haben einen Typ.

- Deklariere Variablen kurz vor ihrer ersten Nutzung.
- Gebe ihr einen guten Wert.

```
int n = read_int(); // einlesen
int sum = 0; // die Summe von nichts ist 0
int index = -1; // immer ein ungültiger Index
bool done = false; // ob bereits fertig
bool are_we_done_yet; // etwas zu lang, done wäre besser
set<pair<int, int>> active_vertices; // später...
```

If-Abfragen

If:

```
if (a == b) {
    print("a und b sind gleich");
}
```

If-else:

```
if (a == b) {
    print("a und b sind gleich");
} else {
    print("a und b sind verschieden");
}
```

If und else verschachteln:

```
if (a == b) {
    print("a und b sind gleich");
} else {
    if (a > b) {
        print("a ist grösser als b");
    } else {
        print("b ist grösser als a");
    }
}
```

Else if:

```
if (a == b) {
    print("a und b sind gleich");
} else if (a > b) {
    print("a ist grösser als b");
} else {
    print("b ist grösser als a");
}
```

Richtlinie: Versuche, so wenig wie möglich zu verschachteln.

Bedingungen

Bedingungen sollten Ausdrücke sein, die ein Wahrheitswert (`true` oder `false`) zurück geben. Hier einige nützliche Ausdrücke, um einige Bedingungen mit zwei Zahlen a und b zu testen:

- `a==b` ist `true` falls a und b gleich sind und `false` sonst.
- `a!=b` ist `true` falls a und b nicht gleich sind und `false` sonst.
- `a<b` ist `true` falls a kleiner ist als b und `false` sonst.
- `a>b` ist `true` falls a grösser ist als b und `false` sonst.
- `a<=b` ist `true` falls a kleiner oder gleich ist als b und `false` sonst.
- `a>=b` ist `true` falls a grösser oder gleich ist als b und `false` sonst.

Du kannst mehrere Wahrheitswerte mit logischen Operatoren kombinieren:

- `!a` ist `true` falls a `false` ist und `false` sonst.
- `a&&b` ist `true` falls sowohl a als auch b `true` sind und `false` sonst.
- `a||b` ist `true` falls entweder a oder b `true` ist und ist `false` falls beide `false` sind.

Die Operatorenpräzedenz ist: `!` > numerische Operatoren > `==`, `!=` > `&&` > `||`.

While-Schleife

Führt den Schleifenrumpf wiederholt aus, solange die Schleifenbedingung gültig bleibt.

```
int i = 0;
while (i < 10) {
    print("das Quadrat von", i, "ist", i*i);
    ++i;
}
```

Eine Schleife kann vorzeitig mit **break** verlassen werden.

```
int i = 0;
while (true) { // führe diese Schleife für immer aus
    print("Überprüfe, ob die Wurzel von 144 gleich ", i, "is
    if (i*i == 144)
        break;
    i++;
}
print("Antwort:", i);
```

For-Schleife

```
for (int i = 0; i < 10; ++i) {
    print("das Quadrat von", i, "ist", i*i);
}
```

Alle drei Teile sind optional.

```
int i = 0; // i wird nach der Schleife noch gebraucht
for (; i < 10; ++i) {
    if (this_i_is_special(i))
        break;
}
print(i); // hat den Wert des speziellen i's oder 10.
```

For-ever-Schleife: Idiomatischer als `while(true)`:

```
int i = 0;
for (;;) { // läuft für immer ...
    i = next(i);
    if (my_special_condition(i)) {
        break; // ... oder zumindest bis hier
    }
    print("current i:", i);
}
```

Beispiel: Primzahltest

Aufgabe: Finde heraus ob eine gegebene Zahl n eine Primzahl ist oder nicht.

```
bool is_prime = true;
for (int i = 2; i < n; i++) {
    if (n % i == 0) {
        is_prime = false;
        break;
    }
}
```

Operatorpräzedenz

1. `!`
2. `*`, `/`, `%`
3. `+`, `-`
4. `<`, `>`, `<=`, `>=`
5. `==`, `!=`
6. `&&`
7. `||`