C++ Cheat-Sheet Lite

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

Input/Output

```
#include <soi>

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

- `read_int()` reads the next line from the input.
- `print(x)` writes the integer x to the output.
- `print("hallo")` writes “Hallo” to the output.
- `print(1, 2, 3, 4)` writes the 4 integers “1 2 3 4” to the output.

Debugging

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

Outputs:

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

“task.cpp”: File, “main”: Function, “x” bzw “x+3”: Expression, “4” i.e. “7” Value, “int”: Type.

Variables

Variables must start with a letter and can contain letters, numbers and underscores afterwards. Each variable has a type.

- Declare variables shortly before you use them the first time.
- Give it a good initial value.

```
int n = read_int(); // input
int sum = 0; // the summe of nothing is 0
int index = -1; // initialize to an invalid index
bool done = false; // whether we are already done
bool are_we_done_yet; // too long, use shorter names
set<pair<int, int>> active_vertices; // later...
```

If-Abfragen

If:

```
if (a == b) {
    print("a and b are equal");
}
```

If-else:

```
if (a == b) {
    print("a and b are equal");
} else {
    print("a and b are different");
}
```

Nested if and else:

```
if (a == b) {
    print("a and b are equal");
} else {
    if (a > b) {
        print("a is bigger than b");
    } else {
        print("b is bigger than a");
    }
}
```

Else if:

```
if (a == b) {
    print("a and b are equal");
} else if (a > b) {
    print("a is bigger than b");
} else {
    print("b is bigger than a");
}
```

Guideline: Try to nest as little as possible.

Conditions

Conditions should be expressions that return a logic value (`true` or `false`) Here are some useful expressions to test conditions between two integers a and b:

- a==b is `true` if a and b are equal, `false` otherwise.
- a!=b is `true` if a and b are not equal, `false` otherwise.
- a<b is `true` if a smaller than b and `false` otherwise.
- a>b is `true` if a bigger than b and `false` otherwise.
- a<=b is `true` if a smaller or equal to b and `false` otherwise
- a>=b is `true` if a bigger or equal to b and `false` otherwise

You can combine multiple logic values with logical operators:

- !a is `true` if a is `false` and `false` otherwise.
- a&&b is `true` if both a are b are `true` ,`false` otherwise.
- a||b is `true` if either a or b is `true` and is `false` if both are `false`.

The operator precedence is: ! > numeric operators > ==, != > && > ||.

While-Loop

Repeatedly executes the loop body as long as the loop condition is true.

```
int i = 0;
while (i < 10) {
    print("the square of", i, "is", i*i);
    ++i;
}
```

A loop can be terminated early with a **break**.

```
int i = 0;
while (true) { // repeat this loop forever
    print("Check whether the square root of 144 is equal to"
    if (i*i == 144)
        break;
    i++;
}
print("Answer:", i);
```

For-Loop

```
for (int i = 0; i < 10; ++i) {
    print("the square of", i, "is", i*i);
}
```

All three parts are optional.

```
int i = 0; // i will be used after the loop
for (; i < 10; ++i) {
    if (this_i_is_special(i))
        break;
}
print(i); // has the value of the special i or 10.
```

For-ever-loop: More idiomatic than `while(true)`:

```
int i = 0;
for (;;) { // loops forever ...
    i = next(i);
    if (my_special_condition(i)) {
        break; // ... or at least until here
    }
    print("current i:", i);
}
```

Example: Primality test

Output: Determine if a given integer n is a prime or not.

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

Operator Precedence

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