for loops

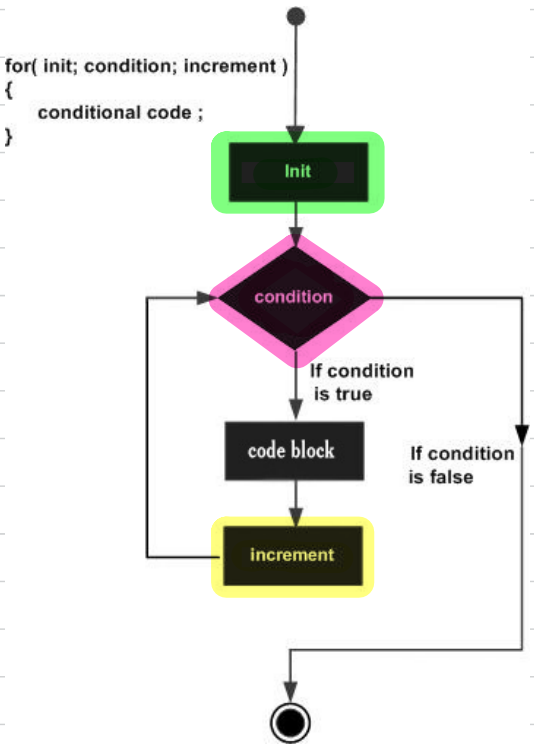
```
for ( init; condition; increment ) {
   statement(s);      // code block
}
```

Here is the flow of control in a for loop –

- The **init** step is executed first, and only once. This step allows you to declare and initialize any loop control variables. You are not required to put a statement here, as long as a semicolon appears.
- Next, the **condition** is evaluated. If it is true, the body of the loop is executed. If it is false, the body of the loop does not execute and flow of control jumps to the next statement just after the for loop.
- After the body of the for loop executes, the flow of control jumps back up to the **increment** statement. This statement can be left blank, as long as a semicolon appears after the condition.
- The condition is now evaluated again. If it is true, the loop executes and the process repeats itself (body of loop, then increment step, and then again condition). After the condition becomes false, the for loop terminates.

→ condition kann auch leer sein e.g. `for ( int i = 0 ; ; ++i ) { //loop }`
↳ heisst es wird immer als true evaluiert.
(Was dann heisst ihr müsst aufpassen, keinen infinite loop zu machen. D.h. ihr müsst den loop anders "beenden" bzw "exiten".)

→ condition kann auch `i += 5` oder `i *= 2` sein



break / continue

- loops können auch mittels break beendet werden.
  - ↳ break: wenn break ausgeführt wird, wird der loop terminiert und das nächste Statement nach dem loop wird ausgeführt

The break statement is used to terminate the loop. As soon as a break statement is executed within the loop, the loop iteration stops and the very next statement after the loop starts executing.

↳ continue: hier wird der Rest der current/aktuelle loop iteration übersprungen und die nächste iteration wird ausgeführt.

A continue statement, just like a break statement, is a loop control statement. Instead of terminating the loop and exiting from it, the continue statement forces the loop to skip the current iteration and continue from the next iteration.

Hier noch mehr dazu: https://www.educative.io/answers/how-to-use-break-and-continue-in-cpp

https://www.simplilearn.com/tutorials/cpp-tutorial/break-and-continue-statements-cpp#:~:text=Break%20statement%20stops%20the%20entire,resumes%20with%20the%20successive%20iterations.

Debugging / Fehler finden Tipps

→ loops

↳ zuerst normale cases anschauen e.g. mit $n = 5$ $n = 10$ ka..

↳ dann versuchen edge values zu finden e.g. for $n = 0$ für $n = \max$ uint

↳ schauen was in der ersten und letzten loop iteration passiert

→ "Gefühl" für den loop / was "soll" passieren?

→ overflow? non-terminating loop? iteriert der loop zu lange? stoppt er zu früh?

→ etwas allgemeiner ...

↳ Benutzt print statements um zu sehen
1) was macht mein Code tatsächlich?
2) Bis wo im Code stimmt alles noch?

=> Bei längeren Sachen hilft es wenn ihr zuerst in grossen Schritten durch den Code geht bis ihr den Block gefunden habt, in welchem ihr den Bug vermutet und erst dann anfängt jede Zeile / Statement genau zu überprüfen.

↳ Benutzt Google / Stackoverflow

→ Hilfreiche Websiten:

https://www.freecodecamp.org/news/what-is-debugging-how-to-debug-code/

https://www.geeksforgeeks.org/debugging-tips-to-get-better-at-it/