Was ist NXC?

NXC ist eine Programmiersprache, mit der man Programme für den NXT-Brick programmieren kann. Sie stammt von der Programmiersprache C ab; NXC bedeutet *Not eXactly C* ("nicht wirklich C").

Ein NXC-Programm besteht aus *Tasks* (Aufgaben), die der Roboter ausführen kann. Jedes Programm muss einen Task namens `main` ("main task" ist Englisch für "Hauptaufgabe") haben. Das ist der Task, den der Brick beim Starten des Programms ausführt. Jeder Task wird mit geschweiften Klammern umschlossen.

Jeder Task — und jedes Programm — wird von oben nach unten abgearbeitet. Wir gehen davon aus, dass der Brick mit der ersten Zeile beginnt und dann das Programm Zeile für Zeile abarbeitet. Dabei gibt es verschiedene Ausnahmen, die du später kennenlernst. Fürs erste sollst du dir merken: Programme werden wie beim Lesen abgearbeitet — von links oben nach rechts unten.

Hier ist ein Grundgerüst, in das man einfache Programme einsetzen kann:

```
task main() {
 // Hier stehen die Anweisungen des Hauptprogramms
}
```

Einige NXC-Funktionen

Funktion	Argumente	Beschreibung
`OnFwd`	Output-Anschluss Prozent	Motoren an den Anschlüssen einschalten, vorwärts
`OnRev`	Output-Anschluss Prozent	Motoren an den Anschlüssen einschalten, rückwärts
`Off`	Output-Anschluss	Motoren an den Anschlüssen ausschalten
`SetSensorLight`	Input-Anschluss	An dem Anschluss ist ein Lichtsensor angeschlossen
`SetSensorSound`	Input-Anschluss	An dem Anschluss ist ein Schallsensor angeschlossen
`SetSensorTouch`	Input-Anschluss	An dem Anschluss ist ein Berührungssensor angeschlossen
`SetSensorUltrasonic`	Input-Anschluss	An dem Anschluss ist ein Ultraschallsensor angeschlossen
`Wait`	Millisekunden	Programm für eine bestimmte Zeit anhalten
`RotateMotor`	Output-Anschluss Geschwindigkeit Winkel	Motor um einen bestimmten Winkel drehen

Ereignisse

Manchmal muss man im Programm des Roboters auf ein Ereignis warten — also darauf, dass etwas bestimmtes passiert. Ganz besonders oft warten wir darauf, dass ein Sensor etwas bestimmtes erkennt, z.B. eine Berührung oder eine helle anstatt einer dunklen Fläche.

Eine Möglichkeit, das Programm zum Warten zu bringen, ist die Verwendung einer `while`-Schleife, die so lange gar nichts tut, bis sich der Zustand eines Sensors ändert. Beispiel:

```
while (Sensor(IN_1) < 30) {}
```

Diese Schleife hält das Programm so lange an, wie der Wert des Sensors am Anschluss 1 kleiner als 30 ist. Anders ausgedrückt: Das Programm tut nichts, bis der Wert des Sensors 30 oder größer ist.

Stelle dir vor, am Anschluss 1 wäre ein Lichtsensor angeschlossen. Dann würde das Programm so lange anhalten, wie der Untergrund schwarz ist.

Wichtig: Wenn du vor der Schleife Motoren eingeschaltet hast, laufen diese trotzdem weiter — das Programm wird daran nur nichts mehr ändern!

Reihenfolge eines Programms

Zuletzt treffen wir noch eine Vereinbarung, in welcher Reihenfolge wir ein Programm schreiben.

1. **Sensoren angeben** — als erstes teilen wir dem Brick mit, welche Sensoren angeschlossen sind.
2. **Motoren einschalten** — dann schalten wir die Motoren ein, die ganz am Anfang laufen sollen.
3. **Auf Ereignisse warten** — wir warten, bis etwas bestimmtes passiert.
4. Nach einem Ereignis werden eventuell andere Motoren eingeschaltet, Motoren rückwärts fahren gelassen oder ähnliches — ab hier hast du freie Bahn!

Beispiel:

```
task main() {
 // 1. Sensoren angeben
 SetSensorLight(IN_1);

 // 2. Motoren einschalten
 OnFwd(OUT_AB, 75);

 // 3. Auf Ereignisse warten
 while (Sensor(IN_1) < 30) {}

 // 4. Motoren ausschalten
 Off(OUT_AB);
}
```