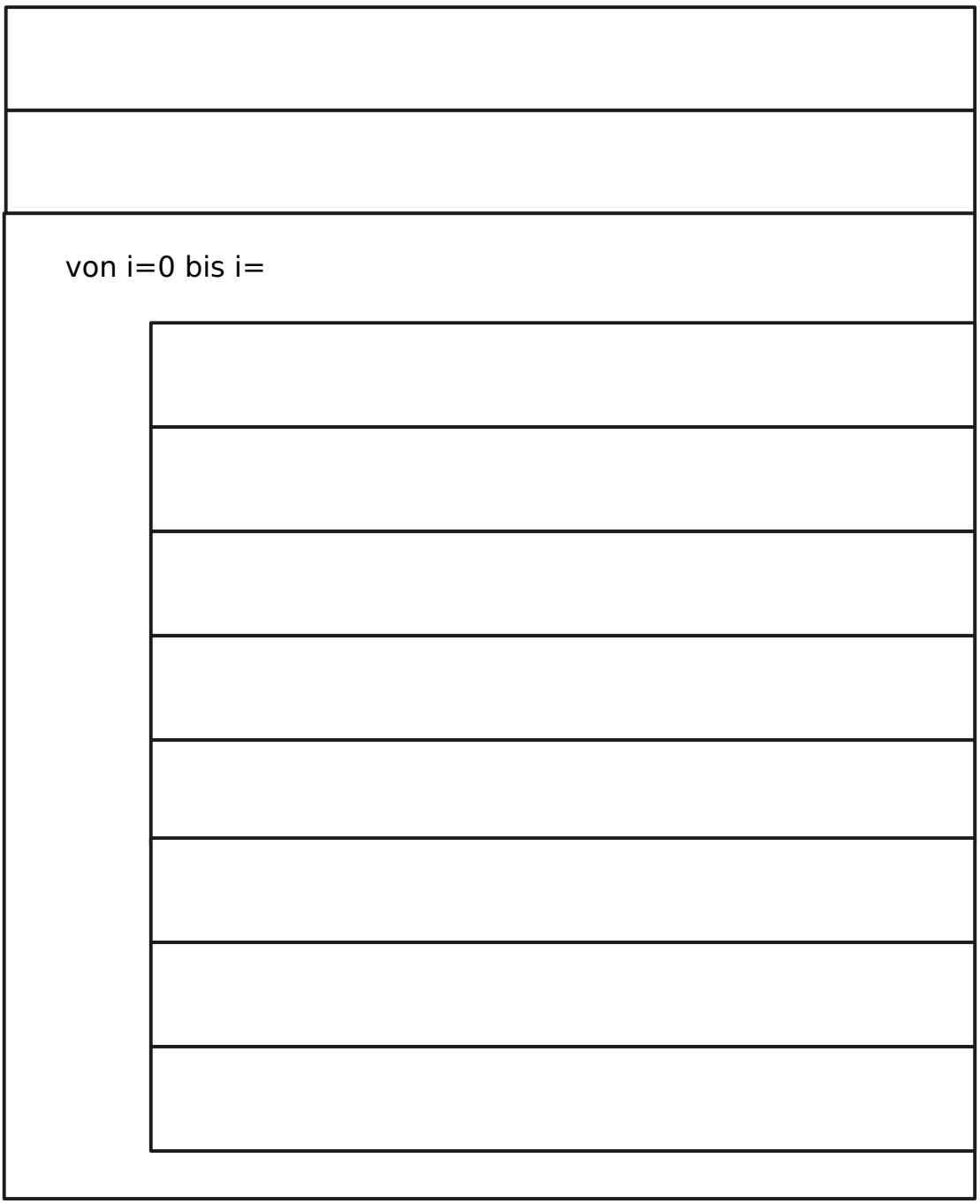
## Wiederholende Übungen:

```
void vorwaerts(int port,int tempo)
  SetPower(port, tempo);
  OnFwd(port);
}

void rueckwaerts(int port,int tempo)
  SetPower(port, tempo);
  OnRev(port);
}

void links(int winkel) {
  vorwaerts(OUT_A,3);
  rueckwaerts(OUT_B,3);
  Wait(winkel);
  Off(OUT_A + OUT_B);
}

void rechts(int winkel) {
  rueckwaerts(OUT_A,3);
  vorwaerts(OUT_B,3);
  Wait(winkel);
  Off(OUT_A + OUT_B);
}

task main() {
  int i;
  for (i=0; i<6; i++) {
    vorwaerts(OUT_A+OUT_B,7);
    Wait(100);
    Off(OUT_A + OUT_B);
    rechts(90);
    vorwaerts(OUT_A+OUT_B,7);
    Wait(100);
    Off(OUT_A + OUT_B);
    links(90);
  }
}
```

1. Definition und Aufruf einer Prozedur:
   a. Nenne die Namen der selbstdefinierten Prozeduren.
   b. Unterstreiche alle Aufrufe dieser Prozeduren. Beachte dabei, dass eine Prozedur auch im Anweisungsteil einer anderen Prozedur aufgerufen werden kann.
   c. Nenne die rekursive Prozedur, falls du eine findest.
2. Hauptprogramm (`main`):
   a. Vervollständige das Struktogramm des Hauptprogramms rechts.
   b. Verändere die formalen Parameter, die für die Stufenlänge und -breite der entstehenden Treppe verantwortlich sind. Nenne eine Möglichkeit, dass die Stufe doppelt so hoch wie lang ist.

c. Verwende die Variable i, um zu erreichen, dass die Länge und Breite der Stufen immer mehr zunimmt. Wieviele Stufen darf die Treppe dann nur höchstens haben?




Verwendete Strukturelemente:

_____ und _____

Teil 3 – **Schleifenvarianten und Sensoren**

Neben der Zählschleife gibt es zwei weitere Schleifentypen. Beiden ist gemein, dass die Zahl der Durchläufe variabel ist. Die Abbruchbedingung der Schleife muss also anders formuliert werden:

| | *NXT-Programm, Schleifenvariante 1* | *Spybot-Programm, Schleifenvariante 2* |
|---|---|---|
| 1 | `// globale Variablen` | `// globale Variablen` |
| 2 | `int wert;` | `int wert;` |
| 3 | | |
| 4 | `task main() {` | `task main() {` |
| 5 | `  SetSensorTouch(IN_1);` | |
| 6 | `  wert = Sensor(IN_1);` | `  SetPower(OUT_A+OUT_B, 4);` |
| 7 | `  OnFwd(OUT_AB, 70);` | `  OnFwd(OUT_A + OUT_B);` |
| 8 | | |
| 9 | `  while (wert == 0) {` | `  do {` |
| 10 | `    Wait(50);` | `    wert = SensorValue(0);` |
| 11 | `    wert = Sensor(IN_1)` | `    Wait(5);` |
| 12 | `  }` | `  } while (wert == 0);` |
| 13 | | |
| | `  Off(OUT_AB);` | `  Off(OUT_A + OUT_B);` |
| | `}` | `}` |

Anmerkung: Da beim Spybot der Tastsensor fest eingebaut ist, muss er nicht erst definiert werden (Zeile 5 im NXT-Programm).

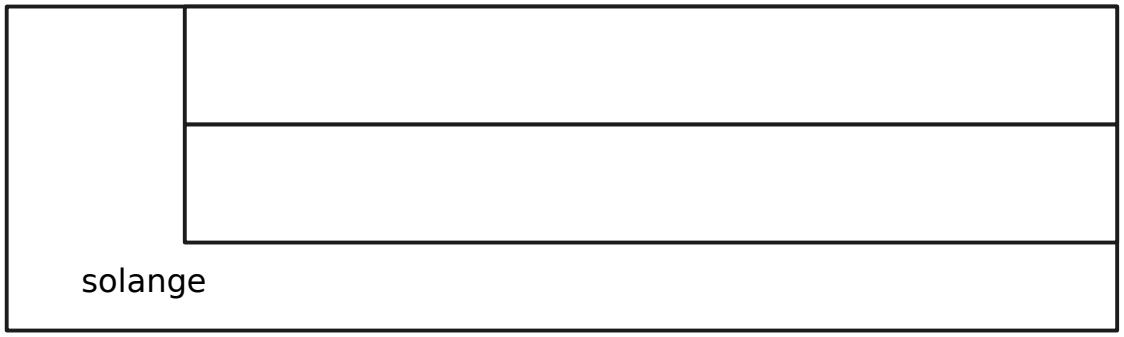
**Aufgaben (Verständnis des Quelltextes)**

1. Lies beide Programme. Notiere die Namen der dort definierten Variablen. Was bedeutet der Datentyp `int`?
2. Finde in beiden Programmen die Schleife und markiere den Bedingungs- und Anweisungsteil.
   Erkläre den Unterschied zwischen den Schleifenvarianten.
   Ordne die Varianten den Struktogrammelementen zu.

**Aufgaben (Testen und Überarbeiten des Quelltextes)**

3. Teste das Programm und finde heraus, was die Bedingung "`wert == 0`" bedeutet: _____
4. Erstelle eine neue Prozedur `zumHindernis()`. Verschiebe alle Anweisungen der `main`-Methode in den Anweisungsteil dieser Prozedur. Rufe `zumHindernis()` so auf, dass der Roboter viermal zwischen zwei Hindernissen hin- und herfährt.

**Aufgabe 2: Struktogrammelemente**




Strukturelement: _____ (Variante ___)


Strukturelement: _____ (Variante ___)

**zum Weiterarbeiten (Schleifen)**

5. Der so genannte Feuerwehralgorithmus navigiert durch ein Labyrinth, indem er immer in dieselbe Richtung ausweicht, z.B. nach links, wenn eine Wand im Weg ist. Programmiere den Roboter so, dass er nach diesem Verfahren arbeitet. Verwende neben `zumHindernis()` auch die bereits vorhandenen Prozeduren.

**Wettkampf**

6. Der Roboter soll von einem Startpunkt zu einem Hindernis und zurück fahren. Es gewinnt, wer dem Startpunkt am nächsten kommt.

**Hinweis zum Wettkampf**

Da die Entfernung zum Hindernis nicht vorhersagbar ist, muss man die Wegstrecke messen. Dazu kann eine Variable x verwendet werden, deren Wert in jedem Schleifendurchlauf erhöht oder vermindert wird: `x = x + 1;` oder, kürzer, `x++;`