OSTBAYERISCHE TECHNISCHE HOCHSCHULE REGENSBURG

INFORMATIK UND MATHEMATIK

Lösung: Übungsaufgaben Blatt 1 - Karel

Programmieren 1 - Prof. Dr. Markus Heckner

Karel und sein Wohnzimmer

```c
/*
 * File: LivingRoomKarel.c
 * -----------------------------------------
 *
 * In this program, Karel simply climbs a cupboard and picks
 * a beeper on its way.
 * The goal is to demonstrate how to think top-down
 * and how to break down a program
 * into functions.
 *
 */

#include "karel.h"

void setup() {
    loadWorld("LivingRoom");
}

void turnRight() {
    turnLeft();
    turnLeft();
    turnLeft();
}

/*
 * Move to the bottom of cupboard
 *
 * pre-condition: Karel stands at bottom, left corner, facing east
 * post-condition: Karel stands at bottom of cupboard, facing east
 */
void moveToCupboard() {
    move();
    move();
    move();
    move();
    move();
}
```

```
38
39
40  /*
41   * Climb up side of cupboard
42   *
43   * pre-condition: karel stands at bottom of cupboard, facing
         east
44   * post-condition: Karel stands at top of cupboard, one field
         left, facing east
45   */
46  void climbCupboard() {
47      turnLeft();
48      move();
49      move();
50      move();
51      turnRight();
52
53  }
54
55  /*
56   * Move to where the beeper lies
57   *
58   * pre-condition: Karel stands at top of cupboard, one field
         left, facing east
59   * post-condition: Karel stands at top of cupboard, on beeper
         spot,
60   * facing east
61   */
62  void moveToBeeper() {
63      move();
64      move();
65  }
66
67  /*
68   * Move to right wall
69   *
70   * pre-condition: Karel stands on beeper spot, facing east
71   * post-condition: Karel stands on top of cupboard, directly
         before right wall, facing east
72   */
73  void moveToWall() {
74      move();
75      move();
76  }
77
78  void run() {
79      moveToCupboard();
80      climbCupboard();
81      moveToBeeper();
```

```
    pickBeeper();
    moveToWall();
}
```

Die zerstörten Säulen

```
/*
 * File: damagedPillarKarel.c
 * --------------------------------------
 *
 * In this program, Karel repairs broken pillars by
 * replacing holes with beepers.
 *
*/

#include "karel.h"

void setup() {
    loadWorld("DamagedPillar2");
}

/*
 * Puts a beeper, if no beeper is there
 *
 */
void repairSpot() {
    if (noBeepersPresent()) {
        putBeeper();
    }
}

/*
 * Turn right
 */
void turnRight() {
    turnLeft();
    turnLeft();
    turnLeft();
}

/*
 * Climb down a repaired pillar
 *
 * post-condition: Karel stands at top of pillar, facing east
 * pre-condition: Karel stands at bottom of repaired pillar,
    facing east
 *
 */
void climbDown() {
    turnRight();
```

```
44      while (frontIsClear()) {
45          move();
46      }
47      turnLeft();
48  }
49
50  /*
51   * Move to the next column, after the current column was
         repaired
52   *
53   * pre-condition: Karel stands at bottom of repaired pillar,
         facing east
54   * post-condition: Karel stands at bottom of next pillar,
         facing east
55   *
56   */
57  void moveToNextPillar() {
58      int i;
59      for (i = 0; i < 4; i++) {
60          move();
61      }
62
63  }
64
65  /*
66   * Repair the pillar by adding beepers while moving up, then
         climb down
67   *
68   * pre-condition: Karel stands at bottom of pillar, facing east
69   *post-condition: Karel stands at top of pillar, facing east
70   *
71   */
72  void repairPillarMovingUp() {
73      turnLeft();
74      while (frontIsClear()) {
75          repairSpot();
76          move();
77      }
78      repairSpot();
79      turnRight();
80  }
81
82  /*
83   * Repair the pillar by adding beepers while moving up, then
         climb down
84   *
85   * pre-condition: Karel stands at bottom of pillar, facing east
86   * post-condition: Karel stands at bottom of pillar, facing
         east
```

```
 *
 */
void repairPillar() {
    repairPillarMovingUp();
    climbDown();
}

void run() {
    while (frontIsClear()) {
        repairPillar();
        moveToNextPillar();
    }
    repairPillar();
}
```

Flag-Karel

```
/*
 * File: FlagDistanceKarel.c
 * -------------------------
 *
 * In this program, Karel measures the distance to a flag
 * and creates a pile of beepers equal to the distance from his
 * starting position to the flag.
 *
*/

#include "karel.h"

void setup() {
    loadWorld("flagDistance1");
}

void turnAround() {
    turnLeft();
    turnLeft();
}

void placeOneBeeperAhead() {
    move();
    putBeeper();
}

void moveOneFieldBack() {
    turnAround();
    move();
    turnAround();
}

/*
```

```
34  * Move a pile of beepers one field ahead
35  * Pre-condition: Karel stands on beeper pile
36  * Post-condition: Karel stands one field ahead of start
        position on moved pile
37  */
38
39  void moveBeeperPileAhead() {
40      while (beepersPresent()) {
41          pickBeeper();
42          placeOneBeeperAhead();
43          moveOneFieldBack();
44      }
45      move();
46  }
47
48  /*
49   * Run until Karel stand before the flag
50  */
51  void runToFlag() {
52      while (frontIsClear()) {
53          move();
54      }
55  }
56
57  void stepUp() {
58      turnLeft();
59      move();
60  }
61
62  /*
63   * Measures the distance between Karel and a flag
64   *
65   * Pre-condition: Karel stands on left edge of world facing
        east
66   * Post-condition: Karel stands one field above starting
        position
67   *
68  */
69  void measureDistance() {
70      runToFlag();
71      turnAround();
72      putBeeper();
73      while (frontIsClear()) {
74          moveBeeperPileAhead();
75          putBeeper();
76      }
77      pickBeeper();
78      turnAround();
79      stepUp();
```

```
80 }
81
82 void run() {
83     if (frontIsClear()) {
84         measureDistance();
85     } else {
86         stepUp();
87     }
88 }
```