OSTBAYERISCHE TECHNISCHE HOCHSCHULE REGENSBURG

INFORMATIK UND MATHEMATIK

Übungsaufgaben Blatt 1 - Karel the Robot

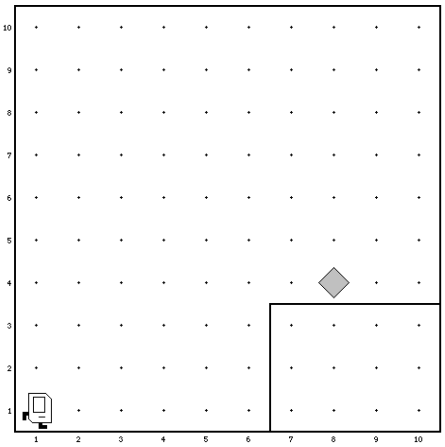
PG 1 Mathematik - Einführung in die Programmierung mit C - Prof. Dr. Markus Heckner

---

Für diese Übung benötigen Sie ein Starterprojekt, das Sie aus *Grips* herunterladen können.
Hinweis: Verwenden Sie bei der Lösung dieses Blatts nur Dinge, die bis zum Zeitpunkt der Aufgabenstellung in Vorlesung und Übung besprochen bzw. im *Reader* zu *Karel the Robot* behandelt wurden. Die Verwendung von Variablen ist beispielsweise nicht zulässig. Halten Sie sich an die Spielregeln, auch wenn Sie schon ein fortgeschrittener Programmierer sind! Sie werden auf jeden Fall lernen Probleme in Teilprobleme zu zerlegen und diese schrittweise zu lösen!
Tipp:
In Grips finden Sie ebenfalls eine *Karel Reference Card*, auf der Sie die Struktur eines Karel-Programms, sowie alle Funktionen, die Karel unterstützt nachlesen können!

Aufgabe 1 : Karel und sein Wohnzimmer

Karels Welt sieht in dieser Aufgabe wie folgt aus:

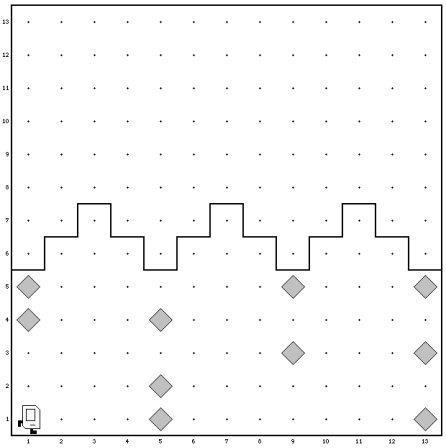


Karel steht in seinem Wohnzimmer (Welt: *LivingRoom.w*) und soll, wie in der Vorlesung, den Beeper auf der Kommode aufsammeln und dann bis zur rechten Wand weitergehen und dort stehen bleiben.
Dieses Beispiel ist einfach, Sie können auf die folgenden Annahmen bauen: Karel startet in der Ecke links unten. Die Welt sieht immer exakt so aus wie in der Abbildung dargestellt, d.h. die Kommode ist gleich hoch und die Größe des Feldes, sowie die Position des Beepers sind immer gleich.
Die Aufgabe besteht daraus, die Kommandos für Karel zu schreiben, um die folgenden Teilaufgaben zu lösen.

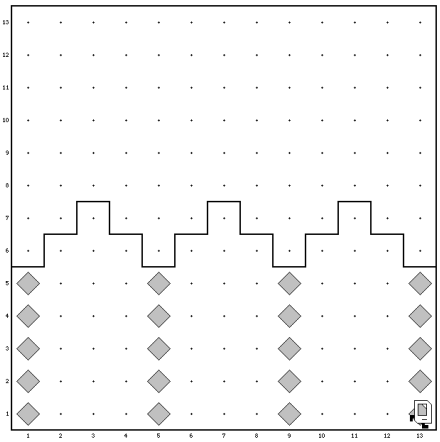
1. Bis zur Kommode laufen
2. Die Kommode seitlich "hochklettern"
3. Bis zum Beeper laufen
4. Den Beeper aufheben
5. Bis zur Wand weitergehen

Hinweise:
Schreiben Sie **eine Funktion für jeden der oben dargestellten Schritte**. Ziel der Übung ist es mit dem dem grundsätzlichen Programmablauf vertraut zu werden und Übung darin zu bekommen, Probleme in kleinere Teilaufgaben zu zerlegen (*Decomposition*!).
Achten Sie darauf, sowohl den Code als auch die Kommentare (z.B. *pre-* und *post-conditions*) in **englisch** zu schreiben: Als Entwickler arbeiten Sie häufig in internationlen Teams mit anderen Entwicklern zusammen, die kein deutsch sprechen. Englisch ist die weltweit anerkannte Standardsprache für Programmiercode. Verwenden Sie aussagekräftige Bezeichnungen für Ihre eigenen Funktionen.

Aufgabe 2 : **Die zerstörten Säulen**
Bei einer Explosion wurden Säulen zerstört, die das Dach eines Gebäudes tragen. Karel wurde angeheuert, um diese Säulen zu reparieren. Die zu reparierenden Säulen bestehen aus *Beepern*, fehlende Stücke muss Karel ersetzen:



Wenn Karel mit seiner Arbeit fertig ist, müssen die Säulen alle intakt sein, wie auf der folgenden Abbildung dargestellt:



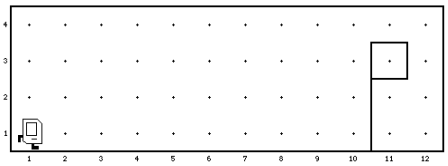
Das Programm muss in der oben dargestellten Welt funktionieren (`DamagedPillar1.w`), aber muss allgemein genug sein, um in jeder Welt zu funktionieren, in der die folgenden Bedingungen gelten:

- Karel startet auf dem Feld (1,1) und blickt nach Osten. Er hat eine unendliche Menge an *Beepern* in seinem Beutel.
- Die Säulen sind immer durch 4 Felder getrennt, d.h. sie stehen auf den Feldern (1,1), (1,5), (1,9) usw.
- Die letzte Säule steht immer direkt vor einer Wand und markiert das Ende des Gebäudes, und damit das Ende der Arbeit für Karel. In der Beispielwelt steht die letzte Säule auf Feld (1,13), aber das Programm muss für eine beliebige Anzahl von Säulen funktionieren.
- Das obere Ende einer Säule wird durch eine Wand begrenzt: Karel kann sich aber nicht darauf verlassen, dass alle Säulen immer 5 Felder hoch sind, und auch nicht darauf, dass alle Säulen in einer Welt die gleiche Höhe haben.
- Die Säulen sind noch teilweise intakt: Wenn die Säule auf einem Feld intakt ist, d.h. ein *Beeper* darauf liegt, dann darf Karel keinen zweiten *Beeper* auf das Feld legen.

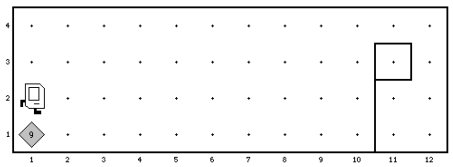
Verwenden Sie die `for`-Anweisung, falls Karel Dinge öfter tun soll und vermeiden Sie damit doppelten Code zu schreiben. Testen Sie Ihr Programm auch mit der Welt `DamagedPillar2.w`.

Aufgabe 3 : **Flag-Karel**

In dieser Aufgabe soll Karel die Distanz bis zu einer in der Welt aufgestellten Flagge messen. Das Ergebnis seiner Berechnung lässt sich an einem Stapel *Beeper* ablesen, den er in der linken unteren Ecke der Welt ablegt. Die Welt vor dem Start des Programms sieht wie folgt aus:



Das Ergebnis zeigt die folgende Abbildung (die 9 Beeper stehen für die Entfernung zwischen Karels Startposition und der Flagge):



Es gelten die folgenden Annahmen:

- Karel steht zu Beginn immer in der linken unteren Ecke der Welt.
- Die Flagge steht immer am unteren Rand der Welt (also auf dem Boden).
- Die Flagge steht immer rechts von Karel.
- Die Flagge kann auch direkt rechts von Karel auf dem Nachbarfeld stehen. In diesem Fall muss das Programm dennoch korrekt funktionieren und Karel darf entsprechend **keinen** Beeper ablegen.

Ihr Programm muss für die im Starterprojekt enthaltenen Welten `flagDistance0.w` (Ergebnis 0), `flagDistance1.w` (Ergebnis: 9) und `flagDistance2.w` (Ergebnis: 24) funktionieren, sowie für alle anderen Welten, die den oben genannten Kriterien entsprechen. Es spielt keine Rolle in welche Richtung Karel am Ende des Programms blickt.

Hinweis: Übungsaufgaben und Konzept basieren zum Teil auf dem Kurs *CS106A: Programming Methodology* der Universität Stanford von Eric Roberts und Mehran Sahami