
Humboldt-Universität zu Berlin
ProMINT-Kolleg


# Arbeitsgemeinschaft Robot Karol

## Version 1.0

Katja Wundermann
wunder@informatik.hu-berlin.de

Michael Rücker
ruecker@informatik.hu-berlin.de

23. Mai 2012

# Inhaltsverzeichnis

# 1 Einleitung

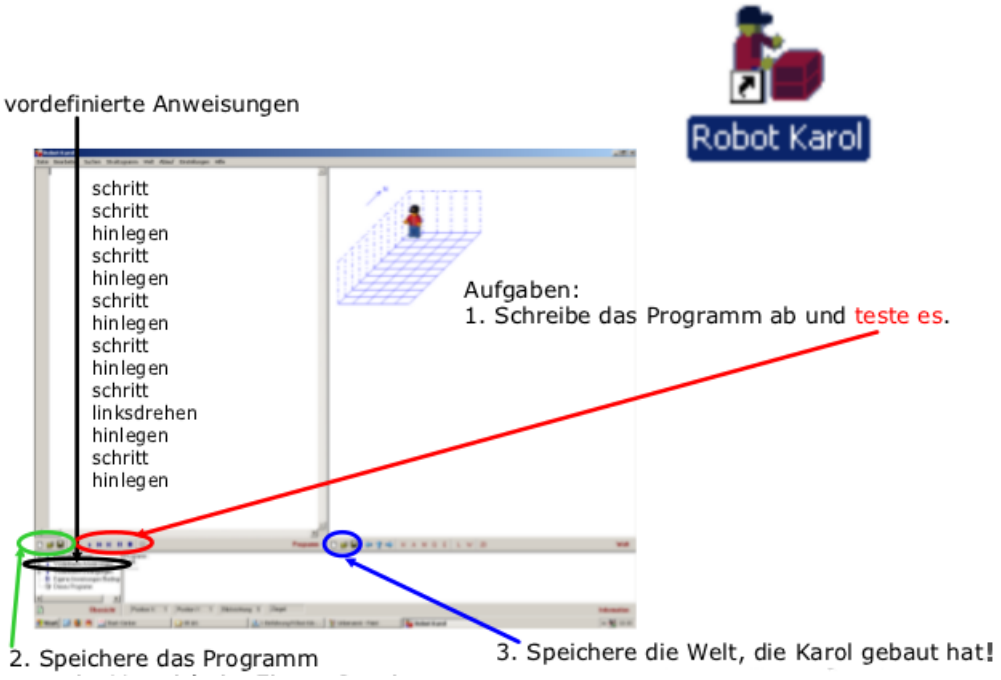
Die hier beschriebene Arbeitsgemeinschaft stellt einen einfachen Einstieg in die Programmierung dar und richtet sich somit an jüngere Schüler, das heißt Schüler der Klassenstufe fünf bis acht. Haben die Schüler keine Programmiervorkenntnisse, kann die Vorbereitung sicherlich in ähnlicher Weise auch für den Informatik-Anfängerunterricht der Wahlpflichtkurse Klasse 9-10 verwendet werden. Der Inhalt ist angedacht für eine ein halbes Jahr dauernde Arbeitsgemeinschaft beziehungsweise einen Kurs bei einer Doppelstunde Unterricht pro Woche.

Hintergrund ist es, anhand des graphischen Programmierwerkzeugs „Robot Karol“ einfache imperative Programmierkonzepte wie Anweisungen, Schleifen und Verzweigungen einzuführen. Etwas untypisch bei der didaktischen Vorgehensweise ist die Einführung der Schleifen vor der Einführung der Verzweigungen, was an gegebener Stelle kurz begründet wird. Der Text richtet sich an Schüler und stellt insbesondere eine Aufgabensammlung dar. Zu den meisten Abschnitten gibt es aber auch didaktische Hinweise und Begründungen, die sich an den Leiter der Arbeitsgemeinschaft beziehungsweise den Lehrer richten.

# 2 Bedienung der Oberfläche

In dieser Arbeitsgemeinschaft sollt ihr lernen, mit Robot Karol zu programmieren. Dazu müssen wir uns aber erst anschauen, wie man mit Robot Karol arbeiten kann, wie man Programme schreibt, ausführt und schließlich speichert. In Fachsprache heißt das, wir machen uns mit der Oberfläche von Robot Karol vertraut.

**Beispiel 1:** Mein erstes Karol-Programm




**Aufgabe 1:**

1. Öffne „Robot Karol“.
2. Schreibe das Programm ab und teste es. Was macht Karol?
3. Speichere das Programm im deinem Homeverzeichnis unter dem Namen „erstesProgramm“.
4. Speichere die Welt, die Karol gebaut hat, in deinem Homeverzeichnis unter dem Namen „ersteWelt“.
5. Verkleinere das Fenster und öffne dein Homeverzeichnis. Finde heraus, welche Endung der Name des Programms hat. Welche Endung hat die Datei mit Karols Welt?

**Bemerkungen zur Vorbereitung:** In diesem Abschnitt geht es darum, Robot Karol kennenzulernen. Die SchülerInnen sollen sich mit der Oberfläche vertraut machen, erste kleine Programme schreiben und diese abspeichern. Auch die Welt, die Robot Karol gebaut hat, kann gespeichert werden. Die beschriebene Einstiegsaufgabe ist geeignet, die Schüler dazu anzuregen.

# 3 Anweisungen

## 3.1 Vordefinierte Anweisungen

In deinem ersten Programm hast du schon einige Anweisungen, wie `schritt`, `hinlegen` oder `linksdrehen` verwendet. Karol kennt aber noch viele weitere Anweisungen. Das sind die sogenannten „vordefinierten Anweisungen“. Eine Liste aller vordefinierten Anweisungen findest du im kleinen Fenster unten links, wenn du auf „vordefinierte Anweisungen“ klickst. Mit diesen Anweisungen kannst du Karol schon eine Menge machen lassen. Probiere es aus!

**Aufgabe 2:**

1. Schreibe ein Programm, durch das Karol selbstständig eine Reihe mit fünf Steinen legt. Speichere das Programm unter dem Namen „Reihe“.

2. Schreibe ein Programm, durch das Karol selbstständig eine Mauer legt, die sieben Steine lang und drei Steine hoch ist. Speichere das Programm unter dem Namen „Mauer“.

3. Schreibe ein Programm, durch das Karol selbstständig ein Schachbrett aus acht mal acht Feldern legt. Speichere das Programm unter dem Namen „Schachbrett“.

**Bemerkungen zur Vorbereitung:** Es ist möglich (unter „Einstellungen“ –¿ „Karol“), im Programm einzustellen wie viele Steine Karol in einem Schritt nach oben oder unten gehen kann. Eine sehr hohe Zahl führt dazu, dass sich Karol frei in der Welt bewegen kann, ohne Rücksicht auf bereits gebaute Strukturen. Eine geringere Zahl fördert unserer Ansicht nach ein räumliches Vorstellungsvermögen und ist zudem realistischer. Ein Mensch (oder auch ein Roboter) kann auch nicht beliebig hohe Stufen steigen.

Karol kann also nur höchstens einen Stein nach oben oder nach unten steigen. Die vielleicht naheliegende Lösung zum Bau der Mauer, immer drei Steine hinzulegen und dann einen Schritt zu machen, funktioniert daher nicht. Wird ein solcher Versuch ausgeführt, bewegt sich Karol so weit wie er kann, dann bricht das Programm ab. Im Fenster rechts unten erscheint eine genaue Fehlermeldung. Weist man die SchülerInnen also nicht im Vorfeld auf das Problem beim Bau der Mauer hin, können die SchülerInnen Karols Fähigkeiten und Grenzen hier selbst entdecken und gleichzeitig lernen, mit Fehlermeldungen umzugehen.

Damit Karol das Schachbrettprogramm legen kann, muss Karols Welt entsprechend vergrößert werden. Unter Umständen sollte dies thematisiert

werden.

Je nachdem, ob der Kurs „nur“ vorbereitenden Charakter hat oder den Einstieg in die weitere Informatikausbildung darstellt, kann man den Begriff „Anweisungen“ als Teil eines Programms weiter thematisieren.

## 3.2 Selbstdefinierte Anweisungen

Robot Karol kann sicherlich schon eine ganze Menge. Doch es gibt auch einige einfache Anweisungen, die er nicht versteht, wie z.B. sich umdrehen. Aber auch andere Dinge muss man ihm erst „kleinschrittig erklären“. Wäre es nicht schön, wenn man ihm zum Beispiel nur ein mal erklären müsste, wie man den Buchstaben `L` legt, und er sich das dann als große Anweisung merkt? Genau das ist das Prinzip von „selbstdefinierten Anweisungen“. Betrachte dazu folgendes Beispiel:

**Beispiel 2:** Anweisung `legeL`

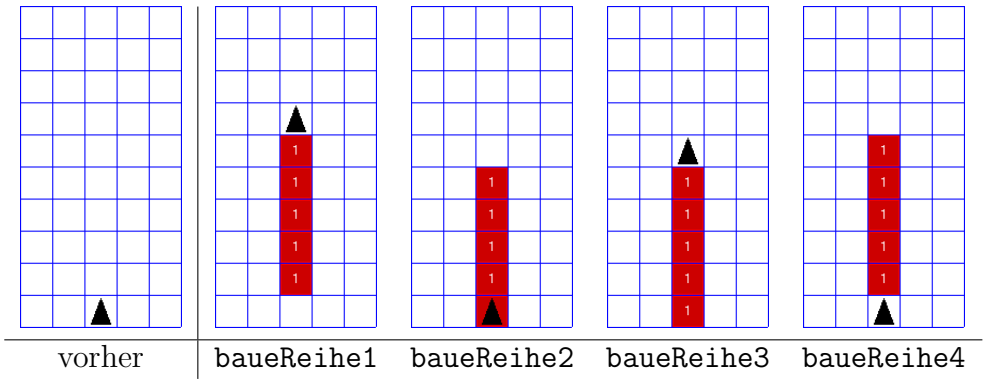
```
Anweisung legeL
  hinlegen
  schritt
  hinlegen
  schritt
  hinlegen
  schritt
  hinlegen
  schritt
  linksdrehen
  hinlegen
  schritt
  hinlegen
  linksdrehen
  schritt
  schritt
  schritt
  schritt
  rechtsdrehen
  schrittjedoch
  schritt
  schritt
  rechtsdrehen
*Anweisung
```

```
// Hauptprogram:
legeL
legeL
legeL
```

**Aufgabe 3:**
Schreibe nun nach dem gleichen Muster weitere Anweisungen und rufe diese im Hauptprogramm auf:

1. Anweisung `umdrehen`
2. Anweisung `SchrittNachRechts`
3. Anweisung `baueReihe` (Länge 5)
4. Anweisung `baueMauer` (Länge 5, Höhe 4)

**Aufgabe 4:**
Schreibe Varianten der Anweisung `baueReihe`, sodass sie jeweils den folgenden Vorher-Nachher-Bildern entspricht:




Wodurch unterscheiden sich die verschiedenen Lösungen? Kennzeichne gleiche Abschnitte und benenne die verschiedenen durch Kommentare.

**Aufgabe 5:**
Lege mit Karols Hilfe deine Initialen.

**Bemerkungen zur Vorbereitung:** Zur Vorbereitung auf das arbeitsteilige Programmieren können alle obigen Aufgaben so erweitert werden wie in Aufgabe 4, dass Karol nach der Abarbeitung an einem bestimmten Punkt stehen soll, z. B. dem Ausgangspunkt. Dies bereitet auf die später benötigten Schnittstellen vor.

## 3.3 Arbeitsteiliges Programmieren

**Aufgabe 6:**
Karol soll Worte und Texte schreiben können:

1. ANANAS
2. DER DIE DAS
3. MEIN NAME IST KAROL
4. AAAA AAAA AAAA AAAA AAAA AAAA

Ihr sollt hier aber in Gruppen arbeiten, das heißt, jedes Paar soll sich nur mit einem Teil der Arbeit beschäftigen. So könnte ein Paar für die Buchstaben `A` und `N`, ein Paar für die Buchstaben `S` und `D` zuständig sein und so weiter. Ein Pärchen könnte die einzelnen Buchstaben im Programm zu den entsprechenden Texten zusammenführen. Bedenkt aber, dass ihr eure Arbeit gut koordinieren müsst. Folgende Schritte und Anregungen sollen euch helfen.

1. Welche Buchstaben muss Karol mit Ziegeln legen können? Schreibe auf dein Blatt die Namen der dazu geeigneten selbst selbstdefinierten Anweisungen.
2. Was muss Karol beachten, wenn er die verschiedenen Buchstaben aneinandersetzt? Zeichne die 2D-Ansicht des Buchstaben `A` ab und trage die Startposition (A-Anfang) und die Schlussposition (E-Ende) ein. Beachte beim Eintragen auch die Blickrichtung. Diese Vereinbarungen nennt man auch Schnittstellenvereinbarungen.
3. Definiere für jeden Buchstaben eine Anweisung.
4. Programmiere Karol so, dass er die vorgegebenen Worte schreibt.

**Bemerkungen zur Vorbereitung:** Für diese Aufgabe bietet es sich an, eine Gruppenarbeit tatsächlich durchzuführen. Nur dann fallen Fehler bei der Schnittstellenvereinbarung wirklich auf. In Anbetracht des Alters der Schüler, sollte die Gruppenarbeit gut angeleitet werden.

Die Aufgaben dieses Abschnittes sind im Anhang als Arbeitsblatt aufgearbeitet.

# 4 Schleifen

## 4.1 Zählschleifen

Sicher ist dir aufgefallen, dass du in vielen Programmen einzelne Schritte aber auch längere Anweisungsfolgen häufig wiederholst. Daher gibt es auch die Möglichkeit Karol eine Anweisungsfolge mehrmals hintereinander ausführen zu lassen. Legst du die Anzahl der Wiederholungen vorher fest, nennt man das eine Zählschleife.

**Beispiel 3:** Zählschleife

```
wiederhole 7 mal
  hinlegen
  schritt
*wiederhole
schritt
```

**Aufgabe 7:**
Was macht das Beispielprogramm? Bearbeite (nach diesem Schema) weitere Aufgaben. Halte deine Quelltexte mit Hilfe von Zählschleifen möglichst kurz.

1. Baue eine Mauer (10 Steine lang, 3 Steine hoch)!
2. Baue einen Quader (3 Steine breit, 6 Steine lang, 2 Steine hoch)!
3. Bedecke den Boden einer 10 mal 10 Karol-Welt!

## 4.2 Bedingte Schleifen

Nicht immer ist es vorher schon möglich oder sinnvoll, anzugeben wie oft Karol eine Anweisung wiederholen muss. Dein Programm zum Bedecken des Bodens wird in einer 10 mal 20 Welt nicht mehr einsetzbar sein. In bedingten Schleifen wird vor jedem Durchlauf eine Bedingung abgeprüft. Nur wenn diese erfüllt ist, wird die Schleife ausgeführt. Eine Reihe mit Ziegeln bis zur nächsten Wand könnte man dann wie folgt bauen:

**Beispiel 4:** Bedingte Schleife

```
wiederhole solange NichtIstWand
  hinlegen
  schritt
*wiederhole
```

Bei Robot Karol gibt es schon eine Reihe vordefinierter Bedingungen wie beispielsweise `IstWand` oder `NichtIstWand`. Informiere dich im linken unteren Fenster über weitere vordefinierte Bedingungen, die du verwenden kannst.

**Aufgabe 8:**
Mit bedingten Schleifen, kann man nun Programme schreiben, die universell, das heißt in möglichst vielen Karol-Welten, einsetzbar sind. Übe dies in folgenden Aufgaben!

1. Karol soll am äußersten Weltrand immer im Kreis gehen.

2. Bedecke den Boden deiner Karol-Welt vollständig.

**Bemerkungen zur Vorbereitung:** In diesem Konzept werden, entgegengesetzt zum häufig üblichen Weg, die Schleifen vor den Verzweigungen eingeführt. In ersten Karol Programmen werden hauptsächlich Schleifen benötigt. Verzweigungen braucht man insbesondere, um abzuprüfen, ob ein weiterer Schleifendurchlauf notwendig ist. Dies wird durch bedingte Schleifen aber vollständig abgedeckt. Das heißt, mit bedingten Schleifen können die Schüler auch ohne Verzweigungen schon sehr viel programmieren, mehr als andersherum. Das rechtfertigt das Vorgehen, hier erst die Schleifen, dann die Verzweigungen einzuführen.

# 5 Verzweigungen

Manchmal möchte man auch Bedingungen abprüfen und dann nur eine Anweisung ausführen, ohne gleich eine Schleife zu programmieren. Dazu verwendet man Verzweigungen. Außerdem hat man bei Verzweiungen die Möglichkeit, alternative Abläufe zu programmieren. Eine Verzweigung hat folgende Struktur:

**Beispiel 5:** Verzweigung

```
wenn NichtIstWand
  dann
    hinlegen
    schritt
  sonst
    linksdrehen
    hinlegen
    schritt
*wenn
```

**Aufgabe 9:**

1.a Karol soll ein Feld (4 mal 5) ablaufen. Dabei soll er alle Felder ohne Marken mit Marken belegen und auf allen Feldern mit Marke die Marke löschen.

1.b Erweitere das Programm aus 2a so, dass es auch auf beliebig großen Feldern funktioniert.

2. Schreibe eine Anweisung mit der Karol von einer beliebigen Position in der Welt an die Startposition zurückkehrt. Teste die Anweisung mit einem älteren Karol-Programm.

**Aufgabe 10:**
Nenne Unterschiede zwischen der Verzweigung und der bedingten Schleife. Kann man manchmal beides verwenden? Wenn ja, wann?

**Bemerkungen zur Vorbereitung:** Um die Verzweigungen besser von den Schleifen abzugrenzen, ist es sinnvoll, Aufgaben zu wählen, die man nicht oder nur ungünstig mit bedingten Schleifen lösen kann.

Diesen Unterschied kann man in einem theoretischen Teil auch mit den Schülern besprechen, z. B. mithilfe der obigen Fragestellung oder auch in einem Unterrichtsgespräch oder einer anderen didaktischen Methode.

Sicherlich kann der sonst-Teil der Verzweigung auch weggelassen werden. Dies kann mit den Schülern diskutiert werden.

# 6 Labyrinthsuche

Das Navigieren durch ein Labyrinth ist eine komplexe Aufgabe, bei der Karol immer wieder sehr verschiedene Probleme lösen muss. Zur Orientierung und Bewegung im Labyrinth sind Verzweigungen und bedingte Schleifen unabdingbar. Oft reichen dazu die vordefinierten Bedingungen nicht mehr aus.

## 6.1 Eigene Bedingungen

Ihr habt bei Verzweigungen und Schleifen bereits vordefinierte Bedingungen verwendet. Man kann jedoch auch eigene Bedingungen definieren. Eine Bedingung ist im Prinzip eine Anweisung mit einer Ausgabe und beinhaltet oft eine Verzweigung. Die Ausgabe kann dabei entweder `wahr` (die Bedingung ist erfüllt) oder `falsch` (die Bedingung ist nicht erfüllt) sein:

**Beispiel 6:** Bedingung

```
Bedingung NichtIstWesten
  wenn IstWesten
    dann falsch
    sonst wahr
  *wenn
*Bedingung
```

**Aufgabe 11:**

1. Schreibe die Bedingung `RechtsIstWand`, die überprüft, ob rechts neben Karol eine Wand ist.

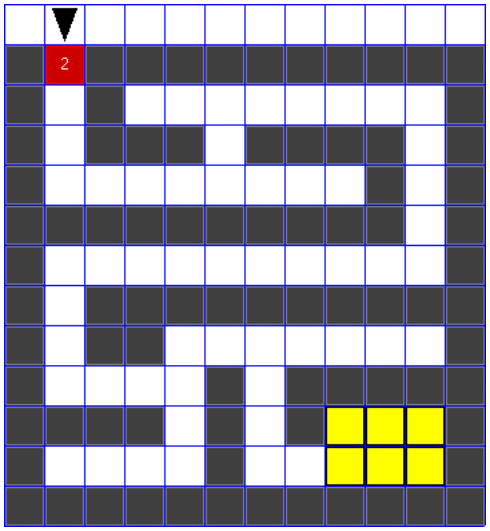
2. Schreibe die Bedingung `ImWestenIstWand`, die überprüft ob westlich von Karol eine Wand ist.

**Bemerkungen zur Vorbereitung:** Bei Bedingungen ist es in der Regel noch viel wichtiger als bei Anweisungen, darauf zu achten, wo und wie Karol nach der Abarbeitung steht. Bei der Bedingung `RechtsIstWand` ist es sehr einfach, Karol im Anschluss durch eine Linksdrehung wieder in die Ausgangposition zu bringen. Bei `ImWestenIstWand` ist dies nicht so einfach, da zuvor mithilfe einer bedingten Schleife eine unbekannte Anzahl von Drehungen ausgeführt werden musste.

## 6.2 Die Grabkammer des Pharaos

Karol sucht in einer alten ägyptischen Pyramide nach der Grabkammer. Der Eingang ist jedoch durch Ziegel verschlossen und dahinter befindet sich ein Labyrinth. In dem Labyrinth kann man nicht im Kreis gehen und die Grabkammer ist durch Markierungen gekennzeichnet. Kannst du Karol helfen, den Schatz der Pharaonen zu finden?

**Beispiel 7:** Pyramide




**Aufgabe 12:**

1. Schreibe ein Programm, mit dem Karol in einer beliebigen Pyramide die Grabkammer findet.

2. Erweitere dein Programm aus 1., sodass Karol den Schatz einsammelt (die Markierungen in der Grabkammer aufnimmt) und danach wieder nach draußen findet.

**Bemerkungen zur Vorbereitung:** Da das Labyrinth keine Zyklen enthält, führt die sogenannte Rechte-Hand-Regel hier zum Erfolg. Karol kann sich einfach immer an der rechten (oder auch linken) Wand entlangbewegen, bis er zum Ziel gefunden hat. Es ist ratsam, verschiedene Pyramiden anzufertigen, sodass die SchülerInnen ihre Programme auf Allgemeingültigkeit testen können.

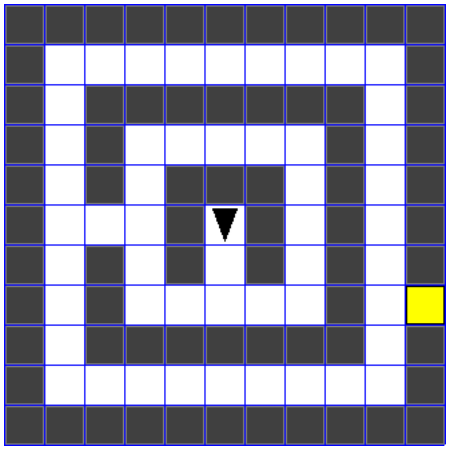
Zudem bietet es sich an (je nach Lerngruppe), den historischen Hintergrund der Pyramiden, der Pharaonen und ihrer Grabkammern zu besprechen.

Warum waren die Grabkammern so reich geschmückt? Auch Grabräuber und Plünderer könnten diskutiert werden. Sollte Karol überhaupt in eine Pyramide einbrechen und den Schatz stehlen?

## 6.3 Das Labyrinth des Minotaurus

Karol ist im Labyrinth des Minotaurus gefangen! Das Labyrinth ist ringförmig aufgebaut. Jeder Ring besitzt genau eine Öffnung und zwei Öffnungen liegen nie genau übereinander. Der Ausgang ist durch eine Markierung gekennzeichnet. Karol muss nun aus dem Innern des Labyrinths nach außen finden. Kannst du ihm dabei helfen?

**Beispiel 8:** Das Lybrinth des Minotaurus



**Aufgabe 13:**
Schreibe ein Programm, mit dem Karol aus einem beliebigen Minotaurus-Labyrinth herausfindet.

**Bemerkungen zur Vorbereitung:** Die Rechte-Hand-Regel aus der Pyramide funktioniert hier nicht, da Karol im Kreis laufen würde. Die Lösung ist es, Karol zuerst nach links, dann nach rechts und schließlich geradeaus gucken zu lassen und jeweils in die erste freie Richtung zu gehen. Eigene Bedingungen sind hier sehr nützlich. Auch hier empfehlen sich verschiedene Labyrinthe zum Testen der Programme.

Wie schon bei der Pyramide bietet es sich (je nach Lerngruppe) auch hier an, den mythologischen Hintergrund des Minotaurus zu besprechen. Was ist ein Minotaurus und was ist die Geschichte hinter dem Labyrinth?