Dr. Michael J. Winckler
IWR, Raum 302
INF 368
69120 Heidelberg
Michael.Winckler@iwr.uni-heidelberg.de
http://www.iwr.uni-heidelberg.de/teaching/Mathe-Star/

Arbeitsblatt 07 *Utilities zum Datentyp* `maze` (Version 1.02)

Theorie

Übungsziele: Ziel dieses Übungsblattes ist es, auf Basis der bisher erlernten Techniken einige zunehmend komplexe Aufgabenstellungen zu implementieren. Dabei gehen wir von dem für die Irrgärten eingeführten komplexen Datentypen `maze` aus und werden weitere Routinen umsetzen, um diesen Datentypen sinnvoll einzusetzen und in immer neuen Funktionen zu bearbeiten.

```
struct maze_s {
  int width   = 0;
  int height  = 0;

  int **field = NULL;
};
typedef struct maze_s maze;
```

Übungen

Übung 7.1: Komplexe Datentypen

Definiere durch

```
struct position_s {
  int x;
  int y;
};
typedef struct position_s position;
```

einen komplexen Datentypen, der eine einzelne Stelle innerhalb eines Irrgartens beschreiben kann.

Schreibe eine Routine `MazeDistance()`, der du zwei Positionen übergibst und die den minimalen Abstand (in Schritten) zwischen diesen beiden Positionen berechnet.

Anm.: Der hier verwendetete Abstandsbegriff geht auf die *Manhattenmetrik* zurück.

Übung 7.2: Verbinden zweier Positionen I

Eine häufig anzutreffenden Aufgabenstellung ist die Verbindung von zwei Punkten innerhalb des Irrgartens durch einen Gang. Schreibe dazu eine Routine, der du einen Zeiger auf ein `maze` sowie zwei Positionen vom Typ `position` übergibst, und die diese beiden Punkte mit einem Gang verbindet. Dazu ist es am einfachsten, vom Ausgangspunkt zuerst waagerecht auf die gleiche Spalte wie die Zielkoordinate zu kommen und sich dann senkrecht zum Ziel “durchzugraben”.

Was passiert, wenn du die beiden Zielpunkte vertauscht und dann nochmals die Verbindungsroutine aufrufst?

Übung 7.3: Verbinden zweier Positionen II

Eine kunstvollerer Verbindung wäre ein etwas unregelmässig geschlängelter Gang: Wähle beim Verbinden zufällig aus, ob der jeweils nächste Verbindungsschritt waagerecht oder senkrecht erfolgen soll.

Implementiere diesen Algorithmus und verfeinere ihn solange, bis er möglichst effizient arbeitet. Beispielsweise kannst du die Zufallsschritte abbrechen, wenn `x`- oder `y`-Koordinate schon übereinstimmen, da dann nur noch Schritte in eine Richtung möglich sind.

Übung 7.4: Verbinden zweier Positionen II

Um noch kunstvollerer Verbindungen zu machen, kannst du vor dem Verbinden zweier Positionen eine zufällige Zwischenstation erzeugen, an der der Gang vorbeikommen muss.

1. Schreibe zunächst eine Routine, die testet, ob ein Irrgarten überhaupt noch Mauern enthält.
2. Erzeuge danach eine Routine, die in einem Irrgarten ein zufälliges Mauerfeld sucht.
3. Baue schliesslich aus den nun vorhandenen Utilities eine weitere Funktion, die zwei Positionen über eine zufällige dritte Zwischenposition miteinander verbindet, an der sich zuvor noch eine Mauerposition befindet. Denke daran, auch das Feld der Zwischenposition zu leeren.

Übung 7.5: Objekte und Personen

Für das weitere Programmhandling ist es besser, bewegliche Objekte (wie z.B. Waffen, Gegenstände etc.) nicht im Irrgarten selbst, sondern in einer eigenen Gegenstandsliste zu implementieren.

Erzeuge einen geeigneten Datentyp für einen Gegenstandsliste und fülle den Irrgarten mit einigen zufälligen Gegenständen. Ebenso kannst du eine Personenliste für den Irrgarten anlegen. Jeder Gegenstand und jede Person kennt eigenen Namen, Darstellungszeichen und aktuelle Position.

Die Zeichenroutine bekommt den Irrgarten und die zwei Listen. Daraus erzeugt sie eine Ansicht des Irrgartens mit Gegenständen und Personen. Ein Gegenstand wird dabei von einer Person am gleichen Ort verdeckt (und nicht umgekehrt!). Implementiere diese erweiterete Darstellungsfunktion. Um hier nicht zuviel Zeit zu vergeuden, kann man zusammen mit dem eigentlichen Feld ein zweites Feld gleicher Grösse anlegen, das von allen Routinen als Zwischenspeicher verwendet werden kann. Dazu ist es zweckmßsig, die Datenstruktur entsprechend zu erweitern.

Diskutiere mit deinen Nachbarn über die Vorzüge und Nachteile der so veränderten Datenstruktur.