Programmieren C: Backtracking: Das 8-Damen-Problem

Klaus Kusche

Das 8-Damen-Problem lautet wie folgt: *„Wie stelle ich* **n** *Damen auf ein* **n*n** *großes Schachbrett, sodass keine zwei Damen einander bedrohen, d.h. sodass in jeder Zeile und jeder Spalte* ***genau*** *eine Dame und in jeder Diagonale* ***höchstens*** *eine Dame steht?“*

Versuche, ein Programm zu schreiben, das *alle* Lösungen des 8-Damen-Problems für eine beim Programmstart auf der Befehlszeile anzugebende Brettgröße (bis maximal 30, definiere 30 als Konstante) berechnet und ausgibt.

Im Unterschied zu den Folien empfehle ich, das *Brett und seine Größe* nicht allen Funktionen als Parameter zu übergeben, sondern als *globale Variablen* zu deklarieren (das macht das Programm übersichtlicher und schneller):

```
// Seitenlänge des Brettes = Anzahl der zu setzenden Damen
int anzahl;
// brett[zeile] speichert die Nummer der Spalte (0...anzahl-1),
// in der die Dame in der jeweiligen Zeile (0...anzahl-1) gerade steht
// Das Brett wird von oben nach unten gefüllt (Zeile 0, Zeile 1, ...)
int brett[MAX_ANZAHL];
```

Weiters brauchst du *4 Funktionen*:

1. **`main`**:

   `main` speichert die Anzahl der Damen von der Befehlzeile in der globalen Variablen **`anzahl`** (und prüft dabei, ob sie zwischen 1 und 30 liegt) und ruft dann die rekursive Funktion für Zeile 0 auf.

2. Die *rekursive Funktion* zum Setzen der Damen:

   Sie funktioniert genau so wie in der Theoriestunde besprochen und auf den Folien angegeben.

3. Die Funktion **`ok`** zum *Prüfen einer Position*:

   Wenn diese Funktion aufgerufen wird, um die Position der Dame in Zeile Nummer **`zeile`** zu prüfen, dann sind die *Damen in den kleineren Zeilen schon richtig gesetzt*, und in den größeren Zeilen sitzt noch *keine* Dame.

   Man muss daher die angegebene Position in einer *Schleife* mit allen gespeicherten Positionen in den *kleineren Zeilen* vergleichen. Wenn man dabei eine Dame findet, die in *derselben Spalte oder derselben Diagonale* sitzt, liefert die Funktion **`false`**, hat die Schleife alle Zeilen ohne „Treffer“ geprüft, endet die Funktion mit **`true`**.

   Ob zwei Damen in derselben Diagonale stehen, prüft man, indem man die *Differenz der Zeilen* und die *Differenz der Spalten* berechnet: Ist Zeilendifferenz gleich Spaltendifferenz oder Zeilendifferenz gleich minus Spaltendifferenz, so ist die Diagonale dieselbe.

4. Die *Ausgabe-Funktion*:

   Im einfachsten Fall gibt sie in einer Schleife die *Spalten aller Damen* aus oder „zeichnet“ beispielsweise mit **`()`** zeilenweise eine *richtig eingerückte Dame*.

Vor der Anzeige jeder Lösung könntest du mit **`system("cls");`** (Windows) bzw. **`system("clear");`** (Linux, aus dem Ncurses-Paket) das *Terminalfenster löschen*.

Nett wäre eine *Verzögerung* nach dem Anzeigen jeder Lösung. Dafür kannst du entweder die SDL mitlinken und mein **`sdlMilliSleep`** aus meinem **`sdlinterf.h`** verwenden, oder du nutzt die Systemfunktionen **`Sleep`** (Windows: **`windows.h`**, Parameter: Millisekunden) oder **`usleep`** (Linux: **`unistd.h`**, Parameter: Mikrosekunden).

Erweiterung 1: Grafische Anzeige

Wir wollen die Lösung grafisch anzeigen. Dazu sind folgende Umbauten nötig:

- Wir müssen meinen Header **`sdlinterf.h`** inkludieren und im **`main`** vor dem ersten Zeichnen **`sdlInit();`** und nach der rekursiven Funktion **`sdlExit();`** aufrufen.
- Ich habe die folgenden Werte global deklariert, damit ich sie an allen Stellen, die etwas zeichnen, zur Verfügung habe:

```
// Größe eines Feldes am Spielbrett in Pixeln
// (muss ungerade sein, damit das Feld einen Mittelpunkt hat)
int feldgr;
// x- und y-Koordinate des Mittelpunktes des linken oberen Feldes
int links, oben;
```

- Nach dem Einlesen von **`anzahl`** berechnet **`main`** diese Werte wie folgt:

```
// lass auf allen Seiten min. 4 Pixel Rand außerhalb des Schachbrettes
feldgr = ((SDL_X_SIZE >= SDL_Y_SIZE) ? SDL_Y_SIZE : SDL_X_SIZE) - 9;
feldgr /= anzahl;
// mach feldgr ungerade, damit die Felder einen Mittelpunkt haben
if (feldgr % 2 == 0) { --feldgr; }
// Mittelpunkt-Position: Halber Rand + halbe Feldgröße
links = (SDL_X_SIZE - feldgr * anzahl) / 2 + feldgr / 2;
oben = (SDL_Y_SIZE - feldgr * anzahl) / 2 + feldgr / 2;
```

  Dann zeichnet **`main`** ein leeres Schachbrett (Funktion **`felder`**) und zeigt es an.

- Diese *Funktion* **`felder`** (*ohne* Parameter und Returnwert) ist neu zu schreiben. Wie du deine *Schleifen* dafür baust und aus **`feldgr`**, **`links`** und **`oben`** die richtigen Parameter-Werte für **`sdlDrawLine`** berechnest, überlasse ich dir.

  Ich habe *ein Pixel breite Linien an allen 4 Seiten* der Felder gezeichnet, d.h. zwischen zwei Feldern sind bei mir jeweils zwei Linien, am Spielfeldrand ist hingegen nur eine solche Linie, innen sind alle Felder schwarz.

- Die *Ausgabe-Funktion* ändert sich komplett:
  - Wir zeichnen in einer *Schleife für jede einzelne Dame* einen Kreis: **`sdlDrawCircle(`***xMittel*`,` *yMittel*`,` *xRad*`,` *yRad*`,` *r*`,` *g*`,` *b*`);`** Den *Mittelpunkt* berechnet man, indem man zum Mittelpunkt links oben *Zeilen mal bzw. Spalten mal die Feldgröße dazurechnet*, und der *Radius* ist in beide Richtungen die *halbe Feldgröße minus drei Pixel Rand*.
  - Vor der Schleife *löscht* man die alte Anzeige komplett (**`sdlSetBlack();`**) und zeichnet mit **`felder`** ein neues, *leeres Schachbrett*. Nach der Schleife *aktualisiert* man die Anzeige (**`sdlUpdate();`**) und gibt etwas *Zeit* zum Anschauen (**`sdlMilliSleep(`***ms*`);`**).

Tipp: Durch Ändern der **#define**'s von **SDL_X_SIZE** und **SDL_Y_SIZE** in **sdlinterf.h** kannst du die Größe des Grafikfensters quadratisch machen und an deinen Bildschirm anpassen. Du musst aber nach der Änderung sowohl dein Programm als auch **sdlinterf.c** frisch kompilieren (es gibt einen Button „Alles neu bauen" im CodeBlocks).

Erweiterung 2: Life-Anzeige

Wir wollen nicht nur die fertige Lösung anzeigen, sondern *jedes Setzen oder Wegnehmen einer Dame*, damit man das Suchen der Lösungen verfolgen kann.
Dazu sind folgende Änderungen notwendig:

- Die Anzeige wird *nie ganz gelöscht*: **sdlSetBlack();** fällt ersatzlos weg.
  Die *Linien für die Felder* werden daher *nur einmal* ganz am Anfang gezeichnet.
- Das *Zeichnen* einer Dame (**sdlDrawCircle(**…**)** mit **sdlUpdate();** und **sdlMilliSleep(***ms***);** danach) wandert *in die rekursive Funktion*:
  Sobald eine Dame einen *zulässigen Platz* hat wird sie gezeichnet (*nach* **if (ok(**…**))** und *vor* dem Setzen der nächsten Dame).
- Unmittelbar *nachdem* die restlichen Damen alle durchprobiert worden sind muss die *Dame wieder weggelöscht* werden:
  Wir machen genau dieselben drei **sdl**-Aufrufe, aber mit *schwarzem* Kreis.
- Die *Anzeige-Funktion* hat nichts mehr zu tun (außer mit einem längeren **sdlMilliSleep(***ms***);** zu warten), denn die Damen sind ja schon alle gezeichnet.