OSTBAYERISCHE TECHNISCHE HOCHSCHULE REGENSBURG
INFORMATIK UND MATHEMATIK

Übungsaufgaben Blatt 7 - Arrays

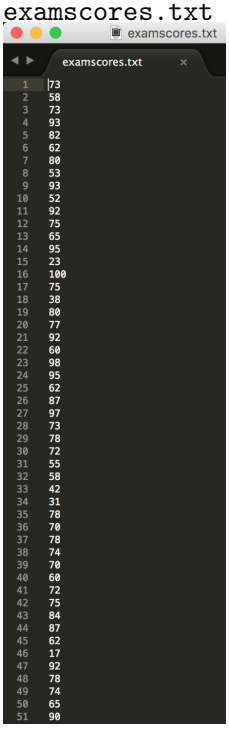
Programmieren 1 - Einführung in die Programmierung mit C - Prof. Dr. Ruben Jubeh

---

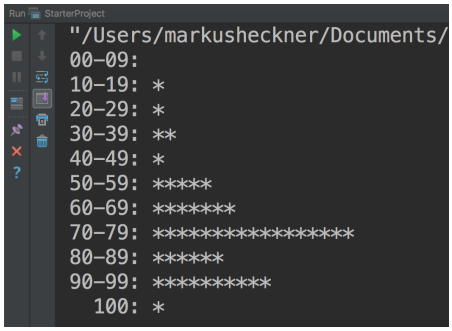
Aufgabe 1 : Histogramme und Dateien: Notenhäufigkeit

Schreiben Sie ein Programm, das eine Liste von Klausurergebnissen (in Punkten) aus einer Datei `examscores.txt` ausliest. Diese Datei enthält pro Zeile immer ein Klausurergebnis. Geben Sie ein Histogramm dieser Punktzahlen aus, unterteilt in die folgenden Bereiche: 0–9, 10–19, 20–29, und wo weiter, bis zu einem einzigen Bereich, der nur den Wert 100 enthält.

`examscores.txt`



Falls `examscores.txt` die links dargestellten Daten enthält, sollte das Ergebnis so wie auf der Abbildung dargestellt aussehen:



Der folgende Code zeigt das zeilenweise Auslesen einer zuvor geöffneten Datei:

```
while (1) {
  char *line = readLine(infile);
  if (line == NULL) {
      break;
  }
  int num = atoi(line);
```

```
}
```

Mit der ebenfalls dort angegebenen Funktion `atoi()` können Sie die ausgelesene Zeile in eine Ganzzahl umwandeln.
Hinweis: Für diese Aufgabe benötigen Sie die Datei `examscores.txt`, die Sie sich aus Grips herunterladen können. Sie müssen diese Datei in den selben Ordner legen, in dem Ihr kompiliertes Programm liegt.
In einem CLion Starterprojekt ist das der Ordner: `cmake-build-debug`.

Erweiterung: Erweitern Sie Ihr Programm so, dass die folgende Fehlermeldung ausgegeben wird, falls die Datei `examscores.txt` ungültige Werte (d.h. Werte kleiner als 0 oder größer als 100 enthält): `ERROR - Score out of range!` Das Programm soll sich anschließend sofort beenden. Fügen Sie der Datei `examscores.txt` ein paar Zeilen mit ungültigen Werten hinzu und testen, ob der Fehler korrekt ausgegeben wird, und sich das Programm korrekt beendet.

Aufgabe 2 : swapEnds

Schreiben Sie eine Funktion, die als Parameter ein Array aus Ganzzahlen erhält. Die Funktion soll das erste und letzte Element des Arrays vertauschen. Verändern Sie direkt das übergebene Array (*Call by reference*). Die Länge des Arrays ist mindestens 1.

Die Signatur Ihrer Funktion sollte wie folgt aussehen:

```
void swapEnds(int a[], int size) {

  /* Your code goes here. */

}
```

Die folgenden Zeilen zeigen verschiedene Aufrufe und die entsprechenden Ausgaben des Programms:
swapEnds({ 1, 2, 3, 4}, 4) → {4, 2, 3, 1}
swapEnds({1, 2, 3}, 3) → {3, 2, 1}
swapEnds({8, 6, 7, 9, 5}, 5) → {5, 6, 7, 9, 8}
swapEnds({1}, 1) → {1}

Rufen Sie die Funktion aus `main` auf und geben Sie nach dem Vertauschen die Inhalte des Arrays auf der Kommandozeile aus. Testen Sie Ihre Implementierung, sodass alle Aufrufe korrekt wie oben dargestellt funktionieren.

Aufgabe 3 : *lucky13*

Gegeben ist ein Array aus Ganzzahlen. Schreiben Sie eine Funktion, die 1 zurückgibt, falls das Array keine 1er oder 3er enthält. Geben Sie ansonsten 0 zurück:
lucky13({0, 2, 4}, 3) → 1
lucky13({1, 2, 3}, 3) → 0

lucky13({1, 2, 4}, 3) → 0

```
int lucky13(int a[], int size) {

   /* your code goes here */

  return 1;
}
```

Kompilieren und testen Sie Ihr Programm anschließend.

Aufgabe 4 : **Arrays mit einer *unlucky1***

Wir nehmen an, dass eine 1 in einem Array, auf die direkt eine 3 folgt, eine *unlucky1* ist. Erstellen Sie eine Funktion, die 1 zurückgibt, falls das übergebene Array eine *unlucky1* an der ersten, zweiten oder vorletzten Stelle enthält.

Die Funktion verhält sich wie folgt:
unlucky1({1, 3, 4, 5}, 4) → 1
unlucky1({2, 1, 3, 4, 5}, 5) → 1
unlucky1({1, 1, 1}, 3) → 0
unlucky1({1, 1, 1, 4, 5, 1, 3}, 7) → 1

`unlucky1()`, die Sie implementieren sollen:

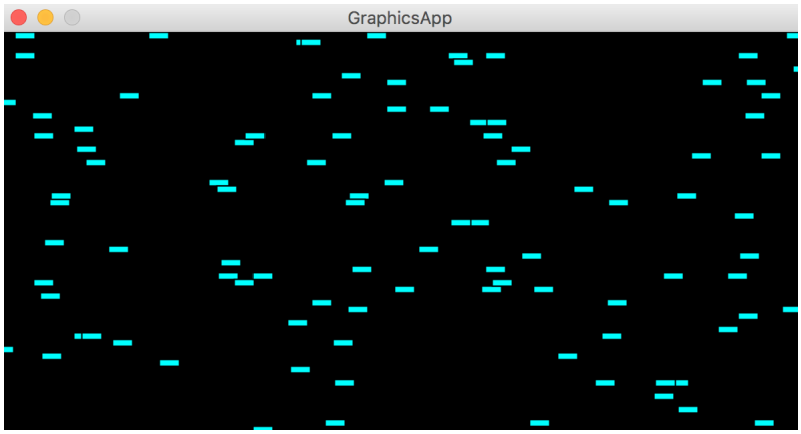
```
int unlucky1(int a[], int size) {

   /* your code goes here */

  return 0;
}
```

Kompilieren und testen Sie Ihr Programm anschließend.

Aufgabe 5 : **Car-Simulator**

Erstellen Sie ein Programm, das den Verkehr auf einer mehrspurigen Straße animiert (vgl. dazu Video aus Grips, neben dieser Aufgabenstellung):

- Alle Fahrzeuge starten am linken Rand der Zeichenfläche.
- Die Fahrzeuge fahren in Spuren, jede Spur ist so hoch wie die Fahrzeuge (die Fahrzeughöhe ist konstant).
- Auf einer Spur können mehrere Fahrzeuge mit unterschiedlichen Geschwindigkeiten fahren.
- Fährt ein Fahrzeug rechts aus dem Bild heraus, so wird seine Position wieder auf den Anfang der selben Spur gesetzt und die Geschwindigkeit wieder zufällig berechnet.
- Die Geschwindigkeit des Fahrzeugs wird ebenfalls zufällig bestimmt (zwischen `1` und `MAX_SPEED` Pixeln pro Animationsschritt).



Hinweise zur Bearbeitung:

- Sehen Sie sich die beiden Arrays `cars` und `carSpeeds` im folgenden Codebeispiel an. Welche Daten werden dort verwaltet? Wann werden diese im Programm befüllt, wann wieder ausgelesen?
- Zur Berechnung der zufälligen Geschwindigkeit und der zufälligen Spur der Fahrzeuge steht Ihnen die Library *random.h* zur Verfügung. Suchen Sie sich dort selbst die passenden Funktionen und integrieren diese in Ihr Programm
(vgl. hier `http://fbim.fh-regensburg.de/~hem38149/doc/cslib/`)

Gegeben ist der folgende Code:

```
#include "gobjects.h"
#include "gwindow.h"

/* Constants */
#define CAR_NUM 100
#define CAR_WIDTH 15
#define CAR_HEIGHT 5
#define WIDTH 600
#define HEIGHT 300
#define PAUSE_TIME 20
#define CAR_COLOR "CYAN"
#define MAX_SPEED 10

GRect cars[CAR_NUM];
int carSpeeds[CAR_NUM];

GWindow gw;

void setupCars() {
    // todo: create cars
}

void moveCars() {
    // todo: move cars
}

void drive() {
    while (1) {
        moveCars();
        pause(PAUSE_TIME);
    }
}

void setupCanvas() {
    gw = newGWindow(WIDTH, HEIGHT);
    fillRect(gw, 0, 0, WIDTH, HEIGHT);  //draws a black
       background
}

int main() {
    setupCanvas();
    setupCars();
    drive();
    return 0;
}
```