OSTBAYERISCHE TECHNISCHE HOCHSCHULE REGENSBURG
INFORMATIK UND MATHEMATIK

Übungsaufgaben Blatt 10 - Komplexe Datentypen

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

---

Aufgabe 1 : Syntax von Strukturen

Welche der folgenden Definitionen für eine Struktur von Punkten im zweidimensionalen Raum ist korrekt? Was stimmt an den anderen Definitionen nicht?

```c
struct point (
  double x, y
)
```

struct point double x, double y ;

```c
struct point {
  double x;
  double y
}
```

```c
struct point {
  double x;
  double y;
};
```

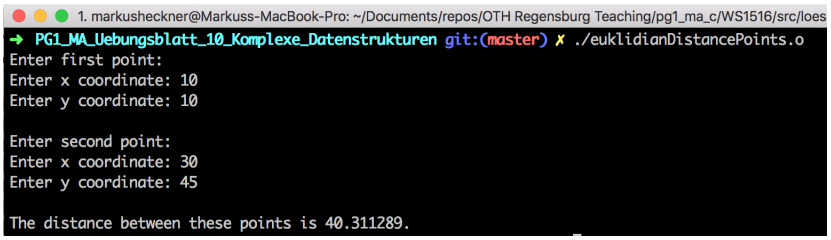
```c
struct point {
  double x;
  double y;
}
```

Aufgabe 2 : Euklidischer Abstand

Schreiben Sie eine Funktion `distance()`, die den euklidischen Abstand zwischen zwei Punkten berechnet. Verwenden Sie dazu die Struktur `point` aus Aufgabe 1.

```c
double distance(struct point p, struct point q){
  return 0.0;  /* Your code goes here. */
}
```

Lesen Sie zwei Punkte vom Nutzer ein, berechnen die euklidische Distanz mithilfe der erstellten Funktion und geben das Ergebnis anschließend an den Nutzer wieder aus. Eine mögliche Ausgabe Ihres Programms könnte wie folgt aussehen:



Aufgabe 3 : Autos mit structs

In dieser Aufgabe sollen Eigenschaften von Autos per struct repräsentiert werden.

- Erstellen Sie zunächst ein struct car. Ein Auto hat folgende Eigenschaften:
  - Marke (VW, PORSCHE oder LAMBORGHINI)
  - Maximalgeschwindigkeit
  - Antiblockiersystem (enthalten/nicht enthalten)
  - Türenanzahl
  - Preis eines Autos

  Entscheiden Sie selbst, wie Sie diese Eigenschaften umsetzen.
- Erstellen Sie eine Funktion `newCar`, welche ein struct car liefert, dessen Eigenschaften sie per Parameter erhält. Diese Funktion ist also in der Lage ein Auto zu bauen, wenn Sie ihr die notwendigen Informationen übergeben. Achtung: Den Preis kennen Sie nicht, Sie müssen sich den Preis aus den anderen Angaben schätzen (siehe nächster Punkt).
- Nun soll der Preis eines Autos geschätzt werden. Schreiben Sie dafür eine Funktion, die ein struct car als Parameter erhält und den Preis folgendermaßen schätzt:
  1. Nimm die Maximalgeschwindigkeit multipliziert mit 50
  2. Multipliziere mit der Türenanzahl
  3. Wenn das Auto ABS hat, addiere 5000 hinzu.
  4. Ist die Marke Porsche, multipliziere mit 2, bei Lamborghini mit 20

  Setzen Sie jetzt die Eigenschaft Preis jedes Autos auf den geschätzten Preis.

Zum Testen der Funktionen: Erstellen Sie in Ihrer main Funktion ein Array mit 3 oder mehr Beispielautos. Berechnen Sie anschließend den Gesamtpreis aller Fahrzeuge und geben diesen Preis auf der Kommandozeile aus. Geben Sie auch jedes Auto inkl. der Eigenschaften an den Benutzer aus. Eine Ausgabe könnte wie auf der folgenden Abbildung dargestellt aussehen:

```
------------------------------------------------
Total price for all cars: EUR 827500
------------------------------------------------
Porsche
Max speed: 250
ABS (0=no/1=yes): 1
Number of doors: 2
Prize: EUR 60000

Lamborghini
Max speed: 300
ABS (0=no/1=yes): 1
Number of doors: 2
Prize: EUR 700000

VW
Max speed: 250
ABS (0=no/1=yes): 1
Number of doors: 5
Prize: EUR 67500
```

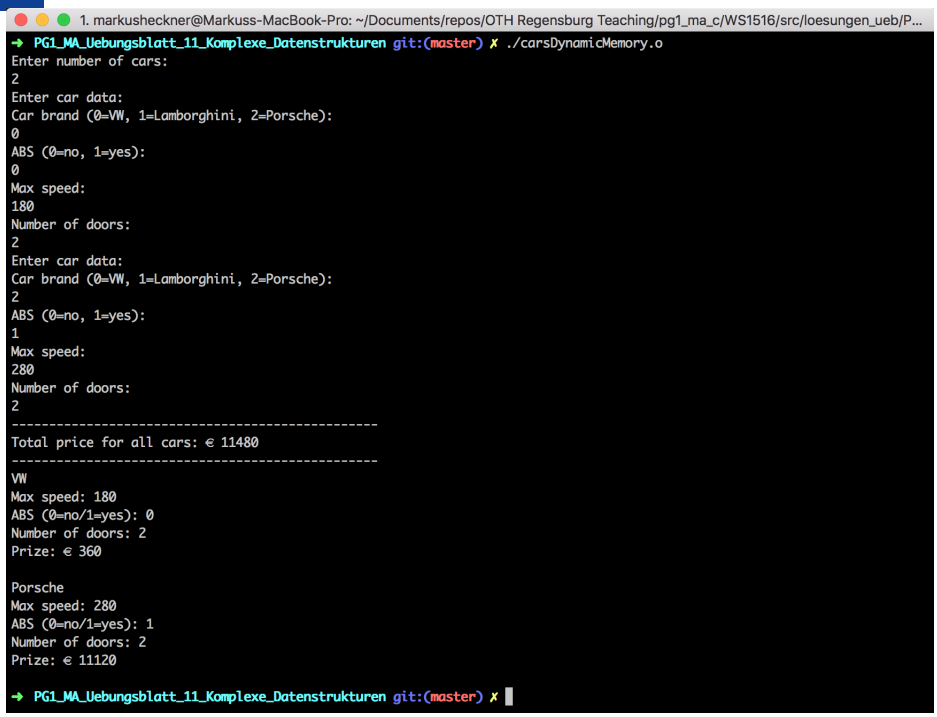
Aufgabe 4 : **Autos mit structs sortieren**

Ergänzen Sie das obige Beispiel um eine Sortierfunktion, die Ihnen das Array aus drei Autos aufsteigend nach dem Preis sortiert. Geben Sie anschließend die sortierte Liste wieder auf der Kommandozeile aus. Verwenden Sie zur Sortierung entweder das Verfahren Bubble Sort oder Selection Sort. Ihre Sortierung sollte für eine beliebige Anzahl an Autos in dem Array funktionieren.

Aufgabe 5 : **Erweiterung mit dynamischer Speicherverwaltung**

Passen Sie Ihr Beispiel so an, dass der Nutzer auswählen kann, wie viele Fahrzeuge er speichern will. Weiterhin soll der Nutzer die Eingaben für die Fahrzeuge auf der Kommandozeile geben können. Verwenden Sie dazu dynamische Speicherverwaltung.

Eine mögliche Ausgabe des Programms ist wie folgt:


1. markusheckner@Markuss-MacBook-Pro: ~/Documents/repos/OTH Regensburg Teaching/pg1_ma_c/WS1516/src/loesungen_ueb/P...
➜ PG1_MA_Uebungsblatt_11_Komplexe_Datenstrukturen git:(master) ✗ ./carsDynamicMemory.o
Enter number of cars:
2
Enter car data:
Car brand (0=VW, 1=Lamborghini, 2=Porsche):
0
ABS (0=no, 1=yes):
0
Max speed:
180
Number of doors:
2
Enter car data:
Car brand (0=VW, 1=Lamborghini, 2=Porsche):
2
ABS (0=no, 1=yes):
1
Max speed:
280
Number of doors:
2

Total price for all cars: € 11480

VW
Max speed: 180
ABS (0=no/1=yes): 0
Number of doors: 2
Prize: € 360
Porsche
Max speed: 280
ABS (0=no/1=yes): 1
Number of doors: 2
Prize: € 11120
➜ PG1_MA_Uebungsblatt_11_Komplexe_Datenstrukturen git:(master) ✗