Teil 3: Datentypen und Arrays

Nutzen Sie bitte ausschließlich die Programmierumgebung VSC

Datentypen

Aufgabe 3.1: Implizite und explizite Typumwandlung

Beschreiben Sie in Ihren eigenen Worten, was eine implizite Typumwandlung und was ein Typ-Cast ist. Zeigen Sie beide Umwandlungsarten anhand eines von Ihnen entwickelten C Programms und zeigen anhand des Programms, welche negativen Auswirkungen auf die Genauigkeit eine Typumwandlung haben kann.

Aufgabe 3.2: Wahr oder falsch?

Erklären Sie, welches Ergebnis der folgende Code liefert und was das mit Typumwandlung zu tun hat.

```c
#include <stdio.h>

int main(void) {
    int isTrue = 42;
    if (isTrue)
    {
        printf("Es ist wahr\n");
    } else
    {
        printf("Es ist nicht wahr\n");
    }

   return EXIT_SUCCESS;
}
```

Aufgabe 3.3: Array initialisieren und deklarieren

- Versucht man, ein Array wie gezeigt zu initialisieren und zu deklarieren, liefert der Compiler die Fehlermeldung für Zeile 4 "*Array Size must be defined*". Beheben Sie den Fehler und machen den Code lauffähig. **Achtung**: Deklaration und Initialisierung sollen getrennt geschehen.

1	`#include <stdio.h>`
2	`#include <stdlib.h>`
3	`int main(void) {`
4	`    int zahlen[];`
5	`    zahlen[] = {1,2,3,4};`
6	`    int len = sizeof(zahlen);`
7	`    printf("Anzahl der Elemente: %d\n", len);`
8	`    return EXIT_SUCCESS;`
9	`    }`

Wenn Ihr Code lauffähig ist, liefert er als Ergebnis höchstwahrscheinlich "*Anzahl der Elemente: 16*". Interpretieren Sie das Ergebnis und erweitern den Code so, dass tatsächlich die Anzahl der Elemente ausgegeben wird.

Arrays

Aufgabe 3.4: Medianfilter

Der Median ist ein statistisches Maß einer Messreihe und stellt eine robuste Alternative zum Mittelwert dar. "Robust" bedeutet in diesem Fall, dass er wenig von Ausreißern (sehr großen oder sehr kleinen Werten einer Messreihe) beeinflusst wird. Der Median einer Messreihe `measValues[ ]` wird dabei einfach wie folgt bestimmt:

- Sortieren Sie die Messreihe basierend auf den Messwerten und erhalten `measValuesSort[]`
- Ist die Anzahl *n* der Messwerte ungerade, so ist der Median in einem C Array definiert als `measValuesSort[(n-1)/2]`. Ist *n* jedoch gerade, so ist der Median definiert als `(measValuesSort[n/2] + measValuesSort[(n/2) -1])/2`.

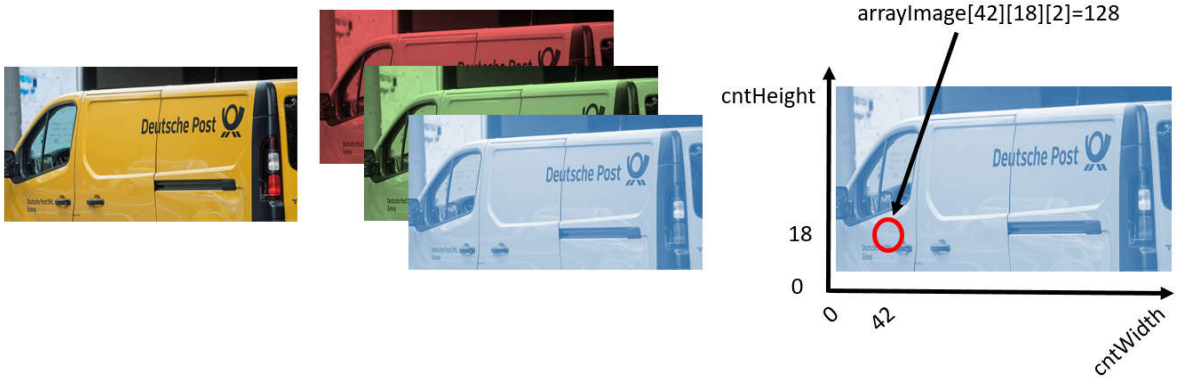
Sie finden im Ilias Ordner dieser Übung dazu ein C-Projekt als zip Archiv namens **medianFilter.zip**. Nutzen Sie die dort vorhandene Funktion `int sortArrayDescending(int sortArray[], int sizeArray)` und den vorhandenen Code, um den Median eines Messwerte-Arrays zu bestimmen.

Aufgabe 3.5: Post-Auto-Detektion

Bilder werden auf dem Rechner abgelegt in Form von Arrays. Handelt es sich dabei um klassische **R**ot**G**rün**B**lau Bilder, haben diese Arrays allgemein die folgende Form;

```
int arrayImage[cntWidth] [cntHeight] [nColors],
```

wobei der Farbwert eines jeden Bildpunkts (Pixels) als Wert zwischen 0 (schwarz) und 255 (voller Farbwert) in dem Array abgelegt wird und wobei `cntWidth` die Anzahl der Pixel auf der Breite des Bildes und `cntHeight` die Anzahl der Pixel auf der Höhe des Bildes sind. Das Bild besteht somit aus (`cntWidth` x `cntHeight`) Pixeln und jeder Pixel hat einen entsprechenden R-, G- und B-Wert. Im dem Beispiel der folgenden Abbildungen beträgt der blaue Farbwert an der Stelle 42(Breite),18(Höhe) also 128. Der entsprechende Grün Wert könnte abgefragt werden über `arrayImage[42][18][1]`.



Nutzen Sie diese Information, um einen Algorithmus zu entwickeln, der in einem 20x10 RGB-Bild, ein Postauto detektiert, sobald mehr als 40% der Pixel "gelb sind". Pixel gelten als "gelb", wenn der entsprechende Rot- **und** Blau-Wert des Pixels größer als 220 ist.

Aufgabe 3.7: Alphabetische Sortierung

Strings sind eine Array vom Datentyp `char`. Daher ist es auch möglich, ein Array von Strings zu erstellen und damit zu arbeiten.

Folgend ist ein Array von Nachnamen gegeben

```
char arr[X][Y] = {"Mustermann", "Bracht", "Zimmermann", "Schneider",
"Ackermann", };
```

Wählen Sie `X` und `Y` so, dass die Ausgabe aller Strings fehlerfrei funktioniert.

```
for (int i = 0; i < (X oder Y); i++)
{
    printf("%s\n", arr[i]);
}
```

Was passiert, wenn `X` und/oder `Y` zu gering gewählt werden und woran liegt das?

Nutzen Sie die Funktion `strcmp`(), um das Array anschließend alphabetisch zu sortieren und sortiert auszugeben.

Aufgabe 3.8: Lottozahlen

Das Ziehen der Lottozahlen soll realisiert werden, in folgenden Teilschritten:

a) Definieren Sie ein Feld für 6 Lottozahlen.
b) Initialisieren Sie den Zufallszahlengenerator, so dass er jedes Mal neue Zahlenserien erzeugt.
c) In einer Schleife sollen 6 zufällige Zahlen zwischen 1 und 49 erzeugt werden.
d) Geben Sie anschließend die Zahlen aus.

Es gibt noch ein kleines Problem: Es könnte sei, dass die gleichen Zahlen mehrmals generiert werden, aber die Lottozahlen können nur einmal vorkommen. Entwerfen Sie einen Algorithmus, um dieses Problem zu beheben.

Aufgabe 3.9: Palindrom prüfen

Ein Palindrom ist ein Wort, das von vorne und von hinten gelesen gleich ist, z.B. „Anna“, „Reittier“, „Otto“, „Rentner“. Erstellen Sie ein Programm, das eine Zeichenkette von der Standardeingabe einliest und prüft ob es sich um ein Palindrom handelt. Hinweis: wandeln Sie alle Buchstaben in Kleinbuchstaben um mit der Funktion `tolower()`.

`int tolower(int c)` - wenn c nicht schon ein Kleinbuchstabe ist, wird c umgewandelt in einen Kleinbuchstaben und als `return` Wert zurückgegeben.

Weitere Datentypen

Aufgabe 3.10: Mau Mau Spiel definieren

Es sollen die Datentypen (`*_t`) für ein Mau Mau Spiel implementiert werden. Entsprechende `typedef` Definitionen werden außerhalb des `main` Programms erstellt.

- **`bool_t`**: unsigned char
- **`farbe_t`**: Enumeration für die Spielfarben
- **`wert_t`**: Enumeration für die Spielwerte
- **`stapel_t`**: Enumeration für den Ort, an dem sich die Karte befindet (Nachziehstapel, Spielerhand, Ablagestapel)
- **`karte_t`**: Struct für eine einzelne Karte mit den Komponenten:
  - Spielfarbe
  - Spielwert
  - Ort, an dem sich die Karte gerade befindet
- **`kartenset_t`**: Struct für eine Menge von Karten mit den Komponenten:
  - Anzahl der Karten
  - Array vom Typ `karte_t`.
  - Bezeichner, um welchen Stapel es sich handelt

Deklarieren und initialisieren Sie im `main` Programm 3 Arrays

- **`farb_bezeichner`** soll die deutschen Bezeichnungen als Strings für die Spielfarben enthalten in der gleichen Reihenfolge wie in der Enumeration `farbe_t.`
- **`wert_bezeichner`** soll die deutschen Bezeichungen als Strings für die Spielwerte enthalten in der gleichen Reihenfolge wie in der Enumeration `wert_t`.
- **`kartendeck`** vom Typ `karte_t` soll mit allen 32 Spielkarten initialisiert werden. Als Ort soll für jede Spielkarte der Enumeration Wert Ablagestapel (vom Typ `stapel_t`) angegeben werden:

```
karte_t kartendeck [MAXCARDS] = {
    {caro, sieben, ablage},
     ...
    {kreuz, as, ablage}
};
```

Bitte definieren Sie für alle verwendeten Arraygrößen entsprechende Konstanten (mit `#define`).

Pointer

Aufgabe 3.11: Pointer verstehen

Erweitern und korrigieren Sie den Code so, dass:

- der Pointer `ptrMyVariable` auf die Variable `myVariable` zeigt und
- die Ausgabe die gewünschten Werte liefert.

```
int myVariable = 42;
int __ ptrMyVariable = __ myVariable;
printf("myVariable hat den Wert %d\n", __ myVariable);
printf("myVariable steht im Speicher bei %p\n", __ myVariable);
printf("Der Zeiger ptrMyVariable auf myVariable zeigt auf den Wert %d\n", __  ptrMyVariable);
printf("Der Zeiger ptrMyVariable auf myVariable hat den Wert %p\n", __ ptrMyVariable);
printf("Der Zeiger ptrMyVariable auf myVariable steht im Speicher bei %p\n", __ ptrMyVariable);
```

Aufgabe 3.12: Ausgabe der Array Elemente über Pointer Arithmetik

Ein Kommilitone kommt zu Ihnen und sagt wutentbrannt "Ich dachte gerade, ich hätte das schwierige Thema der Pointer endlich verstanden und dann kommt das…". Er schreibt Ihnen zwei Codezeilen auf

```
int myVariable = 42;
int *ptrMyVariable = &myVariable;
```

"Wenn ich jetzt aber auf ein Array zeigen möchte, muss ich programmieren…" und notiert folgende zwei Codezeilen

```
int myArray[] = {1,1,2,3,5,8,13};
int *ptrMyArray = myArray;
```

"Das ist doch zum Verrücktwerden!". Können Sie Ihrem Kommilitonen erklären, warum es "…=`&myVariable`" aber "…=`myArray`" heißen muss? Versuchen Sie Ihre Antwort durch ein kleines Programm zu belegen.

Schreiben Sie in einem zweiten Schritt ein Programm, dass alle Elemente von `myArray` ausgibt, indem Sie `ptrMyArray` inkrementieren.

Aufgabe 3.13: Pointer auf Strukturen

Erläutern Sie zunächst die Funktionsweise des folgenden Codes

```
#include <stdio.h>
#include <string.h>
enum sex
{
    MALE=0,
    FEMALE
};

typedef struct
{
    char name[30];
    int age;
    enum sex geschlecht;
```

```
} structPerson;

int main(void)
{
    structPerson myOnkelEnno;
    myOnkelEnno.geschlecht = 0;
    myOnkelEnno.age = 54;
    strcpy(myOnkelEnno.name, "Onkel Enno");
    return 0;
}
```

Ändern Sie den Code von einer direkten Zuweisung hin zu einer indirekten Zuweisung so, dass die Zuweisung über den Pointer `ptrMyOnkelEnno` auf die Struktur realisiert wird.

Welchen Vorteil hat ein solcher Pointer auf eine Struktur?