C-Kurs E2: Aufbau

Zielsetzung:
- Umgang mit Strings in C
- Auswertungsreihenfolge von Operatoren
- Die **`typedef`** Anweisung
- Vektoren einsetzen können
- Strukturen definieren und einsetzen
- Den Datentyp **`union`** kennenlernen

Hinweis:
Lösen Sie zuerst die mit (***) bezeichneten Aufgaben: Diese sind die Wichtigsten.

Aufgabenblock D: Zeichenketten (strings)
Der Umgang mit Zeichenketten (strings, char-Arrays) gestaltet sich in C einiges anders als bei anderen Hochsprachen. An dieser Stelle sind einige Aufgaben zum Verarbeiten von Strings aufgeführt.

A 16: (***) Ausgabe eines Strings, Funktion **`strlen()`**
Schreiben Sie ein Programm, wo Sie Ihren Namen einer Variablen zuweisen und diesen wieder auf der Konsole ausgeben. Zusätzlich soll die Anzahl Zeichen des Namens ausgegeben werden, verwenden Sie hierzu den **`strlen()`**-Befehl. Ist das Resultat von **`strlen()`** korrekt oder wird das Terminierungszeichen '**`\0`**' auch dazugezählt?

A 17: (***) **`sizeof()`** bei einem String
Warum wird im untenstehenden Programm **`4`** ausgegeben?

```
int main() {
    char * dummy= "zwei";
    printf("%d\n", sizeof(dummy));
    return 0;
}
```

A 18: Umwandeln einer Zahl in einen String: **`sprintf()`**
Die Jahreszahl sei in einer Datenbank als ganzzahliger Wert gespeichert. Der Darstellungsschicht soll das Jahr aber als String übergeben werden. Schreiben Sie eine Funktion, die die Jahreszahl entgegennimmt und ein String als Resultat zurückliefert. Ein Vorschlag für die Signatur der Funktion lautet wie folgt:

```
void convertYear(int year, char * buffer);
```

Beim zweiten Parameter wird eine Referenz auf einen Buffer übergeben, wo der erzeugte String abgelegt werden soll.

A 19: Umwandeln eines Strings in eine Zahl: **`sscanf()`**

Bei einem GUI (Graphical User Interface) erhält man die Benutzereingaben von den GUI-Komponenten als String. Bei einem Eingabefeld habe der Anwender die Möglichkeit, eine Jahreszahl einzugeben. Schreiben Sie eine Funktion, welche die Jahreszahl als String entgegennimmt und sie als **`int`** liefert. Falls die Jahrzahl ungültig ist (Jahr<1900 || Jahr>2100) soll dies durch den Rückgabewert **`1`** angezeigt werden, bei korrektem Datum soll **`0`** zurückgegeben werden.

Wie lautet die Signatur der Funktion?

Programmieren Sie die Funktion aus und schreiben Sie ein Testprogramm mit einem gültigen und einem ungültigen Datum.

A 20: Verketten von Strings: **`strcat()`**

Schreiben Sie ein Programm, welches Ihren Namen und Ihren Vornamen zu einem einzigen String zusammenfügt. Diese Aufgabe ist nicht so einfach zu lösen, da C den + Operator zur Verkettung von Strings nicht anbietet. Gehen Sie wie folgt vor, um die Aufgabe zu lösen:

- Erzeugen Sie einen Buffer, welcher genügend gross ist, um beide Strings aufnehmen zu können.
- Füllen Sie den Buffer zuerst mit dem Befehl **`memset()`** mit Nullen auf. Das "Ausnullen" des Buffers ist nötig, da sich bereits '**`\0`**' Zeichen in einem nicht initialisierten Array befinden können.
- Fügen Sie den ersten, und anschliessend den zweiten String mit **`strcat()`** dem Buffer hinzu.

Aufgabenblock E: Operatoren und Operatorenpräferenz, typedef

A 21: (***) Operatoren & Auswertungsreihenfolge

Analysieren Sie die untenstehenden Anweisungen. Die Variable **`a`** sei mit **`10`**, die Variable **`b`** sei mit **`7`** initialisiert. Welcher Wert wird jeweils der Variablen **`C`** zugewiesen? (jede Zeile soll einzeln interpretiert werden)

Ausdruck	Wert von c
c = ++a;	
c = b-- * a;	
c = --b * a;	
c = a % b * a;	
c = a % (b * a);	

A 22: (***) Eigene Datentypen mit **`typedef`** definieren
Schreiben Sie ein kleines Programm, welches den Typbezeichner **`real`** für einen **`double`** und **`pReal`** für einen Zeiger auf **`double`** definiert. Testen Sie die beiden neuen Datentypen, indem Sie ihnen einen Wert zuweisen und sie auf der Konsole ausgeben.

Aufgabenblock F: Abgeleitete Datentypen, Funktionen
In diesem Block werden die abgeleiteten Datentypen beübt.

A 23: (***) Vektoren initialisieren und ausgeben
Initialisieren Sie einen Vektor mit drei Elementen vom Typ **`int`**. Geben Sie den Inhalt des Vektors anschliessend in einer Schleife auf der Konsole aus (man nennt dies auch über den Vektor iterieren).

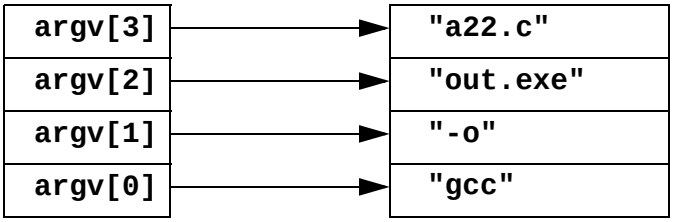
A 24: (***) Vektor einer Funktion übergeben
Schreiben Sie eine Funktion, welche den Mittelwert aller Elemente eines Vektors bestimmt. Die Elemente haben den Datentyp **`double`**. Da es bei der Programmiersprache C nicht möglich ist, aus einem Vektor seine Länge zu bestimmen, soll die Anzahl Elemente als zweites Argument der Funktion übergeben werden. Die zu implementierende Funktion hat damit die untenstehende Signatur:

```
double mean(double array[], int length);
```

A 25: (***) Übergabeparameter eines Programms
Die Argumente, die beim Starten eines Programmes angegeben werden (z.B. **`gcc -o out.exe a22.c`**), werden der **`main`**-Funktion in Form eines Vektors übergeben. Die vollständige Signatur der **`main`**-Funktion lautet nämlich:

```
int main(int argc, char * argv[])
```

Der erste Parameter gibt die Anzahl Argumente an, beim zweiten Parameter wird der Argumentvektor übergeben. Für das oben genannte Beispiel schaut der Argumentvektor wie folgt aus:



Das erste Element des Vektors zeigt immer auf den Programmnamen. Da die Elemente des Argumentvektors wiederum auf **`char *`** zeigen, ist auch die Notation **`char ** argv`** für den zweiten Parameter von **`main`** zulässig. Schreiben Sie ein Programm, das alle dem Programm übergebenen Argumente auf der Konsole ausgibt.

A 26: (***) Ein Struct für einen Artikel

Definieren Sie einen Struct, welcher in der Lage ist, die Daten eines Artikels eines Warenautomaten aufzunehmen. Der Struct soll wie folgt aufgebaut sein:

```
typedef struct {
    int artikelNummer;
    int anzahl;
    float preis;
    char bezeichnung[50];
} t_artikel;
```

Im Feld **anzahl** soll die Anzahl des Artikels gespeichert sein, die der Warenautomat am Lager hat. Schreiben Sie ausserdem eine Funktion **printArtikel**, welche in der Lage ist, die Felder des Artikels auf der Konsole auszugeben. Testen Sie Ihr Programm, indem Sie einen Artikel definieren und diesen der Funktion **printArtikel** übergeben.

A 27: (***) Inhalt von Structs bearbeiten

Wenn wir die Felder eines Structs innerhalb einer Funktion ändern wollen, müssen wir der Funktion einen Zeiger auf den Struct übergeben. Implementieren Sie im Programm von Aufgabe A26 folgende Funktion:

```
float sell(int anzahl, t_artikel *partikel);
```

Die Funktion soll Artikel des Warenautomates "verkaufen". Der Funktion wird der Artikel und die gewünschte Anzahl übergeben, als Resultat liefert die Funktin den Verkaufspreis. Die Anzahl Artikel im Struct (Feld **t_artikel.anzahl**) soll um die geforderte Anzahl reduziert werden. Geben Sie den Artikel vor und nach dem Aufruf von **sell** mit der Funktion **printArtikel** auf der Konsole aus.

A 28: Datentyp Union

Ein Programm soll die Verarbeitung von komplexen Zahlen ermöglichen. Eine komplexe Zahl setzt sich aus einem Real- und einem Imaginärteil zusammen, im Prinzip kann man sich die Zahl als eine Koordinate vorstellen.

Es soll ein Union-Typ **u_complex** verwendet werden können, dem man eine komplexe Zahl entweder als ein Array von zwei **double** Werten oder als Struct **t_complex** zuweisen kann. Der Struct **t_complex** soll die beiden Felder **re** (für den Realteil) und **im** (für den Imaginärteil) aufweisen, beide Felder haben den Datentyp **double**.

Schreiben Sie ein Programm, das die beiden Datentypen **u_complex** und **t_complex** definiert, zwei komplexe Zahlen auf die zwei verschiedenen Varianten initialisiert und deren Inhalt auf der Konsole ausgibt.