# Einführung in die Programmierung
# WS 2016/2017
# Blatt 10

Alex Hirsch, Simon Hangl, Sebastian Stabinger

2017–01–24

- Abgabe bis spätestens Montag 21:59 über OLAT (`https://lms.uibk.ac.at/olat/dmz/`).
- Bereiten Sie jede Aufgabe so vor, dass Sie Ihre Lösung im Proseminar präsentieren können!
- Benennen Sie Ihre Abgabe nach folgendem Schema: *Gruppennummer*-*Nachname*-`blatt`*Übungsblattnummer*`.tar.gz` Wenn Sie also Max Mustermann heißen und Gruppe 1 besuchen, heißt die Datei von Übung 10: `1-mustermann-blatt10.tar.gz`
- Compilieren Sie alle Programme mit den Optionen `-Wall -Werror -std=c99`

## Feedback

Nutzen Sie die angebotenen Möglichkeiten, uns Feedback zu geben (eMail, Tutorium, Proseminar). Hier können Sie uns auf Probleme, notwendige Stoffwiederholungen, Unklarheiten, aber auch positive Dinge, die beibehalten werden sollten, hinweisen.

## Testen und Dokumentation

Stellen Sie sicher, dass alle Lösungen fehlerfrei kompilieren. Testen Sie Ihre Lösungen ausführlich (z.B. auf falsche Eingaben, falsche Berechnungen, Sonderfälle) und dokumentieren Sie sie. Dies hilft Ihnen bei der Präsentation und uns beim Nachvollziehen Ihrer Entscheidungen. Lesen Sie die Aufgaben *vollständig* durch.

## Aufgabe 1 (4 Punkte)

- Fragen Sie die Größe eines Arrays vom Benutzer ab und erzeugen Sie das Array mittels dynamischer Speicherverwaltung
- Füllen Sie das Array mit Zahlen. (z.B. der Benutzer gibt `100` ein. Es wird ein Array der Größe 100 erzeugt und mit Zahlen von 1 bis 100 gefüllt)
- Anschließend bilden Sie die Summe aller Elemente des Arrays und geben das Ergebnis aus. Achtung: Lesen Sie die Werte tatsächlich aus dem Array!
- Geben Sie den Speicher des Arrays wieder frei und berechnen Sie die Summe erneut.
- Funktioniert das? Bekommen Sie das richtige Ergebnis? Bekommen Sie Laufzeitfehler?

## Aufgabe 2 (6 Punkte)

Gegeben ist folgendes `struct`:

```
struct Person {
    char firstname[64];
    char lastname[64];
    int age;
};
```

Implementieren Sie *Vergleichs-Operatoren* welche 2 Personen miteinander vergleichen. Jede Funktion soll eine Eigenschaft der 2 Personen (`firstname`, `lastname` oder `age`) vergleichen.

```
int Person_cmp_firstname(const void* x, const void* y) { /* TODO */ }

int Person_cmp_lastname(const void* x, const void* y ) { /* TODO */ }

int Person_cmp_age(const void* x, const void* y) { /* TODO */ }
```

Lesen Sie sich die man-page der Funktion `qsort` durch. Die Signaturen der Vergleichs-Operatoren wurde so gewählt, dass sie mit dieser Funktion kompatibel sind.

Definieren Sie ein **Array von Zeigern** auf, **mittels dynamischer Speicherverwaltung** erzeugten `Person` Strukturen. Das Array soll Platz für drei Zeiger haben.

Lesen Sie 3 Personen vom User ein (jeweils `firstname`, `lastname` und `age`) und speichern sie diese in den im Array verlinkten Strukturen.

Danach soll der User auswählen können, nach welcher Eigenschaft er die 3 Personen sortiert haben will. Das Program sortiert dann entsprechend (mittels `qsort`) und gibt die 3 Personen (sortiert) aus.

Geben Sie am Ende des Programms den reservierten Speicher wieder frei.

***Hinweis:*** *Abgabe: 1-mustermann-a2.c*