**Programmierkurs (IPK) WS20/21**
Dr. Ole Klein, René Heß
IWR, Universität Heidelberg

**Aufgabenblatt 12**
Abgabedatum 19. Februar 2021
ipk-exercises:2020-ws-30-g7c572da

# Aufgabenblatt 12

**Hinweise zur Klausur:**

- Der Termin der Klausur ist Freitag der 12. März zwischen 14 und 16 Uhr.
- Sie müssen sich bis zum Mittwoch den 10. März, 12 Uhr für die Klausur anmelden: https://muesli.mathi.uni-heidelberg.de/lecture/view/1248
- Überprüfen Sie ob Ihre Zulassung korrekt angezeigt wird. Ab dem 20. Februar sollten die Daten korrekt in Müsli eingetragen sein. Sollten Sie eine Zulassung aus dem letzten Semester haben ist dieser Schritt besonders wichtig. Am 19. Februar werden Sie per Email daran erinnert dies zu überprüfen.
- Ablauf:
  - Die Klausur wird online stattfinden. Dafür wird am 12. März um 14 Uhr ein Zettel namens Klausur in Mampf https://mampf.mathi.uni-heidelberg.de/lectures/64 veröffentlicht.
  - Sie haben zwei Stunden Zeit um diesen Zettel selbstständig zu bearbeiten und in Mampf hochzuladen. Dabei sind 90 Minuten Bearbeitungszeit und 30 Minuten Zeit für das Herunterladen der Aufgaben und das Hochladen Ihrer Lösung eingeplant.
  - Stellen Sie sicher, dass Sie vor Ablauf der Frist eine Lösung hochladen. Mampf erlaubt es Ihnen bis zum Ende der Abgabefrist die hochgeladene Lösung auszutauschen.
  - Sie müssen die Klausur handschriftlich bearbeiten. Dabei sind sowohl eingescannte/fotografierte Zettel als auch handschriftliche Tablet Notizen zulässig.
  - Sie werden in der Klausur C++ Code schreiben. Auch diese Aufgabenteile müssen handschriftlich (digitalisiert auf Papier/Tablet) eingereicht werden. Einfache Syntaxfehler sind dabei nebensächlich.
  - Das Abgabeformat wird pdf sein. Sie müssen sicherstellen, dass Sie Ihre digitalisierten handschriftlichen Notizen in dieses Format transformieren können.
  - Kommunikation bei Fragen: Wir werden während der Klausur den IPK HeiConf Raum https://audimax.heiconf.uni-heidelberg.de/ht6r-kjtw-xguu-dxkr öffnen um Unklarheiten in der Aufgabenstellung zu klären. Dort können Sie Fragen über den **privaten Chat** stellen (schreiben in den öffentliche Chat wird für Sie deaktiviert sein). Ihre Frage wird entweder im privaten Chat oder in einer separaten HeiConf Sitzung geklärt. Sollte die Antwort für andere relevant sein, wird die Antwort über den öffentlichen Chat und über Audio kommuniziert.
- Die Klausur wird ein spezielles Deckblatt haben. Auf diesem Deckblatt müssen Sie Ihre persönlichen Daten eintragen und eine Selbstständigkeitserklärung unterzeichnen. Dieses Deckblatt wird vorab veröffentlicht um Ihnen Zeit zu geben dieses Blatt auszudrucken oder handschriftlich zu reproduzieren. Fügen Sie dieses Deckblatt Ihrer Abgabe hinzu.
- Sie können die Abgabe in Mampf mit diesem Zettel üben. Sie können bis zum 19. Februar 12 Uhr eine Lösung im pdf Format in Mampf hochladen. Dafür wurde ein Tutorium namens Probeklausur eingerichtet. Ihre Lösung wird nicht korrigiert, es geht nur darum den Workflow zu testen.
- **Wichtig:** Sollten Sie nicht die nötigen technischen Mittel haben um in diesem Format an der Klausur teilzunehmen (z.B. keine Möglichkeit der Digitalisierung der handschriftlichen Notizen)

sollten Sie sich umgehend unter rene.hess@iwr.uni-heidelberg.de melden.

- Das Ergebnis Ihrer Klausur wird zeitnah für Sie in Müsli einsehbar sein. Zur Zeit können wir noch keine genaueren Angaben machen, da der Korrekturzeitpunkt vom Pandemieverlauf abhängt.
- Sollten Sie die Klausur nicht bestehen wird es die Möglichkeit einer Nachklausur geben. Es kann unter Umständen möglich sein, dass für die Nachklausur eine alternative Prüfungsform gewählt wird (z.B. Präsenz- oder mündliche Prüfung).
- **Hilfsmittel:** Die Klausur wird im open-book Format durchgeführt. Sie dürfen sämtliche Vorlesungsmaterialien, Bücher, Informationen aus dem Internet, Texteditoren und Compiler verwenden. Sie sind jedoch dazu verpflichtet die Klausur selbstständig zu bearbeiten, d.h. Sie dürfen nicht die Hilfe anderer Leute einholen, die Aufgaben diskutieren oder im Internet nach Lösungen fragen.
- Sollten Sie am Tag der Klausur krank werden müssen Sie sich bis 13:30 Uhr per Email an rene.hess@iwr.uni-heidelberg.de abmelden. In dringenden Fällen können Sie auch die Telefonnummer +4962215414511 verwenden. Senden Sie in diesem Falle bis zum 13. März 12 Uhr ein (digitalisiertes) ärztliches Attest an rene.hess@iwr.uni-heidelberg.de.
- Die Aufgaben auf diesem Zettel dienen der Klausurvorbereitung, werden nicht korrigiert und sind irrelevant für die Zulassung zur Klausur. Der Umfang dieses Zettels entspricht nicht dem der Klausur. Die Aufgaben sollen stattdessen einen Einblick geben, welche Aufgabentypen Sie in der Klausur erwarten können.
- Der Klausurumfang wird deutlich höher sein. Stellen Sie sich mental darauf ein und laden Sie rechtzeitig eine erste Lösung in Mampf hoch.
- Hinweis: Der Inhalt dieses Zettels ist Klausurrelevant. Insbesondere werden in Aufgabe 5 Lambda Funktionen behandelt. Diese könnten also in der tatsächlichen Klausur abgefragt werden.

---

## Aufgabe 1: Verständisfragen (0 Punkte)

Geben Sie kurze Antworten auf die folgenden Fragen:

(a) Was ist der Unterschied zwischen call by value und call by reference?

(b) Was macht der Linux Befehl `grep`?

(c) Nennen Sie drei Konzepte, welche dabei helfen können C++ Klassen wiederverwendbar zu machen.

(d) Was ist eine dangling reference?

## Aufgabe 2: Code Verstehen (0 Punkte)

(a) Gegeben sei der folgende Codeausschnitt[1].

```
1 class Krokodil : public Tier
2 {
3   // Number of teeth
4   int _teeth;
5 public:
6   Krokodil() : Tier(), _teeth(66) {}
7 };
```

Was bedeutet die erste Zeile? Wie nennt man die Methode, die in Zeile 6 implementiert ist? Was passiert in Zeile 6?

(b) Gegeben sei der folgende Codeausschnitt

[1] Die genaue Anzahl der Zähne hängt wohl von der Art ab. Salzwasserkrokodile haben anscheinend 66 Zähne.

```cpp
void warn_visitor()
{
  std::cout << "Warnung! Freilaufende Krokodile!" << std::endl;
}
```

Was bedeuten `void`, `std::cout` und `std::endl`?

## Aufgabe 3: Fehler Finden (0 Punkte)

(a) Finden Sie den/die Fehler in dem folgenden C++ Programmausschnitt. Beschreiben Sie den Fehler und geben Sie dabei an, ob es sich um einen Kompilierfehler oder Laufzeitfehler handelt.

```cpp
int main(int argc, char** argv)
{
  int x = 4;
  int& ref_1 = x;
  int& ref_2 = ref_1;
  double ref_3 = x;
  int& ref_4;
}
```

(b) Finden Sie den/die Fehler in dem folgenden C++ Programmausschnitt. Beschreiben Sie den Fehler und geben Sie dabei an, ob es sich um einen Kompilierfehler oder Laufzeitfehler handelt.

```cpp
#include <iostream>

class Image {
  void resize(double factor);

  // Rest of class and implementation omitted
  ...
};

void resize_image(const Image& image, const int factor){
  image.resize(factor);
}
```

## Aufgabe 4: Code Ergänzen (0 Punkte)

Gegeben sei der unten stehende Codeausschnitt. Ihre Aufgabe ist es eine Klasse namens `Loesung` zu schreiben, sodass ein Objekt dieser Klasse von der Funktion `func` aufgerufen werden kann. Schreiben Sie die Klasse `Loesung` so, dass der Code kompilieren würde und fehlerfrei ausgeführt werden kann.

```cpp
template <typename T>
typename T::value_type func(const T& t)
{
    auto result = t.doSomething(3.8);
    return result;
}
```

## Aufgabe 5: Komplexität (0 Punkte)

Lesen Sie sich folgenden Code durch:

```cpp
template<typename Calculate>
int f(const std::vector<int>& vec, Calculate calculate)
{
  int x = 0;

```

```
6     for (std::size_t i = 0 ; i < vec.size() ; ++i)
7       for (std::size_t j = 0 ; j < vec.size() ; ++j)
8         x = std::max(x,calculate(vec,i,j));
9
10    return x;
11  }
12
13  int g(const std::vector<int>& vec)
14  {
15    return f(vec,[](std::vector<int> v, std::size_t i, std::size_t j)
16      {
17        return v[i] - v[j];
18      });
19  }
```

(a) Was für einen Wert `x` berechnet die Funktion `g()` bezüglich der Werte in `vec`?

(b) Welche Laufzeit-Komplexität besitzt `g()` bezüglich der Größe `n = vec.size()` von `vec`, d.h. wie viele Operationen (Addition, Subtraktion, Multiplikation, Division, Kopieren) führt die Funktion in Abhängigkeit von $n$ aus (1 (unabhängig von $n$), $\log n$, $n$, $n^2$, $n^3$, $2^n$, $\dots$)? (Hinweis: Das Inkrementieren von Schleifenvariablen wird dabei nicht mitgezählt werden.)

(c) Mit welcher minimalen Änderung an `g()` können Sie die Laufzeit-Komplexität von `g()` deutlich verbessern, und welche Laufzeit-Komplexität erhalten Sie dann?

(d) Können Sie einen effizienteren Algorithmus angeben (in Codeform oder in Worten), der das gleiche berechnet? Dieser muss nicht weiter `f()` verwenden. Welche Laufzeit-Komplexität hat Ihre verbesserte Variante?