

Übungsblatt 1

C++ Grundlagen

Versuchen Sie die folgenden Fragen auf (digitalem) Papier zu lösen. Bei manchen Aufgaben können Sie zur (Selbst-) Kontrolle einen C++-Compiler zu Rate ziehen. Die Fragen orientieren sich an möglichen Klausuraufgaben. Denken Sie bei der Lösung der Aufgaben also daran, dass Sie in einer Klausur keinen Compiler benutzen dürfen!

Aufgabe 1 - Erzeugen eines C++ Programms

1. Beschreiben Sie die Schritte zur Erzeugung eines C++ Programms (Skizze).
2. Was ist eine Objektdaten und wann wird diese erzeugt?
3. Wozu dient die Präprozessordirektive `#include`?

Aufgabe 2 - Namensgebung

Welche/r der folgenden Bezeichner in C++ sind/ist korrekt?

- ~~new~~
- `_hallo` ✓
- ~~843~~
- ~~5h~~
- ~~c++~~
- ~~_____~~

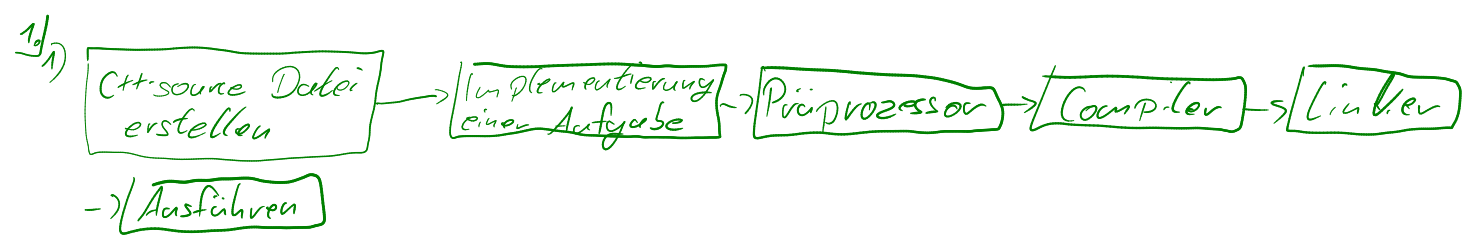
Aufgabe 3 - Typumwandlung

1. Was ist der Unterschied zwischen impliziter und expliziter Typumwandlung?
2. Geben Sie ein Beispiel für eine implizite Typumwandlung in C/C++.
3. Geben Sie ein Beispiel für eine explizite Typumwandlung im C-Stil.
4. Geben Sie ein Beispiel für eine explizite Typumwandlung im C++-Stil.

Aufgabe 4 - Referenzen

In C++ kann sowohl mit Werten (Values) als auch mit Referenzen (References) gearbeitet werden.

1. Geben Sie jeweils ein Beispiel für Call-by-Value und Call-by-Reference.
2. Wo liegt der Unterschied?
3. Wann sollte man Call-by-Value Aufrufe nutzen und wann Call-by-Reference?



2) Es ist eine Zwischen-Datei, die beim Kompilieren als eine kombinierte Datei aus header- und source-Datei entsteht

3) Sie bindet eine header-Datei in die source-Datei ein

3.1) 1) bei impliziter Typenumwandlung wird ohne es direkt anzugeben der Datentyp einer Variable verändert, bei expliziter wird direkt im Code geschrieben wie die Variable verändert werden soll

2) `int i = 4.3;`

3) `int i = (int) 4.3;`

4) `int i = static_cast<int>(4.3);`

4) 1) `foo(int i) { };`
`foo(&int, i) { };`

2) Call-By-reference übergibt die Speicherstelle einer Variable, sie kann nicht verändert werden, Call-By-value übergibt direkt den Wert.

3) Value nutzt man wenn man den Wert verändern möchte, reference, wenn man nur mit dem Wert rechnen möchte

5) 1) 2
 4
 3
 0
 4
 4

2) Nur a ist global

3) Weil globale Variablen von jeder Methode beschrieben werden können und dadurch schnell Fehler passieren.

Aufgabe 5 - Gültigkeitsbereich

Der sogenannte Gültigkeitsbereich (Scope) ist insbesondere wichtig für den Zugriff auf Variablen. ¹ Gegeben ist folgendes Beispiel Listing 1.

1. Welche Ausgabe hat der Quelltext?
2. Markieren Sie die Gültigkeitsbereiche der Variablen a, b, c im Quelltext.
3. Bei welcher der Variablen handelt es sich um eine globale bzw. lokale Variable.
4. Warum sollten globale Variablen in gutem Quelltext nicht eingesetzt werden?

```
#include <print>
using namespace std;
```

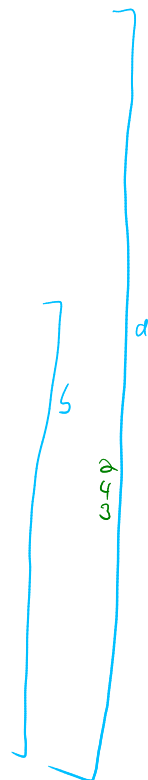
```
int a = 3;
```

```
void foo()
```

```
{
    a -= 3;
    int b = a - 1;
}
```

```
int main()
```

```
{
    int b = a;
    a--;
    {
        int c = b;
        b++;
        println("{} ", a);
        println("{} ", b);
        println("{} ", c);
    }
    foo();
    int c = b;
    println("{} ", a);
    println("{} ", b);
    println("{} ", c);
}
```



Listing 1: Gültigkeitsbereiche

¹Wir werden später sehen, dass im Konzept der Objektorientierung auch Funktionen in C++ einen Gültigkeitsbereich erhalten.