# Aufgabenblatt 7: Zeiger und Funktionen

Florian Ludewig (Übungsgruppe 2)

December 6, 2019

## Aufgabe 1 – Zeiger in C

```
1  #include <stdio.h>
2
3  int main()
4  {
5    int var = 10;
6    int *p;
7    p = &var;
8
9    printf("Ausgabe 1: %p \n", &var);
10   printf("Ausgabe 2: %d \n", *p);
11   printf("Ausgabe 3: %d \n", *(&var));
12   printf("Ausgabe 4: %p \n", &p);
13
14   return 0;
15 }
```

### Ausgabe 1

`printf("Ausgabe 1: %p \n", &var);` liefert foldende Ausgabe: `Ausgabe 1: 0x7fff086e611c`. Durch das `&` Symbol wird die Speicheradresse der nachstehenden Variable, in diesem Fall von `var`, ausgegeben.

### Ausgabe 2

`printf("Ausgabe 2: %d \n", *p);` gibt `Ausgabe 2: 10` aus. Durch `p = &var;` wurde in Zeile 7 dem Zeiger `p` die Speicheradresse von dem Integer `var` zugewiesen. Nun wird mithilfe des `*` Symbols der Wert ausgelesen der in der Speicheradresse von dem Zeiger `p` gepspeichert wird (in dem Fall der Wert der Variable `var`, also `10`).

### Ausgabe 3

`printf("Ausgabe 3: %d \n", *(&var));` liefert foldende Ausgabe: `Ausgabe 3: 10`. Hier gibt `&var` die Speicheradresse von der Variable `var` zurück. Durch das `*` Symbol wird dann der Wert an dieser Speicheradresse ausgelesen. Dieser Wert ist `10`, weil in Zeile `5` dem Integer `var` der Wert `10` zugewiesen wurde.

### Ausgabe 4

`printf("Ausgabe 4: %p \n", &p);` gibt `Ausgabe 4: 0x7fff086e6120` aus. Durch das `&` Symbol wird die Speicheradresse von dem Pointer `p` ausgegeben. Es handelt sich hierbei also nicht um die Adresse, die in dem Pointer `p` gepspeichert ist, sondern um die Adresse an welcher der Pointer `p` selbst gepspeichert ist. Aus diesem Grund ist Ausgabe 4 auch verschieden von Ausgabe 1.

## Aufgabe 2 – Wertetabellen

### a)

```c
#include<stdio.h>

double quadrat(double x) { return x*x; }

void wertetabelle(double (*f)(double), double min, double max, double step)
    {
  int steps = (max - min) / step;
  for (int i = 0; i <= steps; i++) {
    double x = min + (i * step);
    printf("f(%lf) = %lf\n", x, f(x));
  }
}

int main() {
  wertetabelle(quadrat, 0.0, 10.0, 1.0);
  return 0;
}
```

### b)

```c
#include<stdio.h>
#include <stdarg.h>

double cubic(double x) { return x*x*x+2.0; }

void wertetabelle(double (*f)(double), int count, ...) {
  va_list args;
  va_start(args, count);
  for (int i = 0; i < count; i++) {
    double x = va_arg(args, double);
    printf("f(%lf) = %lf\n", x, f(x));
  }
  va_end(args);
}

int main() {
  wertetabelle(cubic, 4, 3.0, 7.0, 11.0, 41.0);
  return 0;
}
```