



Programmieren in C++

Vorlesung im Sommersemester 2018
Prof. Dr. habil. Christian Heinlein

3. Übungsblatt (5. April 2018)

Aufgabe 5: Deklarationen

- a) Was bedeuten die folgenden Deklarationen?
Geben Sie jeweils eine mögliche Initialisierung an!

```
int*& a;  
const int*& b;  
int const*& c;  
int* const& d;
```



Hilfsdeklarationen für die Initialisierungen:

```
// Zeiger auf int.  
int* x;  
  
// Zeiger auf ein konstantes int-Objekt.  
const int* y;
```

Eigentliche Deklarationen:

```
// Referenz auf einen Zeiger auf int.  
// Initialisierung mit einem Zeiger auf int (L-Wert).  
int*& a = x;  
  
// Referenz auf einen Zeiger auf ein konstantes int-Objekt.  
// Initialisierung mit einem Zeiger auf ein konstantes int-Objekt (L-Wert).  
const int*& b = y;  
int const*& c = y;  
  
// Referenz auf einen konstanten Zeiger auf int.  
// Initialisierung mit einem Zeiger auf int (L- oder R-Wert).  
int* const& d = x;
```



- b) Deklarieren Sie folgende Objekte:
- ein Feld mit 10 Zeigern auf int;
 - eine Referenz auf ein solches Feld;
 - einen Zeiger auf ein Feld mit 10 ints;

- einen Zeiger auf eine Funktion mit Parameter- und Resultattyp `int`;
- ein Feld mit 10 solchen Zeigern.
- eine Referenz auf ein solches Feld.



```
// Feld mit 10 Zeigern auf int.
int* a [10];

// Referenz auf ein solches Feld (mit Initialisierung).
int* (&b) [10] = a;

// Zeiger auf ein Feld mit 10 int-Objekten.
int (*c) [10];

// Zeiger auf eine Funktion mit Parameter- und Resultattyp int.
int (*d) (int);

// Feld mit 10 solchen Zeigern.
int (*e[10]) (int);

// Referenz auf ein solches Feld (mit Initialisierung).
int (*(&f)[10]) (int) = e;
```

Hinweis: Die postfix angewandten Typkonstruktoren `[]` und `()` zur Deklaration von Feldern bzw. Funktionen haben Vorrang vor den präfix angewandten Typkonstruktoren `*` und `&` zur Deklaration von Zeigern bzw. Referenzen. Durch explizite Klammerung kann man den Vorrang jedoch explizit verändern.

Um die Lesbarkeit komplexer Deklarationen zu verbessern, kann man sie schrittweise mittels `typedef` aufbauen, z. B.:

```
// Zeiger auf eine Funktion mit Parameter- und Resultattyp int.
typedef int (*D) (int);
D d;

// Feld mit 10 solchen Zeigern.
typedef D E [10];
E e;

// Referenz auf ein solches Feld (mit Initialisierung).
typedef E& F;
F f = e;
```



Aufgabe 6: Referenzen

Lösen Sie die folgenden „Referenz-Puzzles“, d. h. sagen Sie die Ausgabe der Programme voraus, bevor Sie sie ausprobieren!

Teilaufgabe 6.a)

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

int& f (int& x, int& y) {
    return x < y ? x : y;
}

int main () {
    int a = 1, b = 2;
    f(a, b) = 3;
    f(a, b) = 4;
    cout << a << " " << b << endl;
}
```



Die Ausgabe lautet: 3 4



Teilaufgabe 6.b)

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

int x = 1;

void f (int& y) {
    x += y;
    y += x;
}

int main () {
    f(x);
    cout << x << endl;
}
```



Die Ausgabe lautet: 4



Teilaufgabe 6.c)

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

int a [] = { 1, 2, 3 };
int* p = a + 3;

void f (int& x, int& y) {
    x = x + y;
    y = x - y;
    x = x - y;
}

int main () {
    f(*a, a[1]);
    f(a[2], p[-2]);
    cout << a[0] << " " << a[1] << " " << a[2] << endl;
}
```



Die Ausgabe lautet: 2 3 1



Teilaufgabe 6.d)

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

int& swap (int& x, int& y) {
    int z = x; x = y; y = z;
    return x;
}

int main () {
    int a = 1, b = 2, c = 3, d = 4;
    swap(a, swap(b, swap(c, d)));
    cout << a << " " << b << " " << c << " " << d << endl;
}
```



Die Ausgabe lautet: 4 1 2 3

