

3. November

Referenzen: Wir können mithilfe von Referenzen ein Alias einer Variable erzeugen. Sie können nur Variablen ihres Typs referenzieren und auch nur mit L-Werten initialisiert werden (da nur L-Werte Speicheradressen haben und eine Referenz auf die Adresse zeigt).

Beispiel:

```
int a = 1;  
int& b = a;  
b++;  
std::cout << a;
```

in diesem Programm wird **2** ausgegeben, da wir den Wert von `b` und somit (da `b` Alias von `a` ist) auch den Wert von `a` verändert haben.

Vor allem sind Referenzen nützlich in Funktionen!

Beispiel:

```
void incre (int& m) {  
    m++;  
}
```

```
int main() {  
    int a = 1;  
    incre(a);  
    std::cout << a << std::endl;  
    return 0;  
}
```

Hier wird **2** ausgegeben, da m ein Alias von a ist und somit die Änderung von m in "void incre" auf a übertragen wird. Dies nennt man "call by reference". Wir haben bis jetzt nur mit "call by value" gearbeitet. Dann wird aber a als Kopie übergeben und somit ist der Output des folgenden Programms **1**.

```
void incre (int call by value m) {  
    m++;  
}  
  
int main() {  
    int a = 1;  
    incre(a);  
    std::cout << a << std::endl;  
    return 0;  
}
```

Vorteile:

1. Es können mehrere Variablen returned, da wir diese einfach als Referenzen übergeben können.
2. Es spart Speicher. Bspw. muss kein Vektor kopiert werden, sondern nur die Adresse angegeben werden.

Vektoren : (benötigt `#include <vector>`) vom selben Typ
Wir können mehrere Werte in einem Vektor abspeichern.

`std::vector<Typ> name (= std::vector<Typ> (länge, Wert);`

oder

`std::vector<Typ> name (= std::vector<Typ> {Wert1, Wert2...});`

das in blauen Klammern kann auch weggelassen werden.

Folgende Funktionen sind nützlich :

- `push-back(Wert)` : hängt Wert hinten an Vektor an.
- `at(n)` : gibt (n+1)-ten Wert von Vektor aus.
- `size()` : gibt Länge des Vektors aus.

char : Ein weiterer Typ, den wir einführen ist `char`.

`char` ist ein Charakter der ASCII-Tabelle. Man kann `chars` einfach in `ints` umwandeln (siehe ASCII) und zurück.

Beispiel:

```
char a = 'd';  
a = a + ('e' - 'd');  
std::cout << a;
```

Die Ausgabe ist hier `'b'`, da `'e' - 'd' = 1` und `'d' + 1 = 'b'`. Die ASCII Tabelle findet ihr auf der nächsten Seite.