

# Funktionen

---

## Gültigkeitsbereiche von Variablen

---

```
int incr(int i) {
    ++i;
    return i;
}

int main() {
    int i = 5;
    int j = incr(i);
    printf("%d %d\n", i, j);
    return 0;
}
```

*Analogie zu Speicheradressen:*

- Die Adresse ist vergleichbar mit einer Wohnadresse
  - Die Variable wohnt in einer Straße, der Funktion, der sie zugeordnet ist
  - Der Compiler vergibt zusätzlich eine Hausnummer innerhalb der Straße zur Identifikation der verschiedenen Variablen einer Funktion
- 
- Variablen sind Speicherbereiche mit Adressen
    - Durch die Adresse kann der Speicher der Variable identifiziert werden (wo liegen die Daten im Rechner)
  - Variablen gibt es nur dort, wo sie deklariert wurden (= Scope einer Variable)
  - Auf Variablen in anderen Scopes kann nicht zugegriffen werden
  - Variablen mit demselben Namen in verschiedenen Scopes sind verschiedene Variablen
  - Werte in Funktionsparametern werden in neue Variablen der Funktion kopiert
    - Der Parameter innerhalb der Funktion ist nicht derselbe Wert wie der übergebene
    - Wird call-by-value oder pass-by-value genannt

## Call-by-value

```
void foo(int a) {
    ++a;
    printf("%d\n", a);
}
```

```
int main() {
    int i = 2;
    foo(i);
    printf("%d\n", i);
}
```

- i in `main()` ist nicht die gleiche Variable wie a in foo
- Der Wert aus i wird mit dem Aufruf von `foo(i)` in die Variable a kopiert

## Globale Variablen

- Der globale Scope gilt für das gesamte Programm
- Er liegt außerhalb aller Funktionen
- Variablen, die dort deklariert wurden sind globale Variablen
- Auf sie kann überall zugegriffen werden

```
int x; // globale Variable

void foo() {}

int main() {}
```

## Funktionen deklarieren

Funktionen müssen vor der ersten Verwendung deklariert werden (genauso wie Variablen)

```
void foo(); // es wird eine Funktion foo geben

int main() {
    foo();
}

void foo() {}
```

- Mit einem Funktionsprototypen (forward declaration) lässt sich eine Funktion ankündigen, ohne sie bereits zu implementieren
- Damit können Zirkelverweise (Funktion A braucht Funktion B und umgekehrt) aufgelöst werden
-