WP-Kalkül:

- ermöglicht es die Korrektheit von Programmen mathematisch zu beweisen
- sinnvoll für sicherheitskritische Systeme (AKWs, etc.)
- Vorbedingung = P , Nachbedingung = Q
- wp = weakest precondition (schwächste Vorbedingung) für die Q (gerade) noch zutrifft
- Programm wird vom Ende her geprüft, Q bekannt
- prinzipiell einfache Ersetzungsregeln, bei Schleifen zusätzliche Bedingungen

Man schreibt den code ins wp :

```
a = b + 3;
a += 3 * b;
// Q : a = 5
```

$$wp(\text{"a = b + 3; a += 3 * b"}, a = 5) \equiv$$
$$wp(\text{"a = b + 3; a = a + 3 * b"}, a = 5) \equiv$$
$$wp(\text{"a = b + 3;"}, a + 3 \cdot b = 5) \equiv$$
$$(b + 3 + 3 \cdot b = 5) \equiv$$
$$(4b = 2) \equiv$$
$$(b = 0.5)$$

Dann fängt man von rechts an die befehle in die Bedingung(Q) einzuarbeiten.
Usw. hier ist die precondition b = 0.5.

If:

If-Abfragen

```
if (a > 0) {
    a = a + 3;
} else {
    a = ++a - a++ - 5;
    a *= -1;
}
// Q : a = 5
```

$$[(a > 0) \wedge wp(\text{"a = a + 3"}, a = 5)] \vee$$
$$[(a \leq 0) \wedge wp(\text{"a = ++a - a++ - 5; a *= 1"}, a = 5)] \equiv$$
$$[(a > 0) \wedge (a = 2)] \vee [(a \leq 0) \wedge (-5 = -5))] \equiv$$
$$[(a = 2) \vee (a \leq 0)]$$

Es muss für beide Fälle ein WP erstellt werden.

Switch:

Switch-Case

```
switch (a) {
    case 5:
        a = 3;
    case 2:
        a = 50;
    default:
        a = 5;
}
// Q: a=5
```

$\equiv$

if

```
if (a == 5) {
    a = 3;
    a = 50;
    a = 5;
} else if (a == 2) {
    a = 50;
    a = 5;
} else {
    a = 5;
}
```

Switch-Case kann immer in if umgewandelt werden.
(Vorsicht hier wurde das 'break;' weggelassen, deswegen werden bei a = 5 alle anderen auch ausgeführt)

Hier werden dann 3 Fälle behandelt.
[(a=5)^wp("a = 3; a= 50; a = 5";a=5)] v [(a=2)^wp("a= 50; a = 5";a=5)] v [(else)^wp("a = 5";a=5)]
…

Jetzt der spaßige Teil: Schleifen:

Schleifen

- Schleifeninvariante I, Schleifenvariante V
- Code vor der Schleife A, Code in der Schleife S
- Schleifenbedingung b

1. I gilt vor der Schleife: $wp(A, I) \Rightarrow P$ (oder `true`)
2. I gilt nach jedem Schleifendurchlauf: $I \wedge b \Rightarrow wp(S, I)$
3. Q gilt nach der Schleife: $I \wedge \neg b \Rightarrow Q$
4. Schleife terminiert:
   z. B. $I \Rightarrow V \geq 0$, $I \wedge b \wedge V = z \Rightarrow wp(S, V < z)$

1.-3. erfüllt: partiell korrekt, 4. erfüllt: total korrekt

(In der Klausur muss man keine Schleifeninvariante erraten, sondern nur aus gegebenen die richtige auswählen)

Geeignete Schleifeninvariante: Beispiel

Beispiel

```
public static int mul(int a, int b) {
  /* P: T */
  if (a < 0) {
    a = -a;
    b = -b;
  }
  int r = 0, x = a;
  while (x >= 1) {
    r = r + b;
    --x;
  }
  /* Q: r = a * b */
  return r;
}
```

Welche dieser Prädikate sind vor der Schleife erfüllt?

$\top$ **?** $\quad x \geq 0$ **?** $\quad r = a \cdot b$ **?** $\quad r = (a - x) \cdot b \wedge x \geq 0$ **?**

T (= True) ist vor der Schleife True, aber bringt nichts, weil wir dadurch keine Mehrinformation haben
X >= 0 ist True
r = a * b ist False
r = (a – x) * b ^ x >= 0 (vor der schleife ist (x = a => a-x = 0)) => r = 0 ^ x>= 0
(x = a und a ist wegen dem if immer > 0) => True

Bleiben nur noch 2 mögliche Schleifeninvarianten. Es muss noch geprüft werden, welche Q implizieren:

Da x >= 0 nichts über Q: r = a * b aussagt, fliegt diese auch raus.
Somit ist r = (a – x) * b ^ x >= 0 die Schleifeninvariante für dieses Beispiel.

Nun zu 4. Die Schleife terminiert:

Schleifenvariante

- zur Korrektheit einer Schleife gehört auch deren Terminierung

$\rightsquigarrow$ Schleife muss nach endlich vielen Durchläufen abbrechen

- um Schleifen-Terminierung zu zeigen: Schleifenvariante finden
- Schleifenvariante: (mathematische) Funktion V mit folgenden Eigenschaften:
  - V ist eine Funktion über den in der Schleife benutzten Variablen
  - V ist ganzzahlig
  - V ist streng monoton fallend
  - V ist nach unten durch eine Konstante c beschränkt
    - lies: sobald V den Wert c annimmt, bricht die Schleife ab

Aus dem Beispiel oben kann nur 'x' eine Schleifenvariante sein, weil es in der While-Bedingung steht.
Mit while(x >=1) und –x; ist erkennbar, dass x mit jedem Durchlauf dekrementiert und bei x = 1 die Schleife abbricht. Somit ist x eine gültige Schleifenvariante.
Und daraus folgt, dass die Schleife total korrekt ist.

Zum Üben:

https://www2.cs.fau.de/teaching/WS2012/AuD/organisation/oldexams/secure/13-02-21_klausur.pdf
Seite 18.

Lösung:
https://fsi.cs.fau.de/dw/pruefungen/bachelor/aud/loesungws12