

Informatik I Übung, Woche 39

Giuseppe Accaputo

25. September 2014

Plan für heute

1. Fragen & Nachbesprechung Übung 1
2. Zusammenfassung der bisherigen Vorlesungsslides
3. Tipps zur Übung 2

IF-THEN-ELSE

```
IF Bedingung  
THEN Anweisung;
```

► **In Worte:**

FALLS Bedingung erfüllt ist
DANN führe Anweisung aus

```
IF Bedingung  
THEN Anweisung_1  
ELSE Anweisung_2;
```

► **In Worte:**

FALLS Bedingung erfüllt ist
DANN führe Anweisung_1 aus
ANSONSTEN führe Anweisung_2 aus

Verschachtelte IFs

```
IF Bedingung_1  
THEN Anweisung_1  
ELSE IF Bedingung_2  
THEN Anweisung_2  
ELSE Anweisung_3;
```

► **In Worte:**

FALLS Bedingung_1 erfüllt ist

DANN führe Anweisung_1 aus

ANSONSTEN FALLS Bedingung_2 erfüllt ist

DANN führe Anweisung_2 aus

ANSONSTEN führe Anweisung_3 aus

(Bedingung_1 und Bedingung_2 sind nicht erfüllt)

Blöcke

- ▶ Sequenz von Anweisungen (zwischen **BEGIN** und **END**)
- ▶ Wird wie eine einzelne Anweisung verwendet

```
BEGIN
```

```
    preis := 12.10;  
    k := 0;
```

```
END
```

⇒ Block bestehend aus mehreren Anweisungen, wird jedoch als eine einzelne Anweisung interpretiert

```
preis := 12.10;  
k := 0;
```

⇒ zwei Anweisungen

Der Datentyp `BOOLEAN`

`VAR`

```
a : BOOLEAN ;
```

- ▶ mögliche Werte: `true` und `false`
- ▶ Vergleichsoperatoren: `=`, `<>`, `<`, `>`, `<=`, `>=`
- ▶ Weitere Operatoren: `and`, `or`, `xor`, `not`

Zu beachten bei IFs (1/2)

IF Bedingung

THEN Anweisung;

- ▶ Bedingung muss vom Typ **BOOLEAN** sein
 - ▶ Verwende nur **BOOLEAN** Variablen (z.B. **VAR a : BOOLEAN;**) oder Operatoren, welche ein Ergebnis vom Typ **BOOLEAN** zurückgeben (z.B. **not**, **or**, **and**, **<**, **>**, **=**, etc.)
- ▶ Anweisung (inkl. Anweisung_* bei **IF-THEN-ELSE** und verschachtelten **IFs**) kann eine einzelne Anweisung sein oder ein Block bestehend aus mehreren Anweisungen
- ▶ Platzierung des Semikolons ;

Zu beachten bei IFs (2/2)

program_1.pas (Ausschnitt):

```
IF a = 10 THEN
    b := 12;
ELSE
    b := 13
```

program_2.pas (Ausschnitt):

```
IF a = 10 THEN
    b := 12;
    c := 14
ELSE
    b := 13;
```

Boole'sche Operatoren

- ▶ `and`, `or`, `xor`, `not`
- ▶ Präzedenz:
 1. `not`
 2. `and`
 3. `or`, `xor`
- ▶ Funktionssignaturen:
 - ▶ `and`, `or`, `xor`:
`BOOLEAN` × `BOOLEAN` → `BOOLEAN`
 - ▶ `not`:
`BOOLEAN` → `BOOLEAN`

and

a **and** b

- ▶ Wenn a **und** b beide den Wert **true** haben, dann evaluiert a **and** b zu **true**, ansonsten zu **false**

a	b	a and b
true	true	true
true	false	false
false	true	false
false	false	false

or

a or b

- ▶ Wenn a **oder** b (oder beide gleichzeitig) den Wert **true** haben, dann evaluiert a **or** b zu **true**, ansonsten zu **false**

a	b	a or b
true	true	true
true	false	true
false	true	true
false	false	false

xor

a xor b

- ▶ Wenn a und b unterschiedliche Werte haben (z.B. a = `true`; und b = `false`;) , dann evaluiert a xor b zu `true`, ansonsten zu `false`

a	b	a xor b
true	true	false
true	false	true
false	true	true
false	false	false

not

not a

- ▶ Aus **true** wird **false** und aus **false** wird **true** (Negierung)

a	not a
true	false
false	true

Vergleichsoperatoren

Ordnung

- ▶ **INTEGER**, **REAL**: entsprechend numerischem Wert
 - ▶ $2.0 < 2.3$
- ▶ **STRING**: Lexikographisch
 - ▶ Sonderzeichen < Grossbuchstaben < Kleinbuchstaben
 - ▶ '@Hello' < 'world!' \Leftrightarrow true
 - ▶ 'world!' < 'Hello' \Leftrightarrow false
 - ▶ 'world_cup' < 'world_domination' \Leftrightarrow true
- ▶ **BOOLEAN**: false < true

Tipps zur Übung 2

- ▶ Aufgabe 2.1: Aufgabe aus einer alten Prüfung \implies löst sie!
- ▶ Aufgabe 2.2:
 - ▶ $u = 2 \cdot r \cdot \pi$
 - ▶ **CONST**-Richtlinie: “Denke bei jeder Variablen darüber nach, ob sie im Verlauf des Programms jemals ihren Wert ändert; ansonsten verwende eine Konstante.”
 - \implies **CONST**-korrektes Programm erfüllt diese Richtlinie