

# Informatik I Übung, Woche 42

Giuseppe Accaputo

15. Oktober, 2015

## Plan für heute

1. Lernziele für die heutige Übungsstunde
2. Nachbesprechung Übung 4
3. Vorbesprechung Übung 5

## Lernziele

- ▶ Normalisierung von Gleitkommazahlen
- ▶ **FUNCTION** und **PROCEDURE**
- ▶ Wert- und Referenzparameter
- ▶ Vor- und Nachbedingungen

## Aufgabe 4.1: Normalisierung von Gleitkommazahlen

Berechnung von  $957.8 + 50.37$  (4 Stellen für die Mantisse und 2 Stellen für den Exponenten):

$$\begin{array}{r}
 9\ 5\ 7\ 8\ |\ 0\ 2 \\
 +\ 0\ 5\ 0\ 3\ |\ 0\ 2 \\
 \hline
 1\ |\ 0\ 0\ 8\ 1\ |\ 0\ 2
 \end{array}$$

Bei der Normalisierung sollte nicht vorne abgeschnitten werden, sondern der Exponent so erhöht werden, dass die Zahl wieder in die Mantisse passt da ansonsten ein zu grosser Wertverlust entsteht:

$$1\ |\ 0\ 0\ 8\ 1\ |\ 0\ 2 \xrightarrow{\text{normalisieren}} 1\ 0\ 0\ 8\ |\ 0\ 3 = 1.008 \cdot 10^3$$

Der Rundungsfehler beträgt also  $1008.17 - 1008 = \underline{0.17}$

## Funktionen und Prozeduren: Warum?

- ▶ Code-Duplikation vermeiden
  - ▶ Mehrfach vorkommende Code-Ausschnitte können in eine Funktion/Prozedur zusammengefasst werden
  - ▶ Änderungen müssen nur noch zentral bei der Funktion/Prozedur vorgenommen werden
- ▶ Lesbarkeit des Codes verbessern
  - ▶ Andere Programmieren werden euch dankbar sein

## FUNCTION: Definition

Eine **FUNCTION**...

- ▶ ...hat einen Namen  
z.B. `isPrime`
- ▶ ...nimmt  $i = 0, \dots, n$  Variablen entgegen  
z.B. (`a : INTEGER`; `b : REAL`)
- ▶ ...gibt ein Resultat zurück, wessen Typ durch den Rückgabetyt definiert ist

## Syntax einer FUNCTION

```
FUNCTION Fkt (Var1 : Typ1; ...) : RTyp;  
VAR  
    i : Typ2;  
    ret : RTyp;  
BEGIN  
    i := ...;  
    ret := ...;  
    Fkt := ret;  
END;
```

## Syntax einer FUNCTION: Der Kopf

```
FUNCTION Fkt (Var1 : Typ1; ...) : RTyp;
```

- ▶ Kopf der Funktion
- ▶ Fkt ist der Name der Funktion
- ▶ (Var1 : Typ1; ...) ist die Liste der Parameter, welche die Funktion entgegennimmt.
  - ▶ Typ1 kann **INTEGER**, **REAL**, etc. sein
  - ▶ Kann beliebig viele Variablen enthalten, z.B. (Var1 : Typ1; ...; VarN : TypN;)
- ▶ RTyp ist der Rückgabetyt der Funktion, z.B. **INTEGER**, **REAL**, etc.
- ▶ Aufruf: res := Fkt(Wert1, ...)
  - ▶ **Wichtig:** Typen müssen passen (z.B. Wert1 muss vom Typ Typ1 sein)

## Syntax einer FUNCTION: Die Deklarationen

### VAR

```
i : Typ2;  
ret : RTyp;
```

- ▶ Deklarationen: Hier werden Variablen deklariert, welche nur in der Funktion `Fkt` verwendet werden können (lokale Variablen, auch Hilfsvariablen genannt)

## Syntax einer FUNCTION: Der Rumpf

```
BEGIN
```

```
  i := ...;
```

```
  ret := ...;
```

```
  Fkt := ret;
```

```
END;
```

- ▶ Hier können die lokalen Variablen, welche im Deklarationen-Teil definiert wurden verwendet werden
- ▶ `Fkt := ret;` muss immer angegeben werden, da dies das Resultat der Funktion ist welches zurückgegeben wird

## PROCEDURE: Definition

Eine **PROCEDURE** ist eine **FUNCTION** die keinen Rückgabewert hat, also auch nichts zurückgibt.

## PROCEDURE: Syntax

```
PROCEDURE Proc (Var1 : Typ1; ...);  
VAR  
    i : Typ2;  
    j : Typ3;  
BEGIN  
    i := ...;  
    j := ...;  
    ...  
END;
```

## Quiz-Frage 2

```
FUNCTION IsPrime (n : INTEGER) : BOOLEAN;  
VAR i : INTEGER;  
BEGIN  
    i := 2;  
    WHILE n mod i <> 0 DO Inc(i);  
    IsPrime := i = n;  
END;
```

### Frage:

- ▶ Funktionsname?
- ▶ Anzahl Variablen?
- ▶ Rückgabebetyp?

## Quiz-Frage 2

```
FUNCTION IsPrime (n : INTEGER) : BOOLEAN;  
VAR i : INTEGER;  
BEGIN  
    i := 2;  
    WHILE n mod i <> 0 DO Inc(i);  
    IsPrime := i = n;  
END;
```

### Antwort:

- ▶ Funktionsname: IsPrime
- ▶ Anzahl Variablen: 1
- ▶ Rückgabetyp: **BOOLEAN**

## Teamarbeit: Die myAND-Funktion definieren

**Aufgabe:** Definiert die Funktion myAND, welche zwei **BOOLEAN**-Variablen entgegennimmt und **TRUE** zurückgibt, falls beide **BOOLEAN**-Variablen **TRUE** sind und **FALSE** in allen anderen Fällen.

## Beispiel-Implementation der myAND-Funktion

```
FUNCTION myAND (a : BOOLEAN; b : BOOLEAN) :  
    ↪ BOOLEAN;  
BEGIN  
    myAND := a and b;  
END
```

## Wertparameter vs. Referenzparameter

Wertparameter	Referenzparameter
PROCEDURE P (i : REAL)	PROCEDURE P (VAR i : ↪ REAL)
P(n): n wird ausgewertet und übergeben	P(n): n wird als Referenz übergeben
i ist eine lokale Variable	i ist ein Alias für die Variable n (neuer Name für die gleiche Variable)
Darf bei FUNCTION und PROCEDURE verwendet werden	Darf nur bei PROCEDURE verwendet werden

## Analogie zu Wertparameter und Referenzparameter

Ich bin eine Funktion, welche ein leeres Blatt Papier als Parameter entgegen nimmt. Ihr übergibt mir nun ein leeres Blatt Papier.

### Fall **Wertparameter**:

- ▶ Ich kopiere das leere Blatt Papier und mache eigene Notizen darauf. Nach meiner Ausführung verwerfe ich das Blatt Papier mit den Notizen
- ▶ Ihr seid immer noch im Besitz des leeren Blatt Papier

### Fall **Referenzparameter**:

- ▶ Ich schreibe direkt auf euer beschriftetes Blatt Papier meine Notizen
- ▶ Nach meiner Ausführung seid ihr neu im Besitz eines Blatt Papiers, das meine Notizen enthält

## Warum Referenzparameter verwenden?

Stellt euch vor, ihr möchtet gerne mehrere Resultate zurückgeben.

**Problem:** `FUNCTION` erlaubt es, nur einen Wert zurückzugeben.

**Lösung:** Verwende eine Prozedur mit Referenzparametern, wobei die Referenzparameter verwendet werden können, um die verschiedenen Resultate zu speichern:

```
PROCEDURE P (VAR res1, res2, res3 : INTEGER)
    ↪ ;
BEGIN
    res1 := 1+2;
    res2 := 4 div 2;
    res3 := 4*5;
END
```

## Quiz-Frage 3

Frage: Was ist der Wert von *i* nach dem Aufruf von `MyF(i)`?

```
PROCEDURE MyF (VAR i : INTEGER);  
BEGIN  
    i := 3;  
END  
  
...  
BEGIN  
    i := 10;  
  
    MyF(i);
```

## Quiz-Frage 3

**Frage:** Was ist der Wert von  $i$  nach dem Aufruf von  $\text{MyF}(i)$ ?

```
PROCEDURE MyF (VAR i : INTEGER);  
BEGIN  
    i := 3;  
END  
  
...  
BEGIN  
    i := 10;  
  
    MyF(i);
```

**Antwort:**  $i$  hat den Wert 3 nach dem Aufruf von  $\text{MyF}(i)$ . Grund dafür ist, dass  $\text{MyF}$  einen Referenzparameter entgegen nimmt.