Agenda für heute, 4. März, 2010

- **Datentypen: integer, char, string, boolean**
- Bedingte Programmausführung
- Zusammengesetzte if-then-else-Anweisungen

Lesen Sie den Begleittext Kapitel 3.

Datentypen

- Pascal ist eine ***streng typisierte*** Programmiersprache
- Für jeden Speicherplatz muss ein ***Datentyp*** (Datenformat) definiert werden.
- Nicht dem Datentyp entsprechende Verwendung der Speicherplätze führt zu ***Fehlermeldungen***.

Wie werden Daten dargestellt?

Datenstruktur	Eigenschaft	Laufzeit
Literal	der zu verwendende ***Wert*** selbst	nicht veränderbar
Konstante	ein ***Bezeichner*** (Name), der einen Wert repräsentiert	nicht veränderbar
Variable	ein ***Bezeichner***, dem Werte ***zugewiesen*** werden können	veränderbar

Siehe Begleittext, Seiten 9, 10 & 89

Wie werden Variablen und Konstanten definiert?

- In der ***Variablendeklaration*** werden Variablen benannt und über den ***Datentyp*** wird definiert, welche Werte eine Variable annehmen kann.

```
var  x, summe, quadrat: integer;
```

- Jede Variable eines Programms muss ***deklariert*** werden.

- Der Datentyp von Konstanten ist ***implizit*** durch das ihnen zugeordnete Literal gegeben.

```
const  a = 20;  pi = 3.1415;  quit = 'q';
```

Wie werden Variablen verändert?

Der Wert einer Variablen kann verändert werden durch:

a) Zuweisungsanweisung

```
Variable := Ausdruck;
```

b) Parameter einer Prozedur oder Funktion

```
Prozedur(Variable);
```

```
const  a = 20;
var  x, summe, quadrat: integer;

     read(x);
     summe:= x + 5 * a;
     quadrat:= sqr(x);
```


Der Datentyp integer

```
Var i: integer;  j: byte;
```

Shortint:	-128 . . 127	Turbo Pascal
Integer:	-32768 . . 32767	
Longint:	-2'147'483'648 . . 2'147'483'647	
Byte:	0 . . 255	
Word:	0 . . 65535	

Symbol	Operation
+	Addition
–	Subtraktion
*	Multiplikation
DIV	ganzzahlige Division
MOD	Rest einer ganzzahligen Division

Arithmetische Ausdrücke

Wir wollen: Fläche und Diagonale von Rechtecken berechnen

```
Var  flaeche, laenge, breite, diag: integer;

read(laenge); read(breite);
flaeche:= laenge * breite;

{Diagonale berechnen}
laenge:= sqr(laenge);   breite:= sqr(breite);
diag:= sqrt(laenge + breite);
```

oder

```
diag:= sqrt(sqr(laenge) + sqr(breite));
```


Der Datentyp char (Abk. für character)

```
Var ch: char;
```

- Für die Speicherung und Verarbeitung von Textzeichen
- Wertebereich: Zeichensatz definiert durch ANSI-Standard

Symbol	Operationen
ORD	Ordinalwert
CHR	Textzeichen für gegebenen Ordinalwert
=, >, <	Vergleichsoperationen
<=, >=, <>	Vergleichsoperationen

Der Datentyp char, Beispiele

- Textzeichen sind normalerweise Symbole des Quellprogramms
- Apostrophiert werden sie zu Literalen für Textdaten

```
const  A = 'A';

var  ch: char;
     ch:= 'A'; ch:= A;
```

'B' < 'C' < 'c'

`Ord('k')` = 107 `Ord('K')` = 75

`Chr(107)` = 'k' `Chr(75)` = 'K' = `Chr(Ord('k')+32)`

Der Datentyp string

```
var
   S1: string;        enthält bis zu 255 Zeichen
   S2: string[12];    enthält bis zu 12 Zeichen
```

Erlaubt die Verarbeitung von Zeichenfolgen:

```
S:= 'Geben Sie ein Datum ein: ';
write(S)
```


Textverarbeitung mit `string`-Variablen

Die Variable `S[i]` stellt das i-te Zeichen von `S` dar.

Die Anweisung `S[3]:= 'C'` weist dem dritten Zeichen von `S` den Wert C zu.

Der + Operator dient der Verkettung von Zeichenfolgen

Operationen mit Strings

```
const  ort = 'Zuerich';
var S1, S2: string;
```

Zuweisungsanweisungen:

```
S1:= 'ETH ';
S2:= ort;
```

Verketten von String-Operanden mit + :

```
S1:= S1 + S2;
```

S1 enthält jetzt den Text: ETH Zuerich

Der Datentyp boolean

Wertebereich: die Wahrheitswerte "TRUE" und "FALSE"

Operationen

NOT Negation
AND Konjunktion
OR Disjunktion
XOR exklusive Disjunktion

Logische Ausdrücke

```
Var   laenge, breite: integer;
      kuerzer: boolean;

kuerzer:= breite < laenge;
```

Die Variable *kuerzer* enthält den Wert "TRUE" falls *breite* kleiner als *laenge* ist

Den gleichen Wert erhält man mit:

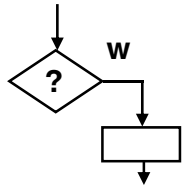
```
kuerzer:= not (breite > laenge);
```


- Datentypen: integer, char, string, boolean
- **Bedingte Programmausführung**
- Zusammengesetzte if-then-else-Anweisungen

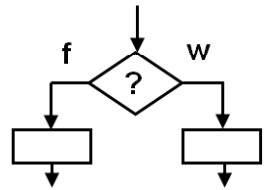
Bedingte Programmausführung

Falls *laenge* länger als *breite* dann drucke *laenge* aus

```
if laenge > breite
  then write(laenge)
```




```
kuerzer:= breite > laenge
if not kuerzer
then write(laenge)
else   begin
       write('laenge ist kürzer');
       read(laenge, breite)
  end;
```



Bedingte Programmausführung, zusammengesetzte Bedingungen

Wir wollen Kleinbuchstaben durch Grossbuchstaben ersetzen:

```
if Ord(ch) >= 97
  then ch:= Chr(Ord(ch)-32)
```

Verwendung logischer Ausdrücke als Bedingung:

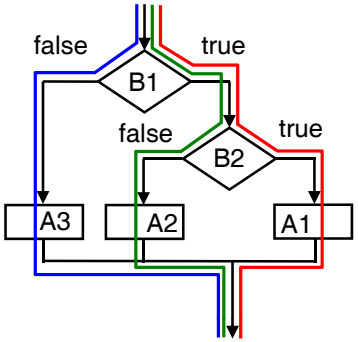
```
if (Ord(ch) >= 97) and (Ord(ch) < 123)
  then ch:= Chr(Ord(ch)-32)
```


- Datentypen: integer, char, string, boolean
- Bedingte Programmausführung
- **Zusammengesetzte if-then-else-Anweisungen**

Zusammengesetzte if-then-else-Anweisungen

Die **then**- und die **else**-Klausel einer **if**-Anweisung können beliebige Anweisungen enthalten, insbesondere auch **if**-Anweisungen:

```
if Bedingung1
   then if Bedingung2
           then Anweisungsfolge1
           else Anweisungsfolge2
   else Anweisungsfolge3
```



Mit "geschachtelten" **if**-Anweisungen Können mehr als zwei Verzweigungen programmiert werden.

Zu welchem if gehört das else?

Ein Nachteil von Pascal: kein Abschlusssymbol für if-then-else

```
if Bedingung1
then if Bedingung2
then Anweisungsfolge1
else Anweisungsfolge2
```

Zu welchem "if" gehört das"else"

Regel: die else-Klausel gehört zum nächstliegenden "if" das kein "else" hat

```
if B1
 then if B2
        then A1
        else A2
```