PGdP Woche #3 – Diverses (v.a. Mathe)

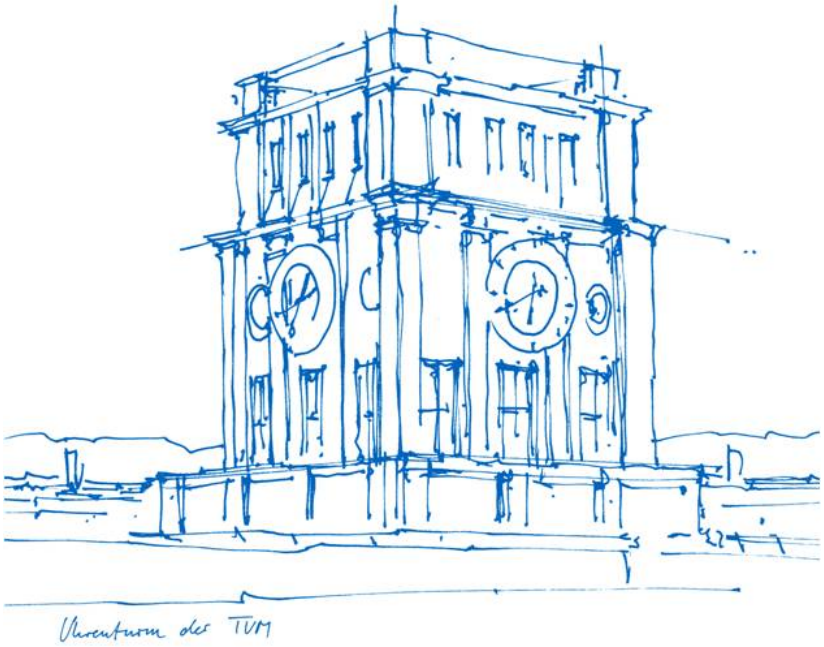
Sebastian Oßner – ossner@in.tum.de

Technische Universität München

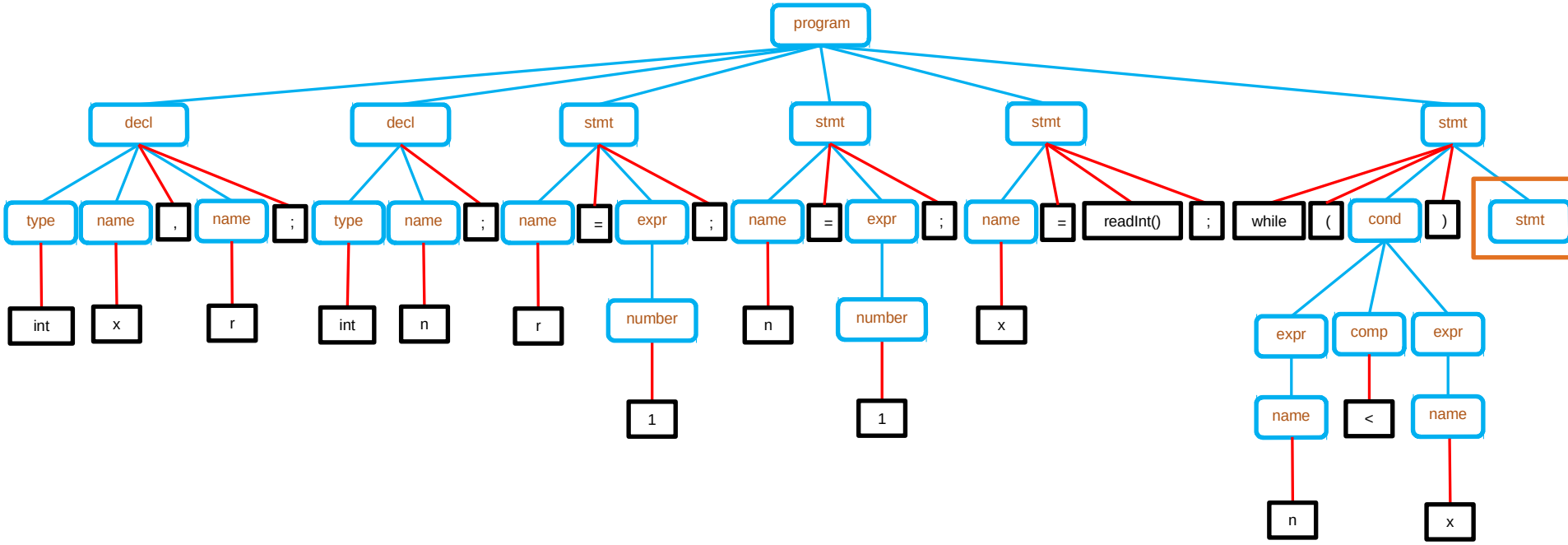
Garching, 4.November 2019

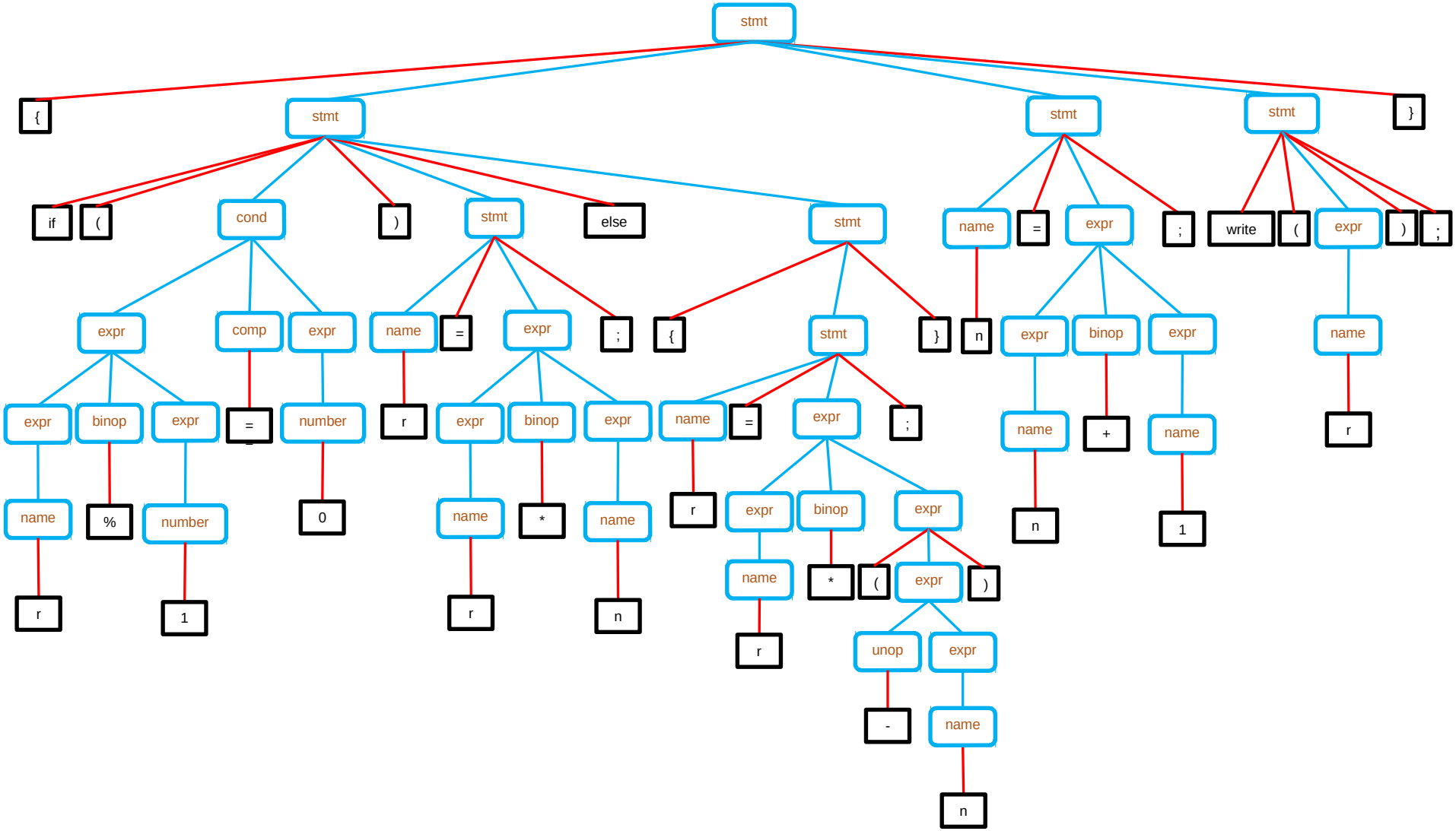
Ablauf:

1. P01 – Syntaxbaum
2. P02 – Binary-RegEx
3. P04 – Pascal
4. P03 – Palindrome



```
program  ::=  decl* stmt*
decl     ::=  type name ( , name )* ;
type     ::=  int
cond     ::=  true | false | ( cond ) |
              expr comp expr |
              bunop cond | cond bbinop cond
comp     ::=  == | != | <= | < | >= | >
bunop    ::=  !
bbinop   ::=  && | ||
stmt     ::=  ; | { stmt* } |
              name = expr; | name = readInt(); | write( expr ); |
              if ( cond ) stmt |
              if ( cond ) stmt else stmt |
              while ( cond ) stmt
expr     ::=  number | name | ( expr ) |
              unop expr | expr binop expr
unop     ::=  -
binop    ::=  - | + | * | / | %
```


program
decl
decl
stmt
stmt
stmt
stmt
type
name
,
name
;
type
name
;
name
=
expr
;
name
=
expr
;
name
=
readInt()
;
while
(
cond
)
stmt
int
x
r
int
n
r
number
n
number
x
expr
comp
expr
1
1
name
<
name
n
x


stmt
{
stmt
stmt
stmt
}
if
(
cond
)
stmt
else
stmt
name
=
expr
;
write
(
expr
)
;
expr
comp
expr
name
=
expr
;
{
stmt
}
n
expr
binop
expr
name
expr
binop
expr
expr
binop
expr
=
number
r
expr
binop
expr
name
=
expr
;
name
+
name
r
name
%
number
0
name
*
name
r
expr
binop
expr
n
1
r
1
r
n
name
*
(
expr
)
r
unop
expr
-
name
n

Regular Expressions (Reguläre Ausdrücke)

Lösen Fundamentales Problem der Informatik: Pattern-Matching

-> String-Extraktion, String-Replacement, Formvalidierung

Beispiele:

[a-z]{2}[0-9]{2}[a-z]{3}

Regular Expressions (Reguläre Ausdrücke)

Lösen Fundamentales Problem der Informatik: Pattern-Matching

-> String-Extraktion, String-Replacement, Formvalidierung

Beispiele:

```
[a-z]{2}[0-9]{2}[a-z]{3}
```

ge93kex, ge38yuc, ge38xag, ge35qac, ge38jiz, ge93luc, ge49bot, ge35voz, ge38man

```
^(?:https?:\/\/)?(?:www\.)?youtu(?:\.be|be\.com)\/(?:watch\?v=)?([\w-]{10,})$
```

Regular Expressions (Reguläre Ausdrücke)

Lösen Fundamentales Problem der Informatik: Pattern-Matching

-> String-Extraktion, String-Replacement, Formvalidierung

Beispiele:

```
[a-z]{2}[0-9]{2}[a-z]{3}
```

ge93kex, ge38yuc, ge38xag, ge35qac, ge38jiz, ge93luc, ge49bot, ge35voz, ge38man

```
^(?:https?:\/\/)?(?:www\.)?youtu(?:\.be|be\.com)\/(?:watch\?v=)?([\w-]{10,})$
```

https://www.youtube.com/watch?v=dQw4w9WgXcQ

www.youtube.com/watch?v=dQw4w9WgXcQ

youtube.com/watch?v=dQw4w9WgXcQ

https://www.youtu.be/dQw4w9WgXcQ

youtu.be/dQw4w9WgXcQ

Binary RegEx

Gültige Binärzahlen:

```
0b0101
0B100
0b1001_1011
0B011_0100
0b0000_0111_0110_1111
0b10_____010
```

Ungültig:

```
0_b0110
0b_0110
0b0110_
```

Verwendbare Ausdrücke:

- |
- *
- ?

Binary RegEx

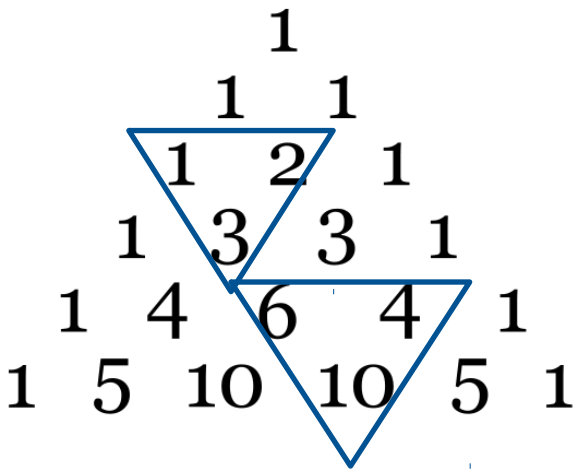
Musterlösung:

```
0(b|B)((0|1)(0|1)*_*)*(0|1)
Fängt immer mit 0 an

Danach b oder B

(mindestens eine Binärzahl
Gefolgt von beliebig vielen Unterstrichen) beliebig oft

Gefolgt von einer Binärzahl
```

Pascal

```
          1
        1   1
      1   2   1
    1   3   3   1
  1   4   6   4   1
1   5  10  10   5   1
```

Pascal

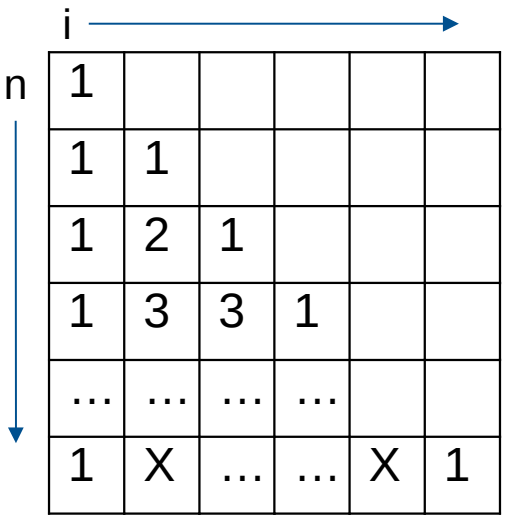


1. Die Anzahl der Elemente von Zeile *n* ist *n* + 1.

2. Die erste und letzte Zahl jeder Zeile ist stets die

3. Das *i*-te Element der Zeile n entspricht der Summe des *i*-ten und des (*i* - 1)-ten Elements der Zeile (*n* - 1)

Tipp: Vorstellung als 2d-Array:



X := array[n-1][i-1] + array[n-1][i]

Pascal - Musterlösung

```java
public static int[][] pascalDreieck(int n) {
    // Deklarieren des Ergebnisfeldes.
    int[][] dreieck = new int[n][];
    // Schleife zur Berechnung der einzelnen Zeilen.
    for (int m = 0; m < n; m++) {
        dreieck[m] = new int[m + 1];
        dreieck[m][0] = 1;
        dreieck[m][m] = 1;
        // Schleife zur Berechnung der Elemente der jeweiligen Zeile.
        for (int i = 1; i < m; i++)
            dreieck[m][i] = dreieck[m - 1][i - 1] + dreieck[m - 1][i];
    }
    return dreieck;
}
```

Palindrome

Lagerregal

Palindrome

Lagerregal
Racecar

Palindrome

Lagerregal

Racecar

Dammit I’m mad.
Evil is a deed as I live.
God, am I reviled? I rise, my bed on a sun, I melt.
To be not one man emanating is sad. I piss.
Alas, it is so late. Who stops to help?
Man, it is hot. I’m in it. I tell.
I am not a devil. I level “Mad Dog”.
Ah, say burning is, as a deified gulp,
In my halo of a mired rum tin.
I erase many men. Oh, to be man, a sin.
Is evil in a clam? In a trap?
No. It is open. On it I was stuck.
Rats peed on hope. Elsewhere dips a web.
Be still if I fill its ebb.
Ew, a spider… eh?
We sleep. Oh no!
Deep, stark cuts saw it in one position.
Part animal, can I live? Sin is a name.
Both, one… my names are in it.
Murder? I’m a fool.
A hymn I plug, deified as a sign in ruby ash,
A Goddam level I lived at.
On mail let it in. I’m it.
Oh, sit in ample hot spots. Oh wet!
A loss it is alas (sip). I’d assign it a name.
Name not one bottle minus an ode by me:
“Sir, I deliver. I’m a dog”
Evil is a deed as I live.
Dammit I’m mad.

Palindrome

Zahlenpalindrome: `101, 123321, 12321`, etc.

- User nach Zahleninput `>=0` fragen
- Berechnen ob die eingegebene Zahl ein Palindrom ist
- Ausgeben des Resultats

Tipps:

- Zählen der Ziffern von n
- Eingabe in ein Array lesen
- Über Array iterieren und erste und letzte Zahl vergleichen
- Zählen wie oft Zahlen NICHT Übereinstimmen
- Zurückgeben/Abbrechen