PGdP Tutorium: Dritte Stunde

Benedikt Werner

München, 07. November 2017

Organisatorisches

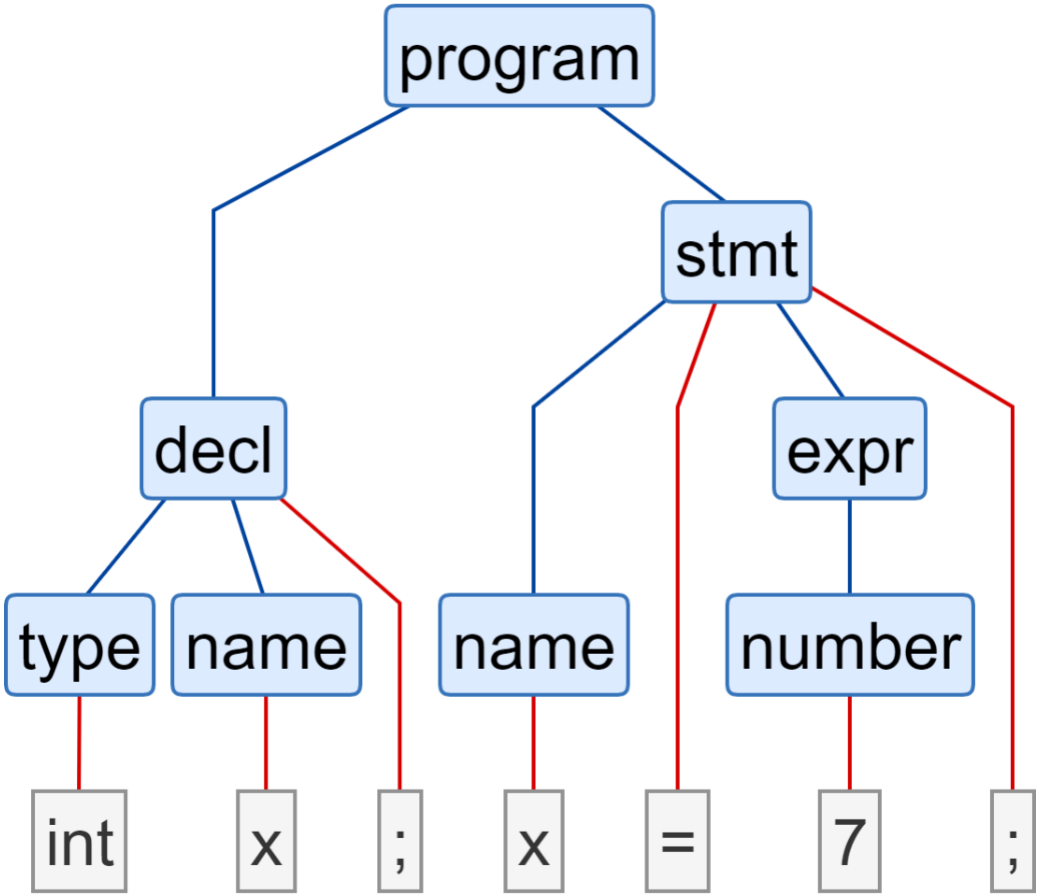
- Website: home.in.tum.de/~wernerbe/pgdp

- Hausaufgabenabgaben
  - Nur .java Dateien abgeben!
  - Ab Blatt 4 muss der Google Style Guide eingehalten werden!
    - Anleitung zur Autoformatierung auf der Website
  - Abgabe nach der Deadline **nicht** mehr bearbeiten!!
  - Hausaufgaben in der Hauptinstanz abgeben
  - Keine Packages benutzen

Syntaxbäume

- MiniJava Grammatik von Moodle herunterladen

```
int x;
x = 7;
```



Aufgabe 3.1

```
int prod, x, n;
x = read();
if (0 < x) {
        prod = 1;
        n = 0;
        while (prod <= x) {
                n = n + 1;
                prod = prod * (-n);
        }
        write(prod);
} else {
        write(n);
}
```

Kontrollflussdiagramme

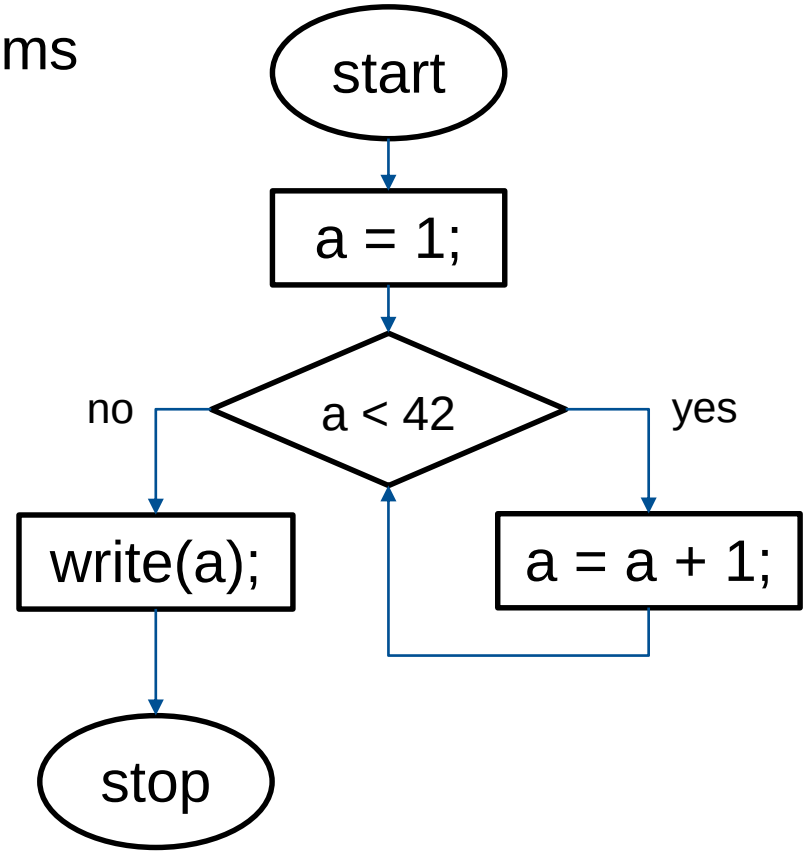
- Zeigen den Ablauf eines Programms

Start: start

Ende: stop

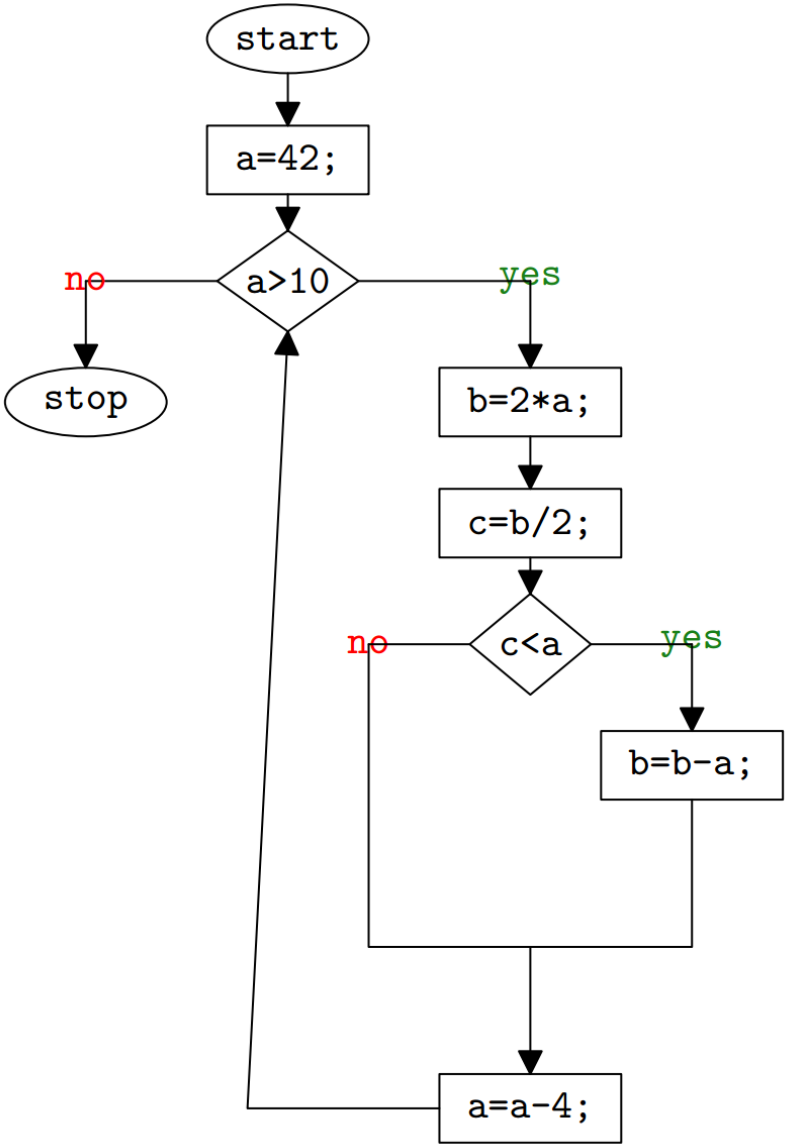
Statement: a = 42;

Condition: a == 5



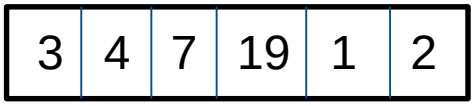
Aufgabe 3.2

```
int a, b, c;
a = 42;
while (a > 10) {
        b = 2*a;
        c = b / 2;
        if (c < a) {
                b=b-a;
        }
        a = a - 4;
}
```



Arrays

- „Box“ um **feste** Anzahl von Werten zu speichern



- Array erstellen:
```
int[] meinArray = new int[7];
```
- Wert setzen:
```
meinArray[3] = 42;
```
- Wert lesen:
```
meinArray[4]
int x = meinArray[4];
write(meinArray[4]);
```
- Länge des Arrays:
```
meinArray.length
```

Aufgabe 3.3

Schreiben Sie ein MiniJava-Programm namens `MinMax.java`, das in einem Array von ganzen Zahlen den kleinsten und den größten Wert findet. Das Programm soll sich wie folgt verhalten:

- Zunächst fragt das Programm ab, wie viele Zahlen in das Array eingegeben werden sollen.
- Dann werden die Zahlen eingegeben und in einem Array gespeichert.
- Anschließend wird das Array durchsucht und in **einem Durchgang** soll sowohl die kleinste als auch die größte Zahl gefunden werden.
- Schließlich sollen die kleinste und die größte Zahl ausgegeben werden.

Aufgabe 3.3 Lösung

```
int laenge, i, min, max;

laenge = read("Laenge?");
int[] array = new int[laenge];

i = 0;
while (i < laenge) {
        array[i] = read("Zahl"+(i+1));
        i++;
}
```

```
min = max = array[0];
i = 0;
while (i < laenge) {
        if (array[i] < min)
                min = array[i];
        if (array[i] > max)
                max = array[i];
        i++;
}
write("Max: " + max);
write("Min: " + min);
```

Aufgabe 3.4

Das Pascalsche Dreieck wird Schritt für Schritt, beginnend mit Zeile 0, aufgebaut. Dazu berechnet man die *n*-te Zeile aus der (*n* - 1)-ten Zeile wie folgt:

- Die Anzahl der Elemente von Zeile *n* ist *n* + 1.
- Die erste und letzte Zahl jeder Zeile ist stets die 1.
- Das *i*-te Element der Zeile *n* entspricht der Summe des *i*-ten und des (*i* - 1)-ten Elements der Zeile (*n* - 1).

Schreiben Sie ein MiniJava-Programm, welche das Pascalsche Dreieck berechnet. Das Programm soll dabei zunächst die Anzahl der Zeilen des Dreiecks vom Benutzer einlesen; gibt die Benutzerin eine negative Anzahl Zeilen ein, soll sie zur Wiederholung der Eingabe aufgefordert werden. Anschließend soll das Pascalsche Dreieck in einem Array berechnet und schließlich ausgegeben werden.

Hinweis: Für die Anzahl der Einträge *e* im Dreieck mit *m* Zeilen gilt:

$$e = \sum_{i=1}^{m}(i) = \frac{m*(m+1)}{2}$$

Aufgabe 3.4 Lösung

```
int m = read("Zeilen?");
while (m < 0) {
    m = read("Zeilen? (>= 0)");
}

int[] d = new int[m*(m+1)/2];
int z = 0, i = 0;
while (z < m) {
    int end = i + z; // Letze Zelle der Zeile
    d[i] = 1;        // Nulltes Element = 1.
    d[end] = 1;      // Letztes Element = 1.
    i++;
    while (i < end) {
        d[i] = d[i-z] + d[i-z-1];
        i++;
    }
    i = end + 1;
    z++;
}
```

```
z = i = 0;
while (z < m) {
    int end = i + z;
    writeConsole("n = " + z + "\t");
    while (i <= end) {
        writeConsole(d[i] + "\t");
        i++;
    }
    writeLineConsole();
    z++;
}
```