PGdP

MiniJava

Aufbau von Klassen

Kommentare

Syntax

Kontrollfluss

Basistypen

P-Aufgaben

Debugging

Folien: go.tum.de/904005

MiniJava

Deklaration von Variablen

```
int x, y;
```

Zuweisung von Werten

```
x = 5;  //Variable x wird Wert 5 zugewiesen
y = x + 2;  //y wird x + 2 (7) zugewiesen
x = x + 1;  //x wird um 1 inkrementiert (6)
```

MiniJava

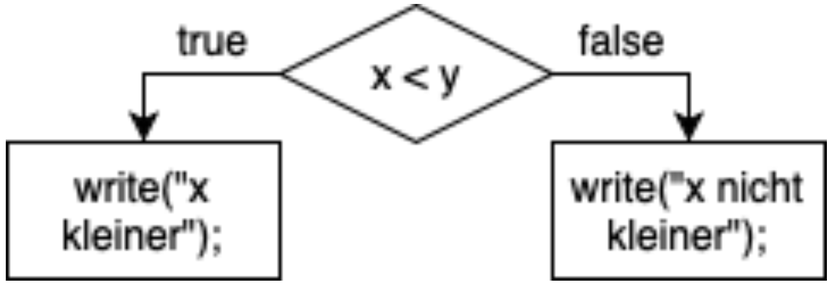
Einlesen von Werten über die Konsole

```
int x;
x = readInt();
x = readInt("text");
```

Ausgabe von Werten auf die Konsole

```
write(x);
write("text");
writeConsole(x); //ohne Zeilenumbruch
```

MiniJava



Selektion

```
int x, y;
x = readInt(); //Einlesen von x und y
y = readInt();
if(x < y)    //Überprüfen der Bed.
  write("x ist kleiner");
else
  write("x ist nicht kleiner");
```

MiniJava

Selektion

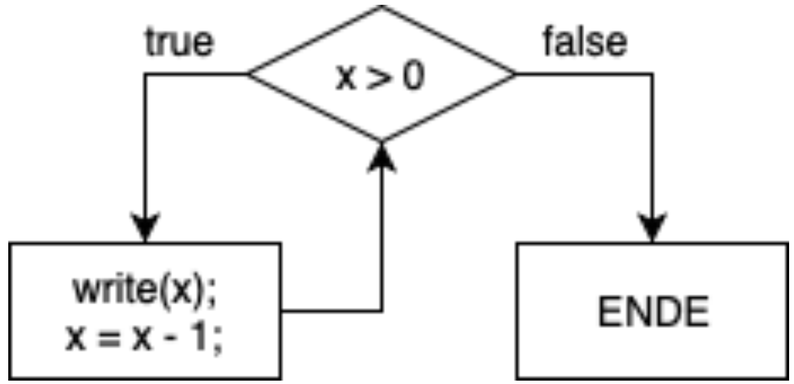
```
int x, y;
x = readInt(); //Einlesen von x und y
y = readInt();
if(x < y){  //Überprüfen der Bed.
  write("x ist kleiner");
}
else {
  write("x ist nicht kleiner");
}
```

Nur nötig, falls mehr als eine Operation auf if/else folgt.

MiniJava

Iteration

```
int x;
x = readInt();
while(x > 0){  //Überprüfen der Bedingung
  write(x);  //Ausgabe von x
  x = x - 1;  //Dekrementiern von x
}
```



MiniJava

Arrays

```
int[] a = new int[4]; //Array d. Länge 4 dekl.
a[0] = 2;  //Zuweisung an 1. Stelle des Arrays
a[1] = 5; a[2] = 1;
a[3] = a[0] + a[1] + a[2];
```

Index	0	1	2	3
Wert	2	3	1	6

Arrays werden von 0 bis $\text{length} - 1$ indiziert.

Aufbau von Klassen

```
public class Test extends MiniJava {
  public static void main(String[] args){
    //"Hauptprogramm"

  }
  //weitere Methoden
}
```

Aufbau von Klassen

```
public class Test extends MiniJava {
   public static void main(String[] args){
      //"Hauptprogramm"


   }
   //weitere Methoden
}
```

Vererbung: So bekommt man die speziellen MiniJava Methoden. (später mehr dazu)

Kommentare

Wir Kommentieren den Code, damit das Programm für uns & andere übersichtlicher und leichter zu lesen ist.

Einzeiliger Kommentar

```
//Kommentar
```

Mehrzeiliger Kommentar

```
/* Kommentar über
   mehrere Zeilen */
```

Syntax

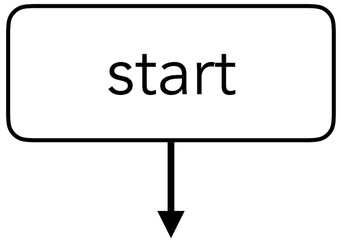
Beliebte Klausuraufgaben

- Syntaxbäume
- Programme auf Korrektheit prüfen

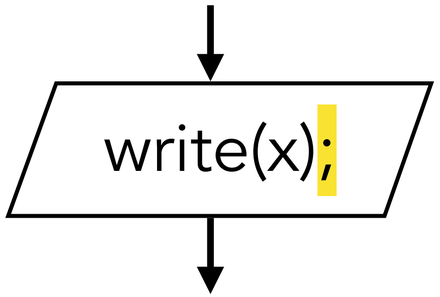
Übersicht (von SS2011)

Kontrollfluss

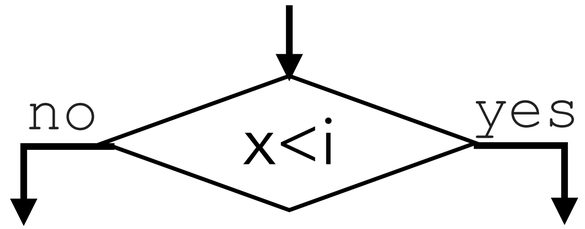
Beliebte Klausuraufgabe: Kontrollflussdiagramme



Startknoten

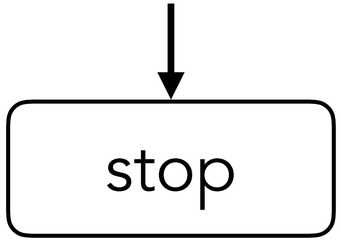


Ausgabe

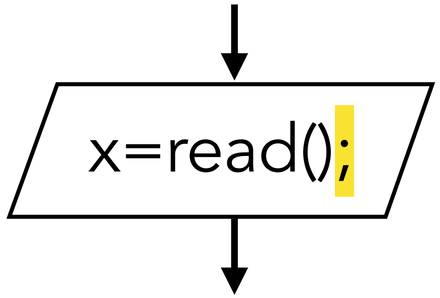


Bedingte Verzweigung

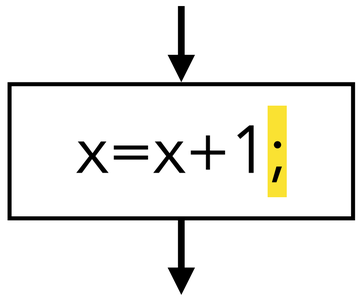
Endknoten



Eingabe



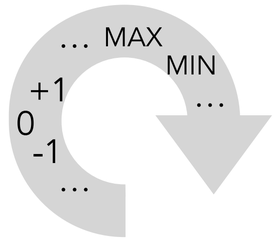
Zuweisung

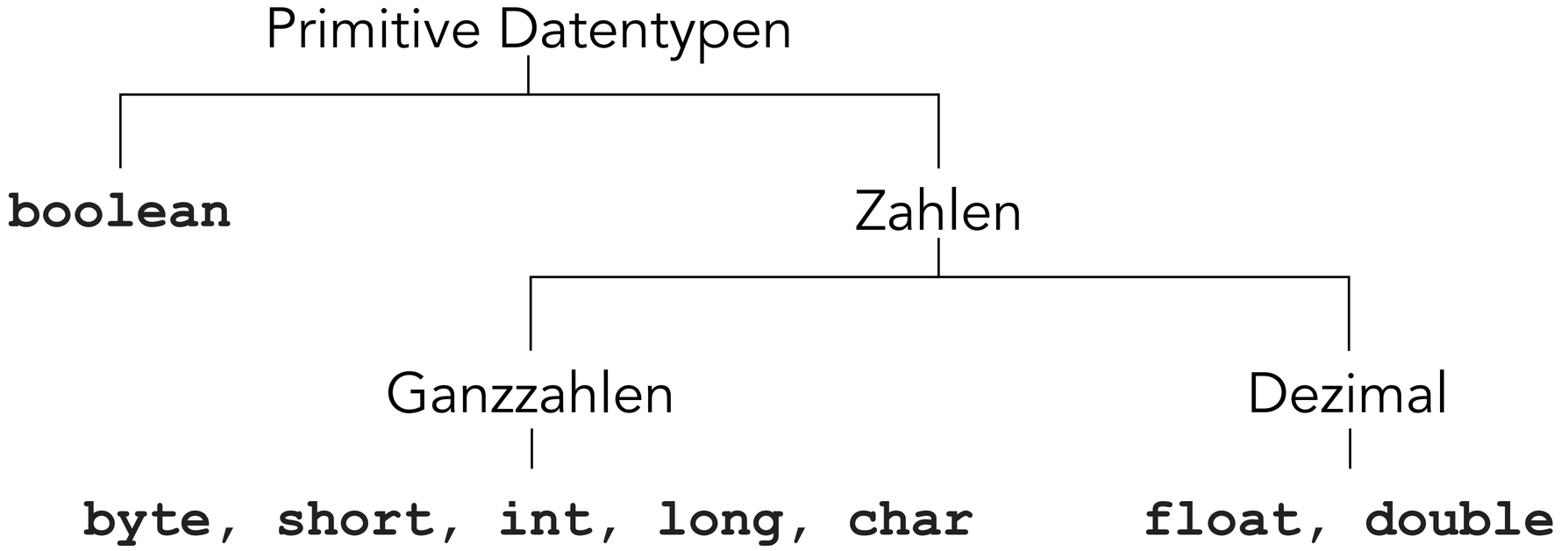


Basistypen

Achtung! Überlauf von Integer:

```
Integer.MAX_VALUE + 1 = Integer.MIN_VALUE
```



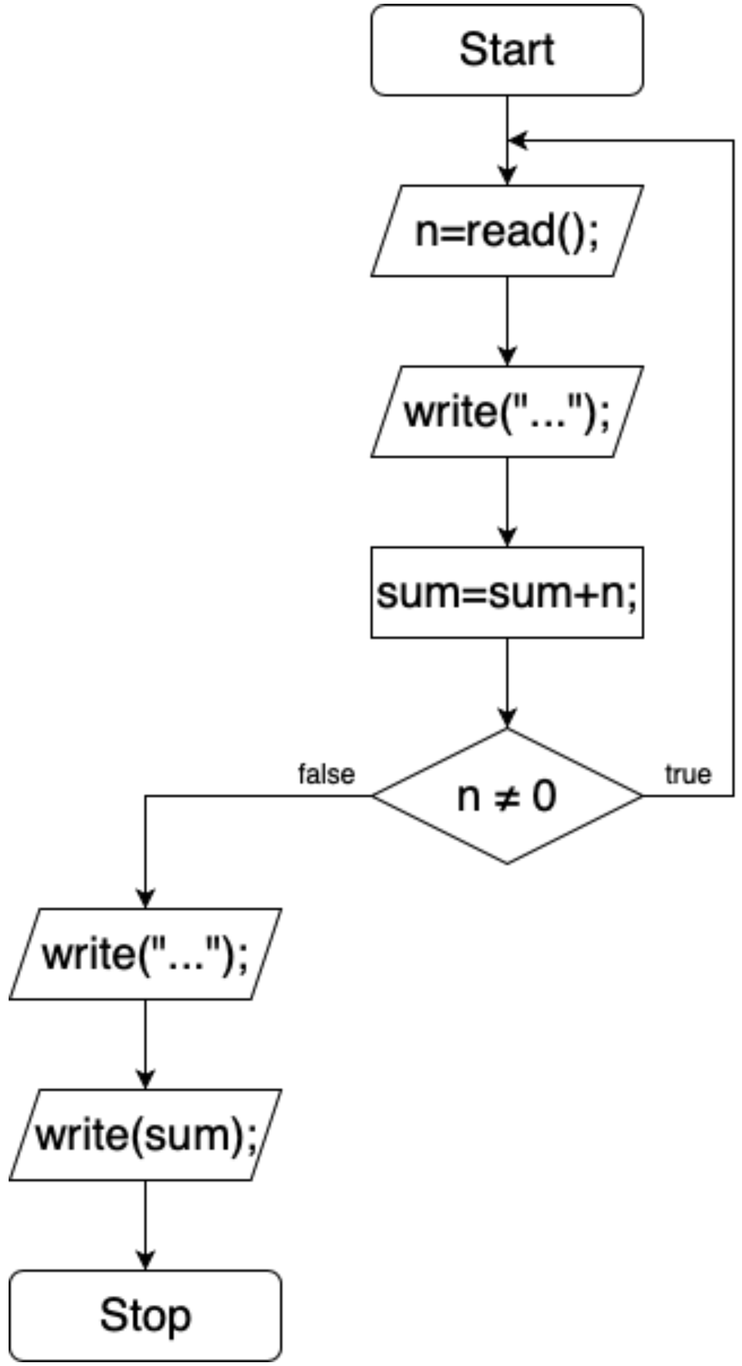


P02.1

Do-While Schleife

```
do {
   //Code
} while(cond);
```

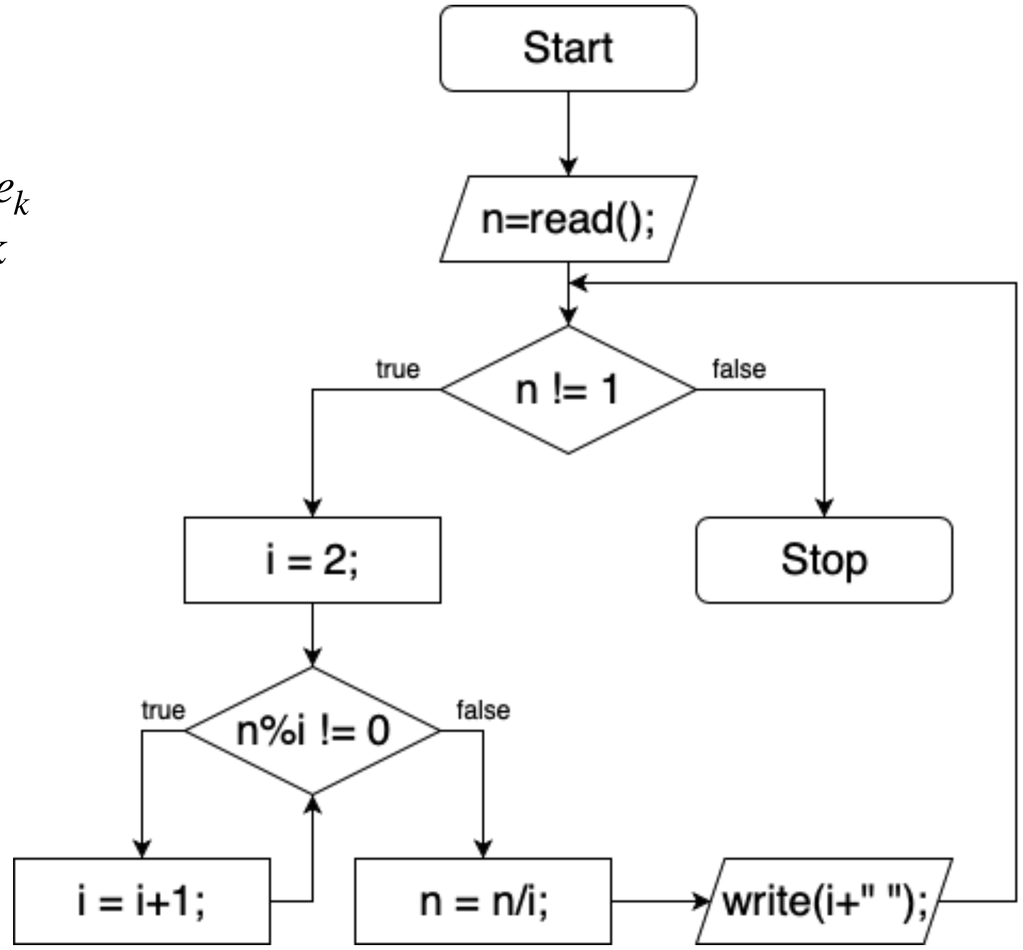
Die erste Iteration erfolgt ohne Prüfung der Bedingung.



P02.2

Primfaktorzerlegung: Darstellung von $n \in \mathbb{N}$ als Produkt von Primzahlen ($\mathbb{P}$).

$$n = p_1^{e_1} \cdot \ldots \cdot p_M^{e_M} = \prod_{k=1}^{M} p_k^{e_k}$$



P02.3

Karnickelpopulation

- 3 Generationen
- Jede Generation bringt in jedem Zyklus (Monat) pro Paar ein neues Paar in *Generation 1*

Monat	1. Gen	2. Gen	3. Gen
1	1	0	0
2	1	1	0
3	2	1	1
4	4	2	1
5	7	4	2

T_n	T_{n+1}	Gen
j	j+a+m	J
m	j	M
a	m	A

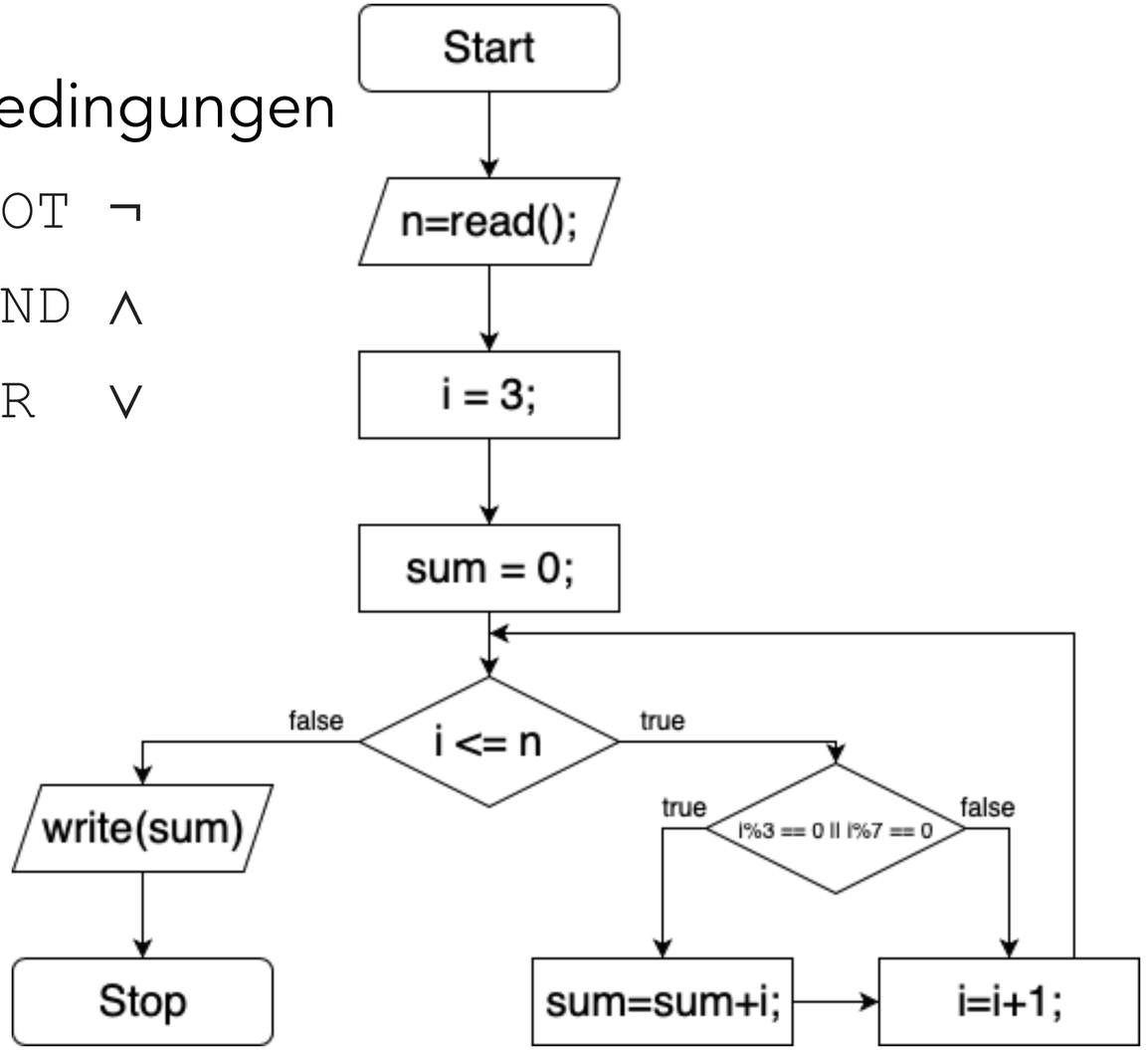
P01.4

NOT, AND und OR bei Bedingungen

```
!a      //logisches NOT ¬
a && b //logisches AND ∧
a || b //logisches OR  ∨
```

&&	1	0
1	1	0
0	0	0

||	1	0
1	1	1
0	1	0



Debugging