```c
#include <stdlib.h>
#include <stdio.h>

int main(){


    printf("Hello World!\n");

    getchar();
    return 0;
}
```

Strukturierte Programmentwicklung

```
myjday@mobian:~/programmingC/introC$ ./HelloWorld
Hello World!
```

… Ok, ich will es richtig lernen ....
Wie gehst Du beim Programmieren vor?

Zu aller erst überlegst Du Dir: **Um was geht es eigentlich?**

Welche Größen sind beteiligt?

Welche Bedingungen gelten?

Wir nennen diesen ersten Schritt:

1. Analyse eines gegebenen Problems

… hey Progman, ein Beispiel?

… ok, pass auf:

Zähle alle positiven Zahlen bis zu einer bestimmten Grenze zusammen. Gib das Ergebnis an.

```
Startwert: 1
Stopwert: 10
1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10 = 55
```

… ok, ich probiers ….

… mathematisch ausgedrückt: $f: \mathbb{N} \to \mathbb{N}\ mit\ f(n) = \sum_{i=1}^{n} i\ \ für\ n \in \mathbb{N}$

Gegeben sei eine Natürliche Zahl n.
Addiere die Natürlichen Zahlen von 1 bis n.
Die Summe ist das Resultat.

Beteiligte Größen: **Startwert → 1**

Grenze → n (bestimmt der Benutzer)

Summe → die aufaddierten Ganzzahlen

Bedingungen: **addiere solange der Grenzwert nicht überschritten ist.**

… absolut korrekt. Du hast das Problem umgangssprachlich beschrieben. Jetzt geht's an die Realisierung ….

… jetzt geht's los! In welcher Sprache? Java, C++, C#, Perl, PHP, C, Delphi ???

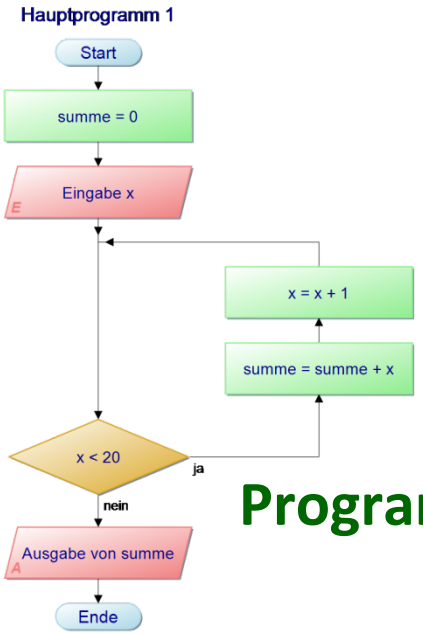
… im Moment in gar keiner. Das ist der Fehler den die meisten Möchtegernproggies machen. Einfach drauf los und wenn's nicht klappt ist die Programmiersprache einfach Mist ….

Der nächste Schritt lautet:

2. Entwickeln einer Problemlösungsbeschreibung

… habe ich doch schon in Schritt 1 gemacht, oder ?

… nein, Du hast das Problem mit seinen Bedingungen erkannt,
aber noch keine Lösung beschrieben wie Du zur Summe kommst



Du hast zwei Möglichkeiten

Programmablaufplan (PAP)

Struktogramm

summe	← 0
x	← Benutzereingabe
wiederhole solange x < 20	
summe ← summe + x	
x	← x + 1
ausgabe summe	

… wo ist denn da der Unterschied ?

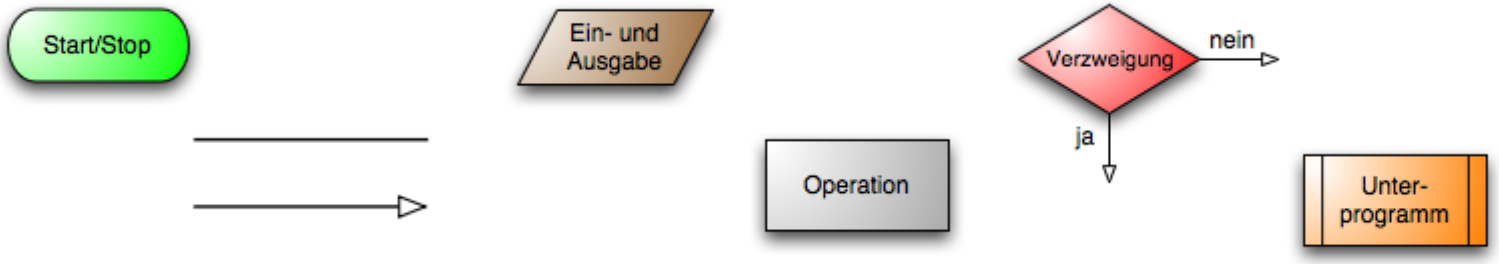
… der Unterschied liegt in der Art der Darstellung, beide stellen den strukturierten Programmablauf dar ….

Programmablaufplan

Ein **Programmablaufplan** (**PAP**) ist ein Ablaufdiagramm für ein Computerprogramm, das auch als *Flussdiagramm* (engl. *flowchart*) oder *Programmstrukturplan* bezeichnet wird. Es ist eine graphische Darstellung zur Umsetzung eines Algorithmus in einem Programm und beschreibt die Folge von Operationen zur Lösung einer Aufgabe.

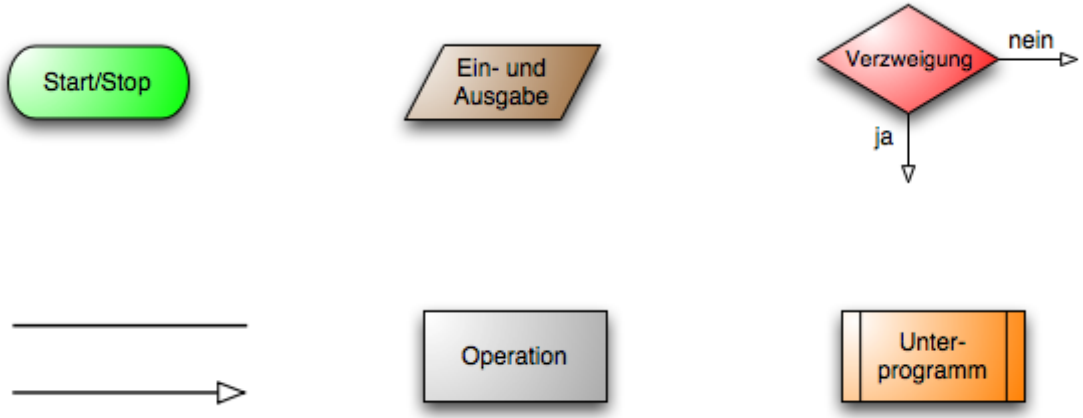
Die Symbole für Programmablaufpläne sind in der **DIN 66001** genormt. Dort werden auch Symbole für Datenflusspläne definiert. Programmablaufpläne werden oft unabhängig von Computerprogrammen auch zur Darstellung von Prozessen und Tätigkeiten eingesetzt (z. B. als Beschreibung des Arbeitsablaufs bei der Angebotserstellung in einem Handelsunternehmen). Im Bereich der Softwareerstellung werden sie nur noch selten verwendet: Programmcode moderner Programmiersprachen bietet ähnlichen Abstraktionsgrad, ist jedoch einfacher zu erstellen und in der Regel sehr viel einfacher zu verändern als ein Ablaufdiagramm.

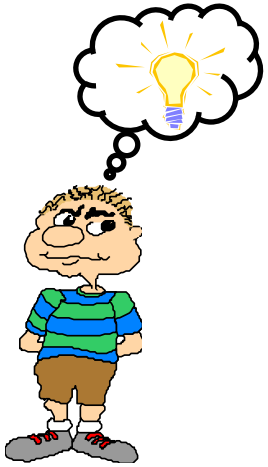
… es stehen die folgenden Symbole zur Verfügung ….



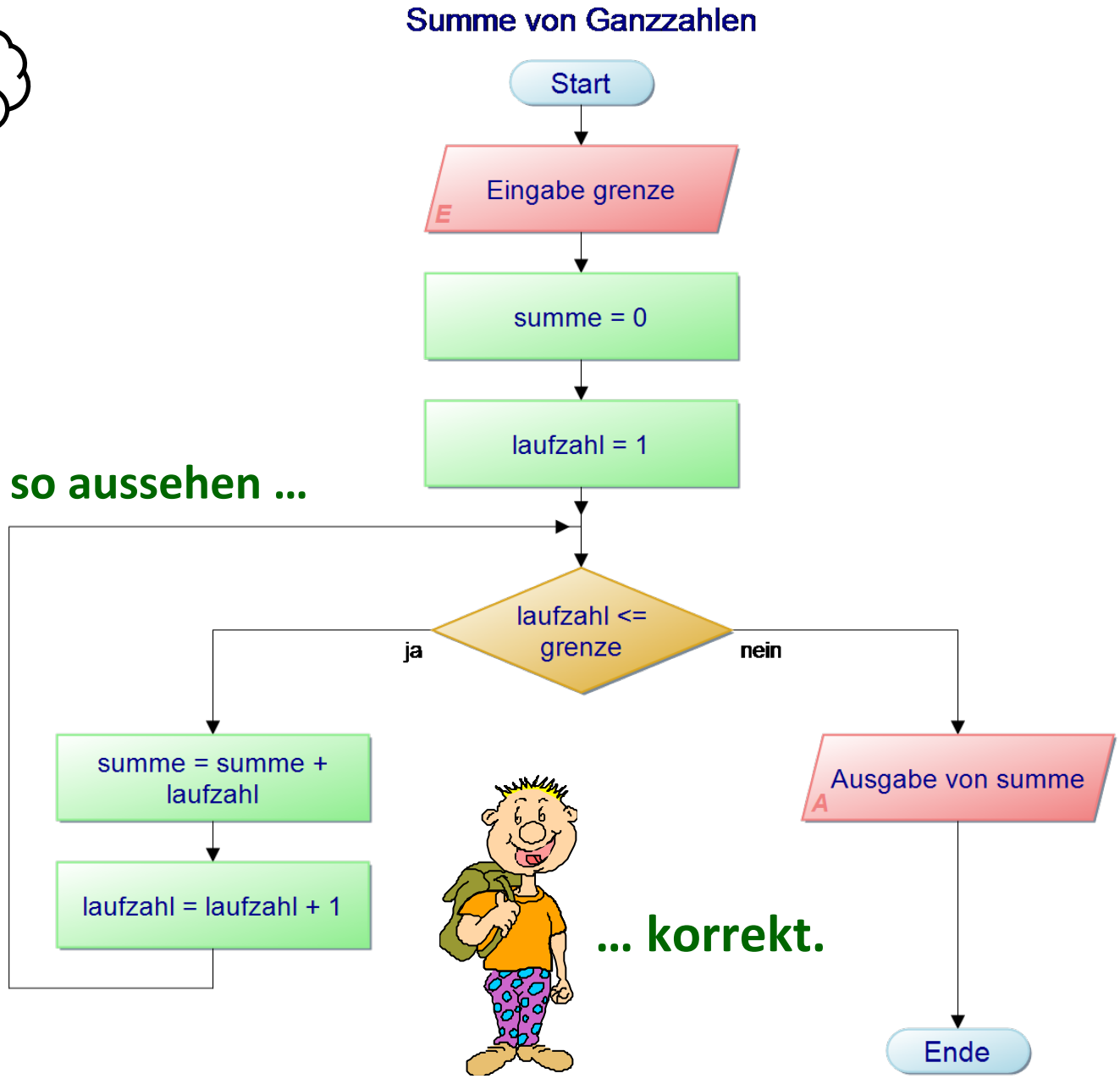
… jetzt musst Du es nur noch richtig zusammenbauen ….

… den ersten Schritt hast Du ja schon erledigt. Jetzt hätte ich gerne im zweiten Schritt den PAP. Du erinnerst Dich?





… müsste so aussehen …



… korrekt.

… noch eine kleine Aufgabe zum Üben. Erstelle mir den PAP ….

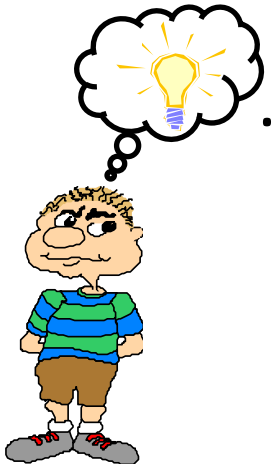
In einem Programm hat ein Benutzer die Möglichkeit 30 Zahlen hintereinander einzugeben.

Das Programm läuft so ab, dass eine eingegebene Zahl nur dann sofort wieder ausgegeben wird, wenn sie positiv ist.

Ist sie negativ wird sie nicht ausgegeben.

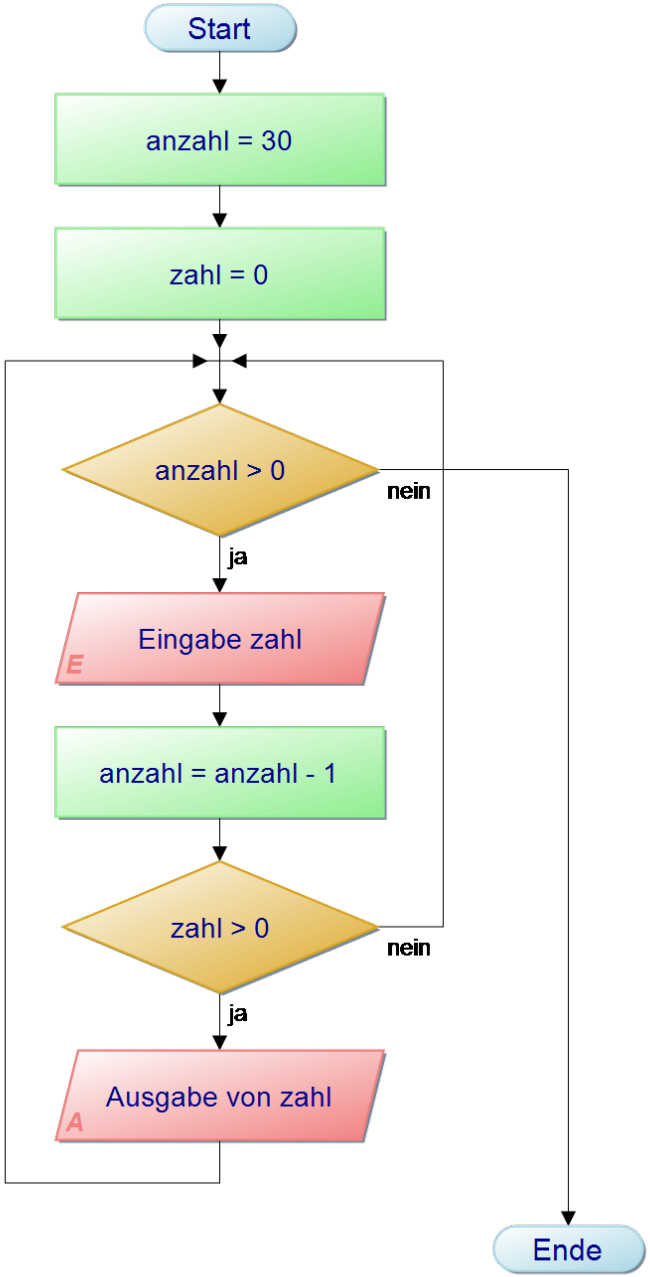
Danach wird die nächste Zahl eingelesen und geprüft.

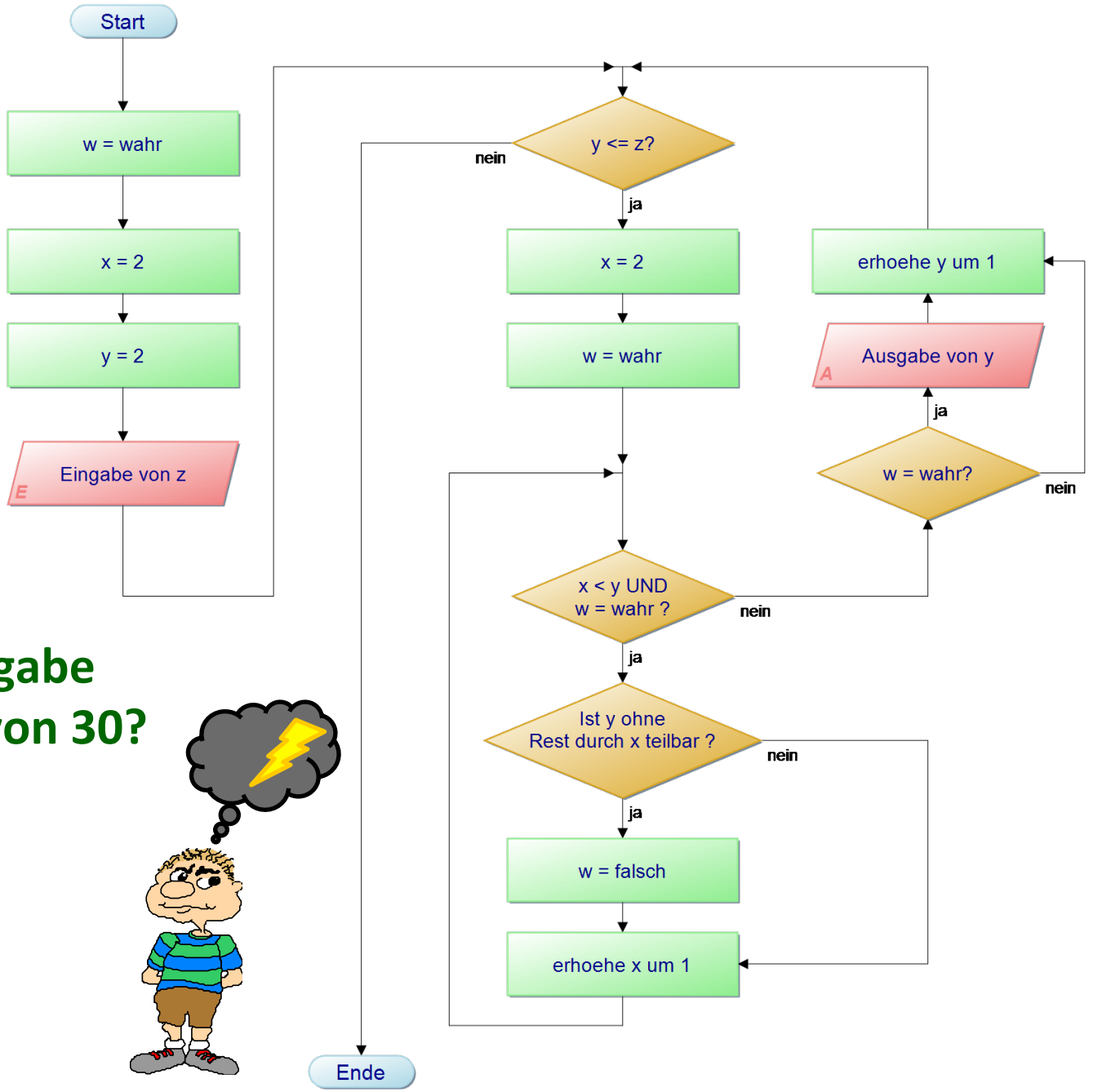
Dieser Vorgang wiederholt sich für 30 Zahlen.



… müsste so aussehen …

… korrekt.
Jetzt mal umgekehrt ….





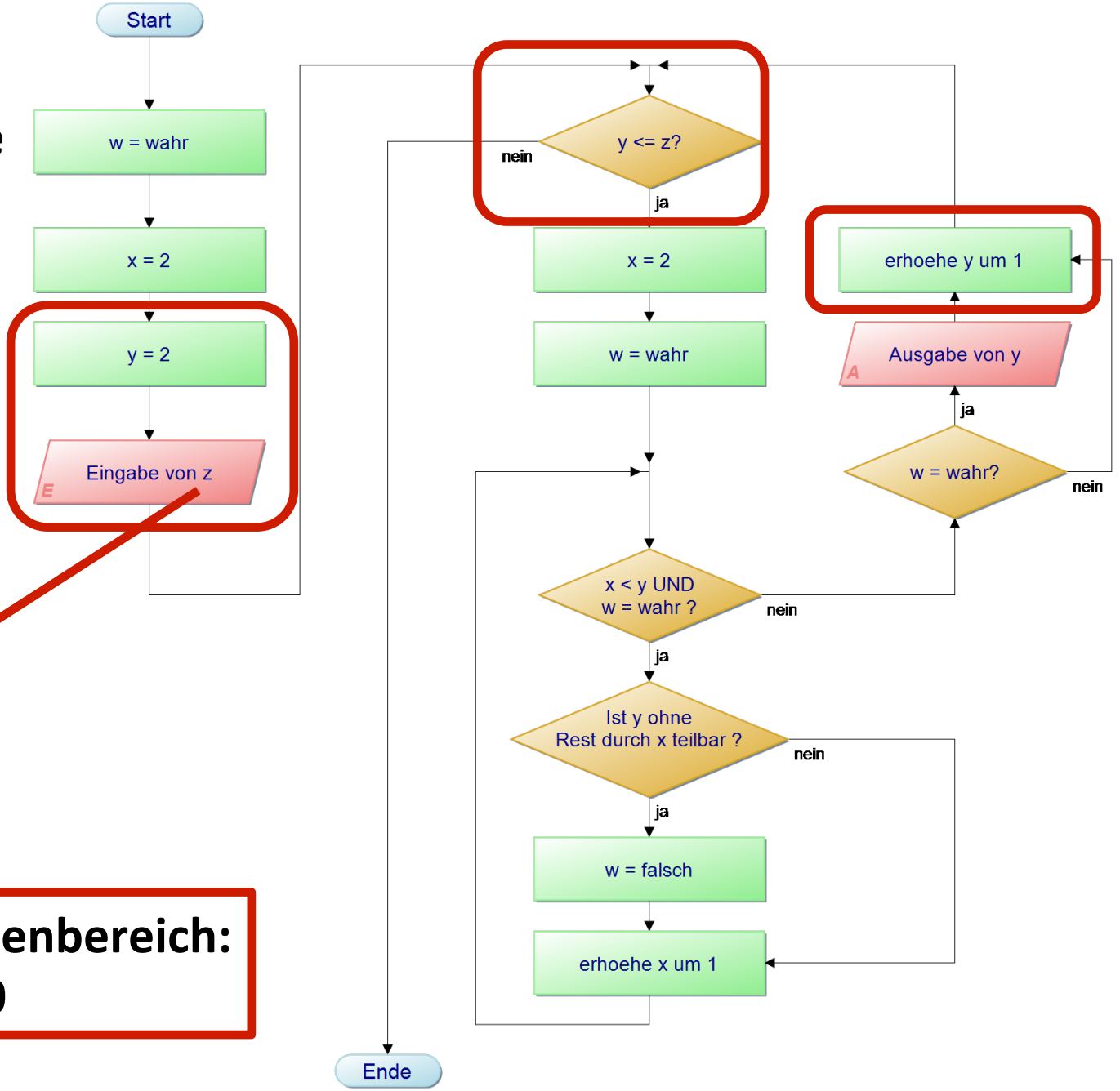
Wie lautet die Ausgabe bei einer Eingabe von 30?

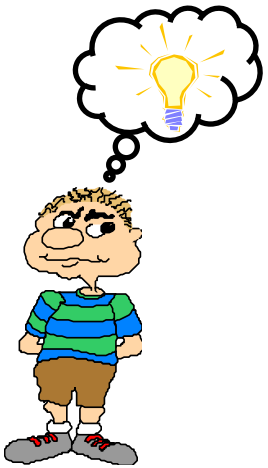
… beteiligte Größen ?

… solange y kleiner gleich z ist tut sich etwas ….

30

Überprüfter Zahlenbereich:
2 … 30





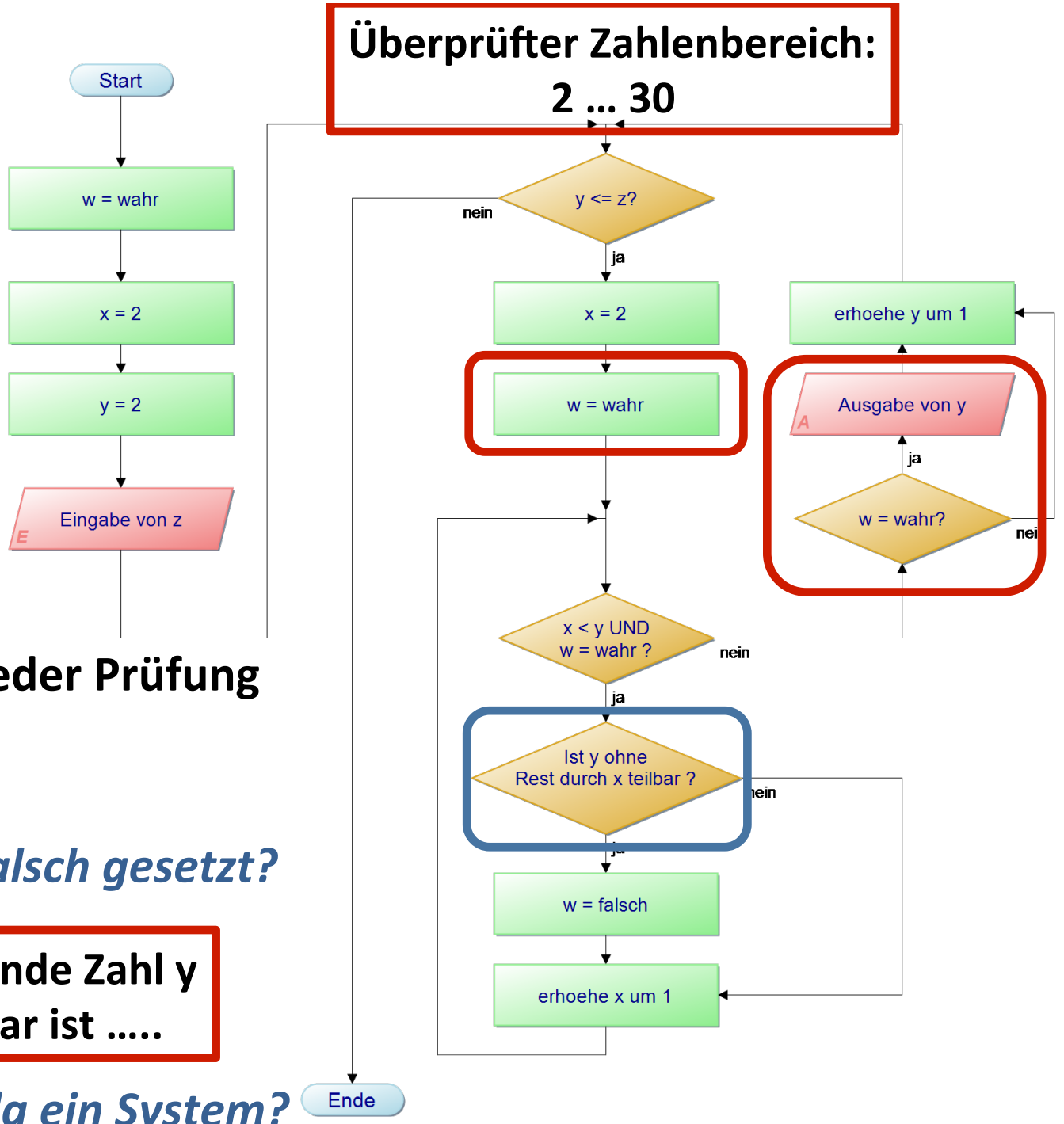
y wird ausgegeben wenn w wahr ist.

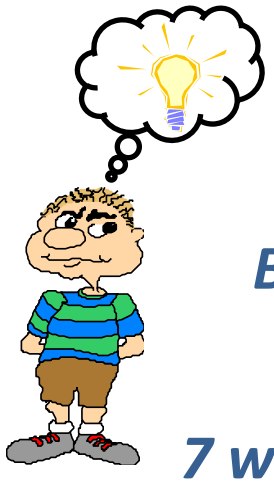
w wird zu Beginn jeder Prüfung auf wahr gesetzt …

wann wird w auf falsch gesetzt?

wenn die zu prüfende Zahl y durch ein x teilbar ist …..

… gibt es da ein System?





Überprüfter Zahlenbereich:
2 … 30

wenn die zu prüfende Zahl y durch ein x teilbar ist …..

wann wird y ausgegeben?

Bsp.: 7,8,9,10,11

10 durch 5 = 2 Rest 0 → w = falsch

7 wird ausgegeben

7 durch 2 = 3 Rest 1 → w = wahr
7 durch 3 = 2 Rest 1 → w = wahr
7 durch 4 = 1 Rest 3 → w = wahr
7 durch 5 = 1 Rest 2 → w = wahr
7 durch 6 = 1 Rest 1 → w = wahr

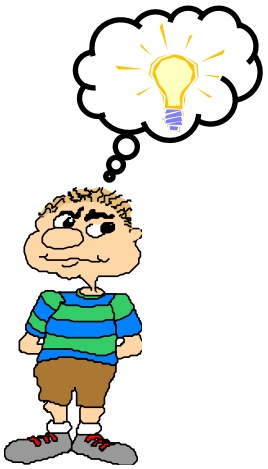
8 durch 2 = 4 Rest 0 → w = falsch

9 durch 2 = 4 Rest 1 → w = wahr
9 durch 3 = 3 Rest 0 → w = falsch

11 wird ausgegeben

11 durch 2 = 5 Rest 1 → w = wahr
11 durch 3 = 3 Rest 2 → w = wahr
11 durch 4 = 2 Rest 3 → w = wahr
11 durch 5 = 2 Rest 1 → w = wahr
11 durch 6 = 1 Rest 5 → w = wahr
11 durch 7 = 1 Rest 4 → w = wahr
11 durch 8 = 1 Rest 3 → w = wahr
11 durch 9 = 1 Rest 2 → w = wahr
11 durch 10 = 1 Rest 1 → w = wahr

YEAH – Ich hab‘s !!!!



… hey, Proggieboy – Ich hab‘s!

Die Ausgabe lautet:

2 3 5 7 11 13 17 19 23 29

Respekt!

Es handelt sich wirklich um Primzahlen!

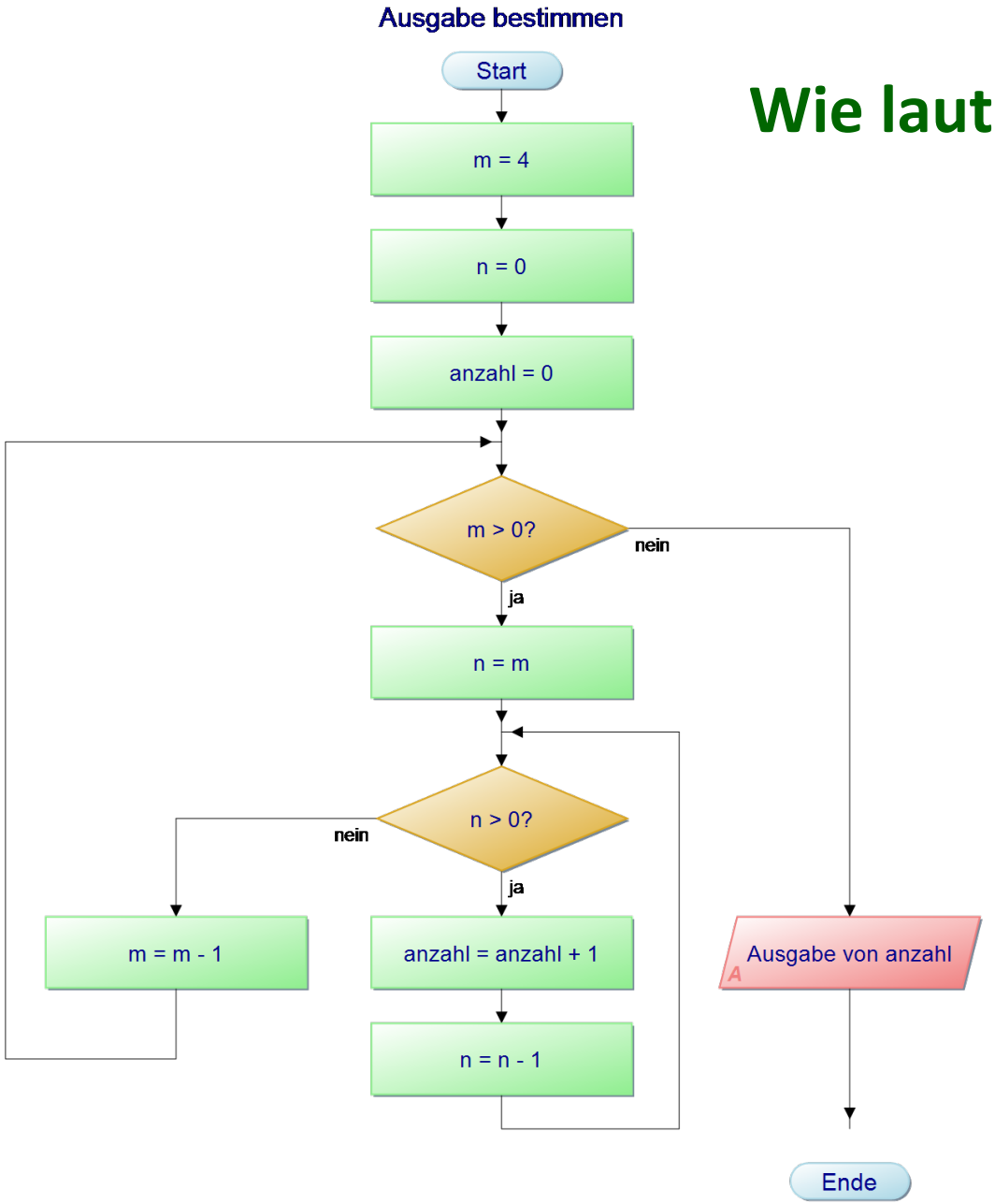
Primzahl

WIKIPEDIA
Die freie Enzyklopädie

Eine **Primzahl** ist eine natürliche Zahl, die größer als eins und nur durch sich selbst und durch eins teilbar ist. Eine Primzahl ist also eine natürliche Zahl mit genau zwei natürlichen Zahlen als Teiler. Die kleinsten Primzahlen sind

> 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97 … (Folge A000040 in OEIS)

Eine natürliche Zahl größer als 1 heißt *prim*, wenn sie eine Primzahl ist, andernfalls heißt sie *zusammengesetzt*. Die Zahlen 0 und 1 sind weder prim noch zusammengesetzt. Warum die Zahl 1 nicht als Primzahl angesehen wird, wird im Abschnitt "Warum ist die Zahl 1 keine Primzahl?" erklärt.



Wie lautet die Ausgabe?

Die Ausgabe lautet:

10

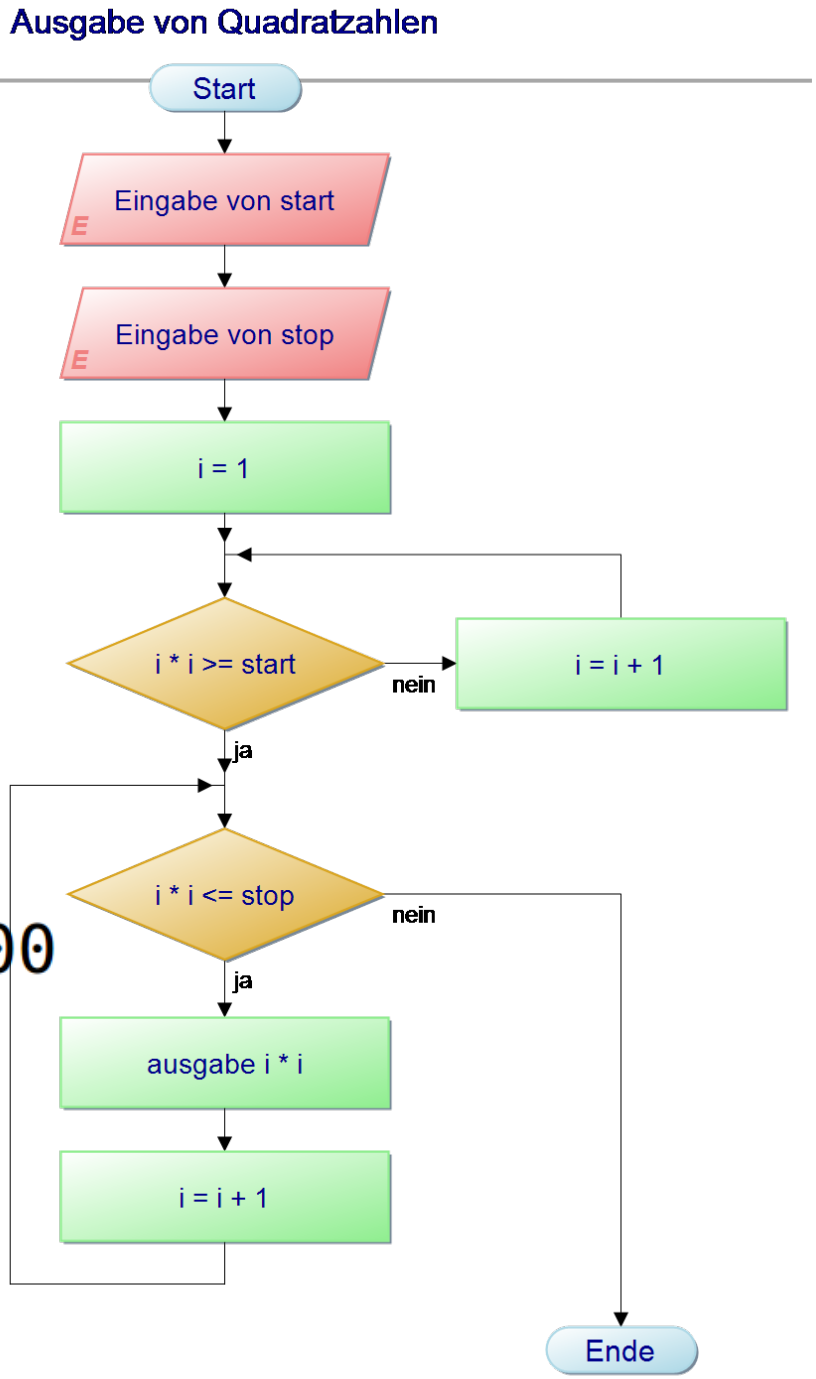
Kleine Übung:

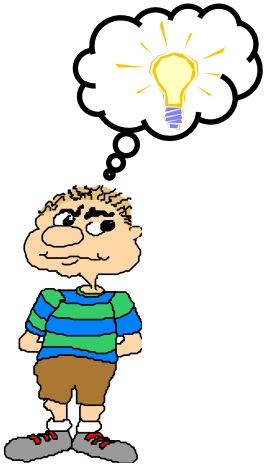
Erstellen Sie einen Programmablaufplan, der alle Quadratzahlen in einem eingegebenen Intervall einschließlich der Intervallgrenzen ausgibt.

```
Startwert: 1
Stopwert : 100
1 4 9 16 25 36 49 64 81 100
```

Lösungsvorschlag:

```
Startwert: 1
Stopwert : 100
1 4 9 16 25 36 49 64 81 100
```





… kann es sein, dass ein PAP mit mehreren Auswahlmöglichkeiten leicht unübersichtlich wird?

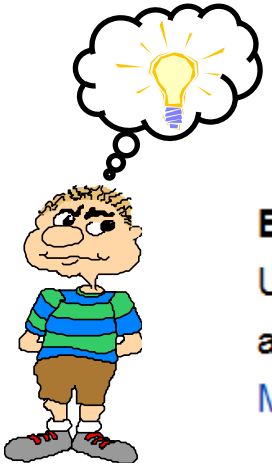
… ja, das haben sich zwei amerikanische Informatiker bereits 1972/73 ebenfalls gedacht.

```
laufzahl ← 1
summe ← 0
grenze ← 1000

wiederhole solange summe kleiner 1000
    summe ← summe + laufzahl
    laufzahl ← laufzahl + 1

laufzahl ← laufzahl - 2
summe ← summe - (laufzahl+1)
ausgabe → summe und laufzahl
```

DIN	DIN 66261
Bereich	Programmierung
Titel	Informationsverarbeitung; Sinnbilder für Struktogramme nach Nassi-Shneiderman
Kurzbeschreibung:	Struktogramme
Letzte Ausgabe	11.1985
ISO	



… welche Zwei waren das?

Ben Shneiderman (* 21. August 1947 in New York) ist ein US-amerikanischer Informatiker, der zurzeit als Professor für Informatik am *Human-Computer Interaction Laboratory* an der University of Maryland, College Park lehrt.

Isaac Nassi

Nassi war u.a. Direktor der Entwicklungsabteilung bei Cisco Systems und arbeitete sieben Jahre bei Apple Computer, zuletzt 1994-1996 als Chef der Betriebssystemabteilung. Gegenwärtig ist er Leiter der Forschungsabteilung von SAP America. Er war Mitbegründer der Encore Computer Corporation.

Nassi-Shneiderman-Diagramm

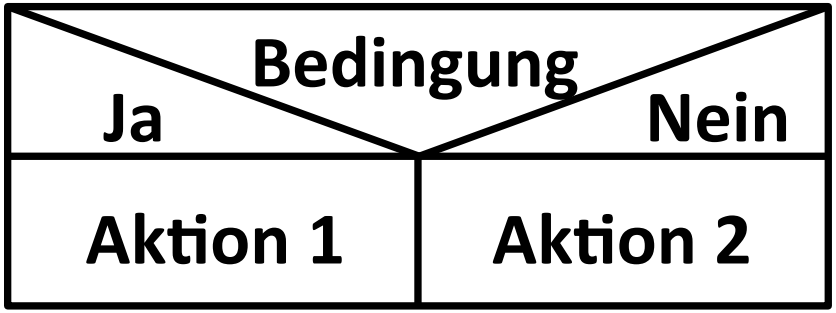
Diagrammtyp zur Darstellung von Programmentwürfen im Rahmen der Methode der strukturierten Programmierung

Im Allgemeinen übersichtlicher als PAPs.

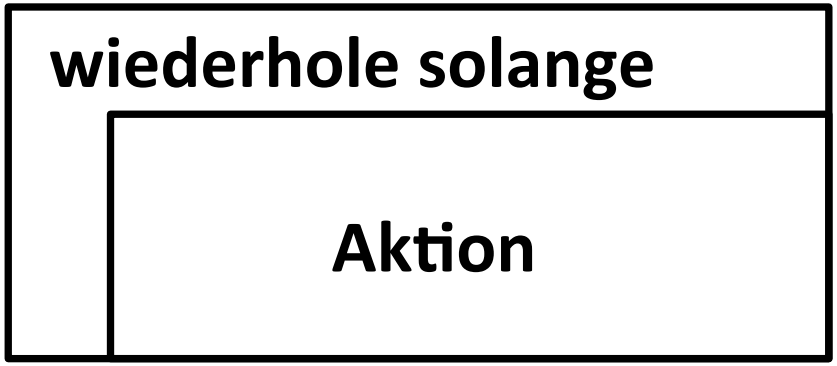
… wie PAPs bestehen Struktogramme aus mehreren Elementen …

Anweisung/Operation

Entscheidungen



Schleifen/Iterationen



… keine Kennzeichnung für Start und Stop, da es immer von oben nach unten gelesen wird.

… nur 3 Elemente, und das reicht??

… es gibt noch mehr Elemente, die ich Dir separat erkläre. Aber mit diesen drei kommst Du schon weit …. oder auch nicht …

… erinnerst Du dich noch an Deine Summenaufgabe?

… erstelle dazu das Struktogramm …

```
Startwert: 1
Stopwert: 10
1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10 = 55
```

… ok, ich probiers ….

… müsste so aussehen …

Struktogramm

E	n ← Benutzereingabe	Eingabebereich
	erg ← 0	
	i ← 1	
V	wiederhole solange i <= n	Verarbeitungs-bereich
	erg ← erg + i	
	i ← i + 1	
A	ausgabe erg	Ausgabebereich

… korrekt, daher kommt auch der Name EVA - Prinzip

… eine kleine Gegenüberstellung

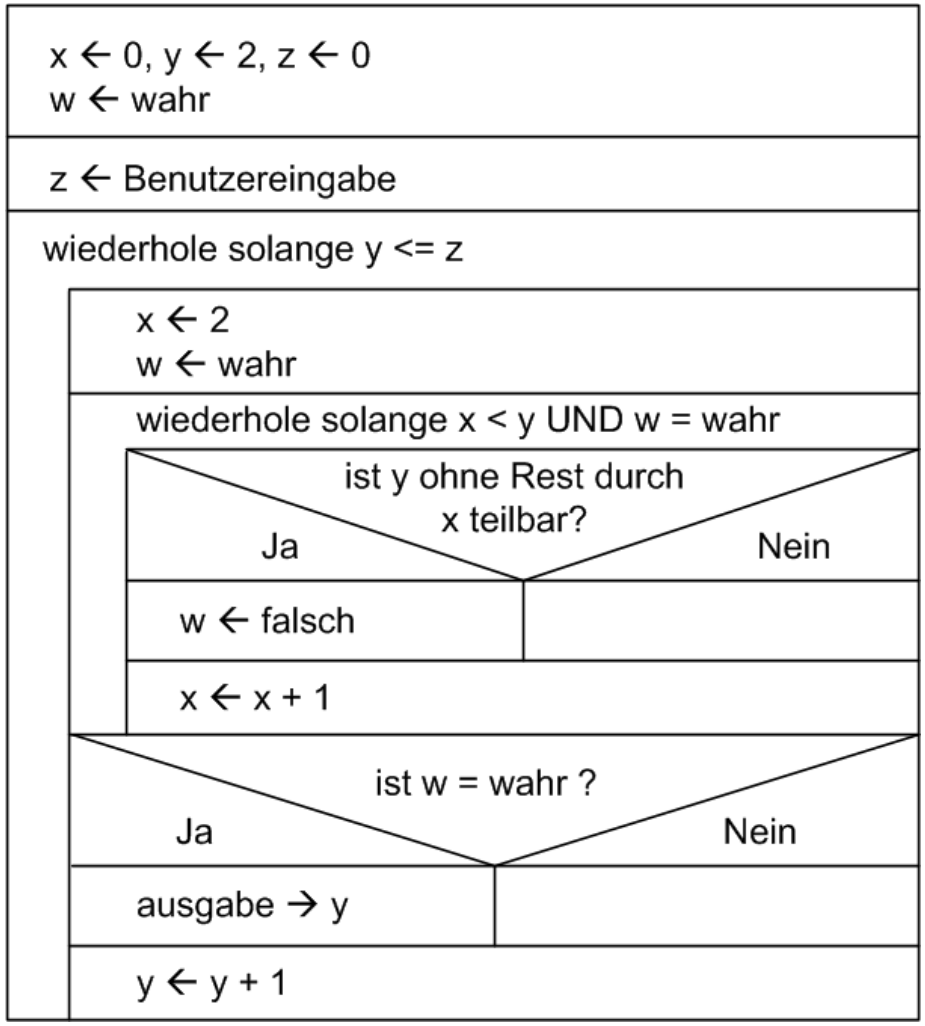
Struktogramm ⟷ **Programmablaufplan**

n ← Benutzereingabe
erg ← 0
i ← 1
wiederhole solange i <= n
erg ← erg + i
i ← i + 1
ausgabe erg

… irgendwie kompakter …

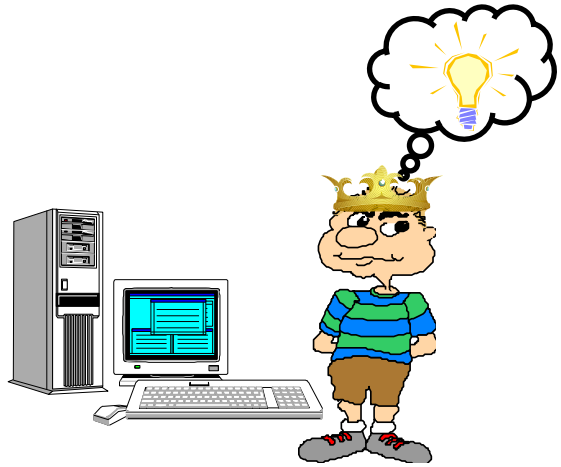
… eine noch zum Abschluss …

… erstelle mir zu der Primzahlausgabe das Struktogramm …



2 3 5 7 11 …

… bitte sehr der Herr …



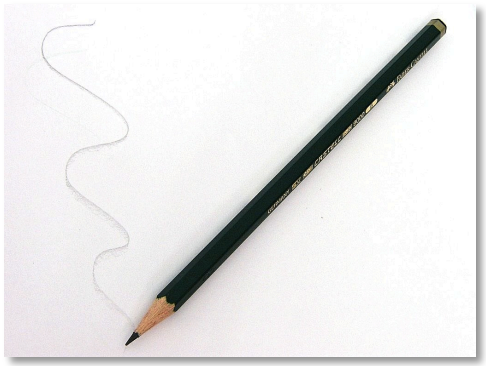
wann darf ich tippen?

sehr gut

in Schritt 3

Wer sind die besten Freunde eines Programmierers?

Papier + Bleistift



do it ...