Grundlagen Programmierung (Herr Scheringer und Frau Lynen)

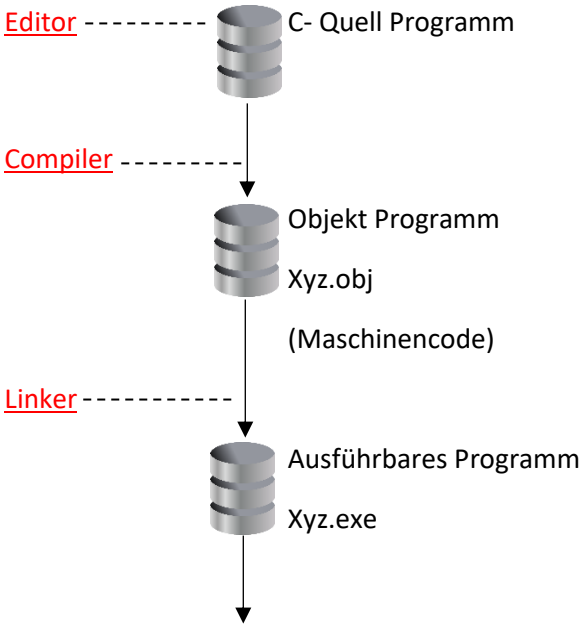
Inhaltsverzeichnis

Programmierung Komponenten

C- Programmierung

Besteht aus:

- Editor
- Compiler (Maschinen- Bytecode
- Linker und Interpreter
- Debugger



Bibliothek mit

Vorgefertigten

Programmteilen

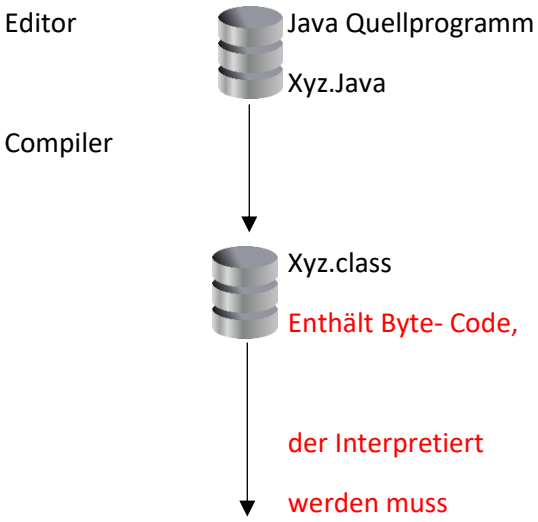
(Maschinencode)

Loader Lädt ausführbares

Programm zur Ausführung

In den Hauptspeicher.

Java- Programmierung



Java Virtual Maschine

- Java Xyz

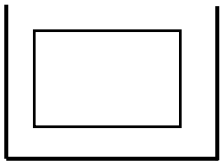
Java ist auch Interpreter

Java ist ein sogenannter Container,

der Xyz.class ausführt

Stellt auch hierzu erforderliche

Programmteile Bereit.



Fehler Erkennung

Zu unterscheiden ist zwischen

- Compile- Time
    - Fehler die der Compiler findet z.B.:
        - Syntax- Fehler,
        - Semantik- Fehler
- Runtime
    - Semantik- Fehler
    - Logik- Fehler

Software Engineering

z.B. bei der Objektorientierten Softwareentwicklung unterscheidet man die Phasen

OOA ⟶ Objektorientierte Analyse

OOD ⟶ Objektorientiertes Design

OOB ⟶ Objektorientierte Programmierung

Überblick über wichtige Konzepte

 Phasenorientiertes Entwerfen

 Trennung von Fachkonzept und DV-Konzept

 Methode der schrittweisen Verfeinerung

 Top-Down-Design bzw. -Implementierung

 Bottom- Up- Design bzw. -Implementierung

 Prototypansatz

Generationen der Programmiersprachen

1. Maschinensprachen
2. Assembler
3. FORTRAN, Cobol, Pascal, PL/1, ……., C
4. SQL (nicht mehr wie, sondern was)
5. C++, C#, Java, …….

UNIX wurde in „C“ geschrieben

Hilfen bei der Fehlersuche in der Programmierung, sind außerdem:

- Struktogramme
- Programmablaufplan
- Schreibtischtest

Übungen 1

1. Welches Werkzeug dient u. a. dazu, den Quelltext eines Programms zu er fassen und zu speichern?
    a. Editor
    b. Integrierte Entwicklungs- Umgebung
2. Welcher Datenbestand dient als Input in den Compilier Prozess und welcher Datenbestand ist das Ergebnis (Output) nach dem Übersetzungslauf?
    a. Maschinencode
    b. Bytecode
3. Mit welchem Werkzeug kann nach dem Compilieren ein ablauffähiger Maschinencode erzeugt werden?
    a. Linker
4. Warum ist der vom Compiler generierte Maschinencode nicht ablauffähig?
    a. Die Laufzeitumgebung fehlt
5. Mit welchem Werkzeug ist es u. a. möglich, den Inhalt von Programmvariablen zur Laufzeit anzusehen und ggf. sogar zu ändern?
    a. Debugger
6. Ein in die Entwicklungsumgebung integrierter Editor bietet meist Unterstützung bei der Editierung des Quelltextes. Nennen Sie dafür zwei Beispiele.
    a. Syntax farblich anzeigen
    b. Schlüsselworte werden farblich hervorgehoben
    c. Fehler werden evtl. angezeigt
7. Nennen Sie zwei wichtige Konzepte des modernen Softwareengineerings.
    a. Top down
    b. Phasenorientiert
    c. Prototyp

1. Wie ist das Übersetzungsverhältnis bei den Programmiersprachen, die zur 2. Generation zählen? (Wählen Sie unter den folgenden Möglichkeiten eine Antwort aus.)
    a. 1:1
2. Ab welcher Progammiersprachengeneration spricht man von problemorientierter Programmentwicklung?
    a. Ab der dritten Generation
3. Ab welcher Sprachengeneration bestehen die Programmiersprachen nicht mehr ausschließlich aus prozeduralen Sprachelementen?
    a. Ab der vierten Generation
4. Nennen Sie zwei Beispiele für Ereignisse, die in ereignisorientierten sprachen erkannt und bearbeitet werden können.
    a. Mouse Over
5. Was unterscheidet prozedurale Sprachen von objektorientierten Sprachen? Beschreiben Sie einen wesentlichen Unterschied.
    a. Die Objektorientierte Sprachen kann an andere Objektorientierte Sprachen, befehle senden. Die Prozedurale Sprache kann das nicht.

Aufbau eines Programms

- Datenteil
    - Jede Programmiersprache „kennt eingebaute“ Datentypen
    - Variablen, um Daten speichern zu können
    - Konstante
- Befehlsteil
    - Zuweisungen
    - (Rechen) Operationen
    - Ein- Ausgabe befehle

Datenteil

- Jede Programmiersprache „kennt eingebaute“ Datentypen (Primitive Datentypen)
- Beispiel JAVA
    - byte a = 5, Java Speicherplatz: 1 Byte
    - short b = 17, Java Speicherplatz: 2 Byte
    - int c = 184, Java Speicherplatz: 4 Byte
    - long d = 4812 Java Speicherplatz: 8 Byte

    Datentypen für ganze Zahlen

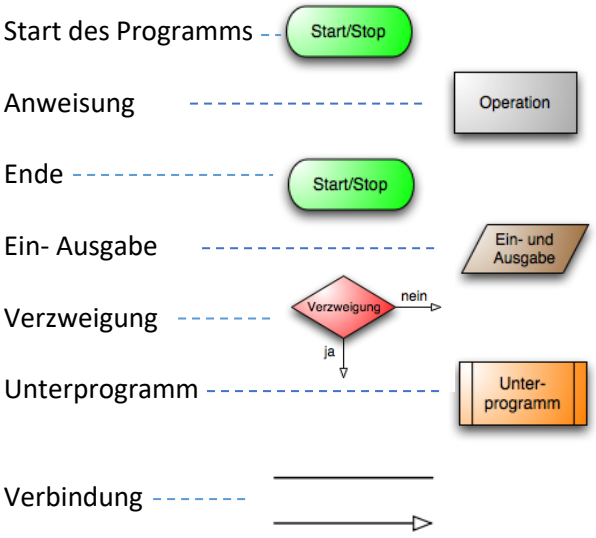
    - float x = 14.17f Java Speicherplatz: 4 Byte
    - double y = -7.89 Java Speicherplatz: 8 Byte

    Datentypen für gleitkommazahlen

    - Char m = ‚A‘; Java Speicherplatz: 2 Byte
    - Boolean p = True/False Java Speicherplatz: 1 Byte

- String ist ein mix aus den einzelnen Datentypen

PAP Symbole

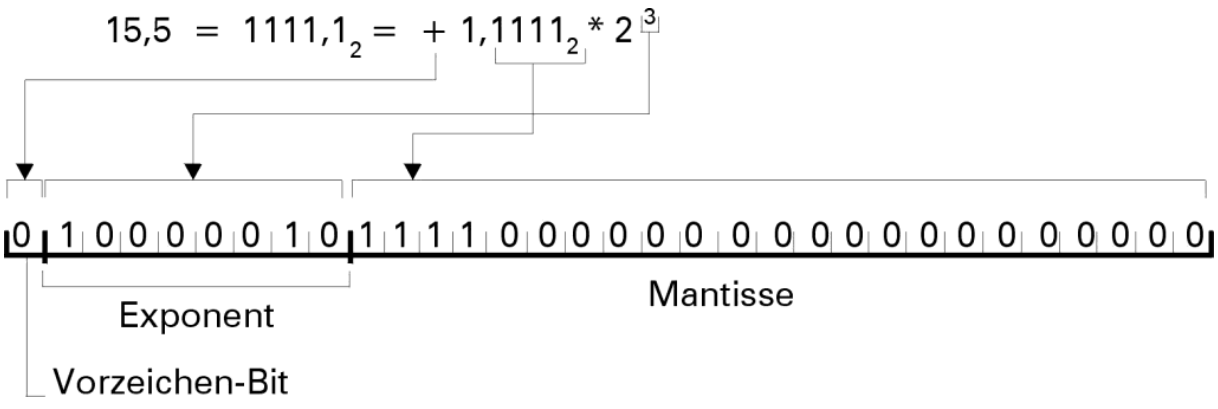


Beispiel Programmierung:

```
package fi_19_2;

public class FI_19_2 {
{

        /**
        *
        * @param args
        */
        public static void main(String[] args)
                {
                //Summe der Zahlen von 1 bis 100 berechnen
                //Nicht berücksichtigen Zahlen zwischen 10 und 20

                 int  summe = 0;

                 int i =1;
                 while (i <= 100)
                {

                 if  (i < 10 || i > 20)
                 {
                 summe = summe + i;
                 }
        System.out.println("summe = "+summe+"i= "+i);
        i = i + 1;
    }
  }

}
```

Fehlerbeschreibungen

- logik- Fehler
- Semantik- Fehler
- Syntax- Fehler



Zuweisungen
Java
byte -> short -> int -> long -> float -> double
char
Zuweisung in diese Richtung -> geht immer

```
int a = 5;
float x;
x = a;
```

umgekehrt gehen Zuweisungen nicht ohne Nachhilfe

```
float b = 2.79;
int c;
c = (int) b;
```

casting

Beispiel Quersummen

3691	%10	→	1
3691	/10	→	369
369	%10	→	9
369	/10	→	36
36	%10	→	6
36	/10	→	3
3	%10	→	3
3	/10	→	0

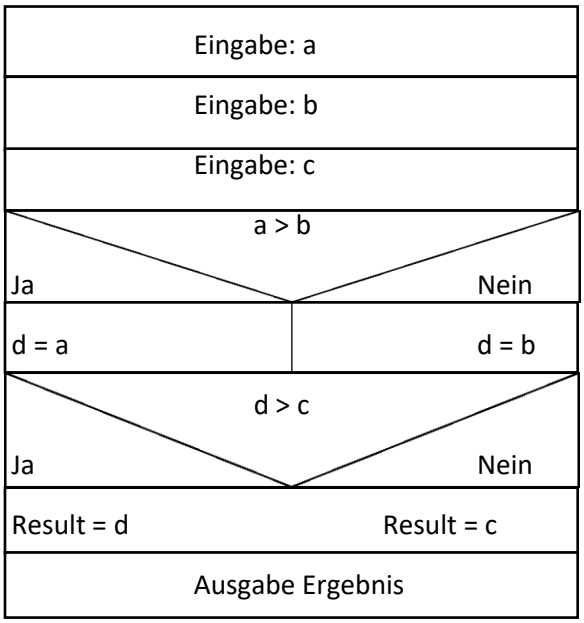
Operator

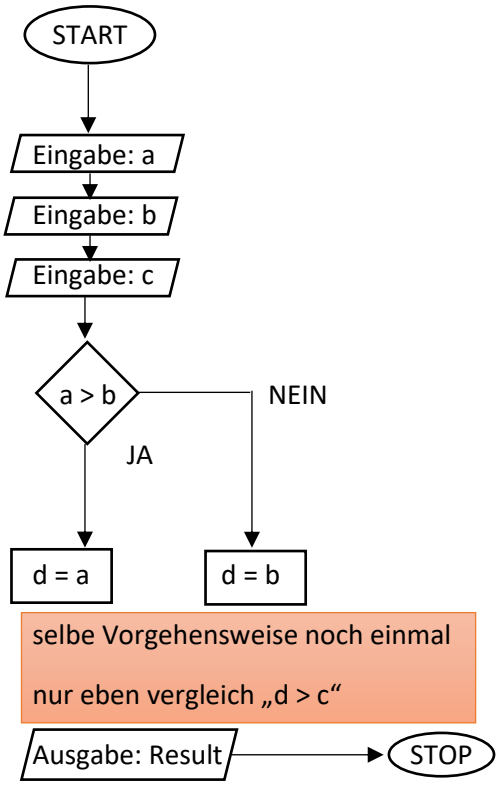
	()		<	kleiner
	+		<=	kleiner gleich
Arithmetik	/	Vergleichs	>	größer
OP	%	OP	>=	größer gleich
	+		==	gleich
	-		!=	in gleich

	&&	UND
Log.	||	ODER
	|	NICHT

Struktogramm und PAP Modell:

Beispiel:





Bedingung: Bei der „ODER“ Bedingung muss mindestens eine wahr sein damit der gesamte vergleich, wahr ist. Bei einer „UND“ Bedingung müssen alle wahr sein!

Vorgehensweise Struktogramm/PAP erstellen

Schritt 1: Datenmodell (= Sammlung von Variablen) erstellen

- z.B. Zähler, Summe, Grenze Zähler, Grenze Summe, Eingabe a, Eingabe b, …….

Schritt 2: Überlegungen zur Logik des Programms

1 Schleife zur Eingabe der Zahlen
2 Ausgabe Summe
3 Schleifen Durchlaufbedingung

Schritt 3: Logik im Detail entwickeln

Schreibtischtest

A > B and C > D or A > X and C > D
Bed1 Bed2 Bed3 Bed4

A	B	C	D	X	Bed1	Bed2	Bed3	Bed4	Bed1 and Bed2	Bed3 and Bed4
3	7	9	4	2	F	T	T	T	F	T
10	4	3	7	5	T	F	T	F	F	F

Übungen 2

1.1.Welche für den Übersetzungsvorgang relevanten Festlegungen trifft der Programmierer durch die Definition einer Variablen?

Der gewählte Typ gibt den Speicherplatz vor

1.2.Welche numerischen Datentypen können unmittelbar zu arithmetischen Operationen (ohne vorherige Konvertierung) herangezogen werden?

Byte, Short, Int, Long, Float, Double

1.3.Neben den elementaren Datentypen kennen die Programmiersprachen auch zwei Datenaggregate. Welche sind das? Was unterscheidet sie voneinander?

Arrays und Strukturen

2.1.Welche Bedeutung haben die Operatoren ** und % ?

Potenzierung und Modulodivision

2.2.Welche Arten von Operanden kommen in arithmetischen Ausdrücken vor? Nennen Sie drei mögliche Operanden.

Variable (eine sich eventuell verändernder Wert), **Konstante** (ein vom Programmierer festgelegter Wert), *Literale* (ein zahlen Wert) z.B. „**final** int X = *12*

2.3.Warum sollten Sie nach Möglichkeit vermeiden, in einem arithmetischen Ausdruck Operanden unterschiedlichen Datentyps zu verwenden?

Ungewollte Typ Konvertierung (besser kein Mixen von Datentypen)

2.4. Setzen Sie diese Formel in einen Pseudocode um. Verwenden Sie dabei die in dieser Unterrichtseinheit eingeführten Operatoren. Verwenden Sie die Variablen des mathematischen Terms auch als Variablennamen für Ihre Anweisung.

$$X = \frac{I(A + B) - 3X}{(C - D)^2 - 3(A + X)}$$

X=(i*(a+b)-3+x) /((c-d**2-3*(a+x))

---

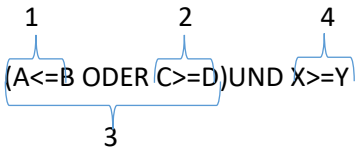
1. Aufgabe

Geben Sie bei den folgenden komplexen Bedingungen an, ob sie insgesamt wahr oder falsch sind. Zur Überprüfung dienen die folgenden ganzzahlig definierten Variablen mit den folgenden Inhalten:
A=11, B=9, C=3, D=9

1. A < B ODER C < D Ergebnis: falsch
2. A < B ODER C > D Ergebnis: wahr
3. A > B ODER NICHT (C > B) Ergebnis: wahr
4. NICHT(A >= B) ODER C = D Ergebnis: falsch
5. A < B UND D < B Ergebnis: falsch
6. A > B UND C > D Ergebnis: falsch
7. A > B UND C <= D Ergebnis: wahr
8. A < B ODER C <= D UND B > 8 Ergebnis: wahr
9. A > B ODER C > D UND B < 8 Ergebnis: wahr
10. (A > B ODER C > D) UND B > 8 Ergebnis: wahr
11. NICHT(A < B UND C < D) Ergebnis: wahr

2. Aufgabe

Gegeben ist die folgende komplexe Bedingung:
NICHT(A > B UND C < D) UND X >= Y
Lösen Sie zuerst das NICHT auf, so dass eine komplexe Bedingung ohne
NICHT-Operator entsteht.
Zeigen Sie anschließend auf, wie diese komplexe Bedingung bei der Bewertung durch den Compiler abgearbeitet wird. Dabei geht es u. a. um die Teilausdrücke, die nicht mehr bewertet werden, weil das Ergebnis bereits feststeht.



3. Aufgabe

Gegeben ist die folgende komplexe Bedingung.
NICHT(A > Z ODER B < C) ODER NICHT (D > A UND X < Z ODER X = B)

UntersuchenSie, ob diese Bedingung insgesamt bei folgen den Variableninhalten wahr oder falsch ist:
A B C D X Z
5 4 6 3 8 2
Begründen Sie bei Ihrer Antwort bitte die Entscheidung „wahr“ oder „falsch“.

Wahr

4. Aufgabe

Schreiben Sie die folgenden drei logischen Ausdrücke neu, ohne dabei den NICHT-Operator zu verwenden.

1. NICHT(A = B ODER C > D)	Lösung:	A<>B UND C<D
2. NICHT(A > C UND NICHT(C = D))		A<=C ODER (C=D)
3. NICHT(A = B ODER C = D UND E = F)		A<>B UND C!=D ODER E!=F

5. Aufgabe

Gegeben ist die folgende komplexe Bedingung (diesmal sehr allgemein formuliert).

NICHT(Bed1 UND Bed2 ODER Bed3)
Geben Sie für die folgenden Fälle an, ob die Bedingung insgesamt wahr oder falsch ist.

Bed1 ist erfüllt		
Bed2 ist nicht erfüllt		
Bed3 ist erfüllt	Aussage:	falsch
Bed1 ist nicht erfüllt		
Bed2 ist nicht erfüllt		
Bed3 ist erfüllt	Aussage:	falsch
Bed1 ist erfüllt		
Bed2 ist erfüllt		
Bed3 ist nicht erfüllt	Aussage:	falsch
Bed1 ist nicht erfüllt		
Bed2 ist erfüllt		
Bed3 ist nicht erfüllt	Aussage:	wahr

Übung 3

1.

Welche der folgenden Aussagen über Struktogramme treffen zu?

- Sie sind besonders dazu geeignet, den Programmablauf übersichtlich abzubilden.
- Sie eignen sich besonders für die Darstellung von Programmabläufen, die nach der prozeduralen Programmiermethode entworfen wurden.

2.

Struktogramme enthalten im Regelfall keinen echten Code irgendeiner konkreten Programmiersprache. Sie enthalten aber auch keine beliebigen Texte.

1- Wie nennt man die Eintragungen in den Struktogrammen?
   a. Pseudocode
2- Aus welchen Bestandteilen setzen sich derartige Eintragungen zusammen?
   a. Operanten, Operatoren und Variablen
3- Warum wird kein Quelltext einer konkreten Programmiersprache in Struktogramme eingetragen?
   a. Struktogramme sollen Programmiersprachen unabhängig, zu Darstellung der Logik darstellen.

3.

Welchen Vorteil bieten Struktogramme gegenüber Programmablaufplänen?

a- Sie eigenen sich besonders gut, da sie sehr übersichtlich sind.

4.

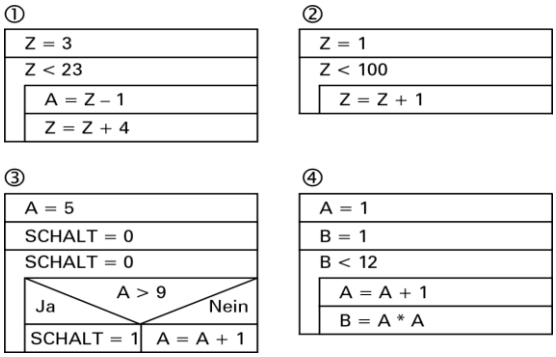
In Struktogrammen können verschiedene Schleifentypen verwendet werden. Dabei werden Laufbedingungen und Abbruchbedingungen zur Schleifensteuerung eingesetzt. Erklären Sie den Unterschied zwischen einer Laufbedingung und einer Abbruchbedingung.

a- Eine Laufbedingung wird so lange durch laufen wie die Bedingung erfüllt ist.
b- Eine Abbruchbedingung wir so lange durch laufen bis Abbruchbedingung erfüllt ist.

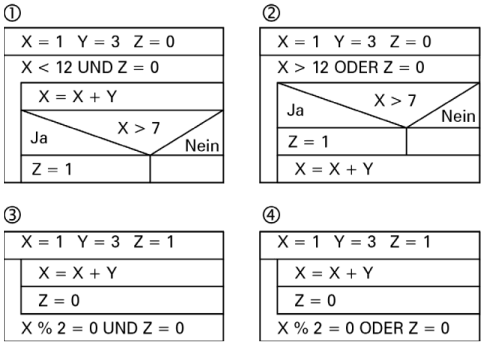
---

1. Nachfolgend finden Sie Struktogramme zu vier Schleifenkonstruktionen. Geben Sie für jedes Struktogramm an,
- wie oft die Schleife durchlaufen wird;
- welchen Inhalt die Variablen nach Ablauf der Schleife haben.

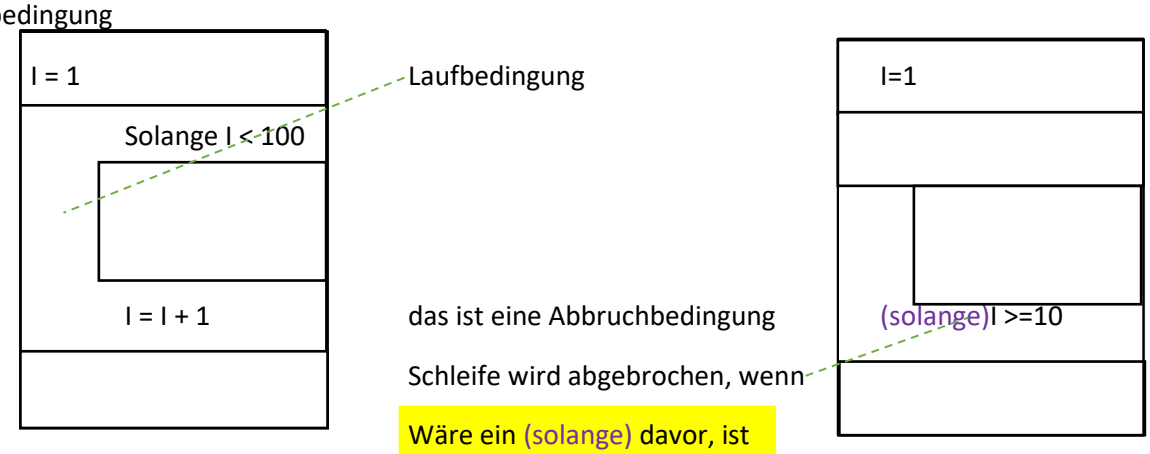
1- Schleife durchlaufen = 5, A = 18, Z = 23
2- Schleife durchlaufen = 99, Z = 100
3- Schleife durchlaufen = 6, Schalt = 1, Z = 10
4- Schleife durchlaufen = 3, A = 4, B = 16



1- Schl = 4, X = 13, Y = 3, Z = 1
2- Schl = entl, X =entl, Y =4, Z = 1
3- Schl = 1, X = 0, Y = 3, Z = 0
4- Schl = 1, X = 0, Y = 3, Z = 0



Laufbedingung



das ist eine Abbruchbedingung

Schleife wird abgebrochen, wenn

Wäre ein (solange) davor, ist

es eine Bedingung vom Typ „Laufbedingung“.

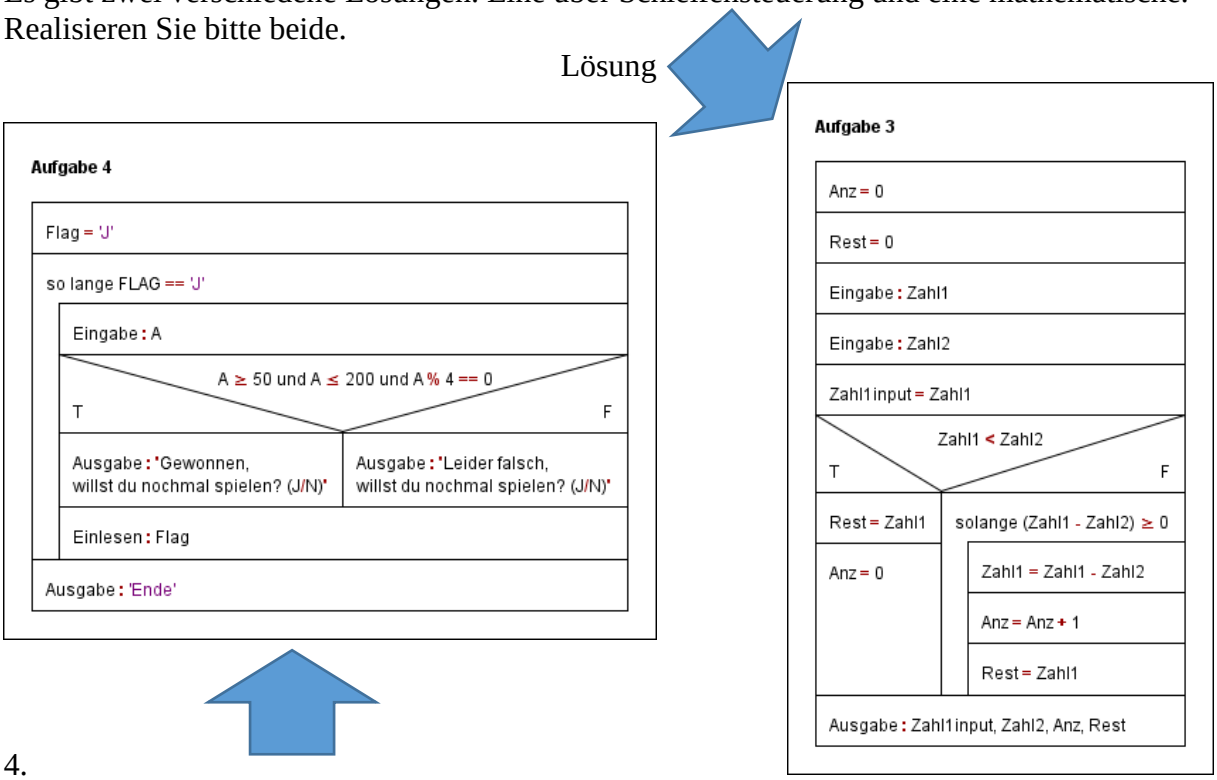
3.

Erstellen Sie ein Struktogramm zu folgendem Programmablauf:
Der Anwender soll zwei Zahlen eingeben. Von der ersten soll die zweite Zahl subtrahiert werden. Vom Ergebnis wird wieder die zweite subtrahiert usw., so lange kein negatives Ergebnis erzielt wird. Das Programm gibt anschließend aus:
Die beiden eingegebenen Zahlen, die Anzahl der Subtraktionen und den verbleibenden Rest.

Hinweis

Es gibt zwei verschiedene Lösungen. Eine über Schleifensteuerung und eine mathematische. Realisieren Sie bitte beide.

Lösung



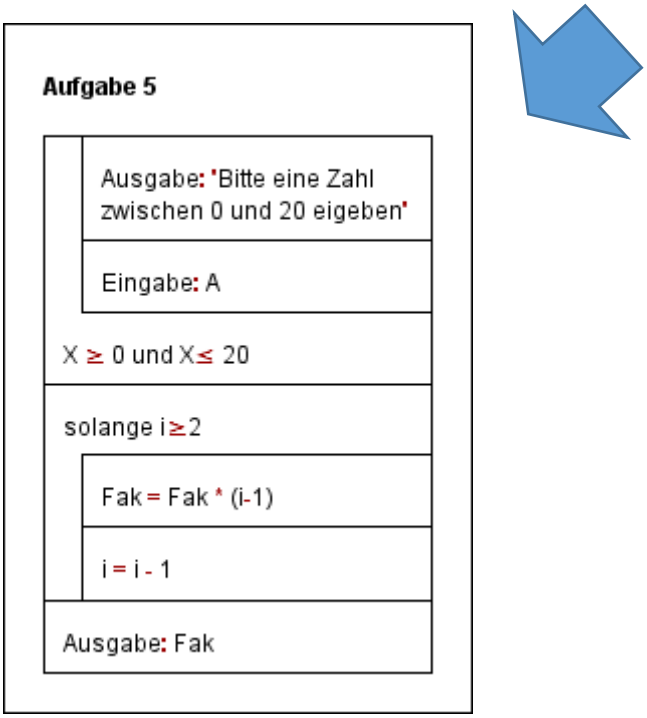
4.

Erstellen Sie ein Struktogramm zu folgendem Programmablauf:
Der Anwender gibt eine Zahl ein. Wenn die Zahl zwischen 50 und 200 li
egt
und auch ohne Rest durch 4 teilbar ist, dann erhält der Anwender eine pos i-
tive Bestätigung, andernfalls einen Fehlerhinweis. Konstruieren Sie auch e i-
ne Schleife, damit der Anwender dieses „Ratespiel“ so oft, wie er möchte,
wiederholen kann.

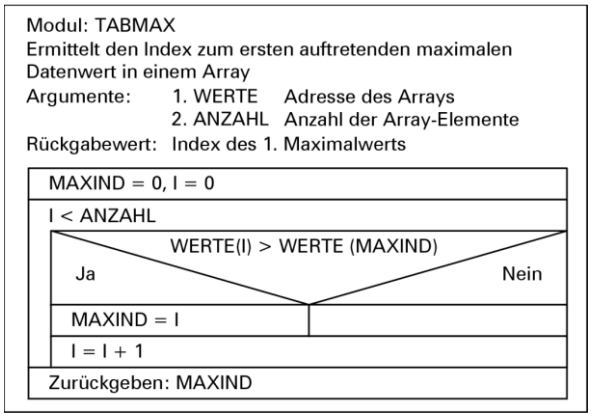
5.
Erstellen Sie ein Struktogramm zu folgendem Programmablauf: Der Anwender gibt eine Zahl ein. Die Zahl wird nur akzeptiert, wenn sie zwischen 0 und 20 liegt (einschließlich der Grenzen). Der Anwender erhält bei Eingabe eines ungültigen Wertes die Gelegenheit, die Eingabe zu wiederholen.
Das Programm berechnet und gibt aus: die Fakultät der eingegebenen Zahl.

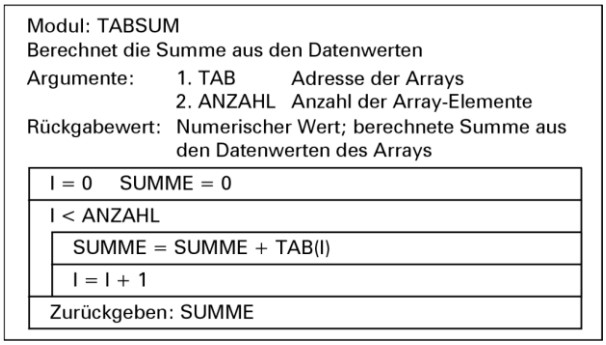
Hinweis

Die Fakultät der Zahl 5 (geschrieben 5!) berechnet sich aus 1 * 2 * 3 * 4 * 5.



- Tabmax, sucht den größten Wert aus einem Array heraus





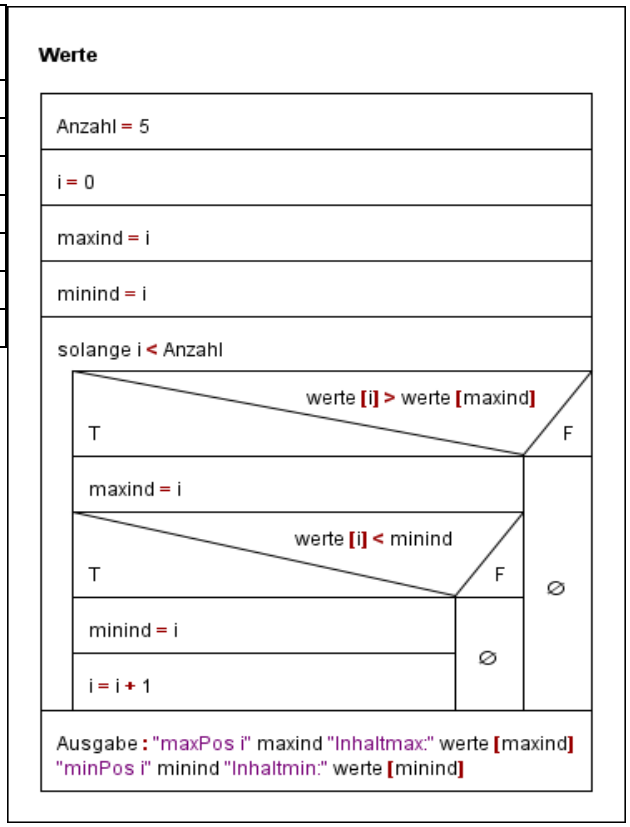
- Tabsum, berechnet die Summe aus den Datenwerten

Array

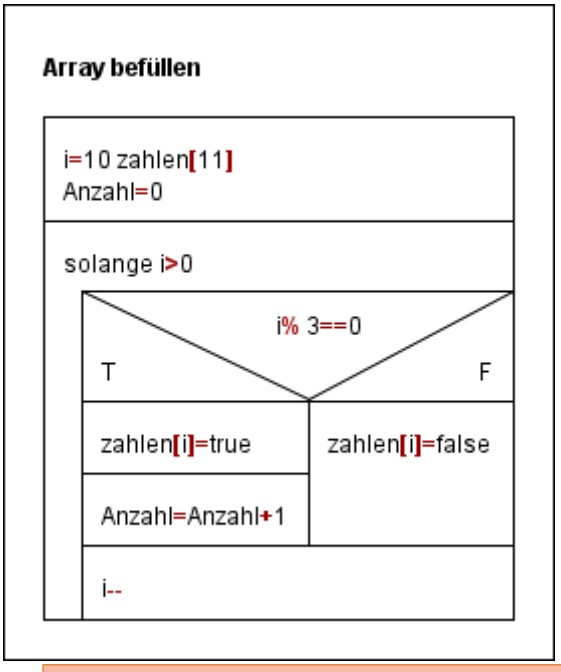
- Das Array darf immer nur einen Datentyp enthalten (byte, short, int, long, float, double)

Schreibtisch- Test

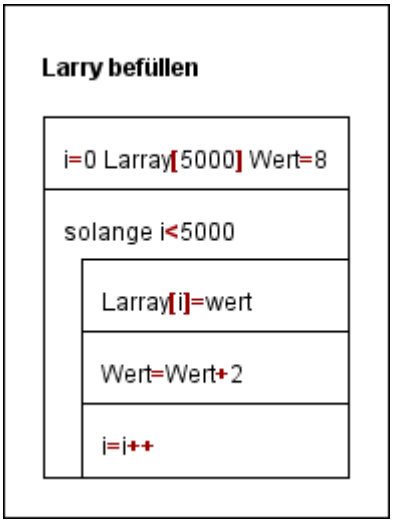
i	maxind	werte [minind]	minind	werte [maxind]
0	0	10	0	10
1	0	10	1	2
2	2	28	1	2
3	2	28	3	1
4	2	28	3	1



Array befüllen:



Larry befüllen:



- Ein Array ist ein Container, mit beliebig vielen Werten die unten einem namen erreichbar ist.

Zusätzlich:

Mit dem Begriff „elementarer Datentyp“ oder Datenelement wird ausgesagt, dass es sich um ein Datum (nicht zu verwechseln mit dem Kalenderbegriff) handelt, das sich aus logischer Sicht nicht mehr weiter untergliedern lässt. Jede im Hauptspeicher abgelegte Information lässt sich aus physischer Sicht weiter untergliedern, z.B. in Bytes oder auch in Bits. Sobald aber ein Datum aus logischer Sicht nicht weiter untergliedert werden kann, handelt es sich um ein elementares, man sagt auch skalares Datum.
Nicht alle Programmiersprachen sind identisch in Bezug auf die Menge der elementaren Datentypen, die sie unterstützen. Darum erhalten Sie hier einen möglichst umfassenden, sprachenübergreifenden Überblick.

Mögliche Klausurinhalte:

- Variablen bestimmte Werte zuweisen (ich habe 8 ist das A oder B)
- Wahr oder falsch (a>b)
- Behauptungen (wahr oder falsch)
- Struktogramm (vorbereitet) zahlen eintrage für eine spezielle Aufgabe
- Struktogramm (bei Herr Scheringer)
- Struktogramme lesen und auswerten
- Struktogramm mit Array schreibtischtest (was verändert sich)
- Struktogramm eines 2-Dimensionalen Array erstellen

Berechnungen: Variablentausch, Minimum, Maximum, durchschnitt, summen, Modulo und Array spiegeln