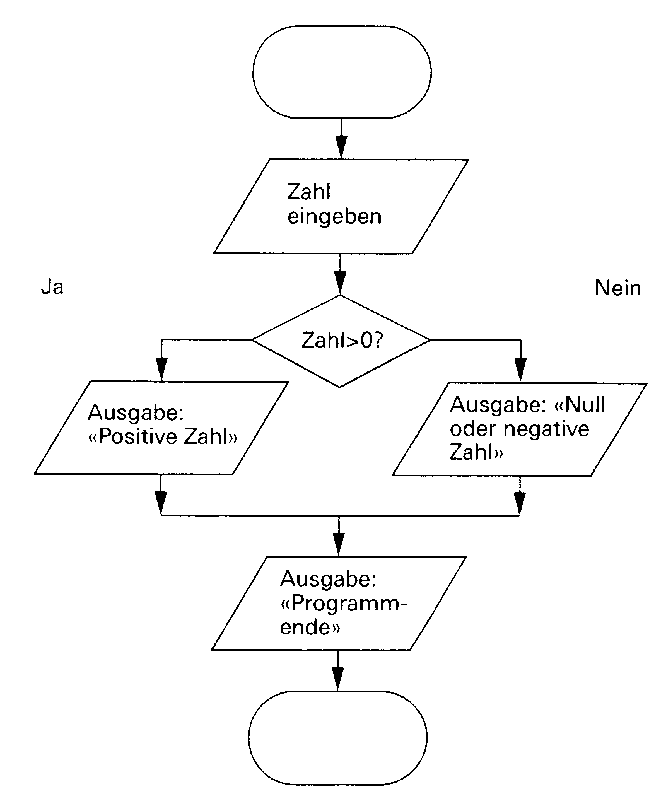
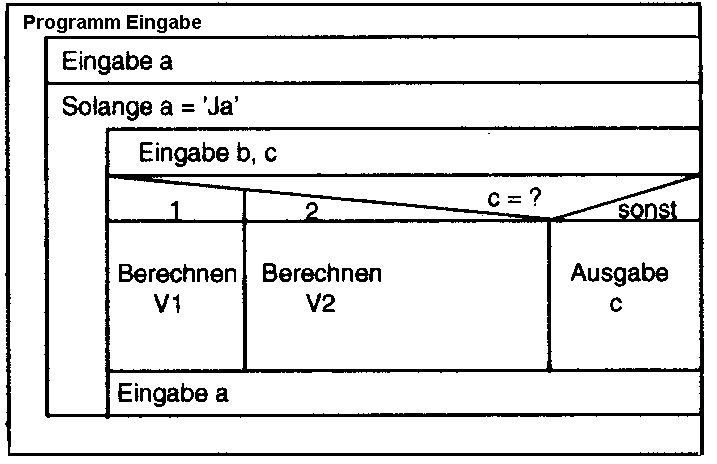
**Modul 319**

***Block 03***

***- Darstellung von Algorithmen -***



****

**Dokumentenversion:**

|  |  |
| --- | --- |
| V1.1 | Erstellung durch Roland Bucher |

**Voraussetzung:**

Dieser Block setzt den Block 1 und 2 des Moduls 319 voraus.

**Aufbau dieses Blocks:**

Der Block visualisiert in den ersten beiden Kapiteln die Darstellungsformen Programmablaufplan und Nassi-Schneidermann. Das letzte Kapitel des Dokumentes enthält div. Übungen zu den Lernzielen.

**Ziele dieses Blocks:**

* Die Lernenden können eine gegebene Aufgabenstellung in Form eines grafisch dargestellten Algorithmus lösen
* Die Lernenden können einen in C#- oder Java-Codeform gegebenen Algorithmus grafisch als PAP darstellen
* Die Lernenden können einen via PAP gegebenen, grafisch dargestellen Algorithmus in C#- oder Java-Codeform realsieren.

**Zeitaufwand:**

Für die Abarbeitung dieses Blockes sind im Unterricht 4 Lektionen vorgesehen. Der Rest ist als Hausaufgabe zu lösen.

**Inhaltsverzeichnis:**

[1 Programmablaufplan 2](#_Toc88438417)

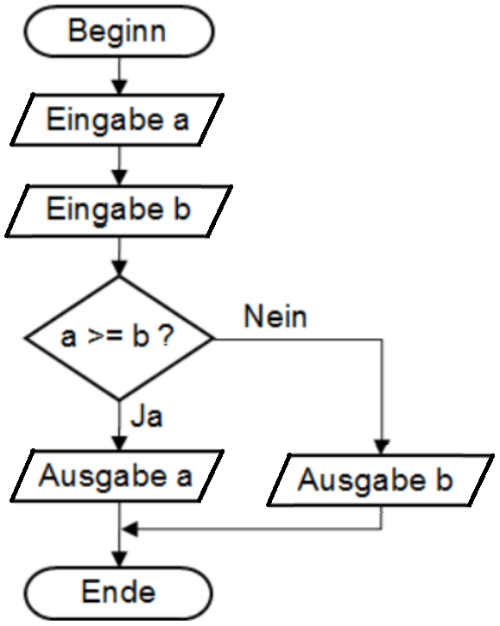
[2 Struktogramme (Nassi-Schneidermann) 3](#_Toc88438418)

[3 Pseudocode 4](#_Toc88438419)

[4 Übungen (180’) 5](#_Toc88438420)

# Programmablaufplan

(Dieses Kapitel ist eine Zusammenfassung ihres Herdt-Buches (Programmieren Grundlagen, Kapitel 3).



Ein Programmablaufplan (PAP) wird auch Ablaufdiagramm, Flussdiagramm oder Blockdiagramm genannt.

Programmablaufpläne sind grafische Darstellungen mit Hilfe genormter Symbole. Sie sind weit verbreitete Hilfsmittel bei der Programmentwicklung. Die Programmstruktur lässt sich bildhaft als Ablauf darstellen. Die Pfeile zeigen dabei die Richtung der Verarbeitung an.

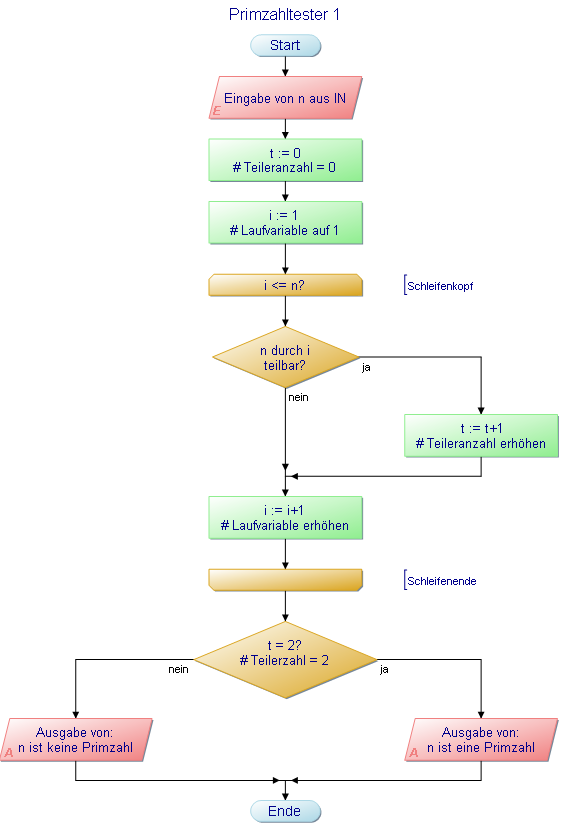
Der PAP reflektiert nicht alle Details des Programms, sondern stellt den allgemeinen Logikfluss dar.

Wenn Sie den PAP korrekt entwickelt haben, ist es nur noch eine Formsache das endgültige Programm zu schreiben. Nachdem das Programm fertig gestellt wurde, können Sie dieses Diagramm ausserdem für die Dokumentation verwenden.

Die einzelnen Symbole werden nachfolgend kurz erklärt.

|  |  |
| --- | --- |
|  | **Verarbeitung**  Das Rechteck ist für Zuweisungen oder Ein- und Ausgabeoperationen vorgesehen. |
|  | **Verbindung**  Zur Verdeutlichung der Ablaufrichtung werden Linien mit einer Pfeilspitze benutzt. |
|  | **Verzweigung**  Bedingungen werden als Raute dargestellt. Eine Verbindungslinie führt hinein, zwei Ver-bindungslinien führen heraus. Je nach Wahrheitswert der Bedingung wird der Ablauf in die eine oder andere Richtung fortgeführt. |
|  | **Manuelle Verarbeitung**  Eingaben des Programmnutzers werden durch ein Trapez oder eine Raute dargestellt. |
|  | **Dokumentation an anderer Stelle**  Durch dieses Symbol wird auf einen anderen PAP hingewiesen, der z. B. ein Unterprogramm darstellt. |
|  | **Verbinder**  Teilt sich ein Programmablauf und wird dann wieder zusammengeführt, wird der Verbinder (Konnektor) eingesetzt. Somit wird vermieden, dass sich Ablauflinien kreuzen. Damit wird die Übersichtlichkeit erhöht. Für größere PAPs, für die mehrere Seiten benötigt werden, wird der Verbinder am Seitenende der vorherigen und Seitenanfang der folgenden Seite verwendet. Zur Identifizierung wird er mit einer eindeutigen Zahl für diese Verbindungsstelle versehen. |
|  | **Grenzstelle**  Eine Grenzstelle kennzeichnet den Anfang und das Ende eines Programmablaufplans. |
|  | **Schleifenbegrenzer**  Zur Darstellung von Programmwiederholungen werden diese zwei Symbole benutzt, die den Anfang und das Ende einer Schleife kennzeichnen.  In der Praxis verwendet man statt dieser Schleifensymbole oft auch Verbindungen, die wieder zu einem vorherigen Verarbeitungsschritt verweisen. |

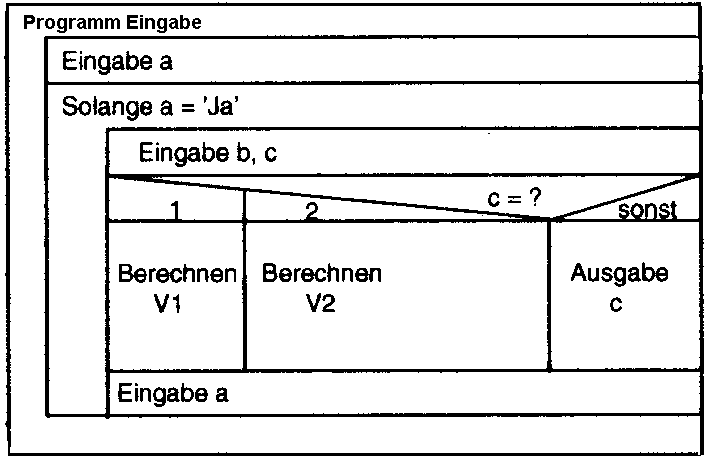
Empfehlenswerte Freeware: **PapDesigner** (**Diagram Editor** (*http://dia-installer.de/*)

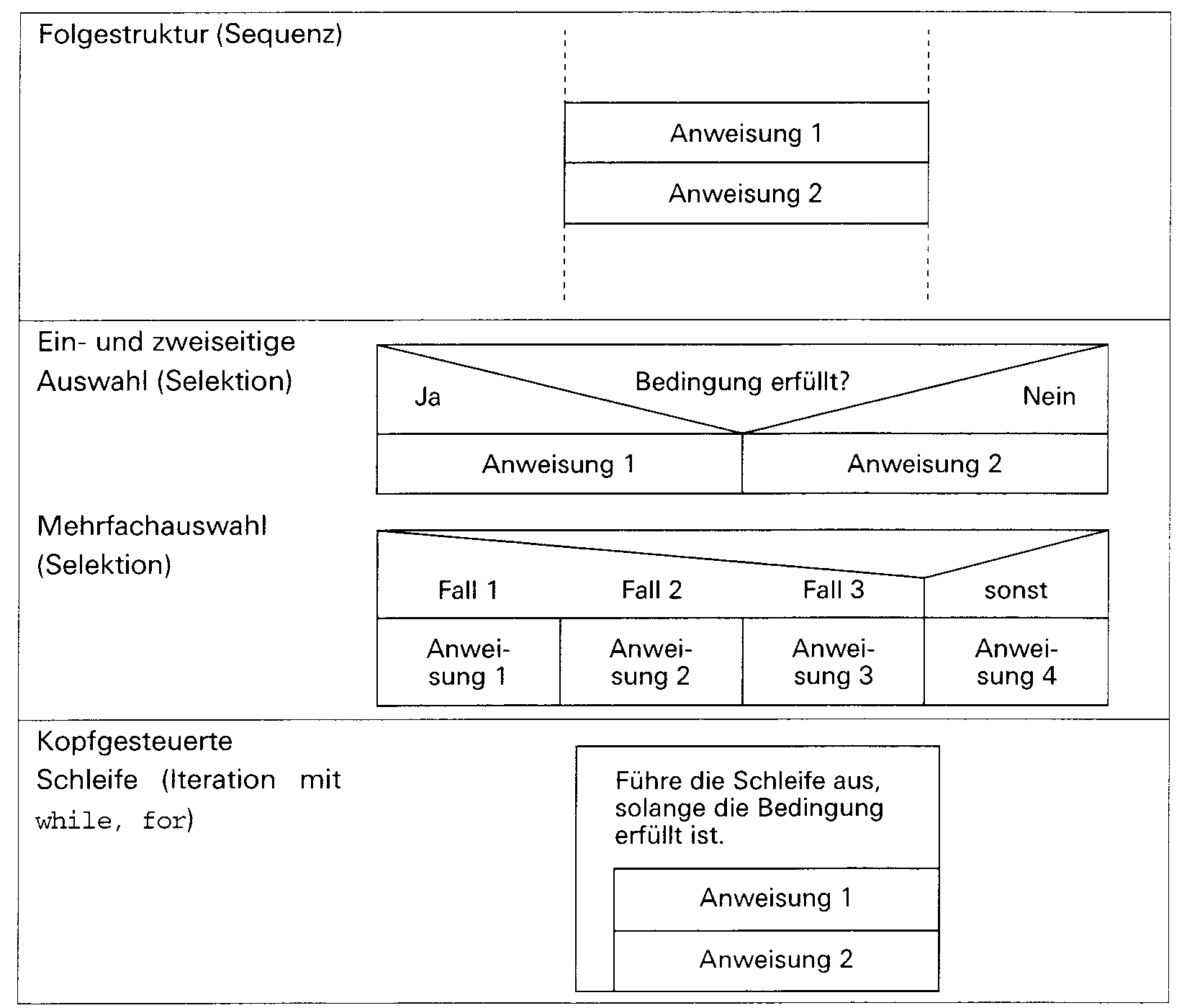


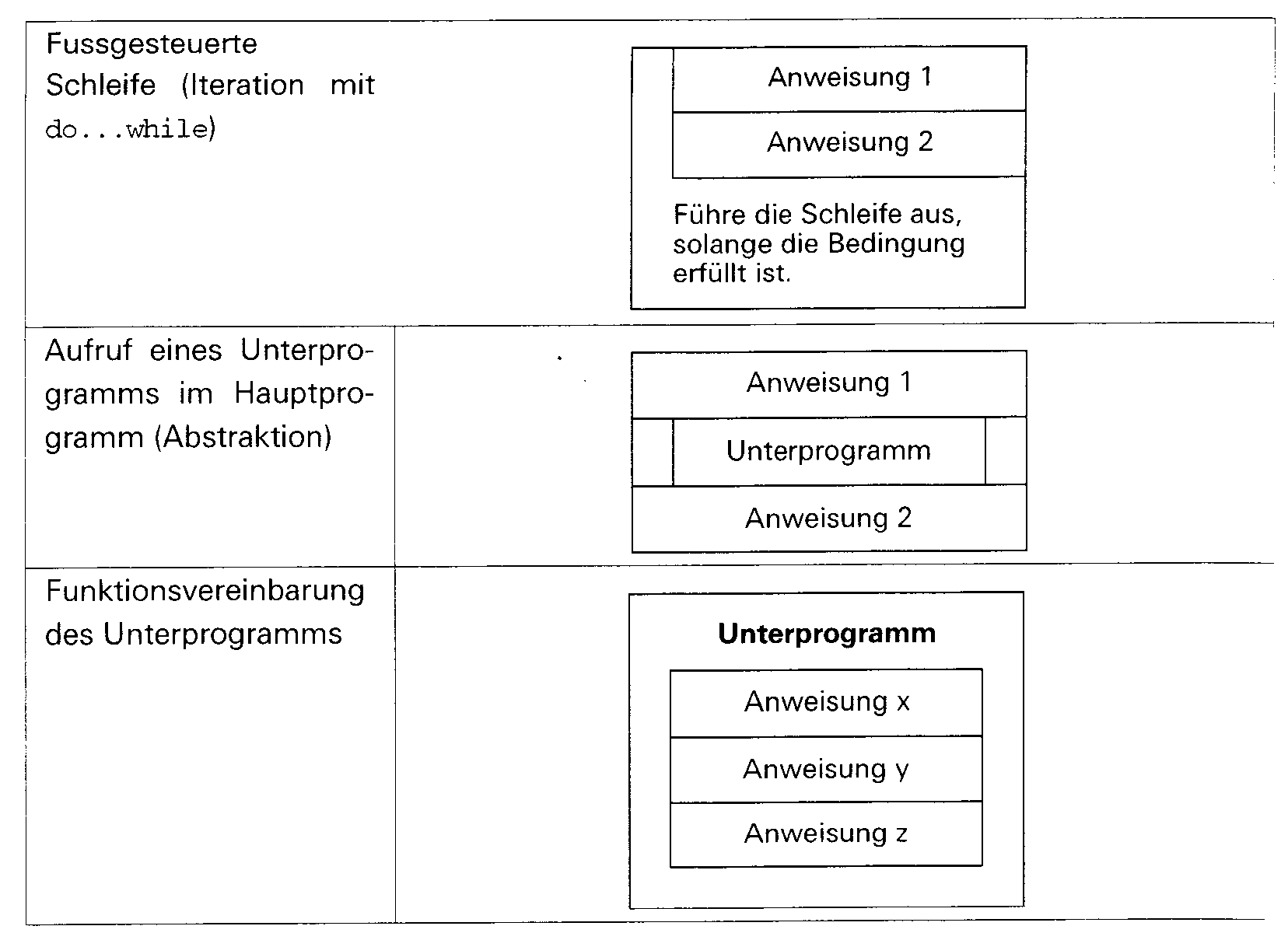
# Struktogramme (Nassi-Schneidermann)

Struktogramme wurden von I. Nassi und B. Shneiderman als Darstellungsmittel für Algorithmen des strukturierten Programmentwurfs entwickelt.

Ein Struktogramm ist die grafische Darstellung eines Programmablaufs in Form eines geschlossenen Blocks, der entsprechend den einzelnen logischen Grundstrukturen in verschiedene unterordnete Blöcke aufgeteilt werden kann. Struktogramme setzen sich aus verschiedenen Symbolen für die verschiedenen Operationsarten zusammen, die von oben nach unten betrachtet werden (top-down).

Bevor man mit dem eigentlichen Programmieren beginnt, sollte man sich Gedanken machen, was das Programm denn überhaupt können soll. Hierzu wird heute überwiegend das Struktogramm mit Symbolen nach Nassi-Schneidermann eingesetzt. Mit diesem grafischen Hilfsmittel wird eine wesentliche Vereinfachung für die Planung, die Programmierung und den Programmtest (z.B. Fehlersuche) erreicht. Die Nassi-Schneidermann Darstellung ist stark verbreitet. Der Vorteil gegenüber dem Flussdiagramm ist, das die Darstellung dem zu programmierenden Algorithmus viel ähnlicher ist. Der Nachteil ist aber, dass diese Darstellung nur von Fachpersonen gelesen werden kann (für Kunden, Marketing etc. ungeeignet).





# Pseudocode

Bei dieser Darstellungsart wird die Programmlogik unter Verwendung von “normalen“ Sätzen beschrieben.

Der Pseudocode ist eine halbformale, textuelle Beschreibung eines Programmablaufs, die sich am Aufbau der Programmiersprachen anlehnt. Der Pseudocode ist im Gegensatz zu den grafischen Struktogrammen nicht genormt. Für Kontrollstrukturen werden oft ähnliche Worte verwendet, wie sie in der Syntax einer Programmiersprache vorkommen. (if…..then….else…. while ……., etc.)

Für die Anweisungen werden verbale Beschreibungen verwendet (z.B. erhöhe i um 1) oder Anweisungen ähnlich einer Programmiersprache (z.B. i = i + 1). Auch Variablen- und Konstantendeklarationen können enthalten sein.

**Ein Beispiel zum Thema Pseudocode:**

Für jeden Angestellten:

Wenn der Angestellte 0 bis 40 Std. gearbeitet hat, dann

Nettolohn = Std.-Anzahl mal Std.-Lohn

Andernfalls

Wenn der Angestellte zwischen 40 und 50 Std. gearbeitet hat, dann

Nettolohn = 40 mal Std.-Lohn

Plus (Std.-Anzahl-40) mal Std.-Lohn mal 1,5

Andernfalls

Nettolohn = 40 mal Std.-Lohn

Plus 10 mal Std.-Lohn mal 1,5

Plus (Std.-Anzahl-50) mal doppeltem Std.-Lohn

Steuern vom Gehalt abziehen

Lohnblatt ausdrucken

Löhne in Fibu verbuchen

Aufgabe gelöst

# Übungen (180’)

**Aufgabe 01 (Bezahlvorgang mit Twint)**

**Lernziel**: Die Lernenden können eine gegebene Aufgabenstellung in Form eines grafisch dargestellten Algorithmus lösen

**Zeit**: 20’

**Aufgabe**: Sie möchten einem Kollegen CHF 25.-- via Twint zukommen lassen, da er Tickets besorgt hat.

Erstellen Sie ein PAP und ein Struktogramm, die Schritt für Schritt zeigen, was sie tun müssen um die Überweisung auszuführen. Beachten sie, dass sie zuerst das Mobile mit der Gesichtserkennung entsperren müssen, danach die TwintApp starten und das korrekte >Twint-Passwort eingeben müssen. Danach können sie Mobilenummer, Betrag und Nachricht eingeben. Sollten sie nicht genügend Geld auf dem Konto haben oder die Mobilenummer ungültig sein, erscheinen entsprechende Meldungen.

gemacht

**Aufgabe 02 (Algorithmus als PAP visualisieren)**

**Lernziel**: Die Lernenden können einen gegebenen Programcode (Algorithmus) in Form eines PAP visualisieren.

**Zeit**: 25’

**Aufgabe**: Visualisieren sie den gegebenen Programmcode als PAP

static void Main(string[] args)

{

int masse = 0, vergleichsmasse = 64; ;

do

{

Console.Write("Masse in Kg [1 bis 127] eingeben:");

} while (int.TryParse(Console.ReadLine(), out masse)==false || masse>127 || masse<0);

Console.Write("Um die Masse {0} zu waegen benoetigen sie:");

while (masse>0)

{

if (masse >= vergleichsmasse)

{

masse -= vergleichsmasse;

Console.Write("{0}Kg ", vergleichsmasse);

}

vergleichsmasse /= 2;

}

Console.ReadLine();

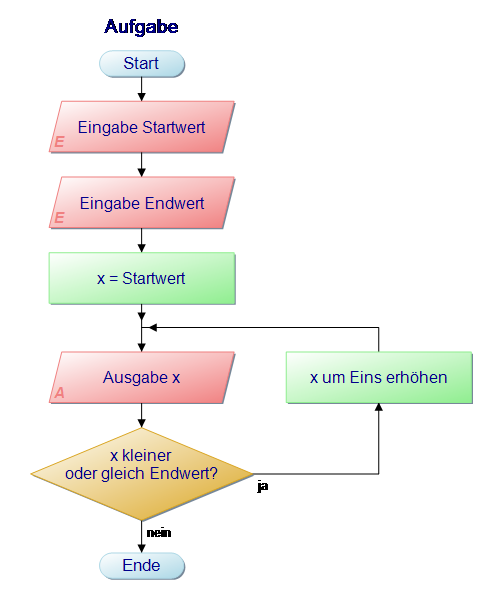
}

**Aufgabe 03 (PAP in Programmcode umsetzen)**

**Lernziel**: Die Lernenden können einen via PAP gegebenen, grafisch dargestellen Algorithmus in C#- oder Java-Codeform realsieren.

**Zeit**: 15’

**Aufgabe**: Setzen sie das gegebene PAP in Programmcode um.

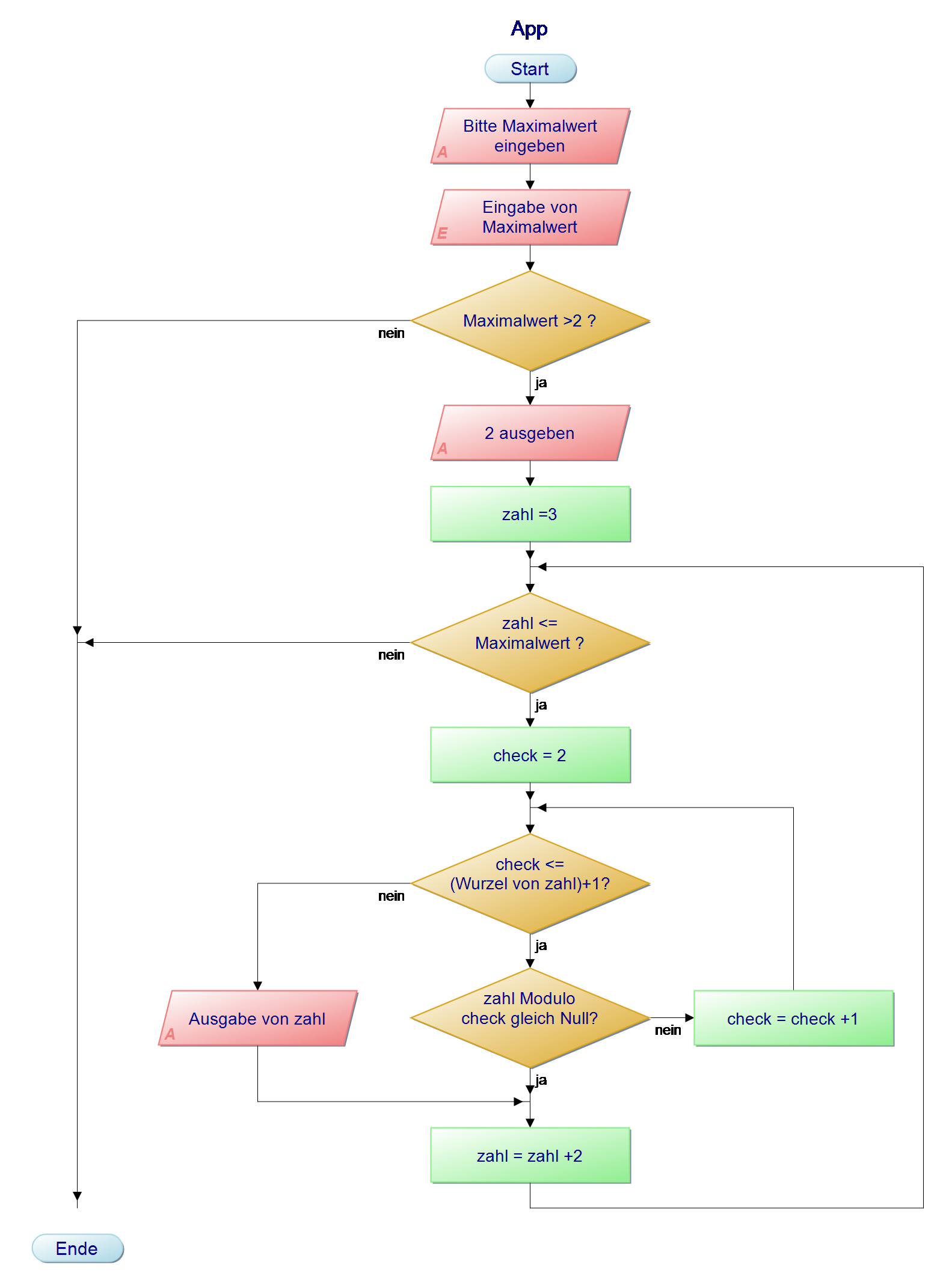
**

**Aufgabe 04 (PAP in Programmcode umsetzen)**

**Lernziel**: Die Lernenden können einen via PAP gegebenen, grafisch dargestellen Algorithmus in C#- oder Java-Codeform realsieren.

**Zeit**: 30’

**Aufgabe**: Setzen sie das gegebene PAP in Programmcode um. Welchen Zweck verfolgt der Algorithmus?



**Aufgabe 05 (Fibonacci)**

**Thema:** Eingabe, Verarbeitung (nur Anweisungen und Abfolgen), Ausgabe

**Lernziele**: Die Lernenden können eine gegebene Aufgabenstellung in Form eines grafisch dargestellten Algorithmus lösen und diesen in C#- oder Java-Codeform realsieren.

**Zeit**: 45’

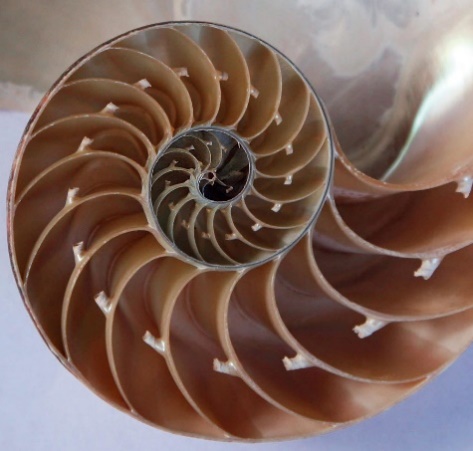
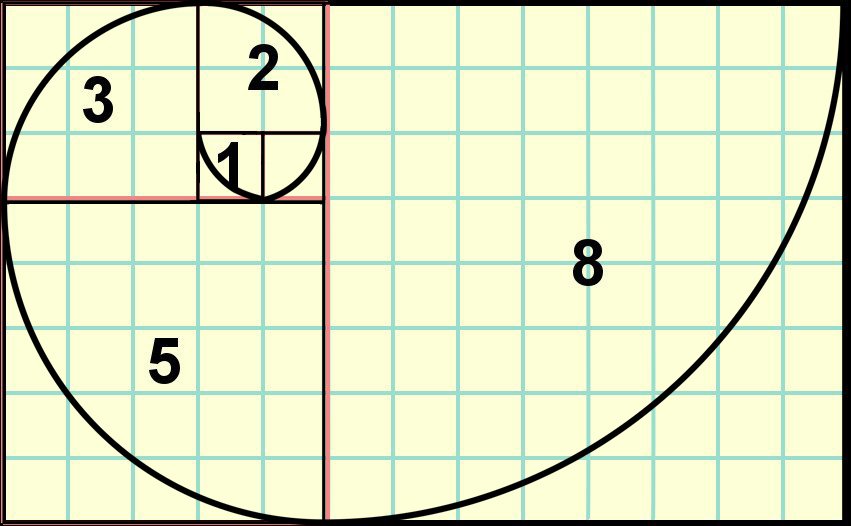
**Infos**: Kennen sie die Fibonacci Folge? Sie liest sich folgendermassen:1, 1, 2, 3, 5, 8, 13, 21, 34, 55...

Man startet mit zwei Zahlen des Wertes Eins (1). Die nachfolgenden Zahlen ergeben sich jeweils aus der Summe der beiden Vorgänger: 1+1 ->2 1+2à3 2+3à 5, etc.

Wo benötigt man dies? à <https://youtu.be/kkGeOWYOFoA>

Die Fibonacci Folge trifft man in der Physik, Biologie und Astronomie oft an. Man benutzt sie z.B. um eine Kaninchenpopulation (Anzahl Kaninchenpaare in jedem Monat) vorauszusagen.

Mit der Fibonacci Folge kann man auch die Fibonacci-Spirale herleiten, die z.B. der Form von Muscheln, Schneckenhäuser, etc. entspricht.



Das Video zeigt, wie man die Fibonacci-Spirale aufzeichnen kann: <https://youtu.be/D-27wYXd5J0>

Die Fibonacci-Folge steht in einem unmittelbaren Zusammenhang zum Goldenen Schnitt, der in der Fotografie und Kunst angewendet wird (siehe Fotos/Bilder am Schluss dieser Aufgabe)

**Aufgabe**: Sie haben die Aufgabe ein Programm zu erstellen, das den Benutzer auffordert eine «Endzahl» einzugeben (z.B. 60). Ihp Programm soll danach die Fibonacci-Folge bis zur Endzahl ausgeben: 1, 1, 2, 3, 5, 8, 13, 21, 34, 55...  
Erstellen sie Programmablaufplan (PAP) und Applikation.



**Aufgabe 06 (Quadratische Gleichungen lösen)**

**Lernziel**: Die Lernenden können eine gegebene Aufgabenstellung in Form eines grafisch dargestellten Algorithmus lösen

**Zeit**: 45’ (PAP) + 45’ Umsetzung PAP in Code

**Aufgabe**: Schon oft haben sie in der Mathematik Aufgaben versucht zu lösen, bei denen sie am Schluss nicht mehr weiterkamen, da die gesuchte Variable im Quadrat vorliegt. (z.B. ). Sie hörten, dass wenn die gesuchte Variable im Quadrat vorkommt, es sich um eine quadratische Gleichung handle. Die konnten sie bisher nicht lösen. Das möchten sie ändern.

Mit Hilfe des nachfolgenden Textes (der von<https://de.wikipedia.org/wiki/Quadratische_Gleichung>abgeleitet wurde), möchten sie ein Programm zur Lösung von quadratischen Gleichungen erstellen. Schritt 1 ist, ein PAP für die verschiedenen Fälle zu erstellen. Erstelln sie einen Programmablaufplan, damit sie künftig nach diesem Muster einfach quadratische Gleichungen lösen können und auf dessen Basis ein Programm umsetzen können.

**Infos**: Eine quadr. Gleichung ist eine Gleichung, die sich in der **allgemeinen Form** befindet und a≠ 0 ist.

Eine quadratische Gleichung hat immer zwei Lösungen. Diese beiden Lösungen können aber manchmal zusammenfallen (es erscheint dann als gäbe es nur eine Lösung) oder teilweise im Bereich der komplexen Zahlen liegen (diese bestehen aus Real und Imaginärteil).

Falls es komplexe Lösungen gibt, soll unser Programm den Text ausgebenà «Die Lösung ist im imaginären Bereich. Wenn wir ausschliesslich die für uns relevanten und lösbaren reellen Zahlen, betrachten, so hat eine quadratische Gleichung null bis zwei Lösungen.

**Falls b= 0 ist,** handelt es sich um eine **reinquadratische Gleichung**.

Wenn bei einer reinquadratischen Gleichung der Term ist, dann kann man die Lösung durch einfaches radizieren lösen. Z.B.

Wenn bei einer reinquadratischen Gleichung das Produkt ist, gibt es keine reellen sondern zwei imaginäre Lösungen.

**Falls b≠ 0 ist,** handelt es sich um eine **gemischtquadratische Gleichung**.

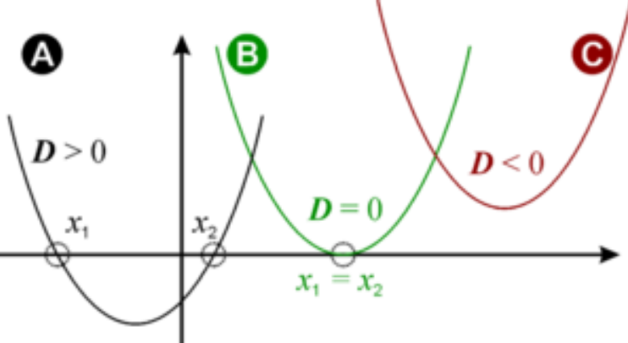
Wenn bei einer gemischtquadratischen Gleichung der Koeffizient c = 0 ist, gibt es zwei mögliche Lösungen. Wir klammern dazu ein x aus. Die eine Möglichkeit ist, dass x = 0 ist, die andere Möglichkeit finden wir, wenn wir nach x auflösen. Z.B. Nun gibt es zwei Möglichkeiten: x ist 0, oder 3x-5 ist 0. à

Wenn aberr bei einer gemischtquadratischen Gleichung der Koeffizient c ≠ 0 ist, kann man anhand der Diskriminianten D (die Anzahl und Typen der Lösungen ermitteln:

Falls D positiv ist: gibt es zwei Schnittpunkte mit der x-Achse à es gibt zwei versch. reelle Lösungen.

Falls D Null ist: gibt es einen Schnittpunkt mit der x-Achse à es gibt eine (doppelte) reelle Lösung

Falls D negativ ist: gibt es keinen Schnittpunkt mit der x-Achse à es gibt keine reellen, sondern imaginäre Lösungen.



Falls D nicht negativ ist, kann man die reellen Lösungen nach folgender Formel berechnen:

=

Übrigens: Falls a=1 ist, kann man die quadr. Gleichung auch in der einfacheren **Normalform** darstellen

. Die Diskriminiante errechnet sich dann: