Programmierung I

- Wie sagen wir einem Computer, was wir von ihm wollen, was er für uns tun soll?
- Die Aufgaben, die ein Rechnersystem erfüllen soll, werden in Programmen formuliert.

Analogie

- Ein Programm zu schreiben, ist so ähnlich, wie ein Koch- oder Backrezept zu formulieren.
- Ein Rezept enthält,
  - Die Zutaten
  - Die Werkzeuge und Hilfmittel
  - Die Verarbeitungsschritte in der gebotenen Reihenfolge

Mein lieber Junge,

wenn Du die leckeren Kekse haben willst, die Du doch immer so gerne isst, so kannst Du Dir selber welche backen. Du brauchst dazu 200g gute Butter, 200g Zucker, 300g Mehl, eine Priese Salz, und etwas Rum oder Cognac (der soll mit in den Teig für die Kekse!), Variieren kannst Du noch mit Kakao, wenn es Schwarzweißgebäck werden soll oder mit einem Kleckschen roter Marmelade.
Du nimmst eine Schüssel. Zuerst gibst Du Butter und Zucker in die Schüssel und rührst beides gut schaumig. Die Butter darf dabei nicht zu kalt sein. Dann gibst Du das Mehl und die weiteren Zutaten (Salz Rum oder Cognac) dazu. Nun mußt Du das Ganze ordentlich durchkneten bis der Teig recht geschmeidig ist. Eventuell gibst Du noch ein wenig Flüssigkeit (kaltes Wasser) dazu (aber nicht zu viel!). Wenn Du Schwarzweißgebäck machen willst, mußt Du den Teig teilen und die eine Hälfte mit 2 Esslöffeln Kakao vermengen. Der fertige Teig soll dann ungefähr eine halbe Stunde ruhen.
Du machst dann ein paar Rollen vom schwarzen und weißen Teig und verdrehst die miteinander, das gibt dann ein sehr schönes Muster.
Schließlich schneidest Du von der fertigen Teigrolle Scheiben, bezuckerst diese am Rand und legst sie auf das Backblech. Wenn Du kein Schwazweißgebäck machen willst, kannst Du die weißen Kekse mit einem Klecks Marmelade verzieren. Das sieht hübsch aus und schmeckt auch sehr gut. Die Kekse werden dann bei nicht allzu großer Hitze (150°) im Backofen gebacken.
Nun wünsche ich Dir recht gutes Gelingen und guten Appetit.

Deine Oma

Programme

- Um Programme herzustellen braucht man Werkzeuge (Compiler, Editor, Debugger, IDE).
- Programme bestehen aus
  - Beschreibung der Daten durch Datentypen und Variablen.
  - Algorithmen, die die Verarbeitungsschritte beschreiben.
- Zur Formulierung von Programmen werden Programmiersprachen verwendet.

Programmiersprachen

- In der „Steinzeit“ der Programmierung herrschte Maschinencodeprogrammierung vor.
- Seit Mitte der 1950-er Jahre gibt es Assemblersprache (1:1 zu Maschinensprache, aber etwas komfortabler).
- Fortran als erste höhere Sprache gibt es seit 1954, später algol 60.
- Weitere höhere Programmiersprachen waren PL/1, Cobol, später Pascal und c.
- Der nächste Meilenstein waren die Objektorientierten Sprachen mit Simula 67 und später smalltalk und C++.

Beispiel Berechnung des arith. Mittels

- Die Berechnung des arithemetischen Mittels von n Zahlen fällt uns nicht schwer. Wir dividieren die Summe der Zahlen durch die Anzahl der Zahlen - fertig.
- Eine Schritt für Schritt Erklärung, wie man es macht, ist schon schwieriger.

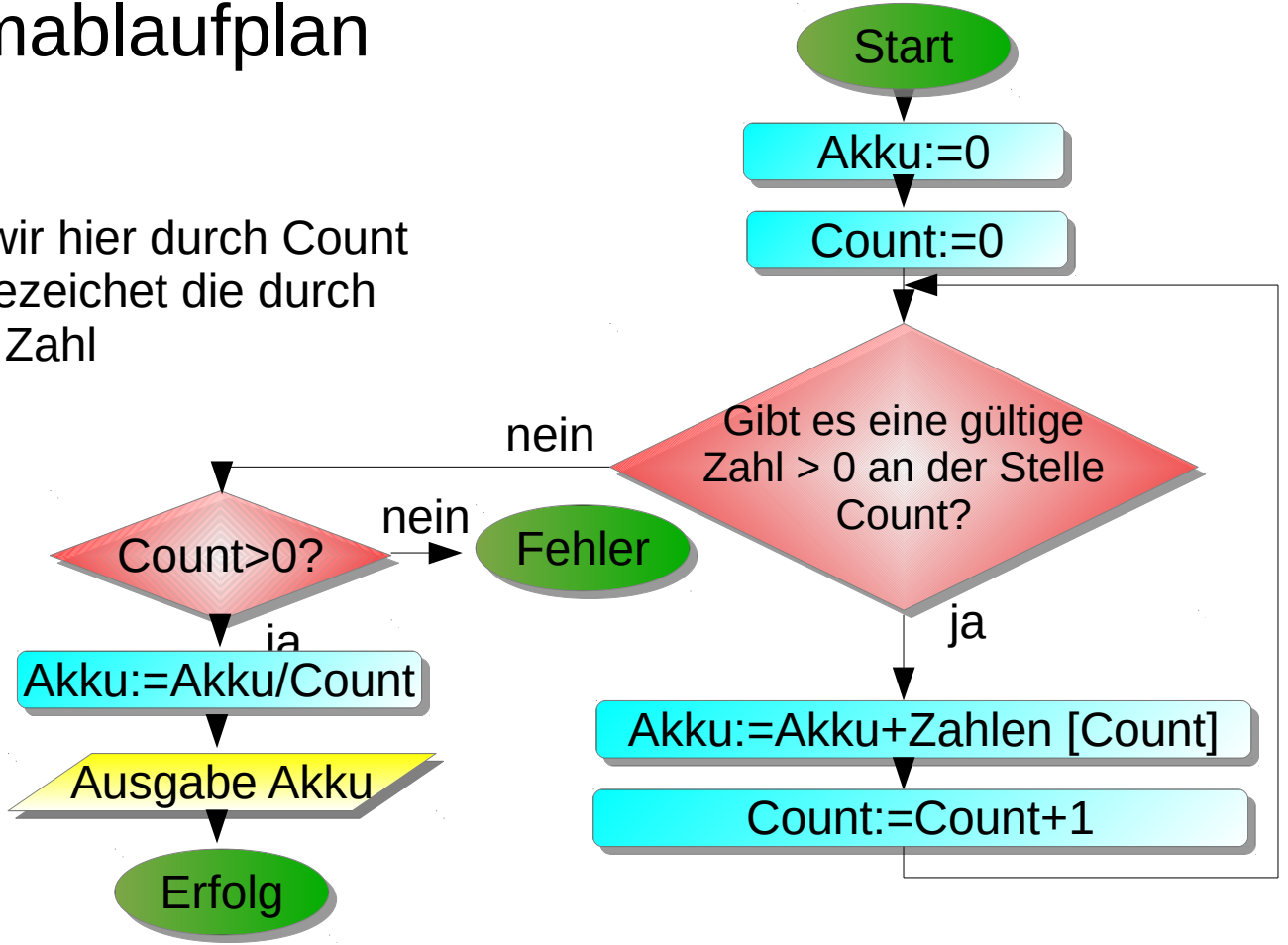
Durchschnitt (verbal)

Die Zahlen liegen als Zahlenfeld (hintereinander) vor. Hinter der letzten Zahl stehe eine 0 (z.Bsp.: 3,1,5,5,2,**0**)
Marker: Variable zum Markieren der jeweils aktuellen Zahl
Akku: Variable zum Aufsummieren der Zahlen und zum Rechnen
Unter einer Variablen wollen wir zunächst einen Ort verstehen, in dem wir eine Zahl aufbewahren können. Wir können die Zahl in der Variablen lesen und eine andere Zahl in die Variable (drüber-)schreiben.

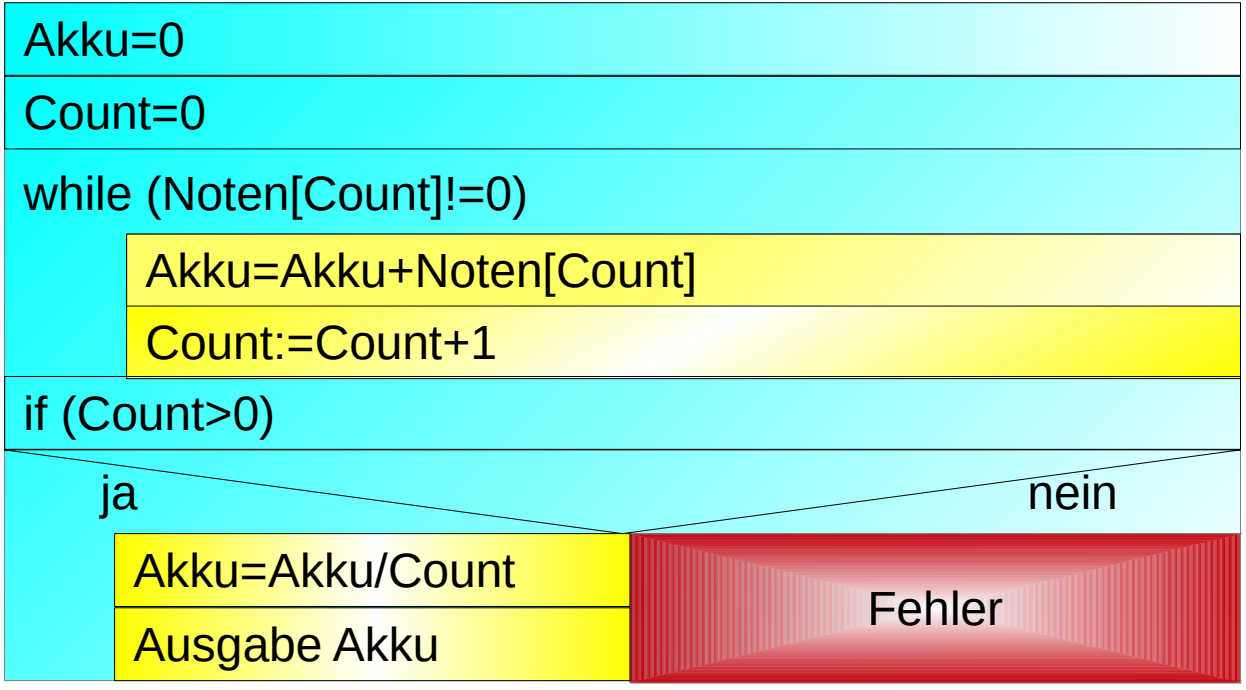
1	Akku auf den Wert 0 setzen	→ 2
2	erste Zahl markieren (Marker bekommt den Wert 0)	→ 3
3	gibt es an der markierten Stelle eine Zahl ? (Wert ungleich 0)	j→ 4 n→ 6
4	Addiere Wert in Akku und die markierte Zahl und schreibe das Ergebnis wieder nach Akku	→ 5
5	markiere die nächste Zahl (erhöhe den Wert von Marker um 1)	→ 3
6	Dividiere den Wert im Akkumulator durch die Anzahl der Werte (Wert von Marker)	→ 7
7	Ausgabe des Wertes von Akku	→ Ende

Programmablaufplan (PAP)

Marker erstzen wir hier durch Count
Zahlen[Count] bezeichet die durch Count markierte Zahl



Struktogramm (Nassi-Shneiderman-Diagramm)



C-Programm

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

int Noten[]={5,2,3,4,5,5,2,3,4,5,0};//38/10

int main()
{
  int Akku=0, Count=0;
  while(Noten[Count]!=0)
  {
    Akku=Akku+Noten[Count];
    Count=Count+1;
  }
  if(Count>0)
  {
    Akku=Akku/Count;
    printf("Durchschnitt: %d\n",Akku);
  }else printf("Fehler - Division durch 0\n");
  return 0;
}
```

Um das c-Programm ausführen zu können, muss es zuvor compiliert werden, nehmen wir an, es ist in einer Datei arimitt.c gespeichert.

Dazu wird ein Werkzeug, der Compiler benötigt.

Die Lehrveranstaltung geht von einem **Linux Betriebssystem** aus. Der verwendete Compiler heißt gcc.

```
gcc arimitt.c -o arimitt
```

Um es in der Konsole auszuführen, gibt man ein:

```
./arimitt
```

Linux

- Es wird dringend empfohlen, ein Linux auf einem passenden Rechner zu installieren. Linux wird auch in anderen Modulen genutzt!
- Folgende Möglichkeiten bestehen dazu:
  - Linux in einem dual-boot-System in einer gesonderten Partition installieren (hierfür sollte man schon über Erfahrung verfügen und wissen, was man macht).
  - Linux in einer virtuellen Maschine installieren.
  - Linux (Knoppix) von einem Stick booten.
  - Linux auf einem RaspberryPi installieren und nutzen.
  - Linux auf einem älteten Laptop/PC installieren (sollte aber schon vorzugsweise ein 64-bit-System sein).
- Recherchieren Sie dazu im Netz, es gibt sehr viele Anleitungen. Wenn Sie ein bestehendes System verwenden, sichern Sie Ihre Daten vorher!

Gcc für Windows

- Als Alternative kann unter Windows MinGW installiert werden. Das ist ein gcc für Windows.
- Eine Anleitung gibt es unter https://wiki.freitagsrunde.org/C_(GCC_unter_Windows)
- Eine andere Alternative ist Cygwin, ebenfalls ein gcc für Windows.

Anmerkungen zum Programm:
Die Zeilen 1 und 2 enthalten Includeanweisungen. An diesen Stellen werden die in < > eingeschlossenen Dateien in den Programmtext eingefügt. Die Dateien befinden sich in einem Systempfad, der plattformabhängig ist (Linux: /usr/include). Diese Dateien stellen die Verbindung zu Bibliotheken dar, die vorgefertigte Programmbausteine enthalten.

Zeile 4 enthält ganze Zahlen in einer Vektorvariablen. Die einzelnen Zahlen erhält man durch Indizieren (Zahlen[0] ist eine 5, Zahlen[1] ist eine 2 usw.)

Zeile 6 enthält den Kopf der main-Funktion. Diese Zeile markiert den Einstieg in das Programm.

Zeilen 7-20 bilden den Körper der main-Funktion, Jede Funktion besteht aus Kopf und Körper.

In Zeile 8 werden zwei Variablen (Speicherplätze) vereinbart. Sie tragen die Namen Akku und Count und können ganze Zahlen aufnehmen, sie werden mit dem Wert 0 initialisiert.

Die Zeilen 9-13 beschreiben eine Schleife. Solange an der durch Count markierten Stelle in den Zahlen eine von 0 verschiedene Zahl steht, wird diese zu Akku hinzuaddiert, anschließend wird Count um eins erhöht. Und die Bedingung in Zeile 9 erneut geprüft. Erreichen wir die Zahl 0 in den Zahlen, wird dieser Kreislauf (diese Schleife) verlassen.

Zeile 14 -18 beschreiben eine bedingte Anweisung. Die Zeilen 16 und 17 werden nur ausgeführt, wenn Count einen Wert von mindestens 1 beinhaltet. Dann erfolgt die Division und die Ausgabe. Anderenfalls erfolgt die Ausgabe einer Fehlermeldung (Zeile 18).

In Zeile 19, wird das Programm mit **`return 0;`** verlassen.

Bei der Ausführung des Programms wird es den Wert 3 als arithmetisches Mittel ausgeben, richtig wäre aber 3,8. Die Ursache für diesen Fehler ist die ganzzahlig rechnende Arithmetik. Es wird nur der ganzzahlige Quotient gebildet. Es gibt verschiedene Möglichkeiten, dies durch Änderungen am Programm zu beheben.

Programmbuild

- Um ein c-Programm ausführen zu können, muss es zuvor compiliert werden. Der Compiler überführt den Quelltext in einen Code, der von der Maschine ausgeführt werden kann.
- Dies kann in
  - einer ide (integrated development environment) erfolgen.
  - der Konsole durch Aufruf des Compilers erfolgen (→ s. 10)
- Der Programmbuild selbst besteht aus mehreren Schritten (Precompiler, Compiler, Linker).
- Weitere Werkzeuge sind Debugger, die eine Ausführung Schritt für Schritt ermöglichen.

Grundelemente von c

- **Großbuchstaben : A...Z**
- **Kleinbuchstaben : a...z**
- **Ziffern : 0...9**
- **Sonderzeichen : ( ) [ ] { } < > + - * / % ^ ~ & |**
  _ = ! ? # \ , . ; : ' "
- **Ersatzsymbolfolgen**

#	=	??=
[	=	??(
]	=	??)
^	=	??'
{	=	??<
}	=	??>
|	=	??!
~	=	??-

Bei der Verwendung von Ersatzsymbolfolgen ist beim Compilieren die Option -trigraphs anzugeben (werden eher selten verwendet)
gcc -trigraphs . . . .

- **white spaces :**

Leerzeichen, Zeilenende(\n), Tabulator(\t), vertikaler Tab(\v), Seitenvorschub(\f), Bell(\a), Backspace(\b), Carriage return(\r)

Der Debugger

- Ein Debugger ist ein Werkzeug, mit dem man ein Programm während der Ausführung „belauschen“ kann. Man kann es Zeile für Zeile ausführen und Werte einsehen.
- Unter Linux gibt es mehrere Debugger, sehr komfortabel ist kdbg.
- Um ein Programm mit einem Debugger untersuchen zu können, muss es mit -g durch den Compiler gcc kompiliert werden.

```
gcc -g arimitt.c -o arimitt
```

```
kdbg arimitt
```

Der Debugger

- In der Konsole kann man den Debugger gdb nutzen.
- Das Programm ist ebenso mit gcc -g zu compilieren.
- Zur Bedienung des gdb findet man viele Hinweise im web.
- Empfohlene Suche: gdb getting started
- man gdb hilft auch erst mal .