

Inhaltsverzeichnis

1. Grundlagen: Struktogramme.....	1
1.1. Ziele.....	1
1.2. Rechneraufbau.....	1
1.3. Programmaufbau.....	2
1.4. Daten.....	3
1.4.1. Datentypen.....	3
1.4.2. Variable definieren.....	3
1.4.3. Konstante.....	4
1.4.4. Initialisierung einer Variablen.....	4
1.4.5. Variablen einen Wert zuweisen.....	4
1.4.6. Den Inhalt einer Variablen um 1 erhöhen.....	4
1.4.7. Den Inhalt von zwei Variablen austauschen.....	4
1.5. Befehle / Algorithmus.....	5
1.5.1. Algorithmus.....	5
1.6. Pseudocode und Struktogramm.....	6
1.6.1. Struktogrammeditor.....	7
1.7. Beispiele: Struktogramme.....	7
1.7.1. Beispiel: flaeche.....	7
1.7.2. Beispiel: tausche.....	7
1.7.3. Beispiel: maxi.....	7
1.7.4. Beispiel: summe1.....	7
1.7.5. Beispiel: summe2.....	7
1.7.6. Beispiel: pow.....	8
1.7.7. Beispiel: teiler1.....	8
1.7.8. Beispiel: teiler2.....	8
1.8. Beispiel: Pseudocode.....	8
1.8.1. Was gibt der folg. Algorithmus aus?.....	8
1.9. Übung: Struktogramme.....	9
1.10. +Flussdiagramme.....	9

1. Grundlagen: Struktogramme

1.1. Ziele

In diesem Kapitel soll geklärt werden,

- ☒ wie ein Computer aufgebaut ist
- ☒ wie man einfache Algorithmen unter Verwendung eines Struktogrammeditors erstellt

1.2. Rechneraufbau

Ein Computer besteht (vereinfacht) aus folgenden Komponenten:

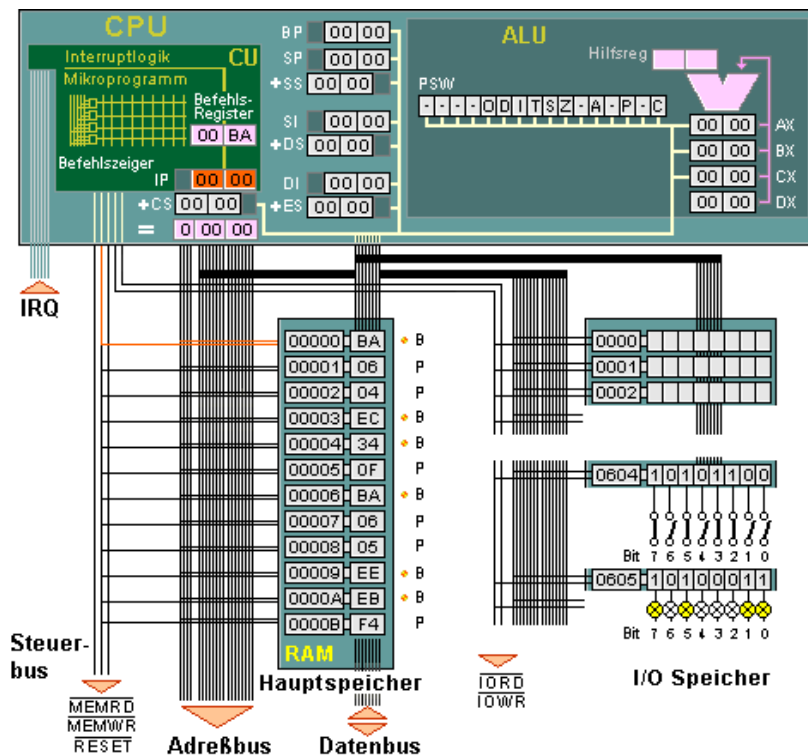
- ☒ **CPU**
 - ☐ BIU (Bus-Interface-Unit)
 - ☐ ALU (Arithmetisch-Logische Einheit)
 - ☐ AKKUMULATOR (Register, das das Ergebnis einer Operation enthält)
 - ☐ Datenregister
- ☒ **RAM** (Random Access Memory)

☒ **BUS** (Verbindung CPU<->RAM)

- ☐ Datenbus
- ☐ Adressbus
- ☐ Steuerbus

☒ **IO-Subsystem**

- ☐ Anschluss von peripheren Devices (Harddisk, Monitor, Tastatur, ...)



Abarbeitung eines Programmes:

1. Befehl und Daten laden (RAM->CPU)
2. Befehl dekodieren (ALU)
3. Befehl ausführen/Daten speichern (ALU) (CPU->RAM)

1.3. Programmaufbau

Im RAM befindet sich das Programm.

Es wird auch Prozess genannt und besteht aus folgenden Teilen:

Datensegment (DS)	Daten, die während der gesamten Laufzeit des Programms zur Verfügung stehen (globale Daten)
Stacksegment (SS)	Daten, die nur zu bestimmten Zeiten zur Verfügung stehen (lokale Daten)
Codesegment (CS)	Die Befehle des Programms.
Heap	Hier können zur Laufzeit des Programms noch Daten abgelegt werden.

Es gilt also:

Programm := Daten + Befehle (Algorithmus)

1.4. Daten

Zunächst wollen wir die Daten genauer besprechen.

1.4.1. Datentypen

Wir unterscheiden folg. Arten/Typen von Daten:

ART	Bereich	Namen	Variablen definieren
Nummerische Datentypen	Ganze Zahlen	short, int, long	int anzahl;
	Reelle Zahlen	float, double	float preis;
Logische Datentypen	true,false	Werden als int interpretiert: 0 ... false sonst true oder boolean	boolean fertig;
Zeichen	ASCII Zeichen,	char	char zeichen;
Texte	Zeichenketten	String od. char[]	String name; od. char name[10];

1.4.2. Variable definieren

☒ Sind Speicherstellen im RAM, deren Wert verändert werden kann. Sie besitzen:

- ☐ einen Namen
- ☐ einen Wert
- ☐ eine Position im RAM (Adresse genannt)

☒ Bsp für die Definition von Variablen:

- ☐ int anzahl;
- ☐ float preis;
- ☐ char zeichen;

☒ Verwenden Sie **aussagekräftige** Namen:

- ☐ summe, mittelwert, zaehler, errorNumber, fertig, gefunden,

1.4.3. Konstante

Sind Speicherstellen im RAM, deren Wert NICHT verändert werden kann.

Bsp. Für namenlose Konstante: 3.14

Bsp. Für benannte Konstante: const int mwst= 20;

1.4.4. Initialisierung einer Variablen

d.h. Einer Variablen bei der Definition gleich einen Wert zuweisen.

```
int anzahl= 50;
```

```
char ch= 'A';
```

```
char text[]= "Hallo, Welt!";
```

1.4.5. Variablen einen Wert zuweisen

Im Laufe des Programmes, können Variablen ihren Wert ändern:

```
anzahl= 17;
```

```
ch= 'B';
```

```
...
```

Achtung: Bevor man eine Variable verwenden kann, muss sie zuvor definiert werden.

1.4.6. Den Inhalt einer Variablen um 1 erhöhen

```
anzahl= anzahl + 1;
```

1.4.7. Den Inhalt von zwei Variablen austauschen

```
// Variablen definieren
```

```
int a;  
int b;  
int hilfe;  
  
//Variablen Werte zuweisen  
a=11;  
b=99;  
  
// Inhalte der Variablen austauschen  
hilfe= a;  
a= b;  
b= hilfe;
```

1.5. Befehle / Algorithmus

1.5.1. Algorithmus

Definition: Strukturierte Programmierung

Bei der **strukturierten Programmierung** werden **nur** Algorithmen mit

- ☑ **einem Eingang (=Start)** und
- ☑ **einem Ausgang (=Ende)** verwendet.

d.h. **Was im Programmtext untereinander steht wird auch nacheinander ausgeführt.**

Definition: Algorithmus

Ein Programm besteht aus einer **Folge von Befehlen**, die der Reihe nach von der CPU abgearbeitet werden. Man nennt diese Befehlsfolge auch **Algorithmus**.

Einen Algorithmus kann man sich wie eine Bedienungsanleitung vorstellen. Hier ein Beispiel:

1. Einsteigen;
2. Gurt anlegen;
3. Schlüssel ins Schloss;
4. **WENN** Handbremse los
 - 4.1. Handbremse anziehen;
5. Kupplung treten;
6. Schlüssel umdrehen;
7. Ersten Gang einlegen;
8. Rückspiegel schauen;
9. **SOLANGE** Fahrzeug kommt

- 9.1. Rückspiegel schauen;
10. Kupplung loslassen und leicht Gas geben;

1.6. Pseudocode und Struktogramm

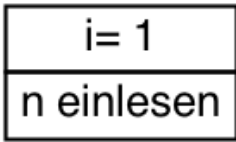
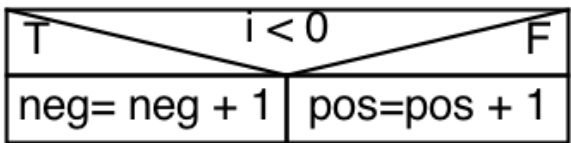
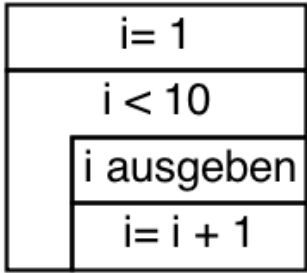
☑ Ein Algorithmus lässt sich durch die Kombination folg. Grundelementen festlegen:

- ☐ **Anweisung**
- ☐ **Verzweigung**
- ☐ **Wiederholung**

☑ Es gibt folgende Darstellungsmöglichkeiten für einen Algorithmus:

- ☐ Pseudocode, Programmablaufplan
- ☐ Struktogramm
- ☐ Flussdiagramm
- ☐ Programmcode

Wir wollen in der Folge Pseudocode und Struktogramme kennen lernen.

Pseudocode		Struktogramm
<pre>{ i= 1; n einlesen; }</pre>	Anweisung	
<pre>{ if (i < 0){ neg= neg + 1; } else { pos= pos + 1; } }</pre>	Verzweigung	
<pre>{ i= 1; while (i<= 10){ i ausgeben; i= i + 1; } }</pre>	Wiederholung	

1.6.1. Struktogrammeditor

- ☑ Online Struktogrammeditor: <http://www.learn2prog.de/>
- ☑ Nessi: Java Struktogrammeditor (Windows/Linux): <http://eii.ucv.cl/nessi/>
- ☑ EasyCode: <http://www.easycode.de/>
- ☑ Vips (Visuelles Programmieren mit Struktogrammen: Win/Lin): <http://partheil.com/vips/>
- ☑ MyFriend (C,Java, Reverse Engineering, Integration mit Compiler und Debugger, Frei für Schulen): <http://www.myfriend.de/>

1.7. Beispiele: Struktogramme

In den folg. Beispielen ist das Struktogramm gesucht.

1.7.1. Beispiel: flaeche

Lies laenge und breite ein und gib die Fläche aus.

1.7.2. Beispiel: tausche

Lies n und m ein und tausche deren Inhalt. Gib n und m aus.

1.7.3. Beispiel: maxi

Lies 2 Zahlen ein (x, y) und gib die grössere der beiden aus.

1.7.4. Beispiel: summe1

Lies n ein und gib die Summe der Zahlen von 1 bis n aus

1.7.5. Beispiel: summe2

Lies 2 Zahlen ein (start,ende).

Berechne die Summe der Zahlen von start bis ende.

Bsp:

start: 2 ende: 4

=> ergebnis: 9

1.7.6. Beispiel: pow

Lies m und n ein.

Gib m hoch n aus.

1.7.7. Beispiel: teiler1

Lies n ein und gib alle Teiler von n aus.

Hinweis: $n \% 2$ liefert 0 wenn n durch 2 teilbar ist

Hinweis: Zum Vergleichen verwende ==

1.7.8. Beispiel: teiler2

Lies 2 Zahlen ein (start,ende).

Gib von jeder Zahl zwischen start und ende die Teiler aus.

Bsp: start:3 ende:5

Teiler von 3

Teiler: 1

Teiler: 3

Teiler von 4

Teiler: 1

Teiler: 2

Teiler: 4

Teiler von 5

Teiler: 1

Teiler: 5

1.8. Beispiel: Pseudocode

1.8.1. Was gibt der folg. Algorithmus aus?

```
{
    f= 1;
    n= 5;

    i=1;
    while (i<=n){
        f=f*i;
        i=i+1; // oder auch das geht i++
    }

    f ausgeben;
}
```

oder statt der while()-Schleife die for()-Schleife

```
{
```



```
f= 1;
n= 5;

for (i=1; i<=n; i++) {
    f= f * i;
}

f ausgeben;
}
```

Antwort:
?????????

1.9. Übung: Struktogramme

Aufgabe: struktogramm

beantworten Sie die Übungen in
02-uebungen*

1.10. +Flussdiagramme

Algorithmen können auch mit sogenannten Flussdiagrammen dargestellt werden.

Kombination von nur drei Grundstrukturblöcken. 1. Anweisung/Sequenz 2. Verzweigung/Selektion 3. Wiederholung/ Iteration (<u>Schleife</u>)	<p>Das Diagramm zeigt drei Grundstrukturblöcke eines Struktogramms:</p> <ul style="list-style-type: none">Sequenz: Zwei gelbe Rechtecke, die nacheinander abgearbeitet werden.Selektion (Auswahl): Ein gelbes Rechteck, das in einen Entscheidungsbaum (Diamant) mündet, der zu zwei weiteren gelben Rechtecken führt.Iteration (Schleife): Ein gelbes Rechteck, das in einen Entscheidungsbaum (Diamant) mündet, der entweder zum Ende führt oder zurück zum Anfang des Blocks springt.
--	---

Weitere Informationen zu Flussdiagramme siehe:

<http://de.wikipedia.org/wiki/Programmablaufplan>

Aufgabe: Algorithmus und Flussdiagramm

Erstellen Sie ein Flussdiagramm für das Führen eines Telefongesprächs. Folgende Tätigkeiten und Abfragen:

1. Start
2. Hörer abnehmen
3. Wählen
4. Meldet sich der Teilnehmer?
Wenn ja, weiter mit „5. Gespräch führen“
Wenn „Nein“, Hörer auflegen (8)
9. prüfen, ob Gespräch aufschiebbar
Wenn ja, Arbeitsablauf beenden (7)
Wenn nein, Rücksprung zu Start
5. Gespräch führen
6. Hörer auflegen
7. Ende

