Ergänzungmodul Infomatik
Programmieren in C/C++

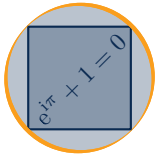
Vorbereitungskurs Ingenieurwissenschaften

Dipl.-Ing. Patrick Suwinski
patrick.suwinski@tu-dresden.de

5. September 2024

MINT-Kolleg

TECHNISCHE UNIVERSITÄT DRESDEN



Vorbereitungskurs
Ingenieurwissenschaften

Inhaltsverzeichnis i

Inhaltsverzeichnis ii

Inhaltsverzeichnis i

Inhaltsverzeichnis ii

Grundlegendes i

Programmieren

Programmieren ist eine Tätigkeit, bei der versucht wird durch systematischen Einsatz einer gewählten Programmiersprache ein vorliegendes Problem zu lösen.

- zur Realisierung wird eine allgemeingültige, eindeutige, detaillierte sowie formal ausführbare Vorschrift entworfen - ein Algorithmus
- jede Programmierung bedingt eines vorliegendes Algorithmus (Algorithmierungsschritt)
- jedes Programm ist lediglich ein Algorithmus, welcher in der jeweiligen Programmiersprache notiert/geschrieben wurde

Grundlegendes ii

- zur Erstellung eines Programmes sind entsprechende Hardware- und Softwarekomponenten notwendig
- der Quelltext des Programmes wird in einem Editor entworfen
- je nach Programmiersprache sorgen Compiler und/oder Interpreter für die Übersetzung des Quellcodes in Maschinencode

Achtung

Der Maschinencode ist von der verwendeten Hardware abhängig!
(Ausnahmen sind möglich, beispielsweise Java-Programme)

Algorithmus i

- muss sinnvoll dokumentiert werden (auch bei kleineren Programmen)
- ist Vorschrift zur Lösung einer Klasse von Problemen
- besteht aus einer endlichen Anzahl an Schritten, mit denen aus bekannten Eingangsdaten zunächste unbekannte Ausgangsdaten berechnet werden können

Algorithmus ii

- Allgemeingültigkeit (Lösung für Klasse von Problemen)
- Endlichkeit (endlich beschreibbar)
- Eindeutigkeit (Abfolge der Schritte unmissverständlich beschrieben)
- Ausführbarkeit (jeder schritt tatsächlich ausführbar)
- Terminiertheit (endlich viele Schritte)
- Determiniertheit (gleiche Eingangsbedingungen führen immer zum gleichen Ergebnis)
- Effektivität (Ausführbarkeit von einerrealen Maschine)
- Effizienz (geringe Auslastung des Arbeitsspeichers und geringe Rechenzeit)

Algorithmus iii

- Beschreibung von Algorithmen verbal und graphisch möglich
- verbal in Form von Pseudocode und Umgangssprache
- graphisch in Form von Flussdiagrammen, Programmablaufplänen und Struktogrammen

Struktogramm

- rechteckige Außengeometrie
- Zerlegung des Algorithmus in Strukturblöcke
- Art der Strukturblöcke werden hinreichend zerlegt

  $\rightarrow$ Programmfunktion unmissverständlich
- Schachtelung der Strukturblöcke möglich
- für verschiedene Programmiersprachen existieren verschiedene Funktionen als Strukturblöcke

Compiler

- wandelt Quelltext in Maschinencode um und dieser gibt Instruktionen für Prozessor
- Compiler führt Übersetzung in zwei Schritten durch:
  1. Quelltext wird in Objektcode übersetzt
  2. Objektcode und weitere Objektcodebestandteile werden vom Linker zu ausführbaren Programm verbunden
- Programm ist auf der verwendeten Hardware wiederholt nutzbar

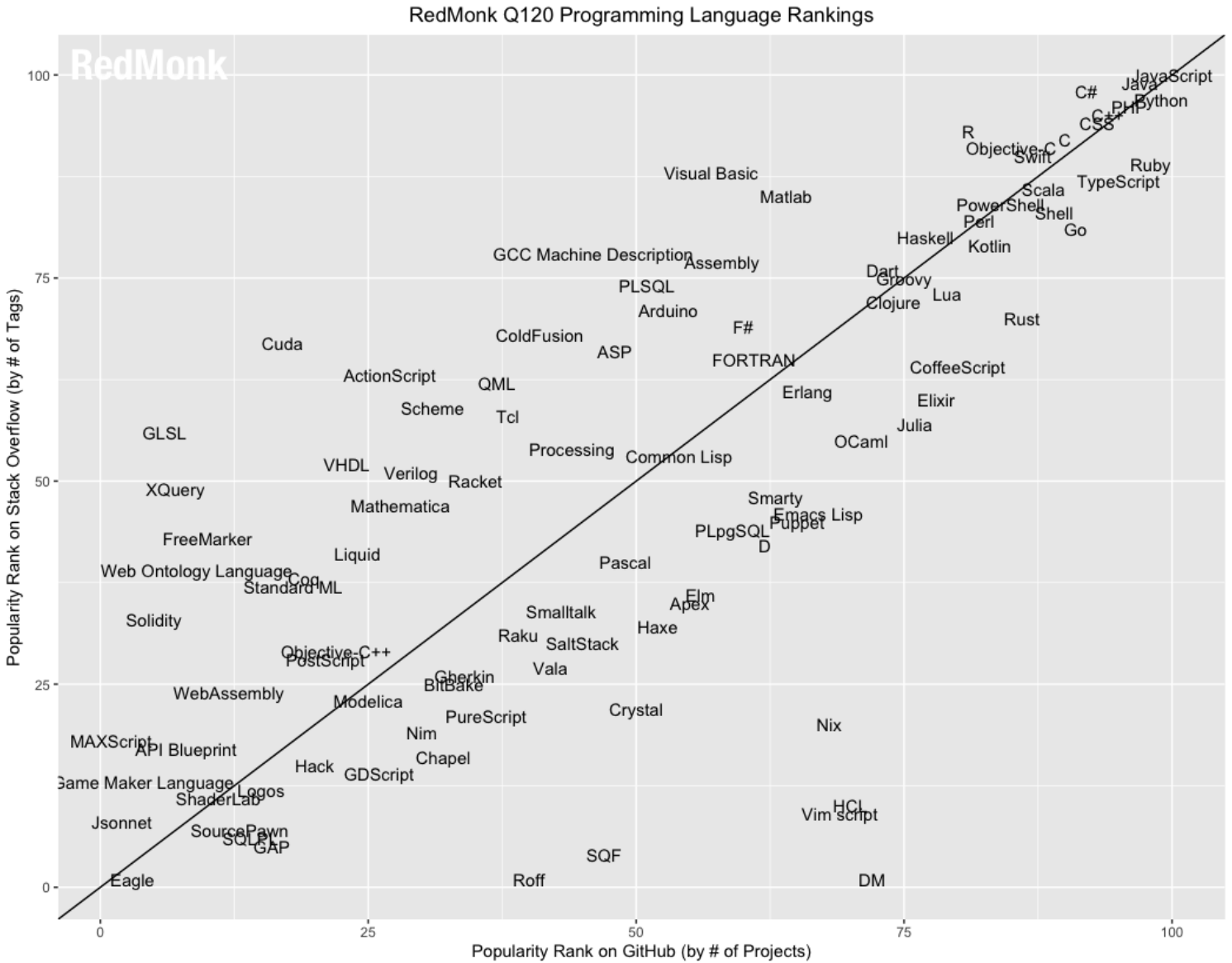
Interpreter

- wandelt Programmtext während Abarbeitung in ausführbare Instruktionen um
- muss bei jeder Programmausführung neu durchgeführt werden

Inhaltsverzeichnis i

Inhaltsverzeichnis ii

Entwicklung der Programmiersprachen i



[1]

[1] https://redmonk.com/sogrady/files/2020/02/lang.rank_.120.wm_.png

Entwicklung der Programmiersprachen ii

Programming Language	2020	2015	2010	2005	2000	1995	1990	1985
Java	1	2	1	2	3	-	-	-
C	2	1	2	1	1	2	1	1
Python	3	7	6	6	23	21	-	-
C++	4	4	4	3	2	1	3	9
C#	5	5	5	7	9	-	-	-
JavaScript	6	8	8	8	6	-	-	-
PHP	7	6	3	4	25	-	-	-
SQL	8	-	-	96	-	-	-	-
Swift	9	181	-	-	-	-	-	-
Ruby	10	11	9	23	31	-	-	-
Objective-C	12	3	15	37	-	-	-	-
Lisp	27	21	14	13	8	5	6	2
Fortran	29	28	21	14	16	4	2	12
Ada	34	26	24	15	15	6	7	3
Pascal	235	13	12	53	13	3	8	5

2

[2] https://www.tiobe.com/tiobe-index//?201201

Hardwarenahes Programmieren

Vorteile

- schnelle Abarbeitung der Programmschritte
- detailliertes Wissen über Hintergrundprozesse

Nachteile

- detailliertes Wissen über Hardware erforderlich
- aufwändige Entwicklung
- Code wird schnell sehr unübersichtlich

Hochsprachen

Vorteile

- kein detailliertes Wissen um verwendete Hardware erforderlich
- komplexe Aufgaben mit geringerem Umfang realisierbar

Nachteile

- ineffizienter Code (hohe Zykluszeiten, Arbeitsspeicherauslastung)
- Compiler sind wesentlich komplexer aufgebaut als Assembler, wodurch Fehler beim Compilieren entstehen können

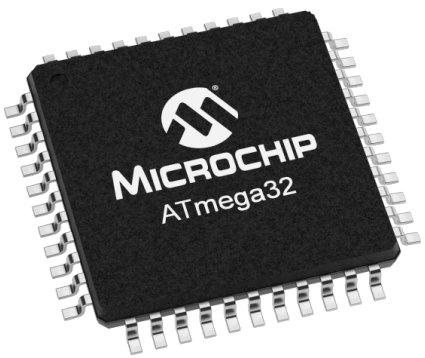
Häufig verwendete Prozessoren

Assembler

- ATmega32 (Microchip AVR, vormals Atmel AVR)
- HC-11 (Motorola)

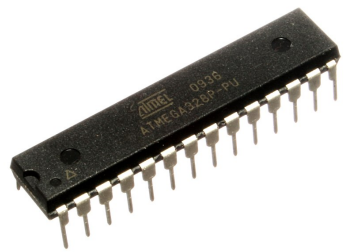
C und C++

- ATmega32 (Microchip AVR, vormals Atmel AVR)
- Intel-, AMD- und ARM-Prozessoren



[3]

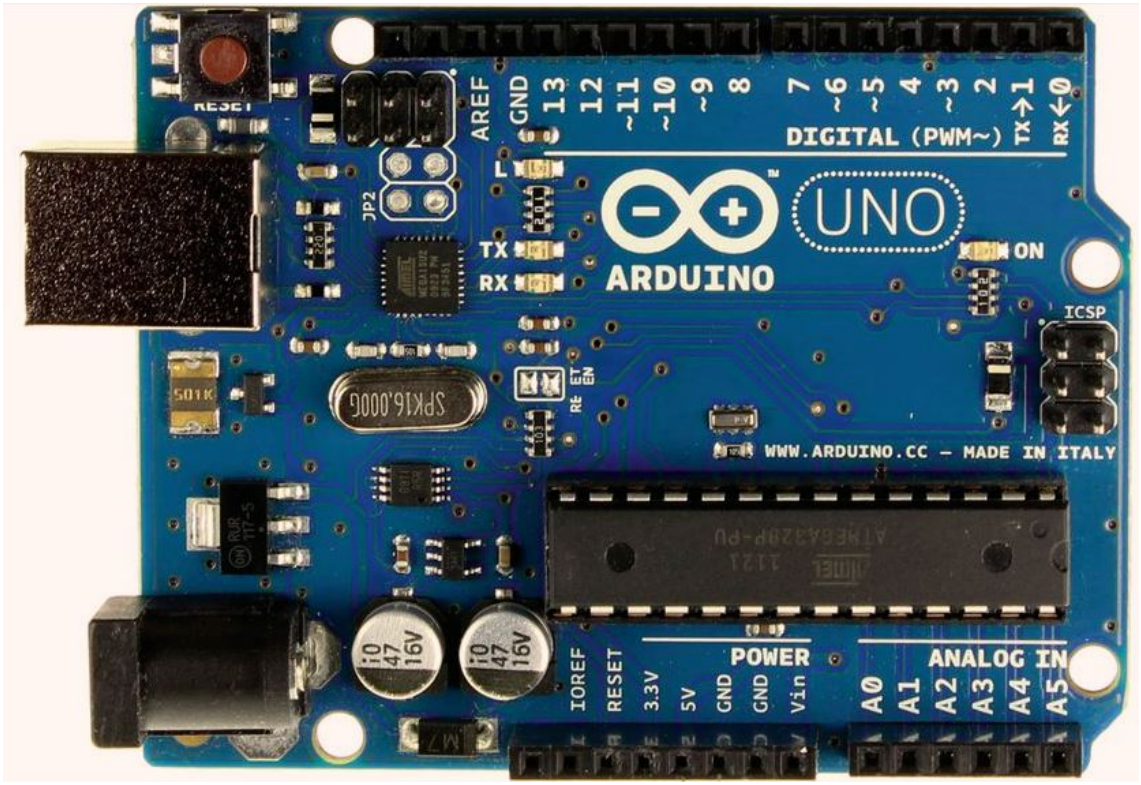
[4]



[3] https://www.microchip.com/_images/products/medium/37c714348bfcf742067533f518122fc9.png

[4] https://hackaday.com/wp-content/uploads/2016/10/atmelmicrochip.png?w=800

Arduino Uno

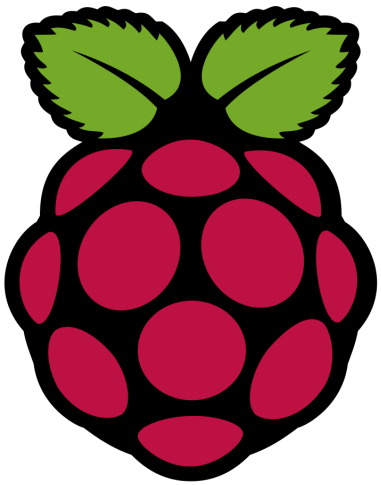


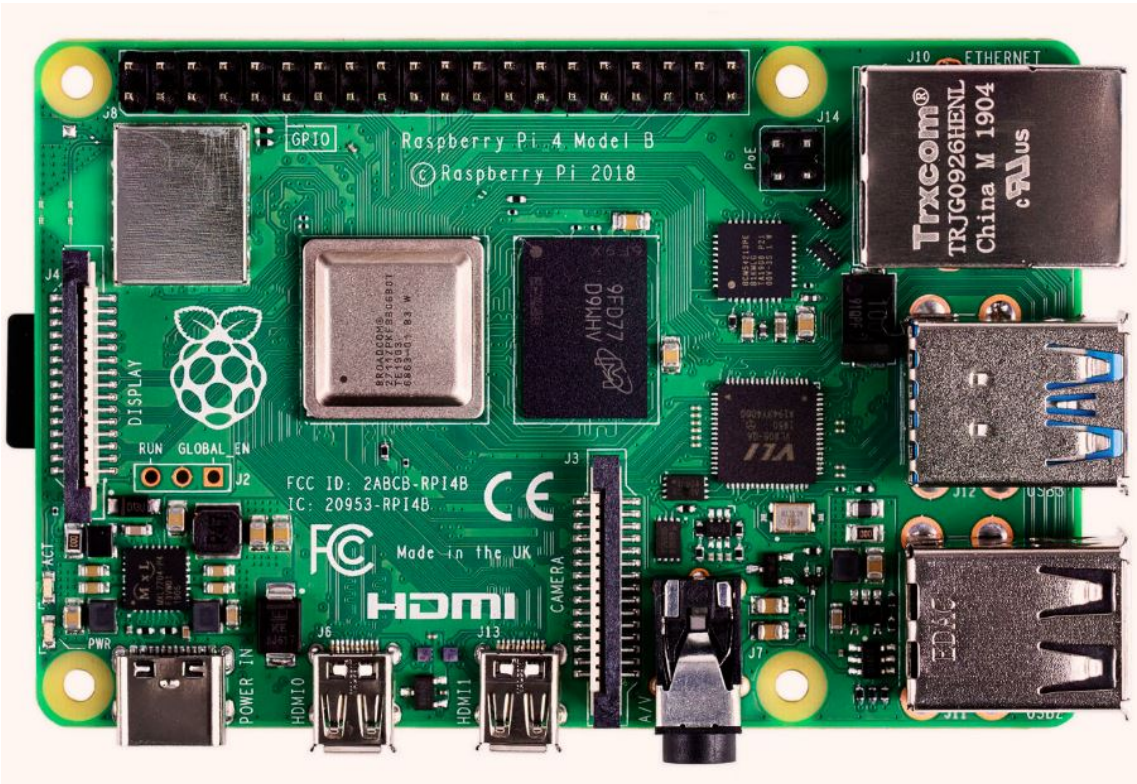
[5]

Abbildung 1: Arduino Logo und Arduino Uno mit ATmega32

[5] https://www.distrelec.de/Web/WebShopImages/landscape_large/9-/01/arduino-a000066.jpg

RaspberryPi





6

Abbildung 2: RaspberryPi Logo und RaspberryPi 4 mit ARM-CPU

[6] https://media.nbb-cdn.de/images/products/490000/492584/RaspberryPi4_PB_01.jpg

Inhaltsverzeichnis i

Inhaltsverzeichnis ii

Grundlegende Datentypen

Datentyp	Potenz	Dezimalzahl
Bit	2^1	2
Nibble	2^4	16
Byte	2^8	256
Word	2^{16}	65.536
Doubleword	2^{32}	4.294.967.296

Datentypen in C/C++

Datentypen in C/C++

bool	2^1	
int	2^{16}	= 65.536 Werte
float	2^{32}	
double	2^{64}	
char	2^8	= 256 Werte
String		

Qualifizierer

- signed, unsigned
- short, long, long long

Deklaration von Datentypen

Die Deklarationen von Datentypen erfolgen im Arbeitsspeicher.

Festspeicher	ROM = Read Only Memory
Arbeitsspeicher	RAM = Random Access Memory

Achtung

Reservierungen im Arbeitsspeicher haben keine definierten Default-Werte der Variablen zur Folge.

Aufbau einer Ganzzahl

Vorzeichen	Datentyp	Bitanzahl
unsigned	int nZahl	16 Bit = 65.536 mögliche Werte
signed	int nZahl	16 Bit = 65.536 mögliche Werte
unsigned	short int nZahl	16 Bit = 65.536 mögliche Werte
unsigned	long int nZahl	32 Bit = 4.294.967.296 mögliche Werte
unsigned	long long int nZahl	64 Bit $\approx 18 * 10^{18}$ mögliche Werte

Achtung

Der Wertebereich der vorzeichenbehafteten Zahlen ist teilt sich sowohl in positive wie auch in negative Richtung auf. Das heißt bei einer 16 Bit Ganzzahl mit Vorzeichen sind Werte von -32.786 bis 32.787 möglich.

Beispiel -356 und 42

Aufbau einer Gleitkommazahl

Gleitkommazahl

$$x = p \cdot 10^e$$

mit x = Zahlenwert
p = Mantisse
e = Exponent

Datentyp	Potenz	Mantisse	Exponent
float	2^{32}	p = 23 Bit	e = 8 Bit
double	2^{64}	p = 52 Bit	e = 11 Bit

Beispiele 3.1415 und 1e+009

Zahlenformate i

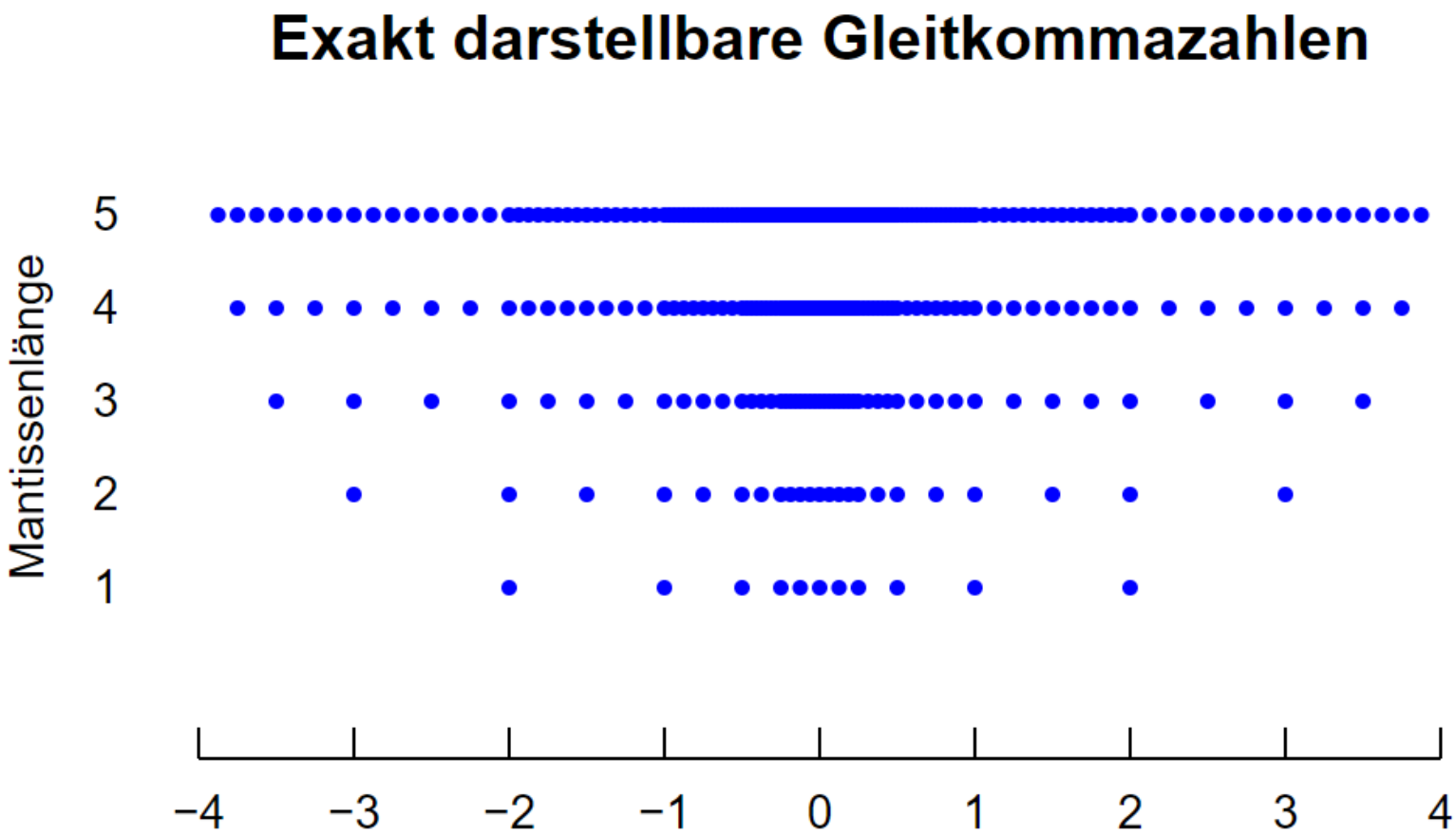
Anwendung von Ganzzahlformaten

- ganze Zahlen, Zählvariablen (Schleifen)
- Abstand zwischen Darstellbaren Zahlenwerten ist konstant

Anwendung von Gleitkommazahlformaten

- rationale Zahlen, extrem kleine Zahlen, extrem große Zahlen
- Abstand zwischen Darstellbaren Zahlenwerten ist **nicht** konstant

Zahlenformate ii



7

[7] https://upload.wikimedia.org/wikipedia/commons/8/8f/Exakt_darstellbare_Gleitkommazahlen.png

Doch wo liegt der Nutzen?

Informatik
- komplexe Modellierungen und Entwurf von Algorithmen und Datenstrukturen

Industrie
- Auswertung von Messwerten in ausreichender Genauigkeit
- Ausgabe von Analogwerten mit ausreichender Genauigkeit

Uni
- Approximation und nummerische Berechnungen von Zuständen, Situationen und Simulationen

Inhaltsverzeichnis i

Inhaltsverzeichnis ii

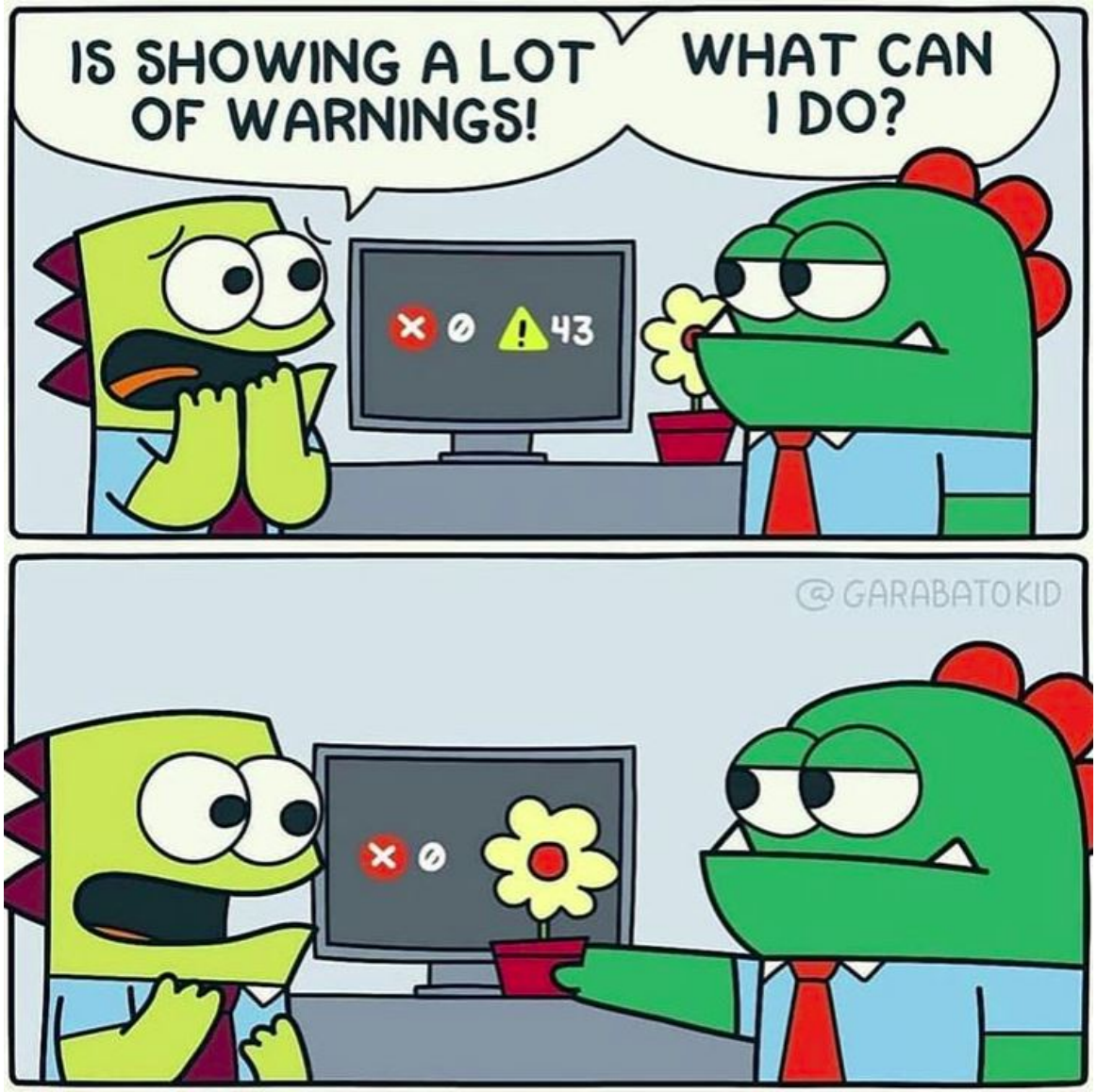
Erste Schritte in C

- Programmierung zunächst in C mit CPP-Sythax

- Programmierumgebung ist eine IDE (Integrated Development Environment)
- wir nutzen IDE Code::Blocks
- Alternativen sind Orwell Dev-C++, Eclipse, Microsoft Visual Studio

- IDE ist eine Integrierte *Entwicklungsumgebung*
- bietet Autovervollständigungen und/oder farbige Kennzeichnungen an
- inkludiert verschiedene Compiler
- generiert Fehler, Meldungen und Hinweise

Fehler und Warnungen



[8]

[8] https://www.instagram.com/p/B6n6sIdHVvD/

Programmbeispiel 4-1 Hello-World.cpp

Beispiel

```
1#include <iostream>      //Bibliothek zur Ein- und Ausgabe durch Konsole
2#include <cstdlib>       //Bibliothek zur Nutzung des system-Befehls
3
4
5using namespace std;     //Nutzung des entsprechenden Namensraumes
6
7main()                   //Hauptmethode, Aufruf nach Programmstart
8
9{
10
11    cout<<"Hello World"<<endl;  //Ausgabe Zeichenkette an Bildschirm
12
13    system("pause");            //Konsolenfenster wird nach Druecken
      beliebiger Taste geschlossen
14}
```

Ausgabe von Werten über Konsole

1. Datentypen deklarieren
2. Werte für Variablen festlegen
3. Ausgabe der Werte über Konsole

Programmbeispiel 4-2 Ausgabe-Variablen.cpp

Beispiel

```
1  #include <iostream>      //Bibliothek zur Ein- und Ausgabe durch Konsole
2  #include <cstdlib>       //Bibliothek zur Nutzung des system-Befehls
3
4  using namespace std;     //Nutzung des entsprechenden Namensraumes
5
6  main()                   //Hauptmethode, Aufruf nach Programmstart
7  {
8      int x;               //Definition der Variablen
9      int y;
10     int z;
11
12     x = 10;
13     y = 20000;
14     z = -3456;
15
16     cout<<x;         //Ausgabe der Variablen
17     cout<<y;
18     cout<<z;
19
20     system("pause");
21 }
```

Eingabe und Ausgabe von Werten über Konsole i

- Datentypen deklarieren
- Werte für Variablen einlesen
- Ausgabe der Werte über Konsole

Programmbeispiel 4-3 Ein-Ausgabe-Variablen.cpp

Beispiel

```
1#include <iostream>      //Bibliothek zur Ein- und Ausgabe
2#include <cstdlib>       //Bibliothek zur Nutzung des system-Befehls
3
4using namespace std;     //Nutzung des entsprechenden Namensraumes
5
6main()                   //Hauptmethode, Aufruf nach Programmstart
7{
8    int x,y,z;          //Definition der Variablen
9
10   cin>>x;
11   cin>>y;
12   cin>>z;
13
14   cout<<x<<endl;      //Ausgabe der Variablen
15   cout<<y<<endl;
16   cout<<z<<endl;
17
18   system("pause");
19}
```

Eingabe und Ausgabe von Werten über Konsole ii

- Programme ohne Handlungsaufforderungen des Nutzers sind schwer bedienbar
- daher zusätzliche Aufforderung an den Nutzer implementieren

Programmbeispiel 4-4 Ein-Ausgabe-Variablen2.cpp

Beispiel

```
1#include <iostream>      //Bibliothek zur Ein- und Ausgabe
2#include <cstdlib>       //Bibliothek zur Nutzung des system-Befehls
3using namespace std;     //Nutzung des entsprechenden Namensraumes
4
5main()                   //Hauptmethode, Aufruf nach Programmstart
6{
7    int x,y,z;          //Definition der Variablen
8
9    cout<<"Bitte geben Sie einen Wert fuer x ein: "<<endl;
10                       //Nutzeraufforderung zur Eingabe von Werten
11   cin>>x;
12   cout<<"Bitte geben Sie einen Wert fuer y ein: "<<endl;
13   cin>>y;
14   cout<<"Bitte geben Sie einen Wert fuer z ein: "<<endl;
15   cin>>z;
16   cout<<"Der Wert von x betraegt "<<x<<endl;
17                       //Ausgabe der Variablen
18   cout<<"Der Wert von y betraegt "<<y<<endl;
19   cout<<"Der Wert von z betraegt "<<z<<endl;
20   system("pause");
21}
```

Zusammenfassung

- Sie können jetzt kleine Programme zur Ein- und Ausgabe über Konsole selbst verfassen
- Datentypen müssen je nach Zahlenformat angepasst werden

Inhaltsverzeichnis i

Inhaltsverzeichnis ii

Arithmetische Operationen

Syntax für arithmeische Operationen

Arithmetische Operationen sind immer in der Form

$$\text{Ergebnis} = \text{Zahl}_1 \square \text{Zahl}_2$$

zu verfassen.

Arithmetische Operatoren

- Addition +
- Subtraktion -
- Multiplikation *
- Division /

Achtung

Datentypen der Zahlen und des Ergebnisses müssen kompatibel sein.

Programmbeispiel 5-1 Addition-Subtraktion.cpp

Beispiel

```
1  #include <iostream>      //Bibliothek zur Ein- und Ausgabe
2  #include <cstdlib>       //Bibliothek zur Nutzung des system-Befehls
3  #include <cmath>         //Bibliothek zur Nutzung arithmetischer Funktionen
4
5  using namespace std;     //Nutzung des entsprechenden Namensraumes
6
7  main()                   //Hauptmethode, Aufruf nach Programmstart
8  {
9      int nZahl1, nZahl2, nErgebnis_addition, nErgebnis_subtraktion;
10     int nErgebnis_multiplikation, nErgebnis_division;
11     float fZahl1, fZahl2, fErgebnis_addition, fErgebnis_subtraktion;
12     float fErgebnis_multiplikation, fErgebnis_division;
13
14     cout<<"Bitte Wert fuer nZahl1 eingeben: ";
15     cin>>nZahl1;
16     cout<<"Bitte Wert fuer nZahl2 eingeben: ";
17     cin>>nZahl2;
18     cout<<"Bitte Wert fuer fZahl1 eingeben: ";
19     cin>>fZahl1;
20     cout<<"Bitte Wert fuer fZahl2 eingeben: ";
21     cin>>fZahl2;
```

Programmbeispiel 5-1 Addition-Subtraktion.cpp

```
1      nErgebnis_addition=nZahl1+nZahl2;
2      nErgebnis_subtraktion=nZahl1-nZahl2;
3      nErgebnis_multiplikation=nZahl1*nZahl2;
4      nErgebnis_division=nZahl1/nZahl2;
5
6      fErgebnis_addition=fZahl1+fZahl2;
7      fErgebnis_subtraktion=fZahl1-fZahl2;
8      fErgebnis_multiplikation=fZahl1*fZahl2;
9      fErgebnis_division=fZahl1/fZahl2;
```

Programmbeispiel 5-1 Addition-Subtraktion.cpp

```
1     cout<<"Der Wert fuer die Addition der int-Zahlen betraegt: "<<
      nErgebnis_addition<<endl;
2     cout<<"Der Wert fuer die Subtraktion der int-Zahlen betraegt: "<<
      nErgebnis_subtraktion<<endl;
3     cout<<"Der Wert fuer die Multiplikation der int-Zahlen betraegt: "<<
      nErgebnis_multiplikation<<endl;
4     cout<<"Der Wert fuer die Division der int-Zahlen betraegt: "<<
      nErgebnis_division<<endl;
5     cout<<"Der Wert fuer die Addition der float-Zahlen betraegt: "<<
      fErgebnis_addition<<endl;
6     cout<<"Der Wert fuer die Subtraktion der float-Zahlen betraegt: "<<
      fErgebnis_subtraktion<<endl;
7     cout<<"Der Wert fuer die Miltiplikation der float-Zahlen betraegt: "<<
      fErgebnis_multiplikation<<endl;
8     cout<<"Der Wert fuer die Division der float-Zahlen betraegt: "<<
      fErgebnis_division<<endl;
9
10    system("pause");
11}
```

Modulo-Operator

%

Der Modulo-Operator ist eine besondere Form des Divisionsoperators.

Er gibt den Rest einer Division zweier int-Zahlen an.

Programmbeispiel 5-2 Modulo.cpp

Beispiel

```
1#include <iostream>      //Bibliothek zur Ein- und Ausgabe
2#include <cstdlib>       //Bibliothek zur Nutzung des system-Befehls
3#include <cmath>         //Bibliothek zur Nutzung arithmetischer Funktionen
4
5using namespace std;     //Nutzung des entsprechenden Namensraumes
6
7main()                   //Hauptmethode, Aufruf nach Programmstart
8{
9    int nDividend, nDivisor, nRest; //Definition der Variablen
10   int nQuotient;
11
12   cout<<"Bitte geben Sie einen ganzahligen Wert fuer den Dividenden an: ";
13   cin>>nDividend;
14   cout<<"Bitte geben Sie einen ganzahligen Wert fuer den Divisor an: ";
15   cin>>nDivisor;
16
17   nQuotient=nDividend/nDivisor;    //Berechnung Quotienten
18   nRest=nDividend%nDivisor;        //Berechnung Rest der Division
```

Programmbeispiel 5-2 Modulo.cpp

```
1      cout<<"\nDer Quotient betraegt: "<<nQuotient<<"."<<endl;
2                    //Befehl \n in Gaensefuesschen bewirkt Zeilenumbruch
3      cout<<"Der Rest der Division betraegt: "<<nRest<<endl;
4
5      system("pause");
6  }
```

Potenzen

$()^x$

- Potenzieren als Funktion nicht ohne Weiteres möglich
- möglich durch Multiplikation der Basis mit sich selbst in einer Schleife
- Nutzung der pow-Funktion
- pow-Funktion in der Form Potenz=pow(Basis,Exponent)
- im Zweifelsfall Datentypen für verwendete Zahlen als float wählen

Programmbeispiel 5-3 Potenz-pow.cpp

Beispiel

```
1#include <iostream>      //Bibliothek zur Ein- und Ausgabe
2#include <cstdlib>       //Bibliothek zur Nutzung des system-Befehls
3#include <cmath>         //Bibliothek zur Nutzung arithmetischer Funktionen
4
5using namespace std;     //Nutzung des entsprechenden Namensraumes
6
7main()                   //Hauptmethode, Aufruf nach Porgrammstart
8{
9    float fBasis, fExponent, fErgebnis;     //Definition der Variablen
10
11   cout<<"Bitte geben Sie einen beliebigen Wert fuer die Basis an: ";
12   cin>>fBasis;
13   cout<<"Bitte geben Sie einen beliebigen Wert fuer den Exponenten an: ";
14   cin>>fExponent;
15
16   fErgebnis=pow(fBasis,fExponent);        //Berechnung der Potenz
17
18   cout<<"\nDas Ergebnis der Potenz betraegt: "<<fErgebnis<<"."<<endl;
19                   //Befehl \n in Gaensefuesschen bewirkt Zeilenumbruch
20
21   system("pause");
22}
```

Wurzeln

$$\sqrt{x}$$

- Ziehen einer Wurzel über sqrt-Funktion möglich
- sqrt-Funktion in der Form Wurzel = sqrt(Zahl)
- im Zweifelsfall Datentypen für verwendete Zahlen als float wählen

Programmbeispiel 5-4 Wurzel.cpp

Beispiel

```
1#include <iostream>      //Bibliothek zur Ein- und Ausgabe
2#include <cstdlib>       //Bibliothek zur Nutzung des system-Befehls
3#include <cmath>         //Bibliothek zur Nutzung arithmetischer Funktionen
4#include <math.h>
5
6using namespace std;     //Nutzung des entsprechenden Namensraumes
7
8main()                   //Hauptmethode, Aufruf nach Porgrammstart
9{
10    float fZahl, fErgebnis;      //Definition der Variablen
11
12    cout<<M_PI<<endl;
13    cout<<"Bitte geben Sie einen beliebigen Wert fuer die Zahl an: ";
14    cin>>fZahl;
15
16    fErgebnis=sqrt(fZahl);       //Berechnung der Potenz
17
18    cout<<"\nDas Ergebnis der Wurzel betraegt: "<<fErgebnis<<"."<<endl;
19                         //Befehl \n in Gaensefuesschen bewirkt Zeilenumbruch
20
21    system("pause");
22}
```

Besonderheiten

- trigonometrische Funktionen und Kosntanten können Bibliotheken entnommen werden
- für Aufsummation von Werten kann anstelle von `i=i+1` Sonderbefehl `i++` verwendet werden

Wichtig

Der Sonderbefehl `i++` ist insbesondere für Laufbedingungen von Schleifen besonders geeignet.

Zusammenfassung

- Sie beherrschen nun die Fähigkeiten arithmetische Berechnungen selbstständig durchzuführen
- Sie verfügen über Kenntnisse des Aufrufes einfacher Methoden aus Bibliotheken [`pow()`, `sqrt()`]
- Sonderbefehl für Inkrementierungen `i++` ist besonders für Laufbedingungen von Schleifen geeignet

Inhaltsverzeichnis i

Inhaltsverzeichnis ii

Nutzen

- Vergleichen von zwei Zahlenwerten mit dem gleichen Datentyp
- Ausgabe des Vergleiches als BOOL

Beziehung	**Operator**	**Beispiel**
kleiner	<	x < y
größer	>	x > y
kleiner/gleich	<=	x <= y
größer/gleich	>=	x >= y
gleich	= =	x == y
ungleich	! =	x ! = y

Programmbeispiel 6-1 Vergleiche.cpp

Beispiel

```
1#include <iostream>
2#include <cstdlib>
3#include <cmath>
4
5using namespace std;
6
7main()
8{
9     int nZahl1,nZahl2;
10    bool bGroesser,bGroesserGleich,bKleiner,bKleinerGleich,bGleich,bUngleich;
11
12    cout<<"Bitte geben Sie einen ganzahligen Wert fuer die Zahl1 an: ";
13    cin>>nZahl1;
14    cout<<"Bitte geben Sie einen ganzahligen Wert fuer die Zahl2 an: ";
15    cin>>nZahl2;
```

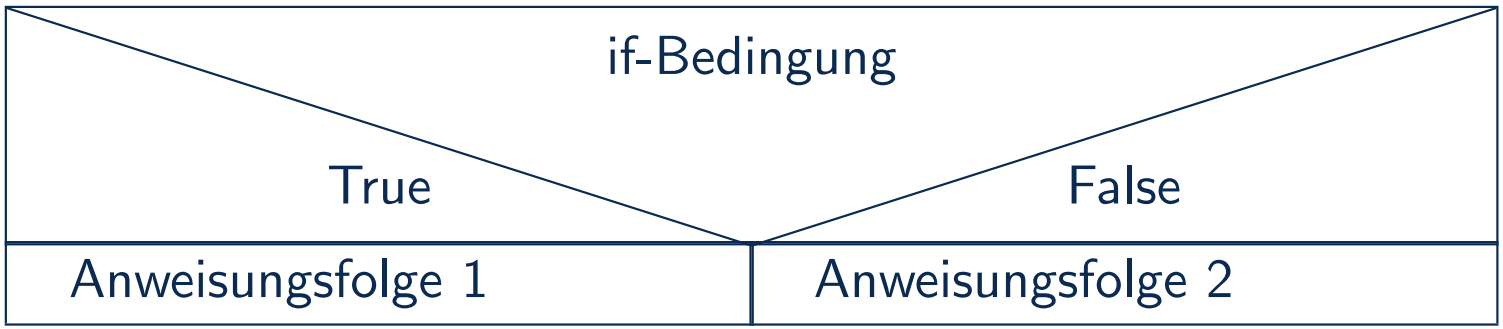
Programmbeispiel 6-1 Vergleiche.cpp

```
1     bGroesser=nZahl1>nZahl2;                    //Groesser
2     bGroesserGleich=nZahl1>=nZahl2;             //Groesser-Gleich
3     bKleiner=nZahl1<nZahl2;                     //Kleiner
4     bKleinerGleich=nZahl1<=nZahl2;              //Kleiner-Gleich
5     bGleich=nZahl1==nZahl2;                     //Gleich
6     bUngleich=nZahl1!=nZahl2;                   //Ungleich
7
8     cout<<"\nDie Zahl1 ist Groesser als Zahl2: "<<bGroesser<<endl;
9     cout<<"Die Zahl1 ist Groesser-Gleich als Zahl2: "<<bGroesserGleich<<endl;
10    cout<<"Die Zahl1 ist Kleiner als Zahl2: "<<bKleiner<<endl;
11    cout<<"Die Zahl1 ist Kleiner-Gleich als Zahl2: "<<bKleinerGleich<<endl;
12    cout<<"Die Zahl1 ist Gleich als Zahl2: "<<bGleich<<endl;
13    cout<<"Die Zahl1 ist Ungleich als Zahl2: "<<bUngleich<<endl<<endl;
14    system("pause");
15}
```

if-Verzweigung

- Verzweigungsstrukturen im Programm möglich
- verschiedene Reaktionen auf unterschiedliche Eingaben möglich
- bedingte Verzweigung anhand eines Vergleiches beziehungswiese bool-Wertes
- zur Abarbeitung muss Wert in Funktion if(Wert) TRUE sein

Struktogramm



Programmbeispiel 6-2 if-else.cpp

Beispiel

```
1#include <iostream>
2#include <cstdlib>
3#include <cmath>
4
5using namespace std;
6
7main()
8{
9     int nZahl1,nZahl2;
10    bool bGroesser,bKleiner;
11
12    cout<<"Bitte geben Sie einen ganzahligen Wert fuer die Zahl1 an: ";
13    cin>>nZahl1;
14    cout<<"Bitte geben Sie einen ganzahligen Wert fuer die Zahl2 an: ";
15    cin>>nZahl2;
```

Programmbeispiel 6-2 if-else.cpp

```
1      if(nZahl1>nZahl2)                                    //Groesser
2          cout<<"Die Zahl1 ist Groesser als Zahl2."<<endl;
3
4      if(nZahl1<nZahl2)                                    //Kleiner
5          cout<<"Die Zahl1 ist Kleiner als Zahl2."<<endl;
6
7      system("pause");
8  }
```

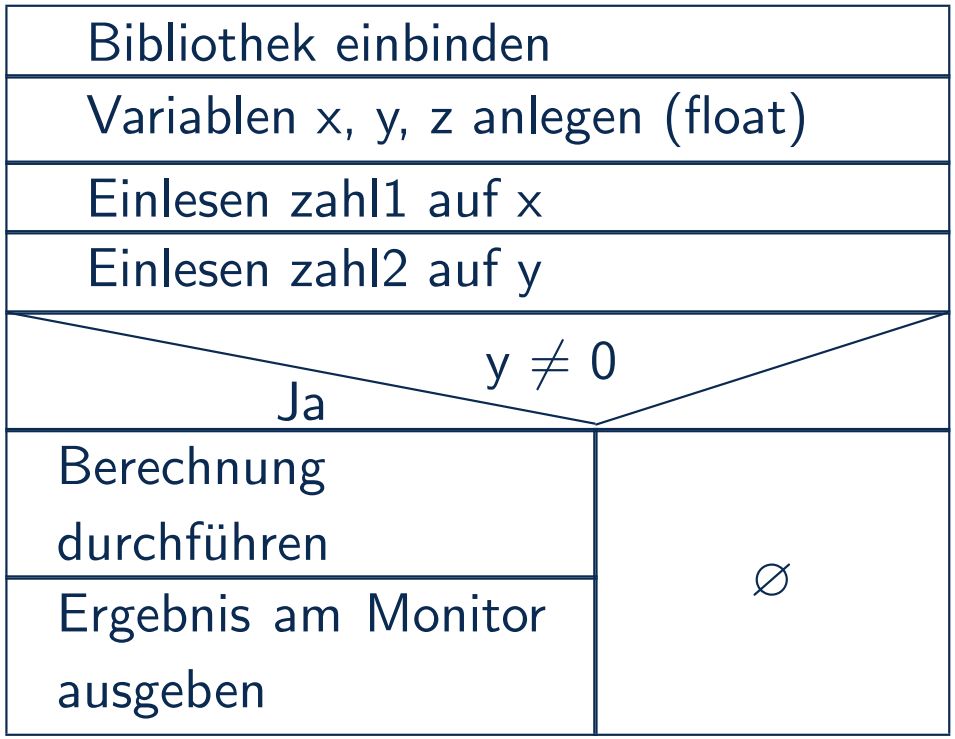
Vorlesungsübung 1 - Aufgabe

Es ist ein Programm zu entwickeln, das zwei reelle Zahlen über die Tastatur einliest und anschließend diese beiden Zahlen durch einander dividiert. Das Ergebnis ist am Monitor auszugeben. Dabei muss beachtet werden, dass der Divisor von Null verschieden ist. In diesem besonderen Fall wird die Berechnung abgebrochen.

1. Erstellen Sie zunächst ein Struktogramm, welches die Anforderungen mittels einer if-Bedingung erfüllt.
2. Entwerfen Sie anschließend ein lauffähiges C/C++-Programm, welches die Anforderungen erfüllt.

Vorlesungsübung 1 - Struktogramm

Struktogramm



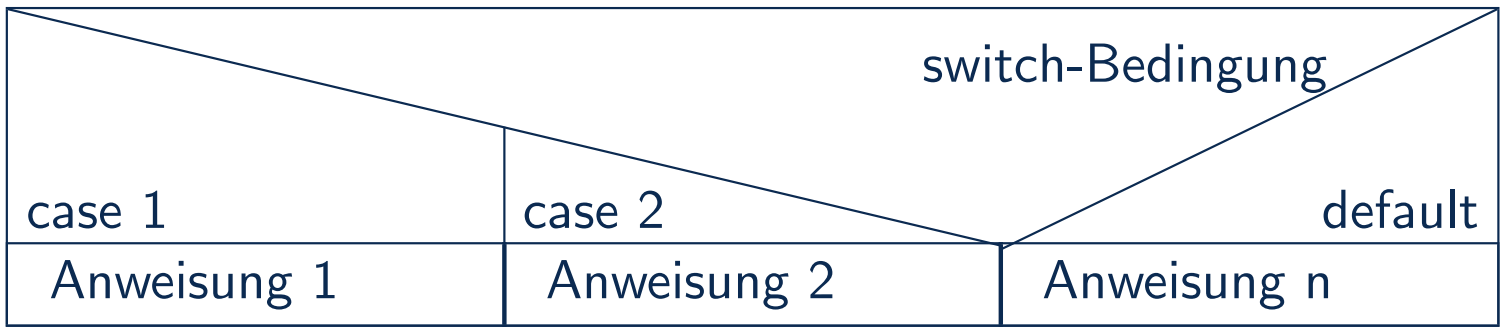
Vorlesungsübung 1 - Lösung

```
#include <iostream>
#include <cstdlib>
using namespace std;
main()
{
float x=0.0, y=0.0, z=0.0;
cout << "\nEingabe Zahl 1: ";
cin >> x;
cout << "\nEingabe Zahl 2: ";
cin >> y;
//if(y!=0)
//{
z=x/y;
cout << "\n" << z;
//}
cout << "\n\n";
system("pause");
}
```

switch-case

- unter Umständen übersichtlicher
- Compiler kann bei größeren switch-case Konstrukten optimieren
- Funktion switch(Wert) wird implementiert
- Vergleich erfolgt im case
- beenden einer case-Anweisung immer mit break

Struktogramm switch-case



Programmbeispiel 6-3 switch-case.cpp

Beispiel

```
1#include <iostream>
2#include <cstdlib>
3#include <cmath>
4using namespace std;
5main()
6{
7     int nZahl;
8     cout<<"Bitte geben Sie einen ganzahligen Wert fuer die Zahl1 an: ";
9     cin>>nZahl;
10    switch(nZahl){
11    case 0:      //Bedingung fuer diesen Fall
12         cout<<"\nDie Zahl ist Gleich 0."<<endl;
13         break;
14    case 100:    //Bedingung fuer diesen Fall
15         cout<<"\nDie Zahl ist Gleich 100."<<endl;
16         break;
17    default:     //Ausfuehrung, wenn kein anderer Fall erfuellt ist
18         cout<<"\nDie Zahl betraegt weder 0 noch 100."<<endl;
19         break;
20    }
21    system("pause");
22}
```

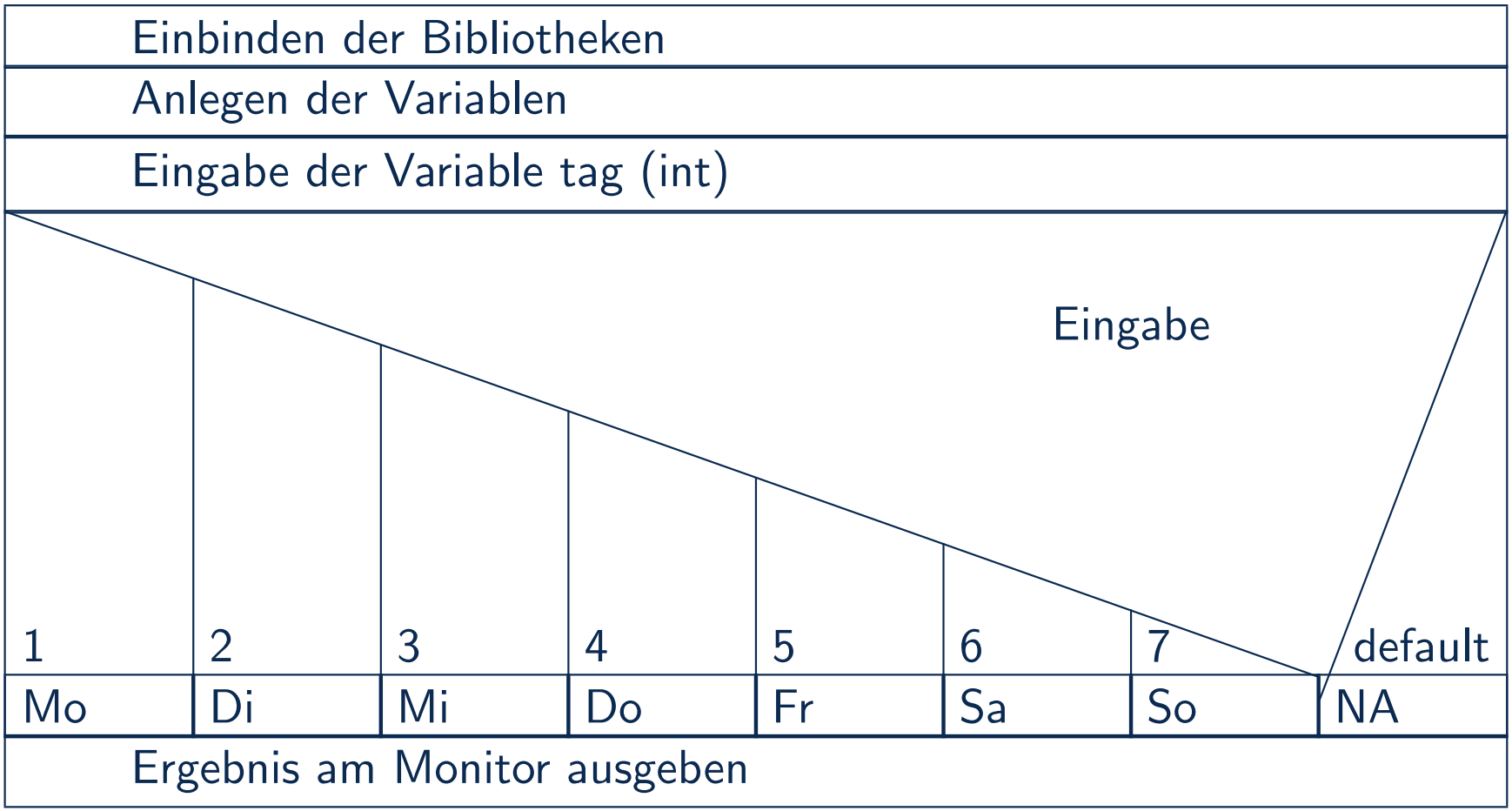
Vorlesungsübung 2 - Aufgabe

Es ist ein Programm zu erstellen, welches den Zahlen von 1 bis 7 den entsprechenden Wochentag (Montag bis Sonntag) zuordnet. Die Eingabe der Zahlen soll über die Tastatur und die Ausgabe des Tages über den Monitor erfolgen. Nutzen Sie zur Realisierung eine switch-case-Anweisung.

1. Erstellen Sie zunächst ein Struktogramm, welches die Anforderungen mittels einer if-Bedingung erfüllt.
2. Entwerfen Sie anschließend ein lauffähiges C/C++-Programm, welches die Anforderungen erfüllt.

Vorlesungsübung 2 - Struktogramm

Struktogramm



Vorlesungsübung 1 - Lösung

```
#include <iostream>
#include <cstdlib>
using namespace std;
main(){
int tag;
cout<<"Programm zur Ausgabe eines Wochentages\n\n";
cout<<"Eingabe der Zahl fuer einen Wochentag: ";
cin>>tag;

switch (tag){
    case 1:     cout<<"Montag"<<endl; break;
    case 2:     cout<<"Dienstag"<<endl; break;
    case 3:     cout<<"Mittwoch"<<endl; break;
    case 4:     cout<<"Donnerstag"<<endl; break;
    case 5:     cout<<"Freitag"<<endl; break;
    case 6:     cout<<"Samstag"<<endl; break;
    case 7:     cout<<"Sonntag"<<endl; break;
    default:    cout<<"Bitte nur Zahlen zwischen 1 und 7 eingeben."<<endl;
                break;
}
system("PAUSE");
}
```

Zusammenfassung

- Sie sind nun in der Lage Verzweigungen in Ihrem Programm zu implementieren
- if-else dient vermehrt für Vergleiche zweier Zahlen
- switch-case dient vermehrt für Vergleiche mit Konstanten

Inhaltsverzeichnis i

Inhaltsverzeichnis ii

Wofür verwendet man Schleifen?

- wiederholtes Ausführen der gleichen Funktion im Programm
- deutliche Reduktion des Programmieraufwandes

for-Schleife

- kopfgesteuerte Schleife
- Funktion for(Anfangswert;Laufbedingung;Zählvariable)
- auszuführende Funktion in Schleife müssen in geschweifte Klammern gefasst werden

Anfangswert

Der Startwert der Laufvariablen wird gesetzt.

Laufbedingung

Bis die Bedingung nicht mehr erfüllt ist, wird der Schleifeninhalt ausgeführt.

Zählvariable

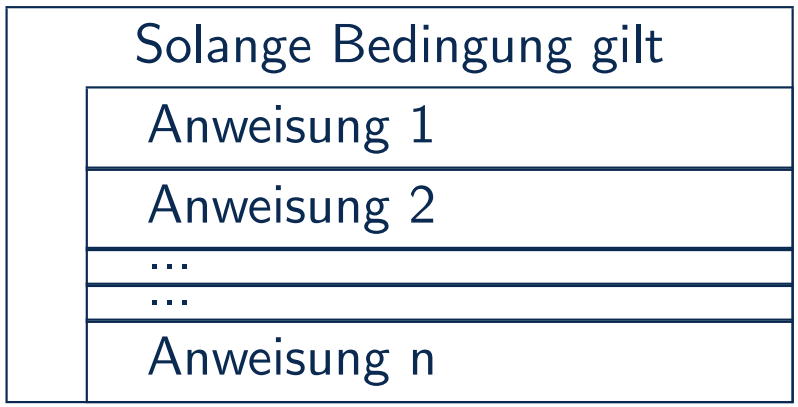
Die Operation wird nach jedem Schleifendurchlauf einmal ausgeführt.

Achtung

Einzeilige Anweisungen müssen NICHT in geschweiften Klammern geschrieben werden.

Struktogramm

Struktogramm for-Schleife



Programmbeispiel 7-1 for-Schleife.cpp

Beispiel

```
1#include <iostream>
2#include <cstdlib>
3#include <cmath>
4
5using namespace std;
6
7main()
8{
9     int i;
10
11     for(i=0;i<10;i++)
12     {
13          cout<<i<<endl;
14     }
15
16 cout<<endl<<endl<<i<<endl;
17     system("pause");
18}
```

Vorlesungsübung 3 - Aufgabe

Entwerfen Sie ein Programm zur Fakultätsberechnung einer natürlichen Zahl. Beachten Sie hierbei die Besonderheit der Zahl 0, welche in diesem Beispiel ebenfalls als mögliche Eingabe gelten soll. Überprüfen Sie die Eingaben auf gültige Zahlen mittels einer if-Bedingung. Entwerfen Sie einen Algorithmus zur Fakultätsberechnung mittels einer for-Schleife

1. Erstellen Sie zunächst ein Struktogramm, welches die Anforderungen mittels einer if-Bedingung erfüllt.
2. Entwerfen Sie anschließend ein lauffähiges C/C++-Programm, welches die Anforderungen erfüllt.

Vorlesungsübung 3 - Struktogramm

Struktogramm

<table>
<tr><td colspan="2">Bibliothek einbinden</td></tr>
<tr><td colspan="2">Variablen anlegen</td></tr>
<tr><td colspan="2">Zahl einlesen Variable n (float)</td></tr>
<tr><td>Ja</td><td>$x >= 0$?
Nein</td></tr>
<tr><td>$f = 1$ (f float)</td><td>Fehlermeldung</td></tr>
<tr><td>von $i = 1$ bis $1 <= n$
(i int)
$f = f * i$</td><td rowspan="2">$\varnothing$</td></tr>
<tr><td>Ergebnis ausgeben</td></tr>
</table>

Vorlesungsübung 3 - Lösung

```
#include <iostream>
#include <cstdlib>
using namespace std;
main()
{
int i, n, f;
cout <<" BERECHNUNG DER FAKULTAET\n\n";
cout << "Eingabe Zahl: ";
cin >> n;
if(n>=0)
{
f=1;
for(i=1; i<=n; i++)
f=f*i;
cout << "\nn! von " << n << ": " << f << "\n\n";
}
system("PAUSE");
}
```

while-Schleife

- kopfgesteuerte Schleife
- Funktion while(Laufbedingung)
- auszuführende Funktion in Schleife müssen in geschweiften Klammern gefasst werden

Laufbedingung

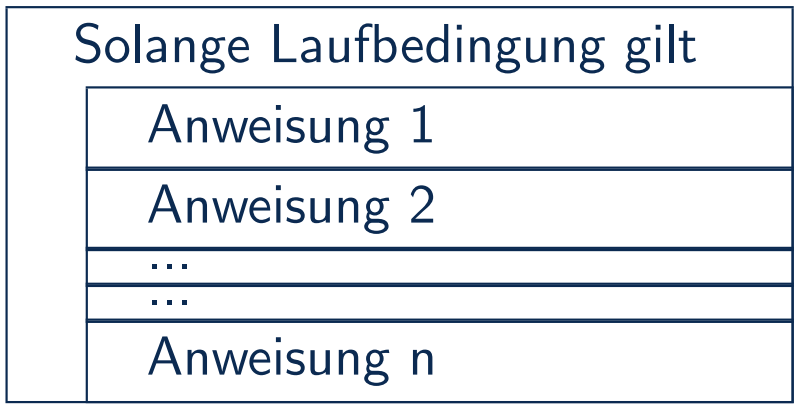
Bis die Bedingung nicht mehr erfüllt ist, wird der Schleifeninhalt ausgeführt.

Achtung

Laufvariable muss innerhalb der while-Schleife so angepasst werden, dass die while-Schleife auch wieder verlassen werden kann.

Struktogramm

Struktogramm while-Schleife



Programmbeispiel 7-2 while-Schleife.cpp

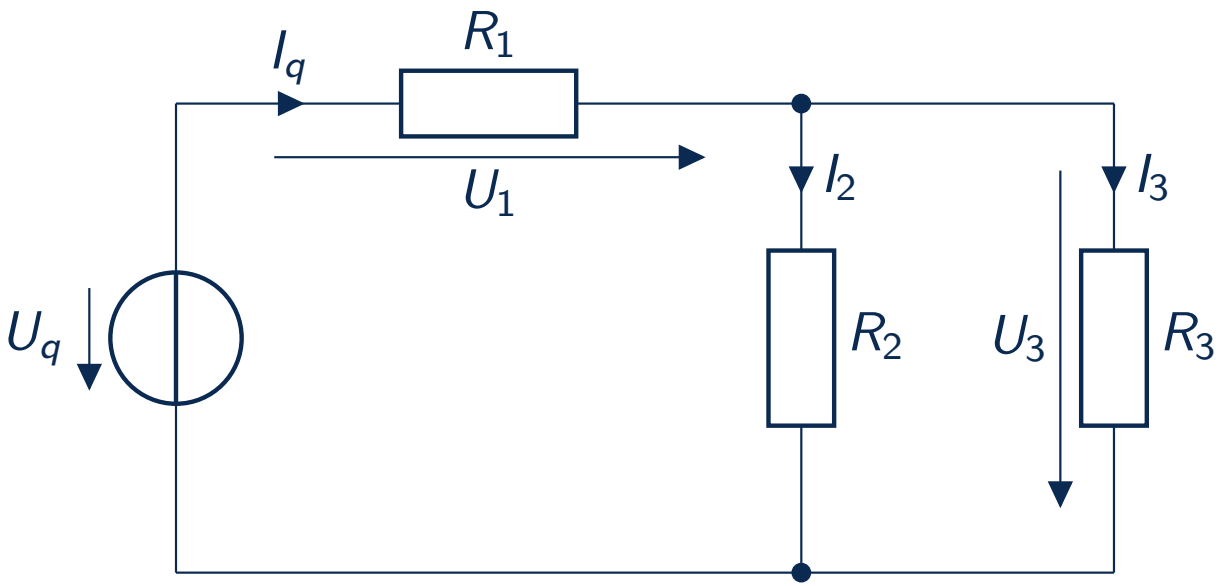
Beispiel

```
1#include <iostream>
2#include <cstdlib>
3#include <cmath>
4
5using namespace std;
6
7main()
8{
9     int i=0;              //Initialisierung der Variablen
10
11    while(i<10)
12    {
13        cout<<i<<endl;
14        i++;              //Inkrementierung der Laufvariablen
15    }
16
17
18    system("pause");
19}
```

Vorlesungsübung 4 - Aufgabe

Jetzt etwas praxisorientierter:

Berechnen Sie die Spannung über dem Widerstand R_3, wenn die Werte der Spannungsquelle U_q sowie die Werte der Widerstände R_1 und R_2 über die Tastatur eingegeben werden können. Als eingebbare Werte für R_3 sollen der minimale Wert R_{3min}, der maximale Wert R_{3max} und die Schrittwerte ΔR_3 vorgesehen werden. Geben Sie auf dem Bildschirm die Spannungswerte über R_3 aus, wenn Sie R_3 beginnend bei R_{3min} bis zum Maximalwert R_{3max} mittels einer while-Schleife um den Wert ΔR_3 erhöhen.



Vorlesungsübung 4 - Struktogramm

Struktogramm

- Bibliothek einbinden
- Variablen anlegen
- Werte über Tastatur einlesen ($U_1, R_1, R_2, R_{3min}, R_{3max}, \Delta R_3$)
- $R_3 = R_{3min}$
- Solange $R3 <= R_{3max}$
  - Berechnung durchführen
  - R_3 sowie U_{R3} am Monitor ausgeben
  - R_3 um ΔR_3 erhöhen
- Programmende

Vorlesungsübung 4 - Lösung

```
#include <iostream>
#include <cstdlib>
using namespace std;
main()
{
float fU1, fU3=0, fR1, fR2, fR3min, fR3max, fDeltaR3, fR3, fRges;

cout <<"Bitte Wert fuer U1 eingeben: "<<endl;
cin >> fU1;
cout <<"Bitte Wert fuer R1 eingeben: "<<endl;
cin >> fR1;
cout <<"Bitte Wert fuer R2 eingeben: "<<endl;
cin >> fR2;
cout <<"Bitte Wert fuer R3_min eingeben: "<<endl;
cin >> fR3min;
cout <<"Bitte Wert fuer R3_max eingeben: "<<endl;
cin >> fR3max;
cout <<"Bitte Wert fuer Delta_R3 eingeben: "<<endl;
cin >> fDeltaR3;

fR3=fR3min;
```

Vorlesungsübung 4 - Lösung

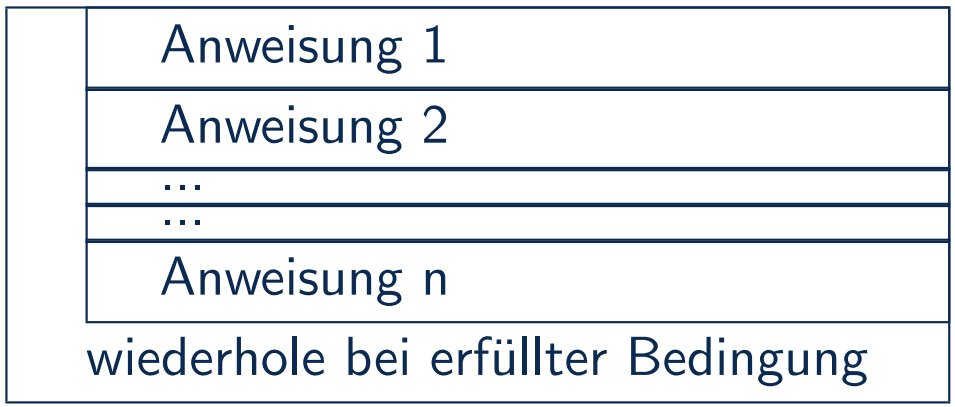
```
1 while(fR3<=fR3max)
2 {
3     //Berechnung U3
4     fRges=fR1+(fR2*fR3/(fR2+fR3));
5     fU3=fU1/fRges*(fR2*fR3/(fR2+fR3));
6     cout<<"Fuer den Widerstandswert R3 = "<<fR3<<" betraegt die Spannung U3 =
        "<<fU3<<" ."<<endl;
7     fR3=fR3+fDeltaR3;
8 }
9 cout<<"\n\t ENDE\n";
10
11 system("PAUSE");
12 }
```

do-while-Schleife

- fußgesteuerte Schleife
- Funktion do{auszuführende Funktion} while(Laufbedingung);
- auszuführende Funktion im do wird immer ausgeführt, bevor Laufbedingung überprüft wird

- ist Laufbedingung true, dann wird do wiederholt ausgeführt
- ist Laufbedingung false, dann wird nachfolgendes Programm ausgeführt

Struktogramm do-while-Schleife



Programmbeispiel 7-3 do-while-Schleife.cpp

Beispiel

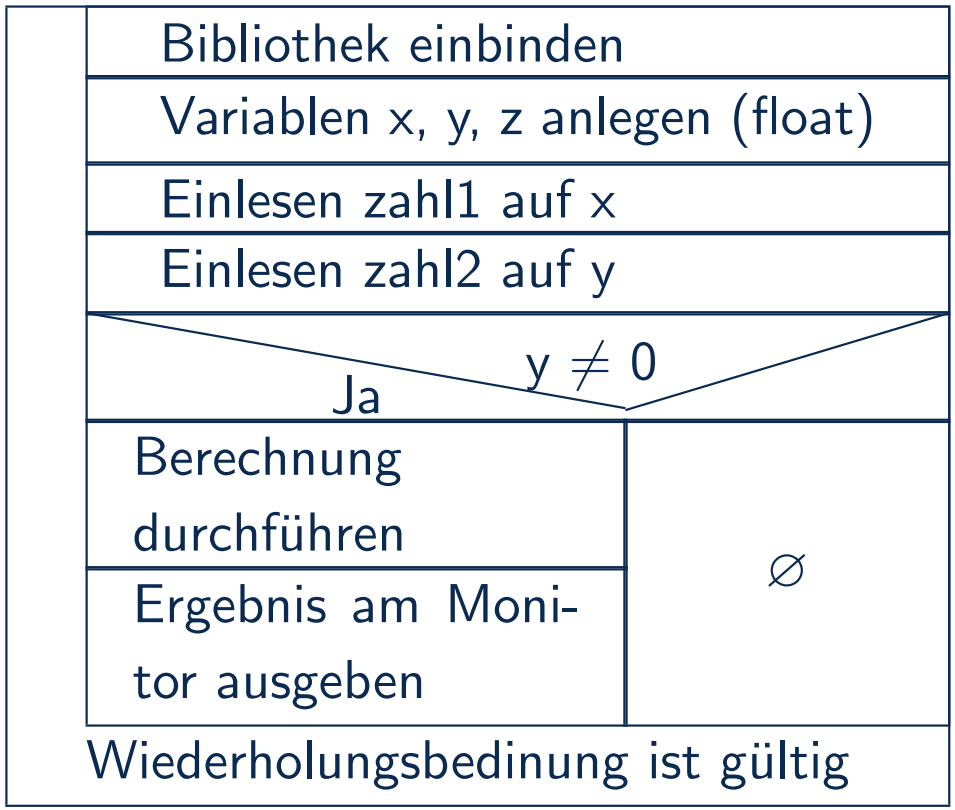
```
1#include <iostream>
2#include <cstdlib>
3#include <cmath>
4
5using namespace std;
6
7main()
8{
9     int i=0;          //Initialisierung der Variablen
10
11    do
12    {
13         cout<<i<<endl;
14         i++;          //Inkrementierung der Laufvariablen
15    }
16    while(i<10);
17
18    system("pause");
19}
```

Vorlesungsübung 5 - Aufgabe

Es ist eine Änderung zu dem Programm aus der Vorlesungsübung 1 zu erweitern.
Aufgabenstellung der Vorlesungsübung 1:
Es ist ein Programm zu entwickeln, das zwei reelle Zahlen über die Tastatur einliest und anschließend diese beiden Zahlen durch einander dividiert. Das Ergebnis ist am Monitor auszugeben. Dabei muss beachtet werden, dass der Quotient von Null verschieden ist. In diesem besonderen Fall wird die Berechnung abgebrochen.
Erweiterung: Erweitern Sie das vorliegende Programm um eine Abfrage zur Wiederholung des Programmes. Dazu soll der Nutzer zu einer Eingabe für die Wiederholung aufgefordert werden. Nutzen Sie hierfür eine do-while-Schleife.
1. Erstellen Sie zunächst ein Struktogramm, welches die Anforderungen mittels einer if-Bedingung erfüllt.
2. Entwerfen Sie anschließend ein lauffähiges C/C++-Programm, welches die Anforderungen erfüllt.

Vorlesungsübung 5 - Struktogramm

Struktogramm



Vorlesungsübung 5 - Lösung

```
#include <iostream>
#include <cstdlib>
using namespace std;
main()
{
float x,y,z=0.0;
char wiederholung;
do{
cout<<"\nEingabe Zahl 1: ";
cin>>x;
cout<<"\nEingabe Zahl 2: ";
cin>>y;
if(y!=0)
{
z=x/y;
cout<<"\n"<<z;
}
cout<<"\n\n";
cout<<"Moechten Sie das Programm erneut ausfuehren?";
cin>>wiederholung;
}
while(wiederholung=='1'||wiederholung=='y'||wiederholung=='j');
system("pause");
}
```

Zusammenfassung

- Sie sind nun in der Lage wiederholte Anweisungen in Ihrem Programm mittels Schleifen zu implementieren.

- verschiedenen Situationen bedingen verschiedene Schleifen-Typen
- besondere Aufmerksamkeit gilt der for-Schleife im Bezug auf Arrays

Inhaltsverzeichnis i

Inhaltsverzeichnis ii

Verwendung von Arrays

- Variablen können mehr als einen Wert speichern
- geschriebenes Programm kann mit Arrays dynamischer gehandhabt werden
- platzsparende (und arbeitssparende) Alternative zu verherigen Lösungsansätzen

Aufbau von Arrays

- Array ist Variable
- `int nArray[Anzahl-der-Elemente]`

Achtung

Die Anzahl der Elemente inkludiert den 0-ten Eintrag des Arrays.

Beschreiben eines Arrays

- Elemente können im Code vorgegeben werden
- einzelnes Beschreiben der Elemente aus der Eingabe
- Beschreiben der Elemente aus Eingabe mittels Schleife

Programmbeispiel 8-1 Array.cpp

Beispiel

```
1#include <iostream>
2#include <cstdlib>
3#include <cmath>
4
5using namespace std;
6
7main()
8{
9     int nArray[3]={1,2,3};   //Array mit 3 Eintraegen
10
11    cout<<nArray[0]<<endl;   //erstes Element des Arrays
12    cout<<nArray[1]<<endl;   //zweites Element des Arrays
13    cout<<nArray[2]<<endl;   //drittes Element des Arrays
14
15    system("pause");
16}
```

Programmbeispiel 8-2 Array-Schleife.cpp

Beispiel

```
1#include <iostream>
2#include <cstdlib>
3#include <cmath>
4
5using namespace std;
6
7main()
8{
9     int i;
10    int nArray[100];          //Array mit 3 Eintraegen
11
12    for(i=0;i<100;i++)        //Beschreiben der Elemente des Arrays
13        nArray[i]=i;
14
15    for(i=0;i<100;i++)
16    cout<<"Das "<<i<<"-te Element dieses Arrays hat des Wert "<<nArray[i]<<".
      "<<endl;
17                              //Ausgabe der Elemente des Arrays
18
19    system("pause");
20}
```

Programmbeispiel 8-3 Array-Schleife-mit-Eingabe.cpp

Beispiel

```
1#include <iostream>
2#include <cstdlib>
3#include <cmath>
4using namespace std;
5
6main()
7{
8     int i;
9     int nArray[3];            //Array mit 3 Eintraegen
10
11    for(i=0;i<3;i++)          //Beschreiben der Elemente des Arrays
12    {
13    cout<<"Bitte geben Sie das "<<i<<"-te Element des Arrays ein: ";
14    cin>>nArray[i];
15    }
16
17    for(i=0;i<3;i++)
18    cout<<"Das "<<i<<"-te Element dieses Arrays hat des Wert "<<nArray[i]<<".
      "<<endl;
19                              //Ausgabe der Elemente des Arrays
20    system("pause");
21}
```

Zusammenfassung

- Sie können nun mit geringem Aufwand viele Werte schnell und übersichtlich abspeichern
- Arrays sind Funktionen, auf denen die Funktionen `vektor()` und `matrix()` aufgebaut sind
- Letztere werden im Rahmen dieses Kurses nicht weiter behandelt

Inhaltsverzeichnis i

Inhaltsverzeichnis ii

Nutzen von Unterprogrammen

- *endlich* Übergang von C zu C++
- Aufruf einzelner Methoden, ohne sich um Inhalt weiter Gedanken machen zu müssen

Aufbauarten von Unterprogrammen

Prozeduales Programmieren

- schrittweises Abarbeiten einzelner Programmschritte
- Vorgehensweise für große Programme nicht optimal

Objektorientiertes Programmieren

- Aufrufen benötigter Unterprogramme/Methoden
- einmalige Arbeit zur Erstellung benötigter Methoden
- beliebig häufiger Aufruf der Methoden

Grundlegender Aufbau von Unterprogrammen

- Methode muss vor der main-Methode aufgeführt werden
- Definition des Datentyps des Rückgabewertes
- Übergabeparameter müssen festgelegt werden

Achtung

Variablen im Unterprogramm und im Hauptprogramm können gleich heißen, sind aber nicht die selben Variablen! (Lokalvariablen)

Programmbeispiel 9-1 Unterprogramme.cpp

Beispiel

```
1  #include <iostream>
2  #include <cstdlib>
3  #include <math.h>
4
5  using namespace std;
6
7  int unterprogramm(int a, int b)
8  {
9      int z;
10     z=a-b;
11     return z;
12 }
13
14 main()
15 {
16     int x,y;
17
18     if(unterprogramm(x,y)==5)
19     cout<<"Der Rueckgabewert des Unterprogrammes ist "<<unterprogramm(x,y)<<"
       ."<<endl;
20
21     else
22     system("pause");
23 }
```

Zusammenfassung

- Sie können nun Unterprogramme erstellen und Parameter übergeben.
- Verwenden Sie Unterprogramme sinnvoll, häufig
- Sie verfügen nun über alle notwendigen Kenntnisse, um die Übungsaufgaben lösen zu können.

Nützliches

- Algorithmusentwurf *bevor* Sie beginnen zu programmieren
- nutzen Sie Unterprogramme häufig
- programmieren Sie nachvollziehbar und nutzen Sie **Kommentare**
- nutzen Sie Programmpassagen, die Sie bereits früher einmal entwickelt haben für neue Programme

Wichtig

Bei neuen Funktionen sind Google & Co. Ihre besten Freunde.
Nutzen Sie als Quellen möglichst **GitHub** und **StackOverflow**.

[9]

[10]

[9] https://cdn.sstatic.net/Sites/stackoverflow/company/img/logos/so/so-logo.png?v=9c558ec15d8a
[10] https://github.com/logos

11

[11] https://www.instagram.com/p/B-i9Ja9A8ia/

Inhaltsverzeichnis i

Inhaltsverzeichnis ii

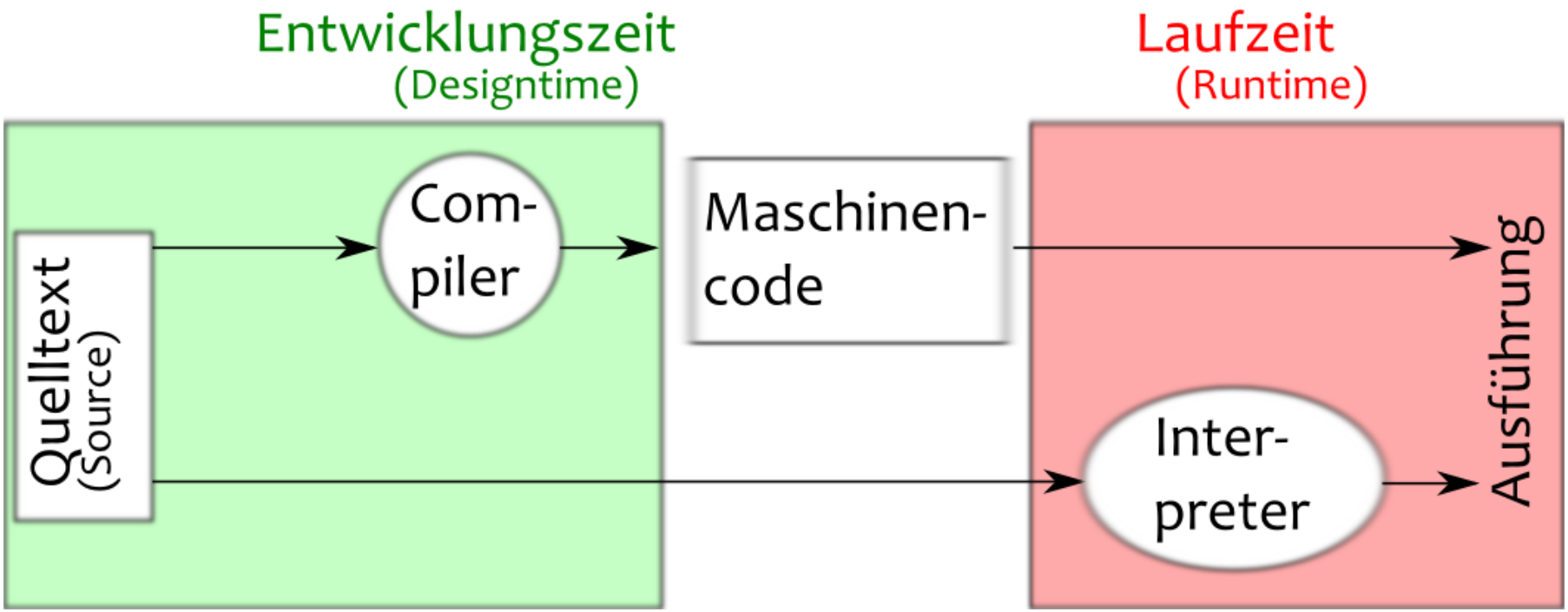
Was ist Python?

[12]

- höhere Programmiersprache
- nutzt Compiler und/oder Interpreter

[12] https://upload.wikimedia.org/wikipedia/commons/thumb/f/f8/Python_logo_and_wordmark.svg/486px-Python_logo_and_wordmark.svg.png

Programmübersetzung



13

[13] http://www.roro-seiten.de/uhu/img.php?ID=659

Die Programmiersprache Python: Vor- und Nachteile

Vorteile

- saubere und einfache Programmiersprache
- kein Semikolon
- Pythonspezifische Datentypen erleichtern Programmierung ungemein
- Verwendung bestimmter Bibliotheken sparen viel Zeit

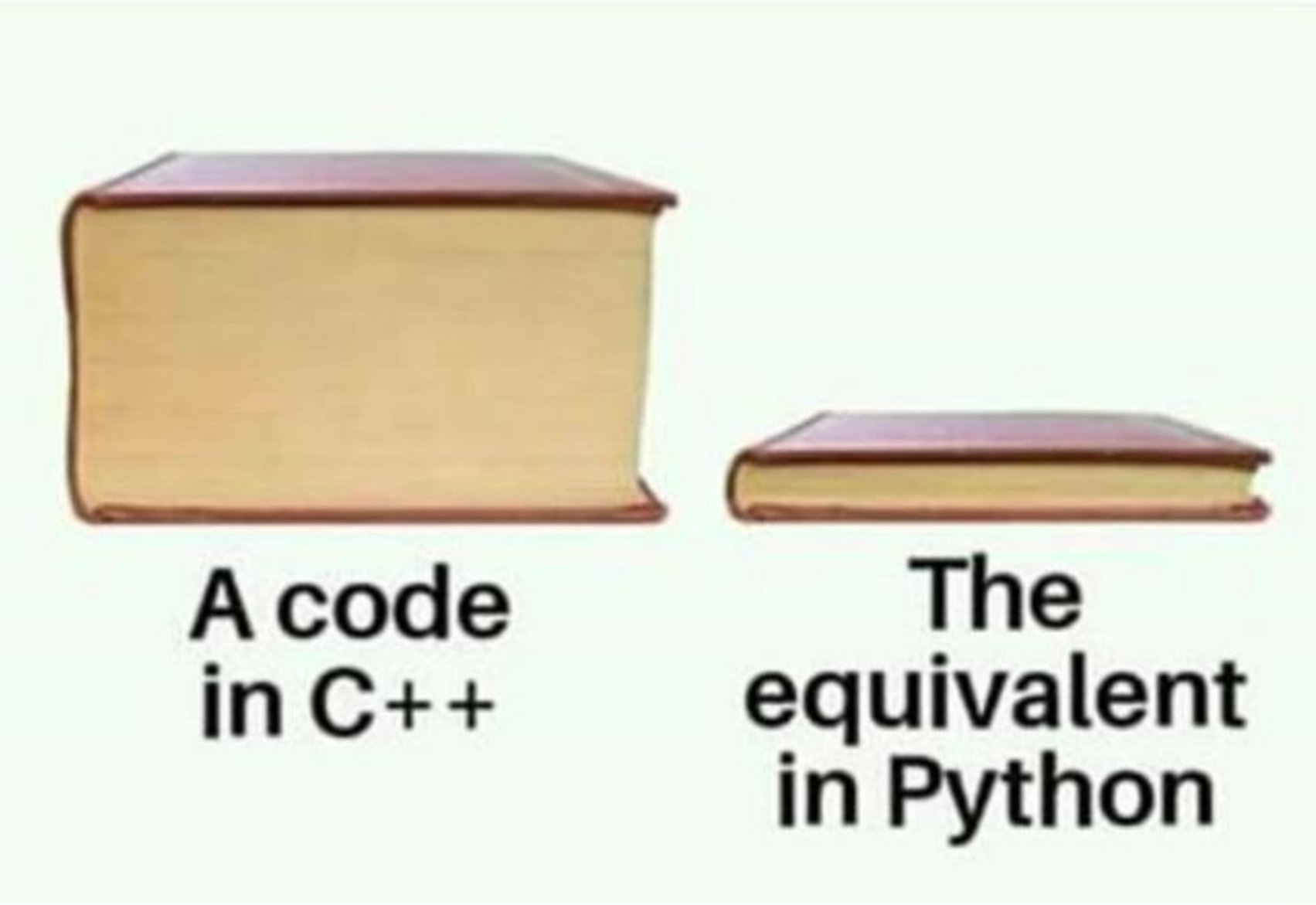
Wichtig

Besonders die Bibliotheken *SciPy* und *NumPy* werden Ihnen häufig begegnen.

Nachteile

- Ausführbarkeit *.py

  Erstellung in *.txt → Umbenennung in *.py → Ausführung in Shell
- Ausführungsgeschwindigkeit (interpretierte Sprache)



14

[14] https://www.instagram.com/p/B-a_6Xqgdrg/

Einführung in die Programmierung

Grundlegendes

- Kommentare beginnen mit #
- Einbinden von Bibliotheken mit `import`
- Bildschirmausgaben mit `print()`
- Tastatureingaben mit `input()`

Programmbeispiel Eingabe

Beispiel

```
1 x = input("Ihr Name? ")
2 print(x)
```

Programmbeispiel if-Bedingung

Beispiel

```
1 a = 10
2 b = 15
3
4 if b > a:
5   print("b ist groesser als a")
6 elif a == b:
7   print("a ist gleich b")
8 else:
9   print("a ist groesser als b")
```

Programmbeispiel switch-case

Beispiel

```
1  def woche(tag):
2          return{
3                  1:'Montag',
4                  2:'Dienstag',
5                  3:'Mittwoch',
6                  4:'Donnerstag',
7                  5:'Freitag',
8                  6:'Samstag',
9                  7:'Sonntag'
10             }.get(tag,"Ungueltiger Wochentag")
11 print(woche(tag=2))
```

Programmbeispiel for-schleife

Beispiel

```
1 for i in range(1, 5):
2         print (i)
3 else:
4         print ('Die for-Schleife ist zu Ende.')
```

Programmbeispiel while-Schleife

Beispiel

```
1 n = 1000
2 s = 0
3 i = 1
4 while i <= n:
5     s = s + i
6     i = i + 1
7
8 print ("Die Summe lautet: ", s)
```

Bibliotheken

Bibliothek NumPy

- ein leistungsfähiges N-dimensionales Array-Objekt
- ausgefeilte (Rundfunk-) Funktionen
- Tools zur Integration von C / C ++ - und Fortran-Code
- nützliche Funktionen für lineare Algebra, Fourier-Transformation und Zufallszahlen

Bibliothek SciPy (baut auf NumPy auf)

- enthält Module für Optimierung, lineare Algebra, Integration, Interpolation,
- beinhaltetet Sonderfunktionen, FFT, Signal- und Bildverarbeitung, ...

Bibliothek Matplotlib (baut auf NumPy auf)

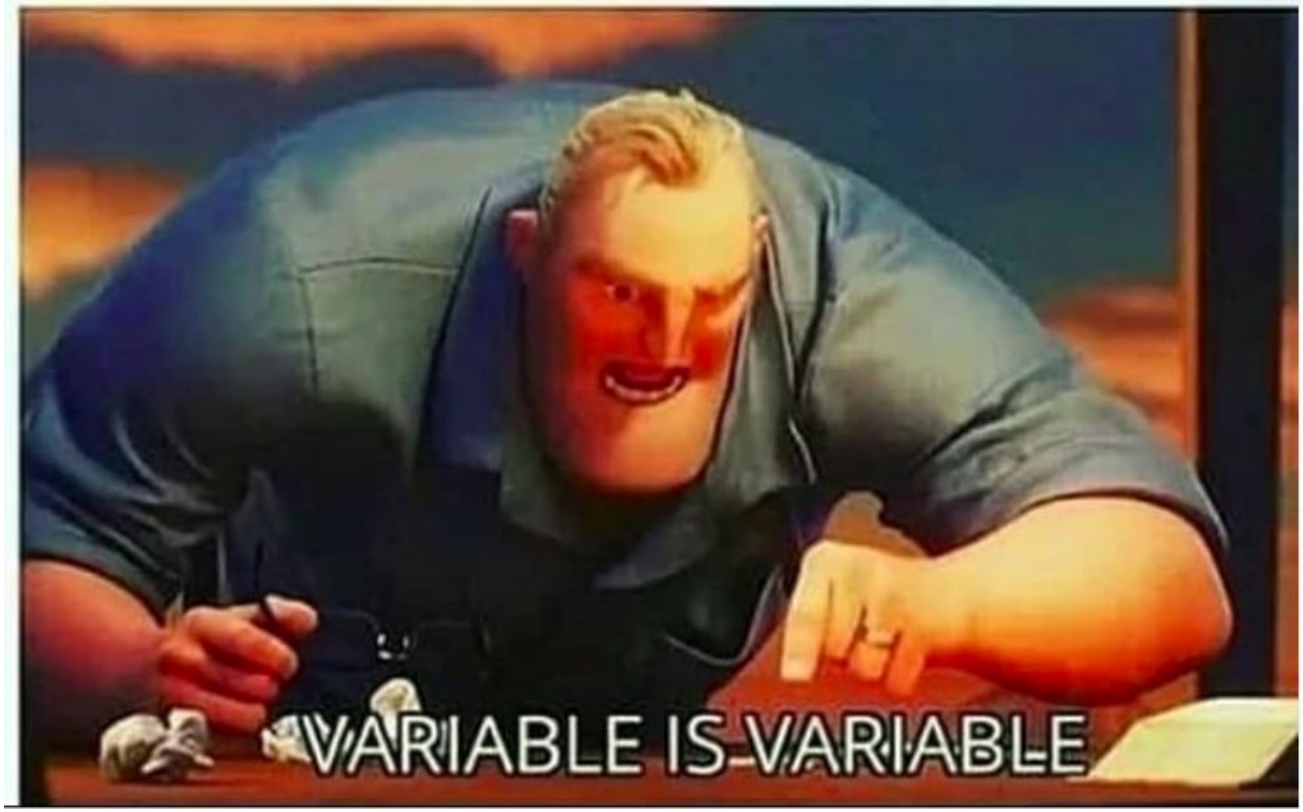
- SciPy verwendet Matplotlib
- objektorientierte API zum Einbetten von Plots

Besonderheiten von Python

- vollständig freie Software
- nutzt Compiler und/oder Interpreter
- kostenfreie Alternative zu Matlab
- direkte Berechnung komplexer Zahlen mit `complex(real[], imag[])` möglich
- Vergleiche von Zahlen mit unterschiedlichen Datentypen

C++: Can not compare float and int

Python:



15

[15] https://www.instagram.com/p/B-eOoIQgtec/

Python - Kursangebot für Einsteiger

Pythonkurs für Ingenieur*innen im Wintersemester 21/22

Herr Dr.-Ing. Carsten Knoll
Institut für Regelungs- und Steuerungstheorie
Institutsgebäude S7a, Etage 4, Raum 409
Georg-Schumann-Straße 7a
01187 Dresden
carsten.knoll@tu-dresden.de

Der Kurs ist vollkommen *kostenfrei*.
Die Einschreibung erfolgt voraussichtlich über OPAL.

Inhaltsverzeichnis i

Inhaltsverzeichnis ii

Die Idee von LaTeX

- MS Word, Open Office writer vs. LaTeX
- What you see is what you get vs. What you mean is what you get
- genießt besondere Bedeutung bei Erstellung wissenschaftlicher Arbeiten

Arbeiten mit LaTeX

- keine Live-Ausgabe des Dokumentes
- Formregeln müssen bei Erstellung von Anfang an beachtet werden
- Schwerpunkt liegt auf Inhalt, Formatierung ergibt sich weitestgehend aus Formregeln
- dank lebendiger Community und Paketservice ist fast nichts unmöglich

Wichtig

Es werden Pakete zur Nutzung verschiedener Funktionen benötigt, die über verschiedene Portale heruntergeladen werden können.

Dokumentenstruktur - Die Präambel

Dokumentenklassen

- viele möglich, auch eigene Dokumentklassen definierbar
- gebräuchliche: `article`, `book`, `beamer`, `dinbrief`
- erste Zeile in der `*.tex` Datei

Pakete

- Einbinden von Paketen für einzelne Funktionen mit `\usepackage[PAKETOPTIONEN]{PAKETNAME}`
- oft verwendete Pakete z. B. `babel`, `graphicx`, `tabularx`, `amsmath`, `color`

Dokumentenstruktur - Textteil

Anfang- und Ende des Dokuments

- Präambel (Definition Dokumentklasse, Einbinden von Paketen)
- Beginn des eigentlichen Dokumenteninhalts mit `\begin{document}`
- Ende des Dokuments mit `\end{document}`

Titelseite

```
\author{Max Mustermann}
\title{Dokumententitel}
\date{Datum} \today für aktuelles Datum
```

- einfache Erzeugung einer Titelseite möglich `\maketitle`
- Definition von Autor, Titel und Datum VOR `\begin{document}`

Minimalbeispiel Dokumentenaufbau

Beispiel

```
\documentclass[11pt,a4paper]{article}

\usepackage[ngerman]{babel}
\usepackage{geometry}

\author{Max Mustermann}
\title{Dokumententitel}
\date{\today}

\begin{document}
\maketitle
TEXT                    %Kommentare werden mit Prozentzeichen
\end{document}
```

Dokumentenstruktur – Textteil

Gliederung

```
\chapter{Überschrift}
\section{Titel der Kapitelüberschrift}
\subsection{Titel der Unterkapitelüberschrift}
\subsubsection{Titel der Unterunterkapitelüberschrift}
\paragraph{Titel Absatz}
\appendix              %Anhang
```

- Überschriften werden in Schriftart und -größe einheitlich formatiert
- konkrete Bennenung der Kapitel wichtig um später problemlos das Inhaltsverzeichnis erzeugen zu können

Achtung

Nicht alle Gliederungspunkte sind für alle Dokumentklassen gültig!

Schriftformatierung i

Schriftgröße

Eine globale Änderung der Schriftgröße ist am Dokumentanfang möglich. Lokal ist eine relative Schriftgrößenänderung wie folgt möglich:

```
\tiny{sehr klein}
\small{klein}
\normalsize{normal}
\large{etwas groesser}
\Large{gross}
\LARGE{sehr gross}
\huge{riesig}
```

sehr klein
klein
normal
etwas größer
groß
sehr groß
riesig

Schriftformatierung ii

Schriftart

`\textrm{Roman}`	Roman
`\texttt{Schreibmaschine}`	`Schreibmaschine`
`\textbf{Fett}`	**Fett**
`\emph{Kursiv}`	*Kursiv*
`\textit{Kursiv}`	*Kursiv*
`\textsl{schief}`	*schief*
`\textsf{Sans Serif}`	Sans Serif
`\textsc{Kapitälchen}`	Kapitälchen
`\underline{unterstrichen}`	unterstrichen

Aufzählungen

ohne Nummerierung

```
\begin{itemize}
\item erster Stichpunkt
\item zweiter Stichpunkt
\end{itemize}
```

- erster Stichpunkt
- zweiter Stichpunkt

mit Nummerierung

```
\begin{enumerate}
\item erster Stichpunkt
\item zweiter Stichpunkt
\end{enumerate}
```

1. erster Stichpunkt
2. zweiter Stichpunkt

Verwendung von besonderen Zeichen

Deutsche Sonderzeichen ä, ö, ü, ß

Einbinden des Pakets `german` oder der Paketoption `ngerman` des Pakets `babel` am Beginn des Dokuments notwendig.

Symbole

- Einbinden von Symbol-Paketen z.B. `tipa` für phonetisches Alphabet oder `textcomp`
- jedes Paket enthält unterschiedliche Symbole, entnehmbar den jeweiligen Paketdokumentationen
- **Beispiele** `\textcopyright` © `\textmusicalnote` ♪ `\textohm` Ω

Formeln

- Einbinden von Mathe-Paketen notwendig z.B. `amsmath` und `amssymb`
- **Beispiel** $\sqrt{\frac{sin(\alpha)}{d}}$ `$\sqrt{\frac{sin(\alpha)}{d}}$`

Verzeichnisse erstellen mit LaTeX

Inhaltsverzeichnis

- automatische Erstellung mit `\tableofcontents` an gewünschter Stelle

Abbildungsverzeichnis

- automatische Erstellung mit `\listoffigures` an gewünschter Stelle

Tabellenverzeichnis

- automatische Erstellung mit `\listoftables` an gewünschter Stelle

Quellcodeverzeichnis

- automatische Erstellung mit `\lstlistoflistings` an gewünschter Stelle

Literaturverzeichnis

- automatische Erstellung mit `\printbibliography` an gewünschter Stelle
- vorheriges Anlegen einer `*.bib` Datei notwendig
- `*.bib` Datei enthält detaillierte Informationen über verwendete Quellen

LaTeX- Kursangebote für Einsteiger

Workshop: Dokumente erstellen mit Latex – eine Einführung

Andreas Kreisel
Technische Universität Dresden
Institut für Wirtschaft und Verkehr
Professur für Kommunikationswirtschaft
Würzburger Straße 35
01187 Dresden
andreas.kreisel@tu-dresden.de

Der Kurs ist vollkommen *kostenfrei*.
Die Einschreibung erfolgt voraussichtlich über OPAL.
(Organisator ist der Career Service der TU Dresden)