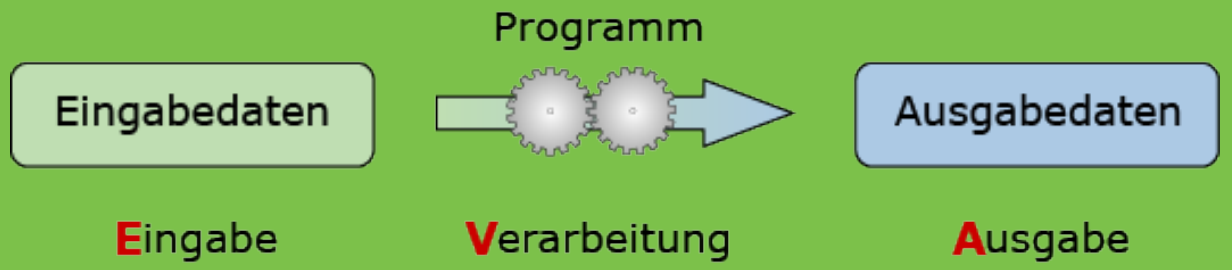
Hochschule Fulda
University of Applied Sciences

Programmierung 1 – Einführung

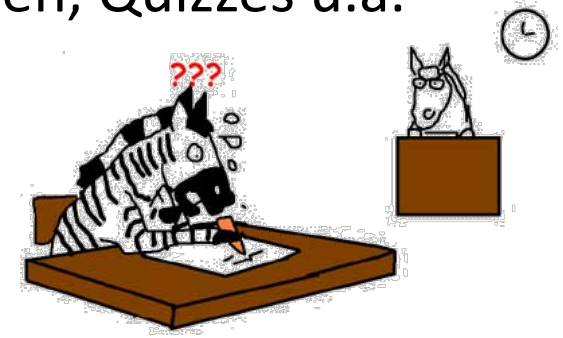


Yvonne Jung

Organisatorisches

Hochschule Fulda
University of Applied Sciences

- Lehrveranstaltung
  - Seminaristischer Unterricht mit begleitenden Übungen im Rechnerraum
    - Vorlesungsstoff gilt stets für Übungen der darauffolgenden Woche
  - Workload: 150h (davon 45h Selbststudium, d.h. ca. 3h pro Woche)
- Lernplattform: Moodle
  - https://elearning.hs-fulda.de/ai/course/view.php?id=1193
  - → hier finden sich Ankündigungen, Folien, Übungsaufgaben, Quizzes u.ä.
- Prüfung
  - Klausur (klassisch schriftlich, wahrscheinlich 90 min.)
  - Ähnlich wie Übungen, aber keine Unterlagen erlaubt



Lernziele

Hochschule Fulda
University of Applied Sciences

- Die Studierenden verstehen mathematische und logische Probleme in natürlicher Sprache
  - Z.B. Zahlenfolgen und -reihen, Sortieren, Game of Life, Türme von Hanoi
- Sie sind in der Lage, diese Probleme algorithmisch zu beschreiben und unter Anwendung der ihnen bekannten Programmkonstrukte programmiersprachliche Lösungen zu entwickeln
  - Diese Lösungen sind in lesbarem Code formuliert
  - Die Studierenden können Einschätzungen zu Laufzeit und Speicherverwaltung dieser Programme treffen
- Sie kennen Strategien zur Fehlereingrenzung, -suche und -behebung und können diese anwenden

Lerninhalte

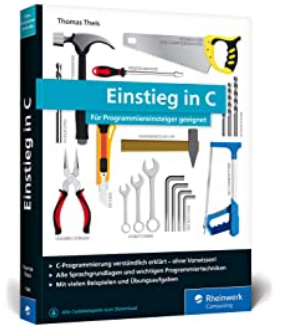
Hochschule Fulda
University of Applied Sciences

- Primitive Datentypen für Zahlen, Wahrheitswerte und Zeichenketten
- Kontrollstrukturen (bedingte Anweisungen, Schleifen)
- Prozeduren und Funktionen, Parameterübergabe, Rückgabewerte
- Strukturierte Datentypen
- Speicherverwaltung, Stack- und Heap-allokierte Daten
- Rekursive Prozeduren und Funktionen
- Einfache rekursive Datentypen wie Listen
- Testen und Debuggen, Lesbarer Code
- Laufzeit

Literaturvorschläge

Hochschule Fulda
University of Applied Sciences

- Thomas Theis: *Einstieg in C – für Programmiereinsteiger geeignet* (3. Auflage). Rheinwerk Verlag, 2020
  - Online als E-Book in Hochschulbibliothek verfügbar: https://hds.hebis.de/hlbfu/Record/HEB464231817
  - Vorlesung wird ggfs. auf entsprechende Kapitel verweisen
  - Buch dient zur Vertiefung und zum Selbststudium

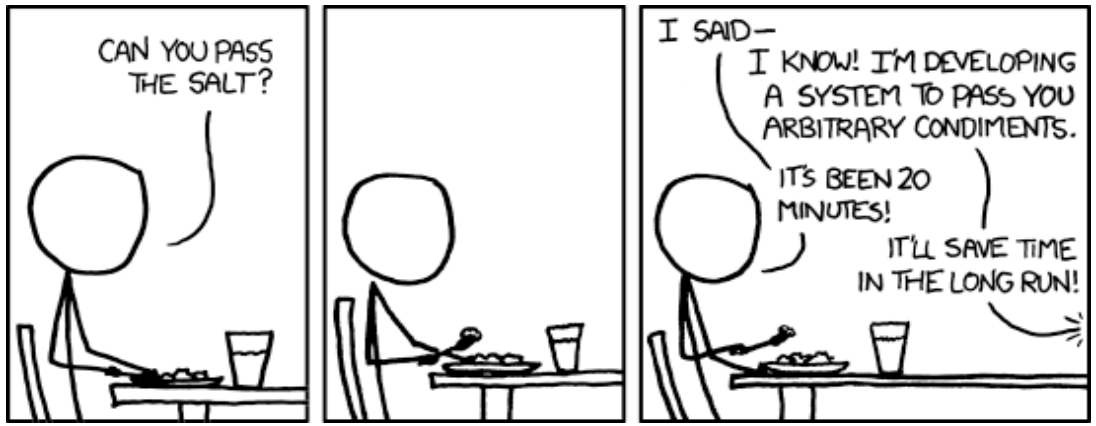


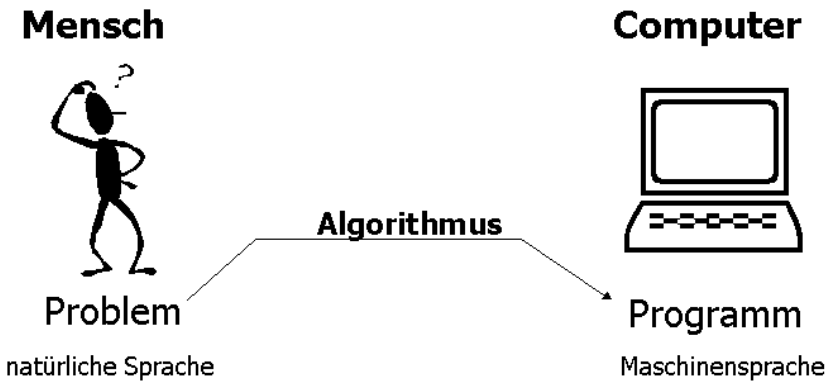
- Jürgen Wolf: *C von A bis Z* (3. Auflage). Rheinwerk Verlag, 2009 https://openbook.rheinwerk-verlag.de/c_von_a_bis_z/
- Brian W. Kernighan & Dennis M. Ritchie: *Programmieren in C* (2. Ausgabe, ANSI C). Hanser Fachbuch, 1990
- Elias Fischer: C-HowTo (https://www.c-howto.de/tutorial/), 2012

Vom Problem zum Programm

Hochschule Fulda
University of Applied Sciences

- Problemstellung: Aus einer unbefriedigenden Ausgangssituation soll eine verbesserte Zielsituation gemacht werden





- Algorithmus: Verfahren zur Lösung eines Problems (Handlungsabfolge)
- Programm: Formulierung eines Algorithmus in einer bestimmten Programmiersprache, die auf konkretem Prozessor ausführbar ist

Algorithmus

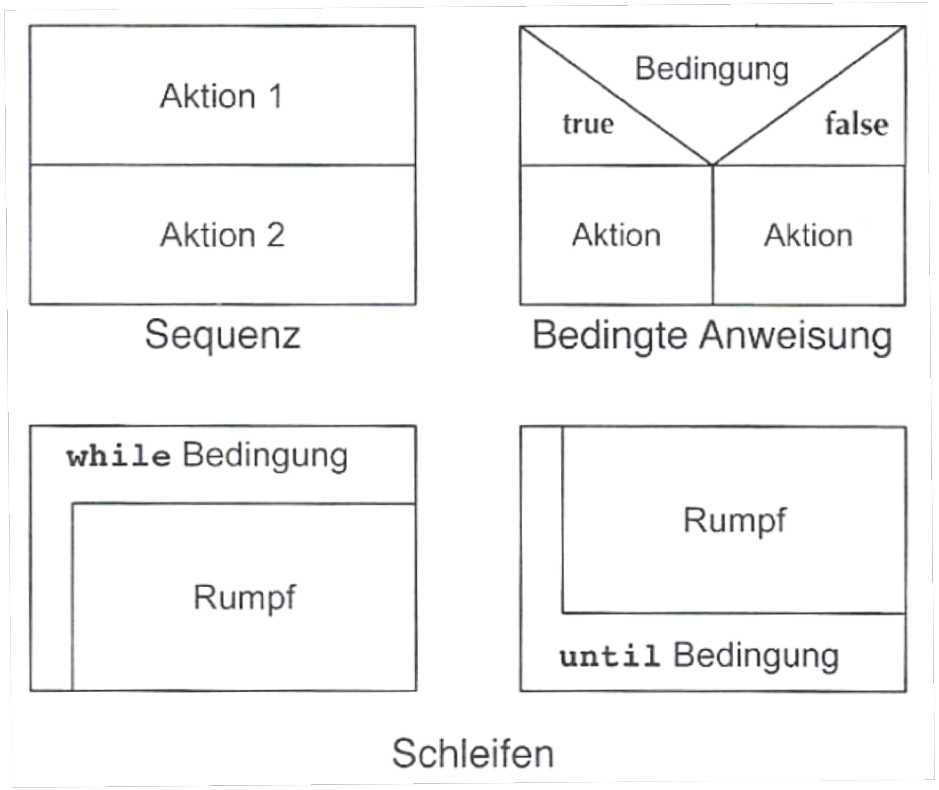
Hochschule Fulda
University of Applied Sciences

- Eindeutige Beschreibung eines Verfahrens zur Lösung bestimmter Problemklassen
  - Menge von Regeln für ein Verfahren, um aus gewissen Eingabegrößen bestimmte Ausgabegrößen herzuleiten
  - Muss korrekt sein, d.h. für alle Eingaben richtige Ergebnisse liefern
- Präzise, endliche Beschreibung eines allgemeingültigen Verfahrens unter Verwendung elementarer ausführbarer Verarbeitungsschritte
  - Abfolge der einzelnen Schritte muss eindeutig festgelegt sein
  - Muss so genau formuliert sein, dass auch von Maschine ausführbar
  - Darf zu jedem Zeitpunkt nur endlich viel Speicherplatz benötigen
  - Und soll nach endlich vielen Schritte terminieren

Darstellung von Algorithmen

Hochschule Fulda
University of Applied Sciences

- (Umgangssprachlich)
- Pseudocode
  - Besser strukturiert als reine "Prosa", aber weniger detailliert als Programmiersprache
- Programmablaufplan (PAP)
  - Graphische Darstellung als Flussdiagramm
- Struktogramm
  - Formalisierte graphische Darstellung von Programmentwürfen (auch bekannt als Nassi-Shneiderman-Diagramme; s. Abb.)
- Programmiersprache



Beispiel für Algorithmus

Hochschule Fulda
University of Applied Sciences

- Bedienung einer Waschmaschine als Sequenz (Folge) von Anweisungen
  - Tür der Waschmaschine öffnen.
  - Max. 5 kg Wäsche (einer Farbe ☺) einfüllen.
  - Tür der Waschmaschine schließen.
  - Waschmittel passend zur Farbe der Wäsche in die kleine Schublade für den Hauptwaschgang füllen.
  - Wasserzulauf öffnen.
  - Waschprogramm wählen.
  - Starttaste drücken.
  - Waschvorgang abwarten.
  - Nach Programm-Ende Maschine abstellen.
  - Wasserzulauf schließen.
  - Tür öffnen und Wäsche entnehmen.

- Könnte man das so auch einem Roboter sagen?
- Vorgehensweise beim Algorithmenentwurf: Verfahren schrittweise entwickeln und immer weiter verfeinern, bis es von Maschine / Computer ausgeführt werden kann

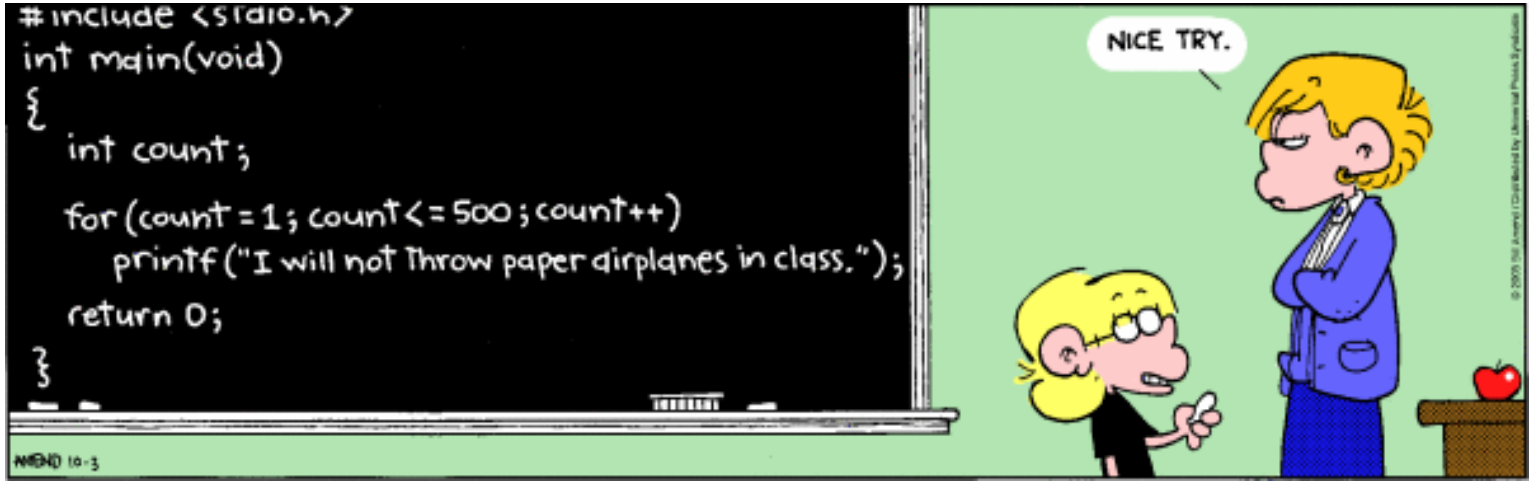
Programmiersprache

Hochschule Fulda
University of Applied Sciences

- Sog. formale Sprache, mit deren Hilfe ein Programm formuliert ist
- Wird festgelegt durch
  - Syntax: definiert Regeln (Grammatik und Orthographie), nach denen ein "Satz" eines Programms formuliert wird
    - Bsp.: "Atlas" ist syntaktisch richtige Aneinanderreihung der Buchstaben A, t, l, a, s, Buchstabenfolge A, *d*, l, a, s ist im Deutschen hingegen syntaktisch falsch
  - Semantik: beschreibt Bedeutung eines "Satzes"
    - Bsp.: Semantisch kann Atlas u.a. sowohl ein geographisches Kartenwerk als auch ein Gebirge in Nordafrika sein
  - Pragmatik: beschreibt praktische Handhabung der "Sätze"
    - Bsp.: Ohne Kenntnisse des Kontexts kann die Frage "Wo ist der Atlas?" (pragmatisch gesehen) kaum beantwortet werden

Programmiersprache C

Hochschule Fulda
University of Applied Sciences



- Weltweit eine der am häufigsten eingesetzten Programmiersprachen
  - Wurde bereits Anfang der 70er (von D. Ritchie) für Unix-Programmierung entwickelt
- Einfach gehaltene höhere Programmiersprache mit hoher Portabilität, jedoch guter Anpassung an jeweilige Rechnerarchitektur
  - Zur Programmierung von Betriebssystemen, Graphik, Robotik, Simulationen usw.

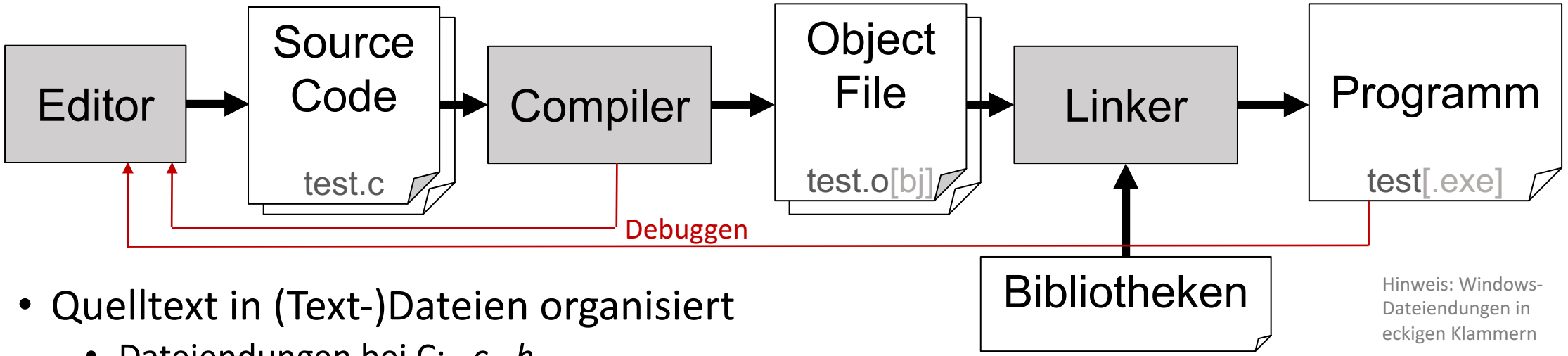
Programm und Maschine

Hochschule Fulda
University of Applied Sciences

- Anwendungsprogramme, die mit Benutzer kommunizieren, benötigen als "Grundprogramm" ein Betriebssystem (z.B. Windows oder Linux)
- Ein darauf laufendes Übersetzerprogramm (i.d.R. Compiler) übersetzt in Programmiersprache gegebenes Quellprogramm in Maschinensprache
  - Bei C erzeugt man für jede C-Datei (z.B. *test.c*) eine Objektdatei (z.B. *test.o[bj]*)
    - Beispiel (unter Linux): Kommandozeilenbefehl "`gcc -c test.c`" erzeugt "*test.o*"
  - Dann werden alle Objektdateien und externe Bibliotheken (also Module, die Funktionalitäten bereitstellen, die andere Personen programmiert haben) mit Hilfe des sog. Linkers zu ausführbarem Programm zusammengebunden
    - Bsp. Linux: Kommandozeilenbefehl "`gcc -o test test.o`" erzeugt Programm "*test*"
    - Kurzform (für Compiler *gcc*): "`gcc -o test test.c`" erzeugt wieder Programm "*test*"

Kompilieren und Linken

Hochschule Fulda
University of Applied Sciences



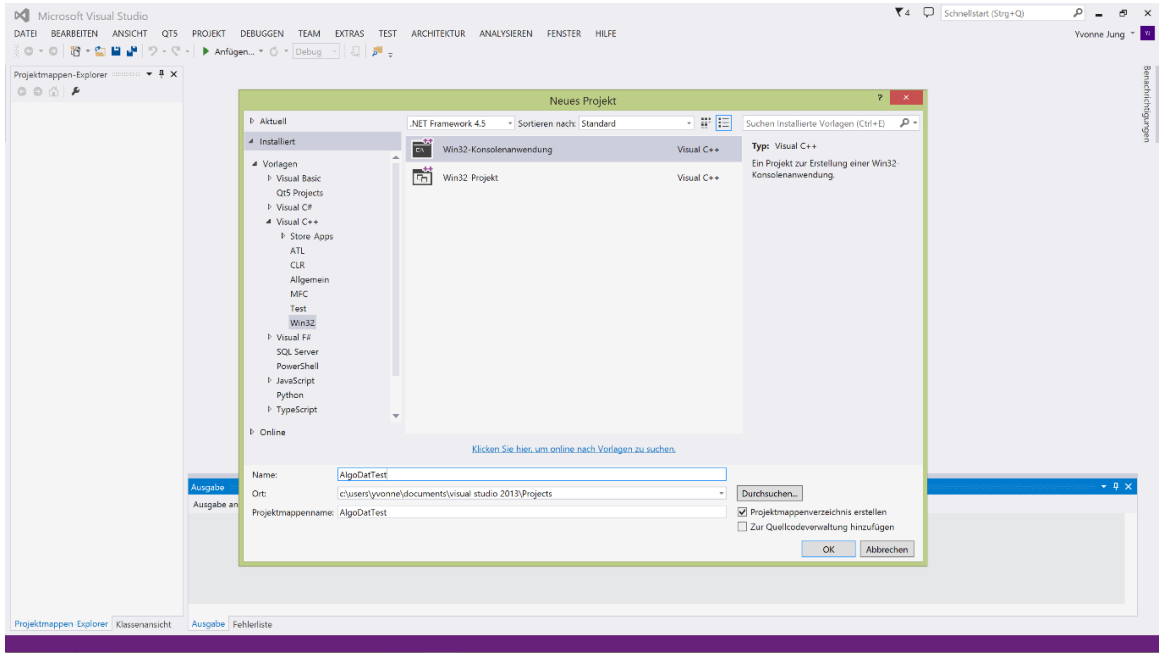
Hinweis: Windows-Dateiendungen in eckigen Klammern

- Quelltext in (Text-)Dateien organisiert
  - Dateiendungen bei C: *.c*, *.h*
- Daraus kompilierte Objektdateien bestehen aus Maschinenbefehlen
  - Sind noch "verschiebbar", d.h. noch nicht auf endgültige Adressen im Hauptspeicher festgelegt
- Linker bindet alles zu ausführbarem Programm zusammen
- Beim Programmieren passieren Fehler, diese muss man finden und verbessern
  - Sogenanntes Debugging

Programmierumgebung

Hochschule Fulda
University of Applied Sciences

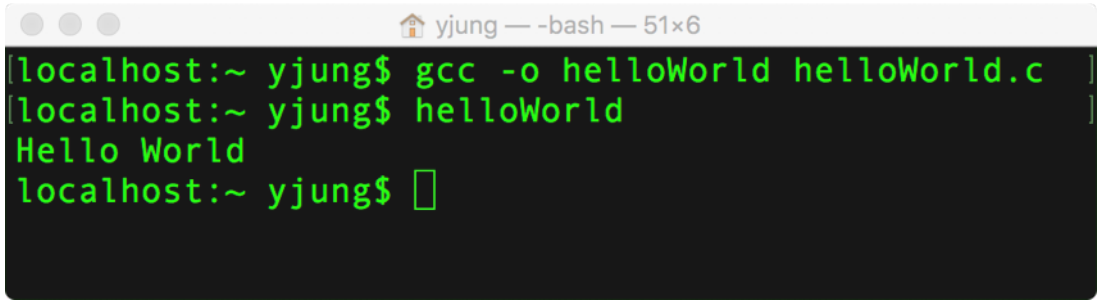
- Theoretisch reichen Texteditor (z.B. "vi") und Compiler (z.B. "gcc")
  - War unter Linux lange üblich, wird selbst heute noch hier und da genutzt
- Komfortabler sind sog. IDEs
  - Beinhalten Texteditor mit Syntax Highlighting sowie Compiler (mit Linker) und Debugger
  - Beispiele
    - Microsoft Visual Studio (Windows)
      - → Verwenden wir in Übungen
    - Xcode (Mac)
    - CLion (alle Plattformen)



Mein erstes Programm in C

Hochschule Fulda
University of Applied Sciences

- Speichern in (Quell-)Textdatei z.B. helloWorld.c
- Kompilieren (inkl. Linken) mit gcc -o helloWorld helloWorld.c
- Ausführen mit ./helloWorld
  - Hinweis: ausführbare Programme haben nur unter Windows Dateiendung *.exe*



Inkludiere Standard-I/O-Deklarationen

```c
#include <stdio.h>

int main()
{
  printf("Hello World\n");
  return 0;
}
```

Zeichenkette mit Zeilenumbruch

Gebe korrekte Terminierung an Konsole weiter

Schreibe auf Standardausgabe (Konsole)

Mein erstes Programm in C

Hochschule Fulda
University of Applied Sciences

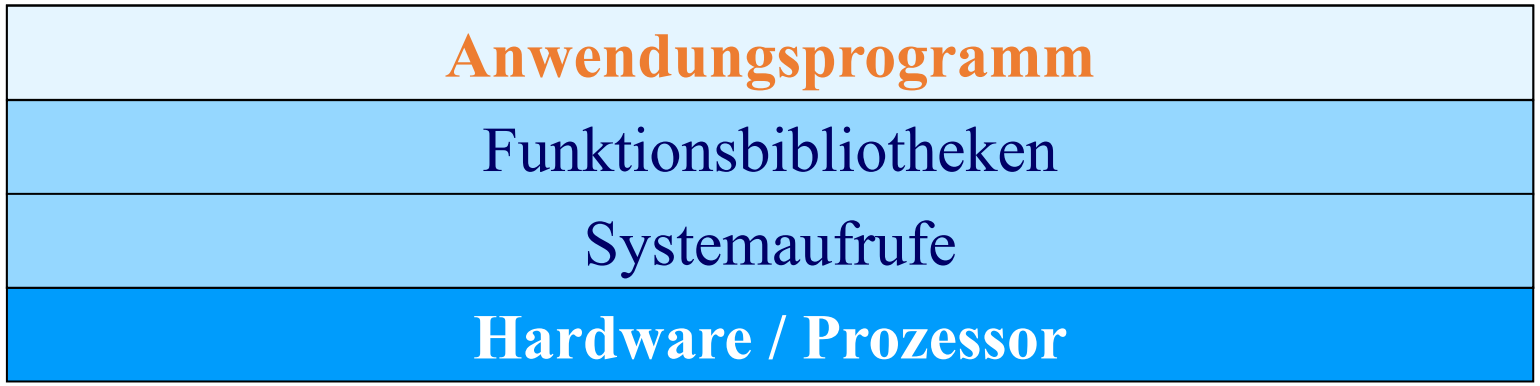
- Das C-Programm ist der Rumpf des Hauptprogrammes, also der Funktion main
- Bei Ausführung startet ein C-Programm stets mit der Funktion main
- Die Funktion printf ist vordefiniert und schreibt ihr Argument auf die Konsole
- Hier ist das Argument eine *Zeichenkette* (man sagt dazu auch String)
- Die Schlüsselwörter **int** und **return** legen fest, was von main zurückgegeben wird

```c
#include <stdio.h>

int main()
{
  printf("Hello World\n");
  return 0;
}
```

Ausführung eines C-Programms

Hochschule Fulda
University of Applied Sciences



- Variante 1: Ausführung auf Betriebssystem (BS)
  - Hardware durch BS gekapselt: Starten/Beenden des Programms über BS
  - Dabei Unterstützung durch BS-Funktionen (z.B. für Dateizugriff usw.)
- Variante 2: Ausführung direkt auf Prozessor
  - Spezielle Mikrocontroller, z.B. für Temperaturregelung
  - Alle Funktionen müssen im Programm vorhanden sein

Funktionen und Variablen

Hochschule Fulda
University of Applied Sciences

- Wie in Mathe werden Funktionen nicht nur mit Argumenten aus je passendem Definitionsbereich befüttert, z.B. `y = sin(x)`, sondern Sie können auch etwas in entsprechendem Wertebereich zurückgeben
  - Bei *main()* ist dieser Wertebereich ganzzahlig, wofür das Schlüsselwort `int` ("Integer") steht
  - Rückgabe erfolgt über Schlüsselwort `return`, wobei hier ganze Zahl 0 zurückgegeben wird
- Wie in Mathe gibt es auch Variablen, die als Platzhalter dienen für eine Zahl aus einem bestimmten Wertebereich (z.B. Ganzzahlen) bzw. beim Programmieren sogar für allgemeine Daten wie z.B. Zeichenketten
  - *Variablen haben* eine eindeutige *Bezeichnung*, einen eindeutigen *Datentyp* (Wertebereich) und zur Laufzeit einen Ort (*Adresse*) im Hauptspeicher, an dem der *Wert* der Variablen steht
  - Beispiel: `int zahl = 42;`

Sprachelemente

Hochschule Fulda
University of Applied Sciences

- In C gibt es etwas mehr als 30 Schlüsselwörter (essentielle Sprachelemente), welche man, inkl. Bedeutung und Anwendung, auswendig kennen sollte
  - Daneben gibt es etliche Operatoren (z.B. +, -, *, /, =) mit fester Bedeutung
- Die Namen von Funktionen und Variablen sind selbstgewählt
  - Enthalten nur Buchstaben aus ASCII-Zeichensatz (keine Umlaute), Ziffern und Unterstrich (keine sonstigen Sonderzeichen oder Leerzeichen)
    - Ziffern dürfen aber nicht das erste Zeichen eines Bezeichners sein
    - Groß- und Kleinschreibung werden unterschieden
  - Dürfen nicht mit Schlüsselwörtern übereinstimmen!
- Kommentare
  - Ergänzungen im Quelltext, die dazu dienen, selbigen möglichst verständlich zu gestalten
  - Werden durch spezielle Zeichenfolge eingeleitet und vom Compiler nicht übersetzt

Schlüsselwörter in C

Hochschule Fulda
University of Applied Sciences

auto	do	~~goto~~	return	typedef
break	double	if	short	union
case	else	inline	signed	unsigned
char	enum	int	sizeof	void
const	extern	long	static	volatile
continue	float	register	struct	while
default	for	restrict	switch	_Bool (*)

Grau gefärbte Schlüsselwörter sind z.T. seltener gebräuchlich und nicht klausurrelevant

Blau gefärbte Schlüsselwörter bezeichnen "nur" verschiedene Datentypen (z.B. int)

(*) Inkludiert man oben in der C-Datei noch *stdbool.h* via

```
#include <stdbool.h>
```

bekommt man noch folgende "Pseudo-Schlüsselwörter":

```
bool, true, false
```

Variablen

Hochschule Fulda
University of Applied Sciences

```
int i = 23, j;
int zahl = 42;
```

Variablen-bezeichner	Datentyp	Speicher-adresse	Speicher-inhalt
i	int	48000	23
j	int	48004	?
zahl	int	48008	42

- Dienen dazu Daten abzuspeichern
  - An jeweiliger Adresse im Hauptspeicher
- Müssen vor Gebrauch eingeführt, d.h. deklariert, werden
  - Im Beispiel oben werden drei Variablen mit Namen *i*, *j* und *zahl* **deklariert**
  - *i* und *zahl* werden mit Werten 5 bzw. 42 **initialisiert** (d.h. mit Startwert versehen)
  - *j* wurde nicht initialisiert, ist daher in undefiniertem Zustand (Wert ist 'Speichermüll')
- Schlüsselwort `int` besagt, dass es sich um ganze Zahlen handelt
  - Der Datentyp der Variablen heißt damit `int`
- Am Ende einer Deklaration steht ein Semikolon

Zuweisungen

Hochschule Fulda
University of Applied Sciences

- Enden auch mit Semikolon
- Modifizieren Werte von Variablen
- Beispiel 1: `zahl = 42;`
  - Variable *zahl* erhält den Wert 42
  - Am der Variablen zugehörigen Speicherplatz liegt 42 (als Bitmuster: 00101010)

Variablen-bezeichner	Datentyp	Speicher-adresse	Speicher-inhalt
i	int	48000	23
j	int	48004	66
zahl	int	48008	42

- Beispiel 2: `j = i + zahl;  // i hat Wert 23`
  - Werte von *i* und *zahl* werden ermittelt, addiert und der Variablen *j* zugewiesen
- Beispiel 3: `j = j + 1;  // erhoehe j um 1`
  - In dieser Zuweisung greift *j* auf der rechten Seite des Zuweisungsoperators auf Wert vor der Zuweisung zu, worauf 1 addiert wird (→ damit hat *j* den Wert 66)

Variablen-Initialisierung

Hochschule Fulda
University of Applied Sciences

```
int n1, n2, n3;


n1 = 5;
n2 = n1 * n1;
n3 = n2 * n2;
```

Wert der Variablen

n1	n2	n3
?	?	?
5	?	?
5	25	?
5	25	625

Nun haben alle Variablen **definierten** Wert

Vorsicht: in C enthalten nicht initialisierte Variablen i.d.R. "Speichermüll", d.h. irgendwelche zufällige Werte (in Tabelle mit "?" gekennzeichnet)

Arithmetische Ausdrücke

Hochschule Fulda
University of Applied Sciences

- Auf der rechten Seite von Zuweisungen können Ausdrücke stehen
  - Konstanten (Werte, die keine Variablen sind, z.B. 4711)
  - Zwei Ausdrücke mit einem Operator
    - +, -, * wie üblich
    - Vorsicht bei Division /
      - Sind beide Werte (oder Variablen) ganzzahlig (Datentyp `int`), handelt es sich um eine ganzzahlige Division (Nachkommateil bleibt unberücksichtigt)
      - Wird Divisionsergebnis Integer-Variablen zugewiesen, ist es auch ganzzahlige Division
    - Modulo-Operation %
      - Rest bei ganzzahliger Division
  - Geklammerter Ausdruck, z.B.: 2 * (17 + 4)
- Beispiel: `zahl = 2 * (17 + 4) - 7 / 3;`

Beispielprogramm

Hochschule Fulda
University of Applied Sciences

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

int main()
{
  int i = 23, j;
  int zahl = 42;

  j = i + zahl;  //i hat Wert 23, zahl hat Wert 42
  j = j + 1;     //erhoehe danach j nochmal um 1
  printf("j hat Wert %d\n", j);

  //Ausdruck auf rechter Seite steht für einen Wert
  zahl = 2 * (17 + 4) - 7 / 3;
  printf("zahl hat Wert %d\n", zahl);

  return 0;
}
```

Sog. Headerdateien wie *stdio.h* werden am Kopf des Quelltexts eingefügt und beim Kompilieren eingebunden.
Sie enthalten zusätzliche Anweisungen wie z.B. Ein- u. Ausgabefunktionen aus der C-Standard-Input/Output-Bibliothek

- Funktion *printf()* schreibt Argument(e) auf Konsole
  - Erstes Argument ist immer Zeichenkette, optional sind weitere Argumente möglich
    - Durch Komma getrennt
  - Die weiteren Argumente sind Ausdrücke, deren Wert man ausgeben möchte
- Erlaubt formatierte *Ausgabe*
  - Platzhalter %d steht für ganze Zahl (→ int) als Dezimalzahl
    - %x für Hexadezimalzahl
  - Für jeden weiteren Ausdruck ist je ein Platzhalter nötig

Sequenz

Hochschule Fulda
University of Applied Sciences

- Jede Zeile im Rumpf des Hauptprogrammes ist eine Anweisung
  - *printf(…)* ist ein Funktionsaufruf, der stets mit einem Semikolon beendet wird
- Es können mehrere Zuweisungen, Deklarationen oder Ausgaben hintereinander stehen – sind damit eine Sequenz
- Zu jedem Zeitpunkt wird nur genau eine Anweisung ausgeführt
  - Und jede nur genau einmal
- Reihenfolge wie im Quelltext angegeben
- Nach der letzten Anweisung endet das Programm
  - Im Beispiel ist das die Zeile: `return 0;`
  - Danach endet die *main()* Funktion (mit schließender geschweifter Klammer)

Vielen Dank!

Noch Fragen?