# Imperative Programmierung

## Übung 4: Programmieren in C

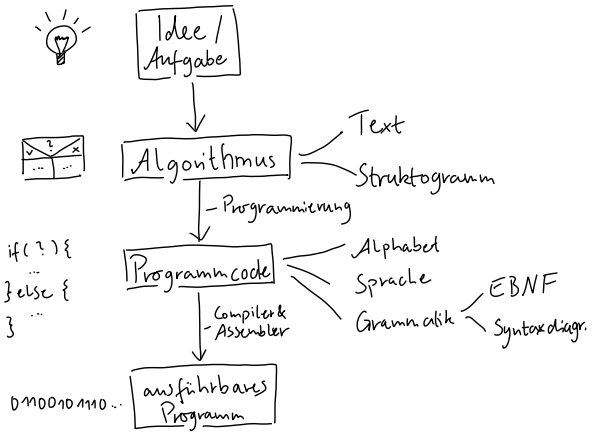
**Justin Kreikemeyer**
Informatik, Uni Rostock

Fragen?

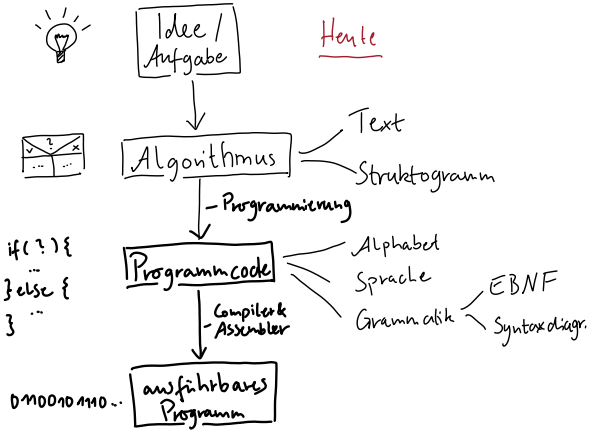
## Leitfragen

- Was sind die Grundkonzepte der Sprache C?
- Wie kommt man mithilfe eines Compilers zu einem ausführbaren Programm?

# Von der Idee zum ausführbaren Programm



# Von der Idee zum ausführbaren Programm



## Empfehlung

- Programmieren lernt man nur durch selber Programmieren!
- Löst auch nach der Übung weiter so viele Programmieraufgaben, wie irgendwie möglich ($\rightarrow$ Praktikum/Selbststudium)
- Hier: Überblick (sehr viel auf einmal!), Lösen einiger Aufgaben, Klärung von Fragen

## Grundstruktur

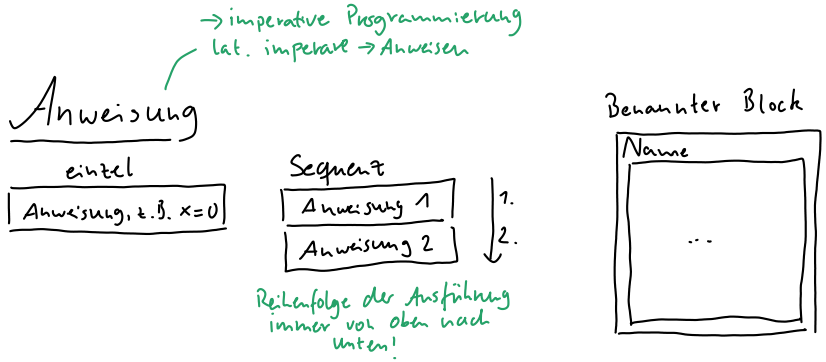
```c
#include <stdio.h>                    // Inklusion von externem Quellcode
int main(int argc, char* argv[]) {    // Anfangspunkt & Eingaben vom Betriebssystem

  // Programmcode steht hier.         // Zwei Schrägstriche (//) kennzeichnen Kommentar

  return 0;                           // RÜckgabewert an das Betriebsystem (0 = Erfolg)
}
```

- Beobachtung: Farbliche Markierung von speziellen Wörtern in C (je nach Editor anders)
- `#include <stdio.h>`: Einbinden des Quelltextes aus der Datei `stdio.h` (beinhaltet Prozeduren zur Ein- und Ausgabe)
- `argc, argv` Anzahl und Werte der Kommandozeilenargumente (kommt noch)
- Warum das alles so aussieht und was genau `int main(...)` bedeutet klärt sich im Laufe des Semesters

# Erinnerung: Grundbausteine




In C: Variablendeklaration, Ein-/Ausgabe, Arithmetik (Zahlen, Logik)

# Deklaration: Variablen & Typen

- Variable = Benennung eines Speicherbereichs
- Typ = Wie ist der Inhalt des Speicherbereichs zu interpretieren?
- `<typ> <name>;`, z.B. `int _myVariable1;`
- `<name>` darf nicht mit Zahl beginnen! (weitere Restriktionen s. VL/Buch/...)
- Zuweisung eines Wertes mit dem Zuweisungsoperator =

```
// ganze Zahlen
int i = 42;
long l = -123;
long long ll = 123456;
// Gleitkommazahlen ("reelle Zahlen")
float f = 0.1f;   // f kann auch weggelassen werden
double d = 1e-10; // Exponentialschreibweise
// Zeichen
char c = 'a';     // ganze Zahl, aber interpretiert als ASCII-Code
// Zuweisung nach Deklaration
i = 43;
```

# Eingabe und Ausgabe

- `printf(format-string, belegungen...);`
- `scanf(format-string, &variable1, &variable2, ...);`
- `name` ≡ “Lesezugriff”
- `&name` ≡ Referenz auf Speicherbereich (“Schreibzugriff”)

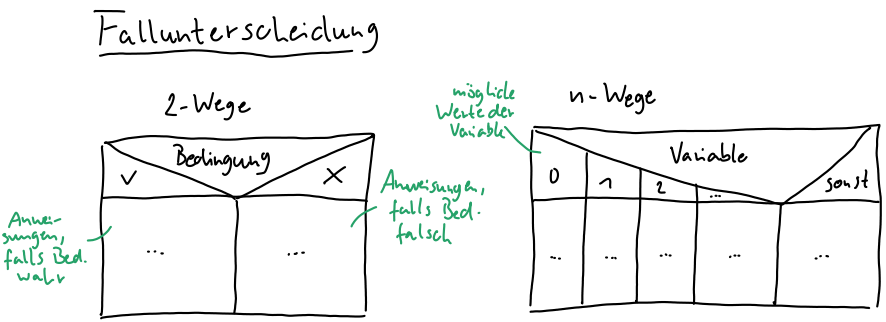
```
int i = 0;
double d = 0.2;
printf("abc");              // Ausgabe Zeichenkette (String)
printf("\non new line");    // Zeilemumbruch mit Sonderstring \n
printf("test: %d", i);      // Ausgabe ganze Zahl
printf("test: %lf", d);     // Ausgabe Gleitkommazahl
scanf("%d", &i);            // Eingabe ganze Zahl
```

## Arithmetik

- Wie in der Mathematik (inkl. Präzedenz und Klammerung)
- Arithmetik: Addition (+), Subtraktion (-), Multiplikation (*), Division (/), Modulo/Restwert (%)
- Vergleiche: < > <= >=, Ungleich (!=) und Gleich (==, nicht mit Zuweisung verwechseln!)
- Logik: Negation (!), Konjunktion (&&), Disjunktion (||)
- 0 ≡ false, 1 ≡ true

```
int a = 42;
int b = 112;
int c = a == b;                    // Wert von c?
int d = a * 3 - b;
printf("%d", (a == 42) || (b != 0)); //Ausgabe?
```

# Erinnerung: Grundbausteine




In C: `if (Bedingung) {...} else {...}`

# Kontrollstrukturen: Verzweigung

s. letzte Übung. Beispiel:

```
int a;
scanf("%d", a);
if (a == 0) {
  // Fall a gleich 0
  printf("1");
} else {
  // Fall a ungleich 0
  printf("0");
}
```

Wo ist der Fehler im obigen Programm?

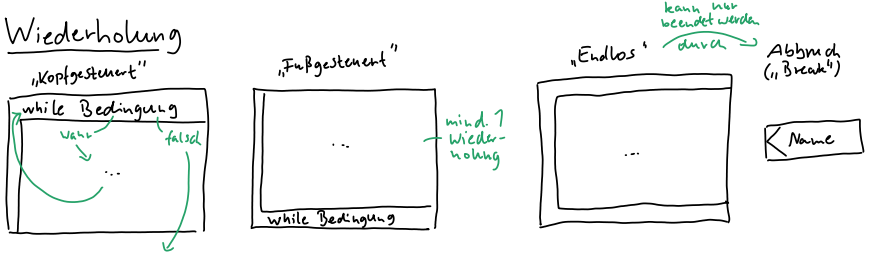
# Kontrollstrukturen: Verzweigung

s. letzte Übung. Beispiel:

```
int a;
scanf("%d", a);
if (a == 0) {
  // Fall a gleich 0
  printf("1");
} else {
  // Fall a ungleich 0
  printf("0");
}
```

Wo ist der Fehler im obigen Programm? → `scanf(..., &a);`! Was ist dann die Ausgabe?

# Erinnerung: Grundbausteine




In C: `while (Bedingung) {...}` oder `do {...} while (Bedingung);`

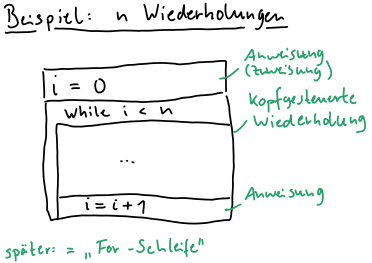
# Kontrollstrukturen: Schleife

s. letzte Übung. Beispiel:

```
int i = 0;
int n;
scanf("%d", &n);
while (i < n) {
  printf("hallo!\n");
  i = i + 1;              // Abkürzung mit: i++;
}
```

Was ist die Ausgabe des obigen Programms?

# Erinnerung: Grundbausteine




In C: `for (int i = 0; i < n; i++) {...}` (neu!)

# Abkürzung: For-Schleife

Geht das auch einfacher? Ja, mit einer For-Schleife!

```
int n;
scanf("%d", &n);
for (int i = 0; i < n; i++) {  // for (Initialisierung; Bedingung; Veränderung) {...}
  printf("hallo!\n");
}
```

For-Schleife, falls Anzahl Iterationen bekannt
While-Schleife (oder do-while), falls nur Bedingung bekannt

## Gemeinsame Beispiele

Na gut, alles klar. Wie führe ich mein Programm jetzt aus?

1. Kompilieren: `gcc -pedantic -Wall -Werror -std=c99 prog.c -o prog`
   Oder Playbutton in VS Code (erfahrungsgemäß unzuverlässig)
2. Ausführen: `./prog`

Hinweis: unter Windows ggf. `.\prog`

**Heaviside** Lese eine Zahl `x` ein und berechne folgende Funktion:

$$f(\mathtt{x}) = \begin{cases} 1, \text{if} x < 0 \\ 0, \text{otherwise} \end{cases}$$

**MultTable** Gebe die Multiplikationstabelle der Zahlen von 0 bis 10 aus, e.g., `0x0=0`, `0x1=0`,...,`10x10=100`.

## Wo finde ich Hilfe?

- Cheatsheet s. Stud.IP/Praktikum
- Leseprobe vom Rheinwerk Verlag
- Originalbuch von Kernighan und Ritchie “The C Programming Language” (1978)
- . . . (nicht vollständig)

# Aufgaben: Programmieren mit C

Schreiben Sie die folgenden C-Programme[1]! Nutzen Sie dazu die Konzepte aus dieser Übung und das Cheat Sheet!
**Bearbeitungszeit: bis 10 Minuten vor Schluss. Dann Besprechung von häufigen Problemen.**

[1] Weitere Aufgaben können jederzeit beim Übungsleiter erfragt werden.

# Programme I

**Modulo** Lesen Sie zwei Zahlen ein. Falls die Summe der Zahlen gerade ist, geben Sie ihr Produkt aus, ansonsten ihre Differenz.

**Range** Schreiben Sie ein Programm, dass zwei ganze Zahlen einliest und alle ganzen Zahlen dazwischen (inklusive Grenzen) ausgibt.

**Min** Schreiben Sie ein Programm, dass drei Gleitkommazahlen einliest, das minimum bestimmt und dieses Auf dem Bildschirm ausgibt. Bonus: Ändern Sie das Programm so ab, dass auch die Nummer des Minimums (1. Zahl, 2. Zahl oder 3. Zahl) mit ausgegeben wird!

## Programme II

MultTable2 Modifizieren sie MutTable so, dass die Ausgabe tatsächlich in Tabellenform erfolgt.

Power Lesen Sie zwei Zahlen `x` und `y` ein und berechnen Sie $x^y$. Falls die Eingabe $y \leq 0$ ist, endet das Programm mit Rückgabewert 1 (“nicht OK”).

## Programme III

**FizzBuzz** Gehe der Reihe nach die Zahlen von 1 bis 100 durch. Falls die aktuelle Zahl durch drei teilbar ist, gebe “Fizz” aus. Falls die aktuelle Zahl durch 5 teilbar ist, gebe “Buzz” aus. Falls die aktuelle Zahl durch 3 *und* durch 5 teilbar ist, gebe “FizzBuzz” aus. Falls keine der vorigen Bedingungen zutrifft, gebe die Zahl aus.

**Fibonacci** Die Fibonacci Reihe ist wie folgt definiert $fib(n+1) = fib(n) + fib(n-1)$ mit $fib(0) = 1$ und $fib(1) = 2$. Die nächste Zahl der Folge ergibt sich also durch die Summe der beiden vorherigen Zahlen. Lese eine Zahl $n$ ein und geben die $n$-te Fibonacci-Zahl $fib(n)$ aus!