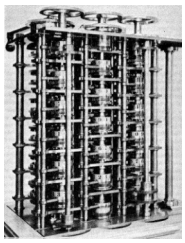
Grundlagen der Programmierung in C
Funktionen

Wintersemester 2005/2006
G. Zachmann
Clausthal University, Germany
zach@in.tu-clausthal.de

- Der erste Mechanismus für *Code-Reuse* !
- Ältester Mechanismus für Code-Reuse:
  Every set of cards made for any formula
  will at any future time recalculate that formula
  with whatever constants may be required.
  Thus the Analytical Engine will possess
  a library of its own. (Charles Babbage, 1864)



- Funktion = parametrisierter Block von Anweisungen
- Library = Sammlung vieler Funktionen zu einem Thema:
  - C-Library (stdio, pthreads, …)
  - Multimedia, Numerik, Datenbank, …

Terminologie

- Analogie:
  - Variablen werden *deklariert*, *belegt* und *verwendet* (=gelesen)
  - Funktionen werden *deklariert*, *definiert*, und *aufgerufen*
- *Deklaration* :=
  Bekanntgabe an Compiler über Existenz und I/O-Verhalten einer Funktion.

Deklaration von Funktionen

- Syntax der Deklaration:

$$T_0\ F(\ T_1\ P_1,\ T_2\ P_2,\ \ldots\ );$$

  - T_0 = Typ des Rückgabewertes der Funktion ("Typ der Funktion")
  - F = Name der Funktion
  - P_i = *formale Parameter* (Variablen innerhalb der Funktion)
  - T_i = Typen der formalen Parameter
  - Das Ganze heißt Prototyp (Mathematiker sagen "Signatur")
- Beispiele:

```
int foo( int x );
void bar();
float evalCubic( float a0, float a1,
                 float a2, float a3 );
```

- Style guidelines:

```
float sampleSignal( int k, int n,
                    char s );
```

Namenskonvention:
kleinGroßGroß (*inCaps*)
verbSubstantiv

Parameter vertikal ausrichten,
falls nicht auf eine Zeile passend

Rückgabetyp auf eigene Zeile,
falls dieser lang wird

```
unsigned long int *
sampleSignal( … );
```


Definition von Funktionen

- Definition:

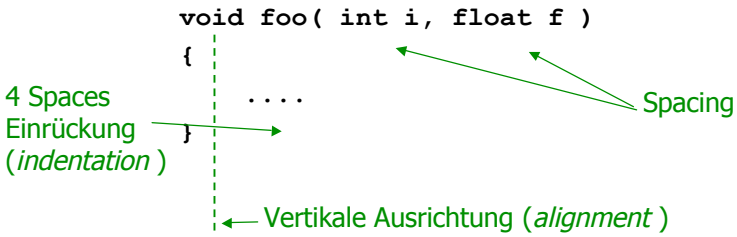
```
T0 F( T1 P1, T2 P2, … )
{
        …
}
```

  - Innerhalb der `{ }` stehen die Anweisungen (*function body*)
  - Formale Parameter wie andere Variablen innerhalb Body
  - Gelten nur innerhalb Body!
  - Verwende **`return x`**, um Funktion zu beenden und Funktionswert vom Typ T_0 zurückzuliefern
  - Falls T_0 = **`void`**, verwende **`return`** ohne Argument (nicht nötig, falls letzer Befehl vor `}`)

- Beispiel:

```
bool sign( int x )
{
    if ( x >= 0 )
       return true;
    else
       return false;
}
```

- Style Guidelines:



- Was in Funktionen *nicht* geht
  (aber manchmal [selten] "nice to have" wäre):
  - Funktion innerhalb einer Fkt definieren (lokale Fkt à la Pascal)
  - Mehrere Rückgabewerte

Aufruf von Funktionen

- Aufruf ("function call"):

$$F(P_1, P_2, \ldots)$$

  - P_i heißen (tatsächliche) Parameter (*actual parameters*)
  - Werden in die formalen Parameter der Fkt.-Definition kopiert
  - Typen müssen mit formalen Parametern übereinstimmen (oder konvertiert werden können)
  - Achtung: sieht *fast* aus wie Deklaration!
  - Kann überall stehen, wo Ausdruck stehen kann
  - Achtung: Funktion muß vor Aufruf mindestens deklariert sein (oder definiert)

- Achtung: Aufruf sieht *fast* aus wie Deklaration!
- Beispiel:

```
void foo();
foo();
```

```
int bar( int x );
bar(x);
```

- Beispiele:

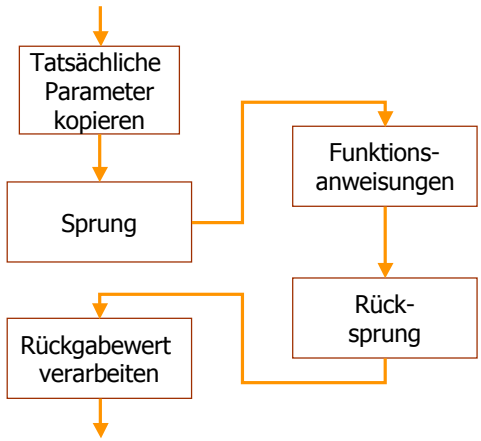
```
puts("hi");
x = sign( y-2 );
if ( sign(x) ) … ;
float samples = evalCubic( 1, 2.0, 3, sign(x) );
```

- Achtung: folgendes ist kein Funktionsaufruf!

```
void foo( void )
{
    ...
}

foo;
```

- Semantik (*flow control*):



Rekursion

- Funktion, die sich selbst aufruft, heißt *rekursive Fkt*
- Beispiel Fibonacci-Zahlen:
  - Rekurrenz-Relation:

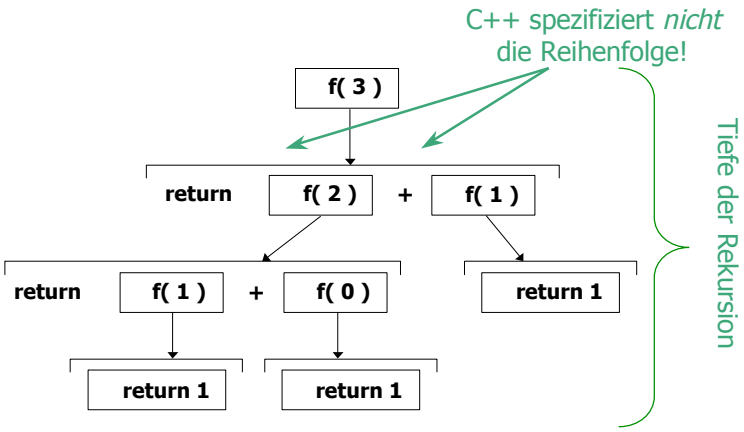
    $F_n = F_{n-1} + F_{n-2}$

    $F_1 = F_0 = 1$
  - Programm:

```
unsigned int f( unsigned int n )
{
    if ( n > 1 )
        // complex case
        return f(n-1) + f(n-2);
    else
        // base case
        return 1;
}
```

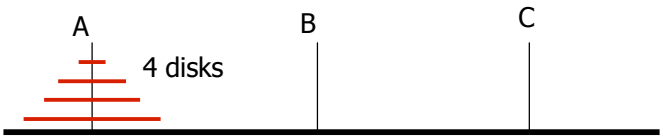

Rekursionsbaum

- Ausführungsbeispiel:



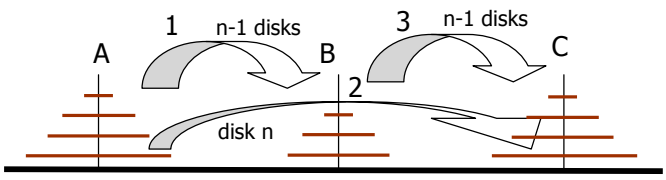
Komplizierteres Beispiel: Türme von Hanoi

- Puzzle, bestehend aus 3 Stäben und Scheiben verschiedener Größe
- Ausgangssituation:



- Ziel: alle Scheiben von A nach B (oder C) schaffen
- Regeln:
  - Nur 1 Scheibe pro Zug
  - Eine größere Scheibe darf nie auf kleinerer Scheibe liegen

- Idee:
  - Angenommen, wir wüßten, wie man n-1 Scheiben verschiebt, dann ...



→Rekursion

- Turm mit 3 Scheiben

- Turm mit 4 Scheiben