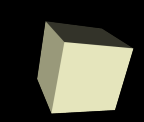
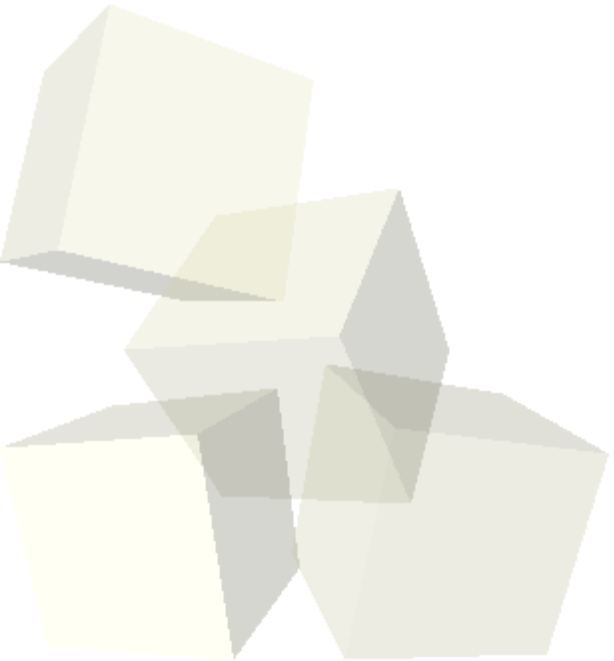
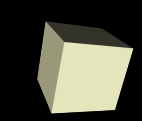


C-Schulung Tag 1

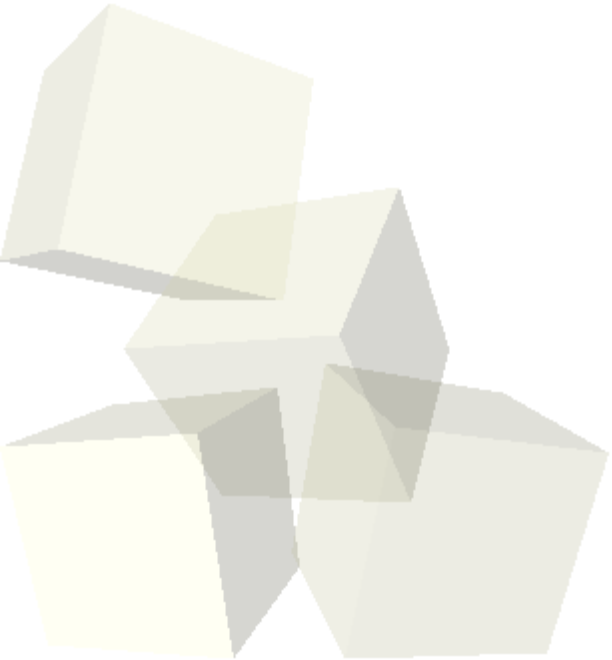
Herzlich willkommen





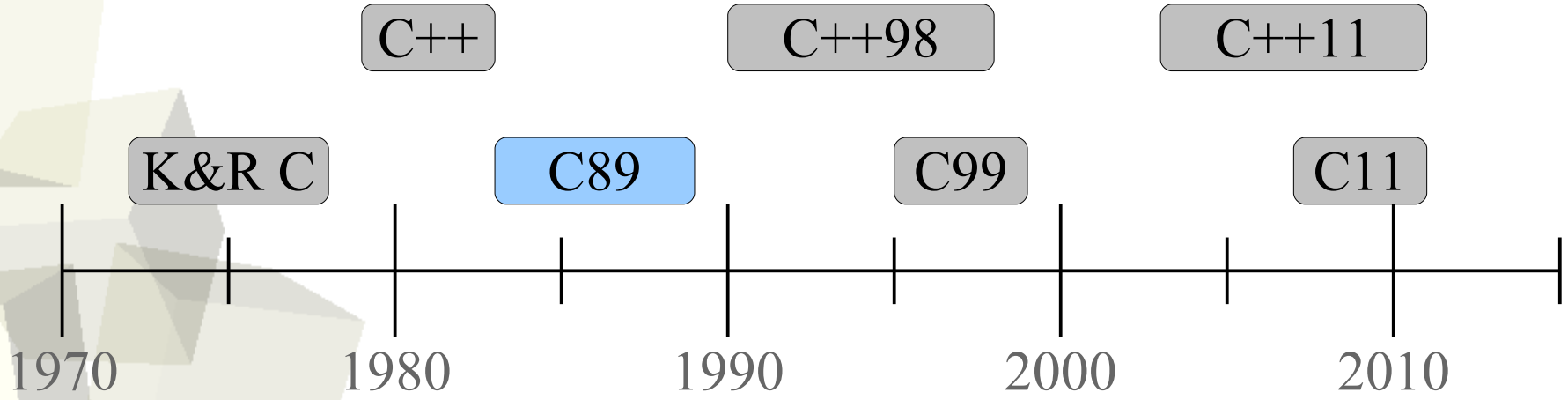
Agenda

- Einführung
- Ein- und Ausgabe
- Arrays
- Zeichenketten



Geschichte

- 1972: K&R C
- 1978: The C Programming Language
- 1988: The C Programming Language, 2nd edition
- 1989: C89 (ANSI)
- 1990: C90 (ISO)
- 1999: C99
- 2011: C11



Welche Rolle spielt C heutzutage?

- C ist eine kleine Sprache. Es ist relativ leicht...
  - ...einen Parser für C zu implementieren.
  - ...den kompletten Sprachumfang zu lernen.
- Systemnahe Programmierung:
  - Linux Kernel, git
  - Python, Perl, PHP
  - MySQL, PostgreSQL, SQLite
  - Doom 1-2, Quake 1-3
- Zwischensprache beim Übersetzen:
  - C with classes
  - Eiffel
- Viele Bibliotheken definieren eine C-Schnittstelle:
  - OpenGL (Open Graphics Library)
  - SDL (Simple DirectMedia Layer)
  - SFML (Simple and Fast Multimedia Library)

Goodbye Java

```
package fitness;      // Es gibt keine Pakete in C.
import java.util.*;
// Klassen  Vererbung
class Waage extends Object
{// Kapselung  Sammlungen  Generizität
    private    List         <Integer>   _messungen;
    // Konstruktoren
    public Waage()
    {              // GC  Subtyp-Polymorphie
        _messungen = new ArrayList<Integer>();
    }
    // Methoden
    public void registriere(int neuesGewicht)
    {             // Dynamisches Binden
        _messungen.add(neuesGewicht);
    }

    public int letztesGewicht()
    {                          // ArrayIndexOutOfBoundsException
        return _messungen.get(0);
    }
}
```

C als portable Assemblersprache

SPARC V8

```
checksum:
    or %g0, %g0, %o2
.addnumbers:
    ld [%o0], %o3
    subcc %o1, 1, %o1
    add %o2, %o3, %o2
    bne .addnumbers
    add %o0, 4, %o0

    jmp %o7 + 8
    or %g0, %o2, %o0
```

Intel IA-32

```
_checksum:
    mov esi, [esp + 4]
    mov ecx, [esp + 8]
    xor edx, edx
addnumbers:
    lodsd
    add edx, eax
    loop addnumbers

    mov eax, edx
    ret
```

C

```
int checksum(int * p, int n)
{
    int sum = 0;
    while (n--)
        sum += *p++;
    return sum;
}
```

C als portable Assemblersprache

- C abstrahiert von vielen Architekturdetails:
  - Speicherzellen werden über Variablen angesprochen.
  - Die Nutzung von Registern geschieht transparent.
  - Kontrollstrukturen statt bedingter Sprünge
  - Aufrufkonventionen werden automatisch eingehalten.
  - ...
- Einige Architekturdetails scheinen weiterhin durch:
  - Der Datentyp int hat die Wortbreite der Maschine.
  - Das Verhalten von vorzeichenbehafteten Datentypen ist an einigen Stellen implementationsabhängig.
  - Bei Zeigercasts kann Endianness eine Rolle spielen.
  - ...

Zeichen ausgeben

```
#include <stdio.h>

int main(void)
{
    putchar(104);
    putchar(101);
    putchar(108);
    putchar(108);
    putchar(111);
    putchar(32);
    putchar(119);
    putchar(111);
    putchar(114);
    putchar(108);
    putchar(100);
    putchar(33);
    putchar(10);

    return 0;
}
```

Zeichen-Literale

```
#include <stdio.h>

int main(void)
{
    putchar('h');
    putchar('e');
    putchar('l');
    putchar('l');
    putchar('o');
    putchar(' ');
    putchar('w');
    putchar('o');
    putchar('r');
    putchar('l');
    putchar('d');
    putchar('!');
    putchar('\n');

    return 0;
}
```

ASCII-Tabelle

```c
#include <stdio.h>

int main(void)
{
    putchar('h');
    putchar('e');
    putchar('l');
    putchar('l');
    putchar('o');
    putchar(' ');
    putchar('w');
    putchar('o');
    putchar('r');
    putchar('l');
    putchar('d');
    putchar('!');
    putchar('\n');

    return 0;
}
```

32:	56: 8	80: P	104: h
33: !	57: 9	81: Q	105: i
34: "	58: :	82: R	106: j
35: #	59: ;	83: S	107: k
36: $	60: <	84: T	108: l
37: %	61: =	85: U	109: m
38: &	62: >	86: V	110: n
39: '	63: ?	87: W	111: o
40: (	64: @	88: X	112: p
41:)	65: A	89: Y	113: q
42: *	66: B	90: Z	114: r
43: +	67: C	91: [	115: s
44: ,	68: D	92: \	116: t
45: -	69: E	93:]	117: u
46: .	70: F	94: ^	118: v
47: /	71: G	95: _	119: w
48: 0	72: H	96: `	120: x
49: 1	73: I	97: a	121: y
50: 2	74: J	98: b	122: z
51: 3	75: K	99: c	123: {
52: 4	76: L	100: d	124: |
53: 5	77: M	101: e	125: }
54: 6	78: N	102: f	126: ~
55: 7	79: O	103: g	127: ⌂

ASCII-Tabelle generieren

```c
#include <stdio.h>

int main(void)
{
    int x, y;   /* Deklarationen zuerst! */
    for (y = 32; y < 56; ++y)
    {
        for (x = 0; x < 4; ++x)
        {
            int z = y + 24*x;
            /* Zahl und Zeichen ausgeben */
        }
        putchar('\n');
    }
    return 0;
}
```

```
32:       56: 8     80: P    104: h
33: !     57: 9     81: Q    105: i
34: "     58: :     82: R    106: j
35: #     59: ;     83: S    107: k
36: $     60: <     84: T    108: l
37: %     61: =     85: U    109: m
38: &     62: >     86: V    110: n
39: '     63: ?     87: W    111: o
40: (     64: @     88: X    112: p
41: )     65: A     89: Y    113: q
42: *     66: B     90: Z    114: r
43: +     67: C     91: [    115: s
44: ,     68: D     92: \    116: t
45: -     69: E     93: ]    117: u
46: .     70: F     94: ^    118: v
47: /     71: G     95: _    119: w
48: 0     72: H     96: `    120: x
49: 1     73: I     97: a    121: y
50: 2     74: J     98: b    122: z
51: 3     75: K     99: c    123: {
52: 4     76: L    100: d    124: |
53: 5     77: M    101: e    125: }
54: 6     78: N    102: f    126: ~
55: 7     79: O    103: g    127: ⌂
```

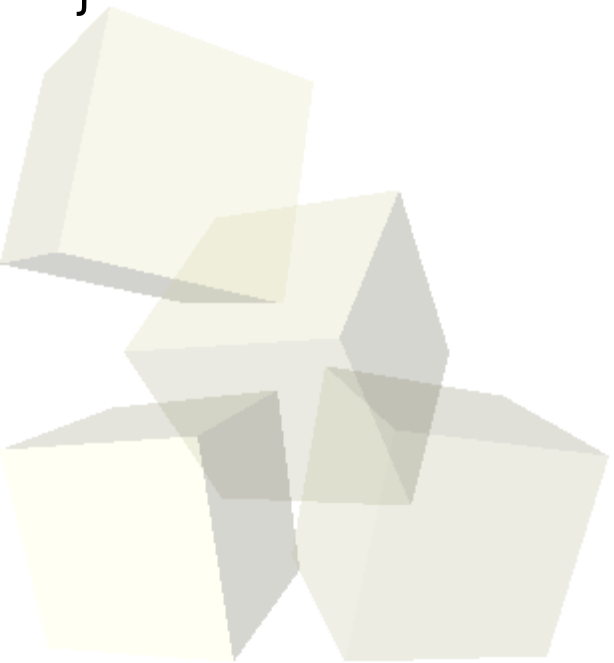
ASCII-Tabelle generieren

```c
#include <stdio.h>

int main(void)
{
    int x, y;
    for (y = 32; y < 56; ++y)
    {
        for (x = 0; x < 4; ++x)
        {
            int z = y + 24*x;
            printf("%3d: %c   ", z, z);
        }
        putchar('\n');
    }
    return 0;
}
```

```
 32:        56: 8      80: P     104: h
 33: !      57: 9      81: Q     105: i
 34: "      58: :      82: R     106: j
 35: #      59: ;      83: S     107: k
 36: $      60: <      84: T     108: l
 37: %      61: =      85: U     109: m
 38: &      62: >      86: V     110: n
 39: '      63: ?      87: W     111: o
 40: (      64: @      88: X     112: p
 41: )      65: A      89: Y     113: q
 42: *      66: B      90: Z     114: r
 43: +      67: C      91: [     115: s
 44: ,      68: D      92: \     116: t
 45: -      69: E      93: ]     117: u
 46: .      70: F      94: ^     118: v
 47: /      71: G      95: _     119: w
 48: 0      72: H      96: `     120: x
 49: 1      73: I      97: a     121: y
 50: 2      74: J      98: b     122: z
 51: 3      75: K      99: c     123: {
 52: 4      76: L     100: d     124: |
 53: 5      77: M     101: e     125: }
 54: 6      78: N     102: f     126: ~
 55: 7      79: O     103: g     127: ⌂
```

String-Literale

```
#include <stdio.h>

int main(void)
{
    puts("hello world!");

    printf("hello world!\n");

    printf("%s\n", "hello world!");

    return 0;
}
```



Ein- und Ausgabe

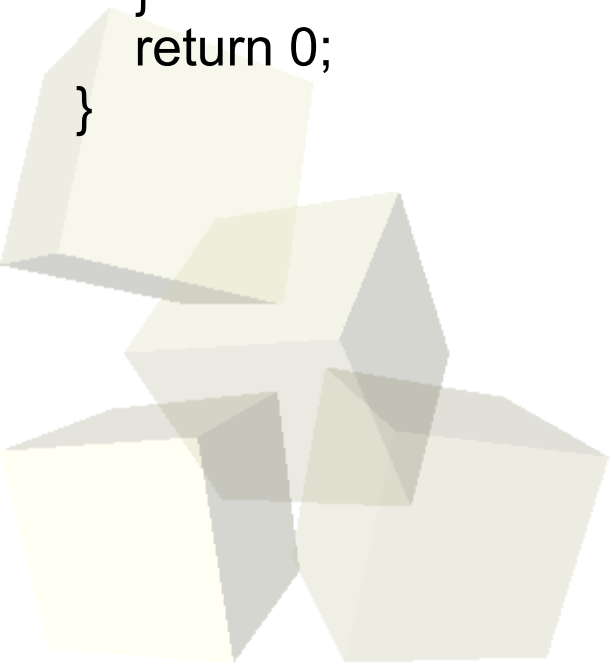
```
#include <stdio.h>

int main(void)
{
    int c;

    c = getchar();
    while (c != EOF)
    {
        putchar(c);
        c = getchar();
    }
    return 0;
}
```

Ein- und Ausgabe

```
#include <stdio.h>

int main(void)
{
    int c;


    while ((c = getchar()) != EOF)
    {
        putchar(c);

    }
    return 0;
}
```

Ein- und Ausgabe

```
#include <stdio.h>

int main(void)
{
    int c;


    while ((c = getchar()) != EOF)
    {
        if (c >= '0' && c <= '9') c = 'X';
        putchar(c);
    }
    return 0;
}
```



Umleiten der Standardströme

Eingabe von Tastatur, Ausgabe auf Bildschirm

```
C:\>ech
```

Eingabe aus Datei, Ausgabe auf Bildschirm

```
C:\>ech <readme.txt
```

Eingabe von Tastatur, Ausgabe in Datei

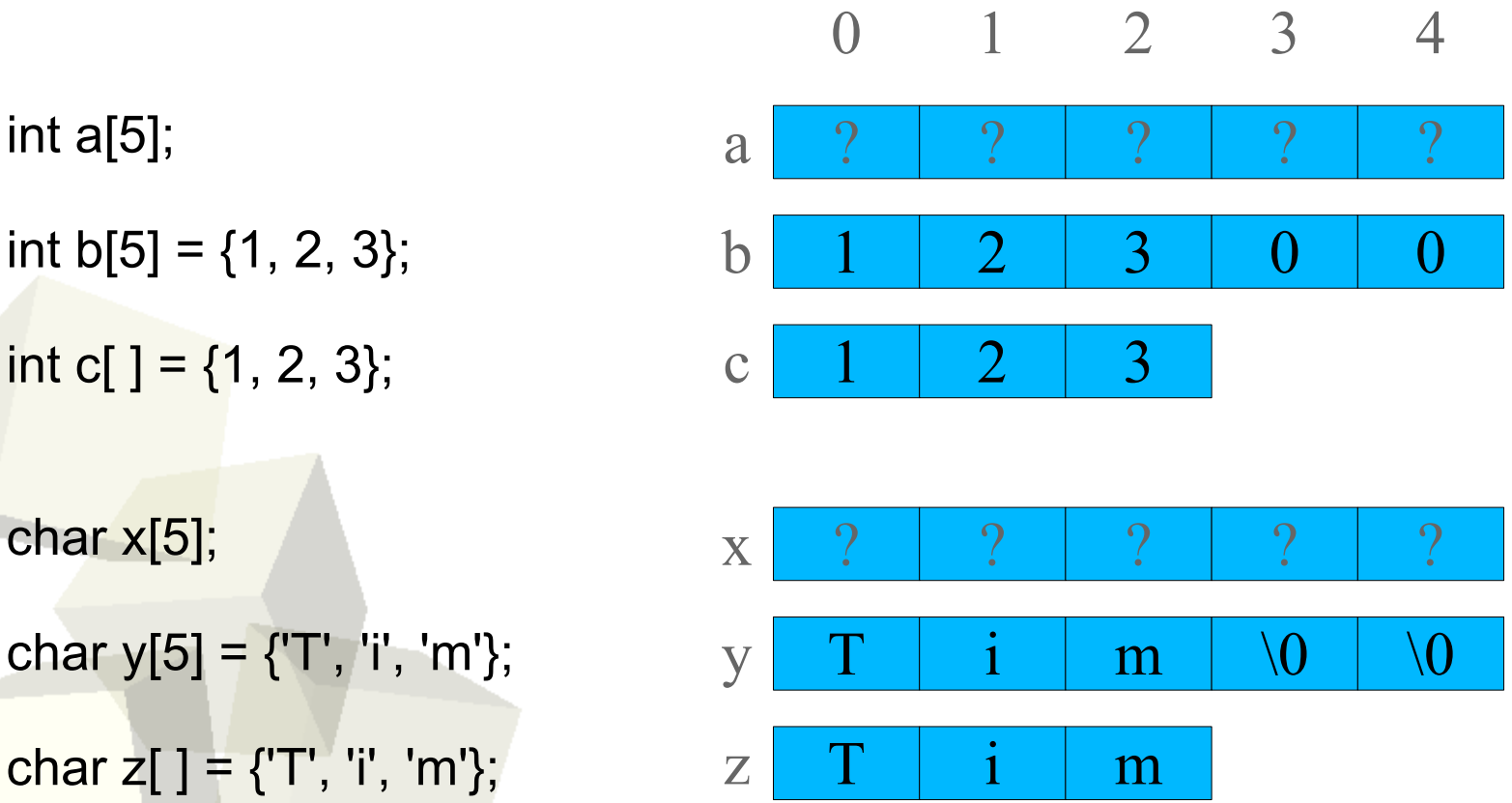
```
C:\>ech >log.txt
```

Eingabe aus Datei, Ausgabe in Datei

```
C:\>ech <readme.txt >log.txt
```

Arrays

- Ein Array ist eine Aneinanderreihung von n anonymen Objekten mit den Indizes 0 bis n-1
  - n ist kein gültiger Index
  - Es findet keinerlei Indexprüfung zur Laufzeit statt!



Terminologie: Objekte

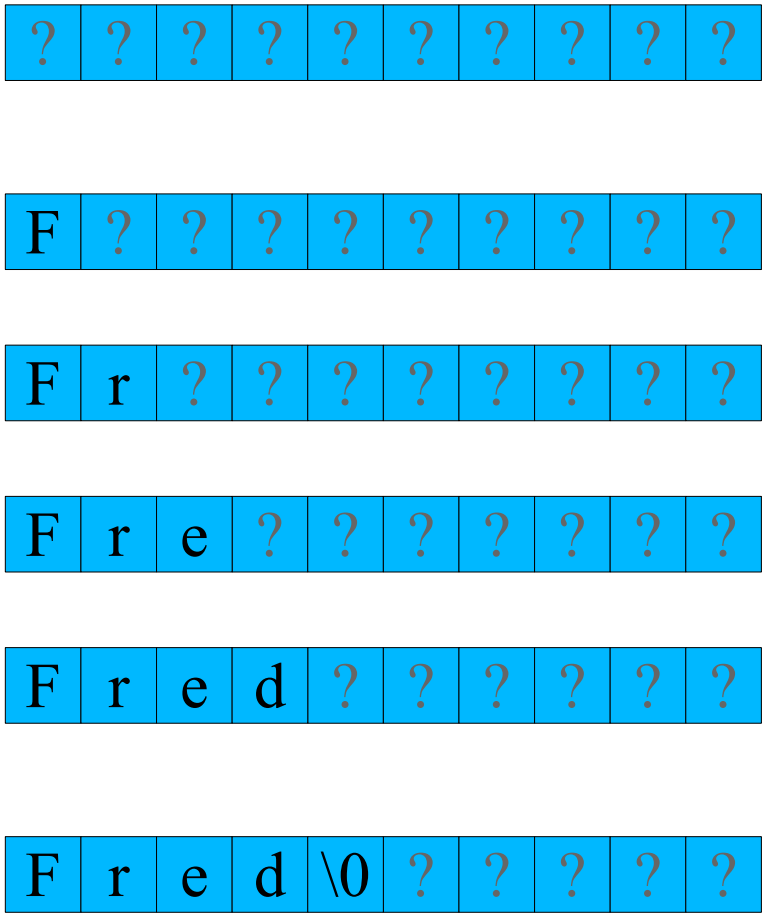
- In objektorientierten Programmiersprachen spielt der Begriff “Objekt” eine zentrale Rolle:
  - Kapselung
  - Vererbung
  - Polymorphie
  - …
- In C ist ein Objekt dagegen ein Speicherbereich.
  - “An *object* is a region of storage that can be examined and stored into.”
- So ist zum Beispiel jede Variable in C ein Objekt.

Zeichenketten

```
#include <stdio.h>

int main(void)
{
    char name[10];
    int p = 0, c;

    printf("Wie lautet Ihr Name? ");
    while ((c = getchar()) != '\n' && p < 9)
    {
        name[p] = c;
        p = p + 1;
    }
    name[p] = '\0';

    printf("Guten Tag, %s!\n", name);
    return 0;
}
```

Zeichenketten

```
#include <stdio.h>

int main(void)
{
    char name[10];
    int p = 0, c;

    printf("Wie lautet Ihr Name? ");
    while ((c = getchar()) != '\n' && p < 9)
    {
        name[p] = c;
        p = p + 1;
    }
    name[p] = '\0';

    printf("Guten Tag, %s!\n", name);
    return 0;
}
```



Zeichenketten

```
#include <stdio.h>

void eingabe(char s[ ], int len)
{
    int p = 0, c;
    while ((c = getchar()) != '\n' && p < len-1)
    {
        s[p] = c;
        p = p + 1;
    }
    s[p] = '\0';
}

int main(void)
{
    char name[10];
    printf("Wie lautet Ihr Name? ");
    eingabe(name, 10);
    printf("Guten Tag, %s!\n", name);
    return 0;
}
```

Arrays als Funktionsparameter

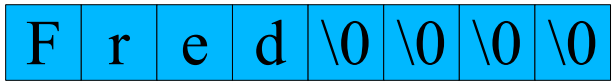
- Arrays können beim Deklarieren mit einer speziellen Syntax initialisiert werden.
- Eine Zuweisung von Arrays ist aber nicht möglich.
  - Arrays sind second-class-citizens in C.
- Beim Übergeben eines Arrays an eine Funktion wird daher nur eine Art Referenz auf das Array übergeben, die Größe geht dabei leider verloren.
  - Dieser Sachverhalt wird später genauer geklärt.
- Falls die Größe nicht aus dem Array selbst erkennbar ist, müssen wir diese mit übergeben.
  - Zeichenketten enden z.B. per Konvention auf '\0'.

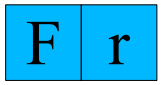
Zeichenketten initialisieren

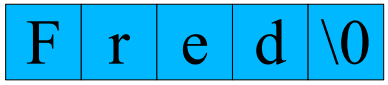
```
#include <stdio.h>

int main(void)
{
    char a[8] = "Fred";

    char b[4] = "Fred";

    // warning: initializer-string
    // for array of chars is too long
    char c[2] = "Fred";

    char d[ ] = "Fred";
    return 0;
}
```

Zeichenketten initialisieren

```
#include <stdio.h>

int main(void)
{
    char a[8] = "Fred";

    char b[4] = "Fred";

    // warning: initializer-string
    // for array of chars is too long
    char c[2] = "Fred";

    char d[ ] = "Fred";
    return 0;
}
```

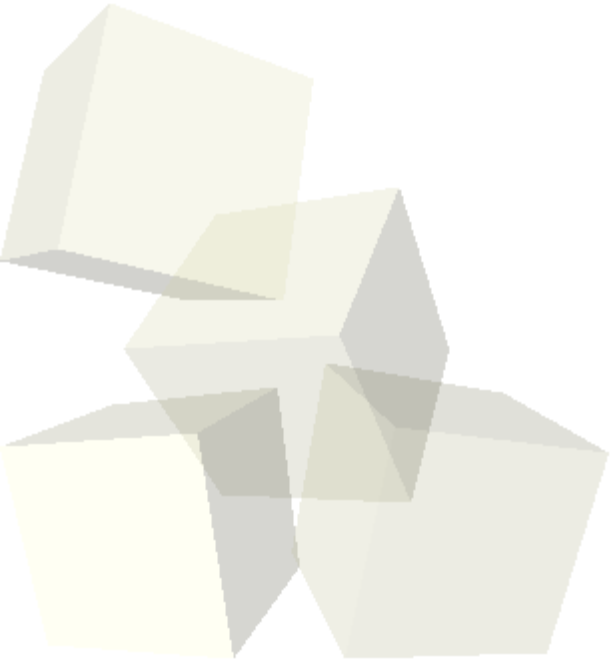






Programmierparadigma

- Der grundlegende Abstraktionsmechanismus ist die Funktion.
- Funktionen bestehen aus Deklarationen und Anweisungen.
- Funktionen können andere Funktionen aufrufen, die Kommunikation erfolgt durch Parameter.
- Der Einstiegspunkt in das Programm ist main.



Kompilierungsmodell

- Der Compiler erzeugt aus einer Textdatei mit der Endung .c ein ausführbares Programm.
  - gcc -o hello.exe hello.c