Programmieren in C

Pointer und so...

thoto

/dev/tal e.V.

6. April 2013 (Version vom 11. April 2013)

Agenda für Heute

1. Wiederholung und Vertiefung
   - Wiederholung
   - Arbeiten mit VIM und Shell
2. Zeiger
   - Was sind Zeiger?
   - Felder und Vektoren?
   - Zeichenketten

Wiederholung

Funktion vs. Methode

- Funktion hat Rückgabewert, Methode nicht

Funktion vs. Methode

- Funktion hat Rückgabewert, Methode nicht
- Variablen und Typen

Funktion vs. Methode

- Funktion hat Rückgabewert, Methode nicht
- Variablen und Typen
- Rekursion

Funktion vs. Methode

- Funktion hat Rückgabewert, Methode nicht
- Variablen und Typen
- Rekursion
- Schleifen

Nachtrag: do-while

```
do{
    a=foo()
} while(a != 4);
```

VIM Teil I

- Underminks Tipp: http://www.worldtimzone.com/res/vi.html

VIM Teil I

- Underminks Tipp: http://www.worldtimzone.com/res/vi.html
- mehrere Modi: Kommandomodus, Befehlsmodus und Einfügemodus

VIM Teil I

- Underminks Tipp: http://www.worldtimzone.com/res/vi.html
- mehrere Modi: Kommandomodus, Befehlsmodus und Einfügemodus

:w, :q Speichern mit `:w`, beenden mit `:q`

VIM Teil I

- Underminks Tipp: http://www.worldtimzone.com/res/vi.html
- mehrere Modi: Kommandomodus, Befehlsmodus und Einfügemodus

:w, :q Speichern mit `:w`, beenden mit `:q`

a, i anhängen hinter Cursor mit `a` einfügen vor mit `i`

VIM Teil I

- Underminks Tipp: http://www.worldtimzone.com/res/vi.html
- mehrere Modi: Kommandomodus, Befehlsmodus und Einfügemodus

:w, :q Speichern mit `:w`, beenden mit `:q`

a, i anhängen hinter Cursor mit `a` einfügen vor mit `i`

r Ersetzen mir `R` (einzelner Buchstabe: `r`)

VIM Teil I

- Underminks Tipp: http://www.worldtimzone.com/res/vi.html
- mehrere Modi: Kommandomodus, Befehlsmodus und Einfügemodus

:w, :q Speichern mit `:w`, beenden mit `:q`

a, i anhängen hinter Cursor mit `a` einfügen vor mit `i`

r Ersetzen mir `R` (einzelner Buchstabe: `r`)

/ Suche mit `/` (nächstes Ergebnis mit `n`)

VIM Teil I

- Underminks Tipp: http://www.worldtimzone.com/res/vi.html
- mehrere Modi: Kommandomodus, Befehlsmodus und Einfügemodus

:w, :q Speichern mit `:w`, beenden mit `:q`

a, i anhängen hinter Cursor mit `a` einfügen vor mit `i`

r Ersetzen mir `R` (einzelner Buchstabe: `r`)

/ Suche mit `/` (nächstes Ergebnis mit `n`)

sed Suchen und Ersetzen mit `%s/suche/ersetzen/g`

VIM Teil I

- Underminks Tipp: http://www.worldtimzone.com/res/vi.html
- mehrere Modi: Kommandomodus, Befehlsmodus und Einfügemodus

:w, :q Speichern mit `:w`, beenden mit `:q`

a, i anhängen hinter Cursor mit `a` einfügen vor mit `i`

r Ersetzen mir `R` (einzelner Buchstabe: `r`)

/ Suche mit / (nächstes Ergebnis mit `n`)

sed Suchen und Ersetzen mit `%s/suche/ersetzen/g`

⇒ % für ges. Dokument, g für mehrere Ersetzungen in Zeilen

VIM Teil II

Zeichen `h/j/k/l` Steuerung

VIM Teil II

Zeichen `h/j/k/l` Steuerung

Worte `w/b` Wort-Steuerung

VIM Teil II

Zeichen `h/j/k/l` Steuerung

Worte `w/b` Wort-Steuerung

visuell `v` Visueller Modus

VIM Teil II

Zeichen `h/j/k/l` Steuerung

Worte `w/b` Wort-Steuerung

visuell `v` Visueller Modus

x `x` löschen

VIM Teil II

Zeichen	`h/j/k/l` Steuerung
Worte	`w/b` Wort-Steuerung
visuell	`v` Visueller Modus
x	`x` löschen
dd	`dd` Zeile löschen
dw	`dw` Wort löschen

VIM Teil II

Zeichen	`h/j/k/l` Steuerung
Worte	`w/b` Wort-Steuerung
visuell	`v` Visueller Modus
x	`x` löschen
dd	`dd` Zeile löschen
dw	`dw` Wort löschen

VIM Teil II

Zeichen	`h/j/k/l` Steuerung
Worte	`w/b` Wort-Steuerung
visuell	`v` Visueller Modus
x	`x` löschen
dd	`dd` Zeile löschen
dw	`dw` Wort löschen
y	`yy` Zeile kopieren

VIM Teil II

Zeichen	`h/j/k/l` Steuerung
Worte	`w/b` Wort-Steuerung
visuell	`v` Visueller Modus
x	`x` löschen
dd	`dd` Zeile löschen
dw	`dw` Wort löschen
y	`yy` Zeile kopieren
u	rückgängig `<ctrl>-r`

VIM Teil II

Zeichen	`h`/`j`/`k`/`l` Steuerung
Worte	`w`/`b` Wort-Steuerung
visuell	`v` Visueller Modus
x	`x` löschen
dd	`dd` Zeile löschen
dw	`dw` Wort löschen
y	`yy` Zeile kopieren
u	rückgängig `<ctrl>-r`
p	einfügen (`P`/`p`)

Shell: Grundlegendes

- Aufruf von Programmen, die Sachen erledigen
- Beispiel: Programm `cp` kopiert Datei

Shell: Grundlegendes

- Aufruf von Programmen, die Sachen erledigen
- Beispiel: Programm `cp` kopiert Datei
- Übergabe von Argumenten
- `cp foo bar` kopiert Datei foo nach bar

Shell: Grundlegendes

- Aufruf von Programmen, die Sachen erledigen
- Beispiel: Programm `cp` kopiert Datei
- Übergabe von Argumenten
- `cp foo bar` kopiert Datei foo nach bar
- Flags, bspw. `-R` `-rf` ...

Shell: Grundlegendes

- Aufruf von Programmen, die Sachen erledigen
- Beispiel: Programm `cp` kopiert Datei
- Übergabe von Argumenten
- `cp foo bar` kopiert Datei foo nach bar
- Flags, bspw. `-R` `-rf` ...
- Wildcards, z.B. `*`

Shell: Grundlegendes

- Aufruf von Programmen, die Sachen erledigen
- Beispiel: Programm `cp` kopiert Datei
- Übergabe von Argumenten
- `cp foo bar` kopiert Datei foo nach bar
- Flags, bspw. `-R` `-rf` ...
- Wildcards, z.B. `*`
- Vorsicht: Unheitliche Flags!

Shell: Überblick

cp	Kopie `cp -R foo-projekt bar-projekt` (-R bei Verz.)
mv	Verschieben `mv datei1 datei2` (immer rekursiv)
mkdir	Erstellt Verzeichnis `mkdir foo`
rm	Löscht Datei `rm foo`.
rmdir	Löscht einzelnes Verzeichnis `rmdir foo-projekt`
ls	Zeigt Verzeichnisinhalt an. `ls foo` oder `foo-projekt`
vim	Editor `vim foo`
less	Zeigt Datei seitenweise an. `less foo` (more)
cat	Gibt Datei aus
cd	Gibt Datei aus

Shell: beliebte Flags

- können *gewöhnlich* verwendet werden

-R Rekursiv $\Rightarrow$ in Unterverzeichnissen

-f erzwingen

-a Alle Dateien

-l nur bei `ls`: zeigt weitere Informationen an.

- Zu beachten: Verzeichnisse `.` und `..`

Ausprobieren!

Ausprobieren und Fragen

Zeiger

Zeiger

- Keine direkte Variable

$\Rightarrow$ Zeiger auf Variable

- Speicheradresse

Zeiger

- Keine direkte Variable

⇒ Zeiger auf Variable

- Speicheradresse
- Siehe http://xkcd.com/138/

Wozu Zeiger?

- Vorteil: In andere Variable schreiben
- Beispiel: Mehrere Variablen füllen
- Variable in anderer Funktion füllen

Felder und Vektoren

- Zeiger ist auch *Feld* oder *Vektor* oder *Array*
- Mehrere Variablen unter einem Namen

Felder und Vektoren

- Zeiger ist auch *Feld* oder *Vektor* oder *Array*
- Mehrere Variablen unter einem Namen
- Deklaration: `int foo[4];` (4 Variablen, Typ int)
- Zugriff: `foo[0]=23;`
  `foo[3]=42;`

Beispiel: Zählschleife II

```
int v[50];
for(i=0;i<50;i++){
   v[i]=rand();
}
```

Multidimensionale Felder

```
int v[2][2];
v[0][0]=0;
v[0][1]=2;
v[1][0]=4;
v[1][1]=8;
```

Zeichenkette

- Vektor von Buchstaben
- Terminiert mit \0 (0x00)
- Zeichenkette darf beliebig lang sein

Beispiel: Zeichenkette

```
char hello[] = "Hello World!\n";
i=0;
while(hello[i]!=0x00){
   putchar(hello[i]);
   i++;
}
```

Zeiger

- Speicherstelle
- Referenzierung: `&stelle`
- Dereferenzierung: `*variable`

Zeiger

- Speicherstelle
- Referenzierung: `&stelle`
- Dereferenzierung: `*variable`
- bspw:

```
int i;
int* pi=&i;
printf("%d\n",*pi);
```

Zeiger

- Speicherstelle
- Referenzierung: `&stelle`
- Dereferenzierung: `*variable`
- bspw:
```
int i;
int* pi=&i;
printf("%d\n",*pi);
```
- Mit Zeigern kann man rechnen!

Zeiger

- Speicherstelle
- Referenzierung: `&stelle`
- Dereferenzierung: `*variable`
- bspw:
```
int i;
int* pi=&i;
printf("%d\n",*pi);
```
- Mit Zeigern kann man rechnen!

$\Rightarrow$ Erhöhung um Typgröße

Zeiger

- Speicherstelle
- Referenzierung: `&stelle`
- Dereferenzierung: `*variable`
- bspw:
```
int i;
int* pi=&i;
printf("%d\n",*pi);
```
- Mit Zeigern kann man rechnen!

$\Rightarrow$ Erhöhung um Typgröße

$\Rightarrow$ Daher auch Zeiger und Wert nicht verwechseln