

KV Betriebssysteme Wiederholung

Peter R. Dietmüller

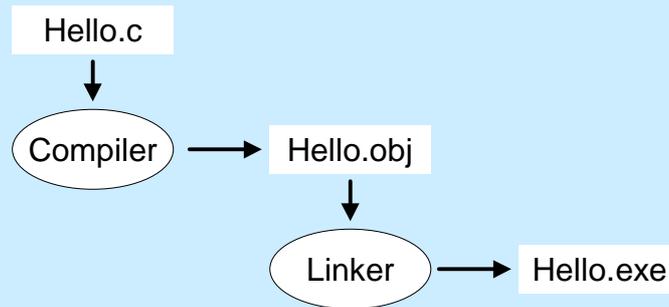
Inst. f. Informationsverarbeitung und Mikroprozessortechnik (FIM)
Johannes Kepler Univ. Linz, Altenbergerstr. 69, A-4040 Linz, Austria
E-Mail: dietmueller@fim.uni-linz.ac.at

Das erste C-Programm

Jedes C-Programm besteht zumindest aus einer Funktion main. Im folgenden Beispiel gibt das Programm den Text "Hello world." aus.

```
#include <stdio.h>
int main (void) {
    printf ("Hello world.\n");
    return 0;
}
```

C-Programm übersetzen



Boolescher Datentyp

C kennt keinen expliziten Datentyp Boolean. Stattdessen gelten die schon bei den logischen Operatoren angegebenen Regeln:

- ein Wert von 0 bedeutet FALSE
- ein Wert ungleich 0 bedeutet TRUE

Diese Regelung ist insofern praktisch, da man statt den in Pascal (Modula-2) umständlich zu schreibenden Abfragen

- **IF (i AND 3) <> 0 ... schreiben kann: if (i & 3) ...**

Pythagoräische Tripel

Problem:

Gesucht sind ganzzahlige Tripel (a,b,c) für die gilt:

$$a^2 + b^2 = c^2$$

Lösung:

[Tripel.c](#)

Datentyp - Umwandlung (1)

Implizit

- char- und int-Werte können zusammen beliebig in arithmetischen Ausdrücken auftreten. Jeder char-Wert wird dabei automatisch in einen int-Wert umgewandelt. Integers, die auf char umgewandelt werden, verlieren ihre höherwertigen Bits (genauso bei long auf int).
Achtung: nur Konvertierungen von unsigned char in int resultieren in Werten von 0..255, sonst von -128..127
- alle float-Werte werden in einem Ausdruck zusammen mit double-Werten in double-Werte umgewandelt.

Explizit (Type-Cast)

- Der sogenannte cast-Operator (Datentyp) erlaubt das explizite Umwandeln von Datentypen.

Datentyp - Umwandlung (2)

```
int i; char c;
```

```
c = 'A';           /* keine Umwandlung notwendig */  
i = c;           /* implizite Umwandlung */  
i = 'B';        /* implizite Umwandlung */  
c = (char) i;   /* explizite Umwandlung */  
                /* (wäre hier nicht notwendig!) */
```

Importieren von Modulen

Geschieht durch das Inkludieren von Header-Files (Extension .h), in denen die Schnittstelle eines Moduls definiert ist. Das Modulkonzept wird später noch genauer erklärt.

Um zum Beispiel das Modul STDIO (Standard Input/Output) zu importieren, muß man am Beginn des Programms folgende Zeile aufnehmen.

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

Importieren von Modulen - Beispiel

```
/* -- 100 Mal Würfeln -- */
#include <stdlib.h>
#include <stdio.h>

int main(void) {
    int i;
    srand(0); /* Zufallszahlen init. */
    printf("100 Mal Würfeln:\n");
    for (i = 0; i < 100; i++) printf("%2d", rand() % 6 + 1);
    return 0;
}
```

KV Betriebssysteme

Einheit 2: Arrays, Zeiger, Zeichenketten

Peter R. Dietmüller

Inst. f. Informationsverarbeitung und Mikroprozessortechnik (FIM)
Johannes Kepler Univ. Linz, Altenbergerstr. 69, A-4040 Linz, Austria
E-Mail: dietmueller@fim.uni-linz.ac.at

Array - Modell

| Vornamen | |
|----------|-----------|
| 0 | Christian |
| 1 | Istvan |
| 2 | Johannes |
| 3 | Günther |
| 4 | Martin |
| | ... |
| N-1 | Peter |

Array - Syntax

Deklaration:

- *BaseType Identifier "[" Elements "]" { "[" Elements "]" }*
- z.B.: `int werte[100]; char alphabet[256];`

Zugriff:

- *ArrayName[index]*
- z.B.: `c = alphabet[255]; i = werte[1];`
- aber: `c = alphabet[256]; /* ?? */`

Array - Initialisierung

```
char hex[16];  
hex[0] = '0'; hex[1] = '1'; ... hex[15] = 'F';  
  
char hex[16] = {'0', '1', '2', '3', '4', '5', '6', '7',  
               '8', '9', 'A', 'B', 'C', 'D', 'E', 'F'};  
  
int values[3] = {1, 2}; /* Rest wird 0 gesetzt! */
```

Array - Open Array

```
int writeName (char name[]) {  
    /* 'name' .. offenes, eindimensionales Array */  
}  
  
int getPairs (int p[][2]) {  
    /* 'p' .. offenes, zweidimensionales Array */  
}
```

Array - Parameter

```
void ArrayAenderung(int werte[]) {
    werte[1] = 5;
}

void ArrayTest(void) {
    int werte[11] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
    printf("%d", werte[1]);           /* 1 */
    ArrayAenderung(werte);
    printf("%d", werte[1]);         /* 5 */
}
```

Array - Regeln (1)

- **Die Anzahl der Elemente muß bei der Definition durch einen konstanten Ausdruck bestimmt sein (auch in Verbindung mit dem sizeof-Operator)**
- **Es gibt zur Laufzeit keine Möglichkeit die Anzahl der Elemente eines Arrays zu bestimmen! (siehe aber sizeof-Operator)**
- **Der erste Index eines Arrays ist immer 0! Die zehn Elemente eines Arrays test[10] sind also test[0], test[1], ..., test[9]!**

Array - Regeln (2)

- **Arrays werden als Parameter entgegen den C-Gepflogenheiten Call-by-Reference übergeben (eigentlich: Call-by-Value Übergabe der Array-Adresse!)**
- **Wenn es nicht dem Anlegen von Speicherplatz für einen Array dient, so kann die Größe der ersten Dimension offen gelassen werden (Open Array), z.B. bei Arrays als Funktionsparametern**

Array - Regeln (3)

- **Zuweisung eines Arrays an ein anderes resultiert NICHT in einem Kopieren der Elemente.**
- **Wenn bei einer Array-Initialisierung nicht alle Elemente mit Werten versehen werden, wird der Rest mit 0 aufgefüllt**

Array - Mehrdimensional (1)

| | 0 | 1 | 2 |
|-----|-----------|----------|-----------|
| 0 | Christian | Katrin | Susanne |
| 1 | Istvan | Aslan | Christine |
| 2 | Johannes | Reinhard | Franz |
| 3 | Günther | Claudia | Inge |
| 4 | Martin | Andrea | Martina |
| ... | ... | ... | ... |
| N-1 | Peter | Irene | Ulrike |

Array - Mehrdimensional (2)

Mehrdimensionale Arrays werden so gespeichert, daß sich jeweils der letzte Index am schnellsten ändert ("Reihen-Spalten-Ordnung")

Bei Deklaration und Zugriff muß jede Dimension in eigene Klammern "[" und "]" gestellt werden ([i1][i2], nicht [i1,i2])!

```
int pairs[3][2] = {{1,2},{3,4}}; /* Rest wird 0 gesetzt! */
```

Array - Mehrdimensional (3)

Speicherabbild eines zweidimensionalen Arrays

```
int pairs[4][3];
```

| | | | | | | |
|--------|--------|--------|--------|--------|----|--------|
| [0][0] | [0][1] | [0][2] | [1][0] | [1][1] | .. | [3][2] |
|--------|--------|--------|--------|--------|----|--------|

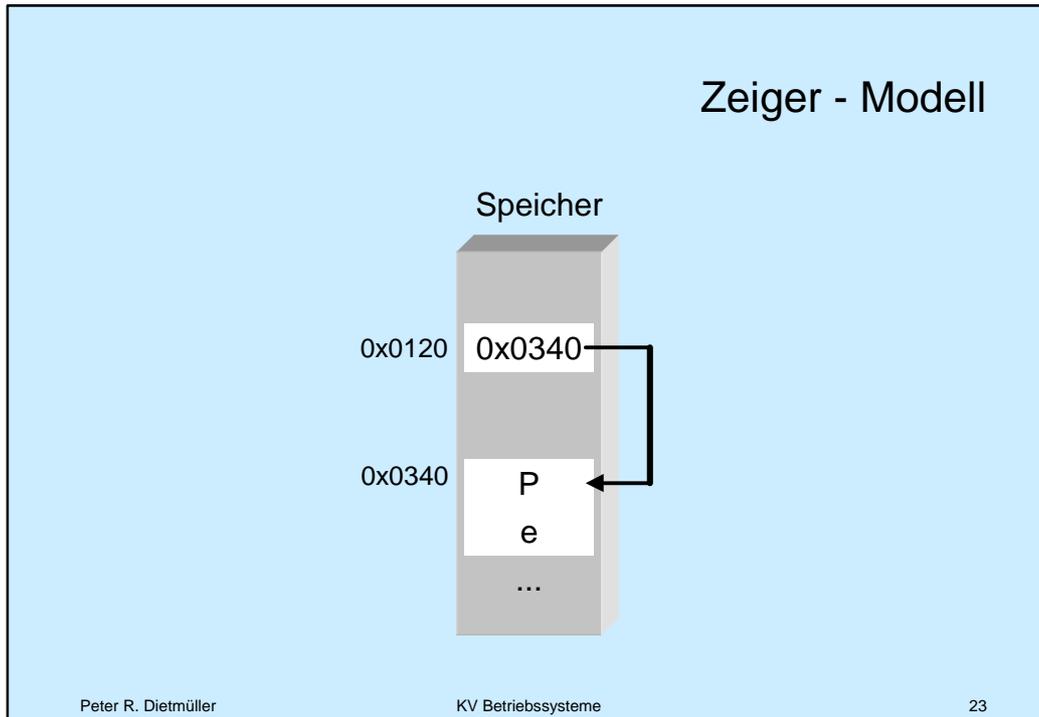
Array - Beispiel

Problem:

Ein Käfer torkelt in einem rechteckigen Raum mit $N \cdot M$ Fliesen. In jedem Zeitschritt tritt er auf irgendeine der Nachbarfliesen (wenn er an die Wand stößt, bleibt er auf seiner Fliese). Gesucht ist die Anzahl der Schritte, die der Käfer braucht, um alle Fliesen mindestens einmal zu betreten.

Lösung:

[Kaefer.c](#)



Zeiger

Deklaration eines Zeigers:

- *BaseType "*" Identifier*

Beispiele:

- `int *zeiger1;`
- `int *zeiger2, wert1; /* Achtung! 'wert 1' ist kein Zeiger! */`

Adresse einer Variablen:

- **&-Operator**

Beispiele:

- `zeiger1 = &wert1;`

Peter R. Dietmüller KV Betriebssysteme 24

Zeigerarithmetik

Addition zu Zeigern entspricht Adress-Addition von $n * \text{sizeof}(\text{element})$

- `int *zeiger, wert; zeiger = &wert; zeiger++;`

Subtraktion zu Zeigern entspricht Adress-Subtraktion von $n * \text{sizeof}(\text{element})$

- `int *zeiger, wert; zeiger = &wert; zeiger--;`

Differenzbildung von zwei Zeigern auf Arrays liefert die Anzahl der dazwischenliegenden Elemente

- `int *zeiger1, *zeiger2; ...; printf („%d“, zeiger2 - zeiger1);`

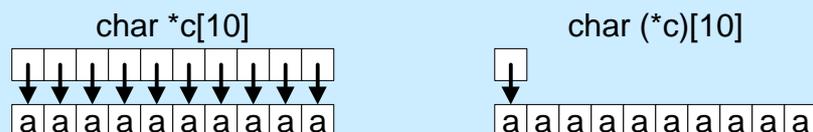
Zeiger und Arrays

Verwendung von `arrayName` entspricht der Adresse des ersten Arrayelementes.

- `int werte[10]; /* werte == &werte[0] */`

Unterschied:

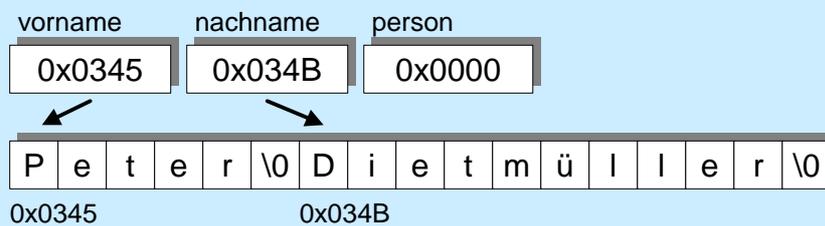
- `char *c[10];` ... array of pointers
- `char (*c)[10];` ... pointer to array



Zeiger und Zeichenketten

In C sind Zeichenketten Character-Arrays, die mit einem Nullbyte abgeschlossen werden.

- `char vorname[] = „Peter“;`
- `char *nachname = „Dietmüller“;`
- `char *person;`



Zeiger - Beispiele (1)

```
/* -- Unterschiedliche Deklarationen -- */
#include <stdio.h>
int main (void) {
    char *a[10],*(b[10]),(*c)[10];
    printf("%d %d %d\n", sizeof(a),sizeof(b),sizeof(c));
    /* Ergebnis: 40, 40, 4 */
    return 0;
}
```

Zeiger - Beispiele (2)

Anhängen eines Zeichens an eine Zeichenkette:

```
while (*c) c++;  
*c++='!';  
*c=0;
```

```
/* dasselbe mit einer Funktion aus <string.h> */  
strcat(c, "!!");
```

Zeiger - Beispiele (3)

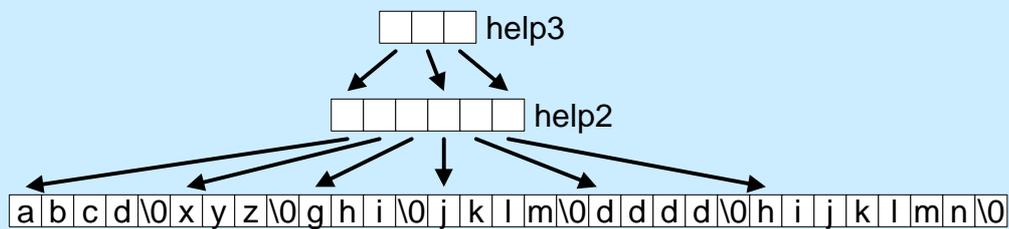
Zeigerarithmetik:

```
int i,*p,fib[]={1,1,2,3,5,8};
```

```
/* -- Ausgabe des Arrays in drei Varianten -- */  
for (i = 0; i <= 5; i++) printf("%d\n", fib[i]);  
for (p = &fib[0]; p <= &fib[5]; p++) printf("%d\n", *p);  
for (p = fib; p <= fib+5; p++) printf("%d\n", *p);
```

Zeigerarithmetik - Beispiel (1)

```
char *string = "abcdefghijklmnopqrstuvwxy";
char *help2[] = { "abcd", "xyz", "ghi",
                 "jklm", "dddd", "hijklmn" };
char **help3[] = { help2, help2+3, help2+5 };
```



Zeigerarithmetik - Beispiel (2)

```
int main (void) {

    char *help,***help4;

    help = string+10;
    printf("%s\n", help);          /* klmnopqrstuvwxyz */
    printf("%s\n", help-3);       /* hijklmnopqrstuvwxyz */
    printf("%s\n\n", help+5);     /* pqrstuvwxyz */
}
```

Zeigerarithmetik - Beispiel (3)

```

printf("%s\n", *(help2+3));    /* jklm */
printf("%s\n", help2[5]);     /* hijklmn */
printf("%s\n\n", help2[1]+2); /* z */

printf("%s\n", **help3);     /* abcd */
printf("%s\n", *help3[1]);   /* jklm */
printf("%s\n\n", **(help3+2)); /* hijklmn */

```

Zeigerarithmetik - Beispiel (4)

```

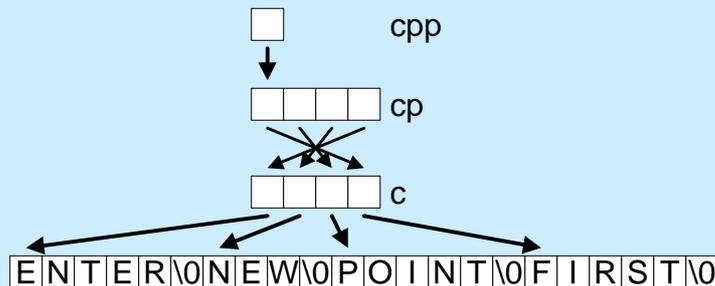
help4 = help3+1;
printf("%s\n", *++*help4);    /* dddd */
printf("%s\n", *++(*help4++)); /* hijklmn */
printf("%s\n", *--*help4+2); /* dd */

return 0;
}

```

Zeigersalat (1)

```
char *c[] = {"ENTER", "NEW", "POINT", "FIRST"};
char **cp[] = {c+3, c+2, c+1, c};
char ***cpp = cp;
```



Zeigersalat (2)

```
int main (void) {

    printf("%s", **++cpp);          /* POINT */
    printf("%s", *--*++cpp+3);     /* ER */
    printf("%s", *cpp[-2]+3);      /* ST */
    printf("%s\n", cpp[-1][-1]+1); /* EW */
    return 0;
}
```

Call by Reference - Parameter (1)

C unterstützt nur die Call-by-Value Übergabe. Durch Pointer kann aber Call-by-Reference simuliert werden:

```
void CallByValue (int i) {  
    i=1;  
}  
void CallByReference (int *i) {  
    *i=2;  
}
```

Call by Reference - Parameter (2)

```
int main (void) {  
  
    int n=3;  
    CallByValue(n);  
    printf("%d\n",n);      /* 3 (nicht 1!!!) */  
    CallByReference(&n);  
    printf("%d\n",n);      /* 2 */  
  
    return 0;  
}
```



Peter R. Dietmüller KV Betriebssysteme 39



Peter R. Dietmüller KV Betriebssysteme 40



Peter R. Dietmüller KV Betriebssysteme 41



Peter R. Dietmüller KV Betriebssysteme 42