

4 Pointer und Arrays

4.1 Einführung

Im vorherigen Kapitel "Pointer-Arithmetik" haben wir gesehen, dass man Pointer addieren und subtrahieren kann. Dies macht aber nur Sinn, wenn die Speicherstelle, auf die ein Pointer zeigt, hintereinander angeordnet sind. Diese Bedingung erfüllen Arrays (bzw. Felder).

Wenn wir ein Integer-Array mit fünf Zahlen definieren, so liegen diese fünf Elemente des Arrays immer hintereinander. Inkrementiert (erhöhen um 1) man einen Pointer, welcher zum Beispiel auf die zweite Speicherstelle zeigt, so zeigt er nach dem Inkrementieren auf die dritte Speicherstelle.

Spezialfall 1: Adresse eines Arrays

In C ist der Variablenname eines Arrays die Adresse des ersten Elementes des Arrays. Also ein Pointer auf das erste Element.

```
int  Zahlen[10];           // Integer-Array mit 10 Elementen

int *pZahlEinfach = Zahlen; // Pointer, der auf das erste Element von Zahlen zeigt
int *pZahlKomplex = &Zahlen[0]; // Geht auch, ist aber komplizierter
```

Spezialfall 2: String in C bzw. char-Array

Ein String in C ist sehr speziell definiert. Es handelt sich nämlich nicht um einen eigenen Datentyp, sondern um ein Zeichen-Array (char-Array), welches als letztes Zeichen immer ein Zeichen mit dem Wert 0 gespeichert hat. Man spricht daher bei Strings von Null terminierten Zeichenarrays.

Bis jetzt haben Sie die Null am Schluss eines Strings nicht beachtet, da Sie eventuell nur die String-Funktionen der C-Bibliothek verwendet haben und diese die Strings automatisch korrekt verarbeiten.

```
#include "stdio.h"
#include "stdlib.h"
#include "string.h"

void main()
{
    char Text[100];

    strcpy (Text, "Hallo");

    printf("Der Text lautet: ");
    for (int i = 0; Text[i] != '\0'; i++) printf ("%c", Text[i]);
    system("pause");
}
```

Das obige Beispiel schreibt «Der Text lautet: Hallo» auf den Bildschirm. Etwas speziell scheint dabei die for-Schleife zu sein, insbesondere die Abbruch-Bedingung: `Text[i] != '\0'`. Dabei ist `'\0'` eine Konstante vom Typ `char`. Durch die einfachen Anführungszeichen erkennt der Compiler, dass es sich um ein Zeichen handelt. Durch den Backslash wird ein Escape-Zeichen definiert, wie z.B. `«\n»` für eine neue Zeile.

Folgen nach dem Backslash nur Ziffern, so werden diese Ziffern als eine Oktal-Zahl (8er-System) interpretiert. Analog wie `«\0x20»`, wegen dem `«x»`, als eine Hex-Zahl interpretiert wird. Daher hat das Zeichen `'\0'` den Wert Null. Diese Schreibweise findet man nicht nur in der Fachliteratur von C, sondern sie wird verwendet, weil sie typenkorrekt ist. So wird ein Zeichen mit einem Zeichen verglichen.

Achtung: Schreiben Sie nie `'0'` statt `'\0'`. `'0'` ist das Zeichen für Null und hat den Wert Hex 30 bzw. Dezimal 48 und nicht 0! (siehe ASCII bzw. ANSI-Tabelle).

4.2 Aufgaben



Verwenden Sie zur Lösung der folgenden Aufgaben weder irgendwelche spezielle Bibliotheksfunktionen, ausser für die Ein- und Ausgabe, noch Indexe beim Zugriff auf Arrays, sondern ausschliesslich Pointer.

Aufgabe 01: Einfaches Zahlen-Array

Schreiben Sie ein Programm mit einem Integer-Array «Zahlen», welches aus 10 Elementen besteht. Initialisieren Sie das Array mit den Werten 10, 20, 30 bis und mit 100. Lesen Sie alle Werte im Array nur über einen Pointer aus.

Aufgabe 02: Summe und Mittelwert

Schreiben Sie ein Programm, welches ein Integer-Array "Zahlen" mit mindestens 10 Elementen hat. Geben Sie die Werte des Arrays entweder über die Tastatur ein oder ermitteln Sie diese mittels der random-Funktion «rand()». Bilden Sie nun aus den im Array gespeicherten Werten mit Hilfe eines Pointers die Summe und danach den Mittelwert. Geben Sie die erhalten Werte aus.

Aufgabe 03: Programmcode mit Array und Zeigerarithmetik verstehen

Welche Ausgaben liefert das folgende Programm?

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

int main()
{
// Variablen deklarieren und initialisieren
    int Zahl = 10; // Adresse von Zahl = 1250
    char Zeichen = 'A'; // Adresse von Zeichen = 1480
    short Liste[8] = {1, 2, 4, 8, 16, 32, 64, 128}; // Adresse von Liste = 2870
    double Werte[2] = {0.00, 5.9}; // Adresse von Werte = 3890

// Pointervariablen deklarieren und initialisieren
    int *pZahl = &Zahl; // Adresse von pZahl = 8000
    char *pZeichen = &Zeichen; // Adresse von pZeichen = 8500
    short *pListe = NULL; // Adresse von pListe = 9000
    double *pWert = Werte; // Adresse von pWert = 9500

// Ausgaben 1
    printf ("%i, %i\n", Zahl, pZahl);
    printf ("%p, %c\n", &Zeichen, *pZeichen);
    printf ("%d, %d\n", *Liste, Liste[1 + 2] + Liste[0]);
    printf ("%p, %lf\n", &(&pWert), Werte[1]);

// Veränderungen
    Zahl = (*pZahl)++;
    Zeichen++;
    pListe = &Liste[1];
    pListe++;
    *pWert = --Werte[0] + 5;

// Ausgaben 2
    printf ("%i, %i\n", Zahl, *pZahl);
    printf ("%c, %p\n", Zeichen, pZeichen);
    printf ("%d, %d\n", *pListe, &Liste[3 + 2]);
    printf ("%lf, %lf\n", Werte[0], Werte[1]);

    system("pause");
    return 0;
}
```

printf	Ausgabe
1. printf	
2. printf	
3. printf	
4. printf	
5. printf	
6. printf	
7. printf	
8. printf	

Aufgabe 04: Fehler im Programmcode finden

Beschreiben Sie mindestens fünf Fehler die einen Compiler- oder Laufzeitfehler liefern könnten?

```

00: #include <stdio.h>
01: #include <stdlib.h>
02:
03: int main(void)
04: {
05:     //Variablen deklarieren und initialisieren
06:     int Zahl = 10.0;
07:     char Buchstabe = 'A';
08:     short Liste[8] = {1, 2, 4, 8, 16, 32, 64, 128};
09:     double Wert = 0.00;
10:
11:     //Pointervariablen deklarieren und initialisieren
12:     int *pZahl = null;
13:     char *pBuchstabe = &Buchstabe;
14:     short *pListe = *Liste;
15:     double *pWert = &Wert;
16:     pZahl = &Zahl;
17:
18:     //Ausgabe 1
19:     printf ("%i, %i\n", Zahl, pZahl);
20:     printf ("%c, %c\n", &Buchstabe, *pBuchstabe);
21:     printf ("%d, %d\n", *Liste, Liste[1 + 2] + Liste[0]);
22:     printf ("%lf, %lf\n", &pWert +1, *Zahl);
23:
24:     //Veränderungen 1
25:     Zahl = *pZahl++;
26:     Buchstabe = "X";
27:     pListe = &Liste[3];
28:     pListe++;
29:     pWert = --Wert + 8;
30:
31:     //Ausgabe 2
32:     printf ("%i, %i\n", Zahl, *pZahl);
33:     printf ("%c, %c\n", Buchstabe, pBuchstabe);
34:     printf ("%lf, %lf\n", *(pWert+5), Wert);
35:
36:     system("pause");
37:     return 0;
38: }

```

Zeile	Fehler-Beschreibung

Aufgabe 05: Arrayzugriffe mit Zeigern verstehen

Die nachfolgenden Fragen beziehen sich auf das unten ersichtliche integer Array mit dem Namen «Feld», welches im Arbeitsspeicher an der Startadresse 13'500₁₀ (Dezimalsystem) liegt.

Feld	87	-99	-33	-30	55	67	994	-874	109	22
	0	1	2	3	4	5	6	7	8	9

Die untenstehende Tabelle enthält diverse Anweisungssequenzen, die sich auf das obige Array «Feld» beziehen. Schreiben Sie zu jedem Eintrag, die entsprechende Adresse oder den entsprechenden Wert in die Spalte «Rückgabewert...». Falls eine Anweisung nicht ausführbar sein sollte, schreiben Sie in der Spalte «Rückgabewert ...» das Stichwort «Fehler:», ob es sich um einen Compiler- («Compiler») oder Laufzeitfehler («Laufzeit») handelt und eine kurze Fehlerbeschreibung.

Nr.	Anweisungssequenz	Rückgabewert der Anweisungssequenz Fehlerbeschreibung
01	Feld	
02	Feld[0]	
03	&Feld[9]	
04	Feld[0] + Feld	
05	Feld[2] + 5	
06	Feld[10]	
07	&Feld[0]	
08	&Feld[2 + 5]	
09	13500[2]	
10	Feld[8 - 3] + Feld[3]	

Aufgabe 06: Stringlänge

Schreiben Sie ein Programm mit welchem Sie einen String «Text» über die Tastatur eingeben, danach mit Hilfe von Pointern die Länge des Strings bestimmen und das Resultat (Stringlänge) ausgeben.

Aufgabe 07: Umgedrehter String

Schreiben Sie ein Programm mit welchem Sie einen String «Text» über die Tastatur eingeben und danach mit Hilfe von Pointern den String verkehrt (letzter Buchstabe zuerst usw.) ausgeben, d.h. von hinten nach vorne.

Aufgabe 08: Leerzeichen nicht ausgeben

Schreiben Sie ein Programm mit welchem Sie einen String «Text» über die Tastatur eingeben und danach mit Hilfe von Pointern den String ausgeben, ohne dass Leerzeichen ausgeben werden.

Aufgabe 09: Leerzeichen am Anfang eines Strings entfernen

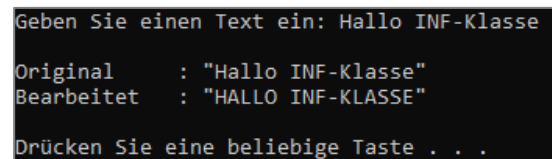
Schreiben Sie ein Programm mit welchem Sie einen String «Text» über die Tastatur eingeben. Entfernen Sie danach mit Hilfe von Pointern alle Leerzeichen am Anfang des Strings. Geben Sie am Schluss den bearbeiteten String aus.

Vorher : " Hallo"
Nachher: "Hallo"

Aufgabe 10: Grossschrift

Schreiben Sie ein Programm mit welchem Sie einen String «Text» über die Tastatur eingeben und danach mit Hilfe von Pointern alle kleingeschriebenen Buchstaben (ohne Umlaute) im Text durch deren grosses Äquivalent ersetzen und danach den bearbeiteten String ausgeben.

Das Bild zeigt eine mögliche Programmdarstellung:



```
Geben Sie einen Text ein: Hallo INF-Klasse  
Original      : "Hallo INF-Klasse"  
Bearbeitet    : "HALLO INF-KLASSE"  
  
Drücken Sie eine beliebige Taste . . .
```

Aufgabe 11: Wortzähler

Schreiben Sie ein Programm mit welchem Sie einen String «Text» über die Tastatur eingeben und danach mit Hilfe von Pointern die Anzahl Wörter im Text zählen und ausgeben.