Zeiger:

Wie wir bereits wissen, müssen Variablen vor ihrer ersten Verwendung deklariert werden. Bei der Deklaration legen wir den Variablenname und den zu speichernden Datentyp fest. Im Vergleich zu herkömmlichen Variablen haben Zeigervariablen andere Eigenschaften. Sie sind kein Platzhalter für Inhalte, wie Zahlen oder Zeichen, sie zeigen vielmehr auf eine Adresse. *(Bsp.: Es wird die Schublade 3 aufgezeigt, in der sich Daten befinden.)*

Schreibt man z.B. **int *z;** so hat dies folgende Bedeutung:
Es handelt sich um einen Zeiger z, der auf eine Variable vom Typ **„integer“** zeigt.

Der Ausdruck **z = &i;** ergibt die Adresse des Speicherplatzes der Variable i und weist sie dem Zeiger z zu.

44.a Übung

Analysieren Sie das nachfolgende Programm und dokumentieren Sie das Ergebnis.
Was wird ausgegeben und warum?

```
#include <stdio.h>

int main()
{
  int i, *z;

  i= 5;
  z= &i;
  printf ("\n%d %d", i, *z);

  getchar();
  return 0;
}
```

44.b Übung

Das nachfolgende Programm enthält einen „schwerwiegenden“ Fehler.

```
#include <stdio.h>

int main()
{
  int *z;

  *z = 10;
  printf ("\n%d", *z);

  getchar();
  return 0;
}
```

Um welchen Fehler handelt es sich hierbei?

45.a Übung

Schreiben Sie bitte ein Programm, das eine Variable a mit 5 und eine Variable b mit 10 belegt und anschließend den Inhalt dieser beiden Variablen miteinander vertauscht und dann am Bildschirm ausgibt.

45.b Übung

Modifizieren Sie bitte das obige Programm derart, dass die Vertauschung nicht mehr im Hauptprogramm selbst durchgeführt wird, sondern in einer Funktion. Es sollen dabei keine globalen Variablen verwendet werden.

Hinweis: Die Vertauschungsfunktion hat sinnvollerweise zwei Argumente. Eine Funktion kann maximal **einen** Ergebniswert zurückliefern. Arbeiten Sie bitte mit Zeigern.

46. Übung

Analysieren und dokumentieren Sie bitte die Ausgabe des folgenden Programms:

```
#include <stdio.h>

int main()
{
  int i; int *p; i = 1; p = &i;

  printf("\nNach p = &i; ist\n");
  printf(" i = %d\n",  i);
  printf(" p = %d\n",  p);
  printf("*p = %d\n", *p);

  *p = 2;

  printf("\nNach *p = 2; ist\n");
  printf(" i = %d\n", i);
  printf(" p = %d\n",  p);
  printf("*p = %d\n", *p);
  printf("\nsizeof(int *) = %d\n", sizeof(int *));

  getchar();
  return 0;
}
```

Hinweis: Operator sizeof liefert die Größe des Typs in Bytes.