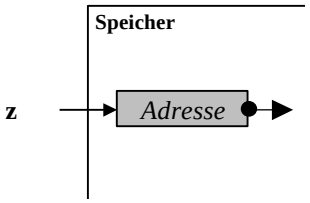
Inhalt

9 Zeiger

Zeiger sind, neben den ganzen Zahlen und den Gleitpunktzahlen, die dritten Grundobjekte der Programmiersprache C.

9.1 Deklaration und Definition von Zeigern

Zeigervariablen sind Variable, deren Wert eine Adresse ist.



Syntax: *Typ* ***** *Zeigername* **;**
int * z ;

Typ:
- Der Zeiger mit dem angegebenen *Zeigernamen* zeigt mit seinem Wert, einer Adresse, auf einen Speicherbereich.
- Der *Typ* des Zeigers bestimmt den Typ des *Wertes* im Speicher auf den der Zeiger zeigt.

Zeigername:
- *C-Name*

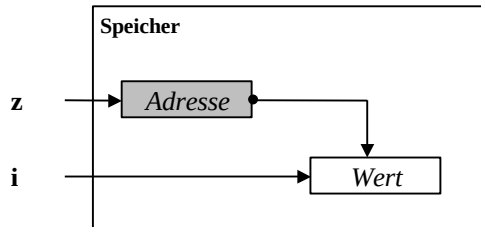
In Abhängigkeit von der jeweiligen Implementierung und der Deklarationsstelle hat ein Zeiger zunächst einen zufälligen Wert bzw. den *Wert* **`NULL`** (**`´\0´`**). **`NULL`** wird in `<stdio.h>` definiert:

```
#ifndef NULL
#define NULL ((void *)0)          /* Zeiger ohne Typ */
#endif
```

Zugriff auf Variablenadresse durch den Referenzierungsoperator bzw. Adreßoperator:

Syntax: **&** *Variablenname*
Semantik:
- Die Adresse der Variablen mit dem angegebenen *Variablenname* wird bestimmt.

z = & i ;




```
int i; scanf( "%d", &i);
```

bzw.
```
int i, *z; z = &i; scanf( "%d", z);
```

Zugriff auf Variablenwert durch den Dereferenzierungsoperator:
Syntax: ***** *Zeigervariable*
Semantik:
- Der Wert der Variablen, auf welche die angegebene *Zeigervariable* zeigt, wird bestimmt.

i = 7 ; ist äquivalent mit *** z = 7 ;**

zeiger.c

```
/*
 * Arbeit mit Zeigern
 */

# include <stdio.h>

int main()
{
  int *z, i = 1;  /* z ist Zeiger auf einen int_Wert */

   /* Zugriff auf Speicheradresse und Speicherinhalt */
                  /* z zeigt auf die int_Variable i */
  z = &i;
  printf( "%p: %d\n", z, *z);          /* 0x28cd0c: 1 */

  i++;
  printf( "%p: %d\n", z, *z);          /* 0x28cd0c: 2 */

          /* Zugriff ueber Dereferenzierungsoperator */
  *z = 3;
  printf( "%p: %d %d\n ", z, *z, i);/* 0x28cd0c: 3 3 */

  return 0;
}
```

- Durch die Arbeit mit Zeigern hat man die Möglichkeit, Inhalte über Speicheradressen zu manipulieren.
- Zeiger können als Parameter von Funktionen und als Funktionswert übergeben werden.

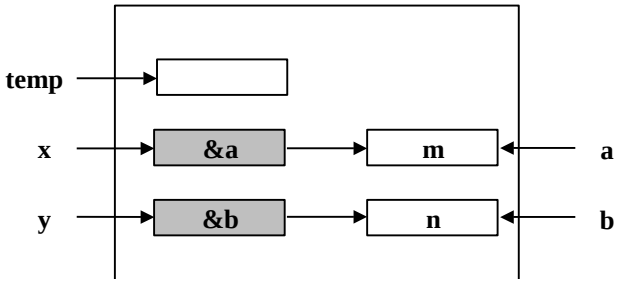
Mit dem nächsten Beispiel wird gezeigt, dass sich innerhalb von C-Funktionen auch mehrere Werte berechnen lassen. Es werden **Adressen als Argumente** übergeben, also Zeiger auf die zu manipulierenden Werte.

⇒ **Mit der Begrenzung der Unterprogrammtechnik in C auf Funktionen besteht keine Einschränkung gegenüber anderen Sprachen.**

swapping.c

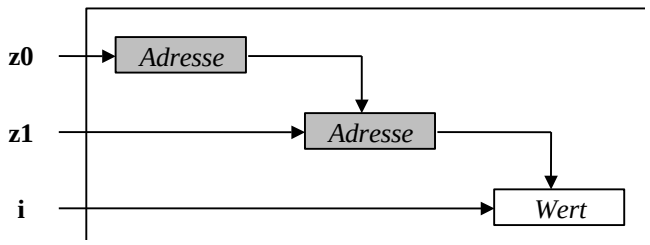
```
/*
 * Vertauschen von Variablenwerten
 */

#include <stdio.h>

                     /* Zeiger als Funktionsparameter */
void swap( int *, int *);
```

```
int main()
{
  int a, b;
                              /* Adressen als Argumente */
  printf( "Eingabe a = "); scanf( "%d", &a);
  printf( "Eingabe b = "); scanf( "%d", &b);
  printf( "\n");

  swap( &a, &b);              /* Adressen als Argumente */

  printf( "Ausgabe a = %d \nAusgabe b = %d.\n", a, b);

  return 0;
}

void swap( int *x, int *y)     /* Funktionsdefinition */
{
  int temp;
                           /* Zugriff auf Speicherinhalte */
  temp = *x; *x = *y; *y = temp;
  return;
}
```



9.2 Zeiger auf Zeiger

Die Bezugsvariable eines *Zeigertyps* **kann wieder einen** *Zeigertyp* **besitzen usw. usf.**




```
int i , * z1 , * * z0 ;
z0 = & z1 ;
z1 = & i ;
```

i = 7 ; ist äquivalent mit *** z1 = 7 ;** ist äquivalent mit *** * z0 = 7 ;**