

Inhalt

14	Zeiger und Funktionen	14-2
14.1	Zeiger als Funktionswert.....	14-2
14.2	Zeiger auf Funktionen	14-5

14 Zeiger und Funktionen

Zeiger können als Parameter von Funktionen oder auch als Funktionswert übergeben werden. Sie können aber auch auf Funktionen zeigen.

⇒ **Im Zusammenhang mit Funktionen kann man vereinbaren:**

Zeiger als Funktionswert: **int * f (...);**
Zeiger auf Funktionen: **int (* f) (...);**

14.1 Zeiger als Funktionswert

Ohne Einschränkung sind Funktionen, die einen Zeiger als Funktionswert liefern, zulässig.

Deklaration eines Zeigers als Funktionswert:

Syntax: *Typ * Funktionsname ([Parametertyp [Parametername] , ...]) ;*

int * f (int, float);

Semantik:

- Die Funktion mit dem angegebenen *Funktionsnamen* gibt einen Zeiger vom angegebenen *Typ* als Funktionswert zurück.

Im folgenden Beispiel soll in einer Zeichenkette von links beginnend, ein Zeichen gesucht und ein Zeiger auf dieses Zeichen zurückgeliefert werden. Im Fall eines Misserfolges ist es der NULL-Zeiger.

zeichensuchen.c

```
/*
 * Suchen eines Zeichens in einer Zeichenkette
 */
#include <stdio.h>
#define LAENGE 10

char *pos( char *, char);/* Zeiger als Funktionswert */

int main()
{
    char z, s[LAENGE], *p;

    printf( "Eingabe Zeichen: ");
    fflush( stdin); scanf( "%c", &z);
    printf( "Eingabe String: "); scanf( "%9s", s);

    p = pos( s, z); if( p != NULL ) *p = '+';

    printf( "Ausgabe String: %s\n", s);
```

```

    return 0;
}

/*
 * Sucht Zeichen (links -> rechts), gibt die Adresse
 * des 1.gefundenen Zeichens oder NULL zurück
 */
char *pos( char *str, char ze)
{
    while( *str != ze)
        if( *str++ == '\0') return NULL;
    return str;
}

```

Im nächsten Beispiel werden zwei Zeichenketten durch eine Funktion verkettet. Diese liefert einen Zeiger auf die Verknüpfung zurück.

kon1.c

```

/*
 * Verknuepfen zweier Zeichenketten
 */
# include <stdio.h>
# include <stdlib.h> /* malloc */
# include <string.h> /* strlen */
# define LAENGE 10

char *kon( char *, char *);

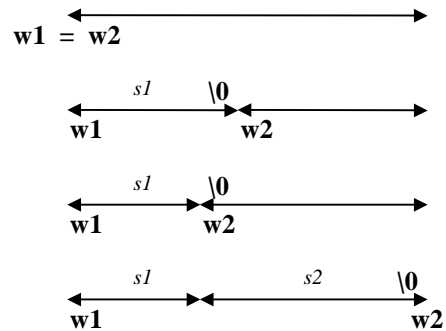
int main()
{
    char z1[ LAENGE], z2[ LAENGE], *s;

    printf( "Eingabe zweier String: ");
    scanf( "%9s%9s", z1, z2);
    s = kon( z1, z2);
    if( s) printf( "String: %s\n", s);

    return 0;
}

char *kon( char *s1, char *s2)
{
    char *w1, *w2;
    w1 = w2 = /* Speicherplatzbereitstellung */
        ( char *) malloc( strlen( s1) + strlen( s2) + 1);
        /* Warnung !!! */
    while( *w2++ = * s1++); /* Uebertragen von s1 */
        /* Ueberschreiben des Zeichenkettenendezeichen */
    w2--;
    while( *w2++ = *s2++); /* Uebertragen von s2 */
    return w1;
}

```



Die Funktion *char *kon (char *, char *)* beschafft Speicher im Heap und setzt einen Zeiger *w1* darauf. Dieser wird dem Zeiger *s* der Hauptfunktion *main()* übergeben, d.h. sein Wert wird in *s* kopiert. Jetzt gibt es Probleme mit der Freigabe, da *w1* in der Funktion selbst noch nicht freigegeben werden kann und in der Hauptfunktion i.R. nicht bekannt ist, dass *s* dynamisch erzeugt wurde! Ein Ausweg wäre, darauf mittels Kommentar zur Funktion *kon* hinzuweisen.

Das obige Programm hätte man auch unter Verwendung zweier Bibliotheksfunktionen, deklariert in `<string.h>`, programmieren können:

```
char *strcpy ( char *, const char * );           /* Stringkopieren */
char *strcat ( char *, const char * );         /* Stringanhängen */
```

Der Wert eines mit *const* vereinbarten Parameters wird in der Funktion nicht verändert.

kon2.c

```
. . .

char *kon( char *s1, char *s2)
{
    char *w1;
                                /* Speicherplatzbereitstellung */
    w1 =
        ( char *) malloc( strlen( s1) + strlen( s2) + 1);
    w1 = strcpy( w1, s1);           /* Stringkopieren */
    w1 = strcat( w1, s2);         /* Stringanhängen */
    return w1;
}
```

14.2 Zeiger auf Funktionen

Funktionen selbst können nicht als Parameter an Funktionen übergeben werden. Es können aber Zeiger auf Funktionen übergeben werden.

Deklaration eines Zeigers auf eine Funktion:

Syntax:

```
[ Funktionstyp ] ( * Zeigername ) ( [ Parametertyp [ Parametername ], ... ] );
```

```
int (* f) ( ... );
```

Semantik:

- Die *Zeigername* zeigt auf eine Funktion mit dem angegebenen *Funktionstyp*, der angegebenen Parameteranzahl und den angegebenen *Parametertypen*.
- Analog zu den Feldern ist der *Funktionsname* ein Zeiger auf die Funktionsdefinition.

<i>Funktionsname</i>	Adresse der Funktionsdefinition
<i>Funktionsname</i> (...)	Funktionsaufruf

⇒ **Leere Parameterlisten müssen mitgeschrieben werden.**

Es soll das Integral einer Funktionen **double f(double);** mittels linearer Interpolation berechnet werden. Für **f** kann wahlweise eine der Standardfunktionen **double sin(double);** bzw. **double cos(double);** oder das neu definierte Polynom **double polynom(double);** zur Berechnung von x^2+1 gewählt werden. Voraussetzung für solch eine Wahlmöglichkeit zwischen mehreren Funktionen ist, dass diese in Parameteranzahl, Parametertypen und Funktionstyp übereinstimmen.

integration.c

```
/*
 * Numerische Integration
 */

# include <stdio.h>
# include <math.h>

/* Zeiger auf Funktion als Parameter*/
double trapezFormel
( double, double, int, double (*)( double));
double polynom( double);

int main()
{
    double ( *f)( double), a, b; /* Zeiger auf Funktion */
    int n, Wahl;

    printf( "Funktion? (1: Sin, 2: Cos, 3: x^2+1)");
    scanf( "%d", &Wahl);
```

```

switch ( Wahl)                                /* Adresszuweisung */
{
  case 1: f = sin; break;    /* Standardfunktionen */
  case 2: f = cos; break;
  case 3: f = polynom; break; /* Benutzerfunktion */
  default: printf( "Wahl unzuulaessig!\n"); return -1;
}

printf( "Intervallgrenzen (untere obere)? ");
scanf( "%lf%lf", &a, &b);

printf(" Anzahl der Teilintervalle? ");
scanf( "%d", &n);

printf( "Integralwert: %g\n",
        trapezFormel( a, b, n, f));

return 0;
}

/*
 * Funktion 'polynom'
 */
double polynom( double x)
{
  return x * x + 1;
}

/*
 * Funktion 'trapezFormel'
 */
double trapezFormel( double a, double b, int n,
                      double (*f)(double))
{
  double s, h; int i;

  h = ( b - a ) / n; s = 0.5 * ( f( a ) + f( b ));
  for ( i = 0; i < n; i++) { a += h; s += f( a );}

  return s * h;
}

```

Die Berechnung des Integrals erfolgt näherungsweise durch lineare Interpolation mit der Trapezformel. Für $n=4$ gilt:

$$\int_a^b f(x)dx \approx \left(\frac{1}{2}(f(a) + f(a+h)) + \frac{1}{2}(f(a+h) + f(a+2h)) + \frac{1}{2}(f(a+2h) + f(a+3h)) + \frac{1}{2}(f(a+3h) + f(b)) \right) \cdot h = \left(\frac{1}{2}(f(a) + f(b)) + \sum_{i=1}^3 f(a+ih) \right) \cdot h \text{ mit } h = \frac{(b-a)}{4}.$$

