Bei den String-Verarbeitungsfunktionen in der Headerdatei <string.h> werden char-Zeiger verwendet, die auf den Anfang des Strings, genauer gesagt auf das erste Zeichen, verweisen.

»strcat()« – Strings aneinanderhängen

```
char *strcat(char *s1, const char *s2);
```

Damit wird s2 an das Ende von s1 angehängt, wobei (logischerweise) das Stringende-Zeichen '\0' am Ende von String s1 überschrieben wird. Voraussetzung ist auch, dass der String s2 Platz in s1 hat.

»strchr()« – ein Zeichen im String suchen

```
char *strchr(const char *s, int ch);
```

Diese Funktion gibt die Position im String s beim ersten Auftreten von ch zurück. Tritt das Zeichen ch nicht auf, wird NULL zurückgegeben.

»strcmp()« – Strings vergleichen

```
int strcmp(const char *s1, const char *s2);
```

Sind beide Strings identisch, gibt diese Funktion 0 zurück. Ist der String s1 kleiner als s2, ist der Rückgabewert kleiner als 0; und ist s1 größer als s2, dann ist der Rückgabewert größer als 0.

»strcpy()« – einen String kopieren

```
char *strcpy(char *s1, const char *s2);
```

Dass hierbei der String-Vektor s1 groß genug sein muss, versteht sich von selbst. Beachte dabei, dass das Ende-Zeichen '\0' auch Platz in s1 benötigt.

»strcspn()« – einen Teilstring ermitteln

```
int strcspn(const char *s1, const char *s2);
```

Sobald ein Zeichen, das in s2 angegeben wurde, im String s1 vorkommt, liefert diese Funktion die Position dazu zurück.

»strlen()« – Länge eines Strings ermitteln

```
size_t strlen(const char *s1);
```

Damit wird die Länge des adressierten Strings s1 ohne das abschließende Stringende-Zeichen zurückgegeben.

»strncat()« – String mit n Zeichen aneinanderhängen

```
char *strncat(char *s1, const char *s2, size_t n);
```

Diese Funktion ist aus Sicherheitsgründen der Funktion strcat() vorzuziehen.

»strncmp()« – n Zeichen von zwei Strings miteinander vergleichen

```
int strncmp(const char *s1, const char *s2, size_t n);
```

Hiermit werden also die ersten n Zeichen von s1 und die ersten n Zeichen von s2 lexikografisch miteinander verglichen. Der Rückgabewert ist dabei derselbe wie schon bei strcmp().

»strncpy()« – String mit n Zeichen kopieren

```
char *strncpy(char *s1, const char *s2, size_t n);
```

Die sicherere Alternative zur Funktion strcpy() lautet strncpy(), die n Zeichen kopiert. Der Ablauf der Funktion ist hingegen wieder derselbe wie bei strcpy()

Hier werden n Zeichen aus dem String s2 in den String s1 ohne das '\0'-Zeichen kopiert.

»strpbrk()« – nach dem Auftreten bestimmter Zeichen suchen

```
char *strpbrk( const char *s1, const char *s2);
```

Die Funktion strpbrk() arbeitet ähnlich wie strcspn(), nur dass hierbei nicht die Länge eines Teilstrings ermittelt wird, sondern das erste Auftreten eines Zeichens in einem String, das im Suchstring enthalten ist.

»strrchr()« – das letzte Auftreten eines bestimmten Zeichens im String suchen

```
char *strrchr(const char *s, int ch);
```

Die Funktion strrchr() ähnelt der Funktion strchr(), nur dass hierbei das erste Auftreten des Zeichens von hinten, genauer gesagt des letzten Zeichens, ermittelt wird.

»strspn()« – das erste Auftreten eines Zeichens, das nicht vorkommt

```
int strspn(const char *s1, const char *s2);
```

Die Funktion strspn() gibt die Position des ersten Auftretens eines Zeichens an, das nicht vorkommt.

»strstr()« – einen String nach dem Auftreten eines Teilstrings durchsuchen

```
char *strstr(const char *s1, const char *s2);
```

Damit wird der String `s1` nach einem String mit der Teilfolge `s2` ohne '\0' durchsucht.

»strtok()« – einen String anhand bestimmter Zeichen zerlegen

```
char *strtok(char *s1, const char *s2);
```

Damit wird der String `s1` durch das Token getrennt, das sich in `s2` befindet. Ein Token ist ein String, der keine Zeichen aus `s2` enthält.