Standard Eingaben in der C-Programmierung

1 Allgemeine Eingaben mit `scanf()`

Mit `scanf()` wird dem C-Programmierer ein mächtiges und flexibles Werkzeug zur Eingabe von Daten über die Standard Eingabe (Tastatur) zur Verfügung gestellt (siehe auch *C_Merkblatt_scanf.pdf*). Das Verhalten von `scanf()` hängt entscheidenden davon ab, welche Formatangaben der Formatstring enthält und was tatsächlich eingegeben wird. Dabei ist zu folgendes berücksichtigen:

- Bei allen gepufferten Tastatur-Eingabefunktion, das sind `scanf()`, `getchar()` und `gets()`, werden Eingaben immer nur vom Eingabepuffer gelesen und interpretiert. Dazu muss erst etwas im Eingabepuffer stehen. Wenn eine dieser Funktion aufgerufen wird und der Eingabepuffer leer ist, dann muss dieser erst befüllt werden. Dazu wird eine Eingabezeile gelesen, indem die ASCII Codes der eingegebenen Zeichen solange hintereinander im Eingabepuffer abgelegt werden, bis diese mit Eingabetaste (LineFeed-Zeichen LF = '\n') abgeschlossen wird. Erst dann werden die Eingaben interpretiert. Das LF-Zeichen (`'\n'`) ist dabei in der Eingabezeile enthalten.
- Wenn der Eingabepuffer nicht leer ist oder zuvor erst befüllt wurde, dann beginnt `scanf()` das vorderste Zeichen im Puffer entsprechend der Formatangabe zu interpretieren. Wenn zum Beispiel mit Format `"%d"` eine Dezimalzahl gelesen werden soll, dann können nur Dezimalziffern '0' – '9', '+' oder '-' als gültige Zeichen interpretiert werden. Andere Zeichen werden abgewiesen und die Eingabe beendet. Ein verwertbares Zeichen wird dem Eingabepuffer entnommen, ein nicht verwertbares Zeichen bleibt im Eingabepuffer stecken. Dies tritt in der Regel dann auf, wenn anstatt Zahlenwerte andere Zeichen eingegeben werden.
- Ist das erste Zeichen im Eingabepuffer bereits ein nicht verwertbares Zeichen, dann erfolgt überhaupt keine Wertzuweisung an die einzulesende Variable. Die Variable bleibt dann unverändert und `scanf()` liefert als Rückgabewert 0, anstatt einer 1 beim Einlesen einer einzigen Variablen. Wird die Eingabe mit Hilfe einer Schleife dann noch auf eine gültige Eingabe überprüft und ggf. wiederholt, dann muss vor einem erneutem Aufruf von `scanf()` der Eingabepuffer erst gelöscht werden. Ansonsten wird das nicht verwertbare Zeichen, dass im vorherigen Aufruf dem Eingabepuffer nicht entnommen wurde, der Eingabefunktion erneut präsentiert, mit demselben Ergebnis, dass dieses wieder nicht entnommen wird. Dadurch kann eine Endlos-Schleife entstehen.
- Leerraumzeichen (sog. Whitespaces, wie Leerzeichen, TAB, LF usw.) im Eingabepuffer werden immer überlesen, d.h. das LF im Eingabepuffer einer vorherigen Eingabe bleibt unbemerkt. Eine Ausnahme macht das Einlesen eines einzelnen Zeichens mit der Formatangabe `"%c"`. In diesem Fall werden auch Leerraumzeichen und damit auch das LF einer vorherigen ohne erneute Eingabe zugewiesen. Abhilfe schafft hier ein Leerzeichen in der Formatangabe `" %c"` (Whitespaces werden überlesen) oder etwa `while((c = getchar()) != '\n' && c != EOF);`

Beispiel:

Mit Hilfe einer Funktion soll eine Ganzzahl in einem gültigen Bereich von 1-10 eingegeben werden. Falls entweder keine Zahl eingegeben wurde oder die eingegebene Zahl nicht im gültigen Intervall liegt, soll die Eingabe wiederholt werden. Unter Eingabe wiederholen ist dabei zu verstehen, dass die Eingabe bei einer Falscheingabe nicht nur einmal, sondern so oft wiederholt werden soll, bis die Eingabe korrekt war. Die Information an den Benutzer, was einzugeben ist, soll der Funktion durch einen String mit übergeben werden. Dieser String wird dann als erstes in der Funktion vor einer Eingabe ausgegeben. Ein Aufruf dieser Funktion sieht zum Beispiel aus wie in Abbildung 1.

```c
#include <stdio.h>

int main()
{
   int zahl;

   zahl = getInt("Geben Sie eine ganze Zahl zwischen [1...10] ein: ");
   printf("Sie haben %d eingegeben\n", zahl);

   return 0;
}
```

Abbildung 1: Aufruf einer Funktion zur Eingabe einer Ganzzahl mit Einschränkung.

Ein erster Versuch einer `getInt()` Funktion, die Eingabe mit Hilfe des Rückgabewertes von `scanf()` zu überprüfen, ob die Eingabe zu einer erfolgreichen Zahlenwandlung geführt hat und diese dann ggf. mit einer fußgesteuerten `do { } while` Schleife zu wiederholen, könnte wie folgt aussehen (Abbildung 2):

```c
int getInt(char *txt)
{
  int n, number;

  do {
      printf("%s", txt);
      n = scanf("%i", &number);
  } while (n == 0);              // Überprüfe, ob Wandlung erfolgreich

  return number;
}
```

Abbildung 2: Erster Versuch einer Funktion zum Einlesen einer Ganzzahl mit Einschränkung.

Dies führt mit mindestens einer gültigen Zahlenziffer zu Beginn der Eingabe zu einer erfolgreiche Zahlenwandlung und korrektem Ergebnis. Falls aber die Eingabe z.B. mit einem Buchstaben beginnt, dann führt dies zu einer Endlosschleife, da dieser Buchstabe dem Eingabepuffer nicht entnommen wird und dieser Buchstabe immer und immer wieder `scanf()` präsentiert wird.

Erst das explizite Löschen des Eingabepuffers nach der Eingabe entnimmt dem Eingabepuffer alle Zeichen und es kann bei einem erforderlichen wiederholten Aufruf von `scanf()` eine korrekte Eingabe erfolgen. Für das Löschen des Eingabepuffers kursieren noch immer Lösungen unter Verwendung von `fflush(stdin)` in verschiedenen Internetforen, allerdings ist diese Funktion für den Eingabekanal stdin undefiniert bzw. wirkungslos. Man muss dazu mit Hilfe einer weiteren `while` Schleife mit `getchar()` dem Eingabepuffer solange Zeichen entnehmen, bis das abschließende Zeilentrenner-Zeichen `'\n'` gefunden wird (siehe Abbildung 3). Vorsicht: das Löschen des Eingabepuffers mit der vorliegenden `while`-Schleife **vor** dem `scanf()`-Aufruf hätte zur Folge, dass beim allerersten Aufruf der Eingabepuffer leer ist und die Eingabe der Eingabetaste angefordert werden würde.

```c
int getInt(char *txt)
{
  int n, number;

  do {
      printf("%s", txt);
      n = scanf("%i", &number);
      while(getchar() != '\n');  // Eingabepuffer löschen
  } while (n == 0);              // Überprüfe, ob Wandlung erfolgreich

  return number;
}
```

Abbildung 3: Verbesserte Version mit Eingabekorrektur aber noch ohne Bereichsüberprüfung.

Erst mit einer zusätzlichen Bereichsüberprüfung, ob die eingegebene Zahl auch im gültigen Bereich ist, ist die Funktion korrekt implementiert (siehe Abbildung 4).

```c
int getInt(char *txt)
{
  int n, number;

  do {
      printf("%s", txt);
      n = scanf("%i", &number);
      while(getchar() != '\n');  // Eingabepuffer löschen
  } while (n == 0 || number < 1 || number > 10);
  // Wiederhole solange n != 1 oder number < 1 oder number > 10

  return number;
```

Abbildung 4: Vollständige Implementierung der Eingabefunktion mit Bereichsüberprüfung [1-10].

2 Eingabe von Zeichen und Text

Die `scanf()` Funktion ist die weitaus komplexeste Eingabefunktion, da sie die Eingabe unterschiedlichster Datentypen erlaubt. Für die Eingabe von nur Zeichen oder Textstrings gibt es auch andere Funktionen, die aber auch ihre Tücken haben.

Beispiele:

- Drei Möglichkeiten, ein einzelnes Zeichen einzulesen:

```c
char z;

scanf("%c", &z); // '\n' bleibt noch im Eingabepuffer stecken
z = getchar();   // liest aus Eingabepuffer, vgl. getc(stdin)
z = getch();     // liest direkt von Tastatur, ohne Ausgabe
```

- Ein Wort einlesen (bis zum ersten Leeraumzeichen):

```
char w[32];
scanf("%s", w);     // Es werden keine Leeraumzeichen übernommen
```

- Eine ganze Zeile (Satz) einlesen, inklusive Leeraumzeichen:

```
char zeile[128];
gets(zeile);       // Es wird eine ganze Eingabezeile übernommen
```

3 Ungepufferte Eingaben mit `getch()` und `_getwch()`

Eine häufig eingesetzte Zeicheneingabe-Funktion ist die `getch()`-Funktion, die allerdings nicht dem C-Standard entspricht und ursprünglich nur auf Windows Plattformen verfügbar war. Die `getch()`-Funktion ist in `<conio.h>` definiert. Sie liest ein einzelnes Zeichen direkt von der Tastatur, ohne dass die Eingabetaste getätigt werden muss. Das Zeichen wird auch nicht in der Standardausgabe ausgegeben (ohne „echo“). Dadurch kann das Eingabeverhalten so ziemlich frei gestaltet werden.

Da sie nicht im C-Standard definiert ist, kommt es auf verschiedenen Plattformen auch zu unterschiedlichem Verhalten. Selbst Microsoft hat mit Einführung von Windows 10 das Eingabeverhalten in der Eingabeaufforderung (Konsole) und damit auch das Verhalten der `getch()` Funktion geändert. So erlaubt Windows 10 mit STRG-C und STRG-V Texte in Eingabeaufforderungen zu kopieren und einzufügen, wo man zuvor mit der rechten Maustaste das Menü heranziehen musste. Da die ASCII Codes für STRG-C (0x03) und STRG-V (0x16) im unteren ASCII Bereich von 0-31 liegen, wurden die entsprechenden Codes maskiert, d.h. zuvor ausgegeben Sonderzeichen erfolgen nicht mehr. Ferner werden seit dem letzten Windows Update von `getch()` von 2018 tatsächlich zwei Character übergeben, die einen zweiten Aufruf von `getch()` erfordern. Dies scheint offensichtlich ein Bug zu sein, der in späteren Version von Windows ab 2020 wieder behoben wurde[1]. Dies gilt auch für die seit geraumer Zeit empfohlene Funktion `_getch()`. Ein Workaround zur Lösung des Problems besteht darin, die ebenfalls in `<conio.h>` definierte Funktion `_getwch()` = GetWideCharacter zu benutzen. Diese liefert auch 16 bit Unicode Character.

4 Interpretation der Eingabetaste und des Zeilenumbruchs

Potential für Inkompatibilitäten auf verschiedenen Plattformen liefert die Interpretation des ASCII Codes der Eingabetaste, mit der eine Eingabe durch einen Zeilenumbruch bestätigt wird. Der Begriff Zeilenumbruch stammt noch aus Zeiten der elektronischen Textverarbeitung, bei der Textein- und ausgaben zeilen- und auch meist spaltenweise organisiert ist. Dies ist auch bei der einfachen Eingabeaufforderung der Fall. Dort sind Ausgaben zeilenweise organisiert und solange eine Ausgabe noch nicht mit einem Zeilenvorschub bestätigt worden ist, kann diese noch nachträglich verändert werden. Auch die meisten Eingaben sind zeilenweise organisiert, so dass Eingaben in der Regel in Einheiten von Eingabenzeilen verarbeitet werden. Dazu müssen bei allen gepufferten Eingabefunktion, wie `scanf()`, `getchar()` und `gets()` auch einfache Eingaben mit der „Eingabetaste“ abgeschlossen

[1] https://developercommunity.visualstudio.com/content/problem/247770/-getch-broken-in-vs-157.html

bzw. bestätigt werden. Die Eingabetaste schließt dazu die Eingabezeile ab liefert dazu einen Zeilenvorschub. Die Darstellung dieses Zeilenvorschub-Zeichens im Eingabepuffer bzw. Eingabestring ist allerdings in verschiedenen Betriebssystemen unterschiedlich.

Prinzipiell unterscheidet man in Anlehnung an die Terminologie einer Schreibmaschine zwischen[2]:

- **Wagenrücklauf** – Positionierung der Schreibstelle zum Zeilenanfang (ganz links).
- **Zeilenvorschub** – Positionierung der Schreibstelle um eine Zeile nach unten.

Bei der Entwicklung des ASCII Zeichensatzes hat man dies berücksichtigt und deshalb zwei getrennte Zeichen eingeführt:

- Das Steuerzeichen für den Zeilenvorschub (englisch line feed, kurz **LF**) ist als ASCII-Zeichen 10 (0x0A) kodiert. Manche Systeme erlauben es, das LF-Zeichen mit der Tastenkombination Strg+J einzugeben.
- Das Steuerzeichen für den Wagenrücklauf (englisch carriage return, kurz **CR**) ist als ASCII-Zeichen 13 (0x0D) kodiert. Manche Systeme erlauben es, das CR-Zeichen mit der Tastenkombination Strg+M einzugeben.

Die gängigen aktuellen Betriebssysteme verwenden die folgenden ASCII-Codes für den Zeilenumbruch:

Betriebssystem	Abkürzung	Code Hex	Code Dezimal	Escape-Sequenz
Unix, Linux, Android, macOS,	LF	0A	10	\n
Windows, DOS,	CR LF	0D 0A	13 10	\r\n

In der Standard C Library ist der Zeilenumbruch als Escape-Sequenz `'\n'` (LF) definiert. Bei im Textmodus geöffneten Dateien erfolgt eine Übersetzung von `'\n'` in die auf dem jeweiligen System üblichen Steuerzeichen für den Zeilenumbruch. Somit erfolgt in unixartigen Betriebssystem keine Umsetzung, da dort LF bereits für den Zeilenumbruch steht. Dagegen findet unter Windows eine Substitution durch CR LF statt. Die resultierenden Dateien sind folglich nicht identisch. Bei Eingaben über die Tastatur muss deshalb beim Überprüfen auf den Zeilenumbruch unterschieden werden, ob die Eingabe über ein Standard C-Funktion, wie `scanf()` oder `getchar()` erfolgt, oder über `getch()`. Während bei den Standard C-Funktion der Zeilenumbruch (Eingabetaste) als LF (10) erkannt wird, ist es bei der Windows-spezifischen `getch()` Funktion das erste der in Windows benutzten Sequenz von CR LF, also nur das CR (13).

Der Einsatz der `getch()` Funktion ist deshalb mit Vorsicht zu genießen und Bedarf einer Anpassung auf die jeweilige Plattform. Unixartige Betriebssystem mit dem gcc Compiler, wie er beispielsweise in Moodle, Code Blocks oder auch in Repl.it eingesetzt wird, verwenden nur das LF, während der Visual Studio Compiler in Windows Betriebssystemen CR LF als Zeilenumbruch verwenden.

[2] https://de.wikipedia.org/wiki/Zeilenumbruch