Fachhochschule Bingen

Programmieren

Ein- und Ausgabe

Prof. Dr. Maximilian Mengel,
Professur Programmiermethodik,
Grundlagen der Informatik und Multimedia
Gebäude 1, Raum 212
Tel.: 06721-409 152
E-Mail: mengel@fh-bingen.de

Formatierte Ein- und Ausgabe

- In C sind die Ein- und Ausgabe nicht als Teil der Sprache definiert.
- In der Standard Bibliothek `stdio.h` sind unter anderem folgende Funktionen zur formatierten Ein- und Ausgabe enthalten
  - printf() Die formatierte Ausgabe auf der sogenannten Standardausgabe (Bildschirm)
  - scanf() Die formatierte Eingabe von der sogenannten Standardeingabe (Tastatur)

Printf()

- Die Funktion printf() besitzt eine variable Anzahl an Parametern (größer 0)
- Der erste Parameter ist der sogenannte **Formatstring**
- Weitere optionale Parameter beinhalten eventuell auszugebende **Argumente**

Der Printf()-Formatstring

- Der Formatstring enthält:
  - Text (z.B.: `Formatstring = "Hello, world"`)
  - Formatelemente (z.B. `Formatstring ="%d"`)
  - Formatelemente beginnen immer mit eine %-Zeichen
- In einem Formatstring können neben dem Text beliebig viele Formatelemente enthalten sein:
  - z.B.: Formatstring = "ggT(%d,%d) = %d\n"

Printf()-Argumente

- Abhängig von dem Formatstring muß die Funktion printf() in Anzahl und Typ korrespondierende Argumente besitzen
  - `printf("ggT(%d,%d) = %d\n",a,b,g);`
  - `printf("Herr %s verdient %d Euro\n", "Müller",3000);`
  - `printf("5 durch 2 = %f\n",5.0/2.0);`

Zuordnung der Formatelemente zu Typen

Zeichen	Typ
d, i	int, als Dezimalzahl
o	int, als Oktalzahl
x,X	int, als Hexadezimalzahl
u	int, als vorzeichenlose Zahl
c	character, int, ein einzelnes Zeichen
s	char *, string, eine Zeichenkette
f	float, double, Gleitpunktzahl als [-]m.dddd
e, E	float, double, Gleitpunktzahl in Expotentialdarstellung: [-]m.dddd e[-]xx
g, G	float, double, Darstellung wie %f oder %e je nachdem welche Darstellung kürzer ist
p	pointer, Darstellung segment:offset
%	Ausgabe des %-Zeichens

Weitere Formatelemente

- Zwischen dem %-Zeichen und den genannten Formatelementen können noch folgende Werte eingefügt werden:

-	erzwingt linksbündige Ausgabe
+	erzwingt die Ausgabe des Vorzeichens
␣	ein Leerzeichen erzwingt ein Leerzeichen vor positiven Zahlen; bei negativen Zahlen wird das - Zeichen ausgegeben
Zahl	Die minimale Feldlänge
.Zahl	Bei f,g,G die Anzahl der Nachkommastellen; bei s die Anzahl der darzustellenden Zeichen

Beispiele

- Integer

"#%d#",15	→ #15#
"#%o#",15	→ #17#
"#%x#",15	→ #F#
"#%u#",-15	→ #4294967281#
"#%4d#",15	→ # 15#

- Float

"#%f#",15.345	→ #15.345#
"#%5.2f#",1.2345	→ # 1.23#
"#%g#",0.001	→ #1e-3#
"#%g#",0.1	→ #0.1#

Beispiele

- Strings

“#%10s#“,“Hello, World“	→ `#Hello, World#`
“#%.10s#“,“Hello, World“	→ `#Hello, Wor#`
“#%20s#“,“Hello, World“	→ `#  Hello, World#`
“#%-20s#“,“Hello, World“	→ `#Hello, World  #`
“#%20.10s#“,“Hello, World“	→ `#  Hello, Wor#`
“#%-20.10s#“,“Hello, World“	→ `#Hello, Wor  #`

Formatierte Ausgabe: Steuerzeichen

- Eine printf() Anweisung gibt Zeichen auf dem Bildschirm an der aktuellen Cursor-Position aus. Insbesondere wird der Cursor nach der Ausgabe **nicht** neu positioniert (z.B. an den Zeilenanfang der nächsten Zeile).
- Das Positionieren des Cursor geschieht durch Steuerzeichen:
  - `“\n“` erzeugt einen Zeilenwechsel
  - `“\t“` erzeugt ein Tabulatorzeichen
  - `“\b“` erzeugt ein backspace

Formatierte Eingabe

- Zur formatierten Eingabe wird die Funktion scanf(); benutzt
- scanf() besitzt wie printf() einen Formatstring und die **Adressen** der Variablen, in die eingelesen werden soll
- Die Adresse einer Variable bestimmt man mittels des Adressoperators **&**
- Die Funktion scanf() gibt einen numerischen Wert zurück, der angibt, wie viele Argumente eingelesen wurden.

Wesentliche Unterschiede zwischen printf() und scanf()

- printf()
  - Argumente: Variablen oder Werte
  - Ausgabe des im Formatstring enthaltenen Textes zusammen mit den formatierten Argumenten
  - Gleitpunktwerte per default als double
- scanf()
  - Argumente: **Adressen** von Variablen
  - Es darf kein Text in dem Formatstring enthalten sein; der Formatstring enthält nur Formatelemente
  - Gleitpunktwerte per default als float, sonst lf, lg oder le

Stolperfallen bei scanf()

- Nicht gelesene Zeichen bleiben im Eingabepuffer stehen und werden dementsprechend beim nächsten scanf() eingelesen
  - scanf("%2d",&i) liest bei Eingabe von 123 nur 12 ein; die 3 bleibt im Eingabepuffer
- Leerzeichen, Tabulatoren oder Zeilenende-Zeichen trennen mehrere Eingaben
  - scanf("%s",Name) liest bei der Eingabe von „Hans Dampf" nur „Hans" in den String mit der Bezeichnung Name ein

Stolperfallen bei scanf()

- Es muß immer eine Adresse angegeben werden
  - scanf("%d",i) führt dazu, daß der Wert von i als Adresse interpretiert wird und versucht wird die Eingabe an diese Stelle zu schreiben!
- Um ein einzelnes Zeichen einzulesen benutzt man i.Allg. "%1s" und nicht "%c", da %c keine Leer- oder Zeilenendezeichen überliest:
  - `printf("Zum Beenden Q eingeben:\n");`
    `scanf("%1s",&weiter);`

Programme / `main()`

- In C befinden sich Anweisungen nur innerhalb von Funktionen.
- Eine ausgezeichnete Funktion, die in jedem C-Programm existieren muß, ist die Funktion **`main()`**
  - Die Funktion **`main()`** muß genau einmal innerhalb eines Programmes existieren
  - Die Funktion **`main()`** wird automatisch beim Starten des Programmes aufgerufen.

Variablendeklaration

- Um in einem Programm mit Variablen arbeiten zu können müssen diese deklariert werden
- Variablen sind ab Ihrer Deklaration bis zum Ende des sie umgebenden Anweisungsblocks gültig.
- Globale Variablen (außerhalb einer Funktion deklariert) sollten nur in absolut unumgänglichen Fällen genutzt werden
- Variablen deklariert man häufig in einer Funktion oder der main() direkt nach der ersten Klammer {
  - Ausnahme: Zählvariablen werden häufig erst direkt vor der Benutzung deklariert

Beispiel:

```
/* Beispiel Ausgabe einer Tabelle sin(2*Pi*x) */
#include <stdio.h>
#include <math.h>
main()
{
   float xmax, x = 0.0;
   const float PI = 3.1415926;
   printf("Geben Sie X (0.0< X <=1.0) ein:\n");
   scanf("%f",&xmax);
   if (xmax > 1.0 || xmax < 0.0) printf("Fehlerhafte Eingabe!\n");
   else
     while (x <= xmax)
     {
       printf("Sin(2Pi*%.2f)= %.3f\n",x,sin(2*PI*x));
       x += 0.05;
     }
   scanf("%f",&x);
}
```