Bölüm 9 – Formatlı Girdi/Çıktı

İçerik

9.1 Giriş

- Bu Bölümde Açıklananlar:
  - Sonuçların gösterimi
  - **`scanf`** ve **`printf`**
  - Cümleler (girdi ve çıktı)
    - **`gets`**, **`puts`**, **`getchar`**, **`putchar`** (in **`<stdio.h>`**)

- Cümleler
  - Satırlar halinde düzenlenen karakter dizileri
    - Her bir satır sıfır yada daha fazla karakterden oluşur ve \n karakteri ile sona erer
    - ANSI C en az 254 karakterli satırları desteklemelidir
  - Tüm girdi ve çıktıları işler
  - Sıklıkla
    - Standard girdi– klavyeye
    - Standard çıktı– ekrana
    - Standard hata– ekrana yönlendirilir
    - Detaylar Bölüm 11 de

9.3 `printf` ile Formatlı Çıktı

- **`printf`**
  - Duyarlı Çıktı Formatı
    - Dönüşüm belirteçleri: bayrak, alan uzunluğu, duyarlılık, v.s.
  - Yuvarlama, sütun ayarlama, sağ/sol yaslama, harf karakterleri ekleme, üstel format, hexadecimal format, sabit uzunluk ve duyarlılık işlemleri yapılabilir
- Format
  - **`printf(`** *format-kontrol-stringi*, *diğer-argümentler* **`);`**
  - Format kontrol stringi: çıktı formatını açıklar
  - Diğer-argümentler: format kontrol stringindeki her bir dönüşüm belirtecine karşılık gelir
    - Her bir belirteç yüzde işareti(`%`) ile başlar, dönüşüm belirteci ile biter

9.4 Tam (int) sayıları Yazma

- Tamsayı (Integer)
  - Desimal kısımsız sayı: 25, 0, -9
  - Pozitif, negatif, veya sıfır
  - Sadece negatif işaret yazılır (default)

Dönüşüm belirteci	Açıklama
`d`	İşaretli tamsayı
`i`	İşaretli tamsayı. (*Not:* **`i`** ve **`d`** belirteci **`scanf`** de farklıdır.)
`o`	Işaretsiz octal tamsayı
`u`	İşaretsiz tamsayı
`x` veya **`X`**	İşaretsiz hexadecimal tamsayı. **`X`** ile **`0-9`** rakamları ve **`A-F`** harfleri, **`x`** ile **`0-9`** rakamları ve **`a-f`** harfleri görüntülenir.
`h` veya **`l`**	Belirtecin önüne yazılır ve tamsayının **`short`** (kısa) veya **`long`** (uzun) gösterimini sağlar. **`h`** ve **`l`** harfleri *uzunluk düzenleyicileri* olarak adlandırılır

```c
/* Fig 9.2: fig09_02.c */
/* tamsavı dönüsüm belirtecleri kullanımı */
#include <stdio.h>

int main()
{
   printf( "%d\n", 455 );
   printf( "%i\n", 455 );  /* printf de i ve d aynıdır */
   printf( "%d\n", +455 );
   printf( "%d\n", -455 );
   printf( "%hd\n", 32000 );
   printf( "%ld\n", 2000000000 );
   printf( "%o\n", 455 );
   printf( "%u\n", 455 );
   printf( "%u\n", -455 );
   printf( "%x\n", 455 );
   printf( "%X\n", 455 );

   return 0;
}
```

```
455
455
455
-455
32000
2000000000
707
455
65081
1c7
1C7
```

9.5 Reel (float) Sayıları Yazma

- Reel (Float) sayılar
  - Desimal noktası vardır (**`33.5`**)
  - Üstel notasyon (bilimsel notasyonun bilgisayar gösterimi)
    - **`150.3 =1.503 x 10² (bilimsel)`**
    - **`150.3`** = **`1.503E+02`** (üstel) (**`E`** eksponent(üs) den geliyor)
    - **`e`** veya **`E`** kullanılır
  - **`f`** – desimal solunda en az bir rakam gösterimi belirteci
  - **`g`** (veya **`G`**) – **`f`** veya **`e`** nin duyarsız sıfırları atılmış gösterimi için (**`1.2300`** yerine **`1.23`** yazılır)
    - **`-4`** den küçük veya duyarlılıktan büyük veya eşit ise üs belirteci kullan (default durumu **`6`** rakamdır)

9.5 Reel (float) Sayıları Yazma

```
1  /* Fig 9.4: fig09 04.c */
2  /* Floating-point sayıların
3     floating-point belirteci ile yazımı */
4
5  #include <stdio.h>
6
7  int main()
8  {
9     printf( "%e\n", 1234567.89 );
10    printf( "%e\n", +1234567.89 );
11    printf( "%e\n", -1234567.89 );
12    printf( "%E\n", 1234567.89 );
13    printf( "%f\n", 1234567.89 );
14    printf( "%g\n", 1234567.89 );
15    printf( "%G\n", 1234567.89 );
16
17    return 0;
18 }
```

```
1.234568e+006
1.234568e+006
-1.234568e+006
1.234568E+006
1234567.890000
1.23457e+006
1.23457E+006
```

9.6 String ve Karakterleri Yazma

- **`c`**
  - **`char`** argumentini yazar
  - Bir stringin ilk karakterini yazmak için kullanılamaz
- **`s`**
  - **`char`** argumentine bir pointer ister
  - **`NULL`** (**`'\0'`**) bulununcaya kadar olan karakterleri yazar
  - **`char`** argumentini yazamaz

Uyarı

- Karakter sabitleri için tek tırnak (**'z'**)
- Stringler için çift tırnak **"z"** (aslında iki karakter içerir; **'z'** ve **'\0'**)

9.6 String ve Karakterleri Yazma

```c
/* Fig 9.5: fig09 05c */
/* string ve karakterleri yazma*/
#include <stdio.h>

int main()
{
   char karakter = 'A';
   char string[] = "Bu bir stringdir";
   const char *stringPtr = "Bu da bir stringdir";

   printf( "%c\n", karakter );
   printf( "%s\n", "Bu bir stringdir" );
   printf( "%s\n", string );
   printf( "%s\n", stringPtr );

   return 0;
}
```

```
A
Bu bir stringdir
Bu bir stringdir
Bu da bir stringdir
```

9.7 Diğer Dönüşüm Belirteçleri

- **`p`**
  - Pointer değerini (adresini) gösterir
- **`n`**
  - O anki **`printf`** deyimindeki çıktı karakterlerinin sayısını alır
  - Argüment olarak bir tamsayıya bir pointer alır
  - **`%n`** belirteci ile hiç bir şey basılmaz
  - Her **`printf`** çağrımı bir değer gönderir
    - Çıktı karakterlerinin sayısı
    - Hata durumunda negatif bir sayı
- **`%`**
  - Yüzde işaretini yazar
  - **`%%`**

```c
/* Fig 9.7: fig09_07.c */
/* The p, n, ve % dönüsüm belirtecleri kullanımı */
#include <stdio.h>

int main()
{
   int *ptr;
   int x = 12345, y;

   ptr = &x;
   printf( " ptr nin değeri %p\n", ptr );
   printf( " x in adresi %p\n\n", &x );

   printf( "Bu satırda basılan toplam karakter sayısı:%n", &y );
   printf( " %d\n\n", y );

   y = printf( "Bu satır 28 karakterdir\n" ); /* bosluk dahil */
   printf( "%d=yazılan karakter sayısı\n\n", y );

   printf( "Bir format kontrol stringinde %% yazmak\n" );

   return 0;
}
```

```
Ptr nin değeri 0065FDF0
x in adresi 0065FDF0

Bu satırda basılan toplam karakter sayısı: 43

Bu satır 24 karakterdir
24=yazılan karakter sayısı

Bir format kontrol stringinde % yazmak
```

9.8 Alan Uzunluklu ve Duyarlı Yazım

- Alan Uzunluğu
  - Verinin yazıldığı alanın uzunluğu
  - Uzunluk veri boyundan fazla ise, (default) sağa yaslar
    - Uzunluk çok küçük ise, veriyi sığdıracak şekilde alanı büyütür
    - Negatif işareti bir birim yer alır
  - Tamsayı uzunluğu `%` ve belirteç arasına yazılır
  - **`%4d`** – alan uzunluğunu 4 birim alır

Duyarlılık

- Veri türüne göre anlamı değişir
- Tamsayılar (default **1**)
  - Basılacak minimum rakam sayısı
    - Veri çok küçük ise, önkısımları sıfır ile doldurur
- Reel sayı
  - Noktadan sonra görünecek rakam sayısı(**e** ve **f**)
    - **g** için– anlamlı rakamların maksimum sayısı
- String
  - Stringden yazılacak maksimum karakter sayısı
- Format
  - **%** den sonra nokta (**.**) ve sonra duyarlılık sayısı
    - **%.3f**

Alan uzunluğu ve duyarlılık

- Birlikte verilebilir
  - **%uzunluk.duyarlılık**
    - **%5.3f**
- Negatif alan uzunluğu– sola yasla
- Pozitif alan uzunluğu– sağa yasla demektir
- Duyarlılık pozitif olmalıdır
- Alan uzunluğunu ve duyarlılık sayısını belirtmek için tamsayı kullanılabilir
  - Alan uzunluğu ve duyarlılık yerine yıldız (*) yaz
  - Argüment listesinde bir **int** argümentine karşılık getir
  - Örnek:
  - **printf("%*.*f", 7, 2, 98.736);**

```c
/* Fig 9.9: fig09_09.c */
/* Yazımda duyarlılık kullanımı: tamsayılar,
   reel sayılar, ve stringler */
#include <stdio.h>

int main()
{
   int i = 873;
   double f = 123.94536;
   char s[] = "Mutlu yıllar";

   printf( "tamsayıda duyarlılık kullanımı\n" );
   printf( "\t%.4d\n\t%.9d\n\n", i, i );
   printf( "reel sayıda duyarlılık kullanımı\n" );
   printf( "\t%.3f\n\t%.3e\n\t%.3g\n\n", f, f, f );
   printf( "stringde duyarlılık kullanımı\n" );
   printf( "\t%.10s\n", s );

   return 0;
}
```

```
tamsayıda duyarlılık kullanımı
        0873
        000000873

Reel sayıda duyarlılık kullanımı
        123.945
        1.239e+02
        124

Using precision for strings
        Mutlu yıll
```

9.9 `printf` Format-Kontrol Stringinde Bayrak Kullanımı

- Bayraklar
  - Ek formatlama kapasitesi
  - Yüzde işaretinden hemen sonra bayrak kullan
  - Birçok bayrak aynı anda kullanılabilir

Bayrak	Açıklama
- (eksi işareti)	Çıktıyı sola yaslar.
+ (artı işareti)	Pozitif sayı önüne + işaretini yazar
boşluk	+ ile yazılmayan pozitif sayı önüne boşluk bırakır
#	Octal belirteç **o** ile kullanıldığında çıktının önüne **0** yazar Hexadecimal belirteç **x** veya **X**. ile kullanıldığında çıktının önüne **0x** veya **0X** yazar Ondalık kısmı sıfır olan **e**, **E**, **f**, **g** veya **G** belirteçli reel sayılarda ondalık kısmı görüntüler. (Normalde desimal kısım sıfır ise görüntülenmez.) **g** ve **G** belirtecinde, anlamsız sıfırlar atılmaz.
0 (zero)	Alan boşluğunu sıfırlar ile doldurur.

9.9 printf Format-Kontrol Stringinde Bayrak Kullanımı

```
1  /* Fig 9.11: fig09_11.c */
2  /* Sağa ve sola yaslama */
3  #include <stdio.h>
4
5  int main()
6  {
7     printf( "%10s%10d%10c%10f\n\n", "merhaba", 7, 'a', 1.23 );
8     printf( "%-10s%-10d%-10c%-10f\n", "merhaba", 7, 'a', 1.23 );
9     return 0;
10 }
```

```
   merhaba         7         a  1.230000

merhaba     7         a         1.230000
```

9.9 printf Format-Kontrol Stringinde Bayrak Kullanımı

```
1  /* Fig 9.14: fig09_14.c */
2  /* # bayrağı kullanımı:
3     o, x, X ve herhangi reel sayı belirteci ile */
4  #include <stdio.h>
5
6  int main()
7  {
8     int c = 1427;
9     double p = 1427.0;
10
11    printf( "%#o\n", c );
12    printf( "%#x\n", c );
13    printf( "%#X\n", c );
14    printf( "\n%g\n", p );
15    printf( "%#g\n", p );
16
17    return 0;
18 }
```

```
02623
0x593
0X593

1427
1427.00
```

9.10 Cümlelerin ve Escape-Dizilerinin Yazımı

- Cümle yazımı
  - Çoğu karakterler yazılabilir
  - Bazı "problem" karakterleri, çift tırnak (") gibi
  - Escape dizisi ile t verilmelidir
    - \ ve ardından escape karakteri yazılır

Escape dizisi	Açıklama
\'	Tek tırnak (') karakterini çıktı birimine yazar.
\"	Çift tırnak (")karakterini çıktı birimine yazar.
\?	? karakterini çıktı birimine yazar.
\\	backslash (\)karakterini çıktı birimine yazar.
\a	Zil sesine (bip) neden olur
\b	Imleci (cursor) bir pozisyon geri kaydırır
\f	İmleci bir sonraki mantıksal sayfa başına getirir.
\n	İmleci bir sonraki satır başına getirir.
\r	İmleci aynı satırın başına getirir.
\t	İmleci bir sonraki yatay tab pozisyonuna getirir.
\v	İmleci bir sonraki dikey tab pozisyonuna getirir.

9.11 `scanf` ile Formatlı Girdi

- **`scanf`**
  - Girdi formatı
  - Kapasiteleri
    - Her tip veri girişi
    - Özel karakter girişi
    - Özel karakter atlaması
- Format
  - **`scanf`**(*format-kontrol-stringi*, *diğer-argümentler*);
  - Format-kontrol-stringi
    - Girdilerin formatını tanımlar
  - Diğer argümentler
    - Girdilerin yükleneceği değişkenlere pointerlar
  - Verilerden belli sayıda karakterler okumak için alan uzunluğu içerebilir

9.11 `scanf` ile Formatlı Girdi

Belirteçler	Açıklama
Tamsayılar	
`d`	Tamsayı okur. Karşılık gelen argüment tamsayıya pointerdır.
`i`	Tamsayı, octal veya hexadecimal tamsayı okur. Karşılık gelen argüment tamsayıya pointerdır.
`o`	octal tamsayı okur. Karşılık gelen argüment işaretsiz tamsayıya pointerdır
`u`	İşaretsiz tamsayı okur. Karşılık gelen argüment işaretsiz tamsayıya pointerdır
`x` veya **`X`**	hexadecimal tamsayı okur. Karşılık gelen argüment işaretsiz tamsayıya pointerdır
`h` veya **`l`**	Girdinin **`short`** veya **`long`** tamsayı olmasına göre, belirtecin önünde yazılır.
Reel sayılar	
`e`, **`E`**, **`f`**, **`g`** veya **`G`**	Reel sayı okur. Karşılık gelen argüment reel sayıya pointerdır
`l` veya **`L`**	**`double`** veya **`long double`** olmasına göre, belirtecin önünde yazılır.

9.11 `scanf` ile Formatlı Girdi

Belirteç	Açıklama
Karakter ve string	
`c`	Karakter okur. Karşılık gelen argüment **`char`** a bir pointerdır. Null (**`'\0'`**) karakteri eklenmez.
`s`	String okur. Karşılık gelen argüment stringi ve null (**`'\0'`**) karakteri(otomatik olarak eklenir) içeren **`char`** tipinde bir diziye pointerdır.
Tarama(scan) seti	
[scan karakterleri	Bir dizide bulunan karakterlerin bir kümesi için bir string tarar
Diğerleri	
`p`	Bir adres bir **`printf`** deyiminde **`%p`** ile üretildiğinde oluşan aynı formdan bir adres okur.
`n`	**`scanf`** deki girdi karakterlerinin sayısını alır. Karşılık gelen argüment tamsayıya pointerdır
`%`	Girdideki yüzde (**`%`**) işaretini atlar.

9.11 `scanf` ile Formatlı Girdi

- Tarama (Scan) seti
  - Köşeli parantezler (**[ ]**)içindeki karakter kümesi
    - **%** işaretini takip eder
  - Girdinin sadece scan setinde bulunan karakterlerini tarar
    - Her bulduğunu belirli bir diziye yükler
    - Scan setinde olmayan ilk karakter bulunduğunda tarama durur
  - Tümleyen tarama seti
    - Tümleyen için şapka (**^**) kullanılır: **[^aeiou]** gibi
    - Scan set içinde olmayan karakterleri yükler
- Karakterleri atlama
  - Format kontrolde atlanacak karakteri içerir
  - Veya, ***** (atama iptali karakteri) kullanılır
    - Her hangi tip karakteri yüklemeden atlar

9.11 scanf ile Formatlı Girdi

```c
/* Fig 9.20: fig09_20.c */
/* Karakter ve string okuma */
#include <stdio.h>

int main()
{
   char x, y[ 9 ];

   printf( "Bir string gir: " );
   scanf( "%c%s", &x, y );

   printf( "Girdi:\n" );
   printf( "Karakter: \"%c\" ", x );
   printf( "ve string: \"%s\"\n", y );

   return 0;
}
```

```
Enter a string: Pazar
Girdi:
Karakter: "P" ve string: "azar"
```

9.11 scanf ile Formatlı Girdi

```
1  /* Fig 9.22: fig09 22.c */
2  /* Tümleven scan set */
3  #include <stdio.h>
4
5  int main()
6  {
7     char z[ 9 ] = { '\0' };
8
9     printf(" Bir string gir: " );
10    scanf( "%[^aeiou]", z );
11    printf(" Girdi: \"%s\"\n", z );
12
13    return 0;
14 }
```

```
Bir string gir: String
Girdi: "Str"
```

9.11 scanf ile Formatlı Girdi

```c
/* Fig 9.24: fig09_24.c */
/* Girdiden karakterleri okuma ve atma */
#include <stdio.h>

int main()
{
   int av1, gun1, yil1, av2, gun2, yil2;

   printf( " av-gün-yıl formunda bir tarih gir : " );
   scanf( "%d%*c%d%*c%d", &av1, &gun1, &yil1 );
   printf( "Av = %d  gün = %d  yıl = %d\n\n",
      av1, gun1, yil1 );
   printf( " av/gün/yıl formunda bir tarih gir : " );
   scanf( "%d%*c%d%*c%d", &av2, &gun2, &yil2 );
   printf( "av = %d  gün = %d  yıl = %d\n",
      av2, gun2, yil2 );

   return 0;
}
```

```
ay-gün-yıl formunda bir tarih gir : 11-18-2004
ay = 11  gün = 18  yıl = 2004

ay/gün/yıl formunda bir tarih gir : 11/18/2004
ay = 11  gün = 18  yıl = 2004
```