C’de Formatlı Giriş/Çıkış

Herhangi bir sorunun çözümünün önemli bir kısmı sonuçların sunumudur. Bu bölümde, standart giriş ve çıkıştan gelen verileri sırasıyla standart çıkış ve giriş için “Printf” ve “Scanf”'ın biçimlendirme özelliklerini derinlemesine tartışacağız. Bu fonksiyonları çağıran programlara <stdio.h> kütüphanesini eklemek gerekir.

Tüm giriş ve çıkış, bayt dizileri olan akışlarla (streems) gerçekleştirilir. Girdi işlemlerinde, baytlar bir aygıttan (örn. klavye, sabit disk, ağ bağlantısı) ana belleğe akar. Çıktı işlemlerinde, baytlar ana bellekten bir aygıta (örneğin, bir görüntü ekranı, bir yazıcı, bir sabit disk, bir ağ bağlantısı vb.) akar. Program başladığında, programa otomatik olarak üç akış bağlanır. Normal olarak, standart giriş akışı klavyeye ve standart çıkış akışı ekrana bağlıdır. Üçüncü bir akış, standart hata akışı da ekrana bağlıdır. İşletim sistemleri genellikle bu akışların başka cihazlara yönlendirilmesine izin verir.

Doğru çıktı biçimlendirmesi “printf” ile gerçekleştirilir. Her “printf” çağrısı, çıktı biçimini açıklayan bir biçim kontrol dizesi içerir. Biçim kontrol dizesi, dönüştürme belirticilerinden, bayraklardan, alan genişliklerinden, ve gerçek karakterlerden oluşur. Yüzde işareti (%) ile birlikte bunlar dönüştürme belirtimlerini oluşturur. “printf” fonksiyonu, her biri bu bölümde ele alınan aşağıdaki biçimlendirme işlemlerini gerçekleştirebilir:

1. Kayan noktalı değerleri belirtilen sayıda ondalık basamağa yuvarlama.
2. Bir sayı sütununu ondalık basamakları üst üste gelecek şekilde hizalama.
3. Çıktıların sağa, sola yaslaması.
4. Bir çıktı satırındaki kesin konumlara değişmez karakterler ekleme.
5. Kayan noktalı sayıların üstel biçimde gösterilmesi.
6. İşaretsiz tamsayıları sekizli ve onaltılık biçimde temsil etme.
7. Her türlü veriyi sabit boyutlu alan genişlikleri ve kesinlikleriyle görüntüleme.

….. gibi.
“printf” fonksiyonunun genel halini şöyle ifade edebiliriz;

printf(*format-control-string*, *other-arguments*);

“printf” fonksiyonu, çıktı biçimini açıklayan format-kontrol-dizesi biçimine sahiptir ve diğer-argümanlar (isteğe bağlı olan), biçim-kontrol-dizesindeki her dönüştürme belirtimine karşılık gelir. Her dönüştürme belirtimi bir yüzde işaretiyle başlar ve bir dönüştürme belirticiyle biter. Bir format kontrol dizgisinde birçok dönüştürme belirtimi olabilir.

Belirteç	Açıklama
d	İşaretli bir ondalık tamsayı görüntüler.
i	İşaretli bir ondalık tamsayı görüntüler. [Not: i ve d belirleyicileri, scanf ile kullanıldığında farklıdır.]
o	İşaretsiz bir sekizlik tamsayı görüntüler.
u	İşaretsiz bir ondalık tamsayı görüntüler.
x veya X	İşaretsiz bir onaltılık (hexadesimal) tamsayı olarak görüntüler. X, 0-9 rakamları ve A-F büyük harflerinin, x ise 0-9 rakamları ve a-f küçük harflerini ekrana yazar.
h, l veya ll	Sırasıyla kısa, long veya long long bir tamsayının görüntülendiğini belirtmek için herhangi bir tamsayı dönüştürme belirleyicisinden önce yerleştirilir. Bunlara uzunluk değiştiriciler denir.

Şekil 9.2, tamsayı dönüştürme belirleyicilerinin her birini kullanarak bir tamsayı yazdırır. Yalnızca eksi işareti yazdırılır; artı işaretleri normalde gizlenir; Ayrıca -455 değeri, %u (satır 15) ile yazdırıldığında, işaretsiz bir değer olan 4294966841 olarak yorumlanır.

```c
// Fig. 9.2: fig09_02.c
// Using the integer conversion specifiers
#include <stdio.h>

int main(void)
{
   printf("%d\n", 455);
   printf("%i\n", 455); // i same as d in printf
   printf("%d\n", +455); // plus sign does not print
   printf("%d\n", -455); // minus sign prints
   printf("%hd\n", 32000);
   printf("%ld\n", 2000000000L); // L suffix makes literal a long int
   printf("%o\n", 455); // octal
   printf("%u\n", 455);
   printf("%u\n", -455);
   printf("%x\n", 455); // hexadecimal with lowercase letters
   printf("%X\n", 455); // hexadecimal with uppercase letters
}
```

Program çalıştırıldığında çıktısı şöyle olur;

```
455
455
455
-455
32000
2000000000
707
455
4294966841
1c7
1C7
```

Bir kayan nokta değeri, 33.5, 0.0 veya -657.983'teki gibi bir ondalık nokta içerir. Kayan nokta değerleri birkaç biçimden birinde görüntülenir. Şekil 9.3, kayan nokta dönüştürme tanımlayıcılarını açıklar. Dönüşüm belirteçleri e ve E, matematikte kullanılan bilimsel gösterimin bilgisayar eşdeğeri olan üstel gösterimde kayan noktalı değerleri görüntüler. Örneğin 150.4582 değeri bilimsel gösterimde;

1.504582×10^2

Exponansiyel gösterimde ise;

1.504582E+02

Bu notasyon, 1.504582'nin 10 ile çarpılıp ikinci kuvvetinin alındığın gösterir (E+02). E, "üs" anlamına gelir.

Dönüşüm belirtici	Açıklama
e veya E	Üstel gösterimde bir kayan nokta değeri görüntüler.
f veya F	Kayan noktalı değerleri sabit nokta notasyonunda görüntüler.
g veya G	Bir kayan nokta değerini, değerin büyüklüğüne bağlı olarak f kayan nokta biçiminde veya üstel e (veya E) biçiminde görüntüler.
L	Bir "uzun çift – long double" kayan nokta değerinin görüntülenmesi gerektiğini belirtmek için herhangi bir kayan nokta dönüştürme belirleyicisinden önce yerleştirilir.

Şekil 9.3 | Kayan nokta dönüştürme belirleyicileri.

e, E ve f dönüştürme tanımlayıcılarıyla görüntülenen değerler, varsayılan olarak ondalık noktanın sağında altı basamaklı kesinlik gösterir (örn. 1.045927); Dönüştürme belirleyicisi f her zaman ondalık noktanın soluna en az bir basamak yazdırır. Dönüştürme belirteçleri e ve E, üsten önce sırasıyla küçük e ve büyük E'yi yazdırır ve ondalık noktanın soluna tam olarak bir basamak yazdırır.

Dönüştürme belirleyicisi g (veya G), sonunda sıfır olmadan e (E) veya f biçiminde yazdırır (1.234000, değerini 1.234 olarak yazdırılır). Üstel gösterime dönüştürüldükten sonra, değerin üssü 4'ten küçükse veya üs belirtilen hassasiyetten büyük veya ona eşitse değerler e (E) ile yazdırılır. Aksi takdirde, değeri yazdırmak için dönüştürme belirleyicisi f kullanılır. Ondalık noktanın çıktısı için en az bir ondalık basamak gereklidir. Örneğin, 0.0000875, 8750000.0, 8.75 ve 87.50 değerleri, g dönüştürme tanımlayıcısı ile 8.75e-05, 8.75e+06, 8.75 ve 87.5 olarak yazdırılır.

Dönüştürme belirteçleri g ve G için kesinlik, ondalık noktanın solundaki basamak dâhil olmak üzere yazdırılan anlamlı basamakların maksimum sayısını gösterir. 1234567.0 değeri, %g dönüştürme belirleyicisi kullanılarak 1.234567e+06 olarak yazdırılır. g ve G arasındaki fark, değer üstel gösterimde yazdırıldığında e ve E arasındaki farkla aynıdır; küçük g harfi, küçük e harfinin çıkmasına neden olur ve büyük G harfi, büyük E harfinin çıkmasına neden olur.

Şekil 9.4, kayan nokta dönüştürme belirleyicilerinin her birini göstermektedir. %E, %e ve %g dönüştürme belirticileri, değerin çıktıda yuvarlanmasına neden olur ve %f dönüştürme belirticisi bunu yapmaz.

```c
// Fig. 9.4: fig09_04.c
// Using the floating-point conversion specifiers
#include <stdio.h>

int main(void)
{
   printf("%e\n", 1234567.89);
   printf("%e\n", +1234567.89); // plus does not print
   printf("%e\n", -1234567.89); // minus prints
   printf("%E\n", 1234567.89);
   printf("%f\n", 1234567.89); // six digits to right of decimal point
   printf("%g\n", 1234567.89); // prints with lowercase e
   printf("%G\n", 1234567.89); // prints with uppercase E
}
```

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

c ve s dönüştürme belirteçleri, sırasıyla tek tek karakterleri ve dizeleri yazdırmak için kullanılır. Dönüşüm belirteci c, bir “char” bağımsız değişkeni gerektirir. Dönüştürme belirticileri, sonlandırıcı bir boş ('\0') karakterle karşılaşılıncaya kadar karakterlerin yazdırılmasını sağlar. Şekil 9.5'teki program, c ve s dönüştürmeleri ile karakterleri ve dizileri görüntüler.

```c
// Fig. 9.5: fig09_05c
// Using the character and string conversion specifiers
#include <stdio.h>

int main(void)
{
   char character = 'A'; // initialize char
   printf("%c\n", character);

   printf("%s\n", "This is a string");

   char string[] = "This is a string"; // initialize char array
   printf("%s\n", string);

   const char *stringPtr = "This is also a string"; // char pointer
   printf("%s\n", stringPtr);
}
```

```
A
This is a string
This is a string
This is also a string
```

Derleyicilerin çoğu, format kontrol dizesindeki hataları yakalamaz, bu nedenle, bir program çalışma zamanında yanlış sonuçlar üretene kadar genellikle bu tür hataların farkına varmazsınız.

Dönüşüm belirtici	Açıklama
p	Uygulama tanımlı bir şekilde bir işaretçi değeri görüntüler.
%	Yüzde karakterini görüntüler.

Şekil 9.7'deki %p, ptr'nin değerini ve x'in adresini yazdırır; ptr'ye x'in adresi atandığından bu değerler aynıdır. Son printf ifadesi, bir karakter dizisindeki % karakterini yazdırmak için %%'yi kullanır.

```c
// Fig. 9.7: fig09_07.c
// Using the p and % conversion specifiers
#include <stdio.h>

int main(void)
{
   int x = 12345; // initialize int x
   int *ptr = &x; // assign address of x to ptr

   printf("The value of ptr is %p\n", ptr);
   printf("The address of x is %p\n\n", &x);

   printf("Printing a %% in a format control string\n");
}
```

```
The value of ptr is 002EF778
The address of x is 002EF778

Printing a % in a format control string
```

Önceki derslerde gördüğünüz gibi, biçim kontrol dizesine dahil edilen gerçek karakterlerin çıktısı “printf” tarafından verilir. Ancak, biçim kontrol dizesinin kendisini sınırlayan tırnak işareti (") gibi birkaç "sorunlu" karakter vardır. Yeni satır ve sekme gibi çeşitli kontrol karakterleri kaçış dizileriyle temsil edilmelidir. Aşağıdaki tablo, kaçış dizilerini ve neden oldukları eylemleri listeler.

İmleç sırası	Açıklama
\' (tek tırnak)	Tek tırnak (') karakterini çıkarır.
\" (çift tırnak)	Çift tırnak (") karakterini çıkarır.
\? (soru işareti)	Soru işareti (?) karakterinin çıktısını alır.
\\ (ters eğik çizgi)	Ters eğik çizgi (\) karakterini çıkarır.
\a (uyarı veya çağrı)	Sesli (zil) veya görsel bir uyarıya neden olur (tipik olarak, programın çalıştığı pencerenin yanıp sönmesi).
\b (geri al)	İmleci geçerli satırda bir konum geriye taşır.
\f (yeni sayfa veya form)	İmleci bir sonraki mantıksal sayfanın başına taşır.
\n (yeni satır)	İmleci bir sonraki satırın başına taşır.
\r (satır başı)	İmleci geçerli satırın başına taşır.
\t (yatay sekme)	İmleci bir sonraki yatay sekme konumuna taşır.
\v (dikey sekme)	İmleci bir sonraki dikey sekme konumuna taşır.

Giriş biçimlendirmesi “scanf” ile gerçekleştirilebilir. Her "scanf" ifadesi, girilecek verinin biçimini açıklayan bir biçim kontrol dizesi içerir. Biçim kontrol dizesi, dönüştürme belirteçlerinden ve değişmez karakterlerden oluşur.

“scanf” fonksiyonu aşağıdaki biçimde yazılır:

scanf(*format-control-string*, *other-arguments*);

Burada, format-control-string, girdinin biçimlerini tanımlar ve diğer-argümanlar, girdinin depolanacağı yerlere işaret eden işaretçileri gösterir.

Aşağıdaki tablo, tüm veri türlerini girmek için kullanılan dönüştürme tanımlayıcılarını özetlemektedir. Bu bölümün geri kalanında, çeşitli "scanf" dönüştürme belirleyicileri ile okuma verilerini gösteren programlar verilmektedir. d ve i dönüştürme belirteçlerinin “scanf” ile girdi için farklı anlamlara sahip olduğunu unutmayalım.

Dönüşüm belirtici	Açıklama
Tamsayılar	
d	İsteğe bağlı olarak işaretli bir ondalık tam sayı okur. Karşılık gelen bağımsız değişken, bir “int” için bir işaretçidir.
i	İsteğe bağlı olarak işaretlenmiş bir ondalık, sekizlik veya onaltılık tamsayı okur. Karşılık gelen bağımsız değişken, bir int için bir işaretçidir.
o	Sekizlik (octal) bir tamsayı okur. Karşılık gelen argüman, işaretsiz bir int için bir işaretçidir.
u	İşaretsiz bir ondalık tam sayı okur. Karşılık gelen argüman, işaretsiz bir int için bir işaretçidir.
x veya X	Onaltılık (Hexadesimal) bir tam sayı okur. Karşılık gelen argüman, işaretsiz bir int için bir işaretçidir.
h, l ve ll	Sırasıyla kısa, uzun veya long long bir tamsayı girileceğini belirtmek için tamsayı dönüştürme belirticilerinden herhangi birinin önüne yerleştirilir.

Kayan Noktalı Sayılar	
e, E, f, g veya G	Bir kayan nokta değeri okur. Karşılık gelen bağımsız değişken, bir kayan nokta değişkenine yönelik bir işaretçidir.
l veya L	Double veya long double değerinin girileceğini belirtmek için herhangi bir kayan nokta dönüştürme belirticisinden önce yerleştirir. Karşılık gelen bağımsız değişken, bir double veya long double değişkene işaretçidir.
Karakterler ve diziler	
c	Bir karakter okur. Karşılık gelen bağımsız değişken, bir karaktere yönelik bir işaretçidir; null ('\0') eklenmez.
s	Bir dizi okur. Karşılık gelen bağımsız değişken, dizeyi ve otomatik olarak eklenen bir sonlandırıcı null ('\0') karakterini tutacak kadar büyük olan “char” türünde bir dizinin işaretçisidir.
scan set	
[scan characters]	Bir dizide saklanan bir dizi karakter için bir dizi tarar.
Çeşitli(Miscellaneous)	
p	Bir printf deyiminde %p ile bir adres çıktısı alındığında üretilen aynı biçimdeki bir adresi okur.
n	Bu scanf çağrısında o ana kadar girilen karakter sayısını saklar. Buna karşılık gelen bağımsız değişken bir int için bir işaretçi olmalıdır.
%	Girişte bir yüzde işaretini (%) atlar.

Şekil 9.18, tamsayıları çeşitli tamsayı dönüştürme belirleyicileri ile okur ve tamsayıları ondalık sayılar olarak görüntüler. Dönüşüm belirteci %i, ondalık, sekizli ve onaltılık tamsayılar girebilir.

```c
// Fig. 9.18: fig09_18.c
// Reading input with integer conversion specifiers
#include <stdio.h>

int main(void)
{
   int a;
   int b;
   int c;
   int d;
   int e;
   int f;
   int g;

   puts("Enter seven integers: ");
   scanf("%d%i%i%i%o%u%x", &a, &b, &c, &d, &e, &f, &g);

   puts("\nThe input displayed as decimal integers is:");

   printf("%d %d %d %d %d %d %d\n", a, b, c, d, e, f, g);
}
```

```
Enter seven integers:
-70 -70 070 0x70 70 70 70

The input displayed as decimal integers is:
-70 -70 56 112 56 70 112
```

Kayan noktalı sayıları girerken, e, E, f, g veya G kayan nokta dönüştürme belirteçlerinden herhangi biri kullanılabilir. Şekil 9.19, üç kayan noktalı sayıyı okur, her biri üç tür kayan dönüştürme belirticisine sahiptir ve üç sayının tümünü f dönüştürme belirticisine sahiptir.

```c
// Fig. 9.19: fig09_19.c
// Reading input with floating-point conversion specifiers
#include <stdio.h>

// function main begins program execution
int main(void)
{
   double a;
   double b;
   double c;

   puts("Enter three floating-point numbers:");
   scanf("%le%lf%lg", &a, &b, &c);

   printf("\nHere are the numbers entered in plain:");
   puts("floating-point notation:\n");
   printf("%f\n%f\n%f\n", a, b, c);
}
```

```
Enter three floating-point numbers:
1.27987 1.27987e+03 3.38476e-06

Here are the numbers entered in plain floating-point notation:
1.279870
1279.870000
0.000003
```

Karakterler ve dizeler, sırasıyla c ve s dönüştürme belirteçleri kullanılarak girilir. Şekil 9.20, kullanıcıdan bir dizi girmesini ister. Program, dizenin ilk karakterini %c ile girer ve bunu x karakter değişkeninde saklar, ardından dizenin geri kalanını %s ile girer ve bunu y karakter dizisinde saklar.

```c
// Fig. 9.20: fig09_20.c
// Reading characters and strings
#include <stdio.h>

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

   printf("%s", "Enter a string: ");
   scanf("%c%8s", &x, y);

   puts("The input was:\n");
   printf("the character \"%c\" and the string \"%s\"\n", x, y);
}
```

```
Enter a string: Sunday
The input was:
the character "S" and the string "unday"
```

Bir tarama seti kullanılarak bir dizi karakter girilebilir. Bir tarama seti, köşeli parantezler [ ] içine alınmış ve format kontrol dizesinde önünde bir yüzde işareti bulunan bir karakter setidir. Bir tarama seti, giriş akışındaki karakterleri tarar ve yalnızca tarama setinde bulunan karakterlerle eşleşen karakterleri arar. Bir karakter her eşleştiğinde, tarama setinin karşılık gelen bağımsız değişkeninde (bir karakter dizisi işaretçisi) depolanır. Tarama seti, tarama setinde yer almayan bir karakterle karşılaşıldığında karakter

girişini durdurur. Giriş akışındaki ilk karakter tarama setindeki bir karakterle eşleşmiyorsa dizi değiştirilmez. Şekil 9.21, giriş akışını ünlüler için taramak için [aeiou] tarama setini kullanır. Girişin ilk yedi harfinin okunduğuna dikkat edin. Sekizinci harf (h) tarama setinde değildir ve bu nedenle tarama sonlandırılır.

```c
// Fig. 9.21: fig09_21.c
// Using a scan set
#include <stdio.h>

// function main begins program execution
int main(void)
{
   char z[9]; // define array z

   printf("%s", "Enter string: ");
   scanf("%8[aeiou]", z); // search for set of characters

   printf("The input was \"%s\"\n", z);
}
```

```
Enter string: ooeeooahah
The input was "ooeeooa"
```