A printf függvény

A printf függvény formázott adatkivitelt valósít meg. Hívásának formája:

```
printf(formátumstring, paraméterlista)
```

A formátumstring, amely rendszerint stringkonstans, kétféle karaktert tartalmazhat:

- Az első csoport karakterei normál karaktersorozatot alkotnak a formátumstringen belül, ezek minden átalakítás nélkül kerülnek kiírásra.
- A másik csoport karakterei speciális konverziót írnak elő, amelyek meghatározzák a paraméterlistában szereplő paraméterek értelmezési módját és a megjelenés formátumát. Ezek a konverziós előírások százalékjellel kezdődnek és valamilyen konverziós karakterrel zárulnak.

A konverziós előírás teljes formája az alábbi (a [és] közötti részek opcionálisak):

```
%[jelzők][min mezőszélesség][.pontosság][méretmódosító]konverziós karakter
```

Konverziós karakterek

Konverziós karakter	A paraméter típusa	Nyomtatási kép
`c`	*int*	*unsigned char* típusra való konvertálás eredménynek megfelelő karakter. pl.: a
`d, i`	*int*	Előjeles decimális egész. pl.: 1123, -7235
`u`	*unsigned int*	Előjel nélküli decimális egész. pl.: 7235
`f`	*double*	*[-]dddd.dddd*, ahol a tizedesjegyek száma az előírt pontosságtól függ. A pontosság alapértelmezésben 6, 0 pontosságot megadva a tizedespont sem íródik ki. pl.: 23.645321, -78.500000
`e, E`	*double*	*[-]d.ddde±ddd* vagy *[-]d.ddddE±ddd*, ahol a tizedesjegyek száma az előírt pontosságtól függ. pl.: 3.9265e+2, 3.9265E-2
`g, G`	*double*	`%e` vagy `%E` alakú kiírást használ, ha a kitevő kisebb mint -4, vagy nagyobb-egyenlő, mint a pontosság, különben a `%f` alakú kiírást használja. A tizedes pont és az utána következő értéktelen nullák nem íródnak ki.
`o`	*unsigned int*	Előjel nélküli oktális egész, a számot megelőző 0 számjegy nélkül. pl.: 610
`x, X`	*unsigned int*	Előjel nélküli hexadecimális egész a 0x előtag nélkül. pl.: 7fa, 7FA
`s`	*char *, char[]*	A string karaktereit írja ki a lezáró 0-ig. pl.: progalap
`p`	*void **	A pointer értéke a géptől függő kiírási formában. pl.: b8000000
`n`	*int **	A *printf* függvény aktuális hívásakor kiírt karakterek száma beíródik a mutató által kijelölt változóba.
`%`	-	Nincs konverzió, két % egymás utáni eredménye, hogy egy % íródik ki.

Megjegyzés

Furcsa és félrevezető lehet, hogy „A paraméter típusa” oszlopban sehol nem találunk *char* vagy *float* típust megadva a

`printf` várt paraméter típusaként, mégis előszeretettel íratunk ki *char* típusú változókat `%c`-vel, vagy éppen *float* típusúakat `%f`-fel. A `printf` valóban olyan típusú paramétert vár, ami az oszlopban szerepel, egy példán keresztül vegyük szemügyre, mi is történik:

```
char c = 'A';
printf(”c == %c”, c);
```

Látható, hogy a `printf`-ben a `%c` konverziós előírás szerepel, aminek megfelelően egy *int* típusú paramétert vár, mégis a `c` *char* típusú változót kapja. Ekkor az történik, hogy ezt a *char* típusú változó *int*-re konvertálódik a C implicit típuskonverzióknak megfelelően, és így adódik át `printf` függvénynek, ami aztán beül visszakonvertálja *unsigned char*-ra, és egy ennek megfelelő kódú karaktert jelenít meg.
Hasonlóan történik mindenez a `%f` és egy *float* típusú változó megadása esetén. A *float* változót egy a `printf` által várt *double*-ra konvertálja a rendszer (ami szintén megtehető veszteség nélkül) és a `printf` így fogja feldolgozni azt.

Méretmódosítók

`h`
A `d`, `i` konverziós karakterekkel együtt megadva a *short int* típust jelöli.
A `o`, `u`, `x` és `X` konverziós karakterekkel pedig az *unsigned short int* típust.

`l`
A `d`, `i` konverziós karakterekkel együtt megadva a *long int* típust jelöli.
A `o`, `u`, `x` és `X` konverziós karakterekkel pedig az *unsigned long int* típust.

`ll`
A `d`, `i` konverziós karakterekkel együtt megadva a *long long int* típust jelöli.
A `o`, `u`, `x` és `X` konverziós karakterekkel pedig az *unsigned long long int* típust.

`L`
Az `e`, `E`, `g`, `G` és `f` konverziós betűkkel együtt használva a *long double* típusnak felel meg.

Példa:

```
long long int a = 32;
long double b = 23.32;
printf(”%lld, %Lf”, a, b);
```

Bővebben:

http://www.cplusplus.com/reference/clibrary/cstdio/printf/

Lásd még:

http://www.cplusplus.com/reference/clibrary/cstdio/fprintf/
http://www.cplusplus.com/reference/clibrary/cstdio/sprintf/

A scanf függvény

A `scanf` függvény formázott adatbevitelt valósít meg. A szabványos beviteli eszközről olvassa be a karaktereket, majd a formátumstring szerint értelmezi és konvertálja azokat. A konverzió eredményét a soron következő paraméter által kijelölt memóriaterületre tölti. (A paraméterek tehát mindig pointerek.) Hívásának általános formája:

```
scanf(formátumstring, paraméterlista)
```

A formátumstring felépítése, hasonlít a `printf` függvénynél használt formátumra, azonban több ponton is eltér attól. Az alábbi elemekből épül fel:

- Tagoló karakterek, melyeket a `scanf` figyelmen kívül hagy. Pl.: vessző, szóköz (vagy egyéb whitespace karakter)
- Konverziós előírások, amelyek általános formátuma az alábbi:

```
%[*][szélesség][méretmódosító]konverziós karakter
```

Konverziós karakterek

Konverziós karakter	**A paraméter típusa**	**Az input adat formátuma**
`d`	*int* *	Decimális egész. pl.: -3121
`i`	*int* *	Egész szám decimális, oktális, vagy akár hexadecimális formában. Oktális szám esetén 0-val, hexadecimális szám esetén pedig a 0x, vagy 0X karakterekkel kell kezdődnie. pl.: 12, 012, 0x12
`u`	*unsigned* *	Előjel nélküli decimális egész. pl.: 3121
`o`	*int* *	Oktális egész a 0 előtag nélkül. pl.: 5677
`x`	*int* *	Hexadecimális egész akár a 0x, vagy 0X előtaggal, akár anélkül. pl.: 12AD, 0x12AD
`c`	*char* *	Karakter. A soron következő karakter kerül a kijelölt memóriaterületre. A tagoló karakterek is beolvasásra kerülnek. A következő nem tagoló karakter beolvasásához a %1s konverziós előírást szokás használni. pl.: a
`s`	*char *, char[]*	String olvasása tagoló karakterig. Az hozzá tartozó argumentum egy olyan karaktertömbre kell, hogy mutasson, amely elegendően nagy a beolvasandó karakterek és a string végét jelző 0 tárolására. pl.: progalap
`f, e, g`	*float* *	Ezen konverziós karakterek szolgálnak lebegőpontos számok beolvasására. Az előjel, a tizedespont és az exponens megadása opcionális. pl.: 23.23, 34.4E+10
`p`	*void* **	A megfelelő formátumú pointer értékét (azaz egy memóriacímet) olvas be.
`n`	*int* *	Nem vár inputot. Az aktuális konverziós előírásig beolvasott karakterek számát adja vissza.
`[`*karakterek*`]`	*char *, char[]*	Csak a szögletes zárójelben szereplő karakterek kerülnek be a stringbe. Például %[INin] csak a kis és nagy I, N betűket tartalmazó stringet olvassa be.
`[^`*karakterek*`]`	*char *, char[]*	Csak a szögletes zárójelben nem szereplő karakterek kerülnek be a stringbe. Például %[^0-9] csak a megadott karakterintervallumon kívülre eső karaktereket olvassa, azaz minden számjegyet nem tartalmazó stringet elfogad.

Méretmódosítók

`h`
A `d`, `i` konverziós karakterekkel együtt megadva a *short int* * típusú változót vár a paraméterlistában.
A `o`, `u`, `x` és `X` konverziós karakterekkel pedig az *unsigned short int* * típust vár.

`l`
A `d`, `i` konverziós karakterekkel együtt megadva a *long int* * típust vár.
A `o`, `u`, `x` és `X` konverziós karakterekkel pedig az *unsigned long int* * típust.

`ll`
A `d`, `i` konverziós karakterekkel együtt megadva a *long long int* * típust vár a `scanf`.
A `o`, `u`, `x` és `X` konverziós karakterekkel pedig az *unsigned long long int* * típust.

`L`

`Az e, E, g, G és f` konverziós betűkkel együtt használva a *long double* * típusú változó megadása szükséges.

*****

A csillag karakter megadásával a beolvasott adatot a scanf eldobja.

Szélesség

Decimális egész szám, amely megmondja, hogy a scanf maximálisan hány darab karaktert olvasson be.

Bővebben:

http://www.cplusplus.com/reference/clibrary/cstdio/scanf/

Lásd még:

http://www.cplusplus.com/reference/clibrary/cstdio/fscanf/

http://www.cplusplus.com/reference/clibrary/cstdio/sscanf/