

FUNKCJA *printf()*

FUNKCJA *printf()*

Prototyp funkcji `printf()` jest zadeklarowany w pliku nagłówkowym `stdio.h`. Ma on następującą postać:

```
int printf( const char *format, ... );
```

Bardziej poglądowo prototyp ten można przedstawić następująco:

```
printf("format" [,arg1 [,arg2 [, ...]]] );
```

Funkcja `printf()` posiada co najmniej 1 argument `format` będący łańcuchem znaków opisującym sposób wyprowadzania napisu. Może przyjmować ciąg argumentów `arg1`, `arg2`, itd, które kojarzy kolejno ze specyfikatorami formatu rozpoczynającymi znakiem `%` zawartymi w pierwszym argumentcie. Uformowany napis jest wyprowadzany na standardowe wyjście `stdout` (najczęściej monitor terminala). Należy zwracać uwagę aby liczba argumentów `arg1`, `arg2`, itd. była równa liczbie specyfikatorów formatu zawartych w tekście formatującym.

Przykład:

Jeśli założyć, że zmienne `x1` i `x2` typu `int` mają nadane wartości odpowiednio 2 i 4 to użycie funkcji `printf()` w postaci:

```
printf("Ala ma %d psy.\nOla ma %d rybki.\n", x1, x2);
```

spowoduje wypisanie następującego tekstu:

```
Ala ma 2 psy.  
Ola ma 4 rybki.
```

Tekst formatujący funkcji *printf()*

Format jest pierwszym argumentem funkcji `printf()`. Jest on najczęściej łańcuchem znaków zapisanym pomiędzy znakami cudzysłowów (" "), który jest wyprowadzany bez zmian na terminal (konsolę) z wyjątkiem sekwencji rozpoczynających się znakami `%` (procent) i `\` (backslash).

Znak procentu (`%`) oznacza rozpoczęcie specyfikatora formatu pozwalającego na wyprowadzenie w jego miejscu wartości kolejnego argumentu funkcji `printf()`. Wyjątek stanowi sekwencja `%%`, która umożliwia wyprowadzenie znaku procentu (`%`).

Znak odwróconego ukośnika (`\`) umożliwia wstawienie znaków specjalnych (nowa linia, tabulacja itp).

Specyfikator formatu ma postać:

```
%[flagi][szer][.prec][modyf]typ
```

Pole **typ** określa typ argumentu wejściowego i sposób jego prezentacji:

Pole „typ”	Typ argumentu wejściowego	Sposób prezentacji
d	int	Dziesiętny ze znakiem
i	int	Dziesiętny ze znakiem
o	int	Ósemkowy bez znaku
u	int	Dziesiętny bez znaku
x	int	Szesnastkowy bez znaku (ze znakami a,b,c,d,e,f)
X	int	Szesnastkowy bez znaku (ze znakami A,B,C,D,E,F)
f	float	Zmiennoprzecinkowy ze znakiem w postaci <i>ddd.dd</i>
e	float	Zmiennoprzecinkowy w postaci wykładniczej <i>dd.deddd</i>
E	float	Zmiennoprzecinkowy w postaci wykładniczej <i>dd.dEddd</i>
g	float	Tak jak format <i>f</i> lub <i>e</i> zależnie od wartości i precyzji
G	float	Tak jak format <i>f</i> lub <i>E</i> zależnie od wartości i precyzji
c	char	Pojedynczy znak
s	char*	Łańcuch znaków (napis)
p	typ* (pointer)	Wartość wskaźnika

FUNKCJA *printf()*

Pole **modyf** jest opcjonalne i określa dokładnie wielkość argumentu wejściowego.

Pole „ modyf ”	Znaczenie
h	Argument wejściowy jest interpretowany jako short int dla pola typ równego: d, i, o, u, s, X, x
l	Argument wejściowy jest interpretowany jako - long int dla pola typ równego: d, i, o, u, x, X - double dla pola typ równego: e, E, f, g, G
L	Argument wejściowy jest interpretowany jako long double dla pola typ równego: e, E, f, g, G
F	Argument wejściowy jest interpretowany jako wskaźnik daleki (far pointer)
N	Argument wejściowy jest interpretowany jako wskaźnik bliski (near pointer)

Pole **szer** określa minimalną szerokość pola w znakach, w którym ma być przedstawiony dany argument (liczba, znak itp). Jeśli w polu **szer** jest wpisany znak * (zamiast liczby całkowitej) to szerokość pola jest określona przez wartość kolejnego argumentu wejściowego funkcji `printf()`.

Pole **.prec** określa dokładność (ilość cyfr znaczących) z jaką ma być przedstawiona liczba zmiennoprzecinkowa. Jeśli w polu **.prec** jest wpisany znak * (zamiast liczby całkowitej), to dokładność jest określona przez wartość kolejnego argumentu wejściowego funkcji `printf()`.

Pole **flagi** dodatkowo określa inne cechy prezentacji argumentu takie jak sposób justowania, wypisywanie zer nieznaczących, pisanie znaku + przed liczbą itp.

Pole „ flagi ”	Znaczenie
-	Justowanie lewostronne (brak tego znaku powoduje justowanie prawostronne)
+	Liczby dodatnie są poprzedzone znakiem +
spacja	Liczby dodatnie są poprzedzone znakiem spacji
0	Wpisywane są zera nieznaczące
#	Alternatywny sposób formatowania. Dla pola „typ” równego: - c, d, i, u, s – nie dają żadnego efektu - o – liczba w postaci ósemkowej jest poprzedzona znakiem 0 (zerem) - x, X – liczba w postaci szesnastkowej jest poprzedzona znakami 0x albo 0X - f, e, E – kropka dziesiętna jest zawsze wyświetlana, nawet gdy część ułamkowa nie zawiera cyfr znaczących - g, G – podobnie jak dla e lub E oraz dodatkowo wyświetlane są nieznaczące zera wiodące

Znaki specjalne wyprowadza się za pomocą sekwencji zaczynającej się znakiem odwróconego ukośnika (\):

Sekwencja specjalna	Znaczenie
\n	Nowa linia (LF lub CR LF)
\t	Znak tabulacji poziomej (HT)
\v	Znak tabulacji pionowej (VT)
\r	Znak powrotu karetki (CR)
\f	Znak nowej strony (FF)
\a	Sygnal dźwiękowy (BEL)
\"	Znak ” (cudzysłów)
\'	Znak ' (apostrof)
\\	Znak \ (odwrócony ukośnik)
\0	Znak o kodzie 0 (NUL)
\ooo	Znak o kodzie <i>ooo</i> (ósemkowo)
\xhh	Znak o kodzie <i>hh</i> (szesnastkowo)
\	Kontynuacja tekstu w następnym wierszu