Funkcje `printf()` i `scanf()` i operatory

Elwira Wachowicz

elwira@ifd.uni.wroc.pl

4 kwietnia 2013

Łańcuch sterujący: printf()

Specyfikator	Dane wyjściowe
%c	Pojedynczy znak
%d	Dziesiętna liczba całkowita ze znakiem
%e	Liczba zmiennoprzecinkowa w notacji z literą e
%E	Liczba zmiennoprzecinkowa w notacji z literą E
%f	Liczba zmiennoprzecinkowa w zapisie dziesiętnym
%g	Równoważny %e, gdy wykładnik mniejszy niż -4 lub większy/równy dokładności, inaczej %f
%E	Jak wyżej, tyle że z E
%o	Ósemkowa liczba całkowita bez znaku
%p	wskaźnik
%s	Łańcuch znakowy
%u	Dziesiętna liczba całkowita bez znaku
%x	Szesnastkowa liczba całkowita, cyfry szesnastkowe małą literą (0f)
%X	Szesnastkowa liczba całkowita, cyfry szesnastkowe dużą literą (0F)
%%	Wyświetla znak procentu

printf(): znaczniki i modyfikatory

Modyfikator	**Znaczenie**
`liczba`	Minimalna szerokość pola. Jeśli wyświetlana wartość nie zmieści się, pole zostanie powiększone.
`.liczba`	Dokładność. Dla zmiennoprzecinkowych: liczba cyfr po przecinku. Dla całkowitych: minimalna liczba cyfr (uzupełniane przez 0). Dla łańcuchów: maksymalna liczba znaków. `.` jest równoważne `.0`
Znacznik	**Znaczenie**
`-`	Wyrównuje do lewej krawędzi pola (`%-20s`).
`+`	Wyświetla ze znakiem + lub - (`%+6.2f`).
`odstęp`	Wyświetla z odstępem na początku (`% 6.2f`).
`0`	Wypełnia pola początkowe zerami zamiast odstępami. Ignorowany, jeśli występuje z `-`.

Przykład: liczby całkowite

```
 /* szerok.c -- szerokosc pola */
#include <stdio.h>
#define STRONY 732

int main(void)
{
    printf("*%d*\n", STRONY);
    printf("*%2d*\n", STRONY);
    printf("*%10d*\n", STRONY);
    printf("*%-10d*\n", STRONY);
    return 0;
}
```

Otrzymamy:

```
*732*
*732*
*␣␣␣␣␣␣␣732*
*732␣␣␣␣␣␣␣*
```

Przykład: liczby zmiennoprzecinkowe

```
/* float.c -- modyfikatory
 liczb zmiennoprzecinkowych */
#include <stdio.h>
#define CZYNSZ 2345.67

int main(void)
{
    printf("*%f*\n", CZYNSZ);
    printf("*%e*\n", CZYNSZ);
    printf("*%4.2f*\n", CZYNSZ);
    printf("*%3.1f*\n", CZYNSZ);
    printf("*%10.3f*\n", CZYNSZ);
    printf("*%10.3e*\n", CZYNSZ);
    printf("*%+4.2f*\n", CZYNSZ);
    printf("*%010.2f*\n", CZYNSZ);
    return 0;
}
```

Otrzymujemy:

```
*2345.670000*
*2.345670e+03*
*2345.67*
*2345.7*
*␣␣2345.670*
*␣2.346e+03*
*+2345.67*
*0002345.67*
```

Przykład: łańcuchy znaków

```
/* lancuchy.c -- formatowanie lancuchow  */
#include <stdio.h>
#define NOTATKA "Doskonala gra aktorow!"

int main(void)
{
    printf("*%2s*\n", NOTATKA);
    printf("*%25s*\n", NOTATKA);
    printf("*%25.5s*\n", NOTATKA);
    printf("*%-25.5s*\n", NOTATKA);
    return 0;
}
```

Otrzymujemy:

```
*Doskonala␣gra␣aktorow!*
*␣␣␣Doskonala␣gra␣aktorow!*
*␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣Dosko*
*Dosko␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣*
```

Wartość zwracana przez printf()

```
/* wartzwr.c -- okreslenie wartosci zwracanej przez printf()  */
#include <stdio.h>

int main(void)
{
    int n = 100;
    int wz;

    wz = printf("Woda wrze w temperaturze %d C.\n", n);
    printf("Funkcja printf() wyswietlila %d znaki.\n", wz);
    return 0;
}
```

Otrzymujemy:

```
Woda wrze w temperaturze 100 C.
Funkcja printf() wyswietlila 32 znaki.
```

Wyświetlanie długich łańcuchów

Nie wolno podzielić na kilka wierszy łańcucha zawartego w cudzysłowie!

```
/* dluglanc.c -- wyswietlanie dlugich lancuchow  */
#include <stdio.h>

int main(void)
{
    printf("Oto jeden ze sposobow wyswietlenia ");
    printf("dlugiego lancucha.\n");
    printf("Oto inny sposob wyswietlenia \
dlugiego lancucha.\n");
    printf("Oto najnowszy sposob wyswietlenia "
             "dlugiego lancucha.\n");
    return 0;
}
```

```
Oto jeden ze sposobow wyswietlenia dlugiego lancucha.
Oto inny sposob wyswietlenia dlugiego lancucha.
Oto najnowszy sposob wyswietlenia dlugiego lancucha.
```

Funkcja `scanf()`

Funkcja `scanf()` przetwarza tekst wejściowy na wartości różnych typów: liczby całkowite, zmiennoprzecinkowe, znaki i łańcuchy znakowe.

- Jeśli używasz `scanf()` do wczytania wartości do zmiennej należącej do któregoś z typów podstawowych, przed nazwą zmiennej dodaj `&`.
- Jeśli używasz `scanf()` do wczytania łańcucha do tablicy znaków, nie dodawaj przedrostka `&`.

Przykład:

```
/* wejscie.c -- kiedy uzywac  */
#include <stdio.h>

int main(void)
{
    int wiek;               /* zmienna  */
    float majatek;          /* zmienna  */
    char zwierzatko[30];    /* lancuch  */

    printf("Podaj swoj wiek, majatek i ulubione zwierzatko.\n");
    scanf("%d %f", &wiek, &majatek); /* tu uzywamy &                    */
    scanf("%s", zwierzatko);         /* przy tablicy znakowej nie ma & */
    printf("%d, %.0f zl, %s.\n", wiek, majatek, zwierzatko);
    return 0;
}
```

```
Podaj swoj wiek, majatek i ulubione zwierzatko.
12
144.50 jez
12, 144 zl, jez.
```

Specyfikatory i modyfikatory

Soecyfikator	Znaczenie
`%c`	Dana wejściowa ma być znakiem.
`%d`	Dana wejściowa ma być liczbą całkowitą ze znakiem.
`%f`, `%e`, `%g`	Dana wejściowa mam być liczbą zmiennoprzecinkową.
`%F`, `%G`	Dana wejściowa mam być liczbą zmiennoprzecinkową.
`%i`	Dana wejściowa ma być liczbą całkowitą ze znakiem.
`%o`	Dana wejściowa ma być liczbą ósemkową ze znakiem.
`%p`	Dana wejściowa ma być wskaźnikiem (adres).
`%u`	Dana wejściowa ma być liczbą całkowitą bez znaku.
`%x`, `%X`	Dana wejściowa ma być liczbą szesnastkową ze znakiem.
`%s`	Dana wejściowa ma być łańcuchem znaków: zaczyna się pierwszym znakiem drukowanym, kończy ostatnim.
`liczba`	Maksymalna szerokość pola. Odczytywanie kończy się po osiągnięciu maksymalnej szerokości pola lub przy wystąpieniu znaku niedrukowanego (`%10s`).

Zwykłe znaki w łańcuchu sterującym

Funkcja `scanf()` pozwala umieszczać zwykłe znaki w łańcuchu sterującym. Muszą one znaleźć się w łańcuchu wejściowym.

```
scanf("%d, %d", &n, &m);
```

Dane należy podać jako:

```
88,121
```

lub

```
88,  121
```

lub

```
88,
121
```

Podsumowanie: `printf()` i `scanf()`

- Funkcje `printf()` i `scanf()` zapewniają obsługę wyjścia i wejścia.
- Wykorzystują łańcuch sterujący, który zawiera specyfikatory konwersji określające liczbę i typy wartości, które należy wyświetlić lub odczytać.
- Specyfikatory mogą służyć do formatowania danych wyjściowych: szerokości pól, liczby miejsc po przecinku oraz umiejscowienia wartości w ramach pola.

Wstęp do pętli

```
/* buty1.c -- przelicza rozmiar buta na cm */
#include <stdio.h>
#define  KOREKTA -1
#define MNOZNIK 0.6666667

int main(void)
{
    double but, stopa;

    but = 42.0;
    stopa = MNOZNIK * but + KOREKTA;
    printf("Rozmiar buta     Długość stopy\n");
    printf("%8.1f %15.2f cm\n", but, stopa);
    return 0;
}
```

Wstęp do pętli c.d.

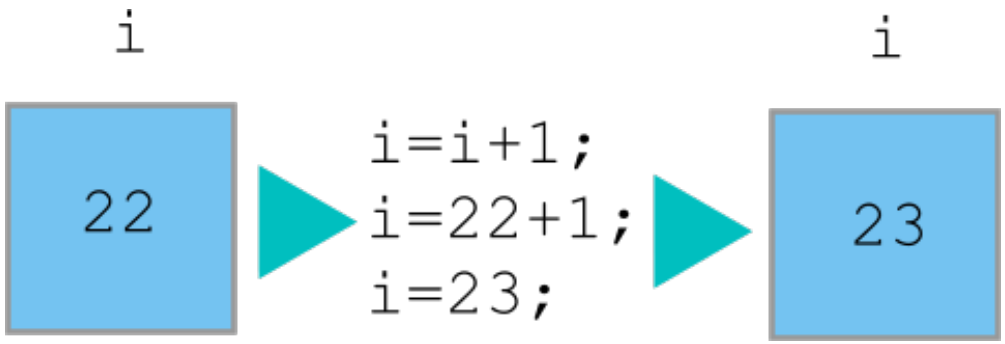
```
/* buty2.c -- oblicza dlugosc stopy dla kilku numerow buta  */
#include <stdio.h>
#define  KOREKTA -1
#define MNOZNIK 0.6666667

int main(void)
{
    double but, stopa;

    printf("Rozmiar buta     Długość stopy\n");
    but = 24.0;
    while ( but < 45)
    {
           stopa = MNOZNIK * but + KOREKTA;
           printf("%8.1f %15.2f cm\n", but, stopa);
           but = but + 1;
    }
    printf("Jeśli but pasuje noś go.\n");
    return 0;
}
```

Podstawowe operatory

- Operator przypisania: =
  zmienna = wartosc

```
bmw = 2010;
```

```
i = i + 1;
```



Podstawowe operatory c.d.

- Operator dodawania: +

```
printf("%d", 4 + 20);
dochod = pensja + lapowki;
```

- Operator odejmowania: -

```
dochod = 224.0 - 24.0;
```

- Operatory jednoargumentowe: + i -

```
bolek = -12;
lolek = -bolek;
```

- Operator mnożenia: *

```
cm = 2.54 * cale;
```

```c
/* kwadraty.c -- wyświetla tabelę pierwszych 20 kwadratów */
#include <stdio.h>

int main(void)
{
    int num = 1;

    while (num < 21)
    {
       printf("%10d %10d\n", num, num * num);
       num = num + 1;
    }
    return 0;
}
```

Podstawowe operatory c.d.

- Operator dzielenia: /

```
cztery = 12.0/3.0;
```

Inaczej działa dzielenie liczb zmiennoprzecinkowych, a inaczej całkowitych. Przy dzieleniu liczb całkowitych część ułamkowa jest odrzucana.

```
/* dziel.c -- rozne rodzaje dzielenia  */
#include <stdio.h>

int main(void)
{
    printf("dzielenie calkowite:   5/4   daje  %d \n",5/4);
    printf("dzielenie calkowite:   6/3   daje  %d \n", 6/3);
    printf("dzielenie calkowite:   7/4   daje  %d \n", 7/4);
    printf("dzielenie zmiennoprz.:   7./4.   daje  %1.2f \n", 7./4.);
    printf("dzielenie mieszane:   7./4   daje  %1.2f \n", 7./4);
    return 0;
}
```

Priorytet operatorów

Mamy działanie:

```
maslo = 25.0 + 60.0 * n / SKALA;
```

Operatory według malejącego priorytetu:

Operatory	Kierunek wiązania
()	od lewej do prawej
+ - (jednoargumentowe)	od prawej do lewej
* /	od lewej do prawej
+ - (dwuargumentowe)	od lewej do prawej
=	od prawej do lewej

Priorytet i kolejność obliczeń

```
/* zasady.c -- test kolejności działań  */
#include <stdio.h>

int main(void)
{
    int rekord, wynik;
    rekord = wynik = -(2 + 5) * 6 + (4 + 3 * (2 + 3));
    printf("rekord = %d \n", rekord);
    return 0;
}
```

Operator modulo: %

Operator modulo jest wykorzystywany wyłącznie w arytmetyce całkowitoliczbowej. Zwraca resztę z dzielenia liczby po lewej stronie przez liczbę po prawej stronie.

```
13 % 5
```

Operator modulo: %

```
/* min_sek.c -- przelicza sekundy na minuty i sekundy  */
#include <stdio.h>
#define SEK_W_MIN 60

int main(void)
{
    int sek, min, reszta;

    printf("Przelicz sekundy na minuty  sekundy!\n");
    printf("Podaj liczbe sekund, ktora chcesz przeliczyc\n");
    scanf("%d", &sek);          /* pobranie liczby sekund                      */
    min = sek /SEK_W_MIN;       /* liczba sekund z obciętą cześcią ułamkową */
    reszta = sek % SEK_W_MIN;   /* pozostała liczba sekund                     */
    printf("%d sekund to %d minut, %d sekund.\n", sek, min, reszta);
    return 0;
}
```

Przykładowe wykonanie programu:

```
Podaj liczbe sekund, ktora chcesz przeliczyc
333
333 sekund to 5 minut, 33 sekund.
```