

# Функції `printf()` і `scanf()`



*Лекція №7*

*Дисципліна «Програмування»*



# Функція printf()

Формат використання функції **printf()**:

```
printf(керувальний_рядок, елемент_1, елемент_2, ...);
```

**елемент\_1, елемент\_2** і т. д. – це елементи, які треба вивести.

Ними можуть бути **змінні, константи** або навіть **вирази**, які обчислюються до того, як значення буде виведено.

**керувальний\_рядок** – символний рядок, який описує спосіб виводу елементів.

Він повинен містити специфікатор перетворення для кожного елемента, що підлягає виводу.



# Специфікатори перетворення

Специфікатор перетворення	Опис виводу
%a	Число з рухомою комою, шістнадцяткові цифри та r-запис (C99/C11)
%A	Число з рухомою комою, шістнадцяткові цифри та R-запис (C99/C11)
%c	Одиночний символ
%d	Десяткове ціле число зі знаком
%e	Число з рухомою комою, експоненціальне представлення
%E	Число з рухомою комою, експоненціальне представлення
%f	Число з рухомою комою, десяткове представлення
%g	В залежності від значення використовує %f або %e. Специфікатор %e використовується, якщо показник степені менше -4 або більше чи дорівнює вказаній точності



# Специфікатори перетворення

Специфікатор перетворення	Опис виводу
<b>%G</b>	В залежності від значення використовує <b>%f</b> або <b>%E</b> . Специфікатор <b>%E</b> використовується, якщо показник степені менше <b>-4</b> або більше чи дорівнює вказаній точності
<b>%i</b>	Десяткове ціле число зі знаком (те ж саме, що й <b>%d</b> )
<b>%o</b>	Вісімкове ціле число без знака
<b>%p</b>	Вказівник
<b>%s</b>	Символьний рядок
<b>%u</b>	Десяткове ціле число без знака
<b>%x</b>	Шістнадцяткове ціле число без знака. Використовуються шістнадцяткові цифри <b>0f</b>
<b>%X</b>	Шістнадцяткове ціле число без знака. Використовуються шістнадцяткові цифри <b>0F</b>
<b>%%</b>	Знак процента



# Використання функції printf()

```
#include <stdio.h>
#include <windows.h>

#define PI 3.141593
```

```
int main(void)
{
    int    number = 7;
    float  pies   = 12.75;
    int    cost    = 7800;

    SetConsoleOutputCP(1251);

    printf("Значення пі дорівнює %.f.\n", PI);
    printf("%d%% учасників змагань з'їли %.f пиріжків з вишнями.\n",
           number, pies);
    printf("До побачення! Ваші здібності занадто дорого обходяться.\n");
    printf("Ми через вас втратили %c%d\n", '$', 2 * cost);
    return 0;
}
```

```
D:\KIT219\A\L7_1\bin\Debug\L7_1.exe
Значення пі дорівнює 3.141593.
7% учасників змагань з'їли 12.750000 пиріжків з вишнями.
До побачення! Ваші здібності занадто дорого обходяться.
Ми через вас втратили $15600
```



# Модифікатори функції printf()

Модифікатор	Опис
прапорець	П'ять допустимих прапорців (-, +, пробіл, # і 0). Можна вказувати декілька прапорців або не вказувати їх зовсім. Приклад: "%-10d".
цифра (цифри)	Мінімальна ширина поля. Якщо число або рядок, які не вміщуються в це поле, буде використовуватися поле більшої ширини. Приклад: "%4d".
.цифра (.цифри)	Точність. Для перетворень %e, %E і %f вказується кількість цифр, які будуть виведені справа від десяткової крапки. Для перетворень %g і %G задається максимальна кількість значущих цифр. Для перетворень %s визначається максимальна кількість символів, які можуть бути виведені. Для цілочислових перетворень вказується мінімальна кількість цифр, що відображаються; за необхідності для відповідності з цим мінімумом застосовуються провідні нулі. Використання тільки крапки (.) передбачає, що далі йде нуль, тобто %.f – те ж саме, що й %.0f. Приклад: "%5.2f" виводить значення типу float у полі шириною п'ять символів і двома цифрами після десяткової крапки.



# Модифікатори функції printf()

Модифікатор	Опис
<b>h</b>	Використовується зі специфікатором цілочислового перетворення для відображення значень типів <b>short int</b> або <b>unsigned short int</b> . Приклади: "%hu", "%hx" і "%6.4hd".
<b>hh</b>	Використовується зі специфікатором цілочислового перетворення для відображення значень типів <b>signed char</b> або <b>unsigned char</b> . Приклади: "%hhu", "%hhx" і "%6.4hhd".
<b>j</b>	Використовується зі специфікатором цілочислового перетворення для відображення значень <b>intmax_t</b> або <b>uintmax_t</b> . Ці типи визначені в файлі <b>stdint.h</b> . Приклади: "%jd" і "%8jX".
<b>l</b>	Використовується зі специфікатором цілочислового перетворення для відображення значень типів <b>long int</b> або <b>unsigned long int</b> . Приклади: "%ld" і "%8lu".



# Модифікатори функції printf()

Модифікатор	Опис
ll	Використовується зі специфікатором цілочислового перетворення для відображення значень типів <b>long long int</b> або <b>unsigned long long int</b> (стандарт C99). Приклади: "%lld" і "%8llu".
L	Використовується зі специфікатором перетворення значень з рухомою комою для відображення значень типу <b>long double</b> . Приклади: "%Lf" і "%10.4Le".
t	Використовується зі специфікатором цілочислового перетворення для відображення значень <b>ptrdiff_t</b> . Цей тип відповідає різниці між двома вказівниками (стандарт C99). Приклади: "%td" і "%12ti".
z	Використовується зі специфікатором цілочислового перетворення для відображення значень <b>size_t</b> . Цей тип повертається операцією <b>sizeof</b> (стандарт C99). Приклади: "%zd" і "%12zx".





# Прапорці функції printf()

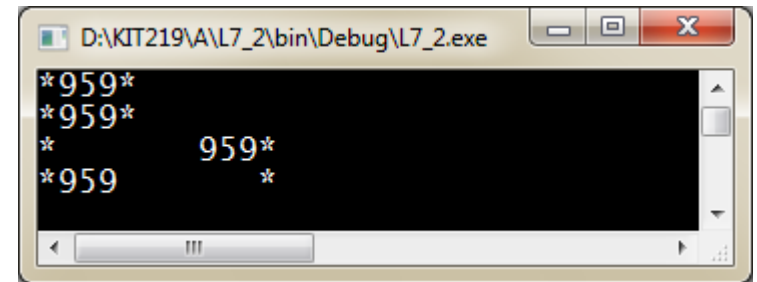
Прапорець	Опис
-	Елемент вирівнюється вліво, тобто зміст буде виведений, починаючи від лівої межі поля. Приклад: "%-20s".
+	Значення зі знаком виводяться зі знаком «+», якщо вони позитивні, та зі знаком «-», якщо вони від'ємні. Приклад: "%+6.2f".
пробіл	Значення зі знаком виводяться з провідним пробілом (але без знаку), якщо вони позитивні, та зі знаком «-», якщо вони від'ємні. Прапорець «+» перевизначає дію пробілу. Приклад: "% 6.2f".
#	Використовує альтернативну форму для специфікаторів перетворення. Виводить провідний 0 для форми %o і провідний 0x або 0X для форм %x і %X. Для усіх форм з рухомою комою прапорець «#» гарантує, що символ десяткової крапки буде виведений, навіть якщо за ним не йдуть цифри. Для форм %g і %G це запобігає видаленню завершальних нулів. Приклади: "%#o", "%#8.0f" і "%+#10.3E".
0	Для числових форм цей прапорець призводить до заповнення порожніх позицій поля провідними нулями, а не пробілами. Даний прапорець ігнорується, якщо присутній прапорець «-» або, якщо для цілочислової форми вказана точність. Приклади: "%010d", "%08.3f".

# Приклади використання

```
#include <stdio.h>
#define  PAGES  959
```

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

Формати цілих чисел



# Приклади використання

## Формати чисел з рухомою комою

```
#include <stdio.h>
```

```
int main(void)
```

```
{  
    // константа, яка оголошена з ключовим словом const  
    const double RENT = 3852.99;
```

```
    printf("%f\n", RENT);
```

```
    printf("%e\n", RENT);
```

```
    printf("%4.2f\n", RENT);
```

```
    printf("%3.1f\n", RENT);
```

```
    printf("%10.3f\n", RENT);
```

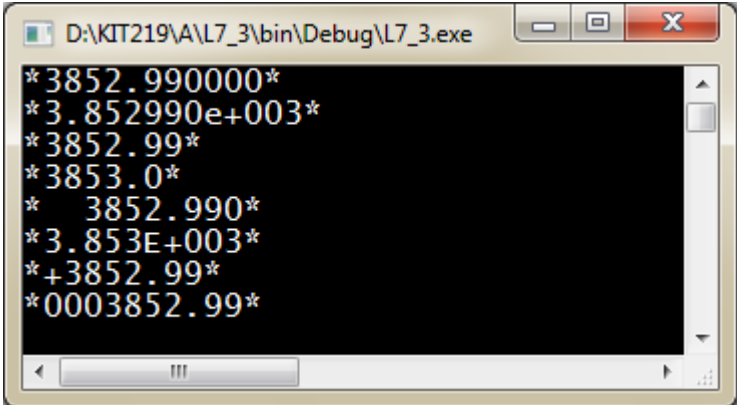
```
    printf("%10.3E\n", RENT);
```

```
    printf("%+4.2f\n", RENT);
```

```
    printf("%010.2f\n", RENT);
```

```
    return 0;
```

```
}
```



```
D:\KIT219\A\L7_3\bin\Debug\L7_3.exe  
*3852.990000*  
*3.852990e+003*  
*3852.99*  
*3853.0*  
* 3852.990*  
*3.853E+003*  
*+3852.99*  
*0003852.99*
```



# Приклади використання

```
#include <stdio.h>
```

Формати цілих чисел

```
int main(void)
```

```
{
```

```
    printf("%x %X %#x\n", 31, 31, 31);
```

```
    printf("**%d**%d**%d**\n", 42, 42, -42);
```

```
    printf("**%5d**%5.3d**%05d**%05.3d**\n", 6, 6, 6, 6);
```

```
    return 0;
```

```
}
```

```
D:\KIT219\A\L7_4\bin\Debug\L7_4.exe
1f 1F 0x1f
**42** 42**-42**
** 6** 006**00006** 006**
```



# Приклади використання

```
#include <stdio.h>
#define BLURB "Authentic imitation!"

int main(void)
{
    printf("[%2s]\n",    BLURB);
    printf("[%24s]\n",   BLURB);
    printf("[%24.5s]\n", BLURB);
    printf("[% -24.5s]\n", BLURB);
    return 0;
}
```

Формати рядків символів

```
D:\KIT219\A\L7_5\bin\Debug\L7_5.exe
[Authentic imitation!]
[ Authentic imitation!]
[ Authentic imitation!]
[Authentic imitation!]
```



# Значення, яке повертає функція printf()

```
#include <stdio.h>
#include <windows.h>
```

```
int main(void)
```

```
{
    int bph2o = 212;
    int rv;
```

```
    SetConsoleOutputCP(1251);
```

```
    rv = printf("Вода закипає при %d градусах за Фаренгейтом.\n",
               bph2o);
```

```
    printf("Функція printf() передала на консоль %d символів.\n",
           rv);
```

```
    return 0;
```

```
}
```

Функція **printf()** повертає кількість символів, що виводиться.



# Виведення довгих рядків

```
#include <stdio.h>  
#include <windows.h>
```

```
int main(void)  
{  
    SetConsoleOutputCP(1251);  
  
    printf("Перший спосіб виводу ");  
    printf("довгого рядка.\n");  
    printf("Другий спосіб виводу \n");  
    printf("довгого рядка.\n");  
    printf("Третій, самий новий спосіб виводу " "довгого рядка.\n");  
  
    return 0;  
}
```

```
D:\KIT219\A\L7_7\bin\Debug\L7_7.exe  
Перший спосіб виводу довгого рядка.  
Другий спосіб виводу довгого рядка.  
Третій, самий новий спосіб виводу довгого рядка.
```



# Функція scanf()

У функції **scanf()** використовується керувальний рядок, за яким йде список параметрів. Керувальний рядок вказує цільові типи даних для потоку символів, що вводяться. У списку аргументів функції **scanf()** застосовуються вказівники на змінні.

## Треба запам'ятати наступні прості правила:

- якщо функція **scanf()** використовується для того, щоб прочитати значення для змінної одного з базових типів, необхідно перед іменем змінної поставити символ **&**;
- якщо функція **scanf()** використовується для читання рядка символьного масиву, символ **&** не потрібен.

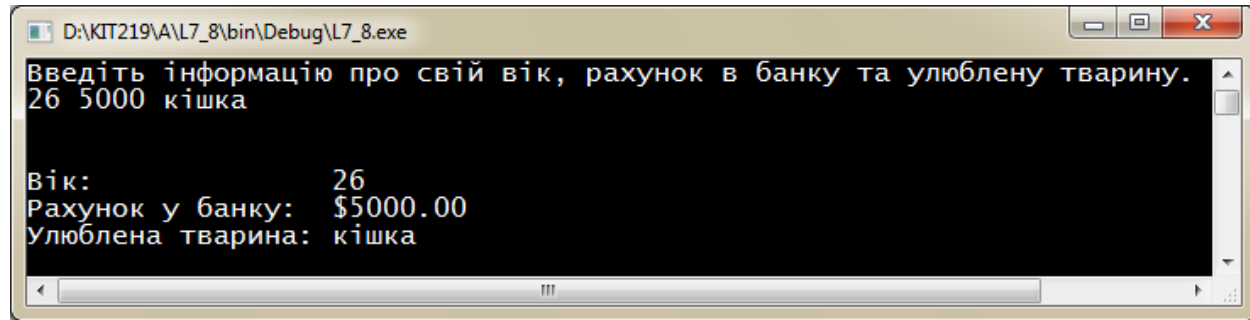
Функція **scanf()** повертає кількість елементів, які вона успішно прочитала. Якщо не прочитано жодного елемента, повертається **0**.



# Приклади використання

```
#include <stdio.h>
#include <windows.h>
int main(void)
{
    int    age;
    float  assets;
    char   pet[30]; // рядок символів

    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);
    printf("Введіть інформацію про свій вік, "
           "рахунок в банку та улюблену тварину.\n");
    // необхідно вказати символ &
    scanf("%d %f", &age, &assets);
    // для масиву символів & не потрібен
    scanf("%s", pet);
    printf("\n\nВік:           %d\n", age);
    printf("Рахунок у банку:  $%.2f\n", assets);
    printf("Улюблена тварина: %s\n", pet);
    return 0;
}
```



```
D:\KIT219\A\L7_8\bin\Debug\L7_8.exe
Введіть інформацію про свій вік, рахунок в банку та улюблену тварину.
26 5000 кішка

Вік:           26
Рахунок у банку:  $5000.00
Улюблена тварина: кішка
```



# Специфікатори функції scanf()

У функції **scanf()** використовується в основному той самий набір специфікаторів перетворення, що й у функції **printf()**.

**Головна відмінність полягає в тому, що**

у функції **printf()** використовуються специфікатори **%f**, **%e**, **%E**, **%g** і **%G** для типів **float** і **double**,

тоді як

у функції **scanf()** вони використовуються тільки для типу **float**, вимагаючи позначення модифікатора **l** для типу **double** (наприклад, **"%le"**, **"%lf"**, **"%lg"**).



# Специфікатори функції scanf()

Специфікатор	Значення
<code>%c</code>	Інтерпретує введені дані як символ
<code>%d</code>	Інтерпретує введені дані як десяткове ціле число зі знаком
<code>%e, %f, %g, %a</code>	Інтерпретує введені дані як число з рухомою комою ( <code>%a</code> з'явився у C99)
<code>%E, %F, %G, %A</code>	Інтерпретує введені дані як число з рухомою комою ( <code>%A</code> з'явився у C99)
<code>%i</code>	Інтерпретує введені дані як десяткове ціле число зі знаком
<code>%o</code>	Інтерпретує введені дані як вісімкове ціле число зі знаком
<code>%p</code>	Інтерпретує введені дані як вказівник (адреса)
<code>%s</code>	Інтерпретує введені дані як рядок. Введення починається з першого символу, що не є пробільним, і містить усі символи до наступного пробільного символу
<code>%u</code>	Інтерпретує введені дані як десяткове ціле число без знаку
<code>%x, %X</code>	Інтерпретує введені дані як шістнадцяткове ціле число зі знаком



# Модифікатори функції scanf()

Модифікатор	Значення
*	Подавляє присвоювання. Приклад: "%*d".
цифра (цифри)	Максимальна ширина поля. Введення припиняється, коли досягнута максимальна ширина поля або якщо виявлено перший символ пробілу (в залежності від того, що станеться раніше). Приклад: "%10s".
hh	Зчитує ціле число як <b>signed char</b> або <b>unsigned char</b> . Приклади: "%hhd", "%hhu".
ll	Зчитує ціле число як <b>long long</b> або як <b>unsigned long long</b> (стандарт C99). Приклади: "%lld", "%llu".



# Модифікатори функції scanf()

Модифікатор	Значення
h, l або L	Специфікатори "%hd" і "%hi" вказують, що значення буде збережено з типом <b>short int</b> .
	Специфікатори "%ho", "%hx" і "%hu" визначають, що значення буде збережено з типом <b>unsigned short int</b> .
	Специфікатори "%ld" і "%li" вказують, що значення буде збережено з типом <b>long</b> .
	Специфікатори "%lo", "%lx" і "%lu" визначають, що значення буде збережено з типом <b>unsigned long</b> .
	Специфікатори "%le", "%lf" і "%lg" вказують, що значення буде збережено з типом <b>double</b> .
	Використання модифікатора L замість l у поєднанні з e, f і g визначає, що значення буде збережено з типом <b>long double</b> .
	У відсутності цих модифікаторів d, i, o і x вказують на тип <b>int</b> , а e, f і g – на тип <b>float</b> .



# Модифікатори функції scanf()

Модифікатор	Значення
j	Коли за ним йде специфікатор цілочислового перетворення, вказує на використання типів <code>intmax_t</code> або <code>uintmax_t</code> (стандарт C99). Приклади: "%jd", "%ju".
z	Коли за ним йде специфікатор цілочислового перетворення, вказує на використання типу, що повертається операцією <code>sizeof</code> (стандарт C99). Приклади: "%zd", "%zo".
t	Коли за ним йде специфікатор цілочислового перетворення, вказує на використання типу, який призначений для представлення різниці між двома вказівниками (стандарт C99). Приклади: "%td", "%tx".



# Модифікатор \* у функції printf()

Якщо не треба фіксувати ширину поля заздалегідь, а бажано, щоб її визначила сама програма, то це можна зробити, вказавши замість числа, що задає ширину поля, **модифікатор \***.

Крім цього, необхідно додати аргумент для повідомлення функції, якою повинна бути ширина поля.

Тобто при наявності специфікатора перетворення **%\*d** список аргументів повинен містити значення для модифікатора **\*** і значення для **d**.

Такий метод можна застосовувати також зі значеннями з рухомою комою, щоб вказувати точність і ширину поля.

# Приклад використання

```
#include <stdio.h>
#include <windows.h>
```

```
int main(void)
```

```
{
```

```
    unsigned    width, precision;
```

```
    int         number = 256;
```

```
    double     weight = 242.5;
```

```
    SetConsoleOutputCP(1251);
```

```
    printf("Введіть ширину поля: ");
```

```
    scanf("%d", &width);
```

```
    printf("Значення дорівнює: %*d:\n\n", width, number);
```

```
    printf("Тепер введіть ширину та точність: ");
```

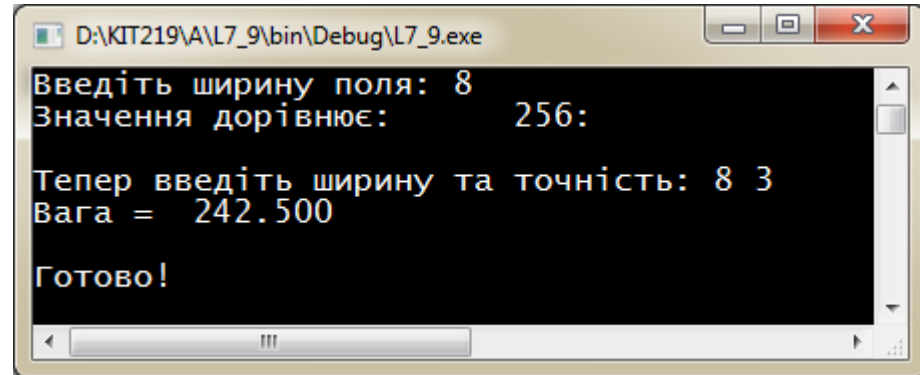
```
    scanf("%d %d", &width, &precision);
```

```
    printf("Вага = %*.*f\n", width, precision, weight);
```

```
    printf("\nГотово!\n");
```

```
    return 0;
```

```
}
```



```
D:\KIT219\A\L7_9\bin\Debug\L7_9.exe
Введіть ширину поля: 8
Значення дорівнює: 256:

Тепер введіть ширину та точність: 8 3
Вага = 242.500

Готово!
```





# Модифікатор \* у функції scanf()

У випадку функції **scanf()** модифікатор **\***, розміщений між символом **%** і буквою специфікатора, змушує функцію пропускати відповідний ввід.

```
#include <stdio.h>
#include <windows.h>

int main(void)
{
```

```
    int n;
```

```
    SetConsoleOutputCP(1251);
```

```
    printf("Введіть три цілих числа: ");
```

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

```
    printf("\nОстаннім цілим числом було %d\n", n);
```

```
    return 0;
```

```
}
```

