

# Рядки. Функції для вводу-виводу рядків



*Лекція №13*

*Дисципліна «Програмування»*



# Визначення рядків

**Символьний рядок** – це масив елементів типу **char**, що завершується нульовим символом (`'\0'`).

Рядки можна визначити за допомогою:

- константних рядків (рядкових літералів);

```
#define MSG "Константа, яка є рядком символів."
```

- масивів типу **char**;

```
char words[MAXLENGTH] =  
    "Рядок символів, що зберігається в масиві.";
```

- вказівників на тип **char**.

```
const char *pt1 = "Вказівник, що посилається на рядок.";
```

Програма також повинна забезпечити місце для зберігання рядка.

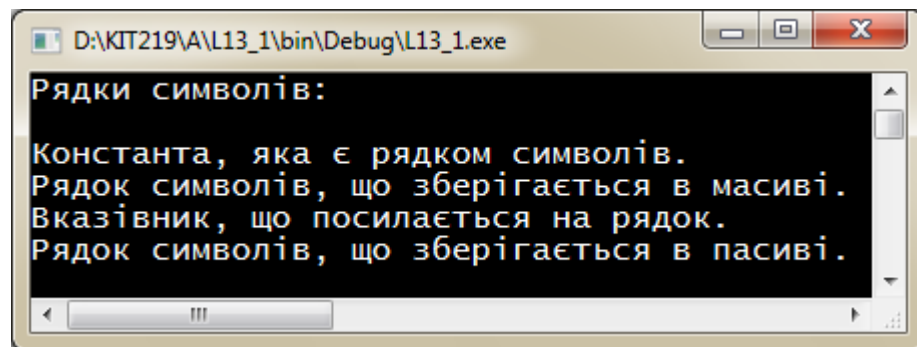
# Способи представлення рядків

```
#include <stdio.h>
#include <windows.h>
#define MSG "Константа, яка є рядком символів."
#define MAXLENGTH 81

int main(void)
{
    char words[MAXLENGTH] =
        "Рядок символів, що зберігається в масиві.";
    const char *pt1 = "Вказівник, що посиляється на рядок.";

    SetConsoleOutputCP(1251);
    puts("Рядки символів:\n");
    puts(MSG);
    puts(words);
    puts(pt1);
    words[34] = 'п';
    puts(words);

    return 0;
}
```





# Масиви символічних рядків

Наступне оголошення ініціалізує масив **m1** символами визначеного рядка:

```
const char m1[40] = "Спробуйте вкластися в один рядок.";
```

Ключове слово **const** позначає намір не змінювати цей рядок. Вказана форма є скороченням для стандартної форми ініціалізації масиву:

```
const char m1[40] = { 'С', 'п', 'р', 'о', 'б', 'у', 'й',  
                      'т', 'е', '_, 'в', 'к', 'л', 'а',  
                      'с', 'т', 'и', 'с', 'я', '_, 'в',  
                      '_, 'о', 'д', 'и', 'н', '_, 'р',  
                      'я', 'д', 'о', 'к', '., '\0' };
```

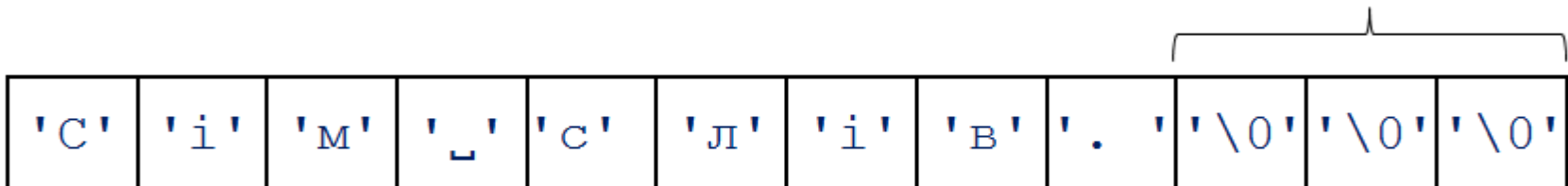
Якщо розмір не вказано в оголошенні, при ініціалізації масиву, компілятор самотійно визначить його розмір:

```
const char m2[] = "Довжина масиву визначається автоматично.";
```



# Ініціалізація символічних масивів

Зайві елементи ініціалізуються символом '\0'



```
const char sentence[12] = "Сім слів.";
```

Всередині оголошення розмір масиву повинен обчислюватися як цілочислове значення.

```
int n = 8;  
char cookies[1];      // допустимо  
char cakes[2 + 5];    // допустимо, оскільки розмір є  
                      // константним виразом  
char pies[2 * sizeof(long double) + 1]; // допустимо  
char crumbs[n];       // в C99 це масив змінної довжини
```



# Ініціалізація символьних масивів

Ім'я символьного масиву видає адресу першого елемента масиву.

Зазначене нижче твердження є справедливим:

```
char car[10] = "Луна";  
car == &car[0];  
*car == 'Л'      і      *(car + 1) == car[1] == 'y'
```

Для ініціалізації рядка можна використовувати форму запису з вказівниками:

```
const char *pt1 = "Рядок символів.";
```

Це оголошення дуже близьке до такого оголошення:

```
const char ar1[] = "Рядок символів.";
```

Обидва оголошення означають, що **pt1** і **ar1** є адресами рядків.



# Масиви та вказівники

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

```
int main(void)
```

```
{
```

```
    char heart[] = "Константа";
```

```
    const char *head = "Вказівник";
```

```
    SetConsoleOutputCP(1251);
```

```
    for(int i = 0; i < 9; i++)
```

```
        putchar(heart[i]);
```

```
    putchar('\n');
```

```
    for(int i = 0; i < 9; i++)
```

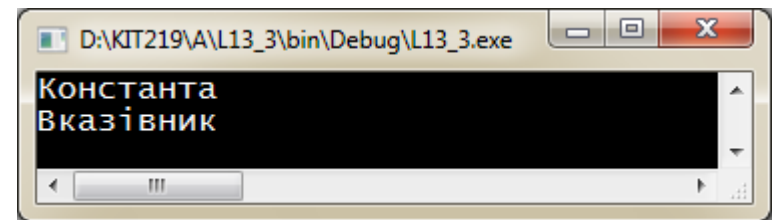
```
        putchar(head[i]);
```

```
    putchar('\n');
```

```
    return 0;
```

```
}
```

Головна різниця між **heart** і **head** полягає в тому, що ім'я масиву **heart** є константою, а вказівник **head** – змінною.





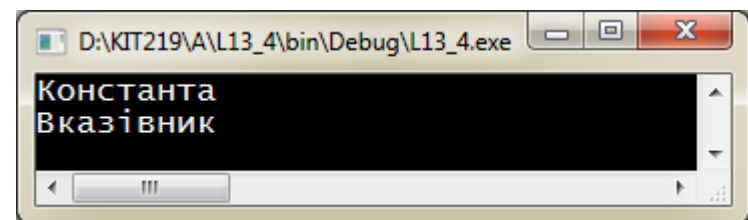
# Масиви та вказівники

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

int main(void)
{
    char heart[] = "Константа";
    const char *head = "Вказівник";

    SetConsoleOutputCP(1251);

    for(int i = 0; i < 9; i++)
        putchar(*(heart + i));
    putchar('\n');
    for(int i = 0; i < 9; i++)
        putchar(*(head + i));
    putchar('\n');
    return 0;
}
```







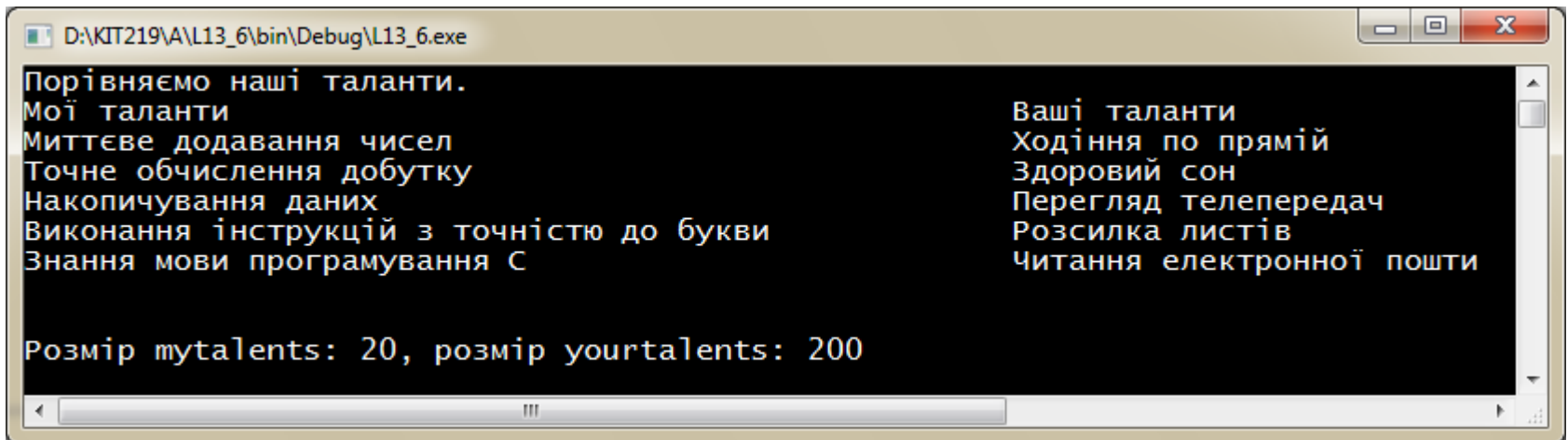
# Масиви символічних рядків

```
#include <stdio.h>
#include <windows.h>
#define SLEN 40
#define LIM 5
int main(void)
{
    const char *mytalents[LIM] = {
        "Миттєве додавання чисел",
        "Точне обчислення добутку",
        "Накопичування даних",
        "Виконання інструкцій з точністю до букви",
        "Знання мови програмування C"    };

    char yourtalents[LIM][SLEN] = {
        "Ходіння по прямій",
        "Здоровий сон",
        "Перегляд телепередач",
        "Розсилка листів",
        "Читання електронної пошти"    };
```

# Масиви символічних рядків

```
SetConsoleOutputCP(1251);
puts("Порівняємо наші таланти.");
printf("%-52s %-25s\n", "Мої таланти", "Ваші таланти");
for(int i = 0; i < LIM; i++)
    printf("%-52s %-25s\n", mytalents[i], yourtalents[i]);
printf("\n\nРозмір mytalents: %d, розмір yourtalents:
    %d\n\n", sizeof(mytalents), sizeof(yourtalents));
return 0;
}
```



```
D:\KIT219\A\L13_6\bin\Debug\L13_6.exe
Порівняємо наші таланти.
Мої таланти                                     Ваші таланти
Миттєве додавання чисел                         Ходіння по прямій
Точне обчислення добутку                       Здоровий сон
Накопичування даних                             Перегляд телепередач
Виконання інструкцій з точністю до букви        Розсилка листів
Знання мови програмування C                     Читання електронної пошти

Розмір mytalents: 20, розмір yourtalents: 200
```



# Прямокутні та зубчаті масиви

'Я'	'б'	'л'	'у'	'к'	'о'	'\0'	'\0'	'\0'
'Г'	'р'	'у'	'ш'	'а'	'\0'	'\0'	'\0'	'\0'
'А'	'п'	'е'	'л'	'ь'	'с'	'и'	'н'	'\0'

'Я'	'б'	'л'	'у'	'к'	'о'	'\0'		
'Г'	'р'	'у'	'ш'	'а'	'\0'			
'А'	'п'	'е'	'л'	'ь'	'с'	'и'	'н'	'\0'

```
char fruit1[3][9] = {  
    "Яблуко",  
    "Груша",  
    "Апельсин"  
};
```

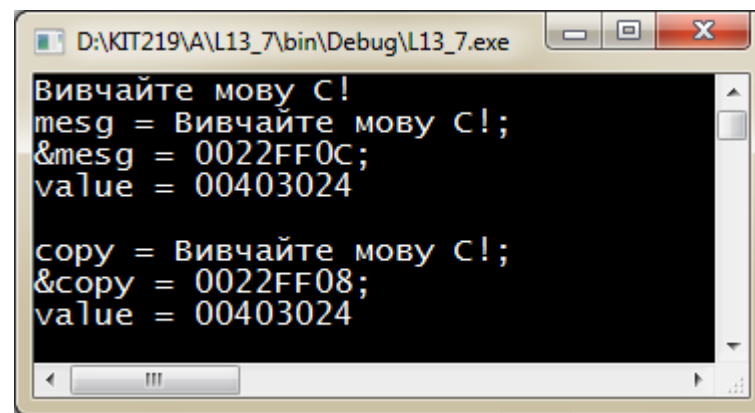
```
const char * fruit2[3] = {  
    "Яблуко",  
    "Груша",  
    "Апельсин"  
};
```

# Вказівники та рядки

```
#include <stdio.h>
#include <windows.h>
int main(void)
{
    const char *mesg = "Вивчайте мову C!";
    const char *copy;

    SetConsoleOutputCP(1251);

    copy = mesg;
    printf("%s\n", copy);
    printf("mesg = %s;\n&mesg = %p;\nvalue = %p\n\n",
           mesg, &mesg, mesg);
    printf("copy = %s;\n&copy = %p;\nvalue = %p\n\n",
           copy, &copy, copy);
    return 0;
}
```





# Функції для вводу рядків

Спочатку необхідно підготувати місце для розміщення рядка після його читання.

```
char name [ 81 ] ;
```

Після надання місця для рядка його можна прочитати. Бібліотека **C** пропонує три функції, які дозволяють зчитувати рядки: **scanf()**, **gets()** і **fgets()**.

Часто бажано, щоб програма могла зчитати одразу весь рядок, а не одне слово.

Саме для цього застосовується функція **gets()**.

Вона читає весь рядок до символу нового рядка, відкидає цей символ і зберігає інші символи, додаючи нульовий символ, щоб створити рядок.

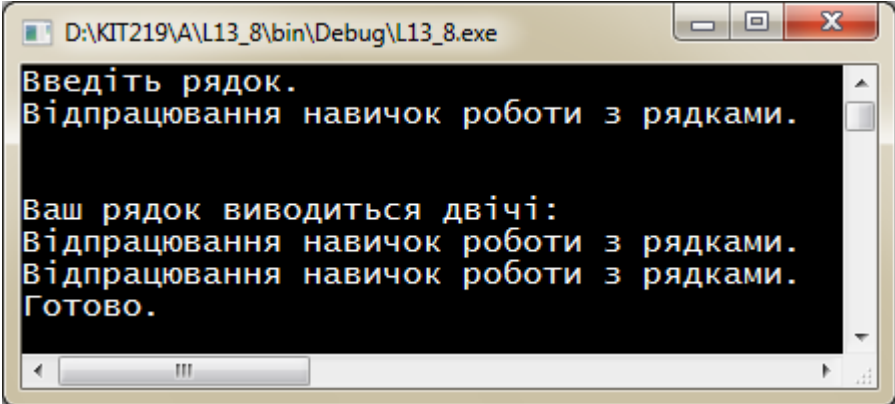
# Функція gets()

```
#include <stdio.h>
#include <windows.h>
#define STLEN 81

int main(void)
{
    char words[STLEN];

    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);

    puts("Введіть рядок.");
    gets(words);
    printf("\n\nВаш рядок виводиться двічі:\n");
    printf("%s\n", words);
    puts(words);
    puts("Готово.\n\n");
    return 0;
}
```





# Альтернативи для функції `gets()`

Традиційною альтернативою для `gets()` є функція `fgets()`, яка має дещо більш складний інтерфейс і трохи по-іншому обробляє введені дані.

Крім того, в стандарті **C11** до загального набору додана функція `gets_s()`.

Вона більш схожа на `gets()` і її легше використовувати в існуючому коді в якості заміни.

Проте, вона є частиною необов'язкового розширення сімейства функцій вводу-виводу **stdio.h**, і тому компілятори стандарту **C11** не зобов'язані її підтримувати.



# Функція fgets()

```
#include <stdio.h>
#include <windows.h>
#define STLEN 15

int main(void)
{
    char words[STLEN];

    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);

    puts("Введіть рядок.");
    fgets(words, STLEN, stdin);
    printf("Ваш рядок виводиться двічі (за допомогою puts(), "
           "а потім fputs()):\n");
    puts(words);
    fputs(words, stdout);
    puts("\nВведіть ще один рядок.");
    fgets(words, STLEN, stdin);
```



# Функція fgets()

```
printf("Ваш рядок виводиться двічі (за допомогою puts()),"  
      " а потім fputs()):\n");  
puts(words);  
fputs(words, stdout);  
puts("\n\nГотово.");  
return 0;  
}
```

```
D:\KIT219\A\L13_9\bin\Debug\L13_9.exe  
Введіть рядок.  
Рядок №1.  
Ваш рядок виводиться двічі (за допомогою puts(), а потім fputs()):  
Рядок №1.  
  
Рядок №1.  
  
Введіть ще один рядок.  
Рядок №2 буде трохи довшим.  
Ваш рядок виводиться двічі (за допомогою puts(), а потім fputs()):  
Рядок №2 буде  
Рядок №2 буде  
  
Готово.
```



# Функція `fgets()`

Відмінності функції `fgets()` від функції `gets()`.

- вона приймає **другий аргумент**, який задає максимальну кількість символів для читання. Якщо цей аргумент має значення **n**, то функція `fgets()` прочитає **n-1** символів або буде читати рядок до появи символу нового рядка в залежності від того, що відбудеться раніше;
- якщо функція `fgets()` стикається з символом нового рядка, вона зберігає його в рядку, на відміну від функції `gets()`, яка відкидає його;
- функція `fgets()` приймає **третій аргумент**, який вказує файл, з якого необхідно проводити зчитування.

Для зчитування з клавіатури в якості цього аргументу використовується **stdin**. Цей ідентифікатор визначений у файлі заголовку **stdio.h**.



# Функція fgets()

```
#include <stdio.h>
#include <windows.h>
#define STLEN 10
```

```
int main(void)
{
```

```
    char words[STLEN];
```

```
    SetConsoleCP(1251);
```

```
    SetConsoleOutputCP(1251);
```

```
    puts("Введіть рядки (або порожній рядок для виходу з"
        " програми):");
```

```
    while(fgets(words, STLEN, stdin) != NULL
        && words[0] != '\n')
```

```
        fputs(words, stdout);
```

```
    puts("Готово.");
```

```
    return 0;
```

```
}
```

```
D:\KIT219\A\L13_10\bin\Debug\L13_10.exe
Введіть рядки (або порожній рядок для виходу з програми):
До речі, функція fgets()
До речі, функція fgets()
також повертає порожній вказівник,
також повертає порожній вказівник,
якщо вона зустрічає кінець файлу.
якщо вона зустрічає кінець файлу.
Готово.
```



# Функція `gets_s()`

Відмінності функції `gets_s()` від функції `fgets()`:

- 1) Функція `gets_s()` просто виконує зчитування зі стандартного вводу, тому вона не потребує третього аргументу.
- 2) Якщо функція `gets_s()` зчитує символ нового рядка, то відкидає його, а не зберігає.
- 3) Якщо `gets_s()` прочитає максимальну кількість символів, і серед них символ нового рядка буде відсутній, вона встановлює перший символ цільового масиву в нульовий. Потім вона читає і відкидає наступні символи, які були введені, поки не зустрінеться символ нового рядка або ознака кінця файлу. Нарешті, функція повертає нульовий вказівник. Вона викликає функцію «обробника», яка залежить від реалізації (або ж обрану функцію) і може призвести до виходу з програми або припиненню її роботи.



# Функція s\_gets()

```
char * s_gets(char *st, int n)
{
    char *ret_val;
    int i = 0;

    ret_val = fgets(st, n, stdin);
    if (ret_val)        // тобто ret_val != NULL
    {
        while (st[i] != '\n' && st[i] != '\0')
            i++;
        if (st[i] == '\n')
            st[i] = '\0';
        else
            while (getchar() != '\n')
                continue;
    }
    return ret_val;
}
```

# Функція scanf()

```
#include <stdio.h>
#include <windows.h>
int main(void)
{
    char name1[11], name2[11];
    int count;

    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);

    printf("Введіть два імені.\n");
    count = scanf("%5s %8s", name1, name2);
    printf("Прочитано %d імені: %s і %s.\n",
        count, name1, name2);
    return 0;
}
```

D:\KIT219\A\L13\_13\bin\Debug\L13\_13.exe  
Введіть два імені.  
Іван Микита  
Прочитано 2 імені: Іван і Микита.

D:\KIT219\A\L13\_13\bin\Debug\L13\_13.exe  
Введіть два імені.  
Павло Варфоломій  
Прочитано 2 імені: Павло і Варфолом.

D:\KIT219\A\L13\_13\bin\Debug\L13\_13.exe  
Введіть два імені.  
Максиміліан Єгор  
Прочитано 2 імені: Макси і міліан.

# Функція puts()

```
#include <stdio.h>
#include <windows.h>
#define DEF "Я - рядок, який визначений директивою #define."
int main(void)
{
    char str1[80] =
        "Масив був ініціалізований певним значенням.";
    const char * str2 =
        "Вказівник був ініціалізований певним значенням.";
    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);
    puts("Я - аргумент функції puts().");
    puts(DEF);
    puts(str1);      puts(str2);
    puts(&str1[5]);
    puts(str2+4);
    return 0;
}
```



# Функція `fputs()`

Функція `fputs()` являє собою версію `puts()`, що орієнтована на файли. Важливі відмінності між ними полягають у наступному:

- 1) Функція `fputs()` приймає другий аргумент, що вказує файл, в який повинен відбуватися запис. Для виводу на консоль можна застосовувати аргумент `stdout` (від `standard output` – стандартний вивід), який визначений в `stdio.h`.
- 2) На відміну від `puts()`, функція `fputs()` не додає автоматично до виводу символ нового рядка.

```
char line[81];  
while(fgets(line, 81, stdin))  
    fputs(line, stdout);
```