

Функції для роботи з рядками



Лекція №14

Дисципліна «Програмування»



Функція `strlen()`

Функція `strlen()` визначає довжину рядка і виконується поки не зустрине нульовий символ, який не входить до загальної довжини рядка.

Прототип функції:

```
size_t strlen(const char * string);
```

Тип `size_t` визначається відносно арифметичних можливостей процесора в файлі `<stddef.h>`.

`size_t` – це беззнаковий цілий тип, що призначений для представлення розміру будь-якого об'єкта в пам'яті для конкретної реалізації.

Максимальний розмір `size_t` записаний в макроконстанті `SIZE_MAX`, яка визначена у файлі заголовку `<stdint.h>`.

`size_t` повинен бути, як мінімум, 16 біт.

Застосування strlen()

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

```
void fit(char *, unsigned int);
```

```
int main(void)
```

```
{
```

```
    char mesg[] = "Все повинно бути максимально простим,"
                  " але не більше.";
```

```
    puts(mesg);
```

```
    fit(mesg, 35);
```

```
    puts(mesg);
```

```
    puts("Розглянемо ще декілька рядків.");
```

```
    puts(mesg + 36);
```

```
    return 0;
```

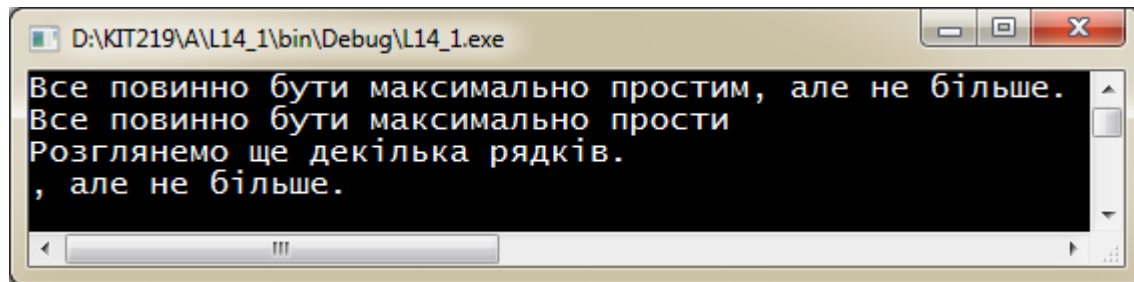
```
}
```

```
void fit(char *string, unsigned int size)
```

```
{
```

```
    if(strlen(string) > size) string[size] = '\0';
```

```
}
```



```
D:\KIT219\A\L14_1\bin\Debug\L14_1.exe
Все повинно бути максимально простим, але не більше.
Все повинно бути максимально прости
Розглянемо ще декілька рядків.
, але не більше.
```



Функція strcat()

Прототип функції

```
char * strcat(char *destptr, const char *srcptr);
```

Функція **strcat()** (від **string concatenation** – конкатенація рядків) додає копію рядка **srcptr** наприкінці рядка **destptr**. В якості аргументів вона приймає два рядки.

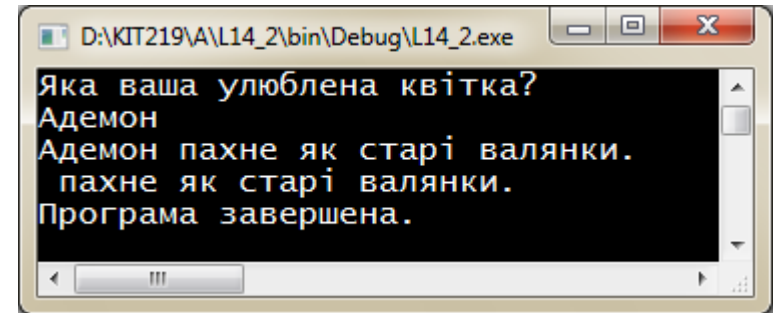
Нульовий символ кінця рядка **destptr** замінюється першим символом рядка **srcptr**, і новий нульовий символ додається наприкінці вже нового рядка, що сформований об'єднанням символів двох рядків в рядку **destptr**.

Функція **strcat()** повертає тип **char *** (тобто вказівник на **char**) – адресу першого символу рядка, наприкінці якого був доданий другий рядок.



Застосування strcat()

```
#include <stdio.h>
#include <windows.h>
#include <string.h>
#define SIZE 80
char *s_gets(char *st, int n);
int main(void)
{
    char flower[SIZE];
    char addon[] = " пахне як старі валянки.";
    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);
    puts("Яка ваша улюблена квітка?");
    if(s_gets(flower, SIZE))
    {
        strcat(flower, addon);
        puts(flower); puts(addon);
    }
    else puts("Знайдено кінець файлу!");
    puts("Програма завершена.");
    return 0;
}
```





Застосування strcat()

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

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



Функція `strncat()`

Прототип функції

```
char *strncat(char *destptr, char *srcptr, size_t num);
```

Функція додає перші **num** символів рядка **srcptr** наприкінці рядка **destptr**, плюс символ кінця рядка.

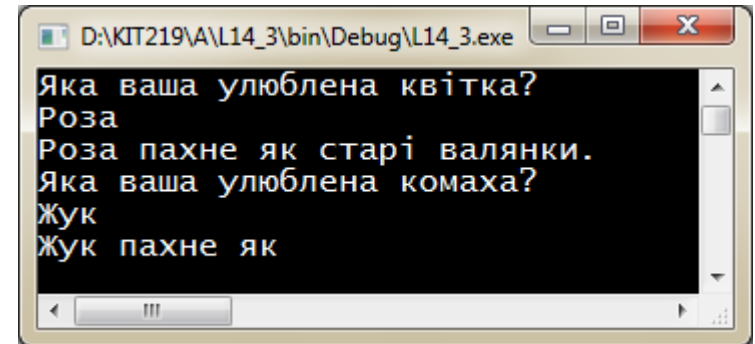
Якщо рядок **srcptr** більше ніж кількість символів **num**, що копіюються, то після скопійованих символів неявно додається символ кінця рядка.

Функція **strcat()** не перевіряє, чи вміщується другий рядок у перший масив.

Якщо не забезпечити достатній простір для першого масиву, то виникне проблема з переповненням сусідніх комірок пам'яті надлишковими символами.

Застосування strncat()

```
#include <stdio.h>
#include <windows.h>
#include <string.h>
#define SIZE 30
#define BUGSIZE 13
char *s_gets(char *st, int n);
int main(void)
{
    char flower[SIZE];
    char addon[] = " пахне як старі валянки.";
    char bug[BUGSIZE];
    int available;
    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);
    puts("Яка ваша улюблена квітка?");
    s_gets(flower, SIZE);
    if((strlen(addon) + strlen(flower) + 1) <= SIZE)
        strncat(flower, addon);
    puts(flower);
    puts("Яка ваша улюблена комаха?");
    s_gets(bug, BUGSIZE);
```





Застосування strncat()

```
available = BUGSIZE - strlen(bug) - 1;
strncat(bug, addon, available); puts(bug);
return 0;
}

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



Функція strcmp()

Прототип функції:

```
int strcmp(const char *string1, const char *string2);
```

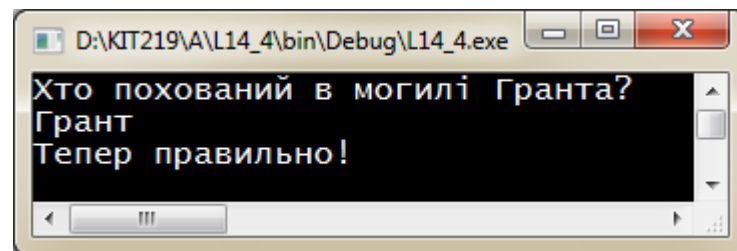
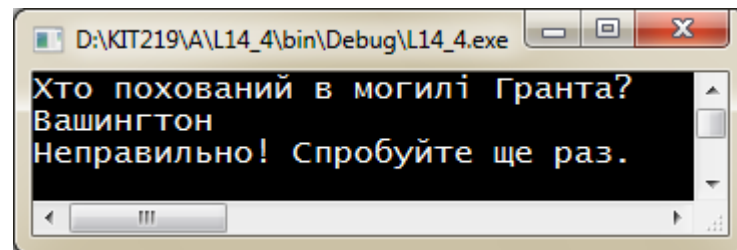
Функція `strcmp()` по черзі порівнює символи двох рядків `string1` і `string2` поки не буде досягнутий кінець рядка.

Як тільки будуть знайдені перші неоднакові символи, функція проаналізує числові коди цих символів. Чий код виявиться більшим, той рядок і буде вважатися більшим.

Функція повертає декілька значень: нульове значення говорить про те, що обидва рядки збігаються; значення більше нуля вказує на те, що рядок `string1` більше ніж рядок `string2`; значення менше нуля свідчить про те, що рядок `string1` менше ніж рядок `string2`.

Застосування strcmp()

```
#include <stdio.h>
#include <windows.h>
#define ANSWER "Грант"
#define SIZE 40
char *s_gets(char *st, int n);
int main(void)
{
    char mas[SIZE];
    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);
    puts("Хто похований в могилі Гранта?");
    s_gets(mas, SIZE);
    while(strcmp(mas, ANSWER) != 0)
    {
        puts("Неправильно! Спробуйте ще раз.");
        s_gets(mas, SIZE);
    }
    puts("Тепер правильно!");
    return 0;
}
```





Застосування strcmp()

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

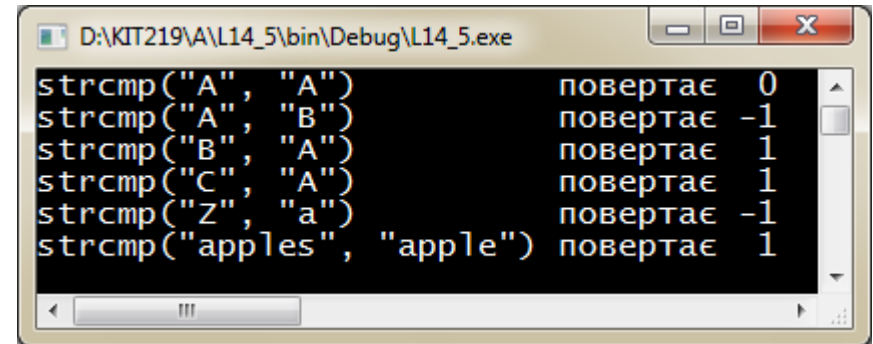
    ret_val = fgets(st, n, stdin);
    if(ret_val)
    {
        while(st[i] != '\n' && st[i] != '\0')
            i++;
        if(st[i] == '\n')
            st[i] = '\0';
        else
            while(getchar() != '\n')
                continue;
    }
    return ret_val;
}
```

Застосування strcmp()

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

int main(void)
{
    SetConsoleOutputCP(1251);

    printf("strcmp(\"A\", \"A\")           повертає ");
    printf("%2d\n", strcmp("A", "A"));
    printf("strcmp(\"A\", \"B\")           повертає ");
    printf("%2d\n", strcmp("A", "B"));
    printf("strcmp(\"B\", \"A\")           повертає ");
    printf("%2d\n", strcmp("B", "A"));
    printf("strcmp(\"C\", \"A\")           повертає ");
    printf("%2d\n", strcmp("C", "A"));
    printf("strcmp(\"Z\", \"a\")           повертає ");
    printf("%2d\n", strcmp("Z", "a"));
    printf("strcmp(\"apples\", \"apple\") повертає ");
    printf("%2d\n", strcmp("apples", "apple"));
    return 0;
}
```



strcmp comparison	повертає
strcmp("A", "A")	0
strcmp("A", "B")	-1
strcmp("B", "A")	1
strcmp("C", "A")	1
strcmp("Z", "a")	-1
strcmp("apples", "apple")	1

Застосування `strncmp()`

```
#include <stdio.h>
#include <windows.h>
#include <string.h>
#define LISTSIZE 6
int main(void)
```

```
{
    const char *list[LISTSIZE] = { "астрономія", "астатизм",
                                     "астрофізика", "остракізм",
                                     "астеризм",    "астролябія" };

```

```
    int count = 0;
```

```
    SetConsoleOutputCP(1251);
```

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

```
        if(strncmp(list[i], "астро", 5) == 0)
```

```
        {
```

```
            printf("Знайдено: %s\n", list[i]);
```

```
            count++;
```

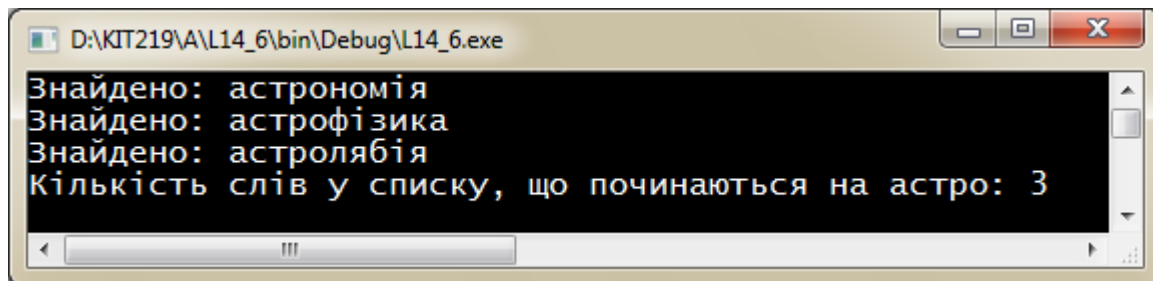
```
        }
```

```
    printf("Кількість слів у списку, "
```

```
           " що починаються на астро: %d\n", count);
```

```
    return 0;
```

```
}
```



```
D:\KIT219\A\L14_6\bin\Debug\L14_6.exe
Знайдено: астрономія
Знайдено: астрофізика
Знайдено: астролябія
Кількість слів у списку, що починаються на астро: 3
```



Функції `strcpy()` і `strncpy()`

Прототипи функцій:

```
char *strcpy(char *destptr, const char *srcptr);
```

```
char *strncpy(char *destptr, const char *srcptr, size_t num);
```

Функція `strncpy()` копіює перші **num** символів рядка **srcptr** до рядка **destptr**.

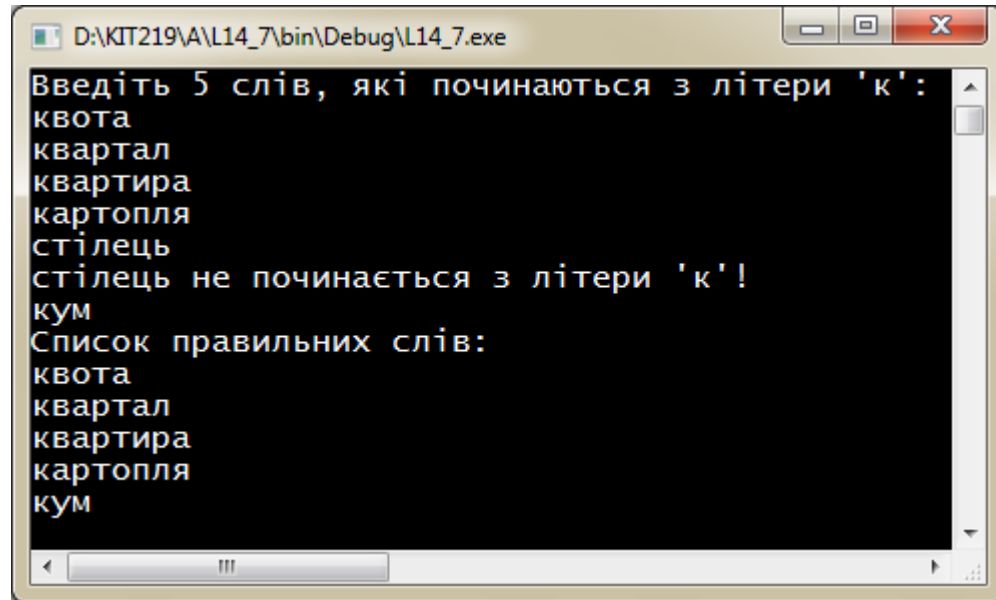
Якщо кінець рядка **srcptr** (символ кінця рядка) досягнутий перш ніж були скопійовані **num** символів, до скопійованих символів наприкінці рядка **destptr** додається нульовий символ, після чого, рядок вважається скопійованим.

Якщо ж рядок призначення виявиться меншим за **num**, тоді будуть скопійовані символи, які помістяться в **destptr**, враховуючи, що наприкінці рядка обов'язково повинен бути символ кінця рядка.

Застосування strcpy()

```
#include <stdio.h>
#include <windows.h>
#include <string.h>
#define SIZE 40
#define LIM 5

char *s_gets(char *st, int n);
int main(void)
{
    char qwords[LIM][SIZE];
    char temp[SIZE];
    int i = 0;
    SetConsoleCP(1251); SetConsoleOutputCP(1251);
    printf("Введіть %d слів, які починаються"
           " з літери \'к\':\n", LIM);
    while(i < LIM && s_gets(temp, SIZE))
    {
        if(temp[0] != 'к')
            printf("%s не починається з літери \'к\':\n", temp);
        else { strcpy(qwords[i], temp); i++; }
    }
}
```



```
D:\KIT219\A\L14_7\bin\Debug\L14_7.exe
Введіть 5 слів, які починаються з літери 'к':
квота
квартал
квартира
картопля
стілець
стілець не починається з літери 'к'!
кум
Список правильних слів:
квота
квартал
квартира
картопля
кум
```




Застосування `strcpy()`

```
puts("Список правильних слів:");  
for(i = 0; i < LIM; i++) puts(qwords[i]);  
return 0;  
}  
  
char *s_gets(char *st, int n)  
{  
    char *ret_val;  
    int i = 0;  
    ret_val = fgets(st, n, stdin);  
    if(ret_val)  
    {  
        while(st[i] != '\n' && st[i] != '\0')  
            i++;  
        if(st[i] == '\n') st[i] = '\0';  
        else  
            while(getchar() != '\n') continue;  
    }  
    return ret_val;  
}
```

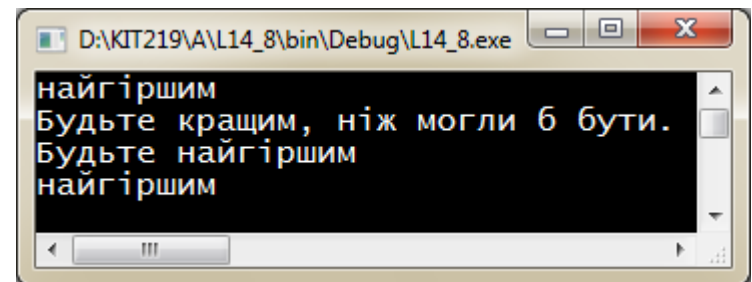
Застосування strcpy()

```
#include <stdio.h>
#include <windows.h>
#include <string.h>           // оголошення strcpy()
#define WORDS "найгіршим"
#define SIZE 40

int main(void)
{
    const char *orig = WORDS;
    char copy[SIZE] = "Будьте кращим, ніж могли б бути.";
    char *ps;

    SetConsoleOutputCP(1251);

    puts(orig);
    puts(copy);
    ps = strcpy(copy + 7, orig);
    puts(copy);
    puts(ps);
    return 0;
}
```



Застосування strncpy()

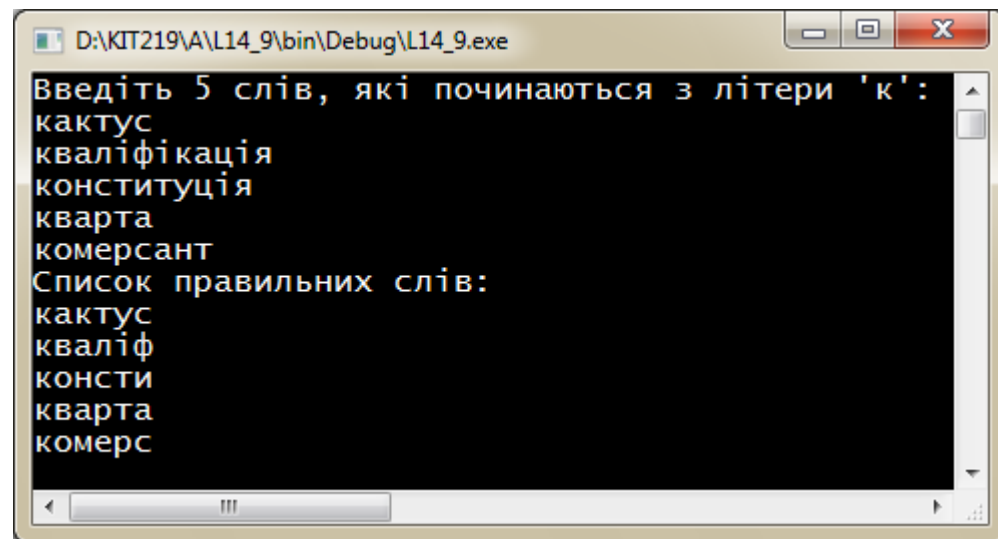
```
#include <stdio.h>
#include <windows.h>
#include <string.h>      // оголошення strncpy()

#define SIZE 40
#define TARGSIZE 7
#define LIM 5

char *s_gets(char *st, int n);

int main(void)
{
    char qwords[LIM][TARGSIZE];
    char temp[SIZE];
    int i = 0;

    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);
    printf("Введіть %d слів, які починаються"
           " з літери \'к\':\n", LIM);
```





Застосування strncpy()

```
while(i < LIM && s_gets(temp, SIZE))
{
    if(temp[0] != 'к')
        printf("%s не починається "
               "з літери \'к\'!\n", temp);
    else
    {
        strncpy(qwords[i], temp, TARGSIZE - 1);
        qwords[i][TARGSIZE - 1] = '\0';
        i++;
    }
}
puts("Список правильних слів:");
for(i = 0; i < LIM; i++)
    puts(qwords[i]);
return 0;
}
```



Функція `sprintf()`

Прототип функції:

```
int sprintf(char *buffer, const char *format, ...);
```

Функція `sprintf()` оголошена в файлі заголовку `stdio.h`, а не в `string.h`. Вона працює подібно `printf()`, але здійснює запис в рядок, а не на екран.

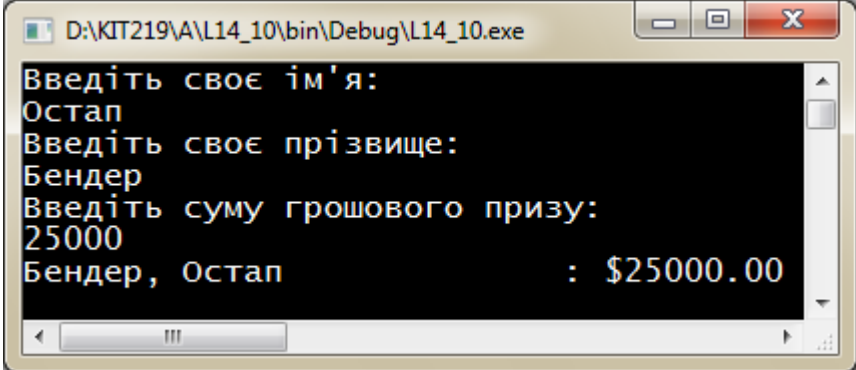
Таким чином, вона надає спосіб об'єднання декількох елементів в єдиний рядок.

Перший аргумент `sprintf()` – це адреса цільового рядка.

Решта аргументів аналогічні аргументам в `printf()` – рядок специфікації перетворення і список елементів, призначених для запису.

Застосування sprintf()

```
#include <stdio.h>
#include <windows.h>
#define MAX 20
char *s_gets(char *st, int n);
int main(void)
{
    char first[MAX];
    char last[MAX];
    char formal[2 * MAX + 10];
    double prize;
    SetConsoleCP(1251);    SetConsoleOutputCP(1251);
    puts("Введіть своє ім'я:");
    s_gets(first, MAX);
    puts("Введіть своє прізвище:");
    s_gets(last, MAX);
    puts("Введіть суму грошового призу:");
    scanf("%lf", &prize);
    sprintf(formal, "%s, %-19s: $%6.2f\n", last, first, prize);
    puts(formal);
    return 0;
}
```



```
D:\KIT219\A\L14_10\bin\Debug\L14_10.exe
Введіть своє ім'я:
Остап
Введіть своє прізвище:
Бендер
Введіть суму грошового призу:
25000
Бендер, Остап : $25000.00
```



Функції `strchr()` і `strpbrk()`

```
char *strchr(const char *s, int c);
```

Ця функція повертає вказівник на першу комірку рядка **s**, в якому міститься символ **c**. Завершальний нульовий символ є частиною рядка, так що його теж можна шукати. Якщо символ не знайдено, функція повертає нульовий вказівник.

```
char *strpbrk(const char *s1, const char *s2);
```

Ця функція повертає вказівник на першу комірку рядка **s1**, в якому міститься будь-який символ, знайдений в рядку **s2**. Функція повертає нульовий вказівник, якщо жодного символу не знайдено.



Функції `strrchr()` і `strstr()`

```
char *strrchr(const char *s, int c);
```

Ця функція повертає вказівник на останнє входження символу **c** в рядку **s**. Завершальний нульовий символ є частиною рядка, тому його також можна шукати. Якщо символ не знайдено, функція повертає нульовий вказівник.

```
char *strstr(const char *s1, const char *s2);
```

Ця функція повертає вказівник на перше входження рядка **s2** всередині рядка **s1**. Якщо рядок не знайдено, функція повертає нульовий вказівник.



Сортування рядків

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

#define SIZE 81
#define LIM 20
#define HALT ""

void stsort(char *strings[], int num);
char *s_gets(char *st, int n);

int main(void)
{
    char input[LIM][SIZE];    // масив для вхідних даних
    char *ptstr[LIM];        // масив вказівників
    int ct = 0;               // лічильник вводу
    int k;                    // лічильник виводу

    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);
```



Сортування рядків

```
printf("Введіть до %d рядків, і вони"
      " будуть відсортовані.\n", LIM);
printf("Щоб зупинити ввід, натисніть"
      " клавішу Enter на початку рядка\n");

while(ct < LIM && s_gets(input[ct], SIZE) != NULL
      && input[ct][0] != '\0')
{
    // встановлюємо вказівники на рядки
    ptstr[ct] = input[ct];
    ct++;
}

// сортувальник рядків
qsort(ptstr, ct);
puts("\nВідсортований список:\n");
for(k = 0; k < ct; k++)
    puts(ptstr[k]);    // відсортовані вказівники
return 0;
}
```



Сортування рядків

```
// функція сортування вказівників на рядки
void stsort(char *strings[], int num)
{
    char *temp;
    int top, seek;

    for(top = 0; top < num - 1; top++)
    {
        for(seek = top + 1; seek < num; seek++)
        {
            if(strcmp(strings[top], strings[seek]) > 0)
            {
                temp = strings[top];
                strings[top] = strings[seek];
                strings[seek] = temp;
            }
        }
    }
}
```

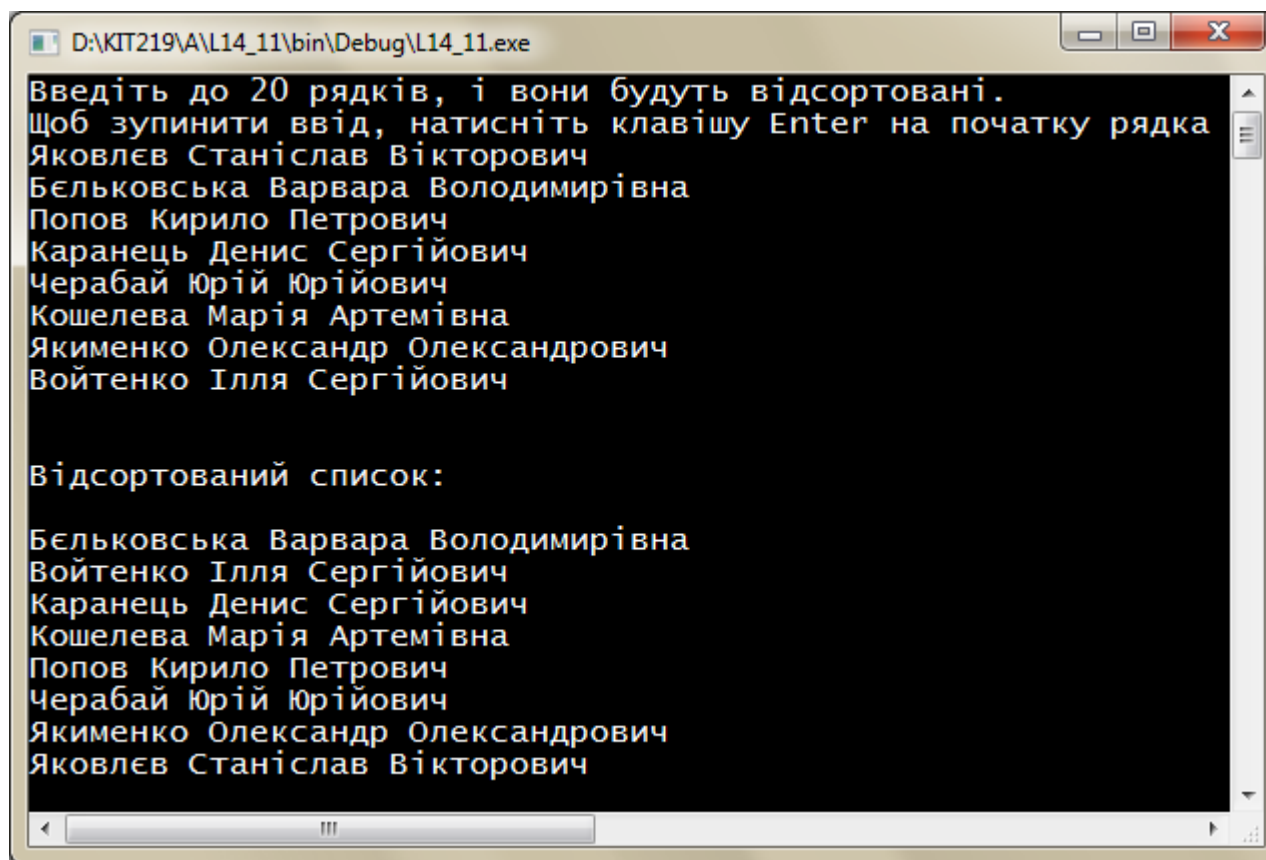


Сортування рядків

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

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

Сортування рядків



```
D:\KT219\A\L14_11\bin\Debug\L14_11.exe

Введіть до 20 рядків, і вони будуть відсортовані.
Щоб зупинити ввід, натисніть клавішу Enter на початку рядка
Яковлев Станіслав Вікторович
Бельковська Варвара Володимирівна
Попов Кирило Петрович
Каранець Денис Сергійович
Черабай Юрій Юрійович
Кошелева Марія Артемівна
Якименко Олександр Олександрович
Войтенко Ілля Сергійович

Відсортований список:

Бельковська Варвара Володимирівна
Войтенко Ілля Сергійович
Каранець Денис Сергійович
Кошелева Марія Артемівна
Попов Кирило Петрович
Черабай Юрій Юрійович
Якименко Олександр Олександрович
Яковлев Станіслав Вікторович
```