**Практична робота №17.** **Написання програм з використанням символьних та рядкових величин**

**Тема: Написання програм з використанням символьних та рядкових величин.**

**Мета:**. Засвоєння навичок розробки програм з використанням символьних та рядкових величин.

**Хід роботи**

1. Прочитати (повторити) основні теоретичні відомості лекції №9 та методологічні вказівки.
2. Запустити середовище програмування С++ .
3. Записати програму, що виконує завдання:

Оголосити масив 10 рядкових змінних (наприклад, char sp[n][l5]), який заповнити ініціалізацією прізвищами та ініціалами осіб (наприклад, char sр [10][15] = {"Петренко В. В.", "Головко С. С.", . . . ,};);

Прізвища розташовані в будь-якому порядку.

Відсортувати прізвища за алфавітом.

1. В першому рядку програми записати

*// Група № Прізвище Номер ЛР*

вказавши номер своєї групи та своє прізвище.

1. Результати надсилати на електронну адресу викладача [**t.i.lumpova@gmail.com**](mailto:t.i.lumpova@gmail.com)у вигляді cpp-файлу з іменем у форматі

**<Номер групи><Номер лабораторної><Прізвище англійською>**

Наприклад, 31-01Ivanov.cpp.

Іншим рішенням є надсилання поштою посилання на текст програми за URL адреси, яку надає C++Shell, вказавши в темі листа, номер групи прізвище студента та номер ПР.

В темі листа вказати, номер групи, прізвище студента та номер ПР як "ПР№17".

**Строк відсилки ПР - 13.04.2024**

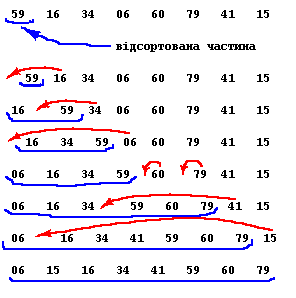
Всі запитання, що виникнуть, надсилайте на електронну адресу викладача, В темі листа вказати, номер групи, прізвище студента та номер ПР як "ПР№17" В темі листа вказати, номер групи, прізвище студента, номер ПР та фразу "Запитання"

**Методологічні вказівки**

**Простий метод сортування - метод прямого включення.**

Метод прямого включення полягає в тому, що один елемент завжди можна вважати відсортованим масивом довжини 1. Розглядаючи послідовно решту елементів масиву будемо включати їх у відсортовану частину масиву, ставлячи на потрібне місце, тобто не порушуючи відсортованості цієї частини. Повторивши операцію включення **n-1** раз, одержимо відсортований масив.

Приклад. Візьмемо кілька випадкових чисел:



На i-му кроці відсортованим вважається масив **a[1] .. a[i]**, а вставляємо елемент **a[i+1]**. Вставляти можна, послідовно порівнюючи **a[i+1]** з **a[1], a[2],...**, поки не знайдеться "дірка" **J**

**a[j] Ј a[i+1] Ј a[j+1]**.

Тоді слід зсунути елементи **a[j+1] .. a[i]** вправо на один елемент, а **a[i+1]** записати на місце **a[j+1]**. Може статися, що **a[i+1]** більше за будь-який з членів відсортованої частини. Тоді дірку не буде знайдено. Тому слід контролювати номер **j**, щоб не вийти за межи відсортованої частини.

Остаточно програма має вигляд

**#include <iostream> // підключення потокового введення-виведення**

**#include <Windows.h>// для зміни кольору консолі**

**using namespace std; // простір імен**

**int main() // головна функція**

**{system("color F0");**

**int n=10,i,j,k,counter;**

**int mas[n]={59,16,34,6,60,79,41,15,5};**

**cout << "Before"<<endl;**

**for(i=0;i<n;i++) cout<<mas[i] <<'\t';**

**cout <<endl;**

**counter=0;**

**cout << "Cycle"<<endl;**

**for(i=1;i<n;i++)**

**{**

**for( j=i; j>0 && mas[j-1]>mas[j];j--){**

**counter++;**

**int tmp=mas[j-1];**

**mas[j-1]=mas[j];**

**mas[j]=tmp;**

**}**

**for(k=0;k<n;k++) cout<<mas[k] <<'\t';**

**cout <<endl;**

**}**

**cout<<"counter="<<counter<<endl;**

**return 0; // повернення результату (вихід із функції)**

**}**

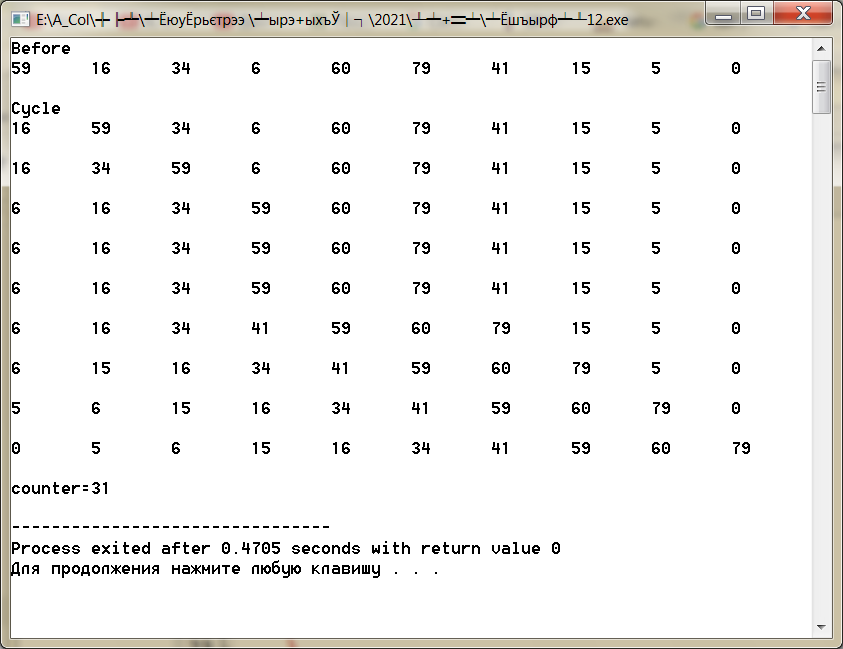
****

Рисунок – результат виконання програми

