

## ЛЕКЦІЯ 6

1. РОБОТА З СИМВОЛЬНИМИ ЗМІННИМИ.
2. МАСИВИ СИМВОЛІВ. РЯДКИ.
3. ФУНКЦІЇ ДЛЯ РОБОТИ З РЯДКАМИ
4. ПРИКЛАДИ ОБРОБКИ РЯДКІВ

## СИМВОЛЬНІ ЗМІННІ

*char* (*character*) – символна змінна. Будь-який символ пов'язаний з цілим числом – кодом цього символу, наприклад, в таблиці ASCII. Символ відтворюється на екрані за його кодом, при введенні з клавіатури символ перетворюється у відповідне значення. Такі перетворення відбуваються автоматично. Усі символи містяться у таблиці, де кожному з них відповідає певний код ASCII. Фрагмент таблиці з кодами ASCII

97	141	0x61	01100001	a
98	142	0x62	01100010	b
99	143	0x63	01100011	c
100	144	0x64	01100100	d
101	145	0x65	01100101	e

Масиви типу *char*, - символні масиви, - займають у мові особливе місце. У мові C робота з рядками реалізована шляхом використання одновимірних масивів типу *char*.

У мові C символний рядок - це одновимірний масив типу *char*, що закінчується *нульовим байтом*. Нульовий байт - це байт, кожен біт якого дорівнює нулю. Для нульового байту визначена спеціальна символна константа `'\0'`. Це варто враховувати при описі відповідного масиву символів. Так, якщо рядок має містити  $n$  символів, то в описі масиву варто вказати  $n+1$  елемент.

Наприклад, опис *char str[11]* припускає, що рядок містить 10 символів, а останній байт зарезервований під нульовий байт. Тобто, якщо необхідно трактувати цей масив як рядок символів, то це буде рядок максимум з 10 елементів.

Мова С допускає рядки-константи. Рядок-константа - це список літер, укладених у подвійні лапки. Наприклад,

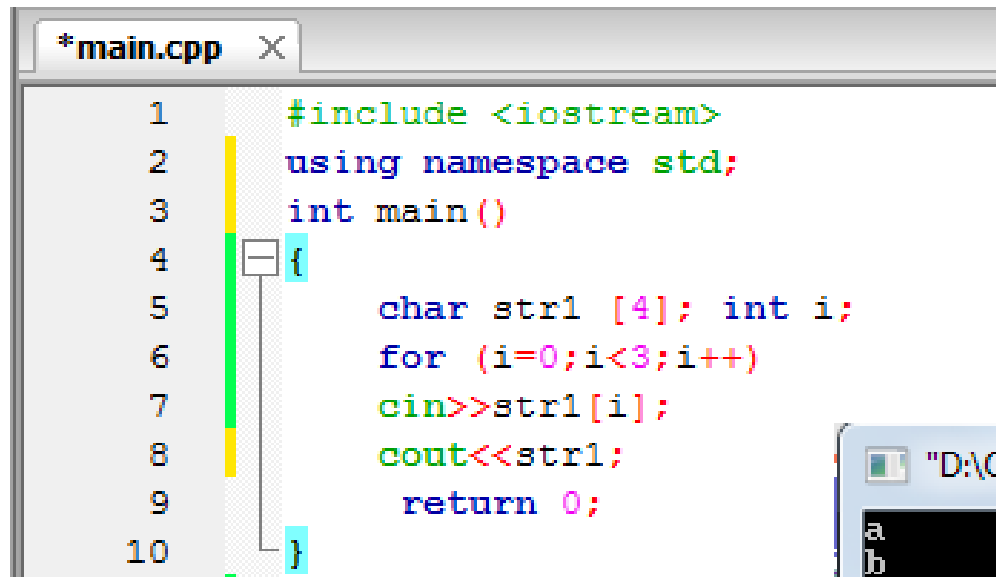
" Це рядок-константа".

У кінець рядку-константи не потрібно ставити символ '\0'. Це буде зроблено при компіляції.

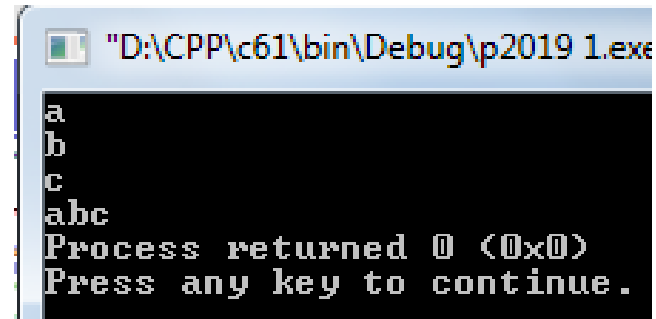
Після створення масиву символів, в якому планується розмістити рядок, робота з ним виконується аналогічно роботі з масивами інших типів.

```
1      #include <iostream>
2
3      using namespace std;
4
5      int main()
6      {
7          char str1 [11]; int i;
8          for (i=0;i<10;i++)
9              cin>>str1[i];
10
11          return 0;
12     }
```

Далі можливо видати на екран повний рядок:



```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5      char str1 [4]; int i;
6      for (i=0;i<3;i++)
7          cin>>str1[i];
8      cout<<str1;
9      return 0;
10 }
```



```
"D:\CPP\c61\bin\Debug\p2019 1.exe"
a
b
c
abc
Process returned 0 (0x0)
Press any key to continue.
```

## ФУНКЦІЇ ДЛЯ РОБОТИ З РЯДКАМИ

Для роботи з рядками існує спеціальна бібліотека, опис якої знаходиться у файлі *string.h*.

Найбільше часто використовуються функції *strcpy()*, *strcat()*, *strlen()*, *strcmp()*.

Виклик функції *strcpy()* має вигляд

```
strcpy(s1, s2);
```

Функція *strcpy()* використовується для копіювання вмісту рядка *s2* у рядок *s1*. Масив *s1* повинний бути досить великим, щоб у нього помістився рядок *s2*. Якщо місця мало, компілятор не видає інформації про помилку; це не припинить виконання програми, але може привести до псування інших даних чи самої програми і неправильній роботі програми надалі.

Виклик функції *strcat()* має вигляд

```
strcat(s1, s2);
```

Функція *strcat()* приєднує рядок *s2* до рядка *s1* і розміщує його в масив, де знаходився рядок *s1*, при цьому рядок *s2* не змінюється. Нульовий байт, що завершував рядок *s1*, буде замінений першим символом рядка *s2*.

Як у функції *strcpy()*, так й у функції *strcat()* рядок, що виходить, автоматично завершується нульовим байтом.

Виклик функції *strcmp()* має вигляд

*strcmp(s1, s2);*

Функція *strcmp()* порівнює рядки *s1* та *s2* й повертає значення 0, якщо рядки рівні, тобто містять те саме число однакових символів.

Під порівнянням рядків розуміють порівняння в лексикографічному сенсі, як це відбувається, наприклад, у словнику. Звичайно, у функції відбувається посимвольне порівняння кодів символів. Код першого символу одного рядка порівнюється з кодом символу другого рядка. Якщо вони рівні, розглядаються другі символи, і т. ін. Якщо *s1* лексиграфічно (у розумінні словника) більше *s2*, то функція *strcmp()* повертає позитивне значення, якщо менше – від'ємне значення.



Виклик функції *strlen()* має вигляд

```
strlen(s);
```

Функція *strlen()* повертає довжину рядка *s*, при цьому завершальний нульовий байт не враховується.

Наприклад, виклик `length("Hello")` поверне значення 5.

Далі наведемо приклад програми, що використовує всі описані функції.

# ПРОГРАМУВАННЯ

```
main.cpp x
1  #include <iostream>
2  #include <string.h>
3  using namespace std;
4  int main()
5  {
6      char str1 [4];
7      strcpy(str1, "abc");
8      cout<<str1<<"\n";
9      /**EXAMPLES*/
10     char str2[4], str3[8];
11     strcpy(str2, str1);
12     cout<<str2<<"\n";
13
14     strcpy(str3, "0");
15     strcat(str3, str1);
16     cout<<str3<<"\n";
17     strcat(str3, str2);
18     cout<<str3<<"\n";
19     cout<<strlen(str3);
20     return 0;
21 }
```

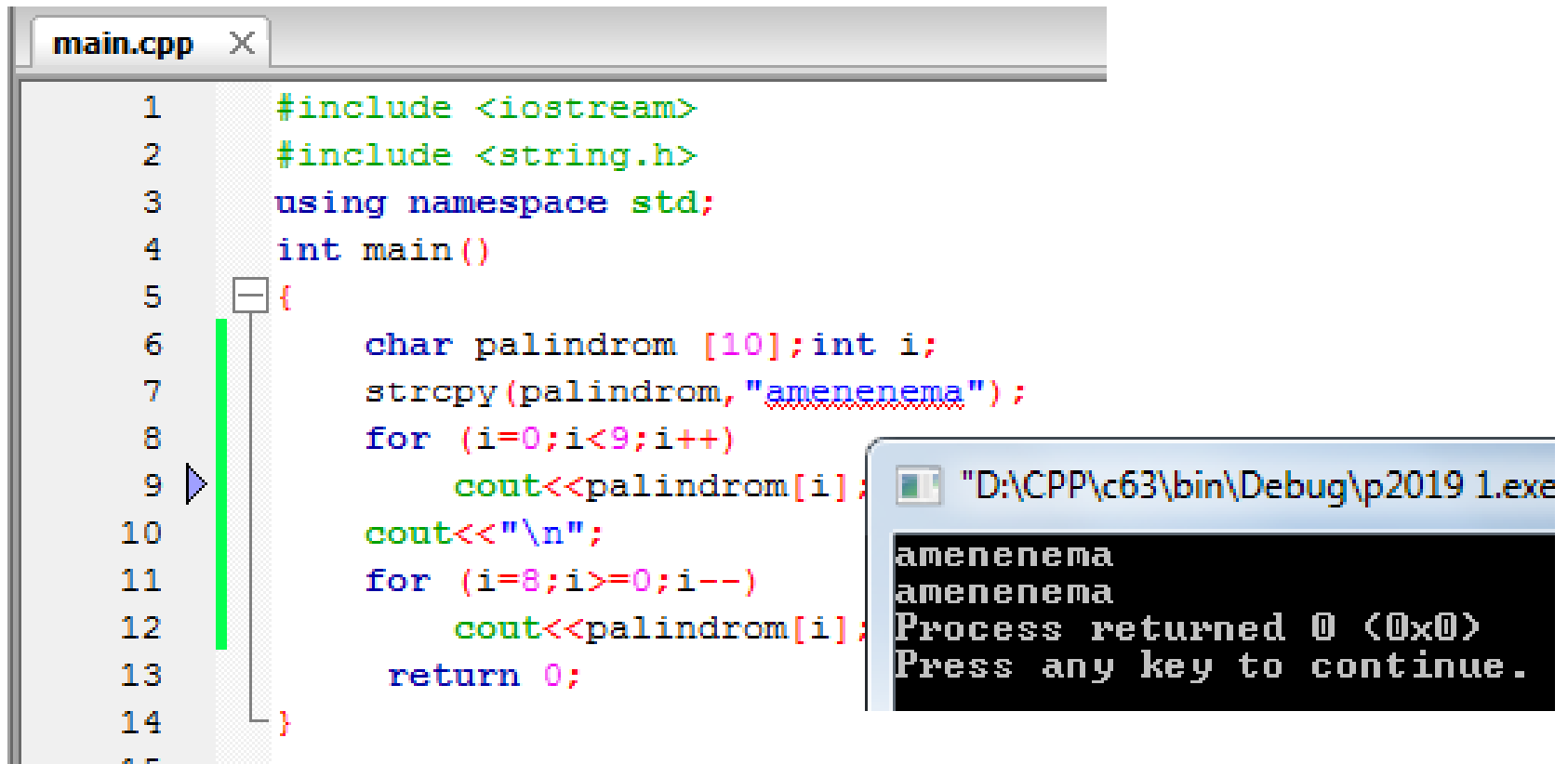
```
"D:\CPP\c61\bin\Debug\p2019 1.exe
abc
abc
0abc
0abcbc
7
Process returned 0 (0x0)
Press any key to continue.
```

## ПРИКЛАД. Створення нових слів

```
main.cpp x
1  #include <iostream>
2  #include <string.h>
3  using namespace std;
4  int main()
5  {
6      char w [7]; int i,j,k;
7      strcpy(w, "POBEDA");
8      k=6;
9      for(i=0;i<6;i++)
10     { j=6-k;
11       while (j<=6)
12       {
13         cout<<w[j];j++;
14       }
15       k--;
16       cout<<"\n";
17     }
18     return 0;
19 }
20
```

```
"D:\CPP\c62\bin\Debug\p2019 1.exe
POBEDA
OBEDA
BEDA
EDA
DA
A
Process returned 0 (0x0)
Press any key to continue.
```

Паліндром – рядок, що читається однаково в обидва боки.



The image shows a C++ IDE window titled 'main.cpp' with a code editor and a console output window. The code defines a character array 'palindrom' with the string 'amenenema' and prints it twice. The first loop prints the string from index 0 to 9, and the second loop prints it from index 8 down to 0. The console output shows the string 'amenenema' printed twice, followed by the message 'Process returned 0 (0x0)' and 'Press any key to continue.'.

```
1  #include <iostream>
2  #include <string.h>
3  using namespace std;
4  int main()
5  {
6      char palindrom [10];int i;
7      strcpy (palindrom, "amenenema");
8      for (i=0;i<9;i++)
9          cout<<palindrom[i];
10     cout<<"\n";
11     for (i=8;i>=0;i--)
12         cout<<palindrom[i];
13     return 0;
14 }
```

"D:\CPP\c63\bin\Debug\p2019 1.exe  
amenenema  
amenenema  
Process returned 0 (0x0)  
Press any key to continue.

Приклад. Визначити частотність входження заданого символу.

```
main.cpp x
1  #include <iostream>
2  #include <string.h>
3  using namespace std;
4  int main()
5  {
6      char text[45];int i, number;
7      strcpy(text,"To be, or not to be, that is the question:");
8      char fsymb='e';number=0;
9      for (i=0;i<43;i++)
10     if (text[i]==fsymb) number++;
11     cout<<"The symbol "<<fsymb<<" appearing in text "<<number<<" times.";
12     return 0;
13 }
```

"D:\CPP\c64\bin\Debug\p2019 1.exe"

```
The symbol e appearing in text 4 times.
Process returned 0 (0x0)   execution time
Press any key to continue.
```

## ДВОВИМІРНІ СИМВОЛЬНІ МАСИВИ.

*char <char array name> [розмір1 ] [розмір2];*

```
1  #include <iostream>
2  #include <string.h>
3  using namespace std;
4  int main()
5  {
6      char text[3][12]; int i,j, number;
7      strcpy(text [0], "First row.");
8      strcpy(text [1], "Second row.");
9      strcpy(text [2], "Third row.");
10
11      return 0;
12  }
```

ДЯКУЮ ЗА УВАГУ!