

# Лекція 8.

## String. C-string.



# План на сьогодні

1

Що таке C-string?

2

Бібліотека C-string

3

Що таке string?

4

Бібліотека string



# Що таке C-string?



# Що таке C-string?

У програмуванні на C, колекція символів зберігається у вигляді масивів. Це також підтримується в програмуванні на C++. Тому їх називають рядками C або C-рядками.

C-рядки — це масиви типу `char`, які закінчуються нульовим символом, тобто `\0` (ASCII значення нульового символу дорівнює 0).

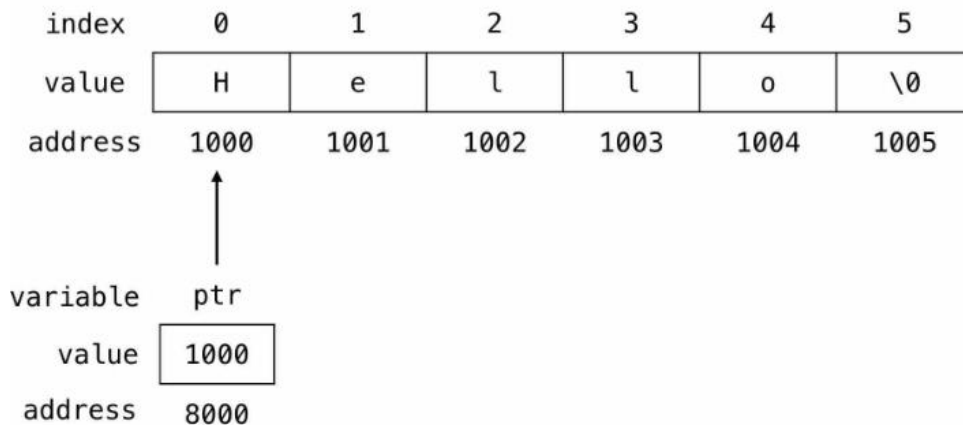
```
char str[] = "C++";  
char str[4] = "C++";  
char str[] = {'C', '+', '+', '\0'};  
char str[4] =  
{ 'C', '+', '+', '\0' };
```

```
const char* str1 = "Hello"; // correct  
char* str2 = "Hello"; // compilation  
error
```

# Розташування C-string в пам'яті

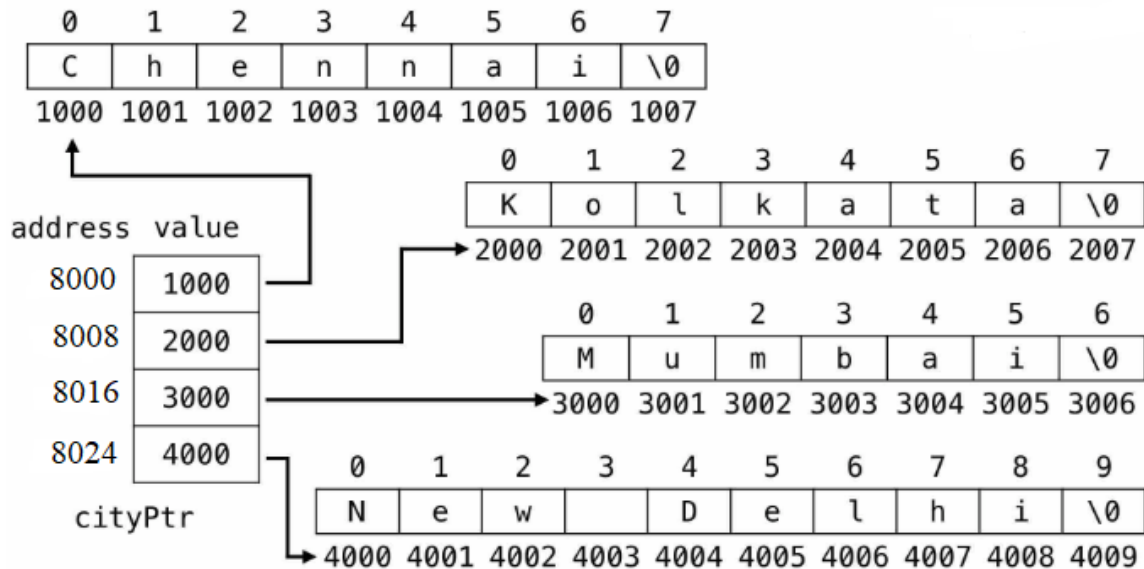
```
// string variable  
char str[ ] = "Hello";
```

```
// pointer variable  
char *ptr = str;
```



# Розташування масиву C-string в пам'яті

```
const char *cityPtr[ ] = {  
    "Chennai",  
    "Kolkata",  
    "Mumbai",  
    "New Delhi"  
};
```



# Зчитування C-string

Зчитувати рядок можна за допомогою:

`cin>>` - зчитує рядок до першого пропуску,

`cin.get()` - зчитує рядок до '\n' і залишає '\n' в потоці зчитування,

`cin.getline()` - зчитує рядок до '\n' і видаляє '\n' з потоку зчитування.

```
char str1[100];
char str2[100];
cout << "Please enter two lines:" << endl;
cin.getline(str1, 100);
cin.get(str2, 100);
cout << "You entered: ";
cout << str1 << " " << str2 << endl;
```

```
Please enter two lines:
1 2
3 4
You entered: 1 2 3 4
Please enter two lines:
1 2
You entered:  1 2
```

```
char str1[100];
char str2[100];
cout << "Please enter two lines:" << endl;
cin.get(str1, 100);
cin.getline(str2, 100);
cout << "You entered: ";
cout << str1 << " " << str2 << endl;
```

```
Please enter two lines:
1 2
You entered:  1 2
```

# Бібліотека C-string

В бібліотеці `<cstring>` є багато корисних функцій для маніпуляції масивами символів

`strcat(char* destination, const char* source)` Додає копію рядка `source` до кінця рядка `destination`.

`strchr(const char* str, int character)` Шукає першу появу символу `character` в рядку `str`.

`strcmp(const char* str1, const char* str2)` Порівнює два рядки `str1` і `str2`. Повертає 0, якщо рядки однакові, або різницю між першими відмінними символами.

`strncmp(const char* str1, const char* str2, size_t num)` Лексикографічно порівнює перші `num` символів двох рядків `str1` і `str2`.

`strcpy(char* destination, const char* source)` Копіює рядок символів з джерела `source` в місце призначення `destination`.

`strncpy(char* destination, const char* source, size_t num)` Копіює до `num` символів з рядка `source` у рядок `destination`.

`strlen(const char* str)` Повертає довжину рядка `str`, не враховуючи нульовий символ.



# strcpy unsafe function

**strcpy** - якщо вхідний рядок довший за розмір буфера призначення це може призвести до небезпечного пошкодження пам'яті, яке важко виявити.

```
int main() {  
    char str[] = "Hello"; // the char array is 6 elements long  
    strcpy(str, "Bad Idea"); // copy 9 chars into a 6-element array - not good!  
    cout << str;  
}
```



C4996

'strcpy': This function or variable may be unsafe. Consider using strcpy\_s instead. To disable deprecation, use \_CRT\_SECURE\_NO\_WARNINGS. See online help for details.

Починаючи з C++11 рекомендовано використовувати [strcpy\\_s](#)

# strncpy unsafe function

**strncpy** - дозволяє уникнути запису занадто великої кількості даних у буфер призначення, АЛЕ не додає нульовий символ, якщо вхідний рядок довший за вказаний розмір:

```
{  
    char str[] = "Hello";  
    strncpy(str, "Bad Idea", sizeof(str)); // no null terminator!  
    cout << str << endl;  
}
```



C4996

'strncpy': This function or variable may be unsafe. Consider using strncpy\_s instead. To disable deprecation, use \_CRT\_SECURE\_NO\_WARNINGS. See online help for details.

```
Bad Id||||||||||||||||||||||||||||||||
```

Починаючи з C++11 рекомендовано використовувати [strncpy\\_s](#)

# strcat/strncat unsafe functions

**strcat** - Додає копію рядка символів, на який вказує `src`, до кінця рядка символів, на який вказує `dest`. Символ `src[0]` замінює нульовий символ у кінці `dest`. Отриманий рядок байтів завершується нульовим значенням. Поведінка не визначена, якщо `dest` масив недостатньо великий для вмісту як `src`, так і `dest` і кінцевого нульового символу.

```
{  
    char dest[50] = "Hello ";  
    const char src[50] = "World!";  
    std::strcat(dest, src);  
    cout << dest << endl;  
    std::strncat(dest, " Goodbye World!", 3);  
    cout << dest << endl;  
}
```

✗ C4996

'strcat': This function or variable may be unsafe. Consider using `strcat_s` instead. To disable deprecation, use `_CRT_SECURE_NO_WARNINGS`. See online help for details.

✗ C4996

'strncat': This function or variable may be unsafe. Consider using `strncat_s` instead. To disable deprecation, use `_CRT_SECURE_NO_WARNINGS`. See online help for details.

Починаючи з C++11 рекомендовано використовувати [strcat\\_s](#)

# Що take string?



FACULTY OF APPLIED  
MATHEMATICS AND  
INFORMATICS  
LVIV UNIVERSITY

# Що string?

У C++ можна створювати об'єкти `string` для зберігання рядків.

На відміну від використання масивів типу `char`, об'єкти `string` не мають фіксованої довжини і можуть бути розширені відповідно до ваших потреб.

Зчитувати рядок можна за допомогою `getline(cin, str);`

# Ініціалізація string

Рядки можна ініціалізувати як змінні типу `string`. Їх можна ініціалізувати відразу або пізніше присвоїти значення.

```
string myString;  
myString = "Hello";
```

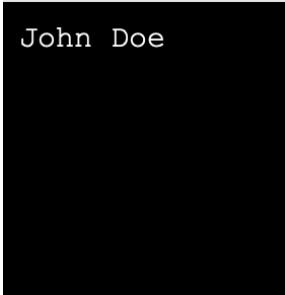
```
string myString("World");  
string myID = "37";
```

# Конкатенація рядків

Оператор `+` можна використовувати між рядками для їх об'єднання в новий рядок. Це називається конкатенацією.

Рядок насправді є об'єктом, який містить функції, що можуть виконувати певні операції з рядками. Наприклад, ви можете об'єднувати рядки також за допомогою функції `append()`.

```
string firstName = "John ";  
string lastName = "Doe";  
string fullName = firstName +  
lastName;  
cout << fullName;
```



John Doe

```
string firstName = "John ";  
string lastName = "Doe";  
string fullName =  
firstName.append(lastName);  
cout << fullName;
```

# Довжина рядків

Щоб отримати довжину рядка, використовуйте функцію `length()`.

```
string txt = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";  
cout << "The length of the txt string is: " << txt.length();
```



```
The length of the txt string is: 26
```

```
string txt = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";  
cout << "The length of the txt string is: " << txt.size();
```



# Доступ до рядків

Можна отримати доступ до символів у рядку, звертаючись до його індексу в квадратних дужках [].

`.at()` - отримує доступ до вказаного символу з перевіркою виходу за межі рядка

```
string myString = "Hello";  
cout << myString[0];  
// Outputs H
```

```
string myString = "Hello";  
cout <<  
myString[myString.length() - 1];  
// Outputs o
```

```
string myString = "Hello";  
cout << myString[1];  
// Outputs e
```

```
string myString = "Hello";  
myString[0] = 'J';  
cout << myString;  
// Outputs Jello instead of Hello
```

```
string myString = "Hello";  
cout << myString; // Outputs Hello  
cout << myString.at(0); // First character  
cout << myString.at(myString.length() - 1);  
// Last character
```

```
myString.at(0) = 'J';  
cout << myString; // Outputs Jello
```

# Вставка рядка в задану позицію

Можна вставити рядок у будь-яку позицію за допомогою функції `insert()`.

```
string myString = "Hello world!";  
cout << "String: " << myString;  
myString.insert(6, "cruel ");
```

# Видалення частини рядка

Можна видалити кілька символів із заданої позиції за допомогою функції `erase()`.

У цьому прикладі функція `erase()` видаляє 6 символів, починаючи з індексу 5 у рядку `myString`.

```
string myString = "Hello world!";  
cout << "String: " << myString;  
myString.erase(5, 6);
```

# Спеціальні символи

Зворотна скісна риска (\) перетворює спеціальні символи на символи рядка.

Послідовність \" вставляє подвійні лапки у рядок

Послідовність \' вставляє одинарну лапку у рядок

Послідовність \\ вставляє одну зворотну скісну риску у рядок

```
string txt = "We are the so-called \"Vikings\" from the north.";
```

```
string txt = "It\'s alright.";
```

```
string txt = "The character \\ is called backslash.";
```

# Рядки типу `std::string` та рядки в С-стилі

```
char myntcs[] = "some text";  
string mystring = myntcs; // конвертація c-рядок до string  
cout << mystring; // друкуєм як string  
cout << mystring.c_str(); // друкуєм як c-рядок
```

# Дякую!