



***СИМВОЛЬНИЙ ТИП. РЯДКИ СИМВОЛІВ.
ФУНКЦІЇ ОПРАЦЮВАННЯ ЛІТЕРНИХ РЯДКІВ***

Символьний тип

Типом символьних змінних у C++ є тип **char**.

Значення символьних констант записують в одиничних лапках.

Коди символів

Для уникнення розходжень при кодуванні символів розроблено єдиний міжнародний стандарт кодів для обміну інформацією – *таблиця ASCII-кодів*

Символ	Код
' ' (пробіл)	32
'0'...'9'	48...57
'A'...'Z'	65...90
'a'...'z'	97...122

Основні ескейп-послідовності

\n – переведення курсору на початок наступного рядка

\r – перехід на початок поточного рядка

\t – горизонтальна табуляція

\v – вертикальна табуляція

\b – вилучення попереднього символу

\0 – завершальний символ рядка символів

Ескейп-послідовність – це спеціальна комбінація символів, починається зі зворотної косої риски і записується в одиничних лапках.

Коли символні змінні отримують певні значення, то комп'ютер зберігає в пам'яті не самі символи, а їхні коди.

якщо символній змінній присвоїти певне ціле число,
то компілятор сприйме його як код символу з ASCII.
правило поширюється *лише на цілі числа*.

```
char c, d = 'Y', r = 115;  
// виведення символу, що зберігається в змінній:  
cout << "Output char variable d: " << d << endl; // Результат: Y  
// виведення коду символу, що зберігається в змінній:  
cout << "\nOutput code variable d: " << (int)d << endl; // Результат: 89  
//виведення символу з кодом 115:  
cout << "\n Output code variable r: " << r << endl; // Результат: s  
// введення символу за допомогою cin:  
cout << "\nEnter char: "; cin >> c;  
int k = c;  
//виведення коду символу, введенного з клавіатури:  
cout << "Code of entered char: " << k;
```

При *порівнянні* символів більшим вважається той, код якого більший, тобто той символ, який у таблиці ASCII-кодів розміщений пізніше. Наприклад, справедливі такі нерівності: '5'<'A', 'B'<'C', 'C'<'c', 'c'<'z'.

Оскільки символний тип у C++ вважається цілим типом,
то змінні цього типу можна *додавати та віднімати*.

Функції для роботи з символами



Функція	Призначення
<code>tolower()</code>	повертає символ у нижньому регістрі
<code>toupper()</code>	повертає символ у верхньому регістрі
<code>isalnum()</code>	перевіряє чи належить символ множині латинських літер та цифр (A...Z, a...z, 0...9)
<code>isalpha()</code>	Перевіряє чи належить символ множині латинських літер (A...Z, a...z)
<code>isctrl()</code>	перевіряє чи належить символ множині керуючих символів (з кодами 0...31 та 127)
<code>isdigit()</code>	перевіряє чи належить символ множині цифр (0...9)
<code>isxdigit()</code>	перевіряє чи належить символ множині шістнадцяткових цифр (A...F, a...f, 0...9)
<code>isgraph()</code>	перевіряє чи належить символ множині друкованих символів, крім пробіла (<code>isalpha()</code> , <code>isdigit()</code> , <code>ispunct()</code>)
<code>islower()</code>	перевіряє чи належить символ множині латинських літер нижнього регістра (a...z)
<code>isupper()</code>	перевіряє чи належить символ множині латинських літер верхнього регістра (A...Z)

зміни великої латинської літери

на малу

```
char c = c + 32;
```

або

```
char c = tolower(c);
```

Рядки символів

На відміну від інших мов програмування, в мові C немає типу даних стрічка чи рядок. Тут рядок символів є послідовністю символів (масивом символів), який закінчується нуль-символом, тобто символом з кодом 0, що записується як керуюча послідовність '\0' і сприймається як один символ. Нуль-символ ще називають **нуль-термінальним** або **термінальним** (завершальним) символом.

Крім нуль-термінальних літерних рядків у стилі мови C, реалізованих на базі символьних масивів `char[]`, мова C++ дає змогу використовувати літерні рядки, реалізовані класом `string`.

```
#include <string>

string s1("world"); // оголошує рядок s1 та ініціалізує його
string s2(10, 'b'); // оголошує рядок s2 та ініціалізує його
string s3(s1); // оголошує рядок s3 та ініціалізує його значенням s1
s1 += " hello"; // до рядка s1 додаємо C-рядок "hello"
string s4; // оголошує порожній рядок s4
char s[] = "word"; // оголошує C-рядок s та ініціалізує його л
s4 = 'D'; // рядку s4 присвоює символ 'D'
string s7(s + 6, s + 10); // оголошує рядок s7 та
```


Введення літерних рядків *string*

```
string str;  
cout << "Input text" << endl;  
//cin >> str;  
getline(cin, str);  
cout << str;
```

Операцію `+` можна використовувати для конкатенації одного літерного рядка `string` з іншим, або літерного рядка `string` із рядком в C-стилі.

Операції над літерними рядками *string*

- = – присвоєння;
- + – конкатенація;
- += – конкатенація та присвоєння.
- == – порівняння на рівність;
- != – порівняння на нерівність;
- < – порівняння, чи менше;
- > – порівняння, чи більше;
- <= – порівняння, чи менше або дорівнює;
- >= – порівняння, чи більше або дорівнює

Доступ до окремого символу рядка здійснюється за його *індексом*.
Індексація починається з **нуля**

Функції для роботи з літерними рядками *string*

Функції опрацювання літерних рядків *string* зазвичай викликаються так:

```
string s;
```

```
s.<функція>(<аргументи>);
```

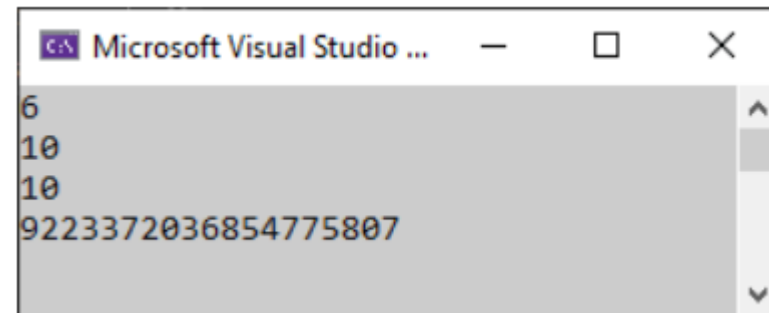
або

```
<змінна> = s.<функція>(<аргументи>);
```

`length()` та `size()` – повертають довжину (кількість символів) літерного рядка.

```
#include <iostream>
using namespace std;
int main()
{
    //масив символів
    char s[10] = "school";
    //виведення довжини s
    cout << strlen(s) << endl; //6
    //змінна рядкова
    string str = "university";
    //виведення довжини str
    cout << str.length() << endl; //10
    cout << str.size() << endl; //10

    return 0;
}
```



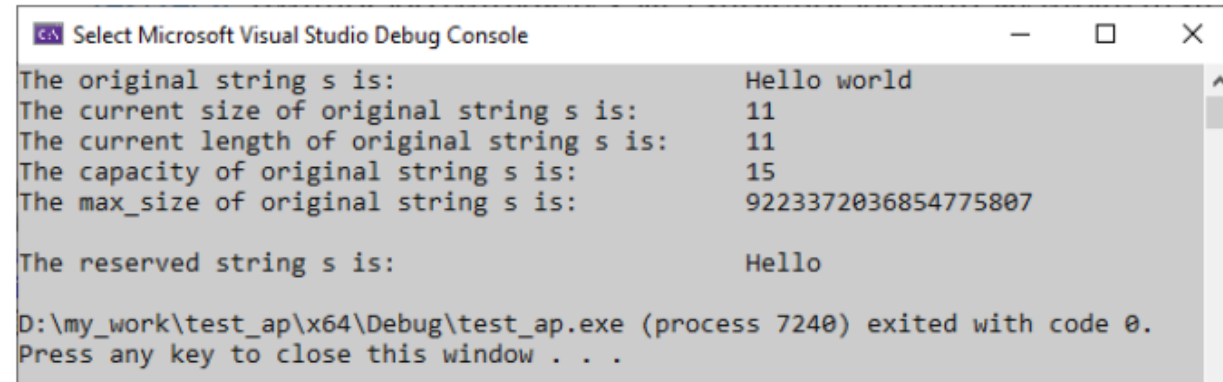
```
Microsoft Visual Studio ...
6
10
10
9223372036854775807
```

`capacity()` – повертає поточну ємкість рядка – кількість символів, які рядок може містити до того, коли йому доведеться дати запит на виділення додаткової пам'яті.

`max_size()` – повертає максимальну кількість символів, яку може містити рядок.

`resize()` – змінює розмір рядка: встановлює розмір, який визначається параметром `num`

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
    string s("Hello world");
    cout << "The original string s is: \t\t\t" << s << endl;
    // Порівнює функції size(), length(), capacity() та max_size()
    cout << "The current size of original string s is: \t"
        << s.size() << endl;
    cout << "The current length of original string s is: \t"
        << s.length() << endl;
    cout << "The capacity of original string s is: \t\t"
        << s.capacity() << endl;
    cout << "The max_size of original string s is: \t\t"
        << s.max_size() << endl << endl;
    s.erase(6, 5);
    // видалення частини рядка
    s.reserve(30);
    // після встановлення нового значення для ємкості рядка
    cout << "The reserved string s is: \t\t\t" << s << endl;
    return 0;
}
```



```
Select Microsoft Visual Studio Debug Console

The original string s is:      Hello world
The current size of original string s is:      11
The current length of original string s is:     11
The capacity of original string s is:          15
The max_size of original string s is:          9223372036854775807

The reserved string s is:      Hello

D:\my_work\test_ap\x64\Debug\test_ap.exe (process 7240) exited with code 0.
Press any key to close this window . . .
```


`find()` – повертає позицію першого входження підрядка `str` у цьому рядку. Пошук починається з позиції `pos`. Якщо входження не виявлено, повертає значення `-1`

`rfind()` – повертає позицію останнього входження підрядка `str` у цьому рядку. починається з позиції `pos`. Якщо входження не виявлено, повертає значення `-1`

`find_first_of()` – повертає позицію першого символу цього рядка, який збігається з будь-яким символом рядка `str`. Пошук починається з позиції `pos`. Якщо входження не виявлено, повертає значення `-1`

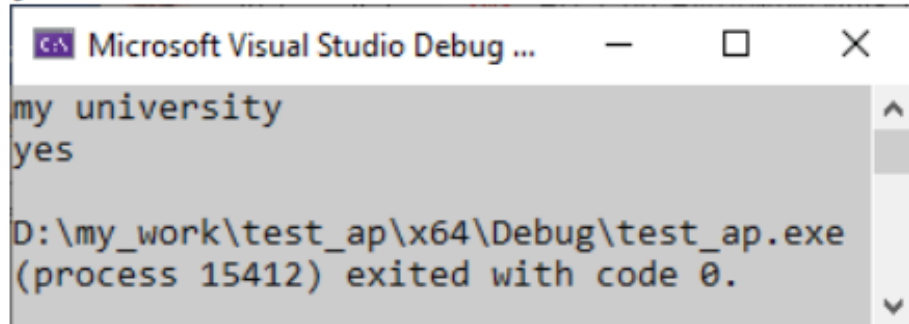
`find_last_of()` – повертає позицію останнього символу цього рядка, який збігається з будь-яким символом рядка `str`. Пошук починається з позиції `pos`. Якщо входження не виявлено, повертає значення `-1`.

`find_first_not_of` – повертає позицію першого символу цього рядка, який не збігається ні з яким символом рядка `str`. Пошук починається з позиції `pos`. Якщо входження не виявлено, повертає значення `-1`

`find_last_not_of()` – повертає позицію останнього символу цього рядка, який не збігається ні з яким символом рядка `str`. Пошук починається з позиції `pos`. Якщо входження не виявлено, повертає значення `-1`.

```
#include <iostream>
using namespace std;
int main()
{
    string str = "university";
    string s = "my ";
    //вставка підрядка
    cout << s.insert(3, str) << endl; // "my university"
    char ch = 't';
    // визначення чи входить символ у рядок
    if (str.find(ch) != -1)
        cout << "yes" << endl;
    else
        cout << "no" << endl;

    return 0;
}
```



```
Microsoft Visual Studio Debug ...
my university
yes

D:\my_work\test_ap\x64\Debug\test_ap.exe
(process 15412) exited with code 0.
```

```
#include <iostream>
#include <string>
using namespace std;
//визначення кількості входжень символу у рядок
int Count(const string s, const char ch);
int main()
{
    string s = "university";
    char ch = 'i';

    cout << Count(s, ch) << endl; //2
    return 0;
}
int Count(const string s, const char ch)
{
    size_t pos = 0;
    int k = 0;
    while ((pos = s.find(ch, pos)) != -1)
    {
        k++;
        pos++;
    }
    return k;
}
```

Порівняння

`compare()` – порівнює рядок `str` з деяким вказаним рядком. Повертає значення: менше нуля, якщо вказаний рядок менший; нуль, якщо рядки рівні; більше нуля, якщо вказаний рядок є більшим.

Додавання

`append()` – добавляє вказаний рядок (або окремий символ) у кінець потрібного рядка. Повертає модифікований рядок.

`string& append(const string& str);` – добавляє рядок `str` у кінець цього рядка.

`string& append(const string& str, size_t pos, size_t len);` – добавляє фрагмент рядка `str` у кінець цього рядка. Фрагмент рядка, що добавляється, починається з позиції `pos` та має довжину `len` символів.

Вставка

`insert()` – вставляє деякий підрядок у вказану позицію цього рядка. Повертає модифікований рядок.

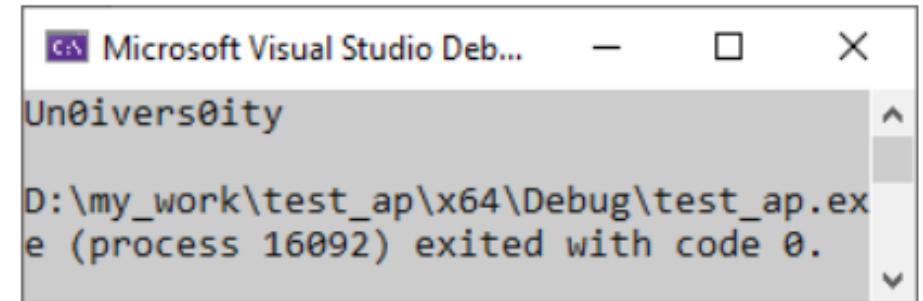
`string& insert(size_t pos, const string& str);` – вставляє рядок `str` у позицію `pos` цього рядка.

`string& insert(size_t pos1, const string& str, size_t pos2, size_t len);` – вставляє підрядок рядка `str` у позицію `pos1` цього рядка. Підрядок рядка `str` починається з позиції `pos2`, і містить `len` символів.

```
#include <iostream>
#include <string>
using namespace std;
//функція для вставки символу
string InsertChar(string& s, const char c1, const char c2);
```

```
int main()
{
    string str = "University";
    char c1 = 'i';
    char c2 = '0';
    cout << InsertChar(str, c1, c2) << endl;
    return 0;
}
```

```
string InsertChar(string& s, const char c1, const char c2)
{
    size_t pos = 0;
    while ((pos = s.find(c1, pos)) != -1)
    {
        s.insert(pos, 1, c2);
        pos += 2;
    }
    return s;
}
```



Видалення

`erase()` – починаючи з деякої позиції, видаляє вказану кількість символів з цього рядка. Повертає модифікований рядок.

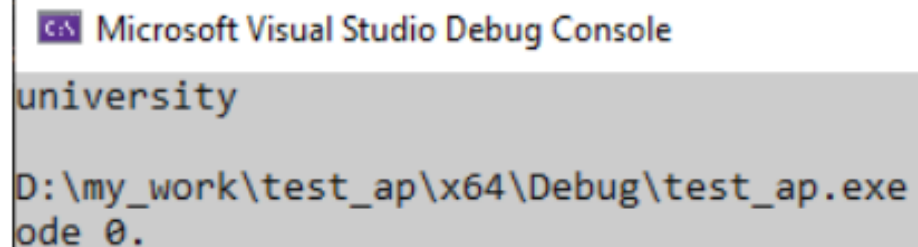
`string& erase(size_t pos = 0, size_t len = -1);` – починаючи з позиції `pos`, видаляє `len` символів з цього рядка.

```
#include <iostream>
#include <string>
using namespace std;
//функція для видалення вказаного символу з рядка
string Delete(string& s, const char ch);

int main()
{
    string str = "university!";
    char ch = '!';

    cout << Delete(str, ch) << endl;
    return 0;
}

string Delete(string& s, const char ch)
{
    size_t pos;
    while ((pos = s.find(ch)) != -1)
        s = s.erase(pos, 1);
    return s;
}
```



```
C:\> Microsoft Visual Studio Debug Console

university

D:\my_work\test_ap\x64\Debug\test_ap.exe
ode 0.
```


Заміна

`replace()` – замінює символи у цьому рядку, починаючи з вказаної позиції, деяким підрядком. Повертає модифікований рядок

`string& replace(size_t pos, size_t len, const string& str);` – замінює до `len` символів у цьому рядку, починаючи з позиції `pos`, рядком `str`.

`string& replace(size_t pos1, size_t len1, const string& str, size_t pos2, size_t len2);` – замінює до `len1` символів у цьому рядку, починаючи з позиції `pos1`, символами (в кількості `len2`) рядка `str`, починаючи з позиції `pos2`.

Копіювання

`copy()` – починаючи з вказаної в якості першого аргумента позиції, копіює символи із цього рядка в символьний масив, що адресується параметром. Повертає кількість скопійованих символів.

`size_t copy(char* str, size_t len, size_t pos = 0) const;` – починаючи з позиції `pos`, копіює `len` символів із цього рядка в символьний масив, що адресується параметром `str`.

Виділення частини

`substr()` – повертає частину рядка згідно вказаних параметрів (починаючи з деякого символу і певної довжини)

`string substr(size_t pos = 0, size_t len = -1) const;` – повертає підрядок цього рядка, що складається із `len` символів, починаючи з позиції `pos`.

Доступ до символу у вказаній позиції

`at()` – повертає символ, що міститься у цьому рядку у вказаній позиції

`char& at(size_t pos);` – повертає символ, що міститься у цьому рядку в позиції `pos`.

Наступні вирази – еквівалентні: `s.at(i)` та `s[i]`

Перевірка, чи рядок – порожній

`empty()` – визначає чи містить вказаний рядок символи.

`bool empty() const;` – повертає `true`, якщо цей рядок – порожній, та `false` у іншому випадку.

```
if (empty(str))
    cout << "Your text is empty" << endl;
else
    cout << str;
```

Визначення кількості входжень групи заданих символів (підрядка) до літерного рядка

Використання функцій для string

```
#include <iostream>
#include <string>

using namespace std;

int main()
{
    string s  = "school";
    string cs = "ch";

    if ( s.find(cs) != -1 )
        cout << "yes" << endl;
    else
        cout << "no" << endl;

    return 0;
}
```

Використання функцій для char []

```
#include <iostream>

using namespace std;

int main()
{
    char s[10] = "school";
    char cs[]  = "oh";

    if ( strstr(s, cs) )
        cout << "yes" << endl;
    else
        cout << "no" << endl;

    return 0;
}
```

Визначення, чи входять всі символи заданого набору до літерного рядка



Використання функцій для рядків string

```
#include <iostream>
#include <string>

using namespace std;

bool Include(const string s, const string cs)
{
    bool r = true;
    for (size_t i=0; i < cs.length(); i++)
        r = r && ( s.find(cs[i]) != -1 );

    return r;
}

int main()
{
    string s = "school";
    string cs = "sh";

    if ( Include(s, cs) )
        cout << "yes" << endl;
    else
        cout << "no" << endl;

    return 0;
}
```

Використання функцій для рядків char []

```
#include <iostream>

using namespace std;

bool Include(const char* s, const char* cs)
{
    bool r = true;
    for (int i=0; cs[i] != '\0'; i++)
        r = r && ( strchr(s, cs[i]) );

    return r;
}

int main()
{
    char s[10] = "school";
    char cs[] = "hs";

    if ( Include(s, cs) )
        cout << "yes" << endl;
    else
        cout << "no" << endl;

    return 0;
}
```

Вилучення символів заданого набору із літерного рядка



Використання функцій для рядків string

```
#include <iostream>
#include <string>

using namespace std;

string Delete(string& s, const string cs)
{
    size_t pos;

    while ( (pos = s.find_first_of(cs)) != -1 )
        s = s.erase(pos, 1);

    return s;
}

int main()
{
    string s = "2sc3ho32o13";
    string cs = "23";

    cout << Delete(s, cs) << endl;

    return 0;
}
```

Використання функцій для рядків char []

```
#define _CRT_SECURE_NO_WARNINGS
#include <iostream>

using namespace std;

char* Delete(char* s, const char* cs)
{
    char* t = new char[strlen(s)];
    char* p;

    int pos1 = 0,
        pos2 = 0;
    *t = 0;

    while ( p = strpbrk(s+pos1, cs) )
    {
        pos2 = p-s+1;
        strncat(t, s+pos1, pos2-pos1-1);
        pos1 = pos2;
    }
    strcat(t, s+pos1);

    strcpy(s, t);

    return t;
}

int main()
{
    char s[20] = "2sc3ho32o13";
    char cs[] = "23";

    cout << Delete(s, cs) << endl;

    return 0;
}
```


Дякую за увагу

Лектор:
кандидат фіз.-мат. наук, доцент
Шаклеїна Ірина
iryna.o.shakleina@lpnu.ua
кафедра ІСМ, ІКНІ