

# Обектно ориентирано програмиране

---

КЛАС `STD::STRING`

# std::string

---

<http://www.cplusplus.com/reference/string/string/>

## Конструктори

default (1)	<code>string();</code>
copy (2)	<code>string (const string&amp; str);</code>
substring (3)	<code>string (const string&amp; str, size_t pos, size_t len = npos);</code>
from c-string (4)	<code>string (const char* s);</code>
from sequence (5)	<code>string (const char* s, size_t n);</code>
fill (6)	<code>string (size_t n, char c);</code>

# std::string

---

## (1) empty string constructor (default constructor)

Constructs an empty string, with a length of zero characters.

## (2) copy constructor

Constructs a copy of str.

## (3) substring constructor

Copies the portion of str that begins at the character position pos and spans len characters (or until the end of str, if either str is too short or if len is string::npos).

## (4) from c-string

Copies the null-terminated character sequence (C-string) pointed by s.

## (5) from buffer

Copies the first n characters from the array of characters pointed by s.

## (6) fill constructor

Fills the string with n consecutive copies of character c.

# std::string

---

```
int main() {  
  
    std::string s0("Initial string");  
  
    // constructors used in the same order as described above:  
  
    std::string s1;  
  
    std::string s2(s0);  
  
    std::string s3(s0, 8, 3);  
  
    std::string s4("A character sequence", 6);  
  
    std::string s5("Another character sequence");  
  
    std::string s6a(10, 'x');  
  
    std::string s6b(10, 42);    // 42 is the ASCII code for '*'  
  
    std::string s7(s0.begin(), s0.begin() + 7);  
  
    std::cout << "s1: " << s1 << "\ns2: " << s2 << "\ns3: " << s3;  
  
    std::cout << "\ns4: " << s4 << "\ns5: " << s5 << "\ns6a: " << s6a;  
  
    std::cout << "\ns6b: " << s6b << "\ns7: " << s7 << '\n';  
  
    return 0;  
}
```

# std::string

---

## Output

s1:

s2: Initial string

s3: str

s4: A char

s5: Another character sequence

s6a: xxxxxxxxxxxx

s6b: \*\*\*\*\*

s7: Initial

# std::string

## Iterators:

---

### **begin**

Return iterator to beginning (public member function )

### **end**

Return iterator to end (public member function )

### **rbegin**

Return reverse iterator to reverse beginning (public member function )

### **rend**

Return reverse iterator to reverse end (public member function )

### **cbegin**

Return const\_iterator to beginning (public member function )

### **cend**

Return const\_iterator to end (public member function )

### **crbegin**

Return const\_reverse\_iterator to reverse beginning (public member function )

### **crend**

Return const\_reverse\_iterator to reverse end (public member function )

# std::string

---

```
int main()
{
    std::string str("Test string");

    for (std::string::iterator it = str.begin(); it !=
str.end(); ++it)
        std::cout << *it;

    std::cout << '\n';

    return 0;
}
```

# std::string

Capacity:

---

## **size**

Return length of string (public member function )

## **length**

Return length of string (public member function )

## **max\_size**

Return maximum size of string (public member function )

## **resize**

Resize string (public member function )

## **capacity**

Return size of allocated storage (public member function )

## **reserve**

Request a change in capacity (public member function )

## **clear**

Clear string (public member function )

## **empty**

Test if string is empty (public member function )



# std::string

---

```
// string::size
#include <iostream>
#include <string>

int main()
{
    std::string str("Test string");
    std::cout << "The size of str is " << str.size() << " bytes.\n";
    return 0;
}
```

# std::string

---

## Element access:

### **operator[]**

Get character of string (public member function )

### **at**

Get character in string (public member function )

### **back**

Access last character (public member function )

### **front**

Access first character (public member function )

# std::string

---

```
// string::operator[]  
  
#include <iostream>  
#include <string>  
  
int main()  
{  
    std::string str("Test string");  
    for (int i = 0; i<str.length(); ++i)  
    {  
        std::cout << str[i];  
    }  
    return 0;  
}
```

# std::string

Modifiers:

---

## **operator+=**

Append to string (public member function )

## **append**

Append to string (public member function )

## **push\_back**

Append character to string (public member function )

## **assign**

Assign content to string (public member function )

## **insert**

Insert into string (public member function )

## **erase**

Erase characters from string (public member function )

## **replace**

Replace portion of string (public member function )

## **swap**

Swap string values (public member function )

## **pop\_back**

Delete last character (public member function )

# std::string

---

```
// string::operator+=
#include <iostream>
#include <string>

int main()
{
    std::string name("John");
    std::string family("Smith");
    name += " K. ";           // c-string
    name += family;           // string
    name += '\n';             // character

    std::cout << name;
    return 0;
}
```

# std::string

---

```
// appending to string
#include <iostream>
#include <string>
int main()
{
    std::string str;
    std::string str2 = "Writing ";
    std::string str3 = "print 10 and then 5 more";

    // used in the same order as described above:
    str.append(str2);                // "Writing "
    str.append(str3, 6, 3);          // "10 "
    str.append("dots are cool", 5);  // "dots "
    str.append("here: ");            // "here: "
    str.append(10u, '.');            // "....."
    str.append(str3.begin() + 8, str3.end()); // " and then 5 more"
    str.append<int>(5, 0x2E);        // "....."

    std::cout << str << '\n';
    return 0;
}
```

# std::string

```
// replacing in a string
#include <iostream>
#include <string>

int main()
{
    std::string base = "this is a test string.";
    std::string str2 = "n example";
    std::string str3 = "sample phrase";
    std::string str4 = "useful.";

    // replace signatures used in the same order as described above:

    // Using positions:
    std::string str = base;
    str.replace(9, 5, str2);
    str.replace(19, 6, str3, 7, 6);
    str.replace(8, 10, "just a");
    str.replace(8, 6, "a shorty", 7);
    str.replace(22, 1, 3, '!');

    0123456789*123456789*12345
    // "this is a test string."
    // "this is an example string." (1)
    // "this is an example phrase." (2)
    // "this is just a phrase." (3)
    // "this is a short phrase." (4)
    // "this is a short phrase!!!" (5)
}
```

# std::string

---

```
// Using iterators:                                0123456789*123456789*

    str.replace(str.begin(), str.end() - 3, str3);    // "sample phrase!!!"
(1)

    str.replace(str.begin(), str.begin() + 6, "replace"); // "replace phrase!!!"
(3)

    str.replace(str.begin() + 8, str.begin() + 14, "is coolness", 7); // "replace is
cool!!!" (4)

    str.replace(str.begin() + 12, str.end() - 4, 4, 'o'); // "replace is coool!!!"
(5)

    str.replace(str.begin() + 11, str.end(), str4.begin(), str4.end()); // "replace is
useful." (6)

    std::cout << str << '\n';

    return 0;

}
```



# std::string

## String operations:

---

### **c\_str**

Get C string equivalent (public member function )

### **copy**

Copy sequence of characters from string (public member function )

### **find**

Find content in string (public member function )

### **rfind**

Find last occurrence of content in string (public member function )

### **find\_first\_of**

Find character in string (public member function )

### **find\_last\_of**

Find character in string from the end (public member function )

### **substr**

Generate substring (public member function )

### **compare**

Compare strings (public member function )

```
// string::substr
```

```
#include <iostream>
```

```
#include <string>
```

---

```
int main()
```

```
{
```

```
    std::string str = "We think in generalities, but we live in details.";
```

```
    // (quoting Alfred N. Whitehead)
```

```
    std::string str2 = str.substr(3, 5);    // "think"
```

```
    std::size_t pos = str.find("live");    // position of "live" in str
```

```
    std::string str3 = str.substr(pos);    // get from "live" to the end
```

```
    std::cout << str2 << ' ' << str3 << '\n';
```

```
    return 0;
```

# std::string

---

## **Non-member function overloads**

### **operator+**

Concatenate strings (function )

### **relational operators**

Relational operators for string (function )

### **swap**

Exchanges the values of two strings (function )

### **operator>>**

Extract string from stream (function )

### **operator<<**

Insert string into stream (function )

### **getline**

Get line from stream into string (function )