

Třída pro uložení řetězců - string

Třída *string* je pro uložení řetězců. Řetězce jsou v objektech této třídy uloženy podle použitého způsobu kódování (ASCII nebo Unicode).

Konstruktory

<code>string()</code>	Prázdný řetězec.
<code>string(const string &str)</code>	Řetězec je kopií řetězce, jenž je uložen v objektu, který je argumentem konstruktoru.
<code>string(const string &str, size_t pos, size_t n=npos)</code> Řetězec, který je podřetězcem řetězce uloženém v objektu, jenž je argumentem konstruktoru. Podřetězec začíná na pozici <i>pos</i> a obsahuje nejvýše <i>n</i> znaků. Implicitní hodnota <i>npos</i> je maximální délka, kterou řetězec může mít (vysoká hodnota).	
<code>string(const char *s, size_t n)</code>	Řetězec, který je kopie řetězce <i>s</i> , přičemž je z něho vzato nejvýše <i>n</i> znaků.
<code>string(const char *s)</code>	Řetězec, který je kopie řetězce <i>s</i> .
<code>string(size_t n, char c)</code>	Řetězec vytvořený <i>n</i> × opakováním znaku <i>c</i> .

Příklad.

```
#include <string>
using namespace std;

string j("Jazyk C++ je rozšířením jazyka C"),
cpp(j,6,3), // "C++"
hv(5,'*'); // "*****"
```

Funkce pro délku

<code>size()</code>	Vrací délku řetězce
<code>length()</code>	Vrací délku řetězce. Stejně jako <code>size()</code> .
<code>resize(size_t n, char c)</code>	Změní délku řetězce na <i>n</i> znaků. Zkrátí, pokud byl delší, nebo doplní znaky <i>c</i> , pokud byl kratší. Nevrací žádnou hodnotu.
<code>resize(size_t n)</code>	Změní délku řetězce na <i>n</i> znaků. Zkrátí, pokud byl delší, nebo doplní binárními nulami, pokud byl kratší. Nevrací žádnou hodnotu.
<code>clear()</code>	Vymaže řetězec (změní ho na prázdný). Nevrací žádnou hodnotu.
<code>empty()</code>	Zjistí, zda řetězec je prázdný.
<code>max_size()</code>	Vrací maximální délku, kterou řetězec může mít.

Příklad.

```
j.resize(9);           // "Jazyk C++"
cpp.resize(5, ' ');    // "C++.."
```

Přístup k jednotlivým znakům

<code>operator [] (size_t pos)</code>	Operátor <code>[]</code> vrací referenci na znak v řetězci s indexem <i>pos</i> .
<code>at(size_t pos)</code>	Funkce <code>at</code> vrací referenci na znak v řetězci s indexem <i>pos</i> . Je-li index <i>pos</i> mimo rozsah řetězce, generuje výjimku <code>out_of_range</code> .
<code>back()</code>	Vrátí referenci na poslední znak v řetězci.
<code>front()</code>	Vrátí referenci na první znak v řetězci.

Příklad.

```
string s("jazyk B++");
cout << s[6];           // B
s[6]='C';
cout << s[6];           // C
s.front()='J';
cout << s;               // Jazyk C++
try { cout << s.at(9); }
catch (out_of_range) { cout << "Mimo retez"; } // Mimo retez
```

Vložení do řetězce

<code>operator += (const string &str)</code>	Přidá řetězec nebo znak na konec řetězce.
<code>operator += (const char *s)</code>	
<code>operator += (char c)</code>	
<code>append(const string &str)</code>	Vrací referenci na objekt řetězce (<code>string &</code>). <i>n</i> je maximální počet znaků, které budou přidány.
<code>append(const string &str, size_t pos, size_t n)</code>	
<code>append(const char *s, size_t n)</code>	
<code>append(const char *s)</code>	
<code>append(size_t n, char c)</code>	

Příklad.

```
string s("C");           // "C"
s += "++";               // "C++"
s += ' ';                // "C++ "
s += "je moderni";       // "C++ je moderni"
string c("C je starsi jazyk");
```

```
s.append(c,11,99); // "C++ je moderni jazyk"
```

<code>push_back(char c)</code>	Přidá znak na konec, stejně jako operator <code>+= (char c)</code> . Nevrací žádnou hodnotu.
--------------------------------	--

<code>assign(const string &str)</code>	Přiřadí řetězci nový obsah. Vrací referenci na objekt řetězce (string &).
<code>assign(const string &str, size_t pos, size_t n)</code>	
<code>assign(const char *s, size_t n)</code>	
<code>assign(const char *s)</code>	
<code>assign(size_t n, char c)</code>	

Příklad.

```
string s; // ""  
s.assign("C++"); // "C++"  
string c("C je starsi jazyk");  
s.assign(c,0,1); // "C"
```

<code>insert(size_t pos1, const string &str)</code>	Vloží na pozici <i>pos1</i> řetězec nebo znaky (poslední formát vloží $n \times c$). Vrací referenci na objekt řetězce (string &).
<code>insert(size_t pos1, const string &str, size_t pos2, size_t n)</code>	
<code>insert(size_t pos1, const char *s, size_t n)</code>	
<code>insert(size_t pos1, const char *s)</code>	
<code>insert(size_t pos1, size_t n, char c)</code>	

Odstranění z řetězce

<code>erase(size_t pos=0, size_t n=npos)</code>	Z pozice <i>pos</i> odstraní <i>n</i> znaků. Je-li uveden jen první argument, vymaže řetězec od pozice <i>pos</i> , není-li uveden žádný argument, vymaže celý řetězec. Vrací referenci na objekt řetězce (string &).
<code>pop_back()</code>	Odstraní poslední znak z řetězce.

Příklad.

```
string s("C++ je moderni jazyk!");  
s.erase(6,8); // "C++ je jazyk!"  
s.pop_back(); // "C++ je jazyk"
```

Nahrazení v řetězci

<code>replace(size_t pos1, size_t n1, const string &str)</code>

<code>replace(size_t pos1, size_t n1, const string &str, size_t pos2, size_t n2)</code>
<code>replace(size_t pos1, size_t n1, const char *s, size_t n2)</code>
<code>replace(size_t pos1, size_t n1, const char *s)</code>
<code>replace(size_t pos1, size_t n1, size_t n2, char c)</code>
Nahradí podřetězec na pozici <i>pos1</i> délky <i>n1</i> řetězcem, podřetězcem nebo znaky. Je-li uvedena délka <i>n2</i> , je nahrazení v této délce. Vrací referenci na objekt řetězce (<code>string &</code>).

Příklad.

```
string s("C je starsi jazyk");
s.replace(5,6,"moderni"); // "C je moderni jazyk"
s.insert(1,"++");         // "C++ je moderni jazyk"
```

<code>swap(string &str)</code>	Vzájemně vymění řetězce uložené v objektech. Nevrací žádnou hodnotu.
------------------------------------	---

<code>const char *c_str()</code>	Vrací znakové pole s uloženým řetězcem (zakočeným binární nulou) – standardní reprezentace řetězce v jazyce C.
<code>const char *data()</code>	

Hledání v řetězci

<code>find(const string &str, size_t pos=0)</code>
<code>find(const char *s, size_t pos, size_t n)</code>
<code>find(const char *s, size_t pos=0)</code>
<code>find(char c, size_t pos=0)</code>
Hledá v uloženém řetězci první výskyt zadaného podřetězce nebo znaku. Je-li zadána počáteční pozice hledání <i>pos</i> , hledání začíná od této pozice (část řetězce před ní se nepohledává). Je-li zadaný podřetězec nebo znak nalezen, funkce vrací pozici jeho výskytu v prohledávaném řetězci, jinak vrací maximální délku, kterou řetězec může mít (hodnota <i>npos</i>).

Příklad.

```
string s("C++ je moderni jazyk");
s.find("moderni") // 7
s.find("starsi")  // 4294967295
```

<code>rfind(const string &str, size_t pos=npes)</code>
<code>rfind(const char *s, size_t pos, size_t n)</code>
<code>rfind(const char *s, size_t pos=npes)</code>

<code>rfind(char c, size_t pos=npos)</code>

Hledá v uloženém řetězci poslední výskyt zadaného podřetězce nebo znaku. Je-li zadána koncová pozice hledání <i>pos</i> , část řetězce za touto pozicí se nepohledává.
--

<code>find_first_of(const string &str, size_t pos=0)</code>

<code>find_first_of(const char *s, size_t pos, size_t n)</code>

<code>find_first_of(const char *s, size_t pos=0)</code>

<code>find_first_of(char c, size_t pos=0)</code>
--

Hledá první výskyt libovolného znaku obsaženém v zadaném řetězci nebo znaku. Je-li zadána počáteční pozice hledání <i>pos</i> , část řetězce před touto pozicí se nepohledává.
--

Příklad.

```
string s("Jazyk C++");  
s.find_first_of("ABCDEFGH") // 6
```

<code>find_last_of(const string &str, size_t pos=npos)</code>

<code>find_last_of(const char *s, size_t pos, size_t n)</code>
--

<code>find_last_of(const char *s, size_t pos=npos)</code>

<code>find_last_of(char c, size_t pos=npos)</code>
--

Hledá poslední výskyt libovolného znaku obsaženém v zadaném řetězci nebo znaku. Je-li zadána koncová pozice hledání <i>pos</i> , část řetězce za touto pozicí se nepohledává.

<code>find_first_not_of(const string &str, size_t pos=0)</code>

<code>find_first_not_of(const char *s, size_t pos, size_t n)</code>

<code>find_first_not_of(const char *s, size_t pos=0)</code>

<code>find_first_not_of(char c, size_t pos=0)</code>
--

Hledá první výskyt libovolného znaku, který není obsažen v zadaném řetězci nebo znaku. Je-li zadána počáteční pozice hledání <i>pos</i> , část řetězce před touto pozicí se nepohledává.
--

Příklad.

```
string s("Jazyk C++");  
s.find_first_not_of("Jazyk C") // 7
```

<code>find_last_not_of(const string &str, size_t pos=npos)</code>

<code>find_last_not_of(const char *s, size_t pos, size_t n)</code>
--

<code>find_last_not_of(const char *s, size_t pos=npos)</code>

<code>find_last_not_of(char c, size_t pos=npos)</code>
--

Hledá poslední výskyt libovolného znaku, který není obsažen v zadaném řetězci nebo znaku. Je-li zadána koncová pozice hledání <i>pos</i> , část řetězce za touto pozicí se nepohledává.

Podřetězec

```
substr(size_t pos=0, size_t n=npos)
```

Vrací (vytvoří) objekt, v němž je uložen podřetězec, který je ve výchozím objektu na pozici *pos* a má délku nejvýše *n*.

Příklad.

```
string s("Jazyk C++ je moderni jazyk");  
s.substr(6)      // "C++ je moderni jazyk"  
s.substr(6,3)    // "C++"
```

Srovnání řetězců

```
compare(const string &str)
```

```
compare(const char *s)
```

```
compare(size_t pos1, size_t n1, const string &str)
```

```
compare(size_t pos1, size_t n1, const char *s)
```

```
compare(size_t pos1, size_t n1, const string &str,  
        size_t pos2, size_t n2)
```

```
compare(size_t pos1, size_t n1, const char *s, size_t n2)
```

Srovná řetězec uložený v objektu s řetězcem zadaným v argumentu. Výsledek:

hodnota 0 – řetězce jsou stejné

záporná hodnota – řetězec v objektu je menší než řetězec v argumentu

kladná hodnota – řetězec v objektu je větší než řetězec v argumentu

==

!=

<

>

<=

>=

Operátory srovnání řetězců. Lze srovnat dva objekty *string* vzájemně nebo objekt *string* s běžným řetězcem.

Příklad.

```
string s("Jazyk C++");  
s.compare("Jazyk C")      // 1  
s > "Jazyk C"              // true  
"Jazyk C" != s            // true
```

Vstup ze streamu a výstup do streamu

```
operator >>
```

Vstup ze streamu do řetězce.

```
operator <<
```

Výstup obsahu řetězce do streamu.

Příklad.

```
string s("Jazyk C++");
```

```
cout << s;
```

Jazyk C++

Konkatenace řetězců

operator +	Lze zřetěžit dva objekty <i>string</i> nebo objekt <i>string</i> s běžným řetězcem.
-------------------	---

Příklad.

```
string C("C"), pp("++");
```

```
string s="Jazyk " + C + pp;
```

```
cout << s;
```

Jazyk C++