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

`string()`	Prázdný řetězec.
`string(const string &str)`	Řetězec je kopií řetězce, jenž je uložen v objektu, který je argumentem konstruktoru.
`string(const string &str, size_t pos, size_t n=npos)` Řetězec je podřetězcem řetězce uloženém v objektu, jenž je argumentem konstruktoru. Podřetězec začíná na pozici ***pos*** a obsahuje nejvýše ***n*** znaků. Hodnota ***npos*** je statická konstanta třídy *string* (její hodnota je obvykle blízká k hodnotě UINT_MAX).	
`string(const char *s, size_t n)`	Řetězec je kopií řetězce ***s***, přičemž z řetězce ***s*** je vzato nejvýše ***n*** znaků.
`string(const char *s)`	Řetězec je kopií řetězce ***s***.
`string(size_t n, char c)`	Řetězec vytvořený ***n***× opakováním znaku ***c***.

Příklady.

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

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

Funkce pro délku řetězce

`size()`	Vrací délku řetězce.
`length()`	Vrací délku řetězce. Stejné jako **`size()`**.
`resize(size_t n, char c)`	Změní délku řetězce na ***n*** znaků. Zkrátí, pokud byl delší, nebo doplní znaky ***c***, pokud byl kratší. Nevrací žádnou hodnotu.
`resize(size_t n)`	Změní délku řetězce na ***n*** znaků. Zkrátí, pokud byl delší, nebo doplní znaky konce řetězce (binárními nulami), pokud byl kratší. Nevrací žádnou hodnotu.
`clear()`	Vymaže obsah řetězce - změní ho na prázdný. Nevrací žádnou hodnotu.
`empty()`	Zjistí, zda řetězec je prázdný. Vrací *true*, je-li délka řetězce 0, jinak *false*.
`max_size()`	Vrací maximální délku, kterou řetězec může mít.

Příklady.

```
j.resize(9);          // "Jazyk C++"
cpp.resize(5,'.');    // "C++.."
cpp.max_size();       // 4294967294  (UINT_MAX = 4294967295)
```

Přístup k jednotlivým znakům

`operator [] (size_t pos)`	Operátor `[]` vrací referenci na znak v řetězci určený indexem ***pos***.
`at(size_t pos)`	Funkce `at` vrací referenci na znak v řetězci určený indexem ***pos***. Je-li index ***pos*** mimo rozsah řetězce, generuje výjimku `out_of_range`.
`back()`	Vrátí referenci na poslední znak v řetězci.
`front()`	Vrátí referenci na první znak v řetězci.

Příklady.

```
string s("jazyk B++");
cout << s[6];       // B
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

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

Příklady.

```
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"
```

`push_back(char c)`	Přidá znak na konec řetězce. Nevrací žádnou hodnotu. Podobné jako `operator += (char c).`

`assign(const string &str)`	Přiřadí řetězci nový obsah. ***n*** je délka. Vrací referenci na objekt řetězce (`string &`).
`assign(const string &str, size_t pos, size_t n)`	
`assign(const char *s, size_t n)`	
`assign(const char *s)`	
`assign(size_t n, char c)`	

Příklady.

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

`insert(size_t pos1, const string &str)`	Vloží na pozici ***pos1*** řetězec nebo znaky (poslední formát vloží ***n***× znak ***c***). Vrací referenci na objekt řetězce (`string &`).
`insert(size_t pos1, const string &str, size_t pos2, size_t n)`	
`insert(size_t pos1, const char *s, size_t n)`	
`insert(size_t pos1, const char *s)`	
`insert(size_t pos1, size_t n, char c)`	

Odstranění z řetězce

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

Příklady.

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

Nahrazení v řetězci

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

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

Příklady.

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

`swap(string &str)`	Vzájemně vymění řetězce uložené v objektech. Nevrací žádnou hodnotu.

`const char *c_str()`	Vrací znakové pole s uloženým řetězcem (zakončeným binární nulou) – vrací standardní reprezentaci řetězce v jazyce C.
`const char *data()`	

Hledání v řetězci

`find(const string &str, size_t pos=0)`
`find(const char *s, size_t pos, size_t n)`
`find(const char *s, size_t pos=0)`
`find(char c, size_t pos=0)`
Hledá v uloženém řetězci první výskyt zadaného podřetězce nebo znaku. Je-li zadána počáteční pozice hledání ***pos***, 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 hodnotu *string::npos*.

Příklady.

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

`rfind(const string &str, size_t pos=npos)`
`rfind(const char *s, size_t pos, size_t n)`
`rfind(const char *s, size_t pos=npos)`
`rfind(char c, size_t pos=npos)`

Hledá v uloženém řetězci poslední výskyt zadaného podřetězce nebo znaku. Je-li zadána koncová pozice hledání ***pos***, prohledává se jen část řetězce před touto pozicí. Vrací hodnotu *string::npos*, není-li nalezen.

```
find_first_of(const string &str, size_t pos=0)
find_first_of(const char *s, size_t pos, size_t n)
find_first_of(const char *s, size_t pos=0)
find_first_of(char c, size_t pos=0)
```

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

Příklad.

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

```
find_last_of(const string &str, size_t pos=npos)
find_last_of(const char *s, size_t pos, size_t n)
find_last_of(const char *s, size_t pos=npos)
find_last_of(char c, size_t pos=npos)
```

Hledá poslední výskyt libovolného znaku obsaženém v zadaném řetězci nebo znaku. Je-li zadána koncová pozice hledání ***pos***, prohledává se jen část řetězce před touto pozicí.

```
find_first_not_of(const string &str, size_t pos=0)
find_first_not_of(const char *s, size_t pos, size_t n)
find_first_not_of(const char *s, size_t pos=0)
find_first_not_of(char c, size_t pos=0)
```

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

Příklad.

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

```
find_last_not_of(const string &str, size_t pos=npos)
find_last_not_of(const char *s, size_t pos, size_t n)
find_last_not_of(const char *s, size_t pos=npos)
find_last_not_of(char c, size_t pos=npos)
```

Hledá poslední výskyt libovolného znaku, který není obsažen v zadaném řetězci nebo znaku. Je-li zadána koncová pozice hledání ***pos***, prohledává se jen část řetězce před touto pozicí.

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***.

Příklady.

```
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: záporná hodnota – řetězec uložený v objektu je menší než řetězec v argumentu hodnota 0 – řetězce jsou stejné kladná hodnota – řetězec uložený v objektu je větší než řetězec v argumentu	

`==`
`!=`
`<`
`>`
`<=`
`>=`
	Operátory srovnání řetězců. Lze srovnat dva objekty *string* nebo objekt *string* s řetězcem ve tvaru jazyka C.

Příklady.

```
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 zřetězit dva objekty *string* nebo objekt *string* s běžným řetězcem.

Příklad.

```
string C("C"),pp("++");
string s="Jazyk " + C + pp;
cout << s;
```

Jazyk C++