Builder.cz - http://www.builder.cz
Autor: Radim Dostál
Rubrika: C/C++
Datum vydání: 24.05. 2001
Url: http://www.builder.cz/art/cpp/cpp_string.html

Řetězce v C++

V tomto článku si ukážeme jak pracovat s řetězci v C++. V jazyce C je řetězec pole znaků a jako s polem se s ním musí pracovat. Některé pomocné funkce pro práci s řetězci jsou v hlavičkovém souboru `string.h` . V C++ je situace jiná. Standardní knihovna C++ obsahuje parametrizovanou třídu (šablonu) `string`. Něco o šablonách si povíme později.

Práce s šablonou `string` je velice snadná. O šablonách nemusíme vědět nic, abychom mohli se stringem pracovat jako s jakýmkoliv jiným datovým typem (třídou). Práce s těmito řetězci je velice podobná práci s řetězci v jiných vyšších programovacích jazycích. `String` má mimo jiné přetíženy operátory `==` a `!=` pro porovnávání řetězců, operátor `=` pro přirovnání, operátory `+`, `+=` pro spojení řetězců. U tohoto řetězce se nemusíme starat o jeho velikost. Řetězec se automaticky zvětší vždy, kdy je to potřeba. Pro `string` jsou také přetíženy operátory `<<` a `>>` pro vstup a výstup z (do) datových proudů.
Třída `string` je deklarována v hlavičkovém souboru `string`. Ještě než ukážu první příklad chtěl bych trochu podrobněji rozvést, co vlastně musím pomocí `include` vložit. V souboru `string.h` jsou deklarace funkcí pro práci s polem znaků z jazyka C. Jsem přesvědčen, že budete-li používat třídu `string`, nebudete tyto funkce potřebovat. V souboru `string` je deklarace šablony `string`. Chcete-li pracovat i s šablonou `string`, i s funkcemi pro pole znaků, musíte vložit oba hlavičkové soubory. Nyní už jednoduchý příklad.

```
#include<string>
#include<iostream>
using namespace std;
int main(void)
{
  string a = "Ahoj", b("svete"); /* Inicializace pomocí operátoru = a pomocí konstruktoru. Jak jsem již dříve doporučoval, druhá možnost je lepší. */
  string c,d;
  cout << a << " " << c << endl;
  c = a + string(" ") + c; /* Operator + je přetížen pouze pro dva stringy. V uvozovkách je ale pole znaků. Proto jsem musel použít konstruktor. */
  cout << c << endl;
  cout << "Neco napis" << endl;
  cin >> d;
  cout << d; << endl;
  cout << "3. znak:" << c[2] << " z poctu:" << c.length() << endl;
  const char *stary = c.c_str();
  cout << stary << endl;
  /* Nyní ukážu některé operátory */    if (a == b)
  {
```

```
      cout << "a == b" << endl;
   }
   if (a == string("Ahoj"))
   {
      cout << "a je Ahoj" << endl;
      a += string(" ctenari");
      cout << a << endl;
   }
   return 0;
}
```

Metoda length vrací velikost řetězce. Metoda `c_str()` vrací konstantní pole znaků. Tato metoda slouží pro kompatibilitu se starými řetězci z jazyka C. Nemusíte se tedy bát používat `string`. Budete-li někde potřebovat pole znaků, pomůžete si "konverzní" metodou `c_str()`.

Závěrem bych ještě chtěl jen dodat, že některé starší překladače nemusejí `string` znát, u jiných překladačů sice `string` je, ale jmenuje se jinak. Například Borland C++ Builder 1 má tuto třídu pod názvem `AnsiString`.

Tolik tedy k řetězcům v C++. Příště se podíváme na vyjímky v C++. Na klíčové slova `catch`, `try`, `throw` a vše, co s vyjímkami souvisí.