Builder.cz - http://www.builder.cz
Autor: Radim Dostál
Rubrika: C/C++
Datum vydání: 07.05. 2001
Url: http://www.builder.cz/art/cpp/cpp_memstream.html

Paměťové proudy v C++

V tomto článku se podíváme na takzvané "paměťové proudy". Ve jednom z mých předchozích článků jménem "Vstupní a výstupní operace pomocí datových proudů v C++ " jsem popsal datový proud jako proud dat od zdroje k cíli. Paměťový proud je proud, jehož cílem, nebo zdrojem je paměť (Nebo přesněji řetězec v paměti.). Paměťové proudy slouží k formátovaní řetězců v paměti počítače. Práce s paměťovými proudy v C++ je vlastně obdoba práce s funkcemi sprintf a sscanf v jazyce C.

Paměťový proud charakterizuje třída `strstream`. Třída `strstream` je potomkem tříd `iostream` a `strstreambase`. Pro nás je nyní důležitý hlavně fakt, že `strstream` je potomkem `iostream`, tedy že vlastně "`strstream` je `iostream`". Tedy všude, kde je očekáván vstupně výstupní proud lze použít paměťový proud. Z toho vyplývá, že pro paměťový proud máme k dispozici přetížené operátory `<<` a `>>` pro všechny datové typy, pro které jsou tyto operátory přetížené u třídy `iostream`. Nyní si ukážeme jednoduchý příklad jak pracovat s paměťovým proudem.

```
#include <iostream>
#include <strstream>
int main(int argc, char *argv[])
{
   char buffer[1000]; /* Paměť, nad kterou bude vše probíhat. */
   strstream proud(buffer,1000,ios::out);
   int cislo = 50;
   proud << "Ahoj svete " << endl << cislo << ends;
   /* Nyní v poli buffer je řetězec "Ahoj světe \n50" */
   cout << buffer << buffer << endl;
   return 0;
}
```

Nejprve jsem vytvořil paměťový blok o velikosti 1000 bytů. Jedná se vlastně o řetězec. Poté jsem na dalším řádku vytvořil paměťový proud jménem `proud`. Jako parametry jsem předal `char *` jako adresu paměti, na kterou je proud "napojen". Tento řetězec bude cílem proudu. Dalším parametrem je velikost paměti, nad kterou bude proud pracovat. Posledním parametrem je mód otevření proudu, v tomto případě bude proud výstupní. O módech otevření jsem psal již ve svém minulém článku. Dalším možným je například `ios::in` - poté proud bude vstupní. Lze použít třídy `istrstream`, resp. `ostrstream`. Poté nemusíme zadávat zda je proud vstupní, nebo výstupní. (Tedy zda řetězec je zdroj, nebo cíl.) Použijeme-li zde mód `ios::app` a `ios::ate`, budou se data připisovat na konec řetězce. Konec řetězce je znak '\0', nikoliv konec bufferu. Ještě bych chtěl upozornit na fakt, že řetězec se neukončí manipulátorem `endl`, ale manipulátorem `ends`. Snažil jsem se to i demonstrovat. Manipulátor `ends` vloží jako další znak '\0', což je konec

řetězce. Při používání paměťových proudů se nemusíme bát zapisování mimo vyznačený buffer. V konstruktoru proudu jsem zapsal velikost paměti 1000 bytů. Budu-li do proudu zapisovat data delší než označených 1000 znaků, proud se dostane do "chybového" stavu (viz můj článek "Přetěžování operátorů << a >> pro datové proudy v C++") a další data nepřijme. Můžete udělat jednoduchý experiment a přepsat v předchozím příkladu obě čísla 1000 třeba na 5. Nemusíte se bát zapsání dat za rozsah pole. Možná ale narazíte na jiný problém. Řetězec končí znakem '\0', což buffer, na který byl napojen takový proud s chybou, končit nebude. Proto se nejspíš stane, že při výstupu uvidíte kromě prvních 5 znaků také nějaký "odpad", dokud nebude v paměti nalezen znak '\0'.

V dalším příkladu předvedu jak zapisovat za konec řetězce.

```
#include <iostream>
#include <strstream>
int main(int argc, char *argv[])
{
   char buffer[40]; /* Paměť, nad kterou bude vše probíhat. */
   ostrstream proud(buffer,40,ios::ate | ios::app);
   cout << "Něco napište" << endl;
   cin.get(buffer,40);
   proud << "Můj přídavek" << ends;
   if (proud)
   {
      cout << buffer << endl;
   }
   else
   {
      cerr << "Bohužel se to tam nevešlo " << endl;
   }
   return 0;
}
```

Před vypsáním jsem zjistil, jestli je proud bez chyby.

Paměťové proudy jsou skvělým nástrojem na zpracovávání řetězců ještě před jejich zapsáním na výstup, nebo do souboru. Dá se také pomocí nic převádět řetězec na číslo a naopak. Stejně tak se nemusejí používat jen pro práci s řetězci. Lze pomocí metod `read`, nebo `write` použít paměťový proud k přenosu bloku dat. Myslím si, že tohle ale není zrovna nejlepší nápad a v tomto případě by bylo lepší použít jiný rychlejší způsob.

Tolik tedy k paměťovým proudům. Dokončili jsme téma datové proudy a příště se podíváme na tak zvané prostory jmen (namespace).

------ http://www.builder.cz ------