Builder.cz - http://www.builder.cz
Autor: Radim Dostál
Rubrika: C/C++
Datum vydání: 09.04. 2001
Url: http://www.builder.cz/art/cpp/cpp_iostream.html

Vstupní a výstupní operace pomocí datových proudů v C++

Vstupní a výstupní operace pomocí datových proudů v C++

V tomto článku si ukážeme jak pracovat se vstupem a výstupem v C++. V grafických operačních systémech se již standardní vstup a výstup příliš nepoužívá. Přesto je dobré tyto možnosti znát. Standardní vstup a výstup se používá například v CGI (O C++ a CGI v tomto seriálu také jednou budu psát.) a také mnohdy v Unixových OS pří psaní filtrů.

Každý, kdo čte mé články, si určitě všimnul, že pro výpis na stdout nepoužívám knihovnu jazyka C - `stdio`, ale používám `iostream`. V C++ lze také používat knihovnu `stdio`, já bych ale doporučoval používat `iostream`, na jejíž výhody bych chtěl v tomto článku poukázat.

Datový proud

Datový proud je reprezentován abstraktní (velmi abstraktní) třídou `ios`. Datový proud reprezentuje vždy nějaký "přesun" dat (plynutí dat, nebo proud dat) od zdroje k cíli. Ze třídy `ios` dědí další třídy jako například `istream` (vstupní proud), `ostream` (výstupní proud), `fstreambase` (datové proudy pro soubory), `strstreambase` (paměťové datové proudy pro řetězce). U tříd dědících z `ios` je naprosto běžná vícenásobná dědičnost. Třídy mají mnoho metod, které zde nebudu vypisovat. Lze je najít v každé dokumentaci, nebo i v hlavičkovém souboru `iostream.h` . V hlavičce `iostream.h` jsou definovány objekty `cout, cin, cerr` . Právě ty pro nás budou nyní důležité.

název objektu	je instance třídy	datový prou pro	v jazyce C
cout	ostream	výstup	stdout
cin	istream	vstup	stdin
cerr	ostream	chybový výstup	stderr

Třída `ostream` má přetížený operátor `<<` (operátor bitového posunu) pro všechny primitivní datové typy a také pro pole `char`, tedy vlastně pro řetězce. Význam tohoto operátoru je poslat svůj pravý operand do datového proudu, který je levým operandem. Operátor `<<` vrací referenci na instanci `ostream`, takže je možné operátory `<<` dávat "za sebe". Vše si můžeme ukázat na následujícím příkladu, kdy pošleme nějaká data proudem na `stdout` a `stderr`.

```
#include <iostream.h>
int main(void)
{
   cout << "Ahoj" << endl;
   char a = 'A';
   unsigned int c = 1000;
   bool pravda = true;
   cout << a << c << pravda << endl;
   cerr << "Toto je chybový výstup:" << pravda << endl;
   return 0;
}
```

Výraz `cout << a` vrací opět `cout` (referenci). Proto je možné napsat `cout << a << c`. Důvody proč používat proudy místo funkcí z jazyka C mě napadají dva:

- Do proudu se dají poslat tak zvané manipulátory.
- Jednoduše se dá přetížit operátor << pro uživatelem definované typy.

Manipulátory

Manipulátory, jak již sám název napovídá, slouží k manipulaci s proudem. Jeden manipulátor již dlouho bez vysvětlení používám. Jedná se o manipulátor `endl` . Význam některých manipulátorů ukážu v tabulce:

Manipulátor	Význam
endl	Vloží konec řádku a vyprázdní buffer (vyrovnávací paměť) proudu.
flush	Vyprázdní buffer proudu.
setw	Minimální počet znaků pro vypsání hodnoty. Tento manipulátor má jeden celočíselný parametr.
dec	Výpis čísel bude v desítkové soustavě.
oct	Výpis čísel bude v osmičkové soustavě.
hex	Výpis čísel bude v šestnáctkové soustavě.
setfill	Tento manipulátor má 1 parametr. Určuje jakým znakem bude vyplňováno volné místo, je-li nastaveno setw.

Doporučuji Vám používat raději manipulátor `endl`, než posílat na proud znak `'\n'`, protože `endl` také vyprázdní buffer. Použijete-li nějaký manipulátor s parametrem, musíte vložit hlavičkový soubor `iomanip.h` . Použití manipulátorů si ukážeme na následujícím příkladě:

```
#include <iostream.h>
```

```
#include <iomanip.h>
int main(void)
{
   unsigned int c = 1000, b = 9;
   cout << setw(3) << "b=" << b << endl << "c=" << c << endl;
   cout << setw(3) << setfill('@') << hex << "c=" << c << endl << "b=" << b << endl;
   return 0;
}
```

Jak vidíme, manipulátory s parametrem "platí" jen do dalšího odeslání dat na `cout`. Pro každý manipulátor s parametrem je k dispozici metoda, která provede to samé. Například místo `cout << setw(6);` lze napsat `cout.width(6);` .

Opomenuli jsme objekt `cin`. Tento objekt reprezentuje datový proud "spojený" se standardním vstupem. Jak asi mnohé napadne třída `istream` má přetížený operátor `>>` pro vstup. Pravý operand operátoru je instance třídy `istream` a levý operand je proměnná, která má přijmout data. Operátor `>>` je přetížen pro všechny primitivní datové typy i pro pole znaků. U pole znaků může nastat jen jeden "malý" problém. Operátor `>>` čte znaky jen do prvního bílého znaku. Bílý znak je v tomto případě i mezera. Tedy stejný problém, jako když funkce `scanf` z jazyka C čte řetězec. Celý řádek lze přečíst pomocí metody `istream& getline(char*, int, char = '\n')` , kde první parametr je řetězec, druhý parametr je maximální počet znaků a třetí parametr je ukončovač řádku, jehož implicitní hodnota je '\n'. Použití ukážu na příkladu:

```
#include <iostream.h>
int main(void)
{
   char a, s[100];
   cout << "Napiš řetězec" << endl;
   cin.getline(s,100);
   cout << "Napsal jsi " << s <<endl;
   cout << "Napiš znak" << endl;
   cin >> a;
   cout << "Napsal jsi " << a <<endl;
   return 0;
}
```

Tolik tedy k úvodu. Možná jsem Vás ještě nepřesvědčil o používání proudů místo funkci z `stdio`. Výhoda proudů se projeví hlavně při přetěžování operátorů << >> pro uživatelské datové typy a třídy. O tom jak tyto operátory přetěžovat, a také jak pracovat s textovými soubory, si povíme v příštím článku.

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