Proudy a soubory v C++

Proudy

- pro vstup a výstup z/na standardní vstup/výstup a standardní chybový výstup jsme používali proudy `cin, cout, cerr`
  - typů `ostream, istream`

Další proudy

- existují i jiné proudy:
  - např. pro soubor:
    - `fstream`
    - `ofstream`
    - `ifstream`
  - pro výstup do řetězce
    - `istringstream`
    - `ostringstream`
    - `stringstream`

Proudy a soubory

- otevření souboru
  - při deklaraci
    - jméno souboru je parametrem konstruktoru
  - metodou `open`
    - parametrem je navíc mód otevření souboru

Příklad:

```
ofstream soubor("vystup.txt");
```

nebo

```
ofstream soubor;
soubor.open("vystup.txt", ios::out);
```

- zavření souboru - metodou `close()`

Módy otevření

`ios::in`	pro vstup (implicitní pro vstupní soubory)
`ios::out`	pro výstup (implicitní pro výstupní soubory)
`ios::binary`	binární mód (nejsou interpretovány řídicí znaky)
`ios::ate`	pozice nastavena na konec
`ios::app`	pozice pro výstupní operace nastavena na konec
`ios::trunc`	soubor bude přepsán (vyprázdněn)

- kombinace módů = ***bitový součet***

Testování stavu souboru

- všechny metody vracejí typ bool:

`is_open()` soubor úspěšně otevřen

`bad()` předchozí operace čtení nebo zápisu neúspěšná

`fail()` totéž co bad(), navíc pokud došlo k formátovací chybě

`eof()` pozice ve vstupním souboru je na konci souboru

`good()` vše je O.K.

`clear()` maže všechny příznaky

Čtení a zápis dat

- přetíženými operátory <<, >>
- metodami
  - `get(char &c), get(), put(char c)`
  - `read(char *str, streamsize count )`
  - `write(const char *str, streamsize count )`

- další pomocné funkce, např:
  - `getline(char *s,int n)` pro čtení celého řádku
  - `gcount()` pro zjištění, kolik bytů bylo přečteno

Příklad

```
#include <iostream>
#include <fstream>
using namespace std;
int main()
{
  ofstream soubor("vystup.txt");
  if (!soubor.is_open()) return -1;
  soubor << "Ahoj, světe" << endl;
  soubor.close();
  return 0;
}
```

Zkopírujeme textový soubor po řádcích

```
#include <iostream>
#include <fstream>
using namespace std;
int main()
{
  char radek[81];
  ifstream vstup("vstup.txt“);
  ofstream vystup("vystup.txt");
  while (vstup.eof()==false)
  {
    vstup.getline(radek,81);
    vystup << radek << endl;
  }
  vstup.close();  vystup.close();
  return 0;
}
```

kopieradky.cpp

- funkce getline() přečte řádek do konce, ale znak konce řádku do řetězce nevloží
  - proto tisknu `vystup << radek << endl;`
- bude-li řádek v souboru delší než 80 znaků, funkce getline() jej nenačte celý
  - nastaví příznak fail, který můžeme testovat voláním metody rdstate() nebo fail():
  - `if ((vstup.rdstate() & failbit)!=0) { ... }`
  - `if ((vstup.fail()==true) { ... }`
- lepší řešení:
  - využijeme typ `string`, který implementuje řetězec „neomezené“ délky
  - pak musíme použít globální funkci getline()

```cpp
#include <iostream>
#include <fstream>
#include <string>
using namespace std;
int main()
{
  string radek;
  ifstream vstsoubor("vstup.txt");
  ofstream vystsoubor("vystup.txt");
  while (vstsoubor.eof()==false)
  {
    getline(vstsoubor,radek);
    vystsoubor << radek << endl;
  }
  vstsoubor.close();  vystsoubor.close();
  return 0;
}
```

kopieradky2.cpp

Řešení má jeden háček...

1. Poslední řádek v souboru není ukončen znakem konce řádku
   - funkce getline() přečte poslední řádek
   - protože při čtení nenarazí na znak konce řádku, ale na konec souboru, nastaví se příznak konce souboru
   - vytiskne se řádek do souboru + endl
   - při následném testu podmínky ve while cyklus končí
   - ve výstupním souboru mám znak konce řádku navíc

Řešení má jeden háček...

2. Poslední řádek v souboru je ukončen znakem konce řádku

- funkce getline() přečte poslední řádek
- protože při čtení narazí na znak konce řádku, nenastaví se příznak konce souboru
- vytiskne se řádek do souboru + endl
- při následném testu podmínky ve while cyklus nekončí, protože eof vrací false
  - cyklus proběhne ještě jednou, getline() do řetězce nic nenačte (je prázdný); vytiskne se prázdný řetězec + endl
- ve výstupním souboru mám opět znak konce řádku navíc

Řešení ...

- tisknu řetězec a znak konce řádku zvlášť
- znak konce řádku vytisknu, není-li konec souboru

```
while(!vstup.eof())
{
  getline(vstup,radek);
  vystup << radek;
  if (!vstup.eof()) vystup << endl;
}
```

kopieradky3.cpp

Jména souborů zadáme z klávesnice

```
#include <iostream>
#include <fstream>
#include <string>
using namespace std;
int main()
{
  string radek;
  string jmeno_soub;
  cout << "Zadej jmeno vstupniho souboru: ";
  getline(cin,jmeno_soub);
  ifstream vstsoubor(jmeno_soub);
  if (vstsoubor.is_open()==false)
  {
    cerr << "Soubor " << jmeno_soub << " nelze otevrit!";
    return -1;
  }
```

```
  cout << "Zadej jmeno vystupniho souboru: ";
  getline(cin,jmeno_soub);
  ofstream vystsoubor(jmeno_soub);
  if (vystsoubor.is_open()==false)
  {
    cerr << "Soubor " << jmeno_soub << " nelze otevrit!";
    vstsoubor.close();
    return -1;
  }
  while(!vstsoubor.eof())
  {
    getline(vstsoubor,radek);
    vystsoubor << radek;
    if (!vstsoubor.eof()) vystsoubor << endl;
  }
  vstsoubor.close();  vystsoubor.close();
  return 0;
}
```

kopieradky4.cpp

Poznámky

- jméno souboru není vhodné číst pomocí `cin >> jmeno_vst,` protože pokud obsahuje jméno souboru mezeru, načte se z klávesnice text pouze do první mezery
- `ofstream vystsoubor(jmeno_soub);`
  - do verze C++ 11 mohl být parametr jméno pole znaků (řetězec v klasickém C), typ string až od C++11; je nutné zapnout překlad dle C++11
- otevření souboru lze provést také takto:

```
int main()
{
  string radek;
  string jmeno_soub;

  ifstream vstsoubor;
  ofstream vystsoubor;

  cout << "Zadej jmeno vstupniho souboru: ";
  getline(cin,jmeno_soub);
  vstsoubor.open(jmeno_soub);
  if (vstsoubor.is_open()==false)
  {
    cerr << "Soubor " << jmeno_soub << " nelze
  otevrit!\n";
    return -1;
  }
```

kopieradky5.cpp

Jména souborů jako parametry programu

```
int main(int argc, char *argv[])
{
  string radek;

  ifstream vstsoubor;
  ofstream vystsoubor;

  if (argc<3)
  {
    cerr << "Spousteni: " << argv[0] << " vstup vystup";
    return -1;
  }
  vstsoubor.open(argv[1]);
```

Jména souborů jako parametry programu

```
if (vstsoubor.is_open()==false)
{
  cerr << "Soubor " << argv[1] << " nelze otevrit!\n";
  return -2;
}

vystsoubor.open(argv[2]);

if (vystsoubor.is_open()==false)
{
  cerr << "Soubor " << argv[2] << " nelze otevrit!\n";
  vstsoubor.close();
  return -2;
}
```

kopieradky6.cpp

Úloha 1

Napište program, který zkopíruje textový soubor (po znacích), všechna malá písmena anglické abecedy převede na velká. Použijte streamy. Jména souborů zadávejte jako parametry hlavního programu, ošetřete správnost otevření souboru. Využijte např. metod get a put:

- přečtení jednoho znaku `c=vstup.get()`
- zápis jednoho znaku `vystup.put(c);`

Kostra čtení znaků

```
přečti znak
while(vstup.eof()==false)
{
    if (c je malé písmeno) c = c - ('a' - 'A');
    ulož znak do souboru
    přečti znak
}
```

Proudy a řetězce

- umožňují zápis do/čtení ze řetězce způsobem práce se streamy
  - srovnejte se `sprintf, sscanf`
  - řetězec je reprezentován typem `string` z knihovny `string`
    - `#include <string>`

Výstup do řetězce

- **`ostringstream`**
- konstruktory

```
ostringstream(openmode which = ios_base::out);
ostringstream( const string & str, openmode
  which = ios_base::out );
```

- metoda `str()`
  - vrací/nastavuje řetězec, se kterým proud pracuje

```
#include <string>
#include <sstream>
int main()
{ string retez;
  ostringstream oss(retez);
  oss << "Pisi do retezce";
  cout << retez;
  return 0;
}
```

Příklad 2:

```
#include <string>
#include <sstream>
int main()
{ string retez;
  ostringstream oss;
  oss << "Pisi do retezce";
  retez = oss.str();
  cout << retez;
  return 0;
}
```

Vstup ze řetězce

- **`istringstream`**
- konstruktory

```
istringstream(openmode which =
  ios_base::in);
istringstream( const string & str, openmode
  which = ios_base::in );
```

- hodí ze pro parsing (analýzu) řetězce

Příklad

- máme textový soubor obsahující na každém řádku město (jednoslovný název) a číslo (počet chodníků)
- úkolem je napsat program, který přečte soubor po řádcích, vynechá prázdné řádky a do řetězce město uloží název, do proměnné x typu int počet chodníků a vytiskne text na obrazovku, počet chodníků zdvojnásobí

Příklad

```
while(!vstup.eof())
{
  getline(vstup,radek);
  if(radek.length()> 0)
  {
    istringstream analyz(radek);
    analyz >> mesto; // cte do bileho znaku
    analyz >> pocet;
    pocet *=2;
    cout << mesto << '\t' << pocet << endl;
  }
}
```

soubor.cpp

Poznámka

- funkce `getline` může mít další parametr
  - ukončující znak (delimiter); pokud na něj při čtení funkce narazí, vyzvedne ho, neuloží do řetězce a zastaví čtení; další volání funkce čte data za tímto znakem

Příklad

- máme 3 údaje na řádku v textovém souboru (jméno, příjmení, rodné číslo) oddělené tabulátorem (čárkou, středníkem, ...)
  - načteme řádek do stringstreamu
  - jednotlivé údaje získáváme pomocí
  - `getline(stream,data,'\t')`

Příklad

```
string radek, jmeno, prijmeni, RC;
istringstream analyz;

while(!vstup.eof())
{
  getline(vstup,radek);
  analyz.str(radek);
  analyz.clear();
  getline(analyz,jmeno,'\t');
  getline(analyz,prijmeni,'\t');
  getline(analyz,RC);
  cout << jmeno << ';' << prijmeni << ';' << endl;
}
```

soubor2.cpp

Příklad jinak

```
string radek, jmeno, prijmeni, RC;
istringstream analyz;
string *data[3] = {&jmeno, &prijmeni, &RC}
while(!vstup.eof())
{
  getline(vstup,radek);
  analyz.str(radek);
  analyz.clear();
  for(i=0;i<3;i++)  getline(analyz,*data[i],'\t');
  cout << jmeno << ';' << prijmeni << ';' << endl;
}
```

soubor3.cpp

Poznámka

- volání metody `clear`

```
analyz.clear();
```

zajistí vymazání příznaků uvnitř proudu

- pokud bychom metodu nezavolali, proud by si "pamatoval", že došel na konec "souboru" (řetězce) a ve druhém běhu cyklu by nic nenačetl
  - přiřazení nového řetězce příznaky automaticky nenuluje