Informatika 3

Objektovo-orientovaný vstup/výstup – Prúdy (streams)

9

Prečo prúdy ako triedy a nie FILE*

```
class cFileClass {
  FILE* f;
public:
  cFileClass(const char* fname, const char* mode = "r")
  {
    if((f = fopen(fname, mode)) == 0)
     throw exception("Chyba otvorenia suboru");
  }

  ~cFileClass() { fclose(f); };
  FILE* fp() { return f;  }
};
```

Prečo prúdy ako triedy a nie FILE*

```
class File {
  FILE* f;
  FILE* F(); // Dáva kontrolovaný smerník na f
public:
  File(); // Vytvor objekt ale neotváraj súbor
  File(const char* path, const char* mode = "r");
  ~File();
  int open(const char* path, const char* mode = "r");
int getc();
  int ungetc(int c);
  int putc(int c);
  int puts(const char* s);
  char* gets(char* s, int n);
  int printf(const char* format, ...);
  size_t read(void* ptr, size_t size, size_t n);
  size_t write(const void* ptr, size_t size, size_t n);
  int eof();
  int close();
  int flush();
  int seek(long offset, int whence);
  int getpos(fpos_t* pos);
  int setpos(const fpos_t* pos);
  long tell();
  void rewind();
  void setbuf(char* buf);
  int setvbuf(char* buf, int type, size_t sz);
};
```

Prečo prúdy ako triedy a nie FILE*

- Deštruktor zatvorí prúd (pre zábudlivcov)
- Veľká knižnica
- Ponechať výhody ??printf(formát,...) ale bez réžie parsovania reťazca formát
- V C neexistuje kontrola chýb vo formáte
- Nie je možné ?printf rozširovať o vlastné formáty

Prúdy

- Základné triedy prúdov
  - istream – vstupný prúd
  - ostream – výstupný stream
  - stream – obsahuje metódy pre vstup aj výstup
- Súborové prúdy
  - ifstream – vstupný prúd
  - ofstream – výstupný stream
  - fstream – obsahuje metódy pre vstup aj výstup
- Prúdy nad reťazcami
  - istringstream – vstupný prúd
  - ostringstream – výstupný stream
  - stringstream – obsahuje metódy pre vstup aj výstup
- Všetky varianty majú identické rozhrania
- V skutočnosti sú prúdy špecializácie šablón. Napr:

```
typedef basic_istream<char> istream;
typedef basic_ostream<char> ostream;
typedef basic_stream<char> stream;
```

KST

Vkladač/vyberač

- Pre pohodlnú prácu s prúdmi
  - Vkladač:

    ostream& operator<<(ostream&, int);
    ostream& operator<<(ostream&, float);
    ostream& operator<<(ostream&, double);
    ...
  - Vyberač:

    istream& operator>>(istream&, int&);
    istream& operator>>(istream&, float&);
    istream& operator>>(istream&, double&);
    ...
  - cout, cin, cerr sú globálne deklarované prúdy štandardného vstupu, výstupu a prúdu pre chybové hlásenia

    cout << “a*b=” << a*b << endl;
  - identické

    operator<<(operator<<(operator<<(cout, “a*b=“), a*b), endl);

Vlastný vkladač

- preťažovaním príslušných operátorov
  - Prvým parametrom je nekonštantný odkaz na prúd (**istream** pre vstup, **ostream** pre výstup)
  - Vykonáme operáciu vložením/vybratím dát do/z prúdu (samozrejme prostredníctvom spracovania prvkov objektu)
  - Vrátime odkaz na prúd

```
ostream& operator<<(ostream& os, const Datum& d) {
  char fillc = os.fill('0');
  os << setw(2) << d.dajDen() << '-'
     << setw(2) << d.dajMesiac() << '-'
     << setw(4) << setfill(fillc) << d.dajRok();
  return os;
}
```

Vlastný vyberač

```
istream& operator>>(istream& is, Datum& d) {
  char pomlcka;
  is >> d.den;
  is >> pomlcka;
  if (pomlcka != '-')
    is.setstate(ios::failbit);
  is >> d.mesiac;
  is >> pomlcka;
  if (pomlcka != '-')
    is.setstate(ios::failbit);
  is >> d.rok;
  return is;
}
```

- nastavením **fail** bitu signalizujeme chybu prúdu
- Po nastavení chybového bitu prúdu sa všetky nasledujúce prúdové operácie ignorujú až kým prúd neobnovíme do ***dobrého*** stavu

```
// Datum.h
#include <iosfwd>  // nemusíme vkladať celú definíciu triedy do H
class Datum {
  friend std::ostream& operator<<(std::ostream&,const Datum&);
  friend std::istream& operator>>(std::istream&, Datum&);
  // atď.
```

Stav prúdu

- ios, definuje štyri indikátory:
  - **badbit** - Nastala nejaká fatálna (napr. hardvérová) chyba. Prúd by mal byť považovaný za nepoužiteľný.
  - **eofbit** - Nastal koniec vstupu (buď narazením na fyzický koniec súborového prúdu alebo ukončenie užívateľského konzolového prúdu napríklad Ctrl-Z alebo Ctrl-D).
  - **failbit** - Zlyhala vstupno/výstupná operácia, najpravdepodobnejšie kvôli neplatným dátam (napríklad narazilo sa na písmena pri pokuse čítať číslice). Prúd je stále použiteľný, tento bit sa nastavuje i v prípade konca vstupu.
  - **goodbit** - Všetko je v poriadku, žiadne chyby. Koniec vstupu ešte nenastal.
- Užitočné metódy:
  - good() – true, ak nie je nastavený žiaden z prvých troch
  - eof() – true, ak je nastavený eofbit
  - fail() – true, ak je nastavený failbit alebo badbit
  - bad() – true, ak je nastavený badbit
  - clear() – vynuluje všetky chybové bity (treba urobiť ručne, nemažú sa automaticky napr. pri presunutí pozície v súbore)

```
int i;
while (mojPrud >> i)  // volá sa ios_base::operator void *() čo volá good()
  cout << i << endl;
```

Prúdy a výnimky

- Prúdy vznikli v C++ skôr ako výnimky - kvôli kompatibilite testy cez good() a bad()
- Ak chceme výnimky, musíme ich zapnúť pre konkrétny typ chyby:
  - mojPrud.exceptions(ios::badbit);
- Generuje sa std::ios_base::failure čo je potomok std::exception
- Nie je rozumné nastavovať vynimku pre eofbit – nie je to nič výnimočné

Formátovacie indikátory

```
fmtflags ios::flags(fmtflags newflags);
fmtflags ios::setf(fmtflags ored_flag);
fmtflags ios::unsetf(fmtflags clear_flag);
```

- Prvá nastaví všetky, ostatné len konkrétny indikátor
- Jednoducho zapnúť/vupnúť cez setf()/unsetf()
  - **ios::skipws** Preskoč bielu medzeru. (pre vstup; implicitne nastavený)
  - **ios::showbase** Indikuj základ čísla (aký je nastavený, napríklad **dec**, **oct** alebo **hex**) pri tlači zabudovanej hodnoty. Vstupné prúdy tiež rozpoznávajú prefix základu ak je **showbase** zapnutý.
  - **ios::showpoint** Zobrazuj desatinnú bodku a koncové nuly pre hodnoty v pohyblivej rádovej čiarke.
  - **ios::uppercase** Zobrazuj veľké písmena A-F pre hexadecimálne hodnoty a E pre vedecké hodnoty.
  - **ios::showpos** Zobrazuj znamienko plus (+) pre kladné hodnoty.
  - **ios::unitbuf** “Jednotkové bufrovanie.” Prúd sa vyprázdňuje po každom vložení.

Skupinové formátovacie indikátory

fmtflags ios::setf(fmtflags bits, fmtflags field);

- Môže byť nastavený iba jeden indikátor zo skupiny, inak nedefinované správanie.
- **ios::basefield – parameter field**
  - **ios::dec**Formátuje zabudované hodnoty v základe 10 (decimálne) (implicitný základ—nie je viditeľný žiaden prefix). Aj s vymazaním ostatných: setf(ios::dec, ios::basefield).
  - **ios::hex**Formátuje zabudované hodnoty v základe 16 (hexadecimálne).
  - **ios::oct**Formátuje zabudované hodnoty v základe 8 (oktálovo).
- **ios::floatfield**
  - **ios::scientific** Zobrazuj čísla v pohyblivej rádovej čiarke vo vedeckom formáte. Položka presnosti určuje počet číslic za desatinnou bodkou.
  - **ios::fixed** Zobrazuj čísla v pohyblivej rádovej čiarke v pevnom formáte. Položka presnosti určuje počet číslic za desatinnou bodkou.
  - **“automatická” (Ani jeden bit nie je nastavený.)** Položka presnosti určuje celkový počet platných číslic.
- **ios::adjustfield**
  - **ios::left** Hodnoty zarovnané vľavo. Vyplňujúci znak doplnený sprava.
  - **ios::right** Hodnoty zarovnané vpravo. Vyplňujúci znak doplnený zľava. Toto je implicitné zarovnanie.
  - **ios::internal** Vlož vyplňovacie znaky za znamienko alebo indikátor základu, ale pred hodnotu. (Inými slovami znamienko, ak sa tlačí, je zarovnané vľavo a čísla vpravo).
- **Šírka, výplň, presnosť**
  - **int ios::width()** Vracia aktuálnu šírku. (Implicitne je 0.) = Minimálny počet znakov
  - **int ios::width(int n)** Nastavuje šírku, vracia predchádzajúcu šírku
  - **int ios::fill()** Vracia aktuálny vyplňovací znak. (Implicitne je to medzera.)
  - **int ios::fill(int n)** Nastavuje vyplňovací znak, vracia predchádzajúci vyplňovací znak
  - **int ios::precision()** Vracia aktuálnu presnosť čísla v pohyblivej rádovej čiarke. (Implicitne je 6.)
  - **int ios::precision(int n)** Nastavuje presnosť čísla v pohyblivej rádovej čiarke, Vracia predchádzajúcu presnosť.

Manipulátory

- Môžeme ich vkladať priamo do prúdu
- Ľahšie sa pracuje (hlavne oproti supinovým indikátorom)
  - **dec**, **oct** a **hex**, vykonávajú rovnaké činnosti ako funkcie **setf(ios::dec, ios::basefield), setf(ios::oct, ios::basefield)** a **setf(ios::hex, ios::basefield)**.
  - **ws** pre vstupný prúd „zje“ biele medzery
  - **endl** = ‘\n’ + vyprázdni buffer
  - **flush** vyprázdni výstupný buffer
  - **showbase/noshowbase** Indikuje číselný základ (**dec**, **oct** alebo **hex**) pri tlači celočíselných hodnôt. Použitý formát môže čítať C++ kompilátor.
  - **showpos/noshowpos** Zobrazuje znamienko plus (+) pre kladné hodnoty.
  - **uppercase/nouppercase** Zobrazuje A-F pre hexadecimálne hodnoty a zobrazuje E pre vedecké hodnoty.
  - **showpoint/noshowpoint** Zobrazuje desatinnú bodku a koncové nuly pre čísla v pohyblivej rádovej bodke.
  - **skipws/noskipws** Preskakuje biele medzery na vstupe.
  - **left** Zarovnanie vľavo, doplnenie vpravo.
  - **right** Zarovnanie vpravo, doplnenie vľavo.
  - **internal** Vyplňovanie medzi znamienkom alebo indikátorom základu a hodnotou.
  - **scientific/fixed** Indikuje prednostné zobrazovanie výstupu čísel v pohyblivej rádovej čiarke. (vedecká notácia vs. pevná desatinná bodka).

```
cout << hex << “0x” << i << endl;
cin >> ws;
```

Manipulátory s argumentami

- Existuje šesť štandardných manipulátorov, ktoré akceptujú argumenty. Tieto sú definované v hlavičkovom súbore **<iomanip>**
  - **setiosflags (fmtflags n)** Ekvivalent volania **setf(n)**. Nastavenie zostáva v platnosti až po ďalšiu zmenu, napríklad **ios::setf()**.
  - **resetiosflags(fmtflags n)** Maže iba formátovacie indikátory, dané hodnotou **n**. Nastavenie zostáva v platnosti až po ďalšiu zmenu, napríklad **ios::unsetf()**.
  - **setbase(base n)** Mení základ na **n**, kde **n** je 10, 8 alebo 16. (Hocičo iné dáva 0.) Ak je **n** nulové, výstup bude so základom 10, ale vstup použije C konvencie: 10 je 10, 010 je 8 a 0xf je 15. Pre výstup by sme mohli tiež použiť **dec**, **oct** a **hex**.
  - **setfill(char n)** Mení vyplňovací znak na **n**, ako to robí funkcia **ios::fill()**.
  - **setprecision(int n)** Mení presnosť na **n**, rovnako ako funkcia **ios::precision()**.
  - **setw(int n)** Mení šírku položky na **n**, rovnako ako **ios::width()**.