Základy programování v C++ 19. cvičení

Zuzana Petříčková

5. prosince 2018

Přehled

Datové proudy

Základní myšlenka

- data proudí mezi zdrojem a spotřebičem
  1. program → soubor, tiskárna, obrazovka (konzole)
  2. soubor, klávesnice → program

Datový proud

- nástroj pro přenos dat mezi zdrojem a spotřebičem (např. souborem a programem)

Standardní datové proudy v C++ (už známe)

- **cin** ... standardní vstup (obvykle klávesnice, lze přesměrovat prostředky OS), instance třídy **istream**
- **cout** ... standardní výstup (obvykle obrazovka, lze přesměrovat prostředky OS), instance třídy **ostream**
- **cerr** ... standardní chybový výstup (obvykle obrazovka, lze přesměrovat prostředky OS), instance třídy **ostream**

Práce se soubory

- se soubory můžeme pracovat podobně jako s datovými proudy **cin, cout**
- datové proudy pro soubory (v ASCII kódování) jsou deklarované v hlavičkovém souboru **fstream**:
  - **ifstream** ... proud pouze pro vstup
  - **ofstream** ... proud pouze pro výstup
  - **fstream** ... pokud chceme střídat vstup a výstup
- datové proudy pro soubory s širokými (Unicode) znaky jsou deklarované v hlavičkovém souboru **wfstream**:
  - **wifstream**, **wofstream**, **wfstream**
- sekvenční přístup k souboru
- rozlišujeme textové a binární soubory

Práce se soubory

Datové proudy (třídy) ifstream, ofstream, fstream

- metoda **open()** ... otevření datového proudu
- metoda **is_open()** ... test, zda je datový proud otevřen
- metoda **close()** ... uzavření datového proudu
- operátory <<, >> a další funkce a metody, které známe pro **cin** a **cout**
  (jsou zděděné od společného předka: **ios, istream, ostream**)
  - **getline(), get()** ... načte řádek/kus textu ze vstupního datového proudu
    (rozdíl: getline() zahodí oddělovač, get() ho nechá „ve frontě")
  - **ignore()** ... načte a zahodí kus textu ze vstupního datového proudu
  - **clear()** ... změní/zruší chybový stav datového proudu
    ...

Zápis do souboru

- **Příklad 1:** funkce (vytvoří a) otevře soubor, zapíše do něj "Ahoj!"a soubor zase zavře.

```
#include<fstream>
...
void zapisDoSouboru()
{
    ofstream soubor;
    soubor.open("text.txt");
    if (soubor.is_open()) // if (soubor)
    {
        soubor << "Ahoj!" << endl;
        soubor.close();
    }
    else
    {
        chyba("soubor nelze otevrit!");
        //throw (string)"soubor nelze otevrit!";
    }
}
```

Zápis do souboru

- **Příklad 2**: funkce zapíše do souboru čísla od 1 do 10

```
void zapisDoSouboru(string nazev)
{
    ofstream soubor;
    soubor.open(nazev);
    if (soubor.is_open()) // if (soubor)
    {
        soubor << "Zapisuji cisla: " << endl;
        for (int i = 1; i <= 10; i++)
        {
            soubor << i << endl;
        }
        soubor.close();
    }
    else { ... }
}
```

Čtení ze souboru

- **Příklad 3**: funkce načte číslo ze souboru

```
void nactiZeSouboru()
{
    ifstream soubor;
    soubor.open("text.txt");
    if (soubor.is_open()) // if (soubor)
    {
        int i
        soubor >> i; // + kontrola, ze se nacteni povedlo
        ...
        soubor.close();
    }
    else { ... }
}
```

Čtení ze souboru

- **Příklad 4**: funkce vypíše obsah souboru na konzoli

```
void nactiZeSouboru(string nazev)
{
    ifstream soubor;
    soubor.open(nazev);
    if (soubor.is_open()) // if (soubor)
    {
        string radek;
        while (getline(soubor, radek))
        {
            cout << radek << endl;
        }
        soubor.close();
    }
    else { ... }
}
```

Metoda open()

```
ofstream soubor1, soubor2, soubor3;
string t = "text.txt";
soubor1.open("text.txt"); // relativni cesta
soubor1.open(t);          // cesta jako string
soubor2.open("D:\\tmp\\text.txt");
                          // absolutni cesta (MS Windows)
soubor3.open("text.txt", ios::out);
                          // druhy parametr: rezim otevreni
fstream soubor4;
soubor4.open("text.txt", ios::out);
```

- režimy otevření:
  - ios::in ... otevřít pro vstup (implicitní pro ifstream)
  - ios::out ... otevřít pro výstup (implicitní pro ofstream)
  - ios::binary ... otevřít binárně (implicitní je textově)
  - ios::app ... výpis na konec souboru
  - ios::trunc ... vymaže soubor
  - ios::in | ios::out ... otevřít pro vstup i výstup (impl. pro fstream)

Načítání ze vstupního datového proudu

```
int x;
soubor >> x;
string s;
getline(soubor, s);    // cely radek
getline(soubor,s,'.'); // text az po znak '.'
...
if (!(soubor>>x)) {/*nacteni se nepodarilo*/}
if(!getline(soubor, s)) {/*nacteni se nepodarilo*/}

soubor >> x;
if (!soubor) /*nacteni se nepodarilo*/
{
    soubor.clear();
}
getline(soubor, s);
if (!soubor) /*nacteni se nepodarilo*/
{
    soubor.clear();
}
```

Práce se soubory

Chybové stavy datových proudů:

- datové proudy obsahují příznaky naznačující chybu
  - vyčištění příznaků metodou **clear()**
- metody pro testování stavu:
  - přetížené chování jako **bool**, přetížený operátor **!**
  - good() ... načtení hodnoty se povedlo
  - fail() ... načtení hodnoty se nepovedlo
  - eof() ... je konec souboru

```
while (soubor.good()) // while (soubor >> x) { ... }
{
    soubor >> x;
    ...
}
if(soubor.eof()) { ... }
else if(soubor.fail()) { ... }
```

Datový proud jako parametr funkce

```
// datovy proud predavame jako referenci!
void zapis(ostream &s, int x)
{
    s << x << " ";
}
int main()
{
    ofstream soubor;
    soubor.open("text.txt");
    zapis(cout, 3);
    if (soubor)
    {
        zapis(soubor, 4);
        soubor.close();
    }
    return 0;
}
```

Manipulátory

- speciální objekty, které je možno přidávat k operátorům <<a >>
- knihovna **iomanip**
  - manipulátory bez parametrů: **endl**, **left**, **right** (mění zarovnání),...
  - manipulátory s parametry: **setw()**, **setprecision()**, **setfill()**,...

```
const double pi = 3.1415;
soubor << pi << endl;
soubor << endl << right << setw(10) << setfill('_')
       << setprecision(3) << pi << endl;
soubor  << endl << left << setw(10) << setfill('_')
        << setprecision(3) << pi << endl;
```

Příklady

```
/* Funkce, ktera vrati pocet radku souboru.*/
int pocetRadku(string nazev);

/* Ulozeni seznamu do souboru a jeho nacteni ze souboru
*/
void vypis(ostream &kam, int x);
void vypis(ostream &kam, const Zamestnanec &z);
void vypis(ostream &kam, Seznam &s);
void vypis(Seznam &s, string nazev);

void nacti(istream &odkud, int &x);
void nacti(istream &odkud, Zamestnanec &z);
void nacti(Seznam &s, string nazev);
```

Uložení seznamu do souboru

```
// 1. upravit existujici funkce pro vypis, aby byl vystupni
//      proud parametrem:
void vypis(ostream &kam, int x);
void vypis(ostream &kam, const Zamestnanec &z);
void vypis(ostream &kam, Seznam &s);

// 2. napsat zastresujici funkci, ktera otevre soubor,
//     zavola predchozi funkci a soubor zase zavre
void vypis(Seznam &s, string nazev);
```

Uložení seznamu do souboru, načtení ze souboru

```
// 1. upravit existujici funkce pro vypis, aby byl vystupni
//      proud parametrem:
void vypis(ostream &kam, int x);
void vypis(ostream &kam, const Zamestnanec &z);
void vypis(ostream &kam, Seznam &s);

// 2. napsat zastresujici funkci, ktera otevre soubor,
//    zavola predchozi funkci a soubor zase zavre
void vypis(Seznam &s, string nazev);


// 1. napsat pomocne funkce pro nacteni dat ulozenych
//    v prvku seznamu
void nacti(istream &odkud, int &x);
void nacti(istream &odkud, Zamestnanec &z);

// 2. napsat zastresujici funkci, ktera otevre soubor,
      v cyklu bude nacitat a pridavat prvky do seznamu
      (zavola predchozi funkce)
      a nakonec soubor zase zavre
void nacti(Seznam &s, string nazev);
```

Ukazatel na funkci

V jazycích C/C++ jsou dva typy ukazatelů:

- ukazatele na data (už známe)
- ukazatele na funkce
  - obsahují adresu vstupního bodu do funkce

K čemu je dobrý ukazatel na funkci?

- funkce jako proměnná / parametr jiné funkce
- pole ukazatelů na funkce (programování nabídky/menu)

Ukazatel na funkci

Deklarace ukazatele na funkci:

```
typ (*identifikator)(seznam_typu_parametru);
```

Příklady

```
int (*F) (int, int);
   /* F je ukazatel na funkci typu int
      se dvema parametry typu int */

void (*G) (Zamestnanec*, double);
   /* G je ukazatel na funkci bez navratove hodnoty
      se dvema parametry typu ukazatel na Zamestnance a double */

bool (*H) (void);
   /* H je ukazatel na funkci typu bool bez parametru */

Prvek* (*I) (Seznam&);
   /* I je ukazaten na funkci, ktera vraci ukazatel na Prvek
      s parametrem typu reference na seznam */
```

Ukazatel na funkci

Přiřazení hodnoty (v C++ dvě ekvivalentní možnosti):

```
double (*S) (double);
S = sin;
S = &sin;          // jazyk C
```

Volání funkce, na kterou ukazatel ukazuje
(v C++ dvě ekvivalentní možnosti)

```
double x;
x = S(3.14);
x = (*S)(3.14);      // jazyk C
```

Pole ukazatelů na funkce

```
// deklarace:
double(*pole_fci[5])(double);

//deklarace s inicializaci:
double(*pole_fci1[])(double) = {sin, cos, tan};

// prace s polem:
pole_fci[0] = sin;
pole_fci[1] = cos;
pole_fci[2] = sqrt;

// priklad volani:
vysledek = pole_fci[1](-1);
```

Příklad: hledání kořenů funkce půlením intervalů

Tvrzení

Buď f spojitá na (a, b). Nechť $f(a)f(b) < 0$. Pak f má na (a, b) alespoň jeden kořen $(x : f(x) = 0)$.

Postup řešení rovnice $f(x) = 0$ půlením intervalů

1. označme $a_0 = a, b_0 = b$
2. spočteme střed intervalu: $c = \frac{a_i + b_i}{2}$
3. určíme další prvky a_{i+1}, b_{i+1}:
   - $a_{i+1} = a_i, b_{i+1} = c$ pokud $f(a_i)f(c) < 0$
   - $a_{i+1} = c, b_{i+1} = b_i$
4. opakováním postupu dostaneme dvě monotónní posloupnosti (a_n) a (b_n): $a \leq a_1 \leq a_2 \leq ... \leq a_n < b_n \leq ... \leq b_1 \leq b$
5. hledaný kořen spočteme jako $lim_{n->\infty} a_n == lim_{n->\infty} b_n$

Příklad: hledání kořenů funkce půlením intervalů

Algoritmus

- Protože $lim_{n\to\infty} a_n == lim_{n\to\infty} b_n$ v programu spočítat nemůžeme, spokojíme se s přibližným výsledkem:
  - konec, pokud $|a_n - b_n| < \varepsilon$, pro $\varepsilon > 0$

Příklad:

```
double pulenIntervalu(double(*F)(double),
                      double a, double b, double eps);
```

pomocí funkce spočtěte kořeny funkcí:

- $(tg(x) - x)$ na intervalu $(\pi/2, 3\pi/2)$
- $(\sqrt{x+2} - 2)$ na intervalu $(-2, 5)$

Spojový seznam ... zbývá setrideni seznamu

```
// setrideni seznamu metodou bublinkoveho trideni
void setrid(Seznam &s);
{
    // 0. pokud je seznam prazdny, konec
    // 1. zaved si pomocnou promennou typu bool (indikator,
    //    zda v dane iteraci cyklu byly nejake prvky prohozeny)
    // 1. provadej nasledujici prikazy, dokud
    //    "v dane iteraci cyklu byly nejake prvky prohozeny":
    // 1.1. poznamenej si, ze (zatim) zadne prvky prohozeny nebyly
    // 1.1. adresu hlavy uloz do pomocneho ukazatele
    // 1.2. dokud se s timto ukazatelem nedostanes k zarazce:
    // 1.2.1. pokud je aktualni prvek vesti nez nasledovnik:
    // 1.2.1.1. prohod data v aktualnim prvku a v jeho nasledovniku
    //          ("stara znama" funkce prohod())
    // 1.2.2.2. poznamenej si, ze nejake prvky byly prohozeny
    // 1.2.2.   posun pomocny ukazatel na dalsi prvek
}
/* pro Zamestnance: analogicky (jedina zmena bude v 1.2.1)
   (popr. predefinuji operator > pro Zamestnance) */
```