Pokročilé programování v jazyce C pro chemiky

(C3220)

Vstup a výstup v C++

Proudy pro standardní vstup a výstup

- V jazyce C++ provádíme textový vstup a výstup prostřednictvím tzv. datových proudů
- Datové proudy jsou ve skutečnosti specializované třídy dostupné ve standardní knihovně jazyka C++, které poskytují metody a operátory pro zápis a čtení dat
- Pro práci se standardními vstupy a výstupy je třeba v záhlaví zdrojového souboru deklarovat `#include <iostream>` a `using namespace std;`
- Standardní knihovna obsahuje následující třídy pro standardní vstup a výstup:
  - `ios` – základní třída, na níž jsou založeny vstupní a výstupní proudy
  - `istream` – pro vstupní operace
  - `ostream` – pro výstupní operace
- Ve standardní knihovně jsou definovány následující proměnné:
  - `istream cin` – pro standardní vstup (z klávesnice)
  - `ostream cout` – pro standardní výstup (na obrazovku)
  - `ostream cerr` – pro chybový výstup, bez bufferování (na obrazovku)
  - `ostream clog` - pro chybový výstup (na obrazovku)

Operátory pro vstup a výstup

- Pro výstup do proudu používáme operátor **<<**, pro vstup **>>**
- Tyto operátory lze použít pouze pro čtení/zápis proměnných základních datových typů (`int`, `double`, `char`) a typu `string`

```
#include <iostream>
using namespace std;


int main()
{
  int i = 5;
  double a = 10;
  string str;

  cout << "Hodnota promenne i: " << i << endl;
  cout << "Hodnota promenne a: " << a << endl;

  cin >> str;
  cout << "Retezec je: " << str << endl;

  return 0;
}
```

Vstup a výstup objektových proměnných

- Operátory **<<** a **>>** nelze přímo použít pro čtení/zápis celých objektových proměnných
- Pro objektové proměnné musíme číst/zapisovat jednotlivé členy třídy samostatně

```
#include <iostream>
using namespace std;

// Na zacatku je definovana trida Circle – viz. minule cviceni

void Circle::readValues()
{
  cin >> x >> y >> radius >> color;
}

int main()
{
  Circle circ;
  circ.readValues();
  cout << circ.getCentreX() << circ.getCentreY() << circ.GetColor();

  // Nasledujici by nefungovalo:
  cin >> circ;
  cout << circ;
  return 0;
}
```

Souborový vstup a výstup

- Pro zápis do souboru a čtení ze souboru používáme následující třídy:
  - `fstream` – pro čtení i zápis téhož souboru
  - `ifstream` – pro vstupní souborové operace
  - `ofstream` – pro výstupní souborové operace
- Tyto třídy jsou odvozeny ze základních tříd pro standardní vstup a výstup `ios`, `istream`, `ostream`
- Pro práci se souborovými vstupy a výstupy je třeba v záhlaví zdrojového souboru deklarovat `#include <fstream>` a `using namespace std;`

Zápis do souboru

- Pro výstup do souboru vytvoříme proměnnou typu `ofstream`
- Soubor otevřeme metodou `open()`, které předáme jméno souboru. Stejného výsledku dosáhneme i předáním jména souboru konstruktoru (v definici proměnné).
- Úspěšnost otevření souboru ověříme pomocí logického operátoru `!`, případně ekvivalentní metody `fail()`. Oboje vrací pravdivou hodnotu, pokud bylo otevření souboru neúspěšné
- Pro zápis dat používáme operátor `<<`
- Soubor můžeme uzavřít pomocí metody `close()`. Neučiníme-li tak, zavře ho destruktor třídy fstream (princip RAII)

```
#include <fstream>
using namespace std;

int main()
{
  ofstream ofile;           // Definujeme promennou proudu
  ofile.open("test.dat");   // Otevreme soubor
  If (!ofile) {             // Nebo take: if (ofile.fail())
    cout << "Nelze otevrit soubor!\n";
    return 1;
  }
  ofile << "Tento text se zapise do souboru" << endl;
  ofile.close(); // Nemusi byt, nechceme-li testovat uspesnost
  return 0;
}
```

Čtení ze souboru

- Pro čtení dat ze souboru vytvoříme proměnnou typu `ifstream`
- Jméno souboru předáme konstruktoru nebo metodě `open()`
- Úspěšnost otevření souboru ověříme pomocí ! nebo `fail()`
- Pro čtení dat používáme operátor `>>`
- Soubor můžeme ručně uzavřít pomocí metody `close()`

```
#include <fstream>
using namespace std;

int main()
{
  double a1 = 0.0, a2 = 0.0;
  ifstream ifile("test.dat"); // Muzeme pouzit i open()/close()
  if (!ifile)
  {
    cout << "Nelze otevrit soubor!\n";
    return 1;
  }
  ifile >> a1 >> a2;
  return 0;
}
```

Diagnostika I/O chyb

- Po otevření souboru a také po načtení nebo zápisu znaku potřebujeme zjistit, zda byla operace úspěšná. K tomu používáme následující metody:

  `fail()` – vrací **true**, pokud byla operace neúspěšná

  `eof()` – vrací **true**, pokud předchozí operace dosáhla konce souboru, bez ohledu na úspěšnost (týká se jen čtení)

  `good()` – vrací **true**, pokud byla poslední operace úspěšná a proud je dále použitelný (tedy nenastal `fail()` ani `eof()`)

```
ifstream ifile("test.dat");
if (ifile.fail())  // testujeme uspesnost otevreni souboru
{
  cout << "Nelze otevrit soubor!\n";
  return 1;
}

double a = 0;
ifile >> a;   // Nacitani hodnoty do promenne a
if (ifile.fail())  // testujeme uspesnost nacteni
{
  cout << "Nepodarilo se nacist hodnotu!" << endl;
}
```

Zrušení chybového stavu proudu

- Pokud byla předchozí operace neúspěšná (tj. `fail()` vrátí pravdivou hodnotu) je další načítání/zápis z/do proudu blokováno, tj. selžou všechny operace, které se o to pokusí
- Tento chybový stav musíme odstranit voláním metody `clear()`, teprve potom můžeme provádět další operace čtení a zápisu

```
double a;
string s;
ifstream ifile("test.dat");

// Nasledujici pokus o nacteni cisla bude neuspesny (napr. protoze
// ve vstupnim souboru se nachazeji neciselne znaky)
ifile >> a;
if (ifile.fail()) {
  cout << "Chyba pri cteni cisla." << endl;
  // Volanim clear() zrusime chybovy stav proudu
  ifile.clear();
}

// Nyni nacteme text. Pokud bychom predtim nezavolali clear(),
// zadny text by se nenacetl, protoze proud by byl v chybovem stavu.
ifile >> s;
cout << "Nacteny text: " << s << endl;
```

Proudy pro vstup a výstup z/do řetězce

- Data lze také načítat nebo zapisovat do řetězce, tj. proměnné typu `string`. K tomu používáme následující třídy:
  - `stringstream` – pro načítání/zápis z/do řetězce
  - `istringstream` – pro načítání z řetězce
  - `ostringstream` – pro zápis do řetězce
- Tyto třídy jsou odvozeny ze základních tříd pro standardní vstup a výstup `ios`, `istream`, `ostream`
- Pro práci se souborovými vstupy a výstupy je třeba v záhlaví zdrojového souboru deklarovat `#include <sstream>` a `using namespace std;`

Proudy pro vstup a výstup z/do řetězce

- Pro načítání dat z řetězce vytvoříme proměnnou typu `istringstream`, pro zápis `ostringstream`, a při inicializaci jim předáme jméno řetězcové proměnné
- Operátor `>>` používáme pro čtení, operátor `<<` pro zápis
- K diagnostice úspěšnosti načítání opět používáme `!` či metodu `fail()`

```
#include <sstream>
using namespace std;

int main()
{
  double a1 = 0.0, a2 = 0.0, a3 = 0.0;
  string s = "3.24 1.2 5.7";

  istringstream sstream(s);

  sstream >> a1 >> a2 >> a3;

  if (sstream.fail()) {
    cout << "Chyba pri nacitani z retezcoveho proudu" << endl;
  }
  return 0;
}
```

Proudy pro vstup a výstup z/do řetězce

- Pokud chceme načítat z jiného řetězce, můžeme použít stejný proud, do kterého nastavíme nový vstupní řetězec pomocí metody `str()`
- Poté zavoláme metodu `clear()`, abychom zrušili případnou chybu z předchozího načítání

```
#include <iostream>
#include <sstream>
using namespace std;

int main()
{
  double a1 = 0.0, a2 = 0.0, a3 = 0.0;
  double b1 = 0.0, b2 = 0.0, b3 = 0.0;
  string s1 = "3.24 1.2 5.7";
  string s2 = "2.0 4.5 6.0";
  istringstream sstream(s1);

  sstream >> a1 >> a2 >> a3;    // Ted nacitame z retezce s1
  sstream.str(s2);              // Do proudu nastavime retezec s2
  sstream.clear();              // Zrusime pripadny chybovy stav
  sstream >> b1 >> b2 >> b3;    // Ted nacitame z retezce s2

  return 0;
}
```

Načítání souboru po řádcích

- Pro načtení celého řádku z proudu používáme funkci `getline()`, která z proudu načte jeden řádek a přiřadí ho do řetězcové proměnné typu `string` předané jako druhý argument
- `getline()` vrací úspěch, pokud se podařilo něco načíst
- Tuto řetězcovou proměnnou nastavíme do příslušného řetězcového proudu metodou `set()` a načítáme

```
string fileName = "soubor.dat";
double a1 = 0.0, a2 = 0.0;
string s;
istringstream sstream;
ifstream ifile(fileName);
if (ifile.fail())
  { cout << "Nelze otevrit soubor: " << fileName << endl; return;}

while (getline(ifile, s)) { // Nacte jeden radek z proudu ifile do retezce s
  sstream.str(s);  // Proud sstream bude nacitat z retezce s
  sstream.clear(); // Zrusime predchozi chybovy stav
  sstream >> a1 >> a2;
}

if (ifile.fail() && !ifile.eof()) {
  // Neco se pokazilo a nebyl to jen konec souboru
  cout << "Chyba pri nacitani souboru!" << endl;
}
```

Datový proud jako argument funkce

- Pokud proudovou proměnnou předáváme jako argument funkce, předáváme ho tzv. referencí, což vyjádříme přidáním & před jméno předávaného parametru (ne před argument ve volání funkce!)

```
// Proud predavame do funkce odkazem (formou reference), coz
// specifikujeme pridanim & pred jmeno parametru
void read(istringstream &istream)
{
  double a1 = 0.0, a2 = 0.0;
  istream >> a1 >> a2;
}

int main()
{
  string s = "2.13  5.67";
  istringstream sstream;

  sstream.str(s);
  sstream.clear();
  read(sstream);  // Funkci predavame proud sstream, tady bez &

  return 0;
}
```

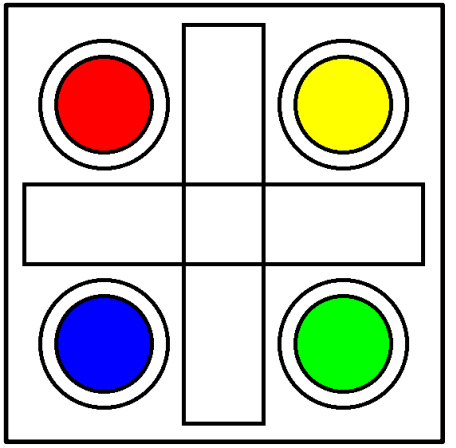
Dodržujte následující pravidla

- Pro práci se souborovými proudy nezapomeňte vložit hlavičkový soubor *fstream* a pro práci s řetězcovými proudy soubor *sstream*
- Pokud předáváte proměnné proudu do metod, předávejte je jako referenci (tj. použijte & před jménem parametru)
- Nepoužívejte v programu globální proměnné, pokud to není nezbytně nutné. Upřednostněte lokální proměnné a ty předávejte do funkcí/metod. Proměnné související s funkcí nějakého objektu umístěte do příslušné třídy.

Cvičení

1. Vytvořte program vycházející z programu v úloze 2 z minulého cvičení. Program uzpůsobte tak, že místo interaktivního vstupu bude údaje o zobrazovaném objektu načítat ze souboru. V souboru bude uveden typ obrazce, který se bude kreslit, souřadnice x, y a případně další potřebné hodnoty (poloměr pro kružnici, šířka a výška pro obdélník, poloměr a číslo barvy kružnice, číslo barvy výplně pro vyplněnou kružnici). Tyto hodnoty budou odděleny mezerami. V souboru bude uveden pouze jeden řádek s jedním objektem (například: *circle 150 150 60 3*). Jméno vstupního souboru bude programu předáno jako parametr na příkazovém řádku. Otestujte se soubory *circle.dat*, *rectangle.dat*, *filled_circle.dat* v adresáři */home/tootea/C3220/data/* **2 body**

2. Vytvořte program, který bude ze souboru načítat grafické objekty. Soubor bude obsahovat více řádků a každý řádek bude obsahovat všechny informace o jednom grafickém objektu ve stejném formátu, jako soubory z předchozí úlohy. Až po načtení celého souboru se všechny tyto objekty vykreslí v jednom okně. Na řádku s klíčovým slovem `window` bude specifikována velikost okna. Program otestujte se souborem */home/tootea/C3220/data/shapes1.dat* (měl by se vykreslit stejný obrazec jako na obrázku, nezáleží na pořadí vykreslování jednotlivých objektů). **3 body**



Úloha 1 – nápověda

- Do tříd Shape, `Circle`, `Rectangle` a `FilledCircle` implementujte metodu `void readFile(ifstream &istream)`, která bude vypadat podobně jako metoda `readValues()`, ale bude načítat data ze souborového proudu, který jí bude předán jako argument.
- Načítání souboru implementujte ve funkci `main()`, jak je uvedeno v příkladu v sekci “Načítání ze souboru”. Ze souboru načtěte řetězec do řetězcové proměnné a podle obsahu této proměnné (“circle”, “rectangle”, “filled_circle”) vytvořte objektovou proměnnou příslušného typu a zavolejte její metodu `readFile()` a následně volejte metodu `printValues()` a pak `draw()`
- Pro snazší vytváření objektů můžete všem třídám přidat prázdné bezparametrové konstruktory (v kombinaci s vhodnými inicializačními hodnotami v definici datových položek), pak lze psát jen `Circle c;` místo `Circle c(0, 0, 0, 0);`
- Argumenty předané programu na příkazovém řádku získáme stejně jako v jazyce C, tj. funkce main() bude přijímat dva parametry `main(int argc, char *argv[])`, kde `argc` obsahuje počet předaných parametrů zvětšený o 1, pole `argv[]` obsahuje seznam parametrů (`argv[0]` obsahuje název souboru s programem, teprve `argv[1]` obsahuje první předaný parametr.)

Úloha 2 – nápověda

- Metodu `readFile()` ve všech objektech modifikujte tak, že bude přijímat proud `istringstream` namísto `ifstream`, tj: `void readFile(istringstream &istream)`. Na začátek souboru nezapomeňte vložit hlavičkový soubor *sstream*.
- Ve funkci `main()` definujte proměnnou pro vektor kružnic `vector<Circle> circles`. Podobně definujte vektory `rectangles` a `filledCircles`.
- Vstupní soubor načítejte v cyklu po řádcích, jak je uvedeno v příkladu v sekci “Načítání souboru po řádcích”. Další postup je podobný jako v předchozí úloze, tj. načtete první slovo a podle toho rozhodnete, který objekt se bude načítat. Použijte lokální proměnnou příslušného typu, na ní zavolejte readFile() k načtení parametrů objektu a připravený objekt vložte do příslušného vektoru pomocí `push_back()`. Pokud je na začátku řádku uvedeno slovo “window”, načtou se rozměry okna (do lokálních proměnných).
- Po načtení celého souboru (a jeho uzavření) dojde k vykreslení všech objektů. Nejdříve se otevře okno (jeho šířka a výška jsou dány hodnotami ve vstupním souboru na řádku „window“) a potom se v cyklu vykreslí všechny načtené kružnice (tj. zavolá se `circle.draw(dev)` pro každou položku `circle` v poli). Potom se totéž provede pro čtverce a vyplněné kružnice. Nakonec program čeká na stisknutí Enter a pak okno zavře.
- Program vylepšete použitím třídy `Drawing`, která bude mít podobnou roli jako v úloze 3 z předchozího cvičení, konkrétně bude obsahovat vektory objektů `circles, filledCircles` a `rectangles` a proměnné pro šířku a výšku okna. Dále bude obsahovat metodu pro načtení dat ze souboru `readFile(string fileName)`, která přijímá jméno souboru jako parametr a metodu `draw()`, která otevře okno a vykreslí do něj všechny grafické objekty. Ve funkci `main()` vytvořte objekt typu `Drawing` a zavolejte jeho metodu `readFile()` a potom `draw()`.