

**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` - pro vstupní a výstupní operace
  - `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 (na obrazovku)
  - `ostream clog` - pro chybový výstup po řádcích (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 použít pro čtení/zápis 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 vstupní a výstupní souborové operace
  - `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
- Úspěšnost otevření souboru ověříme pomocí metody `fail()`, která vrací pravdivou hodnotu pokud bylo otevření souboru neúspěšné
- Pro zápis dat používáme operátor `<<`
- Na konci soubor uzavřeme pomocí metody `close()`

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

int main()
{
    ofstream ofile;           // Definujeme promennou proudu
    ofile.open("test.dat");   // Otevreme soubor
    if (ofile.fail())
    {
        cout << "Nelze otevrit soubor!\n";
        return 1;
    }
    ofile << "Tento text se zapise do souboru" << endl;
    ofile.close();
    return 0;
}
```

# Čtení ze souboru

- Pro čtení dat ze souboru vytvoříme proměnnou typu `ifstream`
- Soubor otevřeme metodou `open()` které předáme jméno souboru
- Úspěšnost otevření souboru ověříme pomocí metody `fail()`
- Pro čtení dat používáme operátor `>>`
- Na konci soubor uzavřeme pomocí metody `close()`

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

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

# Argument metody `open()`

- U staršího standardu C++98 musí být argument, který předáváme do metody `open()` typu `char*`, nelze mu tedy předat přímo proměnnou typu `string`
- Z proměnné typu `string` získáme řetězec `char*` pomocí metody `c_str()`

```
string fileName = "test.dat";
ofstream ofile;
ofile.open(fileName.c_str());

// Nejaké operace pro zápis do proudu ofile

ofile.close();
```

- V novějším standardu C++11 může být argument, který předáváme do metody `open()` typu `string` nebo typu `char*`
- Chceme-li aby překladač `g++` používal novější standard, použijeme při překladači parametr `-std=c++11` (u starších verzí je dostupný pouze předběžný standard pomocí parametru `-std=c++0x`)



# Diagnostika I/O chyb

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

`fail()` - vrací **true** pokud byla operace neúspěšná

`good()` vrací **true** pokud byla operace úspěšná, tedy opačnou hodnotu než `fail()`

`eof()` - vrací pravdivou hodnotu pokud se proud nachází na konci souboru (týká se jen čtení ze souboru)

```
ifstream ifile;
ifile.open("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 **pozastaveno**, 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;
ifile.open("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 funkci `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 nacistani z retezceho 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 řetězec pro načítání 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
- 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;
ifile.open(fileName.c_str());
if (ifile.fail())
    { cout << "Nelze otevrit soubor: " << fileName << endl; return;}

while (!ifile.eof())
{
    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())
        { cout << "Chyba pri nacitani souboru!" << endl; break;}
}
ifile.close();
```

# Datový proud jako argument funkce

- Pokud proudovou proměnnou předáváme jako argument funkce, předáváme ho tzv. **referencí**, což vyjádříme přidáním & před jméno předávaného parametru

```
// Proud predavame do funkce formou reference, coz
// specifikujeme pridanim & pred jmeno parametru
void read(istream &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 read() predavame proud sstream

    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 proudy (streams) 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, které souvisí s funkcí objektů umístěte do třídy.

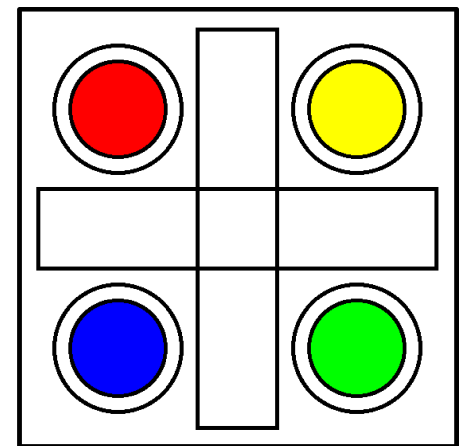


# Cvičení

1. Vytvořte program vycházející z programu v úloze 2 z minulého cvičení. Program upravitel tak, že místo interaktivního vstupu bude údaje o zobrazovaném objektu načítat ze souboru. V souboru bude uveden typ grafiky která se bude kreslit, souřadnice x, y a případně další 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/martinp/C3220/data/*

2 body

2. Vytvořte program, který bude ze souboru načítat grafické objekty. Soubor bude obsahovat více řádků a každý řádek bude obsahovat informace o jednom grafickém objektu ve stejném formátu jako soubory z předchozí úlohy. Po načtení se všechny tyto objekty vykreslí v jednom okně (nemusí se vykreslovat v pořadí jak jsou uvedeny v souboru). Na prvním řádku načítaného souboru bude specifikována velikost okna. Program otestujte se souborem */home/martinp/C3220/data/graphic1.dat* (měl by se vykreslit stejný obrazec jako je na obrázku).



3 body

# Úloha 1 - nápověda

- Do tříd `Graphic`, `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 načtete 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()`
- Argumenty předané programu na příkazovém řádku získáme stejně jako při programování 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 pole kružnic `Circle circles[MAX_GRAPHIC];` (`MAX_GRAPHIC` bude definováno na začátku souboru jako symbolická konstanta `#define MAX_GRAPHIC 100`). Podobně definujte pole `rectangles` a `filledCircles`. Také definujte celočíselné proměnné, které budou obsahovat počet načtených položek (`circlesCount`, `rectanglesCount`, `filledCirclesCount`).
- 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. Po načtení objektu nezapomeňte vždy zvýšit o 1 hodnotu proměnné obsahující počet načtených položek (`circlesCount`, ...). Pokud je na začátku řádku uvedeno slovo “window” načtou se rozměry okna (do lokálních proměnných).
- Po načtení souboru (a jeho uzavření) dojde k vykreslení všech objektů. Nejdříve se otevře okno (jeho šířka a výška jsou dány hodnotami ve vstupním souboru na prvním řádku) a potom se v cyklu vykreslí všechny načtené kružnice (tj. zavolá se `circles[i].draw(dev);`). 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 úloze3 z předchozího cvičení, konkrétně bude obsahovat pole 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()`.