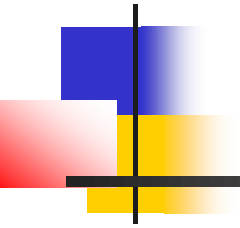
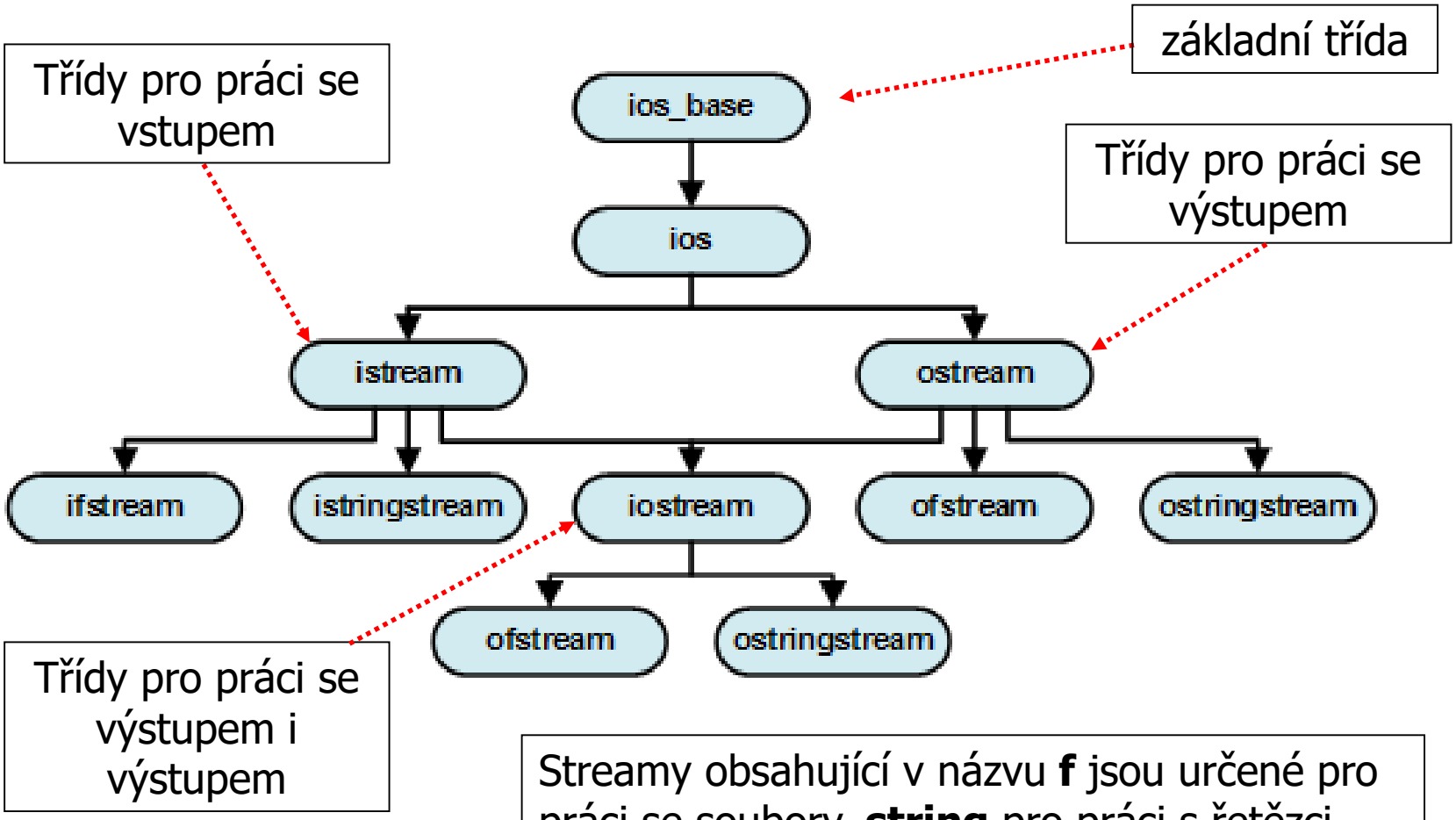


# Programování v C++ I

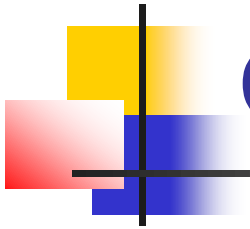
Práce se soubory v  C++ a C

Jan Fesl
jfesl@prf.jcu.cz

# Hierarchie datových proudů



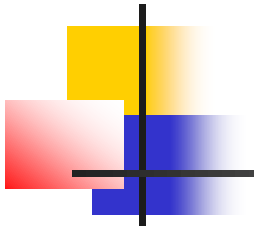

Streamy obsahující v názvu **f** jsou určené pro práci se soubory, **string** pro práci s řetězci



# Co to jsou datové proudy (streamy) ?

- Jsou to třídy, které umožňují vstupně-výstupní operace s daty
- Vstupní, výstupní, vstupně/výstupní
- Paměťové, pro práci se soubory
- Lze samozřejmě pro vstup a výstup používat datový typ **FILE** z C, ale práce se streamy je mnohem pohodlnější

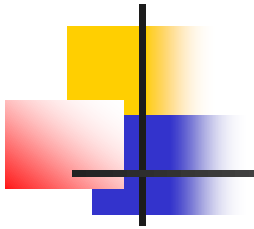
# Kde už jsme se se streamy setkali ?



- cout je instance streamu ostream, která je definována v **prostoru jmen std**
- cin je obdoba cout nicméně, je to instance istream
- Obdobné je to u cerr (chybový výstup), clog (událostní výstup)
- **V C++ nelze vytvořit jinou instanci cout se stejnou funkcí, která je odvozena od ostreamu, cout se chápe jako jedinečný**
- Nicméně je možné vytvořit referenční proměnnou

ostream &mujVystup = cout, který funguje obdobně, nicméně **výkonná část je stále std::cout**

# Čtení a zápis do souboru pomocí streamů



- Používají se **ifstream** (pouze čtení), **ofstream** (pouze zápis), **iofstream** (čtení i zápis)
- Popřípadě lze užít jejich předka **fstream** s módem otevření (ios::in, ios::out) – tím vznikne defakto to samé
- Pro použití těchto streamů je nutné používat direktivu #include <fstream>
- Zda je soubor otevřený se lze dozvědět pomocí metody **bool is_open()**
- Zda jsme na již na konci souboru je možné pomocí funkce **bool eof()**
- Vlastní uzavření souboru způsobí i volání destruktoru příslušného streamu

# Zápis do souboru - ukázka

Zápis čísel do souboru pomocí **ofstream** a **fstream**

```
ofstream out("c:\\vystup.txt");
if (!out)
   cout << "Chyba" << en
else
{
for (int i=0; i < 10; i++)
     out << i << endl;
out.close();
}
```

```
fstream out("c:\\vystup.txt", ios::out);
if (!out)
   cout << "Chyba" << endl;
else
{
for (int i=0; i < 10; i++)
     out << i << endl;
out.close();
}
```

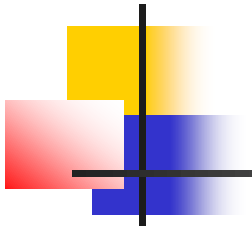
Zápis zde na rozdíl od fprintf() v C probíhá okamžitě a nečeká se na uzavření souboru

# Čtení ze souboru - ukázka

```
ifstream in("c:\\vystup.txt");
if (in == NULL)
   cout << "Chyba" << endl;
else
   {
   int n;
   while (in != NULL)
          {
          in >> n;
          cout << n << endl;
          }
   in.close();
   }
```

Na detekci chyby poslední operace zde používáme adresu instance

Čtení probíhá dokud se neobjeví nějaká chyba



# Čtení z textového souboru

- **ifstream** obsahuje metody na práci s textovým souborem
- metoda **char get()**, pro načtení jednoho znaku
- metoda **void getline(char *znaky, int velikost)**

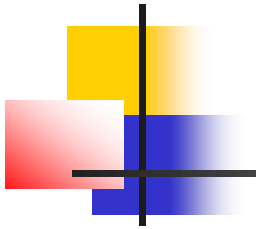
# Načtení a vypsání souboru dvěma způsoby

**Po znacích**

```
ifstream in("c:\\vystup.txt");
if (in == NULL)
cout << "Chyba" << endl;
else
{
char c;
while (in != NULL)
        {
        c = in.get();
        cout << c;
        }
in.close();
}
```

**Po řádkách**

```
ifstream in("c:\\vystup.txt");
if (in == NULL)
cout << "Chyba" << endl;
else
{
char r[256];
while (in != NULL)
        {
        in.getline(r,256);
        cout << r << endl;
        }
in.close();
}
```



# Binární soubory

- Pro vytvoření binárního streamu se používá třída **fstream** a logicky se přičítá binární mód
- **fstream inb**(“cesta”,ios::in | ios::binary);
- **fstream outb**(“cesta”,ios::out | ios::binary);
- Pro čtení a zápis se nepoužívají operátory >> a << nýbrž metody read() a write()
- Funkce read, write mají vždy dva parametry – adresu místa s daty a velikost dat v bytech
- Jedná se o analogii fread a fwrite v C

# Práce s binárními soubory

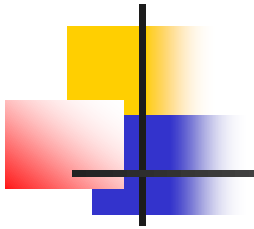
```
char pole[6] = "BCDEF"; char pole1[6];

fstream in1("c:\\vystup.txt",ios::out| ios::binary);
if (in1 != NULL)
  {
  in1.write(pole, 6*sizeof(char));
  in1.close();
  }

fstream in2("c:\\vystup.txt",ios::in| ios::binary);
if (in2 != NULL)
   {
   in2.read(pole1, 6*sizeof(char));
   in2.close();
   cout << pole1 << endl;
    }
```

Zápis do binárního souboru

Čtení z binárního souboru



# Serializace objektů

- Jedná se o proces, kdy se ukládá obsah určitých neprimitivních datových struktur **najednou do výstupního streamu nebo souboru**

- **Výhodou** tohoto procesu je to, že se najednou uloží celá **struktura či třída**, která se nemusí ukládat po částech

- **Nevýhodou** ovšem v C++ je to, že atributy příslušné struktury musí mít fixní velikost, což může znamenat redudanci

- Pro realizaci těchto operací slouží právě funkce **read(char * data, int velikost)** a **write(char * data, int velikost)**

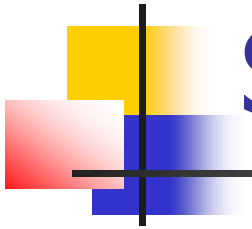
# Ukázka serializace

**Deklarace třídy**

```
class komplexniCislo
{
double realna;
double imaginarni;
public:
komplexniCislo(double realna,
                        double imaginarni)
{
      this->realna = realna;
      this->imaginarni = imaginarni;
}
};
```

**Kód funkce main**

```
komplexniCislo c(1.1, 2.2);
ofstream out("C:\pokus.txt",
ios::binary);
if (out != NULL)
   {
   out.write((char *)&c, sizeof(c));
   out.close();
   }
```



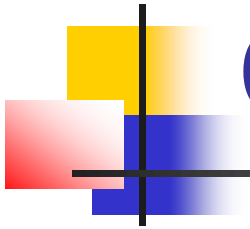
# Soubory v C

Pro práci se soubory je nutné užití knihovny <stdio.h>

Pro deklaraci proměnné typu soubor se používá klíčové slovo **FILE**

Proměnná která se používá ve funkci při práci se soubory je ovšem vždy typu ukazatel

**FILE** *soubor;

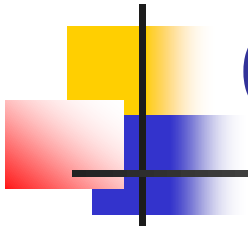


# Obecná práce se souborem

1) Otevření souboru

2) Čtení / zápis ze/do souboru

3) Uzavření souboru

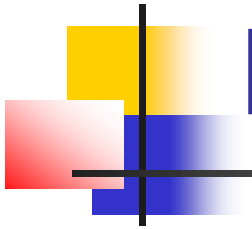


# Otevření souboru

Pomocí funkce FILE *fopen(char *jmeno, char * mod_otevreni)

Parametrem je název souboru a mód otevření souboru

Pokud se soubor nepodaří otevřít, vrací funkce fopen() NULL



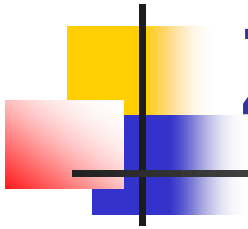
# Mód otevření souboru

“w” – pro zápis, pokud soubor neexistuje, je vytvořen
“r” – pro čtení
“a” – pro zápis nakonec souboru

Pokud se za písmenem uvede +, tak se jedná o mód pro čtení i zápis (a+, r+)

Pokud se uvede písmeno b, soubor se bere jako binární (rb, wb, ab)

Pokud neuvedeme typ souboru, chápe se tento jako textový (lze implicitně uvést i písmeno t)



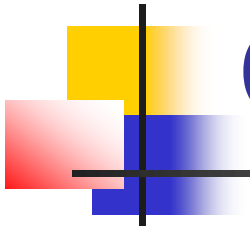
# Zápis do textového souboru

Pomocí funkce fprintf(FILE *soubor, format, parametry);

Užití je defakto shodné s printf()

Jediný rozdíl je v tom, že výstup se provede do souboru a ne na standardní výstup

# Čtení z textového souboru



Pomocí funkce fscanf(FILE *soubor, char *format, & promenna);

Nebo fgetc(FILE *soubor)

Na oddělení více různých záznamů se musí použít určitý znak jako oddělovač – zpravidla ‘\n’

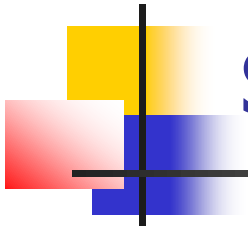
# Zjištění konce souboru

Pomocí tzv. EOF znaku – který přečte funce která z něj čte

Čteme funkcí fgetc(f) soubor po znaku a ukládáme do proměnné c, dokud nepřečteme znak konce souboru

```
char c;
while ( (c = fgetc(f)) != EOF)
        {
        printf("%c\n", c);
        }
```

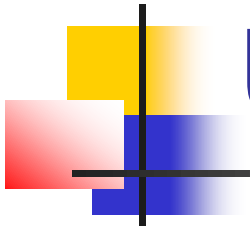
# Čtení a zápis z binárního souboru



Pomocí funkcí fread() a fwrite()

fread(buffer *char, int velikostPolozky, int pocetPolozek, FILE *soubor);

fwrite(buffer *char, int velikostPolozky, int pocetPolozek, FILE *soubor);



# Ukázka použití fwrite

```
int pole[10];
FILE *f1;
f1 = fopen("C:\\pokus1.bin","wb");
fwrite(pole, sizeof(int), 10, f1);
fclose(f1);
```