16. Práce se soubory v programovacích jazycích

Soubor

Pojmenovaná sada dat uložených na paměťovém médiu, se kterou lze pracovat jako s jedním celkem.

V programování slouží k uložení dat z programu nebo načtení dat do programu. Data ze souboru a do souboru putují přes streamy (proudy).

K souborům lze přistupovat dvěma způsoby a to **textově** nebo **binárně**. Rozdíl je v tom, jak se data čtou (zapisují).

Práce se soubory v jazyce JAVA (viz. PRM 17)

Pro práci se soubory v jazyce JAVA slouží třída **File**. Zároveň slouží jako manažer souborů. Na začátku programu je nutné importovat knihovnu **java.io.File**.

Popřípadě **java.io.FileWriter**, **java.io.FileReader,** a jiné knihovny, které jsou potřeba. Popřípadě rovnou importovat celou input/output knihovnu **java.io***.

Aby bylo možné pracovat se soubory (textovými či binárními), musí se vytvořit **Streamy** (proudy). Stream je proud dat z programu na nějaké místo na disku, konkrétně k souboru, s nímž pracuji.

Stream může být vstupní pro čtení, nebo výstupní pro zápis, existuje i varianta, že lze číst a zapisovat do souboru pomocí jednoho proudu.

Streamy lze dál dělit dle toho, jaká data v něm proudí. Bajtové, znakové, datové, standardní, objektové a proudy vyrovnávací paměti.

Po dokončení práce se soubory, by se měly uzavřít všechny proudy, které byly otevřené. Hlavně výstupní proud. Pokud se neuzavře výstupní proud, data se neuloží.

Třídy pro práci se soubory

Základní třídy pro práci s textovými soubory jako jsou ***.txt**, ***.java**, ***.sql** a mnoho dalších jsou **java.io.FileReader** pro čtení a **java.io.FileWriter** pro zápis.

Obou třídám v konstruktoru lze předat buď objekt třídy File, nebo řetězec s cestou k souboru.

Při vytváření instance třídy FileReader může nastat vžjimka **java.io.FileNotFoundException** pokud soubor neexistuje. Pokud soubor neexistuje a vytváří se instanci třídy FileWriter, tak se soubor vytvoří, ale může nastat **java.io.IOException** pokud soubor nelze vytvořit.

Práce se soubory v jazyce PHP

V jazyce PHP k vytvoření souboru slouží metoda fopen:

```
$soubor = fopen("soubor.txt", "w+");
```

Místo w+ lze použít další hodnoty:

- **a** – otevře soubor pro přidání, pokud soubor neexistuje, PHP jej vytvoří
- **a+** - otevře soubor pro přidávání a čtení, pokud soubor neexistuje, PHP jej vytvoří
- **r** – otevře soubor pouze pro čtení.
- **r+** - otevře soubor pro čtení a zápis
- **w** – otevře soubor pro zápis, původní data budou ztracena. Pokud soubor neexistuje, PHP jej vytvoří
- **w+** - otevře soubor pro zápis a čtení, původní data budou ztracena, pokud soubor neexistuje, PHP jej vytvoří
- **x** – vyhodí error, pokud soubor existuje. Nemusí se existence souboru ověřovat pomocí **file_exists()**;
- **c** – zamyká soubor. Nemusí se použít flock, aby se zamkl soubor a další request nemohl zapisovat do stejného souboru.

Zavření souboru se provádí stejnou funkcí jako vytvoření. Po dokončení práce se souborem je rozumné jej zavřít.

```
fclose(id_souboru);
fclose($soubor); //fopen vrací ID souboru
```

Práce se soubory v jazyce VBA

V jazyce VBA slouží pro práci se soubory funkce:

```
Open nazev_souboru For rezim As #cislo
```

Režimy pro práci se soubory:

- Input
  - Jednosměrný vstup ze souboru do programu (čtení dat ze souboru)
- Output
  - Jednosměrný výstup z programu do souboru (zápis dat do souboru přepsáním, pokud již existuje soubor se stejným jménem)
- Append
  - Jednosměrný výstup z programu do souboru (zápis konec - přidávání na konec)
- Random
  - Náhodný přístup, čtení i zápis
- Binary
  - Binární režim pro čtení i zápis

Příkaz CLOSE

```
Close #cislo
```

- Uzavírá (ukončuje přístup) otevřený soubor s uvedeným číslem
- Bez uvedení čísla uzavírá všechny otevřené soubory v programu
- Dokončuje poslední zápisy z vyrovnávací paměti – vyprázdní buffer
- Přesunuje celý soubor z operační paměti na trvalé paměťové médium

Funkce EOF

```
Eof (cislo)
```

- EOF = End Of File (konec souboru)
- Funkce vrací logickou hodnotu
  - True – zda bylo dosaženo konce souboru
  - False – zda nebylo dosaženo konce souboru

Příkaz Input

```
Input #cislo, seznam
```

- Seznam identifikátorů jednoduchých proměnných (nikoli polí)
- Jednotlivé identifikátory jsou odděleny čárkou
- Data ve čteném souboru musí svým charakterem a polohou odpovídat typu jednotlivých proměnných v seznamu

Příkaz LINE INPUT

```
Line Input #cislo, promenna
```

- Přečte ze souboru jeden řádek zakončený znakem s kódem 13 (Cariege Return – Konec řádku, Návrat vozíku)
- Čteny jsou všechny znaky v daném řádku (včetně mezer, čísel, uvozovek a jiných znaků)
- Znaky konce řádku – Chr(13) a Chr(10) – jsou z řetězce vypuštěny
- Další příkaz přečte následující řádek

Příkaz GET

```
Get #cislo, zaznam, promenna
```

- Obdobně jako LINE INPUT přečte jeden řádek (záznam), ale lze specifikovat libovolný záznam (řádek v databázi)
- Cislo – manipulační číslo otevřeného souboru
- Zaznam – pořadové číslo záznamu (řádku), který se má přečíst

Příkaz PRINT

```
Print #cislo, seznam
```

- Zapisuje data do otevřeného souboru
- Oddělovače v seznamu
  - ; – znak se zapíše na další pozici
  - , – znak se napíše do další zóny (jedna tisková zóna je 14 znaků)
- Každý příkaz PRINT bude psát na nový řádek
- Oddělovač použitý na konci seznamu způsobí, že další příkaz PRINT bude zapisovat na stejný řádek (potlačení přechodu na další řádek)

Příkaz WRITE

```
Write #cislo, seznam
```

- Výrazy jsou v seznamu odděleny čárkou
- Každý příkaz WRITE píše na nový řádek
- Řetězcové hodnoty jsou ve výstupním souboru ohraničeny uvozovkami
- Jednotlivé hodnoty jsou ve výstupním souboru odděleny čárkou

Příkaz PUT

```
PUT #cislo, zaznam, promenna
```

- Výstup do souboru

Příkazy GET a PUT jsou vhodné pro jednoduchou obsluhu databází, neboť dovolují manipulovat se záznamy v libovolném pořadí na rozdíl od sekvenčního přístupu.

Práce s adresáři

```
ChDir cesta 'nastaví pracovní cestu
ChDrive disk 'změna aktuální jednoty
MkDir cesta 'vytvoří adresář
```