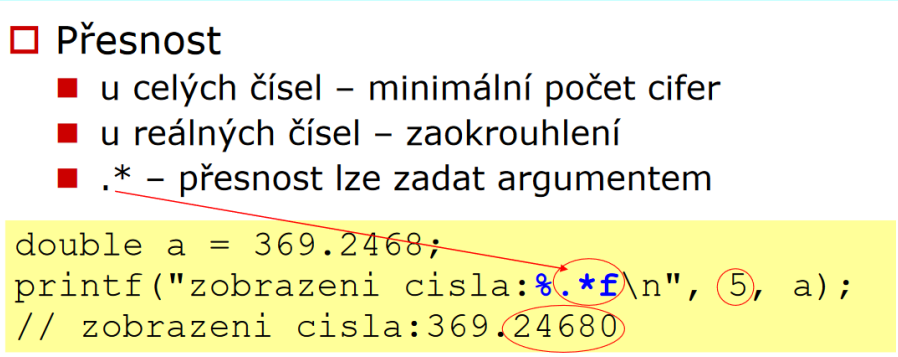
Textové soubory a standardní IO

- **Standardní vstup a výstup**
  - **Tři standardní soubory**
    - **stdin** – typicky vstup z klávesnice
    - **stdout** – typicky zápis na obrazovku
    - **stderr** – typicky zápis chyb na obrazovku
    - **Jsou otevřené se začátkem programu**
  - **Vstup a výstup**
    - **int getchar(void);**
      - Vstup po znacích ze stdin
      - Vrací znak v typu int (!)
      - Vrací EOF, pokud dojde na konec souboru (to se dá detekovat while loopem)
    - **int putchar(int c);**
      - Výstup po znacích na stdout
      - Zapisuje znak uložený v int (symetrické ke getchar)
      - Vrací EOF, pokud při zápisu dojde k chybě
    - **Formátovaný vstup - funkce scanf()**
      - **int scanf(const char *format, ...);**
        - Formátovaný vstup ze stdin
        - Během čtení přeskakuje bílé znaky
      - **Formátovací řetězec**
        - Vzor, očekávaný na vstupu
      - **Proměnný počet argumentů**
        - Nutno předávat odkazem!
      - **POZOR!**
        - scanf("%s", buffer) – hrozí čtení za hranicí alokované paměti!
      - Můžeme použít například na přímé čtení numerických hodnot
      - **Formátovací značka**
        - %[šířka][modifikátor]konverze
        - Význam podobný jako u printf()
        - %*konverze – přečti a zahoď
      - **Návratová hodnota**
        - Počet úspěšně přečtených parametrů
        - Pokud se vstup neshoduje se vzorem, funkce ukončí čtení těsně před prvním nerozpoznaným znakem ➔ funkce vrátí menší výsledek, než se čekalo
        - Pokus o čtení za koncem souboru ➔ vrací EOF
    - **Formátovaný výstup - funkce printf()**
      - **int printf(const char *format, ...);**
        - Formátovaný výstup do stdout
      - **Formátovací řetězec**
        - Libovolný text + formátovací značky
      - **Proměnný počet argumentů**
        - **Musí odpovídat formátovacím značkám**
          - počet
          - pořadí

        - typy
    - Umožňuje přímé vypisování numerických hodnot
    - **Formátovací značka**
        - **%[příznak][šířka][.přesnost][modifikátor]konverze**
        - Mezi znakem % a konverzí mohou být nepovinné parametry
        - Konverze je povinný parametr (d, f, s, c, ..)
        - Modifikátor
            - hh – short short (char), h – short, l – long, ll – long long, L – long double
        - Příznak
            - + zarovnání doprava, – zarovnání doleva
        - *Příklad*: Ukázky různých konverzí

Hodnota:	Konverze:	Výstup:	Poznámka:
3.141592	%7.3f	□□3.142	mezery zleva
3.141592	%-7.3	3.142□□	mezery zprava
369.24	%+11.3E	+3.692E+002	
369.24	%+9.3E	+3.692E+002	
369.24	%+13.3E	□□+3.692E+002	
3.6	%4.0f	□□□4	
Ahoj!	%2s	Ahoj!	
Ahoj!	%.2s	Ah	

        - 


```
double a = 369.2468;
printf("zobrazeni cisla:%.*f\n", 5, a);
// zobrazeni cisla:369.24680
```

- **Vstup po řádcích -** char *gets(char *str);
    - Přečte celý řádek ukončený '\n' a uloží jej do bufferu str
    - Čte i bílé znaky, znak '\n' se neukládá
    - Automaticky vkládá '\0' na konec
    - **POZOR! raději NEPOUŽÍVAT! NEBEZPEČNÉ!**
    - Neumožňuje omezit počet čtených znaků
    - Hrozí zápis mimo alokovaný buffer
    - Raději používat fgets(), která je bezpečná
- **Výstup po řádcích** - int puts(char *str);
    - Vytiskne řetězec str a odřádkuje
    - Na rozdíl od printf neinterpretuje řetězec
    - Ekvivalentní volání
        - printf("%s\n", str);
    - Při neúspěchu zápisu vrací EOF

- **Vstup ze souboru, výstup do souboru**
    - **Textové soubory**

        - Knihovní funkce interpretují konce řádků podle zvyklostí OS (CR LF x LF x CR)
    - **Binární soubory**
        - Knihovní funkce neinterpretují konce řádků
        - Knihovní funkce pracují přesně s daty, která jsou v souboru uložena
        - Jiným principiálním způsobem se **textové a binární soubory neliší**
    - **Jazyk C**
        - Nezná datový typ soubor
        - Nemá syntaktické prostředky pro práci se soubory
        - Práce se soubory pomocí knihovních funkcí
        - **FILE ***
            - Typ ze stdio.h
            - FILE – struktura popisující soubor, nelze s ní pracovat přímo
            - Práce výhradně přes ukazatel a knihovní funkce
        - **Funkce -** FILE *fopen(const char *jmeno, const char *mod);
            - Otevře soubor zadaného jména
                - Pokud se nepovede otevřít – vrací NULL
            - mod – způsob otevření souboru, řetězec (!)
                - "r" textový soubor pro čtení
                - "w" textový soubor pro zápis nebo pro přepsání
                - "a" textový soubor pro připisování na konec
                - "r+" textový soubor pro čtení a zápis
                - "w+" textový soubor pro čtení, zápis nebo přepsání
                - "a+" textový soubor pro čtení a zápis na konec
                - **+ znamená že když neexistuje, tak se vytvoří**
        - **Funkce -** int fclose(FILE *file);
            - Uzavře soubor.
            - Pokud se nepovede uzavřít – vrací **EOF**.
            - (Ne)uzavírání souborů
                - Std. knihovna musí zapsat všechny vnitřní buffery.
                - Předčasné ukončení (např. funkcí abort())➔ poškození souborů.
                - Slušný program uzavírá své otevřené soubory a nespoléhá na operační systém
        - Standardní vstup/výstup
            - **int getchar(void);**
            - **int putchar(int c);**
        - Souborový vstup/výstup [HePa13, str. 73]
            - **int getc(FILE *file);**
            - **int putc(int c, FILE *file);**
            - Pracují s otevřenými soubory.
            - Práce s neotevřeným souborem ➔ chyba
        - **ftell(File *file)**
            - Vrací pozici kurzoru
        - **fseek(File *file, offset, _Cur)**
            - Nastaví pozici kurzoru vzhledem na offset k hodnotě _Cur (ta může být konstanta: začátek souboru, konec souboru, aktuální pozice kurzoru)
        - **Formátovaný vstup a výstup**
            - Standardní vstup/výstup
                - **int scanf(const char *format, ...);**
                - **int printf(const char *format, ...);**

- Souborový vstup/výstup
  - **int fscanf(FILE *file, const char *format, ...);**
  - **int fprintf(FILE *file, const char *format, ...);**

- **Vstup a výstup řádků**
  - **char * fgets(char *str, int max, FILE *file);**
    - Čte nejvýše max znaků do konce řádku.
    - Znak '\n' se ukládá (!).
    - Pokud nenačte celý řádek, nepřejde na nový, ale příště bude pokračovat, kde skončila.
    - Při úspěchu vrací ukazatel na řetězec, jinak NULL.
  - **int fputs(char *str, FILE *file);**
    - Zapíše řetězec do souboru a neodřádkuje.
    - Při neúspěchu vrací **EOF.**
- **Testování chyb**
  - **I/O funkce vracejí NULL nebo EOF pro detekci chyby.**
  - **int ferror(FILE *file);**
    - Vrací nenulovou hodnotu, pokud poslední operace se souborem způsobila chybu.

```
int ch = fgetc(file);
if(ferror(file))
{
  printf("Chyba souboru\n");
  break;
}
```

  - **int errno;**
    - Globální proměnná z <errno.h>
    - Obsahuje kód chyby poslední I/O operace.
    - Tuto proměnnou nastavují všechny I/O operace.
  - **char * strerror (int errnum);**
    - Funkce z <string.h>
    - Vypíše anglicky chybové hlášení.
    - Parametrem je kód chyby (errno).

```
// Pokusí se otevřít soubor zadaného jména.
// Pokud dojde k chybě, vypíše hlášení
FILE *testOpen(const char *name,const char *mod)
{
  FILE *f = fopen(name, mod);
  int error = errno;
  if(f == NULL)
  {
    fprintf(stderr, "%s", strerror(error));
    exit(EXIT_FAILURE); // TAKTO NE!
  }
  return f;
}
```