KATEDRA INFORMATIKY
PŘÍRODOVĚDECKÁ FAKULTA
UNIVERZITA PALACKÉHO V OLOMOUCI

Práce s binárními soubory

9. cvičení

Jiří Zacpal

KMI/ZP2 – Základy programování 2

Binární soubory

- mohou mít libovolnou strukturu
- data jsou v nich zapsána ve stejné podobě, v jaké existují za běhu programu v paměti
  - nelze je vytvářet, číst, editovat v běžných textových editorech (zobrazené znaky většinou nedávají smysl)
  - pro uložení dat je potřeba méně místa než v případě textových souborů
  - pracuje se s nimi rychleji než s textovými soubory (nejsou nutné konverze textu na čísla)
- při práci se souborem otevřeným jako binární, neprovádějí knihovní I/O funkce žádné dodatečné akce
- nejsou vždy přenositelné (různé velikosti typů)

Čtení bloku dat

- funkce pro čtení bloku dat
```
size_t fread(void *kam, size_t rozmer, size_t
pocet, FILE *f);
```
  - `f` je datový proud, ze kterého se čte
  - `rozmer` je velikost jedné položky
  - `pocet` udává počet položek
  - `kam` je adresa paměti, do které se data ukládají
  - funkce vrací počet úspěšně přečtených položek
- příklad:
```
#define VELIKOST_BLOKU 10
...
int data[VELIKOST_BLOKU];
FILE *fr = fopen("a.dat", "rb");
fread(data, sizeof(int), VELIKOST_BLOKU, fr);
```

Zápis bloku dat

- funkce pro čtení bloku dat

```
size_t fwrite(void *odkud, size_t rozmer, size_t pocet, FILE *f);
```

  - `f` je datový proud, do kterého zapisujeme
  - `rozmer` je velikost jedné položky
  - `pocet` udává počet položek
  - `odkud` je adresa dat, která chceme zapisovat
  - funkce vrací počet úspěšně zapsaných položek

- příklad:

```
#define VELIKOST_BLOKU 10
...
int data[VELIKOST_BLOKU];
FILE *fw = fopen("out.dat", "wb");
...
fwrite(data, sizeof(int), VELIKOST_BLOKU,fw);
```

Posun pozice v souboru

- někdy je třeba zapisovat (resp. číst) data na konkrétní místo souboru (resp. z konkrétního místa souboru)
- funkce pro posun pozice v souboru
  ```
  int fseek(FILE *f, long posun, int odkud);
  ```
  - `posun` udává počet bytů (může být i záporné číslo), o kolik se má pozice změnit směrem ke konci souboru
  - `odkud` udává výchozí pozici pro výpočet posunu
    - `SEEK_SET` (posouvá se od začátku souboru)
    - `SEEK_CUR` (posouvá se od aktuální pozice)
    - `SEEK_END` (posouvá se od konce souboru)
  - funkce vrací nulu v případě úspěchu, nenulovou hodnotu v ostatních případech (posun mimo rozsah souboru apod.)

Zjištění pozice

- funkce pro zjištění pozice v souboru
  ```
  long ftell(FILE *f);
  ```
  - funkce vrací aktuální posunutí pozice od začátku souboru v bytech
- příklad:
  ```
  /* navrat na puvodni misto*/
  akt_pos = ftell(f);
  fseek(f, 0L, SEEK_SET);
  if (hledej(f, "ahoj")==NULL)
          fseek(f, akt_pos, SEEK_SET);
  ```
- žádná vstupní operace (např. `fread`) nesmí přímo následovat po výstupní operaci (např. `fwrite`) nebo naopak bez předchozího volání funkce `fseek`. Řešením je volání `fseek` s nulovou změnou pozice.
  ```
  fseek(f, 0L, SEEK_CUR);
  ```

Další užitečné funkce 1/2

- funkce pro přesměrování proudu
```
FILE *freopen(const char* name, const char* mode,
FILE *f);
```
  - otevře soubor `name` v režimu `mode` a přesměruje do něj existující proud `f`, který zavře
  - pokud je `f` standardní proud (`stdin`, `stdout` nebo `stderr`), stane se jméno tohoto standardního proudu synonymem pro nově otevřený proud
  - používá se pro přesměrování std. proudu do souboru
  - v případě úspěchu vrací ukazatel na vytvořený proud, jinak je návratovou hodnotou `NULL`
- funkce pro přejmenování souboru
```
int rename(const char *old_name, const char
*new_name);
```
  - v případě úspěchu vrací nulu, jinak nenulovou hodnotu

Další užitečné funkce 2/2

- funkce pro smazání souboru
  `int remove(const char *name);`
  - v případě úspěchu vrací nulu, jinak nenulovou hodnotu
- funkce pro vytvoření dočasného souboru
  `FILE *tmpfile();`
  - otevře v režimu `"w+b"` pomocný soubor, který se po uzavření proudu automaticky smaže
  - vrací ukazatel na vytvořený soubor nebo `NULL`
- funkce pro generování jmen pomocných souborů
  `char *tmpname(char *str);`
  - vygeneruje unikátní textový řetězec a uloží jej do `str`, návratouvou hodnotou je ukazatel na vytvořený text
  - pokud má parametr `str` hodnotu `NULL`, alokuje funkce paměť pro výsledný textový řetězec

Změny bufferu

- funkce pro přiřazení konkrétního bufferu
  `void setbuf(FILE *f, char *buf);`
  - volá se ihned po otevření proudu `f` pro nastavení jeho bufferu na paměť `buf`, která musí mít velikost alespoň `BUFSIZ` bytů
  - pokud má parametr `buf` hodnotu `NULL`, dojde k vypnutí bufferování (I/O operace se budou provádět přímo)
- rozšířená verze předcházející funkce
  `void setvbuf(FILE *f, char *buf, int mode, size_t size);`
  - parametr `mode` umožňuje nastavit režim bufferování
    - `_IOFBF` – data se načítají do zaplnění bufferu
    - `_IOLBF` – načítání do konce řádky nebo do zaplnění bufferu
    - `_IONBF` – vypnutí bufferování
  - parametr `size` umožňuje zadat velikost bufferu

Úkol

Prostudujte si zdrojový kód v připraveném souboru a dopište funkci `int uprav_data(char *nazev)`. Tato funkce by měla číst vektory - trojice čísel (používejte definovanou konstantu DIMENZE) typu double z binárního souboru nazev. Pro každý přečtený vektor funkce vypočítá jemu odpovídající vektor o jednotkové velikosti (směr a orientace vektoru zůstanou zachovány) a upraveným vektorem přepíše stará data v souboru. Funkce poté pokračuje čtením dalšího vektoru, dokud není celý obsah datového souboru zpracován.

Úkol – řešení

```
int uprav_data(char *nazev)
{
        FILE *f;
        double data[DIMENZE],a;
        if ((f = fopen(nazev, "r+b")) == NULL) return 1;
        while((fread(data, sizeof(double), DIMENZE , f))==DIMENZE)
    {
            a=vel_vektor(data);
            if(a==0)break;
            for(i=0;i<DIMENZE;i++) data[i]=data[i]/a;
            fseek(f,-DIMENZE*sizeof(double),SEEK_CUR);
            fwrite(data,sizeof(double),DIMENZE,f);
            fseek(f,0L,SEEK_CUR);
    }
    fclose(f);
    return 0;

}
```

Bodovaný úkol

Prostudujte si zdrojový kód v připraveném souboru a dopište funkci `int vyhledej(char* soubor, char *kriteria, ...)`. Tato funkce by měla vyhledat v binární databázi `soubor` všechny osoby odpovídající daným vlastnostem a vypsat je pomocí funkce void vypis(osoba o) na obrazovku. Vlastnosti osob, které budou při vyhledávání zkoumány, určují jednotlivé znaky řetězce kriteria (znak "*j*" pro jméno osoby, "*p*" pro příjmení, "*n*" pro datum narození, "*d*" pro den narození, "*m*" pro měsíc narození, "*r*" pro rok narození, "*P*" pro pohlaví a "*s*" pro stav), za kterým pak následují vyhledávané hodnoty těchto vlastností.