IJC: DU2

```
Jazyk C                         DU2                        21.3.2012
--------------------------------------------------------------------

                          Domácí úkol č.2


Termín odevzdání: 24.4.2012                          (Max. 15 bodů)


1) a) V jazyku C napište program "tail.c", který  ze zadaného
   vstupního souboru vytiskne posledních  10 řádků. Není-li
   zadán vstupní soubor, čte ze stdin. Je-li programu zadán
   parametr -n číslo, nebude se tisknout  10  řádků  ale  tolik,
   kolik je zadáno parametrem 'číslo'. Pokud je programu zadán
   parametr -n +číslo, bude se tisknout od řádku 'číslo'
   (první řádek má číslo 1) až do konce souboru.
   Případná chybová hlášení tiskněte do stderr. Příklady:

     tail  soubor
     tail  -n 20 <soubor
     tail  -n +3

   [Poznámka: výsledky by měly být stejné jako u POSIX příkazu tail]

   Je povolen implementační limit na délku řádku (např. max 1024 znaků),
   v případě prvního překročení mezí hlaste chybu na stderr (řádně otestujte).

   b) Napište stejný program jako v a) v C++ s použitím standardní
   knihovny C++. Jméno programu: "tail2.cc". Tento program
   musí zvládnout řádky libovolné délky a jejich libovolný počet,
   jediným možným omezením je volná paměť.
   Použijte funkci
      std::getline(istream, string)
   a vhodný kontejner (např. std::deque<string>).
   [Poznámka: je jednodušší než předchozí varianta v C]
(5b)


2) Přepište následující C++ program do jazyka ISO C

    // stl-map-example.cc
    // příklad použití STL kontejneru map<>
    // program počítá četnost slov ve vstupním textu
    // slovo je cokoli oddělené "bílým znakem" === isspace

    #include <string>
    #include <iostream>
    #if 1       // zkuste si i variantu s 0
    # include <tr1/unordered_map>
      typedef std::tr1::unordered_map<std::string,int>  map_t;
    #else
    # include <map>
```

```
  typedef std::map<std::string,int>  map_t;
#endif
typedef map_t::iterator            mapiter_t;

int main() {
  std::string word;
  map_t m; // asociativní pole - indexem je slovo

  while( std::cin >> word )  // čtení slova
    m[word]++;               // počítání výskytů slova

  // tisk
  for(mapiter_t i=m.begin(); i!=m.end(); ++i)
    std::cout << i->first <<"\t"<< i->second <<"\n";
  //             slovo (klíč)         počet (data)
}
```

Výstupy obou programů musí být stejné (kromě pořadí a příliš dlouhých slov).
Výsledný program se musí jmenovat "wordcount.c".

Veškeré operace s tabulkou budou v samostatné knihovně (vytvořte statickou
i dynamickou/sdílenou verzi). V knihovně musí být každá funkce ve
zvláštním modulu - to umožní případnou výměnu hash_function() ve vašem
staticky sestaveném programu (vyzkoušejte si to).

Knihovna s tabulkou se musí jmenovat
"libhtable.a" (na Windows je možné i "htable.lib") pro statickou variantu,
"libhtable.so" (na Windows je možné i "htable.dll") pro sdílenou variantu
a rozhraní "htable.h".

Podmínky:

- Implementace musí být dynamická (malloc/free) a musíte zvládnout správu paměti v C (použijte valgrind, nebo jiný podobný nástroj).

- Asociativní pole implementujte nejdříve prototypově jednoduchým seznamem a potom tabulkou (hash table). Odevzdává se řešení s tabulkou.

  Vhodná rozptylovací funkce pro řetězce je podle literatury:

```
unsigned int hash_function(const char *str, unsigned htable_size) {
  unsigned int h=0;
  unsigned char *p;
  for(p=(unsigned char*)str; *p!='\0'; p++)
      h = 31*h + *p;
  return h % htable_size;
}
```

  její výsledek určuje index do tabulky.

- Tabulka je struktura obsahující pole seznamů a velikost:

```
 +-------------+
 | htable_size |
 +-------------+
 +---+
 |ptr|-->[key,data,next]-->[key,data,next]-->[key,data,next]--|
 +---+
 |ptr|-->[key,data,next]-->[key,data,next]--|
 +---+
 |ptr|--|
```

```
     +---+

  Položka htable_size je velikost následujícího pole ukazatelů
  (použijte C99: "flexible array member"). V programu zvolte vhodnou velikost pole
  a v komentáři zdůvodněte vaše rozhodnutí.

- Napište funkce
    t=htable_init(size)     pro vytvoření a inicializaci tabulky
    ptr=htable_lookup(t,key)    vyhledávání - viz dále
    htable_foreach(t,function)  volání funkce pro každý prvek
    htable_remove(t,key)        vyhledání a zrušení položky
    htable_clear(t)             zrušení všech položek v tabulce
    htable_free(t)              zrušení tabulky (volá clear)

  kde t     je ukazatel na tabulku (typu htable_t *),
      b     je typu bool,
      ptr   je ukazatel na záznam (položku tabulky),
      function je funkce s parametry (key,value)

- Vhodně zvolte typy parametrů funkcí.

- Zvažte, které z uvedených operací bude vhodné udělat inline a které ne.

- Záznam [key,data,next] je typu
      struct htable_listitem
  a obsahuje položky:
      key .... ukazatel na dynamicky alokovaný řetězec,
      data ... počet výskytů a
      next ... ukazatel na další záznam

- Funkce  htable_foreach(t,function)  volá zadanou funkci pro každý prvek
  tabulky, obsah tabulky nemění. (Vhodné např. pro tisk obsahu.)

- Musíte také napsat funkci
    struct htable_listitem * htable_lookup(htable_t *t, const char *key);
  která v tabulce t vyhledá záznam odpovídající řetězci key a
    pokud jej nalezne vrátí ukazatel na záznam
    pokud nenalezne, automaticky přidá záznam a vrátí ukazatel
  Poznámka: Dobře promyslete chování této funkce k parametru key.

- Pokud htable_init nebo htable_lookup nemohou alokovat paměť, vrací NULL

- Napište funkci
    int fgetword(char *s, int max, FILE *f);
  která čte jedno slovo ze souboru f do zadaného pole znaků
  a vrátí délku slova (z delších slov vrátí prvních max-1 znaků,
  a zbytek přeskočí). Funkce vrací EOF, pokud je konec souboru.
  Umístěte ji do zvláštního modulu "io.c" (nepatří do knihovny).
  Poznámka: Slovo je souvislá posloupnost znaků oddělená isspace znaky.

Omezení: řešení v C může tisknout jinak seřazený výstup
  a je povoleno použít implementační limit na maximální
  délku slova (zvolte 255 znaků), delší slova se ZKRÁTÍ a program
  při prvním delším slovu vytiskne varování na stderr.

Poznámka: vhodný soubor pro testování je například seznam slov
          v souboru /usr/share/dict/words
          nebo texty z http://www.gutenberg.org/

[Pokud se někdo nudí, napíše si varinatu tabulky s automatickým
```

```
    zvětšováním/zmenšováním velikosti tak, aby průměrná délka seznamů
    nepřesahovala rozumnou mez (experimentálně zjistit). Toto řešení se
    neodevzdává ani nehodnotí, ale může se hodit po zkoušce na přidání
    několika bodů...]

(10b)

Napište soubor Makefile tak, aby příkaz make vytvořil programy
"tail", "tail2", "wordcount", "wordcount-dynamic" a knihovny "libhtable.a",
"libhtable.so" (nebo "htable.DLL").
Program "wordcount" musí být staticky sestaven s knihovnou "libhtable.a".
Program "wordcount-dynamic" musí být dynamicky sestaven s knihovnou "libhtable.so".
Tento program otestujte se stejnými vstupy jako u staticky sestavené verze.

Porovnejte efektivitu obou implementací (příkazy time a gprof)
a zamyslete se nad výsledky (pozor na vliv vyrovnávacích paměťí)
Použijte profiler gprof a prozkoumejte kde váš program tráví nejvíce času.

Poznámky:
 - v 1b) maximálně využívejte standardní knihovny C++
 - čtěte pokyny pro vypracování domácích úkolů (viz dále)

-----------------------------------------------------------------

Obecné pokyny pro vypracování domácích úkolů

*  Pro úkoly v jazyce C používejte ISO C99 (soubory *.c)
   Pro úkoly v jazyce C++ používejte ISO C++ (soubory *.cc)
   Použití nepřenositelných konstrukcí není dovoleno.

*  Úkoly zkontrolujte překladačem například takto:
      gcc -std=c99 -pedantic -Wall priklad1.c
      g++ -std=c++98 -pedantic -Wall priklad.cc
   Místo gcc můžete použít i jiný překladač - podle vašeho prostředí.
   V  souvislosti s tím napište do poznámky na začátku
   souboru jméno a verzi překladače, kterým byl program přeložen
   (implicitní je GCC na počítači merlin).

*  Programy  pište, pokud je to možné, do jednoho zdrojového
   souboru. Dodržujte předepsaná jména souborů.

*  Na začátek každého souboru napište poznámku, která bude
   obsahovat jméno, fakultu, označení příkladu a datum.

* Úkoly je nutné zabalit programem zip takto:
       zip xnovak99.zip *.c *.cc *.h Makefile

  Jméno xnovak99 nahradíte vlastním. Formát souboru bude ZIP.
  Archiv neobsahuje adresáře.  Každý si zkontroluje obsah ZIP archivu jeho
  rozbalením v prázdném adresáři a napsáním "make".

* Posílejte pouze nezbytně nutné soubory -- ne *.EXE !

* Řešení se odevzdává elektronicky v IS FIT

* Úkoly neodevzdané v termínu budou za 0 bodů.

* Opsané úkoly budou hodnoceny 0 bodů pro všechny zůčastněné
  a to bez výjimky (+ bonus v podobě návštěvy u disciplinární komise).
```

Poslední modifikace: 20. března 2012
Pokud naleznete na této stránce chybu, oznamte to dopisem na adresu peringer AT fit.vutbr.cz