МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний технічний університет
«Харківський політехнічний інститут»

МЕТОДИЧНІ ВКАЗІВКИ
до лабораторної роботи
«Обробка текстових файлів з використанням стандартних засобів мови Сі»

з курсу «Програмування»
для студентів напряму 6.040302 – Інформатика
і курсу «Програмування та алгоритмічні мови»
для студентів напряму 6.040303 – Системний аналіз

Затверджено редакційно-видавничою
радою університету,
протокол № 1 від 04.06.14.

Харків
НТУ «ХПІ»
2014

Методичні вказівки до лабораторної роботи «Обробка текстових файлів з використанням стандартних засобів мови Сі» з курсу «Програмування» для студентів напряму 6.040302 – Інформатика і курсу «Програмування та алгоритмічні мови» для студентів напряму 6.040303 – Системний аналіз / Уклад. М. І. Безменов, О. М. Безменова. – Х. : НТУ «ХПІ», 2014. – 21 с.

Укладачі: М. І. Безменов,
О. М. Безменова

Рецензент І. П. Гамаюн

Кафедра системного аналізу і управління

ВСТУП

Текстові файли є одним з видів файлів, призначених для зберігання текстової та числової інформації. Використання файлів цього виду в програмах і одержання навичок читання даних з таких файлів і запису інформації у них відносяться до азів програмування.

Метою даної лабораторної роботи є освоєння методики використання текстових файлів у програмах, написаних мовою Сі.

1. ТЕОРЕТИЧНІ ОСНОВИ

Зовнішнє ім’я файлу зазначається з урахуванням методики його подання для операційної системи. Воно може бути подане в повному вигляді (диск, шлях, ім’я: `"C:\\Users\\a.txt"`) або в скороченому – з урахуванням принципів умовчання, що закладені в роботу операційної системи, а саме:

- якщо не зазначене ім’я диска, то розглядається активний диск;
- якщо не зазначений шлях до файлу, то розглядається поточна папка (поточний каталог);
- якщо шлях починається не із символу `\`, то відлік шляху починається з поточної папки (слід пам’ятати про те, що в мові Сі символ `\` є управляючим і в разі необхідності його подання він подвоюється: `'\\'`).

1.1. Потокове введення/виведення

На рівні потокового введення/виведення обмін даними здійснюється за байтами. Такі введення/виведення можливі для друкувального пристрою, дисплея і дискових файлів. Функції бібліотеки введення/виведення, дозволяючи обробляти дані різних форматів, забезпечують при цьому буферизацію процесів введення і виведення.

Потік – це файл разом із засобами буферизації. При роботі з потоком можна:

- відкривати і закривати потоки (зв’язувати покажчики на потоки з конкретними файлами);
- вводити і виводити символ, рядок, відформатовані дані, порцію даних довільної довжини;

- аналізувати помилки потокового введення/виведення і умову досягнення кінця потоку (файлу);
- керувати буферизацією;
- керувати покажчиком поточної позиції у потоці.

Якщо потік відкритий, то з ним зв'язується так званий покажчик (індикатор) поточної позиції. Читання/записування даних здійснюється починаючи з того байту файлу, на який указує цей покажчик. При цьому покажчик поточної позиції автоматично «переміщається» по файлу на ту кількість байт, що відповідає зчитаному/записаному обсягу інформації. Результатом є те, що наступний сеанс читання/записування здійснюватиметься з орієнтацією на нове положення покажчика поточної позиції.

1.2. Обробка текстових файлів

Бібліотечні функції уведення/виведення підключаються через заголовний файл `stdio.h`, який, у свою чергу, автоматично підключається при підключенні файлу `iostream`.

Одним з видів файлів, якими може оперувати програма, написана мовою Сі, є текстові файли, що складаються з символьних рядків змінної довжини, завершуваних спеціальною комбінацією символів, називаною «кінцем рядка». Комбінація «кінець рядка» складається з двох символів – «повернення каретки» (`'\xD'`, десятковий код 13) і «переведення рядка» (`'\xA'`, десятковий код 10). Крім фізичного кінця файлу, як ознака кінця файлу в текстових файлах сприймається символ «кінець файлу» (`'\x1A'`, десятковий код 26). Такий файл можна підготувати будь-яким текстовим редактором, у тому числі редактором, що вбудований в інтегроване середовище Visual Studio.

Для забезпечення доступу до файлового потоку використовується покажчик на тип `FILE`, що визначений у заголовному файлі `stdio.h`:

```
FILE *ім'я_покажчика;
```

Перед тим, як почати роботу з потоком, його потрібно відкрити, для чого застосовується функція `fopen()`, яка зв'язує потік з конкретним файлом, повертаючи при цьому значення покажчика на потік:

```
FILE *fopen(char *ім'я_файлу, char *режим_відкриття);
```

Параметри ім'я_файлу і режим_відкриття задаються або звичайними символьними масивами, або покажчиками на початок області пам'яті, у якій зберігається рядок символів.

Стандартними режимами відкриття для текстових файлів є:

- "w" – відкриття нового текстового файлу для записування;
- "r" – відкриття існуючого текстового файлу для читання;
- "a" – текстовий файл відкривається (або створюється, якщо він відсутній) для дозаписування в кінець;
- "w+" – новий текстовий файл відкривається для записування, читання і подальшого багаторазового виправлення з можливістю дозаписування в кінець файлу;
- "r+" – існуючий текстовий файл відкривається як для читання, так і записування;
- "a+" – текстовий файл відкривається або створюється (якщо він відсутній) як для читання, так і записування, причому записування здійснюється завжди в кінець файлу незалежно від положення покажчика поточної позиції.

Відкриття існуючого файлу в режимі "w" або "w+" забезпечує його повне відновлення.

Якщо потік відкритий у текстовому режимі, то прочитана з нього послідовність символів «повернення каретки» і «переведення рядка» перетворюється на символ нового рядка '\n' (значення 10). При записуванні в потік здійснюється зворотне перетворення.

Текстовий режим можна явно позначити буквою t (наприклад, "r+t").

Приклади відкриття текстових файлів:

```
FILE *f1, *f2, *f3;
    // Текст програми
f1 = fopen("a.txt", "rt");
char OutFileName[] = "C:\\USERS\\result.txt";
f2 = fopen(OutFileName, "w");
char *FileName = new char[261];
cout << "\nInput the name of new file\n";
cin.getline(FileName, 261);
f3 = fopen (FileName, "w");
```

При помилках у відкритті потоку покажчик набуває значення NULL. Тому необхідно здійснювати перевірку значення покажчика, що можна зробити, наприклад, так:

```
FILE *f;
if ((f = fopen("a.txt", "r")) == NULL)
{         // Дії при помилці
}
```

Функція fopen() є небезпечною. Замість неї рекомендується використовувати функцію fopen_s(), що має такий формат:

```
errno_t fopen_s(FILE** покажчик_файлу,
        char * ім'я_файлу, char *режим_відкриття);
```

У порівнянні з функцією fopen() у цієї функції є ще один параметр (перший), який є покажчиком на покажчик і **отримує покажчик на відкритий файл** або значення NULL, якщо файл не буде відкритий. Функція повертає 0 при нормальному завершенні і код помилки – при помилці відкриття файлу.

Закриття потоку здійснюється функцією

```
int fclose(FILE * покажчик_на_потік);
```

Повторне відкриття потоку потрібно передувати його закриттям.

Можливе також використання функції

```
int _fcloseall(void);
```

яка закриває всі відкриті потоки, за винятком стандартного введення, стандартного виведення результатів обчислень і стандартного виведення помилок.

При відкритті файлу в режимах "w", "w+", "r", "r+" покажчик поточної позиції установлюється на початок файлу, а при відкритті в режимах "a" і "a+" – на його кінець.

Для читання (уведення) одного символу із файлу можна використовувати функцію

```
int fgetc(FILE * покажчик_на_потік);
```

або макрос

```
int getc(FILE * покажчик_на_потік),
```

а для записування (виведення) одного символу у файл – функцію

```
int fputc(int символ, FILE * покажчик_на_потік);
```

або макрос

```
int putc(int символ, FILE * покажчик_на_потік).
```

Як перший параметр функцій `fputc()` і `putc()` може бути використаний будь-який вираз, результатом обчислення якого є деякий символ (у тому числі змінна або константа).

При помилках уведення/виведення, а також при виявленні ознаки кінця файлу перелічені вище функції повертають значення `EOF`.

Функція

```
char* fgets(char * рядок, int макс_кількість,
                FILE * покажчик_на_потік);
```

здійснює читання (уведення) не більше (макс_кількість – 1) символів з файлу в рядок, що задається першим параметром (звичайний масив або покажчик на область пам'яті для зберігання символів). Якщо при цьому буде прочитаний символ нового рядка `'\n'`, то він переноситься в символьний масив, що адресується першим параметром) і читання припиняється. За останніми занесеним у рядок символом записується ознака закінчення рядка (символ `'\0'`). При нормальному завершенні читання, функція повертає значення першого параметра (покажчик на перший елемент символьного масиву). У випадку помилки читання (у тому числі при досягненні кінця файлу, якщо жоден символ з файлу не прочитаний) функція повертає значення `NULL`, не змінюючи вміст масиву, що адресується першим параметром.

Для записування (виведення) у файл рядкових даних може бути використана функція

```
int fputs(const char * рядок, FILE * покажчик_на_потік);
```

При виконанні цієї функції символ `'\0'` у файл не переноситься і не замінюється символом `'\n'`. У разі нормального завершення функція повертає останній виведений символ, а при помилці – значення `EOF`.

Крім того, використовуються такі функції форматного введення і виведення відповідно:

```
int fscanf(FILE * покажчик_на_потік,
           const char * форматний_рядок, список_покажчиків);

int fprintf(FILE * покажчик_на_потік,
           const char * форматний_рядок, список_значень);
```

Форматне введення/виведення при роботі з файлами базується на тих же принципах, що і звичайне консольне введення/виведення.

Другий параметр двох останніх функцій (форматний_рядок) може вміщувати як звичайні символи, так і специфікації перетворення при введенні та виведенні, що задаються деякими послідовностями символів, яким передує символ `%`.

Зокрема, слідом за символом `%` у форматному рядку функції `fscanf()` можуть йти такі символи (або послідовності символів), що визначають специфікації перетворення:

- `d` – уведення десяткового цілого (відповідний покажчик у списку_покажчиків повинен мати тип **`int`** `*`);
- `i` – уведення десяткового, вісімкового чи шістнадцяткового цілого (відповідний покажчик у списку_покажчиків повинен мати тип **`int`** `*`);
- `f` – уведення дійсного значення типу **`float`** (відповідний покажчик у списку_покажчиків повинен мати тип **`float`** `*`);
- `lf` – уведення дійсного значення типу **`double`** (відповідний покажчик у списку_покажчиків повинен мати тип **`double`** `*`);
- `c` – уведення одного символу (відповідний покажчик у списку_покажчиків повинен мати тип **`char`** `*`);
- `s` – уведення текстового рядка (відповідний покажчик у списку_покажчиків повинен мати тип **`char`** `*`).

У форматному рядку функції `fprintf()` слідом за символом `%` можуть йти такі символи специфікації перетворення (перелік не повний):

- `d` або `i` – виведення десяткового цілого зі знаком;
- `f` – виведення дійсного значення (**`float`** або **`double`**) у вигляді числа з фіксованою точкою;
- `e` або `E` – виведення дійсного значення (**`float`** або **`double`**) у експонентній формі;

- `c` – виведення одного символу;
- `s` – виведення текстового рядка.

При нормальному завершенні функція `fscanf()` повертає кількість об'єктів, що отримали значення при введенні, а функція `fprintf()` – кількість виведених символів. У разі помилок уведення/виведення (у тому числі при виявленні ознаки кінця файлу під час уведення) обидві функції повертають значення `EOF`.

Функції `fprintf()` і `fscanf()` є небезпечними. Замість них рекомендується використовувати функції `fprintf_s()` і `fscanf_s()`, які, на відміну від функцій `fprintf()` і `fscanf()`, перевіряють параметри на достовірність (у тому числі коректність форматного рядка).

При введенні даних із файлу треба обов'язково перевіряти умову кінця файлу, оскільки читання ознаки кінця файлу призводить до помилки часу виконання. Для описаного вище потоку `f1` це можна зробити, наприклад, так:

```
int n;
while (fscanf(f1, "%d", &n) != EOF)
{
    // Дії
}
```

або

```
char s[300];
while (fgets(s, 300, f1) != NULL)
{
    // Дії
}
```

чи

```
char c;
c = fgetc(f1);
while (!feof(f1))                    // Див. нижче
{
    // Дії
    c = fgetc(f1);
}
```

Для позиціонування в потоці можна переміщати покажчик поточної позиції за допомогою наступної функції:

```
int fseek(FILE * покажчик_на_потік, long зміщення,
          int початок_відліку);
```

При звертанні до цієї функції параметр зміщення задається змінною або виразом типу **long** і вказує, на скільки байт потрібно перемістити покажчик поточної позиції у прямому (> 0) або зворотному (< 0) напрямку. Початок_відліку вказує, по відношенню до якої позиції здійснюється зсув покажчика поточної позиції. Він задається однією з трьох констант:

`SEEK_SET` (`== 0`) – початок файлу;
`SEEK_CUR` (`== 1`) – поточна позиція;
`SEEK_END` (`== 2`) – кінець файлу.

Слід пам'ятати, що константа типу **long** записується у вигляді десяткового значення, слідом за яким додається суфікс `L` або `l` (наприклад, `128L`).

Функція `fseek()` повертає значення 0, якщо переміщення виконано успішно; інакше вона повертає ненульове значення:

`fseek(f, -1L, SEEK_CUR);` – повернутися на 1 байт;
`fseek(f, 0L, SEEK_END);` – перейти на кінець потоку.

Деякі інші корисні функції:

long `ftell(FILE *` покажчик_на_потік`);` – повертає положення покажчика поточної позиції, що вимірюється в байтах від початку файлу, або значення (`-1L`) при помилці;

int `fgetpos(FILE *` покажчик_на_потік`, fpos_t *` позиція`);` – записує в область пам'яті, що адресується покажчиком позиція, положення покажчика поточної позиції файлу і повертає значення 0 при успішному виконанні або ненульове значення при помилці (тип `fpos_t` – це цілочисловий тип, що використовується для визначення положення покажчика поточної позиції файлу за допомогою функції `fgetpos()` або зміщення покажчика поточної позиції файлу функцією `fsetpos()`; див. нижче);

int fsetpos(FILE * покажчик_на_потік, fpos_t * позиція); – переводить покажчик поточної позиції файлу в позицію, яка зберігається в області пам'яті, що адресується покажчиком позиція, і повертає значення 0 при успішному виконанні або ненульове значення при помилці;

void rewind(FILE * покажчик_на_потік); – переміщає покажчик поточної позиції на початок потоку;

int feof(FILE * покажчик_на_потік); – повертає **ненульове** значення, якщо перед звертанням до цієї функції було прочитано ознаку кінця файлу, і 0, якщо спроби читання за кінцем файлу не було.

Якщо файл відкритий і для введення, і для виведення (режими "r+" і "a+"), то в разі потреби переходу від введення до виведення (або навпаки) необхідно обов'язково виконати примусове зміщення покажчика поточної позиції файлу (як варіант – на 0 байт відносно поточної позиції).

При роботі з файлами (будь-якими, а не тільки текстовими) корисними можуть бути такі дві функції, оголошені в заголовному файлі io.h:

int remove(**const char** *шлях); – знищення файлу, ім'я якого задане як параметр (файл повинен бути закритим);

int rename(**const char** *старе_ім'я, **const char** * нове_ім'я); – перейменування файлу або каталогу (файл повинен бути закритим). Якщо в новому імені файлу вказати інший шлях відносно старого, файл буде перенесений. Каталог перенести неможливо.

2. ПРИКЛАДИ ПРОГРАМ

Приклад 1. Дано текстовий файл з рядками довжиною не більше 255 символів. Переписати його вміст в інший текстовий файл за рядками, починаючи з останнього рядка і закінчуючи першим.

Розв'язок.

```
#include <stdio.h>
#include <string.h>
#include <conio.h>
int main()
{
    FILE *text1, *text2;
    char name1[261], name2[261], s[256];
    puts("Enter the name of the source file:");
    gets(name1);
```

```
            // Відкриття початкового файлу з перевіркою
if ((text1 = fopen(name1, "r")) == NULL)
{
    perror("1");
    getch();
    return 1;
}
puts("Enter the file name of the destination:");
gets(name2);
          // Відкриття результуючого файлу з перевіркою
if ((text2 = fopen(name2, "w")) == NULL)
{
    perror("2");
    getch();
    return 2;
}
int n = 0;
                  // Визначаємо кількість рядків у файлі
while(fgets(s, 256, text1) != NULL)
    n++;
for (int i = n; i > 0; i--)
{                 // Повертаємо покажчик поточної позиції
    rewind(text1);                    // на початок файлу
    // Можна інакше:
    // fseek(text1, 0L, SEEK_SET);
    // Читаємо до потрібного рядка. Останній прочитаний
             // у вкладеному циклі рядок - це останній,
              // передостанній, ..., перший рядок файлу
    for (int j = 0; j < i; j++)
        fgets(s, 256, text1);
        // Знищуємо символ '\n' у рядку, якщо він там є
            // (у файлі за останньою порцією інформації
           // могло не здійснюватися переведення рядка)
    if (s[strlen(s) - 1] == '\n')
        s[strlen(s) - 1] = '\0';
    fputs(s, text2);              // Пишемо рядок у файл
          // Після кожного рядка, що записується у файл
    if (i != 1)            // (крім останнього) виводимо
        putc('\n', text2);        // символ нового рядка
}
fclose(text2);                    // Закриваємо вихідний
fclose(text1);                        // і вхідний файли
```

```
    return 0;
}
```

Приклад 2. У текстовому файлі data.txt записано цілі числа, розділені довільною кількістю будь-яких пробільних символів, з можливими порожніми рядками в будь-якому місці файлу. Переписати в новий текстовий файл res.txt по одному в рядок без додаткових пробільних символів тільки ті з чисел, які кратні 3.

Розв'язок.

```
#include <stdio.h>
#include <conio.h>
int main()
{
   FILE *fInp, *fOut;
                  // Існуючий файл відкриваємо для читання
   fInp = fopen("data.txt", "r");
   if (!fInp)
      puts("Error of opening of a input-file" " data.txt");
   else
   {              // Відкриваємо новий файл для записування
      fOut = fopen("res.txt", "w");
      if (!fOut)
         puts("Error of opening of a output-file"
              " res.txt");
      else
      {
         int m, flag = 0;
         while (fscanf(fInp, "%d", &m) != EOF)
            if (m % 3 == 0)
            {                    // Усі числа, крім першого,
               if (flag)          // виводимо після переходу
                  putc('\n', fOut);      // до нового рядка
               fprintf(fOut, "%d", m);
               flag = 1;
            }
         fclose(fOut);       // Закриття результуючого файлу
         puts("Copying is completed!");
      }
      fclose(fInp);              // Закриття вхідного файлу
   }
   puts("Press any key");
   getch();
```

```
    return 0;
}
```

Приклад 3. Занести в текстовий файл інформацію про багаж пасажирів а саме, прізвище – текстовий рядок, перехід до нового рядка, кількість речей – ціле число, загальна вага – дійсне число. Ознака закінчення введення – порожній рядок замість прізвища. Інформацію про кожного пасажира виводити з нового рядка файлу. Якщо файл із указаним ім'ям уже існує, здійснювати дозаписування в його кінець.

Розв'язок.

```
#include <stdio.h>
#include <string.h>
#include <conio.h>

struct TLuggage
{
    char Surname[31];
    int Count;
    double Weight;
};

int main()
{
    TLuggage Luggage;
    FILE *f;
    char FileName[261];
    printf("Input name of file: ");
    gets_s(FileName, 261);
    if ((f = fopen(FileName, "a")) == NULL)
    {
        perror("Trouble opening a file ");
        getch();
        return 1;
    }
    printf("Ener surname: ");
    gets_s(Luggage.Surname, 31);
    while (strlen(Luggage.Surname))
    {
        printf("Enter the number of things (integer): ");
        scanf("%d", &Luggage.Count);
        printf("Enter the weight of luggage"
               " (a real number): ");
```

```
        scanf("%lf", &Luggage.Weight);
        getchar();// Зчитування ознаки кінця рядка введення
        fprintf(f, "%s\n%d %lf\n", Luggage.Surname,
                                Luggage.Count, Luggage.Weight);
        printf("Ener surname: ");
        gets_s(Luggage.Surname, 31);
    }
    fclose(f);
    return 0;
}
```

Приклад 4. Дано текстовий файл з інформацією про багаж пасажирів. У файлі кожна запис про багаж починається з нового рядка і має такий формат: прізвище, кінець рядка, кількість речей (ціле число), пробіл, вага речей (дійсне число). Вивести на екран прізвища пасажирів, чия вага речей більша за середню вагу всіх речей.

Розв'язок.

```
#include <stdio.h>
#include <string.h>
#include <conio.h>

struct TLuggage
{
    char Surname[31];
    int Count;
    double Weight;
};
int main()
{
    TLuggage Luggage;
    FILE *f;
    char FileName[261];
    int TotalCount = 0;
    double TotalWeight = 0.0;
    printf("Input name of file: ");
    gets_s(FileName, 261);
    if ((f = fopen(FileName, "r")) == NULL)
    {
        perror("Trouble opening a file\n");
        getch();
        return 1;
    }
```

```
    while (fgets(Luggage.Surname, 31, f) != NULL)
    {
        fscanf(f, "%d%lf", &Luggage.Count, &Luggage.Weight);
        TotalCount += Luggage.Count;
        TotalWeight += Luggage.Weight;
                  // Після ваги багажу у файлі здійснюється
                // переведення рядка. Зчитуємо один символ,
        getc(f);       // щоб він не сприймався як прізвище
    }
    fseek(f, 0L, SEEK_SET);// Повертаємось на початок файлу
    while(fgets(Luggage.Surname, 31, f) != NULL)
    {                          // Знищуємо символ нового рядка
        Luggage.Surname[strlen(Luggage.Surname) - 1] = '\0';
        fscanf(f, "%d%lf", &Luggage.Count, &Luggage.Weight);
        getc(f);                                  // Див. вище
        if (Luggage.Weight > TotalWeight / TotalCount)
            puts(Luggage.Surname);
    }
    fclose(f);
    getch();
    return 0;
}
```

Приклад 5. Дано текстовий файл із рядками довжиною не більше 255 символів. Розвернути кожен його рядок на 180°.

Розв'язання.

```
#include <stdio.h>
#include <string.h>
#include <conio.h>

int main()
{
    FILE *f;
    char FileName[261], str[256], ch;
    puts("Enter the name of file:");
    gets_s(FileName, 261);
      // Відкриття існуючого файлу для введення і виведення
    if ((f = fopen(FileName, "r+")) == NULL)
    {
        perror("Trouble opening a file\n");
        getch();
        return 1;
```

```
    }

    int k;
    long pos = 0;                  // Поточна позиція покажчика
                                        // потоку (початкова)
    while (fgets(str, 256, f))
    {           // Зараз активний режим читання. Далі писати
              // не можна Покажчик поточної позиції зсунувся
                                      // на strlen(str) байт
        k = strlen(str);
        if (str[strlen(str) - 1] == '\n')      // У str буде
            k--;           // міститись символ '\n', якщо це
                                  // не останній рядок файлу
        for (int i = 0; i < k / 2; i++)       // Розвертаємо
        {                               // вміст рядка без '\n'
            ch = str[i];
            str[i] = str[k - 1 - i];
            str[k - 1 - i] = ch;
        }
        fseek(f, pos, SEEK_SET);      // Повертаємо покажчик
          // поточної позиції. Тепер можна писати. Пишемо у
        fprintf(f, "%s", str);// файл. Далі читати не можна
        pos = ftell(f);     // Запам'ятовуємо нове положення
                             // покажчика поточної позиції
        fseek(f, 0L, SEEK_CUR);         // Зсуваємо покажчик
             // поточної позиції на 0 байт для забезпечення
                   // можливості наступного читання з файлу
    }
    fclose(f);                                // Закриваємо файл
    return 0;
}
```

3. ЗАВДАННЯ НА ЛАБОРАТОРНУ РОБОТУ

За час, відведений для виконання лабораторної роботи (2 академічні години), студент повинен:

1. Розробити алгоритм розв’язання задачі, запропонованої для програмування.
2. Здійснити програмну реалізацію розробленого алгоритму.
3. Здійснити налаштування програми, виправивши синтаксичні та логічні помилки.

4. Підібрати тестові дані для перевірки програми, включаючи винятковi випадки.
5. Оформити звіт до лабораторної роботи.
6. Відповісти на контрольні запитання.

4. ВАРІАНТИ ЗАДАЧ

1. Дано текстовий файл. Переписати в інший файл усі рядки початкового файлу, що не містять латинських літер.
2. Дано текстовий файл. Переписати в інший файл частини рядків, починаючи з останнього слова[1], що не містить цифри.
3. Дано текстовий файл. Вивести найдовше слово тексту, що міститься в ньому. Якщо таких слів декілька, вивести:
   а) останнє з них;
   б) перше з них.
4. Дано текстовий файл, що містить декілька рядків довжиною не більше 255 символів. У кожному з рядків без помилок записано єдиний вираз, який має вигляд a#b, де a, b – невід'ємні цілочислові величини, # – одна з операцій +, –, /, *. У файлі за останнім виразом переведення рядка не виконувалося. Вивести кожен із виразів і їх значення на екран і в новий текстовий файл.
5. Дано натуральне число *n*. Записати в текстовий файл усі подання цього числа сумою двох натуральних чисел. Перестановка доданків нового способу не дає.
6. Дано текстовий файл із довжинами рядків не більше 255 символів. Переписати в інший файл через пробіл усі числа, які представлені в початковому файлі як окремі слова. Довжини рядків у результуючому файлі не повинні перевищувати 255. У разі невиконання цієї умови, продовжувати виведення у новому рядку.
7. Дано текстовий файл. Переписати в інший файл через один пробіл тільки неповторювані слова вихідного файлу.
8. Дано текстовий файл. Переписати в інший файл його рядки відповідно до зростання їх довжин.

[1] Тут і далі під словом будемо розуміти будь яку послідовність символів, що не містить пробільні символи і символи керування, тобто символів з кодами, меншими за 33.

9. Дано текстовий файл із текстом, написаним англійською мовою. Переписати в інший файл його вміст за таким правилом: якщо у файлі зустрілася крапка, то наступний текст повинен сприйматися як нове речення, що починається з великої літери; усі символи «крапка з комою» замінити крапками з відповідним перетворенням наступного тексту.
10. Дано текстовий файл. Переписати в інший файл усі рядки, що не містять слова-«перевертні» (паліндроми). У результуючому файлі рядки повинні йти в порядку, зворотному порядку рядків початкового файлу.
11. Вводячи дані з клавіатури, сформувати текстовий файл fish.txt, рядки якого містять відомості «Суднового добового донесення»: назва судна, бортовий номер судна, дата вилову риби, назва риби, вага вилову (у тонах). Дано дійсне число *R* і текстовий рядок *S*. Вивести на екран рядки файлу fish.txt, у яких вага вилову перевищує *R* тон. Сформувати текстовий файл selection.txt, записавши в нього рядки файлу fish.txt, що задовольняють умові: назва риби збігається з *S*.
12. Дано два текстових файли. Чи мають їх відповідні рядки однакову довжину?
13. Дано текстовий файл, що містить деякий текст із декількох абзаців, кожен з яких починається з нового рядка, позначуваного символом (символами) табуляції або (та) одним чи декількома пробілами. Не вважаючи за абзаци порожні рядки і рядки, що містять лише пробіли та символи табуляції, визначити кількість абзаців у тексті.
14. Дано текстовий файл f, що містить деякий текст з кількох абзаців, кожен з яких починається з нового рядка, позначуваного більш ніж одним пробілом. Файл f не містить порожніх рядків і рядків, що складаються з одних пробілів. Переписати в новий файл уміст файлу f, вставити між сусідніми абзацами по одному порожньому рядку і видаливши у всіх рядках початкові пробіли.
15. Дано текстовий файл f. Переписати його вміст у файл *g*, вставивши в початок кожного рядка його номер, відлічуваний від 1. Номер відділяти від умісту рядка одним пробілом.
16. Дано текст, записаний у текстовому файлі f. Переписати у файл *g* уміст файлу f, виключивши повторення слів. У третій файл (файл *h*) записати по одному слову в рядку ті слова, що не були записані у файл *g*. У файлі *g* не повинно бути порожніх рядків. Варіанти:
    а) Слова, що повторюються у файлі f, не повинні потрапляти у файл *g*.

б) Повторюване у файлі f слово має бути записане у файл g один раз у тому місці, де воно зустрілося вперше у файлі f.

17. Дано два текстових файли f_1 і f_2. Переписати вміст файлу f_1 у новий файл f_3, додавши до кожного рядка файлу f_1 відповідний рядок файлу f_2. Якщо файл f_2 вичерпається раніше, ніж файл f_1, то рядки файлу f_1, що залишилися, переписати без їх змінювання. Додавання рядка виконувати через один пробіл:
    а) у початок відповідного рядка файлу f_1;
    б) у кінець відповідного рядка файлу f_1.

5. КОНТРОЛЬНІ ЗАПИТАННЯ

1. Для чого використовуються файли даних?
2. Що таке потік і які дії можна виконувати над ним?
3. Яке призначення покажчика поточної позиції?
4. У чому особливість текстових файлів?
5. Які попередні дії треба виконати для забезпечення роботи з текстовим файлом?
6. Охарактеризуйте режими відкриття текстових файлів. Як режими відкриття задаються при використанні засобів мови Сі?
7. Перелічіть і охарактеризуйте основні функції, що забезпечують уведення/виведення даних при роботі з текстовими файлами.
8. До яких наслідків приводить читання ознаки кінця файлу під час уведення даних з файлу? Що рекомендується робити для запобігання цих наслідків?
9. Як здійснюється читання даних з текстового файлу та запис у нього?
10. Як здійснюється читання числової інформації з тестового файлу?
11. Як здійснюється закриття потоку?
12. Як можна керувати положенням поточного покажчика?
13. Як можна повернутися на початок файлу, не здійснюючи повторне відкриття потоку?
14. Як здійснюється перейменування файлу?
15. Яка функція може бути використана для знищення файлу?

СПИСОК ЛІТЕРАТУРИ

1. Керниган, Б. Язык программирования Си / Б. Керниган, Д. Ритчи. – М. : Финансы и статистика, 1992. – 272 с.
2. Павловская, Т. А. C/C++. Программирование на языке высокого уровня / Т. А. Павловская. – СПб. : Питер, 2003. – 461 с.
3. Подбельский, В. В. Программирование на языке Си / В. В Подбельский, С. С. Фомин. – М. : Финансы и статистика, 1999. – 600 с.

Навчальне видання

Методичні вказівки
до лабораторної роботи
«Обробка текстових файлів з використанням
стандартних засобів мови Сі»
з курсу «Програмування» для студентів напряму
6.040302 – Інформатика і курсу «Програмування
та алгоритмічні мови» для студентів напряму
6.040303 – Системний аналіз

Укладачі: БЕЗМЕНОВ Микола Іванович,
БЕЗМЕНОВА Ольга Миколаївна

Відповідальний за випуск О. С. Куценко
Роботу до видання рекомендував О. В. Горілий

За авторською редакцією

План 2014 р., поз. 103.

Підп. до друку 07.07.2014 р. Формат 60×84 1/16. Папір офсетний.
Riso-друк. Гарнітура Таймс. Ум. друк. арк. 1,2. Наклад 50 пр.
Зам. № 247. Ціна договірна.

Видавець і виготовлювач
Видавничий центр НТУ «ХПІ»,
вул. Фрунзе, 21, Харків, 61002.

Свідоцтво суб'єкта видавничої справи ДК № 3657 від 27.12.2009 р.