Лабораторная работа № 8

Организация хранения данных в файлах (списки строк)

Цель работы: Изучение организации хранения символьных строк в файлах и работы с ними при использовании языка программирования Си.

Теоретические сведения

Организация хранения символьной информации в файлах

Под текстовыми файлами в вычислительной системе понимаются файлы, содержащие последовательность байт, представляющие символьные строки в определенной кодировке, а также специальные управляющие символы, определяющие конец строки.

Таким образом, текстовый файл может быть представлен как массив строк, каждый элемент-строка которого представляет собой также массив переменной длины из кодов символов.

Остальные виды файлов текстовых документов распознаваемые текстовыми редакторами, являются или двоичными файлами (например, doc, xls), содержащими сложные структуры двоичных данных или тектовыми файлами (например, html, rtf), содержащими специальную текстовую разметку данных. Такие файлы позволяют хранить не только символьную информацию, но информацию о формате текста (стиль шрифта, разметку текста), а также дополнительную несимвольную информацию – разметку и оформление таблиц, рисунки и т.п.

При обработке символьной информации надо учитывать, что кодировка текстовых файлов может быть любая, поэтому перед и после обработки необходимо выполнять преобразование кодировки символьной информации в требуемую.

В качестве управляющих символов, встречающиеся в текстовых файлах, используются символы ASCII с кодами меньшими 32, которые изначально предназначались для специальных кодов управления

символьными терминалами и принтерами. В настоящее время большинство из этих кодов непосредственно в текстовых файлах неиспользуются. Основные из них, использующиеся в настоящее время, приведены в таблице ниже.

Код символа dec	**Код символа hex**	**Обозначение символа в Си**	**Назначение**
9	0x09	'\t'	Символ отступа (табуляция)
10	0x0A	'\n'	Перевод каретки (курсора) на новую строку
13	0x0D	'\r'	Возврат каретки (курсора) в начало строки

Символ табуляции 9 ('\t') при обработке может интерпретироваться просто как символ пробела, а при отображении текста в программе редакторе или просмотрщике, обозначает отступ последующих символов, при этом, как и насколько выполнять отсуп решает сама программа.

В операционных системах Windows и Unix-совместимых системах используются разные способы для обозначения конца строки в текстовых файлах. В Unix-системах и основанных на ней (Linux) используется один символ с кодом 10 ('\n'), а в Windows используется последовательность двух символов 13,10 ('\r\n').

В Windows реализован автоматический перевод кода 10 в последовательность 13,10 при операциях чтении/записи текстовых файлов (но не наоборот). Перевод возникает, когда байт 10 записывается в текстовый файл или на консоль, а обратный перевод из 10 выполняется при чтении текстовых файлов.

Так как коды символов с кодами меньшими 128 совпадают в ASCII, Unicode и ANSI-кдироваках, то управляющие символы обрабатываются одинакого во всех основных кодировках.

Работа с текстовыми файлами в языке Си

В языке Си для работы с тектовыми файлами можно использовать библиотеку stdio.h и структуру FILE. Для работы с файлом в текстовом режиме при открытии файла функцией fopen необходимо указать в параметре режима открытия символ "t". После открытия файла можно использовать ряд функций для потокового ввода/вывода символьной информации.

Ввод/вывод строки из потока осуществляется с помощью функций fgets и fputs:

```
char *fgets( char *strbuf, int max, FILE *filestream )
int fputs( const char *strbuf, FILE * filestream )
```

Функция fgets считывает символы из потока и сохраняет их в виде строки по указателю strbuf до тех пор, пока или не встретиться конец строки, или не будет достигнут конец файла, или не будет прочитано max-1 символ (max задает размер памяти в strbuf, включая место под нулевой символ). Символ новой строки прекращает работу функции fgets, но он считается допустимым символом, и поэтому тоже копируется в strbuf.

В случае успеха, функция возвращает указатель на strbuf. Если конец файла был достигнут и ни один символ не был прочитан, содержимое strbuf остается неизменными и возвращается нулевой указатель.

Функция fputs записывает строку, указанную в параметре strbuf в поток filestream. В случае ошибки, функция возвращает значение EOF.

Примеры:

```
fgets(str, 20, ifile);// считывает строку из ifile
                      // в str (не более 19 символов)
fputs(str, ofile); // записывает в ofile строку из str
```

Для ввода/вывода строк по заданному шаблону для файлов существуют функции, аналогичные printf и scanf:

```
int fprintf (FILE *stream, const char *template, ...);
```

```
int fscanf (FILE *stream, const char *template, ...);
```

Форматная строка template задает входной/выходной формат аналогично функциям printf/scanf.

При необходимости чтения/записи отдельных символов могут использоваться функции:

```
int fgetc( FILE *filestream )
int fputc( int c, FILE *filestream )
```

Функция fgetc читает один символ из указанного потока filestream. После чего, внутренний индикатор позиции в файле сдвигается на одни символ, таким образом, он уже указывает на следующий символ.

Функция возвращает код считанного символа или при возникновении ошибки, возвращается EOF и устанавливается индикатор ошибки.

Функция fputc записывает символ в поток и перемещает позицию индикатора положения. Если ошибок нет, то возвращается символ, который был записан в поток. При возникновении ошибки, возвращается EOF и устанавливается индикатор ошибки.

Пример работы с текстовым файлом – вывод в отдельный файл всех чисел из исходного файла:

```
#include <stdlib.h>
#include <stdio.h>
...

 FILE *f_in, *f_out;
 char *strbuf;
 int i, dig_c, dig_o, dig_idx;

 f_in = fopen("indat.txt","rt");
 if (f_in == NULL) {
      printf("Ошибка открытия файла\n");
      exit(0);
 }
 f_out = fopen("outdat.txt","wt");

 strbuf = new char[255]; // буфер для читаемой строки

 while (fgets(strbuf,254,f_in)) { // цикл чтения строк
     printf("Строка: %s", strbuf);

     i = 0; dig_o = 0;
     while (strbuf[i]) // цикл обработки строки
```

```
      {
       // определение типа текущего символа
       if ((strbuf[i]>='0') && (strbuf[i]<='9'))
            dig_c = 1;
       else dig_c = 0;

       if (dig_c) { // если текущий символ цифра
         if (!dig_o) // и не было цифр
         {
           dig_idx = i; // запомнить начало числа
           dig_o = 1; // установить признак числа
         }
         //else ; // иначе - идет число, продолжить
       } else { // иначе текущий не цифра
         if (dig_o) { // и были цифры, то конец чмсла
           strbuf[i] = '\0'; // ограничить строку до числа
           printf("Число: %s\n", strbuf+dig_idx);
           fprintf(f_out,"%s\n", strbuf+dig_idx);
           dig_o = 0; // сбросить признак числа
         }
         //else ; // иначе - пока нет числа, продолжить
       }
       i++;
      }
  }
  free(strbuf);
  fclose(f_out);
  fclose(f_in);
```

Задание на лабораторную работу

1. Разработать программу ввода исходной последовательности символьных строк с учетом варианта задания и сохранения вводимых строк в текстовый файл.

2. Разработать программу чтения строк из файла, обработки строк, согласно варианту задания, и сохранения результата в новый файл, с выводом исходных и результирующих данных.

Содержание отчета по лабораторной работе

– титульный лист;

– тема и цель работы; задание на лабораторную работу согласно варианту.

– ход выполнения работы:

- исходные тексты полученных программ на языке Си;
- снимки экранов с результатами выполнения программ;
- снимки экранов Hex-редактора и текстового просмотрщика, демонстрирующие исходный и результирующий файлы;
- выводы о проделанной работе.

Контрольные вопросы

1. Что такое текстовые файлы в вычислительной системе?
2. Как организуется хранеиие списка строк текстовом файле?
3. Что такое управляющие символы?
4. Как организуется работа с текстовыми файлами в языке программирования С?
5. Какие есть операции по вводом/вывода строк для файлов в языке программирования С?

Варианты для задания

1. Дан текстовый файл. Переписать в другой файл только те строки, длина которых длиннее K символов (пробелы не учитываются).
2. Дан текстовый файл. Сформировать два новых текстовых файла – переписать в один новый файл только буквы, в другой новый файл только цифры исходных строк.
3. Дан текстовый файл. Сформировать новый текстовый файл – организовать замену слов, "старое" и "новое" слово вводятся с клавиатуры
4. Дан текстовый файл. Сформировать новый текстовый файл – удалить из исходных строк лишние повторяющиеся пробелы.
5. Дан текстовый файл. Сформировать новый текстовый файл - заменить в строках исходного файла все цифры на '*'.
6. Дан текстовый файл. Сформировать новый текстовый файл - удалить из строк исходного файла все цифры.

7. Дан текстовый файл. Переписать в новый файл все символы за исключением символов разделителей ('.' , ',' , ';' , '?' и т.п.).

8. Дан текстовый файл. Переписать в новый файл все слова из исходного файла, каждое с новой строки.

9. Дан текстовый файл. Сформировать два новых текстовых файла – переписать в один новый файл слова с нечетным номером, в другой – с четным.

10. Дан текстовый файл. Сформировать новый текстовый файл – переписать только те слова, которые начинаются на вводимую букву.

11. Дан текстовый файл. Сформировать новый текстовый файл, содержащий количество строк, символов и слов в исходном файле.

12. Дан текстовый файл. Сформировать новый текстовый файл – переписать только те слова, которые состоят из русских букв.

13. Дан текстовый файл. Сформировать новый текстовый файл, содержащий количество разделительных знаков ('.' , ',' , ';' , '?' и т.п.) в каждой строке исходного файла.

14. Дан текстовый файл. Сформировать новый текстовый файл, содержащий количество чисел в каждой строке исходного файла.

15. Дан текстовый файл. Сформировать новый текстовый файл, содержащий каждое предложение из исходного файла с новой строки. Предложение рассматривается как последовательность символов заканчивающееся '.' , '?' , '!'.