МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА – Российский технологический университет»
РТУ МИРЭА

Институт комплексной безопасности и специального приборостроения
Кафедра КБ-2 «Прикладные информационные технологии»

А.А. МЕРСОВ

МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ ПРАКТИЧЕСКОЙ РАБОТЫ № 4
Работа с файлом.

по дисциплине: **«Языки программирования»**
(наименование дисциплины)

Москва – 2021

УДК
ББК

Печатается по решению редакционно-издательского совета «МИРЭА – Российский технологический университет»

Мерсов А.А.

Методические указания по выполнению практической работы № 4 по языкам программирования / А.А. Мерсов– М.: МИРЭА – Российский технологический университет, 2021.

Методические указания предназначены для выполнения практической работы по дисциплине «Языки программирования» и содержит перечень вариантов практической работы, а также краткое изложение теоретического материала в форме пояснений к заданию на работу. Для студентов, обучающихся по направлениям 09.03.02, 10.03.01, 10.05.02, 10.05.03, 10.05.04.

Материалы рассмотрены на заседании учебно-методической комиссии КБ-2 Протокол №1 от «28» августа 2021 г. и одобрены на заседании кафедры КБ-2.

зав. кафедрой КБ-2

к.т.н. _____ / О.В.Трубиенко /

УДК ББК

Содержание

Общие указания к выполнению практической работы

Практические работы выполняются с использованием персональных компьютеров. Указания по технике безопасности совпадают с требованиями, предъявляемыми к пользователю ЭВМ. Другие опасные факторы отсутствуют.

Цель практической работы

Цель работы: ознакомление с функциями работы с файлами.

Практическая работа предполагает выполнение задания разработке и тестированию программного обеспечения.

Файл – это именованная область ячеек памяти, в которой хранятся данные одного типа. Файл имеет следующие характерные особенности:

уникальное имя;

однотипность данных;

произвольная длина, которая ограничивается только емкостью диска.

Файлы бывают текстовыми и двоичными.

Текстовый файл – файл, в котором каждый символ из используемого набора хранится в виде одного байта (код, соответствующий символу). Текстовые файлы разбиваются на несколько строк с помощью специального символа "конец строки". Текстовый файл заканчивается специальным символом "конец файла".

Двоичный файл – файл, данные которого представлены в бинарном виде. При записи в двоичный файл символы и числа записываются в виде последовательности байт (в своем внутреннем двоичном представлении в памяти компьютера).

Все операции ввода-вывода реализуются с помощью функций, которые находятся в библиотеке C++. Библиотека C++ поддерживает три уровня ввода-вывода:

потоковый ввод-вывод;

ввод-вывод нижнего уровня;

ввод-вывод для консоли и портов (зависит от ОС).

Поток – это абстрактное понятие, относящееся к любому переносу данных от источника к приемнику.

Функции библиотеки ввода-вывода языка C++, поддерживающие обмен данными с файлами на уровне потока, позволяют обрабатывать данные различных размеров и форматов, обеспечивая при этом буферизованный ввод и вывод. Таким образом, поток представляет собой этот файл вместе с предоставленными средствами буферизации.

Чтение данных из потока называется извлечением, вывод в поток – помещением (включением).

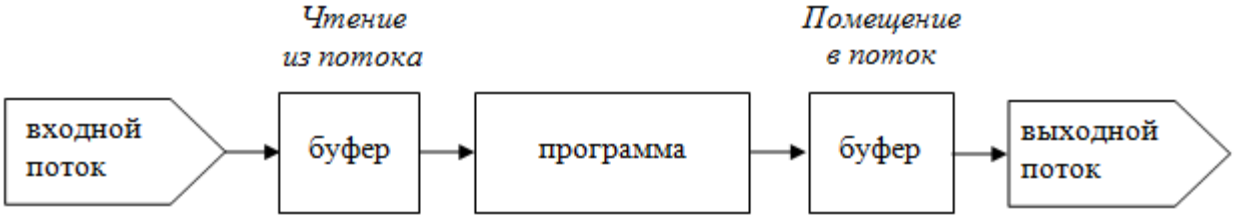


Рис. 1 Буферизация данных при работе с потоками

Для работы с файлом в языке C++ необходима ссылка на файл. Для определения такой ссылки существует структура FILE, описанная в заголовочном файле stdio.h. Данная структура содержит все необходимые поля для управления файлами, например: текущий указатель буфера, текущий счетчик байтов, базовый адрес буфера ввода-вывода, номер файла.

Функция открытия файла

При открытии файла (потока) в программу возвращается указатель на поток (файловый указатель), являющийся указателем на объект структурного типа FILE. Этот указатель идентифицирует поток во всех последующих операциях.

Например:

#include

.....

FILE *fp;

Для открытия файла существует функция fopen, которая инициализирует файл.

Синтаксис:

fp=fopen(ИмяФайла, РежимОткрытия);

где fp – указатель на поток (файловый указатель);

ИмяФайла – указатель на строку символов, представляющую собой допустимое имя файла, в которое может входить спецификация файла (включает обозначение логического устройства, путь к файлу и собственно имя файла);

РежимОткрытия – указатель на строку режима открытия файла.

Например:

fp=fopen("t.txt","r");

Существуют несколько режимов открытия файлов.

Поток можно открыть в текстовом (t) или двоичном (b) режиме. По умолчанию используется текстовый режим. В явном виде режим указывается следующим образом:

"r+b" или "rb" – двоичный (бинарный) режим;

"r+t" или "rt" – текстовый режим.

Функция закрытия файла

Открытые на диске файлы после окончания работы с ними рекомендуется закрыть явно. Это является хорошим тоном в программировании.

Синтаксис:

int fclose(УказательНаПоток);

Возвращает 0 при успешном закрытии файла и -1 в противном случае.

Открытый файл можно открыть повторно (например, для изменения режима работы с ним) только после того, как файл будет закрыт с помощью функции fclose().

Функция удаления файла

Синтаксис:

int remove(const char *filename);

Эта функция удаляет с диска файл, указатель на который хранится в файловой переменной filename. Функция возвращает ненулевое значение, если файл невозможно удалить.

Функция переименования файла

Синтаксис:

int rename(const char *oldfilename, const char *newfilename);

Функция переименовывает файл; первый параметр – старое имя файла, второй – новое. Возвращает 0 при неудачном выполнении.

Функция контроля конца файла

Для контроля достижения конца файла есть функция feof.

int feof(FILE * filename);

Функция возвращает ненулевое значение, если достигнут конец файла.

Функции ввода-вывода данных файла

1) Символьный ввод-вывод

Для символьного ввода-вывода используются функции:

int fgetc(FILE *fp);

где fp – указатель на поток, из которого выполняется считывание.

Функция возвращает очередной символ в формате int из потока fp. Если символ не может быть прочитан, то возвращается значение EOF.

int fputc(int c, FILE*fp);

где fp – указатель на поток, в который выполняется запись;

c – переменная типа int, в которой содержится записываемый в поток символ.

Функция возвращает записанный в поток fp символ в формате int. Если символ не может быть записан, то возвращается значение EOF.

Пример 1.

```
#include "stdafx.h"
#include “iostream”
using namespace std;
int main()
{
FILE *f;
int c;
char *filename="t.txt";
if ((f=fopen(filename,"r"))==0)
perror(filename);
else
while((c = fgetc(f)) !=EOF)
putchar(c);
//вывод с на стандартное устройство вывода
fclose(f);
system("pause");
return 0;
```

```
}
```

2) Строковый ввод-вывод

Для построчного ввода-вывода используются следующие функции:

```
char *fgets(char *s, int n, FILE *f);
```

где char *s – адрес, по которому размещаются считанные байты;

int n – количество считанных байтов;

FILE *f – указатель на файл, из которого производится считывание.

Прием байтов заканчивается после передачи n-1 байтов или при получении управляющего символа '\n'. Управляющий символ тоже передается в принимающую строку. Строка в любом случае заканчивается '\0'. При успешном завершении считывания функция возвращает указатель на прочитанную строку, при неуспешном – 0.

```
int fputs(char *s, FILE *f);
```

где char *s – адрес, из которого берутся записываемые в файл байты;

FILE *f – указатель на файл, в который производится запись.

Символ конца строки ('\0') в файл не записывается. Функция возвращает EOF, если при записи в файл произошла ошибка, при успешной записи возвращает неотрицательное число.

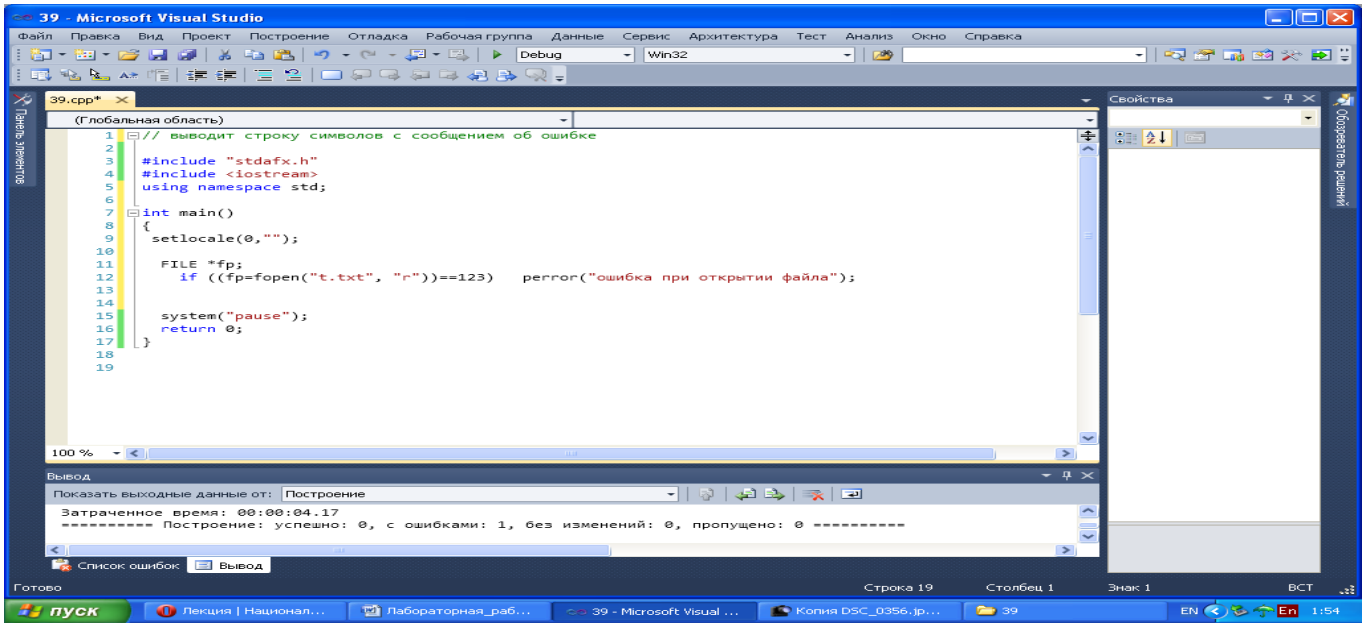
Пример 2. Построчное копирование данных из файла f1.txt в файл f2.txt.

```
#include “iostream"
using namespace std;
#define MAXLINE 255 //максимальная длина строки
int main()
{
//копирование файла in в файл out
FILE *in, //исходный файл
*out; //принимающий файл
char buf[MAXLINE];
```

```
//строка, с помощью которой выполняется копирование
in=fopen("f1.txt","r");
//открыть исходный файл для чтения
out=fopen("f2.txt","w");
//открыть принимающий файл для записи
while(fgets(buf, MAXLINE, in)!=0)
//прочитать байты из файла in в строку buf
fputs(buf, out);
//записать байты из строки buf в файл out
fclose(in); //закрыть исходный файл
fclose(out);//закрыть принимающий файл
system("pause");
return 0;
}
```

Методический пример

1. Создайте новое консольное приложение C++ (*Файл→Создать→Проект→ "Консольное приложение Win32"*).
2. Напишите программу обнаружения и вывода ошибки при открытии файла.



Пояснение к коду:
В файле **stdio.h** определена константа **EOF**, которая сообщает об окончании файла (отрицательное целое число).

При открытии файла могут возникать следующие ошибки:

- файл, связанный с потоком не найден (при чтении из файла);
- диск заполнен (при записи);
- диск защищен от записи (при записи) и т. п.

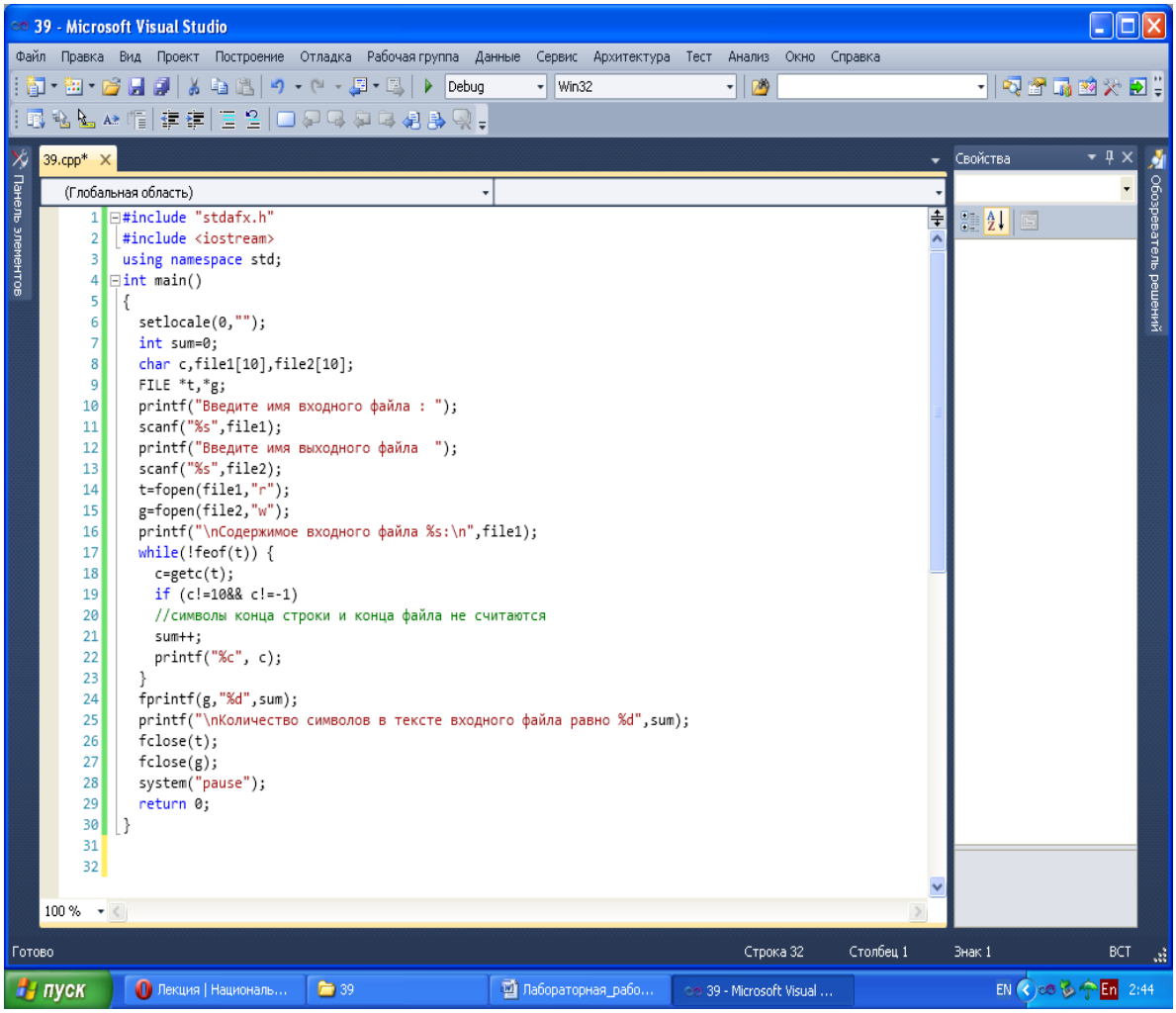
В этих случаях указатель на поток приобретет значение **NULL (0).** Указатель на поток, отличный от аварийного, не бывает равен **NULL**.

Для вывода сообщения об ошибке при открытии файла используется стандартная библиотечная функция из файла **:**

void perror (const char*s);

Функция **perror()** выводит строку символов, адресуемую указателем **s**, за которой размещаются: двоеточие, пробел и сообщение об ошибке.

3. Внимательно изучите код и комментарии к нему.
4. Выполните **примеры 1** и **2** из теоретической части. Внимательно построчно разберите код.
5. Напишите программу подсчета количества символов в заданном тексте и файловый ввод-вывод данных. Работа программы должна включать ввод пользователем с клавиатуры имен входного и выходного файлов. Результат работы программы сохраняется в выходном файле, а также выводится на экран.



6. Внимательно изучите код и комментарии к нему.
7. Выполните отладку (F5) и проверьте работоспособность программы.

По аналогии с предыдущими примерами напишите программный код (**текстовые исходные файлы задаются самостоятельно**).

Варианты задания

0. определения чаще всего встречающейся в заданном файле буквы;
1. удвоения в содержимом файла каждой литеры(символа);
2. подсчёта числа цифр в данном файле и их суммы;
3. подсчета количества слов в файле (отделяются пробелами);
4. подсчитать количество строк в файле
5. в выходной файл записать исходный без разбивки по строкам
6. разбить входной файл(состоящий из одной строки не более 256 символов) на файл, содержащий информацию из входного, разбитого на строки, каждая размером соответствующая числу, вводимому с клавиатуры;
7. подсчитать в исходном файле количество цифр и символов. В выходной файл записать те данные, количество которых больше.

8. записать в выходной файл данные из исходного без символов, которые будут указаны при вводе с клавиатуры(может быть несколько но не более 5)
9. записать в выходной файл данные из исходного. Данные выходного файла должны располагаться в следующем порядке: сначала все цифры, а потом все остальные символы.