

## Тема 11 ФАЙЛЫ. ОБРАБОТКА ТЕКСТОВОЙ ИНФОРМАЦИИ

**Цель занятия:** освоение основных приемов работы с файлами; получение практических навыков в создании и обработке текстовой информации.

### Теоретические сведения

Данные различной формы хранятся в файлах. Это программы и документы, изображения и звуки. К файлам относятся прикладные и сервисные программы, данные, полученные с помощью программ.

**Файл** – это поименованная область памяти на диске или другом внешнем запоминающем устройстве.

Ввод данных в программу из файла – это один из способов ввода данных. Он необходим, как правило, в больших профессиональных программах, т.к. программа с размещёнными внутри неё данными не очень удобна для реального применения. Данные могут меняться часто, а всякое изменение данных в теле программы приводит к изменению самой программы. Внешние файлы данных повышают универсальность и гибкость программ обработки данных. Операция *ввода* данных означает заполнение ячеек памяти данными, полученными из файла, а операция *вывода* – пересылку данных из рабочей памяти (ОЗУ) в файл. Эти операции осуществляются через область памяти, называемую буфером.

Любая программа, использующая существующий файл для ввода или создающая новый файл при выводе, неизбежно содержит шаги:

- 1) *открытие файла* (в соответствующем режиме);
- 2) *обработка файла* (чтение или запись);
- 3) *закрытие файла*.

Существуют также функции, которые в сложных программах можно использовать для обработки ошибок ввода-вывода (проверки наличия файла данных, проверки формата данных, выбора режима обработки файла, и др.)

В C++ файловый ввод-вывод ничем не отличается от консольного. За единственным исключением – если данные читаются из файла, то в любой момент можно вернуться к началу файла и считать все заново.

Для того, чтобы в C++ работать с файлами, необходимо подключить заголовочный файл ***fstream***:

```
#include <fstream>
```

После этого можно объявлять объекты, привязанные к файлам: для чтения данных из файла используются объекты типа ***ifstream*** (аббревиатура от ***input file stream***, для записи данных в файл используются объекты типа ***ofstream*** (***output file stream***).

Например:

```
ifstream in;           // Поток in будет использоваться для чтения
ofstream out;          // Поток out будет использоваться для записи
```

Чтобы привязать тот или иной поток к файлу (открыть файл для чтения или для записи) используется метод ***open***, которому необходимо передать параметр – текстовую строку, содержащую имя открываемого файла.

```
in.open("input.txt");
out.open("output.txt");
```

После открытия файлов и привязки их к файловым потокам, работать с файлами можно так же, как со стандартными потоками ввода-вывода *cin* и *cout*. Например, чтобы вывести значение переменной *x* в поток *out*, используется следующая операция:

```
out<<x;
```

А чтобы считать значение переменной из потока *in*

```
in>>x;
```

Для закрытия ранее открытого файла используется метод *close()* без аргументов:

```
in.close();  
out.close();
```

Закрытый файловый поток можно открыть заново при помощи метода *open*, привязав его к тому же или другому файлу.

Чтобы определить конец файла, программы могут использовать функцию *eof* потокового объекта. Эта функция возвращает значение 0, если конец файла еще не встретился, и 1, если встретился конец файла. Используя цикл *while*, программы могут непрерывно читать содержимое файла, пока не найдут конец файла, как показано ниже:

```
while (! in.eof())  
{  
    // Операторы  
}
```

Кроме того, состояние файлового потока можно проверить, если просто использовать идентификатор потока в качестве логического условия:

```
if (in)  
{  
    }
```

Также можно использовать в качестве условия результат, возвращаемый операцией считывания. Если считывание было удачным, то результат считается истиной, а если неудачным – ложью. Например, организовать считывание последовательности целых чисел можно так:

```
int d;  
while(in>>d)  
{  
    }
```

А организовать считывание файла построчно (считая, что строка заканчивается символом перехода на новую строку) так:

```
char S[100];  
while ( getline(in,S))  
{  
    }
```

Чтобы помочь программам следить за ошибками, можно использовать функцию *fail* файлового объекта. Если в процессе файловой операции ошибок не было, функция возвратит ложь (0). Однако, если встретилась

ошибка, функция *fail* возвратит истину. Например, если программа открывает файл, ей следует использовать функцию *fail*, чтобы определить, произошла ли ошибка, как это показано ниже:

```
if (in.fail())
{
    cerr << "Ошибка открытия файла" << endl;
    getch();
    exit(1);
}
или
if(!in.is_open())           //если не удалось открыть файл
{
    cout<<"Файл не найден";
    getch();
    return 1;
}
```

**Задача 11.1.** Дан файл. Получить все его строки на экране, указав длину каждой строки, обозначив пустые строки фразой «пустая строка» и подсчитав общее количество строк в файле.

```
#include "stdafx.h"
#include <iostream>
#include <fstream> //для работы с файлами
#include <string>
#include <locale>
#include <conio.h>
using namespace std;
int main()
{
    setlocale (LC_CTYPE,"rus");
    string str;
    ifstream fin;           //файл для чтения
    fin.open("e:\\x.txt");
    if(!fin.is_open())      //если не удалось открыть файл
    {
        cout<<"Файл не найден";
        getch();
        return 1;
    }
    int c=0;
    while(!fin.eof())       //пока не конец файла
    {
        getline(fin,str,'\n');
        c++;
        if (str.size()==0)   //если строка пустая
            cout<<"П у с т а я   с т р о к а\n";
        else                //выводится строка и ее длина
            cout<<str<<" ДЛИНА " <<str.size()<<"\n";

    }
    cout<<"Всего строк в файле: " <<c-1;
    fin.close();           //закрывать файл
    getch();               //ждать нажатия любой клавиши
    return 0;
}
```

**Задача 11.2.** Дан файл. Получить в новом файле данные исходного файла, за исключением пробелов. Вывести содержимое полученного файла на экран.

```
#include "stdafx.h"
#include <iostream>
#include <fstream> //для работы с файлами
#include <locale>
#include <conio.h>
using namespace std;
int main()
{
    setlocale (LC_CTYPE,"rus");
    ifstream fin;           //файл для чтения
    fin.open("e:\\x.txt");
    ofstream fout;          //файл для записи
    fout.open("e:\\y.txt");
    if(!fin.is_open())      //если не удалось открыть файл
    {
        cout<<"Файл не найден";
        getch();
        return 1;
    }
    char letter;
    while(!fin.eof())        //пока не конец файла
    {
        letter = fin.get(); //читает содержимое файла по одному символу за один раз
        if (letter!=' ')    //если не пробел
        {
            fout<<letter;   //запись символа в файл
            cout<<letter;    //вывод символа на экран
        }
    }
    fin.close();            //закрыть файл
    fout.close();
    getch();                //ждать нажатия любой клавиши
    return 0;
}
```

Из предыдущей задачи можно сделать вывод, что обычно работа с файлами с текстовой информацией организуется построчно, но в данной задаче, очевидно, необходимо посимвольное чтение элементов из файла, соответствующий их анализ и посимвольная запись в новый файл и на экран с сохранением построчной структуры.

**Задача 11.3.** Дан файл с текстовой информацией. Получить новый файл, ограничив длину всех слов до 6 символов, если слово короче, то добавить '!'.

```
#include "stdafx.h"
#include <iostream>
#include <fstream> //для работы с файлами
#include <string>
#include <locale>
#include <conio.h>
using namespace std;
string Neg(string st)
{
    string nst,s;
    st+=" ";
```

```

int len = st.size() ; //длина введенного текста
int i = 0;
int begin = 0, end = 0;
while (i < st.size()-1)
{
    while (st[i] == ' ' && i < st.size()-1)
        ++i; //пропустить пробелы
    begin = i; // номер первого символа слова
    while (st[i] != ' ' && i < st.size()-1)
        ++i; // пропустить символы слова

    end = i; // номер символа, следующего за последним символом слова

    //сформировать новую строку, ограничив длину всех слов до 6, если слово короче, то добавить !
    s=st.substr(begin,end-begin) ;// копируем в s (end-begin)символов строки st, начиная с begin
    s.resize(6,' ');
    nst+=s+' ';
}
cout<<nst<<"\n";
return nst;
}

int main()
{
    setlocale (LC_CTYPE,"rus");
    string str, nstr;
    ifstream fin; //файл для чтения
    fin.open("e:\\x.txt");
    ofstream fout; //файл для записи отсортированного массива
    fout.open("e:\\y.txt");
    if(!fin.is_open()) //если не удалось открыть файл
    {
        cout<<"Файл не найден";
        getch();
        return 1;
    }
    while(!fin.eof()) //пока не конец файла
    {
        getline(fin,str,'\n'); //чтение строки из файла
        nstr=Neg(str);
        fout<<str<<"\n"; //запись строки в файл
    }
    fin.close(); //закрыть файл
    fout.close();
    getch(); // ждать нажатия любой клавиши
    return 0;
}

```

### Методические указания

При подготовке к занятию необходимо изучить назначение и параметры функций работы с файлами; особенности работы с текстовой информацией.

### Аудиторные и домашние задания

1. Дан файл. Найти строку, в которой доля гласных (*a, e, i, o*) максимальна.
2. Дан файл Строки, начинающиеся с цифры, поместить в новый файл.
3. Дан файл и строка *st*. Если строка файла содержит в качестве фрагмента строку *st*, то в начало этой строки вставить звездочку и пробел.
4. Дан файл. В строках каждую точку заменить многоточием (т.е. тремя точками). Результат записать в новый файл.
5. Дан файл. Преобразовать строки файла следующим образом: если строка является палиндромом, т.е. 1-й ее символ равен последнему, 2-й - предпоследнему и т.д., то оставить ее без изменения, иначе строку удвоить, дописав в ее конец эту же строку. Результат поместить в новый файл.
6. Дан файл, содержащий программу на языке C++. Проверить эту программу на несоответствие числа открывающих и закрывающих фигурных скобок.
7. В файле заменить каждую цифру на следующую по величине цифру ('9' заменить на '0'), результат поместить в новый файл.
8. Дан файл. Слова в строках разделены пробелами. Найти строку с наибольшим числом слов.
9. Дан файл. Слова в строках разделены пробелами. В строках с четными номерами записать слова в обратном порядке.
10. Дан файл. Слова в строках разделены пробелами. В словах, начинающихся с буквы *a*, удалить эту букву.
11. Дан файл. Слова в строках разделены пробелами. В каждой строке найти самое короткое слово и его номер.
12. Дан файл. Слова в строках разделены пробелами. В каждой строке удалить последнее слово. Результат поместить в новый файл.

### Контрольные вопросы

1. Что такое файл?
2. Какой заголовочный файл необходимо подключить для работы с файлами?
3. Назовите функции для работы с файлами.
4. Какие методы используют для открытия и закрытия файлов?
5. Как проверить достижение конца файла?
6. Как прочитать информацию из файла построчно, посимвольно?