**9 Работа с файлами**

**Сроки сдачи работы и оценки:**

* **04.02-05.03 – 5**
* **06.03-07.03 – 4**
* **с 8.03 – 3**

# Обработка текстовых файлов

**Файл** (от англ. file-досье, документ) - это произвольная последовательность данных некоторой длины, имеющая имя. Другими словами, файл - это поименованное место на внешнем носителе.

**Файлы, открытые в текстовом режиме,** являются файлами *последовательного* доступа. Их можно считывать только последовательно, от первой строки до последней.

Запись информации в текстовый файл происходит либо от начала файла (при этом вся информация, находившаяся там ранее, стирается), либо в конец файла (добавление информации в файл).

**Назначение команд работы с файлами:**

|  |  |
| --- | --- |
| ifstream f1;  ofstream f2; | Объявление потоков (файловых переменных) |
| f2.open("file1.txt", ios::out); | Открытие файла в режиме записи |
| f1.open("file1.txt", ios::in); | Открытие файла в режиме чтения |
| f2.open("file1.txt", ios::app); | Открытие файла в режиме добавления информации в конец файла |
| int x;  f1.getline(x); | Чтение из файла целого значения |
| char str[120];  f1.getline(str, 120); | Чтение из файла строки |
| f2 << str << endl; | Запись в файл строки |
| if (!f1.is\_open())  cout << "Такого файла нет!" << endl; | Проверка открытия файла |
| char str[120];  while (!f1.eof())  {  f1.getline(str, 120);  } | Цикл для чтения из текстового файла всех строк |
| char ch;  cin >> ch;  while (!f1.eof())  {  f1.getline(str, 120);  if (str[0]==ch)  f2 << str << endl;  } | Чтение из одного файла всех строк и запись в другой файл строк, начинающихся на заданную букву |
| f1.close(); | Закрытие файла |

**Пример 1**

Создать в редакторе текстовый файл, внести в него 10 строк произвольного текста. Разработать программу, которая определяет номер строки в файле, в которой содержится больше всего пробелов.

**Код программного модуля:**

#include "stdafx.h" //Объявление библиотек

#include "iostream"

#include "fstream"

#include "clocale"

#include "windows.h"

using namespace std;

int Kolichestvo(const char \*c);

//Функция подсчета количества пробелов в строке

void Vivod(string n); //Функция вывода содержимого файлов на экран

void main()

{

int kol = 0, max = 0, numstr, m = 0, kolstr, k = 1;

setlocale(LC\_ALL, "RUS");

char str[120];//Строка, в которую будет считываться содержимое файлов

ifstream f1; //Открытие потоков для записи информации

f1.open("file1.txt", ios::in); //Привязка переменной к файлу

while (!f1.eof()) //Пока не конец первого или второго файла

{

m++;

f1.getline(str, 120); //Считывание строки в переменную str

kol = Kolichestvo(str); //Вызов функции подсчета количества пробелов в строке

if (max < kol)

{

max = kol;

numstr = m;

}

}

f1.close(); //Закрытие файлов

Vivod("file1.txt"); //Функция вывода содержимого файла на экран

cout << endl << endl;

cout << "Максимальное кол - во пробелов в " << numstr << " строке и их кол-во = " << max << endl;

system("pause");

}

# Обработка двоичных файлов

Бинарные (двоичные) файлы – это файлы, в которых информация хранится в двоичном виде, то есть во внутренней форме представления. Двоичные файлы используются для их последующей обработки программными средствами.

**Двоичные файлы** являются файлами *прямого* доступа, то есть мы можем обратиться к любой компоненте файла (указатель текущей позиции файла настраивается нужный байт).

**Режимы открытия файлов при работе с потоками**

В таблице 1 перечислены флаги управления режимами открытия файлов, определенные в классе **ios\_base**. Флаги относятся к типу **openmode** и группируются в битовые маски по аналогии с флагами **fmtflags**.

|  |  |
| --- | --- |
| Таблица 1. **Флаги открытия файлов** | |
| **Флаг** | **Описание** |
| **in** | Открытие файла для чтения (используется по умолчанию для **ifstream**) |
| **out** | Открытие файла для записи (используется по умолчанию для **ofstream**) |
| **арр** | Запись данных производится только в конец файла |
| **ate** | Позиционирование в конец файла после открытия (**"at end"**) |
| **trunc** | Удаление старого содержимого файла |
| **binary** | Специальные символы не заменяются |

Флаг **binary** запрещает преобразование специальных символов или символьных последовательностей (например, конца строки или конца файла). В операционных системах типа **MS-DOS** или **OS/2** конец логической строки в тексте обозначается двумя символами (**CR** и **LF**). При открытии файла в обычном текстовом режиме (сброшенный флаг **binary**) символы новой строки заменяются последовательностью из двух символов, и наоборот. При открытии файла в двоичном режиме (с установленным флагом **binary**) эти преобразования не выполняются.

    Флаг **binary** должен использоваться всегда, когда файл не содержит чисто текстовой информации и обрабатывается как двоичные данные. Пример - копирование файла с последовательным чтением символов и их записью без модификации. Если файл обрабатывается в текстовом виде, флаг **binary** не устанавливается, потому что в этом случае символы новой строки нуждаются в специальной обработке.

    В некоторых реализациях имеются дополнительные флаги типа **nocreate** (файл должен существовать при открытии) и **noreplace** (файл не должен существовать). Однако эти флаги отсутствуют в стандарте, поэтому их использование влияет на переносимость программы.

    Флаги объединяются оператором |. Полученный результат типа **openmode** может передаваться конструктору во втором аргументе. Например, следующая команда открывает файл для присоединения текста в конце:

std::ofstream file("xyz.out", std::ios::out | std::ios::app);

    В таблице 2 представлены различные комбинации флагов и их аналоги - строковые обозначения режимов, используемые функцией открытия файлов **fopen()** в интерфейсе языка **С**. Комбинации с флагами **binary** и **ate** не приводятся. Установленный флаг **binary** соответствует строке с присоединенным символом **b**, а установленный флаг **ate** соответствует позиционированию в конец файла немедленно после открытия. Другие комбинации, отсутствующие в таблице (например, **trunc|app**), недопустимы.

|  |  |  |  |
| --- | --- | --- | --- |
| Таблица 2. **Описание режимов открытия файлов в C++** | | | |
| **Флаги ios\_base** | | **Описание** | **Обозначения режимов в С** |
| **in** | Чтение (файл должен существовать) | | **"r"** |
| **out** | Стирание и запись (файл создается при необходимости) | | **"w"** |
| **out|trunc** | Стирание и запись (файл создается при необходимости) | | **"w"** |
| **out|app** | Присоединение (файл создается при необходимости) | | **"a"** |
| **in|out** | Чтение и запись с исходным позиционированием в начало файла | | **"r+"** |
| **in|out|trunc** | Стирание, чтение и запись (файл создается при необходимости) | | **"w+"** |

Открытие файла для чтения и/или записи не зависит от класса соответствующего объекта потока данных. Класс лишь определяет режим открытия по умолчанию при отсутствии второго аргумента. Это означает, что файлы, используемые только классом **ifstream** или **ofstream**, могут открываться для чтения и записи. Режим открытия передается соответствующему классу потокового буфера, который открывает файл. Тем не менее операции, разрешенные для данного объекта, определяются классом потока данных.

    Также существуют три функции для открытия и закрытия файлов, принадлежащих файловым потокам данных (таблица 3).

|  |  |
| --- | --- |
| Таблица 3. **Функции открытия и закрытия файлов** | |
| **Функция** | **Описание** |
| **ореn(*имя*)** | Открытие файла для потока в режиме по умолчанию |
| **ореn(*имя, флаги*)** | Открытие файла для потока в режиме, определяемом переданными флагами |
| **close()** | Закрытие файлового потока |
| **is\_open()** | Проверка открытия файла |

**Пример работы с бинарными файлами**

Заполнить бинарный файл целыми числами с клавиатуры**.** Найти сумму чётных по значениям компонент файла и записать её в начало файла, сдвинув все компоненты вперёд. Массив не использовать.

int main()

{

setlocale(LC\_ALL, "Rus");

int n = 0; //n - количество чисел

int x;

//открытие бинарного файла на запись

fstream S("first.bin", ios::binary | ios::out | ios::in);

cout << "Введите n: ";

cin >> n;

cout << endl << "Целые числа:" << endl;

//ввод целых чисел и запись их в файл

for (int i = 0; i < n; i++)

{

cin >> x;

S.write((char\*)&x, sizeof(int));//запись целых чисел в файл

}

int pos = 0;

int sum = 0;

S.seekp(0, S.beg); //устанавливаем указатель файла на позицию 0

for (int i = 0; i < n; i++)

{

S.read((char\*)&x, sizeof(int));

if (x % 2 == 0) //если число чётное

{

sum += x;

}

}

cout << endl;

for (int i = n-1; i >= 0; i--) // цикл для сдвига элементов вперёд

{

S.seekp(i\*sizeof(int), S.beg); //устанавливаем указатель файла на позицию i

S.read((char\*)&x, sizeof(int));

S.seekp((i+1)\*sizeof(int), S.beg); //устанавливаем указатель файла на позицию i+1

S.write((char\*)&x, sizeof(int));//запись целых чисел в файл

}

S.seekp(0, S.beg);

S.write((char\*)&sum, sizeof(int)); // запись суммы вместо первой компоненты

cout << endl;

S.seekp(0, S.beg);// читаем файл с начала

for (int i = 0; i < n+1; i++)

{

S.read((char\*)&x, sizeof(int));

cout << x << " ";

}

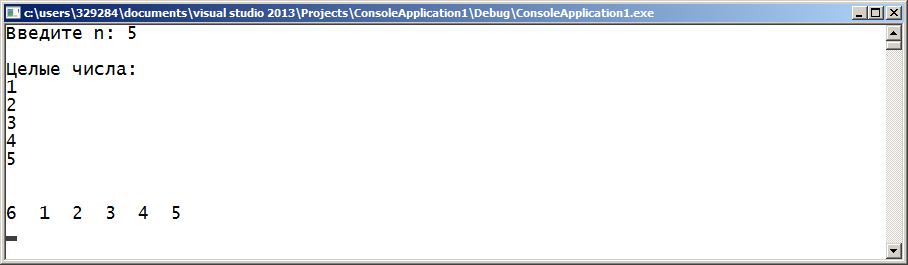
cout << endl;

S.close(); //закрытие потока

\_getch();

return 0;

}



# ВАРИАНТ 1

**Задача 4.1.**

Создать в редакторе текстовый файл, внести в него несколько строк произвольного текста. Разработать *алгоритм* и программу, которая:

* подсчитывает количество цифр, встречающихся в каждой строке
* выводит на экран те строки файла, в которых встречаются цифры
* формирует другой файл, в который переносит строки исходного файла,   
  не содержащие цифры

**Задача 4.2.**

Разработать *алгоритм* и программу для заполнения двоичного файла целыми числами в интервале [-200; 500]. Количество компонент файла вводится с клавиатуры. Найти количество и сумму нечетных чисел. Количество нечетных чисел записать перед предпоследней компонентой, сдвинув два значения в файле вперёд, а сумму – в конец файла. Массив не использовать.

# ВАРИАНТ 2

**Задача 4.1.**

Создать в редакторе текстовый файл, внести в него несколько строк произвольного текста. Разработать *алгоритм* и программу, которая:

* выводит на экран строки файла с k-ой по n-ую
* подсчитывает количество строк файла, в которых встречаются цифры
* формирует другой файл, в который переписывает строки исходного файла в обратном порядке

**Задача 4.2.**

Разработать *алгоритм* и программу для заполнения двоичного файла целыми числами в интервале [-350; 350]. Количество компонент файла вводится с клавиатуры. Поменять местами компоненты с максимальным значением (таких может быть и несколько), и последнюю компоненту. Массив не использовать.

# ВАРИАНТ 3

**Задача 4.1.**

Создать в редакторе текстовый файл, внести в него несколько строк произвольного текста. Разработать *алгоритм* и программу, которая:

* выводит на экран строки файла, заканчивающиеся на заданные символы
* формирует другой файл, в который переписывает только те строки исходного файла, в которых встречается заданный слог

**Задача 4.2.**

Разработать *алгоритм* и программу для заполнения двоичного файла целыми числами в интервале [-150; 150]. Количество компонент файла вводится с клавиатуры. Удвоить значения отрицательных компонент в файле, стоящих после последнего минимального значения. Массив не использовать.

# ВАРИАНТ 4

**Задача 4.1.**

В текстовом файле хранятся несколько вопросов, на которые можно отвечать «да» или «нет», после каждого вопроса на следующей строке хранится правильный ответ. Разработать *алгоритм* и программу, которая считывает и выводит на экран вопросы из файла, получает ответы пользователя, подсчитывает количество правильных ответов и выставляет оценку по сто балльной системе.

**Задача 4.2.**

Разработать *алгоритм* и программу для заполнения двоичного файла целыми числами в интервале [-150; 150]. Количество компонент файла вводится с клавиатуры. Обнулить компоненты файла с максимальным значением (таких может быть несколько). Массив не использовать.

# ВАРИАНТ 5

**Задача 4.1.**

Создать в редакторе текстовый файл, внести в него несколько строк произвольного текста. Разработать *алгоритм* и программу, которая:

* выводит на экран строки файла, в которых встречается заданный символ
* формирует другой файл, в который переписывает строки исходного файла, содержащие латинские буквы, и указывает после каждой строки количество латинских букв в ней.

**Задача 4.2.**

Разработать *алгоритм* и программу для заполнения двоичного файла случайными целыми числами в интервале [1; 500]. Найти компоненты файла, являющимися двузначными числами, и переместить их в начало файла. Массив не использовать.

# ВАРИАНТ 6

**Задача 4.1.**

Создать в редакторе текстовый файл, внести в него несколько строк произвольного текста. Разработать *алгоритм* и программу, которая:

* выводит на экран строки файла, в которых встречается более 1 цифры
* формирует другой файл, в который переписывает строки исходного файла, заменив в каждой строке сочетания цифр на одну «**\***»

**Задача 4.2.**

Разработать *алгоритм* и программу для заполнения двоичного файла целыми числами в следующем порядке: 10 положительных, 10 отрицательных, 10 положительных, 10 отрицательных и т.д. (количество элементов файла кратно 20, вводиться с клавиатуры). Получить новый файл из исходного, компоненты которого идут в таком порядке: 5 положительных, 5 отрицательных и т.д. Массив не использовать.

# ВАРИАНТ 7

**Задача 4.1.**

Создать в редакторе текстовый файл, внести в него несколько строк произвольного текста. Разработать *алгоритм* и программу, которая:

* выводит на экран строки файла, в которых встречается не более 1 пробела
* формирует другой файл, в который переписывает строки исходного файла, удалив из каждой строки все пробелы

**Задача 4.2.**

Разработать *алгоритм* и программу для заполнения двоичного файла целыми значениями, вводимыми с клавиатуры, окончание ввода – пустая строка. Проверить, стоит ли в файле минимальное значение раньше максимального, и если это не так, поменять их местами. Массив не использовать.

# ВАРИАНТ 8

**Задача 4.1.**

Создать в редакторе текстовый файл, внести в него несколько строк произвольного текста. Разработать *алгоритм* и программу, которая:

* выводит на экран строки файла, от N-ой до конца файла
* формирует другой файл, в который переписывает только те строки исходного файла, в которых нет пробелов.

**Задача 4.2.**

Разработать *алгоритм* и программу для заполнения двоичного файла числами Фибоначчи (первые две компоненты будут равны 1, каждая следующая компонента вычисляется как сумма двух предыдущих). Количество компонент вводится с клавиатуры. Программа должна также выводить на экран любое из чисел Фибоначчи с заданным номером, прочитывая его из файла, если компонента с таким номером есть в файле. Массив не использовать.

# ВАРИАНТ 9

**Задача 4.1.**

Составить программу, которая находит периметр многоугольника, заданного при помощи координат вершин. Координаты вершин многоугольника должны задаваться последовательно по периметру, и храниться в тестовом файле, каждая точка с новой строки.

**Задача 4.2.**

Разработать *алгоритм* и программу для заполнения двоичного файла числами арифметической прогрессии с первым элементом, равным *x* и разностью *r*, число компонент по желанию пользователя. Программа должна также выводить два ближайших члена арифметической прогрессии, между которыми находится заданное число, и вставлять это число в файл на найденное место (при этом часть элементов файла сдвигается вперёд). Массив не использовать.

# ВАРИАНТ 10

**Задача 4.1.**

В текстовом файле хранятся результаты измерений температуры воздуха в течение недели: на каждой строке результат измерения температуры днем и через пробел – температуры ночью. Вычислить среднюю температуру воздуха в дневное время и среднюю температуру воздуха в ночное время по имеющимся данным. Дополнительный массив не использовать.

**Задача 4.2.**

Заполнить двоичный файл целыми числами из интервала [-12 000; 12 000]. Количество компонент файла задать с клавиатуры. Затем увеличить вдвое компоненты файла, стоящие после максимального элемента. Если максимальных элементов в файле несколько, предварительно удалить повторяющиеся максимальные значения, оставив только одно. Массив не использовать.

# ВАРИАНТ 11

**Задача 4.1.**

В текстовом файле хранятся результаты измерений артериального давления пациента в течение недели: на каждой строке через пробел указано систолическое давление (верхняя цифра) и диастолическое давление (нижняя цифра). Определить средние цифры давления пациента по результатам измерений и если они более 130 и 80 соответственно, выдать сообщение, какое давление повышено. Дополнительный массив не использовать.

**Задача 4.2.**

Заполнить двоичный файл целыми числами из интервала [-16 000; 16 000]. Количество компонент файла вводится с клавиатуры. Увеличить на 100 компоненты файла, стоящие до минимального элемента. Если минимальных элементов в файле несколько, предварительно удалить повторяющиеся минимальные значения, оставив только одно. Массив не использовать.

# ВАРИАНТ 12

**Задача 4.1.**

В текстовом файле хранятся слова, каждое слово на отдельной строке. Найти и вывести на экран слова, включающие все буквы заданного слова плюс любые другие.

**Задача 4.2.**

Заполнить двоичный файл целыми числами из интервала [-3000; 8000]. Количество компонент файла вводится с клавиатуры. Поменять местами в файле первый четный по значению компонент и последний четный, первый нечётный компонент и последний нечётный. Массив не использовать.

# ВАРИАНТ 13

**Задача 4.1.**

В текстовом файле хранятся слова, каждое слово на отдельной строке. Найти в файле слова, начинающиеся на заданный слог. Если таких слов нет, выдать сообщение об этом.

**Задача 4.2.**

Заполнить двоичный файл целыми числами из интервала [-4000; 7000]. Количество компонент файла вводится с клавиатуры. Поменять местами в файле первый положительный элемент с максимальным значением (максимальных может быть несколько). Массив не использовать.

# ВАРИАНТ 14

**Задача 4.1.**

В текстовом файле хранятся слова, каждое слово на отдельной строке. Найти и вывести на экран слова, включающие некоторый слог, введенный пользователем, плюс любые другие буквы.

**Задача 4.2.**

Заполнить двоичный файл целыми числами из интервала [-2000; 5000]. Количество компонент файла вводится с клавиатуры. Поменять местами в файле первый отрицательный компонент с минимальным значением (минимальных может быть несколько). Массив не использовать.

# ВАРИАНТ 15

**Задача 4.1.**

Создать в редакторе текстовый файл, внести в него несколько строк произвольного текста. Разработать *алгоритм* и программу, которая:

* выводит на экран первые n строк файла
* подсчитывает количество строк в файле, начинающихся на заданный символ
* формирует другой файл, в который переносит строки исходного файла, вставляя пустую строку после каждых двух строк

**Задача 4.2.**

Разработать *алгоритм* и программу для заполнения двоичного файла целыми числами в интервале [-300; 100]. Количество компонент файла вводится с клавиатуры. Найти сумму положительных чисел, хранящихся в файле, и записать ее перед предпоследней компонентой файла (при этом два значения сдвинуть вправо). Массив не использовать.

# ВАРИАНТ 16

**Задача 4.1.**

Создать в редакторе текстовый файл, внести в него несколько строк произвольного текста. Разработать *алгоритм* и программу, которая:

* определяет номер строки в файле, в которой содержится больше всего пробелов
* выводит на экран последние n строк файла
* формирует другой файл, в который переносит строки исходного файла, добавляя символ # перед каждой третьей строкой

**Задача 4.2.**

Разработать *алгоритм* и программу для заполнения двоичного файла целыми числами в интервале [-600; 600]. Количество компонент файла вводится с клавиатуры. Найти сумму четных чисел, хранящихся в файле, и записать ее перед компонентой с заданным номером (при этом элементы, начиная с указанного номера, сдвинуть вправо). Массив не использовать.

# ВАРИАНТ 17

**Задача 4.1.**

Создать в редакторе два текстовых файла, внести в них по 10 строк произвольного текста. Разработать *алгоритм* и программу, которая:

* выводит на экран каждую вторую строку первого файла
* формирует новый файл из двух исходных файлов, чередуя строки первого и второго файла

**Задача 4.2.**

Разработать *алгоритм* и программу для заполнения двоичного файла случайными целыми числами в интервале [-250; 250]. Количество компонент файла вводится с клавиатуры. Подсчитать сумму отрицательных чисел, хранящихся в файле, и вставить перед последним отрицательным значением полученную сумму. Массив не использовать.

# ВАРИАНТ 18

**Задача 4.1.**

Создать в редакторе текстовый файл, внести в него несколько строк произвольного текста. Разработать *алгоритм* и программу, которая:

* выводит на экран последние n строк файла
* подсчитывает количество строк в файле, включающих заданный символ
* формирует другой файл, в который переносит строки исходного файла, включающие заданный символ, вставляя пустую строку после каждой из таких строк.

**Задача 4.2.**

Разработать *алгоритм* и программу для заполнения двоичного файла случайными целыми числами в интервале [-500; 500]. Количество компонент файла вводится с клавиатуры. Найти сумму компонент первой половины файла, и записать ее перед компонентой файла, стоящей посередине (при этом оставшиеся значения сдвинуть вправо). Массив не использовать.

# ВАРИАНТ 19

**Задача 4.1.**

Создать в редакторе текстовый файл, внести в него несколько строк произвольного текста. Разработать *алгоритм* и программу, которая:

* определяет номер строки в файле, в которой содержится больше всего знаков препинания
* выводит на экран n строк файла, начиная со строки с указанным номером
* формирует другой файл, в который переносит строки исходного файла, добавляя символ \* после каждой второй строки

**Задача 4.2.**

Разработать *алгоритм* и программу для заполнения двоичного файла целыми числами в интервале [-400; 400]. Количество компонент файла вводится с клавиатуры. Найти сумму нечетных чисел, хранящихся в файле, и записать ее после компоненты с заданным номером (при этом элементы, начиная с указанного номера, сдвинуть вправо). Массив не использовать.

# ВАРИАНТ 20

**Задача 4.1.**

Создать в редакторе два текстовых файла, внести в них по 10 строк произвольного текста. Разработать *алгоритм* и программу, которая:

* выводит на экран каждую третью строку первого файла
* формирует новый файл из двух исходных файлов, чередуя каждые три строки первого и второго файла

**Задача 4.2.**

Разработать *алгоритм* и программу для заполнения двоичного файла случайными целыми числами в интервале [-150; 150]. Количество компонент файла вводится с клавиатуры. Подсчитать сумму четных чисел, хранящихся в файле, и вставить перед последним четным значением полученную сумму. Массив не использовать.