Лабораторная работа № 20

«Разработка алгоритмов и программ с использованием бинарных файлов. Сравнение работы с текстовым и бинарным файлами в нотации С++»

Сделать **1** программу с меню, реализующем заданный ниже функционал.

Пункты 6 и 8, 7 и 9 можно попарно объединить, то есть одна структура будет записываться в текстовый и бинарный файлы одновременно и также из текстового файла и из бинарного будет считываться очередная структура и печататься на консоль.

В этой лабораторной работе **запись файлов и чтение из файлов осуществлять в нотации С++**. Если лабораторная работа № 19 была выполнена в нотации С++, то, наоборот, все задания данной лабораторной работы № 20 выполнить в нотации POSIX (синтаксис си)

Имя файла (путь к файлу) можно запросить у пользователя, а не использовать готовое имя файла в коде.

Пример работы с бинарным файлом в нотации С++:

Создадим программу для записи текстового файла в текстовом режиме записи данных в файл. Пусть пользователь вводит вещественные числа, которые должны записаться в файл, а между числами будет разделительный символ, например, пробел ‘ ‘. Будем использовать нотацию собственно языка С++, для чего надо подключить библиотеку fstream – file stream – (англ.) «файловый поток», то есть поддержка потоков для работы с файлами (чтения, записи, дозаписи файлов). В программе f – это поток записи данных в текстовый файл, а не сам файл, полное имя которого (это полный путь к нему на компьютере) "D:\\2019\\Labs\\Lab6_0\\File.txt". Как и в нотации POSIX (лабораторная работа № 19), после работы с файлом надо сразу закрыть файл, то есть закрыть открытый поток работы с ним.

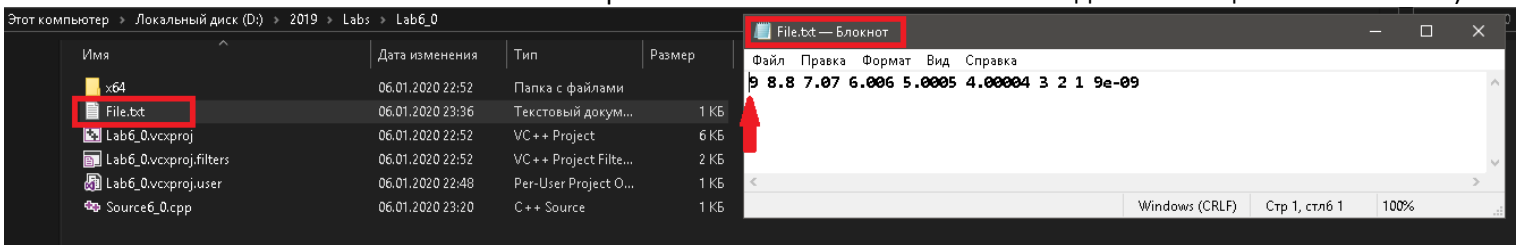
```cpp
#include <iostream>
#include <fstream>//подключить библиотеку fstream для чтения-записи файлов в нотации С++
#include <Windows.h>
using namespace std;

int main()
{
    SetConsoleOutputCP(1251);
    SetConsoleCP(1251);
    ofstream f("D:\\2019\\Labs\\Lab6_0\\File.txt", ios::out | ios::trunc);//создать поток f для записи файла (output stream - поток для вывода данных из оперативной памяти в файл)
    //с именем D:\\2019\\Labs\\Lab6_0\\File.txt в режимах ios::out (запись файла) и ios::trunc (если файл с таким именем по такому пути уже существует, то перезаписать его, удалив
    //"старое" содержимое)
    if (!f)//если не удалось создать поток для записи в файл (не удалось открыть или создать файл с указанным именем по указанному пути, не удалось открыть файл в нужном режиме)
    {
        cout << "Файл D:\\2019\\Labs\\Lab6_0\\File.txt для записи не удалось открыть.\n";
        system("pause");
        return 0;//завершаем программу, поскольку нет файла для работы
    }
    int n;//если выполнение программы дошло до этой строки кода, значит требуемый файл удалось открыть в нужном режиме. Создаем переменную n для хранения количества вводимых чисел
    do
    {
        cout << "Введите количество чисел (>0), которые вы хотите записать в файл: ";
        cin >> n;
    }
    while (n<1);//если пользователь вводит некорректные данные, посим его ввести заново количество чисел, которое нужно записать в файл
    double x;//записывать будем вещественные числа, поэтому создадим одну переменную для хранения одного вводимого вещественного числа
    for (int i = 0; i < n; i++)
    {
        cout << "Введите число № " << i + 1 << " для записи в файл: ";
        cin >> x;
        f << x << ' ';//собственно запись информации в текстовый файл в нотации С++. За одну итерацию в файл (поток для записи f) записывается одно число (значение переменной x) и
    }//сразу за ним записывается символ ПРОБЕЛ. Курсор остается стоять сразу за пробелом и в следующей итерации начнется запись с этого места, где стоит курсор
    f.close();//после работы с файлом, закрываем файл (закрываем поток f для записи файла)
    cout << "Файл D:\\2019\\Labs\\Lab6_0\\File.txt записан и закрыт.\n";//сообщаем пользователю об окончании записи файла
    system("pause");
    return 0;
}
```

Тестируем работу программы. Специально введем числа с различной дробной частью, чтобы увидеть, запишется ли она в файл:

```
D:\2019\Labs\x64\Debug\Lab6_0.exe
Введите количество чисел, которые вы хотите записать в файл: 10
Введите число № 1 для записи в файл: 9
Введите число № 2 для записи в файл: 8.8
Введите число № 3 для записи в файл: 7.07
Введите число № 4 для записи в файл: 6.006
Введите число № 5 для записи в файл: 5.0005
Введите число № 6 для записи в файл: 4.00004
Введите число № 7 для записи в файл: 3.000003
Введите число № 8 для записи в файл: 2.0000002
Введите число № 9 для записи в файл: 1.00000001
Введите число № 10 для записи в файл: 0.0000000009
Файл D:\2019\Labs\Lab6_0\File.txt записан и закрыт.
Для продолжения нажмите любую клавишу . . .
```

Откроем в программе «Блокнот» записанный файл. Числа 3, 2 и 1 записались без дробной части, которая могла «потеряться» или при вводе (если количество цифр в дробной части превышает «лимит», установленный в операционной системе по умолчанию) или ввиду того, что превышен размер числа, который можно поместить в переменную вещественного типа (8 байт). Последнее число операционная система записала в научной нотации (scientific notation) 9e-09 = $9*10^{-9}$ = 0.000000009 (значение правильное, а запись такую выбрала система, поскольку такая запись числа занимает меньше места по сравнению с «классической» записью данного вещественного числа).



Теперь напишем программу для чтения текстового файла, куда записаны вещественные числа с пробелами между ними. Разумеется, мы не знаем, сколько именно чисел записано в файл, поэтому читаем файл до тех пор, пока есть что из него читать, то есть пока не достигнут маркер EOF – End of file – маркер-флаг «Конец файла». По сути дела, EOF – это терминатор конца файла, EOF-терминатор. В нотации C++ у потока посредством оператора доступа «точка» вызывается функция eof(), которая в случае ее вызова проверяет, не достигнут ли конец файла, то есть маркер EOF. Если маркер достигнут, то функция eof() возвращает значение true, иначе – false.

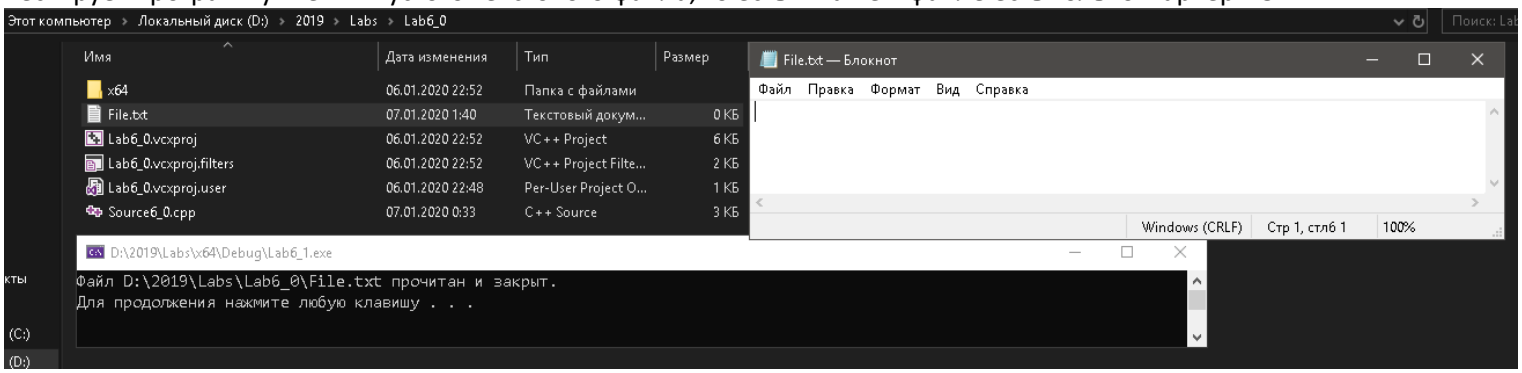
```cpp
#include <iostream>
#include <fstream>//подключить библиотеку fstream для чтения-записи файлов в нотации C++
#include <Windows.h>
using namespace std;

int main()
{
    SetConsoleOutputCP(1251);
    SetConsoleCP(1251);
    ifstream f("D:\\2019\\Labs\\Lab6_0\\File.txt", ios::in);//создать поток для чтения файла (ifstream - input stream - входной поток, поток для ввода данных в оперативную память
    //из файла; данные читаются из файла и помещаются в оперативную память. Режим ios::in - режим чтения текстового файла
    if (!f)//если не удалось создать поток для чтения из файла (не удалось найти или открыть файл с указанным именем по указанному пути, не удалось открыть файл в режиме для чтения
    {
        cout << "Файл D:\\2019\\Labs\\Lab6_0\\File.txt для чтения не удалось открыть.\n";//сообщаем об этом пользователю
        system("pause");
        return 0;//и завершаем работу программы
    }
    double y;//созадим переменную вещественного типа, в которую будем помещать считываемое из файла число
    f >> y;//ПЕРЕД циклом читаем из файла одно число и сохраняем это число в переменную y. Такой подход исключает печать последнего прочитанного числа дважды. Тут нотация C++
    while (!f.eof())//выполнять тело цикла, пока курсор чтения не достигнет конца файла. Функция eof() проверяет, не достиг ли курсор чтения конца файла, то есть маркера EOF - End
    {// of file. Функция eof() вызывается у объекта (потока чтения файла) f посредством оператора доступа "точка" '.'. Аналогично можно открыть другие файлы и проверить их
        cout << y << endl;//печатаем содержимое перменной y, в которую помещено прочитанное из файла значение
        f >> y;//читаем из открытого потока чтения файла следующее число. Разделитель ПРОБЕЛ используется для определения границ чисел при чтении по умолчанию, но если бы
    }//использовался другой разделитель, например '|', то его бы следовало читать в отдельную символьную переменную: char c; f >> y >> c;, чтобы курсор чтения сдвигался к
    //следующему числу и в следующей итерации фактические данные в файле корректно считались в подходящую по типу данных переменную и так далее
    f.close();//закрываем поток f чтения файла, закрывая тем самым файл D:\\2019\\Labs\\Lab6_0\\File.txt
    cout << "Файл D:\\2019\\Labs\\Lab6_0\\File.txt прочитан и закрыт.\n";//сообщаем пользователю о завершении чтения файла и его закрытии (наша программа освободила ресурс - файл)
    //печать данной фразы важна, поскольку укажет пользователю о завершении чтения файла в случае, если файл для чтения открылся, но оказался пустым (без данных или с некорректными
    //данными, которые не подошли для чтения в том формате, как написан код нашей программы)
    system("pause");
    return 0;
}
```

Тестирует программу чтения текстового файла:

```
D:\2019\Labs\x64\Debug\Lab6_1.exe
9
8.8
7.07
6.006
5.0005
4.00004
3
2
1
9e-09
Файл D:\2019\Labs\Lab6_0\File.txt прочитан и закрыт.
Для продолжения нажмите любую клавишу . . .
```

Тестирует программу чтения пустого текстового файла, то есть в нашем файле есть только маркер EOF:



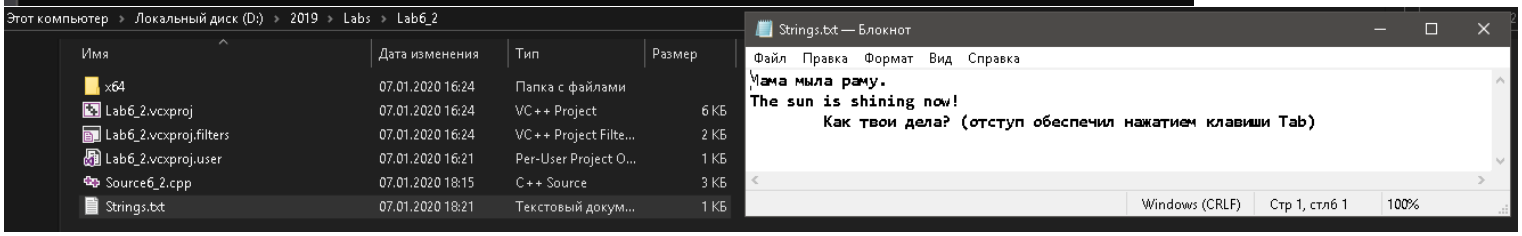
Построчная запись текста в текстовый файл

Напишем программу записи текста в файл. Программа поддерживает русский язык, записывает текст в файл или дописывает новый текст к старому в уже существующем текстовом файле. Текст может содержать пробелы.

```cpp
#include <iostream>
#include <fstream>//подключить библиотеку fstream для чтения-записи файлов в нотации C++
#include <Windows.h>
using namespace std;
int main()
{
    SetConsoleOutputCP(1251);
    SetConsoleCP(1251);
    int n;
    do
    {
        cout << "Сколько строк (абзацев) текста вы хотите ввести (>0): ";
        cin >> n;
    } while (n < 1);
    char s[100];//создадим символьный массив размером до 100 символов, в который будем помещать строку, которую будет вводить с клавиатуры пользователь
    char file[] = "D:\\2019\\Labs\\Lab6_2\\Strings.txt";//символьный массив размером 32 символа (31 символ + ноль-терминатор '\0'). Сдвоенные наклонные линии \\ считаются за один
    //символ, поскольку первый из пары наклоненных назад слэшей является экранирующим (управляющим, техническим, вспомогательным символом) и не считается видимым символом
    ofstream f(file, ios::out | ios::app);//создать поток для записи (режим ios::out) и дозаписи (режим ios::app) в конец файла новых строк в файл с именем, хранимым в массиве file
    if (!f)//если не удалось создать поток f для записи (дозаписи) в файл
    {
        cout << "Файл " << file << " для записи (дозаписи) не удалось открыть.\n";//сообщаем об этом пользователю
        system("pause");
        return 0;//и завершаем работу программы
    }
    cin.ignore();//нейтрализуем последний нажатый пользователем Enter. Сделаем это ДО цикла, поскольку, если это выражение поместить в цикл, то во второй и последующих итерациях
    //цикла пропустится первая введенная буква в строке, ведь это выражение тогда будет работать в КАЖДОЙ итерации, а нам нужен этот эффект только перед первой итерацией
    for (int i = 0; i < n; i++)//обеспечим ввод в цикле затребованного пользователем количества строк и запись их в файл (поток f для записи в файл)
    {
        cout << "Введите строку № " << i + 1 << " размером до 100 символов:\n";
        cin.getline(s, 100);//можно вводить текст с пробелами. Ввод строки заканчивается нажатием кнопки Enter
        f << s << endl;//в открытый файловый поток записываем содержимое символьного массива s и переводим курсор на новую строку сразу после этого
    }
    f.close();//закрываем файловый поток f записи (дозаписи) файла, закрывая тем самым файл с именем, хранимом в символьном массиве file
    cout << "Файл " << file << " записан (дозаписан) и закрыт.\n";//сообщаем пользователю о завершении записи и закрытии файла - файл теперь освобожден нашей программой
    system("pause");
    return 0;
}
```

Тестируем. Сначала введем 3 строки текста, используем один табуляционный пробел для имитации отступа в начале абзаца текста:

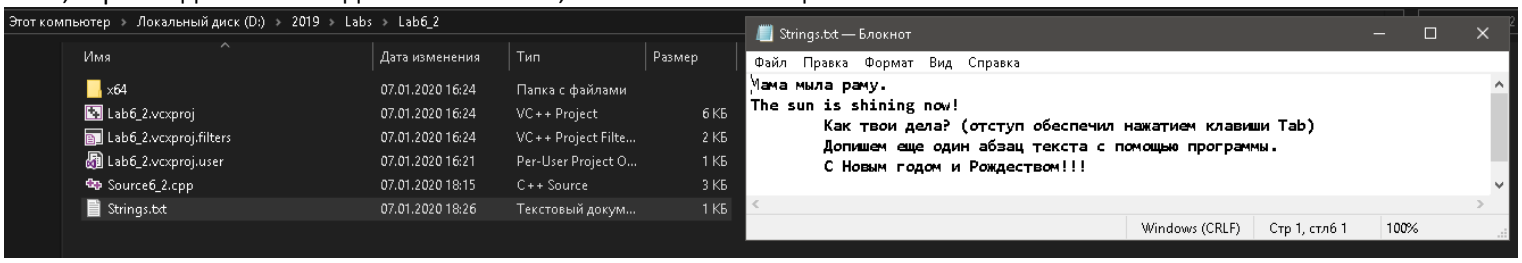
```
D:\2019\Labs\x64\Debug\Lab6_2.exe
Сколько строк (абзацев) текста вы хотите ввести (>0): 3
Введите строку № 1 размером до 100 символов:
Мама мыла раму.
Введите строку № 2 размером до 100 символов:
The sun is shining now!
Введите строку № 3 размером до 100 символов:
        Как твои дела? (отступ обеспечил нажатием клавиши Tab)
Файл D:\2019\Labs\Lab6_2\Strings.txt записан (дозаписан) и закрыт.
Для продолжения нажмите любую клавишу . . .
```



Теперь допишем текст в файл с помощью нашей программы:

```
D:\2019\Labs\x64\Debug\Lab6_2.exe
Сколько строк (абзацев) текста вы хотите ввести (>0): 2
Введите строку № 1 размером до 100 символов:
        Допишем еще один абзац текста с помощью программы.
Введите строку № 2 размером до 100 символов:
        С Новым годом и Рождеством!!!
Файл D:\2019\Labs\Lab6_2\Strings.txt записан (дозаписан) и закрыт.
Для продолжения нажмите любую клавишу . . .
```

Итак, в файле должен находиться весь текст, то есть все 5 абзацев. Так и есть:



Построчное чтение текста из текстового файла

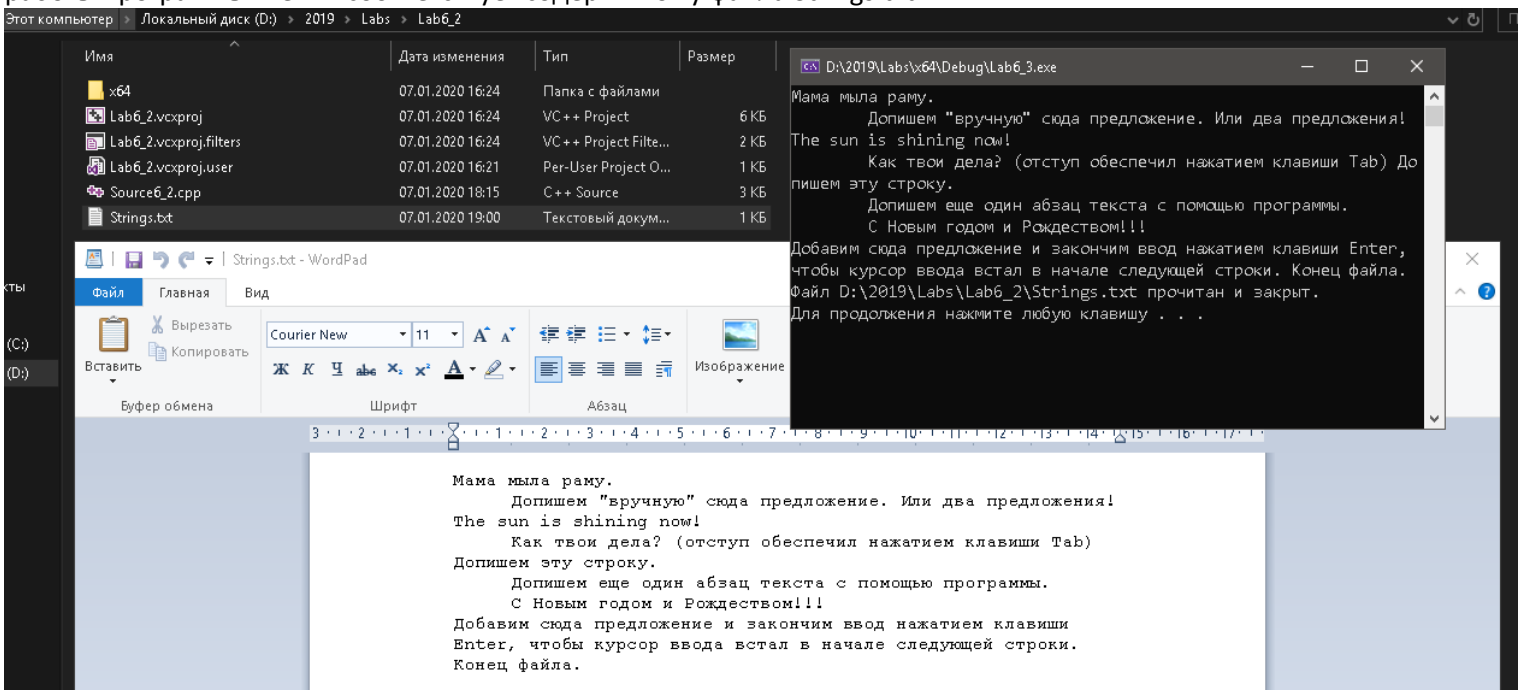
Напишем программу для построчного чтения из текстового файла в нотации C++. Программа не знает, сколько строк в файле есть, поэтому она должна читать построчно файл до тех пор, пока не встретит маркер EOF – терминатор «конец файла». Также в коде закомментирован способ читать файл не по строкам (разделитель Enter), а по словам (разделитель – символ пробел).

```
D:\2019\Labs\x64\Debug\Lab6_3.exe
Мама мыла раму.
The sun is shining now!
        Как твои дела? (отступ обеспечил нажатием клавиши Tab)
        Допишем еще один абзац текста с помощью программы.
        С Новым годом и Рождеством!!!
Файл D:\2019\Labs\Lab6_2\Strings.txt прочитан и закрыт.
Для продолжения нажмите любую клавишу . . .
```

```cpp
#include <iostream>
#include <fstream>//подключить библиотеку fstream для чтения-записи файлов в нотации C++
#include <Windows.h>
using namespace std;

int main()
{
    SetConsoleOutputCP(1251);
    SetConsoleCP(1251);
    char str[100];
    char file[] = "D:\\2019\\Labs\\Lab6_2\\Strings.txt";//символьный массив размером 32 символа (31 символ + ноль-терминатор '\0'). Сдвоенные наклонные линии \\ считаются за один
    //символ, поскольку первый из пары наклоненных назад слешей является экранирующим (управляющим, техническим, вспомогательным символом) и не считается видимым символом
    ifstream f(file, ios::in);//создадим поток f для чтения (режим ios::in) из текстового файла, имя которого хранится в символьном массиве file
    if (!f)//если не удалось создать поток f для чтения файла
    {
        cout << "Файл " << file << " для чтения не удалось открыть.\n";//сообщаем об этом пользователю
        system("pause");
        return 0;//и завершаем работу программы
    }
    //f >> str;//чтение из файлового потока f по отдельным словам (разделителем считается пробел или Enter, в зависимости от того, что из них встретится раньше). Такое выражение
    //прочитает из файла только одно слово (то, что находится между положением текущего курсора до ПЕРВОГО встреченного правее пробела или Enter'а)
    f.getline(str, 100, '\n');//из файлового потока f читается(берется) строка символов, которая отрезается по символу '\n' и помещается в массив str размером не более 100 символов
    while (!f.eof())//мы не знаем, сколько строк в файле, поэтому читаем до тех пор, пока не достигнем конца файла - маркера EOF
    {
        cout << str << endl;//печать содержимого массива str
        f.getline(str, 100, '\n');//чтение из потока f следующей строки символов файла и помещение ее в символьный массив str. Это выражение есть ДО цикла и в нем, чтобы не
    }//допустить печати на консоль последней строки дважды
    f.close();//по окончании работы с файлом закрываем поток чтения файла f и тем самым закрываем файл
    cout << "Файл " << file << " прочитан и закрыт.\n";//сообщаем пользователю о завершении чтения и закрытии файла
    system("pause");
    return 0;
}
```

Наконец, вручную в программе «Блокнот» допишем файл и попробуем его прочитать в нашей программе. Я дописал в файл достаточно длинные строки, длина некоторых из которых **превышает 100 символов**, поэтому надо в программе построчного чтения файла **заменить размер буфера** символьного массива со 100 символов **на 200 во всех местах** (или однократно, если использовать целочисленную константу, что более оптимально). После этих изменений результат работы программы чтения соответствует содержимому файла Strings.txt.



Запись экземпляров структур в бинарные и текстовые файлы в нотации C++

Мы возьмем пример программы записи экземпляров структур struct из лабораторной работы № 19, но вместо нотации POSIX будем использовать нотацию собственно языка C++, то есть перепишем соответствующие участки кода «старой» программы.

```
#include <iostream>
#include <Windows.h>//для обеспечения поддержки русского языка, в том числе при записи и чтении файлов (по крайней мере в ОС Windows)
#include <fstream>//нужно подключить эту библиотеку для обеспечения потокового ввода-вывода в файлы (из файлов)
using namespace std;

struct Human//структура Human объявлена глобально, хотя было достаточно объявить ее в начале функции main()
{
    int age;
    double height;//измеряется в метрах
    char fam[20];
    char name[15];
    bool sex;//0-female, 1-male
} h;//тоже самое, что написать отдельно: Human h;//декларация переменной типа структуры Human
bool f = false;//глобальная (видимая всем в этой программе) переменная-флаг для проверки, заполнены поля экземпляра h типа структуры Human (true-да) или нет (false)

int main()
{
    SetConsoleOutputCP(1251);
    SetConsoleCP(1251);
    char binFileName[] = "D:\\2019\\Labs\\Lab6_4\\BinFileWithStructs.dat";//сразу создадим символьный массив, в который поместим имя бинарного файла
    char textfileName[] = "D:\\2019\\Labs\\Lab6_4\\TextFileWithStructs.txt";//сразу создадим символьный массив, в который поместим имя текстового файла
    int p = -1;
    while (p != 0)//цикл остановится, когда пользователь введет пункт меню 0
    {
        cout << "Меню:\n0-Завершить программу\n1-Ввести данные о следующем человеке в оперативную память\n2-Записать (дописать в конец бинарного файла) данные о текущем человеке \
из оперативной памяти\n3-Читать и поочередно распечатывать на консоль данные из бинарного файла о всех записанных в нем людях\n4-Записать (дописать в конец текстового файла) \
данные о текущем человеке\n5-Читать и поочередно распечатывать на консоль данные из текстового файла о всех записанных в нем людях\nВведите номер нужного вам пункта меню: ";
        cin >> p;//пользователь с клавиатуры выбирает пункт меню, в том числе и вариант завершения работы этой программы
        switch (p)
        {
        case 0://если p==0
        {
            cout << "Завершение работы программы.\n";
            break;
        }
        case 1://ввод данных о человеке
        {
```

```
            cout << "Введите возраст человека (полных лет): ";
            cin >> h.age;
            cout << "Введите рост человека (в метрах): ";
            cin >> h.height;
            cout << "Введите фамилию человека (можно с пробелами): ";
            cin.ignore();//внутри кейса нужно, так как иначе cin.getline(h.fam, 20); успевает "словить" нажатие пользователем клавиши Enter при вводе пункта 1
            cin.getline(h.fam, 20);
            cout << "Введите имя человека: ";
            cin >> h.name;
            cout << "Введите пол человека (0-женский, 1-мужской): ";
            cin >> h.sex;
            cout << "Вы ввели данные:\n" << h.fam << "\t|\t" << h.name << "\t|\t" << h.age << "\t|\t" << h.height << "\t|\t";//проверяем сразу введенные данные
            if (h.sex == 0)
            {
                cout << "женский\n";
            }
            else
            {
                cout << "мужской\n";
            }
            f = true;//поля экземпляра структуры h заполнены -> можно значения этих полей записывать в бинарный и (или) текстовый файлы
            break;
        }
        case 2:
        {
            if (f == false)//если экземпляр h структуры Human не содержит данных в своих полях
            {
                cout << "Нет данных для записи.\n";//то записывать в файл нечего и кейс завершаем сразу
                break;
            }
            ofstream wbf(binFileName, ios::out | ios::app | ios::binary);//создадим поток для записи (режим ios::out), дозаписи к уже существующим данным в файле (режим ios::app)
            //для бинарного файла (режим работы с бинарный файлом ios::binary). Имя файла хранится в символьном массиве binFileName
            if (!wbf)//если поток работы wbf для записи (дозаписи) бинарного файла не удалось создать
            {
                cout << "Error.\n";
                break;//выходим из кейса, а не программы
            }
            wbf.write((char*)&h, sizeof(h));//у потока wbf вызываем функцию (метод) write() для бинарной записи всей структуры h размером sizeof(h) или sizeof(Human) в файл
            wbf.close();//сразу закрываем поток записи в бинарный файл, так как пока писать больше нечего
            cout << "Файл " << binFileName << " записан (дозаписан) и закрыт.\n";//сообщаем пользователю
            break;//завершаем кейс 2
        }
        case 3:
        {
            Human h0;//создать отдельную переменную h0, чтобы использовалась "пустая" локальная переменная и мы гарантировано видели результат работы функции бинарного чтения файла
            ifstream rbf(binFileName, ios::in | ios::binary);//создадим поток для чтения (режим ios::in) бинарного файла в бинарном режиме ios::binary
            if (!rbf)//если поток rbf чтения данных из бинарного файла не удалось создать
            {
                cout << "Error.\n";
                break;//завершаем кейс
            }
            rbf.read((char*)&h0, sizeof(Human));//у потока rbf вызываем функцию (метод) бинарного чтения read(), которая считывает sizeof(Human) байт из бинарного файла и помещает
            //их по адресу переменной h0 типа структуры Human
            while (!rbf.eof())//пока не достигнем конца файла, открытого в потоке rbf, выполнять в цикле нижеследующий код итераций
            {
                cout << h0.fam << "\t|\t" << h0.name << "\t|\t" << h0.age << "\t|\t" << h0.height << "\t|\t";//распечатывать значения полей уже заполненного объекта h0
                if (h0.sex == 0)//пол Человека печатаем на консоль пользователю в понятном для него виде
                {
                    cout << "женский\n";
                }
                else//то есть если if(h0.sex==1)
                {
                    cout << "мужской\n";
                }
                rbf.read((char*)&h0, sizeof(Human));//читать следующие байты в количестве sizeof(Human) из файла, открытого в потоке rbf чтения бинарного файла
            }
            rbf.close();//закрываем поток rbf чтения бинарного файла и, соответственно, сам файл тоже закрываем
            cout << "Файл " << binFileName << " прочитан и закрыт.\n";//сообщаем пользователю
            break;
        }
        case 4://кейс записи структуры типа Human в текстовый файл в текстовом режиме в нотации C++
        {
            if (f == false)//если экземпляр h структуры Human не содержит данных в своих полях
            {
                cout << "Нет данных для записи.\n";//то записывать в файл нечего и кейс завершаем сразу
                break;
            }
            ofstream wtf(textfileName, ios::out | ios::app);//создаем поток wtf и связываем его с файлом по имени, хранимом в символьном массиве textfileName, открываемом в режиме
            if (!wtf)//записи ios::out и дозаписи ios::app новых данных в конец файла, уже имеющего "старые" записи, которые сохраняются. Если поток не удалось создать
            {
                cout << "Error.\n";
                break;//завершаем кейс досрочно
            }
            wtf << h.fam << ' ' << h.name << ' ' << h.age << ' ' << h.height << ' ' << h.sex << endl;//записываем данные в текстовый файл в текстовом режиме записи. Данные
            //разделяем пробелами, которые играют роль разделителей полей, но в текстовом файле это просто символы и слова в строках файла
            wtf.close();//данные записали - сразу закрываем поток записи в файл, закрывая, тем самым, файл
            cout << "Данные записаны в текстовый файл: " << textfileName << ". Файл закрыт.\n";//сообщаем пользователю
            break;
        }
        case 5://кейс чтения структур типа Human из текстового файла. Программа не знает, сколько именно структур записано в файле
        {
            ifstream rtf(textfileName, ios::in);//создаем поток rtf для чтения (режим ios::in) текстовых данных дданных из файла с именем, хранимом в массиве textfileName
            if (!rtf)//если поток чтения текстового файла не удалось создать
            {
                cout << "Не удалось открыть файл " << textfileName << " для чтения.\n";//сообщаем об этом пользователю
                break;//завершаем кейс досрочно
            }
            Human h1;//создаем переменную h1 типа структуры Human, в которую будем читать данные из файла
            rtf >> h1.fam >> h1.name >> h1.age >> h1.height >> h1.sex;//читаем первую строку данных, которые разделены пробелами, на основе чего система их и различает
            while (!rtf.eof())//пока не достигли конца файла - маркера EOF
            {
                cout << h1.fam << '|' << h1.name << '|' << h1.age << '|' << h1.height << '|' << h1.sex << endl;//печатаем одержимое полей переменной h1 типа структуры Human
                rtf >> h1.fam >> h1.name >> h1.age >> h1.height >> h1.sex;//читаем из файлового потока rtf новые данные и сразу помещаем их в соответствующие поля переменной h1
            }
            rtf.close();//закрываем поток rtf и, тем самым, закрываем прочитанный файл
            cout << "Данные прочитаны из текстового файла: " << textfileName << ". Файл закрыт.\n";//сообщаем пользователю
            break;//завершаем кейс 5
        }
        default:
        {
            cout << "Вы ввели неверный номер пункта меню. Перечитайте меню:\n";
```

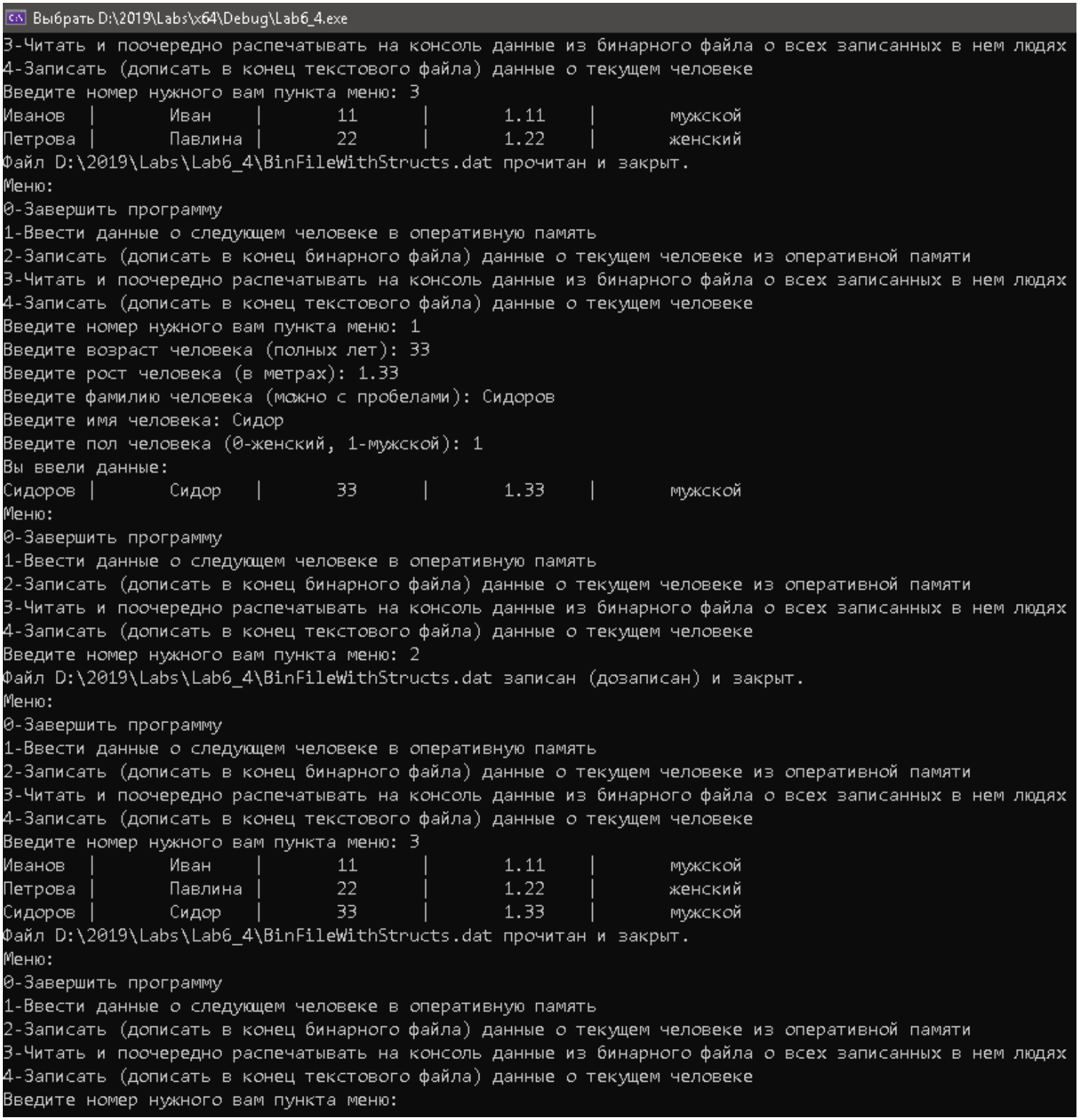
```
            break;
        }
        }
    }
    system("pause");
    return 0;
}
```

Тестирование. Поскольку программное консольное меню занимает немало места, то для демонстрации ввода 3-х человек (з-х экземпляров структуры типа Human), сохранения их в бинарный файл и чтения бинарного файла мною сделана копия данных из консоли работы программы (это своего рода дамп («копия, фотография») консоли):

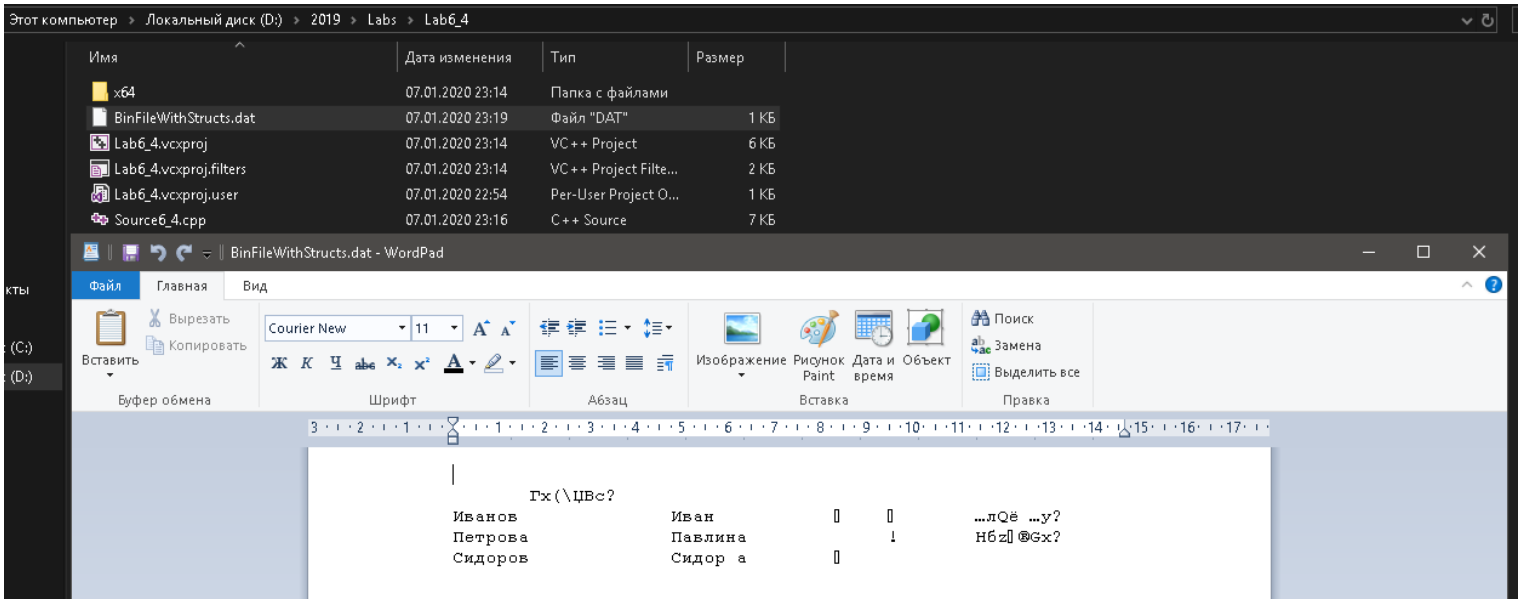
Меню:
0-Завершить программу
1-Ввести данные о следующем человеке в оперативную память
2-Записать (дописать в конец бинарного файла) данные о текущем человеке из оперативной памяти
3-Читать и поочередно распечатывать на консоль данные из бинарного файла о всех записанных в нем людях
4-Записать (дописать в конец текстового файла) данные о текущем человеке
Введите номер нужного вам пункта меню: 1
Введите возраст человека (полных лет): 11
Введите рост человека (в метрах): 1.11
Введите фамилию человека (можно с пробелами): Иванов
Введите имя человека: Иван
Введите пол человека (0-женский, 1-мужской): 1
Вы ввели данные:
Иванов | Иван | 11 | 1.11 | мужской
Меню:
0-Завершить программу
1-Ввести данные о следующем человеке в оперативную память
2-Записать (дописать в конец бинарного файла) данные о текущем человеке из оперативной памяти
3-Читать и поочередно распечатывать на консоль данные из бинарного файла о всех записанных в нем людях
4-Записать (дописать в конец текстового файла) данные о текущем человеке
Введите номер нужного вам пункта меню: 2
Файл D:\2019\Labs\Lab6_4\BinFileWithStructs.dat записан (дозаписан) и закрыт.
Меню:
0-Завершить программу
1-Ввести данные о следующем человеке в оперативную память
2-Записать (дописать в конец бинарного файла) данные о текущем человеке из оперативной памяти
3-Читать и поочередно распечатывать на консоль данные из бинарного файла о всех записанных в нем людях
4-Записать (дописать в конец текстового файла) данные о текущем человеке
Введите номер нужного вам пункта меню: 3
Иванов | Иван | 11 | 1.11 | мужской
Файл D:\2019\Labs\Lab6_4\BinFileWithStructs.dat прочитан и закрыт.
Меню:
0-Завершить программу
1-Ввести данные о следующем человеке в оперативную память
2-Записать (дописать в конец бинарного файла) данные о текущем человеке из оперативной памяти
3-Читать и поочередно распечатывать на консоль данные из бинарного файла о всех записанных в нем людях
4-Записать (дописать в конец текстового файла) данные о текущем человеке
Введите номер нужного вам пункта меню: 1
Введите возраст человека (полных лет): 22
Введите рост человека (в метрах): 1.22
Введите фамилию человека (можно с пробелами): Петрова
Введите имя человека: Павлина
Введите пол человека (0-женский, 1-мужской): 0
Вы ввели данные:
Петрова | Павлина | 22 | 1.22 | женский
Меню:
0-Завершить программу
1-Ввести данные о следующем человеке в оперативную память
2-Записать (дописать в конец бинарного файла) данные о текущем человеке из оперативной памяти
3-Читать и поочередно распечатывать на консоль данные из бинарного файла о всех записанных в нем людях
4-Записать (дописать в конец текстового файла) данные о текущем человеке
Введите номер нужного вам пункта меню: 2

Файл D:\2019\Labs\Lab6_4\BinFileWithStructs.dat записан (дозаписан) и закрыт.
Меню:
0-Завершить программу
1-Ввести данные о следующем человеке в оперативную память
2-Записать (дописать в конец бинарного файла) данные о текущем человеке из оперативной памяти
3-Читать и поочередно распечатывать на консоль данные из бинарного файла о всех записанных в нем людях
4-Записать (дописать в конец текстового файла) данные о текущем человеке
Введите номер нужного вам пункта меню: 3
Иванов | Иван | 11 | 1.11 | мужской
Петрова | Павлина | 22 | 1.22 | женский
Файл D:\2019\Labs\Lab6_4\BinFileWithStructs.dat прочитан и закрыт.
Меню:
0-Завершить программу
1-Ввести данные о следующем человеке в оперативную память
2-Записать (дописать в конец бинарного файла) данные о текущем человеке из оперативной памяти
3-Читать и поочередно распечатывать на консоль данные из бинарного файла о всех записанных в нем людях
4-Записать (дописать в конец текстового файла) данные о текущем человеке
Введите номер нужного вам пункта меню: 1
Введите возраст человека (полных лет): 33
Введите рост человека (в метрах): 1.33
Введите фамилию человека (можно с пробелами): Сидоров
Введите имя человека: Сидор
Введите пол человека (0-женский, 1-мужской): 1
Вы ввели данные:
Сидоров | Сидор | 33 | 1.33 | мужской
Меню:
0-Завершить программу
1-Ввести данные о следующем человеке в оперативную память
2-Записать (дописать в конец бинарного файла) данные о текущем человеке из оперативной памяти
3-Читать и поочередно распечатывать на консоль данные из бинарного файла о всех записанных в нем людях
4-Записать (дописать в конец текстового файла) данные о текущем человеке
Введите номер нужного вам пункта меню: 2
Файл D:\2019\Labs\Lab6_4\BinFileWithStructs.dat записан (дозаписан) и закрыт.
Меню:
0-Завершить программу
1-Ввести данные о следующем человеке в оперативную память
2-Записать (дописать в конец бинарного файла) данные о текущем человеке из оперативной памяти
3-Читать и поочередно распечатывать на консоль данные из бинарного файла о всех записанных в нем людях
4-Записать (дописать в конец текстового файла) данные о текущем человеке
Введите номер нужного вам пункта меню: 3
Иванов | Иван | 11 | 1.11 | мужской
Петрова | Павлина | 22 | 1.22 | женский
Сидоров | Сидор | 33 | 1.33 | мужской
Файл D:\2019\Labs\Lab6_4\BinFileWithStructs.dat прочитан и закрыт.
Меню:
0-Завершить программу
1-Ввести данные о следующем человеке в оперативную память
2-Записать (дописать в конец бинарного файла) данные о текущем человеке из оперативной памяти
3-Читать и поочередно распечатывать на консоль данные из бинарного файла о всех записанных в нем людях
4-Записать (дописать в конец текстового файла) данные о текущем человеке
Введите номер нужного вам пункта меню:

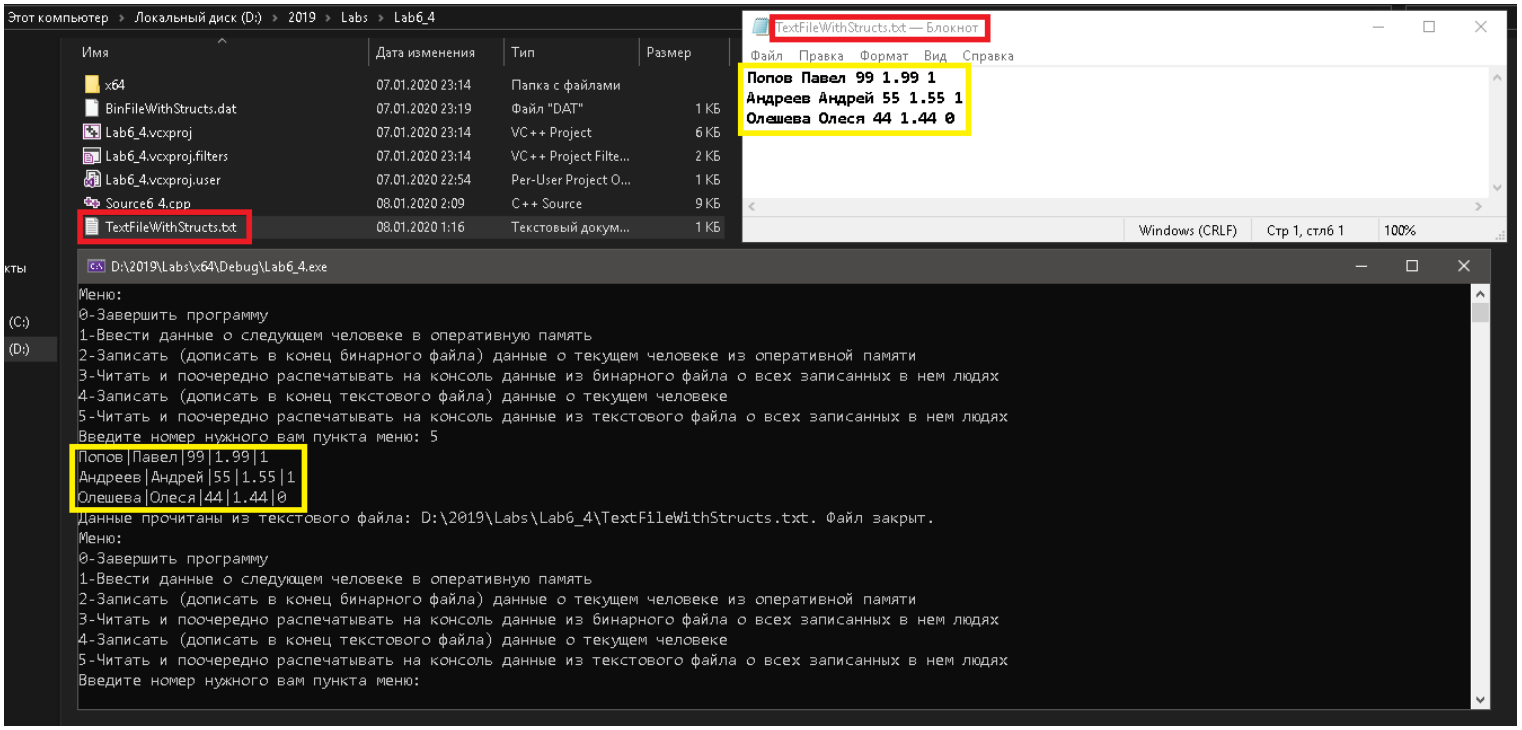
Ниже расположен скриншот последнего участка консоли, на котором внизу видно, что программа корректно читает из бинарного файла все 3 записанные (дозаписанные) в него ранее экземпляры структуры типа Human:



А вот что можно видеть, если открыть сам бинарный файл в текстовом редакторе, например, MS WordPad. Символы в словах совпали с читаемым программой форматом символов, а числа не «попали» в формат и отображаются символами, коды которых совпали с находящимися в этих местах файла нулями и единицами. Можно примерно увидеть, где начинается и заканчивается каждый экземпляр структуры типа Человек из трех имеющихся. Это нормально, так как бинарные файлы разработаны для быстрой и компактной записи данных из оперативной памяти в файл на диске. Разработчик знает формат своих структур, а потому пишет программы, которые должны корректно читать бинарные файлы с данными его программы. Скриншот выше показывает, что наш кейс чтения структур из бинарного файла работает корректно: все 3 структуры распечатались в виде таблицы с теми данными, которые и были внесены пользователем. Разумеется, если пользователь введет с клавиатуры данные, не подходящие по формату к запрашиваемым полям (даже одному полю), то это с высокой вероятностью приведет к ошибке работы программы в кейсе записи и кейсе чтения бинарного файла.



Также реализуем в кейсах 4 и 5 запись и чтение полей структур типа Human в (из) текстовый файл. Протестируем эти кейсы:

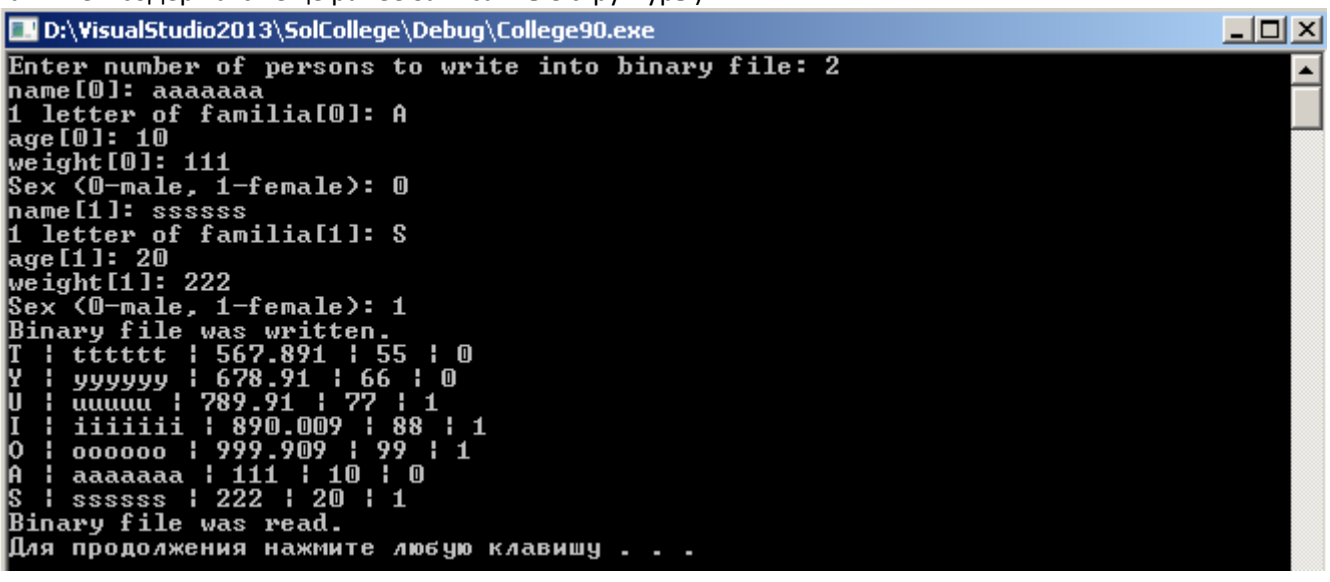


Еще пример записи и чтения экземпляров структур в (из) бинарного файла в бинарном режиме:

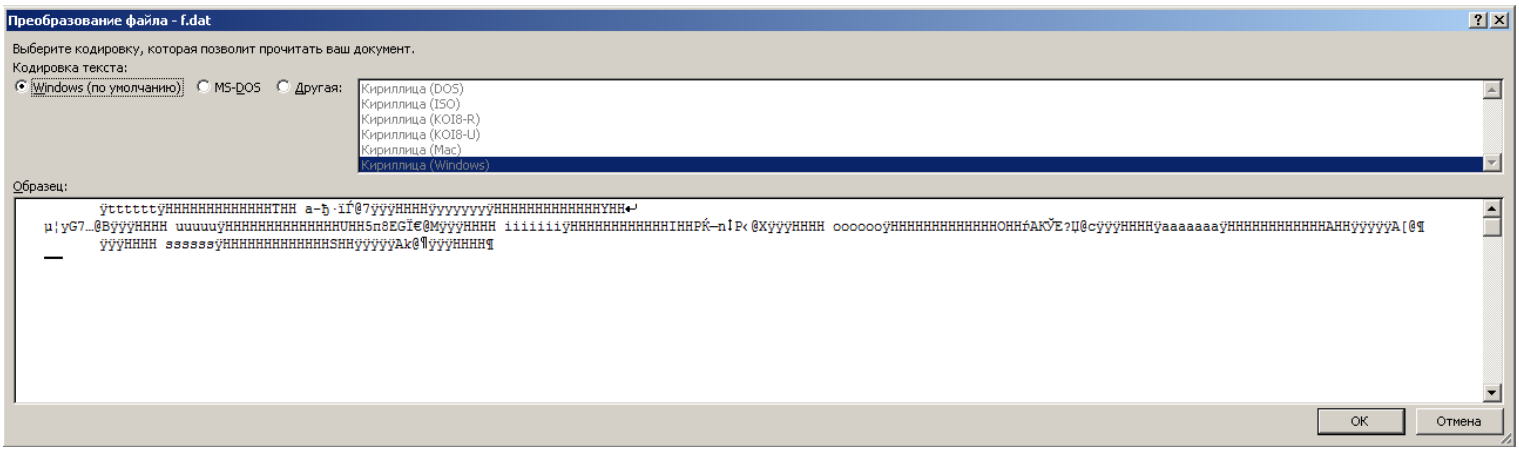
```cpp
#include <iostream>
#include <fstream>
using namespace std;

int main()
{
    struct person
    {
        bool sex;//0-male, 1-female
        char name[20];
        char fam;//1 letter//поле для хранения только одной буквы
        double weight;
        int age;
    };
    person* mas;
    char* s = "d:\\f.dat";//путь к файлу
    fstream f (s, ios::app | ios::binary);//создать поток для ДОзаписи бинарного файла (если файл с таким именем уже есть - он будет дописан в конец, иначе - создастся новый файл)
    if (!f)
    {
        cout << "Error.\n";
        system("pause");
        return 0;
    }
    int n;
    cout << "Enter number of persons to write into binary file: ";
    cin >> n;
    mas = new person[n];//создать массив из n людей (person)
    for (int i = 0; i < n; i++)
    {
        cout << "name[" << i << "]: ";
        cin >> mas[i].name;
        cout << "1 letter of familia[" << i << "]: ";
        cin >> mas[i].fam;
        cout << "age[" << i << "]: ";
        cin >> mas[i].age;
        cout << "weight[" << i << "]: ";
        cin >> mas[i].weight;
        cout << "Sex (0-male, 1-female): ";
        cin >> mas[i].sex;
        f.write((char*)&mas[i], sizeof(mas[i]));//бинарная запись целой структуры person в бинарный файл на C++.
        //((char*) - явное приведение типа из типа данных person к массиву символов (char*) писать для любого элемента, записываемого в бинарный файл.
        //потоковый ввод f << mas[i]; не позволит бинарно записать структуру в файл, а код f << mas[i].name << ' ' << mas[i].sex << '\n'; означает запись текстового файла
    }
    f.close();
    cout << "Binary file was written.\n";
    person p;//создать временную структуру для заполнения ее полей данными из бинарного файла и последующей печати данных из этих полей на консоль
    f.open(s, ios::in | ios::binary);//чтение из бинарного файла
    if (!f)
    {
        cout << "Error1.\n";
        system("pause");
        return 0;
    }
    do
    {
        f.read((char*)&p, sizeof(p));//бинарное чтение целой структуры из бинарного файла на C++. Явное приведение типа (char*) обязательно для любого считываемого элемента
        //потоковое чтение f >> mas[i]; не позволит бинарно считать структуру из файла, а код f >> p.name >> p.sex; означает чтение текстового файла
        if (f.eof())
        {
            break;
        }
        cout << p.fam << " | " << p.name << " | " << p.weight << " | " << p.age << " | " << p.sex << endl;
    }
    while (!f.eof());
    f.close();
    cout << "Binary file was read.\n";
    system("pause");
    return 0;
}
```

Результат работы программы (добавлено 2 структуры, но из бинарного файла распечатано больше структур, так как в нем содержатся еще ранее записанные структуры):



Если попробовать прочитать содержимое бинарного файла в текстовом редакторе, то он читается некорректно, хотя можно найти английские символы, но числа при этом стали неузнаваемыми символами, так как их формат не поддерживается данным текстовым редактором и требуется программа-«читатель», которая будет знать формат записанных данных в бинарный файл (у нас это структура struct person с ее полями) и сможет их прочитать корректно (что можно наблюдать в окне консоли выше).



Пример решения задачи: записать в файл рандомные целые числа, прочитать их из файла и распечатать на консоль. Далее прочитать из файла все числа и записать только четные из них в другой файл.

```cpp
#include <iostream>
#include <fstream>
#include <time.h>
using namespace std;

int main()
{
    ofstream f("d:\\A1.txt", ios::out | ios::trunc);//write file from memory
    if (!f)
    {
        cout << "Error.\n";
        system("pause");
        return 0;
    }
    cout << "Number of random numbers: ";
    int n;
    cin >> n;
    srand(time(NULL));
    for (int i = 0; i < n; i++)
    {
        f << (rand() % 100) << ' ';
    }
    f.close();
    cout << "file with random numbers ready.\nThis numbers are:\n";
    ifstream f0("d:\\A1.txt", ios::in);//read file into memory
    if (!f0)
    {
        cout << "Error0.\n";
        system("pause");
        return 0;
    }
    int k;
    f0 >> k;
    while (!f0.eof())
    {
        cout << k << ' ';
        f0 >> k;
```

```
    }
    f0.close();
    cout << "\nFile was read.\n";
    ofstream f1("d:\\A2.txt", ios::out | ios::trunc);
    if (!f1)
    {
        cout << "Error1.\n";
        system("pause");
        return 0;
    }
    ifstream f2("d:\\A1.txt", ios::in);
    if (!f2)
    {
        cout << "Error2.\n";
        system("pause");
        return 0;
    }
    f2 >> k;
    while (f2)
    {
        if (k % 2 == 0)
        {
            f1 << k << ' ';
        }
        f2 >> k;
    }
    _fcloseall();//закрыть все файлы
    cout << "\nResult file ready.\n";
    ifstream f3("d:\\A2.txt", ios::in);
    if (!f3)
    {
        cout << "Error3.\n";
        system("pause");
        return 0;
    }
    f3 >> k;
    while (f3)
    {
        cout << k << ' ';
        f3 >> k;
    }
    f3.close();
    cout << "\nResult file was read.\n";
    system("pause");
    return 0;
}
```

Выполнить все задания от № 1 до № 13:

В названиях файлов вместо %Фамилия% записывать свою фамилию.

1	Написать программу, взаимодействующую с пользователем посредством консольного меню (switch-case внутри бесконечного цикла, прерывающегося при вводе некоторого значения пользователем; каждый case реализует один пункт меню). Кейсы могут содержать подменю.
2	Запросить у пользователя ввести один абзац текста (до нажатия пользователем Enter'а). Сохранить данный текст в текстовый файл с именем «ФамилияЛаб20.txt» посредством посимвольной записи (в каждой итерации цикла записывается один символ типа char).
3	Посимвольно считать из вашего текстового файла с именем «ФамилияЛаб20.txt» текст и вывести его на консоль.
4	Запросить у пользователя размер массива и вводимые значения элементов массива. Сохранить эти элементы

	массива в текстовый файл с именем «ФамилияЛаб20_0.txt» посредством потокового ввода (направляющие скобки <<). Записываемые значения разделяйте символами по варианту. Тип массива и разделяющий символ определяются вашим номером в журнале группы: 1 – float, разделитель '_' 2 – int, разделитель '\' 3 – long int, разделитель '!' 4 – short int, разделитель '?' 5 – char, разделитель ' ' (пробел) 6 – double, разделитель '#' 7 – float, разделитель '$' 8 – int, разделитель '^' 9 – long int, разделитель '%' 10 – short int, разделитель '*' 11 – char, разделитель '{' 12 – double, разделитель '}' 13 – float, разделитель ';' 14 – int, разделитель ':' 15 – long int, разделитель '~' 16 – short int, разделитель '+' 17 – char, разделитель '-' 18 – double, разделитель '=' 19 – float, разделитель ')' 20 – int, разделитель ''' (апостроф) 21 – long int, разделитель ',' 22 – short int, разделитель '|' 23 – char, разделитель '~' (тильда) 24 – double, разделитель '<' 25 – float, разделитель '>' 26 – int, разделитель '.' 27 – long int, разделитель ',' 28 – short int, разделитель ')' 29 – char, разделитель '/' 30 – double, разделитель '\'
5	Считать из текстового файла с именем «ФамилияЛаб20_0.txt» элементы массива посредством потокового вывода (направляющие скобки >>) и вывести на консоль считанный из файла массив (его элементы), разделяя печатаемые значения символом «пробел» ' '.
6	Объявить в вашей программе структуру struct по варианту лабораторной работы № 15. Запросить у пользователя ввести содержимое полей объекта типа вашей структуры. Сохранить содержимое всех полей структуры в текстовый файл с именем «ФамилияЛаб20_1.txt». Используйте режим ДОзаписи файла (записи в конец файла), чтобы в файл записывались (сохранялись, накапливались) все вводимые пользователем структуры, когда он будет выбирать этот кейс в меню.
7	Прочитать из текстового файла с именем «ФамилияЛаб20_1.txt» все записанные в нем структуры и распечатать содержимое их полей на консоль, разделяя содержимое полей знаком '|'. Причем программа не знает количество записанных в текстовый файл структур, поэтому читает их из файла до тех пор, пока есть что читать из файла.
8	Запросить у пользователя ввести содержимое полей объекта типа вашей структуры. Сохранить содержимое всех полей структуры в бинарный файл с именем «ФамилияЛаб20_2.bin» посредством функции write((char*)&записываемыйЭлемент, размерЗаписываемогоЭлементаВБайтах). Используйте режим ДОзаписи файла (записи в конец файла), чтобы в файл записывались (сохранялись, накапливались) все вводимые пользователем структуры, когда он будет выбирать этот кейс в меню.
9	Прочитать из бинарного файла с именем «ФамилияЛаб20_2.bin» все записанные в нем структуры с помощью функции read((char*)&считываемыйЭлемент, размерСчитываемогоЭлементаВБайтах) и распечатать содержимое их полей на консоль, разделяя содержимое полей символами " ; ". Причем программа не знает количество записанных в бинарный файл структур, поэтому читает их из файла до тех пор, пока есть что читать из файла.
10	Запросить у пользователя ввести предложение (слова с пробелами, завершается нажатием Enter'а) и **до**писать данное предложение в текстовый файл с именем «ФамилияЛаб20_3.txt» в виде отдельной строки.
11	Построчно считать из текстового файла с именем «ФамилияЛаб20_3.txt» все предложения и вывести их на консоль.
12	Закрывайте файл (закрывайте поток работы с файлом) в конце **каждого** кейса.
13	Предусмотрите пункт меню для завершения работы вашей программы.

Выполнить индивидуальное задание по своему варианту:

1	Прочитать из текстового файла три предложения и записать их в новый файл в обратном порядке. Концом предложения считать символы '.', '!', '?'.
2	Прочитать из текстового файла и записать в новый файл только строки, содержащие двузначные числа.
3	Прочитать текст из файла и записать в новый файл все цитаты (т.е. текст, заключенный в кавычки).
4	Прочитать текст из файла и записать в новый файл слова, начинающиеся с английских гласных букв.
5	Прочитать текст из файла и записать его в новый файл, заменив первую букву в каждом слове на заглавную (Большую).
6	Прочитать текст из файла и записать его в новый файл, заменив все слова, длиннее 5 букв на соответствующее количество звездочек '*'.
7	Прочитать текст из файла и записать в новый файл сначала вопросительные предложения, а затем все остальные.
8	Прочитать текст из файла и записать в новый файл сначала предложения, начинающиеся с тире, а затем все остальные.
9	Прочитать английский текст из файла и записать в новый файл тот же текст, заменив все буквы на заглавные (Большие).
10	Прочитать английский текст из файла и записать в новый файл тот же текст, заменив все буквы на прописные (маленькие).
11	В файле содержится текстовая строка. Определить частоту повторяемости каждой буквы в тексте и вывести ее. Частота повторяемости равна частному от всех букв в тексте на количество данных букв
12	В файле содержится совокупность текстовых строк. Изменить первую букву каждого слова на заглавную.
13	Дан текстовый файл. Создать новый файл, каждая строка которого получается из соответствующей строки исходного файла перестановкой слов в обратном порядке.
14	В данном текстовом файле удалить все слова, которые содержат хотя бы одну цифру.
15	Дан текстовый файл. Из текстового файла удалить все слова, содержащие от трех до пяти символов, но при этом из каждой строки должно быть удалено только четное количество таких слов.
16	Текстовый файл содержит квадратную матрицу, которая записана по принципу: одна строка – один элемент матрицы. Необходимо определить размерность матрицы и построить двухмерный массив. Вывести на экран исходную матрицу и результат ее поворота на 90º по часовой стрелке.
17	Текстовый файл содержит квадратную матрицу, которая записана по принципу: одна строка файла – одна строка матрицы. Необходимо построить двухмерный массив и вывести на экран исходную матрицу и результат ее транспонирования (строки становятся столбцами – матрица «переворачивается» на 90 градусов).
18	Текстовый файл содержит вектор (одномерный массив), который записан по принципу: одна строка файла – один элемент вектора. Необходимо посчитать длину вектора и сохранить его в отдельный файл.
19	Пусть файлы A и B содержат целые числа, упорядоченные по неубыванию. Получить в файле C все числа из файлов A и B без повторений. Файл C должен быть упорядочен по возрастанию.
20	Пусть файл содержит целые числа, которые записаны по 10 в каждой строке. Необходимо изменить порядок чисел в каждой десятке так, чтобы вначале шли числа, делящиеся на 3 без остатка, затем числа, дающие при делении на 3 остаток 1, затем числа, дающие при делении на 3 остаток 2. Порядок самих десятков должен быть сохранен.
21	В файле содержится текстовая строка. Определить частоту повторяемости каждой буквы в тексте и вывести ее. Частота повторяемости равна частному от всех букв в тексте на количество данных букв
22	В файле содержится совокупность текстовых строк. Изменить первую букву каждого слова на заглавную.
23	Дан текстовый файл. Создать новый файл, каждая строка которого получается из соответствующей строки исходного файла перестановкой слов в обратном порядке.
24	В данном текстовом файле удалить все слова, которые содержат хотя бы одну цифру.
25	Дан текстовый файл. Из текстового файла удалить все слова, содержащие от трех до пяти символов, но при этом из каждой строки должно быть удалено только четное количество таких слов.
26	Текстовый файл содержит квадратную матрицу, которая записана по принципу: одна строка – один элемент матрицы. Необходимо определить размерность матрицы и построить двухмерный массив. Вывести на экран исходную матрицу и результат ее поворота на 90º против часовой стрелки.
27	Текстовый файл содержит квадратную матрицу, которая записана по принципу: одна строка файла – одна строка матрицы. Необходимо построить двухмерный массив и вывести на экран исходную матрицу и результат ее транспонирования (строки становятся столбцами – матрица «переворачивается» на 90 градусов).
28	Текстовый файл содержит вектор (одномерный массив), который записан по принципу: одна строка файла – один элемент вектора. Необходимо посчитать длину вектора и сохранить его в отдельный файл.
29	Пусть файлы A и B содержат целые числа, упорядоченные по неубыванию. Получить в файле C все числа из файлов A и B без повторений. Файл C должен быть упорядочен по возрастанию.
30	Пусть файл содержит целые числа, которые записаны по 10 в каждой строке. Необходимо изменить порядок чисел в каждой десятке так, чтобы вначале шли числа, делящиеся на 3 без остатка, затем числа, дающие при

	делении на 3 остаток 1, затем числа, дающие при делении на 3 остаток 2. Порядок самих десятков должен быть сохранен.