Лабораторная работа № 19

«**Разработка алгоритмов и программ с использованием текстовых файлов. Сравнение работы с текстовым и бинарным файлами**»

Для каждого учащегося предназначено **11 заданий,** которые нужно реализовать **в одной программе** с меню и кейсами. Сделать **1** программу с заданным ниже функционалом.

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.

При использовании «небезопасных» функций начинайте программу с директивы препроцессору:

#define _CRT_SECURE_NO_WARNINGS

Режимы работы с файлами в POSIX-нотации:

wt – режим записи в текстовый.txt-файл. Если файла с таким именем по такому пути нет, то он создастся, а если есть, то «старый» файл открывается, все его содержимое удаляется и записывается новое содержимое (файл перезаписывается). Курсор для ввода ставится в самое начало файла. В файл можно писать и нельзя из него читать.

rt – режим чтения из .txt-файла. Если файла по такому пути нет, то будет ошибка error, а если есть, то курсор для чтения поставится в начало файла. Из файла можно читать, но нельзя в него писать.

at – режим дозаписи информации в конец .txt-файла. Если файла по такому пути нет, то он создастся, а если есть, то он будет открыт для записи, курсор поставится в самый конец файла, и с этого места можно осуществлять дозапись новой информации в файл, «старая» информация при этом в файле остается.

wt+, rt+, at+ - комбинированные режимы записи и чтения из файла, то есть обеспечивают возможность открыть файл и осуществлять в него запись и чтение из него информации. Имеют свою специфику и для безопасности и порядка работы с файлом рекомендуется открывать файл для одной цели (записи или чтения), потом закрывать и далее открывать для другой цели и так далее.

t – режим открытия файла в текстовом формате. Подходит для .txt, .doc, .xml, .html, .json-файлов.

b – режим открытия файла в бинарном (двоичном) режиме. Подходит для всех файлов, кроме текстовых, то есть .bin, .dat и других расширений, созданных программистом, например .bmp, .myfile и т.д.

Режимы для бинарных файлов именуются аналогично режимам для текстовых файлов:

wb

rb,

ab,

wb+, rb+, ab+.

Для чего предназначены режимы, перечисленные выше? Опишите их назначение.

Если указан режим, а тип файла не указан (t или b), то по умолчанию предполагается компилятором текстовый файл, то есть t. Например, запись “w” компилятором будет интерпретирована как “wt”. Для бинарных файлов надо писать полный формат. Рекомендуется во всех случаях писать режим работы с файлом полностью с указанием типа файла.

При записи и чтении в файл в нотации POSIX надо знать спецификаторы преобразования для разных типов данных. Эти спецификаторы преобразования типов данных можно будет записывать в функциях, чтобы указывать, значение какого формата нужно записать за каким другим значением и так далее. А при чтении нужно указать эти же спецификаторы в том же порядке, чтобы правильно считать последовательность данных из файла. Перед спецификаторами формата ставится знак %, например “%d %lf %c %s” (тут задан формат «одноЦелоеЧисло одноВещественноеЧисло одинСимвол однаСтрокаСимволовБезПробелов». Обратите внимание, что между ними находится по одному пробелу – и в файле между этими данными будет по одному пробелу. Можно добавлять и сами символы с строку форматирования данных, например "%d: %lf_%c\n” означает:

«одноЦелоеЧисло: одноВещественноеЧисло_одинСимвол*СделатьПереносКурсораНаСледующуюСтроку*». Сами знаки процентов % не будут записаны в файл, но если в файл надо записать знак процента % как символ, то необходимо написать два знака %% подряд (один из них станет экранирующим для второго и в файл запишется один знак % как простой символ (проверьте способ с \%). Аналогично при необходимости записи в файл символа \”, \\ и так далее.

Помните, что значения переменных, которые мы пишем в файл должны по порядку следования и количеству полностью совпадать с последовательностью спецификаторов формата.

d	Одно целое число типа int, signed int
i	Одно целое число типа int, signed int
li	Одно целое число типа long int, signed long int
u	Одно беззнаковое число типа unsigned int
x	Одно число в шестнадцатеричном формате без знака (hex). Все буквы будут маленькие (нижний регистр)
X	Одно число в шестнадцатеричном формате без знака (hex). Все буквы будут большие (верхний регистр)
e	Одно число в экспоненциальной форме записи (научной нотации). Сам символ е будет в нижнем регистре, то есть 1.2345e-16
E	Одно число в экспоненциальной форме записи (научной нотации). Сам символ е будет в верхнем регистре, то есть 1.2345E+34
b	Одно число в бинарном виде (двоичной системе счисления – binary) или bool

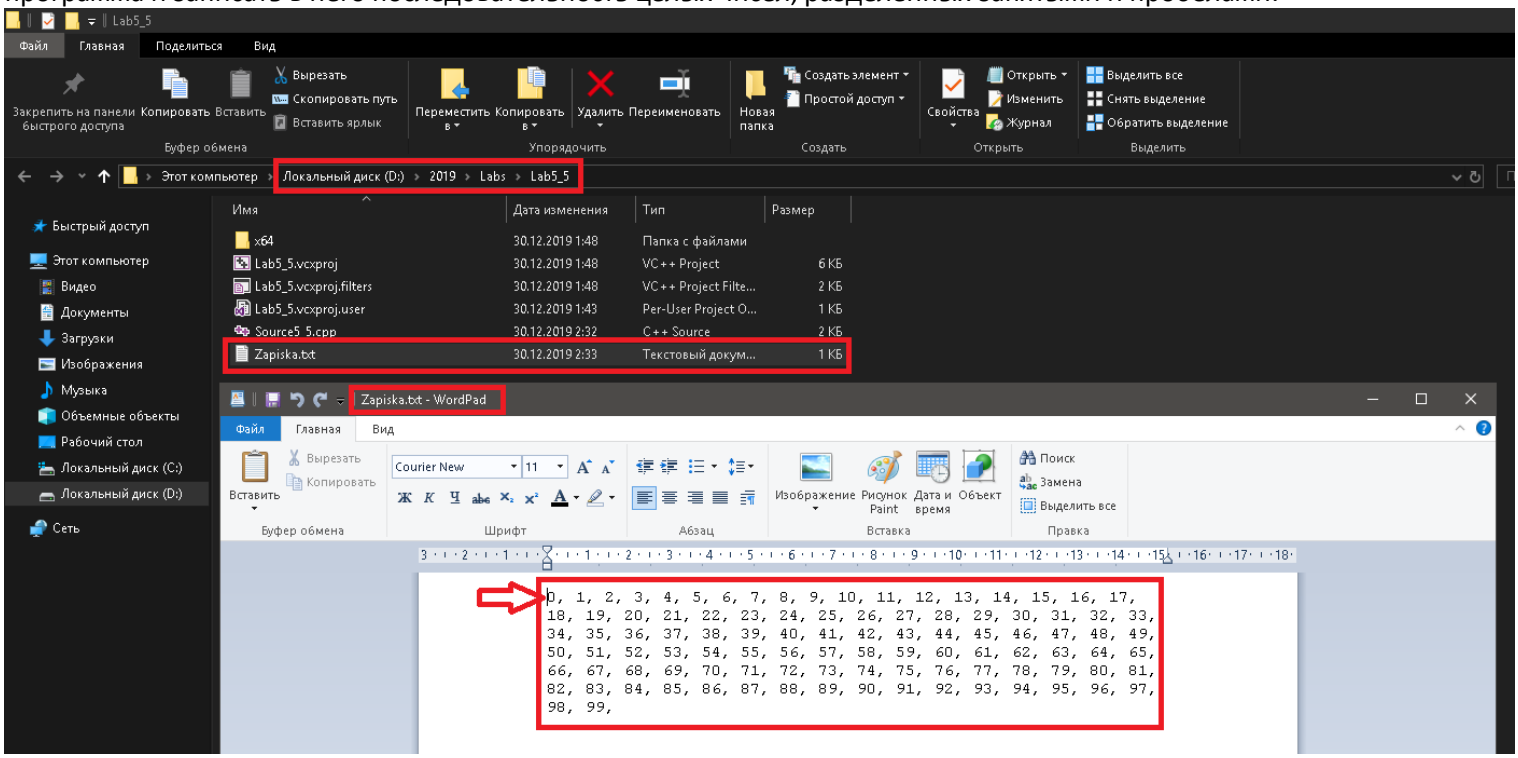
o	Одно число в восьмеричном виде (восьмеричной системе счисления – octa)
p	Один указатель (хранимое в нем значение, то есть адрес в шестнадцатеричном формате)
lf	Одно число типа double, long float
c	Один символ типа char
s	Строка символов (от англ. string – строка). Строка считывается с первого символа, не являющегося пробельным, и включает все символы до следующего пробельного символа

```
#define _CRT_SECURE_NO_WARNINGS//надо для использования функций fopen() и некоторых других
#include <iostream>
#include <Windows.h>
using namespace std;

int main()
{
    SetConsoleOutputCP(1251);
    SetConsoleCP(1251);
    FILE* f = fopen("D:\\2019\\Labs\\Lab5_5\\Zapiska.txt", "wt");//создать поток для записи в текстовый файл, находящийся по указанному пути.
    //Открыть файл в режиме записи текстового файла (wt)
    if (f == NULL)//аналогично if(!f)//если не удалось открыть файл
    {
        cout << "Ошибка открытия файла для записи.\n";//то сообщаем об этом пользователю
        system("pause");
        return 0;//и завершаем программу
    }
    for (int i = 0; i < 100; i++)//иначе в цикле записываем в файл числа от 0 до 99 включительно
    {//fprintf_s() - функция печати (записи) в файл данных (безопасная функция _secured). fprintf_s() принимает три параметра: в поток работы с каким открытым файлом вести запись
        fprintf_s(f, "%d, ", i);// - в поток f; формат записи (как записывать) "%d, "; что значит "целоеЧисло, "; откуда брать данное для записи - из переменной i (она имеет тип
    }// int - целочисленный, а потому подходит по формату для %d (десятичное целое знаковое число)
    cout << "Запись файла завершена.\n";//сообщайте пользователю о завершении записи и чтения файла - это долгие процессы
    fclose(f);//всегда закрывайте файл (и поток работы с ним), когда он вам не нужен, поскольку файл - ресурс монопольного доступа - пока с ним работает ваша программа, с ним не
    system("pause");// могут работать другие программы (процессы)
    return 0;
}
```

```
D:\2019\Labs\x64\Debug\Lab5_5.exe
Запись файла завершена.
Для продолжения нажмите любую клавишу . . .
```

Если программа проработала с нужным результатом, то найдем и откроем файл, который должна была создать наша программа и записать в него последовательность целых чисел, разделенных запятыми и пробелами:



Также введем в код программы несуществующий путь (папки с названием ККК нет на диске D) и протестируем работу программы в этом случае:

```
#define _CRT_SECURE_NO_WARNINGS//надо для использования функций fopen() и некоторых других
#include <iostream>
#include <Windows.h>
using namespace std;

int main()
{
    SetConsoleOutputCP(1251);
    SetConsoleCP(1251);
    FILE* f = fopen("D:\\KKK\\tf.txt", "wt");//подпапки KKK не существует, то есть путь ошибочен
    //Открыть файл в режиме записи текстового файла (wt)
    if (f == NULL)//аналогично if(!f)//если не удалось открыть файл
    {
        cout << "Ошибка открытия файла для записи.\n";//то сообщаем об этом пользователю
        system("pause");
        return 0;//и завершаем программу
    }
    for (
    {//fp
        f
    }// i
    cout
    fclos
    syste
    return 0;
}
```

```
D:\2019\Labs\x64\Debug\Lab5_5.exe
Ошибка открытия файла для записи.
Для продолжения нажмите любую клавишу . . .
```

Теперь напишем другую программу для чтения информации из файла.

```
#define _CRT_SECURE_NO_WARNINGS//надо для использования функций fopen() и некоторых других
#include <iostream>
#include <Windows.h>
using namespace std;

int main()
{
    SetConsoleOutputCP(1251);
    SetConsoleCP(1251);
    FILE* f;//создать поток для чтения из текстового файла, находящимуся по указанному ниже пути. Открыть файл в режиме чтения из текстового файла (rt)
    if (!(f = fopen("D:\\2019\\Labs\\Lab5_5\\Zapiska.txt", "rt")))//вызовем работу функции fopen() и присвоим возвращенное ею значение (адрес) указателю f внутри проверки if()
    {//если вернуло пустой адресс NULL
        cout << "Ошибка открытия файла для чтения.\n";//то сообщаем об этом пользователю
        system("pause");
        return 0;//и завершаем программу
    }
    int x;//создадим переменную, в которую будем помещать очередное считываемое из файла целое число
    while (!feof(f))//читать из файла до тех пор, пока не достигнем конца файла.Функция feof() принимает имя f потока работы с открытым файлом и проверяет, достигнут ли конец файла
    {//find end of file (англ. искать (проверить) не достигнут ли конец файла). Функция вернет значение >0 - если проверка покажет, что достигнут конец файла, и вернет значение <=0
        //если проверка покажет, что не достигнут. Функция отпределяет конец файла по находящемуся в конце файла флагу EOF - End Of File (конец файла)
        fscanf_s(f, "%d, ", &x);//функция считывания данных из файла и помещения их по адресам переменных. fscanf_s() читает (сканирует) файл и проверяет, не находятся ли
        //считываемые данные в формате "%d, ", если да, то считанное целое число помещается по адресу переменной x. Обратите внимание, формат данных при записи fprintf_s() и
        //считывании fscanf_s() совпадает полностью, поскольку для корректного считывания данные должны располагаться в нужном формате вплоть до одного символа. Считанные данные
        //помещаются по адресу переменной. Но если считывается строка символов (массив char), то писать & не надо, поскольку имя массива есть адрес его начала, а потому брать
        //адрес у адреса будет уже ошибкой
        cout << x << '_';//сразу распечатываем число, помешенное в перменую, поскольку в следующей итерации в перменную поместится другое число, удалив "старое" значение
    }
    cout << "\nЧтение файла завершено.\n";//сообщайте пользователю о завершении записи и чтения файла - это долгие процессы
    fclose(f);//всегда закрывайте файл (и поток работы с ним), когда он вам не нужен, поскольку файл - ресурс монопольного доступа - пока с ним работает ваша программа, с ним не
    system("pause");// могут работать другие программы (процессы)
    return 0;
}
```

```
D:\2019\Labs\x64\Debug\Lab5_6.exe
0_1_2_3_4_5_6_7_8_9_10_11_12_13_14_15_16_17_18_19_20_21_22_23_24_25_26_27_28_29_30_31_32_33_34_35_36_37_38_39_40_41_42_43_44_45_46_47_48_49_50
_51_52_53_54_55_56_57_58_59_60_61_62_63_64_65_66_67_68_69_70_71_72_73_74_75_76_77_78_79_80_81_82_83_84_85_86_87_88_89_90_91_92_93_94_95_96_97_
98_99_
Чтение файла завершено.
Для продолжения нажмите любую клавишу . . .
```

Проверим, что будет, если искомого файла программа не найдет:

```cpp
#define _CRT_SECURE_NO_WARNINGS//надо для использования функций fopen() и некоторых других
#include <iostream>
#include <Windows.h>
using namespace std;

int main()
{
    SetConsoleOutputCP(1251);
    SetConsoleCP(1251);
    FILE* f;//создать поток для чтения из текстового файла, находящимуся по указанному ниже пути. Открыть файл в режиме чтения из текстового файла (rt)
    if (!(f = fopen("D:\\2019\\Hj.txt", "rt")));//изменим путь к файлу на несуществующий
    {//если вернуло пустой адрес NULL
        cout << "Ошибка открытия файла для чтения.\n";//то сообщаем об этом пользователю
        system("pause");
        return 0;//и завершаем программу
    }
    int x;//создадим переменную, в которую будем помещать очередное считываемое из файла целое число
    while (!feof(f))//читать из файла до тех пор, пока не достигнем конца файла.Функция feof() принимает имя f потока работы с открытым файлом и проверяет, достигнут ли конец файла
    {//find end of file (англ. искать (проверить) не достигнут ли конец файла). Функция вернет значение >0 - если проверка покажет, что достигнут конец файла, и вернет значение <=0
        //если проверка покажет, что не достигнут. Функция отпределяет конец файла по находящемуся в конце файла флагу EOF - End Of File (конец файла)
        fscanf_s(f, "%d, ", &x);//функция считывания данных из файла и помещения их по адресам переменных. fscanf_s() читает (сканирует) файл и проверяет, не находятся ли
        //считываемые данные в формате "%d, ", если да, то считанное целое число помещается по адресу переменной x. Обратите внимание, формат данных при записи fprintf_s() и
        //считывании fscanf_s() совпадает полностью, поскольку для корректного считывания данные должны располагаться в нужном формате вплоть до одного символа. Считанные данные
        //помещаются по адресу переменной. Но если считывается строка символов (массив char), то писать & не надо, поскольку имя массива есть адрес его начала, а потому брать
        //адрес у адреса будет уже ошибкой
        cout << x << '_';//сразу распечатываем число, помешенное в перменую, поскольку в следующей итерации в перменную поместится другое число, удалив "старое" значение
    }
    cout << "\nЧтение файла завершено.\n";//сообщайте пользователю о завершении записи и чтения файла - это долгие процессы
    fclose(f);//всегда закрывайте файл (и поток работы с ним), когда он вам не нужен, поскольку файл - ресурс монопольного доступа - пока с ним работает ваша программа, с ним не
    system("pause");// могут работать другие программы (процессы)
    return 0;
}
```

```
D:\2019\Labs\x64\Debug\Lab5_6.exe
Ошибка открытия файла для чтения.
Для продолжения нажмите любую клавишу . . .
```

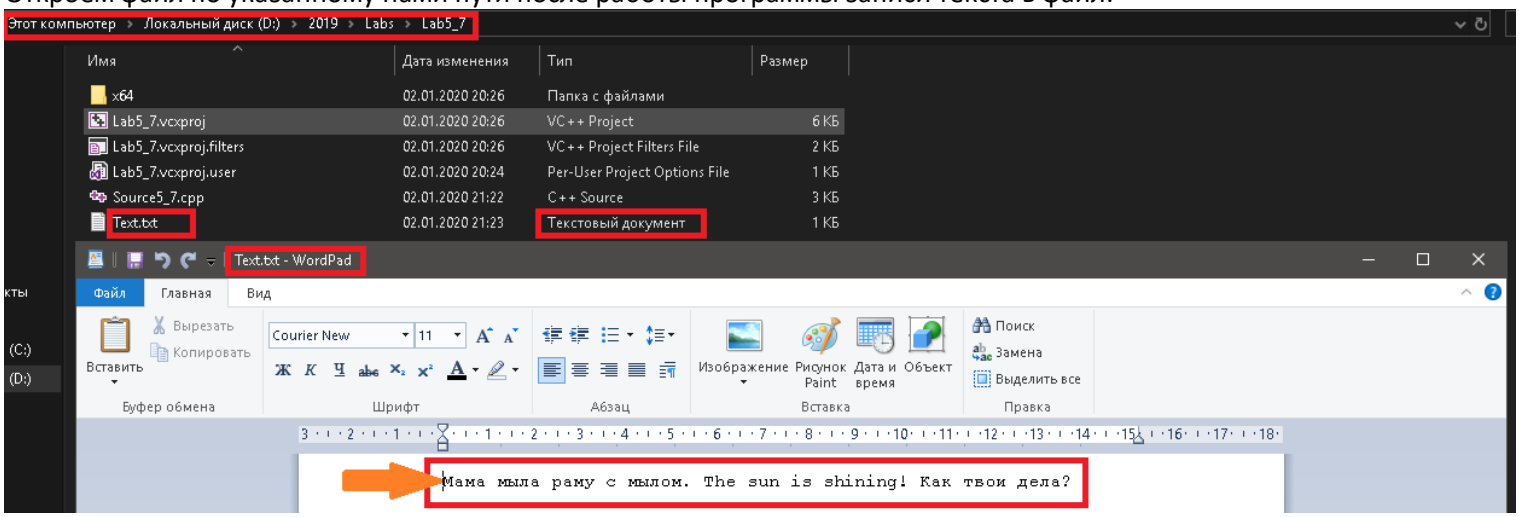
Посимвольная запись текста в текстовый файл

```cpp
#define _CRT_SECURE_NO_WARNINGS//надо для использования функций fopen() и некоторых других
#include <iostream>//вам может понадобится подключить библиотеки <stdio.h><conio.h>
#include <Windows.h>//для обеспечения поддержки русского языка, в том числе при записи и чтении файлов (по крайней мере в ОС Windows)
using namespace std;

int main()
{
    SetConsoleOutputCP(1251);
    SetConsoleCP(1251);
    FILE* wf;//создадим (продекларируем) поток для работы с файлом - это указатель на переменную типа структуры struct FILE
    char s[100];//создадим массив для хранения текста, вводимого пользователем с клавиатуры
    cout << "Введите абзац текста для записи в файл:\n";
    cin.getline(s, 100);//пользователь вводит текст с пробелами, русскими буквами до 99 символов (включая пробелы) и ноль-терминатор '\0'
    cout << "Печать введенного текста до записи в файл:\n";
    cout << s << endl;//распечатаем из массива в оперативной памяти введенный пользователем текст, чтобы удостовериться, что там есть
    if (!(wf = fopen("D:\\2019\\Labs\\Lab5_7\\Text.txt", "wt")))//проинициализируем наш указатель на поток работы с файлом wf с помощью функции fopen(), которая принимает первым
    {//параметром путь к файлу и вторым параметром - режим открытия файла: "wt" - это "write text file" (записывать в текстовый файл). Если функция fopen() не сможет создать или
        cout << "Ошибка открытия (создания) файла для записи.\n";//открыть файл для записи, то она вернет пустой адрес (NULL), который присвоится указателю wf и мы в проверке if()
        system("pause");//это сразу проверям и завершаем программу при неустраивающем нас результате проверки
        return 0;
    }
    //если выполнение дошло до этой строки, то, значит, файл D:\\2019\\Labs\\Lab5_7\\Text.txt создан и открыт для текстовой записи
    int i = 0;//записывать в файл будем посимвольно (по одному символу типа char), поэтому нам нужен счетчик - переменная типа int
    while (s[i] != NULL)//while(s[i])//пока есть что читать из массива символов s. Проверку можно записать двумя вариантами
    {
        fputc(s[i++], wf);//функция fputc(символДляЗаписиВФайл, названиеПотокаРаботыСЗаписываемымФайлом) берет ОДИН символ типа char и записывает его в открытый для записи файл
    }//расшифровка имени функции file put character (char) - поместить символ в файл. Поскольку мы вызываем эту функцию в цикле, то она записывает файл посимвольно (по ОДНОМУ
    //символу за один вызов функции в одной итерации цикла while-do. Обратите внимание, что итерационная переменная i нарашивается ПОСТФИКСНО при вызове функции fputc(). Зачем?
    fclose(wf);//сразу, как только файл нам не нужен, закрываем поток работы с ним, закрывая тем самым файл и освобождая его для использования другими потоками и процессами
    cout << "Текст записан в файл.\n";//текст может быть большим, а текстовая запись медленнее бинарной, поэтому нужно сообщить пользователю о завершении записи файла
    system("pause");
    return 0;
}
```

```
D:\2019\Labs\x64\Debug\Lab5_7.exe
Введите абзац текста для записи в файл:
Мама мыла раму с мылом. The sun is shining! Как твои дела?
Печать введенного текста до записи в файл:
Мама мыла раму с мылом. The sun is shining! Как твои дела?
Текст записан в файл.
Для продолжения нажмите любую клавишу . . .
```

Откроем файл по указанному нами пути после работы программы записи текста в файл:



Обратите внимание, что курсор работы с текстом программа MS WordPad поставила в начале файла. Также заметим, что одновременно корректно отображаются как русские буквы (кириллица), так и английские.

Посимвольное чтение текстового файла

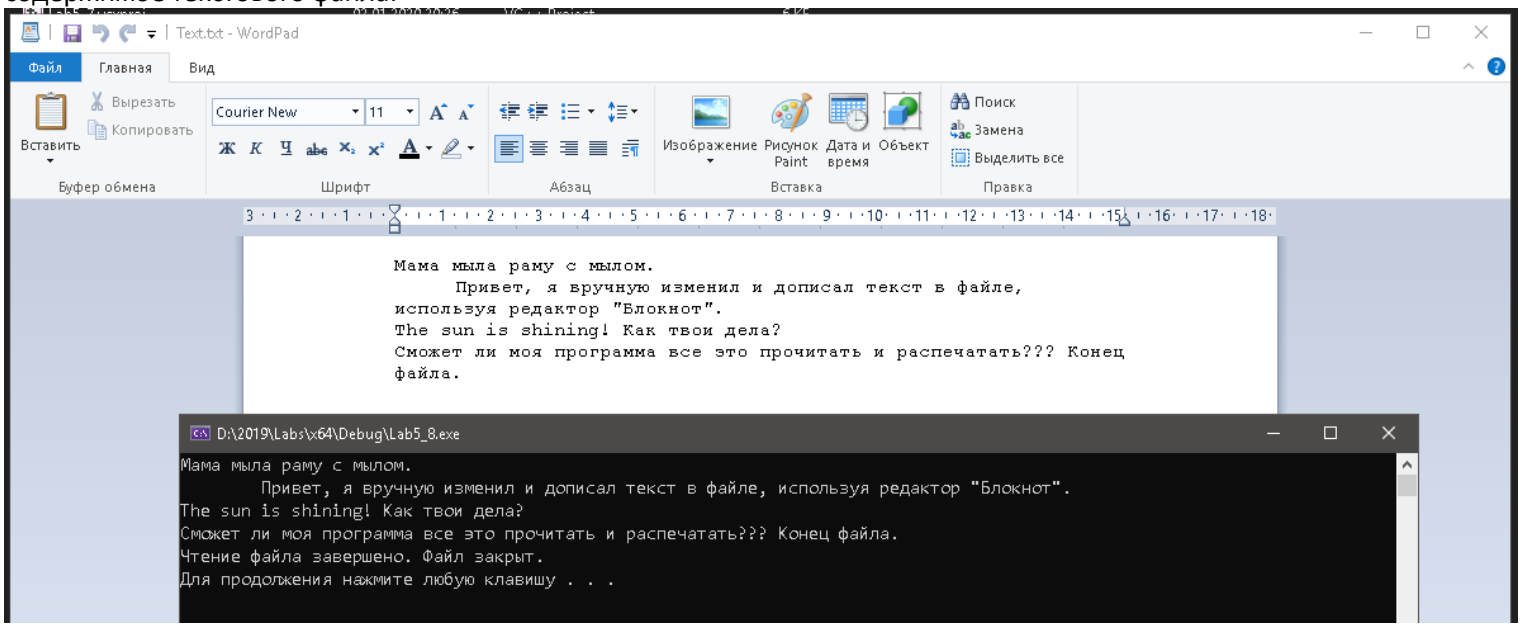
Текстовый файл пользователь может открыть и прочитать в текстовом редакторе, но нужно уметь написать программу, которая сама сможет прочитать данные (символы char) из текстового файла (записанные как другой программой, так и

пользователем или ими **обоими попеременно**) и, например, обработать их (сделать вычисления, сравнения, распечатать их на консоль).

```
#define _CRT_SECURE_NO_WARNINGS//надо для использования функций fopen() и некоторых других
#include <iostream>//вам может понадобится подключить библиотеки <stdio.h><conio.h>
#include <Windows.h>//для обеспечения поддержки русского языка, в том числе при записи и чтении файлов (по крайней мере в ОС Windows)
using namespace std;

int main()
{
    SetConsoleOutputCP(1251);
    SetConsoleCP(1251);
    FILE* rf;//указатель для потока работы с файлом. Проинициализируем наш указатель на поток работы с файлом rf с помощью функции fopen(), которая принимает первым параметром путь
    if ((rf = fopen("D:\\2019\\Labs\\Lab5_7\\Text.txt", "rt")) == NULL)//к файлу и вторым параметром - режим открытия файла: "rt" - "read text file" (читать из текстового файла).
    {//Если функция fopen() не сможет открыть файл для чтения, то она вернет пустой адрес (NULL), который присвоится указателю rf и мы в проверке if() это сразу проверяем и
        cout << "Ошибка открытия файла для чтения. Проверьте, что файл \"D:\\2019\\Labs\\Lab5_7\\Text.txt\" существует по указанному пути и не используется другими пользователями \
и приложениями.\n";//завершаем программу при неустраивающем нас результате проверки. Обратите внимание, как можно "разорвать" в коде строку символов в выражении cout<< спецсимволом
        system("pause");// '\' (обратным слэшем), при этом для компилятора это одна длинная строка символов (одна фраза)
        return 0;
    }
    char c = fgetc(rf);//читаем ДО цикла один первый символ из файла (потока чтения из файла rf) и помещаем его в буферную символьную переменную, содержимое которой распечатаем на
    while (!feof(rf))//консоль В цикле. Это делается из-за проверки на feof() в цикле while(!feof(rf)), в результате чего при более "простом" подходе, когда в самом цикле читаем
    {//символ из файла и сразу его печатаем на консоль, последний символ из файла прочитается и распечается на консоль ДВАЖДЫ, что НЕ отражает реального положения и количества
        cout << c;//символов в файле. Эта ошибка предупреждается таким подходом, когда мы сначала ДО цикла читаем первый символ, а печатать будем его В цикле, читая далее В цикле
        c = fgetc(rf);//следующие символы, но печатая их, получается, в СЛЕДУЮЩЕЙ итерации относительно чтения этих символов. В результате такого кода, последний символ
    }//напечатается ОДИН раз, что нам и нужно. При чтении словами или строками данный принцип тоже нужен. Проверьте это, "упростив" код.
    fclose(rf);//сразу, как только файл нам не нужен, закрываем поток работы с ним, закрывая тем самым файл и освобождая его для использования другими потоками и процессами
    cout << "\nЧтение файла завершено. Файл закрыт.\n";//сообщаем пользователю о завершении чтения файла (ведь файл может быть большим, а чтение текстовых файлов дольше по времени,
    system("pause");//чем чтение бинарных файлов с аналогичным (эквивалентным) содержимым) и закрытии потока чтения файла (и закрытии, тем самым, самого файла)
    return 0;
}//задание: попробуйте читаемые символы помещать в символьный массив char str[200];, закрыть файл и распечатать символьный массив str целиком на консоль cout << str << endl;
```

Вручную изменим текстовый файл, сохраним его и закроем. Протестируем, сможет ли программа прочитать все содержимое текстового файла:



Программа все прочитала корректно, включая набранные вручную пользователем Enter'ы (переносы курсора на следующую строку, что позволяет делать абзацы текста) и табуляционный пробел (клавиша Tab слева клавиатуры). Учтите, что если вы поменяете кодировку документа, то не гарантировано, что наша программа сможет корректно отобразить его содержимое.

Запись и чтение бинарных файлов в бинарном режиме (двоичном режиме)

Как записывать в файл сложные (составные) объекты, например, структуры struct, массивы структур struct, классы и так далее? Можно записывать содержимое их полей в текстовые файлы, но еще лучшим и более быстрым по времени выполнения является способ записи («сериализации» от англ. serialization – «публикация по частям, запись, сохранение в файл») объекта в бинарный (двоичный) файл. Запись и чтение данных в (из) бинарный файл нужно производить только в бинарном режиме. При бинарной записи в файл из оперативной памяти копируется участок байт с данными (содержимым) некоторого объекта, например, структуры Человек. Данные копируются побитово, то есть копируются нули 0 и единицы 1 в бинарной (двоичной) системе, как они и хранятся на самом деле в оперативной памяти, когда их создала наша программа. Участки с этими нулями и единицами копируются в файл и обычно их уже не возможно полностью корректно прочитать (просмотреть) посредством текстового редактора, но зато мы, зная декларацию, например, структуры struct, экземпляр которой записан в файл, можем написать программу или пункт в меню программы для чтения этих данных «кусками» байт из бинарного файла и помещения их в оперативную память, где эти данные можно программно снова превратить в объекты, распечатать содержимое их полей, ведь мы, как разработчики данной структуры, знаем ее декларацию (объявление), а значит знаем **формат данных** (количество, порядок расположения и типы данных полей структуры struct). Это правило действует и наоборот: если мы не знаем формата данных (что за структуры с какими полями записаны в бинарный файл), то мы не сможем написать программу для их гарантированного корректного чтения из бинарного файла. На практике, анализируя данные в бинарных файлах, открытые в специальных бинарных редакторах, методом **обратной разработки** программисты **воссоздают**

поля структур и классов и могут в итоге написать программу, читающую бинарные файлы, в которые записаны данные структур, разработанные другими программистами. Здесь уже возможны вопросы по авторским правам и лицензиям на ПО (программное обеспечение, форматы данных и файлов). Расширения бинарных (двоичных) файлов можно создавать самому, но обычно используют *.bin, *.dat, что является общеизвестным «говорящим» расширением для бинарных файлов, а собственное расширение разрабатывают и регистрируют (публикуют) для файлов определенного формата для определенной сферы использования, например, изображений *.bmp и так далее. Продуктивно этим заниматься могут коллективы разработчиков, фирмы, учреждения, в то время как в одиночку это делать малоперспективно. Тем не менее, по специальному «авторскому» расширению файла программа может находить файлы «своего» формата, которые она умеет читать, поскольку знает формат данных, в них помещенных. Если «вручную» в текстовом редакторе изменить бинарный файл, то высока вероятность того, что данные в нем будут повреждены и нечитаемы даже «родной» программой этого же разработчика. Таким образом, бинарные файлы не предназначены для текстовых редакторов. Бинарные файлы должны записываться и читаться специальными программами, которые «знают» формат хранимых в них данных. Такие программы являются специализированными редакторами бинарных файлов, но, как правило, обладают дополнительным функционалом, ради которого собственно и создаются бинарные файлы. Бинарные и текстовые файлы предназначены для хранения данных пользователя, программ, в то время, когда компьютер выключен или нужно перенести данные с одного компьютера (одного экземпляра программы) на другой компьютер (другому экземпляру этой же программы или аналогичной ей). Файлы можно дописывать (дозаписывать), в результате чего их можно использовать как накопители данных, то есть как базы данных. Речь идет о файлах на **устройствах постоянного (долговременного) хранения информации** – жестких дисках HDD, CD, DVD, Blue-Ray-дисках, твердотельных SSD-дисках, флешках, картах памяти типа microSD и так далее, но не **оперативной памяти**, в которой запускается на выполнение наша программа, но которая является **энергозависимой**, то есть вся информация безвозвратно исчезнет с планки оперативной памяти как только прекратится подача на нее электроэнергии даже на полсекунды (для оперативной памяти это большой промежуток времени).

```
#define _CRT_SECURE_NO_WARNINGS//надо для использования функций fopen() и некоторых других
#include <iostream>//вам может понадобится подключить библиотеки <stdio.h><conio.h>
#include <Windows.h>//для обеспечения поддержки русского языка, в том числе при записи и чтении файлов (по крайней мере в ОС Windows)
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);
    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\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;//есть что записывать в бинарный файл
            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;
            }
            FILE* f;//иначе записывем
            if (!(f = fopen("D:\\2019\\Labs\\Lab5_9\\BinaryHumans.bin", "ab")))//дозапись нового экземпляра типа структуры Human в конец бинарного файла
            {//проверка успешности открытия бинарного файла для записи данных в его конец (дозаписи в конец файла БЕЗ удаления "старых" записей - это режим "ab" add binary добавить
                cout << "Error.\n";//в конец бинарного файла (дописать еще, без удаления уже существующих в файле записей
                break;
            }
            fwrite(&h, sizeof(Human), 1, f);//пишем целиком весь экземпляр h типа структуры Human (первым параметром функция записи в бинарный файл fwrite() принимает укзатель на
            fclose(f);//объект, поэтому передаем ей адрес объекта, вторым параметром функции надо передать размер объекта в байтах (можно sizeof(Human) или sizeof(h)), третьим
            break;//параметром функция получает количество записываемых объектов (1 объект записать), четвертым параметром функция принимает указатель на поток записи в открытый
        }//бинарный файл. Функция fwrite() запишет в файл целиком весь объект h размером структуры Human из оперативной памяти, где он находится, копируя последобательно бит за
         //битом (нули 0 и единицы 1). Функция может скопировать за один раз целый объект, ей не надо его форматировать в процессе копирования, поэтому бинарная запись БЫСТРАЯ
        case 3:
        {
            Human h0;//создадим отдельную переменную h0, чтобы использовалась "пустая" локальная переменная и мы гарантировано видели результат работы функции бинарного чтения из
            FILE* f;//бинарного файла
            if (!(f = fopen("D:\\2019\\Labs\\Lab5_9\\BinaryHumans.bin", "rb")))//режим бинарного чтения read binary экземпляров структур типа структуры Human из бинарного файла
            {//стандартная проверка на открытие файла в нужном режиме
                cout << "Error.\n";
                break;
            }
            fread(&h0, sizeof(Human), 1, f);//читаем в бинарном режиме из файла, открытого в потоке f, 1 блок данных размером sizeof(Human) байт (или sizeof(h0)) и помещаем
            while (!feof(f))//результат по адресу перменной h0. Пока не достигнем конца файла, открытого в потоке f, 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";
                }
                fread(&h0, sizeof(Human), 1, f);//читаем функцией бинарного чтения fread() - file read, следующий участок размером sizeof(Human) байт бинарного файла и помещаем
            }//его (этот прочитанный участок байт) по адресу переменной h0, поля которой мы перезаписываем (это буферная переменная, она используется многократно в цикле чтения
            fclose(f);//бинарного файла в бинарном режиме. Подход тут аналогичный ранее рассмотренному для исключения чтения и печати последней структуры из файла дважды
            break;//в конце любой работы с файлом закрываем поток работы с файлом, в том числе и с бинарным. Можно закрывать все открытые файлы функцией fcloseall();, но она
        }//закроет вообще все открытые в данной программе файлы, что не всегда приемлемо
        }
    }
    system("pause");
    return 0;
}
```

```
D:\2019\Labs\x64\Debug\Lab5_9.exe
3-Читать и поочередно распечатывать на консоль данные из бинарного файла о всех записанных в нем людях
Введите номер нужного вам пункта меню: 2
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
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
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
Введите номер нужного вам пункта меню: 1
Введите возраст человека (полных лет): 44
Введите рост человека (в метрах): 1.44
Введите фамилию человека (можно с пробелами): Попов
Введите имя человека: Павел
Введите пол человека (0-женский, 1-мужской): 1
Вы ввели данные:
Попов   |       Павел   |       44      |       1.44    |       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
Введите номер нужного вам пункта меню: 2
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
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
Попов   |       Павел   |       44      |       1.44    |       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
Введите номер нужного вам пункта меню:
```

Запись структур struct в текстовый файл в текстовом режиме

Аналогичного результата можно добиться, записывая текстовый файл в текстовом режиме. В этом случае мы должны из оперативной памяти читать содержимое полей экземпляров структур и записывать их в текстовый(-ые) файл(-ы) в определенном порядке с определенными разделителями (например, пробелами). Поскольку в файл может понадобиться записывать много объектов, то удобно каждый отдельный объект (содержимое его полей) записывать с новой строчки текстового файла и также построчно читать и извлекать содержимое полей, помещая данные в поля буферной переменной или даже полей элементов динамического массива (менее удобно, но можно и статического). Код будет примерно следующий (нужно дописать кейс дозаписи (дописывания) и чтения в текстовом режиме текстового файла и дописать соответственно пункты меню программы):

```
    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\nВведите номер нужного вам пункта меню: ";
        cin >> p;//пользователь с клавиатуры выбирает пункт меню, в том числе и вариант завершения работы этой программы
```

```
        case 4://кейс записи структуры в текстовый файл в текстовом режиме
        {
            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;
            }
            FILE* f;
            if (!(f = fopen("D:\\2019\\Labs\\Lab5_9\\TextHumans.txt", "at")))//режим текстовой записи (дозаписи, дописывания при сохранении уже имеющихся в файле записей)
            {//содержимого полей экземпляров структур типа структуры Human в конец текстового файла
                cout << "Error.\n";//стандартная проверка на открытие файла в нужном режиме
                break;
            }
            fprintf_s(f, "%d %lf %d %s %s\n", h.age, h.height, h.sex, h.name, h.fam);//каждое поле, перечисленное в конце списка входных параметров функции fprintf_s() имеет
            fclose(f);//свой спецификатор формата во втором параметре функции, например, %d - h.age, %lf - h.height, %d - h.sex, %s - h.name, %s - h.fam, они соотносятся друг с
            break;//другом попарно по порядку следования каждый из своего списка. Контролируйте соответствие их типов, количества и порядка следования
        }
        }
    }
    system("pause");
    return 0;
}
```

```
D:\2019\Labs\x64\Debug\Lab5_9.exe
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): 66
Введите рост человека (в метрах): 1.66
Введите фамилию человека (можно с пробелами): Антонова
Введите имя человека: Анна
Введите пол человека (0-женский, 1-мужской): 0
Вы ввели данные:
Антонова        |       Анна    |       66      |       1.66    |       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: 4
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): 77
Введите рост человека (в метрах): 1.77
Введите фамилию человека (можно с пробелами): Леонова
Введите имя человека: Елена
Введите пол человека (0-женский, 1-мужской): 0
Вы ввели данные:
Леонова |       Елена    |       77      |       1.77    |       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: 4
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:
```

TextHumans.txt — Блокнот

Файл Правка Формат Вид Справка

```
55 1.550000 1 Андрей Скворцов
66 1.660000 0 Анна Антонова
77 1.770000 0 Елена Леонова
```

Windows Стр 1, ст 100%

Задания по вариантам

Каждому учащемуся выполнить все 11 заданий, реализовав их в виде кейсов, выбираемых посредством меню в «бесконечном» цикле

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 «вашаФамилияИмя.txt» посредством посимвольной записи.
3	Посимвольно считать из вашего текстового файла с именем «вашаФамилияИмя.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 «вашаФамилияИмя0.txt» посредством функции fprintf(). Записываемые значения разделяйте символами. Тип массива и разделяющие символы (один или несколько) определяются вашим номером в журнале группы: 1 – double, разделитель " | " 2 – float, разделитель " @ " 3 – int, разделитель " # "

	4 – long int, разделитель “ % “ 5 – short int, разделитель “ $ “ 6 – char, разделитель '_' 7 – double, разделитель '|' 8 – float, разделитель '@' 9 – int, разделитель '#' 10 – long int, разделитель '%' 11 – short int, разделитель '$' 12 – char, разделитель '{' 13 – double, разделитель “, “ 14 – float, разделитель ”; “ 15 – int, разделитель ':' 16 – long int, разделитель '*' 17 – short int, разделитель '+' 18 – char, разделитель '=' 19 – double, разделитель '-' 20 – float, разделитель '?' 21 – int, разделитель '!' 22 – long int, разделитель '/' 23 – short int, разделитель '\' 24 – char, разделитель '~' 25 – double, разделитель ”} “ 26 – float, разделитель ”. “ 27 – int, разделитель '.' 28 – long int, разделитель '(' 29 – short int, разделитель ')' 30 – char, разделитель '^'
5	Считать из текстового файла с именем «ваша**Фамилия**Имя0.txt» элементы массива посредством функции fscanf() 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 «ваша**Фамилия**Имя1.txt» посредством функции fprintf(). 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 «ваша**Фамилия**Имя1.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 «ваша**Фамилия**Имя2.bin» посредством функции fwrite(). 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 «ваша**Фамилия**Имя2.bin» все записанные в нем структуры 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	Закрывайте файл (закрывайте поток работы с файлом) в конце **каждого** кейса.
11	Предусмотрите пункт меню для завершения работы вашей программы.

**В качестве дополнительного задания можно реализовать работу программы, которая ведет журнал своей работы (log-файл), то есть протоколирует (дозаписывает, добавляет в конец файла) все действия, которые она совершает (по чьему требованию (сеанс какого пользователя), с каким файлом работа (путь к файлу), (не)успешность действия), в один отдельный текстовый файл.*