

Балтийский государственный технический университет
«ВОЕНМЕХ» им. Д. Ф. Устинова

Кафедра И5 «Информационные системы и программная инженерия»

Практическая работа №7
по дисциплине «Информатика: Основы программирования»
на тему «Файлы»

Выполнил:
Студент Альков В.С.
Группа И407Б

Преподаватель:
Першин Д.В.

Санкт-Петербург
2020 г.

Задача 1. Дан файл, содержащий некоторый текст. Удалить из файла все фразы, заканчивающиеся и начинающиеся на одну и ту же букву.

Уровень сложности – повышенный. Имя входного файла должно передаваться программе при ее запуске (через параметры функции `main()`). Если параметры пользователем при запуске программы не заданы, имя файла вводится с клавиатуры. Исходный файл может содержать как латинские, так и русские буквы, фразы могут быть любой длины, соответственно, и на одной строке может находиться несколько фраз, и одна фраза может располагаться на нескольких строках. Фразы отделяются друг от друга точками, а слова – пробелами, знаками препинания и символами конца строки. Последняя фраза в файле может быть без точки в конце.

Исходные данные:

argc – кол-во аргументов, переданных в `main`, тип *int*;

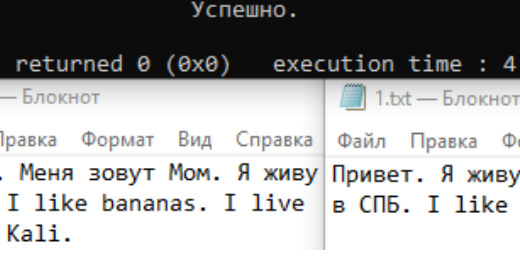
files – указатель на массив строк (аргументов, в нашем случае имена файлов), переданных в `main`, тип `char**`;

filename – имя файла, введенное пользователем, массив символов типа *char*.

Результирующие данные:

вывод на экран сообщения об успехе иди неудачи работы алгоритма, печать в файл обработанный исходный файл.

Таблица тестирования:



```
Введите названия файлов:1.txt
Файл                Результат
1.txt                Успешно.

Process returned 0 (0x0)   execution time : 4.283 s

1.txt — Блокнот
Файл  Правка  Формат  Вид  Справка
Привет. Меня зовут Мом. Я живу
в СПб. I like bananas. I live
in the Kali.

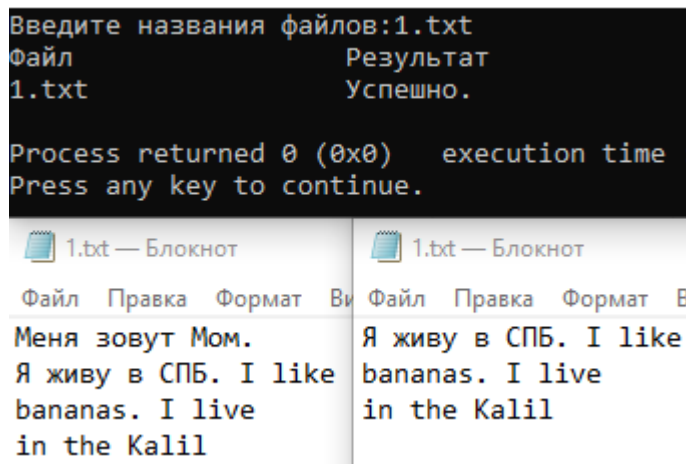
1.txt — Блокнот
Файл  Правка  Формат  Вид
Привет. Я живу
в СПб. I like bananas.

Введите названия файлов:1.txt
Файл                Результат
1.txt                Успешно.

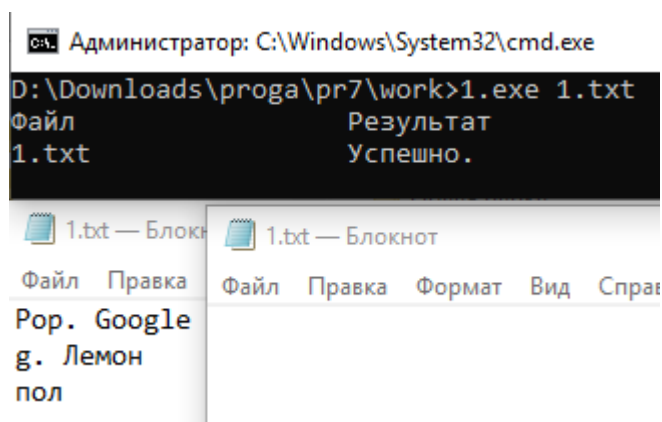
Process returned 0 (0x0)   execution time : 4.060 s
Press any key to continue.

1.txt — Блокнот
Файл  Правка  Формат  Вид  Справка
Меня зовут Мом.
Я живу в СПб. I like
bananas. I live
in the Kali

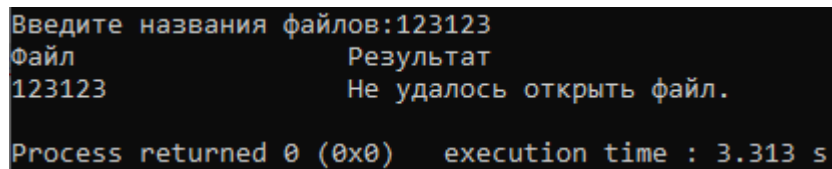
1.txt — Блокнот
Файл  Правка  Формат  Вид  Справка
Я живу в СПб. I like
bananas.
```



Запуск через cmd



Несуществующий файл



Вспомогательные алгоритмы:

- 1) Алгоритм, отвечающий за работу программы, если были переданы параметры в main.

Входные данные: кол-во аргументов, имя указателя на массив аргументов.

Результирующие данные: отсутствуют.

Этот алгоритм описывается в программе функцией

`void files_is_not_empty(int, char*[])`

Параметры:

первый параметр – кол-во аргументов;

второй параметр – адрес первого аргумента.

Возвращаемое значение – отсутствует.

Вспомогательные переменные:

i – тип *int*, индекс массива аргументов.

- 2) Алгоритм, отвечающий за работу программы, если не были переданы параметры в *main*.

Входные данные: отсутствуют.

Резльтирующие данные: ввод имен файлов.

Этот алгоритм описывается в программе функцией

void files_is_empty()

Параметры: отсутствуют.

Возвращаемое значение – отсутствует.

Вспомогательные переменные:

filename – массив символов типа *char*, имя файла.

c – тип *char*, буферная переменная для определения конца ввода.

- 3) Алгоритм удаления предложений, начинающихся и заканчивающихся на одну букву из файла.

Входные данные: имя файла (строка).

Резльтирующие данные: удалены предложения начинающиеся и заканчивающиеся на одну букву из файла.

Этот алгоритм описывается в программе функцией

*int files_work(char *)*

Параметры:

первый параметр – адрес начала строки.

Возвращаемое значение – отсутствует.

Вспомогательные переменные:

f1, f2 – указатели на поток, на файл, *f1* – исходный файл, *f2* – для перезаписи нужных данных из первого, тип *FILE **;

c, cbuf – считанный символ из файла, тип *int*;

i – длина предложения, строки *buf*, тип *int*;

buf – строка, динамический массив символов, содержит считанное предложение, тип *char**.

- 4) Алгоритм проверки символа на букву.

Входные данные: символ.

Резльтирующие данные: отсутствуют.

Этот алгоритм описывается в программе функцией

int check(char)

Параметры:

первый параметр – код символа.

Возвращаемое значение – 1, если переданный символ – буква, 0, если нет.

- 5) Алгоритм проверки предложения на различие первой и последней буквы.

Входные данные: имя строки.

Результирующие данные: отсутствуют.

`int check_sentence(char*)`

Параметры:

первый параметр – адрес начала строки.

Возвращаемое значение – 1, если первая и последняя буква различны, 0 – если одинаковы.

Вспомогательные переменные:

c1, c2 – первая и последняя буква строки, тип *char*;

i – индекс переданной строки, тип *int*;

- 6) Алгоритм перевода буквы в верхний регистр.

Входные данные: символ.

Результирующие данные: отсутствуют.

Этот алгоритм описывается в программе функцией

`int toupper(char)`

Параметры:

первый параметр – код символа.

Возвращаемое значение – символ.

- 7) Алгоритм удаления символов с начала строки.

Входные данные: имя строки, кол-во символов к удалению.

Результирующие данные: символы удалены.

Этот алгоритм описывается в программе функцией

`void delete_symbols(char *, int)`

Параметры:

первый параметр – адрес начала строки;

второй параметр – кол-во символов к удалению.

Возвращаемое значение – отсутствует.

Вспомогательные переменные:

i – индекс переданной строки, тип *int*;

len – длина переданной строки, тип *int*.

8) Алгоритм удаления пробелов и переносов каретки с начала строки.

Входные данные: имя строки.

Результирующие данные: символы удалены.

Этот алгоритм описывается в программе функцией

void replace_n(char)*

Параметры:

первый параметр – адрес начала строки.

Возвращаемое значение – отсутствует.

Вспомогательные переменные:

i – индекс переданной строки, тип *int*.

Текст программы

```
#include <stdio.h>
#include <locale.h>
#include <stdlib.h>
#include <string.h>
/*объявление ф-ий*/
int toup(char);
int check_sentence(char*);
int check(char);
void delete_symbols(char*, int);
void replace_n(char*);
int files_work(char*);
void files_is_empty();
void files_is_not_empty(int, char*[]);

int main(int argc, char* files[])
{
    /*добавление поддержки русского языка*/
    setlocale(LC_ALL, "rus");
    /*объявление переменных*/
    char filename[80] = "";
    /*проверяем были ли переданы параметры в ф-ию main*/
    if(argc<2)
        files_is_empty();
    else
        files_is_not_empty(argc, files);
    return 0;
}

/*ф-ия, если были переданы параметры в main*/
void files_is_not_empty(int argc, char* files[])
{
    int i;
    printf("%-20.20s %-20.20s\n", "Файл", "Результат");
    /*обрабатываем каждый переданный файл*/
    for (i=1; i<argc; i++)
    {
        printf("%-20.20s ", files[i]);
        files_work(files[i]);
        printf("\n");
    };
}

/*ф-ия, если не были переданы параметры в main*/
void files_is_empty()
```

```

{
    int i=0;
    char c, filename[80] = "";
    printf("Введите названия файлов:");
    /*названия файлов вводить на одной строке*/
    do
    {
        /*ввод названия файла*/
        scanf("%s", filename);
        if(i==0)
        {
            printf("%-20.20s %-20.20s\n", "Файл", "Результат");
            i++;
        };
        printf("%-20.20s ", filename);
        /*обработка файла*/
        files_work(filename);
        printf("\n");
        /*очищаем поток ввода от пробелов*/
        while((c=fgetc(stdin))==' ');
        /*если не достигли конца потока ввода, то есть, если считанный
        символ не перевод каретки, то возвращаем символ в поток*/
        if(c!='\n')
            ungetc(c, stdin);
        /*пока символ не перевод каретки*/
    } while(c!='\n');
}

/*Ф-ия обработки файла*/
int files_work(char *filename)
{
    /*объявление переменных*/
    FILE *f1, *f2;
    int c, cbuf, i=0;
    char *buf;
    /*открываем файл для чтения*/
    f1 = fopen(filename, "r");
    /*проверяем открылся ли файл*/
    if (!f1)
    {
        printf("Не удалось открыть файл.");
        return 1;
    };
    /*открываем файл для редактирования*/
    f2 = fopen("temp.txt", "w");
    /*проверяем открылся ли файл*/
    if (!f2)
    {
        printf("Не удалось создать файл для записи.");
        return 1;
    };
    /*выделяем память, чтобы можно было ее перевыделять*/
    buf = calloc(1,1);
    /*пока не достигли конца файла*/
    while ((c=fgetc(f1)) != EOF)
    {
        /*перевыделяем память, чтобы можно было записать в buf еще один
        символ, i - длина строки buf*/
        buf = realloc(buf, 2+i);
        /*записываем считанный символ в конец строки*/
        buf[i] = c;
        /*проверяем достигли ли конца предложения,
        если считанный символ точка или последний в файле*/
    }
}

```

```

        if((cbuf=fgetc(f1)) == EOF || c=='.')
        {
            /*то формируем предложение, то есть отделяем все символы после
            в buf, ведь там может быть мусор, оставшийся от предыдущих
            предложений*/
            buf[i+1] = '\0';
            /*если файл, в который производится запись, пустой, то удаляем
            пробелы и переносы каретки в начале строки buf*/
            if(!ftell(f2))
                replace_n(buf);
            /*если первая и последняя буквы строки разные,
            то записываем строку в файл*/
            if (check_sentence(buf))
                fprintf(f2, "%s", buf);
            /*сбрасываем счетчик длины строки buf*/
            i=-1;
        };
        /*возвращаем символ в поток ввода*/
        ungetc(cbuf,f1);
        /*увеличиваем счетчик длины строки*/
        i++;
    };
    /*закрываем файлы*/
    fclose(f1);
    fclose(f2);
    /*освобождаем память, выделенную под buf*/
    free(buf);
    /*производим замену исходного файла новым*/
    remove(filename);
    rename("temp.txt", filename);
    printf("Успешно.");
    return 0;
}

/*Ф-ия проверки символа на букву*/
int check(char a)
{
    if ((a>='a' && a <='z') || (a>='A' && a <='Z') || (a>='А' && a
        <='Я') || (a>='а' && a <='я'))
        return 1;
    return 0;
}

/*Ф-ия проверки строки на различие первой и последней буквы*/
int check_sentence(char* a)
{
    int i;
    char c1, c2;
    for(i=0; !check(a[i])&&a[i]!='\0'; i++);
    c1=a[i];
    for(i; a[i]!='\0'; i++)
        if(check(a[i]))
            c2=a[i];
    return toupper(c1)!=toupper(c2);
}

/*Ф-ия перевода в верхний регистр*/
int toupper(char c)
{
    if ((c>='a'&& c<='z') || (c>='а'&& c<='я'))
        c-=32;
    return c;
}

```



```

/*Ф-ия удаления символов с начала строки*/
void delete_symbols(char *a, int b)
{
    int i, len = strlen(a);
    for (i=0; i<len-b+1&& a[i+b-1]!='\0'; i++)
        a[i] = a[i+b];
}
/*Ф-ия удаления перевода каретки и пробелов с начала строки*/
void replace_n(char* a)
{
    int i;
    for(i=0; !check(a[i])&&a[i]!='\0'; i++)
        if (a[i] == '\n' || a[i] == ' ')
        {
            delete_symbols(a, 1);
            i--;
        };
}

```

Задача 2. В текстовом файле содержится целочисленная матрица. Определить количество простых чисел в каждой строке матрицы, результаты записать в новый текстовый файл с указанием номеров строк исходного файла.

Уровень сложности – повышенный. Имя входного файла должно передаваться программе при ее запуске (через параметры функции main()). Если параметры пользователем при запуске программы не заданы, имя файла вводится с клавиатуры.

Исходные данные:

argc – кол-во аргументов, переданных в main, тип *int*;

files – указатель на массив строк (аргументов, в нашем случае имена файлов), переданных в main, тип *char*[]*;

filename – имя файла, введенное пользователем, массив символов типа *char*.

Результирующие данные:

вывод на экран сообщения об успехе или неудаче работы алгоритма, печать в файл кол-во простых чисел в каждой строке.

Таблица тестирования:

The screenshot shows a Windows desktop with the following elements:

- Command Prompt:** Displays the execution of a program. It prompts for file names, takes '1.txt' as input, and outputs 'Успешно.' (Successfully). The process returns 0 (0x0).
- 1.txt (Блокнот):** Contains the input matrix:

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
- 1_res2.txt (Блокнот):** Contains the output results:

1	3
2	1
3	2
4	2
- 1.txt (Блокнот):** Contains the input matrix (repeated from above).
- 1_res2.txt (Блокнот):** Contains the output results (repeated from above).

```
Администратор: C:\Windows\System32\cmd.exe
D:\Downloads\proga\pr7\work>2.exe 1.txt
Файл          Результат
1.txt          Успешно.
```

Вспомогательные алгоритмы:

- 1) Алгоритм, отвечающий за работу программы, если были переданы параметры в main и описывающая его функция те же, что и в предыдущей задаче.
- 2) Алгоритм, отвечающий за работу программы, если не были переданы параметры в main и описывающая его функция те же, что и в предыдущей задаче.
- 3) Алгоритм считывания числа из файла.

Входные данные: имя потока, имя указателя.

Результирующие данные: считано число из файла по переданному адресу.

Этот алгоритм описывается в программе функцией

int read_int(FILE, int*)*

Параметры:

первый параметр – адрес потока;

второй параметр – адрес, по которому записать значение.

Возвращаемое значение – 1 - достигли конца строки, 2 - достигли конца файла,
0 - можно считывать дальше.

Вспомогательные переменные:

c – считанный символ, тип int.

- 4) Алгоритм считывания строки матрицы из файла и запись кол-ва простых чисел в файл.

Входные данные: имя потока чтения, имя потока записи.

Результирующие данные: записано в файл кол-во простых чисел в каждой строке матрицы.

Этот алгоритм описывается в программе функцией

int read_line(FILE, FILE*)*

Параметры:

первый параметр – адрес потока чтения;

второй параметр – адрес потока записи.

Возвращаемое значение – 1 - достигли конца строки, 2 - достигли конца файла.

Вспомогательные переменные:

c – буферная переменная для проверки конца файла, тип int;

num – считанное число, тип int;

p – флаг конца строки – 1, или конца файла – 2, тип *int*;

k – кол-во простых чисел в строке, тип *int*;

i – номер строки матрицы, тип *static int*.

5) Алгоритм проверки числа на простоту.

Входные данные: число.

Результирующие данные: отсутствуют.

Этот алгоритм описывается в программе функцией

int check_prime(int)

Параметры:

первый параметр – число.

Возвращаемое значение – 1 – число простое, 0 – число непростое.

6) Алгоритм проверки символа на цифру.

Входные данные: символ.

Результирующие данные: отсутствуют.

Этот алгоритм описывается в программе функцией

int check_num(char)

Параметры:

первый параметр – символ.

Возвращаемое значение – 1 – число, 0 – не число.

Текст программы

```
#include <stdio.h>
#include <locale.h>
#include <string.h>
/*объявление ф-ий*/
int check_prime(int);
int read_line(FILE*, FILE*);
int read_int(FILE*, int*);
int check_num(char);
int files_work(char*);
void files_is_empty();
void files_is_not_empty(int, char*[]);

int main(int argc, char* files[])
{
    /*поддержка русского языка*/
    setlocale(LC_ALL, "rus");
    /*объявление переменных*/
    char c, filename[80] = "";
    if(argc<2)
        files_is_empty();
    else
        files_is_not_empty(argc, files);
    return 0;
}

void files_is_not_empty(int argc, char* files[])
```

```

{
    int i;
    printf("%-20.20s %-20.20s\n", "Файл", "Результат");
    for (i=1; i<argc; i++)
    {
        printf("%-20.20s ", files[i]);
        files_work(files[i]);
        printf("\n");
    };
}
int files_work(char* filename)
{
    /*объявление переменных*/
    int p;
    char newfilename[80];
    FILE *f1, *f2;
    /*создаем имя нового файла*/
    strcpy(newfilename, filename);
    newfilename[strlen(newfilename)-4] = '\0';
    strcat(newfilename, "_res2.txt");
    f1 = fopen(filename, "r");
    if (!f1)
    {
        printf("Не удалось открыть файл.");
        return 1;
    };
    f2 = fopen(newfilename, "w");
    if (!f2)
    {
        printf("Не удалось создать файл для записи.");
        return 2;
    };
    /*считываем, пока не достигли конца файла*/
    while((p=read_line(f1, f2))!=2);
    /*закрываем файлы*/
    fclose(f1);
    fclose(f2);
    rename("temp.txt", newfilename);
    printf("Успешно.");
    return 0;
}

void files_is_empty()
{
    int i=0;
    char c, filename[80] = "";
    printf("Введите названия файлов:");
    do
    {
        scanf("%s", filename);
        if(i==0)
        {
            printf("%-20.20s %-20.20s\n", "Файл", "Результат");
            i++;
        };
        printf("%-20.20s ", filename);
        files_work(filename);
        printf("\n");
        while((c=fgetc(stdin))==' ');
        if(c!='\n')
            ungetc(c, stdin);
    } while(c!='\n');
}

```

```

/*Ф-ия чтения числа из файла и запись по адресу, возвращает:
1 - достигли конца строки, 2 - достигли конца файла, 0 - можно считать
дальше*/
int read_int(FILE*f, int* num)
{
    int c=fgetc(f);
    /*пропускаем не цифры*/
    while(!check_num(c))
    {
        if(c =='\n')
            return 1;
        if(c ==EOF)
            return 2;
        c=fgetc(f);
    };
    /*вводим число*/
    if(check_num(c));
    {
        ungetc(c,f);
        fscanf(f,"%d",num);
    };
    return 0;
}
/*Ф-ия чтения строки матрицы из файла и записи кол-ва простых чисел в
файл*/
int read_line(FILE* f, FILE* f1)
{
    /*объявление переменных*/
    int num, p, k=0, c;
    /*статичная для сохранения номера строки матрицы*/
    static int i=1;
    /*считываем числа*/
    while(!(p=read_int(f, &num)))
    {
        /*считаем простые числа*/
        if(check_prime(num))
            k++;
    };
    /*очищаем ввод от не цифр*/
    while(!check_num(c=fgetc(f))&&c!=EOF);
    /*если достигли конца файла, запоминаем это*/
    if (c==EOF)
        p=2;
    else
        /*если нет, то возвращаем символ в поток*/
        ungetc(c,f);
    /*печатаем в файл кол-во простых чисел в строке*/
    fprintf(f1, "%d %d\n", i++, k);
    return p;
}

/*Ф-ий проверки числа на простоту*/
int check_prime(int a)
{
    int i;
    if (a<2)
        return 0;
    for(i=2; i<=a/2; i++)
        if(a%i==0)
            return 0;
    return 1;
}
/*Ф-ия проверки символа на цифру*/

```

```

int check_num(char c)
{
    if(c>='0'&&c<='9' || c == '-')
        return 1;
    return 0;
}

```

Задача 3. Компоненты бинарного файла – вещественные числа.

Поменять местами первый компонент файла с минимальным, а последний – с максимальным. Для создания исходного бинарного файла написать отдельную программу, в программе его обработки выводить на экран содержимое файла до и после изменения.

Уровень сложности – повышенный. Имя входного файла должно передаваться программе при ее запуске (через параметры функции `main()`). Если параметры пользователем при запуске программы не заданы, имя файла вводится с клавиатуры.

Исходные данные:

Программа, создающая файл:

f – указатель на поток, тип *FILE**.

Программа, обрабатывающая файл:

argc – кол-во аргументов, переданных в `main`, тип *int*;

files – указатель на массив строк (аргументов, в нашем случае имена файлов), переданных в `main`, тип *char*[]*;

filename – имя файла, введенное пользователем, массив символов типа *char*.

Результирующие данные:

измененный файл, печать на экран содержимого файла до и после.

Вспомогательные переменные: отсутствуют.

Таблица тестирования:

```

100 1 2 3 -12 4 5 6 7 8
Process returned 0 (0x0)
Enter files names:3.bin
3.bin
File before changes
100.00 1.00 2.00 3.00 -12.00 4.00 5.00 6.00 7.00 8.00
File after changes
-12.00 1.00 2.00 3.00 8.00 4.00 5.00 6.00 7.00 100.00
Process returned 0 (0x0)   execution time : 2.495 s

100 -12
Process returned 0

```

```
Enter files names:3.bin
3.bin
File before changes
100.00 -12.00
File after changes
-12.00 100.00

Process returned 0 (0x0)
```

Вспомогательные алгоритмы:

Программа, создающая файл:

- 1) Алгоритм считывания числа типа double из потока.

Входные данные: имя потока.

Результирующие данные: считано число из потока по переданному адресу.

Этот алгоритм описывается в программе функцией

int read_double(FILE, double*)*

Параметры:

первый параметр – адрес потока;

второй параметр – адрес, по которому записать значение.

Возвращаемое значение – 1 - достигли конца строки, 0 – не достигли.

Вспомогательные переменные:

c – считанный символ, тип *int*.

- 2) Алгоритм считывания строки чтения строки чисел из потока и запись ее в файл

Входные данные: имя потока записи.

Результирующие данные: числа записаны в файл.

Этот алгоритм описывается в программе функцией

void read_line(FILE f)*

Параметры:

первый параметр – адрес потока записи.

Возвращаемое значение – отсутствует.

Вспомогательные переменные:

num – считанное число, тип *double*;

- 3) Алгоритм проверки символа на цифру и описывающая его функция те же, что и в предыдущей задаче.

Программа, обрабатывающая файлы:

- 1) Алгоритм, отвечающий за работу программы, если были переданы параметры в *main* и описывающая его функция те же, что и в предыдущей задаче.

- 2) Алгоритм, отвечающий за работу программы, если не были переданы параметры в main и описывающая его функция те же, что и в предыдущей задаче.
- 3) Алгоритм считывания числа по его позиции из потока.

Входные данные: имя потока, позиция.

Результирующие данные: отсутствуют.

Этот алгоритм описывается в программе функцией

double readpos(FILE, int)*

Параметры:

первый параметр – адрес потока;

второй параметр – позиция.

Возвращаемое значение – считанное число.

- 4) Алгоритм поиска минимума или максимума в файле.

Входные данные: имя потока, параметр

Результирующие данные: отсутствуют.

Этот алгоритм описывается в программе функцией

int minmax(FILE, int)*

Параметры:

первый параметр – адрес потока записи.

второй параметр – 1 – поиск максимума, -1 – поиск минимума.

Возвращаемое значение – найденное число.

Вспомогательные переменные:

buf – первое число для сравнения, тип *double*;

num – считанное число, тип *double*;

pos – позиция подходящего числа, тип *int*.

- 5) Алгоритм перестановки двух чисел местами в файле.

Входные данные: имя потока, позиция первого, позиция второго.

Результирующие данные: числа переставлены.

Этот алгоритм описывается в программе функцией

void replace(FILE, int, int)*

Параметры:

первый параметр – адрес потока записи;

второй параметр – позиция первого;

третий параметр – позиция второго.

Возвращаемое значение – отсутствует.

Вспомогательные переменные:

num1 – значение первого, тип *double*;

num2 – значение второго, тип *double*.

- 6) Алгоритм перестановки местам первого числа в файле с минимальным, последнего с максимальным.

Входные данные: имя файла.

Результирующие данные: числа переставлены.

Этот алгоритм описывается в программе функцией

void files_work(char)*

Параметры:

первый параметр – адрес начала строки.

Возвращаемое значение – отсутствует.

Вспомогательные переменные:

posmax, posmin, posend – позиции максимального числа, минимального числа, конца файла, тип *int*;

- 7) Алгоритм вывода содержимого файла на экран.

Входные данные: имя потока.

Результирующие данные: отсутствуют.

Этот алгоритм описывается в программе функцией

void printf_file(FILE)*

Параметры:

первый параметр – адрес потока.

Возвращаемое значение – отсутствует.

Вспомогательные переменные:

num – считанное число, тип *double*.

Текст программы, создающей файл

```
#include <stdio.h>
int check_num(char);
int read_double(FILE*, double*);
void read_line(FILE*);

int main()
{
    FILE* f;
    /*открываем файл в режиме записи для бинарных*/
    f=fopen("3.bin", "wb");
    if(!f)
    {
        printf("Error.\n");
    }
}
```

```

        return 1;
    };
    /*читаем числа из потока ввода*/
    read_line(f);
    /*закрываем файл*/
    fclose(f);
    return 0;
}
/*Ф-ия чтения числа из потока ввода, возвращает: если достигли конца строки
- 1, в другом случае - 0*/
int read_double(FILE*f, double* num)
{
    int c=fgetc(f);
    /*пропускаем не цифры*/
    while(!check_num(c))
    {
        /*если достигли конца строки, возвращаем сигнал об этом*/
        if(c =='\n')
            return 1;
        c=fgetc(f);
    };
    /*если считанный символ - цифра, то возвращаем
    последний считанный в поток ввода, и считываем число*/
    if(check_num(c));
    {
        ungetc(c,f);
        fscanf(f,"%lf",num);
    };
    return 0;
}
/*Ф-ия чтения строки чисел из потока и запись ее в файл*/
void read_line(FILE* f)
{
    double num;
    /*читаем по одному числу из потока ввода,
    пока read_double не вернет сигнал о конце строки = 1*/
    while(!read_double(stdin, &num))
        fwrite(&num, sizeof(double), 1, f);
}
/*проверка символа на цифру*/
int check_num(char c)
{
    if(c>='0'&&c<='9' || c == '-')
        return 1;
    return 0;
}

```

Текст программы, обрабатывающей файлы

```

#include <stdio.h>
/*объявление Ф-ий*/
void printf_file(FILE*);
int minmax(FILE*, int);
double readpos(FILE*, int);
void replace(FILE*, int, int);
void files_work(char*);
void files_is_empty();
void files_is_not_empty(int, char*[]);

int main(int argc, char* files[])
{
    if(argc<2)
        files_is_empty();
    else

```

```

        files_is_not_empty(argc, files);
    return 0;
}

void files_is_not_empty(int argc, char* files[])
{
    int i;
    for (i=1; i<argc; i++)
    {
        printf("%-20.20s ",files[i]);
        files_work(files[i]);
        printf("\n");
    };
}

void files_is_empty()
{
    char c, filename[80] = "";
    printf("Enter files names:");
    do
    {
        scanf("%s", filename);
        printf("%s\n",filename);
        files_work(filename);
        printf("\n");
        while((c=fgetc(stdin))== ' ');
        if(c!='\n')
            ungetc(c,stdin);
    } while(c!='\n');
}

void files_work(char* filename)
{
    /*объявление переменных*/
    FILE* f;
    int posmax, posmin, posend;
    f = fopen(filename, "rb+");
    if (!f)
    {
        printf("Can't open file.\n");
        return;
    };
    printf("File before changes\n");
    printf_file(f);
    /*определяем позицию минимального*/
    posmin = minmax(f, -1);
    /*меняем минимальное с первым местами*/
    replace(f,posmin, 0);
    /*определяем позицию последнего числа*/
    fseek(f,0,SEEK_END);
    posend = ftell(f);
    /*определяем позицию максимального*/
    posmax = minmax(f, 1);
    /*меняем максимальное с последним местами*/
    replace(f,posmax, posend-sizeof(double));
    printf("File after changes\n");
    printf_file(f);
    fclose(f);
}

/*ф-ия перестановки двух чисел местами в файле*/
void replace(FILE* f, int pos1, int pos2)
{
    double num1, num2;

```

```

        /*читаем числа из файла по их позиции*/
        num1 = readpos(f,pos1);
        num2 = readpos(f,pos2);
        /*задаем позицию в файле на первое число*/
        fseek(f,pos1,SEEK_SET);
        /*записываем на его место второе*/
        fwrite(&num2,sizeof(double),1,f);
        /*задаем позицию в файле на второе число*/
        fseek(f,pos2,SEEK_SET);
        /*записываем на его место первое*/
        fwrite(&num1,sizeof(double),1,f);
    }
    /*функция чтения числа по позиции*/
    double readpos(FILE* f, int pos)
    {
        double num;
        /*задаем позицию в файле на число*/
        fseek(f, pos, SEEK_SET);
        /*записываем число в num*/
        fread(&num, sizeof(double), 1, f);
        return num;
    }

    /*ф-ия поиска в файле, в зависимости от флага p, минимального - p=-1, или
        максимального числа - p=1*/
    int minmax(FILE* f, int p)
    {
        double num, buf;
        int pos=0;
        /*устанавливаем курсор на начало файла*/
        rewind(f);
        /*читаем первое число*/
        fread(&buf, sizeof(double), 1, f);
        /*пока не достигли конца файла, ищем минимум или максимум*/
        while(fread(&num, sizeof(double), 1, f))
            if (num*p>buf*p)
            {
                /*запоминаем его позицию и значение*/
                pos = ftell(f)-sizeof(double);
                buf = num;
            };
        /*возвращаем найденную позицию*/
        return pos;
    }
    /*ф-ия вывода содержимого файла на экран*/
    void printf_file(FILE*f)
    {
        double num;
        /*устанавливаем курсор на начало файла*/
        rewind(f);
        /*читаем и выводим по одному числу,
        пока не достигли конца файла*/
        while(fread(&num, sizeof(double), 1, f))
            printf("%.2lf ", num);
        printf("\n");
    }
}

```

Задача 4.

Создать информационно-справочной системы на базе бинарного файла записей со следующими возможностями: создание файла, просмотр содержимого файла, добавление, удаление и корректировка данных, а также выполнение запросов в соответствии с заданием.

Поиск требуемых данных осуществлять по ключевому полю. Для организации интерфейса должно использоваться меню.

В файле содержатся сведения о спортсменах: фамилия, пол, вид спорта, год рождения, рост. Найти самого высокого спортсмена, занимающегося плаванием, среди мужчин. Вывести сведения о спортсменках, выступающих в юниорском разряде (14-17 лет).

Уровень сложности – повышенный. Имя входного файла должно передаваться программе при ее запуске (через параметры функции `main()`). Если параметры пользователем при запуске программы не заданы, имя файла вводится с клавиатуры. Вывод содержимого файла осуществлять постранично в табличном виде с графлением визуальными подходящими символами, предусмотреть возможность «листания» страниц как в прямом, так и в обратном направлении.

Исходные данные:

argc – кол-во аргументов, переданных в `main`, тип *int*;

files – указатель на массив строк (аргументов, в нашем случае имена файлов), переданных в `main`, тип *char*[]*;

filename – имя файла, введенное пользователем, массив символов типа *char*.

Результирующие данные:

вывод содержимого файла.

Вспомогательные переменные:

menu – пункт меню, выбранный пользователем, тип *int*.

Вспомогательные алгоритмы:

1) Алгоритм считывания данных из файла.

Входные данные: имя файла, кол-во считанных записей за раз, адрес по которому записать кол-во записей в файле.

Результирующие данные: записи прочитаны из файла.

Этот алгоритм описывается в программе функцией

struct sportperson readmany(char*, int, int*)*

Параметры:

первый параметр – адрес начала строки;

второй параметр – кол-во записей, считанных за раз;

третий параметр – адрес, по которому записать кол-во записей.

Возвращаемое значение – адрес первого эл-та массива структур, NULL - в случае ошибок.

Вспомогательные переменные:

k – общее кол-во считанных записей, тип *int*;

i – кол-во считанных записей за раз, тип *int*;
size – размер типа *struct sportperson* в байтах, тип *int*;
buf – динамический массив структур, тип *struct sportperson**;
f – указатель на поток, тип *FILE**.

- 2) Алгоритм поиска самого высокого мужчины, занимающегося плаванием и вывод на экран.

Входные данные: имя файла.

Результирующие данные: отсутствуют.

Этот алгоритм описывается в программе функцией

void findman(char)*

Параметры:

первый параметр – адрес начала строки.

Возвращаемое значение – отсутствует.

Вспомогательные переменные:

buf – динамический массив структур, тип *struct sportperson**;

entry – результат поиска, тип *struct sportperson*;

i – индекс массива *buf*, тип *int*;

k2 – флаг, сообщающий найдена ли запись, тип *int*;

k – размер массива *buf*, тип *int*;

- 3) Алгоритм вывода на экран структуры в виде визитки.

Входные данные: адрес структуры.

Результирующие данные: отсутствуют.

Этот алгоритм описывается в программе функцией

*void print1(struct sportperson*buf)*

Параметры:

первый параметр – адрес структуры;

Возвращаемое значение – отсутствует.

- 4) Алгоритм поиска девушек в юниорском разряде (14-17 лет) и вывода на экран в табличном виде.

Входные данные: имя файла.

Результирующие данные: отсутствуют.

Этот алгоритм описывается в программе функцией

void findwomen(char)*

Параметры:

первый параметр – адрес начала строки.

Возвращаемое значение – отсутствует.

Вспомогательные переменные:

buf – динамический массив структур, тип *struct sportperson**;

i, j – индексы массива *buf*, *j* – для удаления неподходящих, тип *int*;

k – размер массива *buf*, тип *int*;

5) Алгоритм ввода структуры.

Входные данные: отсутствуют.

Результирующие данные: отсутствуют.

Этот алгоритм описывается в программе функцией

struct sportperson input_sportsman (void)

Параметры: отсутствуют.

Возвращаемое значение – введенная структура.

Вспомогательные переменные:

sportsman – вводимая структура, тип *struct sportperson*;

buf – массив символов типа *char* размером 10 для ввода пола;

6) Алгоритм редактирования записи.

Входные данные: имя файла.

Результирующие данные: запись изменена, если пользователь подтвердил редактирование.

Этот алгоритм описывается в программе функцией

void edit (char filename)*

Параметры: адрес начала строки.

Возвращаемое значение - отсутствует.

Вспомогательные переменные:

buf – динамический массив структур, тип *struct sportperson**;

f – указатель на поток, тип *FILE**;

k – размер массива *buf*, тип *int*;

num – номер записи, которую нужно удалить, тип *int*;

str – массив символов типа *char* размером 5 для ввода подтверждения редактирования.

7) Алгоритм добавления записи в файл.

Входные данные: имя файла.

Результирующие данные: запись добавлена.

Этот алгоритм описывается в программе функцией

void add(char)*

Параметры:

первый параметр – адрес начала строки.

Возвращаемое значение – отсутствует.

Вспомогательные переменные:

sportsman – добавляемая структура, тип *struct sportperson*;

f – указатель на поток, тип *FILE**.

- 8) Алгоритм вывода массива структур на экран в табличном виде.

Входные данные: имя массива, размер массива.

Результирующие данные: отсутствуют.

Этот алгоритм описывается в программе функцией

void print_entries(struct sportperson, int)*

Параметры:

первый параметр – адрес первого эл-та массива;

второй параметр – размер массива.

Возвращаемое значение – отсутствует.

Вспомогательные переменные:

n – считанное число, в зависимости от которого выводятся предыдущие записи - 1, следующие - 2, либо прекращается вывод - 3, тип *int*;

i, j – индексы переданного массива, *j* – для фиксированного вывода по 10 записей, тип *int*.

- 9) Алгоритм вывода содержимого файла на экран в табличном виде.

Входные данные: имя файла.

Результирующие данные: отсутствуют.

Этот алгоритм описывается в программе функцией

void show(char filename)*

Параметры:

первый параметр – адрес начала строки.

Возвращаемое значение – отсутствует.

Вспомогательные переменные:

buf – динамический массив структур, тип *struct sportperson**;

k – размер массива *buf*, тип *int*.

- 10) Алгоритм удаления записи.

Входные данные: имя файла.

Результирующие данные: запись удалена.

Этот алгоритм описывается в программе функцией

void delete_entry(char filename)*

Параметры:

первый параметр – адрес начала строки.

Возвращаемое значение – отсутствует.

Вспомогательные переменные:

f – указатель на поток, тип *FILE**;

buf – динамический массив структур, тип *struct sportperson**;

k – размер массива *buf*, тип *int*;

num – номер записи к удалению, тип *int*;

i – индекс массива *buf* для удаления записей, тип *int*.

- 11) Алгоритм перевода буквы в верхний регистр и описывающая его функция те же, что и в первой задаче.
- 12) Алгоритм перевода буквы в нижний регистр.

Входные данные: символ.

Результирующие данные: отсутствуют.

Этот алгоритм описывается в программе функцией

char tolow(char)

Параметры:

первый параметр – символ.

Возвращаемое значение – символ.

- 13) Алгоритм перевода строки в нижний регистр.

Входные данные: имя строки.

Результирующие данные: строка переведена в нижний регистр.

Этот алгоритм описывается в программе функцией

void tolowerstring(char)*

Параметры:

первый параметр – адрес начала строки.

Возвращаемое значение – отсутствует.

Текст программы

```
#include <stdio.h>
#include <locale.h>
#include <stdlib.h>
#include <string.h>
/*объявление ф-ий*/
char toupper(char);
char tolower(char);
void tolowerstring(char*);
void add (char*);
void show (char*);
struct sportperson* readmany(char*, int, int*);
```

```

void delete_entry(char*);
void findman(char*);
void findwomen(char*);
void print_entries(struct sportperson*, int);
void printl(struct sportperson*);
void edit (char*);

/*объявление структурного типа*/
struct sportperson
{
    char name[79];
    char gender;
    char sport[32];
    int year, height;
};

int main(int argc, char* files[])
{
    /*поддержка русского языка*/
    setlocale(LC_ALL, "cp1251");
    system("chcp 1251");
    system ("CLS");
    /*объявление переменных*/
    char filename[50];
    int menu;
    if(argc<2)
    {
        puts ("Имя файла:");
        gets (filename);
    }
    else
        strcpy(filename,files[1]);
    do
    {
        system ("CLS");
        puts ("1. Добавление записи");
        puts ("2. Просмотр записей");
        puts ("3. Удаление записи");
        puts ("4. Редактировать запись");
        puts ("5. Найти мужчину");
        puts ("6. Найти женщин");
        puts ("7. Выход");
        scanf ("%d%c", &menu);
        switch (menu)
        {
            case 1 : add (filename); break;
            case 2 : show (filename); break;
            case 3 : delete_entry (filename); break;
            case 4 : edit (filename); break;
            case 5 : findman(filename); break;
            case 6 : findwomen(filename); break;
        }
    } while (menu!=7);
    return 0;
}

/*Ф-ия чтения из бинарного файла*/
struct sportperson* readmany(char*filename, int n, int* c)
{
    int k=0, i=0, size = sizeof(struct sportperson);
    struct sportperson *buf = calloc(1,1);
    FILE* f;
    system("cls");
    if(!(f=fopen(filename, "rb")))

```

```

    {
        perror("Не удалось открыть файл.");
        getchar();
        return NULL;
    };
do
{
    /*добавляем память под структуры*/
    buf = realloc(buf, (n+i)*size);
    /*читаем n записей*/
    i=fread(buf+k, size, n, f);
    /*фиксируем общее кол-во записей*/
    k+=i;
} while(i==n);
fclose(f);
if(!k)
{
    perror("Файл пуст");
    getchar();
    return NULL;
};
/*присваиваем кол-во записей по переданному адресу*/
*c=k;
/*перевыделяем память, чтобы не занимать лишнюю память*/
buf = realloc(buf, k*size+1);
return buf;
}

/*Ф-ия нахождения самого высокого мужчины, занимающегося плаванием*/
void findman(char* filename)
{
    struct sportperson *buf, entry;
    int k, i, k2=0;
    /*читаем весь файл в buf*/
    if(!(buf = readmany(filename, 10, &k)))
        return;
    /*находим первый подходящий эл-нт для будущего сравнения*/
    for(i=0; i<k; i++)
        if(buf[i].gender == 'M' && strcmp(buf[i].sport, "плавание")==0)
        {
            /*фиксируем что был найден*/
            k2=1;
            entry = buf[i];
            break;
        };
    /*если был найден, то ищем самого высокого*/
    if(k2)
    {
        for(; i<k; i++)
            if(buf[i].gender == 'M' && buf[i].height > entry.height &&
                (strcmp(buf[i].sport, "плавание")==0))
                entry = buf[i];
        printl(&entry);
    }
    else
        printf("Не найдено");
    free(buf);
    getchar();
}

/*Ф-ия вывода структуры в виде визитки*/
void printl(struct sportperson*buf)
{
    printf("ФИО:%s\nПол:%c\nВид спорта:%s\nГод:%d\nРост:%d\n",

```

```

        buf->name, buf->gender, buf->sport, buf->year, buf->height);
    }

/*Ф-ия поиска девушек в возрасте от 14 до 17*/
void findwomen(char* filename)
{
    struct sportperson *buf;
    int k, i, j;
    /*читаем весь файл в buf*/
    if(!(buf = readmany(filename, 10, &k)))
        return;
    /*проходим и удаляем неподходящие записи*/
    for(i=0; i<k; i++)
        if(buf[i].gender!= 'Ж' || (2020 - buf[i].year<14) || (2020 -
        buf[i].year>17))
        {
            for (j=i; j<k-1; j++)
                buf[j] = buf[j+1];
            k--;
            i--;
        };
    /*выводим, если что то осталось*/
    if(k)
        print_entries(buf, k);
    else
    {
        printf("Не найдено");
        getchar();
    };
    free(buf);
}

/*Ф-ия ввода структуры*/
struct sportperson input_sportsman (void)
{
    struct sportperson sportsman;
    char buf[10];
    system ("cls");
    puts ("ФИО");
    gets(sportsman.name);
    do
    {
        puts ("Пол(М/Ж)");
        gets(buf);
        buf[0]=toupper(buf[0]);
        if (buf[0]=='M' || buf[0]=='Ж')
            break;
        else
            puts("Неправильный ввод. Введите \"М\" или \"Ж\".");
    } while(1);
    sportsman.gender = buf[0];
    puts ("Вид спорта");
    gets (sportsman.sport);
    tolowerstring(sportsman.sport);
    puts ("Год рождения");
    scanf ("%d", &sportsman.year);
    puts ("Рост");
    scanf ("%d", &sportsman.height);
    return sportsman;
}

/*Ф-ия редактирования записи по номеру*/
void edit (char* filename)
{
    FILE * f;

```

```

int num,k;
struct sportperson *buf;
char str[5];
if(!(buf = readmany(filename, 10, &k)))
    return;
system ("cls");
puts("Введите номер записи");
scanf("%d", &num);
getchar();
if(num<1 || num>k)
{
    system ("cls");
    printf("Неправильный ввод");
    free(buf);
    getchar();
    return;
};
printf("%d", num);
puts("Редактировать (Y/N)");
gets(str);
str[0] = toupper(str[0]);
if (str[0] == 'Y' || str[0]=='N')
{
    if(str[0] == 'Y')
    {
        buf[num-1] = input_sportsman();
        f = fopen(filename, "wb");
        fwrite(buf, sizeof(struct sportperson), k, f);
        fclose(f);
    }
    else
        return;
}
else
    printf("Неправильный ввод");
free(buf);
getchar();
}
/*Ф-ия добавления записи в конец файла*/
void add(char* filename)
{
    FILE*f;
    struct sportperson sportsman;
    system ("cls");
    if(!(f=fopen(filename, "ab")))
    {
        perror ("Ошибка открытия файла");
        getchar();
        return;
    };
    sportsman = input_sportsman();
    fwrite(&sportsman, sizeof(sportsman), 1, f);
    fclose(f);
    getchar();
}
/*Ф-ия вывода записей по страницам*/
void print_entries(struct sportperson*buf, int k)
{
    int i=0,j,num;
    if (!buf)
        return;
    while (1)
    {

```

```

        system ("cls");
        puts("| N |                               ФИО                               | Пол | Вид спорта |
Год | Рост |");
        puts("-----");
        /*вывод по десять записей*/
        for(j=0; j<10&& i<k; j++,i++)
        {
            printf("|%3d|%-31.31s|%-5c|%-12.12s|%5d|%6d|\n",
                i+1, buf[i].name, buf[i].gender, buf[i].sport, buf[i].year,
                buf[i].height);
            puts("-----");
        };
        puts("1. Вывести предыдущие 10 записей");
        puts("2. Вывести следующие 10 записей");
        puts("3. Прекратить вывод");
        scanf("%d%c", &num);
        system ("cls");
        switch (num)
        {
            case 1 :
                if(i<11)
                {
                    puts("Предыдущих записей нет.");
                    getchar();
                    return;
                }
                else
                    i-=10+j;
                break;
            case 2 :
                if(i>=k)
                {
                    puts("Следующих записей нет.");
                    getchar();
                    return;
                }
                break;
            case 3 : return; break;
        };
    };
}

void show(char* filename)
{
    struct sportperson *buf;
    int k;
    if(!(buf = readmany(filename, 10, &k)))
        return;
    print_entries(buf, k);
    free(buf);
}

/*Ф-ия удаления записи по ее номеру*/
void delete_entry(char* filename)
{
    system ("cls");
    int num;
    FILE* f;
    struct sportperson *buf;
    int k, i;
    if(!(buf = readmany(filename, 10, &k)))
        return;
    puts("Введите номер записи");

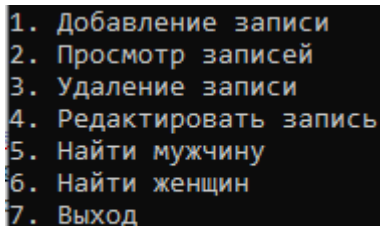
```

```

scanf("%d", &num);
getchar();
if(num<1 || num>k)
{
    system("cls");
    printf("Неправильный ввод");
    free(buf);
    getchar();
    return;
};
if (!(f=fopen(filename, "wb")))
{
    perror("Не удалось открыть файл.");
    free(buf);
    getchar();
    return;
};
for (i=num-1; i<k-1; i++)
    buf[i] = buf[i+1];
fwrite(buf, sizeof(struct sportperson), k-1, f);
fclose(f);
free(buf);
}
/*Ф-ия повышения регистра*/
char toupper(char c)
{
    if ((c>='a' && c<='z') || (c>='a' && c<='я'))
        c-=32;
    return c;
}
/*Ф-ия понижения регистра*/
char tolower(char c)
{
    if ((c>='A' && c<='Z') || (c>='A' && c<='Я'))
        c+=32;
    return c;
}
/*Ф-ия понижения регистра строки*/
void tolowerstring(char *a)
{
    int i;
    for (i=0; a[i]!='\0'; i++)
        a[i]=tolower(a[i]);
}

```

Пример работы программы



```

1. Добавление записи
2. Просмотр записей
3. Удаление записи
4. Редактировать запись
5. Найти мужчину
6. Найти женщин
7. Выход

```

2. Просмотр записей

N	ФИО	Пол	Вид спорта	Год	Рост
1	Живова	Ж	плавание	2002	177
2	Сидорова	Ж	плавание	2003	170
3	Иванова	Ж	бокс	2004	180
4	Машкина	Ж	футбол	2005	170
5	Рябцева	Ж	волейбол	2006	180
6	Галкина	Ж	баскетбол	2007	150
7	Альков	М	плавание	2001	170
8	Кравченко	М	баскетбол	2002	180
9	Пискурев	М	плавание	2002	177
10	Уманец	М	волейбол	2001	180

1. Вывести предыдущие 10 записей
2. Вывести следующие 10 записей
3. Прекратить вывод

Вывести предыдущие 10 записей

Предыдущих записей нет.

Вывести следующие 10 записей

N	ФИО	Пол	Вид спорта	Год	Рост
11	Михайлов	М	плавание	2002	181
12	Минкин	М	бокс	2002	178

1. Вывести предыдущие 10 записей
2. Вывести следующие 10 записей
3. Прекратить вывод

Вывести следующие 10 записей

Следующих записей нет.

1.Добавление записи

ФИО	Иванов Иван Иванович
Пол (М/Ж)	М
Вид спорта	бокс
Год рождения	2002
Рост	180

N	ФИО	Пол	Вид спорта	Год	Рост
11	Михайлов	М	плавание	2002	181
12	Минкин	М	бокс	2002	178
13	Иванов Иван Иванович	М	бокс	2002	180

После удаления первой записи

N	ФИО	Пол	Вид спорта	Год	Рост
1	Сидорова	Ж	плавание	2003	170
2	Иванова	Ж	бокс	2004	180
3	Машкина	Ж	футбол	2005	170
4	Рябцева	Ж	волейбол	2006	180
5	Галкина	Ж	баскетбол	2007	150
6	Альков	М	плавание	2001	170
7	Кравченко	М	баскетбол	2002	180
8	Пискурев	М	плавание	2002	177
9	Уманец	М	волейбол	2001	180
10	Михайлов	М	плавание	2002	181

1. Вывести предыдущие 10 записей
2. Вывести следующие 10 записей
3. Прекратить вывод

Редактировать запись

Введите номер записи	ФИО
7	Полежаикин
ФИО:Кравченко	Пол(М/Ж)
Пол:М	М
Вид спорта:баскетбол	Вид спорта
Год:2002	танцы
Рост:180	Год рождения
Редактировать(Y/N)	1999
у	Рост
	177

N	ФИО	Пол	Вид спорта	Год	Рост
1	Сидорова	Ж	плавание	2003	170
2	Иванова	Ж	бокс	2004	180
3	Машкина	Ж	футбол	2005	170
4	Рябцева	Ж	волейбол	2006	180
5	Галкина	Ж	баскетбол	2007	150
6	Альков	М	плавание	2001	170
7	Полежаикин	М	танцы	1999	177
8	Пискурев	М	плавание	2002	177
9	Уманец	М	волейбол	2001	180
10	Михайлов	М	плавание	2002	181

1. Вывести предыдущие 10 записей
2. Вывести следующие 10 записей
3. Прекратить вывод

Найти самого высокого мужчину среди пловцов

ФИО:Михайлов
Пол:М
Вид спорта:плавание
Год:2002
Рост:181

Найти девушек от 14 до 17

N	ФИО	Пол	Вид спорта	Год	Рост
1	Сидорова	Ж	плавание	2003	170
2	Иванова	Ж	бокс	2004	180
3	Машкина	Ж	футбол	2005	170
4	Рябцева	Ж	волейбол	2006	180

1. Вывести предыдущие 10 записей
2. Вывести следующие 10 записей
3. Прекратить вывод