Лекция 19.11.2021

Файлы прямого доступа и бинарные файлы

Особенности файла в Си – бестиповый.

Единственный признак – свойство потока: текстовый или двоичный.

Тип файла указывается при открытии (создании) с помощью fopen() добавлением к способу открытия буквы ‘t’ для текстовых файлов и ‘b’ для бинарных.

```
FILE *input, *output;

input = fopen(“in.txt”, “r+t”);

output = fopen(“out.txt”, “wb”);
```

Открытие/закрытие файлов прямого доступа выполняется теми же функциями, что и для файлов последовательного доступа.

Функции ввода \ вывода

Запись и чтение в бинарный файл осуществляется с помощью функций fread() и fwrite().

size_t fread (void *ptr, size_t size, size_t n, FILE *stream);

Читает из файла ***stream*** в массив ***ptr*** ***n*** объектов-записей размера ***size***.

Возвращает количество действительно прочитанных блоков заданного размера (результат может быть отличен от ***n***).

Состояние потока после завершения операции чтения необходимо проверять другими функциями – feof() и ferror().

size_t fwrite(const void *ptr, size_t size, size_t n, FILE *stream);

Записывает в файл ***stream*** из массива ***ptr n*** объектов-записей размера ***size***.

Возвращает количество действительно записанных блоков заданного размера (результат может быть отличен от ***n***)

Пример 78. Запись несимвольных данных в файл и последующее их чтение.

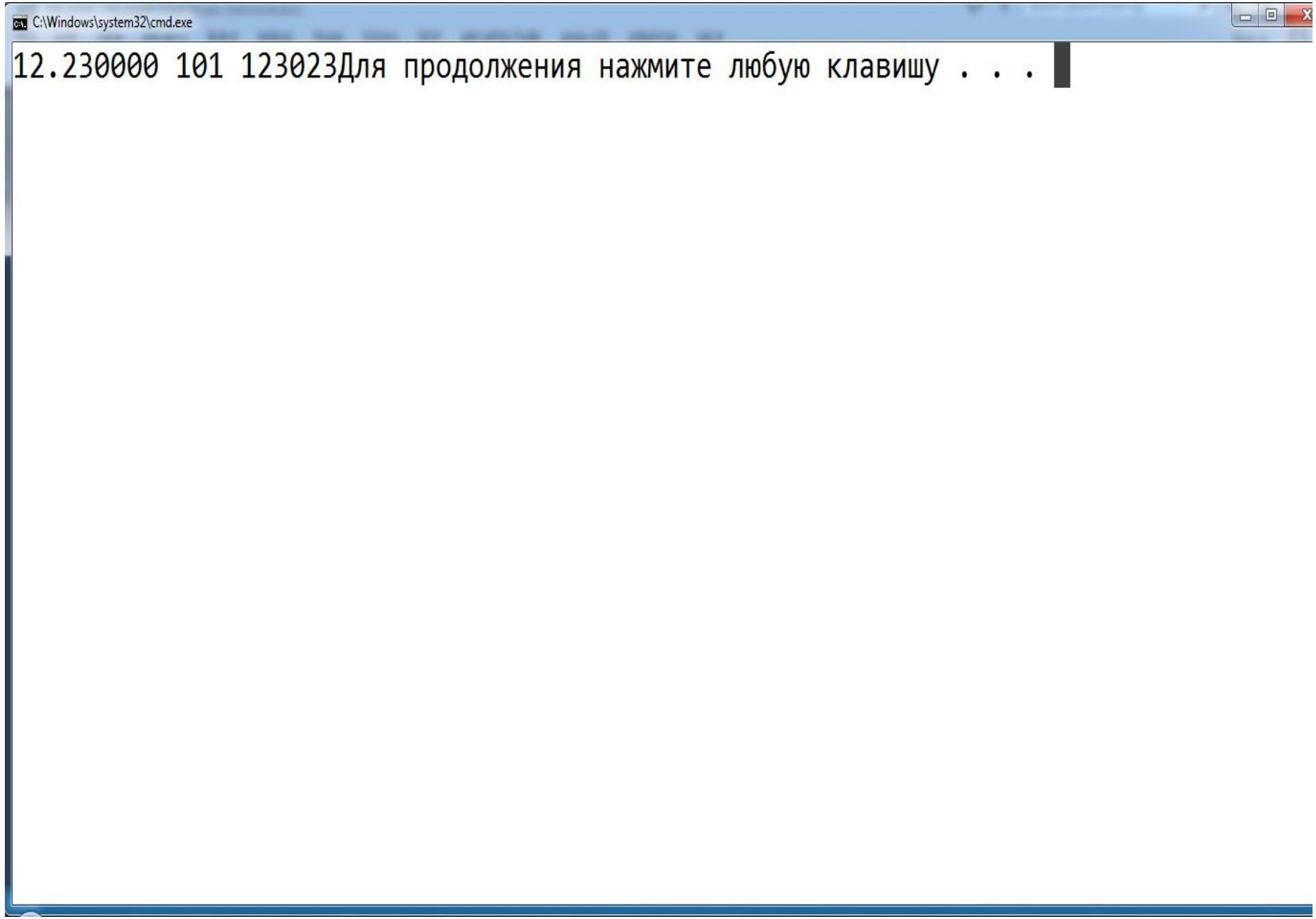
```
#include <stdio.h>
#include <stdlib.h>
int main()
{
        FILE *fp;
        double d = 12.23;
        int i = 101;
        long l = 123023L;

        if((fp=fopen("test","wb+"))==NULL)
        {
        printf("Ошибка при открытии файла.\n");
            exit(-1);
    }
```

```
    fwrite(&d, sizeof(double), 1, fp);
    fwrite(&i, sizeof(int), 1, fp);
    fwrite(&l, sizeof(long), 1, fp);

rewind(fp); // fp в начальное положение

    fread(&d, sizeof(double), 1, fp);
    fread(&i, sizeof(int), 1, fp);
    fread(&l, sizeof(long), 1, fp);
printf("%f %d %ld", d, i, l);
fclose(fp);
    return 0;
}
```


C:\Windows\system32\cmd.exe
12.230000 101 123023Для продолжения нажмите любую клавишу . . .

void rewind(FILE *stream);

Устанавливает указатель в начало файла.

int ftell(FILE *stream);

Возвращает текущую позицию stream. При ошибке возвращает –1L.

fseek() и произвольный доступ

Можно выполнять операции произвольного чтения и записи с помощью fseek(), устанавливающей текущую файловую позицию.

int fseek(FILE *fp, long offset, int wherefrom);

fp – это указатель на файл, возвращенный fopen().

Аргументы ***offset*** и ***wherefrom*** зависят от того, текстовый файл или двоичный.

Для двоичного файла:

offset – число байтов смещения от ***точки***, определяемой ***wherefrom***:

- SEEK_SET – начало файла;
- SEEK_CUR – текущая позиция;
- SEEK_END – конец файл.

Двоичные файлы можно читать и в обратном порядке.

Для текстового файла:

offset – может быть либо =0, либо результату, который возвращает функция ftell.

wherefrom – SEEK_SET.

// Открыть существующий как двоичный для чтения и записи

```
FILE *fd;
fd = fopen("a.dat","rb+wb");
```

// Создать новый как двоичный для записи и чтения

```
fd = fopen("a.dat","wb+");
```

Пример 79. Получение клиента из списка по номеру.

```
struct addr
{
        char name[40], street[40], city[40];
        char state[3];
    char zip[10];
} info;
```

```
void find(long int client_num)
{
        FILE *fp;
        if((fp=fopen("mail", "rb")) == NULL)
        {
            printf("Не удается открыть файл.\n");
            exit(1);
        }

    fseek(fp, client_num*sizeof(struct addr),
                                        SEEK_SET);

    fread(&info, sizeof(struct addr), 1, fp);

    fclose(fp);
}
```

Функции обработки ошибок

int ferror(FILE *stream);

Возвращает 0, если нет ошибок, и число, отличное от 0, если ошибки есть.

Вызывается после функции, которая может вызвать ошибку, например, fread() и fwrite().

void clearerr(FILE *stream);

Очищает состояние признака ошибки в ***stream.***

int feof(FILE *stream);

Возвращает EOF если конец файла, иначе 0.

Особенности обработки файлов прямого доступа

Последовательный файл считывают с помощью цикла с неизвестным числом повторений до наступления условия «прочитан конец файла».

У файлов прямого доступа следует использовать цикл с известным числом повторений для обработки такого файла.

Можно получить размер файла:

1) С помощью fseek установить текущую позицию на конец файла:

```
fseek(stream, 0L, SEEK_END);
```

2) Запросить значение текущего указателя (размер файла) с помощью ftell:

```
size = ftell(stream);
```

Далее в цикле позиционировать текущую позицию файла с помощью fseek, пересчитывая каждый раз величину смещения – второй аргумент (позиционировать от начала).

Пример сортировки

```
struct Record
{                       //структуры данных
    char country[20];   //Название страны
    float el55;         //Произведено
электроэнергии в 1955
    float el58;         //в 1958
} a, b;

int cmp(const void * a, const void * b) {
    return strcmp(((Record*)a)->country,
    ((Record*)b)->country);
```

```
 int i=0, j, pos,n;
        if ((fv = fopen(name,"rb+")) == NULL)
//Открытие файла для чтения
        {
                printf("Error open file!\n");
                exit(-1);
        }
        fseek(fv,0,SEEK_END);
        n = ftell(fv)/sizeof(a);                //Определение
количества записей в файле
```

```
fseek(fv,0,SEEK_SET);

        for (i=0; i<n; i++) {
            fread(&rec[i],sizeof(Record),1,fv);
            puts(rec[i].country);
        }
        qsort(rec,n,sizeof(Record),cmp());
        fseek(fv,0,SEEK_SET);
        for (i=0; i<n; i++) {
            puts(rec[i].country);
            fwrite(&rec[i],sizeof(Record),1,fv);
        }
    free(rec);
        fclose(fv);
```

void **qsort** (void * first, size_t number, size_t size, int (* comparator)

first указатель на 1й элемент сортируемого массива.
number количество элементов в сортируемом массиве, на который ссылается указатель first.
Size размер одного элемента массива в байтах.
Comparator Функция, которая сравнивает два элемента.