📄 Шпаргалка по работе с файлами в C

📌 1. Подключение библиотеки

```
c
#include <stdio.h>
```

📌 2. Объявление файловой переменной

```
c
FILE *fp;  // fp – file pointer (указатель на файл)
```

📌 3. Открытие файла: fopen()

```
c
fp = fopen("имя_файла.расширение", "режим");
```

Режимы открытия:

Режим	Описание	Если файл существует	Если файл не существует
"r"	Чтение (текстовый)	✓ Открывается	✕ Ошибка
"w"	Запись (текстовый)	✕ Перезаписывается	✓ Создаётся
"a"	Добавление (текстовый)	✓ Добавление в конец	✓ Создаётся

`"rb"`	Чтение (бинарный)	✓ Открывается	✗ Ошибка
`"wb"`	Запись (бинарный)	✗ Перезаписывается	✓ Создаётся
`"ab"`	Добавление (бинарный)	✓ Добавление в конец	✓ Создаётся
`"r+"`	Чтение/запись (текстовый)	✓ Открывается	✗ Ошибка
`"w+"`	Чтение/запись (текстовый)	✗ Перезаписывается	✓ Создаётся
`"a+"`	Чтение/добавление (текстовый)	✓ Открывается	✓ Создаётся

Проверка успешного открытия:

```
c
if (fp == NULL) {
    printf("Ошибка открытия файла!\n");
    perror("fopen");  // Детали ошибки
    return 1;
}
```

📌 4. Закрытие файла: fclose()

```
c
fclose(fp);  // Всегда закрывайте файлы!
```

📌 5. Работа с ТЕКСТОВЫМИ файлами

Запись в файл:

Функция	Формат	Пример
`fputc()`	1 символ	`fputc('A', fp);`
`fputs()`	Строка	`fputs("Привет\n", fp);`
`fprintf()`	Форматированно	`fprintf(fp, "Число: %d\n", x);`

```
c
// Пример записи
fprintf(fp, "Имя: %s, Возраст: %d\n", name, age);
```

Чтение из файла:

Функция	Читает	Возвращает
`fgetc()`	1 символ	символ или EOF
`fgets()`	Строку	указатель или NULL
`fscanf()`	Форматированно	кол-во прочитанных

```
c
// Чтение строки
char buffer[100];
while (fgets(buffer, sizeof(buffer), fp) != NULL) {
    printf("%s", buffer);
}

// Форматированное чтение
int num;
char str[50];
fscanf(fp, "%s %d", str, &num);
```

Построчное чтение (шаблон):

```
c
char line[256];
while (fgets(line, sizeof(line), fp)) {
    // обработать line
}
```

Посимвольное чтение (шаблон):

```
c
int ch;
while ((ch = fgetc(fp)) != EOF) {
    putchar(ch);
}
```

📌 6. Работа с БИНАРНЫМИ файлами

Структура для примера:

```
c
struct Student {
    char name[50];
    int age;
    float gpa;
};
```

Запись: fwrite()

```
c
size_t fwrite(const void *ptr, size_t size, size_t count,
FILE *fp);
// ptr    - указатель на данные
// size   - размер одного элемента (sizeof)
// count  - сколько элементов
// возвращает: сколько записано на самом деле
```

Примеры:

```
c
struct Student s = {"Иванов", 20, 4.5};

// Запись одной структуры
fwrite(&s, sizeof(struct Student), 1, fp);

// Запись массива структур
struct Student group[10];
fwrite(group, sizeof(struct Student), 10, fp);
```

```
// Запись числа
int x = 42;
fwrite(&x, sizeof(int), 1, fp);
```

Чтение: fread()

```
c
size_t fread(void *ptr, size_t size, size_t count, FILE
*fp);
// возвращает: сколько прочитано на самом деле
```

Примеры:

```
c
struct Student s;

// Чтение одной структуры
while (fread(&s, sizeof(struct Student), 1, fp) == 1) {
    printf("%s: %d лет\n", s.name, s.age);
}

// Чтение массива
struct Student group[10];
int count = fread(group, sizeof(struct Student), 10, fp);

// Проверка конца файла
if (feof(fp)) {
    printf("Достигнут конец файла\n");
}
```

📌 7. Навигация по файлу

fseek() - перемещение по файлу

```
c
int fseek(FILE *fp, long offset, int whence);
// offset - смещение в байтах
// whence: SEEK_SET (начало), SEEK_CUR (текущая),
SEEK_END (конец)
```

Примеры:

```
c
fseek(fp, 0, SEEK_SET);    // В начало
fseek(fp, 10, SEEK_CUR);   // На 10 байт вперед от
текущей
fseek(fp, -20, SEEK_END);  // За 20 байт до конца
fseek(fp, sizeof(struct Student) * 5, SEEK_SET); // К 6-й
```

структуре

ftell() - текущая позиция

```
c
long pos = ftell(fp);  // позиция от начала в байтах
```

rewind() - в начало файла

```
c
rewind(fp);  // Эквивалентно fseek(fp, 0, SEEK_SET);
```

📌 8. Проверка состояния файла

Функция	Возвращает	Описание
`feof(fp)`	Ненулевое	Если конец файла
`ferror(fp)`	Ненулевое	Если ошибка
`clearerr(fp)`	—	Сброс ошибок

```
c
// Шаблон чтения с проверкой
while (1) {
    struct Student s;
    size_t result = fread(&s, sizeof(s), 1, fp);

    if (result == 1) {
        // успешно прочитано
    } else if (feof(fp)) {
        break;  // конец файла
    } else if (ferror(fp)) {
        perror("Ошибка чтения");
        break;
    }
}
```

📌 9. Полезные функции

Удаление файла:

```
c
```

```
remove("file.txt");
```

Переименование файла:

```
c
rename("old.txt", "new.txt");
```

Вывод ошибки:

```
c
perror("Сообщение");  // Выводит: Сообщение: текст ошибки
// Пример: perror("fopen");
```

📌 10. Шаблоны программ

Шаблон 1: Копирование текстового файла

```
c
#include <stdio.h>

int main() {
    FILE *src, *dest;
    char ch;

    src = fopen("input.txt", "r");
    dest = fopen("output.txt", "w");

    if (!src || !dest) {
        printf("Ошибка открытия файлов\n");
        return 1;
    }

    while ((ch = fgetc(src)) != EOF) {
        fputc(ch, dest);
    }

    fclose(src);
    fclose(dest);
    return 0;
}
```

Шаблон 2: Чтение/запись структуры в бинарный файл

```
c
#include <stdio.h>

struct Data {
```

```
    int id;
    float value;
    char name[30];
};

int main() {
    struct Data d = {1, 3.14, "Пример"};
    FILE *fp;

    // Запись
    fp = fopen("data.dat", "wb");
    fwrite(&d, sizeof(struct Data), 1, fp);
    fclose(fp);

    // Чтение
    fp = fopen("data.dat", "rb");
    struct Data read_data;
    fread(&read_data, sizeof(struct Data), 1, fp);

    printf("%d %.2f %s\n", read_data.id, read_data.value,
read_data.name);
    fclose(fp);

    return 0;
}
```

Шаблон 3: Добавление записей в конец файла

```
c
struct Record {
    char info[100];
    int number;
};

void addRecord() {
    FILE *fp = fopen("records.dat", "ab");  // режим
добавления
    struct Record rec;

    printf("Введите информацию: ");
    fgets(rec.info, sizeof(rec.info), stdin);
    printf("Введите число: ");
    scanf("%d", &rec.number);

    fwrite(&rec, sizeof(struct Record), 1, fp);
    fclose(fp);
```

```
}
```

Шаблон 4: Поиск записи в бинарном файле

```c
int findStudentById(int id) {
    FILE *fp = fopen("students.dat", "rb");
    struct Student s;

    while (fread(&s, sizeof(struct Student), 1, fp)) {
        if (s.id == id) {
            printf("Найден: %s\n", s.name);
            fclose(fp);
            return 1;  // нашли
        }
    }

    fclose(fp);
    printf("Не найден\n");
    return 0;  // не нашли
}
```

📌 11. Частые ошибки и решения

1. **Не проверяете fopen() на NULL**

```c
// ПЛОХО:
FILE *fp = fopen("file.txt", "r");

// ХОРОШО:
FILE *fp = fopen("file.txt", "r");
if (fp == NULL) {
    perror("Ошибка");
    return 1;
}
```

10. **Забываете fclose()**
    - Всегда закрывайте файлы!
11. **Путаете "r" и "w"**
    - "r" — только чтение (файл должен быть)
    - "w" — запись (создаст или перезапишет)
12. **Для структур используйте бинарный режим**

```c
```

```
    // ПЛОХО для структур:
13  fp = fopen("data.txt", "w");
14
15  // ХОРОШО для структур:
16  fp = fopen("data.dat", "wb");
```

17 Проверяйте возвращаемые значения

```
    c

    // ПЛОХО:
18  fwrite(&data, sizeof(data), 1, fp);
19
20  // ХОРОШО:
21  if (fwrite(&data, sizeof(data), 1, fp) != 1) {
22      printf("Ошибка записи\n");
23  }
```

📌 12. Быстрая памятка

```
c
// ОТКРЫТЬ → ПРОВЕРИТЬ → РАБОТАТЬ → ЗАКРЫТЬ

FILE *fp = fopen("file", "режим");
if (!fp) { perror("Ошибка"); return 1; }

// ... работа с файлом ...

fclose(fp);
```

Что делать	Текстовый файл	Бинарный файл
Записать	`fprintf()`	`fwrite()`
Прочитать	`fscanf()`/ `fgets()`	`fread()`
Добавить	режим `"a"`	режим `"ab"`

Структуры	не рекомендуется	`fwrite(&struct, sizeof, 1, fp)`