8. Работа с файлами: FILE, fopen, fclose, r/w, t/b

- **FILE:**

  - FILE — это структура, определенная в stdio.h, которая представляет поток ввода-вывода (например, открытый файл).
  - Когда вы открываете файл с помощью fopen, вы получаете указатель типа FILE*.
  - Этот указатель используется для всех последующих операций с файлом (чтение, запись, закрытие).

- **fopen:**

  - Функция fopen открывает файл.
  - Синтаксис: FILE* fopen(const char* filename, const char* mode);
  - filename: Имя файла.
  - mode: Режим открытия файла (например, “r”, “w”, “rb”, “wb”).
  - Возвращает:
    - Указатель FILE* на открытый файл в случае успеха.
    - NULL в случае ошибки.
  - **Режимы открытия:**
    - "r": Чтение. Файл должен существовать.
    - "w": Запись. Если файл существует, он будет перезаписан. Если файл не существует, он будет создан.
    - "rb": Чтение в бинарном режиме.
    - "wb": Запись в бинарном режиме.
  - Режим t (text) используется для текстовых файлов (например "rt"), в то время как b используется для бинарных файлов (например "rb"). Однако на многих системах разницы между "r" и "rt" нету.

  fclose:

  - Функция fclose закрывает открытый файл.
  - Синтаксис: int fclose(FILE* file);
  - file: Указатель FILE* на открытый файл.
  - Возвращает:
    - 0 в случае успеха.
    - EOF в случае ошибки.

- Закрытие файла освобождает ресурсы, связанные с ним. Важно всегда закрывать файлы после использования.

- **r/w (чтение/запись):**

- После открытия файла можно использовать разные функции для чтения и записи:
- **Чтение (для текстовых файлов):** fscanf(произвольный тип), fgets(строки), fgetc(char'ы).
- **Запись (для текстовых файлов):** fprintf(произвольный тип), fputs(строки), fputc(char'ы).

2. Стандартные потоки: stdin, stdout, stderr

- **stdin (стандартный поток ввода):**
- По умолчанию представляет ввод с клавиатуры.
- Используется для получения данных от пользователя или другой программы.
- Тип: FILE*
- **stdout (стандартный поток вывода):**
- По умолчанию представляет вывод на экран (или в терминал).
- Используется для вывода результатов работы программы.
- Тип: FILE*
- **stderr (стандартный поток ошибок):**
- По умолчанию представляет вывод на экран (или в терминал).
- Используется для вывода сообщений об ошибках и предупреждений.
- Тип: FILE*
- Обычно не буферизован, что обеспечивает немедленный вывод ошибок.

3. Форматированный ввод/вывод:

- **printf:**
- Функция для форматированного вывода в stdout.
- Синтаксис: int printf(const char* format, ...);
- format: Строка формата (содержит спецификаторы формата).
- ...: Список аргументов, соответствующих спецификаторам формата.
- %d: Целое число (int).
- %f: Число с плавающей точкой (float).
- %lf: Число с плавающей точкой двойной точности (double).

- %s: Строка (char*).
- %c: Символ (char).

- **scanf:**
  - Функция для форматированного ввода из stdin.
  - Синтаксис: int scanf(const char* format, ...);
  - format: Строка формата.
  - ...: Список адресов переменных, в которые нужно прочитать данные.
- **fprintf:**
  - Функция для форматированного вывода в файл.
  - Синтаксис: int fprintf(FILE* file, const char* format, ...);
  - file: Указатель FILE* на открытый файл.
  - Работает аналогично printf.
- **fscanf:**
  - Функция для форматированного ввода из файла.
  - Синтаксис: int fscanf(FILE* file, const char* format, ...);
  - file: Указатель FILE* на открытый файл.
  - Работает аналогично scanf.
- **sprintf:**
  - Функция для форматированного вывода в строку.
  - Синтаксис: int sprintf(char* str, const char* format, ...);
  - str: Указатель на буфер для записи строки.
  - format: Строка формата.
  - ...: Список аргументов.
  - Похоже на printf, но выводит данные в строку.
  - Работает медленно, лучше не использовать
- **sscanf:**
  - Функция для форматированного ввода из строки.
  - Синтаксис: int sscanf(const char* str, const char* format, ...);
  - str: Указатель на строку, из которой нужно читать.
  - format: Строка формата.
  - ...: Список адресов переменных.
  - Похоже на scanf, но читает данные из строки.
  - Работает медленно, лучше не использовать
- **fgets:**
  - Функция для чтения строки из файла.
  - Синтаксис: char* fgets(char* str, int n, FILE* file);
  - str: Указатель на буфер для записи строки.
  - n: Максимальное количество символов для чтения.
  - file: Указатель FILE* на открытый файл.

- Читает строку до символа новой строки \n или пока не будет прочитано n-1 символов.
- Возвращает NULL, если произошла ошибка или достигнут конец файла.

4. Обработка ошибок, feof, ferror

- **Обработка ошибок:**
- После каждой операции ввода-вывода важно проверять, не возникло ли ошибки.
- fopen может вернуть NULL, если файл не удалось открыть.
- fclose, fprintf, fscanf, fgets, fread, fwrite и другие функции могут вернуть значение, отличное от нуля или EOF, в случае ошибки.
- **feof:**
- Функция для проверки, достигнут ли конец файла.
- Синтаксис: int feof(FILE* file);
- file: Указатель FILE* на открытый файл.
- Возвращает ненулевое значение, если достигнут конец файла, и 0 в противном случае.
- Важно помнить, что feof проверяет *флаг* конца файла, который устанавливается *после* попытки чтения за концом файла.
- **ferror:**
- Функция для проверки, возникла ли ошибка при работе с файлом.
- Синтаксис: int ferror(FILE* file);
- file: Указатель FILE* на открытый файл.
- Возвращает ненулевое значение, если возникла ошибка, и 0 в противном случае.