Описание методов

Основные методы программы

1. \*\*main(int argc, char **argv)**
   * Точка входа в программу
   * Проверяет наличие аргументов командной строки
   * Инициализирует флаги, парсит аргументы и обрабатывает входные файлы
   * Возвращает код завершения программы
2. **initialize\_flags(void)**
   * Инициализирует все флаги программы значениями по умолчанию (false)
   * Устанавливает начальное состояние для всех опций программы
3. \*\*parse\_command\_line\_options(int argc, char **argv)**
   * Обрабатывает аргументы командной строки
   * Использует getopt\_long для разбора опций
   * Устанавливает соответствующие флаги в зависимости от полученных опций
   * Обрабатывает конфликты между опциями (например, приоритет -b над -n)
4. \**print\_usage\_error(const char invalid\_option)*
   * Выводит сообщение об ошибке при неверном использовании программы
   * Показывает недопустимую опцию и правильный формат команды
5. \*\*process\_input\_files(int argc, char **argv)**
   * Обрабатывает все входные файлы, указанные в аргументах
   * Открывает каждый файл и передает его на обработку
   * Обрабатывает ошибки открытия файлов
6. \**print\_file\_error(const char filename)*
   * Выводит сообщение об ошибке при невозможности открыть файл
   * Указывает имя проблемного файла
7. \*\*process\_file\_contents(FILE \*file, size\_t *line\_counter, int consecutive\_empty\_lines)*
   * Основная функция обработки содержимого файла
   * Читает файл посимвольно и применяет все установленные флаги
   * Управляет нумерацией строк, сжатием пустых строк и отображением специальных символов

Вспомогательные методы

1. \**handle\_nonprinting\_characters(int character)*
   * Преобразует непечатаемые символы для отображения (флаг -v)
   * Заменяет управляющие символы на читаемую форму (например, ^A для символа 1)
2. \*\*print\_line\_number(size\_t *line\_counter, bool is\_new\_line)*
   * Выводит номер текущей строки с форматированием (для флагов -n и -b)
   * Обновляет счетчик строк и сбрасывает флаг новой строки
3. **print\_tab\_character(void)**
   * Заменяет символ табуляции на "^I" (для флага -t)
4. **print\_end\_of\_line(void)**
   * Добавляет символ '$' в конец строки (для флага -e)
5. \**should\_skip\_repeated\_empty\_lines(int character, int empty\_line\_count)*
   * Проверяет, нужно ли пропустить текущую пустую строку (флаг -s)
   * Следит за количеством подряд идущих пустых строк

Системные методы, используемые в коде

1. \*\*fprintf(FILE *stream, const char format, ...)*
   * Выводит форматированную строку в указанный поток
   * Используется для вывода ошибок в stderr
2. \*\*fopen(const char *filename, const char mode)*
   * Открывает файл для чтения
   * Возвращает FILE\* или NULL при ошибке
3. \**fgetc(FILE stream)*
   * Читает следующий символ из файла
   * Возвращает EOF при достижении конца файла
4. \**fclose(FILE stream)*
   * Закрывает открытый файл
5. **putchar(int char)**
   * Выводит символ в стандартный поток вывода
6. \*\*getopt\_long(int argc, char \*const argv[], const char \*optstring, const struct option *longopts, int longindex)*
   * Парсит аргументы командной строки
   * Возвращает следующую опцию или -1 по завершении

Алгоритм работы программы

1. **Инициализация**
   * Программа запускается с аргументами командной строки
   * Проверяется минимальное количество аргументов
   * Все флаги устанавливаются в значения по умолчанию (false)
2. **Парсинг аргументов командной строки**
   * Обрабатываются все переданные опции (-b, -e, -E, -n, -s, -t, -T, -v)
   * Устанавливаются соответствующие флаги
   * Обрабатываются конфликты между опциями
   * При обнаружении недопустимой опции выводится сообщение об ошибке и программа завершается
3. **Обработка входных файлов**
   * Для каждого файла из аргументов командной строки:
     + Файл открывается для чтения
     + При ошибке открытия выводится сообщение и переходится к следующему файлу
     + Содержимое файла обрабатывается построчно
4. **Обработка содержимого файла**
   * Для каждого символа в файле:
     + Проверяется флаг сжатия пустых строк (-s), при необходимости строки пропускаются
     + При начале новой строки проверяются флаги нумерации (-n, -b) и при необходимости выводится номер строки
     + Обрабатываются табуляции (флаг -t или -T) с заменой на "^I"
     + Обрабатываются непечатаемые символы (флаг -v) с преобразованием в читаемый вид
     + Обрабатываются концы строк (флаг -e или -E) с добавлением '$'
     + Символ выводится на экран
     + Обновляется состояние новой строки
5. **Завершение работы**
   * После обработки всех файлов программа завершается с кодом успеха
   * В случае ошибок на любом этапе программа может завершиться с кодом ошибки