int	ft_isspace(char c);	возвращает 1 если символ - пробельный (9-13 || 32) и 0 в противном случае **
int	ft_isdigit(int c);	возвращает 1 если символ - цифра и 0 в противном случае
int	ft_isalpha(int c);	возвращает 1 если символ - буква и 0 в противном случае
int	ft_isalnum(int c);	возвращает 1 если символ - цифра или буква и 0 в противном случае
int	ft_isprint(int c);	возвращает 1 если символ - печатный и 0 в противном случае
int	ft_isascii(int c);	
int	ft_islower(char c);	*
int	ft_isupper(char c);	*
int	ft_tolower(int c);	
int	ft_toupper(int c);	
int	ft_memcmp(const void *s1, const void *s2, size_t n);	сравнивает n байт участков памяти. возвращает 0 если участки идентичны.
		возвращает 1 если встретились различающиеся байты и *s1 > *s2. иначе -1
void	*ft_memcpy(void *dest, const void *src, size_t n);	копирует n байт из src в dest. области памяти не должны перекрываться
void	*ft_memrcpy(void *dest, const void *src, size_t n);	копирует память, двигаясь от последнего байта к первому **
void	*ft_memmove(void *dest, const void *src, size_t n);	копирует n байт из src в dest. области памяти могут перекрываться
void	*ft_memccpy(void *dest, const void *src, int c, size_t n);	копирует байты (не более n). может скопировать меньше n:
		в случае если в src встретился символ c, он будет последним скопированным символом.
void	*ft_memchr(const void *s, int c, size_t n);	ищет первое вхождение символа c в область памяти s, ограниченной n байт
		возвращает указатель на искомый байт, либо NULL, если не найдено
void	*ft_memrchr(const void *s, int c, size_t n);	ищет последнее вхождение символа c в область памяти s, ограниченной n байт **
		возвращает указатель на искомый байт, либо NULL, если не найдено
void	*ft_memset(void *s, int c, size_t n);	заполняет n байт памяти байтом c. возвращает по сути тот же указатель, что получила на вход
void	ft_bzero(void *s, size_t n);	заполняет n байт памяти нулями. ничего не возвращает
void	*ft_calloc(size_t nmemb, size_t size);	возвращает указатель на занулённую область памяти (под nmemb объектов размера size каждый)
void	*ft_realloc(void *ptr, size_t oldsize, size_t newsize);	в моей реализации, маллочит область памяти размером newsize *
		копирует байты из области ptr. количество байт: меньшее число из oldsize / newsize
		освобождает память на которую указывает ptr
		возвращает указатель на новую область памяти
int	ft_atoi(const char *nptr);	возвращает число, вытащенное из строки
char	*ft_itoa(int n);	возвращает замалоченную строку с числом n
size_t	ft_strlen(const char *s);	возвращает длину нуль-терминированной строки
size_t	ft_strnlen(const char *s, size_t maxlen);	возвращает длину строки, ограниченной maxlen символов **
int	ft_strncmp(const char *s1, const char *s2, size_t n);	сравнивает не более первых n символов двух строк. возвращает 0 если идентичны.
		возвращает *s1 - *s2 если встретились различающиеся символы
char	*ft_strjoin(char const *s1, char const *s2);	возвращает указатель на замалоченную конкатенацию строк s1 и s2
char	*ft_strdup(const char *s);	возвращает указатель на замалоченный дубликат строки s
char	*ft_substr(char const *s, unsigned int start, size_t maxlen);	возвращает указатель на замолоченную подстроку строки s.
		подстрока начинается с позиции s + start и имеет максимальную длину maxlen.
		возвращает пустую строку в случае, если индекс вне пределов строки (source: Sky)
char	*ft_strtrim(char const *s1, char const *set);	возвращает указатель на замолоченную строку
		содержимое новой строки это остаток от s1, подрезанной с начала и с конца.
		выкидываем с начала и с конца символы, которые содержатся в строке set
char	**ft_split(char const *s, char c);	возвращает указатель на замалоченный массив строк, полученный разбивкой s по разделителю c
char	*ft_strchr(const char *s, int c);	ищет первое вхождение символа c в строку s. область поиска ограничена strlen+1
		(\0 рассматривается как часть строки)
char	*ft_strrchr(const char *s, int c);	ищет последнее вхождение символа c в строку s. область поиска ограничена strlen+1
		(\0 рассматривается как часть строки)
char	*ft_strnstr(const char *big, const char *little, size_t len);	ищет первое вхождение подстроки little в строку big. поиск по big ограничен len количеством символов.
		возвращает поинтер на найденное вхождение / либо NULL. если little - пустая строка, возвращает big.
size_t	ft_strlcpy(char *dst, const char *src, size_t size);	копирует из строки src в буфер dst не более чем size - 1 символов
		если size > 0 и запись символов состоится, строка dst будет гарантированно нуль-терминирована.
		возвращает длину src
size_t	ft_strlcat(char *dst, const char *src, size_t size);	конкатенирует строки, прилепляя src к dest.
		size - максимальный размер буфера(включая \0), с которым работает функция
		функция гарантирует, что в результате конкатенирования(в случае если оно состоится),
		результирующая строка будет нуль-терминирована.
		если конкатенирование состоится: вне зависимости от количества фактически скопированных символов,
		возвращает длину строки, которую требовалось запихнуть в буфер: len_dst + len_src.
		возвращает size + len_src если конкат. не состоится по причине того, что в буфере dst не найдено '\0' .
		ВАЖНО: искать '\0' надо, строго ограничиваясь размером буфера (size). Можно использовать ft_strnlen,
		можно ft_memchr.
char	*ft_strmapi(char const *s, char (*f)(unsigned int, char));	Applies the function 'f' to each character of the
		string 's' to create a new string (with malloc(3))
		resulting from successive applications of 'f'.
void	ft_putchar_fd(char c, int fd);	
void	ft_putstr_fd(char *s, int fd);	
void	ft_putendl_fd(char *s, int fd);	Outputs the string 's' to the given file descriptor, followed by \n

void	ft_putnbr_fd(int n, int fd);										
	strnstr:										
			len = 6	little = "cat"							
		big			big + len - len_l		big + len - 1	big + len			
		d	o	g	c	a	t	4	2		