חומר עזר

פונקציות לטיפול תווים

Prototype	Description
`int isdigit( int c );`	Returns `true` if `c` is a digit and `false` otherwise.
`int isalpha( int c );`	Returns `true` if `c` is a letter and `false` otherwise.
`int isalnum( int c );`	Returns `true` if `c` is a digit or a letter and `false` otherwise.
`int isxdigit( int c );`	Returns `true` if `c` is a hexadecimal digit character and `false` otherwise.
`int islower( int c );`	Returns `true` if `c` is a lowercase letter and `false` otherwise.
`int isupper( int c );`	Returns `true` if `c` is an uppercase letter; `false` otherwise.
`int tolower( int c );`	If `c` is an uppercase letter, `tolower` returns `c` as a lowercase letter. Otherwise, `tolower` returns the argument unchanged.
`int toupper( int c );`	If `c` is a lowercase letter, `toupper` returns `c` as an uppercase letter. Otherwise, `toupper` returns the argument unchanged.
`int isspace( int c );`	Returns `true` if `c` is a white-space character—newline (`'\n'`), space (`' '`), form feed (`'\f'`), carriage return (`'\r'`), horizontal tab (`'\t'`), or vertical tab (`'\v'`)—and `false` otherwise
`int iscntrl( int c );`	Returns `true` if `c` is a control character and `false` otherwise.
`int ispunct( int c );`	Returns `true` if `c` is a printing character other than a space, a digit, or a letter and `false` otherwise.
`int isprint( int c );`	Returns `true` value if `c` is a printing character including space (`' '`) and `false` otherwise.
`int isgraph( int c );`	Returns `true` if `c` is a printing character other than space (`' '`) and `false` otherwise.

כמה פונקציות על מחרוזות

Function prototype	Function description
`char *strncpy( char *s1, const char *s2, size_t n )`	Copies at most `n` characters of string `s2` into array `s1`. Does not copy '\0'. The value of `s1` is returned.
`char *strcat( char *s1, const char *s2 )`	Appends string `s2` to array `s1`. The first character of `s2` overwrites the terminating null character of `s1`. The value of `s1` is returned.
`char *strncat( char *s1, const char *s2, size_t n )`	Appends at most `n` characters of string `s2` to array `s1`. The first character of `s2` overwrites the terminating null character of `s1`. Places '\0' after last character. The value of `s1` is returned.

Function prototype	Function description
`char *strchr( const char *s, int c );`	Locates the first occurrence of character `c` in string `s`. If `c` is found, a pointer to `c` in `s` is returned. Otherwise, a `NULL` pointer is returned.
`size_t strcspn( const char *s1, const char *s2 );`	Determines and returns the length of the initial segment of string `s1` consisting of characters not contained in string `s2`.
`size_t strspn( const char *s1, const char *s2 );`	Determines and returns the length of the initial segment of string `s1` consisting only of characters contained in string `s2`.
`char *strpbrk( const char *s1, const char *s2 );`	Locates the first occurrence in string `s1` of any character in string `s2`. If a character from string `s2` is found, a pointer to the character in string `s1` is returned. Otherwise, a `NULL` pointer is returned.
`char *strrchr( const char *s, int c );`	Locates the last occurrence of `c` in string `s`. If `c` is found, a pointer to `c` in string `s` is returned. Otherwise, a `NULL` pointer is returned.
`char *strstr( const char *s1, const char *s2 );`	Locates the first occurrence in string `s1` of string `s2`. If the string is found, a pointer to the string in `s1` is returned. Otherwise, a `NULL` pointer is returned.
`char *strtok( char *s1, const char *s2 );`	A sequence of calls to `strtok` breaks string `s1` into “tokens”—logical pieces such as words in a line of text—separated by characters contained in string `s2`. The first call contains `s1` as the first argument, and subsequent calls to continue tokenizing the same string contain `NULL` as the first argument. A pointer to the current token is returned by each call. If there are no more tokens when the function is called, `NULL` is returned.

פונקציות לשינוי ומציאת המיקום בקובץ

rewind(מצביע ל**קובץ**)

מזיזה את המיקום הנוכחי ב**קובץ** (file position pointer) לתחילתו.

ftell(מצביע ל**קובץ**)

מחזירה את המיקום הנוכחי ב**קובץ**.

fseek(מצביע ל**קובץ**, offset, origin)

מזיזה את המיקום הנוכחי ב**קובץ**:

- offset הוא מספר הבתים שיש לזוז (יכול להיות גם שלילי).
- origin הוא מוצא התזוזה (אחד משלושה):
  1. SEEK_CUR - המיקום הנוכחי
  2. SEEK_SET - תחילת ה**קובץ**
  3. SEEK_END - סוף ה**קובץ**

פונקציות לקלט ופלט לקבצים בינאריים

size_t fread(void *buffer, size_t size, size_t count, FILE *stream)

- קוראת נתונים מהקובץ

- buffer – מצביע לבלוק הזיכרון
- size – גודל בבתים של כל איבר לקריאה
- count – מספר האיברים לקריאה, כל אחד בגודל size
- stream – מצביע לקובץ

- מחזירה את מספר האיברים שנקראו בהצלחה

size_t fwrite(const void *buffer, size_t size, size_t count, FILE *stream)

- כותבת נתונים לקובץ

- buffer – מצביע למערך הנתונים להיכתב
- size – גודל בבתים של כל איבר להיכתב
- count – מספר האיברים להיכתב, כל אחד בגודל size
- stream – מצביע לקובץ

- מחזירה את מספר האיברים שנכתבו בהצלחה