רשימת תווי בקרה לפונקציות קלט/פלט בשפת C

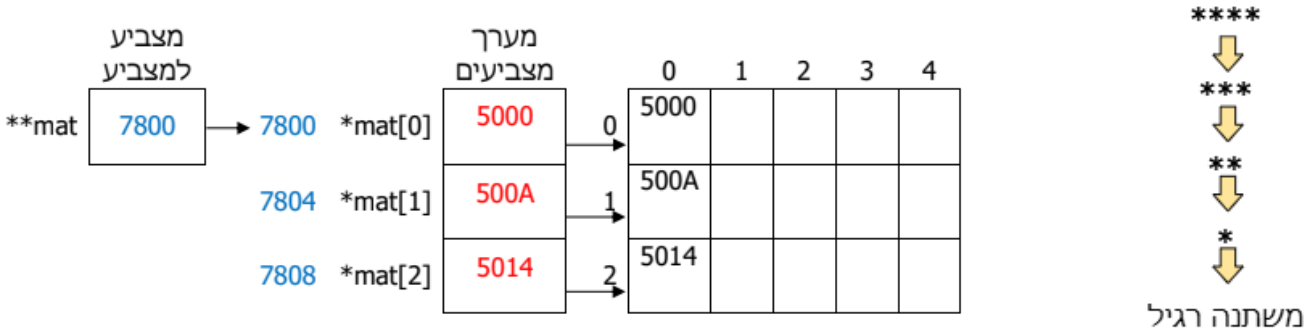
הפונקציות scanf() ו – printf() משתמשות באותם תווי בקרה על מנת לקלוט / לפלוט נתונים. להלן רשימת תווי הבקרה הקיימים בשפת C :

תו הבקרה	מייצג	טיפוס
%d	שלם בבסיס 10	char / int
%hd	שלם בבסיס 10	short int
%ld	שלם בבסיס 10	long int
%u	שלם בבסיס 10	unsigned int
%lu	שלם בבסיס 10	unsigned long
%o	שלם בבסיס 8	char / int
%x	שלם בבסיס 16	char / int
%c	תו בודד	char / unsigned char
%s	מחרוזת	char ?[?]
%e / %f	ממשי בתצוגה מדעית/עשרונית	float
%le / %lf	ממשי בתצוגה מדעית/עשרונית	double
%g	הקצר מבין מדעית/עשרונית	float / double

טבלת אופרטורים בשפת C

מס'	שם הקטגוריה	האופרטור	הסבר האופרטור
1	Highest	()	Function call
		[]	Array subscript
		->	indirect component selector
		.	direct component selector
2	Unary	!	Logical NOT
		~	Bitwise (1's) complement
		-	Unary minus
		++	Preincrement or postincrement
		--	Predecrement or postdecrement
		&	Address
		*	Indirection (pointer)
3	Multiplicative	*	Multiply
		/	Divide
		%	Remainder (modulus(
4	Additive	+	Plus
		-	minus
5	Shift	<<	Shift left
		>>	Shift right
6	Relational	<	Less than
		<=	Less than or equal to
		>	Greater than
		>=	Greater than or equal to
7	Equality	==	Equal to
		!=	Not equal to
8		&	Bitwise AND
9		^	Bitwise XOR
10		|	Bitwise OR
11		&&	Logical AND
12		||	Logical OR
13	Conditional	? :	a ? x : y means "if a then x, else y"
14	Assignment	=	Simple assignment
		*=	Assign product
		/=	Assign quotient
		=%	Assign remainder (modulus(
		+=	Assign sum
		-=	Assign difference

מצביעים למערך דו-מימדי



((m+x)+y) => m[x][y]
**(m+x) = *(*(m+x)+0) => m[x][0]
*(*m+y) = *(*(m+0)+y) => m[0][y]

טבלת ASCII
American Standard Code for Information Interchange

Dec	Hx	Oct	Char		Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL	(null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH	(start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX	(start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX	(end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT	(end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ	(enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK	(acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL	(bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS	(backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB	(horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF	(NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT	(vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF	(NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR	(carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO	(shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI	(shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE	(data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1	(device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2	(device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3	(device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4	(device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK	(negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN	(synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB	(end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN	(cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM	(end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB	(substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC	(escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS	(file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS	(group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS	(record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US	(unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

פקודות לבדיקת תו

- **isalnum**
  תפקידה : בודקת האם תו הוא ספרה או אות.

int isalnum (int c);

- **isalpha**
  תפקידה : בודקת האם תו הוא אות.

int isalpha (int c);

- **islower**
  תפקידה : בודקת האם תו הוא אות קטנה.

int islower(int c);

- **isupper**
  תפקידה : בודקת האם תו הוא אות גדולה.

int isupper(int c);

- **isdigit**
  תפקידה : בודקת האם תו הוא ספרה עשרונית.

int isdigit(int c);

- **isxdigit**
  תפקידה : בודקת האם תו הוא ספרה הקסא-דצימלית.

int isxdigit(int c);

מחרוזות

מחרוזת בשפת C מסתיימת תמיד בתו מיוחד ששמו NULL וסימנו '\0' .

```
char  שם המחרוזת  [] = "טקסט" ;
char  שם המחרוזת  [] = {תו , תו , ... , '\0'} ;  :או
```

- **strlen**

תפקידה : אורך מחרוזת.

int strlen(const char *s) ;

הפונקציה מחזירה :
את אורך המחרוזת, לא כולל null.

- **strcmp**

תפקידה : השוואה בין מחרוזות.

int strcmp(const char *s1, const char *s2) ;

הפונקציה מחזירה :
0 – במידה והמחרוזות זהות.
מספר חיובי – במידה והראשונה גדולה מהשנייה.
מספר שלילי – במידה והראשונה קטנה מהשנייה

- **strcmpi , stricmp**

תפקידה : השוואה בין מחרוזות, ללא case sensitive.

int strcmpi(const char *s1, const char *s2) ;

הפונקציה מחזירה :
0 – במידה והמחרוזות זהות.
מספר חיובי – במידה והראשונה גדולה מהשניה.
מספר שלילי – במידה והראשונה קטנה מהשניה.

- **strspn**

תפקידה : מחשבת את הרצף ההתחלתי של מחרוזת ראשונה, המכיל תווים הנמצאים במחרוזת שנייה.

int strspn(const char *s1, const char*s2) ;

הפונקציה מחזירה :
את מספר התווים הרציפים שבתחילת המחרוזת s1[] הנמצאים במחרוזת s2[].

- **strstr**

תפקידה : איתור מחרוזת שנייה בתוך המחרוזת הראשונה.

char *strstr(const char *s1, const char *s2) ;

הפונקציה מחזירה :
במידה ומצאה את s2[] במלואה - מצביע למקום הראשון במחרוזת s1[].
במידה ולא מצאה – NULL.

- **strpbrk**

תפקידה : מחפשת תו במחרוזת הראשונה הנמצא במחרוזת השנייה.

char *strpbrk(const char *s1, const char *s2) ;

הפונקציה מחזירה :
מצביע לתו הראשון במחרוזת s1[], הנמצא גם במחרוזת s2[].
NULL – במידה ולא נמצא אף תו משותף.

- **strchr**

תפקידה : איתור תו במחרוזת.

char *strchr(const char *s, char c) ;

הפונקציה מחזירה :
במידה ונמצא התו c במחרוזת s[] - מצביע למקום הראשון בו נמצא התו.
במידה ולא נמצא – מוחזר NULL.

- **strrchr**

תפקידה : לאתר תו החל **מסוף** המחרוזת.

char *strrchr(const char *s, char c) ;

הפונקציה מחזירה :
במידה ונמצא התו c במחרוזת s[] - מצביע למקום האחרון בו נמצא התו.
במידה ולא נמצא – מוחזר NULL.

- **strcat**

תפקידה : שרשור מחרוזות, צירוף מחרוזת מקור לסוף מחרוזת יעד.

char *strcat(char *dest, const char *src) ;

הפונקציה מקבלת :
dest – מחרוזת היעד אליה משרשרים.
src – מחרוזת המקור המשורשרת לסוף מחרוזת היעד.
הפונקציה מחזירה :
מצביע למחרוזת היעד.

- **strcpy**

תפקידה : העתקה (דריסה) של מחרוזת המקור למחרוזת היעד.

char *strcpy(char *dest, char *src) ;

הפונקציה מקבלת :
dest – מחרוזת היעד אליה מעתיקים.
src – מחרוזת המקור המועתקת לתחילת מחרוזת היעד.
הפונקציה מחזירה :
מצביע למחרוזת היעד.

- **sprintf**

תפקידה : להדפיס לתוך מחרוזת (במקום מסך)

int sprint(char * str, const char * format, ...);

הפונקציה מקבלת :
str – מחרוזת היעד אליה מדפיסים.
Format – פורמט הפלט של printf()
הפונקציה מחזירה :
בהצלחה – הפונקציה חותמת את המחרוזת ב-'\0' ומחזירה את אורכה.
בכישלון – מחזירה ערך שלילי.

- **sscanf**

תפקידה : לקלוט תוך מחרוזת (במקום מהמקלדת)

int sscanf (const char * s, const char * format, ...);

הפונקציה מקבלת :
s – מחרוזת המקור ממנה קולטים.
Format – פורמט הקלט של scanf()
הפונקציה מחזירה :
בהצלחה –המחזירה את מספר המשתנים שהצליחה לקלוט לתוכם.
בכישלון – מחזירה EOF (לרוב מדובר על 1-).

- **strtok**

תפקידה : לפצל מחרוזת.

char * strtok (char * str, const char * delimiters);

הפונקציה מקבלת :
str – מחרוזת לפיצול.
delimiters – מחרוזת התווים המהווים אינדיקטור לפיצול.
הפונקציה מחזירה :
בקריאה הראשונה – מחזירה מצביע למקום הראשון ב-str שאינו תו פיצול.
כמו כן, מציבה '\0' בתו הפיצול הבא.
בקריאות הבאות – יש לשלוח לה NULL במקום str. הפונקציה מחזירה למקום הבא שאינו תו פיצול ומציבה '\0' בתו הפיצול הבא.
ברגע שהמחרוזת מסתיימת או לא נמצא תו פיצול – מחזירה NULL.

מצביעים מסוג VOID ומצביעים על פונקציות

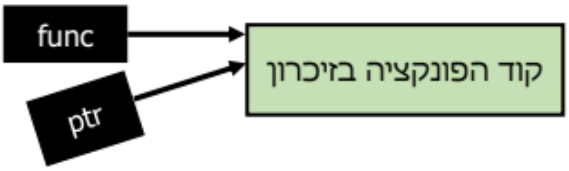
היתרון של מצביע מסוג void הוא שניתן להמירו לכל טיפוס אחר, מבלי לאבד מידע. ברגע שטיפוס המשתנה ידוע, נבצע casting למצביע מטיפוס void לטיפוס הרצוי.

לדוגמא :

```
void *p ;
int num ;
p = &num ;
*p=500;        // בעייתי! למצביע אין טיפוס
*(int *)p= 50 ;        // הנתון 50 יתפרש על פני 4 בתים
*(char *)p= 50 ;      // הנתון 50 יתפרש על פני בית אחד
```

לדוגמא :

```
float func(int,char) ;// הגדרת הפונקציה
float (*ptr)(int,char) ;        // הגדרת מצביע חופשית, זבלי
ptr=func ;     // ptr=&func ; קישור המצביע החופשי לראש הפונקציה
```



מסקנה א' :

שם הפונקציה הוא גם מצביע קבוע לראשה.

אם נרצה לשלוח לפונקציה f() את הפונקציה func(), עליה הצהרנו קודם :

```
f(func) ;        // קריאה לפונקציה y
f(ptr) ;// : או
```

להלן כותרת הפונקציה f() :

```
void f (float (*pFunc)(int,char))
{
        ...
}
```

הפונקציה qsort()

הגדרת הפונקציה :

```
void qsort( void *buf, size_t num, size_t size, int (*compare)(const void *, const void *) );
```

הפונקציה מקבלת :

- buf - מצביע לתחילת בלוק נתונים המיועד למיון.
- num – מספר האיברים המיועדים למיון בבלוק.
- size – גודל ב – bytes של כל איבר בבלוק.
- compare – מצביע לפונקציה המוגדרת ע"י המתכנת. בזמן המיון שולחת הפונקציה qsort() לפונקציה compare() שני מצביעים מסוג void לשני איברים מהבלוק.
  compare() אחראית לבצע את ההשוואה ולהחזיר :
  - ערך אפס – אם האיברים זהים.
  - ערך חיובי – אם האיבר הראשון גדול מהשני.
  - ערך שלילי – אם האיבר הראשון קטן מהשני.

בצורה זו שומרת הפונקציה qsort() על אי-תלותה בטיפוס הבלוק הממוין.

הפונקציה bsearch()

הגדרת הפונקציה :

```
void *bsearch( const void *key, const void *buf, size_t num, size_t size, int (*compare)(const void *, const void *) );
```

הפונקציה מקבלת :

- key – מצביע לערך המיועד לחיפוש.
- buf - מצביע לתחילת בלוק נתונים ממוין בסדר עולה בו מחפשים את key.
- num – מספר האיברים הקיימים בבלוק.
- size – גודל ב – bytes של כל איבר בבלוק.
- compare – מצביע לפונקציה המוגדרת ע"י המתכנת. בזמן החיפוש שולחת הפונקציה bsearch() לפונקציה compare() שני מצביעים מסוג void (key ואיבר מהבלוק).
  compare() אחראית לבצע את ההשוואה ולהחזיר :
  - ערך אפס – אם האיברים זהים.
  - ערך חיובי – אם האיבר הראשון גדול מהשני.
  - ערך שלילי – אם האיבר הראשון קטן מהשני.

בצורה זו שומרת הפונקציה bsearch() על אי-תלותה בטיפוס הבלוק.

הפונקציה מחזירה : מצביע לאיבר המתאים או NULL במידה ולא נמצא.

הקצאה דינמית

malloc()

תחביר הקריאה לפונקציה :

```
מצביע =(casting) malloc (bytes 'מס) ;
```

calloc()

תחביר הקריאה לפונקציה :

```
מצביע =(casting) calloc (nitems,size) ;
```

* בשונה מmalloc, פעולה זו גם מאפסת את בלוק הנתונים המוחזר

realloc()

תחביר הקריאה לפונקציה :

```
מצביע =(casting) realloc (ptr,size) ;
```

free()

תחביר הקריאה לפונקציה :

```
free (ptr) ;
```

קבצים

FILE *fopen(const char *fname, const char *mode);

הפונקציה fopen() מקבלת :

- fname - שם הקובץ שיש לפתוח.
  השם צריך לכלול נתיב, שם וסיומת.
  - במידה ולא נציין את הנתיב, התוכנית תחפש את הקובץ בספריה בה נמצא קובץ ההפעלה. לדוגמא : "filename.txt".
  - במידה והקובץ נמצא בתת ספריה בה נמצא קובץ ההפעלה, נציין תחילה את תת הספריה עם הסימן \\. לדוגמא : "aaa\\filename.txt".
  - במידה והקובץ נמצא בספריה אחרת, יש לציין נתיב מלא, תוך שימוש בסימן \\. לדוגמא : "c:\\bbb\\ccc\\filename.txt".

הפונקציה מחזירה את הערכים הבאים :

- במידה ופתיחת הקובץ עברה בהצלחה – הפונקציה תחזיר מצביע לתחילת הקובץ הפתוח.
- במידה ופתיחת הקובץ נכשלה – הפונקציה תחזיר NULL.

פתיחת קובץ יכולה להיכשל מכל מיני סיבות כגון : קובץ לא קיים, ספריה לא קיימת, בעיית חומרה וכו'. ולכן, בכל בקשה לפתיחת קובץ, יש לוודא שהיא עברה בהצלחה.

הרשאה לקובץ טקסט	הרשאה לקובץ בינארי	משמעות ההרשאה
"wt" (write)	"wb" (write)	פתיחת קובץ חדש לכתיבה בלבד. במידה והקובץ קיים, תוכנו ימחק.
"rt" (read)	"rb" (read)	פתיחת קובץ לקריאה בלבד. הקובץ חייב להיות קיים.
"at"	"ab"	פתיחת קובץ להוספת נתונים בסופו. אם הקובץ לא קיים, יווצר חדש.
"r+t"	"r+b"	פתיחת קובץ לקריאה ולכתיבה. הקובץ חייב להיות קיים.
"w+t"	"w+b"	פתיחת קובץ חדש לקריאה ולכתיבה. במידה והקובץ קיים, תוכנו ימחק.
"a+t" (add)	"a+b" (add)	פתיחת קובץ לקריאה והוספה. פעולת הכתיבה תבוצע בסוף הקובץ.

int fclose(FILE *stream);

הפונקציה fclose() מקבלת :

- stream - מצביע לקובץ פתוח.

הפונקציה מחזירה את הערכים הבאים :

- 0 – במידה והסגירה עברה בהצלחה.
- EOF - במידה והסגירה לא עברה בהצלחה

int fprintf(FILE *stream, const char *format, ...);

הפונקציה fprintf() מקבלת :

- stream - מצביע לקובץ פתוח.
- format - מחרוזת בקרה.
- ... - פרמטרים.

הפונקציה כותבת את הפרמטרים לקובץ, בהתאם לפורמט שנקבע במחרוזת הבקרה.

הפונקציה מחזירה את הערכים הבאים :

- מספר התווים שנכתבו לקובץ – במידה והכתיבה הסתיימה בהצלחה.
  תווי בקרה (כגון : \t\ , n) נספרים כתו לכל דבר.
- מספר שלילי - במידה והפונקציה נכשלה בכתיבה לקובץ.

<u>הערה</u> : ניתן להשתמש בפונקציה fprintf() גם כאשר רוצים להציג נתונים להתקן הפלט הסטנדרטי (מסך) על ידי כך שבשדה הראשון של הפונקציה fprintf() נכתוב stdout.

int fputc(int ch, FILE *stream);

הפונקציה fputc() מקבלת :

- ch - תו אותו יש לכתוב לקובץ.
- stream - מצביע לקובץ פתוח.

הפונקציה מחזירה את הערכים הבאים :

- ch – במידה והכתיבה הסתיימה בהצלחה.
- EOF – במידה והכתיבה נכשלה.

int fputs(const char *str, FILE *stream);

הפונקציה fputs() מקבלת :

- str – מחרוזת לכתיבה לקובץ.
- stream - מצביע לקובץ פתוח.

הפונקציה מחזירה את הערכים הבאים :

- מספר חיובי (0 ומעלה) – במידה והכתיבה הסתיימה בהצלחה.
- EOF – במידה והכתיבה נכשלה.

<u>הערה</u> : בניגוד לפונקציה puts(), הפונקציה fputs() אינה גורמת לסמן הכתיבה לרדת לתחילת שורה חדשה בסיום כתיבת המחרוזת.

קריאה וכתיבה לקבצי טקסט

int fscanf(FILE *stream, const char *format, ...);

הפונקציה fscanf() מקבלת :

- stream - מצביע לקובץ פתוח.
- format - מחרוזת בקרה.
- ... - פרמטרים.

הפונקציה קוראת את הפרמטרים מהקובץ, בהתאם לפורמט שנקבע במחרוזת הבקרה.
מחרוזת (s%) היא קוראת עד סוף השורה או הקובץ, משתנה היא קוראת עד לסופו (בתנאי שהוא מופיע בתחילת שורה).

הפונקציה מחזירה את הערכים הבאים :

- מספר המחרוזות/המשתנים שנקראו מהקובץ – במידה והקריאה הסתיימה בהצלחה.
- מספר שלילי - במידה והפונקציה נכשלה בקריאה מהקובץ.

<u>הערה</u>: ניתן להשתמש בפונקציה fscanf() גם כאשר רוצים לקלוט נתונים מהתקן הקלט הסטנדרטי (מקלדת) על ידי כך שבשדה הראשון של הפונקציה fscanf() נכתוב stdin.

int fgetc(FILE *stream);

הפונקציה fgetc() מקבלת :

- stream - מצביע לקובץ פתוח.

הפונקציה מחזירה את הערכים הבאים :

- תו – במידה והקריאה התבצעה בהצלחה.
- EOF – במידה והקריאה נכשלה או הגענו לסוף הקובץ.

char *fgets(char *str, int num, FILE *stream);

הפונקציה fgets() מקבלת :

- str - מחרוזת יעד היושבת בזיכרון הראשי.
- num - מספר תווים לקריאה+1.
- stream - מצביע לקובץ פתוח.

הפונקציה תקרא תווים מהקובץ, כל עוד שלושת התנאים הבאים מתקיימים :

1. כל עוד לא סיימה לקרוא num-1 תווים.
2. כל עוד לא הגיעה לסוף שורה.
3. כל עוד לא הגיעה סוף הקובץ (EOF).

ברגע שאחד מהתנאים הללו מפסיק להתקיים, היא מפסיקה את הקריאה ומעתיקה את התווים שקראה למחרוזת היעד str.

הפונקציה מחזירה את הערכים הבאים :

- מחרוזת – במידה והקריאה תקינה.
- NULL – במידה והקריאה לא התבצעה באופן תקין.

קריאה וכתיבה לקבצים בינארים

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

הפונקציה fwrite() מקבלת :

- buffer - מצביע לתחילת קטע הנתונים אותו רוצים לכתוב לקובץ.
- size - גודל כל איבר ב – bytes.
- count - כמות איברים.
- stream - מצביע לקובץ בינארי פתוח.

הפונקציה fwrite() מחזירה את מספר האיברים שנכתבו בפועל.

כלומר, ניתן להשוות את הערך המוחזר למספר האיברים המבוקש count על מנת לבדוק האם הצליחה במשימה.

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

הפונקציה fread() מקבלת את הפרמטרים הבאים :

- buffer - מצביע למשתנה בזיכרון הראשי אליו רוצים להעתיק את תוכן הקובץ.
- size - גודל כל איבר ב – bytes.
- count - כמות איברים.
- stream - מצביע לקובץ בינארי פתוח.

הפונקציה fread() מחזירה את מספר האיברים שנקראו בפועל.
כלומר, ניתן להשוות את הערך המוחזר למספר האיברים המבוקש count על מנת לבדוק האם הצליחה במשימה.

פקודות קבצים נוספות

int feof(FILE *stream);

הפונקציה feof() מקבלת :

- stream - מצביע לקובץ פתוח.

היא מחזירה את הערכים הבאים :

- 0 – במידה והמצביע לקובץ לא הגיע לסוף הקובץ.
- ערך השונה מ-0 - כאשר המצביע לקובץ הגיע לתו eof.

int fseek(FILE *stream, long offset, int origin);

לכל קובץ קיים מצביע המציין את המיקום הנוכחי של הקובץ. המיקום הנוכחי הוא המקום של ה – byte הבא ממנו יש לקרוא או לכתוב לקובץ.

הפונקציה fseek() מקבלת את הפרמטרים הבאים :

- stream – מצביע לקובץ בינארי פתוח.
- Offset – מספר bytes להזזה קדימה (חיובי) או אחורה (שלילי).
- Origin – נקודת המוצא לתזוזה (קבוע).
  נקודת המוצא origin יכולה להיות אחד מהקבועים הבאים :
  - SEEK_SET – תחילת הקובץ.
  - SEEK_CUR – המיקום הנוכחי של סמן הקובץ.
  - SEEK_END – סוף הקובץ.

הפונקציה fseek() מחזירה את הערכים הבאים :

- 0 - במידה והסתיימה בהצלחה.
- ערך השונה מ-0 - במידה ולא הסתיימה בהצלחה.

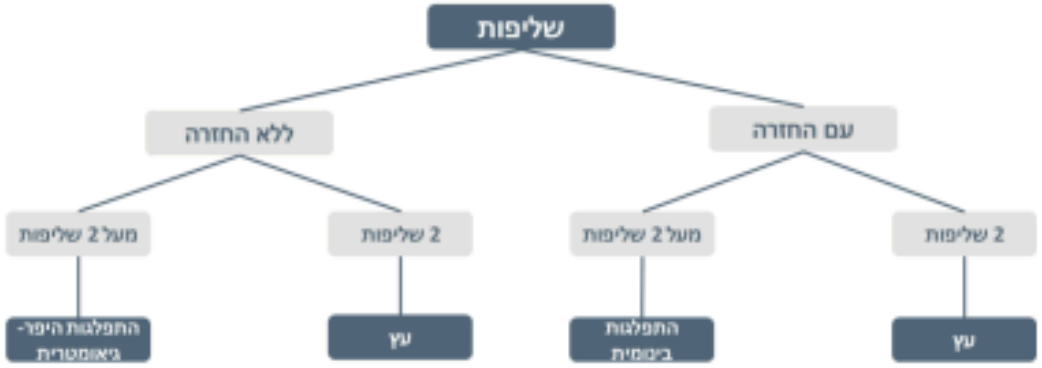
long ftell(FILE *stream);

הפונקציה ftell() מקבלת :

*stream - מצביע לקובץ בינארי פתוח.

היא מחזירה את מיקומו, כלומר את המרחק שלו ב - bytes מתחילת הקובץ.

זיהוי שאלה


שליפות
עם החזרה
ללא החזרה
2 שליפות
מעל 2 שליפות
2 שליפות
מעל 2 שליפות
עץ
התפלגות בינומית
עץ
התפלגות היפר-גיאומטרית


ריבוע קסם
2 מאורעות בלבד
נתונה הסתברות שלפחות מאורע אחד יתרחש
נתונה הסתברות שאף מאורע לא יתרחש
נתונה הסתברות שלכל היותר מאורע אחד יתרחש
נתונה הסתברות ש2 מאורעות יתרחשו


עץ
נתונות הסתברויות מותנות
(=מיון, מבט, מקבל...)