Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	00	Null	32	20	Space	64	40	@	96	60	`
1	01	Start of heading	33	21	!	65	41	A	97	61	a
2	02	Start of text	34	22	"	66	42	B	98	62	b
3	03	End of text	35	23	#	67	43	C	99	63	c
4	04	End of transmit	36	24	$	68	44	D	100	64	d
5	05	Enquiry	37	25	%	69	45	E	101	65	e
6	06	Acknowledge	38	26	&	70	46	F	102	66	f
7	07	Audible bell	39	27	'	71	47	G	103	67	g
8	08	Backspace	40	28	(	72	48	H	104	68	h
9	09	Horizontal tab	41	29	)	73	49	I	105	69	i
10	0A	Line feed	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage return	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	47	2F	/	79	4F	O	111	6F	o
16	10	Data link escape	48	30	0	80	50	P	112	70	p
17	11	Device control 1	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	50	32	2	82	52	R	114	72	r
19	13	Device control 3	51	33	3	83	53	S	115	73	s
20	14	Device control 4	52	34	4	84	54	T	116	74	t
21	15	Neg. acknowledge	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	54	36	6	86	56	V	118	76	v
23	17	End trans. block	55	37	7	87	57	W	119	77	w
24	18	Cancel	56	38	8	88	58	X	120	78	x
25	19	End of medium	57	39	9	89	59	Y	121	79	y
26	1A	Substitution	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	59	3B	;	91	5B	[	123	7B	{
28	1C	File separator	60	3C	<	92	5C	\	124	7C	|
29	1D	Group separator	61	3D	=	93	5D	]	125	7D	}
30	1E	Record separator	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	63	3F	?	95	5F	_	127	7F	▢

ASCII

128 Chars

ord( )

↓

ASCII Value.

32:	64: @	96: `	128:	160:	192: Р	224: р
33: !	65: A	97: a	129:	161: Ё	193: С	225: с
34: "	66: B	98: b	130:	162: Ђ	194: Т	226: т
35: #	67: C	99: c	131:	163: Ґ	195: У	227: у
36: $	68: D	100: d	132:	164: Є	196: Ф	228: ф
37: %	69: E	101: e	133:	165: S	197: Х	229: х
38: &	70: F	102: f	134:	166: I	198: Ц	230: ц
39: '	71: G	103: g	135:	167: Ї	199: Ч	231: ч
40: (	72: H	104: h	136:	168: J	200: Ш	232: ш
41:)	73: I	105: i	137:	169: Љ	201: Щ	233: щ
42: *	74: J	106: j	138:	170: Њ	202: Ъ	234: ъ
43: +	75: K	107: k	139:	171: Ћ	203: Ы	235: ы
44: ,	76: L	108: l	140:	172: Ќ	204: Ь	236: ь
45: -	77: M	109: m	141:	173:	205: Э	237: э
46: .	78: N	110: n	142:	174: Ў	206: Ю	238: ю
47: /	79: O	111: o	143:	175: Џ	207: Я	239: я
48: 0	80: P	112: p	144:	176: А	208: а	240: №
49: 1	81: Q	113: q	145:	177: Б	209: б	241: ё
50: 2	82: R	114: r	146:	178: В	210: в	242: ђ
51: 3	83: S	115: s	147:	179: Г	211: г	243: ѓ
52: 4	84: T	116: t	148:	180: Д	212: д	244: є
53: 5	85: U	117: u	149:	181: Е	213: е	245: s
54: 6	86: V	118: v	150:	182: Ж	214: ж	246: i
55: 7	87: W	119: w	151:	183: З	215: з	247: ї
56: 8	88: X	120: x	152:	184: И	216: и	248: j
57: 9	89: Y	121: y	153:	185: Й	217: й	249: љ
58: :	90: Z	122: z	154:	186: К	218: к	250: њ
59: ;	91: [	123: {	155:	187: Л	219: л	251: ћ
60: <	92: \	124: |	156:	188: М	220: м	252: ќ
61: =	93:]	125: }	157:	189: Н	221: н	253: §
62: >	94: ^	126: ~	158:	190: О	222: о	254: ў
63: ?	95: _	127:	159:	191: П	223: п	255: џ

w, a, r w+,

text files (those opened without a `b` in the mode string), only seeks relative to the beginning of the file are owed (the exception being seeking to the very file end with `seek(0, 2)`) and the only valid *offset* values are ose returned from the `f.tell()`, or zero. Any other *offset* value produces undefined behaviour.

fp.seek(arg1, arg2)

offset

relative

fp.tell + x

0

relative → 0 beginning

→ 2 End

?

10 => fp.seek(fp.tell()+10, 0)

10th place → 12th place

fp.seek(fp.tell()+2, 0)