Library Facilities Alphabetized by Name

Syntax	Header File	Purpose
`void abort(void);`	`stdlib.h`	Abnormally terminates a program.
`int abs(int x);`	`stdlib.h`	Returns the absolute value of an integer.
`double acos(double x);`	`math.h`	Returns the arc cosine of the input value (argument must be in the range –1 to 1).
`char *asctime (const struct tm *tblock);`	`time.h`	Converts a time stored as a structure in `*tblock` to a 26-character string.
`double asin(double x);`	`math.h`	Returns the arc sine of the input value (argument must be in the range–1 to 1).
`void assert(int test);`	`assert.h`	If `test` evaluates to zero, `assert` prints a message on `stderr` and aborts the program.
`double atan(double x);`	`math.h`	Calculates the arc tangent of the input value.
`double atan2(double y, double x);`	`math.h`	Calculates the arc tangent of `y/x`.
`int atexit(void (*func)(void));`	`stdlib.h`	Registers a function to be called at normal program termination.
`double atof(const char *s);`	`math.h`	Converts a string pointed to by `s` to `double`.
`int atoi(const char *s);`	`stdlib.h`	Converts a string pointed to by `s` to `int`.
`long int atol(const char *s);`	`stdlib.h`	Converts a string pointed to by `s` to `long int`.
`void *bsearch(const void *key, const void *base, size_t nelem, size_t width, int (*fcmp)(const void *, const void *));`	`stdlib.h`	Binary search of the sorted array `base`: returns the address of the first entry in the array that matches the search key using the comparison routine `*fcmp`; if no match is found, returns 0.

Library Facilities Alphabetized by Name

Syntax	Header File	Purpose
`void *calloc(size_t nitems, size_t size);`	stdlib.h	Allocates a memory block of size `nitems` × `size`, clears the block to zeros, and returns a pointer to the newly allocated block.
`double ceil(double x);`	math.h	Returns the smallest integer not less than `x`.
`void clearerr(FILE *stream);`	stdio.h	Resets `stream`'s error and end-of-file indicators to 0.
`clock_t clock(void);`	time.h	Returns processor time elapsed since the beginning of program invocation.
`double cos(double x);`	math.h	Calculates the cosine of a value (angle in radians).
`double cosh(double x);`	math.h	Calculates the hyperbolic cosine of a value.
`char *ctime (const time_t *time);`	time.h	Converts date and time value pointed to by `time` (the value returned by function `time`) into a 26-character string representing local time.
`double difftime (time_t time2, time_t time1);`	time.h	Calculates the difference between two times in seconds.
`div_t div(int numer, int denom);`	stdlib.h	Divides two integers, returning quotient and remainder in a structure whose components are `quot` and `rem`.
`void exit(int status);`	stdlib.h	Terminates program. Before termination, all files are closed, buffered output (waiting to be output) is written, and any registered "exit functions" (posted with `atexit`) are called; status of 0 indicates normal exit; a nonzero status indicates some error.
`double exp(double x);`	math.h	Calculates the exponential function e^x.
`double fabs(double x);`	math.h	Calculates the absolute value of a floating-point number.
`int fclose(FILE *stream);`	stdio.h	Closes the named stream.
`int feof(FILE *stream);`	stdio.h	Predicate that detects end of file on a stream.
`int ferror(FILE *stream);`	stdio.h	Predicate that detects errors on a stream.
`int fflush(FILE *stream);`	stdio.h	Flushes a stream: If the stream has buffered output, `fflush` writes the output for `stream` to the associated file.
`int fgetc(FILE *stream);`	stdio.h	Gets a character from a stream.

Library Facilities Alphabetized by Name

Syntax	Header File	Purpose
`int fgetpos(FILE *stream, fpos_t *pos);`	stdio.h	Gets the current file pointer and stores it in the location pointed to by pos.
`char *fgets(char *s, int n, FILE *stream);`	stdio.h	Copies characters from stream into the string s until it has read n-1 characters or newline character, whichever comes first. Marks the end of s with the null character.
`double floor(double x);`	math.h	Returns the largest whole number not greater than x.
`double fmod(double x, double y);`	math.h	Calculates x modulo y, the remainder of x divided by y.
`FILE *fopen (const char *filename, const char *mode);`	stdio.h	Opens file named by filename and associates a stream with it. Modes and meanings "r" is read, "w" is write, "a" is append, "r+" is existing file update (reading and writing), "w+" is new file update (reading and writing), "a+" is update at the end of the file
`int fprintf(FILE *stream, const char *format [, argument, ...]);`	stdio.h	Writes formatted output to a stream.
`int fputc(int c, FILE *stream);`	stdio.h	Outputs a character to a stream.
`int fputs(const char *s, FILE *stream);`	stdio.h	Outputs a string to a stream.
`size_t fread(void *ptr, size_t size, size_t n, FILE *stream);`	stdio.h	Reads up to n items of data, each of length size bytes, from the given stream into a block pointed to by ptr; returns number of items read.
`void free(void *block);`	stdlib.h	Deallocates a memory block allocated by a previous call to calloc, malloc, or realloc
`FILE *freopen (const char *filename, const char *mode, FILE *stream);`	stdio.h	Associates a new file with an open stream; often used for redirecting standard streams.
`double frexp(double x, int *exponent);`	math.h	Splits a double number into mantissa and exponent.
`int fscanf(FILE *stream, const char *format [, address, ...]);`	stdio.h	Scans and formats input from a stream.

Syntax	Header File	Purpose
`int fseek(FILE *stream, long int offset, int whence);`	stdio.h	Repositions the file pointer associated with `stream` to a new position that is `offset` bytes from the file location given by whence.
`int fsetpos(FILE *stream, const fpos_t *pos);`	stdio.h	Positions the file pointer of a stream to a new position that is the value obtained by a previous call to `fgetpos` on that stream.
`long int ftell(FILE *stream);`	stdio.h	Returns the current file pointer for `stream` as the number of bytes from the beginning of the file.
`size_t fwrite (const void *ptr, size_t size, size_t n, FILE *stream);`	stdio.h	Writes n × `size` bytes to `stream` from the memory block pointed to by `ptr`.
`int getc(FILE *stream);`	stdio.h	Gets a character from `stream`.
`int getchar(void);`	stdio.h	Gets a character from `stdin`.
`char *getenv(const char *name);`	stdlib.h	Returns the value of a specified variable.
`char *gets(char *s);`	stdio.h	Gets a string (one line) from `stdin`; discards any newline character.
`struct tm *gmtime (const time_t *timer);`	time.h	Converts date and time to Greenwich mean time (GMT).
`int isalnum(int c);`	ctype.h	Predicate returning nonzero if `c` is a letter or a decimal digit.
`int isalpha(int c);`	ctype.h	Predicate returning nonzero if `c` is a letter.
`int iscntrl(int c);`	ctype.h	Predicate returning nonzero if `c` is a delete character or an ordinary control character.
`int isdigit(int c);`	ctype.h	Predicate returning nonzero if `c` is a decimal digit.
`int isgraph(int c);`	ctype.h	Predicate returning nonzero if `c` is a printing character other than a space.
`int islower(int c);`	ctype.h	Predicate returning nonzero if `c` is a lowercase letter.
`int isprint(int c);`	ctype.h	Predicate returning nonzero if `c` is a printing character.
`int ispunct(int c);`	ctype.h	Predicate returning nonzero if `c` is a punctuation character.

Library Facilities Alphabetized by Name

Syntax	Header File	Purpose
`int isspace(int c);`	`ctype.h`	Predicate returning nonzero if `c` is a space, tab, carriage return, new line, vertical tab, or form feed.
`int isupper(int c);`	`ctype.h`	Predicate returning nonzero if `c` is an uppercase letter.
`int isxdigit(int c);`	`ctype.h`	Predicate returning nonzero if `c` is a hexadecimal digit (0 to 9, A to F, a to f).
`long int labs(long int x);`	`math.h`	Computes the absolute value of the parameter `x`.
`double ldexp(double x, int exp);`	`math.h`	Calculates $x \times 2^{exp}$.
`ldiv_t ldiv(long int numer, long int denom);`	`stdlib.h`	Divides two `long ints`, returning quotient and remainder in a structure whose components are `quot` and `rem`.
`struct lconv *localeconv(void);`	`locale.h`	Sets up country-specific monetary and other numeric formats.
`struct tm *localtime (const time_t *timer);`	`time.h`	Accepts the address of a value returned by `time` and returns a pointer to a structure of type `tm` in which the time is corrected for the time zone and possible daylight savings time.
`double log(double x);`	`math.h`	Calculates the natural logarithm of `x`.
`double log10(double x);`	`math.h`	Calculates $\log_{10}(x)$.
`void longjmp(jmp_buf jmpb, int retval);`	`setjmp.h`	Restores the task state captured by the last call to `setjmp` with the argument `jmpb`; then returns in such a way that `setjmp` appears to have returned with the value `retval`.
`void *malloc(size_t size);`	`stdlib.h`	Allocates a block of `size` bytes from the memory heap and returns a pointer to the newly allocated block.
`int mblen (const char *s, size_t n);`	`stdlib.h`	Returns the size in bytes of the multibyte character pointed to by `s` (`n` is the maximum size of the character).
`size_t mbstowcs(wchar_t *pwcs, const char *s, size_t n);`	`stdlib.h`	Converts up to `n` multibyte characters from string `s` to wide characters stored in array `pwcs`.
`int mbtowc(wchar_t *pwc, const char *s, size_t n);`	`stdlib.h`	Converts the multibyte character accessed by `s` to a wide character.

Library Facilities Alphabetized by Name

Syntax	Header File	Purpose
`void *memchr(const void *s, int c, size_t n);`	`string.h`	Searches the first n bytes of the block pointed to by `s` for first occurrence of character `c`.
`int memcmp(const void *s1, const void *s2, size_t n);`	`string.h`	Compares two blocks for a length of exactly n bytes; return value < 0 means `s1` less than `s2`, value = 0 means same as, and value > 0 means greater than.
`void *memcpy(void *dest, const void *src, size_t n);`	`string.h`	Copies a block of n bytes from `src` to `dest` (behavior undefined if `src` and `dest` overlap); returns `dest`.
`void *memmove(void *dest, const void *src, size_t n);`	`string.h`	Copies a block of n bytes from `src` to `dest` (copy is correct even if `src` and `dest` overlap); returns `dest`.
`void *memset(void *s, int c, size_t n);`	`string.h`	Sets the first n bytes of the array `s` to the character `c`.
`time_t mktime(struct tm *t);`	`time.h`	Converts the time in the structure pointed to by `t` into a calendar time.
`double modf(double x, double *ipart);`	`math.h`	Splits a `double` into integer and fractional parts, both with the same sign as `x`.
`void perror(const char *s);`	`stdio.h`	Prints to the `stderr` stream the system error message for the last library routine that produced the error.
`double pow(double x, double y);`	`math.h`	Calculates x^y.
`int printf(const char *format [, argument, ...]);`	`stdio.h`	Writes formatted output to `stdout`.
`int putc(int c, FILE *stream);`	`stdio.h`	Outputs a character to `stream`.
`int putchar(int c);`	`stdio.h`	Outputs a character to `stdout`.
`int puts(const char *s);`	`stdio.h`	Outputs a string to `stdout`; terminates output by a newline character.
`void qsort(void *base, size_t nelem, size_t width, int (*fcmp)(const void *, const void *));`	`stdlib.h`	Sorts array `base` using the quicksort algorithm based on the comparison function pointed to by `fcmp`.
`int raise(int sig);`	`signal.h`	Sends a signal of type `sig` to the program. If the program has installed a signal handler for the signal type specified by `sig`, that handler will be executed.

Syntax	Header File	Purpose
`int rand(void);`	`stdlib.h`	Returns successive pseudorandom numbers in the range from 0 to `RAND_MAX` (constant defined in `stdlib.h`).
`void *realloc(void *block, size_t size);`	`stdlib.h`	Attempts to shrink or expand the previously allocated block to `size` bytes, copying the contents to a new location if necessary.
`int remove (const char *filename);`	`stdio.h`	Deletes the file specified by `filename`.
`int rename(const char *oldname, const char *newname);`	`stdio.h`	Changes the name of a file from `oldname` to `newname`.
`void rewind(FILE *stream);`	`stdio.h`	Repositions a file pointer to the beginning of a stream.
`int scanf(const char *format [, address, ...]);`	`stdio.h`	Scans and formats input from `stdin` stream.
`void setbuf(FILE *stream, char *buf);`	`stdio.h`	Causes the buffer `buf` to be used for I/O buffering instead of an automatically allocated buffer.
`int setjmp(jmp_buf jmpb);`	`setjmp.h`	Captures the complete task state in `jmpb` and returns 0.
`char *setlocale(int category, char *locale);`	`locale.h`	Selects a locale; if selection is successful, returns a string indicating the locale that was in effect prior to invoking the function.
`int setvbuf(FILE *stream, char *buf, int type, size_t size);`	`stdio.h`	Causes the buffer `buf` to be used for I/O buffering instead of an automatically allocated buffer. The `type` parameter may be `_IOFBF` (fully buffered), `_IOLBF` (line buffered), or `_IONBF` (unbuffered).
`void (*signal(int sig, void (*func)(int sig))) (int);`	`signal.h`	Specifies signal-handling actions.
`double sin(double x);`	`math.h`	Calculates the sine of the input value (angles in radians).
`double sinh(double x);`	`math.h`	Calculates hyperbolic sine.
`int sprintf(char *buffer, const char *format [, argument, ...]);`	`stdio.h`	Writes formatted output to a string.
`double sqrt(double x);`	`math.h`	Calculates the positive square root of a nonnegative input value.

Syntax	Header File	Purpose
`void srand(unsigned int seed);`	`stdlib.h`	Initializes random number generator.
`int sscanf(const char *buffer, const char *format [, address, ...]);`	`stdio.h`	Scans and formats input from a string.
`char *strcat(char *dest, const char *src);`	`string.h`	Appends a copy of `src` to the end of `dest`; returns `dest`.
`char *strchr(const char *s, int c);`	`string.h`	Returns a pointer to the first occurrence of the character `c` in the string `s` (or null).
`int strcmp(const char *s1, const char *s2);`	`string.h`	Compares one string to another; return value < 0 means `s1` less than `s2`, value = 0 means same as, and value > 0 means greater than.
`int strcoll(const char *s1, const char *s2);`	`string.h`	Compares two strings according to the collating sequence set by `setlocale`; return value < 0 means `s1` less than `s2`, value = 0 means same as, and value > 0 means greater than.
`char *strcpy(char *dest, const char *src);`	`string.h`	Copies string `src` to `dest`, stopping after copying the terminating null character; returns `dest`.
`size_t strcspn(const char *s1, const char *s2);`	`string.h`	Returns the length of the initial segment of string `s1` that consists entirely of characters *not* from string `s2`.
`char *strerror(int errnum);`	`string.h`	Returns a pointer to an error message string associated with `errnum`.
`size_t strftime(char *s, size_t maxsize, const char *fmt, const struct tm *t);`	`time.h`	Formats time for output according to the `fmt` specifications; returns the number of characters placed into `s`.
`size_t strlen(const char *s);`	`string.h`	Returns the number of characters in `s`, not counting the null terminating character.
`char *strncat(char *dest, const char *src, size_t maxlen);`	`string.h`	Copies at most `maxlen` characters of `src` to the end of `dest` and appends a null character.
`int strncmp(const char *s1, const char *s2, size_t maxlen);`	`string.h`	Compares a portion (no more than `maxlen` characters) of one string to a portion of another; return value < 0 means portion of `s1` less than portion of `s2`, value = 0 means same as, and value > 0 means greater than.

Syntax	Header File	Purpose
`char *strncpy(char *dest, const char *src, size_t maxlen);`	`string.h`	Copies up to `maxlen` characters from `src` into `dest`, truncating or null-padding `dest` (which might not be null-terminated).
`char *strpbrk(const char *s1, const char *s2);`	`string.h`	Returns a pointer to the first occurrence in `s1` of any of the characters in `s2` (or returns null).
`char *strrchr(const char *s, int c);`	`string.h`	Returns a pointer to the last occurrence of the character `c` in string `s` (or returns null).
`size_t strspn(const char *s1, const char *s2);`	`string.h`	Returns the length of the initial segment of `s1` that consists entirely of characters from `s2`.
`char *strstr(const char *s1, const char *s2);`	`string.h`	Scans `s1` for the first occurrence of the substring `s2`.
`double strtod(const char *s, char **endptr);`	`stdlib.h`	Converts string `s` to a `double` value; if `endptr` is not null, it sets `*endptr` to point to the character that stopped the scan.
`char *strtok(char *s1, const char *s2);`	`string.h`	Searches `s1` for tokens, which are separated by delimiters defined in `s2`.
`long int strtol(const char *s, char **endptr, int radix);`	`stdlib.h`	Converts a string `s` to a `long int` value in the given radix; if `endptr` is not null, it sets `*endptr` to point to the character that stopped the scan.
`unsigned long int strtoul (const char *s, char **endptr, int radix);`	`stdlib.h`	Converts a string `s` to an `unsigned long int` value in the given radix; if `endptr` is not null, it sets `*endptr` to point to the character that stopped the scan.
`size_t strxfrm(char *s1, const char *s2, size_t n);`	`string.h`	Transforms strings so that `strcmp` of new strings has the same result as `strcoll` of original strings. Changes up to `n` characters of `s1`.
`int system (const char *command);`	`stdlib.h`	Executes an operating system command.
`double tan(double x);`	`math.h`	Calculates the tangent of an angle specified in radians.
`double tanh(double x);`	`math.h`	Calculates the hyperbolic tangent.
`time_t time(time_t *timer);`	`time.h`	Gives the current time, in seconds, elapsed since 00:00:00 GMT, January 1, 1970, and stores that value in the location pointed to by `timer`.

Library Facilities Alphabetized by Name

Syntax	Header File	Purpose
`FILE *tmpfile(void);`	`stdio.h`	Creates a temporary binary file and opens it for update.
`char *tmpnam(char *s);`	`stdio.h`	Creates a unique file name.
`int tolower(int ch);`	`ctype.h`	Converts an integer `ch` to its lowercase value. Non-uppercase letter values are returned unchanged.
`int toupper(int ch);`	`ctype.h`	Converts an integer `ch` to its uppercase value. Non-lowercase letter values are returned unchanged.
`int ungetc(int c, FILE *stream);`	`stdio.h`	Pushes a character back into an open input stream.

`void va_start(va_list ap, lastfix);`
type `va_arg(va_list ap, type);`
`void va_end(va_list ap);`
	`stdarg.h`	Macros for implementing a variable argument list.

`int vfprintf(FILE *stream, const char *format, va_list arglist);`	`stdio.h`	Writes formatted output to a stream: Writes the values of a series of arguments, applying the format specifiers from the format string.
`int vprintf(const char *format, va_list arglist);`	`stdio.h`	Writes formatted output to `stdout`: Writes the values of a series of arguments, applying the format specifiers from the format string.
`int vsprintf(char *buffer, const char *format, va_list arglist);`	`stdio.h`	Writes formatted output to a string: Writes the values of a series of arguments, applying the format specifiers from the format string.
`size_t wcstombs(char *s, const wchar_t *pwcs, size_t n);`	`stdlib.h`	Converts a string of wide characters to a string of multibyte characters (changes no more than `n` bytes of `s`).
`int wctomb(char *s, wchar_t wchar);`	`stdlib.h`	Stores in `s` the multibyte representation of wide character `wchar`.

Skrivbara ASCII-tecken

I ASCII-tabellen nedan är inte teckenkoderna 0-31 medtagna, ty dessa är styrtecken och inte skrivbara i vanlig mening (se ASCII-styrkoder).

Vilka tecken som du kommer att se i nedanstående tabell med 8-bitars (s k utvidgad ASCII) beror faktiskt på vilket operativsystem och vilket verktyg du använder dig av. Det är nämligen så att det är enbart de "ursprungliga" 7-bitars ASCII-koderna, dvs tecknen med ASCII-koderna 0-127, som är gemensamma för alla s k *teckentabeller*. Resten (koderna 128-255) finns det många olika varianter av.

Om du tittar på ASCII-tabellen med en grafiskt baserad webbläsare, som Netscape eller Internet Explorer, då är det de tecken som ingår i *Latin-1* (ISO 8859-1) som du ser. Denna teckenuppsättning är idag också den vanligaste.

Om du i stället studerar HTML-koden från MS-DOS, då blir det tecknen som ingår i någon variant av IBM:s utvidgade ASCII som ses (bortsett från några tecken som har specialbeydelse i HTML och som därför angetts med specialkoder för att inte feltolkas, detta gäller t ex `&` för att visa &-tecknet).

Om du är riktigt uppmärksam (och använder en vanlig webbläsare), då kommer förmodligen en del tecken saknas i tabellen, eller rättare sagt det visas små rutor (eller frågetecken) i stället för vissa tecken. Detta beror på att teckenkoderna 128-159 inte används i *Latin-1* och 127 är en styrkod (och därför inte visbart).

För att ytterligare komplicerat till det: Det kan mycket väl hända att du ändå ser tecken som tillhör detta intervall. Detta gäller t ex Windows 98/NT/2000/XP, vilket beror på att Microsoft utökat *Latin 1* med dessa tecken. Detta gäller även Linux, om du använder nyare webbläsare (Netscape 6.x, Opera, Konqueror). Det där med att hålla sig till standarder är, som du förstår, inte så lätt.

kod	tkn	kod	tkn	kod	tkn	kod	tkn	kod	tkn	kod	tkn	kod	tkn
32		64	@	96	`	128	€	160		192	À	224	à
33	!	65	A	97	a	129	□	161	¡	193	Á	225	á
34	"	66	B	98	b	130	,	162	¢	194	Â	226	â
35	#	67	C	99	c	131	*f*	163	£	195	Ã	227	ã
36	$	68	D	100	d	132	„	164	¤	196	Ä	228	ä
37	%	69	E	101	e	133	...	165	¥	197	Å	229	å
38	&	70	F	102	f	134	†	166	¦	198	Æ	230	æ
39	'	71	G	103	g	135	‡	167	§	199	Ç	231	ç
40	(	72	H	104	h	136	ˆ	168	¨	200	È	232	è
41	)	73	I	105	i	137	‰	169	©	201	É	233	é
42	*	74	J	106	j	138	Š	170	ª	202	Ê	234	ê
43	+	75	K	107	k	139	‹	171	«	203	Ë	235	ë

44	,	76	L	108	l	140	Œ	172	¬	204	Ì	236	ì
45	-	77	M	109	m	141	□	173		205	Í	237	í
46	.	78	N	110	n	142	Ž	174	®	206	Î	238	î
47	/	79	O	111	o	143	□	175	¯	207	Ï	239	ï
48	0	80	P	112	p	144	□	176	°	208	Đ	240	ð
49	1	81	Q	113	q	145	‘	177	±	209	Ñ	241	ñ
50	2	82	R	114	r	146	’	178	²	210	Ò	242	ò
51	3	83	S	115	s	147	“	179	³	211	Ó	243	ó
52	4	84	T	116	t	148	”	180	´	212	Ô	244	ô
53	5	85	U	117	u	149		181	µ	213	Õ	245	õ
54	6	86	V	118	v	150	–	182	¶	214	Ö	246	ö
55	7	87	W	119	w	151	—	183	·	215	×	247	÷
56	8	88	X	120	x	152	˜	184	¸	216	Ø	248	ø
57	9	89	Y	121	y	153	™	185	¹	217	Ù	249	ù
58	:	90	Z	122	z	154	š	186	º	218	Ú	250	ú
59	;	91	[	123	{	155	›	187	»	219	Û	251	û
60	<	92	\	124	|	156	œ	188	¼	220	Ü	252	ü
61	=	93	]	125	}	157	□	189	½	221	Ý	253	ý
62	>	94	^	126	~	158	ž	190	¾	222	Þ	254	þ
63	?	95	_	127	□	159	Ÿ	191	¿	223	ß	255	ÿ

Skrivbara IBM-ASCII-tecken

Denna ASCII-tabell visar den IBM:s utvidgade ASCII (den flerspråkiga teckentabellen 850) som används i MS-DOS (de tecken som är specifika för IBM-ASCII är de med teckenkoder över 127).

Teckenkoderna 0-31 är inte medtagna, ty dessa är styrtecken och normalt inte skrivbara, även om åtskilliga av dem faktiskt tilldelats symboler i IBM-ASCII.

Anmärkning: För att DOS-tecknen ska visas korrekt med en vanlig webbläsare (som kan visa Latin 1, men inte IBM-ASCII), har de enskilda tecknen handkodats med hjälp av *Latin-1* (ISO 8859-1, som är standard i Windows och Unix). Korrekt återgivning av DOS-tecknen förutsätter därför att webbläsaren använder denna teckenuppsättning. De speciella grafiska tecknen i IBM-ASCII (hörn, sidor, block) saknar dock motsvarigheter i Latin 1, varför dessa inte kan visas. Däremot finns alla bokstäver (inkl grekiska) med.

kod	tkn	kod	tkn	kod	tkn	kod	tkn	kod	tkn	kod	tkn	kod	tkn
32		64	@	96	`	128	Ç	160	á	192	+	224	Ó
33	!	65	A	97	a	129	ü	161	í	193	-	225	ß
34	"	66	B	98	b	130	é	162	ó	194	-	226	Ô
35	#	67	C	99	c	131	â	163	ú	195	+	227	Ò
36	$	68	D	100	d	132	ä	164	ñ	196	-	228	õ
37	%	69	E	101	e	133	à	165	Ñ	197	+	229	Õ
38	&	70	F	102	f	134	å	166	ª	198	ã	230	µ
39	'	71	G	103	g	135	ç	167	º	199	Ã	231	þ
40	(	72	H	104	h	136	ê	168	¿	200	+	232	Þ
41	)	73	I	105	i	137	ë	169	®	201	+	233	Ú
42	*	74	J	106	j	138	è	170	¬	202	-	234	Û
43	+	75	K	107	k	139	ï	171	½	203	-	235	Ù
44	,	76	L	108	l	140	î	172	¼	204	¦	236	ý
45	-	77	M	109	m	141	ì	173	¡	205	-	237	Ý
46	.	78	N	110	n	142	Ä	174	«	206	+	238	¯
47	/	79	O	111	o	143	Å	175	»	207	¤	239	´
48	0	80	P	112	p	144	É	176	_	208	ð	240	
49	1	81	Q	113	q	145	æ	177	_	209	Ð	241	±
50	2	82	R	114	r	146	Æ	178	_	210	Ê	242	_
51	3	83	S	115	s	147	ô	179	¦	211	Ë	243	¾
52	4	84	T	116	t	148	ö	180	¦	212	È	244	¶

53	**5**	85	**U**	117	**u**	149	**ò**	181	**Á**	213	**i**	245	§
54	**6**	86	**V**	118	**v**	150	**û**	182	**Â**	214	**Í**	246	÷
55	**7**	87	**W**	119	**w**	151	**ù**	183	**À**	215	**Î**	247	¸
56	**8**	88	**X**	120	**x**	152	**ÿ**	184	**©**	216	**Ï**	248	°
57	**9**	89	**Y**	121	**y**	153	**Ö**	185	**¦**	217	**+**	249	¨
58	**:**	90	**Z**	122	**z**	154	**Ü**	186	**¦**	218	**+**	250	·
59	**;**	91	**[**	123	**{**	155	**ø**	187	**+**	219	**_**	251	_
60	**<**	92	****	124	**|**	156	**£**	188	**+**	220	**_**	252	³
61	**=**	93	**]**	125	**}**	157	**Ø**	189	**¢**	221	**_**	253	²
62	**>**	94	**^**	126	**~**	158	**×**	190	**¥**	222	**Ì**	254	_
63	**?**	95	**_**	127	□	159	***f***	191	**+**	223	**_**	255	

Svenska tecken i DOS som styrtecken i C

Å	**\217**
å	**\206**
Ä	**\216**
ä	**\204**
Ö	**\231**
ö	**\224**

The PC Keyboard

The computer keyboard produces codes that are associated with letters and symbols. One key, however, can produce a different set of codes when you press other keys at the same time. For example, the A key normally produces the letter "a" (ASCII code 97), but when pressed along with the SHIFT key, it produces the letter "A" (ASCII code 65). Two other keys—CTRL and ALT—produce even more codes!

Some keys, such as the function keys (F1 through F10), produce two codes: a scan code (≠0) and another code that indicates the key pressed.

The following table lists the keys on the PC keyboard and the codes they return.

Key	Normal		SHIFT		CTRL		ALT	
F1	0	59	0	84	0	94	0	104
F2	0	60	0	85	0	95	0	105
F3	0	61	0	86	0	96	0	106
F4	0	62	0	87	0	97	0	107
F5	0	63	0	88	0	98	0	108
F6	0	64	0	89	0	99	0	109
F7	0	65	0	90	0	100	0	110
F8	0	66	0	91	0	101	0	111
F9	0	67	0	92	0	102	0	112
F10	0	68	0	93	0	103	0	113
←	0	75		52	0	115		none
→	0	77		54	0	116		none
↑	0	72		56		none		none
↓	0	80		50		none		none
HOME	0	71		55	0	119		none
END	0	79		49	0	117		none
PGUP	0	73		57	0	132		none
PGDN	0	81		51	0	118		none
INS	0	82		48		none		none
DEL	0	83		46	0	255		none
ESC		27		27		27		none
BACKSPACE		8		8		127		none
TAB		9	0	15		none		none
ENTER		13		13		10		none
A		97		65		1	0	30
B		98		66		2	0	48
C		99		67		3	0	46
D		100		68		4	0	32
E		101		69		5	0	18
F		102		70		6	0	33
G		103		71		7	0	34
H		104		72		8	0	35
I		105		73		9	0	23
J		106		74		10	0	36
K		107		75		11	0	37

Key	Normal	SHIFT	CTRL	ALT
L	108	76	12	0 38
M	109	77	13	0 50
N	110	78	14	0 49
O	111	79	15	0 24
P	112	80	16	0 25
Q	113	81	17	0 16
R	114	82	18	0 19
S	115	83	19	0 31
T	116	84	20	0 20
U	117	85	21	0 22
V	118	86	22	0 47
W	119	87	23	0 17
X	120	88	24	0 45
Y	121	89	25	0 21
Z	122	90	26	0 44
[	91	123	27	none
\	92	124	28	none
]	93	125	29	none
`	96	126	none	none
0	48	41	none	0 129
1	49	33	none	0 120
2	50	64	0 3	0 121
3	51	35	none	0 122
4	52	36	none	0 123
5	53	37	none	0 124
6	54	94	30	0 125
7	55	38	none	0 126
8	56	42	none	0 127
9	57	40	none	0 128
*	42	none	0 114	none
Keypad +	43	43	none	none
Keypad -	45	45	none	none
=	61	43	none	0 131
/	47	63	none	none
;	59	58	none	none
-	45	95	31	0 130