Biblioteca standard C

CURS NR. 2/2

Biblioteca standard

API-ul bibliotecii standard C este declarat într-un set de fișiere header

- Fișierele header conțin declarări de funcții, definiții de tipuri și macrodefiniții

<assert.h>	Macrodefiniție care compară argumentul cu zero
<complex.h> (since C99)	Aritmetica numerelor complexe
<ctype.h>	Funcții care determină tipul caracterelor
<errno.h>	Macrouri care raportează condiții de eroare
<fenv.h> (since C99)	Controlul mediului pentru valori flotante
<float.h>	Limitele valorilor flotante
<inttypes.h> (since C99)	Conversii de format pentru tipuri întregi
<iso646.h> (since C95)	Moduri alternative de exprimare a conceptelor standard
<limits.h>	Dimensiunile tipurilor întregi
<locale.h>	Utilitare de localizare
<math.h>	Funcții matematice comune
<setjmp.h>	Salturi nelocale
<signal.h>	Tratarea semnalelor
<stdalign.h> (since C11)	Alinierea obiectelor
<stdarg.h>	Număr variabil de argumente
<stdatomic.h> (since C11)	Tipuri atomice
<stdbool.h> (since C99)	Tipul Boolean
<stddef.h>	Definiții de macrouri utilizate frecvent
<stdint.h> (since C99)	Tipuri întregi de dimnesiune specificată
<stdio.h>	Intrare / Ieșire
<stdlib.h>	Utilitare generale: gestiunea memoriei, conversie de string-uri, numere aleatoare, utilitare program
<stdnoreturn.h> (since C11)	Specificare funcțiilor care nu se returnează
<string.h>	Gestiunea string-urilor
<tgmath.h> (since C99)	Funcții matematice generice
<threads.h> (since C11)	Biblioteca de threaduri – gestiunea threadurilor multiple, a lacătelor și a variabilelor condiționale
<time.h>	Utilitare pentru timp / dată
<uchar.h> (since C11)	Utilitare pentru caractere Unicode UTF-16 și UTF-32
<wchar.h> (since C95)	Utilitare pentru caractere multibyte și caractere extinse (wide)
<wctype.h> (since C95)	Utilitare pentru clasificarea și maparea caracterelor extinse (wide)

Suport pentru gestiunea numerelor

Fișierul float.h

- Furnizează caracteristicile (ex. Domeniul de valori, precizia) tipurilor flotante – float, double și long double
  - Lista completă a macrourilor poate fi consultat aici

- Exemplu de utilizare a macrourilor din float.h și un posibil rezultat

```c
#include <stdio.h>
#include <float.h>
#include <math.h>

int main(void) {
   printf("FLT_RADIX     = %d\n", FLT_RADIX);
   printf("DECIMAL_DIG   = %d\n", DECIMAL_DIG);
   printf("FLT_MIN       = %e\n", FLT_MIN);
   printf("FLT_MAX       = %e\n", FLT_MAX);
   printf("FLT_EPSILON   = %e\n", FLT_EPSILON);
   printf("FLT_DIG       = %d\n", FLT_DIG);
   printf("FLT_MANT_DIG  = %d\n", FLT_MANT_DIG);
   printf("FLT_MIN_EXP   = %d\n", FLT_MIN_EXP);
   printf("FLT_MIN_10_EXP  = %d\n", FLT_MIN_10_EXP);
   printf("FLT_MAX_EXP     = %d\n", FLT_MAX_EXP);
   printf("FLT_MAX_10_EXP  = %d\n", FLT_MAX_10_EXP);
   printf("FLT_ROUNDS      = %d\n", FLT_ROUNDS);
   printf("FLT_EVAL_METHOD = %d\n", FLT_EVAL_METHOD);
   printf("FLT_HAS_SUBNORM = %d\n", FLT_HAS_SUBNORM);
}
```

```
FLT_RADIX    = 2
DECIMAL_DIG  = 17
FLT_MIN      = 1.175494e-38
FLT_MAX      = 3.402823e+38
FLT_EPSILON  = 1.192093e-07
FLT_DIG      = 6
FLT_MANT_DIG = 24
FLT_MIN_EXP  = -125
FLT_MIN_10_EXP  = -37
FLT_MAX_EXP     = 128
FLT_MAX_10_EXP  = 38
FLT_ROUNDS      = 1
FLT_EVAL_METHOD = 0
FLT_HAS_SUBNORM = 1
Press any key to continue . . .
```

Suport pentru gestiunea numerelor

Fișierul header limits.h

- Furnizează macrodefiniții pentru limitele tipurilor întregi (inclusiv caractere)
  - Lista completă a macrourilor poate fi consultat aici

- Exemplu de utilizare a macrourilor și un posibil rezultat

```c
#include <stdio.h>
#include <limits.h>

int main(void) {
   printf("CHAR_BIT   = %d\n", CHAR_BIT);
   printf("MB_LEN_MAX = %d\n\n", MB_LEN_MAX);

   printf("CHAR_MIN   = %+d\n", CHAR_MIN);
   printf("CHAR_MAX   = %+d\n", CHAR_MAX);
   printf("SCHAR_MIN  = %+d\n", SCHAR_MIN);
   printf("SCHAR_MAX  = %+d\n", SCHAR_MAX);
   printf("UCHAR_MAX  = %u\n\n", UCHAR_MAX);

   printf("SHRT_MIN   = %+d\n", SHRT_MIN);
   printf("SHRT_MAX   = %+d\n", SHRT_MAX);
   printf("USHRT_MAX  = %u\n\n", USHRT_MAX);

   printf("INT_MIN    = %+d\n", INT_MIN);
   printf("INT_MAX    = %+d\n", INT_MAX);
   printf("UINT_MAX   = %u\n\n", UINT_MAX);

   printf("LONG_MIN   = %+ld\n", LONG_MIN);
   printf("LONG_MAX   = %+ld\n", LONG_MAX);
   printf("ULONG_MAX  = %lu\n\n", ULONG_MAX);

   printf("LLONG_MIN  = %+lld\n", LLONG_MIN);
   printf("LLONG_MAX  = %+lld\n", LLONG_MAX);
   printf("ULLONG_MAX = %llu\n\n", ULLONG_MAX);
}
```

```
CHAR_BIT   = 8
MB_LEN_MAX = 5

CHAR_MIN   = -128
CHAR_MAX   = +127
SCHAR_MIN  = -128
SCHAR_MAX  = +127
UCHAR_MAX  = 255

SHRT_MIN   = -32768
SHRT_MAX   = +32767
USHRT_MAX  = 65535

INT_MIN    = -2147483648
INT_MAX    = +2147483647
UINT_MAX   = 4294967295

LONG_MIN   = -2147483648
LONG_MAX   = +2147483647
ULONG_MAX  = 4294967295

LLONG_MIN  = -9223372036854775808
LLONG_MAX  = +9223372036854775807
ULLONG_MAX = 18446744073709551615

Press any key to continue . . .
```

Suport pentru gestiunea numerelor

Fișierul header stdint.h

- Furnizează definiții de tipuri și macrouri pentru tipurile întregi de dimensiune specificată
  - Lista completă a macrourilor poate fi consultat aici
- Exemplu de utilizare a macrourilor și un posibil rezultat

```c
#include <stdio.h>
#include <stdint.h>
#include <wchar.h>

int main(void) {
   printf("PTRDIFF_MIN    = %td\n", PTRDIFF_MIN);
   printf("PTRDIFF_MAX    = %+td\n", PTRDIFF_MAX);
   printf("SIZE_MAX       = %zu\n", SIZE_MAX);
   printf("SIG_ATOMIC_MIN = %+jd\n", (intmax_t)SIG_ATOMIC_MIN);
   printf("SIG_ATOMIC_MAX = %+jd\n", (intmax_t)SIG_ATOMIC_MAX);
   printf("WCHAR_MIN      = %+jd\n", (intmax_t)WCHAR_MIN);
   printf("WCHAR_MAX      = %+jd\n", (intmax_t)WCHAR_MAX);
   printf("WINT_MIN       = %jd\n", (intmax_t)WINT_MIN);
   printf("WINT_MAX       = %jd\n", (intmax_t)WINT_MAX);
}
```

```
PTRDIFF_MIN    = -2147483648
PTRDIFF_MAX    = +2147483647
SIZE_MAX       = 4294967295
SIG_ATOMIC_MIN = -2147483648
SIG_ATOMIC_MAX = +2147483647
WCHAR_MIN      = +0
WCHAR_MAX      = +65535
WINT_MIN       = 0
WINT_MAX       = 65535
Press any key to continue . . .
```

Fixed width integer types (since C99)

int8_t int16_t int32_t int64_t	signed integer type with width of exactly 8, 16, 32 and 64 bits respectively with no padding bits and using 2's complement for negative values (provided only if the implementation directly supports the type)
int_fast8_t int_fast16_t int_fast32_t int_fast64_t	fastest signed integer type with width of at least 8, 16, 32 and 64 bits respectively
int_least8_t int_least16_t int_least32_t int_least64_t	smallest signed integer type with width of at least 8, 16, 32 and 64 bits respectively
intmax_t	maximum width integer type
intptr_t	integer type capable of holding a pointer
uint8_t uint16_t uint32_t uint64_t	unsigned integer type with width of exactly 8, 16, 32 and 64 bits respectively (provided only if the implementation directly supports the type)
uint_fast8_t uint_fast16_t uint_fast32_t uint_fast64_t	fastest unsigned integer type with width of at least 8, 16, 32 and 64 bits respectively
uint_least8_t uint_least16_t uint_least32_t uint_least64_t	smallest unsigned integer type with width of at least 8, 16, 32 and 64 bits respectively
uintmax_t	maximum width unsigned integer type
uintptr_t	unsigned integer type capable of holding a pointer

Suport pentru gestiunea numerelor

Fișierul header math.h

- Funcții matematice comune
  - funcții trigonometrice, funcții hiperbolice, funcții logaritmice și exponențiale,
  - funcții de putere, valoare absolută, restul împărțirii, etc.

- Lista completă a funcțiilor și macrourilor poate fi consultat aici

Power functions

pow
powf (C99)
powl (C99)

sqrt
sqrtf (C99)
sqrtl (C99)

cbrt (C99)
cbrtf (C99)
cbrtl (C99)

hypot (C99)
hypotf (C99)
hypotl (C99)

Trigonometric functions

sin
sinf (C99)
sinl (C99)

cos
cosf (C99)
cosl (C99)

tan
tanf (C99)
tanl (C99)

asin
asinf (C99)
asinl (C99)

acos
acosf (C99)
acosl (C99)

atan
atanf (C99)
atanl (C99)

atan2
atan2f (C99)
atan2l (C99)

Exponential functions

exp
expf (C99)
expl (C99)

exp2 (C99)
exp2f (C99)
exp2l (C99)

expm1 (C99)
expm1f (C99)
expm1l (C99)

log
logf (C99)
logl (C99)

log10
log10f (C99)
log10l (C99)

log2 (C99)
log2f (C99)
log2l (C99)

log1p (C99)
log1pf (C99)
log1pl (C99)

Classification and comparison

fpclassify (C99)

isfinite (C99)

isinf (C99)

isnan (C99)

isnormal (C99)

signbit (C99)

isgreater (C99)

isgreaterequal (C99)

isless (C99)

islessequal (C99)

islessgreater (C99)

isunordered (C99)

Nearest integer floating-point operations

ceil
ceilf (C99)
ceill (C99)

floor
floorf (C99)
floorl (C99)

trunc (C99)
truncf (C99)
truncl (C99)

round (C99)
lround (C99)
llround (C99)

nearbyint (C99)
nearbyintf (C99)
nearbyintl (C99)

rint (C99)
rintf (C99)
rintl (C99)
lrint (C99)
lrintf (C99)
lrintl (C99)
llrint (C99)
llrintf (C99)
llrintl (C99)

Basic operations

fabs
fabsf (C99)
fabsl (C99)

fmod
fmodf (C99)
fmodl (C99)

remainder (C99)
remainderf (C99)
remainderl (C99)

remquo (C99)
remquof (C99)
remquol (C99)

fma (C99)
fmaf (C99)
fmal (C99)

fmax (C99)
fmaxf (C99)
fmaxl (C99)

fmin (C99)
fminf (C99)
fminl (C99)

fdim (C99)
fdimf (C99)
fdiml (C99)

nan (C99)
nanf (C99)
nanl (C99)

Floating-point manipulation functions

frexp
frexpf (C99)
frexpl (C99)

ldexp
ldexpf (C99)
ldexpl (C99)

modf
modff (C99)
modfl (C99)

Suport pentru gestiunea caracterelor

Fișierul header ctype.h

- Furnizează două tipuri de funcții: de clasificare a caracterelor și de mapare între caractere majuscule-minuscule (toupper și tolower)
  - Lista completă a macrourilor poate fi consultat aici

- Exemplificarea modului de clasificare

ASCII values (hex)		characters	iscntrl iswcntrl	isprint iswprint	isspace iswspace	isblank iswblank	isgraph iswgraph	ispunct iswpunct	isalnum iswalnum	isalpha iswalpha	isupper iswupper	islower iswlower	isdigit iswdigit	isxdigit iswxdigit
0 - 8	0x00-0x08	control codes (NUL, etc.)	≠0	0	0	0	0	0	0	0	0	0	0	0
9	0x09	tab (\t)	≠0	0	≠0	≠0	0	0	0	0	0	0	0	0
10 - 13	0x0A-0x0D	whitespaces (\n,\v,\f,\r)	≠0	0	≠0	0	0	0	0	0	0	0	0	0
14 - 31	0x0E-0x1F	control codes	≠0	0	0	0	0	0	0	0	0	0	0	0
32	0x20	space	0	≠0	≠0	≠0	0	0	0	0	0	0	0	0
33 - 47	0x21-0x2F	!"#$%&'()*+,-./	0	≠0	0	0	≠0	≠0	0	0	0	0	0	0
48 - 57	0x30-0x39	0123456789	0	≠0	0	0	≠0	0	≠0	0	0	0	≠0	≠0
58 - 64	0x3a-0x40	:;<=>?@	0	≠0	0	0	≠0	≠0	0	0	0	0	0	0
65 - 70	0x41-0x46	ABCDEF	0	≠0	0	0	≠0	0	≠0	≠0	≠0	0	0	≠0
71 - 90	0x47-0x5A	GHIJKLMNOPQRSTUVWXYZ	0	≠0	0	0	≠0	0	≠0	≠0	≠0	0	0	0
91 - 96	0x5B-0x60	[\]^_`	0	≠0	0	0	≠0	≠0	0	0	0	0	0	0
97 -102	0x61-0x66	abcdef	0	≠0	0	0	≠0	0	≠0	≠0	0	≠0	0	≠0
103-122	0x67-0x7A	ghijklmnopqrstuvwxyz	0	≠0	0	0	≠0	0	≠0	≠0	0	≠0	0	0
123-126	0x7B-0x7E	{|}~	0	≠0	0	0	≠0	≠0	0	0	0	0	0	0
127	0x7F	backspace character (DEL)	≠0	0	0	0	0	0	0	0	0	0	0	0

Suport pentru gestiunea șirurilor de caractere

Fișierul header string.h

- Am studiat funcțiile furnizate
  - Lista completă a macrourilor poate fi consultată aici

- Exemplu de utilizare a macrourilor și un posibil rezultat

String manipulation

- strcpy
- strcpy_s (C11)
- strncpy
- strncpy_s (C11)
- strcat
- strcat_s (C11)
- strncat
- strncat_s (C11)
- strxfrm

String examination

- strlen
- strnlen_s (C11)
- strcmp
- strncmp
- strcoll
- strchr
- strrchr
- strspn
- strcspn
- strpbrk
- strstr
- strtok
- strtok_s (C11)

Character array manipulation

- memchr
- memcmp
- memset
- memset_s (C11)
- memcpy
- memcpy_s (C11)
- memmove
- memmove_s (C11)

Miscellaneous

- strerror
- strerror_s (C11)
- strerrorlen_s (C11)

```c
#define __STDC_WANT_LIB_EXT1__ 1
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

int main(void) {
   char str[] = "ghghghghghghghghghghgh";
   puts(str);
   memset(str, 'a', 5);
   puts(str);

   #ifdef __STDC_LIB_EXT1__
      set_constraint_handler_s(ignore_handler_s);
      int r = memset_s(str, sizeof str, 'b', 5);
      printf("str = \"%s\", r = %d\n", str, r);
      r = memset_s(str, 5, 'c', 10);   // count is greater than destsz
      printf("str = \"%s\", r = %d\n", str, r);
   #endif
}
```

```
ghghghghghghghghghghgh
aaaaahghghghghghghgh
str = "bbbbbhghghghghghghgh", r = 0
str = "ccccchghghghghghghgh", r = 22
```

Suport pentru caractere extinse

Fișierul header wchar.h și wctype

- Când setul de caractere ASCII sau Latin 1 nu mai este suficient
- Caracter extins (wide) este un întreg (2 octeți) care reprezintă un caracter,
  - Tipul wchar_t care este un unsigned short int

```
// A 16-bit character
wchar_t c = L'A';
// An array of 9 16-bit characters
// the 9th character is an 16-bit null character
wchar_t szBuffer[] = L"A String";
```

Types

Defined in header <wctype.h>

wchar_t	integer type that can hold any valid wide character(C++ keyword)
wint_t(C95)	integer type that can hold any valid wide character and at least one more value

String manipulation

Defined in header <wchar.h>

wcscpy (C95) wcscpy_s (C11)	copies one wide string to another (function)
wcsncpy (C95) wcsncpy_s (C11)	copies a certain amount of wide characters from one string to another (function)
wcscat (C95) wcscat_s (C11)	appends a copy of one wide string to another (function)
wcsncat (C95) wcsncat_s (C11)	appends a certain amount of wide characters from one wide string to another (function)
wcsxfrm(C95)	transform a wide string so that wcscmp would produce the same result as wcscoll (function)

Conversions to numeric formats

Defined in header <wchar.h>

wcstol (C95) wcstoll(C99)	converts a wide string to an integer value (function)
wcstoul (C95) wcstoull(C99)	converts a wide string to an unsigned integer value (function)
wcstof (C99) wcstod (C95) wcstold(C99)	converts a wide string to a floating point value (function)

Defined in header <inttypes.h>

wcstoimax(C99) wcstoumax(C99)	converts a wide string to `intmax_t` or `uintmax_t` (function)

String examination

wcslen (C95) wcsnlen_s (C11)	
wcscmp (C95)	
wcsncmp (C95)	
wcscoll (C95)	
wcschr (C95)	
wcsrchr (C95)	
wcsspn (C95)	
wcscspn (C95)	
wcspbrk (C95)	
wcsstr (C95)	
wcstok (C95) wcstok_s (C11)	

wmemcpy (C95) wmemcpy_s (C11)	
wmemmove (C95) wmemmove_s (C11)	
wmemcmp (C95)	
wmemchr (C95)	
wmemset (C95)	

Suport pentru caractere extinse

Fișierul header wchar.h și wctype

- Când setul de caractere ASCII sau Latin 1 nu mai este suficient
- Caracter extins (wide) este un întreg (2 octeți) care reprezintă un caracter,
  - Tipul wchar_t care este un unsigned short int

```
// A 16-bit character
wchar_t c = L'A';
// An array of 9 16-bit characters
// the 9th character is an 16-bit null character
wchar_t szBuffer[] = L"A String";
```

wscanf (C95) **fwscanf** (C95) **swscanf** (C95) **wscanf_s** (C11) **fwscanf_s** (C11) **swscanf_s** (C11)	reads formatted wide character input from stdin, a file stream or a buffer (function)
fgetwc **getwc** (C95)	gets a wide character from a file stream (function)
fputws (C95)	writes a wide string to a file stream (function)

```
#include <wchar.h>
#include <stdio.h>

int main(void) {
   wchar_t input[] = L"A bird came down the walk";
   printf("Parsing the input string '%ls'\n", input);
   wchar_t *buffer;
   wchar_t *token = wcstok(input, L" ", &buffer);
   while (token) {
      printf("%ls\n", token);
      token = wcstok(NULL, L" ", &buffer);
   }

   printf("Contents of the input string now: '");
   for (size_t n = 0; n < sizeof input / sizeof *input; ++n)
      input[n] ? printf("%lc", input[n]) : printf("\\0");
   puts("'");
}
```

```
Parsing the input string 'A bird came down the walk'
A
bird
came
down
the
walk
Contents of the input string now: 'A\0bird\0came\0down\0the\0walk\0'
Press any key to continue . . .
```

Funcții utilitare

Fișierul header stdlib.h

- Furnizează multe funcții utilitare din categoriile:
  - Funcții de conversie numerică
  - Funcții de generare a secvențelor de numere pseudoaleatoare
  - Funcții de gestiune a memoriei,
  - Funcții de comunicare cu mediul
  - Utilitare de sortare și căutare
  - Funcții de conversie multibyte / (șiruri de) caractere extinse
- Lista completă a funcțiilor poate fi consultată aici și aici

C memory management library

`malloc`	allocates memory (function)
`calloc`	allocates and zeroes memory (function)
`realloc`	expands previously allocated memory block (function)
`free`	deallocates previously allocated memory (function)
`aligned_alloc` (C11)	allocates aligned memory (function)

Communicating with the environment

`system`	calls the host environment's command processor (function)
`getenv` `getenv_s` (C11)	access to the list of environment variables (function)

Program termination

The following functions manage program termination and resource cleanup.

`abort`	causes abnormal program termination (without cleaning up) (function)
`exit`	causes normal program termination with cleaning up (function)
`quick_exit` (C11)	causes normal program termination without completely cleaning up (function)
`_Exit` (C99)	causes normal program termination without cleaning up (function)
`atexit`	registers a function to be called on `exit()` invocation (function)
`at_quick_exit` (C11)	registers a function to be called on `quick_exit` invocation (function)
`EXIT_SUCCESS` `EXIT_FAILURE`	indicates program execution execution status (macro constant)

Funcții utilitare

Fișierul header stdlib.h

- Furnizează multe funcții utilitare din categoriile:
  - Funcții de conversie numerică
  - Funcții de generare a secvențelor de numere pseudoaleatoare
  - Funcții de gestiune a memoriei,
  - Funcții de comunicare cu mediul
  - Utilitare de sortare și căutare
  - Funcții de conversie multibyte / (șiruri de) caractere extinse
- Lista completă a funcțiilor poate fi consultată aici

Conversions to numeric formats

`atof`	converts a byte string to a floating point value (function)

`atoi`
`atol`
`atoll` (C99)
	converts a byte string to an integer value (function)

`strtol` `strtoll` (C99)	converts a byte string to an integer value (function)
`strtoul` `strtoull` (C99)	converts a byte string to an unsigned integer value (function)
`strtof` (C99) `strtod` `strtold` (C99)	converts a byte string to a floating point value (function)
Defined in header <inttypes.h>	
`strtoimax` (C99) `strtoumax` (C99)	converts a byte string to `intmax_t` or `uintmax_t` (function)

- Exemplu de utilizare a macrourilor și un posibil rezultat

```c
#include <stdio.h>
#include <stdlib.h>

int main(void) {
   printf("%i\n", atoi(" -123junk"));
   printf("%i\n", atoi("junk"));          // no conversion can be performed
   printf("%i\n", atoi("2147483648"));    // UB: out of range of integer type
}
```

```
-123
0
2147483647
Press any key to continue . . .
```

Funcții utilitare

Fișierul header stdlib.h

- Furnizează multe funcții utilitare din categoriile:
  - Funcții de conversie numerică
  - Funcții de generare a secvențelor de numere pseudoaleatoare
  - Funcții de gestiune a memoriei,
  - Funcții de comunicare cu mediul
  - Utilitare de sortare și căutare
  - Funcții de conversie multibyte / (șiruri de) caractere extinse

- Lista completă a funcțiilor poate fi consultată aici

Multibyte/wide character conversions

`mblen`	returns the number of bytes in the next multibyte character (function)
`mbtowc`	converts the next multibyte character to wide character (function)
`wctomb` `wctomb_s` (C11)	converts a wide character to its multibyte representation (function)
`mbstowcs` `mbstowcs_s` (C11)	converts a narrow multibyte character string to wide string (function)
`wcstombs` `wcstombs_s` (C11)	converts a wide string to narrow multibyte character string (function)
Defined in header <wchar.h>	
`mbsinit` (C95)	checks if the mbstate_t object represents initial shift state (function)
`btowc` (C95)	widens a single-byte narrow character to wide character, if possible (function)
`wctob` (C95)	narrows a wide character to a single-byte narrow character, if possible (function)
`mbrlen` (C95)	returns the number of bytes in the next multibyte character, given state (function)
`mbrtowc` (C95)	converts the next multibyte character to wide character, given state (function)
`wcrtomb` (C95) `wcrtomb_s` (C11)	converts a wide character to its multibyte representation, given state (function)
`mbsrtowcs` (C95) `mbsrtowcs_s` (C11)	converts a narrow multibyte character string to wide string, given state (function)
`wcsrtombs` (C95) `wcsrtombs_s` (C11)	converts a wide string to narrow multibyte character string, given state (function)
Defined in header <uchar.h>	
`mbrtoc16` (C11)	generate the next 16-bit wide character from a narrow multibyte string (function)
`c16rtomb` (C11)	convert a 16-bit wide character to narrow multibyte string (function)
`mbrtoc32` (C11)	generate the next 32-bit wide character from a narrow multibyte string (function)
`c32rtomb` (C11)	convert a 32-bit wide character to narrow multibyte string (function)

Funcții utilitare

Fișierul header stdlib.h

- Furnizează multe funcții utilitare din categoriile:
  - Funcții de conversie numerică
  - Funcții de generare a secvențelor de numere pseudoaleatoare
  - Funcții de gestiune a memoriei,
  - Funcții de comunicare cu mediul
  - Utilitare de sortare și căutare
  - Funcții de conversie multibyte / (șiruri de) caractere extinse
- Lista completă a funcțiilor poate fi consultată aici

Pseudo-random number generation

rand	generates a pseudo-random number (function)
srand	seeds pseudo-random number generator (function)
RAND_MAX	maximum possible value generated by rand() (macro constant)

- Exemplu de utilizare și un posibil rezultat

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main(void)
{
srand(time(0)); //use current time as seed for random generator
int random_variable = rand();
printf("Random value on [0,%d]: %d\n", RAND_MAX, random_variable);
}
```

```
Random value on [0,32767]: 25354
Press any key to continue . . .
```

Funcții utilitare

Fișierul header assert.h

- Furnizează macrodefiniția assert
  - Definiția depinde de un alt macrou NDEBUG, dar care nu este definit de biblioteca standard
    - Dacă NDEBUG este definit, atunci assert nu are nici un efect
    - Dacă NDEBUG nu este definit, atunci macroul assert verifică dacă argumentul (expresie scalară) primit este egal cu zero
      - dacă da, afișează un mesaj de diagnostic (specific implementării) și apelează abort()

```
#ifdef NDEBUG
#define assert(condition) ((void)0)
#else
#define assert(condition) /*implementation defined*/

#endif
```

- Mai multe despre macrodefiniția assert găsim aici

- Exemplu de utilizare și un posibil rezultat

```
#include <stdio.h>
// uncomment to disable assert()
//#define NDEBUG
#include <assert.h>
#include <math.h>

int main(void) {
   double x = -1.0;
   assert(x >= 0.0);
   printf("sqrt(x) = %f\n", sqrt(x));

   return 0;
}
```

output with NDEBUG not defined:
a.out: main.cpp:10: main: Assertion `x >= 0.0' failed.

output with NDEBUG defined:
sqrt(x) = -nan

Funcții utilitare pentru dată și timp

Fișierul header time.h

- Furnizează funcții și macrodefiniții pentru gestiunea timpului
- Mai multe despre macrodefinița assert găsim aici

Time manipulation

Defined in header <time.h>

difftime	computes the difference between times (function)
time	returns the current calendar time of the system as time since epoch (function)
clock	returns raw processor clock time since the program is started (function)
timespec_get (since C11)	returns the calendar time based on a given time base (function)

Format conversions

Defined in header <time.h>

asctime asctime_s (C11)	converts a tm object to a textual representation (function)
ctime ctime_s (C11)	converts a time_t object to a textual representation (function)
strftime	converts a tm object to custom textual representation (function)
Defined in header <wchar.h>	
wcsftime (C95)	converts a tm object to custom wide string textual representation (function)
Defined in header <time.h>	
gmtime gmtime_s (C11)	converts time since epoch to calendar time expressed as Coordinated Universal Time (UTC) (function)
localtime localtime_s (C11)	converts time since epoch to calendar time expressed as local time (function)
mktime	converts calendar time to time since epoch (function)

```c
#include <stdio.h>
#include <time.h>
#include <stdint.h>

int main(void) {
   time_t result = time(NULL);
   if (result != -1)
      printf("The current time is %s(%ju seconds since the Epoch)\n",
               asctime(gmtime(&result)), (uintmax_t)result);
}
```

```
The current time is Sat Sep 10 08:41:52 2016
(1473496912 seconds since the Epoch)
Press any key to continue . . .
```

```c
#include <time.h>
#include <stdio.h>

int main(void) {
   time_t t = time(NULL);
   struct tm tm = *localtime(&t);
   printf("%s", asctime(&tm));
}
```

```
Sat Sep 10 10:48:10 2016
Press any key to continue . . .
```

Funcții utilitare pentru dată și timp

Fișierul header time.h

- Furnizează funcții și macrodefiniții pentru gestiunea timpului
- Mai multe despre macrodefinița assert găsim aici

Constants

Defined in header <time.h>

CLOCKS_PER_SEC	number of processor clock ticks per second (macro constant)

Types

Defined in header <time.h>

tm	calendar time type (struct)
time_t	calendar time since epoch type (typedef)
clock_t	processor time since era type (typedef)
timespec (since C11)	time in seconds and nanoseconds (struct)

Member objects

int tm_sec	seconds after the minute – [0, 61](until C99) / [0, 60] (since C99)[note 1]
int tm_min	minutes after the hour – [0, 59]
int tm_hour	hours since midnight – [0, 23]
int tm_mday	day of the month – [1, 31]
int tm_mon	months since January – [0, 11]
int tm_year	years since 1900
int tm_wday	days since Sunday – [0, 6]
int tm_yday	days since January 1 – [0, 365]
int tm_isdst	Daylight Saving Time flag. The value is positive if DST is in effect, zero if not and negative if no information is available

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

// Pauses for a specified number of milliseconds.
void sleep(clock_t wait) {
   clock_t goal;
   goal = wait + clock();
   while (goal > clock())
      ;
}

int main(void) {
   long    i = 60000000L;
   clock_t start, finish;
   double  duration;

   // Delay for a specified time.
   printf("Delay for three seconds\n");
   sleep((clock_t)3 * CLOCKS_PER_SEC);
   printf("Done!\n");

   // Measure the duration of an event.
   printf("Time to do %ld empty loops is ", i);
   start = clock();
   while (i--)
      ;
   finish = clock();
   duration = (double)(finish - start) / CLOCKS_PER_SEC;
   printf("%2.1f seconds\n", duration);
}
```

```
Delay for three seconds
Done!
Time to do 60000000 empty loops is 0.2 seconds
Press any key to continue . . .
```

Surse bibliografice

- Pagina de referință pentru standardul C, http://en.cppreference.com/w/c

- K. N. King, C Programming – A Modern Approach, 2nd edition, W. W. Norton & Co., 2008
  - Capitolele 23 - 27