stdlib.h

Methods

int rand();	
Parameters	
void	
Returns	
int	Pseudo-random integer value between 0 and RAND_MAX, inclusive.

void srand(unsigned int seed);	
Parameters	
unsigned int seed	the seed value
Returns	
void	

void *malloc(size_t size);	
Parameters	
size_t size	number of bytes to allocate
Returns	
void *	Pointer to the beginning of newly allocated memory.

void free(void *ptr);	
Parameters	
void *ptr	Pointer to the memory to deallocate
Returns	
void	

int atoi(const char *str);	
Parameters	
const char *str	The string representation of an integral number.
Returns	
int	Int of str or 0 if no conversion could be performed

int sizeof(type);	
Parameters	
type	The type to get the size of
Returns	
int	Size of the type

stdio.h

Types

stdin	Standard input stream (console)
FILE	File type for file streams
EOF	Integer constant expression of type int and negative value

Methods

FILE *fopen(const char *filename, const char *mode);	
Parameters	
const char *filename	file name to associate the file stream to
const char *mode	string determining file access mode. r, w, a
Returns	
FILE*	If successful, a pointer to the file, else NULL pointer.

int fscanf(FILE *stream, const char *format, ...);	
Parameters	
FILE *stream	Pointer to the file to read from
const char *format	String defining the format of the data to read
…	Arguments for the format string
Returns	
int	

int fprintf(FILE *stream, const char *format, ...);	
Parameters	
FILE *stream	Pointer to the file to read from
const char *format	String defining the format of the data to write
…	Arguments for the format string
Returns	
int	

int getc(FILE *stream);	
Parameters	
FILE *stream	Pointer to the file to read from
Returns	
int	The character being read converted to int

char *fgets(char *str, int count, FILE *stream);	
Parameters	
char *str	Pointer to a char array
int count	maximum number of characters to write
FILE *stream	Pointer to the file to read from
Returns	
char*	str on success

int fclose(FILE *stream);	
Parameters	
FILE *stream	Pointer to the file to read from
Returns	
int	0 on success, EOF otherwise

int feof(FILE *stream);	
Parameters	
FILE *stream	Pointer to the file to read from
Returns	
int	Nonzero value if the end of the stream has been reached, otherwise 0

int printf(const char *format, ...);	
Parameters	
const char *format	String defining the format of the data to print
Returns	
int	

int scanf(const char *format, ...);	
Parameters	
const char *format	String defining the format of the data to read
Returns	
int	

string.h

Methods

char *strcpy(char *dest, const char *src);	
Parameters	
char *dest	Pointer to the character array to write to
const char *src	pointer to the character array to copy from
Returns	
char *	Copy of dest pointer

char *strcat(char *dest, const char *src);	
Parameters	
char *dest	Pointer to the character array to append to
const char *src	pointer to the character array to sppend from
Returns	
char *	Copy of dest pointer

size_t strlen(const char *str);	
Parameters	
const char *str	Pointer to the string to be examined
Returns	
size_t	The length of the string, including '\0'

char *strstr(const char *str, const char *substr);	
Parameters	
const char *str	Pointer to the string to be examined
const char *substr	Pointer to the string to search for
Returns	
char *	Pointer to the first character of substr in str or NULL if not found

ctype.h

Methods

int toupper(int ch);	
Parameters	
int	Character to be converted
Returns	
int	Uppercase version of ch

int isdigit(int ch);	
Parameters	
int ch	Character to classify
Returns	
int	Non-zero value if the character is a numeric character, zero otherwise.

int isalpha(int ch);	
Parameters	
int ch	Character to classify
Returns	
int	Non-zero value if the character is a numeric character, zero otherwise.

LesData.h

Methods

char lesChar(const char* t);	
Parameters	
const char * t	Ledetekst til brukeren når ber om ett tegn
Returns	
char	Ett (upcaset) tegn.

float lesFloat(const char* t, const float min, const float max);	
Parameters	
const char * t	Ledetekst til brukeren når ber om input/et tall
const float min	Minimum for innlest og godtatt tallverdi
const float max	Maksimum for innlest og godtatt tallverdi
Returns	
float	Godtatt verdi i intervallet 'min' - 'max'

int lesInt(const char* t, const int min, const int max);	
Parameters	
const char * t	Ledetekst til brukeren når ber om input/et tall
const int min	Minimum for innlest og godtatt tallverdi
const int max	Maksimum for innlest og godtatt tallverdi
Returns	
int	Godtatt verdi i intervallet 'min' - 'max'

void lesText(const char* t, char* tekst, const int len);	
Parameters	
const char * t	Ledetekst til brukeren når ber om input/et tall
char* tekst	Peker til memoryområdet med char'er
const int len	Max. lengde på innlest tekst
Returns	
void	

char* lagOgLesText(const char* t);	
Parameters	
const char * t	Peker til ledetekst om hva som skal leses inn
Returns	
char*	Peker til nyopprettet og datafylt tekst

math.h

Methods

double sqrt(double arg);	
Parameters	
double arg	Value to find square root of
Returns	
double	Square root of arg

bool.h

Types

bool	Boolean value (1 or 0)

time.h

Types

struct tm	Standard input stream (console)
int tm_sec	seconds after the minute – [0, 61]
int tm_min	Minutes after the hour – [0, 59]
int tm_hour	Minutes after the hour – [0, 59]
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
time_t	This is a type suitable for storing the calendar time.

Methods

time_t time(time_t *arg);	
Parameters	
time_t *arg	Pointer to a time_t object where the time will be stored, or a null pointer.
Returns	
time_t	Current calendar time encoded as time_t object. If arg is not a null pointer, the return value is also stored in the object pointed to by arg.

struct tm *gmtime (const time_t *timer);	
Parameters	
const time_t *timer	Pointer to a time_t object to convert
Returns	
struct tm*	Pointer to a static internal tm object on success, or null pointer otherwise. (UTC)

struct tm *localtime (const time_t *timer);	
Parameters	
const time_t *timer	Pointer to a time_t object to convert
Returns	
struct tm*	Pointer to a static internal tm object on success, or null pointer otherwise. (Local time)

char* asctime(const struct tm* time_ptr);	
Parameters	
const struct tm* time_ptr	Pointer to a time_t object to convert
Returns	
char*	Format: <weekday month dayofmonth hh:mm:ss yyyy>