CSC215
Lecture

Standard Library in C

Outline

- Introduction
- stdio.h
- stdlib.h
- ctype.h
- stdarg.h
- math.h
- String.h
- assert.h
- errno.h
- time.h

Introduction

- Standard libraries:
  - type definitions
  - variable declarations
  - constant and macro definitions
  - Functions

- Description and usage information can be obtained from `man` pages on unix-like OS or the web
  - section 3
  - on unix and unix-like OS: in the terminal type: `man [<section>] <library_function_name>⏎`
  - Many websites host copies of the man pages: Die, HE , MAN7 , …

- List of standard library header files:

`assert.h` `ctype.h` `errno.h` `float.h` `limits.h` `locale.h` `math.h` `setjmp.h`
`signal.h` `stdarg.h` `stddef.h` `stdio.h` `stdlib.h` `string.h` `time.h`

Library stdio.h

- Types
  - size_t
  - FILE

- Constants
  - NULL
  - EOF
  - SEEK_CUR
    SEEK_END
    SEEK_SET
  - stderr
    stdin
    stdout

- Functions
  - FILE *fopen(const char *, const char *)
  - int fclose(FILE *)
  - int fflush(FILE *)

  - int getchar(void)
  - char *gets(char *)
  - int scanf(const char *, ...)
  - int putchar(int char)
  - int puts(const char *)
  - int printf(const char *, ...)

  - int fgetc(FILE *)
  - int ungetc(int char, FILE *stream)
  - char *fgets(char *, int , FILE *)
  - int fscanf(FILE *, const char *, ...)
  - int fputc(int, FILE *)
  - int fputs(const char *, FILE *)
  - int fprintf(FILE *, const char *, ...)

Library `stdio.h`

- Functions (cont.)
  - `size_t fread(void *, size_t, size_t, FILE *)`
  - `size_t fwrite(const void *, size_t, size_t, FILE *)`
    `----------------------------------------------------`
  - `void rewind(FILE *)`
  - `int fseek(FILE *, long int, int)`
  - `int fgetpos(FILE *, fpos_t *)`
  - `int fsetpos(FILE *, const fpos_t *)`
  - `long int ftell(FILE *)`
  - `----------------------------------------------------`
  - `int remove(const char *)`
  - `int rename(const char *, const char *)`
    `----------------------------------------------------`

Library stdlib.h

- Types
  - size_t

- Constants
  - NULL
  - EXIT_FAILURE EXIT_SUCCESS
  - RAND_MAX

- Functions (cont.)
  - void *malloc(size_t)
  - void *calloc(size_t, size_t)
  - void *realloc(void *, size_t)
  - void free(void *)

  - double atof(const char *)
  - int atoi(const char *)
  - long int atol(const char *)
  - double strtod(const char *, char **)
  - long int strtol(const char *, char **, int)
  - unsigned long int strtoul(const char *, char **, int)

  - void abort(void)
  - void exit(int)
  - int atexit(void (*func)(void))
  - int system(const char *string)

  - int abs(int x)
  - long int labs(long int x)

Library `stdlib.h`

- Functions (cont.)
  - `int rand(void)`
  - `void srand(unsigned int seed)`

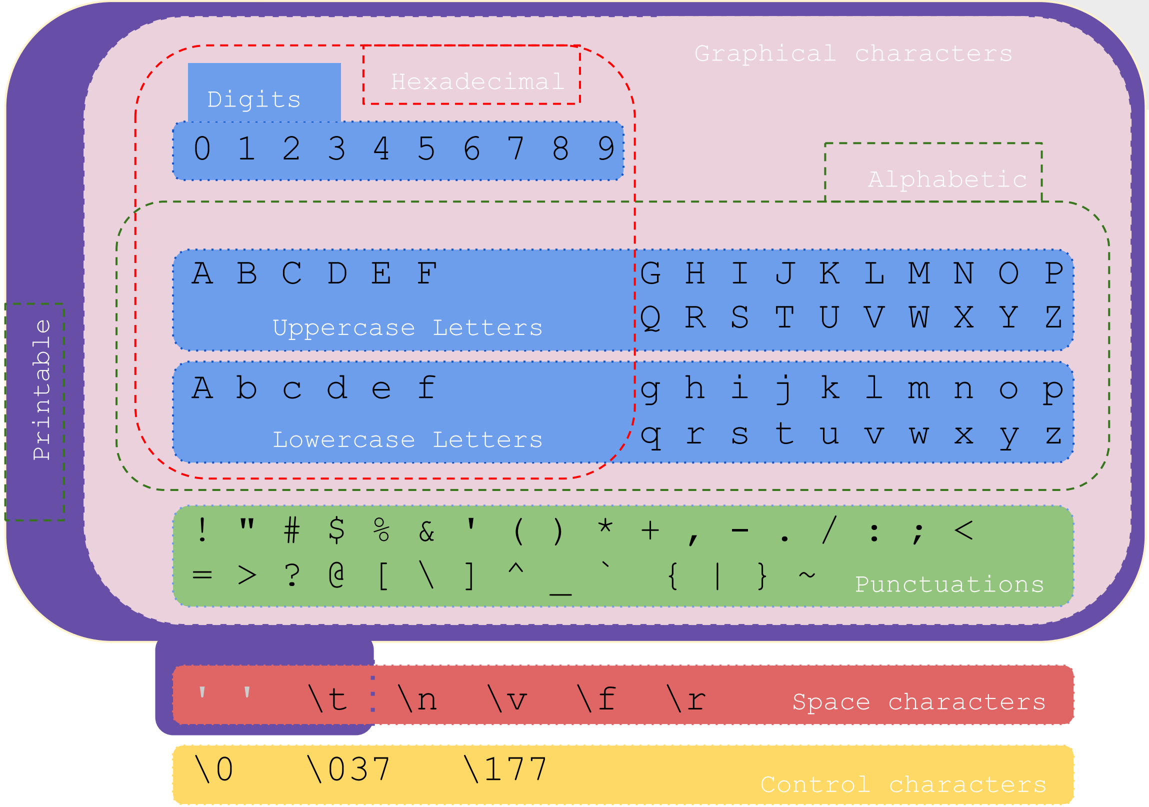
    `-----------------------------------------------------`
  - `void *bsearch(const void *, const void *, size_t, size_t,`

    `int (*compar)(const void *, const void *))`
  - `void qsort(void *, size_t, size_t, int (*compar)(const void *, const void*))`

Library ctype.h

- Functions (cont.)
  - int isalnum(int c)
  - int isalpha(int c)
  - int iscntrl(int c)
  - int isdigit(int c)
  - int isgraph(int c)
  - int islower(int c)
  - int isprint(int c)
  - int ispunct(int c)
  - int isspace(int c)
  - int isupper(int c)
  - int isxdigit(int c)

  - int tolower(int c)
  - int toupper(int c)



Graphical characters

Digits

Hexadecimal

0 1 2 3 4 5 6 7 8 9

Alphabetic

A B C D E F G H I J K L M N O P

Q R S T U V W X Y Z

Uppercase Letters

A b c d e f g h i j k l m n o p

q r s t u v w x y z

Lowercase Letters

Printable

! " # $ % & ' ( ) * + , - . / : ; <

= > ? @ [\ ] ^ _ ` { | } ~

Punctuations

' ' \t \n \v \f \r

Space characters

\0 \037 \177

Control characters

Library `stdarg.h`

- Types
  - `va_list`

- Macros
  - `void va_start(va_list, last_arg)`
  - `type va_arg(va_list, type)`
  - `void va_end(va_list)`
  - `void va_copy(va_list, va_list)`

Library `math.h`

❖ Arithmetic functions

`double fabs(double )`	$|x|$
`double ceil(double )`	$\lceil x \rceil$
`double floor(double )`	$\lfloor x \rfloor$
`double fmod(double , double )`	$x\%y$
`double modf(double , double *)`	$x-\lfloor x \rfloor$, $\lfloor x \rfloor$

❖ Exponential functions

`double pow(double , double )`	x^y
`double sqrt(double )`	$\sqrt{x}$
`double exp(double )`	e^x
`double ldexp(double , int )`	$x.2^y$
`double log(double )`	$\log_e x$
`double log10(double )`	$\log_{10} x$

❖ Trigonometric functions

`double sin(double )`	$\sin(x)$
`double cos(double )`	$\cos(x)$
`double asin(double )`	$\sin^{-1}(x)$
`double acos(double )`	$\cos^{-1}(x)$
`double atan(double )`	$\tan^{-1}(x)$

❑ All functions take and yields double precision floating point values.

❑ Trigonometric functions deals with input and output angles in radians.

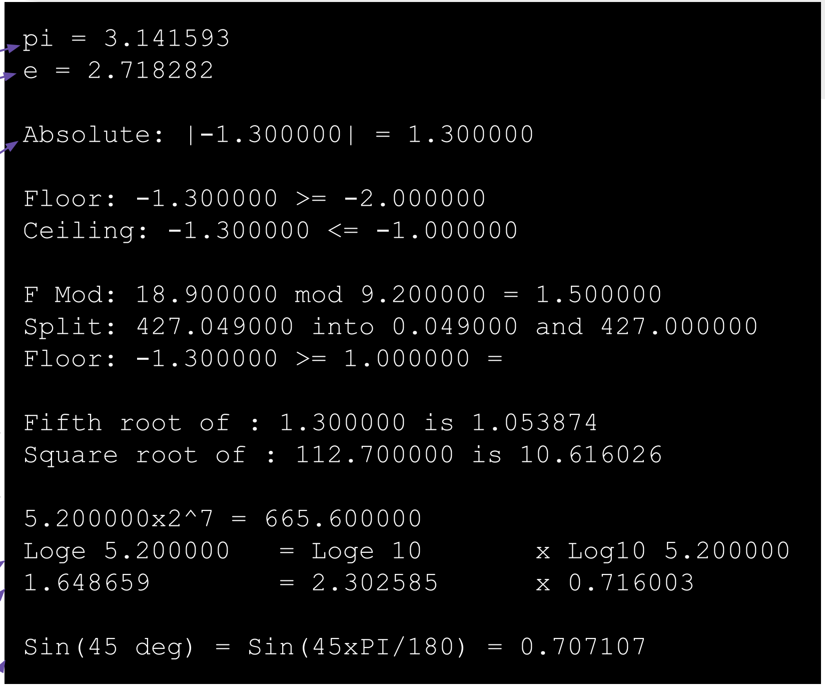
Example

```c
#include <stdio.h>
#include <math.h>
const double PI = acos(-1);
const double E = exp(1);
int main(){
  double buf;
  printf("pi = %f\n", PI);
  printf("e = %f\n", E);

  printf("Absolute: |%f| = %f \n", -1.3, fabs(-1.3));
  printf("Floor: %f >= %f\n", -1.3, floor(-1.3));
  printf("Ceiling: %f <= %f\n", -1.3, ceil(-1.3));
  printf("F Mod: %f mod %f = %f\n", 18.9, 9.2, fmod(19.9, 9.2));
  printf("Split: %f into %f and ", 427.049, modf(427.049, &buf));
  printf("%f\n", buf);
  printf("Floor: %f >= %f = \n", -1.3, floor(1.0/3+1.0/3+1.0/3));

  printf("Fifth root of : %f is %f\n", 1.3, pow(1.3, 1.0/5));
  printf("Square root of : %f is %f\n", 112.7, sqrt(112.7));
  printf("%fx2^%d = %f\n", 5.2, 7, ldexp(5.2, 7));
  printf("Loge %f\t= Loge 10\tx Log10 %f\n", 5.2, 5.2);
  printf("%f\t= %f\tx %f\n", log(5.2), log(10), log10(5.2));

  printf("Sin(%d deg) = Sin(%dxPI/180 rad) = %f\n", 45, 45, sin(45*PI/180));
  return 0;
}
```



```
pi = 3.141593
e = 2.718282

Absolute: |-1.300000| = 1.300000

Floor: -1.300000 >= -2.000000
Ceiling: -1.300000 <= -1.000000

F Mod: 18.900000 mod 9.200000 = 1.500000
Split: 427.049000 into 0.049000 and 427.000000
Floor: -1.300000 >= 1.000000 =

Fifth root of : 1.300000 is 1.053874
Square root of : 112.700000 is 10.616026

5.200000x2^7 = 665.600000
Loge 5.200000   = Loge 10       x Log10 5.200000
1.648659        = 2.302585      x 0.716003

Sin(45 deg) = Sin(45xPI/180) = 0.707107
```

Library `string.h`

- Memory functions
  - `int memcmp(const void *, const void *, size_t)`
  - `void *memchr(const void *, int, size_t)`
  - `void *memcpy(void *, const void *, size_t)`
  - `void *memset(void *, int, size_t )`
- String functions
  - `size_t strlen(const char *)`
  - `char *strcat(char *, const char *)`
  - `char *strncat(char *, const char *, size_t )`
  - `char *strcpy(char *, const char *)`
  - `char *strncpy(char *, const char *, size_t )`
  - `int strcmp(const char *, const char *)`
  - `int strncmp(const char *, const char *, size_t )`
  - `char *strchr(const char *, int)`
  - `char *strrchr(const char *, int)`
  - `char *strstr(const char *, const char *)`
  - `char *strpbrk(const char *, const char *)`
  - `size_t strspn(const char *, const char *)`
  - `size_t strcspn(const char *, const char *)`
  - `char *strtok(char *, const char *)`

❑ In coping functions, the first parameter is the destination and the second is the source.

❑ In search functions, first parameter is the haystack (text) and the second is the needle (pattern).

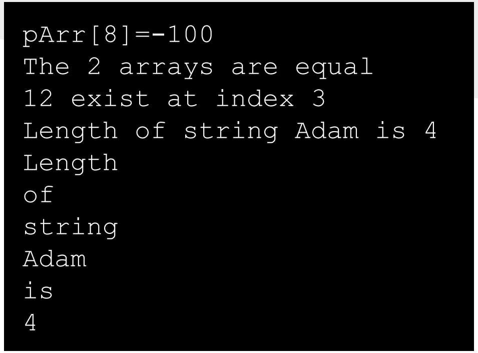
Example

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

int main(){
  int* pArr = (int*)malloc(10*sizeof(int));
  char sentence[255], *word;
  memset(pArr, -100, 10*sizeof(int));
  printf("pArr[8]=%d\n", (char)pArr[8]);
  int iArr[] = {-3, 5, 0, 12, -8, 27};
  memcpy(pArr, iArr, 6*sizeof(int));
  printf("The 2 arrays are%s equal\n",memcmp(pArr,iArr,6*sizeof(int))?" not":"");
  int* ind = (int*)memchr(pArr, 12, 6*sizeof(int));
  printf("%d exist at index %d\n", 12, (int)(ind-pArr));

  char* name = "Adam";
  printf("Length of string %s is %d\n", name, (int)strlen(name));
  sprintf(sentence, "Length of string %s is %d\n", name, (int)strlen(name));
  word = strtok(sentence, ", ");
  do {
    printf("%s\n", word);
  } while (word = strtok(NULL, ", "));

  return 0;
}
```



```
pArr[8]=-100
The 2 arrays are equal
12 exist at index 3
Length of string Adam is 4
Length
of
string
Adam
is
4
```

Libraries: assert.h, errno.h and time.h

- Macro of assert.h
  - `void assert(int expression)`
- Macro of errno.h
  - `extern int errno`
- time.h
  - `clock_t`
  - `time_t`
  - `struct tm`
- Functions of time.h
  - `clock_t clock()`
  - `time_t time(time_t*)`
  - `double difftime(time_t, time_t)`
  - `time_t mktime(struct tm*)`
  - `char* asctime(const struct tm*)`
  - `char* ctime(const time_t)`
  - `struct tm* gmtime(const time_t)`
  - `struct tm* localtime(const time_t )`
  - `size_t strftime(char* , size_t , const char* , const struct tm* )`

```
struct tm {
  int tm_sec;      /* Seconds (0-60) */
  int tm_min;      /* Minutes (0-59) */
  int tm_hour;     /* Hours (0-23) */
  int tm_mday;     /* Day of the month (1-31) */
  int tm_mon;      /* Month (0-11) */
  int tm_year;     /* Year - 1900 */
  int tm_wday;     /* Day of week (0-6, Sunday=0) */
  int tm_yday;     /* Day in year (0-365,1 Jan=0) */
  int tm_isdst;    /* Daylight saving time */
};
```

```
Unix time epoch:
1970, Jan, 1 00:00:00 UTC
```

Example

```
#include <stdio.h>
#include <time.h>
int main(){
  time_t rawtime;
  struct tm * timeinfo;
  char buffer [80];

  time(&rawtime);
  printf("%s\n", ctime(&rawtime));

  timeinfo = localtime(&rawtime);
  printf("%s\n", asctime(timeinfo));

  strftime(buffer,80,"Now it's %y/%m/%d.",timeinfo);
  puts(buffer);
  strftime(buffer,80,"Now it's %Y/%m/%d.",timeinfo);
  puts(buffer);
  return 0;
}
```

```
Tue Apr 18 04:55:50 2017
Tue Apr 18 04:55:50 2017
Now it's 17/04/18.
Now it's 2017/04/18.
```

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

int main(){
  clock_t start_t, end_t;
  float total_t;
  int i;
  start_t = clock();
  printf("Starting @ start_t = %ld\n", start_t);
  printf("Run a big loop\n", start_t);
  for(i=0; i< 10000000; i++) { }
  end_t = clock();
  printf("Ending @ end_t = %ld\n", end_t);
  total_t=1000*(float)(end_t-start_t)/CLOCKS_PER_SEC;
  printf("Total CPU time: %f ms\n",total_t );
  return(0);
}
```

```
Starting @ start_t = 7865
Run a big loop
Ending @ end_t = 7915
Total CPU time: 0.050000 ms
```