

# Outline for May 29, 2002

**Handouts:** *Homework 5***Reading:** Johnsonbaugh and Kalin, pp. 679–702 (appendix of useful functions)

---

1. Greetings and felicitations!
2. Standard I/O Library (#include <stdio.h>)
  - a. open file: fopen
  - b. unstructured read/write: getchar, fgetc (getc), putchar, fputc (putc)
  - c. formatted read/write: fgets (gets), fscanf, fputs (puts), fprintf
  - d. structured read/write: fread, fwrite
  - e. random access: fseek, ftell, rewind
  - f. close file: fclose
  - g. miscellaneous: feof, ferror, clearerr
3. Character types and conversions (#include <ctype.h>)
  - a. alphabetic, numeric, alphanumeric: isalnum, isdigit, isalpha
  - b. upper, lower, and conversions: isupper, islower, toupper, tolower
  - c. types of chars: iscntrl, isgraph (not blank, printable), isprint (printable), ispunct, isspace
4. String conversion (#include <stdlib.h>)
  - a. string to number: atoi, atof, atol
5. String functions (#include <string.h>)
  - a. compare: strcmp, strncmp, strcasecmp, strncasecmp; memcmp
  - b. copy: strcpy, strncpy; memcpy (no overlap), memmove (overlap okay)
  - c. find character: strchr (index), strrchr (rindex), strpbrk; memchr
  - d. length: strlen
6. Memory management (#include <stdlib.h>)
  - a. Allocation: malloc, calloc
  - b. Release: free, cfree (deprecated)
  - c. Reallocation: realloc()
7. Miscellaneous
  - a. terminate program (exit); include <stdlib.h>
  - b. sort array of data (qsort); include <stdlib.h>
  - c. time of day (time, ctime); include <time.h>
  - d. execute ccommand (system); include <stdlib.h>
8. Debugging
  - a. programs have bugs; find and fix them
  - b. static debugging: insert debugging code into source, recompile and run
  - c. dynamic debugging: look at the program as it runs, observing (and maybe changing) variables, etc.
9. Static debugging
  - a. using printf to print variable values; mention %p (prints pointer value, usually as a hex integer)
  - b. using printf to print where you are (ie, on function entry printf("in function\n"));
  - c. #ifdef DEBUG ... #endif around the printf's so you can leave them in the source if you need them again

- d. `assert(x)` macro: `assert(0 <= i && i <= n)` causes program to exit with error message if `(0 <= I && I <= n)` is false; must include `<assert.h>`. To delete, say `#define NDEBUG` and they will not be in the compiled code.

10. Dynamic debugging

- a. debugging tool instruments executable program so it can be stopped, examined, altered, and continued interactively
- b. go through the handout
- c. mention the “where” command which shows you the program stack