# CS262 Lecture 03
# Chapter 4 Functions

Zoran Duric
Department of Computer Science

# array

- very similar to java but there are some important differences
- **int tmp[3];**
- **int tmp[3]={0,0,0};**
- **const int tmp[]={0,0,0};**
- **tmp** is a pointer to the first element of the array
- cannot use variable to define the size of the array
  - ex: you cannot say: **int x=10; double tmp[x];**
  - this won't compile, but you can say:
  - **#define x 10**, then **double tmp[x];**

# char array (string)

- c has no string (class)
- a string is simply a “char array” with the last element being null (‘\0’)
- ex:
  - char msg[]=”hello\n”;


  - char msg[7]=”hello\n”; //size 7 or larger

# function

- call-by-value
  - the arguments are local variables whose values are copied from the callers
  - each function call allocates all these local variables which are placed on the top of the **call stack**
  - **ex:** long ans=fib(n); //in ex4.c
    - variable n in main function and variable n in fib function are different variables even though they have the same value.
  - **ex:** void swap(int a, int b); //won;t work
  - void swap(int * a, int * b); //need to use pointers

# function

- Since array variables are pointers so:
  - char A[]="GMU", B[]="UMD";
  - swap(A,B); //call by value
  - void swap(int X[], int Y[]){...}
    - X will have address A
    - Y will have address B

- java is also "call-by-value" and "references" (i.e. pointers) are passed when arguments are objects
  - so, java does have pointers (references), but you cannot manipulate them

# scopes

- scopes
  - life span (global, local)
  - visibility (static, extern)
- Life span
  - variables **outside** all functions are global variables (has life span of the program)
  - variables **inside** a function is local to a **function call** (does not span different calls) unless "static" is used
    - **int foo(){ static int x=0; printf("x=%d",x++); }**
    - call foo multiple times will output different values

# scopes

- see
  - static.c

# scopes

- Visibility (for global variables)
  - similar to private, protected, public in java/c++
  - **static** means “only visible to the file contains that variable”
  - **extern** means “visible to the entire program”
    - this is default for all global variables

# scopes

- see
  - longest-line-2.c
  - longest-line-3 (dir)

# typedef and call-back functions

- see call-back.c

# typedef and call-back functions

- see call-back.c

# variadic functions

- a function that take arbitrary number of arguments
- ex:in c, it can have this prototype:
  - int foo(char * format, int size, ...);
  - there must be one fixed parameter
  - there is "..." to indicate the rest of variables
- macros in stdarg.h are used to retrieve the rest of arguments
- there is also variadic marco for the same purpose

# variadic functions

- see varags-full.c