



CMSC 106 Lecture Set #7

Set Started:
Tuesday, October 6, 2010



Function Philosophy

- Tracing with the Call Stack
- Modularity
- Encapsulation
- Reusability
- Modifiability



Parameter Passing Details

- Parameters must match in position and type
 - except automatic promotion is allowed
 - if there is a mismatch – compilation error
- `void f1(float x, y);`
 - `y` is an integer type parameter by default
 - bad style
- All arguments are **passed by value**



Introduction to Scope

- local variables
 - are declared within a pair of curly braces
 - are only available from the time they are declared until the end of those curly braces
- parameters
 - are initialized by their corresponding argument (get a copy of that value)
 - are only available in the function for which they are a parameter



Swapping values example

- `void swap1(int, int);`
- `void swap2(int*, int*);`
- `void swap3(int*, int*);`



Passing a Reference

- address = reference = where it is located in memory
- `&`
 - before variable name indicates its address rather than its contents
 - `scanf("%d",&a);`
- `*`
 - after a type name in a declaration, it means it is storing an address to something of that type
 - `int *p;`
 - before a variable name in lines that are not declarations of variables, it means to follow that pointer as a map to find the actual variable
 - `printf("%d\n",*p);`



Variables: Scoping Rules and Storage Classes

- Scopes
 - Where the variable is visible
 - Options
 - local scope
 - global scope/file scope
- Storage Classes
 - Where and how long the variable remains in existence
 - Options
 - automatic
 - register
 - static
 - extern



Random Number Generation

- provided in <stdlib.h>
 - function: rand
 - function: srand
 - constant: RAND_MAX
 - terminology: random, pseudorandom, seed
- function provided in <time.h>
 - time



Character input and output

- in <stdio.h>
 - function: getchar
 - function: putchar
 - constant: EOF
- Using them in Loops