EASTERN WASHINGTON UNIVERSITY
start something big

C Conditional Statement & I/O

Computer Science Department
Eastern Washington University
Yun Tian (Tony) Ph.D.

Last Class

- Operators
- Bitwise Operators
- Operator Precedence

Topic for today

- If statement
- Basic I/O

Arithmetic Operators

- The C if statements have the same usage as in Java.
- The C loop statement have the same usage as in Java.

If Statements

If demo1 and if demo 2

Switch Statements

switch demo1 and demo 2

Basic I/O

- int getchar(void)
- This function reads the next available character from the keyboard and returns it as an integer.
- This function reads only single character at a time.
- You can use this method in the loop in case you want to read more than one characters from the keyboard.

Basic I/O

- int getchar(void)
  - When use getchar() in a loop, be careful.
  - Since you press **Enter** key to send your input character to your program,
  - The Enter key is left in the input buffer after getchar consumes one char in the buffer.
  - Enter key will be the input character for the next iteration.
  - DEMO of the Problem!!!!

Basic I/O

- int putchar(int c)
- This function puts the passed character on the screen and returns the same character.
- This function puts only single character at a time.
- You can use this method in the loop in case you want to display more than one character on the screen

Basic I/O

- The char *gets(char *s) function reads a line from stdin into the buffer pointed to by **s** until either a terminating newline or EOF.
  - The parameter **s** could be an character array.
- The int puts(const char *s) function writes the string **s** and a trailing newline to stdout.
- http://www.tutorialspoint.com/c_standard_library/

Basic I/O

- The int scanf(const char *format, ...) function reads input from the standard input stream stdin and scans that input according to **format** provided.
- The int printf(const char *format, ...) function writes output to the standard output stream stdout and produces output according to a **format** provided.

Basic I/O

- The **format** can be a simple constant string, but you can specify %s, %d, %c, %f, etc., to print or read strings, integer, character or float respectively.
- E.g.
- printf("The temperature is %10.2f.", temp);
- printf("your name is %8s," name);

Basic I/O

- It should be noted that scanf() expect input in the same format as you provided %s and %d, which means you have to provide valid input like "string integer",
  - If you provide "string string" or "integer integer" then it will be assumed as wrong input.
- While reading a string scanf() stops reading as soon as it encounters a **space**, so "this is test" are three strings for scanf().

Basic I/O

- Demo of scanf()
- Scanf with %c to input single character has the same behavior as getchar(), with new line feed left in the input buffer,
  - Cause trouble for next iteration of input in loops
- Scanf with %d to input integer without any issues.
- Scanf with %s will NOT include into target string the linefeed at the end of the input buffer, but it stops at the first white space.

Basic I/O

- Demo of scanf()
- But gets and fgets will include the new line feed into the target string.

Summary

- If
- Switch
- Basic I/O