CS 137 General notes

**Printing**

To print a statement:

* printf(string, argument(s));
* printf(“one \n two \n three”);
  + one
  + two
  + three
* must have first argument be a string

Argument(s)

* can use %d, %s, …
* will give them values using the arguments
* eg:

1. int a = 3;
2. Printf(“there are %d days”, a);

**Math functions:**

* Some functions include (), %, /, +, -, \*
* There are NO exponent functions in C
* Performed in last order of operations
* When there is a value assigned (like a = 3\*2-10), the right side will be completed first

1. a = b = 10
2. the b = 10 will be completed first, then a = b

* be careful, some operations will not work as expected
* eg:

1. a \*= b + c
2. could be:
3. a = a\*b + c
4. or
5. a = a\*(b + c)
6. a += 2
7. a =+ 2
8. one is a = a + 2
9. two is a = positive 2

Incremental Operators:

1. int a = 10
2. int b = 0
3. ++a
4. --a
5. b = ++a
6. a becomes 12 then b becomes a
7. b = a++
8. b becomes a then it will increment by one, b becomes 1
9. if a = 3, b = 1
10. b += a+++2
11. a+2 is done first, then a++, so a becomes 5, b becomes 6

* prefix:
* ++a will increment a by one and then possibly use a
* --a will increment a by -1 and then possibly use a
* suffix:
* a++ will USE a first then increment by 1
* the variable of a is only changed once processes are done

1. int main ( void ){
2. int x = 10;
3. int y = printf ( " % d % d \ n " , printf ( " % d " ,x ) , x \*4);
4. //x will be 40, printf returns 2 (2 characters)
5. //second printf returns 5, (40 2 \n), \n is a character
6. printf ( " % d \ n " , y );
7. return 0;
8. }
9. x is set at 10
10. y will be the number of chars printed by outside printf
11. inside printf will be printed, and return its number of chars (prints 10\_, returns 3)
12. second argument is 40 (x\*4)
13. y will print 3\_40\n, return 5
14. final line will print 5 and the new line char
15. x is set at 10
16. printf will print the statement and return the value of the number of characters

Associativity:

- will go left to right unless there is an equal sign, where it will move right to left

Relational and Logical Operators:

* Relational operators ( <,>,<=,...)
* take lower precedence from arithmetic and are left associative
* Equality operators (!=) are lower than relational, left associative
* Logical operators (!)
* like !(q<=b)right associative
* || and && are used as **and** and **or** and are left associative and lower than equality
* will return 1 for true and 0 for false

|  |  |  |  |
| --- | --- | --- | --- |
| p | q | p||q | p&&q |
| T | T | T | T |
| F | F | F | F |
| T | F | T | F |
| F | T | T | F |

1. int a = 0;
2. a != 0 && 4/a > 2;
3. this can be used as a condition to avoid generating an error (4/0)
4. returns false on the left side and both have to be true, so keeps program smooth