ECE 177 – Programming I: From C Foundations to Hardware Interaction Lecture 5

Vince Weaver

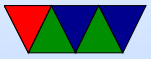
https://web.eece.maine.edu/~vweaver

vincent.weaver@maine.edu

2 February 2026

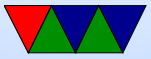
Announcements

- Reminder, we are taking attendance
- Homeworks, working on getting that set up



Lab #1 Update

- I know there are people who still don't have VS Code and/or WSL set up. Keep working on it and try to get checked off this week at the start of lab
- Sorry if Brightspace is harassing you with notifications, I'll try to turn them off



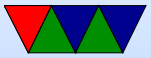
Question From Last Time

- When would you stick L on end to force size of constants?
- Usually when mixing different data sizes
- This can be important when dealing with C promoting types. So if you do something like

```
signed char y;
int x = 127 * y;
```

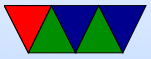
and y is 10 then this might do math at 8 bit, overflow get truncated, *then* upgrade to int, when you really wanted int all along

```
signed char y;
```



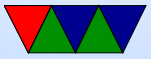
```
int x=127L * y;
```

this would force the math at 32-bits so no truncation



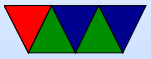
Basic Math

- You have an amazing computer, with billions of traditions, running at 3+GHz (billions of cycles per second)
- Gigaflops of floating point performance
- An unfathomable amount of computer power
- I'm going to tell you how to do basic arithmetic with it



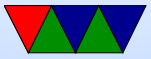
Addition

- Just use +
- `a=b+c;`
- Note the destination is on the left



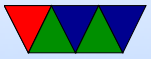
Subtraction

- Just use -
- `a=b-c;`



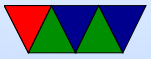
Multiplication

- Just use *
- `a=b*c;`



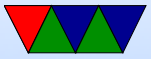
Division

- Just use /
- `a=b/c;`
- Have to be careful with integers
  - Tricky – what happens with something like 5/2? Can't represent 2.5 as an integer.
  - C doesn't round, just truncates, so get 2. Have to be careful as that might not be what you want



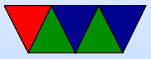
Remainder / Modulus

- Just use %
- `a=b%c;`
- Complications: what happens with negative numbers? That's a bit beyond this class



Exponentiation

- You can't easily do this in C!
- It is NOT `a=b^c;` (that's xor we'll learn later)
- There is an `exp()` function we might talk about later



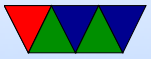
Can Combined to Make Arbitrarily Complicated Expressions

- `x=((a+b)*c)/d;`
- What if you have something like this?

  `y=z*q+w/t*3-4*f`

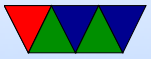
  What order do things happen in?

  (also as an aside, please use parenthesis in cases like this just to make it more clear what's going on without having to think about it forever)



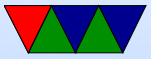
Order of Operations

- Like algebra class PEMDAS
- This will get *extremely* complicated later :(
- For now:
  - Multiply/Divide are equal, highest priority
  - Add/Subtract are equal, lowest priority
  - You can use parenthesis to group things together
  - Evaluation is left to right



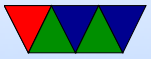
Unary operators

- A single - means make a number negative, `x=-y;`
- A single + means something is positive `x=+y;`
- A single * is special involving pointers, we'll discuss later



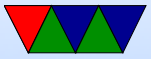
Shortcuts you'll see

- `x=x+1;` can be shortened to `x+=1;`
- `y=y-1;` can be shortened to `y-=1;`
- `z=z*4;` can be shortened to `z*=4;`
- and so forth, this works with most operators
- (yes, to those looking ahead, you can also special case increment with `x++;`, we'll get to that)



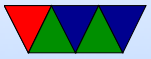
Show off a 30-year old C Program by Me

- Seabattle
- Surprisingly people still play it occasionally? (FreeBSD?)
- `http://www.deater.net/weave/vmwprod/seabattle.html`



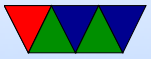
Deconstructions Hello World

```c
/* Hello World Example */

#include <stdio.h>

int main(int argc, char **argv) {

    int x=0;

    printf("Hello World!\n");
    printf("%d\n",x*4);

    return 0;
}
```



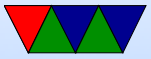
Pound Sign (Hash Tag? Number Sigh? Octothorp)?

- Invokes pre-processor which we will talk more about later, cpp can do many things
- Here we are using `#include` which includes another file
  - Including a .h or "header" file
  - If filename in quotes it means look in local search path
  - If filename in angle brackets it means look in system files (/usr/include on Linux)
  - Can include headers that describe interfaces, pre-



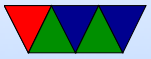
declare functions and values

- Can also include code but usually frowned upon in C, and can cause trouble if include twice



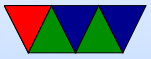
Functions

- Function is a routine that you can transfer program control to
- You can pass along values (parameters)
- It will do something, then can optionally return one value back to program
- C lets you ignore the return value though often it is good to check this, sometimes it reports errors that way
- All C programs have at least one function, `main()` which is the entry point (aside, lot goes on before it gets here



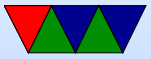
behind scenes)

- Like variables, must declare function before you can call it
- Declare it first with the return value (void if none) Then after name, in () have parameters you can pass in
- If you want no parameters technically you need to put void but you can get away with nothing



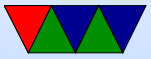
Function Block

- Code can be split up into chunks called blocks
- Curly braces indicate a block
- We'll get into more details later, but each functions contains at least one block



printf() function

- We will talk more about later
- Where does it live? C library
- It is magic in that we'll see it can take a variable number of arguments (of varying types)
- In addition to printing messages, can print the values of constants and variables



Return

- Returns value from a function
- For `main()` this gets passed back to OS
- You can see this on Linux with echo `$?` at command line