ECE 177 – Programming I: From C Foundations to Hardware Interaction Lecture 8

Vince Weaver

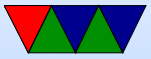
https://web.eece.maine.edu/~vweaver

vincent.weaver@maine.edu

9 February 2026

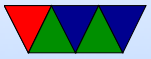
Announcements

- Homework #1 is in codelab, due Wednesday!



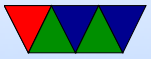
Lab Update

- Monday Lab: do Lab #2 today
- Rest of this week: makeup lab. You only need to attend if you still need to check off a lab or if you still need help getting VS Code going
- Next week: no Lab Monday (President's day) so Monday lab is still going to be behind. Everyone else will do Lab #3



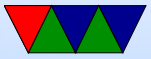
Code Lab Update

- No need to write entire C program, just what it asks for
- If you get stuck, feel free to e-mail me. Tell me which problem you are having but I can probably tell as it shows me which ones you are having trouble with and your most recent answer



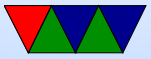
Conditionals

- We showed straight line code (just executing statements in a block in order)
- We also showed loops (executing statements in a block but optionally repeating the block multiple times)
- What if we want to conditionally execute one block or another based on some sort of condition?



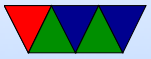
Conditional Example

```
int i,x;

x=3;          // set x to 3

if (i==1) { // if i is equal to 1 run this block
    x=4;     // set x to 4
}         // if i was not equal to 1 execution picks up
          // after end of block
```



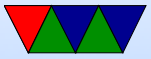
Two way Conditional Example

```
int i,x;

x=3;

if (i==1) {    // if i equals 1
    x=4;   // set x to 4
}
if (i!=1) {    // if i not equal to 1
    x=5;   // set x to 5
}
```



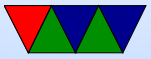
Instead use “else”

```
int i,x;

x=3;

if (i==1) { // if i equals 1, x=4
     x=4;
} else {     // otherwise if i not equal to 1, x=5
     x=5;
}
```

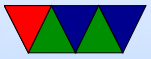


Chained if/else Example

```
#include <stdio.h>

int main (int argc, char **argv) {
    int x;
    for(x = -10; x < 10; x = x + 1) {
        if (x < -5) {
            printf("%d is less than -5\n", x);
        } else if (x < 0) {
            printf("%d is between -5 and 0\n", x);
        } else if (x < 5) {
            printf("%d is between 0 and 5\n", x);
        } else {
            printf("%d is greater than or equal to 5\n", x);
        }
    }
```

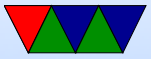


```
    return 0;
}
```



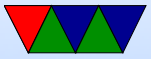
Indenting

- Indentation, inside of a block usually you should indent (either with tabs or spaces). This makes it easier to tell what block you are in
- Modern C compiler might warn these days if it thinks you have confusing indentation



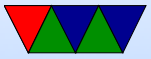
Single Statement Blocks

- As with loops, if you only have one statement after a if you can have it stand alone, no need for a block.
- I strongly discourage this because it's easy to make a mistake if/when you decide to add an additional statement



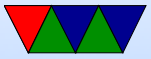
Values of Expressions in C

- This is sort of advanced/confusing but it comes up a lot
- In C you can have an expression like `x==3`
- You can think of this as a boolean value, meaning it can be either True or False
- Note that traditional C does not have a boolean type, you can't declare something a boolean
- Instead in C:
  - 0 is false
  - anything else is true



stdbool

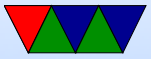
- C99 did add a stdbool.h header
- This defines true as 1 and false as 0
- They are still ints though, I don't think it's a compiler enforced type



Some boolean examples

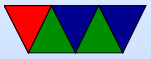
- 1 means true. So things like `while(1)` is an infinite loop, because 1 is always true
- Expressions like `i==3` will be 1 if true, 0 if false
- This means you can do weird things like

```
printf("%d\n",i==3);
```



Expression Shortcuts

- You can also do some shortcuts. You can just do `if (i)` and this is short for if i is non-zero `if (i!=0)`
- You can use the ! to do a boolean (not bitwise) not.
- So `if (!i)` means is short for `if (i==0)`

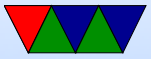


Forcing 1/0

- What if you have a value and you want to convert it to 1 if non-zero and leave if at 0 if it is zero?

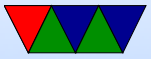
```
x=5;
y=!!x;
```

- There is a lot of advanced C tricks you can do with this knowledge but for this class probably best to not do fancy tricks



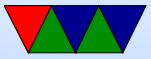
Value of Not-comparison Expressions

- Every expression in C has a value, even if not a comparison
- If you do something like x=4, this sets x equal to 4 but also the whole expression has the value 4
- This can cause issues



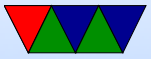
Being Careful With Equal Signs

```
i=0;
/* what if you meant to do */
if (i==1) printf("i is 1\n");
/* but instead you did */
if (i=1) printf("i is 1\n");

/* the latter doesn't check if i is 1, but
   sets i to 1, and 1 means true. so it will
   always do the printf and modify i */
/* the compiler will warn you */
/* If you wanted to do this for some reason,
    put in extra parenthesis */
if ((i=1)) printf("i is 1\n");
```



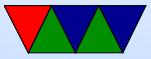
Logical/Boolean Operations

- We already saw that C, 0 is false and anything else is true
- These are the logical operators useful when doing comparisons
  - `!` logical not
  - `&&` logical and
  - `||` logical or
- Note these are different than bitwise boolean operations that we'll learn about soon



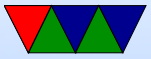
Logical AND Example

```
if (x==5) {
     if (y==3) {
          // do something
     }
}
/* can instead be
if ((x==5) && (y==3)) {
     // do something
}
/* read it as "if x is equal to 5 AND y is equal to 3" */
```



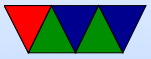
Logical OR Example

```
if (x==5) {
    // do something
}
if (y==3) {
    // do the same thing
}
/* can instead be
if ((x==5) || (y==3)) {
    // do something
}
/* read it as "if x is equal to 5 OR y is equal to 3" */
```



Comparison Shortcutting

- In some cases C will know a condition isn't true early
- It can "short-cut" and not complete evaluating the expression
- For example `if ((x==1) || (y==2))` if x is 1 then we know the whole thing will end up true, there's no need to evaluate anything else
- This is good for performance, but might be surprising if you are trying to be overly fancy and doing actual calculations in later conditions



Showed off TB1

- "Tom Bombem: Invasion of the Inanimate Objects"
- `http://www.deater.net/weave/vmwprod/tb1/tb1.htm`
- Game I wrote a *long* time ago in Pascal, converted to C
- Still compiles despite being old (A nice feature of C's stability)