ECE 177 – Programming I: From C Foundations to Hardware Interaction Lecture 21

Vince Weaver

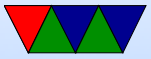
https://web.eece.maine.edu/~vweaver

vincent.weaver@maine.edu

25 March 2026

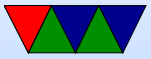
Announcements

- Advising talk by Elizabeth Bushnel
- Don't forget HW#5
- Adding additional HW questions about bit-manipulation/masking difficult.
- Hand back midterms at end of class
  please wait at least a day before requesting regrades



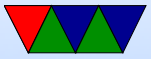
Lab#6 Update

- Many of the issues are nesting your loops properly
- Indent them! Makes it easier to catch things like that

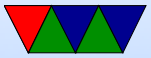


HW5 Extra Info

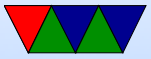
- Don't write entire programs or functions unless it asks you to
- Don't include extra chunks of code like `return 0`
- You can try testing things out on your own machine if it makes debugging easier. Codelab probably tries to block cut and paste though mostly to try to keep you from using AI.
- Remember strings in C are case sensitive, so if asks for `"Hello"` don't use `"hello"`



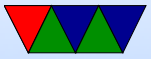
- Practice a bit with printf. If it wants two values on the same line it looks something like `printf("%d %d\n",x,y);`
- I know the strcmp() ones are a pain, especially the three-way ones. It's usually possible to break things down into multiple steps if that makes things easier



Operator Precedence

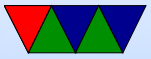


`() [] . ->` `++ -- (postfix)`	left to right	highest
`+ - ! & * ~ ++ -- (prefix) (type) sizeof`	right to left	unary
`* / %`	left to right	arithmetic
`+ -`	left to right	arithmetic
`>> <<`	left to right	shifting
`< <= > >=`	left to right	relational
`== !=`	left to right	equality
`&`	left to right	bitwise AND
`^`	left to right	bitwise XOR
`|`	left to right	bitwise OR
`&&`	left to right	logical AND
`||`	left to right	logical OR
`?:`	right to left	ternary
`= += -= *= /= %=`	right to left	assignment
`,`	left to right	lowest



Operator Precedence – Complicated in C

- There are 15 levels
- Each level has various operations
- Even K&R admitted some of the ordering was a mistake but it was too late to change
- It was too late to change because a few people already depended on the old way, unwittingly inconveniencing millions of people in the future



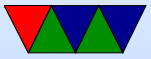
Precedence Gotchas – Bitwise

- The big one is that assignment has higher priority than bitwise operations
- (This was because originally C didn't have the logical operators like `&&`, you used `&` instead, so for `if (x==4 & y==3)` to work you wanted == to have higher priority)
- This means things like

  ```
  if (x == y&3)
  ```

  is evaluated like `if ((x==y)&3)`

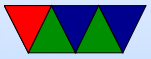
  which is not typically what you'd want



- Also the similar issues `if (y&3 == 2)`
  is evaluated like

  ```
  if (y&(3==2))
  ```
- Modern C compilers will warn you in this case

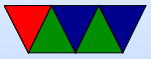
```
warning: suggest parentheses around comparison in operand of ‘&’
```



Precedence Gotchas – Shifting

- Another issue is that even though bitwise shifting acts like multiplication/division, it actually has a lower priority than addition/subtraction
- This means things like `y=x+1<<8`
  is evaluated like `y=(x+1)<<8`
  which might be unexpected
- Modern C compilers will warn you in this case

```
warning: suggest parentheses around ‘+’ inside ‘<<’
```

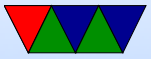


Precedence Gotchas – Inverting

- This may have come up in lab when setting up masks
- `y=~1<<8;`

  when you wanted

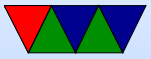
  `y=~(1<<8);`
- The first is actually `y=(~1)<<8;`
- There is *NO WARNING* on this one



Precedence Gotchas – AND vs OR

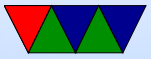
- `if (x || y && z)`
- Is this `(x||y)&&z)` or `x||(y&&z)`
- It's the latter, but you have to know the rules to know that AND has higher precedence than OR
- The compiler will warn here

```
warning: suggest parentheses around ‘&&’ within ‘||’
```



Operator Precedence: Advice

- Useful quote from the internet:
  "If you ever find yourself going to look up the operator precedence, please don't. Use parens. They are free."



Related Aside: Confusing Math

- `if (0<x<5)`? Can't do this, it will do `((0<x)<5)` which is always true

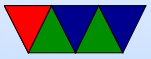
```
warning: comparisons like ‘X<=Y<=Z’ do not have their mathematical m
```

- `if (x==y==z)`? Can't do this, it will do `((x==y)==z)` which compares z against 1 or 0, not x or y

```
warning: suggest parentheses around comparison in operand of ‘==’
```

- `2^X`?

  This is XOR not exponentiation There is a warning for this but not enabled by default?



Examples to Show – These are sadly all valid C

- `1+5/4+2*8`
- `2&7&&4&3`
- `5>4>>2<<4<5`
- `9&12|3^5+1`
- `6-4==2?3:1+1`
- `3+x++---y`
- `x+=y*2^=x`