ECE 177 – Programming I: From C Foundations to Hardware Interaction Lecture 13

Vince Weaver

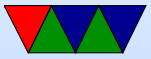
https://web.eece.maine.edu/~vweaver

vincent.weaver@maine.edu

25 February 2026

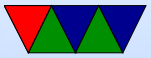
Announcements

- Monday lecture/Lab was cancelled due to snow (again!)
- Monday lab: please pick up parts kit from me (in class or in office hours) and try to get started on it on your own



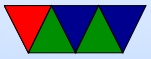
Midterm Preview

- Scheduled March 6th (Friday) in class
- Hopefully it doesn't snow
- Closed book, closed notes, no phone, no laptop, no electronics of any kind, no calculator
- If you have an accommodation let me know in advance



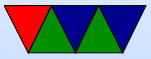
Midterm Setup

- Will be on paper (bring a pen or pencil!)

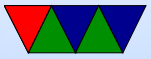


Possible Midterm Topics

- binary / octal / hex numbers
- be able to convert binary to hex and vice versa
- Variable types, char, short, int, long, float, double
- Assignment
- printf()
- functions
- Loops for/while/do-while
- if / else statements
- Arrays

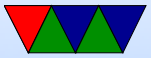


- Strings



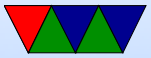
Lab4 – Loops

- Practice writing some C code with loops in it
- Run on the Pi Pico
- Will display on the TFT LCD display
  - TFT = Thin Film Transistors
  - LCD = Liquid Crystal Display



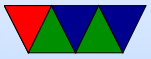
Lab4 – Bare Metal Programming on Pico

- Unlike Windows/Linux/OSX there's no "Operating System" running on the Pico
- An operating system abstracts things away, handles input/output, makes sure things like `printf()` and `scanf()` interact with user
- For the Pico we have none of that so if we want to interface with I/O (like keypad or display) we have to provide code to do that



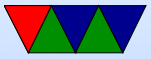
Lab4 – Provided Code

- `Adafruit_ST7735.c` – driver for LCD screen
- `glcdfont.c` – 5x7 ASCII font (array!)
- `tft_stdout.c` – provides "stdout" interface so `printf()` will work
- `bootrom_api.c` – hack that will switch to BOOTSEL mode if you press "*" and "D" on the keypad
- `graphics.c` – graphics library from adafruit, lines, circles, rectangles, etc
- `syscalls.c` – stub syscalls so code compiles



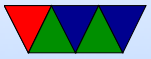
Lab4 – Info on Display

- 320x200 grid of pixels, color
- SPI display, program by sending some serial signals down wires to the display
- For now just treat it as magic, if you ever take ECE471 you'll learn more about it. Can read data sheet if curious
- For this lab just use `printf()` and the characters will get mapped by the font and drawn to the screen
- **It is really slow and flickery** (it redraws whole screen after each character)

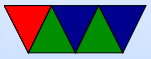


Lab4 – Simple Loops

- Write numbers (e.g. 1 ... 10 one per line)
- Write numbers (e.g. -10 ... 10 one per line)
- Write even numbers (many ways to do this)
- Be careful, easy to be off by one in C
- Be careful cutting and pasting loops too, easy to forget to update index value
- Aside, if you use say i as loop index, can you re-use it later without re-declaring it? (yes)



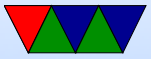
Lab4 – Aside on incrementing



Incrementing in C

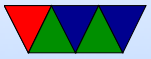
- Add one to number, assign back to oneself

  `i=i+1;`
- C has the shortcut `i+=1;`
- C in addition has a special operator, `++`
- So to increment i, you can do `i++` which is the same as `i=i+1;` The assignment is assumed



Aside on incrementing

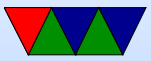
- You can do -- as well to decrement
- To extra confuse things you can also do `++i`
- What is the difference? Pre or post increment



C post increment

- `i++`
- If you do `x=i++;` X gets the value of i, *then* i is incremented

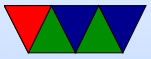
```
x=1;
printf("\tx = %d\n",x);          // 1
printf("\tx++ = %d\n",x++);      // 1
printf("\tx after = %d\n",x);    // 2
```



C pre increment

- `++i`
- If you do `x=i++;` i is incremented first, then x is assigned

```
x=1;
printf("\tx = %d\n",x);          // 1
printf("\t++x = %d\n",++x);      // 2
printf("\tx after = %d\n",x);    // 2
```

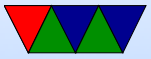


Why to avoid sometimes

- Why didn't I do an example like

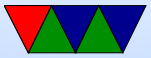
```
printf("%d %d %d\n",x,x++,x);
```

- It turns out there's no guarantee in C what order the comma separate arguments to a function are evaluated, so if they have "side effects" (update a value with an assignment or increment) it can do unexpected things



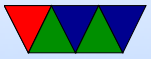
Lab4 – Clearing the screen

- Just print 20 newlines (screen is 20x20 chars big)
- There is probably a more efficient way of doing this



Lab4 – Pausing

- How can you wait for a certain amount of time?
- Just have a busy loop?
- `for(i=0;i<1000000;i++)`?
- Problem is compiler might optimize that away
- Better way is to use "timers" on chip. Complex, you will see those in ECE271
- Provided `sleep_ms();`



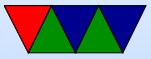
Lab4 – Nested Loops and conditionals

- Making ASCII art patterns
- Let's say you want to print a line of 4 dollar signs using a loop

```
int i;
for(i=0;i<4;i++) {
    printf("$");
}
printf("\n");
```

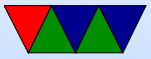
- Let's say you want 4 lines of this

```
int i,j;
for(j=0;j<4;j++) {
    for(i=0;i<4;i++) {
```



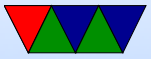
```
            printf("$");
        }
        printf("\n");
    }
```

- How would you move this to the bottom of the screen?



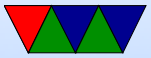
Lab4 – Aside on Sierpinski

- Just use bitwise and to get triangle pattern
- Why does it happen? Just one of the fun things in our Universe



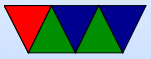
Lab4 – Comment your Code!

- Provided code has comments, but add them to your own code
- Trying to explain what your code is doing can help you think through it



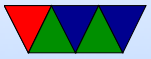
Lab4 – Compiling/Running Your Code

- Use VS code to edit "loops.c"
- I think we have you use BOOTSEL mode where you hold the white button while plugging in via USB
- Drag the loops.uf2 file created when building to the pi drive that appears
- It should run through your code
- At end you can press * and D at same time to re-enter boot mode so you don't have to reconnect with button down



Lab4 – Possible without VS Code?

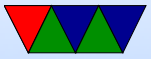
- I've been running w/o VS Code on Linux but it's a huge pain
- Mostly CMake which I hate
- If I had more time I'd get some better instructions for this



HW3 – Print Multiple Values in One line

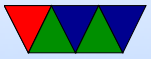
- Many people having trouble on the one saying print an int and a float
- It says to use a single statement, so only use one printf
- Remember you can print more than one value

```
printf("Print many things %d %d %d\n",a,b,c);
```



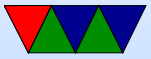
HW3 – Print a Particular String

- Almost a trick question
- Asks you to print people's names like "Smith, Bob"
- If all it wants is a string printed (no substitutions) you don't need any % chars. Just print the string



HW3 – Total Value

- Wants total value, maybe a sum of squares
- A loop adding them all up
- It says total is provided, but unless it says total is initialized for you, you'll need to init it yourself. To what value? 0.
- This is a bit confusing because some of the previous ones pre-init it for you.



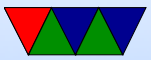
HW3 – if/else statements

- Simple case, comparison, and otherwise

```
if (snow<4) printf("too little\n");
else printf("no school!\n");
```

- If you nest them, need to start a new if after the else

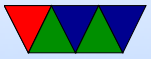
```
if (snow<4) printf("too little\n");
else if (snow<8) printf("getting there\n");
else printf("no school!\n");
```



HW3 – printing a single character

- printf expects a string (a value in double quotes)
- You can have single character strings "*"

- You can `printf("*")`
- This is not the same as a one-byte character (single quotes)
- You cannot `printf('*')`
- putchar(); – maybe C even does this



HW3 – splitting printfs

- if you don't include a newline C will run everything together
- So if you need 97 *s you can do it with a loop and just no newline