ECE 177 – Programming I: From C Foundations to Hardware Interaction Lecture 9

Vince Weaver

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

vincent.weaver@maine.edu

11 February 2026

Announcements

- CodeLab: if you run into trouble, feel free to e-mail, stop by office hours, or ask after class. Do be aware I might not be checking my e-mail as often after hours
- Question that came up with literal values and L vs l. The question bank we are using is from a textbook and it expects L over l even though both allowed (L is usually preferred as it avoids confusion betweek l and 1). I might not be able to change the answer so might not get fixed until next year.

Lab Reminder

- Makeup labs this week.
  We possibly have solutions to most of the VS Code issues now
- Lab#3 will start up Tuesday
  I got the new displays working
- Reminder no class on Monday (President's Day)

What to do with Complex if trees?

```
if (x==0) printf("Zero\n");
else if (x==1) printf("One\n");
else if (x==2) printf("Two\n");
else if (x==3) printf("Three\n");
else if (x==4) printf("Four\n");
else if (x==5) printf("Five\n");
else printf("Too many!\n");
```

Can replace with switch/case

```
switch(x) {
   case 0: printf("Zero\n");
           break;
   case 1: printf("One\n");
           break;
   case 2: printf("Two\n");
           break;
   case 3: printf("Three\n");
           break;
   case 4: printf("Four\n");
           break;
   case 5: printf("Five\n");
           break;
   default: printf("Too many!\n");
           break;
```

Things to note

- The value passed to `switch()` must be char/short/int/long
- Floating point and strings not allowed
- It must be a literal (constant) expression for each case. Can't be a variable
- Can't have a range of values like `case 0..2` which some languages allow.
- Each case must be unique. Don't have to be in order.
- `default` case will run if none of the other cases triggered

Fallthrough

- If you leave `break` statements off, it will just fall through to the next case
- People often do this to be clever

Fallthrough Example

```
switch(x) {
    case 0: case 2: case 4: case 6: case 8: case 10:
        printf("Even!\n");
        break;
    case 1: case 3: case 5: case 7: case 9:
        printf("Odd!\n");
        break;
    default:
        printf("I don't know!\n");
        break;
}
```

Switch Case Accidental Fallthrough

- It is easy to do this accidentally and get unintentional behavior

Can compiler warn on Accidental Fallthrough?

- gcc has `-Wimplicit-fallthrough` enabled by -Wextra
- Some compilers will notice if you have a /* FALLTHROUGH */ comment and not warn
- old gcc/clang you could put `__attribute__((fallthrough))`
- C17/C23 you should use `[[fallthrough]];`

Aside on variable names with leading underscores

- Rules on C variable/function names
  - can contain letters, digits, and underscores
  - cannot begin with number
  - names are case sensitive
  - name cannot have whitespace
  - cannot be reserved word (like int)
  - variables with leading underscores are in theory reserved for the compiler/implementation

Duff's Device (Famous abuse of switch() in C)

```
int send(int *to, int *from, int count) {
    int n = (count + 7) / 8;
    switch (count % 8) {
    case 0: do { *to = *from++;
    case 7:      *to = *from++;
    case 6:      *to = *from++;
    case 5:      *to = *from++;
    case 4:      *to = *from++;
    case 3:      *to = *from++;
    case 2:      *to = *from++;
    case 1:      *to = *from++;
            } while (n-- > 0);
    } }
```

simple input with getchar

- `getchar()` defined in stdio.h
- Returns an integer. Why? Shouldn't it return a char? (it lets you return -1 to indicate an error

Show off “guessing game” example

```c
/* Cool game for ECE177 */

#include <stdio.h>

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

  int c;          // input character
  int number=3;   // number trying to guess
  int guesses=0;  // guesses so far

  while(1) {      // loop forever

    printf("Please guess a number from 0 to 9\n");

    /* getchar also returns the carriage return when */
    /* we press enter so skip reporting that as a guess */
    while(1) {
      c=getchar();
      if ((c=='\n') || (c=='\r')) continue;
      break;
    }

    guesses=guesses+1;  // increment times guessed

    if (c=='q') break;  // if press q then exit

    /* handle if proper number */
    if ((c>='0') && (c<='9')) {

       c=c-'0';  // convert from ASCII to number 0..9

       if (c==number) {  // check if guessed right
```

```
            printf("You guessed right!\n");
            printf("It took you %d guesses\n",guesses);
            break;           // exit out of infinite loop
         }
         else if (c>number) {
            printf("You guessed too high!\n");
         }
         else {
            printf("You guessed too low!\n");
         }
      }
      else {
        printf("Invalid keypress %c, try again!\n\n",c);
      }
    }

    /* we've exited from infinite game loop */
```

```c
    printf("GAME OVER!\n");

    return 0;
}
```

Aside: why no GUI?

- So far we've been running programs at the text console (or teminal window)
- Why not write fancy GUI based programs? With graphics and mouse input?
- You can do that in C, it's just a lot more complicated
- This simple console base stuff makes it a bit more clear what's going on

getchar() notes

- Note by default Linux/UNIX reads a line at a time
- `getchar()` doesn't return until after ENTER pressed
- If you typed multiple things, they buffer up and you get all of them one by one, including the linefeed itself
- There are ways to make it non-blocking but too complex for now
- There are better ways to read values like this but they require knowledge of strings, arrays, and pointers and we aren't quite there yet

- Note: getchar returns ASCII data so to convert to a raw integer need to subtract off 48 ('0') which is ASCII for zero

Making our game better

- Fancy ASCII art title screen?
- Press 'h' for help?
- Pick a random number to guess?
  Use `rand()` or `random()` but you might want to seed it
- Non-blocking I/O
- ANSI escape chars: Colors, clear screen
- Sound effects? (difficult, but can in theory make it beep with `\a`

bitwise logic

- We started bitwise logic, see the next lecture's notes for that