CS 240: Programming in C

Lecture 9: Bit Masking, Bit Flags

PURDUE UNIVERSITY

Announcements

- Homework 3 due tonight!
- Code style grade for homework 2 coming soon

PURDUE UNIVERSITY®

Announcements

- Midterm 1 seating charts will be posted
    - Look for an announcement on Ed
- DRC accommodated exams will overlap the exam time
    - Held in MATH 215, starting at 6:00 pm
    - I’ll send an email with more details

Bit masking

- Represent a color with an 8-bit integer

```
char color = 44;  /* binary 00101100 */
                          /* RRRGGBBB */
```

- How would we get the Blue component?

Bit masking

- Represent a color with an 8-bit integer

```
char color = 44;  /* binary 00101100 */
                         /* RRRGGBBB */
```

- How would we get the Blue component?
- We want to isolate the first (right-most) 3 bits

```
char color = 44;               /*   00101100 */
char mask = 7;                 /* & 00000111 */
char blue = color & mask;      /* = 00000100 */
```

Bit masking

- What if we want to change the number of bits?

```
char color = 44;  /* binary 00101100 */
                          /* RRGGBBBB */
char bits = 4;
```

Bit masking

- What if we want to change the number of bits?

```
char color = 44;  /* binary 00101100 */
                         /* RRGGBBBB */
char bits = 4;
```

- We can construct a mask

```
char mask = (1 << bits) - 1;    /* 00001111  */
                                /* = 15      */
                                /* = 2^4 - 1 */
char blue = color & mask;
       /* = color & ((1 << bits) - 1);
```

Bit masking

```
char color = 44;  /* binary 00101100 */
                          /* RRRGGBBB */
```

- What if we want the red channel?

```
char mask = 224;          /* 11100000 */
```

Bit masking

```
char color = 44;  /* binary 00101100 */
                               /* RRRGGBBB */
```

- What if we want the red channel?

```
char bits = 3;
char offset = 5;
char mask = ((1 << bits) - 1) << offset;
       /* = (2^3 - 1) * 2^5 */
char red = color & mask;
      /* = color & (((1 << bits) - 1) << offset); */
      /* 00100000 */
```

Bit masking

```
char bits = 3;
char offset = 5;
char mask = ((1 << bits) - 1) << offset;
char red = color & mask;    /* 00100000 = 32 */
```

- This value is still shifted -- if we want Red itself, we need to shift it to the right
  - Might not always be what you want!

```
char red = (color & mask) >> offset;
```

PURDUE UNIVERSITY

Bit flags

- Sometimes you want to store a bunch of yes/no values inside a single number

```
enum burger_topping {
  NONE    =   0,    /* 00000000 */
  CHEESE  =   1,    /* 00000001 */
  LETTUCE =   2,    /* 00000010 */
  TOMATO  =   4,    /* 00000100 */
  PICKLES =   8,    /* 00001000 */
  BACON   =  16,    /* 00010000 */
  KETCHUP =  32,    /* 00100000 */
  MUSTARD =  64,    /* 01000000 */
  MAYO    = 128,    /* 10000000 */
};
```

Bit flags

```
char burger = CHEESE | LETTUCE | TOMATO | KETCHUP;
              /* 00100111 */
```

- Let’s add bacon to it

```
burger |= BACON;    /* burger = burger | BACON; */
```

- Were there pickles on this burger?

```
if (burger & PICKLES) { fprintf(stderr, “yuck!”); }
```

- I don’t want tomatoes on it anymore

```
burger = burger & ~TOMATO;
```

A note on enums

- sizeof(enum) is implementation-defined
    - Usually int (i.e., 4), but can be smaller or larger
    - Never float
- Why use enums over #define?
    - Better compiler errors / warnings
    - You don’t need to give each one a value
    - Switch statements can warn you if you don’t use all values

A note on enums

```
void print_topping(enum burger_topping t) {
  switch (t) {
    case NONE:    printf(“None\n”);    break;
    case CHEESE:  printf(“Cheese\n”);  break;
    case LETTUCE: printf(“Lettuce\n”); break;
    case TOMATO:  printf(“Tomato\n”);  break;
    case BACON:   printf(“Bacon\n”);   break;
    case KETCHUP: printf(“Ketchup\n”); break;
    case MUSTARD: printf(“Mustard\n”); break;
    case MAYO:    printf(“Mayo\n”);    break;
  }
}
```

```
$ gcc -Wall -Werror -o burger burger.c
burger.c: In function ‘print_topping’:
burger.c:16:3: error: enumeration value ‘PICKLES’ not handled in switch [-Werror=switch]
   16 |   switch (t) {
      |   ^~~~~~
cc1: all warnings being treated as errors
```

For next lecture

- We will have a midterm review
- Think of things you want to go over
- Prepare questions

Slides

- Slides are heavily based on Prof. Turkstra's material from previous semesters.