601.220 Intermediate Programming

Summer 2023, Meeting 8 (June 23th)

Today's agenda

- Exercises 13 and 14 review
- Midterm project overview
- "Day 15" material
  - Number representation, type conversion/casting
  - Exercise 15

Reminders/Announcements

- Midterm project: due Friday, June 30
  - Team repositories for everybody were created last night. Please, let us know if you don't see your repo.
- Midterm exam: in class on Wednesday, July 5th

Exercise 13 review

In a `struct` data type, "`struct`" is part of the name of the data type.

E.g.:

```
struct Stat {
  int num_of_goals;
  int num_of_assists;
  float pass_accuracy;
  int min_played;
  int num_of_shots;
  float shot_accuracy;
};
```

The data type is "`struct Stat`".

Exercise 13 review

Common shortcut: use `typedef` to avoid the need to use "`struct`" when referring to the data type. E.g.:

```
typedef struct {
  int num_of_goals;
  int num_of_assists;
  float pass_accuracy;
  int min_played;
  int num_of_shots;
  float shot_accuracy;
} Stat;
```

Now "`Stat`" can be used as the name of the data type. The `main` function assumes you have done this for each `struct` type.

Exercise 13 review

Find the player with the latest signing date:

```
index = 0;
for (int i = 1; i < TEAMSIZE; i++) {
  Player *last = &team[index];
  Player *p = &team[i];
  if ( /* p's date is later than last's date */ ) {
    index = i;
  }
}
```

Exercise 13 review

Use valgrind to analyze memory use:

```
==2049== 24 bytes in 1 blocks are definitely lost in loss record 1 of 3
==2049==    at 0x483B7F3: malloc (in /usr/lib/x86_64-linux-gnu/valgrind/vgprelo
==2049==    by 0x109314: main (main.c:10)
==2049==
==2049== 132 bytes in 11 blocks are definitely lost in loss record 2 of 3
==2049==    at 0x483B7F3: malloc (in /usr/lib/x86_64-linux-gnu/valgrind/vgprelo
==2049==    by 0x1095A0: create_player (soccer.c:10)
==2049==    by 0x10981B: create_team (soccer.c:41)
==2049==    by 0x10932B: main (main.c:11)
==2049==
==2049== 240 bytes in 10 blocks are definitely lost in loss record 3 of 3
==2049==    at 0x483B7F3: malloc (in /usr/lib/x86_64-linux-gnu/valgrind/vgprelo
==2049==    by 0x10965A: create_player (soccer.c:22)
==2049==    by 0x10981B: create_team (soccer.c:41)
==2049==    by 0x10932B: main (main.c:11)
```

Exercise 13 review

Freeing memory:

```
for (int i = 0; i < TEAMSIZE; i++) {
  free(team[i].date);
  free(team[i].stat);
}
```

Observation: for each `Player` instance in the `team` array, the `date` and `stat` fields are pointers to dynamically allocated instances of `Date` and `Stat`.

Exercise 14 review

Converting from string of 0 and 1 digits to a binary integer:

```
int str_to_int(char msg[], int len) {
  int result = 0;

  for (int i = 0; i < len; i++) {
    int index = len - i - 1;
    char c = msg[index];
    if (c == '1') {
      result |= (1 << i);
    }
  }

  return result;
}
```

Exercise 14 review

Converting from binary integer to string of 0 and 1 digits:

```
void int_to_str(int num_encrypted, char msg_encrypted[],
                int len) {
  for (int i = 0; i < len; i++) {
    int bit_pos = (len - i - 1);
    char bit =
      (num_encrypted & (1 << bit_pos)) == 0 ? '0' : '1';
    msg_encrypted[i] = bit;
  }
  msg_encrypted[len] = 0; // NUL terminator
}
```

Exercise 14 review

Performing the encryption:

```
for (int i = 1; i < n; i++) {
  num_encrypted ^= (num_encrypted << 1);
}
```

Day 15 recap questions

1. What is two's complement representation?
2. How does representation of integers and floating-point values differ in C?
3. What is type narrowing?
4. What is type promotion?
5. What is type casting?
6. What is the output of the code segment below?

1. What is two's complement representation?

Two's complement is used as the representation of signed integers on all modern computer architectures.

Idea: most significant bit makes a *negative* contribution to the value of the integer.

Consider the bit string 10000101:

- As an 8 bit unsigned value: $128 + 4 + 1 = 133$
- As an 8 bit signed two's complement value: $-128 + 4 + 1 = -123$

Big advantage of two's complement representation: addition and subtraction work the same way for both unsigned and signed values.

Negating a two's complement value

To invert a two's complement value, invert all of the bits and add 1.

Why?

A bit string where every bit is 1 has the value -1.

a is an integer, ~a is the "complement" of a (all bits inverted).

For any a, a + ~a = -1 (e.g., 10010110 + 01101001 = 11111111)

Rearranging: -a = ~a + 1

2. How does representation of integers and floating-point values differ in C?

Integer representation: either unsigned or signed two's complement.

Floating point representation: IEEE 754.

IEEE 754 is essentially base-2 scientific notation. "Normalized" floating point values have the form $\pm 1.x \times 2^y$

x is the fraction (represented in base 2)

y is the exponent (representeed in base 2, can be positive or negative)

Limitations of floating point

Arithmetic on floating point values may involve rounding. Results should generally be considered to be approximate.

Also: some numbers can't be represented exactly. For example, 0.1 has no exact representation (becomes a "repeating decimal" in the fraction.)

3. What is type narrowing?

Type narrowing is converting a value belonging to a "larger" numeric type to a "smaller" numeric type. E.g., converting a `double` value to an `int`.

Narrowing conversions may lose information.

For example:

```
float f_val = 3.5;
int i_val = f_val;  // narrowing conversion, i_val=3
```

4. What is type promotion?

A type promotion is converting a value belonging to a "smaller" numeric type to a "larger" numeric type. E.g., converting an `int` value to `double`.

Will *generally* not lose information, although some promotions (e.g., `int` to `float`) may lose information in some cases.

For example:

```
int i_val = 3;
double d_val = i_val; // promotion, d_val=3.0
```

5. What is type casting?

Type casting is an *explicit* conversion from one type to another.

Can be used to eliminate warnings in some cases:

```
// Without the cast, there is a warning
// (comparison of signed and unsigned values)
// in the loop condition
size_t len = strlen(str);
for (int i = 0; i < (int) len; i++) {
  char c = str[i];
  // ...
}
```

Other motivations for casts

In addition to avoiding compiler warnings, casts can also be useful to explicitly indicate where narrowing conversions are happening in the program.

6. What is the output of the code segment below?

```
int n = 32065; // in binary: 111110101000001
float x = 24.79;


printf("int n = %d but (char) n = %c\n", n, (char) n);


printf("float x = %f but (long) x = %ld\n", x, (long) x);
```

Midterm project

Image processing: like Photoshop(tm), but a command line program which

- reads an input image
- applies a transformation
- writes an output image

PPM files

All widely-used image formats (PNG, JPEG, etc.) are compressed, so the data representation is very complicated.

We will use the PPM format, which is uncompressed, and relatively easy to read and write.

The starter code provides `ppm_io.h`/`ppm_io.c` for reading and writing PPM files.

Images and pixels

An image is a rectangular grid of *pixels*. Each pixel is a colored dot. Pixel color is represented as *red*, *green*, and *blue* intensity values in the range 0–255. An RGB triple can represent (essentially) any color that can be perceived by the human eye.

Example colors:

- $(0, 0, 0)$: is black
- $(255, 255, 255)$: is white
- $(17, 59, 94)$: the darker shade of blue on this slide
- $(26, 89, 142)$: the lighter shade of blue on this slide

Pixel data type

```
typedef struct _pixel {
  unsigned char r;
  unsigned char g;
  unsigned char b;
} Pixel;
```

Image data type

```
typedef struct _image {
  Pixel *data;  // pointer to array of Pixels
  int rows;     // number of rows of Pixels
  int cols;     // number of columns of Pixels
} Image;
```

Note that:

- The `data` field points to a dynamically-allocated array of `Pixel` elements
- The pixels are stored in *row major* order: the top row of pixels is first, then the second row of pixels, etc.

Pixel coordinates

X coordinates: 0 is the left most column.

Y coordinates: 0 is the *top* row. (Not the bottom row!)

Original image

Grayscale

Convert each pixel in the input image so that

1. The "intensity" of the pixel is preserved, but
2. The RGB component values are equal to each other

Project description has a formula for doing the converstion.

Grayscale example output

Binarize

Convert each pixel to either black or white based on comparing its computed grayscale value to a threshold value.

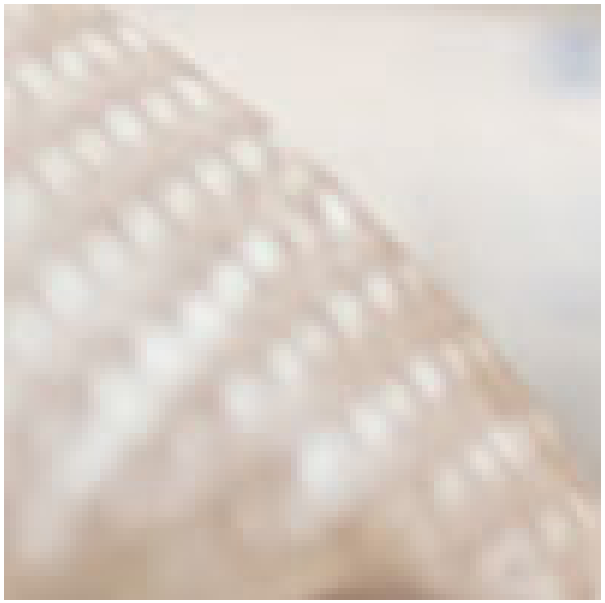
Binarize example output

(With threshold value set to 127.)

Crop

Select a rectangular region of the input image. The region is specified as coordinates of top left and bottom right corners of region.

Crop example output



(With region specified as $(200, 200)$ and $(300, 300)$.)

Transpose

Output image has the same pixels as input image, but the x/y coordinates are swapped.

Transpose example output

Gradient

Convert image to grayscale.

Compute "gradient" grayscale intensity for each pixel in both horizontal and vertical directions.

Result image grayscale intensity for each pixel is the sum of the absolute values of the horizontal and vertical gradient values.

Gradient example output

Seam carving

Rescale an image while avoiding changing the aspect ratio of the "interesting" parts of the image.

Uses gradiant image to determine where "seams" should be identified (to guide which parts of the original image should be removed.)

Details in project description. This will be fairly tricky to implement!

Seam carving example output

Viewing images

The image files, particularly the *result* image files, will be in your ugrad account. How to view them on your local computer?

Option 1: Preview them in VS Code. If you have VS Code set up to access your ugrad account, this should work transparently.

Option 2: Use `scp` to copy them to your local machine.

Option 3: Use X forwarding (`ssh -X`) and the `feh` image viewer program.

Team dynamics

This is a team project.

Team projects work best when team members work together synchronously. This is known as *pair programming*.

Recommendation: avoid having overly-specific division of responsibilities between team members. The best scenario is when

1. All team members know what everyone is working on, and
2. Any team member can contribute to *any* part of the project

Avoid "critical path" dependencies. I.e., team member 1 can't make progress because they are waiting for team member 2 to complete something.

Using Git for a team project

Everyone on the team should clone the team's midterm project repository.

Each team member will use `git push` and `git pull` to synchronize their work with the project repository.

Merges and merge conflicts

Sometimes, before you `git push` your own work, you will need to `git pull` to bring your local repo up to date with your teammates' work.

Most of the time, `git` automatically creates a *merge commit* to reconcile your work with your teammates' work. This is possible when your changes affect different parts of the code than the code your teammates modified.

However, this could also result in a *merge conflict* if your changes conflict with your teammates' changes.

Resolving a merge conflict

If `git` tells you a merge conflict occurs, it will add *conflict markers* to the affected code to indicate where the conflicts are. (They are very obvious: look for places where there are <<<< and >>>>.)

You will need to edit the affected files to

1. Remove the conflict markers, and
2. Manually reconcile your changes with your teammates' changes

When you're done, and you've tested the code, just do `git add` on the files were conflicts where found, then `git commit`.

Planning your time

As always:

1. Start early (i.e., *now*)
2. Make steady, incremental progress
3. Work on the simpler image transformations first (the order in the project description is the recommended order)

Let us know if you have questions!

Exercise 15

- Integer representation, random number generation
- Note that Part 3 is optional!
- Breakout rooms 1–10 are “social”
- Use Slack to let us know if you have a question!