EECS10 Discussion Week10

TA: Emad Arasteh
emalekza@uci.edu
eecs10@eecs.uci.edu

Office Hours: Fri, 8:00-9:00am EH 3404

University of California, Irvine

UCIrvine
University of California, Irvine

Assignment 7

- A manual driven digital image processing program

- Using function calls for image file handling, image processing, and testing.
  - Function declaration, function definition, function call
  - Function parameters
  - Scope of the variables

- Two-week assignment. Plan and start early!
  - Week1: Setup the working environment. Design the user menu. Try 1~2 operations on the image.
  - Week2: Complete the operations. Test your program.

- Use the web browser to view your image.

Image processing functions

- Black and white
  - For each pixel at coordinate (x,y), compute the average of three color channels
  - Set the new value for all three color channels equal to the average
- Negative
  - Subtract R[x][y], G[x][y] and B[x][y] from the max intensity value (255) and update the pixel value
- Flip horizontally
  - Hint: scan only half of the image
- Mirror horizontally
  - Hint: scan only half of the image

Image processing functions

- Zoom-in
  - Typo: the correct coordinates of the zoom area are: (250, 100), (**569**, 100), (250, 339) and (**569**, 339)
  - Hint: arrows are pointing to coordinates in the new image

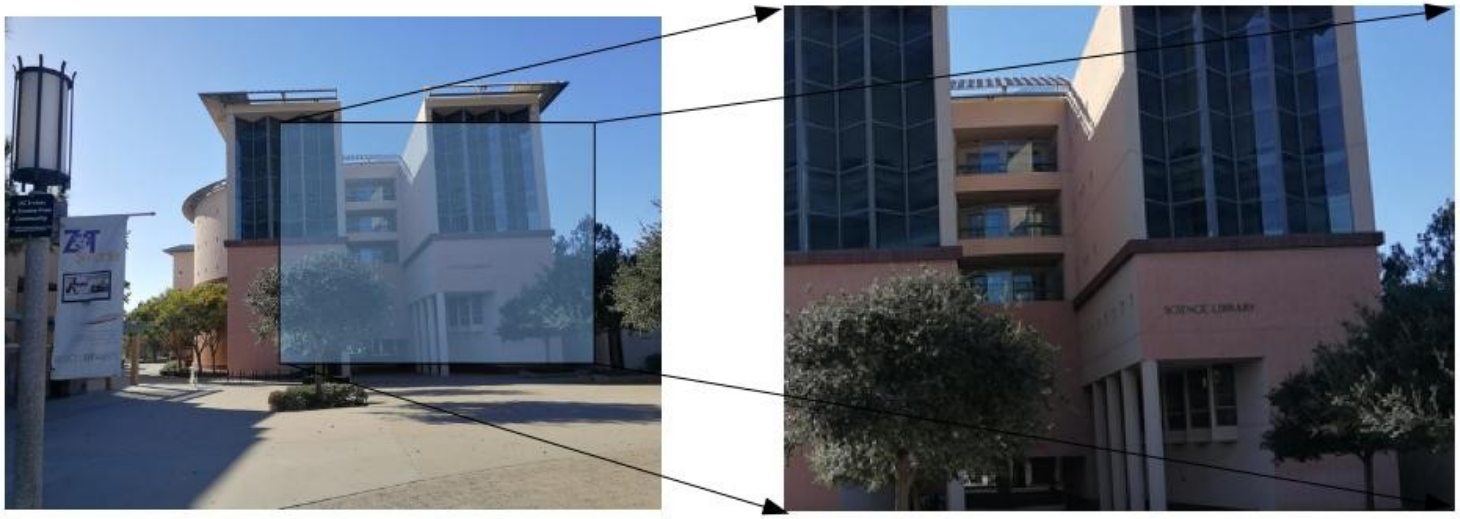


Image processing functions

- Add noise
  - – Randomly generate coordinates (2 random number for x and y)
  - – Set the intensity values to maximum (255, 255, 255) or minimum (0, 0, 0) alternatively for those noisy pixels

- Sharpen
  - – Slide the filter on the image and compute the weighted sum for each pixel
  - – Watch out for pixel values greater than max intensity (255) or less than min intensity (0)
  - – Watch out for pixel coordinates at the border of the image

Image processing functions (bonus)

- Overlay
  - – Pick either a pixel from the original image or a pixel from the overlay image depending on the background pixel intensity

- Add borders
  - – Turn the pixels on the border into a specific color (defined by the user)

Data structures

- Arrays

- Structures: **struct**
  - user-defined, composite data type

- Unions: **union**
  - user-defined, composite data type but only one member may be used at a time!

- Enumerators: **enum**
  - user-defined data type that members are an enumeration of integral constants

- Understand meaning of *declaration*, *definition*, *instantiation* and *initialization* and use these in the context of programming properly

Data structures

• By using **typedef**, you can write cleaner code and save extra keystrokes typing **struct** all over the code :

```
struct point_t {
    int x;
    int y;
}

struct point_t p1;
```

```
typedef struct {
    int x;
    int y;
} point_t

point_t p1;
```

• Let’s do some coding by defining a struct to describe a point in the plane, modify the coordinate and find perimeter of triangle

Pointers

- Pointers are variables whose values are addresses
  - The “address-of”operator (**&**) returns a pointer.

- Pointer Definition
  - The unary ***** operator indicates a pointer type in a definition

- Pointer Definition
  - A pointer may be set to the “address-of” another variable
  - A pointer may be set to 0 (points to no object)
  - A pointer may be set to NULL (points to “NULL” object)

- Pointer Dereferencing
  - The unary * operator dereferences a pointer to the value it points to (“content-of” operator)
  - The -> operator dereferences a pointer to a structure to the content of a structure member

Files

- Up to now, all data processed is available only during program run time

- Persistent data is stored even after a program exits

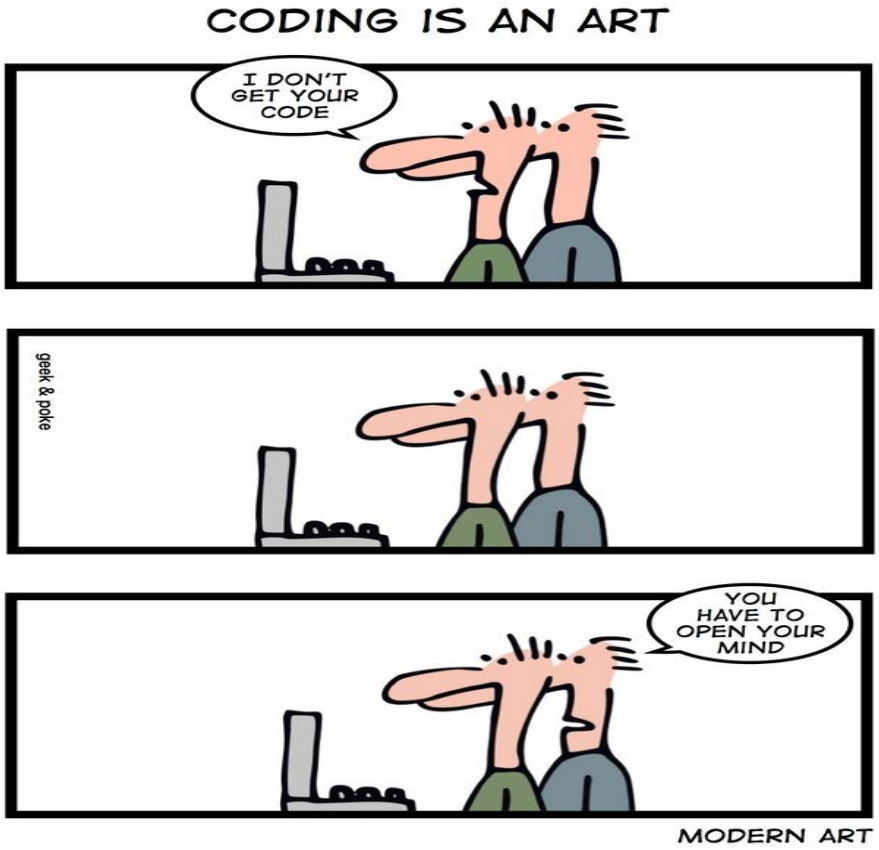
- Persistent data is stored in files...
  - ... on the harddisk
  - ... on a removable disk (CD, memory stick, …)
  - … on network drive
  - ...on a tape,..

- Input and output from/to files is organized as I/O streams (a *stream* is a source or sink of data usually individual bytes or characters)

Standard I/O Functions

- I/O streams:
  - Standard I/O streams (opened by the system)
    - `stdin  i.e. scanf()`
    - `stdout i.e. printf()`
    - `stderr i.e. perror()`
  - File I/O streams (explicitly opened by a program)
    - Open a file `fopen()`
    - Write data to a file `fprintf()`
    - Read data from a file `fscanf()`
    - Close a file `fclose()`
- In C, all I/O functions are
  - ... declared in header file stdio.h
  - ... implemented in the C standard library

Thank you!



Geek & poke, used under CC-BY-3.0