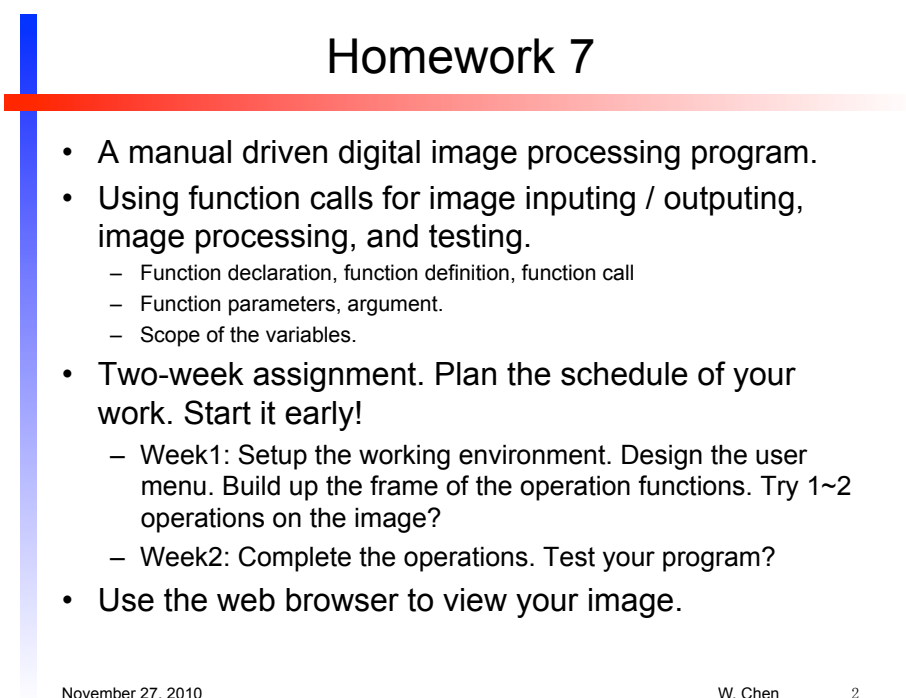


EECS10 Discussion Week10

TA: Weiwei CHEN
weiwei.chen@uci.edu
eecs10@eecs.uci.edu



Homework 7

- A manual driven digital image processing program.
- Using function calls for image inputing / outputing, image processing, and testing.
 - Function declaration, function definition, function call
 - Function parameters, argument.
 - Scope of the variables.
- Two-week assignment. Plan the schedule of your work. Start it early!
 - Week1: Setup the working environment. Design the user menu. Build up the frame of the operation functions. Try 1~2 operations on the image?
 - Week2: Complete the operations. Test your program?
- Use the web browser to view your image.

November 27, 2010 W. Chen 2

Homework 7

- **Black & White**
 - Get the average value of the three color channels for each pixel (x,y).
 - Set $R[x][y]$, $B[x][y]$ and $G[x][y]$ to be the average value.
- **Negative**
 - Subtract $R[x][y]$, $B[x][y]$ and $G[x][y]$ from 255 and set the new value back.
- **Flip horizontally**
 - Swap pixel (x,y) and pixel (width-1-x, y)
 - Scan half of the picture

November 27, 2010 W. Chen 3

Homework 7

- **Mirror horizontally**
 - Copy pixel (width-1-x, y) to pixel (x,y)
 - Scan half of the picture.
- **Add Noise**
 - Randomly generate size/100 pixels. 2 random number for x, y coordinate respectively.
 - Set the pixel into white (255, 255, 255) and black (0,0,0) alternatively

November 27, 2010 W. Chen 4

Homework 7

- Color correction
 - Pick up the “green” pixels (if-statement)
 - Reset the color-tuple
- Image overlay
 - Load the second picture (anteater.ppm)
 - Scan the small picture, if the pixel is not white or the blue water (check the color value in the assignment), copy the pixel to the correspondent position in the original picture (swan.ppm)
 - Pixel (x', y') in anteater to (x'+offsetx, y'+offsety)

November 27, 2010 W. Chen 5

Homework 7

- Blur
 - Get the average values of the three channels of the current pixel and the 24 neighbors.
 - Set the pixel's color components to the average values respectively.
 - In order not to contaminate the original value of the picture, use temporary arrays for computation and copy the result back to the original arrays. ()
- Add border
 - Turn the pixels on the border into a specific color (defined by the user)
 - Scan the picture and test the coordinate of the pixels

November 27, 2010 W. Chen 6

Homework 7

- Test your program
 - AutoTest() function
 - Call all the other operation functions together in the program.
 - Be careful with the arguments for each functions.
 - Sample function calls are listed in the assignment.
- Global constants
- Scope of the variables
- Pass by reference when using array parameters.
- Function prototypes mentioned in the assignment are very helpful hints.

November 27, 2010

W. Chen

7

Data Structures

- Structures: **struct**
- Unions: **union**
- Enumerators: **enum**
- Type definitions: **typedef**

11/27/10

W. Chen

8

Data Structures

- Unions: union
 - User-defined, composite data type
 - Type is a composition of (different) sub-types
 - Fixed set of *mutually exclusive members*
 - *Names and types of members are fixed at union definition*
 - *Member access by name*
 - *Member access operator: union_name.member_name*
 - *Only one member may be used at a time*
 - *All members share the same location in memory!*

11/27/10 W. Chen 9

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 initialization or assignment
 - 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)

11/27/10 W. Chen 10

Pointers

- 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
- String Manipulation: string.c

11/27/10 W. Chen 11

FILE I/O streams

- Up to now, all data processed is available only during program run time
- At program completion, all data is lost
- *Persistent data is stored even after a program exits*
- *Persistent data is stored in files...*
 - ... on the haddisk
 - ... on a removable disk (CD, memory stick, ...)
 - ...on a tape,...
- *Input and output from/to files is organized as I/O streams*

11/27/10 W. Chen 12

Standard I/O Functions (part)

- Functions declared in `stdio.h`
 - `typedef __FILE FILE;`
 - opaque type for a file handle
 - `FILE *fopen(const char *n, const char *m);`
 - open file named `n` for input ("`r`"), output ("`w`"), or append ("`a`")
 - returns a file handle, or `NULL` in case of an error
 - `int fclose(FILE *f);`
 - closes an open file handle
 - `int fprintf(FILE *f, const char *fmt, ...);`
 - `int fscanf(FILE *f, const char *fmt, ...);`
 - `int fgetc(FILE *f);`
 - `char *fgets(char *s, int n, FILE *f);`
 - `int fputc(int c, FILE *f);`
 - `int fputs(const char *s, FILE *f);`
 - input/output functions from/to stream `f`
 - `int fflush(FILE *f);`
 - flushes any unwritten data from a buffer into the file

11/27/10 W. Chen 13

- **Thank you very much for this quarter!**
- **Good luck for the finals!**

11/27/10 W. Chen 14