Digital Design Verification

FYP/Internship Program

Lab # 03

Arrays & Functions

Release Date: 30-July-2024
Name: 10 April 2025

TASK1:

CODE:

```
#include <stdio.h>
#include <math.h>

#define N 100
#define PI 3.14159265

double v[N];

void initialize_signal() {
for (int i = 0; i < N; i++) {
double t = (double)i / N;
v[i] = sin(2 * PI * t);
}
}

double compute_dc_value() {
double sum = 0;
for (int i = 0; i < N; i++) {
sum += v[i];
}
return sum / N;
}

void detect_crossings() {
for (int i = 0; i < N - 1; i++) {
if ((v[i] > 0 && v[i + 1] < 0) || (v[i] < 0 && v[i + 1] > 0)) {
printf("Zero crossing at index %d\n", i);
}
}
}

void detect_glitches() {
for (int i = 1; i < N - 1; i++) {
if (fabs(v[i] - v[i - 1]) > 0.5 && fabs(v[i] - v[i + 1]) > 0.5) {
printf("Possible glitch detected at index %d\n", i);
}
}
}

int main() {
initialize_signal();
printf("DC Value: %lf\n", compute_dc_value());
printf("\nDetecting Zero Crossings:\n");
detect_crossings();
printf("\nDetecting Glitches:\n");
detect_glitches();
return 0;
}
```

SS:

```
# ./main.exe
DC Value: 0.000000

Detecting Zero Crossings:
Zero crossing at index 25

Detecting Glitches:
```

TASK2:

CODE:

String.h

```
#ifndef STRING_H
#define STRING_H

#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAX_LEN 100  // Define MAX_LEN correctly

int findLength(const char *str);
  void toLowercase(char *str);
  void toUppercase(char *str);
int countWords(const char *str);
int countVowels(const char *str);
void vowelFrequency(const char *str, int freq[5]);

#endif // STRING_H
```

String.c

```
#include "string.h"

int findLength(const char *str) {
        return strlen(str);
              }

  void toLowercase(char *str) {
for (int i = 0; str[i] != '\0'; i++) {
        str[i] = tolower(str[i]);
                }
              }

  void toUppercase(char *str) {
for (int i = 0; str[i] != '\0'; i++) {
        str[i] = toupper(str[i]);
                }
              }

int countWords(const char *str) {
     int count = 0, inWord = 0;

for (int i = 0; str[i] != '\0'; i++) {
         if (isspace(str[i])) {
                inWord = 0;
          } else if (!inWord) {
```

```
            inWord = 1;
            count++;
            }
        }
    return count;
    }

int countVowels(const char *str) {
        int count = 0;
    char vowels[] = "aeiouAEIOU";

    for (int i = 0; str[i] != '\0'; i++) {
        if (strchr(vowels, str[i])) {
            count++;
            }
        }
    return count;
    }

void vowelFrequency(const char *str, int freq[5]) {
    for (int i = 0; i < 5; i++) freq[i] = 0;

    for (int i = 0; str[i] != '\0'; i++) {
        switch (tolower(str[i])) {
        case 'a': freq[0]++; break;
        case 'e': freq[1]++; break;
        case 'i': freq[2]++; break;
        case 'o': freq[3]++; break;
        case 'u': freq[4]++; break;
            }
        }
    }
```

Main.c

```
#include <stdio.h>
#include "string.h"

    int main() {
char sentence[MAX_LEN];

printf("Enter a sentence (max 100 characters): ");
    fgets(sentence, MAX_LEN, stdin);

// Removing the newline character if present
    int len = findLength(sentence);
    if (sentence[len - 1] == '\n') {
        sentence[len - 1] = '\0';
        }

printf("\nLength of sentence: %d\n", findLength(sentence));

char lowercase[MAX_LEN], uppercase[MAX_LEN];
    strcpy(lowercase, sentence);
    strcpy(uppercase, sentence);

    toLowercase(lowercase);
    toUppercase(uppercase);

printf("Sentence in lowercase: %s\n", lowercase);
```

```
    printf("Sentence in uppercase: %s\n", uppercase);
    printf("Number of words: %d\n", countWords(sentence));
    printf("Number of vowels: %d\n", countVowels(sentence));

    int freq[5] = {0};
    vowelFrequency(sentence, freq);

    printf("Vowel frequencies: A: %d, E: %d, I: %d, O: %d, U: %d\n",
           freq[0], freq[1], freq[2], freq[3], freq[4]);

    return 0;
}
```

SS:

```
Enter a sentence (max 100 characters): RED DEAD REDEMPTION 2 was the game of the year in 2020

Length of sentence: 54
Sentence in lowercase: red dead redemption 2 was the game of the year in 2020
Sentence in uppercase: RED DEAD REDEMPTION 2 WAS THE GAME OF THE YEAR IN 2020
Number of words: 12
Number of vowels: 16
Vowel frequencies: A: 4, E: 8, I: 2, O: 2, U: 0
```

TASK3:

CODE:

image_processing.h

```
#ifndef IMAGE_PROCESSING_H
#define IMAGE_PROCESSING_H

#include <stdio.h>
#include <stdlib.h>

#define WIDTH 5   // Example width
#define HEIGHT 5  // Example height

// Function Prototypes
void findPixelType(int r, int g, int b);
void rgbToGrayscale(int image[HEIGHT][WIDTH][3], int
grayscale[HEIGHT][WIDTH]);
void applyConvolution(int grayscale[HEIGHT][WIDTH], int
output[HEIGHT][WIDTH]);

#endif
```

image_processing.c

```
#include "image_processing.h"

// Function to find pixel type (0 - Black, 1 - White, 2 - Yellow)
void findPixelType(int r, int g, int b) {
    if (r == 0 && g == 0 && b == 0) {
        printf("Pixel is Black\n");
    } else if (r == 255 && g == 255 && b == 255) {
        printf("Pixel is White\n");
    } else if (r == 255 && g == 255 && b == 0) {
        printf("Pixel is Yellow\n");
    } else {
        printf("Pixel is of an unknown type\n");
    }
}
```

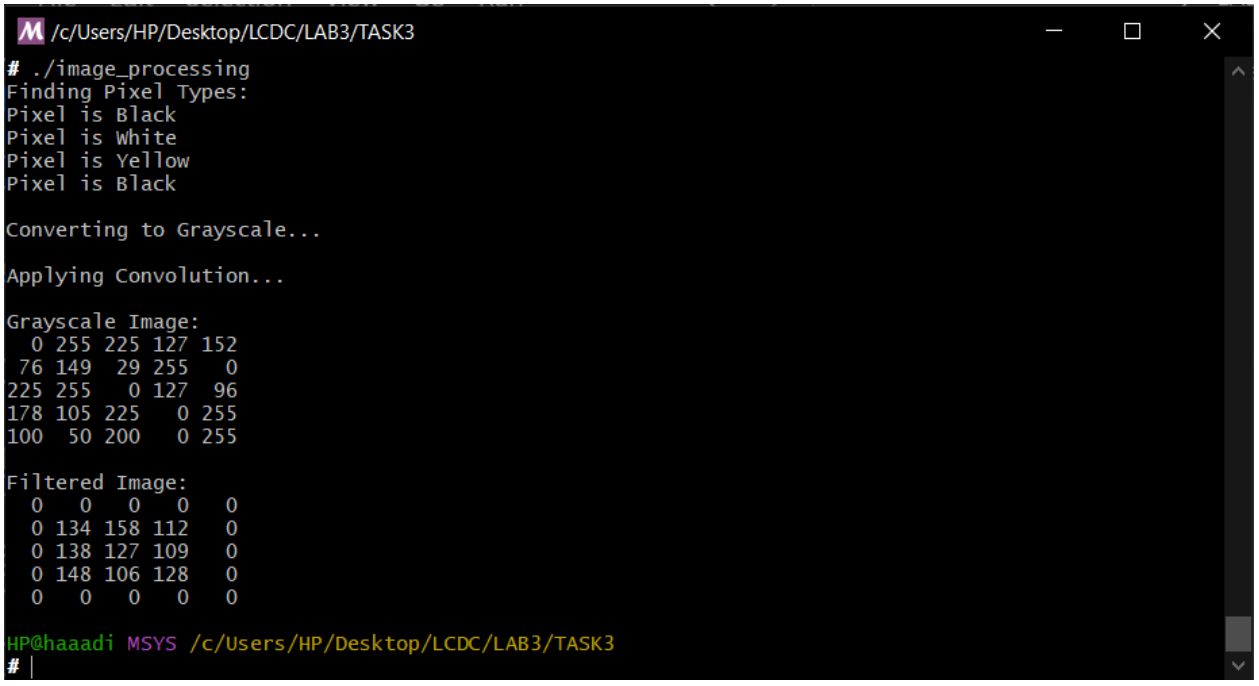
```
// Convert RGB image to Grayscale using a weighted average
   void rgbToGrayscale(int image[HEIGHT][WIDTH][3], int
                grayscale[HEIGHT][WIDTH]) {
              for (int i = 0; i < HEIGHT; i++) {
                for (int j = 0; j < WIDTH; j++) {
                       int r = image[i][j][0];
                       int g = image[i][j][1];
                       int b = image[i][j][2];
      grayscale[i][j] = (0.299 * r + 0.587 * g + 0.114 * b);  //
                    Weighted average
                                }
                              }
                            }

   // Perform 2D Convolution using an averaging filter
  void applyConvolution(int grayscale[HEIGHT][WIDTH], int
                output[HEIGHT][WIDTH]) {
    int filter[3][3] = {{1, 1, 1}, {1, 1, 1}, {1, 1, 1}};
                  int filterSize = 3;
                    int sum, x, y;

                // Initialize output array to zero
                for (int i = 0; i < HEIGHT; i++) {
                  for (int j = 0; j < WIDTH; j++) {
                           output[i][j] = 0;
                                }
                              }

            for (int i = 1; i < HEIGHT - 1; i++) {
               for (int j = 1; j < WIDTH - 1; j++) {
                                sum = 0;
                    for (x = 0; x < filterSize; x++) {
                      for (y = 0; y < filterSize; y++) {
           sum += grayscale[i + x - 1][j + y - 1] * filter[x][y];
                                    }
                                  }
      output[i][j] = sum / 9;  // Normalize with filter sum (9)
                                }
                              }
                            }
```

main.c

```
                #include "image_processing.h"

                        int main() {
                  int image[HEIGHT][WIDTH][3] = {
{{0, 0, 0}, {255, 255, 255}, {255, 255, 0}, {128, 128, 128}, {100,
                            200, 50}},
  {{255, 0, 0}, {0, 255, 0}, {0, 0, 255}, {255, 255, 255}, {0, 0,
                              0}},
{{255, 255, 0}, {255, 255, 255}, {0, 0, 0}, {128, 128, 128}, {50,
                            100, 200}},
{{0, 255, 255}, {255, 0, 255}, {255, 255, 0}, {0, 0, 0}, {255, 255,
                              255}},
 {{100, 100, 100}, {50, 50, 50}, {200, 200, 200}, {0, 0, 0}, {255,
                            255, 255}}
                              };
```

```
    int grayscale[HEIGHT][WIDTH];
    int output[HEIGHT][WIDTH];

    printf("Finding Pixel Types:\n");
    findPixelType(0, 0, 0);          // Black
    findPixelType(255, 255, 255);    // White
    findPixelType(255, 255, 0);      // Yellow
    findPixelType(0,0,0);    // Unknown

    printf("\nConverting to Grayscale...\n");
    rgbToGrayscale(image, grayscale);

    printf("\nApplying Convolution...\n");
    applyConvolution(grayscale, output);

    printf("\nGrayscale Image:\n");
    for (int i = 0; i < HEIGHT; i++) {
        for (int j = 0; j < WIDTH; j++) {
            printf("%3d ", grayscale[i][j]);
                }
            printf("\n");
                }

    printf("\nFiltered Image:\n");
    for (int i = 0; i < HEIGHT; i++) {
        for (int j = 0; j < WIDTH; j++) {
            printf("%3d ", output[i][j]);
                }
            printf("\n");
                }

    return 0;
    }
```

SS:



CONCLUSION:

In this lab we used basic arrays and did some projects using it.