Lectures/Week_13/1_file_print.c

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

int main()
{
  printf("Example of writing a string to txt file:\n");
  // Similarly, declare a pointer to a file
  FILE *filePointer; // Same as int, FILE is a data type
                     // * indicates that this is a pointer to a file
                     // Syntax: FILE *any_name;

  // Open a new file for different task mode
  filePointer = fopen("output.txt", "w"); // Creates file output.txt in write mode
                                          // Syntax: assume "any_name" before
                                          // any_name = fopen("file_name", "mode");

  // Check if the file was opened successfully
  // This is especially important when in read or append mode
  // since the file may not exist
  if (filePointer == NULL)
  // == NULL is to check whether filePointer points to a valid file
  {
    printf("Error: Could not open file.\n");
    return 1; // Exit with an error code
              // remember that main() returns an int
              // return 0 means success, return 1 means error
  }

  // If you are here, it means the file was opened successfully
  int yint = 15.0;
  printf("%d is the value printed to console screen and file\n\n", yint);

  fprintf(filePointer, "%d is the value printed to console screen and file\n", yint);
  // Syntax: supposing you use any_name before
  // fprintf(any_name, CONTENT);
  // CONTENT can be just a string, or a string with variables
  // CONTENT follows the same format as printf
  fprintf(filePointer, "Finished.\n");

  // Close the file
  fclose(filePointer); // This is a must. Syntax: fclose(any_name);
  return 0;
}
```

```
44
45 // Declare ptr to a file
46 // Ptr equals to fopen, in write mode
47 // Check if ptr is NULL
48 // if no, fprintf to the file
49 // finally, fclose the file
```

Lectures/Week_13/2_append_tofile.c

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

int main()
{
  FILE *filePointer2;
  filePointer2 = fopen("output.txt", "a"); // "a" is for appending

  if (filePointer2 == NULL)
  {
    printf("Error: Could not open file.\n");
    return 1;
  }

  fprintf(filePointer2, "This is an appended line. So cool\n");
  printf("Text Added: This is an appended line. Isn't it cool\n\n");
  fclose(filePointer2);
  return 0;
}

// Declare ptr to a file
// Ptr equals to fopen, in append mode
// Check if ptr is NULL
// if no, fprintf to the file (which literally appends)
// finally, fclose the file
```

Lectures/Week_13/3_file_errorcheck.c

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

int main()
{
  FILE *filePointer3;
  char filename[100]; // Declare a long char array to store the filename

  do
  {
    printf("Please enter the name of the file to read first integer from: ");
    scanf("%s", filename);
    filePointer3 = fopen(filename, "r"); // try to open the file in append mode
  } while (filePointer3 == NULL); // Check if the file exists
  // If the file does not exist, the loop will continue to prompt for a new name

  // Read the first integer from the file and store at integer z
  int z;
  fscanf(filePointer3, "%d", &z);
  printf("The first integer in the file is %d\n\n", z);
  fclose(filePointer3); // Close the file
  return 0;
}
```

Lectures/Week_13/4_read_toarray.c

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

int main()
{
  FILE *fileptr4;
  char filename[100];

  do
  {
    printf("please enter a file name:\n");
    scanf("%s", &filename);
    fileptr4 = fopen(filename, "r"); // try to open the file in read mode
  } while (fileptr4 == NULL);

  // Read an array of 12 grades from a file
  int grades[12]; // Declare an array to hold 12 grades
  int max = 0;    // Initialize max to a small number
  for (int i = 0; i < 12; i++)
  {
    fscanf(fileptr4, "%d", &grades[i]); // Read each grade from the file
    // only one %d is needed
    // using for-loop so we can put into the array one by one

    // Below is how to find the maximum grade within the loop
    if (grades[i] > max)
    {
      max = grades[i];
    }
  }
  fclose(fileptr4);
  // Close the file after reading
  // since printing in the following would only need the grades[] array

  printf("The grades are:\n");
  for (int i = 0; i < 12; i++)
  {
    printf("%d\n", grades[i]);
  }
  printf("The maximum grade is %d\n\n", max);

  // Read a 2D array of grades from a file
  fileptr4 = fopen(filename, "r"); // Have to open the file again

  // Declare a 2D array to hold 4 rows and 3 columns of grades
  int grades_2d[4][3];
```

```c
  for (int r = 0; r < 4; r++)
  {
    printf("We are now in row %d\n", r);
    for (int c = 0; c < 3; c++)
    {
      printf("We are now in column %d\n", c);
      fscanf(fileptr4, "%d", &grades_2d[r][c]);

      // Text file structures doesn't matter
      // the structure is decided by the program

      // If your text file looks like:
      // 1 2 3
      // 4 5 6 7 8
      // 9 10 11
      // 12
      // The above code will still work
    }
  }
  fclose(fileptr4);

  // Then print the 2d array in a table format on screen
  printf("\nThe grades in 2D array are:\n");
  for (int r = 0; r < 4; r++) // out loop is for row number
  {
    for (int c = 0; c < 3; c++) // inner loop is for column number
    {
      printf("%d\t", grades_2d[r][c]);
    }
    printf("\n");
  }

  return 0;
}

/* Output of the program:
please enter a file name:
grades.txt
The grades are:
99
88
78
92
100
78
78
89
65
```

```
82
86
67
The maximum grade is 100

We are now in row 0
We are now in column 0
We are now in column 1
We are now in column 2
We are now in row 1
We are now in column 0
We are now in column 1
We are now in column 2
We are now in row 2
We are now in column 0
We are now in column 1
We are now in column 2
We are now in row 3
We are now in column 0
We are now in column 1
We are now in column 2

The grades in 2D array are:
99  88  78
92  100 78
78  89  65
82  86  67
*/
```

Lectures/Week_13/5_read_until_EOF.c

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

int main()
{
  // Read a text file called "unknown_grades.txt"
  FILE *fileptr6;
  fileptr6 = fopen("unknown_grades.txt", "r");

  int long_grade_array[1000]; // Declare a very long array to hold the grades
                              // Assuming the file has at most 1000 grades
  int index = 0;
  int check; // check for EOF, if fscanf is successful, it returns 1
             // if it fails, it returns EOF (EOF equal to -1)
  int sum = 0;
  int min = 100; // Initialize min to a large number

  do
  {
    // instead of using a for-loop and incrementing i
    // we can use a do-while loop and increment index until EOF
    check = fscanf(fileptr6, "%d", &long_grade_array[index]);
    if (check != EOF) // this if-part is only to avoid the last item being added
    {
      sum = sum + long_grade_array[index]; // keep a running sum

      // Update the minimum grade within the loop
      if (long_grade_array[index] < min)
      {
        min = long_grade_array[index];
      }

      index = index + 1; // increment the index
    }
  } while (check != EOF); // or check != -1
  // don't forget the ; at the end of the do-while loop

  fclose(fileptr6); // Close the file after reading

  // After reading, index will be the number of grades read
  // long_grade_array[] holds the grades
  // so we can safely close the file now

  // Print the information
  printf("The number of grades in the file is %d\n", index);
  printf("The first five grades are:\n");
```

```c
  for (int i = 0; i < 5; i++)
  {
    printf("%d\n", long_grade_array[i]);
  }
  printf("The last five grades are:\n");
  for (int i = index - 5; i < index; i++)
  {
    printf("%d\n", long_grade_array[i]);
  }
  double average = (sum + 0.0) / index;
  // +0.0 to convert to double and avoid integer division
  printf("The average grade is %.2f\n", average);
  printf("The minimum grade is %d\n", min);

  return 0;
}

/* Another method with while-loop together with a break statement:

while (1)
{
  success = fscanf(file_ptr, "%d", &large_grade_array[i]);
  if (success == EOF)
  {
    break; // Break out of the loop if EOF is reached
  }
  sum += large_grade_array[i];
  i++;
}

*/
```

Lectures/Week_13/6_quiz5_review.c

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

int main()
{
  printf("Quiz 5 Part A Review\n");
  printf("============================================================\n\n");

  char phrase[50] = "boldly, I bound beyond blue bridges";
  char proceed = 'y', retry = 'n';

  printf("Starting the sequence...\n");

  int p = 72, q = 9, r = 121, s = 0, t, arr[6] = {15, 27, 39, 51, 63, 75};

  do
  {
    t = p / 3;

    switch (s)
    {
    case 3:
      printf("In case 3 now\n");
      t = pow(t - 14, 2);
      printf("t is now %d\n", t);
      proceed = 'y';
      break;

    case 0:
      printf("Handling base case\n");
      t = 0;
      printf("t is %d\n", t);
      proceed = 'y';
      break;

    case 4:
      printf("Executing case 4\n");
      t = p % q;
      arr[s] = sqrt(r);
      printf("t is %d\n", t);
      printf("Array snapshot: ");
      for (int i = 4; i >= 0; i = i - 2)
      {
        printf("%d ", arr[i]);
      }
      printf("\n");
```

```
      proceed = 'z';
      break;

    case 2:
      printf("Inside case 2\n");
      t = t + r;
      printf("t is now %d\n", t);
      proceed = phrase[5];
      break;

    default:
      printf("Default case active\n");
      t = r / q;
      proceed = phrase[18];
      printf("t is %d\n", t);
    }

    s = s + 1;
    retry = proceed;

  } while (retry == 'y');

  printf("Final value of s: %d\n\n", s);

  printf("Classwork from Tuesday: Optional 5 Print 2 3 6 18 108 etc\n");
  printf("=============================================================\n");

  int a, b;
  printf("Enter number a: ");
  scanf("%d", &a); // a=2
  printf("Enter number b: ");
  scanf("%d", &b); // b=3
  printf("\n\n");
  int iteration = 0;

  printf("it\tprod\n");
  printf("------------\n");

  printf("%d\t%d\n", iteration, a); // a=2

  iteration++;
  printf("%d\t%d\n", iteration, b); // b=3

  int c = a * b;

  iteration++;
  printf("%d\t%d\n", iteration, c); // c=6=2*3

```

```
  while (c < 9999999)
  {
    // Update first multiplicand to be the second from the previous iteration
    a = b;

    // Update second multiplicand to the product from the previous iteration
    b = c;

    c = a * b;   // Calculate the new product
    iteration++; // Increment the iteration counter
    printf("%d\t%d\n", iteration, c);
  }

  printf("\nFind the standard deviation of an array\n");
  printf("=============================================================\n");

  int newarr[10] = {23, 45, 12, 56, 9, 34, 67, 89, 2, 15};

  int sum = 0;
  for (int i = 0; i < 10; i++)
  {
    sum += newarr[i]; // Add each element to the sum
  }
  double average = (sum + 0.0) / 10; // avoid integer division

  double result = 0;
  for (int i = 0; i < 10; i++)
  {
    result = result + (newarr[i] - average) * (newarr[i] - average);
    // or using pow()
  }
  double stddev = result / 9; // Sample standard deviation, divide by n-1

  printf("The average is: %.2f\n", average);
  printf("The standard deviation is: %.2f\n\n", stddev);

  printf("Create an output array of same size to store new information\n");
  printf("=============================================================\n");
  // Assuming we know that file "unknown_grades.txt" contains 521 grades
  // Read it to an 1d array first
  // then create another char array to hold the letter grades

  // letter grades are A, B, C, D, F (based on grade>90, 80, 70, 60)
  // write the letter grade array into a file called "letter_grades.txt"

  // we also need to know how many students get above B grade and
  // the percentage of B-above's

```

```
  FILE *fileptr6;

  fileptr6 = fopen("unknown_grades.txt", "r");
  int grade_array[521];
  char letter_grades[521];
  int B_above_count = 0;

  for (int i = 0; i < 521; i++)
  {
    fscanf(fileptr6, "%d", &grade_array[i]); // Read each grade from the file

    int current_grade = grade_array[i];

    if (current_grade >= 80)
    {
      B_above_count++;
      // Count the number of students with grades in the B category
    }

    char current_letter_grade;
    if (current_grade >= 90)
    {
      current_letter_grade = 'A';
    }
    else if (current_grade >= 80)
    {
      current_letter_grade = 'B';
    }
    else if (current_grade >= 70)
    {
      current_letter_grade = 'C';
    }
    else if (current_grade >= 60)
    {
      current_letter_grade = 'D';
    }
    else
    {
      current_letter_grade = 'F';
    }
    letter_grades[i] = current_letter_grade;
    // Store the letter grade in the array
  }
  fclose(fileptr6); // Close the file after reading

  // Now, write the letter grades to a new file
  FILE *output_file;
  output_file = fopen("letter_grades.txt", "w");
```

```
  if (output_file == NULL)
  {
    printf("Error: Could not open output file.\n");
    return 1; // Exit with an error code
  }
  for (int i = 0; i < 521; i++)
  {
    fprintf(output_file, "%c\n", letter_grades[i]);
    // Write each letter grade to the file
  }
  fclose(output_file); // Close the output file

  printf("The number of students with grades above B: %d\n", B_above_count);
  double percentage_B_above = (B_above_count / 521.0) * 100;
  // Calculate the percentage, use 521.0 to avoid integer division
  printf("Percentage of students with grades above B: %.2f%%\n\n",
percentage_B_above);

  printf("Load an irregular text file to 2d array\n");
  printf("=============================================================\n");

  // If your text file looks like:
  // 1 2 3
  // 4 5 6 7 8
  // 9 10 11
  // 12
  // The above code will still work

  // Read the file into a 2D array of 3*4
  FILE *fileptr;
  char filename[100] = "random.txt";

  fileptr = fopen(filename, "r"); // try to open the file in read mode

  int new_grid[3][4];

  for (int j = 0; j < 3; j++)
  {
    for (int i = 0; i < 4; i++)
    {
      fscanf(fileptr, "%d", &new_grid[j][i]);
    } // Read the first element into first row first col
  }
  fclose(fileptr);

  // Print the 2D array
  printf("The 2D array is:\n");
  for (int j = 0; j < 3; j++)
```

```
  {
    for (int i = 0; i < 4; i++)
    {
      printf("%d ", new_grid[j][i]);
    }
    printf("\n");
  }
  printf("\n");
}

/**
Output of the program:


Quiz 5 Part A Review
================================================================

Starting the sequence...
Handling base case
t is 0
Default case active
t is 13
Inside case 2
t is now 145
In case 3 now
t is now 100
Executing case 4
t is 0
Array snapshot: 11 39 15
Final value of s: 5

Classwork from Tuesday: Optional 5 Print 2 3 6 18 108 etc
================================================================

Enter number a: 2
Enter number b: 3

it  prod
--------------
0 2
1 3
2 6
3 18
4 108
5 1944
6 209952
7 408146688
```

```
Find the standard deviation of an array
============================================================

The average is: 35.20
The standard deviation is: 813.29

Create an output array of same size to store new information
============================================================

The number of students with grades above B: 229
Percentage of students with grades above B: 43.95%

Load an irregular text file to 2d array
============================================================

The 2D array is:
1 2 3 4
5 6 7 8
9 10 11 12

 */
```