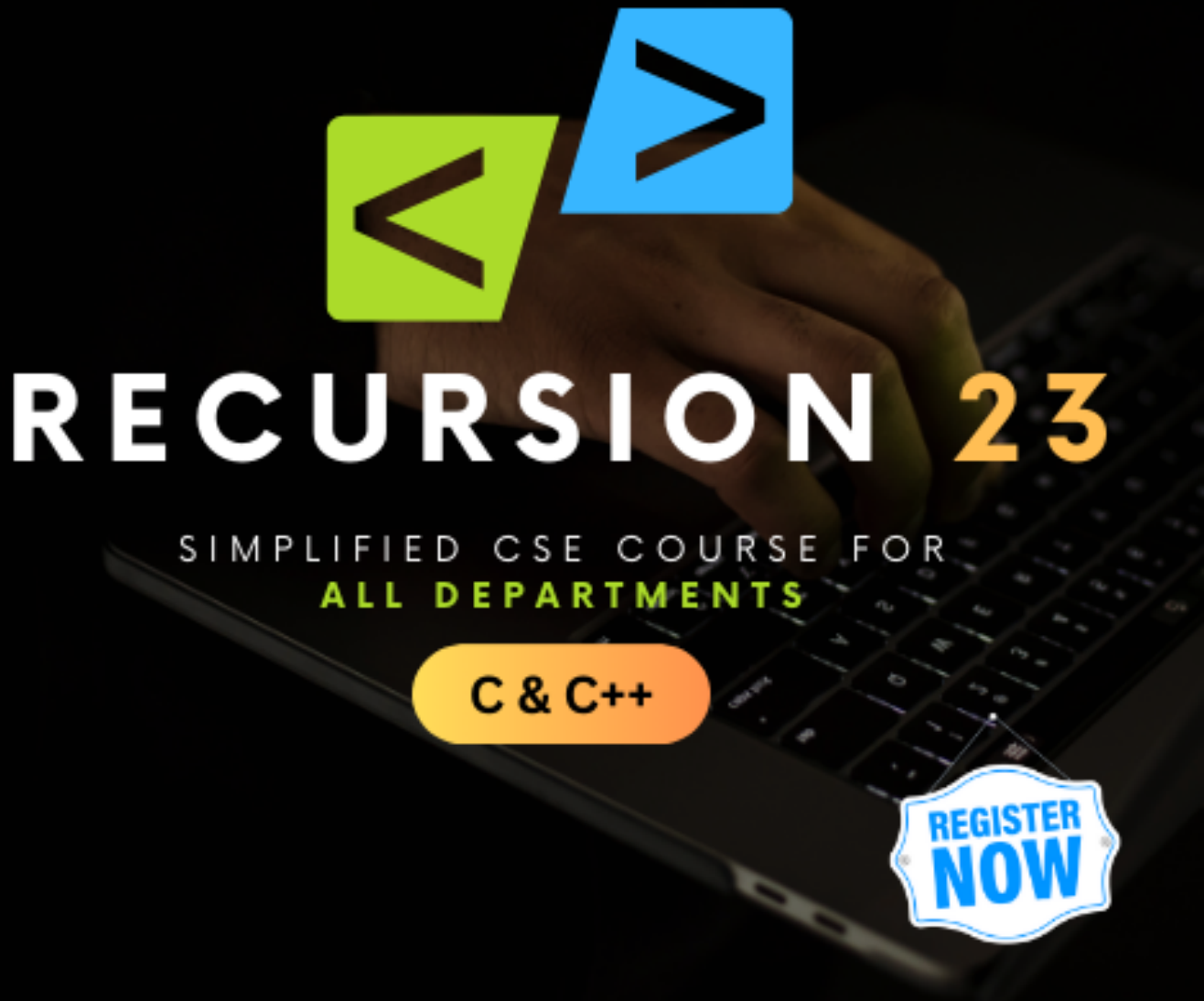
Lecture 26

File and Extra



File Write

```c
#include <stdio.h>

int main() {
    // Declare a file pointer
    FILE *fptr;

    // Open a file in write mode ("w")
    fptr = fopen("output.txt", "w");

    // Check if the file was opened successfully
    if (fptr == NULL) {
        printf("Error: Could not open file for writing.\n");
        return 1;
    }

    // Write a string to the file
    fprintf(fptr, "Hello, World!\n");

    // Write formatted data to the file
    int num = 42;
    float pi = 3.14159;
    fprintf(fptr, "Number: %d\n", num);
    fprintf(fptr, "Pi: %.5f\n", pi);

    // Close the file
    fclose(fptr);

    printf("Data written to file successfully.\n");

    return 0;
}
```

fopen() modes

File Opening Modes

Mode	Meaning	Description
r	Read	Only reading possible. Not create file if not exist
w	Write	Only writing possible. Create file if not exist otherwise erase the old content of file and open as a blank file
a	Append	Only writing possible. Create file if not exist, otherwise open file and write from the end of file (do not erase the old content)
r+	Reading + Writing	R & W possible. Create file if not exist. Overwriting existing data. Used for modifying content
w+	Reading + Writing	R & W possible. Create file if not exist. Erase old content.
a+	Reading + Appending	R & W possible. Create file if not exist. Append content at the end of file

Use Case

Problem: Read From a file and output on another

```
Read two numbers from in.txt and output their sum in
out.txt
```

File read and write

```c
#include <stdio.h>
int main() {
    int num1, num2, sum;
    FILE *inputFile, *outputFile;
    // Open in.txt for reading
    inputFile = fopen("in.txt", "r");
    if (inputFile == NULL) {
        printf("Error: Could not open in.txt for reading.\n");
        return 1;
    }
    // Read two integers from in.txt
    fscanf(inputFile, "%d %d", &num1, &num2);
    // Close the input file after reading
    fclose(inputFile);
    // Calculate the sum of the two numbers
    sum = num1 + num2;
    // Open out.txt for writing
    outputFile = fopen("out.txt", "w");
    if (outputFile == NULL) {
        printf("Error: Could not open out.txt for writing.\n");
        return 1;
    }
    // Write the sum to out.txt
    fprintf(outputFile, "Sum: %d\n", sum);
    // Close the output file after writing
    fclose(outputFile);
    printf("Sum of %d and %d is %d. Result written to out.txt\n", num1, num2,
sum);return 0;
}
```

Reading till EOF

```c
#include <stdio.h>

int main() {
    FILE *file;
    int number;

    // Open the file for reading
    file = fopen("numbers.txt", "r");

    // Check if the file was opened successfully
    if (file == NULL) {
        printf("Error: Could not open file.\n");
        return 1;
    }

    // Read and print numbers from the file until EOF
    while (fscanf(file, "%d", &number) != EOF) {
        printf("%d\n", number);
    }

    // Close the file
    fclose(file);

    return 0;
}
```

fgets() review

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

#define MAX_LINE_LENGTH 100

int main() {
    char line[MAX_LINE_LENGTH];

    // Prompt the user to enter a line of text
    printf("Enter a line of text: ");

    // Read a line of input from the console using fgets
    if (fgets(line, sizeof(line), stdin) != NULL) {
        // Remove the newline character if present
        size_t len = strlen(line);
        if (len > 0 && line[len-1] == '\n') {
            line[len-1] = '\0';
        }

        // Print the line back to the console without the newline
        printf("You entered: %s\n", line);
    } else {
        printf("Error reading input.\n");
    }

    return 0;
}
```

Reading till EOF

```c
#include <stdio.h>

#define MAX_LINE_LENGTH 100

int main() {
    FILE *file;
    char line[MAX_LINE_LENGTH];

    // Open the file for reading
    file = fopen("lines.txt", "r");

    // Check if the file was opened successfully
    if (file == NULL) {
        printf("Error: Could not open file.\n");
        return 1;
    }

    // Read and print lines from the file until EOF
    while (fgets(line, sizeof(line), file) != NULL)
{       printf("%s", line);
    }

    // Close the file
    fclose(file);

    return 0;
}
```

sscanf()

```c
#include <stdio.h>

#define MAX_LINE_LENGTH 100

int main() {
    FILE *file;
    char line[MAX_LINE_LENGTH];
    int number;
    char text[50];

    // Open the file for reading
    file = fopen("data.txt", "r");

    // Check if the file was opened successfully
    if (file == NULL) {
        printf("Error: Could not open file.\n");
        return 1;
    }

    // Read and parse lines from the file
    while (fgets(line, sizeof(line), file) != NULL) {
        // Parse the line using sscanf
        sscanf(line, "%d %49s", &number, text);
        // Print the parsed data
        printf("Number: %d, Text: %s\n", number,
text});

    // Close the file
    fclose(file);

    return 0;
}
```

Time to run a code

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

int main() {
    // Variables to store start and end times
    clock_t start, end;
    double cpu_time_used;

    // Start the clock
    start = clock();

    // Perform a loop till 1,000,000
    for (int i = 0; i < 1000000; i++) {
        // Loop body (currently empty)
    }

    // Stop the clock
    end = clock();

    // Calculate the time taken
    cpu_time_used = ((double) (end - start)) / CLOCKS_PER_SEC;

    // Print the time taken
    printf("Time taken by the loop: %f seconds\n",
cpu_time_used);
    return 0;
}
```

Random Number Generator

```c
#include <stdio.h>
#include <stdlib.h>  // Needed for rand() and srand()
#include <time.h>    // Needed for time()

int main() {
    // Seed the random number generator with the current time
    srand(time(0));

    // Generate a random number between 0 and RAND_MAX
    int random_number = rand();

    // Print the random number
    printf("Random number: %d\n", random_number);

    // Generate a random number between 1 and 100
    int random_number_1_to_100 = rand() % 100 + 1;

    // Print the random number between 1 and 100
    printf("Random number between 1 and 100: %d\n",
random_number_1_to_100);
    return 0;
}
```

You can create your own header

Ternary operator

```
condition ? expression_if_true : expression_if_false;
```

Ternary operator

```c
#include <stdio.h>

int main() {
    int a = 10, b = 20;
    int max;

    // Using ternary operator to find the maximum
    max = (a > b) ? a : b;

    printf("Maximum of %d and %d is %d\n", a, b,
max);
    return 0;
}
```

Ternary operator

```c
#include <stdio.h>

int main() {
    int num = 15;

    // Using ternary operator to check even or odd
    printf("%d is %s\n", num, (num % 2 == 0) ? "Even" :
"Odd");
    return 0;
}
```

Chaining

```c
#include <stdio.h>

int main() {
    int marks = 85;
    char grade;

    // Using ternary operator to assign grade
    grade = (marks ≥ 90) ? 'A' :
            (marks ≥ 80) ? 'B' :
            (marks ≥ 70) ? 'C' :
            (marks ≥ 60) ? 'D' : 'F';

    printf("Grade: %c\n", grade);

    return 0;
}
```

Constants

```c
#include <stdio.h>

// Define PI as 3.14159
#define PI 3.14159


int main() {
    // Output the value of PI
    printf("The value of PI is: %f\n",
PI);
    return 0;
}
```

Macro

```c
#include <stdio.h>

// Define MAX as the maximum of two numbers, x and y
#define MAX(x, y) ((x) > (y) ? (x) : (y))

int main() {
    int a = 15, b = 25;

    // Use MAX to find the maximum of two numbers
    int max_value = MAX(a, b);

    // Output the maximum value
    printf("The maximum of %d and %d is: %d\n", a, b,
max_value);
    return 0;
}
```

Multiline Macro

```c
#include <stdio.h>

// Define SWAP_ARITHMETIC macro without do-while
#define SWAP_ARITHMETIC(x, y) { \
    (x) = (x) + (y); \
    (y) = (x) - (y); \
    (x) = (x) - (y); \
}

int main() {
    int a = 5, b = 10;

    // Print values before swapping
    printf("Before swapping: a = %d, b = %d\n", a, b);

    // Use SWAP_ARITHMETIC macro to swap the values of a and
b   SWAP_ARITHMETIC(a, b);

    // Print values after swapping
    printf("After swapping: a = %d, b = %d\n", a, b);

    return 0;
}
```

Enum

```c
#include <stdio.h>

// Define an enum for game states
enum GameState {
    START_MENU,     // 0
    PLAYING,        // 1
    PAUSED,         // 2
    GAME_OVER       // 3
};

int main() {
    // Declare a variable of type enum GameState
    enum GameState currentState;

    // Set the current state to PLAYING
    currentState = PLAYING;

    // Print the current state
    printf("Current game state: %d\n", currentState); // Output: Current game state:
1
    // Change the state to GAME_OVER
    currentState = GAME_OVER;

    // Print the new state
    printf("New game state: %d\n", currentState); // Output: New game state: 3

    return 0;
}
```

Switch Case

```c
// C Program to create a simple calculator
#include <stdio.h>
#include <stdlib.h>
int main()
{
    // switch variable
    char choice;
    // operands
    int x, y;
    while (1) {
        printf("Enter the Operator (+,-,*,/)\nEnter x to
"              "exit\n");
        scanf(" %c", &choice);
        // for exit
        if (choice == 'x') {
            exit(0);
        }
        printf("Enter the two numbers: ");
        scanf("%d %d", &x, &y);
        // switch case with operation for each operator
        switch (choice) {
          case '+':
              printf("%d + %d = %d\n", x, y, x + y);
              break;
          case '-':
              printf("%d - %d = %d\n", x, y, x - y);
              break;
          case '*':
              printf("%d * %d = %d\n", x, y, x * y);
              break;
          case '/':
              printf("%d / %d = %d\n", x, y, x / y);
              break;
          default:
              printf("Invalid Operator Input\n");
        }
    }
    return 0;
}
```

Problem

Use enum and switch case to output what day is today.

```c
#include <stdio.h>
// Define an enum for the days of the week
enum Day {
    SUNDAY,    // Automatically assigned the value
0   MONDAY,    // Automatically assigned the value
1   TUESDAY,   // Automatically assigned the value
2   WEDNESDAY, // Automatically assigned the value
3   THURSDAY,  // Automatically assigned the value
4   FRIDAY,    // Automatically assigned the value
5   SATURDAY   // Automatically assigned the value
};
```

```c
int main() {
    // Declare a variable of type enum Day
    enum Day today;
    // Assign a value to the enum variable
    today = WEDNESDAY;
    // Use a switch statement to print the day of the week
    switch (today) {
        case SUNDAY:
            printf("Today is Sunday.\n");
            break;
        case MONDAY:
            printf("Today is Monday.\n");
            break;
        case TUESDAY:
            printf("Today is Tuesday.\n");
            break;
        case WEDNESDAY:
            printf("Today is Wednesday.\n");
            break;
        case THURSDAY:
            printf("Today is Thursday.\n");
            break;
        case FRIDAY:
            printf("Today is Friday.\n");
            break;
        case SATURDAY:
            printf("Today is Saturday.\n");
            break;
        default:
            printf("Invalid day.\n");
            break;
    }
    return 0;
}
```