4. Input and Output

- Purpose of input and output
- Formatted I/O
  - printf()
  - scanf()
- Format specifiers
  - %d, %f, %c, %s
- Escape sequences (\n, \t)
- Common I/O mistakes (wrong format specifier)

4.1 Introduction to Data Input and Output in C

Concept of Input and Output:

- **Input**: Process of taking data from the user or input device into the program
- **Output**: Process of displaying results to the user or output device

Standard Input and Output: C provides standard streams:

- Standard Input (stdin) → Keyboard
- Standard Output (stdout) → Screen

These streams are defined in the stdio.h header file.

Format specifiers connect variables to I/O functions

Example:

```
printf() → Used for displaying output
scanf() → Used for taking input
```

4.2 Unformatted Input and Output in C

Concept of Input and Output: Unformatted I/O functions are used to read or write data without any formatting. They mainly deal with characters and strings.

4.2.1 Character Input and Output

Character Output:

- putchar() → outputs a single character

Example:

```
putchar('A');
```

Character Input:

- getchar() → reads a single character from keyboard

Example:

```
char ch;
ch = getchar();
```

4.2 Unformatted Input and Output in C

4.2.1 Character Input and Output

- **Read and print a single character using getchar(), putchar()**

Example:

```
#include <stdio.h>

int main() {
    char ch;
    printf("Enter a character: ");
    ch = getchar();
    putchar(ch);
    return 0;
}
```

4.2 Unformatted Input and Output in C

4.2.2 String Input and Output

String Output:

- puts() → prints a string and moves to new line

Example:

```
puts("Hello World");
```

String Input:

- gets() → reads a string
  - Not recommended (unsafe)
- fgets() → safe alternative

Example:

```
char name[20];
fgets(name, 20, stdin);
```

4.2 Unformatted Input and Output in C

4.2.2 String Input and Output

- **Read a string using fgets()**
- **Display the string using puts()**

Example:

```
#include <stdio.h>

int main() {
    char name[20];
    printf("Enter your name: ");
    fgets(name, 20, stdin);
    puts(name);
    return 0;
}
```

4.3 Formatted Input and Output in C

Definitions: Formatted I/O allows data to be read or displayed in a specific format using format specifiers.
The most commonly used formatted I/O functions are:

- printf() → output
- scanf() → input

4.3.1 Control String

Field Width & Precision:

- `printf("%6.2f", 12.3456);`
- Field width → 6
- Precision → 2

Specifier	Data Type
%d	Integer
%f	Float
%c	Char
%s	String
%lf	Double

4.3.2 printf() Function

Syntax:

```
printf("control string", variables);
```

Example:

```
int a = 10;
float b = 3.14;
printf("a = %d, b = %.2f", a, b);
```

4.3 Formatted Input and Output in C

4.3.3 scanf() Function

Syntax:

```
scanf("control string", &variables);
```

Example:

```
int x;
float y;
scanf("%d %f", &x, &y);
```

Practice:

- Input and print student details
- Read two numbers and print sum with precision
- Format output using width and precision

Specifier	Data Type
%d	Integer
%f	Float
%c	Char
%s	String
%lf	Double