✅ UNIT – V : File Handling, Searching & Sorting

PART – A : FILE HANDLING

---

1. Text and Binary Files – Introduction

File

A file is a collection of data stored permanently on secondary storage (hard disk).

Types of Files

1. Text Files
2. Binary Files

---

2. Input / Output Files

Purpose

- Store data permanently
- Read and write large data
- Used in databases and applications

---

3. File Operations

Operation	Function
Open file	fopen()
Close file	fclose()
Read file	fscanf(), fgets(), fread()
Write file	fprintf(), fputs(), fwrite()

---

4. File Modes

Mode	Meaning
"r"	Read
"w"	Write

Mode Meaning

"a" Append

"rb" Read binary

"wb" Write binary

---

5. Text Files

Example – Writing to a Text File

```
FILE *fp;
fp = fopen("data.txt", "w");
fprintf(fp, "Hello World");
fclose(fp);
```

Reading from a File

```
char ch;
fp = fopen("data.txt", "r");
while((ch = fgetc(fp)) != EOF)
    printf("%c", ch);
fclose(fp);
```

---

6. Binary Files

Definition

Binary files store data in binary format (faster and efficient).

Example

```
FILE *fp;
int num = 100;

fp = fopen("data.bin", "wb");
fwrite(&num, sizeof(num), 1, fp);
fclose(fp);
```

Reading Binary File

```
fp = fopen("data.bin", "rb");
fread(&num, sizeof(num), 1, fp);
```

---

7. Searching a Database

Definition

Searching means finding a required record from a file or array.

Example (Simple Search in File)

```
while(fread(&s, sizeof(s), 1, fp)) {
    if(s.id == key)
        printf("Found");
}
```

PART – B : SEARCHING & SORTING

---

8. Linear Search

Definition

Searches elements one by one.

Algorithm

1. Start from first element
2. Compare with key
3. Stop if found

Code

```
int linearSearch(int a[], int n, int key) {
    for(int i=0;i<n;i++) {
        if(a[i] == key)
            return i;
    }
    return -1;
}
```

---

9. Binary Search

Definition

Searches by dividing array into halves (only for sorted arrays).

Algorithm

1. Find middle element
2. Compare with key
3. Move left or right

Code

```
int binarySearch(int a[], int n, int key) {
   int low=0, high=n-1;
   while(low <= high) {
      int mid = (low + high) / 2;
      if(a[mid] == key)
         return mid;
      else if(a[mid] < key)
         low = mid + 1;
      else
         high = mid - 1;
   }
   return -1;
}
```

---

10. Bubble Sort

Definition

Compares adjacent elements and swaps them.

Algorithm

```
for(int i=0;i<n-1;i++)
   for(int j=0;j<n-i-1;j++)
```

```
        if(a[j] > a[j+1])
            swap;
```

---

11. Insertion Sort

Definition

Places element in correct position.

Code

```
for(int i=1;i<n;i++) {
    int key = a[i];
    int j = i - 1;
    while(j >= 0 && a[j] > key) {
        a[j+1] = a[j];
        j--;
    }
    a[j+1] = key;
}
```

---

12. Selection Sort

Definition

Selects minimum element and swaps.

Code

```
for(int i=0;i<n-1;i++) {
    int min = i;
    for(int j=i+1;j<n;j++)
        if(a[j] < a[min])
            min = j;

    int temp = a[i];
    a[i] = a[min];
```

```
    a[min] = temp;
}
```

---

13. Comparison of Sorting Techniques

Algorithm	Time Complexity	Best Case
Bubble	$O(n^2)$	$O(n)$
Insertion	$O(n^2)$	$O(n)$
Selection	$O(n^2)$	$O(n^2)$

Text Diagram – Sorting

Before: 5 3 1 4 2

After: 1 2 3 4 5

Important Keywords (For Exams)

- File pointer
- Text file
- Binary file
- Searching
- Sorting
- Linear search
- Binary search
- Bubble sort
- Insertion sort
- Selection sort