**1. Structures**

* **Definition**: Structures are user-defined data types that allow grouping of variables of different types under a single name.
* **Syntax**:

c

struct StructureName {

dataType member1;

dataType member2;

};

* **Accessing Members**: Use the dot operator . to access structure members.

**2. Unions**

* **Definition**: Unions allow storing different data types in the same memory location. Unlike structures, all members share the same memory space, and only one member can hold a value at any given time.
* **Syntax**:

c

union UnionName {

dataType member1;

dataType member2;

};

* **Usage**: Save memory when storing variables of different types, but only one value can be used at a time.

**3. Array of Structures**

* **Definition**: You can create arrays where each element is a structure. This allows you to handle large collections of structures easily.
* **Syntax**:

c

struct StructureName arrayName[arraySize];

* **Accessing Members**: Use array indexing followed by the dot operator.

**4. Enumerations**

* **Definition**: Enumeration (enum) is a user-defined data type that consists of integral constants. It improves code readability by assigning names to integer values.
* **Syntax**:

c

enum EnumName {constant1, constant2, constant3};

* **Example**:

c

enum week {Sunday, Monday, Tuesday};

**5. File Handling**

* **Opening a File**: Use fopen() to open a file.

c

FILE \*fp = fopen("filename.txt", "mode");

* **Closing a File**: Use fclose() to close the file.

c

fclose(fp);

* **Opening Modes**:
  + "r": Read (file must exist).
  + "w": Write (creates new or overwrites existing file).
  + "a": Append.
  + "r+": Read/Write.
  + "w+": Write/Read.
  + "a+": Append/Read.
* **Reading from and Writing to a File**:
  + **Reading**: Use fscanf(), fgets(), fgetc().
  + **Writing**: Use fprintf(), fputs(), fputc().
* **Copying File Content**:

c

FILE \*source = fopen("source.txt", "r");

FILE \*destination = fopen("destination.txt", "w");

// Copy content

fclose(source);

fclose(destination);

**6. Command Line Arguments**

* **Definition**: These are parameters passed to the program at runtime via the command line.
* **argc**: Argument count (number of arguments passed).
* **argv**: Argument vector (array of arguments).

c

int main(int argc, char \*argv[]) {

// Access arguments

}

**7. Pre-processor Directives**

* **Definition**: Pre-processor directives are instructions given to the compiler to process before actual compilation starts.
* **Common Directives**:
  + #define: Defines constants.
  + #include: Includes header files.
  + #ifdef, #ifndef: Conditional compilation.
  + **Example**:

c

#define PI 3.14

#include <stdio.h>