1. Main:
    a. The program includes a header file named "fat32Dir.h", contains declarations and definitions needed for working with FAT32 file systems.
    b. In the `main()` function:
        i. It declares an array `sector` of size 512 bytes.
        ii. Calls `ReadSector(L"\\\\.\\F:", 0, sector);`. This function seems to read the first sector of the disk represented by `"\\\\.\\F:"` and stores the data in the `sector` array.
    c. It checks if the disk is NTFS or FAT32 by calling the function `isNTFSorFAT32(sector)`. If the function returns 0, it indicates NTFS; if it returns 1, it indicates FAT32.
    d. If the disk is FAT32, it proceeds to do the following:
        i. It declares an integer variable `option`.
        ii. It declares an instance of the `FATbootSector` class named `disk1`.
        iii. It calls `disk1.getInfo(L"\\\\.\\F:")`, which seems to retrieve information about the FAT32 volume.
        iv. If the information retrieval is successful, it enters a loop where the user is presented with a menu of options:
            1. Option 1: Display volume information.
            2. Option 2: Display the root directory.
            3. Option -1: Exit the program.
        v. Depending on the user's input, it either displays volume information, displays the root directory, or exits the loop.
    e. The program returns 0, indicating successful execution.
    f.
2. FATbootSector
    a. **ReadSector function**: This function reads a sector from a specified drive and stores the data in the provided buffer.
    b. **hexToString function**: Converts a byte array to a string
    c. **clearExcessSpace function**: Removes excess space characters from a string.
    d. **firstSectorofCluster function**: Calculates the first sector of a cluster based on the FAT32 boot sector information.
    e. **isNTFSorFAT32 function**: Determines whether the file system is NTFS or FAT32 by examining specific signatures in the boot sector.
    f. **getListClusters function**: Retrieves a list of cluster numbers starting from a given cluster in the FAT32 file system.
    g. **FATbootSector class**: This class represents the boot sector of a FAT32 file system. It contains methods for initializing the boot sector information, displaying information about the boot sector, and accessing specific parameters of the boot sector.
        i. **getInfo method**: Reads the boot sector from the specified disk location and initializes the boot sector information.
        ii. **showInfo method**: Displays information about the boot sector.

iii. **getBytesPerSec, getSecPerClus, getBootSecSize, getFirstRootClus, getFirstDataSector methods**: Accessors for specific parameters of the boot sector.

3. fat32Dir

   a. **File class**: Represents a file in the FAT32 file system. It stores attributes such as file name, extension, attributes, first cluster, and file size.
   b. **Constructor**: Initializes a File object with default values.
   c. **getFiles function**: Retrieves information about files from a specified cluster in the FAT32 file system. It reads directory entries, parses them, and populates a vector of File objects.
   d. **convertAttrNumToAttrString function**: Converts file attributes from numeric representation to a human-readable string format.
   e. **interactFile function**: Interacts with a specific file. If the file is a text file (identified by its extension), it reads its content and prints it to the console.
   f. **Directory function**: Displays a list of files and subdirectories in a specified directory cluster. It allows the user to navigate through the directory structure, interact with files, or enter subdirectories recursively.