1. **C Program to illustrate opening of a File**

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
  // file pointer variable to store the value returned by
  // fopen
  FILE* fptr;

  // opening the file in read mode
  fptr = fopen("filename.txt", "r");

  // checking if the file is opened successfully
  if (fptr == NULL) {
    printf("The file is not opened. The program will "
        "now exit.");
    exit(0);
  }

  return 0;
}
```

2. **C Program to create a file**

```
#include <stdio.h>
#include <stdlib.h>

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

  // creating file using fopen() access mode "w"
  fptr = fopen("file.txt", "w");

  // checking if the file is created
  if (fptr == NULL) {
    printf("The file is not opened. The program will "
        "exit now");
    exit(0);
  }
  else {
    printf("The file is created Successfully.");
  }
```

```
    return 0;
}
```

3. **C Program to create a file, write in it and close the file**

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

int main()
{

    // Declare the file pointer
    FILE* filePointer;

    // Get the data to be written in file
    char dataToBeWritten[50] = "GeeksforGeeks-A Computer "
                               "Science Portal for Geeks";

    // Open the existing file GfgTest.c using fopen()
    // in write mode using "w" attribute
    filePointer = fopen("GfgTest.c", "w");

    // Check if this filePointer is null
    // which maybe if the file does not exist
    if (filePointer == NULL) {
        printf("GfgTest.c file failed to open.");
    }
    else {

        printf("The file is now opened.\n");

        // Write the dataToBeWritten into the file
        if (strlen(dataToBeWritten) > 0) {

            // writing in the file using fputs()
            fputs(dataToBeWritten, filePointer);
            fputs("\n", filePointer);
        }

        // Closing the file using fclose()
        fclose(filePointer);

        printf("Data successfully written in file "
               "GfgTest.c\n");
        printf("The file is now closed.");
    }

    return 0;
}
```

4. C Program to open a file, read from it and close the file

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

int main()
{

    // Declare the file pointer
    FILE* filePointer;

    // Declare the variable for the data to be read from
    // file
    char dataToBeRead[50];

    // Open the existing file GfgTest.c using fopen()
    // in read mode using "r" attribute
    filePointer = fopen("GfgTest.c", "r");

    // Check if this filePointer is null
    // which maybe if the file does not exist
    if (filePointer == NULL) {
        printf("GfgTest.c file failed to open.");
    }
    else {

        printf("The file is now opened.\n");

        // Read the dataToBeRead from the file
        // using fgets() method
        while (fgets(dataToBeRead, 50, filePointer)
               != NULL) {

            // Print the dataToBeRead
            printf("%s", dataToBeRead);
        }

        // Closing the file using fclose()
        fclose(filePointer);

        printf(
            "Data successfully read from file GfgTest.c\n");
        printf("The file is now closed.");
    }
    return 0;
}
```

5. **C Program to write to a Binary file**

```
#include <stdio.h>
#include <stdlib.h>
struct threeNum {
    int n1, n2, n3;
};
int main()
{
    int n;
    // Structure variable declared here.
    struct threeNum num;
    FILE* fptr;
    if ((fptr = fopen("C:\\program.bin", "wb")) == NULL) {
        printf("Error! opening file");
        // If file pointer will return NULL
        // Program will exit.
        exit(1);
    }
    int flag = 0;
    // else it will return a pointer to the file.
    for (n = 1; n < 5; ++n) {
        num.n1 = n;
        num.n2 = 5 * n;
        num.n3 = 5 * n + 1;
        flag = fwrite(&num, sizeof(struct threeNum), 1,
                      fptr);
    }

    // checking if the data is written
    if (!flag) {
        printf("Write Operation Failure");
    }
    else {
        printf("Write Operation Successful");
    }

    fclose(fptr);

    return 0;
}
```

6. C Program to read from a Binary file

```
#include <stdio.h>
#include <stdlib.h>
struct threeNum {
   int n1, n2, n3;
};
int main()
{
   int n;
   struct threeNum num;
   FILE* fptr;
   if ((fptr = fopen("C:\\program.bin", "rb")) == NULL) {
      printf("Error! opening file");
      // If file pointer will return NULL
      // Program will exit.
      exit(1);
   }
   // else it will return a pointer to the file.
   for (n = 1; n < 5; ++n) {
      fread(&num, sizeof(struct threeNum), 1, fptr);
      printf("n1: %d\tn2: %d\tn3: %d\n", num.n1, num.n2,
            num.n3);
   }
   fclose(fptr);

   return 0;
}
```

7. C Program to illustrate the use of *fseek()*

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
   FILE* fp;
   fp = fopen("test.txt", "r");

   // Moving pointer to end
   fseek(fp, 0, SEEK_END);

   // Printing position of pointer
   printf("%ld", ftell(fp));
```

```
    return 0;
}
```

8. C Program to illustrate the use of *rewind()*

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
    FILE* fptr;
    fptr = fopen("file.txt", "w+");
    fprintf(fptr, "Geeks for Geeks\n");

    // using rewind()
    rewind(fptr);

    // reading from file
    char buf[50];
    fscanf(fptr, "%[^\n]s", buf);

    printf("%s", buf);

    return 0;
}
```