**Exercise –IV :File Management in C and Dynamic Memory Allocation**

***♣ File Management in C***

**1. Write a program to read data from a file and display it on the screen.**

#include <stdio.h>

int main() {

FILE \*file;

char ch;

file = fopen("sample.txt", "r");

if (file == NULL) {

printf("Error opening file.\n");

return 1;

}

while ((ch = fgetc(file)) != EOF) {

putchar(ch);

}

fclose(file);

return 0;

}

**2. Write a program to write data to a file.**

#include <stdio.h>

int main() {

FILE \*file;

char data[] = "Hello, World!";

file = fopen("output.txt", "w");

if (file == NULL) {

printf("Error opening file.\n");

return 1;

}

fprintf(file, "%s", data);

fclose(file);

return 0;

}

**3. Write a program to read a binary file and display its contents on the screen.**

#include <stdio.h>

int main() {

FILE \*file;

char buffer[100];

file = fopen("binary\_data.bin", "rb");

if (file == NULL) {

printf("Error opening file.\n");

return 1;

}

fread(buffer, sizeof(char), sizeof(buffer), file);

printf("%s", buffer);

fclose(file);

return 0;

}

**4. Write a program to write data to a binary file.**

#include <stdio.h>

int main() {

FILE \*file;

char data[] = "Binary Data";

file = fopen("binary\_output.bin", "wb");

if (file == NULL) {

printf("Error opening file.\n");

return 1;

}

fwrite(data, sizeof(char), sizeof(data), file);

fclose(file);

return 0;

}

**5. Write a program to append data to an existing file.**

#include <stdio.h>

int main() {

FILE \*file;

char data[] = " Appended Data";

file = fopen("existing\_file.txt", "a");

if (file == NULL) {

printf("Error opening file.\n");

return 1;

}

fprintf(file, "%s", data);

fclose(file);

return 0;

}

**6. Write a program to copy the contents of one file to another.**

#include <stdio.h>

int main() {

FILE \*source, \*destination;

char ch;

source = fopen("source.txt", "r");

destination = fopen("destination.txt", "w");

if (source == NULL || destination == NULL) {

printf("Error opening files.\n");

return 1;

}

while ((ch = fgetc(source)) != EOF) {

fputc(ch, destination);

}

fclose(source);

fclose(destination);

return 0;

}

**7. Write a program to count the number of words in a file.**

#include <stdio.h>

#include <stdbool.h>

int main() {

FILE \*file;

char ch;

int wordCount = 0;

bool inWord = false;

file = fopen("word\_count.txt", "r");

if (file == NULL) {

printf("Error opening file.\n");

return 1;

}

while ((ch = fgetc(file)) != EOF) {

if (ch == ' ' || ch == '\n' || ch == '\t') {

inWord = false;

} else {

if (!inWord) {

inWord = true;

wordCount++;

}

}

}

printf("Number of words: %d\n", wordCount);

fclose(file);

return 0;

}

**8. Write a program to search for a specific string in a file.**

#include <stdio.h>

#include <string.h>

int main() {

FILE \*file;

char searchStr[50];

char buffer[100];

printf("Enter the string to search: ");

fgets(searchStr, sizeof(searchStr), stdin);

strtok(searchStr, "\n"); // Remove the newline character

file = fopen("search\_file.txt", "r");

if (file == NULL) {

printf("Error opening file.\n");

return 1;

}

while (fgets(buffer, sizeof(buffer), file) != NULL) {

if (strstr(buffer, searchStr) != NULL) {

printf("String found: %s", buffer);

}

}

fclose(file);

return 0;

}

**9. Write a program to encrypt and decrypt a file.**

#include <stdio.h>

void encryptDecrypt(FILE \*input, FILE \*output, int key) {

char ch;

while ((ch = fgetc(input)) != EOF) {

ch = ch ^ key;

fputc(ch, output);

}

}

int main() {

FILE \*input, \*encrypted, \*decrypted;

int key = 5; // Example key

input = fopen("original.txt", "r");

encrypted = fopen("encrypted.txt", "w");

decrypted = fopen("decrypted.txt", "w");

if (input == NULL || encrypted == NULL || decrypted == NULL) {

printf("Error opening files.\n");

return 1;

}

// Encrypt

encryptDecrypt(input, encrypted, key);

// Decrypt

encryptDecrypt(encrypted, decrypted, key);

fclose(input);

fclose(encrypted);

fclose(decrypted);

return 0;

}