WEEKLY INTERNSHIP ASSIGNMENT 4

Students will create a program that reads and writes structured data to files using fopen(), fprintf(), fscanf(), fgets(), and fputs() while working with different file modes.

Source Code

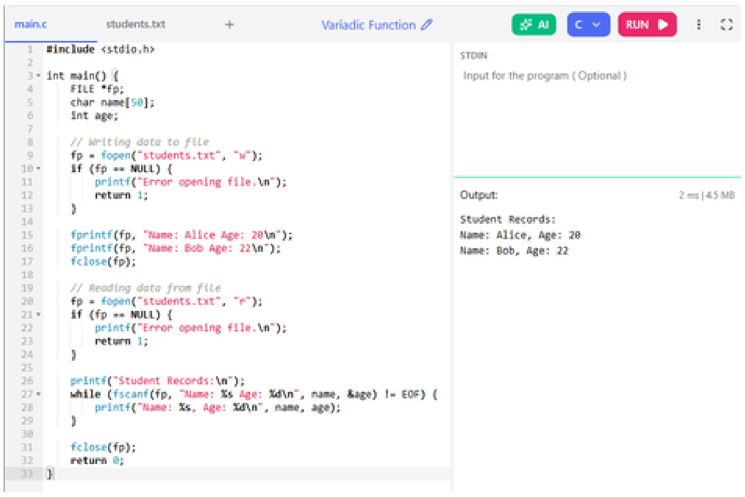
```
#include <stdio.h>

int main() {
    FILE *fp;
    char name[50];
    int age;

    // Writing data to file
    fp = fopen("students.txt", "w");
    if (fp == NULL) {
        printf("Error opening file.\n");
        return 1;
    }

    fprintf(fp, "Name: Alice Age: 20\n");
    fprintf(fp, "Name: Bob Age: 22\n");
    fclose(fp);

    // Reading data from file
    fp = fopen("students.txt", "r");
    if (fp == NULL) {
        printf("Error opening file.\n");
        return 1;
    }

    printf("Student Records:\n");
    while (fscanf(fp, "Name: %s Age: %d\n", name, &age) != EOF) {
        printf("Name: %s, Age: %d\n", name, age);
    }

    fclose(fp);
    return 0;
}
```

students.txt

```
Name: Alice Age: 20
Name: Bob Age: 22
```

Program Output

```
Student Records:
Name: Alice, Age: 20
Name: Bob, Age: 22
```



```c
#include <stdio.h>

int main() {
    FILE *fp;
    char name[50];
    int age;

    // Writing data to file
    fp = fopen("students.txt", "w");
    if (fp == NULL) {
        printf("Error opening file.\n");
        return 1;
    }

    fprintf(fp, "Name: Alice Age: 20\n");
    fprintf(fp, "Name: Bob Age: 22\n");
    fclose(fp);

    // Reading data from file
    fp = fopen("students.txt", "r");
    if (fp == NULL) {
        printf("Error opening file.\n");
        return 1;
    }

    printf("Student Records:\n");
    while (fscanf(fp, "Name: %s Age: %d\n", name, &age) != EOF) {
        printf("Name: %s, Age: %d\n", name, age);
    }

    fclose(fp);
    return 0;
}
```

```
Student Records:
Name: Alice, Age: 20
Name: Bob, Age: 22
```

Explanation:

File handling is a crucial concept in C programming that allows programs to store data permanently and retrieve it later. Unlike variables, which store data temporarily in memory, files enable long-term data storage. This assignment demonstrates basic file handling operations using structured data.

The program uses a **FILE** pointer to interact with a text file named **students.txt**. The **fopen()** function is used to open the file in different modes. When opened in **"w"** mode, the file is created if it does not exist, or its contents are overwritten if it already exists. This mode is suitable for writing fresh data.

The **fprintf()** function is used to write formatted text into the file. In this program, student names and ages are stored in a structured format, making the data readable and easy to parse later. After writing, **fclose()** is called to properly close the file and ensure that data is saved.

To read the data back, the file is reopened in **"r"** mode. This mode allows reading existing file content without modifying it. The program uses **fscanf()** to read formatted data from the file. The while loop continues reading until the end of the file (**EOF**) is reached.

The use of structured formatting ensures that the data can be reliably parsed into variables. This approach is commonly used in basic databases, configuration files, and logging systems.

Functions such as **fgets()** and **fputs()** can also be used for line-based file operations, while **fprintf()** and **fscanf()** are preferred when working with formatted data. Understanding these functions helps programmers choose the appropriate method depending on the application.

File modes like **"a"** (append) are useful when adding new data without deleting existing records, such as maintaining logs or incremental data storage.