**Header Files**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

* #include <stdio.h> → Provides standard input/output functions like printf(), scanf(), fopen(), fwrite(), fread(), etc.
* #include <stdlib.h> → Provides memory allocation, exit(), and other utility functions.
* #include <string.h> → Provides string manipulation functions like strcmp(), strcpy(), etc.

**Defining a Structure for Student Data**

struct Student {

int id;

char name[50];

float marks;

};

* Defines a **structure** Student to store student details:
  + int id → Stores student ID.
  + char name[50] → Stores student name (array of characters).
  + float marks → Stores student marks.

**Function Prototypes**

void addStudent();

void displayStudents();

void searchStudent();

void modifyStudent();

void deleteStudent();

void saveToFile();

void loadFromFile();

* These **function prototypes** declare the functions before they are implemented.
* This helps in organizing the code and ensures that functions can be called before they are defined.

**Global Variables**

struct Student students[100]; // Array to store up to 100 student records

int count = 0; // Keeps track of the number of students

* students[100] → Array of structures to hold student details.
* count → Keeps track of the number of students currently stored.

**Main Function**

int main() {

int choice;

loadFromFile(); // Load data from file at the start

* loadFromFile() loads previously saved student data from a file.

while (1) {

* **Infinite loop** → Keeps the program running until the user exits.

printf("\nStudent Management System\n");

printf("1. Add Student\n");

printf("2. Display Students\n");

printf("3. Search Student\n");

printf("4. Modify Student\n");

printf("5. Delete Student\n");

printf("6. Save & Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

* Displays a **menu** and takes user input for their choice.

switch (choice) {

case 1: addStudent(); break;

case 2: displayStudents(); break;

case 3: searchStudent(); break;

case 4: modifyStudent(); break;

case 5: deleteStudent(); break;

case 6: saveToFile(); exit(0);

default: printf("Invalid choice! Try again.\n");

}

* **Switch case** executes the selected function.
* If the user selects **option 6**, saveToFile() is called, and exit(0) terminates the program.

**Adding a Student**

void addStudent() {

if (count < 100) {

* **Checks if the database is full** before adding a new student.

printf("Enter Student ID: ");

scanf("%d", &students[count].id);

printf("Enter Name: ");

scanf("%s", students[count].name);

printf("Enter Marks: ");

scanf("%f", &students[count].marks);

* Takes **ID, name, and marks** from the user.

count++;

printf("Student Added Successfully!\n");

* Increases count to track the new student and displays a confirmation message.

**Displaying All Students**

void displayStudents() {

if (count == 0) {

printf("No Records Found!\n");

return;

}

* If no students are stored, a message is displayed.

for (int i = 0; i < count; i++) {

printf("ID: %d, Name: %s, Marks: %.2f\n", students[i].id, students[i].name, students[i].marks);

}

* Loops through all students and prints their details.

**Searching for a Student**

void searchStudent() {

int id, found = 0;

printf("Enter Student ID to Search: ");

scanf("%d", &id);

* Takes **ID** input from the user.

for (int i = 0; i < count; i++) {

if (students[i].id == id) {

printf("ID: %d, Name: %s, Marks: %.2f\n", students[i].id, students[i].name, students[i].marks);

found = 1;

break;

}

}

if (!found) printf("Student Not Found!\n");

* Loops through the student list and prints details if ID matches.
* If no match is found, it displays "Student Not Found!".

**Modifying a Student's Details**

void modifyStudent() {

int id, found = 0;

printf("Enter Student ID to Modify: ");

scanf("%d", &id);

* Takes **ID** input from the user.

for (int i = 0; i < count; i++) {

if (students[i].id == id) {

printf("Enter New Name: ");

scanf("%s", students[i].name);

printf("Enter New Marks: ");

scanf("%f", &students[i].marks);

printf("Student Updated Successfully!\n");

found = 1;

break;

}

}

* If the student is found, it allows the user to update **name** and **marks**.

if (!found) printf("Student Not Found!\n");

* Displays an error message if the student ID is not found.

**Deleting a Student**

void deleteStudent() {

int id, found = 0;

printf("Enter Student ID to Delete: ");

scanf("%d", &id);

* Takes **ID** input from the user.

for (int i = 0; i < count; i++) {

if (students[i].id == id) {

for (int j = i; j < count - 1; j++) {

students[j] = students[j + 1];

}

count--;

printf("Student Deleted Successfully!\n");

found = 1;

break;

}

}

* If the student is found, it **shifts all students down** by one index, effectively deleting the record.

if (!found) printf("Student Not Found!\n");

* Displays an error message if the student ID is not found.

**Saving Data to a File**

void saveToFile() {

FILE \*file = fopen("students.dat", "wb");

* Opens students.dat in **write binary (wb) mode**.

if (file) {

fwrite(&count, sizeof(int), 1, file);

fwrite(students, sizeof(struct Student), count, file);

fclose(file);

printf("Data Saved Successfully!\n");

} else {

printf("Error Saving Data!\n");

}

* Writes the count value and students array to the file.
* Closes the file and confirms that the data was saved.

**Loading Data from a File**

void loadFromFile() {

FILE \*file = fopen("students.dat", "rb");

* Opens students.dat in **read binary (rb) mode**.

if (file) {

fread(&count, sizeof(int), 1, file);

fread(students, sizeof(struct Student), count, file);

fclose(file);

}

}

* Reads the count value and student records from the file.
* This ensures that **previously saved data is loaded** when the program starts.

**Conclusion**

This **Student Management System** efficiently handles student data using: ✔ **Structures**  
✔ **File Handling**  
✔ **Loops & Arrays**  
✔ **Decision Control Statements**  
✔ **Functions & Modular Programming**