



STUDENT GRADE TRACKER

Tracking Student Performance Made Easy

AIM OF THE PROGRAM

- This tool aims to efficiently track and manage students' grades
- All we have to do is input student names and their corresponding grades . The tracker will generate comprehensive reports instantly
- The main contents of the program include , to insert a student , to display all students , to find a student by name, to delete a student by name, to update a student information by name, to sort the students by name ,to calculate the class statistics,to display top performers, and to clear all students

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

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

```
#define MAX_STUDENTS 50 //declaring variables globally
```

```
#define MAX_NAME_LENGTH 50
```

```
#define NUM_GRADES 5
```

```
struct Student { //creating a structure
```

```
    char name[MAX_NAME_LENGTH];
```

```
    int grades[NUM_GRADES];
```

```
};
```

```
// Function to insert a new student
```

```
int insertStudent(struct Student students[], int *numStudents) {
```

```
    if (*numStudents >= MAX_STUDENTS) {
```

```
        printf("Student database full. Cannot add more students.\n");
```

```
        return 0; } // Return 0 to indicate failure
```




```
//Entering details of the student
```

```
printf("Enter name for new student: ");
```

```
scanf("%s", students[*numStudents].name);
```

```
printf("Enter %d grades for %s: ", NUM_GRADES, students[*numStudents].name);
```

```
for (int i = 0; i < NUM_GRADES; ++i) {
```

```
    scanf("%d", &students[*numStudents].grades[i]);
```

```
}
```

```
(*numStudents)++; // Incrementing the total number of students by 1
```

```
return 1; // Return 1 to indicate success
```

```
}
```

```
// Function to display all students
```

```
void displayStudents(struct Student students[], int numStudents) {
```

```
    if (numStudents == 0) {
```

```
        printf("No students in the database.\n");
```

```
        return; }
```

```
printf("\nStudent Grade Details:\n");  
  
for (int i = 0; i < numStudents; ++i) {  
    printf("\nStudent: %s\n", students[i].name);  
    printf("Grades: ");  
    for (int j = 0; j < NUM_GRADES; ++j) {  
        printf("%d ", students[i].grades[j]);  
    }  
  
    float sum = 0;  
    for (int j = 0; j < NUM_GRADES; ++j) {  
        sum += students[i].grades[j]; // Calculating total marks of each student  
    }  
    float average = sum / NUM_GRADES;  
    printf("\nAverage Grade: %.2f\n", average);  
}  
}
```

// Function to find a student by name

```
void findStudent(struct Student students[], int numStudents, char *searchName) {
```

```
    int found = 0;
```

```
    for (int i = 0; i < numStudents; ++i) {
```

```
        if (strcmp(students[i].name, searchName) == 0) { //Checking if the name to be searched with the student's name is the same
```

```
            printf("\nStudent found!\n");
```

```
            printf("Name: %s\n", students[i].name); // Displaying the details of the found student
```

```
            printf("Grades: ");
```

```
            for (int j = 0; j < NUM_GRADES; ++j) {
```

```
                printf("%d ", students[i].grades[j]);
```

```
            }
```

```
float sum = 0;
```

```
    for (int j = 0; j < NUM_GRADES; ++j) {
```

```
        sum += students[i].grades[j];
```

```
    }
```

```
float average = sum / NUM_GRADES;
```

```
    printf("\nAverage Grade: %.2f\n", average);
```

```
found = 1;

    break;

}

}

if (!found) {

    printf("Student '%s' not found.\n", searchName);

}

}

// Function to delete a student by name

int deleteStudent(struct Student students[], int *numStudents, char *deleteName) {

    int found = 0;

    for (int i = 0; i < *numStudents; ++i) {

        if (strcmp(students[i].name, deleteName) == 0) {

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

                strcpy(students[j].name, students[j + 1].name); //Copying the details of the deleted student to the previous one

                memcpy(students[j].grades, students[j + 1].grades, sizeof(int) * NUM_GRADES);

            }

        }

    }

}
```

```
(*numStudents)--;//Decrementing the value of total number of students by one
```

```
    found = 1;
```

```
    break;
```

```
}
```

```
}
```

```
return found;
```

```
}
```

```
// Function to update student information by name
```

```
void updateStudent(struct Student students[], int numStudents, char *searchName) {
```

```
    int found = 0;
```

```
    for (int i = 0; i < numStudents; ++i) {
```

```
        if (strcmp(students[i].name, searchName) == 0) {
```

```
            printf("Enter new name for %s: ", searchName);//Updating the details of the student
```

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

```
            printf("Enter %d grades for %s: ", NUM_GRADES, students[i].name);
```

```
            for (int j = 0; j < NUM_GRADES; ++j) {
```

```
                scanf("%d", &students[i].grades[j]);}
```



```
found = 1;

    break;

}

}

if (!found) {

    printf("Student '%s' not found.\n", searchName);

}

// Function to sort students by name

void sortStudentsByName(struct Student students[], int numStudents) {

    struct Student temp;

    for (int i = 0; i < numStudents - 1; ++i) {

        for (int j = 0; j < numStudents - i - 1; ++j) {

            if (strcmp(students[j].name, students[j + 1].name) > 0) {

//Checking if the first string has a greater ASCII value than the second string

                temp = students[j];

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

                students[j + 1] = temp;}}}} //Swapping the two strings
```

```
// Function to calculate class statistics

void classStatistics(struct Student students[], int numStudents) {

    float classAverage = 0;

    float highestAverage = 0;

    float lowestAverage = 100; // Assuming maximum grade is 100

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

        float sum = 0;

        for (int j = 0; j < NUM_GRADES; ++j) {

            sum += students[i].grades[j];    }

        float average = sum / NUM_GRADES; //Calculating average

        classAverage += average;

    if (average > highestAverage) {

        highestAverage = average;

    }

    if (average < lowestAverage) {

        lowestAverage = average;}}}
```

```
if (numStudents > 0) {  
    classAverage /= numStudents;  
    printf("Class Average: %.2f\n", classAverage);  
    printf("Highest Average: %.2f\n", highestAverage);  
    printf("Lowest Average: %.2f\n", lowestAverage);  
} else {  
    printf("No students in the database.\n");  
}}  
  
// Function to display top performers  
void displayTopPerformers(struct Student students[], int numStudents) {  
    if (numStudents == 0) {  
        printf("No students in the database.\n");  
        return;  
    }  
  
    printf("\nTop Performers:\n");  
    printf("Name\t\tAverage Grade\n");  
    printf("----\t\t-----\n");
```

```
float topAverage = 0;

for (int i = 0; i < numStudents; ++i) {
    float sum = 0;
    for (int j = 0; j < NUM_GRADES; ++j) {
        sum += students[i].grades[j];
    }
    float average = sum / NUM_GRADES;
    if (average > topAverage) {
        topAverage = average;}}

for (int i = 0; i < numStudents; ++i) {
    float sum = 0;
    for (int j = 0; j < NUM_GRADES; ++j) {
        sum += students[i].grades[j];}
    float average = sum / NUM_GRADES;

    if (average == topAverage) {//Comparing the average of each student with the highest average of the class
        printf("%s\t\t%.2f\n", students[i].name, average);}}}
```

```
// Function to clear all students
```

```
void clearAllStudents(int *numStudents) {
```

```
    *numStudents = 0;
```

```
    printf("All students have been cleared from the database.\n");
```

```
}
```

```
int main() {
```

```
    struct Student students[MAX_STUDENTS]; //Declaring a variable to access the structure
```

```
    int numStudents = 0;
```

```
    int choice;
```

```
    char searchName[MAX_NAME_LENGTH];
```

```
    char deleteName[MAX_NAME_LENGTH];
```

```
do {
```

```
    printf("\nStudent Grade Tracker Menu:\n");
```

```
    printf("1. Insert a new student\n");
```

```
    printf("2. Display all students\n");
```

```
    printf("3. Find a student by name\n");
```

```
    printf("4. Delete a student by name\n");
```

```
printf("5. Update student information by name\n");  
printf("6. Sort students by name\n");  
printf("7. Calculate class statistics\n");  
printf("8. Display top performers\n");  
printf("9. Clear all students\n");  
printf("10. Exit\n");  
printf("Enter your choice: ");  
scanf("%d", &choice);
```

```
switch (choice) {
```

```
    case 1:
```

```
        insertStudent(students, &numStudents);
```

```
        break;
```

```
    case 2:
```

```
        displayStudents(students, numStudents);
```

```
        break;
```

```
    case 3:
```

```
        printf("Enter name to search: ");
```

```
scanf("%s", searchName);

    findStudent(students, numStudents, searchName);

    break;

case 4:

printf("Enter name to delete: ");

    scanf("%s", deleteName);

    if (deleteStudent(students, &numStudents, deleteName)) {

        printf("Student '%s' deleted successfully.\n", deleteName);

    } else {

        printf("Student '%s' not found.\n", deleteName);

    }

    break;

case 5:

    printf("Enter name to update: ");

    scanf("%s", searchName);

    updateStudent(students, numStudents, searchName);

    break;
```


case 6:

```
    sortStudentsByName(students, numStudents);  
    printf("Students sorted by name.\n");  
    break;
```

case 7:

```
    classStatistics(students, numStudents);  
    break;
```

case 8:

```
    displayTopPerformers(students, numStudents);  
    break;
```

case 9:

```
    clearAllStudents(&numStudents);  
    break;
```

case 10:

```
    printf("Exiting program.\n");  
    break;
```

default:

```
printf("Invalid choice. Please enter a valid option.\n");
```

```
}}
```

```
while (choice != 10);
```

```
return 0;
```

```
}
```

OutPut of the Code

```
main.c
196     }
197     float average = sum / NUM_GRADES;
198     if (average == topAverage) {
199         printf("%s\t\t%.2f\n", students[i].name, average);
200     }
201 }
202 }
203
204 // Function to clear all students
205 void clearAllStudents(int *numStudents) {
206     *numStudents = 0;
207     printf("All students have been cleared from the database.\n");
208 }
209 int main() {
210     struct Student students[MAX_STUDENTS];
211     int numStudents = 0;
212     int choice;
213     char searchName[MAX_NAME_LENGTH];
214     char deleteName[MAX_NAME_LENGTH];
215     printf("Consider the choices");
216
217     do {
218         printf("\nStudent Grade Tracker Menu:\n");
219         printf("1. Insert a new student\n");
220         printf("2. Display all students\n");
221         printf("3. Find a student by name\n");
222         printf("4. Delete a student by name\n");
223         printf("5. Update student information by name\n");
224
225         int choice;
226         printf("Enter your choice: ");
227         scanf("%d", &choice);
228
229         switch (choice) {
230             case 1:
231                 printf("Enter name for new student: ");
232                 scanf("%s", searchName);
233                 printf("Enter 5 grades for %s: ", searchName);
234                 float sum = 0;
235                 for (int i = 0; i < NUM_GRADES; i++) {
236                     float grade;
237                     printf("Grade %d: ", i + 1);
238                     scanf("%f", &grade);
239                     sum += grade;
240                 }
241                 float average = sum / NUM_GRADES;
242                 if (average == topAverage) {
243                     printf("%s\t\t%.2f\n", searchName, average);
244                 }
245             case 2:
246                 // Display all students
247                 for (int i = 0; i < numStudents; i++) {
248                     printf("%s\t\t%.2f\n", students[i].name, students[i].average);
249                 }
250             case 3:
251                 // Find a student by name
252                 for (int i = 0; i < numStudents; i++) {
253                     if (strcmp(students[i].name, searchName) == 0) {
254                         printf("%s\t\t%.2f\n", students[i].name, students[i].average);
255                     }
256                 }
257             case 4:
258                 // Delete a student by name
259                 for (int i = 0; i < numStudents; i++) {
260                     if (strcmp(students[i].name, deleteName) == 0) {
261                         printf("Student %s deleted.\n", deleteName);
262                         numStudents--;
263                         break;
264                     }
265                 }
266             case 5:
267                 // Update student information by name
268                 for (int i = 0; i < numStudents; i++) {
269                     if (strcmp(students[i].name, searchName) == 0) {
270                         printf("Enter new grade for %s: ", searchName);
271                         float grade;
272                         scanf("%f", &grade);
273                         students[i].average = (students[i].average * (i + 1) + grade) / (i + 2);
274                     }
275                 }
276             default:
277                 printf("Invalid choice. Please enter a valid choice.\n");
278         }
279     } while (choice != 10);
280 }
```

10. Exit
Enter your choice: 1
Enter name for new student: abin
Enter 5 grades for abin: 56

```
OnlineGDB beta
main.c
196     }
197     float average = sum / NUM_GRADES;
198     if (average == topAverage) {
199         printf("%s\t\t%.2f\n", students[i].name, average);
200     }
201 }
202 }
203
204 // Function to clear all students
205 void clearAllStudents(int *numStudents) {
206     *numStudents = 0;
207     printf("All students have been cleared from the database.\n");
208 }
209 int main() {
210     struct Student students[MAX_STUDENTS];
211     int numStudents = 0;
212     int choice;
213     char searchName[MAX_NAME_LENGTH];
214     char deleteName[MAX_NAME_LENGTH];
215     printf("Consider the choices");
216
217     do {
218         printf("\nStudent Grade Tracker Menu:\n");
219         printf("1. Insert a new student\n");
220         printf("2. Display all students\n");
221         printf("3. Find a student by name\n");
222         printf("4. Delete a student by name\n");
223         printf("5. Update student information by name\n");
224
225         int choice;
226         printf("Enter your choice: ");
227         scanf("%d", &choice);
228
229         switch (choice) {
230             case 1:
231                 printf("Enter name for new student: ");
232                 scanf("%s", searchName);
233                 printf("Enter 5 grades for %s: ", searchName);
234                 float sum = 0;
235                 for (int i = 0; i < NUM_GRADES; i++) {
236                     float grade;
237                     printf("Grade %d: ", i + 1);
238                     scanf("%f", &grade);
239                     sum += grade;
240                 }
241                 float average = sum / NUM_GRADES;
242                 if (average == topAverage) {
243                     printf("%s\t\t%.2f\n", searchName, average);
244                 }
245             case 2:
246                 // Display all students
247                 for (int i = 0; i < numStudents; i++) {
248                     printf("%s\t\t%.2f\n", students[i].name, students[i].average);
249                 }
250             case 3:
251                 // Find a student by name
252                 for (int i = 0; i < numStudents; i++) {
253                     if (strcmp(students[i].name, searchName) == 0) {
254                         printf("%s\t\t%.2f\n", students[i].name, students[i].average);
255                     }
256                 }
257             case 4:
258                 // Delete a student by name
259                 for (int i = 0; i < numStudents; i++) {
260                     if (strcmp(students[i].name, deleteName) == 0) {
261                         printf("Student %s deleted.\n", deleteName);
262                         numStudents--;
263                         break;
264                     }
265                 }
266             case 5:
267                 // Update student information by name
268                 for (int i = 0; i < numStudents; i++) {
269                     if (strcmp(students[i].name, searchName) == 0) {
270                         printf("Enter new grade for %s: ", searchName);
271                         float grade;
272                         scanf("%f", &grade);
273                         students[i].average = (students[i].average * (i + 1) + grade) / (i + 2);
274                     }
275                 }
276             default:
277                 printf("Invalid choice. Please enter a valid choice.\n");
278         }
279     } while (choice != 10);
280 }
```

7. Calculate class statistics
8. Display top performers
9. Clear all students
10. Exit
Enter your choice: 7
Class Average: 49.40
Highest Average: 49.40
Lowest Average: 49.40

Thank you