Write a C program to process an array of student marks and classify them into grades using switch statements. The program must handle additional requirements such as input validation, dynamic thresholds, and detailed classification.

Detailed Requirements:

Input Constraints:

The program should accept marks for N students, where N is provided by the user at runtime.

The value of N must be between 10 and 200, inclusive. Validate the input and prompt the user until a valid size is entered.

Each student’s marks must be an integer between 0 and 100, inclusive. If any input is invalid, prompt the user to re-enter it.

Grading Scheme:

The grading must be determined dynamically based on thresholds provided by the user for Grade A, B, C, and D.

For example, if the thresholds are:

A: Marks ≥ 90

B: Marks ≥ 75 but < 90

C: Marks ≥ 50 but < 75

D: Marks ≥ 35 but < 50

F: Marks < 35

Ensure the thresholds are non-overlapping and valid. For instance, the Grade B threshold cannot be higher than the Grade A threshold.

Classification Logic:

Use a switch statement to classify marks into grades. Implement the grading logic dynamically based on the thresholds provided by the user.

Use integer ranges in the switch cases (e.g., case 90 ... 100 for Grade A).

Output Requirements:

Display the original array of marks.

For each student, display the student’s mark and their corresponding grade.

Count and display the total number of students in each grade category.

Edge Case Handling:

If all students receive the same grade (e.g., all fail), display a message indicating this.

Handle invalid thresholds gracefully by prompting the user to re-enter them.

Performance Considerations:

Ensure the program handles arrays close to the maximum size of 200 efficiently.

Avoid repetitive checks by grouping marks into ranges before applying the grading logic.

Additional Constraints:

Use functions to handle input validation, grade classification, and result display.

Implement sorting of the marks array before classification to display students’ marks in ascending order.

Example Execution:

Input 1:

Number of Students (N): 6

Marks: [95, 40, 76, 89, 54, 20]

Thresholds:

A: ≥ 90

B: ≥ 75 but < 90

C: ≥ 50 but < 75

D: ≥ 35 but < 50

F: < 35

Output:

Sorted Marks: [20, 40, 54, 76, 89, 95]

Classification:

Mark: 20 → Grade F

Mark: 40 → Grade D

Mark: 54 → Grade C

Mark: 76 → Grade B

Mark: 89 → Grade B

Mark: 95 → Grade A

Grade Counts:

Grade A: 1

Grade B: 2

Grade C: 1

Grade D: 1

Grade F: 1

Input 2:

Marks: [35, 34, 33, 30, 25, 20]

Output:

"All students failed (Grade F)."

#include <stdio.h>

#include <stdlib.h>

int validateStudents() {

int n;

do {

printf("Enter the number of students (range 10 to 200): ");

scanf("%d", &n);

if (n < 10 || n > 200) {

printf("Invalid input. Please enter a number in the range 10 to 200.\n");

}

} while (n < 10 || n > 200);

return n;

}

void validateMarks(int marks[], int n) {

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

do {

printf("Enter marks for student %d (range 0 to 100): ", i + 1);

scanf("%d", &marks[i]);

if (marks[i] < 0 || marks[i] > 100) {

printf("Invalid marks. Please enter a value in the range 0 to 100.\n");

}

} while (marks[i] < 0 || marks[i] > 100);

}

}

void validateThresholds(int \*gradeA, int \*gradeB, int \*gradeC, int \*gradeD) {

do {

printf("Enter the threshold for Grade A (>=90): ");

scanf("%d", gradeA);

printf("Enter the threshold for Grade B (>=75): ");

scanf("%d", gradeB);

printf("Enter the threshold for Grade C (>=50): ");

scanf("%d", gradeC);

printf("Enter the threshold for Grade D (>=35): ");

scanf("%d", gradeD);

if (\*gradeA <= \*gradeB || \*gradeB <= \*gradeC || \*gradeC <= \*gradeD || \*gradeD < 0 || \*gradeA > 100) {

printf("Invalid thresholds. Please re-enter.\n");

}

} while (\*gradeA <= \*gradeB || \*gradeB <= \*gradeC || \*gradeC <= \*gradeD || \*gradeD < 0 || \*gradeA > 100);

}

void marksSorting(int marks[], int n) {

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

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

if (marks[j] > marks[j + 1]) {

int temp = marks[j];

marks[j] = marks[j + 1];

marks[j + 1] = temp;

}

}

}

}

void display(int marks[], int n, int gradeA, int gradeB, int gradeC, int gradeD) {

int count[5] = {0}; // For grades A, B, C, D, F

char grade;

printf("\nSorted Marks: ");

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

printf("%d%s", marks[i], (i == n - 1) ? "\n" : ", ");

}

printf("\nGrade Distribution:\n");

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

switch (marks[i]) {

case 90 ... 100:

grade = 'A';

count[0]++;

break;

case 75 ... 89:

grade = 'B';

count[1]++;

break;

case 50 ... 74:

grade = 'C';

count[2]++;

break;

case 35 ... 49:

grade = 'D';

count[3]++;

break;

default:

grade = 'F';

count[4]++;

break;

}

printf("Marks of student %d: %d ---> Grade %c\n", i + 1, marks[i], grade);

}

printf("\nGrade Counts:\n");

printf("Grade A: %d\n", count[0]);

printf("Grade B: %d\n", count[1]);

printf("Grade C: %d\n", count[2]);

printf("Grade D: %d\n", count[3]);

printf("Grade F: %d\n", count[4]);

if (count[0] == 0 && count[1] == 0 && count[2] == 0 && count[3] == 0) {

printf("\nAll students have failed (Grade F).\n");

}

}

int main() {

int n = validateStudents();

int marks[n];

validateMarks(marks, n);

int gradeA, gradeB, gradeC, gradeD;

validateThresholds(&gradeA, &gradeB, &gradeC, &gradeD);

marksSorting(marks, n);

display(marks, n, gradeA, gradeB, gradeC, gradeD);

return 0;

}

OUTPUT

A screen shot of a computer

Description automatically generated