Everest Engineering College

Sanepa - 2, Lalitpur

Lab Sheet - 9

C Files I/O and Structures

Submitted by

Santa Basnet
Roll No. - 310
2022 Batch

Date: 2022-04-01

1. Problem:

Manipulation of array of structures to represent students results.

Roll No.	Name	Programming	Mathematics	Physics
301	Hari Kunwar	45	60	36
302	Manita Thapa	52	15	65
303	Puskar Shah	78	85	79
304	Usha Karki	48	45	45
305	Bikash Rajat	92	95	88

a. Represents the student and their associated marks using array of structures.

b. Write a function to count the number of students failing in each subjects. Assume that the pass percentage is 45%.

c. Given the following result categories (Division),

   i. Less than 45%: Fail
   ii. Above 45%: Pass
   iii. Above 50%: Second Division
   iv. Above 75%: First Division
   v. Above 90%: Distinction.

   Write a function that takes an input of Roll No. of the student and returns the result category (division) of the student.

d. Display the result sheet of all the student that shows the pass division in the following format. In case of failed students, you should show “N/A” in place of division.

Roll No.	Name	Pass Division
301	Hari Kunwar	??
302	Manita Thapa	??
303	Puskar Shah	??
304	Usha Karki	??
305	Bikash Rajat	??

2. Flowchart: Flowchart for result calculation.

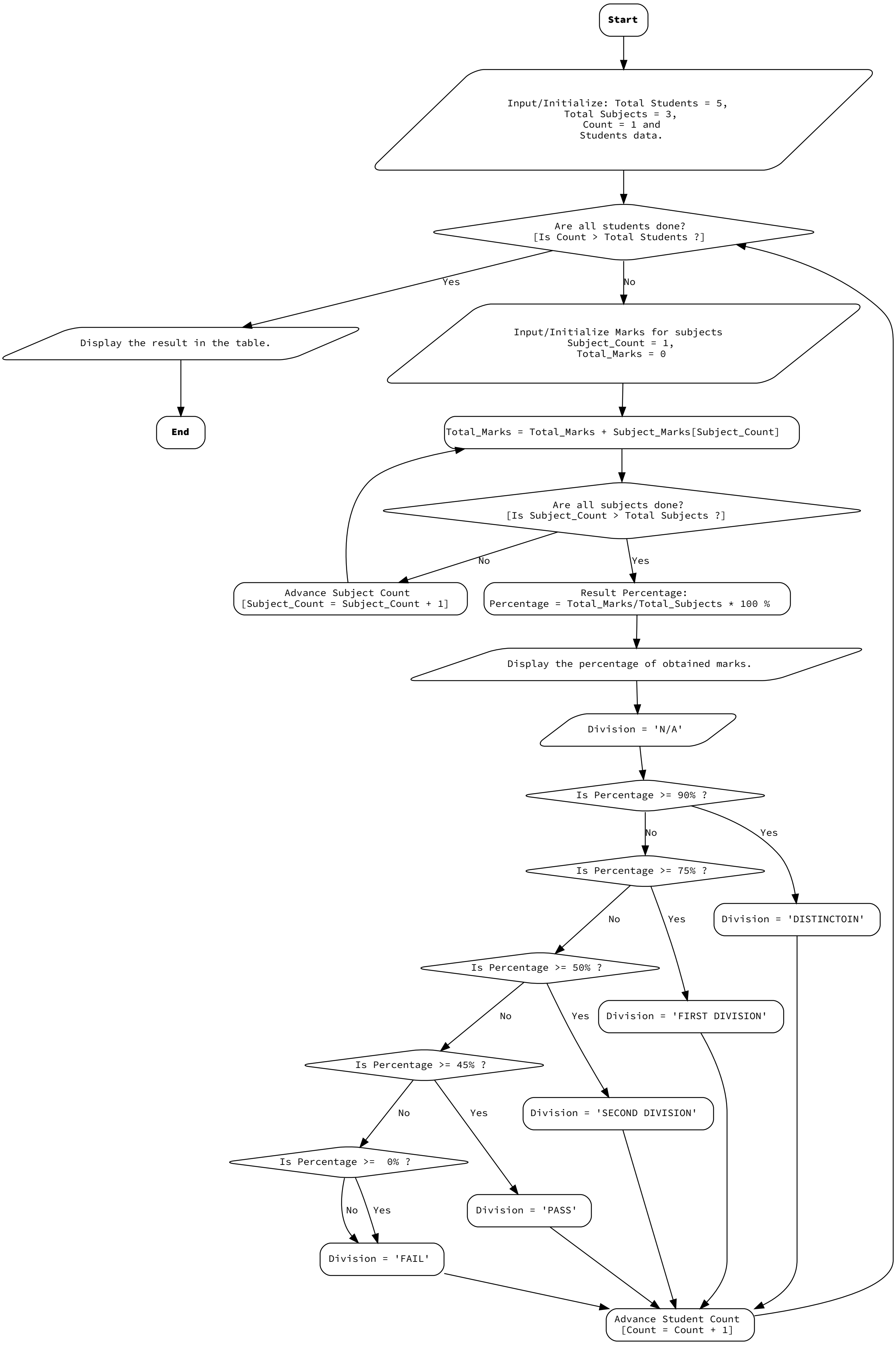

fig.: Flowchart for the calculation of students result.

3. Source Code:

```c
/**
 * Subject: Programming in C.
 * Lab Sheet 9: Files I/O and Structures.
 *
 * Solution given by Santa Basnet.
 * Everest Engineering College, Lalitpur.
 * Date: 01/04/2022
 */

/**
 * Header files.
 */
#include <stdio.h>
#include <string.h>

/**
 * Defines the maximum length of chars allowed for the names
 * and subjects.
 */
#define MAX_CHARS 64

/**
 * Literal for Fail.
 */
#define FAIL "Fail"

/**
 * All the students.
 */
const int NO_OF_STUDENTS    = 5;
const float PASS_PERCENTAGE = 45.0f;
const int NO_OF_SUBJECTS    = 3;
const float FULL_MARKS      = 300.0f;

/**
 * Subject representation.
 */
struct Subject {
    char name[MAX_CHARS];
    int mark;
};

/**
 * Student with subjects representation.
 */
struct Student {
    int crn;
    char name[MAX_CHARS];
    struct Subject subjects[NO_OF_SUBJECTS];
};
```

```
struct Student allStudents[NO_OF_STUDENTS];

/**
 * Perform data initialization of NO_OF_STUDENTS (=5) students.
 */
void initialize() {
    /**
     * Initialize Hari Kunwar, using accessor(dot) operator.
     */
    struct Student hariKunwar;
    hariKunwar.crn = 301;
    strcpy(hariKunwar.name, "Hari Kunwar");
    strcpy(hariKunwar.subjects[0].name, "Programming");
    hariKunwar.subjects[0].mark = 45;
    strcpy(hariKunwar.subjects[1].name, "Mathematics");
    hariKunwar.subjects[1].mark = 60;
    strcpy(hariKunwar.subjects[2].name, "Physics");
    hariKunwar.subjects[2].mark = 36;

    /**
     * Initialize Manita Thapa.
     */
    struct Student manitaThapa = {
            302,
            "Manita Thapa",
            {
                    {"Programming", 52},
                    {"Math", 15},
                    {"Physics", 65}
            }
    };

    /**
    * Initialize Manita Thapa.
    */
    struct Student puskarShah = {
            303,
            "Puskar Shah",
            {
                    {"Programming", 78},
                    {"Math", 85},
                    {"Physics", 79}
            }
    };

    /**
     * Initialize Usha Karki.
     */
    struct Student ushaKarki = {
            304,
            "Usha Karki",
            {
                    {"Programming", 48},
                    {"Math", 45},
                    {"Physics", 45}
            }
    };

    /**
     * Initialize Usha Karki.
     */
    struct Student bikashRajat = {
            305,
            "Bikash Rajat",
            {
                    {"Programming", 92},
                    {"Math", 95},
                    {"Physics", 88}
            }
    };

    /**
     * Initialize all the students.
     */
    allStudents[0] = hariKunwar;
    allStudents[1] = manitaThapa;
    allStudents[2] = puskarShah;
    allStudents[3] = ushaKarki;
    allStudents[4] = bikashRajat;
}

/**
 * Identifies if the given student is passed or not.
 * @param student
 * @return isPassed, 1 means Pass and 0 means failed.
 */
int isPassed(Student student) {
    int isPass = 1;
    for (int index = 0; index < NO_OF_SUBJECTS; index++) {
        int result = student.subjects[index].mark >= PASS_PERCENTAGE ? 1 : 0;
        isPass *= result;
    }
    return isPass;
}

/**
 * Identifies the failed student.
 */
int isFailed(Student student) {
    return !isPassed(student);
}
```

```
/**
 * Displays individual student data in console.
 * @param student
 */
void displayIndividual(Student student) {
    printf("| %4d |%24s | %10d | %10d | %10d |", student.crn, student.name,
    student.subjects[0].mark, student.subjects[1].mark,
    student.subjects[2].mark);
}

/**
 * Displays all the student's data in console.
 */
void display() {
    printf("\n1) All Students: \n");
    for (int index = 0; index < NO_OF_STUDENTS; index++) {
        displayIndividual(allStudents[index]);
        printf("\n");
    }
}

/**
 * Returns the division literals in string format.
 * @param percentage
 * @return divisionLiteral
 */
const char *divisionOf(float percentage) {
    if (percentage < 45.0f) return "Fail";
    else if (percentage < 50.0f) return "Pass";
    else if (percentage < 75.0f) return "Second";
    else if (percentage < 90.0f) return "First";
    else return "Distinction";
}

/**
 * Calculate percentage of a student.
 */
float calculatePercentage(Student student) {
    int total = 0;
    for (int index = 0; index < NO_OF_SUBJECTS; index++)
        total += student.subjects[index].mark;
    float percentage = (float) total / FULL_MARKS * 100.0f;
    return percentage;
}

/**
 * Display individual percentage.
 * @param student
 */
void displayIndividualPercentage(Student student) {
    float percentage = calculatePercentage(student);
    if (isPassed(student))
        printf("| %4d |%24s | %4.2f |", student.crn, student.name, percentage);
    else
        printf("| %4d |%24s | %4.2f |", student.crn, student.name, 0.0f);
}

/**
 * Display individual percentage.
 * @param student
 */
void displayIndividualDivision(Student student) {
    float percentage = calculatePercentage(student);
    if (isPassed(student))
        printf("| %4d |%24s | %12s |", student.crn, student.name,
        divisionOf(percentage));
    else
        printf("| %4d |%24s | %12s |", student.crn, student.name, "N/A");
}

/**
 * Display percentage report of all students.
 */
void displayDivisionReport() {
    printf("\n\n4) Division(Result Category) Report: \n");
    for (int index = 0; index < NO_OF_STUDENTS; index++) {
        displayIndividualDivision(allStudents[index]);
        printf("\n");
    }
}

/**
 * Counts all the failed student.
 * Initially programs assumes, individual student is passed with value 1,
 * for every subject, it should return passed value and multiply it for
 * final result.
 * Individual student count is increased only if, it passes all the subjects.
 *
 * @return failedStudentsCount
 */
int countFailingStudents() {
    int count = 0;
    for (int index = 0; index < NO_OF_STUDENTS; index++)
        if (isFailed(allStudents[index])) count++;
    return count;
}

/**
 * Displays the result category of the student given with CRN.
 * a. Less than 45%: Fail
 * b. Above 45%: Pass
 * c. Above 50%: Second Division
 * d. Above 75%: First Division
 * e. Above 90%: Distinction.
 */
void displayResultCategory(int crn) {
    char result[16];
    strcpy(result, FAIL);

    struct Student student;
    for (int index = 0; index < NO_OF_STUDENTS; index++) {
        if (crn == allStudents[index].crn) {
            student = allStudents[index];
        }
    }
    if (isPassed(student)) {
        float percentage = calculatePercentage(student);
        strcpy(result, divisionOf(percentage));
    }
    printf("\n\n3) Result of %d is \"%s\".", crn, result);
}

/**
 * Main Function.
 * @return 0 for exit.
 */
int main() {
    /**
     * 1. Data representation and initialization.
     */
    initialize();
    display();
    /**
     * 2. Count failing students.
     */
    printf("\n2) Total failed students: %d", countFailingStudents());
    /**
     * 3. Calculated division obtained for a student.
     */
    int crn = 303;
    displayResultCategory(crn);
    /**
     * 4. Display division report.
     */
    displayDivisionReport();
    return 0;
}
```

4. Output:

```
1) All Students:
|  301 |             Hari Kunwar |         45 |          60 |          36 |
|  302 |            Manita Thapa |         52 |          15 |          65 |
|  303 |             Puskar Shah |         78 |          85 |          79 |
|  304 |              Usha Karki |         48 |          45 |          45 |
|  305 |            Bikash Rajat |         92 |          95 |          88 |

2) Total failed students: 2

3) Result of 303 is "First".

4) Division(Result Category) Report:
|  301 |             Hari Kunwar |          N/A |
|  302 |            Manita Thapa |          N/A |
|  303 |             Puskar Shah |        First |
|  304 |              Usha Karki |         Pass |
|  305 |            Bikash Rajat |  Distinction |

Process finished with exit code 0
```

5. Conclusion:

I learned the utilization of user defined data types by representing the students and their results. Especially, the multiple attributes are gathered in a structure with nested implementation for array of subjects. I also got the understanding of multiple casting of basic data types to avoid the errors caused by the integer arithmetic. Finally, two dimension tables are constructed through various escape sequences and formatting provided by the **printf** function. This lab work also helped me to realize the string initialization through string copy library.

6. Appendix:

a. error: ‘struct Student’ has no member named ‘nam’; did you mean ‘name’?
   ```
   printf("| %4d |%24s | %4.2f |", student.crn, student.nam, percentage);
                                                             ^~~
   ```
b. error: expected primary-expression before ‘)’ token
   ```
   strcpy(hariKunwar.subjects[1].name, );
   ```

***