1. Student Report Card using Nested Structures  
Problem Statement:  
Define a nested structure where:

* Student structure contains ID, Name, and Marks (another structure with 3 subjects).
* Write a program to:
  1. Read student details for N students.
  2. Compute their total marks and average.
  3. Display details along with grade classification (A for >=90%, B for 75-89%, etc.).

#include <stdio.h>

typedef struct {

int subject1;

int subject2;

int subject3;

} Marks;

typedef struct {

int id;

char name[50];

Marks marks;

int total;

float average;

char grade;

} Student;

void readStudents(Student students[], int n) {

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

printf("Enter details (ID, Name, Marks in 3 subjects) for student %d: ", i + 1);

scanf("%d %s %d %d %d", &students[i].id, students[i].name,

&students[i].marks.subject1, &students[i].marks.subject2,

&students[i].marks.subject3);

}

}

void calculateGrades(Student students[], int n) {

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

students[i].total = students[i].marks.subject1 + students[i].marks.subject2 + students[i].marks.subject3;

students[i].average = students[i].total / 3.0;

if (students[i].average >= 90) {

students[i].grade = 'A';

} else if (students[i].average >= 75) {

students[i].grade = 'B';

} else if (students[i].average >= 60) {

students[i].grade = 'C';

} else {

students[i].grade = 'D';

}

}

}

void displayStudents(Student students[], int n) {

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

printf("%s - Total: %d, Average: %.2f, Grade: %c\n",

students[i].name, students[i].total, students[i].average, students[i].grade);

}

}

int main() {

int n;

printf("Enter number of students: ");

scanf("%d", &n);

Student students[n];

readStudents(students, n);

calculateGrades(students, n);

displayStudents(students, n);

return 0;

}

Enter number of students: 2

Enter details (ID, Name, Marks in 3 subjects) for student 1: 10

Ryan

56

78

90

Enter details (ID, Name, Marks in 3 subjects) for student 2: 12

Prem

78

60

52

Ryan - Total: 224, Average: 74.67, Grade: C

Prem - Total: 190, Average: 63.33, Grade: C

2. Read & Write Employee Records to a File  
Problem Statement:  
Using file handling, write a program to:

1. Read N employees and store them in a file "employees.txt".
2. Read data back from the file and display it.

#include <stdio.h>

typedef struct {

int id;

char name[50];

float salary;

} Employee;

void writeToFile(Employee emp[], int n) {

FILE \*file = fopen("employees.txt", "w");

if (file == NULL) {

printf("Error opening file for writing.\n");

return;

}

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

fprintf(file, "%d %s %.2f\n", emp[i].id, emp[i].name, emp[i].salary);

}

fclose(file);

}

void readFromFile() {

FILE \*file = fopen("employees.txt", "r");

if (file == NULL) {

printf("Error opening file for reading.\n");

return;

}

printf("Employees Data from File:\n");

Employee emp;

while (fscanf(file, "%d %s %f", &emp.id, emp.name, &emp.salary) != EOF) {

printf("%d %s %.2f\n", emp.id, emp.name, emp.salary);

}

fclose(file);

}

int main() {

int n;

printf("Enter number of employees: ");

scanf("%d", &n);

Employee emp[n];

printf("Enter details (ID, Name, Salary):\n");

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

scanf("%d %s %f", &emp[i].id, emp[i].name, &emp[i].salary);

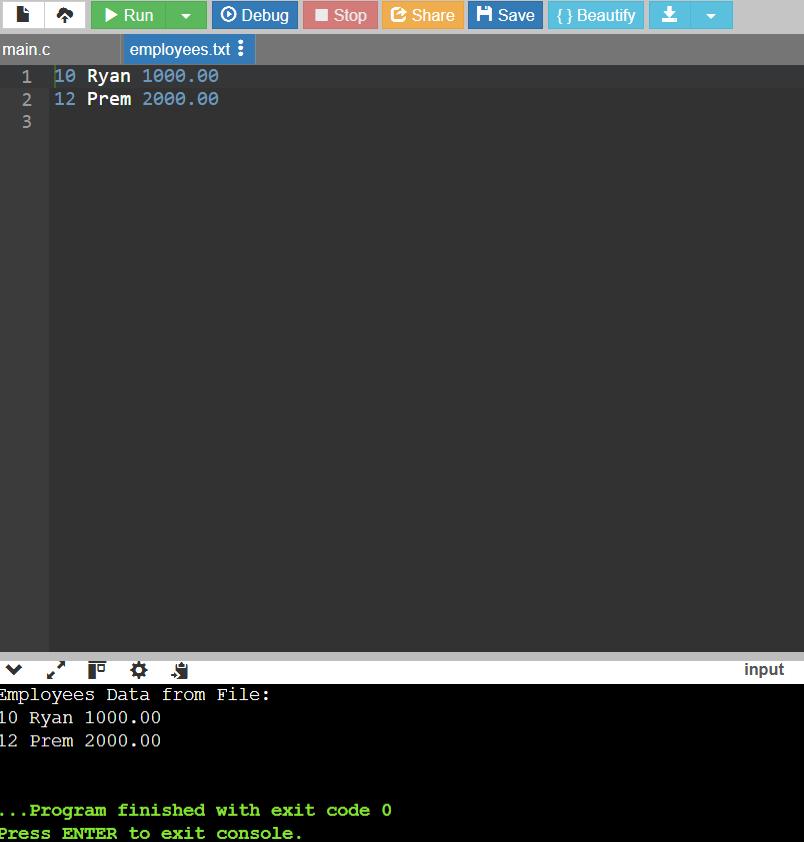
}

writeToFile(emp, n);

readFromFile();

return 0;

}



2. Count Number of Words & Lines in a File  
Problem Statement:  
Write a C program that:

1. Reads a text file.
2. Counts and displays the number of words and lines.

#include <stdio.h>

#include <ctype.h>

void countWordsAndLines(char filename[]) {

FILE \*file = fopen(filename, "r");

if (file == NULL) {

printf("Error opening file.\n");

return;

}

int lines = 0, words = 0;

char ch;

int inWord = 0; // Flag to track if we are inside a word

while ((ch = fgetc(file)) != EOF) {

// Count lines

if (ch == '\n') {

lines++;

}

// Count words

if (isspace(ch)) {

inWord = 0; // We are not in a word anymore

} else {

if (inWord == 0) {

words++; // We found a new word

inWord = 1; // Set the flag to indicate we are in a word

}

}

}

// If the last line does not end with a newline, count it

if (ch == EOF && inWord) {

lines++;

}

fclose(file);

printf("Total Lines: %d\n", lines);

printf("Total Words: %d\n", words);

}

int main() {

char filename[] = "data.txt"; // Specify the input file name

countWordsAndLines(filename);

return 0;

}

4. Append Student Records to a File  
Problem Statement:  
Modify the previous Student Report Card problem to:

1. Store records in a file (students.txt).
2. Allow users to append new student data.

#include <stdio.h>

typedef struct {

int id;

char name[50];

int marks[3];

} Student;

void appendStudentRecord() {

FILE \*file = fopen("students.txt", "a"); // Open file in append mode

if (file == NULL) {

printf("Error opening file for appending.\n");

return;

}

Student student;

printf("Appending student (ID, Name, Marks in 3 subjects):\n");

scanf("%d %s %d %d %d", &student.id, student.name,

&student.marks[0], &student.marks[1], &student.marks[2]);

fprintf(file, "%d %s %d %d %d\n", student.id, student.name,

student.marks[0], student.marks[1], student.marks[2]);

fclose(file);

}

void displayStudentRecords() {

FILE \*file = fopen("students.txt", "r"); // Open file in read mode

if (file == NULL) {

printf("Error opening file for reading.\n");

return;

}

printf("Student Records:\n");

Student student;

while (fscanf(file, "%d %s %d %d %d", &student.id, student.name,

&student.marks[0], &student.marks[1], &student.marks[2]) != EOF) {

printf("%d %s %d %d %d\n", student.id, student.name,

student.marks[0], student.marks[1], student.marks[2]);

}

fclose(file);

}

int main() {

int choice;

do {

printf("1. Append Student Record\n");

printf("2. Display Student Records\n");

printf("3. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

appendStudentRecord();

break;

case 2:

displayStudentRecords();

break;

case 3:

printf("Exiting...\n");

break;

default:

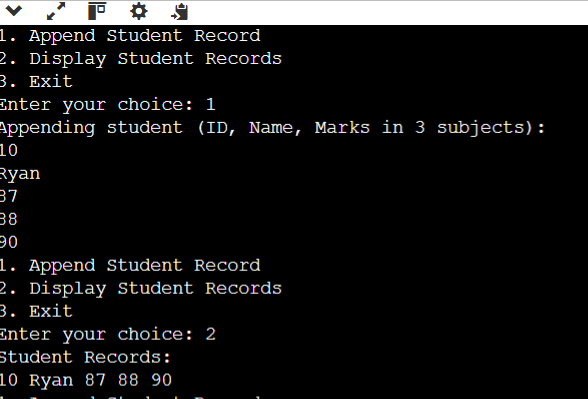
printf("Invalid choice. Please try again.\n");

}

} while (choice != 3);

return 0;

}



`1. Employee Database using Structures`

`Problem Statement:`

Create a structure `Employee` with the following fields:

- `ID` (integer)

- `Name` (string)

- `Salary` (float)

- `Department` (string)

Write a program that:

1. Reads `N employees` and stores them in an array of structures.

2. Displays all employee details.

3. Finds and displays the employee with the `highest salary`.

#include<stdio.h>

struct Employee\_t {

int id;

char name[255];

float salary;

char department[50];

};

typedef struct Employee\_t Employee;

void readEmployees(Employee emp[], int n);

void displayEmployees(Employee emp[], int n);

Employee findHighestSalary(Employee emp[], int n);

int main() {

int employeeCount;

printf("Enter number of employees:");

scanf("%d", &employeeCount);

Employee employees[1000];

readEmployees(employees, employeeCount);

displayEmployees(employees, employeeCount);

Employee highestSalariedEmployee = findHighestSalary(employees, employeeCount);

printf("Employee with highest salary: %s, %.2f (%s)\n",

highestSalariedEmployee.name,

highestSalariedEmployee.salary,

highestSalariedEmployee.department);

return 0;

}

void readEmployees(Employee emp[], int n) {

printf("Enter details (ID, Name, Salary, Department):\n");

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

scanf("%d%s%f%s", &emp[I].id, emp[I].name,

&emp[I].salary, emp[I].department);

}

}

void displayEmployees(Employee emp[], int n) {

printf("Employees:\n");

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

printf("%d - %s - %.2f - %s\n", emp[I].id, emp[I].name,

emp[I].salary, emp[I].department);

}

}

Employee findHighestSalary(Employee emp[], int n) {

Employee highestSalaried = emp[0];

for(int I = 1; I < n; I++) {

if(emp[I].salary > highestSalaried.salary) {

highestSalaried = emp[I];

}

}

return highestSalaried;

}

Enter number of employees:

3

Enter details (ID, Name, Salary, Department):

1 Ryan 40000 MECH

2 Prem 45000 CSE

3 Rohit 37000 ISE

Employees:

1 - Ryan - 40000.00 - MECH

2 - Prem - 45000.00 - CSE

3 - Rohit - 37000.00 - ISE

Employee with highest salary: Prem, 45000.00 (CSE)

`1. Employee Database using Structures`

`Problem Statement:`

Create a structure `Employee` with the following fields:

- `ID` (integer)

- `Name` (string)

- `Salary` (float)

- `Department` (string)

Write a program that:

1. Reads `N employees` and stores them in an array of structures.

1.1 Write N employees into text file "employees.txt" (space separated fields

and write each employee in new line)

1.2 Read N employees from employees.txt file

2. Displays all employee details.

3. Finds and displays the employee with the `highest salary`.

#include<stdio.h>

struct Employee\_t {

int id;

char name[255];

float salary;

char department[50];

};

typedef struct Employee\_t Employee;

void readEmployees(Employee emp[], int n);

void saveEmployees(Employee emp[], int n);

void loadEmployees(Employee emp[], int n);

void displayEmployees(Employee emp[], int n);

Employee findHighestSalary(Employee emp[], int n);

int main() {

int employeeCount;

printf("Enter number of employees:");

scanf("%d", &employeeCount);

Employee employees[1000];

readEmployees(employees, employeeCount);

saveEmployees(employees, employeeCount);

loadEmployees(employees, employeeCount);

displayEmployees(employees, employeeCount);

Employee highestSalariedEmployee = findHighestSalary(employees, employeeCount);

printf("Employee with highest salary: %s, %.2f (%s)\n",

highestSalariedEmployee.name,

highestSalariedEmployee.salary,

highestSalariedEmployee.department);

return 0;

}

void readEmployees(Employee emp[], int n) {

printf("Enter details (ID, Name, Salary, Department):\n");

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

scanf("%d%s%f%s", &emp[I].id, emp[I].name,

&emp[I].salary, emp[I].department);

}

}

void saveEmployees(Employee emp[], int n) {

FILE\* file = fopen("employees.txt", "w");

if(file == NULL) {

printf("Cannot create file.\n");

return;

}

//

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

fprintf(file, "%d %s %.2f %s\n", emp[I].id, emp[I].name,

emp[I].salary, emp[I].department);

}

//

fclose(file);

}

void loadEmployees(Employee emp[], int n) {

FILE\* file = fopen("employees.txt", "r");

if(file == NULL) {

printf("Cannot read file.\n");

return;

}

//

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

fscanf(file, "%d %s %f %s\n", &emp[I].id, emp[I].name,

&emp[I].salary, emp[I].department);

}

//

fclose(file);

}

void displayEmployees(Employee emp[], int n) {

printf("Employees:\n");

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

printf("%d - %s - %.2f - %s\n", emp[I].id, emp[I].name,

emp[I].salary, emp[I].department);

}

}

Employee findHighestSalary(Employee emp[], int n) {

Employee highestSalaried = emp[0];

for(int I = 1; I < n; I++) {

if(emp[I].salary > highestSalaried.salary) {

highestSalaried = emp[I];

}

}

return highestSalaried;

}

`3. Book Inventory Management using Unions`

`Problem Statement:`

Create a `union` to store different types of books:

- `Fiction` (title, author, price)

- `Non-Fiction` (title, subject, price)

Write a program to:

1. Allow users to `enter details of multiple books`.

2. Display details of books based on their category.

#include<stdio.h>

struct Fiction\_t { //383

char title[250];

char author[125];

float price;

};

typedef struct Fiction\_t Fiction;

struct NonFiction\_t { //758

char title[250];

char subject[500];

float price;

};

typedef struct NonFiction\_t NonFiction;

union BookType\_t {

Fiction fiction;

NonFiction nonFiction;

};

typedef union BookType\_t BookType;

struct Book\_t {

int type;

BookType booktype;

};

typedef struct Book\_t Book;

Book books[1000];

int bookCount;

void addBooks();

void displayBooks();

int main() {

addBooks();

displayBooks();

return 0;

}

void addBooks() {

printf("Enter number of books:");

scanf("%d", &bookCount);

for(int I = 0; I < bookCount; I++) {

printf("Type of book(1-Fiction, 2-Non fiction):");

scanf("%d", &books[I].type);

switch(books[I].type) {

case 1 : {

printf("Enter Fiction Book Details(Title, Author, Price):");

scanf("%s%s%f", books[I].booktype.fiction.title,

books[I].booktype.fiction.author,

&books[I].booktype.fiction.price);

} break;

case 2 : {

printf("Enter Non Fiction Book Details(Title, Subject, Price):");

scanf("%s%s%f", books[I].booktype.nonFiction.title,

books[I].booktype.nonFiction.subject,

&books[I].booktype.nonFiction.price);

} break;

}

}

}

void displayBooks() {

for(int I = 0; I < bookCount; I++) {

switch(books[I].type) {

case 1 : {

printf("Fiction Book: %s, Author: %s, Price: $%.2f\n", books[I].booktype.fiction.title,

books[I].booktype.fiction.author,

books[I].booktype.fiction.price);

} break;

case 2 : {

printf("Non-Fiction Book: %s, Subject: %s, Price: $%.2f\n", books[I].booktype.nonFiction.title,

books[I].booktype.nonFiction.subject,

books[I].booktype.nonFiction.price);

} break;

}

}

}

Enter number of books:1

Type of book(1-Fiction, 2-Non fiction):1

Enter Fiction Book Details(Title, Author, Price):HarryPotter JKRowling 45

Fiction Book: HarryPotter, Author: JKRowling, Price: $45.00