DAY 13-DAILY ASSIGNMENTS

ANSU MARIUM SHIBU

22-11-2024

1. Problem 1: Dynamic Student Record Management

Objective: Manage student records using pointers to structures and dynamically allocate memory for student names.

Description:

Define a structure Student with fields:

int roll_no: Roll number

char *name: Pointer to dynamically allocated memory for the student's name

float marks: Marks obtained

Write a program to:

Dynamically allocate memory for n students.

Accept details of each student, dynamically allocating memory for their names.

Display all student details.

Free all allocated memory before exiting.

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

struct student{
    int rollno;
    char *names;
    float marks;
};
int main(){
    int n;

    printf("enter no of students:");
    scanf("%d",&n);
    struct student *students=(struct student*)malloc(n*sizeof(struct student));
    if(students==NULL){
        printf("memory allocation failed:\n");
        return 1;
    }
    for(int i=0;i<n;i++){
        students[i].names=(char*)malloc(100*sizeof(char));

        if(students[i].names==NULL){
            printf("memory allocation failed\n");
            return 1;
        }
        printf("enter details of  studemts %d:\n",i+1);
        printf("names:");
        scanf("%s",students[i].names);
        printf("roll no:");
```

```c
tructassi1.c > main()
int main(){
    for(int i=0;i<n;i++){
        if(students[i].names==NULL){
        }
        printf("enter details of  studemts %d:\n",i+1);
        printf("names:");
        scanf("%s",students[i].names);
        printf("roll no:");
        scanf("%d",&students[i].rollno);
        printf("marks:");
        scanf("%f",&students[i].marks);
    }
    printf("student records:\n");
    for(int i=0;i<n;i++){
        printf("names:%s\n",students[i].names);
        printf("roll no:%d\n",students[i].rollno);
        printf("marks:%.2f\n",students[i].marks);

    }
    for(int i=0;i<n;i++){
        free(students[i].names);
    }
    free(students);
    return 0;
}
```

TERMINAL

```
PS D:\c progrms coding> gcc structassi1.c
PS D:\c progrms coding> ./a
enter no of students:2
enter details of  studemts 1:
names:ansu
roll no:23
marks:90
enter details of  studemts 2:
names:thara
roll no:45
marks:98
student records:
names:ansu
roll no:23
marks:90.00
names:thara
roll no:45
marks:98.00
PS D:\c progrms coding>
```

2.Problem 2: Library System with Dynamic Allocation

Objective: Manage a library system where book details are dynamically stored using pointers inside a structure.

Description:

Define a structure Book with fields:

char *title: Pointer to dynamically allocated memory for the book's title

char *author: Pointer to dynamically allocated memory for the author's name

int *copies: Pointer to the number of available copies (stored dynamically)

Write a program to:

Dynamically allocate memory for n books.

Accept and display book details.

Update the number of copies of a specific book.

Free all allocated memory before exiting.

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

struct Book{
    char *title;
    char *author;
    int *copies;
};

int main(){
    int n;
    printf("enetr the number of books:");
    scanf("%d",&n);

    struct Book *books=(struct Book*)malloc(n*sizeof(struct Book));
    if(books==NULL){
        printf("memory allocation failed:\n");
        return 1;
    }
    for(int i=0;i<n;i++){
        books[i].title=(char*)malloc(100*sizeof(char));
        books[i].author=(char*)malloc(100*sizeof(char));
        books[i].copies=(int *)malloc(sizeof(int));

        printf("Enter details for book %d:\n", i + 1);
        printf("Title: ");
        scanf(" %[^\n]", books[i].title);
        printf("Author: ");
        scanf(" %[^\n]", books[i].author);
```

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

        printf("Enter details for book %d:\n", i + 1);
        printf("Title: ");
        scanf(" %[^\n]", books[i].title);
        printf("Author: ");
        scanf(" %[^\n]", books[i].author);
        printf("Number of copies: ");
        scanf("%d", books[i].copies);

    }
    printf("\nBook records:\n");
    for(int i=0;i<n;i++)
    {
        printf("\nBook %d:\n", i + 1);
        printf("Title: %s\n", books[i].title);
        printf("Author: %s\n", books[i].author);
        printf("Number of copies: %d\n", *(books[i].copies));
    }
    for(int i=0;i<n;i++){
        free(books[i].title);
        free(books[i].author);
        free(books[i].copies);
    }
    free(books);


}
```

```
Title: Author: number of copies: en
PS D:\c progrms coding> gcc structassi2.c
PS D:\c progrms coding> ./a
enetr the number of books:2
Enter details for book 1:
Title: dont
Author: ansu
Number of copies: 4
Enter details for book 2:
Title: fear
Author: thara
Number of copies: 6

Book records:

Book 1:
Title: dont
Author: ansu
Number of copies: 4

Book 2:
Title: fear
Author: thara
Number of copies: 6
PS D:\c progrms coding>
```

3. Problem 1: Complex Number Operations

Objective: Perform addition and multiplication of two complex numbers using structures passed to functions.

Description:

Define a structure Complex with fields:

float real: Real part of the complex number

float imag: Imaginary part of the complex number

Write functions to:

Add two complex numbers and return the result.

Multiply two complex numbers and return the result.

Pass the structures as arguments to these functions and display the results.

```c
#include <stdio.h>

struct complex {
    float real;
    float imag;
};

struct complex addcomplex(struct complex c1, struct complex c2);
struct complex multiplycomplex(struct complex c1, struct complex c2);

int main() {
    struct complex num1, num2, sum, product;

    printf("Enter the real and imaginary parts of the first number: ");
    scanf("%f %f", &num1.real, &num1.imag);

    printf("Enter the real and imaginary parts of the second number: ");
    scanf("%f %f", &num2.real, &num2.imag);

    sum = addcomplex(num1, num2);
    product = multiplycomplex(num1, num2);

    printf("Sum: %.2f + %.2fi\n", sum.real, sum.imag);
    printf("Product: %.2f + %.2fi\n", product.real, product.imag);

    return 0;
}

struct complex addcomplex(struct complex c1, struct complex c2) {
```

```c
    product = multiplycomplex(num1, num2);

    printf("Sum: %.2f + %.2fi\n", sum.real, sum.imag);
    printf("Product: %.2f + %.2fi\n", product.real, product.imag);

    return 0;
}

struct complex addcomplex(struct complex c1, struct complex c2) {
    struct complex result;
    result.real = c1.real + c2.real;
    result.imag = c1.imag + c2.imag;
    return result;
}

struct complex multiplycomplex(struct complex c1, struct complex c2) {
    struct complex result;
    result.real = (c1.real * c2.real) - (c1.imag * c2.imag);
    result.imag = (c1.real * c2.imag) + (c1.imag * c2.real);
    return result;
}
```

```
PS D:\c progrms coding> gcc structfunass1.c
PS D:\c progrms coding> ./a
Enter the real and imaginary parts of the first number:  3 2
Enter the real and imaginary parts of the second number: 1 7
Sum: 4.00 + 9.00i
Product: -11.00 + 23.00i
PS D:\c progrms coding>
```

4. Problem 2: Rectangle Area and Perimeter Calculator

Objective: Calculate the area and perimeter of a rectangle by passing a structure to functions.

Description:

Define a structure Rectangle with fields:

float length: Length of the rectangle

float width: Width of the rectangle

Write functions to:

Calculate and return the area of the rectangle.

Calculate and return the perimeter of the rectangle.

Pass the structure to these functions by value and display the results in main.

```c
#include<stdio.h>

struct rectangle{
    float length;
    float width;
};

float calculatearea(struct rectangle rect);
float calculateperimeter(struct rectangle rect);

int main(){
    struct rectangle rect;

    printf("enter the length of rect:");
    scanf("%f",&rect.length);

    printf("enter the width of triangle:");
    scanf("%f",&rect.width);

    float area=calculatearea(rect);
    float perimeter=calculateperimeter(rect);

    printf("are:%.2f\n",area);
    printf("perimeter:%.2f",perimeter);
```

```c
    struct rectangle rect;

    printf("enter the length of rect:");
    scanf("%f",&rect.length);

    printf("enter the width of triangle:");
    scanf("%f",&rect.width);

    float area=calculatearea(rect);
    float perimeter=calculateperimeter(rect);

    printf("are:%.2f\n",area);
    printf("perimeter:%.2f",perimeter);

}
float calculatearea(struct rectangle rect){
    return rect.length*rect.width;
}
float calculateperimeter(struct rectangle rect){
    return 2*(rect.length+rect.width);
}
```

```
PS D:\c progrms coding> gcc structfunass2.c
PS D:\c progrms coding> ./a
enter the length of rect:5
enter the width of triangle:3
are:15.00
perimeter:16.00
PS D:\c progrms coding>
```

5. Problem 3: Student Grade Calculation

Objective: Calculate and assign grades to students based on their marks by passing a structure to a function.

Description:

Define a structure Student with fields:

char name[50]: Name of the student

int roll_no: Roll number

float marks[5]: Marks in 5 subjects

char grade: Grade assigned to the student

Write a function to:

Calculate the average marks and assign a grade (A, B, etc.) based on predefined criteria.

Pass the structure by reference to the function and modify the grade field.

```c
#include<stdio.h>

struct Student{
    char name[50];
    int roll_no;
    float marks[5];
    char grade;
};

void Grade(struct Student *s);

int main(){
    struct Student student;

    printf("enetr name:\n");
    scanf("%s",student.name);

    printf("enetr roll no:\n");
    scanf("%d",&student.roll_no);

    printf("enter marks for 5 sub:\n");
    for(int i=0;i<5;i++){
        printf("sub %d:",i+1);
        scanf("%f",&student.marks[i]);
    }
    Grade(&student);

    printf("\nStudent Name: %s\n", student.name);
    printf("Roll Number: %d\n", student.roll_no);
    printf("Grades: %c\n", student.grade);

}

void Grade(struct Student *s){
    float total=0;
    for(int i=0;i<5;i++){
        total+=s->marks[i];
    }
    float average=total/5;

    if(average >= 90) {
        s->grade = 'A';
    } else if(average >= 80) {
        s->grade = 'B';
    } else if(average >= 70) {
        s->grade = 'C';
```

```c
void Grade(struct Student *s){
    float total=0;
    for(int i=0;i<5;i++){
        total+=s->marks[i];
    }
    float average=total/5;

    if(average >= 90) {
        s->grade = 'A';
    } else if(average >= 80) {
        s->grade = 'B';
    } else if(average >= 70) {
        s->grade = 'C';
    } else {
        s->grade = 'D';
    }
}
```

```
PS D:\c progrms coding> ./a
enetr name:
ansu
enetr roll no:
23
enter marks for 5 sub:
sub 1:100
sub 2:40
sub 3:90
sub 4:79
sub 5:45

Student Name: ansu
Roll Number: 23
Grades: C
PS D:\c progrms coding>
```

6. Problem 4: Point Operations in 2D Space

Objective: Calculate the distance between two points and check if a point lies within a circle using structures.

Description:

Define a structure Point with fields:

float x: X-coordinate of the point

float y: Y-coordinate of the point

Write functions to:

Calculate the distance between two points.

Check if a given point lies inside a circle of a specified radius (center at origin).

Pass the Point structure to these functions and display the results.

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

struct Point {
    float x;
    float y;
};

float caldistance(struct Point p1, struct Point p2);
int pointincircle(struct Point p, float radius);

int main() {
    struct Point p1, p2;
    float radius;

    printf("Enter coordinates of point 1 (x, y): ");
    scanf("%f %f", &p1.x, &p1.y);

    printf("Enter coordinates of point 2 (x, y): ");
    scanf("%f %f", &p2.x, &p2.y);

    float distance = caldistance(p1, p2);
    printf("The distance between the points is: %.2f\n", distance);

    printf("Enter radius of the circle: ");
```

```c
printf("Enter coordinates of point 1 (x, y): ");
scanf("%f %f", &p1.x, &p1.y);

printf("Enter coordinates of point 2 (x, y): ");
scanf("%f %f", &p2.x, &p2.y);

float distance = caldistance(p1, p2);
printf("The distance between the points is: %.2f\n", distance

printf("Enter radius of the circle: ");
scanf("%f", &radius);

if (pointincircle(p1, radius)) {
    printf("Point 1 is inside the circle.\n");
} else {
    printf("Point 1 is outside the circle.\n");
}

if (pointincircle(p2, radius)) {
    printf("Point 2 is inside the circle.\n");
} else {
    printf("Point 2 is outside the circle.\n");
}
```

```c
    } else {
        printf("Point 1 is outside the circle.\n");
    }

    if (pointincircle(p2, radius)) {
        printf("Point 2 is inside the circle.\n");
    } else {
        printf("Point 2 is outside the circle.\n");
    }

    return 0;
}

float caldistance(struct Point p1, struct Point p2) {
    return sqrt(pow(p2.x - p1.x, 2) + pow(p2.y - p1.y, 2));
}

int pointincircle(struct Point p, float radius) {
    float distance = sqrt(pow(p.x, 2) + pow(p.y, 2));
    if (distance < radius) {
        return 1;
    } else {
```

```c
    return 0;
}

float caldistance(struct Point p1, struct Point p2) {
    return sqrt(pow(p2.x - p1.x, 2) + pow(p2.y - p1.y, 2));
}

int pointincircle(struct Point p, float radius) {
    float distance = sqrt(pow(p.x, 2) + pow(p.y, 2));
    if (distance < radius) {
        return 1;
    } else {
        return 0;
    }
}
```

```
PS D:\c progrms coding> gcc structfunassi4.c
PS D:\c progrms coding> ./a
Enter coordinates of point 1 (x, y): 3 4
Enter coordinates of point 2 (x, y): 7 1
The distance between the points is: 5.00
Enter radius of the circle: 6
Point 1 is inside the circle.
Point 2 is outside the circle.
PS D:\c progrms coding>
```

7. Problem 5: Employee Tax Calculation

Objective: Calculate income tax for an employee based on their salary by passing a structure to a function.

Description:

Define a structure Employee with fields:

char name[50]: Employee name

int emp_id: Employee ID

float salary: Employee salary

float tax: Tax to be calculated (initialized to 0)

Write a function to:

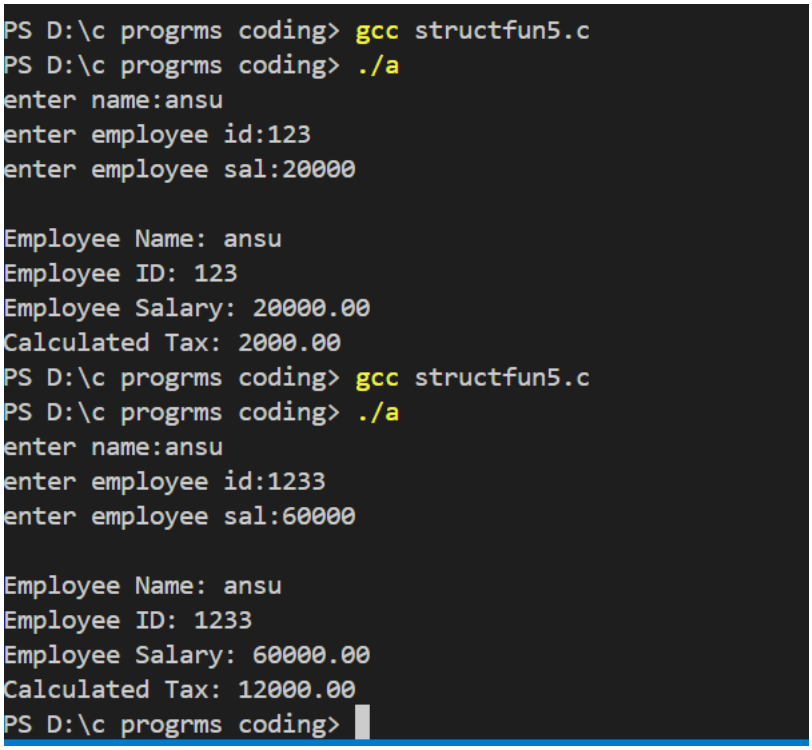
Calculate tax based on salary slabs (e.g., 10% for salaries below $50,000, 20% otherwise).

Modify the tax field of the structure.

Pass the structure by reference to the function and display the updated tax in main.

```c
#include<stdio.h>

struct Employee{
    char name[50];
    int emp_id;
    float salary;
    float tax;

};

void calculatetax(struct Employee *e);

int main(){
    struct Employee emp;

    printf("enter name:");
    scanf("%s",emp.name);

    printf("enter employee id:");
    scanf("%d",&emp.emp_id);

    printf("enter employee sal:");
    scanf("%f",&emp.salary);

    emp.tax=0;

    calculatetax(&emp);

    printf("\nEmployee Name: %s\n", emp.name);
```

```c
    printf("enter name:");
    scanf("%s",emp.name);

    printf("enter employee id:");
    scanf("%d",&emp.emp_id);

    printf("enter employee sal:");
    scanf("%f",&emp.salary);

    emp.tax=0;

    calculatetax(&emp);

    printf("\nEmployee Name: %s\n", emp.name);
    printf("Employee ID: %d\n", emp.emp_id);
    printf("Employee Salary: %.2f\n", emp.salary);
    printf("Calculated Tax: %.2f\n", emp.tax);
}
void calculatetax(struct Employee *e){
    if(e->salary<50000){
        e->tax=e->salary * 0.10;
    }
    else{
        e->tax=e->salary * 0.20;
    }
}
```



8. Problem Statement: Vehicle Service Center Management

Objective: Build a system to manage vehicle servicing records using nested structures.

Description:

Define a structure Vehicle with fields:

char license_plate[15]: Vehicle's license plate number

char owner_name[50]: Owner's name

char vehicle_type[20]: Type of vehicle (e.g., car, bike)

Define a nested structure Service inside Vehicle with fields:

char service_type[30]: Type of service performed

float cost: Cost of the service

char service_date[12]: Date of service

Implement the following features:

Add a vehicle to the service center record.

Update the service history for a vehicle.

Display the service details of a specific vehicle.

Generate and display a summary report of all vehicles serviced, including total revenue.

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

struct Service {
    char service_type[30];
    float cost;
    char service_date[12];
};

struct Vehicle {
    char license_plate[15];
    char owner_name[50];
    char vehicle_type[20];
    struct Service service;
};

void add_vehicle(struct Vehicle *v);
void update_service(struct Vehicle *v);
void display_service(struct Vehicle v);
void generate_report(struct Vehicle vehicles[], int count);

int main() {
    struct Vehicle vehicles[100];
    int vehicle_count = 0;
    int choice;

    while (1) {
        printf("\n1. Add Vehicle\n");
        printf("2. Update Service History\n");
```

```c
void add_vehicle(struct Vehicle *v);
void update_service(struct Vehicle *v);
void display_service(struct Vehicle v);
void generate_report(struct Vehicle vehicles[], int count);

int main() {
    struct Vehicle vehicles[100];
    int vehicle_count = 0;
    int choice;

    while (1) {
        printf("\n1. Add Vehicle\n");
        printf("2. Update Service History\n");
        printf("3. Display Service Details\n");
        printf("4. Generate Report\n");
        printf("5. Exit\n");
        printf("Enter choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                add_vehicle(&vehicles[vehicle_count]);
                vehicle_count++;
                break;
            case 2:
                update_service(&vehicles[vehicle_count - 1]);
```

```c
    while (1) {
        switch (choice) {
                add_vehicle(&vehicles[vehicle_count]);
                vehicle_count++;
                break;
            case 2:
                update_service(&vehicles[vehicle_count - 1]);
                break;
            case 3:
                display_service(vehicles[vehicle_count - 1]);
                break;
            case 4:
                generate_report(vehicles, vehicle_count);
                break;
            case 5:
                return 0;
            default:
                printf("Invalid choice.\n");
        }
    }
}

void add_vehicle(struct Vehicle *v) {
    printf("Enter license plate: ");
    scanf("%s", v->license_plate);
    printf("Enter owner name: ");
    scanf("%s", v->owner_name);
    printf("Enter vehicle type (e.g., car, bike): ");
```

```c
void add_vehicle(struct Vehicle *v) {
    printf("Enter license plate: ");
    scanf("%s", v->license_plate);
    printf("Enter owner name: ");
    scanf("%s", v->owner_name);
    printf("Enter vehicle type (e.g., car, bike): ");
    scanf("%s", v->vehicle_type);
}

void update_service(struct Vehicle *v) {
    printf("Enter service type: ");
    scanf("%s", v->service.service_type);
    printf("Enter service cost: ");
    scanf("%f", &v->service.cost);
    printf("Enter service date (DD/MM/YYYY): ");
    scanf("%s", v->service.service_date);
}

void display_service(struct Vehicle v) {
    printf("Service details for vehicle %s:\n", v.license_plate);
    printf("Owner: %s\n", v.owner_name);
    printf("Vehicle Type: %s\n", v.vehicle_type);
    printf("Service Type: %s\n", v.service.service_type);
    printf("Cost: %.2f\n", v.service.cost);
    printf("Date: %s\n", v.service.service_date);
}
```

```c
void display_service(struct Vehicle v) {
    printf("Service details for vehicle %s:\n", v.license_plate);
    printf("Owner: %s\n", v.owner_name);
    printf("Vehicle Type: %s\n", v.vehicle_type);
    printf("Service Type: %s\n", v.service.service_type);
    printf("Cost: %.2f\n", v.service.cost);
    printf("Date: %s\n", v.service.service_date);
}

void generate_report(struct Vehicle vehicles[], int count) {
    float total_revenue = 0;
    printf("\nReport:\n");
    for (int i = 0; i < count; i++) {
        printf("Vehicle %s - Owner: %s - Vehicle Type: %s - Service: %s - Cost: %.2f - Date: %s\n",
            vehicles[i].license_plate, vehicles[i].owner_name, vehicles[i].vehicle_type,
            vehicles[i].service.service_type, vehicles[i].service.cost,
            vehicles[i].service.service_date);
        total_revenue += vehicles[i].service.cost;
    }
    printf("Total Revenue: %.2f\n", total_revenue);
}
```

```
V
PS D:\c progrms coding> gcc nestedstrassi1.c
PS D:\c progrms coding> ./a

1. Add Vehicle
2. Update Service History
3. Display Service Details
4. Generate Report
5. Exit
Enter choice: 1
Enter license plate: abc123
Enter owner name: ansu
Enter vehicle type (e.g., car, bike): bike

1. Add Vehicle
2. Update Service History
3. Display Service Details
4. Generate Report
5. Exit
Enter choice: 2
Enter service type: brake
Enter service cost: 2000
```

```
Enter vehicle type (e.g., car, bike): bike

1. Add Vehicle
2. Update Service History
3. Display Service Details
4. Generate Report
5. Exit
Enter choice: 2
Enter service type: brake
Enter service cost: 2000
Enter service date (DD/MM/YYYY): 22/11/2024

1. Add Vehicle
2. Update Service History
3. Display Service Details
4. Generate Report
5. Exit
Enter choice: 3
Service details for vehicle abc123:
Owner: ansu
Vehicle Type: bike
Service Type: brake
Cost: 2000.00
Date: 22/11/2024

1. Add Vehicle
2. Update Service History
3. Display Service Details
4. Generate Report
5. Exit
Enter choice:
```

```
Date: 22/11/2024

1. Add Vehicle
2. Update Service History
3. Display Service Details
4. Generate Report
5. Exit
Enter choice: 4

Report:
Vehicle abc123 - Owner: ansu - Vehicle Type: bike - Service: brake - Cost: 2000.00 - Date: 22/11/2024
Total Revenue: 2000.00

1. Add Vehicle
2. Update Service History
3. Display Service Details
4. Generate Report
5. Exit
Enter choice:
```