DAY 15

```
#include<stdio.h>

typedef int my_int;

int main(){
    //alias name my_int has been used for declaring the variable
    my_int a=28;

    printf("a=%d\n",a);


    return 0;
}
```

2)//implementing typedef along with structures

```
#include<stdio.h>

typedef struct date{
  int day;
  int month;
  int year;

}dt;

int main(){

  dt var1={26,11,2024};

  printf("size of var1=%ld\n",sizeof(var1));

  //var1={26,11,2024};

  printf("Todays date=%d-%d-%d",var1.day,var1.month,var1.year);

    return 0;
}
```

3)//using typedef with pointers

```
#include<stdio.h>
typedef int * intptr;
   int main(){
       int a=20;
       intptr ptr1=&a;

       printf("001a=%d\n",*ptr1);
```

```
    *ptr1=30;

    printf("002a=%d\n",*ptr1);

  return 0;
}

4)//using typedef for arrays
#include<stdio.h>
typedef int arr[4];//arr is an alias for an array of 4 intrger elements
  int main(){
    arr t={1,2,3,4};

    for(int i=0;i<4;i++){
        printf("%d\t",t[i]);
    }

  return 0;
}
5)/*Problem Statement:
Write a program that defines a custom data type Complex using typedef to represent a
complex number with real and imaginary parts. Implement functions to:
Add two complex numbers.
Multiply two complex numbers.
Display a complex number in the format "a + bi".
Input Example
Enter first complex number (real and imaginary): 3 4
Enter second complex number (real and imaginary): 1 2
Output Example
Sum: 4 + 6i
Product: -5 + 10i
has context menu*/
#include <stdio.h>

typedef struct {
  float real;
  float imag;
} Complex;

int main() {
  Complex c1, c2, sum, product;


  printf("Enter the first complex number (real and imaginary): ");
  scanf("%f %f", &c1.real, &c1.imag);

  printf("Enter the second complex number (real and imaginary): ");
```

```
    scanf("%f %f", &c2.real, &c2.imag);


    sum.real = c1.real + c2.real;
    sum.imag = c1.imag + c2.imag;


    product.real = c1.real * c2.real - c1.imag * c2.imag;
    product.imag = c1.real * c2.imag + c1.imag * c2.real;


    printf("Sum: %.0f + %.0fi\n", sum.real, sum.imag);
    printf("Product: %.0f + %.0fi\n", product.real, product.imag);

    return 0;
}
```

2)/*Typedef for Structures
Problem Statement:
Define a custom data type Rectangle using typedef to represent a rectangle with width and height as float values. Write functions to:
Compute the area of a rectangle.
Compute the perimeter of a rectangle.
Input Example:
Enter width and height of the rectangle: 5 10
Output Example:
Area: 50.00
Perimeter: 30.00*/

```
#include<stdio.h>
typedef struct rectangle{
    float length;
    float breadth;
}rec;

int main(){
    rec r1;
    float area,perimeter;
    printf("enter length:");
    scanf("%f",&r1.length);

    printf("enter width:");
    scanf("%f",&r1.breadth);
    area=r1.length*r1.breadth;
    perimeter=2*(r1.length+r1.breadth);

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

```
    printf("perimeter:%.2f",perimeter);




    return 0;
}
```

1)//function pointers

```
#include<stdio.h>

void display(int);

int main(){
    //declaration a poinetr to the function display

    void (*fun_ptr)(int)=&display;

    //fun_ptr=&display;//intialisation of pointer with address of display function

    (*fun_ptr)(20);//calling the funusing pointers as well as passing the parameters


    return 0;
}

void display(int a){
    printf("a=%d",a);

}
```

2)//Array of function pointers

```
#include<stdio.h>

void add(int,int);
void sub(int,int);
void mul(int,int);

int main(){

  void(*fun_ptr_arr[])(int,int)={add,sub,mul};
```

```
  int a=10,b=20;

  (*fun_ptr_arr[0])(a,b);
  (*fun_ptr_arr[1])(a,b);
  (*fun_ptr_arr[2])(a,b);


    return 0;
}

void add(int a,int b){
    int sum=a+b;
    printf("sum=%d\n",sum);

}
void sub(int a,int b){
    int sub=a-b;
    printf("sub=%d\n",sub);
}
void mul(int a,int b){
    int mul=a*b;
    printf("mul=%d",mul);
}
```

3)

```
/*Simple Calculator Using Function Pointers
Problem Statement:
Write a C program to implement a simple calculator. Use
function pointers to dynamically call functions for
addition, subtraction, multiplication, and division
based on user input.
Input Example:
Enter two numbers: 10 5
Choose operation (+, -, *, /): *
Output Example:
Result: 50*/

#include<stdio.h>
void add(int,int);
void sub(int,int);
void mul(int,int);
void div(int,int);

int main(){
    int a,b;
    char choice;
```

```
    void(*fun_ptr_arr[])(int,int)={add,sub,mul,div};
    printf("enter two numbers:");
    scanf("%d%d",&a,&b);
    printf("choose operation(+,-,*,/)\n");
    scanf(" %c",&choice);
    getchar();
    switch(choice){
        case '+':
                (*fun_ptr_arr[0])(a,b);
                break;
        case '-':
                (*fun_ptr_arr[1])(a,b);
                break;
        case '*':
                 (*fun_ptr_arr[2])(a,b);
                break;
        case '/':
                 (*fun_ptr_arr[3])(a,b);
                 break;
        default:
                printf("invalid");
                break;

        }


    return 0;
}
 void add(int a,int b){
     int sum=a+b;
     printf("sum:%d\n",sum);
 }
void sub(int a,int b){
    int subtraction=a-b;
     printf("sub:%d\n",subtraction);
}
void mul(int a,int b){
    int multiplication=a*b;
     printf("mul:%d\n",multiplication);
}
void div(int a,int b){
    int divison=a/b;
     printf("div:%d\n",divison);
}


/*Array Operations Using Function Pointers
Problem Statement:
```

Write a C program that applies different operations to an array of integers using function pointers. Implement operations like finding the maximum, minimum, and sum of elements.
Input Example:
Enter size of array: 4
Enter elements: 10 20 30 40
Choose operation (1 for Max, 2 for Min, 3 for Sum): 3
Output Example:
Result: 100*/

```
#include<stdio.h>
void max(int *arr,int n);
void min(int *arr,int n);
void sum(int *arr,int n);
int main(){
    int n;
    void(*fun_ptr_array[])(int *,int)={max,min,sum};

    printf("enter size of the array:");
    scanf("%d",&n);
    int arr[n];
    printf("enter array elements:");
    for(int i=0;i<n;i++){
        scanf("%d",&arr[i]);
    }
    for(int i=0;i<n;i++){
        printf("%d\t",arr[i]);
    }

    (*fun_ptr_array[0])(arr,n);
    (*fun_ptr_array[1])(arr,n);
    (*fun_ptr_array[2])(arr,n);


   return 0;
}
void max(int *arr,int n){
   int maximum=arr[0];

    for(int i=0;i<n;i++){
        if(arr[i]>maximum){
            maximum=arr[i];
        }
    }
    printf("max=%d\n",maximum);
}
void min(int *arr,int n){
```

```
    int minimum=arr[0];
    for(int i=0;i<n;i++){
        if(arr[i]<minimum){
            minimum=arr[i];
        }
    }
    printf("min=%d\n",minimum);
}
void sum(int *arr,int n){
    int sum1=0;
    for(int i=0;i<n;i++){
        sum1=sum1+arr[i];
    }
    printf("array_sum=%d\n",sum1);
}
```

3)/*vent System Using Function Pointers
Problem Statement:
Write a C program to simulate a simple event system.
Define three events: onStart, onProcess, and onEnd.
Use function pointers to call appropriate event handlers
dynamically based on user selection.
Input Example:
Choose event (1 for onStart, 2 for onProcess, 3 for onEnd):
1
Output Example:
Event: onStart
Starting the process...*/

```
#include<stdio.h>
void onStart(void);
void onProcess(void);
void onEnd(void);
int main(){
   int n;
   void(*fun_ptr_arr[])(void)={onStart,onProcess,onEnd};
   printf("Choose event (1 for onStart, 2 for onProcess, 3 for onEnd):");
   scanf("%d",&n);
   switch(n){
      case 1:
           (*fun_ptr_arr[0])();
           break;
      case 2:
          (*fun_ptr_arr[1])();
          break;
      case 3:
          (*fun_ptr_arr[2])();
          break;
      default:
           printf("invalid");
           break;
   }
   return 0;
}
void onStart(){
   printf("Event:Onstart\n");
   printf("starting the process.....\n");
}
void onProcess(){
   printf("Event:OnProcess\n");
   printf("Processing.....\n");
}
void onEnd(){
```

```
    printf("Event:OnEnd\n");
    printf("Ending.....\n");
}


    /*Matrix Operations with Function Pointers
Problem Statement:
Write a C program to perform matrix operations using
function pointers. Implement functions to add, subtract,
and multiply matrices. Pass the function pointer to a wrapper
function to perform the desired operation.
Input Example:
Enter matrix size (rows and columns): 2  2
Enter first matrix:
1  2
3  4
Enter second matrix:
5 6
7 8
Choose operation (1 for Add, 2 for Subtract, 3 for Multiply): 1
Output Example:
Result:
6 8
10 12*/
#include <stdio.h>

// Function declarations
void addMatrices(int a[][10], int b[][10], int result[][10], int rows, int cols);
void subtractMatrices(int a[][10], int b[][10], int result[][10], int rows, int cols);
void multiplyMatrices(int a[][10], int b[][10], int result[][10], int rows, int cols);

int main() {
    int rows, cols, choice;
    int a[10][10], b[10][10], result[10][10];

    // Input matrix dimensions
    printf("Enter matrix size (rows and columns): ");
    scanf("%d %d", &rows, &cols);

    // Input matrices
    printf("Enter first matrix:\n");
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            scanf("%d", &a[i][j]);
        }
    }

    printf("Enter second matrix:\n");
```

```
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            scanf("%d", &b[i][j]);
        }
    }

    // Choose operation
    printf("Choose operation (1 for Add, 2 for Subtract, 3 for Multiply): ");
    scanf("%d", &choice);

    // Perform the selected operation using a function pointer
    void (*operation)(int[][10], int[][10], int[][10], int, int);

    switch (choice) {
        case 1:
            operation = addMatrices;
            break;
        case 2:
            operation = subtractMatrices;
            break;
        case 3:
            operation = multiplyMatrices;
            break;
        default:
            printf("Invalid operation choice.\n");
            return 1;
    }

    // Call the selected function
    operation(a, b, result, rows, cols);


    printf("Result:\n");
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            printf("%d ", result[i][j]);
        }
        printf("\n");
    }

    return 0;
}

// Function to add matrices
void addMatrices(int a[][10], int b[][10], int result[][10], int rows, int cols) {
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            result[i][j] = a[i][j] + b[i][j];
```

```
        }
    }
}

// Function to subtract matrices
void subtractMatrices(int a[][10], int b[][10], int result[][10], int rows, int cols) {
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            result[i][j] = a[i][j] - b[i][j];
        }
    }
}

// Function to multiply matrices
void multiplyMatrices(int a[][10], int b[][10], int result[][10], int rows, int cols) {
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            result[i][j] = 0; // Initialize the result element to 0
            for (int k = 0; k < cols; k++) {
                result[i][j] += a[i][k] * b[k][j]; // Multiply and accumulate the sum
            }
        }
    }
}
```

1)Problem Statement: Vehicle Management System

Write a C program to manage information about various vehicles. The program should demonstrate the following:

Structures: Use structures to store common attributes of a vehicle, such as vehicle type, manufacturer name, and model year.

Unions: Use a union to represent type-specific attributes, such as:

Car: Number of doors and seating capacity.

Bike: Engine capacity and type (e.g., sports, cruiser).

Truck: Load capacity and number of axles.

Typedefs: Define meaningful aliases for complex data types using typedef (e.g., for the structure and union types).

Bitfields: Use bitfields tostore flags for vehicle features like airbags, ABS, and sunroof.

Function Pointers: Use a function pointer to dynamically select a function to display specific information about a vehicle based on its type.

Requirements

Create a structure Vehicle that includes:

A char array for the manufacturer name.

An integer for the model year.

A union VehicleDetails for type-specific attributes.

A bitfield to store vehicle features (e.g., airbags, ABS, sunroof).

A function pointer to display type-specific details.
Write functions to:
Input vehicle data, including type-specific details and features.
Display all the details of a vehicle, including the type-specific attributes.
Set the function pointer based on the vehicle type.
Provide a menu-driven interface to:
Add a vehicle.
Display vehicle details.
Exit the program.

Example Input/Output
Input:
1. Add Vehicle
2. Display Vehicle Details
3. Exit
Enter your choice: 1

Enter vehicle type (1: Car, 2: Bike, 3: Truck): 1
Enter manufacturer name: Toyota
Enter model year: 2021
Enter number of doors: 4
Enter seating capacity: 5
Enter features (Airbags[1/0], ABS[1/0], Sunroof[1/0]): 1 1 0

1. Add Vehicle
2. Display Vehicle Details
3. Exit
Enter your choice: 2

Output:
Manufacturer: Toyota
Model Year: 2021
Type: Car
Number of Doors: 4
Seating Capacity: 5
Features: Airbags: Yes, ABS: Yes, Sunroof: No

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

// Common attributes of a vehicle
typedef struct {
    char manufacturer_name[80];
    int model_year;
} sVehicle;

// Type-specific attributes
```

```
typedef union {
    struct {
        int doors;
        int seating_cc;
    } car;
    struct {
        int engine_cc;
        char type[30];
    } bike;
    struct {
        int load_cc;
        int num_axles;
    } truck;
} uVehicle;

// To store vehicle features
struct vehicle_features {
    int airbags;
    int abs;
    int sunroof;
};

// Function prototypes
void add_vehicle(sVehicle vehicle, uVehicle attributes, struct vehicle_features features);
void display(sVehicle vehicle, uVehicle attributes, struct vehicle_features features);

int main() {
    sVehicle vehicle;
    uVehicle attributes;
    struct vehicle_features f1;

    // Array of function pointers
    void (*fun_arr[])(sVehicle, uVehicle, struct vehicle_features) = {add_vehicle, display};

    int choice;
    while (1) {
        printf("1. Add vehicle\n2. Display Vehicle Details\n3. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                (*fun_arr[0])(vehicle, attributes, f1);
                break;
            case 2:
                (*fun_arr[1])(vehicle, attributes, f1);
                break;
            case 3:
                printf("EXIT\n");
```

```
            return 0;
        default:
            printf("Invalid choice!\n");
    }
  }
}

void add_vehicle(sVehicle vehicle, uVehicle attributes, struct vehicle_features features) {
  int type;
  printf("Enter vehicle type (1: Car, 2: Bike, 3: Truck): ");
  scanf("%d", &type);

  // Input common details
  printf("Enter manufacturer name: ");
  getchar(); // Consume newline from previous input
  fgets(vehicle.manufacturer_name, sizeof(vehicle.manufacturer_name), stdin);


  printf("Enter model year: ");
  scanf("%d", &vehicle.model_year);


  switch (type) {
    case 1: // Car
      printf("Enter number of doors: ");
      scanf("%d", &attributes.car.doors);
      printf("Enter seating capacity: ");
      scanf("%d", &attributes.car.seating_cc);
      break;
    case 2: // Bike
      printf("Enter engine capacity: ");
      scanf("%d", &attributes.bike.engine_cc);
      printf("Enter bike type (e.g., sports, cruiser): ");
      getchar(); // Consume newline
      fgets(attributes.bike.type, sizeof(attributes.bike.type), stdin);
      attributes.bike.type[strcspn(attributes.bike.type, "\n")] = '\0'; // Remove newline
      break;
    case 3: // Truck
      printf("Enter load capacity: ");
      scanf("%d", &attributes.truck.load_cc);
      printf("Enter number of axles: ");
      scanf("%d", &attributes.truck.num_axles);
      break;
    default:
      printf("Invalid vehicle type!\n");
      return;
  }
```

```
    // Input features
    printf("Enter features (Airbags[1/0], ABS[1/0], Sunroof[1/0]): ");
    scanf("%d %d %d", &features.airbags, &features.abs, &features.sunroof);
}

void display(sVehicle vehicle, uVehicle attributes, struct vehicle_features features) {
    // Display common details
    printf("\nVehicle Details:\n");
    printf("Manufacturer: %s\n", vehicle.manufacturer_name);
    printf("Model Year: %d\n", vehicle.model_year);

    // Display type-specific details
    if (attributes.car.doors != 0) {
        printf("Type: Car\n");
        printf("Number of Doors: %d\n", attributes.car.doors);
        printf("Seating Capacity: %d\n", attributes.car.seating_cc);
    } else if (attributes.bike.engine_cc != 0) {
        printf("Type: Bike\n");
        printf("Engine Capacity: %d\n", attributes.bike.engine_cc);
        printf("Bike Type: %s\n", attributes.bike.type);
    } else if (attributes.truck.load_cc != 0) {
        printf("Type: Truck\n");
        printf("Load Capacity: %d\n", attributes.truck.load_cc);
        printf("Number of Axles: %d\n", attributes.truck.num_axles);
    }


    printf("Features: ");
    printf("Airbags: %s, ", features.airbags ? "Yes" : "No");
    printf("ABS: %s, ", features.abs ? "Yes" : "No");
    printf("Sunroof: %s\n", features.sunroof ? "Yes" : "No");
}
```

```
1)//wap to calculate the sum of first n natural numbers
//using recursion

#include<stdio.h>

int sumNatural(int);

int main(){
    int n;
    printf("enter the limit:");
    scanf("%d",&n);
```

```
    printf("\n");

    int sum=sumNatural(n);
    printf("sum=%d",sum);
    return 0;
}

int sumNatural(int n){
    int res=0;

    //base condition
    if(n==0){
        return 0;
    }
    //recursive call
    res=n+sumNatural(n-1);

    return res;

}
```

2)factorial using recursion

```
#include<stdio.h>

int fact(int);
int main(){
    int n;
    printf("enter limit:");
    scanf("%d",&n);
    int factorial=fact(n);
    printf("factorial=%d",factorial);


    return 0;
}
int fact(int n){
    if(n==0){
        return 1;
    }
    int fact1=n*fact(n-1);
    return fact1;

}
```

3)//2. WAP to find the sum of digits of a number using recursion.

```
#include<stdio.h>
int digits(int);
```

```
int main(){
    int n;
    printf("enter a number:");
    scanf("%d",&n);
    int result=digits(n);
    printf("sum of digits:%d",result);



    return 0;
}

int digits(int n){

   if(n==0){
       return 0;
   }
   int res=(n%10)+digits(n/10);
   return res;
}
```

4)//3. With Recursion Findout the maximum number in a given array

```
#include<stdio.h>
int max(int [],int);
int main(){
    int n;
    printf("enter size of the array:");
    scanf("%d",&n);
    int arr[n];
    printf("enter array elements:");
    for(int i=0;i<n;i++){
        scanf("%d",&arr[i]);
    }


    for(int i=0;i<n;i++){
        printf("%d\t",arr[i]);
    }

    int result=max(arr,n);
    printf("max element=%d",result);
    return 0;
}
int max(int arr[],int n){
    if(n==1){
        return arr[0];
    }
```

```
    int res=max(arr,n-1);

    if(arr[n - 1] > res) {
        return arr[n - 1];
    } else {
        return res;
    }
    return res;



}
```

5)//. With recurion calculate the power of a given number
//n*power(n)

```
#include <stdio.h>

int power(int base, int exp);

int main() {
    int base, exp;

    printf("Enter base: ");
    scanf("%d", &base);

    printf("Enter exponent: ");
    scanf("%d", &exp);

    int result = power(base, exp);
    printf("%d^%d = %d\n", base, exp, result);

    return 0;
}

int power(int base, int exp) {
    // Base condition
    if (exp == 0) {
        return 1;
    }
    // Recursive call
    return base * power(base, exp - 1);
}
```

6)string length

```
#include<stdio.h>
int strlength(char arr[]);

int main(){
```

```
    char arr[20];
    printf("enter the string");
    scanf("%[^\n]",arr);
    int result=strlength(arr);
    printf("result =%d",result);

}
int strlength(char arr[]){
    int res=0;
    if(arr[0]=='\0'){
        return 0;
    }
    res=1+strlength(arr+1);

}
```
7)