**Program: 1**

**Write a program to print elements of a 5-element integer array initialized with numbers.**

#include <stdio.h>

int main() {

int numbers[5] = {10, 20, 30, 40, 50}; // Array of integers

printf("Array elements:\n");

for (int i = 0; i < 5; i++) {

printf("%d ", numbers[i]);

}

return 0;

}

**Program: 2**

**Write a program to create a Student structure and print its members.**

#include <stdio.h>

struct Student {

int rollNo;

char name[30];

float marks;

};

int main() {

struct Student s1 = {101, "Ravi", 85.5};

printf("Roll No: %d\n", s1.rollNo);

printf("Name: %s\n", s1.name);

printf("Marks: %.2f\n", s1.marks);

return 0;

}

**Program:3**

**Write a C program that accepts four integers as input from the user and determines the minimum number among them.**

#include <stdio.h>

int main() {

int a, b, c,d;

printf("Enter the numbers: ");

scanf("%d %d %d %d", &a,&b,&c,&d);

// Using else-if ladder to find the minimum

if (a <= b && a <= c && a<=d) {

printf("The minimum number is: %d\n", a);

} else if (b <= a && b <= c && b<=d) {

printf("The minimum number is: %d\n", b);

}

else if (c <= a && c <= b && c<=d) {

printf("The minimum number is: %d\n", c);

}

else {

printf("The minimum number is: %d\n", d);

}

return 0;

}

**Program:4**

**Write a program to print and modify elements of an integer array using a pointer.**

#include <stdio.h>

int main() {

int arr[3] = {1, 2, 3};

int \*ptr = arr; // Pointer to first element of array

printf("Original array values: ");

for(int i = 0; i < 3; i++) {

printf("%d ", \*(ptr + i));

}

// Modify using pointer

\*(ptr + 1) = 20;

printf("\nModified array values: ");

for(int i = 0; i < 3; i++) {

printf("%d ", arr[i]);

}

return 0;

}

**Program:5**

**Write a program to access and print Book structure members using a pointer to structure.**

#include <stdio.h>

struct Book {

char title[50];

int pages;

};

int main() {

struct Book b1 = {"Data Structures", 350};

struct Book \*ptr = &b1;

printf("Title: %s\n", ptr->title);

printf("Pages: %d\n", ptr->pages);

return 0;

}

**Program:6**

**Write a program to print an array of fruit names using an array of pointers.**

#include <stdio.h>

int main() {

char \*fruits[] = {"Apple", "Banana", "Cherry", "Date"};

for (int i = 0; i < 4; i++) {

printf("Fruit %d: %s\n", i+1, fruits[i]);

}

return 0;

}

**Program:7**

**Write a program to create and print an array of two Employee structures.**

#include <stdio.h>

struct Employee {

int id;

char name[20];

float salary;

};

int main() {

struct Employee emp[2] = {

{1, "Alice", 50000},

{2, "Bob", 60000}

};

for (int i = 0; i < 2; i++) {

printf("ID: %d, Name: %s, Salary: %.2f\n", emp[i].id, emp[i].name, emp[i].salary);

}

return 0;

}

**Program:8**

**Write a program to print employee details using an array of pointers to Employee structures.**

#include <stdio.h>

struct Employee {

int id;

char name[20];

};

int main() {

struct Employee e1 = {101, "John"};

struct Employee e2 = {102, "Mary"};

struct Employee \*empPtrs[2] = {&e1, &e2};

for (int i = 0; i < 2; i++) {

printf("ID: %d, Name: %s\n", empPtrs[i]->id, empPtrs[i]->name);

}

return 0;

}

**Program:9**

**Write a program to print details of multiple students using an array of structures and pointer to structure.**

#include <stdio.h>

struct Student {

int id;

char name[30];

float marks;

};

void printStudent(struct Student \*s) {

printf("ID: %d\nName: %s\nMarks: %.2f\n\n", s->id, s->name, s->marks);

}

int main() {

struct Student students[3] = {

{101, "Anu", 85.5},

{102, "Bala", 90.0},

{103, "Chetan", 78.5}

};

// Using pointer to structure to print all students

for (int i = 0; i < 3; i++) {

printStudent(&students[i]);

}

return 0;

}

**Program:10**

**Write a program to print employee directory using array of Pointers to Structures.**

#include <stdio.h>

struct Employee {

int empId;

char name[20];

float salary;

};

int main() {

struct Employee e1 = {201, "John", 50000};

struct Employee e2 = {202, "Sara", 60000};

struct Employee e3 = {203, "Mike", 55000};

// Array of pointers to Employee structures

struct Employee \*empPtrs[3] = {&e1, &e2, &e3};

for (int i = 0; i < 3; i++) {

printf("Employee ID: %d, Name: %s, Salary: %.2f\n",

empPtrs[i]->empId, empPtrs[i]->name, empPtrs[i]->salary);

}

return 0;

}

**Program:11**

**Write a C program to calculate and display the count, sum, and average of array elements using a structure and pointers.**

#include <stdio.h>

struct Summary {

int count;

int sum;

float average;

};

void calculateSummary(int \*arr, int size, struct Summary \*summary) {

summary->count = size;

summary->sum = 0;

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

summary->sum += \*(arr + i);

}

summary->average = (float)summary->sum / size;

}

int main() {

int numbers[5] = {10, 20, 30, 40, 50};

int \*ptr = numbers;

// Modify second element to 25 using pointer

\*(ptr + 1) = 25;

struct Summary result;

calculateSummary(numbers, 5, &result);

printf("Count: %d\nSum: %d\nAverage: %.2f\n", result.count, result.sum, result.average);

return 0;

}

**Program:12**

**Write a C program to dynamically allocate memory for an integer array, input values, display them, and free the memory.**

#include <stdio.h>

#include <stdlib.h>

int main() {

int n;

printf("Enter number of elements: ");

scanf("%d", &n);

int \*arr = (int \*)malloc(n \* sizeof(int)); // dynamic allocation

if (arr == NULL) {

printf("Memory allocation failed!\n");

return 1;

}

printf("Enter %d elements:\n", n);

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

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

}

printf("You entered: ");

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

printf("%d ", arr[i]);

}

free(arr); // free allocated memory

return 0;

}

**Program:13**

**Write a C program to dynamically allocate memory for an array of student structures, input and display their details, then free the memory.**

#include <stdio.h>

#include <stdlib.h>

struct Student {

int id;

char name[30];

float marks;

}

int main() {

int n;

printf("Enter number 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++) {

printf("Enter details for student %d\n", i + 1);

printf("ID: ");

scanf("%d", &students[i].id);

printf("Name: ");

scanf("%s", students[i].name);

printf("Marks: ");

scanf("%f", &students[i].marks);

}

printf("\nStudent Details:\n");

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

printf("ID: %d, Name: %s, Marks: %.2f\n", students[i].id, students[i].name, students[i].marks);

}

free(students); // free the allocated memory

return 0;

}

**Program:14**

**Write a C program to create an array of pointers to employee structures, input and display their details using dynamic memory allocation.**

#include <stdio.h>

#include <stdlib.h>

struct Employee {

int id;

char name[20];

};

int main() {

int n;

printf("Enter number of employees: ");

scanf("%d", &n);

// Allocate array of pointers to Employee structures

struct Employee \*\*empPtrs = (struct Employee \*\*)malloc(n \* sizeof(struct Employee \*));

if (empPtrs == NULL) {

printf("Memory allocation failed!\n");

return 1;

}

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

empPtrs[i] = (struct Employee \*)malloc(sizeof(struct Employee));

if (empPtrs[i] == NULL) {

printf("Memory allocation failed for employee %d\n", i + 1);

// Free previously allocated memory before exiting

for (int j = 0; j < i; j++)

free(empPtrs[j]);

free(empPtrs);

return 1;

}

printf("Enter details of employee %d\n", i + 1);

printf("ID: ");

scanf("%d", &empPtrs[i]->id);

printf("Name: ");

scanf("%s", empPtrs[i]->name);

}

printf("\nEmployee Details:\n");

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

printf("ID: %d, Name: %s\n", empPtrs[i]->id, empPtrs[i]->name);

}

// Free all allocated memory

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

free(empPtrs[i]);

free(empPtrs);

return 0;

}