1. Define a structure for student record and print details.

|  |  |
| --- | --- |
| **Input** | - Student ID (integer)  - Student Name (string)  - Student Age (integer)  - Student Marks (float) |

|  |  |
| --- | --- |
| **Process** | - Store the entered data in a structure  - Organize the data for display |

|  |  |
| --- | --- |
| **Output** | - Display the student details: ID, Name, Age, and Marks |

#include <stdio.h>

struct Student

{

int id;

char name[50];

int age;

float marks;

};

int main()

{

struct Student s;

printf("Enter ID: ");

scanf("%d", &s.id);

printf("Enter Name: ");

scanf("%s", s.name);

printf("Enter Age: ");

scanf("%d", &s.age);

printf("Enter Marks: ");

scanf("%f", &s.marks);

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

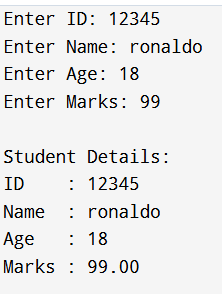
printf("ID : %d\n", s.id);

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

printf("Age : %d\n", s.age);

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

}



1. Write a program to store and display employee details using structures.

|  |  |
| --- | --- |
| **Input** | - Employee ID (int)  - Employee Name (string)  - Employee Salary (float) |
| **Process** | - Store the entered data in an Employee structure |
| **Output** | - Display the stored employee details |

#include <stdio.h>

struct Employee

{

int id;

char name[50];

float salary;

};

int main()

{

struct Employee emp;

printf("Enter Employee ID: ");

scanf("%d", &emp.id);

printf("Enter Employee Name: ");

scanf("%s", emp.name);

printf("Enter Employee Salary: ");

scanf("%f", &emp.salary);

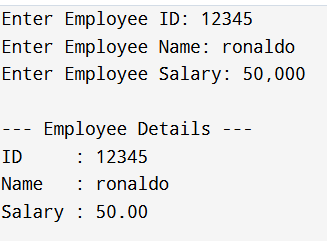
printf("\n--- Employee Details ---\n");

printf("ID : %d\n", emp.id);

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

printf("Salary : %.2f\n", emp.salary);

}



1. Write a program to pass a structure to a function.

|  |  |
| --- | --- |
| **Input** | - Student ID (int)  - Student Name (string) |
| **Process** | - Store input in structure  - Pass structure to a function |
| **Output** | - Display student ID and name using the function |

#include <stdio.h>

struct Student {

int id;

char name[50];

};

void display(struct Student s) {

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

printf("ID : %d\n", s.id);

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

}

int main() {

struct Student s;

printf("Enter ID: ");

scanf("%d", &s.id);

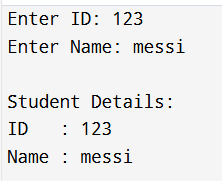
printf("Enter Name: ");

scanf("%s", s.name); // simple name input

display(s);

return 0;

}



1. Write a program to store multiple student records using array of structures.

|  |  |
| --- | --- |
| **Input** | - Number of students n  - For each student: ID, Name, Marks |
| **Process** | - Use a loop to store input in an array of structures |
| **Output** | - Display all student details using a loop |

#include <stdio.h>

struct Student {

int id;

char name[50];

float marks;

};

int main() {

struct Student students[100];

int n, i;

printf("Enter number of students: ");

scanf("%d", &n);

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

printf("\nEnter 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("\n--- Student Records ---\n");

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

printf("\nStudent %d:\n", i + 1);

printf("ID : %d\n", students[i].id);

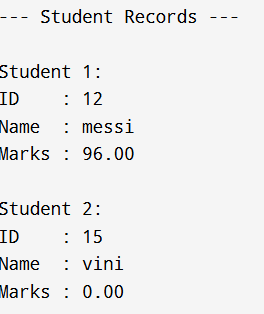
printf("Name : %s\n", students[i].name);

printf("Marks : %.2f\n", students[i].marks);

}

return 0;

}



1. Write a program to demonstrate nested structures.

|  |  |
| --- | --- |
| **Input** | - Student ID, Name  - City, Pincode |
| **Process** | - Store student and address data using nested structure |
| **Output** | - Display student details including address |

#include <stdio.h>

struct Address {

char city[50];

int pincode;

};

struct Student {

int id;

char name[50];

struct Address address;

};

int main() {

struct Student s;

printf("Enter Student ID: ");

scanf("%d", &s.id);

printf("Enter Student Name: ");

scanf("%s", s.name);

printf("Enter City: ");

scanf("%s", s.address.city);

printf("Enter Pincode: ");

scanf("%d", &s.address.pincode);

printf("\n--- Student Details ---\n");

printf("ID : %d\n", s.id);

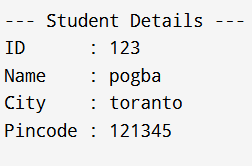
printf("Name : %s\n", s.name);

printf("City : %s\n", s.address.city);

printf("Pincode : %d\n", s.address.pincode);

return 0;

}



1. Write a program to calculate total and average marks using structures.

|  |  |
| --- | --- |
| **Input** | - Student name  - Marks for 3 subjects |
| **Process** | - Calculate total = mark1 + mark2 + mark3  - Calculate average = total / 3 |
| **Output** | - Display total and average marks |

#include <stdio.h>

struct Student {

char name[50];

float mark1, mark2, mark3;

float total;

float average;

};

int main() {

struct Student s;

printf("Enter student name: ");

scanf("%s", s.name);

printf("Enter marks for 3 subjects:\n");

printf("Mark 1: ");

scanf("%f", &s.mark1);

printf("Mark 2: ");

scanf("%f", &s.mark2);

printf("Mark 3: ");

scanf("%f", &s.mark3);

s.total = s.mark1 + s.mark2 + s.mark3;

s.average = s.total / 3;

printf("\n--- Student Marks Report ---\n");

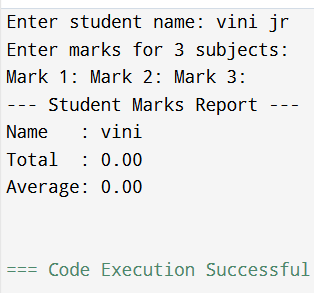
printf("Name : %s\n", s.name);

printf("Total : %.2f\n", s.total);

printf("Average: %.2f\n", s.average);

return 0;

}



1. Write a program to find the highest marks among students.

|  |  |
| --- | --- |
| **Input** | - Number of students  - Each student's name and marks |

|  |  |
| --- | --- |
| **Process** | - Compare all students' marks to find the highest |

|  |  |
| --- | --- |
| **Output** | - Display the student with the highest marks |

#include <stdio.h>

struct Student {

char name[50];

float marks;

};

int main() {

struct Student students[100];

int n, i;

float highest;

int topIndex = 0;

printf("Enter number of students: ");

scanf("%d", &n);

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

printf("\nEnter details for Student %d\n", i + 1);

printf("Name: ");

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

printf("Marks: ");

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

}

highest = students[0].marks;

for (i = 1; i < n; i++) {

if (students[i].marks > highest) {

highest = students[i].marks;

topIndex = i;

}

}

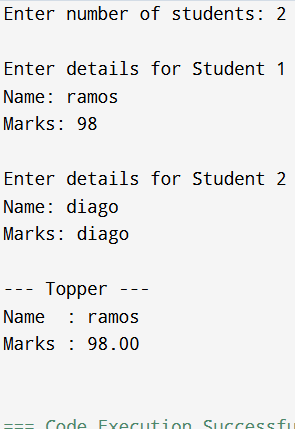
printf("\n--- Topper ---\n");

printf("Name : %s\n", students[topIndex].name);

printf("Marks : %.2f\n", students[topIndex].marks);

return 0;

}



1. Write a program to sort student records by name using structure.

|  |  |
| --- | --- |
| **Input** | - Number of students  - For each student: ID, Name, Marks |
| **Process** | - Sort the array of structures by name using strcmp() and swapping |
| **Output** | - Display student records sorted alphabetically by name |

#include <stdio.h>

#include <string.h>

struct Student {

char name[50];

int id;

float marks;

};

int main() {

struct Student students[100], temp;

int n, i, j;

printf("Enter number of students: ");

scanf("%d", &n);

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

printf("\nEnter 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);

}

for (i = 0; i < n - 1; i++) {

for (j = i + 1; j < n; j++) {

if (strcmp(students[i].name, students[j].name) > 0) {

temp = students[i];

students[i] = students[j];

students[j] = temp;

}

}

}

printf("\n--- Sorted Student Records by Name ---\n");

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

printf("\nStudent %d:\n", i + 1);

printf("ID : %d\n", students[i].id);

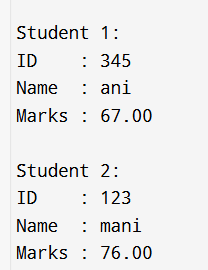
printf("Name : %s\n", students[i].name);

printf("Marks : %.2f\n", students[i].marks);

}

return 0;

}



1. Write a program using union to store data of different types.

|  |  |
| --- | --- |
| **Input** | - Integer value: 25  - Float value: 3.14  - String: "Hello World" |
| **Process** | - Assign values one by one to the union members |
| **Output** | - Prints each assigned value (only last remains valid due to memory sharing) |

#include <stdio.h>

#include <string.h>

union Data {

int i;

float f;

char str[50];

};

int main() {

union Data data;

data.i = 25;

printf("Data as Integer: %d\n", data.i);

data.f = 3.14;

printf("Data as Float : %.2f\n", data.f);

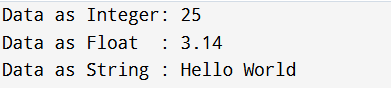
strcpy(data.str, "Hello World");

printf("Data as String : %s\n", data.str);

printf("\nNote: In union, only the last stored value is valid.\n");

return 0;

}



1. Compare and contrast structure vs union with a sample program.

|  |  |  |
| --- | --- | --- |
| **Input** | - Integer, float, and string values assigned to each structure member. |
| **Process** | - All values are stored in separate memory locations.  - Values remain intact. |
| **Output** | - Display all structure values correctly.  - Show structure size. |

#include <stdio.h>

#include <string.h>

struct MyStruct {

int i;

float f;

char str[20];

};

union MyUnion {

int i;

float f;

char str[20];

};

int main() {

struct MyStruct s;

union MyUnion u;

s.i = 10;

s.f = 3.14;

strcpy(s.str, "Structure");

u.i = 10;

u.f = 3.14;

strcpy(u.str, "Union");

printf("Structure Values:\n");

printf(" Integer: %d\n", s.i);

printf(" Float : %.2f\n", s.f);

printf(" String : %s\n", s.str);

// Display union values

printf("\nUnion Values:\n");

printf(" Integer: %d\n", u.i);

printf(" Float : %.2f\n", u.f);

printf(" String : %s\n", u.str);

printf("\nSize of Structure: %lu bytes\n", sizeof(s));

printf("Size of Union : %lu bytes\n", sizeof(u));

return 0;

}