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

IPO

Input: name, age

Process: Store values in a struct Student and print the fields

Output: Student ID, Name, and Age printed to the screen

Code:

#include <stdio.h>

struct Student {

    int id;

    char name[50];

    int age;

};

int main() {

    struct Student s = {1, "Alice", 20};

    printf("Student ID: %d\n", s.id);

    printf("Name: %s\n", s.name);

    printf("Age: %d\n", s.age);

    return 0;

}

Output:

A screenshot of a computer

Description automatically generated

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

IPO  
Input: Employee ID, Name, and Salary

Process: Store the input in a structure

Output: Print the employee’s ID, Name, and Salary

Code:

#include <stdio.h>

struct Employee {

int id;

char name[50];

float salary;

};

int main() {

struct Employee e;

printf("Enter Employee ID: ");

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

printf("Enter Name: ");

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

printf("Enter Salary: ");

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

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

printf("ID: %d\nName: %s\nSalary: %.2f\n", e.id, e.name, e.salary);

return 0;

}

Ouput:

A screenshot of a computer

Description automatically generated

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

IPO:  
Input: id, name

Process: Pass structure to a function; function prints its values

Output: ID and Name printed by the function

Code:

#include <stdio.h>

struct Student {

int id;

char name[50];

};

void display(struct Student s) {

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

}

int main() {

struct Student s = {1, "John"};

display(s);

return 0;

}

Output:

A screenshot of a computer

Description automatically generated

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

IPO:

Input: User enters ID and Name for 3 students

Process: Store each student’s data in an array of structures

Output: Display all student IDs and Names

Code:

#include <stdio.h>

struct Student {

    int id;

    char name[50];

};

int main() {

    int i;

    struct Student s[3];

    for(i = 0; i < 3; i++) {

        printf("Enter ID and Name for student %d: ", i + 1);

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

    }

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

    for(i = 0; i < 3; i++) {

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

    }

    return 0;

}

Output:

A black background with white text

Description automatically generated

1. Write a program to demonstrate nested structures.

IPO:

Input: id , name , dob

Process: Store data in nested structures (struct Student containing struct Date)

Output: Display ID, Name, and Date of Birth in DD-MM-YYYY format

Code:

#include <stdio.h>

struct Date {

int day, month, year;

};

struct Student {

int id;

char name[50];

struct Date dob;

};

int main() {

struct Student s = {1, "Amit", {15, 8, 2000}};

printf("ID: %d\nName: %s\nDOB: %02d-%02d-%04d\n", s.id, s.name, s.dob.day, s.dob.month, s.dob.year);

return 0;

}

Output:

A black screen with white text

AI-generated content may be incorrect.

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

IPO:

Input: User enters Name and 3 subject Marks

Process: Calculate total and average marks

Output: Display total marks and average

Code:

#include <stdio.h>

struct Student {

char name[50];

int marks[3];

};

int main() {

struct Student s;

int total = 0;

printf("Enter student name: ");

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

printf("Enter marks in 3 subjects: ");

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

scanf("%d", &s.marks[i]);

total += s.marks[i];

}

float average = total / 3.0;

printf("Total: %d\nAverage: %.2f\n", total, average);

return 0;

}

Output:

A black screen with white text

Description automatically generated

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

IPO

Input: User enters Name and Marks for 3 students

Process: Find the student with the highest marks

Output: Display the name and marks of the topper

Code:

#include <stdio.h>

struct Student {

char name[50];

int marks[3];

};

int main() {

struct Student s;

int total = 0;

printf("Enter student name: ");

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

printf("Enter marks in 3 subjects: ");

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

scanf("%d", &s.marks[i]);

total += s.marks[i];

}

float average = total / 3.0;

printf("Total: %d\nAverage: %.2f\n", total, average);

return 0;

}

Output:

A black screen with white text

Description automatically generated

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

IPO

Input: User enters ID and Name for 3 students

Process: Sort the array of structures alphabetically by Name using bubble sort

Output: Display sorted student records by Name

Code:

#include <stdio.h>

#include <string.h>

struct Student {

char name[50];

int id;

};

int main() {

struct Student s[3], temp;

int i, j;

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

printf("Enter ID and Name of student %d: ", i + 1);

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

}

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

for(j = i + 1; j < 3; j++) {

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

temp = s[i];

s[i] = s[j];

s[j] = temp;

}

}

}

printf("\nSorted by Name:\n");

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

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

}

Output:

A screen shot of a computer

AI-generated content may be incorrect.

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

IPO

Input: i, f ,str

Process: Assign different data types to union and print each

Output: Print each union member value

Code:

#include <stdio.h>

#include <string.h>

union Data {

int i;

float f;

char str[20];

};

int main() {

union Data data;

data.i = 10;

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

data.f = 220.5;

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

strcpy(data.str, "Hello");

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

// Note: previous values are lost due to union

return 0;

}

Output:

A black background with white text

Description automatically generated

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

IPO

**Input:**

* Structure: id, marks, name
* Union: id, marks, name

**Process:** Display how data is stored and overwritten in structure vs union

**Output:**

* All structure fields are retained and printed
* In union, only the last assigned value is printed

Code:

#include <stdio.h>

#include <string.h>

struct StudentStruct {

int id;

float marks;

char name[20];

};

union StudentUnion {

int id;

float marks;

char name[20];

};

int main() {

struct StudentStruct s1 = {1, 88.5, "Asha"};

union StudentUnion u1;

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

printf("ID: %d\nMarks: %.2f\nName: %s\n", s1.id, s1.marks, s1.name);

printf("\n--- Union (Only last assigned is valid) ---\n");

u1.id = 2;

printf("ID: %d\n", u1.id);

u1.marks = 90.2;

printf("Marks: %.2f\n", u1.marks);

strcpy(u1.name, "Raj");

printf("Name: %s\n", u1.name);

return 0;

}

Output:

A screen shot of a computer

Description automatically generated