1) #include <stdio.h>

// Define a structure to represent an employee

struct Employee {

int empID;

char empName[50];

float empSalary;

};

int main() {

int n; // Number of employees

printf("Enter the number of employees: ");

scanf("%d", &n);

// Declare an array of structures to store employee information

struct Employee employees[n];

// Input employee details

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

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

printf("Employee ID: ");

scanf("%d", &employees[i].empID);

printf("Employee Name: ");

scanf("%s", employees[i].empName);

printf("Employee Salary: ");

scanf("%f", &employees[i].empSalary);

}

// Display employee details

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

printf("%-10s %-20s %-15s\n", "Employee ID", "Employee Name", "Employee Salary");

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

printf("%-10d %-20s %-15.2f\n", employees[i].empID, employees[i].empName, employees[i].empSalary);

}

return 0;

}

2) #include <stdio.h>

// Function to transpose a 2-D array

void transpose(int matrix[10][10], int rows, int cols, int result[10][10]) {

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

for (int j = 0; j < cols; j++) {

result[j][i] = matrix[i][j];

}

}

}

// Function to display a 2-D array

void displayMatrix(int matrix[10][10], int rows, int cols) {

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

for (int j = 0; j < cols; j++) {

printf("%d\t", matrix[i][j]);

}

printf("\n");

}

}

int main() {

int matrix[10][10], transposed[10][10];

int rows, cols;

// Input the dimensions of the matrix

printf("Enter the number of rows: ");

scanf("%d", &rows);

printf("Enter the number of columns: ");

scanf("%d", &cols);

// Input elements for the matrix

printf("Enter elements of the matrix:\n");

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

for (int j = 0; j < cols; j++) {

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

}

}

// Call the function to transpose the matrix

transpose(matrix, rows, cols, transposed);

// Display the original matrix

printf("\nOriginal Matrix:\n");

displayMatrix(matrix, rows, cols);

// Display the transposed matrix

printf("\nTransposed Matrix:\n");

displayMatrix(transposed, cols, rows);

return 0;

}

3) #include <stdio.h>

// Function prototypes for arithmetic operations

int add(int a, int b);

int subtract(int a, int b);

int multiply(int a, int b);

int divide(int a, int b);

// Function pointer type for arithmetic operations

typedef int (\*ArithmeticFunction)(int, int);

int main() {

int num1, num2;

ArithmeticFunction operation;

// Input two numbers

printf("Enter the first number: ");

scanf("%d", &num1);

printf("Enter the second number: ");

scanf("%d", &num2);

// Display menu for arithmetic operations

printf("\nArithmetic Operations Menu:\n");

printf("1. Addition\n");

printf("2. Subtraction\n");

printf("3. Multiplication\n");

printf("4. Division\n");

// Input user choice

int choice;

printf("Enter your choice (1-4): ");

scanf("%d", &choice);

// Set the function pointer based on the user's choice

switch (choice) {

case 1:

operation = add;

break;

case 2:

operation = subtract;

break;

case 3:

operation = multiply;

break;

case 4:

operation = divide;

break;

default:

printf("Invalid choice. Exiting the program.\n");

return 1;

}

// Perform the selected arithmetic operation using the function pointer

int result = operation(num1, num2);

// Display the result

printf("Result: %d\n", result);

return 0;

}

// Function definitions for arithmetic operations

int add(int a, int b) {

return a + b;

}

int subtract(int a, int b) {

return a - b;

}

int multiply(int a, int b) {

return a \* b;

}

int divide(int a, int b) {

if (b != 0) {

return a / b;

} else {

printf("Error: Cannot divide by zero.\n");

return 0;

}

}

4) #include <stdio.h>

#include <stdlib.h>

#include <string.h>

// Enum to represent data types

typedef enum {

INTEGER,

FLOAT,

CHARACTER,

STRING

} DataType;

// Struct for the generic data container

typedef struct {

DataType type;

void\* data;

} DataContainer;

// Function to set data in the container

void setData(DataContainer\* container, DataType type, void\* data) {

container->type = type;

// Allocate memory and copy data based on the data type

switch (type) {

case INTEGER:

container->data = malloc(sizeof(int));

\*(int\*)(container->data) = \*(int\*)data;

break;

case FLOAT:

container->data = malloc(sizeof(float));

\*(float\*)(container->data) = \*(float\*)data;

break;

case CHARACTER:

container->data = malloc(sizeof(char));

\*(char\*)(container->data) = \*(char\*)data;

break;

case STRING:

container->data = strdup((char\*)data);

break;

default:

printf("Invalid data type.\n");

break;

}

}

// Function to get data from the container

void getData(DataContainer\* container, void\* result) {

if (container->type == INTEGER || container->type == FLOAT || container->type == CHARACTER) {

memcpy(result, container->data, sizeof(int));

} else if (container->type == STRING) {

strcpy((char\*)result, (char\*)container->data);

} else {

printf("Invalid data type.\n");

}

}

// Function to free memory allocated for data in the container

void freeData(DataContainer\* container) {

if (container->type == STRING) {

free(container->data);

}

free(container->data);

}

int main() {

// Example usage of the generic data container

DataContainer container;

// Set integer data

int intValue = 42;

setData(&container, INTEGER, &intValue);

// Get and display integer data

int resultInt;

getData(&container, &resultInt);

printf("Integer value: %d\n", resultInt);

// Set string data

char stringValue[] = "Hello, World!";

setData(&container, STRING, stringValue);

// Get and display string data

char resultString[50];

getData(&container, resultString);

printf("String value: %s\n", resultString);

// Free memory allocated for data in the container

freeData(&container);

return 0;

}

5) #include <stdio.h>

void countDuplicates(int arr[], int size) {

int duplicateCount = 0;

// Iterate through each element in the array

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

// Check if the current element is a duplicate

if (arr[i] != -1) {

for (int j = i + 1; j < size; j++) {

if (arr[i] == arr[j]) {

// Mark the duplicate elements with -1

arr[j] = -1;

duplicateCount++;

printf("Duplicate found at indices %d and %d\n", i, j);

}

}

}

}

if (duplicateCount == 0) {

printf("No duplicates found in the array.\n");

} else {

printf("Total number of duplicate elements: %d\n", duplicateCount);

}

}

int main() {

int size;

// Input the size of the array

printf("Enter the size of the array: ");

scanf("%d", &size);

int arr[size];

// Input elements for the array

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

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

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

}

// Call the function to count and display duplicates

countDuplicates(arr, size);

return 0;

}

6) #include <stdio.h>

// Function to point to the upper triangle of a matrix

void upperTriangle(int matrix[10][10], int rows, int cols, int \*upperTriangleArray) {

int index = 0;

// Iterate through the rows

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

// Iterate through the columns

for (int j = 0; j < cols; j++) {

// Check if the element is above or on the diagonal

if (j >= i) {

// Point to the upper triangle element

upperTriangleArray[index] = matrix[i][j];

index++;

}

}

}

}

int main() {

int matrix[10][10];

int rows, cols;

// Input the dimensions of the matrix

printf("Enter the number of rows: ");

scanf("%d", &rows);

printf("Enter the number of columns: ");

scanf("%d", &cols);

// Input elements for the matrix

printf("Enter elements of the matrix:\n");

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

for (int j = 0; j < cols; j++) {

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

}

}

// Calculate the size of the upper triangle array

int upperTriangleSize = (rows \* (rows + 1)) / 2;

// Declare an array to store the upper triangle elements

int upperTriangleArray[upperTriangleSize];

// Call the function to point to the upper triangle

upperTriangle(matrix, rows, cols, upperTriangleArray);

// Display the upper triangle elements

printf("\nUpper Triangle Elements:\n");

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

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

}

printf("\n");

return 0;

}

7) #include <stdio.h>

#include <stdlib.h>

#include <string.h>

// Custom comparison function for sorting integers to maximize the resulting value

int compare(const void \*a, const void \*b) {

char str1[20], str2[20];

sprintf(str1, "%d%d", \*(int\*)a, \*(int\*)b);

sprintf(str2, "%d%d", \*(int\*)b, \*(int\*)a);

return strcmp(str2, str1);

}

// Function to print the largest integer that can be made from an array of distinct integers

void printLargestInteger(int arr[], int n) {

// Sort the array using the custom comparison function

qsort(arr, n, sizeof(int), compare);

// Concatenate the sorted integers to form the largest integer

printf("The largest integer: ");

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

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

}

printf("\n");

}

int main() {

int n;

// Input the size of the array

printf("Enter the size of the array: ");

scanf("%d", &n);

int arr[n];

// Input elements for the array

printf("Enter %d distinct integers from the range [0, 10000]:\n", n);

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

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

}

// Call the function to print the largest integer

printLargestInteger(arr, n);

return 0;

}

8) #include <stdio.h>

#include <string.h>

// Define a structure to represent a person with name and address

struct Person {

char name[50];

char address[100];

};

// Function to compare two names for sorting

int compareNames(const void \*a, const void \*b) {

return strcmp(((struct Person \*)a)->name, ((struct Person \*)b)->name);

}

int main() {

int n;

// Input the number of persons

printf("Enter the number of persons: ");

scanf("%d", &n);

// Declare an array of structures to store persons' information

struct Person persons[n];

// Input names and addresses

printf("Enter names and corresponding addresses:\n");

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

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

printf("Name: ");

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

printf("Address: ");

scanf("%s", persons[i].address);

}

// Sort the names alphabetically

qsort(persons, n, sizeof(struct Person), compareNames);

// Display the alphabetically sorted names and corresponding addresses

printf("\nAlphabetically Sorted Names and Addresses:\n");

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

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

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

printf("Address: %s\n", persons[i].address);

}

return 0;

}

9) #include <stdio.h>

#include <string.h>

// Function to check the validity of a credit card number

int isValidCreditCard(char \*creditCardNumber) {

int len = strlen(creditCardNumber);

int sum = 0;

// Start with the rightmost digit (last character in the string)

for (int i = len - 1; i >= 0; i--) {

int digit = creditCardNumber[i] - '0';

// Multiply every other digit by 2

if ((len - i) % 2 == 0) {

digit \*= 2;

// Subtract 9 from any number larger than 10

if (digit > 9) {

digit -= 9;

}

}

// Add the digit to the sum

sum += digit;

}

// Check if the sum is divisible by 10

return (sum % 10 == 0);

}

int main() {

char creditCardNumber[17];

// Input the credit card number

printf("Enter the 16-digit credit card number (without spaces): ");

scanf("%s", creditCardNumber);

// Check the validity of the credit card number

if (strlen(creditCardNumber) == 16 && isValidCreditCard(creditCardNumber)) {

printf("The credit card number is valid.\n");

} else {

printf("The credit card number is not valid.\n");

}

return 0;

}

10) #include <stdio.h>

#include <ctype.h>

// Function to check if a character is a vowel

int isVowel(char ch) {

ch = tolower(ch);

return (ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u');

}

// Function to count occurrences of any two vowels in succession

int countSuccessiveVowels(char \*text) {

int count = 0;

while (\*text) {

if (isVowel(\*text) && isVowel(\*(text + 1))) {

count++;

printf("Found: %c%c\n", \*text, \*(text + 1));

}

text++;

}

return count;

}

int main() {

char text[100];

// Input the line of text

printf("Enter a line of text: ");

fgets(text, sizeof(text), stdin);

// Count occurrences of any two vowels in succession

int result = countSuccessiveVowels(text);

// Display the result

printf("Number of occurrences of any two vowels in succession: %d\n", result);

return 0;

}