**Problem Solving**

**Difficulty Level:Easy**

**Problems:**

1. Check Whether a Given Number is Even or Odd

**Even Number:**  
A number is said to be an **even number** if it is completely divisible by 2.  
In other words, if a number is divided by 2 and leaves a remainder of 0, then it is said to be an even number.  
**Example:** 36, 24

**Odd Number:**  
A number is said to be an **odd number** if it is not completely divisible by 2.  
In other words, if a number is divided by 2 and the remainder is 1, it is said to be an odd number.  
**Example:** 21, 15

#include <stdio.h>

int main() {

int num;

printf("Enter an integer: ");

scanf("%d", &num);

// true if num is perfectly divisible by 2

if(num % 2 == 0)

printf("%d is even.", num);

else

printf("%d is odd.", num);

return 0;

}

Output : Enter an integer: 56

56 is even.

1. Find the Sum of Even and Odd Numbers

Problem Description:

The program takes the number N and finds the sum of odd and even numbers from 1 to N.

#include <stdio.h>

void main()

{

int i, num, odd\_sum = 0, even\_sum = 0;

printf("Enter the value of num\n");

scanf("%d", &num);

for (i = 1; i <= num; i++)

{

if (i % 2 == 0)

even\_sum = even\_sum + i;

else

odd\_sum = odd\_sum + i;

}

printf("Sum of all odd numbers = %d\n", odd\_sum);

printf("Sum of all even numbers = %d\n", even\_sum);

}

1. Check Whether a Number is Positive or Negative

Problem Description:

The program takes the given integer and checks whether the integer is positive or negative.

#include <stdio.h>

int main() {

double num;

printf("Enter a number: ");

scanf("%lf", &num);

if (num <= 0.0) {

if (num == 0.0)

printf("You entered 0.");

else

printf("You entered a negative number.");

}

else

printf("You entered a positive number.");

return 0;

}

1. Find the Largest Number Among Three Numbers.

In C programming, the largest element of a number is the number with the highest numerical value of the three numbers.

For example, if three numbers are given, 1, 2, 3, the largest of the three numbers is 3.

Problem Description

Write a C program that takes the 3 numbers and finds the largest number Aamong three numbers.

#include <stdio.h>

int main() {

double n1, n2, n3;

printf("Enter three different numbers: ");

scanf("%lf %lf %lf", &n1, &n2, &n3);

// if n1 is greater than both n2 and n3, n1 is the largest

if (n1 >= n2 && n1 >= n3)

printf("%.2f is the largest number.", n1);

// if n2 is greater than both n1 and n3, n2 is the largest

if (n2 >= n1 && n2 >= n3)

printf("%.2f is the largest number.", n2);

// if n3 is greater than both n1 and n2, n3 is the largest

if (n3 >= n1 && n3 >= n2)

printf("%.2f is the largest number.", n3);

return 0;

1. Swap Two Numbers

Swapping two numbers in C programming means swapping the values of two variables. For example, there are two variables m & n. Value of m is “2” & value of n is “3”.

Before Swapping: m value = 2; n value = 3

After Swapping: m value = 3; n value = 2

#include<stdio.h>

int main() {

double first, second, temp;

printf("Enter first number: ");

scanf("%lf", &first);

printf("Enter second number: ");

scanf("%lf", &second);

// value of first is assigned to temp

temp = first;

// value of second is assigned to first

first = second;

// value of temp (initial value of first) is assigned to second

second = temp;

// %.2lf displays number up to 2 decimal points

printf("\nAfter swapping, first number = %.2lf\n", first);

printf("After swapping, second number = %.2lf", second);

return 0;

}

1. Find the Number of Integers Divisible by 5

Problem Description

1. This problem takes the range as input and finds the number of integers divisible by 5 in the given range.

2. Also finds the sum of all integers that are divisible by 5 in the given range.

#include <stdio.h>

void main()

{

int i, num1, num2, count = 0, sum = 0;

printf("Enter the value of num1 and num2 \n");

scanf("%d %d", &num1, &num2);

/\* Count the number and compute their sum\*/

printf("Integers divisible by 5 are \n");

for (i = num1; i < num2; i++)

{

if (i % 5 == 0)

{

printf("%3d,", i);

count++;

sum = sum + i;

}

}

printf("\n Number of integers divisible by 5 between %d and %d =

%d\n", num1, num2, count);

printf("Sum of all integers that are divisible by 5 = %d\n", sum);

}

1. Check if Two Numbers are Equal

Problem Description

This problem accepts two integers and check if they are equal or not.

#include <stdio.h>

int main() {

int num1, num2;

printf("Enter First Number: ");

scanf("%d", &num1);

printf("Enter Second Number: ");

scanf("%d", &num2);

if (num1 == num2) {

printf("Both numbers are equal.");

} else {

printf("Numbers are not equal");

}

return 0;

}

1. Sum of Digits

Sum of digitsproblem allows a user to enter any number, divide that number into individual numbers, and sum those individual numbers.

Example 1:

Given number = 14892 => 1 + 4 + 8 + 9 + 2 = 24.

Sum of digits of a given number “14892” is 24.

Example 2:

Given number = 3721 => 3 + 7 + 2 + 1 = 13.

Sum of digits of a given number “3721” is 13.

#include<stdio.h>

int main()

{

int n,sum=0,m;

printf("Enter a number:");

scanf("%d",&n);

while(n>0)

{

m=n%10;

sum=sum+m;

n=n/10;

}

printf("Sum is=%d",sum);

return 0;

}

1. Increment by 1 to all the Digits of a Given Integer

Problem Description

Increases 1 to all of the given integer digit and print the sum of all digits.

#include <stdio.h>

int main()

{

int number, sum = 0, remainder, count;

printf("Enter a number: ");

scanf("%d", &number);

while (number)

{

remainder = number % 10;

sum = sum + (remainder + 1);

number /= 10;

}

printf("increasing 1 to all digits: %d", sum);

return 0;

}

1. Multiplication Table

A multiplication table of numbers is created by multiplying a constant integer by a number of repetitions ranging from 1 to 10.

#include <stdio.h>

int main() {

int n;

printf("Enter an integer: ");

scanf("%d", &n);

for (int i = 1; i <= 10; ++i) {

printf("%d \* %d = %d \n", n, i, n \* i);

}

return 0;

}

1. Count the Number of Vowels and Consonants in a Sentence

Problem Description

This problem takes the sentence as input and counts the number of vowels & consonants in a sentence.

#include <stdio.h>

int main() {

char line[150];

int vowels, consonant, digit, space;

// initialize all variables to 0

vowels = consonant = digit = space = 0;

// get full line of string input

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

fgets(line, sizeof(line), stdin);

// loop through each character of the string

for (int i = 0; line[i] != '\0'; ++i) {

// convert character to lowercase

line[i] = tolower(line[i]);

// check if the character is a vowel

if (line[i] == 'a' || line[i] == 'e' || line[i] == 'i' ||

line[i] == 'o' || line[i] == 'u') {

// increment value of vowels by 1

++vowels;

}

// if it is not a vowel and if it is an alphabet, it is a consonant

else if ((line[i] >= 'a' && line[i] <= 'z')) {

++consonant;

}

// check if the character is a digit

else if (line[i] >= '0' && line[i] <= '9') {

++digit;

}

// check if the character is an empty space

else if (line[i] == ' ') {

++space;

}

}

printf("Vowels: %d", vowels);

printf("\nConsonants: %d", consonant);

printf("\nDigits: %d", digit);

printf("\nWhite spaces: %d", space);

return 0;

}

1. Accept the Height of a Person & Categorize as Taller, Dwarf & Average

Problem Description

This problem accepts the height of a person as input and categorizes as taller, dwarf & average.

#include <stdio.h>

void main()

{

float height;

printf("Enter the Height (in centimetres) \n");

scanf("%f", &height);

if (height < 150.0)

printf("Dwarf \n");

else if ((height >= 150.0) && (height <= 165.0))

printf(" Average Height \n");

else if ((height > 165.0) && (height <= 195.0))

printf("Taller \n");

else

printf("Abnormal height \n");

}

1. Prime Number

A prime number is a natural number that is greater than 1 and is only divisible by 1 and itself. In other words, no number except the number itself, and 1 can divide a prime number.

Example: 2, 3, 5, 7, 11, 13, 17, 19 …., etc.

Problem Description

check if a given number is Prime number. If the number is Prime, then display it is a prime number else display it is not a prime number.

#include<stdio.h>

int main(){

int n,i,m=0,flag=0;

printf("Enter the number to check prime:");

scanf("%d",&n);

m=n/2;

for(i=2;i<=m;i++)

{

if(n%i==0)

{

printf("Number is not prime");

flag=1;

break;

}

}

if(flag==0)

printf("Number is prime");

return 0;

}

1. Check Whether a Given Number is Perfect Number

A perfect number is a number that is equal to the sum of its proper divisors. For example, the divisors of 6 are 1, 2 and 3. The sum of the proper divisors of 6 is 1 + 2 + 3 = 6, which is a perfect number. The sum of the proper divisors of 28 is 1 + 2 + 4 + 7 + 14 = 28, which is also a perfect number.

Problem Description

Ask the user for a number and then check whether the number is a perfect number or not.

#include<stdio.h>

int main(){

int number,i,result=0;//declare variables and initialize result to 0

printf("enter the number:");

scanf("%d",&number);

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

if(number%i==0)

result=result+i;

}

if(result==2\*number) //checking the sum of factors==2\*number

printf("perfect number");

else

printf("not perfect number");

}

1. Check Armstrong Number

Armstrong Number in C: An Armstrong number is an n-digit base b number such that the sum of its (base b) digits raised to the power n is the number itself. Armstrong numbers are 0, 1, 153, 370, 371, 407, etc.

Armstrong Number Formula: wxyz = pow(w,n) + pow(x,n) + pow(y,n) + pow(z,n)

#include <stdio.h>

int main() {

int num, originalNum, remainder, result = 0;

printf("Enter a three-digit integer: ");

scanf("%d", &num);

originalNum = num;

while (originalNum != 0) {

// remainder contains the last digit

remainder = originalNum % 10;

result += remainder \* remainder \* remainder;

// removing last digit from the orignal number

originalNum /= 10;

}

if (result == num)

printf("%d is an Armstrong number.", num);

else

printf("%d is not an Armstrong number.", num);

return 0;

}

1. Reverse a Number

Reverse a Number means moving the digit at the last position to the first position and vice versa.

For example, if the given number is “1234”, the reverse number will be “4321”.

#include <stdio.h>

int main() {

int n, reverse = 0, remainder;

printf("Enter an integer: ");

scanf("%d", &n);

while (n != 0) {

remainder = n % 10;

reverse = reverse \* 10 + remainder;

n /= 10;

}

printf("Reversed number = %d", reverse);

return 0;

}

1. Reverse a Number and Check if it is a Palindrome

Problem Description

Accepts an integer, reverse it and also checks if it is a palindrome or not.

#include <stdio.h>

int main() {

int n, reversed = 0, remainder, original;

printf("Enter an integer: ");

scanf("%d", &n);

original = n;

// reversed integer is stored in reversed variable

while (n != 0) {

remainder = n % 10;

reversed = reversed \* 10 + remainder;

n /= 10;

}

// palindrome if orignal and reversed are equal

if (original == reversed)

printf("%d is a palindrome.", original);

else

printf("%d is not a palindrome.", original);

return 0;

}

1. C Program to Add Two Binary Numbers

Problem Description

This program finds the sum of two binary numbers.

#include <stdio.h>

int main()

{

long binary1, binary2;

int i = 0, remainder = 0, sum[20];

printf("Enter the first binary number: ");

scanf("%ld", &binary1);

printf("Enter the second binary number: ");

scanf("%ld", &binary2);

while (binary1 != 0 || binary2 != 0)

{

sum[i++] =(binary1 % 10 + binary2 % 10 + remainder) % 2;

remainder =(binary1 % 10 + binary2 % 10 + remainder) / 2;

binary1 = binary1 / 10;

binary2 = binary2 / 10;

}

if (remainder != 0)

sum[i++] = remainder;

--i;

printf("Sum of two binary numbers: ");

while (i >= 0)

printf("%d", sum[i--]);

return 0;

}

1. Find Prime Numbers in a Given Range

Problem Description

Take the range and finds all the prime numbers between the range and also prints the number of prime numbers.

#include <stdio.h>

int main() {

int low, high, i, flag;

printf("Enter two numbers(intervals): ");

scanf("%d %d", &low, &high);

printf("Prime numbers between %d and %d are: ", low, high);

// iteration until low is not equal to high

while (low < high) {

flag = 0;

// ignore numbers less than 2

if (low <= 1) {

++low;

continue;

}

// if low is a non-prime number, flag will be 1

for (i = 2; i <= low / 2; ++i) {

if (low % i == 0) {

flag = 1;

break;

}

}

if (flag == 0)

printf("%d ", low);

// to check prime for the next number

// increase low by 1

++low;

}

return 0;

}

1. Leap Year

Leap Year:A year is a Leap Year if it satisfies the following conditions:

The year is exactly divisible by 400 (such as 2000,2400) or,

The year is exactly divisible by 4 (such as 2008, 2012, 2016) and not a multiple of 100 (such as 1900, 2100, 2200).

// C program to check if a given

// year is leap year or not

#include <stdio.h>

#include <stdbool.h>

bool checkYear(int year)

{

// If a year is multiple of 400,

// then it is a leap year

if (year % 400 == 0)

return true;

// Else If a year is multiple of 100,

// then it is not a leap year

if (year % 100 == 0)

return false;

// Else If a year is multiple of 4,

// then it is a leap year

if (year % 4 == 0)

return true;

return false;

}

// driver code

int main()

{

int year = 2000;

checkYear(year)? printf("Leap Year"):

printf("Not a Leap Year");

return 0;

}

1. Fibonacci Series

Fibonacci series are the numbers in the sequence 0, 1, 1, 2, 3, 5, 8, 13, 21….. The series in the Fibonacci sequence is equal to the sum of the previous two terms. The Fibonacci sequence’s first two terms are 0 and 1 respectively.

#include <stdio.h>

int main() {

int i, n;

// initialize first and second terms

int t1 = 0, t2 = 1;

// initialize the next term (3rd term)

int nextTerm = t1 + t2;

// get no. of terms from user

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

scanf("%d", &n);

// print the first two terms t1 and t2

printf("Fibonacci Series: %d, %d, ", t1, t2);

// print 3rd to nth terms

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

printf("%d, ", nextTerm);

t1 = t2;

t2 = nextTerm;

nextTerm = t1 + t2;

}

return 0;

}

1. Factorial

When you multiply a positive integer by all the integers smaller than that positive integer, you get its factorial.

For example, factorial of 3 is 3! = 1\*2\*3 = 6 and factorial of 6 is 6! = 6 \* 5 \* 4 \* 3 \* 2 \* 1 which equals to 720.

By default, the factorial of 0 is 1, and Factorial of a negative number is not defined.

#include <stdio.h>

int main() {

int n, i;

unsigned long long fact = 1;

printf("Enter an integer: ");

scanf("%d", &n);

// shows error if the user enters a negative integer

if (n < 0)

printf("Error! Factorial of a negative number doesn't exist.");

else {

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

fact \*= i;

}

printf("Factorial of %d = %llu", n, fact);

}

return 0;

}

1. Floyd’s Triangle

Floyds Triangle in C is a right-angled triangular array of natural numbers. It is defined by filling the rows of the triangle with consecutive numbers, starting with a 1 in the top left corner: 1. Number of rows of Floyd’s triangle to print is entered by the user. For loop is used to print the output of the program.

Example:

A Floyd’s triangle is a triangle in which each number is the sum of the two numbers above it. For example, the first row of the Floyd’s triangle is 1, the second row is 2 + 1 = 3, and so on. The following is a diagram of the Floyd’s triangle:

1

2 3

4 5 6

7 8 9 10

#include <stdio.h>

#include <conio.h>

void main()

{

int num, i, j, k = 1;

printf( " Enter a number to define the rows in Floyd's triangle: \n");

scanf( "%d", &num);

// use nested for loop

// outer for loop define the rows and check rows condition

for (i = 1; i <= num; i++)

{

// inner loop check j should be less than equal to 1 and print the data.

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

{

printf(" %2d", k++); // print the number

}

printf( "\n");

}

getch();

}

1. Pascal Triangle

Pascal Triangle is a pattern similar to a triangle. Firstly, 1 is placed at the top, and then we start putting the numbers in a triangular pattern. The numbers which we get in each step are the addition of the above two numbers.

Enter the Number of Rows in the Pascal Triangle:: 6

1

1 1

1 2 1

1 3 3 1

1 4 6 4 1

1 5 10 10 5 1

#include

int main()

{

int rows, coef = 1, space, i, j;

printf(“\nEnter the number of rows : “);

scanf(“%d”,&rows);

printf(“\n”);

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

{

for(space=1; space <= rows-i; space++)

printf(” “);

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

{

if (j==0 || i==0)

coef = 1;

else

coef = coef\*(i-j+1)/j;

printf(“%4d”, coef);

}

printf(“\n\n”);

}

return 0;

}

1. A star pattern is a pattern that shows up as a staircase of stars.

\*

\*\*

\*\*\*

\*\*\*\*

\*\*\*\*\*

#include <stdio.h>

int main()

{

int n;

printf("Enter the number of rows");

scanf("%d",&n);

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

{

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

{

printf("\* ");

}

printf("\n");

}

return 0;

}

1. Rhombus Star Pattern in C

\*  
\*\*\*  
\*\*\*\*\*  
\*\*\*  
\*

#include <stdio.h>

int main() {

int i, j, rows;

printf("Enter the number of rows\n");

scanf("%d", &rows);

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

/\* Print spaces before stars in a row \*/

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

printf(" ");

}

/\* Print rows stars after spaces in a row \*/

for (j = 0; j < rows; j++) {

printf("\*");

}

/\* jump to next row \*/

printf("\n");

}

return 0;

}

1. Diamond Star Pattern

Enter the number of rows: 5

\*

\*\*\*

\*\*\*\*\*

\*\*\*

\*

#include<stdio.h>

int main() {

int i, space, rows=7, star=0;

/\* Printing upper triangle \*/

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

/\* Printing spaces \*/

for(space = 1; space <= rows-i; space++) {

printf(" ");

}

/\* Printing stars \*/

while(star != (2\*i - 1)) {

printf("\*");

star++;;

}

star=0;

/\* move to next row \*/

printf("\n");

}

rows--;

/\* Printing lower triangle \*/

for(i = rows;i >= 1; i--) {

/\* Printing spaces \*/

for(space = 0; space <= rows-i; space++) {

printf(" ");

}

/\* Printing stars \*/

star = 0;

while(star != (2\*i - 1)) {

printf("\*");

star++;

}

printf("\n");

}

return 0;

}

Output

\*

\*\*\*

\*\*\*\*\*

\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*

\*\*\*\*\*

\*\*\*

\*

1. Find the Area of a Circle

Area of circle is defined as pi\*r\*r where pi is a constant whose value is (22/7 or 3.142) and r is the radius of a circle.

Formula to calculate the area of circle is: Area = pi\*r\*r

#include <stdio.h>

int main(void) {

float pie = 3.14;

int radius = 6;

printf("The radius of the circle is %d

" , radius);

float area = (float)(pie\* radius \* radius);

printf("The area of the given circle is %f", area);

return 0;

}

1. Find the Area of a Triangle

The area of ​​a triangle is defined as the total area bounded by the three sides of a given triangle.

Area of a Triangle Formula:

If the base and height are given, the area of the triangle is determined using the formula

A = 1/2∗b∗h

Enter Base and Height: 10 5

Area of Triangle is 25.00

#include<stdio.h>

int main()

{ float b ,h, area;

b= 5;

h= 13;

area = (b\*h) / 2 ;

printf("\n\n Area of Triangle is: %f",area);

return (0);

}

1. Find GCD and LCM of Two Integers

GCD (Greatest Common Divisor)

GCD stands for Greatest Common Divisor. GCD of two numbers is the largest positive integer that completely divides both the given numbers.

Example: GCD(10,15) = 15, GCD(12,15) = 3.

LCM (Least Common Multiple)

LCM stands for Least Common Multiple. It is a method to find the lowest common multiple between the two numbers. LCM of two numbers is the lowest possible number that is divisible by both numbers.

Examples: LCM(10,15) = 30, LCM(12,15) = 60.

Enter two numbers:

12 15

GCD of 12 and 15 = 3

LCM of 12 and 15 = 60

#include <stdio.h>

int main() {

int n1, n2, max;

printf("Enter two positive integers: ");

scanf("%d %d", &n1, &n2);

// maximum number between n1 and n2 is stored in max

max = (n1 > n2) ? n1 : n2;

while (1) {

if ((max % n1 == 0) && (max % n2 == 0)) {

printf("The LCM of %d and %d is %d.", n1, n2, max);

break;

}

++max;

}

return 0;

}

1. Find HCF of Two Numbers

HCF stands for Highest Common Factor. HCF of two numbers is the largest positive integer that completely divides both the given numbers.

Example: HCF(10,15) = 15, HCF(12,15) = 3.

Enter two numbers

12 15

HCF of 12 and 15 = 3.

#include <stdio.h>

int main()

{

int n1, n2, i, gcd;

printf("Enter two integers: ");

scanf("%d %d", &n1, &n2);

for(i=1; i <= n1 && i <= n2; ++i)

{

// Checks if i is factor of both integers

if(n1%i==0 && n2%i==0)

gcd = i;

}

printf("G.C.D of %d and %d is %d", n1, n2, gcd);

return 0;

}

1. Compare Two Strings

String is a sequence of characters terminated by the special character ‘\0’. Strings can be compared with or without using the string function.

Example:

String1=”Hello” String2=”Hello” Both string are equal

String1=”Hello” String2=”Hell” String1 is greater

String1=”Hello” String2=”Helz” String2 is greater

Enter the First string

hello

Enter the Second string

hell

First string is greater than Second string

#include <stdio.h>

#include<string.h>

int main()

{

char str1[20]; // declaration of char array

char str2[20]; // declaration of char array

int value; // declaration of integer variable

printf("Enter the first string : ");

scanf("%s",str1);

printf("Enter the second string : ");

scanf("%s",str2);

// comparing both the strings using strcmp() function

value=strcmp(str1,str2);

if(value==0)

printf("strings are same");

else

printf("strings are not same");

return 0;

}

1. Check Whether a Number is Palindrome or Not

A number is said to be a palindrome number if it reads the same forward and backward i.e., on reversing the digits of the number we get the same number.

Enter the number: 121

121 is a palindrome number.

Enter the number: 342

342 is not a palindrome number.

1. String Palindrome

String Palindrome: A palindrome is a word, phrase or sentence that reads the same backward or forward. A string is said to be a palindromic string when we traverse it from start to end or end to start then we get the same result.

Enter a string: sanfoundry

sanfoundry is not a palindrome

Enter a string: malayalam

malayalam is a palindrome

#include <stdio.h>

#include <string.h>

int main(){

char string1[20];

int i, length;

int flag = 0;

printf("Enter a string:");

scanf("%s", string1);

length = strlen(string1);

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

if(string1[i] != string1[length-i-1]){

flag = 1;

break;

}

}

if (flag) {

printf("%s is not a palindrome", string1);

}

else {

printf("%s is a palindrome", string1);

}

return 0;

}

1. Anagram

Anagram: Two strings are said to be anagrams if they satisfy two conditions, the length of both strings must be equal to each other and second the strings must have the same set of characters.

Example 1:

First String = “hectare” and Second String = “teacher”

Case 1:

Lengths must be equal to each other.

length of “hectare” = 7

length of “teacher” = 7

Case 1 passed.

Case 2:

Set of characters in

hectare {‘h’ , ’e’ , ’c’ , ’t’ , ’a’ , ’r’ , ’e’}

teacher {‘t’ , ’e’ , ’a’ , ’c’ , ’h’ , ’e’, ’r’}

Every character from the first string has a similar character to it in the other string. Case 2 passed.

”teacher” and ”hectare” are anagrams.

Enter the string

study

Enter another string

dusty

"study" and "dusty" are anagrams.

#include<stdio.h>

#include<conio.h>

#include<string.h>

int main()

{

char str1[20], str2[20];

int len, len1, len2, i, j, found=0, not\_found=0;

printf("Enter first string: ");

gets(str1);

printf("Enter second string: ");

gets(str2);

len1 = strlen(str1);

len2 = strlen(str2);

if(len1 == len2)

{

len = len1;

for(i=0; i<len; i++)

{

found = 0;

for(j=0; j<len; j++)

{

if(str1[i] == str2[j])

{

found = 1;

break;

}

}

if(found == 0)

{

not\_found = 1;

break;

}

}

if(not\_found == 1)

printf("\nStrings are not Anagram");

else

printf("\nStrings are Anagram");

}

else

printf("\nBoth string must contain same number of character to be an Anagram Strings");

getch();

return 0;

}

1. Calculate the Power of a Number

For example: In the case of 23

2 is the base number

3 is the exponent

And, the power is equal to 2\*2\*2

Sample input

Base number: 2

Exponent number: 3

Output:

8

#include <stdio.h>

int main() {

int base, exp;

long double result = 1.0;

printf("Enter a base number: ");

scanf("%d", &base);

printf("Enter an exponent: ");

scanf("%d", &exp);

while (exp != 0) {

result \*= base;

--exp;

}

printf("Answer = %.0Lf", result);

return 0;

}

1. Print the sum of all even numbers between 1 and 100.

Example Solution: Sum of even numbers between 1 and 100: 2550

#include <stdio.h>

int main()

{

int i, n, sum=0;

/\* Input upper limit from user \*/

printf("Enter upper limit: ");

scanf("%d", &n);

for(i=2; i<=n; i+=2)

{

/\* Add current even number to sum \*/

sum += i;

}

printf("Sum of all even number between 1 to %d = %d", n, sum);

return 0;

}

1. Check if a given number is a perfect square or not.

Sample Input: Enter a number: 25

Sample Output: 25 is a perfect square.

#include<stdio.h>

int main()

{

printf("\n\n\t\tStudytonight - Best place to learn\n\n\n");

// variable declaration

int i, number;

// take user input

printf("Enter a number: ");

scanf("%d", &number);

// loop to check number is perfect square or not

for(i = 0; i <= number; i++)

{

if(number == i\*i)

{

printf("\n\n\n\t\t\t%d is a perfect square\n\n\n", number);

printf("\n\n\t\t\tCoding is Fun !\n\n\n");

return 0; // same as using break in this case to end the program

}

}

printf("\n\n\n\t\t\t%d is not a perfect square\n", number);

printf("\n\n\t\t\tCoding is Fun !\n\n\n");

return 0;

}

1. Find the sum of all even digits in a given number.

Sample:

Input: Enter a number: 356824

Output:Sum of even digits: 20

#include<stdio.h>

int sum\_of\_even\_digits(int n) {

int r, sum = 0;

// reading each digit of n

while (n > 0) {

r = n % 10; // storing rightmost digit of n in r

n = n / 10; // removing rightmost digit of n

// if r is even, add r to sum

if (r % 2 == 0){

sum = sum + r;

}

}

return sum;

}

int main() {

int n;

printf("Enter a Number: ");

scanf("%d", &n);

printf("Sum of Even Digits: %d", sum\_of\_even\_digits(n));

}

1. Swap two numbers without using a temporary variable.

Sample:

Input:

Enter first number: 10

Enter second number: 20

Output:

Before swapping: num1 = 10, num2 = 20

After swapping: num1 = 20, num2 = 10

#include<stdio.h>

int main()

{

int a=10, b=20;

printf("Before swap a=%d b=%d",a,b);

a=a+b;//a=30 (10+20)

b=a-b;//b=10 (30-20)

a=a-b;//a=20 (30-10)

printf("\nAfter swap a=%d b=%d",a,b);

return 0;

}

1. Find the Number of Elements in an Array

array[]={15,50,34,20,10,79,100};

Size of the given array is 7

#include <stdio.h>

int main()

{

//Initialize array

int arr[] = {1, 2, 3, 4, 5};

//Number of elements present in an array can be calculated as follows

int length = sizeof(arr)/sizeof(arr[0]);

printf("Number of elements present in given array: %d", length);

return 0;

}

1. Delete an Element from an Array

Example: arr[6] = {12,65,32,75,48,11}

Value: 12 65 32 75 48 11

↑ ↑ ↑ ↑ ↑ ↑

Index: 0 1 2 3 4 5

The Element we are deleting here is “75”.

Original Array:

12 65 32 75 48 11

New Array:

12 65 32 48 11

/\* program to remove the specific elements from an array in C. \*/

#include <stdio.h>

#include <conio.h>

int main ()

{

// declaration of the int type variable

int arr[50];

int pos, i, num; // declare int type variable

printf (" \n Enter the number of elements in an array: \n ");

scanf (" %d", &num);

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

// use for loop to insert elements one by one in array

for (i = 0; i < num; i++ )

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

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

}

// enter the position of the element to be deleted

printf( " Define the position of the array element where you want to delete: \n ");

scanf (" %d", &pos);

// check whether the deletion is possible or not

if (pos >= num+1)

{

printf (" \n Deletion is not possible in the array.");

}

else

{

// use for loop to delete the element and update the index

for (i = pos - 1; i < num -1; i++)

{

arr[i] = arr[i+1]; // assign arr[i+1] to arr[i]

}

printf (" \n The resultant array is: \n");

// display the final array

for (i = 0; i< num - 1; i++)

{

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

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

}

}

return 0;

}

Output:

Enter the number of elements in an array:

8

Enter 8 elements in array:

arr[0] = 3

arr[1] = 6

arr[2] = 2

arr[3] = 15

arr[4] = 10

arr[5] = 5

arr[6] = 8

arr[7] = 12

Define the position of the array element where you want to delete:

5

The resultant array is:

arr[0] = 3

arr[1] = 6

arr[2] = 2

arr[3] = 15

arr[4] = 5

arr[5] = 8

arr[6] = 12

1. Find Sum of Array Elements using Pointer

Expected Input and Output

If we are entering 5 elements (N = 5), with array element values as 4, 9, 10, 56 and 100 then,

Sum of Elements of the array will be: 4 + 9 + 10 + 56 + 100 = 179

#include <stdio.h>

#include <malloc.h>

void main()

{

int i, n, sum = 0;

int \*a;

printf("Enter the size of array A \n");

scanf("%d", &n);

a = (int \*) malloc(n \* sizeof(int));

printf("Enter Elements of the List \n");

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

{

scanf("%d", a + i);

}

/\* Compute the sum of all elements in the given array \*/

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

{

sum = sum + \*(a + i);

/\* this \*(a+i) is used to access the value stored at the address\*/

}

printf("Sum of all elements in array = %d\n", sum);

return 0;

}

1. Print all Non Repeated Elements in an Array

Enter size of the array: 6

Enter 6 elements of an array: 12

10

4

10

12

56

The array after removing duplicates is: 12 10 4 56

1. Cyclically Permute the Elements of an Array

Enter the value of the n = 4

Enter the numbers

3

40

100

68

Cyclically permuted numbers are given below

40

100

68

3

#include <stdio.h>

void main ()

{

int i, n, number[30];

printf("Enter the value of the n = ");

scanf("%d", &n);

printf("Enter the numbers\n");

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

{

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

}

number[n] = number[0];

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

{

number[i] = number[i + 1];

}

printf("Cyclically permuted numbers are given below \n");

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

printf("%d\n", number[i]);

}

1. Find Missing Numbers in Array

Enter size of array : 6

Enter elements into array :

1

2

3

5

6

Missing element is : 4

1. Find Union and Intersection of Two Arrays

Enter the elements of Array 1:

Enter element 1: 12

Enter element 2: 34

Enter element 3: 23

Enter element 4: 56

Enter element 5: 45

Elements of Array 1: { 12 34 23 56 45 }

Sorted elements of Array 1: { 12 23 34 45 56 }

Enter the elements of Array 2:

Enter element 1: 34

Enter element 2: 56

Enter element 3: 12

Enter element 4: 78

Enter element 5: 66

Elements of Array 2: { 34 56 12 78 66 }

Sorted elements of Array 2: { 12 34 56 66 78 }

Intersection is: { 12 34 56 }

Union is: { 12 23 34 45 56 66 78 }

#include <stdio.h>

#define SIZE 5

void get\_value(int arr[]);

void print\_value(int arr[], int n);

void function\_sort(int arr[]);

int find\_intersection(int array1[], int array2[], int intersection\_array[]);

int find\_union(int array1[], int array2[], int union\_array[]);

void main()

{

int array1[SIZE], array2[SIZE], intersection\_array[SIZE], union\_array[SIZE\*2];

int num\_elements;

//input elements of Array1

printf("\n Enter the elements of Array 1: n");

get\_value(array1);

printf("\n\n Elements of Array 1: ");

print\_value(array1, SIZE);

//Sort array 1

function\_sort(array1);

printf("nnSorted elements of Array 1: ");

print\_value(array1, SIZE);

//input elements of Array2

printf("nnEnter the elements of Array 2: n");

get\_value(array2);

printf("\n\n Elements of Array 2: ");

print\_value(array2, SIZE);

//Sort array 2

function\_sort(array2);

printf("\n\nSorted elements of Array 2: ");

print\_value(array2, SIZE);

//Find Intersection

num\_elements = find\_intersection(array1, array2, intersection\_array);

printf("\n\n Intersection is: ");

print\_value(intersection\_array, num\_elements);

//Find Union

num\_elements = find\_union(array1, array2, union\_array);

printf("\n\n Union is: ");

print\_value(union\_array, num\_elements);

}

void get\_value(int arr[])

{

int i, j;

for (i = 0; i < SIZE; i++)

{

j = i + 1;

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

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

}

}

void print\_value(int arr[], int n)

{

int i;

printf("{ ");

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

{

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

}

printf("}");

}

void function\_sort(int arr[])

{

int i, j, temp, swapping;

for (i = 1; i < size; i++)

{

swapping = 0;

for (j = 0; j < size-i; j++)

{

if (arr[j] > arr[j+1])

{

temp = arr[j];

arr[j] = arr[j + 1];

arr[j + 1] = temp;

swapping = 1;

}

}

if (swapping == 0)

{

break;

}

}

}

int find\_intersection(int array1[], int array2[], int intersection\_array[])

{

int i = 0, j = 0, k = 0;

while ((i < size) && (j < size))

{

if (array1[i] < array2[j])

{

i++;

}

else if (array1[i] > array2[j])

{

j++;

}

else

{

intersection\_array[k] = array1[i];

i++;

j++;

k++;

}

}

return(k);

}

int find\_union(int array1[], int array2[], int union\_array[])

{

int i = 0, j = 0, k = 0;

while ((i < SIZE) && (j < SIZE))

{

if (array1[i] < array2[j])

{

union\_array[k] = array1[i];

i++;

k++;

}

else if (array1[i] > array2[j])

{

union\_array[k] = array2[j];

j++;

k++;

}

else

{

union\_array[k] = array1[i];

i++;

j++;

k++;

}

}

if (i == SIZE)

{

while (j < SIZE)

{

union\_array[k] = array2[j];

j++;

k++;

}

}

else

{

while (i < SIZE)

{

union\_array[k] = array1[i];

i++;

k++;

}

}

return(k);

}

$ cc pgm98.c

$ a.out

Enter the elements of Array 1:

Enter element 1: 12

Enter element 2: 34

Enter element 3: 23

Enter element 4: 56

Enter element 5: 45

Elements of Array 1: { 12 34 23 56 45 }

Sorted elements of Array 1: { 12 23 34 45 56 }

Enter the elements of Array 2:

Enter element 1: 34

Enter element 2: 56

Enter element 3: 12

Enter element 4: 78

Enter element 5: 66

Elements of Array 2: { 34 56 12 78 66 }

Sorted elements of Array 2: { 12 34 56 66 78 }

Intersection is: { 12 34 56 }

Union is: { 12 23 34 45 56 66 78 }

1. Split the Array and Add First Part to the End

Enter the value of n

4

enter the numbers

3

678

345

876

Enter the position of the element to split the array

3

The resultant array is

876

3

678

345

// CPP program to split array and move first

// part to end.

#include <stdio.h>

void splitArr(int arr[], int n, int k)

{

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

// Rotate array by 1.

int x = arr[0];

for (int j = 0; j < n - 1; ++j)

arr[j] = arr[j + 1];

arr[n - 1] = x;

}

}

// Driver code

int main()

{

int arr[] = { 12, 10, 5, 6, 52, 36 };

int n = sizeof(arr) / sizeof(arr[0]);

int position = 2;

splitArr(arr, n, position);

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

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

return 0;

}

1. Matrix Multiplication

**Example 1:**

[1 4

3 2] \*

[1 2

2 1]

= [1∗1+3∗24∗1+2∗21∗2+3∗14∗2+2∗1] = [78510]

1. #include<stdio.h>
2. #include<stdlib.h>
3. **int** main(){
4. **int** a[10][10],b[10][10],mul[10][10],r,c,i,j,k;
5. system("cls");
6. printf("enter the number of row=");
7. scanf("%d",&r);
8. printf("enter the number of column=");
9. scanf("%d",&c);
10. printf("enter the first matrix element=\n");
11. **for**(i=0;i<r;i++)
12. {
13. **for**(j=0;j<c;j++)
14. {
15. scanf("%d",&a[i][j]);
16. }
17. }
18. printf("enter the second matrix element=\n");
19. **for**(i=0;i<r;i++)
20. {
21. **for**(j=0;j<c;j++)
22. {
23. scanf("%d",&b[i][j]);
24. }
25. }
27. printf("multiply of the matrix=\n");
28. **for**(i=0;i<r;i++)
29. {
30. **for**(j=0;j<c;j++)
31. {
32. mul[i][j]=0;
33. **for**(k=0;k<c;k++)
34. {
35. mul[i][j]+=a[i][k]\*b[k][j];
36. }
37. }
38. }
39. //for printing result
40. **for**(i=0;i<r;i++)
41. {
42. **for**(j=0;j<c;j++)
43. {
44. printf("%d\t",mul[i][j]);
45. }
46. printf("\n");
47. }
48. **return** 0;
49. }

**Output:**

enter the number of row=3

enter the number of column=3

enter the first matrix element=

1 1 1

2 2 2

3 3 3

enter the second matrix element=

1 1 1

2 2 2

3 3 3

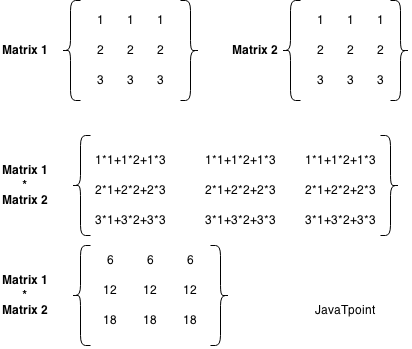
multiply of the matrix=

6 6 6

12 12 12

18 18 18

Let's try to understand the matrix multiplication of **3\*3 and 3\*3** matrices by the figure given below:



1. Find the Perimeter of a Circle, Rectangle and Triangle

perimeter of rectangle: 2 \* (a + b)

perimeter of General triangle: a + b + c

perimeter of Equilateral triangle: 3 \* a

perimeter of Right angled triangle: width + height + sqrt(width ^ 2 + height ^ 2)

perimeter of circle: 2 \* pi \* r

/\*

\* C Program to Find the Perimeter of a Circle, Rectangle and Triangle

\*/

#include <stdio.h>

#include <math.h>

int main()

{

float radius, length, width, a, b, c, height;

int n;

float perimeter;

//Perimeter of rectangle

printf(" \n Perimeter of rectangle \n");

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

printf("\n Enter width and length of the rectangle : ");

scanf("%f%f", &width,& length);

perimeter = 2 \* (width + length);

printf("Perimeter of rectangle is: %.3f", perimeter);

//Perimeter of triangle

printf("\n Perimeter of triangle n");

printf("---------------------------n");

printf("\n Enter the size of all sides of the triangle : ");

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

perimeter = a + b + c;

printf("Perimeter of triangle is: %.3f", perimeter);

//Perimeter of circle

printf(" \n Perimeter of circle \n");

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

printf("\n Enter the radius of the circle : ");

scanf("%f", &radius);

perimeter = 2 \* (22 / 7) \* radius;

printf("Perimeter of circle is: %.3f", perimeter);

//Perimeter of equilateral triangle

printf(" \n Perimeter of equilateral triangle \n");

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

printf("\n Enter any side of the equilateral triangle : ");

scanf("%f", &a);

perimeter = 3 \* a;

printf("Perimeter of equilateral triangle is: %.3f", perimeter);

//Perimeter of right angled triangle

printf(" \n Perimeter of right angled triangle \n");

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

printf("\n Enter the width and height of the right angled triangle : ");

scanf("%f%f", &width, &height);

perimeter = width + height + sqrt(width \* width + height \* height);

printf("Perimeter of right angled triangle is: %.3f", perimeter);

return 0;

}