**C Programs**

**Basic Programs in C**

1. Hello World to print in C

#include <stdio.h>

int main()

{

printf("Hello Welcome to Itishmita World");

return 0;

}

1. Addition of two numbers in C

#include <stdio.h>

int main()

{

int sum,a=1,b=2;

sum=a+b;

printf("Sum of %d and %d is %d",a,b,sum);

return 0;

}

1. Swapping number with the help of third number in C

#include <stdio.h>

int main()

{

int temp,a=1,b=2;

printf(" Before Swapping %d and %d ",a,b);

temp=a;

a=b;

b=temp;

printf(" After Swapping %d and %d ",a,b);

return 0;

}

1. Swapping of two numbers without the help of third number

#include <stdio.h>

int main()

{

int a=1,b=2;

printf(" Before Swapping %d and %d ",a,b);

a=a+b;

b=a-b;

a=a-b;

printf(" After Swapping %d and %d ",a,b);

return 0;

}

or

#include <stdio.h>

int main()

{

int a=1,b=2;

printf(" Before Swapping %d and %d ",a,b);

a=a\*b;

b=a/b;

a=a/b;

printf(" After Swapping %d and %d ",a,b);

return 0;

}

1. To check whether the string contains how many alphabets and digits

#include<stdio.h>

#include<stdlib.h>

int main()

{

char str[] = "12abc12";

int alphabet = 0, number = 0, i;

for (i=0; str[i]!= '\0'; i++)

{

// check for alphabets

if (isalpha(str[i]) != 0)

alphabet++;

// check for decimal digits

else if (isdigit(str[i]) != 0)

number++;

}

printf("Alphabetic\_letters = %d ",alphabet);

printf("Number\_digits = %d ",number);

return 0;

}

\*In C programming, isalpha() function checks whether a character is an alphabet (a to z and A-Z) or not. If it is a character it returns non zero integer. If it is not a character it will return 0

\*In C programming, isdigit(c) is a function which can be used to check if the passed character is a digit or not. It returns a non-zero value if it’s a digit else it returns 0.

1. Check whether the number is prime or not

#include <stdio.h>

int main() {

int n, i, fact = 0;

printf("Enter a positive integer: ");

scanf("%d", &n);

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

// condition for non-prime

if (n % i == 0) {

fact = 1;

break;

}

}

if (n == 1) {

printf("1 is neither prime nor composite.");

}

else {

if (fact == 0)

printf("%d is a prime number.", n);

else

printf("%d is not a prime number.", n);

}

return 0;

}

1. Dividing the number and getting quotient and remainder

#include <stdio.h>

int main() {

int dividend, divisor, quotient, remainder;

printf("Enter dividend: ");

scanf("%d", &dividend);

printf("Enter divisor: ");

scanf("%d", &divisor);

// Computes quotient

quotient = dividend / divisor;

// Computes remainder

remainder = dividend % divisor;

printf("Quotient = %d\n", quotient);

printf("Remainder = %d", remainder);

return 0;

}

1. Program to calculate reverse of a number

#include<stdio.h>

int main()

{

int n, reverse=0, rem;

printf("Enter a number: ");

scanf("%d", &n);

while(n!=0)

{

rem=n%10;

reverse=(reverse\*10)+rem;

n/=10;

}

printf("Reversed Number: %d",reverse);

return 0;

}

1. Number is Palindrome or not

#include<stdio.h>

int main()

{

int n,num, reverse=0, rem;

printf("Enter a number: ");

scanf("%d", &n);

num=n;

while(n!=0)

{

rem=n%10;

reverse=(reverse\*10)+rem;

n/=10;

}

if(num==reverse)

printf("Number %d is Palindrome",num);

else

printf("Number %d is not a Palindrome",num);

return 0;

}

1. To find whether the number is Armstrong or not

\*Armstrong number is a number that is equal to the sum of cubes of its digits. For example 0, 1, 153, 370, 371 and 407 are the Armstrong numbers.

#include<stdio.h>

int main()

{

int n,rem,sum=0,temp;

printf("enter the number=");

scanf("%d",&n);

temp=n;

while(n>0)

{

rem=n%10;

sum=sum+(rem\*rem\*rem);

n=n/10;

}

if(temp==sum)

printf("armstrong number ");

else

printf("not armstrong number");

return 0;

}

1. Calculate LCM of two numbers

#include <stdio.h>

int main()

{

int a, b, lcm;

printf("Enter two numbers: ");

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

lcm = (a > b) ? a : b;

while(1)

{

if( lcm % a == 0 && lcm % b == 0 )

{

printf("LCM of %d and %d is %d", a, b,lcm);

break;

}

++lcm;

}

return 0;

}

1. To calculate the HCF or GCD of two numbers

#include<stdio.h>

#include<stdlib.h>

int main()

{

int a,b,gcd;

printf("\nEnter two numbers : ");

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

int i;

for(i = 1; i <= a && i <= b; i++)

{

if((a % i == 0) && (b % i == 0))

{

gcd = i;

}

}

printf("\nGCD of %d and %d is %d ",a,b,gcd);

return 0;

}

1. Largest of three numbers

#include<stdio.h>

#include<stdlib.h>

int main()

{

int a=1,b=2,c=3;

if(a>b && a>c)

{

printf("Largest number is %d",a);

}

else if(b>a && b>c)

{

printf("Largest number is %d",b);

}

else

{

printf("Largest number is %d",c);

}

return 0;

}

1. Count the number of digits

#include <stdio.h>

int main()

{

int n;

int count = 0;

printf("\nEnter the number: ");

scanf("%d", &n);

while(n != 0)

{

n = n/10;

++count;

}

printf("\nNumber of digits: %d\n", count);

}

1. Sum of Digits of a number

#include <stdio.h>

int main()

{

int n;

int sum=0,rem = 0;

printf("\nEnter the number: ");

scanf("%d", &n);

while(n != 0)

{

rem=n%10;

sum=sum+rem;

n = n/10;

}

printf("\nSum of digits: %d\n", sum);

}

1. Sum of N natural number

#include <stdio.h>

int sum\_of\_natural\_numbers(int n)

{

int sum = 0;

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

{

sum += i;

}

return sum;

}

int main()

{

int n;

printf("\nEnter the number : ");

scanf("%d", &n);

printf("\nSum of %d Natural Numbers is %d\n ",n,sum\_of\_natural\_numbers(n));

return 0;

}

1. Factorial of the number

#include <stdio.h>

int factorial\_of\_a\_number(int n)

{

int fact = 1, i;

if(n == 0)

return 1;

else

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

{

fact = fact \* i;

}

return fact;

}

int main()

{

int n;

printf("Enter the number : ");

scanf("%d",&n);

if(n < 0)

printf("Invalid output");

else

printf("Factorial of the number %d is %d" ,n, factorial\_of\_a\_number(n));

return 0;

}

1. Leap Year

#include <stdio.h>

int main() {

int year;

year = 2018;

if (((year % 4 == 0) && (year % 100!= 0)) || (year%400 == 0))

printf("%d is a leap year", year);

else

printf("%d is not a leap year", year);

return 0;

}

1. Fibonacci Series up to n

#include<stdio.h>

int main()

{

int sum = 0, n;

int a = 0;

int b = 1;

printf("Enter the nth value: ");

scanf("%d", &n);

printf("Fibonacci series: ");

while(sum <= n)

{

printf("%d ", sum);

a = b; // swap elements

b = sum;

sum = a + b; // next term is the sum of the last two terms

}

return 0;

}

1. Strong number

\*A strong number is a number in which the **sum of the factorial of the digits is equal to the number itself**.

#include<stdio.h>

int main()

{

int n,i;

int fact,rem;

printf("\nEnter a number : ");

scanf("%d",&n);

printf(“\n”);

int sum = 0;

int temp = n;

while(n)

{

i = 1,fact = 1;

rem = n % 10;

while(i <= rem)

{

fact = fact \* i;

i++;

}

sum = sum + fact;

n = n / 10;

}

if(sum == temp)

printf("%d is a strong number\n",temp);

else

printf("%d is not a strong number\n",temp);

return 0;

}

1. Perfect number

\*A perfect number is a number which is equal to the sum of its proper positive divisors.

#include<stdio.h>

int main()

{

int i, num, sum = 0;

printf("\nEnter a number : ");

scanf("%d", &num);

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

{

if(num%i == 0)

{

sum += i;

}

}

if(sum == num)

{

printf("\n%d is a Perfect Number\n", num);

}

else

{

printf("\n%d is not a Perfect Number\n", num);

}

return 0;

}

1. Friendly pair or not

#include<stdio.h>

int main()

{

int num1, num2;

scanf("%d",&num1);

scanf("%d",&num2);

int sum1 = 0, sum2 = 0;

for(int i = 1; i < num1; i++)

{

if(num1 % i == 0)

{

sum1 = sum1 + i;

}

}

for(int i = 1; i < num2; i++)

{

if(num2 % i == 0)

{

sum2 = sum2 + i;

}

}

if(sum1 == num1 && sum2 == num2)

printf("Friendly Pair");

else

printf("Not Friendly Pair");

return 0;

}

1. Power of a number

#include <stdio.h>

int main() {

int base, exponent;

long long result = 1;

printf("Enter a base number: ");

scanf("%d", &base);

printf("Enter an exponent: ");

scanf("%d", &exponent);

while (exponent != 0) {

result = result\*base;

--exponent;

}

printf("Answer = %lld", result);

return 0;

}

or

**Using Power**

#include <math.h>

#include <stdio.h>

int main() {

double base, exponent, result;

printf("Enter a base number: ");

scanf("%lf", &base);

printf("Enter an exponent: ");

scanf("%lf", &exponent);

// calculates the power

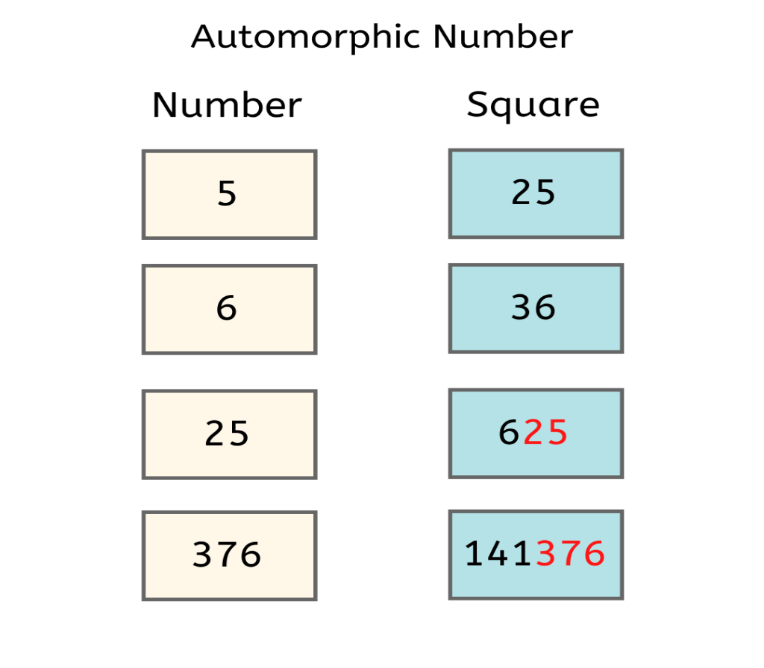
result = pow(base, exponent);

printf("%.1lf^%.1lf = %.2lf", base, exp, result);

return 0;

}

1. Automorphic Number or not



#include<stdio.h>

int checkAutomorphic(int num){

int square = num \* num;

while (num > 0) {

if (num % 10 != square % 10)

return 0;

// Reduce N and square

num = num / 10;

square = square / 10;

}

return 1;

}

int main(){

//enter value

int num;

scanf("%d",&num);

//checking condition

if(checkAutomorphic(num))

printf("Automorphic");

else

printf("Not Automorphic");

return 0;

}

1. Harshad Number or not

\*Harshad Number is an integer that is divisible by the sum of its digits.

#include<stdio.h>

int main()

{

//fill the code

int num;

int temp,rem;

scanf("%d",&num);

int sum = 0;

temp = num;

while(temp>0)

{

rem = temp % 10;

sum = sum + rem;

temp = temp / 10;

}

int res = num % sum;

if(res == 0)

printf("Harshad Number");

else

printf("Not Harshad Number");

return 0;

}

1. Abundant Number or not

\*An abundant number is a number for which the sum of its proper divisors is greater than the number itself.

#include<stdio.h>

int main()

{

//fill the code

int num;

int temp;

scanf("%d",&num);

int sum = 0;

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

{

if(num % i == 0)

{

sum = sum + i;

}

}

if(num < sum)

printf("Abundant Number");

else

printf("Not Abundant Number");

return 0;

}

**Medium Programs**

1. Prime Numbers between given range

#include <stdio.h>

int main()

{

int num1, num2, factor, i, j;

/\* Ask user to input the from/to range

\* like 1 to 100, 10 to 1000 etc.

\*/

printf("Enter two range(input integer numbers only):");

//Store the range in variables using scanf

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

//Display prime numbers for input range

printf("Prime numbers from %d and %d are:\n", num1, num2);

for(i=num1+1; i<num2; ++i)

{

factor=0;

for(j=2; j<=i/2; ++j)

{

if(i%j==0)

{

factor=1;

break;

}

}

if(factor==0)

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

}

return 0;

}

1. Armstrong numbers between range

#include <math.h>

#include <stdio.h>

int main() {

int low, high, number, originalNumber, rem, count = 0;

double result = 0.0;

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

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

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

// iterate number from (low + 1) to (high - 1)

// In each iteration, check if number is Armstrong

for (number = low + 1; number < high; ++number) {

originalNumber = number;

// number of digits calculation

while (originalNumber != 0) {

originalNumber /= 10;

++count;

}

originalNumber = number;

// result contains sum of nth power of individual digits

while (originalNumber != 0) {

rem = originalNumber % 10;

result += pow(rem, count);

originalNumber /= 10;

}

// check if number is equal to the sum of nth power of individual digits

if ((int)result == number) {

printf("%d ", number);

}

// resetting the values

count = 0;

result = 0;

}

return 0;

}

1. Program to replace all 0's with 1 in a given integer

#include<stdio.h>

int replace(long int number)

{

// Base case for recursion termination

if (number == 0)

return 0;

// Extract the last digit and change it if needed

int digit = number % 10;

if (digit == 0)

digit = 1;

// Convert remaining digits and append the last digit

return replace(number/10) \* 10 + digit;

}

int Convert(long int number)

{

if (number == 0)

return 1;

else

return replace(number);

}

int main()

{

long int number;

printf("\nEnter the number : ");

scanf("%d", &number);

printf("\nNumber after replacement : %d", Convert(number));

return 0;

}

1. Convert given binary number into decimal number

Consider the binary number from the last.

For the above mentioned example,

1 \* 2^0 = 1

1 \* 2^1 = 2

1 \* 2^2 = 4

1 \* 2^3 = 6

Decimal number = 1 + 2 + 4 + 6 = 15

1111 in binary form is represented as 15 in decimal.

#include<stdio.h>

#include<math.h>

int binary\_to\_decimal(long int n)

{

int decimal = 0, i = 0, remainder;

while (n!=0)

{

remainder = n%10;

n /= 10;

decimal += remainder\*pow(2,i);

++i;

}

return decimal;

}

int main()

{

long int n;

printf("Enter a binary number: ");

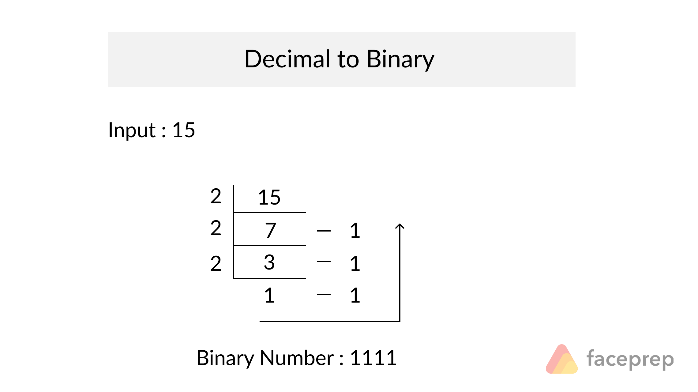
scanf("%ld", &n);

printf("\nDecimal number : %d\n ", binary\_to\_decimal(n));

return 0;

}

1. Convert Decimal to binary



#include <stdio.h>

long int decimal\_to\_binary(int n)

{

long int binary = 0;

int remainder, i, flag = 1;

for(i = 1; n != 0; i = i \* 10)

{

remainder = n % 2;

n /= 2;

binary += remainder \* i;

}

return binary;

}

int main()

{

int n;

printf("Enter a decimal number: ");

scanf("%d", &n);

printf("Equivalent binary number: %d\n", decimal\_to\_binary(n));

return 0;

}

1. Program to convert decimal to octal

#include <stdio.h>

#include <math.h>

int decimal\_to\_octal(int decimal);

int main()

{

int decimal;

printf("\nEnter a decimal number: ");

scanf("%d", &decimal);

printf("\nEquivalent octal number : %d\n", decimal\_to\_octal(decimal));

return 0;

}

int decimal\_to\_octal(int decimal)

{

int octal = 0, i = 1;

while (decimal != 0)

{

octal += (decimal % 8) \* i;

decimal /= 8;

i \*= 10;

}

return octal;

}

1. Program to convert octal to decimal

#include <stdio.h>

#include <math.h>

long int octal\_to\_decimal(int octal)

{

int decimal = 0, i = 0;

while(octal != 0)

{

decimal += (octal%10) \* pow(8,i); // multiplying with powers of 8

++i;

octal/=10; // Divide by 10 to make it as decimal

}

i = 1;

return decimal;

}

int main()

{

int octal;

printf("\nEnter an octal number: ");

scanf("%d", &octal);

printf("\nDecimal Equivalent : %d\n",octal\_to\_decimal(octal));

return 0;}

1. Program to convert binary to octal

#include <stdio.h>

#include <math.h>

int binary\_to\_octal(long int binary)

{

int octal = 0, decimal = 0, i = 0;

while(binary != 0)

{

decimal += (binary%10) \* pow(2,i);

++i;

binary/=10;

}

i = 1;

while (decimal != 0)

{

octal += (decimal % 8) \* i;

decimal /= 8;

i \*= 10;

}

return octal;

}

int main()

{

long int binary;

printf("\nEnter a binary number: ");

scanf("%lld", &binary);

printf("\nOctal Equivalent : %d\n", binary\_to\_octal(binary));

return 0;

}

1. Program to convert octal to binary

#include <stdio.h>

#include <math.h>

long long octal\_to\_binary(int octal)

{

int decimal = 0, i = 0;

long long binary = 0;

while(octal != 0)

{

decimal += (octal%10) \* pow(8,i);

++i;

octal/=10;

}

i = 1;

while (decimal != 0)

{

binary += (decimal % 2) \* i;

decimal /= 2;

i \*= 10;

}

return binary;

}

int main()

{

int octal;

printf("\nEnter an octal number: ");

scanf("%d", &octal);

printf("\nBinary Equivalent : %d\n", octal\_to\_binary(octal));

return 0;

}

1. Find maximum number of hanshakes

#include<stdio.h>

int main()

{

//fill the code

int num;

scanf("%d",&num);

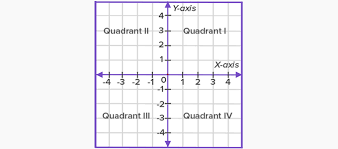
int total = num \* (num-1) / 2; // Combination nC2

printf("%d",total);

return 0;

}

1. Program to find the quadrant in which the given coordinates lie



#include <stdio.h>

int main()

{

//Fill the code

int a,b;

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

if(a > 0 && b > 0)

printf("Ist Quadrant");

else if(a < 0 && b > 0)

printf("IInd Quadrant");

else if(a < 0 && b < 0)

printf("IIIrd Quadrant");

else if(a > 0 && b < 0)

printf("IVth Quadrant");

else

printf("Origin");

return 0;

}

1. Number of days in a given month of a given year

#include<stdio.h>

int main()

{

//fill the code

int year, month;

scanf("%d %d",&month,&year);

if(month == 1 || month == 3 || month == 5 || month == 7 || month == 8 || month == 10 || month == 12)

printf("Number of days is 31");

else if((month == 2) && ((year%400==0) || (year%4==0 && year%100!=0)))

{

printf("Number of days is 29");

}

else if(month == 2)

{

printf("Number of days is 28");

}

else

printf("Number of days is 30");

return 0;

}

1. Program to find all possible permutations in which n people can occupy r seats in a theater

// C program to find all possible permutations in which n people can occupy r seats in a theater

#include<stdio.h>

// Function to find the factorial of the number

int fact(long int x)

{

long int f=1,i;

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

{

f=f\*i;

}

return f;

}

int main()

{

long int n,r,p,temp;

long int num,den;

// Enter the number of seats

printf("Enter the number of seats available : ");

scanf("%ld",&r);

// Enter the number of people

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

scanf("%ld",&n);

// Base condition

// Swap n and r

if(n < r)

{

temp=n;

n=r;

r=temp;

}

num=fact(n);

den=fact(n-r);

p=num/den;

printf("nNumber of ways people can be seated : ");

printf("%ld",p);

}

1. Program to find the number of times digit 3 occurs in each and every number from 0 to n

// C program to find the number of times digit 3 occurs in each and every number from 0 to n

#include <stdio.h>

int count\_3s(int n)

{

int count = 0;

while (n > 0)

{

if (n % 10 == 3)

{

count++;

}

n = n / 10;

}

return count;

}

int count\_in\_range(int n)

{

int count = 0 ;

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

{

count += count\_3s(i);

}

return count;

}

int main()

{

int n;

printf("\nEnter the end value : ");

scanf("%d", &n);

printf("\nTotal occurrences of 3 from 0 to %d is %d\n", n,count\_in\_range(n));

return 0;

}

1. Number of integers with exactly 9 divisors

#include <stdio.h>

int count\_no\_of\_divisors(int num)

{

int count = 0;

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

{

if (num % i == 0)

count = count + 1;

}

return count;

}

void check\_9\_factors(int n)

{

int c = 0;

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

{

if (count\_no\_of\_divisors(i) == 9)

{

printf("%d ", i);

c = c + 1;

}

}

printf("\n\nTotal = %d\n", c);

}

int main()

{

int n;

printf("\nEnter the number : ");

scanf("%d", &n);

printf("\nThe number which has exactly 9 divisors : ");

check\_9\_factors(n);

return 0;

}

1. Program to find out roots of a quadratic equation

#include <stdio.h>

#include <math.h>

int main()

{

double a, b, c, discriminant, root1, root2, realPart, imaginaryPart;

printf("Enter coefficients a, b and c: ");

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

discriminant = b\*b-4\*a\*c;

// condition for real and different roots

if (discriminant > 0)

{

// sqrt() function returns square root

root1 = (-b+sqrt(discriminant))/(2\*a);

root2 = (-b-sqrt(discriminant))/(2\*a);

printf("root1 = %.2lf and root2 = %.2lf",root1 , root2);

}

//condition for real and equal roots

else if (discriminant == 0)

{

root1 = root2 = -b/(2\*a);

printf("root1 = root2 = %.2lf;", root1);

}

// if roots are not real

else

{

realPart = -b/(2\*a);

imaginaryPart = sqrt(-discriminant)/(2\*a);

printf("root1 = %.2lf+%.2lfi and root2 = %.2f-%.2fi", realPart, imaginaryPart, realPart, imaginaryPart);

}

return 0;

}