

**DRONACHARYA GROUP OF INSTITUTIONS, GREATER NOIDA**

**Affiliated to Mahamaya Technical University, Noida**

**Approved by AICTE**



**DEPARTMENT OF APPLIED SCIENCES & HUMANITIES**

**Lab Manual for**

**COMPUTER PROGRAMMING LAB CS201**

## List of C Programs

1) **C hello world program** :- c programming language code to print hello world. This program prints hello world, printf function is used to display text on screen, '\n' places cursor on the beginning of next line, stdio.h header file contains declaration of printf function. The code will work on all operating systems may be it's Linux, Mac or any other and compilers. To learn a programming language you must start writing programs in it and may be your first c code while learning programming.

### **C hello world example**

```
#include <stdio.h>
int main()
{
    printf("Hello world\n");
    return 0;
}
```

### **Hello world program in c**

We may store "hello world" in a character array and then print it.

```
#include <stdio.h>

int main()
{
    char string[] = "Hello World";

    printf("%s\n", string);

    return 0;
}
```

Hello world program.

Output of program:

```
Hello World
```

## 2) **C program print integer**

This c program first inputs an integer and then prints it. Input is done using scanf function and number is printed on screen using printf.

### **C programming code**

```
#include <stdio.h>

int main()
{
    int a;

    printf("Enter an integer\n");
    scanf("%d", &a);

    printf("Integer that you have entered is %d\n", a);

    return 0;
}
```

Output of program:

```
Enter an integer
45
Integer that you have entered is 45
```

### 3) C program to add two numbers

C program to add two numbers: This c language program perform the basic arithmetic operation of addition on two numbers and then prints the sum on the screen. For example if the user entered two numbers as 5, 6 then 11 ( 5 + 6 ) will be printed on the screen.

#### C programming code

```
#include<stdio.h>

main()
{
    int a, b, c;

    printf("Enter two numbers to add\n");
    scanf("%d%d",&a,&b);

    c = a + b;

    printf("Sum of entered numbers = %d\n",c);

    return 0;
}
```

Add numbers program executable.

Output of program

```
Enter two numbers to add
4
5
Sum of entered numbers = 9
```

#### Addition without using third variable

```
#include<stdio.h>

main()
{
    int a = 1, b = 2;

    /* Storing result of addition in variable a */

    a = a + b;

    /* Not recommended because original value of a is lost
    * and you may be using it some where in code considering it
    * as it was entered by the user.
    */

    printf("Sum of a and b = %d\n", a);

    return 0;
}
```

#### C program to add two numbers repeatedly

```
#include<stdio.h>
```

```

main()
{
    int a, b, c;
    char ch;

    while(1)
    {
        printf("Enter values of a and b\n");
        scanf("%d%d",&a,&b);

        c = a + b;

        printf("a + b = %d\n", c);

        printf("Do you wish to add more numbers(y/n)\n");
        scanf("%c",&ch);

        if ( ch == 'y' || ch == 'Y' )
            continue;
        else
            break;
    }

    return 0;
}

```

#### Adding numbers in c using function

We have used long data type as it can handle large numbers.

```

#include<stdio.h>

long addition(long, long);

main()
{
    long first, second, sum;

    scanf("%ld%ld", &first, &second);

    sum = addition(first, second);

    printf("%ld\n", sum);

    return 0;
}

long addition(long a, long b)
{
    long result;

    result = a + b;

    return result;
}

```

#### 4) C program to check odd or even

c program to check odd or even: We will determine whether a number is odd or even by using different methods all are provided with a code in c language. As you have study in mathematics that in decimal number system even numbers are divisible by 2 while odd are not so we may use modulus operator(%) which returns remainder, For example  $4\%3$  gives 1 ( remainder when four is divided by three). Even numbers are of the form  $2*p$  and odd are of the form  $(2*p+1)$  where p is an integer.

We can use bitwise AND (&) operator to check odd or even, as an example consider binary of 7 (0111) when we perform  $7 \& 1$  the result will be one and you may observe that the least significant bit of every odd number is 1, so  $(\text{odd\_number} \& 1)$  will be one always and also  $(\text{even\_number} \& 1)$  is zero.

In c programming language when we divide two integers we get an integer result, For example the result of  $7/3$  will be 2. So we can take advantage of this and may use it to find whether the number is odd or even. Consider an integer  $n$  we can first divide by 2 and then multiply it by 2 if the result is the original number then the number is even otherwise the number is odd. For example  $11/2 = 5$ ,  $5*2 = 10$  ( which is not equal to eleven), now consider  $12/2 = 6$  and  $6*2 = 12$  ( same as original number). These are some logic which may help you in finding if a number is odd or not.

### C program to check odd or even using modulus operator

```
#include<stdio.h>

main()
{
    int n;

    printf("Enter an integer\n");
    scanf("%d",&n);

    if ( n%2 == 0 )
        printf("Even\n");
    else
        printf("Odd\n");

    return 0;
}
```

### C program to check odd or even using bitwise operator

```
#include<stdio.h>

main()
{
    int n;

    printf("Enter an integer\n");
    scanf("%d",&n);

    if ( n & 1 == 1 )
        printf("Odd\n");
    else
        printf("Even\n");

    return 0;
}
```

### C program to check odd or even without using bitwise or modulus operator

```
#include<stdio.h>

main()
{
    int n;

    printf("Enter an integer\n");
    scanf("%d",&n);

    if ( (n/2)*2 == n )
        printf("Even\n");
    else
        printf("Odd\n");

    return 0;
}
```

### Find odd or even using conditional operator

```
#include<stdio.h>

main()
{
    int n;

    printf("Enter an integer\n");
    scanf("%d",&n);

    n%2 == 0 ? printf("Even number\n") : printf("Odd number\n");

    return 0;
}
```

### 5) C program to check odd or even

C program to check odd or even: We will determine whether a number is odd or even by using different methods all are provided with a code in c language. As you have study in mathematics that in decimal number system even numbers are divisible by 2 while odd are not so we may use modulus operator(%) which returns remainder, For example  $4\%3$  gives 1 ( remainder when four is divided by three). Even numbers are of the form  $2*p$  and odd are of the form  $(2*p+1)$  where p is an integer. We can use bitwise AND (&) operator to check odd or even, as an example consider binary of 7 (0111) when we perform  $7 \& 1$  the result will be one and you may observe that the least significant bit of every odd number is 1, so  $(\text{odd\_number} \& 1)$  will be one always and also  $(\text{even\_number} \& 1)$  is zero.

In c programming language when we divide two integers we get an integer result, For example the result of  $7/3$  will be 2. So we can take advantage of this and may use it to find whether the number is odd or even. Consider an integer n we can first divide by 2 and then multiply it by 2 if the result is the original number then the number is even otherwise the number is odd. For example  $11/2 = 5$ ,  $5*2 = 10$  ( which is not equal to eleven), now consider  $12/2 = 6$  and  $6*2 = 12$  ( same as original number). These are some logic which may help you in finding if a number is odd or not.

### C program to check odd or even using modulus operator

```
#include<stdio.h>

main()
{
    int n;

    printf("Enter an integer\n");
    scanf("%d",&n);

    if ( n%2 == 0 )
        printf("Even\n");
    else
        printf("Odd\n");

    return 0;
}
```

### C program to check odd or even using bitwise operator

```
#include<stdio.h>

main()
{
    int n;

    printf("Enter an integer\n");
    scanf("%d",&n);
```

```
if ( n & 1 == 1 )
    printf("Odd\n");
else
    printf("Even\n");

return 0;
}
```

### **C program to check odd or even without using bitwise or modulus operator**

```
#include<stdio.h>

main()
{
    int n;

    printf("Enter an integer\n");
    scanf("%d",&n);

    if ( (n/2)*2 == n )
        printf("Even\n");
    else
        printf("Odd\n");

    return 0;
}
```

### **Find odd or even using conditional operator**

```
#include<stdio.h>

main()
{
    int n;

    printf("Enter an integer\n");
    scanf("%d",&n);

    n%2 == 0 ? printf("Even number\n") : printf("Odd number\n");

    return 0;
}
```

## **6) C program to perform addition, subtraction, multiplication and division**

C program to perform basic arithmetic operations i.e. addition , subtraction, multiplication and division of two numbers. Numbers are assumed to be integers and will be entered by the user.

### **C programming code**

```
#include <stdio.h>

int main()
```

```

{
int first, second, add, subtract, multiply;
float divide;

printf("Enter two integers\n");
scanf("%d%d", &first, &second);

add = first + second;
subtract = first - second;
multiply = first * second;
divide = first / (float)second; //typecasting

printf("Sum = %d\n",add);
printf("Difference = %d\n",subtract);
printf("Multiplication = %d\n",multiply);
printf("Division = %.2f\n",divide);

return 0;
}

```

In c language when we divide two integers we get integer result for example  $5/2$  evaluates to 2. As a general rule integer/integer = integer and float/integer = float or integer/float = float. So we convert denominator to float in our program, you may also write float in numerator. This explicit conversion is known as typecasting. [Arithmetic operations.](#)

Output of program:

```

Enter two integers
5 3
Sum = 8
Difference = 2
Multiplication = 15
Division = 1.67

```

## 7) C program to check whether input alphabet is a vowel or not

This code checks whether an input alphabet is a vowel or not. Both lower-case and upper-case are checked.

### C programming code

```

#include <stdio.h>

main()
{
char ch;

printf("Enter a character\n");
scanf("%c", &ch);

if (ch == 'a' || ch == 'A' || ch == 'e' || ch == 'E' || ch == 'i' || ch == 'I' || ch == 'o' || ch == 'O' || ch == 'u' || ch == 'U')

printf("%c is a vowel.\n", ch);

else

printf("%c is not a vowel.\n", ch);

```

```
return 0;
}
```

Output:

```
Enter a character
a
a is a vowel.
```

### Check vowel using switch statement

```
#include <stdio.h>
```

```
main()
```

```
{
```

```
char ch;
```

```
printf("Enter a character\n");
```

```
scanf("%c", &ch);
```

```
switch(ch)
```

```
{
```

```
case 'a':
```

```
case 'A':
```

```
case 'e':
```

```
case 'E':
```

```
case 'i':
```

```
case 'I':
```

```
case 'o':
```

```
case 'O':
```

```
case 'u':
```

```
case 'U':
```

```
printf("%c is a vowel.\n", ch);
```

```
break;
```

```
default:
```

```
printf("%c is not a vowel.\n", ch);
```

```
}

return 0;

}
```

### Function to check vowel

```
int check_vowel(char a)
{
    if (a >= 'A' && a <= 'Z')
        a = a + 'a' - 'A'; /* Converting to lower case or use a = a + 32 */

    if (a == 'a' || a == 'e' || a == 'i' || a == 'o' || a == 'u')
        return 1;

    return 0;
}
```

### 8) c program to check leap year

c program to check leap year: c code to check leap year, year will be entered by the user. Please read the [leap year](#) article before reading the code, it will help you to understand the program.

### C programming code

```
#include <stdio.h>

int main()
{
    int year;

    printf("Enter a year to check if it is a leap year\n");
    scanf("%d", &year);

    if ( year%400 == 0)
        printf("%d is a leap year.\n", year);
    else if ( year%100 == 0)
        printf("%d is not a leap year.\n", year);
    else if ( year%4 == 0 )
        printf("%d is a leap year.\n", year);
    else
        printf("%d is not a leap year.\n", year);
}
```

```
return 0;
}
```

Compiler used:

GCC

Output of program:

```
Enter a year to check if it is a leap year
2012
2012 is a leap year.
Process returned 0 (0x0)   execution time : 1.727 s
Press ENTER to continue.
█
```

## 9) Add digits of number in c

C program to add digits of a number: Here we are using modulus operator(%) to extract individual digits of number and adding them.

### C programming code

```
#include <stdio.h>

main()
{
    int n, sum = 0, remainder;

    printf("Enter an integer\n");
    scanf("%d",&n);

    while(n != 0)
    {
        remainder = n % 10;
        sum = sum + remainder;
        n = n / 10;
    }

    printf("Sum of digits of entered number = %d\n",sum);

    return 0;
}
```

For example if the input is 98, sum(variable) is 0 initially

$98 \% 10 = 8$  (% is modulus operator which gives us remainder when 98 is divided by 10).

$sum = sum + remainder$

so  $sum = 8$  now.

$98 / 10 = 9$  because in c whenever we divide integer by another integer we get an integer.

$9 \% 10 = 9$

$sum = 8(\text{previous value}) + 9$

$sum = 17$

$9 / 10 = 0$ .

So finally  $n = 0$ , loop ends we get the required sum.

Output of program:

```
Enter a number
12358
Sum of digits of entered number = 19
```

### Add digits using recursion

```
#include <stdio.h>

int add_digits(int);

int main() {
    int n, result;

    scanf("%d", &n);

    result = add_digits(n);

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

    return 0;
}

int add_digits(int n) {
    static int sum = 0;

    if (n == 0) {
        return 0;
    }

    sum = n%10 + add_digits(n/10);

    return sum;
}
```

### 10) Factorial program in c

Factorial program in c: c code to find and print factorial of a number, three methods are given, first one uses a for loop, second uses a [function](#) to find factorial and third using [recursion](#). Factorial is represented using !, so five factorial will be written as 5!, n factorial as n!. Also  $n! = n*(n-1)*(n-2)*(n-3)...3.2.1$  and zero factorial is defined as one i.e.  $0!=1$ .

#### Factorial program in c using for loop

: Here we find factorial using for loop.

```
#include <stdio.h>

int main()
{
    int c, n, fact = 1;

    printf("Enter a number to calculate it's factorial\n");
    scanf("%d", &n);

    for (c = 1; c <= n; c++)
        fact = fact * c;

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

    return 0;
}
```

```
}
```

Factorial

Output of code:

```
C:\Dev-Cpp>factorial.exe
Enter a number to calculate it's factorial
6
Factorial of 6 = 720
```

### Factorial program in c using function

```
#include <stdio.h>

long factorial(int);

int main()
{
    int number;
    long fact = 1;

    printf("Enter a number to calculate it's factorial\n");
    scanf("%d", &number);

    printf("%d! = %ld\n", number, factorial(number));

    return 0;
}

long factorial(int n)
{
    int c;
    long result = 1;

    for (c = 1; c <= n; c++)
        result = result * c;

    return result;
}
```

### Factorial program in c using recursion

```
#include<stdio.h>

long factorial(int);

int main()
{
    int num;
    long f;

    printf("Enter a number to find factorial\n");
    scanf("%d", &num);

    if (num < 0)
        printf("Negative numbers are not allowed.\n");
    else
    {
        f = factorial(num);
        printf("%d! = %ld\n", num, f);
    }
}
```

```

return 0;
}

long factorial(int n)
{
if (n == 0)
return 1;
else
return(n * factorial(n-1));
}

```

### 11) C program to find hcf and lcm

C program to find hcf and lcm: The code below finds highest common factor and least common multiple of two integers. HCF is also known as greatest common divisor(GCD) or greatest common factor(gcf).

C programming code

```

#include <stdio.h>

int main() {
int a, b, x, y, t, gcd, lcm;

printf("Enter two integers\n");
scanf("%d%d", &x, &y);

a = x;
b = y;

while (b != 0) {
t = b;
b = a % b;
a = t;
}

gcd = a;
lcm = (x*y)/gcd;

printf("Greatest common divisor of %d and %d = %d\n", x, y, gcd);
printf("Least common multiple of %d and %d = %d\n", x, y, lcm);

return 0;
}

```

C program to find hcf and lcm using recursion

```

#include <stdio.h>

long gcd(long, long);

int main() {
long x, y, hcf, lcm;

printf("Enter two integers\n");
scanf("%ld%ld", &x, &y);

hcf = gcd(x, y);
lcm = (x*y)/hcf;

printf("Greatest common divisor of %ld and %ld = %ld\n", x, y, hcf);
printf("Least common multiple of %ld and %ld = %ld\n", x, y, lcm);
}

```

```
return 0;
}

long gcd(long a, long b) {
    if (b == 0) {
        return a;
    }
    else {
        return gcd(b, a % b);
    }
}
```

C program to find hcf and lcm using function

```
#include <stdio.h>

long gcd(long, long);

int main() {
    long x, y, hcf, lcm;

    printf("Enter two integers\n");
    scanf("%ld%ld", &x, &y);

    hcf = gcd(x, y);
    lcm = (x*y)/hcf;

    printf("Greatest common divisor of %ld and %ld = %ld\n", x, y, hcf);
    printf("Least common multiple of %ld and %ld = %ld\n", x, y, lcm);

    return 0;
}

long gcd(long x, long y) {
    if (x == 0) {
        return y;
    }

    while (y != 0) {
        if (x > y) {
            x = x - y;
        }
        else {
            y = y - x;
        }
    }

    return x;
}
```

## Output of program:

```
Enter two integers
9 15
Greatest common divisor of 9 and 15 = 3
Least common multiple of 9 and 15 = 45
Process returned 0 (0x0)   execution time : 17.105 s
Press ENTER to continue.
█
```

## 12) Decimal to binary conversion

C program to convert decimal to binary: c language code to convert an integer from decimal number system(base-10) to binary number system(base-2). Size of integer is assumed to be 32 bits. We use bitwise operators to perform the desired task. We right shift the original number by 31, 30, 29, ..., 1, 0 bits using a loop and bitwise AND the number obtained with 1(one), if the result is 1 then that bit is 1 otherwise it is 0(zero).

### C programming code

```
#include <stdio.h>

int main()
{
    int n, c, k;

    printf("Enter an integer in decimal number system\n");
    scanf("%d", &n);

    printf("%d in binary number system is:\n", n);

    for (c = 31; c >= 0; c--)
    {
        k = n >> c;

        if (k & 1)
            printf("1");
        else
            printf("0");
    }

    printf("\n");

    return 0;
}
```

Above code only prints binary of integer, but we may wish to perform operations on binary so in the code below we are storing the binary in a string. We create a function which returns a pointer to string which is the binary of the number passed as argument to the function.

### C code to store decimal to binary conversion in a string

```

#include <stdio.h>
#include <stdlib.h>

char *decimal_to_binary(int);

main()
{
    int n, c, k;
    char *pointer;

    printf("Enter an integer in decimal number system\n");
    scanf("%d",&n);

    pointer = decimal_to_binary(n);
    printf("Binary string of %d is: %s\n", n, t);

    free(pointer);

    return 0;
}

char *decimal_to_binary(int n)
{
    int c, d, count;
    char *pointer;

    count = 0;
    pointer = (char*)malloc(32+1);

    if ( pointer == NULL )
        exit(EXIT_FAILURE);

    for ( c = 31 ; c >= 0 ; c-- )
    {
        d = n >> c;

        if ( d & 1 )
            *(pointer+count) = 1 + '0';
        else
            *(pointer+count) = 0 + '0';

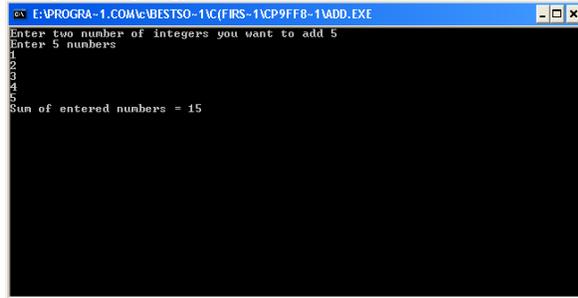
        count++;
    }
    *(pointer+count) = '\0';

    return pointer;
}

```



## Output of program:



```
E:\PROGRA-1.COM\BESTSO-1\C(FIRS-1\CP9FF8-1\ADD.EXE
Enter two number of integers you want to add 5
Enter 5 numbers
1
2
3
4
5
Sum of entered numbers = 15
```

## 14) C program to reverse a number

C Program to reverse a number :- This program reverse the number entered by the user and then prints the reversed number on the screen. For example if user enter 123 as input then 321 is printed as output. In our program we use modulus(%) operator to obtain the digits of a number. To invert number look at it and write it from opposite direction or the output of code is a number obtained by writing original number from right to left. To reverse large numbers use long data type or long long data type if your compiler supports it, if you still have large numbers then use strings or other data structure.

### C programming code

```
#include <stdio.h>

main()
{
    int n, reverse = 0;

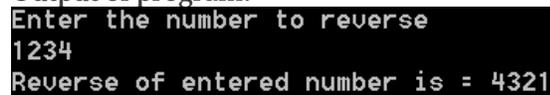
    printf("Enter a number to reverse\n");
    scanf("%d",&n);

    while (n != 0)
    {
        reverse = reverse * 10;
        reverse = reverse + n%10;
        n = n/10;
    }

    printf("Reverse of entered number is = %d\n", reverse);

    return 0;
}
```

## Output of program:



```
Enter the number to reverse
1234
Reverse of entered number is = 4321
```

## 15) Palindrome Numbers

Palindrome number in c: A palindrome number is a number such that if we reverse it, it will not change. For example some palindrome numbers examples are 121, 212, 12321, -454. To check whether a number is palindrome or not first we reverse it and then compare the number obtained with the original, if both are same then number is palindrome otherwise not. C program for palindrome number is given below.

### Palindrome number algorithm

1. Get the number from user.
2. Reverse it.
3. Compare it with the number entered by the user.

4. If both are same then print palindrome number
5. Else print not a palindrome number.

### Palindrome number program c

```
#include<stdio.h>

main()
{
    int n, reverse = 0, temp;

    printf("Enter a number to check if it is a palindrome or not\n");
    scanf("%d",&n);

    temp = n;

    while( temp != 0 )
    {
        reverse = reverse * 10;
        reverse = reverse + temp%10;
        temp = temp/10;
    }

    if ( n == reverse )
        printf("%d is a palindrome number.\n", n);
    else
        printf("%d is not a palindrome number.\n", n);

    return 0;
}
```

### 16) C program to print patterns of numbers and stars

These program prints various different patterns of numbers and stars. These codes illustrate how to create various patterns using c programming. Most of these c programs involve usage of nested loops and space. A pattern of numbers, star or characters is a way of arranging these in some logical manner or they may form a sequence. Some of these patterns are triangles which have special importance in mathematics. Some patterns are symmetrical while other are not. Please see the complete page and look at comments for many different patterns.

```
*
***
*****
*****
*****
```

We have shown five rows above, in the program you will be asked to enter the numbers of rows you want to print in the pyramid of stars.

### C programming code

```
#include<stdio.h>

main()
{
    int row, c, n, temp;

    printf("Enter the number of rows in pyramid of stars you wish to see ");
    scanf("%d",&n);

    temp = n;

    for ( row = 1 ; row <= n ; row++ )
    {
        for ( c = 1 ; c < temp ; c++ )
            printf(" ");
    }
}
```

```

temp--;

for ( c = 1 ; c <= 2*row - 1 ; c++)
    printf("*");

    printf("\n");
}

return 0;
}

```

Output:

```

Enter the number of rows in pyramid of stars you wish to see 10
      *
     ***
    *****
   *********
  ***********
 *****
*****
*****
*****
*****
*****
*****
*****

```

### 17) C program to print diamond pattern

Diamond pattern in c: This code print diamond pattern of stars. Diamond shape is as follows:

```

*
***
*****
***
*

```

### C programming code

```

#include <stdio.h>

int main()
{
    int n, c, k, space = 1;

    printf("Enter number of rows\n");
    scanf("%d", &n);

    space = n - 1;

    for (k = 1; k <= n; k++)
    {
        for (c = 1; c <= space; c++)
            printf(" ");

        space--;

        for (c = 1; c <= 2*k-1; c++)
            printf("*");

        printf("\n");
    }
}

```

```

}

space = 1;

for (k = 1; k <= n - 1; k++)
{
    for (c = 1; c <= space; c++)
        printf(" ");

    space++;

    for (c = 1 ; c <= 2*(n-k)-1; c++)
        printf("*");

    printf("\n");
}

return 0;
}

```

### 18) C program for prime number

Prime number program in c: c program for prime number, this code prints prime numbers using c programming language. To check whether a number is prime or not see another code below. Prime number logic: a number is prime if it is divisible only by one and itself. Remember two is the only even and also the smallest prime number. First few prime numbers are 2, 3, 5, 7, 11, 13, 17....etc. Prime numbers have many applications in computer science and mathematics. A number greater than one can be factorized into prime numbers, For example  $540 = 2^2 * 3^3 * 5^1$

### Prime number program in c language

```

#include<stdio.h>

main()
{
    int n, i = 3, count, c;

    printf("Enter the number of prime numbers required\n");
    scanf("%d",&n);

    if ( n >= 1 )
    {
        printf("First %d prime numbers are :\n",n);
        printf("2\n");
    }

    for ( count = 2 ; count <= n ; )
    {
        for ( c = 2 ; c <= i - 1 ; c++ )
        {
            if ( i%c == 0 )
                break;
        }
        if ( c == i )
        {

```

```

    printf("%d\n",i);
    count++;
}
i++;
}

return 0;
}

```

There are many logic to check prime numbers, one given below is more efficient then above method.  
for ( c = 2 ; c <= (int)sqrt(n) ; c++ )  
//only checking from 2 to square root of number is sufficient.

There are many more efficient logic then written above.

### C program for prime number or not

```

#include<stdio.h>

main()
{
    int n, c = 2;

    printf("Enter a number to check if it is prime\n");
    scanf("%d",&n);

    for ( c = 2 ; c <= n - 1 ; c++ )
    {
        if ( n%c == 0 )
        {
            printf("%d is not prime.\n", n);
            break;
        }
    }
    if ( c == n )
        printf("%d is prime.\n", n);

    return 0;
}

```

### C program for prime number using function

```

#include<stdio.h>

int check_prime(int);

main()
{
    int n, result;

    printf("Enter an integer to check whether it is prime or not.\n");
    scanf("%d",&n);

    result = check_prime(n);

    if ( result == 1 )

```

```

    printf("%d is prime.\n", n);
else
    printf("%d is not prime.\n", n);

return 0;
}

int check_prime(int a)
{
    int c;

    for ( c = 2 ; c <= a - 1 ; c++ )
    {
        if ( a%c == 0 )
            return 0;
    }
    if ( c == a )
        return 1;
}

```

### 19) C program to generate and print armstrong numbers

armstrong number in c: This program prints armstrong number. In our program we ask the user to enter a number and then we use a loop from one to the entered number and check if it is an armstrong number and if it is then the number is printed on the screen. Remember a number is armstrong if the sum of cubes of individual digits of a number is equal to the number itself. For example 371 is an armstrong number as  $3^3 + 7^3 + 1^3 = 371$ . Some other armstrong numbers are 0, 1, 153, 370, 407.

#### C code

```

#include<stdio.h>
#include<conio.h>

main()
{
    int r;
    long number = 0, c, sum = 0, temp;

    printf("Enter the maximum range upto which you want to find armstrong numbers ");
    scanf("%ld",&number);

    printf("Following armstrong numbers are found from 1 to %ld\n",number);

    for( c = 1 ; c <= number ; c++ )
    {
        temp = c;
        while( temp != 0 )
        {
            r = temp%10;
            sum = sum + r*r*r;
            temp = temp/10;
        }
    }
}

```

```

    if ( c == sum )
        printf("%ld\n", c);
    sum = 0;
}

getch();
return 0;
}

```

## 20) Fibonacci series in c

Fibonacci series in c programming: c program for Fibonacci series without and with recursion. Using the code below you can print as many number of terms of series as desired. Numbers of Fibonacci sequence are known as Fibonacci numbers. First few numbers of series are 0, 1, 1, 2, 3, 5, 8 etc, Except first two terms in sequence every other term is the sum of two previous terms, For example  $8 = 3 + 5$  (addition of 3, 5). This sequence has many applications in mathematics and Computer Science.

### Fibonacci series in c using for loop

```

/* Fibonacci Series c language */
#include<stdio.h>

main()
{
    int n, first = 0, second = 1, next, c;

    printf("Enter the number of terms\n");
    scanf("%d",&n);

    printf("First %d terms of Fibonacci series are :-\n",n);

    for ( c = 0 ; c < n ; c++ )
    {
        if ( c <= 1 )
            next = c;
        else
        {
            next = first + second;
            first = second;
            second = next;
        }
        printf("%d\n",next);
    }

    return 0;
}

```

Output of program:

```
Enter the number of terms
10
First 10 terms of fibonacci series are :
0
1
1
2
3
5
8
13
21
34
```

### Fibonacci series program in c using recursion

```
#include<stdio.h>

int Fibonacci(int);

main()
{
    int n, i = 0, c;

    scanf("%d",&n);

    printf("Fibonacci series\n");

    for ( c = 1 ; c <= n ; c++ )
    {
        printf("%d\n", Fibonacci(i));
        i++;
    }

    return 0;
}

int Fibonacci(int n)
{
    if ( n == 0 )
        return 0;
    else if ( n == 1 )
        return 1;
    else
        return ( Fibonacci(n-1) + Fibonacci(n-2) );
}
```

### 21) Linear search in c

Linear search in c programming: The following code implements linear search ( Searching algorithm ) which is used to find whether a given number is present in an array and if it is present then at what location it occurs. It is also known as sequential search. It is very simple and works as follows: We keep on comparing each element with the element to search until the desired element is found or list ends. Linear search in c language for multiple occurrences and using function.

### Linear search c program

```
#include<stdio.h>

main()
{
```

```

int array[100], search, c, number;

printf("Enter the number of elements in array\n");
scanf("%d",&number);

printf("Enter %d numbers\n", number);

for ( c = 0 ; c < number ; c++ )
    scanf("%d",&array[c]);

printf("Enter the number to search\n");
scanf("%d",&search);

for ( c = 0 ; c < number ; c++ )
{
    if ( array[c] == search ) /* if required element found */
    {
        printf("%d is present at location %d.\n", search, c+1);
        break;
    }
}
if ( c == number )
    printf("%d is not present in array.\n", search);

return 0;
}

```

C program for binary search

Output of code:

```

Enter the number of elements in array
5
Enter 5 numbers
123
56
99
-4568
957
Enter the number to search
99
99 is present at location 3.

```

### Linear search for multiple occurrences

In the code below we will print all the locations at which required element is found and also the number of times it occur in the list.

```

#include<stdio.h>

main()
{
    int array[100], search, c, n, count = 0;

    printf("Enter the number of elements in array\n");
    scanf("%d",&n);

    printf("Enter %d numbers\n", n);

    for ( c = 0 ; c < n ; c++ )
        scanf("%d",&array[c]);

    printf("Enter the number to search\n");

```

```

scanf("%d",&search);

for ( c = 0 ; c < n ; c++ )
{
    if ( array[c] == search )
    {
        printf("%d is present at location %d.\n", search, c+1);
        count++;
    }
}
if ( count == 0 )
    printf("%d is not present in array.\n", search);
else
    printf("%d is present %d times in array.\n", search, count);

return 0;
}

```

Output of code:

```

Enter the number of elements in array
4
Enter 4 numbers
9
56
-9
9
Enter the number to search
9
9 is present at location 1.
9 is present at location 4.
9 is present 2 times in array.

```

### C program for linear search using function

```

#include<stdio.h>

int linear_search(int*, int, int);

main()
{
    int array[100], search, c, n, position;

    printf("Enter the number of elements in array\n");
    scanf("%d",&n);

    printf("Enter %d numbers\n", n);

    for ( c = 0 ; c < n ; c++ )
        scanf("%d",&array[c]);

    printf("Enter the number to search\n");
    scanf("%d",&search);

    position = linear_search(array, n, search);

    if ( position == -1 )
        printf("%d is not present in array.\n", search);
    else

```

```

    printf("%d is present at location %d.\n", search, position+1);

    return 0;
}

int linear_search(int *pointer, int n, int find)
{
    int c;

    for ( c = 0 ; c < n ; c++ )
    {
        if ( *(pointer+c) == find )
            return c;
    }

    return -1;
}

```

## 22) C program for binary search

C program for binary search: This code implements binary search in c language. It can only be used for sorted arrays, but it's fast as compared to linear search. If you wish to use binary search on an array which is not sorted then you must sort it using some sorting technique say merge sort and then use binary search algorithm to find the desired element in the list. If the element to be searched is found then its position is printed.

The code below assumes that the input numbers are in ascending order.

C programming code for binary search

```

#include<stdio.h>

main()
{
    int c, first, last, middle, n, search, array[100];

    printf("Enter number of elements\n");
    scanf("%d",&n);

    printf("Enter %d integers\n", n);

    for ( c = 0 ; c < n ; c++ )
        scanf("%d",&array[c]);

    printf("Enter value to find\n");
    scanf("%d",&search);

    first = 0;
    last = n - 1;
    middle = (first+last)/2;

    while( first <= last )
    {
        if ( array[middle] < search )
            first = middle + 1;
        else if ( array[middle] == search )
        {
            printf("%d found at location %d.\n", search, middle+1);
            break;
        }
        else

```

```

    last = middle - 1;

    middle = (first + last)/2;
}
if ( first > last )
    printf("Not found! %d is not present in the list.\n", search);

return 0;
}

```

### 23) Insertion sort in c

Insertion sort in c: c program for insertion sort to sort numbers. This code implements insertion sort algorithm to arrange numbers of an array in ascending order. With a little modification it will arrange numbers in descending order.

#### Insertion sort algorithm implementation in c

```

/* insertion sort ascending order */

#include <stdio.h>

int main()
{
    int n, array[1000], c, d, t;

    printf("Enter number of elements\n");
    scanf("%d", &n);

    printf("Enter %d integers\n", n);

    for (c = 0; c < n; c++) {
        scanf("%d", &array[c]);
    }

    for (c = 1 ; c <= n - 1; c++) {
        d = c;

        while ( d > 0 && array[d] < array[d-1]) {
            t = array[d];
            array[d] = array[d-1];
            array[d-1] = t;

            d--;
        }
    }
}

```

```

printf("Sorted list in ascending order:\n");

for (c = 0; c <= n - 1; c++) {
    printf("%d\n", array[c]);
}

return 0;
}

```

Output of program:

```

Enter number of elements
5
Enter 5 integers
4
1
-2
8
3
Sorted list in ascending order:
-2
1
3
4
8
Process returned 0 (0x0)   execution time : 11.527 s
Press ENTER to continue.

```

## 24) C program to add two matrix

This c program add two matrices i.e. compute the sum of two matrices and then print it. Firstly user will be asked to enter the order of matrix ( number of rows and columns ) and then two matrices. For example if the user entered order as 2, 2 i.e. two rows and two columns and matrices as

First Matrix :-

```
1 2
3 4
```

Second matrix :-

```
4 5
-1 5
```

then output of the program ( sum of First and Second matrix ) will be

```
5 7
2 9
```

C programming code

```

#include <stdio.h>

main()
{
    int m, n, c, d, first[10][10], second[10][10], sum[10][10];

    printf("Enter the number of rows and columns of matrix ");
    scanf("%d%d", &m, &n);
    printf("Enter the elements of first matrix\n");

    for ( c = 0 ; c < m ; c++ )
        for ( d = 0 ; d < n ; d++ )
            scanf("%d", &first[c][d]);

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

```

```

for ( c = 0 ; c < m ; c++ )
    for ( d = 0 ; d < n ; d++ )
        scanf("%d", &second[c][d]);

for ( c = 0 ; c < m ; c++ )
    for ( d = 0 ; d < n ; d++ )
        sum[c][d] = first[c][d] + second[c][d];

printf("Sum of entered matrices:\n");

for ( c = 0 ; c < m ; c++ )
{
    for ( d = 0 ; d < n ; d++ )
        printf("%d\t", sum[c][d]);

    printf("\n");
}

return 0;
}

```

## 25) c program to transpose a matrix

This c program prints transpose of a matrix. It is obtained by interchanging rows and columns of a matrix. For example if a matrix is

```

1 2
3 4
5 6

```

then transpose of above matrix will be

```

1 3 5
2 4 6

```

When we transpose a matrix then the order of matrix changes, but for a square matrix order remains same.

### C programming code

```

#include<stdio.h>

main()
{
    int m, n, c, d, matrix[10][10], transpose[10][10];

    printf("Enter the number of rows and columns of matrix ");
    scanf("%d%d",&m,&n);
    printf("Enter the elements of matrix \n");

    for( c = 0 ; c < m ; c++ )
    {
        for( d = 0 ; d < n ; d++ )
        {
            scanf("%d",&matrix[c][d]);
        }
    }

    for( c = 0 ; c < m ; c++ )
    {
        for( d = 0 ; d < n ; d++ )
        {

```

```

        transpose[d][c] = matrix[c][d];
    }
}

printf("Transpose of entered matrix :-\n");

for( c = 0 ; c < n ; c++ )
{
    for( d = 0 ; d < m ; d++ )
    {
        printf("%d\t",transpose[c][d]);
    }
    printf("\n");
}

```

#### Output of program:

```

E:\programmingsimplified.com\java>javac TransposeAMatrix.java

E:\programmingsimplified.com\java>java TransposeAMatrix
Enter the number of rows and columns of matrix
4
2
Enter the elements of matrix
1      2
3      4
5      6
7      8
Transpose of entered matrix:-
1      3      5      7
2      4      6      8

```

## 26) C program to compare two strings

This c program compares two strings using strcmp, without strcmp and using pointers. For comparing strings without using library function see another code below.

C program to compare two strings using strcmp

```

#include<stdio.h>
#include<string.h>

main()
{
    char a[100], b[100];

    printf("Enter the first string\n");
    gets(a);

    printf("Enter the second string\n");
    gets(b);

    if( strcmp(a,b) == 0 )
        printf("Entered strings are equal.\n");
    else
        printf("Entered strings are not equal.\n");

    return 0;
}

```

C program to compare two strings without using strcmp  
Here we create our own function to compare strings.

```

int compare(char a[], char b[])
{
    int c = 0;

    while( a[c] == b[c] )
    {
        if( a[c] == '\0' || b[c] == '\0' )
            break;
        c++;
    }
    if( a[c] == '\0' && b[c] == '\0' )
        return 0;
    else
        return -1;
}

```

C program to compare two strings using pointers

In this method we will make our own function to perform string comparison, we will use character pointers in our function to manipulate string.

```

#include<stdio.h>

int compare_string(char*, char*);

main()
{
    char first[100], second[100], result;

    printf("Enter first string\n");
    gets(first);

    printf("Enter second string\n");
    gets(second);

    result = compare_string(first, second);

    if ( result == 0 )
        printf("Both strings are same.\n");
    else
        printf("Entered strings are not equal.\n");

    return 0;
}

int compare_string(char *first, char *second)
{
    while(*first==*second)
    {
        if ( *first == '\0' || *second == '\0' )
            break;

        first++;
        second++;
    }
    if( *first == '\0' && *second == '\0' )
        return 0;
    else
        return -1;
}

```

## 27) String copying in c programming

This program copy string using library function strcpy, to copy string without using strcpy see source code below in which we have made our own function to copy string.

C program to copy a string

```
#include<stdio.h>
#include<string.h>

main()
{
    char source[] = "C program";
    char destination[50];

    strcpy(destination, source);

    printf("Source string: %s\n", source);
    printf("Destination string: %s\n", destination);

    return 0;
}
```

c program to copy a string using pointers

: here we copy string without using strcmp by creating our own function which uses pointers.

```
#include<stdio.h>

void copy_string(char*, char*);

main()
{
    char source[100], target[100];

    printf("Enter source string\n");
    gets(source);

    copy_string(target, source);

    printf("Target string is \"%s\"\n", target);

    return 0;
}

void copy_string(char *target, char *source)
{
    while(*source)
    {
        *target = *source;
        source++;
        target++;
    }
    *target = '\0';
}
```

## 28) C program to concatenate strings

This program concatenates strings, for example if the first string is "c " and second string is "program" then on concatenating these two strings we get the string "c program". To concatenate two strings we use strcat function of string.h, to concatenate without using library function see another code below which uses pointers.

C code

```
#include<stdio.h>
#include<conio.h>
#include<string.h>

main()
{
    char a[100], b[100];

    printf("Enter the first string\n");
    gets(a);

    printf("Enter the second string\n");
    gets(b);

    strcat(a,b);

    printf("String obtained on concatenation is %s\n",a);

    getch();
    return 0;
}
```

string

concatenation.

Output:

```
Enter the first string
under
Enter the second string
stand
String obtained on concatenation is understand
```

String concatenation without strcat

```
#include<stdio.h>

void concatenate_string(char*, char*);

main()
{
    char original[100], add[100];

    printf("Enter source string\n");
    gets(original);

    printf("Enter string to concatenate\n");
    gets(add);

    concatenate_string(original, add);

    printf("String after concatenation is \"%s\"", original);

    return 0;
}

void concatenate_string(char *original, char *add)
{
    while(*original)
```

```

    original++;

while(*add)
{
    *original = *add;
    add++;
    original++;
}
*original = '\0';
}

```

## 29) Reverse string

This program reverses a string entered by the user. For example if a user enters a string "reverse me" then on reversing the string will be "em esrever". We show you three different methods to reverse string the first one uses `strrev` library function of `string.h` header file and in second we make our own function to reverse string using pointers, reverse string using recursion and Reverse words in string. If you are using first method then you must include `string.h` in your program.

C programming code  
 /\* String reverse in c\*/

```

#include<stdio.h>
#include<string.h>

main()
{
    char arr[100];

    printf("Enter a string to reverse\n");
    gets(arr);

    strrev(arr);

    printf("Reverse of entered string is \n%s\n",arr);

    return 0;
}

/* Second method */

```

C program to reverse a string using pointers

: Now we will invert string using pointers or without using library function `strrev`.

```

#include<stdio.h>

int string_length(char*);
void reverse(char*);

main()
{
    char string[100];

    printf("Enter a string\n");
    gets(string);

    reverse(string);
}

```

```

printf("Reverse of entered string is \"%s\".\n", string);

return 0;
}

void reverse(char *string)
{
    int length, c;
    char *begin, *end, temp;

    length = string_length(string);

    begin = string;
    end = string;

    for ( c = 0 ; c < ( length - 1 ) ; c++ )
        end++;

    for ( c = 0 ; c < length/2 ; c++ )
    {
        temp = *end;
        *end = *begin;
        *begin = temp;

        begin++;
        end--;
    }
}

int string_length(char *pointer)
{
    int c = 0;

    while( *(pointer+c) != '\0' )
        c++;

    return c;
}

```

C program to reverse a string using recursion

```

#include<stdio.h>
#include<string.h>

void reverse(char*,int,int);

main()
{
    char a[100];

    gets(a);

    reverse(a, 0, strlen(a)-1);

    printf("%s\n",a);

    return 0;
}

void reverse(char *x, int beg, int end)
{

```

```

char a, b, c;

if ( beg >= end )
    return;

c = *(x+beg);
*(x+beg) = *(x+end);
*(x+end) = c;

reverse(x, ++beg, --end);
}

```

Reverse string program executable.

**Output of program:**

```

Enter a string to reverse
hello
Reverse of entered string is
olleh

```

### 30) C program to print Pascal triangle

Pascal Triangle in c: C program to print Pascal triangle which you might have studied in Binomial Theorem in Mathematics. Number of rows of Pascal triangle to print is entered by the user. First four rows of Pascal triangle are shown below :-

```

1
1 1
1 2 1
1 3 3 1

```

**Pascal triangle in c**

```

#include<stdio.h>

long factorial(int);

main()
{
    int i, n, c;

    printf("Enter the number of rows you wish to see in pascal triangle\n");
    scanf("%d",&n);

    for ( i = 0 ; i < n ; i++ )
    {
        for ( c = 0 ; c <= ( n - i - 2 ) ; c++ )
            printf(" ");

        for( c = 0 ; c <= i ; c++ )
            printf("%ld ",factorial(i)/(factorial(c)*factorial(i-c)));

        printf("\n");
    }

    return 0;
}

```

```
long factorial(int n)
{
    int c;
    long result = 1;

    for( c = 1 ; c <= n ; c++ )
        result = result*c;

    return ( result );
}
```

#### Output of program:

```
Enter the number of rows you wish to see in pascal triangle
5
    1
   1 1
  1 2 1
 1 3 3 1
1 4 6 4 1
```