```c
/**
 * C program to print the sum of all odd numbers from 1 to n
 */

#include <stdio.h>

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

    /* Input range to find sum of odd numbers */
    printf("Enter upper limit: ");
    scanf("%d", &n);

    /* Find the sum of all odd number */
    for(i=1; i<=n; i+=2)
    {
        sum += i;
    }

    printf("Sum of odd numbers = %d", sum);

    return

/**
 * C program to check whether a number is palindrome or not
 */

#include <stdio.h>

int main()
{
    int n, num, rev = 0;

    /* Input a number from user */
    printf("Enter any number to check palindrome: ");
    scanf("%d", &n);

    /* Copy original value to 'num' to 'n'*/
    num = n;

    /* Find reverse of n and store in rev */
    while(n != 0)
    {
        rev = (rev * 10) + (n % 10);
        n  /= 10;
    }
```

```
    /* Check if reverse is equal to 'num' or not */
    if(rev == num)
    {
        printf("%d is palindrome.", num);
    }
    else
    {
        printf("%d is not palindrome.", num);
    }

    return 0;
}



/**
 * C program to swap first and last digit of a number
 */

#include <stdio.h>
#include <math.h>

int main()
{
    int num, swappedNum;
    int firstDigit, lastDigit, digits;

    /* Input number from user */
    printf("Enter any number: ");
    scanf("%d", &num);

    /* Find last digit */
    lastDigit  = num % 10;

    /* Find total number of digit - 1 */
    digits     = (int)log10(num);

    /* Find first digit */
    firstDigit = (int)(num / pow(10, digits));

    swappedNum  = lastDigit;
    swappedNum *= (int) pow(10, digits);
    swappedNum += num % ((int) pow(10, digits));
    swappedNum -= lastDigit;
    swappedNum += firstDigit;

    printf("Original number = %d", num);
```

```
    printf("Number after swapping first and last digit: %d", swappedNum);

    return 0;
}


/**
 * C program to print Fibonacci series up to n terms
 */

#include <stdio.h>

int main()
{
    int a, b, c, i, terms;

    /* Input number from user */
    printf("Enter number of terms: ");
    scanf("%d", &terms);

    /* Fibonacci magic initialization */
    a = 0;
    b = 1;
    c = 0;

    printf("Fibonacci terms: \n");

    /* Iterate through n terms */
    for(i=1; i<=terms; i++)
    {
        printf("%d, ", c);

        a = b;     // Copy n-1 to n-2
        b = c;     // Copy current to n-1
        c = a + b; // New term
    }

    return 0;
}



/**
 * C program to check whether a number is Perfect number or not
 */

#include <stdio.h>
```

```
int main()
{
    int i, num, sum = 0;

    /* Input a number from user */
    printf("Enter any number to check perfect number: ");
    scanf("%d", &num);

    /* Calculate sum of all proper divisors */
    for(i = 1; i <= num / 2; i++)
    {
        /* If i is a divisor of num */
        if(num%i == 0)
        {
            sum += i;
        }
    }

    /* Check whether the sum of proper divisors is equal to num */
    if(sum == num && num > 0)
    {
        printf("%d is PERFECT NUMBER", num);
    }
    else
    {
        printf("%d is NOT PERFECT NUMBER", num);
    }

    return 0;
}
```


C program to print reverse Pyramid star pattern

July 2, 2015PankajC programmingC, Loop, Program, Star Patterns

Write a C program to print reverse Pyramid or reverse equilateral triangle star pattern series using for loop. How to print inverted pyramid or inverted equilateral triangle star pattern series in C program. Logic to print reverse pyramid star pattern series in C programming.

Example

Input

Input rows: 5

Output

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

Required knowledge
Basic C programming, For loop, Nested loop

Logic to print reverse pyramid star pattern

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

The above pattern has N (in this case 5) rows. Each row has exactly N * 2 - (i * 2 - 1) stars. In addition the pattern consists of leading spaces. For each row it contain i - 1 leading space (where i is current row number).

Step by step descriptive logic to print reverse pyramid star pattern.

Input number of rows to print from user. Store it in a variable say rows.
To iterate through rows, run an outer loop from 1 to rows. The loop structure should look like for(i=1; i<=rows; i++).
To print spaces, run an inner loop from 1 to i - 1. The loop structure should look like for(j=1; j<i; j++). Inside this loop print single space.
To print stars, run another inner loop from 1 to rows * 2 - (i * 2 - 1). The loop structure should look like for(j=1; j<= (rows*2 - (i*2-1)); j++). Inside this loop print star.
After printing all stars for each row, move to next line i.e. print new line.
Program to print reverse pyramid star pattern

```
/**
 * C program to print reverse pyramid star pattern
 */

#include <stdio.h>

int main()
{
    int i, j, rows;

    /* Input rows to print from user */
    printf("Enter number of rows : ");
    scanf("%d", &rows);
```

```
    for(i=1; i<=rows; i++)
    {
        /* Print leading spaces */
        for(j=1; j<i; j++)
        {
            printf(" ");
        }

        /* Print stars */
        for(j=1; j<=(rows*2 -(2*i-1)); j++)
        {
            printf("*");
        }

        /* Move to next line */
        printf("\n");
    }

    return 0;
}

/**
 * C program to find cube of any number using function
 */
#include <stdio.h>

/* Function declaration */
double cube(double num);

int main()
{
    int num;
    double c;

    /* Input number to find cube from user */
    printf("Enter any number: ");
    scanf("%d", &num);

    c = cube(num);

    printf("Cube of %d is %.2f", num, c);

    return 0;
}

/**
 * Function to find cube of any number
 */
```

```
double cube(double num)
{
    return (num * num * num);
}




/**
 * C program to check prime, armstrong and perfect numbers using functions
 */

#include <stdio.h>
#include <math.h>


/* Function declarations */
int isPrime(int num);
int isArmstrong(int num);
int isPerfect(int num);


int main()
{
    int num;

    printf("Enter any number: ");
    scanf("%d", &num);

    // Call isPrime() functions
    if(isPrime(num))
    {
        printf("%d is Prime number.\n", num);
    }
    else
    {
        printf("%d is not Prime number.\n", num);
    }

    // Call isArmstrong() function
    if(isArmstrong(num))
    {
        printf("%d is Armstrong number.\n", num);
    }
    else
    {
        printf("%d is not Armstrong number.\n", num);
    }
```

```
    // Call isPerfect() function
    if(isPerfect(num))
    {
        printf("%d is Perfect number.\n", num);
    }
    else
    {
        printf("%d is not Perfect number.\n", num);
    }

    return 0;
}



/**
 * Check whether a number is prime or not.
 * Returns 1 if the number is prime otherwise 0.
 */
int isPrime(int num)
{
    int i;

    for(i=2; i<=num/2; i++)
    {
        /*
         * If the number is divisible by any number
         * other than 1 and self then it is not prime
         */
        if(num%i == 0)
        {
            return 0;
        }
    }

    return 1;
}



/**
 * Check whether a number is Armstrong number or not.
 * Returns 1 if the number is Armstrong number otherwise 0.
 */
int isArmstrong(int num)
{
    int lastDigit, sum, originalNum, digits;
```

```
    sum = 0;

    originalNum = num;

    /* Find total digits in num */
    digits = (int) log10(num) + 1;

    /*
     * Calculate sum of power of digits
     */
    while(num > 0)
    {
        // Extract the last digit
        lastDigit = num % 10;

        // Compute sum of power of last digit
        sum = sum + round(pow(lastDigit, digits));

        // Remove the last digit
        num = num / 10;
    }

    return (originalNum == sum);
}



/**
 * Check whether the number is perfect number or not.
 * Returns 1 if the number is perfect otherwise 0.
 */
int isPerfect(int num)
{
    int i, sum, n;
    sum = 0;
    n = num;

    for(i=1; i<n; i++)
    {
        /* If i is a divisor of num */
        if(n%i == 0)
        {
            sum += i;
        }
    }

    return (num == sum);
}
```

```c
/**
 * C program to find power of a number using recursion
 */

#include <stdio.h>


/* Power function declaration */
double pow(double base, int expo);


int main()
{
    double base, power;
    int expo;

    /* Input base and exponent from user */
    printf("Enter base: ");
    scanf("%lf", &base);
    printf("Enter exponent: ");
    scanf("%d", &expo);

    // Call pow function
    power = pow(base, expo);

    printf("%.2lf ^ %d = %f", base, expo, power);

    return 0;
}


/**
 * Calculate power of any number.
 * Returns base ^ expo
 */
double pow(double base, int expo)
{
    /* Base condition */
    if(expo == 0)
        return 1;
    else if(expo > 0)
        return base * pow(base, expo - 1);
    else
        return 1 / pow(base, -expo);
}

/**
```

```
 * C program to print all natural numbers from 1 to n using recursion
 */

#include <stdio.h>


/* Function declaration */
void printNaturalNumbers(int lowerLimit, int upperLimit);



int main()
{
    int lowerLimit, upperLimit;

    /* Input lower and upper limit from user */
    printf("Enter lower limit: ");
    scanf("%d", &lowerLimit);
    printf("Enter upper limit: ");
    scanf("%d", &upperLimit);

    printf("All natural numbers from %d to %d are: ", lowerLimit, upperLimit);
    printNaturalNumbers(lowerLimit, upperLimit);

    return 0;
}


/**
 * Recursively prints all natural number between the given range.
 */
void printNaturalNumbers(int lowerLimit, int upperLimit)
{
    if(lowerLimit > upperLimit)
        return;

    printf("%d, ", lowerLimit);

    // Recursively call the function to print next number
    printNaturalNumbers(lowerLimit + 1, upperLimit);
}


/**
 * C program to find sum of array elements using recursion
 */

#include <stdio.h>
```

```c
#define MAX_SIZE 100

/* Function declaration to find sum of array */
int sum(int arr[], int start, int len);


int main()
{
    int arr[MAX_SIZE];
    int N, i, sumofarray;


    /* Input size and elements in array  */
    printf("Enter size of the array: ");
    scanf("%d", &N);
    printf("Enter elements in the array: ");
    for(i=0; i<N; i++)
    {
        scanf("%d", &arr[i]);
    }


    sumofarray = sum(arr, 0, N);
    printf("Sum of array elements: %d", sumofarray);

    return 0;
}


/**
 * Recursively find the sum of elements in an array.
 */
int sum(int arr[], int start, int len)
{
    /* Recursion base condition */
    if(start >= len)
        return 0;

    return (arr[start] + sum(arr, start + 1, len));
}



/**
 * C program to merge two sorted array in ascending order
 */

#include <stdio.h>
```

```
#define MAX_SIZE 100      // Maximum size of the array

int main()
{
    int arr1[MAX_SIZE], arr2[MAX_SIZE], mergeArray[MAX_SIZE * 2];
    int size1, size2, mergeSize;
    int index1, index2, mergeIndex;
    int i;

    /* Input size of first array */
    printf("Enter the size of first array : ");
    scanf("%d", &size1);

    /* Input elements in first array */
    printf("Enter elements in first array : ");
    for(i=0; i<size1; i++)
    {
        scanf("%d", &arr1[i]);
    }

    /* Input size of second array */
    printf("\nEnter the size of second array : ");
    scanf("%d", &size2);

    /* Input elements in second array */
    printf("Enter elements in second array : ");
    for(i=0; i<size2; i++)
    {
        scanf("%d", &arr2[i]);
    }


    mergeSize = size1 + size2;


    /*
     * Merge two array in ascending order
     */
    index1 = 0;
    index2 = 0;
    for(mergeIndex=0; mergeIndex < mergeSize; mergeIndex++)
    {
        /*
         * If all elements of one array
         * is merged to final array
         */
        if(index1 >= size1 || index2 >= size2)
        {
```

```
            break;
        }


        if(arr1[index1] < arr2[index2])
        {
            mergeArray[mergeIndex] = arr1[index1];
            index1++;
        }
        else
        {
            mergeArray[mergeIndex] = arr2[index2];
            index2++;
        }
    }

    /*
     * Merge remaining array elements
     */
    while(index1 < size1)
    {
        mergeArray[mergeIndex] = arr1[index1];
        mergeIndex++;
        index1++;
    }
    while(index2 < size2)
    {
        mergeArray[mergeIndex] = arr2[index2];
        mergeIndex++;
        index2++;
    }


    /*
     * Print merged array
     */
    printf("\nArray merged in ascending order : ");
    for(i=0; i<mergeSize; i++)
    {
        printf("%d\t", mergeArray[i]);
    }

    return 0;
}
```


C program to find reverse of array

July 17, 2015PankajC programmingArray, C, Program

Write a C program to input elements in array and find reverse of array. How to find reverse of array in C programming. Logic to find reverse of array in C program.

Example

Input

Input array elements: 10, 5, 16, 35, 500
Output

Array elements after reverse: 500, 35, 16, 5, 10

Required knowledge
Basic Input Output, For loop, While loop, Array

There are various ways to reverse an array. Here I will explain the basic three algorithms to reverse a given array. First the simplest and easiest one, so that ever beginner can get what I am up to.

Logic to print array in reverse order
This algorithm in real does not produces a reversed array. Instead it just prints array in reverse order. If you are looking to reverse the elements then skip to next logic. So here goes step by step descriptive logic to print array in reverse order.

Input size and elements in array from user. Store it in some variable say size and arr.
Run a loop from size - 1 to 0 in decremented style. The loop structure should look like for(i=size-1; i>=0; i--).
Inside loop print current array element i.e. arr[i].
Program to print array in reverse

```
/**
 * C program to print array in reverse order
 */

#include <stdio.h>
#define MAX_SIZE 100      // Defines maximum size of array

int main()
{
    int arr[MAX_SIZE];
    int size, i;

    /* Input size of array */
    printf("Enter size of the array: ");
    scanf("%d", &size);
```

```
    /* Input array elements */
    printf("Enter elements in array: ");
    for(i=0; i<size; i++)
    {
        scanf("%d", &arr[i]);
    }

    /*
     * Print array in reversed order
     */
    printf("\nArray in reverse order: ");
    for(i = size-1; i>=0; i--)
    {
        printf("%d\t", arr[i]);
    }

    return 0;
}


/**
 * C program to search element in array
 */

#include <stdio.h>

#define MAX_SIZE 100  // Maximum array size

int main()
{
    int arr[MAX_SIZE];
    int size, i, toSearch, found;

    /* Input size of array */
    printf("Enter size of array: ");
    scanf("%d", &size);

    /* Input elements of array */
    printf("Enter elements in array: ");
    for(i=0; i<size; i++)
    {
        scanf("%d", &arr[i]);
    }

    printf("\nEnter element to search: ");
    scanf("%d", &toSearch);

    /* Assume that element does not exists in array */
```

```
    found = 0;

    for(i=0; i<size; i++)
    {
        /*
         * If element is found in array then raise found flag
         * and terminate from loop.
         */
        if(arr[i] == toSearch)
        {
            found = 1;
            break;
        }
    }

    /*
     * If element is not found in array
     */
    if(found == 1)
    {
        printf("\n%d is found at position %d", toSearch, i + 1);
    }
    else
    {
        printf("\n%d is not found in the array", toSearch);
    }

    return 0;
}



/**
 * C program to find sum of two matrices of size 3x3
 */

#include <stdio.h>

#define SIZE 3 // Size of the matrix

int main()
{
    int A[SIZE][SIZE]; // Matrix 1
    int B[SIZE][SIZE]; // Matrix 2
    int C[SIZE][SIZE]; // Resultant matrix

    int row, col;
```

```
    /* Input elements in first matrix*/
    printf("Enter elements in matrix A of size 3x3: \n");
    for(row=0; row<SIZE; row++)
    {
        for(col=0; col<SIZE; col++)
        {
            scanf("%d", &A[row][col]);
        }
    }

    /* Input elements in second matrix */
    printf("\nEnter elements in matrix B of size 3x3: \n");
    for(row=0; row<SIZE; row++)
    {
        for(col=0; col<SIZE; col++)
        {
            scanf("%d", &B[row][col]);
        }
    }

    /*
     * Add both matrices A and B entry wise or element wise
     * and stores result in matrix C
     */
    for(row=0; row<SIZE; row++)
    {
        for(col=0; col<SIZE; col++)
        {
            /* Cij = Aij + Bij */
            C[row][col] = A[row][col] + B[row][col];
        }
    }


    /* Print the value of resultant matrix C */
    printf("\nSum of matrices A+B = \n");
    for(row=0; row<SIZE; row++)
    {
        for(col=0; col<SIZE; col++)
        {
            printf("%d ", C[row][col]);
        }
        printf("\n");
    }

    return 0;
}
```

```c
/**
 * C program to find upper triangular matrix
 */

#include <stdio.h>
#define MAX_ROWS 3
#define MAX_COLS 3

int main()
{
    int array[MAX_ROWS][MAX_COLS];
    int row, col, isUpper;

    /* Input elements in matrix from user */
    printf("Enter elements in matrix of size %dx%d: \n", MAX_ROWS, MAX_COLS);
    for(row=0; row<MAX_ROWS; row++)
    {
        for(col=0; col<MAX_COLS; col++)
        {
            scanf("%d", &array[row][col]);
        }
    }

    /* Check Upper triangular matrix condition */
    isUpper = 1;
    for(row=0; row<MAX_ROWS; row++)
    {
        for(col=0; col<MAX_COLS; col++)
        {
            /*
             * If elements below the main diagonal (col<row)
             * is not equal to zero then it is not upper triangular matrix
             */
            if(col<row && array[row][col]!=0)
            {
                isUpper = 0;
            }
        }
    }

    /* Print elements of upper triangular matrix  */
    if(isUpper == 1)
    {
        printf("\nThe matrix is Upper triangular matrix.\n");

        for(row=0; row<MAX_ROWS; row++)
        {
            for(col=0; col<MAX_COLS; col++)
```

```
            {
                printf("%d ", array[row][col]);
            }

            printf("\n");
        }
    }
    else
    {
        printf("\nThe matrix is not Upper triangular matrix.");
    }

    return 0;
}/**
 * C program to find transpose of a matrix
 */

#include <stdio.h>
#define MAX_ROWS 3
#define MAX_COLS 3

int main()
{
    int A[MAX_ROWS][MAX_COLS];  // Original matrix
    int B[MAX_COLS][MAX_ROWS];  // Transpose matrix

    int row, col;

    /* Input elements in matrix A from user */
    printf("Enter elements in matrix of size %dx%d: \n", MAX_ROWS, MAX_COLS);
    for(row=0; row<MAX_ROWS; row++)
    {
        for(col=0; col<MAX_COLS; col++)
        {
            scanf("%d", &A[row][col]);
        }
    }

    /*
     * Find transpose of matrix A
     */
    for(row=0; row<MAX_ROWS; row++)
    {
        for(col=0; col<MAX_COLS; col++)
        {
            /* Store each row of matrix A to each column of matrix B */
            B[col][row] = A[row][col];
        }
```

```
    }

    /* Print the original matrix A */
    printf("\nOriginal matrix: \n");
    for(row=0; row<MAX_ROWS; row++)
    {
        for(col=0; col<MAX_COLS; col++)
        {
            printf("%d ", A[row][col]);
        }

        printf("\n");
    }

    /* Print the transpose of matrix A */
    printf("Transpose of matrix A: \n");
    for(row=0; row<MAX_COLS; row++)
    {
        for(col=0; col<MAX_ROWS; col++)
        {
            printf("%d ", B[row][col]);
        }

        printf("\n");
    }

    return 0;
}

/**
 * C program to check whether a matrix is symmetric matrix or not
 */

#include <stdio.h>
#define SIZE 3

int main()
{
    int A[SIZE][SIZE];  // Original matrix
    int B[SIZE][SIZE];  // Transpose matrix

    int row, col, isSymmetric;

    /* Input elements in matrix A from user */
    printf("Enter elements in matrix of size 3x3: \n");
    for(row=0; row<SIZE; row++)
    {
        for(col=0; col<SIZE; col++)
```

```
    {
        scanf("%d", &A[row][col]);
    }
}

/*
 * Find transpose of matrix A
 */
for(row=0; row<SIZE; row++)
{
    for(col=0; col<SIZE; col++)
    {
        /* Store each row of matrix A to each column of matrix B */
        B[row][col] = A[col][row];
    }
}


/*
 * Check whether matrix A is equal to its transpose or not
 */
isSymmetric = 1;
for(row=0; row<SIZE && isSymmetric; row++)
{
    for(col=0; col<SIZE; col++)
    {
        /* If matrix A is not equal to its transpose */
        if(A[row][col] != B[row][col])
        {
            isSymmetric = 0;
            break;
        }
    }
}

/*
 * If the given matrix is symmetric.
 */
if(isSymmetric == 1)
{
    printf("\nThe given matrix is Symmetric matrix: \n");

    for(row=0; row<SIZE; row++)
    {
        for(col=0; col<SIZE; col++)
        {
            printf("%d ", A[row][col]);
        }
```

```
            printf("\n");
        }
    }
    else
    {
        printf("\nThe given matrix is not Symmetric matrix.");
    }

    return 0;
}

/**
 * C program to find length of a string using for loop
 */

#include <stdio.h>
#define MAX_SIZE 100 // Maximum size of the string

int main()
{
    char text[MAX_SIZE]; /* Declares a string of size 100 */
    int i;
    int count= 0;

    /* Input a string from user */
    printf("Enter any string: ");
    gets(text);

    /* Iterate till the last character of string */
    for(i=0; text[i]!='\0'; i++)
    {
        count++;
    }

    printf("Length of '%s' = %d", text, count);

    return 0;
}
```


C program to concatenate two strings

November 10, 2015PankajC programmingC, Pointer, Program, String

Write a C program to concatenate two strings in single string. How to concatenate two strings to one without using strcat() library function. Adding two strings into one without using inbuilt library function. Logic to concatenate two strings in C programming. C program to concatenate two strings using strcat() library function.

Example

Input

Input string1: I love
Input string2: Codeforwin
Output

Concatenated string: I love Codeforwin

Required knowledge
Basic C programming, Loop, String

Must know -

Program to find length of string
Program to copy string
Concatenation of strings
Concatenation of two strings is the process of joining them together to form a new string. Concatenation appends the second string after first string. Concatenation is sometimes also referred as binary addition of strings i.e. + operation.
For example: I love + Codeforwin = I love Codeforwin

Logic to concatenate two strings

Concatenation of two strings is simple copying a string to another. To concatenate two strings str1 and str2, you will copy all characters of str2 at the end of str1.
Below is the step by step descriptive logic to concatenate two string.

Input two string from user. Store it in some variable say str1 and str2. Here we need to concatenate str2 to str1
Find length of str1 and store in some variable say i = length_of_str;.
Run a loop from 0 till end of str2 and copy each character to str1 from the ith index.
Program to concatenate two strings using while loop

```
/**
 * C program to concatenate two strings
 */

#include <stdio.h>
#define MAX_SIZE 100 // Maximum string size

int main()
{
    char str1[MAX_SIZE], str2[MAX_SIZE];
    int i, j;
```

```
    /* Input two strings from user */
    printf("Enter first string: ");
    gets(str1);
    printf("Enter second string: ");
    gets(str2);


    /* Move till the end of str1 */
    i=0;
    while(str1[i] != '\0')
    {
        i++;
    }

    /* Copy str2 to str1 */
    j = 0;
    while(str2[j] != '\0')
    {
        str1[i] = str2[j];
        i++;
        j++;
    }

    // Make sure that str1 is NULL terminated
    str1[i] = '\0';

    printf("Concatenated string = %s", str1);

    return 0;
}

/**
 * C program to toggle case of each character in a string
 */

#include <stdio.h>
#define MAX_SIZE 100 // Maximum string size

/* Toggle case function declaration */
void toggleCase(char * str);


int main()
{
    char str[MAX_SIZE];

    /* Input string from user */
```

```
    printf("Enter any string: ");
    gets(str);

    printf("String before toggling case: %s", str);

    toggleCase(str);

    printf("String after toggling case: %s", str);

    return 0;
}


/**
 * Toggle case of each character in given string
 */
void toggleCase(char * str)
{
    int i = 0;

    while(str[i] != '\0')
    {
        if(str[i]>='a' && str[i]<='z')
        {
            str[i] = str[i] - 32;
        }
        else if(str[i]>='A' && str[i]<='Z')
        {
            str[i] = str[i] + 32;
        }

        i++;
    }
}


/**
 * C program to find reverse of a string
 */

#include <stdio.h>
#define MAX_SIZE 100 // Maximum string size

int main()
{
    char str[MAX_SIZE], reverse[MAX_SIZE];
    int i, strIndex, revIndex, len;
```

```
    /* Input string from user */
    printf("Enter any string: ");
    gets(str);

    /* Find length of string */
    i = 0;
    while(str[i] != '\0') i++;

    len = i;

    /*
     * Store each character from end of original string
     * to reverse string
     */
    revIndex = 0;
    strIndex = len - 1;
    while(strIndex >= 0)
    {
        reverse[revIndex] = str[strIndex];

        strIndex--;
        revIndex++;
    }
    reverse[revIndex] = '\0';

    printf("\nOriginal string = %s\n", str);
    printf("Reverse string = %s", reverse);

    return 0;
}

/**
 * C program to check whether a string is palindrome or not
 */

#include <stdio.h>
#define MAX_SIZE 100 // Maximum string size

int main()
{
    char str[MAX_SIZE];
    int len, startIndex, endIndex;

    /* Input string from user */
    printf("Enter any string: ");
    gets(str);
```

```
    /* Find length of the string */
    len = 0;
    while(str[len] != '\0') len++;

    startIndex = 0;
    endIndex   = len-1;


    while(startIndex <= endIndex)
    {
        if(str[startIndex] != str[endIndex])
            break;

        startIndex++;
        endIndex--;
    }

    if(startIndex >= endIndex)
    {
        printf("String is Palindrome.");
    }
    else
    {
        printf("String is Not Palindrome.");
    }

    return 0;
}
```

You can also use inbuilt string library functions to make the task easier. Using inbuilt string library function you just need to find reverse of string, then compare it with original string.

Program to check palindrome string using string functions

```
/**
 * C program to check whether a string is palindrome or not using string functions
 */

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

#define MAX_SIZE 100 // Maximum string size

int main()
{
    char str[MAX_SIZE], reverse[MAX_SIZE];
    int flag;

    /* Input string from user */
```

```
    printf("Enter any string: ");
    gets(str);


    strcpy(reverse, str); //Copies original string to reverse
    strrev(reverse);      //Finds the reverse of string

    flag = strcmp(str, reverse); //Checks whether both are equal or not

    /* If both strings are equal */
    if(flag == 0)
    {
        printf("String is Palindrome.");
    }
    else
    {
        printf("String is Not Palindrome.");
    }

    return 0;
}

/**
 * C program to remove all repeated characters from a given string
 */
#include <stdio.h>
#define MAX_SIZE 100 // Maximum string size

/* Function declarations */
void removeDuplicates(char * str);
void removeAll(char * str, const char toRemove, int index);


int main()
{
    char str[MAX_SIZE];

    /* Input string from user */
    printf("Enter any string: ");
    gets(str);

    printf("String before removing duplicates: %s\n", str);

    removeDuplicates(str);

    printf("String after removing duplicates: %s\n", str);

    return 0;
```

```
}


/**
 * Remove all duplicate characters from the given string
 */
void removeDuplicates(char * str)
{
   int i = 0;

   while(str[i] != '\0')
   {
      /* Remove all duplicate of character string[i] */
      removeAll(str, str[i], i + 1);
      i++;
   }
}

/**
 * Remove all occurrences of a given character from string.
 */
void removeAll(char * str, const char toRemove, int index)
{
   int i;

   while(str[index] != '\0')
   {
      /* If duplicate character is found */
      if(str[index] == toRemove)
      {
         /* Shift all characters from current position to one place left */
         i = index;
         while(str[i] != '\0')
         {
            str[i] = str[i + 1];
            i++;
         }
      }
      else
      {
         index++;
      }
   }
}

/**
 * C program to convert string to uppercase
 */
```

```c
#include <stdio.h>
#define MAX_SIZE 100 // Maximum string size


int main()
{
    char str[MAX_SIZE];
    int i;

    /* Input string from user */
    printf("Enter your text : ");
    gets(str);

    for(i=0; str[i]!='\0'; i++)
    {
        /*
         * If current character is lowercase alphabet then
         * convert it to uppercase.
         */
        if(str[i]>='a' && str[i]<='z')
        {
            str[i] = str[i] - 32;
        }
    }

    printf("Uppercase string : %s",str);
    return 0;
}

/**
 * C program to compare two string without using string library functions
 */

#include <stdio.h>
#define MAX_SIZE 100 // Maximum string size

/* Compare function declaration */
int compare(char * str1, char * str2);


int main()
{
    char str1[MAX_SIZE], str2[MAX_SIZE];
    int res;

    /* Input two strings from user */
    printf("Enter first string: ");
```

```
    gets(str1);
    printf("Enter second string: ");
    gets(str2);


    /* Call the compare function to compare strings */
    res = compare(str1, str2);

    if(res == 0)
    {
        printf("Both strings are equal.");
    }
    else if(res < 0)
    {
        printf("First string is lexicographically smaller than second.");
    }
    else
    {
        printf("First string is lexicographically greater than second.");
    }

    return 0;
}

**
 * C program to add two number using pointers
 */

#include <stdio.h>

int main()
{
    int num1, num2, sum;
    int *ptr1, *ptr2;

    ptr1 = &num1; // ptr1 stores the address of num1
    ptr2 = &num2; // ptr2 stores the address of num2

    printf("Enter any two numbers: ");
    scanf("%d%d", ptr1, ptr2);

    sum = *ptr1 + *ptr2;

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

    return 0;
}
```

gram to swap two arrays using pointers

```
/**
 * C program to swap two arrays using pointers
 */

#include <stdio.h>

#define MAX_SIZE 100    // Maximum array size


/* Function declarations */
void inputArray(int *arr, int size);
void printArray(int *arr, int size);
void swapArray(int *sourceArr, int *destArr, int size);




int main()
{
    int sourceArr[MAX_SIZE];
    int destArr[MAX_SIZE];

    int size;


    // Input array size
    printf("Enter size of array: ");
    scanf("%d", &size);

    // Input elements of destination array
    printf("Enter %d elements in source array: ", size);
    inputArray(sourceArr, size);


    // Input element of destination array
    printf("Enter %d elements in destination array: ", size);
    inputArray(destArr, size);


    /*
     * Print elements of both arrays before swapping
     */
    printf("\n\nSource array before swapping: ");
    printArray(sourceArr, size);

    printf("\nDestination array before swapping: ");
```

```
    printArray(destArr, size);


    /* Swap elements of both arrays. */
    swapArray(sourceArr, destArr, size);


    /*
     * Print elements of both arrays after swapping
     */
    printf("\n\nSource array after swapping: ");
    printArray(sourceArr, size);

    printf("\nDestination array after swapping: ");
    printArray(destArr, size);


    return 0;
}



/**
 * C program to multiply two matrix using pointers
 */

#include <stdio.h>

#define ROW 3
#define COL 3


/* Function declarations */
void matrixInput(int mat[][COL]);
void matrixPrint(int mat[][COL]);
void matrixMultiply(int mat1[][COL], int mat2[][COL], int res[][COL]);



int main()
{
    int mat1[ROW][COL];
    int mat2[ROW][COL];
    int product[ROW][COL];


    /*
     * Input elements in matrices.
```

```
     */
    printf("Enter elements in first matrix of size %dx%d\n", ROW, COL);
    matrixInput(mat1);

    printf("Enter elements in second matrix of size %dx%d\n", ROW, COL);
    matrixInput(mat2);


    // Call function to multiply both matrices
    matrixMultiply(mat1, mat2, product);


    // Print product of both matrix
    printf("Product of both matrices is : \n");
    matrixPrint(product);

    return 0;
}



/**
 * C program to concatenate two strings using pointer
 */

#include <stdio.h>
#define MAX_SIZE 100 // Maximum string size

int main()
{
    char str1[MAX_SIZE], str2[MAX_SIZE];
    char * s1 = str1;
    char * s2 = str2;

    /* Input two strings from user */
    printf("Enter first string: ");
    gets(str1);
    printf("Enter second string: ");
    gets(str2);

    /* Move till the end of str1 */
    while(*(++s1));

    /* Copy str2 to str1 */
    while(*(s1++) = *(s2++));

    printf("Concatenated string = %s", str1);
```

```
    return 0;
}
```

Skip to content
Search for:
C program to find reverse of a string
April 28, 2015PankajC programmingC, Pointer, Program, String

Write a C program to find reverse of a given string using loop. How to find reverse of any given string using loop in C programming. Logic to find reverse of a string without using strrev() function in C. C program to reverse a string using strrev() string function.

Example

Input

Output

Reverse string: olleH

Required knowledge
Basic C programming, For loop, String

Must know - Program to find reverse of an array

Logic to find reverse of a string

There are numerous ways to find reverse of a string. Here in this lesson I am going to explain few of them. First let us see the easiest method to find reverse of a string. Below is the step by step descriptive logic to find reverse of a string.

Input a string from user, store it in some variable say str.
Declare another array that will store reverse of the string, say char reverse[SIZE].
Find length of the string and store it in some variable say len.
Initialize two variables that will keep track of original and reverse string. Here we will access original string from last and reverse array from first. Hence, initialize strIndex = len - 1 and revIndex = 0.
Run a loop from len - 1 to 0 in decremented style. The loop structure should look like while(strIndex >= 0).
Inside the loop copy current character from original string to reverse string. Say reverse[revIndex] = str[strIndex];.
After copying, increment revIndex and decrement strIndex.
Read more - Program to reverse order of words in a given string

Program to find reverse of a string

```
/**
 * C program to find reverse of a string
```

```
 */

#include <stdio.h>
#define MAX_SIZE 100 // Maximum string size

int main()
{
    char str[MAX_SIZE], reverse[MAX_SIZE];
    int i, strIndex, revIndex, len;

    /* Input string from user */
    printf("Enter any string: ");
    gets(str);

    /* Find length of string */
    i = 0;
    while(str[i] != '\0') i++;

    len = i;

    /*
     * Store each character from end of original string
     * to reverse string
     */
    revIndex = 0;
    strIndex = len - 1;
    while(strIndex >= 0)
    {
        reverse[revIndex] = str[strIndex];

        strIndex--;
        revIndex++;
    }
    reverse[revIndex] = '\0';

    printf("\nOriginal string = %s\n", str);
    printf("Reverse string = %s", reverse);

    return 0;
}
```

Once you got the above approach, you can easily transform the program in pointers context. Let us re-write the above program more efficiently using pointers.

Program to find reverse of a string using pointers

```
/**
 * C program to find reverse of a string using pointers
 */
```

```c
#include <stdio.h>
#define MAX_SIZE 100 // Maximum string size

int main()
{
    char str[MAX_SIZE], reverse[MAX_SIZE];
    char *s = str;
    char *r = reverse;
    int len = 0;

    /* Input string from user */
    printf("Enter any string: ");
    gets(str);

    /* Find length of string */
    while(*(s++)) len++;

    /*
     * Store each character from end of original string
     * to reverse string
     */
    s--;
    while(len >= 0)
    {
        *(r++) = *(--s);
        len--;
    }
    *r = '\0';

    printf("\nOriginal string = %s\n", str);
    printf("Reverse string = %s", reverse);

    return 0;
}

/**
 * C program to copy one string to another string using pointer
 */

#include <stdio.h>
#define MAX_SIZE 100 // Maximum size of the string

int main()
{
    char text1[MAX_SIZE], text2[MAX_SIZE];
    char * str1 = text1;
    char * str2 = text2;
```

```
    /* Input string from user */
    printf("Enter any string: ");
    gets(text1);

    /* Copy text1 to text2 character by character */
    while(*(str2++) = *(str1++));

    printf("First string = %s\n", text1);
    printf("Second string = %s\n", text2);

    return 0;
}


/**
 * C program to reverse an array using pointers
 */

#include <stdio.h>

#define MAX_SIZE 100

/* Function declaration */
void printArr(int *arr, int size);


int main()
{
    int arr[MAX_SIZE];
    int size;
    int *left = arr;  // Pointer to arr[0]
    int *right;


    // Input size of array
    printf("Enter size of array: ");
    scanf("%d", &size);

    right = &arr[size - 1];  // Pointer to arr[size - 1]

    /*
     * Input elements in array
     */
    printf("Enter elements in array: ");
    while(left <= right)
    {
        scanf("%d", left++);
```

```
    }


    printf("\nArray before reverse: ");
    printArr(arr, size);


    // Make sure that left points to arr[0]
    left = arr;


    // Loop to reverse array
    while(left < right)
    {
        /*
         * Swap element from left of array to right of array.
         */
        *left    ^= *right;
        *right   ^= *left;
        *left    ^= *right;

        // Increment left array pointer and decrement right array pointer
        left++;
        right--;
    }


    printf("\nArray after reverse: ");
    printArr(arr, size);


    return 0;
}
```

Program to add two matrix using pointers

```
/**
 * C proogram to add two matrix using pointers.
 */

#include <stdio.h>

#define ROWS 3
#define COLS 3
```

```
/* Function declaration to input, add and print matrix */
void matrixInput(int mat[][COLS]);
void matrixPrint(int mat[][COLS]);
void matrixAdd(int mat1[][COLS], int mat2[][COLS], int res[][COLS]);


int main()
{
    int mat1[ROWS][COLS], mat2[ROWS][COLS], res[ROWS][COLS];

    // Input elements in first matrix
    printf("Enter elements in first matrix of size %dx%d: \n", ROWS, COLS);
    matrixInput(mat1);

    // Input element in second matrix
    printf("\nEnter elemetns in second matrix of size %dx%d: \n", ROWS, COLS);
    matrixInput(mat2);

    // Finc sum of both matrices and print result
    matrixAdd(mat1, mat2, res);

    printf("\nSum of first and second matrix: \n");
    matrixPrint(res);


    return 0;
}


/**
 * C program to find length of a string using pointer
 */

#include <stdio.h>
#define MAX_SIZE 100 // Maximum size of the string

int main()
{
    char text[MAX_SIZE]; /* Declares a string of size 100 */
    char * str = text; /* Declare pointer that points to text */
    int count = 0;

    /* Input string from user */
    printf("Enter any string: ");
    gets(text);

    /* Iterate though last element of the string */
    while(*(str++) != '\0') count++;
```

```
    printf("Length of '%s' = %d", text, count);

    return 0;
}
```


C program to check whether two matrices are equal or not

July 27, 2015PankajC programmingArray, C, Matrix, Program

Write a C program to enter elements in two matrices and check whether both matrices are equal or not. C program to check whether elements of two matrices are equal or not. Logic to check if two matrices are equal or not in C programming.

Example

Input

```
Input elements of matrix1:
1 2 3
4 5 6
7 8 9

Input elements of matrix2:
1 2 3
4 5 6
7 8 9
```

Output

```
Both matrices are equal
```

Required knowledge
Basic C programming, For loop, Array

Equality of matrix

Two matrices are said to be equal if and only if they are of same size and they have equal corresponding entries. Equality of two matrices A and B can be defined as -

$A_{ij} = B_{ij}$ (Where $1 \le i \le m$ and $1 \le j \le n$).

Equal matrices

Both the matrices are of same dimension and also their corresponding elements are equal. Hence both Matrix A and Matrix B are equal.

Program to check matrix equality

```c
/**
 * C program to check whether two matrices are equal or not
 */

#include <stdio.h>

#define SIZE 3 // Matrix size

int main()
{
    int A[SIZE][SIZE];
    int B[SIZE][SIZE];

    int row, col, isEqual;

    /* Input elements in first matrix from user */
    printf("Enter elements in matrix A of size %dx%d: \n", SIZE, SIZE);
    for(row=0; row<SIZE; row++)
    {
        for(col=0; col<SIZE; col++)
        {
            scanf("%d", &A[row][col]);
        }
    }

    /* Input elements in second matrix from user */
    printf("\nEnter elements in matrix B of size %dx%d: \n");
    for(row=0; row<SIZE; row++)
    {
        for(col=0; col<SIZE; col++)
        {
            scanf("%d", &B[row][col]);
        }
    }

    /* Assumes that the matrices are equal */
    isEqual = 1;

    for(row=0; row<SIZE; row++)
    {
        for(col=0; col<SIZE; col++)
        {
            /*
             * If the corresponding entries of matrices are not equal
             */
            if(A[row][col] != B[row][col])
            {
                isEqual = 0;
```

```
                break;
            }
        }
    }

/**
 * C program to check even or odd number
 */

#include <stdio.h>

int main()
{
    int num;

    /* Input number from user */
    printf("Enter any number to check even or odd: ");
    scanf("%d", &num);

    /* Check if the number is divisible by 2 then it is even */
    if(num % 2 == 0)
    {
        /* num % 2 is 0 */
        printf("Number is Even.");
    }
    else
    {
        /* num % 2 is 1 */
        printf("Number is Odd.");
    }

    return 0;
}

/**
 * C program to check whether a character is vowel or consonant
 */

#include <stdio.h>

int main()
{
    char ch;

    /* Input character from user */
    printf("Enter any character: ");
    scanf("%c", &ch);
```

```
    /* Condition for vowel */
    if(ch=='a' || ch=='e' || ch=='i' || ch=='o' || ch=='u' ||
       ch=='A' || ch=='E' || ch=='I' || ch=='O' || ch=='U')
    {
        printf("'%c' is Vowel.", ch);
    }
    else if((ch >= 'a' && ch <= 'z') || (ch >= 'A' && ch <= 'Z'))
    {
        /* Condition for consonant */
        printf("'%c' is Consonant.", ch);
    }
    else
    {
        /*
         * If it is neither vowel nor consonant
         * then it is not an alphabet.
         */
        printf("'%c' is not an alphabet.", ch);
    }

    return 0;
}

/**
 * C program to check Leap Year
 */

#include <stdio.h>

int main()
{
    int year;

    /* Input year from user */
    printf("Enter year : ");
    scanf("%d", &year);


    /*
     * If year is exactly divisible by 4  and year is not divisible by 100
     * or year is exactly divisible by 400 then
     * the year is leap year.
     * Else year is normal year
     */
    if(((year % 4 == 0) && (year % 100 !=0)) || (year % 400==0))
    {
        printf("LEAP YEAR");
```

```
    }
    else
    {
        printf("COMMON YEAR");
    }

    return 0;
}

/**
 * C program to swap two numbers using bitwise operator
 */

#include <stdio.h>

int main()
{
    int num1, num2;

    /* Input two numbers from user */
    printf("Enter any two numbers: ");
    scanf("%d%d", &num1, &num2);

    printf("Original value of num1 = %d\n", num1);
    printf("Original value of num2 = %d\n", num2);

    /* Swap two numbers */
    num1 ^= num2;
    num2 ^= num1;
    num1 ^= num2;

    printf("Num1 after swapping = %d\n", num1);
    printf("Num2 after swapping = %d\n", num2);

    return 0;
}

/**
 * C program to check even or odd number using bitwise operator
 */

#include <stdio.h>

int main()
{
    int num;

    /* Input number from user */
```

```
    printf("Enter any number: ");
    scanf("%d", &num);

    if(num & 1)
    {
        printf("%d is odd.", num);
    }
    else
    {
        printf("%d is even.", num);
    }

    return 0;
}

/**
 * C program to print diamond star pattern
 */

#include <stdio.h>

int main()
{
    int i, j, rows;
    int stars, spaces;

    printf("Enter rows to print : ");
    scanf("%d", &rows);


    stars = 1;
    spaces = rows - 1;

    /* Iterate through rows */
    for(i=1; i<rows*2; i++)
    {
        /* Print spaces */
        for(j=1; j<=spaces; j++)
            printf(" ");

        /* Print stars */
        for(j=1; j<stars*2; j++)
            printf("*");

        /* Move to next line */
        printf("\n");

        if(i<rows)
```

```
        {
            spaces--;
            stars++;
        }
        else
        {
            spaces++;
            stars--;
        }
    }

    return 0;
}
```