# C hello world program

1. #include <stdio.h>
3. int main()
4. {
5. printf("Hello world**\n**");
6. return 0;
7. }

# 2. Print an integer in C

1. #include <stdio.h>
3. int main()
4. {
5. int a;
7. printf("Enter an integer**\n**");
8. scanf("%d", &a);
10. printf("The integer is: %d**\n**", a);
12. return 0;
13. }

# 3. Addition of two numbers in C

1. #include<stdio.h>
3. int main()
4. {
5. int a, b, c;
7. printf("Enter two numbers to add**\n**");
8. scanf("%d%d", &a, &b);
10. c = a + b;
12. printf("Sum of the numbers = %d**\n**", c);
14. return 0;
15. }

# 4. Even or odd program in C

1. #include <stdio.h>
3. int main()
4. {
5. int n;
7. printf("Enter an integer**\n**");
8. scanf("%d", &n);
10. if (n%2 == 0)
11. printf("Even**\n**");
12. else
13. printf("Odd**\n**");
15. return 0;
16. }

# 5. C program to perform addition, subtraction, multiplication and division

1. #include <stdio.h>
3. int main()
4. {
5. int first, second, add, subtract, multiply;
6. float divide;
8. printf("Enter two integers**\n**");
9. scanf("%d%d", &first, &second);
11. add = first + second;
12. subtract = first - second;
13. multiply = first \* second;
14. divide = first / (float)second;   *//typecasting*
16. printf("Sum = %d**\n**", add);
17. printf("Difference = %d**\n**", subtract);
18. printf("Multiplication = %d**\n**", multiply);
19. printf("Division = %.2f**\n**", divide);
21. return 0;
22. }

# 6. C program to check whether a character is vowel or consonant

1. #include <stdio.h>
3. int main()
4. {
5. char ch;
7. printf("Enter a character**\n**");
8. scanf("%c", &ch);
10. *// Checking both lower and upper case, || is the OR operator*
12. if (ch == 'a' || ch == 'A' || ch == 'e' || ch == 'E' || ch == 'i' || ch == 'I' || ch =='o' || ch=='O' || ch == 'u' || ch == 'U')
13. printf("%c is a vowel.**\n**", ch);
14. else
15. printf("%c isn't a vowel.**\n**", ch);
17. return 0;
18. }

# 7. Leap year program in C

1. #include <stdio.h>
3. int main()
4. {
5. int year;
7. printf("Enter a year to check if it is a leap year**\n**");
8. scanf("%d", &year);
10. if (year%400 == 0) *// Exactly divisible by 400 e.g. 1600, 2000*
11. printf("%d is a leap year.**\n**", year);
12. else if (year%100 == 0) *// Exactly divisible by 100 and not by 400 e.g. 1900, 2100*
13. printf("%d isn't a leap year.**\n**", year);
14. else if (year%4 == 0) *// Exactly divisible by 4 and neither by 100 nor 400 e.g. 2020*
15. printf("%d is a leap year.**\n**", year);
16. else *// Not divisible by 4 or 100 or 400 e.g. 2017, 2018, 2019*
17. printf("%d isn't a leap year.**\n**", year);
19. return 0;
20. }

# 8.Sum of digits in C

1. #include <stdio.h>
3. int main()
4. {
5. int n, t, sum = 0, remainder;
7. printf("Enter an integer**\n**");
8. scanf("%d", &n);
10. t = n;
12. while (t != 0)
13. {
14. remainder = t % 10;
15. sum       = sum + remainder;
16. t         = t / 10;
17. }
19. printf("Sum of digits of %d = %d**\n**", n, sum);
21. return 0;
22. }

# 9. Factorial program in C

1. #include<stdio.h>
3. long factorial(int);
5. int main()
6. {
7. int n;
8. long f;
10. printf("Enter an integer to find its factorial**\n**");
11. scanf("%d", &n);
13. if (n < 0)
14. printf("Factorial of negative integers isn't defined.**\n**");
15. else
16. {
17. f = factorial(n);
18. printf("%d! = %ld**\n**", n, f);
19. }
21. return 0;
22. }
24. long factorial(int n)
25. {
26. if (n == 0)
27. return 1;
28. else
29. return(n \* factorial(n-1));
30. }

# 10. C program to find HCF and LCM

1. #include <stdio.h>
3. int main() {
4. int a, b, x, y, t, gcd, lcm;
6. printf("Enter two integers**\n**");
7. scanf("%d%d", &x, &y);
9. a = x;
10. b = y;
12. while (b != 0) {
13. t = b;
14. b = a % b;
15. a = t;
16. }
18. gcd = a;
19. lcm = (x\*y)/gcd;
21. printf("Greatest common divisor of %d and %d = %d**\n**", x, y, gcd);
22. printf("Least common multiple of %d and %d = %d**\n**", x, y, lcm);
24. return 0;
25. }

# 11. Decimal to binary in C

1. #include <stdio.h>
3. int main()
4. {
5. int n, c, k;
7. printf("Enter an integer in decimal number system**\n**");
8. scanf("%d", &n);
10. printf("%d in binary number system is:**\n**", n);
12. for (c = 31; c >= 0; c--)
13. {
14. k = n >> c;
16. if (k & 1)
17. printf("1");
18. else
19. printf("0");
20. }
22. printf("**\n**");
24. return 0;
25. }

# 12. C program to find ncr and npr

1. #include <stdio.h>
3. long factorial(int);
4. long find\_ncr(int, int);
5. long find\_npr(int, int);
7. int main()
8. {
9. int n, r;
10. long ncr, npr;
12. printf("Enter the value of n and r**\n**");
13. scanf("%d%d",&n,&r);
15. ncr = find\_ncr(n, r);
16. npr = find\_npr(n, r);
18. printf("%dC%d = %ld**\n**", n, r, ncr);
19. printf("%dP%d = %ld**\n**", n, r, npr);
21. return 0;
22. }
24. long find\_ncr(int n, int r) {
25. long result;
27. result = factorial(n)/(factorial(r)\*factorial(n-r));
29. return result;
30. }
32. long find\_npr(int n, int r) {
33. long result;
35. result = factorial(n)/factorial(n-r);
37. return result;
38. }
40. long factorial(int n) {
41. int c;
42. long result = 1;
44. for (c = 1; c <= n; c++)
45. result = result\*c;
47. return result;
48. }

# 13. Sum of n numbers in C

1. #include <stdio.h>
3. int main()
4. {
5. int n, sum = 0, c, value;
7. printf("How many numbers you want to add?**\n**");
8. scanf("%d", &n);
10. printf("Enter %d integers**\n**", n);
12. for (c = 1; c <= n; c++)
13. {
14. scanf("%d", &value);
15. sum = sum + value;
16. }
18. printf("Sum of the integers = %d**\n**", sum);
20. return 0;
21. }

# 14.C program to swap two numbers

1. #include <stdio.h>
3. int main()
4. {
5. int x, y, t;
7. printf("Enter two integers**\n**");
8. scanf("%d%d", &x, &y);
10. printf("Before Swapping**\n**First integer = %d**\n**Second integer = %d**\n**", x, y);
12. t = x;
13. x = y;
14. y = t;
16. printf("After Swapping**\n**First integer = %d**\n**Second integer = %d**\n**", x, y);
18. return 0;
19. }

# 15. C program to reverse a number

1. #include <stdio.h>
3. int main()
4. {
5. int n, reverse = 0;
7. printf("Enter a number to reverse**\n**");
8. scanf("%d", &n);
10. while (n != 0)
11. {
12. reverse = reverse \* 10;
13. reverse = reverse + n%10;
14. n = n/10;
15. }
17. printf("Reverse of entered number is = %d**\n**", reverse);
19. return 0;
20. }

# 16. Palindrome number in C

1. #include <stdio.h>
3. int main()
4. {
5. int n, reverse = 0, t;
7. printf("Enter a number to check if it is a palindrome or not**\n**");
8. scanf("%d", &n);
10. t = n;
12. while (t != 0)
13. {
14. reverse = reverse \* 10;
15. reverse = reverse + t%10;
16. t = t/10;
17. }
19. if (n == reverse)
20. printf("%d is a palindrome number.**\n**", n);
21. else
22. printf("%d isn't a palindrome number.**\n**", n);
24. return 0;
25. }

# Pattern programs in C

1. #include <stdio.h>
3. int main()
4. {
5. int row, c, n, s;
7. printf("Enter the number of rows in pyramid of stars you wish to see**\n**");
8. scanf("%d", &n);
10. s = n;
12. for (row = 1; row <= n; row++)  *// Loop to print rows*
13. {
14. for (c = 1; c < s; c++)  *// Loop to print spaces in a row*
15. printf(" ");
17. s--;
19. for (c = 1; c <= 2\*row - 1; c++) *// Loop to print stars in a row*
20. printf("\*");
22. printf("**\n**");
23. }
25. return 0;
26. }

# 18. C program to print diamond pattern

1. #include <stdio.h>
3. int main()
4. {
5. int n, c, k, space = 1;
7. printf("Enter number of rows**\n**");
8. scanf("%d", &n);
10. space = n - 1;
12. for (k = 1; k <= n; k++)
13. {
14. for (c = 1; c <= space; c++)
15. printf(" ");
17. space--;
19. for (c = 1; c <= 2\*k-1; c++)
20. printf("\*");
22. printf("**\n**");
23. }
25. space = 1;
27. for (k = 1; k <= n - 1; k++)
28. {
29. for (c = 1; c <= space; c++)
30. printf(" ");
32. space++;
34. for (c = 1 ; c <= 2\*(n-k)-1; c++)
35. printf("\*");
37. printf("**\n**");
38. }
40. return 0;
41. }

# 19. Prime number program in C

1. #include<stdio.h>
3. int main()
4. {
5. int n, i = 3, count, c;
7. printf("Enter the number of prime numbers required**\n**");
8. scanf("%d",&n);
10. if ( n >= 1 )
11. {
12. printf("First %d prime numbers are :**\n**",n);
13. printf("2**\n**");
14. }
16. for ( count = 2 ; count <= n ;  )
17. {
18. for ( c = 2 ; c <= i - 1 ; c++ )
19. {
20. if ( i%c == 0 )
21. **break**;
22. }
23. if ( c == i )
24. {
25. printf("%d**\n**", i);
26. count++;
27. }
28. i++;
29. }
31. return 0;
32. }

# 20. Armstrong number C program

1. #include <stdio.h>
3. int power(int, int);
5. int main()
6. {
7. int n, sum = 0, temp, remainder, digits = 0;
9. printf("Input an integer**\n**");
10. scanf("%d", &n);
12. temp = n;
13. *// Count number of digits*
14. while (temp != 0) {
15. digits++;
16. temp = temp/10;
17. }
19. temp = n;
21. while (temp != 0) {
22. remainder = temp%10;
23. sum = sum + power(remainder, digits);
24. temp = temp/10;
25. }
27. if (n == sum)
28. printf("%d is an Armstrong number.**\n**", n);
29. else
30. printf("%d isn't an Armstrong number.**\n**", n);
32. return 0;
33. }
35. int power(int n, int r) {
36. int c, p = 1;
38. for (c = 1; c <= r; c++)
39. p = p\*n;
41. return p;
42. }

# 21. C program to generate and print armstrong numbers

1. #include <stdio.h>
3. int check\_armstrong(int);
4. int power(int, int);
6. int main () {
7. int c, a, b;
9. printf("Input two integers**\n**");
10. scanf("%d%d", &a, &b);
12. for (c = a; c <= b; c++) {
13. if (check\_armstrong(c) == 1)
14. printf("%d**\n**", c);
15. }
17. return 0;
18. }
20. int check\_armstrong(int n) {
21. long long sum = 0, temp;
22. int remainder, digits = 0;
24. temp = n;
26. while (temp != 0) {
27. digits++;
28. temp = temp/10;
29. }
31. temp = n;
33. while (temp != 0) {
34. remainder = temp%10;
35. sum = sum + power(remainder, digits);
36. temp = temp/10;
37. }
39. if (n == sum)
40. return 1;
41. else
42. return 0;
43. }
45. int  power(int n, int r) {
46. int c, p = 1;
48. for (c = 1; c <= r; c++)
49. p = p\*n;
51. return p;
52. }

# 22. Fibonacci series C program

1. */\* Fibonacci series program in C language \*/*
2. #include <stdio.h>
4. int main()
5. {
6. int n, first = 0, second = 1, next, c;
8. printf("Enter the number of terms**\n**");
9. scanf("%d", &n);
11. printf("First %d terms of Fibonacci series are:**\n**", n);
13. for (c = 0; c < n; c++)
14. {
15. if (c <= 1)
16. next = c;
17. else
18. {
19. next = first + second;
20. first = second;
21. second = next;
22. }
23. printf("%d**\n**", next);
24. }
26. return 0;
27. }

# 23. C program to print Floyd's triangle

1. #include <stdio.h>
3. int main()
4. {
5. int n, i,  c, a = 1;
7. printf("Enter the number of rows of Floyd's triangle to print**\n**");
8. scanf("%d", &n);
10. for (i = 1; i <= n; i++)
11. {
12. for (c = 1; c <= i; c++)
13. {
14. printf("%d ",a);
15. a++;
16. }
17. printf("**\n**");
18. }
20. return 0;
21. }

# 24. Pascal triangle C program

1. #include <stdio.h>
3. long factorial(int);
5. int main()
6. {
7. int i, n, c;
9. printf("Enter the number of rows you wish to see in pascal triangle**\n**");
10. scanf("%d",&n);
12. for (i = 0; i < n; i++)
13. {
14. for (c = 0; c <= (n - i - 2); c++)
15. printf(" ");
17. for (c = 0 ; c <= i; c++)
18. printf("%ld ",factorial(i)/(factorial(c)\*factorial(i-c)));
20. printf("**\n**");
21. }
23. return 0;
24. }
26. long factorial(int n)
27. {
28. int c;
29. long result = 1;
31. for (c = 1; c <= n; c++)
32. result = result\*c;
34. return result;
35. }

# 25. C program to add two numbers using pointers

1. #include <stdio.h>
3. int main()
4. {
5. int first, second, \*p, \*q, sum;
7. printf("Enter two integers to add**\n**");
8. scanf("%d%d", &first, &second);
10. p = &first;
11. q = &second;
13. sum = \*p + \*q;
15. printf("Sum of the numbers = %d**\n**", sum);
17. return 0;
18. }

# 26. C program to find maximum element in an array

1. #include <stdio.h>
3. int main()
4. {
5. int array[100], maximum, size, c, location = 1;
7. printf("Enter the number of elements in array**\n**");
8. scanf("%d", &size);
10. printf("Enter %d integers**\n**", size);
12. for (c = 0; c < size; c++)
13. scanf("%d", &array[c]);
15. maximum = array[0];
17. for (c = 1; c < size; c++)
18. {
19. if (array[c] > maximum)
20. {
21. maximum  = array[c];
22. location = c+1;
23. }
24. }
26. printf("Maximum element is present at location %d and it's value is %d.**\n**", location, maximum);
27. return 0;
28. }

# 27. C program to find minimum element in array

1. #include <stdio.h>
3. int main()
4. {
5. int array[100], minimum, size, c, location = 1;
7. printf("Enter number of elements in array**\n**");
8. scanf("%d", &size);
10. printf("Enter %d integers**\n**", size);
12. for (c = 0; c < size; c++)
13. scanf("%d", &array[c]);
15. minimum = array[0];
17. for (c = 1; c < size; c++)
18. {
19. if (array[c] < minimum)
20. {
21. minimum = array[c];
22. location = c+1;
23. }
24. }
26. printf("Minimum element is present at location %d and it's value is %d.**\n**", location, minimum);
27. return 0;
28. }

# 28. Linear search in C

1. #include <stdio.h>
3. int main()
4. {
5. int array[100], search, c, n;
7. printf("Enter number of elements in array**\n**");
8. scanf("%d", &n);
10. printf("Enter %d integer(s)**\n**", n);
12. for (c = 0; c < n; c++)
13. scanf("%d", &array[c]);
15. printf("Enter a number to search**\n**");
16. scanf("%d", &search);
18. for (c = 0; c < n; c++)
19. {
20. if (array[c] == search)    */\* If required element is found \*/*
21. {
22. printf("%d is present at location %d.**\n**", search, c+1);
23. **break**;
24. }
25. }
26. if (c == n)
27. printf("%d isn't present in the array.**\n**", search);
29. return 0;
30. }

# 29. C program for binary search

1. #include <stdio.h>
3. int main()
4. {
5. int c, first, last, middle, n, search, array[100];
7. printf("Enter number of elements**\n**");
8. scanf("%d",&n);
10. printf("Enter %d integers**\n**", n);
12. for (c = 0; c < n; c++)
13. scanf("%d",&array[c]);
15. printf("Enter value to find**\n**");
16. scanf("%d", &search);
18. first = 0;
19. last = n - 1;
20. middle = (first+last)/2;
22. while (first <= last) {
23. if (array[middle] < search)
24. first = middle + 1;
25. else if (array[middle] == search) {
26. printf("%d found at location %d.**\n**", search, middle+1);
27. **break**;
28. }
29. else
30. last = middle - 1;
32. middle = (first + last)/2;
33. }
34. if (first > last)
35. printf("Not found! %d isn't present in the list.**\n**", search);
37. return 0;
38. }

# 30. C program to reverse an array

1. #include <stdio.h>
3. int main()
4. {
5. int n, c, d, a[100], b[100];
7. printf("Enter the number of elements in array**\n**");
8. scanf("%d", &n);
10. printf("Enter array elements**\n**");
12. for (c = 0; c < n ; c++)
13. scanf("%d", &a[c]);
15. */\**
16. *\* Copying elements into array b starting from end of array a*
17. *\*/*
19. for (c = n - 1, d = 0; c >= 0; c--, d++)
20. b[d] = a[c];
22. */\**
23. *\* Copying reversed array into the original.*
24. *\* Here we are modifying original array, this is optional.*
25. *\*/*
27. for (c = 0; c < n; c++)
28. a[c] = b[c];
30. printf("Reverse array is**\n**");
32. for (c = 0; c < n; c++)
33. printf("%d**\n**", a[c]);
35. return 0;
36. }

# 31. C program to insert an element in an array

1. #include <stdio.h>
3. int main()
4. {
5. int array[100], position, c, n, value;
7. printf("Enter number of elements in array**\n**");
8. scanf("%d", &n);
10. printf("Enter %d elements**\n**", n);
12. for (c = 0; c < n; c++)
13. scanf("%d", &array[c]);
15. printf("Enter the location where you wish to insert an element**\n**");
16. scanf("%d", &position);
18. printf("Enter the value to insert**\n**");
19. scanf("%d", &value);
21. for (c = n - 1; c >= position - 1; c--)
22. array[c+1] = array[c];
24. array[position-1] = value;
26. printf("Resultant array is**\n**");
28. for (c = 0; c <= n; c++)
29. printf("%d**\n**", array[c]);
31. return 0;
32. }

# 32. C program to delete an element from an array

1. #include <stdio.h>
3. int main()
4. {
5. int array[100], position, c, n;
7. printf("Enter number of elements in array**\n**");
8. scanf("%d", &n);
10. printf("Enter %d elements**\n**", n);
12. for (c = 0; c < n; c++)
13. scanf("%d", &array[c]);
15. printf("Enter the location where you wish to delete element**\n**");
16. scanf("%d", &position);
18. if (position >= n+1)
19. printf("Deletion not possible.**\n**");
20. else
21. {
22. for (c = position - 1; c < n - 1; c++)
23. array[c] = array[c+1];
25. printf("Resultant array:**\n**");
27. for (c = 0; c < n - 1; c++)
28. printf("%d**\n**", array[c]);
29. }
31. return 0;
32. }

# 33. C program to merge two arrays

1. *// It is assumed that user will enter arrays in ascending order*
2. #include <stdio.h>
4. void merge(int [], int, int [], int, int []);
6. int main() {
7. int a[100], b[100], m, n, c, sorted[200];
9. printf("Input number of elements in first array**\n**");
10. scanf("%d", &m);
12. printf("Input %d integers**\n**", m);
13. for (c = 0; c < m; c++) {
14. scanf("%d", &a[c]);
15. }
17. printf("Input number of elements in second array**\n**");
18. scanf("%d", &n);
20. printf("Input %d integers**\n**", n);
21. for (c = 0; c < n; c++) {
22. scanf("%d", &b[c]);
23. }
25. merge(a, m, b, n, sorted);
27. printf("Sorted array:**\n**");
29. for (c = 0; c < m + n; c++) {
30. printf("%d**\n**", sorted[c]);
31. }
33. return 0;
34. }
36. void merge(int a[], int m, int b[], int n, int sorted[]) {
37. int i, j, k;
39. j = k = 0;
41. for (i = 0; i < m + n;) {
42. if (j < m && k < n) {
43. if (a[j] < b[k]) {
44. sorted[i] = a[j];
45. j++;
46. }
47. else {
48. sorted[i] = b[k];
49. k++;
50. }
51. i++;
52. }
53. else if (j == m) {
54. for (; i < m + n;) {
55. sorted[i] = b[k];
56. k++;
57. i++;
58. }
59. }
60. else {
61. for (; i < m + n;) {
62. sorted[i] = a[j];
63. j++;
64. i++;
65. }
66. }
67. }
68. }

# 34. C program for bubble sort

1. */\* Bubble sort code \*/*
3. #include <stdio.h>
5. int main()
6. {
7. int array[100], n, c, d, swap;
9. printf("Enter number of elements**\n**");
10. scanf("%d", &n);
12. printf("Enter %d integers**\n**", n);
14. for (c = 0; c < n; c++)
15. scanf("%d", &array[c]);
17. for (c = 0 ; c < n - 1; c++)
18. {
19. for (d = 0 ; d < n - c - 1; d++)
20. {
21. if (array[d] > array[d+1]) */\* For decreasing order use < \*/*
22. {
23. swap       = array[d];
24. array[d]   = array[d+1];
25. array[d+1] = swap;
26. }
27. }
28. }
30. printf("Sorted list in ascending order:**\n**");
32. for (c = 0; c < n; c++)
33. printf("%d**\n**", array[c]);
35. return 0;
36. }

# 35. Insertion sort in C

1. */\* Insertion sort ascending order \*/*
3. #include <stdio.h>
5. int main()
6. {
7. int n, array[1000], c, d, t;
9. printf("Enter number of elements**\n**");
10. scanf("%d", &n);
12. printf("Enter %d integers**\n**", n);
14. for (c = 0; c < n; c++)
15. scanf("%d", &array[c]);
17. for (c = 1 ; c <= n - 1; c++) {
18. d = c;
20. while ( d > 0 && array[d-1] > array[d]) {
21. t          = array[d];
22. array[d]   = array[d-1];
23. array[d-1] = t;
25. d--;
26. }
27. }
29. printf("Sorted list in ascending order:**\n**");
31. for (c = 0; c <= n - 1; c++) {
32. printf("%d**\n**", array[c]);
33. }
35. return 0;
36. }

# 36. Selection sort in C

1. #include <stdio.h>
3. int main()
4. {
5. int array[100], n, c, d, position, swap;
7. printf("Enter number of elements**\n**");
8. scanf("%d", &n);
10. printf("Enter %d integers**\n**", n);
12. for (c = 0; c < n; c++)
13. scanf("%d", &array[c]);
15. for (c = 0; c < (n - 1); c++)
16. {
17. position = c;
19. for (d = c + 1; d < n; d++)
20. {
21. if (array[position] > array[d])
22. position = d;
23. }
24. if (position != c)
25. {
26. swap = array[c];
27. array[c] = array[position];
28. array[position] = swap;
29. }
30. }
32. printf("Sorted list in ascending order:**\n**");
34. for (c = 0; c < n; c++)
35. printf("%d**\n**", array[c]);
37. return 0;
38. }

# 37. Matrix addition in C

1. #include <stdio.h>
3. int main()
4. {
5. int m, n, c, d, first[10][10], second[10][10], sum[10][10];
7. printf("Enter the number of rows and columns of matrix**\n**");
8. scanf("%d%d", &m, &n);
9. printf("Enter the elements of first matrix**\n**");
11. for (c = 0; c < m; c++)
12. for (d = 0; d < n; d++)
13. scanf("%d", &first[c][d]);
15. printf("Enter the elements of second matrix**\n**");
17. for (c = 0; c < m; c++)
18. for (d = 0 ; d < n; d++)
19. scanf("%d", &second[c][d]);
21. printf("Sum of entered matrices:-**\n**");
23. for (c = 0; c < m; c++) {
24. for (d = 0 ; d < n; d++) {
25. sum[c][d] = first[c][d] + second[c][d];
26. printf("%d**\t**", sum[c][d]);
27. }
28. printf("**\n**");
29. }
31. return 0;
32. }

# 38. Subtract matrices

1. #include <stdio.h>
3. int main()
4. {
5. int m, n, c, d, first[10][10], second[10][10], difference[10][10];
7. printf("Enter the number of rows and columns of matrix**\n**");
8. scanf("%d%d", &m, &n);
9. printf("Enter the elements of first matrix**\n**");
11. for (c = 0; c < m; c++)
12. for (d = 0 ; d < n; d++)
13. scanf("%d", &first[c][d]);
15. printf("Enter the elements of second matrix**\n**");
17. for (c = 0; c < m; c++)
18. for (d = 0; d < n; d++)
19. scanf("%d", &second[c][d]);
21. printf("Difference of entered matrices:-**\n**");
23. for (c = 0; c < m; c++) {
24. for (d = 0; d < n; d++) {
25. difference[c][d] = first[c][d] - second[c][d];
26. printf("%d**\t**",difference[c][d]);
27. }
28. printf("**\n**");
29. }
31. return 0;
32. }

# 39. Transpose of a matrix in C

1. #include <stdio.h>
3. int main()
4. {
5. int m, n, c, d, matrix[10][10], transpose[10][10];
7. printf("Enter the number of rows and columns of matrix**\n**");
8. scanf("%d%d", &m, &n);
10. printf("Enter elements of the matrix**\n**");
12. for (c = 0; c < m; c++)
13. for(d = 0; d < n; d++)
14. scanf("%d", &matrix[c][d]);
16. for (c = 0; c < m; c++)
17. for( d = 0 ; d < n ; d++ )
18. transpose[d][c] = matrix[c][d];
20. printf("Transpose of the matrix:**\n**");
22. for (c = 0; c < n; c++) {
23. for (d = 0; d < m; d++)
24. printf("%d**\t**", transpose[c][d]);
25. printf("**\n**");
26. }
28. return 0;
29. }

# 40. Matrix multiplication in C

1. #include <stdio.h>
3. int main()
4. {
5. int m, n, p, q, c, d, k, sum = 0;
6. int first[10][10], second[10][10], multiply[10][10];
8. printf("Enter number of rows and columns of first matrix**\n**");
9. scanf("%d%d", &m, &n);
10. printf("Enter elements of first matrix**\n**");
12. for (c = 0; c < m; c++)
13. for (d = 0; d < n; d++)
14. scanf("%d", &first[c][d]);
16. printf("Enter number of rows and columns of second matrix**\n**");
17. scanf("%d%d", &p, &q);
19. if (n != p)
20. printf("The matrices can't be multiplied with each other.**\n**");
21. else
22. {
23. printf("Enter elements of second matrix**\n**");
25. for (c = 0; c < p; c++)
26. for (d = 0; d < q; d++)
27. scanf("%d", &second[c][d]);
29. for (c = 0; c < m; c++) {
30. for (d = 0; d < q; d++) {
31. for (k = 0; k < p; k++) {
32. sum = sum + first[c][k]\*second[k][d];
33. }
35. multiply[c][d] = sum;
36. sum = 0;
37. }
38. }
40. printf("Product of the matrices:**\n**");
42. for (c = 0; c < m; c++) {
43. for (d = 0; d < q; d++)
44. printf("%d**\t**", multiply[c][d]);
46. printf("**\n**");
47. }
48. }
50. return 0;
51. }

# 41. C program to print a string

1. #include <stdio.h>
3. int main()
4. {
5. char z[100] = "I am learning C programming language.";
7. printf("%s", z); *// %s is format specifier*
9. return 0;
10. }

# 42. C string length program

1. #include <stdio.h>
2. #include <string.h>
4. int main()
5. {
6. char a[100];
7. int length;
9. printf("Enter a string to calculate it's length**\n**");
10. gets(a);
12. length = strlen(a);
14. printf("Length of the string = %d**\n**", length);
16. return 0;
17. }

# 43. C program to compare two strings

1. #include <stdio.h>
2. #include <string.h>
4. int main()
5. {
6. char a[100], b[100];
8. printf("Enter a string**\n**");
9. gets(a);
11. printf("Enter a string**\n**");
12. gets(b);
14. if (strcmp(a,b) == 0)
15. printf("The strings are equal.**\n**");
16. else
17. printf("The strings are not equal.**\n**");
19. return 0;
20. }

# 44. C copy string program

1. #include <stdio.h>
2. #include <string.h>
4. int main()
5. {
6. char source[1000], destination[1000];
8. printf("Input a string**\n**");
9. gets(source);
11. strcpy(destination, source);
13. printf("Source string: %s**\n**", source);
14. printf("Destination string: %s**\n**", destination);
16. return 0;
17. }

# 45. C program to concatenate strings

1. #include <stdio.h>
2. #include <string.h>
4. int main()
5. {
6. char a[1000], b[1000];
8. printf("Enter the first string**\n**");
9. gets(a);
11. printf("Enter the second string**\n**");
12. gets(b);
14. strcat(a, b);
16. printf("String obtained on concatenation: %s**\n**", a);
18. return 0;
19. }

# 46. C program to reverse a string

1. #include <stdio.h>
2. #include <string.h>
4. int main()
5. {
6. char arr[100];
8. printf("Enter a string to reverse**\n**");
9. gets(arr);
11. strrev(arr);
13. printf("Reverse of the string is **\n**%s**\n**", arr);
15. return 0;
16. }

# 47. Palindrome in C

1. #include <stdio.h>
2. #include <string.h>
4. int main()
5. {
6. char a[100], b[100];
8. printf("Enter a string to check if it is a palindrome**\n**");
9. gets(a);
11. strcpy(b, a);  *// Copying input string*
12. strrev(b);  *// Reversing the string*
14. if (strcmp(a, b) == 0)  *// Comparing input string with the reverse string*
15. printf("The string is a palindrome.**\n**");
16. else
17. printf("The string isn't a palindrome.**\n**");
19. return 0;
20. }

# 48. C program to delete vowels from a string

1. #include <stdio.h>
2. #include <string.h>
4. int check\_vowel(char);
6. int main()
7. {
8. char s[100], t[100];
9. int c, d = 0;
11. printf("Enter a string to delete vowels**\n**");
12. gets(s);
14. for(c = 0; s[c] != '**\0**'; c++) {
15. if(check\_vowel(s[c]) == 0) {       *// If not a vowel*
16. t[d] = s[c];
17. d++;
18. }
19. }
21. t[d] = '**\0**';
23. strcpy(s, t);  *// We are changing initial string. This is optional.*
25. printf("String after deleting vowels: %s**\n**", s);
27. return 0;
28. }
30. int check\_vowel(char ch)
31. {
32. if (ch == 'a' || ch == 'A' || ch == 'e' || ch == 'E' || ch == 'i' || ch == 'I' || ch =='o' || ch=='O' || ch == 'u' || ch == 'U')
33. return 1;
34. else
35. return 0;
36. }

# 49. C substring, substring in C

1. #include <stdio.h>
3. int main()
4. {
5. char string[1000], sub[1000];
6. int position, length, c = 0;
8. printf("Input a string**\n**");
9. gets(string);
11. printf("Enter the position and length of substring**\n**");
12. scanf("%d%d", &position, &length);
14. while (c < length) {
15. sub[c] = string[position+c-1];
16. c++;
17. }
18. sub[c] = '**\0**';
20. printf("Required substring is **\"**%s**\"\n**", sub);
22. return 0;
23. }

# 50. C program to check subsequence

1. #include <stdio.h>
2. #include <string.h>
4. int check\_subsequence (char [], char[]);
6. int main () {
7. int flag;
8. char s1[1000], s2[1000];
10. printf("Input first string**\n**");
11. gets(s1);
13. printf("Input second string**\n**");
14. gets(s2);
16. */\*\* Passing smaller length string first \*/*
18. if (strlen(s1) < strlen(s2))
19. flag = check\_subsequence(s1, s2);
20. else
21. flag = check\_subsequence(s2, s1);
23. if (flag)
24. printf("YES**\n**");
25. else
26. printf("NO**\n**");
28. return 0;
29. }
31. int check\_subsequence (char a[], char b[]) {
32. int c, d;
34. c = d = 0;
36. while (a[c] != '**\0**') {
37. while ((a[c] != b[d]) && b[d] != '**\0**') {
38. d++;
39. }
40. if (b[d] == '**\0**')
41. **break**;
42. d++;
43. c++;
44. }
45. if (a[c] == '**\0**')
46. return 1;
47. else
48. return 0;
49. }

# 51. C program to sort a string in alphabetic order

1. #include <stdio.h>
2. #include <stdlib.h>
3. #include <string.h>
5. int main()
6. {
7. char ch, input[100], output[100];
8. int no[26] = {0}, n, c, t, x;
10. printf("Enter some text**\n**");
11. scanf("%s", input);
13. n = strlen(input);
15. */\*\* Storing how many times characters (a to z)*
16. *appears in input string in an array \*/*
18. for (c = 0; c < n; c++)
19. {
20. ch = input[c] - 'a';
21. no[ch]++;
22. }
24. t = 0;
26. */\*\* Insert characters 'a' to 'z' in output string as many times*
27. *as they appear in the input string \*/*
29. for (ch = 'a'; ch <= 'z'; ch++)
30. {
31. x = ch - 'a';
33. for (c = 0; c < no[x]; c++)
34. {
35. output[t] = ch;
36. t++;
37. }
38. }
39. output[t] = '**\0**';
41. printf("%s**\n**", output);
43. return 0;
44. }

# 52. C program remove spaces, blanks from a string

1. #include <stdio.h>
3. int main()
4. {
5. char text[1000], blank[1000];
6. int c = 0, d = 0;
8. printf("Enter some text**\n**");
9. gets(text);
11. while (text[c] != '**\0**') {
12. if (text[c] == ' ') {
13. int temp = c + 1;
14. if (text[temp] != '**\0**') {
15. while (text[temp] == ' ' && text[temp] != '**\0**') {
16. if (text[temp] == ' ') {
17. c++;
18. }
19. temp++;
20. }
21. }
22. }
23. blank[d] = text[c];
24. c++;
25. d++;
26. }
28. blank[d] = '**\0**';
30. printf("Text after removing blanks**\n**%s**\n**", blank);
32. return 0;
33. }

# 53. C program to change case of a string

1. #include <stdio.h>
2. #include <string.h>
4. int main()
5. {
6. char string[1000];
8. printf("Input a string to convert to lower case**\n**");
9. gets(string);
11. printf("The string in lower case: %s**\n**", strlwr(string));
13. return  0;
14. }

# 54. C program to swap two strings

1. #include <stdio.h>
2. #include <string.h>
4. int main()
5. {
6. char first[100], second[100], temp[100];
8. printf("Enter first string**\n**");
9. gets(first);
11. printf("Enter second string**\n**");
12. gets(second);
14. printf("**\n**Before Swapping**\n**");
15. printf("First string: %s**\n**", first);
16. printf("Second string: %s**\n\n**", second);
18. strcpy(temp, first);
19. strcpy(first, second);
20. strcpy(second, temp);
22. printf("After Swapping**\n**");
23. printf("First string: %s**\n**", first);
24. printf("Second string: %s**\n**", second);
26. return 0;
27. }

# 55. C program to find the frequency of characters in a string

1. #include <stdio.h>
2. #include <string.h>
4. int main()
5. {
6. char string[100];
7. int c = 0, count[26] = {0}, x;
9. printf("Enter a string**\n**");
10. gets(string);
12. while (string[c] != '**\0**') {
13. */\*\* Considering characters from 'a' to 'z' only and ignoring others. \*/*
15. if (string[c] >= 'a' && string[c] <= 'z') {
16. x = string[c] - 'a';
17. count[x]++;
18. }
20. c++;
21. }
23. for (c = 0; c < 26; c++)
24. printf("%c occurs %d times in the string.**\n**", c + 'a', count[c]);
26. return 0;
27. }

# 56. Anagram program in C

1. #include <stdio.h>
3. int check\_anagram(char [], char []);
5. int main()
6. {
7. char a[100], b[100];
9. printf("Enter a string**\n**");
10. gets(a);
12. printf("Enter a string**\n**");
13. gets(b);
15. if (check\_anagram(a, b) == 1)
16. printf("The strings are anagrams.**\n**");
17. else
18. printf("The strings aren't anagrams.**\n**");
20. return 0;
21. }
23. int check\_anagram(char a[], char b[])
24. {
25. int first[26] = {0}, second[26] = {0}, c=0;
27. *// Calculating frequency of characters of first string*
29. while (a[c] != '**\0**')
30. {
31. first[a[c]-'a']++;
32. c++;
33. }
35. c = 0;
37. while (b[c] != '**\0**')
38. {
39. second[b[c]-'a']++;
40. c++;
41. }
43. *// Comparing frequency of characters*
45. for (c = 0; c < 26; c++)
46. {
47. if (first[c] != second[c])
48. return 0;
49. }
51. return 1;
52. }

# 57. C read file program

1. #include <stdio.h>
2. #include <stdlib.h>
4. int main()
5. {
6. char ch, file\_name[25];
7. FILE \*fp;
9. printf("Enter name of a file you wish to see**\n**");
10. gets(file\_name);
12. fp = fopen(file\_name, "r"); *// read mode*
14. if (fp == NULL)
15. {
16. perror("Error while opening the file.**\n**");
17. exit(EXIT\_FAILURE);
18. }
20. printf("The contents of %s file are:**\n**", file\_name);
22. while((ch = fgetc(fp)) != EOF)
23. printf("%c", ch);
25. fclose(fp);
26. return 0;
27. }

# 58. C program to copy a file

1. #include <stdio.h>
2. #include <stdlib.h>
4. int main()
5. {
6. char ch, source\_file[20], target\_file[20];
7. FILE \*source, \*target;
9. printf("Enter name of file to copy**\n**");
10. gets(source\_file);
12. source = fopen(source\_file, "r");
14. if (source == NULL)
15. {
16. printf("Press any key to exit...**\n**");
17. exit(EXIT\_FAILURE);
18. }
20. printf("Enter name of target file**\n**");
21. gets(target\_file);
23. target = fopen(target\_file, "w");
25. if (target == NULL)
26. {
27. fclose(source);
28. printf("Press any key to exit...**\n**");
29. exit(EXIT\_FAILURE);
30. }
32. while ((ch = fgetc(source)) != EOF)
33. fputc(ch, target);
35. printf("File copied successfully.**\n**");
37. fclose(source);
38. fclose(target);
40. return 0;
41. }

# 59. C program to merge two files

1. #include <stdio.h>
2. #include <stdlib.h>
4. int main()
5. {
6. FILE \*fs1, \*fs2, \*ft;
8. char ch, file1[20], file2[20], file3[20];
10. printf("Enter name of first file**\n**");
11. gets(file1);
13. printf("Enter name of second file**\n**");
14. gets(file2);
16. printf("Enter name of file which will store contents of the two files**\n**");
17. gets(file3);
19. fs1 = fopen(file1, "r");
20. fs2 = fopen(file2, "r");
22. if(fs1 == NULL || fs2 == NULL)
23. {
24. perror("Error ");
25. printf("Press any key to exit...**\n**");
26. exit(EXIT\_FAILURE);
27. }
29. ft = fopen(file3, "w"); *// Opening in write mode*
31. if(ft == NULL)
32. {
33. perror("Error ");
34. printf("Press any key to exit...**\n**");
35. exit(EXIT\_FAILURE);
36. }
38. while((ch = fgetc(fs1)) != EOF)
39. fputc(ch,ft);
41. while((ch = fgetc(fs2)) != EOF)
42. fputc(ch,ft);
44. printf("The two files were merged into %s file successfully.**\n**", file3);
46. fclose(fs1);
47. fclose(fs2);
48. fclose(ft);
50. return 0;
51. }

# 60. C program to list files of a directory

1. #include <stdio.h>
2. #include <conio.h>
3. #include <dir.h>
5. int main()
6. {
7. int done;
8. struct ffblk a;
10. printf("Press any key to view the files in the current directory**\n**");
12. getch();
14. done = findfirst("\*.\*", &a, 0); *// The first '\*' is for all file names and the second one is for all file extensions*
16. while(!done)
17. {
18. printf("%s**\n**", a.ff\_name);
19. done = findnext(&a);
20. }
22. getch();
23. return 0;
24. }

# 61. C program to delete a file

1. #include <stdio.h>
3. int main()
4. {
5. int status;
6. char file\_name[25];
8. printf("Enter name of a file you wish to delete**\n**");
9. gets(file\_name);
11. status = remove(file\_name);
13. if (status == 0)
14. printf("%s file deleted successfully.**\n**", file\_name);
15. else
16. {
17. printf("Unable to delete the file**\n**");
18. perror("Following error occurred");
19. }
21. return 0;
22. }

# 62. C program to generate random numbers