

# VISVESVARAYA TECHNOLOGICAL UNIVERSITY

JNANA SANGAMA, BELGAVI -590 014



## Lab Manual (EC STREAM)

“Introduction to C Programming”

**INTEGRATED**

**(Theory & Practical)**

(Effective from the academic Year 2023-2024)

SEMESTER – I

Subject Code: BESCK104E

(Choice Based Credit System)

**Prof. MONISHA R** B.E.,M.Tech.

Asst. Professor. Dept. of CS&E,  
S.J.M.I.T, CHITRADURGA.

**Prof. NOOR FATHIMA** B.E.,M.Tech.

Asst. Professor. Dept. of EEE,  
S.J.M.I.T, CHITRADURGA.



**DEPARTMENT OF CS&E AND EE AND ENGINEERING**

SJM Vidyapeetha®

**Sri Jagadguru Mallikarjuna Murugharajendra  
Institute of Technology**

CHITRADURGA -577 502



SJM Vidyapeetha®

# **Sri Jagadguru Mallikarjuna Murugharajendra Institute of Technology**

(Recognized by AICTE, New Delhi and Affiliated to Visvesvaraya Technological  
University, Belagavi)

**NAAC Accredited with 'B++' Grade**

NH4 Bypass, PB No: 73, CHITRADURGA -577502, Karnataka State, INDIA.



Year: 2023 - 2024

## **LAB MANUAL (EC Stream)**

**INTEGRATED**

**(Theory / Practical)**

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

**“Introduction to C Programming”**

(Effective from the academic Year 2023-24)

SEMESTER – I

Subject Code: BESCK104E

(Choice Based Credit System)

STUDENT'S NAME: .....

USN: .....

BRANCH: .....

SECTION: ..... YEAR: .....

## Procedure to Write and Execute C Program

1. **Install Turbo C (if not already installed):** Download and install Turbo C/C++ on your system. Make sure it's properly installed and working.
2. **Launch Turbo C:** Open the Turbo C IDE by running the "Turbo C" or "Turbo C++" application, depending on your version.
3. **Create a New Project (Optional):** Turbo C allows you to create and manage projects, but for a simple program, you can also work without creating a project.
  - To create a new project, go to the "File" menu, select "New," and choose "Project."
4. **Write Your C Program:**
  - In the Turbo C IDE, you'll see a text editor where you can write your C program.
5. **Save Your Program:** Save your C program with a .c extension. For example, you can save it as hello.c.
6. **Compile Your Program:** Go to the "Compile" menu and select "Compile" (or press Alt+F9).
  - If there are no errors in your program, the compiler will create an executable file (usually with a .exe extension) in the same directory where your .c file is saved.
7. **Run Your Program:** Go to the "Run" menu and select "Run" (or press Ctrl+F9).
  - Your program will execute, and you should see the output in the Turbo C console window.
8. **View Output:** The output of your program will be displayed in the Turbo C console window.
9. **Debug (if necessary):** If your program has errors, you can use the Turbo C IDE's debugging tools to identify and fix them.
10. **Exit Turbo C:** After you have finished working with Turbo C, you can exit the IDE.

**PROGRAMMING EXERCISE**

1.	C Program to find Mechanical Energy of a particle using $E = mgh + \frac{1}{2}mv^2$
2.	C Program to convert Kilometers into Meters and Centimeters.
3.	C Program To Check the Given Character is Lowercase or Uppercase or Special Character.
4.	Program to balance the given Chemical Equation values x, y, p, q of a simple chemical equation of the type: The task is to find the values of constants b1, b2, b3 such that the equation is balanced on both sides and it must be the reduced form.
5.	Implement Matrix multiplication and validate the rules of multiplication.
6.	Compute $\sin(x)/\cos(x)$ using Taylor series approximation. Compare your result with the builtin library function. Print both the results with appropriate.
7.	Sort the given set of N numbers using Bubble sort.
8.	Write functions to implement string operations such as compare, concatenate, string length. Convince the parameter passing techniques.
9.	Implement structures to read, write and compute averagemarks and the students scoring above and below the average marks for a class of N students.
10.	Develop a program using pointers to compute the sum, mean and standard deviation of all elements stored in an array of N real numbers.

**PRACTICE PROGRAMS****1. Write a C code to Add Two Numbers.**

```
#include <stdio.h>
int main() {
    int num1, num2, sum;

    // Input the first integer
    printf("Enter the first integer: ");
    scanf("%d", &num1);

    // Input the second integer
    printf("Enter the second integer: ");
    scanf("%d", &num2);

    // Add the two integers
    sum = num1 + num2;

    // Display the result
    printf("The sum of %d and %d is %d\n", num1, num2, sum);

    return 0;
}
```

**OUTPUT**

Enter the first integer: 5

Enter the second integer: 7

The sum of 5 and 7 is 12

**2. Write a C code to check whether the given number is even or odd.**

```
#include <stdio.h>

int main() {
    int number;

    // Input the number
    printf("Enter an integer: ");
    scanf("%d", &number);

    // Check if the number is even or odd
    if (number % 2 == 0) {
        printf("%d is an even number.\n", number);
    } else {
        printf("%d is an odd number.\n", number);
    }

    return 0;
}
```

**OUTPUT**

Enter an integer: 7

7 is an odd number.

Enter an integer: 12

12 is an even number.

**3. Write a C code to check whether the given alphabet is vowel or consonant.**

```
#include <stdio.h>

int main() {
    char ch;

    // Input a character
    printf("Enter a character: ");
    scanf(" %c", &ch); // Note the space before %c to consume any leading whitespace
    // Check if the character is an alphabet
    if ((ch >= 'a' && ch <= 'z') || (ch >= 'A' && ch <= 'Z')) {
        // Convert the character to lowercase
        ch = tolower(ch);
        // Check if it's a vowel
        if (ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u') {
            printf("%c is a vowel.\n", ch);
        } else {
            printf("%c is a consonant.\n", ch);
        }
    } else {
        printf("Invalid input. Please enter an alphabet.\n");
    }
    return 0;
}
```

**OUTPUT**

Enter a character: A

A is a vowel.

Enter a character: G

G is a consonant.

**4. Write a C code check whether the given number is positive or negative.**

```
#include <stdio.h>

int main() {
    double number;
    // Input the number
    printf("Enter a number: ");
    scanf("%lf", &number);
    // Check if the number is positive, negative, or zero
    if (number > 0) {
        printf("%.2lf is a positive number.\n", number);
    } else if (number < 0) {
        printf("%.2lf is a negative number.\n", number);
    } else {
        printf("The number is zero.\n");
    }
    return 0;
}
```

**OUTPUT**

Enter a number: 5.25

5.25 is a positive number.

Enter a number: -3.75

-3.75 is a negative number.



**5. Write a C code to find greatest of three numbers.**

```
#include <stdio.h>

int main() {

    int num1, num2, num3, greatest;

    // Input the first integer

    printf("Enter the first integer: ");

    scanf("%d", &num1);

    // Input the second integer

    printf("Enter the second integer: ");

    scanf("%d", &num2);

    // Input the third integer

    printf("Enter the third integer: ");

    scanf("%d", &num3);

    // Assume the first integer is the greatest initially

    greatest = num1;

    // Compare num2 with the current greatest

    if (num2 > greatest) {

        greatest = num2;

    }

    // Compare num3 with the current greatest

    if (num3 > greatest) {

        greatest = num3;

    }

}
```

**// Display the greatest integer**

```
    printf("The greatest integer among %d, %d, and %d is %d\n", num1, num2,  
num3, greatest);  
    return 0; }
```

## **OUTPUT**

Enter the first integer: 7

Enter the second integer: 2

Enter the third integer: 9

The greatest integer among 7, 2, and 9 is 9

**1. C Program to find Mechanical Energy of a particle using  $E = mgh + \frac{1}{2}mv^2$ .**

Here's a C program that calculates the mechanical energy of a particle using the formula:  $E = mgh + \frac{1}{2}mv^2$ , where "m" is the mass of the particle, "g" is the acceleration due to gravity, "h" is the height, "v" is the velocity.

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

```
int main() {
```

```
    double mass, height, velocity, gravity, energy;
```

```
    // Input the mass of the particle (in kilograms)
```

```
    printf("Enter the mass of the particle (in kilograms): ");
```

```
    scanf("%lf", &mass);
```

```
    // Input the height (in meters)
```

```
    printf("Enter the height (in meters): ");
```

```
    scanf("%lf", &height);
```

```
    // Input the velocity (in meters per second)
```

```
    printf("Enter the velocity (in meters per second): ");
```

```
    scanf("%lf", &velocity);
```

```
    // Input the acceleration due to gravity (in meters per second squared)
```

```
    printf("Enter the acceleration due to gravity (in meters per second squared): ");
```

```
    scanf("%lf", &gravity);
```

**// Calculate mechanical energy using the formula**

```
energy = (mass * gravity * height) + (0.5 * mass * velocity * velocity);
```

**// Display the mechanical energy**

```
printf("The mechanical energy of the particle is %.2lf joules.\n", energy);
```

```
return 0;
```

```
}
```

## OUTPUT

Enter the mass of the particle (in kilograms): 2.5

Enter the height (in meters): 10.0

Enter the velocity (in meters per second): 4.0

Enter the acceleration due to gravity (in meters per second squared): 9.81

The mechanical energy of the particle is 1202.25 joules.

## 2. C Program to convert Kilometers into Meters, Centimeters and Millimeters.

Converting kilometers into meters, centimeters, and millimeters is a common task in measurements and calculations.

- Meters (m): 1 kilometer is equivalent to 1000 meters. To convert kilometers to meters, you multiply the number of kilometers by 1000.
- Centimeters (cm): 1 kilometer is equivalent to 100,000 centimeters. To convert kilometers to centimeters, you multiply the number of kilometers by 100,000.
- Millimeters (mm): 1 kilometer is equivalent to 1,000,000 millimeters. To convert kilometers to millimeters, you multiply the number of kilometers by 1,000,000.

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

```
int main() {
```

```
    double kilometers;
```

```
    // Input the distance in kilometers
```

```
    printf("Enter distance in kilometers: ");
```

```
    scanf("%lf", &kilometers);
```

```
    // Conversion factors
```

```
    double meters = kilometers * 1000;
```

```
    double centimeters = kilometers * 100000;
```

```
    double millimeters = kilometers * 1000000;
```

```
// Display the converted distances

printf("%.2lf kilometers is equal to:\n", kilometers);

printf("%.2lf meters\n", meters);

printf("%.2lf centimeters\n", centimeters);

printf("%.2lf millimeters\n", millimeters);

return 0;

}
```

### **OUTPUT**

**Enter distance in kilometers: 5.75**

5.75 kilometers is equal to:

5750.00 meters

575000.00 centimeters

5750000.00 millimeters

### 3. C Program to Check the Given Character is Lowercase or Uppercase or Special Character.

```
#include <stdio.h>

int main() {
    char ch;

    // Input a character

    printf("Enter a character: ");
    scanf(" %c", &ch); // Note the space before %c to consume any leading
                        // whitespace

    // Check if the character is lowercase
    if (ch >= 'a' && ch <= 'z') {
        printf("%c is a lowercase character.\n", ch);
    }

    // Check if the character is uppercase
    else if (ch >= 'A' && ch <= 'Z') {
        printf("%c is an uppercase character.\n", ch);
    }

    // Check if the character is a special character
    else {
        printf("%c is a special character.\n", ch);
    } return 0;
}
```

## OUTPUT

Enter a character: a

a is a lowercase character.

Enter a character: B

B is an uppercase character.

Enter a character: \$

\$ is a special character.

Prof. Monisha R



4. Write c code Program to balance the given Chemical Equation values x, y, p, q of a simple chemical equation of the type: The task is to find the values of constants b1, b2, b3 such that the equation is balanced on both sides and it must be the reduced form with output

Balancing a chemical equation is a common problem in chemistry. To balance a simple chemical equation using C code, you need to implement an algorithm that takes the coefficients of the reactants and products and balances them. Here's a program to do that:

```
#include <stdio.h>

int main()

{

    int a, b, c, d; // coefficients for the reactants and products

    // Chemical equation: aA + bB -> cC + dD

    // Example: 2H2 + O2 -> 2H2O

    printf("Enter coefficients for the reactants (aA + bB): ");

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

    printf("Enter coefficients for the products (cC + dD): ");

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

```
// Check if the equation is balanced
if (a * c == b * d)
{
    printf("The chemical equation is balanced!\n");
}
else
{
    printf("The chemical equation is not balanced.\n");
}
return 0;
}
```

### OUTPUT

Enter coefficients for the reactants (aA + bB): 2 1

Enter coefficients for the products (cC + dD): 2 2

The chemical equation is not balanced!

Enter coefficients for the reactants (aA + bB): 2 1

Enter coefficients for the products (cC + dD): 2 1

The chemical equation is balanced

**5. Implement Matrix multiplication and validate the rules of multiplication.**

AIM: To read two matrices A(m x n ) and B(p x q) and Compute the product of A and B.

```
#include<stdio.h>

void main( )
{
    int a[10][10], b[10][10], c[10][10];
    int m, n, p, q, i, j, k ;
    printf("\n Enter the order of the matrix A :");
    scanf("%d%d", &m, &n);
    printf("\n Enter the order of the matrix B :");
    scanf("%d%d", &p, &q);
    if (n!=p)
    {
        printf("Matrix A & Matrix B cannot be multiplied! \n");
    }
    else
    {
        printf("\n Enter the elements of matrix A :\n");
        for (i=0; i<m; i++)
        {
            for (j=0; j<n; j++)
                scanf("%d", &a[i][j]);
        }
        printf("\n Enter the elements of matrix B : \n");
```

```
for (i=0; i<p; i++)
{
for (j=0; j<q; j++)
scanf("%d", &b[i][j]);
}

/*Multiplication Process*/

for (i=0; i<m; i++)
{
for (j=0; j<q; j++)
{
c[i][j] = 0;
for (k=0; k<n; k++)
c[i][j] = c[i][j] + a[i][k] * b[k][j];
}
}

printf("\n Matrix A: \n");
for (i=0; i<m; i++)
{
for (j=0; j<n; j++)
{
printf(" %d \t", a[i][j]);
}
printf("\n");
}

printf("\n Matrix B: \n");
```

```
for (i=0; i<p; i++)
{
for (j=0; j<q; j++)
{
printf(" %d\t", b[i][j]);

}

printf("\n");

}

printf("\n The product of Matrix A and Matrix B is :\n");

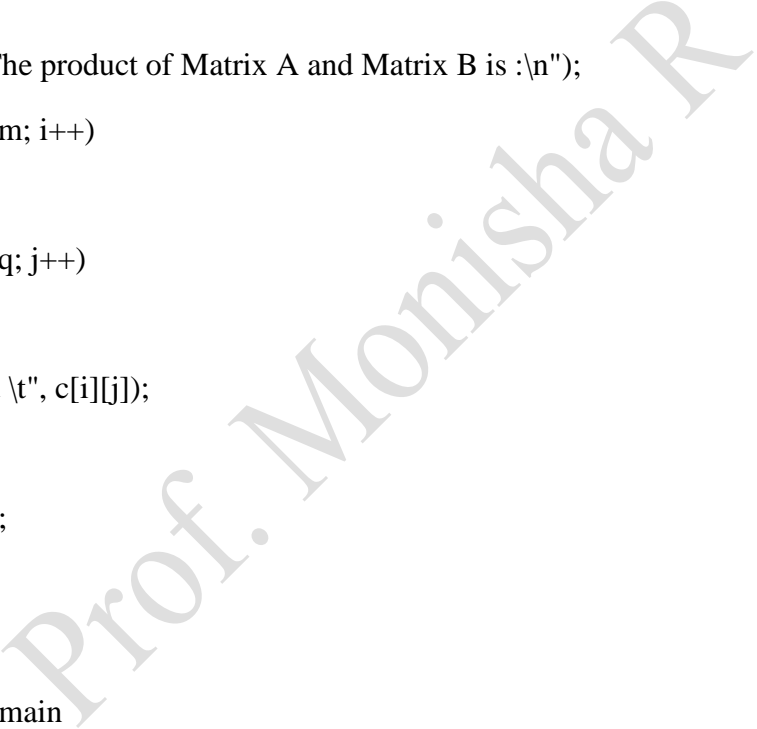
for (i=0; i<m; i++)
{
for (j=0; j<q; j++)
{
printf(" %d \t", c[i][j]);

}

printf("\n");

}

} //End of main
```



**OUTPUT**

1 Enter the order of the matrix A :

2

2

Enter the order of the matrix B :

1

2

Matrix A & Matrix B cannot be multiplied!

2 Enter the order of the matrix A :

2

2

Enter the order of the matrix B :

2

2

Enter the elements of matrix A :

1

2

3

4

Enter the elements of matrix B :

5

6

7

8

Matrix A is:

1 2

3 4

Matrix B is:

5 6

7 8

The Product of Matrix A and Matrix B is:

19 22

43 50

- 6. Compute  $\sin(x)/\cos(x)$  using Taylor series approximation. Compare your result with the built-in library function. Print both the results with appropriate inferences.**

**Taylor series** is the polynomial or a function of an infinite sum of terms. Each successive term will have a larger exponent or higher degree than the preceding term.

```
#include<stdio.h>

#include<math.h>

void main( )

{

float x, num, deno, term, sum;

int deg, i;

printf("Enter the value of degree:\n");

scanf("%d", &deg);

x = deg * (3.1412 / 180);

num = x;

deno = 1;

term = sum = x;

for (i=2; fabs(term) > 0.00001; i = i+2)

{

num = -- num * x * x;

deno = deno * i * (i + 1);

term = num / deno;

sum = sum + term;

}
```



```
printf("\n The sin(%d) = %f using Taylors series\n", deg, sum);  
printf("\n The sin(%d) = %f using math function", deg, sin(x));  
} //End of main
```

## OUTPUT

1. Enter the value of degree:

30

The sin(30) = 0.499943 using Taylors series

The sin(30) = 0.499943 using math function

2. Enter the value of degree:

60

The sin(60) = 0.865960 using Taylors series

The sin(60) = 0.865960 using math function

3. Enter the value of degree:

90

The sin(90) = 1.000000 using Taylors series

The sin(90) = 1.000000 using math function

**7. Write a C code Sort the given set of N numbers using Bubble sort**

Bubble Sort Algorithm is used to arrange N elements in ascending order, and for that, you have to begin with 0th element and compare it with the first element. If the 0th element is found greater than the 1st element, then the swapping operation will be performed, i.e., the two values will get interchanged.

```
#include<stdio.h>

#include<stdlib.h>

void main( )
{
    int n, i, j, a[10], temp;
    printf("\n Enter the number of elements: \n");
    scanf("%d", &n);
    printf("\n Enter %d elements:\n", n);
    for(i=0; i<n; i++)
        scanf("%d", &a[i]);
    printf("\n The Given array is:\n");
    for(i=0; i<n; i++)
        printf("%d\n", a[i]);
    for(j=1; j<n ; j++)
    {
        for(i=0; i<n-j; i++)
        {
            if(a[i] > a[i+1])
```

```
{  
temp = a[i];  
a[i] = a[i+1];  
a[i+1] = temp;  
}  
}  
}  
printf("\n The sorted array is:\n ");  
for (i=0; i<n; i++)  
    printf("%d\n", a[i]);  
} //End of main
```

Prof. Monisha R

**OUTPUT**

Enter the number of elements:

5

Enter 5 elements:

3

1

0

-1

2

The Given array is:

3

1

0

-1

2

The sorted array is:

-1

0

1

2

3

- 8. Write functions to implement string operations such as compare, concatenate, string length, convince the parameter passing techniques.**

```
#include<stdio.h>

#include<string.h>

void stringlength(char a[100], char b[100]);

void concatenate(char a[100], char b[100]);

void stringcompare(char a[100], char b[100]);

void main()
{
char p[100], q[100], ch[100];
int len1, len2;
printf("\n Enter the first string: ");
gets(p);
printf("\n Enter the second string: ");
gets(q);
stringlength(p,q);
stringcompare(p,q);
concatenate(p,q);
}

void stringlength(char a[100], char b[100])
{
int len1, len2;

len1 = strlen(a);
```

```
len2 = strlen(b);

printf("\n First string length is :%d \n Second string length is:%d", len1, len2);

}

void concatenate(char a[100], char b[100])
{
printf("\n The concatenated String is :%s ", strcat(a,b));
}

void stringcompare(char a[100], char b[100])
{
if(strcmp(a,b) == 0)
printf("\n STRINGS ARE EQUAL\n");
else
    printf("\n STRINGS ARE NOT EQUAL\n");
```

### OUTPUT

- |  |   |
|--|---|
| 1. Enter the first string: Hello       | 2. Enter the first string: Hello          |
| Enter the second string: World         | Enter the second string: Hello            |
| First string length is: 5              | First string length is: 5                 |
| Second string length is: 5             | Second string length is: 5                |
| STRINGS ARE NOT EQUAL                  | STRINGS ARE EQUAL                         |
| The concatenated string is: HelloWorld | The concatenated string is:<br>HelloHello |

- 9. Implement structures to read, write and compute average marks and the students scoring above and below the average marks for a class of N students.**

```
#include<stdio.h>

struct stud

{

char sname[30];

char sUSN[11];

int marks;

} s[20];

void main( )

{

int num, i;

float sum = 0.0, avg;

printf("\n Enter the number of students : ");

scanf("%d", &num);

printf("\n Enter the Details of students: \n");

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

{

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

printf("\n Name : ");

scanf("%s", s[i].sname);

printf("\n USN : ");
```

```
scanf("%s", s[i].sUSN);

printf("\n Marks : ");

scanf("%d", &s[i].marks);

sum = sum + s[i].marks;

}

avg = sum / num;

printf("\nThe average marks for the class is : %f \n", avg);

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

{

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

printf("\n Name\t: %s", s[i].sname);

printf("\n USN\t: %s", s[i].sUSN);

printf("\n Marks\t: %d", s[i].marks);

if (s[i].marks < avg)

printf("\n The student has scored below average marks:!\n");

else

printf("\n The student has scored above average marks !\n");

}

} //End of main
```



**OUTPUT**

1. Enter the number of students: 3

Enter the Details of students:

-----

Name : Manukumar

USN : 4SM15CS001

Marks : 20

-----

Name : Ravikumar

USN : 4SM15CS002

Marks : 15

-----

Name : Manju

USN : 4SM15CS003

Marks : 25

The average marks for the class is : 20

-----

Name : Manukumar

USN : 4SM15CS001

Marks : 20

The student has scored above average!

-----

Name : Ravikumar

USN : 4SM15CS002

Marks : 15

The student has scored below average!

-----

Name : Manju

USN : 4SM15CS003

Marks : 25

The student has scored above average!

Prof. Monisha R

**10. Develop a program using pointers to compute the sum, mean and standard deviation of all elements stored in an array of N real numbers.**

```
#include<stdio.h>

#include<math.h>

void main( )

{

float a[10], *ptr, mean, std, sum=0, sumstd=0;

int n, i;

printf("Enter the no of elements: \n");

scanf("%d", &n);

printf("Enter the array elements: \n");

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

scanf("%f", &a[i]);

ptr = a;

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

{

sum = sum + *ptr;

ptr++;

}

mean = sum / n;

ptr = a;

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

{
```

```
sumstd = sumstd + pow((*ptr - mean), 2);  
ptr++;  
}  
std = sqrt(sumstd / n);  
printf("Sum = %f \t", sum);  
printf("Mean = %f \t", mean);  
printf("Standard deviation = %f \n", std);  
} //End of main
```

### OUTPUT

1. Enter the no of elements: 3

Enter the array elements:

10

20

30

Sum = 60.000000 Mean = 20.000000 Standard Deviation= 8.164966

2. Enter the no of elements: 5

Enter the array elements:

10.2

20.4

30.7

40.6

50.9

Sum =152.799988 Mean = 30.559998 Standard Deviation = 14.368661