- tags: [c, codes, college]
- Solving BPOPS103

---

title: "Solving BPOPS103."

date: 2023-05-26 00:00:00 -500

categories: [c, code, college]

# tags: [c, codes, college]

# Solving BPOPS103

1. Simple Calculator

```c
#include <stdio.h>

int main() {
    int num1, num2;
    char operator;

    printf("Enter the first number: ");
    scanf("%d", &num1);

    printf("Enter an operator (+, -, *, /): ");
    scanf(" %c", &operator);

    printf("Enter the second number: ");
    scanf("%d", &num2);

    if (operator == '+') {
        printf("%d + %d = %d\n", num1, num2, num1 + num2);
    } else if (operator == '-') {
        printf("%d - %d = %d\n", num1, num2, num1 - num2);
    } else if (operator == '*') {
        printf("%d * %d = %d\n", num1, num2, num1 * num2);
    } else if (operator == '/') {
        if (num2 != 0)
            printf("%d / %d = %.2f\n", num1, num2, (float)num1 / num2);
        else
            printf("Error: Division by zero is not allowed.\n");
    } else {
        printf("Error: Invalid operator.\n");
    }
```

```
    return 0;
}
```

2. Compute the roots of a quadratic equation by accepting the coefficients. Print appropriate messages

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

int main() {
  double a, b, c, d, root1, root2;
  printf("Input the value of a, b & c: : ");
  scanf("%lf %lf %lf", &a, &b, &c);
  d = b * b - 4 * a * c;

  if (d > 0) {
    root1 = (-b + sqrt(d)) / (2 * a);
    root2 = (-b - sqrt(d)) / (2 * a);
    printf("The roots of the equation are %.2f and %.2f.\n", root1, root2);
  } else if (d == 0) {
    root1 = root2 = -b / (2 * a);
    printf("The roots of the equation are both equal to %.2f.\n", root1);
  } else {
    printf("The roots of the equation are complex.\n");
  }
  return 0;
}
```

3. An electricity board charges the following rates for the use of electricity: for the first 200 units 80 paise per unit: for the next 100 units 90 paise per unit: beyond 300 units Rs 1 per unit. All users are charged a minimum of Rs. 100 as meter charge. If the total amount is more than Rs 400, then an additional surcharge of 15% of total amount is charged. Write a program to read the name of the user, number of units consumed and print out the charges.

```
#include <stdio.h>

int main() {
  char name[100];
  int units;
  float charge, surcharge;

  printf("Enter the name of the user: ");
  scanf("%s", name);
  printf("Enter the number of units consumed: ");
  scanf("%d", &units);

  if (units <= 200) {
```

```
    charge = units * 0.80;
  } else if (units <= 300) {
    charge = 200 * 0.80 + (units - 200) * 0.90;
  } else {
    charge = 200 * 0.80 + 100 * 0.90 + (units - 300) * 1.00;
  }

  charge += 100;
  if (charge > 400) {
    surcharge = charge * 0.15;
    charge += surcharge;
  }
  printf("The charges for %s are Rs %.2f\n", name, charge);
  return 0;
}
```

5. Write a C Program to display the following by reading the number of rows as input

```
 #include <stdio.h>

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

    printf("Enter the number of rows: ");
    scanf("%d", &rows);

    for (i = 1; i <= rows; i++) {

        for (j = i; j < rows; j++) {
            printf("   ");
        }
        for (j = 1; j <= i; j++) {
            printf("%2d ", j);
        }
        for (k = i - 1; k >= 1; k--) {
            printf("%2d ", k);
        }
        printf("\n");
    }
    return 0;
}
```

5. Binary Search function

```
#include <stdio.h>

int binarySearch(int arr[], int size, int target) {
    int left = 0;
    int right = size - 1;

    while (left <= right) {
```

```
        int mid = left + (right - left) / 2;

        if (arr[mid] == target)
            return mid;

        if (arr[mid] < target)
            left = mid + 1;
        else
            right = mid - 1;
    }
    return -1;
}

int main() {
    int size;
    printf("Enter the size of the array: ");
    scanf("%d", &size);

    int arr[size];
    printf("Enter the elements of the array in sorted order:\n");
    for (int i = 0; i < size; i++) {
        scanf("%d", &arr[i]);
    }

    int target;
    printf("Enter the target element: ");
    scanf("%d", &target);

    int index = binarySearch(arr, size, target);

    if (index != -1)
        printf("Element %d found at index %d.\n", target, index);
    else
        printf("Element %d not found in the array.\n", target);

    return 0;
}
```

6. Implement Matrix multiplication and validate the rules of multiplication

```
#include <stdio.h>

#define MAX_SIZE 10

void matrixMultiplication(int m1[][MAX_SIZE], int r1, int c1, int m2[][MAX_SIZE],
int r2, int c2, int res[][MAX_SIZE]) {
    if (c1 != r2) {
        printf("Error: Invalid matrix dimensions for multiplication.\n");
        return;
    }

    for (int i = 0; i < r1; i++) {
        for (int j = 0; j < c2; j++) {
            res[i][j] = 0;
```

```
            for (int k = 0; k < c1; k++) {
                res[i][j] += m1[i][k] * m2[k][j];
            }
        }
    }
}

void displayMatrix(int matrix[][MAX_SIZE], int rows, int cols) {
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            printf("%d\t", matrix[i][j]);
        }
        printf("\n");
    }
    printf("\n");
}

int main() {
    int m1[MAX_SIZE][MAX_SIZE], r1, c1;
    printf("Enter the number of rows for matrix 1: ");
    scanf("%d", &r1);
    printf("Enter the number of columns for matrix 1: ");
    scanf("%d", &c1);
    printf("Enter the elements of matrix 1:\n");
    for (int i = 0; i < r1; i++)
        for (int j = 0; j < c1; j++)
            scanf("%d", &m1[i][j]);

    int m2[MAX_SIZE][MAX_SIZE], r2, c2;
    printf("Enter the number of rows for matrix 2: ");
    scanf("%d", &r2);
    printf("Enter the number of columns for matrix 2: ");
    scanf("%d", &c2);
    printf("Enter the elements of matrix 2:\n");
    for (int i = 0; i < r2; i++)
        for (int j = 0; j < c2; j++)
            scanf("%d", &m2[i][j]);

    int res[MAX_SIZE][MAX_SIZE];
    matrixMultiplication(m1, r1, c1, m2, r2, c2, res);

    printf("\nMatrix 1:\n");
    displayMatrix(m1, r1, c1);
    printf("Matrix 2:\n");
    displayMatrix(m2, r2, c2);
    printf("Resultant Matrix:\n");
    displayMatrix(res, r1, c2);

    return 0;
}
```

7. 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.

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

double taylorApproximation(double x, int terms) {
    double result = 0.0;
    double numerator = x;
    double denominator = 1.0;
    double term = numerator / denominator;
    int sign = 1;

    for (int i = 1; i <= terms; i++) {
        result += sign * term;
        numerator *= x * x;
        denominator *= (2 * i) * (2 * i + 1);
        term = numerator / denominator;
        sign *= -1;
    }

    return result;
}

int main() {
    double x;
    int terms;

    printf("Enter the value of x in radians: ");
    scanf("%lf", &x);

    printf("Enter the number of terms to compute in the Taylor series: ");
    scanf("%d", &terms);

    double taylorResult = taylorApproximation(x, terms);
    double libraryResult = tan(x);

    printf("Approximation using Taylor series: %.6f\n", taylorResult);
    printf("Result using library function (tan(x)): %.6f\n", libraryResult);

    printf("\nComparison:\n");

    if (fabs(taylorResult - libraryResult) < 0.000001) {
        printf("The results are approximately equal.\n");
    } else {
        printf("The results are not equal.\n");
    }
    return 0;
}
```

8. Sort the given set of N numbers using Bubble sort.

```c
#include <stdio.h>
int main()
{
    int n, j, i, swap;
```

```
    printf("Enter number of elements\n");
    scanf("%d", &n);
    int array[n];
    printf("Enter %d integers\n", n);
    for (i = 0; i < n; i++)
    {
        scanf("%d", &array[i]);
    }
    for (i = 0; i < n - 1; i++)
    {
        for (j = 0; j < n - i - 1; j++)
        {
            if (array[j] > array[j + 1])
            {
                swap = array[j];
                array[j] = array[j + 1];
                array[j + 1] = swap;
            }
        }
    }

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

    for (i = 0; i < n; i++)
        printf("%d\n", array[i]);
    return 0;
}
```

9. Write functions to implement string operations such as compare, concatenate, and find string length. Use the parameter passing techniques.

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

int stringLength(const char *str) {
    return strlen(str);
}

int stringCompare(const char *str1, const char *str2) {
    return strcmp(str1, str2);
}

void stringConcatenate(const char *str1, const char *str2, char *result) {
    strcpy(result, str1);
    strcat(result, str2);
}

int main() {
    char str1[100], str2[100], result[200];

    printf("Enter the first string: ");
    scanf("%s", str1);

    printf("Enter the second string: ");
```

```
    scanf("%s", str2);

    int length1 = stringLength(str1);
    int length2 = stringLength(str2);
    int compareResult = stringCompare(str1, str2);

    printf("Length of first string: %d\n", length1);
    printf("Length of second string: %d\n", length2);

    if (compareResult < 0) {
        printf("The first string is lexicographically smaller than the second
string.\n");
    } else if (compareResult > 0) {
        printf("The first string is lexicographically larger than the second
string.\n");
    } else {
        printf("The first string is lexicographically equal to the second
string.\n");
    }

    stringConcatenate(str1, str2, result);
    printf("Concatenated string: %s\n", result);

    return 0;
}
```

or

```
#include <stdio.h>
#include <string.h>
int main()
{
    int length(char str[100]);
    int compare(char s1[100], char s2[100]);
    void concat(char s1[100], char s2[100]);
    int option, result;
    char str[100], s1[100], s2[100];

    do
    {
        printf("1.String length \n");
        printf("2.string comparision \n");
        printf("3.string concatenation \n");
        printf("4.quit \n");
        printf("enter your choice \n");
        scanf("%d", &option);
        switch (option)
        {
        case 1:
            printf("enter string \n");
            scanf("%s", &str);
            result = length(str);
            printf("the length of string= %d\n", result);
            break;
```

```
        case 2:
            printf("enter 1st string\n");
            scanf("%s", &s1);
            printf("enter 2nd string\n");
            scanf("%s", &s2);
            result = compare(s1, s2);
            if (result == 0)
            {
                printf("strings are equal \n");
            }
            else
            {
                printf("strings are not equal \n");
            }
                break;

        case 3:
            printf("enter two strings\n");
            scanf("%s%s", s1, s2);
            concat(s1, s2);
            printf("result=%s \n", s1);
            break;
        }
    } while (option <= 3);
    return 0;
}
int length(char str[100])
{
    int i = 0;
    while (str[i] != '\0')
        i++;
    return (i);
}
int compare(char s1[100], char s2[100])
{
    int i = 0;
    while (s1[i] != '\0')
    {
        if (s1[i] > s2[i])
            return (1);
        else if (s1[i] < s2[i])
            return (-1);
        i++;
    }
    return 0;
}
void concat(char s1[100], char s2[100])
{
    int i, j;
    i = 0;
    while (s1[i] != '\0')
        i++;
    for (j = 0; s2[j] != '\0'; i++, j++)
        s1[i] = s2[j];
    s1[i] = '\0';
}
```

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

```c
#include <stdio.h>

#define MAX_NAME_LENGTH 50

struct Student {
    char name[MAX_NAME_LENGTH];
    int rollNumber;
    float marks;
};

void readStudentData(struct Student *student) {
    printf("Enter student name: ");
    scanf("%s", student->name);

    printf("Enter student roll number: ");
    scanf("%d", &student->rollNumber);

    printf("Enter student marks: ");
    scanf("%f", &student->marks);
}

void writeStudentData(const struct Student *student) {
    printf("Name: %s\n", student->name);
    printf("Roll Number: %d\n", student->rollNumber);
    printf("Marks: %.2f\n", student->marks);
}

float computeAverageMarks(const struct Student *students, int numStudents) {
    float totalMarks = 0.0;

    for (int i = 0; i < numStudents; i++) {
        totalMarks += students[i].marks;
    }

    return totalMarks / numStudents;
}

void listStudentsAboveAverage(const struct Student *students, int numStudents,
float averageMarks) {
    printf("\nStudents scoring above average marks:\n");

    for (int i = 0; i < numStudents; i++) {
        if (students[i].marks > averageMarks) {
            writeStudentData(&students[i]);
        }
    }
}

void listStudentsBelowAverage(const struct Student *students, int numStudents,
float averageMarks) {
```

```
    printf("\nStudents scoring below average marks:\n");

    for (int i = 0; i < numStudents; i++) {
        if (students[i].marks < averageMarks) {
            writeStudentData(&students[i]);
        }
    }
}

int main() {
    int numStudents;

    printf("Enter the number of students: ");
    scanf("%d", &numStudents);

    struct Student students[numStudents];

    printf("\nEnter student details:\n");
    for (int i = 0; i < numStudents; i++) {
        printf("\nStudent %d:\n", i + 1);
        readStudentData(&students[i]);
    }

    float averageMarks = computeAverageMarks(students, numStudents);
    printf("\nAverage marks: %.2f\n", averageMarks);
    listStudentsAboveAverage(students, numStudents, averageMarks);
    listStudentsBelowAverage(students, numStudents, averageMarks);
    return 0;
}
```

11. 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 computeStatistics(const float *arr, int size, float *sum, float *mean, float
*stdDev) {
    *sum = *mean = *stdDev = 0.0;

    for (int i = 0; i < size; i++) {
        *sum += arr[i];
    }

    *mean = *sum / size;

    for (int i = 0; i < size; i++) {
        float deviation = arr[i] - *mean;
        *stdDev += deviation * deviation;
    }

    *stdDev = sqrt(*stdDev / size);
}
```

```c
int main() {
    int N;

    printf("Enter the number of elements: ");
    scanf("%d", &N);

    float arr[N];

    printf("Enter the elements:\n");
    for (int i = 0; i < N; i++) {
        scanf("%f", &arr[i]);
    }

    float sum, mean, stdDev;
    computeStatistics(arr, N, &sum, &mean, &stdDev);

    printf("Sum: %.2f\n", sum);
    printf("Mean: %.2f\n", mean);
    printf("Standard Deviation: %.2f\n", stdDev);

    return 0;
}
```

12. Write a C program to copy a text file to another, read both the input file name and target file name

```c
#include <stdio.h>

#define MAX_FILENAME_LENGTH 100
#define BUFFER_SIZE 1024

int main() {
    char inputFile[MAX_FILENAME_LENGTH];
    char outputFile[MAX_FILENAME_LENGTH];
    FILE *sourceFile, *targetFile;
    char buffer[BUFFER_SIZE];
    size_t bytesRead;

    printf("Enter the input file name: ");
    scanf("%s", inputFile);

    printf("Enter the target file name: ");
    scanf("%s", outputFile);

    sourceFile = fopen(inputFile, "r");
    targetFile = fopen(outputFile, "w");

    if (sourceFile == NULL || targetFile == NULL) {
        printf("Failed to open the files.\n");
        return 1;
    }

    while ((bytesRead = fread(buffer, sizeof(char), BUFFER_SIZE, sourceFile)) > 0)
{
```

```
            fwrite(buffer, sizeof(char), bytesRead, targetFile);
        }

        fclose(sourceFile);
        fclose(targetFile);

        printf("File copied successfully.\n");

        return 0;
    }
```