|  |  |  |
| --- | --- | --- |
| ***Sl.No*** | ***Program*** | ***Page No*** |
| 1 | Write a C program using switch statement to design a basic calculator that performs the basic operations such as addition, subtraction, multiplication, and division. | 2 |
| 2 | Write a C program to find the coefficients of a quadratic equation and compute its roots. | 5 |
| 3 | Write a C program to find GCD and LCM of 2 integers. | 8 |
| 4 | The Fibonacci sequence is a sequence where the next term is the sum of the previous two terms. The first two terms of the Fibonacci sequence are 0 followed by 1. Read an integer Value: N from the user and print the Fibonacci series of ‘N’ terms | 10 |
| 5 | Write a C program to check if a given number is a Palindrome. | 12 |
| 6 | Write a C program to print pascal’s triangle based on number of rows. | 14 |
| 7 | Consider student’s marks in Computer Test. Write a C Program to display the grade obtained by a student in Computer Test based on range of marks. | 16 |
| 8 | Write a C program to read the size of a one-dimensional array from the user and read 10 elements into the array (positive integers) and  1. print alternate numbers starting from index value 1.  2. Reverse the array elements and print the same. | 18 |
| 9 | In computer based applications, matrices play a vital role in the projection of three dimensional image into a two dimensional screen, creating the realistic seeming motions. Write a C program using 2-dimensional array to check for compatibility of two matrices and perform matrix Multiplication. | 21 |
| 10 | Write a C program to implement bubble sort with appropriate input and output. | 26 |
| 11 | Write a C program to implement binary search with appropriate input and output. | 30 |
| 12 | Write a C program to read 2 strings from the user and perform concatenation | 34 |
| 13 | Write a C program to define a structure named Student with name and DOB, where, DOB in turn is a structure with day, month and year. Using the concept of nested structures display your name and date of birth. | 39 |

1. **Write a C program using switch statement to design a basic calculator that performs the basic operations such as addition, subtraction, multiplication, and division.**

***Algorithm***

**Read input:**

- Prompt the user to enter the first number (num1).

- Prompt the user to enter the operation code (operator).

- Prompt the user to enter the second number (num2).

**Perform the operation using a switch statement:**

- Use a switch statement with `operator` as the controlling expression.

- For each case (+, -, \*, /):

- Calculate the result based on the selected operation.

- Display the result.

**Handle division by zero:**

- For the division operation:

- Check if `num2` is not equal to zero.

- If true, perform the division and display the result.

- If false, display an error message for division by zero.

***Program:***

#include <stdio.h>

int main()

{

float num1, num2, result;

char operator;

printf("Enter two integer numbers: \n");

scanf("%f%f", &num1, &num2);

printf("Enter the operation (+, -, \*, /): ");

scanf(" %c", &operator);

switch (operator)

{

case '+':

result = num1 + num2;

printf("Sum: %f\n", result);

break;

case '-':

result = num1 - num2;

printf("Difference: %f\n", result);

break;

case '\*':

result = num1 \* num2;

printf("Product: %.2f\n", result);

break;

case '/':

// Check for division by zero

if (num2 != 0)

{

result = num1 / num2;

printf("Result: %.2f\n", result);

}

else

printf("Division by zero is not allowed.\n");

break;

default:

printf("Invalid operator\n");

break;

}

return 0;

}

***Output:***

Enter two integer numbers:

12

2

Enter the operation (+, -, \*, /): +

Sum: 14.000000

Enter two integer numbers:

10

0

Enter the operation (+, -, \*, /): /

Division by zero is not allowed.

Enter two integer numbers:

12

4

Enter the operation (+, -, \*, /): &

Invalid operator

1. **Write a C program to find the coefficients of a quadratic equation and compute its roots.**

***Algorithm:***

**Read coefficients from the user.**

- Prompt the user to enter coefficients `a`, ‘b’ and ‘c’

**Calculate discriminant.**

- Calculate the discriminant using the formula ‘discriminant = b2- 4ac`.

**Check discriminant.**

- If `discriminant` is greater than 0:

- Calculate real and different roots using the quadratic formula:

root1 = (-b + sqrt(discriminant)) / (2 \* a);

root2 = (-b - sqrt(discriminant)) / (2 \* a);

- Display the roots.

- If `discriminant` is equal to 0:

- Calculate real and equal roots using the quadratic formula:

root1 = -b / (2 \* a);

- Display the roots.

- If `discriminant` is less than 0:

- Calculate complex and different roots using the quadratic formula:

realPart = -b / (2 \* a);

imaginaryPart = sqrt(-discriminant) / (2 \* a);

- Display the roots.

***Program:***

#include <stdio.h>

#include <math.h>

int main()

{

float a, b, c, discriminant, root1, root2;

printf("Enter coefficients a, b, and c: ");

scanf("%f%f%f", &a, &b, &c);

discriminant = b \* b - 4 \* a \* c;

if (discriminant > 0)

{

// Roots are real and different

root1 = (-b + sqrt(discriminant)) / (2 \* a);

root2 = (-b - sqrt(discriminant)) / (2 \* a);

printf("Roots are real and different.\n");

printf("Root 1 = %.2f\n", root1);

printf("Root 2 = %.2f\n", root2);

} else if (discriminant == 0)

{

// Roots are real and equal

root1 = -b / (2 \* a);

printf("Roots are real and equal.\n");

printf("Root 1 = Root 2 = %.2f\n", root1);

} else {

// Roots are complex and different

float realPart = -b / (2 \* a);

float imaginaryPart = sqrt(-discriminant) / (2 \* a);

printf("Roots are complex and different.\n");

printf("Root 1 = %.2f + %.2fi\n", realPart, imaginaryPart);

printf("Root 2 = %.2f - %.2fi\n", realPart, imaginaryPart);

}

return 0;

}

***Output:***

Enter coefficients a, b, and c: 1 4 1

Roots are real and different.

Root 1 = -0.27

Root 2 = -3.73

Enter coefficients a, b, and c: 2 4 2

Roots are real and equal.

Root 1 = Root 2 = -1.00

Enter coefficients a, b, and c: 2 2 3

Roots are complex and different.

Root 1 = -0.50 + 1.12i

Root 2 = -0.50 - 1.12i

1. **Write a C program to find GCD and LCM of 2 integers.**

***Algorithm***

**Read input:**

Prompt the user to enter the first integer (num1), Second integer (num2).

**Calculate the GCD:**

Set a to the value of num1 and b to the value of num2.

Use a while loop to perform the Euclidean algorithm until b becomes 0.

After the loop, the GCD is stored in gcd.

**Calculate LCM using GCD:**

Calculate the LCM using the relationship: LCM = (num1 \* num2) / gcd.

Display the GCD and LCM.

***Program:***

#include <stdio.h>

int main()

{

int num1, num2, lcm, gcd;

printf("Enter two integer numbers: \n");

scanf("%d%d", &num1, &num2);

int a = num1, b = num2;

while (b != 0)

{

int temp = b;

b = a % b;

a = temp;

}

gcd = a;

// Find LCM using GCD

lcm = (num1 \* num2) / gcd;

// Display GCD and LCM

printf("GCD of %d and %d is: %d\n", num1, num2, gcd);

printf("LCM of %d and %d is: %d\n", num1, num2, lcm);

return 0;

}

***Output:***

Enter two integer numbers:

12

4

GCD of 12 and 4 is: 4

LCM of 12 and 4 is: 12

Enter two integer numbers:

5

20

GCD of 5 and 20 is: 5

LCM of 5 and 20 is: 20

1. **The Fibonacci sequence is a sequence where the next term is the sum of the previous two terms. The first two terms of the Fibonacci sequence are 0 followed by 1. Read an integer Value: N from the user and print the Fibonacci series of ‘N’ terms**

***Algorithm:***

Read the value of N from the user.

Initialize variables first and second to 0 and 1, respectively.

Print "Fibonacci Series of first N numbers:" where N is the user-input value.

Repeat N times:

a. Print the value of first.

b. Calculate the next Fibonacci number: next = first + second.

c. Update first with the value of second.

d. Update second with the value of next.

***Program:***

#include <stdio.h>

int main()

{

int n, first = 0, second = 1, next;

printf("Enter the number of terms for Fibonacci series: \n");

scanf("%d", &n);

// Check for valid input

if (n <= 0)

{

printf("Please enter a positive integer for the number of terms.\n");

return 1; // Return an error code

}

// Print Fibonacci series

printf("Fibonacci Series of first %d numbers: \n", n);

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

{

printf("%d, ", first);

next = first + second;

first = second;

second = next;

}

return 0;

}

***Output:***

Enter the number of terms for Fibonacci series:

7

Fibonacci Series of first 7 numbers:

0, 1, 1, 2, 3, 5, 8,

Enter the number of terms for Fibonacci series:

-5

Please enter a positive integer for the number of terms.

1. **Write a C program to check if a given number is a Palindrome.**

***Algorithm:***

Read the input number from the user and store it in a variable (num).

Initialize variables originalNum to store the original number and reversedNum to store the reversed number to 0.

Save the original number by setting originalNum equal to num.

Repeat until num is not equal to 0:

a. Calculate the remainder (remainder) when num is divided by 10.

b. Multiply reversedNum by 10 and add the remainder to it.

c. Divide num by 10.

Check if originalNum is equal to reversedNum.

a. If true, print that the number is a palindrome.

b. If false, print that the number is not a palindrome.

***Program:***

#include <stdio.h>

int main()

{

int num, originalNum, reversedNum = 0, remainder;

printf("Enter an integer: \n");

scanf("%d", &num);

// Save the original number

originalNum = num;

// Reverse the number

while (num != 0)

{

remainder = num % 10;

reversedNum = reversedNum \* 10 + remainder;

num /= 10;

}

// Check if the original number is equal to its reverse

if (originalNum == reversedNum)

printf("%d is a palindrome.\n", originalNum);

else

printf("%d is not a palindrome.\n", originalNum);

return 0;

}

***Output:***

Enter an integer:

1234

1234 is not a palindrome.

Enter an integer:

1221

1221 is a palindrome.

1. **Write a C program to print pascal’s triangle based on number of rows.**

***Algorithm:***

Read number of rows

Use a loop to iterate over each line from 1 to `n`.

For each line, use another loop to print the required spaces before the coefficients.

- Initialize a variable `coef` to 1, representing the binomial coefficient C(line, i).

- Use another loop to calculate and print the coefficients for the current line.

- The first value in each line is always 1.

- Update the coefficient for the next iteration using the formula `coef = coef \* (line - i) / i`.

Print a newline after each line to separate them.

***Program:***

#include <stdio.h>

int main ()

{

int n;

printf ("Enter number of rows: \n");

scanf ("%d", &n);

for (int line = 1; line <= n; line++) {

for (int space = 1; space <= n - line; space++)

printf(" ");

// used to represent C(line, i)

int coef = 1;

for (int i = 1; i <= line; i++) {

// The first value in a line

// is always 1

printf("%4d", coef);

coef = coef \* (line - i) / i;

}

printf("\n");

}

}

***Output:***

Enter number of rows:

5

1

1 1

1 2 1

1 3 3 1

1 4 6 4 1

Enter number of rows:

7

1

1 1

1 2 1

1 3 3 1

1 4 6 4 1

1 5 10 10 5 1

1 6 15 20 15 6 1

1. **Consider student’s marks in Computer Test. Write a C Program to display the grade obtained by a student in Computer Test based on range of marks.**

***Algorithm:***

Accept the user's input for the grade.

Check the value of grade using a series of if-else statements:

If grade is equal to 10, print "Grade: O, Performance: Outstanding."

Else, if grade is greater than or equal to 9, print "Grade: A+, Performance: Excellent."

Else, if grade is greater than or equal to 8, print "Grade: A, Performance: Very Good."

Else, if grade is greater than or equal to 7, print "Grade: B+, Performance: Good."

Else, if grade is greater than or equal to 6, print "Grade: B, Performance: Above Average."

Else, if grade is greater than or equal to 5.5, print "Grade: C+, Performance: Average."

Else, if grade is greater than or equal to 5, print "Grade: C, Performance: Satisfactory."

Else, print "Grade: F, Performance: Fail."

***Program:***

# include <stdio.h>

int main ()

{

float grade;

printf ("Enter your grade (0 to 10)\n");

scanf ("%f", &grade);

if (grade == 10)

printf ("Grade:O, Performance: Outstanding");

else if (grade >= 9)

printf ("Grade:A+, Performance: Excellent");

else if (grade >= 8)

printf ("Grade:A, Performance: Very Good");

else if (grade >= 7)

printf ("Grade:B+, Performance: Good");

else if (grade >= 6)

printf ("Grade:B, Performance: Above Average");

else if (grade >= 5.5)

printf ("Grade:C+, Performance: Average");

else if (grade >= 5)

printf ("Grade:C, Performance: Satisfactory");

else

printf ("Grade:F, Performance: Fail");

}

***Output:***

Enter your grade (0 to 10)

7

Grade:B+, Performance: Good

Enter your grade (0 to 10)

1.2

Grade:F, Performance: Fail

1. **Write a C program to read the size of a one-dimensional array from the user and read 10 elements into the array (positive integers) and**

**1. print alternate numbers starting from index value 1.**

**2. Reverse the array elements and print the same.**

***Algorithm:***

Read the size of the array from the user:

Ensure the size is at least 10:

Check if the entered size is less than 10.

If it is less than 10, print an error message and exit the program with an error code.

Declare an array of the specified size:

Read 10 positive integers into the array:

Print alternate numbers starting from index 1:

Use a loop to iterate over the array starting from index 1 with a step of 2.

Print each element at the current index.

Reverse the array elements and print:

Use a loop to iterate over the array in reverse order.

Print each element as it is traversed in reverse.

***Program:***

#include <stdio.h>

int main()

{

int size;

// Read the size of the array from the user

printf("Enter the size of the array: \n");

scanf("%d", &size);

// Ensure that the size is at least 10

if (size < 10)

{

printf("Please enter a size of at least 10.\n");

return 1; // Exit with an error code

}

int array[size];

// Read 10 elements into the array

printf("Enter 10 positive integers:\n");

for (int i = 0; i < 10; ++i)

scanf("%d", &array[i]);

// 1. Print alternate numbers starting from index value 1

printf("\nAlternate numbers starting from index 1:\n");

for (int i = 1; i < size; i += 2)

printf("%d ", array[i]);

printf("\n");

// 2. Reverse the array elements and print

printf("\nReversed array elements:\n");

for (int i = size - 1; i >= 0; --i)

printf("%d ", array[i]);

printf("\n");

return 0; // Exit successfully

}

***Output:***

Enter the size of the array:

5

Please enter a size of at least 10.

Enter the size of the array: 10

Enter 10 positive integers:

1 2 3 4 5 6 7 8 9 10

Alternate numbers starting from index 1:

2 4 6 8 10

Reversed array elements:

10 9 8 7 6 5 4 3 2 1

1. **In computer based applications, matrices play a vital role in the projection of three dimensional image into a two dimensional screen, creating the realistic seeming motions. Write a C program using 2-dimensional array to check for compatibility of two matrices and perform matrix Multiplication.**

***Algorithm:***

Input dimensions of the first matrix (matrix A):

Read the number of rows and columns for matrix A from the user.

Input dimensions of the second matrix (matrix B):

Read the number of rows and columns for matrix B from the user.

Check for compatibility for matrix multiplication:

If the number of columns in matrix A is not equal to the number of rows in matrix B, print an error message and exit with an error code.

Input elements of matrix A.

Input elements of matrix B.

Perform matrix multiplication.

Print matrix A, matrix B, and the result matrix.

Return 0 to indicate successful execution.

***Program:***

#include <stdio.h>

int main() {

int rowsA, colsA, rowsB, colsB;

// Input dimensions of the first matrix

printf("Enter the number of rows for matrix A: ");

scanf("%d", &rowsA);

printf("Enter the number of columns for matrix A: ");

scanf("%d", &colsA);

// Input dimensions of the second matrix

printf("Enter the number of rows for matrix B: ");

scanf("%d", &rowsB);

printf("Enter the number of columns for matrix B: ");

scanf("%d", &colsB);

// Check for compatibility for matrix multiplication

if (colsA != rowsB) {

printf("Matrix multiplication is not possible due to incompatible dimensions.\n");

return 1; // Exit with an error code

}

int matrixA[rowsA][colsA];

int matrixB[rowsB][colsB];

int result[rowsA][colsB];

// Input elements of the matrices A and B

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

for (int i = 0; i < rowsA; ++i) {

for (int j = 0; j < colsA; ++j) {

printf("Enter element at position (%d, %d): ", i + 1, j + 1);

scanf("%d", &matrixA[i][j]);

}

}

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

for (int i = 0; i < rowsB; ++i) {

for (int j = 0; j < colsB; ++j) {

printf("Enter element at position (%d, %d): ", i + 1, j + 1);

scanf("%d", &matrixB[i][j]);

}

}

// Perform matrix multiplication

for (int i = 0; i < rowsA; ++i) {

for (int j = 0; j < colsB; ++j) {

result[i][j] = 0;

for (int k = 0; k < colsA; ++k) {

result[i][j] += matrixA[i][k] \* matrixB[k][j];

}

}

}

// Display matrices and the result

printf("\nMatrix A:\n");

for (int i = 0; i < rowsA; ++i) {

for (int j = 0; j < colsA; ++j) {

printf("%d\t", matrixA[i][j]);

}

printf("\n");

}

printf("\nMatrix B:\n");

for (int i = 0; i < rowsB; ++i) {

for (int j = 0; j < colsB; ++j) {

printf("%d\t", matrixB[i][j]);

}

printf("\n");

}

printf("\nResultant Matrix (A \* B):\n");

for (int i = 0; i < rowsA; ++i) {

for (int j = 0; j < colsB; ++j) {

printf("%d\t", result[i][j]);

}

printf("\n");

}

return 0; // Exit successfully

}

***Output:***

Enter the number of rows for matrix A: 2

Enter the number of columns for matrix A: 2

Enter the number of rows for matrix B: 2

Enter the number of columns for matrix B: 2

Enter the elements of matrix A:

Enter element at position (1, 1): 1

Enter element at position (1, 2): 2

Enter element at position (2, 1): 3

Enter element at position (2, 2): 4

Enter the elements of matrix B:

Enter element at position (1, 1): 5

Enter element at position (1, 2): 6

Enter element at position (2, 1): 7

Enter element at position (2, 2): 8

Matrix A:

1 2

3 4

Matrix B:

5 6

7 8

Resultant Matrix (A \* B):

19 22

43 50

Enter the number of rows for matrix A: 2

Enter the number of columns for matrix A: 3

Enter the number of rows for matrix B: 2

Enter the number of columns for matrix B: 3

Matrix multiplication is not possible due to incompatible dimensions.

1. **Write a C program to implement bubble sort with appropriate input and output.**

***Algorithm:***

Input the size of the array.

Input elements into the array.

Use nested loops to iterate over the array elements and perform the bubble sort algorithm:

Iterate over the array elements from the beginning to the second-to-last element.

Within the above loop, iterate over the array elements from the beginning to size - i - 2.

Swap adjacent elements if they are in the wrong order.

Display the sorted array:

***Program:***

#include <stdio.h>

// Function to perform bubble sort on an array

void bubbleSort(int arr[], int size)

{

for (int i = 0; i < size - 1; ++i)

{

for (int j = 0; j < size - i - 1; ++j)

{

// Swap if the element found is greater than the next element

if (arr[j] > arr[j + 1])

{

// Swap the elements using a temporary variable

int temp = arr[j];

arr[j] = arr[j + 1];

arr[j + 1] = temp;

}

}

}

}

// Function to input elements into an array

void inputArray(int arr[], int size)

{

printf("Enter %d elements:\n", size);

for (int i = 0; i < size; ++i)

scanf("%d", &arr[i]);

}

// Function to display elements of an array

void displayArray(int arr[], int size)

{

printf("Sorted Array:\n");

for (int i = 0; i < size; ++i)

printf("%d ", arr[i]);

printf("\n");

}

int main()

{

int size;

// Input the size of the array

printf("Enter the size of the array: ");

scanf("%d", &size);

int arr[size];

// Input elements into the array

inputArray(arr, size);

// Perform bubble sort

bubbleSort(arr, size);

// Display the sorted array

displayArray(arr, size);

return 0; // Exit successfully

}

***Output:***

Enter the size of the array: 10

Enter 10 elements:

12

21

43

32

11

56

54

87

76

65

Sorted Array:

11 12 21 32 43 54 56 65 76 87

Enter the size of the array: 5

Enter 5 elements:

67

65

54

43

32

Sorted Array:

32 43 54 65 67

1. **Write a C program to implement binary search with appropriate input and output.**

***Algorithm:***

Input the size of the array (size).

Input the sorted array elements into the array.

Input the element to be searched (key).

Call the binarySearch function with the array, low index as 0, high index as size - 1, and the key.

Initialize a while loop with the condition low <= high.

Calculate the middle index mid using the formula mid = low + (high - low) / 2.

Check if the element at index mid is equal to the key.

If yes, return mid (element found).

If no, check if the key is smaller than the element at index mid.

If yes, update high = mid - 1 (ignore the right half).

If no, update low = mid + 1 (ignore the left half).

If the while loop exits, return -1 (element not found).

***Program:***

#include <stdio.h>

// Function to perform binary search

int binarySearch(int arr[], int low, int high, int key) {

while (low <= high) {

int mid = low + (high - low) / 2;

// Check if the key is present at the middle

if (arr[mid] == key)

return mid;

// If the key is smaller, ignore the right half

else if (arr[mid] > key)

high = mid - 1;

// If the key is larger, ignore the left half

else

low = mid + 1;

}

// Key not present in the array

return -1;

}

int main() {

int size, key, result;

// Input the size of the array

printf("Enter the size of the array: ");

scanf("%d", &size);

int arr[size];

// Input array elements in sorted order

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

for (int i = 0; i < size; i++) {

scanf("%d", &arr[i]);

}

// Input the element to be searched

printf("Enter the element to be searched: ");

scanf("%d", &key);

// Perform binary search

result = binarySearch(arr, 0, size - 1, key);

// Output the result

if (result != -1)

printf("Element %d found at index %d\n", key, result+1);

else

printf("Element %d not found in the array\n", key);

return 0;

}

***Output:***

Enter the size of the array: 6

Enter the sorted array elements:

12

23

34

45

56

67

Enter the element to be searched: 56

Element 56 found at index 5

Enter the size of the array: 5

Enter the sorted array elements:

12

23

34

45

56

Enter the element to be searched: 78

Element 78 not found in the array

1. **Write a C program to read 2 strings from the user and perform the concatenation**

***Algorithm:***

Input the first string (str1).

Input the second string (str2).

Call **concatenateStrings** with str1, str2, and result.

Print the concatenated string.

**Function concatenateStrings (str1, str2, result)**

Copy the content of str1 to result using strcpy.

Concatenate the content of str2 to result using strcat.

***Program:***

#include <stdio.h>

#include <string.h>

// Function to concatenate two strings

void concatenateStrings(char str1[], char str2[], char result[])

{

strcpy(result, str1);

strcat(result, str2);

}

int main()

{

char str1[100], str2[100], result[200];

// Input two strings from the user

printf("Enter the first string: ");

gets(str1);

printf("Enter the second string: ");

gets(str2);

concatenateStrings(str1, str2, result);

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

return 0;

}

***Output:***

Enter the first string: REVA

Enter the second string: University

Concatenated string: REVAUniversity

1. **Write a C program to define a structure named Student with name and DOB, where, DOB in turn is a structure with day, month and year. Using the concept of nested structures display your name and date of birth.**

***Algorithm:***

Define a structure DOB to represent the Date of Birth with three members: day, month, and year.

Define another structure Student with two members: name (an array of characters) and birthDate (an instance of the DOB structure).

Use fgets to read the name input from the user and store it in the name member of the student structure.

Prompt the user to enter their date of birth (DD MM YYYY) using printf.

Use scanf to read the day, month, and year inputs from the user and store them in the corresponding members of the birthDate structure within the student structure.

Display the student's information

***Program:***

#include <stdio.h>

// Define the structure for Date of Birth (DOB)

struct DOB {

int day;

int month;

int year;

};

// Define the structure for Student

struct Student {

char name[50];

struct DOB birthDate;

};

int main() {

// Declare a variable of type Student

struct Student student;

// Input the student's details

printf("Enter your name: ");

fgets(student.name, sizeof(student.name), stdin);

printf("Enter your date of birth (DD MM YYYY): ");

scanf("%d%d%d",&student.birthDate.day,&student.birthDate.month, &student.birthDate.year);

// Display the information

printf("\nStudent Information:\n");

printf("Name: %s", student.name);

printf("Date of Birth: %02d/%02d/%d\n", student.birthDate.day, student.birthDate.month, student.birthDate.year);

return 0;

}

***Output:***

Enter your name: Praveen

Enter your date of birth (DD MM YYYY): 12 12 2004

Student Information:

Name: Praveen

Date of Birth: 12/12/2004