**C Programs**

**Program 1**

**Aim: to**  Check a number is Armstrong or not

**Pseudo code:**

* Accept a three-digit number as input.
* Initialize a variable **sum** to zero.
* Create a copy of the input number and store it in a variable **temp**.
* For each digit in the input number:
* a. Calculate the cube of the digit. b. Add the cube to the **sum**. c. Remove the last digit from the input number.
* Check if the calculated **sum** is equal to the original input number **temp**.
* If they are equal, the number is an Armstrong number.
* If they are not equal, the number is not an Armstrong number.

**Output:**





**Result:** Process finished with exit code 0.

**Program 2**

**Aim: impement a** Programfor bubble sort.

**algorithm:**

* Input:
* Create an integer array a to hold the elements.
* Read the number of elements (num) from the user.
* Read the array elements into vatiable a
* Repeat the following steps until no swaps are made in a pass:
* Set a boolean variable swapped to false.
* Iterate through the array from the first element to the second-to-last element:
* Compare a[i] with a[i+1].
* If a[i] is greater than a[i+1], swap them and set swapped to true.
* After each pass, the largest unsorted element will have bubbled up to the end of the array.
* Output:
* Print the sorted array in ascending order.

**Output:**



**Result:** Process finished with exit code 0.

**Program 3**

**Aim:** to implement a multioperational Calculator using case

**algorithm:**

// Declare variables

Integer n1, n2, sum, diff, multi, div

Character ch

// Input the first number

Output "Enter the first value: "

Input n1

// Input the second number

Output "Enter the second value: "

Input n2

// Display menu

Output "1. Sum"

Output "2. Difference"

Output "3. Multiplication"

Output "4. Division"

// Perform operations

For i = 0 to 3

Output "Enter the choice: "

Input ch

Switch ch

Case 1:

sum = n1 + n2

Output "Sum is =", sum

Case 2:

diff = n1 - n2

Output "Difference is =", diff

Case 3:

multi = n1 \* n2

Output "Multiplication is =", multi

Case 4:

div = n1 / n2

Output "Division is =", div

Default:

Output "Invalid statement"

End Switch

End For

**Output:**



**Result: :** Process finished with exit code 0.

**Program 4**

**Aim: to implement** Pascal’s triangle

**algorithm:**

// Initialize Pascal's Triangle with zeros

Create a 2D array T[5][5]

// Set the values in the first row (row 0) to 1

For column = 0 to 4:

T[0][column] = 1

// Generate Pascal's Triangle

For row = 1 to 4:

// Set the first and last column of the current row to 1

T[row][0] = 1

T[row][row] = 1

For column = 1 to row:

// Calculate the value in T[row][column] by adding values from the previous row

T[row][column] = T[row - 1][column - 1] + T[row - 1][column]

// Print Pascal's Triangle

For row = 0 to 4:

For column = 0 to row:

Output T[row][column] followed by a space

Output a newline to move to the next row

**Output:**



**Result:** Process finished with exit code 0.

**Program 5**

**Aim: for** Swapping two numbers using pointers.

**algorithm:**

Declare Integer a,b

// Prompt the user to input values for a and b

Output "Enter the first value (a): "

Input a

Output "Enter the second value (b): "

Input b

// Display the values of a and b before swapping

Output "Before swap:"

Output "a=" + a

Output "b=" + b

// Swap the values of a and b using a temporary variable

Declare Integer temp

temp = a

a = b

b = temp

// Display the values of a and b after swapping

Output "After swap:"

Output "a=" + a

Output "b=" + b

End Algorithm

// Call the main swapping algorithm

Call SwapTwoVariables

**Output:**



**Result:** Process finished with exit code 0.

**Program 6**

**Aim:** to check String is Palindrome or not**.**

**algorithm:**

// Input: a string str

// Output: true if str is a palindrome, false otherwise

// Initialize two pointers, left and right

left = 0

right = length of str – 1

// While left is less than right

while left < right

// If the characters at positions left and right are not the same

if str[left] is not equal to str[right]

return false // It's not a palindrome

// Move the left pointer to the right

left = left + 1

// Move the right pointer to the left

right = right - 1

// If the loop completes without returning false, it's a palindrome

return true

**Output:**

 

**Result:** Process finished with exit code 0.

**Program 7**

**Aim: for** Reading an input string and sort that input string**.**

**allgorithm:**

1. Initialize a character array 'str' with a size of 100.

2. Initialize a character variable 'temp'.

3. Initialize integer variables 'len', 'i', and 'j'.

4. Clear the screen (if needed).

5. Output a message: "Enter the string:"

6. Read a string from the user and store it in 'str'.

7. Calculate the length of the string and store it in 'len'.

8. Iterate through the characters in 'str' using a nested loop:

a. Outer loop 'i' from 0 to 'len - 1':

i. Inner loop 'j' from 0 to 'len - i - 1':

1. If 'str[j]' is greater than 'str[j+1]':

- Swap 'str[j]' and 'str[j+1]' by using 'temp'.

9. Output the sorted string: "String in alphabetic order: " followed by 'str'.

10. Wait for user input (if needed).

11. Exit the program.

**Output:**



**Result:** Process finished with exit code 0.

**Program 8**

**Aim: for** Swapping two numbers using call by reference.

**algorithm:**

1. Declare integer variables a and b and initialize them with values 5 and 6.

2. Call a function named clrscr() (assumed to clear the screen, not shown in the code).

3. Print the values of a and b before swapping: "before swap a=5, b=6".

4. Call a function named swap and pass the addresses of a and b as arguments.

5. Return 0 to indicate successful program execution.

Function swap(int \*m, int \*n):

1. Declare an integer variable named temp.

2. Assign the value pointed to by m to temp (temp = \*m).

3. Assign the value pointed to by n to the memory location pointed to by m (\*m = \*n).

4. Assign the value of temp to the memory location pointed to by n (\*n = temp).

5. Print the values of a and b after swapping: "after swap a=<new value of a>, b=<new value of b>".

6. Wait for a key press (getch() function, assumed to pause the program).

**Output:**



**Result:** Process finished with exit code 0.

**Program 9**

**Aim: for** Swapping two numbers using call by value.

**algorithm:**

1. Declare integer variables a and b and initialize them with values 5 and 6.

2. Print the values of a and b before swapping: "before swap a=5, b=6".

3. Call a function named swap and pass the values of a and b as arguments.

4. Print the values of a and b after swapping: "after swap a=<new value of a>, b=<new value of b>".

Function swap(int m, int n):

1. Declare an integer variable named temp.

2. Assign the value of m to temp (temp = m).

3. Assign the value of n to m (m = n).

4. Assign the value of temp to n (n = temp).

5. Print the values of m and n inside the swap function: "inside swap m=<new value of m>, n=<new value of n>".

**Output:**



**Result:** Process finished with exit code 0.

**Program 10**

**Aim: for** Using the structures to develop an employee payroll system.

Salary=basic pay+ da+ hra

Da=0.35% of basic salary

Hra=3% of basic salary

**algorithm:**

Declare a structure named "emp" with fields: name (string), hra (integer), da (integer), bp (integer), salary (integer), and id (integer).

Declare an integer variable i and n (not used in the code).

Clear the screen (not shown in the code).

Print "Enter the name:".

Read the name into the "name" field of the "emp" structure.

Print "Employee ID:".

Read the employee ID into the "id" field of the "emp" structure.

Print "Enter the basic pay:".

Read the basic pay into the "bp" field of the "emp" structure.

Calculate the dearness allowance (da) as 0.35% of the basic pay.

Calculate the house rent allowance (hra) as 3% of the basic pay.

Calculate the total salary as the sum of da, hra, and bp.

Print a line of dashes as a separator.

Print "Salary Details:".

Print "Name: " followed by the "name" field of the "emp" structure.

Print "Employee ID: " followed by the "id" field of the "emp" structure.

Print "Total Salary: " followed by the "salary" field of the "emp" structure.

**Output:**



**Program 11**

**Aim: To** Perform matrix multiplication

**algorithm:**

Declare a variable sum and initialize it to 0. This variable will store the sum of the numbers.

Declare a variable count and initialize it to 0. This variable will count the number of numbers entered.

Print "Enter the number of numbers to average:".

Read the number of numbers (n) from the user.

Initialize a loop counter i to 1.

While i is less than or equal to n, do the following:

Print "Enter number #" followed by i.

Read a number (num) from the user.

Add num to the sum.

Increment count by 1.

Increment i by 1.

Calculate the average by dividing sum by count and store it in a variable named average. Print "The average of the numbers is:" followed by the value of average.

**Output:**



**Result:** Process finished with exit code 0.