

Programs using Iteration Statements

1. Calculate the factorial of a given number n (for loop)

AIM: To write a C program to calculate the factorial of a given number n using a for loop.

ALGORITHM:

FLOWCHART:

PROGRAM :

```
#include <stdio.h> void main()
{
int i,f=1,num;
printf("Input the number : "); scanf("%d",&num); for(i=1;i<=num;i++)
f=f*i;
printf("The Factorial of %d is: %d\n",num,f);
}
```

OUTPUT:

Input the number : 5 The Factorial of 5 is: 120

2. Display the count of positive, negative and zero numbers from the given set of inputs. (while loop)

AIM: To display the count of positive, negative and zero numbers from the given set of inputs using while loop.

ALGORITHM:

FLOWCHART:

PROGRAM:

```
#include<stdio.h>
int main()
{
int num,p,n,z;
```

```

p=n=z=0;

int choice;

printf("Enter your choice: \n Press '1' to enter a new number \n Press '0' to quit entering\n");

scanf("%d",&choice);

while(choice==1)

{

printf("Enter a new number:");

scanf("%d",&num);

if(num>0)

++p;

if(num<0)

++n;

if(num==0)

++z;

printf("Enter your choice: \n Press '1' to enter a new number \n Press '0' to quit entering");

scanf("%d",&choice);

}

printf("Count of positive Numbers:%d\n",p);

printf("Count of Negative Numbers:%d\n",n);

printf("Count of Zeros:%d\n",z);

return(0);

}

```

OUTPUT:

Enter your choice:

Press '1' to enter a new number

Press '0' to quit entering

Enter a new number: 5

Enter your choice:

Press '1' to enter a new number

Press '0' to quit entering

1

Enter a new number:-8

Enter your choice:

Press '1' to enter a new number

Press '0' to quit entering

1

Enter a new number:0

Enter your choice:

Press '1' to enter a new number

Press '0' to quit entering

0

Count of positive Numbers:1

Count of Negative Numbers:1

Count of Zeros:1

RESULT: Thus the C program to display the count of positive, negative and zero numbers from the given set of inputs using while loop structure is executed successfully.

3. Display the square and cube of first N natural numbers. (do..while loop)

AIM: To display the square and cube of first N natural numbers using do..while loop structure.

ALGORITHM:

FLOWCHART:

PROGRAM:

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

```
int main()
```

```
{
```

```
    int i,num;
```

```
int sq,cube;

i=1;

printf("Enter a number:");

scanf("%d",&num);

do

{

    sq=i*i;

    printf("Square of %d is %d\n",i,sq);

    cube=i*i*i;

    printf("Cube of %d is %d\n",i,cube);

    ++i;

}

while(i<=num);


return 0;

}
```

OUTPUT:

Enter a number:5

Square of 1 is 1

Cube of 1 is 1

Square of 2 is 4

Cube of 2 is 8

Square of 3 is 9

Cube of 3 is 27

Square of 4 is 16

Cube of 4 is 64

Square of 5 is 25

Cube of 5 is 125

RESULT:

Thus the C program to display the square and cube of first N natural numbers using do..while loop structure is executed successfully.

4. Display a rectangular number pattern using nested for loop

AIM: To display a rectangular number pattern using nested for loop structure.

ALGORITHM:

PROGRAM:

```
#include <stdio.h>

#include <stdlib.h>

int main()

{

int a,b;

for(a=1; a<=5; a++){

for(b=1; b<=10; b++){

printf("%c",'*');

}

printf("\n");

}

getch();

return 0;

}
```

OUTPUT:

```
*****

*****

*****

*****
```

.....

RESULT:

Thus the C program to display a rectangular number pattern using nested for loop structure is executed successfully.

Programs using Arrays

1. Linear Search

AIM: To perform Linear search using arrays.

ALGORITHM:

PROGRAM:

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

```
void main()
```

```
{ int num;
```

```
int i, keynum, found = 0;
```

```
printf("Enter the number of elements ");
```

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

```
int array[num];
```

```
printf("Enter the elements one by one \n");
```

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

```
{
```

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

```
}
```

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

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

```
/* Linear search begins */
```

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

```
{
```

```
    if (keynum == array[i] )
```

```
{
```

```

        found = 1;

        break;
    }
}

if (found == 1)
    printf("Element is present in the array at position %d",i+1);
else
    printf("Element is not present in the array\n");
}

```

OUTPUT:

Enter the number of elements 6

Enter the elements one by one

4

6

1

2

5

3

Enter the element to be searched 6

Element is present in the array at position 2

RESULT:

Thus the C program to perform Linear search is executed successfully.

2. Bubble Sort

AIM: To perform Bubble sort using arrays.

ALGORITHM:

PROGRAM:

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

```
int main()
```

```

{
    int array[100], n, c, d, swap;

    printf("Enter number of elements\n");

    scanf("%d", &n);

    printf("Enter %d integers\n", n);

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

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

    for (c = 0 ; c < n - 1; c++)

    {
        for (d = 0 ; d < n - c - 1; d++)

        {
            if (array[d] > array[d+1]) /* For decreasing order use '<' instead of '>' */

            {
                swap    = array[d];

                array[d] = array[d+1];

                array[d+1] = swap;

            }

        }

    }

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

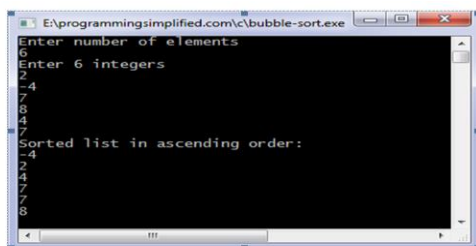
    for (c = 0; c < n; c++)

        printf("%d\n", array[c]);

    return 0;

}

```



```

E:\programmingsimplified.com\c\bubble-sort.exe
Enter number of elements
6
Enter 6 integers
2
4
7
8
4
7
Sorted list in ascending order:
-4
2
4
7
7
8

```


RESULT:

Thus the C program to perform bubble sort is executed successfully.

3. Matrix Addition

AIM: To perform Matrix Addition using two dimensional arrays.

ALGORITHM:**PROGRAM:**

```
#include <stdio.h>

int main() {

    int r, c, a[100][100], b[100][100], sum[100][100], i, j;

    printf("Enter the number of rows (between 1 and 100): ");

    scanf("%d", &r);

    printf("Enter the number of columns (between 1 and 100): ");

    scanf("%d", &c);


    printf("\nEnter elements of 1st matrix:\n");

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

        for (j = 0; j < c; ++j) {

            printf("Enter element a[%d][%d]: ", i, j);

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

        }

    printf("Enter elements of 2nd matrix:\n");

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

        for (j = 0; j < c; ++j) {

            printf("Enter element a[%d][%d]: ", i, j);

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

        }

    // adding two matrices

    for (i = 0; i < r; ++i)
```

```

    for (j = 0; j < c; ++j) {
        sum[i][j] = a[i][j] + b[i][j];
    }
// printing the result
printf("\nSum of two matrices: \n");
for (i = 0; i < r; ++i)
    for (j = 0; j < c; ++j) {
        printf("%d  ", sum[i][j]);
        if (j == c - 1) {
            printf("\n\n");
        }
    }
return 0;
}

```

OUTPUT:

Enter the number of rows (between 1 and 100): 2

Enter the number of columns (between 1 and 100): 3

Enter elements of 1st matrix:

Enter element a[0][0]: 1

Enter element a[0][1]: 2

Enter element a[0][2]: 3

Enter element a[1][0]: 1

Enter element a[1][1]: 2

Enter element a[1][2]: 3

Enter elements of 2nd matrix:

Enter element a[0][0]: 1

Enter element a[0][1]: 2

Enter element a[0][2]: 3

Enter element a[1][0]: 1

Enter element a[1][1]: 2

Enter element a[1][2]: 3

Sum of two matrices:

2 4 6

2 4 6

RESULT:

Thus the C program to perform Matrix addition using two dimensional arrays is executed successfully.

4. Find the Transpose of a matrix

AIM: To find Transpose of a Matrix using two dimensional arrays.

ALGORITHM:

PROGRAM:

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

```
int main() {
```

```
    int a[10][10], transpose[10][10], r, c, i, j;
```

```
    printf("Enter rows and columns: ");
```

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

```
    // Assigning elements to the matrix
```

```
    printf("\nEnter matrix elements:\n");
```

```
    for (i = 0; i < r; ++i)
```

```
        for (j = 0; j < c; ++j) {
```

```
            printf("Enter element a[%d][%d]: ", i, j);
```

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

```
        }
```

```

// Displaying the matrix a[][]

printf("\nEnter matrix: \n");

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

    for (j = 0; j < c; ++j) {

        printf("%d ", a[i][j]);

        if (j == c - 1)

            printf("\n");

    }


// Finding the transpose of matrix a

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

    for (j = 0; j < c; ++j) {

        transpose[j][i] = a[i][j];

    }


// Displaying the transpose of matrix a

printf("\nTranspose of the matrix:\n");

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

    for (j = 0; j < r; ++j) {

        printf("%d ", transpose[i][j]);

        if (j == r - 1)

            printf("\n");

    }

return 0;

}

```

OUTPUT:

Enter rows and columns: 2

Enter matrix elements:

Enter element a[0][0]: 1

Enter element a[0][1]: 2

Enter element a[0][2]: 3

Enter element a[1][0]: 4

Enter element a[1][1]: 1

Enter element a[1][2]: 2

Entered matrix:

1 2 3

4 1 2

Transpose of the matrix:

1 4

2 1

3 2

RESULT:

Thus the C program to find Transpose of a Matrix using two dimensional arrays is executed successfully.

Programs using Functions

1. Generate Fibonacci series using Recursive Functions

AIM: To generate Fibonacci series using Recursive functions.

ALGORITHM:

PROGRAM:

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

```
int fibonacci(int i) {
```

```
    if(i == 0) {
```

```

        return 0;
    }

    if(i == 1) {
        return 1;
    }

    return fibonacci(i-1) + fibonacci(i-2);
}

int main() {

    int i,n;

    printf("For generating Fibonacci series till n-->Enter n value:\n");

    scanf("%d",&n);

    printf("Fibonacci series generated:");

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

        printf("%d\t", fibonacci(i));

    }

    return 0;
}

```

Output:

For generating Fibonacci series till n-->Enter n value:

7

Fibonacci series generated:

0

1

1

2

3

5

8

RESULT:

Thus the C program to generate Fibonacci series using Recursive functions is executed successfully.

Pointers

1. Swap two numbers without using the 3rd variable implementing Pointer Concept.

AIM: To swap two numbers without using the 3rd variable implementing Pointer Concept.

ALGORITHM:

PROGRAM:

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

```
int main()
```

```
{
```

```
    int x, y, *a, *b, temp;
```

```
    printf("Enter the value of x and y\n");
```

```
    scanf("%d%d", &x, &y);
```

```
    printf("Before Swapping\nx = %d\ny = %d\n", x, y);
```

```
    a = &x;
```

```
    b = &y;
```

```
    temp = *b;
```

```
    *b = *a;
```

```
    *a = temp;
```

```
printf("After Swapping\nx = %d\ny = %d\n", x, y);
```

```
return 0;
```

```
}
```

OUTPUT:

Enter the value of x and y

10

20

Before Swapping

x = 10

y = 20

After Swapping

x = 20

y = 10

Structures and Union

1. Employee database creation using Structures

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

```
/*structure declaration*/
```

```
struct employee{
```

```
    char   name[30];
```

```
    int    empId;
```

```
    float  salary;
```

```
};
```

```
int main()
```

```
{
```



```

/*declare structure variable*/

struct employee emp;

/*read employee details*/

printf("\nEnter details :\n");

printf("Name ?:" );    gets(emp.name);

printf("ID ?:" );      scanf("%d",&emp.empId);

printf("Salary ?:" );  scanf("%f",&emp.salary);


/*print employee details*/

printf("\nEnter detail is:");

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

printf("Id: %d" ,emp.empId);

printf("Salary: %f\n",emp.salary);

return 0;

}

```

Output:

Enter details :

Name?:Mike

ID?:1120

Salary?:76543

Entered detail is:

Name: Mike

Id: 1120

Salary: 76543.000000

2. Student database creation using Union

```
#include <stdio.h>

#include <string.h>

union student
{
    char name[20];
    char subject[20];
    float percentage;
};

int main()
{
    union student record1;
    union student record2;

    // assigning values to record1 union variable
    strcpy(record1.name, "Raju");
    strcpy(record1.subject, "Maths");
    record1.percentage = 86.50;

    printf("Union record1 values example\n");
    printf(" Name      : %s \n", record1.name);
    printf(" Subject   : %s \n", record1.subject);
    printf(" Percentage : %f \n\n", record1.percentage);

    // assigning values to record2 union variable
```

```
printf("Union record2 values example\n");

strcpy(record2.name, "Mani");

printf(" Name      : %s \n", record2.name);


strcpy(record2.subject, "Physics");

printf(" Subject   : %s \n", record2.subject);


record2.percentage = 99.50;

printf(" Percentage : %f \n", record2.percentage);

return 0;

}
```

Output:

```
Union record1 values example
Name :
Subject :
Percentage : 86.500000;
Union record2 values example
Name : Mani
Subject : Physics
Percentage : 99.500000
```