INSTITUTE OF ENGINEERING

Pulchowk Campus, Lalitpur

Subject: C Programming

Lab Report 1 and 2

Tittle: **Repetitive Structure (Branching Looping)**

Submitted by:

Susheel Thapa 077BCT090

Submitted to:

Department of Electronics and

Computer Engineering

Checked by

Content of Lab Report:

Background Information

What is C Programming?

C programming is a general-purpose, procedural, imperative computer programming language developed in 1972 by Dennis M. Ritchie at the Bell Telephone Laboratories to develop the UNIX operating system. C is the most widely used computer language.

Why to Learn C Programming?

- Easy to learn
- Structured language
- It produces efficient programs
- It can handle low-level activities
- It can be compiled on a variety of computer platforms

Editor

Here, I have used Visual Studio Code as my editor. You can download the editor from Download Visual Studio Code - Mac, Linux, Windows . Select your operating system and download it.

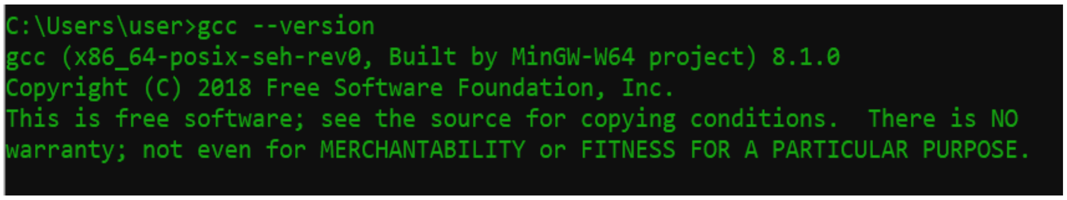
Compiler

Here, I have used **gcc** as my compiler provided by MinGWw64. You can download it via Download MinGW-w64 - for 32 and 64 bit Windows from SourceForge.net. Your download will start automatically. Run the downloaded .exe file. After, you have installed MinGW-w64, you need to configure it.

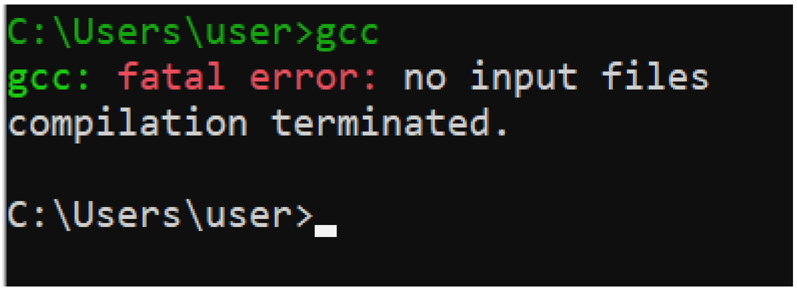
In the Windows search bar, type 'settings' to open your Windows Settings.
Search for Edit environment variables for your account.
Choose the Path variable and then select Edit.
Select New and add the Mingw-w64 destination folder path to the system path. The exact path depends on which version of Mingw-w64 you have installed and where you installed it. If you used the settings above to install Mingw-w64, then add this to the path: **C:\Program Files\mingw-w64\x86_64-8.1.0-posix-seh-rt_v6-rev0\mingw64\bin.**
Select OK to save the updated PATH. You will need to reopen any console windows for the new PATH location to be available.

Check your installation

Open command prompt or power shell and type:



```
C:\Users\user>gcc --version
gcc (x86_64-posix-seh-rev0, Built by MinGW-W64 project) 8.1.0
Copyright (C) 2018 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```



```
C:\Users\user>gcc
gcc: fatal error: no input files
compilation terminated.

C:\Users\user>_
```

If you get similar result, you are good to go.

Repetitive Structure

For Loop

A for loop is a repetition control structure that allows you to efficiently write a loop that needs to execute a specific number of times.

Syntax:

for (init; condition; increment)

{statement(s);}

Here is the flow of control in a 'for' loop −

- The init step is executed first, and only once. This step allows you to declare and initialize any loop control variables. You do not have to put a statement here, as long as a semicolon appears.
- Next, the condition is evaluated. If it is true, the body of the loop is executed. If it is false, the body of the loop does not execute and the flow of control jumps to the next statement just after the 'for' loop.
- After the body of the 'for' loop executes, the flow of control jumps back up to the increment statement. This statement allows you to update any loop control variables. This statement can be left blank, as long as a semicolon appears after the condition.

- The condition is now evaluated again. If it is true, the loop executes and the process repeats itself (body of loop, then increment step, and then again condition). After the condition becomes false, the 'for' loop stops.

While …. Loop

A while loop in C programming repeatedly executes a target statement as long as a given condition is true.

Syntax

> *while(condition)*
>
> *{statement(s);}*

Here, statement(s) may be a single statement or a block of statements. The condition may be any expression, and true is any nonzero value. The loop iterates while the condition is true.

When the condition becomes false, the program control passes to the line at once following the loop.

Do ...While Loop

A do...while loop is similar to a while loop, except the fact that it is guaranteed to execute at least one time.

Syntax:

> *do {statement(s);}*
>
> *while (condition);*

The conditional expression appears at the end of the loop, so the statement(s) in the loop executes once before the condition is tested.

If the condition is true, the flow of control jumps back up to do, and the statement(s) in the loop executes again. This process repeats until the given condition becomes false.

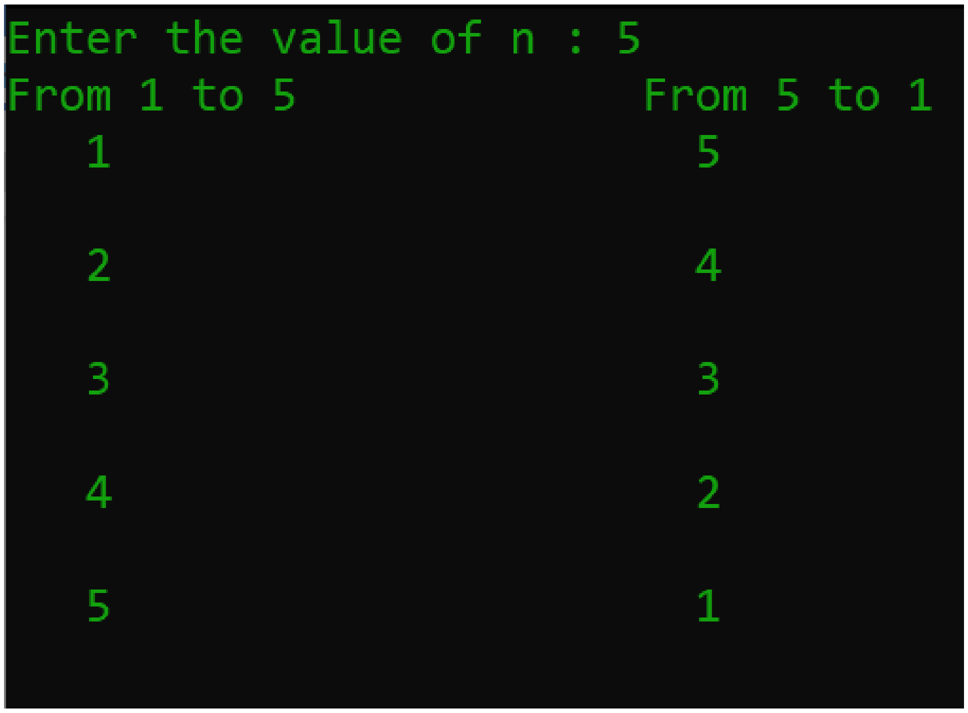
CODE AND OUTPUT

2.Write a program to read an unsigned integer (suppose n) and display from 1 to n and n to 1

Source Code:

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

int main ()
{
    int n;
    system("cls");
    printf("Enter the value of n: ");
    scanf("%d", &n);
    printf("From 1 to %d",n);
    printf("\t\tFrom %d to 1", n);
    for (int i = 1; i <= n; i++)
    {
        printf("\n %3d\t\t\t%3d\n", i, (n - i + 1));
    }
    getch();
    return 0;
}
```

Output:



```
Enter the value of n : 5
From 1 to 5             From 5 to 1
   1                       5

   2                       4

   3                       3

   4                       2

   5                       1
```

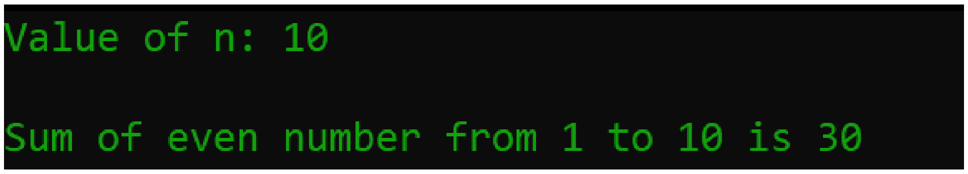
3.Write a program to display the sum of even numbers from 1 ton (n is an unsigned integer)

Source Code:

```
#include <stdio.h>

#include <conio.h>

#include <stdlib.h>


int main ()

{

   unsigned int n;

   int sum_of_even;


   system("cls");

   printf("Value of n: ");
```

```
    scanf("%u", &n);
    for (int i = 1; i <= n; i++)
    {

        if ((i % 2) == 0)
        {
            sum_of_even = sum_of_even + i;
        }
    }
    printf("\nSum of even number from 1 to %d is %d",n, sum_of_even);
    getch();
    return 0;
}
```

Output:



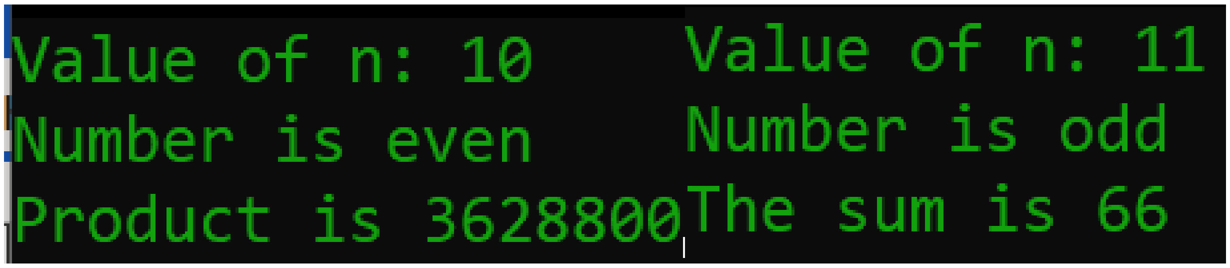
```
Value of n: 10

Sum of even number from 1 to 10 is 30
```

4.Write a program to read an integer and find out product from 1 to n if n is even and sum from 0 to n if n is odd.

Source Code:

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

int main ()
{
    int n,product =1, sum=0;
    system("cls");
    printf("Value of n: ");
    scanf("%u", &n);
    if(n%2==0) {
            printf("Number is even\n");
        for (int i = 1; i <= n;i++)
        {
            product = product * i;
        }
            printf("Product is %d", product);
    }
    else {
            printf("Number is odd\n");
        for (int i = 0; i <= n;i++) {
            sum = sum + i;
        }
            printf("The sum is %d", sum);
    }
    getch();
    return 0;
}
```

Output:



```
Value of n: 10
Number is even
Product is 3628800
```

```
Value of n: 11
Number is odd
The sum is 66
```

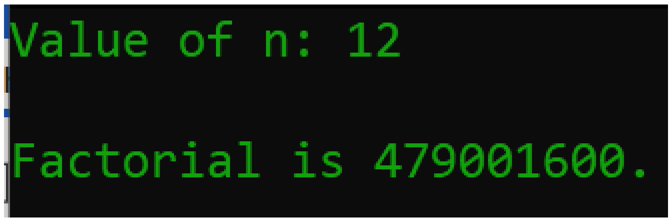
5.Write a program to read an integer and compute its factorial and display proper message if its factorial cannot be completed

Source Code:

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

int main ()
{
    float n;
    int factorial = 1;
    system("cls");
    printf("Value of n: ");
    scanf("%f", &n);

    if ((n - (int)n) == ((float)2-2))
    {
        if (n > 0)
        {
            if (n == 0 || n == 1)
            {
                printf("\nFactorial is %d", 1);
            }
            else
            {
                for (int i = 1; i <= n; i++)
                {
```

```
                factorial = factorial * i;
            }
            printf("\nFactorial is %d.", factorial);
        }
    }
    else
    {
        printf("Expected: Positive Integer\nGiven: Negative Integer\n");
    }
}
else
{
    printf("Expected: Integer\nGiven: floating number.");
}

getch();
return 0;
}
```

Output:



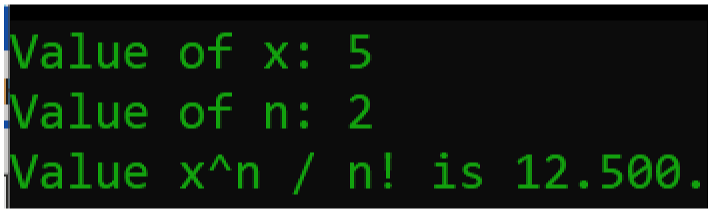
```
Value of n: 12

Factorial is 479001600.
```

6.Write a program to calcualte x^n/n! where x is floating point number and n is integer

Source Code

```
#include <stdio.h>

#include <conio.h>

#include <stdlib.h>

#include<math.h>


int main ()

{

    float x;

    int n, factorial=1;

    system("cls");

    printf("Value of x: ");

    scanf("%f", &x);

    printf("Value of n: ");

    scanf("%d", &n);

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

        factorial = factorial * i;

    }

    printf("Value x^n / n! is %.3f.", (pow (x, n) / (double)factorial));

    getch();

    return 0;

}
```

Output:



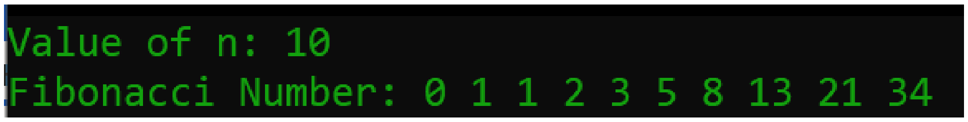
```
Value of x: 5
Value of n: 2
Value x^n / n! is 12.500.
```

7.Write a program to display the fibonacci series

Source Code:

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

int main ()
{
    int a = 0, b = 1, c;
    int n;

    system("cls");
    printf("Value of n: ");
    scanf("%d", &n);
    printf("Fibonacci Number: ");
    for (int i = 0; i <n;i++) {
        printf("%d ", a);
        c = a + b;
        a = b;
        b = c;
    }
        getch();
    return 0;
}
```

Output:



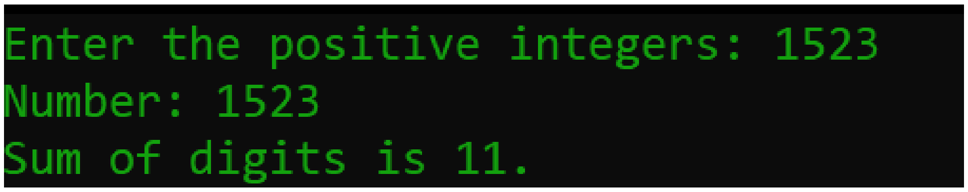
```
Value of n: 10
Fibonacci Number: 0 1 1 2 3 5 8 13 21 34
```

8.Write a program to read a positive number and fins its sum of digits in it.

Source Code:

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

int main ()
{
    int number, remainder, sum;
    system("cls");
    printf("Enter the positive integers: ");
    scanf("%d", &number);



    printf("Number: %d", number);
    while (number! = 0)
    {
        remainder = number % 10;
        sum = sum + remainder;
        number = number / 10;
    }
    printf("\nSum of digits is %d.", sum);
    getch();
    return 0;
}
```

Output:



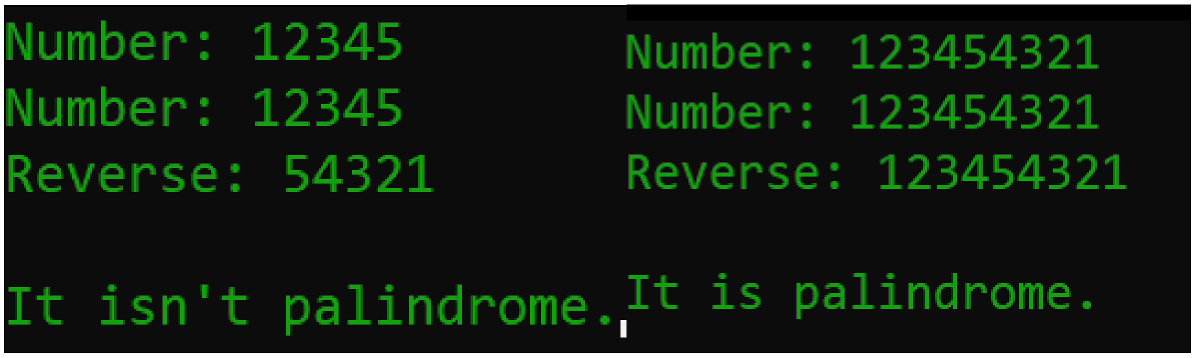
```
Enter the positive integers: 1523
Number: 1523
Sum of digits is 11.
```

9.1] Write a program to read number and check whether it is palindrome or not

Source Code:

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

int main ()
{
    int number, reverse, remainder, copy_number;
    system("cls");
    printf("Number: ");
    scanf("%d", &number);
    copy_number = number;
    while (copy_number != 0)
    {
        remainder = copy_number % 10;
        reverse = reverse * 10 + remainder;
        copy_number = copy_number / 10;
    }
    printf("Number: %d\nReverse: %d\n\n", number, reverse);
    if (number == reverse)
    {
        printf("It is palindrome.");
    }
    else
    {
        printf("It isn't palindrome.");
    }
    getch();
    return 0;
}
```

Output:



```
Number: 12345
Number: 12345
Reverse: 54321

It isn't palindrome.
```

```
Number: 123454321
Number: 123454321
Reverse: 123454321

It is palindrome.
```

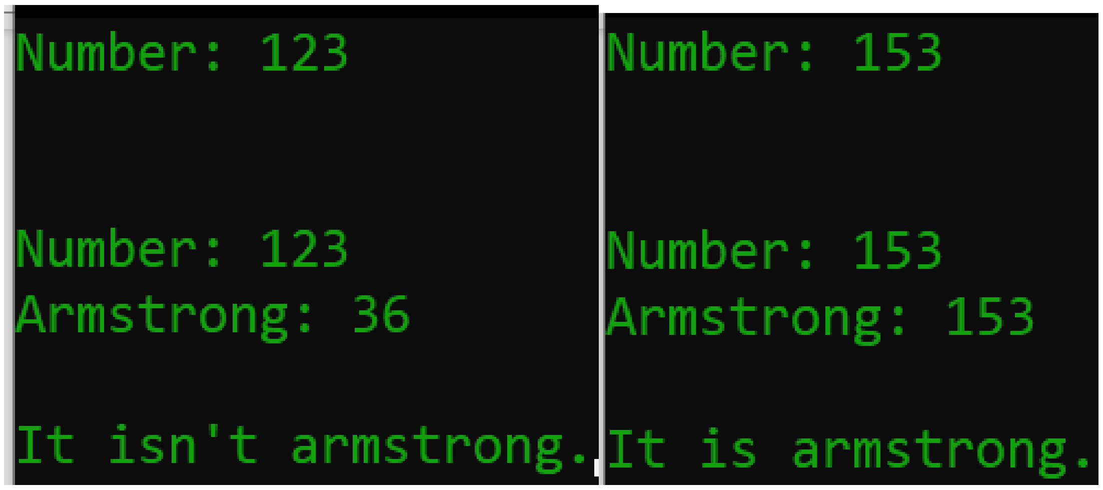
9.2] Write a program to read number and check whether it is Armstrong or not

Source Code:

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

int main ()
{
   int armstrong = 0, remainder, number, copy_number;
   system("cls");
   printf("Number: ");
   scanf("%d", &number);
   copy_number = number;
   while (copy_number != 0)
   {
      remainder = copy_number % 10;
      armstrong = pow (remainder, 3) + armstrong;
      copy_number = copy_number / 10;
   }
   printf("\n\nNumber: %d\nArmstrong: %d\n\n", number, armstrong);

   if (number == armstrong)
   {
      printf("It is armstrong.");
   }
   else
   {
      printf("It isn't armstrong.");
```

```
    }
    getch();
    return 0;
}
```

Output:



```
Number: 123


Number: 123
Armstrong: 36

It isn't armstrong.
```

```
Number: 153


Number: 153
Armstrong: 153

It is armstrong.
```

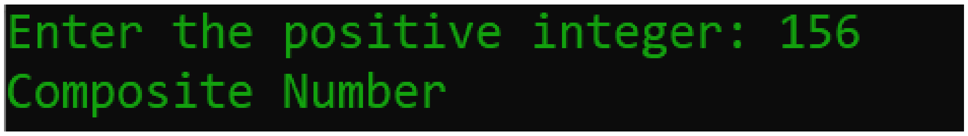
9.3] Write a program to read number and check whether it is prime or not

Source Code:

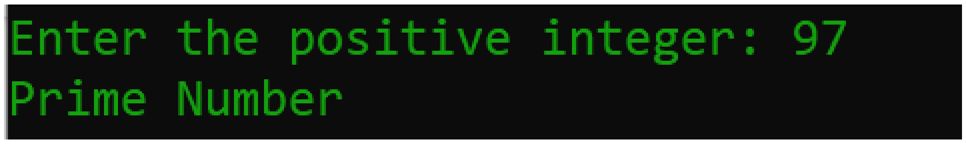
```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>

int main ()
{

    int number, i;
    system("cls");
```

```
printf("Enter the positive integer: ");
scanf("%d", &number);

i = number - 1;

if (number == 0 || number == 1)
{
    printf("It is neither prime nor composite.");
}
else
{
    for (; i; i--)
    {

        if (!(i == 0 || i == 1))

        {
            if (number % i == 0)
            {
                printf("Composite Number\n");
                break;
            }
        }
        else
        {
            printf("Prime Number");
            }
        }
    }

    getch();
    return 0;
}
```

Output:



```
Enter the positive integer: 156
Composite Number
```



```
Enter the positive integer: 97
Prime Number
```

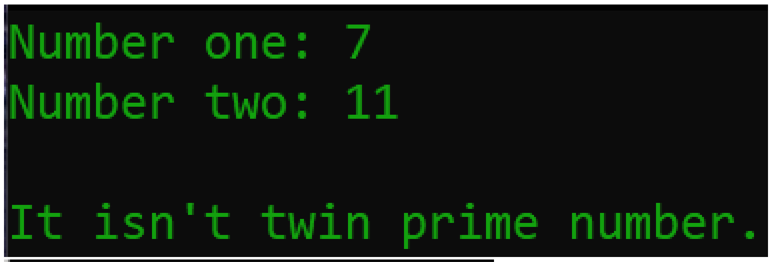
9.4] Write a program to read two number and check whether it is twin prime or not

Source Code:

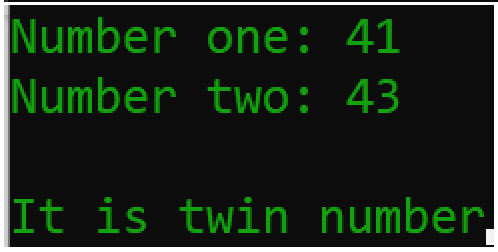
```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>

int main ()
{
    int number1, number2;
    int twin_number1 = 1, twin_number2 = 1;

    float n1, n2;
```

```
system("cls");

printf("Number one: ");
scanf("%d", &number1);
printf("Number two: ");
scanf("%d", &number2);
if (number1 == 1 || number2 == 1 || number1 == 1 || number2 == 1)
{
    printf("\nIt isn't twin number.");
    getch();
    exit (0);
}
if ((number1 - number2) == 2 || (number2 - number1) == 2)
{
    for (int i = 2; i < number1; i++)
    {
        if (number1 % i == 0)
        {
            twin_number1 = 0;
            break;
        }
    }

    for (int i = 2; i < number2; i++)
    {
        if (number2 % i == 0)
        {
                twin_number2 = 0;
                break;
            }
        }
        if (twin_number1 == 1 && twin_number2 == 1)
        {
            printf("\nIt is twin number");
        }
        else
        {
            printf("\nIt isn't twin number.");
        }
    }
    else
    {
        printf("\nIt isn't twin prime number.");
    }
    getch();
    return 0;
}
```

Output:



```
Number one: 7
Number two: 11

It isn't twin prime number.
```



```
Number one: 41
Number two: 43

It is twin number
```

10.Write a program to read a number (n) adn display its multiplication table up to 10.

Source Code:

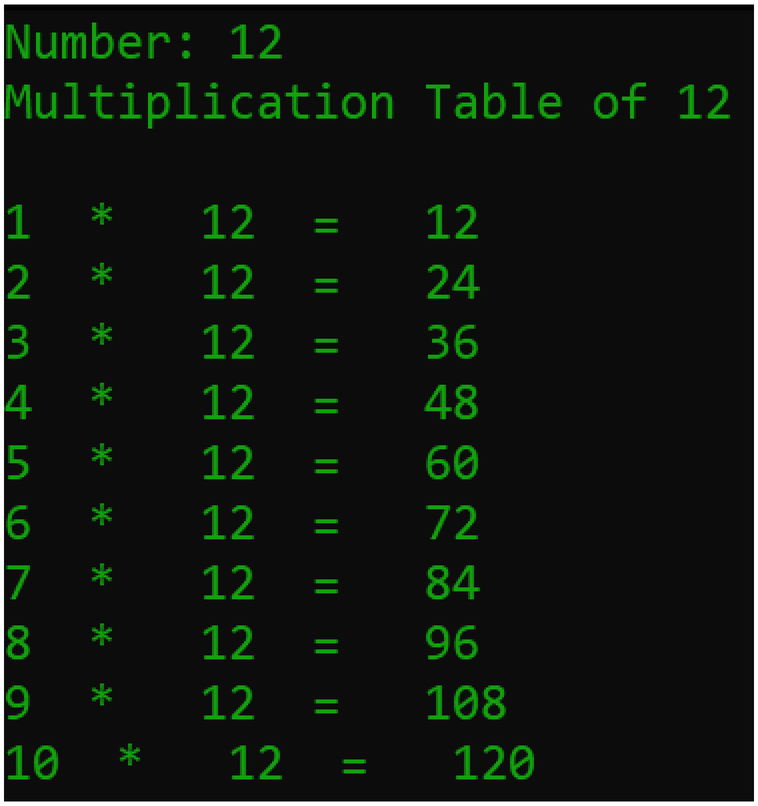
```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>

int main ()
{
   int number;
   system("cls");
   printf("Number: ");
   scanf("%d", &number);
   printf("Multiplication Table of %d\n\n", number);
   for (int i = 1; i <= 10; i++)
   {
      printf("%d *   %d =   %d\n", i, number, i * number);
   }
```

getch();

return 0;

}

Output:



```
Number: 12
Multiplication Table of 12

1   *   12  =   12
2   *   12  =   24
3   *   12  =   36
4   *   12  =   48
5   *   12  =   60
6   *   12  =   72
7   *   12  =   84
8   *   12  =   96
9   *   12  =   108
10   *   12  =   120
```

Analysis

Here, we have learned above various ways iteration over the loop. We come across knowledge of **for loop**, **while loop**, **do...while loop.** We learnt in what kind of condition we should use this looping statement. We use for loop when we know the exact number of iterations and we use while or do...while loop when we do not know number of iterations.

Also, it helps to understand the workings of these statements and building logic on Looping and iterations taking program.

Conclusion

Conclusion

In nut shell, we became acquaintance with **Branching structure.** Also, we shape ourselves with these newly learnt statements.