Computer Programming

By: Engr. Shumail Zehra

Contents

LL 02 = Learning Level 02 – Comprehension, LL 04 = Learning Level 04 – Analysis

Iterative Control Flow

- Iterative control flow is also referred to as **repetition logic** or **loop**.
- It is one of the order in which the program instructions are executed.
- It executes the instructions repetitively multiple number of times.
- Repetitive logic allows us to execute a statement or set of statements multiple number of times just by writing down them once.
- Iteration is the act of repeating the statements, and each of the repetition is
  also called as **iteration**.

Iterative Control Flow

In iterative logic:

- Statements are executed multiple number of times on the basis of the condition.
- All statements are repeated until a certain condition is reached.
- A condition may be **open-ended** as in a "sentinel" repetition or it may be **predefined** as in the "counter-controlled" repetition.

Types of Iterative Control Flow

- In programming we normally have two types of iterative/repetitive control flow:

Counter-Controlled Repetition

Sentinel-Controlled Repetition

Counter-Controlled Repetition

- In counter-controlled repetition, a counter variable is used that specifies how many times the statements are to be repeated.

- The case, when we know exactly how many times we have to repeat the statements, we will use counter-controlled repetition logic.

- Suppose the set of statements need to be executed is N number of times. We
  will first set the *counter* to 1, and every time we check the *counter* (*counter <= N*) and increment it (*counter = counter + 1*).

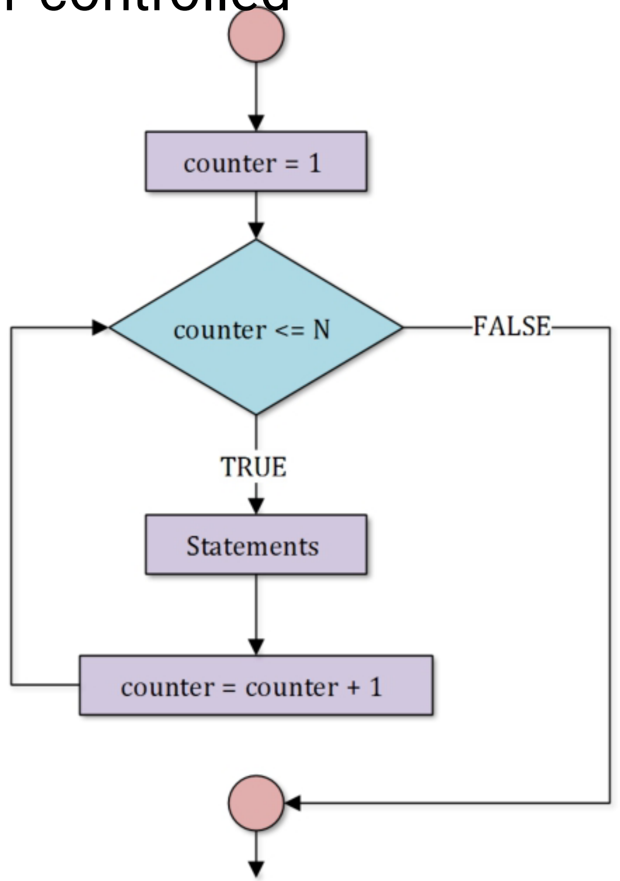
Counter-Controlled Repetition

In counter-controlled repetition:

- The condition is predefined.
- The statements are repeated if the condition is satisfied (true).
- The repetition structure is terminated when condition is not satisfied (false).

Counter-Controlled Repetition

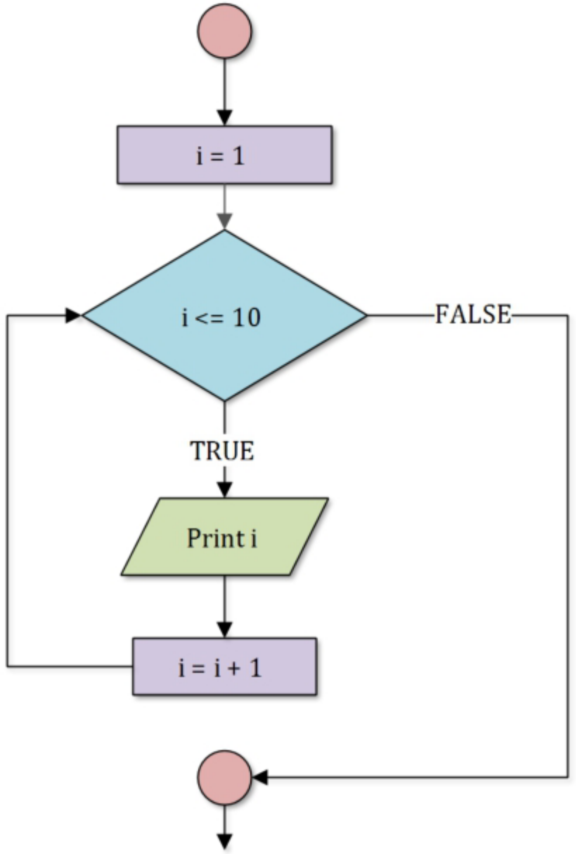
Following is the flow of execution of counter-controlled repetition:



Counter-Controlled Repetition - Example

Problem Statement:

Display the integer numbers from 1 to 10



Sentinel-Controlled Repetition

- In sentinel-controlled repetition, a condition specifies how many times the statements are to be repeated.

- The case, when we do not know exactly how many times we have to repeat the statements, we will use sentinel-controlled repetition logic.

- Suppose we have to create a program that continuously reads lines from a text file and displays them until it reaches the end of the file. In this case we do not know how many lines will be there in different text files.

- It is an example of sentinel-controlled repetition.

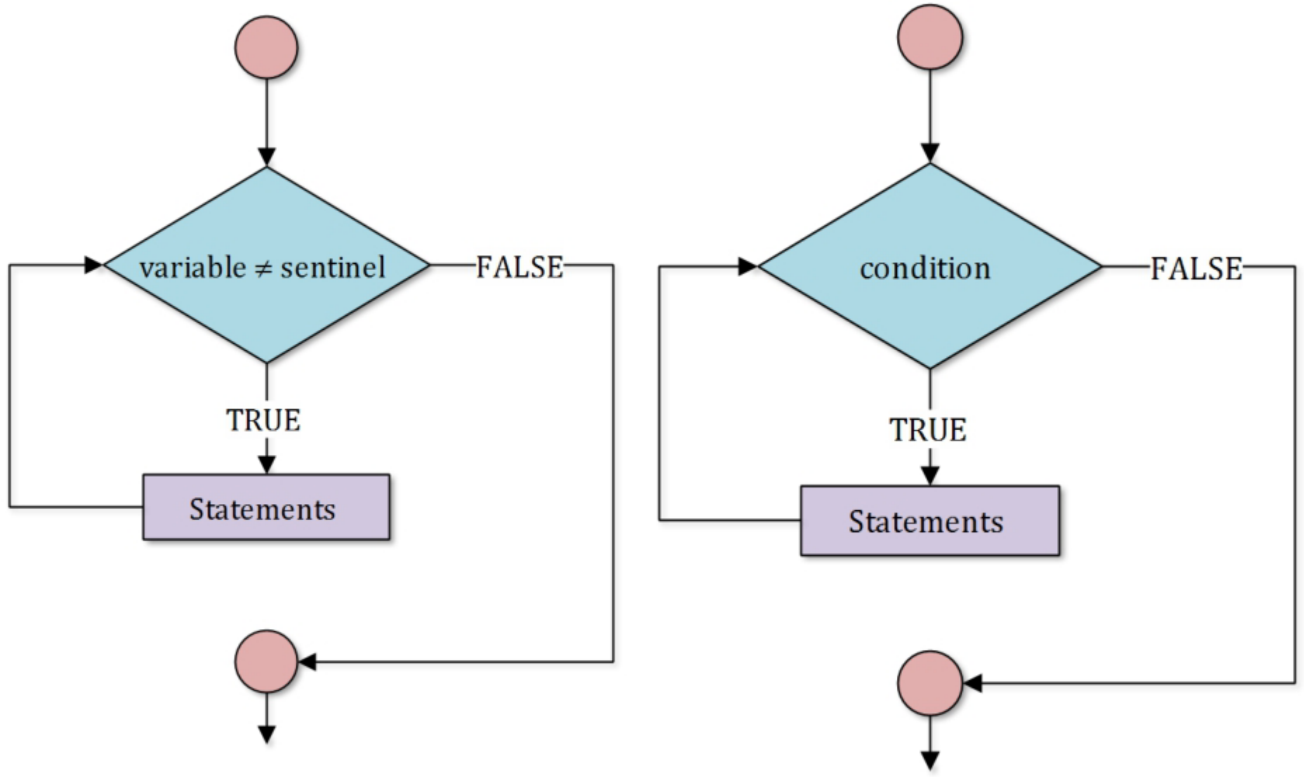
Sentinel-Controlled Repetition

In sentinel-controlled repetition :

- The condition is open-ended.
- The statements are repeated if the condition is satisfied (true).
- The repetition structure is terminated when condition is not satisfied (false).

Sentinel-Controlled Repetition

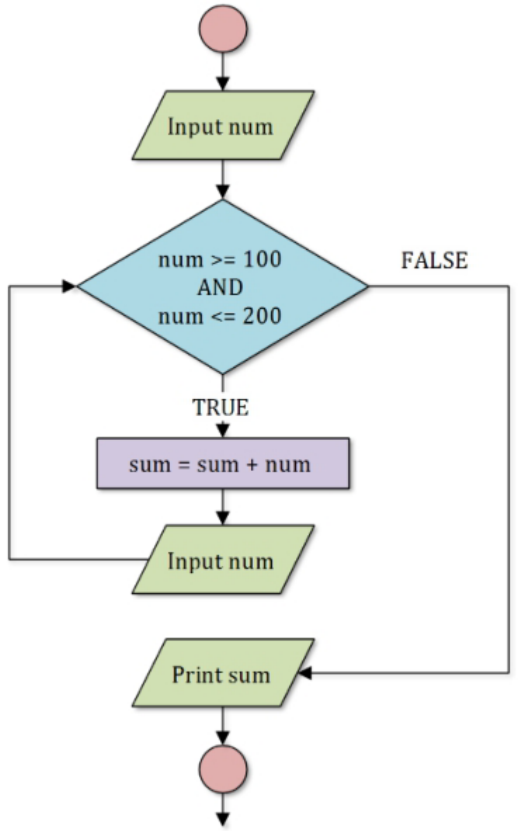
Following is the flow of execution of sentinel-controlled repetition:



Sentinel-Controlled Repetition - Example

Problem Statement:

The user continuously enters integer numbers, the program stops when the user enters any number other than the numbers between 100 and 200. Finally it displays the sum of all the numbers entered by the user.



Iterative Control Structures in C++

- Iterative control structures are used to execute set of statements multiple number of times on the basis of a condition.

- The statements that are needed to be executed repetitively are placed with
  in the iterative structure and the number of times to be executed is specified either predefinedly or open endedly.

- The iterative control structures implement the iterative/repetitive logic in C++.

for Loop/Statement in C++

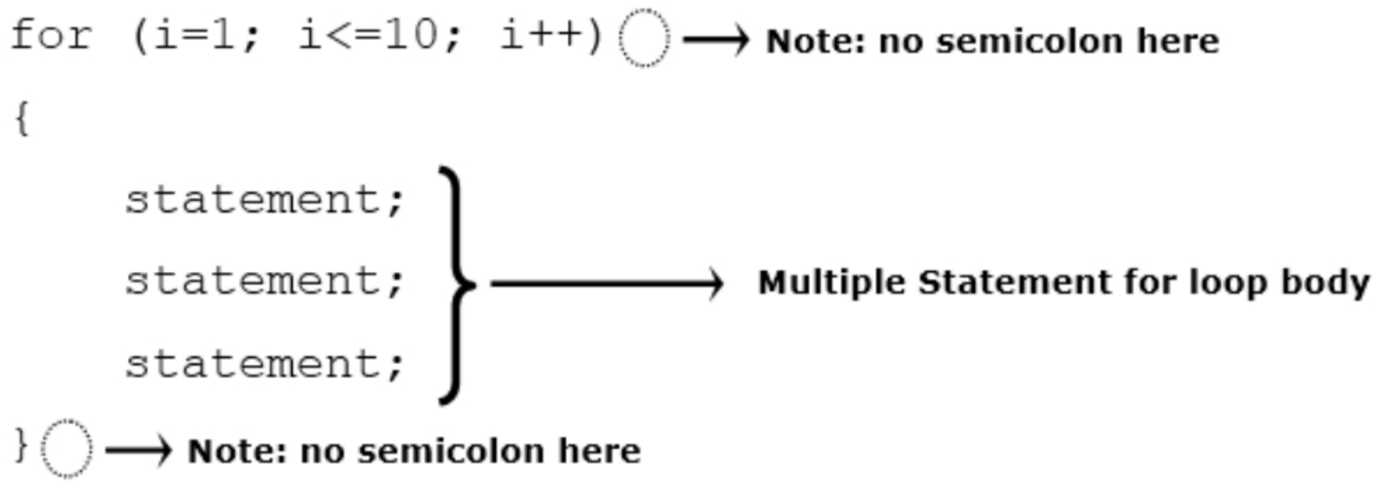
- *for loop/statement* implements **counter-controlled repetition logic**.
- The for loop is used to execute the particular statements of code fixed number of
  times with out writing those statements that much number of time.
- The for loop is used when we know that how many times the loop will be executed.
- It repeats statements on the basis of a condition.
- Here the condition is predefined.
- If the condition is ***true***, statements are repeated again.
- If the condition is ***false***, the loop is terminated.
- It uses one counter variable that counts the number of times, the loop is to be repeated.

for Loop/Statement – Syntax

```
for(initialization; test; increment)
{
    statement set ;
}
```

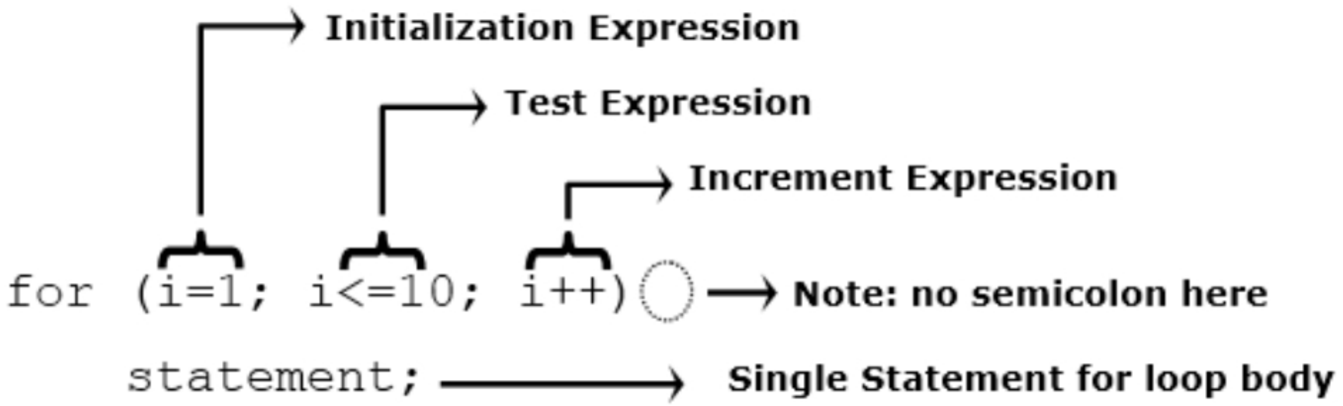
```
for(initialization; test; increment)
    statement;
```

for Loop/Statement – Syntax



```
for (i=1; i<=10; i++)
{
    statement;
    statement;
    statement;
}
```

for Loop/Statement – Syntax

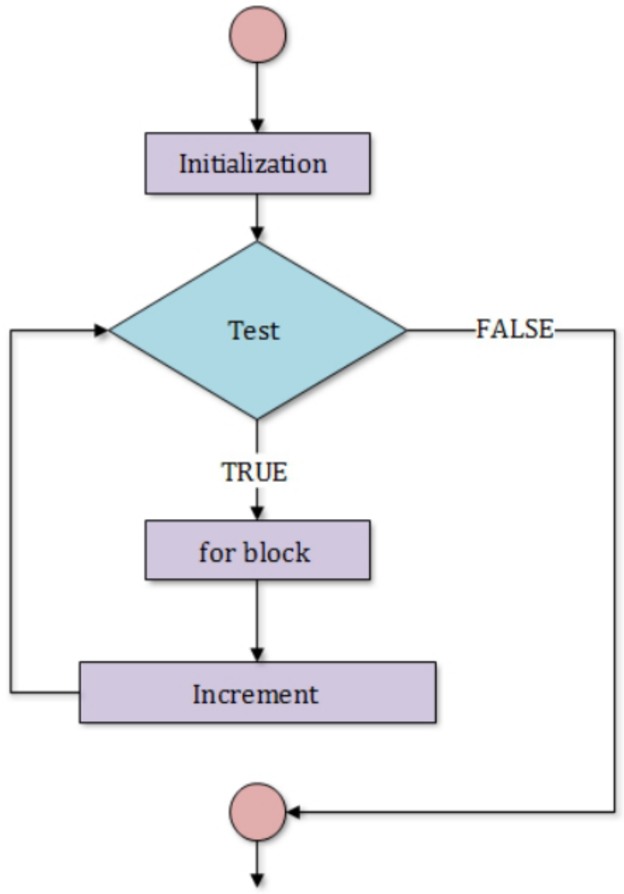


```
for (i=1; i<=10; i++)
    statement;
```

for Loop/Statement – Syntax

- There are two syntaxes of for statement.
- In first syntax we have multiple statements inside the for statement.
- In this case it is compulsory to enclose all the statements in the braces { } .
- In second syntax we have just one statement inside the for statement.
- In this case it is optional to enclose the statement in the braces { } .
- All the statements enclosed in { } is called as the **for block**.

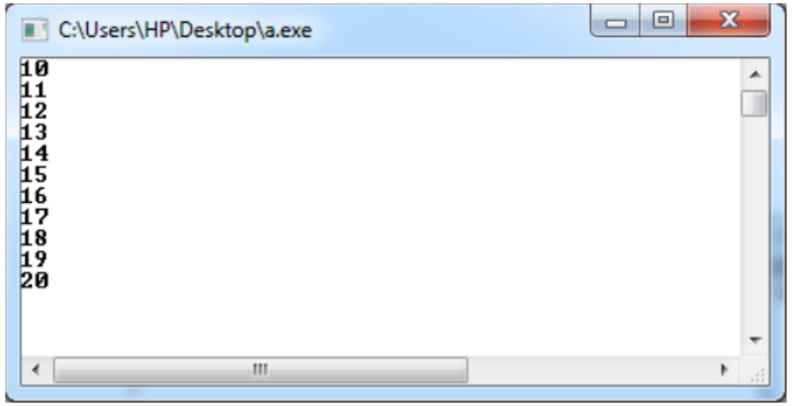
for Loop/Statement – Flow Chart



for Loop/Statement - Example

Problem Statement 1: Display all the integers between 10 and 20.

```
#include<iostream>
#include<conio.h>

using namespace std;

int main()
{
    for(int i=10; i<=20; i++)
    {
        cout<<i<<endl;
    }

    getch();
    return 0;
}
```



```
10
11
12
13
14
15
16
17
18
19
20
```

while Loop/Statement in C++

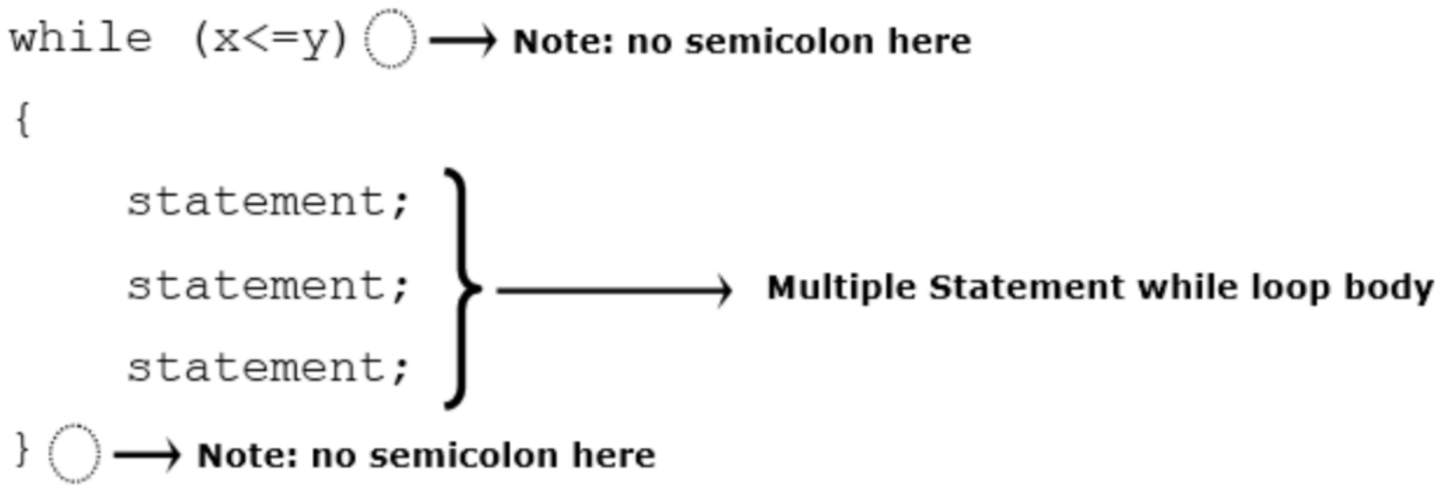
- *while loop/statement* implements sentinel-controlled repetition logic.
- The while loop is used to execute the particular statements of code multiple number of times without writing those statements that much number of time.
- The while loop is used when we do not know that how many times the loop will be executed.
- It repeats statements on the basis of a condition.
- Here the condition is open-ended.
- If the condition is *true*, statements are repeated again.
- If the condition is *false*, the loop is terminated.

while Loop/Statement – Syntax

```
while( Condition )
{
    statement set ;
}
```

```
while( Condition )
    statement;
```

while Loop/Statement – Syntax



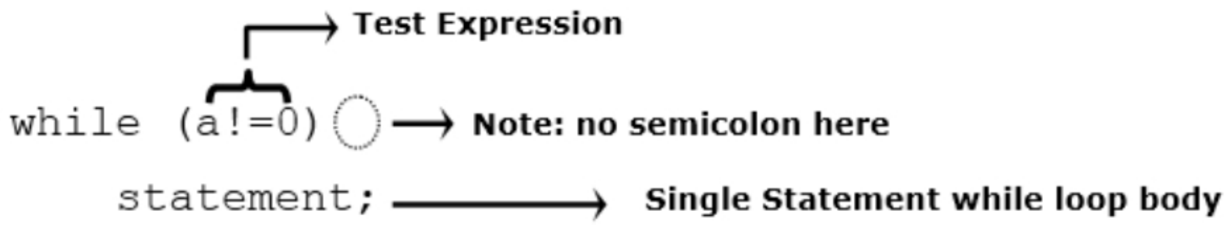
```
while (x<=y)
{
     statement;
     statement;
     statement;
}
```

Note: no semicolon here

Multiple Statement while loop body

Note: no semicolon here

while Loop/Statement – Syntax



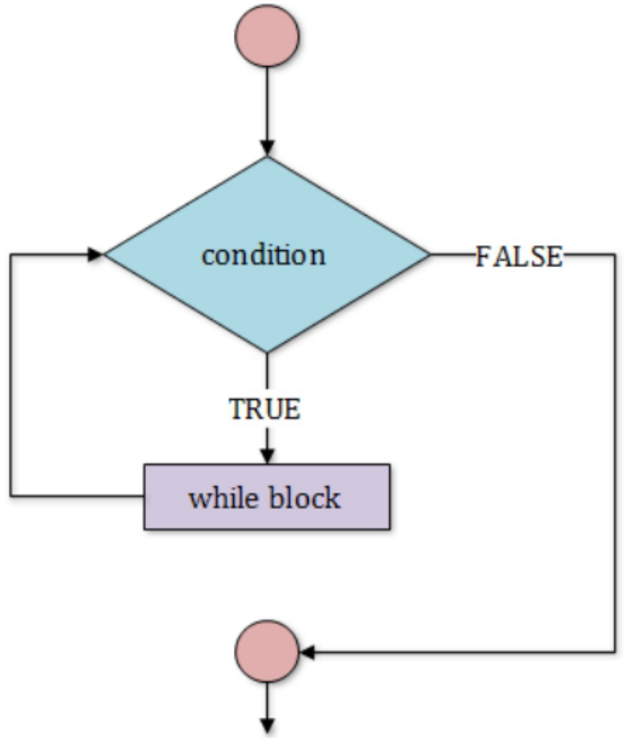
```
while (a!=0)
    statement;
```

while Loop/Statement – Syntax

- There are two syntaxes of while statement.
- In first syntax we have multiple statements inside the while statement.
- In this case it is compulsory to enclose all the statements in the braces { } .
- In second syntax we have just one statement inside the while statement.
- In this case it is optional to enclose the statement in the braces { } .

All the statements enclosed in { } is called as the while block.

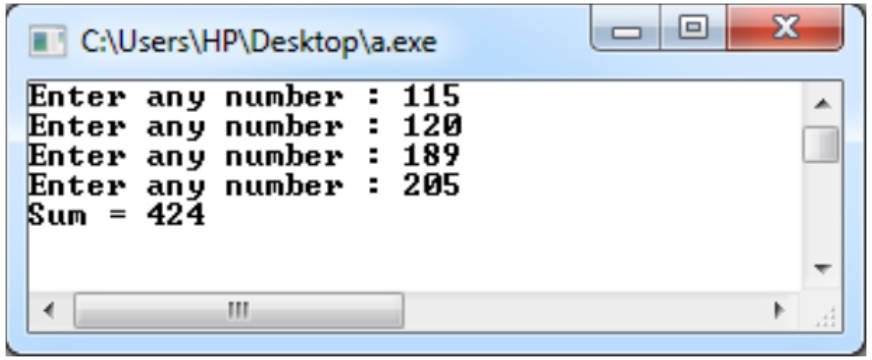
while Loop/Statement – Flow Chart



while Loop/Statement - Example

Problem Statement: The user continuously enters integer numbers, the program stops when the user enters any number other than the numbers between 100 and 200. Finally it displays the sum of all the numbers entered by the user.

```cpp
#include<iostream>
#include<conio.h>

using namespace std;

int main()
{
	int num, sum=0;
	cout<<"Enter any number : ";
	cin>>num;

	while(num>=100 && num<=200)
	{
		sum+=num;
		cout<<"Enter any number : ";
		cin>>num;
	}

	cout<<"Sum = "<<sum;

	getch();
	return 0;
}
```


C:\Users\HP\Desktop\a.exe

```
Enter any number : 115
Enter any number : 120
Enter any number : 189
Enter any number : 205
Sum = 424
```

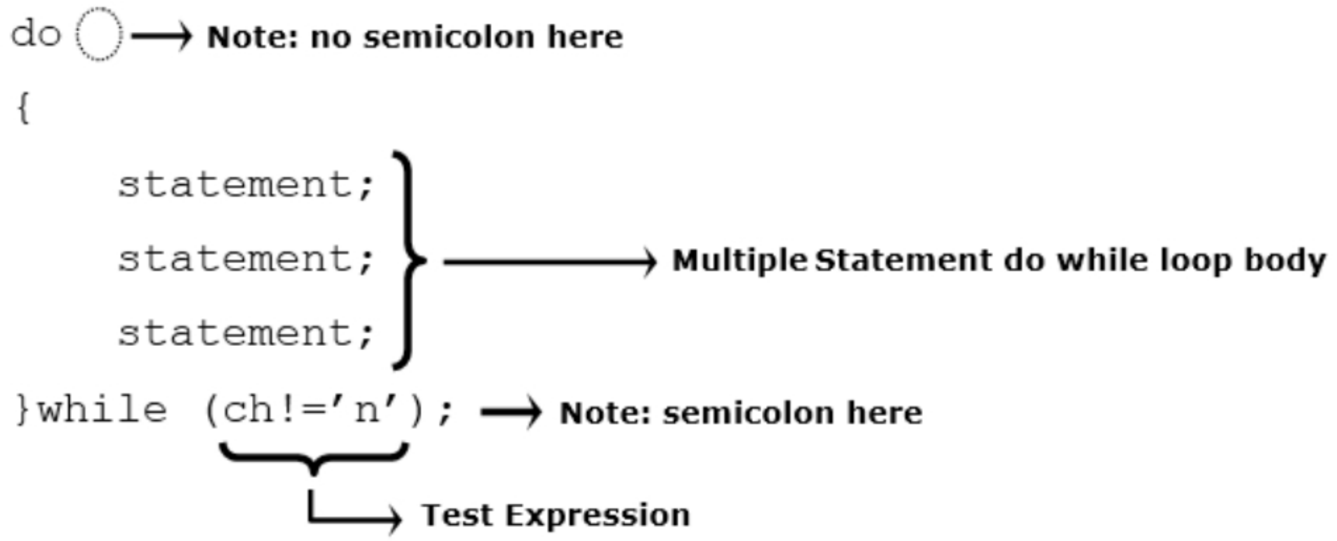
do-while Loop/Statement in C++

- *do-while loop/statement* implements sentinel-controlled repetition logic.
- The do-while loop is used to execute the particular statements of code multiple number of times without writing those statements that much number of time.
- The do-while loop is used when we do not know that how many times the loop will be executed but we know at least one time it is to be executed.
- It repeats statements on the basis of a condition.
- Here the condition is open-ended.
- If the condition is *true*, statements are repeated again.
- If the condition is *false*, the loop is terminated.

do-while Loop/Statement – Syntax

```
do  do
{ statement;
    statement set ;          while( Condition ) ;
} while( Condition ) ;
```

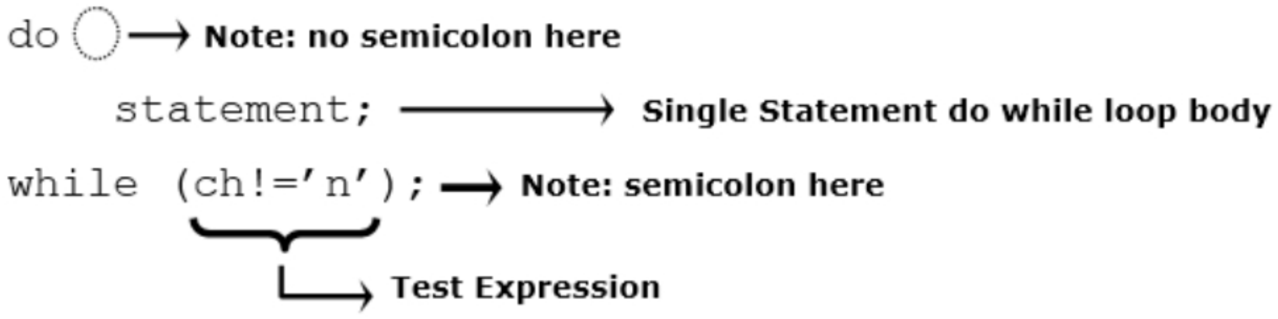
do-while Loop/Statement – Syntax



```
do          → Note: no semicolon here
{
     statement;
     statement;   → Multiple Statement do while loop body
     statement;
}while (ch!='n');  → Note: semicolon here
```

Test Expression

do-while Loop/Statement – Syntax

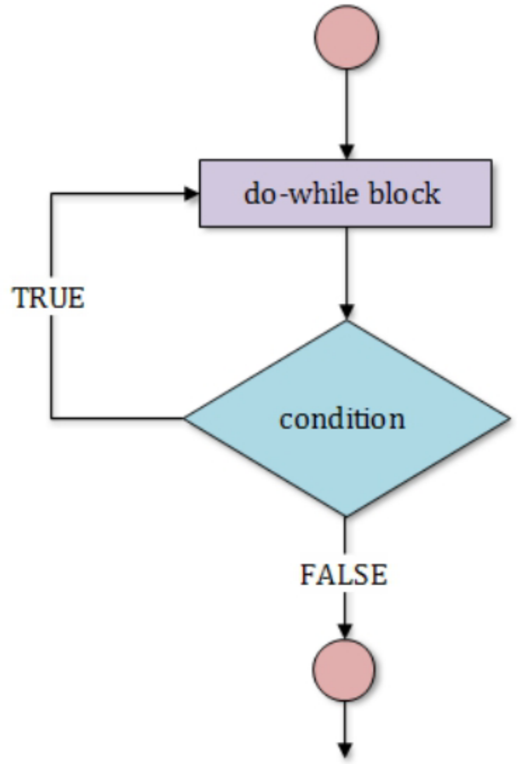


```
do
     statement;
while (ch!='n');
```

do-while Loop/Statement – Syntax

- There are two syntaxes of do-while statement.
- In first syntax we have multiple statements inside the do-while statement.
- In this case it is compulsory to enclose all the statements in the braces **{ }** .
- In second syntax we have just one statement inside the do-while statement.
- In this case it is optional to enclose the statement in the braces **{ }** .
- All the statements enclosed in **{ }** is called as the **do-while block**

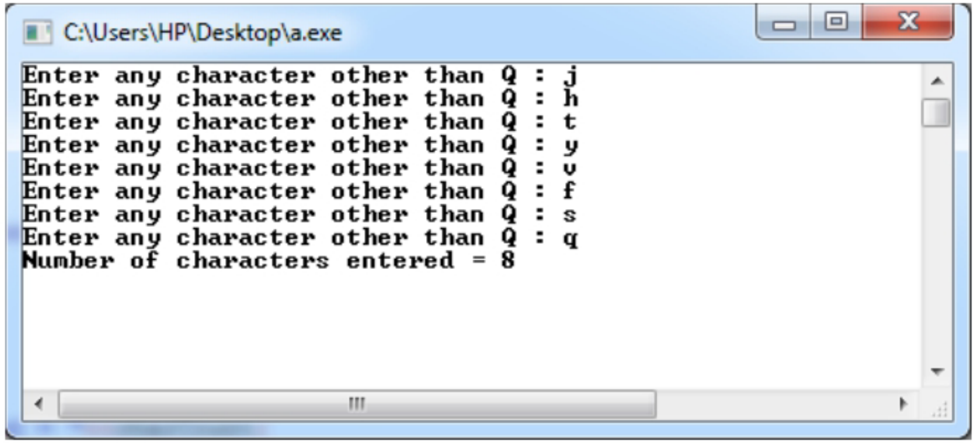
do-while Loop/Statement – Flow Chart



do-while Loop/Statement - Example

Problem Statement: The user continuously enters a character, the program stops when the user enters ‘q’ character and finally displays the number of characters entered.

```cpp
#include<iostream>
#include<conio.h>

using namespace std;

int main()
{
    char ch;
    int charCount = 0;

    do
    {
        cout<<"Enter any character other than Q : ";
        ch = getche();
        cout<<endl;
        charCount++;
    }while(ch!='Q'&&ch!='q');

    cout<<"Number of characters entered = "<<charCount;
    getch();
    return 0;
}
```



C:\Users\HP\Desktop\a.exe

```
Enter any character other than Q : j
Enter any character other than Q : h
Enter any character other than Q : t
Enter any character other than Q : y
Enter any character other than Q : v
Enter any character other than Q : f
Enter any character other than Q : s
Enter any character other than Q : q
Number of characters entered = 8
```

Decision to Choose Between Iterative Control Structure

- If you exactly know, how any times the statements are to be repeated then use **for loop**.
- If you do not know, exactly how any times the statements are to be repeated then use **while loop**.
- If you do not know, exactly how any times the statements are to be repeated but you know that at least once they are to be executed then use **do-while loop**.

Nested Loops

- A nested loop is a loop with in a loop, an inner loop with in the body of an outer one.
- For every iteration of outer loop multiple iterations of inner loop are executed.
- If a for loop is written inside another for loop, it is said as nested for loop.
- Similarly, we have nested while and nested do-while loops.
- Different types of loops can also be nested together like, a while loop inside a for loop; for loop inside a while loop; while loop inside a do-while loop, do-
  while loop inside a for loop etc.

Nested for Loop

Outer Loop

Inner Loop

```
for(initialization; test; increment)
{
    //statements for outer loop
    for(initialization; test; increment) {
        //statements for inner loop
    }
}
```

Nested while Loop

Outer Loop ←

Inner Loop ←

```
while( Condition )
{
    //statements for outer loop while(
    Condition )
    {
        //statements for inner loop
    }
}
```

Nested do-while Loop

```
Outer Loop  ←  do
               {
                   //statements for outer loop
            ←      do
Inner Loop         {
                       //statements for inner loop
                   } while( Condition ) ;
               } while( Condition ) ;
```

Nested Loops - Example

Problem Statement: Generate and display the following pattern of number:

1

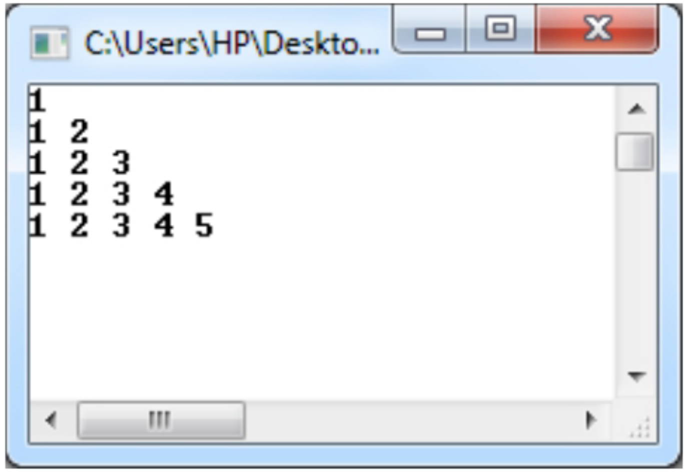
1 2

1 2 3

1 2 3 4

1 2 3 4 5

```cpp
#include<iostream>
#include<conio.h>

using namespace std;

int main()
{
    for(int i=1; i<=5; i++)
    {
        for(int j=1; j<=i; j++)
        {
            cout<<j<<" ";
        }
        cout<<endl;
    }

    getch();
    return 0;
}
```



```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

Program Examples

for and nested for statements

Program Example 01

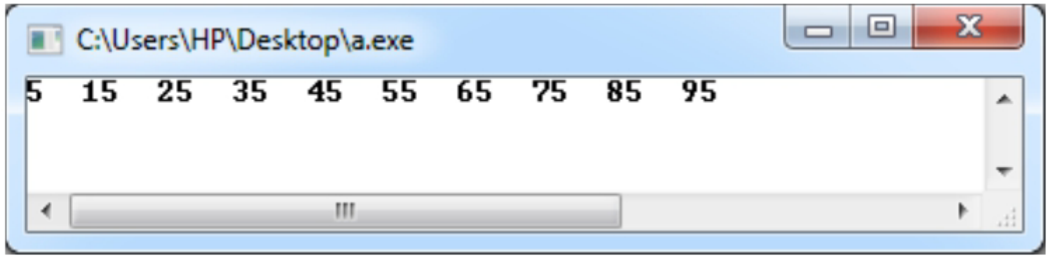
Problem Statement:

Write a program in C++ that generates and displays all the odd multiples of 5 in the range of 1 and 100.

Program Example 01

```
#include<iostream>
#include<conio.h>

using namespace std;

int main()
{
    for(int i=1; i<=100; i++)
    {
        if( (i%2)!=0 && (i%5)==0)
            cout<<i<<"  ";
    }

    getch();
    return 0;
}
```

Program Example 01


C:\Users\HP\Desktop\a.exe

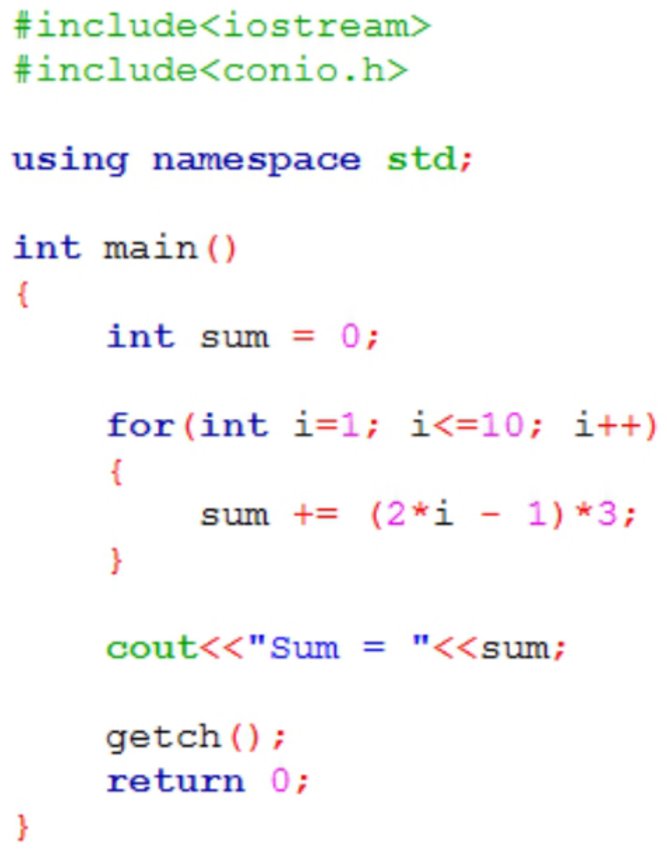
```
5   15   25   35   45   55   65   75   85   95
```

Program Example 02

Problem Statement:

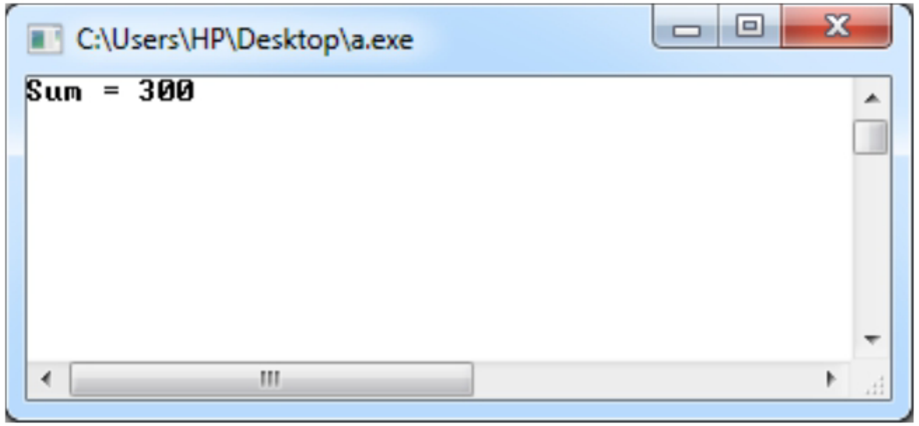
Write a program in C++ that displays the sum of first 10 odd multiples of 3.

Program Example 02



```
#include<iostream>
#include<conio.h>

using namespace std;

int main()
{
    int sum = 0;

    for(int i=1; i<=10; i++)
    {
        sum += (2*i - 1)*3;
    }

    cout<<"Sum = "<<sum;

    getch();
    return 0;
}
```

Program Example 02



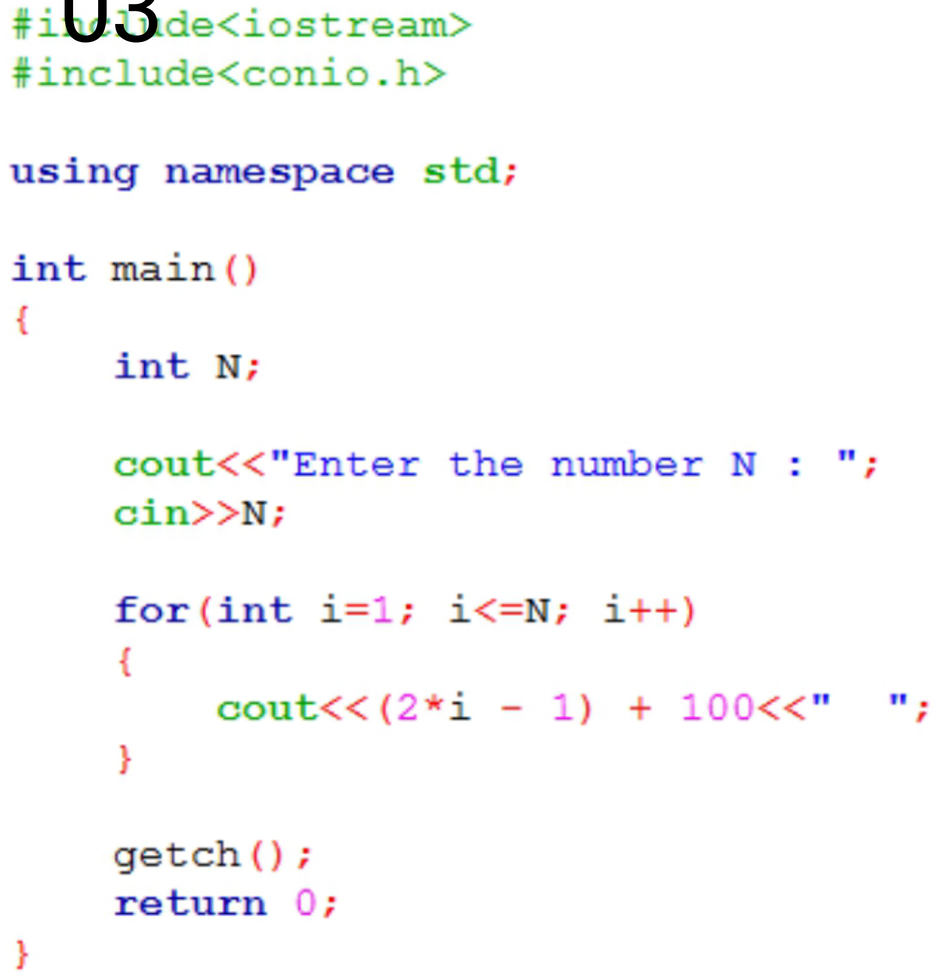
```
Sum = 300
```

Program Example 03

Problem Statement:

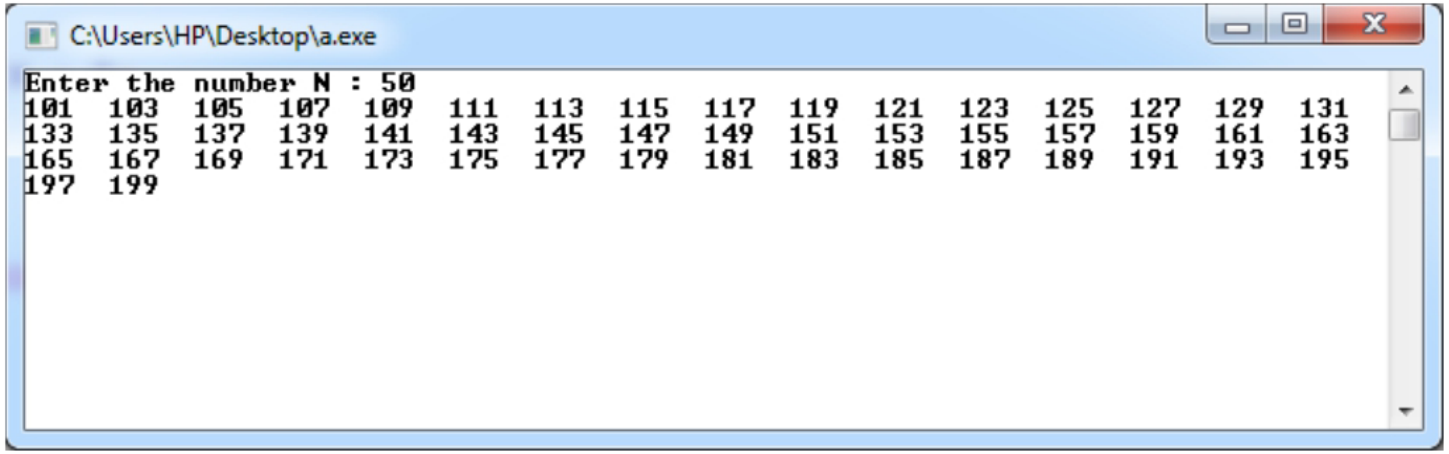
Write a program in C++ that generates and displays the first *N* three digit odd numbers. Whereas the number *N* is provided by the user.

Program Example 03



```
#include<iostream>
#include<conio.h>

using namespace std;

int main()
{
    int N;

    cout<<"Enter the number N : ";
    cin>>N;

    for(int i=1; i<=N; i++)
    {
        cout<<(2*i - 1) + 100<<"  ";
    }

    getch();
    return 0;
}
```

Program Example 03


C:\Users\HP\Desktop\a.exe

```
Enter the number N : 50
101   103   105   107   109   111   113   115   117   119   121   123   125   127   129   131
133   135   137   139   141   143   145   147   149   151   153   155   157   159   161   163
165   167   169   171   173   175   177   179   181   183   185   187   189   191   193   195
197   199
```

Program Example 05

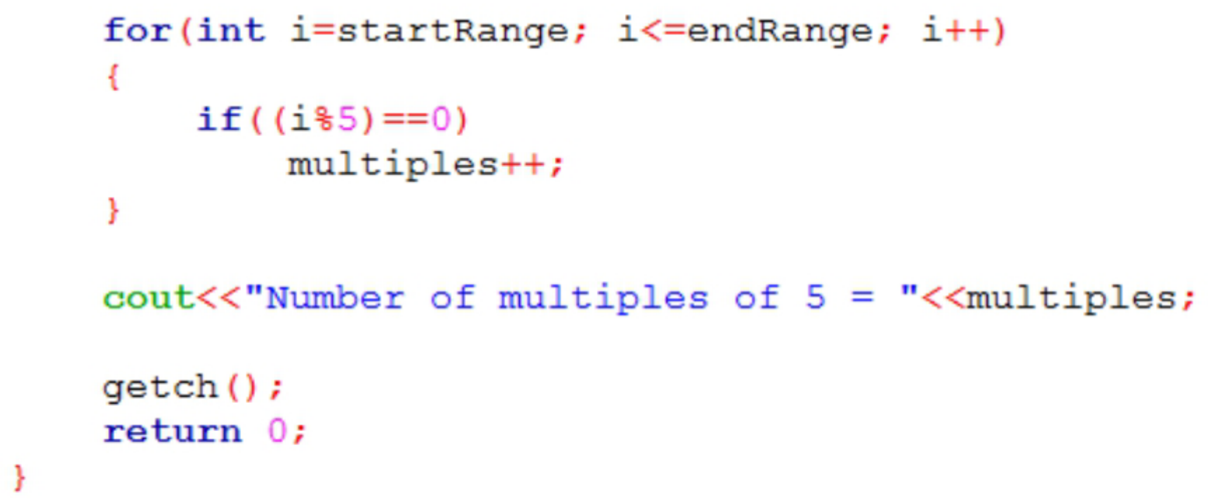
Problem Statement:

Write a program in C++ that asks the user to input the starting number and ending number of the range. The program should display the number of multiples of 5 in between that range.

Program Example 05

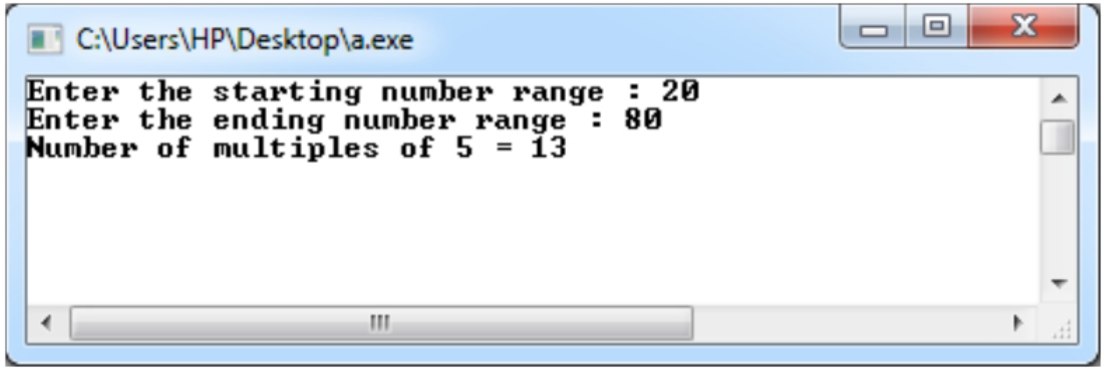
```
#include<iostream>
#include<conio.h>

using namespace std;

int main()
{
    int multiples=0, startRange, endRange;

    cout<<"Enter the starting number range : ";
    cin>>startRange;
    cout<<"Enter the ending number range : ";
    cin>>endRange;
```

Program Example 05



```
    for(int i=startRange; i<=endRange; i++)
    {
        if((i%5)==0)
            multiples++;
    }

    cout<<"Number of multiples of 5 = "<<multiples;

    getch();
    return 0;
}
```

Program Example 05



```
Enter the starting number range : 20
Enter the ending number range : 80
Number of multiples of 5 = 13
```

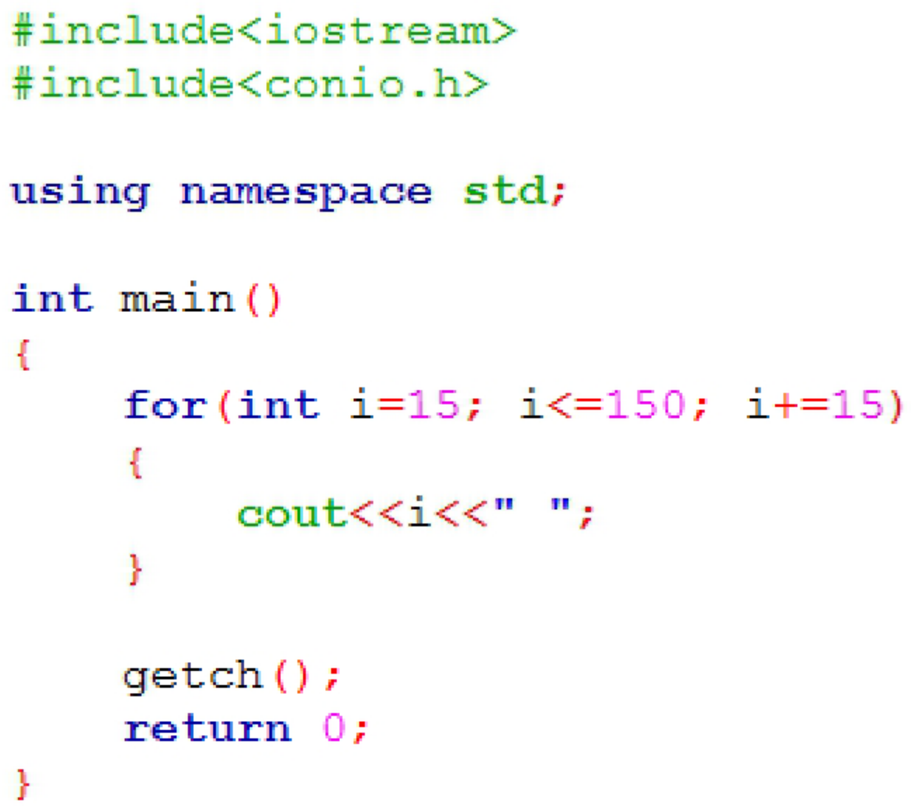
Program Example 06

Problem Statement:

Write a program in C++ that generates and displays the following series of numbers:

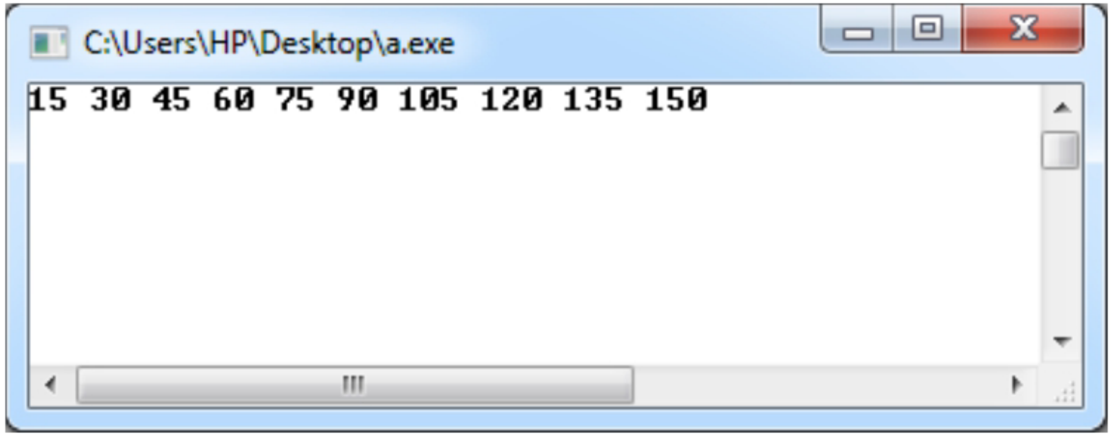
15, 30, 45, 60, 75, 90, 105, 120, 135, 150

Program Example 06



```
#include<iostream>
#include<conio.h>

using namespace std;

int main()
{
    for(int i=15; i<=150; i+=15)
    {
        cout<<i<<" ";
    }

    getch();
    return 0;
}
```

Program Example 06


C:\Users\HP\Desktop\a.exe

```
15 30 45 60 75 90 105 120 135 150
```

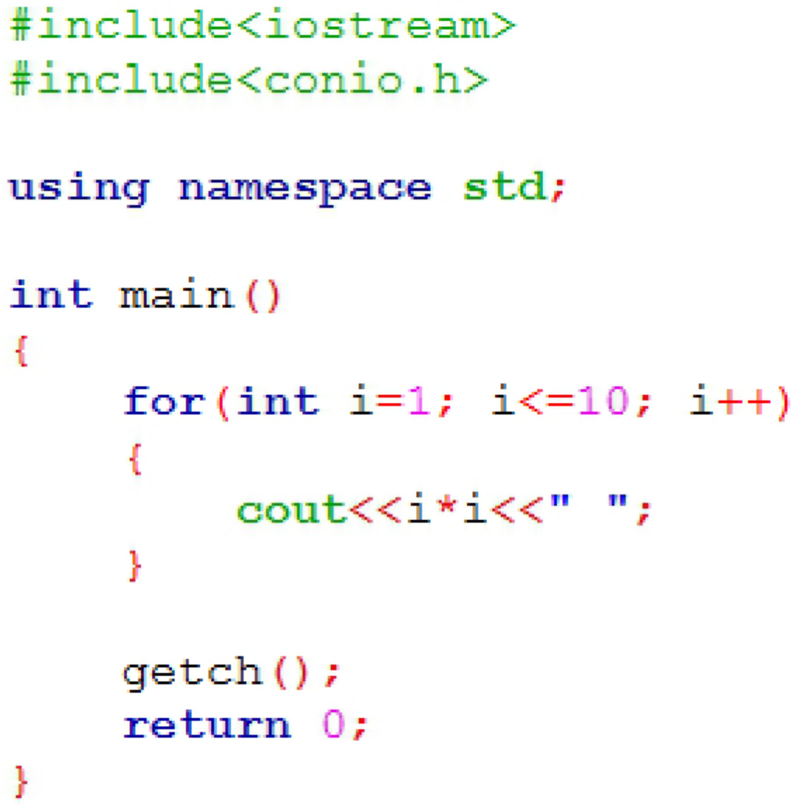
Program Example 07

Problem Statement:

Write a program in C++ that generates and displays the following series of numbers:

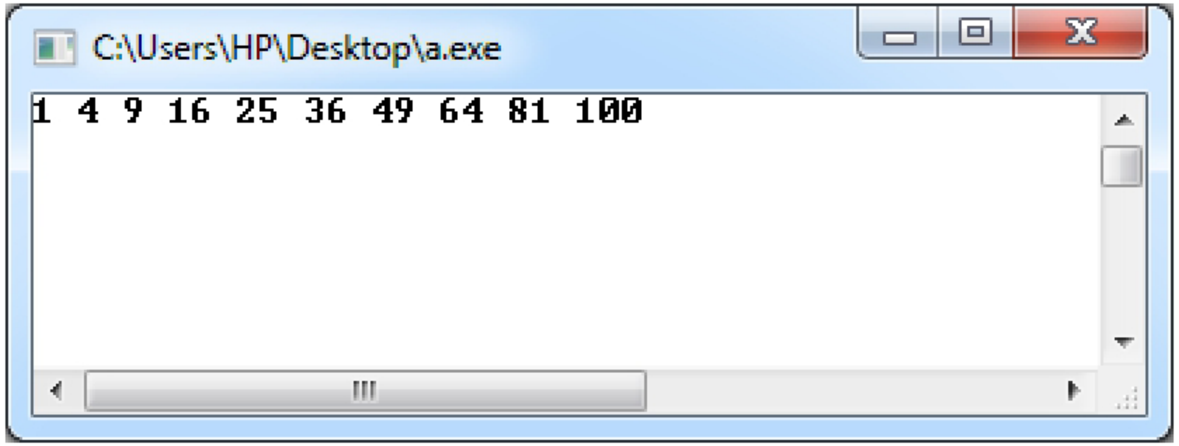
1, 4, 9, 16, 25, 36, 49, 64, 81, 100

Program Example 07



```cpp
#include<iostream>
#include<conio.h>

using namespace std;

int main()
{
    for(int i=1; i<=10; i++)
    {
        cout<<i*i<<" ";
    }

    getch();
    return 0;
}
```

Program Example 07



```
1 4 9 16 25 36 49 64 81 100
```

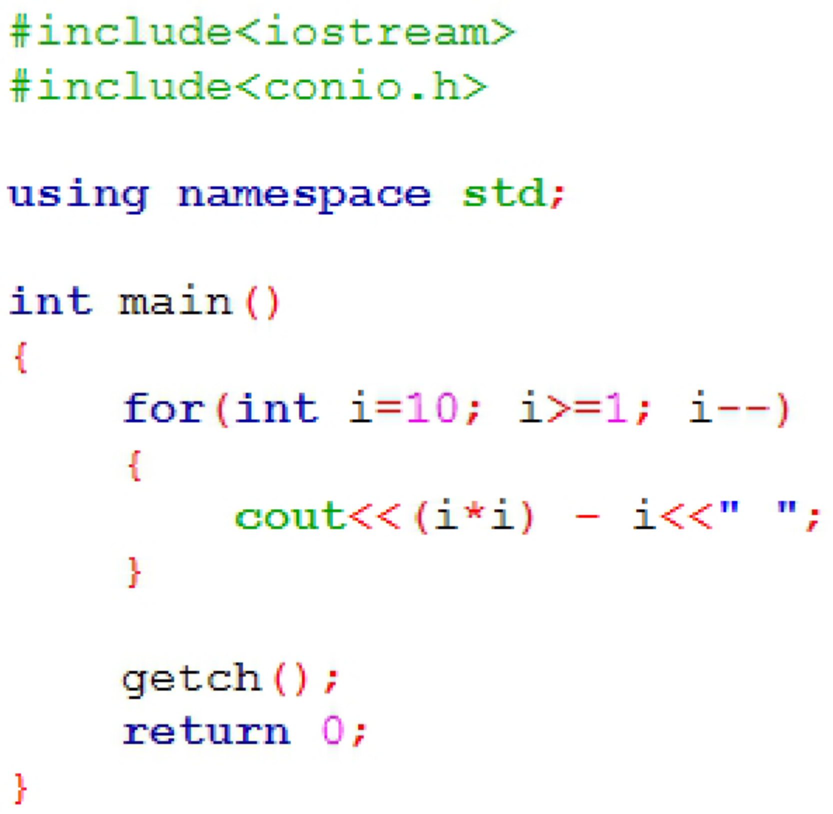
Program Example 08

Problem Statement:

Write a computer program that generates and displays the following series of numbers:

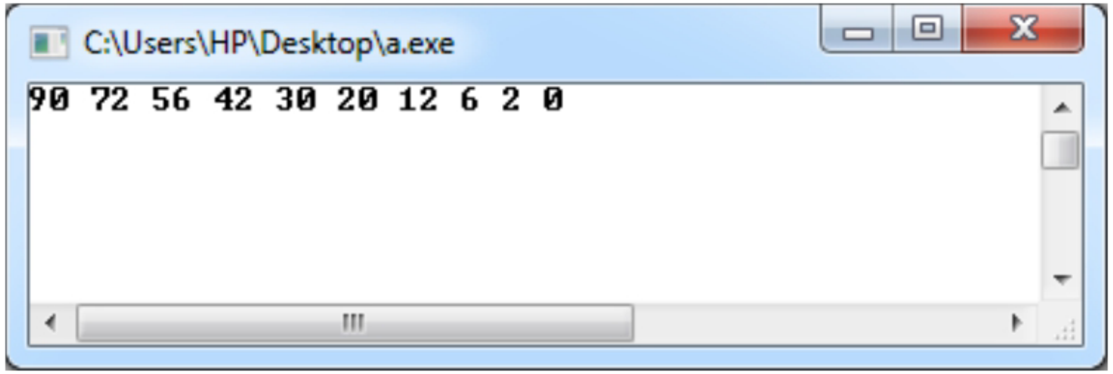
90, 72, 56, 42, 30, 20, 12, 6, 2, 0

Program Example 08



```
#include<iostream>
#include<conio.h>

using namespace std;

int main()
{
    for(int i=10; i>=1; i--)
    {
        cout<<(i*i) - i<<" ";
    }

    getch();
    return 0;
}
```

Program Example 08


C:\Users\HP\Desktop\a.exe

```
90 72 56 42 30 20 12 6 2 0
```

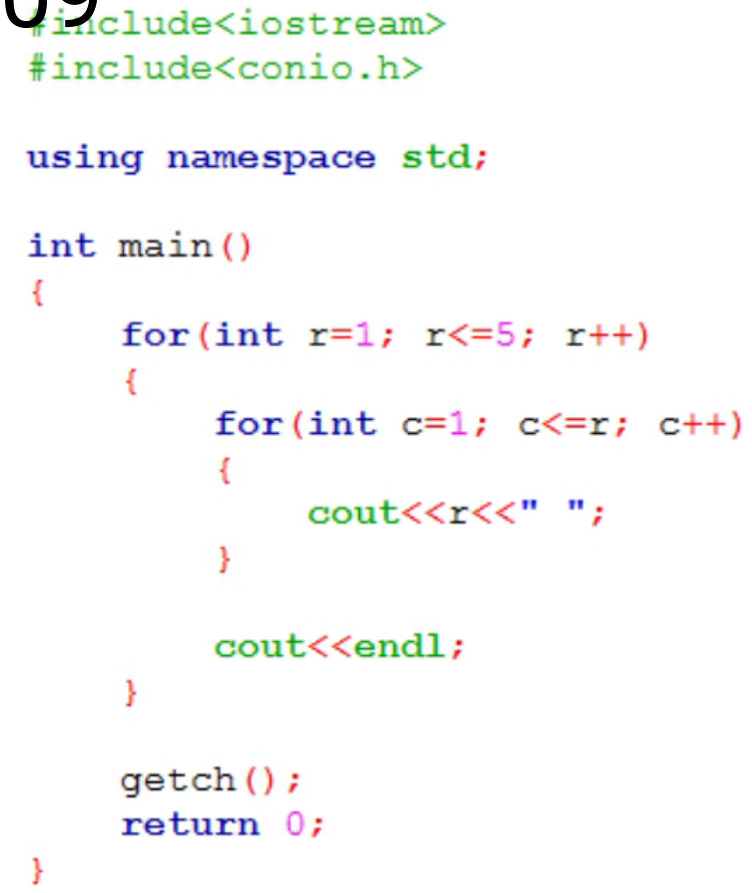
Program Example 09

Problem Statement:

Write a C++ program to print half pyramid using numbers

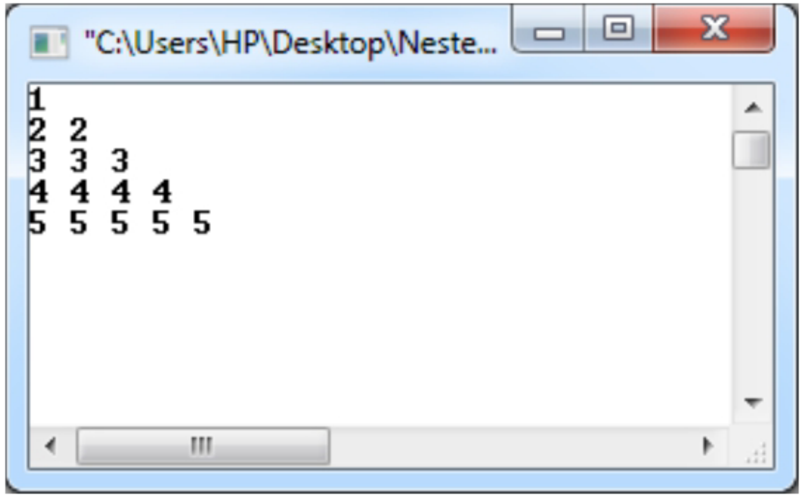
```
1
2 2
3 3 3
4 4  4 4
5 5  5 5  5
```

Program Example 09



```
#include<iostream>
#include<conio.h>

using namespace std;

int main()
{
    for(int r=1; r<=5; r++)
    {
        for(int c=1; c<=r; c++)
        {
            cout<<r<<" ";
        }

        cout<<endl;
    }

    getch();
    return 0;
}
```

Program Example 09



"C:\Users\HP\Desktop\Neste...

```
1
2  2
3  3  3
4  4  4  4
5  5  5  5  5
```

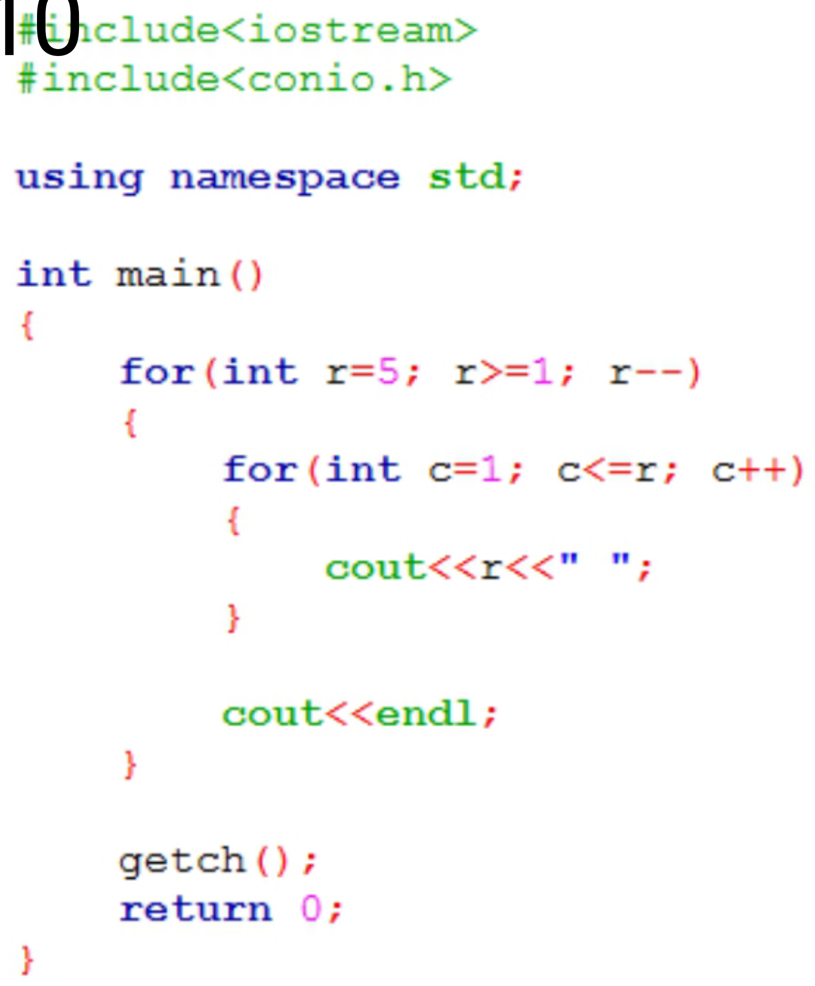
Program Example 10

Problem Statement:

Write a C++ program to print inverted half pyramid as using numbers

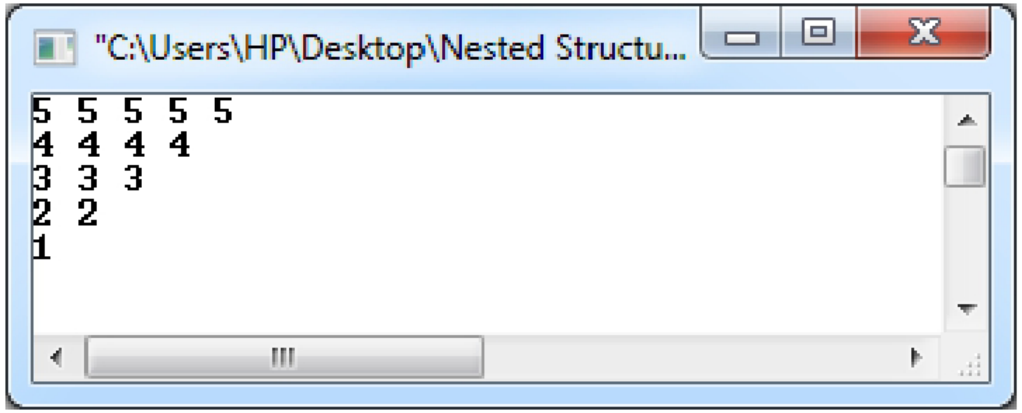
```
5  5  5  5  5
4  4  4  4
3  3  3
2  2
1
```

Program Example 10



```
#include<iostream>
#include<conio.h>

using namespace std;

int main()
{
    for(int r=5; r>=1; r--)
    {
        for(int c=1; c<=r; c++)
        {
            cout<<r<<" ";
        }

        cout<<endl;
    }

    getch();
    return 0;
}
```

Program Example 10


"C:\Users\HP\Desktop\Nested Structu...

```
5  5  5  5  5
4  4  4  4
3  3  3
2  2
1
```