



E1123 Computer Programming (a)

(Fall 2020)



Control Structures (Repetition)

INSTRUCTOR

DR / AYMAN SOLIMAN

➤ Why Is Repetition Needed?

- Repetition allows you to efficiently use variables
- Can input, add, and average multiple numbers using a limited number of variables
- For example, to add five numbers:
 - ❑ Declare a variable for each number, input the numbers and add the variables together
 - ❑ Create a loop that reads a number into a variable and adds it to a variable that contains the sum of the numbers

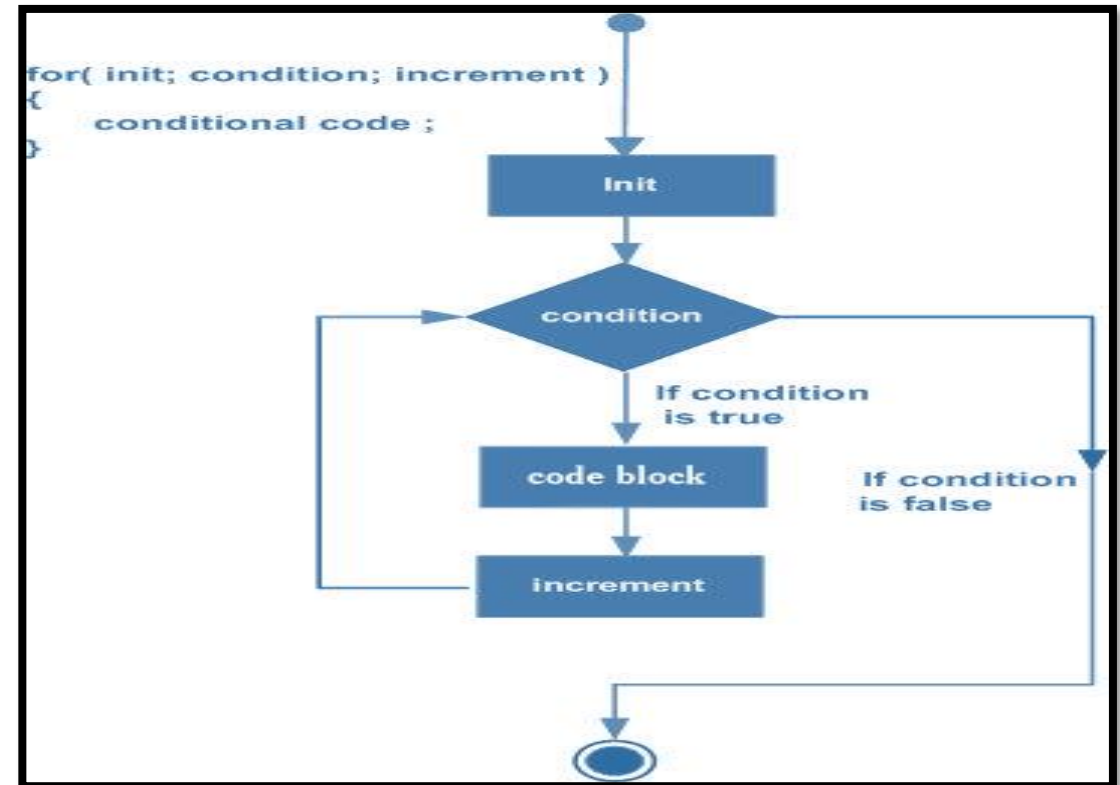
➤ There are four basic types of loops which are:

Loop Type	Description
For loop	Execute a sequence of statements multiple times and abbreviates the code that manages the loop variable.
While loop	Repeats a statement or group of statements while a given condition is true. It tests the condition before executing the loop body
Do-while loop	Like a while statement, except that it tests the condition at the end of the loop body
nested loops	You can use one or more loop inside any another while, for or do-while loop.

➤ 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.
- The syntax of a for loop in C++ is:

```
for ( initial; condition; increment )  
{  
    statement(s);  
}
```



➤ The flow of control in a for loop

- The initial step is executed first, and only once. This step allows you to declare and initialize any loop control variables. You are not required to put a statement here, if 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 flow of control jumps to the next statement just after the for loop.

➤ The flow of control in a for loop

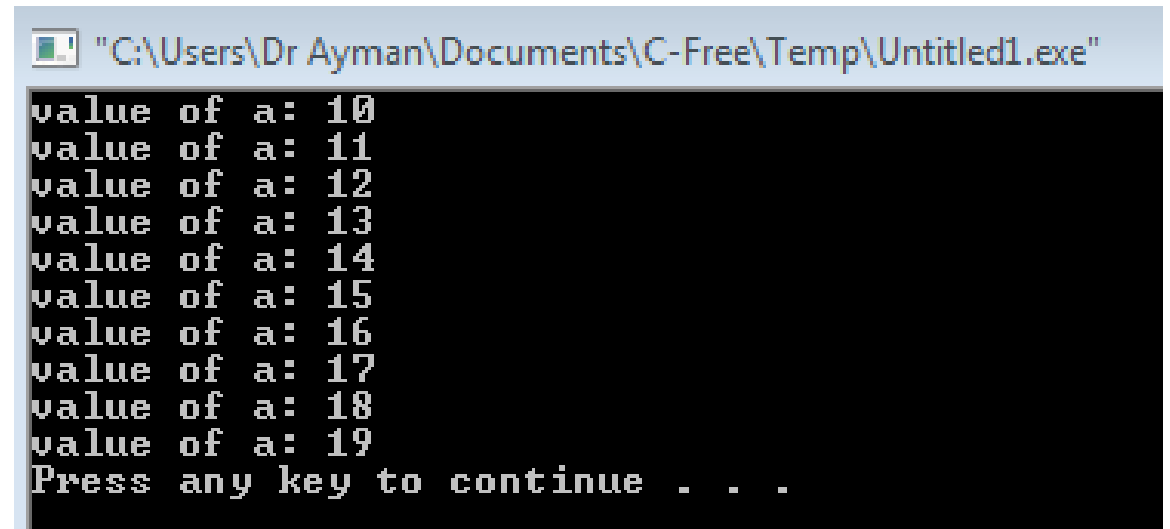
- After the body of the for loop executes, the flow of control jumps back up to the increment statement. This statement allows you to update any loop control variables. This statement can be left blank, as long as a semicolon
- The condition is now evaluated again. If it is true, the loop executes and the process repeats itself (body of loop, then increment step, and then again condition).
After the condition becomes false, the for loop terminates.

➤ Example 1

Write a program to print numbers from 10 to 19 on screen

```
#include <iostream>
using namespace std;

int main ()
{
for( int a = 10; a < 20; a = a + 1 )
    {
        cout << "value of a: " << a << endl;
    }
}
```

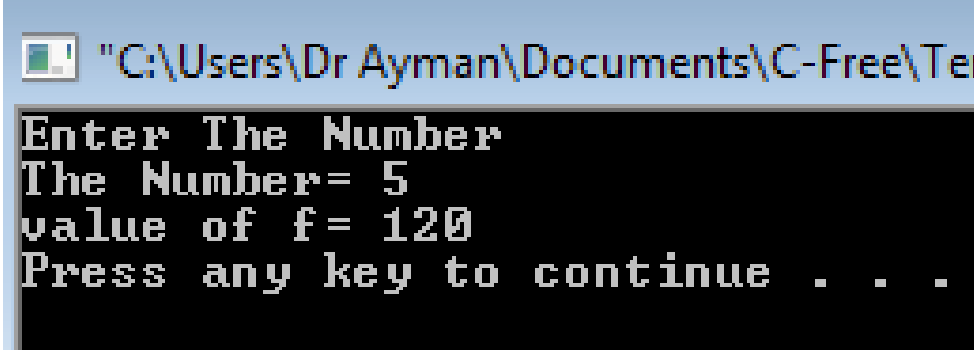


```
"C:\Users\Dr Ayman\Documents\C-Free\Temp\Untitled1.exe"
value of a: 10
value of a: 11
value of a: 12
value of a: 13
value of a: 14
value of a: 15
value of a: 16
value of a: 17
value of a: 18
value of a: 19
Press any key to continue . . .
```

➤ Example 2

Write a program to calculate the factorial of the number (a).

```
#include <iostream.h>
int main()
{ int f=1,a,i;
cout << "Enter The Number " << endl;
cout << "The Number= " ;
cin>>a;
    for(i=a; i >= 1; i = i-1 )
    {
f=f*i;
    }
    cout << "value of f= " << f<< endl;
return 0;}
```

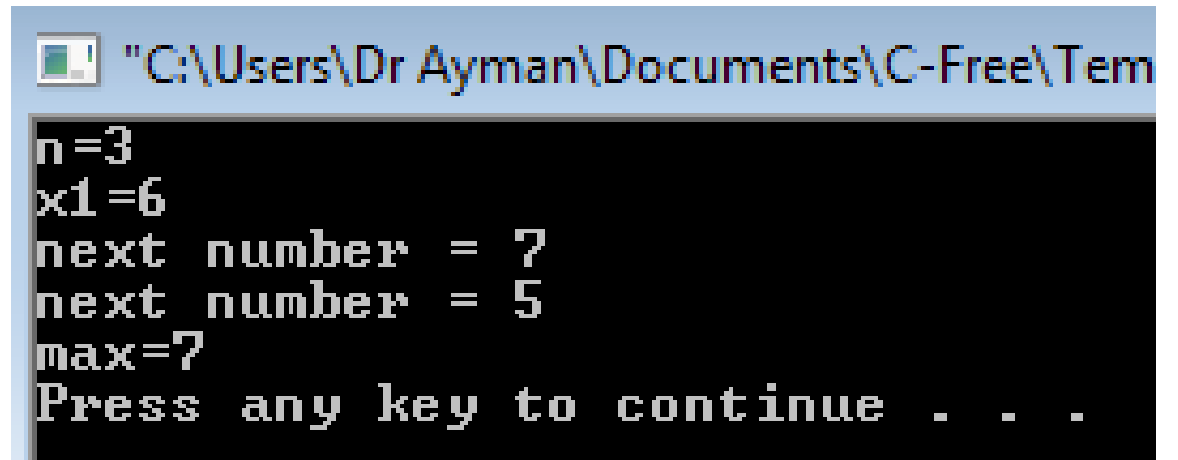


```
"C:\Users\Dr Ayman\Documents\C-Free\Ter
Enter The Number
The Number= 5
value of f= 120
Press any key to continue . . .
```


➤ Example 3

Write a program to calculate the maximum value for a group of (n) numbers.

```
#include <iostream.h>
int main()
{int n,max,x,i;
cout<<"n=";
cin>>n;
cout<<"x1=";
cin>>x;
max=x;
for(i=2;i<= n;i++)
{cout<<"next number = ";
    cin>>x;
    if(x>max)
    max=x;}
cout<<"max="<<max<<endl;
return 0;}
```



The screenshot shows a terminal window with the following output:

```
n=3
x1=6
next number = 7
next number = 5
max=7
Press any key to continue . . .
```

➤ The command goto

- The goto statement is a control flow statement that causes the CPU to jump to another spot in the code. This spot is identified through use of a statement label.
- The syntax of a goto statement in C++ is:

`goto label;`

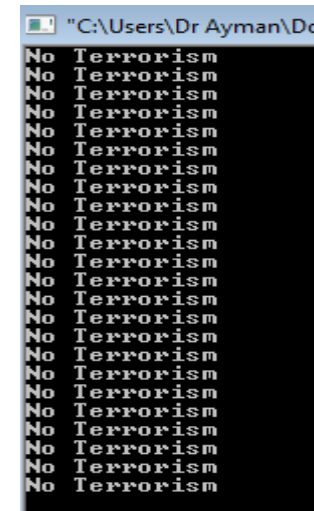
`label: statement;`

➤ The Infinite Loop

- A loop becomes infinite loop if a condition never becomes false.
- The for loop is used for this purpose.
- You can make an infinite loop by leaving the conditional expression empty.

```
#include <iostream.h>

int main()
{
    for(;;)
        cout<<"No Terrorism"<<endl;
        return 0;
}
```

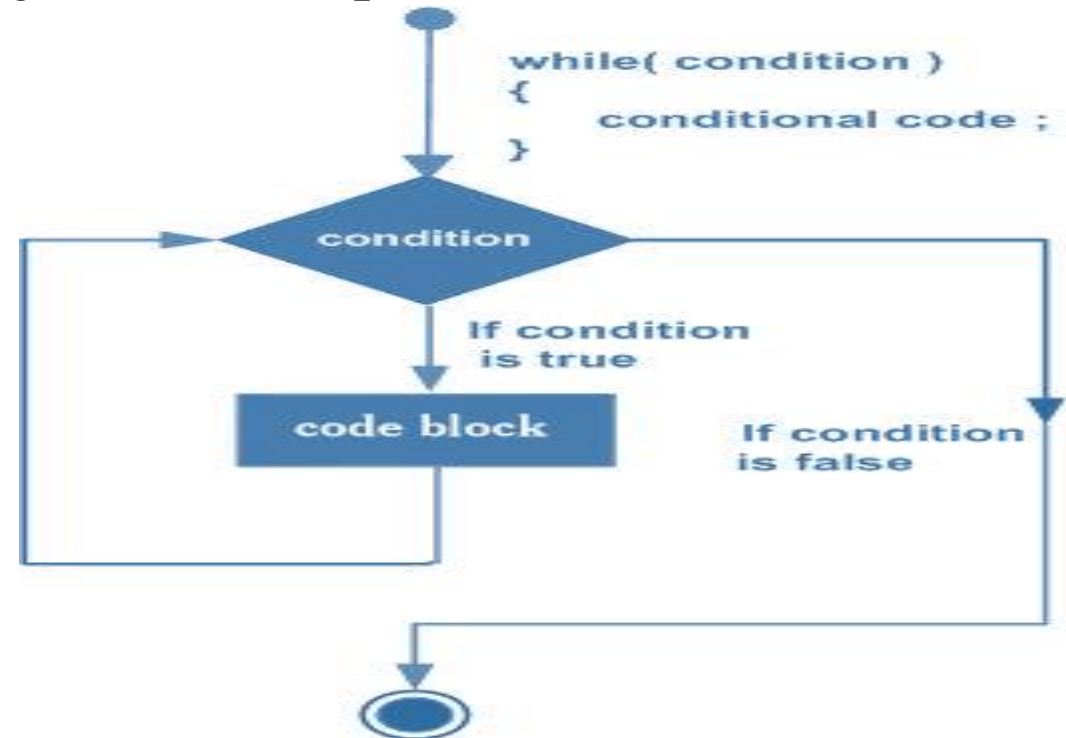


The screenshot shows a terminal window titled "C:\Users\Dr Ayman\De" with a black background and white text. The text consists of the phrase "No Terrorism" repeated on multiple lines, demonstrating the output of an infinite loop.

➤ While Loop

- The condition may be any expression
- The loop iterates while the condition is true.
- When the condition becomes false, program control passes to the line immediately following the loop.
- The syntax of a while loop in C++ is:

```
while(condition)  
{  
    statement(s);  
}
```



➤ Example 1

Consider the following C++ program segment:

```
i = 0; //Line 1
```

```
while (i <= 20) //Line 2
```

```
{
```

```
    cout << i << " "; //Line 3
```

```
    i = i + 5; //Line 4
```

```
}
```

```
cout << endl;
```

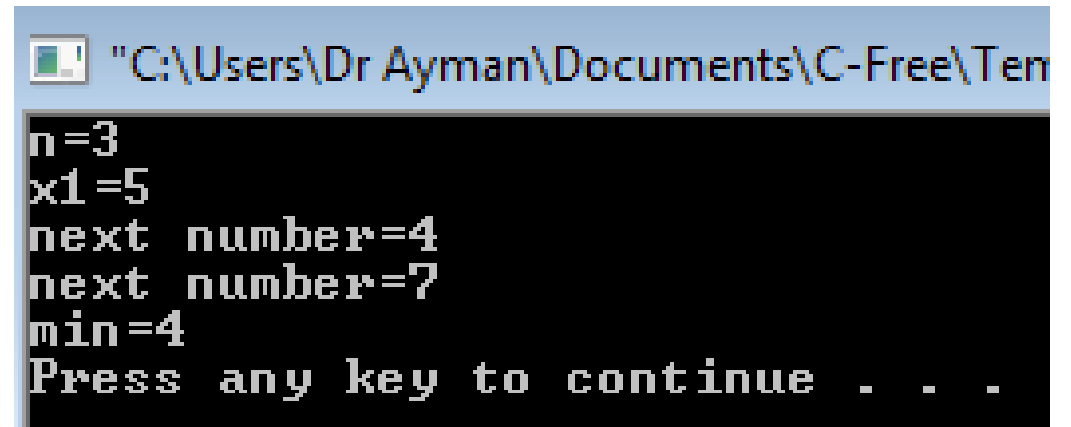
Sample Run:

```
0 5 10 15 20
```

➤ Example 2

➤ Write a program to calculate the minimum value for a group of (n) numbers.

```
#include <iostream.h>
int main()
{int n,min,x,i=2;
cout<<"n=";
cin>>n;
cout<<"x1=";
cin>>x;
min=x;
while(i<=n)
{      cout<<"next number=";
      cin>>x;
      if(x<min)
      min=x;
      i++;}
cout <<"min="<<min<<endl;
      return 0;}
```



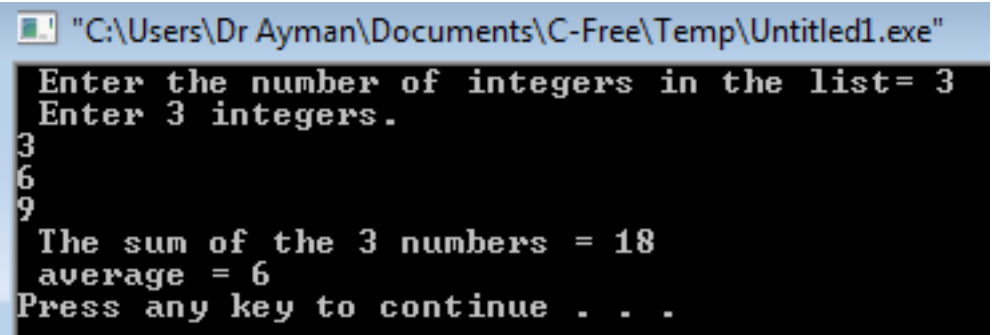
The screenshot shows a Windows command prompt window with the following text:

```
"C:\Users\Dr Ayman\Documents\C-Free\Ten
n=3
x1=5
next number=4
next number=7
min=4
Press any key to continue . . .
```

➤ Example 3

- Write a program to add a group of (n) numbers are different in value and calculate the average of these numbers.

```
#include <iostream.h>
int main()
{int n, number, sum=0, i=1;
float average;
ccc:
cout << " Enter the number of integers in the list= ";
cin >> n;
if(n==0)
goto ccc;
cout << " Enter " << n<< " integers." << endl;
while (i <= n)
{cin >> number;
sum = sum + number;
i++; }
cout << " The sum of the " << n<< " numbers = " << sum << endl;
average = ( sum / n );
cout << " average = "<< average << endl;}
```



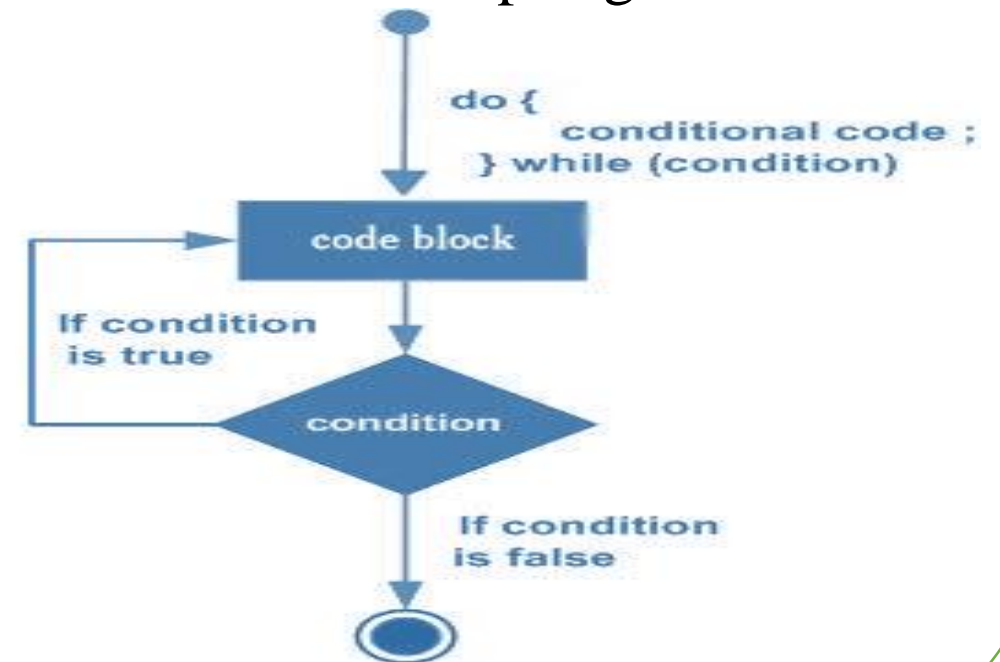
The screenshot shows a Windows command prompt window titled "C:\Users\Dr Ayman\Documents\C-Free\Temp\Untitled1.exe". The program prompts the user to enter the number of integers in the list, which is 3. It then prompts for 3 integers: 3, 6, and 9. The program outputs the sum of these numbers as 18 and the average as 6. It ends with a prompt to press any key to continue.

```
"C:\Users\Dr Ayman\Documents\C-Free\Temp\Untitled1.exe"
Enter the number of integers in the list= 3
Enter 3 integers.
3
6
9
The sum of the 3 numbers = 18
average = 6
Press any key to continue . . .
```

➤ Do While Loop

- Unlike for and while loops, which test the loop condition at the top of the loop, the do...while loop checks its condition at the bottom of the loop.
- A do...while loop is like a while loop, except that a do...while loop is guaranteed to execute at least one time.
- **The syntax of a do...while loop in C++ is:**

```
do  
{  
    statement(s);  
}while( condition );
```

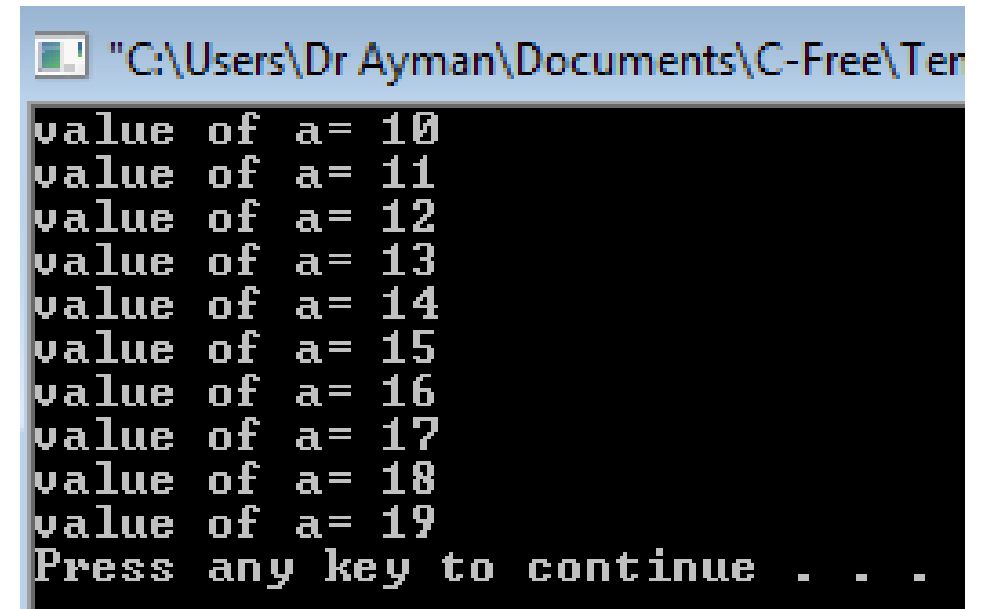


➤ Example 1

➤ Write a program to print numbers from 10 to 19 on screen

```
#include <iostream>
using namespace std;
```

```
int main ()
{
    int a = 10;
do
    {
        cout << "value of a= " << a << endl;
        a = a + 1;
    }while( a < 20 );
}
```

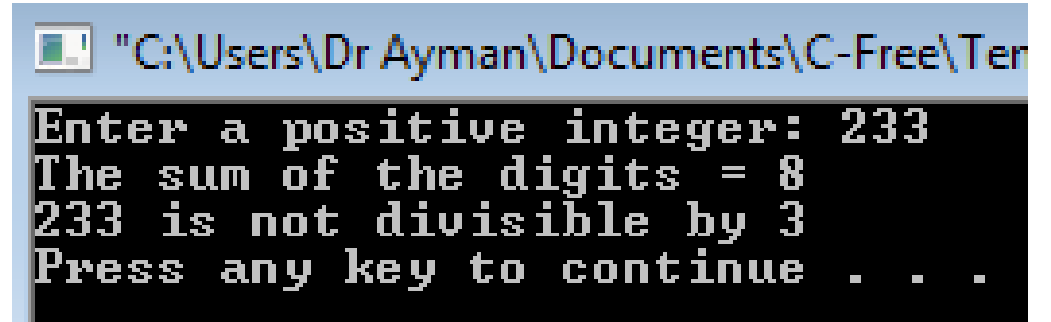


```
"C:\Users\Dr Ayman\Documents\C-Free\Ter
value of a= 10
value of a= 11
value of a= 12
value of a= 13
value of a= 14
value of a= 15
value of a= 16
value of a= 17
value of a= 18
value of a= 19
Press any key to continue . . .
```

➤ Example 2

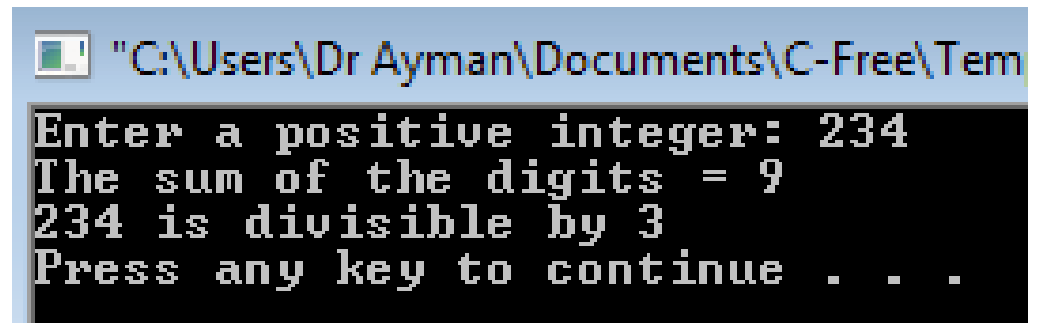
➤ Write a program to test Divisibility by 3

```
#include <iostream.h>
int main()
{int num, x, sum;
cout << "Enter a positive integer: ";
cin >> num;
if(num<0)
num=num*-1;
x = num;
sum = 0;
do
{
sum = sum + num % 10;
num = num / 10;
}
while (num > 0);
cout << "The sum of the digits = " << sum << endl;
if (sum % 3 == 0)
cout << x << " is divisible by 3" << endl;
else
cout << x << " is not divisible by 3" << endl;}
```



A screenshot of a Windows command prompt window. The title bar shows the file path: "C:\Users\Dr Ayman\Documents\C-Free\Tem...". The terminal output is as follows:

```
Enter a positive integer: 233
The sum of the digits = 8
233 is not divisible by 3
Press any key to continue . . .
```



A screenshot of a Windows command prompt window. The title bar shows the file path: "C:\Users\Dr Ayman\Documents\C-Free\Tem...". The terminal output is as follows:

```
Enter a positive integer: 234
The sum of the digits = 9
234 is divisible by 3
Press any key to continue . . .
```

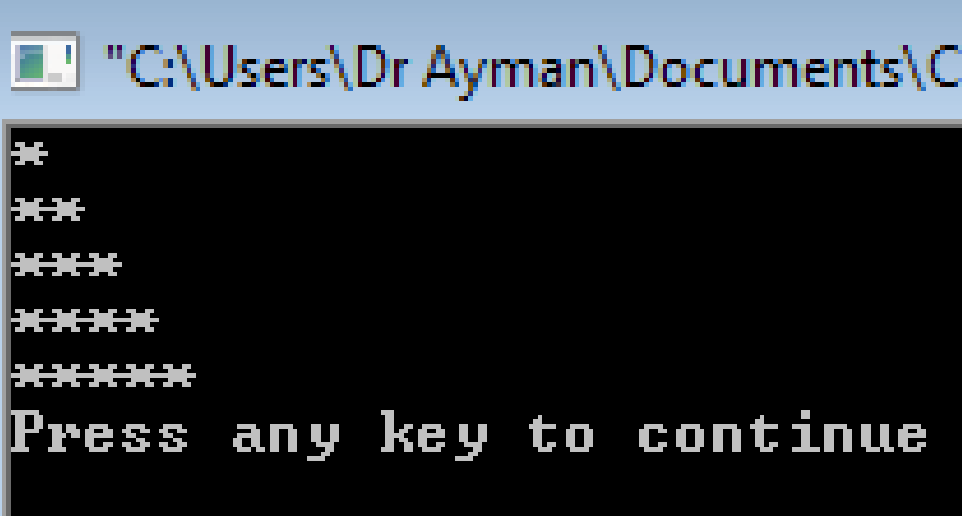
➤ Nested Control Structures (Example 1)

➤ To create the following pattern:

```
*  
**  
***  
****  
*****
```

➤ We can use the following code:

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



```
"C:\Users\Dr Ayman\Documents\C  
**  
***  
****  
*****  
Press any key to continue
```

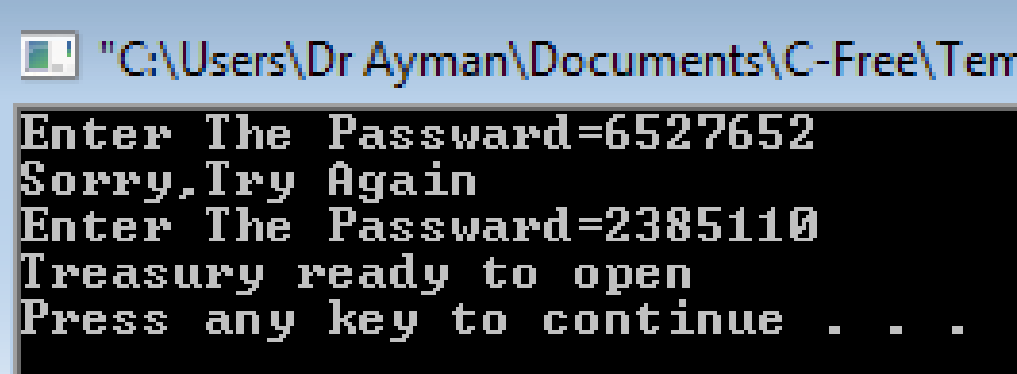
➤ Nested Control Structures (Example 2)

- During your work in a factory, your boss asked you to work a program using the language C++ to establish a password consists of seven numbers for the treasury of the factory.
- Whereas in the case of entering incorrect password, a message will appear to the user (sorry, try again)
- Whereas in the case of entering correct password, a message will appear to the user (Treasury ready to open)
- Remark a password is (2385110)



➤ Example 2 solution

```
#include <iostream>
using namespace std;
int main()
{
int a;
ccc:
cout<<"Enter The Password=";
cin>>a;
if (a!=2385110)
{
cout<<"Sorry, Try Again"<<endl;
goto ccc;
}
cout<<"Treasury ready to open"<<endl;
}
```



```
"C:\Users\Dr Ayman\Documents\C-Free\Terr
Enter The Password=6527652
Sorry, Try Again
Enter The Password=2385110
Treasury ready to open
Press any key to continue . . .
```

Thank

you

