# Chapter 5

## Loops

Section 5.1–5.16, 5.9

# Introduction

- في البرمجة ، أحيانا نحتاج لإجراء بعض العمليات اكثر من مرة او عدد (n) من المرات

مثال : لو طلبنا طباعة كلمة Hello word 5 مرات .

```
cout << "Hello World!" << endl;
cout << "Hello World!" << endl;
cout << "Hello World!" << endl;
cout << "Hello World!" << endl;
cout << "Hello World!" << endl;
```

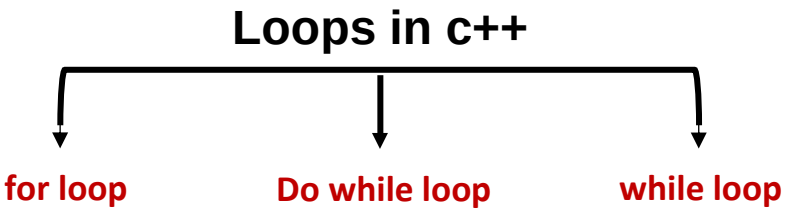
كان الحل بسيط وسهل ، لكن ماذا لو طلبنا نطبعها 100 مرة ؟ او 1000 مرة ؟

100 **times !!**

```
cout << "Hello World!" << endl;
cout << "Hello World!" << endl;
cout << "Hello World!" << endl;
cout << "Hello World!" << endl;
                ...
                ...
                ...
cout << "Hello World!" << endl;
cout << "Hello World!" << endl;
cout << "Hello World!" << endl;
cout << "Hello World!" << endl;
```

هون الكود كان طويل وصعب كتابته ، فكيف نحل هاي المشكلة ؟

عن طريق استخدام ال loops:

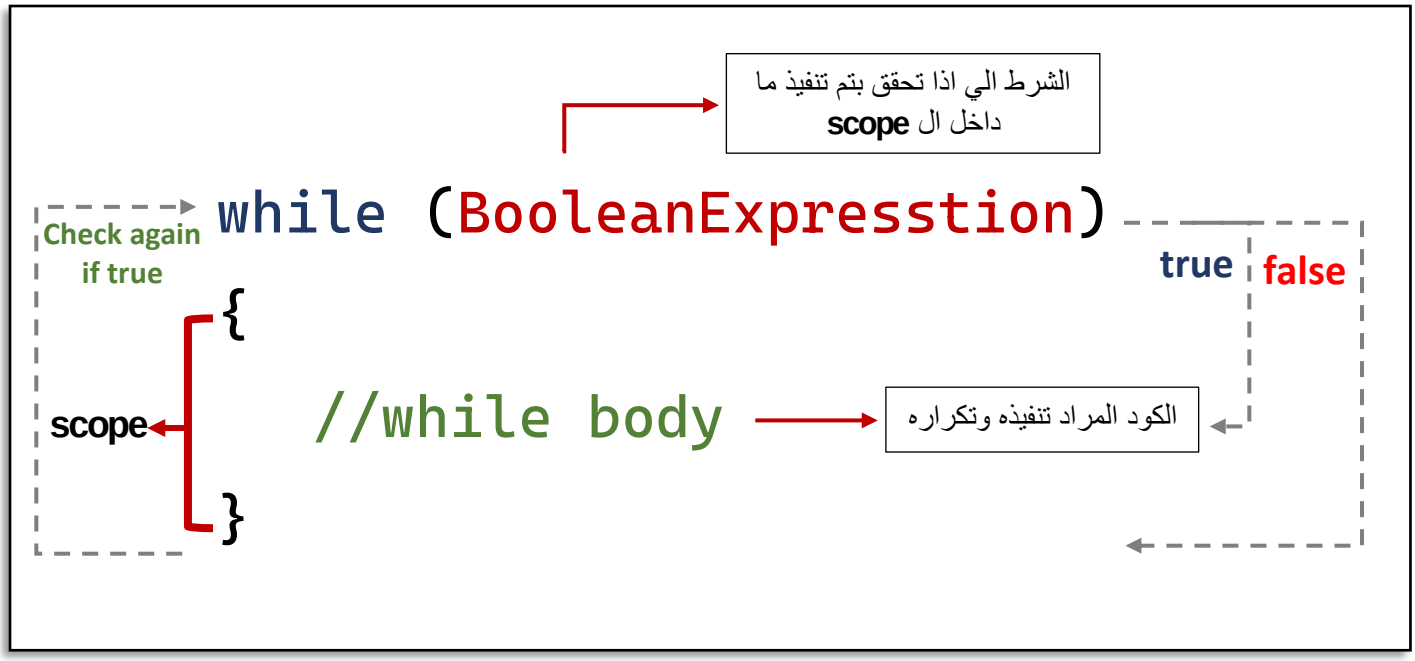



ال loops في ال c++ تستخدم لتنفيذ جزء من الكود عدة مرات. ويوجد 3 أنواع سنتعرف عليها وعلى استخداماتها والفرق بينهم :

# while loop

ابسط اشكال الـ **loop** ، وبتتكون من **condition** و **scope** اذا تحقق الشرط (**true**) ،

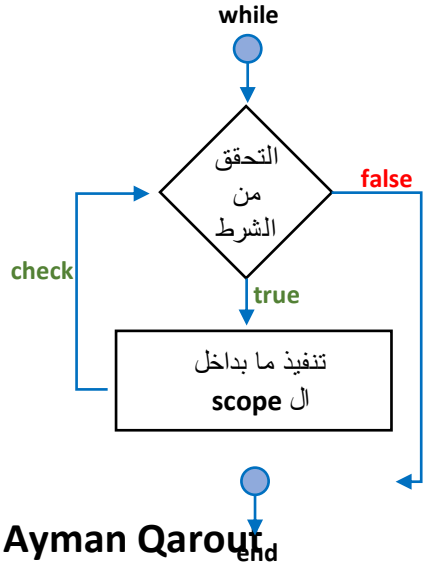
رح يتم تنفيذ ما داخل ال scope واذا ما تحقق الشرط (**false**) فما رح يتم تنفيذه.




اذا كان جواب ال **Boolean Expression** :

**true**: رح يتم تنفيذ ما بداخل ال **scope** وبعدها برجع يتأكد من الشرط مرة أخرى .

**false**: ما رح يتم تنفيذ ال **scope** ورح يكمل باقي الكود طبيعي.




- آلية عمل ال **while** :

التحقق من الشرط ← true ← تنفيذ ما بداخل ال **scope**

التحقق من الشرط مرة اخرى

false ← رح يخرج من ال **while** (ما رح ينفذ ال **scope**)

**Example: write a program that display Hello c++ 3 times.**

```
int count = 0;
while (count < 3)
{
     cout << "Hello C++!\n"
     count++;
}
```

# When:

**count = 0**

```
while (0 < 3)
{
     cout << "Hello C++!\n"
     count++;
}
```

true

count = 1

**Hello C++!**

**count = 1**

```
while (1 < 3)
{
   cout << "HelloC++!\n"
   count++;
}
```

true

count = 2

**Hello C++!**
**Hello C++!**

**count = 2**

```
while (0 < 3)
{
     cout << "Hello C++!\n"
     count++;
}
```

true

count = 3

**Hello C++!**
**Hello C++!**
**Hello C++!**

**count = 3**

```
while (3 < 3)
{
   cout << "Hello C++!\n"
   count++;
}
```

false

**Hello C++!**
**Hello C++!**
**Hello C++!**

**and count = 3**

- زي ما لاحظنا انه دائما ال while الها شرط حتى يتم تنفيذها ، لكن من الإمكان نخلي ال user هو الي يتحكم بعدد اللفات لل while وتسمى بال **controlling a loop**

```
char condition = 'Y';
while (condition == 'Y')
{
  // the loop body

  // ask user to continue or quit:
  cout<<"Enter Y to continue or anything else to quit: ";
  cin >> condition;
}
```

رح يتم تنفيذ الكود وما بداخل ال while loop ، طول ما ال user بدخل **Y** لانه شرط ال while رح يكون true في كل مرة ، لكن اذا دخل أي حرف او رقم اخر الشرط رح يكون false ورح يطلع من ال loop

**Ex: A program that reads and calculates the sum of number of integers.**
**if user enter 0 , you have to stop input , and display the sum .**

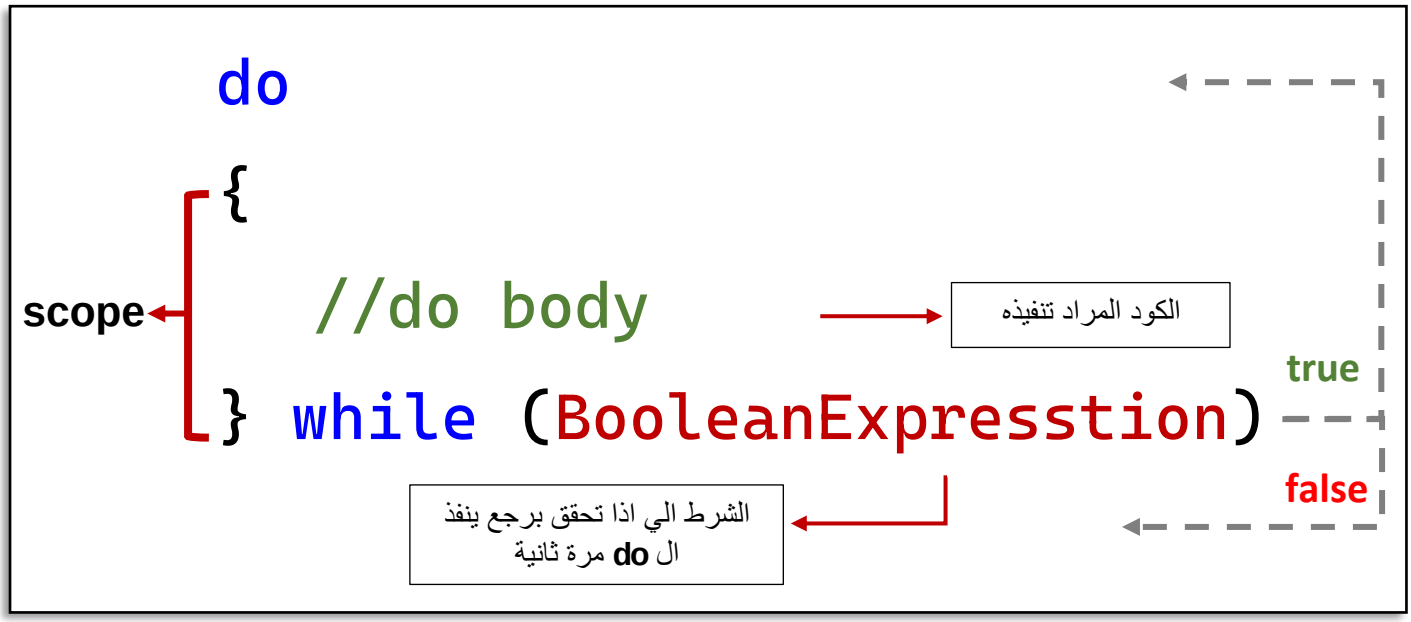
```
int data = 1;
int sum = 0;
while (data != 0) {
     cout << "Enter an integer (Enter 0 to stop):";
     cin >> data;
     sum += data;
}
cout << "The sum = " << sum << endl;
```

.

```
Output :
Enter an integer (Enter 0 to stop): 1
Enter an integer (Enter 0 to stop): 2
Enter an integer (Enter 0 to stop): 5
Enter an integer (Enter 0 to stop): 0
The sum = 8
```

# do/while loop

الـ do/while بتشبه بمبدأ عملها الـ while. لكن الاختلاف انه رح يتم تنفيذ الـ scope لمرة واحدة فقط بدون ما يتحقق من الشرط اذا true ، بعد هيك رح يتحقق من الشرط

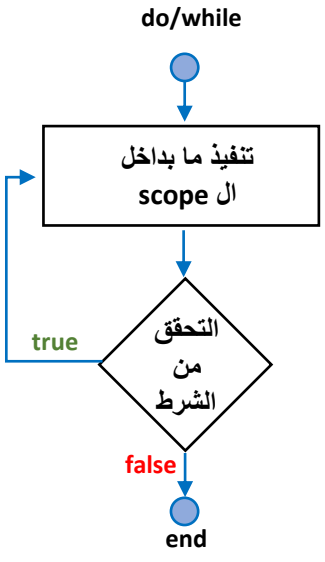



والفائدة من ال do/while اذا كنت بدي انفذ ال body قبل ما اتأكد من الشرط يعني الـ scope بتم تنفيذه مرة واحدة على الأقل .

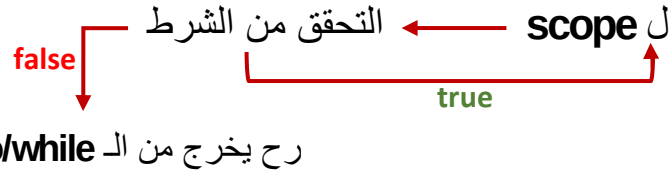
اذا كان جواب ال Boolean Expression :

true: رح يرجع لبداية ال **scope** وينفذه مرة ثانية .

false: ما رح يرجع يعيد ال **scope** وبكمل باقي الكود طبيعي.

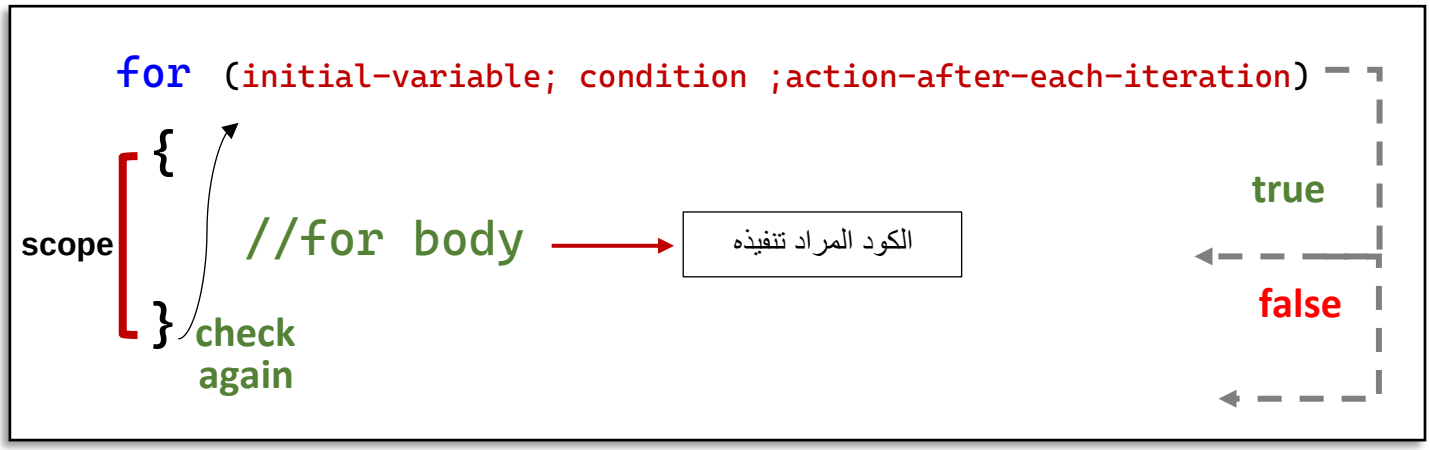



**- آلية عمل ال do/while :**



# for loops

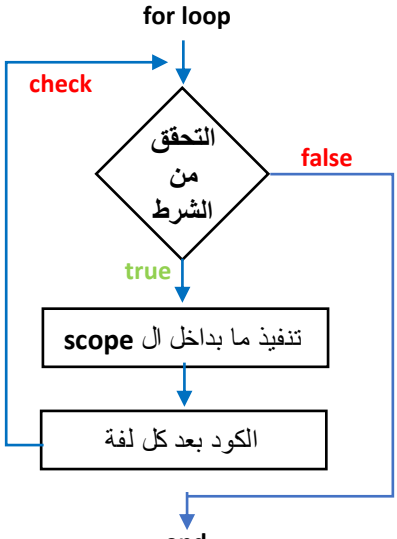
نفس فكرة ومبدأ الـ while لكن بتم استخدام الـ for لما نكون عارفين بالضبط عدد المرات الي بدك تنفذ فيها ال loop.




- **initial-variable:** هي عبارة عن counter variable ، بنحط قيمته الابتدائية بالرقم الي بدك تبلش منه العد ، وبتم تنفيذها اول ما نفوت ال loop ولمرة واحدة فقط.

- **condition:** شرط حتى يتأكد من خلاله بتنفيذ ال scope او لا ، وبتم تحديده على عدد اللفات الي بدك ياها .

- **action after each iteration:** هو كود بتم تنفيذه بعد ما يخلص الـ scope كامل و قبل ما يتم **التأكد** من الشرط مرة أخرى

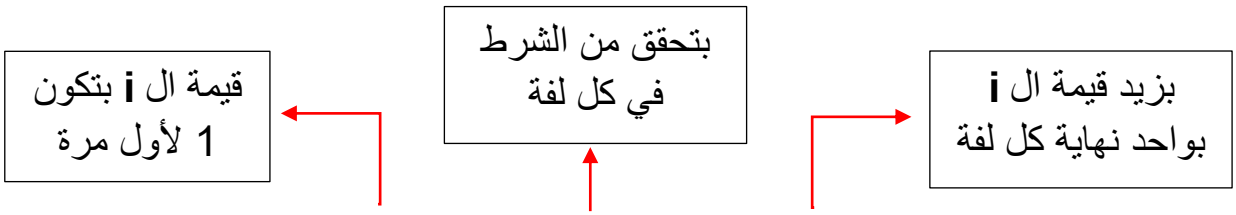



- **اذا كان جواب ال condition :**

true: بنفذ ما بداخل ال **scope** بعدين بنفذ الـ action

false: ما رح ينفذ الي بداخل ال **scope** ولا الـ action

## Example: print 1-3 using for loop




```
for (int i = 1; i <= 3; i++)
{

    cout << i << endl;
}
```

| Iteration | Variable | i<=3 | Action |
|---|---|---|---|
| 1st | i = 1 | true | بطبع 1 وبتزيد قيمة ال i الى 2 |
| 2nd | i = 2 | true | بطبع 2 وبتزيد قيمة ال i الى 3 |
| 3rd | i = 3 | true | بطبع 3 وبتزيد قيمة ال i الى 4 |
| 4th | i = 4 | false | بطلع من ال loop |

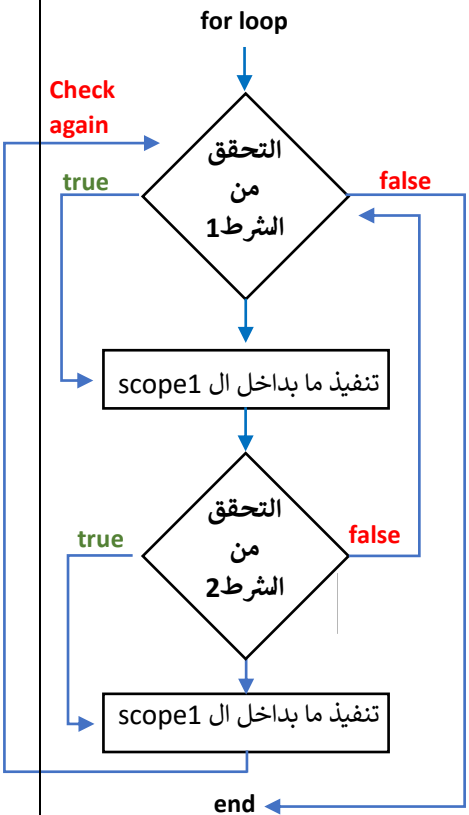
## - Nested loops:

هي ان يتم انشاء **loop** بداخل body ل **loop** أخرى ، يعني **loop** جوا **loop**

ومن الممكن يكون عنا (while بداخل while) او (for بداخل for ).

```
for (var1; condition1; action1)
{
    // body of the outer loop
    for (var2; condition2; action2)
    {
        // body of the inner loop
    }
    // body of the outer loop
}
// other statements
```

check — true — false




**- آلية عمل ال nested for loops :**

**بتم التحقق من ال condition1 :**

**true**: بتم تنفيذ ال **body1** بعد هيك بفوت بال **loop2**

**بتم التحقق من ال condition2 :**

**true**: بتم تنفيذ ال **body2**

**false**: بطلع من ال **loop2** وبرجع يتأكد من **condtion1**

**false**: بطلع من ال **loop1**

Example: A program that uses nested for loops to print a sequence of (1 2 3 4 5), 3 times.

```
for (int i = 1; i <= 3; i++)
{
    for (int j = 1; j <= 5; j++)
    {
        cout << j << " ";
    }
}
```

# When:

في كل مرة بنفوت **loop2**
برجع ال **var2** لل **initial**

**i=1, j=1 to 5**

```
for (1 <= 3)   true
{
  for (j <= 5)   true 5 times
  {
    cout << j <<" ";
  }
}
```

```
1 2 3 4 5
```

**i=2, j=1 to 5**

```
for (2 <= 3)   true
{
  for (j <= 5)   true 5 times
  {
    cout << j <<" ";
  }
}
```

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

**i=3, j=1 to 5**

```
for (3 <= 3)   true
{
  for (j <= 5)   true 5 times
  {
    cout << j <<" ";
  }
}
```

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

**i=4**

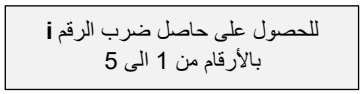
```
for (4 <= 3)   false
{
  for (j <= 5)
  {
    cout << j <<" ";
  }
}
```

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

Example: write a program that print a multiplication table 5x5.

المطلوب نعمل جدول الضرب للأرقام من 1-5 مضروبة من 1-5 ، ورح نحتاج الnested loop ال inner loop حتى تجيب جدول الضرب للرقم ، والouter loop حتى تعيد العملية لجميع الأرقام .

inner loop:

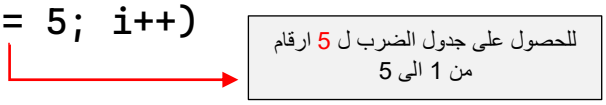



```
for (int j = 1; j <= 5; j++)
{
  cout << i << " x " << j << " = " << i * j << "\t";
}
```

من خلال هاي ال **loop** رح نجيب جدول الضرب للرقم **i**

Outer loop:

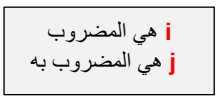
```
for (int i = 1; i <= 5; i++)
{
    //inner loop
}
```

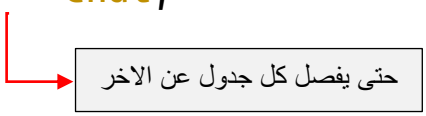



من خلال هاي ال **loop** رح نكرر ال **inner loop** ، ورح نطبع جدول الضرب لأكثر من رقم

## Answer is :

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



# break and continue

**break**: استخدمنا جملة ال break من قبل في ال switch وكان الهدف هو **الخروج** وعدم اكمال ال switch ، وكذلك الامر رح نستخدمها في ال loop والهدف هو **الخروج** من ال loop وعدم اكمال جميع اللفات

**Example:**

```
for (int i = 0; i < 10; i++)
{
    if (i == 4)
    {
        break;
    }
    cout << i << " ";
}
//out of the loop
```

output: 0 1 2 3

لو نلاحظ في هذا المثال لما كانت ال(**i = 4**)فات على جملة ال if وكان فيها **break** وتسببت بالخروج نهائياً من ال **loop** حتى لو ال **condition** لل loop كان **true**

- دائما ال break بتطلع من ال loop ، وما بتنفذ أي اشي تحتها .

continue: جملة ال continue بتعمل break للفة **واحدة** فقط ، يعني اذا صار عندي continue بوقف اللفة الي هو فيها **وبكمل** اللفات الباقية بمعنى اخر بعمل skip لِما تبقى من الكود في **الفة** الي صار فيها continue .

**Example:**

```
for (int i = 0; i < 10; i++)
{
    if (i == 4)
    {
        continue;
    }
    cout << i << " ";
}
//out of the loop
```

output: 0 1 2 3 5 6 7 8 9

لو نلاحظ في هذا المثال لما كانت ال(i = **4**)فات على جملة ال if وكان فيها continue وتسببت بالخروج من ال **اللفة الي كان فيها** ورجع كمل باقي اللفات بشكل طبيعي

Example: write a program that calculate the sum of 5 integer numbers , and print the result

note: if the number is negative don't include it in the sum

if the sum is greater than 100 then stop

```
int num, sum = 0;
for (int i = 0; i < 5; i++)
{
    cout << "Enter number: ";
    cin >> num;
    if (num < 0)
        continue;

    sum += num;
    if (sum >= 100)
        break;
}
cout << " sum = " <<sum;
```

input and output:

Enter number: 1
Enter number: 2
Enter number: 3
continue ← Enter number: -4
Enter number: 5
sum = 11

input and output:

continue ← Enter number: -2
Enter number: 1
continue ← Enter number: -3
break ← Enter number: 120
sum>100
sum = 121

# Notes:

1- في ال for loop بامكانك تعرف اكثر من متغير في خانة ال initial-action ،
تستخدمهم خلال ال loop :

مهم الفصل بـ ,
بين كل **var** والاخر

```
for (int i = 0, j = 0; i + j < 10; j++, i++)
{
    //loop body
}
```

2- بإمكانك إضافة أجزاء أخرى من الأوامر في خانة ال action-after-each-iteration

```
for (int i = 0; i < 5; i++, cout << i << endl)
{}
```

Empty loop body

مهم الفصل بـ ,
بين كل امر والاخر

3- ال infinity loop : هي loop لا يوجد شرط ينهيها وانواعها :

A - انك انت تنشئها حتى تعمل شرط يوقفها بداخله break ، وتشبه فكرة ال control loop ، لكن الي بوقف ال loop هي ال break

```
for ( ; ; )
{}
```

```
while (true)
{}
```

Example: write a program that takes an integer input and calculate the sum of inputs until if the user enters negative input

```
int num, sum = 0;
while (true) {
    cout << "Enter number: ";
    cin >> num;
    if (num < 0)
    {
        break;
    }
    sum += num;
}
cout << "sum = " << sum << endl;
```

input and output:

Enter number: 10

Enter number: 24

Enter number: 22

Enter number: -1 ⟶ break

sum = 58

B – انها تُنشئ عن طريق الخطأ ، وانه شرط التوقف لل loop ما يتحقق ، وهاي الحالة بتسبب error يسمى بال logical error

## Examples:

```
int i = 1;
while (i != 0)
{
    cout << "i = " << endl;
    i++;
}
```

ال **i** مستحيل توصل لل 0 وتتوقف ال **loop**

```
int num, sum = 0;
while (true)
{
    cout << "Enter number: ";
    cin >> num;
    sum += num;
}
cout << "sum = " << sum;
```

مافي شرط يوقف ال **loop** او **break**

الحالات السابقة بتسبب logical error

## 4- كيف تقرأ file كامل باستخدام الـ for loop :

- اذا كان عنا مجوعة ارقام او مجموعة جمل بداخل الملف وبدنا نقرأهم كلهم ، رح نستخدم الـ function اسمه eof (end of file) وظيفته :

يرجع قيمة true اذا وصل لنهاية الـ file و false اذا ما وصل لنهايته

```
ifstream input("fileName.txt");

while (!input.eof()) // Read data to the end of file
{
  input >> variable; // Read data
}
```

رح اتضل الـ while loop تلف لحد ما يقرأ كل الـ data

Example: if we have a file (f1) that contain (1 2 3 4 5 6 7 8 9) read all the data and display it in the console

```
ifstream input("f1.txt");
int x;
while (!input.eof())
{
    input >> x;
    cout << x << " " << endl;
}
```

رح اتضلها تلف لحد ما توصل لرقم 0

Output: 1 2 3 4 5 6 7 8 9

## - Problems with answers :

- Write the code for the following questions:

1) write a program that display the first 10 natural numbers:

Output :

1 2 3 4 5 6 7 8 9 10

2) write a program that takes the range [x , y] and print:

all numbers , all even number , all odd number, summation, average between x and y

sample input1:

Enter two numbers: 0 9

sample output1:

0 1 2 3 4 5 6 7 8 9

0 2 4 6 8

1 3 5 7 9

Sum = 9

Avg = 4.5

Sample input2:

Enter two numbers: -20 -15

sample output2:

-20 -19 -18 -17 -16 -15

-20 -18 -16

-19 -17 -15

Sum = -35

Avg = -17.5

3) write a program that read n numbers from user and find their sum and average :

| Sample test input : | Output: |
| --- | --- |
| Enter n : 3 | Sum = 6 |
| N1: 1 N2: 2 N3 : 3 | Avg = 2 |

4) write a program that display the multiplication table for given number and range :

| Sample test input : | Output : |
| --- | --- |
| Enter your number : 5 | 5 X 1 = 5 |
| Enter your range : 10 | …. …. …. . |
| | 5 X 10 = 50 |

5) write a program that calculate the factorial of a given number:

| Sample test input : | Output : |
| --- | --- |
| Enter your number : 5 | The factorial of 5 = 120 |

6) write a program to check whether a given number is a 'Perfect' number or not:

note: perfect number is divisor of n and the sum of divisors is equal n

divisor of 28 : 1, 2, 4, 7 ,14  sum of divisors : 28 then 28 is perfect

| sample test input : | output : |
| --- | --- |
| Enter your number : 56 | divisors is : 1 2 4 7 8 14 28 |
| | Sum of divisors = 64 |
| | 56 is not perfect |

7) write a program to check whether a number is Armstrong or not :

Example : 153 --> $1^3 + 5^3 + 3^3 = 153$ , then 153 is Armstrong

10 --> $1^3 + 0^3 = 1$, then 10 is not Armstrong

| Sample test input : | Output : |
|---|---|
| Enter number : 153 | 153 is Armstrong number |

8) Write a program to find the prime numbers within a range of numbers.

Note : A prime number is a number that is only **divisible** by itself & 1

| Sample test input : | Output : |
|---|---|
| input starting range : 1 | the prime number between [1,10]: |
| Input ending of range : 10 | 1 2 3 5 7 |

9) write a program to display the number in reverse order :

| Sample test input : | Output : |
|---|---|
| input a number : 12345 | the number in reverse : 54321 |

10) write a program to display a string in reverse order :

| Sample test input : | Output : |
|---|---|
| input a string : Welcome | String in reverse : emocleW |

11) write program to check whether a number is a palindrome or not

| Sample test input1 : | Output1 : |
|---|---|
| input a number : 121 | 121 is a palindrome number |
| Sample test input2 : | Output2 : |
| input a number : 432 | 432 is not a palindrome number |

12) write a program that count the letters ,spaces, numbers and other characters of an input string :

Sample test input :
Enter a string : This is askMeAboutEng22 !

Output:

The number of alphabets : 19

The number of digits : 2

The number of spaces : 3

The number of characters : 1

13) write a program that print all ASCII character with their values :

Sample test input :
input the starting value : 65

Input the ending value : 75

Output :
65 --> A

….. …. ..

75 --> K

14) write a program the find the frequency of each digit :

Sample test input :
Input any number: 1242443

Output :
the freq of 1 : 1

The freq of 2: 2

The freq of 3: 1

The freq of 4 : 3

15) write a program that ask the user to enter number and find the sum of these numbers, if user enter a negative number then skip if user enter character then stop

Sample test input :

Enter number : 1

Enter number : 2

Enter number : -3

Enter number : a

output :

the sum is : 3

16) Write a program to print following :

i)
```
******
******
******
```

ii)
```
*
**
***
****
```

iii)
```
   *
  **
 ***
****
```

iv)
```
    1
  2 2
 3 3 3
4 4 4
4
  5 5 5 5 5
```

v)
```
  *
 ***
*****
 ***
  *
```

vi)
```
5 5 5 5 5
 4 4 4 4
  3 3 3
   2 2
    1
```