Control Structures II *(Repetition)"*

بسم الله الذي لا يضر مع اسمه شيء نبدأ

تحدثنا في الدرس السابق عن ان عملية تنفيذ البرنامج تبدأ بصورة متوالية وفي حال اردنا التغيير على هذه العملية يتم استخدام جمل التحكم وتم تقسيمها الى قسمين .. جمل الاختيار (if-switch) والقسم الثاني الذي سيتم دراسته في هذا الشابتر وهو جمل الدوران *(Repetition)* و يستخدم في حالة الجمل التي تحتاج الى تكرار ..

الجملة الاولى :-

while Looping (Repetition) Structure

الصورة العامة :-

```
while (expression)
statement
```

statement يطلق عليها جسم الدوران وهي الجمل والاوامر التي ستتنفذ طالما ناتج ال exp صحيح (true) ويمكن لهذه الجملة ان تكون مفردة او مركبة . في حال كانت مفردة تكتب بالصورة السابقة لكن في حال كانت مركبة يتم وضعها بعني قوسين راقصين {} .

**يجب ان تحتوي هذه الجملة على تغيير وتعديل على قيمة الشرط لتجنب الدخول في دوران لا نهائي لان ناتج العملية سيبقى صحيح الى المالانهاية .

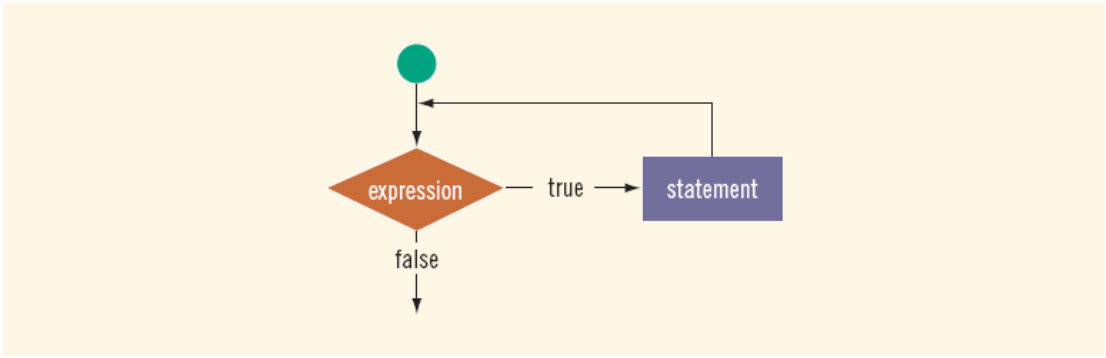

FIGURE 5-1 while loop

**اكيد اكيد هسا بتحكو بدون لف ودوران مش عارفين نفهم !! كيف مع لف ودوران !! والله لتدوخوا !!

EXAMPLE 5-1

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
```

في المثال السابق :- تم اعطاء المتغير i قيمة ابتدائية وتساوي 0 ومن ثم تم الوصول الى جملة الدوران while وتم السؤال i اقل او تساوي 20 ناتج هذا التعبير = true لذا تم الدخول الى جمل الدوران وطباعة قيمة i وتساوي 0 .ثم انتقلنا الى الجملة التالية وهو التغيير والتحديث على قيمة i لتجنب الدخول في دوران لا نهائي حيث تم اضافة 5 لقيمتها لتصبح في الذاكرة i=5

ومن ثم العودة الى الدوران والتحقق من الشرط 5 اقل او تساوي 20 الناتج true لذا سيتم الدخول الى جسم الدوران مرة اخرى وطباعة قيمة i وتساوي 5 ومن ثم التحديث على قيمتها وتبقى العملية متكررة الى ان يصبح ناتج التعبير false .

EXAMPLE 5-2

Consider the following C++ program segment:

```
i = 20;                         //Line 1
while (i < 20)                  //Line 2
{
    cout << i << " ";           //Line 3
    i = i + 5;                  //Line 4
}
cout << endl;                   //Line 5
```

It is easy to overlook the difference between this example and Example 5-1. In this example, in Line 1, `i` is set to 20. Because `i` is 20, the expression `i < 20` in the **`while`** statement (Line 2) evaluates to **`false`**. Because initially the loop entry condition, `i < 20`, is **`false`**, the body of the **`while`** loop never executes. Hence, no values are output and the value of `i` remains 20.

في هذا المثال تم اعطاء i قيمة ابتدائية مقدارها 20 ومن ثم تم الوصول الى جملة الدوران لنجد ان ناتج التعبير flase ..هل 20 اقل من 20. =false لذا سيتم تجاهل جسم الدوران والانتقال الى خارج هذا الجسم وتنفيذ جملة cout<<endl; فقط ..وتبقى قيمة i =20 .

يوجد ثلاثة انواع لجملة الدوران while:-

١- Counter-Controlled ٢- Sentinel-Controlled ٣- Flag-Controlled

والان مع استخدام كل منها ..

Counter-Controlled

تستخدم في حال كان عدد مرات الدوران معلوم حيث يتم تعريف متغير واعطائه قيمة ابتدائية يبدأ عدد الدوران من عنده وينتهي عند قيمة تحدد بالشرط داخل جملة الدوران .

مثال :-

```
counter = 0;          //initialize the loop control variable

while (counter < N) //test the loop control variable
{
     .
     .
     .
     counter++;       //update the loop control variable
     .
     .
     .
}
```

Sentinel-Controlled

تستخدم في حال الرغبة بالدخول للدوران حتى ادخال قيمة معينة يتم التوقف عندها

```
cin >> variable;              //initialize the loop control variable

while (variable != sentinel)  //test the loop control variable
{
     .
     .
     .
     cin >> variable;       //update the loop control variable
     .
     .
     .
}
```

اهمية جملة cin قبل اللوب .. لاعطاء قيمة ابتدائية للمتغير حتى يتم الدخول من خلالها الى شرط اللوب وفي حال لم يتم اعطائه قيمة تظهر رسالة خطأ اثناء تفيذ البرنامج وليس اثناء عملية ال debug .

Flag-Controlled

يتم استخدام متغير من نوع bool للتحكم في عدد مرات الدوران لهذا النوع ..وفي حال اصبح هذا المتغير false يتم الخروج من الدوران .

```
found = false;        //initialize the loop control variable

while (!found)        //test the loop control variable
{
    .
    .
    .
    if (expression)
        found = true; //update the loop control variable
    .
    .
    .
}
```

جملة الدوران الثانية :-

do…while Looping (Repetition)

هذه الجملة هي الاخت التوأم لجملة الدوران الاولى while ولكن الفرق الوحيد والرئيسي انه بهذه الجملة سيتم الدخول الى جسم الدوران body loop مرة واحدة على الاقل ومن ثم فحص الشرط ..

الصورة العامة :-

```
do
    statement
while (expression);
```

statement يطلق عليها جسم الدوران وهي الجمل والاوامر التي ستتنفذ طالما ناتج ال exp صحيح (true) ويمكن لهذه الجملة ان تكون مفردة او مركبة . في حال كانت مفردة تكتب بالصورة السابقة لكن في حال كانت مركبة يتم وضعها بعني قوسين راقصين {} .

آلية العمل :-

١- يتم تنفيذ جسم الدوران بالبداية ومن ثم النظر الى الشرط في حال كان صحيح يتم تنفيذها مرة واخرى وهكذا حتى يصبح ناتج الـ exp =false .
٢- لتجنب الدخول في دوران لا نهائي يجب ان يحتوي جسم الدوران على تعديل على المتغير الموجود بالشرط بحيث يصبح ناتج الشرط =false .
٣- يمكن لجسم الدوران ان يكون عبارة عن جملة واحدة او اكثر اذا كان عبارة عن اكثر من جملة واحدة يجب وضعو بين قوسين راقصين .

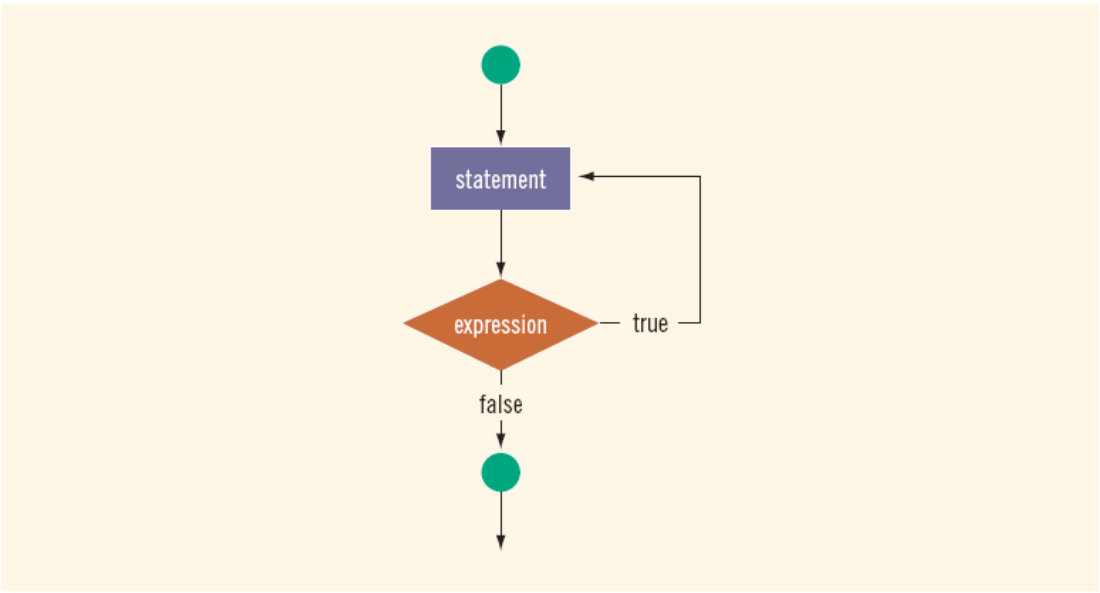

FIGURE 5-3 do...while loop

EXAMPLE 5-15

```
i = 0;

do
{
    cout << i << " ";
    i = i + 5;
}
while (i <= 20);
```

The output of this code is:

```
0 5 10 15 20
```

١- بالبداية يتم اعطاء المتغير i قيمة وهي صفر كما في المثال .

٢- يتم الدخول الى جملة الدوران ..وطباعة قيمة المتغير i وهي 0

٣- يتم التعديل على قيمة المتغير بزيادة 5 لتصبح قيمته في الذاكرة 5

٤- يتم فحص الشرط من خلال جملة while .5 اقل او تساوي 20 الناتج سيكون true

٥- يتم الدخول مرة اخرى الى اللوب وتنفيذ الجمل الموجودة بداخله من خلال طباعة قيمة المتغير i وزيادة 5 على قيمته لتصبح في الذاكرة قيمته = 10

٦- يتم فحص شرط جملة while وهكذا حتى يصبح ناتج التعبير false يتم الخروج من اللوب .

للتمييز بين الجملتين علينا تتبع الكودين في المثال الاتي :-

EXAMPLE 5-16

Consider the following two loops:

a.
```
i = 11;
while (i <= 10)
{
    cout << i << " ";
    i = i + 5;
}
cout << endl;
```

b.
```
i = 11;
do
{
    cout << i << " ";
    i = i + 5;
}
while (i <= 10);

cout << endl;
```

In (a), the `while` loop produces nothing. In (b), the `do...while` loop outputs the number 11 and also changes the value of i to 16.

في البرنامج الاول سوف يتم فحص الشرط اولاً وسيكون ناتجه = flase لذا سوف يتم تجاهل جسم الدوران .
في البرنامج الثاني سيتم الدخول الى جسم الدوران اولاً ومن ثم فحص الشرط .

جملة الدوران الثالثة :-

for Looping (Repetition)

جملة الدوران المرتبة والاكثر استخدامً وشيوعاً والي رح تعتمدوها لقدام لسهولتها ، ولانو ما بتنسى تعمل update على قيمة العداد الخاص فيها لانو منها وفيها .

الصورة العامة :-

```
for (initial statement; loop condition; update statement)
    statement
```

initial statement:- جملة تعريف العداد واعطائه قيمة ابتدائية .وهي اختيارية اي انه يمكنني تعريف العداد بالخارج واستخدامه في جملة الدوران .
loop condition:- وهو الشرط الخاص بالدوران ويجب ان يكون بين فاصلتين منقوطتين .
update statement :- وهي عملية التعديل على قيمة العداد لتجنب الدخول في دوران لا نهائي .

statement :- يطلق عليها جسم الدوران وهي الجمل والاوامر التي ستتنفذ طالما ناتج ال exp صحيح (true) ويمكن لهذه الجملة ان تكون مفردة او مركبة . في حال كانت مفردة تكتب بالصورة السابقة لكن في حال كانت مركبة يتم وضعها بعني قوسين راقصين {} .

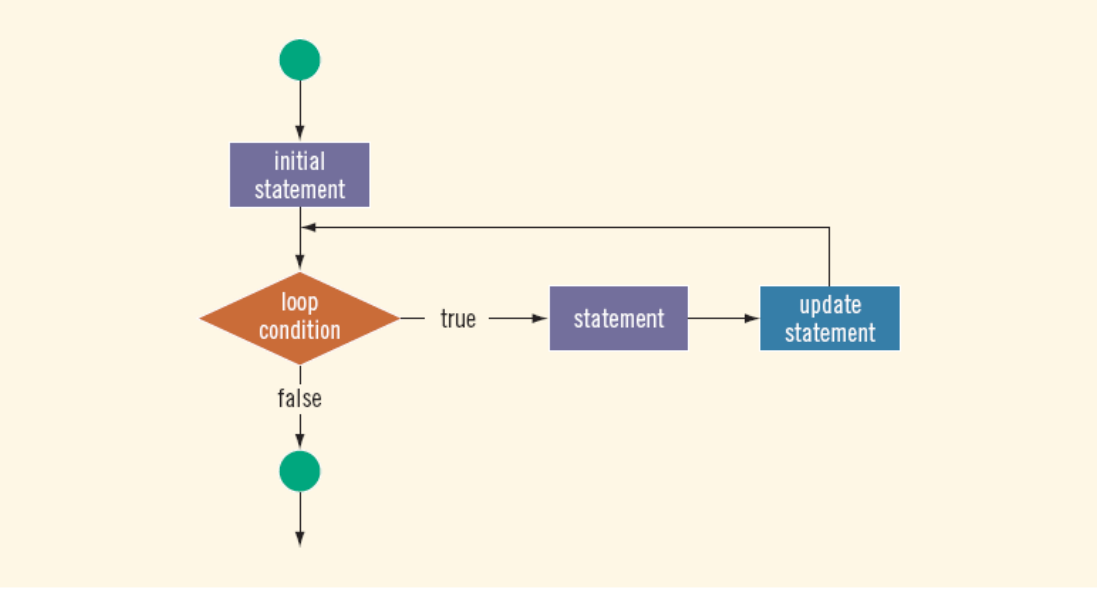

FIGURE 5-2 for loop

١- في بداية تنفيذ جملة الدوران for يتم النظر اول مرة ولمرة واحد فقط الى جملة التعريف initial statement
٢- من ثم يتم الانتقال الى شرط الدوران
٣- من ثم تنفيذ جسم الدوران
٤- من ثم الذهاب الى جملة ال updat .
٥- يتم الرجوع مرة اخرى الى الشرط وهكذا حتى يصبح ناتج التعبير false والخروج من الدوران .

EXAMPLE 5-7

The following for loop prints the first 10 non negative integers:

```
for (i = 0; i < 10; i++)
    cout << i << " ";
cout << endl;
```

EXAMPLE 5-8

1. The following for loop outputs Hello! and a star (on separate lines) five times:

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

2. Consider the following for loop:

```
for (i = 1; i <= 5; i++)
    cout << "Hello!" << endl;
    cout << "*" << endl;
```

في المثال الاول (طبعاً بكون معرفين ;int i قبل) يتم الدخول الى جملة الدوران واعطاء i قيمة ابتدائية وهي 0

بعد ذلك يتم فحص الشرط والدخول الى جسم الدوران وتنفيذه وعند الانتهاء منو يتم العودة مرة اخرى الى جملة update وبعد ذلك فحص الشرط والدخول الى الجسم والانتهاء منو والعودة الى ال update وهكذا حتى يصبح ناتج التعبير false

---

المثال الثاني :-

سيتم اعطاء المتغير i قيمة ومن ثم فحص الشرط ومن ثم الدخول الى جسم الدوران وطباعة hello ومن ثم ترك سطر وطباعة النجمة . والعودة الى جملة الـ update وهكذا حتى يصبح الشرط false

---

المثال الثالث :-

يجب الانتباه انو جسم الدوران موجود بدون قوسين راقصين لذا تكون جمة طباعة hello فقط هي المتكررة والنجمة سيتم طباعتها مرة واحدة عند الانتهاء من جملة الدوران .

EXAMPLE 5-9

The following for loop executes five empty statements:

```
for (i = 0; i < 5; i++);      //Line 1
    cout << "*" << endl;      //Line 2
```

في هذه الحالة يجب الانتباه على مكان الفاصلة المنقوطة ..حيث سيتم الدخول الى الدوران واعطاء المتغير قيمة ابتدائية وهي صفر ومن ثم فحص الشرط ومن ثم الانتقال الى جملة update مباشرة لانو في هذا الحالة يعتبر دوران فارغ ولا يقوم بطباعة اي شي لخلوه من جسم الدوران ..حيث سيتم تعديل قيمة العداد فقط ليصبح الشرط خاطئ.ومن ثم طباعة النجمة مرة

```
for (;;)

cout << "Hello" << endl;
```

هذه الحالة ايضاً تعبتر صحيحة ولكن سيتم الدخول في لوب لا نهائي وطباعة hello بعدد لا نهائي من المرات

يمكن ان تكون جملة update داخل جسم الدوران . والتخلي عنها في عملية تعريف جملة for كما يمكن ايضا اعطاء قيمة للعداد خارج جملة الدوران مثال :-

```
int i=0;

for( ;i<5;)

{cout<<i<<endl;

i++;}
```

break and continue Statements

جملة break تستخدم في حالتين :-

١- في جملة switch
٢- وتستخدم في جمل الدوران .للخروج مبكراً من الدوران .

بالعامية :- لما اشوف بريك بجسم اللوب بوقف عندو وبطلع من اللوب كامل .وبالعادة بحطوها ع شرط if .
**البريك بتطلعني من اللوب

example:-

```
#include <iostream>
using namespace std;
int main ()
{  // Local variable declaration:
   int a = 10;
  // do loop execution
   do   {
       cout << "value of a: " << a << endl;
       a = a + 1;
       if( a > 15)
       {
          // terminate the loop
          break;  }
   }while( a < 20 );
   return 0;}
```

في المثال السابق سيتم طباعة قيمة المتغير الى ان تصبح اكبر من ١٦ سيتم تحقيق شرط البريك والخروج من اللوب !
علماً انو شرط جملة while للرقم ٢٠ .

---

continue

تستخدم بجمل الدوران فقط وتعني تجاهل كل ما تحتها والعودة لبداية اللوب ..

example:-

```
// Local variable declaration:
   int a = 10;

   // do loop execution
   do
   {
       if( a == 15)
       {
          // skip the iteration.
          a = a + 1;
          continue;
       }
       cout << "value of a: " << a << endl;
       a = a + 1;
   }while( a < 20 );

   return 0;
}
```

output will be :-

```
value of a: 10
value of a: 11
value of a: 12
value of a: 13
value of a: 14
value of a: 16
value of a: 17
value of a: 18
value of a: 19
```

nested loop

من اهم المواضيع وهو استخدام جملة دوران داخل جملة اخرى .. يستخدم لرسم الاشكال او لاي شي له بعدين .صف وعمود .

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

١- يتم الدخول الى اللوب الاول (الخارجي) واعطاء المتغير i قيمة ابتدائية وتساوي 1
٢- يتم فحص الشرط ولانه ناتجه =true يتم الدخول الى جسم الدوران الموضوع بين اشارتي {}
٣- لنجد لوب ثاني ف يتم الدخول عليه واعطاء المتغير j قيمة ابتدائية وتساوي 1
٤- يتم فحص الشرط والناتج true فيتم الدخول الى جسم الدوران وهو عبارة عن جملة احادية ليتم طباعة * ثم العودة لجملة التعديل على القيمة لجملة الدوران الداخلية لتصبح قيمة j=٢
٥- فحص الشرط ويكون الناتج false نخرج من جملة الدوران الداخلية ونكمل تنفيذ جملة الدوران الخارجية لنجد جملة cout<<endl; فيتم تنفيذها ثم العودة الى تعديل قيمة اللوب الخارجي لتصبح i=2
٦- التحقق من الشرط وناتجه = true لذا سيتم الدخول مرة اخرى الى جسم الدوران لنجد جملة الدوران الداخلية فنعطي المتغير j القيمة الخاصة به وهي 1 ومن ثم يتم التحقق من الشرط فيكون نتاجه true
٧- ليتم الدخول الى جسم الدوران الداخلي وطباعة الجملة الخاصة به وهي *
٨- يتم العودة الى جملة التعديل على قيمة العداد الخاصة بجملة الدوران الداخلية لتصبح قيمة j=2
٩- التحقق من الشرط ويكون ناتجه true (حيث ٢ اقل او تساوي ٢)
١٠- سيتم الدخول مرة اخرى الى جسم الدورن الداخلي وطباعة *
١١- العودة الى جملة التعديل الخاصة بجملة الدوران الداخلية لتصبح قيمة j=3
١٢- التحقق من الشرط ويكون ناتجه false (حيث ٣ اقل او تساوي ٢)
١٣- ليتم الخروج من جسم الدوران الداخلي واتمام جسم الدوران الخارجي وطباعة cout<<endl;
١٤- والعودة الى جملة التعديل الخاصة بجملة الدوران الخارجية لتصبح قيمة i=3 ومن ثم التحقق من الشرط وتكرار الخطوات السابقة الى ان يصبح شرط الدوران الخارجي =false يتم الخروج من كلا الجملتين

وتصبح المخرجات :-

output will be :-

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

تفكر :-

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

what the output :-

الخلاصة لكل ما سبق :-

1- الحلقة for (for loop).
2- الحلقة while (while loop).
3- الحلقة do.... while (do-while loop).

وفيما يلي سنتناول كل حلقة بالدراسة من حيث الشكل العام و أسلوب الاستخدام وأمثلة توضيحية.

الحلقة for (for loop):

تستخدم الحلقة for لتكرار أمر معين (أو مجموعة من الأوامر) عددا من المرات وتحتاج الحلقة إلي ثلاث عناصر أساسية

```
for ( counter statement; condition; step)
```

و هذه العناصر هي:

1- العداد : (counter) وظيفة العداد هي تسجيل عدد مرات التكرار.
2- الشرط :(condition) والشرط الذي يحدد نهاية التكرار إذ يظل التكرار قائما حتى ينتفي الشرط.
3- الخطوة : (step)وهي القيمة التي تحدد عدد مرات التكرار.

```
{
int counter;
for ( counter=1;counter<=20;counter++)
cout<<counter<<" ";
}
```

ومن البرنامج السابق نجد أن الحلقة for بدأت بكلمة (for) متبوعة بقوسين بينهما ثلاثة عبارات تفصل بينها علامة الفاصلة المنقوطة.
العبارة الأولى تخزن القيمة الابتدائية في العداد.
والعبارة الثانية هي الشرط وهنا الشرط أن قيمة العداد أقل من أو تساوي ٢٠.
أما العبارة الثالثة فهي تحدد الخطوة، وفي هذا البرنامج يزاد العداد بمقدار ١ كل مرة تنفذ فيها الحلقة.
والبرنامج السابق ينتج عنه طباعة الأرقام من ١ إلى ٢٠..

ملاحظات:

1- العبارات الثلاثة المكونة لحلقة for يجب أن تفصل عن بعضها بالفاصلة المنقوطة،

الحلقة(while loop) while

في هذه الحلقة التكرارية نحتاج إلى الشرط فقط وطالما كان هذا الشرط متحققا استمرت الحلقة في التكرار..

```
while ( conditon )
{
statement 1;
statement 2;
.
.
statement n;
}
```

حيث (condition) هو الشرط اللازم لأداء التكرار، والعبارات بداخل أقواس البلوكات هي العبارات المراد تكرارها.
والمثال يوضح استخدام الحلقة while لطباعة الأعداد من ١ إلى ٢٠

```
{
int counter=1;
while ( counter <=20 )
{
cout<<counter<<" ";
counter++;
}
}
```

من المثال السابق يمكننا استخلاص النتائج التالية عن الحلقة:while
-1تخصيص القيمة الابتدائية للعداد تتم خارج الحلقة.while
-2زيادة العداد تتم داخل الحلقةwhile

الحلقة التكرارية:do-while

تختلف هذه الحلقة عن الحلقتين السابقتين في مكان كتابة الشرط ، حيث يكتب الشرط هنا بعد العبارات المطلوب تكرارها.
والشكل التالي يوضح الصورة العامة للحلقةdo-while

```
do
{
statement 1;
statement 2;
.
.
statement n;
}
while ( condition);
```

وأهم ملاحظة على الحلقة التكرارية do-while أنها تنفذ العبارات المطلوب تكرارها مرة واحدة على الأقل حتى ولو كان الشرط غير متحقق!!!
وتفسير ذلك أن التحقق من الشرط يتم بعد التنفيذ وليس قبله كما في الحلقتين السابقتين.

امثلة وتمارين :-

```
#include <iostream>
 using namespace std;
int main ()
 { int n = 10; while (n>0)
 { cout << n << ", ";
 --n; }
 cout << "lift_off!\n";
return 0; }
```

output : 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, lift_off!

```
int main ()
 { int n=10;
while(n>0)
 cout<<n<<", ";
 n--;
 return 0; }
```

output: infinite loop

```
int main ()
 { int n=10;
 while ( n >0);
 { cout << n << ", "; --n; }
 cout << "lift_off!\n";
 return 0; }
```

output:-no thing because semicolon

يعني بضل البرنامج واقف عند شرط ال whileوما بطلع منو

```
for( int a = 10; a < 20; a = a + 1 )
{ cout << "value of a: " << a << endl; }
 return 0; }
```

output :

```
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
```

```
int count(0);
while (count < 10)
{
   if (count == 5)
      continue;
   cout << count << " ";
   ++count;
}
```

output:-

0 1 2 3 4

```
void main()
{
    int evencount = 0, i = 0;
    for(i = 0; ; i++)
    {
        if(i >= 60)
            break; // Terminate the for loop
        if((i % 2) != 0)
            continue; // Will skip the reminder of the for loop
        evencount++;
    }
   cout<<"Total Even Numbers Between 0 - 60 is: "<< evencount;
}
```

برنامج لحساب مجموع الاعداد الزوجية ما بين الصفر وال ٦٠ مهم جداً

output will be ➔

"Total Even Numbers Between 0 – 60 is: 31"

أسأل الله العظيم ان اكون قد وفقت في تغطية كافة افكار هذا الشابتر وما قد اصبت به فمن الله وما اخطأت به فمني ومن الشيطان .

هذا الملخص هو وسيلة لفهم اساس كل جملة وطريقة التعامل معها لتحقيق الفائدة التامة يجب كتابة البرامج بشكل مكثف والمحاولة اكثر من مرة .واي استفسار يمكنك ارساله عبر قروب المادة على الفيسبوك "قطاعات C++" والله ولي التوفيق (محمد المشرقي)

الحمدلله الذي بحمده تتم الصالحات