الفصل 4 : الحلقات

1. المقدمة

الحلقة بالإنجليزيّة loop وبالفرنسيّة boucle: هي بُنية تحكّم، تهدف إلى تنفيذ مجموعة من التعليمات عدّة مرّات **متتالية**، ويتمّ تنفيذ الحلقة حسب الاقتضاء، أمّا بعدد معروف من المرّات مقدمًا (حلقة تكراريّة)، أو إلى أن يسمح الشرط بالخروج من الحلقة (حلقة شرطيّة).

ويوجد ثلاث أنواع من الحلقات:

- حلقة شرطيّة بشرط مسبّق (pre-condition): يتمّ فيها فحص الشرط قبل الحلقة الأولى.
- حلقة شرطيّة بشرط بعديّ (post-condition): يتمّ فيها فحص الشرط بعد الحلقة الأولى.
- حلقة تكرارية (itérative): يتمّ فيها استخدام **عدّاد** (compteur) لحساب عدد التكرارات.

قد يؤدّي خطأ في البرمجة إلى عدم تحقّق شرط الخروج نهائيا. ممّا يجعل البرنامج يعمل إلى الأبد، وتسمّى بالحلقة اللّانهائية (boucle infinie).

2. الحلقة Tant Que

الحلقة Tant Que (ما دام، while) هي حلقة شرطيّة بشرط مسبّق يتمّ فيها تنفيذ مجموعة من التعليمات بشكل متكرّر بناء على شرط منطقيّ. ويمكن اعتبار الحلقة while تكرارا للتعليمة if. وتُستعمل لمّا يكون لدينا مجموعة من التعليمات تتكرّر مع احتمال أنّها لا تنفّذ ولا مرّة (0 أو أكثر)، حسب الشرط المعرّف مسبّقا. وتتكوّن الحلقة من جزئين هما:

- الشرط: وهو عبارة عن تعبير expression من نوع منطقيّ، تكون قيمتها أمّا صحيح vrai أو خاطئ faux
- كتلة تعليمات: يتمّ تنفيذها ما دام الشرط صحيحا.

2.1.الصيغة

الخوارزم	لغة C
TantQue شرط **Faire** كتلة التعليمات **FinTantQue** بقية التعليمات	**while (** شرط **)** **{** كتلة التعليمات **}** بقية التعليمات

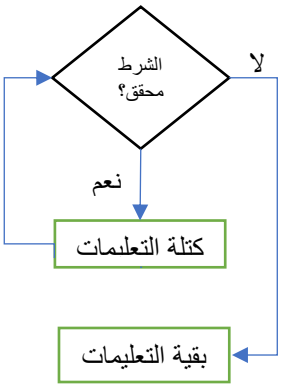
حيث أنّ الكلمات **TantQue** أو **TQ**، **Faire** و**FinTantQue** أو **fTQ** هي كلمات محجوزة في الخوارزم. كذلك الحال بالنسبة لـ while في C. ويكون الشرط دوما في الخوارزم بين كلمتي **TantQue** وfaire، أمّا في C فيكون دوما بين قوسين (). ولبناء الشرط نستعمل عمليّات المقارنة (<، >، =، ≠، ...) والعمليّات المنطقية (et &&، ou ||، non !، ...).

تكون التعليمات التابعة لـ while في C محصورة بين حاضنتين {}، ويمكن حذفهما إذا كانت لا تحتوي إلّا على تعليمة واحدة ({} اختيارية). وإذا وجدنا مجموعة من التعليمات بعد while ولم نجد الحاضنتين، فهذا يعني أنّ التعليمة الأولى فقط هي التي تتكرّر.

ملاحظة

- في C يُعبّر عن النوع المنطقيّ بعدد صحيح int. حيث يُعبّر عن خاطئ faux بـ 0، وعن صحيح vrai بأيّ عدد يختلف عن 0.
- لا توجد ";" بعد }.

2.2. الهيكل التنظيمي algorigramme



2.3. التنفيذ

وتتمّ عمليّة تنفيذ الحلقة الشرطيّة TantQue بحساب عبارة الشرط، والتي تكون نتيجتها من نوع منطقيّ Boolean، فإذا كانت النتيجة صحيحة vrai يتمّ تنفيذ كتلة التعليمات التي بين كلمة Faire وFinTantQue في الخوارزم، أو بين {} في C، ثمّ نقوم بتقييم الاختبار مرّة أخرى ونبدأ من جديد. عندما تصبح نتيجة الاختبار خاطئة، نترك الحلقة بالقفز إلى التعليمات التي تليها مباشرة. عادة ما يطلق على الاختبار بـ "شرط الاستمرار".

ملاحظة

نظرا بأنّ الحلقة TantQue تقوم بفحص الشرط قبل الدورة الأولى، فمن الممكن ألّا يتحقّق الشرط من المرّة الأولى، وبالتالي لا يتمّ تنفيذ تعليماتها ولا مرّة.

2.4. مثال

اكتب البرنامج الذي يقرأ عددين صحيحين، ثمّ يظهر حاصل قسمة الأول على الثاني، دون استعمال عمليّة القسمة (div, /).

ملاحظة القسمة هي عملية طرح متتالية.

الخوارزم	لغة C	الشاشة

algorithme quotient
var x, y, q, r :entier
/*x premier nbr, y deuxième, q quotient, r reste*/
début
écrire("entrer 2 nbrs")
lire(x, y)
q←0
r←x
TQ r>y **Faire**
r←r-y
q←q+1
FinTQ
écrire("le quotient de", x, "sur", y, "est", q, "le reste est", r)
fin
	#include <stdio.h>
	int main()
	{
	int x, y, q, r ;
	printf("entrer 2 nbrs\n") ;
	scanf("%d%d", &x, &y) ;
	q=0 ;
	r=x ;
	while (r>y)
	{
	r-=y ;
	q++ ;
	}
	printf("le quotient de %d sur %d est %d le reste est %d\n", x, y, q, r) ;
	}
		entrer 2 nbrs
		17 5
		le quotient de 17 sur 5 est 3 le reste est 2

الخوارزميّة تأخذ عددين x وy، وتُرجع حاصل القسمة q والباقي r. في البداية نفرض أنّ الباقي هو x، وفي كلّ دورة ننقص منه القاسم y إلى أن يصبح أقلّ من القاسم، وكلما ننقص مرّة نظف 1 إلى حاصل القسمة q. يمكن للحلقة ألّا تنفّذ ولا مرّة، وذلك في حالة x أقلّ من y. فيبقى q=0 وr=x.

مثال2

اكتب البرنامج الذي لا يتوقّف عن التنفيذ إلى أن يضغط المستعمل على المفتاح enter.

```
#include <string.h>
int main()
{
 while (getchar() != '\n');
 return 0;
}
```

3. الحلقة Faire…Tant Que

باستخدام الحلقة **Tant Que**، من الممكن عدم تنفيذ الحلقة على الإطلاق، إذا كان الشرط خاطئًا من التقييم الأول. لكن في بعض الحالات، يريد المبرمج التأكّد من تنفيذ الحلقة مرّة واحدة على الأقل. الحلقة Faire…Tant Que (افعل...ما دام...، do…while…) هي حلقة شرطيّة بشرط بعديّ، يتمّ فيها تنفيذ مجموعة من التعليمات بشكل متكرّر، بناء على شرط منطقيّ. وتُستعمل لمّا يكون لدينا مجموعة من التعليمات تتكرّر، وتنفّذ على الاقل مرّة واحدة مهما يكن الشرط (1 أو أكثر). وتتكوّن الحلقة من جزئين هما:

- كتلة تعليمات: يتمّ تنفيذها ما دام الشرط صحيحا، عدا المرّة الأولى التي تنفذ مهما يكن الشرط.
- الشرط: وهو عبارة عن تعبير expression من نوع منطقيّ، تكون قيمتها إمّا صحيح vrai أو خاطئ faux.

3.1.الصيغة

الخوارزم	لغة C
Faire كتلة التعليمات **TantQue** شرط بقية التعليمات	**do {** كتلة التعليمات **}** **while (** شرط **) ;** بقية التعليمات

حيث أن الكلمات **Faire** و**TantQue** هي كلمات محجوزة في الخوارزم. كذلك الحال بالنسبة لـdo وwhile في C. ويكون الشرط دوما في الخوارزم في الأخير بعد كلمة **TantQue**، أمّا في C فيكون دوما بين قوسين (). ولبناء الشرط نستعمل عمليّات المقارنة (<، >، =، ≠، ...) والعمليّات المنطقيّة (et &&، ou ||، non !، ...).

تكون التعليمات التابعة لـ while في C محصورة بين حاضنتين {} بين do وwhile. ويمكن حذف {}، إذا كانت لا تحتوي إلّا على تعليمة واحدة ({} اختيارية).

ملاحظة

- تنتهي الحلقة do…while دوما بفاصلة منقوطة « ; »

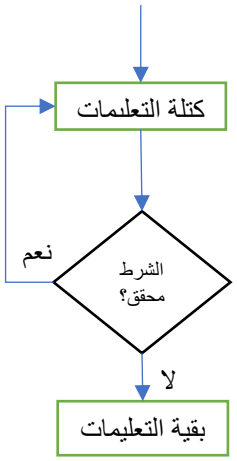
يمكن أن نُعبّر على الحلقة faire…TantQue بالتعبير Répéter…Jusqu'à (أعد... حتّى يصبح الشرط محققا). وفي هذه الحالة يصبح الشرط شرط توقّف، وليس شرط استمرارية. وهو نفي شرط الحلقة TantQue.

مثال

باستعمال faire…TantQue	باستعمال Répéter…Jusqu'à
Faire كتلة التعليمات **TantQue** x>y بقية التعليمات	**Répéter** كتلة التعليمات **Jusqu'à** x≤y بقية التعليمات

ملاحظة نفي < هو ≥.

3.2. لهيكل التنظيمي algorigramme



3.3. التنفيذ

وتتمّ عمليّة تنفيذ الحلقة الشرطيّة Faire…Tant Que بتنفيذ كتلة التعليمات التّي بين كلمة Faire وTantQue في الخوارزم، أو بين do وwhile في C، ثمّ نقوم بحساب عبارة الشرط، والتي تكون نتيجتها من نوع منطقيّ Boolean، فإذا كانت النتيجة صحيحة vrai، يقوم بتنفيذ كتلة التعليمات مرّة أخرى، وهكذا إلى أن تصبح نتيجة الاختبار خاطئة، نترك الحلقة إلى التعليمات التي تليها مباشرة.

ملاحظة

نظرا بأنّ الحلقة Faire...TantQue تقوم بتنفيذ تعليمات الدورة الأولى قبل فحص الشرط، فسيتمّ تنفيذ الحلقة على الأقل دورة واحدة، حتى ولو كان الشرط غير محقق من البداية.

3.4. مثال

اكتب البرنامج الذي يقرأ مجموعة من الأعداد الصحيحة باستعمال متغير واحد، ويتوقف عند أوّل 0 يقرأه، ثمّ يظهر عدد الأعداد التّي أدخلها.

الشاشة	لغة C	الخوارزم

entrer un nbr: 5
entrer un nbr: 7
entrer un nbr: -2
entrer un nbr: 0
Le nombre de nombres est 3
	`#include <stdio.h>`
	`int main()`
	`{`
	`int x, nb ;`
	`nb=0 ;`
	`do {`
	`printf("entrer un nbr") ;`
	`scanf("%d", &x) ;`
	`nb++ ;`
	`}`
	`while ( x!=0 ) ;`
	`printf("Le nombre de nombres est %d", nb-1) ;`
	`}`
		`algorithme lireNbrs`
		`var x, nb :entier`
		`/*x pour lire nbrs, nb pour compter les nbrs*/`
		`début`
		`nb←0`
		`Faire`
		`écrire("entrer un nbr")`
		`lire(x)`
		`nb←nb+1`
		`TQ x≠0`
		`écrire("Le nombre de nombres est", nb-1)`
		`fin`

الخوارزميّة تحتاج لمتغيّر x لقراءة الأعداد، ومتغير nb لحسابها. نضع في nb 0 كقيمة ابتدائية، ثمّ ندخل عددا x ونضيف 1 لـnb، فإذا كان هذا العدد يساوي 0 نتوقف، وإلّا نُعد الكرّة إلى أن يدخل المستعمل العدد 0. سيتمّ تنفيذ الحلقة على الأقل مرّة واحدة. في الأخير نظهر قيمة nb-1 لكيلا يتمّ حساب العدد 0.

4. الحلقة pour

الحلَقة pour (من أجل، for) هي حلقة تكراريّة غير شرطيّة، يتمّ فيها تنفيذ مجموعة من التعليمات بشكل متكرّر عددًا محددًا مسبقا من المرّات. وتتكوّن الحلقة من جزئين هما:

- العدّاد (compteur): لحساب عدد التكرارات، وهو متغيّر variable من نوع صحيح أو رمز. حيث يتمّ تحديد قيمته الابتدائية، وقيمته النهائية، وطريقة زيادته أو إنقاصه.
- كتلة تعليمات: التي سيتمّ تنفيذها في كلّ دورة.

4.1.صيغة pour في الخوارزم

الخوارزم
Pour compteur←بداية **à** نهاية **pas** الخطوة **Faire** كتلة التعليمات **FinPour** بقية التعليمات

حيث أنّ الكلمات **Pour**، **jusqu'à أو à**، **Faire** و**FinPour** أو **FPour** هي كلمات محجوزة في الخوارزم.

- compteur العدّاد: وهو اسم لمتغيّر من نوع عدد صحيح أو رمز.
- بداية : وهي القيمة الابتدائيّة التي يأخذها المتغيّر compteur.
- نهاية : وهي القيمة النهائيّة التي يمكن أن يأخذها المتغيّر compteur.
- الخطوة : وهي القيمة التي يتغيّر بها المتغيّر compteur في نهاية كل دورة. حيث compteur → compteur+خطوة. في العادة تساوي 1.

ملاحظات

- يتمّ حساب قيمة النهاية مرّة واحدة قبل تنفيذ الحلقة.
- الجزء (pas الخطوة) هو اختياري، وفي حالة غيابه، يعني أنّ الخطوة تساوي 1.
- في حالة الخطوة موجبة، يتمّ إضافتها إلى compteur إلى أن يصبح compteur ≤ النهاية. وفي حالة الخطوة سالبة، يتمّ إنقاصها إلى أن يصبح compteur ≥ النهاية.
- compteur =النهاية داخلة في مجال التنفيذ.
- في حالة البداية أكبر من النهاية والخطوة موجبة، أو البداية أقلّ من النهاية والخطوة سالبة، لا يتمّ تنفيذ الحلقة pour.
- لا يمكن تغيير قيمة العدّاد داخل الحلقة.

4.2. صيغة for في لغة C

حلقة for في لغة C هي أكثر عموما من حلقة pour في الخوارزم. وهي أقرب إلى الحلقة الشرطيّة TantQue منها إلى pour.

الحالة العامة	المكافئة للخوارزم
for (initialisation ; test ; itération) **{** كتلة التعليمات **}** بقية التعليمات	**for** (c=بداية ; c<=نهاية ; c++) **{** كتلة التعليمات **}** بقية التعليمات

حيث أن كلمة **for** هي كلمة محجوزة في لغة C.

ويتكوّن السطر الأول لـfor من ثلاث اخزاء داخل قوسين ()، كلّها اختيارية مفصولة بفاصلة منقوطة « ; ».

- initialisation: ينفّذ هذا الجزء مرّة واحدة قبل تنفيذ الحلقة. ويستعمل، في العادة، لإسناد قيمة ابتدائية للعدّاد. مثل: i=0

- test الشرط: وهو عبارة عن تعبير expression من نوع منطقيّ. يجب أن تكون قيمته صحيح vrai لتنفيذ الحلقة. وفي حالة الشرط خاطئ faux، تتمّ مغادرة الحلقة. وينفّذ في بداية كلّ دورة من الحلقة. في العادة، يتمّ فيه اختبار العدّاد. مثل i<10.
- itération التكرار: تنفّذ في نهاية كل دورة. وتستعمل، في العادة، لزيادة أو انقاص العدّاد. مثل i++ أو i--.

وتكون التعليمات التابعة لـ for في C محصورة بين حاضنتين {}. ويمكن حذفهما، إذا كانت لا تحتوي إلاّ على تعليمة واحدة ({} اختيارية). وإذا وجدنا مجموعة من التعليمات بعد for، ولم نجد الحاضنتين، فهذا يعني أنّ التعليمة الأولى فقط هي التي تتكرّر.

ملاحظات

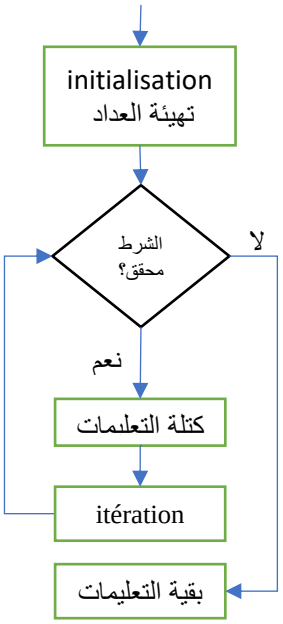
- يمكن التصريح عن المتغيّر (العدّاد) في جزء initialisation، وفي هذه الحالة، فإنّ نطاق تعريفه يكون داخل الحلقة for فقط، ولا وجود له خارجها.
- يمكن زيادة قيمة العدّاد في جزء itération أو إنقاصه أو تعديله بأيّ طريقة أخرى.
- كل أجزاء for (initialisation ، test ، itération) اختيارية، يمكن حذفها وترك مكانها فارغا. لكنّ « ; » إلزاميّة، ولا يمكن حذفها. الكتابة التالية صحيحة `for( ; ; )`
- يمكن أن يحتوي جزء initialisation وجزء itération على عبارات متعددة مفصولة بفواصل « , ».
- تعتبر التعليمة « ; » تعليمة فارغة.

مثال الكتابة التالية متكافئة

```
int i=0;
j=10;
for ( ; ; ){
  if (!(i<j)) break;
  i++;
  j--;
}
```

```
for (int i=0,j=10 ;i<j ; i++,j--);
```

4.3. الهيكل التنظيمي algorigramme



4.4. التنفيذ

تتمّ عمليّة تنفيذ الحلقة التكراريّة pour بإسناد القيمة الابتدائية للعدّاد، وإذا كانت قيمة العدّاد ≤ من القيمة النهائيّة والخطوة موجبة، يتمّ تنفيذ كتلة التعليمات الموجودة بين كلمة faire وfinPour، ثمّ يقوم بإضافة الخطوة إلى العدّاد، ثمّ يعيد الكَرّة من جديد، إلى أن يصبح العدّاد > من النهاية.

وفي حالة الخطوة سالبة وكانت قيمة العدّاد ≤ من القيمة النهائية، يتمّ تنفيذ كتلة التعليمات الموجودة بين كلمة faire وFinPour، ثمّ يقوم بإنقاص الخطوة إلى أن يصبح العدّاد > من النهاية.

أمّا في C، فيتمّ تنفيذ تعبير initialisation مرّة واحدة فقط قبل تنفيذ الحلَقة. ثم ينتقل التحكّم إلى الشرط test. حيث يتمّ اختبار الشرط قبل كلّ تكرار. فإذا كانت النتيجة صحيحة vrai، يتمّ تنفيذ كتلة التعليمات التي بين {} في C، ثمّ يقوم بتنفيذ جزء التكرار itération، ثمّ يقوم بتقييم الاختبار مرّة أخرى ويبدأ من جديد. حيث أنّ جزء التكرار يُنفّذ في نهاية كل دورة. عندما تصبح نتيجة الاختبار خاطئة ، نترك الحلقة بالقفز إلى التعليمات التي تليها مباشرة.

4.5. مثال

اكتب البرنامج الذي يقرأ عددين صحيحين ثم يظهر جميع الأعداد الصحيحة التي بينهما.

الخوارزم	لغة C	الشاشة

algorithme nombres
var x, y, i :entier
/*i est le compteur*/
début
écrire("entrer 2 nbrs")
lire(x, y)
pour i←x **jusqu'à** y **Faire**
écrire(i)
FinPour
fin
	#include <stdio.h>
	int main()
	{
	int x, y, i ;
	printf("entrer 2 nbrs\n") ;
	scanf("%d%d", &x, &y) ;
	for (i=x ;i<=y ;i++)
	printf("%d\t", i) ;
	return 0 ;
	}
		entrer 2 nbrs
		5 9
		5 6 7 8 9

الخوارزميّة تأخذ عددين x وy، وتحتاج إلى متغير i، الذي يلعب دور العدّاد. حيث يأخذ قيما متتالية من المجال x إلى y. وفي نهاية كل دورة يتمّ إضافة 1 إلى العدّاد i. حيث أنّ الخطوة ضمنيّا تساوي 1. وفي حالة x أكبر من y، فلا يتمّ اظهار ولا عدد. أمّا في C فيجب كتابتها. وتمّ الاستغناء عن {} في الحلقة for لأنّها لا تحوي إلاّ على تعليمة واحدة.

5. الحلقات المتداخلة

الحلَقة هي مجموعة من التعليمات التي تتكرّر لعدّة مرّات **متتالية**، ويمكن لهاته التعليمات أن تكون من أيّ نوع وبأيّ عدد. حيث يمكنها أن تكون هي أيضا حلقَة. وفي هذه الحالة نتكلّم عن الحلقات المتداخلة، أي، حلقَة داخل حلقَة. وفي هذه الحالة، يتمّ تنفيذها على النحو التالي:

يتمّ الدخول إلى الحلقَة الخارجيّة.

يتمّ الدخول إلى الحلقَة الداخليّة.

يتمّ تنفيذ تكرارات الحلقَة الداخليّة إلى أن يتمّ الخروج منها.

يتمّ الرجوع إلى الحلقَة الخارجيّة لتنفيذ بقيّة تعليماتها.

وهكذا نعيد تنفيذ الحلقَة الخارجيّة من جديد إلى أن يتمّ الخروج منها.

مثال

اكتب البرنامج الذي يقرأ عدد الاسطر n، ثمّ يظهر على الشاشة في السطر الأول *، وفي الثاني **، وفي الثالث ***، وهكذا إلى أن يظهر في السطر الأخير n *.

الخوارزم	لغة C	الشاشة

algorithme etoile
var n, i, j :entier
/*i, j des compteurs*/
début
écrire("entrer nb de lignes")
lire(n)
pour i←1 **à** n **Faire**
pour j←1 **à** i **Faire**
écrire("*")
FinPour
FinPour
fin
	#include <stdio.h>
	int main()
	{
	int n, i, j ;
	printf("entrer nb de lignes\n") ;
	scanf("%d", &n) ;
	for (i=1 ;i<=n ;i++) {
	for (j=1 ;j<=i ;j++)
	printf("*") ;
	printf("\n") ;
	}
	return 0 ;
	}
		entrer nb de lignes
		5
		*
		**
		***
		****
		*****

for(i) الخارجيّة تحوي تعليمتين for(j) و printf("\n"). أمّا for(j) الداخليّة فتحوي تعليمة واحدة هي printf("*").

printf("\n") تتكرّر n مرّة. أمّا printf("*") فتتكرّر 1+2+...+n مرّة.

6. التكافؤ بين الحلقات

تستخدم الحلَقة Tantque عندما لا نعرف مسبّقا عدد التكرارات، وعندما يكون من الممكن عدم تنفيذ كتلة التعليمات على الإطلاق.

تستخدم الحلَقة Faire…Tantque عندما لا نعرف مسبّقا عدد التكرارات، وعندما يجب تنفيذ كتلة التعليمات على الأقل مرّة واحدة.

تستخدم الحلَقة Pour عندما نعرف مسبّقا عدد التكرارات، أو عندما نعرف بداية مجال العدّاد ونهايته.

كقاعدة عامّة، يمكن التعبير عن أيّ حلقة Tantque بواسطة الحلقة Faire، وذلك بإضافة شرط قبل الحلقة Faire. ويمكن التعبير عن أيّ حلقة Faire بواسطة الحلقة Tantque، وذلك بإضافة كتلة التعليمات قبل الحلقة Tantque. ويمكن التعبير عن أيّ حلقة Pour بواسطة الحلقة Tantque، وذلك بإسناد القيمة الابتدائية للعدّاد قبل الحلقة Tantque، واستعمال القيمة النهائية كشرط للتوقّف، وإضافة التعليمة التي تغير قيمة العدّاد في نهاية الحلقة. ولكن، ليس دوما من الممكن التعبير عن الحلقة Tantque أو الحلقة Faire بواسطة الحلقة pour، إلّا إذا كان هناك عدّاد. أمّا في C فيمكن التعبير عن while أو do…while بواسطة for، كما يمكن التعبير عن جميع الحلقات بواسطة goto وif.

أمثلة

while

while	do…while	for	goto +if

…
r=x ;
while (r>y) {
r-=y ;
q++ ;
}
printf(…) ;
}
	…
	r=x ;
	if (r>y)
	do {
	r-=y ;
	q++ ;
	} **while** (r>y)
	printf(…) ;
	}
		…
		r=x ;
		for (;r>y;) {
		r-=y ;
		q++ ;
		}
		printf(…) ;
		}
			…
			r=x ;
			again :
			if (r>y) {
			r-=y ;
			q++ ;
			goto again ;
			}
			printf(…) ;
			}

do…while

goto +if	for	while	do…while
… r=x ; again : printf("entrer un nbr") ; scanf("%d", &x) ; nb++ ; **if (r>y)** **goto** again ; printf(…) ; }	… nb=0 ; printf("entrer un nbr") ; scanf("%d", &x) ; nb++ ; **for (;x!=0 ;) {** printf("entrer un nbr") ; scanf("%d", &x) ; nb++ ; **}** printf(…) ; }	… nb=0 ; printf("entrer un nbr") ; scanf("%d", &x) ; nb++ ; **while (x!=0) {** printf("entrer un nbr") ; scanf("%d", &x) ; nb++ ; **}** printf(…) ; }	… nb=0 ; **do {** printf("entrer un nbr") ; scanf("%d", &x) ; nb++ ; **} while (x!=0) ;** printf(…) ; }

for

goto +if	do…while	while	for
… **i=x ;** again : **if (i<=y) {** printf("%d\t",i); **i++ ;** **goto** again ; **}** …	… **i=x ;** **if (i<=y)** **do {** printf("%d\t",i); **i++ ;** **} while (i<=y)** …	… **i=x ;** **while (i<=y){** printf("%d\t",i); **i++ ;** **}** …	… **for (i=x;i<=y;i++)** printf("%d\t",i); …

7. أوامر انهاء الحلقات

تُستخدم هذه الأوامر في منتصف الحلقة لتنفيذ خروج مبكّر من الحلقة. عادة عند التحقّق من شرط معين. حيث يمكن انهاء أيّ حلقة for، while، do…while بتنفيذ أيّ تعليمة قفز مثل: break، return أوgoto (إلى تسمية خارج نطاق الحلقة). أمّا التعليمة continue فلا تنهي إلّا الدورة الحالية، لتقفز إلى نهاية الحلقة، لتبدأ الدورة التالية من جديد. وتستعمل هذه التعليمات داخل if. في حالة حلقات متداخلة، تؤدّي هذه الأوامر إلى الخروج من الحلقة التابعة لها فقط.

مثال

for (int i=1 ;i<10 ;i++){
if(i%3==0) continue ;
printf("%d\t", i) ;
}
	for (int i=1 ;i<10 ;i++){
	if(i%3==0) break ;
	printf("%d\t", i) ;
	}

ستظهر جميع الأعداد، وتتخطّى مضاعفات 3 1 2 4 5 7 8	تتوقّف الحلقة عند أوّل مضاعف لـ 3 1 2