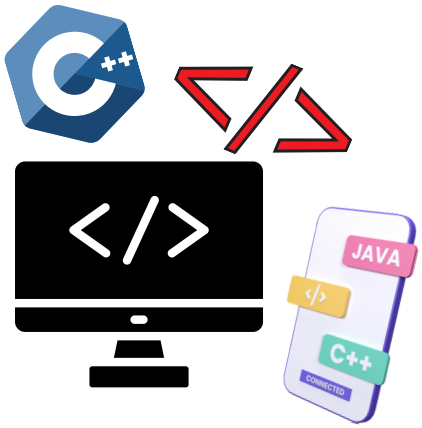
ROOT TEAM

ROOT: WHERE SECURITY BEGINS

روت:حيث يبدأ الأمن

This summary was prepared for educational purposes within the "ROOT TEAM"



ROOT TEAM

ROOT: WHERE SECURITY BEGINS

رووت:حيث يبدأ الأمن

"لغة برمجة (1)"

هذا الملخص وُضع ليساعدك على فهم المادة ومراجعة أهم الأفكار والملاحظات الأساسية فيها، كما يتضمن أهم النقاط التي يكثر السؤال عنها في الامتحانات.

ومع ذلك، فهو لا يكفي أبدًا وحده،
لأن مادة البرمجة لا تُتقن بالمطالعة فقط، بل تحتاج إلى تطبيق فعلي وفهم عملي لكل ما ورد في الدوسية.

لذلك، يجب تطبيق جميع البرامج الموجودة في الدوسية والتأكد من فهمها فهمًا كاملًا، من حيث الفكرة، التنفيذ، والمنطق المستخدم في الحل.

هذا الملخص أداة قوية للمراجعة وترتيب الأفكار ومعرفة أهم مواضع التركيز،
لكن لا يمكن الاعتماد عليه وحده دون التطبيق العملي، لأن ذلك لا يكفي للتمكن من المادة أو حل الأسئلة البرمجية بشكل صحيح.

Subject : C++ programming summary

- Chapter one :-

يوجد فيها أوامر الادخال والاخراج

من هنا يبدأ البرنامج # include <iostream.h>

هي أمر، معناها تضمين مكتبة

الهيكل العام
#include <iostream.h>
main()
{ cout << "Hello" ;
}

main() → رئيسي

begin ← {

أمر الطباعة ← cout << " ... " ;

→ instructions التعليمات تكتب هنا.

end ← }

- chapter two :-

Output statment (cout)

تعرض المخرجات على شاشة الكمبيوتر

① cout << "Hello" ; → is a string (نص) # اي شيء بين (" ") يسمى string

② cout << 27 ; → is a value (قيمة)

③ cout << 2+8 ; → is an expression (تعبير رياضي) نتيجة ناتج عملية رياضية

④ يمكنك دمج العوامل في سطر واحد

ex ↳ cout << " the result " << 5+7 ;

ROOT TEAM
ROOT: WHERE SECURITY BEGINS

Subject : C++ Programming Sumary.

ex :-

① cout << "3+2" ; $\xrightarrow{\text{output}}$ 3+2 (string) (" ")

لانه كان بين (" ")

② cout << 3+2 ; $\xrightarrow{\text{output}}$ 5 (expression) اصبح (" ") # لعدم وجود

يمكننا استخدام << (insertion operator) اكثر من مرة في نفس السطر

① cout << "First " << "program " ; $\xrightarrow{\text{output}}$ First program

② cout << "welcome" << " to " << "jordan." ; $\xrightarrow{\text{output}}$ welcome to jordan.

ملاحظة # حتى يتم طباعة فراغ بين الكلمات يجب وضع مسافة داخل ال (" ")
في اثناء كتابة الكود.

سطر الجديد → "\n" or <<endl .
سطر جديد — نهاية سطر

ex1:

cout << "First " << endl ; $\xrightarrow{\text{output}}$ First
cout << "second" ; second

نفس الشيء

cout << "First " << endl << "second" ;

ex2:

cout << " First sentence. \n" ; $\xrightarrow{\text{output}}$ First sentence.
cout << " second sentence. \n Third sentence. " ; second sentence
Third sentence

ROOT TEAM
ROOT : WHERE SECURITY BEGINS
روووت ... بدأ الأمن

Subject : C++ Programming Summary.

Escape characters :- أحرف الهروب

\n new line :- سطر جديد

\t tap 8 :- تنقل من الى tap الذي يليه وكل تاب (8) احرف وعددهم (15) تاب.

\a Sound :- رسالة تحذيرية

\r return line :- عودة لبداية السطر وكتابة مابعده بدل المكتوب) تقيمنا لأول السطر

\b تحذف آخر حرف

\0 الغاء طباعة مابعدها

\\ → →	لكي تطبع (\)	
\' → →	لكي تطبع (')	
\" → →	لكي تطبع (")	
\? → →	لكي تطبع (?)	

Ex :-

① cout << "123 \r 456"; output → 456

② cout << "123 \b 456"; output → 12456 (تحذف)

③ cout << "A\\\\\B\?\"C\' "; output → A\\\B?"C'

④ cout << "A\\\\B\?\?\"C\' "; output → A\\B??"C'

⑤ cout << "A\t CDE\b\0ABCD"; output → A..... CD

(F) تحذف — لا تهتم بما بعد \0

فراغ بمقدار 8 تاب

ROOT TEAM
ROOT WHERE SECURITY BEGINS
روووت حيث يبدأ الأمن

▸ Subject : C++ Programming Summary.

Escape characters :- أحرف الهروب

\n new line :- سطر جديد

\t tap 8 :- تنقل من الى tap الذي يليه وكل تاب (8) احرف وعددهم (15) تاب.

\a Sound :- رسالة تحذيرية

\r return line :- عودة لبداية السطر وكتابة مابعده يدل المكتوب) تعيدنا لأول السطر

\b تحذف آخر حرف

\0 الغاء طباعة مابعدها

\\	→	→	لكي تطبع (\)
\'	→	→	لكي تطبع (')
\"	→	→	لكي تطبع (")
\?	→	→	لكي تطبع (?)

ex :-

① cout << "123\r456"; output → 456

② cout << "123\b456"; output → 12456 (تحذف)

③ cout << "A\\\\\B\?\"c\'"; output → A\\\B?"c'

④ cout << "A\\\\B\?\?\"c\'"; output → A\\B??"c'

⑤ cout << "A\t CDE\b\0ABCD"; output → A..... CD

(تحذف (E) — \0 لا تهتم لما بعد)

فراغ بمقدار 8 تاب

Subject : C++ Programming Sumary.

قواعد اختيار اسم المتغير

اولاً الشروط :-

① ان لا تكون كلمة محجوزة قبل (main)

② ان لا تبدأ برقم (يجب بحرف) او (شرطة سفلية _)

③ ان لا تحتوي على رموز او فراغات

④ لا يمكن استخدام الأحرف الخاصة قبل ($ & * %)

ملاحظات

- لأن لغة C++ حساسة لحالة الأحرف يجوز كتابة متغير (Main) لان اول حرف كبير
- بعد الحرف الاول يمكن ان تكون المتغيرات من احرف او ارقام او شرطات سفلية.

Data type.

short →	2 byte		أقل من رينج الـ int قليلاً
int →	2 byte	(32,767) الى (-32,768)	من
long →	4 byte		أعلى من رينج الـ int
unsigned int →	4 byte		للاعداد الصحيحة الموجبة
char →	1 byte		تستخدم لتعريف متغير يساوي رمز او حرف او رقم
bool →	1 byte	(F , T)	العمليات المنطقية
float →	4 byte		للأعداد الحقيقية (مع اعشار)
double →	8 byte		ايضاً للاعداد الحقيقية ولكن الرينج اضعف

ملاحظات

① اضافة كلمة const عند تعريف متغير هذا يجعل المتغير يثبت على قيمة واحدة لا يمكن تعديلها واذا غيرنا قيمته او حاولنا قيمة جديدة يعطي Error

② داخل (' ') single لا تستطيع وضع إلا حرفين او رقمين بعد ذلك يعطينا Error ولا يظهر إلا اول رقم اذا كنا قد كتبنا الحرف واحد. عند تعريف المتغير بـ char

Subject : C++ Programming Summary.

Ascll code :- هو نظام عالمي يعطي لكل حرف ، رقم ، او رمز (رقم تعريفي خاص به) يفهمه الكمبيوتر.

A → 65	a → 97	0 → 48	
B → 66	b → 98	1 → 49	space → 32
C → 67	c → 99	2 → 50	
⋮	⋮	⋮	
Z → 90	z → 122	9 → 57	
(الاحرف الكبيرة)	(الاحرف الصغيرة)	(الارقام.)	

ملاحظة :- اذا سجلنا بعد char رقم قبل (97) فسوف يطبع حرف (a) لأن نوع المتغير char اما اذا وضعنا ال (97) بين ('') يطبع فقط اول رقم وهو 9.

ex :-

① char x = 97 ; cout << x ; → output → a

② char x = '97' ; cout << x ; → output → 9

③ char x = "97" ; cout << x ; → output → Error

ملاحظة :- اي شيء بين (" ") فهو (string)

الكلمات المحجوزة

const	double	float	int	short	struch		unsigneed
break	continue	else	for	long	signed	switch	void
case	default	char	do	if	return	static	which
while							

→ لا يمكن استخدامهم كأسم متغير

Subject : C++ Programming Summary. / /

Operations : العمليات

ملاحظة :- العلامة (%) خاصة للارقام الصحيحة وتعطي Error اذا كان نوع البيانات Float

① Mathmatical ⇒ ( ) , * / , + - , %

( ) ↓ الاقواس
* / ↓ ضرب وقسمة
+ - ↓ جمع وطرح
% ↓ باقي القسمة

② Relational : العلائقية ⇒ > , < , == , >= , <= , !=

> ↓ grether than
< ↓ less than
== ↓ equal
>= ↓ grether than or equal
<= ↓ less than or equal
!= ↓ not equal

③ logical : المنطقية ⇒ && || !

&& ↓ and
|| ↓ or
! ↓ not

cin : characterinput هي اختصار الى
وهو الكائن المسؤول عن استقبال
البيانات من المستخدم عبر لوحة المفاتيح

ex :- (47 / 2.0) → Float

A. 23.5 B. 23
C. 24 D. Error

هذه الإجابة لان اشارة القسمة لاتستعمل الا مع
اعداد صحيحة

ex :- (47/2)

A. 23.5 B. 23
C. 24 D. Error

هذه الإجابة لأن صحيح على صحيح لايعطي
الا صحيح.

ملاحظة :- اي int على int يعطي قيمة صحيحة وليس Float

وينطبق هذا ~~الحالة~~ أيضًا على (%) mod.

ROOT TEAM
ROOT : WHERE SECURITY BEGINS

▸ Subject : C++ Programming Summary. / /

Special characters :- الرموز الخاصة

1. Double slash (//) لكتابة ملاحظة (comment) اي نص تكتبه بعد هذا الرمز يتجاهله البرنامج ولا ينفذه ، ويستخدم لشرح الكود .

2. Pound sign (#) يستخدم غالباً في بداية البرنامج لاستدعاء المكتبات مثل
include <iostream.h>

3. Angle brackets (<>) تستخدم لإحاطة اسماء المكتبات التي نريد تضمينها في البرنامج

4. Prentheses (()) تستخدم مع الدوال (Functions) مثل main() او لترتيب العمليات الحسابية

5. Braces ({}) تستخدم للتحديد بداية ونهاية بلوك معين من الكود .

6. Quotation marks (" ") نضع بينها اي نص نريد طباعته كما هو String

7. Semicolon (;) بداية الفصلة في نهاية الجملة ، لانهاء اي امر وبدونها يظهر خطأ

Assignment operators :- معاملات التعيين المختصرة .
فكرتها هي اختصار الكود بدلاً من تكرار اسم المتغير مرتين

ما اللذي نفعله فعليًا ؟	العملية	الرمز
يعطي المتغير قيمة جديدة مباشرة	التعيين العادي	=
يجمع القيمة الجديدة على القيمة القديمة ويخزنها	الجمع ثم التعيين	+=
يطرح القيمة الجديدة من القيمة القديمة ويخزنها	الطرح ثم التعيين	-=
يضرب القيمة الجديدة في القيمة القديمة ويخزنها	الضرب ثم التعيين	*=
يقسم القيمة القديمة على الجديدة ثم يخزن	القسمة ثم التعيين	/=
يأخذ باقي القسمة للقديمة على الجديدة ويخزنها	باقي القسمة ثم التعيين	%=

ROOT TEAM
ROOT : WHERE SECURITY BEGINS
روت : حيث يبدأ الأمن

Subject : C++ Programming Summary.

C++ Comments (التعليقات)

هو نص يكتبه المبرمج داخل الكود لشرح فكرة معينة ويتجاهله الـ (compiler) تماماً عند التشغيل.

① (//) يستخدم لكتابة ملاحظة في سطر واحد فقط

② (/* */) يستخدم اذا كانت الملاحظة طويلة وتريد كتابتها في عدة اسطر.

ex:- $x = 7$ اذا كانت قيمة

$x = x + 3 \rightarrow 7 + 3 = 10$

$x += 3 \rightarrow 7 + 3 = 10$

$x -= 3 \rightarrow 7 - 3 = 4$

$x *= 3 \rightarrow 7 * 3 = 21$

$x /= 3 \rightarrow 7/3 = 2$

$x \% = 3 \rightarrow 7 \% 3 = 1$

Postfix, Prefix increment and decrement.

Increment الزيادة (++)		decrement النقصان (--)	
Pre قبلية	Post بعدية	Pre قبلية	Post بعدية
++x	x++	--x	x--

Post (البعدية) يقصد بها (استخدم اولاً ثم زد) حيث يستخدم قيمة المتغير الحالية كما هي في العملية وبعد ان ينتهي من السطر تماماً يقوم بزيادة المتغير واحد في الذاكرة

Pre (القبلية) يقصد بها (زد اولاً ثم استخدم) حيث يقوم البرنامج بزيادة قيمة المتغير بواحد فوراً، وبعدها يستخدم القيمة الجديدة في العملية الحسابية او الطباعة.

في بعض الحالات لا تفرق اذا استخدمنا Pre او Post مثل :-

ex:- int x = 5;
++x;
cout << x;
output: 6

ex:- int x = 5;
x++;
cout << x;
output: 6

ROOT TEAM
ROOT : WHERE SECURITY BEGINS
روت : حيث يبدأ الأمن

Subject : C++ Programming Summury

الأولويات

ملاحظة :-

لو كانت n=9 وكان الكود

① الأقواس

```
cout << n-1;
cout << n
```

output

8
9

لا تتغير قيمة (n) في الذاكرة

② Pre (++x), (--x)

```
cout << n--1;
cout << n;
```

output

8
8

تغيرت قيمة (n) في الذاكرة لوجود (=)

③ الضرب، القسمة، باقي القسمة

④ الجمع والطرح

⑤ عمليات المقارنة <, >, <=, >=

⑥ عمليات التساوي وعدم التساوي == , !=

⑦ Not

⑧ And

⑨ OR

⑩ عمليات الإسناد -=, +=, %=, /=, *=

⑪ Post (x++), (x--)

(++x) الأقواس لا تعطي للعبارة أولوية فتنفذ بعد انتهاء العملية الحسابية على جميع الجهات.

ملاحظة :-

① (&) تستخدم عندما تريد التحقق من الشرطين حتى وإذ كان الأول خاطئ.

② (&&) تستخدم لتقليل العبء بحيث اذا الشرط الأول خطأ لا تقوم بفحص الشرط الثاني.

ROOT TEAM
ROOT: WHERE SECURITY BEGINS
روت: حيث يبدأ الأمن

Subject : C++ Programming Summary.

Chapter three :-

Selection Statement : الجمل الشرطية

① if (الشرط البسيط)

* تستخدم لتنفيذ كود معين فقط اذا كان الشرط صحيحًا

ex:
```
int degree;
cin>> degree;
if (degree>=0)
{
cout << "Positive";
}
```

② if/else (الاختيار الثنائي)

* تستخدم للاختيار بين طريقتين ، تنفيذ كود اذا كان الشرط صحيحًا وكود آخر مختلف اذا كان الشرط خاطئاً.

ex:
```
int x;
cin>> x;
if ( x / 2 ==0)
cout << "even";
else
cout <<"odd";
```

③ if/elseif (تعدد الشروط)

* تستخدم عندما يكون لديك اكثر من احتمال او شروط متعددة متتالية.

ex:
```
int mark;
cin>> mark;
if (mark>=84) cout << 'A';
elseif (mark >=76) cout << 'B';
elseif (mark >=60) cout <<'D';
else cout <<'F';
```

تكون بدون (if).

ROOT TEAM
ROOT: WHERE SECURITY BEGINS
روت: حيث يبدأ الأمن

Subject : C++ Programming Summary

ملاحظات مهمة :-

① اذا كان في جملتين تابعات لل (if) يجب وضع بلوك (اقواس) .

② اذا وضعنا فاصلة منقوطة بعد جملة (if) تعني ان الشرط انتهى ومابعدها غير تابع لها .

③ (else if) يمكنك تكرارها على حسب عدد الشروط

④ الفرق بين (=) و (==)

(==) ← للمقارنة

(=) ← تستخدم لتحديد قيمة

```
EX:- int x ;
     cout << "Enter the value of x : " ;
     cin >> x ;
     if (x>0)
     {
       cout << " x = " << x << endl ;
       cout << " x is positive " ;
     }  else if (x<0)
     { cout << " x = " << x << endl ;
       cout << " x is negative " ;
     }
     cout << "Hello" ;
     else
     cout << " x is 0 " ;
     }
```

هذه تعد خارج الشرطين لذلك يعطي Error

```
int x=0;
if(x=2);
cout<< "yes";
```

output → yes

```
int x=0;
if(x==2);
cout<< "yes";
```

output → لا يطبع لانها عملية مقارنة

```
EX :- int x ;
      cin>> x ;
      if (x>0);
      cout << "Hello" ;
```

→ Hello

تطبع في كل الحالات لاننا وضعنا بعد الشرط (;)

SIGMA NOTEBOOK

The Switch Statement :-

هي طريقة انتقائية لتنفيذ كود معين بين خيارات كثيرة .

① Switch :- المتغير الذي يراد فحصه (مثلا رقم او نوع العملية)
② case :- القيمة اللي بنقارن فيها (اذا المتغير ١ افعل كذا)
③ Break :- توقف التنفيذ وتطلع برى ال Switch
④ Default :- الخطة البديلة لي بتنفذ في حال ولا اشغل case

الصورة العامة للكود

```
switch (expression)
{
cas1 : (statment) ; break ;
cas2 : (statment) ; break ;
default : (statment) ;
}
```

اذا كان ال cas1 غلط فالبرنامج ينفذ يلي بعدها cas2 , واذا كان ايضا cas2 غلط ينفذ default

يجب وضع ال ({ })

الانتباه الى ال (break) لانها تتحكم في ماذا يطبع وقت تخرج من switch

مثال :-

EX1 :-
```
switch (x=5)
{
cas 5 : cout << "ok" ; break ;
cas 4 : cout << "bye" ; break ;
}
```
the output → ok

ظهرت ok فقط لاننا وضعنا break نهاية ال case.

EX2 :-
```
switch (x=5)
{
cas 5 : cout << "ok" ;
cas 4 : cout << "bye" ; break ;
}
```
the output → okbye

ظهرت هكذا لاننا لم نضع break في cas1

ROOT TEAM
BOOT WHERE SECURITY BEGINS
روووت حيث يبدأ الأمن

Subject : C++ Programming Summury.

ex 3:-

```
switch (2)
{
case 0 : cout << "welcome" ; break ;
case 1 : cout << " ok" ; break ;
case 2 : // cout << " Good bye" ; break
default : cout << " Try again" ;
}
```

the out Put ⟹ Try again

ظهرت بالمخرجات هذه الجملة لان (case 2) مابعدها كانت ملاحظة وليست statment لذلك انتقلنا الى default .

ex 4:-

```
switch (10)
{
case 10 : cout << "Hello" << endl << "C++" ; break ;
case 20 : cout << "ok" ; break ;
defult : cout << " Goodbye" ;
}
```

output ⟹ Hello C++

ex 5:-

```
switch (20)
{
case 10 : cout << " Hello " ; break ;
cout << " Ahmad " ;
case 20 : cout << "ok" ; break ;
defult : cout << " Goodbye " ;
}
```

output ⟹ ok

تمت مجاهلة هذه الجملة لانها اتت بعد فاصلة منقوطة .

ROOT TEAM
ROOT: WHERE SECURITY BEGINS

ال (else) دائماً تكون تابعة لل (if) الأقرب لها . مثل

```
if (x>5)
    if (x < 25)
        cout << x;
    else
        cout << "error";
```

هذه تكون تابعة لأقرب if

اذا كان ال case صحيح لكن لا يوجد بعده break فإنه ينفذ ما بعده ايضاً .

ex:-

```
switch (x=5)
{
    case 4: cout<<"bye";
    case 5:
    case 6: cout<<"Hello"; break;
}
```

⇒ Hello.

ROOT TEAM
ROOT WE BUILD YOUR FUTURE
روت نحن نبني مستقبلك

Subject : C++ Programming Summury

Chapter Four :-

Loops :- الحلقات التكرارية

تعد من اهم الادوات في لغة C++ ووظيفتها ببساطة هي تنفيذ كود معين لعدد من المرات دون الحاجة لتكرار كتابته .

① الحلقة (for) للتكرار المحدود

تستخدمها عندما نكون على علم مسبق بعدد المرات التي نريد فيها تكرار الكود

for (int i = x ; i <= y ; i++)

طريقة عمل الـ (for)

في البداية يذهب الى المتغير بعدها الى الشرط اذا تحقق الشرط ينتقل الى العملية الحسابية وهي زيادة او نقصان .

② الحلقة (while) للتكرار المشروط

تستخدمها عندما لا نعرف عدد المرات بالضبط لكننا نعرف الشرط الذي يجب ان يتوقف عنده التكرار .

while (condition)

* هنا نكتب الكود المراد تنفيذه ؟

{

ملاحظة : يتم فحص الشرط قبل دخول اللوب فاذا كان الشرط خاطئ من البداية لن ينفذ الكود ولا مرة واحدة .

③ الحلقة (do-while) التنفيذ اولاً

تشبه (while) تماماً لكن الفرق الجوهري هو انها تنفذ الكود مرة واحدة قبل ان تفحص الشرط

do {

* الكود المراد تنفيذه

} while (condition);

مبدأ عمل الـ for بشكل عام

بداية عداد * — جملة تنقل من البداية الى النهاية . — نهاية عداد *

ROOT TEAM
ROOT: WHERE SECURITY BEGINS

▸ Subject : C++ Programing Summury. / /

ex:-

```
include <iostream.h>
main()
{
int i=1;
while (i<=10)
{ cout<<i<<endl;
  i++;
}
}
```

the output →

1
2
3
4
5
6
7
8
9
10

في هذا البرنامج تكون قيمة (i) في الذاكرة (11).

ex2:-

```
int x=0
while (x!=0)
{ cout << "Hello";
}
```

output ↓

لا يطبع شيء.

ex3:-

```
int x=0
do {
cout<<"Hello";
} while (x!=0);
```

output ↓

يطبعها مرة واحدة

Hello

ex:-

```
int x, sum=0;
do {
cout<<"Enter Value:";
cin>>x;
sum+=x;
} while (x!=0);
cout << "The total is: " << sum;
```

قيم افتراضية يدخلها المستخدم ويتوقف البرنامج عند ادخال قيمة (0).

output →

1
3
4
5
0
The total is: 13

ROOT TEAM
ROOT WHERE SECURITY BEGINS
روت جذر يبدأ الأمن

Subject : C++ Programming Summury.

في طبعة (4.5) اذا عرفنا المتغير (i) مثلاً داخل ال (For) وبعدها استخدمنا
فيتعامل معه بشكل طبيعي ويعمل البرنامج بينما في طبعة (5.5) وعرفنا (i)
داخل ال (For) فلا يمكننا استخدامه الا اذا عرفناه مرة اخرى.
* لذلك ينصح بتعريف ال (i) خارج ال (For) تجنباً للمشاكل *

اذا عرفنا ال (i) خارج ال (For) وكانت (i=0) فيمكننا وضع قيمة
جديدة لل (i) داخل ال (For) فإنها لا تؤثر الا اذا كانت (i const) ثابت.

في حال كنا نريد طباعة كل مضروب للأرقام من 1-5 نضع ال cout داخل البلوك
بدون ال "!" مثل 8-

```
# include <iostream.h>
main() {
int i, n, fac=1;
cin>>n;
for( i=1; i<=n; i++)
  fac = fac* i;
cout <<n<<"! = "<< fac;
}
```

ROOT TEAM
ROOT: WHERE SECURITY BEGINS

Set with :-

تعتبر الأداة (setw) اختصار لـ (set width) من الأدوات المهمة في لغة C++ لتنسيق المخرجات وهي تستخدم لتحديد الحد الأدنى للعرض (عدد الخانات) الذي سيشغله النص أو الرقم التالي في شاشة المخرجات.

```
# include <iomanip.h>
```

يجب عليك تضمين المكتبة الخاصة بالتنسيق

ex 1:-

```
#include <iostream.h>
# include <iomanip.h>
main()
{
 for (int i=0 ; i<=4 ; i++)
  cout << " Hello " << setw (i) << " C++ \n" ;
}
```

⇒

```
Hello C++
Hello    C++
Hello      C++
Hello        C++
Hello          C++
```

ex 2:-

```
# include <iostream.h>
# include <iomanip.h>
main ()
{
 int x ;
 For( x = 3 ; x >=0 ; x--)
  cout << setw (x) << " C++ \n " ;
}
```

⇒

```
   C++
  C++
 C++
```

ROOT : WHERE SECURITY BEGINS
روت حيث يبدأ الأمن

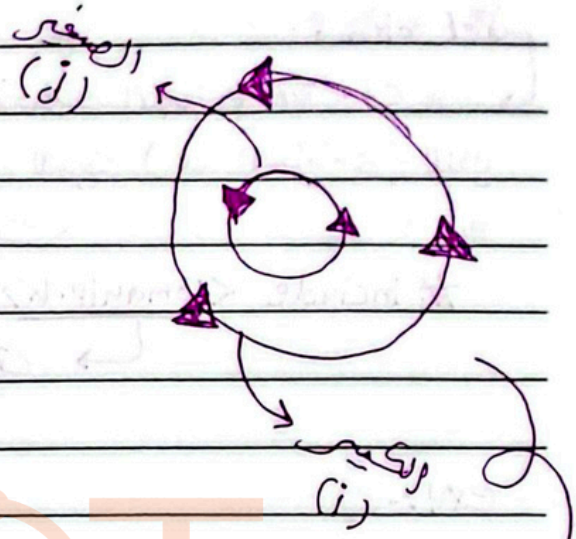
Subject : C++ Programming Summary.

The Nested loop.

تتكون العملية من حلقتين او أكثر

• الحلقة الخارجية : تبدأ اولاً وتتحكم في عدد المرات التي سيعاد فيها تشغيل الحلقة الداخلية بالكامل.

• الحلقة الداخلية : تكمل جميع دوراتها في كل مرة تتحرك فيها الحلقة الخارجية خطوة واحدة فقط.



كل لفة للكبير الصغير يخلص كامل.

Ex:-

```
# include <iostream.h>
main()
{
  for(int i=1 ; i<=3 ; i++)
    {
      for( int j=1 ; j<=5 ; j++)
      cout <<j ;
      cout <<endl;
    }
}
```

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

يرجى حل جميع التمارين الموجودة في سلايد رقم (45).

ROOT TEAM
ROOT : WHERE SECURITY BEGINS

Subject: C++ Programming Summury.

dump Statement (the break and the continue)

1. The break Statment (الخروج النهائي) ex:-

تستخدم لإنهاء الحلقة التكرارية (loop) او جملة الـ (Switch) فوراً .

تقوم بكسر الحلقة والخروج منها تماماً .
وبعدها ينتقل البرنامج لتنفيذ اول سطر بعد نهاية قوس الحلقة .
باختصار : هي بمثابة زر " ايقاف الطوارئ "

```
#include <iostream.h>
main()
{
for(int i=10 ; i>=0 ; i--)
{
if (i==6)
break;
cout<<i;
}
}
```

The output

1 2 3 4 5

2. The continue (تجاوز الدورة الحالية) ex:-

تستخدم لتخطي الجزء المتبقي من الكود في " الدورة الحالية " فقط .

تتجاهل الأوامر التي تأتي بعدها داخل الحلقة وتنتقل مباشرة الى بداية الدورة الثانية
باختصار : هي بمثابة زر " تخطي " (Skip) لهذه المرة فقط مع الاستمرار في التكرار

```
#include <iostream.h>
main()
{
for(int i=10 ; i>=1 ; i--)
{
if(i==4)
continue;
cout<<i;
}
}
```

the output

10 9 8 7 6 5 3 2 1

ROOT TEAM
ROOT WHERE SECURITY BEGINS
روت فريق يبدأ الأمن

Subject : C++ Programming Summary.

Scoping

① Local Scope :- النطاق المحلي

Ex :-

هو المتغير الذي يتم تعريفه داخل اقواس {} او في (loop) مثل

• ينتهي بمجرد الخروج من الاقواس التي احتضنته

• لا يمكن استخدامه خارج هذه الأقواس ابداً .

② Global Scope :- النطاق العام

هو المتغير الذي يتم تعريفه خارج جميع الدوال (وعادة في اعلى الملف)

• يبقى طوال فترة تشغيل البرنامج

• يمكن لأي دالة في الملف الوصول اليه وتعديله .

③ Block Scope :- نطاق الأقواس

اي زوج من الأقواس {} يخلق نطاقاً خاصاً به

• اذا عرفت متغير داخل For loop كالتالي (for(int i=0; ...) فإن i لا تُرى الا داخل هذا ال (loop) فقط .

ROOT TEAM
ROOT : WHERE SECURITY BEGINS
روووت : حيث يبدأ الأمن

▸ Subject : C++ Programming Summary.

ex1:-

```
#include <iostream.h>
int main()
int x=0;
{
  cout << x;
    {
      int x=1;
    }
}
```

the output → 0

ex2:-

```
#include <iostream.h>
int main()
int x=1;
{
  cout << x;
    {
      int x=0;
        { int y=10;
          cout << x;
        }
    }
  cout << y;
}
```

output → 1

output → 1

output → Error

ROOT TEAM
ROOT: WHERE SECURITY BEGINS
روت حيث يبدأ الأمن