كلية العلوم

القسم : **الفيزياء**

السنة : **الثانية**

المادة : لغات البرمجة١

المحاضرة : الثانية /ن+ع/دكتور

{{ مكتبة A to Z }}

Facebook Group : مكتبة A to Z

لغة البرمجة (++C)

2. مدخل إلى لغة البرمجة (++C)

تعتبر لغة (++C) لغة عالية المستوى، وتصنف كلغة برمجية **للأغراض العامة**، وسنذكر فيما يلي أساسيات هذه اللغة:

1.2. الأبجدية (The Alphabet):

هي مجموعة الرموز المستخدمة في بناء عبارات اللغة وأسماء المتحولات والوظائف، وتتألف من مجموعة الحروف اللاتينية الكبيرة (A,B,…,Z) والحروف الصغيرة (a,b,…,z)، ومجموعة الأرقام العشرية (0,1,2,…,9) ومحرف التسطير السفلي (_) (under score)، إضافة إلى مجموعة من الرموز الخاصة التي تملك دلالات مختلفة حسب طريقة تركيبها ومواضع وجودها وهي: (= <> ^ * / - …$ @ & . ' {} [] () ; : , % ! ~)، وتستخدم لغة (++C) العديد من بدائل الخلط بين هذه الرموز مثل : (\t, **\n) السطر الجديد**، المسافة السطرية على التوالي كما سنرى لاحقاً.

2.2. المعرفات والكلمات المحجوزة (Identifiers And Reserved Words):

المعرفات هي كلمات يتم تشكيلها من تتابعات من رموز الأبجدية وتستخدم للدلالة على القيم المستخدمة في البرنامج، أو على وظائف اللغة الأصلية، أو المعرفة من قبل المستخدم، ويخضع تشكيل هذه الكلمات إلى قواعد نذكر منها :

1. يمكن أن تتضمن أياً من الأحرف الكبيرة أو الصغيرة، الأرقام العشرية، محرف التسطير السفلي مثل: …AB1, sum2, x1, Y2, table_name.
2. لا تبدأ برقم.
3. لا تحوي فراغات.
4. يمكن أن تكون بأي طول، قد تصل لـ (31) محرف حسب الرغبة.
5. لا يجوز أن يكون المعرف أياً من الكلمات المحجوزة.

تميز لغة (++C) بين المحارف الكبيرة و الصغيرة أي حساسة لحالة الأحرف في الحالة العامة.

وسنورد فيما يلي بعض الكلمات المحجوزة والتي لا يسمح باستخدامها كمعرفات والمبينة بالجدول(1.2):

الجدول(12.)، أهم الكلمات المحجوزة بلغة (++C)

auto	break	case	char	const	continue
default	do	doubl	else	enum	long
extern	float	for	goto	if	int
register	return	short	sinned	sizeof	static
struct	voil	typedef	union	while	volatile
Switch	unsigned	asm	catch	class	delete
friend	inline	new	operator	private	public
template	this	protected	throw	try	virtual

3.2. أنواع البيانات:

يوجد في لغة (C++) عدة أنواع من البيانات تختلف بعدد البايتات التي تُخصص لها في الذاكرة، وفي النوع الذي تمثله ويختلف هذا العدد من مترجم لآخر.

إن نوع المعطيات (int) ومشتقاته (short int, long int)، يسمح بتخزين معطيات من النوع الصحيح، أما (float, double)، فتسمح بتخزين معطيات من النوع الحقيقي، حيث يسمى المتغير من النوع (double) بالمتغير ذي الدقة المضاعفة، أما (char)، فيسمح بتخزين معطيات محرفية، ويبين الجدول(2.2)، أنواع المعطيات بلغة (C++) وحجم كل منها.

4.2. الثوابت constants:

تطلق تسمية الثوابت على جملة القيم التي تدون مباشرة لتسند قيماً لمتحولات البرنامج وهذه الثوابت يمكن أن تكون صحيحة، حقيقية، محرفية، (string) متعددة المحارف، وتتميز الثوابت الصحيحة والحقيقية بما يلي:

1- **ثوابت صحيحة** : وهي الأرقام العشرية التي لا تملك جزءاً كسرياً، يمكن أن يعرف الثابت الصحيح بدون إشارة (unsigned) بإضافة الحرف(u) صغير أو كبير إلى الطرف اليساري للعدد، كما يُعرف الثابت الصحيح الطويل بدون إشارة (unsigned long)، بإضافة (uL) إلى ذلك الطرف.
2- **ثوابت حقيقية:** وهي الأعداد التي تملك جزءاً كسرياً أو أس عشري والتي تكتب وفق أحد الأشكال التالية :

- **صيغة الفاصلة الثابتة(Fixed Point Format):**

حيث تفصل النقطة العشرية بين الجزء الصحيح والجزء الكسري للعدد، ويمكن أن يسبق العدد بالإشارة الجبرية كما يمكن ألا يحوي العدد على الجزء الصحيح (قيمة الجزء الصحيح تساوي الصفر) مثل : (.456 , 0.00234 - , 45.6- , 100.02, 1.0).

- **صيغة الفاصلة العائمة (Floating Point Format):**

تكون الصيغة العلمية (Scientific Format)، حيث يكتب جزء من العدد على شكل عدد كسري ذي فاصلة ثابتة مضروباً بقوة من قوى العدد عشرة ويفصل المحرف (e) بين القوة والعدد ذي الفاصلة الثابتة بدون وجود فراغات، ويمكن أن يملك كل من العدد الكسري والقوة إشارة جبرية .

أمثلة على ذلك

$$5.2345e+04 = 5.2345 \times 10^{+4} = 52345$$

-6.1234e+11 , +7.00e-05 , - 9.111e-20

كما يمكن كتابة العدد بعدة أشكال كما يلي:

3.15e00, 31.5e-1, 0.315e1, 0.315e+1, 315000000e-8, 0.0000315e+5

3- الثوابت المحرفية أو سلاسل المحارف:

هي رمز واحد بين فاصلتين علويتين('A' ,'x')، حيث أن أياً من الحروف وعلامات الترقيم والفراغات وعلامات الجدولة والرموز المدرجة في جدول (**ASCCI**) (المقياس الأمريكي لترميز وتبادل المعلوماتAmerican Standard Code For Information) Interchange)، محصورة بين علامتي اقتباس مفردتين (' ').

يُعتبر جدول ASCII كمقياس لشفرات الرموز الفردية، حيث يتم حجز (1byte) لكل رمز، ونذكر في (الجدول (3.2))، تسلسل الهروب لبعض الرموز التحكمية والتي تبدأ بـ(\ ) التي تدعى بمحرف الهروب (escape character)، والتي تدل على وجود عمل خاص يجب فعله ثم يليها حرف أو رمز.

جدول(3.2) سلاسل الهروب الشائعة

الفعل	تسلسل الهرب	ASCII
لسماع الجرص Alert(beep)	\a	007
للانتقال إلى الخلف	\b	008
للانتقال جدول أفقي	\t	009
للانتقال جدول رأسي	\v	011
للانتقال إلى سطر جديد	\n	010
تغذية صيغة الورقة	\f	012
عودة العربة إلى بداية السطر الحالي carriage return	\r	013
لطباعة علامة تنصيص مزدوجة	\"	034
لطباعة علامة تنصيص مفردة	\'	039
لطباعة علامة استفهام	\?	063
لطباعة شرطة مائلة للخلف	\ /	092
لطباعة صفر	\0	000

كما يمكن أن يعبر عن تسلسل الهروب كثوابت عددية محرفية: ('\', '\\', '\b', '\t', '\n'")، والرمز (\0) له تمثيل خاص ويمثل نهاية السلسلة كما أن ('\0')لا يكافئ('0')، ويمثل الحرف (A) بقيمة عشرية (065) في (ASCII).

4-ثوابت سلسلة string constants:

توضع الرموز المثالية بين إشارتي تنصيص مزدوجة معبرة عن السلاسل:

("xxx", "1234", "line1\n line2\n line3")

إن الشريط الأخير عبارة عن ثلاث أسطر وتعرض السلسلة على الشكل:

Line1

Line2

Line3

السلسلة التالية: ("\tTo continue, press the \"return\"key\n") ، يوجد في هذه السلسلة (38) رمزاً بما فيها الفراغات الخمس وأربعة رموز خاصة: (\t)، الانتقال إلى بداية الجدولة الأفقية التالية، وعلامتي تنصيص (\")، وسطر جديد (\n)، ويوجد في النهاية رمز نهاية السلسلة غير المرئي (\0)، ويجب التمييز بين الثابت المحرفي('A')، وثابت السلسلة ("A")، وهما غير متكافئين، حيث يملك الثابت المحرفي قيمة عددية مكافئة في (ASCII) وهي (65) ولا يوجد لثابت السلسلة("A") قيمة عددية.

5 - المتغيرات variables:

المتغير يمثل قيمة معطيات ضمن البرنامج، عددية أو محرفية، وتملك المتغيرات قيماً وأنواع مختلفة، ولكن المتغير يملك نوع واحد من المطيات وقيم مختلفة، وتعرف وفق البرنامج التالي :

Int a, b, c; Char d;

يجب نسب القيم الصحيحة العددية للمتحولات الثلاث الأولى ونسب قيمة محرفية للمتغير(d) ضمن البرنامج، ويمكن تغيير القيم العددية والمحرفية ضمن البرنامج مع المحافظة على النوع، وبالتالي فإن المتغير يملك نوع للمعطيات (data type) أي حجم محدد (size) وفق الجدول(3.1) وقيمة هي القيمة المسندة له ومكان تخزين في الذاكرة (reference).

5.3. المصفوفات ARRAYS:

المصفوفة عبارة عن تجميع عدد من القيم من نفس النوع ضمن نفس الاسم، يدعى أي عنصر ضمن المصفوفة بعنصر المصفوفة، وتميز العناصر باختلاف دليل العنصر ضمن المصفوفة، والذي يبدأ من (0) وحتى (n-1)، إذا احتوت المنظومة (n) عنصر، ويمكن أن تتضمن أعداد صحيحة أو حقيقية أو محرفية، كما يمكن أن تكون ببعد واحد أو أكثر.

المصفوفة المحرفية ببعد واحد هي سلسلة محارف تنتهي بـ (\0) وبالتالي لتخزين (n) حرف يلزمنا (n+1) مكان في الذاكرة لتخزين الرمز الأخير (\0) ويوضع الدليل بين قوسين مربعين كما يلي:

A[1], A[2], …, A[n-1]. نحتاج مثلاً لتسع أماكن لتخزين الكلمة LATTAKIA

L	A	T	T	A	K	I	A	\0

ويجب التصريح عن المتغيرات والمصفوفات قبل استخدامها ضمن البرنامج وذلك لتحديد حجمها كي يقوم المطابق بالحجز اللازم.

مثال: int a ,b ,c; float a1, b1, c1; char ax , bx[10];

حيث تم تعريف ثلاث متحولات صحيحة في التعليمة البرمجية الأولى، وثلاث متغيرات حقيقية في التعليمة الثانية، ومتغير محرفي ax وسلسلة محرفية من 10 حروف في التعليمة الثالثة.

يمكن أن تنسب قيم بدائية للمتغيرات من خلال التعريف كما يلي:

Int a=5; له تعريف متحول عددي صحيح وإسناد القيمة العددية (5).

Char b='*'; له تعريف متحول محرفي وإسناد القيمة (*).

Float sum=0; له تعريف متحول عددي حقيقي وإسناد القيمة (0).

يمكن تعريف مصفوفة محرفية كما يلي: (char text[ ]="lattakia")، بوضع قوسين فارغين ومن ثم إشارة النسب تليها سلسلة المحارف بين إشارتي تنصيص، ويمكن تعريف السلسلة السابقة كما يلي: (char text[9]="lattakia")، وهنا يجب حساب الطول بدقة وأخذ نهاية السلسلة بعين الاعتبار، كما يمكن حجز char text[20] حيث يتم تعبئة المواقع الباقية بفراغات.

6.3. التعابير (expressions):

تملك قيمة فردية عددية محرفية أو مصفوفة أو إشارة تابع ويمكن أن تكون مزيج مما سبق، تتصل مع بعضها **بمعاملات** ويمكن أن تكون نتيجة التعبير قيمة واحدة عددية أو منطقية حيث تملك القيمة (false=0, true=1)، وفي الجدول بعض الأمثلة:

الجدول(4.3) أمثلة لبعض التعابير البرمجية.

لجمع القيم a,b	a+b
نسب قيمة y إلى x	x=y
الجمع والنسب في آن واحد	$d=a_1+a_2$
تكون النتيجة true=1 إذا كان الشرط محقق وإلا 0	x<=y
اختبار تساوي xمع y	X==y
زيادة العداد بمقار واحد	I=i+1

وهذا ما يدعى بالمؤثر الأحادي أو المعامل الأحادي.

7.3. العبارات (statements):

يوجد نوعين للعبارات :

- **عبارة بسيطة simple statement** وتنتهي بـ(;) كما في الأمثلة التالية:

```
a=5;
t=x+y;
```

ملاحظة : عند وجود; مفردة على السطر دون أي تعليمة برمجية تسمى بالعبارة الفارغة.

- **عبارة مركبة complex statement:** تتألف من عدة عبارات بسيطة، تكون مُضمنة بأقواس مجموعة { }، ولا تملك ; بعد قوس الإغلاق، وتستخدم عند وجود عدة عبارات تتبع لنفس الحالة وتصبح عندها العبارة المركبة وكأنها عبارة بسيطة كما في المثال التالي: لدينا عبارة مركبة من أربع عبارات بسيطة ويستمر تنفيذ العبارة المركبة طالما أن

c لم تصبح أقل من n ويجب أن تزداد قيمة c ضمن العبارة كي يتوقف تنفيذ العبارة المركبة:

```
While (c<=n)
{
Cout<<("x= ");
Cin>>("%f", &x);
Sum+=x;
++c;}
```

معاملات لغة Operators(++C):

تستخدم المعاملات الأحادية، والحسابية، والعلائقية، ومعاملات التخصيص أو النسب، والمعاملات الشرطية في تكوين العبارات البرمجية، وتؤثر المعاملات على العوامل أو العناصر وقد تتطلب عامل أو أكثر.

8.3. العمليات الحسابية Arithmetic Operators:

العمليات الحسابية المعروفة كالجمع (+)، والطرح (-)، والضرب (*)، والقسمة (/)، وباقي القسمة (%).تستخدم المعاملات الحسابية عاملين وفي حالتي القسمة وباقي القسمة يجب أن يكون العامل الثاني مخالفاً للصفر كما أن عاملي باقي القسمة يجب أن يكونا صحيحين.

ملاحظات:

1- **يمكن أن نقوم بضرب حرفين أو إجراء أي عملية حسابية عليهما**، حيث سيتم التعامل معهما وفق ترميز ASCII لهما. بفرض أن e1, e2 متغيران يمثلان المحارف P, T على التوالي، وترميز ASCII للمحرف P هو 80 وترميز T هو 84 فإن:

$$e1 + e2 \rightarrow 164$$
$$e1 + e2 + 5 \rightarrow 169$$
$$e1 + e2 + '5' \rightarrow 217$$

حيث أن ترميز ASCII لـ '5' هو 53.

2- إن إجراء عملية على عددين صحيحين نحصل على عدد صحيح بالنتيجة:

```
B=3,a=10;
```

$$a + b \rightarrow 13$$
$$a - b \rightarrow 7$$
$$a/b \rightarrow 3$$
$$a\%b \rightarrow 1$$

3- في إجراء باقي القسمة تحدد إشارة الناتج من إشارة أول عامل، بفرض :

$$b = -3, a = 11$$
$$a + b \to 8,\ \ a - b \to 14,\ \ a * b \to -33,\ \ \ \ a/b \to -3,\ \ \ \ a\%b \to 2$$

4- عند إجراء عملية على عاملين محددي الإشارة وبدقة مختلفة يتم العمل بالدقة الأكبر.

5- **يمكن أن يذكر أحد أنواع المعطيات بين قوسين قبل العبارة ليتم تحويلها بشكل** ضمني implicit conversion)، ويدعى بالتحويل القسري إلى هذا النوع فإذا كان كل من i=7, f=8.5 فإن(i+f)%4 غير صحيح، والتعبير((int)(i+f))%4صحيح،وهنا تطبق (int) على أول عامل وليس على كل العوامل ضمن العلاقة وفي العبارة الواحدة فقط.

9.2. أولوية تنفيذ العمليات الحسابية:

يتم تنفيذ العمليات الحسابية وفق الأولويات التالية:

1- الأقواس

2- الضرب والقسمة و باقي القسمة (% , * , /).

3- الجمع والطرح (- , +).

4- And&&.

5- Or||.

6- الإدخال والإخراج >>,<<.

7- النسب أو التخصيص=, +=, -=, *=, /=, %=.

ويجري التنفيذ من اليسار إلى اليمين عند وجود عمليات متساوية الأولوية.

```
2*((i%5)*(4+(j-3)/(k+2)))
7  3   6 5 1 4 2
```

1.9.2. المعاملات الأحادية unary operators:

تتعامل هذه المعاملات مع عامل واحد، وقد تسبق العامل أو تكون خلفه في بعض الحالات. أكثر هذه المعاملات شيوعاً هو المؤثر السالب، ويسبق الأعداد والثوابت العددية والتعابير الحسابية، كما يوجد مؤثر زيادة الداد ++، ومؤثر إنقاص العداد --، وهما يؤثران بالزيادة أو الإنقاص بمقدار واحد، وقد تسبق العامل وقد تليه كما سنرى لاحقاً.

$$m \to 5;\ \ m + + \to 6$$
$$n \to 4;\ \ + + n \to 5$$
$$q \to 6;\ - - q \to 5$$
$$t \to 5;\ \ t - - \to 4$$

وتملك المعاملات الأحادية أولوية تنفيذ على المعاملات الحسابية.

2.9.2. المعاملات العلائقية relational operators:

هي المعاملات التي تمكننا من إجراء المقارنات المنطقية، وتنجز هذه العوامل عمليات مقارنة مختلفة، وهذه العوامل هي :

- (<) عامل أكبر تماماً .
- (>) عامل أصغر تماماً.
- (=>) عامل أكبر أو يساوي .
- (=<) عامل أصغر أو يساوي .
- (==) عامل المساواة .
- (=!) عامل عدم المساواة.

وهذه العوامل كلها تعطي نتائج منطقية (true=1,false=0) كنتيجة لعملية المقارنة التي يتم تنفيذها

3.9.2. العوامل المنطقية logical operators:

تملك لغة C++ التابعين المنطقيينor (||)، و and (&&). فيما يلي جدول الحقيقة عند تطبيق كل من المعاملين على a,b :

1) **المعامل المنطقي (&&) and** : والذي يعطي نتيجة منطقية (true) في الحالة التي يكون فيها كل من معامليه مساوياً للقيمة المنطقية (true) كما هو موضح بجدول الحقيقة التالي:

A	B	A&&B
0	0	0
1	0	0
0	1	0
1	1	1

2) **المعامل المنطقي (||) OR** : والذي يعطي نتيجة منطقية (true) في الحالة التي يكون فيها أحد معامليه على الأقل مساوياً للقيمة المنطقية (true) والجدول التالي يوضح ذلك :

A	B	A || B
0	0	0
1	0	1
0	1	1
1	1	1

ملاحظة: معامل (&&) لها أولوية تنفيذ أكبر من || وتقع بعد المساواة وعدم المساواة، بالإضافة للمعاملين السابقين يوجد معامل النفي المنطقي الأحادي not (!) والذي يقوم بنفي التعبير المنطقي وله أولوية على المعاملات الأحادية.

4.9.2. الفرق بين (&& و & أو || و |):

تمثل الأداتين && و || عملية منطقية بين موقعين في الذاكرة أو بين متغيرين بشكل كامل وليس جزئي أو تعبيرين نتيجتهما True أو False بحيث تكون النتيجة النهائية إما True أو False

يمثل العدد 0 في لغة الـC++ الحالة المنطقية False بينما يمثل أي رقم أخر عن الحالة المنطقية True

مثال:

```
Int a=5;
Int b=7;
(A >0) &&(b>0)
(True)  &&  (True)→ True
```

تمثل الأداتين & و | عملية منطقية بين كل بت من المتغير الأول مع البت الذي يقابله في المتغير الثاني وليس مع المتغير كاملاً بحيث تكون ناتج العملية أي رقم.

مثال:

```
Int a=5;      0111 → 7
Int b=7;      0101 → 5
(A & b)       0101 →5
```

كما يوجد العديد من المعاملات المنطقية الأخرى مثل xor الذي يعطي صفراً إذا اختلف المدخلين وإلا واحد، وأيضاً << ,>> الإزاحة لليمين واليسار على التوالي وتعمل مع الخانة الثنائية كما هو موضح بالجدول4 التالي:

جدول (4.2) المعاملات المنطقية.

Op	asm	Description
&	AND	Logical AND
|	OR	Logical OR
^	XOR	Logical exclusive OR
~	NOT	Complement to one (bit inversion)
<<	SHL	Shift Left
>>	SHR	Shift Right

عوامل التخصيص أو النسب (Assegment)

إن أكثر عوامل التخصيص شيوعاً هو = وله الشكل التالي: identifer = expression;، حيث: identifer هو معرف بصورة عامة أما exression فهو متغير أو تعبير أو كلاهما معاً مثلاً: $a = 5;\ \ d = 0.1;\ \ s = length * width;$، أما المعامل == فيدعى بالمساواة ويختبر فيما إذا كانت القيمة الطرف الأيمن تطابق القيمة في الطرف الايسر تماماً، والنتيجة منطقية، ويمكن أن نصادف تخصيصات متعددة:

$$identifier1 = identifier2 = \cdots = expression;$$

كما تملك لغة C++ خمس مؤثرات للتخصيص هي: $+=, -=, *=, /=, \%=$

نبدأ بـ =+ حيث نعبر عنه كما يلي:

$$expression1 += expression2; \equiv$$
$$expression1 = expression1 + expression2;$$

كذلك الأمر بالنسبة لبقية العوامل ويحوي الجدول5 التالي أمثلة توضيحية:

الجدول (5.3) مؤثرات التخصيص.

التعبير	المكافئ
i+=5	i=i+5
f-=y	f=f-y
J*=(i-3)	J=J*(i-3)
f /=3	f=f/3
i%=(J-2)	i=i%(j-2)

ملاحظة: تملك عوامل التخصيص أدنى أولوية تنفيذ كما ذكرنا سابقاً.

البرنامج:هو عبارة عن مجموعة من التعليمات تكتب من قبل مبرمجين مختصين بإحدى لغات البرمجة وتحتاج إلى تحويل إلى لغة الآلة كي يعالجها الحاسب.

10.2. خطوات كتابة البرنامج (الكود) source code:

بعد تصميم خوارزمية الحل لمسألة ما، والتي يتم فيها تحديد مدخلات ومخرجات البرنامج بشكل عام،يتم تحويلها إلى تعليمات برمجيةبلغة برمجة ماوفق الخطوات التالية:

1- تعريف المتغيرات (المدخلات والمخرجات وغيرها)
2- كتابة كود إدخال مدخلات البرنامج.
3- كتابة تعمليات البرنامج الرئيسية.
4- كتابة كود طباعة النتائج.
5- إنهاء البرنامج.

1.10.2. بنية البرنامج بلغة C++program structure:

1- رأس البرنامجHeader:

يبدأ البرنامج بالتوجيه <> #incloude والذي يتضمن بين قوسيه أحد ملفات الرأس الضروريةلعمل التوابع المستخدمة ضمن البرنامج، فمثلاً الملف iostream.h يستخدم مع البرامج التي تتضمن تعليمات إدخال وإخراج، والتوجيه <#incloude<string.h ضروري عند استخدام التوابع المصممة لمعالجة السلاسل المحرفية.

بعد كتابة التوجيهات الضرورية لعمل البرنامج يمكن تعريف الثوابت والمتحولات العامة، كما يمكن الإعلان عن التوابع المصممة من قبل المستخدم أو تعريفها بالكامل.

2- جسم البرنامج code body:

يبدأ جسم البرنامج الرئيسي بالتابع ( )main، الذي يمكن اعتباره يعيد قيمة صحيحة وله الشكل التالي:

```
main( )
{
Statements;
Return 0;
}
```

وهنا يعيد البرنامج القيمة 0 إلى نظام التشغيل ليخبره أن البرنامج نفذ بنجاح من خلال تعليمة return، ويخبر التابع ( )main المترجم عن بداية البرنامج.

القوسين { } يحصران جسم التابع ( )main وهما ضروريان في أي برنامج.

Statments: هي التعليمات المستخدمة في التابع ()main تتضمن تصاريح المتحولات، الإضافة إلى الدوال المستخدمة في البرنامج، وكل تعليمة يجب أن تنتهي بـ ;

.

3- إدراج التعليقات (Comments):

تتيح لغة البرمجة C++ للمبرمجين إدراج تعليقات ضمن نص البرنامج الأمر الذي يساعد المستخدم على قراءة وفهم البرنامج حيث لا تطلب هذه التعليقات من الحاسب القيام بأي فعل عند تنفيذ البرنامج. فالمترجم يوم بتجاهل التعليقات أثناء عملية توليد البرنامج التنفيذي المقابل للبرنامج المكتوب بلغة الـ C++.

4- يمكن التمييز بين نوعين من أدوات التعليقات:

1- إشارة التعليق المؤلف من سطر واحد والتي تبدأ بالإشارة // وتنتهي بنهاية السطر.

2- إشارة التعليق المؤلف من عدة أسطر والتي تبدأ بالإشارة */ وتنتهي بالإشارة /*.

ملاحظة : يجب التصريح عن كل المتحولات بتحديد اسمائها ونمط البيانات التي سوف تخزن فيها وذلك قبل أن يتم استخدامها في البرنامج.

تمثل الـ VarType نوع المتحول فيما إذا كان عدد (int , float , double) أو محرف (char) أو سلسلة (String) أو قيمة منطقية (bool)، ويمكن التصريح عن عدة متحولات من نفس النمط بواسطة عملية تصريح واحدة أو عدة عمليات تصريح.

يمثل المعامل initial Value ضمن المتحول لقيمة ابتدائية وهو عبارة عن إجراء اختياري أي أنه ليس من الضروري تضمين المتحولات قيم ابتدائية حيث سيقوم المترجم بإعطاء المتحول قيمة افتراضية تتناسب مع النوع الذي تم التصريح به.

دوال الإدخال والإخراج:

1- دوال الإخراج:

وهي الدوال والتوابع والتعليمات التي يتم استخدامها في عملية إخراج البيانات إلى وحدة خرج كالشاشة أو الطابعة.

أبسط أشكال دوال الإخراج هي Cout أو std::cout (في حال لم نستخدم الأمر Using namespace std;) والتي تأتي مع المكتبة المعيارية iostream حيث تقوم بإخراج البيانات إلى منفذ الخرج المعياري وهو الشاشة.

يأخذ الصيغة البرمجية التالية:

```
cout<<Variable;
cout<<var1<<var2<<…<<varn;
cout<<”data”;
```

تستخدم العملية >> للدلالة إلى إرسال البيانات المكتوبة على يمين العملية إلى منفذ الخرج المرتبط بالتعليمة، تستخدم إشارتي الاقتباس للدلالة على البيانات (القيم المحرفية) المراد إظهارها على الشاشة بينما يتم استخدام أسماء المتحولات مباشرة عند الرغبة في إظهار محتوياتها.

يجب التصريح عن المتحولات المستخدمة ضمن الـ cout قبل استخدامها ضمن التعليمة.

يستخدم المحرف \ والذي يسمى بلغة البرمجة بحرف الهروب من أجل الدلالة على أن الحرف الذي يليه هو حرف خاص يجب إظهاره أو تنفيذ أمر مرتبط مع هذا الحرف حيث يسمى التركيب المكون من حرف الهروب \ والمحرف الخاص الذي يليه مباشرة بسلسلة الهروب.

اعتماداً على تأثيرات سلاسل الهروب يمكن إظهار البيانات بأشكال متعددة وذلك باستخدام تعليمة إخراج واحدة أو أكثر.

مثال1: أوجد خرج البرنامج التالي:

```
#include<iostream.h>
//this is the first program
Void main(){
Cout<<"hello.\n";}
```

شرح الكود البرمجي:

تتم معالجة الأسطر التي تبدأ بالرمز # من قبل مرحلة ما قبل الترجمة والتي تقوم بضم محتوى الملف الرأسي والحاوي على العمليات الخاصة بالدخل والخرج إلى جسم البرنامج.

يجب ضم المكتبة المعيارية (iostream.h) إلى أي برنامج يحتوي على عمليات خرج للمعطيات على الشاشة أو دخل لها من لوحة المفاتيح.

السطر الثاني تعليق لا يراه المترجم لأنه سبق بـ (//).

السطر الثالث يعلن عن بدء التابع ()main يليه قوس البداية الذي يحصر تعليماته مع قوس النهاية، السطر الخامس يتضمن التعبير :

```
Cout<<"hello.\n";
```

يعني إرسال الكلمة (hello) بواسطة(cout) إلى مجرى خرج قياسي عادة الشاشة، والرمز(\n) يخبر المترجم بالانتقال إلى سطر جديد بعد إخراج الرسالة.

2- دوال الإدخال:

هي الدوال والتوابع والتعليمات التي يتم استخدامها في عملية إدخال البيانات إلى البرنامج عبر منفذ دخل كلوحة المفاتيح أو الماسح الضوئي...الخ.

أبسط أشكال دوال الإدخال هي Cinيستخدم لإدخال المعطيات من لوحة المفاتيح ليتم معالجتها بعد تخزينها في ذاكرة الحاسب.

يأخذ الصيغة البرمجية التالية:

cin>>var1>>var2; أو cin>>Variable;

تستخدم العملية (<<)، والتي توضع بعد الـ (cin) للدلالة إلى عملية إرسال البيانات المدخلة عن طريق منفذ الدخل المرتبط بالتعليمة، يجب التصريح عن المتحولات المستخدمة ضمن الـ (cin) قبل استخدامها ضمن التعليمة.

تنتهي عملية إدخال البيانات من لوحة المفاتيح بمجرد الضغط على زر الإدخال (Enter).

لا يسمح باستخدام العبارات النصية المكتوبة ضمن علامتي اقتباس ضمن تعليمة الإدخال.

مثال2:

```
#include<iostream.h>
Void main(){
Int m, n;
Cout<<"m";
Cin>>m;
N=++m;
Cout<<"m after n=++m is: m="<<m<<" and n="<<n<<endl;
N=m++;
Cout<<"n after n=m++ is: n="<<n<<" and m="<<m<<endl;}
```

خرج البرنامج:

```
m=4
m after n=++m is: m=5 and n=5
n after n=m++ is : n=5 and m=6
```

شرح الكود البرمجي:

تم استخدام التابع cin لإدخال البيانات من لوحة المفاتيح، والمعامل << يسمى معامل الإدخال.

تم استخدام الزيادة السابقة والتالية للمتغير m، عند إدخال 4 للمتغير m تم تخصيص المتغير n بعد زيادة سابقة للمتغير m يجعل كل من n, m مساوياً 5 وعند تخصيص قيمة ++m للمتغير n نجد أن

المتغير n يأخذ قيمة 5 أولاً، ثم يزيد قيمة m لتصبح 6 وبالتالي يمكننا القول أن للتخصيص أولوية أكبر من الزيادة التالية وأقل من الزيادة السابقة.

مثال 3:

```
#include<iostream.h>
void main()
{
int x, y;
cin>>x>>y;
cout<<x<<'+'<<'='<<x+y<<endl;
}
```

خرج البرنامج:

```
3 8
3+8=11
```

تمارين العملي:

1. **جمع عددين صحيحين وطباعة الناتج؟**
2. **تمارين الزيادة والنقصان؟ اي وضع برنامج بثوابت محددة والمطلوب معرفة خرجه، مثلا:**

```
using namespace std;
# include <iostream>
main()  { int x=-100, y=50; int m;
cout<<"x--="<<x--<<endl;
cout<<"--x="<<--x<<endl;
cout<<"++y="<<++y<<endl;
        cout<<"y--="<<y--<<endl;
 return 0;}
```

3. وضع تمرين للمقارنة بين العبارة البسيطة التي لا تحتاج ({}) والعبارة المركبة التي تحتاجها:

مثلا:

X=10;

اذا كانت قيمة x أصغر من العشرة تطبع عبارة محدة أما اذا كانت تساوي أو أكبر تطبع العبارة وتجري عملية رياضية محددة.

نهاية المحاضرة