كلية العلوم

القسم : **الرياضيات**

السنة : **الاولى**

المادة : لغات البرمجة١

المحاضرة : الثانية /نظري/

{{ مكتبة A to Z }}

Facebook Group : مكتبة A to Z

الفصل الأول

أساسيات البرمجة في لغة C++

1-1 مقدمة

إن جميع لغات البرمجة عالية المستوى تقوم في جوهرها بالشيء نفسه، فهي تؤمن طريقة لكتابة البرامج بأبسط صورة ممكنة باستخدام شيفرة الآلة.

فلم هذا التنوع الكبير في لغات البرمجة؟

لقد طوّرت كل لغة لغرض مختلف وقد تبين لنا أن لغة pascalلغة برمجية هيكلية تستخدم في بيئات أكادمية وذلك لتعليم مفاهيم البرمجة المعقدة.إنها لغة جيدة ولكن معظم المبرمجين يفضلون لغة تفرض قدرا أقل من التعقيدات. وكانت لغة Cمصممة أصلاً لكتابة أنظمة التشغيل، فهي لغة نظيفة تدعم الاختصارات وتجعل من الممكن كتابة برامج مختصرة وقد أثبتت قدرتها على الانتشار بين المبرمجين على مدى السنين، والتي خرجت منها إصدارات مثل C++ و C++.net و C# وهي لغات برمجة غرضية التوجه والتي تعتبر من الجيل الرابع للغات البرمجة.

إن لغات البرمجة التقليدية عادةً غير قابلة للتوسيع كما أن كتابتها وصيانتها معقدة. فكانت الفكرة الأساسية في اللغات البرمجية غرضية التوجه هي دمج المتغيرات والدوال التي تعمل على تلك المتغيرات في غرض واحد يسمى (object)، ولتعديل البيانات المخزنة في الغرض نحدد الدوال التي نتعامل معها (الأعضاء الدالية member functions) في الغرض ثم نقوم بالتعديل المطلوب، حيث لا تستطيع أي دالة أخرى الوصول إلى البيانات في هذا الغرض، وبذلك يمكن كتابة البرامج وصيانتها بشكل أسهل.

2-1 عناصر لغة C++ :

رموز اللغة: تبنى اللغة على مجموعة من الرموز يمكن ان تصنف وفق ما يلي:

- الحروف الانكليزية الصغيرة a,b,…,z
- الحروف الانكليزية الكبيرة A,B,…,Z
- الأرقام العربية 0,1,…..9
- رموز خاصة مثل :

[]	"	!	<	-	+
*	،	||	>	( )	_
>>	<>	<=	>=	\	/
!=	&	%	$	#	<<

الأسماء التعريفية:

وهي الأسماء التي نطلقها على المتغيرات أو الثوابت أو الدوال
وهناك بعض القواعد الناظمة لهذه الأسماء:

1- أن لا يكون الاسم التعريفي أحد الكلمات المحجوزة للغة.
2- تصاغ الأسماء التعريفية من الحروف الانكليزية الصغيرة او الكبيرة (مع ملاحظة أن اللغة تميز بين الأحرف الصغيرة و الكبيرة) أو كليهما معاً، ومن الأرقام (9 …0)
ويمكن استخدام الشرطة السفلية (_) إلا أنه لا يصح أن يبدأ برقم. ولا قيود على طوله

الكلمات المحجوزة:

وهي كلمات قياسية معرّفة مسبقا وتكتب عادةً بحروف صغيرة ولها معاني خاصة بها تؤديها في برنامج C++، ولا يجوز إعادة تعريفها أو استعمالها لغير ما خصصت له، على أن تكتب بأحرف صغيرة وهذه الكلمات المحجوزة هي:

جدول بالكلمات المحجوزة بلغة C++

near	static	asm	double	long	sizeof
do	int	while	new	auto	else
for	this	void	delet	goto	if
const	entity	char	class	public	case
continue	extern	struct	inline	float	private
virtual	volatile	frinde	enum	near	static
cdecl	default	overload	unsigned	typedef	signed
pascal	operator	switch	template	union	register
protected	far	catch	char	break	return

جدول (1-1)

1-3 أنواع البيانات الأساسية في لغة البرمجة C++ :

كل عنصر من البيانات في البرنامج إما ان تكون قيمته ثابتة أو متغيرة (قيمة المتغير ربما تتغير خلال تنفيذ البرنامج). وكل متغير يجب أن يكون له نوع وبموجب هذا النوع سيتم تحديد المساحة التخزينية اللازمة لقيمة هذا المتغير وكذلك تحدد العمليات الممكن إجراؤها على هذا المتغير.

1-3-1 المتغيرات:

تمكن لغة البرمجة C++ من تعريف أي نوع من أنواع متغيرات البيانات ، ولكن الأنواع المسبقة التعريف توفر العناء في أغلب الحالات البرمجية .

هناك سبعة أنواع من متغيرات البيانات في لغة البرمجة C++،أحد هذه الأنواع يمثل المحارف وثلاثة تمثل الأعداد الصحيحة وثلاثة تمثل الأعداد الحقيقية ، والجدول الآتي يلخص هذه الأنواع :

النوع	***الحجم في الذاكرة***	***مجال القيم***
unsigned char	8 bits	0 to 255
char	8 bits	-128 to 127
unsigned int	16 bits	0 to 65535
short int	16 bits	-32768 to +32767
int	16 bits	-32768 to +32767
unsigned long	32 bits	0 to 4294967295
long	32 bits	-2147483648 to 2147483647
float	32 bits	$3.4*10^{-38}$ to $3.4*10^{+38}$
double	64 bits	$1.7*10^{-308}$ to $1.7*10^{+308}$
long double	80 bits	$3.4*10^{-4932}$ to $1.1*10^{+4932}$

جدول(1-2)

الاعلان عن المتغيرات:

الإعلان declaration عن متغير يمثل (هو)عبارة تعطي معلومات عن هذا المتغير لمترجم C++.والإعلان عن المتغير يتم بكتابة النوع أولا ثم يتبع ذلك اسم المتغير.

ويأخذ الشكل الآتي:

Type variables

حيث type اسم لأحد أنواع المتغيرات في C++ فعلى سبيل المثال الاعلان الآتي:

int n ;

يخبر المترجم بأمرين: الأول بأن اسم المتغير هو n، والأمر الثاني هذا المتغير من نوع الأعداد الصحيحة int. كل متغير يجب ان يكون له نوع.

مع ملاحظة أن كل أمر بلغة C++ يجب أن ينتهي بفاصلة منقوطة(;).

وبالإمكان الإعلان عن أكثر من متغير من ذات النوع بنفس الطريقة أعلاه على أن تفصل فارزة (فاصلة) بين اسم متغير وآخر، مثال:

int x,y,z;

في لغة C++ يمكن أن يكون الإعلان في أي مكان في برنامج C++ شرط أن المتغير لا يمكن أن يستخدم قبل الإعلان عنه.

عمليات الإسناد او التخصيص assignment:

أكثر الطرق شيوعا لحصول المتغير على قيمة هي عملية التخصيص أو الإسناد ويتم ذلك باستخدام رمز المساواة (=) وهذه العملية تأخذ الشكل الآتي:

Variable = expression;

التعبير expression تقدر قيمته أولا ثم تخصص هذه القيمة للمتغير variable، عند تخصيص قيمة لمتغير نضع في الطرف الأيسر من المساواة اسم متغير و

في الطرف الأيمن القيمة .القيمة الموجودة في الطرف الأيمن يمكن أن تأخذ أحدى الأشكال الآتية:

قيمة ما – اسم متغير يأخذ قيمة – تعبير حسابي – دالة رياضية

و الموضحة بالأمثلة التالية:

مثال :

```
int n;
n=66;
```

مثال :

```
int a,b;
a=20;
b=a;
```

مثال :

```
int a,b,c;
a=15  ; b= 30;
c=5*a+7*b-3;
```

مثال :

```
int x = 10 , y ;
y=abs(x) ;
```

يمكن في الواقع الإعلان عن المتغير و تخصيص قيمة ابتدائية عند الاعلان عنه مثال:

```
int  n=66;
```

إعطاء قيم ابتدائية يمكن أن يستخدم أيضا عند الإعلان عن أكثر من متغير بعبارة إعلان واحدة كما يوضح المثال الآتي:

```
int n1,n2=55 , n3, n4, n5 = 44, n6 ;
```

أعلن عن المتغيرات n1,n3,n4,n6 على أنها أعداد صحيحة ولكن المتغيرين n2,n5 خصص لهما قيماً في جملة الإعلان.

يمكن لقيمة التخصيص أن تستخدم في تخصيص آخر :

مثال :

```
int a,b;
a=b=25;
```

مثال :

```
int a,b,c;
a=b=c=0;
```

بشكل عام قيمة التخصيص هي آخر قيمة خصصت.

<u>الأنواع القياسية التي تستخدم في لغة C++ هي :</u>

1-3-1 المحارفchar :

يتم تخزين المحرف في متغيرات من النوع **char** كما يلي :

```
char ch ;
```

يحجز المترجم مساحة في الذاكرة لمتغر محرفي ويسمىch.

مثال:لتخزين محرف في المتغير نكتب :

```
ch = 'a' ;
```

الأحرف الثابتة مثل '4' , '&' , 'B' , 'a' يجب أن تكون محصورة بين علامتي اقتباس فردية .

يمكن استعمال المتغيرات ذات النوع **char** أيضاً من أجل تخزين أرقام صحيحة بدلاً من الأحرف مثل :

```
char c;
c = 65 ;
```

عند طباعة المتغير c يظهر الرمز A وليس وليس الرقم 65 .علماً ان الرقم 65 يمثل الرمز A في جدول الآسكي.

سلاسل الهروب :

هي عبارة عن رموز تحكم تستخدم للقيام بمهام معينة ضمن البرنامج C++. حيث يوجد ضمن C++ عدة سلاسل هروب، وكل منها يقوم بأداء المهمة الموكلة إليه لدى استخدامه.

إن أكثر سلاسل الهروب استخداماً هما : \n ، \t

ضمن C++ يوجد عدة أحرف خاصة تلي الشرطة المائلة (إشارة التقسيم) ، تسمى هذه الحروف رموز تحكم لأن الشرطة المائلة تجعل تفسير الحرف الذي يلي "يهرب" من جدول الأسكي ويشير إلى شيء مختلف ، ويبين الجدول الآتي بعض سلاسل هروب:

سلاسل الهروب	الحرف المقصود
'\a'	إصدار صوت الجرس
'\b'	حرف التراجع backspace
'\f'	للتقدم صفحة واحدة للأمام
'\r'	حرف الإرجاع يجبر المؤشر على الانتقال إلى بداية هذا السطر (يتم توليده

	أيضاً بضغط enter)
'\n'	سطر جديد يجبر المؤشر على الانتقال إلى بداية السطر التالي .
'\\'	لاستخدام الاشارة \
' \' '	لاستخدام اشارة التنصيص المفردة '
' \" '	لاستخدام اشارة التنصيص المزدوجة "

جدول (1-3)

1-3-2 متغيرات الأعداد الصحيحة :

هناك ثلاثة أنواع من متغيرات الأعداد الصحيحة في لغة البرمجة C++ هي : short (قصير) و int (عدد صحيح) و long (طويل) ، وهي متشابهة فيما بينها لكنها تحتل مساحات مختلفة في الذاكرة ويمكنها معالجة أرقام تقع في مجالات مختلفة .

مثال :

تعريف بعض المتغيرات ذات النوع عدد صحيح وإسناد قيماً لها كما يلي :

```
int x ;  long y ;
x = 321 ;    y = 1506823L ;
```

يتم استعمال الحرف L لتحديد ثابت من النوع **long** .

الجدول الآتي يبين مجالات الأعداد الصحيحة (وقد تم ذكر النوع **char** – رغم أنه يستعمل في أغلب الأحيان للمحارف – لأنه يمكن استعماله لتخزين أرقام صحيحة صغيرة) :

النوع	***الحجم***	***المجال***
char	1 bytes (8 bits)	[-128 , 127]
int	مثل short في الأنظمة التي تعمل بـ 16 bits ومثل long في الأنظمة التي تعمل بـ 32 bits	

long	4 bytes (32 bits)	[-2147483648 , 2147483647]

جدول (1-4)

متغيرات الأعداد الصحيحة التي ليس لها إشارة :

إن كل أنواع متغيرات الأعداد الصحيحة لها إصدارات من دون إشارة (unsigned)، والمتغيرات التي ليس إشارة لا تستطيع تخزين أرقام سالبة ، لكن حجم مجال قيمها الموجبة يساوي ضعف حجم المجال التي لها إشارة ، تكون الأعداد الصحيحة الاعتيادية من دون ميزة unsigned لها إشارة بشكل افتراضي .

يبين الجدول الآتي متغيرات الأعداد الصحيحة التي ليس لها إشارة :

النوع	***الحجم***	***المجال***
unsigned char	1 bytes (8 bits)	[0 , 255]
unsigned short	2 bytes (16 bits)	[0 , 65535]
unsigned int	مثل unsigned short في الأنظمة التي تعمل بـ 16 bits ومثل unsigned long في الأنظمة التي تعمل بـ 32 bits	
unsigned long	4 bytes (32 bits)	[0 , 4294967295]

جدول (1-5)

1-3-3 متغيرات الأعداد الحقيقية (الأرقام العائمة) :

يتم تمثيل الأعداد الحقيقية عادة برقم صحيح على اليسار مع نقطة عشرية أو كسر على اليمين وبدلاً من النقطة العشرية يمكن استعمال الأسي لذا القيمة 124.65في الشكل العادي هي1.2465e2في الشكل الأسي ، حيث يشير الرقم الذي يلي الحرفeإلى كم مرة يجب نقل

النقطة العشرية إلى اليمين لاسترجاع القيمة إلى الشكل العادي ، وغالباً ما يستعمل الشكل الأسي لعرض الأرقام الطويلة في الشكل العشري .

مثال :

العدد 000 000 000 000 000 876 9 في الشكل العادي هو في الشكل الأسي 9.876e18 .

هناك ثلاثة أنواع من متغيرات الأعداد الحقيقية في أنظمة التشغيل مبينة في الجدول التالي :

اسم النوع	***الحجم***	***المجال***	***الدقة***
float	4 bytes (32 bits)	[10e-38 , 10e38]	5 أعداد
double	8 bytes (64 bits)	[10e-308 , 10e308]	15 عدداً
long double	10 bytes (80 bits)	[10e-4932 , 10e4932]	19 عدداً

جدول (1-6)

إن أشهر نوع من متغيرات الأعداد الحقيقية هو النوع **double** الذي يستعمل في معظم الدوال المكتبية الرياضية في لغة البرمجة C++ .

يتطلب النوع **float** ذاكرة أقل من النوع **double** وقد يسرع إنجاز العمليات الحسابية ، ويستعمل النوع long double في معالج الأرقام الكبيرة .

مثال على تعريف واستعمال متغيرات من أنواع الأعداد الحقيقية المختلفة :

```
float  pi_float ;
double  pi_double ;
long double  pi_long_double ;
```

<u>العمليات الحسابية :</u>

تضمن لغة البرمجة C++ العمليات الحسابية الأربعة بالإضافة إلى عملية باقي القسمة كما هو موضح في الجدول التالي:

العملية	وظيفته
+	الجمع
–	الطرح
*	الضرب
/	القسمة الصحيحة
%	باقي القسمة

جدول (1–7)

إن عملية باقي القسمة % فتقوم لاحتساب باقي قسمة عدد صحيح على عدد صحيح آخر ، فمثلاً التعبير (3 % 20) يساوي (2) لأن 20 مقسومة على 3 هي 6 والباقي 2 .

تستعمل العمليات الحسابية بشكل مشابه لاستعمالها في لغات البرمجة الأخرى وهي تسمى عمليات ثنائية لأنها تعمل على كميتين فبذلك يمكن أن نكتب التعابير الرياضية بالطريقة التي نريدها .

العمليتان (* و /) لهما أولوية التنفيذ على العمليتين (+ و –) فإذا كان المطلوب أولاً تنفيذ عملية جمع أو طرح فيجب وضعهما ضمن قوسين .

مثال :

```
c = (f – 32) * 5 / 9 ;
```

هنا يتم أولاً تنفيذ عملية الطرح ثم الضرب والقسمة (وهذه العبارة للتحويل من درجة مئوية إلى فهرنهايت) .

عملية التزايد والتناقص :

تمكن لغة البرمجة C++ من استخدام عمليتين خاصتين تنفذان زيادة (1) إلى المتغير أو طرح (1) إلى المتغير هما عملية الزيادة و عملية النقصان .

عملية الزيادة :

يرمز لها بالرمز ++ وعند استخدامها فإن المترجم يزيد قيمة المتغير بمقدار (1) وهنا نميز حالتين :

*) الزيادة على المتغير قبل التنفيذ :

```
int i ;
++ i ;
```

*) الزيادة على المتغير بعد التنفيذ :

```
int i ;
i ++ ;
```

في هذه الحالة يتم تنفيذ العملية المخصصة على المتغير ثم يقوم المترجم بالزيادة .

ملاحظة :

يمكن زيادة قيمة المتغير بالقيمة التي نريد ونراها مناسبة وكمثال :

```
int i ;
i = i + 5 ;
```

عملية النقصان :

يرمز لها بالرمز -- وعند استخدامها فإن المترجم ينقص قيمة المتغير بمقدار (1) وهنا نميز حالتين :

*) إنقاص المتغير قبل التنفيذ :

```
int i ;
-- i ;
```

*) إنقاص المتغير بعد التنفيذ :

```
int i ;
i -- ;
```

ملاحظة :

يمكن إنقاص قيمة المتغير بالقيمة التي نريد ونراها مناسبة وكمثال :

```
int i ;
i = i - 5 ;
```

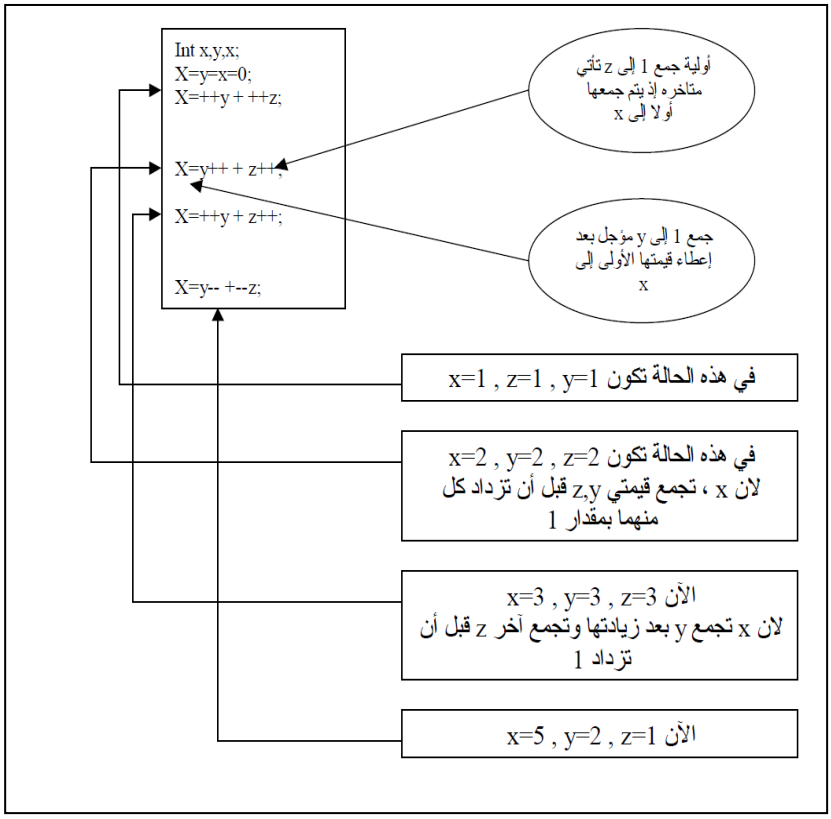
تسمى عمليتا التزايد والتناقص عمليتان أحاديتان لأنهما تعملان على متغير واحد فقط ولهما الأولوية على العمليات الحسابية .

ملاحظة :

تمكن لغة البرمجة C++ من تنفيذ العمليات الحسابية على المتغير بإحدى الطريقتين :

```
int  x ;
x =  x + 3 ;    أو x + = 3 ;
x =  x - 3 ;    أو x - = 3 ;
x =  x * 3 ;    أو  x * = 3 ;
x =  x / 3 ;    أو x / = 3 ;
```

مثال:



1-3-4 النوع المنطقي (boolean)

لايوجد ضمن C++ نوع معطيات بوليانية مثلما هو موجود في لغة الباسكال مثلاً ، و لكن بدلاً من ذلك نستخدم القيمتين 0,1 للتعبيرعن الحالة المنطقية(true,false) و لكن استخدام نوع المعطيات التعدادية يتيح لنا تعريف نوع بولياني و ذلك كما يلي:

```
enum  boolean { false,true};
```

مثال :

```
enum  boolean { false,true};
Boolean isword=false;
```

الأدوات العلائقية والمنطقية :

الأدوات العلائقية موضحة في الجدول التالي :

الرمز	المعنى	مثال
= =	يساوي	a = = b
! =	لا يساوي	a ! = b
<	أصغر من	a < b
>	أكبر من	a > b
< =	أصغر من	a <= b
> =	أكبر من	a >= b

جدول (1-8)

تمكننا الأدوات المعرفة في لغة البرمجة C++ من مقارنة المحارف والأرقام لأن المحارف لها قيم في جدول أسكي ، لذا التعبير 'a' < 'b' والتعبير 'a' ==56 هما صحيحان على عكس التعبير 'a' => 'z' لأن المحرف z له قيمة أسكي أكبر من قيمة المحرف a .

أما الأدوات المنطقية فهي :

تتضمن لغة البرمجة C++ ثلاث أدوات منطقية اثنان منها تدمج قيمتين والثالثة تعكس القيمة كما هو موضح في الجدول التالي :

جدول الأدوات المنطقية:

الأداة المنطقي	معناه	مثال
&&	AND	x>0 && x<10
||	OR	x == 3 || x < 1
!	NOT	! x

جدول (1-9)

*) تكون العبارة AND صحيحاً فقط إذا كان التعبيران الموجودان على جانبي المعامل && صح ، وغالباً ما يتم استعمال AND لتحديد فيما إذا كان متغير ما يقع ضمن مجال محدد .

مثال :

```
x > 0  &&  x <= 10
```

تكون العبارة صحيحة إذا كان المتغير x يقع في المجال [10 .. 1] .

```
ch >= 'a'  &&  ch <= 'z'
```

تكون العبارة صحيحة إذا كان المتغير ch حرفاً صغيراً .

*) تكون عبارة OR صحيحة إذا كان أحد التعبيرين أو كلاهما صحيحاً ، وغالباً ما يتم استعمال OR لتحديد فيما إذا كانت قيمة متغير ما تقع خارج مجال محدد .

مثال :

```
x < 75  ||  x > 100
```

تكون العبارة صحيحة إذا كان المتغير x يقع خارج المجال [100 .. 75] .

*) يكون المعامل NOT صحيحاً إذا كان التعبير الذي يليه خطأ ويكون خطأ إذا كان التعبير صحيحاً .

أولوية العمليات الحسابية والمنطقية :

الجدول الآتي يوضح ترتيب الأولويات لمختلف الأدوات السابقة :

الأداة	نوعها	الأولوية
* / %	الضرب – القسمة – باقي القسمة	أعلى
+ -	الجمع – الطرح	
!= == >= <= ><	علائقية	
&&||	منطقية (و ، أو)	
=	تخصيص	أدنى

جدول (1-10)

ملاحظة:

عند تساوي الأولويات سيتم تنفيذ العمليات من اليسار إلى اليمين.

1-3-2 الثوابت Constants:

وهي عبارة عن قيم ثابتة لا تتغير طوال تنفيذ البرنامج وتنقسم إلى ثوابت عددية وثوابت رمزية.

1- الثوابت العددية وتشمل:

- الثابت العددي الصحيح
- الثابت العددي الحقيقي

يمكن الإعلان عن الثوابت باستخدام الكلمة المفتاحية const:

const TYPE *variable_name* = *value* ;

مثال:

const float Pi = 3.1413926535 ;

2- الثوابت الرمزية:

وهي عبارة عن رموز اللغة وتتكون من الحروف والأرقام وتكون بين علامتي اقتباس فردية

مثال:

const char ch =' N ' ;

المسافات الفارغة :

تمكن لغة البرمجة C++ من وضع مسافات فارغة إضافية في أسطر البرنامج لتسهيل القراءة والكتابة .

مثال :

```
int  x    =12 ;
float  y    =3.14 ;
```

حيث يمكن وضع قدر ما نشاء من المسافات وعلامات الجدولة والسطور الجديدة في البرنامج ويتجاهلها المترجم في لغة البرمجة C++ .

التعليقات :

تساعد التعليقات على القراءة الصحيحة للبرنامج ومن ثم على الفهم الكامل لطريقة عمله وتسلسل أفكاره ، وكلما كان البرنامج مدعما بالتعليقات المناسبة كلما دل ذلك على أناقة المبرمج الذي قام بتطوير هذا البرنامج.

تستخدم التعليقات غالباً لإعطاء نبذة بسيطة عن عمل دالة، أو تعريف الهدف من استخدام متغير ما، أو حتى توضيح الهدف من استخدام جزء معين من الكود. تستعمل جملة التعليق في أي مكان من البرنامج لإبداء ملاحظة ما عند سطر معين في البرنامج ولا تعتبر جملة تنفيذية.

يوجد نوعين من التعليقات في لغة C++ :

تعليق السطر :

ويستخدم لكتابة تعليق مكتوب من سطر واحد فقط، وهذا النوع دائماً يبدأ بوضع الإشارة المزدوجة // قبل التعليقات وأي نص يلي هذا الرمز إلى نهاية السطر فقط يتجاهله المترجم، ويمكن استعمال التعليق في كامل السطر أو تعليقات تلي كل عبارة ، مع مراعاة عدم وجود فراغ بين الإشارتين.

مثال :

```
// my first program .
```

تعليق الفقرة:

هذا النوع من التعليق يتيح لنا كتابة التعليق على سطر أو عدة أسطر حيث يبدأ التعليق بالرمز */ وينتهي بالرمز /* .

مثال :

```
/*
I am student .
I write my program .
*/
```

حيث يعتبر المترجم كل الحروف والمسافات والأسطر الموجودة بين التركيب ضمن نطاق التعليق.

ملاحظة: – لا ينصح بوضع تعليق ضمن تعليق آخر، لكن يمكن وضعه في أي مكان من البرنامج، ما عدا وسط اسم تعريفي أو كلمة محجوزة.