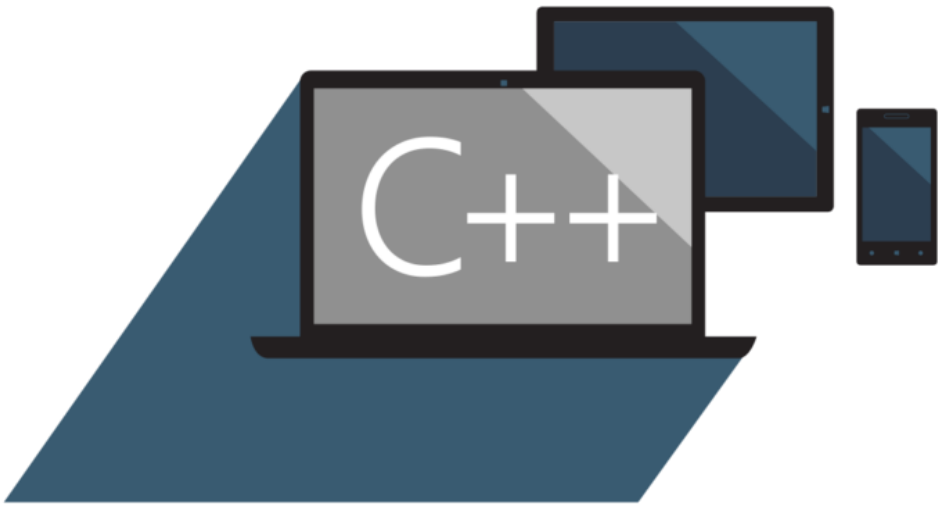
C++

the second Lecture



الكلية التربية

قسم الرياضيات

١-٦ التعليقات Comments:

وهي العبارات التي لا تقرئ ضمن البرنامج واذا كان التعليق عبارة عن سطر واحد يتم سبقه بعلامة (//).

مثال (٧):

```
// comment with one line
```

اما اذا كان التعليق لأكثر من سطر فيبتدأ بعلامة (/*) وينتهي بعلامة (*/).

مثال (٨):

```
/* this is a comment
 using with multiline  */
```

١-٧ الرموز غير المطبوعة Non-printable character:

الرموز غير قابلة للطبع نعني بها التي لا تملك صورة مرئية عن الكتابة مثل رمز مفتاح الادخال Enter او الحذف الخلفي Backspace، وبعض الرموز الاخرى التي تكون لها معاني خاصة في اللغة اذ سيعوض عنها برموز اخرى لتمثيلها عند الاستخدام توضع بين علامات الاقتباس المفردة او علامات اقتباس مزدوجةsingle and double Quotation تعرف هذه الرموز في لغة C++ بسلاسل الهروب Escape Sequences، والجدول (١-٣) يستعرض بعضاً منها.

الجدول (١-٣)

newline	\n	horizontal tab	\t
vertical tab	\v	backspace	\b
carriage return	\r	formfeed	\f
alert (bell)	\a	backslash	\\
question mark	\?	single quote	\'
double quote	\"		

مثال (٩):

```
#include<iostream.h>
void main()
{
cout<<"Dr. ALI \t A. L. AHMED \n**************************";
}
```

ستكون المخرجات بالشكل الاتي:

```
Dr. ALI       A. L. AHMED
**************************
```

اذ ان رمز "\t" اعطى مسافة محددة بثمان فراغات (حسب النظام المستخدم) والرمز "\n" حول الطباعة الى سطر جديد فظهرت سلسلة النجمات على السطر الثاني.

١-٨ المشغلات الحسابية Arithmetic Operators:

تزود لغة C++ خمس مشغلات حسابية اساسية وهي موضحة في الجدول (١-٤).

الجدول (١-٤)

Operator	Name	Example
+	Addition الجمع	12 + 4.9 // gives 16.9
-	Subtraction الطرح	3.98 - 4 // gives -0.02
*	Multiplication الضرب	2 * 3.4 // gives 6.8
/	Division القسمة	9 / 2.0 // gives 4.5
%	Remainder باقي القسمة	13 % 3 // gives 1

كل المشغلات الحسابية تقبل المزج بين الحدد الصحيح والحقيقي عدا مشغل باقي القسمة (%)، طبعا اذا كان كلا المعاملان عدد صحيح فسيكون الناتج عدد صحيح، اما اذا كان احد المعاملين عدد حقيقي فسيكون الناتج في الغالب عدد حقيقي.

ان ناتج القسمة وان كان بين رقمين صحيح سيؤدي الى ناتج صحيح (اذ سيتم تقريب العدد الناتج الى اقرب عدد صحيح وليس كما تعودنا عليه).

مثال (١٠):

```
9 / 2 // gives 4, not 4.5!
-9 / 2 // gives -5, not -4!
```

للحصول على ناتج بالعدد الحقيقي يجب ان تجعل احد المعاملات بالعدد الحقيقي.

مثال (١١):

```
int cost = 100;
int volume = 80;
double unitPrice = cost / (double) volume; // gives 1.25
```

اذا كان ناتج العملية كبير جدا عن الخزن في المتغير المخصص لحفظ الناتج هذه الحالة تدعى **بالفيض Overflow**.

١-٩ اسبقية المشغلات الحسابية Arithmetic Operators Precedence

لكل مشغل اسبقية تنفيذ ضمن العمليات الحسابية التي تنفذ ضمن اللغة.

مثال (١٢) لحل المشكلة الاتية:

```
int var = 2 * 3 + 1;
```

فان عملية الضرب سيتم تنفيذها اولا لتنتج العدد ٦ ثم يتم تنفيذ عملية الجمع مع العدد ١ لتنتج الرقم ٧.

اما اذا كانت العملية تحوي على اقواس لحصر المعاملات سيتم تنفيذ الاقواس الداخلية ومن ثم الاقواس الخارجية، فمثلاً:

((a + b) + c)

اذ سيتم تنفيذ عملية جمع المعاملين a وb ومن ثم جمع الناتج مع المعامل c، ولا ننسى ان كان هنالك مشغلات من نفس الاسبقية فانه ننفذ المعادلة بترتيب من اليسار الى اليمين.

وهنا نؤكد ان الاولوية في التنفيذ تكون للأقواس ومن ثم لعمليات الضرب والقسمة وباقي القسمة وبعدها تكون للجمع والطرح، وهنا نورد مثال عن اسبقية التنفيذ من خلال المعادلة الاتية:

```
z=   p   *   r   %   q   +   w   /   x   -   y   ;
6        1       2       4       3       5
```

اذ توضح الارقام اسفل المعادلة التسلسل المنطقي لتنفيذ المعادلة وحسب قواعد الاسبقيات.

١٠-١ مشغلات التعيين Assignment Operators

ان عملية تحويل القيم من اليمين الى اليسار يشترط فيها ان المعامل الموجود في جهة اليسار ليس متغير معرف على انه ثابت اي لا تسبقه عبارة const عند التعريف، ونورد هنا مجموعة من الامثلة الخاطئة عن تعيين القيم لمتغيرات.

```
int i, j, ival;
const int ci = i;  // لا توجد قيمة للمتغير المحول.
1024 = ival;       // لا يجوز التحويل الى قيمة ثابته.
i + j = ival;      // لا يجوز وضع عملية رياضية في جهة اليسار.
ci = ival;         // لا يجوز اعطاء قيمة جديدة لمتغير معرف على انه ثابت.
```

وهنا يجب ملاحظة انه يمكن تخصيص قيمة لأكثر من متغير على ان تكون من نفس نوع التعريف، مثلاً:

```
int ival, jval;
   ival = jval = 0; //  each assigned 0
```

١١-١ الدوال الرياضية القياسية Standard Mathematical Functions

تحتوي مكتبة <math.h> على مجموعة كبيرة من الدوال الرياضية التي تساعد في حل العديد من المسائل الجبرية، ونورد هنا عدداً من هذه الدوال مع تعريفها:

```
double abs(double) ;     / / absolute value;
double sqrt(double d) ; / / square root of d, d must be nonnegative
double pow(double d, double e) ; / / d to the power of e,
```

// error if d==0 and e<=0 or if d<0 and e isn't an integer.
double pow(double d, int i) ; / / d to the power of i;
double cos(double) ; / / cosine
double sin(double) ; / / sine
double tan(double) ; / / tangent
double exp(double) ; / / exponential, base e
double log(double d) ; / / natural (base e) logarithm, d must be > 0
double log10(double d) ; / / base 10 logarithm, d must be > 0

مثال (١٣): برنامج لإيجاد مساحة ومحيط الدائرة. ونصف القطر يساوي ١٠.

```
#include<iostream.h>
#include<math.h>
void main()
{
const float pi=3.14; int r=10;
double v,a;
a=pow(r,2)*pi;
v=2*r*pi;
cout<<"volume="<<v<<endl;
cout<<"area="<<a<<endl;
}
```

في المثال السابق تم استخدام دالة pow لايجاد تربيع نصف القطر، واستخدام الامر endl لانهاء الطباعة على السطر الاول وليطبع ناتج المساحة a على سطر جديد.

١-١٢ مشغلات الزيادة والنقصان Increment/ Decrement Operators

ان الزيادة التلقائية (++) والنقصان التلقائي (--) يوفر طريقة ملائمة للزيادة والنقصان بمقدار واحد.

مثال (١٢): بعد تعريف المتغير k:

```
int k=5;
```

سنوضح التأثير الذي سيحدث ان كانت مشغلات الزيادة والنقصان قبل المعامل او بعده في الجدول (١-٥).

الجدول (١-٥)

Operator	Name	Example	
++	Auto Increment (prefix)	++k + 10	// gives 16
++	Auto Increment (postfix)	k++ + 10	// gives 15
--	Auto Decrement (prefix)	--k + 10	// gives 14
--	Auto Decrement (postfix)	k-- + 10	// gives 15

ومن خلال الجدول (١-٥) نلاحظ ان علامة (++) اذا جاءت قبل المتغير فان الزيادة ستنفذ في الخطوة الحالية، اما اذا جاءت العلامة (++) بعد المتغير فان الزيادة ستنفذ للخطوة التالية، وكذلك بالنسبة للمشغل (--).

وهنا نذكر ان عمليات الزيادة والنقصان على المتغيرات يمكن ان تكون بأكثر من واحد وتكتب بالشكل الاتي في لغة C++ سواء كان التغيير بقيم رقمية ثابتة، الجدول (١-٦).

الجدول (١-٦)

Mathematic Expression	C++ expression
X=X+2	X+=2;
X=X-3	X-=3;
X=X*2	X*=2;
X=X/2	X/=2;

او كان التغيير بقيم متغيره، الجدول (١-٧).

الجدول (١-٧)

Mathematic Expression	C++ expression
X=X+Y;	X+=Y;
X=X-Y;	X-=Y;
X=X*Y;	X*=Y;
X=X/Y;	X/=Y;

مثال (١٣): يبين اظهار التأثيرات المتسلسلة عندما يكتب برنامج كامل يحتوي على خطوات الزيادة التلقائية وبمقدار واحد.

```
#include<iostream.h>
void main()
{
int x,y,z;
x=y=z=0;
x=++y + ++z;
cout<<x<<y<<z<<endl;
x=y++ + z++;
cout<<x<<y<<z<<endl;
x=++y + z++;
cout<<x<<y<<z<<endl;
x=y-- + --z;
cout<<x<<y<<z<<endl;
}
```

مخرجات الشاشة

```
--------------------
211
222
533
522
```

١-١٣ المشغلات العلائقية Relational Operators:

توفر لغة C++ ستة مشغلات لمقارنة الكميات الرقمية وحسب الجدول (١-٨).

الجدول (١-٨)

Operator	Name	Example
==	Equality تساوي	5 == 5 // gives 1
!=	Inequality لا تساوي	5 != 5 // gives 0
<	Less Than أقل من	5 < 5.5 // gives 1
<=	Less Than or Equal اقل من او يساوي	5 <= 5 // gives 1
>	Greater Than أكبر من	5 > 5.5 // gives 0
>=	Greater Than or Equal أكبر من أو يساوي	6.3 >= 5 // gives 1

عندما تكون نتيجة العلاقة صحيحة True تعطي رقم ١ واذا كانت نتيجة العلاقة خطأ False تعطي صفر.

١-١٤ المشغلات المنطقية Logical Operators:

تزودنا لغة C++ بثلاث مشغلات للجمع بين تعابير منطقية، وهي كالمشغلات العلائقية تزودنا بنتيجة (١،٠)، الجدول (١-٩).

الجدول (١-٩)

Operator	Name	Example
&&و	Logical And"و" علاقة	5 < 6 && 6 < 6 // gives 0
||أو	Logical Or" أو " علاقة	5 < 6 || 6 < 5 // gives 1
!نفي	Logical Negation (Not) علاقة النفي	!(5 == 5) // gives 0

وللتعويض عن التعريف المنطقي في لغة C++ والمسمى Boolean يتم استخدام التعويض الرقمي ويعرف بشكل صحيح int.