طريق البرمجة في

هاني الأتاسي – الفريق العربي للبرمجة
آخر تعديل في 11/20/2001

الآن من أجل كتابة البرنامج الأول يجب علينا من انشاء ملف كود جديد وهذا من الأمر New من القائمة File . سوف تحصل على نفس النافذة في الشكل 1-1 ولكن هنا الصفحة Files هي الظاهرة عوضا عن Projects . طبعا يمكنك أن تخمن ماهو الذي سوف نختاره . طبعا هو C++ Source File . بعد أن تختاره اكتب اسم الملف وليكن test ومن ثم اضغط موافق .

سوف تجد أنه تم اضافة الملف test.cpp إلى الدليل الوهمي Source Files وذلك بسبب أن هذا الملف سوف يحتوي على كود للمشروع . الآن قم بكتابة الكود التالي في الملف الفارغ الذي أنشئته .

```
#include <iostream.h>

int main()
{
     cout << "My first program.";
     return (0);
}
```

PROGRAM 1

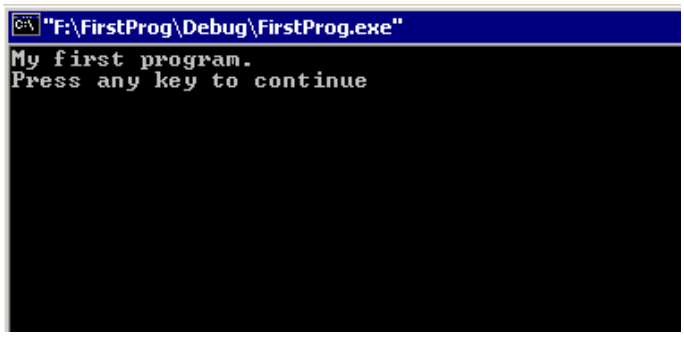
يجب أن تنتبه أن لغة السي++ تفرق بين الأحرف الكبيرة والأحرف الصغيرة . على سبيل المثال فإن اسم التابع main يجب أن يكتب كما هو ولا يمكن كتابته ك Main أو MAIN وهي كلها تعتبر كأسماء مختلفة عن التابع main .

1.2 ترجمة وبناء المشروع

من أجل تنفيذ البرنامج السابق وبناء ملف تنفيذي exe قابل للتنفيذ في أي وقت يحب أولا ترجمته وتحويله من اللغة المقروئة إلى لغة الآلة التي كلها أصفار وواحدات وبعدها يمكن تنفيذه .

عملية ترجمة المشروع وبناء الملف التنفيذي في الفيجوال سي++ تتم بخطوة واحدة وهي عن طريق اختيار الأمر Build من القائمة Build أو الضغط على F7 . ومن أجل تنفيذ المشروع أو الملف التنفيذي اختر الأمر Execute من القائمة Build أو عن طريق المفتاحين Ctrl+F5 . طبعا يمكنك أولا بناء الملف التنفيذي FirstProg.exe ومن بعدها تنفيذه بخطوتين ولكن الأمر Execute يقوم بشكل آلي بعملية الترجمة وبناء الملف التنفيذي FirstProg.exe إذا دعا الأمر إلى ذلك .

إذا كان كل شئ على مايرام سوف ترى نافذة مماثلة لنافذة الدوس أو ال Command Line وسوف ترى العبارة My first program في أعلى الشاشة كما في الشكل 1-4 .



شكل 1 -5

1.3 الأهم من تعلم لغة البرمجة

هل سئلنا أنفسنا في لحظة من اللحظات ماهم الأهم من تعلم لغة البرمجة من أجل أن نصبح مبرمجين ؟ بصراحة الإجابة على هذا السؤال ضخمة جدا ويوجد كتب كاملة عن هذا الموضوع أو عن موضوع هندسة البرمجيات . الإجابة هي كلمتين طبعا وهي هندسة البرنامج ، تماما كما يفعل إخواننا في الهندسة المدنية أم المعمارية . تخيل برنامجك هو عبارة عن عمارة أو بناء تريد تشييده ، سوف تبدأ بالتصميم وتحليل مواردك وامكانياتك ومن ثم وضع (سكيتش) أو تصميم أولي للبناء وبعد ذلك تبدأ بالبناء بلوك بلوك وطابق طابق .

لن أتطرق في هذه الدروس إلى مواضيع التصميم وتحليل الموارد لأنها قد تكون خارجة عن اطار هذه الدروس ولكن كدروس لتعليم لغة السي++ يجب أن أنوه إلى مواضيع تنظيم البرنامج والتي هي جزء لا يتجزء من هندسة البرنامج.

سوف أتطرق الآن إلى أهم الأمور وأسهلها وهي اضافة التعليقات في برنامجك . التعليقات هي جزء من البرنامج الذي يهمل أثناء عملية ترجمة البرنامج وهي تفيد في تفسير الخطوات التي تقوم بها . سوف أطلب منك مقارنة البرنامجين التالين وملاحظة أيهما أسهل للفهم بغض النظر عن لغة السي++ المستخدمة التي قد لا تكون ملم بها في هذه اللحظة .

PROGRAM 1.2 (a)

```
int main()
{
int sum;
sum = 10 + 20;
cout << sum << endl;
return (0);
}
```

PROGRAM 1.2 (b)

```
/*
**  This program calculates the sum of 10 and 20 and prints it on screen
**  Programmed by : Anonymous
**  Date : 1/1/3000
*/

int main()
{
    int sum;                // This will hold the sum of the numbers

    sum = 10 + 20;          // Store 10 + 20 in sum
    cout << sum << endl;    // Print the sum value

    return (0);             // End the program
}
```

طبعا سوف تقول أن البرنامج الثاني أسهل للقرائة والفهم من الأول لعدة أسباب .

- استخدام التعليقات أينما دعت الحاجة لها .
- وضع أسطر فارغة بين قسم وآخر في الكود ، الأقسام تكون مترابطة منطقيا فيما بينها .
- ترك مسافات بادئة من أجل سهولة القرائة وتتبع البرنامج .

بالنسبة إلى التعليقات فيمكن كتابتها بشكلين كما هو موضح في البرنامج الأخير . الأولى باستخدام // وهي تقوم باعتبار كل الذي بعدها حتى نهاية السطر كتعليق . أما الشكل الآخر باستخدام */ و /* الرمز الأول يفتح فقرة تعليق والرمز الثاني يغلق هذه الفقرة ، ذكرت فقرة لأن هذا النوع من التعليق ممكن أن يمتد على عدة أسطر أو يمكن أن يكون في سطر واحد .

1.4 ملخص الدرس

- من أجل البدأ بمشروع برمجي يجب انشاء مشروع جديد في الفيجوال سي++ عن طريق اختيار New من القائمة File .
- يتم انشاء ملفات كود جديدة واضافتها للمشروع أيضا من خلال الأمر New من القائمة File .
- قمنا بكتابة أول برنامج في السي++ وتنفيذه .
- تعلمنا كيفية كتابة تعليقات في البرنامج .

من أجل تنزيل ملفات مشروع هذا الدرس انقر هنا .

Types, Operators, and Expressions – الأنواع والعوامل والتعابير

البرنامج في السي++ يصف العمليات والمهام التي سوف تطبق على العناصر التي تحمل البيانات . من الأمور المهمة جدا كما هو الحال في باقي اللغات هو عملية تحديد ما هو نوع هذه العناصر أو بكلمة أخرى أنواع المعطيات. يمكن تصنيف هذه الأنواع في السي++ كأنواع أساسية وأنواع مشتقة . على سبيل المثال الأنواع المشتقة هي تلك الأنواع التي تعتمد على الأنواع الأساسية مثل المصفوفات .

إن البيانات التي يتم التعامل معها في البرنامج يمكن تصنيفها إلى قسمين . الأولى هي التي تبقى ثابتة طول فترة تنفيذ البرنامج ، أما الثانية هي التي تتغير قيمتها . الاسم الذي يطلق على الأولى هي الثوابت أو Constants أما الثانية فسوف نطلق عليها اسم المتحولات أو Variables . طبعا كلا الصنفين السابقين يجب أن تنطبق عليه ماذكرناه بالنسبة إلى الأنواع ؛ على سبيل المثال المتحولات يجب تحديد نوعها حتى نستطيع التعامل معها .

أما التعبير expression فهو عبارة عن الخطوات والقوانين التي سوف يتم من خلالها حساب قيمة معينة . التعبير يتكون من مجموعة من الحدود operands أو القيم و العوامل operators . لغة السي++ غنية جدا بالعوامل التي تعكس العمليات التي تتم في المعالج مثل عمليات الجمع والضرب والازاحة . العوامل يمكن تصنيفها إلى عدة أصناف أيضا فمنها عوامل من أجل العمليات الحسابية كالجمع والضرب ومنها عوامل المقارنة كالأصغر والأكبر . سوف نقوم بشرح بعض العوامل في هذا الدرس مع تكميلها في باقي الدروس .

- هذا الدرس يوجد فيه عدد هائل من المعلومات مع قليل من التطبيق ، لذا ليس من الضروري فهمه بشكل كامل ولكن يجب قرائته بشكل كامل والرجوع إليه إذا دعت الحاجة في الدروس اللاحقة .

2.1 الثوابت الصحيحة – Integer Constants

هذه الثوابت هي عبارة عن أعداد قد تكون موجبة أو سالبة . ويمكن تمثيلها كأعداد بالنظام العشري أو الثماني أو الست عشري . الأعداد بالنظام العشري تتكون من الأرقام من 0 إلى 9 والرقم الأول يجب أن يكون غير الرقم 0 . وكأمثلة على هذا لدينا الأعداد :

1234 801 9999 10

كلنا نعلم أن الحاسب الآلي يستطيع التعامل مع مجال محدد للأعداد . دقة هذه الأعداد تعتمد على المساحة التي يحجزها المترجم للعدد ، وهذا يختلف من حاسب إلى آخر ففي الأجهزة التي تحتوي على مسجلات بحجم 16بت فإن ال integer سوف يحجز له 16بت وبالتالي فإن مجاله يتراوح بين 32768- إلى 32767 . في الأجهزة الحالية وتحت نظام وندوز 32بت فإن ال integer يحجز له 32بت وبالتالي يكون مجاله بين 2147483648- إلى 2147483647 .

بالنسبة إلى الأعداد في النظام الثماني فإنها يجب أن تبدأ بالرقم 0 وهذه أمثلة عليها :

0235 0123 0777 020

أما الأعداد بالنظام الست عشري فهي تبدأ بالرمز 0x أو 0X وهذا أمثلة على هذا :

0x0A2F 0XECA2 0x1234 0xFFFF

في حال تخطى الثابت المجال المسموح فيه فإن معظم المترجمات تقوم بالتحذير عن ذلك .

- في الأجهزة 16بت فإنه يمكن اجبار المترجم على اعتبار العدد ذو حجم 32بت باضافة اللاحقة L أو (الحرق الصغير l) مثل التالي 0x23A1L أو 982L .

2.2 الثوابت للأعداد الحقيقة – Floating point Constants

أي رقم يحتوي على قسم كسري يعتبر عدد حقيقي . وتكتب هذه الأرقام بصيغتين . الصيغة المباشرة والبسيطة أمثلة على هذا :

3.1415926 12.0 0.1 0.00001

النقطة التي تفصل القسم الحقيقي والكسري لا تشترط أن تحطاط برقمين حيث يمكن كتابة الرقم الثانية والثالث في المثال السابق كالتالي :

12. .1

ولكن يفضل ، على كل الأحول ، استخدام الصيغة التي تجعل البرنامج قابل للقرائة بشكل واضح .

الصيغة الثانية أو مايسمى بالتمثيل العلمي وهو عبارة عن رقم مثل السابق ولكن مضروب برقم مرفوع إلى العدد 10 هذا الأس يمثله الحرف e أو E أمثلة على هذه الصيغة هي :

12.34e-3 (= 0.01234) 1.0e2 (= 100)

2.3 ثوابت المحارف – Character constants

الثابت المحرفي هو أي حرف كتب بين اثنتين من الفاصلة العلوية الواحدة أو single quotes ؛ مثل 'X' . إن قيمة الثابت المحرفي هي عبارة عن قيمة هذا الحرف في جدول المحارف في النظام أو الجهاز . من أشهر جداول المحارف استخداما هو ASCII (American Standard Code for Information Interchange) . ففي جدول ASCII تبلغ قيمة المحرف 'X' القيمة 88 .

يوجد أيضا العديد من المحارف في لغة السي والتي تسمى بمحارف الهروب أو escape character . وهي تستخدم للتعامل مع المحارف التي من الصعب التعامل معها أو من الصعب ادخالها من لوحة المفاتيح مباشرة . وهي تبدأ بالرمز \ ثم يليه محرف واحد . المحرفين السابقين يعتبرو كمحرف واحد من وجهة نظر المترجم ، سوف أسرد هذه المحارف هنا :

\a	Alarm bell
\n	New line
\t	Horizontal tab
\v	Vertical tab
\b	backspace
\r	Carriage return
\f	Formfeed
\'	Single quote
\"	Double quote
\\	backslash
\?	Question mark
\0	NULL character

الآن على سبيل المثال إذا أردت التعامل مع المحرف \ فيجب أن تكتب '\\' عوضا عن '\' . أيضا بالإضافة إلى محارف الهروب السابقة يمكن أن أن تتكون سلسلة الهروب من المحرف \ يتبعها x ومن ثم واحد أو أكثر من الأرقام الست العشرية . الناتج هو عبارة عن رقم ست عشري يمثل دليل في جدول المحارف في الجهاز على سبيل المثال فإن المحرف 'X' قيمته 0x58 في جدول ASCII وبالتالي يمكن تمثيله أيضا بالصيغة '\x58' . حتى نكون دقيقين أكثر الصيغة الأخيرة تستخدم لطباعة المحارف الموجودة في جدول ASCII التي لا يمكن ادخالها من لوحة المفاتيح كالمحارف اللاتينية وغيرها .

المحرف الأخير الموجودة في القائمة السابقة يعتبر المحرف الأول في جدول ASCII ويسمى بمحرف الصفر أو ال NULL Character '\0' . هذا المحرف له استخدامات خاصة في السي وهذا ماسوف نراه لاحقا في هذا الدرس .

2.4 ثوابت السلاسل النصية – String constants

السلسلة النصية هي مجموعة من المحارف محاطة بزوج من الفاصلتين العلويات أي " . ويمكن أيضا استخدام نفس محارف الهروب السابقة بداخل السلسلة النصية . وكأمثلة على السلاسل النصية هي :

```
"Hello world"
"The C++ Programming Language"
""
"He said \"Good morning\""
"This string terminates with a newline \n"
```

أيضا السلسلة النصية يمكن أن تمتد إلى أكثر من سطر عن طريق استخدام المحرف \ في آخر السطر الأول ومنه الاستمرار في السطر الثاني وكمثال على هذا إليك التالي :

```
"This string extends \
over two lines."
```

المترجم يعتبر السلسلة السابقة كسلسلة واحدة أي يقوم بتجاهل المحرف \ بالاضافة إلى رمز السطر الجديد الذي بعده . وهي مشابهة تماما إلى السلاسلة :

```
"This string extends over two lines."
```

وأيضا السلسلتين المنفصلتين التي تلي أحدهما الأخرى يقوم المترجم باضافتهما إلى بعضهما لتكوين سلسلة واحدة . مثلا هذا المثال :

```
"This "  "string"
```

يحول إلى :

```
"This string"
```

الطريقة السابقة جدا مهمة في قص السلاسل الطويلة ووضعهم على أكثر من سطر.

أيضا الأمر الهام جدا بالنسبة إلى موضوع السلاسل في السي++ فإن أي سلسلة سوف تنتهي ب NULL Character أو '\0' . وهذا المحرف هو الذي يحدد انتهاء السلسلة النصية . هذا يعني إذا كانت السلسلة تحتوي على عدد N من المحارق فإنها حقيقة سوف تحتوي على N+1 من أجل احتواء محرف النهاية . لذلك تسمى السلاسل النصية في لغة السي ب Null terminated string أو السلسلة المنتهية بمحرف الصفر . وإليك هذا الشكل :

h	e	l	l	o	\0

شكل 2 -6

بالاعتماد على سبق يجب أن نلاحظ أن المحرف 'X' والسلسلة "X" ليسا نفس الشئ فالمحرف في معظم الأجهزة عبارة عن 8بت أو بايت أما السلسة "X" فسوف يحجز لها محرفين وبالتالي سوف تكون بايتين . وبالتالي يجب الانتباه في حالة استخدام المحارق والسلاسل والتفريق فيما بينهما .

2.5 المعرفات – Identifiers

لغة السي++ مثل باقي اللغات تتطلب منك اعطاء أسماء معينة للمعطيات التي تستخدمها في برنامجك . هذه الأسماء تدعى بالمعرفات أو Identifiers ويتم وضعها أو تكوينها من قبل المبرمج . ويجب أن يصنع المعرف بالاعتماد على القاعدة التالية :

المعرف هو عبارة عن خليط من المحارف والأرقام والتي يجب أن تبدأ بمحرف . الرمز (_) أو Underscore يمكن استخدامه بالمعرف وهو يعتبر كحرف عادي .

من القاعدة السابقة يمكن أن نضع بعض الأمثلة لمعرفات تعتبر صحيحة لمترجم السي++ :

```
time        counter       BUFFER
x           unit_cost     h2o
_MAX        programName   AVeryLongIdentifier
```

يعتبر أي معرفين هما واحد في حالة كان لهما نفس الاملاء ونفس حالة الأحرف (الصغيرة أم الكبيرة) . وبالتالي المعرفين abcd و abcD لا يعتبرا واحد.

يفضل أن يعطي المبرمج اسماء للمعرفات تدل على استخدامها . على سبيل المثال إذا كان المبرمج يتعامل مع الوقت في برنامجه فيفضل أن يسمي معرفاته ب hours و minutes و seconds وهذا طبعا أفضل بكثير من تسميتها h و m و s .

بعض المعرفات تكون محجوزة لاستخدام لغة السي++ وهي ماتسمى ب language keywords . السرد الكامل لهذه المعرفات معطى في الأسفل ولكن شرحها سوف ياتي لاحقا كل في قسمه .

asm	auto	bad_cast	bad_typeid
bool	break	case	catch
char	class	const	const_cast
continue	default	delete	do
double	dynamic_cast	else	enum
except	explicit	extern	false
finally	float	for	friend
goto	if	inline	int
long	mutable	namespace	new
operator	private	protected	public
register	reinterpret_cast	return	short
signed	sizeof	static	static_cast
struct	switch	template	this
throw	true	try	type_info
typedef	typeid	typename	union
unsigned	using	virtual	void
volatile	while		

بقي أمر هام يجب التنويه له وذلك أنه لا ينصح باستخدام رمزين (_) في بداية المعرف ، مثل MAX__ ، وذلك بسبب أن المعرفات بهذا الشكل محجوزة لاستخدام المكتبات القياسية للسي++ . لذلك تجنب من استخدامها .

2.6 تعريف المتحولات – Variable definition

من اسمها فإن المتحولات هي عبارة عن عناصر معطيات تتغير قيمها أثناء وقت تنفيذ البرنامج . هذه المتحولات يجب أن يكون لها نوع يتم تحديده حتى يستطيع المترجم التعامل معها وتوليد الكود الصحيح . طبعا إن عملية تحديد نوع المتحول تعرف بتعريف المتحولات أو Variable definition . إن التعريف يكون على الهيئة التالية :

```
type-specifier    list-of-variables;
```

إن list-of-variables هي عبارة عن معرفات يفصل بينها علامة الفاصلة هذه المعرفات تمثل المتحولات في برنامجك . كل متحول أو مجموعة من المتحولات يتم تحديد نوعها عن طريق ال type-specifier . إن الأنواع الأساسية في السي++ يمكن اعتبارها :

```
int       (integer)
char      (character)
float     (single precision floating point)
double    (double precision floating point)
```

وكمثال على تعريف المتحولات :

```
int       hours, minutes, seconds;
```

هنا المتحولات hours و minutes و seconds كلها عبارة عن متحولات لأعداد صحيحة integer وبالتالي يمكن تخزين أي عدد integer ضمنها واستخدام هذا العدد لاحقا في برنامجك .

طبعا يمكن أن يتم تعريف مجموعة من المتحولات ذات أنواع مختلفة وهذا طبعا يتم باستخدام الفاصلة المنقوطة كفاصل . ويفضل كتابة كل تعريف بسطر منفصل من أجل زيادة الوضوح . وهذا مثال على ذلك :

```
int       day, month, year;
float     centigrade, fahrenheit;
char      initial;
double    epsilon;
```

وعند التعريف يفضل تعريف المتحولات ذات العلاقة فيما بينها في مجموعة والباقي في مجموعة أخرى كالتالي :

```
int       counter, value;
int       day, month, year;
```

وأيضا يمكن كتابة التعريف على عدة أسطر كالمثال التالي :

```
int     day, month, year,
        hours, minutes, seconds;
```

أيضا يمكن اعطاء قيم معينة للمتحولات أثناء تعريفها وهذا يتم عن طريق استخدام رمز الالحاق (=) كالتالي :

```
int     sum = 0;
float   pi = 3.14;
```

المتحولات السابقة تبقى محتفظة بقيمتها حتى يتم تغييرها في البرنامج . أما المتحولات التي يتم تعريفها من غير اعطائها قيمة معينة فقيمتها غير معروفة ولا ينصح باستخدامها حتى يتم اعطائها قيمة معينة في البرنامج .

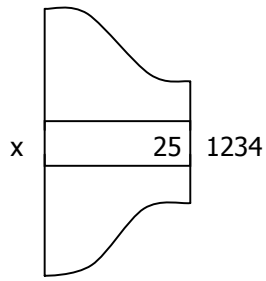
بصراحة فإن هذا العرض البسيط لتعريف المتحولات غير كافي وليس من المعقول اعطاء كل النواحي في تعريف المتحولات في الدروس الأولى لذلك سوف نتوسع في ذلك لاحقا . يكفي الآن أن تعرف أن كل متحول له الخواص التالية :

- إنها من نوع ما .
- تملك عنوان معين في الذاكرة .

كل متحول يجب أن يكون متواجد في الذاكرة في عنوان معين . ويقوم المترجم بالرجوع إلى هذا العنوان في حال وجد المتحول في السياق . على سبيل المثال اعتبر التعريف التالي :

```
int     x = 25;
```

على مستوى فهمنا فإنه يمكن القول أن تم حجز مساحة في الذاكرة من أجل استيعاب أرقام integer وهذه المساحة يتم التعامل معها عن طريق المعرف x . بالنسبة إلى المستوى المنخفض للحاسب فإن هذه المساحة تملك العنوان (على سبيل المثال 1234) . انظر الشكل 2-2 .



شكل 2 -7

2.7 الواصفات – Qualifiers

الواصفات في لغة السي++ التي سوف نشرحها في هذه الفقرة هي :

- long و short .
- signed و unsigned .
- const .

الواصفين long و short يستخدان قبل النوع int من أجل التعديل في طول المتحول . إن استخدام long يعطي المتحول أكبر مجال ممكن للأعداد التي يمكن استخدامها . وكمثال على استخدام long :

```
long int        memory_address;
```

في معظم الأجهزة القديمة كانت ال int تأخذ مامقداره 16بت وال long مامقداره 32بت . أما حاليا تحت الوندوز وباستخدام الأجهزة الحديثة فإن ال int وال long كلاهما تأخذ مساحة مقدارها 32بت . أي لا يوجد أي اختلاف بين استخدام int أو long . ولكن من أجل التوافقية بين المترجمات المختلفة يجب أن تعلم أن int تتعلق بالحاسب المستخدم والمترجم المستخدم أما long فهي دوما تحجز أكبر مساحة ممكنة من أجل التعامل مع الأعداد وقد تكون 64بت في على معالجات 64بت الجديدة .

أما الواصف short الذي يوضع قبل int . فيدل على أن المتحول سوف يستخدم من أجل مجال صغير للأعداد . وعلى الأغلب فإن short تقوم بحجز 16بت للمتحول . ويكمن استخدامها في حالة أردنا التقليل من استهلاك الذاكرة . ومثال على تعريف المتحولا باستخدام الواصف short :

```
short int       day_of_week;
```

في المثالين السابقين يمكن تجاهل النوع int ، أي يمكن التعريف كالتالي :

```
long    memory_address;
short   day_of_week;
```

أيضا استخدام long قبل double يعطي المتحول مجال أكبر من المجال المستخدم في double .

إن جميع ماذكرناه من متحولات كانت متحولات مؤشرة أي يمكن أن تحمل قيم موجبة وسالبة . أما في حالة أردنا أن يكون مجال الأعداد المستخدمة هي فقط موجبة فيجب أن نسبق int أو char بالواصف unsigned . هذا يعني أن المتحول في هذه الحالة يعتبر موجب فقط ضمن المجال 0 إلى 65535 في حالة int ذات 16بت و من 0 إلى 255 في حالة char . وكمثال على استخدام هذا الواصف :

```
unsigned int    natural;
unsigned        record_number;
unsigned char   i_am_byte;
```

إن استخدام int في المثال السابق كان اختياري ويمكن تجاهلها . والواصف signed تعني أن المتحولات هي مؤشرة ولكن كما ذكرنا فإنه بشكل افتراضي المتحولات حين حجزها تكون مؤشرة فإن استخدام signed هو تحصيل حاصل .

أخيرا ، إن الواصف const يمكن استخدامه مع تعريف أي متحول وذلك لتحديد أن قيمة هذا المتحول لن تتغير أبدا أثناء تنفيذ البرنامج . مثل هذه المتحولات لا يمكن استخدامها على الطرف الأيسر لعلامة الالحاق (=) أو لا يمكن استخدامها في أي شكل من الأشكال يؤدي إلى تغيير قيمتها . المترجم في هذه الحالة يستطيع أن يضع هذا المتحول في ذاكرة قابلة للقرائة فقط أو يقوم بالتعديل على البرنامج بالشكل الذي يريد (Optimization) . طبعا في حالة استخدمنا الواصف const يجب علينا اعطاء قيمة للمتحول وقت تعريفه كالتالي :

```
const double    pi = 3.1415926;
const int       student_number = 10;
```

من غير أي عملية تهيئة فإن هذا يعتبر خطأ أن تكتب :

```
const double    pi;
```

هنا لم نعطي المتحول pi أي قيمة وبما أنه لا يمكن اعطاء pi أي قيمة أثناء تنفيذ البرنامج فإن pi تبقى ذات قيمة غير معرفة وبالتالي فإن مثل هذا النوع من التعريفات يعطينا خطأ أثناء الترجمة .

بما أنه انتهينا من ذكر أنواع المتحولات وجميع الواصفات فإن الجدول التالي (جدول 2-1) يذكر جميع الأنواع الأساسية التي يمكن استخدامها في السي++ (وأخص بالذكر فيجوال سي++) . (الجدول مأخوذ من MSDN) .

Type Name	Bytes	Other Names	Range of Values
int	*	signed, signed int	System dependent
unsigned int	*	unsigned	System dependent
__int8	1	char, signed char	–128 to 127
__int16	2	short, short int, signed short int	–32,768 to 32,767
__int32	4	signed, signed int	–2,147,483,648 to 2,147,483,647
__int64	8	none	–9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
char	1	signed char	–128 to 127
unsigned char	1	none	0 to 255
short	2	short int, signed short int	–32,768 to 32,767
unsigned short	2	unsigned short int	0 to 65,535
long	4	long int, signed long int	–2,147,483,648 to 2,147,483,647
unsigned long	4	unsigned long int	0 to 4,294,967,295
float	4	none	3.4E +/- 38 (7 digits)
double	8	none	1.7E +/- 308 (15 digits)
long double	10	none	1.2E +/- 4932 (19 digits)

جدول 2 -1

- في أنظمة وندوز 95 ومابعدها (أي الأنظمة المعتمدة على 32بت) فإن int سوف تكون 4 بايتات أي أنها مثل ال long تماما . ولاحظ في الجدول السابق فإن الأنواع المسبوقة ب علامتين (_) هي اضافة من الفيجوال سي++ أي أن استخدامها يعني أن برنامجك سوف يترجم فقط في بيئة الفيجوال سي++ . لذلك لا ينصح باستخدامها بكثرة وبالتالي يفقد برنامجك خاصية التوافق مع باقي المترجمات . الشئ الوحيد الذي قد يفيدك هو __int64 التي تعطيك مجال ضخم جدا للاعداد ولكن يجب أن تنتبه على أن استخدام __int64 هو أبطأ من استخدام long أو int وخاصة في حساب التعابير الرياضية .

2.8 التعابير الحسابية – Arithmetic expressions

إن لغة السي++ تدعم العمليات الحسابية العادية مثل الجمع (+) و الطرح (-) و الضرب (*) والقسمة (/) وأيضا تدعم عملية باقي القسمة للأعداد الصحيحة وهي (%) . العمليات السابقة كلها تطبق على حدين أي operands . المعاملات السابقة تسمى ثنائية أو binary لأنها تطبق على حدين . يوجد معاملات أخرى تطبق على حد واحد وهي في هذه الحالة معامل الاشارة السالبة (-) ومعامل الاشارة الموجبة (+) ، الأخيرتان تسمى معاملات أحادية أو unary وذلك بسبب أنها تطبق على حد واحد . وكأمثلة على هذا :

```
12 * length
-1024
3.14 * radius * radius
distance / time
money % 100
```

كما تلاحظ في المثال الأول فإن القيمة 12 ضربت مع ما يحتوي عليه المتحول length . أما في المثال الثاني فإن معامل الاشارة السالبة قام بعكس قيمة العدد 1024 الموجبة إلى سالبة .

إن التعبير الحسابي في لغة سي++ يتم حسابه على حسب أولوية العوامل المستخدمة في التعبير . إن أولوية العوامل هي التي تحدد ترتيب حساب العمليات الرياضية في التعبير . الجدول 2-2 يوضح المعاملا الحسابية حسب أولويتها . طبعا الجدول الكامل للأولويات سوف يأتي لاحقا في دروس قادمة .

Operator	Description	Associativity
+ -	Unary plus Unary minus	Right to left Right to left
* / %	Multiplication Division Modules	Left to right Left to right Left to right
+ -	Addition Subtraction	Left to right Left to right

جدول 2 -2

من الجدول السابق تلاحظ أن معاملات الاشارة لها أعلى أولية ومعاملات الجمع والطرح لها أدنى الأولويات . والضرب والقسمة وباقي القسمة هي في المنتصف . وبالتالي فإن أي تعبير يحتوي على خليط من المعاملات السابقة فإن الذي ينفذ أولا هو جميع معاملات الاشارة إن وجدت وبعدها على الترتيب حسب الجدول السابق . وكمثال على هذا إليك بالتعبير التالي :

```
2 + 3 * 4
```

التعبير السابق قيمته تساوي 14 لأنه أولا يتم تنفيذ معامل الضرب على 3 و 4 الذي نتيجته 12 ومنه الجمع على 2 و 12 الذي ينتج 14 .

المعاملات في نفس المجموعة عند وجودها في التعبير فإنه يتم تنفيذها على حسب ورودها . في المجموعة الأولى بتم التنفيذ من اليمين إلى اليسار ، أما الثانية والثالثة فيتم تنفيذها من اليسار إلى اليمين . وكأمثلة على هذا.

```
-+-10          (right to left)  ➔  (-(+(-10)))
10 * 2 / 5 * 4 (left to right)  ➔  (((10 * 2) / 5) * 4)
2 – 3 + 5      (left to right)  ➔  ((2 – 3) + 5)
```

المثال الأول لاحظ أن التنفيذ يتم من اليمين إلى اليسار وبالتالي يتم تنفيذ اشارة الناقص الأقرب إلى العدد أولا لترجع القيمة 10- ومنه اشارة الموجب لترجع 10- وأخيار السالب لترجع 10 . المثال الثاني يتم تنفيذ أول عملية ضرب على 10 و 2 لترجع 20 وبعدها القسمة على 20 و 5 لترجع 4 وبعدها الضرب على 4 و 4 لترجع 16 . المثال الأخير يتم تنفيذ أول عملية طرح لترجع 1- وبعدها الجمع على 1- و 5 لتكون النتيجة 4 . وأضع أيضا المثال التالي :

```
2 + 3 * 4 + 5
```

هنا يتم أولا حساب معامل الضرب لأن له أولوية أعلى من الجمع . وبالتالي التعبير يصبح 5 + 12 + 2 الآن يتم حساب الجمع من اليسار إلى اليمين بدأ من 2 و 12 ليصبح التعبير 5 + 14 وأخيرا تكون نتيجة التعبير 19 .

بالنسبة إلى عملية القسمة (/) فإن وظيفتها هي القسمة طبعا ولكن يجب الانتباه أنه عند قسمة عددين صحيحين integer فإن الجواب يكون عدد صحيح . أما في حالة كان أحد الحدين على الأقل عدد حقيقي float فإن الجواب يكون عدد حقيقي . وإليك بالأمثلة التالية :

13 / 5	تساوي	2
13.0 / 5	تساوي	2.6
13 / 5.0	تساوي	2.6

يعني في حالة القسمة الصحيحة فإن العدد الناتج هو القسم الصحيح من العدد ويزال مابعد الفاصلة . وكمثال على باقي القسمة لدينا :

13 % 5	تساوي	3
15 % 5	تساوي	0
-13 % 5	تساوي	-3
13 % -5	تسواي	3

في حالة أحد الحدين كان سالبا فإن اشارة باقي القسمة تكون مشابهة لأشارة المقسوم عليه كما في المثال السابق . وحتى نكون دقيقين أكثر فإن عملية باقي القسمة دائما تساوي إلى التعبير التالي :

a % b == a – (a / b) * b

حيث a و b هي أعداد صحيحة وقيمة b لاتساوي الصفر ، أي أن القسمة المستخدمة هي قسمة صحيحة .

- قمت بالاستفاضة في الشرح على هذه المعاملات السهلة لأنه في السي++ وكما سوف نرى لاحقا في قسم البرمجة غرضية التوجه Object Oriented Programming أنه يمكن اعادة برمجة العوامل من أجل اعطائها خواص جديدة .

2.9 التحويل بين الأنواع – Type conversions

في هذا القسم سوف نحاول الاجابة على السؤال التالي : ماذا يحدث إذا أردنا تنفيذ تعبير حسابي يتركب من حدود ذات أنواع مختلفة ؟

في البدية يجب أن أنوه على أن معظم العمليات الحسابية تتم على نوع int وذلك بسبب أن int دائما تكون بحجم مسجل المعالج الداخلي وبالتالي تكون العمليات الحسابية عليها أسرع مايمكن . علي سبيل المثال الأرقام الثابتة أو المحارف يتم تحويلها إلى int قبل القيام بالعمليات الحسابية عليها وذلك في حالة كانة ال int يمكن أن تحوي الرقم وإلا يتم استخدام unsigned int . طبعا هذا في حالة لم نقم باجبار الثابت أو الرقم على أن يكون long وذلك بذكر اللاحقة L بعد الرقم وعندها يجب أن نقوم بتحويلات أخرى على الأرقام أو المتحولات في التعبير .

القاعدة هنا بشكل عام أنه يتم تحويل الحدود من الأنواع الصغيرة إلى الأنواع الكبيرة . ويتم هذا على حسب الترتيب التالي :

- إذا كان أحد الحدين long double ، يتم تحويل الآخر إلى long double .
- إذا كان أحد الحدين double ، يتم تحويل الآخر إلى double .
- إذا كان أحد الحدين float ، يتم تحويل الآخر إلى float .
- نطبق عملية تحويل الintالسابقة .
- إذا كان أحد الحدين unsigned long int ، يتم تحويل الآخر إلى unsigned long int . أما إذا كان أحد الحدود long int ، والآخر unsigned int وكانة ال long int يمكن أن تحوي جميع القيم الممكنة في unsigned int فإنه يتم تحويل الأخيرة إلى long int ؛ وإلا يتم تحويل الاثنين إلى unsigned long int .
- إذا كان أحد الحدين long int ، يتم تحويل الآخر إلى long int .
- إذا كان أحد الحدين unsigned int ، يتم تحويل الآخر إلى unsigned int .

العملية تبدو معقدة عند التعامل مع الأعداد الغير مؤشرة أو unsigned . على سبيل المثال اعتبر التصريح التالي:

```
unsigned int ui  =  10;
         int si  =  -7;
```

واعتبر التعبير التالي : ui + si . بالاعتماد على القواعد السابقة فإن الحد الثاني سوف يحول إلى unsigned int . على فرض أن int هنا هي 16بت . فإن مجال القيم الممكن تمثيلها باستخدام unsigned int هو من 0 إلى 65535 . القيمة 7- بالنظام الست عشري تساوي 0xFFF9 وبالتالي إذا اعتبرنا القيمة السابقة كعدد غير مؤشر

(unsigned int) فإننا نحصل على القيمة 65529 . وإذا جمعناها مع القيمة 10 نحصل على 65539 أو 0x10003 بالنظام الست عشري وبالتالي يكون الجواب 3 لأننا نتعامل مع أعداد 16بت .

في الأنظمة الحديثة نجد أن int هو 32بت وأيضا long هو 32بت . فعملية التحويل تصبح أسهل . وأيضا تجنب استخدام الأعداد الغير مؤشرة قدر المستطاع وكن حذرا في حالة تم استخدامها .

2.10 معامل الالحاق – The assignment operator

معامل الالحاق يسمح بالحاق قيمة تعبير ما إلى متحول في البرنامج . وأبسط صيغة إلى معامل الالحاق هي :

```
variable = expression;
```

عند تنفيذ عملية الالحاق فإنه يتم أولا حساب قيمة التعبير وبعدها يتم اعطاء المتحول القيمة الجديدة للتعبير . أيضا يتم تحويل نوع القيمة حتى توافق نوع المتحول . إن معامل الالحاق مثله مثل باقي المعاملات له أولوية وترتيب في الحساب ، طبعا أولويته هي أصغر مايمكن ويتم حسابه من اليمين إلى اليسار كما في معامل الاشارة السالبة والموجبة . أوليته صغرى هذا يضمن تنفيذ التعبير قبل عملية الالحاق .

وكما أن التعبير a + b يرجع قيمة جديدة فأيضا التعبير a = b يرجع قيمة جديدة وذلك بعد حساب قيمة التعبير واسنادها إلى المتحول ، هذه القيمة التي يرجعها تعبير الالحاق تساوي قيمة المتحول بعد الاسناد . هذا مفيد جدا في بناء سلسلة الاسنادات مثل المثال التالي :

```
variable1 = variable2 = variable3 = … variablek = expression;
```

كما ذكرنا فإن تنفيذ معامل الالحاق يتم من اليمين إلى اليسار فإنه أولا يتم اسناد قيمة التعبير expression إلى المتحول variablek وبعدها قيمة هذا المتحول تسند إلى variable(k-1) وهكذا حتى يتم اسنادها إلى variable2 ومنه إلى variable1. ومنه فإن التعبير التالي :

```
X = y = z = p + q;
```

يتم تفسيره على الشكل التالي :

```
x = (y = (z = p + q));
```

كما ذكرنا فإن عملية التحويل تتم عبر معامل الالحاق فمثلا إذا أردنا الحاق ماهو من نوع int إلى double فسوف يتم تحويل ال int إلى ال double أما العكس أي من double إلى int فطبعا هذه العملية نسميها عملية مدمرة لأن الرقم سوف يقرب إلى أقرب عدد صحيح وقد يفقد من قيمته . على سبيل المثال انظر ماذا سوف يحدث في التعابير التالي :

int = int	تتم من غير تحويل
float = double	قص القيمة مع التقريب
int = long	قص القيمة إذا كانت int هي 16بت
char = int	قص القيمة دوما .

أيضا يجب الحرص عندما يتم استخدام سلسلة الالحاقات مع أنواع مختلفة من المتحولات . إذا كانت iii هي من نوع int فإن التعبير التالي :

```
iii = 12.34;
```

يضع ويرجع القيمة 12 وبالتالي ففي التعبير التالي حيث fff هي من نوع float .

```
fff = iii = 12.34;
```

فإن القيمة 12 من نوع int سوف تحول إلى float أي القيمة 12.0 سوف يتم اسنادها إلى المتحول fff .

2.11 معاملات الالحاق المركبة – The compound assignment operator

إن عملية الالحاق من الشكل

```
count = count + 2
```

تتكرر بشكل كبير في برامجنا وهي تعني أنه يتم اضافة مامقداره 2 إلى المتحول count وبعدها وضع القيمة الجديدة في المتحول count . أي بكلمة أخرى يتم اضافة اثنان إلى المتحول count . مثل الالحاق السابق يمكن كتابته في السي++ بالصيغة التالي :

```
count += 2
```

في الواقع فإن يمكن تطبيق نفس الأمر على باقي المعاملات الحسابية الأربعة وبالتالي يمكن أن نكتب مايلي :

```
count += 2
count -= 1
power *= 2.71828
divisor /= 10.0
remainder %= 10
```

في كل الحالات السابقة الحدين على طرفين المعامل يمكن أن يكونا من أي نوع ماعدا في الحالة الأخيرة حيث كما نعرف فإن معامل باقي القسمة يحتاج إلى نوع صحيح على طرفيه . أيضا فإن الأولية وطريقة التنفيذ هي تماما مثا معامل الالحاق الذي شرحناه في الفقرة السابقة .

أيضا يجب أن ننتبه إلى أن :

variable op= expression

تفسر كالتالي :

Variable = variable op (expression)

وبالتالي فإن المثال التالي :sum /= 3 + 7 يتم حسابه كالتالي : sum = sum / (3 + 7) .

أيضا يمكن استخدام سلسلة الالحاقات هنا أيضا ولكن مرة أخرى مع الحذر في بعض الحالات لاحظ المثال التالي:

```
fff = iii *= fff
```

إذا كانت fff من نوع float وتحتوي على القيمة 1.234 والمتحول iii من نوع int ويحتوي على القيمة 12 فإن التعبير السابق يتم حسابه كالتالي :

```
fff = (iii = iii * fff)
```

وبالتالي iii سوف يتم اسناد القيمة (14.808 = 1.234 * 12) 14 بعد ازالة القسم بعد الفاصلة .وبعدها يتم اسناد القيمة 14.0 للمتحول fff .

2.12 معاملا الزيادة والنقصان – The increment and decrement operators

في القسم السابق شرحنا معامل الاسناد المركب . ويمكن أن نستخدم معامل الاسناد المركب كالتالي من أجل اضافة القيمة 1 إلى المتحول :

```
x += 1
```

في الحقيقة فإن الزيادة بمقدار واحد هي من العمليات التي تمر علينا كثرة في برامجنا فلا يخلو برنامج منها . وأيضا هي من العمليات التي يتم تنفيذها بسرعة لأنها تكافئ إلى تعليمة واحده على الأغلب للمعالج . هذه العملية تسمى بمعامل الاضافة البعدية أو postincrement operator و معامل الاضافة القبلية أو preincrement operator . الثانية لها الصيغة :

```
++ variable
```

أما التعبير ذو الاضافة البعدية فهو على الصيغة :

```
variable ++
```

في كلا الحالتين السابقتين فإنه يتم الزيادة بمقدار واحد إلى المتحول variable . وأيضا كلا الحالتين السابقتين فإنها يرجعا قيمة . في الأولى (preincrement) فإنه يتم ارجاع القيمة الجديدة بعد الزيادة . أما الثانية (postincrement) فإنه يتم ارجاع قيمة المتحول قبل الزيادة . لو اعتبرنا المثال التالي :

```
sum += ++count
```

فإن المتحول count سوف يزاد بقيمة 1 ومن ثم يتم استخدام القيمة الجديدة في التعبير أي يتم اضافتها مع sum وتخزين النتيجة في sum . ولو اعتبرنا المثال التالي :

```
sum += count++
```

يتم اضافة قيمة المتحول count أيضا بمقدار 1 ولكن يتم استخدام القيمة السابقة لcount في التعبير . على سبيل المثال إذا كانت قيمة sum و count هي 10 و 20 على الترتيب فإن count سوف تصبح 21 ويتم جمع القيمة 20 (القديمة) مع 10 من أجل أن نحصل على 30 وتخزينها في sum .

في حالة استخدام معامل الانقاص فاستخدامه تماما مثل معامل الزيادة ولكن يتم هنا استخدام (--) عوضا عن (++) ويتم الانقاص بمقدار 1 .

```
-- variable

variable --
```

بالنسبة إلى أولوية وكيفية تنفيذ هذين المعاملين فسوف تجدهل في جدول في آخر الدرس يجمع جميع المعاملات التي تم شرحها في هذا الدرس .

2.13 معامل التحويل – The type cast operator

جميع المعاملات السابقة كانة عبارة عن رموز ، أما هنا فالأمر مختلف ، فالمعامل هنا ليس ثابت وهو عبارة عن كلمة . تحدثنا سابقا عن عملية التحويل الضمنية التي تتم في التعابير الحسابية وخلال معامل الاسناد . أما عملية التحويل التي نقوم بها نحن فتسمى cast . إن الطريقة القياسية في عملية تحويل نوع إلى آخر هو عن طريق استخدام اسم النوع الذي نريد التحويل إليه ومنه نذكر اسم التابع أو القيمة بين قوسين كالتالي :

```
double(date)
```

هنا إذا كان المتحول date من نوع int فيتم تحويله إلى double . إن معامل التحويل تماما مثل باقي المعاملات أي له أولوية وكيفية في التنفيذ وهذا ماسوف تراه في آخر الدرس .

الطريقة السابقة في كتابة نوع التحويل هي تشابه طريقة استدعاء التوابع كما سوف نرى في دروس لاحقة . أما الطريقة الأخرى في كتابة التحويل هي التالية :

```
(type-specifier) expression
```

أي يتم كتابة النوع داخل أقواس ثم يليها التعبير المراد تحويله . هذه الطريقة مأخوذة من لغة السي ومازال يمكن استخدامها هنا من أجل التوافقية . وكأمثلة على هذه لدينا :

```
(double) date
(char) x
(int) d1 + d2
(int) (d1 + d2)
```

في المثال قبل الأخير سوف يتم تحويل d1 وجمع القيمة الجديدة مع d2 . أنا في المثال الأخير فسوف يتم جمع d1 و d2 ومن ثم القيام بعملية التحويل . هذا بسبب أن معاملات التحويل لها أولوية أعلى من الجمع وهذا ما سوف تراه في جدول الأولويات في آخر الدرس .

2.14 معامل الفاصلة – The comma operator

آخر المعاملات التي سوف نشرحها في هذا الدرس هو معامل الفاصلة التي تجد لها استخدامات في شتى التطبيقات . سوف أقوم بشرح سريع عليها على أن يأتي مزيد من الشرح والتطبيقات عليها لاحقا في دروس أخرى . إن معامل الفاصلة يتكون من تعبيرين يفصل بينها فاصلة كالتالي :

```
expression1 , expression2
```

بين كل المعاملات السابقة فإن معامل الفاصلة لديه الأولوية الدنيا حتى أقل من معامل الاسناد ويتم تنفيذها من اليسار إلى اليمين . وبالتالي في المثال السابق فإن expression1 يتم حسابها بشكل كامل ومن ثم expression2 . إن قيمة التعبير السابق تعتبر هي قيمة التعبير الثاني أي expression2 . مثال على هذا :

```
s = ( t = 2, t + 3 )
```

أولا يتم تنفيذ التعبير t = 2 وبعدها التعبير t + 3 الذي تكون نتيجته 5 ونتيجة تعبير الفاصلة السابق يكون 5 وبالتالي يتم اسناد 5 إلى المتحول s . طبعا لاحظ أهمية الأقواس في المثال السابق وإلا لاحظ المثال التالي :

```
s = t = 2, t + 3
```

هنا يتم اسناد القيمة 2 إلى كلا المتحولين s و t ومن ثم يتم تنفيذ التعبير t+3 الذي تكون قيمته 5 ولكن هذه القيمة التي تكون قيمة معامل الفاصلة يتم تدميرها لأنها غير مستخدمة .

أولويات المعاملات المذكورة في هذا الدرس :

(الجدول التالي مأخوذ من MSDN)

Symbol	Name or Meaning	Associativity
	Highest Precedence	
++ --	Post-increment Post-decrement	Left to right
++ --	Pre-increment Pre-decrement	Right to left
- +	Unary minus Unary plus	Right to left
(type)	Type cast [for example, (float) i]	Right to left
* / %	Multiply Divide Remainder	Left to right
+ -	Add Subtract	Left to right
=	Assignment	Right to left
*=, /=, %=, +=, -=	Compound assignment	Right to left
,	Comma	Left to right
	Lowest Precedence	

مبادئ الادخال والاخراج – Simple Input and Output

من أهم الأمور في البرامج الحقيقة هي أن يستطيع البرنامج التعامل مع الوسط المحيط به . في هذا الدرس سوف أقدم قسم صغير من مكتبة سي++ القياسية في الادخال والاخراج . لن أقوم بشرح كل التعليمات والعمليات الموجودة في هذه المكتبة وسوف أترك هذا إلى دروس لاحقة . الآن سوف أقوم بشرح العمليات اللازمة للتعامل مع لوحة المفاتيح والشاشة فقط . هذا سوف يعطينا خطوة أولية في بناء برمجيات تتعامل مع الأجهزة الطرفية السابقة. في دروس قادمة باذن الله سوف نكمل عمليات الادخال والاخراج حتى تضم التعامل مع الملفات على القرص أو العمليات على السلاسل .

3.1 الوصول إلى مكتبة سي++ القياسية – Access to the standard C++ library

في أي برنامج سي++ يتم تنفيذه فإن أربع متحولات يتم حجزها هذه المتحولات هي عبارة عن مسارات (streams) للمعلومات من وإلى الأجهزة الطرفية . اثنان من هذه المسارات يسميان ب standard input و standard output . لماذا سمينا المتحولات السابقة مسارات هذا بسبب أنها تمثل كمسار للبيانات بين البرنامج ومصدر البيانات أو وجهتها . المصدر أو الوجهة قد تكونا أحد الملفات أو حتى الذاكرة . ويمكن أيضا تحويل هذا المسار في أي وقت كان من مصدر إلى آخر أو من وجهة إلى أخرى . الافتراضي في السي++ أن يكونا المسارين الافتراضيين هما من الكيبورد (standard input) أو إلى الشاشة (standard output) . ولكن على حسب ضوء البرنامج فيمكن تحويل هذه المسارات لأن يكون الدخل الافتراضي من ملف معين ؛ وهذا ما نلاحظه إذا فكرنا في برمجة مترجم فالبديهي أن يكون مسار الدخل الافتراضي من الملف بدلا من الكيبورد .

جميع ملفات برامجك التي تستخدم مكتبة الدخل والخرج يجب أن تحتوي على هذا السطر في بداية الملف :

```
#include <iostream.h>
```

سوف نقوم بشرح عبارة #include في الدرس القادم . الآن أريدك أن تتقبل أن ملف التعريف iostream.h (header file) يحتوي على معلومات وتعريفات ضرورية من أجل استخدام مكتبة الدخل والخرج وهو الذي يقوم بتعريف المسارات التي تحدثنا عنها سابقا . هذا المعلومات جميعها سوف تضمن في الملف الحالي لأننا كتبنا عبارة #include في أعلى الملف .

3.2 الاخراج غير المنسق – Unformatted output

عمليات الاخراج تتم عن طريق المعامل >> وهي تدعى بمعامل الاضافة أو insertion operator . اسمه معامل الاضافة لأنه يستخدم في اضافة البيانات إلى ال stream أو المسار الذي يؤدي في النهاية إلى ظهور البيانات على الشاشة بشكل افتراضي . باستخدام المتحول cout الذي هو (standard output) المعرف في iostream.h و المعامل السابق يمكن تحويل البيانات إلى الشاشة بشكل افتراضي أو إلى أي stream آخر ، كما في المثال التالي :

```
cout << x;
```

إن معامل الاضافة تماما مثل بقية معاملات السي++ له أولوية وتكون دنيا ويتم تنفيذ أو تجميع التعبير من اليسار إلى اليمين. وبالتالي بما أن أولويته دنيا فإن العبارة التالية سوف تنفذ المطلوب باخراج ناتج جمع المتحولين x و y :

```
cout << x + y;
```

إن معامل الاضافة >> يمكن استخدامه أكثر من مرة تماما كما في سلسلة الاسنادات لأن هذا المعامل يرجع قيمة تشير إلى مسار الخرج المستخدم . وبما أن تجميع أو طريقة تنفيذ التعبير تتم من اليسار إلى اليمين فيمكن استخدامه في عملية اخراج أنواع مختلفة من البيانات إلى الشاشة كالتالي :

```
cout << "The sum of " << x << " and " << y << " is " << x + y;
```

أولا يتم اخراج الجملة "The sum of " إلى ال stream ونتيجة المعامل هي عبارة عن مؤشر إلى ال stream المستخدم هنا وهو cout . هذه الأخيرة سوف تستخدم مرة أخرى من أجل اخراج المتحول x وهكذا ..

3.3 الادخال غير المنسق – Unformatted input

عمليات الادخال تتم عن طريق المعامل >> وهي تدعى بمعامل الاستخراج أو extraction operator . اسمه معامل الاستخراج لأنه يستخدم في استخراج البيانات من ال stream أو المسار إلى متحول في البرنامج . باستخدام المتحول cin الذي هو (standard input) المعرف مسبقا في iostream.h جنبا إلى جنب مع المعامل السابق يمكن استخلاص البيانات من لوحة المفاتيح بشكل افتراضي أو من أي stream آخر ، كما في المثال التالي :

```
cin >> x;
```

إن معامل الاستخراج أولويته تماما مثل معامل الاضافة وأيضا بالنسبة إلى كيفية تنفيذه أو تجميعه . أيضا هذا المعامل يرجع قيمة تشير إلى مسار الدخل المستخدم وبالتالي يمكن لهذا المعامل أن يتم استخدامه كسلسلة من الادخالات كالتالي :

```
cout << "Please enter three numbers : ";
cin >> first >> second >> third;
```

إن هذا المعامل متوافق مع جميع الأنواع الأساسية في سي++ . يقوم هذا المعامل باهمال أي مسافات أو محارف ال (tab) أو محارف (new line) حتى يتم قرائة المحارف المتوافقة مع النوع الذي نريد قرائته . على سبيل المثال إذا كانت first و second هما من نوع int أما third فهي من نوع سلسلة محارف . وكان الدخل على الصورة التالية :

```
167     34hello
```

فإن سوف يتم استخراج القيمة 167 ووضعها في first وبعدها يتم اهمال المسافات التي بعدها حتى يتم قرائة القيمة 34 ووضعها في المتحول second . لاحظ أنه لم يتم قرائة hello مع أنها موصولة مع القيمة 34 لأنها من نوع آخر . وأخيرا يتم قرائة hello ووضعها في المتحول third .

3.4 تنسيق الخرج – Manipulating the output

إن مكتبة الدخل والخرج تحتوي على عدة طرق من أجل تنسيق الخرج إلى ال stream من داخل البرنامج . سوف نشرح هنا كيفية استخدام المعدلات أو Manipulators . سوف أكمل شرح هذه المعدلات وطرق أخرى في دروس قادمة باذن الله . أما هنا فسوف أقدم طريقة سهلة لتنسيق الخرج . وهذا المنسق يظهر في عبارة الخرج تماما مثل بقية الحدود في الخرج ولكنه لا يخرج أي شئ إلى الstream بل له تأثير على الكمية التي تخرج بعده . مثلا المثال التالي :

```
int index = 10;
cout << "[" << index << "]";
```

سوف ينتج لنا الخرج التالي :

```
[10]
```

على فرض أننا نريد طباعة index في مساحة تكفي لجميع القيم الممكنة لها فإننا نستخدم التالي :

```
cout << "[" << setw(4) << index << "]";
```

سوف نحصل على :

```
[  10]
```

حيث تقوم setw باخبار أن تظهر القيمة التي بعدها في مساحة مكونة من أربع محارف . أما إن كانت القيمة index تحتاج إلى أكثر من ذلك فطبعا سوف يتم طباعة الرقم كامل .

بالنسبة إلى طباعة الأرقام الحقيقة فإنه بشكل افتراضي يتم طباعة ستة أرقام بعد الفاصلة ويمكن تغيير هذا باستخدام المعدل setprecision كالتالي :

```
double pi = 3.1415926;
cout << setprecision(4) << pi;
```

سوف تحدد أننا مهتمين فقط بأربع أرقام كمجموع كلي وبالتالي سوف نحصل على الخرج التالي :

```
3.142
```

لاحظ كيف أنه تم تقريب العدد قبل الاخراج . يجب أن ننتبه إلى أمر هام وهو من أجل استخدام هذه المعدلات أو غيرها يجب تضمين الملف iomanip.h في البرنامج :

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

3.5 التابعين get و put – The functions get and put

أحد الميزات السهلة التي لاغنى عنها أحيانا وهي كيفية اخراج أو ادخال محرف وحيد . بالنسبة للاخراج فهذا يتم عن طريق التابع put وهذا مثال على هذا :

```
char  ch = 'h';
cout.put(ch);
```

إن التابع السابق put يأخذ متحول واحد ويطبعه في cout أو standard output stream . إن كيفية الكتابة السابقة سوف تتعرف عليها أكثر عندما نبدأ في دروس عن ال classes لاحقا .

التابع get يقرأ المحرف التالي من standard input stream ويضعه في متحول ما كالتالي :

```
char ch;
cin.get(ch);
```

3.6 استخدام مكتبة الدخل والخرج الجديدة – Using the new standard I/O stream library

لا يوجد لغة من اللغات تتطور أو يضاف لها توابع داعمة جديدة ، وكذلك المكتبة القياسية في الادخال والاخراج في السي++ . ومن أجل استخدام آخر النسخ من هذه المكتبة في برنامجك فيجب أن يتم التضمين بالطريقة التالية :

```
#include <iostream>
using namespace std;
```

لاحظ أننا لم نستخدم الامتداد h للملف iostream وهذا مايخبر المترجم أننا نريد استخدام النسخة الحديثة من المكتبة . أما السطر الثاني فهو ضروري وسوف نتقبله حاليا مثل ماهو حتى نأتي لشرحه لاحقا في دروس أخرى . إذا يجب أن تنتبه أنه إذا أردت أن تستخدم مكتبات السي++ الجديدة قم بتضمين ملفات التعريف بدون استخدام الامتداد h حتى يعرف المترجم أي من المكتبات سوف يربطها مع برنامجك . مثلا إذا أردت استخدام المعدلات في الخرج سوف تكتب التالي :

```
#include <iostream>
#include <iomanip>
using namespace std;
```

في الدروس القادمة سوف نستخدم هذه الصيغة في تضمين مكتبة الدخل والخرج .

3.6 ملخص الدرس

- ميزات الدخل والخرج في سي++ يتم الحصول عليها من مكتبة الدخل والخرج القياسية Standard I/O stream library .
- عند استخدام المكتبة السابقة يجب تضمين الملف iostream.h مع البرنامج باستخدام العبارة #include<iostream.h>
- يمكن تنسيق الخرج باستخدام المعدلات manipulators .
- يفضل استخدام التابعين get و put من أجل تبادل محرف وحيد مع المسار (stream) .

موسوعة البرمجة بلغة C++

C++ encyclopedia Programming

المؤلف: مجلاد مشاري السبيعي
البريد الإلكتروني: magedxl@hotmail.com

يمكنكم مراسلتي إذا كان هناك آي مشاكل متعلقة بعدم التوضيح أو نحوه مما يوجد في هذا الكتاب ...

*هذا الكتاب مجاني للجميع

إهداء

هذا الكتاب إهداء إلى ارض العراق الحبيبة .. نعم العراق لقد أضعنا العراق ارض الفرات ودجلة ، فخر الإسلام والمسلمين بـغـداد ...
أضعناها .. أضعناها..
لكنك لست ببعيد وستعودين أرضا للخلافه الإسلامية..
قريبا .. قريبا

الـيـك يـاعـراق

مقدمة عامة

أعزائي..
تم وبحمد الله الانتهاء من هذا الكتاب .. المفيد للجميع بإذن الله المبتدئين والمحترفين على حد السواء..
وقد قمت بإنشاء هذا الكتاب بعد أن لاحظت قلت الكتب التي تتحدث عن لغة Visual C++..
وهي اللغة التي تم منها بناء حزمة اوفيس المعروفة لتطبيقات الحاسب..
لذلك أرجو أن يحوز هذا الكتاب على رضاكم وهو يحتوى على العديد من الأمثلة التطبيقية مع شرح الأساسيات والعديد من الأجوبة التي يحتاجها مبرمجي C++.
وبالله التوفيق..

فهرس المحتويات

الأساسيات
مكونات C++ وادواتها
Basic Elements of C++

رموز لغة C++

*الرموز المستخدمة في لغة C++

١- الحروف الإنجليزية الكبيرة A.B.C

٢- الحروف الإنجليزية الصغيرة a.b.c

٣- الأرقام العربية الأصل 1.2.3

٤- رموز خاصة مثل:

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

الجدول١-١

وتعد هذه الرموز بأنواعها المادة الخام التي تتكون منها مفردات لغة C++ ، وإذا درست لغة أخرى قبل لغة C++ ، فانك تلاحظ أن لغة C++ ، تستعمل رموزا إضافية في لوحة مفاتيح الحاسب لا توجد في بعض اللغات.

*كلمات لغة C++

الكلمات نوعين:-

١- أسماء تعريفية (Identifiers)

وهي الأسماء التي نسميها نحن " المبرمجون" تعرف الحاسوب بما تريد.

وتطلق الأسماء التعريفية على:-

A- المتغيرات.

B- الاختزانات (الدوال).

C- المؤشرات.

*قواعد تسمية الأسماء التعريفية في لغة C++ :-

١- أن يكون الاسم مكتوبا من سلسلة متصلة من الحروف أو الأرقام بشرط أن يبدا بحرف أو بخط تحتي "_"

٢- أن لا يحتوى الاسم على رموز خاصة عدا الخط التحتي "_"

٢- أن لا يكون الاسم إحدى الكلمات المحجوزة.

بعض الأمثلة الصحيحة على الأسماء التعريفية:

a. B6
b. X_ray
c. Matrix
d. Ok_
e. A
f. Soft_fine
g. Door12
h. _new

وهذه أسماء تعريفية غير مقبول (invalid) للأسباب المبينة إزاء كل منها:

7-up Ù لانه بدا برقم وليس بحرف.

b6.1 Ù لاستعماله الرمز الخاص (.)

salim! Ù لاستعماله الرمز الخاص (!)

Ъ2 Ù لا يجوز استعمال حروف غير إنجليزية.

No#1 Ù لاستعماله الرمز الخاص (#)

ومن الجدير بالذكر ، أن لغة C++ تفرق بين الحروف الأبجدية الصغيرة والكبيرة ، فمثلا الأسماء : SYSTEM , system, System تعامل كأسماء مختلفة عن بعضها البعض بسبب اختلاف معاملة المترجم للحروف الصغيرة والكبيرة.

٢- الكلمات المحجوزة

وهي كلمات قياسية معروفة مسبقا لمترجم C++ ، وتكتب عادة بحروف صغيرة ، ولها معان خاصة بها تؤديها في برنامج C++ ، وهذه الكلمات المحجوزة حسب الترتيب الأبجدي هي:

near	Static	asm	Double	long	Sizeof
do	int	While	new	auto	else
For	This	Void	Delete	Goto	if
const	Entry	char	Class	Public	Case
Continue	Extern	struct	inline	float	Private
Virtual	Volatile	Frinde	enum	near	Static
cdecl	Default	inline	Overload	Unsigned	Typedef
Signed	Pascal	Operator	Switch	Template	Union
Register	Protected	far	Catch	char	Const
				break	Return

الجدول٢-١

وينبغى التنبيه إلى أن هذه الكلمات المحجوزة ، لا يجوز إعادة تعريفها أو استعمالها لغير ما خصصت له.

وكما تلاحظ من قائمة الكلمات المحجوزة ، أن لغة C++ تعد لغة صغيرة إذ تتكون من عدد قليل من الكلمات المحجوزة تقريبا ٥٢ كلمة محجوزة فقط.

- تمثيل الثوابت العددية Numeric Constants

يمكن تمثيل الثوابت العددية ، في لغة C++ بثلاث صور هي:-

a. الثابت العددي الصحيح integer
   - هو عدد مكون من الأرقام من 0 Ù 9
   - لا يحتوى على فاصلة عشرية.
   - يمكن أن يحوى الإشارة "+" أو "-"

أمثلة صحيحة على الثابت العددي الصحيح:-
0
15
1000
321
-61

والأعداد التالية غير صحيحة للأسباب المبينة إزاء كل منها:
3.31 : لانه يحتوى على فاصلة عشرية.
1,000 : لانه يحتوى على فارزة.
J72 : لانه يحتوى على حرف أبجدي.
2 4 : لوجود فراغ بين العديدين.
1992 1992 1999 : لوجود فراغ وأيضا لان العدد كبير.

كما يمكن تصنيف الأعداد الصحيحة في لغة C++ ، حسب طولها ، والسعة التخزينية لها في الذاكرة مثلا:-
الثوابت الصحيحة 19897 , 40000 تسمى ثوابت صحيحة طويلة long int.
الثوابت 16- , 80 , 45 تسمى ثوابت صحيحة قصيرة short int.
الثوابت 20000 , 967 تسمى ثوابت صحيحة بدون إشارة unsigned int.

والفرق بين الثوابت الطويلة والقصيرة هو في عدد الوحدات التخزينية المطلوبة لكل نوع في الذاكرة ، فالطويلة تأخذ حيزا اكبر ، والقصيرة توفر عدد الوحدات التخزينية المستعملة ، أما الثوابت الصحيحة بدون إشارة unsigned int ، فان استعمالها يوفر وحدة تخزينية واحدة تستعمل للاشاره عندما تذكر كلمة unsigned ، قبل int ،

وذلك بإزاحة القيمة إلى قيمة موجبة بدون إشارة ، ولكل نوع من الأنواع السابقة تطبيقاته المناسبة.

b- الثابت العددي الحقيقي Floating-point Constants

- هو عدد مكون من الأرقام 0 Ù 9
- يجب أن يحتوى على فاصلة عشرية
- يمكن أن يحوى الاشاره "+" أو "-"
- لا يجوز أن يحتوى على فارزة "،"

أمثلة على ثوابت عدد حقيقي تستعمل الفاصلة العشرية بشكل صحيح :-

421.5
10.6
0.0
0
01
-68.0

والأمثلة الاتيه غير صحيحة للأسباب المبنية إزاء كل منها:-

1000 : لانه لا يحتوى علي فاصلة عشرية.
4,000.21 : لانه يحتوى على فارزة.
2 83.4 : لان يحتوى على فراغ .

- تمثيل الثوابت الرمزية Non-numeric
  - سلسلة من رموز اللغة (أحرف أرقام رموز خاصة) محصورة بين حواصر علوية مزدوجة (علامات تنصيص أو اقتباس)

ومن الأمثلة على الثابت الرمزي ما يأتي :-
"first"
"my name is"
"30+50=80"
"my,no=123.04"
"Islam"

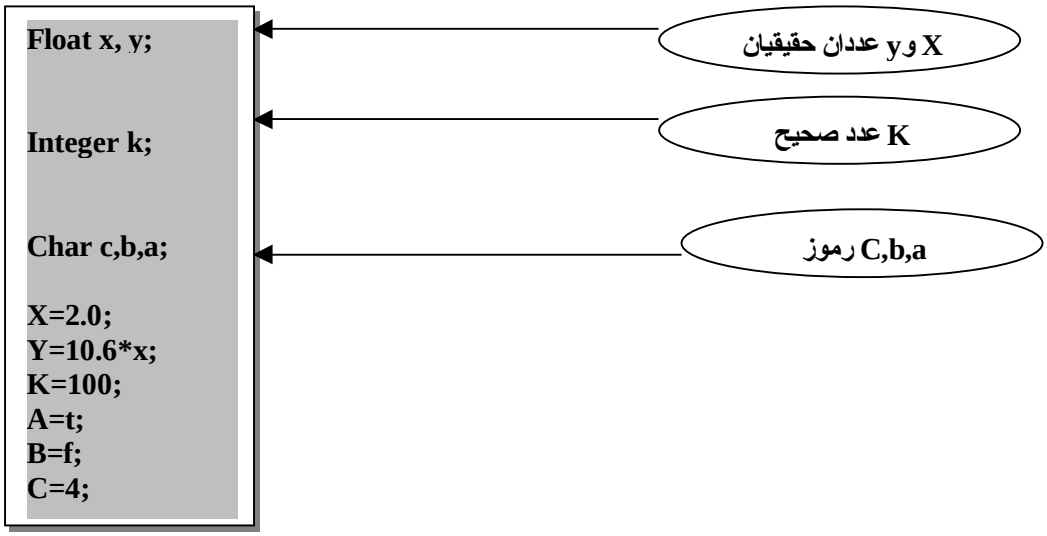
وتلاحظ أننا سمينا أي نص موضوع بين حاصرتين مزدوجتين ثابتا رمزيا والصحيح أن تسميته ثابتا رمزيا هي من قبيل المجاز والاصطلاح لا الحقيقة ، واما كلمة رمزي : فلان النص مكون من عدد من الرموز ، وتسمية بعض الكتب بالثابت غير العدد Non-numeric.

ملاحظة/
كل الثوابت الرمزية الواردة أعلاه ، وان استخدمت ارقاما حسابية داخلها ، ألا أنا لا تحمل أي قيمة حسابية ، وليس لها معنى حسابي ، وتستخدم مثل هذه الثوابت عادة كمعلومات توضيحية مع نتائج البرنامج.

المتغيرات

هي أسماء (عناوين) لمواقع في ذاكرة الحاسوب ، يخزن بها رموز أو أعداد.

وبما أن أنواع المعلومات المراد تخزينها تكون عادة مختلفة مثل القيم الصحيحة أو الحقيقية أو الرمزية ... الخ فانك تحتاج أن تعلم المترجم في بداية البرنامج عن أنواع المتغيرات التي تريد استعمالها في البرنامج ، فمثلا في السطور التالية تبين أن المتغيرين x و y حقيقيان ، والمتغير k صحيح ، والمتغير c,b,a رمزية.



لاحظ السطرين الأول ، والثالث يحتويان على اكثر من متغير حيث يفصل بين كل متغيرين ، فاصلة وكذلك يمكن تحديد أنواع المتغيرات ، بذكر التفصيل الدقيق للنوع ، من حيث طول السعة التخزينية ، أي هل هو صحيح قصير أم طويل حقيقي قصير أو مضاعف ... الخ

- وتقسم المتغيرات لنوعين :-

١- متغيرات عددية

وهي مواقع في الذاكرة تخزن بها أعداد .

٢- متغيرات رمزية

وهي مواقع في الذاكرة تخزن بها رموز.

٣- متغيرات منطقية

وتخزن بها قيمة منطقية أما FALSE =0 أو TRUE=1

الأدوات المستعملة في لغة C++

يوجد في لغة C++ ثلاثة أنواع من الأدوات وهي:
الأدوات الحسابية – الأدوات المنطقية والعلاقة – الأدوات الدقيقة وفيما يأتي تفصيل بالأنواع الثلاثة:-

الأدوات الحسابية Arithmetic Operators

تسمح لغة C++ باستعمال الأدوات الحسابية من جمع وطرح وضري وقسمة ، كاللغات الأخرى ، إلا أن عملية الرف إلى أس ، ليس لها أدوات مباشرة مثل الأداة h في Basic والأداة ** في فورتران ، وانما تتم علمية الرفع إلى أس في لغة C++ بطريقة أخرى ..
كما تختلف القسمة في لغة C++ عنها في Basic إذا أن أي جزء كسري ينتج عن القسمة يهمل مهمها كان كبيرا ، كما في لغتي باسكال وكوبول فمثلا ناتج القسمة 8/3 هو 2 لا الكسر 0.666 يهمل ، ويكون ناتج القسمة باستعمال الأداة / صحيح العدد.
ويمكننا الآن أن نلخص الأدوات الحسابية المستعملة في لغة C++ فيما يأتي:-

الأداة	وظيفتها
-	للطرح أو كأشاره سالبة
+	للجمع
*	للضرب
/	للقسمة
%	لباقي القسمة الصحيحة
- -	للنقصان
+ +	للزيادة

الجدول٣-١

ويختلف أداء بعض الأدوات الحسابية حسب نوع المعطيات الصحيحة ، أو الحقيقة ، أو الرمزية فعند معاملة المعطيات الحقيقية للأدوات الحسابية ، يمكن القول أن العلميات الأساسية من جمع وطرح وضرب ، تجري بالطريقة التي نعرفها ، إلا أن هناك محذورا يجب أن نذكر به ، وهو أن تتعدي قيمة النتيجة من أية عملية حسابية الحدود المرسومة لنوع المتغير الناتج ، لان لكل نوع من أنواع المتغيرات حدودا ، يعد تجاوزها خطأ ينتج عنه خطأ في النتائج ، وعند معاملة المعطيات الصحيحة بالأدوات الحسابية تعمل الأدوات بالطريقة التي نتوقعها ، وعند تعدي الحدود المسموح بها في القيم الصحيحة ، فان هذا يعني أن خطأ قد وقع overflow ، وفي هذه الحالة لن تتلقى من المترجم أية رسالة خطأ ، فمثلا إذا كان لدينا البرنامج التالي:

```
Main( )
{
int n = 33000;
n = n * 3;
}
```

عند طباعة النتيجة n النهائية نتوقع أن يكون الجواب 99000 ، ألا أن الجواب في هذه الحالة لن يتعدى 30464 ، وهو الحد الاعلى المسموح به للقيمة الصحيحة ، وهناك أمر أخرى يتعلق بالقسمة فعندما نقسم 8 على 3 قسمة صحيحة 8/3 فان الناتج يكون صحيحا وهو 2 فقط ، وإذا ما رغبت أن تحافظ على الجزء الكسري الذي أهمل واسقط ، يمكنك أن تحول القسمة إلى قسمة حقيقة 8.0/3.0 حينئذ فان الناتج سيكون 2.667 لهذا السبب أدخلت لغة C++ أداة باقي ا لقسمة % ويسمى Modulus Operator ويستعمل على النحو التالي:

```
7 % 3
```

تعطي الجواب 1 وهو باقي القسمة الصحيحة 7/3 ، ومن الجدير بالذكر أن كلا من باسكال وكوبول تستعملان مثل هذه العملية ، ففي باسكال تكتب هذه العملية على النحو 7 mod 3 ، وكلمة MOD هي اختصار Modulus ، أما في لغة C++ فتستعمل الأداة % لتقوم بهذا العمل.

الأدوات الأحادية والثنائية Unary and Binary Operators

تعد جميع أدوات الجمع والطرح والضرب والقسمة وباقي القسمة أدوات ثنائية binary أي أنها تأخذ (تتعامل مع) قيمتين وتنتج قيمة واحدة ، فمثلا نتيجة 3*2 هي القيمة 6 وهناك الأداة الأحادية – عندما تتعامل مع قيمة واحد فمثلا (1992-) تمثل الإشارة ا لسالبه وهي هنا أداة أحادية Unary ، والعملية هنا ليست عملية طرح كما نعلم.

الزيادة والنقصان Increment and Decrement

من مزايا لغة C++ أنها تستعمل الأداتين الحسابيتين + + و - - لزيادة القيم بمقدار 1 أو إنقاصها بمقدار 1 ، والمثال التالي يبين طريقة الاستعمال:

```
A++;
```

```
++a;
```

معناه إضافة قيمة 1 إلى a ويمكن كتابتها بصوره مكافئة على النحو التالي:-

```
A=a+1;
```

وبالطريقة نفسها يمكن إنقاص 1 من قيمة a على النحو:-

```
--a;
```

أو

```
a--;
```

وهو يكافئ الصورة

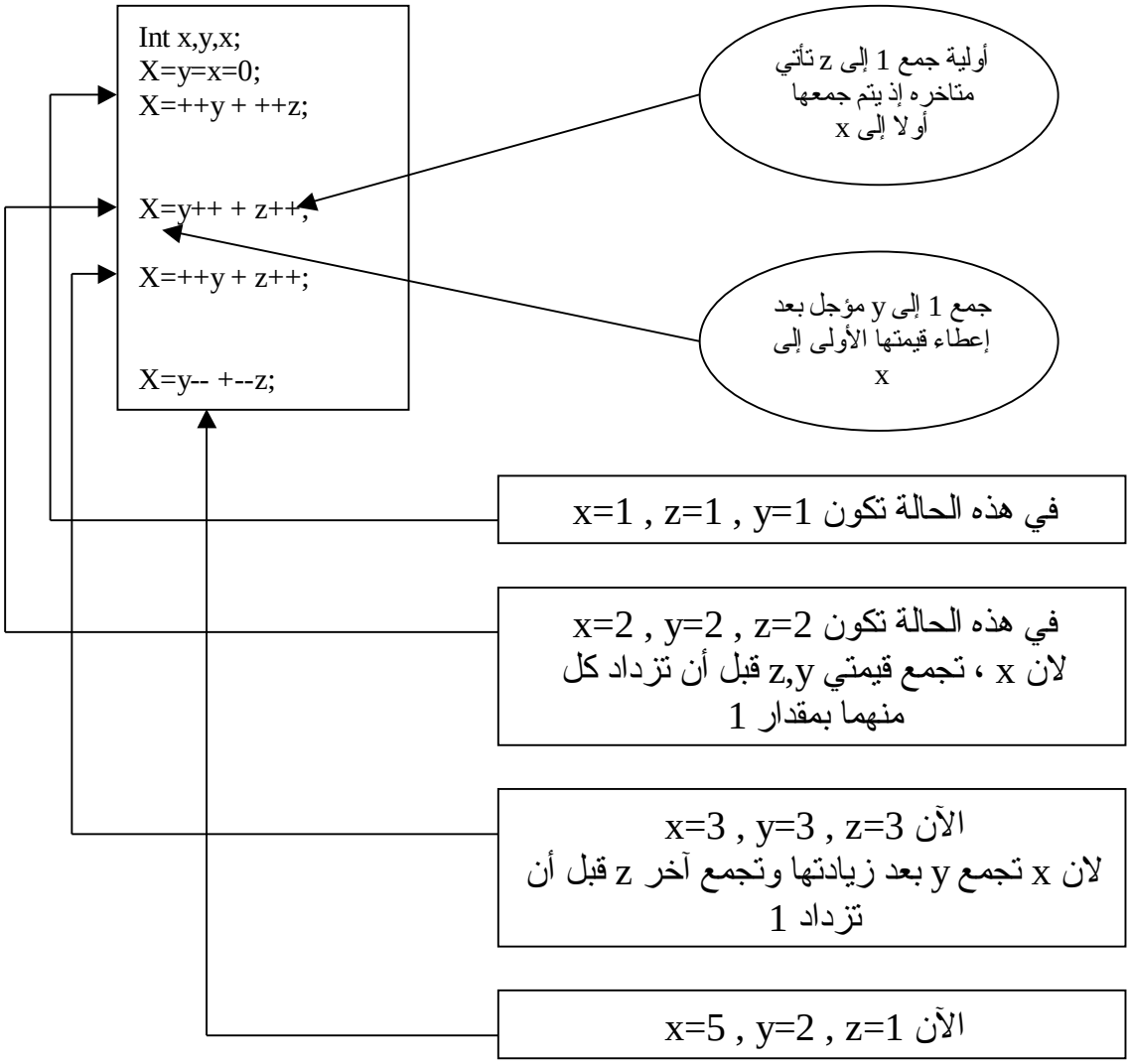
```
A=a-1;
```

لكن هناك فرقا في سرعة التنفيذ ، فالتعبير ;a++أسرع من التعبير ;a=a+1 وهذه هي الفائدة من جراء استخدام مثل هذه الأدوات .

ومما ينبغي التنبيه إليه هنا أن هناك فرقا بين a++ و ++a ، صحيح أن كلا من التعبيرين يجمع 1 إلى a ، لكن عند استعمال a++ في تعبير من التعابير ، فان a

تـزداد قـبل اسـتخراج قـيمة التعبـير ، بيـنما فـي حالـة a++ تسـتخرج قـيمة التعبـير باستعمال قيمة a الحالية قبل زيادتها بمقدار 1 ، وبعد ذلك تتم زيادة a بمقدار 1 آي أن العملية الأولى جمع تقديم ، والثانية جمع تأخير ، وينطبق هذا الكلام أيضا على – a و a-- .

مثال:



وبإمكانك كتابة الجملتين:

Int x,y,z
X=y=z=0

في جملة واحد على النحو:

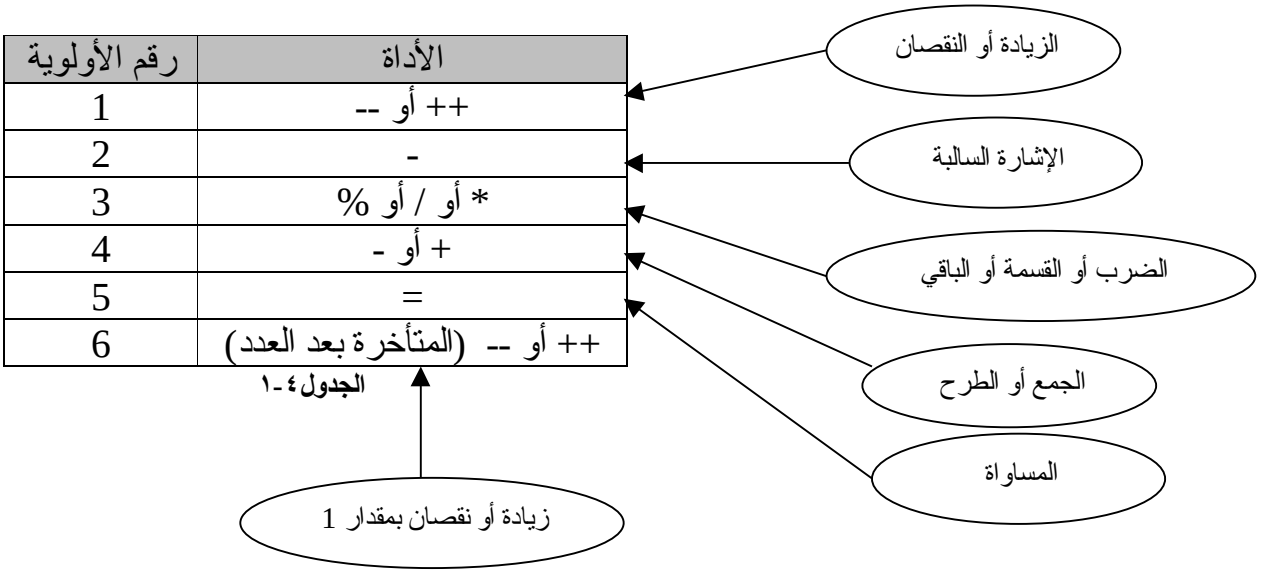
Int x=y=z=0

أولية العمليات للأدوات الحسابية Precedence of Arithmetic Operations

يمكن القول أن أولية تنفيذ العمليات كما يجريها مترجم C++ بالنسبة للأدوات الحسابية هي على النحو التالي:

الأداة	رقم الأولوية
++ أو --	1
-	2
* أو / أو %	3
+ أو -	4
=	5
++ أو -- (المتأخرة بعد العدد)	6

الجدول٤-١



ملحوظة:

إذا تساوت أوليتان مثل الجمع والطرح في تعبير ، فتقدم العملية الأقرب إلى يسار التعبير ، وعند استعمال الأقواس لأي تعبير فان الأقواس تأخذ الأولوية الأولى في التنفيذ قبل (الزيادة أو النقصان) ، كما في لغات البرمجة الأخرى ، والأمثلة الآتية تبين مفهوم الأولوية (الأسبقية):-

```
X + y / z * a
```

يأخذ تسلسل أولويات عملياته الشكل والخطوات التالية:-

١- العملية الأولى: القسمة y / z

٢- العملية الثانية: a * (y/z)

٣- العملية الثالثة: جمع الناتج في الخطوة 2 إلى x فتكون النتيجة:

X + y / z * z

لاحظ أننا بدأنا بإجراء العلميات الحسابية من اليسار إلى اليمين ، وتعطى ألا ولويه لأية عملية حسب قاعدة الأولوية ، فجاءت القسمة ، في المثال قبل الجمع ، كما جاء الضرب بعد القسمة وتلا ذلك الجمع كأخر عملية.

الأدوات العلاقية والمنطقية Relational and Logical Operations

يرجع اسم الأدوات العلاقية إلى العمليات المختصة بالقيم التي بينها علاقات وهو أجراء عمليات مقارنة منطقية بين كميات حسابية أو رمزية ، وتكون نتيجته منطقية وهي أما نعم (true) أو (false) ، ويكثر استخدام التعابير المنطقية في الجمل الشرطية ، والأمثلة الآتية تبين لك ما هو التعبير المنطقي:

التعبير المنطقي: x= =y جواب أما نعم أو لا .
والتعبير المنطقي: matrix > 100.0 جواب أما نعم أو لا .

وفي لغة C++ تعامل النتيجة لا (false) على أنها صفر (0) وتأخذ النتيجة نعم (true) أية قيمة غير الصفر والمشهور أنها (1) .
ويبين لنا الجدول التالي الأدوات العلاقية والمنطقية:

الأدوات العلاقية

الأداة	معناها
>	اكبر من
<	اصغر من
>=	اكبر من أو يساوي
<=	اصغر من أو يساوي
= =	يساوي
!=	لا يساوي

الجدول٥-١

الأدوات المنطقية

الأداة	معناها
&&	And (حرف العطف واو)
||	Or (حرف العطف أو)
!	Not (للنفي) أداة أحادية unary

الجدول٦-١

إليك الآن هذه الأمثلة : افرض أن int a=b=3;
فان التعبير a < 3 نتيجته false أي 0
التعبير a<=3 نتيجته true أي 1
التعبير a>b نتيجته false أي 0
التعبير a!=b نتيجته false أي 0
التعبير a= = b نتيجته true أي 1

جدول الصدق سوف نسوق هذا الجدول كالتالي:-

جدول الجمع X && y (x and y)			جدول التخيير X‖y (x or y)			جدول النفي !x (not x)	
X&&y	Y	X	X‖y	Y	x	!y	x
F	F	F	F	F	F	T	F
F	T	F	T	T	F	F	T
F	F	T	F	T	T		
T	T	T	T	T	T		

الجدول٧-١

!! المساعدة على فهم جداول الجمع والتخيير والنفي أعلاه:-

جدول الجمع:

تخيل أن F تمثل السم ، وان T تمثل العسل ، وبناء على ذلك فان F&&T تعني سما مع سم والنتيجة سم أي F ، كذلك F&&T تعني خلط السم مع العسل والنتيجة سم أي F ، وكذلك T&&F ينتج عنها F أما T&&T فهي عسل على عسل أي أن النتيجة T .

جدول التخيير:

فلو خيرت بين السم F والسم F F‖F فالنتيجة معروفه F أما بين السم والعسل F‖T فالنتيجة سوف تكون بالطبع للنجاة عسل T ، ونتيجة T‖T هي عسل T...

الأدوات الدقيقة Bowties Operators

تتميز لغة C++ عن سائر اللغات الراقية مثل فيجوال بيسك وباسكال أنها تستخدم أدوات دقيقة على مستوى وحدة التخزين الأولية [Bit] والمختصرة من Binary Digit*

سميت هذه الأدوات بالدقيقة أو أدوات (البت) لأنها تتعامل مع [bit] (وحدة التخزين الأولية) مباشرة ، فحصا ، وضبطا ، وإزاحة ، وتستعمل هذه الأدوات مع المعطيات الصحيحة int والرمزية char فقط ، ولا تستعمل مع غيرها من أنواع المعطيات..

والجدول التالي يبين الأدوات الدقيقة ووظيفة كل منها:

الأداة	عملها
~	آداه أحادية (not)
&	حرف الواو (و) (and)
|	حرف العطف (و) (or)
>>	إزاحة إلى اليسار
<<	إزاحة إلى اليمين
^	(أو) الاستثنائية (xor)

الجدول٨-١

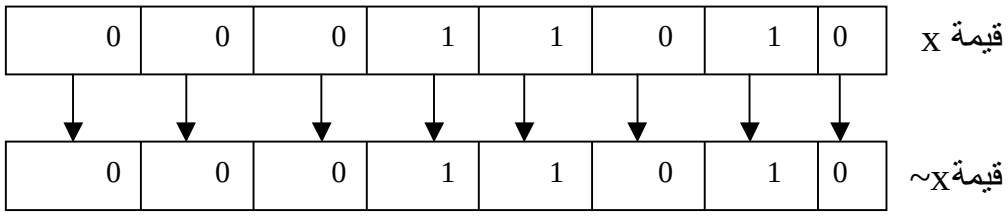
وكغيرها من الأدوات ، فان الأدوات الدقيقة تتبع قواعد الأولية وحسب الترتيب التالي:

الأداة	أولويتها
~	الأولى
>>أو<<	الثانية
&	الثالثة
^	الرابعة
|	الخامسة

الجدول٩-١

أداة النفي (~)

تعمل هذه الأداة على إبدال الصفر (0) بواحد(1) أو العكس ، ومعنى هذا أنها تضع 0 مكان 1 وكذلك 1 مكان 0 ، فمثلا لو كان لدينا قيمة x ممثلة في النظام العددي الثنائي التالي (من 8 بت):-

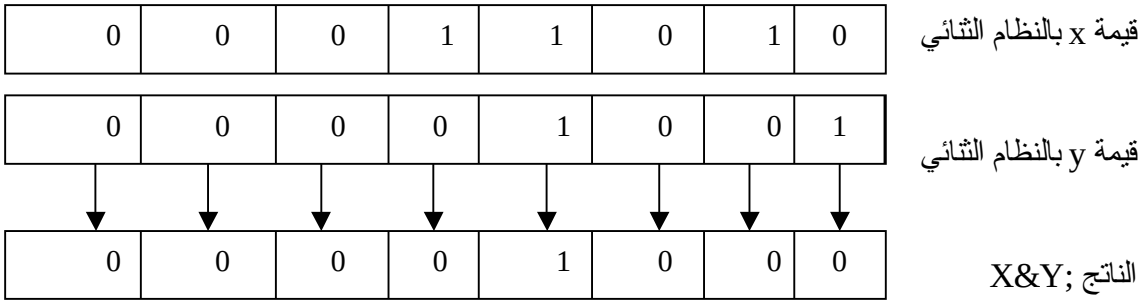


ومعنى ~ النفي (not) ومعنى النفي هنا التضاد بين 0 و 1 في النظام العددي الثنائي ، فعندما تنفى 0 تثبت بدلا منه 1 والعكس صحيح ، وهذا يوضحه لك المثال السابق إذ تم (نفي) قيمة x بالبت ليصبح ~x في جميع مكونات من البت.

أداة الجمع &

المثالي التالي يوضح كيفية جمع القيم عند تمثيلها بالنظام العددي الثنائي:

العملية ;x&y

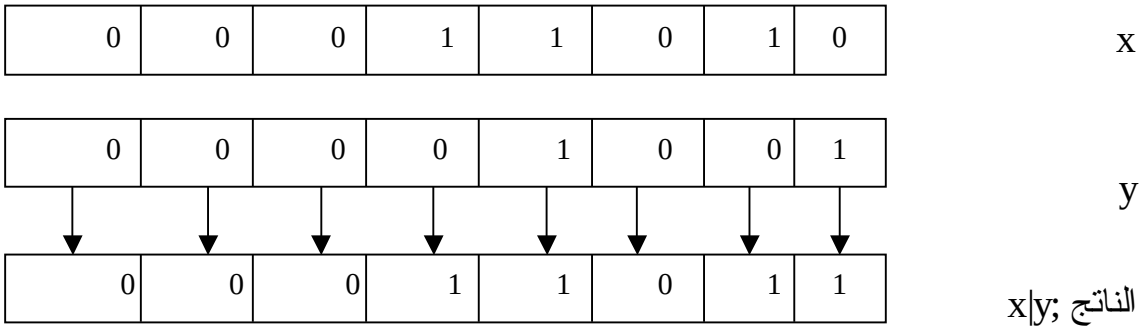


حيث يجمع 0+0 ويعطي 0 ، ويجمع 1+0 ليعطي 0 ويجمع 1+1 ويعطي 1

T Ù T&T F Ù F&T F Ù F&F (انظر جداول الصدق السابقة)

أداة الاختيار|

إذا أردنا استعمال أداة الاختيار مع المثال السابق لقيمتي X وy على النحو ;x|y

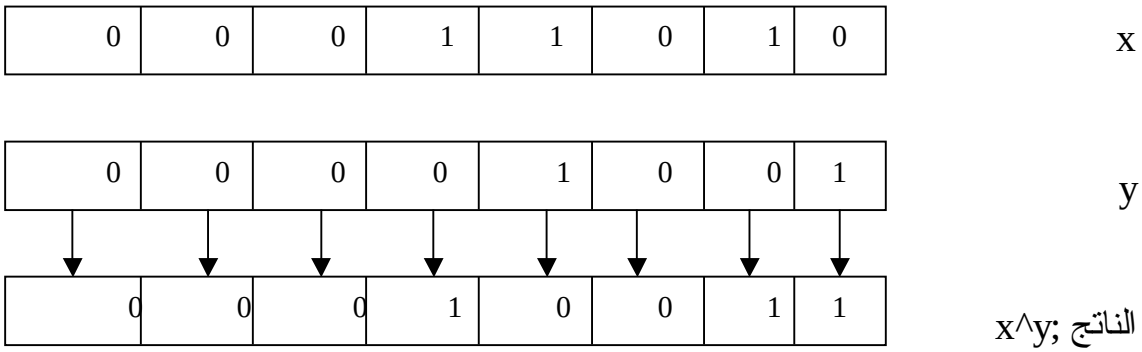


حيث الاختيار بين 0 و1 هو 1 ، والاختيار بين 1 و1 هو 1 ، وبين 0 و0 هو 0 .

T Ù T|F T Ù T|T T Ù F|T (انظر جداول الصدق السابقة)

أداة الاختيار الاستثنائي ^

خذ المثال السابق نفسة لترى كيف تؤثر هذه الأداة



لاحظ أن الاختيار بين 0 و0 نتيجته 0 ، والاختيار بين 1 و0 نتيجته 1 ، وبين 1 و1 نتيجته 0 ، أن أداة الاختيار الاستثنائي ^ (XOR) تختلف عن أداة الاختيار (OR) | في أن الأولى حالة خاصة من الثانية إذ تكون نتيجتها true (1) عندما يكون أحد طرفي الاختيار x أو y نتيجته true ، وما عدا ذلك فنتيجتها دائما false (0) ، كما لاحظت في هذا المثال ، أما في حالة الاختيار العام (OR) فان نتيجة 1|1 هي 1.

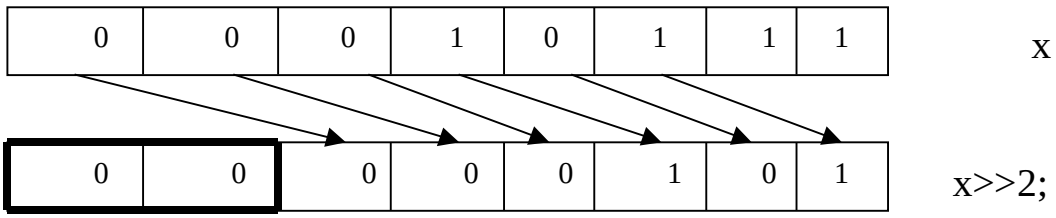
أدوات الإزاحة << و >>

قد تبدو أدوات الإزاحة غريبة على المبرمج الذي يستخدم لغات أخرى مثل Basic و Pascal .. الخ . حيث ينتج عن استعمال إحداهما إزاحة قيمة المتغير الصحيح بالنظام الثنائي (بالبت) يمينا أو يسارا عددا من الخانات حسب الطلب ، وتملا الخانات المفرغة من الجهة الموجبة أصفارا ، ومن الجهة السالبة تملا آحادا.

والأمثلة التالية توضح طريقة الاستعمال.

مثال:

الجملة x>>2; عند تنفيذها على قيمة x (وهي 23 بالنظام العشري) بالنظام الثنائي فان العملية تتم على النحو التالي:



النتيجة من الإزاحة بمقدار خانتين (٢ بت) لليمين تصبح قيمتها :

5 بالنظام العشري.

وهذا يعني أن 23>>2; تعطي النتيجة 5 .

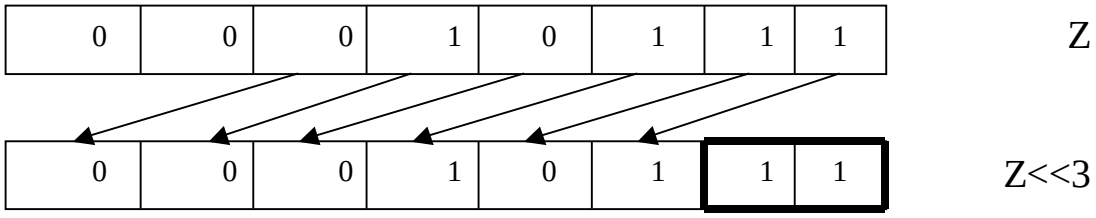
حيث 23 القيمة المزاحة.

2 عدد خانات الإزاحة المطلوبة لليمين.

لاحظ أن الخانتين المفرغتين بسبب الإزاحة لليمين قد ملئتا بمصفرين.

مثال على إزاحة قيمة سالبة:

Z = - 50<<2;



إزاحة (٢بت) لليسار.

الجديد في هذا المثال أن الإزاحة لقيمة سالبة ينتج عن كل خانة مفرغة القيمة 1 وليس 0 كما في المثال السابق.

أدوات أخرى لم تذكر Other Operations
الأداة الشرطية the conditional operator

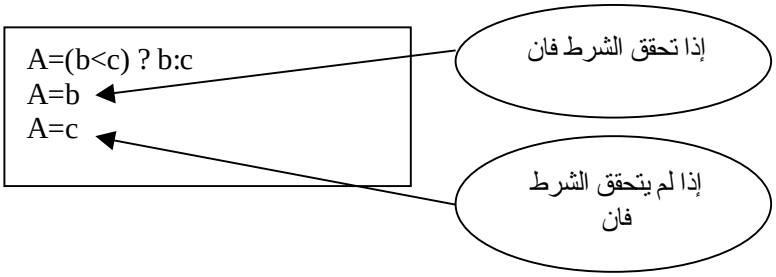
وهذه الأداة تتميز بها لغة c++ عن غيرها ، إذ تحل محل جملة شرطية مشهورة في بيسك وفورتران وباسكال وهي if-then-else ، وتعتبر هذه الأداة أداة ثلاثية لأنها تتعامل مع ثلاث كميات حسب صيغتها العامة التالية:

```
Expression1 ? Expression2: Expression3
```

فلو كان لدينا الجملة الشرطية التالية على سبيل المثال:

```
If (b<c)
A=b;
Else
A=c
```

معناها: انه إذا تحقق الشرط (b<c) فينفذ السطر a=b ، وإلا فان a=c وباستعمال الأداة الشرطية ؟ يمكننا أن نكتب بدلا من الجملة الشرطية كلها الجملة المختصرة التالية:



أداة العنونة (&) و (*) the & and * Pointer Operator

المؤشر Pointer هو عنوان المتغير المؤشر في الذاكرة ، وللمتغير المؤشر فوائد جمة في عملية البرمجة نذكرها عند التعرض لها في الفصول القادمة بإذن الله ، ويكثر استعمال هاتين الدالتين مع المتغيرات المؤشرة المحجوزة لها في أماكن خاصة في الذاكرة .

وتعد الأداة & هنا أداة أحادية تتعامل مع كمية واحدة فقط ، حيث تقوم بإعطاء عنوان الطرف الأيمن للمعادلة ، للطرف الأيسر منها ، فمثلا العملية:

```
X=&y;
```

تعطي عنوان y في الذاكرة ، ووضعه في مخزن x ، وهذه الجملة تختلف طبعا عن الجملة الحسابية x=y التي معناها وضع قيمة y في مخزن x أما عند استعمال & قبل y فمعناها إعطاء عنوان مخزن y في الذاكرة فقط للمتغير x ، وليس قيمة y فلو كانت y=10 ، ورقم (عنوان) موضعها في الذاكرة هو 120 ، فان x تأخذ قيمة 120 عند استعمالنا & مع y وبالنسبة للأداة الثانية * فهي أداة أحادية أيضا ، ومكملة للأداة & ، ولذلك لو كان لدينا الجملة التالية التي تستعمل الأداة * .

```
S=*x;
```

فانه يفهم منها أن x تحوى عنوان (موقع) المتغير y في الذاكرة ، وان هذه الجملة تضع في مخزن s قيمة المتغير ، صاحب المخزن الأصل ، آي قيمة y وهي 10 كما في المثال السابق ، وبالتالي فان قيمة 10 تخزن في مخزن s في الموقع (العنوان) 120 لذلك نرى أن جملة x=*&y تكافئها الجملة x=y وهذا يعني أن الأداتين تعملان وكأن الواحدة معكوسة للأخرى

ومما يجب الانتباه إليه ، أن الأداة * تستخدم أيضا لعمليات الضرب الحسابي كما تستعمل الأداة & كأداة AND دقيقة ، ولذا لا يلتبس عليك الأمر بين الاستعمالين المختلفين.

أداة تعيين الطول sizeof

تعد هذه الأداة أداة أحادية (unary) ، وتستعمل لتعيين طول المتغيرات (بالبايت) ، وتختلف أطوال المتغيرات حسب أنواعها ، ولذا طلب تعيين طول متغير باستعمال sizeof ، ينبغي ذكر نوع هذا المتغير بين قوسي sizeof، فمثلا:

```
Int n;
N=sizeof (n);
```

حيث ستكون نتيجة n هنا تساوي 2 بايت ، هي طول المتغير n الصحيح (int) لان طول الصحيح عادة هو 2 بايت ، وطول الحقيقي 4 بايت ، كما في المثال التالي:

```
Float x;
Z=sizeof (x);
```

حيث ستكون نتيجة z هي 4 بايت ، وهي طول x الحقيقي.

الفاصلة (,) كأداة The Comma Operator

وهي أداة استثنائية (binary) وتحتل الأولوية الأخيرة في سلم الأداة المختلفة وتأخذ الصيغة العامة التالية:

```
Experssion1, Experssion2
```

فعندما تفصل فاصلة بين تعبيرين فان تسلسل العمليات يأخذ الترتيب التالي:

١- تستخرج قيمة التعبير الأول (علي يسار الفاصلة) ثم تعطى للتعبير الثاني.

٢- تستخرج قيمة التعبير الثاني (علي يمين الفاصلة) كقيمة نهائية للتعبير كله كما في المثال التالي:

```
A=(b=2,b+1);
```

حيث يعطى المتغير b قيمة 2 في التعبير الأول (يسار الفاصلة) ، ثم وضع هذه القيمة في b الأخرى في التعبير الثاني (يمين الفاصلة) ، فتصبح قيمة التعبير على اليمين (b+1) تساوي 3 وتكون هذه القيمة نتيجة التعبيرين النهائية .

مثال آخر:

```
B=8;
A=(b=b-4,12/b);
```

في هذا المثال يتم إعطاء b القيمة 8 أولا ، ثم عند تنفيذ السطر الثاني ، يعطي b في التعبير الأول داخل القوسين القيمة (b-4) آي (8-4) ، وتساوي 4 ، وهذه تعطى للتعبير الأيمن ، حيث تتم القسمة (12/b) آي (12/4) فتصبح نتيجة التعبير كله 3 ، التي تعطي بالتالي للمتغير a .

جمل التعريف

جمل التعريف هي جمل تقوم بتعريف القيم.

مثال:

```
Int a;
```

يقابل هذه الجملة في فيجوال بيسك

```
Dim a as integer
```

وتقوم بحجز مكان في الذاكرة المشار إليه ، بالاسم a لتخزين قيمة عددية صحيحة.

أنواع البيانات الممكن تخزينها في الذاكرة المستخدمة لـ C++

١. char لتخزين رمز واحد فقط.
٢. int لتخزين عدد صحيح.
٣. float لتخزين عدد حقيقي.
٤. double لتخزين عدد حقيقي كبير.
٥. void لتخزين بيانات خالية.

أن معرفة أنواع البيانات ، وكيفية استعمالها ، تعد ضرورية لفهم لغة c++ فلاستعمال المتغيرات ، مثلا ، نحتاج أن نعلن في بداية كل برنامج ، أو بداية الدوال عن أنواع هذه المتغيرات ، ويتم التعامل معها ، خلال البرنامج ، في ضوء أنواع معطياتها فمثلا الإعلان عن التالية:

```
Int a,b,x;
```

تخبر مترجم c++ أن يتعامل مع هذه المتغيرات ، على أنها متغيرات صحيحة وكذلك جملة الإعلان التالية:

```
Float m,y;
```

تخبر مترجم c++ (C++ compiler) أن هذه المتغيرات من النوع الحقيقي.

الثوابت الرمزية ذات الشرطة المعكوسة

حيث أننا لا نستطيع استعمال بعض الرموز الموجودة في لوحة مفاتيح الحاسب كثوابت رمزية ، فقد استحدث لغة c++ شفرات رمزية خاصة تستعمل شرطة معكوسة لها ، وهذه الشفرات مدونة في الجدول التالي:

الشفرة	معناها	القيمة الصحيحة لها
"\b"	رجوع بمقدار خانة واحدة	8
"\n"	سطر جديد	13
"\t"	ترتيب أفقي	9
"\0"	للقيمة الخالية	0
"\r"	علامة رجوع	13
"\v"	ترتيب عمودي	11
"\\"	الشرطة المعكوسة \	92
"/f"	تقديم صفحة	12

الجدول ١٠-١

ولبيان أهمية هذه الشفرات ، خذ المثال التالي:

"first line\n second line"

لو طبع هذا النص (الثابت الرمزي) فانه سيظهر في سطرين متتالين على النحو التالي:

First line
Second line

ومن الجدير بالذكر ، أن أهم تطبيقات المعطيات الرمزية واستعمالاتها ، هو معالجة النصوص ، وما يستحق التسجيل والاهتمام ، انه يمكن إجراء عمليات على المعطيات الرمزية.

الملاحظات والتعليقات في c++ Comments

تستعمل سائر لغات البرمجة جملا للتعليقات والملاحظات ، وكذلك لغة C++ مثلا الجملة التالية:

```
10 rem this is Islam
```

هي جملة ملاحظ في لغة بيسك ، تقابلها جملة تعليق التالية في لغة C++:

```
// this is Islam
```

التي توضح بعد شرطتين (خطين مائلين)
وتستعمل جمل التعليق ، في أي مكان من البرنامج لإبداء ملاحظة ما ، عند سطر ما في البرنامج ، ولا تعد جملة تنفيذية ، بمعنى أنها لو حذفت من البرنامج ، لا يؤثر فيه ذلك شيئا ، وعادتا ما يتجاهلها المبرمجين .

مثال: لاحظ جمل التعليق التالية:

```
this is comment//
/*an example on comment in c++ language */
/*main() /* start your program
{
int x,y,z; //this line declares type of x,y,z
}
```

ومن الجدير بالذكر هنا ، ما يأتي:

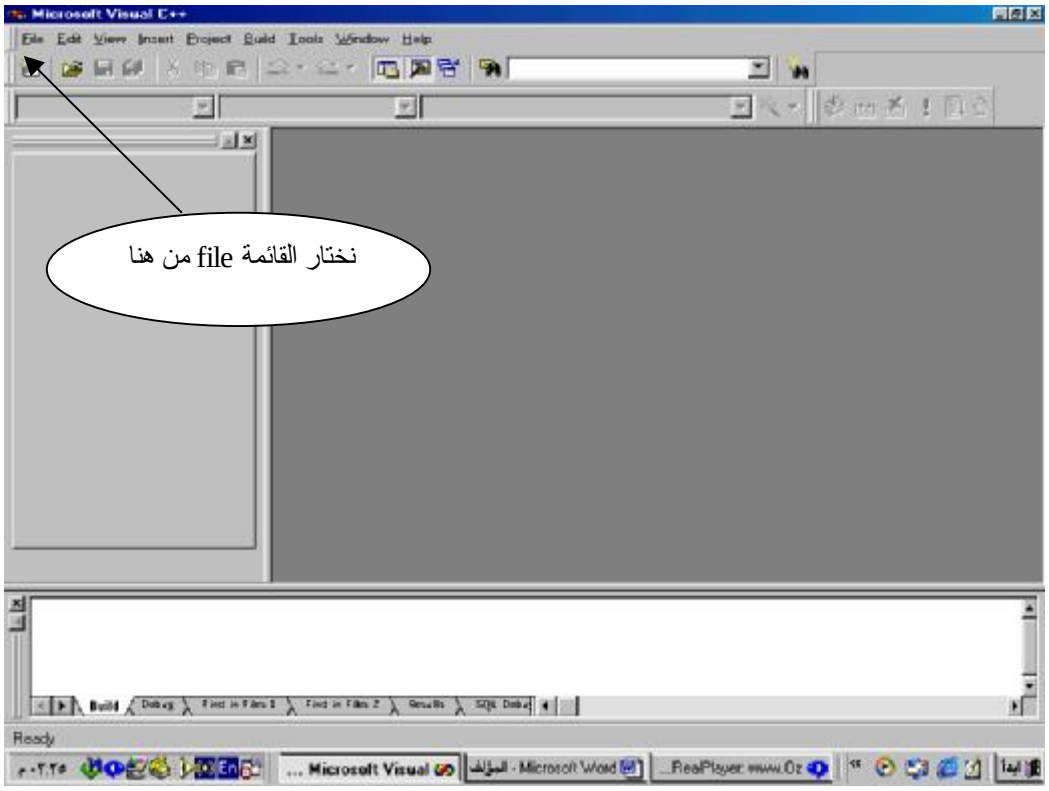
لا يترك أي فراغ بين الشرطة / والنجمة * من جهتي جملة التعليق.

١- يقوم مترجم c++ بإهمال النصوص المستعملة في جمل التعليق ، أي أنها لا تنفذ ، بل هي جمل توضيحية تظهر مع قائمة البرنامج أو سطورا فقط .

٢- يمكن وضع جملة الملاحظة والتعليق في أي مكان من البرنامج ، ما عدا وسط اسم تعريفي identifier ، أو كلمة محجوزة keyword .

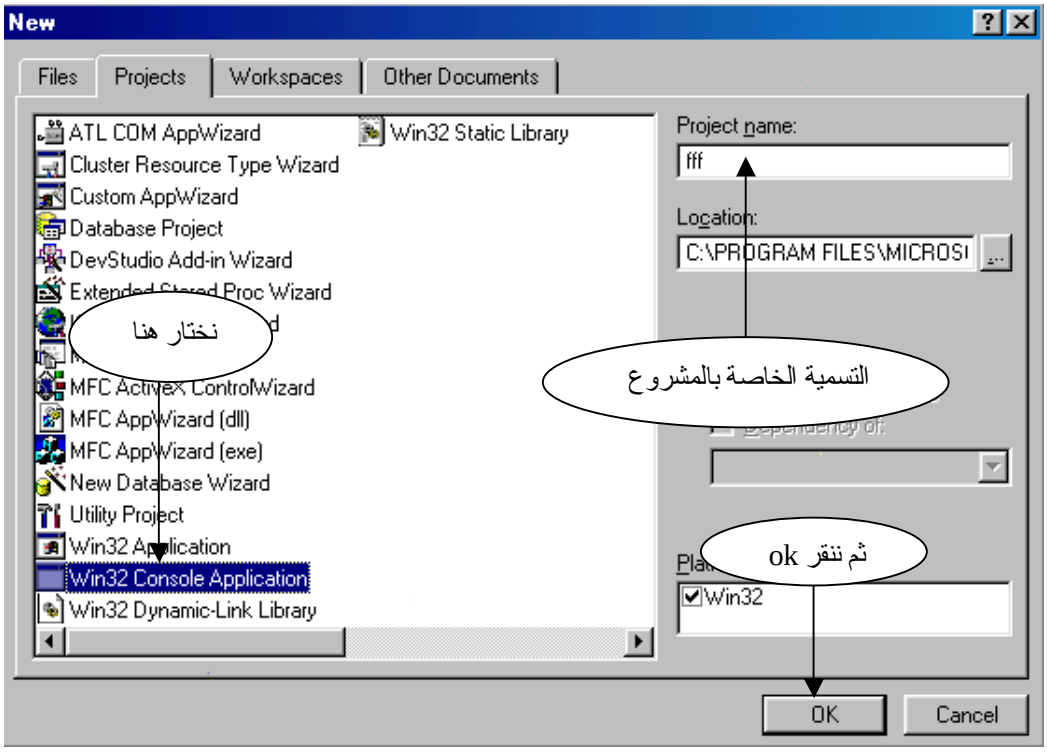
تشغيل visual c++6.0
Run visual c++6.0

لتشغيل برنامج فيجوال سي ++ نتبع التالي:
أبدا Ù البرامج Ù visualc++6.0
ثم بعد ذلك ستظهر لنا الشاشة التالية:

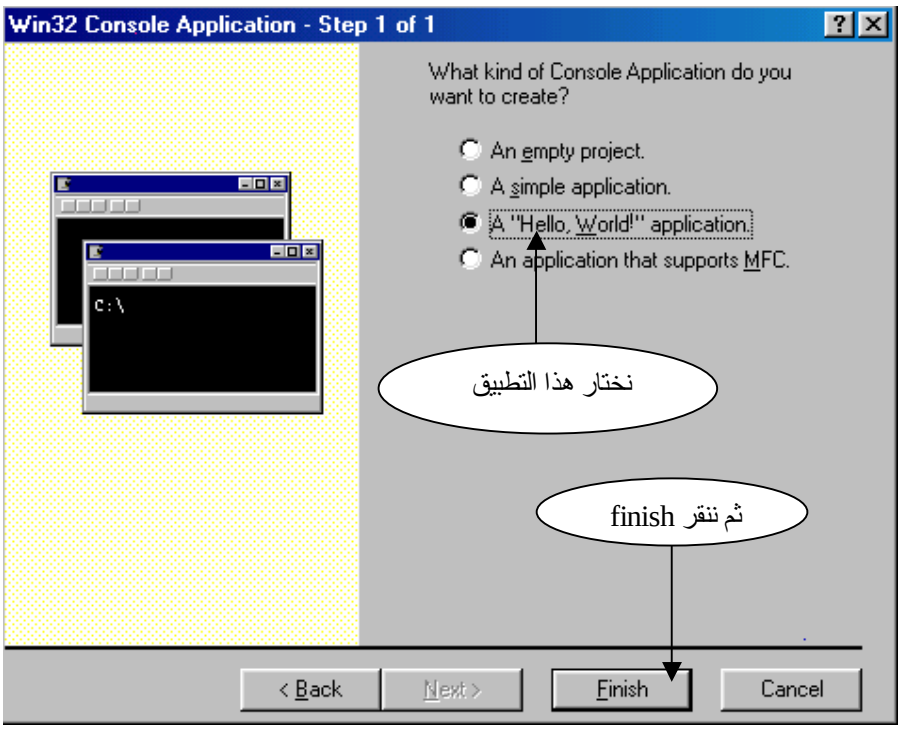


نختار من هذه الشاشة كما هو واضح القائمة File ثم بعد ذلك نختار من القائمة New لتظهر لنا الشاشة التالية..

نطبق ما يوجد بالصورة بالأسفل ثم نختار موافق..



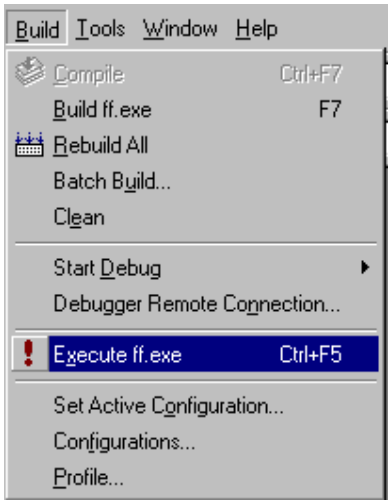
نختار التطبيق الموجود بالأسفل ثم نختار إنهاء كما هو موضح بالأسفل..



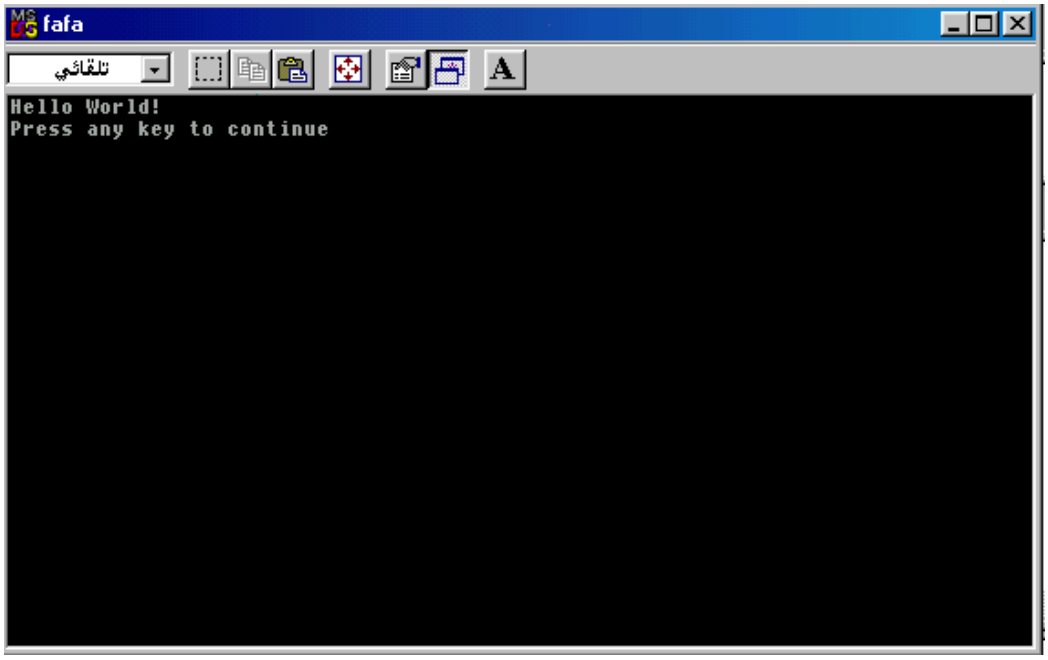
الآن ستظهر لنا شاشة الكود ونلاحظ بالأسفل الشاشة..

```
// fafa.cpp : Defines the entry point for the console application.
//

#include "stdafx.h"

int main(int argc, char* argv[])
{
    printf("Hello World!\n");
    return 0;
}
```

بعد ذلك نختار من القائمة Build ثم نختار Execute .exe ليطبق لنا المشروع ..
أو بالنقر من لوحة المفاتيح على الزر F5 .



طبعا بعد تنفيذ البرنامج ستظهر لنا النتائج كما في الشكل التالي:



طبعا أعزائي هذا البرنامج الصغير هو برنامج تلاحظون أن الكود تبعه خاص بلغة C الام وليس c++ لكن ما نعمل عليه هو مترجم يقبل اللغتين معا والمترجم هو Visual C++ ، ونلاحظ انه تم طباعة عبارة Hello World! وهي نتيجة تنفيذ الدالة printf() الموجودة في سطور البرنامج.

أساليب الإدخال والإخراج
Input/output Techniques

مقدمة introduction

يتناول هذا الجزء أساليب إدخال القيم الحسابية والرمزية ، وأساليب إخراج وطباعة نتائج البرامج والتعابير الحسابية والرمزية ، وطباعة المعطيات المختلفة حسب الحاجة.

لقد تعودنا في لغة بيسك ، أن نستعمل دوال مبنية وجاهزة عند الطلب للقيام بالإدخال مثل (input أو read) أو بالإخراج مثل (print) ، وفي هذا الصدد ، فان لغة C++ ، تتعامل مع الإدخال والإخراج ، بطريقة مختلفة، حيث توفر اللغة ، عددا كبيرا من دوال الإخراج والإدخال ، حيث يمكن للمبرمج أن يستدعيها ، ويستفيد منها ، حسب نوع المعطيات والمتغيرات ، كيفما يناسبه ، وسوف نورد أن شاء الله في هذا الفصل أهم هذه الدوال واشهرها لـ C++ .

الإدخال والإخراج input\output

توفر لغة C++ ، مجموعة من الدوال والروتينيات المعرفة ضمن Iostream مثل cout للإخراج و cin للإدخال وسوف نعرف الملف iostream.h

الملف Iostream.h يعني:
io : مختصر لـ input/output آي الإدخال والإخراج.
Stream : مكتبة قياسية خاصة بالإخراج والإدخال الـخ..
H : header file آي الملف الدليل.

مثال ١:
إذا أردت طباعة العدد 100 في لغة بيسك فالجملة:

Print 100

تؤدي عملية الطباعة ، أما في لغة C++ فان الدالة التالية تعمل ذلك:

```
#include  " stdafx.h"
#include " iostream.h"
main ()
{
cout<<100;
return0;
}
```

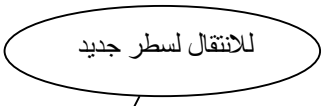
تـؤدي إلـى طـباعة العـدد 100 حيـث cout اسـم وحـدة الإخـراج القياسـي والأداة<< تـؤدي إلـى إرسـال العـدد 100 إلـى وحـدة الإخـراج ، أن هـذا الأسـلوب الجديـد فـي الإخراج يختلف عما في لغة c .

طباعة النصوص (الثوابت الرمزية)

مثال ٢:

تأمل قطعة البرنامج التالي:

```
#include  " stdafx.h"
#include " iostream.h"
main ()
{
cout<<"smoking is dangerous \n";;
return0;
}
```



بعبارة أخرى \n إيعاز للانتقال إلى سطر جديد ، وقد يمكن استخدام الدالة endl بدلا من \n وكما يلي:

```
#include  " stdafx.h"
#include " iostream.h"
main ()
{
cout<<" smoking is dangerous"<<endl;
return0;
}
```

وعند تنفيذ البرنامج يطبع الناتج التالي:

smoking is dangerous

مثال ٣:
للاستفادة من إمكانيات الإيعاز \n في عمليات الطباعة :تأمل البرنامج التالي:

```
#include  " stdafx.h"
#include " iostream.h"
main ()
{
cout<<"matrix";
cout<<"matrix \n";
cout<<"matrix \n\n";
cout<<"matrix \n\n\n";
cout<<"matrix";
return0;
}
```

عند تنفيذ البرنامج ترى الطباعة التالية على الشاشة:
السطر الأول Ù matrixmatrix
السطر الثاني Ù matrix
السطر الثالث Ù سطر فارغ
السطر الرابع Ù matrix
السطر الخامس Ù سطر فارغ
السطر السادس Ù سطر فارغ

نلاحظ في هذا البرنامج:
١- انه يتم الانتقال من السطر الأول بعد طباعة matrix إلى السطر الثاني لعدم وجود إيعاز الانتقال \n ، ولذا فان جملة الطابعة التالية ظهرت نتائجها في ا لسطر الأول نفسه ، متصلة بطاعة matrix الأولى ، وينتقل المؤشر الضوئي إلى سطر جديد لوجود إيعاز \n .
٢- يتم تنفيذ جملة الطباعة الثالثة في السطر الجديد (الثاني) ، ويتم الانتقال إلى السطر الرابع قفزا عن السطر الثالث ، وذلك لوجود الإيعاز \n\n حيث يقوم كل إيعاز \n بنقل المؤشر الضوئي سطرا واحدا ، وفي السطر الرابع تطبع جملة الطباعة الرابعة ، ويتم بعدها الانتقال إلى السطر السابع فورا حسب الإيعاز \n\n\n .

طباعة القيم العددية

مثال٤:

يقوم البرنامج التالي بطاعة العدد 446 كقيمة صحيحة على شاشة الحاسوب:

```
#include  " stdafx.h"
#include " iostream.h"
main ()
{
cout<<446;
return0;
}
```

عند الطباعة يظهر لنا التالي:

446

مثال٥:

برنامج C++ ، التالي يطبع القيمة الحقيقة 10.5:

```
#include  " stdafx.h"
#include " iostream.h"
main ()
{
cout<<10.5;
return0;
}
```

عند الطباعة يظهر التالي:

10.5

مثال٦:
انظر ماذا يفعل برنامج C++ التالي:

```
#include  " stdafx.h"
#include " iostream.h"
main ()
{
int a=100;
cout<<a;
return0;
}
```

عند الطباعة يظهر لنا التالي:

100

مثال٧:
البرنامج التالي يقوم بطباعة قيمة متغير حقيقي:

```
#include  " stdafx.h"
#include " iostream.h"
main ()
{
float x;
x=4.25
cout<<x;
return0;
}
```

عند الطباعة سيظهر لنا التالي:

4.25

مثال٨:
إذا تطلب الأمر طباعة المتغيرين a الصحيح ، و x الحقيقي الواردين في المثالين السابقين ، في برنامج واحد ، فالبرنامج سيكون على النحو التالي:

```
#include  " stdafx.h"
#include " iostream.h"
main ()
{
int=100;
float x=4.25
cout<<a<<x;
return0;
}
```

وستظهر نتائج هذا البرنامج كما طلبنا (الصحيح يسبق الحقيقي) ، هكذا:

100 4.25

طـباعة القـيم العدديـة والرمزية (النصوص) في جملة واحدة

مثال٩:

سوف نقوم في هذا المثال بطباعة قيم عددية ونصية مع البعض كالتالي:

```
#include  “ stdafx.h”
#include “ iostream.h”
main ()
{
int a=100;
cout<<“a=”<<a;
return0;
}
```

عند الطباعة يكون الناتج كالتالي:

A=100

مثال١٠:

ماذا إذا أردنا طباعة عدد صحيح وحقيقي مع نصوص بنفس الوقت:

```
#include  “ stdafx.h”
#include “ iostream.h”
main ()
{
int=100;
float x=4.25
cout<<“a=”<<a<<“x=”<<x;
return0;
}
```

وعند الطباعة سيظهر لنا التالي:

A=100 x=4.25

مثال ١١:
إذا أردنا أن تظهر نتائج المثال السابق في سطرين بدلا من سطر واحد ، فجملة الطباعة ستكون كالتالي:

```
#include  " stdafx.h"
#include " iostream.h"
main ()
{
int a=100;
float x=4.25
cout<<"a="<<a<<"\n"<<"x="<<x;
return0;
}
```

وتكون نتيجة الطباعة على الصورة التالية:

A=100
X=4.25

الإدخال بلغة C++ Streams

يتناول هذا المبدأ معالجة الإدخال حيث يعد استخدام streams افضل من دوال الإدخال للغة C .
وصيغة الجملة كالتالي:

```
Cin>>a;
```

ونشاهد أنها عكس عملية الإخراج حيث الإخراج >> أما الإدخال << .
وتستخدم هذه الجملة لإدخال قيم عبر لوحة المفاتيح للمتغيرات في الذاكرة ، ويتم تعيين قيمة المتغير في الذاكرة باستخدام لوحة المفاتيح .

ملاحظة/
لا يجوز أن نستخدم المتغير قبل تعريفة.

مثال صحيح:

```
Int x;
Cin>>x;
```

مثال خاطئ:

```
Cin>>x;
```

مثال ١٢:
سوف نقوم بإدخال عدد صحيح في هذا التطبيق ثم نقوم بطباعته كالتالي:

```
#include  " stdafx.h"
#include " iostream.h"
main ()
{
int=a;
cin>>a;
cout<<a;
return0;
}
```

نلاحظ في هذا المثال أننا قمنا بتعريف المتغير a بأنه عدد صحيح بعد ذلك عند تنفيذ البرنامج سيطلب منا إدخال عدد سندخل العدد 10 مثلا عند ذلك سيكون الناتج كالتالي:

مثال١٣:
اكتب برنامجا لإدخال عمرك ثم طباعته ، وطباعه نصف وضعفه؟

```
#include  " stdafx.h"
#include " iostream.h"
main ()
{
int a;
cin>>a;
cout<<a;
cout<<a/2;
cout<<a*2;
return0;
}
```

في المثال أعلاه قمنا أولا بتعريف المتغير كالتالي:

Int a;

ثم بعد ذلك طلب منا إدخال العمر:

عند الطلب سندخل مثلا 20

Cin>>a

وبعدها قمنا بطباعة العمر:

Cout<<a;

ثم قمنا حسب المطلوب بطباعة نصف العمر:

Cout<<a/2;

ثم قمنا حسب المطلوب الأخير بطباعة ضعف العمر:

Cout<<a*2;

لتكون النتيجة النهائية كالتالي:

20 10 40

مثال ١٤:
اكتب برنامجا لإدخال عدد ما وليكن العدد 7 ومن ثم طباعة جدول الضرب له؟

```
#include  " stdafx.h"
#include " iostream.h"
main ()
{
int a;
cin>>a;
cout<<a*1;
cout<<a*2;
cout<<a*3;
cout<<a*4;
cout<<a*5;
cout<<a*6;
cout<<a*7;
cout<<a*8;
cout<<a*9;
cout<<a*10;
return0;
}
```

عند طلب إدخال قيمة ندخل
الرقم 7 حسب طلب السؤال..

عند تنفيذ البرنامج ستكون النتيجة كالتالي:

7 14 21 28 35 42 49 56 63 70

مثال ١٥:
اكتب برنامج لإدخال ثلاث علامات لطالب 30 25 40 وطباعة معدل العلامات؟

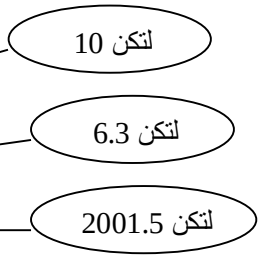
```
#include  " stdafx.h"
#include " iostream.h"
main ()
{
int a,b,c;
cin>>a>>b>>c;
cout<<(a+b+c)/3;
return0;
}
```

ندخل حسب المطلوب 25 40
30

نلاحظ أن في جمل الإخراج قمنا بكتابة قاعدة وهي جمع الثلاث أعداد مع بعضها ثم قسمتها على عددها وهي قاعدة معروفة لإظهار المعدل..
وسف يكون الناتج كالتالي:

مثال١٦:
سنحاول الآن إدخال ثلاث قيم عددية ، ومن ثم طباعتها:

```
#include  " stdafx.h"
#include " iostream.h"
main ()
{
int a;
float x;
double z;
cin>>a>>x>>z; // input all data
cout<<endl;
cout<<"\n print the following items";
cout<<endl;
cout<<a<<"\n"<<x<<"\n"<<z; // output all data
cout<<"\n this is the end"<<endl;
return0;
}
```



نلاحظ في السؤال أعلاه أننا قمنا بإدخال ثلاث قيم مختلفة من حيث النوع وأيضا قمن باستخدام التعليقات وهي لا تؤثر في البرنامج فقط للتوضيح وهي التعليق:

// input all data

// output all data

لنوضح للمستخدم أين الإدخال والإخراج ..

وسوف يكون الناتج كالتالي:

print the following items
10
6.3
2001.5
this is the end

جمل التحكم والشرط والتكرار
ProgramControl,Conditional&Loop Statements

مقدمة introduction

قد نحتاج أن ننتقل من سطر إلى آخر في برنامج C++ ، وقد نحتاج أن نقوم بتنفيذ بعض الجمل عند تحقق بعض الشروط ، وقد نحتاج أن نكرر عملية من عمليات الإدخال أو الإخراج أو الحساب عددا من المرات ، وقد نحتاج أن نبني شبكة من توزيع الأوامر على عدد من سطور البرنامج ، حسب مقتضيات بعض الأحوال ، وحسبما تقتضيه طبيعة المسألة ، في هذه الحالات: نحتاج أن نتعلم أساليب الشرط ، وأساليب التكرار ، وكيفية التحكم في مسار البرنامج ، وتعد أساليب الشرط والتحكم والتكرار بمثابة القلب في جسم لغات البرمجة ، وبدونها لا يمكن تنظيم آي برنامج.

وتوفر لغة C++ للمبرمج عددا من الأساليب والدوال الفعالة ، المتعلقة بهذا الشأن ، وتمتاز هذه الأساليب بأنها أساليب بنائية أو بنيوية structured آي يمكن تنظيم عمليات التحكم والتكرار فيها ، بأسلوب ذاتي من بداية العمليات وحتى نهايتها دون تدخل من المبرمج أثناء هذه ا لعمليات ، للأشراف على التوجيه والتخطيط لكل خطوه من خطوات البرنامج ، ويعرف بعض الخبراء والمختصين البرمجة البنيوية: أنها البرمجة التي لا تستعمل جملة الانتقال GOTO ، لتوجيه البرنامج في كل خطوة ، ومع ذلك فان لغة C++ ، توفر جملة الانتقال هذه ، لكنها لا تستعمل إلا للضرورة.

وحيث أن جواب الشرط أما أن يكون صوابا true أو زائفا false فان لغة C++ ، تعطي الحالة الصائبة قيمة عددية تختلف عن الصفر ، وتعطي قيمة صفر للحالة الزائفة (عند عدم تحقق الشرط أو الشروط) ولذا فان لغة C++ توفر مرونة كبيرة في استخدام عدد كبير من الدوال ، وفي توجيه البرنامج بطريقة فعالة وفائقة.

الجمل الشرطية

تتعامل لغة C++ مع ثلاثة أنواع من جمل الشرط وهي:

١- جملة إذا الشرطية وأخواتها if statements

٢- جملة التوزيع switch statement

٣- جملة أداة الشرط ?

جملة الشرط إذا وأخواتها if statements

- جملة الشرط إذا وأخواتها if statements

تأخذ هذه الجملة الشكل العام التالي:

```
If (condition) statement1;
```

تقوم جملة إذا الشرطية هنا ، بنقل تسلسل تنفيذ البرنامج إلى الجملة (أو سلسلة الجملة) statement1 عندما يتحقق الشرط (أو الشروط) condition وتكون نتيجته true ، وإذا لم يتحقق الشرط ، أي تكون النتيجة false ، فان التنفيذ ينتقل فورا إلى الجملة (أو سلسلة الجمل) statment2 ويعد استعمال else في C++ اختياريا ، أي يمكن حذفها دون أن تتأثر الجملة الشرطية تركيبا واداء ويكون شكلها العام على النحو التالي:

```
If (condition) statement1;
Else statment2;
```

وفي هذه الحالة ستنفذ الجملة statement1 أن تحقق الشرط condition وإلا فان التنفيذ ينتقل إلى سطر C++ التالي لجملة if .

الصيغة الأولى

وتأخذ البنية العاملة If (condition) statement1

مثال ١:

اكتب برنامجا بلغة C++ لإظهار العبارة x is positive على شاشة العرض؟

```
#include  " stdafx.h"
#include " iostream.h"
main ()
{
int x=5;
if (x>0)
cout<<x<<" Saudi";
return0;
}
```

في هذا المثال ستظهر الجملة Saudi على الشاشة لان الشرط (x>0) متحقق فالخمسة بالطبع اكبر من العدد صفر فالنتيجة كالتالي:

Saudi

مثال ٢:

اكتب برنامج C++ التالي ليحسب القيمة المطلقة لـ Y المعرفة على النحو التالي:

$$Y=|x|=\begin{cases} x;x>=0 \\ -x;x<0 \end{cases}$$

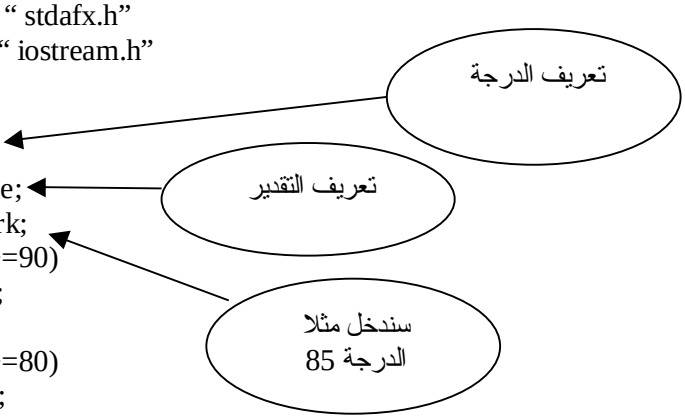
```
#include  " stdafx.h"
#include " iostream.h"
main ()
{
int x;
cin>>x;
if (x>=0) cout<<x;
else cout<<-x;
return0;
}
```

في المثال أعلاه سوف ندخل الرقم 10 لتكون النتيجة :

10

مثال٣:
قم بإنشاء برنامج لإدخال علامة طالب فإذا كانت العلامة اكبر أو تساوي 90 فالتقدير (A) أما إذا كانت اكبر أو تساوي 80 فالتقدير (B) أما إذا كانت اكبر أو تساوي 70 فالتقدير (C) أما إذا كانت اكبر أو تساوي 60 فالتقدير (D)أما إذا كانت اكبر أو تساوي 50 فالتقدير (E) ما عدا ذلك فالتقدير (F)؟

```
#include  " stdafx.h"
#include " iostream.h"
main ()
{
int mark;
char grade;
cin>>mark;
if (mark>=90)
grade='a';
else
if (mark>=80)
grade='b';
else
if (mark>=70)
grade='c';
else
if (mark>=60)
grade='d';
else
if (mark>=50)
grade='e';
else
grade='f';
cout<<grade;
return0;
}
```



في المثال أعلاه قمنا بتعريف المتغير mark بأنه عدد صحيح ثم قمنا بعد ذلك بتعريف المتغير grade بأنه قيمة نصيه وهو التقدير.
طبعا قمنا بإدخال الدرجة وهي 85 سوف تكون العلامة كالتالي:

B

الصيغة الثانية

وتأخذ البنية العاملة لجملة إذا .... وإلا(if..else) الشكل العام التالي:

```
If (condition)
{
statmenet1;
}
else
{
statmenet1;
}
```

مثال ٤:

سوف نطبق المثال السابق (3) لكن بالشكل (if..else) أعلاه كما يلي:

```
#include  " stdafx.h"
#include " iostream.h"
main ()
{
int mark;
char grade;
cin>>mark;
if ( mark>=90){
grade='A';
{
else
if (mark>=80){
grade='B';
{
else
if (mark>=70){
grade='C';
{
else
if (mark>=60){
grade='D';
{
else
if (mark>=50){
grade='E';
}
cout<<grade;
return0;
}
```

إذا الأولى وجوابها

إذا الثانية وجوابها

إذا الثالثة وجوابها

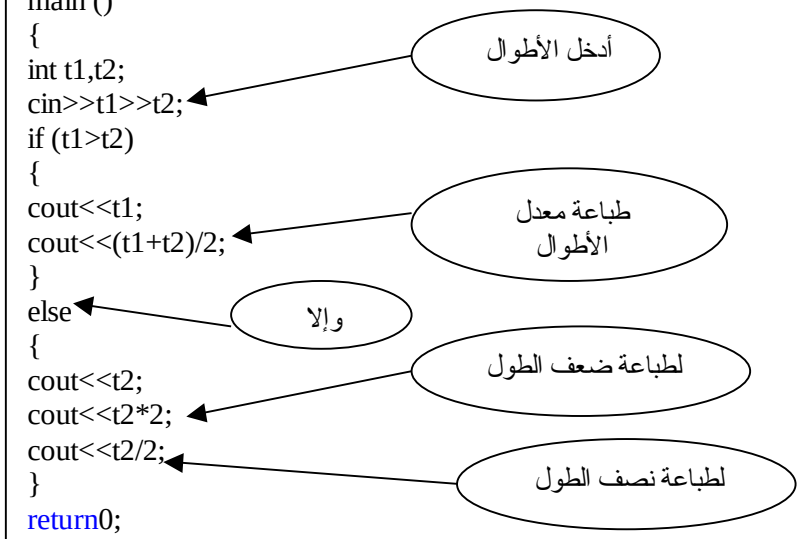
إذا الرابعة وجوابها

وإلا فالنتيجة الباقية
هى الخامسة

لطباعة التقدير

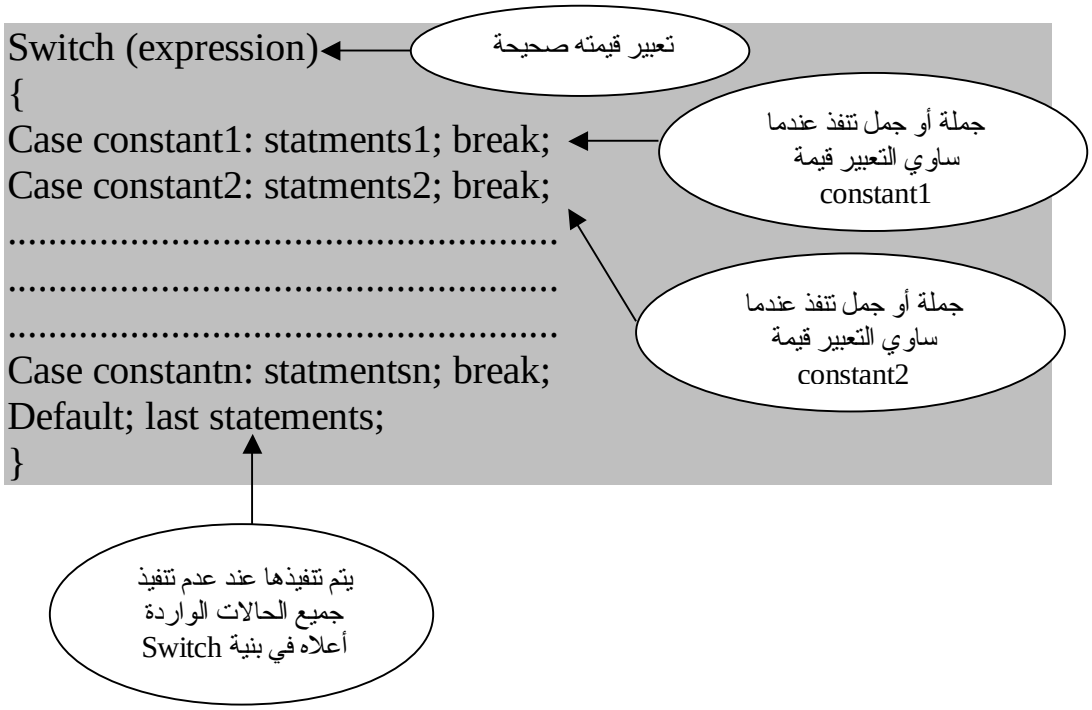
مثال٥:
اكتب برنامجا لإدخال طولك وطول زميلك ، وإذا كان طولك اكبر من طول زميلك
اطبع طولك ، واحسب معدل الأطوال ، ثم اطبعه وألا
اطبع طول زميلك ، واطبع ضعف الطول ونصف الطول؟
الحل/
سنرمز لطولك t1 وسنرمز لطول زميلك t2

```
#include  " stdafx.h"
#include " iostream.h"
main ()
{
int t1,t2;
cin>>t1>>t2;
if (t1>t2)
{
cout<<t1;
cout<<(t1+t2)/2;
}
else
{
cout<<t2;
cout<<t2*2;
cout<<t2/2;
}
return0;
}
```



جملة التوزيع switch statement

تأخذ جملة Switch الشكل العام التالي في لغة c++ :



مثال٦:

```
#include  " stdafx.h"
#include " iostream.h"
void main()
{
int s1;
s1=2;
switch (s1)
{
case 2 :cout<<"y";
          break;
case 3: cout<<"x";
           break;
case 4: cout<<"m";
           break;
default: cout<<"w";
}
 }
```

و النتيجة:

y

جملة أداة الشرط ؟

وهي أداة سريعة مكافئة لبنية إذا ... وإلا ، وقد مر معنا كيفية استعمالها في أول الكتاب وسوف نورد هنا صورتها العامة:

```
Variable=(condition)? Result:result2;
```

ومعناها: انه يتم تنفيذ النتيجة الأولى result1 عندما يكون جواب الشرط condition متحققا (true) ، وإلا فيتم تنفيذ النتيجة الثانية result2 عندما يكون جواب الشرط (false) .

مثال٧:

```
#include  " stdafx.h"
#include " iostream.h"
void main()
{
int a,b;
a=5;
if (a>1) b=10;
else
b=20;
cout<<b;
}
```

ومعناها أن b تأخذ القيمة 10 إذا تحقق الشرط a>1 وتأخذ القيمة 20 إذا لم يتحقق الشرط نفسه .

والنتيجة:

10

التكرار وحلقات التكرار Repetition and Loops

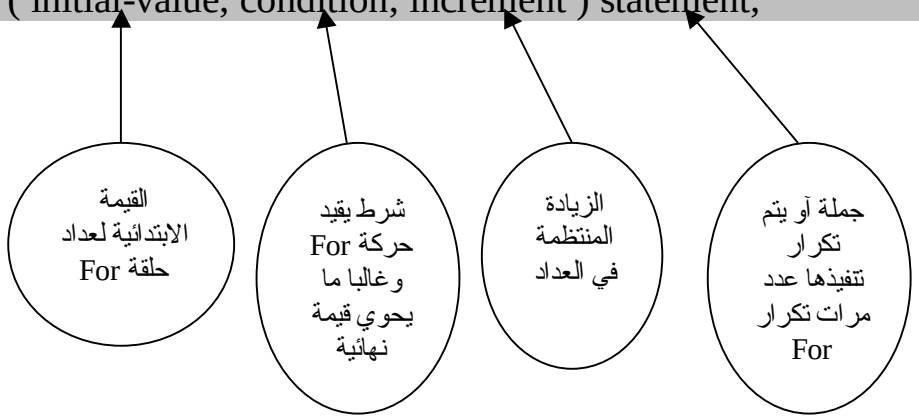
توفر لغة C++ ، كسائر لغات البرمجة ، عددا من أساليب التكرار المشروط ، وغير المشروط ومن هذه الأساليب:

أسلوب التكرار باستعمال حلقة For

يمتلك أسلوب التكرار باستعمال for قوة ومرونة ، لا تتوفران في غيرها من اللغات.

الصيغة العامة الأولى

```
For ( initial-value; condition; increment ) statement;
```



تقوم جملة For هنا مبتدئة بقيمة العداد الابتدائية بتنفيذ الجملة statement1أول مرة ، وفي المرة التالية تزداد القيمة الابتدائية للعداد بمقدار الزيادة ثم تنفذ جملة statement1 مرة ثانية .. وهكذا حتى يستكمل الشرطconditionأمر إنهاء عمليات التكرار والخروج من حلقة For ، والأمثلة التالي توضح كيفية استعمال حلقات التكرار بجملة For:

مثال٨:

```
#include  " stdafx.h"
#include " iostream.h"
main ()
{
char x;
for (x=1;x<=10;x=x+1)
cout<<"Islam";
cout<<endl;
return0;
}
```

القيمة التنازلية للعداد

الزيادة بمقدار 1

التكرار إلى 10

والنتيجة كالتالي:

Islam Islam Islam Islam Islam Islam Islam Islam Islam Islam

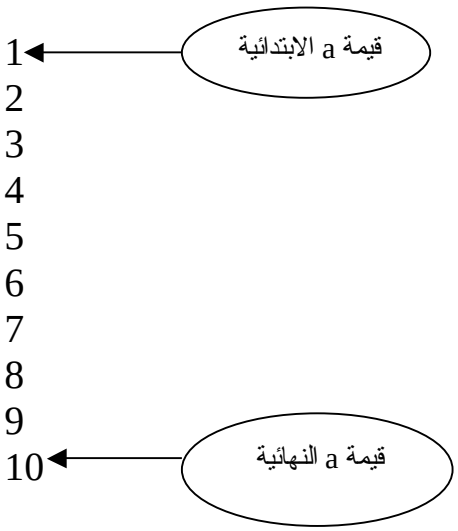
نلاحظ هنا انه تم تكرار كلمة Islam 10 مرات بداية من القيمة 1 Ù 10

مثال٩:

اكتب برنامجا لطباعة قيمة العداد من 1 إلى 10؟

```
#include  " stdafx.h"
#include " iostream.h"
main ()
{
int a;
for (a=1;a<=10;++a)
cout<<a<<endl;
return0;
}
```

وتكون نتائج الطباعة على الشاشة هكذا:



مثال ١٠:
اكتب برنامجا لطباعة الأعداد الفردية من 1 إلى 15؟

```
#include  " stdafx.h"
#include " iostream.h"
main ()
{
int a;
for (a=1;a<=15;a=a+2)
cout<<a<<endl;
return0;
}
```

ومن الملاحظ أننا جعلنا قيمة الزيادة 2 وليس 1 لانه طلب أعداد فردية بداية بالقيمة 1 وحتى 15
والنتيجة كالتالي:

1
3
5
7
9
11
13

الصيغة العامة الثانية

```
For ( initial-value; condition; increment )
{
statement;
}
```

جملة أو اكثر

شاهد الأمثلة التالية لنتعرف اكثر على الصيغة أعلاه:

مثال ١١:

```
#include  " stdafx.h"
#include " iostream.h"
main ()
{ int x,y,z;
y=-4;
for(x=1;x>y;x=x-2)
}
z=x;
cout<<x<<endl;
{
return0;
}
```

اكثر من جملة بين القطعتين Block

والناتج سوف يكون التالي:

1
-1
-3

مثال ١٢:

```
#include  " stdafx.h"
#include " iostream.h"
main ()
{ int a,b,c,x;
a=6;
b=1;
c=3;
for (x=11;(a*c)>(x*b);x--)
{
x=x+3;
c=c-2;
cout<<x<<"*";
}
return0;
}
```

القيمة الابتدائية

اكثر من جملة بين القطعتين Block

14*

مثال ١٣:

```
#include  " stdafx.h"
#include " iostream.h"
main ()
{ int a,c;
a=1;
c=3;
for (a=c;c;)
{
c--;
cout<<c<<endl;
}
return0;
}
```

تنقص من قيمة C قيمة 1

والناتج:

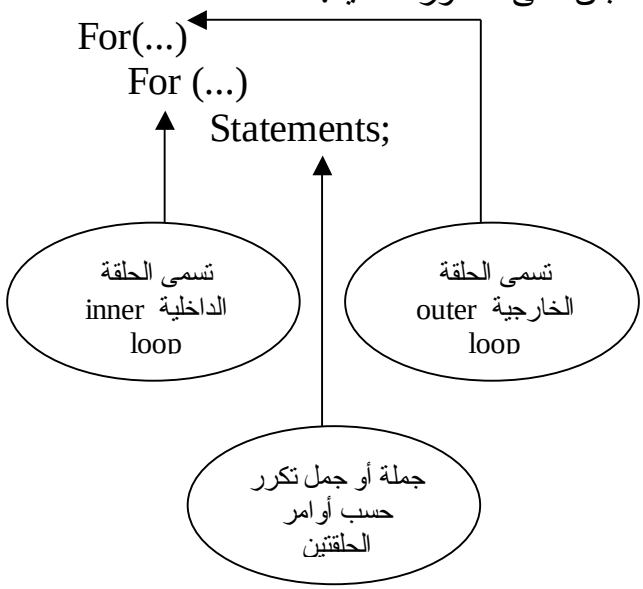
2
1
0

حلقـات الـتكرار المـتداخلة Nested (Multiple) for Loops

تأخذ صيغة حلقات التكرار المتداخلة الشكل العام التالي:

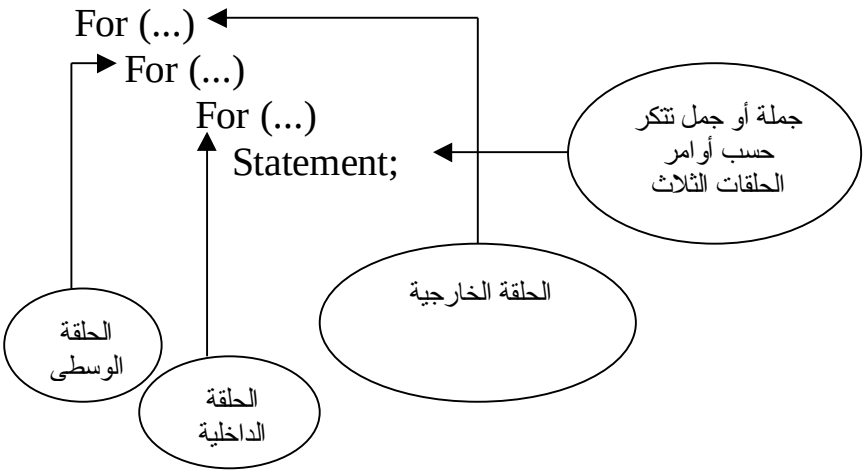
```
For (...)
    For (...)
        For (...)
         ……….
                ……….
                    Statements;
```

فلو أخذنا حالة حلقتين متداخلتين فانهما تكتبان على الصورة التالية:



وتكـون فـي هـذه الحالـة الجملـة (أو الجمـل) جـزءا مكـررا مرتـبطا بالحلقـة الداخلـية ، والحلقة الخارجية تتكرر حسب أوامر الحلقة الخارجية وهكذا ...

وفي حالة الثلاث حلقات المتداخلة ، فإنها تكتب على الصورة التالية:



مثال ١٤:

```
#include  " stdafx.h"
#include " iostream.h"
main ()
{
int i,j;
for (i=1;i<=3;++i)
for (j=1;j<=4;++j)
cout<<i*j;
return0;
}
```

الحلقة الخارجية تتكر 3 مرات

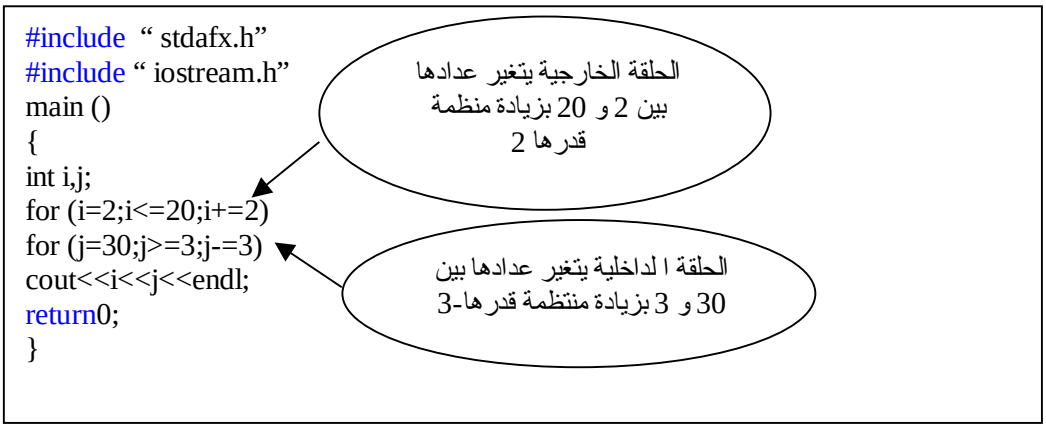
الحلقة الداخلية تتكرر 4 مرات لكل قيمة من قيم I أي مجموع 12 مرة

لاحظ هنا أن الحلقة الداخلية تتكرر ٤ مرات لكل قيمة من قيم I ، عداد الحلقة الخارجية ، وكذلك جملة cout<< ، وبما أن I ، تأخذ 3 قيم فان الحلقة الداخلية تتكرر 12 مرة ، أما الحلقة الخارجية فتكرر نفسها بنفسها فتتكر 12 مرة فقط.

والناتج:

1234246836912

مثال ١٥:



والناتج سيكون كبير لذلك سأعطيكم جواب الحل للسطر الأول والسطر الأخير وما بينهما لكم.

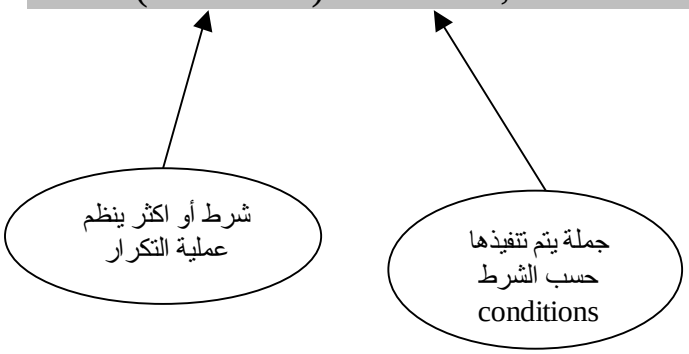
السطر الأول: 1612

السطر الأخير : 203

أسلوب التكرار باستعمال حلقة While & Do

أسلوب التكرار باستعمال حلقة while أسلوب آخر يماثل أسلوب حلقة for ، مع بعض الاختلافات البسيطة ، وهو أسلوب يثرى لغة C++ ، ويزدها قوة ومرونة ، والشكل العام لهذا الأسلوب:

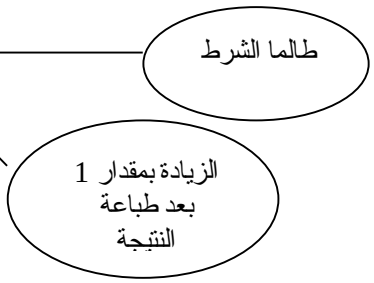
```
While (conditions) statements;
```



ومعنى حلقة التكرار while هو الأتي:
آي ما دام الشرط (أو الشروط) متحققا (وجوابه true) ، فيتم تكرار تنفيذ الجملة أو الجمل (statements) ، وينتقل تسلسل تنفيذ البرنامج إلى الجملة التي تلي حلقة while .
والأمثلة التالية توضح ذلك:

مثال١٦:

```
#include  " stdafx.h"
#include " iostream.h"
main ()
{ int a;
a=1;
while (a<3)
cout<<a++;
return0;
}
```

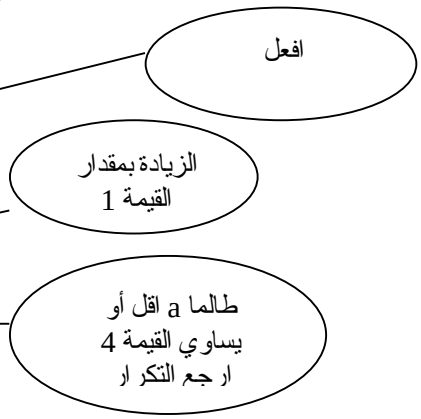


سوف يكون:

1 2

مثال١٧:

```
#include  " stdafx.h"
#include " iostream.h"
main ()
{ int a;
a=1;
do
{
cout<<"*";
a++;
}
while (a<=4)
return0;
}
```



والناتج

****

مثال١٨:

```
#include  " stdafx.h"
#include " iostream.h"
main ()
{ int x,y;
x=2;
y=-2;
do
{
cout<<x<<"-"<<endl;
x--;
y++;
}
else
cout<<"finish";
while(x!=y)
return0;
}
```

2--21--1

مثال١٩:
اكتب برنامجا يطلب من المستخدم إدخال قيمة عددية ، وطالما أن القيمة المدخلة + يطبع * على سطر جديد؟
الحل/

```
#include  " stdafx.h"
#include " iostream.h"
main ()
{ int a;
cout<<"please enter a number";
cin>>a;
while (a>=0)
{
cout<<"*"<<endl;
cout<<" please enter a number";
cin>>a;
}
return0;
}
```

طبعا قمنا بإدخال الرقم 10 والنتيجة ستكون:

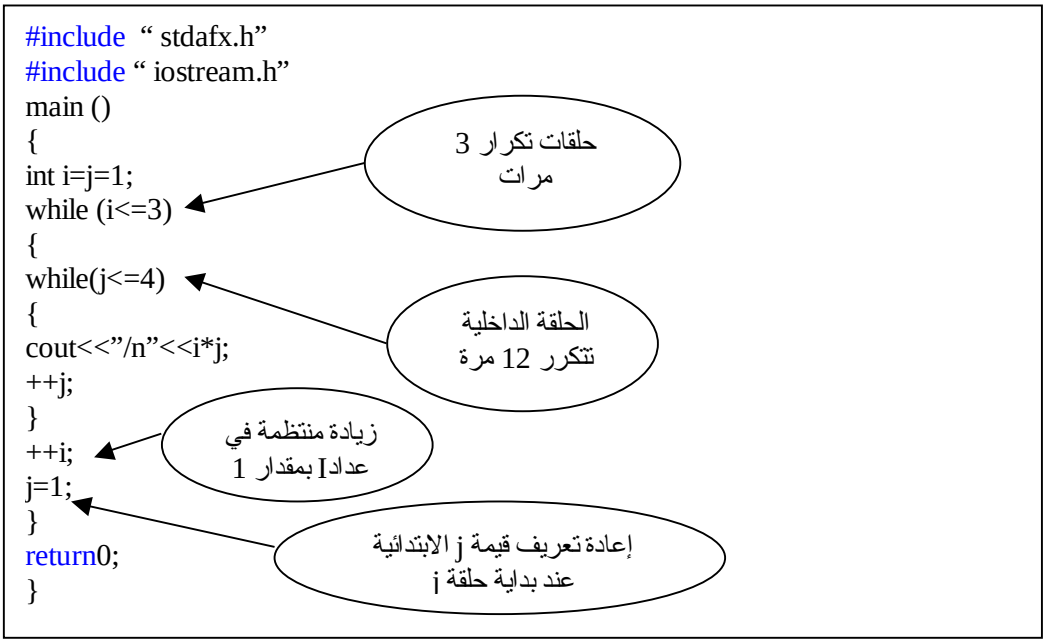
*

لان العدد 10 عدد موجب لكن حاول أن تدخل عدد سالب فلن يطبع لك شي لعدم تحقق الشرط ، ولا ننسى أخواني أننا وضعنا الجمل أو الجملة الخاصة بالطباعة بين القطع Block {} لانه وجد اكثر من جملة لذلك يجب وضع القطع لكن عند عدم وجود اكثر من سطر أو جمله كمثال (١٢) فلا يجوز وضعها.

حلقات While المتداخلة Nested While Loops

تشـبة حلقـات While المـتداخلة حلقـات for المـتداخلة ، فمـثلا خـذ حلقتـي الـتكرار المتداخلتين التاليتين:

مثال ٢٠:



جملة الإيقاف Break

من الاسم نستطيع أن نلاحظ أن وظيفة Break هي إيقاف بنية أو حلقة تكرار عند تحقق شرط أو شروط معينة ، وعند تنفيذها يتم القفز إلى سلسلة الجمل التالية للبنية أو حلقة التكرار ، وتستعمل Break أيضا في إيقاف حلقة التكرار لانهائي ، أو الخروج منها إلى الجمل التي تليها وكما في المثال التالي:

مثال ٢١:

```
#include  " stdafx.h"
#include " iostream.h"
main ()
{
int i;
for (i=1;i<100;++i)
{
cout<<i;
if (i= =10) break;
}
return0;
}
```

يوقف تنفيذ هذه الجملة حلقات التكرار عندما i=10 يصبح

وطبعا سيقوم بتنفيذ البرنامج حتى العدد 10
والنتائج:

12345678910

مثال ٢٢:

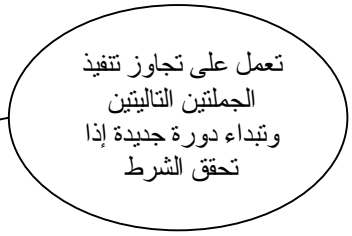
```
#include  “ stdafx.h”
#include “ iostream.h”
main ()
{
int number;
for (number=1; number<=100;++ number)
{
if (number%2) // for ood values
break;
else if (number%4)
break;
else if (number%6)
break;
else
cout<< number<<endl;
}
return0;
```

جملة الاستمرار continue

تعمل جملة الاستمرار continue على تجاوز تنفيذ بقية الجمل في التكرار خلال الدورة الحالية والانتقال إلى الدورة الثانية:

مثال٢٣:

```
#include  " stdafx.h"
#include " iostream.h"
main ()
{
int x,n;
do
{
cin>>x>>n;
if (n<1) continue;
cout<<x;
--n;
}
while (n<1);
return0;
}
```



مثال ٢٤:
تطبع جميع الأرقام من 1 إلى 100 ما عدا الأرقام التي تقسم على 2 و 4 و6 بدون باق:

```
#include  " stdafx.h"
#include " iostream.h"
main ()
{
int number;
for (number=1;number<=100;++number)
{
if (number%2)
continue;
else if (number%4)
continue ;
else if (number%6)
continue;
else
cout<<number<<endl;
}
return0;
}
```

الناتج:

12
24
36
48
60
72
84
96

حملة الخروج exit()

تعمل هذه الدالة على إيقاف (أو الخروج من) البرنامج في مكان منه ، وتشبة end في لغة بيسك ، وتكون قيمة الدالة صفرا exit(0) عندما يتم الخروج من البرنامج بنجاح وألا فان قيمة الدالة تكون exit(1) وتوقف البرنامج عند وجود خطأ أو نحو ذلك ، وفي هذه الحالة ، وتلك يعود البرنامج تنفيذه إلى نظام التشغيل operating system .

مثال٢٥:

```
#include  " stdafx.h"
#include " iostream.h"
main ()
{
chat ma;
cin>>ma;
if ( ma != 'A') exit(1);
cout<<"\n"<<ma;
return0;
}
```

جملة الانتقال goto

من المعروف أن معظم لغات البنيوية الحديثة ، تحرص ، في غالب الأحيان ألا تستعمل جملة goto من اجل التأكيد على المبرمج ، أن يتعلم برامجه بطريقة بنيوية structured ذاتية المداخل والمخارج ، والعمليات ، دون تدخل من المبرمج بقوله : اذهب goto من هنا ، أو اذهب من هناك آي أن البرنامج في هذه الحالة يعتمد على نفسه .

مثال٢٦:

```
#include  " stdafx.h"
#include " iostream.h"
main ()
{
int x;
cin>>x;
if (x<10) goto negative;
negative: cout<<"value is under 10"<<endl;
return0;
}
```

في هذا السؤال يطلب من المستخدم إدخال قيمة فإذا كانت القيمة اقل من 10 حسب الشرط فانه يعرض لك الرسالة value is under 10 .
أما إذا كانت اكبر من العدد 10 فانه يطبع العدد مباشرة من دون الذهاب للسطر الأخير لتحقق الشرط .

المتغيرات المرقمة والمصفوفات
Arrays and Matrices

مقدمة introduction

أن طرق التعامل مع أسماء المتغيرات والثوابت العددية والرمزية ، التي وردت في الفصول السابقة ، تعد صالحة للتعامل مع عدد محدود من هذه الثوابت والمتغيرات ، سواء في عمليات الإدخال والإخراج أو في العمليات الحسابية والمنطقية ، وعندما يصبح عدد المتغيرات كبيرا جدا ، تصبح تلك الطرق غير عملية ، فمثلا لو أردنا إدخال مائة قيمة للمتغيرات x2,x1- .... إلي x100 ، فكم الحيز المطلوب من البرنامج لعمليات الإدخال والإخراج والعمليات الحسابية والمنطقية لهذه المتغيرات ؟ هذا من جهة ، ومن جهة أخرى : فأننا نوفر مخزنا خاصا لكل متغير نتعامل معه ، أثناء تنفيذ البرنامج ، ولذلك لحفظ قيمته في مخون ، ومن ثم لاستعمال قيمته في عمليات أخرى تالية ، ومن ناحية ثالثة ، فان من الصعوبة بمكان ، بل من المستحيل استعمال اسم المتغير العددي أو الرمزي كمصفوفة ذات بعدين ، وثلاثة أبعاد ...الخ

للأسباب الثلاثة الواردة أعلاه ، جاءت فكرة استعمال متغير جماعي يضم تحت اسمه عددا من العناصر يسمى بالمتغير الرقمي subscripted variable ، ويتم ترقميه بين قوسين مربعين [ ] يوضع بينهما قيمة العداد المرقم subscript ، وقد نسمية الدليل index أحيانا ، ويمكننا تشبيه المتغير المرقم بقسم الهاتف لمؤسسة ما ، فهو مقسم واحد ، تنظم تحته عدد من الأرقام الفرعية للموظفين وكل رقم من هذه الأرقام مستقل ومتميز عن الأرقام الفرعية الأخرى ، وله مخزن خاص في الذاكرة ، الآن انه كغيره من الأرقام الفرعية تابع للرقم العام لمقسم المؤسسة ، كما يمكن تشبيه المتغير المرقم بالجيش الذي يعامل كاسم متغير واحد ، لكن يضم عددا كبيرا من العناصر ، فمثلا العناصر التالية :(من اليمين إلى اليسار):

A[n] ...a[2], a[1], a[0]

تابع للمتغير الجماعي a[ ]

وكل عنصر من هذه العناصر له عنوان في الذاكر address ، فالعنوان الأول يكون للعنصر الأول والثاني للثاني والثالث للثالث ... وهكذا.

ويستعمل المتغير الجماعي [المرقم] أو المصفوفة ، في لغة c++ وغيرها ، حجز جماعي مسبق في الذاكرة لجميع عناصره ، فلو كان يتبعه خمسون عنصرا ، فانه يحجز له 50 مخزنا ، على الأقل في الذاكرة .

من الفوائد المهمة للمتغيرات المرقمة والمصفوفات : هو استعمالها في الترتيب التصاعدي والتنازلي للعناصر والقيم المختلفة ، وعمليات ترتيب الأسماء الأبجدي

النصوص الرمزية ، وفي عمليات ضرب المصفوفات ، وإيجاد معكوس المصفوفة وعملياتها الأخرى ، وفي التحليل العددي ... الخ.

المتغير المرقم (المصفوفة) ذو البعد الواحد one-dimensional Array

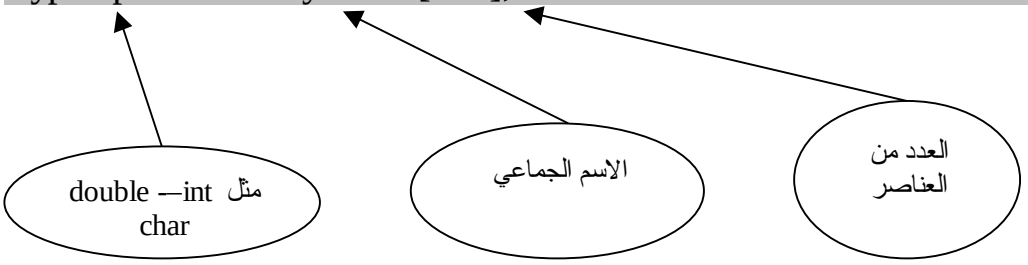
المتغير المرقم ذو البعد الواحد هو مصفوفة ذات بعد واحد أو متجه (vector) ويمثل في الجبر على النحو الأفقي [a1 a2 ...a3]

أو العمودي

A1
A2
:
:
:
a3

ويأخذ المرقم المتغير في c++ الشكل العام التالي:

```
Type-specifier array-name[size];
```



ويبدأ العداد المرقم عادة من الصفر ، آي أن العنصر الأول من المصفوفة a[] هو a[0] والثاني a[1] ... وهكذا فمثلا المصفوفة التالية:

```
Int a[20];
```

اسمها a ، وقد حجز لها 20 موقعا لعشرين عنصرا من النوع الصحيح .

والمصفوفة التالية:

```
Char name[15];
```

مصفوفة رمزية ، اسمها name يحجز لها خمسة عشر عنصرا من النوع الرمزي لها .

وهكذا...

مثال ١:
مثال على عملية إدخال ذاتي لقيم عناصر متغير مرقم (مصفوفة) ذي بعد واحد

```
#include  " stdafx.h"
#include " iostream.h"
main ()
{
int a[20];
int I;
for (I=0;I<20;++I)
a[I]=I+1;
return0;
}
```

في هذه الحالة يتم إدخال عشرين عنصرا من عناصر المصفوفة a
A[0]=1 عندما يكون I=0
A[1]=2 عندما يكون I=1
…
…
…
a[19]=20 عندما يكون I=19

مثال ٢:
مثال على عمليات إدخال ، وحساب ، وعمليات طباعة عناصر مصفوفة:

```
#include  " stdafx.h"
#include " iostream.h"
main ()
{
int x[5], y[5];
int I;
for (I=0;I<5;++I)
{
x[I]=I;
y[I]=I*I;
cout<<endl<<x[I]<<y[I];
}
return0;
}
```

وستكون قيم النتائج على النحو التالي:

0 0
1 1
2 4
3 9
4 16

إعطـاء قيمة أولية للمصفوفة ذات البعد الواحد Array Initialization

مثال على إدخال عدة عناصر من مصفوفة الدرجات grade[]

Int grade[5]={80,90,54,50,95}

ومثال على إدخال قيم عناصر المصفوفة الرمزية name[]

Char name[4]="nor"

لاحظ أن المتغير المرقم name[] مكون من أربعة عناصر بينما تم إعطاؤه ثلاثة عناصر فقط والسبب أن العنصر الرابع بالنسبة إلى المعطيات الرمزية يكون خاليا.

مثال٣:

```
#include  " stdafx.h"
#include " iostream.h"
main ()
{
int a[6]={40,60,50,70,80,90}
int I;
for(I=0;I<6;I++)
cout<<a[I]<<endl;
return0;
}
```

تم إعطاء القيم من قبل المستخدم مسبقا هنا

والناتج طبعا سيكون كالتالي:

40
60
50
70
80
90

مثال ٤:
قم بكتابة برنامج يقوم بإيجاد مجموع ، ومعدل علامات الطالب في 5 مواد وهذه العلامات كالتالي:
87,67,81,90,55

```
#include  " stdafx.h"
#include " iostream.h"
int m,i;
main ()
{
int a[5]={87,67,81,90,55}
int s=0;
for(i=0;i<5;i++)
s=s+m[a];
float avg=s/5;
cout<<avg<<endl;<<s<<endl;
return0;
}
```

والناتج سيكون كالتالي:

87
735

المعدل 87
والمجموع 735

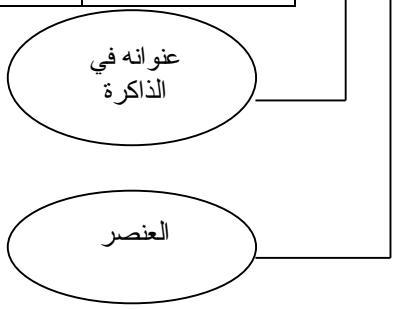
عـنوان عناصـر المصـفوفة فـي الذاكـرة Addressing Array Elements in Memory

ذكـرنا من قبل أن آي متغير أو عنصر من متغير ذاتي مرقم ، يحتل موقعا من الذاكرة يسـتعمل عـادة مؤشـرا لكـل متغـير أو عنصـر ، لـيكون دلـيلا علـى اسـتعمال هـذه المتغـيرات والعناصـر بسـهولة ويسـر ، والمـثال التالـي يوضـح هـذه العملـية بالنسـبة للمصفوفة ذات بعد واحد.

Int x[5];

يمكـن تمثـيل عناصـر المصـفوفة x المعلـن عـنها ، مـع عناوينـها بالشـكل التوضـيحي التالي (من اليسار إلى اليمين)

الأول	الثاني	الثالث	الرابع	الخامس
X[0]	X[1]	X[2]	X[3]	X[4]
100	101	102	103	104



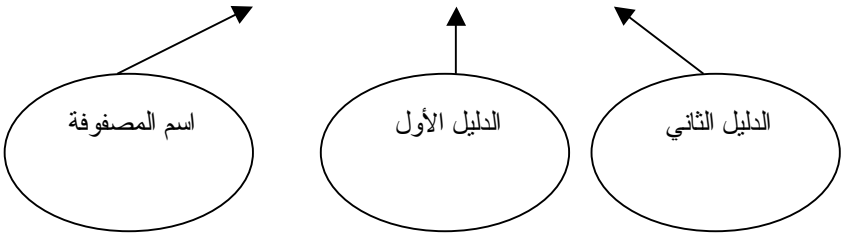
إذا فرضـنا أن عنوان موقـع العنصـر الأول x[0] فـي الذاكـرة هو 100 ، فان عناوين العناصر الأخرى تكون على التوالي 101 102 103 104

يمكن تشبيه العلاقة بين قيمة العنصر ، وعنوانه ، بالعلاقة بين علامة طالب ، ورقمه الجامعي ، إذ علامته هي قيمة نشطه كعنصر ، ليس لها علاقة برقم مقعده الجامعي.

المصفوفة ذات البعدين Two-Dimensional Arrays

تشبة المصفوفة ذات البعدين في طريقة تعاملها ، المصفوفة ذات البعد الواحد إلا أن لها عداديـن(index2) دليليـن أو مرقميـن إحداهمـا عـداد للصـفوف ، والأخـر عـداد للأعمدة ويأخذ الإعلان عن المصفوفة الشكل العام التالي:

```
Type-specifier arraey_name [index 1][index 2];
```



فمثلا المصفوفة :

```
Int x[2][3];
```

وهي مصفوفة صحيحة العناصر int أبعادها هي عدد الصفوف=2 ، وعدد الأعمدة =3

لاحظ أن عدد الصفوف يوضع بين قوسين وحده ، وكذلك عداد الأعمدة .

مثال٥:
شاهد هذا المثال الذي يستخدم 5 طلاب و 3 علامات:

```
#include  " stdafx.h"
#include " iostream.h"
main ()
{
int m[5][3];
int I,j;
for(I=0;I<5;I++)
for(j=0;j>3;j++)
cin>>m[I][j];
return0;
}
```

وبالنسبة لعناوين العناصر المصفوفة متعددة الأبعاد في الذاكرة ، لا يختلف عما ذكرنا بالنسبة للمصفوفات ذات البعد الواحد ، ولذلك لو فرضنا ، في المثال السابق أن العنصر x[0,0] كان عنوانه 100 مثلا فان عناوين العناصر التالية: حسب ترتيبها المذكور أعلاه هي 100-101-102 لعناصر الصف الأول 103-104-105 لعناصر الصف الثاني.

الدوال
Functions

مقدمة Introduction

تعرف الدالة (الاقتران) على أنها : جملة أو مجموعة جمل أو تعليمات ، ذات كيان خاص ، تقوم بعملية أو مجموعة عمليات ، سواء عمليات إدخال أو إخراج أو عمليات حسابية أو منطقية ، وتحتل الدالة موقعا من البرنامج ، آي أنها جزء منه ، أو يمكن القول أن برنامج c++ ، يتكون من مجموعة من الدوال.

ومن فوائد الدوال التالي:

١- تساعد الدوال المخزنة في ذاكرة الحاسب على اختصار البرنامج إذ يكتفى باستعادتها باسمها فقط لتقوم بالعمل المطلوب.
٢- تساعد الدوال المخزنة في مكتبة الحاسب ، أو التي يكتبها المبرمج على تلافي عمليات التكرار في خطوات البرنامج التي تتطلب عملا طويلا وشاقا.
٣- تساعد الدوال الجاهزة على تسهيل عملية البرمجة نفسها.
٤- توفر مساحة من الذاكرة المطلوبة.
٥- اختصار عمليات زمن البرمجة وتنفيذ البرنامج بأسرع وقت ممكن.

وللتدليل على أهمية الدوال في برمجة c++ خذ المثال التالي:
لو أردنا كتابة خوارزمية لخطوات صنع كأس من الشاي فأننا نكتب ما يأتي:

١- ضع الماء في غلاية الشاي.
٢- سخن الماء حتى يغلى.
٣- أضف شايا إلى الماء.
٤- أضف سكرا إليه.
٥- أطفئ النار.
٦- صب شايا في كأس.

افرض الآن أننا نود طلب كأس من الشاي من مقهى مجاور : أن خطوات الخوارزمية التي نحتاجها الآن هي خطوه واحده فقط وهي:

١- استدع كأس من الشاي.

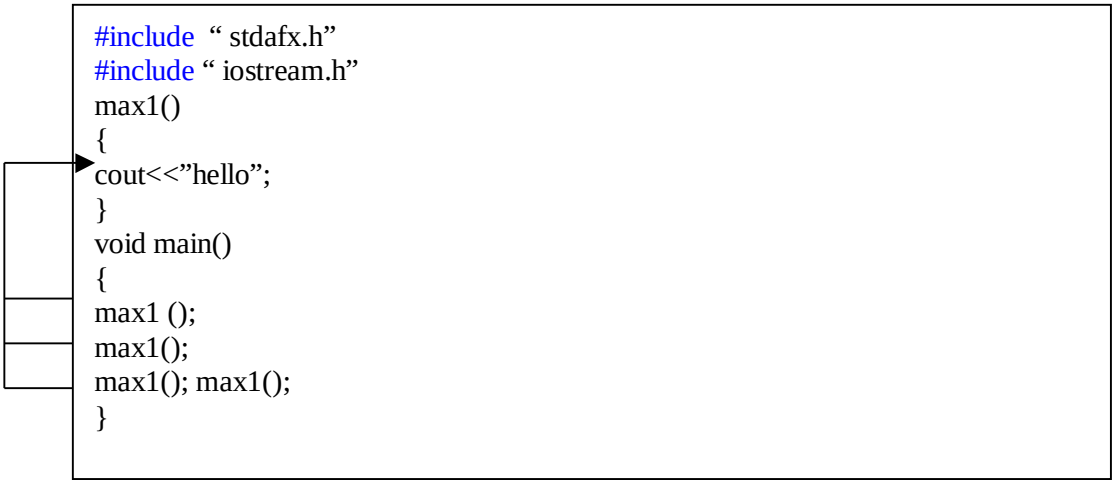
تخيل الآن كم وفرنا من الخطوات لو استعملنا الدوال الجاهزة (أو التي يجهزها المبرمج من قبل) بدلا من خطواتها التفصيلية وبخاصة في برنامج يتطلب حسابات وعمليات كثيرة وكم يكون البرنامج سهلا وواضحا وقتذاك.

وتأخذ الدالة الشكل العام التالي:

```
Type-specified function-name (formal parameters;
{
function body
}
```

وقد ذكرنا من قبل أن الدالة قد تعتمد على متغير أو اكثر ، وقد لا تعتمد على آي متغير ، وفي كلا الحالتين ، يستعمل بعد اسم الدالة قوسين () سواء كان بينهما متغيرات أم لا .

مثال ١:



```
#include  " stdafx.h"
#include " iostream.h"
max1()
{
cout<<"hello";
}
void main()
{
max1 ();
max1();
max1(); max1();
}
```

والناتج:

hello hello hello hello

طبعا للعلم أعزائي أننا في هذا الفصل الدوال نلاحظ أن بداية قراءة المترجم للبرنامج لا تبدأ من أول البرنامج كالمعتاد فالقراءة تبدأ من الأسفل آي أنها تبدا بالماين main سواء كان في الوسط أو الأسفل فأنها تقراء أولا الـmain ثم تبحث ما داخله وتبدا بالبحث عن معنى الكلمة max1() في الدالة max1() في الأعلى لتجد أن هناك جملة طباعة وهكذا تتكرر حتى يتم تعريف ما بداخل الـmain .

تطبيقات على الدوال

مثال ٢:

شاهد هذا البرنامج وتتبعه أولا بالـmain وانظر للنتائج:

```
#include  " stdafx.h"
#include " iostream.h"
int x,y;
void max()
{
x=x+y;
}
void fax()
{
max();
max();
{
void main()
{
y=10; x=0;
max();
fax();
cout<<x<<y;
}
```

أعزائي سأشرح النتائج قبل إظهارها للتسهيل عليكم في الأمثلة القادمة:

لنتعبر أن هناك ثلاث كواكب:

كواكب main الرئيسي

كوكب fax

كوكب max

من المعروف أننا سوف نبدا بكوكب main لنشاهد ما بداخل نشاهد أن هناك قيمتين عدديتين x y لكنه لا يعرفه هل هي أعداد حقيقية أم صحيحه لذلك يذهب في الأعلى ليبحث عنها في أول البرنامج ليجد أنها أعداد صحيحة int ، ثم بعد ذلك يرجع للكوكب الرئيسي main ليشاهد عبارة max() فيذهب للبحث عنها في الكوكب max طبعا ليجد بداخلها أن قيمة x تساوي x+y أي أن x=0+10 لتصبح قيمة x=10 بعد ذلك يخرج من الكوكب max ويرجع للكوكب الرئيسي ليشاهد العبارة fax() فيذهب للتعرف عليها بالكوكب fax() ويشاهد بداخلها عبارة max ليذهب بذلك للكوكب max ويجمع من مرة أخرى فتصبح كالتالي:

X=10+10 وبذلك تصبح قيمة x=20 بعد ذلك يرجع للكوكب fax ليشاهد عبارة max() فيذهب للكوكب max ويجمع مرة أخرى كالتالي:

X=20+10 وبذلك تصبح قيمة x=30

ثم بعدها يرجع للكوكب الرئيسي main ليشاهد جملة الطباعة والناتج كالتالي:

30 10

مثال ٣:

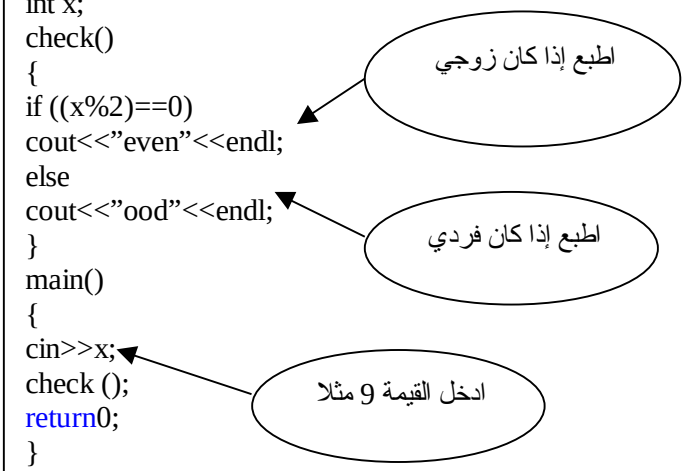
```
#include  " stdafx.h"
#include " iostream.h"
void x1()
{
cout<<"*";
}
void x2()
{
cout<<"\t";
}
void yaya()
{
x1();
x2();
x1();
}
void kiki()
{
cout<<"\n":
}
void main()
{
int I;
for(I=0;I<=3;I++)
{
yaya();
kiki();
}
}
```

والناتج سيكون كالتالي:

```
*       *
*       *
*       *
*       *
```

مثال ٤:
قم بكتابة برنامج يقوم بقراءة عدد صحيح ومن ثم طباعة ما إذا كان الرقم زوجي أم فردي من خلال دالة أو اقتران؟

```
#include  " stdafx.h"
#include " iostream.h"
int x;
check()
{
if ((x%2)==0)
cout<<"even"<<endl;
else
cout<<"ood"<<endl;
}
main()
{
cin>>x;
check ();
return0;
}
```



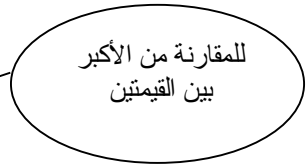
طبعا والناتج سيكون حسبما أدخلت لندخل مثلا القيمة 9.. والناتج سيكون:

ood

لان القيمة المدخلة 9 عدد فردي وليس زوجي .

مثال٥:
قم بكتابة برنامج يقوم باستخراج اكبر رقم ما بين رقمين مدخلين من قبل المستخدم وذلك من خلال دالة اسمها max ؟

```
#include  " stdafx.h"
#include " iostream.h"
int x,y;
max()
{
if (x>y)
cout<<x<<endl;
else
cout<<y<<endl;
}
main()
{
cin>>x>>y;
max ();
{
return0;
}
```



والناتج طبعا سيكون:

20

لان العدد الذي أدخلنا 20 اكبر من العدد الذي أدخلنا 10 فقام بطباعته حسب الشرط .

تقنية الأقراص و دوال الملفات الانتقالية
disk Files

مقدمة Introduction

صممت هذه الدوال للتعامل مع الملفات الانتقالية للأقراص Buffered filw system لمعالجة النصوص ، كما كان متوفرا في لغة C++ النظام الأخر غير الانتقالي unbuffered المشابهة لنظام يونيكس للإدخال والإخراج UNIX-like I/O ، وكان النظام الأخير يستعمل للتعامل مع المعطيات بنسق النظام الثنائي binary system ، ألا أن لجنة C في معهد المقاييس الأمريكي الوطني للغات البرمجة ANISI-C Committee قررت مؤخرا الاستغناء عن النظام غير الانتقالي من أنظمة التعامل مع المعطيات الثنائية ، فأوجدت البديل ، بحيث يصبح بمقدور دوال النظام الانتقالي التعامل مع كل من النصوص text والمعطيات الثنائية binary system في وقت واحد . أي أن آخر صورة (version) من C ، يتعامل بنظام واحد فقط هو نظام الملفات الانتقالية .

ونحتاج عند التعامل مع دوال هذا النظام ، أن نستعمل ملف الدليل للإدخال والإخراج stdio.h ويلخص لنا الجدول التالي اشهر هذه الدوال:

اسم الدالة	وظيفتها
fopen()	تفتح لك ملفا
fclose()	تغلق لك ملفا
putc()	تخرج (تطبع) لبنة (رمزا) وهي مثل char
getc()	تدخل لبنة (رمزا) إلى الملف ، وهي مثل char
fseek()	تبحث لك عن بعض الرموز (نص) في الملف
fprintf()	مثل وظيفة printf لكن للملفات
fscanf()	مثل وظيفة scanf لكن للملفات
feof()	تعطي النتيجة true عند وصول نهاية الملف
ferror()	تعطي النتيجة true عند حدوث خطأ
rewind()	ترجع الملف إلى بدايته

الجدول ١ - ٥

دالة فتح الملف fopen ()

مثال ١:

```
#include “stdio.h”
main()
{
FILE *f;
F=fopen(“C:\\matrix.txt”,”w”);
Fclose (f);

return0;
}
```

نكتب آي اسم للتخزين بموقع الذاكرة مثلا f

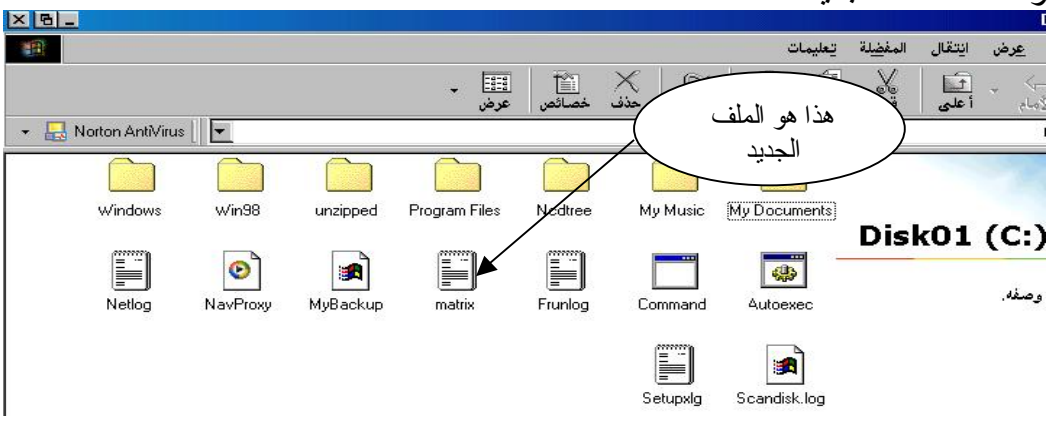
الرمز w لاستحداث نص جديد للكتابة

اسم الملف ونحدد موقعه(مساره) فى القرص C

نلاحظ أننا قمنا في السطر الأول أعلاه بإنشاء موقع للملف في ذاكرة الحاسوب طبعا نكتب FILE بالأحرف الأبجدية الكبرى ثم بعد ذلك * وبعدها نطلق المسمى مثلا f

بعد ذلك نقوم بعمل الدالة الخاص بفتح ملف في قرص disk وتعود (تعطي قيمة) بمؤشر الملف المعرف له وهو f طبعا.
ثم نقوم بتحديد موقع الملف المراد إنشاءه ويجب التحذير هنا انه يجب عدم كتابة اسم ملف من ملفات النظام system لتجنب عدم حدوث خلل ومشاكل بالجهاز مثلا نكتب كما أعلاه matrix.txt وهو ملف نصي ويجب وضع العلامتين \\ بعد كتابة القرص C ولا يصح وضع علامة واحده \ فقط ، بعد ذلك نكتب الرمز w لاستحداث الملف الجديد ثم بعد ذلك في السطر الأخير نقوم بإغلاقه بالدالة (f) fclose ويجب كتابة هذه الدالة كي يصبح البرنامج صحيح .

الآن قم بتنفيذ البرنامج بعد ذلك لن يظهر لك شي قد تستغرب ما الفائدة الآن اذهب عزيزي إلى القرص C وهو الذي قمت بإنشاء الملف فيه (المسار أعلاه)
وشاهد الملف الجديد



جدول الأنماط ، حسب ما قررته ANSI مؤخرا:

رمز النمط	وظيفته
"r"	لفتح ملف النص للقراءة
"w"	لاستحدث ملف نص للكتابة
"a"	للإلحاق بملف نص
"rb"	لفتح ملف ثنائي القراءة
"wb"	لاستحداث ملف ثنائي الكتابة
"ab"	للإلحاق بملف ثنائي
"r+"	لفتح ملف نص للقراءة أو الكتابة
"w+"	لاستحداث نص للقراءة أو الكتابة
"a+"	لفتح ملف نص للقراءة أو الكتابة
"r+b"	لفتح ملف ثنائي للقراءة أو الكتابة
"w+b"	لاستحداث ملف ثنائي للقراءة أو الكتابة
"a+b"	لفتح ملف ثنائي للقراءة أو الكتابة

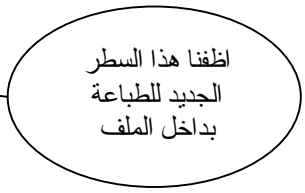
الجدول ٢ - ٥

نلاحظ أن هذا الجدول يمكن استعماله لكل من ملفات النص والملفات الثنائية.

دالة الكتابة داخل الملف fprintf ()

مثال ٢:
قم بكتابة النص "welcome to c++ " داخل الملف الذي قمنا بإنشائه في المثال السابق (1) وهو ملف matrix ؟

```
#include “stdio.h”
main()
{
FILE *f;
f=fopen(“C:\\matrix.txt”,”w”);
fprintf (f,”welcome to c++”);


return 0;
}
```



لاحظ عزيزي القارئ أننا قمنا بوضع الدالة (fprintf) وهي الدالة الخاصة بالكتابة داخل الملفات بإمكانك الرجوع للجدول ١-٥ الآن نفذ البرنامج وبعد تنفيذ ارجع للملف في القرص c وافتحه لتشاهد العبارة welcome to c++ قد كتبة بداخله.

دالة إغلاق الملف fclose ()

وتعمل عكس الدالة fopen() ، فتغلق الملف الذي فتحناه بالدالة fopen() ، وكل الملفات المطلوبة منك إغلاقها قبل نهاية البرنامج ، وفي حالة عدم قيامنا بإغلاق الملف فان عددا من المشكلات قد تقع ، ومنها ضياع بعض المعطيات ، واتلاف الملف ، ووقوع أخطاء في البرنامج .

مثال٣:

```
#include “stdio.h”
main()
{
FILE *f;
f=fopen(“C:\\matrix.txt”,”w”);
fprintf (f,”welcome to c++”);
fclose (f)

return 0;
}
```

قمنا بإغلاق الملف في هذا السطر كما تلاحظ

الدالتان putw() getw()

تقوم معظم مترجمات c++ باستعمال هاتين الدالتين الإضافيتين لتقوما بعمليتي قراءة وكتابة الأعداد الصحيحة من والي ملفات الأقراص ، وهاتان الدالتان غير معتمدين من قبل معهد المقاييس الوطني الأمريكي ANSI ، وتعمل هاتان الدالتان تماما كالدالتين getc() و putc() والفرق الوحيد انهما تتعاملان مع عدد صحيح بدلا من رمزي.

مثال٤:
الدالة التالية تقوم بكتابة (طباعة) العدد الصحيح 1000 في ملف القرص المشار إليه بمؤشر الملف f :

```
Putw (1000,f);
```

مثال٥:
الدالة التالية تقوم بكتابة قيمة المتغير الصحيح x في ملف القرص المشار إليه بمؤشر الملف f :

```
Putw (x,f);
```

مثال٦:
الدالة التالية تقوم بقراءة عدد صحيح ، من ملف مشار إليه بمؤشر الملف f :

```
#include “stdio.h”
main()
{
FILE *f;
int x;
x=getw (f);
printf (“%d”,x);
return 0;
}
```

النهاية

٢٨ تشرين الأول ٢٠٠٣

السلام عليكم ورحمةالله وبركاته.....
الحمدلله والصلاة والسلام على افضل الأنبياء والمرسلين قدوتنا وحبيبنا وشفيعنا ان شاء الله محمد بن عبد الله وعلى اله وصحبه اجمعين.....

نبدا ان شاء الله تعالى سلسله دروس لغة من اهم وأروع لغات البرمجه الا وهي لغة C++ ,في هذه الدورة ان شاء الله سنبدأ في شرح هذه الدوره من الصفر حتى يتسنى للجميع ان شاء الله فهم هذه اللغه بيسر وسهوله وسنركز في شرحنا على اللأمثله العمليه لأن خير طريقة لتعلم هذه اللغه في وجهة هو تعلمها من خلال الأمثلة العمليه ومحاولة تطبيقها على البرنامج وتجريبها.وتستطيع استخدام برنامج Borland C++ أو برنامج Visual C++ 6.0 , وسنتطرف بأذن الله الى شرح كيفية استخدام هذين البرنامجين ليتسنى لك العمل عليهم وسنتوسع في شرح برنامج Visual C++ 6.0 بما انه الأفضل والأكثر أنتشارا.
والذي يجيد لغة ال C او لديه خلفيه عنها سيسهل عليه جدا تعلم لغة C++ فهما متشابهتان تقريبا. وفي الحقيقة هما من اهم اللغات وبالنسبة لي أعتبرهما المدخل لتعلم لغات البرمجة فعندما تتقن تعلم هاتين اللغتين او واحده منهما تكون قد اكتسبت أساسا قويا في البرمجة ويسهل عليك الأنطلاق بعدها في عالم البرمجه .وهما اول لغتين تعلمتهما من لغات البرمجه,وسهل علي بفضل الله تعلم لغات برمجة اخرى.

الدرس الأول:

بسم الله نبدا أول دروسنا على بركة الله وكما اتفقنا سنركز على فهم هذه اللغه عن طريق الأمثلة العمليه...
وسنبدا بأول برنامج تراه في عالم ال C وال C++ .

```
1. #include <iostream>

2. int main()  // function main begins program execution

3. {
4.    std::cout << "Welcome to C++!\n";

5.   return 0;   // indicate that program ended successfully

6.    }
```

شرح البرنامج:

السطر الأول:
وهذا السطر اساسي جدا في أغلب البرامج ويجب أن تعود نفسك على كتابته عند البدء في كتابة البرنامج , ويعني ان برنامجك يستخدم مكتبة iostream. *وفي لغة C++ العديد من المكتبات المستخدمه سنتطرق اليها في مرحلة متقدمه ان شاء الله نستخدم كل مكتبة على حسب احتياجنا لها في البرنامج ومكتبة ال* iostream مسؤلة عن عمليتان الأدخال والأخراج في الC++ . وهي اختصار لinput output stream
ويتمثلان هتين العمليتين في cin للأدخال(أي أدخال المعلومه او تخزين القيمه في متغير بأسم ونوع تقوم بتحديده للبرنامج), وال cout للأخراج (أي اخراج المعلومة أو ما نريد كتابته على الشاشه).
خلاصة هذا السطر انه يقوم بتوجيه المترجم ليقوم بجعل البرنامج الذي تقوم بكتابته يحتوي على مكتبة iostream.

السطر الثاني:

ملاحظه مهمه: ستلاحظ أحتواء السطر الثاني على // function main begins program execution
هذه العباره ليس لها علاقة بالبرنامج ,تستخدم كهامش للتنويه اي ان اي شيء بعد // يستخدم كتوضيح يكتبه المبرمج في برنامجه ولا ينظر له المترجم عند تنفيذ أوامر البرنامج او عند تنفيذ أسطر البرنامج فجميع اللأسطر او العبارات بعد // وقد لونتها باللون الحمر ليسهل عليك تميزها لا يضع لها المبرمج اعتبار ولا يراها أصلا عند تنفيذ اسطر البرنامج,هي فقط يكتبها المبرمج لتوضيح معني السطر او كتابة ملاحظه تذكيريه له وليس بالضرورة كتابة هذه الملاحظات فيمكنك الأستغناء عن العبارات الي باللون الحمر في البرنامج التي تأتي بعد // .
فمثلا في السطر الثاني ما بعد ال // وهو عبارة function main begins program execution مجرد ملاحظة لتعطي قاريء البرنامج تنويه عن وظيفة int main() .

هذا السطر يمثل الدالة الرئيسيه للبرنامج وجميع البرامج تحتوي على دالة الmain ,فجميع العمليات التي سيقوم بها البرنامج الذي ستقوم بكتابته يجب ان يكون داخل هذه الدالة التي يجب ان تبدئها بقوس { كما في السطر الثالث وتنهيها بقوس } كما في السطر السادس ومحتوى برنامجك يكون بين هذين القوسين.

السطر الرابع:
في هذا السطر سترى ان هذا السطر يحتوي في بدايته على std:: طبعا هذه الجمله نقوم بكتابتها قبل ال cin وال cout وتستطيع تعريفهما قبل ال main وهذا الأفضل لك لا تحتاج لتعريفهما عند كل استخدام لل cin وال cout وسيكون البرنامج هكذا.

```
1. #include <iostream>
2. using std::cout;  // program uses cout

3. int main()  // function main begins program execution

4. {
5.    cout << "Welcome to C++!\n";

6.   return 0;   // indicate that program ended successfully
7.  }
```

ولك ان تلاحظ الأختلاف بين البرنامجين وأنصحك بأستخدام الأسلوب الثاني الأكثر اختصارا.
نعود لأستكمال شرحنا ولاحظ اننا نشرح البرنامج بالترتيب الأول اي البرنامج الذي كتبناه في بداية شرحنا.
في السطر الرابع قمنا بأستخدام ال cout<< ونستخدم هذا الخدمة اتي توفرها مكتبة الiostream كما اسلفنا عند طباعة معلومات او قيم على الشاشة وتمكننا هذه العمليه من اخراج المعلومات على الشاشه وتكون جملة cout متبوعه بمعاملين<< وبعدهما يجب وضع المعلومات المراد أخراجها على الشاشة بين "" أما اذا كنا نريد ان نظهر قيمة متغير نكتب أسم المتغير بدون "" لأنه عند وضع اي شيء بين ال "" يتم طباعته كما هو فمثلا عند كتابة السطر cout<<"salam 3alekom"; يقوم البرنامج بأظهار على الشاشهsalam 3alekom , ومعنى \n يعني يقوم البرنامج بالنزول الى سطر جديد بعد كتابة Welcome to C++! على الشاشة ونلاحظ ان برنامجنا سيقوم بكتابة Welcome to C++! وهذه هي العملية التي كتب من اجلها هذا البرنامج البسيط .
وأخيرا ننهي السطر بعلامة الفاصلة المنقوطه; ويجب انهاء كل سطر بعلامة الفاصلة المنقوطه فهي تعطي المترجم معنى بأن السطر قد انتهى وانتهت الأوامر لهذا السطر فعليه الذهاب للسطر الثاني وتنفيذ الأوامر فيه .

السطر الخامس:
هذا السطر يحتوي على return 0; وهذا السطر يجب كتابته عند نهاية دالة ال main فهو يخبرنا بان البرنامج نفذ بنجاح وتستطيع استبداله بكتابة void main() بدل int main() فعند كتابة void main لست بحاجة لكتابة return 0; قبل غلق دالة ال main بـ}.
ملاحظة: لاحظ ان return 0; تحتوي على صفر وليس حرف الo, فمن أكثر الأخطاء شيوعا كتابة حرف ال o بدل الصفر.

وصلنا اخوتي وأخواتي في الله الى نهاية الدرس الأول الذي أسال الله ان يكون واضحا وبفترض ان تكون قد فهمت من خلال هذا البرنامج بعض الأساسيات في ال C++ وسنواصل في بقية الدروس شرح برامج اكثر تطورا وشرح مفاهيم واساسيات اعمق فهذا الدرس عباره عن مقدمة تدخلك الى عالم C++ .
وفي نهاية الدرس نقول هذا اجتهادنا فأن اخطانا فمن انفسنا وان اصبنا فمن الله سبحانه وتعالى.....

°^~*¤®§(*§سبحانك اللهم وبحمدك أشهد ان لا اله الاانت استغفرك وأتوب أليك §*)§®¤*~^°

`~'*¤!||!¤*'~`((كاتبة الدرس .....بنت النور.....عفوا يمنع اجراء اي تعديل للملف))`~'*¤!||!¤*'~`

`~'*¤!||!¤*'~`((نسألكم دعوة صادقة لأختكم في ظهر الغيب))`~'*¤!||!¤*'~`

السلام عليكم ورحمة الله وبركاته.....
الحمدلله والصلاة والسلام على افضل الأنبياء والمرسلين قدوتنا وحبيبنا وشفيعنا ان شاء الله محمد بن عبد الله وعلى اله وصحبه اجمعين.....

نواصل في هذا الموضوع الدرس الثاني في دورة لغة الC++ وبعد ان اخذنا فكرة ارجو ان تكون كمقدمة جيدة ووافيه للدخول في تفاصيل هذه اللغة.

في هذا الدرس سوف نقوم بالتعرف على المتغيرات (Variables) ,تسمية المتغيرات ,انماط المتغيرات وانواعها ,الثوابت,العمليات الحسابيه(Arithmetic Operation),عمليات المقارنه(relation operator).

١- **المتغيرات (Variables) وتسميتها.**

تتكون ذاكرة الحاسوب من خانات اما ان يكون في الخانه صفر او واحد , يترجم كل شيء الى لغة البانيرى(binary)وهذه اللغه عباره عن صفر ا واحد ٠١٠١٠١٠١ والذاكره تنقسم الى bytes وكل bytes=8bit وال bit =one or zero .فعندما تريد ان تدخل قيمة بالذاكره وتحفظها يجب عليك ان تحجز لها مساحة في الذاكره وتسميها وتعرف نوعها ومساحتها,وهذا ما يجب القيام به في عملية الأدخال التي شرحناها في الدرس السابق.

مثال:int box;
Cin >>box;

لا حظ هنا البرنامج ليس كاملا فقط قمت بأقتطاع ما نحتاجه لفهم جزيئة المتغيرات وتسميتها,نلاحظ في البداية قمنا بتعريف نوع المتغير الذي اسميناه box ونوعه int يعني يحتوي على ارقام صحيحه وسندرج كل الانواع وشرح لها وهنا لم نحدد قيمتة لننا سندخلها عن طريق ال cin ولو اننا حددنا قيمتها لعملنا int box =5 فبهذه الطريقه قمنا بحجز مساحه اسمها box ونوعها int يعني تحوي ارقام صحيحه وقيمتها خمسه هنا ثبتنا القيمه على خمسه لكن في الأعلى نحن لم نحدد القيمه وقمنا بأدخال القيمه فالقيمه التي سيدخلها المستخدم ستوضع في المساحه المحجوزة بأسم box.

ملاحظه: عند تسمية المتغير لابأس بأستخدام الحروف والأرقام وعلامة ويمنع استخدام @ ,$ والرموز التي على هذه الشاكلة.

وسندرج أكثر الأنواع استخداما ومعناها..

النوع	ملاحظات
int	يعني المساحه المحجوزة تحوي ارقام صحيحه. ويمكنك هنا استخدام long int box; أو short int box; أو long box أو short box ;
float	يعني المساحه المحجوزة تحوي أرقام عشرية مثال ١,٢ ,٣,٥
double	يحوي أعداد كبيره.
char	يحتوي على حروف ورموز ولا بأس ان يحوي أرقام.
bool	تحوي true او false

٢- **الثوابت(constants):**

عندما تريد حجز مساحه لمتغير بنوع محدد وأسم تختاره انت وتريد ان تجعل هذه القيمه ثابته , فعليك بأضافة const قبل تعريف المتغير فيصبح const int box =5; فهنا تكون قيمة المتغير box تساوي خمسة دائما.

٣- **العمليات الحسابيه(Arithmetic Operation)**

١- عملية الجمع (+).
مثال : sum=num1 + num2;

٢- عملية الطرح (-).
مثال : sum=num1 - num2;

٣- عكلية الضرب(*).
مثال : sum=num1* num2;

٤- عملية باقي القسمه(%).
مثال : sum=num1 % num2;

٥- عملية القسمة (/).
مثال : sum=num1/num2;

٣- عمليات المقارنه(relation operator).

١- (==) وهذه لمقارنة المساواة .
مثال : اذا قلنا num 1 == num2 نحن نقصد هنا ان num1 يساوي num2 وانتبه ان تخلط بين هذه العمليه واشارة = فأشارة ال = تعني ان القيمة التي باليمين تنتقل الى القيمة التي باليسار
مثلا: num1=num2 يعني قيمة num2 تنسخ في num1 .

٢- (=!) يعني لا تساوي.
مثال : num1 != num2 يعني num1 لا يساوي num2 .

٣ – (>)
مثال : num1>num2 يعني num1 أكبر من num2 .

٤ – (<)
مثال : num1<num2 يعني num1 أصغر من num2 .

٥ - (>=)
مثال : num1 >= num2 يعني num1 أكبر من أو يساوي num2 .

٦ - (<=)
مثال : num1<=num2 يعني num1 أصغر من أو يساوي num2 .

وهنا نصل الى نهاية درسنا اليوم ولكن قبل ان ننتهي اليكم هذا السؤال لتختبروا مدى فهمكم لما شرحنا وسنقوم بحله ان شاء الله بداية الدرس القادم....وساقوم دائما بكتابة السؤال بالعربية والانكليزية.

السؤال: كتابة برنامج يقوم بجمع وطرح وقسمة عددين صحيحين وكتابة الناتج للعمليات الثلاث على الشاشة كناتج للبرنامج.

****

Questions: write a program which add, subtract and divide two integer numbers and show the result for three arithmetic operations as an output for the program.

°^~*¤®§(*§سبحانك اللهم وبحمدك أشهد ان لا اله الاانت استغفرك وأتوب أليك §*)§®¤*~^°

`~'*¤!||!¤*'~`((كاتبة الدرس .....بنت النور.....عفوا يمنع اجراء اي تعديل للملف))`~'*¤!||!¤*'~`

`~'*¤!||!¤*'~`((نسألكم دعوة صادقة لأختكم في ظهر الغيب))`~'*¤!||!¤*'~`

السلام عليكم ورحمة الله وبركاته.....
الحمدلله والصلاة والسلام على افضل الأنبياء والمرسلين قدوتنا وحبيبنا وشفيعنا ان شاء الله محمد بن عبد الله وعلى اله وصحبه اجمعين.....

نبدا بأذن الله الدرس الثالث وقد انتهينا في الدرس السابق بطرح سؤال في أخر الدرس وسنبدا هذا الدرس بالأجابه عن السؤال السابق وسنطرح اسأله أخرى والأجابه عنها ان شاء الله,ملاحظه لم انوه لها في الدرس السابق لاحظ عند تسمية المتغيرات يجب مراعاة أن التسميه تكون حساسه للحروف الصغيره والكبيره يعني يجب الانتباه عند كتابة الأسم في البرنامج ان يكون نفس الأسم بالضبط دون التغيير في شيء.
حاول ان تجيب على الأسأله المرفقه بنفسك أولا ثم بعدها طابق حلك بالحل الذي وضعته حتى تستفيد لأن البرمجه لا يمكنك تعلمها من غير الممارسه وحل الأسأله فبمجرد القرآءه لا يمكنك ان تكون مبرمج.

السؤال١: كتابة برنامج يقوم بجمع وطرح وقسمة عددين صحيحين وكتابة الناتج للعمليات الثلاث على الشاشة كناتج للبرنامج.

****

Questions1: write a program which add, subtract and divide two integer numbers and show the result for three arithmetic operations as an output for the program.

الأجابة:

ملاحظه: سنقوم بكتابة البرنامج وتوضيح الفكرة وقد أسلفنا في الدرس الاول شرح تفصيلي للأساسيات فأن كنت قد نسيتها فأرجع للدرس الأول ثم واصل معنا هنا.

```
#include <iostream>

int main()
{
   int integer1;  // أسم المساحة لتخزين الرقم الأول
   int integer2;  // أسم المساحه لتخزين الرقم الثاني
   int sum,sub;       // أسم المساحه التي سنخزن فيها ناتج الجمع والطرح على التوالي
   float div;  //جعلنا النوع له لعداد عشريه لن القسمة لا تعطي دائما اعداد صحيحه

   std::cout << "Enter first integer\n";//لأعطاء المستخدم رسالة لأدخال رقم
   std::cin >> integer1;                  // لأدخال الرقم في المساحه المجوزة

   std::cout << "Enter second integer\n"; //لأعطاء المستخدم رسالة لأدخال رقم
   std::cin >> integer2;                  // لأدخال الرقم في المساحه المجوزة

   sum = integer1 + integer2;  // للقيام بعملية الجمع
   sub = integer1 - integer2;  //للقيام بعملية الطرح
   div = integer1 / integer2;  // للقيام بعملية القسمه

   std::cout << "Sum is " << sum << std::endl; //لطباعة ناتج الجمع

   std::cout << "sub is " << sub << std::endl; // لطباعة ناتج الطرح

   std::cout << "div is " << div << std::endl; // لطباعة ناتج القسمة



   return 0;   // indicate that program ended successfully

} // end function main
```

ملاحظه : الذى اللون الأزرق يمثل الكود وما سواه مجرد ملاحظات للتوضيح . . . . . .

الناتج على الشاشة سيكون(output):

Enter first integer
٤
Enter second integer
٥
Sum is ٩
sub is -1
div is 0.8

سؤال آخر: وسأقوم بكتابته بالعربية والأنكليزية

Questions2 : Write a C++program to read two integers (first and second), and two real numbers (third and fourth), then calculate the value of:

Result1 = 7 first^3 + second^2

$$Result2 = \frac{(third^2 - fourth)^{(1/2)}}{third - fourth}$$

Your program should print the values of Result1 and Result2.

Note: 2^3 means the number 2 is raised to the power of 3.

- السؤال ٢ : المطلوب كتابة برنامج بلغة ال C++ يقوم بأدخال عددان صحيحان وآخرين عشريين ويجب ان يقوم البرنامج بتطبيق المعادله الاولى على العددين الصحيحين وتطبيق المعادلة الثانيه على العددين العشريين والمعادلتين كالتالي:

Result1 = 7 first^3 + second^2

$$Result2 = \frac{(third^2 - fourth)^{(1/2)}}{third - fourth}$$

- ملاحظه : يجب ان يقوم البرنامج بطباعة الناتج الأول والثاني على الشاشة , و 2^3 تعني الرقم ٢ مرفوع للقوة ٣ او ٢ أس ٣ ويجب عليك تحويل هذه العبارة الى لغة السي بلس

- الأجابة:

Solution:

```
#include <iostream>
#include <math.h>

void main ()
{

 int first, second, Result1;
 float third, fourth, Result2;

cout<<"enter the first number"<<end1;
cin>>first;

cout<<"enter the second number"<<end1;
cin>>second;


 Result1 = 7* pow(first,3) + pow(second,2);

Cout<<"the result1 is"<<Result1<<end1;


Cout<<"enter the third number"<<end1;
Cin>>third;

Cout<<"enter the fourth number"<<end1;
Cin>>fourth;



 Result2 = pow(pow(third,2) - fourth,1/2)/ (third - fourth);
Cout<<"the result2 is"<<Result2<<end1;


}
```

ملاحظه: استخدمنا في البداية #include <math.h>
لأننا أستخدمنا pow وهو ضمن هذه المكتبة فيجب تعريفها في البداية

و لمعرفة المزيد عن المكتبات المستخدمة في السي بلس بلس ودوالها انصحكم بزيارة
هذا الموقع الرائع للسي بلس بلس http://www.cplusplus.com/

Question 3: write a program in C++ that accept a Fahrenheit temperature and output the equivalent Celsius temperature.

And the equation as follow:
Celsius=9/5(Fahrenheit-32)

السؤال٣: المطلوب كتابة برنامج بأستخدام لغة ال C++ يقوم بأخذ درجه الحراره بالفهرينهيت وأعطاء درجه الحراره بالسيلسيوس كناتج للبرنامج وطبعا المعادله المستخدمه للتحويل من الفهرينهيت للسيليسيوس كالتالي:

Celsius=9/5(Fahrenheit-32)

Solution:

```
#include <iostream>

void main ()
{

 float Celsius, Fahrenheit;

cout<<"please enter the temperature in Fahrenheit\n ";
cin>>Fahrenheit;

Celsius=9/5*(Fahrenheit-32)

Cout<<" the temperature in Celsius is "<<Celsius<<end1;


}
```

Question 4:write a complete C program that ask the user to enter three integer number and store them in three variable k,L,and M.The program should find the value of N according to the following formula.
N=K+L-3M.

SOLUTION:

```
#include <iostream>

void main ()
{
int K,L,M,N;

cout<<"please enter the value of K"<<end1;
cin>>K;
cout<<"please enter the value of L"<<end1;
cin>>L;
cout<<"please enter the value of M"<<end1;
cin>>M;

N=K+L-3*M;

cout<<"the value of N is"<<N<<end1;
}
```

*هنا نصل لنهاية درس اليوم اتمنى ان يكون واضح ومفيد وسننتقل باذن الله في الدرس القادم الى مرحلة اكثر تطورا في ال **C++** وأي استفسار يسرني الاجابه عليه....*

°^~*¤®§(*§سبحانك اللهم وبحمدك أشهد ان لا اله الاانت استغفرك وأتوب أليك §*)§®¤*~^°

`~'*¤!||!¤*'~`((كاتبة الدرس .....بنت النور.....عفوا يمنع اجراء اي تعديل للملف))`~'*¤!||!¤*'~`

`~'*¤!||!¤*'~`((نسألكم دعوة صادقة لأختكم في ظهر الغيب))`~'*¤!||!¤*'~`

السلام عليكم ورحمة الله وبركاته.....
الحمدلله والصلاة والسلام على افضل الأنبياء والمرسلين قدوتنا وحبيبنا وشفيعنا ان شاء الله محمد بن عبد الله وعلى اله وصحبه اجمعين.....

لقد انهينا التعرف على لغة ال C++ وكتابة برامج بسيطه بهذه اللغة القوية ولكن هل تعتقد ان يمكنك مواصلة البرمجه بهذه المباديء البسيطه؟ طبعا لا فقد يطلب منك كتابة برنامج لأدخال بيانات عدد كبير من الموظفين وتذكر ان المبرمج الناجح يكتب البرنامج الذي ينفذ المهمه بأقل سطور ممكنه لك يكون برنامج سريع عند التنفيذ,طبعا من الغير عملي جدا ان تدخل بيانات ١٠٠ موظف مثلا ب١٠٠ cout و١٠٠ cin . هنا تاتي ضرورة استخدام هذه الجمل المفيده التي سنتطرق اليها في دورتنا.

اليوم بأذن الله سنبدأ في التعرف على جمل مفيدة في السي ++ وهي التي ستقودنا الى بداية برمجة جيده..... سنتناول في هذه الدوره :

- if ,if/else, else if
- Switch
- do while
- while
- for
- break ,continue
- function
- arrays

في الدرس الرابع سنتناول شرح if ,if/else, else if :-

```
if (expression) {

 do the code inside the two brackets;


}
```

نبدا بال if :- كما ترى هذه الصيغه العامة لكتابة جملة ال if, يحتوي القوس الذي يأتي بعد ال if على الشرط الذي بموجبه يتم تنفيذ العمليات التي بداخل القوسين{ } ,فعند انطباق الشرط يتم تنفيذ ما بداخل القوسين{ } ومن ثم الانتقال الى العمليات التالية ان وجدت ونقصد بالعمليات التاليه العمليات التي تأتي بعد تنفيذ العمليات التي بداخل قوسين ال if, اما اذا لم يتحقق الشرط فلا ينظر لا بداخل ال if وينتقل مباشره الى العمليات التاليه التي تكون خارج نطاق ال if . وننتقل للمثال التالي لتتمكن من فهم الموضوع بشكل أكبر.

السؤال:قم بكابة برنامج بلغة ال C++ يقوم بأدخال رقمين وكتابة العلاقة بينهما .فمثلا يقوم البرنامج بكتابة اذا ما كانا الرقمين متساويين واذا لم يكونا يقوم بكتابة ذلك وتوضيح اي العددين اكبر .

```
QUESTIONS: write program using C++ that accept two number and give
the relationships they satisfy.
```

Answer:-

```
1. #include <iostream>

2. using std::cout;  // program uses cout
3. using std::cin;   // program uses cin
4. using std::endl;  // program uses endl

5. int main()
6. {
7.    int num1;  // first number to be read from user
8.  int num2;  // second number to be read from user

9.  cout << "Enter two integers, and I will tell you\n"
10.          << "the relationships they satisfy: ";


11.   cin >> num1 >> num2;   // read two integers


12.   if ( num1 == num2 )
13.      cout << num1 << " is equal to " << num2 << endl;

14.   if ( num1 != num2 )
15.      cout << num1 << " is not equal to " << num2 << endl;

16.   if ( num1 < num2 )
17.      cout << num1 << " is less than " << num2 << endl;

18.   if ( num1 > num2 )
19.      cout << num1 << " is greater than " << num2 << endl;

20.   if ( num1 <= num2 )
21.      cout << num1 << " is less than or equal to "
22.           << num2 << endl;

23.   if ( num1 >= num2 )
24.      cout << num1 << " is greater than or equal to "
25.           << num2 << endl;

26.   return 0;   // indicate that program ended successfully

27.      } // end function main
```

سنبدأ بالشرح من السطر ال ١٢ وذلك لأننا قمنا بشرح السطور من ١ الى ١١ بالتفصيل الممل في الدورس السابقه. في السطر الثاني عشر if (num1 == num2) **ومعناها اذا تحقق ان num1=num2 نقوم بتطبيق السطر ١٣ وأكيد انكم لاحظتم عدم وجود القوسين{ } وهذا لأنه اذا كان لدينا فقط سطر واحد في داخل ال if لا داعي لوضع هذين القوسين و لا مشكلة اذا وضعتهما اما اذا كان لديك اكثر من سطر وجب عليك وضعهما ,للأن المترجم اذا احتوت ال ifاكثر من سطر ولم تضعهما داخل القوسين { } سيقوم بأخذ اول سطر ضمن ال if والآخرين يحسبهما خارج ال if ويتعامل معهما على هذه الأساس.**

وبقية السطور في البرنامج السابق تقارن بالمثل ولكن بأختلاف وجه المقارنه .

هكذا سيكون الناتج للبرنامج بعد تجريبه (الناتج على الشاشه):

```
Enter two integers, and I will tell you
the relationships they satisfy:
4
5
٤ is not equal to 5
٤  is less than 5
٤ is less than or equal to 5
```

أرجو ان تكون قد وضحت الصورة بالنسبة لل if.وننتقل لل if /else .

ثانيا: if/else

```
if (expression) {
code inside if;
    }

else {
code inside else;
}
```

نلاحظ في السابق عند استخدام ال if بمفردها اذا لم يتحقق الشرط يتجاوز المترجم ال if وما بداخلها من كود ويتابع طريقه اما في حالة وجود ال else فهنا كما ترون بالشكل العام لل if/else يمر المترجم بال if يرى الشرط اذا تحقق الشرط يطبق ما بداخل ال if واذا لم يتحقق يطبق ما بداخل ال else قد تسائلون اذا ما الفرق بينه وبين ما سبق شرحه في ال if طبعا الفرق بهذا المثال :برنامج يطلب من المستخدم ان يدخل رقما, اذا ادخل المستخدم رقم ٣ نطبع على الشاشه hello غير ذلك نطبع error .نلاحظ هنا اننا سنجعل if للمقارنه اذا كان الرقم الذي ادخله المستخدم ٣ لنطبع hello , غير ذلك نطبع error وسيكون البرنامج كالتالي....

```
#include <iostream>

using std::cout;  // program uses cout
using std::cin;   // program uses cin

// function main begins program execution
int main()
{
   int num1;  // first number to be read from user


   cout << "Enter the integer number \n";
   cin >> num1;   // read the integer


 if ( num1 == 3)
      cout<<"hello";

else
    cout<<"error";

   return 0;   // indicate that program ended successful

} // end function main
```

نلاحظ في البرنامج السابق اذا ادخل المستخدم ٣ سيطبع hello اما اذا ادخل اي رقم آخر مهما كانت قيمته وهذا ما قصدناه ب else سيطبع error . أرجو ان يكون المعنى واضح.

ثالثا: else if

هذه الجملة مفيده جدا فهي تجعل برنامجك اكثر ذكاء وحنكه والشكل العام لهذه الجملة كالتالي:-

```
if (expression) {
 do the code inside if;
}

else if (expression) {

do the code inside else if;
}


else if (expression) {
do the code inside else if;
}


else {
do the code inside else;
}
```

هنا المترجم سيقوم بتاكد من القاعدة التي بداخل ال if اذا تطبقت نفذها ولا يمر على باقي الخيارات المتمثلة في ال if else وال else, اذا لم تطبق القاعده يذهب للخيار الآخر في ال else if واذا لم يتطبق يذهب للذي بعده واخيرا اذا لم يتطبق شيء ينفذ ال else.اي انه اذا طبق احد الخيارات يخرج من هذه المقارنه وطبعا هنا يجب ان تلاحظ لو انك استبدلت ال else if ب if لتم المرور على كل if والتاكد من قاعدتها وتطبيقها اذا تحققت القاعده لكن في حالة ال else if يمر على ال if الاولى اذا تحققت يتجاوز الجميع ويخرج لما وراء ال else اما اذا لم تتحقق يجرب البقية وبمجرد تحقق احداهم يخرج لذلك يكون البرنامج اسرع بال else if فلا داعي لمقارنة البقية مادام تحقق شرط من الشروط والمثال التالي لتوضيح الموضوع أكثر.
لاحظ في هذا البرنامج يمر المترجم على ال `if (num1 > num2 )` اذا تحقق الشرط يكتب العبارة `The number1 is bigger than number2` وينتقل الى ما بعد ال else اي لا يقارن المقارنات الأخرى,اما اذا لم يتحقق ينتقل الى `else if (num2 > num1)` ويطبق عليه نفس ما قلناه في الأولى وهكذا اذا لم تتحقق القاعده فيهم جميعا نصل الى ال else وننفذ ما بها ويطبع `error` .

```
1- #include <iostream>
2- int main()
3- {
4- int num1,num2;
5- cout<<"please enter two integer numbers";
6- cin >> num1 >> num2 ;
```

```
7- if (num1 > num2 ) {
8- cout << "The number1 is bigger than number2" ;
9- }

10- else if (num2 > num1) {
11- cout << "The number2 is bigger than number1" ;
12- }

13- else if ( num2=num1) {
14- cout << "there is no bigger number" ;}

15- else { cout << "error" ; }
16- return 0;
17- }
```

هنا نصل لنهاية درس اليوم اتمنى ان يكون واضح ومفيد وسننتقل باذن الله في الدرس القادم الى مرحلة اكثر تطورا في ال **C++** *وسنقوم بحل تمارين على درس اليوم وأي استفسار يسرني الاجابه عليه....*

°^~*¤®§(*§سبحانك اللهم وبحمدك أشهد ان لا اله الاانت استغفرك وأتوب أليك §*)§®¤*~^°

`~'*¤!||!¤*'~`((كاتبة الدرس .....بنت النور.....عفوا يمنع اجراء اي تعديل للملف))`~'*¤!||!¤*'~`

`~'*¤!||!¤*'~`((نسألكم دعوة صادقة لأختكم في ظهر الغيب))`~'*¤!||!¤*'~`

السلام عليكم ورحمة الله وبركاته.....
الحمدلله والصلاة والسلام على افضل الأنبياء والمرسلين قدوتنا وحبيبنا وشفيعنا ان شاء الله محمد بن عبد الله وعلى اله وصحبه اجمعين.....

اتمنى ان تكونوا بصح جيدة .....بعد ان انتهينا من الدرس الرابع نأخذ هذة التمارين لنعزز ما تم دراسته وبعدها ننتقل للدرس الخامس.....

Question1: write a program that reads an integer and determine whether it is odd or even. (Hint: use the modulus operator .an even number is a multiple of two. any multiple of two leaves a reminder of zero when divide by 2).

السؤال الأول: قم بكتابة برنامج بأستخدام لغة الC++ يقوم بأدخال عدد صحيح وتحديد اذا كان هذا العدد الصحيح عدد فردي ام زوجي(ملاحظه: استخدم % لتساعدك في الحل واي عدد زوجي يقسم على ٢ يكون الباقي دائما صفرا)
بهذه الملاحظه اصبحت الفكرة جاهزة لكم .....

```
#include <iostream.h >
int main ( )
{

int number ;

cout << " please enter the number : " ;
cin >>number ;

if (number % 2 == 0 )
cout <<"  \n The number  is even " ;

else (number % 2 == 1 )
cout <<  " \n The number is odd " ;

return 0 ;

}
```

Questions2 :write a program that inputs one five digit number, separates the number in to individuals digits and prints the digits separated from one another by three spaces each .(hint :use combinations of integer division and the reminder operation)

Example: enter a five digit number: 12345
1 2 3 4 5

السؤال الثاني:قم بكتابة برنامج بأستخدام لغة ال C++ يقوم بأدخال عدد مكون من خمسة أرقام ويفصلهما ويطبع كل عدد مفصول عن الذي يليه بثلاث مسافات(ملاحظه استخدم /, %) والمثال التالي للتوضيح.....

Example: enter a five digit number: 12345
1 2 3 4 5

```
#include <iostream.h >
int main ( )
{

int number,num1,num2,num3,num4,num5 ;

cout << " please enter a number  with five digit: " ;
cin >>number ;

number =number%10000;
num1= number/10000;


number =number%1000;
num2= number/1000;



number =number%100;
num3= number/100;


number =number%10;
num4= number/10;

num5= number;

cout<<num1<</t<<num2<</t<<num3<</t<<num4<</t<<num5<<end1;


return 0 ;

}
```

الدرس الخامس:

نتناول اليوم في درسنا هذا ال switch وهي مشابهة لل else if التي سبق ودرسناها وفيه اختلاف بسيط لكن لة تأثير على كتابة البرنامج سنتطرق له ان شاء الله والشكل التالي هو الشكل العام للأستخدام ال switch

```
switch (هنا يكتب المتغير المراد مقارنته) {
case هنا تكتب المقارنة الأولى: do the following code ;

case هنا تكتب المقارنة الثانية: do the following code ;
.
.
default: do the following code;
}
```

لاحظ ان الشكل العام لل switch موضح علية مكان وضع المتغير المراد مقارنته والحالات التي يقارن به هذا المتغير وستتضح الفكرة اكثر عندما نأخذ مثال عملي على ال switch ننتقل لمثال عملي .....

المثال١: برنامج يقوم بأدخال أرقام صحيحه ويقوم بطباعتها بالحروف .مثلا عندما يدخل المستخدم ١ يقوم البرنامج بطباعة one. وهكذا طبعا سنجعله الى رقم خمسة فقط لتوضيح فكرة عمل ال switch .

```
1- #include <iostream>
2- int main()
3- {
4- int number;
5- cout << "Enter Number:\t" ;
6- cin >> number;

7- switch (number) {
8- case '1':
9-    cout<<"one";
10- break;
11-case '2':
12-   cout<<"two";
13- break;
14-case '3':
15-   cout<<"three";
16-break;
17-case '4':
18-   cout<<"four";
19- break;
20 case '5':
21-   cout<<"five";
22-break;
23-default:
24-   cout<<"error";

25- } //end the switch
26-    return 0;
27-             } // end the main
```

سنبدا شرح البرنامج من السطر السابع لأن السطور السابقه تم شرحها سابقا,لاحظ بعد ان ادخل المستخدم الرقم وخزن في متغير اسمه number هنا جعلنا ال number داخل ال switch لأنه هو الذي سنقارنه,سيمر المترجم بأول حالة وهي case '1' اذا تطابقت قيمة المتغير number مع ١ فأنه يطبق ما يندرج تحت هذا ال case فيطبع على الشاشة one وبعدها ينتقل ال break وهذه الجملة تجعل المترجم يخرج خارج الswitch وهذا ما يجعل ال switch اكثر فائده.فلو انك لم تضع جملة break بعد كل case لمر المترجم على كل case حتى ولو وجد ظالته في ال case السابق وهذه طبعا اضاعة للوقت ناهيك انه سيطبق ال case الذي يتطابق معه

بالأضافة الى خيار ال default لأن ال default هنا لا يعني استثناءا كما تعني ال else بل هي عامة اي سيتم تنفيذها في الحالتين لذلك انصحكم باتباع هذه العاده الحسنه وهي وضع break بعد كل case كما فعلت في المثال السابق بهذه الطريقة يكون برنامجك محكم اكثر وسليم من المفاجآت الغير سارة طبعا.

في مثالنا لو ان المستخدم ادخل ٢ يمر المترجم بأول حالة وطبعا ال١ لا يساوي ال ٢ ويتجه للحالة الثانيه وفعلا يتطابقا فيطبع على الشاشة two ويخرج خارج ال switch طبعا لوجود ال break.

وبهذا نكون انهينا درس اليوم وهو ال switch اتمنى ان يكون واضح ومفهوم..وعليكم عجل السؤال التالي لتطبقوا ما درسناه للتو....

السؤال: قم بكتابة برنامج بأستخدام لغة السي بلس بلس يقوم بعمل آلة حاسبة ,يستقبل من المستخدم رقمين والعملية (*,-,+,/) ويقوم باجراء العملية التي يدخلها المستخدم على العددين ويطبع الناتج على الشاشة..

سيرفق الحل في الدرس القادم ان شاء الله :)

°^~*¤®§(*§سبحانك اللهم وبحمدك أشهد ان لا اله الاانت استغفرك وأتوب أليك §*)§®¤*~^°

`~'*¤!||!¤*'~`((كاتبة الدرس .....بنت النور.....عفوا يمنع اجراء اي تعديل للملف))`~'*¤!||!¤*'~`

`~'*¤!||!¤*'~`((نسألكم دعوة صادقة لأختكم في ظهر الغيب))`~'*¤!||!¤*'~`

السلام عليكم ورحمة الله وبركاته.....
الحمدلله والصلاة والسلام على افضل الأنبياء والمرسلين قدوتنا وحبيبنا وشفيعنا ان شاء الله محمد بن عبد الله وعلى اله وصحبه اجمعين.....

كيف حالكم أخواني واخواتي اسأل الله ان تكونوا في أحسن حال.....اليوم بعد ان انتهينا من جملة ال switch ننتقل بأذن الله الى موضوع مهم وعملي جدا...الا وهو جمل التكرار وتتمثل في ال,for loop ,while do while وتستخدم هذه الجمل بدرجة واسعه في البرمجة بل معظم البرامج تحتوي عليها.

تخيل انه طلب منك عمل برنامج لأدخال بيانات موظفين هذا البرنامج يقوم بأدخال الاسم و رقم التسلسل للموظف ووظيفته وعدد ساعات عمله طبعا هنا ستحتاج تطلب من المستخدم اسمه ورقم التسلسل التابع له ووظيفته و عدد ساعاته وهنا سيكون برنامجك يحوي اربع cout واربع cin لكل موظف تخيل لو انه طلب منك كتابة هذه البرنامج لمئة موظف او اكثر طبعا المسألة غير مقبولة اذا استخدمت هذه الطريقة لأن البرنامج سيكون طويل جدا وممل جدا وبدون معنى ,فهنا يأتي دور جمل التكرار التي ستنقذك من هذا الموقف الحرج ومن كتابة كل هذا الكلام.وسنشرحهم ان شاء الله بالتفصيل.
نصيحة ذهبية : المبرمج الناجح حينما يكتب برنامج يجعله مرن بحيث يستخدم لأي استخدام للغرض الذي انشأ من اجله مثلا لو طلب منك عمل برنامج يحسب المتوسط طبعا يجب ان تجعله برنامج يحسب المتوسط لأي عدد من الدرجات ايا كان عددها وهكذا لك يستخدم في اي وقت وحاول تختار اسماء متغيرات واضحه وذات معني اي اذا اردت ان تختار متغير تخزن فيه الدرجات اجعل اسمه grade اي درجه لك تتذكر هدفه ومعناه اذا رجعت له بعد زمن او اردت تطويره.....
أعتذر على الأطاله ننتقل مباشره لأول جمل التكرار وهي ال while.

جملة ال while:

نبدا الآن بالشكل العام لل while ....

```
while ( هنا يكتب الشرط) {

هنا تكتب الكود الذي تريد ان يطبقه البرنامج عند توفر الشرط فيه

}
```

لاحظ عند استخدام ال while يجب ان تستخدم الشكل السابق ,بين القوسين تكتب الشرط لدخول ال while ويجب ان تبدأ ال while بقوس مفتوح وتنهيها بقوس مغلق كما هو موضح بالصورة ,وبين قوسين البداية والنهاية تكتب الكود الذي سيطبقه البرنامج عند توفر الشرط لدخول الwhile.
طبعا لك نفهم اكثر ننتقل لمثال عملي ستفهمون منه ان شاء الله كيفية التعامل مع ال while أكثر.
-المطلوب كتابة برنامج يقوم بجمع الاعداد من ١ الى ١٠ ,يعني ١+٢+٣+٤+٥+.....+١٠.

```
1-#include <iostream>

2-using std::cout;
3-using std::endl;

4-int main()
5-{
6-   int sum=0;
7-   int x=1;
```

```
8-    while ( x <= 10 ) {
9-       sum =sum+x;
10-       x=x+1;

11-    } // end while

12-    cout << "The sum is: " << sum << endl;

13-    return 0;

14-            } // end function main
```

ننتقل الآن لشرح البرنامج:
-لاحظ في السطر السادس قمنا بحجز مساحه لنحفظ فيها المجموع واسميناها sum ونوعها integer لأن الناتج مع لأعداد صحيحه اكيد لن يكون الا عدد صحيح ,يمكن تتسائل لماذا جعلنا قيمة ال sum =0 هنا صفرنا الخانه التي سنخزن فيها المجموع لأننا نحتاج لحفظ المجموع خانه خالية من اي رقم لك نحصل على ناتج صحيح وفي العاده الذاكره يكون موجود فيها ١ أو ٠ لذلك لا نريد ان ندع المسأله للحظ اذا كان صفر حصلنا على ناتج صحيح واذا كان ١ كان الناتج غير صحيح .نصفر الخانه ونكون في الجانب الآمن.
خذوا هذه العادة الحسنه: عندما تريدون ان تحجزوا مساحه لحفظ الناتج بداخلها في البداية عند تعريفها قوموا بتصفيرها كما فعلنا الآن.

- في السطر السابع قمنا بحجز مساحه لمتغير اسميناه x ونوعه عدد صحيح,وهذا المتغير سيكون لنا بمثابة المؤشر الذي يؤشر على العدد الذي نصل له بعد جمع اي عدد وطبعا جعلنا قيمته المبدئيه بواحد لأننا نريد جمع الأرقام من واحد الى ١٠ . ولو اردنا جمع الأعداد من ٣ الى ١٠ لجعلنا قيمته المبدئية ٣ .

-السطر السابع: هنا نبتدي ال while لاحظ يبدأ المترجم في البداية بالتأكد من الشرط , والشرط هنا(x <= 10) ,سيقارن المترجم قيمة الx وهي حاليا واحد وسينتج ان الشرط تحقق لأن الواحد أقل من ال عشرة . فسيدخل التكرار ويطبق السطور الكودية التي بداخل القوسين للwhile .

-السطر الثامن : هنا سيقوم البرنامج بجمع العدد الذي بالخانه المسماه x مع قيمة ال sum وسينتج ٠+١=١ وستتغير قيمة ال sum الى واحد .
بعدها ينتقل البرنامج للسطر التاسع وقوم بزيادة قيمة المتغير x بمقدار واحد حتى ننتقل للرقم ٢ ونقوم بجمعه مرة أخرى .لاحظ لن يخرج البرنامج من ال while اولا سيقارن الشرط وفعلا ال٢ اقل من العشرة فسيدخل التكرار مرة اخرى ويطبق السطور التي بداخل ال while, فيجمع الناتج وهو الآن يحتوي القيمة واحد بالقيمة التي في المتغير x ويضيف القيمة الجديدة وهي ٣ في المتغير sum وينزل للسطر الذي يليه وهو السطر التاسع ويزيد قيمة ال x بمقدار واحد فتصبح قيمة ال x =3 ثم يرجع البرنامج لل while ويقارن الشرط مرة اخرى وفعلا يتحقق لأن ال ٣ أقل من ١٠ ويدخل مرة اخرى داخل ال while ويطبق السطور الكودية التي بداخل ال while ويستمر هكذا الى ان تصل قيمة ال x الى عشرة يقارنها بالشرط فيتحقق الشرط فيطبق الأسطر الكودية التي بداخل ال while هنا ستصل قيمة ال sum =45 يضيف لها قيمة ال x الحالية وهي ١٠ فيصبح لدينا ١٠+٤٥ =٥٥ فتتغير قيمة الsum=55 بعدها تزيد قيمة ال x

مقدار واحد فتصبح x=11 يرجع البرنامج ويقارن الشرط هنا لا يتطبق الشرط فيخرج البرنامج من ال while ويذهب الى السطر ١٢ الناتج على الشاشة ٥٥:The sum is.
وهكذا ينتهي البرنامج وبهذا النمط والتكرار تكون طريقة سير البرنامج بأستخدام ال while اتمنى ان تكون قد وصلت الفكرة وسأسرد أمثلة اخرى لتوضيح عمل ال while ولكن لن نفصل سير البرنامج بهذا التفصيل السابق لأننا قد سردنا التفصيل في المثال الأول ...

مثال ٢: برنامج يقوم بطباعة مضروب العدد في نفسة من واحد الى عشرة وطباعة مجموعهم.
مثال الناتج على الشاشة سيكون كالتالي :

```
١
٤
٩
١٦
٢٥
٣٦
٤٩
٦٤
٨١
١٠٠
The total is :385
```

الحل:

```
#include <iostream>

using std::cout;
using std::endl;

int main()
{
   int y;           // declare y
   int x = 1;       // initialize x
   int total = 0;   // initialize total

   while ( x <= 10 ) {    // loop 10 times
      y = x * x;          // perform calculation
      cout << y << endl;  // output result
      total += y;         // add y to total
      ++x;                // increment counter x

   } // end while

   cout << "Total is " << total << endl;  // display result

   return 0;  // indicate successful termination

} // end function main
```

ملاحظه : التعليقات التي بعد علامة // مجرد تعليق لك انت اي ان البرنامج لا يعطيه اي اهميه وقد وضحت هذه النقطه في الدروس السابقه لكن اعادة هذه اللاحظه للتنبية لا تضر شيئا.

كما تعودنا اليكم هذا السؤال الذي اتمنى ان تحاولوا ان تجدوا الحل له و هو سهل جدا وسأضيف اجابته في الدرس القادم ان شاء الله.

- قم بكتابة برنامج بأستخدام لغة السي ++ يقوم بطلب الاساس والاس من المستخدم وايجاد الناتج . مثال

```
Enter base as an integer:
٣
Enter exponent as an integer
٢
Power =9
```

السؤال سهل اليس كذلك ☺
هنا نصل الى نهاية الدرس اليوم وفي الدرس القادم سنأخذ امثلة اكثر على ال while وتمارين أعمق .فيجب ان تتمكن من استخدامها جيدا للأهميتها البالغة في برمجة السي ++.

°^~*¤®§(*§سبحانك اللهم وبحمدك أشهد ان لا اله الاانت استغفرك وأتوب أليك §*)§®¤*~^°

`~'*¤!||!¤*'~`((كاتبة الدرس .....بنت النور.....عفوا يمنع اجراء اي تعديل للملف))`~'*¤!||!¤*'~`

`~'*¤!||!¤*'~`((نسألكم دعوة صادقة لأختكم في ظهر الغيب))`~'*¤!||!¤*'~`

بسم الله الرحمن الرحيم

السلام عليكم ورحمة الله وبركاته.....
الحمدلله والصلاة والسلام على افضل الأنبياء والمرسلين قدوتنا وحبيبنا وشفيعنا ان شاء الله محمد بن عبد الله وعلى اله وصحبه اجمعين.....

أجمل تحية لكم اخوتي واخواتي في الله .....
اتهينا في الدرس السابق من شرح ال whileوذكرنا مثالين عليها وقمنا بطرح سؤال والآن ان شاء الله نواصل الجزء الثاني من درس ال while حيث سنتدرب على برامج أكثر بأستخدام ال while.

نبدأ بحل السؤال الذي ذكرناه في الدرس السابق .....

السؤال ١:
- قم بكتابة برنامج بأستخدام لغة السي ++ يقوم بطلب الاساس والاس من المستخدم وايجاد الناتج . مثال

```
Enter base as an integer:
٣
Enter exponent as an integer
٢
Power =9
```

الجواب :

```
#include <iostream>

using std::cout;
using std::cin;
using std::endl;


int main()
{
   int x;       // base
   int y;       // exponent
   int i;       // counts from 1 to y
   int power;   // used to calculate x raised to power y

   i = 1;       // initialize i to begin counting from 1
   power = 1;   // initialize power

   cout << "Enter base as an integer: ";  // prompt for base
   cin >> x;                              // input base


   cout << "Enter exponent as an integer: ";
   cin >> y;                              // input exponent

   // count from 1 to y and multiply power by x each time
   while ( i <= y ) {
      power *= x;
      ++i;

   } // end while

   cout <<"power="<<power<< endl;  // display result
```

```
   return 0;

} // end function main
```

ملاحظه : التعليقات التي بعد علامة // مجرد تعليق لك انت اي ان البرنامج لا يعطيه اي اهميه وقد وضحت هذه النقطه في الدروس السابقه **يعني ليست من ضمن الكود للبرنامج ...**

تستطيع استخدام تكرار داخل تكرار اي while loop داخل while loop والمثال التالي سيوضح الفكره.وايضا ينقسم التكرار في ال while الى محدد (ثابت لعدد معين) وغير محدد يحدده المستخدم فالرقم للتكرار في الغير المحدد ليس ثابت يتحكم فيه المستخدم وهذا الأفضل حيث يكون اكثر مرونه .وسنوضح الحالتين سنبدأبالمحدد ومن ثم الغير محدد.

السؤال ٢(التكرار المحدد) :

مطلوب عمل برنامج بأستخدام لغة السي++ يقوم هذا البرنامج بحساب المتوسط لدرجات ٥٠٠ طالب ويجب على البرنامج ان يطلب من المستخدم ادخال الدرجات ويتأكد البرنامج من ان الدرجه التي تم ادخالها بين الصفر وال١٠٠ ولا تتجاوز هذا الحد.

```
#include <iostream>

using std::cout;
using std::cin;
using std::endl;


int main()
{
Float grade,total=0,avg;
int count=1;


While(count<=500)
 {
cout<<"please enter the grade";
cin>>grade;

       while(grade<0 || grade>100)
          {

          Cout<<"sorry you enter invalid grade, enter grade 0-100;
          Cin>>grade;

          } //second while


       total+=grade;

       count++;

  }//first while
```

```
avg=total/500;

Cout<<"the avg is "<<avg;

} // end function main
```

الشرح:
نلاحظ في هذا السؤال العدد محدد ل٥٠٠ طالب اي ان التكرار سيكون ٥٠٠ مرة سنقوم بشرح الاجزاء التي تتعلق بدرسنا اليوم والبقية شرحناهم في دروس سابقه.
لاحظ في البداية حجزنا خانة لمتغير اسميناه ال count لك يسير لنا العد ففي البدايه كانت قيمته واحد فمر على ال while تأكد من تحقق الشرط والشرط هو ان قيمة ال count لا تزيد عن ٥٠٠ لأننا نريد التكرار ٥٠٠ مره فقط,بعدها يطلب البرنامج من المستخدم ان يدخل له الدرجه ومن ثم ينتقل الى ال while الداخليه التي يقوم بالدخول لها اذا تحقق شرطها وهو ان تكون الدرجه التي ادخلها المستخدم اكبر من مئه او اصغر من صفر ,طبعا اذا تحقق هذا الشرط هذا يعني ان المستخدم اخل درجه خاطئه فيطلب منه البرنامج اعادة ادخال الدرجه كما ترون داخل ال while الداخلية ولا ينتقل خارج ال while الداخلية الا اذا ادخل درجه صحيحه هنا نضمن ان جميع الدرجات صحيحه ,فأذا ادخل المستخدم درجه في حدود ١٠٠-٠ هنا ينتقل المترجم الى جمع الدرجه للحصول على المجموع وقد شرحنا هذا بالتفصيل في درس سابق ومن ثم يزيد قيمة ال count بمقدار واحد.
ربما يتبادر الى ذهنكم ما هذا total+=grade; طبعا هذا يساوي total=total+grade ولكن اختصارا كتبناها بالطريقة السابقه وايضا count++; تعني count =count+1; ولكن اختصرناها بالطريق السابقة .
دعونا نتدرب على هذه الاختصارات:

count--; تعني count =count-1;
Sum*=grade تعني sum=sum*grade;
أرجو ان تكون قد وضحت الفكره.....نعود الى برنامجنا.

بعد ان يجمع البرنامج الدرجه في الtotal ويزيد قيمة ال count بمقدار واحد يعيد هذه الخطوات الى ان يصبح مقار ال count=500 عندها يخرج من الwhile ويذهب ليقوم بقسم المجموع للدرجات على ٥٠٠ للحصول على المتوسط ومن ثم يقوم بطباعته ...
لاحظ جعلنا العمليه الحسابية لأيجاد المتوسط خارج ال while لأننا نريد المجموع للدرجات واذا وضعنا العملية لأيجاد المتوسط avg=total/500; ووضعنا ايضا cout<<"the average is "<<avg;

هنا سيقوم البرنامج بأيجاد المتوسط وطباعته في كل مرة يدخل فيها لل while يعني خمسمائة مرة وطبعا هذا لا يعطينا المطلوب لذلك وضعناه بالخارج لك عندما ينتهي البرنامج من الحصول على المجموع يقوم بأيجاد المتوسط وطباعته ....وهنا سيطبع مره واحده.

اتمنى ان تكونوا قد فهمتم كيف يكون التكرار محدد وهو يتمثل في تحديد قيمة للتكرار وفي مثالنا هذا كانت ٥٠٠ وجعلنا ال count كمؤشر يعد لنا ٥٠٠ مرة.

مطلوب عمل برنامج بأستخدام لغة السي++ يقوم هذا البرنامج بحساب المتوسط لدرجات عدد غير محدد من الطلاب يتحكم في هذا العدد المستخدم ويجب على البرنامج ان يطلب من المستخدم ادخال الدرجات ويتأكد البرنامج من ان الدرجه التي تم ادخالها بين الصفر وال١٠٠ ولا تتجاوز هذا الحد.

```
#include <iostream>

using std::cout;
using std::cin;
using std::endl;


int main()
{
 Float grade,total=0,avg;
  int count=0;

cout<<"please enter the student grade(enter -1 to end)";
cin>>grade;

 while(grade!=-1)
{


       while(grade<0 || grade>100)
          {

          Cout<<"sorry you enter invalid grade, enter grade 0-100;
          Cin>>grade;

          } //second while


       total+=grade;

       count++;



cout<<"please enter the student grade(enter -1 to end)";
cin>>grade;


  }//first while

Arg=total/count;

Cout<<"the avg is "<<avg;

} // end function main
```

الشرح:

نلاحظ في هذا السؤال التكرار غير محدد اي يدخل المستخدم درجات الى ان يكتفي ويطلب انهاء البرنامج...

تابع معي البرنامج,في البداية سيطلب البرنامج من المستخدم ادخال اول درجة أو ١- لأنهاء البرنامج هنا جعلنا الأشارة التي نعرف اذا اكتفى المستخدم من أدخال الدرجات ام لا هي ١- ,أي عندما ينتهي من ادال الدرجات وتظهر له رساله ادخال الدرجه يدخل ١- وسينتهي البرنامج ويعطيه المتوسط.

وأخترنا -١ لأن لا يمكن ان توجد درجه بال -١ وتستطيع اختيار اي شيء ماعدا ان يكون من ضمن الدرجات ٠-١٠٠.

بعد ان يدخل المستخدم درجة الطالب يقوم البرنامج في ال while بالتأكد من ان الدرجة ليست -١ واذا كانت -١ يقف البرنامج ويعطي المتوسط .اذا لم تكن -١ يدخل البرنامج داخل ال while الأولى ومن ثم يدخل داخل ال while الثانية التي تتأكد من ان الدرجه من ٠-١٠٠ وان ادخل المستخدم درجه خارج هذا النطاق يبقى البرنامج داخل ال while ويطلب من المستخدم ادخال درجه صحيحه ومن ثم بعد ادخال درجة صحيحه ينتقل البرنامج الى جمع الدرجه مع المجموع وزيادة ال count بمقدار واحد. قد تسأل نفسك هنا لا نحتاج count لتسيير البرنامج وتحديد قيمة التكرار لن العدد غير محدد والمستخدم هو الذي يحدد العدد .لكننا وضعنا ال count هنا لأننا نريد معرفة كم عدد الدرجات التي ادخلت لأننا نحتاج قيمة ال count لحساب المتوسط لأن المتوسط عباره عن مجموع الدرجات على عددها.مجموع الدرجات نحصل عليه من الtotal وعددها من ال count بعدها وضعنا

```
cout<<"please enter the student grade(enter -1 to end)";
cin>>grade;
```

تخيل لو اننا لم نضع هذه الجمله اخر ال while الأولى لكان البرنامج بعد ان ينتهي من زيادة ال count بواحد لرجع لل while الأولى وطبعا قيمتة ال gradeستبقى قيمة ال grade الأولى ولن تتغير وسيكون البرنامج في تكرار غير منتهي مما يسبب مشكلة كبيرة ولن يعطينا المتوسط ابدا..لأننا لم نعطي المستخدم مجال لأدخال درجة أخرى.

لكن مع وضع هذين السطرين يسطلب البرنامج من المستخدم ادخال درجة جديده ويرجع للwhile الأولى ويتأكد ما اذا كان الشرط صحيح وان الدرجه لا تساوي -١ واذا كانت -١ يتوقف ويعطي المتوسط.اما اذا كانت لا تساوي -١ دخل البرنامج الى ال whileالثانية وتأكد من ان الدرجه في النطاق ٠-١٠٠ وبعدها جمع الدرجه الجديده مع المجموع السابق وزاد قيمة ال count بواحد ويطلب البرنامج مرة اخرى من المستخدم ادخال درجة جديده فأذا كانت -١ توقف واعطى المتوسط واذا كانت غير ذلك واصل ما قلناه سابقا وهكذا يستمر البرنامج حتى يدخل المستخدم -١ .فقط عندها يتوقف ويعطي المتوسط والا يبقى مستمر في أخذ الدرجات وجمعها .

بعد ادخال -١ يخرج البرنامج من ال while ويقسم المجموع للدرجات على عددها الذي يتمثل في ال count ويعطي المتوسط ويطبعه على الشاشة.

الفرق بين المثال الأول والثاني اي بين التكرار المحدد والغير محدد ان في الأول قيمة التكرار تحدد في ال while الأولى أما في الثاني الشرط الذي يوقف البرنامج يوضع في ال while الأولى وطبعا في التكرار الغير محدد لا تنسى من وضح cout لك تخبر المستخدم بأدخال قيمة جديده وcin لك تحفظ هذه القيمة في المتغير المخصص لها كما فعلنا في هذين السطرين

```
cout<<"please enter the student grade(enter -1 to end)";
cin>>grade;
```

أرجو ان يكون الفرق واضح لديكم .....

الآن ننتقل لمثال قمنا بحله سابقا لكن بدون استخدام التكرار وكان لأدخال رقم واحد فقط طبعا كان برنامج غير مرن للأنه بعد ان يعطيك نتيجة عدد واحد ادخلته ينتهي عمله انظروا للبرنامج بعد تطويره بأستخدام التكرار وقارنوا بينهم وانظروا الفرق.....

Question1: write a program that reads integers and determine whether it is odd or even. (Hint: use the modulus operator .an even number is a multiple of two. any multiple of two leaves a reminder of zero when divide by 2).
Make sure to make this program for non fix number.

السؤال ٣: قم بكتابة برنامج بأستخدام لغة الC++ يقوم بأدخال اعداد صحيحه وفي كل مره يدخل المستخدم عدد يقوم البرنامج بتحديد اذا كان العدد فردي او زوجي (ملاحظه: استخدم % لتساعدك في الحل واي عدد زوجي يقسم على ٢ يكون الباقي دائما صفرا)
بهذه الملاحظه اصبحت الفكرة جاهزة لكم .....

```
#include <iostream.h >
int main ( )
{
char flag=y
int number ;

while(flag!=n)
{
cout << " please enter the number : " ;
cin >>number ;

if (number % 2 == 0 )
cout <<"  \n The number  is even " ;

else (number % 2 == 1 )
cout <<  " \n The number is odd " ;

cout<<"do u want to check another number,y=yes,n=no"<<end1;
cin>>flag;

  while(flag!=y && flag !=n)
    {
      cout<<"sorry rong command";
      cout<<"do u want to check another number,y=yes,n=no"<<end1;
      cin>>flag;

     }

  }
```

```
return 0 ;

}
```

نصل هنا لنهاية الدرس السابع وانهاء دراسة ال whileارجو ان تكونوا قد فهمتوها جيدا فهي جملة مفيده جدا وسننتقل في الدرس القادم بأذن الله الى الfor loop ..

الى ذلك الحين دمتم بخير وعافيه والسلام عليكم ورحمة الله وبركاته.

°^~*¤®§(*§سبحانك اللهم وبحمدك أشهد ان لا اله الانت استغفرك وأتوب أليك §*)§®¤*~^°

`~'*¤!||!¤*'~`((كاتبة الدرس .....بنت النور.....عفوا يمنع اجراء اي تعديل للملف))`~'*¤!||!¤*'~`

`~'*¤!||!¤*'~`((نسألكم دعوة صادقة لأختكم في ظهر الغيب))`~'*¤!||!¤*'~`

بسم الله الرحمن الرحيم

السلام عليكم ورحمة الله وبركاته.....
الحمدلله والصلاة والسلام على افضل الأنبياء والمرسلين قدوتنا وحبيبنا وشفيعنا ان شاء الله محمد بن عبد الله وعلى اله وصحبه اجمعين.....

أجمل تحية لكم اخوتي واخواتي في الله .....

نبدأ اليوم ان شاء الله بشرح جمله من اروع واشهر جمل التكرار واكثرهم استخداما وهي الfor loop..هي جمله تأخذ الشكل التالي..

for (initial value ; condition ; increment or decrement)
{ your statment}

لاحظ في الشكل العام لل for يتخلل كل جزء علامة ; يجب ان تضعها كما وضحتها بالشكل وداخل القوسين تضع الاسطر التي تريد ان ينفذها البرنامج في كل مره يدخل فيها التكرار.

١- (initial value) :هنا نكتب نقطة البداية التي سينطلق منها العداد في الجمله for لاحظ في هذا الجزء تستطيع ان تعرف المتغير الذي ستستخدمه كعداد وتستطيع ايضا اعطائه قيمة ابتدائيه مثال: int i=0
لاحظ في هذا المثال عرفنا المتغير i بانه عدد صحيح int وايضا اعطيناه قيمه ابتدائيه وهي ٠.

٢- (condition) : هنا تكتب القاعده التي يتوقف عندها العد عند الوصول لها مثال: i<=10 وهنا يتوقف العد عندما يصل الى عدد اكبر من ١٠ يعني يستمر في العد طالما ال i اقل من او يساوي العشره.

٣- (increment or decrement) : هنا نكتب كيف نريد ان يسير العد تصاعدي او تنازلي اي increment or decrement مثال i++ يزيد قيمة ال i كل مره بمقدار واحد و i-- يقلل قيمة ال i بمقدار واحد.

في المثال التالي ستتضح هذه الجمله اكثر :
وفي هذا المثال برنامج يقوم بطاعة الاعداد من ١ الى ١٠ على النحو التالي :

١
٢
٣
٤
٥
٦
٧
٨
٩
١٠

والان ننتقل الى البرنامج<<<<<<

```
#include <iostream>

using std::cout;
using std::endl;

int main()
{
   for ( int counter = 1; counter <= 10; counter++ )
      cout << counter << endl;

   return 0;

}
```

شرح البرنامج :
كتابة البرنامج لا تتجاوز سطور قليلة مقارنة بكتابته باستخدام ال while والبرنامج بكامله يتمثل في السطر التالي
for (int counter = 1; counter <= 10; counter++)
نلاحظ كما شرحنا في البداية الشكل العام لجملة ال for سنستخدم للعد المتغير counter وسنجعل قيمته الايتدائية واحد1 = int counter, وبعدها جعلنا القاعده التي يتوقف عندها هذا العد عندما يتجاوز العد الرقم عشرة counter <= 10,وطبعا نزيد قيمة الcounter بمقدار واحد في كل مره.
تابع معي سيبدأ العد وقيمة ال counter =1 عندها يتاكد من القاعده وسيجد ان ال counter<10 وسيزيد قيمة الcounter بمقدار واحد وينزل للسطر ;cout << counter << endl ويطبع الرقم ١ ثم يدخل التكرار مره اخرى ويتأكد ان الرقم ٢ أقل من العشره ويزيد ال counter بمقدار واحد ثم يطبع ٢ ويواصل الى ان يصل العددالى الرقم عشرة يتاكد من ان ال counter الذي وصلت قيمته عشرة أقل من او يساوي ١٠ , وطبعا هنا يساوي العشرة يزيد ال counter بمقدار واحد ويطبع العشرة ويخرج البرنامج من الforلأن ال counter اصبحت قيمته تساوي ١١.وكما تلاحظون بكل بساطه ينتهي البرنامج هنا, اضنكم لاحظتم سهولة واهمية استخدام الfor.
ننتقل الى مثال عملي آخر :

المطلوب كتابة برنامج يقوم بطباعة المتوسط ل١٠٠ طالب وايجاد اقل درجه بين الطلاب واعلى درجه وطباعتهم:

```
#include <iostream>

using std::cout;
using std::endl;

int main()
{float total=0,max=0,min=100,avg;
   for ( int counter = 1; counter <= 100; counter++ )
     { cout << "enter the grade" << endl;
      cin>>grade;
 total=total+grade;
if(grade>max)
{  max=grade;}

If(grade<min)
{min=grade;}
```

```
 }
Avg=total/100;
Cout>>"the average is ">>avg>>end`;
Cout>>"the maximum grade is">>max>>end1;
Cout>>"the minimum grade is">>min>>end1;



  return 0;

}
```

نأخذ مثال آخر برنامج يقوم بجمع الاعداد الزوجية من ٠ الى ال ١٠٠ :-

```
#include <iostream>

using std::cout;
using std::endl;

int main()
{
   int sum = 0;


   for ( int number = 2; number <= 100; number += 2 )
      sum += number;

   cout << "Sum is " << sum << endl;
   return 0;
}
```

هنا نصل الى نهاية درس ال for الذي اتمنى ان يكون درس مفيد لكم .....اترككم برعاية الله وحفظه حتى الدرس القادم...

الى ذلك الحين دمتم بخير وعافيه والسلام عليكم ورحمة الله وبركاته.

°^~*¤®§(*§سبحانك اللهم وبحمدك أشهد ان لا اله الاانت استغفرك وأتوب أليك §*)§®¤*~^°

`~'*¤!||!¤*'~`((كاتبة الدرس .....بنت النور.....عفوا يمنع اجراء اي تعديل للملف))`~'*¤!||!¤*'~`

`~'*¤!||!¤*'~`((نسألكم دعوة صادقة لأختكم في ظهر الغيب))`~'*¤!||!¤*'~`

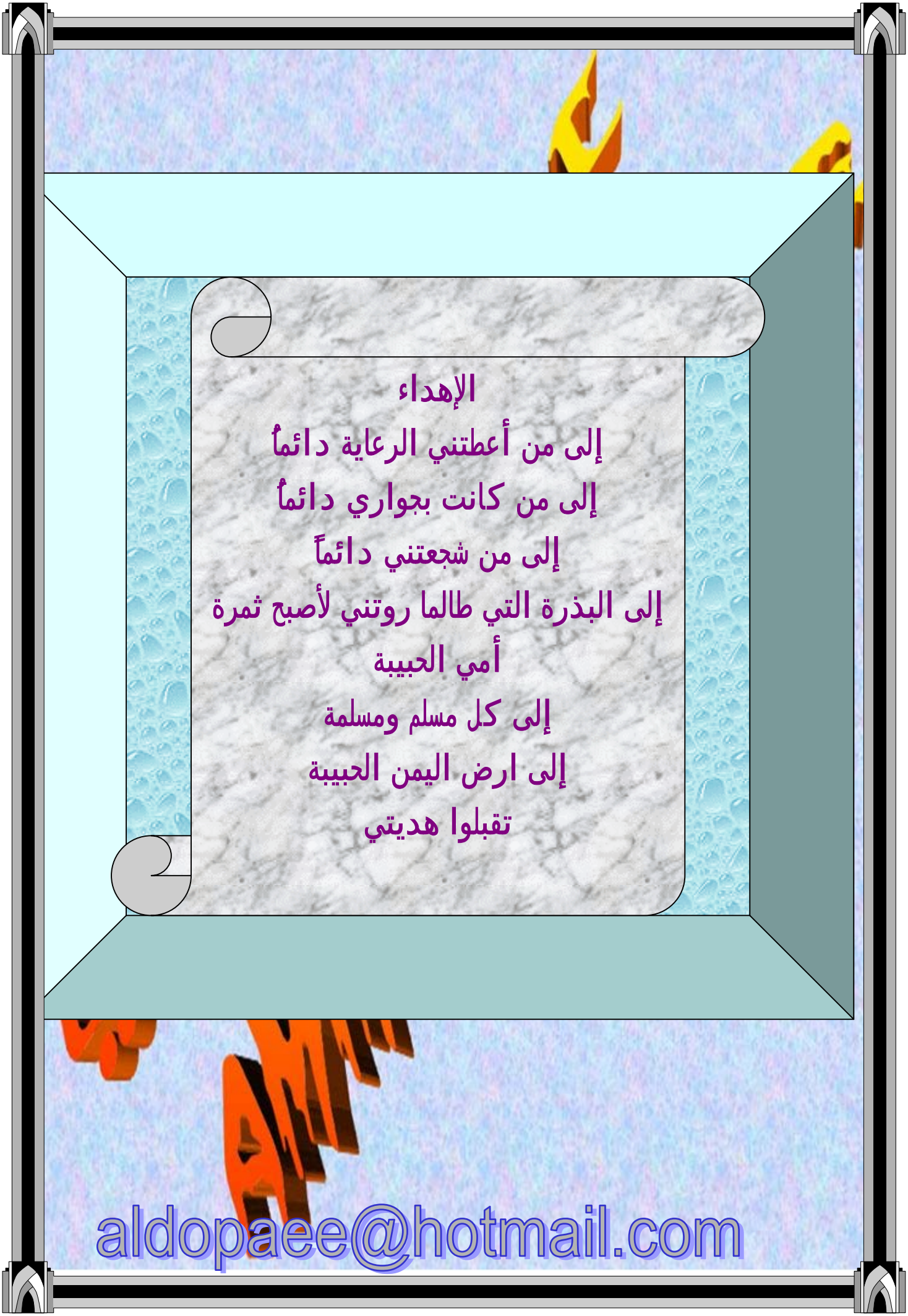

اصول البرمجة
بلغة ++C
اعداد / عمار محمد عيسى الدبعي
الإصدار الأول ٢٠٠٥
aldopaee@hotmail.com

بسم الله الرحمن الرحيم

الحمد لله رب العالمين الواحد الأحد الفرد الصمد الذي لم يلد ولم يولد والذي علم بالقلم علم الإنسان ما لم يعلم ونصلي ونسلم على المبعوث رحمة للعالمين محمد ابن عبد الله الصادق الآمين وعلى صحابته الأخيار ومن بعدة آلي يوم الذين أما بعد

تعد لغة c++ من اكتر اللغات انتشاراً واستخداماً كونها مناسبة ومفضلة في كثير من التطبيقات المهمة إذ نستطيع أن نقول أنها وسط تطوير البرمجيات و إنتاجها وهي ما تزال تحتل موقع الصدارة مقارنة مع غيرها من لغات البرمجة إذ تعتبر هي لغة إلام ونظرا لهذه الأهمية وعدم وجود مراجع عربية كافية بالغرض فقد قمت بتأليف هذا الكتاب الذي قد حاولت تلخيص الفكرة لإخواني الطلاب والمحبين لهذه اللغة من خلال دراسي الجامعية ومن محبتي لهذه اللغة راجياً من الله ان يستفيدوا إخواني المسلمين في مشرق الأرض ومغربة وإعطائي دعوة صالحة في ظهر الغيب من إخواني الذين قرئوا هذا الكتاب راجياً من الله إن يكون هذا الكتاب يفي بالغرض وان ينال إعجابكم ونأسف عن أي خطأ او تقصير فالكمال لله عز وجل وصلى الله على محمد وعلى آلة وصحبة أجمعين وأخر دعوانا إن الحمد لله رب العالمين ولا عدوان إلى على الظالمين والمشركين من اليوم إلى يوم الذين .

aldopaee@hotmail.com



الإهداء

إلى من أعطتني الرعاية دائماً

إلى من كانت بجواري دائماً

إلى من شجعتني دائماً

إلى البذرة التي طالما روتني لأصبح ثمرة

أمي الحبيبة

إلى كل مسلم ومسلمة

إلى ارض اليمن الحبيبة

تقبلوا هديتي

aldopaee@hotmail.com

لا يحق لأحد بيع هذا الكتاب فهو مجاني

ملاحظه

- يوجد بعض البرامج مكتوبة بلغة السي وذلك لتذكير بها من النسيان.
- الخطوة الوحيدة لتطوير هذا الكتاب أبداء إي ملاحظة أو استفسار نحو هذا الكتاب أو اكتشاف آي خطأ الرجاء أبلاغيي بذالك على البريد الالكتروني
- إلى كل طالب في قسم الحاسوب أي سؤال عجز عن حلة يرجى أرسالة للبريد الالكتروني وانأ سأوافيه الحل إنشاء الله وسأكون تحت الخدمة دائماً..... وشكراً

السيرة الذاتية

الاسم: عمار محمد عيسى الدبعي

الجنسية: يمني العمر: ٢١ سنة محل الإقامة: الجمهورية اليمن

طالب جامعي ----------كلية علوم وهندسة حاسوب/ المؤهلات

aldopaee@hotmail.com

لغة C++

المحتويات

aldopaee@hotmail.com

المؤثرات العلاقية و المنطقية

مؤثر العنوان (Address of Operator)

الفصل الثالث : اتخاذ القرارات

العبارة الشرطية البسيطة (if statement)

العبارة الشرطية الكاملة (if ……. else statement)

العبارة الشرطية المتدرجة (if-else- if Ladder)

الاختيار متعدد البدائل (switch statement)

مسائل على الباب

الفصل الرابع : الحلقات التكرارية

الحلقة for (for loop)

الحلقة while (while loop)

الحلقة التكرارية do-while

مسائل على الباب

الفصل الخامس: الدوال و الماكرو (Function & Macro)

أنواع الدوال

معاملات الدوال

معاملات الدالة الرئيسية (main function arguments)

المؤشرات

السجلات

الماكرو

مسائل على الباب

الفصل السادس : المصفوفات

مصفوفات البعد الواحد

مصفوفات ذات بعدين

مسائل على الباب

الفصل السابع : الفصائل والكائنات (Classes & Objects)

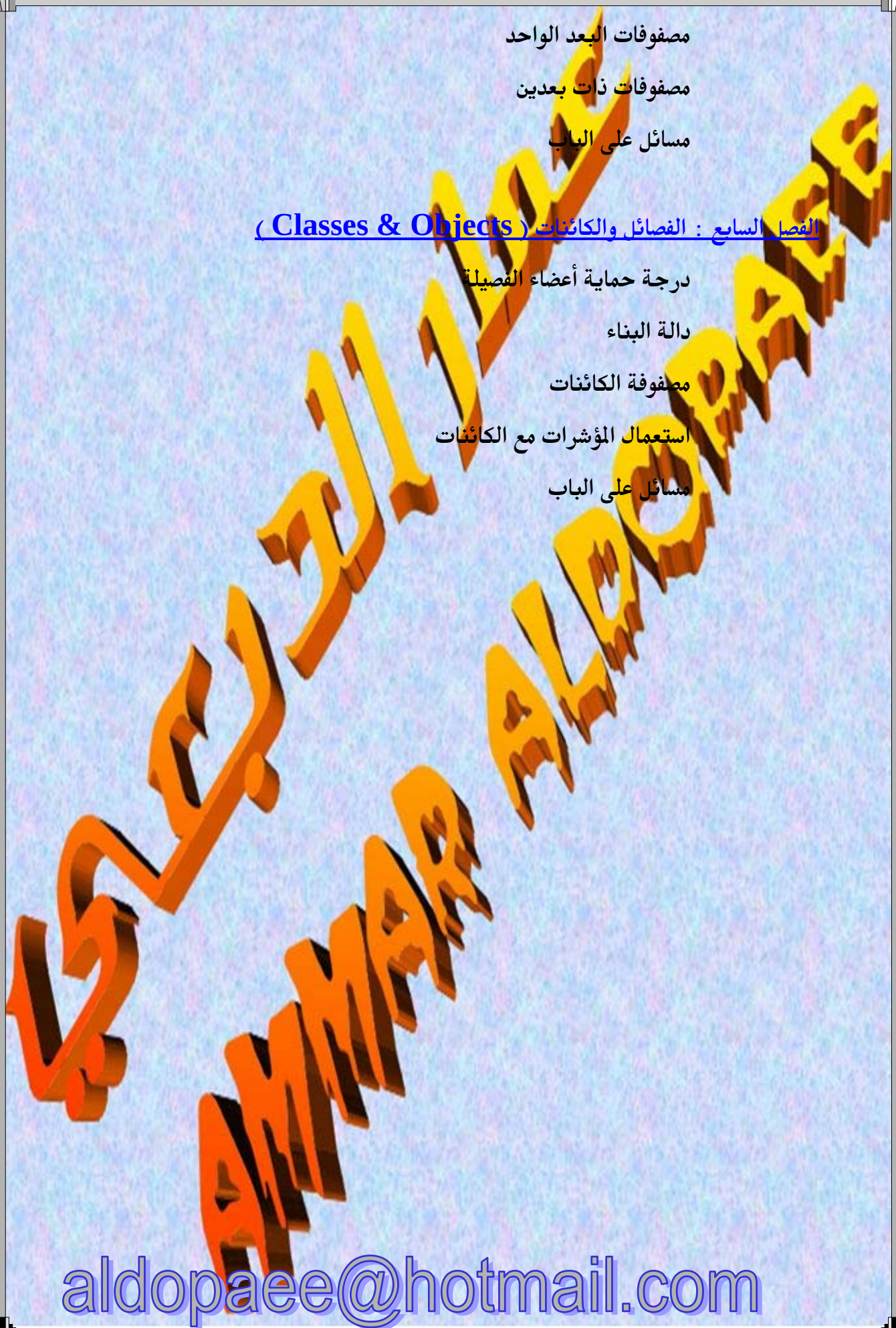
درجة حماية أعضاء الفصيلة

دالة البناء

مصفوفة الكائنات

استعمال المؤشرات مع الكائنات

مسائل على الباب



aldopaee@hotmail.com

مميزات لغة C++ عن لغة C:

تدعم لغة C++ أسلوب برمجة الكائنات الموجهة وبالإضافة لذلك تمتاز لغة C++ بالعديد من المزايا والتي سنتناولها فيما يلي بشيء من التفصيل.

المزيد من الحرية في الإعلان عن البيانات :

في لغة C يشترط الإعلان عن المتغيرات في مستهل البرنامج، وعند الحاجة لمتغير جديد لابد من الرجوع لأول البرنامج و الإعلان عنه.

ومع لغة C++ ينتفي هذا الشرط إذ يتمكن المبرمج من تعريف المتغيرات وقت الحاجة إليها وفي أي مكان.

حيث يعطى اسم الفصيلة بعد الكلمة المحجوزة (class)ويتوالى بعد ذلك تعريف البيانات والدوال.

تحديد درجة حماية البيانات :

تتيح لغة C++ تحديد درجات لحماية البيانات وذلك على مستوى الفصيلة، وتتحدد درجة الحماية باستخدام الكلمات (public , private , protected) ويوضح الجدول التالي درجات الحماية المختلفة

محدد الحماية	متاح لنفس الفصيلة	متاح للفصائل المشتقة	متاح للكائنات من فصائل أخرى
public	نعم	نعم	نعم
protected	نعم	نعم	لا
private	نعم	لا	لا

و بالتقدم في البرمجة سنألف استخدام محددات الحماية، وسنتعرض لها بشيء من التفصيل عند الحديث عن الفصائل والكائنات.

دوال البناء والهدم (constructors and destructors):

كما ذكرنا سابقا فالفصيلة تتكون من بيانات و دوال تتعامل مع هذه البيانات، وتتيح لغة C ++ للمبرمج أن ينشئ دالتين خاصتين تسمى إحداهما دالة البناء (constructor) وهي دالة تنفذ تلقائيا عند الإعلان عن كائن من هذه الفصيلة. وتظهر فائدة هذه الدالة عندما نرغب في تخصيص قيم ابتدائية لبيانات الفصيلة.

أما الدالة الأخرى فهي دالة الهدم (destructor) وتنفذ تلقائيا عند انتهاء استخدام الفصيلة وتستخدم هذه الدالة لتحرير أجزاء من الذاكرة كنا نستخدمها أثناء استعمال الفصيلة ولم نعد بحاجة إليها، أو لتنفيذ سطور معينة عند الانتهاء من استخدام الفصيلة.
ودالة البناء تحمل نفس اسم الفصيلة، فمثلا لو كان اسم الفصيلة (Ball) كانت دالة البناء تحمل الاسم (Ball).
أما دالة الهدم فتأتي بنفس اسم الفصيلة مسبوقا بالعلامة(~) فللفصيلة السابقة دالة الهدم تحمل الاسم (Ball~).

التوريث (Inheritance) :

من أقوى خصائص برمجة الكائنات الموجهة خاصية التوريث. ونعني هنا توريث فصيلة إلى فصيلة أخرى.

وهنا ترث الفصيلة المشتقة (derived class) من الفصيلة الأساسية (parent class) كل بياناتها ودوالها ويمكن التعديل بعد ذلك في خصائص الفصيلة المشتقة لتناسب الاحتياجات الجديدة، بإضافة المزيد من البيانات والدوال.

وبذلك نجد أن برمجة الكائنات الموجهة تعفي المبرمج من إعادة بناء البرامج من الصفر بل يعتمد على ما سبق لإنجاز البرامج الجديدة، فتمكنه من استخدام الفصائل السابقة و عمل فصائل جديدة للاستفادة منها مستقبلا.

الدوال الصديقة(friend functions):

عندما تعلن فصيلة عن دالة صديقة أو عدة دوال صديقة فإنها تسمح لهذه الدوال باستعمال البيانات الأعضاء فيها ولا تسمح لغير هذه الدوال بذلك.

وكذلك الحال عندما تعلن فصيلة عن فصيلة صديقة، فإنها تسمح لجميع دوال الفصيلة الصديقة باستخدام بيانات الفصيلة الأساسية.

وسيأتي الحديث بالتفصيل عن الدوال الصديقة في فصل الفصائل والكائنات.

برمجة الكائنات الموجهة ولغة C++

تعتبر برمجة الكائنات الموجهة من أحدث أساليب البرمجة، وليست بالأسلوب الوحيد حيث سبقتها أسلوب عرف بالبرمجة المنهجية (Procedural programming) والتي تعتمد على الدوال كوحدات بناء للبرنامج .

إذ يتكون البرنامج من مجموعة من الدوال التي تؤدي كل منها وظيفة محددة، وتقوم الدالة الرئيسية باستدعاء تلك الدوال وتنظيم العمل بينها.

أما أسلوب برمجة الكائنات الموجهة فيعتمد الفصيلة (class) كوحدة بناء البرنامج، وتتكون الفصيلة من مجموعة من البيانات والدوال التي تعمل على هذه البيانات.

وسنتعرف في هذا الفصل على عناصر ومزايا برمجة الكائنات الموجهة في هذا الفصل.بالإضافة إلى بعض مزايا لغة C++والتي تعتمد أسلوب برمجة الكائنات الموجهة في بناء البرامج

.

عناصر برمجة الكائنات الموجهة :

الفصائل (classes):

الفصيلة كما ذكرنا ما هي إلا بناء يتكون من بعض البيانات بالإضافة إلى دوال تتعامل مع هذه البيانات،

والفصيلة هي تكوين يقترب كثيرا من الواقع ، إذ أننا نجد في الحياة العملية الكثير من الأشياء والتي يمكن اعتبارها فصائلا.

وكمثال على ذلك يمكننا اعتبار الكتاب فصيلة، وبيانات فصيلة الكتاب عديدة مثل: اسم المؤلف، موضوع الكتاب، اسم الكتاب.....

أما دوال فصيلة الكتاب فهي مثلا قراءته، تأليفه طباعته .....

وكما تصورنا الكتاب كفصيلة يمكننا أن نجد فصيلة لكل الأشياء الموجودة في الحياة الواقعية.

الكائنات (Objects):

الكائن هو صورة من الفصيلة يتعامل معها المبرمج، فكما نعرف متغيرات من النوع الصحيح مثلا يمكننا باستخدام أسلوب برمجة الكائنات الموجهة أن نعرف كائنات من فصائل موجودة لدينا لنتعامل معها.

ومع التقدم في البرمجة باستخدام برمجة الكائنات الموجهة ستصبح هذه المصطلحات أكثر وضوحا وأقرب للذهن.

الفصل الثاني: أساسيات البرمجة بلغة C++

توجد لكل لغة أساسياتها التي ينبغي الإلمام بها قبل كتابة البرامج بواسطتها، وهذا الفصل يوضح هذه الأساسيات مثل: هيكل البرنامج، المتغيرات، الإدخال والإخراج.

وبجانب هذا يلمس الباب العديد من مزايا اللغة من كتابة التعليقات، والعمليات الحسابية، وتحويل البيانات. وغيرها من المزايا.

البناء الأساسي للبرنامج:

لنلق نظرة متعمقة على البرنامج التالي

```
#include <iostream.h>
void main()
{
    cout << " Every age has its own language . . .";
}
```

وبغض النظر عن صغر حجمه فإنه يوضح البناء الأساسي للبرنامج في لغة C++ ويتضح ذلك عندما نتناوله بالتفصيل كما يلي.

الدوال:

الدوال تشكل البلوكات الأساسية لبناء البرنامج ، ويتكون البرنامج هنا من دالة واحدة وهي الدالة الرئيسية (main()) والدوال في بناء برمجة الكائنات الموجهة قد تكون أعضاء في فصائل محددة أو تكون مستقلة بذاتها ، والدالة الرئيسية دالة مستقلة بذاتها حيث لا تنتمي لأي فصيلة.

والدالة لها اسمها ويليه قوسين توضع بينهما معاملات الدالة، ونلاحظ أن الدالة الرئيسية في هذا المثال ليس لها معاملات.

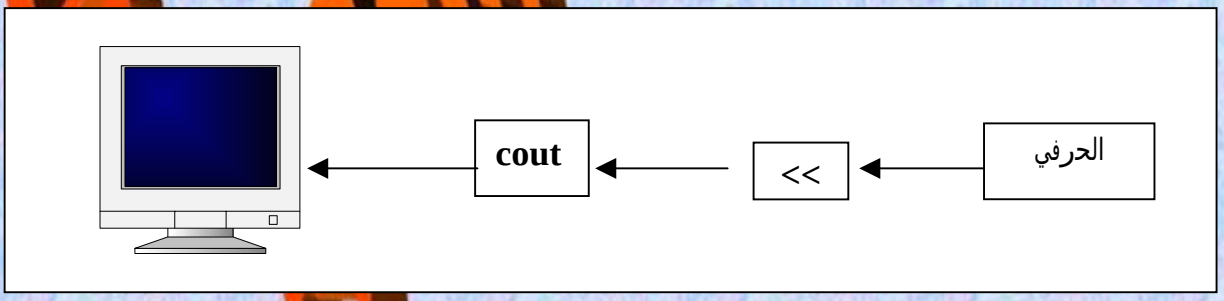
أما الكلمة المحجوزة (void) والتي تسبق اسم الدالة فتوضح أن الدالة ليس لها قيمة ترجع بها ، بخلاف بعض الدوال التي نخصص لها نوعا من البيانات بحيث ترجع قيمة من نوع هذا البيان.

وعبارات الدالة نفسها تحاط بقوسين خاصين" { " ، " } " يسميان بقوسي البلوكات.

و الدالة الرئيسية هي أول ما ينفذه الكمبيوتر عند تنفيذ البرنامج.

عبارات الإخراج

في البرنامج السابق نلاحظ أننا قد استخدمنا عبارة لطباعة الحرفيات، وتختلف هذه العبارة عن العبارات التي تعودنا عليها عند استخدام اللغات الأخرى فهنا لم نستخدم دالة خاصة لتطبع الخرج على الشاشة، بل قمنا بكتابة الحرف بين علامتي تنصيص واستخدمنا الكلمة المحجوزة (cout) والمعامل (>>) والعبارة التي استخدمناها للطباعة يفهمها الكمبيوتر بكما هو موضح بالشكل التالي



موجهات ما قبل المعالجة(preprocessor directives):

العبارة التي بدأ بها البرنامج (#include <iostream.h>) ليست في الواقع جزءا منه بل هي إعلان عن ملف يحتوي على تعريفات العديد من الدوال التي نحتاجها أثناء البرمجة.

وتبدأ العبارة بما يسمى بموجه قبل المعالجة وهو الرمز (#) والأمر الذي يليه موجه للمعالج مباشرة وهناك جزء من المعالج يتعامل مع مثل هذة الأوامر.

ويقوم بتنفيذ الأوامر الصادرة إلية لتتم عملية المعالجة اعتمادا على المعلومات التي وفرها للمعالج.

التعليقات (Comments):

عند كتابة برنامج بأية لغة يستحب كتابة التعليقات لتوضيح العبارات المكونة للبرنامج.

والمبرمج الذكي يحرص دوما على كتابة كل ما يمكنه من تعليقات على برنامجه ليسهل عليه تصحيحه أو استخدام بعض أجزاءه إن دعت الحاجة لذلك.

وتسمح لغة C++ بكتابة التعليقات بطريقتين تسهلان على المبرمج وضع ما يشاء من التعليقات على البرنامج .

والطريقة الأولى هي كتابة التعليق بعد العلامة " // " حيث يتجاهل المترجم السطر الذي يلي هذه العلامة.

ولكن لو تجاوز التعليق السطر لزم إضافة المزيد من الرموز " //" أمام كل سطر من التعليقات.

وللاستغناء عن الحاجة لكتابة العلامة " // " أمام كل سطر يمكن للمبرمج أن يستخدم الطريقة الثانية وهي كتابة التعليق بين العلامتين " /*" و " */" ويسمح في هذة الحالة كتابة التعليق على أكثر من سطر دون التسبب في الخطأ، طالما كان التعليق بين العلامتين المذكورتين.

والمثال التالي يوضح كيفية استخدام الطريقتين

```
// this is the first method        } 1

// this is the
// first method                    } 2

/* this is the second method */    } 3

/* this is
the second
method*/                           } 4
```

في المرة الأولى استخدمنا الطريقة الأولى ولم يتجاوز التعليق السطر فلم نستخدم سوى علامة تعليق واحدة. أما في المرة الثانية تجاوز التعليق السطر فلزم علينا استخدام علامة تعليق ثانية .

وفي المرة الرابعة استخدمنا الطريقة الثانية لكتابة التعليقات ومع ان التعليق تجاوز السطر فلم نستخدم علامة تعليق جديدة لكل سطر بل اكتفينا بوجود العلامة " /*" في بداية التعليق والعلامة " */" في نهايته.

المتغيرات في لغة C++

استخدام المتغيرات:

يقوم الكمبيوتر بتخزين البيانات التي يحتاجها في الذاكرة والمتغيرات ما هي إلا عناوين خانات في الذاكرة التي نحفظ فيها البيانات. ولتسهيل الوصول للبيانات المختزنة يتم في لغات البرمجة عالية المستوى استبدال العناوين الرقمية بأسماء المتغيرات

ويكفينا هنا – لو كنا مبتدئين في البرمجة– أن نتذكر دائما أن المتغير ما هو إلا اسم لأحد الأماكن التي تختزن فيها البيانات في الذاكرة.

وأسماء المتغيرات يصطلح عليها في لغة الC بأسماء البيانات (Identifiers)

وهناك قواعد محددة لاختيار أسماء البيانات وهي:

١- ألا يكون اسم البيان أحد الكلمات المحجوزة باللغة (Reserved words) أو الكلمات التي تحمل معنى خاصا مثل(main) ويمكن التعرف على الكلمات المحجوزة باللغة من دفتر التشغيل المصاحب للمترجم.

٢- يمكن أن يحتوي الاسم على أى حرف من الحروف الأبجدية (A-Z) سواء صغيرة كانت أم كبيرة، وأي رقم من الأرقام (٠-٩) كما يمكن أن تحتوي على علامة الشرطة السفلى (_) ولكن لا يجوز أن يبدأ الاسم برقم.

٣- لا قيود على طول الاسم ، وتتيح هذه الميزة استخدام أسماء معبرة عن مضمونها، ومن الأفضل دائما استخدام الاسم المعبر عن محتوى المتغير لتسهيل عملية فحص البرنامج في حالة الخطأ من جهة، ولتسهيل عملية الإضافة والتعديل للبرنامج.

٤- الحروف الكبيرة و الصغيرة ليست متكافئة في لغة C فمثلا اسم البيان(MY_NUMBER) يختلف عن الاسم (my_number) وكلاهما يختلف عن الاسم (My_Number).

الإعلان عن المتغيرات:

ليتمكن المستخدم من استخدام المتغيرات التي يريدها يتطلب البرنامج المكتوب بلغة C الإعلان المسبق عن أسمائها ونوعياتها في مستهل البرنامج .

وتصنف المتغيرات بحسب البيانات التي يمكن أن تختزن فيها فهناك المتغيرات الصحيحة (أي التي تصلح لإختزان الأعداد الصحيحة) و هناك المتغيرات الحقيقية (أي التي تختزن الأعداد الحقيقية) ، ومع تقدمنا في اللغة سنتعرف على نوعيات أخرى من المتغيرات.

وكما نرى في البرنامج أنه قد تم الإعلان عن متغيرين الأول (a) وهو من النوع الصحيح (integer) وقد استخدمنا الكلمة int للاعلان عنه.

وأما المتغير الثاني (b) فهو يختزن الأعداد الحقيقية (Real) وقد استخدمنا معه الكلمة float للإعلان عنه.

وكما ذكرنا سابقا، نلاحظ أن عبارة الإعلان تنتهي بفاصلة منقوطة كسائر عبارات البرنامج، كما أنه يلزم ترك مسافة خالية على الأقل بعد كل من الكلمات المحجوزة (int أو float) وبعد ذلك تقوم بقية البرنامج بطباعة محتوى المتغيرات a,b ولأننا لم نخزن في هذين المتغيرين أية بيانات فإن ما نحصل علية ليس إلا بعض المخلفات الموجودة في الذاكرة، وهي بلا معنى على الإطلاق

تخزين البيانات في المتغيرات (Assignment):

في البرنامج السابق لاحظنا أنه لابد من أن تختزن عددا ما في المتغير العددي الذي أعلنا عنه ويتم ذلك باستخدام عبارة التخصيص (assignment statement) ويوضح الشكل التالي(٢-١٣) برنامجا قمنا فية بالإعلان عن متغيرين و إختزان بيانين عدديين في كل منهما ، ثم نطبع محتويات هذين المتغيرين على الشاشة.

عبارة التخصيص (Assignment statment) :

إن العبارة

a=1000;

يمكن قرائتها على النحو التالي:

" خصص العدد ١٠٠٠ للمتغير a "

ومن الجائز أن نخصص متغيرا لمتغير آخر ، ومعنى ذلك أننا نضع نسخة من المغير الأول في المتغير الثاني.

أمالو فمنا بتخصيص تعبير حسابي يحتوي على متغيرات وقيم عددية لمتغير ما فأن البرنامج في هذه الحالة يجري عملية تقييم للتعبير الحسابي ويضع قيمته النهائية في المتغير المقصود.

ويوضح المثال التالي ثلاث عمليات تخصيص كالآتي:

١- تخصيص قيمة عددية للمتغير " a"

٢- قسمة محتويات المتغير " a" على 2 وتخصيص الناتج للمتغير " b"

٣- جمع محتويات كل من " a" ، " b" وتخصيصها للمتغير " c".

ومن الملاحظ في هذا البرنامج أنه قد تم إعلان المتغيرين " b" ، " c" في عبارة واحدة وقمنا بإستخدام علامة الفاصلة للفصل بينهما.

ونتيجة البرنامج النهائية هي طباعة محتويات المتغير " c"

التخصيص المتعدد:

يمكننا في لغة C++ أن نخصص قيمة ما لأكثر من متغير في نفس العبارة كالآتي:

```
a = b = c = 24;
```

تخصيص قيم ابتدائية المتغيرات:

يمكن أيضا شحن المتغير بقيمة إبتدائية أثناء الأعلان عنه كالآتي:

```
float a = 5.6 ;
```

ونقوم بشحن المتغيرات بقيمة ابتدائية عند الإعلان عنها لضمان تنظيف وعاء المتغير من مخلفات الذاكرة.

المتغيرات العددية الصحيحة (integer variables):

لنتعرف على كيفية تعريف المتغيرات العددية الصحيحة نلقي نظرة على البرنامج التالي

```
#include <iostream.h>
void main()
{
        int var1;      //define var1
        int var2;    //define var2

        var1=20;       //assign value to var1
        var2=var1+10;

        cout<< "var1+10 is "; //output text
        cout<<var2;                    // output value of var2

}
```

قمنا في هذا البرنامج بتعريف متغيرين من النوع الصحيح بالاسمين " **var1** " و " **var2** "

ولتعريف المتغير نستخدم الكلمة المحجوزة " **int** " وهي اختصار " **integer** " أو عدد صحيح، متبوعة باسم المتغير والذي يتبع القواعد المحددة السابق ذكرها لاختيار أسماء المتغيرات.

ونلاحظ في هذا البرنامج أن المتغيرات تم تعريفها في أول البرنامج وليس هذا شرطا في لغة C ++ إذ تتيح لنا إمكانية تعريف المتغيرات وقت الحاجة في أي مكان نشاء.

وبعد عبارة الإعلان عن المتغيرين ننتقل إلى عبارة أخرى وهي عبارة تخصيص القيم للمتغيرات ، حيث نخزن قيما فعلية في الأماكن التي حجزناها سالفا .

وفي هذا البرنامج نخزن القيمة " ٢٠ " في المتغير الأول، والعبارة المستخدمة لتخصيص قيمة المتغير الثاني ليست مباشرة، إذ يقوم المعالج بأداء عملية حسابية قبل تخصيص القيمة، حيث يجمع القيمة " ١٠ " على المتغير الأول.

ولإخراج قيمة المتغيرات على الشاشة نستخدم العبارتين الأخيرتين .

المتغيرات الرمزية (**char variables**):

المتغير الرمزي هو المتغير الذي يسمح بتخزين رمز فيه، والرمز في لغة الكمبيوتر هو كل ما يرد في جدول الكود آسكي والذي يحدد الرموز التي يمكن للكمبيوتر التعامل معها.

والرموز تحتوي الحروف الكبيرة والصغيرة والأعداد بالإضافة إلى العديد من رموز التحكم.

ولتعريف متغير رمزي نستخدم العبارة

char variable_name;

حيث (**variable_name**) هو اسم المتغير الرمزي، ويخضع أيضا للقواعد العامة لتسمية المتغيرات.

وعند تخصيص قيمة لمتغير رمزي نستخدم علامتي اقتباس مفردتين كما بالعبارة التالية

variable = 'A';

وهذه العبارة تخصص الرمز (**A**) للمتغير (**variable**)

تتابعات الهروب (**escape sequences**) :

من إمكانيات لغة C++ استخدام بعض رموز الحروف لأداء مهام خاصة ولنلق نظرة على البرنامج التالي

```
#include <iostream.h>
main()
{
char var1='A';
char var2='\t';
char var3='\n';

cout << var1;
cout << var2;
cout << var1;
cout << var3;

}
```

في هذا البرنامج نعلن عن ثلاثة متغيرات من النوع الرمزي ونخصص الرمز (A) للمتغير الأول.

أما المتغيرين الثاني و الثالث فنخصص لكل منهما رمز جديد مكون من علامة الشرطة المائلة العكسية (back slash) والتي تعني للمترجم أن الرمز الذي يليها ليس رمزا عاديا بل يحمل دلالة خاصة، والرمز الذي يلي علامة الشرطة المائلة العكسية يقوم بأداء عملية خاصة ، فمثلا إذا جاء بعد علامة الشرطة المائلة العكسية الرمز (n) كانت النتيجة الانتقال لسطر جديد.

ولو جاء الحرف (t) كانت النتيجة طباعة عدد من المسافات الفارغة و مماثلة للتي تنتج من استخدام المفتاح (tab).

وهناك العديد من تتابعات الهروب والتي نلخصها في الجدول التالي

تتابع الهروب	المعنى أو المفتاح المناظر
\a	إطلاق صفارة الجهاز مرة واحدة
\b	العودة للخلف مسافة رمز واحد (مثل استعمال المفتاح backspace)
\n	الانتقال لسطر جديد (مماثل لاستعمال المفتاح Enter)
\t	الانتقال للأمام مسافة عدة رموز (مماثل لاستخدام المفتاح tab)
\\	طباعة علامة الشرطة المائلة العكسية (\)
\”	طباعة علامة اقتباس مزدوجة
\’	طباعة علامة اقتباس مفردة

المتغيرات العددية العشرية(floating point variables)

تعرفنا في جزء سابق على المتغيرات العددية الصحيحة، ونعني بالصحيحة تلك التي لا تحتوي على كسور أي لا توجد بها علامة عشرية.والآن نتعرف على المتغيرات العددية العشرية أو كما تسميها بعض الكتب المتغيرات الحقيقية وهي المتغيرات التي تسمح لنا بالتعامل مع الأعداد التي تحوي الكسور أو العلامات العشرية، ومن هنا جاء اسمها.

ولتعريف متغيرات من النوع الحقيقي نلقي نظرة على البرنامج التالي:

```
#include <iostream.h>
void main()
{
        float var1;      //define var1
        float var2;        //define var2

        var1= 50.79;       //assign value to var1
        var2= var1 + 56.9;

        cout<< "var1+ 56.9 is "; //output text
        cout<<var2;                    // output value of var2
}
```

وتعريف المتغيرات الحقيقية لا يختلف عن المتغيرات الأخرى إذ يتم بنفس الطريقة وباستخدام الكلمة المحجوزة (float) وهي اختصار لكلمة (floating point) والتي تعني علامة عشرية، وهي ما يميز الأعداد الحقيقية.

ويتم التعامل مع الأعداد الحقيقية بنفس طريقة التعامل مع المتغيرات العددية الصحيحة.

وتجب ملاحظة أنه لو خصصنا ناتج عملية حسابية تحتوي على متغيرات حقيقية و أخرى صحيحة لابد من أن يكون الناتج مخصصا لمتغير من النوع الحقيقي، وإلا حصلنا على أخطاء عند التنفيذ.

عبارات الدخل والخرج في c:

حتى الآن قمنا بالطباعة على الشاشة باستخدام الدالة **printf** لطباعة الخرج وفقا لصيغة محددة (فورمات). و لكن قد يحتاج المبرمج لإدخال البيانات في وقت تنفيذ البرنامج ويستلزم ذلك استخدام دوال لإدخال البيانات، وهو ما سنتعرض له الآن بشيء من التفصيل.

أما الدالة المناظرة للدالة **printf**، والمخصصه لإدخال البيانات وفقا لصيغة محددة، فهي الدالة **scanf** ، ويعتبر الحرف " **f**" الذي تنتهي به كل من الدالتين هو الحرف الأول من كلمة " **format**"

يبدأ البرنامج بالإعلان عن ثلاثة متغيرات من النوع الحقيقي " **x,y,z** " ثم يتم استقبال قيمة المتغير " **x**" من لوحة الأزرار بموجب العبارة :

scanf ("%f",&x)

ثم يتم استقبال المتغير الثاني"**y**" بعبارة مماثلة ثم يتم جمع المتغيرين " **x,y** " وتخصيص الناتج للمتغير " **z**"

وفي النهاية نطبع قيمة المتغير " **z** " المحتوي على المجموع.

عند تشغيل البرنامج سوف ينتظر إدخال قيمة المتغير "**x**" فإذا أدخلنا العدد المطلوب وأتبعنا ذلك بالضغط على الزر **Enter** ، فإن البرنامج يتوقف مرة أخرى منتظرا إدخال قيمة المتغير " **y** " متبوعة بالضغط على الزر **Enter** وعندئذ يوافينا البرنامج بالنتيجة.

والآن فلننظر بتفحص لإحدى العبارات التي تحتوي على الدالة scanf فنلاحظ ما يلي:

١- ضرورة استخدام توصيف للفورمات بنفس الأسلوب المتبع مع الدالة printf وفي المثال السابق قد استخدمنا التوصيف " %f " الذي يناظر المتغير الحقيقي "x" أو " y".

٢- لم تستخدم الدالة المتغير " x" أو "y" صراحة بل استخدمت صورة محورة منه وهي (x&) ، وهذه الصورة الجديدة تسمى مؤشر العنوان (address operator) وهي عبارة عن عنوان المتغير لا المتغير نفسه. أما المؤثر الجديد & فيسمى مؤثر العنوان إلى (address-of operator)

إدخال أكثر من قيمة متغير واحد بنفس العبارة:

تماما كما مع الدالة printf يمكننا مع الدالة scanf استخدام عبارة واحدة ودالة واحدة لاستقبال قيم عدة متغيرات

نلاحظ أن الجزء الخاص بالفورمات (والواقع بين علامتي الاقتباس) يحتوي على توصيفين للفورمات " %f %f " بنفس عدد المتغيرات التي تأتي مفصولة عن بعضها البعض باستخدام الفاصلة " , " (أنظر العبارة المحتوية على الدالة scanf)

ومن الملاحظات الهامه أن ترتيب الفورمات في الدالة scanf يجب أن يكون بنفس ترتيب المتغيرات التي سيتم إدخالها. وهذه الملاحظة غير واضحة في المثال السابق نظرا لأن كلا المتغيرين المراد إدخالهما من نفس النوع.

الفصل بين المدخلات:

في المثال السابق كانت المتغيرات تدخل كل على حدة متبوعا بالضغط على الزر Enter ، ولكن ماذا لو أردنا إدخال المتغيرين في سطر واحد؟؟؟

المثال التالي يوضح الطريقة الجديدة لإدخال المتغيرين في سطر واحد ويتم الفصل بينهما بفاصلة ، ويتم ذلك بكتابة الفاصلة في البرنامج نفسه كفاصل بين توصيفات الفورمات.

رسالة لتنبية مستخدم البرنامج :

من عيوب الدالة cin او scanf أنها لا يمكن استخدامها لطباعة أي نص على الشاشة كما مع دوال الدخل في لغة مثل البيسك . وهذا معناه ضرورة الاستعانة بدالة الطباعة printf او cout إذا أردنا أن نطبع على الشاشة رسالة تنبه المستخدم إلى أن

البرنامج ينتظر إدخال بيان مثل:

Please Enter the number

في المثال التالي نرى صورة محسنة لإدخال قيمتي متغيرين مع طباعة الرسائل اللازمة لتنبية المستخدم.

ملاحظة هامة:

لا يوصى باستخدام الدالة scanf لاستقبال الحرفيات من لوحة المفاتيح، حيث يتطلب الأمر احتياطات كثيرة . ولاستقبال الحرفيات من لوحة المفاتيح توجد طرق أفضل سيأتي الحديث عنها.

طرق جديدة للتعامل مع الحرفيات:

لقد رأينا من قبل كيف يمكننا تخزين الحرفي بالاستعانة بالمؤشرات حيث يشير المؤشر إلى الرمز الأول من الحرفي المختزن في الذاكرة . هذا من ناحية بداية الحرفي . أما من ناحية نهاية الحرفي فإن البرنامج من تلقاء نفسه يضيف إلى مؤخرة الحرفي الرمز الصفري (NULL character) وهو الرمز رقم صفر في جدول الكود آسكي.

ويفيد هذا الرمز في تمييز مؤخرة الحرفي و بالتالي في تحديد طوله لتسهيل التعامل معه قراءة وكتابة ومعالجة بالطرق المختلفة.

وفي الواقع أن هذه الطريقة برغم ما تحتويه من تفصيلات فنية دقيقة لكنها أفضل من الطرق المستخدمة في اللغات الأخرى التي تتوفر بها المتغيرات الحرفية (**string variables**)، فمع هذه الطريقة في لغة C لا توجد أية قيود على طول الحرفي المستخدم.

وهنا سنتناول طريقة أخرى لتمثيل الحرفيات وهي مصفوفة الرموز (**character arrays**) ومن اسم هذه الطريقة يتضح أنه يتم حجز خانات الذاكرة اللازمة للحرفي مقدما.

الأعلان عن مصفوفة الرموز:

لننشئ مصفوفة من الرموز فإننا نبدأ بالإعلان عنها في بداية البرنامج . ويشمل الإعلان اسم المصفوفة وسعتها (**size**) أي الحد الأقصى لعدد الرموز بها .

فمثلا الجملة التالية يتم فيها الإعلان عن مصفوفة رموز بالاسم (**employee_name**):

```
char employee_name[20];
```

في هذا الإعلان يتم حجز عشرين خانة في الذاكرة تتسع كل منها لرمز واحد ، كما تخصص الخانة الأخيرة للرمز الصفري(**NULL**).

ولشحن هذه المصفوفة بأحد الحرفيات، فإن دالة خاصة تستخدم لهذا الغرض وهي الدالة (**strcpy (a,b)**) حيث " **a** " هو اسم مصفوفة الرموز، و "**b**" هو الحرفي المراد تخزينه في المصفوفة.

والمثال التالي يوضح الإعلان عن مصفوفة رموز بالاسم " **a** " تتسع لعشرين رمزا ثم ننسخ إلى عناصرها الحرفي " **Hello again** " وفي النهاية نطبع محتويات المصفوفة باستخدام دالة الطباعة **printf** مع استخدام الفورمات المناسبة للحرفيات **%s**.

ومن الملاحظ في هذا البرنامج ظهور توجيه جديد هو :

#include <string.h>

إن هذا التوجيه يصبح لازما عند استخدام الدالة strcpy حيث أن الملف " string.h" هو الملف الذي يحتوي على تعريف الدالة " strcpy " وبقية دوال الحرفيات، ويطلق على هذا الملف اسم ملف العناوين للحرفيات " string header file".

والآن سنتناول طريقة عمل البرنامج بشيء من التفصيل، ولنبدأ بدالة الطباعة printf .

فعندما تتعامل مع مصفوفة الرموز " a " فغنها تقرأ و تطبع عناصر المصفوفة واحدا بعد الآخر حتى تصادف الرمز الصفري فتتوقف.

أما عن طريقة تخزين الرموز في المصفوفة فهناك نقاط جديرة باهتمامنا .

إننا عندما نعلن عن المصفوفة " a[20]" فإن عناصر المصفوفة تأخذ الأرقام المسلسلة من " ٠ " إلى " ١٩" كالتالي:

a[0], a[1],.....,a[19]

ولا يشترط عندما نخصص أحد الحرفيات لهذه المصفوفة أن نشغل جميع العناصر (الخانات) ففي المثال السبق مثلا عدد رموز الحرفي كانت ١١ حرفا و استخدم العنصر الثاني عشر من المصفوفة لتخزين الرمز الصفري.

طرق مختلفة لإدخال الحرفيات:

ذكرنا من قبل أنه لا يوصى باستخدام الدالة scanf لإدخال الحرفيات من لوحة المفاتيح والآن سنستعرض البدائل المختلفة التي تتيحها اللغة لإدخال الحرفيات.

الدالة gets :

يعتبر اسم الدالة اختصارا للعبارة " get string " وهي تقوم بقراءة الحرفي المدخل من لوحة المفاتيح ، وتضيف إليه الرمز الصفري (NULL) ثم تقوم بتخصيصه للمتغير المطلوب و الذي يستخدم كدليل للدالة. وصيغة الدالة كالآتي:

gets(a);

حيث " a " مصفوفة الرموز.

وعندما يبدأ البرنامج سوف ينتظر منك إدخال الحرفي المطلوب وهو اسم الموظف " employee name " ثم يخصصه لمصفوفة الرموز المكونة من عشرين عنصرا. وفي النهاية يطبع البرنامج الاسم على الشاشة كتأكيد لتمام الاستلام و الحفظ.

ويمكننا هنا إدخال الاسم محتويا على مسافات خالية وذلك على العكس من الدالة scanf التي تعتبر المسافة الخالية مماثلة للضغط على المفتاح Enter.

ولكن هناك قيد على الحرفي المدخل إذ يجب مراعاة ألا يزيد طوله عن الحجم المحجوز للمصفوفة مع العلم بأن المترجم يستغل خانة من المصفوفة لتخزين الرمز الصفري. ففي هذا المثال لا يمكن إدخال أكثر من ١٩ رمز فقط.

الدالة fgets :

تستخدم هذه الدالة لقراءة حرفي من ملف أو جهاز للدخل (input device). ويتم تعريف الملف (أو جهاز الإدخال) ضمن صيغة الدالة نفسها كالتالي :

fgets(a, n, stdin);

حيث " a" مصفوفة رموز

و " n " الحد الأقصى للرموز المدخلة.

و " stdin" اسم جهاز الدخل القياسي (لوحة المفاتيح)

ويمكن بالطبع استبدال جهاز الدخل القياسي stdin بأجهزة أخرى حسب الموقف ولكننا في الوقت الحالي سوف نكتفي بلوحة المفاتيح كجهاز للدخل.

عند استخدام هذه الدالة في إدخال الحرفيات فإنها تضيف إلى مؤخرة الحرفي كلا من :

١- علامة السطر الجديد (n\ ).

٢- الرمز الصفري (NULL).

ولذلك فإنه مع هذه الدالة لابد وأن نخصص عنصرين في المصفوفة لهذين الرمزين .

طرق مختلفة لطباعة الحرفيات:

سنتناول الآن بعضا من دوال الخرج التي تصلح لطباعة الحرفيات بطريقة مبسطة.

الدالة **puts**:

اسم هذه الدالة إختصار للعبارة " **put string** " وهي الدالة المقابلة لدالة الدخل **gets**

وصيغة هذه الدالة كالآتي:

```
puts ( a);
```

حيث **a** ثابت حرفي ، أو مصفوفة رموز.

والمثال التالي يوضح استخدام هذه الدالة لطباعة رسالة لتنبيه المستخدم قبل استخدام الدالة **gets** لاستقبال البيان

وعند تنفيذ البرنامج نلاحظ أن الاسم المدخل قد جاء على سطر مستقل بعد رسالة التنبيه .

وذلك لأن الدالة **puts** عندما تطبع حرفيا على الشاشة تطبع في مؤخرته علامة السطر الجديد " **\n**"

الدالة **fputs**:

هذه الدالة هي المناظرة للدالة **fgets** فهي تستخدم لإرسال الخرج إلى ملف أو جهاز الخرج المذكور اسمه ضمن بارامترات الدالة.

وصيغة الدالة كالآتي:

```
fputs( a, stdout );
```

حيث **a** مصفوفة رموز أو ثابت حرفي.

و " **stdout**" اسم جهاز الخرج القياسي وهو جهاز الشاشة.

ومن الطبيعي استبدال جهاز الشاشة كما يتطلب التطبيق.

والدالة **fputs** تختلف عن **puts** في أنها لا تطبع علامة السطر الجديد في نهاية الحرفي.

عبارات الإدخال باستخدام **cin**

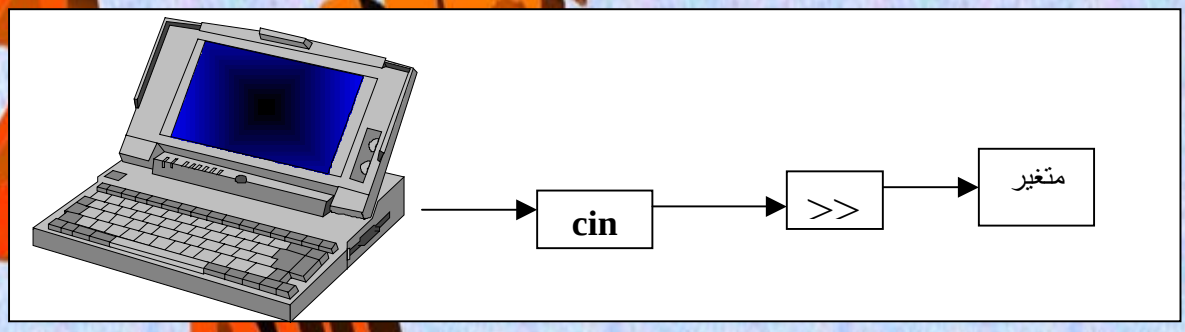
تعرفنا على العبارة المستخدمة في الإخراج ونتناول الآن العبارة التي تستخدم للإدخال.

والمثال التالي يوضح العبارة قيد الاستخدام

```
#include<iostream>
void main()
{
        int ftemp;
        cout << " Enter temperature in Fahrenheit: ";
        cin >> ftemp;
        int ctemp= (ftemp-32) * 5/9;
        cout<<"The temperature in Celsius is :  " <<ctemp<<"\n";
}
```

والإدخال في هذا البرنامج يتم بالعبارة التي تحوي الكلمة المحجوزة (**cin**) ويليها المؤثر (>>) ثم اسم المتغير الذي سنحتفظ فيه بالقيمة المدخلة.

والشكل التالي يوضح استخدام عبارة الإدخال



وتنتظر عبارة الإدخال المستخدم ليضغط على الرمز المراد إدخاله متبوعا بالمفتاح (Enter) ليضع القيمة في المتغير المحجوز سابقا.

المؤثرات (Operators):

إن لغة C++ – كأي لغة أخرى – تتعامل مع التعبيرات، وتتكون التعبيرات من الثوابت و المتغيرات المرتبطة ببعضها البعض بواسطة المؤثرات

وتنقسم المؤثرات إلى:

١- المؤثرات الحسابية (Arithmetic Operators)

٢- المؤثرات العلاقية. (Relational operators)

٣- المؤثرات المنطقية. (Logical operators)

المؤثرات الحسابية:

تتيح لغة C++ استخدام العديد من المؤثرات الحسابية، منها المؤثرات الأساسية والتي تقوم بالعمليات الحسابية الأساسية وهي الموضحة بالجدول التالي

المؤثر	المعني
+	الجمع
-	الطرح
*	الضرب
/	القسمة

وبالإضافة لهذه المؤثرات توجد مؤثرات خاصة وهي الموضحة بالجدول التالي

المؤثر	المعني
%	باقي القسمة
--	النقصان
++	الزيادة

وسنتناول بشيء من التفصيل استخدام هذه المؤثرات الخاصة.

مؤثر باقي القسمة

الصورة العامة لاستخدام هذا المؤثر هي :

```
x % y
```

ويكون الناتج هو باقي قسمة " x " على " y " ، والشكل التالي يوضح استخدام المؤثر والناتج

الاستخدام	10%3
الناتج	1

مؤثر النقصان و مؤثر الزيادة:

يعتبر هذان المؤثران من أهم ملامح اللغة ، ويمكنان المبرمج من كتابة عبرات البرنامج باختصار شديد.

والأمثلة التالية توضح استخدام هذين المؤثرين

مثال : بدلا من استخدام العبارة الآتية لزيادة قيمة المتغير (a) بمقدار (1)

```
a = a + 1;
```

يمكننا استخدام مؤثر الزيادة مباشرة كالآتي

```
a++;
```

والعبارة الأخيرة تؤدي أيضا إلى زيادة المتغير ولكن يلاحظ أن العبارة أصبحت أكثر اختصارا.

مثال بدلا من العبارة التالية والتي تستخدم لإنقاص المتغير (a) بمقدار ١

```
a = a –1;
```

يمكننا الآن استخدام العبارة المختصرة

```
a--;
```

ونلاحظ أيضا أن استخدام مؤثر النقصان يؤدي إلى الاختصار.

ونلاحظ ايضاً انه يوجد فرق بين a++ و ++a في الحالة الاولى تزيد قيمة aبواحد بعد طباعتها و الحالة الثانية عكس

وهذا المثال يوضح ذلك

```
#include <iostream.h>
int main()
{
int myAge = 39;
int yourAge = 39;
cout << "I am: " << myAge << " years
old.\n";
cout << "You are: " << yourAge << " years
old\n";
myAge++;
++yourAge;
cout << "One year passes...\n";
cout << "I am: " << myAge << " years
old.\n";
cout << "You are: " << yourAge << " years
old\n";
cout << "Another year passes\n";
cout << "I am: " << myAge++ << " years
old.\n";

cout << "You are: " << ++yourAge << " years
old\n";
cout << "Let's print it again.\n";
cout << "I am: " << myAge << " years
old.\n";
cout << "You are: " << yourAge << " years
old\n";
return 0;
}
```

```
Output: I am      39 years old
You are   39 years old
One year passes
I am      40 years old
You are   40 years old
Another year passes
I am      40 years old
You are   41 years old
Let's print it again
I am      41 years old
You are   41 years old
```

المؤثرات العلاقية والمنطقية:

تستخدم المؤثرات العلاقية لبناء التعبيرات العلاقية المستخدمة في المقارنات مثل

- x أكبر من 5 وتكتب للكمبيوتر بالصورة x>5.
- y أصغر من أو تساوي ٤٠ وتكتب y <= 40.
- x تساوي 15 وتكتب x = = 15.

والجدول الآتي يوضح المؤثرات العلاقية المختلفة المستخدمة في لغة C++.

المؤثر	المعني
>	أكبر من
>=	أكبر من أو يساوي
<	أصغر من
<=	أصغر من أو يساوي
= =	يساوي
!=	لا يساوي

ومن الملاحظات الهامة أن مؤثر التساوي (= =)يختلف عن مؤثر التخصيص (=) في لغة C ++، وذلك على العكس من اللغات الأخرى والتي تستخدم المؤثر (=) للتساوي.

والمؤثرات المنطقية هي التي تستخدم لتحديد العلاقات المنطقية في العبارات المستخدمة، فمثلا لو أردنا تنفيذ شرط معين عندما يتحقق تساوي أحد المتغيرات وليكن (x) بالمقدار (4) و عدم تساوي متغير آخر وليكن (y) بالقيمة (9) فإننا نلجأ الى استخدام المؤثرات المنطقية (مع استخدام المؤثرات العلاقية طبعا) وتكون العبارة كما يلي

```
if ((x = = 4) && (y != 9))
```

والمؤثر (&&) هو مؤثر منطقي بمعنى (و) أي عند تحقق الشرط الأول والشرط الثاني معا.

والجدول التالي يوضح المؤثرات المنطقية المستخدمة في لغة C++

المؤثر	المعني
&&	المؤثر (و) يستخدم عند لضمان تحقق الشروط كلها معا
||	المؤثر (أو) يستخدم لضمان تحقق أحد الشروط على الأقل
!	مؤثر النفي يستخدم للتأكد من عدم تحقق شرط (أو مجموعة شروط)

مؤثر العنوان (Address of Operator):

لنلق نظرة على البرنامج التالي

```
#include <iostream.h>
void main()
```

```
{
        int a =11;

        cout << &a<<'\n';
        cout << a;
}
```

عند تنفيذ هذا البرنامج نحصل على الناتج التالي

0x8f4ffff4

11

و القيمة الثانية تمثل قيمة المتغير (a) ولكن القيمة الأولى تمثل عنوان المكان المخزن فيه قيمة المتغير في الذاكرة .

الفصل الثالث : اتخاذ القرارات

تعرضنا حتى الآن لبرامج متتالية الأوامر، حيث ينفذ الكمبيوتر العبارات الموجودة في البرنامج بالترتيب الذي وردت به .

ولكن في الحياة العملية نحتاج لاتخاذ بعض القرارات تبعا لشروط معينة، ومن هنا ظهرت الحاجة لوجود طرق لجعل البرنامج قادرا على تغيير تسلسل تنفيذ التعليمات تبعا للشروط المطلوبة.

وسنتعرض هنا لطرق اتخاذ القرار في لغة C++ كيفية تغيير تسلسل التنفيذ تبعا للشروط الموضوعة.

العبارة الشرطية البسيطة (if statement):

تكوين العبارة الشرطية البسيطة كما هو موضح بالشكل التالي

```
if ( condition )
      statement;
```

العبارة الشرطية البسيطة

حيث (condition) هو الشرط و (statement) هو القرار المراد اتخاذه عند تحقق الشرط المعطى.

وعندما ترغب في تنفيذ أكثر من عبارة بتحقق الشرط نستبدل العبارة التي تمثل القرار المراد اتخاذه ببلوك به العبارات المراد تنفيذها.

ولتوضيح استخدام العبارة الشرطية البسيطة أنظر البرنامج التالي

```
#include <iostream.h>
main()
{       float sum;
        cout<< " Enter the sum ";
        cin >> sum;

        if(sum>50)
        cout<<" The student had passed";
```

}

وفي هذا البرنامج يطبع الكمبيوتر رسالة ليسأل المستخدم عن مجموع الطالب وبعد ذلك يقوم

بمقارنتها بالشرط اللازم للتأكد من النجاح (وهو تجاوز المجموع ٥٠)فإذا تحقق الشرط

يطبع الكمبيوتر رسالة للمستخدم يعلمه أن الطالب ناجح،

العبارة الشرطية الكاملة (if ..... else statement)

إن اتخاذ القرارات في الحياة العملية ليست بالسهولة التي ذكرت في البرنامج السابق، إذ نحتاج في معظم الأحيان لاتخاذ اجراء تبعا لشرط معين، واتخاذ إجراء آخر إذا لم يتحقق هذا الشرط.

لو نظرنا للبرنامج السابق لوجدنا سؤالا ملحا : ماذا لو كان مجموع الطالب أقل من ٥٠ ؟؟

الاجابة على هذا السؤال هي أن الطالب يكون راسبا. ولكن البرنامج لا يتضمن أمرا بإعطاء حالة الرسوب، لأننا استخدمنا عبارة الشرط البسيطة والتي تستجيب لشرط واحد.

وسنتعرض الآن لعبارة مركبة كما في البرنامج التالي

```
#include <iostream.h>
main()
{
      float sum;
      cout<< " Enter the sum ";
      cin >> sum;

      if(sum>50)
            cout<<" The student had passed";
      else
            cout<<" The student had failed";
```

}

وفي هذا البرنامج استخدمنا العبارة الشرطية الكاملة والتي تأتي على الصورة الموضحة بالشكل التالي

```
if ( condition)
        statement-1;
else
        statement-2;
```

العبارة الشرطية الكاملة

حيث أن (condition) هو الشرط

و (statement -1) هي عبارة النتيجة الأصلية.

و (statement -2) هي عبارة النتيجة البديلة.

ومنطق اتخاذ القرار هنا هو : " لو تحقق الشرط يقوم الكمبيوتر بتنفيذ عبارة النتيجة الأصلية أما لو لم يتحقق الشرط فيقوم الكمبيوتر بتنفيذ عبارة النتيجة البديلة"

وهكذا –باستخدام العبارة الشرطية الكاملة – تمكننا من اتخاذ القرار لحالتين متضادتين ، والآن ماذا لو كانت النتيجة الأصلية و النتيجة البديلة تتضمنان أكثر من أمر للكمبيوتر؟

في هذه الحالة نحتاج إلى احتواء عبارات النتيجة الأصلية بين قوسين من أقواس البلوكات، وهو الموضح بالشكل التالي

```
if ( condition)
{       statement 1;
        statement 2;
                statement n;
} else
{       statement 1;
```

```
        statement 2;
                .
                .
        statement m;
}
```

العبارة الشرطية الكاملة باستخدام بلوكات للنتائج

نلاحظ أن عبارة النتيجة تم استبدالها ببلوك النتيجة، والمثال التالي هو نفس البرنامج السابق بعد تعديل عبارات النتائج لتصبح بلوكات، وذلك ليتمكن البرنامج من إعطاء تقرير بالنجاح أو الرسوب متضمنا النسبة المئوية في حالة النجاح أو رسالة تفيد بأنه لا يمكن احتساب النسبة المئوية لطالب راسب.

```
#include <iostream.h>
main()
{
        float sum;
        cout<< " Enter the sum ";
        cin >> sum;

        if(sum>50)
                {
                        cout<<" The student had passed";
                        cout<< " His points are "<< sum/100;
                }
        else
                {
                        cout<<" The student had failed";
                        cout<<" No points are calculated for failed
                student !!";
                }
}
```

العبارة الشرطية المتدرجة (if-else- if Ladder):

لو افترضنا انه قد طلب منك – كمبرمج – عمل برنامج يمكنه احتساب التقديرات اعتمادا على مجموع الطالب، في هذه الحالة نستخدم عبارة شرطية أيضا ولكن بها عدد من الشروط وعدد مناظر من النتائج. أو ما يطلق علية العبارة الشرطية المتدرجة.

والشكل التالي يوضح التكوين العام للعبارة الشرطية المتدرجة

```
if  ( condition –1)
      statement –1;
else if ( condition-2)
      statement-2;
else if( condition-3)
      statement-3;
……….
else
      statement-n;
```

العبارة الشرطية المتدرجة

الاختيار متعدد البدائل (**switch statement**)

يعتبر الاختيار المتعدد البدائل بديلا للعبارة الشرطية المتدرجة التي تعرضنا لها سابقا، والواقع أن الاختيار المتعدد البدائل أعد خصيصا ليكون أسهل استخداما من العبارة الشرطية المتدرجة.ويتميز عنها بأنه أفضل توضيحا.

والشكل التالي يوضح الصورة العامة للاختيار متعدد البدائل

```
switch (variable)
{
      case value1;
            statement 1;
            break;
      case value2;
            statement 2;
            break;
      case value 3;
```

```
        statement 3;
        break;
    ..............
    default:
        statement;
}
```

الاختيار متعدد البدائل

وكما نرى فإن الاختيار المتعدد البدائل يبدأ بكلمة (switch) يليها متغير الاختيار والذي تحدد قيمته الاختيار الذي سيتم تنفيذه، ويلي ذلك قوس بلوك كبير يحتوي داخله بلوكات صغيرة كل منها يمثل اختيارا من البدائل المطروحة و كل بلوك من بلوكات البدائل يبدأ بكلمة (case) متبوعة بقيمة لتغير الاختيار – والتي تمثل الشرط– وبعد ذلك تأتي عبارة النتيجة.

ويختتم بلوك البديل بكلمة (break) والغرض من هذه الكلمة هو منع الكمبيوتر من تنفيذ عبارة النتيجة التالية!!!

وقد تبدو هذه العبارة غريبة للوهلة الأولى ويتبادر للذهن سؤال ملح : ألم يتحقق الشرط الأول مثلا فماذا يدفع الكمبيوتر لتنفيذ بقية عبارات النتائج؟؟

والإجابة عن هذا السؤال هي أن عبارة الاختيار متعدد البدائل لا ترسل للكمبيوتر أمرا بالتوقف بعد تحقق أي شرط فيها، لذا لزم الاستعانة بكلمة (break)

وبعد نهاية بلوكات البدائل تأتي كلمة (default) متبوعة بعبارة أو عبارات ينفذها الكمبيوتر في حالة عدم تحقق أي من الشروط السابقة.

مسائل على الباب

*برنامج لمعرفة العدد المدخل هل هو زوجي ام فردي

```
#include<iostream.h>
Main(){int a;
Cin>>a;
If (a%2=0)
Cout<<"zojee";
Else
Cout<<"frdee";
}
```

*برنامج لاستخراج الاعداد الفردية من ١-١٠ ومجموعهم

```
#include<iostream.h>
Main(){int a=1,b=0;
Q:If (a%2!=0){cout<<a;b+=a;}
a++;
if (a<=10)goto q;
cout<<"\n"<<b;
}
```

*برنامج لمعرفة العدد هل هو اولي ام لا من بين مجموعة من الاعداد ؟

*برنامج يجمع اخر عددين مدخلين من لوحة المفاتيح ؟

*برنامج يجمع الاعداد السالبة والاعداد الموجبة من بين عشرة اعداد ؟

*برنامج يطبع تقدير الطالب للدرجة المدخلة ؟

+++++>هذة التمارين على القارى.

الفصل الرابع: الحلقات التكرارية

كثيرا ما نحتاج في البرامج إلى تكرار أمر موجه للكمبيوتر عددا من المرات، وتوفر لغة C++ عدة وسائل تمكن المبرمج من أداء هذا التكرار.

وعادة ما تسمى هذه الوسائل " الحلقات التكرارية "، ويوجد العديد من الحلقات التكرارية في لغة C سنتناول منها هنا

١- الحلقة for (for loop).

٢- الحلقة while (while loop).

٣- الحلقة do.... while (do-while loop).

وفيما يلي سنتناول كل حلقة بالدراسة من حيث الشكل العام و أسلوب الاستخدام وأمثلة توضيحية.

الحلقة for (for loop):

تستخدم الحلقة for لتكرار أمر معين (أو مجموعة من الأوامر) عددا من المرات وتحتاج الحلقة إلي ثلاث عناصر أساسية كما هو موضح بالشكل التالي

```
for ( counter statement ; condition ; step)
```

شكل ٦-١ الصورة العامة للحلقة for

و هذه العناصر هي:

١- العداد (counter) : وظيفة العداد هي تسجيل عدد مرات التكرار.

٢- الشرط (condition): والشرط الذي يحدد نهاية التكرار إذ يظل التكرار قائما حتى ينتفي الشرط.

٣- الخطوة(step) : وهي القيمة التي تحدد عدد مرات التكرار.

والشكل التالي (شكل ٦-٢) يوضح برنامجا قمنا فيه باستخدام الحلقة for :

```
#include <iostream.h>
main()
```

```
{
    int counter ;
    for ( counter=1;counter<=20;counter++)
        cout<<counter;
}
```

برنامج يوضح استخدام الحلقة for

ومن البرنامج السابق نجد أن الحلقة for بدأت بكلمة (for) متبوعة بقوسين بينهما ثلاثة عبارات تفصل بينها علامة الفاصلة المنقوطة.

العبارة الأولى تخزن القيمة الابتدائية في العداد.

والعبارة الثانية هي الشرط وهنا الشرط أن قيمة العداد أقل من أو تساوي ٢٠.

أما العبارة الثالثة فهي تحدد الخطوة، وفي هذا البرنامج يزاد العداد بمقدار ١ كل مرة تنفذ فيها الحلقة.

والبرنامج السابق ينتج عنه طباعة الأرقام من ١ إلى ٢٠.

ملاحظات:

١- العبارات الثلاثة المكونة لحلقة for يجب أن تفصل عن بعضها بالفاصلة المنقوطة، وهذا الخطأ من الأخطاء الشهيرة جدا في عالم البرمجة لذا وجب توخي الحذر.

٢- في حالة تكرار أكثر من أمر يتم استبدال العبارة التي تلي بداية الحلقة for (في المثال السابق هي العبارة (cout << counter;) ببلوك يحتوي العبارات المراد تنفيذها.

الحلقة while (while loop):

في هذه الحلقة التكرارية نحتاج إلى الشرط فقط وطالما كان هذا الشرط متحققا استمرت الحلقة في التكرار..

والصورة العامة للحلقة **while** موضحة بالشكل التالي

```
while ( conditon )
{
        statement 1;
        statement 2;
                .
                .
        statement n;
}
```

الصورة العامة للحلقة while

حيث (**condition**) هو الشرط اللازم لأداء التكرار، والعبارات بداخل أقواس البلوكات هي العبارات المراد تكرارها.

والمثال الموضح بالشكل التالي يوضح استخدام الحلقة **while** لطباعة الأعداد من ١ إلى ٢٠

```
#include <iostream.h>
main()
{
        int counter=1;
while ( counter <=20 )
{
                cout<< counter;
                counter++;
        }
}
```

مثال لاستخدام الحلقة while

من المثال السابق يمكننا استخلاص النتاتج التالية عن الحلقة **while**:

١- تخصيص القيمة الابتدائية للعداد تتم خارج الحلقة **while**.

٢- زيادة العداد تتم داخل الحلقة **while**.

الحلقة التكرارية **do-while**:

تختلف هذه الحلقة عن الحلقتين السابقتين في مكان كتابة الشرط ، حيث يكتب الشرط هنا بعد العبارات المطلوب تكرارها.

والشكل التالي يوضح الصورة العامة للحلقة **do-while**

```
do
{
        statement 1;
        statement 2;
                .
                .
        statement n;
}
while ( conditon )
```

الصورة العامة للحلقة do-while

وأهم ملاحظة على الحلقة التكرارية **do-while** أنها تنفذ العبارات المطلوب تكرارها مرة واحدة على الأقل حتى ولو كان الشرط غير متحقق !!!

وتفسير ذلك أن التحقق من الشرط يتم بعد التنفيذ وليس قبله كما في الحلقتين السابقتين.

مسائل على الباب

*برنامج يطبع اكبر قيمة واصغر قيمة

```
#include<conio.h>
#include<iostream.h>

main()
{int i,a,s,d,f;
```

```
a=s=0;
clrscr();
cin>>a;
d=f=a;
for ( i=2; i<=6; i++){
if(d<a){
d=a;}
if(f>a){
f=a;}
cin>>a;
}
cout<<d<<"  "<<f;
getch();
}
```

*برنامج لايجاد مضروب العدد

```
#include<iostream.h>
#include<conio.h>
main(){
clrscr();
int a,s;long d;d=1;
cin>>a;
for(s=1;s<=a;s++){
d*=a;cout<<d<<"\n";
}
getch();
}
```

*برنامج لإيجاد عوامل العدد

```
#include<iostream.h>
#include<conio.h>
#include<math.h>
main(){
clrscr();
int a,s,d;
```

```
cin>>a;
for(s=1;s<=a;s++){
d=floor(a/s)*s;
if(a!=d){goto h;}
cout<<s<<"\n";
h:
}
getch();

}
```

*برنامج لإيجاد القاسم المشترك ؟

* // // المضاعف // ؟

* // يطبع الارقام التي تقبل القسمة على ٣ و٥ ؟

* // يطبع ويجمع السلسلة الثالية ١,١,٢,٣,٥,٨,١٣,٢١,٣٣ ؟

* // يعرف هل العدد المدخل متساوي المراتب ١١١ يطبع نعم /١٠١ يطبع لا؟

الفصل الخامس: الدوال و الماكرو (Function & Macro)

معنى الدالة:

الدالة هي مجموعة من الأوامر المحددة التي تعطى للكمبيوتر وغالبا ما تكون هذه الأوامر مرتبطة بأداء وظيفة محددة.

والدوال تمنح اللغة بعض المزايا مثل:

١- توفر في حجم البرنامج، حيث نستعيض عن تكرار عدد السطور التي تؤدي مهمة الدالة بإعادة استدعاء الدالة فقط.

٢- توفر مكتبة دائمة للمبرمج، حيث يمكن الاحتفاظ بالدوال وإعادة استخدامها حين الحاجة دون كتابتها من البداية.

٣- يؤدي استخدام الدوال الى زيادة وضوح البرنامج وتسهيل عملية تصحيحه، حيث يبدو البرنامج مع استخدام الدوال مقسما إلى أجزاء محددة واضحة او ما يسمى بالبلوكات.

وسنتناول الآن طريقة استخدام الدوال في البرامج، ليتم استخدام الدالة يجب أولا الإعلان عنها، وبعد عملية الإعلان عن الدالة يمكننا استخدامها بواسطة ما يسمى باستدعاء الدالة، ولابد من كتابة التعليمات التي تؤديها الدالة فيما يعرف باسم تعريف الدالة.

والمثال التالي (مثال ١)يوضح استخدام دالة عرفها المبرمج

```
#include <iostream.h>

void DrawLine();                    (1)
void main()
{
        cout << " This is the output of the function : " << '\n';
        DrawLine();                 (2)
}

void DrawLine()
{
        for (I=1;I<=40;I++)         (3)
                cout<<'*';
}
```

مثال (١)

وفي هذا المثال استخدمنا الدالة المسماة (**DrawLine**)والتي صممناها لرسم سطر من العلامة (*)

وفي السطر المشار إلية برقم (1) في البرنامج السابق قمنا بالإعلان عن الدالة أو (**function declaration**) وهو مجرد ذكر اسم الدالة وأنواع المتغيرات التي تأخذها ونوع القيمة التي تعيدها.

وفي هذا المثال لا تأخذ الدالة أية متغيرات وهو الموضح بالقوسين الفارغين بعد اسم الدالة مباشرة، ولا تعيد الدالة قيما أيضا وهو الموضح بالكلمة (**void**) والتي تسبق اسم الدالة.

أما السطر المشار إلية برقم (2) ففيه قمنا باستدعاء الدالة أو (function calling) والمراد منه توجيه الأمر للكمبيوتر بتنفيذ مضمون الدالة.

ومجموعة السطور المشار إليها بالرقم (3) هي تعريف الدالة أو (function definition)، وتعريف الدالة يتم بين قوسي بلوكات " { " و " } " ويتضمن التعليمات المطلوب من الدالة تنفيذها، وهنا تقوم الدالة بتنفيذ طباعة العلامة " *" أربعين مرة متتالية على نفس السطر مما يشكل الخرج المطلوب من الدالة.

ومن أهم الملاحظات التي يجب وضعها دائما في الاعتبار :

١- يمكن أن يأتي تعريف الدالة قبل استدعائها، وفي هذه الحالة لا حاجة بنا للإعلان عن الدالة في سطر مستقل.

٢- لا يمكن بأية حال أن يتم استدعاء الدالة قبل الإعلان عنها أو تعريفها.

أنواع الدوال:

تصنف الدوال تبعا للقيمة التي تعيدها، وتبعا لذلك نجد الأنواع التالية:

١- دوال أعداد صحيحة (int functions) وهي التي تعيد بيانا من النوع العددي الصحيح(integer).

٢- دوال أعداد حقيقية (float functions) والقيمة المعادة في هذه الحالة تكون من النوع الحقيقي (float).

٣- دوال حرفيات (string functions) وتعيد بيانا من النوع الحرفي وهو سلسلة من الرموز .

٤- دوال الرموز (char functions) وتعيد بيانا من النوع الرمزي (char).

٥- دوال لا تعيد قيما (void function) ولا تعيد قيما من اي نوع.

والمثال التالي (@ مثال ٢) يوضح دالة أعداد حقيقية وكيفية استخدامها.

```
#include<iostream.h>
float sum(float x, float y)
{
        float result;
        result = x + y;
        return result;          (1)
};
void main()
{
        cout << sum( 4.5 , 8.9 );
}
```

مثال ٢

نلاحظ أننا قمنا بالإعلان عن الدالة (**sum**) وتعريفها قبل الدالة الرئيسية (**main**)

وتأخذ الدالة (**sum**) متغيرين من النوع الحقيقي وتقوم بجمعهما وتعيد الناتج في صورة عدد حقيقي.

وعملية إعادة الناتج من الدالة تتم في السطر المشار إلية بالرقم (**1**) وتتم باستخدام الكلمة المحجوزة (**return**) ويليها المتغير المراد إعادة قيمته.

معاملات الدوال:

بعض الدوال تحتاج عند استدعائها إلى متغيرات مثل الدالة (**sum**) في المثال ٢

والمعاملات هي القيم التي تحتاجها الدالة لأداء مهمتها عليها ، في هذه الحالة جمع المعاملين.

وعلى العكس من ذلك توجد دوال لا تأخذ معاملات مثل الدالة (**DrawLine**) التي استخدمناها في المثال ١

معاملات الدالة الرئيسية (**main function arguments**)

كل البرامج التي تعرضنا لها حتى الآن تستخدم الدالة الرئيسية (**main**) بدون معاملات أي تكون متبوعة بقوسين فارغين، وبعد معرفتنا بالدوال نتساءل ألا يمكن أن نستخدم الدالة الرئيسية بمعاملات؟

والجواب على هذا السؤال أنه يمكن بالفعل استخدام الدالة الرئيسية بمعاملات والمثال التالي

مثال٣ يوضح برنامجا فيه الدالة الرئيسية تم استدعائها بمعاملاتها

```
#include < iostream.h >
main (int argc, char*argv[])
{
        if(afgc!=3)
        {
                cout<<" Arguments number error ….";
                exit(1);
        }
        cout<<"the first argument is"<<argv[1]<<'\n';
        cout<<"the second argument is"<<srgv[2];
}
```

مثال ٣

نلاحظ أن الدالة الرئيسية تستخدم معاملين هما (**argc**) وهو من النوع العددي الصحيح، ويستخدم لتخزين عدد المعاملات التي سيكتبها المستخدم عند استدعاء الدالة، والاسم (**argc**) اختصار لعدد المعاملات (**argument counter**)

أما المعامل الثاني فهو (**argv**) وهو عبارة عن مصفوفة حرفيات تختزن المعاملات التي يكتبها المستخدم عند استدعاء البرنامج.

وتكتب المعاملات الخاصة بالدالة الرئيسية عند استدعاء البرنامج فمثلا لو كان البرنامج السابق في صورته القابلة للتنفيذ محفوظا باسم (**prog1.exe**)

وكتبنا السطر الآتي لتنفيذه:

```
C:> prog1 First Second
```

فإن المعاملات تختزن في مصفوفة المعاملات بالشكل الموضح بالجدول أدناه

عنصر المصفوفة	القيمة المختزنة
argv[0]	Prog1
argv[1]	First
argv[2]	Second

ويقوم البرنامج بالتأكد من عدد المعاملات المعطاة فإذا كان غير ثلاثة طبع البرنامج رسالة خطأ.

و لو كان العدد مساويا لثلاثة (كما هو الحال في السطر المعطى بعالية) فإن البرنامج يطبع قيمة المعامل الأول .

ثم ينتقل لسطر جديد ليطبع المعامل الثاني.

ويكون خرج البرنامج كالآتي

```
the first argument is First
the second argument is Second
```

المؤشرات:

فكرة المؤشرات قد تبدو للوهلة الأولى صعبة ولكن مع الفهم الجيد يصبح استعمال المؤشرات في غاية السهولة.

والفكرة الأساسية هي أن ذاكرة الكمبيوتر مقسمة إلى أماكن لتخزين البيانات المختلفة ولكل مكان من هذه الأماكن عنوانه الخاص، وهذا العنوان يفهمه الكمبيوتر بصورته العددية (أي أن هذه العناوين ما هي إلا أعداد).

والبرنامج عندما يعلن عن متغير من نوع معين فإن الكمبيوتر يحجز مكانا له في الذاكرة.

وبالتالي يكون لكل متغير من متغيرات البرنامج عنوانه الخاص.

والمؤشر هو متغير يحمل العنوان، ويمكننا تعريف مؤشرات لكل أنواع المتغيرات في لغة C ++ .

ولتعريف مؤشر ما يذكر نوعه أولا ثم اسم المتغير مسبوقا بالعلامة (*)

وذلك كما في العبارة

```
float *ptr;
```

وفي هذه العبارة قمنا بتعريف مؤشر لعدد حقيقي، واسم المؤشر هو ptr.

ويمكننا بنفس الطريقة تعريف مؤشرات لكل أنواع البيانات التي توجد في لغة C++.

```
#include "stdio.h"

int main ()
{
int *px;
int a;

01: px = &a; /* 'px' will point on 'a'
*/

02: *px = 10; /* Changes on 'px' will
effect on 'a' */

03: printf("px = %d \n\n", *px);
04: printf("a = %d \n\n", a); /* 'a' and
'px' will have the same value that is 10
*/


05: a = 20; /* Changes on 'a' will
effect on 'px' */

06: printf("px = %d \n\n", *px);
07: printf("a = %d \n\n", a); /* 'a' and
```

```
'px' will have the same value that is 20
*/

return 0;
}
```

لاحظ عزيزي أنه في السطر ١ جعلنا px يؤشر على a أي أن px سيحتوي على عنوان a .

في السطر ٢ جلعا غيرنا px و جعلناها تساوي ١٠ و لكن في الحقيقة نحن لم نغير px و لكننا قمنا بتغيير العنوان الذي يوجد بداخل px و المعنى الحرفي لهذا السطر هو:

" إذهب إلى العنوان الموجود داخل px و إجعل هذا العنوان يحتوي على القيمة ١٠ " و نحن نعلم أن هذا العنوان هو عنوان a (من السطر ١) إذا سوف يغيير هذا السطر المتغير a و سنلاحظ هذا التغيير في السطرين ٣ و ٤ في السطر ٣ قلنا للدالة printf إطبعي محتوى العنوان الموجود داخل px و في السطر ٤ قلنا لها إطبعي قيمة المتغير a و سنلاحظ أن هذين القيمتين هما ١٠ إذاً سيطبع على الشاشة القيمة ١٠.

أما في السطر ٥ أسندنا للمتغير a القيمة ٢٠.

و في السطر ٦ قلنا للدالة printf إطبعي القيمة الموجوده في العنوان الموجود داخل px و نحن نعرف أن العنوان الموجود داخل px هو عنوان a (من السطر ١)، ثم في السطر ٧ طبعنا قيمة a و سيبطع على الشاشة القيمة ٢٠ مرتين، مره من المؤشر px و مرة أخرى من المتغير a .

طبعاً هذا المثال سهل و لكن الذي يفهم هذا المثال سيفهم ٩٠٪ من موضوع المؤشرات، و هذا الموضوع ليس صعب كما يعتقده الكثيرون :).

الآن قد تعرفنا على كيفية جعل المؤشر يؤشر على متغير و على كيفية حجز عنوان في الذاكرة للمؤشر و كيفية إسناد القيم للمؤشر و لكن بقي علينا أن نتعرف على كيفية إسناد المؤشرات لبعضها البعض و على كيفية جعل المؤشر يؤشر على مؤشر آخر.

aldopaee@hotmail.com

ففرضاً لو كان لدينا التعريق التالي:

```
int *p1, *p2;
int a = 5;
```

و جعلنا p1 يؤشر على a بكتابة هذا السطر:

```
p1 = &a;
```

(أي جعلنا p1 يحوي عنوان a)

الآن نريد أن نجعل p2 يؤشر على p1 كيف سيكون ذلك ؟!؟

الحل ببساطة نجعل في p2 عنوان p1 الذي هو في الأصل عنوان a، إذاً يكون السطر الذي يجعل p2 يؤشر على P1 هو:

```
p2 = p1;
```

أي عنوان p2 يساوي عنوان p1 .

الآن لو طبعاً قيم:

```
*p1, *p2, a
```

المؤشرات و المصفوفات:

المؤشرات و المصفوفات في لغة سي و سي++ متقاربين جداً، لذلك تجدون في أغلب الكتب أنهما يكونان في فصل واحد، و في الحقيقة كمبايلر السي و السي++ يجعل إسم المصفوفة مؤشر إلى أول عنصر فيها، لذلك الجملتين التاليتين متماثله تماماً:

```
r = *(x+1 تساوي r = x[1];
```

و لأن اسم المصفوفة مؤشر إلى أول عنصر بداخالها فإنه من الممكن إسناد مؤشر إلى المصفوفة عن طريق هذه الجملة:

```
ptr = x;
```

أو:

```
ptr = &x[0];
```

هاتين الطريقتين لهما نفس الفاعليه و لكن الفرق هو انه في الأول عاملنا المصفوفه X كمؤشر إلى أول عنصر و لكن في الطريقة الثانية عاملنا المصفوفه X كمصفوفه أي أن X[0] هو متغير عادي مثل أي متغير.

و لنأخذ هذا المثال:

```
#include "stdio.h"

int main ()
{
int x[] = {0, 1, 2, 3, 4, 5, 6, 7, 8,
9};
int i;

printf("Arrays As Pointers:\n");
for( i = 0 ; i < 10 ; i++)
{
printf("*(x + %d) = %d\n", i, *(x + i));
}

printf("\n\nArrays As Arrays:\n");
for( i = 0 ; i < 10 ; i++)
{
printf("x[%d] = %d\n", i, x[i]);
}
```

```
printf("\n\n\n");

return 0;
}
```

في هذا المثال يتبين لنا كيف إستخدمنا المصفوفات كمؤشرات تاره و كمصفوفات تارة أخرى و أن لهما نفس النتائج.

مصفوفة المؤشرات:

المصفوفات من الممكن أن تكون مؤشر إلى نوع ما بحيث كل عنصر من المصفوفة يصبح مؤشر إلى نوع معين.

من أكثر الأمثله لهذا التركيب هو مصوفة مؤشرات إلى char أي كل عنصر عبارة عن string أي كالتالي:

```
char *names[4];
```

أي أن المصوفه name تحتوي على أربع عناصر كل منها هو string .

لنأخذ مثال بارسم على ذلك:

لنفرض انه لدينا هذا التعريف:

```
char *name[4] = { "Talal", "Abdullah",
"Thamer", "Mohammad" };

name[0] ===> "Talal"

name[1] ===> "Abdullah"

name[2] ===> "Thamer"

name[3] ===> "Mohammad"
```

سيكون تمثيل العناصر كما هو مبين أعلاه.

السجلات (Structures):

السجل عبارة عن مجموعة مترابطة من البيانات كما في المصفوفات ولكن السجل يحتوي بيانات مختلفة الأنواع وليست من نوع واحد كما في المصفوفة.

والسجل يتكون من عدة حقول (fields) تحوي البيانات المختلفة ويستخدم السجل لتخزين بيانات مترابطة متكررة، كما في قاعدة البيانات حيث تتكون قاعدة البيانات من سجلات بكل سجل منها نفس الحقول، ولكن قيم تلك الحقول تختلف من سجل لآخر.

كيفية تعريف السجل في لغة سي:

أولاً لابد ان نعلم ان كلمة struct كلمة محجوزه في لغة سي و سي++ ، و نستطيع تعريف السجل كالتالي:

```
struct (إسم السجل
{
أعضاء السجل
};
```

– طبعاً هذه الطريقة هي أحدا الطرق التي تستطيع تعريف السجل بها.

فلو اردنا ان نعرف سجل إسمه data و يحتوي على إسم من نوع char* و العمر من نوع int

إذا سيكون التعريف كالتالي:

```
struct data
{
    char namr[30];
    int age;
```

};

حيث (stucture_name) هو اسم السجل وبداخل السجل تتوالى الحقول المختلفة(الأنواع field1, field2،.....) ولكل حقل نوعه الخاص.

وبتعريفنا للسجل يمكننا بعد ذلك تعريف متغيرات من نوع هذا السجل لاستخدامها في البرنامج حسب الحاجة

ويتم تعريف المتغيرات من السجل كما هو موضح بالشكل التالي الذي يوضح تعريف متغير (var1) من نوع السجل (structure1)

```
struct structure1
    {
        type   field1;
        type   field2;
        …
    }  var1;
```

ويمكننا تعريف أي عدد من المتغيرات من نوع هذا السجل كما يتطلب البرنامج.

والآن كيف نتعامل مع السجلات؟؟

إننا نحتاج مثلا لتخزين قيمة معينة في أحد الحقول، وفي هذه الحالة نستخدم المؤثر (.)

والمثال التالي يوضح عمل سجل باسم (Student) وتخصيص اسم (Mohammed) لحقل الاسم (name)

```
#include<iostream.h>
struct Student
    {
        char* name;
        int number;
    };
main()
{
    Student Sdt1;
    Std1.name=”Mohammed”;
    …
```

```
        Cout << Std1.name;
}
```

وعند تنفيذ البرنامج تقوم العبارة الأخيرة بطباعة الاسم " Mohammed" وهو الذي قمنا بتخزينه في الحقل (name) من المتغير (Std1).

مصفوفة السجلات :

لقد علمنا ان السجل نوع كأي نوع من انواع البيانات، لذلك من الممكن ان يكون السجل مصفوفة ايضاً و الطريقة سهله جداً كالتالي :

```
structure_name var[NUM] ;
```

فلو اخذنا السجل :

```
typedef struct
{
        char name[30];
        int age;
}data;
```

و اردنا ان نعرف مصفوفة من نوع data يسكون كالتالي :

```
data student[100] ;
```

طبعاً العدد ١٠٠ إختياري .

و نحن في السابق أخذنا نوع student من السجل data و سيكون سجل واحد و لكن هنا سيتضح اهمية السجلات فعندما عرفنا student كمصفوفة من نوع data أصبح كأنه لدينا

١٠٠ طالب و كل عنصر في المصفوفة عباره عن سجل بحد ذاته.

و للوصول إلى محتويات السجل نتبع الطريقة التاليه :

student[indix].name & student[indix].age ...

و غالباً تستخدم مصفوفة السجلات إذا كان العدد محدداً أما إذا كان العدد غير محدد نستخدم طريقة من طريق الـ Data Structure منها اللنك لست درسنا القادم.

السجلات و المؤشرات :

و نعيد و نكرر انه بعد تعريف السجل يصبح نوع كأي نوع آخر من انواع البيانات، إذا يمكن للسجل ان يكون مؤشر (Pointer) و العمليه كالتالي:

```
typedef struct
{
        char name[30];
        int age;
}data;
```

و سنعرف مؤشر للسجل كالتاالي :

```
data *s ;
```

فالنأخذ البرنامج التالي للتوضيح :

```
#include<stdio.h>
#include <conio.h>
```

```
typedef struct
{
	char name[30];
	int age;
}data;

int main()
{
	data *s, std;

	s = &std;	// Assign std to s

	strcpy(std.name,"Talal");
	std.age = 20;

	printf("std.name = %s, std.age = %d\n\n",std.name, std.age);
	printf("s->name = %s, s->age = %d\n\n",s->name, s->age);

return 0;
}
```

طبعاً نلاحظ الآن ظهور العلامة '->' بدل من النقطة عند إستخدام المتغير S ؟! لماذا ؟

الجواب : لأنه مؤشر لسجل و مؤشر السجل يستعمل في لغة السي و السي++ هذه العلامة بدلاً من العلامة '.' ، و هذا من الاختلافات التي تميز لغة السي و السي++ عن باقي اللغات مثل الجافا و الدلفي فهي لا تفرق إذا كان مؤشر أو لا .

إذا قاعدة في لغة سي و سي++ هي إنه عند إستخدام مؤشر لسجل نستخدم -> بدلاً من '.'

طبعاً هناك طريقة أخرى و هي هكذا :

s->name بدل (*s).name

طبعاً العلامة '->' أسهل :).

- **السجلات و الدوال :**

عند إستخدام السجلات مع الدوال إما أن يكون السجل مرسل للدالة أو إما ان يكون معاد من الدالة و إما ان يكون مستخدم في ضمن الدالة .

الحالة الأخيره معروفة و عملنا عليها في السابق داخل الدالة **main** و الـ **main** دالة اصلاً.

أما الحالتين الأولى و الثانيه فسنتطرق لها الآن.

- أولاً السجل معامل من معاملات الدالة :

أي أن نرسل السجل للدالة و الدالة تقوم بالعمليات على هذا السجل مثلاً: طباعة، معالجة، ... إلخ

و لنأخذ هذا المثال و نشرحة بعد قراءة المثال جيداً:

```
//-------------------------------------------------------------

#include<stdio.h>
#include <conio.h>
//-------------------------------------------------------------
typedef struct
{
        char name[30];
        int age;
```

```
}data;
//-------------------------------------------------------
void display(data r);
//-------------------------------------------------------
main()
{
	data std;

	strcpy(std.name,"Talal");
	std.age = 20;

	display(std);
}

//-------------------------------------------------------
void display(data r)
{
	printf("(r.name) = %s,\n(r.age) =
%d\n\n",r.name, r.age);
}

//-------------------------------------------------------
```

و في هذا المثال لقد كتبنا رأس الدالة كالتالي:

```
void display(data r) ;
```

أي أنه يوجد دالة إسمها display تستقبل السجل r من نوع data و تقوم بإرجاع شئ.

و عند إستدعاينا الدالة و بعد إعطائها القيم كالتالي:

```
display( std ) ;
```

ارسلنا لها السجل كاملاً لتسقبله و تطبعه في جسم الدالة display .

و لنأخذ مثالاً آخر لإعطاء قيم السجل في الدالة و طبعاتها في الـ main :

```
#include<stdio.h>
#include <conio.h>

typedef struct
{
        char name[30];
        int age;
}data;

void assign(data *r);

main()
{
        data std;

        assign(&std);
        printf("std.name = %s,\nstd.age =
%d\n\n",std.name, std.age);
}

void assign(data *r)
{
        strcpy(r->name,"Talal");
```

```
        r->age = 20;
}
```

و في هذا المثال كتبنا رأس الدالة (التعريف) هكذا :

```
void assign(data *r) ;
```

و جعلنا r كمؤشر لأن قيمة r ستتغير (نحن نريد ذلك) لإعطائها القيم.

و قمنا بإرسال السجل كالتالي :

```
assign( &std ) ;
```

لأن الدالة assign تستقبل مؤشر للسجل لذلك نرسل لها عنوان السجل و ليس السجل نفسه.

و داخل الدالة assign إستخدمنا r->name و r->age لأن r في الدالة مؤشر (و مع المؤشرات نستخدم -> بدلاً من '.') .

<u>– ثانياً إرجاع سجل من الدالة :</u>

أي أن الدالة تقوم بإرجاع السجل عند الانتهاء من عملها و نستطيع تغيير البرنامج السابق ليرجع السجل بدلاً من إرسال السجل كعنوان و إستقباله كمؤشر.

سيتغير البرنامج ليصبح هكذا :

```
include<stdio.h> #
#include <conio.h>

typedef struct
```

```
{
        char name[30];
        int age;
}data;

data assign(void);

main()
{
        data std;

        std = assign();
        printf("std.name = %s,\nstd.age =
%d\n\n",std.name, std.age);
}

data assign(void)
{
        data r;
        strcpy(r.name,"Talal");
        r.age = 20;
    return r;
}
```

و هنا عرفنا الدالة كالتالي :

```
  data assign(void) ;
```

أي أن الدالة assign لا تستقبل شئ و القيمة المرجعة من الدالة هي عباره عن سجل من نوع data .

و قمنا بإستدعا الدالة هكذا :

```
std = assign() ;
```

أي أن القيمة المرجعة من الدالة ستوضع قيمتها في السجل std .

وفي جسم الدالة assign عرفنا المتغير r من نوع سجل data و أعطينا لها قيم و قمنا بإرجاع هذا السجل من الدالة عن طريق الامر

```
return r ;
```

• إسناد السجلات :

نستطيع ان نسند سجلين لبعضهما البعض لكن شريطة أن يكونا من نفس النوع .

فلو أنشئنا السجل التالي :

```
typedef struct
{
char name[30];
int age;
}data;
```

و عرفنا منه متغيرين هكذا :

```
data a, b ;
```

و أعطينا المتغير a هذه القيم :

```
strcpy( a.name, "talal" ) ;
a.age = 20 ;
```

فبإمكاني ان اسند للمتغير b نفس محتويات المتغير a عن طريق هذه الجملة :

```
b = a ;
```

° إعطاء السجل أكثر من إسم أو إعطائه المتغيرات لحظة بناء السجل :

فلو كان لدينا السجل التالي :

```
typedef struct
{
char name[30];
int age;
}data, MyData ;
```

أستطيع أن اعرف المتغيرات سواء كان بـ data أو بـ MyData و كلها صحيحه.

فلو قلت :

```
MyData student ;
```

أو

data student ;

كانا سواء .

و هذا هو إعطاء السجل اكثر من إسم . أما إعطاء السجل أكثر من متغير لحظة بناء السجل و بدون تحديد إسم للسجل يكون كالتالي :

```
struct
{
الاعضاء
}إسم المتغير ؛
```

فلو اردنا ان نعمل على ١٠٠ طالب فقط و متأكيدن أن العدد لن يزيد عن ١٠٠ طالب فالأفضل بناء السجل هكذا :

```
struct
{
    char name[30] ;
    int age ;
} student;
```

و هكذا يصبح student متغير و نقول :

student. name & student. age

طبعاً إلى الآن تعلمنا كيف ننشئ السجل بثلاثة طرق بقي الطريقة الرابعة و الاخيره و هي كالتالي :

struct (إسم السجل

{

الاعضاء

}(المتغيرات) ؛

أي نستطيع أن ننشئ سجل الطالب الذي تكرر علينا كثيراً بالطريقة الرابعه هكذا :

```
struct data
{
    char name[30] ;
    int age ;
} student;
```

هنا student سيكون متغير و data هو إسم السجل و هنا نستطيع في كل مرة نحتاج فيها لإنشاء سجل أن نشئ سجل بالطريقة :

```
struct data VAR ;
```

و إستعمال student كمتغير جاهر غير محتاج للتعريف .

** نقطة أخيره :

في كل جزئ من أجزاء البرامج التي كتبتها و التعريفات و انشاء المتغيرات في الدرس إستخدمت غالباً التعريف التالي :

```
typedef struct
{
```

```
    char name[30];
    int age;
}data;
```

و أنشئت المتغيرات كالتالي :

```
data VAR ;
```

ممكن تغييره إلى

```
struct data
{
char name[30] ;
int age ;
};
```

و لكن تعريف المتغير سيكون :

```
struct data VAR ;
```

الماكرو:

الماكرو هو مجموعة أوامر مصممة لأداء غرض معين. والمثال التالي يوضح برنامجا استخدمنا فيه ماكرو لحساب مربع العدد

```
#include <iostream.h>
#define SQUARE(A) A*A
main()
{
        int x;
        cout<< " Please enter a number to calculate it's square ";
```

```
    cin >> x;
    cout << " The square of "<< x << "is :" << SQUARE(x);
}
```

والبرنامج هنا ينتظر من المستخدم إدخال قيمة عددية للحصول على مربعها، ويحسب البرنامج قيمة مربع العدد باستخدام الماكرو المعلن في السطر الثاني الملون بالأحمر.

ولحساب القيمة يقوم البرنامج باستدعاء الماكرو (في الجزء الملون من آخر عبارة من البرنامج)

والماكرو يشبه الدالة إلى حد ما وإن كان هناك اختلاف بينهما نتناوله الآن بالتفصيل.

يمر البرنامج بعدة مراحل قبل الحصول على النسخة القابلة للتنفيذ منه وهذه المراحل هي:

١- كتابة البرنامج وحفظة باستخدام أحد برامج التحرير (**Editors**) وتسمى هذه العملية بكتابة الكود (**coding**) ويحتفظ بالملف في هذه الحالة بالإمتداد " **.cpp**" ويسمى بالملف المصدر (**source file**).

٢- عملية الترجمة (**compilation**) وينتج عن هذه العملية البرنامج الهدف الذي يحمل عادة الأمتداد " **OBJ**" .

٣- عملية الربط بمكتبة اللغة (**Linking**) وينتج عن هذة العملية البرنامج التنفيذي الذي يحمل الأمتداد "**EXE**" والبرنامج التنفيذي هو البرنامج الذي يتم تنفيذه بمجرد إدخال اسمه .

والدالة بعد كتابتها في البرنامج تمر بمرحلة الترجمة إلى لغة الآلة ولا تنفذ إلا في مرحلة الربط، أما الماكرو وأثناء عملية الترجمة فيتم استبداله في كل سطر يتم استدعاؤه فيه بنتيجته النهائية ولا ينتظر مرحلة التنفيذ كالدالة.

ويمتاز الماكرو عن الدالة بالسرعة والسهولة في الكتابة بالإضافة لاستخدامه أنواعا محايدة من البيانات (فلم نشترط نوعا معينا من المتغيرات في تعريفنا للماكرو (**SQUARE(A)**) فهو لا يحتاج إلى تحديد النوع كما في الدوال.

وذلك بالإضافة إلى حصولنا على ملف تنفيذي أصغر في حالة استعمال الماكرو.

وبصفة عامة يوصى باستخدام الماكرو في العمليات القصيرة التي لا تتعدى سطرا واحدا.

مسائل على الباب

*دالة تاخد عددين من نوع انتجر وتعيد قيمة وتجمعهما

```
#include <iostream.h>
int addition (int a, int b)
{
int r;
r=a+b;
return (r);
}

int main ()
{  int z,b,n;
Cin>>b,n;
z = addition (b,n);
cout << "The result is " << z;
return 0;
}
```

*دالة لا تعيد قيمة من نوع فيود

```
#include <iostream.h>
void dummyfunction (void)
                {
cout << "I'm a function!";
}
int main ()
{
dummyfunction ();
```

```
return 0;
}
```

*دالة لإيجاد المضروب من نوع انتجر

```
#include <iostream.h>
long factorial (long a)
{
if (a > 1)
return (a * factorial (a-1));
else
return (1);
}

int main ()
{
long l;
cout << "Type a number: ";
cin >> l;
cout << "!" << l << " = " << factorial (l);
return 0;
}
```

*دالة تعمل عمل الاس ؟

*// // السلسلة الثالية ؟ ١,٢,٧,١٥,٣١,٦٣,١٢٧

*// ترسم الاشكال التالية

```
*          *          *****
***       ***        ***
*****     *****      *
          ***
           *
```

*برنامج على المؤشرات

```
#include <iostream.h>
int main ()
```

```
{
int value1 = 5, value2 = 15;
int * mypointer;

mypointer = &value1;
*mypointer = 10;
mypointer = &value2;
*mypointer = 20;
cout << "value1==" << value1 << "/ value2==" <<
value2;
return 0;
} value1==10 / value2==20
```

*ما ناتج البرنامج الاتي

```
#include <iostream.h>

int main ()
{
int numbers[5];
int * p;
p = numbers;  *p = 10;
p++;  *p = 20;
p = &numbers[2];  *p = 30;
p = numbers + 3;  *p = 40;
p = numbers;  *(p+4) = 50;
for (int n=0; n<5; n++)
cout << numbers[n] << ", ";
return 0;
}
```

10, 20, 30, 40, 50,

*اكتب دالة من نوع انتجر تعيد قيمة بواسطة مؤشر من نوع انتجر لايجاد المضروب للعدد المدخل ؟

* اكتب برنامج يقوم بعملية البحث عن احرف مكونة من سلسلة حرفية من نوع مؤشر ؟

* اكتب برنامج رئيسي يحتوي على استدعاء للدوال التالية

- دالة تستقبل عدد من نوع صحيح مؤشر وتقوم بايجاد الاس لة

- دالة تستقبل عدد من نوع اقتجر مؤشر وتقوم بايجاد عوامل العدد ؟

*برنامج للسجلات والدوال

```
#include <iostream.h>
#include <string.h>
#include <stdlib.h>

struct movies_t {
char title [50];
int year;
} mine, yours;

void printmovie (movies_t movie);

int main ()
{
char buffer [50];

strcpy (mine.title, "2001 A Space Odyssey");
mine.year = 1968;

cout << "Enter title: ";
cin.getline (yours.title,50);
cout << "Enter year: ";
cin.getline (buffer,50);
yours.year = atoi (buffer);
```

```
Enter title: Alien
Enter year: 1979

My favourite movie is:
 2001 A Space Odyssey (1968)
And yours:
 Alien (1979)
```

```
cout << "My favourite movie is:\n ";
printmovie (mine);
cout << "And yours:\n ";
printmovie (yours);
return 0;
}

void printmovie (movies_t movie)
{
cout << movie.title;
cout << " (" << movie.year << ")\n";
}
```

*متال للسجلات والمؤشرات

```
#inculde<iostream.h>
#include<string.h>
Typ struct{
Char name[30];
Int age;
}
Data;
Int main(){
Data*s,std;
S=&std;
Strcpy(std.name,''ammar aldopaee');
Std.age=20;
Cout<<std.name;
Cout<<std.age;
Cout<<s->name;
Return();
}
```


aldopaee@hotmail.com

المصفوفة هي مجموعة من العناصر من نفس النوع، وتكون عناصر المصفوفة مرتبة بحيث يمكننا الوصول لأي عنصر نريده بتحديد ترتيبه في المصفوفة.

والمصفوفات تنقسم لنوعين فهناك المصفوفات ذات البعد الواحد، والمصفوفات ذات البعدين.

مصفوفات البعد الواحد

المصفوفة ذات البعد الواحد هي مجموعة من العناصر مرتبة بحيث يمكن الوصول إلى أى عنصر فيها باستخدام ترتيبه بالنسبة لأول عنصر في المصفوفة

وفي لغة C++ يأخذ أول عنصر الرقم صفر.

والشكل التالي يوضح مصفوفة ذات بعد واحد

A= [2 3 4 5 6]

وعناصر المصفوفة مرتبة بدءا من العنصر الأول والذي يأخذ الرقم صفر

ويكون العنصر الأول A[0] مساويا للقيمة ٢.

وبالمثل يكون A[1] = 3 ، A[2] = 4 ، وهكذا...

مثال لجمع عناصر صفوفه احاديه

```
#include <iostream.h>

int billy [] = {16, 2, 77, 40, 12071};
int n, result=0;

int main ()
{
for ( n=0 ; n<5 ; n++ )
{
result += billy[n];
```

```
}
cout << result;
return 0;
}
12206
```

وسنتناول فيما يلي كيفية التعامل مع المصفوفات من خلال لغة C++ والإعلان عنها وتخصيص قيم للعناصر وطباعة العناصر وغيرها من اساليب معالجة المصفوفات.

المثال الموضح بالشكل التالي يوضح كيفية التعامل مع مصفوفة ذات بعد واحد بالاسم A

```
#include <iostream.h>
main()
{
1:      int A[4];

        for(int I=0 ; I<4 ; I++)
        {
              cout<<" Please enter the value of the element A[" << I
<< "]";
              cin >> A[I];
        }

        for(int I=0 ; I<4 ; I++)
        {
              cout<<" The value of the element A[" << I << "] is"
<< A[I];
        }
}
```

ويقوم المثال بعد ذلك بعدة عمليات نتناولها بالتفصيل

السطر المشار إليه بالرقم 1 يعلن عن المصفوفة وعناصر المصفوفة من النوع العددي الصحيح (int) وعدد العناصر أربعة.

والإعلان عن المصفوفة كالإعلان عن المتغيرات العادية يذكر نوع المتغيرات أولا ثم اسم المصفوفة متبوعا بعدد العناصر بين قوسين مربعين.

والحلقة **for** الأولى تقوم بتعبئة المصفوفة بالبيانات التي يدخلها المستخدم واحدا بعد الآخر، ويلحظ أنه لا بد لنا من حلقة تكرارية لإدخال البيانات في المصفوفة.

أما الحلقة **for** الثانية فتقوم بعرض عناصر المصفوفة التي تم إدخالها عنصرا عنصرا.

مصفوفات ذات بعدين

المصفوفة ذات البعدين تحتوي على عناصر من نفس النوع، ولكنها مرتبة في صفوف و أعمدة . وبالتالي تختلف طريقة الوصول للعناصر إذ يلزم لتحديد العنصر استخدام رقم الصف و رقم العمود و الشكل التالي يوضح مصفوفة ذات بعدين

$$B=\begin{pmatrix} 12 & 23 & 15 \\ 89 & 35 & 25 \\ 90 & 80 & 16 \end{pmatrix}$$

وعناصر المصفوفة في هذه الحالة كما ذكرنا تحدد باستخدام رقمين رقم الصف ورقم العمود، فالعنصر ١٢ يقع في العمود الأول والصف الأول أو بلغة الكمبيوتر

B[0][0]=12. لاحظ أن الترقيم في المصفوفة يبدأ بالرقم صفر دائما.

وبالمثل يمكن تحديد العناصر المختلفة ، ويذكر رقم الصف أولا ثم رقم العمود، والشكل التالي يوضح أمثلة لتحديد عناصر مختلفة من المصفوفة

B .

B[1][2] = 35

B[2][1] = 80

B[0][2] = 15

B[2][2] = 16

مثال

في أحد الفصول الدراسية كانت نتائج ثلاثة طلاب كما هو موضح بالجدول التالي

	المادة الأولى	المادة الثانية	المادة الثالثة
الطالب الأول	68	52	70
الطالب الثاني	82	92	90
الطالب الثالث	85	85	83

والآن لو طلب منا برنامج يمكنه التعامل مع هذا الجزء من النتيجة، فإننا نحتاج بكل تأكيد لمصفوفة ذات بعدين والبرنامج التالي يوضح كيفية إنشاء مصفوفة ذات بعدين، وبعد ذلك يطلب من المستخدم إدخال البيانات الموضحة في الجدول ويقوم بتخزينها في عناصر المصفوفة المناظرة، وبعد ذلك يطبع البرنامج القيم المدخلة.

```
#include <iostream.h>
main()
{
float Degrees[3][3];    // Array declaration

// Array Element entry

for (int I=0 ; I<3 ; I++)
{
        for(int J=0 ; J<3 ; J++)
        {
                cout<<" Enter the result of subject " << I  << "for
student  "<< J;
```

```
            cin >> Degrees[I][J] ;
        }
};
// Array elements display
for (int I=0 ; I<3 ; I++)
{
  for(int J=0 ; J<3 ; J++)
  {
      cout<<" the result of subject " << I  << "for student  "<< J
<<"is";
      cout<< Degrees[I][J] ;
  }
};
}
```

ويلاحظ استعمال حلقتين تكراريتين من النوع **for** لتخصيص البيانات للمصفوفة ولعرضها بعد التخصيص وكل من الحلقتين التكراريتين تتكونان من حلقة خارجية تقوم بزيادة عداد الأعمدة وحلقة الداخلية تقوم بزيادة عداد الصفوف.

وترمز الأعمدة هنا لرقم المادة بينما ترمز الصفوف لرقم الطالب.

مسائل على الباب

*برنامج يطبع الشكل الاتي

```
#include<iostream.h>
#include<conio.h>
main(){
clrscr();
int s,d,a,b,m,c,x;
cin>>s;x=s;
```

```
1
212
32123
4321234
```

```
for(d=1;d<=s;d++){
for(m=x;m>=1;m--){cout<<" ";}
x-=1;
if (d>=2){
for(c=d;c>=2;c--){
cout<<c;
}}
for(a=1;a<=d;a++){
cout<<a;}
cout<<"\n";
}
```

*برنامج يعكس اقطار المصفوفة

```
#include<iostream.h>
#include<conio.h>
main(){int a,b,i,j;
clrscr();a=4;
int
x[5][5]={1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20
,21,22,23,24,25};
for(i=0;i<5;i++){
for(j=0;j<5;j++){
if(i==j){b=x[i][j];x[i][j]=x[i][a];x[i][a]=b;}
} a--=1;
}
for(i=0;i<5;i++){
for(j=0;j<5;j++){

cout<<x[i][j]<<" ";
}
cout<<"\n";
}
getch();
}
```

*يجمع العناصر الواقعة تحت القطر الرئيسي

```
#include<iostream.h>
#include<conio.h>
main(){int a,b,i,j;
clrscr();a=4;
int
x[5][5]={11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,
27,28,29,30,31,32,33,34,35};
for(i=0;i<5;i++){
for(j=0;j<5;j++){
if(i>j){b+=x[i][j];}
}
}
for(i=0;i<5;i++){
for(j=0;j<5;j++){

cout<<x[i][j]<<" ";
}
cout<<"\n";
}       cout<<b;
getch();
}
```

*اكتب برنامج يعكس مصفوفة احادية بدون استخدام اداة شرطية ؟

*اكتب برنامج يحول الاعداد الفردية الى جهة اليمين والزوجية الى اليسار ؟

*اكتب برنامج للبحث عن اسم شخص معين في مصفوفة مؤشرات من نوع تشار ؟

*اكتب برنامج لادخال قيم لمصفوفة ذات بعدين من نوع مؤشر ويقوم بطباعتها؟

aldopaee@hotmail.com

الفصل السابع : الفصائل والكائنات (Classes & Objects)

سنتناول في هذا الفصل بشيء من التفصيل الفصائل والكائنات لنتعرف عن قرب على برمجة الكائنات الموجهة .

الفصيلة تتكون من بينات ودوال تتعامل مع هذه البيانات والشكل التالي (شكل١) يوضح الصورة العامة للإعلان عن الفصيلة

```
class class_name{
private:
      private data and functions
public :
      public data and functions
}
```

شكل ١ الصورة العامة للإعلان عن الفصيلة

والإعلان عن الفصيلة يتكون من:

أولا : الكلمة المحجوزة (class) يليها اسم الفصيلة (class_name) ويخضع اسم الفصيلة لقواعد عامة هي:

-ألا يكون اسم الفصيلة أحد الكلمات المحجوزة باللغة (Reserved words) او الكلمات التي تحمل معنى خاصا مثل(main) ويمكن التعرف على الكلمات المحجوزة باللغة من دفتر التشغيل المصاحب للمترجم.

- يمكن أن يحتوي الاسم على أى حرف من الحروف الأبجدية (A-Z) سواء صغيرة كانت أم كبيرة، وأي رقم من الأرقام (0-9) كما يمكن أن تحتوي على علامة الشرطة السفلى (_) ولكن لا يجوز أن يبدأ الاسم برقم.

– لا قيود على طول الاسم ، وتتيح هذه الميزة استخدام أسماء معبرة عن مضمونها، ومن الأفضل دائما استخدام الاسم المعبر عن محتوى الفصيلة لتسهيل التعامل مع الفصائل.

–الحروف الكبيرة و الصغيرة ليست متكافئة في لغة C++ فمثلا اسم البيان(MY_CLASS) يختلف عن الاسم (my_class) وكلاهما يختلف عن الاسم (My_Class).

ثانيا: تحديد درجة الحماية ، ونبدأ عادة بدرجة الحماية الخاصة (private) وتلي هذه الكلمة البيانات و الدوال الخاصة بالفصيلة.

ثالثا: تحديد درجة حماية أخرى ، وفي هذا المثال استخدمنا الدرجة العامة (public) وسنتعرف على درجات الحماية بالتفصيل في وقت لاحق.

والمثال الموضح بالشكل التالي (شكل ٢) يوضح كيفية استخدام الفصيلة في برنامج.

```
01: #include<iostream.h>
02: class smallobj                  // class name
03:{
04:    private:
05:           int somedata;       //class data
06:    public:
07:          void setdata (int d); // member function to set data
08:                 {somedata= d;}
09:          void showdata()  // mamber function to display data
10:                 { cout << " \n Data is " << somedata;}
11:};

12: void main()
13:{
14:    smallobj   s1,s2;    // define two objects of the class

15:    s1.setdata(1096);  //call member function to set data
```

```
16:    s2.setdata(200);

17:    s1.showdata();
18:    s2.showdata();
19:}
```

شكل ٢

يبدأ البرنامج بالتوجيه في السطر الأول ، وفائدة هذا التوجيه إخبار المترجم بمكان الملف المحتوي على تعريفات الدوال الأساسية والتي سنستخدمها في البرنامج.

وتتابع السطور بعد ذلك كالآتي:

السطر 02 : تعريف فصيلة جديدة تحمل الاسم (smallobj) ويلاحظ التعليق المكتوب بعد العلامة " //" ، وهذة الميزة لم تسمح بها لغة C .

السطرين 04 و 05: تعلنان عن بيان من النوع الصحيح.

السطر 06: يعلن درجة الحماية العامة، بمعنى أن ما سيأتي بعد ذلك سيكون عاما فيمكن للفصائل المشتقة أن تتعامل معه.

السطور من 07 إلى 10 فيهما تعريف الدالتين الوحيدتين في الفصيلة.

ويلاحظ في السطر 10 كيفية الطباعة على الشاشة وهو أسلوب جديد لم يكن مستعملا من قبل في لغة C. وسنتعرض للأساليب الجديدة في فصل مستقل.

وبداية من السطر 12 يبدأ البرنامج فعليا كالعادة بالدالة main() .

وفي السطر 14 نعرف كائنين من الفصيلة السابقة، ويلاحظ أن تعريف الكائنات يتم كتعريف المتغيرات العادية.

السطر 15 يستدعي الدالة (setdata()) للكائن الأول، وتجب ملاحظة طريقة الاستدعاء باستخدام المؤثر (.)، حيث يذكر اسم الكائن متبوعا بالمؤثر (.) ثم اسم الدالة المراد تنفيذها مع تخصيص قيم لمتغيرات الدالة.

السطرين 17 و١٨ يتم فيهما استدعاء الدالة (showdata()) لكل من الكائنين (s1,s2).

درجة حماية أعضاء الفصيلة:

تعرضنا لعبارة " درجة حماية أعضاء الفصيلة " والآن نتناول هذه العبارة بشيء من التفصيل.

أن درجة الحماية هي تحديد مدى القدرة على التعامل مع أعضاء الفصيلة (البيانات و الدوال)

والكلمات المستخدمة لتحديد درجة الحماية موضحة بالجدول التالي

الكلمة	المعنى
public	تعني أن البيانات التي تليها عامة ويمكن لأي دالة الوصول إليها و استعمالها.
protected	تفيد في حالة توريث الفصيلة، حيث يسمح للفصائل التي ورثت باستعمال أعضاء الفصيلة الأساسية
private	تعني أن البيانات خاصة بهذه الفصيلة ولا يمكن الوصول إليها إلا بواسطة دوال الفصيلة

دالة البناء :

ذكرنا سابقا أن دالة البناء ما هي إلا عضو من الفصيلة وتحمل نفس اسمها، وتنفذ هذه الدالة تلقائيا عند الإعلان عن كائن من الفصيلة.

ويمكننا أن نستفيد من هذه الدالة في تخصيص قيم لبعض بيانات الكائن عند الإعلان عنه.

و المثال التالي يوضح برنامجا قمنا فيه بالإعلان عن فصيلة ودالة البناء تقوم بعملية تخصيص القيم التي تأخذها لبيانات الفصيلة (a , b)

```
# include “iostream.h”
class MyClass
{
        int a,b;
public:
        MyClass (int i, int j)
        {
                a=i;
                b=j;
        }
        void Show()
        {
                cout << a<< “          “<<b;
        }
};
void main()
{
        MyClass      C1(2,6);
        C1.Show();
}
```

نلاحظ في هذا المثال أن الإعلان عن الكائن من الفصيلة (MyClass) لم يتم بالطريقة المعتادة حيث قمنا باستخدام قوسين بعد اسم الكائن وبينهما قيمتين عدديتين.

والإعلان هنا قام باستدعاء دالة البناء الخاصة بالفصيلة والتي بدورها تأخذ القيم المعطاة وتخصصها للبيانين (a,b) الموجودين بالفصيلة.

وعند استدعاء دالة (Show()) والتي تعرض قيم المتغيرين (a,b) نجد أنها تعرض القيم التي تم تخصيصها عند الإعلان عن الكائن (C1)

مصفوفة الكائنات:

المصفوفة هي مجموعة من العناصر من نفس النوع، وجرت العادة أن نعرف مصفوفات من أنواع المتغيرات المتاحة في اللغة.

و مع لغة C++ يتمكن المبرمج من الإعلان عن مصفوفة من الكائنات أيضا بنفس الكيفية التي يستخدمها للإعلان عن مصفوفة من المتغيرات العادية.

والمثال التالي يوضح كيفية الإعلان عن مصفوفة الكائنات

```
#include <iostream.h>
class Date
{     public:
            int day,month,year;
            set_date(int d, int m, int y)
            {day=d;   month=m; yaer=y;}
};
main()
{
      Date       date_array[3];
      date_array[0].set_date(2,3,1990);
}
```

ونلاحظ من هذا المثال أننا أعلنا عن مصفوفة كائنات من نوع (**Date**) وهي الفصيلة التي أعلنا عنها قبل الدالة الرئيسية مباشرة.

ونتعامل مع عناصر مصفوفة الكائنات بطريقة مماثلة لتعاملنا مع عناصر المصفوفات الأخرى، والسطر الثاني يوضح مثلا كيفية استدعاء الدالة (**set_date**) للعنصر الأول من عناصر المصفوفة.

استعمال المؤشرات مع الكائنات

المؤشرات (**pointers**) في لغات البرمجة لها أهميتها القصوى (والتي قد لا يدركها المبتدئ) ونتيجة لهذة الأهمية ظهرت الحاجة لاستخدام المؤشرات مع الكائنات.

وتدعم لغة C++ استخدام المؤشرات مع الكائنات، ويتم الإعلان عن مؤشر لكائن ما وللإعلان عن مؤشر لكائن من الفصيلة (Date) الموضحة في المثال السابق نستخدم العبارة (Date *dptr؛) كما بالمثال التالي

```
#include <iostream.h>
class Date
{     public:
            int day,month,year;
            set_date(int d, int m, int y)
            {day=d;   month=m; yaer=y;}
};
main()
{
      Date *dptr;
      Date   date;
      Dptr  -> day=3;
      Date.day=4;
}
```

ويلاحظ أنه في هذا المثال قد تم الإعلان عن كائن (date) ومؤشر إلى كائن (dptr) ، ومعاملة كل منهما تختلف عن الآخر حيث نستخدم المؤثر (<-) مع المؤشر للكائن للوصول إلى البيان (day) فيه بينما استخدمنا المؤثر (.) مع الكائن (date).

ويجب توخي الحرص دائما عند التعامل مع المؤشرات لتلافي الأخطاء التي يمكن أن تحدث.

اختبار تجريبي

١/أوجد المتسلسلات التالية

*أوجد هذه السلسلة بدون استخدام عداد مجمع للرقم ٤ مثل c+=4

1 5 9 13 17 21 25

1 5 9 17 3 36 4 20 81 *

2 6 12 20 30 42 56 72 *

1 3 7 15 31 63 127 *

1 3 6 18 24 30 36 *

٢/اكتب برنامج يستقبل عدد ويقوم بالأعمال التالية N=245496

مجموعهم A=30

متوسط الأعداد B=5

عددهم C=6

أصغرهم D=2

العدد المكرر E=4

أكبرهم H=9

٣/اكتب برنامج يطبع العنصر الاكثر تكرارا ؟

٤/كتب برنامج بواسطة دالة تأخذ قيمتين من نوع أتتجر وترد للبرنامج الرئيسي أربعة نواتج .....جمعهم.ضربهم.طرحهم.باقي القسمة ؟

٥/دالة تعمل على ترتيب مصفوفة وقت الإدخال ولاتقبل الأعداد المكررة ؟

٦/ دالة بواسطة مؤشر تدخل n من الأعداد وتطبع العناصر الغير متكررة ؟

٧/برنامج يبحث عن اكبر عنصر ويبدله مع أول عدد زوجي ؟

٨/مصفوفة ثنائية البعد تعرف بالدالة الرئيسية وتستدعيها دالة فرعية بواسطة مؤشر ليغير عناصر الصف الثاني فقط وتطبع المصفوفة ؟

aldopaee@hotmail.com

٩/بواسطة دالة واحدة فقط اعمل برنامج بإدخال عناصر لمصفوفة وتبديل الأعداد الفردية مكان الزوجية وطباعة المصفوفة ؟

١/ أنشئ الأشكال التالية

```
1111111111
1222222221
1233333321
1234444321
1234554321
1234444321
1233333321
1222222221
1111111111  n=10
```

```
1111111111111111
1222222222222221
1233333333321
1234444321
1234321
12321
121
1   n=4
```

```
*           *
**         **
***       ***
****     ****
*****   *****
**********
*****   *****
****     ****
***       ***
**         **
*           *
```

الخاتمة

إلى كل من انتفع بهدى الكتاب

إلى كل مسلم ومسلمة

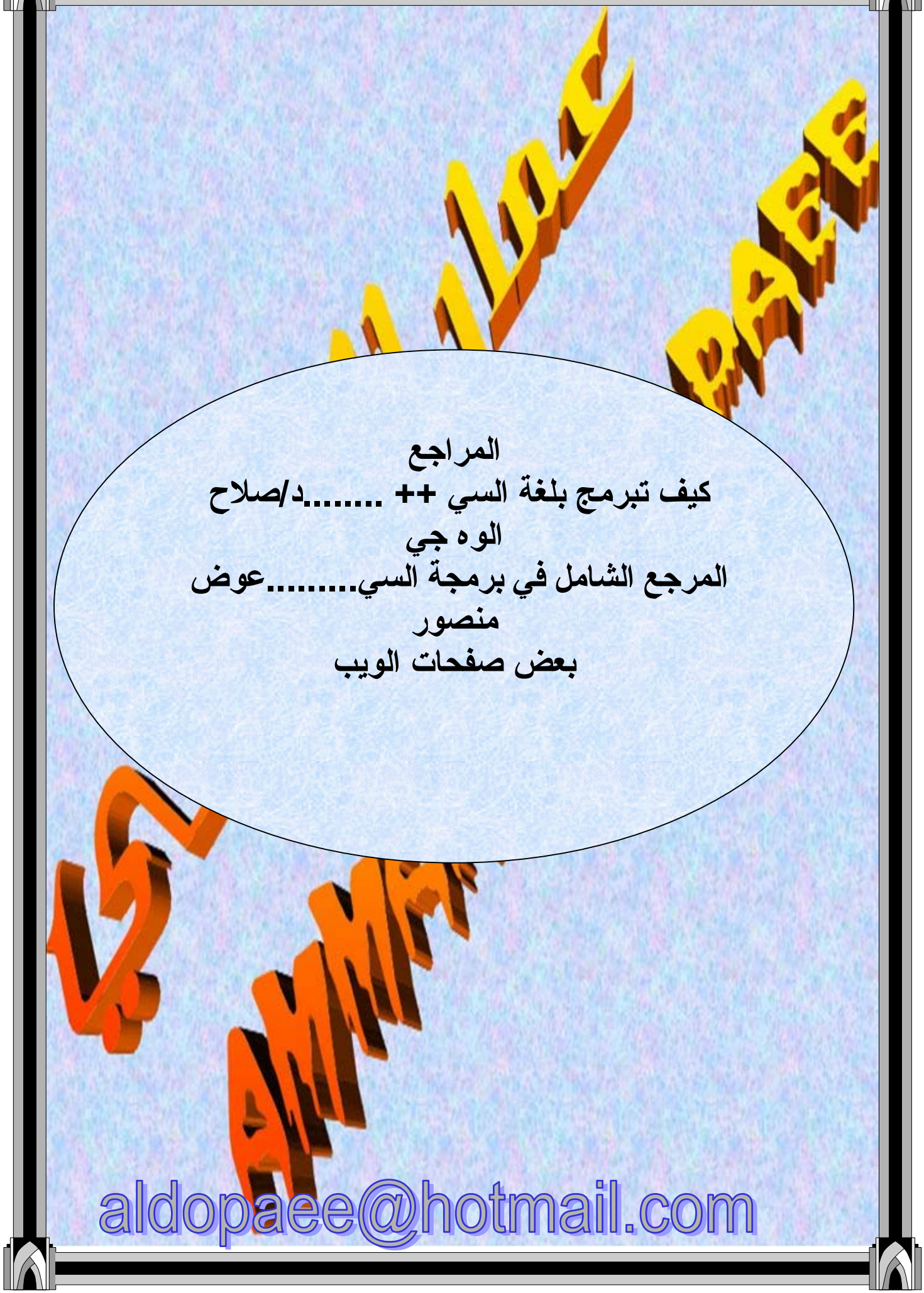
إلى كل من يشهد أن لإله إلا الله محمد رسول الله

ارجوا الدعاء لي ولمن يصلي على النبي الأمي

صلوات الله وسلامه عليك يا حبيبي محمد ابن عبد لله

والحمد لله.

aldopaee@hotmail.com



المراجع

كيف تبرمج بلغة السي ++ .....د/صلاح الوه جي

المرجع الشامل في برمجة السي.....عوض منصور

بعض صفحات الويب

aldopaee@hotmail.com