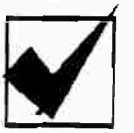
الباب الثانى

إضافات لغة سى++

محتويات الباب

باعتبار أن سى++ صورة متطورة من لغة سى فقد قدمت بعض العناصر الجديدة لتطوير اللغة علاوة على دورها الأصلى كلغة موجهة نحو الأهداف. وفى هذا الباب نقدم أهم هذه العناصر.



فى هذا الباب وفى الباب التالى ، سوف نعرض بعض شرائح من البرامج لمجرد التوضيح. ومن المفهوم أن البرنامج لكى يمكن تنفيذه ، لابد وأن تكتمل عناصره من إعلانات وملفات العناوين علاوة على البرنامج الرئيسى (main). فإذا أردت تجربة هذه الشرائح فعليك أن تكمل معالم البرنامج بنفسك. ومع ذلك فإنه علاوة على ذلك سوف نقدم بعض الأمثلة الكاملة القابلة للتنفيذ والتى سوف تحمل العناوين مثل "مثال (١-٢) ، مثال (٢-٤) .. الخ" ، ولأن هذه الأمثلة أمثلة عامة على لغة سى++ فيمكنك تنفيذها باستخدام أى مترجم للغة سى++.

(٢-١) علامة التعليقات Comments

علاوة على الطريقة القديمة لكتابة التعليقات فى لغة سى بوضع التعليق بين العلامتين "*/" و "/*" ، فإنك تستطيع تحويل السطر كله إلى تعليق باستخدام العلامة "//" فى بداية السطر ، مثل:

```
// Example 2-1
```

كما يجوز وضع العلامة فى أى مكان بالسطر ، فيتحول كل ما بعد العلامة إلى تعليق ، مثل:

```
int *ptr; // This is a pointer to an int.
```

(٢-٢) الإعلانات Declarations

يمكنك فى برنامج لغة سى++ أن تعلن عن المتغيرات فى أى مكـلن بالبرنامج ، بما فى ذلك متغيرات العدادات التى قد نعلن عنها بداخل قوس حلقة تكرارية كالمثال التالى:

```
for (int i=1; i<10; i++) {..
```

ومن الجدير بالذكر أنه عند استخدام هذه الرخصة تجـــب مراعـــاة حدود نطاقات المتغيرات ، ففى المثال السابق لا يجوز أن نعلن عن متغير آخر بالاسم i بداخل نفس البلوك (سيلى عـــرض موضـــوع النطاقات بالتفصيل).

(٢-٣) المراجع References

قدمت اللغة المراجع التى تماثل فى عملـــها المؤشـــرات (Pointers) لكنها تتميز عنها بسهولة الاستخدام.

فى المثال التالى نعلن عن مرجع Ref يشير إلى متغير صحيح x:

```
int x;
int &Ref = x;
```

إن هذا يناظر الإعلان عن المؤشر Ptr بالصورة الآتية:

```
int *Ptr;
```

مع الفارق أن المرجع لابد من شحنه بقيمة ابتدائية عند الإعلان عنه. وعندما يستخدم المرجع فى البرنامج فإنه يستخدم مباشرة كما المتغيرات العادية ، مثل:

```
Ref = 123;
```

وتخضع المراجع للقيود الآتية:

- لا يجوز تعريف مرجع إلى مرجع.
- لا يجوز تعريف مؤشر إلى مرجع (ولكن العكس جائز).
- لا بد من شحن المرجع بقيمة ابتدائية (بحيث يشير إلى بيان معين) ولا يجوز تغيير اتجاهه بعد إعلانه.

(٢-٤) الدوال الخطية Inline Functions

تستخدم الدوال الخطية لبرمجة الدوال الصغيرة فهى تتمتع بمميزات للماكرو (Macro) حيث أنها تترجم فوراً ولا تستدعى أية تحضيرات مسبقة من المترجم ، فوق أنها تتميز بما تتميز به الدوال من تمرير البارامترات واختبارها عند التنفيذ. فإذا أردت تعريف دالة خطية مثل دالة تربيع العدد ، فيمكنك كتابتها بالصورة الآتية:

```
inline int SquareIt(int x) {return x*x; }
```

وهذا يكافئ الماكرو:

```
#define SQUAREIT(Y) X*X
```

وإذا كنت قد استخدمت الماكرو من قبل فلعلك تعرف أن له مزالق يلزم الحذر منها ، أما الدوال الخطية فهى آمنة. وهذه هى أهم خصائص الدوال الخطية:

- إن إعلان الدالة كدالة خطية هو مجرد توصية للمترجم ولكنه ليس ملزماً. فقد يحولها المترجم إلى دالة عادية بحسب ظروف الترجمة.
- إذا استخدمت الدالة الخطية بداخل فصيلة (كما سيلى) فيجوز إغفال كلمة inline.

(٢-٥) المؤثرات الجديدة فى لغة سى++

قدمت اللغة مجموعة من المؤثرات الجديدة هى:

مؤثرات حجز وإتاحة الذاكرة (new, delete)

new	لحجز مكان فى الذاكرة ، وهو يناظر الدالة malloc.
delete	لإتاحة الذاكرة المحجوزة ، وهو يناظر الدالة free.

ويتميز المؤثر new بأنه لا يحتاج إلى عملية إسقاط (casting) كما الدالة malloc. أنظر المثال التالى حيث نستخدم الدالة malloc فى حجز مساحة من الذاكرة لمؤشر إلى عدد صحيح (int):

```
int *Ptr = (int *) malloc(sizeof(int));
```

إن هذا يمكن إنجازه بالمؤثر new كالآتى:

```
int *Ptr = new int;
```

ويتم إتاحة الذاكرة كالآتى:

```
delete Ptr;
```

كما يجوز حجز الذاكرة لمصفوفة كالمثال التالى:

```
char *name = new char[10];
```

وإتاحتها كالآتى:

```
delete [] name;
```

لاحظ أن بعض طرازات اللغة لا تستلزم استخدام الأقواس [] عند استخدام المؤثر delete مع المصفوفات.

مؤثرات الدخل والخرج (cin, cout)

مؤثر قناة الخرج <<
(Output Stream Operator)
(يسمى أيضاً مؤثر الحشر
Insertion Operator)
	يستخدم لحشر البيانات فى قناة الخرج cout.
	مثال ــ طباعة قيمة متغير مثل x:
	`cout << "The value of x is: " << x;`
	وهذا يكافئ عبارة دالة الطباعة printf الآتية:
	`printf("The value of x: %d", x);`

مؤثر قناة الدخل >> (Input Stream Operator) (يسمى أيضاً مؤثر الاستخراج Extraction Operator)	يستخدم فى استخراج البيانات من قناة الدخل cin. مثال ــ إدخال قيمة المتغيرات x, y: `cin >> x << y;`

وبالطبع فإن لهذه المؤثرات دوراً هاما فى التعامل مع الملفات. فالقنوات cin (هدف قناة الدخل) و cout (هدف قناة الخرج) يجوز

ربطها بأجهزة الخرج والدخل بأنواعها مثل لوحة الأزرار والشاشة والملفات على القرص الصلب. وتتميز مؤثرات الخرج والدخل بأنها مؤثرات "ذكية" لا تحتاج إلى فورمات فهى تتعرف على أنواع البيانات تلقائياً.

مؤثر النطاق :: (Scope Resolution Operator)

يقسم البرنامج إلى النطاقات الموضحة بالجدول التالى:

النطاق	التعريف
نطاق الملف (File Scope)	كل ما يقع خارج حدود الدوال وإعلانات الفصائل.
نطاق البوك (Block Scope)	كل ما يقع بين قوسى البلوك {}
نطاق الدالة (Function Scope)	يضم العناوين المستخدمة بداخل الدالة.
نطاق عينة الدالة (Prototype Scope)	نطاق بارامترات الدالة.
نطاق الفصيلة (Class Scope)	كل ما يقع بين قوسى الفصيلة {} مع ملاحظة أن الفصيلة قد يعلن عنها بأحد الكلمات: class أو struct أو union كما سيلى.

نطاقات البرنامج (Program Scopes)

ويستخدم مؤثر النطاق فى التوصل إلى البيانات المحجوبة عن نطاق البلوك الحالى. فلو كان لديك متغيران بالاسم x ، واحد بداخل نطاق

الملف ، والآخر بداخل نطاق البلوك ، فإنك ــ من داخل البلــوك ــ تستطيع طباعة محتويات المتغير التابع للبلوك كالآتى:

```
cout << x;
```

أما المتغير التابع لنطاق الملف فيمكنك طباعة محتوياته بالاســتعانة بمؤثر النطاق كالآتى:

```
cout << ::x;
```

كما أنك مع الفصائل سوف تستخدم هذا المؤثر لتحديد تبعيــة دالــة معينة (موجودة بنطاق الملف) إلى فصيلة معينة كالمثال الآتى:

```
void MyClass::MyMemberFunction(int x, int y)
{
...
```

فى هذا المثال تعتبر الدالـــة MyMemberFunction تابعــة للفصيلــة MyClass ، أو عضوة من أعضاء الفصيلة بتعبير أدق.

(٢-٦) التحميل الزائد للدوال والمؤثرات Overloading

تتميز لغة سى++ بإمكانية التحميل الزائد للدوال بمعنى أنك تستطيع أن تطلق نفس الاسم على أكثر من دالة بشرط أن تختلف الدوال فى نوع البارامترات المستخدمة حتى لا يحــدث الإبــهام (Ambiguity) أثناء الترجمة. وعلى سبيل المثال:

```
int SquareIt(int x) {return x*x;}
double SquareIt(double x) {return x*x;}
```

تستخدم الدالة الأولى لإيجاد مربع العدد الصحيــح (int) وتســتخدم الثانية لإيجاد مربع العدد الحقيقى ذى الدقــة المضاعفــة (double). ويتم استدعاء كل منهما بنفس الاسم وعلى المــترجم أن يســتدعى

الدالة المناسبة بحسب نوع البارامتر المستخدم. وفى الأبواب المتقدمة سوف نتعرف بطريقة بديلة لتحقيق نفس الهدف وهى استخدام العينات (Templates).

ملاحظة:

فى مقابل هذه الرخصة (التحميل الزائد) فإن سى++ أضافت قيداً على استخدام الدوال وهو ضرورة إعلان عينة الدالة (Prototype) قبل أى محاولة لاستخدامها ، بما فى ذلك الدوال المبنية فى اللغة مثل printf (والتى تم تعريفها فى ملف العناوين stdio.h).

ملاحظة

ولا يقتصر التحميل الزائد على هذا التطبيق البسيط ، فله دور هام فى تحميل المؤثرات بحيث أنك تستطيع التعامل مع الأهداف (objects) كما لو كنت تتعامل مع متغيرات بسيطة مثل:

```
MyObject > YourObject+1;
```

إن كلاً من الأهداف MyObject و YourObject عبارة عن عمارات متكاملة من المتغيرات والدوال ، ولكن خاصية التحميل الزائد للمؤثرات تمكنك من استخدامها كما لو كانت متغيرات بسيطة.

(٢-٧) البارامترات سابقة التعريف Default Parameters

يمكنك فى لغة سى++ أن تشحن بارامترات الدالة عند الإعلان عنها بقيم سابقة التعريف كالأمثلة الآتية:

```
MyFunction(int x=2, int y=3);
```

```
YourFunction(int x, int y=3);
```

ويلاحظ فى المثال الثانى أننا قد استخدمنا القيمة سابقة التعريف مع بارامتر واحد فقط (وهو البارامتر y).

وتخضع البارامترات سابقة التعريف للقيود التالية:

- تأتى البارامترات سابقة التعريف فى المؤخرة بينما تأتى البارامترات العادية فى المقدمة.
- تظهر البارامترات سابقة التعريف فى مكان واحد فقط: إما فى عينة الدالة (Prototype) أو فى عنوان الدالة (Header).

(٢-٨)الثوابت المسماة const

يمكنك فى لغة سى++ أن تستخدم الثوابت المسماة (const) فى تعريف المتغيرات كالمثال الآتى:

```
const int SIZE = 100;
int MyArray[SIZE];
```

لم يكن مثل هذا الإعلان ممكناً فى لغة سى. كما يمكن الإعلان عن الثوابت المسماة فى ملفات العناوين (Header Files) حيث أنها أصبحت ثوابت داخلية (Internal Linkage) ، وقد كانت فى لغة سى من قبل ثوابت خارجية (External Linkage) يمكن رؤيتها من داخل أى ملف من ملفات البرنامج ، وبالتالى لا يجوز وضعها فى ملف العناوين الذى قد تتضمنه جميع الملفات.

(٢-٩) النمط المتطاير volatile

أضافت سى++ نمطاً جديداً من أنماط البيانات وهو النمط المتطـــاير volatile. ويعلن عنه كالآتى:

```
volatile int x;
```

ويتميز المتغير المتطاير ــ كما يوحى اسمه ــ أن قيمته قد تتغـــير فى أى لحظة أثناء تشغيل البرنامج (بدون أى عملية تخصيـــص). وهو يعتبر النقيض للثابت المسمى الـــذى يحتفـــظ بقيمتـــه طـــوال البرنامج.

(٢-١٠) تطوير الأنماط المبتكرة User-defined Types

أضيفت الأنماط المبتكرة الآتية إلى اللغة بحيـــث يمكـــن معاماتـــها كالأنماط المبنية فى اللغة:

struct, enum, union, class

ففى لغة سى كان من اللازم استخدام اسم النمط (مثل struct) عنـــد إعلان متغير من متغيرات المنشأ ، كالمثال الآتى:

```
struct Student {
        ...
        ...
};
void main()
{
        struct Student record;
        ...
```

أما فى لغة سى++ فيمكنك التعبير عـــن نفـــس المضمـــون بـــدون استخدام كلمة struct فى إعلان المتغير ، كالآتى:

```
struct Student {
        ...
        ...
};
void main()
{
```

```
Student record;
...
```

مثال (٢-١): تطبيق عام على التحميل الزائد والدوال الخطية

```
// ***************** Example 2-1 ********************
#include <iostream.h>
inline double SquareIt(double x) {return x*x;}
inline int SquareIt(int x) {return x*x;}
void main()
{
        cout << SquareIt(5) << endl;
        cout << SquareIt(2.5) << endl;
}
// ************************************************
```

مثال (٢-١) يوجد البرنامج على القرص تحت الاسم Ex2-1.cpp

تنفيذ البرنامج

عند تنفيذ هذا البرنامج تحصل على النتيجة التالية:

```
25
6.25
Press any key to continue
```

تذكر هذه المصطلحات:

Type	نمط (طراز / نوع)
User-defined Type	نمط مبتكر
struct	منشأ
enum	نمط قائمة / قائمة
union	منشأ مشترك
volatile	النمط المتطاير
Typing System	نظام الأنماط
Header File	ملف العناوين
Prototype	عينة الدالة
Stream I/O Library	مكتبة قنوات الدخل والخرج

Output Stream Operator	مؤثر قناة الخرج >>
Left Shift Operator	مؤثر الإزاحة إلى اليسار >>
Input Stream Operator	مؤثر قناة الدخل <<
Right Shift Operator	مؤثر الإزاحة إلى اليمين <<
Input Stream Object	هدف قناة الدخل (**cin**)
Output Stream Object	هدف قناة الخرج (**cout**)
Memory Allocation Operator	مؤثر حجز الذاكرة (**new**)
Memory Deallocation Operator	مؤثر إتاحة الذاكرة (**delete**)
Scope	نطاق
Scope Resolution Operator	مؤثر النطاق ::
Name Space / Namespace	حيز الاسم
Visibility	الرؤية
Pointer	مؤشر
Reference	مرجع
Inline Function	دالة خطية
Ambiguity	الإبهام
Overloading	التحميل الزائد (للدوال أو للمؤثرات)
Default Parameters	البارامترات سابقة التعريف
External Linkage	الربط الخارجى
Internal Linkage	الربط الداخلى

ملاحظة: الكلمات المكتوبة بالبنط الثقيل كلمات محجوزة بلغة سي++