

الباب الثاني إضافات لغة سي++

محتويات الباب

التعليقات (Comments)

الدوال الخطية (Inline Functions)

المراجع والمؤشرات (References and Pointers)

التحميل الزائد للدوال (Overloading)

البارامترات سابقة التعريف (Default Parameters)

النمط المتطاير (volatile)

الثوابت المسماة (const)

الأنماط المبتكرة (User-Defined Types)

المؤثرات الجديدة في لغة سي ++ (Operators)

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



فى هذا الباب وفى الباب التالى ، سوف نعرض بعض شرائح من البرامج لمجرد التوضيح. ومن المفهوم أن البرنامج لى 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 والتي 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++.

Comments

(١-٢) علامة التعليقات

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.

Declarations

(٢-٢) الإعلانات

يمكنك في برنامج لغة سي ++ أن تعلن عن المتغيرات في أي مكان بالبرنامج ، بما في ذلك متغيرات العدادات التي قد نعلن عنها بداخل قوس حلقة تكرارية كالمثال التالي:

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

ومن الجدير بالذكر أنه عند استخدام هذه الرخصة تجب مراعاة حدود نطاقات المتغيرات ، ففي المثال السابق لا يجوز أن نعلن عن متغير آخر بالاسم i بداخل نفس البوك (سيلي عرض موضوع النطاقات بالتفصيل).

References

(٣-٢) المراجع

قدمت اللغة المراجع التي تماثل في عملها المؤشرات (Pointers) لكنها تتميز عنها بسهولة الاستخدام.

في المثال التالي 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 في 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.

Inline Functions

(٢-٤) الدوال الخطية

تستخدم الدوال الخطية لبرمجة الدوال الصغيرة فهي تتمتع بمميزات للماكرو (Macro) حيث أنها تترجم فوراً ولا تستدعي 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) Y*Y
```

وإذا كنت قد استخدمت الماكرو من قبل فلعلك تعرف أن له مزالِق يلزم الحذر منها ، أما الدوال الخطية فهي آمنة. وهذه هي 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 سيلي) فيجوز إغفال كلمة inline.

(٢-٥) المؤثرات الجديدة فى لغة سى ++

قدمت اللغة مجموعة من المؤثرات الجديدة هي:

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 كالاتى:

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

ويتم إتاحة الذاكرة كالاتى:

```
delete Ptr;
```

كما يجوز حجز الذاكرة لمصفوفة كالمثال التالى:

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

وإتاحتها كالاتى:

```
delete [] name;
```

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

مؤثرات الدخل والخرج (cin, cout)

<p>يستخدم لحشر البيانات فى قناة الخرج cout.</p> <p>مثال — طباعة قيمة متغير مثل x:</p> <pre>cout << "The value of x is: " << x;</pre> <p>وهذا يكافئ عبارة دالة الطباعة printf الآتية:</p> <pre>printf("The value of x: %d", x);</pre>	<p>مؤثر قناة الخرج <<</p> <p>(Output Stream Operator)</p> <p>(يسمى أيضاً مؤثر الحشر)</p> <p>(Insertion Operator)</p>
<p>يستخدم فى استخراج البيانات من قناة الدخل cin.</p> <p>مثال — إدخال قيمة المتغيرات x, y:</p> <pre>cin >> x << y;</pre>	<p>مؤثر قناة الدخل >></p> <p>(Input Stream Operator)</p> <p>(يسمى أيضاً مؤثر الاستخراج)</p> <p>(Extraction Operator)</p>

وبالطبع فإن لهذه المؤثرات دوراً هاماً 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 فهي تتعرف على أنواع البيانات تلقائياً.

مؤثر النطاق :: (Scope Resolution Operator)

يقسم البرنامج إلى النطاقات الموضحة بالجدول التالي:

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

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 — من داخل البلوك — تستطيع طباعة محتويات المتغير التابع للبلوك كالاتي:

```
cout << x;
```

أما المتغير التابع لنطاق الملف فيمكنك طباعة محتوياته بالاستعانة بمؤثر النطاق كالاتي:

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

كما أنك مع الفصائل سوف تستخدم هذا المؤثر لتحديد تبعية دالة معينة (موجودة بنطاق الملف) إلى فصيلة معينة كالمثال الآتي:

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

```
{
```

```
...
```

في هذا المثال تعتبر الدالة MyMemberFunction تابعة للفصيلة MyClass ، أو عضوة من أعضاء الفصيلة بتعبير أدق.

(٢-٦) التحميل الزائد للدوال والمؤثرات Overloading

تتميز لغة سي++ بإمكانية التحميل الزائد للدوال بمعنى أنك تستطيع أن تطلق نفس الاسم على أكثر من دالة بشرط أن تختلف الدوال في نوع البارامترات المستخدمة حتى لا يحدث الإبهام (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 الصحيح (int) وتستخدم الثانية لإيجاد مربع العدد الحقيقي ذي الدقة المضاعفة (double). ويتم استدعاء كل منهما بنفس الاسم وعلى المترجم أن يستدعي

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

ملاحظة:

في مقابل هذه الرخصة (التحميل الزائد) فإن سى++ أضافت قيوداً على استخدام الدوال وهو ضرورة إعلان عينة الدالة (Prototype) قبل أى محاولة لاستخدامها ، بما فى ذلك الدوال المبنية فى اللغة مثل printf (والتي 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.

Default Parameters (٧-٢) البارامترات سابقة التعريف

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

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

YourFunction(int x, int y=3);

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

وتخضع البارامترات سابقة التعريف للقيود التالية:

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

const

(٢-٨) الثوابت المسماة

يمكنك في لغة سي++ أن تستخدم الثوابت المسماة (const) في تعريف المتغيرات كالمثال الآتي:

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

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

volatile

(٢-٩) النمط المتطاير

أضافت سي++ نمطاً جديداً من أنماط البيانات وهو النمط المتطاير volatile. ويعلن عنه كالاتي:

```
volatile int x;
```

ويتميز المتغير المتطاير – كما يوحي اسمه – أن قيمته قد تتغير في أي لحظة أثناء تشغيل البرنامج (بدون أي عملية تخصيص). وهو يعتبر النقيض للثابت المسمى الذي يحتفظ بقيمته طوال البرنامج.

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

أضيفت الأنماط المبتكرة الآتية إلى اللغة بحيث يمكن معاماتها كالأنماط المبنية في اللغة:

struct, enum, union, class

ففي لغة سي كان من اللازم استخدام اسم النمط (مثل struct) عند إعلان متغير من متغيرات المنشأ ، كالمثال الآتي:

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

أما في لغة سي++ فيمكنك التعبير عن نفس المضمون بدون استخدام كلمة struct في إعلان المتغير ، كالاتي:

```
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 للدالة
Stream I/O Library	مكتبة قنوات للدخل 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 الخارجي
Internal Linkage	الربط الداخلي

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