مكتبة القوالب المعيارية Standard Template Library

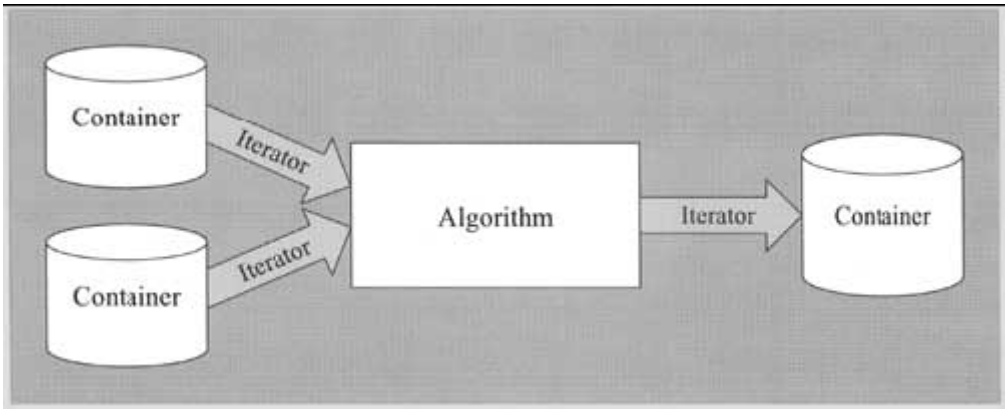
الدرس الأول :

وتختصر بـــ STL وتمثل دعم للغة C++ من توفير عدد من القوالب والخوارزميات الجاهزة التي تزيد من إنتاجية اللغة وتساعد على اختصار الوقت في كتابة الأكواد وغير ذلك فالمكتبة تعتبر من أسرع المكتبات وأقواها أداءً وهذه واضح من بُنيتها وأسس علاقاتها الداخلية وهذه المكتبة مبنية على القوالب "template" أي أنها تدعم أي نوع وهذا ما زادها قوة بالإضافة إلى دعمها مفهوم **generic programming** و **meta programming** وأيضاً تدعم مبادئ **Object Oriented Programming** وهذا مما يزيد في قوتها.

و لكي تفهم تركيبة هذه المكتبة و تحسن التعامل معها يجب أن تعرف أنها تقوم على ثلاثة مفاهيم أساسية و هي :

الوعاء (Container) : وهو يحتوي على عدد من الكائنات التي لها نفس النوع ويقوم بإدارتها بواسطة الطرق "method" الموجودة به مثل . vector , list , stack , set , map , multiset
الخوارزمية (Algorithm) : من اسمها هي خوارزميات تقوم بعدد من الوظائف والمهام على الوعاء مثل الحذف أو الإضافة أو البحث مثل. copy , search , find , find_if , sort , qsort ,remove
المكرر (Iterator) : وهو يستخدم للتنقل بين عناصر الوعاء و هو الوسيط الذي يتم به التخاطب بين الوعاء و بين الخوارزميات فالخوارزميات لا تستقبل Container كوسيط بل تستقبل iterator يؤشر على بداية عناصر الوعاء ونهايته أو يؤشر على مجموعه جزئية منه.

و هذا رسم يوضح العلاقة بين هذه العناصر الثلاثة.



ومن الرسم يتضح أن الـ Container تُرسل إلى الـــ Algorithm عن طريق الـ Iterator فالمكرر هو لغة التخاطب بين الوعاء والخوارزمية.

إلى هنا شرحنا المفاهيم الأساسية لكن هل تكفي لفهم هذه المكتبة ؟؟؟ بالطبع لا توجد عدة مفاهيم أخرى يجب الإلمام بها مثل:

Adapter : وهو يقوم بإضافة واجهة جديدة لـــ component موجودة بحيث تستطيع استخدام نفس component لكن بواجهة جديدة تتناسب مع ما حاجتك .
Fucntion Object : وتسمى بإختصار functor وهو كائن لكن يقوم بمهام function العادية إي أنك لن تفرق بينه وبين الدالة العادية لأنه قام بإعادة تعريف للمعامل ( ) فصار مثل الدالة العادية في الاستدعاء وله فوائد كثيرة سنذكرها في حينها إن شاء الله .
Allocator : وهو المسؤول عن إدارة الذاكرة وحجز وإرجاع الذاكرة واستخراج العناوين للكائنات في هذه المكتبة . والحالة الافتراضية له هو أنه يستخدم new و delete العادية لكن ممكن تطويرة ليقوم مقام Garbage Collection في المكتبة.

عموماً هذه تعتبر مقدمة عن STL وعن المفاهيم المتعلقة بها وفي الدرس المقبل سأشرح عن Containers وسنأخذ مثالين عمليين منها وهي vector و string

Generic Programming :

هو مصطلح يطلق على برمجة تقوم على كتبة خوارزميات و أصناف عامة يمكن إستخدامها مع أغلب الأنواع دون الحاجة إلى إعادة كتابتها مرة أخرى . وفي لغة C++ يتم ذلك عن طريق إستخدام الـ template مثال :

```
template <class T>
class Array{
public:
	T  obj;
};
```

الآن لو قلنا Object <int> فسيترجم الكود السابق إلى :

```
class Object{
public:
		int obj;
};
```

ونفس الكلام لو عرفنا صنف وسمينها مثلاً Student فنقدر نقول Object <Student> فيتفسر الكود السابق كالتالي :

```
class Object{
public:
		Student obj;
};
```

قد يتساءل البعض ما الفائدة من هذا ؟ أقول وبكل بساطة افرض أنك تريد تكون صنف (class) ويعمل عمل المصفوفات ويحتوي على طرق تساعده على معالجة وإدارة نفسه مثل class vecotr في STL فإن أسهل الطرق هي أن تجعله من نوع template بحيث يصبح :

```
template <class T>
class Object{
public:
		T * obj;
};
```

وبكذا تستطيع تكون Array <int> أو من أي نوع .

ونفس الكلام ينطبق على الخوارزميات بدل ما تكون عندك خوارزمية تستقبل مصفوفة من أعدد صحيحة وتطبعهم يمكن نخليها مصفوفة عامة من أي نوع و بهذا سيوفر علينا وقت كتابة الكود لأننا سنكتب خوارزمية واحدة فقط.

Meta Programming :

هي برمجة تقوم على كتابة برامج تقوم بنفسها بتوليد و إنتاج أكواد أو بيانات خاصة بها عند ترجمة الملف . طبعا في اللغات ذات interpreter مثل php و غيرها اللي تكون الترجمة فيها وقت التنفيذ تكون هذه البرمجة قوية ويمكن منها تعمل أي شئ يخطر على بالك لكن في لغات الـ Compiler تكون البرمجة أضيق قليلاً إلا أنها محتفظه بقوتها . في لغة C++ يمكن تطبيق هذا المفهوم عن طريق preprocessor أو عن طريق template الذي يعتبر الأفضل وأقوى . فمثلاً : تأمل هذا الكود

```
#define MAX(x,y) ((x) > (y) ? (x) : (y))

int hello()
{
    int i1 , i2;
    cin>>i1;
    cin>>i2;
    return MAX(i1,i2);
}
```

هنا سيتحول البرنامج بعد ترجمته إلى :

```
int hello()
{
    int i1 , i2;
    cin>>i1;
    cin>>i2;
    return ((x) > (y) ? (x) : (y));
}
```

أنت الآن كتبت برنامج قام بإنتاج كود بنفسه لكن هذا بإستخدام preprocessor اما باستخدام الـ template فإنك تستطيع عمل أكثر من هذا تستطيع عمل if-statement أو while او switch أو حتى Recursive أو حتى bubble Sort و بهذا بيختصر عليك عملية كتابة الكود لكن راح يكون الكود أكبر ومحدود في وقت الترجمة .

وللاستزادة زوروا هذا الرابط http://osl.iu.edu/~tveldhui/papers/Templat...s/meta-art.html
والذي يملك إضافة ممكن يضيف و منها أستفيد و يستفيد الأعضاء

أتمنى أني شرحت هذه المفاهيم و السلام عليكم و رحمة الله و بركاته

B.M.S