البرمجة غرضية التوجه في لغة C++ :

تعتمد البرمجة غرضية التوجه على ثلاث مبادئ أساسية :

1- التجريد وهو يتمثل بـ الصفوف والأغراض / objects and classes
2- الوراثة
3- تعددية الأشكال

الصفوف والأغراض : (هام جداً)

الصف : هو طريقة برمجية تمكن من تعريف أنواع جديدة غير موجودة في اللغة.

الشكل العام للصف :

class اسم الصف {
فراغ

متحولات من أنواع موجودة (تسمى الأعضاء البيانية للصف)

دوال (تسمى الدوال الأعضاء للصف)

};

ملاحظات :

1- اسم الصف هو اسم كيفي
2- نسمي الدوال بالطرق عندما توجد داخل الصف

اصطلاح :

عندما نعرف صف نبدأ به بحرف كبير وعندما نعرف طريقة نبدأ بحرف صغير.

مثال : تعريف عدد عقدي ((كتابة صف يمثل عدد عقدي))

نحتاج إلى بناء نوع جديد حقيقي وتخيلي لذلك نستخدم الصفوف , حيث أن العدد المركب لا يمكن تمثيله بمتحول واحد.

```
class Complex {
double x;        يمثل القسم التخيلي
double y;        يمثل القسم الحقيقي
void print () {
if (x>0)  cout<<y<<"+i"<<x;
else cout<<y<<"-i"<<-x;
}
};
```

ملاحظات :

- جميع الطرق المعرفة داخل الصف تستطيع الوصول و رؤية لكافة الأعضاء البيانية الخاصة بالصف.
- لا يمكن استخدام الأنواع بشكل مباشر في البرامج وإنما يجب أن نعرف عليها متغيرات ونعمل عليها أي :

```
int x;
cout<<int+int;        خطأ
cout<<x+x;            صح
```

حتى نستطيع استخدام النوع الجديد Complex يجب أن نصرح (نعرف) عن متحول من هذا النوع (الصف)

```
Complex c,c1,c2;
```

حيث c,c1,c2 نسخة طبق الأصل عن متحولات الصف Complex أي أن له ذات الأعضاء البيانية.

أي : c

x
y

c1

x
y

c2

x
y

وعندما نريد إعطاء قيمة لـ x و y فإننا نقوم بالكتابة بالشكل :

```
c.y=1;        c.x=2;

c1.y=-3;      c1.x=4;
```

وذلك لكي نفرق بين كل من c,c1,c2 بسبب تشابهها من حيث أعضائها البيانية.

الغرض : هو متحول من نوع صف

ملاحظة : عندما نريد استدعاء دالة من أجل غرض (متحول من نوع الصف) نقوم بكتابة ما يلي :

(وسطاء الدالة) اسم الدالة.اسم الغرض ;

طباعة العدد العقدي c :

```
c.print ();

c1.print ();
```

ملاحظة :

تعليمة cout تقوم بعملية الطباعة البسيطة ولا تطبع complex.

ملاحظة :

إن مجال الرؤية لمكونات الصف متاح لكل ما هو داخل الصف.

أما خارج الصف فإن مجال الرؤية (الوصول) إلى مكونات الصف يكون باستخدام محددات الوصول وهي ثلاث أنواع :

1- المحدد العام (public) :

عند وضع كلمة public أمام عضو بياني أو طريقة داخل الصف فهذا يعني أنه يمكن الوصول لها من خارج الصف ولأي كان.

2- المحدد الخاص (private) :

يجعل كل ما يحدد به من أعضاء بيانية أو طرق خاص بهذا الصف ولا يمكن رؤيته أو الوصل إليه من خارج الصف مهما كان.

3- المحدد المحمي (protected) : "خاص بالوراثة"

يمكن رؤية ما يحدد به من خلال صفوف الوراثة فقط.

نقطة هامة :

إذا لم نحدد محدد وصول لمكونات الصف فالمحدد الافتراضي يكون من النوع private وبالتالي لا يمكن الوصول إليها من خارج الصف.

ملاحظة حول المحددات :

- إن المحددات تأخذ مجالاً ولا توضع أمام كل متحول لوحده.
- فإذا أردنا إخفاء متحول ما وعدم إخفاء متحول أخر فنضع (محدد) أمام الذي نريد إخفاءه ويبقى ذلك المحدد ساري المفعول حتى يرى محدد أخر فيتغير حسب الجديد.

ملاحظة (هامة جداً جداً) :

لبناء غرض ما أي (حجز مكان له في الذاكرة وإعطاء قيم لأعضائه البيانية) نحن بحاجة (صف) وذلك لوجود طريقة خاصة وتدعى "الباني" (باني الصف) تمكننا من إنجاز المطلوب.

تذكر : الطريقة هي دالة داخل الصف.

باني الصف :

هو طريقة خاصة لا يوجد لها نوع إرجاع وتحمل نفس اسم الصف.

ومهمتها الأساسية هي إعطاء قيم للأعضاء البيانية الخاصة بالغرض.

ملاحظة :

الهدف من الباني هو حجز مكان وإعطاء قيم للأعضاء البيانية.

مثال :

```
class complex {
public :
double x;
double y;
complex (double a,double b) {
x=a; y=b;
}
void print () {
if (x>0)  cout<<y<<"+i"<<x;
else cout<<y<<"-i"<<-x;
}
};
```

ومن خارج الصف وبوجود الباني إذا كتبنا :

```
complex c(2,2);
```

فهذا يعني استدعاء الباني ووضع x=2,y=2

الباني الافتراضي :

هو باني لا يأخذ وسطاء وإنما هو فقط يأخذ مكان ويقوم بإعطاء القيم الافتراضية للأعضاء البيانية التابعة للغرض.

ملاحظات :

- لاستخدام الأغراض يجب بناؤها لذلك عند عدم وجود باني في صف ما تقوم اللغة بتوليد باني افتراضي بشكل أوتوماتيكي.
- الصف الذي لا يوجد فيه باني لا يمكن استخدامه.
- إذا أردنا بناء الغرض بأكثر من شكل فيمكن وضع أكثر من باني في الصف ونميز بينها بتغير نوع الوسطاء أو عددها أو ترتيبها في حال كانت من أنواع مختلفة (مثل فقرة التحميل الزائد في الدوال).

مثال :

```
complex (double a) {
x=a;
y=0;
}
```

ملاحظة هامة للامتحان :

اقرأ السؤال كاملاً قبل الإجابة لمعرفة بماذا يتعلق كل طلب بسابقه لتفادي إعادة صياغة البرنامج من جديد.

توضيح حول الباني :

عندما كنا نعرف متحول ونُسند له قيمة ما فإن يقوم البرنامج يقوم بحجز مكان للقيمة في الذاكرة

مثال :

```
int x=3;
```

يقوم بحجز خانة للرقم 3 كما يلي $\overbrace{\boxed{3}}^{x}$

أما الآن فعندما نعرف صف فيقوم البرنامج بالتالي

complex c; $\Longrightarrow$ $\overbrace{\begin{array}{|c|c|}\hline x & \text{قيمة} \\ \hline y & \text{قيمة} \\ \hline \end{array}}^{c}$

... انتهت المحاضرة (9) ...