البرمجة غرضية التوجه

تهدف البرمجة غرضية التوجه (Object-Oriented Programming) إلى تضمين المعطيات (الصفات Attributes) والتوابع (الأفعال Behaviors) ضمن مكونات تسمى الصفوف Classes وهما مرتبطان مع بعضهما البعض بشكل كبير.

يمكن استخدام الصف عدة مرات لإنشاء العدد الذي نريده من الأغراض المشتقة من نفس الصف.
يطلق على المعطيات الخاصة بصف بالمعطيات الأعضاء (Data Member)، وعلى التوابع الخاصة بهذا الصف بالتوابع الأعضاء (Member Functions) أو بالطرائق (Methods)، وبنفس الطريقة التي نسمي فيها اشتقاق لنسخة من نمط معرف سابقاً مثل (int) متحولاً فإننا نسمي أي اشتقاق من صف معرف من قبل المبرمج بالغرض (object).

بناء نمط المعطيات Date باستخدام مفهوم الصفوف :

تساعد الصفوف على نمذجة الأغراض التي لها عدة صفات والتي يتم تمثيلها على شكل معطيات أعضاء، كما يمكنها أن تقوم بتنفيذ عدة أفعال أو عمليات والتي يتم تمثيلها على شكل توابع أعضاء.
تُعرف الصفوف في لغة الـ C++ بواسطة الكلمة المفتاحية class، ويستخدم اسم الصف للتصريح عن الأغراض التي تنتمي إليه ، أما الطرائق فيتم التعامل معها من خلال رسائل مرسلة من قبل الغرض بحد ذاته بحيث ترتبط كل رسالة بعملية استدعاء لإحدى الطرائق.

```
class date{
        private:
          int day,month,year;
        public:
              date(int=1,int=1,int=2000);
              void set(int,int,int);
              int getday( ) {return day;}
              int getmonth( ) {return month;}
              int getyear( ) {return year;}
              void print( );
              ~date(){}
};
```

```
date::date(int d1,int m1,int y1)
  { day=d; month=m; year=y; }
void date::set(int d,int m,int y)
 { day=d; month=m; year=y; }
void date::print( )
 {cout<<day<<"-"<<month<<"-"<<year<<endl;}
```

يبدأ تعريف الصف date بالكلمة المفتاحية class، ويتم تحديد جسمه بواسطة القوسين الكبيرين {} وينتهي التعريف بالفاصلة المنقوطة .

يتضمن تعريف الصف date المعطيات الأعضاء التالية day, month, year ، والتوابع الأعضاء date, set, getday, getmonth, getyear, print

تدعى الأسماء private, public بمحددات الوصول إلى الأعضاء، وتستخدم للتحكم بالوصول إلى المعطيات والتوابع الأعضاء المتعلقة بالصف، حيث يفيد التصريح عن المعطيات والتوابع الأعضاء بعد المحدد public في جعل هذه المعطيات أو التوابع متاحة للاستخدام من أي نقطة ضمن البرنامج، أما المحدد private فهو نمط الوصول الافتراضي يجعل المعطيات والتوابع الأعضاء المصرحة بعده متاحة فقط للتوابع الأعضاء المرتبطة بالصف.

تنتهي أسماء المحددات دائماً بنقطتين (:) ويمكن أن تظهر عدة مرات ولايوجد أي ترتيب لورودها ضمن تعريف الصف، وينتهي مفعول محدد ما بورود اسم محدد آخر.

يتضمن تعريف الصف date مجموعة التوابع الأعضاء التالية الخاصة بالمحدد public :

date, set, getday, getmonth, getyear, print

والتي تشكل التوابع الأعضاء الممكنة الاستخدام من أجزاء البرنامج المختلفة.

يظهر تعريف ثلاثة أعداد صحيحة بعد المحدد private وهي تدل على أنه لايمكن الوصول إلى المعطيات الأعضاء إلا من خلال التوابع الأعضاء.

نلاحظ استخدام العملية الثنائية (::) لتحديد مجال الرؤية وذلك عند تعريف أي تابع من التوابع الأعضاء الواردة بعد تعريف الصف، حيث يسبق اسم التابع ذكر اسم الصف متبوعاً بالعملية الثنائية لتحديد مجال رؤيته على اعتبار أنه يمكن أن يكون لعدة توابع أعضاء من صفوف مختلفة نفس الترويسة بالتالي تقوم العملية الثنائية بالربط مابين اسم التابع العضو و الصف بحيث نحصل على تحديد وحيد للتوابع الأعضاء المرتبطة بصف محدد.

- **إعطاء قيم ابتدائية لأغراض صف (الباني Constructor):**

عند إنشاء غرض من صف يمكن لمعطياته أن تأخذ قيم ابتدائية بواسطة تابع يدعى بالباني مرتبط بهذا الصف. يحمل باني الصف نفس اسم الصف، ولا يمكن أن تحدد نمط القيمة المعادة منه، ويتم استدعاؤه بشكل أتوماتيكي عند إنشاء أي غرض من الصف.

يمكن تحديد قيم افتراضية للوسطاء الممررة للباني ففي المثال السابق صف التاريخ date تم تحديد القيمة الافتراضية ١ لليوم و القيمة الافتراضية ١ للشهر و القيمة الافتراضية ٢٠٠٠ للسنة و يتم إسناد هذه القيم للغرض في حال عدم تحديد قيم مغايرة عند إنشاء الغرض.

```
date(int=1,int=1,int=2000); //التصريح عن الباني
```

```
date::date(int d1,int m1,int y1)
  { day=d; month=m; year=y; }//جسم الباني
```

و نلاحظ هنا أن التصريح عن الباني موجود ضمن الصف date بينما الباني بحد ذاته (أي جسم الباني) موجود خارج الصف date لذلك لا بد من استخدام العملية الثنائية (::) لتحديد مجال الرؤية لتوضيح انتماء هذا الباني إلى الصف date.

- **كيفية الوصول إلى أعضاء الصف:**

تشابه العمليات المستخدمة للوصول إلى أعضاء صف العمليات المستخدمة للوصول إلى حقول السجلات، بالتالي الوصول إلى أعضاء الصف باستخدام العملية (.) مصحوبة مع اسم الغرض.

يجب ضبط عملية الوصول إلى المعطيات وخاصة المعطيات الخاصة بشكل جيد من خلال توابع الوصول التي يطلق عليها توابع (get) وتوابع(set) ، فعلى سبيل المثال للسماح لمستخدمي الصف بقراءة قيمة من المعطيات الخاصة نقوم بكتابة التابع العضو من النوع get المناسب للقيام بذلك، وللسماح لمستخدمي الصف بتعديل قيمة من المعطيات الخاصة نقوم بكتابة التابع العضو من النوع set المناسب لتنفيذ مثل هذه العملية.

يتضمن المثال السابق صف التاريخ date توابع الوصول التالية :

١. توابع الـ get :

```
int getday( ) {return day;}
int getmonth( ) {return month;}
int getyear( ) {return year;}
```

و نلاحظ هنا أن جسم كل تابع من هذه التوابع يأتي مباشرة بعد ترويسة التابع مباشرة و لذلك لا داعي لاستخدام العملية الثنائية (::) لتحديد مجال الرؤية فالتابع موجود بشكل كامل ضمن الصف date

٢. تابع الـ set :

```
void set(int,int,int);//التصريح عن التابع
```

```
void date::set(int d,int m,int y)
  { day=d; month=m; year=y; }//جسم التابع
```

و نلاحظ هنا أن التصريح عن التابع set موجود ضمن الصف date بينما التابع بحد ذاته (أي جسم التابع) موجود خارج الصف date لذلك لا بد من استخدام العملية الثنائية (::) لتحديد مجال الرؤية لتوضيح انتماء هذا التابع إلى الصف date.

ويتم استدعاء الطرائق من خلال مجموعة من الأغراض ضمن التابع الرئيسي كما يلي :

```
main( )
{
  date da,da1(1,2,2003);
  int x,y,z;
  cin>>x>>y>>z;
  da1.set(x,y,z);
  da.print( );
  da1.print( );
  cout<<da.getday( )<<endl;
  return 0;
}
```

- **استخدام الهادم :**

الهادم (destructor) عبارة عن تابع عضو يحمل نفس اسم الصف مسبوقاً بالحرف (~)، ولايأخذ أي وسيط و لا يعيد أي قيمة، والهدف من استخدامه اتمام دور الباني، ويكون لكل صف هادم وحيد فقط.

يتم استدعاء الهادم عند تدمير الغرض، و لا يقوم الهادم بتدمير الغرض بنفسه و إنما يقوم بإنهاء عملية الاحتفاظ بفضاء الذاكرة المخصص للغرض قبل أن يبدأ النظام بالمطالبة به و استخدامه من أجل الاحتفاظ بأغراض جديدة.

- **متى يتم استدعاء الباني و الهادم :**

يتم استدعاء التوابع البناءة والمدمرة بشكل أتوماتيكي، ويجري عادة استدعاء المدمرات حسب الترتيب العكسي لعمليات استدعاء التوابع البناءة.

يتم استدعاء التوابع البناءة من أجل الأغراض المصرح عنها ضمن مجال الرؤية العام وذلك قبل تنفيذ أية عملية استدعاء لتابع آخر (بما في ذلك التابع الرئيسي)، ويجري استدعاء التوابع المدمرة لها لحظة انتهاء التابع main من عمله أو عند استدعاء التابع exit.

يتم استدعاء التوابع البناءة من أجل الأغراض المحلية أتوماتيكياً عندما يصل التنفيذ إلى النقطة التي تتضمن التصريح عنها، أما التوابع المدمرة فتستدعى عندما يترك البرنامج مجال الرؤية الخاص بتلك الأغراض(أي الكتلة التي تم ضمنها التصريح عنها).