المحاضرة الرابعة

البيانات والدوال الساكنة والمؤشر This

Static Data and Functions, This Pointer

البيانات الساكنة Static Data:-

هي عبارة عن خصائص(متغيرات) مشتركة بين كل كائنات الصنف أي أن كل الكائنات لها نسخة واحدة من هذه الخصائص وللإعلان عن المتغيرات الساكنة نستخدم الكلمة المحجوزة **static** وفقاً للصيغة التالية :-

static datatype var_name;

مثال:-

```
#include<iostream.h>
class staticproperty
{
public:
   static int n;
   staticproperty( )
   {
        ++n;
   }
   ~staticproperty( )
   {
        --n;

   }
};
int staticproperty::n=0;
int main( )
{
   staticproperty s1,s2,s3;
   cout<<staticproperty::n<<"\n";
   cout<<s1.n<<"\n";
   cout<<s3.n<<"\n";
    return 0;
}
```

1. المتغير **n** هو خاصية أو متغير مشترك للكائنات لأنه استاتيكي **static** فقيمة هذا المتغير ثابتة عند كل الكائنات
2. لابد للمتغير **n** من النوع **int** وهو ساكن أن يأخذ قيمة ابتدائية

```
int    staticproperty::n  = 0;
```

3. يمكن الوصول إلى المتغير **n** باستخدام مؤثر دقة الوصول

staticproperty::n

أو باستخدام الكائنات لأنه من النوع **public**

مثلاً **s1.n**

ملاحظة مهمة:-

☑ يجب أن تكون الدوال التي تصل إلى متغير ساكن من النوع **static** أيضا مثلاً في المثال السابق:-

لو أضفنا داخل الصنف الدالة التالية:-

```
static void display(    )
{
  cout<<n;
}
```

لاحظ:-

طالما أن المتغير **n** المستخدم داخل الدالة ساكن لذلك لابد من أن تكون الدالة **display ( )** من النوع **static**

استدعاء الدالة الساكنة:-

يتم استدعاء الدالة الساكنة بطريقتين

1. باستخدام مؤثر دقة الوصول

```
void staticproperty::display( );
```

2. عن طريق الكائن

مثلاً **s1.display( );**

وألان نكتب البرنامج بصورة متكاملة باستخدام الدالة الساكنة:-

```
#include<iostream.h>
class staticproperty
{
public:

   static int n;
   staticproperty(   )
   {
        ++n;
   }
   ~staticproperty(  )
   {
        --n;
   }
static void display(  )
{
   cout<<n;
}
};
int staticproperty::n=0;
int main( )
{
   staticproperty s1,s2,s3;
   void staticproperty::display(  );
   cout<<"\n";
   s1.display(  );
   cout<<"\n";
       s3.display(  );
   cout<<"\n";
return 0;
}
```

مثال آخر عن الدالة الساكنة:-

```
#include <iostream.h>
class sample
{
   public:
        static int    n;
        sample (  )
        {
        n++;
        }
      ~sample (  )
```

```
        {
          n--;
        }
};
int sample::n=2;
int main ( )
{
    sample   a;
    sample   b[4];
      sample * c = new sample[2];
      cout << c->n <<"\n";
      delete[ ]  c;
      cout << sample::n <<"\n" ;
return 0;
  }
```

المؤشر This

يستخدم المؤشر **This** داخل الصنف للدلالة على أن الخاصية التي يتم التعامل معها ألان تنتمي للكائن الذي استدعى هذه الخاصية أو الدالة التي تتعامل مع هذه الخاصية

ولكن الاستخدام الأساسي للمؤشر**This** في التحميل الزائد للمؤثرات لإرجاع كائن من دالة التحميل الزائد (إحدى طرق إرجاع كائن)

مثال:-

```
#include<iostream.h>
class thispointer
{
private:
   int z;
public:
   thispointer(  )
   {
        this->z=0;
   }
   thispointer(int x)
   {
        this->z=x;
   }
   void display(  )
   {
        cout<<this->z<<"\n";
   }
};
```

```
int main( )
{
   thispointer t1,t2(20);
   t1.display( );
   t2.display( );
return 0;
}
```

☑	استخدمنا المؤشر **This** فقط للتأكيد على أن الخاصية تنتمي للكائن المعني ولكن وجود المؤشر غير ضروري لان البرنامج يتعرف تلقائياً على الكائنات والمتغيرات التابعة له

مثال:-

برنامج لإنشاء صنف النقطة **Point class**

خصائص الصنف:-

1. قيم الإحداثيات **x** و **y**
2. اللون **color**

الدوال(العمليات):-

1. دوال البناء
   - point ( )
   - point (int ,int)
   - point (int,int,int)
2. دالة تغيير اللون **setcolor(int)**
3. دالة تغيير الإحداثي **x** وهي **setx(int)**
4. دالة تغيير الإحداثي **y** وهي **sety(int)**
5. دالة تغيير اللون **setcolor(int)**
6. دالة الحصول على الإحداثي **x** وهي **getx( )**
7. دالة الحصول على الإحداثي **y** وهي **gety( )**
8. دالة الحصول على اللون **getcolor( )**
9. دالة عرض الإحداثيات واللون **display( )**

نص البرنامج :-

```
#include<iostream.h>
class point
{
private:
   int x,y,color;
public:
   point( )
   {
        x=0;
        y=0;
        color=0;
   }
   point(int,int);
   point(int,int,int);
   void setx(int);
   void sety(int);
   void  setcolor(int);
   int getx( );
   int gety( );
   int getcolor( );
   void display( );
};
point::point(int x1,int y1)
{
   this->x=x1;
   this->y=y1;
}
point::point(int x1,int y1,int c)
{
   this->x=x1;
   this->y=y1;
   this->color=c;
}
void point::setx(int x1)
{

   this->x=x1;
}
void point::sety(int y1)
{

   this->y=y1;
}
```

```
void point::setcolor(int c)
{

   this->color=c;
}
int point::getx( )
{
   return this->x;
}
int point::gety( )
{
   return this->y;
}
int point::getcolor( )
{
   return this->color;
}
void point::display( )
{
   cout<<"("<<x<<","<<y<<","<<color<<")"<<"\n";
}
int main( )
{
   point p1,p2(10,20),p3(50,100,0);
   p1.display( );
   p2.display( );
   p3.display( );
   p1=p2;
   p1.display( );
   p2.setx(100);
   p2.sety(100);
   cout<<"p2.x="<<p2.getx( )<<"\n";
   cout<<"p2.y="<<p2.gety( )<<"\n";
   point p4(p2.getx( ),p3.gety( ),p2.getcolor( ));
   p4.display( ) ;
return 0;
}
```

☑	يمكن نسخ كائن إلى كائن آخر من نفس النوع مثل **p1=p3**