المحاضرة السادسة

الدوال الخطية و أمثلة على الدوال الصديقة والخطية

الدوال الخطية:Inline Functions

كما نعلم أن لغة ++C تسمح باستخدام الدالة الخطية **Inline Function** والتي تعمل بحد ذاتها على تقليل وقت تنفيذ البرنامج وبالتالي هذا يؤثر على سرعة التنفيذ حيث كما تعلمنا في ++C بان استخدام **Inline** لا يتطلب أماكن جديدة في الذاكرة مما يقلل من وقت الترجمة (استهلاك وقت الحاسب)

ونذكر بان استخدام Inline في ++C كما في البرنامج التالي:-

```
#include<iostream.h>
inline int cube(int a)
{
return a*a*a;
}
int main( )
{
int i;
for(i=1;i<=5;i++)
cout<<cube(i)<<"\n";
return 0;
}
```

بنفس الأسلوب يمكن استخدام هذا المفهوم في الأصناف حيث:

1. أن أي داله يتم الإعلان عنها داخل الصنف يتم التعامل معها من قبل المترجم على أنها خطية ماعدا الدوال الكبيرة فيعتمدها المترجم على أنها عادية.
2. الدوال الخطية هي عبارة عن دوال بحجم صغير تساعد في تسريع تنفيذ البرنامج
3. يقوم المترجم بكتابة تفاصيل الدالة في كل موضع يظهر فيه استدعاء لهذه الدالة مما يزيد من سرعة تنفيذ البرنامج وذلك بتقليل زمن الذهاب والرجوع عند استدعاء الدالة
4. كل داله تكتب تفاصيلها داخل الصنف يتم اعتمادها من قبل المترجم كداله خطية ما لم تكن الدالة تحوي أوامر كثيرة أو حلقات تكرارية

مثال:-

```
#include<iostream.h>
class  xy
 {
    private:
         int   g;
    public :
            void set (int x)
              {
                g = x ;
              }
 };
```

لاحظ:-

الدالة **set** تعتبر خطية لسببين:

- تم الإعلان عنها داخل الصنف
- أن تعليماتها قليلة وليس بها حلقات تكرارية

☑	إذا كتبت تفاصيل الدالة خارج الصنف لابد من إضافة الكلمة المحجوزة **inline** إلى الدالة حتى تعتبر داله خطية .

```
#include<iostream.h>
class  xy
 {
    private:
         int   g;
     public :
         void set (int);
 };
inline void   xy :: set(int x)
{
    g = x ;
}
```

مثال:-

```
#include<iostream.h>
class samp
{
int i,j,k ;
public:
samp (int a, int b)
{
i = a ;
j = b ;
}
int divisible (  );
};
inline int samp :: divisible (  )
{
k = ! ( i % j ) ;
return k ;
}
int main(  )
{
int x,y;
cout<<"enter x and y \n";
cin>>x>>y;
samp  s(x,y);
if (s. divisible (  ))
    cout <<x<<"divided by"<<y<<"\n" ;
else
    cout <<x<<" is not divided by"<<y<<"\n" ;
return 0;
}
```

أمثلة محلولة

1. اكتب برنامج لتعريف واستخدام صنف به عدد يحسب مربع العدد والجزر التربيعي له ومكعبه

```
#include<iostream.h>
#include<math.h>
class number
{
int n;
public:
number(  )
{
cout<<"enter anumber \n";
cin>>n;
}
```

```
int square( )
{
return  n*n;
}
float square_root( )
{
return  sqrt(n);
}
int cube( )
{
return  n*n*n;
}
};
int main( )
{
number num;
cout<<"square="<<num.square( )<<"\n";
cout<<"square root="<<num.square_root( )<<"\n";
cout<<"cube ="<<num.cube( )<<"\n";
return 0;
}
```

2. أنشئ صنف يقوم بإيجاد حاصل جمع المتوالية العددية

a+(a+d)+(a+2d)+……+(a+(n-1)d)

حيث:

a الحد الأول و **d** الفرق الثابت و **n** عدد الحدود

```
#include<iostream.h>
class nseries
{
private:
int a,d,n,s;
public:
nseries( )
{
s=0;
}
void set_n( )
{
cout<<"enter elements number \n";
cin>>n;
cout<<"enter  first element\n";
cin>>a;
cout<<"enter  the differance\n";
```

```
cin>>d;
}
void sum( )
{
for(int i=1;i<=n;i++)
s+=a+(i-1)*d;
}
void display( )
{
for(int i=1;i<=n;i++)
cout<<(a+(i-1)*d)<<",";
cout<<"\n";
cout<<"sum ="<<s<<"\n";
}
};
int main( )
{
nseries  ns;
ns.set_n( );
ns.sum( );
ns.display( );
return 0;
}
```

3. باستخدام الأصناف اكتب برنامج لإيجاد صافي الراتب **Net salary** لموظف حيث العمولة تمثل **10%** من الراتب الأساسي **Basic salary** والضريبة **Tax** تمثل **4%** من الراتب الأساسي

```
#include<iostream.h>
class employee
{
public:
float b,n,c,t;
employee( )
{
cout<<"enter the basic salary \n";
cin>>b;
}
void commission( )
{
c=b*10/100;
}
void tax( )
{
t=b*4/100;
}
```

```
void display( )
{
cout<<"commission ="<<c<<"\n";
cout<<"tax ="<<t<<"\n";
}
};
int main( )
{
employee  emp;
emp.commission( );
emp.tax( );
emp.display( );
emp.n=emp.b+emp.c-emp.t;
cout<<"net salaty ="<<emp.n<<"\n";
return 0;
}
```

4. باستخدام الأصناف اكتب برنامج لإيجاد التوافيق وفقاً للقانون التالي:

$$m = {}^{n}C_{r} = \frac{n!}{(n-r)!\ r!}$$

حيث $n>r>=0$

```
#include<iostream.h>
class computation
{
int i;
public:
int n,r,w,f;
computation( )
{
f=1;
}
int factorial(int n)
{
int i;
for(i=1;i<=n;i++)
f=f*i;
return f ;
}
};
int main( )
{
computation  c,c1,c2,c3;
int i,m;
long n1,r1,w1;
```

```
cout<<"enter n \n";
cin>>c.n;
n1=c1.factorial(c.n);
cout<<"n!="<<n1<<"\n";
cout<<"enter r \n";
cin>>c.r;
r1=c2.factorial(c.r);
cout<<"r!="<<r1<<"\n";
c.w=c.n-c.r;
if(c.w>=0)
{
w1=c3.factorial(c.w);
cout<<"(n-r)!="<<w1<<"\n";
m=n1/(r1*w1);
cout<<"m="<<m<<"\n";
}
else
cout<<"factorial cant be found because the differance (n-r) is negative \n";
return 0;
}
```