الدرس الثامن: التصنيفات

Classes

1- لغة C++ والبرمجة الكائنية

تدعم لغة C++ المقومات الأساسية للبرمجة الكائنية أو الشيئية : التغليف وإخفاء البيانات، الوراثة وإعادة الاستعمال، تعددية الأشكال، ... الخ.

- **التغليف و إخفاء البيانات (Encapsulation)**

تدعم C++ تقنية التغليف و إخفاء البيانات عبر إنشاء أنواع جديدة تسمى بالـ تصنيفات (classes) ومن ثم يعمل هذا الصنف ككائن مغلف تماما و يستعمل كوحدة متكاملة بحيث يبقى عمله الداخلي مخفيا و لا يحتاج المستخدم إلى فهم هذا العمل المخبأ، ينبغي أن يعرف كيفية استعماله فقط.

- **الوراثة و إعادة الاستعمال (Inheritance and Reusability)**

توفر C++ دعما فعالا لإعادة الاستعمال من خلال الوراثة (inheritance) بحيث يمكن إعلان نموذج جديد يكون امتدادا لصنف موجود. ويسمى هذا الصنف الجديد صنفا مشتقا.

- **التعددية الشكلية (Polymorphism)**

إن دالة الرسم واحدة، و لكن أطوارها متعددة بحسب الشكل المطلوب، وهذا هو أحد المبادئ الاقتصادية في لغة C++. لتوفير الوقت و الجهد، يمكن استخدام نفس الدالة لتحقيق مهام مختلفة.

2- تعريف التصنيفات (الفئات)

التصنيف (class) هو وصف نظري (Abstract Data Type) لمجموعة متماثلة من الأشياء التي تسمى كائنات أو كينونات (objects).

يحتوي التصنيف على مجموعة من البيانات (data members) التي تمثل الخصائص ومجموعة من الدوال (function members) التي تقوم بمعالجة هذه البيانات.

يتم تعريف التصنيفات في أغلب الأحيان باستخدام قاعدة البناء التالية:

```
class Class_Name
{
      private:
            type1  data1;
            type2  data 2;
            ...
      public:
            function1();
            function2();
            ...
};
```

حيث Class_Name هو اسم التصنيف وبداخل التصنيف يوجد قسمان:

- **قسم خاص** (private) لتعريف البيانات (Data) أو الخصائص (Properties)
- **قسم عام** (public) لتعريف الدوال (Functions) أو الطرق (Methods).

بعد تعريف التصنيف يمكننا تعريف كينونات (objects) من نوع هذا التصنيف لاستخدامها في البرنامج حسب الحاجة وذالك بإستخدام الصيغة التالية:

```
Class_Name  Object_Name;
```

وتعتبر الكينونة تمثيل للتصنيف خلال وأثناء تنفيذ البرنامج.

مثال

```
Class CRectangle        // تعريف صنف المستطيل
{
      private:
            float Length;           //الطول
            float Width;            // العرض
      public:
            void SetValues(float x, float y)    // تحديد الطول والعرض
            {
                  Length = x;
                  Width = y;
            }
            float Area(void);       // حساب المساحة
            float Perimeter(void);  // حساب المحيط
};

float CRectangle::Area(void)
{
      return  Length*Width;
}

float CRectanle::Perimeter(void)
{
      return 2*(Length+Width);
}
```

وقع تعريف الدالة SetValues داخل التصنيف. أما بقية الدوال فقد وقع تعريفها خارج التصنيف باستخدام معامل المجال (::).

لتعريف متغيرات (كينونات) من فئة التصنيف CRectangle يمكن أن نستخدم الصيغة التالية:

CRectangle rect1, rect2;

ملاحظة

الفرق بين السجل والتصنيف هو أن التصنيف يحتوي على مجموعة من البيانات والدوال بينما لا يحتوي السجل عادة إلا على مجموعة من البيانات.

3- الوصول إلى عناصر الكينونات

يمكن الوصول إلى العناصر العمومية للكينونات باستخدام معامل النقطة (.) كما في المثال التالي:

```
rect1.SetValues(4,3);
float  a = rect1.Area();            // a = 12
float  p = rect1.Perimeter();       // p = 14
```

تمرين رقم 1

قم بتعريف تصنيف الدوائر CCircle الذي يحتوي على بيان واحد هو نصف القطر (Radius) وثلاث دوال:

- SetRadius() لتحديد نصف قطر دائرة معينة
- Area() لحساب مساحة دائرة معينة
- Perimeter() لحساب محيط دائرة معينة.

- **الحل**

```
#include <iostream>
#include <cmath>
using namespace std;
const float PI = 3.14;

class  CCircle
{
      private:
            float  Radius;
      public:
            void  SetRadius(float);
            float  Area(void);
            float  Perimeter(void);
};
void  CCircle::SetRadius(float  x)
{
      Radius = x;
}
float  CCircle::Area(void)
{
      return  PI*pow(Radius,2);
}
float  CCircle::Perimeter(void)
{
      return  2*PI*Radius;
}
int main( )
{
      CCircle  c;
      int r;
      cout<<"Enter the radius of the circle in cm: ";  cin>>r;
      c.SetRadius(r);
      cout<<"Area = "<<c.Area( )<<" cm2"<<endl;
      cout<<"Perimeter = "<<c.Perimeter( )<<" cm"<<endl;
      return 0;
}
```

- **تتبع تنفيذ البرنامج**

```
Enter the radius of the circle in cm: 10
Area = 314 cm2
Perimeter = 62.8 cm

Press any key to continue
```

تمرين رقم 2: قم بإصلاح الأخطاء الواردة في البرنامج التالي:

```
#include <iostream>

using namespace std;

class Cstudent
{
      private:
            int Id;                 // الرقم الجامعي
            char Name[20];          // اسم الطالب
            int Age;                // العمر
            char Address[30];       // العنوان

      Public
            void SetData(int, char[],int, char[]);
            void ShowData(void);
}

void Cstudent::SetData(int n, char name[],int age, char adr)
{
      Id = n;
      strcpy(Name,name);
      Age = age;
      Address = adr;
}

void Cstudent::ShowData(void)
{
      cout<<"\t Student No "<<Id<<endl;
      cout<<"\t\t Name = "<<Name<<endl;
      cout<<"\t\t Age = "<<Age<<endl;
      cout<<"\t\t Address = "<<Address<<endl;
      cout<<endl;
}

int main( )
{
      Cstudent stud1, stud2;
      stud1.SetData(1001,"Mohamed Rached", 23, "Abha");
      stud2.SetData(1002,"Ahmed Salem", 21, "Riyadh");
      stud1.ShowData;
      cout<<endl;
      stud2.ShowData;
      return 0;
}
```

المخرجات المنتظرة على الشاشة

```
Student No 1001
        Name = Mohamed Rached
        Age = 23
        Address = Abha


Student No 1002
        Name = Ahmed Salem
        Age = 21
        Address = Riyadh

Press any key to continue . . .
```

تمرين رقم 3

قم بتعريف تصنيف الموظفين (Employees) الذي يحتوي على البيانات التالية:

- الرقم الوظيفي (Number)
- الاسم (Name)
- العنوان (Address)
- الراتب الأساسي (BasicSalary)
- عدد سنوات الخبرة (Seniority)

تتم معالجة هذه البيانات من خلال الوظائف التالية :

- SetData( ) التي تقوم بتحديد بيانات موظف معين
- ComputeSalary( ) التي تقوم بحساب الراتب الاجمالي لأحد الموظفين طبقا للصيغة التالية:

الراتب الإجمالي = الراتب الأساسي + 350 ريال عن كل سنة خبرة

- ShowData( ) التي تقوم بطباعة بيانات أحد الموظفين (بما في ذلك الراتب الإجمالي) على الشاشة.

- **الحل**

```cpp
#include <iostream>

using namespace std;

class CEmployee
{
      private:
            int number;
            char name[20];
            char address[30];
            float BasicSalary;
            int seniority;

      public:
            void SetData(int, char[ ], char[ ], float, int);
            float ComputeSalary (void);
            void ShowData(void);
};
```

```
void CEmployee::SetData(int n, char nameE[ ], char adrE[ ], float
BasSal, int senior)
{
      number = n;
      strcpy(name,nameE);
      strcpy(address,adrE);
      BasicSalary = BasSal;
      seniority = senior;
}

float CEmployee:: ComputeSalary(void)
{
      return BasicSalary + 350 * seniority;
}

void CEmployee::ShowData(void)
{
      cout<<"Employee No "<<number<<endl;
      cout<<"\tName = "<<name<<endl;
      cout<<"\tAddress = "<<address<<endl;
      cout<<"\tBasic Salary = "<<BasicSalary<<" SAR"<<endl;
      cout<<"\tSenority = "<<seniority<<" years"<<endl;
      cout<<"\tTotal Salary = "<<ComputeSalary( )<<" SAR"<<endl;
}

int main( )
{
      CEmployee Emp;
      Emp.SetData(2525,"Sultan","Riyadh",3000,5);
      Emp.ShowData( );
      cout<<endl;
      return 0;
}
```

- **مخرجات البرنامج**

```
Employee No 2525
        Name = Sultan
        Address = Riyadh
        Basic Salary = 3000 SAR
        Senority = 5 years
        Total Salary = 4750 SAR

Press any key to continue . . .
```

4- دوال البناء والهدم (Constructor & Destructor)

تستخدم دوال البناء أو التشييد لإنشاء الكائنات (objects) وإعطائها قيما أولية. إذا لم يقم المبرمج بكتابة هذه الدالة فإن معالج C++ سيقوم بإضافتها آليا.

يجب أن تحمل دالة البناء نفس اسم الصنف الذي تنتمي إليه.

أما دالة الهدم (destructor) التي تضع نهاية لوجود كائن معين فتحمل نفس اسم الصنف هي الأخرى ويكون مسبوقا بالعلامة "~".

مثال

```
#include <iostream>
using namespace std;
class Test
{
      public:
            Test( )           // Constructor
            {
                  cout<<"\tActivation of the constructor"<<endl;
            }
            ~ Test( )         // Destructor
            {
                  cout<<"\tActivation of the destructor"<<endl;
            }
};
int main()
{
      cout<<"Starting of main( )"<<endl;
      Test t1, t2;
      cout<<"Ending of main( )"<<endl;
      return 0;
}
```

مخرجات البرنامج

```
Starting of main( )
        Activation of the constructor
        Activation of the constructor

Ending of main( )
        Activation of the destructor
        Activation of the destructor

Press any key to continue . . .
```

في البرنامج السابق تم في البداية استدعاء دالة البناء مرتين لإنشاء الكائنين t1 و t2، كما وقع استدعاء دالة الهدم مرتين في نهاية البرنامج لوضح حد لوجود هذين الكائنين.

تمرين

استخدم تقنية البرمجة الشيئية (الكينونية) لكتابة برنامج يمكن شركة Info++ من مراقبة المخزون الذي يحتوي على لوحات مفاتيح (Keybords) ووحدات معالجة (CPU) وشاشات (Screens).

يحتوي الصنف stock على البيانات التالية :

- عدد لوحات المفاتيح (NKeyb)
- عدد وحدات المعالجة (NCPU)
- عدد الشاشات (NScreen).

تتم معالجة هذه البيانات من خلال الوظائف التالية :

- InitStock( ) التي يتم تنفيذها كل صباح لتمكين المستخدم من إدخال الكمية المتوفرة من مختلف التجهيزات.
- SoldKeyb(n) التي تقوم بتسجيل عملية بيع عدد من لوحات المفاتيح و تحديث المخزون
- SoldCPU(n) التي تقوم بتسجيل عملية بيع عدد من وحدات المعالجة و تحديث المخزون
- SoldScreen(n) التي تقوم بتسجيل عملية بيع عدد من الشاشات وتحديث المخزون

- SoldPC(n) التي تقوم بتسجيل عملية بيع عدد من الحاسبات (لوحة مفاتيح + وحدة معالجة + شاشة) وتحديث المخزون
- ShowStock( ) التي تقوم بطباعة المخزون المتوفر من مختلف التجهيزات على الشاشة.

- **الحل**

```
#include <iostream>
using namespace std;

class stock
{
      private:
            int NKeyb;
            int NCPU;
            int Nscreen;
      public:
            void InitStock(void);
            void SoldKeyb(int);
            void SoldCPU(int);
            void SoldScreen(int);
            void SoldPC(int);
            void ShowStock(void);
};

void stock::InitStock(void)
{
      cout<<"Enter the number of keyboards : ";
      cin>>NKeyb;
      cout<<"Enter the number of CPU : ";
      cin>>NCPU;
      cout<<"Enter the number of screens : ";
      cin>>Nscreen;
}

void  stock::SoldKeyb(int  n)
{
      cout<<"A sale of "<<n<<" keyboards..."<<endl;
      NKeyb = NKeyb - n;
}

void  stock::SoldCPU(int  n)
{
      cout<<"A sale of "<<n<<" CPU..."<<endl;
      NCPU = NCPU - n;
}

void  stock::SoldScreen(int  n)
{
      cout<<"A sale of "<<n<<" screens..."<<endl;
      Nscreen = Nscreen - n;
}

void  stock::SoldPC(int  n)
{
      cout<<"A sale of "<<n<<" PC..."<<endl;
      SoldKeyb(n);      // NKeyb = NKeyb - n;
```

```
      SoldCPU(n);       // NCPU = NCPU - n;
      SoldScreen(n);    // Nscreen = Nscreen - n;
}

void stock::ShowStock(void)
{
      cout<<"\nACTUAL STOCK\n";
      cout<<"------ -----\n";
      cout<<"Number of keyboards = "<<NKeyb<<endl;
      cout<<"Number of CPU = "<<NCPU<<endl;
      cout<<"Number of screens = "<<Nscreen<<endl;
}

int main( )
{
      stock  st;
      st.InitStock( );
      st.SoldKeyb(5);
      st.SoldScreen(3);
      st.SoldPC(2);
      st.ShowStock( );
      cout<<endl;
      return 0;
}
```

- **تتبع تنفيذ البرنامج**

```
Enter the number of keyboards : 10
Enter the number of CPU : 10
Enter the number of screens : 10

A sale of 5 keyboards...
A sale of 3 screens...
A sale of 2 PC...
A sale of 2 keyboards...
A sale of 2 CPU...
A sale of 2 screens...

ACTUAL STOCK
------ -----
Number of keyboards = 3
Number of CPU = 8
Number of screens = 5

Press any key to continue . . .
```

5- تغليف و إخفاء البيانات Encapsulation

في البرمجة الشيئية (Object Oriented Programming) لا يمكن الوصول للأعضاء الخاصة للكائنات (Private Members) إلا من خلال الدوال العمومية (Public Functions) التي تكوّن ما يعرف بالواجهة (Interface).

مثال

```
#include <iostream>
using namespace std;
class CStudent
{
      private:
            int  number;
            char  name[20];
```

```
      public:
            void  ShowData(void )
            {
                  cout<<number<<"\t"<<name;
                  cout<<endl;
            }
};

int main( )
{
      CStudent  st;
      cout<<"Enter the student number";
      cin>>st.number;
      cout<<"Enter the student name";
      cin>>st.name;
      st.ShowData( );
      return 0;
}
```

عند محاولة تنفيذ هذا البرنامج يعلن ا لمترجم عن وجود أخطاء تتعلق بمحاولة الوصول إلى البيانات الخاصة داخل الصنف CStudent وهو ما يتنافى مع مبدأ التغليف أو الإخفاء (Encapsulation).

```
error C2248: 'CStudent::number' : cannot access private member declared
in class 'CStudent'

error C2248: 'CStudent::name' : cannot access private member declared
in class 'CStudent'
```

لتصحيح البرنامج يمكن اللجوء إلى أحد الحلول التالية:

- **الحل الأول :** جعل بيانات الطلاب عامة (public) وليس خاصة:

```
class CStudent
{
      public:
            int  number;
            char  name[20];

            void  ShowData(void )
            {
                  cout<<number<<"\t"<<name;
                  cout<<endl;
            }
};
```

هذا الحل يحتوي على مخاطر لأنه يتيح إمكانية الإطلاع وتعديل البيانات لجميع المبرمجين وهو ما يتناقض مع مبدأ السلامة.

- **الحل الثاني :** إضافة دوال عمومية للإطلاع على البيانات (Getters) وأخرى لتعديلها (Setters) وذلك فقط من طرف الأشخاص المصرح لهم:

```
# include <iostream>
using namespace std;

class CStudent
{
      private:
            int  number;
            char  name[20];
      public:
            int GetNumber(void)
            {
                  return number;
            }
            void SetNumber(int x)
            {
                  number = x;
            }
            char* GetName(void)
            {
                  return name;
            }
            void SetName(char Name[])
            {
                  strcpy(name,Name);
            }
};

int main( )
{
      CStudent  st;
      int StudNo;
      char StudName[20];
      cout<<"Enter the student number : ";
      cin>>StudNo;
      st.SetNumber(StudNo);
      cout<<"Enter the student name : ";
      cin>>StudName;
      st.SetName(StudName);
      cout<<endl;
      cout<<"----------------------------------------"<<endl;
      cout<<"Number="<<st.GetNumber()<<"\tName="<<st.GetName()<<endl;
      cout<<"----------------------------------------"<<endl;
      cout<<endl;
      return 0;
}
```

تتبع تنفيذ البرنامج

```
Enter the student number : 427320100
Enter the student name : Mohamed

----------------------------------------
Number = 427320100      Name = Mohamed
----------------------------------------

Press any key to continue . . .
```

تمرين

استخدم تقنية البرمجة الشيئية (الكينونية) لكتابة برنامج يمكن شركة البنك من إدارة حسابات العملاء.

يحتوي الصنف Account على البيانات التالية :

- رقم الحساب (Number)
- إسم صاحب الحساب (Name)
- الرصيد (Balance).

تتم معالجة هذه البيانات من خلال الوظائف التالية :

- دالة بناء : Account(int number, char name[], float balance)
- الاطلاع على الرصيد : CheckBalance( )
- إيداع المبلغ m في الحساب الحالي : Deposit(float m)
- سحب المبلغ m من الرصيد الحالي (إذا كان الرصيد يسمح بذلك) : Withdraw(float m)
- تحويل المبلغ m من الحساب الحالي إلى الحساب X (إذا كان الرصيد يسمح بذلك) : Transfer(float m, Account &X)

- **البرنامج**

```
#include <iostream>
using namespace std;
class  Account
{
      private:
            int  Number;
            char  Name[20];
            float  Balance;
      public:
            Account(int number, char name[], float balance)
            {
                  Number = number;
                  strcpy(Name, name);
                  Balance = balance;
            }

            void CheckBalance(void)
            {
                  cout<<"Balance = "<<Balance<<" Riyals";
                  cout<<endl;
            }
            void Deposit(float  m)
            {
                  Balance = Balance + m;
            }
            void  Withdraw(float  m)
            {
                  if (Balance >= m)
                        Balance = Balance - m;
                  else
                        cout<<"insufficient balance..."<<endl;
            }
```

```
            void  Transfer(float  m, Account  &X)
            {
                  if (Balance >= m)
                  {
                        Balance= Balance - m;
                        X.Deposit(m);
                  }
                  else
                        cout<<"insufficient balance..."<<endl;
            }
};
int main( )
{
      Account  A(1001, "Ali", 5000);
      Account B(2002, "Ahmed", 2000);
      A.Withdraw(1000);
      B.Deposit(3000);
      A.Transfer(500, B);
      cout<<"Account A, "; A.CheckBalance( );
      cout<<"Account B, "; B. CheckBalance( );
      cout<<endl;
      return 0;
}
```

- **مخرجات البرنامج**

```
Account A, Balance = 3500 Riyals
Account B, Balance = 5500 Riyals
Press any key to continue
```

6- التحميل الزائد لدوال الصنف (Function Overloading)

تسمح لغة C++ بإنشاء أكثر من دالة في نفس الصنف تحمل كلها نفس الاسم ولكنها تختلف من حيث عدد ونوع المعاملات وهو ما يسمى بالتحميل الزائد للدوال (Function Overloading). في كل مرة يتم فيها استدعاء الدالة يقوم المترجم بتنشيط الدالة المناسبة حسب عدد ونوع المعاملات التي يقع تمريرها.

في المثال التالي سنقوم بتحميل دالة البناء (Constructor) التي تحمل نفس اسم التصنيف Cstudent والتي تنفذ آليا كلما تم إضافة طالب جديد:

```
#include <iostream>
using namespace std;

class CStudent
{
      private:
            int  number;
            char  name[20];
      public:
            CStudent ( )                        //  دالة بناء أولى
            { }

            CStudent(int  x, char  Name[])      // دالة بناء ثانية
            {
                  number = x;
                  strcpy(name,Name);
            }
```

```
            void  ShowData(void )
            {
                  cout<<"Sudent number = "<<number;
                  cout<<"\nStudent name = "<<name;
                  cout<<endl;
            }
};

int main( )
{
      CStudent  st1;                        // إستدعاء دالة البناء الأولى
      CStudent st2(427320100, "Fahd");      // إستدعاء دالة البناء الثانية
      st2.ShowData( );
      return 0;
}
```

مخرجات البرنامج

```
Sudent number = 427320100
Student name = Fahd
Press any key to continue . . .
```

7- التحميل الزائد للعوامل (Operator Overloading)

بالإضافة للدوال، تسمح لغة C++ بالتحميل الزائد للعوامل الثنائية (+ - * /) والأحادية (++ -- إلخ).

مثال

أكتب برنامجا لمعالجة الصنف Time الذي يحتوي على البيانات التالية :

- الساعات (Hours) // 0 .. 23
- الدقائق (Minutes) // 0 .. 59
- الثواني (Seconds) // 0 .. 59

يتم الإعلان عن الكائنات من فئة Time باستخدام إحدى دوال البناء (constructor) التالية:

- Time( )
- Time(int H , int M)
- Time(int H , int M, int S)

تتم معالجة هذه البيانات من خلال الوظائف التالية :

- SetTime(int H , int M, int S) : تحديد الوقت
- ShowTimeUniversal( ) : طباعة الوقت على الشكل 23:30:15
- ShowTimeStandard( ) : طباعة الوقت على الشكل 11:30:15 PM

كما يتم في هذا البرنامج التحميل الزائد (overload) للمعاملين:

+ : حساب مجموع وقتين
++ : إضافة ثانية إلى الوقت الحالي.

- **الحل**

```
#include <iostream>
using namespace std;
class Time
{
	private:
		int  Hours;    // 0..23
		int Minutes;   // 0 .. 59
		int Seconds;   // 0 .. 59
	public:
		Time()
		{ }
		Time (int  H, int M)
		{
			Hours = H;
			Minutes = M;
		}
		Time (int  H, int M, int S)
		{
			Hours = H;
			Minutes = M;
			Seconds = S;
		}
		void SetTime (int  H, int M, int S)
		{
			Hours = H;
			Minutes = M;
			Seconds = S;
		}
		void ShowTimeUniversal(void )
		{
			cout<<Hours<<":"<<Minutes<<":"<<Seconds<<endl;
		}
		void ShowTimeStandard(void )
		{
			if ((Hours == 12) || (Hours == 0))
				cout<<"12";
			else
				cout<<Hours%12;
			cout<<":"<<Minutes<<":"<<Seconds;
			if (Hours <= 12)
				cout<<" AM";
			else
				cout<<" PM";
			cout<<endl;
		}
		Time operator+(Time t)
		{
			Hours = Hours + t.Hours;
			Minutes = Minutes + t.Minutes;
			Seconds = Seconds + t.Seconds;
			if (Seconds > 59)
			{
				Minutes++;
				Seconds = Seconds - 60;
			}
```

```
                    if (Minutes > 59)
                    {
                          Hours++;
                          Minutes = Minutes - 60;
                    }
                    if (Hours > 23)
                    {
                          Hours = Hours % 24;
                    }
                    return Time(Hours,Minutes,Seconds);
              }
              Time operator++(void)
              {
                    Seconds++;
                    if (Seconds > 59)
                    {
                          Minutes++;
                          Seconds = Seconds - 60;
                    }
                    if (Minutes > 59)
                    {
                          Hours++;
                          Minutes = Minutes - 60;
                    }

                    if (Hours > 23)
                    {
                          Hours = Hours % 24;
                    }
                    return Time(Hours,Minutes,Seconds);
              }
};
int main( )
{
      Time t(23,59,59);
      cout<<"Time t in universal format =   ";
      t.ShowTimeUniversal();
      cout<<"Time t in standard format =   ";
      t.ShowTimeStandard( );

      t++;
      cout<<"After one second, t will be  = ";
      t.ShowTimeUniversal();
      cout<<"-----------------------------------------"<<endl;
      Time u,v;
      u.SetTime (2,20,22);
      cout<<"Time u in universal format =   ";
      u.ShowTimeUniversal();
      v.SetTime (3,45,33);
      cout<<"Time v in universal format =   ";
      v.ShowTimeUniversal();
      Time w = u + v;
      cout<<"Time w = u + v in universal format =   ";
      w.ShowTimeUniversal( );
      cout<<"-----------------------------------------"<<endl;
      return 0;
}
```

- **مخرجات البرنامج**

```
Time t in universal format =   23:59:59
Time t in standard format =   11:59:59 PM
After one second, t will be  = 0:0:0
------------------------------------------------
Time u in universal format =   2:20:22
Time v in universal format =   3:45:33
Time w = u + v in universal format =   6:5:55
------------------------------------------------
Press any key to continue . . .
```

8- مؤشرات الكائنات

مثل بقية أنواع البيانات، يمكن استخدام مؤشرات نحو متغيرات من فئة صنف معين (كائنات). ويتم الوصول إلى أعضاء الكائن (البيانات والوظائف) عن طريق علامة العضوية (**->**) باستخدام الصيغة التالية:

```
Pointer -> Member
```

مثال

```
#include <iostream>

using namespace std;

class person
{
      private:
            char name[20];
            int age;
      public:
            void SetData(char Name[], int Age)
            {
                  strcpy(name,Name);
                  age = Age;
            }

            void ShowData(void)
            {
                  cout<<"Name = "<<name<<"\t"<<"Age = "<<age<<" years";
                  cout<<endl;
            }
};

int main( )
{
      person pers;
      person *p = &pers;          // pers المؤشر p يشير إلى الكائن
      p->SetData("Mohammed",21); // Same as pers.SetData("Mohammed",21);
      p->ShowData( );             // Same as pers.ShowData( );
      return 0;
}
```

مخرجات البرنامج

```
Name = Mohammed          Age = 21 years
Press any key to continue . . .
```

ملاحظات

- تمكن الكلمة المحجوزة **new** من إنشاء كائن جديد بطريقة ديناميكية أثناء تنفيذ البرنامج.
- يقوم الأمر **delete** بحذف الكائن وبالتالي تحرير الذاكرة المحجوزة لهذا الكائن حتى يتم استخدامها لتخزين كائنات أخرى.
- عند إنشاء أي صنف جديد يتم بصفة آلية إنشاء المؤشر **this** الذي يشير دائما إلى الكائن الحالي.

مثال

```
#include <iostream>
using namespace std;

class person
{
      private:
            char name[20];
            int age;
      public:
            void SetData(char name[], int age)
            {
                  strcpy(this -> name,name);
                  this -> age = age;
            }
            void ShowData(void)
            {
                  cout<<"Name = "<<name<<"\t"<<"Age = "<<age<<" years";
                  cout<<endl;
            }
};

int main( )
{
      person *p;
      p = new person;    // إنشاء كائن من فئة person وتخزين عنوانه في المؤشر p
      p->SetData("Ali",22);
      p->ShowData();
      delete p;          // حذف الكائن الذي يشير إليه المؤشر p
      return 0;
}
```

مخرجات البرنامج

```
Name = Ali               Age = 22 years
Press any key to continue . . .
```

في البرنامج السابق تم استخدام المؤشر this لتجاوز مشكلة التشابه في الأسماء name و age بين أعضاء التصنيف person ومعاملات الدالة SetData.

8- مصفوفات الكائنات

تسمح لغة C++ بإنشاء مصفوفات تحتوي على كائنات من صنف معين وذلك بإتباع قاعدة البناء التالية:

```
calss Class_Name
{

};

Class_Name   Array_Name[n];      // مصفوفة تحتوي على عدد n كائن
```

مثال

قم بإنشاء مصفوفة تحتوي على 5 كائنات من صنف person.

- **الحل**

```
#include <iostream>

using namespace std;

class Person
{
      private:
            char name[20];
            int   age;
      public:
            Person(void)
            {}
            void SetData(char name[ ],  int  age)
            {
                  strcpy(this -> name, name);
                  this -> age = age;
            }
            void ShowData(void)
            {
                  cout<<name<<"\t\t"<<age;
                  cout<<endl;
            }
};
int main( )
{
      char  Name[20];
      int i;
      int Age;
      Person A[5];      // إنشاء المصفوفة
      for(i = 0; i <= 4; i++)
      {
            cout<<"Ente the name : ";
            cin>>Name;
            cout<<"Enter the age : ";
            cin>>Age;
```

```
            A[i].SetData(Name, Age); // i تحديد بيانات الكيان رقم
            cout<<"-------------------------"<<endl;
      }
      cout<<"NAME\t\tAGE"<<endl;
      cout<<"----\t\t---"<<endl;
      for(i = 0; i <= 4; i++)
      {
                  A[i].ShowData( );
      }
      cout<<"-------------------------"<<endl;
      return 0;
}
```

- **تتبع تنفيذ البرنامج**

```
Enter the name : Ali
Enter the age : 22
-------------------------
Enter the name : Akram
Enter the age : 20
-------------------------
Enter the name : Majed
Enter the age : 25
-------------------------
Enter the name : Rached
Enter the age : 23
-------------------------
Enter the name : Ahmed
Enter the age : 20
-------------------------
NAME            AGE
----            ---
Ali             22
Akram           20
Majed           25
Rached          23
Ahmed           20
-------------------------

Press any key to continue . . .
```

9- الدوال الصديقة Friend functions

من حيث المبدأ، لا يمكن الوصول إلى الأعضاء الخاصة (private) والمحمية (protected) لصنف معين إلا من خلال الدوال العمومية التابعة لهذا الصنف. هذه القاعدة لا تنطبق على الدوال الصديقة.

إذا تم التصريح بدالة خارجية كصديقة للصنف، فهذا يعني السماح لهذه الدالة بالوصول للأعضاء الخاصة والمحمية للصنف. يتم ذلك بالإعلان عن توقيع الدالة داخل الصنف مسبوقا بالكلمة المحجوزة **friend** .

مثال

```
// friend functions
#include <iostream>
using namespace std;

class CRectangle
{
  friend CRectangle duplicate (CRectangle);

  private:
      int length, width;      //  الطول والعرض بحساب السنتمتر
```

```
public:
      void set_values (int a, int b)
      {
            width = a;
            length = b;
      }
      int area ()
      {
            return (width * length);
      }
 };

CRectangle duplicate (CRectangle rectparam)
{
  CRectangle rectres;
  rectres.width = rectparam.width*2;
  rectres.length = rectparam.length*2;
  return rectres;
}

int main ()
{
  CRectangle rect1, rect2;
  rect1.set_values (2,3);
  rect2 = duplicate (rect1);
  cout <<"Area of rect2 = "<<rect2.area()<<" cm2";
  cout<<endl<<endl;
  return 0;
}
```

مخرجات البرنامج

```
Area of rect2 = 24 cm2
Press any key to continue . . .
```

تقوم الدالة duplicate بإرجاع مستطيل جديد تكون أبعاده ضعف أبعاد المستطيل الذي يتم تمريره كوسيط.

رغم أن هذه الدالة ليست من أعضاء الصنف CRectangle، فقد تمكّنت من الوصول إلى الأعضاء الخصوصية length و width باعتبارها صديقة لهذا الصنف.

10- الأصناف الصديقة Friend classes

إضافة إلى الدوال الصديقة friend functions، تتيح البرمجة الشيئية إمكانية الإعلان عن أصناف صديقة لصنف معين، بحيث يمكن لكل دوال هذا الصنف الصديق الوصول إلى البيانات الخاصة والمحمية.

مثال

```
// friend classes
#include <iostream>

using namespace std;
```

```
class CSquare
{
  friend class CRectangle;
  private:
      int side;                // الضلع بحساب السنتمتر
  public:
      void set_side (int a)
      {
            side=a;
      }
};

class CRectangle
{
  private:
     int width, height;        // الطول والعرض بحساب السنتمتر
  public:
      int area ()
      {
            return (width * height);
      }
      void convert (CSquare a)
      {
            width = a.side;
            height = a.side;
      }
};

int main ()
{
      CSquare sqr;
      sqr.set_side(4);
      CRectangle rect;
      rect.convert(sqr);
      cout <<"Area of rect = "<<rect.area()<<" cm2"<<endl;
      return 0;
}
```

مخرجات البرنامج (Output)

```
Area of rect = 16
Press any key to continue . . .
```

تقوم الدالة convert بتحويل مربع معين إلى مستطيل بنفس الطول والعرض.

في هذا البرنامج، تم الإعلان عن الصنف مستطيل rectangle كصديق للصنف مربع square فأصبح بإمكان دوال الصنف rectangle وتحديدا الدالة convert الوصول إلى بيانات الصنف square وتحديدا قيمة الضلع (square::side).

هام جدا: الصنف rectangle لا يعتبر الصنف square صديقا وبالتالي لا يسمح له بالوصول إلى بياناته الخاصة والمحمية.

تمرين: أعد كتابة البرنامج السابق بحيث تكون الدالة convert مستقلة عن كل الأصناف.

- **البرنامج**

```
// friend classes
#include <iostream>
using namespace std;

class CSquare;

class CRectangle
{
      friend CRectangle convert(CSquare a);
      private:
            int width, height;
      public:
            int area ()
            {
                  return (width * height);
            }
};

class CSquare
{
  friend CRectangle convert(CSquare a);
  private:
      int side;
  public:
      void set_side (int a)
      {
                  side=a;
            }
};

CRectangle convert (CSquare a)
{
      CRectangle R;
      R.width = a.side;
      R.height = a.side;
      return R;
}

int main ()
{
  CSquare sqr;
  sqr.set_side(7);
  CRectangle rect;
  rect = convert(sqr);
  cout <<"Area of rect = "<<rect.area()<<" cm2"<<endl;
  return 0;
}
```

- **مخرجات البرنامج**

```
Area of rect = 49 cm2

Press any key to continue . . .
```