بسم الله الرحمن الرحيم

الصفوف class

الصفوف هي مجموعة من المتغيرات , مختلفة الأنواع عادة ومؤتلفة مع مجموعة دوال مترابطة

- **نستطيع باستعمال الصفوف إنشاء نوع جديد .**
- **النوع هو فئة أو بمعني اخر النوع هو كائن يملك حجما وصفات atribute وحالة ومجموعة قدرات .**
- **يحوي الصف class بداخلة الدوال + المتغيرات فهو شـــــامل .**

التغليف :encapsulation : -

عبارة عن تحزيم كل المعلومات وقدرات ومسئوليات كيان ما في كائن واحد .

Clients : عملاء الصف :

هي صفوف أو دوال تستعمل الصف حيث يتيح التغليف encapsulation

لعملاء الصف باستمالة دون معرفة تفاصيلة أو معرفة كيفية عملة (مثل عملية قيادة السيارة دون الاهتمام بكيفية عملها .)

- **يشار للمتغيرات المتوافرة داخل الصف علي أنها أعضاء متغيرات أعضاء أو بيانات أعضاء member data .**
- **تعالج الدوال في الصف المتغيرات الأعضاء وتسمي الدوال الأعضاء member function أو نهج الصف methods .**

كيفية الإعلان عن الصف :

```
Class   class name

{

المتغيرات + الدوال الأعضاء

};
```

إعلان الصف لا يحجز ذاكرة إنما يخبر المصرف ما هو الصف class وما البيانات التي يتضمنها وحجمة ومدي الذاكرة الضرورية

الكائن object :

يعرف الكائن من النوع الجديد كتعريف المتغير
فمثلا إذا كان عندنا ألclass

```
Class employee
        {
    int age ;
 string name ;
   salary( );
        };
```

فيمكن تعريف كائن من النوع employee كالتالي :

```
Employee  A ;
```

عرفنا كائن a من النوع employee

يستعمل الكائن لوصول إلي المتغيرات الأعضاء والدوال الأعضاء في الصف.

وذلك باستخدام مؤثر (.) كالتالي :

```
A. age = 22;

A. name = " ali"
```

الخاص اذاء العام :

نستعمل كلمات أساسية لإعلان الصف

```
Public
private
```

كل أعضاء الصف (متغيرات + دوال) تكون خاصة private
لذلك يجب وضعها عامة public

```
Class employee
        {
    public:
    int age;
 string name;
   salary( );
        };
```

أصبح لان age و name و alary ( ) عاما مما يمكن من نيلها أو الوصول إليها عن طريق الكائنات دون أخطاء

دوال البناء والهدم : constructor& destructor

لكي نستطيع تمهيد البيانات الأعضاء والدوال الأعضاء في الصف توفر الصفوف داله عضو تسمي constructor البانية وهي قادرة علي اخذ وسائط الضرورية حسب الحوجة ولكنها ليست قادرة علي إرجاع أية قيمة ولا حتي void
والدالة البانية تأخذ اسم الصف class مثلا :

Employee ( ).

كلما أعلنا عن بانية يجب الإعلان عن هادمة destructor حيث تقوم بإخلاء الذاكرة التي قد حجزتها وتسبقها (~) وهي أيضا لا تأخذ أية وسائط ولا ترجع أية قيمة .

~ employee( )

مثال :

```
#include <iostream.h>
class employee
{
public:
employ(int age ); \\  constructor
~employee( );
int getage( );
private :
int itsage ;
\\ constructor of employee
employee : : (int age);
{
its age = age ;
}

employee : : ~ employee( ) \\ destructor
int employee : : getage ()
{
return   itsage ;

}
```

فرط تحميل البانيات :

تستعمل البانية لإنشاء الكائنات أما بعد انتهاء عمل البانية فسيكون لدينا كائن كامل وجاهز للاستعمال .
فمثلا يمكن الحصول علي كائن يتضمن بانيتين اثنين :

- تأخذ الأولي العمر age
- والثانية الاسم name أما الثانية فلا تأخذ أي شيء .

نستطيع فرط تحميل البانيات ولكن لا نستطيع فر تحميل الهادمات .

بانية النسخ :

يقدم المصرف بانية ناسخة افتراضية تستدعي كل مرة تنشئ نسخة عن الكائن عندما نمرر الكائن بقيمة إلي الدالة أو خارجها كقيمة مرجعة ينشئ المصرف نسخة عن هذا الكائن .

تأخذ كل بنيات النسخ باراميترا واحدا مرجعا لكائن من نفس الصف ويستحسن جعلة ثابتا لان البانية لن تحتاج لتعديل الكائن الممرر مثلا :

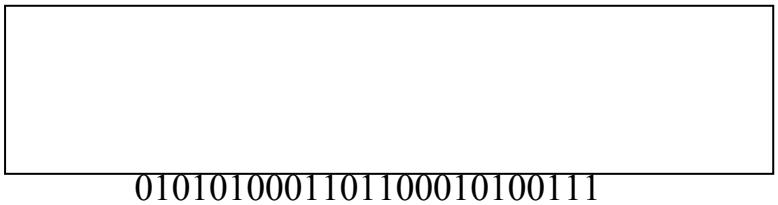
```
Employee (const employee& the employee);
```

هنا أخذت البانية employee مرجعا ثابنا لكائن employee اخر ي.
وهدف بانية النسخ هو إحداث نسخة من the employee .

- الصفوف class هي إنوا ع جديدة
-
- يتضمن الصف بيانات أعضاء وهي متغيرات من عدة صفوف أخري .
-
- يتضمن الصف دوال أعضاء تسمي بالنهج methods وتستعمل للتعامل مع
- البيانات الأعضاء meber data .
-
- قد تكون بينات الصف عامة او خاصة public or private
-
- حيث تتوفر العامة لأي جزء من البرنامج , ما الخاصة فهي متوفرة للدوال الأعضاء
-
- فقط داخل الصف الواحد .
- يسمح فرط التحميل لبيانات الصف بإنشاء صفوف مرنة تعتمد علي كائنات أخري
-
- إذا لم تنشئ بانية نسخ فان المصرف يوفر واحدة تقوم بالنسخ السطحي .

المؤشرات pointer

المؤشر هو متغير variable يحمل عنوان موقع في الذاكرة وذلك لان ذاكرة الحاسوب هو الموقع الذي نحفظ فيه القيم وقسمت الذاكرة إلي مواقع كل موقع هو عنوان في الذاكرة .



01010100011011000010100111

معرفة عنوان المتغير :

يتم باستخدام المؤثر address of (&) :

```
Int main
{

Int age = 5
Cout << "address of age:\t<<&age<<"\n;
Return 0
}
```

خزن العنوان في المؤشر :

كل متغير يملك عنوان خاص به يمكن تخزين عنوان متغير في مؤشر محدد حتي دون معرفة هذا العنوان .

```
Int age = 30 ;
Int *p = 0

P = &age;
```

مؤثر الإسناد غير المباشر :

يستعمل مؤثر الإسناد غير المباشر (*) بطريقتين مختلفتين مع المؤشرات :
1 – الإعلان

2 – والإسناد غير المباشر

عند الاستناد الي المؤشر يشير مؤشر الإسناد غير المباشر * إلي نيل القيمة الموجودة في موقع الذاكرة المخزونة في المؤشر لا إلي العنوان ذاته .

دواعي استعمال المؤشرات :

تستعمل المؤشرات غالبا للاتي :

1 – معالجة المعلومات لموجودة في المخزن المطلق free store

2 – الوصول الي البيانات والدوال الأعضاء في الصف class

3 – تمرير المتغيرات بالرجوع إلي الدوال .

المكدس والخزن المطلق

تتعامل البرامج عموما مع خمسة مناطق في الذاكرة :

1 – حيز الأسماء الشاملة

2 –المخزن المطلق

3– السجلات register

4– حيز الكود

5– المكدس

تحفظ المتغيرات المحلية في المكدس مع باراميترات الدوال . أما الكود فيحف في حيز الكود , والمتغيرات الشاملة مع حيز الأسماء الشاملة , أما السجلات فتستعمل لإدارة شؤون الدوال الداخلية مثل متابعة قمة المكدس ومؤشر التعليمات أما ما تبقي من الذاكر ة فيكرس للمخـــــــــــــزن المـــــطلق free store .

المخزن المطلق :

تحدد الذاكرة في المخزن المطلق باستعمال الكلمة الأساسية new يتبع ذلك نوع الكائن الذي تريد تحديده حتي بعرف المصرف كمية الذاكرة المطلوبة .

عندما تنتهي من المساحة التي حجزتها في الذاكرة ينبقي استدعاء delete لحذف المؤشر حيث تعيد المساحة إلي المخزن المطلق .

تســـرب الذاكرة :

عندما يتم إعادة تعين المؤشر قبل تحرير الذاكرة التي يشير إليها يؤدي لتتسرب الذاكرة

```
Int *p = new short int ;
*p = 72 ;
p = new short int ;
*p = 84
```

ليس هناك طريقة نيل المساحة الأصلية الأولي لتحرير هزة المساحة قبل انتهاء البرنامج

لتجنب تســـــــــــــــــرب الذاكرة يجب كتابة الأتي :

```
int *p  = new  int;

p = 72;

delete   p ;

p = new int;


*p = 84;
```

نلاحظ أنة كلما استدعينا new يجب أن نستدعي delete .

المراجع references

*المرجع هو اسم مستعار تمهيد باسم كائن اخر الاسم البديل للهدف

لإنشاء مرجع نكتب نوع كائن الهدف يليه مؤشر الإسناد & ثم يتبعه اسم المرجع :

Int &rage = someint;

حيث rage هي مرجع لعدد صحيح int .

```
#include<iostream.h>

int main ( )
{

int  age ;

int &rage = age ;

age = 10 ;
cout << " age"<< age<<endl;
cout << " rage"<<r age<<endl;


rage = 12;

cout << " age"<< age<<endl;
cout << " rage"<<r age<<endl;

return 0;
}
```

مؤثر العنوان مع المراجع :

عنوان المرجع هو هدفه target وذلك لان المراجع هي أسماء مستعارة للأهداف :

```
#include<iostream.h>

int main ( )
{
```

```
int  age ;

int &rage = age ;

age = 10 ;
cout << " age"<< age<<endl;
cout << " rage"<<r age<<endl;


cout <<"&  age"<< &age<<endl;
cout << "& rage"<<&r age<<endl;

return 0;
}
```

يمكن الرجوع إلي أي كائن بما في ذلك الكائنات المستحدثة

***نلاحظ أن المراجع أسهل للاستعمال وللفهم والإسناد غير المباشر مخفي وليس من حاجة لإسناد المتغير أكثر من مرة و<u>لكن</u> لا غني عن استعمال المؤشرات لان المؤشرات تتيح مرونة اكبر رغم أنها صعبة الاستعمال ويمكن أن تكون لاغيه null ولا يمكن للمراجع أن تكون لاغيه ولا يمكن إعادة تعينها .**

ولا تنسونا من صالح الدعاء

ابوبكر الجزولي.