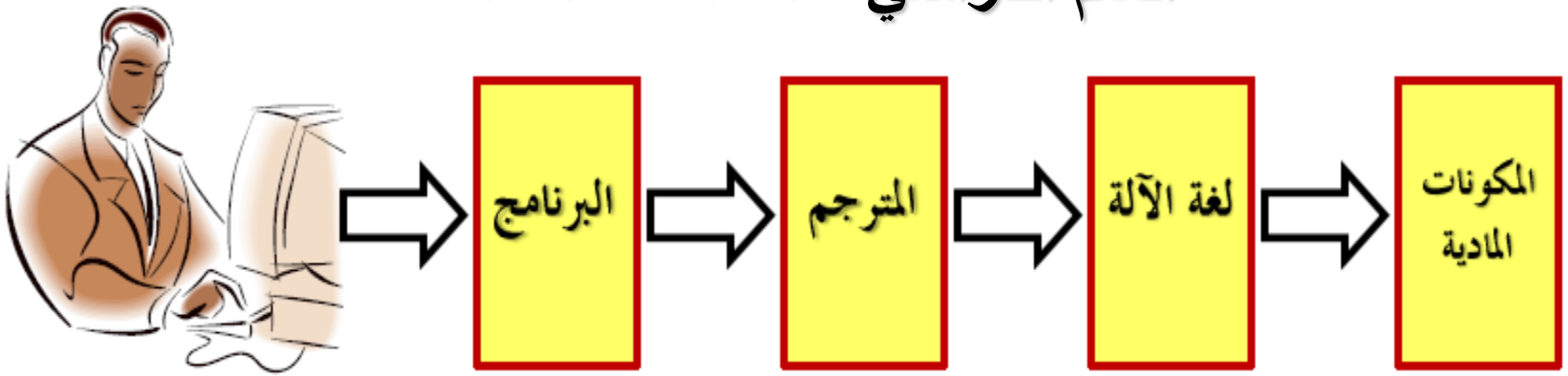
جامعة حماة
كلية العلوم في مصياف / السنة الأولى

المادة: البرمجة و الخوارزميات

المحاضرة السابعة: مقدمة عن الصفوف (الأصناف) في لغة البرمجة

Classes in C++

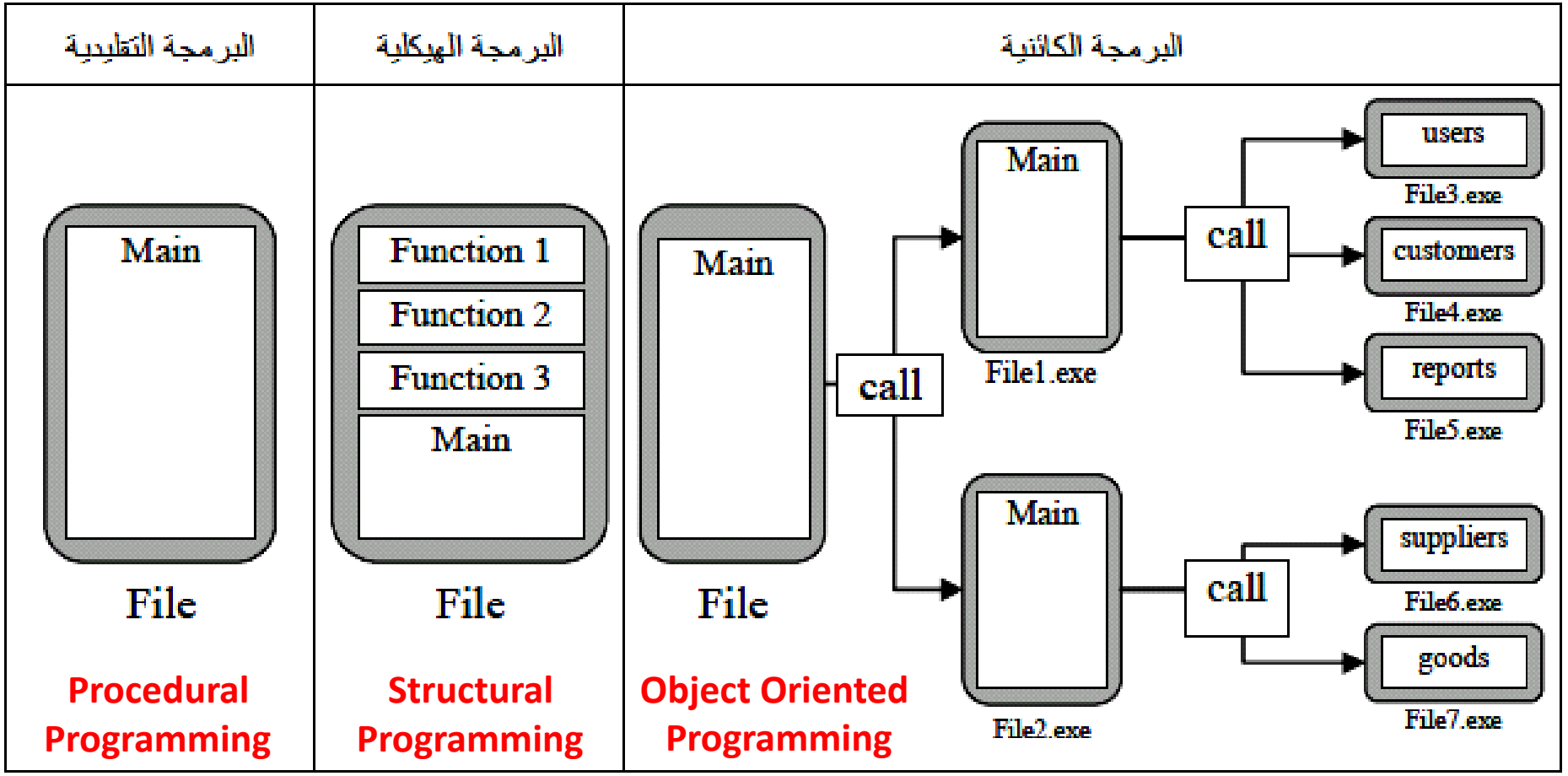
العام الدراسي ٢٠١٨ - ٢٠١٩



تاريخ تطور لغات البرمجة
The History of Programming Languages

- أطلق على لغات برمجة الجيل الأول اسم البرمجة الإجرائية التقليدية Procedural Programming التي تعتمد على سرد السطور البرمجية دون الاعتماد على أي تقنية بنيوية مثل الدوال، حيث تكون جميع متغيراتها عامة Public أي لا يمكن إخفائها، ولكن هذا المسار أوصل البرمجة إلى درجة كبيرة من التعقيد مما دفع بالعاملين في مجال البرمجة إلى التفكير بإيجاد حلول جديدة تسهّل كتابة البرامج الضخمة فتوصلوا إلى البرمجة البنيوية (الهيكلية) Structural Programming التي ساهمت في تقسيم شيفرة البرنامج وتصغيرها وتسهيل التعامل معها من خلال استخدام الدوال والإجراءات.
- مع تطور البرامج و كثرة الدوال عادت مشكلة التعقيدات فظهرت لغات البرمجة كائنية التوّجه (Object Oriented Programming-OOP) التي تتعامل بطريقة مختلفة ومنظمة مع البرامج الضخمة حيث أنها تعتمد على دمج المتغيرات والدوال التي تعمل عليها في كينونة واحدة تسمى الكائن، حيث نستطيع إظهار بعض المتغيرات لتصبح Public وإخفاء بعضها الآخر لتصبح Private، ويتم ذلك من خلال بنية بيانات تدعى الصفوف Classes، وهذا ما يسمى بـ Information Hiding .
- البرمجة كائنية التوّجه OOP تقدم لنا واجهة Interface للتعامل مع الصفوف Classes من دون معرفة التفاصيل التي تجري داخل الصف، فالذي يهم المستخدم في النهاية هو النتيجة وليس كيفية الوصول إلى النتيجة، من أهم هذه اللغات Visual Basic , Visual C++ , Java Builder.
- تختص OOP بالمشاريع البرمجية الضخمة حيث تعتمد على تجزئة البرنامج إلى عدة برامج على شكل قوالب من صفوف classes يتم إنشائها بهدف تنظيم و تسهيل عملها.

تاريخ تطور لغات البرمجة



مدخل إلى البرمجة كائنية (غرضية) التوّجه OOP
Object Oriented Programming

- **OOP**: هي برمجة تقوم على الاهتمام بطريقة حل المشكلة (هندسة الحل) قبل تحويلها إلى شيفرة برمجية (Program (Code، و من تقنياتها الدنيا (المصفوفات Arrays، الدوال Functions، السجلات structs) ومن تقنياتها العليا (المؤشرات Pointers، القوائم المتصلة ...).
- **OOP**: هي طريقة لتجميع المتغيّرات (**Attributes**) و الدوال (**methods**) معاً في قالب واحد يسمّى الصف (الصنف) **Class**. ويتكون الصف class من:
  ١. **صفات Attributes** (متغيرات **Variables**)
  ٢. **خصائص Properties** (دوال **Functions**).

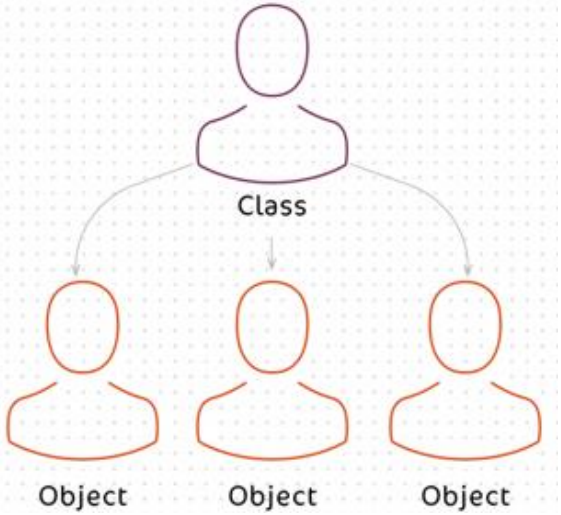
class	
+StudentName:string +StudentID:int -cardID:int	**Attributes**
+Addstuden(StudentName, StudentID) -AddCard(CardID);	**methods**

struct
-StudentName:string -StudentID:int -cardID:int

- **OOP** هي تطوير للبرمجة الهيكلية Struct حيث تمت إضافة خصائص للسجل و سميّ صفاً.
- **OOP** تدعم اسلوب البرمجة المرئية (تصميم الواجهات الرسومية GUI).

الصفوف classes
و
الكائنات objects

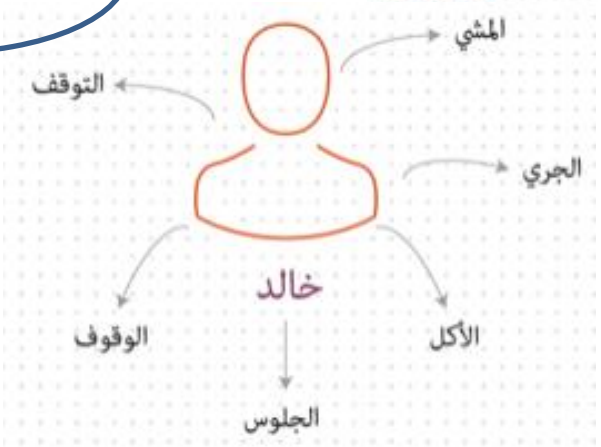
Person	
+Name:string +NationalID:int -cardID:int	**Attributes**
+Add (Name, NationalID) -AddCard(CardID);	**methods**

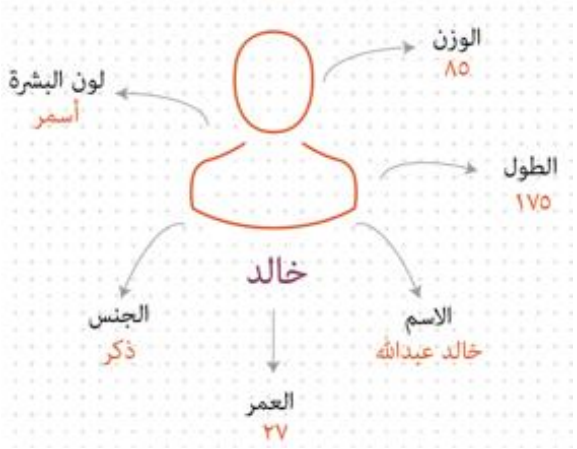



```
main()
{
Person Khaled;
.....
.....
}
```

Object

Class





بعض المفاهيم الأساسية في البرمجة الكائنية OOP
Object Oriented Programming

- **OOP:** هي طريقه برمجة جديدة تمّكننا من تحليل وتصميم التطبيقات Applications على شكل كائنات Objects تحتوي على المتغيّرات Variables وتعرّف عليها مجموعة من العمليات Functions.

١. **الكائن Object**: بنية برمجية مرّكبة تحتوي على مجموعة من البيانات Variables تسمى خصائص أو صفات Attributes ومعرّف عليها مجموعة عمليات (دوال Functions). مثال (طالب، قلم، حاسب).

٢. **الصف (الصنف) Class**: بنية برمجية مرّكبة تحتوي على مجموعة من الكائنات Object التي تشترك في الخصائص والعمليات. مثال (صف الحاسبات، صف الطلاب). الصف يمثل المواصفات العامة للكائنات التي تنتمي إليه، بينما الكائنات تمثل شئ قائم بذاته أو شئ له ذاتية تنتمي لذلك الصف .

- **<u>الفرق بين الصف Class والكائن Object:</u>**
- كل شيء في الوجود يعتبر كائن (الإنسان، الحيوان، النبات، السيارة، الطائرة، الحاسوب ...) كلها كائنات Objects ولكل منها خصائص محددة Attributes ويستطيع القيام بعمليات محددة Functions.
- أما الصف فهو مجموعة من الكائنات المتشابهة فالرجال (صف) وزيد (كائن) منه، والنساء (صف) وأمل (كائن) منه. وكمثال آخر الصف البرمجي: "بطاقة دوام الموظف" التي تحوي الطرق المطلوبة لحساب الأجر وعدد ساعات الدوام أما الكائن "بطاقة دوام محمد" والكائن الآخر "بطاقة دوام حسن" فإنهما يستفيدان من الدوال الموجودة في الصف "بطاقة دوام الموظف" لحساب أجر محمد أو أجر حسن.

الصفوف Classes في لغة البرمجة C++

- **الصف Class**: هو بنية معطيات مهيكلة تحتوي على أعضاء members وتقسم إلى :

١. **Attributes (data members)**: المتغيرات والكائنات والمؤشرات وغيرها.

- تستخدم لتخزين البيانات كما كنا نستخدم المتغيرات في تخزين المعطيات، الآن أصبحت هذه المتغيرات مهيكلة ضمن ما يسمى بالصف Class .
- يجب عدم إعطاء قيم ابتدائية لـ data members الموجودة ضمن الصف مباشرة وإن حاولنا ذلك فإنه سينتج لدينا Compiler Error .
- يوجد دالّة سنتعرف عليها لاحقاً وهي التي ستقوم بعملية التهيئة وإعطاء قيمة ابتدائية لـ data members في الصف وتدعى تلك الدالّة بالباني Constructor .

٢. **Behaviour (methods)**: الدوال (التوابع) التي توّصف سلوك الصف .

- تستخدم للتعبير عن سلوك الصف .
- تستطيع هذه الدوال رؤية جميع المتحولات data members المعرفة ضمن نفس الصف .
- من خلال هذه الدوال يتم التعامل مع الصف، حيث تشكل مجموعة من الدوال واجهة المستخدم في التعامل مع الصف.

الصفوف Classes في لغة البرمجة C++

- **الصف class:** هو بناء برمجي لتجميع البيانات والمهام معاً على شكل مجموعة من الكيانات المتشابهة في الخصائص والسلوك كما هي موجودة في العالم الحقيقي. قد يكون كائن أو أداة او بروتوكول يحتوي على مجموعة من الخصائص و الدوال التي تنفذ عليها للقيام بمهمات محددة.
- **الصف class**: هو بنية معطيات مرّكبة يتم إنشائها بشكل مشابه للسجلات Struct الموجودة في لغة C++ إلا أن الفارق هو أن أعضاء الصف تكون أعضاء خاصة بشكل افتراضي (إذا لم تستخدم **Public** و **Private**) .
- **يعرّف الصف في بداية البرنامج (بعد تضمين المكتبات) على الشكل التالي:**
- **بعد تعريف الصف يتم إنشاء متغير منه ويدعى المتغير عندها بـ Object أو Instance .**

```
class ClassName
 {
Private:
//.....
Public:
 //.....
Protected:
//.....
};
```

class: كلمة محجوزة لإنشاء صف (تكتب بأحرف صغيرة)

ClassName: اسم الصف (أي اسم يفضّل أن يبدأ بحرف كبير)

Private: متغيرات و دوال خاصة داخل الصف نفسه و لا يمكن رؤيتها إلا من داخل الصف نفسه و من أصدقائه فقط

Public: متغيرات و دوال عامة داخل الصف يمكن رؤيتها من داخل أي صف آخر أو برنامج رئيسي يستدعي هذا الصف.

Protected: متغيرات و دوال موّرثة.

القالب العام لإنشاء كائن من نوع صف Class

```
class ClassName
 {
 Public:
data type var1; …..
data type function_name1; …..
 …..
 Private:
data type var1;  …..
data type function_name1; …..
 …..
 Protected:
data type var1; …..
data type function_name1; …..
  …..  …..
 };
```

- كلمة مفتاحية Class و اسم الصف
- بداية تعاريف الصف
- منطقة المتغيّرات و الدوال **العامّة**
- تعريف متغيّرات عامة
- تعريف دوال عامة
- .....
- منطقة المتغيّرات و الدوال **الخاصة**
- تعريف متغيّرات خاصة
- تعريف دوال خاصة
- .....
- منطقة المتغيّرات و الدوال **الموّرثة**
- تعريف متغيّرات يمكن توريثها
- تعريف دوال يمكن توريثها
- .....
- انتهاء إنشاء الصف

تصنيف class members في لغة C++

يتم تصنيف class members في لغة C++ إلى ثلاث أصناف وهي :

١. **Private**:هو النمط الافتراضي أي إذا لم نذكر Access modifier للأعضاء Member في الصف فيتم وضعه افتراضياً على أنه Private. ويتم رؤيته فقط من داخل الصف ومن الأصدقاء Friends .

٢. **Public**: تتم رؤيته من داخل الصف ومن الأصدقاء ومن أي مكان تتم رؤية الغرض .

٣. **Protected**: يحتوي الصف على سماحيات تمنع المستخدمين من الوصول إلى البيانات المحمية، و الصفات الموجودة داخل الصف لا يعرفها إلا من قام بإنشاء الصف، وللوصول إلى تتم رؤيته من داخل الصف ومن الأصدقاء ومن الصفوف الأبناء .

- الكلمات الثلاث السابقة هي كلمات معرفة مسبقاً في اللغة .
- لا يمكن الوصول إلى Private Member من خارج الصف.
- يمكن الوصول إلى Public Member من خارج الصف .
- إن Private Member يبقى مخفي للمستخدم Client الذي سوف يستخدم هذا الصف.
  - **سؤال:** كيف سيتعامل هذا المستخدم مع هذه المتحولات المخفية الخاصة ؟؟
  - جواب: عن طريق دوال methods الـ Access Modifiers لها هو public .

الصفوف Classes في لغة البرمجة C++

```
class Student {
Public:
string StudentName;
int StudentID;
int Addstuden(StudentID);
string Addstuden(StudentName);
Private:
int cardID;
int AddCard(CardID);
};
```

- مثال صف الطالب كما في الأسفل يحتوي الطالب على مجموعة من الخصائص وهي (اسمه ورقمه الجامعي ومعلومات بطاقته المصرفية). وهناك مجموعة دوال تنفذ على هذه الخصائص وهي (اضافة طالب واضافة بطاقة لطالب).
- الخصائص (المتغيرات): أي شيء يوصف به الصف
- الدوال: هي عمليات تنفذ على الخصائص الموجودة في الصف
- الصف في البرمجة كائنية التوّجه يعرّف على الشكل التالي:

Class

ClassName	Student
Attributes (data members)	+StudentName:string +StudentID:int -cardID:int
Behaviour (methods)	+Addstuden(StudentName, StudentID) -AddCard(CardID);

التعامل مع الكائن Object والصف Class

- قبل التعامل مع الكائن يجب أن يتم إنشاؤه و طريقة إنشاءه تتلخص في إسباق اسم الكائن باسم الصف و بهذه العملية تكون قد حجزت في الذاكرة مكان باسم الكائن فيه مكان للتخزين يتلائم مع نوع و عدد المتغيرات الموجود في Private في الصف. وللتعامل مع هذا الكائن فإننا نطلب تنفيذ أحد الأعضاء الداليّة لتخزين هذه النتائج في تلك المساحة.
- يتم التعامل مع الكائن من داخل الدالة الرئيسية main() حيث يتم إنشاء متغير منه ويدعى المتغير عندها بـالكائن object أو instance، ثم يتم التعامل مع متغيرات هذا الكائن الجديد عن طريق الدوال المعرّفة ضمن الصف أو خارجه، بشكل مشابه للتعامل مع السجلات struct .

```
void main()
{
Class_name Object_name ;          إنشاء كائن
Object_name.function member();    استدعاء دالّة لتنفيذها
}
```

مثال على الصفوف classes

برنامج لطباعة الوقت المدخل عن طريق تعريف الدوال خارج الصف

1

```
#include<iostream>
using namespace std;
class readtime
{
private:
int min;
int hor;
int sec;
public:
void input();
void output();
};
```

2

```
void time::input()
 {
cout<<"Enter the time please\n";
cin>>hor>>min>>sec;
 }
```

```
void time::output()
 {
cout<<"\nTime now is:";
cout<<hor<<":"<<min<<":"<<sec<<endl;
 }
```

```
void main()
{
readtime obj;
obj.input();
obj.output();
}
```

Output الخرج

```
Enter the time please
01
29
45
Time now is: 1:29:45
Press any key to
continue
```

مثال على المصفوفات و الصفوف classes

برنامج لإدخال وطباعة بيانات الموظفين باستخدام الصفوف

1

```
#include<iostream>
using namespace std;
class employee
{
Private:
char Name[50];  char Address[100];
int ID;  float Salary;
char Birthday[20];  int Boolean;
Public:
   void input(int i)  {
cout<<"Enter the data of("<<i<<
")employee:";
cout<<"\nName:“;     cin>>Name;
cout<<"\nID:";          cin>>ID;
cout<<"\nBirthday:"; cin>>Birthday;
cout<<"\nSalary=";    cin>>Salary;
cout<<"\nAddress:"; cin>>Address;
   Boolean=1;       }
```

2

```
   void output(int i)
    {
if(Boolean==1)
 {
cout<<"The data of("<<i<<")employee:\n";
cout<<"Name:"<<Name<<endl;
cout<<"ID:"<<ID<<endl;
cout<<“Birthday:"<<Birthday<<endl;
cout<<“Salary="<<Salary<<“$\n”;
cout<<“Address:"<<Address<<endl;
  }
else if(Boolean!=1)
cout<<“No data for this emplyee"<<endl;
    }
};
```

3

```
void main()
{
employee obj[100];
char ans; int i;
cout<<"Choose what you want:\n”;
cout<<" 1-input (press 1).\n”;
cout<<" 2-output (press 2).\n”;
cout<<“ 3-quit (press q).\n";
cin>>ans;
while(ans!='q')
{
if(ans=='1')
{
cout<<"Enter the SN of employee:";
cin>>i;
obj[i-1].input(i);
}
```

4

```
else if(ans=='2')
{
cout<<"Enter employee Number\n";
cin>>i;
obj[i-1].output(i);
}
cout<<"continue ?"<<endl;
cout<<"chose what you want:\n”;
cout<<" 1-input (press 1).\n”;
cout<<" 2-output (press 2).\n”;
cout<<“ 3-quit (press q).\n";
cin>>ans;
}
}
```

مقارنة بين البرمجة الإجرائية والبرمجة بالكائنات

م	البرمجة الإجرائية	البرمجة بالكائنات
١	البرنامج يتحكم في مسار تنفيذه أمراً بعد أمر	المستخدم ونظام التشغيل والبرنامج جميعهم يتحكمون في مسار تنفيذ البرنامج
٢	صعوبة إنشاء واجهة للمستخدم	سهولة إنشاء واجهة للمستخدم
٣	صعوبة الربط مع قواعد البيانات المختلفة.	سهولة الربط مع قواعد البيانات المختلفة.
٤	يتم كتابة الأوامر والتعليمات من المستخدم لتنفيذ البرنامج	يتم استخدام الكائنات لتنفيذ البرنامج

البرمجة البنيوية (الدوال) Functions و البرمجة بالكائنات Objects

أوجه الاختلاف بين البرمجة بالدوال والبرمجة بالكائنات:

البرمجة بالدوال	البرمجة بالكائنات
يمكن بواسطتها بناء برنامج متكامل.	يمكن بواسطتها بناء مشروع متكامل.
تقسم البرنامج إلى أجزاء.	تقسم المشروع إلى برامج مستقلة فرعية.
البرنامج ضمن ملف واحد يحوي كل الدوال	كل برنامج مستقل خارج الملف الرئيسي، ويتم استدعاء البرنامج عند الحاجة إليه.
تحتاج من واحد إلى ثلاثة مبرمجين لبناء البرنامج	تحتاج إلى مبرمج أو أكثر لكل برنامج مستقل
يشترك المبرمجين في بناء البرنامج الواحد	كل مبرمج أو فريق مبرمجين مسئول عن برنامجه.
إذا وجد خطأ في دالة يتوقف البرنامج بأكمله	إذا وجد خطأ في برنامج فرعي، يستمر البرنامج الرئيسي للمشروع في العمل.
يحتوي البرنامج على عدة دوال	قد يحتوي البرنامج على أكثر من كائن.
تعمل الدوال بشكل مستقل عن بعضها	تتكامل الدوال "الخصائص" في انجاز المهمة الواحدة. وتتكامل الكائنات بواسطة الوراثة أو تعدد الواجهات.
الكود يشكل جزء من البرنامج	الكائن يمثل جزء من المشروع

ملاحظة: إذا تم وضع الكلاسات داخل الملف الأساسي ولم توضع في برامج فرعية فإن الكثير من مميزات البرمجة بالكائنات ستختفي!