برمجة :++ Cمن تحليل المشكلات إلى تصميم البرنامج ، الإصدار الخامس

C++ Programming: From Problem Analysis to Program Design, Fifth Edition

Chapter 11: Classes and Data Abstraction

الفصل 11: الفئات واستخراج البيانات

```
class classIdentifier
{
    classMembersList
};
```

الطبقات

Classes

- فصل :جمع عدد ثابت من المكونات)الأعضاء(
- صيغة التعريف:

- Class: collection of a fixed number of components (members)
- Definition syntax:

  - يحدد نوع البيانات ، لا يتم تخصيص ذاكرة
  - لا تنس الفاصلة المنقوطة بعد قوس الإغلاق

  – Defines a data type, no memory is allocated
  – Don’t forget the semicolon after closing brace

Classes (cont'd.) فئات)تابع(

- Class member can be a variable or a function
- If a member of a `class` is a variable
  - It is declared like any other variable
- In the definition of the `class`
  - You cannot initialize a variable when you declare it
- If a member of a `class` is a function
  - Function prototype is listed
  - Function members can (directly) access any member of the `class`

- يمكن أن يكون عضو الفصل متغيرًا أو دالة
- إذا كان عضوًا في صف دراسي متغير
  - يتم الإعلان عنه مثل أي متغير آخر
- في تعريف صف دراسي
  - لا يمكنك تهيئة متغير عندما تعلنه
- إذا كان عضوًا في صف دراسي هي وظيفة
  - تم سرد النموذج الأولي للوظيفة
  - يمكن لأعضاء الوظيفة)مباشرة (الوصول إلى أي عضو في صف دراسي

Classes (cont'd.)

فئات)تابع(

- Three categories of class members
  - `private` (default)
    - Member cannot be accessed outside the `class`
  - `public`
    - Member is accessible outside the class
  - `protected`

- ثلاث فئات من أعضاء الصف
  - نشر)إفتراضي(
    - لا يمكن الوصول إلى العضو خارج صف دراسي
  - عامة
    - العضو يمكن الوصول إليه من خارج الفصل
  - محمي

Classes (cont'd.) فئات)تابع(

```
class clockType
{
public:
    void setTime(int, int, int);
    void getTime(int&, int&, int&) const;
    void printTime() const;
    void incrementSeconds();
    void incrementMinutes();
    void incrementHours();
    bool equalTime(const clockType&) const;

private:
    int hr;
    int min;
    int sec;
};
```

لا يمكن لهذه الدالات تعديل متغيرات العضو لمتغير من النوع `ClockType`

These functions cannot modify the member variables of a variable of type `clockType`

`const`: formal parameter can't modify the value of the actual parameter

مقدار ثابت: لا يمكن تعديل المعلمة الرسمية قيمة المعلمة الفعلية

`private` members, can't be accessed from outside the class

نشر أفراد، لا يمكن الوصول إليها من خارج الفصل

إعلان متغير)كائن(

Variable (Object) Declaration

- Once a `class` is defined, you can declare variables of that type

```
clockType     myClock;
clockType     yourClock;
```

- A `class` variable is called a `class` object or `class` instance

- مرة صف دراسي يمكن تعريف المتغيرات من هذا النوع

`ClockType myClock ؛`

على مدار الساعة اكتب `yourClock`؛

- أ صف دراسي المتغير يسمى أ صف دراسي كائن أو صف دراسي نموذج

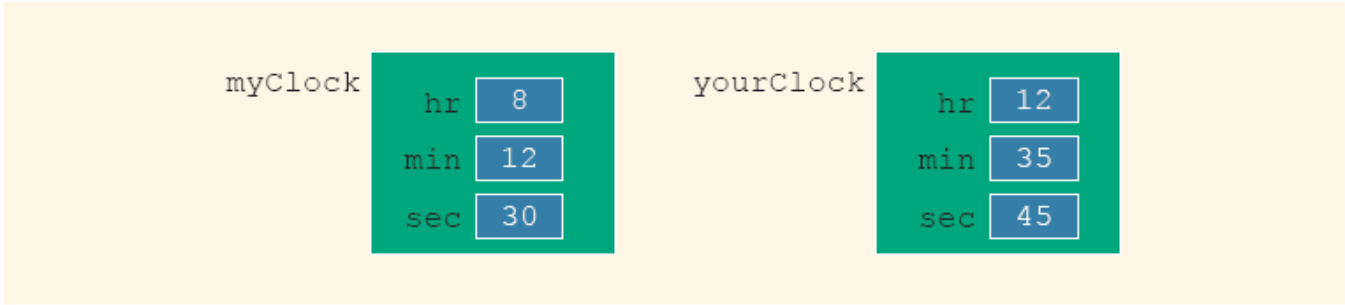


FIGURE 12-2 Objects `myClock` and `yourClock`

```
classObjectName.memberName
```

الوصول إلى أعضاء الفصل الدراسي

Accessing Class Members

- Once an object is declared, it can access the `public` members of the class
- Syntax:

- بمجرد إعلان كائن ما ، يمكنه الوصول إلى ملف عامة أعضاء الفصل
- بناء الجملة:

  - The dot (`.`) is the **member access operator**

- If object is declared in the definition of a member function of the `class`, it can access the `public` and `private` members

  - النقطة (`.`)هل **عضو وصول عامل**

- إذا تم التصريح عن الكائن في تعريف وظيفة عضو في صف دراسي، يمكنه الوصول إلى ملف عامة و نشر أفراد

Accessing Class Members (cont'd.)

الوصول إلى أعضاء الفصل الدراسي)تابع(

EXAMPLE 12-1

Suppose we have the following declaration (say, in a user's program):

```
clockType myClock;
clockType yourClock;
```

Consider the following statements:

```
myClock.setTime(5, 2, 30);
myClock.printTime();
yourClock.setTime(x, y, z);    //assume x, y, and z are
                               //variables of type int

if (myClock.equalTime(yourClock))
.
.
.

myClock.hr = 10;               //illegal
myClock.min = yourClock.min;   //illegal
```

Built-in Operations on Classes

- Most of C++'s built-in operations do not apply to classes
  - Arithmetic operators cannot be used on class objects unless the operators are overloaded
  - You cannot use relational operators to compare two class objects for equality
- Built-in operations valid for class objects:
  - Member access (.)
  - Assignment (=)

- لا تنطبق معظم عمليات C ++المضمنة على الفئات
  - لا يمكن استخدام العوامل الحسابية في كائنات الفئة ما لم يتم تحميل العوامل فوق طاقتها
  - لا يمكنك استخدام العوامل العلائقية لمقارنة كائنين من فئة المساواة
- العمليات المضمنة الصالحة لكائنات الفئة:
  - وصول الأعضاء (.)
  - واجب (=)

Assignment Operator and Classes

عامل التخصيص والفئات

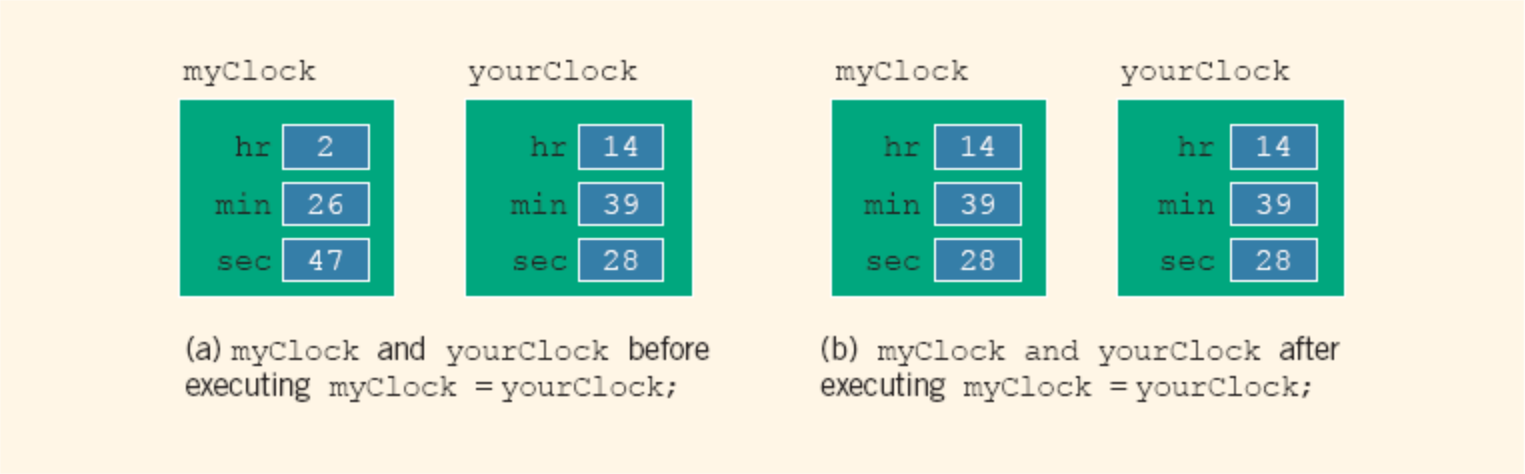


FIGURE 12-3 myClock and yourClock before and after executing the statement myClock = yourClock;

Functions and Classes

- Objects can be passed as parameters to functions and returned as function values
- As parameters to functions
  - Objects can be passed by value or by reference
- If an object is passed by value
  - Contents of data members of the actual parameter are copied into the corresponding data members of the formal parameter

- يمكن تمرير الكائنات كمعلمات إلى الوظائف وإعادتها كقيم دالة
- كمعلمات للوظائف
  - يمكن تمرير الكائنات بالقيمة أو بالإشارة
- إذا تم تمرير كائن بالقيمة
  - يتم نسخ محتويات بيانات أعضاء المعلمة الفعلية إلى أعضاء البيانات المقابلة للمعلمة الرسمية

Reference Parameters and Class Objects (Variables)

- Passing by value might require a large amount of storage space and a considerable amount of computer time to copy the value of the actual parameter into the formal parameter
- If a variable is passed by reference
  - The formal parameter receives only the address of the actual parameter

- قد يتطلب تمرير القيمة قدرًا كبيرًا من مساحة التخزين وقدرًا كبيرًا من وقت الكمبيوتر لنسخ قيمة المعلمة الفعلية إلى المعلمة الرسمية
- إذا تم تمرير متغير عن طريق المرجع
  - المعلمة الرسمية تتلقى فقط عنوان المعلمة الفعلية

Reference Parameters and Class Objects (Variables) (cont'd.)

- Pass by reference is an efficient way to pass a variable as a parameter
  - Problem: when passing by reference, the actual parameter changes when formal parameter changes
  - Solution: use `const` in the formal parameter declaration

- يعد التمرير بالمرجع طريقة فعالة لتمرير متغير كمعامل
  - المشكلة :عند تمرير المرجع ، تتغير المعلمة الفعلية عندما تتغير المعلمة الرسمية
  - الحل :استخدم مقدار ثابت في إعلان المعلمة الرسمي

```
void clockType::setTime(int hours, int minutes, int seconds)
{
    if (0 <= hours && hours < 24)
        hr = hours;
    else
        hr = 0;

    if (0 <= minutes && minutes < 60)
        min = minutes;
    else
        min = 0;

    if (0 <= seconds && seconds < 60)
        sec = seconds;
    else
        sec = 0;
}
```

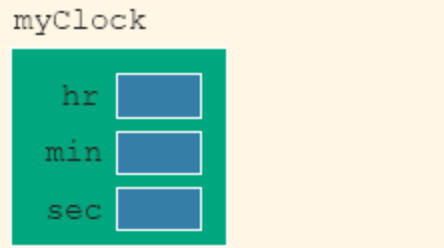
تنفيذ وظائف الأعضاء

مشغل دقة النطاق

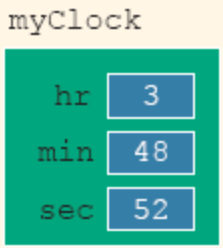
Scope resolution operator

Implementation of Member Functions (cont'd.)

تنفيذ وظائف الأعضاء)تابع(



(a) myClock before executing
myClock.setTime(3, 48, 52);



(b) myClock after executing
myClock.setTime(3, 48, 52);

FIGURE 12-4 myClock before and after executing the statement myClock.setTime(3, 48, 52);

Implementation of Member Functions (cont'd.)

تنفيذ وظائف الأعضاء)تابع(

```
void clockType::getTime(int& hours, int& minutes,
                        int& seconds) const
{
    hours = hr;
    minutes = min;
    seconds = sec;
}

void clockType::printTime() const
{
    if (hr < 10)
        cout << "0";
    cout << hr << ":";

    if (min < 10)
        cout << "0";
    cout << min << ":";

    if (sec < 10)
        cout << "0";
    cout << sec;
}
```

ber

تنفيذ وظائف الأعضاء)تابع(

```
void clockType::incrementHours()
{
    hr++;
    if (hr > 23)
        hr = 0;
}

void clockType::incrementMinutes()
{
    min++;
    if (min > 59)
    {
        min = 0;
        incrementHours(); //increment hours
    }
}

void clockType::incrementSeconds()
{
    sec++;

    if (sec > 59)
    {
        sec = 0;
        incrementMinutes(); //increment minutes
    }
```

```
bool clockType::equalTime(const clockType& otherClock) const
{
    return (hr == otherClock.hr
            && min == otherClock.min
            && sec == otherClock.sec);
}
```

تنفيذ وظائف الأعضاء)تابع(

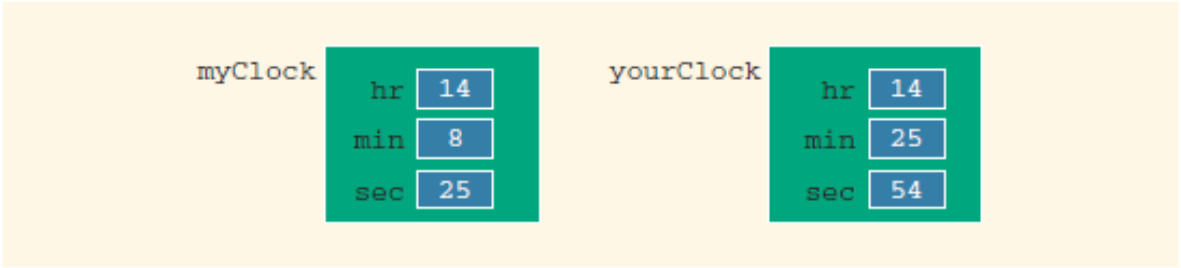


FIGURE 12-5 Objects `myClock` and `yourClock`

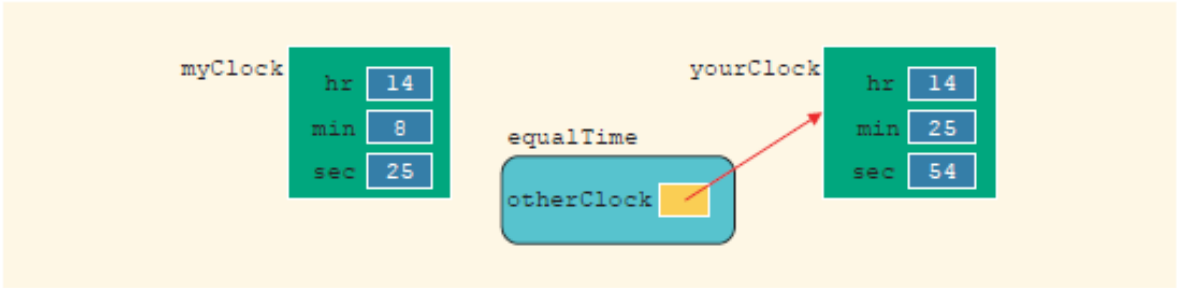


FIGURE 12-6 Object `myClock` and parameter `otherClock`

Implementation of Member Functions (cont'd.)

تنفيذ وظائف الأعضاء)تابع(

- Once a `class` is properly defined and implemented, it can be used in a program
  - A program that uses/manipulates the objects of a class is called a **client** of that class
- When you declare objects of the `class clockType`, every object has its own copy of the member variables (`hr`, `min`, and `sec`)
- Variables such as `hr`, `min`, and `sec` are called instance variables of the class
  - Every object has its own instance of the data

- مرة صف دراسي يتم تعريفه وتنفيذه بشكل صحيح ، ويمكن استخدامه في البرنامج
  - يُطلق على البرنامج الذي يستخدم /يتعامل مع كائنات فئة ما **عميل** من تلك الفئة
- عندما تعلن عن كائنات صف دراسي `ClockType`، كل كائن له نسخته الخاصة من المتغيرات الأعضاء)ساعةو دقيقة، و ثانية(
- المتغيرات مثل ساعةو دقيقة، و ثانية وتسمى متغيرات سريعة الطبقة
  - كل كائن له مثيله الخاص من البيانات

Accessor and Mutator Functions

وظائف الموصل والمفتاح

- Accessor function: member function that only accesses the value(s) of member variable(s)
- Mutator function: member function that modifies the value(s) of member variable(s)
- Constant function:
  - Member function that cannot modify member variables
  - Use `const` in function heading

- وظيفة الموصّل :وظيفة العضو التي تصل فقط إلى قيمة)قيم (متغير)متغيرات (العضو
- وظيفة المطفر :دالة العضو التي تعدل قيمة)قيم (متغير)متغيرات (العضو
- وظيفة ثابتة:
  - دالة العضو التي لا يمكنها تعديل متغيرات الأعضاء
  - يستخدم مقدار ثابت في عنوان الوظيفة

Order of `public` and `private` Members of a Class

- C++ has no fixed order in which you declare `public` and `private` members
- By default all members of a class are `private`
- Use the member access specifier `public` to make a member available for `public` access

- C ++ليس له ترتيب ثابت تعلن فيه عامة و نشر أفراد
- افتراضيا جميع أعضاء الفصل هم نشر
- استخدم محدد وصول العضو عامة لإتاحة العضو لـ عامة التمكن من

Order of `public` and `private` Members of a Class (cont'd.)

ترتيب عامة و نشر أعضاء الفصل)تابع(

EXAMPLE 12-3

This declaration is the same as before. For the sake of completeness, we include the class definition:

```
class clockType
{
public:
    void setTime(int, int, int);
    void getTime(int&, int&, int&) const;
    void printTime() const;
    void incrementSeconds();
    void incrementMinutes();
    void incrementHours();
    bool equalTime(const clockType&) const;

private:
    int hr;
    int min;
    int sec;
};
```

Order of `public` and `private` Members of a Class (cont'd.)

ترتيب عامة و نشر أعضاء الفصل)تابع(

EXAMPLE 12-4

```
class clockType
{
private:
    int hr;
    int min;
    int sec;

public:
    void setTime(int, int, int);
    void getTime(int&, int&, int&) const;
    void printTime() const;
    void incrementSeconds();
    void incrementMinutes();
    void incrementHours();
    bool equalTime(const clockType&) const;
};
```

Order of `public` and `private` Members of a Class (cont'd.)

ترتيب عامة و نشر أعضاء الفصل)تابع(

EXAMPLE 12-5

```
class clockType
{
    int hr;
    int min;
    int sec;

public:
    void setTime(int, int, int);
    void getTime(int&, int&, int&) const;
    void printTime() const;
    void incrementSeconds();
    void incrementMinutes();
    void incrementHours();
    bool equalTime(const clockType&) const;
};
```

Constructors

- Use constructors to guarantee that data members of a class are initialized
- Two types of constructors:
  - With parameters
  - Without parameters (**default constructor**)
- The name of a constructor is the same as the name of the class
- A constructor has no type

- استخدم المُنشئين لضمان تهيئة أعضاء البيانات للفئة
- نوعان من الصانعين :
  - مع المعلمات
  - بدون معلمات)**المُنشئ الافتراضي**(
- اسم المُنشئ هو نفس اسم الفئة
- المنشئ ليس له نوع

Constructors (cont'd.)

- A class can have more than one constructor
  - Each must have a different formal parameter list
- Constructors execute automatically when a class object enters its scope
  - They cannot be called like other functions
  - Which constructor executes depends on the types of values passed to the class object when the class object is declared

- يمكن للفصل أن يحتوي على أكثر من مُنشئ واحد
  - يجب أن يكون لكل منها قائمة معلمات رسمية مختلفة
- يتم تنفيذ البناة تلقائيًا عند دخول كائن فئة إلى نطاقه
  - لا يمكن تسميتها مثل الوظائف الأخرى
  - يعتمد المُنشئ الذي يتم تنفيذه على أنواع القيم التي تم تمريرها إلى كائن الفئة عند التصريح عن كائن الفئة

```
class clockType
{
public:
    void setTime(int, int, int);
    void getTime(int&, int&, int&) const;
    void printTime() const;
    void incrementSeconds();
    void incrementMinutes();
    void incrementHours();
    bool equalTime(const clockType&) const;
    clockType(int, int, int);  //constructor with parameters
    clockType();  //default constructor

private:
    int hr;
    int min;
    int sec;
};
```

بناة)تابع(

Constructors (cont'd.) بناة)تابع(

```
        hr = hours;                                    t seconds)
    else
        hr = 0;

    if (0 <= minutes && minutes < 60)
        min = minutes;
    else
        min = 0;

    if (0 <= seconds && seconds < 60)
        sec = seconds;
    else
        sec = 0;
}

clockType::clockType()  //default constructor
{
    hr = 0;
    min = 0;
    sec = 0;
}
```

Can be replaced with:

```
setTime(hours, minutes, seconds);
```

يمكن الاستعاضة عنها بـ:

```
setTime ( الساعات ، الدقائق ، الثواني ؛)
```

Invoking a Constructor

- A constructor is automatically executed when a class variable is declared
- يتم تنفيذ المُنشئ تلقائيًا عند الإعلان عن متغير فئة

```
className classObjectName;
```

استدعاء المنشئ الافتراضي

Invoking the Default Constructor

- To invoke the default constructor:

- لاستدعاء المُنشئ الافتراضي:

- Example:

```
clockType yourClock;
```

- مثال:

على مدار الساعة اكتب `yourClock`؛

Invoking a Constructor with Parameters

استدعاء منشئ بالمعلمات

- Syntax:
- بناء الجملة:

```
className classObjectName(argument1, argument2, ...);
```

- The number of arguments and their type should match the formal parameters (in the order given) of one of the constructors
  - Otherwise, C++ uses type conversion and looks for the best match
  - Any ambiguity leads to a compile-time error

- يجب أن يتطابق عدد الوسائط ونوعها مع المعلمات الرسمية)بالترتيب المعطى (لأحد المنشئين
  - خلاف ذلك ، يستخدم C ++تحويل النوع ويبحث عن أفضل تطابق
  - أي غموض يؤدي إلى خطأ في وقت الترجمة

```
clockType clockType(int = 0, int = 0, int = 0);    //Line 1
```

Constructors and Default Parameters

المنشئات والمعلمات الافتراضية

- If you replace the constructors of `clockType` with the constructor in Line 1, you can declare `clockType` objects with zero, one, two, or three arguments as follows:

```
clockType clock1;                  //Line 2
clockType clock2(5);               //Line 3
clockType clock3(12, 30);          //Line 4
clockType clock4(7, 34, 18);       //Line 5
```

- إذا قمت باستبدال المنشئين من `ClockType` مع المُنشئ في السطر 1، يمكنك التصريح `ClockType` كائنات ذات وسيطات صفرية أو واحدة أو اثنتين أو ثلاث على النحو التالي:

Classes and Constructors: A Precaution

الفئات والمُنشئون: احتياطي

- If a class has no constructor(s), C++ provides the default constructor
  - However, object declared is still uninitialized
- If a class includes constructor(s) with parameter(s), but not the default constructor
  - C++ does not provide the default constructor

- إذا لم يكن للفئة مُنشئ)مُنشئ (، فإن C ++توفر المُنشئ الافتراضي
  - ومع ذلك ، لا يزال الكائن الذي تم التصريح عنه غير مهيأ
- إذا كانت الفئة تشتمل على مُنشئ)مُنشئ (مع معلمة)معلمات (، ولكن ليس المُنشئ الافتراضي
  - لا يوفر C ++المُنشئ الافتراضي

المدمرات

Destructors

- Destructors are functions without any type
- The name of a destructor is the character '~' followed by class name
  - For example:

```
~clockType();
```

- A class can have only one destructor
  - The destructor has no parameters
- The destructor is automatically executed when the class object goes out of scope

- المدمرات هي وظائف بدون أي نوع
- اسم المدمر هو الحرف ~'متبوعًا باسم الفصل
  - على سبيل المثال:

```
~ clockType ();
```

- يمكن للفئة أن تحتوي على مدمر واحد فقط
  - المدمر ليس له معلمات
- يتم تنفيذ التدمير تلقائيًا عندما يخرج كائن الفئة عن النطاق