

C++

# 程式語言（二）

Introduction to Programming (II)

Classes & Objects

Joseph Chuang-Chieh Lin

Dept. CSE, NTOU

# Platform/IDE

- Dev-C++



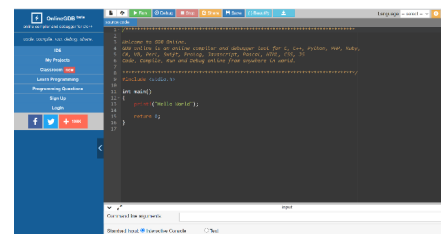
<https://www.pngegg.com/en/search?q=Dev-C>

- Codeblocks

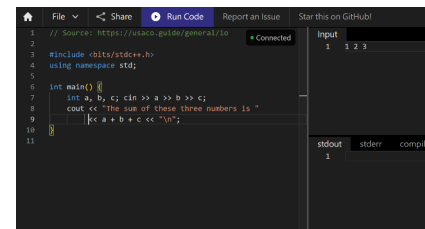


<https://icons8.com/icons/set/code-blocks>

- OnlineGDB (<https://www.onlinegdb.com/>)



- Real-Time Collaborative Online IDE (<https://ide.usaco.guide/>)



# Textbooks (We focusing on C++11)

- *Learn C++ Programming by Refactoring* (由重構學習 C++ 程式設計). Pang-Feng Liu (劉邦鋒). NTU Press. 2023.
- *C++ Primer. 5th Edition.* Stanley B. Lippman, Josée Lajoie, Barbara E. Moo. 2019.
- *Effective C++.* Scott Meyers. O'Reilly. 2016.
- *Thinking in C++. Vol. 1: Introducing to Standard C++.* 2nd Edition. Bruce Eckel. Prentice Hall PTR. 2000.

# Useful Resources

- Tutorialspoint
  - <https://www.tutorialspoint.com/cplusplus/index.htm>
  - Online C++ Compiler
- Programiz
  - <https://www.programiz.com/cpp-programming>
- LEARN C++
  - <https://www.learncpp.com/>
- MIT OpenCourseWare - Introduction to C++
  - <https://ocw.mit.edu/courses/6-096-introduction-to-c-january-iap-2011/pages/lecture-notes/>
- Learning C++ Programming
  - <https://www.programiz.com/cpp-programming>
- GeeksforGeeks
  - <https://www.geeksforgeeks.org/c-plus-plus/>



# Classes & Objects

# Class

- A blueprint/prototype/sketch for the **object**.
- To design a “car”, you need:
  - Wheels
  - Engines
  - A steering wheel
  - Windows
  - Lights
  - ...
  -

# Class

- To design a “school”, you need:
  - Buildings
    - Windows
    - Chairs
    - A Blackboard
    - ...
  - Teachers
  - Students
  - Walls
  - Staffs

# Class

- We define our own data structures by defining a **class**.
- A class defines a **type** along with a collection of **operations** that are related to that type.
- A primary focus of the design of C++ codes is to make it possible to define **class types** that *behave as naturally as the built-in types*.



# Create a class (by an example)

```
class Room {  
    public:  
        double length;  
        double breadth;  
        double height; } data members  
                           資料成員  
  
        double calculateArea() {  
            return length * breadth;  
        }  
  
        double calculateVolume() {  
            return length * breadth * height;  
        }  
};
```

member functions  
成員函式

# Create a class (by an example)

```
class Room {  
    public:  
        double length;  
        double breadth;  
        double height;  
  
        double calculateArea(){  
            return length * breadth;  
        }  
  
        double calculateVolume(){  
            return length * breadth * height;  
        }  
  
};
```

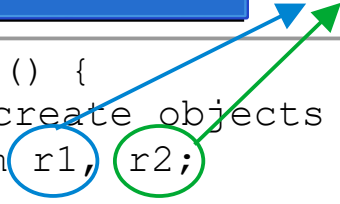
```
int main() {  
    // create objects of Room  
    Room r1, r2;  
  
    //assign values to data members  
    r1.length = 42.5;  
    r1.breadth = 30.8;  
    r1.height = 19.2;  
  
    //calculate the area  
    cout << "area: ";  
    cout << r1.calculateArea();  
    cout << endl;  
}
```

# Create a class (by an example)

different instances!

```
class Room {  
    public:  
        double length;  
        double breadth;  
        double height;  
  
        double calculateArea() {  
            return length * breadth;  
        }  
  
        double calculateVolume() {  
            return length * breadth * height;  
        }  
};
```

```
int main() {  
    // create objects of Room  
    Room r1, r2;  
  
    //assign values to data members  
    r1.length = 42.5;  
    r1.breadth = 30.8;  
    r1.height = 19.2;  
  
    //calculate the area  
    cout << "area: ";  
    cout << r1.calculateArea();  
    cout << endl;  
}
```



# The keyword “public” and “private”

- **public:** the members in the class can be accessed anywhere from the program.
- **private:** the members can only be accessed from within the class (i.e., member functions).
  - Information hiding

# Example of using private

```
class Room {  
    private:  
        double length;  
        double breadth;  
        double height;  
    public:  
        void initData(double len, double brth, double hgt) {  
            length = len;  
            breadth = brth;  
            height = hgt;  
        }  
        double calculateArea(){  
            return length * breadth;  
        }  
        double calculateVolume(){  
            return length * breadth * height;  
        }  
};
```

# Example of using private

```
class Room {  
    private:  
        double length;  
        double breadth;  
        double height;  
    public:  
        void initData(double len, double brth,  
            length = len;  
            breadth = brth;  
            height = hgt;  
        }  
        double calculateArea(){  
            return length * breadth;  
        }  
        double calculateVolume(){  
            return length * breadth * height;  
        }  
};
```

```
int main() {  
    // create objects of Room  
    Room r1;  
  
    //initial the Room object  
    r1.initData(42.5, 30.8, 19.2);  
  
    //calculate the area  
    cout << "area: ";  
    cout << r1.calculateArea();  
    cout << endl;  
  
    cout << "volume: ";  
    cout << r1.calculateVolume();  
    cout << endl;  
}
```

# Example: Animal

```
class Animals {  
    public:  
        void sound() {  
            cout << "Animals' sound."  
                << endl;  
        }  
};
```

```
int main () {  
    Animals dog, cat;  
    dog.sound();  
    cat.sound();  
    return 0;  
}
```

# A Boring Example

```
class Homework {
    public:
        void DO();
};
void Homework::DO() {
    // need to "#include <utility>" for swap
    string s1;
    getline(cin, s1);
    auto len {s1.length()};
    string s2 {s1};
    for (unsigned int i=0; i <len/2; i++)
        swap(s2.at(i), s2.at(len-i-1));
    cout << s2 << endl;
}
```

```
int main () {
    Homework h1;
    h1.DO();
    return 0;
}
```

<https://onlinegdb.com/K3lwpNE8T>





Learn from a well-designed class in a header file.

# The Bookstore transactions

- We shall learn how to “use” a class.
- Sales\_item.h [[link](#) to download]

```
#include <iostream>
#include "Sales_item.h"
int main()
{
    Sales_item book;
    // read ISBN, number of copies sold, and sales price
    std::cin >> book;
    // write ISBN, number of copies sold, total revenue, and average price
    std::cout << book << std::endl;
    return 0;
}
```

# The Bookstore transactions

- Sample input of previous code:

```
0-201-70353-X 4 24.99
```

- The sample output:

```
0-201-70353-X 4 99.96 24.99
```

# The Bookstore transactions

## - Adding Sales\_items

```
#include <iostream>
#include "Sales_item.h"
int main()
{
    Sales_item item1, item2;
    std::cin >> item1 >> item2;
    // read a pair of transactions
    std::cout << item1 + item2 << std::endl;
    // print their sum
    return 0;
}
```

Sample input of previous code:

0-201-78345-X 3 20.00

0-201-78345-X 2 25.00

The sample output:

0-201-70353-X 5 110 22

# The Bookstore transactions

## - Example of Member functions/methods

```
#include <iostream>
#include "Sales_item.h"
int main()
{
    Sales_item item1, item2;
    std::cin >> item1 >> item2;
    // first check that item1 and item2 represent the same book
    if (item1.isbn() == item2.isbn()) {
        std::cout << item1 + item2 << std::endl;
        return 0; // indicate success
    } else {
        std::cerr << "Data must refer to same ISBN"
            << std::endl;
        return -1; // indicate failure
    }
}
```

# The Bookstore Program

```
#include <iostream>
#include "Sales_item.h"
int main()
{
    Sales_item total; // variable to hold data for the next transaction
    // read the first transaction and ensure that there are data to process
    if (std::cin >> total) {
        Sales_item trans; // variable to hold the running sum
        // read and process the remaining transactions
        while (std::cin >> trans) {
            // if we're still processing the same book
            if (total.isbn() == trans.isbn())
                total += trans; // update the running total
            else {
                // print results for the previous book
                std::cout << total << std::endl;
                total = trans; // total now refers to the next book
            }
        }
        std::cout << total << std::endl; // print the last transaction
    } else {
        // no input! warn the user
        std::cerr << "No data?!" << std::endl;
        return -1; // indicate failure
    }
    return 0;
}
```

# Global Scope/Function Scope /Block Scope

Refer to: <https://en.cppreference.com/w/cpp/language/scope>  
<https://www.geeksforgeeks.org/scope-of-variables-in-c/>

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

//global
int global = 5;

int main() {
    //local variable within a function scope
    int global = 2;
    cout << global << endl;

    return 0;
}
```

Global variables can be assessed from ANY part of the program.

- Usually declared outside all of the functions or blocks.

# Global Scope/Function Scope /Block Scope

Refer to: <https://en.cppreference.com/w/cpp/language/scope>  
<https://www.geeksforgeeks.org/scope-of-variables-in-c/>

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

int number = 1;

int main ()
{
    cout << "Global number: " << number << endl;
    int number = 2;
    cout << "Local number: " << number << endl;
    {
        int number = 3;
        cout << "Block number: " << number << endl;
    }
    return 0;
}
```

```
Global number: 1
Local number: 2
Block number: 3
```

Variables defined in a block “{ }” can be seen inside that block!



# Namespace Scope

- Purpose of using namespaces: [\[reference link\]](#)
  - Organize code into logical groups.
    - Prevent name collisions (namespace pollution).
  - Suitable for teamwork in case collisions happen.

- Usage:

`using namespace::name;`

- Note :

Don't use `using` in a header file.

(potential unexpected name collisions)

```
#include <iostream>
using std::cin;

int main()
{
    int i;
    cin >> i;
    cout << i; // error!;
    std::cout << i; // OK!
    return 0;
}
```

# Namespace Examples

<https://www.geeksforgeeks.org/namespace-in-c/>

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

namespace first
{
    int val = 500;
}

int val = 100;

int main()
{
    int val = 200;

    // These variables can be accessed from outside the namespace using the scope operator ::
    cout << first::val << endl;

    return 0;
}
```

# Namespace in the same file

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

namespace NTOU {
    class Student{
    public:
        string name; int age;
        double height; double weight;
        string department;
    };
    class Professor{
    public:
        string name; int age;
        double height; double weight;
        string department;
    };
}
```

```
int main() {
    NTOU::Student s1;
    s1.age = 20;
    s1.name = "Betty";
    return 0;
}
```

# Namespace in different files

- Suppose that we have

[Codes on OnlineGDB](#)

- `main.cpp`
- Files for NTOU:
  - `ntou.h`
  - `ntou.cpp`
- Files for CMU
  - `cmu.h`
  - `cmu.cpp`

Class assignment:  
Modify the code (line 19) to have  
the following output:

```
Betty is playing in CMU!  
Betty is studying hard in CMU!
```

# Class Scope

- Every class defines its own new scope.
- Outside the class scope, ordinary **data and function members** may be accessed only through an **object**, a **reference**, or a **pointer** using a **member access operator**  
  
    . or ->
- We access **type members** from the class using the **scope operator ::**
- In either case, the name that follows the operator must be a member of the associated class.

# Class Scope (Example)

(click to the code in my GitHub page)

```
class rectangle {  
public:  
    typedef int unit;  
    void area();  
    void set(unit wd, unit ht);  
private:  
    unit width;  
    unit height;  
};
```

```
void rectangle::set(unit wd, unit ht)  
{  
    width = wd;  
    height = ht;  
}
```

```
void rectangle::area() {  
    cout << "The area: " << width * height << endl;  
}
```

```
int main()  
{  
    rectangle obj, *obj2; //creating object of rectangle class  
    rectangle::unit x, y;  
    cin >> x;  
    cin >> y;  
    obj.set(x, y);  
    obj2 = &obj;  
    obj.area();  
    obj2->area();  
    return 0;  
}
```

# Class Scope (Example)

(click to the code in my GitHub page)

<https://onlinegdb.com/NsLIANoyo>

```
class rectangle {
public:
    typedef int unit;
    void area();
    void set(unit wd, unit ht);
private:
    unit width;
    unit height;
};
```

```
void rectangle::set(unit wd, unit ht)
{
    width = wd;
    height = ht;
}
```

define the member function  
outside a class



```
void rectangle::area() {
    cout << "The area: " << width * height << endl;
}
```

```
int main()
{
    rectangle obj; //creating object of rectangle class
    rectangle::unit x, y;
    cin >> x;
    cin >> y;
    obj.set(x, y);
    rectangle &obj2 = obj; // this usage of & is special in C++
    obj.area();
    obj2.area();
    return 0;
}
```

# When a local variable has the same name as a global variable...

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

int val; // val is global

int main()
{
    int val = 10; // val is local here
    cout << "Value of global val is " << ::val;
    cout << "\nValue of local val is " << val;
    return 0;
}
```

```
Value of global val is 0
Value of local val is 10
```



# To access a static variable in a class

```
using namespace std;

class Test {
    static int x;
public:
    static int y;

    void func(int x) {
        cout << "Value of static x is " << Test::x;

        cout << "\nValue of local x is " << x;
    }
};
```

```
// static members must be explicitly
// defined
int Test::x = 1;
int Test::y = 2;

int main() {
    Test obj;
    int x = 3;
    obj.func(x);

    cout << "\nTest::y = " << Test::y;
    return 0;
}
```

```
Value of static x is 1
Value of local x is 3
Test::y = 2
```

# To access a static variable in a class

<https://onlinegdb.com/5H2zEGyWk>

```
using namespace std;

class Test {
    static int x;
public:
    static int y;

    void increase() {
        x++;
        y++;
    }
    void show() {
        cout << "x: " << x << endl;
        cout << "y: " << y << endl;
    }
};
```

```
// static members must be explicitly
// defined
int Test::x = 1;
int Test::y = 2;

int main() {
    Test obj1, obj2;
    obj1.increase();
    obj1.show();
    obj2.increase();
    obj2.show();
}
```

```
x: 2
y: 3
x: 3
y: 4
```

# To access a static variable in a class

<https://onlinegdb.com/5H2zEGyWk>

```
using namespace std;

class Test {
    static int x;
public:
    static int y;

    void increase() {
        x++;
        y++;
    }
    void show() {
        cout << "x: " << x << endl;
        cout << "y: " << y << endl;
    }
};
```

```
// static members must be explicitly
// defined
int Test::x = 1;
int Test::y = 2;

int main() {
    Test obj1, obj2;
    obj1.increase();
    obj1.increase();
    obj2.show();
    return 0;
}
```

```
x: 3
y: 4
```

# Another Example: Circle

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

class Circle
{
    private :
        double radius; // data members
    public:
        void setRadius(double r);
        double getArea(); //member functions
};

void Circle::setRadius(double r)
{
    radius = r;
}

double Circle::getArea()
{
    return 3.14 * radius * radius;
}
```

```
int main()
{
    Circle c1; //an object of class circle
    c1.setRadius(2.5); //call member function
                        // to initialize radius
    cout << c1.getArea(); //display the area
    return 0;
}
```



# Illustrating Example: Structure & Constant Member Function

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

struct Person {
    // default: everything is public
    string name;
    int age;
    double salary;
};

// structure variable as an argument here
void display(const Person&);

int main() {
    Person p {"Joseph", 45, 10000000};
    display(p);
    return 0;
}

void display(const Person& p) {
    cout << "Name: " << p.first_name << endl;
    cout << "Age: " << p.age << endl;
    cout << "Salary: " << p.salary;
}
```

Let's play around this program.

```

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

struct Person {
    // default: everything is public
    string name;
    int age;
    double salary;
    // structure variable as an argument here
    void display(const Person&) const;
};

int main() {
    Person p {"Joseph", 45, 10000000};
    p.display(p);
    return 0;
}

void Person::display(const Person& p) const {
    cout << "Name: " << p.name << endl;
    cout << "Age: " << p.age << endl;
    cout << "Salary: " << p.salary;
}

```

## Constant member function of a class

<https://onlinegdb.com/mXYp4x-kN>

```

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

struct Person {
    // default: everything is public
    string name;
    int age;
    double salary;
    // structure variable as an argument here
    void display() const;
};

int main() {
    Person p {"Joseph", 45, 10000000};
    p.display();
    return 0;
}

void Person::display() const {
    cout << "Name: " << name << endl;
    cout << "Age: " << age << endl;
    cout << "Salary: " << salary;
}

```

Another way (more concise)



```

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

struct Person {
    // default: everything is public
    string name;
    int age;
    double salary;
    // structure variable as an argument here
    void display(double salary) const;
};

int main() {
    Person p {"Joseph", 45, 10000000};
    p.display(100);
    cout << endl;
    cout << p.salary << endl;
    return 0;
}

void Person::display(double salary) const {
    this->salary += salary;
    cout << "Name: " << name << endl;
    cout << "Age: " << age << endl;
    cout << "Salary: " << salary;
}

```

## Using this pointer...

error: assignment of member 'Person::salary' in read-only object

```

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

struct Person {
    // default: everything is public
    string name;
    int age;
    double salary;
    // structure variable as an argument here
    void display(double salary) const;
};

int main() {
    Person p {"Joseph", 45, 10000000};
    p.display(100);
    cout << endl;
    cout << p.salary << endl;
    return 0;
}

void Person::display(double salary) const {
    this->salary += salary;
    cout << "Name: " << name << endl;
    cout << "Age: " << age << endl;
    cout << "Salary: " << salary;
}

```

## Using this pointer... (UPDATED VERSION)

```

Name: Joseph
Age: 45
Salary: 100
10100

```


<https://onlinegdb.com/6hNFd9zrrA>

# Exercise - BOOK Specification

Define a class **BOOK** with the following specification

- Private members:
  - **BOOK\_NO**: (int)
  - **TITLE**: 20 characters (string)
  - **PRICE**: float (price per copy)
  - **TOTAL\_COST (N)** : float (a function calculating the total cost for N copies; N is passed as argument)
- Public member functions:
  - **INPUT ()** : Function to read BOOK\_NO, TITLE, and PRICE.
  - **PURCHASE ()** : Function to ask the user to input the number of copies to be purchased. It invokes TOTAL\_COST () and prints the total cost to be paid by the user.

# Sample Input & Output

 C:\CPP\book\_purchase.exe

```
In INPUT():
Enter Book Title: 海大資工讚讚讚
Enter Book Number: 20240301
Enter price per copy: 999
In PURCHASE:
Enter number of copies to purchase: 10
Total cost: 9990

Process returned 0 (0x0)   execution time : 0.329 s
Press any key to continue.
```

The main function:


```
int main()
{
    BOOK obj;
    obj.INPUT();
    obj.PURCHASE();
    return 0;
}
```

# Another Example

Define a class **student** with the following specification

- Private members:
  - **studentID**: (int)
  - **name**: 20 characters (string)
  - **eng, math, phy**: float
  - **total**: float (the sum of eng, math, and phy)
  - **grades ()**: a function to calculate eng + math + phy with float return type.
- Public member functions:
  - **Takedata ()**: Function to accept values for **studentID**, **name**, **eng**, **math** grades (float type) and invoke **grades ()** to calculate **total**.
  - **Showdata ()**: Function to display all the data members on the screen.

# Sample Input & Output

 C:\Users\josep\\_programs\PL\students.exe

```
In Takedata():
Enter studentID: 693410001
Enter student name: 林金毛
Enter grades in English, Math, and Physics: 100 99 92
In Showdata():
StudentID: 693410001
Student Name: 林金毛
English: 100
Math: 99
Physics: 92
Total: 291
-----
Process exited after 30.34 seconds with return value 0
請按任意鍵繼續 . . .
```

The main function:


```
int main ()
{
    student obj ;
    obj.Takedata() ;
    obj.Showdata() ;
    return 0;
}
```

# Another Exercise

Define a class **student** with the following specification

- Private members:
  - **studentID**: (int)
  - **name**: 20 characters (string)
  - **eng, math, phy**: float
  - **avg ()**: a function to calculate the average grades of eng, math, and phy with float return type.
- Public member functions:
  - **Takedata ()**: Function to accept values for **studentID**, **name**, **eng**, **math** grades and invoke **avg ()** to calculate the average grade.
  - **Showdata ()**: Function to display all the data members on the screen.
  - **PassOrFail ()**: Function to display "**pass**" if **avg\_grade**  $\geq 60$  or "**fail**". otherwise.

# Sample Input & Output

 C:\Users\josep\Study\Programming Language\C++\hw\_2.exe

```
In Takedata()
Enter studentID: 693410001
Enter student name: 張大勇
Enter grades in English, Math, and Physics: 100 90 80
In Showdata()
StudentID:693410001
Student Name:張大勇
English:100
Math:90
Physics:80
Pass

-----
Process exited after 19.95 seconds with return value 0
請按任意鍵繼續 . . .
```

The main function:

```
int main ()
{
    student obj ;
    obj.Takedata() ;
    obj.Showdata() ;
    obj.PassOrFail() ;
    return 0;
}
```