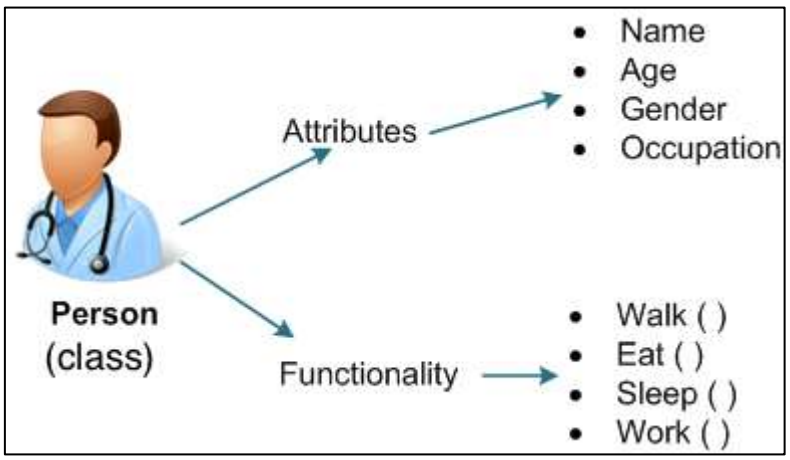
หน่วยที่ 4

คลาสและอ็อบเจ็กต์ (Class and Object)



หัวข้อเรื่อง

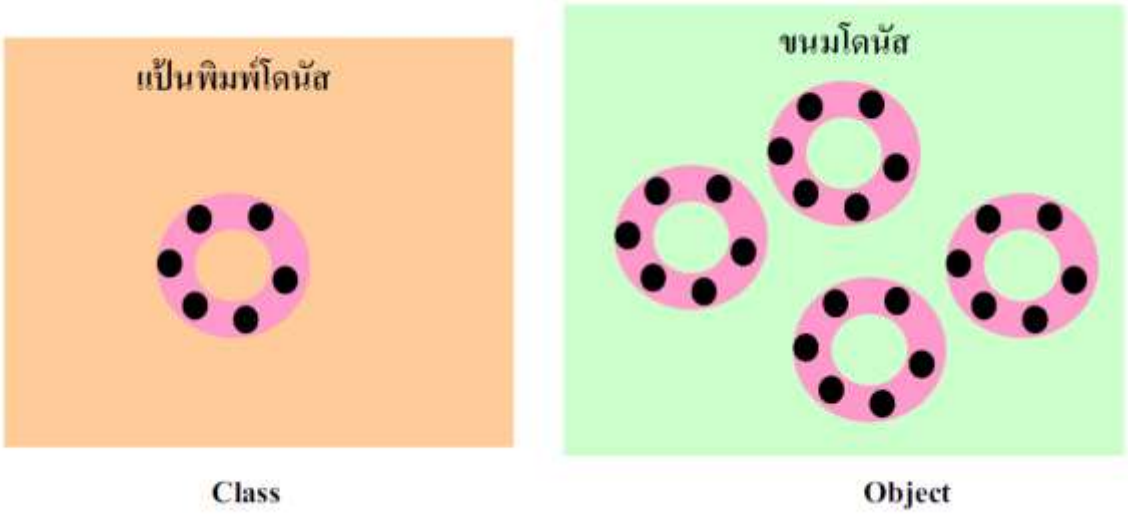
4.1 คลาสและอ็อบเจ็กต์

4.2 การสร้างคลาส

4.3 การห่อหุ้ม

4.1 คลาสและอ็อบเจ็กต์ (Classes and Object)

Class คือ สิ่งที่ใช้อธิบายลักษณะและความสามารถของวัตถุ (Object) Class อาจจะเปรียบได้กับพิมพ์เขียวหรือแบบแปลนของวัตถุ ดังนั้นสามารถสร้างวัตถุ ให้มีคุณลักษณะเหมือนกับแบบพิมพ์เขียวหรือแบบแปลนได้ในจำนวนมาก เช่น การที่สามารถทำขนมขึ้นมาได้หลาย ๆ ขึ้น จากแบบพิมพ์เดียวกัน ดังรูปที่ 5-1



รูปที่ 5-1 แสดงความสัมพันธ์ระหว่าง Class และ Object

Object คือ สิ่งต่าง ๆ ที่ปรากฏอยู่รอบ ๆ ตัว ภายใต้โลกแห่งความเป็นจริง (Real World) ที่อาศัยอยู่นี้ ซึ่งอาจเป็นได้ทั้งสิ่งที่จับต้องได้และจับต้องไม่ได้ (Tangible) ซึ่งจะสอดคล้องกับสิ่งที่มีลักษณะทางกายภาพ (Physical Things) เช่น ปากกา, เครื่องคอมพิวเตอร์, คน, รถยนต์ เป็นต้นอยู่ในลักษณะที่จับต้องไม่ได้(Intangible Things) ที่สามารถแบ่งออกได้ดังนี้

บทบาท (Role) ของคนหรือองค์กร เช่น ผู้บริหาร, ผู้ป่วย, พนักงาน, ลูกค้า เป็นต้น

สิ่งต่าง ๆ ที่เกิดขึ้น (Incident) หรือเหตุการณ์ต่าง ๆ (Event) เช่น เที่ยวบิน (Flight), อุบัติเหตุ (Accident) เป็นต้น

การโต้ตอบ (Interactions) ซึ่งลักษณะของ Object เหล่านี้จะมีการทำธุรกรรม (Transaction) หรือ การติดต่อกัน (Concept) ระหว่าง Object ตั้งแต่ 2 Object ขึ้นไปภายในแบบจำลองระบบ (System Mode) ที่พัฒนา เช่น การซื้อ (Purchase) จะมีความสัมพันธ์ระหว่างผู้ซื้อ, ผู้ขายและสิ่งที่ซื้อ และการแต่งงานซึ่งจะมีความสัมพันธ์ระหว่างผู้ชายกับผู้หญิง

โดยทั่วไปแล้ว Object จะอยู่ในลักษณะที่สอดคล้องกับ "คำนาม" ที่มีในภาษาที่สื่อสารกันประจำวัน (Natural Language) ซึ่งจะมี 3 องค์ประกอบ คือ

4.1.1 คุณลักษณะเฉพาะ (Identity) เช่น ชื่อของวัตถุ (Object)

4.1.2 สถานะ (state) ค่าของคุณสมบัติ (attributes) จะเป็นตัวกำหนดสถานะให้กับวัตถ (Object)

4.1.3 ความสามารถ (operations / method / behavior) วิธีการหรือพฤติกรรมของวัตถุ (Object) แต่ละตัวจะเป็นตัวเปลี่ยนสถานะของวัตถุ (Object) นั้น ๆ ได้

การเขียนโปรแกรมเชิงวัตถุในภาษา C++

ในภาษา C++ ข้อมูลที่บ่งบอกถึงคุณลักษณะคลาสที่เรียกว่า แอตทริบิวต์ (Attribute) นั้นเรียกว่า data member ส่วนพฤติกรรมหรือวิธีการ (method) ของคลาสเรียกว่า member function

4.1.1 การสร้างคลาส (class) ในภาษา C++

คลาส (Class) ก็คือ การรวมเอาคุณลักษณะ (Attribute) และวิธีการ (method) เช่น ข้อมูล ตัวแปรหรือฟังก์ชันของวัตถุนั้นมารวมไว้ในกลุ่ม ๆ เดียวกัน โดยมีการกำหนดสิทธิ์ในการอ้างถึงข้อมูล คือ private, protected และ public

4.1.2 การประกาศคลาส

คลาสเปรียบเสมือนแม่พิมพ์ วัตถุเป็นผลิตผลที่เกิดจากแม่พิมพ์ ดังนั้น การที่จะสร้างวัตถุได้จึงจำเป็นต้องอาศัยแม่พิมพ์หรือคลาสนี้ สำหรับการประกาศคลาสเริ่มต้นด้วยคำหลัก Class ตามด้วยชื่อของ Class กำหนดขอบเขตด้วย {} และจบด้วยเครื่องหมาย เซมิโคลอน (;)

รูปแบบคำสั่ง

```
class class_name {
        private :
        data member;
        member function();
        protected :
        data member;
        member function();
        public :
        member function();
        } object_names;
```

มีรายละเอียดดังนี้

private ดาต้าเมมเบอร์หรือเมมเบอร์ฟังก์ชัน ที่ถูกประกาศแบบ private จะถูกจำกัดการใช้งานได้เฉพาะภายในขอบเขตคลาสเท่านั้น ไม่สามารถนำดาดำเมมเบอร์หรือเมมเบอร์ฟังก์ชันไปใช้ภายนอกขอบเขต

protected ดาต้าเมมเบอร์หรือเมมเบอร์ฟังก์ชัน ที่ถูกประกาศเข้าถึงอยู่ใน protected จะถูกจำกัดการใช้งานได้เฉพาะภายในขอบเขตคลาสและคลาสสืบทอด

public ดาต้าเมมเบอร์หรือเมมเบอร์ฟังก์ชัน ที่ถูกประกาศเข้าถึงอยู่ใน public จะสามารถถูกเรียกใช้ได้ทุกที่กายในโปรแกรม ถ้าเรียกภายนอกขอบเขตคลาสต้องเรียกผ่านอ็อบเจ็กต์

data member คือสมาชิกคลาส ดาต้าเมมเบอร์ส่วนใหญ่จะเป็นข้อมูลชนิดพื้นฐาน เช่น double, int, float, bool หรือ พอยน์เตอร์ที่ชี้ไปยังข้อมูลชนิดต่าง ๆ เป็นต้น นอกจากนี้ก็จะเป็นข้อมูลชนิดที่สร้างขึ้นใหม่ เช่น string คลาสที่สร้างขึ้นใหม่ เป็นต้น บางครั้งจะมีการใช้คำสงวน const และ static วางหน้าดาต้าเมมเบอร์

member function คือ สมาชิกหนึ่งของคลาส เมมเบอร์ฟังก์ชันส่วนใหญ่จะเป็นคอนสตรัคเตอร์ (Constructor) ดีสตรัคเตอร์ (Destructor) โอเวอร์โหลดโอเปอร์เรเตอร์ (Overload Operator) สแตติกเมมเบอร์ฟังก์ชัน (Static Member Function) และ ฟังก์ชัน (Function) ต่าง ๆ เป็นต้น

ตัวอย่างที่ 4.1 โปรแกรมคำนวณหาพื้นที่สีเหลี่ยม

```
#include <iostream>
using namespace std;
    class CRectangle {
        int x, y;
    public:
        void set_values (int,int);
        int area ()
        {return (x*y);}
    };
    void CRectangle::set_values (int a, int b) {
        x = a;
        y = b;
    }
```

```
int main () {
        CRectangle rect;
        rect.set_values (3,4);
        cout << "area: " << rect.area();
        return 0;
}
```

ผลลัพธ์

area :12

จากตัวอย่างที่ 4.1 คลาสจะมีชื่อ CRectangle และได้สร้างอ็อบเจ็กต์ rect ประกอบด้วยดาต้าเมมเบอร์คือ x และ y กำหนดสิทธิการเข้าถึงแบบ private (โดยปกติถ้าไม่ได้กำหนดการเข้าถึงถือว่าเป็น private) และมีเมมเบอร์ฟังก์ชัน set_values() และ area() กำหนดสิทธิแบบ public และกำหนดเมมเบอร์ฟังก์ชัน area อยู่ภายในคลาส ส่วนเมมเบอร์ฟังก์ชัน set_values จะมีการกำหนดรายละเอียดการทำงานภายนอกคลาส ในภาษา C++ จะมีรูปแบบ type_data class_name :: member_function_name สำหรับเมมเบอร์ฟังก์ชันที่อยู่นอกคลาส การเรียกใช้งานเมมเบอร์ฟังก์ชันมีรูปแบบดังนี้ อ็อบเจ็กต์.เมมเบอร์ฟังก์ชัน ในตัวอย่างคือ rect.set_values (3,4) และ rect.area()

4.2 คอนสตรัคเตอร์และดีสตรัคเตอร์ (Constructors and Destructor)

คอนสตรัคเตอร์ (Constructor) เป็นเมมเบอร์ฟังก์ชันพิเศษ ที่ใช้ในการกำหนดค่าเริ่มต้นแก่ดาต้าเมมเบอร์ทั้งหลาย ถ้าไม่ได้สร้างคอนสตรัคเตอร์ไว้ในคลาส เมื่อมีการคอมไพล์ซอร์สโค้ด ตัวคอมไพล์ก็จะสร้างให้โดยอัตโนมัติ ซึ่งเรียกว่าดีฟอลต์คอนตรัคเตอร์ และคอนสตรัคเตอร์จะถูกเรียกใช้ทุกครั้งที่มีการสร้างอ็อบเจ็กต์ใหม่

จากข้อกำหนดที่กล่าวมา สามารถเขียนรูปแบบการสร้างคอนสตรัคเตอร์ได้ดังนี้

รูปแบบคำสั่ง constructor

```
class class_name {
        private :
                data member;
        public :
                class_name ( ); //constructor
        member function();
```

```
        } object_names;
```

ตัวอย่างที่ 4.2 โปรแกรมบวกเลขและนับจำนวนเลขที่มาบวกเข้าด้วยกัน

```
#include <iostream>
using namespace std;
class sum
{
    int total;
    int counter;
public:
    sum( )
    {       total=0;
            counter=0;
    }
    void addsum(int temp)
    {
            total+=temp;
            counter++;
    }
    void showdata()
    {
    cout<<"total= "<<total<<" counter ="<<counter;
            }
};
    int main(){
            sum s1;
            s1.addsum(5);
            s1.addsum(15);
    s1.showdata();
    return 0;
```

```
}
```

ผลลัพธ์

```
total= 20 counter=2
```

คอนสตรัคเตอร์ที่ประกอบด้วยพารามิเตอร์ (Constructors with One Parameter) การกำหนดค่าคอนสตรัคเตอร์ทีมีพารามิเตอร์นั้น สามารถทำได้ดังนี้ โดยไปกำหนดค่าให้กับอ็อบเจ็กต์เมื่อเริ่มต้นสร้างอ็อบเจ็กต์ เมื่อมีการใช้งานอ็อบเจ็กต์ คอนสตรักเตอร์จะทำการรับข้อมูลจากอ็อบเจ็กต์และสงข้อมูลไปยังดาต้า เมมเบอร์เพื่อกำหนดค่าเริ่มต้น ต่อจากนั้นเมมเบอร์ฟังก์ชันจึงนำค่าเหล่านี้มาประมวลผลเพื่อแสดงผลที่ได้

ดีสตรัคเตอร์ (Destructor) เมื่อมีการสร้างอบเจ็กต์ คอนสตรักเตอร์จะถูกสร้างโดยอัตโนมัติ เมื่ออ็อบเจ็กต์ที่ถูกประกาศไว้หมดอายุการใช้งาน เมมเบอร์ฟังก์ชันเฉพาะกิจ จะถูกเรียกใช้โดยอัตโนมัติเพื่อจัดการทำลายคอนสตรักเตอร์ เมมเบอร์ฟังก์ชันนี้เรียกว่า ดีสตรักเตอร์ (Destructor)

เช่นเดียวกับคอนสตรัคเตอร์ เมื่อไม่ได้กำหนดดีสตรัคเตอร์ ตัวคอมไพล์จะสร้างดีฟอลต์ดีสตรัคเตอร์ โดยอัตโนมัติ ดีสตรัคเตอร์สามารถสร้างขึ้นมาได้เช่นเดียวกับคอนสตรักเตอร์ โดยปกติ การสร้างดีสตรัคเตอร์นั้นเพื่อจัดการทำลายคอนสตรักเตอร์ ที่สร้างขึ้น

ข้อกำหนดในการสร้างดีสตรัคเตอร์ มีดังต่อไปนี้

1. ดีสตรัคเตอร์ใช้ชื่อเดียวกับคลาส และมีเครื่องหมาย ~ (Tilde) นำหน้าชื่อดีสตรัคเตอร์
2. ไม่มีการคืนค่า แม้แต่ void ก็ไม่ได้ และไม่มีพารามิเตอร์
3. ไม่สามารถใช้คำสงวนต่าง ๆ นำหน้าชื่อดีสตรัคเตอร์ ได้แก่ const, volatile และ static แต่สามารถเป็น virtual ถ้าต้องการทำคลาสสืบทอดจะต้องถูกประกาศเข้าถึง แบบ pubic

จากข้อกำหนดที่กล่าวมา สามรถเขียนรูปแบบการสร้างดีสตรัคเตอร์ได้ดังนี้

```
//ภายในคลาส
class Class_name{
        ...
public
        ~Class_name() {
        ...
        }
};
```

ตัวอย่างที่ 4.3 การใช้คอนสตรัคเตอร์ ดีสตรัคเตอร์ ในคำนวณหาขนาดของพื้นที่สี่เหลี่ยม

```
#include <iostream>
using namespace std;
class CRectangle {
        int width, height;
    public:
        CRectangle (int,int);
        ~CRectangle () {
        cout << endl << "\nGood-Bye (from the destructor function)";
            };
        int area () {return (width * height);}
    };
CRectangle::CRectangle (int a, int b) {
    width = a;
    height = b;
}
int main () {
    CRectangle rect (3,4), rectb (5,6);
    cout << "rect area: " << rect.area() << endl;
    cout << "rectb area: " << rectb.area() << endl;
    return 0;
    }
```

ผลลัพธ์

```
rect area : 12
rectb area : 30
Good-Bye (from the destructor function)
Good-Bye (from the destructor function)
```

4.3 โอเวอร์โหลดคอนสตรัคเตอร์ (Overloaded Constructors)

คือ วิธีการที่สามารถสร้างคอนสตรัคเตอร์ ได้หลาย ๆ ตัวโดยทีเยอมให้มีการใช้ชื่อเหมือนกัน (แต่พารามิเตอร์ทีคอนสตรัคเตอร์ที่ไว้รับค่าจากการเรียกใช้คอนสตรัคเตอร์นั้นแตกต่างกัน) เรียกวิธีการนี้ว่า Overloading

ตัวอย่างที 5.4 การใช้โอเวอร์โหลดคอนสตรัคเตอร์ในการคำนวณเกียวกับความยาว

```
1   #include <iostream>
2   using namespace std;
3   class Distance {
4           private:
5                   int feet;
6                   float inches;
7           public:
8                   Distance(){ }
9                   Distance(int ft, float in){
10                          feet = ft; inches = in;
11          }
12          void getdist(){
13          cout << "\nEnter feet: "; cin >> feet;
14          cout << "Enter inches: "; cin >> inches;
15          }
16          void showdist(){
17                  cout << feet <<"'-" << inches << "\""<<endl;
18                  }
19          void add_dist( Distance, Distance );
20          };
21          void Distance::add_dist(Distance d2, Distance d3)
22          {
23                  inches = d2.inches + d3.inches;
```

```
24          feet = 0;
25          if(inches >= 12.0)
26                  {
27                  inches -= 12.0;
28                  feet++;
29                  }
30                  feet += d2.feet + d3.feet;
31                  }
32                  int main()
33                          {
34                          Distance dist1, dist3;
35                          Distance dist2(11, 6.25);
36                          dist1.getdist();
37                          dist3.add_dist(dist1, dist2);
38                          cout << "\ndist1 = "; dist1.showdist();
39                          cout << "\ndist2 = "; dist2.showdist();
40                          cout << "\ndist3 = "; dist3.showdist();
41                          cout << endl;
42                          return 0;
43                          }
```

ผลลัพธ์

```
Enter feet :10
Enter inches :8
dist1= 10'-8"
dist2= 11'-6.25"
dist3= 22'-2.25"
```

อธิบายการทำงานของโปรแกรม

บรรทัดที่	คำอธิบาย
3	คลาสชื่อ Distance
8	ฟังก์ชัน Distance ทีเป็น constructor ทีไม่มีพารามิเตอร์
9	ฟังก์ชัน Distance ทีเป็น constructor ทีมีพารามิเตอร์ 2 ตัว
12	ฟังก์ชัน getdist รับข้อมูลจากแป้ นพิมพ์
16	ฟังก์ชัน showdist แสดงความยาวเป็นฟุต และนิOว
19	ฟังก์ชัน add_dist ใช้บวกความยาว 2 จำนวนเข้าด้วยกัน
22	จากบรรทัด 19 ให้บวกความยาวแเป็นนิOวเข้าด้วยกันก่อน
25	ใช้คำสังI if ตรวจสอบว่าเกิน 12 นิOว ก็ให้ปัดขึOนไปเป็นฟุต
30	บวกความยาวแเป็นฟุตเข้าด้วยกันรวมทัOงทีปัดขึOนไปเป็นฟุต
34	สร้างอ็อบเจ็กต์ dist1 , dist3 จากคลาส Distance
35	สร้างอ็อบเจ็กต์ dist2 พร้อมกำหนดความยาว 11'-6.25"
36	เรียกใช้ฟังก์ชัน getdist รับข้อมูลจากผู้ใช้ไปไว้ที อ็อบเจ็กต์ dist1
37	เรียกใช้ฟังก์ชัน add_dist ทำการบวก อ็อบเจ็กต์ของ dist1 และ dist2 ไปเก็บที dist3
38-40	เรียกใช้ฟังก์ชัน showdist แสดงความยาวของอ็อบเจ็กต์ของ dist1, dist2 และ dist3

4.4 สแตติกดาต้าเมมเบอร์และสแตติกเมมเบอร์ฟังก์ชัน (Static Data Member and Static Member Function)

ปกติการกำหนดค่าเริ่มต้นให้กับดาต้าเมมเบอร์จะใช้คอนสตรัคเตอร์แล้วยังสามารถใช้คำสงวน static ในการกำหนดค่าเริ่มต้นได้ด้วย

การใช้คำสงวน static วางไว้หน้าดาต้าเมมเบอร์ เมมเบอร์ฟังก์ชัน จะเป็นการกำหนดให้ค่าเหล่านี้คงอยู่ในหน่วยความจำ จะไม่ถูกยกเลิกเหมือนกับการกำหนดแบบปกติ และจะอยู่ในหน่วยความจำ จนกว่าจะเลิกใช้โปรแกรม และตัวแปลภาษาจะกำหนดค่าให้เป็น 0 โดยอัตโนมัติ

สแตติกดาต้าเมมเบอร์ (Static Data Member)

คือดาต้าเมมเบอร์ทีใช้คำสงวน static วางหน้าการประกาศดาต้าเมมเบอร์ เมื่อเป็นสแตติกดาต้าเมมเบอร์แล้ว สแตติกดาต้าเมมเบอร์นั้น สามารถใช้ร่วมกับอ็อบเจ็กต์อื่น ๆ ที่สร้างจากคลาสเดียวกันได้อีกด้วย หรือกล่าวอีกลักษณะหนึ่งว่า ถ้าคลาสหนึ่งมีสแตติกดาต้าเมมเบอร์แล้ว และอ็อบเจ็กต์ต่าง ๆ ที่สร้างจากคลาสนั้น จะมีสแตติกดาต้าเมมเบอร์เพียงตัวเดียว ขณะที่ดาต้าเมมเบอร์อื่น ๆ ก็จะถูกสร้างตามจำนวนอ็อบเจ็กต์

สแตติกดาต้าเมมเบอร์ สามารถถูกประกาศการเข้าถึงได้ทั้งแบบ private, protected, และ public เฉพาะ public เท่านั้น ที่สแตติกดาต้าเมมเบอร์จะทำหน้าที่เหมือนกับตัวแปรโกลบอล ฉะนั้น สามารถเรียกใช้งานโดยใช้ชื่อคลาสร่วมกับโอเปอร์เรเตอร์สโคป โดยไม่จำเป็นต้องสร้างอ็อบเจ็กต์ แล้วใช้อ็อบเจ็กต์เรียกใช้งาน

ส่วนสแตติกดาต้าเมมเบอร์ที่ถูกประกาศการเข้าถึงเป็นแบบ protected และ private จะถูกจำกัดการใช้งานภายในขอบเขตคลาส กรณีนี้เมื่อใช้ภายนอกขอบเขตคลาสจะต้องเรียกผ่าน สแตติกเมมเบอร์ฟังก์ชันเท่านั้น หรือเมมเบอร์ฟังก์ชันธรรมดา แต่ต้องเรียกผ่านอ็อบเจ็กต์ (ต้องสร้างอ็อบเจ็กต์ แล้วจึงใช้ชื่อออ็อบเจ็กต์เรียกใช้งาน)

เมื่อสร้างสแตติกดาต้าเมมเบอร์แล้ว ต่อไปจะต้องกำหนดค่าเริ่มต้นทุกครั้ง และต้องกำหนดไว้ภายนอกคลาสเท่านั้น แต่ยังอยู่ในขอบเขตของคลาส

ตัวอย่างที่ 4.5 การใช้งานสแตติกดาต้าเมมเบอร์

```
1   #include <iostream>
2   using namespace std;
3   class shared {
4           static int a;
5           int b;
6   public:
7           void set(int i, int j) {a=i; b=j;}
8           void show();
9   } ;
10  int shared::a;
11  void shared::show(){
12          cout << "This is static a: " << a;
13          cout << "\nThis is non-static b: " << b;
14          cout << "\n";
15  }
16  int main()
17  {
18          shared x, y;
```

```
19          x.show();
20          y.show();
21          x.set(1, 1);
22          x.show();
23          y.set(2,2);
24          y.show();
25          return 0;
26   }
```

ผลลัพธ์

```
This is static a:0
This is non-static b:0
This is static a:0
This is non-static b:2147307520
This is static a:1
This is non-static b:1
This is static a:2
This is non-static b:2
```

อธิบายการทำงานของโปรแกรม

บรรทัดที่	คำอธิบาย
3	คลาสชื่อ shared
4	a เป็นตัวแปรชนิด int และกำหนดให้เป็น static
7	ฟังก์ชัน set กำหนดตัวแปรเพื่อใช้งาน
8	ฟังก์ชัน show แสดงผลตัวแปร a และ b
10	กำหนดการใช้งานตัวแปร a เนื่องจากเป็น static
18	สร้างอ็อบเจ็กต์ x , y จากคลาส shared
19	เรียกใช้ฟังก์ชัน show แสดงค่าตัวแปร a และ b ผ่านอ็อบเจ็กต์ของ x
20	เรียกใช้ฟังก์ชัน show แสดงค่าตัวแปร a และ b ผ่านอ็อบเจ็กต์ของ y
21	เรียกใช้ฟังก์ชัน set เพื่อกำหนดค่าให้ตัวแปรตัวแปร a และ b ผ่านอ็อบเจ็กต์ของ x

22	เรียกใช้ฟังก์ชัน show แสดงค่าตัวแปร a และ b ผ่านอ็อบเจ็กต์ของ x อีกครั้ง
23	เรียกใช้ฟังก์ชัน set เพื่อกำหนดค่าให้ตัวแปรตัวแปร a และ b ผ่านอ็อบเจ็กต์ของ y
24	เรียกใช้ฟังก์ชัน show แสดงค่าตัวแปร a และ b ผ่านอ็อบเจ็กต์ของ y อีกครั0ง

สแตติกเมมเบอร์ฟังก์ชัน (Static Member Function)

คือเมมเบอร์ฟังก์ชันที่ใช้คำสงวน static วางหน้าการประกาศเมมเบอร์ฟังก์ชัน เมื่อเป็นสแตติกเมมเบอร์ฟังก์ชันแล้ว สแตติกเมมเบอร์ฟังก์ชันนั้นก็จะกลายเป็นฟังก์ชันโกลบอล ถ้าถูกประกาศการเข้าถึงเป็นแบบ public สแตติกเมมเบอร์ฟังก์ชันไม่เพียงแต่จะถูกเรียกใช้เหมือนกับเรียกเมมเบอร์ฟังก์ชันทั่ว ๆ ไป แต่ยังสามารถเรียกใช้ได้โดยตรงด้วยการใช้ชื่อคลาสร่วมกับโอเปอร์เรเตอร์ :: ได้อีกด้วย

ตัวอย่างที่ 4.6 การใช้สแตติกเมมเบอร์ฟังก์ชัน

```
1   #include <iostream>
2   using namespace std;
3   class static_type {
4           static int i;
5   public:
6           static void init(int x) {i = x;}
7           void show() {cout << i <<endl;}
8   };
9   int static_type::i;
10  int main()
11  {
12          static_type x;
13          x.int(200);
14          x.show();
15          static_type::init(100);
16          x.show();
17          return 0;
18          }
```

ผลลัพธ์

200

100

อธิบายการทำงานของโปรแกรม

บรรทัดที่	คำอธิบาย
3	คลาสชื่อ static_type
4	i เป็นตัวแปรชนิด int และกำหนดให้เป็น static
6	init เป็นฟังก์ชัน กำหนดให้เป็นแบบ static
9	กำหนดการใช้งานตัวแปร i ก่อนใช้งานจากเนืองจากเป็น static
13	เรียกใช้ฟังก์ชัน init และส่งค่า 200 ไปให้ตัวแปร i ผ่านอ็อบเจ็กต์ของ x
14	เรียกใช้ฟังก์ชัน show แสดงค่าตัวแปร i ผ่านอ็อบเจ็กต์ของ x
15	เรียกใช้ฟังก์ชัน init โดยใช้คลาสเรียกโดยตรงพร้อมส่งค่า 100 ไปให้ตัวแปร i
16	เรียกใช้ฟังก์ชัน show แสดงค่าตัวแปร i ผ่านอ็อบเจ็กต์ของ x

สรุป

คลาสคือโครงสร้างของอ็อบเจ็กต์ หรือต้นแบบของวัตถุ โดยคลาสจะเป็นตัวกำหนดว่า อ็อบเจ็กต์นั้นจะมีข้อมูล (data member) และฟังก์ชัน (member function) อะไร

การเข้าถึงข้อมูลและฟังก์ชัน ในคลาส จะถูกควบคุมระดับการเข้าถึง ด้วยคำสั่ง private, protected, public

คอนสตรัคเตอร์และดีสตรัคเตอร์ เป็นฟังก์ชันที่มีชื่อเหมือนคลาส จะทำงานอัตโนมัติ เมื่อมีการสร้างอ็อบเจ็กต์จากคลาส กรณีที่มีฟังก์ชันคอนสตรัคเตอร์หลายฟังก์ชัน เรียกว่าโอเวอร์โหลดคอนสตรัคเตอร์ (Overloaded Constructors)

การใช้คำสงวน static มาวางหน้า ดาต้าเมมเบอร์หรือเมมเบอร์ฟังก์ชัน จะเป็นเสมือนการกำหนดค่าเหล่านี้ เป็นข้อมูลแบบโกลบอล ซึ่งมีประโยชนในการที่ต้องการใช้ตัวแปรแบบโกลบอล ในการเขียนโปรแกรม