บทที่ 11
การเขียนโปรแกรมแบบ OOP

♦ วิวัฒนาการของเทคนิคการเขียนโปรแกรม

เมื่อมีการพัฒนาเครื่องคอมพิวเตอร์ขึ้นในช่วงประมาณปี 1950 โปรแกรมเมอร์ในยุคแรกสุดใช้วิธีการเขียนโปรแกรมโดยการโยกสวิตช์ไฟฟ้าที่มีอยู่จำนวนมากที่แผงวงจรไฟฟ้า เพื่อสลับไปมาระหว่างการปิดและการเปิด เพื่อให้เกิดรหัสคำสั่ง 0 และ 1 โปรแกรมเมอร์ต้องใช้เวลา ความรอบคอบและความอดทนเป็นอย่างยิ่ง ต่อมาได้มีการคิดค้นพัฒนาวิธีการเขียนโปรแกรมให้สะดวกและรวดเร็วขึ้น จึงทำให้เกิดภาษาระดับต่ำขึ้น คือ ภาษาแอสเซมบลี้ (Assembly) และในเวลาต่อมาก็เกิดภาษาระดับสูงภาษาแรกขึ้นคือ ฟอร์แทน (Fortran)

ภาษาระดับสูง หมายถึงภาษาที่มีการกำหนดชนิดข้อมูล มีการควบคุมลำดับการทำงานอย่างเป็นระบบ เขียนรหัสด้วยภาษาอังกฤษ มีการใช้คำและรูปแบบคำสั่ง ที่ใกล้เคียงกับภาษาที่ใช้ในการสื่อสารทั่วไป ภาษาระดับสูงจึงทำให้งานเขียนโปรแกรมของโปรแกรมเมอร์ง่ายขึ้น คนทั่วไปก็สามารถศึกษาวิธีการเขียนโปรแกรมนี้ได้

ปัญหาที่เกิดขึ้นในสมัยนั้นคือ การเขียนโปรแกรมที่มีความซับซ้อนและมีความยาวมาก ภาษาคอมพิวเตอร์ที่มีอยู่เมื่อนำมาเขียนโปรแกรมทำให้เกิด**ความสับสนและวกวน (spaghetti code)** ซึ่งมีผลทำให้การติดตามแก้ไขและปรับปรุงโปรแกรมยุ่งยาก ใช้เวลามากเกินไป เช่น การเขียนโปรแกรมด้วยภาษา BASIC เป็นต้น จึงทำให้มีการคิดค้นวิธีการแก้ปัญหาดังกล่าว ด้วยเหตุนี้จึงทำให้เกิดภาษาคอมพิวเตอร์ที่ใช้เทคนิคการเขียนที่เรียกว่า **Structured Programming** โดยวิธีการนี้ ผู้เขียนจะสนใจว่าโปรแกรมทำงานอะไรมากกว่าการที่จะสนใจว่าโปรแกรมนั้นทำงานอย่างไร หมายถึงว่าต้องคำนึงถึงก่อนว่าในโปรแกรมมีการทำงานต่างๆ อะไรบ้าง มีการจัดแบ่งการทำงานของโปรแกรมเป็นส่วน ๆ ตามประเภทของงาน จึงทำให้ผู้เขียนโปรแกรมสามารถแบ่งงานกันทำ สามารถนำงานย่อย ๆ ที่มีอยู่แล้วไปใช้ในโปรแกรมอื่น ๆ อีกโดยไม่ต้องสร้างใหม่ เช่น การเขียนโปรแกรมด้วยภาษาระดับสูงต่าง ๆ ได้แก่ PASCAL, COBOL, TURBO, BASIC Cเป็นต้น

ในปัจจุบัน ถึงแม้ว่าวิธีการของ Structure Programming จะเพิ่มประสิทธิภาพของการเขียนโปรแกรมได้ดีขึ้น แต่ตลอดเวลาที่ผ่านมามีการพัฒนาขีดความสามารถของฮาร์ดแวร์คอมพิวเตอร์อย่างไม่มีข้อจำกัด จึงทำให้การเขียนโปรแกรมหรือซอฟต์แวร์ต้องมีการพัฒนาควบคู่ไปด้วย ลักษณะของโปรแกรมสมัยปัจจุบัน มีขนาดยาวมาก เป็นล้าน ๆ บรรทัด มีความซับซ้อนยุ่งยาก ต้องใช้ทีมงานนักโปรแกรมเมอร์ในการพัฒนาซอฟต์แวร์เป็นจำนวนมาก จึงทำให้มีการคิดค้นวิธีการเขียนโปรแกรมแบบใหม่ ที่มีประสิทธิภาพยิ่งขึ้น เหมาะกับการพัฒนาซอฟต์แวร์ขนาดใหญ่ วิธีดังกล่าวได้แก่ วิธีที่เรียกว่า

การโปรแกรมแบบกำหนดวัตถุเป้าหมาย หรือโอโอพี OOP (Object Oriented Programming) ซึ่งผู้เชี่ยวชาญทางด้านโปรแกรมกล่าวว่า โปรแกรมที่มีขนาดตั้งแต่ 100,000 บรรทัดขึ้นไป วิธีการของ OOP จะช่วยลดความซ้ำซ้อนของโปรแกรมได้อย่างมีประสิทธิภาพ

◆ ความหมายของ OOP

OOP (Object Oriented Programming) เป็นวิธีการเขียนโปรแกรมซึ่งจัดดำเนินการกับกลุ่มของ **ออบเจ็กต์ (Object)** ที่มีอยู่ในโปรแกรม

ออบเจ็กต์ เป็นชนิดของข้อมูลซึ่งประกอบไปด้วยกลุ่มของ **ข้อมูล(data)** และกลุ่มของ **ฟังก์ชัน(Function)** โดยการใช้ข้อมูลและฟังก์ชันเหล่านี้ แต่ละออบเจ็กต์จะทำงาน 1 งานได้สมบูรณ์ (ทั้งนี้เนื่องจากตัวออบเจ็กต์เองประกอบไปด้วยข้อมูลและฟังก์ชัน)

ออบเจ็กต์ เป็นสมาชิกของ **ตัวแปรคลาส (class variable)** มีลักษณะเป็นโมดูล (modularity) ซึ่งประกอบไปด้วย **ตัวแปร** ชนิดต่าง ๆ ที่สัมพันธ์กัน และประกอบด้วย **ฟังก์ชัน** ต่าง ๆ โดยที่ **คลาส (class)** จะห่อหุ้มข้อมูลและฟังก์ชันรวมไว้ด้วยกันมีลักษณะที่เรียกว่า **encapsulation** ดังนั้นจึงมีความสะดวกในการใช้งาน สามารถป้องกันส่วนอื่น ๆ ของโปรแกรมไม่ให้เข้าถึงตัวแปรชนิดโลคอลภายในคลาสได้อย่างดีเยี่ยม

ดังนั้น การเขียนโปรแกรมแบบ OOP คือ การสร้างและ/หรือการเรียกใช้ออบเจ็กต์ให้ทำงานตามที่เราต้องการ ในการเรียกใช้ออบเจ็กต์นั้น เราจะสนใจเฉพาะการทำงานของออบเจ็กต์เท่านั้น ไม่จำเป็นต้องสนใจรายละเอียดภายในของออบเจ็กต์ว่าเป็นอย่างไร

การใช้ออบเจ็กต์ของโปรแกรมจะมีลักษณะคล้ายกับการใช้สิ่งของในชีวิตประจำวันของเรา เช่น การใช้โทรทัศน์ เราสามารถใช้ได้โดยไม่จำเป็นต้องรู้ว่าภายในเครื่องโทรทัศน์มี **"ส่วนประกอบ"** อะไรบ้าง และไม่จำเป็นต้องรู้ว่าแต่ละส่วนประกอบทำงานอย่างไร เราจะรู้เพียงแค่วิธีใช้ เช่น วิธีเปิด วิธีเปลี่ยนช่อง วิธีปรับเสียง วิธีปรับสี วิธีตั้งเวลา วิธีปิดเครื่อง เป็นต้น ลักษณะของโปรแกรมแบบ OOP ก็มีลักษณะคล้ายกับการใช้โทรทัศน์ในชีวิตประจำวัน ซึ่งจะได้ศึกษาถึงวิธีการสร้างและวิธีการใช้ OOP ต่อไป

◆ คุณสมบัติของ OOP

ลักษณะของ OOP มีคุณสมบัติสำคัญ 3 ประการ ดังนี้

1. มีการรวมข้อมูลเข้ากับฟังก์ชัน (encapsulation) เพื่อให้เป็นข้อมูลชนิดออบเจ็กต์ โดยออบเจ็กต์หนึ่งจะมีคุณสมบัติเหมือนสิ่งของอย่างหนึ่ง คือมีทั้ง **"ส่วนประกอบ"** และ **"การทำงาน"** ซึ่งสามารถใช้งานได้ทันที จะได้กล่าวถึงรายละเอียดในหัวข้อต่อไป

2. มีการสืบทอด (inheritance) เป็นคุณสมบัติต่าง ๆ ของออบเจ็กต์หนึ่งซึ่งเป็น **บรรพบุรุษหรือต้นตอ** เรียกว่า **แอนเซสเตอร์ (ancestor)** และได้สืบทอดคุณสมบัติต่าง ๆ ที่มีอยู่ให้กับออบเจ็กต์ที่เป็น **ลูกหลานหรือผู้สืบทอด** เรียกว่า **ดีเซนแด็นต์ (descendant)** ได้หลายออบเจ็กต์ ทำให้เกิดความสัมพันธ์เกี่ยวโยงเป็นลำดับชั้น จะได้กล่าวถึงรายละเอียดในหัวข้อต่อไป

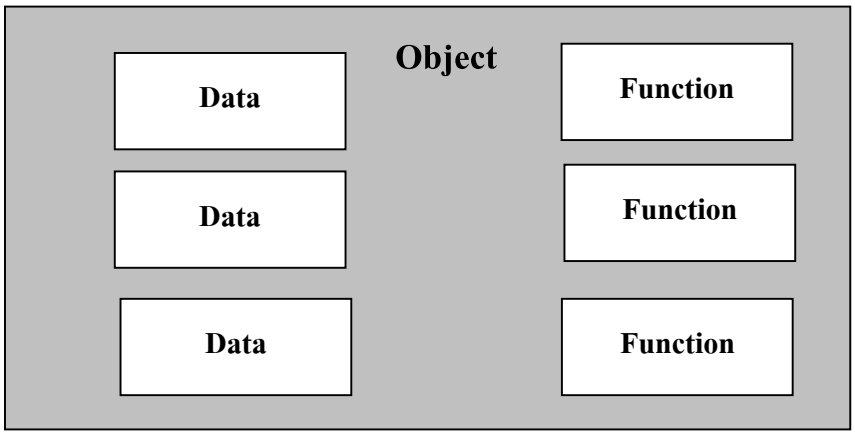
3. มีหลายรูปแบบ (polymorphism) คือคุณสมบัติที่เมื่อออบเจ็กต์ต่างๆ ได้รับคำสั่งเดียวกันจากโปรแกรมแล้ว แต่ละออบเจ็กต์จะทำงานตามแบบของตัวเอง ซึ่งทำให้ได้ผลลัพธ์แตกต่างกัน คุณสมบัตินี้จะกล่าวในหัวข้อต่อ ๆ ไป

◆ ความหมายของออปเจ็กต์ (Object)

Object หมายถึง สิ่งของหรือวัตถุเป้าหมาย การเขียนโปรแกรมแบบ OOP นั้น หมายถึง กลุ่มข้อมูล(data) และกลุ่มหน้าที่การทำงาน (function) ที่ถูกรวมเป็นหนึ่งหน่วย เราเรียกว่า **Object**

สามารถสรุปได้ว่า Object เป็นการนำเอาข้อมูลและฟังก์ชันรวมไว้ด้วยกัน ดังภาพ

หรือแสดงลักษณะของ Object และส่วนประกอบคือ กลุ่มของฟังก์ชันและข้อมูล ได้ดังภาพ



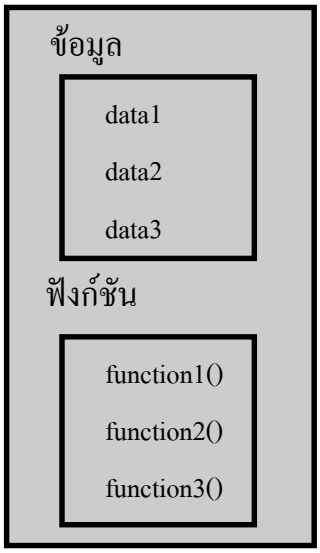
ลักษณะของออปเจ็กต์ดังกล่าว เป็นลักษณะของการรวมเอาข้อมูลและฟังก์ชันไว้ในหน่วยเดียวกัน เรียกว่า **เอนทิตี (Entity)** เดี่ยว ๆ ซึ่งเป็นแนวคิดสำคัญของการเขียนโปรแกรมแบบ OOP

การสร้างออปเจ็กต์ต้องสร้างจากข้อมูลชนิดที่เรียกว่า <u>คลาส (class)</u> ดังจะได้ศึกษาในหัวข้อต่อไป

◆ วิธีสร้างคลาส (class) ต้นกำเนิดของออปเจ็กต์

ออบเจ็กต์ (object) ต้องสร้างขึ้นจากข้อมูลชนิด **คลาส (class)** รูปแบบของข้อมูลชนิดคลาส มีลักษณะโครงสร้าง ดังนี้

คลาส



รูปแบบของการสร้างข้อมูลชนิด คลาส (class) เป็นดังนี้

```
class ชื่อคลาส
        {
            private: หรือ public: หรือ protected:
            ดาตาเมมเบอร์;
            ดาตาเมมเบอร์;
            ดาตาเมมเบอร์;
            private: หรือ public: หรือ protected:
            เมมเบอร์ฟังก์ชัน;
            เมมเบอร์ฟังก์ชัน;
            เมมเบอร์ฟังก์ชัน;
        };
```

คลาส (class) เป็นข้อมูลแบบสตรักเจอร์(strucuture) ซึ่งประกอบไปด้วยสมาชิกที่เรียกว่า **คลาสเมมเบอร์ (class member)** หรือสมาชิกของคลาส ซึ่งมีสมาชิกอยู่ 2 ประเภท คือ

1. ดาตาเมมเบอร์(data member) หรือ กลุ่มข้อมูลที่เป็นสมาชิกของคลาส ซึ่งอาจเรียกอย่างอื่นได้ เช่น **member variable, data item** เป็นต้น

2. เมมเบอร์ฟังก์ชัน (member functions) หรือกลุ่มของฟังก์ชันที่เป็นสมาชิกของคลาส อาจเรียกชื่อเป็นอย่างอื่นได้ เช่น **Method, Behavior**

จากรูปแบบส่วนประกอบของคลาส มีความหมายดังนี้

- **class ชื่อคลาส** คำว่า **class** เป็นคีย์เวิร์ด(keyword)ที่ต้องพิมพ์ไว้เพื่อกำหนดให้เป็นข้อมูลชนิด **class** เพื่อ ***เริ่มต้นสร้างแหล่งกำเนิดของออปเจ็กต์(object)*** ที่จะใช้ในโปรแกรม ส่วน **ชื่อคลาส** ตั้งตามกฎเกณฑ์การตั้งชื่อของ C++ เพื่อนำไว้ใช้อ้างอิงใน **การสร้างออปเจ็กต์** ในโปรแกรมต่อไป ส่วนรายละเอียดอื่น ๆ ของคลาสกำหนดไว้ในเครื่องหมาย **{ ….. };**

- **private: หรือ public: หรือ protected:** เป็นคีย์เวิร์ด(keyword) ต้องพิมพ์เพื่อกำหนดคุณสมบัติของดาต้าเมมเบอร์และเมมเบอร์ฟังก์ชัน มีความหมายดังนี้

private: หมายถึง ชื่อของดาต้าเมมเบอร์และเมมเบอร์ฟังก์ชันนั้น จะเรียกใช้ได้เฉพาะภายในคลาสนี้เท่านั้น

public: หมายถึง ชื่อของดาต้าเมมเบอร์และเมมเบอร์ฟังก์ชันนั้น จะเรียกใช้ได้ภายในคลาสนี้และจากภายนอกคลาสได้

protected: หมายถึง ชื่อของดาต้าเมมเบอร์และเมมเบอร์ฟังก์ชันนั้น จะเรียกใช้ได้เฉพาะภายในคลาส และใน **คลาสที่สืบทอดต่อ ๆ กันไป** เท่านั้น

- **ตัวอย่างการสร้างคลาสชื่อ Date อย่างง่าย ๆ มีดังต่อไปนี้**

```
class  Date                          //class name
    {
        private:                     // use only in this class
            int  Year;               //ดาต้าเมมเบอร์
            int  Month;              //ดาต้าเมมเบอร์
            int  Day;                //ดาต้าเมมเบอร์
        public:                      //use internal  and external from another class
            void  SetDate(int Y, int M, int D);   //เมมเบอร์ฟังก์ชัน
            void  Display();                      //เมมเบอร์ฟังก์ชัน
    };
```

จากตัวอย่างการกำหนดคลาสนี้ ชื่อคลาสคือ **Date** ประกอบไปด้วยดาต้าเมมเบอร์ที่เรียกใช้ได้เฉพาะในคลาสนี้เท่านั้น 3 ตัว คือ **Year, Month, Day** และในคลาสประกอบไปด้วยฟังก์ชันเมมเบอร์ ที่สามารถเรียกใช้ได้ทั้งภายในและภายนอกของคลาส (เพราะกำหนดไว้ด้วยคำว่า public) อีก 2 ฟังก์ชัน คือ **SetDate() และ Display()** ซึ่งรายละเอียดการเขียน code ของทั้ง 2 ฟังก์ชันจะได้ศึกษาในตัวอย่างต่อไป

รูปแบบของการสร้าง เมมเบอร์ฟังก์ชัน มีวิธีการเขียนเหมือนกับฟังก์ชันปกติของ C++ ที่ไม่ได้เป็นสมาชิกของคลาส ดังที่ได้ศึกษามาแล้ว แต่มีรูปแบบที่แตกต่างกันบ้าง ดังนี้

<u>รูปแบบที่ 1</u> กำหนดรายละเอียดของเมมเบอร์ฟังก์ชันต่อจากฟังก์ชัน main() จะต้องเขียนชื่อคลาส ใช้เครื่องหมายแบ่งแยกขอบเขต **:: (scope resolution operator)** ตามหลังชื่อคลาสเชื่อมกับชื่อเมมเบอร์ฟังก์ชัน เพื่อบอกว่าเป็นสมาชิกของคลาสนั้น เช่น

```
class Date          <-------  สร้างคลาสชื่อ Date ก่อนเพื่อนำไปสร้าง Object
{
    private:                        // use only in this class
        int  Year;                  //ดาต้าเมมเบอร์
        int  Month;                 //ดาต้าเมมเบอร์
        int  Day;                   //ดาต้าเมมเบอร์
    public:                         //use internal and external of this class
        void  SetDate(int Y, int M, int D);   //เมมเบอร์ฟังก์ชัน
        void  Display();                      //เมมเบอร์ฟังก์ชัน
};
void main()
{
    สเตตเมนต์ในโปรแกรม;
    สเตตเมนต์ในโปรแกรม;
}
```

```
void Date::SetDate(int Y, int M, int D);  //เมมเบอร์ฟังก์ชันของคลาส Date
{
      สเตตเมนต์ในฟังก์ชัน;
      สเตตเมนต์ในฟังก์ชัน;
}
```

สร้าง member function ชื่อ SetDate เป็นสมาชิกของคลาส Date โดยกำหนดด้วยเครื่องหมาย ::

```
void Date:: Display()  //เมมเบอร์ฟังก์ชันของคลาส Date
{
      สเตตเมนต์ในฟังก์ชัน;
      สเตตเมนต์ในฟังก์ชัน;
}
```

สร้าง member function ชื่อ Display() เป็นสมาชิกของคลาส Date โดยกำหนดด้วยเครื่องหมาย ::

<u>รูปแบบที่ 2</u> กำหนดรายละเอียดของเมมเบอร์ฟังก์ชันก่อนฟังก์ชัน main และอยู่ภายนอกคลาส จะต้องเขียนชื่อคลาสใช้เครื่องหมายแบ่งแยกขอบเขต **::** (scope resolution operator) ตามหลังชื่อคลาสเชื่อมกับชื่อเมมเบอร์ฟังก์ชัน เพื่อบอกว่าเป็นสมาชิกของคลาสนั้น

```
class Date
{
      private:                        // use only in this class
          int  Year;                  //ดาต้าเมมเบอร์
          int  Month;                 //ดาต้าเมมเบอร์
          int  Day;                   //ดาต้าเมมเบอร์
      public:                         //use internal and external of this class
          void  SetDate(int Y, int M, int D);   //เมมเบอร์ฟังก์ชัน
          void  Display();                      //เมมเบอร์ฟังก์ชัน
};
void Date::SetDate(int Y, int M, int D);  //เมมเบอร์ฟังก์ชันของคลาส Date
{
      สเตตเมนต์ในฟังก์ชัน;
      สเตตเมนต์ในฟังก์ชัน;
}
```

สร้าง member function ชื่อ SetDate() เป็นสมาชิกของคลาส Date โดยกำหนดด้วยเครื่องหมาย ::

```
void Date:: Display()  //เมมเบอร์ฟังก์ชันของคลาส Date
{
      สเตตเมนต์ในฟังก์ชัน;
      สเตตเมนต์ในฟังก์ชัน;
}

void main()
{
      สเตตเมนต์ในโปรแกรม;
      สเตตเมนต์ในโปรแกรม;
}
```

สร้าง member function ชื่อ Display() เป็นสมาชิกของคลาส Date โดยกำหนดด้วยเครื่องหมาย ::

<u>รูปแบบที่ 3</u> กำหนดรายละเอียดของเมมเบอร์ฟังก์ชันอยู่ภายในคลาส ***<u>ไม่ต้องเขียนชื่อคลาส</u>*** และไม่ต้องเขียนเครื่องหมายแบ่งแยกขอบเขต **:: (scope resolution operator)** นำหน้าชื่อเมมเบอร์ฟังก์ชัน เหมาะสำหรับโปรแกรมที่มีเมมเบอร์ฟังก์ชันจำนวนไม่มาก และมีรายละเอียดของฟังก์ชันสั้น ๆ มีรูปแบบดังนี้

```
class Date
{
      private:                  // use only in this class
         int  Year;             //ตาต้าเมมเบอร์
         int  Month;            //ตาต้าเมมเบอร์
         int  Day;              //ตาต้าเมมเบอร์
      public:                   //use internal and external of this class
      void SetDate(int Y, int M, int D);  //เมมเบอร์ฟังก์ชันของคลาส Date
      {
         สเตตเมนต์ในฟังก์ชัน;
      }
      void  Display()  //เมมเบอร์ฟังก์ชันของคลาส Date
      {
         สเตตเมนต์ในฟังก์ชัน;
      }
};
```

สร้าง member function ชื่อ SetDate() เป็นสมาชิกของคลาส Date โดยสร้างไว้ในคลาส

สร้าง member function ชื่อ Display() เป็นสมาชิกของคลาส Date โดยสร้างไว้ในคลาส

```
void main()
{
      สเตตเมนต์ในโปรแกรม;
}
```

<u>หมายเหตุ</u> การสร้างคลาสและรายละเอียดของเมมเบอร์ฟังก์ชัน อาจใช้หลายรูปแบบร่วมกันได้

♦ การสร้างและเรียกใช้ออบจ็กต์จากคลาส

<u>ก่อนที่จะสร้างออบเจ็กต์ใช้ในโปรแกรมได้จะต้องสร้างคลาสให้ถูกต้องก่อนเพราะ คลาส(class) เป็นต้นกำหนดของออบเจ็กต์</u> ซึ่งภายในคลาสจะต้องกำหนด ดาต้าเมมเบอร์และเมมเบอร์ฟังก์ชันไว้ และจะต้องเขียน code ของเมมเบอร์ฟังก์ชันให้เสร็จสมบูรณ์ก่อน จึงจะสามารถนำชื่อคลาสไปสร้างออบเจ็กต์และเรียกใช้ออกเจ็กต์ได้

- ***ตัวอย่างโปรแกรม clas_ex1.cpp เป็นตัวอย่างการสร้างคลาส การสร้างออบเจ็กต์และการเรียกใช้ออบเจ็กต์เบื้องต้น โดยสร้างเมมเบอร์ฟังก์ชันไว้ต่อจากฟังก์ชัน main() ดังนี้***

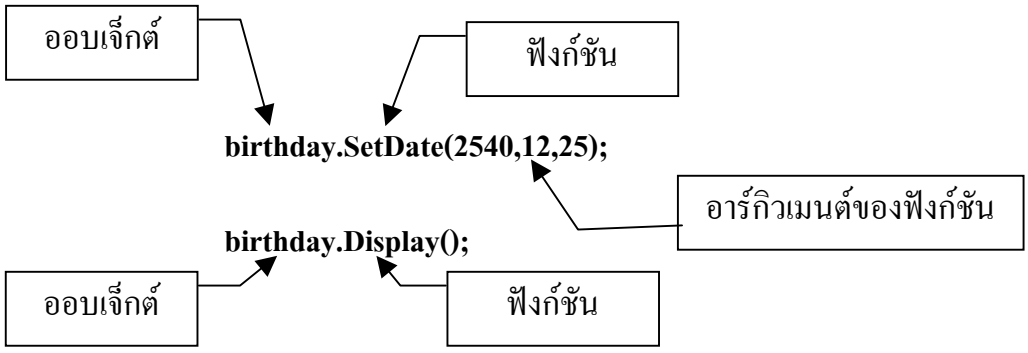
```
/*Program : clas_ex1.cpp
 Process : simple example of class*/
 #include <iostream.h>
 #include <conio.h>
 class Date  //create class Date
  {
    private:
      int  Year;    //data member
      int  Month;   //data member
      int  Day;     //data member
    public:
      void SetDate(int Y, int M, int D);    //member function
      void Display();                       //member function
  };

 void main()  //begin main program
  {
     Date birthday;    //สร้างออบเจ็กต์ชื่อ birthday จากคลาสชื่อ Date
    clrscr();
    cout<<"Display First OOP Programming"<<endl;
    birthday.SetDate(2540,12,25);       //เรียกใช้ออปเจ็กต์ birthday และฟังก์ชั่น SetDate() ทำงาน
    birthday.Display();                 //เรียกใช้ออปเจ็กต์ birthday และฟังก์ชัน display() ทำงาน
    getch();
  } //end main program
```

```
void Date::SetDate(int Y, int M, int D)  //detail of member function
 {
  Year = Y;
  Month = M;
  Day = D;
 }
void Date::Display()  //detail of member function
 {
  cout<<"Year : "<<Year<<endl;
  cout<<"Month: "<<Month<<endl;
  cout<<"Date : "<<Day<<endl;
 }
```

จากโปรแกรมตัวอย่าง สรุปได้ส่วนสำคัญได้ ดังนี้

1. การสร้างคลาสหรือกำหนดคลาส**ไม่ใช่การสร้างออบเจ็กต์** เป็นเพียงแต่แสดงให้ทราบว่า เมื่อมีการสร้างออบเจ็กต์แล้ว ออบเจ็กต์ที่ได้จะมีส่วนประกอบเหมือนที่มีอยู่ในคลาส ซึ่งในโปรแกรมได้สร้างออบเจ็กต์ขึ้น 1 ออบเจ็กต์ชื่อว่า **birthday** จากประโยค **Date birthday;** แสดงว่า **birthday** เป็นออบเจ็กต์ที่มีส่วนประกอบเหมือนที่มีอยู่ในคลาส **Date**

2. การเรียกใช้เมมเบอร์ฟังก์ชันของออบเจ็กต์ ต้องเขียนสเตตเมนต์ให้ถูกต้อง คือมีส่วนประกอบ 3 ส่วนคือ **ชื่อออบเจ็กต์.ชื่อเมมเบอร์ฟังก์ชัน() พร้อม argument ของฟังก์ชัน(ถ้ามี)** เช่น



เครื่องหมาย **.** มีชื่อเรียกว่า **คลาสเมมเบอร์แอคเซสโอเปอเรเตอร์ (class member access operator)** เป็นโอเปอเรเตอร์เพื่อเรียกใช้สมาชิกของคลาส

- *ตัวอย่างโปรแกรม clas_ex2.cpp แสดงถึงการกำหนดให้ดาต้าเมมเบอร์และเมมเบอร์ฟังก์ชันเป็นลักษณะ public ดังนั้นการเรียกใช้ออบเจ็กต์ birthday ในฟังก์ชัน main() จึงสามารถเรียกใช้ดาต้าเมมเบอร์ของ class ได้ด้วย จึงสามารถกำหนดค่าคงที่ให้ Year, Month, Day ได้โดยตรง ดังรายละเอียดโปรแกรมต่อไปนี้*

```
/*Program : clas_ex2.cpp
 Process : using public member of class*/
 #include <iostream.h>
 #include <conio.h>

 class Date  //create class Date
  {
     public:  //use internal and external class
      int  Year;    //data member
      int  Month;   //data member
      int  Day;     //data member
      void SetDate(int Y, int M, int D);  //member function
      void Display();                     //member function
  };

 void main()  //begin main program
  {
   Date birthday;  //create object name …birthday
   clrscr();
   //set value to data member of class from object
   birthday.Year=2540;
   birthday.Month=12;
   birthday.Day = 25;
   cout<<"Display Second OOP Programming"<<endl;
   birthday.Display();  getch();
  } //end main program

 void Date::SetDate(int Y, int M, int D)  //detail of member function
  {
   Year = Y;
   Month = M;
   Day = D;
  }

 void Date::Display()  //detail of member function
  {
   cout<<"Year : "<<Year<<endl;
   cout<<"Month: "<<Month<<endl;
   cout<<"Date : "<<Day<<endl;
  }
```

- *โปรแกรม oop_exam.cpp แสดงการสร้างคลาสที่ชื่อว่า VarClass ภายในคลาสประกอบไปด้วย ดาต้าเมมเบอร์ CharVar, IntVar, FloatVar และมีเมมเบอร์ฟังก์ชันชื่อ SetValue และ DisplayData โดยเขียนไว้ภายในคลาส จากนั้นได้สร้างออบเจ็กต์ชื่อ Obj_number ดังต่อไปนี้*

```
/*Program : oop_exam.cpp
  Process : Create New Class and Object*/
  #include <iostream.h>
  #include <conio.h>

  class VarClass   //Begin create class name is... VarClass
  {
    private:  //data member
      char CharVar;
      int  IntVar;
      float FloatVar;
    public:  //member function
      void SetValue()   //function in class
      {
         CharVar = 'A';
         IntVar  = 100;
         FloatVar= 3.14159;
       }

    void DisplayData() //function in class
      {
       cout<<"Character : "<<CharVar<<endl;
       cout<<"Integer : "<<IntVar<<endl;
       cout<<"Float : "<<FloatVar<<endl;
      }
  };   //end of class

  void main()   //begin main program
  {
   VarClass  Obj_number;   //create object name is Obj_number
   clrscr();
   Obj_number.SetValue();
   Obj_number.DisplayData();
   getch();
  }  // end main program
```

- *ตัวอย่างโปรแกรม oop_exa2.cpp มีผลการทำงานเหมือนโปรแกรม oop_exam.cpp แต่เขียนเมมเบอร์ฟังก์ชันไว้นอกคลาส และสร้างไว้หลังฟังก์ชัน main() มีรูปแบบดังนี้*

```
/*Program : oop_exam.cpp
  Process : Create New Class and Object*/
  #include <iostream.h>
  #include <conio.h>
  class VarClass   //Begin create class name is... VarClass
  {
    private:  //data member
      char CharVar;
      int  IntVar;
      float FloatVar;
    public:
     void SetValue();     //member function
     void DisplayData();  //member function
  };   //end of class

void main()   //begin main program
  {
    VarClass  Obj_number;  //create object
    clrscr();
    Obj_number.SetValue();          //call object and function
    Obj_number.DisplayData();       //call object and function
    getch();
  }

void VarClass::SetValue()   //function in class
  {
     CharVar = 'A';
     IntVar  = 100;
     FloatVar= 3.14159;
  }

void VarClass::DisplayData() //function in class
  {
       cout<<"Character : "<<CharVar<<endl;
       cout<<"Integer : "<<IntVar<<endl;
       cout<<"Float : "<<FloatVar<<endl;
  }
```

- ***ตัวอยางโปรแกรม clas_ex3.cpp มีการสร้างเมมเบอร์ฟังก์ชัน GetDate() เพิ่มในโปรแกรมเพื่อทำหน้าที่รับค่าคงที่ทางคีย์บอร์ด และสร้างออบเจ็กต์ชื่อ Dday เพิ่มขึ้นอีกหนึ่งออบเจ็กต์ ดังนี้***

```
/*Program : clas_ex3.cpp
  Process : using public member of class */
  #include <iostream.h>
  #include <conio.h>

  class Date  //create class Date
    {  public:  //use internal and external class
         int  Year;    //data member
         int  Month;   //data member
         int  Day;     //data member
         void GetDate(); //member function ...enter date
         void SetDate(int Y, int M, int D);    //member function...send argumen
         void Display();                              //member function...display date
    };

void main()  //begin main program
  {
    Date birthday,Dday;   //create 2 objects from Date class are birthday and Dday
    clrscr();
    //set value to data member of class
    birthday.GetDate();
    cout<<"Display Date enter from keyboard by object :birthday"<<endl;
    birthday.Display();
    getch();clrscr();
    cout<<endl<<"Display Date from argument by object : Dday"<<endl;
    Dday.SetDate(2530,11,29);
    Dday.Display();
    getch();
  } //end main program

void Date::GetDate()  //detail of member function GetDate()
  {
     cout<<"Enter new Year<yyyy>: ";cin>>Year;
     cout<<"Enter new Month<mm>: ";cin>>Month;
     cout<<"Enter new Day<dd>: ";cin>>Day;
  }
```

```
void Date::SetDate(int Y, int M, int D)  //detail of member function SetDate()
 {
  Year = Y;
  Month = M;
  Day = D;
 }

void Date::Display()  //detail of member function display()
 {
  cout<<"Year : "<<Year<<endl;
  cout<<"Month: "<<Month<<endl;
  cout<<"Date : "<<Day<<endl;
 }
```

- ***โปรแกรม oop_ex1.cpp แสดงการสร้างคลาสชื่อ Calculate กำหนดให้มีสมาชิกเป็น data member ลักษณะ private จำนวน 2 ตัว คือ first และ second กำหนดให้มี member function จำนวน 4 ฟังก์ชัน คือ sum(), subtract(), multiply(), divide() เพื่อทำหน้าที่ในการคำนวณการบวก,ลบ,คูณและหาร ของเลข 2 จำนวนตามลำดับ ในโปรแกรมมีการสร้าง object จำนวน 1 objcect ชื่อ math แล้วนำออบเจ็กต์นี้มาเรียกใช้ในโปรแกรม ดังต่อไปนี้***

```
/*Program : oop_exp1.cpp
 Process : calculate */
 #include <iostream.h>
 #include  <conio.h>

 class Calculate  //create class
 {
  private:
    float first;
    float second;
  public:
    float sum(float x, float y);
    float subtract(float a, float b);
    float multiply(float a, float b);
    float divide(float a, float b);
 };

 //declaration prototype
 void input(); //Not member function
 //declar global variable
  float number1,number2;
 Calculate  math;  //create object ....math from Calculate class
```

```
void main()  //begin main program
{
  input();
  cout<<"\nResult of sum = "<<math.sum(number1,number2);
  cout<<"\nResult of subtract = "<<math.subtract(number1,number2);
  cout<<"\nResult of multiply = "<<math.multiply(number1,number2);
  cout<<"\nResult of devide = "<<math.divide(number1,number2);
  getch();
}  //end main program

void input() // Normal function enter 2 number , non member of class
{ clrscr();
 cout<< "input first number : ";cin>>number1;
 cout<< "input first number : ";cin>>number2;
}
//member function sum() of class Calculate

float Calculate::sum(float first, float second)
{
 return first+second;
}

//member function subtract() of class Calculate
float Calculate::subtract(float first, float second)
{
 return first-second;
}

//member function multiply() of class Calculate
float Calculate::multiply(float first, float second)
{
 return first*second;
}
//member function divide() of class Calculate
float Calculate::divide(float first, float second)
{
 return first/second;
}
```

- ***ตัวอย่างโปรแกรม oop_ex2.cpp แสดงการคำนวณการตัดเกรด โดยการสร้างคลาสชื่อ Grade และสร้างออบเจ็กต์ชื่อ Evalue มีกระบวนการทำงานของ member function อยู่ 3 ฟังก์ชัน คือ GetScore() ทำหน้าที่รับคะแนนระหว่างภาคและปลายภาค รวมคะแนน, Calculate() ทำหน้าที่นำคะแนนรวมไปคำนวณตัดเกรด และ Display() ทำหน้าที่แสดงผลคะแนนรวมและเกรดที่ได้รับ ดังรายละเอียดในโปรแกรมต่อไปนี้***

```
/*Program: grade2.cpp
  Process: calculate grade from total score by OOP Programming*/
#include <iostream.h>
#include <conio.h>
class Grade   //create class ...Grade
  { private:
     float midterm;  //data member
     float final;
     float total;
     char  gd;
   public:
     void GetScore();  //member function
     void Calculate();
     void Display();
  };
Grade Evalue;  //Create object ....Evalue
void main()  //begin main program
{
  clrscr();
  cout<< "Program Calculate Grade"<<endl;
  //use object and function of class
  Evalue.GetScore();
  Evalue.Calculate();
  Evalue.Display();
  getch();
}

void Grade::GetScore()  //member function of Grade class
{
   cout<< "Enter midterm score: ";cin>>midterm;
   cout<< "Enter midterm score: ";cin>>final;
   total=midterm+final;
}
```

```
void Grade::Calculate() //member function of Grade class
{
 //calculate grade use if...else if..
 if (total<0 || total>100) //check enter error score
    gd = '*';
 else if (total>=0 && total<=49)
        gd='F';
     else if(total>=50 && total<=59)
           gd='D';
         else if(total>=60 && total<=69)
              gd='C';
            else if(total>=70 && total<=79)
                 gd='B';
               else
                  gd='A';
}
void Grade::Display()  //member function of Grade class
{
  cout<< "Total of your score: \a"<<total<<endl;
  cout<< "You get grade : "<<gd<<endl;
  if (gd=='*')
    cout<< "Your score = "<<total<< " is error range !!!"<<endl;
}
```

- ***ตัวอย่างโปรแกรม exp_oop1.cpp แสดงการเขียนโปรแกรมแบบ OOP โดยมีเมมเบอร์ฟังก์ชัน 2 ฟังก์ชัน คือ input() และ display() โดยที่ฟังก์ชัน input() ทำหน้าที่รับข้อมูลทางแป้นพิมพ์ และฟังก์ชัน display() ทำหน้าที่แสดงผลข้อมูลที่กรอกทางแป้นพิมพ์ไว้***

```
/*Program: EXP_OOP1.CPP
 Process: Input,Display data function*/

#include <iostream.h>
#include <conio.h>
#include <stdio.h>
class TEST   //create class
{  public:
   char  name[30];        //data member
   char  address[50];
   int  age;
  public:                 //member function
   void input();
   void display();
};
```

```
TEST  Info;  //create object ...Info... from class

void main()   //begin main program
{ clrscr();
  Info.input();
  Info.display();
}

void TEST::input()  //Create member function of class TEST
{  cout<<"Enter your information: "<<endl;
   cout<<"Name: ";
   gets(name);
   cout<<"Address: ";
   gets(address);
   cout<<"Age: ";
   cin>>age;
}
void TEST::display() //Create member function of class TEST
{  clrscr();
   cout<<"**********************"<<endl;
   cout<<"Your information... "<<endl;
   cout<<"**********************"<<endl;
   cout<<"Name: "<<name<<endl;
   cout<<"Address: "<<address<<endl;
   cout<<"Age: "<<age <<endl;
   cout<<"**********************\a\a"<<endl;
   getch();
}
```

- ***ตัวอย่างโปรแกรม Draw_cir.cpp เป็นโปรแกรมสร้างออบเจ็กต์ ชื่อ TheCircle จากคลาส Circle โดยกำหนดเมมเบอร์ฟังก์ชันเพื่อกำหนดตำแหน่งวาดวงกลมโดยการกำหนดค่า SetCircle() และรับค่าตำแหน่งของวงกลม InputCircle() พร้อมแสดงผลการวาดวงกลม Display() ดังต่อไปนี้***

```
/*Program : Draw_cir.cpp
  Process : set position and draw circle graphics by OOP */
#include <iostream.h>
#include <stdlib.h>
#include <graphics.h>
```

```
#include <conio.h>
class Circle
  {
   private:
     int X, Y, Radius;
   public:
     void SetCircle(int,int,int);
     void InputCircle();
     void Display();
  };

void main()
 {
  Circle TheCircle;   //create object ....TheCircle
  clrscr();
  /* request auto detection */
  int gdriver = DETECT, gmode, errorcode;
  /* initialize graphics mode */
  initgraph(&gdriver, &gmode, "");
  TheCircle.SetCircle(300,200,200);  //set X,Y and Radius
  TheCircle.Display();              //Display Circle
  TheCircle.SetCircle(300,200,100);  //set X,Y and Radius
  TheCircle.Display();              //Display Circle
  getch();clrscr();
  TheCircle.InputCircle();          //Enter  X,Y and Radius
  TheCircle.Display();              //Display Circle
  getch();
  closegraph;
 }

void Circle::SetCircle(int Xc, int Yc,  int Rc)
{  X=Xc;
   Y=Yc;
   Radius=Rc;
}

void Circle::Display()
{
  circle(X,Y,Radius);
}
```

```
void Circle::InputCircle()
{
 closegraph;
 cout<< "Enter data for circle:"<<endl;
 cout<< "Center point : X = ";cin>>X;
 cout<< "Center point : Y = ";cin>>Y;
 cout<< "Radius : ";cin>>Radius;
}
```

♦ ตัวอย่างการเขียนโปรแกรมแบบ OOP จัดการ Array และ Structure

เราสามารถจัดการข้อมูลที่เป็น array โดยสร้างให้เป็นข้อมูลชนิด Object ได้ ดังตัวอย่างโปรแกรมต่อไปนี้

ตัวอย่างโปรแกรม OOP_ARR.CPP แสดงการใช้ array ที่เป็น Object ชื่อ Object คือ **result [5]** สร้างจาก class ที่ชื่อ **TEST** โดยจอง array ไว้ทั้งหมด 5 ช่อง มี member function ของ class คือ **input(), summation(), display()** โ ดยที่ฟังก์ชัน input() ทำหน้าที่รับข้อมูลเข้าไปเก็บใน data ที่ชื่อ number1, number2 ฟังก์ชัน summation() ทำหน้าทีรวมจำนวน number1+number2 เก็บไว้ใน sum และฟังก์ชัน display() ทำหน้าที่แสดงข้อมูล array ใน data ที่ชื่อ number1, number2 ที่ได้กรอกไว้ 5 จำนวน

```
/*Program: OOP_ARR.CPP
 Process: Uses array of Object for Input,Summation,Display data function*/

#include <iostream.h>
#include <conio.h>
#include <stdio.h>

class TEST          //create class
{  public:          //data members
  int number1;
  int number2;
  int sum;
  public:           //member function
   void input();
   void display();
   void summation();
};
```

```
TEST result[5];  //Create Object ....is Array  result[5]

void main()
{
 clrscr();
 for(int i=0;i<=4;i++)
 {
  result[i].input();
 }
 cout<<endl<<"Display number 1, number 2 in array"<<endl;
 for(i=0;i<=4;i++)
 {
  result[i].display();
 }
//Calculate Summation number1+number2 in array
 for(i=0;i<=4;i++)
 {
    result[i].summation();
 }
 getch();
 cout<<endl<<"Display Summation: "<<endl;
 for(i=0;i<=4;i++)
 {
  cout<<"Sum["<<i<<"]"<<result[i].sum<<endl;
 }
 getch();

}  //end main program

void TEST::input()  //Create member function of class TEST
{  cout<<"Enter your number: "<<endl;
  cout<<"Number1: ";cin>>number1;
  cout<<"Number2: ";cin>>number2;
}
void TEST::display() //Create member function of class TEST
{
  cout<<"Your number : "<<endl;
  cout<<"Number1: "<<number1<<endl;
  cout<<"Number2: "<<number2<<endl<<endl;
}
```

```
void TEST::summation()
{
 sum=number1+number2;
}
```

- ***ตัวอย่างโปรแกรม OOP_STRU.CPP เป็นตัวอย่างโปรแกรมเขียนแบบ OOP เพื่อจัดการข้อมูลที่เป็นอาร์เรย์ของโครงสร้าง มีดังต่อไปนี้***

```
/*Program: OOP_STRU.CPP
 Process: Used OOP manage Array of Structure */

#include <iostream.h>
#include <conio.h>
#include <stdio.h>    //for gets() function

struct  PER   //Create Structure Data Type ....Global
 {
  char code[5];
  char name[30];
  char position[20];
  float salary;
 };

class EMP      //Create Class for origin of Object
 { public:
    PER person[5];   //data member is ...array of PER
  public:
    void Input();    //member function
    void Display();
    void Report();
 };

 EMP  Employee;  //Creat Object from class ...EMP

 void main()
 { clrscr();
  Employee.Input();
  Employee.Display();
  getch();clrscr();
```

```
  Employee.Report();
  getch();
 }

void EMP::Input()   //Detail of member function
{ int i;
 for(i=0;i<=4;i++)
 {  cout<<"Code: ";gets(person[i].code);
    cout<<"Name: ";gets(person[i].name);
    cout<<"Position: ";gets(person[i].position);
    cout<<"Salary: ";cin>>person[i].salary;
 }
}

void EMP::Display()   //Detail of member function
{ int i;
 clrscr();cout<<"Display Information"<<endl;
 for(i=0;i<=4;i++)
 {  cout<<"Record#"<<i+1<<endl;
    cout<<person[i].code<<endl;
    cout<<person[i].name<<endl;
    cout<<person[i].position<<endl;
    cout<<person[i].salary<<endl;
    getch();clrscr();
   }
}

void EMP::Report()    //Detail of member function
{ int r=6,i;
  clrscr();
  gotoxy(25,1);cout<<"Report Salary Expense";
  gotoxy(25,2);cout<<"Sirichai Export Co.ltd";
  gotoxy(1,3);cout<<"-------------------------------------------------";
  gotoxy(1,4);cout<<"Code     Name - Surname    Position    Salary";
  gotoxy(1,5);cout<<"-------------------------------------------------";
  for(i=0;i<=4;i++)
  { gotoxy(1,r);cout<<person[i].code;
    gotoxy(8,r);cout<<person[i].name;
    gotoxy(25,r);cout<<person[i].position;
    gotoxy(40,r);cout<<person[i].salary;
```

```
    r++;
    if(r>=22)
    { getch();
      r=6;
     }
  }
}
```

♦ ฟังก์ชันชนิดคอนสตรักเตอร์และดิสตรักเตอร์

จากตัวอย่างที่ผ่านมาในการกำหนดค่าให้แก่ดาตาเมมเบอร์ของคลาสในขณะเรียกใช้ออปเจ็กต์นั้น เราจะกำหนดค่าคงที่ให้ขณะที่มีการเรียกใช้ เช่น Dday.SetDate(2530,11,29); หรือมีการรับค่าทางคีย์บอร์ดเพื่อกำหนดค่าดาตาเมมเบอร์

การกำหนดค่าให้แก่ดาตาเมมเบอร์สามารถกำหนดให้มีค่าเริ่มต้นแบบอัตโนมัติได้ โดยการใช้เมมเบอร์ฟังก์ชันชนิดที่เรียกว่า **คอนสตรักเตอร์ (constructor)** ซึ่งจะเป็นเมมเบอร์ฟังก์ชันที่ทำงานโดยอัตโนมัติทันทีที่ออบเจ็กต์ถูกสร้างขึ้นเพราะฉะนั้นจะต้องกำหนดค่าเริ่มต้นของดาตาเมมเบอร์ไว้เป็นค่าเริ่มต้นด้วย การสร้างเมมเบอร์ฟังก์ชันชนิดคอนสตรักเตอร์ สร้างไว้ใน **class** กำหนดชื่อให้เหมือนกับชื่อของ **class** และกำหนดค่าคงที่ให้แก่ ดาตาเมมเบอร์ภายในเครื่องหมาย { } ดังรูปแบบตัวอย่างในโปรแกรม cons_oop.cpp

- ***ตัวอย่างโปรแกรม cons_oop.cpp แสดงการสร้าง constructor เพื่อกำหนดค่าเริ่มต้นอัตโนมัติเมื่อมีการใช้ออปเจ็กต์ มีรายละเอียดดังนี้***

```
/*Program : cons_oop.cpp
 Process : create constructor member function of class*/
 #include <iostream.h>
 #include <conio.h>
 class Date  //create class Date
  {
    private:
      int  Year;   //data member
      int  Month;  //data member
      int  Day;    //data member
    public:
      Date()  //constructor member function…function name same as class name
      { Year = 1997;    //ค่าคงที่ของ data member ที่เป็นค่าเริ่มต้น
        Month=12;
        Day=31;
      }
```

```
    void SetDate(int Y, int M, int D);          //member function
    void Display();                             //member function
};

void main()  //begin main program
  {
   Date birthday;
   clrscr();
   cout<<"Display Constructor member function : OOP Programming"<<endl;
   //use constructor
   cout<<endl<<"Display Date from constructor member function"<<endl;
   birthday.Display();
   //set value of data member
   cout<<endl<<"Display Date from setting value"<<endl;
   birthday.SetDate(2540,10,22);
   birthday.Display();
   getch();
  } //end main program

void Date::SetDate(int Y, int M, int D)  //detail of member function
   {
    Year = Y;
    Month = M;
    Day = D;
   }

 void Date::Display()  //detail of member function
   {
    cout<<"Year : "<<Year<<endl;
    cout<<"Month: "<<Month<<endl;
    cout<<"Date : "<<Day<<endl;
   }
```

ผลการทำงานของโปรแกรม

Display Constructor member function : OOP Programming

Display Date from constructor member function //ค่าที่ได้เกิดจากการใช้ constructor

Year : 1997

Month: 12

Date : 31

Display Date from setting value //ค่าที่ได้เกิดจากการกำหนดค่า

Year : 2540

Month: 10

Date : 22

เมื่อมีการสร้าง constructor ซึ่งจะทำให้ออบเจ็กต์สามารถมีค่าเริ่มต้นทำงานได้โดยอัตโนมัติได้ การยกเลิก constructor หรือยกเลิกการทำงานของออบเจ็กต์โดยอัตโนมัติ **เพื่อยกเลิกการใช้หน่วยความจำ** จะต้องสร้างเมมเบอร์ฟังก์ชันชนิดที่เรียกว่า **ดิสตรักเตอร์ (destructor)** ซึ่งจะมีชื่อเดียวกับคอนสตรักเตอร์เพียงแต่มีเครื่องหมาย ~ กำกับที่หน้าชื่อฟังก์ชันไม่มีพารามิเตอร์และไม่มีรายละเอียดในเครื่องหมาย { } เช่น

```
class Date  //create class Date
  {
    private:
      int  Year;   //data member
      int  Month;  //data member
      int  Day;    //data member
    public:
      Date()  //constructor member function…function name same as class name
       { Year = 1997;    //ค่าคงที่ของ data member ที่เป็นค่าเริ่มต้น
         Month=12;
         Day=31;
       }
        ~Date() { };   //สร้างดิสตรักเตอร์
      void SetDate(int Y, int M, int D);          //member function
      void Display();                             //member function
  };
```

<u>หมายเหตุ</u> ทั้งฟังก์ชันประเภท คอนสตรักเตอร์และดิสตรักเตอร์ จะไม่มีการส่งค่าออกจากฟังก์ชัน

◆ คุณสมบัติการสืบทอด (inheritance)

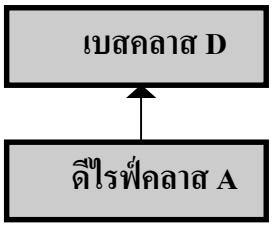
การสืบทอด (inheritance) เป็นคุณสมบัติสำคัญของการเขียนโปรแกรมเชิงวัตถุเป้าหมาย (OOP) หมายถึง การถ่ายทอดคุณสมบัติต่าง ๆ ของออบเจ็กต์ที่เป็นต้นกำเนิดหรือบรรพบุรุษ ซึ่งเราเรียกว่า **แอนเซสเตอร์ (ancestor)** ไปยังออบเจ็กต์ที่เป็นลูกหลานหรือผู้สืบทอด เราเรียกว่า **ดีเซนแด็นต์ (desendant)** ได้หลายออบเจ็กต์ ทำให้เกิดความสัมพันธ์ระหว่างคลาสและออบเจ็กต์เป็นลำดับชั้นเหมือนกับการลำดับความสัมพันธ์ของเครือญาติ

การถ่ายทอดคุณสมบัติของคลาสจะเริ่มจากคลาสเริ่มต้น ที่เรียกว่า **เบสคลาส (base class)** หรือ **แอนเซสเตอร์ (ancestor)** ไปยังคลาสใหม่ที่สืบเนื่องจากคลาสเริ่มต้น เรียกคลาสที่สืบเนื่องนี้ว่า **ดีไรฟ์คลาส (derived class)** หรือ **ดีเซนแด็นต์ (desendant)**

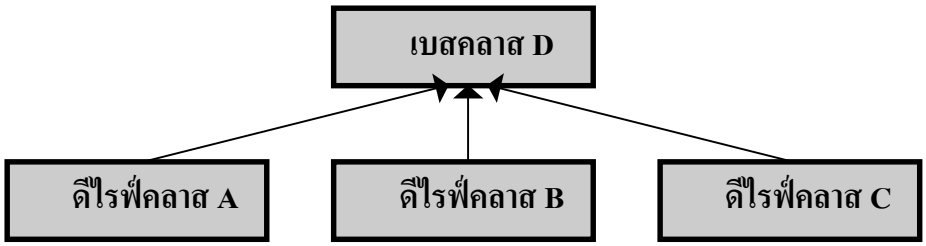
ดีไรฟ์คลาส (derived class) จะสืบทอดคุณสมบัติทั้งหมดของ ดาต้าเมมเบอร์และเมมเบอร์ฟังก์ชันที่มีอยู่ใน **เบสคลาส (base class)** **<u>ยกเว้น</u>** คอนสตรักเตอร์และดิสตรักเตอร์

ลักษณะความสัมพันธ์ระหว่าง base class กับ derived class จะอยู่ในรูปแบบ 4 ลักษณะ ดังนี้

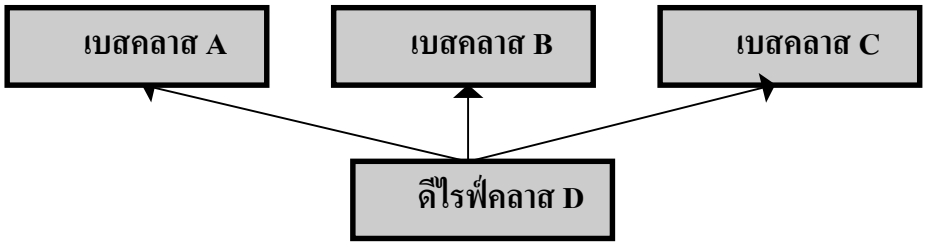
1. หนึ่งดีไรฟ์คลาส สืบทอดคุณสมบัติจาก หนึ่งเบสคลาส ดังรูป



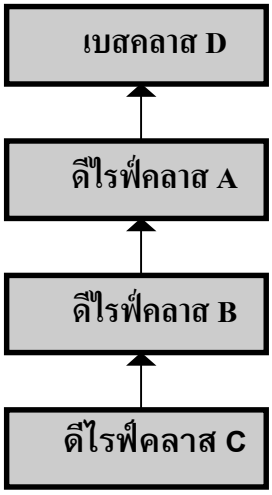
2. หลายดีไรฟ์คลาส สืบทอดคุณสมบัติจาก หนึ่งเบสคลาส ดังรูป



3. หนึ่งดีไรฟ์คลาส สืบทอดคุณสมบัติจาก หลายเบสคลาส ดังรูป



4. หลายดีไรฟ์คลาส สืบทอดคุณสมบัติต่อเนื่องกันไปหลายลำดับจาก หนึ่งเบสคลาส ดังรูป



รูปแบบการสร้าง **ดีไรฟ์คลาส (derived class)** จะมีรูปแบบและส่วนประกอบต่างๆ เหมือนกับการสร้าง เบสคลาส มีรูปแบบ ดังนี้

class ชื่อดีไรฟ์คลาส : private หรือ public หรือ protected **ชื่อเบสคลาส**

```
{
  private หรือ public หรือ protected :
        ดาตาเมมเบอร์;
        ดาตาเมมเบอร์;
        ดาตาเมมเบอร์;
  private หรือ public หรือ protected :
        เมมเบอร์ฟังก์ชัน;
        เมมเบอร์ฟังก์ชั่น;
};
```

ตัวอย่างเช่น การสร้างเบสคลาส ชื่อ Draw และสร้างดีไรฟ์คลาส ชื่อ Point

```
class Draw   //เบสคลาส
      {
        protected:
              int  X, Y, Color;
        public:
              Draw(int  InitX,  int  InitY,  int  InitColor)  //คอนสตรักเตอร์
                {   X = InitX;
                    Y = InitY;
                     Color = InitColor;
                 }
              ~Draw()     //ดิสตรักเตอร์
                  {
                   closegraph();
                    }
      };
class Point :   public Draw   //ดีไรน์คลาสชื่อ Point สืบทอดมาจากคลาส Draw
      {
        public:
            //คอนสตรักเตอร์ฟังก์ชันชื่อ  Point
            Point(int  InitX,  int  InitY,   Int   InitColor) : Draw(InitX, InitY, InitColor)
{};
        void  Display();
        void Hide();
      };
```

การกำหนดเบสคลาสและดีไรฟ์คลาสจากตัวอย่าง มีความหมายดังนี้

- **Draw** เป็นเบสคลาส มีคลาส **Point** เป็นดีไรฟ์คลาส แสดงว่าคลาส Point จะสามารถสืบทอดคุณสมบัติต่าง ๆ มาจากคลาส Draw ได้

- **class Point : public Draw** เป็นประโยคกำหนดให้คลาส Point เป็นดีไรฟ์คลาสของคลาส Draw คำว่า public กำหนดอยู่หน้าชื่อเบสคลาส Draw หมายถึง ออบเจ็กต์ที่สร้างโดยใช้ดีไรฟ์คลาสนั้น จะสามารถเรียกใช้สมาชิก (ดาต้าเมมเบอร์และเมมเบอร์ฟังก์ชัน) ของเบสคลาสในส่วนที่เป็น public ได้ทั้งหมด ตามตัวอย่าง ถ้าสร้างออบเจ็กต์ชื่อ Apoint จากดีไรฟ์คลาส Point

Point Apoint;

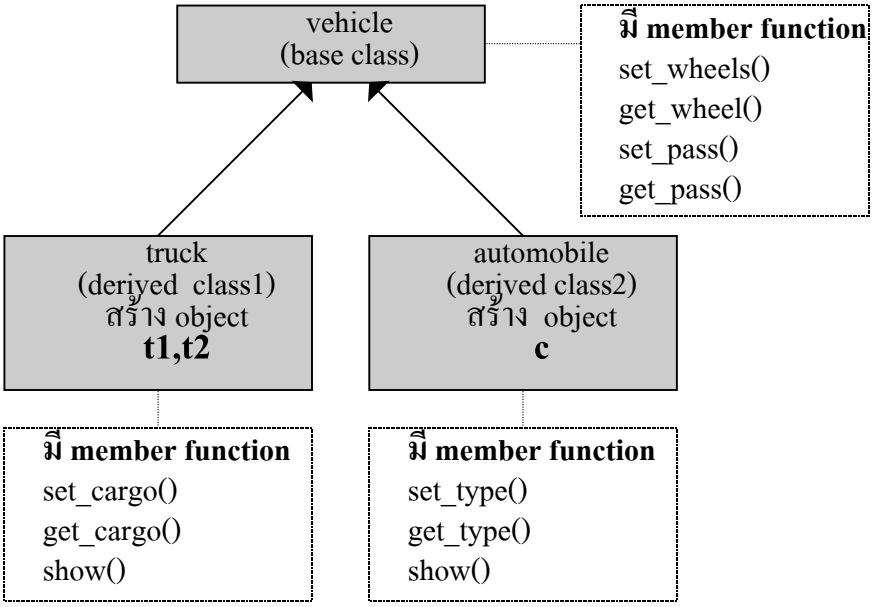
แสดงว่าออบเจ็ตก์ **Apoint** เป็นออบเจ็กต์ที่สร้างจากดีไรฟ์คลาสที่ชื่อ **Point** จะสามารถเรียกใช้ฟังก์ชันในคลาส Draw ได้ทุกฟังก์ชัน

- **protected:** ในคลาส Draw หมายถึง สมาชิกของคลาส (อาจเป็นดาต้าเมมเบอร์หรือเมมเบอร์ฟังก์ชันก็ได้) ที่อยู่ภายใต้คำว่า protected จะยอมให้ดีไรฟ์คลาสของคลาสนี้เท่านั้นที่สามารถเรียกใช้ได้

ในตัวอย่างแสดงว่า นอกจากการเรียกใช้สมาชิกในคลาส Draw ได้แล้ว คลาส Point จะสามารถเรียกใช้ดาตาเมมเบอร์ X, Y และ Color ของคลาส Draw ไปใช้ได้ด้วย เนื่องจากคลาส Point เป็นดีไรฟ์คลาสของ Draw นั่นเอง **(เรียกว่าลักษณะการสืบทอดคุณสมบัติ)**

♦ ตัวอย่างการสืบทอดคุณสมบัติในโปรแกรม

โปรแกรม INHER_EX.CPP แสดงการสืบทอดคุณสมบัติ (ฟังก์ชันและข้อมูล) ของคลาสในรูปแบบ หนึ่งเบสคลาสสืบทอดคุณสมบัติไปสู่หลายดีไรฟ์คลาส มีโครงสร้างการสืบทอด ดังนี้



จากโครงสร้างการสืบทอดของเบสคลาส vehicle ไปสู่ดีไรฟ์คลาส คือ **truck , automobile** ได้มีเรียกใช้ object ที่สร้างจากคลาส truck และ automobile แต่เรียกใช้คุณสมบัติหรือสืบทอดคุณสมบัติที่เป็น member function ของคลาส vehicle ซึ่งเป็นเบสคลาส ในโปรแกรมดังนี้

```
t1.set_wheels(18);
t1.set_cargo(3200);
t2.set_wheels(6);
t2.set_pass(3);
c.set_wheels(4);
c.set_pass(6);
```

- ***ตัวอย่างโปรแกรม INHER_EX.CPP เป็นโปรแกรมแสดงการใช้คุณสมบัติ Inheritance ของการเขียนโปรแกรมแบบ OOP***

```
/*Program : INHER_EX.CPP
  Process : show inheritance from base class to derived class
            class vehicle --> class turck, automobile
*/
  #include <iostream.h>
  #include <conio.h>
  class vehicle              //base class
  { int wheels;
    int passengers;
    public:                  //declared public function in class
      void set_wheels(int num);
      int get_wheels();
      void set_pass(int num);
      int get_pass();
  };

  class truck:public vehicle  //derive class of class vehicle
  {  int cargo;
     public:
     void set_cargo(int size);
     int get_cargo();
     void show();
  };

  enum type_auto{car,van,wagon};          //set enumerated data type

  class automobile:public vehicle //derive class of class vehicle
  { type_auto car_type;                   //enumerated variable
    public:
      void set_type(enum type_auto t);
      enum type_auto get_type();
      void show();
  };

  void vehicle::set_wheels(int num)
  {
    wheels=num;
  }
int vehicle::get_wheels()
{
   return wheels;
}
```

```
void vehicle::set_pass(int num)
 {
  passengers=num;
 }

int vehicle::get_pass()
 {
  return passengers;
 }

 //begin created member function of truck class
 void truck::set_cargo(int num)
 {
  cargo=num;
 }
int truck::get_cargo()
 {
  return cargo;
 }
 void truck::show()
 {
   cout<< "Wheels : "<<get_wheels()<<"\n";
   cout<< "Passengers : "<<get_pass()<<"\n";
   cout<< "Cargo capacity in cubic feet : "<<cargo<<"\n";
 }

 //begin created member function of automobile class
 void automobile::set_type(enum type_auto t)
 {
   car_type=t;
 }

type_auto automobile::get_type()
 {
   return car_type;
 }


void automobile::show()
 {
  cout<<"Wheels : "<<get_wheels()<<"\n";
  cout<<"Passengers: "<<get_pass()<<"\n";
```

```
   cout<<"Type : ";
   switch(get_type())
   {
    case van:
      cout<<"Van\n";break;
    case car:
      cout<<"Car\n";break;
    case wagon:
      cout<<"Wagon\n";break;
   }
 }

void main()   //Begin main program
 {  clrscr();
   cout<<"Display Inheritance from Class vehicle to truck and automobile...."<<endl<<endl;
   truck t1,t2;          //created object...t1 and t2 from class truck
   automobile c;         //created object...c from class automobile
   //used object and called function member
   t1.set_wheels(18);     //use inheritance set_wheels() from based class...vehicle
   t1.set_pass(2);        //use inheritance set_pass() from based class...vehicle
   t1.set_cargo(3200);    //use function set_cargo() from derived class...truck
   t2.set_wheels(6);    //use inheritance set_wheels from based class...vehicle
   t2.set_pass(3);      //use inheritance set_pass() from based class...vehicle
   t2.set_cargo(1200);  //use function set_cargo() from derived class...truck
   t1.show();            //use function show() derived class...truck
   cout<<endl;
   t2.show();            //use function show() derived class...truck
   c.set_wheels(4);  //use inheritance set_wheels() from base class...vehicle
   c.set_pass(6);    //use inheritance set_pass() from base class...vehicle
   c.set_type(van);  //use function set_type() from derived class...automobile
   cout<<endl;
   c.show();        //use function show() from derived class...automobile
   getch();
}
```

ผลการ run โปรแกรม เป็นดังนี้

Display Inheritant from Class vehicle to truck and automobile....

Wheels : 18

Passengers : 2

Cargo capacity in cubic feet : 3200

Wheels : 6

Passengers : 3

Cargo capacity in cubic feet : 1200

Wheels : 4

Passengers: 6

Type : Van

- ***ตัวอย่างโปรแกรม Dra_cir2.cpp แสดงการถ่ายทอดคุณสมบัติ (inheritance) จากเบสคลาส Draw ไปสู่ดีไรฟ์คลาสที่ชื่อ Point***

```
/*Program : Dra_cir2.cpp
  Process : display and hide point from derived class */
#include <iostream.h>
#include <stdlib.h>
#include <graphics.h>
#include <conio.h>

class Draw  //base class
 {
  protected:   //ใช้ได้เฉพาะในเบสคลาสและดีไรฟ์คลาส
    int X, Y, Color;
  public:
    Draw(int InitX,  int InitY,  int InitColor) //construture
      { X=InitX;
        Y=InitY;
        Color=InitColor;
      }


    ~Draw()   //destructure  function
      {
        closegraph();  //close graphic mode
      }
```

```
 };

class Point : public Draw  //derived class
 {
  public:
    Point(int InitX,int InitY,int InitColor):Draw(InitX,InitY,InitColor){}
    void Display();
    void Hide();
  };

void main()
 {
  /* request auto detection graphic display hardware*/
    int gdriver = DETECT, gmode, errorcode;
  /* initialize graphics mode */
  initgraph(&gdriver, &gmode, "");
  //create object from derived class and inheritance attribute
  Point Apoint(320,240,WHITE);  //สร้างออบเจ็กต์ชื่อ Apoint
  Apoint.Display();  //function for display white point on screen
  getch();
  Apoint.Hide();  //function for display background color point on screen
  Apoint.Display();
  getch();
 }

void Point::Display()
{
 putpixel(X,Y,Color);
}
void Point::Hide()
{
 putpixel(X,Y,getbkcolor());
}
```

♦ แบบฝึกหัดท้ายบท

1. ให้นักศึกษาเขียนโปรแกรมแบบ OOP เพื่อคำนวณค่าและแสดงผลลัพธ์พื้นที่ของรูปเรขาคณิต โดยให้สร้างเมมเบอร์ฟังก์ชันของคลาสเพื่อทำหน้าที่คำนวณค่าของพื้นที่และแสดงผล ดังต่อไปนี้
   - พื้นที่รูปสี่เหลี่ยม
   - พื้นที่รูปสามเหลี่ยม
   - พื้นที่วงกลม
2. ให้เขียนโปรแกรมเพื่อจัดเก็บข้อมูลประวัติพนักงานไม่เกิน 100 คน มีรายละเอียดข้อมูลได้แก่ รหัสพนักงาน, ชื่อพนักงาน และเงินเดือน

   โดยกำหนดให้มี member function สำคัญ ๆ ดังนี้
   - Input() ทำหน้าที่รับข้อมูล รหัสพนักงาน, ชื่อพนักงานและเงินเดือน
   - Cal_Net_Tax() ทำหน้าที่คำนวณภาษีหัก ณ ที่จ่ายจากเงินเดือน 7% และคำนวณเงินเดือนคงเหลือสุทธิ
   - Cal_Total() ทำหน้าที่คำนวณเงินเดือนรวม, ภาษีรวม, เงินคงเหลือสุทธิรวมของพนักงานทุกคน
   - Report() ทำหน้าที่แสดงรายงานที่มีรายละเอียดข้อมูล รหัสพนักงาน, ชื่อพนักงาน, เงินเดือน, ภาษีหัก ณ ที่จ่าย, เงินเดือนเหลือสุทธิของพนักงานแต่ละคน และยอดสรุปรวมเงินเดือน, ภาษีหัก ณ ที่จ่ายรวม และเงินเดือนสุทธิรวม
3. ให้เขียนโปรแกรมคำนวณการตัดเกรดแบบอิงเกณฑ์ โดยใช้วิธีการเขียนโปรแกรมแบบ OOP ให้โปรแกรมมีความสามารถ ดังต่อไปนี้
   - รับข้อมูลรหัสนักศึกษา, ชื่อนักศึกษา, คะแนนระหว่างภาค, คะแนนปลายภาค
   - คำนวณการตัดเกรดตามเกณฑ์ คะแนนดังนี้

     0-49 เกรด F

     50-59 เกรด D

     60-69 เกรด C

     70-79 เกรด B

     80-100 เกรด A
   - แสดงรายละเอียดที่กรอกทั้งหมด พร้อมแสดงคะแนนรวมและเกรดที่ได้

ข้อกำหนด ให้โปรแกรมสามารถเก็บข้อมูลนักศึกษาพร้อมกันได้ไม่เกิน 100 คน