OOP & data struct

6.

(1) static variable

(2) Templates

BY SOMSIN THONGKRAIRAT

ATTASIT LASAKUL EDITTION

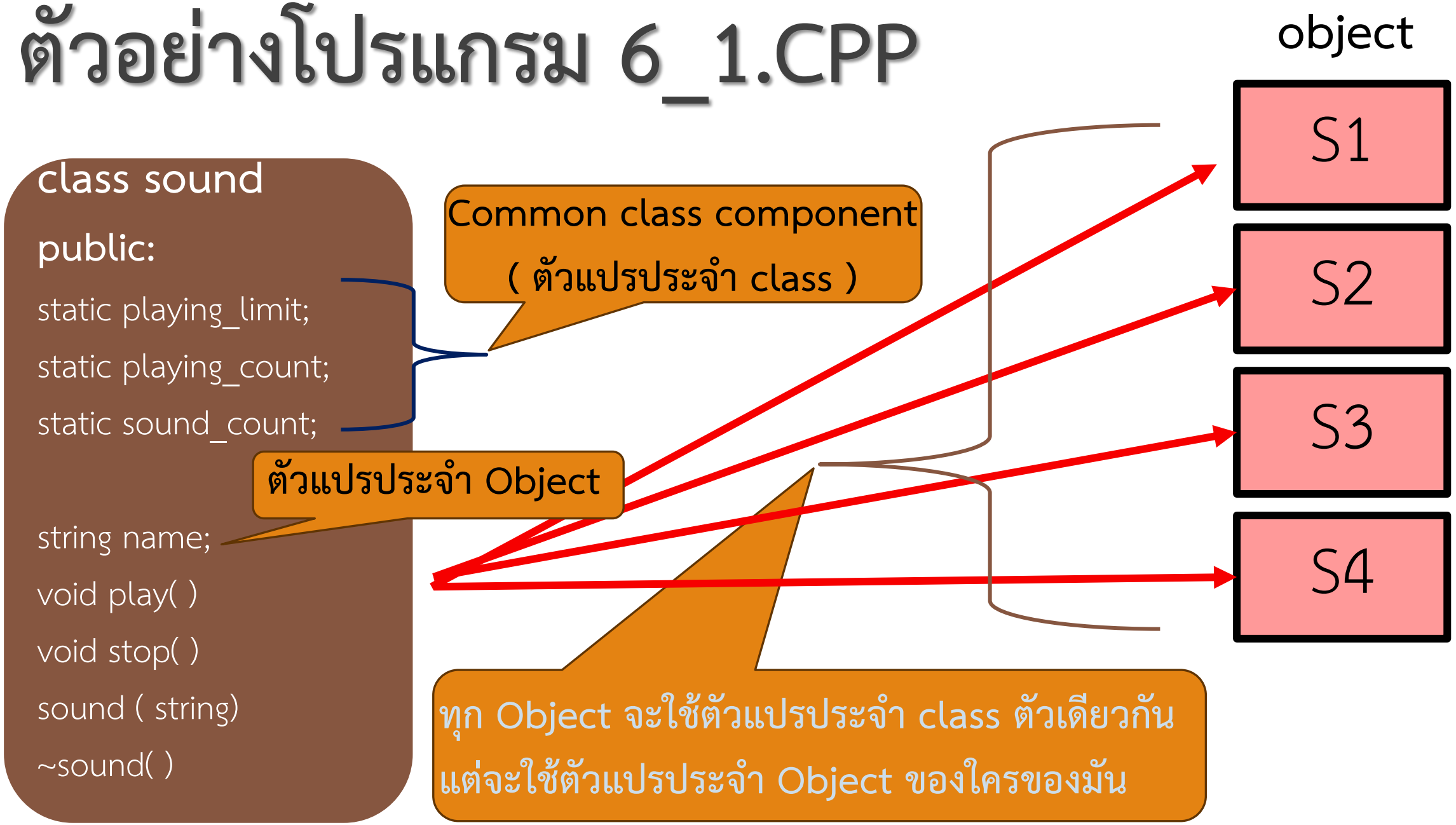
static variable (Common class variable) หรือ เรียกได้อีกอย่างว่า “ตัวแปรประจำ class”

เป็นตัวแปรที่ใช้ได้ทั้งกับ class ต้นฉบับ และ class อื่น (โดยใช้ชื่อคลาสต้นฉบับในการอ้างอิง) หรือตัว Object ที่สืบทอดจากคลาสต้นฉบับนี้ (เหมือนกับ global variable)

** ต้องเขียนไว้ในส่วนของ public ที่คลาสต้นฉบับ

ส่วนตัวแปรอื่นปกติจะเรียกว่า “ตัวแปรประจำ Object”

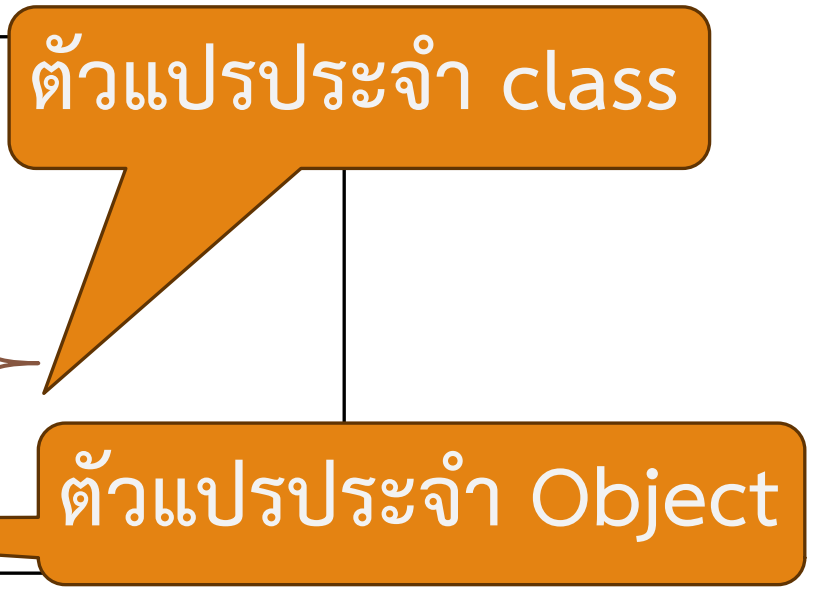
ตัวอย่างโปรแกรม 6_1.CPP



ตัวอย่างโปรแกรม 6_1.CPP

มีกำหนด ตัวแปรแบบ static ดังข้างล่างนี้ *(อย่าลืมว่าเราไม่สามารถกำหนดค่าเริ่มต้นในนี้ได้ทันทีเลย...จำเป็นต้องกำหนดโดยเขียนคำสั่งไว้นอกคลาส)*

```
class sound{
public :
    static int playing_limit;
    static int sound_count;
    string  name;
```



กำหนดค่าเริ่มต้นหรือใช้งานตัวแปรประจำ class ได้ทันทีต้องทำไว้นอกคลาสโดยต้องใส่ :: ไว้หลังชื่อคลาสและตามด้วยชื่อตัวแปร

```
class sound{
      ……….
      ……….
};
int sound::sound_count = 0; // init

int main(){
     sound::sound_count = 120;
    cout << sound::sound_count << endl;
```

กำหนดค่าเริ่มต้นได้ตรงนี้

กำหนดค่า

แสดงค่า

Result : 120

ตัวแปร static ใช้งานได้ปกติเมื่ออยู่ในคลาสนั้น เช่น ในส่วนของ constructure ดังนี้

```
sound(string _name){
        name = _name;
        sound_count++;
 }
```

ตัวแปรประจำ Object

ตัวแปรประจำ class

```
class sound
public:
static playing_limit;
static playing_count;
static sound_count;

string name;
void play( )
void stop( )
sound ( string)
~sound( )
```

```
cout << sound::sound_count << endl;

sound s1("Som san");
sound s2("Jai sang ma");
sound s3("s3");
sound s4("s4");
sound s5("s5");
sound s6("s6");

cout << sound::sound_count << endl;
```

ตัวแปรประจำ class

class sound

public:

static playing_limit;

static playing_count;

static sound_count;

string name;

void play( )

void stop( )

sound (string)

~sound( )

Result : 0

6

มีค่า=6 เพราะเราสร้าง 6 Objects

```
    sound(string _name){
        name = _name;
        sound_count++;
        cout << "this sound number : " << sound_count << endl;
    }
```

main( ) {

```
    cout << sound::sound_count << endl;
    sound s1("Som san");
    sound s2("Jai sang ma");
    sound s3("s3");
    sound s4("s4");
    sound s5("s5");
    cout << sound::sound_count << endl;
```

Result :

0

this sound number : 1

this sound number : 2

this sound number : 3

this sound number : 4

this sound number : 5

5

อย่าลืมว่า Object จะมีขอบเขตของตนเอง

```
cout << sound::sound_count << endl;
sound s1("Som san");
sound s2("Jai sang ma");
sound s3("s3");
if(true){
    sound s7("song yang bad");
}
sound s4("s4");
sound s5("s5");
sound s6("s6");
cout << sound::sound_count << endl;
```

```
0
this sound number : 1
this sound number : 2
this sound number : 3
this sound number : 4
current sound count : 3
this sound number : 4
this sound number : 5
this sound number : 6
6
```

ให้ทดลองรันโปรแกรม 6_1.CPP พิจารณาดูผล...

```
class sound ... {   public:
                        static int sound_count;
                        static int playing_count;
                                ..................... };
int sound::sound_count = 0;    // total number of sounds created
int sound::playing_count = 0; // number of sounds currently playing
int sound::playing_limit = 4; // total number of sounds that can be played at once

int main()
{
    cout << "------(1)----------" << endl;
    cout << "total number of sounds created: ";
    cout << sound::sound_count << endl;
    sound s1("Som san");
    sound s2("Jai sang ma");
    sound s3("s3");
    if (true){sound s7("555");}
    sound s4("s4");
    sound s5("s5");
    sound s6("s6");
    cout << "total number of sounds created: ";
    cout << sound::sound_count << endl;
    cout << "------(2)----------" << endl;
```

ตัวแปรประจำ class

```
------(1)----------
total number of sounds created: 0
this sound number : 1
this sound number : 2
this sound number : 3
this sound number : 4
current sound count : 3
this sound number : 4
this sound number : 5
this sound number : 6
total number of sounds created: 6
------(2)----------
```

ดังนั้น สรุปโปรแกรมที่ 6_1.CPP

- ใส่เพลงในโปรแกรมได้ไม่จำกัด
- สามารถเล่นได้แค่ 4 เพลงพร้อมกัน(playing_limit)
- ลบ/เพิ่ม เพลงได้ (play( ) , stop( ))
- แสดงจำนวนเพลงทั้งหมด (sound_count)
- แสดงจำนวนเพลงที่กำลังเล่นพร้อมกัน (playing_count)

(2) Templates

สามารถใช้เพื่อสร้าง function ที่กำหนดชนิดของข้อมูล (datatype) ที่ใช้ใน function ได้แบบอัตโนมัติ

ตัวอย่างโปรแกรม 6_2.CPP

```
int multiply(int a, int b)
{
    int c = a * b;
    return c;
}

float multiply(int a, float b)
{
    float c = a * b;
    return c;
}

float multiply(float a, int b)
{
    float c = a * b;
    return c;
}
```

ฟังก์ชั่น ทำงานเหมือนกัน
แต่ว่าใช้กับชนิดข้อมูลคนละแบบ
(Overloading methods)

ทำเป็นรูปแบบ (Template) ให้ใช้ร่วมกันได้

```
template <typename myType>
myType multiple(myType a, myType b)
{
    myType c = a * b;
    return c;
}

template <typename T, typename U>
T multiple(T a, U b)
{
    T c = a * b;
    return c;
}
```

รูปแบบที่มีตัวแปรตัวเดียวเหมือนกัน

รูปแบบที่มีตัวแปรหลายตัว

การใช้งานจากตัว main( ) กรณีตัวแปรตัวเดียว

```
int main()
{

    int a = multiple<int>(3, 4);
    float b = multiple<float>(3, 5);
    double c = multiple<double>(3, 6);
```

การใช้งานจากตัว main( ) กรณีตัวแปรหลายตัว

```
int d = multiple<float, int>(3, 7);
float e = multiple<float, float>(3, 8);
double f = multiple<double, float>(3, 9);
```

ผลลัพธ์การทำงาน (ตัวแปรชนิดเดียว)

```
    int a = multiple<int>(3, 4);
    float b = multiple<float>(3.2, 5.8);
    double c = multiple<double>(3, 6);
    system("cls");
    cout << a << endl;
    cout << b << endl;
    cout << c << endl;
cout << "-----(1)-------------" << endl;
```

```
12
18.56
18
-----(1)-------------
```

ผลลัพธ์การทำงาน (หลายชนิดของตัวแปร)

```
    int d = multiple<int, int>(3.3, 7);
    float e = multiple<float, float>(3.5, 8.2);
    double f = multiple<double, float>(3.4, 9.2);

    cout << d << endl;
    cout << e << endl;
    cout << f << endl;
cout << "-----(2)--------------" << endl;
```

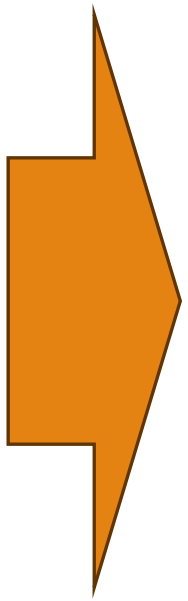
```
21
28.7
31.28
-----(2)----
```

Template ช่วยลดจำนวนฟังก์ชั่นลงได้

```
int multiply(int a, int b)
{
    int c = a * b;
    return c;
}

float multiply(int a, float b)
{
    float c = a * b;
    return c;
}

float multiply(float a, int b)
{
    float c = a * b;
    return c;
}
```



```
template <typename myType>
myType multiple(myType a, myType b)
{
    myType c = a * b;
    return c;
}

template <typename T, typename U>
T multiple(T a, U b)
{
    T c = a * b;
    return c;
}
```

การใช้ Template กับ ฟังก์ชั่น (Program 6_3.CPP)

```
int main(){

    int my_int[] = {5,6,8,4,1,3,9,11,2};
    char my_char[] = {'e','a','i','u','o'};
    double my_double[] = {5.3,6.4,8.3,6.5,9.3,4.4,1.2,2.6};
cout<< "----- before sort -----" << endl;
    print_array<int>(my_int,sizeof(my_int)/sizeof(int));
    print_array<char>(my_char,sizeof(my_char)/sizeof(char));
    print_array<double>(my_double,sizeof(my_double)/sizeof(double));

cout<< "----- after sort -----" << endl;
    selection_sort<int>(my_int,sizeof(my_int)/sizeof(int));
    print_array<int>(my_int,sizeof(my_int)/sizeof(int));

    selection_sort<char>(my_char,sizeof(my_char)/sizeof(char));
    print_array<char>(my_char,sizeof(my_char)/sizeof(char));

    selection_sort<double>(my_double,sizeof(my_double)/sizeof(double));
    print_array<double>(my_double,sizeof(my_double)/sizeof(double));

    return 0;
}
```

Result :

```
----- before sort -----
5 6 8 4 1 3 9 11 2
e a i u o
5.3 6.4 8.3 6.5 9.3 4.4 1.2 2.6
----- after sort -----
1 2 3 4 5 6 8 9 11
a e i o u
1.2 2.6 4.4 5.3 6.4 6.5 8.3 9.3
```

```cpp
template <typename T>
void selection_sort(T a[],int n){
    for(int i=0;i<n;i++){
        int min_idx = i;
        for(int j=i;j<n;j++){
            if(a[j] < a[min_idx]){
                min_idx = j;
            }
        }
        T tmp = a[min_idx];
        a[min_idx] = a[i];
        a[i] = tmp;
    }
}
```

อันนี้เป็นฟังก์ชั่นเรียงตัวเลข
จากน้อยไปมาก (Sort number)
เราใช้คำสั่ง Template
กำหนดให้ชนิดตัวแปรเป็นอะไร
ก็ได้ แทนที่ในตัวอักษร T

```cpp
template <typename T>
void print_array(T a[],int n){
    for(int i=0;i<n;i++){
        cout << a[i] << " ";
    }
    cout << endl;
}
```

อันนี้ก็เช่นเดียวกันเป็นฟังก์ชั่นแสดงผล Array เราใช้คำสั่ง Template กำหนดให้ชนิดตัวแปรเป็นอะไรก็ได้ แทนที่ในตัวอักษร T

```cpp
int main(){

    int my_int[] = {5,6,8,4,1,3,9,11,2};
    char my_char[] = {'e','a','i','u','o'};
    double my_double[] = {5.3,6.4,8.3,6.5,9.3,4.4,1.2,2.6};
cout<< "----- before sort -----" << endl;
    print_array<int>(my_int,sizeof(my_int)/sizeof(int));
    print_array<char>(my_char,sizeof(my_char)/sizeof(char));
    print_array<double>(my_double,sizeof(my_double)/sizeof(double));

cout<< "----- after sort -----" << endl;
    selection_sort<int>(my_int,sizeof(my_int)/sizeof(int));
    print_array<int>(my_int,sizeof(my_int)/sizeof(int));

    selection_sort<char>(my_char,sizeof(my_char)/sizeof(char));
    print_array<char>(my_char,sizeof(my_char)/sizeof(char));

    selection_sort<double>(my_double,sizeof(my_double)/sizeof(double));
    print_array<double>(my_double,sizeof(my_double)/sizeof(double));

    return 0;
}
```

Result :

```
----- before sort -----
5 6 8 4 1 3 9 11 2
e a i u o
5.3 6.4 8.3 6.5 9.3 4.4 1.2 2.6
----- after sort -----
1 2 3 4 5 6 8 9 11
a e i o u
1.2 2.6 4.4 5.3 6.4 6.5 8.3 9.3
```

แม้แต่ตัว class เอง เราก็สามารถใช้ Template เพื่อกำหนดชนิดของตัวแปรได้ ดังนี้ (เป็นตัวอย่างไว้ใช้ตอนทำ LAB ได้)

```
template <typename T>
class prototype_calculator
{

public:
    virtual T add(T x, T y) = 0;
    virtual T subtract(T x, T y) = 0;
    virtual T multiply(T x, T y) = 0;
    virtual T sum(T x1, T x2 = 0, T x3 = 0,
                  T x4 = 0, T x5 = 0, T x6 = 0,
                  T x7 = 0, T x8 = 0, T x9 = 0, T x10 = 0) = 0;
};
```

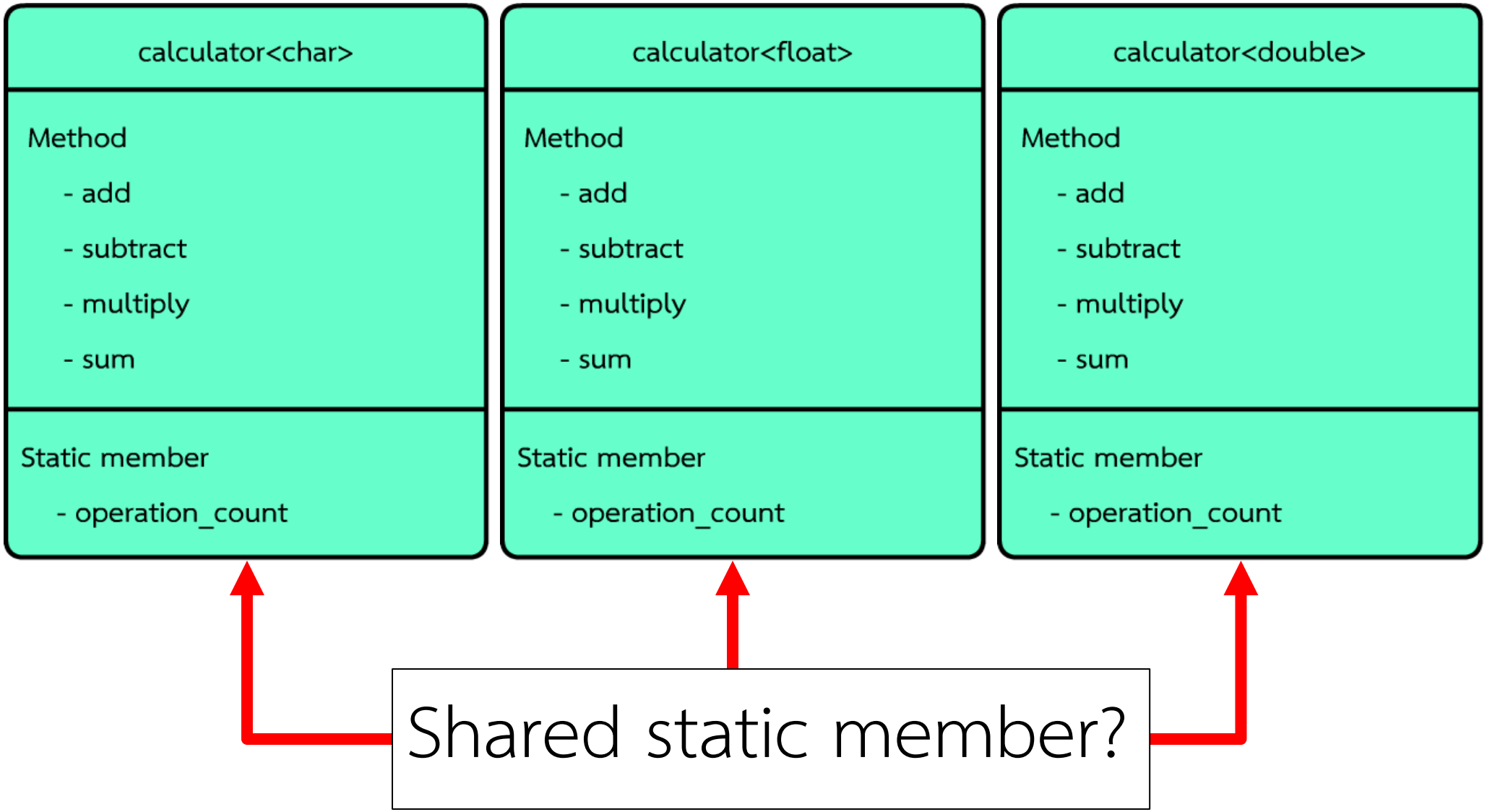
และถ้าเราสร้าง class universal_calculator สืบทอดจาก prototype_calculator ดังนี้ โดยมีเพิ่มตัวแปร static ด้วย

```
template <typename T>
class universal_calculator : public prototype_calculator<T>
{
public:
    static int operate_count;

    T add(T x, T y)
    {
        operate_count++;
        return x + y;
    }
    T subtract(T x, T y)
    {
        operate_count++;
        return x - y;
    }
```

- เมื่อเราสร้าง Object ใช้งานก็จะได้ต่างชนิดกัน
- และตัวแปร static ก็จะเป็นของใครของมัน

```
int main()
{
    universal_calculator<int> int_calculator;
    universal_calculator<float> float_calculator;
    universal_calculator<double> double_calculator;
    universal_calculator<char> char_calculator;
```


calculator<char>
Method
- add
- subtract
- multiply
- sum
Static member
- operation_count
calculator<float>
Method
- add
- subtract
- multiply
- sum
Static member
- operation_count
calculator<double>
Method
- add
- subtract
- multiply
- sum
Static member
- operation_count
Shared static member?

เวลาเรียกใช้ก็จะเป็นของใครของมัน

```
universal_calculator<int>::operator_count
universal_calculator<float>::operator_count
universal_calculator<double>::operator_count
universal_calculator<char>::operator_count
```

LAB

1) เติมช่องว่าง ใน PDF ไฟล์ (โปรแกรม 6_1.CPP)
2) ปรับแก้ไขโปรแกรมให้แสดงผลลัพธ์ตามโจทย์ 6_4_LAB.CPP กำหนด