

# C++언어 10강

강사 | 최정호

# INDEX

1. Friend
2. 연산자 오버로딩
3. 템플릿

**Friend**

## Friend

- **Friend 멤버 함수** : 전역 함수를 Friend 선언화 함으로써 멤버 함수처럼 **private 멤버 변수**에 접근이 가능하다.
- **Friend Class** : 다른 Class를 마치 **자신의 Class** 처럼 **private**에 접근할 수 있다.
- **Friend 전역 함수** : 자신의 멤버함수를 **전역 함수로 변환**시킬 수 있다.

=====

### ※ 장단점

#### -장점

1. 연산자 오버로딩등 예외적인 경우에 필요하다.
2. Class사용 하는 측면에서 추가적으로 private 영역에 접근하는 함수를 만들지않아도 되서 편리하다.

#### -단점

1. 객체 지향개념이 모호해진다.
2. Class간의 의존도를 높혀 유연성있는 코드가 아니게 된다.

## >> Friend 멤버 함수

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

class A
{
private:
    int x, y;
public:
    friend void Setxy(A& a);
    A() { x = 0; y = 0; }
    void Showxy()
    {
        cout << "x = " << x << endl;
        cout << "y = " << y << endl;
    }
};

void Setxy(A& a)
{
    cout << "x, y좌표 입력 : " << endl;
    cin >> a.x >> a.y;
}

void main()
{
    A a;
    a.Showxy();
    Setxy(a);
    a.Showxy();
}
```

## >> Friend Class

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

```
class A
{
private:
    int x, y;

public:
    friend class B;
    A(int a, int b) : x(a), y(b) { }

    void Showxy()
    {
        cout << "x = " << x << endl;
        cout << "y = " << y << endl;
    }
};

class B
{
private:
    int x, y;

public:
    B() : x(0), y(0) {}

    void GetA(A &a)
    {
        x = a.x; y = a.y;
    }

    void Showxy()
    {
        cout << "x = " << x << endl;
        cout << "y = " << y << endl;
    }
};
```

```
void main()
{
    A a(10, 15);
    B b;
    a.Showxy();
    b.Showxy();
    b.GetA(a);
    a.Showxy();
    b.Showxy();
}
```

# Friend

## >> Friend 전역 함수

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

```
class A
{
private:
    int x, y;
public:
    A(int a, int b) : x(a), y(b) { }
    friend void Showxy(A& a)
    {
        cout << "x = " << a.x << endl;
        cout << "y = " << a.y << endl;
    }
};

void main()
{
    A a(10, 15);
    Showxy(a);
}
```

**연산자 오버로딩**



## 연산자 오버로딩

- 객체를 대상으로 직접 연산할 수 있도록 연산자를 재정의 한다.
- 기존 연산자의 의미를 유지해야 한다.(ex.  $+$   $\rightarrow$   $/(x)$ )
- 연산자의 우선순위 변경이 불가능하다.
- 디폴드 매개변수를 가질 수 없다.
- 연산에 사용되는 피연산자 중 하나 이상은 객체이어야 한다.

01

02

03

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

```
void main()  
{  
    cout << 3 + 4 << endl;  
    cout << "korea" + 1 << endl;  
    cout << "korea" + 2 << endl;  
    cout << "korea" + 3 << endl;  
    cout << 4 + "korea" << endl;  
    //cout << "kor" + "ea" << endl;  
}
```

01

02

03

## >> 오버로딩이 불가능한 연산자

연산자	설명
.	멤버 접근 연산자
.*	멤버 지시 포인터
::	범위 지정 연산자
? :	조건 연산자

## &gt;&gt; 객체 + 정수

```
#include <iostream>
using namespace std;
class Point
{
private:
    int x, y;
public:
    Point(int a, int b) { x = a; y = b; }
    void Print() { cout << "x = " << x << ", y = " << y << "\n"; }
    friend Point operator + (Point tmp, int data);
};
Point operator + (Point tmp, int data)
{
    cout << "객체 + 정수" << "\n";
    tmp.x = tmp.x + data;
    tmp.y = tmp.y + data;
    return tmp;
}
void main()
{
    Point ov1(10, 20), ov2(0, 0);
    ov2 = ov1 + 10;
    ov2.Print();
}
```

## &gt;&gt; 객체 == 객체

```
#include <iostream>
using namespace std;
class Point
{
private:
    int x, y;
public:
    Point(int a, int b) { x = a; y = b; }
    void Print() { cout << "x = " << x << ", y = " << y << "\n"; }
    bool operator == (Point tmp);
};
bool Point::operator == (Point tmp)
{
    if ((this->x == tmp.x) && (this->y == tmp.y))
        return true;
    else
        return false;
}
void main()
{
    Point ov1(10, 20), ov2(10, 20);
    if (ov1 == ov2)
        cout << "같다." << "\n";
}
```

01

02

03

**>> 증감연산자**

```
#include<iostream>
using namespace std;
class Count
{
    int cnt1;
    int cnt2;
public:
    Count() { cnt1 = 0; cnt2 = 0; }
    Count(int n1, int n2) { cnt1 = n1; cnt2 = n2; }
    int GetCnt1() { return cnt1; }
    int GetCnt2() { return cnt2; }
    void operator ++() { ++cnt1; ++cnt2; }
    void operator --() { --cnt1; --cnt2; }
};
void main()
{
    Count co1(5,10), co2;
    ++co1;
    --co2;
    cout << "co1.cnt1 : " << co1.GetCnt1() << "\tco2.cnt2 : " << co1.GetCnt2() << endl;
    cout << "co2.cnt1 : " << co2.GetCnt1() << "\tco2.cnt2 : " << co2.GetCnt2() << endl;
}
```

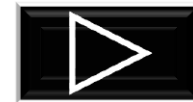
## Quiz

- 객체 / 객체 연산자 오버로딩을 작성 하시오.  
(ex. 작은수에서 큰수로 나눌 수있게 예외처리)
- Time Class를 만들어 객체 + 객체 가 가능한 연산자 오버로딩을 작성 후 프로그램을 완성하시오.

(ex.예시 Time class)

```
class Time
{
private:
    int m_iHour;
    int m_iMin;

public:
    void ShowTime();
    Time operator + (Time time);
    Time();
    Time(int Hour,int Min);
    ~Time();
};
```



(실행파일)

**템플릿**



## 템플릿

- **자료형과 무관하게 처리**할 수 있는 함수, Class를 만들 수 있다.
- 데이터 타입을 인자로 전달 할 수 있으며 함수, Class의 **일반화**가 가능하다.

## >> 템플릿 함수

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

void Sum(int su1, int su2)
{
    int sum;
    sum = su1 + su2;
    cout << sum << endl;
}

void main()
{
    int a = 1, b = 2;
    Sum(a, b);
}
```

## >> 템플릿 함수

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

template <typename Type>
void Sum(Type su1, Type su2)
{
    Type sum;
    sum = su1 + su2;
    cout << sum << endl;
}

void main()
{
    int a = 1, b = 2;
    Sum(a, b);
}
```

## >> 템플릿 함수

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

template <typename Type>
void Add(Type n1, Type n2);
void main()
{
    int su1, su2;
    double num1, num2;
    cout << "두 정수 입력 : ";
    cin >> su1 >> su2;
    cout << "두 실수 입력 : ";
    cin >> num1 >> num2;
    Add(su1, su2);
    Add(num1, num2);
}

template <typename Type>
void Add(Type n1, Type n2)
{
    Type sum;
    sum = n1 + n2;
    cout << "n1 + n2 = " << sum ;
    cout << endl;
}
```

## >> 템플릿 함수

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

template <typename t>
t Big(t x, t y)
{
    if (x > y)
        return x;
    else
        return y;
}

void main()
{
    cout << Big(20, 10) << endl;
    cout << Big(2.2, 1.1) << endl;
    cout << Big('z', 'a') << endl;
    cout << Big("def", "abc") << endl;
    cout << Big(4.4f, 3.3f) << endl;
}
```

## >> 템플릿 함수

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

template <typename t1, typename t2>
t1 func(t2 num)
{
    cout << num << endl;
    return num;
}

void main()
{
    double su = 10.2;
    //func(su);
    cout << func<int>(su) << endl;
}
```

## >> 템플릿 함수

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

template <typename val>
void func(val v)
{
    cout << "Template Function \t" << v << endl;
}

template <>
void func<char>(char v)
{
    cout << "Specialization Template Function \t" << v << endl;
}

void main()
{
    func(10);
    func(10.1);
    func(0x10);
    func('a');
    func("abc");
}
```

## Quiz

- 전달 받은 값을 1증가 시켜주는 템플릿 함수를 만드시오
- 두 수를 입력 받아 최소값을 구하는 템플릿 함수를 만드시오
- 세 수를 입력 받아 최대값을 구하는 템플릿 함수를 만드시오



(실행파일)



>> 템플릿 함수

01

02

03

• **템플릿\_연산자오버로딩 예제.txt 참고**

## >> 템플릿 Class

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

template <typename Type>
class Point
{
private:
    Type x, y;
public:
    Point(Type a, Type b) { x = a; y = b; }
    void Display()
    {
        cout << "x : " << x << "\n";
        cout << "y : " << y << "\n";
    }
};

void main()
{
    Point<int> PInt(10, 20);
    Point<double> PDouble(10.4, 20.6);
    PInt.Display();
    PDouble.Display();
}
```

## >> 템플릿 Class

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

template <typename Type>
class Point
{
private:
    Type x, y;
public:
    Point(Type a, Type b) { x = a; y = b; }
    void Display();
};

template <typename Type>
void Point<Type>::Display()
{
    cout << "x : " << x << "\n";
    cout << "y : " << y << "\n";
}

void main()
{
    Point<int> PInt(10, 20);
    Point<double> PDouble(10.4, 20.6);
    PInt.Display();
    PDouble.Display();
}
```

## >> 템플릿 Class

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

template <typename A, typename B>
class Size
{
private:
    int a, b;
public:
    Size() { a = sizeof(A); b = sizeof(B); }
    void print()
    {
        cout << "a = " << a << ", b = " << b << endl;
    }
};

void main()
{
    Size <char, int> si1;
    Size <float, double> si2;
    si1.print();
    si2.print();
}
```

## >> 템플릿 Class

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

template <typename T>
class STACK
{
private:
    T* top; int size;
public:
    STACK(int s) { size = s; top = new T[size]; }
    void push(T a) { *top = a; top++; }
    void pop() { cout << *--top << endl; }
};

void main()
{
    STACK <float> stack(3);
    stack.push(2.2f);
    stack.push(1.1f);
    stack.pop();
    stack.pop();
}
```

**Thank you**

강사 | 최정호