

# C++ Programming

11<sup>th</sup> Study: Object-Oriented  
Programming (7/8)

- Operator overloading



C++ Korea 옥찬호 (utilForever@gmail.com)

# Operator overloading

A specific case of polymorphism, where different operators have different implementations depending on their arguments

# Operator overloading

- 연산자 오버로딩(Operator overloading)?
  - C나 C++에서 기본으로 제공하는 연산자를 함수 처럼 매개변수의 개수나 자료 형에 따라 오버로딩 하여 사용할 수 있도록 하는 것.
  - +, -, \*, / 등의 기본 이항 연산자부터 =, ->, \* 등의 연산자 까지 대부분의 연산자에 대해 오버로딩이 가능하다.
  - 단, operator.(멤버 참조 연산자), operator?: (조건 연산자), operator:: (범위 결정 연산자), operator.\* (멤버 포인트 연산자)는 오버로딩이 불가능하다.

# Operator overloading

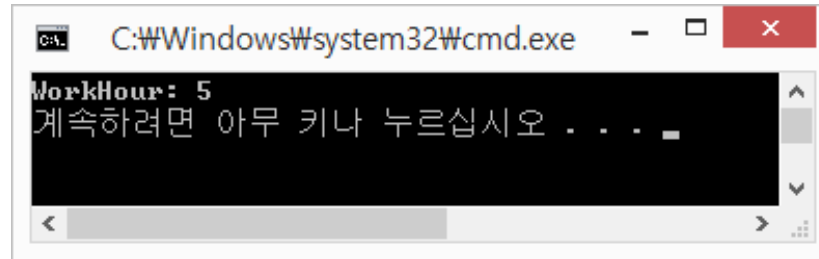
- 연산자 오버로딩의 문법적 표현
  - `Return_type operator+(parameter);`
- 예
  - `WorkHour operator+ (const WorkHour& workobj);`
  - `Const ,&`로 파라미터를 넘기는 이유는 복사에 의한 오버헤드를 줄이고, `workobj`의 안정성을 유지하기 위해

# Operator overloading: Example

```
class WorkHour {  
private:  
    int workHour;  
public:  
    WorkHour(int w) : workHour(w)  
    {}  
  
    void printWorkHour()  
    {  
        cout << "WorkHour: " << workHour << endl;  
    }  
  
    WorkHour add(const WorkHour& work) {  
        WorkHour wh(this->workHour + work.workHour);  
        return wh;  
    }  
};
```

```
int main() {  
    WorkHour Aworker(2);  
    WorkHour Bworker(3);  
    WorkHour total = Aworker.add(Bworker);  
    total.printWorkhour();  
}
```

# Operator overloading: Example

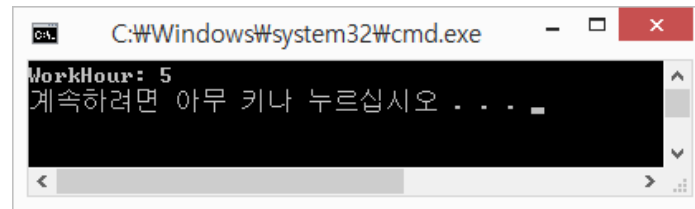


# Operator overloading: Example

```
class WorkHour {  
private:  
    int workHour;  
public:  
    WorkHour(int w) : workHour(w)  
    {}  
  
    void printWorkHour()  
    {  
        cout << "WorkHour: " << workHour << endl;  
    }  
  
    WorkHour add(const WorkHour& work) {  
        WorkHour wh(this->workHour + work.workHour);  
        return wh;  
    }  
  
    WorkHour operator+(const WorkHour& work) {  
        WorkHour wh(this->workHour + work.workHour);  
        return wh;  
    }  
};
```

```
int main() {  
    WorkHour Aworker(2);  
    WorkHour Bworker(3);  
    WorkHour total = Aworker + Bworker;  
    total.printWorkhour();  
}
```

# Operator overloading: Example



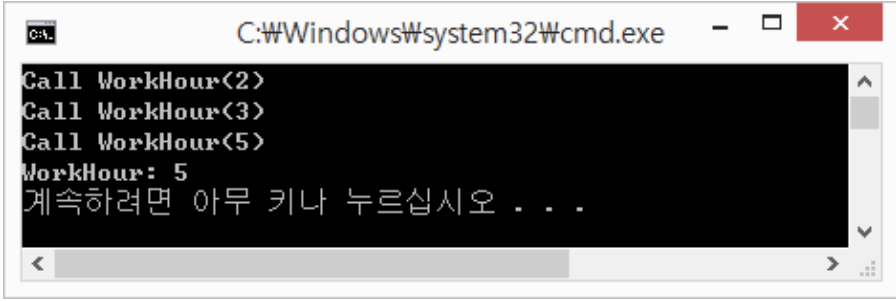


# Operator overloading: Example

```
class WorkHour {  
private:  
    int workHour;  
public:  
    WorkHour(int w) : workHour(w)  
    {  
        cout << "Call WorkHour(" << w << ")" endl;  
    }  
  
    void printWorkHour()  
    {  
        cout << "WorkHour: " << workHour << endl;  
    }  
  
    WorkHour add(const WorkHour& work) {  
        WorkHour wh(this->workHour + work.workHour);  
        return wh;  
    }  
  
    WorkHour operator+(const WorkHour& work) {  
        WorkHour wh(this->workHour + work.workHour);  
        return wh;  
    }  
};
```

```
int main() {  
    WorkHour Aworker(2);  
    WorkHour total = Aworker + 3;  
    total.printWorkhour();  
}
```

# Operator overloading: Example



```
C:\Windows\system32\cmd.exe
Call WorkHour<2>
Call WorkHour<3>
Call WorkHour<5>
WorkHour: 5
계속하려면 아무 키나 누르십시오 . . .
```

The screenshot shows a Windows command prompt window titled "C:\Windows\system32\cmd.exe". The window contains the following text: "Call WorkHour<2>", "Call WorkHour<3>", "Call WorkHour<5>", "WorkHour: 5", and "계속하려면 아무 키나 누르십시오 . . .". The text is displayed in a black font on a white background. The window has a standard Windows title bar with a minimize button, a maximize button, and a close button.

# Operator overloading: Example

```
class WorkHour {  
private:  
    int workHour;  
public:  
    WorkHour(int w) : workHour(w)  
    {  
        cout << "Call WorkHour(" << w << ")" endl;  
    }  
  
    void printWorkHour()  
    {  
        cout << "WorkHour: " << workHour << endl;  
    }  
  
    WorkHour operator+(const WorkHour& work) {  
        WorkHour wh(this->workHour + work.workHour);  
        return wh;  
    }  
};
```

```
int main() {  
    WorkHour Aworker(2);  
    WorkHour total = 3 + Aworker;  
    total.printWorkhour();  
}
```

# Operator overloading: Example

error: C2677: 이항 '+' : 'WorkHour' 형식을 사용하는 전역 연산자가 없거나 허용되는 변환이 없습니다.

# Operator overloading: Example

```
class WorkHour {
private:
    int workHour;
public:
    WorkHour(int w) : workHour(w)
    {
        cout << "Call WorkHour(" << w << ")" endl;
    }

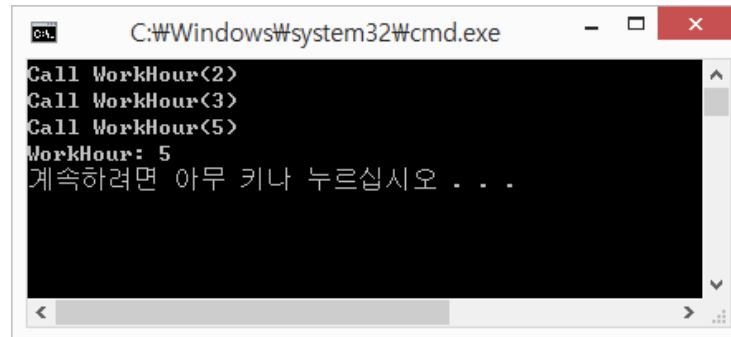
    void printWorkHour()
    {
        cout << "WorkHour: " << workHour << endl;
    }

    friend WorkHour operator+(const WorkHour& work1,
        const WorkHour& work2);
};

WorkHour operator+(const WorkHour& work1, const WorkHour&
work2) {
    WorkHour work(work1.workHour + work2.workHour);
    return work;
}
```

```
int main() {
    WorkHour Aworker(2);
    WorkHour total = 3 + Aworker;
    total.printWorkhour();
}
```

# Operator overloading: Example



A screenshot of a Windows command prompt window titled "C:\Windows\system32\cmd.exe". The window has a black background with white text. The text shows three lines of C++ code: `Call WorkHour(2)`, `Call WorkHour(3)`, and `Call WorkHour(5)`. Below the code, the output `WorkHour: 5` is displayed. At the bottom, there is a line of Korean text: `계속하려면 아무 키나 누르십시오 . . .`. The window includes standard Windows window controls (minimize, maximize, close) in the top right corner and a scrollbar on the right side.

```
C:\Windows\system32\cmd.exe
Call WorkHour(2)
Call WorkHour(3)
Call WorkHour(5)
WorkHour: 5
계속하려면 아무 키나 누르십시오 . . .
```

# Operator overloading: Example

```
class WorkHour {
private:
    int workHour;
public:
    WorkHour(int w) : workHour(w)
    {

    }

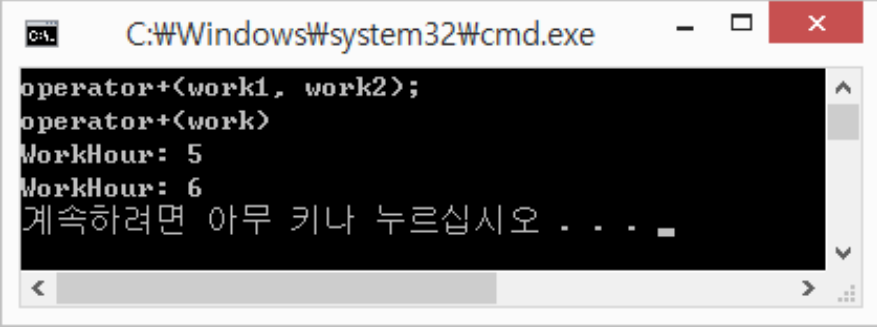
    void printWorkHour()
    {
        cout << "WorkHour: " << workHour << endl;
    }

    WorkHour operator+(const WorkHour& work) {
        WorkHour wh(this->workHour + work.workHour);
        cout << "operator+(work)" << endl;
        return wh;
    }
    friend WorkHour operator+(const WorkHour& work1,
        const WorkHour& work2);
};
```

```
WorkHour operator+(const WorkHour& work1, const WorkHour&
work2)
{
    WorkHour work(work1.workHour + work2.workHour);
    cout << "operator+(work1, work2);" << endl;
    return work;
}

int main() {
    WorkHour Aworker(2);
    WorkHour Bworker(3)
    WorkHour total1 = 3 + Aworker;
    WorkHour total2 = Bworker + 3
    total1.printWorkhour();
    total2.printWorkhour();
}
```

# Operator overloading: Example



A screenshot of a Windows command prompt window titled "C:\Windows\system32\cmd.exe". The window has a black background with white text. The text displayed is as follows:

```
operator+(work1, work2);  
operator+(work)  
WorkHour: 5  
WorkHour: 6  
계속하려면 아무 키나 누르십시오 . . .
```

The text is left-aligned. The first two lines are code, and the next three lines are output. The last line is a Korean prompt for the user to press a key to continue.