



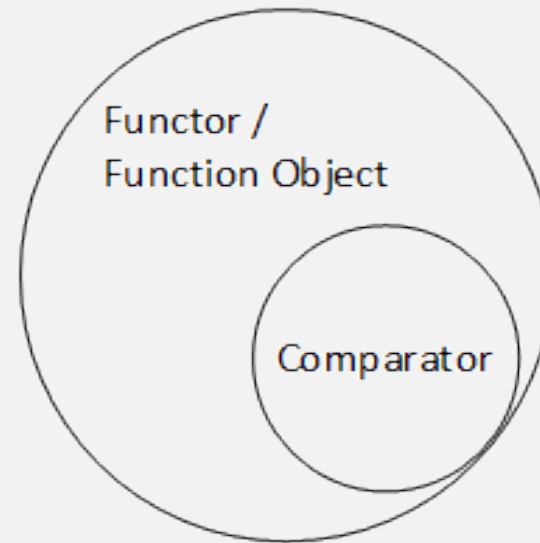
operator()

An Introduction To Functors



PARENTHESES

- C++ implements three operations with parentheses
 - Grouping
 - Casting
 - Functions
- `operator()`
 - Function objects
 - Functors





OVERLOADING operator()

- `return-type operator()`(parameter-list)
- `return-type result = foo(args)`
- `bool operator()`()
- `bool operator()`(const Employee& e1, const Employee& e2) const



THE NPL PSEUDO RANDOM NUMBER GENERATOR

```
class NPL
{
    public:
        double  operator()( );
        long    operator()(long max);
};

long NPL::operator()(long max)
{
    return (long)fmod(max * operator()( ), max);
    //return (long)fmod(max * (*this)(), max);
}
```



USING FUNCTORS

```
int main()
{
    NPL npl(41, 67, 91);

    for (int i = 100; i < 110; i++)
        cout < npl() << " " << npl(i) << endl;

    return 0;
}
```



APPLICATION CLASS

```
class Employee
{
    private:
        string  name;
        int     position;
        int     id;
        .
        .
        .
}
```



COMPARATOR CLASS I

```
class compByName
{
    public:
        bool operator()(const Employee& e1, const Employee& e2) const
        {
            return e1.getName() <= e2.getName();
        }
};
```



COMPARATOR CLASS 2

```
class compByNumber
{
    public:
        bool operator()(const Employee& e1, const Employee& e2) const
        {
            if (e1.getPosition() == e2.getPosition())
                return e1.getID() <= e2.getID();
            return e1.getPosition() <= e2.getPosition();
        }
};
```




COMPARATORS AS FUNCTION ARGUMENTS

```
cout << min(e1, e2, compByNumber()) << endl;
```

```
cout << min(e1, e2, compByName()) << endl;
```