

C++: Constness

Miro Jurišić
meeroh@meeroh.org

Why const?

- Symbolic constants
- Immutable objects
- Read-only data

const T

```
int x = 0;          // OK
x = 1;             // OK

const int y = 1;   // OK
y = 2;            // Error
```

Pointers and constness

- `const T*`
- `T const*`
- `T* const`

```
const char* x = "x";  
char const* y = "y";  
char* const z = "z";
```

```
x = y;           // OK  
y = z;           // OK  
z = x;           // Error
```

```
x [0] = 'a';     // Error  
y [0] = 'a';     // Error  
z [0] = 'a';     // OK
```

```
char a = 'a';  
const char* a1 = &a;    // OK  
char const* a2 = &a;    // OK  
char* const a3 = &a;    // OK
```

```
const char b = 'b';  
const char* b1 = &b;    // OK  
char const* b2 = &b;    // OK  
char* const b3 = &b;    // Error
```

References and constness

- `const T&`, `T const&`
- `T&` `const`

```
char a;  
const char& a1 = a;    // OK  
char const& a2 = a;   // OK  
  
a1 = 'b';             // Error  
a2 = 'b';             // Error
```

```
const char b = 'b';  
const char& b1 = b;   // OK  
char const& b2 = b;   // OK  
char& b3 = b;         // Error  
  
b1 = 'b';             // Error  
b2 = 'b';             // Error
```

More pointers and constness

- `const T* const`
- `T const* const`
- `const T**`
- `T const**`
- `T* const*`
- `T** const`
- `const T* const*`
- `T const* const*`
- `const T** const`
- `T const** const`
- `T* const* const`
- `const T* const* const`
- `T const* const* const`

Constness and function arguments

```
void g (char& p)
{
    p = 'a';          // OK
}
```

```
void f (const char& p)
{
    p = 'a';          // Error
}
```

```
int main ()
{
    char c1 = 'a';
    const char c2 = 'b';

    f (c1);           // OK
    f (c2);           // OK
    g (c1);           // OK
    g (c2);           // Error
}
```

Constness of member functions

```
class C {
    public:
        void f () const;
        void g ();

    private:
        int i;
};

void C::f () const
{
    i = 2;          // Error
}

int main ()
{
    C c1;
    const C c2;

    c1.f ();      // OK
    c1.g ();      // OK
    c2.f ();      // OK
    c2.g ();      // Error
}
```

Mutability

- Immutability of abstraction vs. immutability of representation

```
class C {
    public:
        void f () const;
        void g ();

    private:
        mutable int i1;
        int i2;
};

void C::f () const
{
    i1 = 2;    // OK
    i2 = 2;    // Error
}

int main ()
{
    C c1;
    const C c2;

    c1.f ();    // OK
    c1.g ();    // OK
    c2.f ();    // OK
    c2.g ();    // Error
}
```

Constness and unnamed temporaries

- Unnamed temporary objects: `f (g ())`
- Implicit conversions: `f (x)`, `f (T (x))`

```
int f ()
{
    return 1;
}

void g (const int& c);
void h (int& c);

int main ()
{
    g (f ());          // OK
    h (f ());          // Error
}
```

Constness and function return values

```
class C {
    public:
        void f1 ();
        void f2 () const;
};

C g1 ();
const C g2 ();

int main ()
{
    g1 ().f1 ();           // OK
    g1 ().f2 ();           // OK

    g2 ().f1 ();           // Error
    g2 ().f2 ();           // OK
}
```

Constness of postfix increment

- Always make postfix operators return const

```
class C {  
    public:  
        C operator ++ ();  
};
```

```
class D {  
    public:  
        const D operator ++ ();  
};
```

```
int main ()  
{  
    C c;  
    c++++;    // OK, but wrong  
  
    D d;  
    d++++;    // Error  
}
```

Overloading on constness

```
class C {
    public:
        void f ();
        void f () const;
};

int main ()
{
    C c1;
    const C c2;

    c1.f ();          // Non-const f
    c2.f ();          // const f

    const C& c3 = c1;
    c3.f ();          // const f!
}
```