C++: const methods

- Version 1: Dr. Ofir Pele
- Version 2: Dr. Miri Ben-Nissan
- Version 3: Dr. Erel Segal-Halevi

Reminder: const variables (in C and C++)

```c
// const pointer to un-const variable
int * const p1 = &i;
```

- p1++; // compile error
- (*p1)++; // ok

```c
// un-const pointer to const variable
const int * p2 = &b;
```

- p2++; // ok
- (*p2)++; // compile error

```c
// const pointer to a const variable
const int * const p3 = &b;
```

Const methods

```cpp
class A
{ int a;
public:
    void print() const;
    void set();
};
void A::print() const {
 // print(const A* const this)
 a=5; // = this→a = 5 = error
 cout << a; // OK
}
void A::set() {
 // set(A* const this)
 this->a=5;  // OK
}
```

```cpp
int main() {
 A a;
 const A ca;
 a.print(); //=print(&a)
 a.set(); //=set(&a)
 ca.print();//=print(&ca)
 ca.set();//=set(&ca) –
compilation error!
}
```

Const methods

```cpp
class A
{ BigInt xx;
public:
 const BigInt& foo() const;
 BigInt& foo();
};


const BigInt& A::foo() const {
    cout << "const foo\n";

    return xx;
}
BigInt& A::foo()
{
    cout << "foo\n";

    return xx;
}
```

```cpp
int main()
{

    A a;

    const A ca;

    A a2 = ca; //OK

    a.foo () = 5;

    BigInt i=ca.foo();
}
```

```
// output
foo
const foo
```

How can we have two "foo" functions?

– Overload resolution:

```
A::foo(A* const this)


A::foo(const A* const this)
```

Why do we need two "foo" functions?

See folder 7.

C++: friend functions

- Version 1: Dr. Ofir Pele
- Version 2: Dr. Erel Segal-Halevi

friend functions

- Friend function in a class:

  - Not a method of the class.

  - Has access to the class's private and protected data members.

  - Defined inside the class scope.

- See folder 8 for an example.