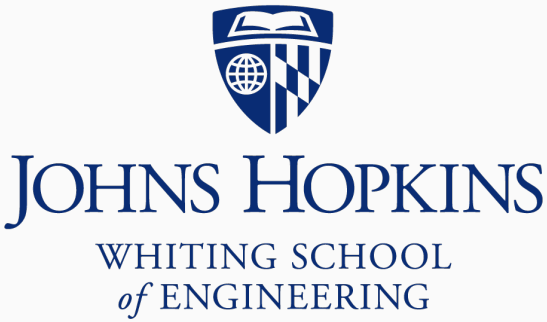
# static class members

Ben Langmead

ben.langmead@gmail.com

www.langmead-lab.org



Source markdown available at github.com/BenLangmead/c-cpp-notes

# `static` class members

A `class` member marked `static` behaves like a "global" variable or function

- Typical members are associated with an *instance* of the `class`; a field takes up space in every `class` variable
- `static` members are associated with the `class`, but not with instances
- `this` pointer is not available in `static` member functions

# static class members

```
class Temperature {
public:
    Temperature(double f) : fahrenheit(f) { }

    static double fahrenheit_to_celcius(double f) {
        return 5.0 / 9.0 * (f - 32);
    }

    static double celcius_to_kelvin(double c) { return c + 273.15; }

    static float fahrenheit_to_kelvin(double f) {
        return celcius_to_kelvin(fahrenheit_to_celcius(f));
    }

    double as_fahrenheit() const { return fahrenheit; }
    double as_celcius()    const { return fahrenheit_to_celcius(fahrenheit); }
    double as_kelvin()     const { return fahrenheit_to_kelvin(fahrenheit); }

    static const int FREEZING = 32;

    bool is_freezing() const { return fahrenheit <= FREEZING; }

private:
    double fahrenheit;
};
```

# `static` class members

Static

- `fahrenheit_to_celcius`
- `celcius_to_kelvin`
- `fahrenheit_to_kelvin`
- `FREEZING`
  - Stored once & doesn't take space in `Temperature` variables

Non-static

- Constructor
- `as_fahrenheit`, `as_celcius`, `as_kelvin`
- `is_freezing`
- `fahrenheit`
  - Only this takes space in `Temperature` variables

# static class members

```
#include <iostream>
#include <cassert>
#include "temp.h"

int main() {
    Temperature t1(30.0);
    assert(t1.is_freezing());    // non-static
    assert(t1.as_celcius() < 0); // non-static
    assert(Temperature::fahrenheit_to_celcius(33) > 0); // static
    std::cout << "Assertions passed" << std::endl;
    return 0;
}

$ g++ -c temp.cpp -std=c++11 -pedantic -Wall -Wextra
$ g++ -o temp temp.o
$ ./temp
Assertions passed
```

## `static` class members

From outside the `class` member functions, use `A::B` to access `static` member `B` of `class` `A`

- `Temperature::fahrenheit_to_celcius(33)`

`static const` fields with integer type can be initialized immediately:

- `static const int FREEZING = 32`
- Typical convention is to capitalize `static const` fields

Otherwise, they are declared in the `class` definition but defined later in a `.cpp` file

# static class members

```
// *** circle.h ***

class Circle {
public:
    static const float PI;

    Circle(double r) : radius(r) { }

    double area() const { return PI * radius * radius; }

private:
    double radius;
};


// *** circle.cpp ***

#include "circle.h"

const float Circle::PI = 3.14f;
```

# static class members

```
// *** circle_main.cpp ***

#include <iostream>
#include "circle.h"

int main() {
    std::cout << "PI is " << Circle::PI << std::endl;
    return 0;
}
```

```
$ g++ -c circle_main.cpp -std=c++11 -pedantic -Wall -Wextra
$ g++ -c circle.cpp -std=c++11 -pedantic -Wall -Wextra
$ g++ -o circle_main circle_main.o circle.o
$ ./circle_main
PI is 3.14
```