static const Data Members

When you have a constant that only applies to the situation represented by your class you can add it as a static const data member.

```
class Bizarro
{
public:
    static const int kAnswer{-42};
    static constexpr double kE{3.14159};
    static const double kPi;
};
```

In the Bizarro world, almost **everything is backwards**. The "answer to everything" is `-42`, not `42` as in our world. The mathematical constant ***e*** is `3.14159`, and the constant `PI` is the square root of `PI` in our own world.

For **integral types**, you may initialize `static const` data members inside the class (as with `kAnswer`); no other initialization is required. Starting in C++11 you can also do this for other types, using the keyword `constexpr`, instead of `const` (as with `kE`), provided that the value can be calculated at compile time.

However, if a type needs to calculate a value at runtime, (as `kPi` does), you'll still need to provide a separate **definition** in the `.cpp` file.

```
const double Bizarro::kPi = sqrt(acos(-1.0));
```

The data members are `static` (there is **only one copy** stored), they are `const`, (they **cannot be changed**), and they are `public` (you can use them outside of the class.)

```
cout << Bizarro::kAnswer << endl;
cout << Bizarro::kPi << endl;
cout << Bizarro::kE << endl;
```