Topics to Learn Later

Kate Gregory

www.gregcons.com/kateblog @gregcons

C++ Has a LOT of Syntax

Lots of ways to do things

Some are faster

Some are more convenient

Some are holdovers from C++98 or C

You don't need to know all of it to write a program

C++ has a LOT of syntax

- Most of it will make more sense when you've written some programs
- You'll have a problem to solve that the syntax deals with
- You'll know how to try using the syntax
- But – you might come across something in other material

Casting

```
double d = 4.9;

int i = d;


i = static_cast<int>(d);
```

dynamic_cast<>

const_cast<>

reinterpret_cast<>

- Why would you put a non integer into an integer? Is this a mistake?

  compiler will warn

- Casting tells the compiler “I meant to do that”
- Tells other developers “look what I’m doing here”
- Other casts make intent obvious

The const Keyword

```
int const amount = 90;



string Transaction::Report() const
{
// …
}
```

- **Promises the compiler that a variable's value won't change**
  - Prevents logic errors
  - Enables optimizations

- **Promises that a member function won't change the value of any member variables**
  - Add to function declaration and definition

Passing Parameters to Functions

```
void display(Transaction t);

// . . .

Transaction deposit(50, “Deposit”);

display(deposit);
```

- **By default, what goes to the function is a copy**
  - Changes inside display() will be to the local variable, not to deposit

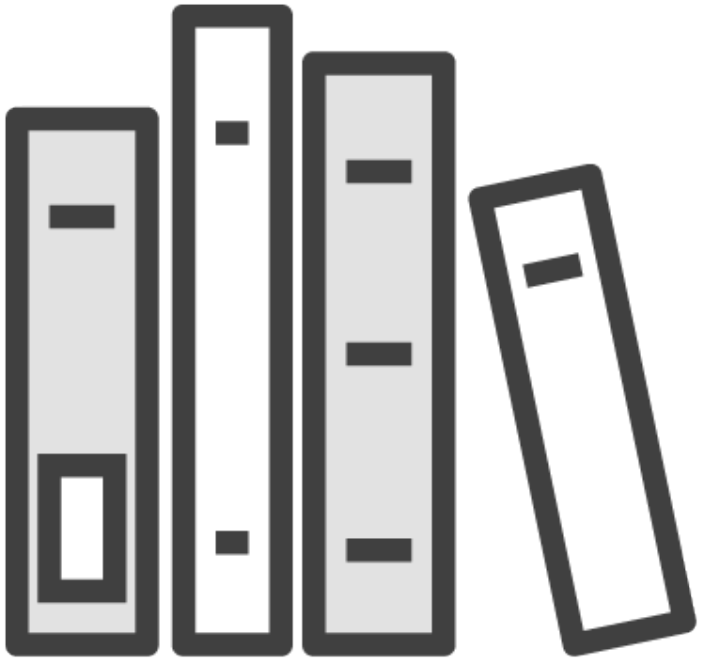
```
void update(Transaction& t);

update(deposit);
```

- **You can arrange for the function to take the parameter by reference**
  - Changes to deposit will “stick”
  - Call it exactly the same way

```
void display(Transaction const& t);
```

- **Even if you don’t want to change the parameter, you might pass by reference**

The Standard Library



So much more than just <iostream>, <string>, and <vector>

Collections

Algorithms (find, sort, ...)

Complex numbers, random numbers, regular expressions

. . .

Standards committee is hard at work adding more

Looking for a library? Check the Standard Library first

Classes That Manage Resources

- A File class
  - Keep a file handle in a member variable
  - Member functions to Open, read, and write a file
- A class to work with a database
  - Keep an open database connection in a member variable
  - Member functions to do queries
- . . .

Classes That Manage Resources

How can you ensure the resource is properly managed?

- Don't leave the file hanging open
- Be sure to close the connection

Could write a function

- Close, Dispose, Cleanup, ...
- People forget to call

C++ has a destructor

- Guarantees that cleanup gets a chance to happen
- Name is ~ and name of class – Eg ~Account()

Scope

```
{
    int i;
    Account a;
    Transaction T(50,"Deposit");
}
```

- Constructor runs when object comes into scope

- Destructor runs when object goes out of scope
- Most common case – flow of control reaches closing brace

- Member variables go out of scope when the instance they belong to does

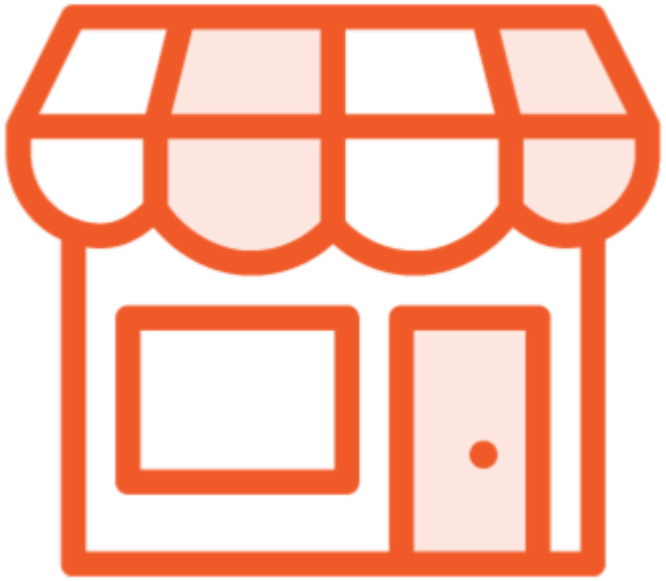
Things to Learn Elsewhere

Exceptions

Alternative to returning error codes

Can make neater and faster code when done right

The Free Store



- Also called the heap, it lets you create really large or long lived objects
  - Access through pointers
  - `std::shared_ptr` and `std::unique_ptr`
- Memory management
  - resource management in general
- Learn from modern material only!
- RAII, Rule of 3, Rule of 5, Rule of 0

Lambdas

A way to use a few lines of code as a parameter to a function, or something to store in a variable

Debugging

Whatever compiler you use, there is a debugger for you

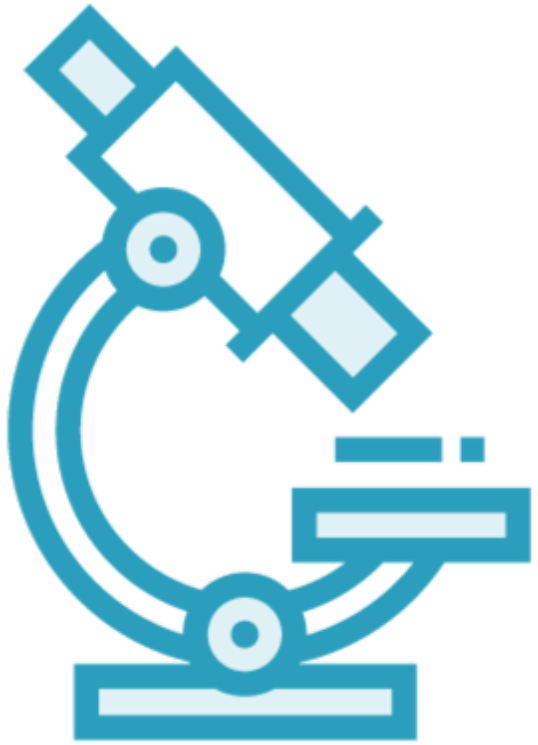
Debugging is a vital skill for all developers

Not just to find bugs

- Understand flow of control
- Watch values change
- See when compiler calls things for you
  - Eg constructor

Learning to use your debugger is the first step towards being a better developer

Minor Details



- Inheritance, virtual functions, polymorphism, multiple inheritance
- the enum keyword
- Boolean operators && and ||, shortcutting
- Bitwise operators & | ^ ! << >>
- The switch statement
- More punctuation you haven't seen yet
  - % & * -> ?
- Default parameters to functions

Advanced Details

- Interacting with the OS – eg calling a Windows API
- Writing templates
- Writing your own operator overloads

Summary

You know enough C++ to write a real program

You'll need to learn a lot more to write some kinds of applications

- Windows application (desktop)
- Phone apps (each platform is different)
- Unix application
- Web service
- Service

Learn frameworks and libraries as a next step

C++ has a lot of syntax

- Learn it when you need it
- If something feels really hard, remember there is more C++ you can learn that might include an easier way to do it