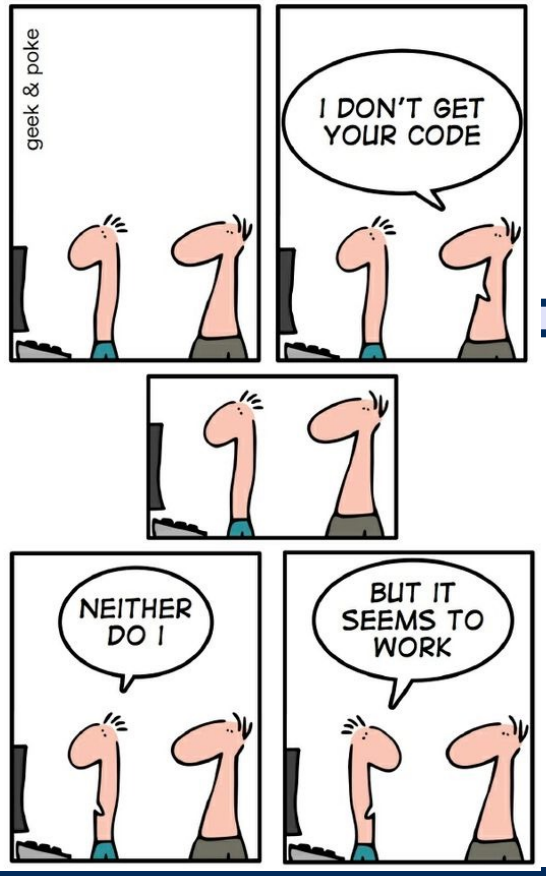
CSCI 200: Foundational Programming Concepts & Lecture 06

Debugging

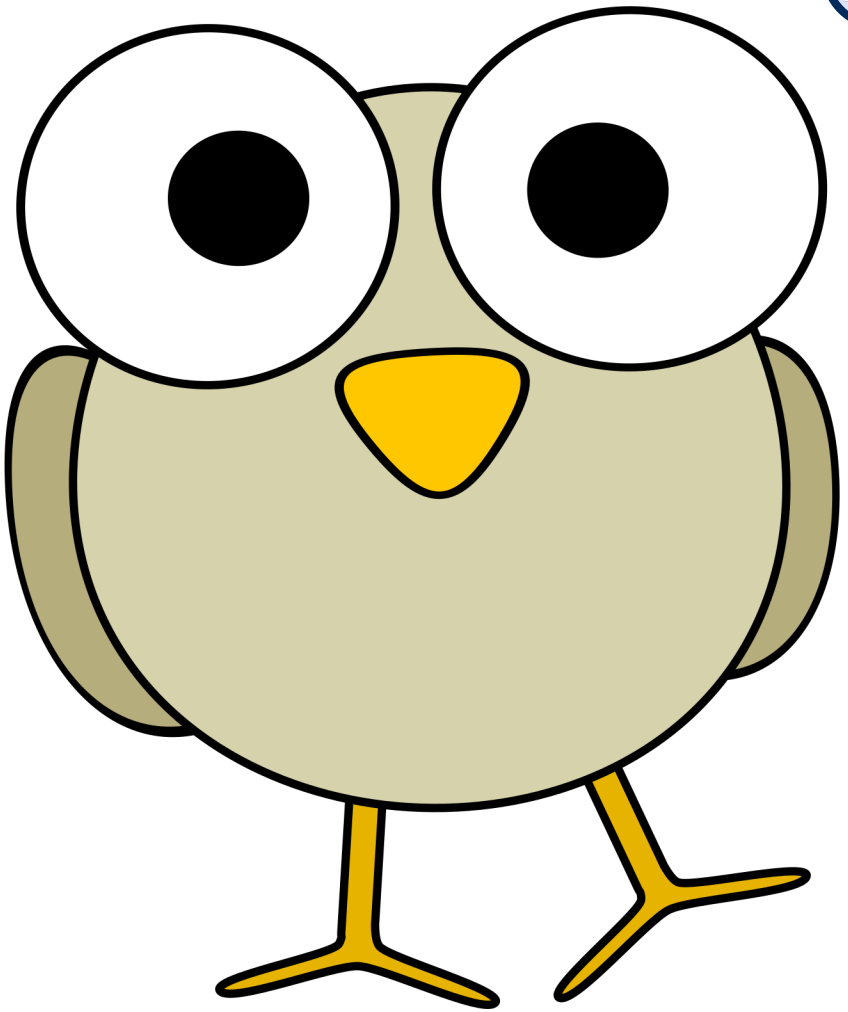


Previously on CSCI200

- Loops!
  - `while` / `do-while`
  - `for`

  - `break` / `continue`

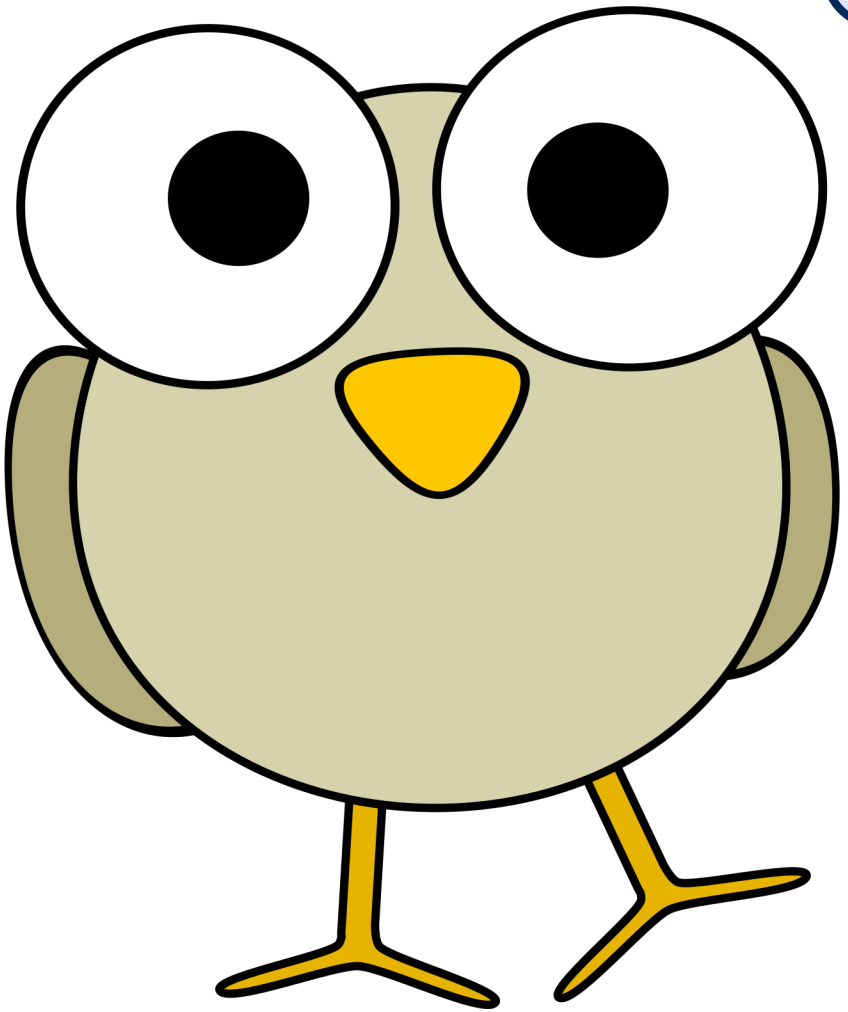
Questions?



Previously on CSCI200

- Loops!
  - `while` / `do-while`
  - `for`
  - `break` / `continue`

Questions?

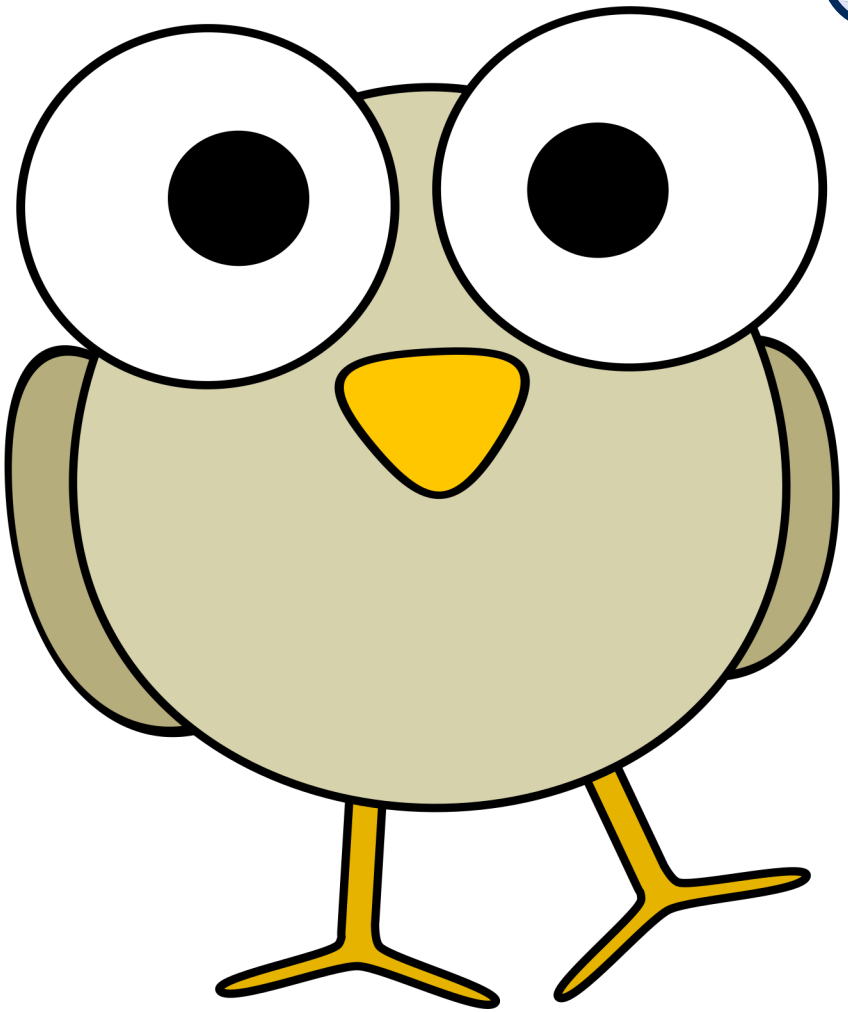


Previously on CSCI200

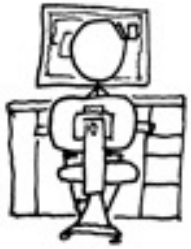
- Loops!
  - `while` / `do-while`
  - `for`

  - `break` / `continue`

Questions?



Don't use goto



Learning Outcomes For Today

- Describe how a computer generates a program from code.
- Discuss the design process and strategies for developing good code.
- Implement various techniques to trace & debug a program.
- Discuss the pros/cons of the various debugging techniques.
- Identify and correct errors in program structure and logic.

On Tap For Today

- Pseudocode
- Compiling Your Program
- Debugging
  - Compiler Warnings
  - Print Lines
  - Debugger
- Practice

On Tap For Today

- Pseudocode
- Compiling Your Program
- Debugging
  - Compiler Warnings
  - Print Lines
  - Debugger
- Practice

Pseudocode

- Intermediary step before typing any C++ code
- English description of algorithm and program
  - Develop solution and program logic before touching the keyboard

- First step in the design process

Simple Pseudocode Example

- Task: Compute the area of a circle
  - Define constant PI
  - Create double variables area, radius
  - Prompt user for radius value
  - Calculate area (PI*r^2)
  - Print area for circle with given radius

Implementing Pseudocode

- Write out pseudocode as comments

```
// Define constant PI

// Create double variables area, radius

// Prompt user for radius value

// Calculate area (PI*r^2)

// Print area for circle with given radius
```

Implementing Code

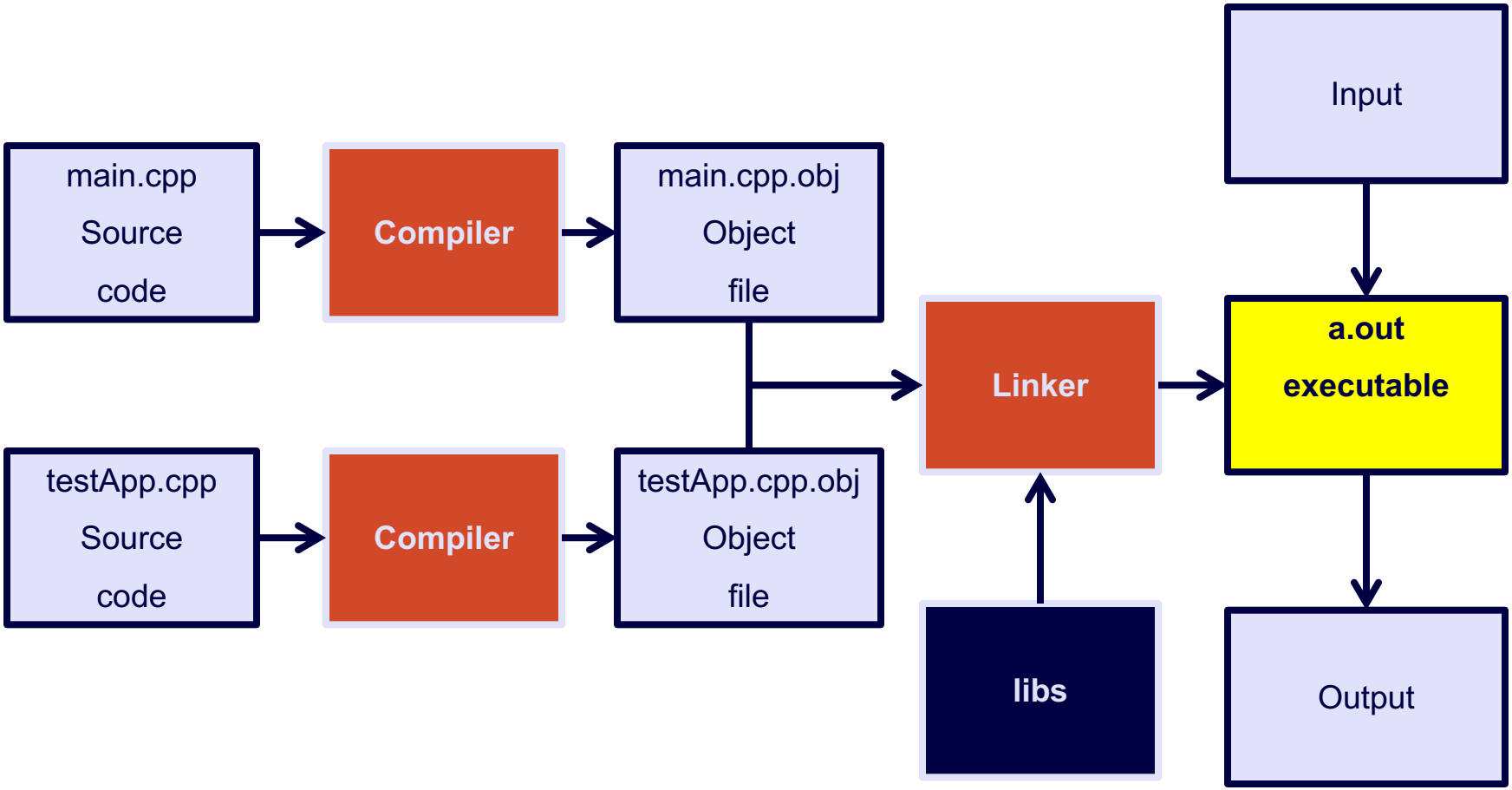
- Fill in code comment by comment
  - Can test that each step is done correctly
  - Code is commented upon completion!

```
// Define constant PI
const double PI = 3.141529;
// Create double variables area, radius
double area, radius;
// Prompt user for radius value


// Calculate area (PI*r^2)


// Print area for circle with given radius
```

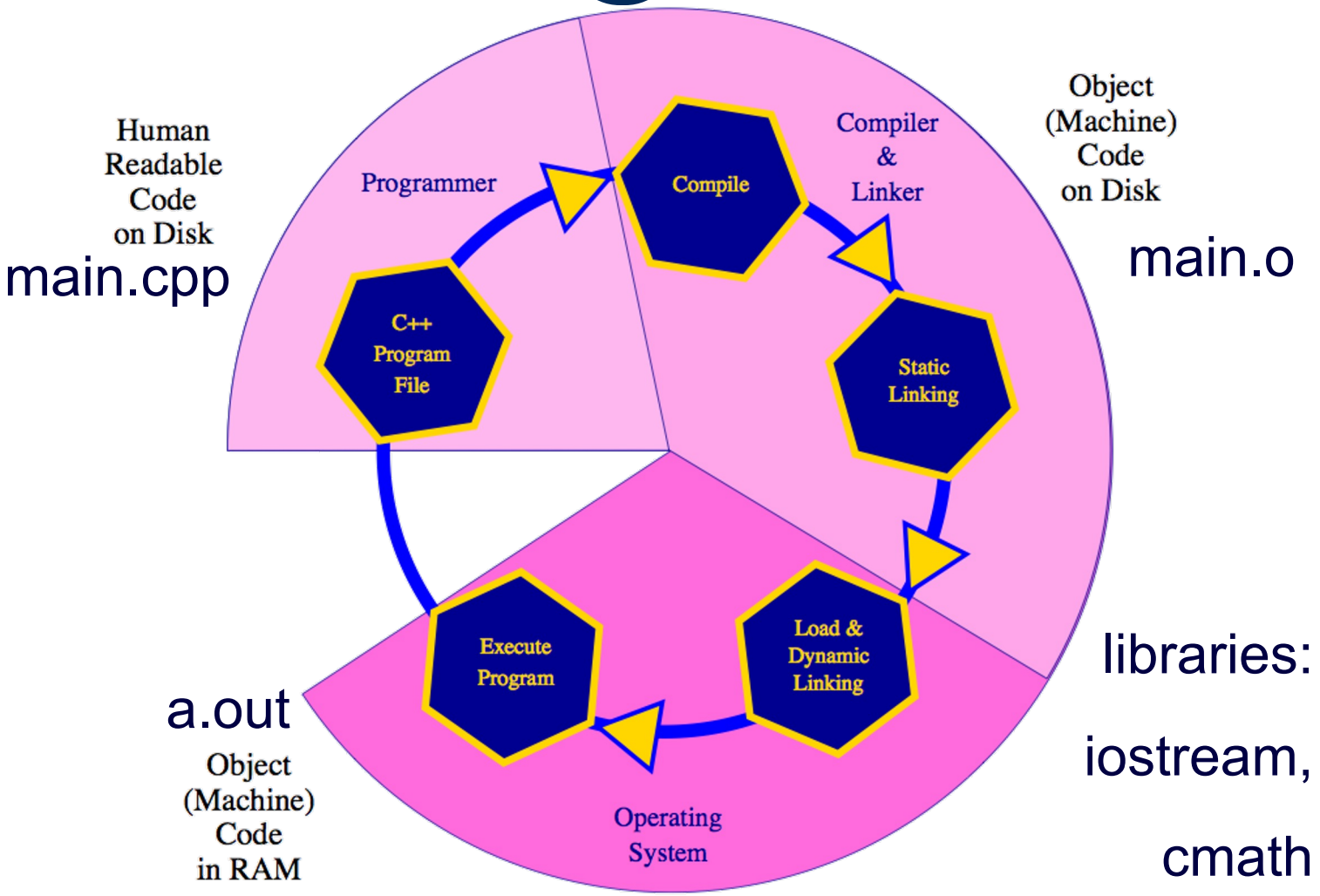
On Tap For Today

- Pseudocode
- Compiling Your Program
- Debugging
  - Compiler Warnings
  - Print Lines
  - Debugger
- Practice

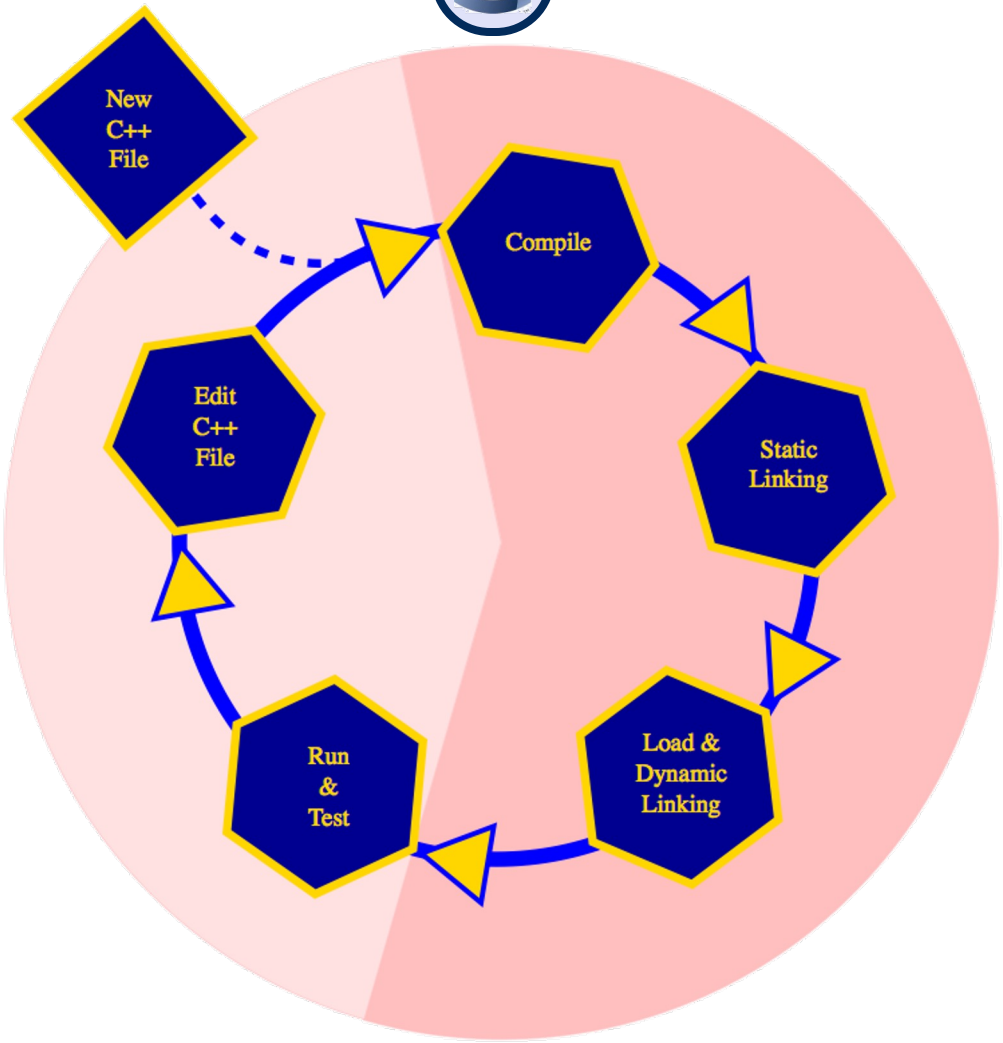
Compile & Link Process



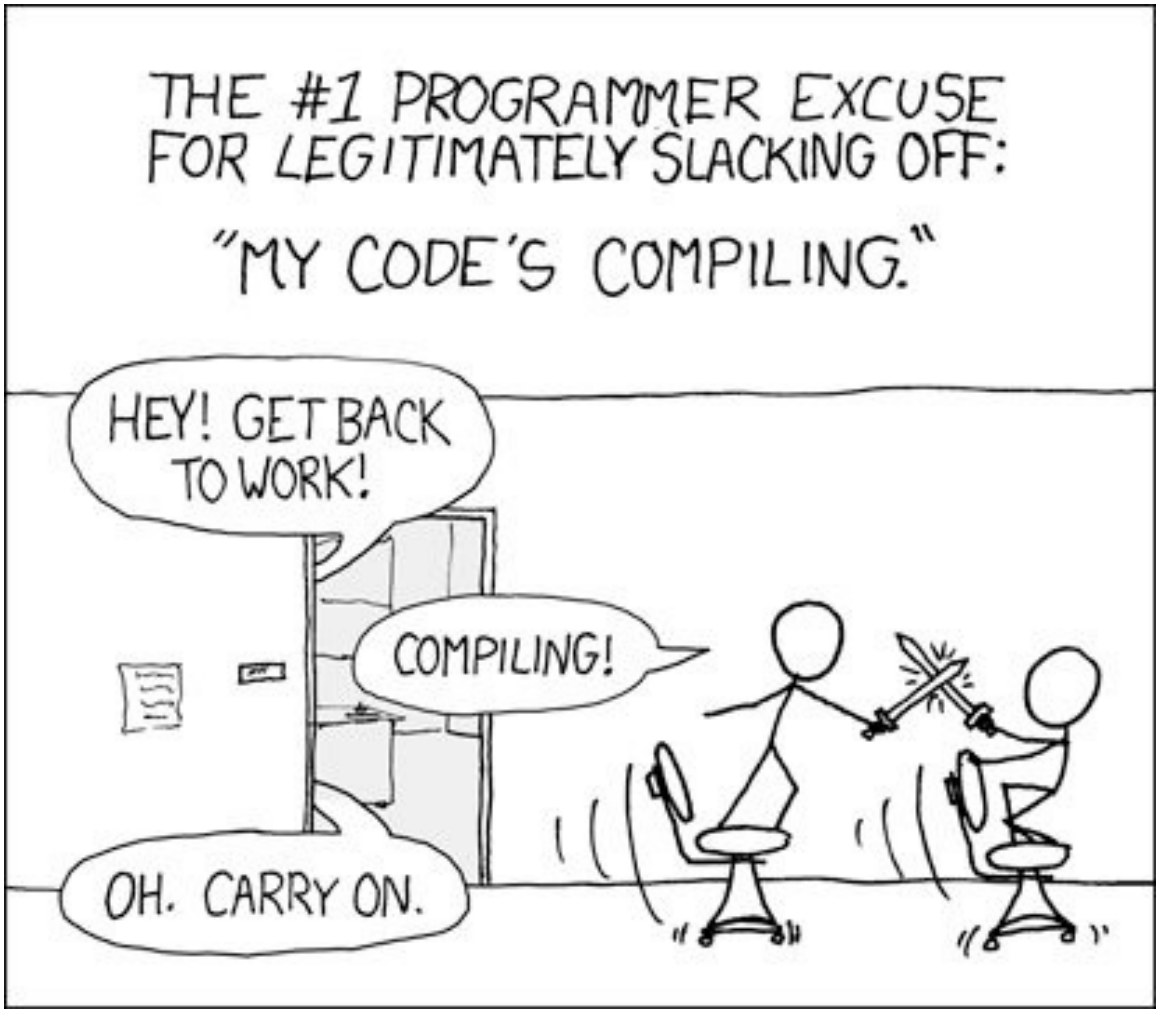
C++ Build Process



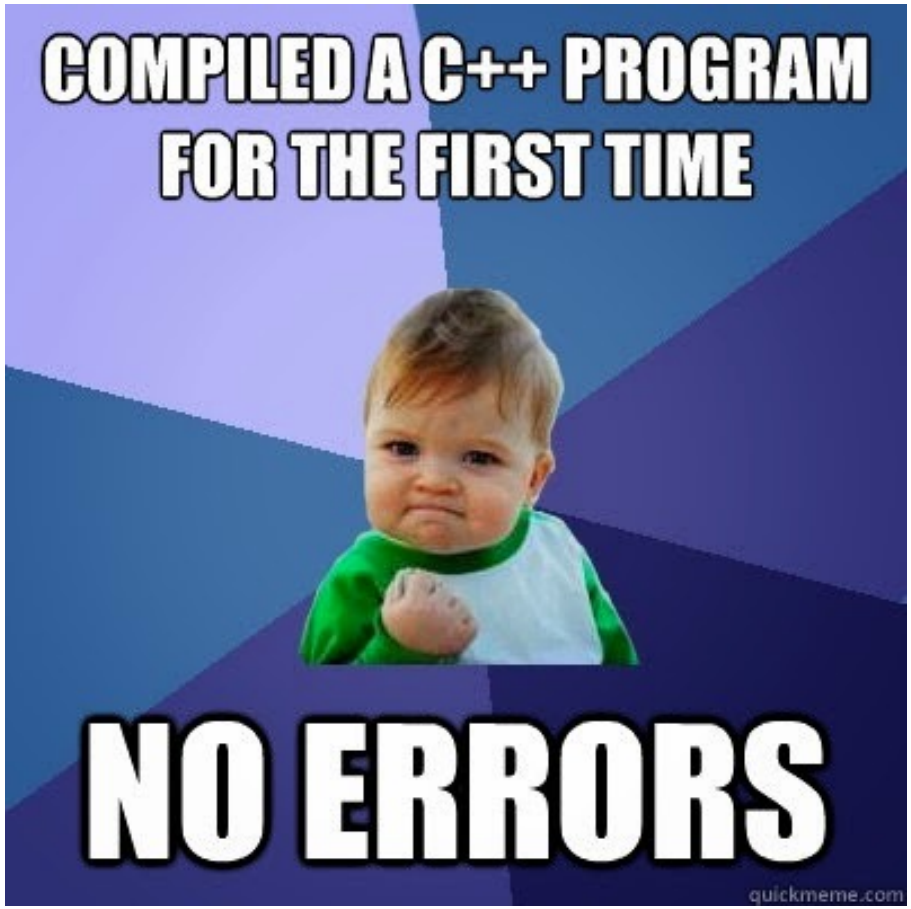
C++ Development Cycle



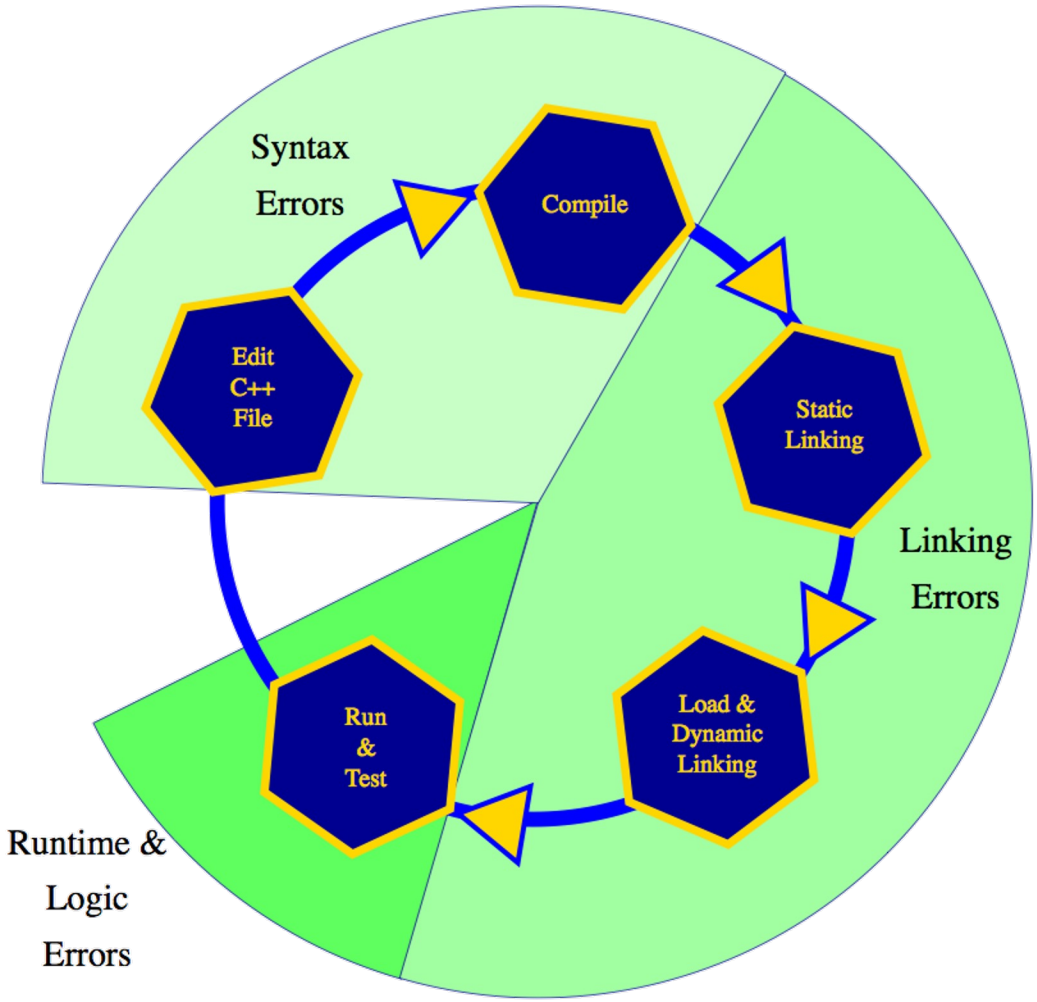
More xkcd


THE #1 PROGRAMMER EXCUSE FOR LEGITIMATELY SLACKING OFF:
"MY CODE'S COMPILING."
HEY! GET BACK TO WORK!
COMPILING!
OH. CARRY ON.

The Goal

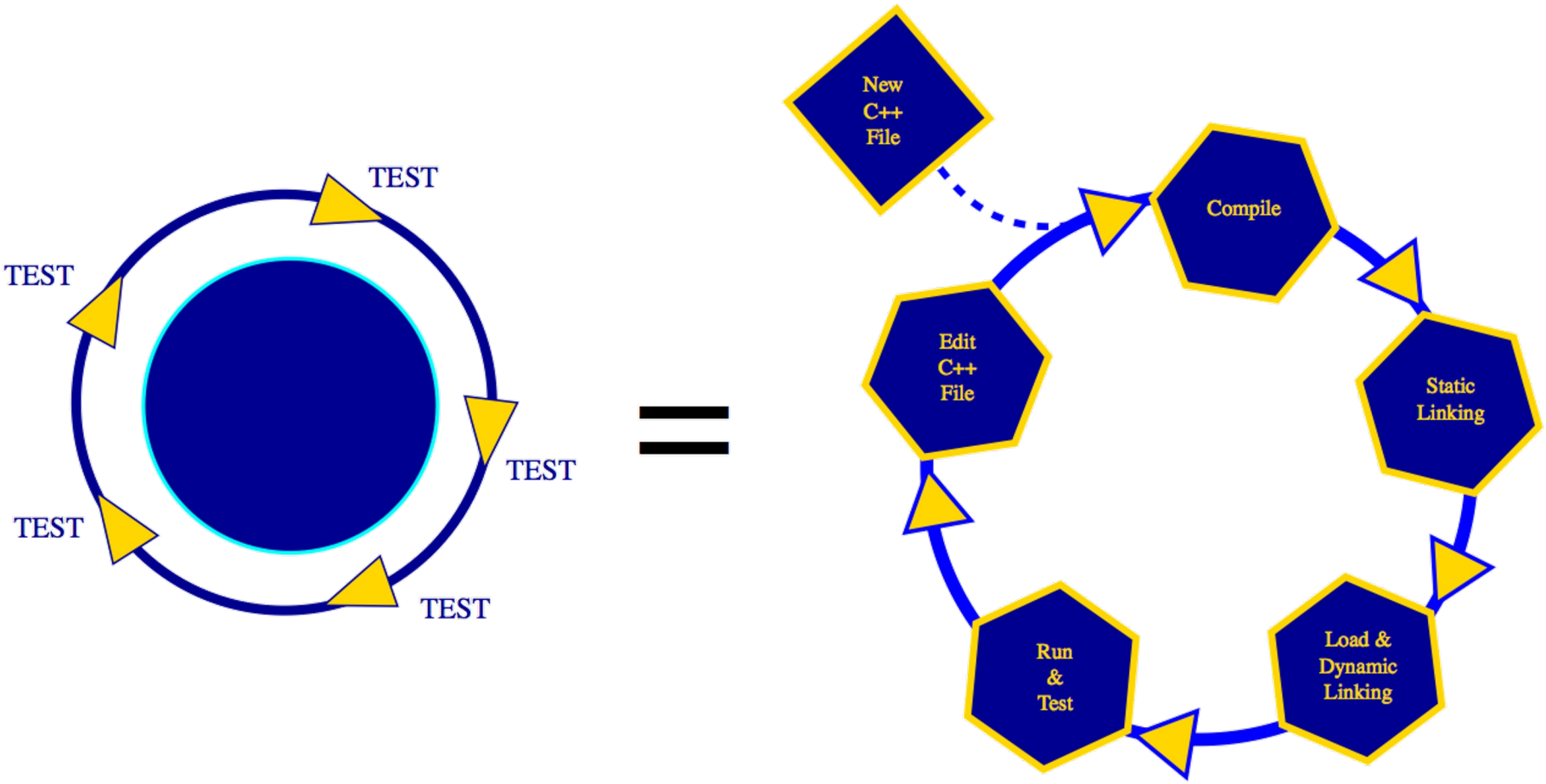

COMPILED A C++ PROGRAM
FOR THE FIRST TIME
NO ERRORS
quickmeme.com

The Reality


Syntax
Errors
Compile
Edit
C++
File
Static
Linking
Linking
Errors
Load &
Dynamic
Linking
Run
&
Test
Runtime &
Logic
Errors

Compile, Link, Execute

- Again and again and again



On Tap For Today

- Pseudocode
- Compiling Your Program
- Debugging
  - Compiler Warnings
  - Print Lines
  - Debugger
- Practice

Debugging aka Fixing Errors

- Syntax Errors
  - Program will not compile
  - Compiler gives you message and line #
- Linker Errors
  - Program will not link
  - Linker gives you message
- Runtime & Logic Errors
  - Program runs & quits unexpectedly (runtime)
  - Program runs & produces wrong output (logic)

Fixing Syntax Errors

- Code

```
5  cout << "Hello World" << enl;
6  cout << "Hello World" << endl
7  cout << "Hello World" << endl;
```

- Compiler Message

```
main.cpp:5:28: error: use of undeclared identifier 'enl'; did you mean 'endl'?
  cout << "Hello World" << enl;
                           ^~~
                           endl
ostream:1004:1: note: 'endl' declared here
endl(basic_ostream<_CharT, _Traits>& __os)
^
main.cpp:6:32: error: expected ';' after expression
  cout << "Hello World" << endl
                               ^
                               ;
```

Fixing Linking Errors

- Stay tuned!
  - Won't see these until next week ☺
- Will discuss when they start appearing

Fixing Runtime/Logic Errors

- Three options
  - Display compiler warnings
  - Add print lines that display helpful debug statements
  - Use a debugger to trace the execution of the program

On Tap For Today

- Pseudocode
- Compiling Your Program
- Debugging
  - Compiler Warnings
  - Print Lines
  - Debugger
- Practice

Example

- Code

```
12  int x;
13  cout << "x is: " << x << endl;
14  x++;
15  cout << "x is: " << x << endl;
```

- Displays

```
x is: 6422352
x is: 6422353
```

Compiler Warnings

- NOT compiler errors
  - Nothing wrong syntactically with code
  - Flow, values, etc. not being used correctly
  - Hint that something could go wrong at runtime

- Enable with compiler flags `-Wall -Wextra`
  - Turns on ALL warnings

```
g++ -Wall -Wextra -o main.o -c main.cpp
```

Compiler Flags

- Add to Makefile

```
CXX = g++

CXXFLAGS = -Wall -Wextra


%.o: %.cpp

    $(CXX) $(CXXFLAGS) -o $@ -c $<
```

- When compiling

```
% make
g++ -Wall -Wextra -o main.o -c main.cpp
main.cpp:13:23: warning: variable 'x' is uninitialized when used here [-Wuninitialized]
  cout << "x is: " << x << endl;
                      ^
main.cpp:12:8: note: initialize the variable 'x' to silence this warning
  int x;
       ^
        = 0
1 warning generated.
```

- ALWAYS have warnings turned on!

Take It Further!

- Add to Makefile

```
CXX = g++

CXXFLAGS = -Wall -Wextra -Werror


%.o: %.cpp

    $(CXX) $(CXXFLAGS) -o $@ -c $<
```

- When compiling

```
% make
g++ -Wall -Wextra -Werror -o main.o -c main.cpp
main.cpp:13:23: error: variable 'x' is uninitialized when used here [-Wuninitialized]
  cout << "x is: " << x << endl;
                      ^
main.cpp:12:8: note: initialize the variable 'x' to silence this warning
  int x;
       ^
        = 0
1 warning generated.
```

- Treat warnings as errors!

On Tap For Today

- Pseudocode
- Compiling Your Program
- Debugging
  - Compiler Warnings
  - Print Lines
  - Debugger
- Practice

Add `cout` Statements

- Useful statements for the developer
  1. Print the value of a variable
  2. Print a message of where currently am within the program
  3. Print the result of an expression
     1. What a condition evaluates to
     2. Split apart a complex expression into its components
  4. Others

Print Lines v1 – print

```
// ...
cout << "Before loop" << endl;
for(int i = 0; i < 10; i++) {
  cout << "i = " << i << endl;
  // ...
}
cout << "After loop" << endl;
// ...
```

- Pros?
- Cons?

Print Lines v2 – print maybe

```
const bool DEBUG = true;
// ...
if(DEBUG) cout << "Before loop" << endl;
for(int i = 0; i < 10; i++) {
  if(DEBUG) cout << "i = " << i << endl;
  // ...
}
if(DEBUG) cout << "After loop" << endl;
// ...
```

- Pros?
- Cons?

Print Lines v3 – compiler directives

```
#define DEBUG
// ...
#ifdef DEBUG
cout << "Before loop" << endl;
#endif
for(int i = 0; i < 10; i++) {
#ifdef DEBUG
  cout << "i = " << i << endl;
#endif
  // ...
}
#ifdef DEBUG
cout << "After loop" << endl;
#endif
// ...
```

Print Lines v3 – compiler directives

- If DEBUG is defined

  code looks like

```
#define DEBUG
// ...

cout << "Before loop" << endl;

for(int i = 0; i < 10; i++) {

  cout << "i = " << i << endl;

  // ...
}

cout << "After loop" << endl;

// ...
```

Print Lines v3 – compiler directives

- If DEBUG is not defined

```
// ...
#ifdef DEBUG
cout << "Before loop" << endl;
#endif
for(int i = 0; i < 10; i++) {
#ifdef DEBUG
  cout << "i = " << i << endl;
#endif
  // ...
}
#ifdef DEBUG
cout << "After loop" << endl;
#endif
// ...
```

Print Lines v3 – compiler directives

- If DEBUG is not defined

  code looks like

```
// ...

for(int i = 0; i < 10; i++) {

  // ...
}

// ...
```

Print Lines v3 – compiler directives

```
#define DEBUG
// ...
#ifdef DEBUG
cout << "Before loop" << endl;
#endif
for(int i = 0; i < 10; i++) {
#ifdef DEBUG
  cout << "i = " << i << endl;
#endif
  // ...
}
#ifdef DEBUG
cout << "After loop" << endl;
#endif
// ...
```

Print Lines v3 – compiler directives

- Don't define DEBUG in code
- Use compiler flag!

```
// ...
#ifdef DEBUG
cout << "Before loop" << endl;
#endif
for(int i = 0; i < 10; i++) {
#ifdef DEBUG
  cout << "i = " << i << endl;
#endif
  // ...
}
#ifdef DEBUG
cout << "After loop" << endl;
#endif
// ...
```

Another Compiler Flag

`g++ -D`

- `-D` defines a compiler variable

```
g++ -DDEBUG -o main.o -c main.cpp
```

Let Makefile Handle It

```
TARGET = HelloWorld
SRC_FILES = main.cpp

CXX = g++
CXXFLAGS = -Wall -Wextra -Werror
OBJECTS = $(SRC_FILES:.cpp=.o)
ifeq ($(shell echo "Windows"), "Windows")
            TARGET := $(TARGET).exe
            DEL = del
else
            DEL = rm
endif

$(TARGET): $(OBJECTS)
            $(CXX) -o $@ $^

debug: CXXFLAGS += -DDEBUG
debug: clean $(TARGET)

%.o: %.cpp
            $(CXX) $(CXXFLAGS) -o $@ -c $<

clean:
            $(DEL) $(TARGET) $(OBJECTS)
```

- Build “release” mode

```
make
```

- Build “debug” mode

```
make debug
```

Print Lines v3 – compiler directives

- Pros?
- Cons?

```
#define DEBUG
// ...
#ifdef DEBUG
cout << "Before loop" << endl;
#endif
for(int i = 0; i < 10; i++) {
#ifdef DEBUG
  cout << "i = " << i << endl;
#endif
  // ...
}
#ifdef DEBUG
cout << "After loop" << endl;
#endif
// ...
```

On Tap For Today

- Pseudocode
- Compiling Your Program
- Debugging
  - Compiler Warnings
  - Print Lines
  - Debugger
- Practice

Debugger

- Another program / tool at your disposal
  - Windows: `gdb`
  - OS X: `lldb`
- Run through terminal
  - Need to compile with `-g` flag to generate debug information

```
g++ -Wall -Wextra -Werror -g -o main.o -c main.cpp
```

- ALWAYS have debug flag turned on!

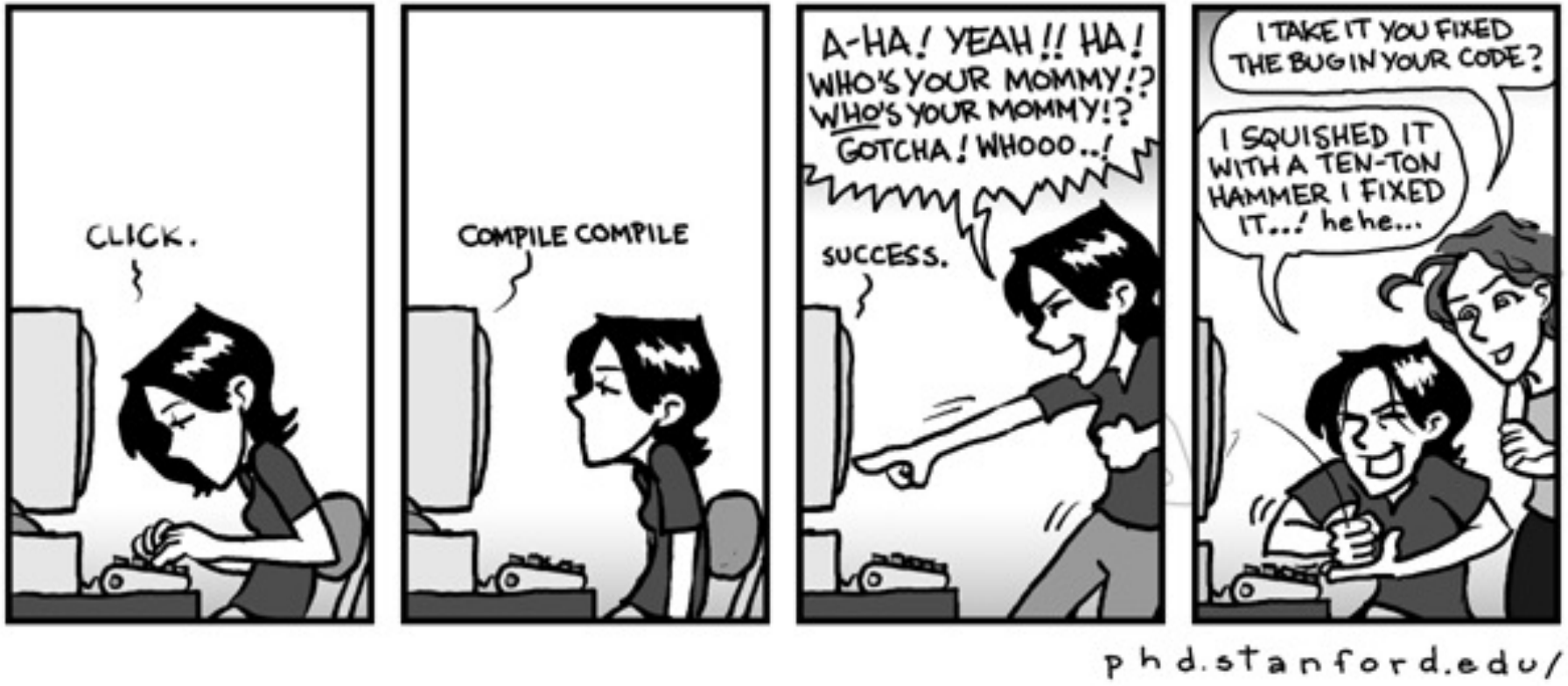
Let Makefile Handle It

```
TARGET = HelloWorld
SRC_FILES = main.cpp

CXX = g++
CXXFLAGS = -Wall -Wextra -Werror -g
OBJECTS = $(SRC_FILES:.cpp=.o)

...

%.o: %.cpp
	$(CXX) $(CXXFLAGS) -o $@ -c $<

...
```

Running Debugger

- gdb .\HelloWorld.exe
  - b <#>
  - run
  - print <*var*>
  - info b
  - step
  - continue
  - kill
  - q
- lldb ./HelloWorld
  - b <#>
  - run
  - print <*var*>
  - br l
  - step
  - continue
  - kill
  - q

The Joy of Debugging



On Tap For Today

- Pseudocode
- Compiling Your Program
- Debugging
  - Compiler Warnings
  - Print Lines
  - Debugger
- Practice

HW Submission

To Do for Next Time

- Set1 due tomorrow
- Set2 beginnings next time
  - Starting with zyBooks Chapter 5
- Complete 9/6 Post-Class Survey
- Quiz during next class
  - Practice quiz available in Canvas