

# Lecture 1b: Compiling C++; Style

---

CS 70: Data Structures and Program Development

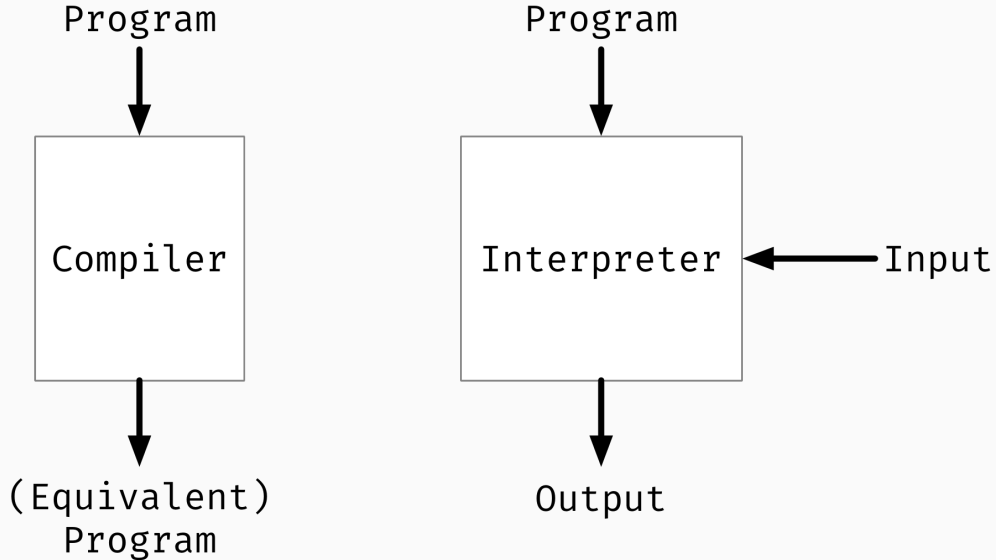
Thursday, January 24

# Today's Learning Targets

- I can explain the steps to compile multi-file C++ code.
- I can contrast the design goals of Java and C++.
- I can identify code with bad style.
- I can write readable and elegant C++ code.

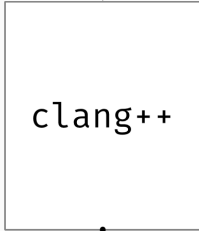
# Compiling C++

# Recall: Compiling vs. Interpreting



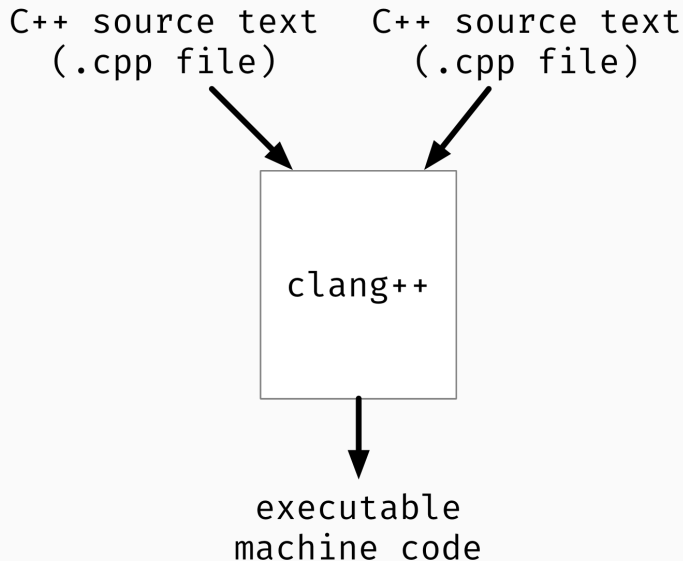
# Compiling C++ (1 file)

C++ source text  
(.cpp file)



executable  
machine code

# Compiling C++ (multiple files)



# Compiling C++ (multiple files, better!)

C++ source text  
(.cpp file)

C++ source text  
(.cpp file)

# Compiling C++ (multiple files, better!)

C++ source text  
(.cpp file)



clang++ -c



some  
machine code  
(.o file)

C++ source text  
(.cpp file)



# Compiling C++ (multiple files, better!)

C++ source text  
(.cpp file)



clang++ -c



some  
machine code  
(.o file)

C++ source text  
(.cpp file)



clang++ -c



some  
machine code  
(.o file)

# Compiling C++ (multiple files, better!)

C++ source text  
(.cpp file)



clang++ -c



some  
machine code  
(.o file)



C++ source text  
(.cpp file)



clang++ -c

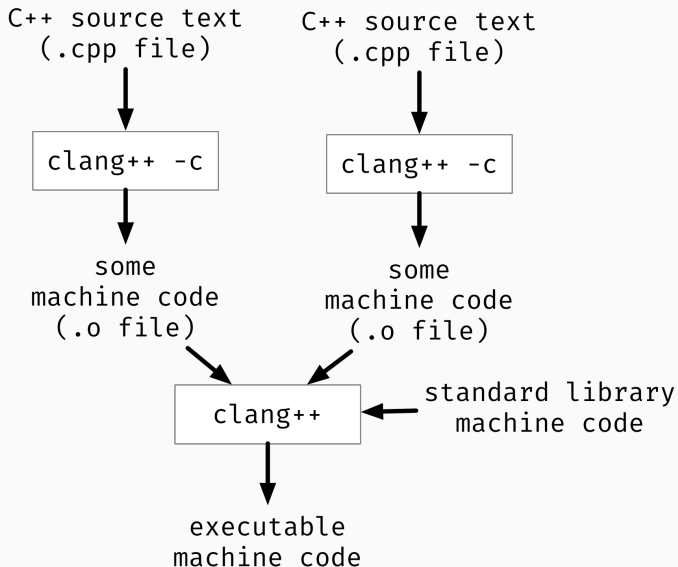


some  
machine code  
(.o file)



← standard library  
machine code

# Compiling C++ (multiple files, better!)



# Question

- a. Suppose our program has three `.cpp` source files. To get a runnable program, how many times should `clang++` run?
- b. Suppose now we change one definition in one `.cpp` file. To get an updated runnable program, how many times should `clang++` run?

# System Header Files

```
#include <iostream>
#include <string>

int main() {
    std::string message = "Hello, World!";
    std::cout << message << "\n";
    return 0;
}
```

# User Header Files

```
----- exclaim.hpp -----  
#include <string>  
  
// Adds an !  
std::string exclaim(std::string sentence);  
-----main.cpp-----  
#include <iostream>  
  
#include "exclaim.hpp"  
  
int main() {  
    std::cout << exclaim("wow") << std::endl;  
}
```

# User Header Files (continued)

```
----- exclaim.hpp -----
```

```
#include <string>
```

```
// Adds an !
```

```
std::string exclaim(std::string sentence);
```

```
----- exclaim.cpp -----
```

```
#include "exclaim.hpp"
```

```
#include <string>
```

```
std::string exclaim(std::string sentence) {
```

```
    return sentence + "!";
```

```
}
```

# I changed exclaim.cpp. Steps to recompile?

```
----- exclaim.cpp -----  
#include "exclaim.hpp"  
#include <string>  
  
// Don't add ! if the string ends in !  
std::string exclaim(std::string sentence) {  
    size_t length = sentence.size();  
    if (length > 0 && sentence[length-1] == '!') {  
        return sentence;  
    } else {  
        return sentence + "!";  
    }  
}
```



# **Programming Language Design Principles**

# Design Principles for C++

C++ is a statically typed, object-oriented, imperative language.

# Design Principles for C++

C++ is a statically typed, object-oriented, imperative language.

Design goals for the language include:

- “C++ is a better C”
- Efficiency (in time and space) **No Overhead**
- Trust (i.e., obey) the programmer

# Design Principles for Java

Java is a statically typed, object-oriented, imperative language.

# Design Principles for Java

Java is a statically typed, object-oriented, imperative language.

Design goals for the language include:

- Safety (ensure bad things can't happen)
- Portability (Write Once, Run Everywhere)
- Familiarity

# C++ or Java?

- (a) A `long` integer is a 64-bit number.
- (b) A `long` integer has some number of bits (perhaps the same number as in a CPU register).

# C++ or Java?

- (a) If you try to access index 100 of a 10-element array of integers, an error (exception) will be reported.
- (b) If you try to access index 100 of a 10-element array of integers, anything could happen (but you'll most likely get back some bits taken from memory past the end of the array, interpreted as an integer.)

**Style**



# Readings

What were the big ideas?

# Style and Elegance

- Readability should be your top priority
- And all other things being equal,
  - Maximize maintainability / extensibility
  - Don't be *gratuitously* inefficient.
- **Nobody** writes beautiful code all the time
  - Go back and fix things!

# Consistency



# Use Consistency and Inconsistency to Your Advantage

- Similar things should *look* similar
- Different things should *look* different

(In)consistent with what?

# Use Consistency and Inconsistency to Your Advantage

- Similar things should *look* similar
- Different things should *look* different

(In)consistent with what?

- With ourselves (within a file)
- With culture (project conventions)

We will ask you to generally follow Google's C++ Style Guide (as implemented in the `cpplint` tool)

# Consistency: Variable Names

Variable names and functions: camelCase

- `count, i, activeTask, launchMissiles()`

Data members (fields): camelCase + trailing underscore

- `front_, currentCapacity_`

Class names: Capitalized CamelCase

- `Gene, StudentTranscript`

Constants: All caps, underscore between words

- `VERSION, MAX_STUDENTS`

# Consistency: Applying Idioms

```
const size_t NUM_LETTERS = 26;
std::string alphabet;
for (size_t i = 0; i < NUM_LETTERS; ++i) {
    alphabet += ('a' + i);
}

for (size_t i = 0; i < alphabet.size(); ++i) {
    std::cout << alphabet[i] << " " << i << "\n";
}
```

# Consider this code example

- What, specifically, is wrong with it?
- Why, specifically, is this a problem?
- How would you suggest fixing the problem(s)?

```
// MUST be set to 1!
```

```
Params.ParentalLevel = 3; // QA now insists this be 2
```



# Consider this code example

- What, specifically, is wrong with it?
- Why, specifically, is this a problem?
- How would you suggest fixing the problem(s)?

```
// if ((typec!="20") && (typec!="13") && (typec!="5") && (typec!="4"))
if (typec != "20") {
    if (typec != "13") {
        if (typec != "5") {
            if (typec != "4") {
                selectType("ALLOC");
                return;
            }
        }
    }
}
```

# Exercise

Find the code fragment for your group

- What, specifically, is wrong with it?
- Why, specifically, is this a problem?
- How would you suggest fixing the problem(s)?
- Is there *anything* good to say about the style?

The last student to join will be the group's spokesperson.

```
std::string capitalizedName(std::string name)
{
    if (name == "aafke") {
        return "Aafke";
    } else if (name == "aaron") {
        return "Aaron";
    }
    // lots and lots of similar lines, not shown
    } else if (name == "zuzana") {
        return "Zuzana";
    } else if (name == "zuzanna") {
        return "Zuzanna";
    } else if (name == "zuzanny") {
        return "Zuzanny";
    } else {
        // Name not in the database yet,
        // but lowercase is better than nothing
        return name;
    }
}
```

```
bool validateSSN(std::string ssn)
{
    if ( ssn[0] != '0' && ssn[0] != '1' && ssn[0] != '2' && ssn[0] != '3' &&
        ssn[0] != '4' && ssn[0] != '5' && ssn[0] != '6' && ssn[0] != '7' &&
        ssn[0] != '8' && ssn[0] != '9' ) {
        return false;
    }
    if ( ssn[1] != '0' && ssn[1] != '1' && ssn[1] != '2' && ssn[1] != '3' &&
        ssn[1] != '4' && ssn[1] != '5' && ssn[1] != '6' && ssn[1] != '7' &&
        ssn[1] != '8' && ssn[1] != '9' ) {
        return false;
    }
    // plus 7 more similar cases

    return true;
}
```

## 4a

```
// This is the first example of C++ code that you should critique  
const std::string twentySpaces = "                ";
```

## 4b

```
void SPdfsR(Graph G, int s)
{
    link u; int i, t; double wt;
    int **p = G->path; double **d = G->dist;
    for (u = G->adj[s]; u != NULL; u = u->next)
    {
        t = u->v; wt = u->wt;
        if (d[s][t] > wt)
            { d[s][t] = wt; p[s][t] = t; }
        if (d[t][t] == maxWT) SPdfsR(G, t);
        for (i = 0; i < G->V; i++)
            if (d[t][i] < maxWT)
                if (d[s][i] > wt+d[t][i])
                    { d[s][i] = wt+d[t][i]; p[s][i] = t; }
    }
}
```

*// This is Java code to critique, not C++ syntax!*

```
synchronized (surelyReachableObjectsWhichHaveToBeMarkedAsSuch) {  
    waitRecommended =  
        surelyReachableObjectsWhichShouldHaveBeenProcessedButWereLockContentedSize  
        == surelyReachableObjectsWhichShouldHaveBeenProcessedButWereLockContented.size();  
  
    surelyReachableObjectsWhichShouldHaveBeenProcessedButWereLockContentedSize =  
        surelyReachableObjectsWhichShouldHaveBeenProcessedButWereLockContented.size();  
  
    while (!surelyReachableObjectsWhichShouldHaveBeenProcessedButWereLockContented.isEmpty())  
    {  
        surelyReachableObjectsWhichHaveToBeMarkedAsSuch.push(  
            surelyReachableObjectsWhichShouldHaveBeenProcessedButWereLockContented.getFirst() );  
    }  
}
```

```
/**  
 * function is03or09or10.  
 * takes: prodCode  
 * returns: bool  
 */  
bool is03or09or10(std::string prodCode)  
{  
    if ("03" == prodCode) return true;  
    else if ("09" == prodCode) return true;  
    else if ("10" == prodCode) return true;  
    else return false;  
}  
  
// other functions not shown: is01, is02, and is004or005
```



```
void validate(Person p)
{
    const int TWO = 2;
    bool validated = true;
    std::string myMessage, val;

    // Name
    val = p.name;
    if (!(val.find(" ")!=std::string::npos) || !(val.size()>TWO)) {
        myMessage += "Please fill in your full Name\n";
        validated = false;
    }

    // Address
    val = p.address;
    if (val.size()>TWO) {
        myMessage += "Please fill in your full Address\n";
    }
}
```

```
bool b(std::string a) {  
    int b = -1, c = a.length();  
    goto p;  
    while(b<c){  
        if (a[b] != a[c]) return false;  
p:    ++b; c--; }  
        return true;  
}
```

```

public boolean foo(... omitted ...) {
    try {
        synchronized (... omitted ...) {
            if (... omitted ...) {
            } else {
            }
            for (... omitted ...) {
                if (... omitted ...) {
                    if (... omitted ...) {
                        if (... omitted ...) {
                            if (... omitted ...) {
                                if (... omitted ...) {
                                    for (... omitted ...) {
                                    }
                                }
                            }
                        }
                    }
                }
            }
        }
    } else {
        if (... omitted ...) {
            for (... omitted ...) {
                if (... omitted ...) {
                } else {
                }
                if (... omitted ...) {
                } else {
                    if (... omitted ...) {
                    }
                }
            }
        }
        if (... omitted ...) {

```

etc.

# See you in lab tomorrow!

- Tomorrow in Lab:
  - Version Control, Git, and Github
  - Starting HW1
- To-do:
  - Do HW0
  - Find a partner