CSI 32
Introductions

Course Structure

- In-class:
  - Topics from textbook
  - Examples
  - Exercises – practice (whenever possible)
  - Quizzes
- At home:
  - Reading
  - Do “**try this**”
  - Work on drills (before exercises)
  - Work on exercises
  - Work on homework (for grade)
  - See the review questions
  - See Terms

Best practices

- Read the chapter ahead of class meeting
- Read the chapter after the class meeting
- Start working on homework assignment right after class
- Get tutoring help immediately
- Do not put issues aside for a later time

This course is

- a continuation of CSI 31
- an introductory programming course in C++
- for beginners who want to become professionals
- for those who are willing to work hard

This course is not

- a course in C++ programming language
- a “washout” course

  (you can handle it!)

The goals

- to review/learn
  - the fundamental programming concepts
  - key useful techniques
  - basic standard C++ facilities
- after the course you will be able to
  - proceed with an “advanced” C++ programming course
  - read large C++ programs
  - write “small” C++ programs

Not the goals of this course

- to become an expert software developer
- to become an expert in C++
- to become an expert user of advanced libraries

Attendance

- Attend every lecture!

- If you cannot attend the class:
  - Notify me
  - Check out the meeting notes, after class notes, and the homework assignment
  - make sure to read the chapters (sections) we covered and look through the lecture slides
  - Follow up with the homework, make sure to submit it by the due date
  - If you missed a quiz, make-ups will be given from time to time on Thursdays, 12pm – 1:50pm

Quizzes

- weekly, most likely on Wednesdays, at the end of the class
- Make-ups:
  - to be eligible for a make-up quiz, your absence has to be an excused absence (if its medical, provide a letter from the doctor’s office)
  - all make-ups will only be on some Thursdays, 12 pm – 1:50 pm
  - the make-up can be taken only at the next available opportunity

Homework assignments

- Will consist of:
  - Reading
  - Drills (work on them!)
  - Exercises for practice
  - Exercises for grade
- You will submit:
  - Exercises for grade only
- To succeed in this class:
  - Practice as much as possible
  - Read the textbook

Self-development / recommended reads

- I will be posting some recommended readings from time to time
- Solely for self-development
- Will not be used in quizzes / exams

Cooperate on learning

- Except for the work you hand in as *individual contributions*, I strongly encourage you to collaborate and help each other
- If in doubt if a collaboration is legitimate: ask!
  - Don’t claim to have written code that you copied from others
  - Don’t give anyone else your code (to hand in for a grade)
  - When you rely on the work of others, explicitly list all of your sources – i.e. give credit to those who did the work
- Use tutoring help
  - Come prepared with questions
  - There are no stupid questions!!!

Cooperate on learning

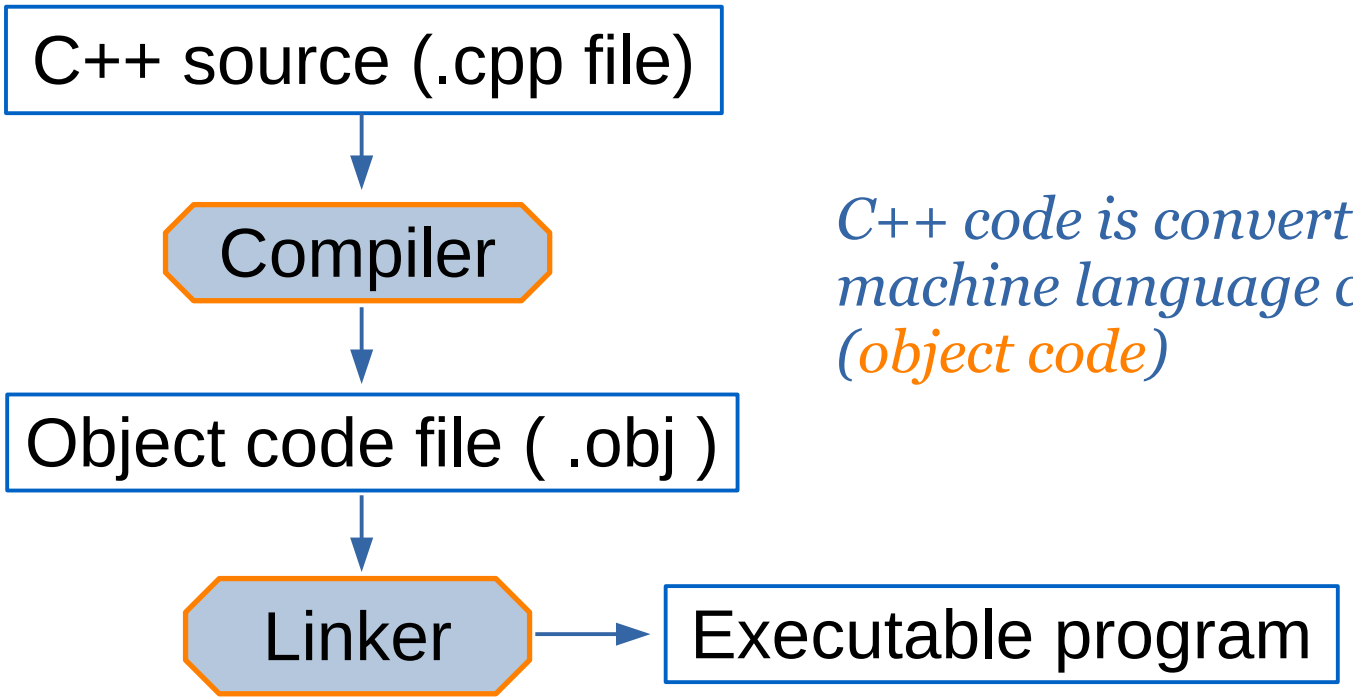
- Don’t study alone when you don’t have to
  - Form study groups
  - Do help each other (without plagiarizing)

Why C++?

- C++ has a number of differences from Python
  - Strictly typed language
  - Access to class data members and member functions is enforced
  - Compilable language
  - Gives us an opportunity to talk about memory management and garbage collection, etc.
- C++ is precisely and comprehensively defined by an ISO standard
  - And that standard is almost universally accepted
  - The most recent standard is ISO C++ 2022

C++ is a compiled language

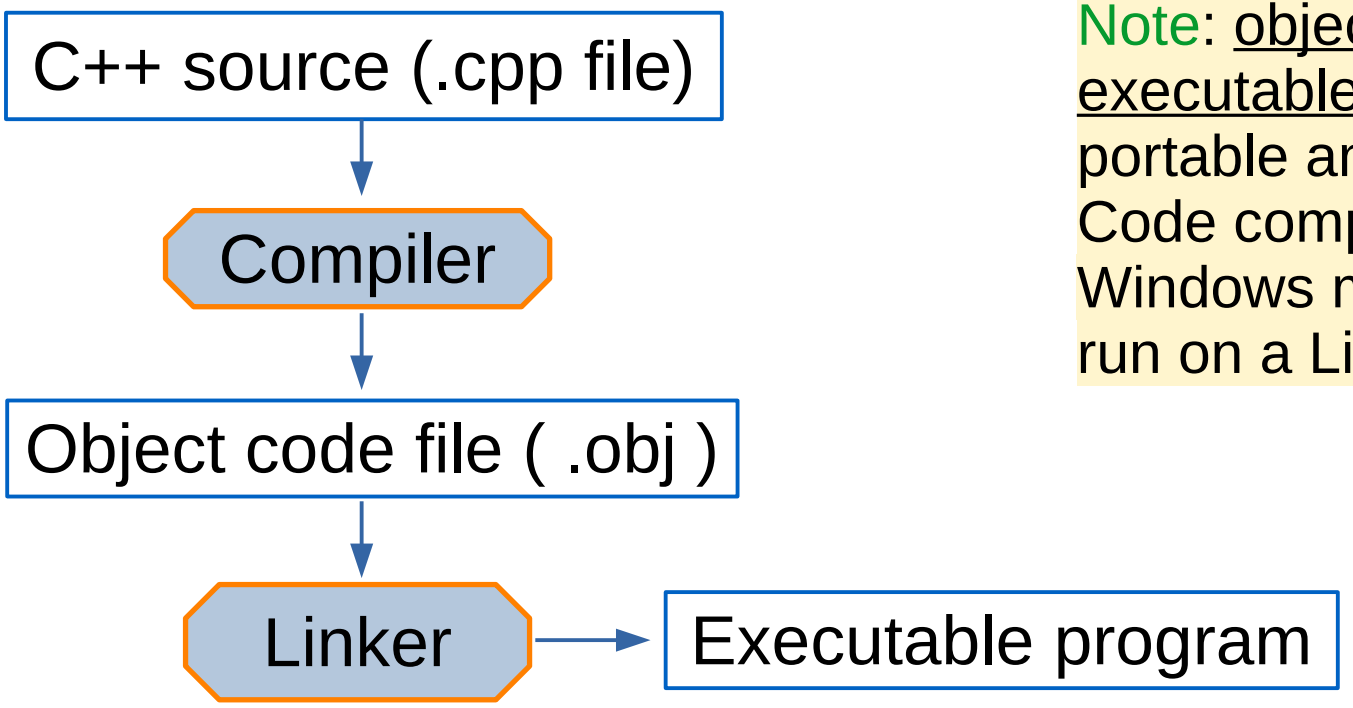
A simplified compilation process:



C++ code is converted into machine language code (object code)

C++ is a compiled language

A simplified compilation process:



Note: object code and executables are not portable among systems. Code compiled for Windows machine will not run on a Linux machine

Errors

Errors found by the compiler are called *compile-time errors*.

Errors found by the linker are called *compile-time errors*.

Errors not found until the program runs are called *run-time errors* or *logic errors*.

A sample program

```
// This program outputs the message
// “Hello World!” to the screen

#include <iostream>

// C++ programs start by executing function main
int main()
{
   std::cout << "Hello, world!\n";

   return 0;
}
```

A sample program

```
// This program outputs the message
// “Hello World!” to the screen

#include <iostream>

// C++ programs start by executing function main
int main()
{
   std::cout << "Hello, world!\n";

   return 0;
}
```

comments (anything written after the token //)

A sample program

```
// This program outputs the message
// "Hello World!" to the screen

#include <iostream>

// C++ programs start by executing function main
int main()
{
   std::cout << "Hello, world!\n";

   return 0;
}
```

preprocessing directive to include the header iostream (used for input/output stream)

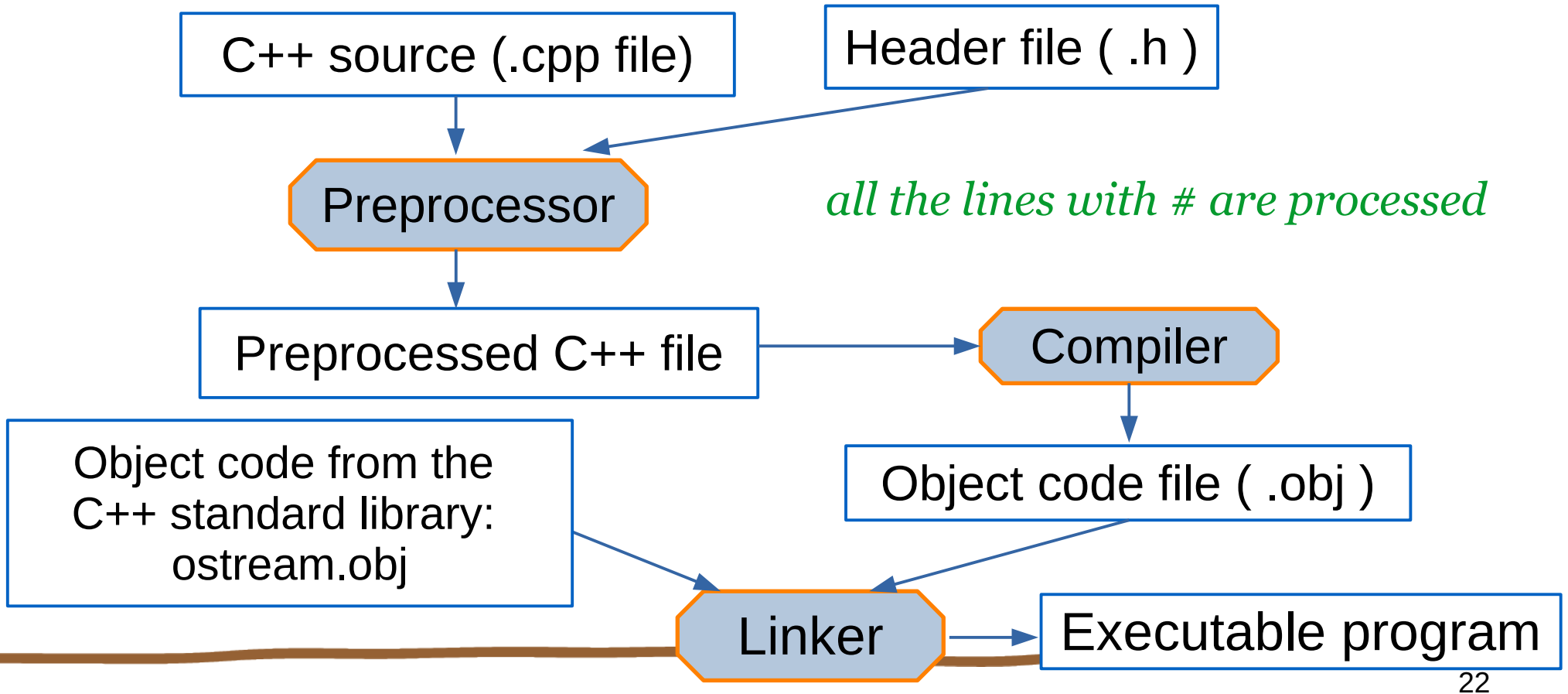
A sample program

```
// This program outputs the message
// "Hello World!" to the screen

#include <iostream>

// C++ programs start by executing function main
int main()
{
   std::cout << "Hello, world!\n";

   return 0;
}
```

preprocessing directive to include the header iostream (used for input/output stream)
Our textbook is using the header std_lib_facilities.h

C++ is a compiled language

A more detailed compilation process for our code:



A sample program

```
// This program outputs the message
// "Hello World!" to the screen

#include <iostream>

// C++ programs start by executing function main
int main()
{
   std::cout << "Hello, world!\n";

   return 0;
}
```

Every C++ program must have a function called main to tell it where to start executing

A function

A function has 4 parts:

- a *return type*
  here `int` (stands for integer, reserved keyword)
- a *name*
  here `main`
- a *parameter list* enclosed in prentheses
  here empty
- a *function body* enclosed in a set of curly braces { }
  lists actions/statements that the function is to perform

A sample program

```
// This program outputs the message
// "Hello World!" to the screen

#include <iostream>

// C++ programs start by executing function main
int main()
{
   std::cout << "Hello, world!\n";

   return 0;
}
```

(abbreviation) standard namespace, or C++ standard library that has definitions of many useful objects.

A sample program

```
// This program outputs the message
// “Hello World!” to the screen

#include <iostream>

// C++ programs start by executing function main
int main()
{
   std::cout << "Hello, world!\n";

   return 0;
}
```

output operator

output object,
standard output stream (character output stream)

A sample program

```
// This program outputs the message
// “Hello World!” to the screen

#include <iostream>

// C++ programs start by executing function main
int main()
{
   std::cout << "Hello, world!\n";

   return 0;
}
```

special character,
indicates a newline

Resources used for these slides

- slides provided by B. Stroustrup at https://www.stroustrup.com/PPP2slides.html
- C++ How To Program, 10th edition, by P. Deitel and H. Deitel
- Class textbook