



CSCE 120/121

Introduction to Program Design & Concepts Introduction

Dr. Tim McGuire

Grateful acknowledgment to Dr. Michael Moore for some of the material on which these slides are based.

Who is this guy, anyway?

- Tim McGuire
 - B.S. in Mathematics, LeTourneau College
 - Minors in Mechanical Engineering and Chemistry
 - M.S. in Mathematics, Colorado State University
 - Ph.D. in Computer Science, **Texas A&M University** (whoop!)
- 44 years experience in academia and industry



Syllabus Review

Why Program?

Why Program?

Computer – programmable machine designed to follow instructions

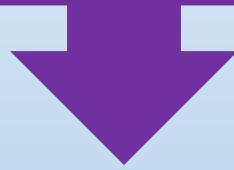
Program – instructions in computer memory to make it do something

Programmer – person who writes instructions (programs) to make computer perform a task

SO, without programmers, no programs; without programs, a computer cannot do anything

Programming

Discovering how, in principle, a problem can be solved by a computer



Translating this solution into a form (a program, or software) that can be executed by a computer

Our civilization runs on software

Most engineering activities involve software

Most programs do not run on things that look like a PC...

Aircraft

- Communication
- Control
- Display
- Signal Processing
- Monitoring



Ships

- Design
- Construction
- Management
- Monitoring
- Hull Design
- Pumps



Energy

- Control
- Monitoring
- Analysis
- Design
- Communications
- Manufacturing
- Visualization



Phones

- Voice Quality
- User Interfaces
- Billing
- Mobility
- Switching
- Reliability
- Provisioning
- Images



C++ allows direct expression of ideas from many application areas

C++ is most widely used language in engineering areas

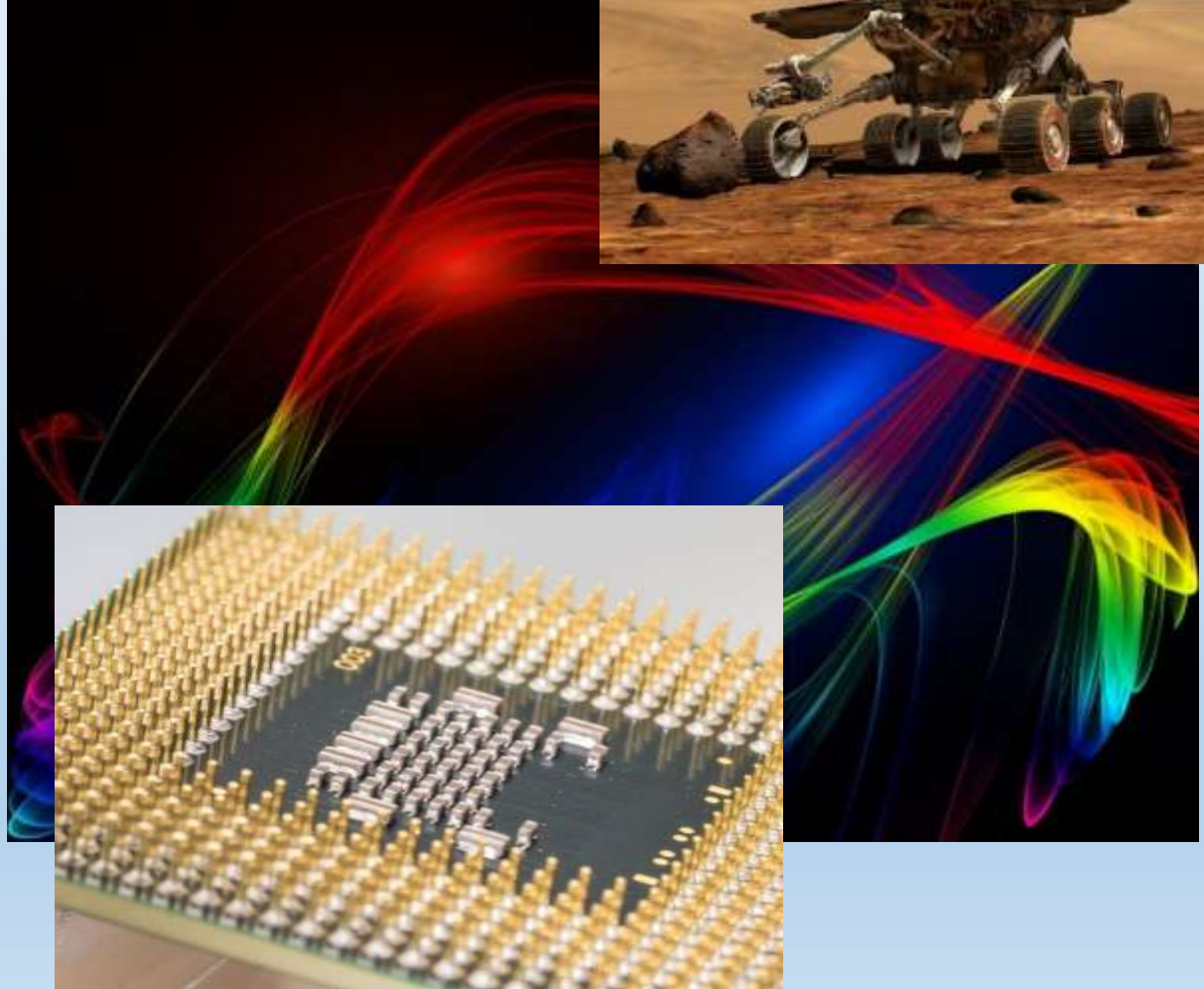
C++ is precisely and completely defined by an ISO standard

C++ is widely available

Concepts you learn with C++ are directly applicable to many other languages

Where is C++ Used

- Mars Rovers,
- Animation,
- Graphics,
- Photoshop,
- GUIs,
- Embedded Systems,
- OS's, Compilers, Slides,
Chip Design,
Chip Manufacturing,
Semiconductor Tools,...



Programming Language Tree

- <https://ccrma.stanford.edu/courses/250a-fall-2005/docs/ComputerLanguagesChart.png>

Who are programmers...

Ada Lovelace

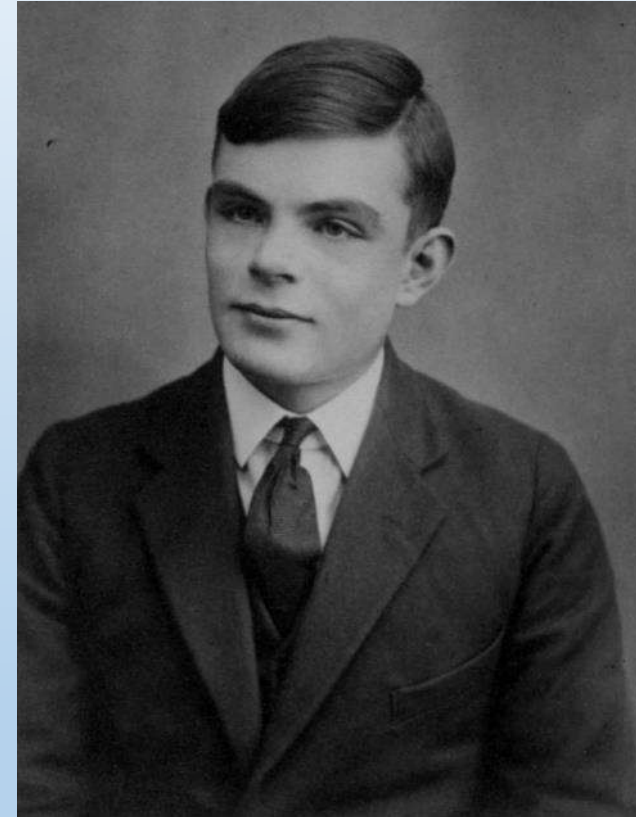
- “First Computer Programmer”
- Born 1815
- Worked with Charles Babbage



["Ada Lovelace portrait" by Alfred Edward Chalon - Science & Society Picture Library. Licensed under Public Domain via Commons](#)

Alan Turing

- British
- Formalized idea of algorithm and computation
- Invented a general model of computing, known as a Turing Machine
- WWII codebreaker
- Turing Award named in his honor
- ["Sackville Park Turing plaque" by User Lmno on en.wikipedia - Photograph taken by Lmno. Licensed under CC BY-SA 3.0 via Commons](#)

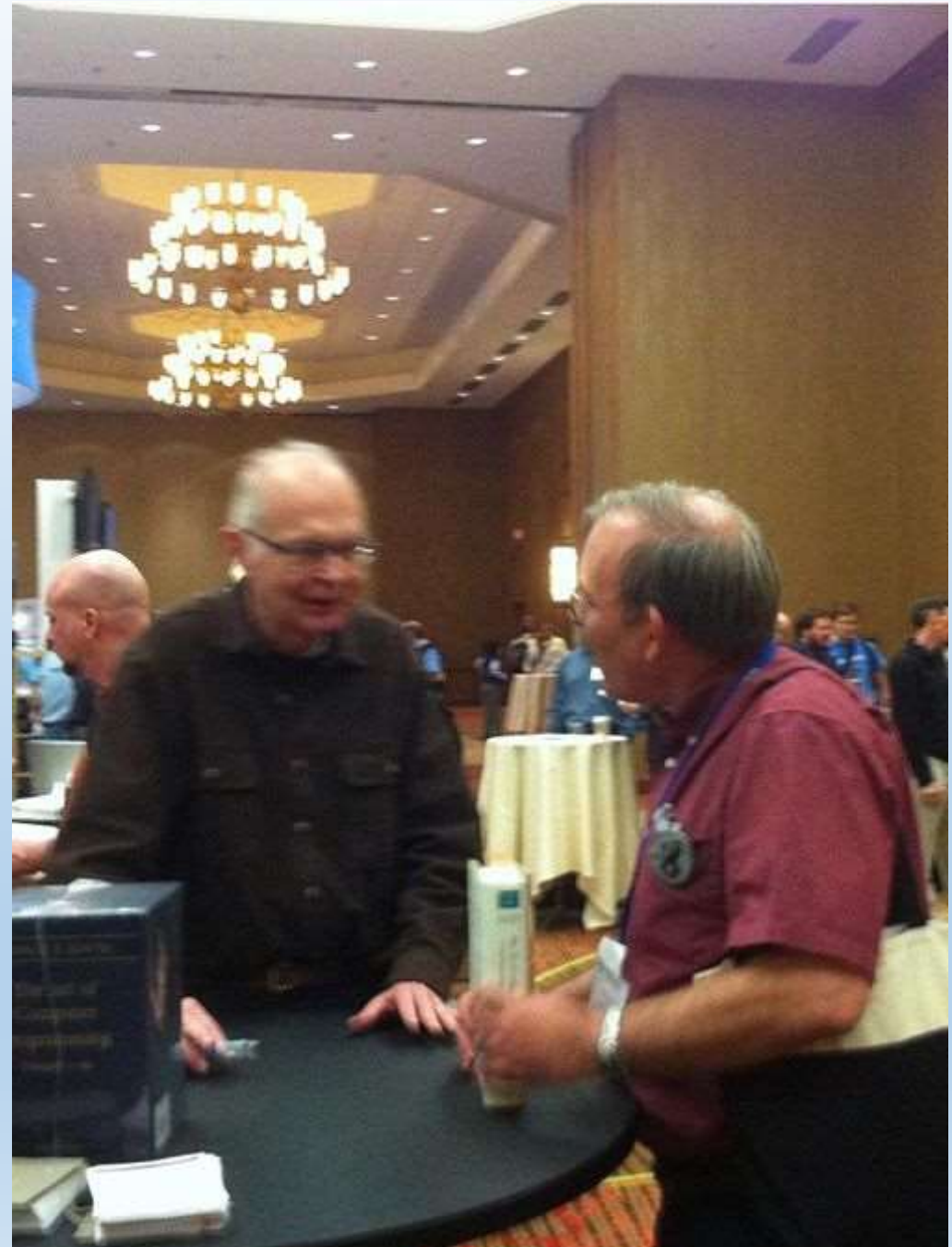


["Alan Turing Aged 16" by Unknown
Licensed under Public Domain via Commons](#)

Donald Knuth

- *The Art of Computer Programming*
- Laid groundwork for computer science as an academic discipline
- Popularized asymptotic notation
- Creator of T_EX

Don Knuth discussing scientific computing environments with Tim McGuire,
March 11, 2011, Dallas, Texas



Acknowledgement

- Based on slides created by Bjarne Stroustrup and Jennifer Welch
- Many images from pixabay.com

Computer Systems: Hardware and Software

Main Hardware Component Categories:

1. Central Processing Unit (CPU)
2. Main Memory
3. Secondary Memory / Storage
4. Input Devices
5. Output Devices

Central Processing Unit (CPU)

Comprised of:

Control Unit

- Retrieves and decodes program instructions

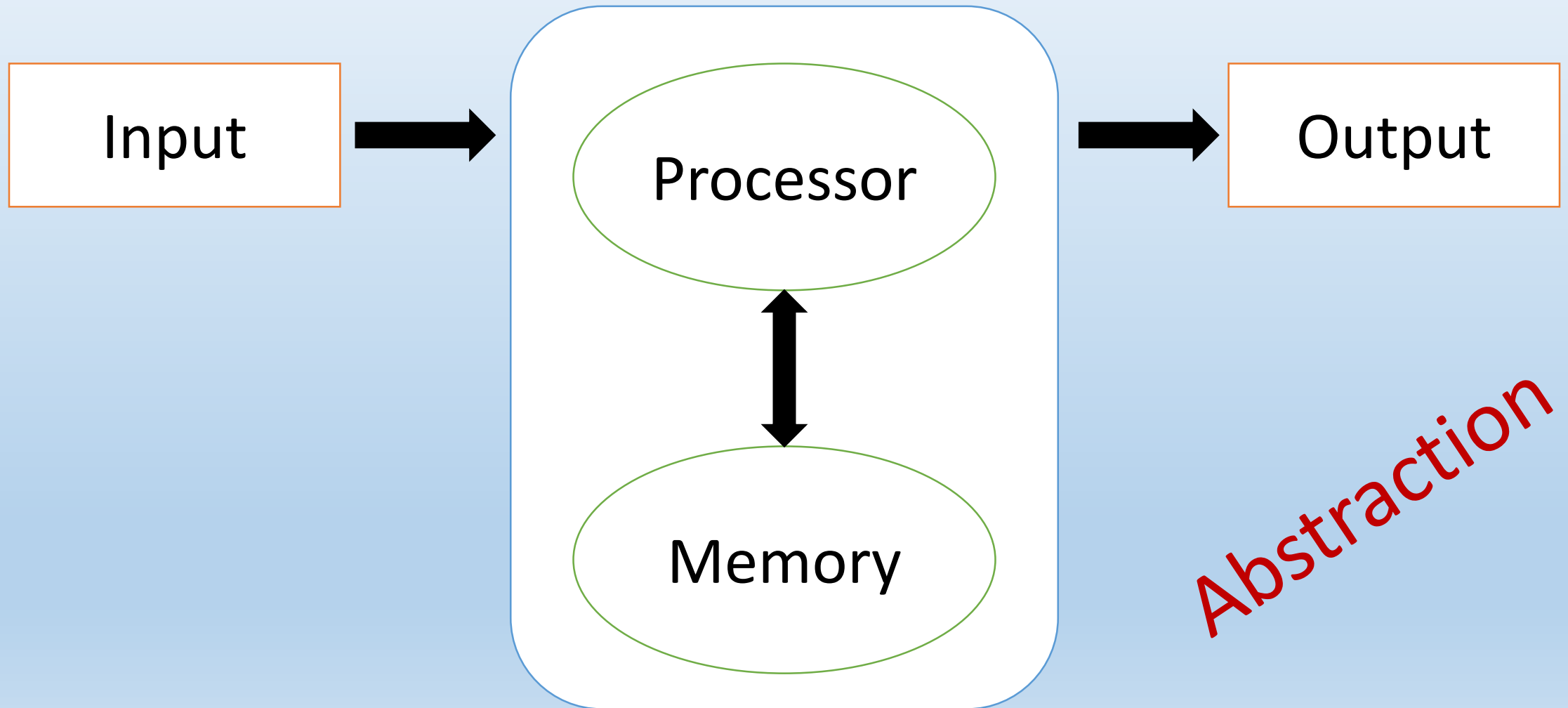
- Coordinates activities of all other parts of computer

Arithmetic & Logic Unit

- Hardware optimized for high-speed numeric calculation

- Hardware designed for true/false, yes/no decisions

Simplified Model



Abstraction

Main Memory

- Usually store in Random Access Memory (RAM)
- Composed of ones and zeros
- Organized as follows:
 - bit: smallest piece of memory.
Has values 0 (off, false) or 1 (on, true)
- byte: 8 consecutive bits.
Addresses – Each byte in memory is identified by a unique number known as an *address* (starting at zero).

Memory Layout



Stack and heap
grow toward each other.

Secondary Storage

- Non-volatile: data retained when program is not running or computer is turned off
- Comes in a variety of media:
 - magnetic: floppy disk, hard drive
 - Solid state drives
 - optical: CD-ROM, DVD
 - Flash drives, connected to the USB port

Input Devices

- Devices that send information to the computer from outside
- Many devices can provide input:
 - Keyboard, mouse, scanner, digital camera, microphone
 - Disk files

Software-Programs That Run on a Computer

- Categories of software:
 - System software: programs that manage the computer hardware and the programs that run on them.
 - *Examples*: operating systems, utility programs, software development tools
 - Application software: programs that provide services to the user.
 - *Examples* : word processing, games, programs to solve specific problems

Programs and Programming Languages

Programming Languages

- A program is a set of instructions a computer follows in order to perform a task.
- A programming language is a special language used to write computer programs.
- A computer program is a set of instructions that enable the computer to solve a problem or perform a task.
- Collectively, these instructions form an *algorithm*

Programming Languages

- An algorithm is a set of well defined steps to completing a task.
- The steps in an algorithm are performed sequentially.
- A computer needs the algorithm to be written in *machine language*.
- Machine language is written using *binary numbers*.
- The binary numbering system (base 2) only has two digits (0 and 1).

Programming Languages

- The binary numbers are encoded as a machine language.
- Each CPU has its own machine language.
 - Motorola 68000 series processors
 - Intel x86 series processors
 - DEC Alpha processors, etc.
- Example of a machine language instruction:
10110100000000101

Programming Languages

- In the distant past, programmers wrote programs in machine language.
- Programmers developed higher level programming languages to make things easier.
- The first of these was *assembler*.
- Assembler made things easier but was also processor dependent.

Programming Languages

- High level programming languages followed that were not processor dependent.
- Some common programming languages:

Java	C	Visual Basic
BASIC	C++	Python
COBOL	C#	Ruby
Pascal	PHP	JavaScript

Programming Languages

Common Language Elements

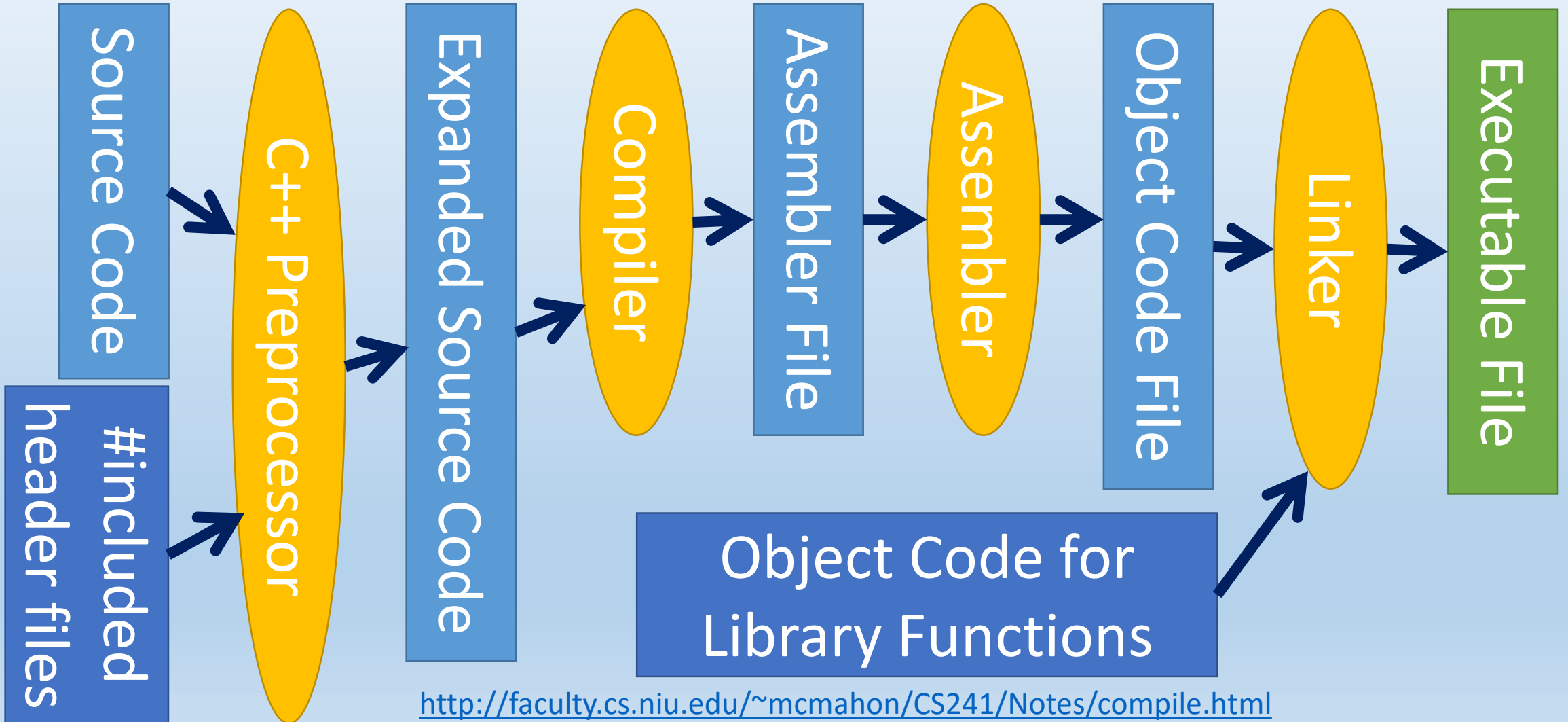
- There are some concepts that are common to virtually all programming languages.
- Common concepts:
 - Key words
 - Operators
 - Punctuation
 - Programmer-defined identifiers
 - Strict syntactic rules.

Compilation Process

- Starting with source code (e.g. C++) and converting it into machine code that the computer can run.
- When using our IDE, the process appears like this:



Actual Process



The Parts of a C++ Program

```
// sample C++ program
#include <iostream>
using namespace std;
int main()
{
    cout << "Hello, there!";
    return 0;
}
```

The diagram illustrates the components of a C++ program using a sample code snippet. Orange arrows point from descriptive labels to specific parts of the code:

- comment**: Points to the line `// sample C++ program`.
- preprocessor directive**: Points to the line `#include <iostream>`.
- which namespace to use**: Points to the line `using namespace std;`.
- beginning of function named `main`**: Points to the line `int main()`.
- beginning of block for `main`**: Points to the opening curly brace `{`.
- output statement**: Points to the line `cout << "Hello, there!";`.
- string literal**: Points to the text `"Hello, there!"` within the output statement.
- Send 0 to operating system**: Points to the line `return 0;`.
- end of block for `main`**: Points to the closing curly brace `}`.

Special Characters

Character	Name	Meaning
//	Double slash	Beginning of a comment
#	Pound sign	Beginning of preprocessor directive
< >	Open/close brackets	Enclose filename in #include
()	Open/close parentheses	Used when naming a function
{ }	Open/close brace	Encloses a group of statements
" "	Open/close quotation marks	Encloses string of characters
;	Semicolon	End of a programming statement

The `cout` Object

- Displays output on the computer screen
- You use the stream insertion operator `<<` to send output to **`cout`**:

```
cout << "Howdy! " ;
```


The `cout` Object

- Can be used to send more than one item to **`cout`**:

```
cout << "Howdy " << "Ags! " ;
```

Or:

```
cout << "Howdy " ;  
cout << "Ags! " ;
```


The `endl` Manipulator

- You can use the `endl` manipulator to start a new line of output. This will produce two lines of output:

```
cout << "Programming is" << endl;  
cout << "fun!";
```

The `endl` Manipulator


```
cout << "Programming is" << endl;  
cout << "fun!";
```

```
[tmcguire]@linux2 ~/csce121> (14:11:27 01/16/20)  
:: ./fun  
Programming is  
fun!  
[tmcguire]@linux2 ~/csce121> (14:11:56 01/16/20)  
:: 
```

The `\n` Escape Sequence

- You can also use the `\n` escape sequence to start a new line of output. This will produce two lines of output:


```
cout << "Programming is\n";  
cout << "fun!";
```



Notice that the `\n` is INSIDE
the string.

The `\n` Escape Sequence

```
cout << "Programming is\n";  
cout << "fun!";
```

```
[tmcguire]@linux2 ~/csce121> (14:11:27 01/16/20)  
:: ./fun  
Programming is  
fun!  
[tmcguire]@linux2 ~/csce121> (14:11:56 01/16/20)  
:: 
```

The `#include` Directive

- Inserts the contents of another file into the program
- This is a preprocessor directive, not part of C++ language
- `#include` lines not seen by compiler
- Do not place a semicolon at end of `#include` line

The END