# Week 1: Introduction to C++

## CSCI 2100 Data Structures

Fall 2024

**Tae-Hyuk (Ted) Ahn**

Department of Computer Science
Program of Bioinformatics and Computational Biology
Saint Louis University

SAINT LOUIS
UNIVERSITY
EST. 1818

# Overview

- Course resources and syllabus
- Expected workload
- Grade structure and policy
- Introduction to Data Structure: Motivation, Definition of Data Structure, ADT, etc.
- Introduction to C++: History, Motivation, Comparison between Python and C++, Data Types, Control Structures, Input/Output, Functions, etc.

# Course Resources

Instructor

- Tae-Hyuk (Ted) Ahn, Ph.D.
  - Associate Professor
  - Department of Computer Science
  - Program of Bioinformatics and Computational Biology
  - Email: taehyuk.ahn@slu.edu
- Office: ISE 234B
- Office Hours: Tuesday 12:30-1:30 pm, Wednesday 1-2 pm, or by email appointment.

# Course Resources

## Mandatory Textbook

- We will be using zyBooks for Data Structures (available on ZyBooks.com.)
  - A zyBook is web-native interactive content that helps students learn challenging topics.
  - zyBooks use less text, and more question sets, animations, interactive tools, and embedded homework, so students can learn by doing, which is long known to be more effective.

- A subscription is $64. You will get the instruction how to sign-up on Monday, Aug 26.

# My Goal of this Fall

- Below is my last semester (Spring 2024) CS course evaluation.
- I want achieve good feedbacks from you.
- I will help you, but I also need your help!

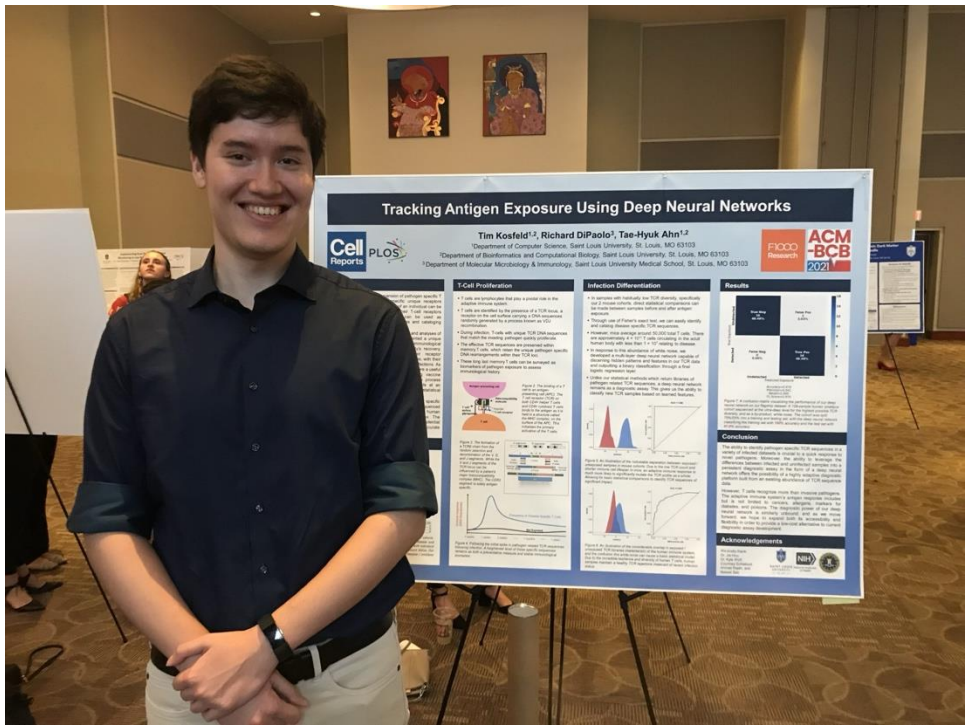
**Questions about the Course**

| Question | Your Score | | | Department (Computer Science) | | | College (Schl of Science & Engineering) | | |
|---|---|---|---|---|---|---|---|---|---|
| | Response Count | Mean | Median | Response Count | Mean | Median | Response Count | Mean | Median |
| Expected learning outcomes for the course were clearly communicated. | 12 | 3.92 | 4.00 | 865 | 3.68 | 4.00 | 2946 | 3.55 | 4.00 |
| The required textbook(s)/readings supported my learning in the course. | 11 | 4.00 | 4.00 | 801 | 3.61 | 4.00 | 2599 | 3.40 | 4.00 |
| Class discussions/activities supported my learning in the course. | 12 | 3.92 | 4.00 | 856 | 3.63 | 4.00 | 2900 | 3.49 | 4.00 |
| Assignments/homework supported my learning in the course. | 12 | 4.00 | 4.00 | 860 | 3.70 | 4.00 | 2881 | 3.52 | 4.00 |
| Exams/quizzes supported my learning in the course. | 12 | 3.83 | 4.00 | 845 | 3.67 | 4.00 | 2711 | 3.48 | 4.00 |
| The multimedia resources supported my learning in the course. | 12 | 4.00 | 4.00 | 816 | 3.65 | 4.00 | 2669 | 3.51 | 4.00 |
| The course was taught at a pace that supported my learning. | 12 | 4.00 | 4.00 | 863 | 3.63 | 4.00 | 2934 | 3.43 | 4.00 |
| The course required me to apply what I learned in new ways. | 12 | 3.92 | 4.00 | 863 | 3.67 | 4.00 | 2938 | 3.55 | 4.00 |
| The course challenged me intellectually. | 12 | 3.92 | 4.00 | 865 | 3.64 | 4.00 | 2940 | 3.58 | 4.00 |
| Overall, I think this course was excellent. | 12 | 3.92 | 4.00 | 866 | 3.57 | 4.00 | 2943 | 3.31 | 3.00 |

**Questions about the Instructor Ted Ahn**

| Question | Your Score | | | Department (Computer Science) | | | College (Schl of Science & Engineering) | | |
|---|---|---|---|---|---|---|---|---|---|
| | Response Count | Mean | Median | Response Count | Mean | Median | Response Count | Mean | Median |
| The instructor communicated ideas and information clearly. | 12 | 4.00 | 4.00 | 867 | 3.71 | 4.00 | 3448 | 3.50 | 4.00 |
| The instructor demonstrated enthusiasm for the subject matter. | 12 | 3.92 | 4.00 | 867 | 3.82 | 4.00 | 3448 | 3.63 | 4.00 |
| The instructor provided feedback/critique that helped me with subsequent work in the course. | 12 | 3.83 | 4.00 | 867 | 3.69 | 4.00 | 3448 | 3.44 | 4.00 |
| The instructor treated students with respect. | 12 | 3.83 | 4.00 | 867 | 3.83 | 4.00 | 3448 | 3.69 | 4.00 |
| The instructor was available for assistance when needed. | 12 | 3.92 | 4.00 | 867 | 3.78 | 4.00 | 3448 | 3.58 | 4.00 |

# Get connected to SLU CS Faculty Members

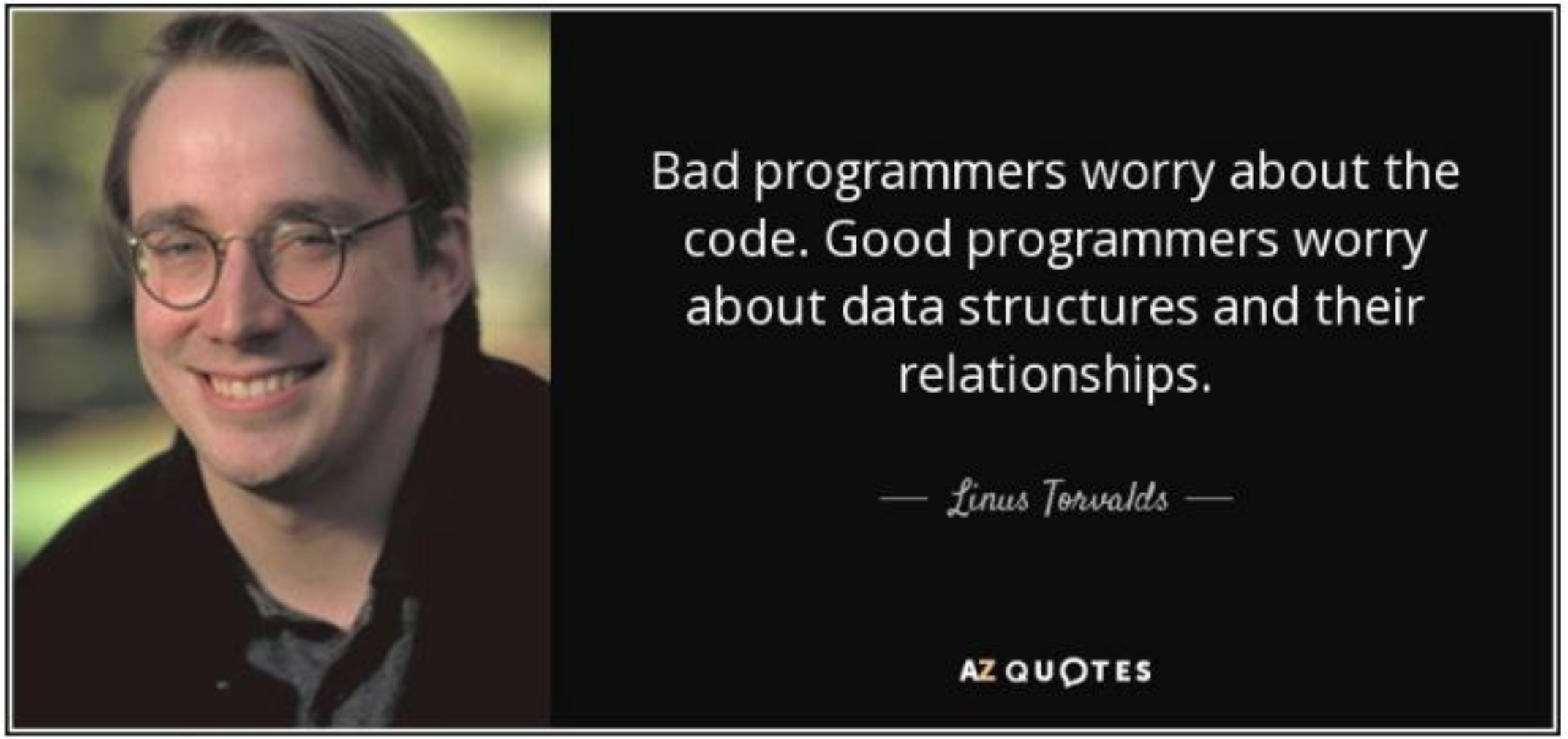



One good example

- I met Tim (CS BS) at Data Structures course
- He then worked with me.
- He then published 3 papers with me for 3 years (ABM BCB program).
- He is then studying his Ph.D. at Princeton University (CS).

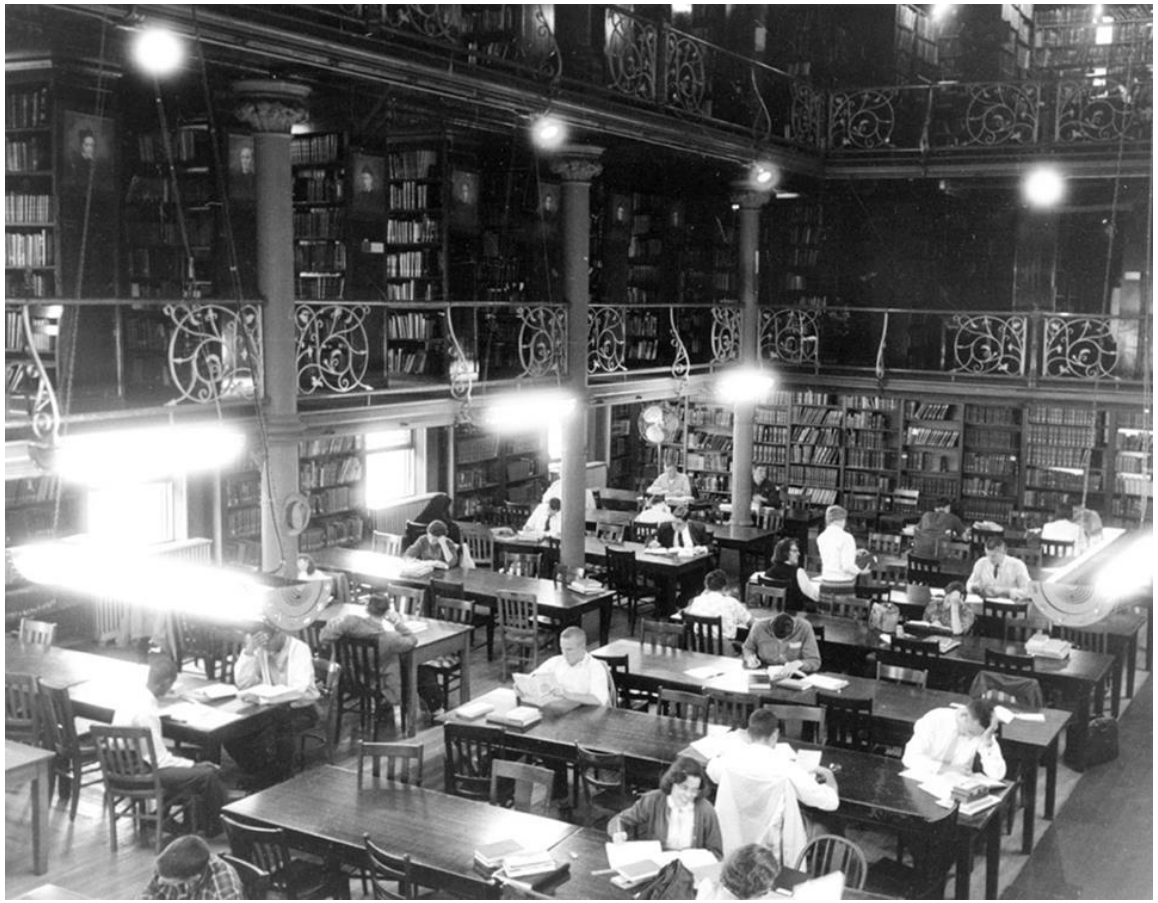
Not only for the research, “dock the door” to ask about your industry career!

# Introduction to Data Structures




https://www.azquotes.com/quotes/topics/data-structures.html

# The Library as a Data Structure



Students studying in the old DuBourg library reading room, 1958

- Imagine a library with thousands of books. To find, add, or remove a book easily, we need an effective way to organize them. This is where data structures come into play.

# Without Proper Data Structures

- If books are placed randomly on shelves:
  - Finding a specific book becomes difficult (inefficient search).
  - Adding new books leads to disorganization and chaos.

# Using Data Structures

- Using a data structure like an array or linked list, books can be ordered alphabetically by the author's last name.

- Benefits:
  - Faster search (O(log n) with binary search if sorted).
  - Systematic addition and removal of books.

# The Need for Data Structures

A data structure is a way to store and organize data in a computer, so that it can be used efficiently

- Data structures organize data
  - more efficient programs.
- More powerful computers
  - more complex applications.
- More complex applications demand more calculations.
- Complex computing tasks are unlike our everyday experience.

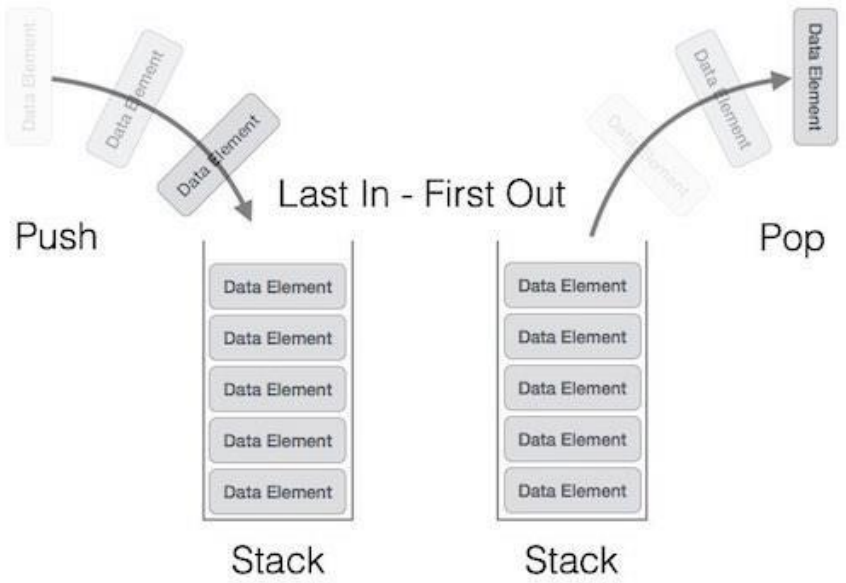
# Efficiency

- A solution is said to be efficient if it solves the problem within its resource constraints.
  - Space
  - Time
- The cost of a solution is the amount of resources that the solution consumes.

# Terminology 1: Abstract Data Types (ADT)

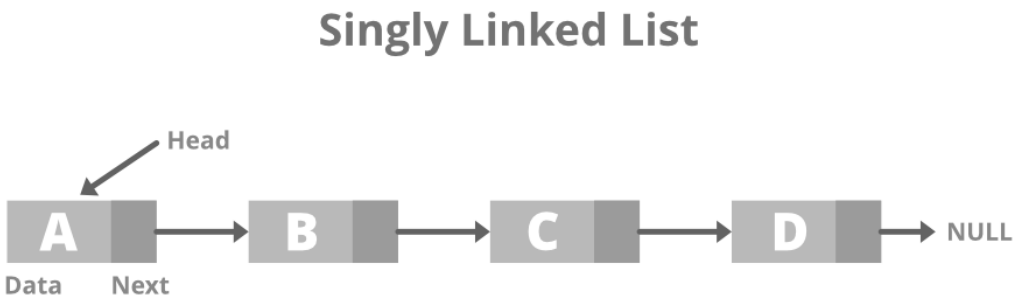
- Abstract Data Type (ADT): a definition for a data type solely in terms of a set of values and a set of operations on that data type.
- ADT is the logical picture of the data and the operations to manipulate the component elements of the data.
- Each ADT operation is defined by its inputs and outputs.
- Encapsulation: Hide implementation details.
- Example: Array, List, Map, Queue, Set,

  Stack, Table, Tree, and Vector



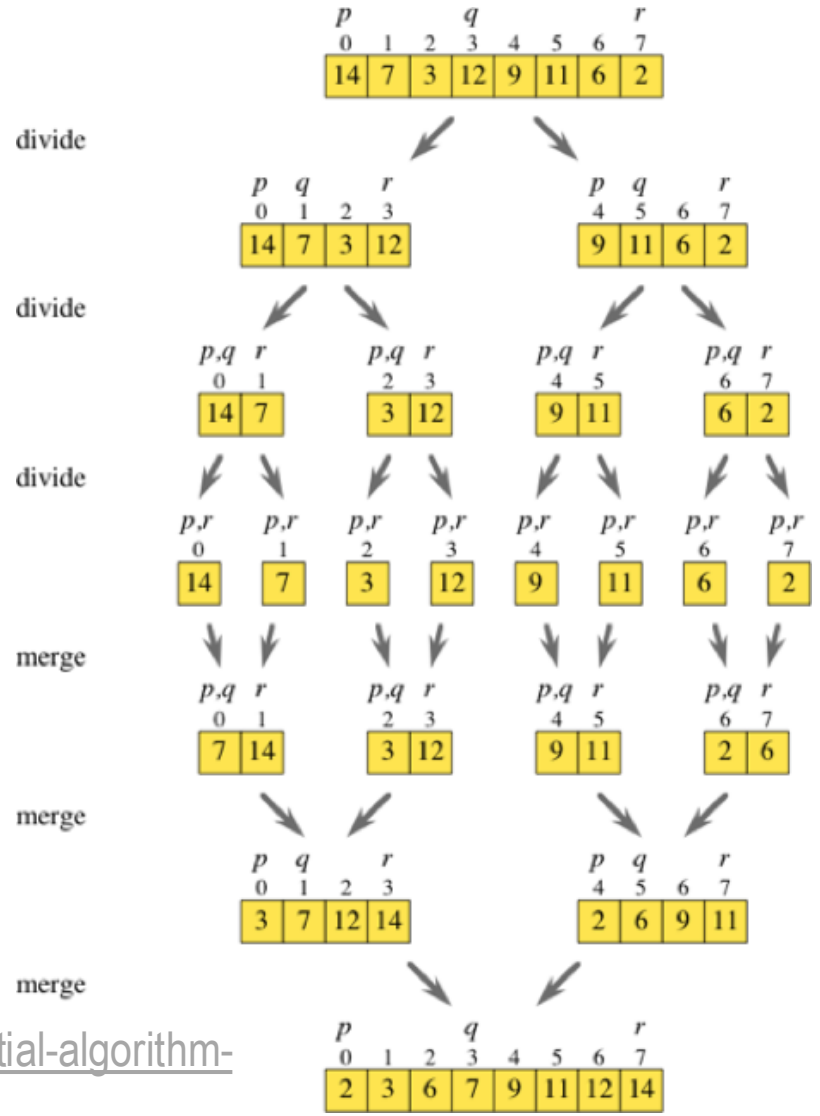
# Terminology 2: Data Structures

- Data structure is the actual representation of the data during the implementation and the algorithms to manipulate the data elements.
- ADT is in the logical level and data structure is in the implementation level.
- Example: C++ list, vector, and so on.
- ADT is implementation independent. For example, it only describes what a data type List consists (data) and what are the operations it can perform, but it has no information about how the List is actually implemented. Whereas data structure is implementation dependent, as in the same example, it is about how the List implemented ie., using array or linked list.
- Ultimately, data structure is how we implement the data in an abstract data type.



# Terminology 3: Algorithms

- In mathematics and computer science, an algorithm is a set of instructions, typically to solve a class of problems or perform a computation.
- Algorithms are unambiguous specifications for performing calculation, data processing, automated reasoning, and other tasks.
- Ex: Merge sort uses a similar “divide and conquer” methodology to efficiently sort arrays. See the following steps for how merge sort is implemented.




https://www.codementor.io/learn-programming/3-essential-algorithm-examples-you-should-know

# C++

- In 1979, Bjarne Stroustrup, a Danish computer scientist, began work on "C with Classes", the predecessor to C++
- In 1983, "C with Classes" was renamed to "C++" (++ being the increment operator in C), adding new features
- It has imperative, object-oriented and generic programming features, while also providing facilities for low-level memory manipulation.

https://www.stroustrup.com/

```
#include <iostream>

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

# Python vs C++

**Python**

- High level
  - Readable
- Dynamically typed
- Interactive
- Industrial strength for rapid development

**C++**

- Low level
  - Close to machine code
- Statically typed
- Compiled
- Useful for performance intensive tasks

# Why learn C++?

- Faster - Python is about 10 to 100 times slower as compared to C++
- Control - C++ requires much more attention to bookkeeping and storage details, and while it allows you very fine control (it's often just not necessary though)
- You get better multithreading support which is somewhat limited in Python
- Can target just about every known platform including embedded systems
- Good way to learn low-level programming

# Comparison

**Python**

```
1  def gcd(u, v):
2    # we will use Euclid's algorithm
3    # for computing the GCD
4    while v != 0:
5      r = u % v     # compute remainder
6      u = v
7      v = r
8    return u
9
10 if __name__ == '__main__':
11   a = int(raw_input('First value: '))
12   b = int(raw_input('Second value: '))
13   print 'gcd:', gcd(a,b)
```

**C++**

Import Library

```
1  #include <iostream>
2  using namespace std;
3
4  int gcd(int u, int v) {
5    /* We will use Euclid's algorithm
6       for computing the GCD */
7    int r;
8    while (v != 0) {
9      r = u % v;    // compute remainder
10     u = v;
11     v = r;
12   }
13   return u;
14 }
15
16 int main( ) {
17   int a, b;
18   cout << "First value: ";
19   cin >> a;
20   cout << "Second value: ";
21   cin >> b;
22   cout << "gcd: " << gcd(a,b) << endl;
23   return 0;
24 }
```

# White Space

- Returns, tabs, etc. are ignored in C++

```
int gcd(int u, int v) { int r; while (v != 0) { r = u % v; u = v; v = r; } return u; }
```

- Recall that these were very import in Python
- Here, we use ( ) and { } to make loops, Booleans, etc.

# Compiling

- In Python, you save code as gcd.py and then type “python gcd.py” to run it
- In C++:
  - Save as gcd.cpp
  - Type “`g++ -o gcd gcd.cpp`”
  - Type “`./gcd`”
  - You can also compile as a simple as “`g++ gcd.cpp`”, then it saves executable as a.out. Then type “`./a.out`”.
  - Filename extensions for C++ are (a) .cc, (b) .cpp, (c) .cxx, (d) .c++, (e) .h, (f) .hh, (g) .hpp, (h) .hxx, and (i) .h++.