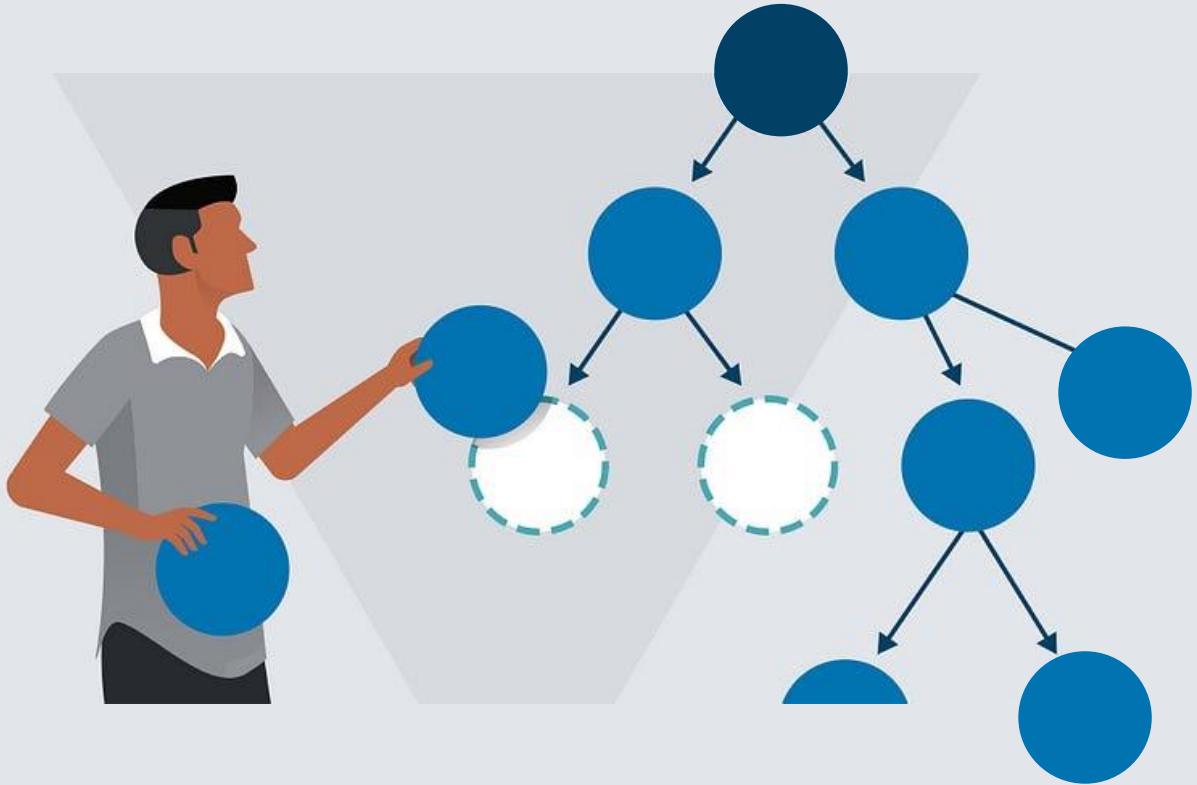
ALGORITHM ADVANCED

C1-S1 – COURSE INTRODUCTION



CADT
IDT

KICK OFF OBJECTIVES

Course Kick off

- Overview of what you will be able to do at the end of this course
- Overview of course map, learning objectives, evaluation and agenda
- Rules about the usage of AI within this course and plagiarism
- What we expect in terms of work and attitudes
- Overview of the tools to work efficiently

A quick start

- A short introduction to C++
- Challenge yourself with a few problems!

Feedbacks

- How this course can serve your future career?
- What kind of involvement do you like to commit during this course?

References Documents

About lecturer

BOU Channa
bouchanna.itc@gmail.com
Telegram: 081976875

- **Bachelor of Engineer**
  - Computer Science, ITC (2011-2016)
- **Master of Science**
  - Software Engineering, SIIT, Thammasat University Thailand (2016-2018)

- **Experiences**:
  - 2017-2018 : Teaching assistant in Python Lab, SIIT Thailand
  - 2018-2020 : Part-time lecturer at WU
  - 2018-Present : Part-time lecturer at CADT (NIPTICT)
  - 2018-Present : Full-time lecturer at GIC, ITC
  - 2020-Present : Technical lead at eLearning center, ITC

What you should be able to do

at the end of this course

- Understand C++ syntax and object-oriented approach.
- Explain the concepts of complexity through various real-world problems.
- Use different ADTs and solve associated problem types

  -Aggregated structures: **Array**, **Classes**

  -Dynamic structures: Linked Lists, Stack, Graphs, etc.

COURSE MODULES

3 **ABSTRACT DATA STRUCTURES**

- Abstract Data Types
- Dynamic Tables
- Trees
- Linked Lists
- Graphs

2 **ANALYSIS OF ALGORITHMS**

- Algorithm Complexity
- Sorting Algorithms

1 **C++ BASICS**

- From C to C++
- Dynamic Memory
- Classes & Objects

COURSE OBJECTIVES

KNOW

- ✓ Understand **C++ syntax** and **object-oriented approach**
- ✓ Understand the concepts of **algorithm complexity**
- ✓ Understand the functioning of **abstract data types (ADT)**

SKILLS

- ✓ Conduct **performance analysis** for a given algorithm
- ✓ Specify an **abstract data type**
- ✓ **Justify the factors** that influence the choice of an ADT for a given problem
- ✓ **Solve a real-world problems** using an appropriate ADT

BEHAVIORS

- ✓ Cultivate **collaborative learning**
  - Give and receive feedback
  - Teamwork spirit to solve complex problems
  - Openness to reconsider and improve your solutions
- ✓ Develop an **analytical approach** to justify your technical choices
- ✓ Develop **rigor and discipline** in programming

COURSE EVALUATION

TYPE OF EVALUATION	RATIO
Attendance and class participation	10 %
Quizzes	10 %
Practice Assessments	20 %
Peer review Assessments	10 %
Project	25 %
Exam	25 %

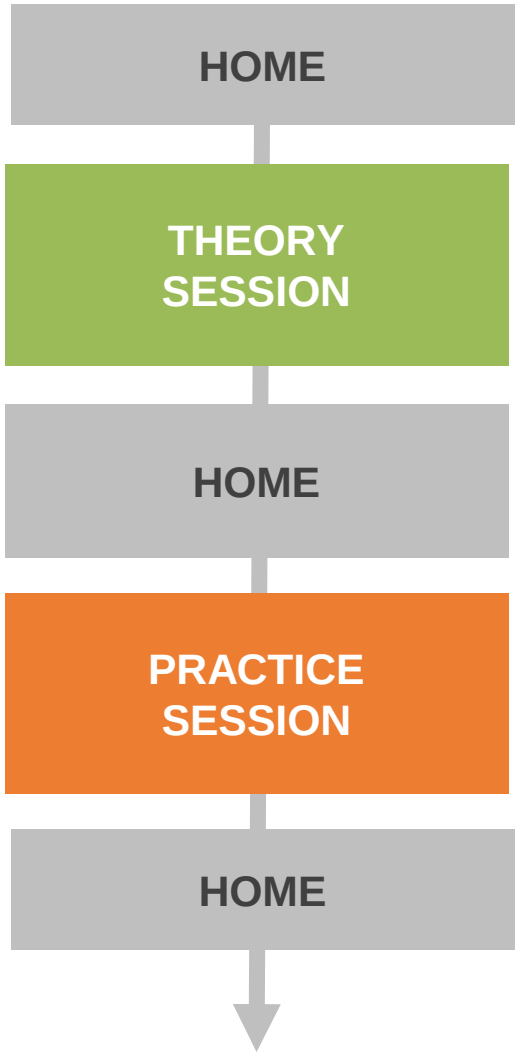
COURSE AGENDA

Please find course agenda in the course syllabus

A **week** in *Algorithm Advanced*

Our expectations in terms of work and attitudes

We recommend you to spend
6 hours per week
Of personal work, outside of class



- ✓ Perform **SELF-LEARNING assignments**
- ✓ Perform **RESEARCHES assignments**

- ✓ Cultivate **COLLABORATIVE LEARNING**
  - *Give and receive feedback*
  - *Teamwork spirit to solve complex problems*
  - *Openness to reconsider and improve your solutions*

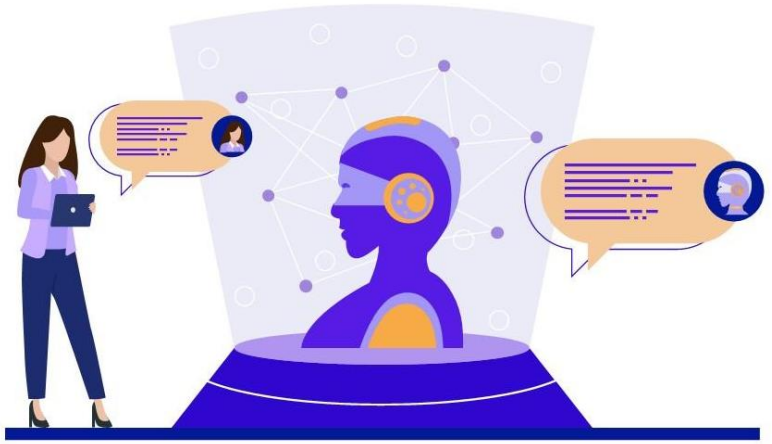
- ✓ Review **concepts**
- ✓ Write your **OWN NOTES**
- ✓ Write your **OWN CODE**

- ✓ **PARTICIPATE ACTIVELY** in group activities

- ✓ Submit the **PRACTICE BEFORE THE DEADLINE**
- ✓ Submit the **SELF-ASSESSMENT SURVEY**

Rules about the usage of AI within this course and plagiarism

- If found copy solution from AI, zero score is given.
- If found duplicate work, zero score are givens to all of those.



It does not help you if you always use AI to help you generate the whole solution.

However, it is not prohibited to use AI. You can get help sometimes to figure out some idea or tips to solve some parts of your problem

Take notes with **MarkDown**

Markdown (MD) is an easy-to-use language to keep your notes organized !

```
# React components

- React components are JavaScript functions that return markup:

```js
function MyButton() {
  return <button>I'm a button</button>;
}
```

- We can use components in other components:

```js
export default function MyApp() {
  return (
    <div>
      <h1>Welcome to my app</h1>
      <MyButton />              <---- HERE !
    </div>
  );
}
```
```

React components

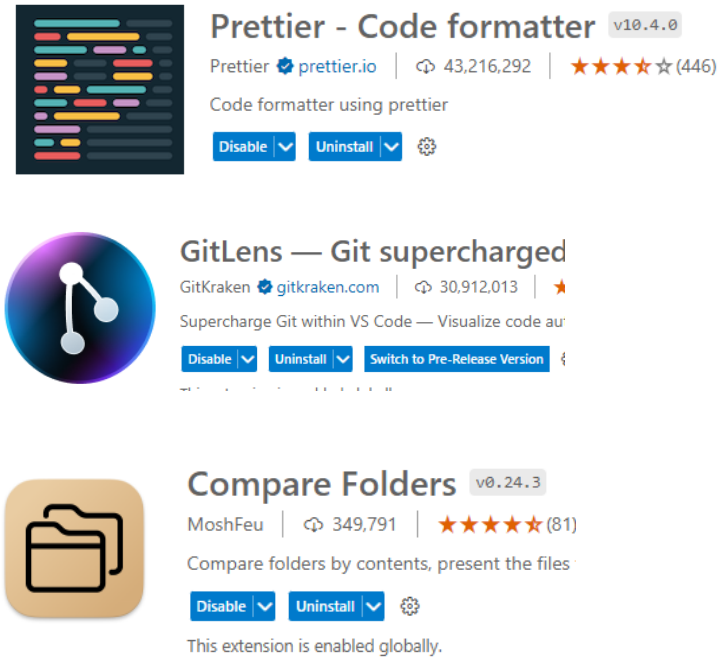
- React components are JavaScript functions that return markup:

```
function MyButton() {
  return <button>I'm a button</button>;
}
```

- We can use components in other components:

```
export default function MyApp() {
  return (
    <div>
      <h1>Welcome to my app</h1>
      <MyButton />              <---- HERE !
    </div>
  );
}
```

VS Code for C++



You can use any tools that you prefer anyway!

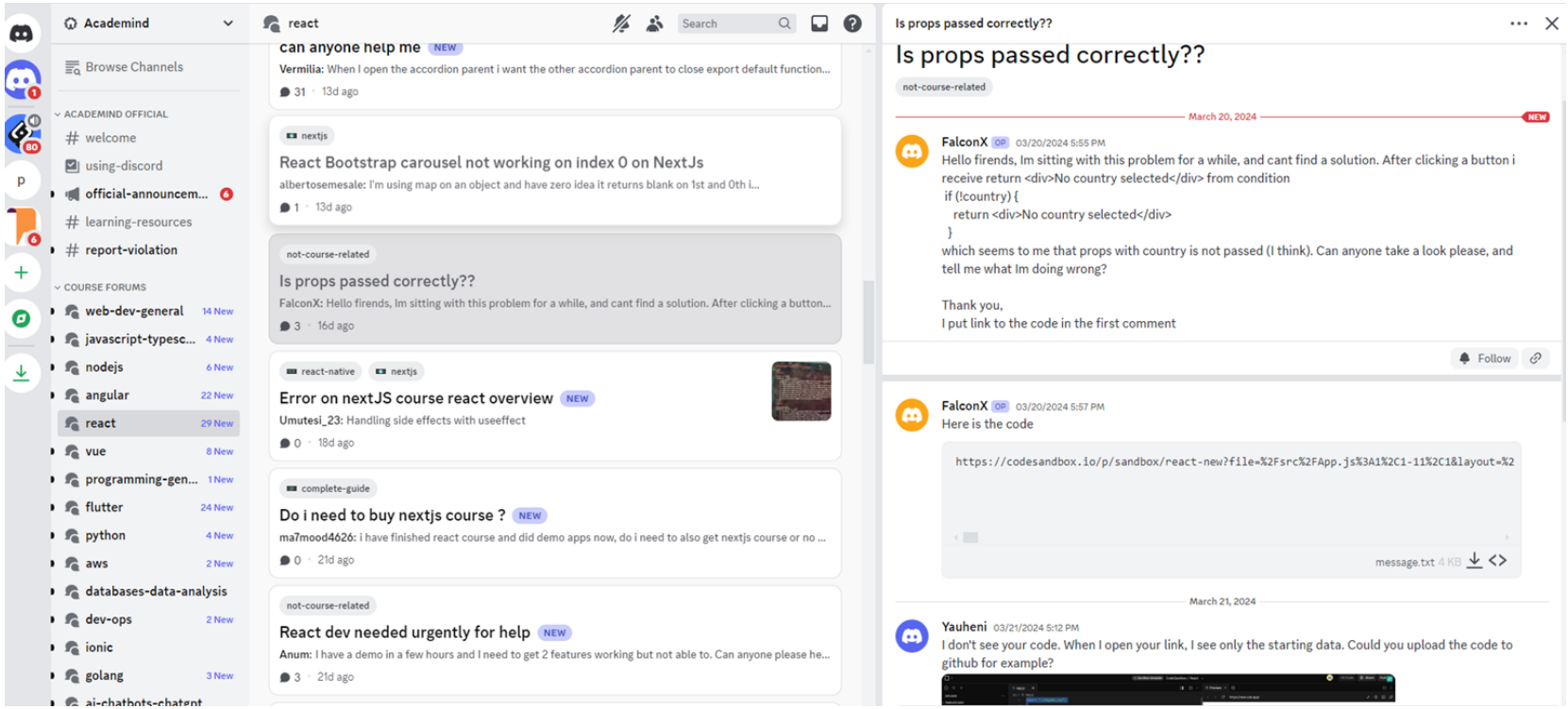
For examples:

-Atom editor

-Notepad++

-CodeBlocks

Usage of **Telegram** channels



Other tools

Online editors

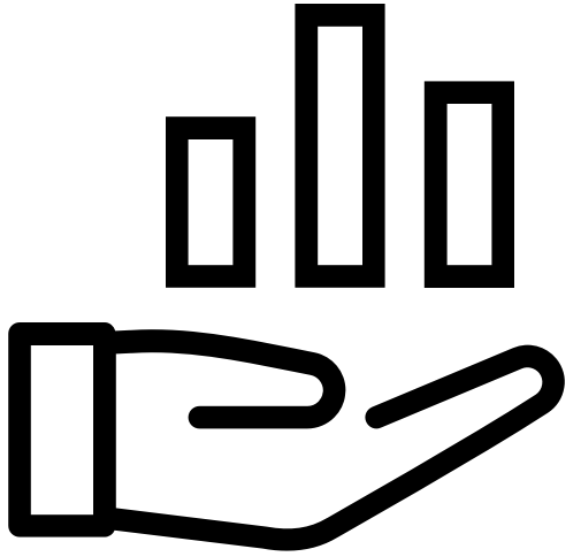
Hacker ranks

Git

Etc..

A short introduction to C++

Introduction to course



Introduction

❑ What is Data Structure?

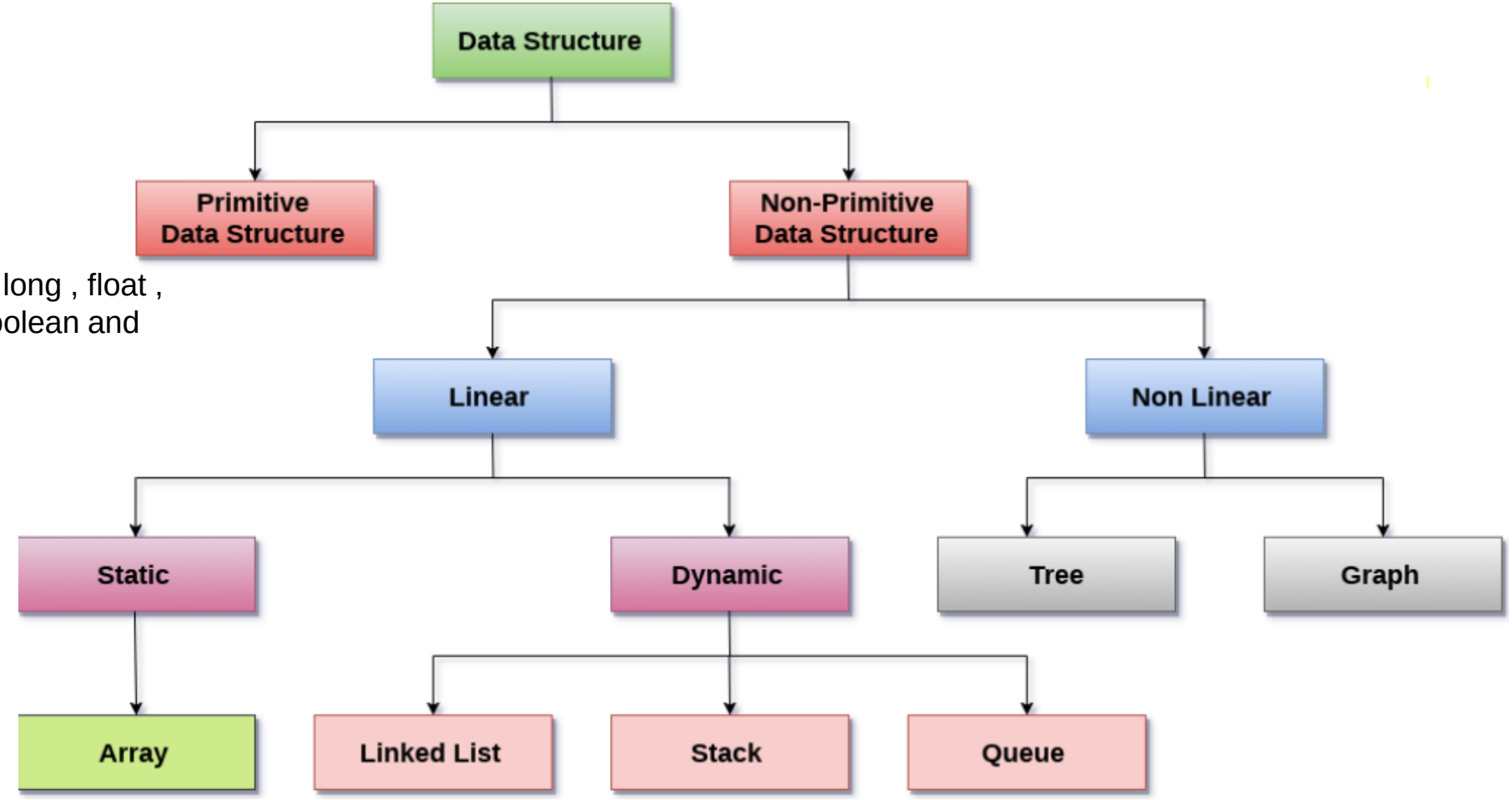
- **Data structure** is a way of storing and organizing information in a computer's memory.

=> The info can then be retrieved and used effectively

- Structure means a set of rules that holds the data together

Introduction

❑ Data Structure: An Overview



Introduction

❑ Efficient Solution

- A solution is said to be *efficient* if it solves the problem within its *resource constrains*
  1. Memory Space
  2. Execution Time

- The cost of a solution is the amount of resources that the solution consumes

C++ Introduction

Introduction

- C Vs. C++ Language

C Language

- Extension: .c
- For output/input
  - **printf**, **scanf** with placeholder
- Variable type:
  - `int, float, char,`
  - `char [ ], bool`

C++ Language

- Extension: .cpp
- For output/input
  - **cout**, **cin** without placeholder
- Variable type:
  - `int, float, char,`
  - `char [ ], bool, string`

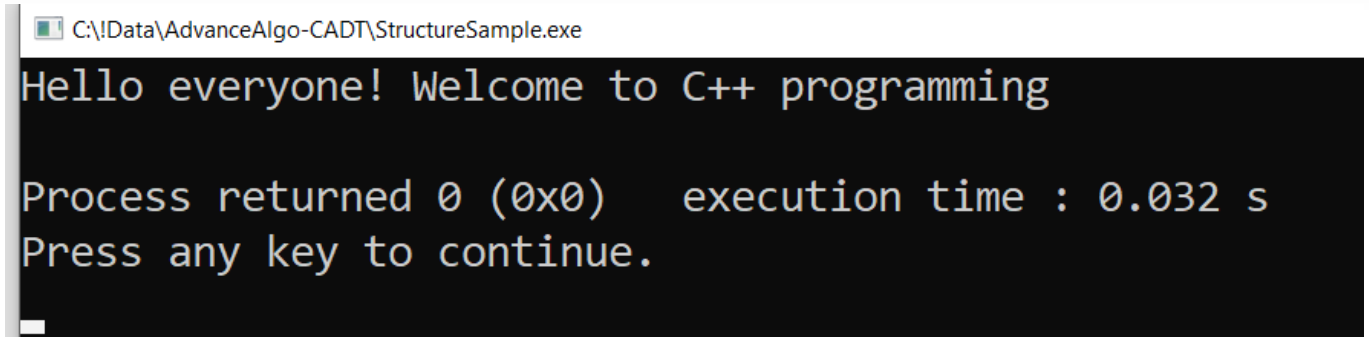
More on C and C++

C	C++
Developed by Dennis Ritchie between 1969 & 1973	Developed by Bjarne Stroustrup in 1979
C is a subset of C++	C++ is a superset of C.
C can not run C++ code.	C++ can run most of C code
Support procedural programming paradigm for code development	Support both procedural and object oriented programming. C++ is also called a hybrid language.

Let’s write your first program in C++
Test.cpp

```cpp
#include<iostream>
using namespace std;

main(){
    cout<<"Hello everyone! Welcome to C++ programming\n";
}
```



Example

A simple program to ask a user for a name and display a greeting message!

❑ Sample code: C Vs. C++

Test.c

```c
#include<stdio.h>
main(){
    char name[20];
    printf("Hello world!\n");
    printf("What is your name?: ");
    scanf("%s", &name);
    printf("Hi, %s!", name);
    printf("Welcome to Advanced Algorithm and Data structure class!\n");
}
```

C

```
Hello world!
What is your name?: Jack
Hi, Jack!Welcome to Advanced Algorithm and Data structure class!
```

Test2.cpp

```cpp
#include<iostream>
using namespace std;
main(){
    string name;

    cout<<"Hello world!\n";
    cout<<"What is your name?: ";
    cin>>name;
    cout<<"Hi, "<<name<<"!";
    cout<<"Welcome to Advanced Algorithm and Data structure class!\n";
}
```

C++

Syntax

❑ Most C and C++ syntax are the same

- Decision making
  - `if, else if, else`
- Loop
  - `for, while, do while`
- Function

- **Structure**
  - Just in C++, after the creation of structure, we dont need to use struct keyword again for creating variable
    - `struct Student{ ... };`
    - `Student st;`

Challenge yourself with a few problems!

Code Practice and Q&A if any

- Write C++ programs to …

1. **Find root** of quadratic equation $\mathbf{ax^2+bx+c=0}$. You need to study the case delta is 0, delta is greater than 0, and delta is less than 0.

2. **Display numbers 1 to 1000** on the screen except the numbers 100, 200, 300, 400 and 500.

3. Ask a user to input a number. **Keep asking the user** for more numbers until the user inputs -1. Display the total summation of all input numbers except -1.

4.Write a **function to display** and compute this suit 1/1 + 1/2 + … + 1/n , where n is the parameter of this function.

Code Practice and Q&A if any

5. Perform some mathematic operations below (also make a menu for your program so that users can test any functions. Run it as infinite loop).
   a. A summation function to calculate the sum of first n integer 1+2+3+...+n

      int sumSuite(int n)

   b. Sum digits of a number (E.g: Let n=152, then sum digits = 1+5+2 = 8)

      int sumDigit(int n)

6. Create a structure of person. This structure contains some information such as name, age, zodiac sign (string), of a person. Create an array that can stores person information up to 20 people. Then …
   - Ask a user to input information for 4 people (your crushes ❤, or your best friends ☺) and store in your array
   - Display information for each person on screen.
   - Show information of the person who has the oldest age.
7. Get two integer numbers (x and y) from a user. Create 2 point variables, say p1 and p2. Let p1 stores the address of x and p2 stores the address of y.
   a. Display address and value of x using p1.
   b. Display address and value of y using p2.
8. Create a function that can exchange two numbers. This function takes 2 parameters of the type pointers.

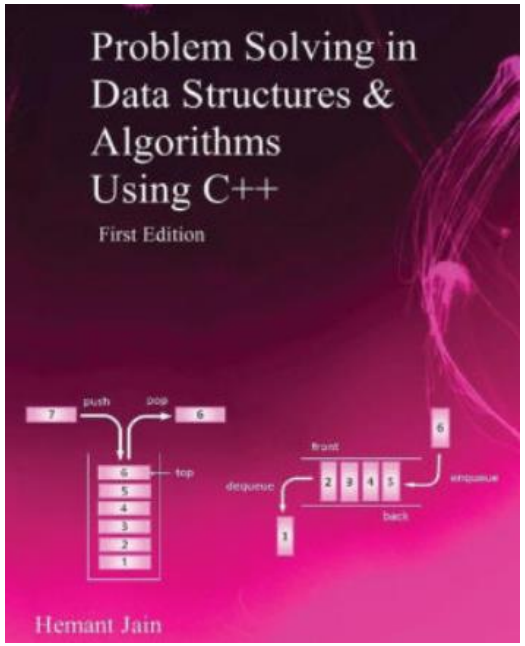
   void **exchangeNumber**(int *m, int *n)

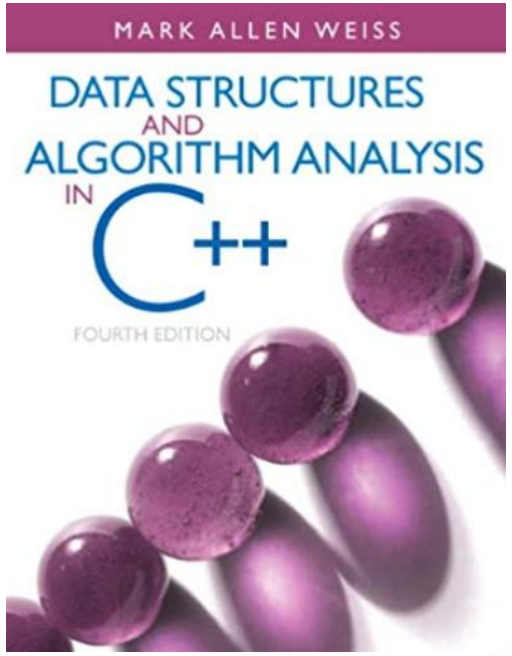
Q&A

RESSOURCES FOR THIS COURSE

1. Hemant Jain, *Problem Solving in Data Structures and Algorithms Using C++* (2017)

2. Mark Allen Weiss, *Data Structures and Algorithm Analysis in C++*, 4th edition (2014)





You can also find some useful video tutorials on advanced algorithm on my YouTube channels: **bou channa**