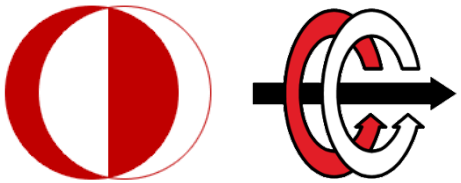
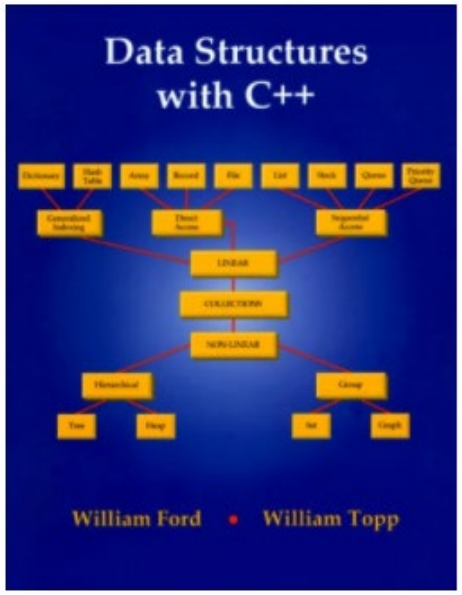


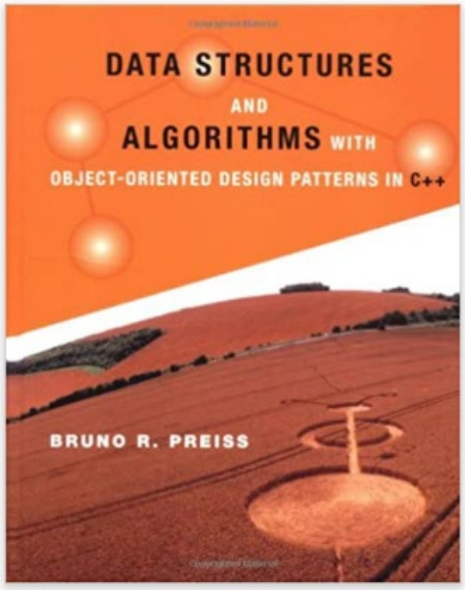
EE 441 Data Structures

Text Books

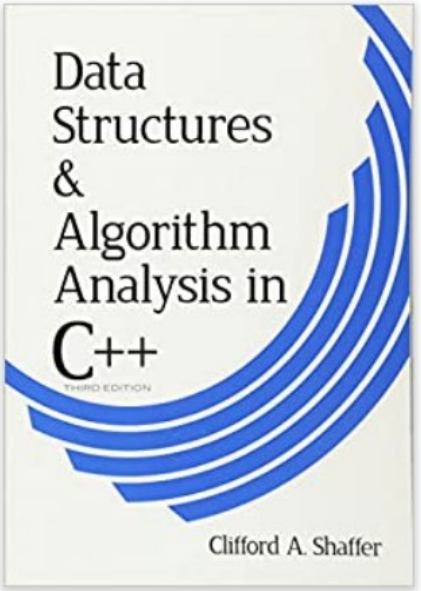
Ford &Topp, Data Structures with C++, Prentice-Hall, 1996.



Preiss, B.R., Data Structures and Algorithms with Object-Oriented Design Patterns in C++, Wiley, 1999.



Shaffer, C., Data Structures & Algorithm Analysis in C++, Dover Publications, 2012



Grading

- Programming Assignments: 20%
  - 1 day late submission: HW will be evaluated out of 90.
  - 2 days late submission: HW will be evaluated out of 75.
  - 3 days late submission: HW will be evaluated out of 55
  - 4 or more days late submission: HW will not be evaluated
- 4 Quizzes: 20%
- Midterm Exam: 25%
- Final exam: 35%
- Attendance Bonus: 5%
- Students who miss all the quizzes or who do not submit any homework will be graded as NA (“Not Available”)
- Make-ups are to be given to those having medical report approved by METU medical center

Course Objective (Why should you take this course?)

- You will work with software and hardware systems
- Software specific gains
  - How to organize, represent and access data
  - How to design algorithms
- Useful for both hardware and software
  - Modular system design and abstraction
  - Interfaces between modules
  - Complexity and design trade offs (space, time, cost)

Data Structures

- A systematic way of organizing and accessing data so that it can be used efficiently
  - Examples: queue, stack, linked list, tree
- Associated algorithms to perform operations that maintain the properties of the data structure
  - Examples: search, insert, balance,
- Properties of a well-designed data structure:
  A variety of critical operations can be performed using as little resources (both execution time and memory space) as possible

Different ways of programming: Unstructured Programming

- One main program
- Data is global throughout the whole program
- Simple for small projects
- Problems
  - If the same statement sequence is needed at different locations within the program, the sequence must be copied
  - Disadvantageous (messy) once the program gets sufficiently large

Example

- Unstructured

```
main()
{
    int a,b,temp,result;
    a=5;
    b=6;
    temp=(a+b)/2;
    result=temp*temp;
    int c=8;
    int d=10;
    temp=(c+d)/2;
    result=temp*temp;
};
```

Different ways of programming: Procedural Programming

- Programs are divided into pieces which can be combined later
- These pieces are written by programmers
- Other users construct their own programs using these pieces
- Abstraction: separates what the user needs to know and the programmer needs to know
  - Users can think in high-level terms
  - Users don’t need low-level details about the piece implementations

Example

- Unstructured

```
main()
{
    int a,b,temp,result;
    a=5;
    b=6;
    temp=(a+b)/2;
    result=temp*temp;
    int c=8;
    int d=10;
    temp=(c+d)/2;
    result=temp*temp;
};
```

- Procedural

```
int avgsq(int x, int y){
    int t=(x+y)/2;
    return t*t;
}

main(){
    int a,b,result;
    a=5;
    b=6;
    result=avgsq(a,b);
    int c=8;
    int d=10;
    result=avgsq(c,d)
};
```

Different ways of programming: Procedure-oriented Programming (POP)

- POP: procedural abstractions
  - Ignore the implementation of the procedure
  - Focus on arguments and return values

```
int avgsq(int x, int y);

main();
{
    int a,b,result;
    a=5;
    b=6;
    result=avgsq(a,b);
    int c=8;
    int d=10;
    result=avgsq(c,d);
};
```

Different ways of programming: Object-oriented Programming (OOP)

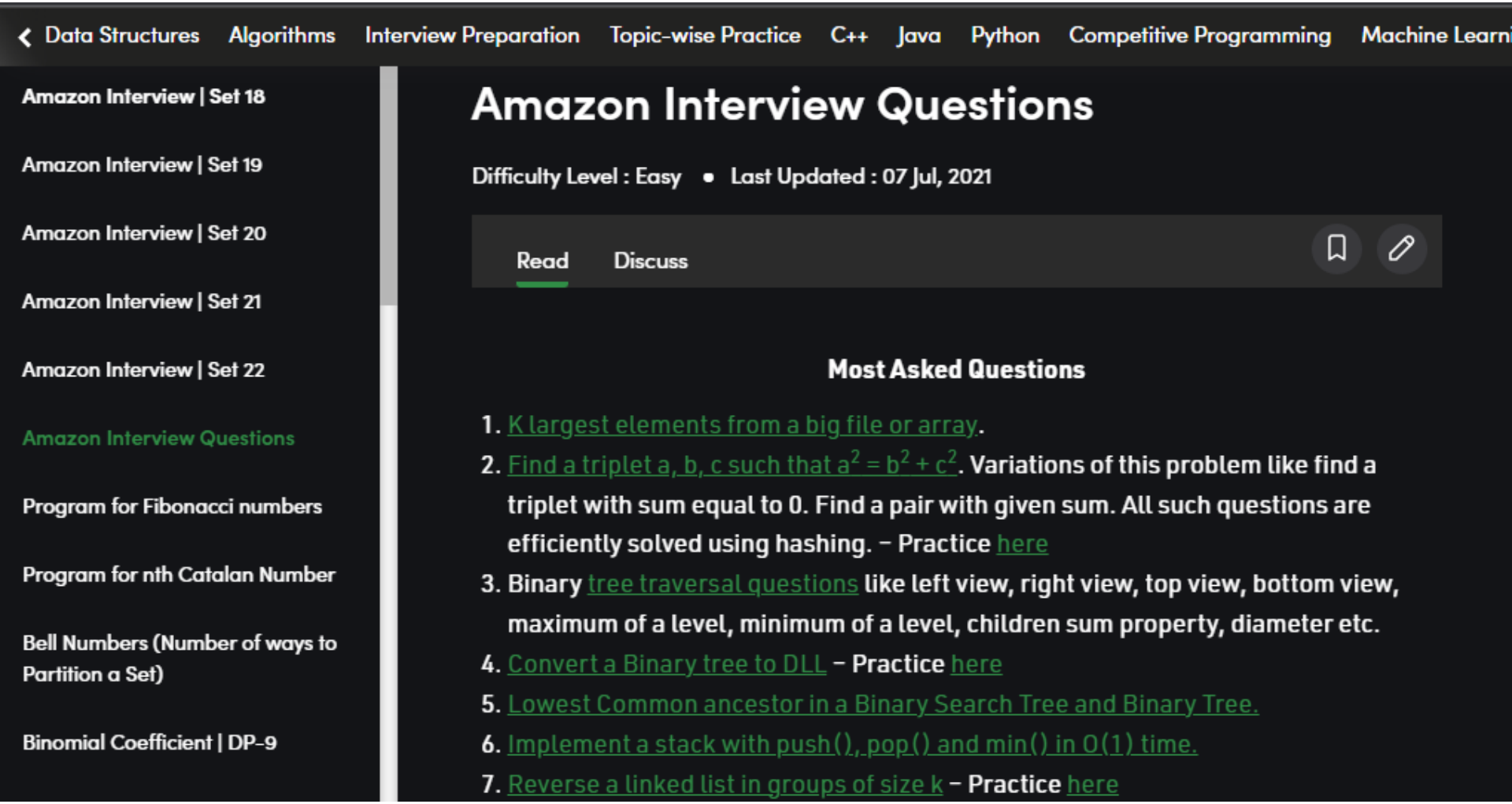
- OOP: procedural abstractions, data abstractions and encapsulation
  - Ignore the way data is represented in memory
  - Focus on operations that can be performed on data
  - Encapsulation aids the software designer by enforcing *information hiding*
  - The implementation details are hidden from the user of that object

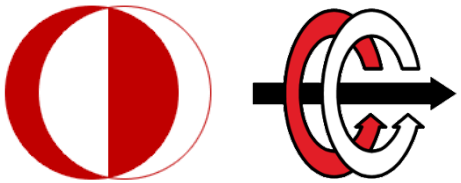
$\Rightarrow$ The stated concepts are the main focus of this course

Course Outline

- Basic C++ syntax
- Arrays, pointers
- Functions and recursion
- Algorithm complexity
- Introduction to object-oriented programming (OOP)
- Abstract data types, classes & objects
- Stacks
- Queues
- Dynamic memory management
- Linked lists
- Trees, Binary trees (B-Trees)
- Graphs
- Sorting and hashing algorithms
- Problem complexity

What you learn is useful ☺





EE 441 Data Structures